顾炯炯 编著

云计算架构技术与实践

（第2版）

CLOUD COMPUTING
ARCHITECTURE TECHNOLOGIES & PRACTICE,
SECOND EDITION

清华大学出版社

顾炯炯 编著

云计算架构技术与实践

（第2版）

清华大学出版社
北 京

内 容 简 介

云计算概念诞生至今约10年的时间，这10年来，相比云计算诞生初期，技术条件、行业和市场环境均发生了巨大变化，广大读者对云计算的认知需求，也从当初的粗浅概念阶段，发展到希望深度探索的阶段。

本书以云计算架构技术为核心，从讨论云计算发展为起点，围绕云计算架构涉及的核心技术与商业实践展开。论及的核心技术包括计算、存储、网络、数据、管理、接入、安全等方面，涵盖了云计算的最新趋势、原理、特性与实践。

本书针对希望了解云计算技术最新进展的读者和希望深入探索云计算架构技术的读者编写，适用于企业IT部门首席信息官(CIO)、IT主管、技术类人员、IT技术公司、互联网公司、教育机构的师生、IT技术工程师等。

图书在版编目(CIP)数据

云计算架构技术与实践 / 顾炯炯 编著. —2版. —北京：清华大学出版社，2016
ISBN 978-7-302-44877-8

Ⅰ. ①云… Ⅱ. ①顾… Ⅲ. ①云计算—架构 Ⅳ. ①TP393.027

中国版本图书馆 CIP 数据核字(2016) 第 194400 号

责任编辑：陈 莉 高 岫
封面设计：周晓亮
版式设计：方加青
责任校对：曹 阳
责任印制：宋 林

出版发行：清华大学出版社
网 址：http://www.tup.com.cn，http://www.wqbook.com
地 址：北京清华大学学研大厦 A 座 **邮 编：**100084
社 总 机：010-62770175 **邮 购：**010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：北京亿浓世纪彩色印刷有限公司
经 销：全国新华书店
开 本：185mm×240mm **印 张：**21.75 **字 数：**597 千字
版 次：2014 年 9 月第 1 版 2016 年 9 月第 2 版 **印 次：**2016 年 9 月第 1 次印刷
印 数：1 ～ 3500
定 价：68.00 元

产品编号：070544-01

编委会

第一作者：顾炯炯

第2版作者团队(排名不分先后)

李 明 章 宇 李金成 罗 浩 刘 阳 王道辉 申 思 李 力 张 森 申 骞 郝玖锋
刘洪宇 李 嘉 熊文辉 夏泉源 陈 嵘 王 昳 金 波 陈 普 鲍 亮 吕鹏啸 邹 睿
闫启明 陈水星 王 政 孙万琪 于 强 尚 岩 尹 青 线超博 高 原 张 妮 王 磊
王 华 丁海洋 李泽帆 黄 毽 苗永强 梁殿鹏 闵小勇 王定军 刘合金 范家超 陈永跃
刘 霞 肖延华 吴亚丽 薛 莹 尚林涛 张大震等

第1版作者团队(排名不分先后)

黄朝意 金 波 李 力 吴天议 王道辉 闵小勇 熊文辉 李 浩 严天科 钟 颙 吴鸿钟
琚列丹 胡斐然 皮楚贤 陈 普 李 嘉 朱照生 周建军 刘红霞 肖江波 谢 宁 孙万琪
潘少钦 胡善勇 王 静 宁志强 张大震等

第2版序言

2014年我们出版了本书第1版，试图帮助大家揭开云计算技术与架构的神秘面纱。然而两年以来，如何让云计算真正走下技术的神坛，脚踏实地地服务好全球各行各业，使其ICT生产效率提升，促使ICT产业尽快完成面向极致开放化、敏捷化与智能化的升级转型，仍旧是摆在广大云计算从业者面前共同的课题与挑战。带着这些问题，针对第1版中已覆盖的云计算技术趋势、OpenStack、软件定义网络、云安全、大数据、分布式软件定义存储、软件定义Overlay网络、云计算实践等内容，我们在第2版中均做了与时俱进的更新，分享了华为在云计算核心竞争力构建与价值转换方面的经验与建议，并补充了业界在公有云、私有云、行业云以及电信网络云化商用落地与技术应用方面的成功优秀实践。 与此同时，针对两年来云计算在前沿创新领域最新进入人们视野的新热点，诸如Docker容器与微服务敏捷迭代、大数据与数据库云化、行业建模与机器学习算法、混合云与管理自动化编排、云生态建设等，我们在第2版中也重点新增了对其技术与架构发展动态以及应用前景的探讨，希望能给大家带来更多的启发与帮助。

在此，我谨代表全体参与本书编纂写作的专家团队成员，再次衷心感谢广大读者的持续关心与支持！愿大家在云计算产业化发展之路上共勉，收获更多、走得更远！

顾炯炯

序言一

早在3000多年前，《易经》出现时，人类便用朴实的哲学思想揭示了一个变化的世界。变化无所不在，体现在政治、经济、社会、科学等方方面面。其中科学技术的发展变化，是这几个领域中被人们最能感同身受的。在科学技术领域，IT技术又成了科技变化发展的急先锋。IT领域的“云计算”在2007年还是个未知概念，到2014年“云计算”不仅已经家喻户晓，而且在基于云计算平台上，创造了一个又一个发展速度的新纪录：一款在云平台上的游戏，可以在几个月发展数千万用户；云平台支撑的电子商务，1秒钟可以完成数万笔交易。目前这种IT云计算领域的发展变化模式已经不再局限于令人瞠目的数字，而是已经开始向传统行业乃至国家整体经济与社会领域进行快速渗透。我们相信，在不久的将来，全球的经济、社会与科学的面貌，在云计算技术、服务与理念的推动下会焕然一新。这就是我们过去十几年常说的“信息技术革命”，在我们眼前正在发生、发展和不断演变着。

“云计算”这个名词虽然目前已经家喻户晓，而且业内基本认可美国国家标准与技术研究院(NIST)对云计算的概念定义，但人们对云计算的理解至今依旧不同。究其根因，一方面是云计算的技术、服务模式和理念在不断演进和发展变化，另一方面，云计算的宣传推广主体，出于商业或理念的差异而对云计算的内涵进行了不同方向的强化和引申。人们对云计算的不同理解，势必引发概念定义上的争议，但一个不争的共识是：云计算已经落地生根，并快速地发展壮大，“像用电一样使用信息服务”的云计算理想虽然还未完全实现，但距离这个目标已经越来越近。我们每个人对云计算的发展阶段同样可能有不同的理解和划分，这很正常，也很有益，因为多样化的观点碰撞是创新灵感的源泉。本书也将“云计算”的技术内涵进行了更大范围的发展和延伸。这让我联想到互联网行业正在发生的变化，早期的互联网公司仅仅开发与其直接相关的web网站，但现如今，大型互联网公司，所做的产品与业务早已远远地超出当初的范畴，不仅在看似简单的web网站下面搭建了一个庞大的数据处理与分析平台，而且支撑其网站的服务器、存储、网络乃至数据中心这些硬件产品均由互联网公司自己研发与设计，领先的互联网公司做的产品还远不止这些，Google推出了手机操作系统、眼镜，并致力于开发、完善无人驾驶汽车；亚马逊开发无人飞机用于物流投递；Facebook在虚拟现实领域进行了高额投入；国内的阿里、腾讯等公司在开发各种金融类创新产品。针对这些巨大的变化，在没有更好的名称之前，我们依然要称呼它们是互联网公司，我们只能说当今互联网的含义比10年前的互联网更加宽泛。“云计算”也是如此。在IT领域，基本上每3～5年便会进行一次产品技术的更新换代，云计算经过多年发展，无论在技术深度还是在技术广度上均会有显著延展。通过阅读此书，可以明显感受

到云计算技术这些年来的发展和进步，以及云计算技术在企业IT和电信网络重构中的作用。

云计算不仅对企业和电信的发展有重大的推动作用，对教育科研领域也将有深远的影响。“云计算”是由企业提出的概念，并一直由企业主导“云计算”技术的研发、推广与应用。特别是云计算所支撑的大数据概念与技术的出现，企业主导，特别是互联网企业主导技术发展的趋势更加明显。之所以企业能够主导云计算与大数据技术的发展，原因可能来自三个方面：一是企业拥有的雄厚资金；二是云计算技术与企业应用及市场需求呈现出了紧耦合的发展；三是企业(特别是互联网企业)拥有足量的数据资源和计算资源。企业主导，也同时意味着传统先有理论后有应用，大学科研机构负责理论(学术)创新，企业负责应用创新(工程实践)的分工模式，在云计算与大数据领域基本不再适用。大学教育科研机构，特别是信息技术学科，需要调整自身角色来顺应这一趋势的发展。云计算领域提供了非常丰富的开源环境，例如OpenStack、Hadoop、 Xen、 KVM等，互联网同时提供了数量庞大的信息技术文章，云计算厂商会提供免费或低价的试用软件或云计算环境，再加上不断出版更新的信息技术类书籍，这些资源构成了一个开放且实时更新的教学乃至实验环境，可以供广大学生学习和实践使用。未来大学对学生的培养工作可能将集中在四个方面：方向性的指导；综合技能的认证；给学生提供专心学习交流的环境；为学生搭建创新的平台。大学的信息技术类学科的科研机构，需要与企业和市场更加紧密的结合，双方共享数据，共享研发测试环境，专利交叉共享，项目收益共享。大学科研机构可侧重承担企业偏远期的云计算与大数据技术的开发，企业侧重承担近期应用技术开发，双方的界限会很模糊。在云计算时代，年龄、资历和职称都不再是科研创新的门槛，大学生可以广泛参与和承担云平台类产品应用开发，学生的自主创新成果(同时参照产品受市场欢迎的程度)将是个人能力认证的一个重要依据。

希望本书的所有读者，在了解到云计算技术的同时，都能够积极地投身到云计算产业实践中来。只有更多的人认识到云计算的价值，才能挖掘出更多云计算的价值，云计算产业才会有源源不断的动力来蓬勃发展，相信读者中的很多人都将成为云计算产业的中坚力量。

李德毅

中国工程院院士，国际欧亚科学院院士

序言二

合作创新，云以致用

云计算从概念到大规模实践，短短数年间迅猛发展。它与诸多行业深度融合，带来了颠覆性的创新，凸显出巨大应用价值和发展前景。读到华为公司云计算首席架构师顾炯炯新著《云计算架构技术与实践(第2版)》，我倍感兴奋。著作概括了华为的云计算构想，表明华为云计算布局已走在行业的前列，也预示着英特尔和华为在云计算方面的合作将更加密切。

随着计算的延伸与扩展、移动互联网的发展，中国消费者拥有了越来越多的智能设备。为了让消费者在不同时间、不同场合获得实时信息服务，需要将各种内容和应用通过云端进行整合、匹配，推向不同的终端用户，实现从云到端高效顺畅的体验。从电信运营商到服务提供商，通过云架构部署各种移动设备和个性化服务，无疑是非常经济、便捷的途径。

同时，在经济发展和行业变革中，人们所遇到的挑战也日益复杂，需要综合的解决方案来应对。例如，在城镇化进程中，利用云计算结合物联网与大数据解决方案，可推动智能交通、平安城市、智慧医疗、环境监控等项目建设，带来更好的城市运行管理和公共服务。将云计算与传统行业融合，将带来跨界创新，催生前所未有的商业模式、产业生态。云计算越来越从一种降低成本与提高效率的方式，演变成向企业和消费者提供新服务的途径。

面对在线应用、服务和数据的高速增长，数据中心规模急剧膨胀，需要对数据中心 IT 基础设施的主要组成部分——服务器、存储和网络设备进行以软件定义为导向的创新，实现自动供给的 IT 资源池。英特尔于2013年开始践行“软件定义基础设施”战略，在开放平台上由用户自行定义他们的基础设施，这将使用户在云资源池内“调兵遣将”时更加轻松，进而也让他们的新应用和新服务更快得到底层支持，实现迅速交付。

英特尔长期专注于计算的创新，我们提供从服务器到存储、网络全面的计算能力及软件优化支持，与产业伙伴共同应对各种云计算发展难题，真正实现“云以致用”。为此英特尔与产业界协同合作，基于开放架构推出了英特尔云构建计划，帮助开放数据中心联盟发掘和定义企业用户的云计算需求，并提供丰富的云计算解决方案。

携手推动云计算创新，英特尔与华为的合作是一个很好的例证。华为不仅在通信行业处于领先地位，过去几年间其云计算业务也成倍增长。英特尔与华为的合作由来已久，双方都较早受到信息通信

产业中最大的潮流——信息技术和通信技术融合的影响，合作领域不断拓展，从最初在通信领域，扩展到服务器、存储、网络领域，现在又把共同的目标放在了云计算和大数据的创新机会上。我们对双方的合作前景充满信心。

中国互联网用户众多，信息终端普及率很高，企业及消费者对于IT技术的了解和接受速度也非常快，再加上政府大力支持云计算产业，并实施宽带中国战略，这就使得中国在云计算部署规模、技术以及商业模式创新方面迅猛发展。我们期待着与国内合作伙伴深化和扩展合作领域，让更多源自中国的云计算创新成果去影响并推动全球信息技术和通信产业的发展进程！

杨叙
英特尔公司
全球副总裁兼中国区总裁

前　言

什么是云计算？美国国家标准与技术协会(NIST)对此有这样一个权威和经典的定义：“所谓云计算，就是这样一种模式，该模式允许用户通过无所不在的、便捷的、按需获得的网络接入到一个可动态配置的共享计算资源池(其中包括了网络设备、服务器、存储、应用以及业务)，并且以最小的管理代价或者业务提供者交互复杂度即可实现这些可配置计算资源的快速发放与发布。”

云计算的核心可以用五大基本特征、三种服务模式以及四类部署模式来概括。五大基本特征是：按需获得的自助服务，广泛的网络接入、资源池化、快捷的弹性伸缩以及可计量的服务。三种服务模式为：云基础设施即服务(IaaS)，云平台即服务(PaaS)，以及云软件即服务(SaaS)。四类部署模式可以划分为：专有云(私有云)、行业云、公有云，以及混合云。

从各类云服务的创建、部署以及消费角度来描述云计算的实质，意味着云计算天然要求支持面向服务的能力。现代企业通常会将其IT基础设施、业务平台以及软件即服务的对外开放作为其整体端到端企业信息架构SOA解决方案中的重要一环来执行。当然软件即服务(SaaS)作为一个流行多年的话题，其最早出现是在云计算概念出现之前，其实已经不是什么新鲜概念了。

以亚马逊2006年3月13日发布的S3服务为起点，到“云计算”概念2008年最早被Google提出，至今已有10年多的历史了，其核心理念已广为人们所传播和接受，也经历了方兴未艾的发展。当云计算还处于概念炒作的初始阶段时，云计算一度成为IT业界、媒体传播渠道，乃至所有涉及IT信息化、政府宏观规划、关系国计民生的各大垂直行业关注的焦点，云计算也因此成为街头巷尾热议的“时髦”话题，与此同时，各种关于云计算的商业和解决方案应运而生，各类理念和包装良莠不齐、不一而足，反而让大家对云计算到底能做些什么，对其潜在的客户到底能够解决什么实际问题，能够带来什么样的实际价值感到迷惘，使得大家对云计算的未来前景产生了怀疑。

关于云计算的社会价值与意义，我们常常用一句话表达云计算的目标诉求：“未来让人们像用水和用电那样使用云计算”。在这里，人们将云计算视为一种“水和电”那样无处不在、人类社会日常生产和生活过程中必不可少的基础资源。这里我们用“电力”来形容云计算可能更为恰当，因为电力是上一个工业文明时代最关键的生产资料。电灯、收音机、电视机、电冰箱、电风扇，乃至于自动化生产线等无不需要依赖于电力驱动。相比电力，云计算则对应于当前的知识与信息时代进行任何信息分析与处理的生产资料，用于支撑ERP/CRM/Email/BI大数据乃至金融实时交易数据处理等所有维持企业业务正常运作所需的按需获取、按需分配的关键资源。

从技术架构演进的视角来看，有人将云计算视为自IT领域冯·诺依曼计算机架构诞生之后的第三

次里程碑式的变革，是对传统计算架构与计算模式的颠覆与创新。也有人认为云计算无非是一种商业理念上的包装，所谓“新瓶装旧酒”，只是各个IT厂商用来“促销”自己产品的一种“营销活动”，并没有带来根本性的技术变革，也并没有给IT架构带来根本性的变化，那么真相究竟是怎样的呢？

回顾企业IT架构演进的整个历史，我们不难看到，冯·诺依曼架构的第一台计算机诞生以来的前30年，计算高度集中化、支持多用户多任务的大型机和小型机是企业IT的主流形态，构成IT系统的软硬件堆栈各层之间缺少统一的工业标准，呈现出内聚与耦合的特征，仅少数厂家拥有提供端到端高度复杂化的IT系统软硬件的能力。那个时代的IT系统造价高昂，往往是少数高端企业才能拥有的“奢侈品”。

于是，20世纪80年代，以x86服务器和PC系统的诞生为标志，企业IT系统迎来了第二次里程碑式的变革：从All in One、全封闭的软硬件栈走向了水平分层的网络、存储、服务器、操作系统、中间件、应用层等多层次水平分工的架构，各层之间接口标准化、规范化，极大简化了每一层的技术复杂度，各层IT产业链获得了大繁荣与大发展，涌现出一批优秀的专业化厂家，聚焦于提供该领域内质量最佳的产品和解决方案，IT系统终于开始走入“寻常百姓家”。

然而，所谓“物极必反”，当这个架构分层发展到一定阶段，弊端逐步显现。由于企业IT的层次太多，各层之间集成交付的难度越来越大，尤其是当今企业软件应用已从单一实例应用，迅速走向大规模分布式应用，一个关键业务的部署往往需要涉及服务器、网络、存储等各方面基础设施资源的协同配合，业务驱动的基础设施层服务器、存储、网络资源的集成管理配置和按需供给成为影响企业IT快速响应企业业务需求的关键制约因素。同时软硬件各层的开发虽然实现了解耦，部署和运行态仍然是软硬件耦合绑定的关系，因此跨服务器的资源出现忙闲不均时，依旧无法有效利用IT资源。

随着企业信息化进程的不断推进，企业IT系统的使用者和维护者们逐渐发现，分层架构体系也存在着诸多弊端：

- 软硬件开发态解耦，但部署和运行态并未解耦。
- 生态链大繁荣的同时，多厂家硬件异构集成与管理的复杂度越来越高。
- 企业信息化的重心向软件转移，但计算、存储、网络硬件弹性供给能力及其相互协同的不足，越来越成为软件价值提升的制约性因素。

那么，是否存在一条IT架构演进路径，可以在代价最小化，即在不对现有软硬件堆栈做颠覆式改动的前提下，有效应对上述关键痛点与挑战呢？

答案是肯定的，这就是IT领域的第三次里程碑式演进变革：从PC+服务器时代迈入云计算时代，通过虚拟化与云调度管理技术，将来自不同厂家的、多台烟囱式的、彼此孤立和割裂的计算、存储、网络设备在逻辑上整合成为一台“超大规模云计算机”，为上层的软件提供弹性的按需资源供给的能力，从而实现软硬件部署过程与运行态的解耦，屏蔽软硬件异构多厂家差异性与复杂度，并填补计算与存储之间的性能鸿沟。

大家也许已经注意到，我们谈到云计算驱动的第三次IT架构变革浪潮，其实早在云计算理念问世前的几年时间里，在众多互联网厂家中已被多次实践过，并且取得了巨大成功。那么普遍意义上的企业IT的云化重构又与互联网成功的实践之间存在着什么样的关联呢？

Google、Facebook的“云计算机”服务于其特定商业模式和业务应用，例如搜索类、社交类应用，而企业IT云化架构所期望的“云计算机”，则面临着大量的、形形色色的面向传统IT基础设施架构开发的企业应用和电信应用，他们的应用场景需求既有相同点，也存在着巨大的差异化。

相同点在于：

- 计算、存储实现了大规模资源池化，实现了规模经济效益；

- 对于分布式架构与负载均衡能力，资源可按业务需求灵活扩展伸缩；
- 依赖分布式软件在系统整体层面而非单点硬件层面实现高可靠性及高性能保障。

不同点在于：

- 普适性——互联网平台一般仅为其特定业务模型定制，企业云平台则要求具备对异构多厂家应用的普遍适用性；
- 异构兼容性——企业云平台需要考虑异构厂家硬件的兼容性，需要对企业IT基础设施现有投资提供最大化的保护；
- 高性能——互联网业务虽然并发量和注册用户量庞大，但企业高端应用在时延和性能方面却有更高的要求；
- 自动化、虚拟化——互联网业务模式一般为自主开发、自主运营(DevOps，Development和Operations的组合)，因此对管理自动化要求不迫切，企业应用则由于应用颗粒度不一，基础设施采购自第三方，因此管理自动化和虚拟化基本为必选能力。

“天下大势，合久必分，分久必合！”，云计算时代IT基础设施演进的下个十年，是从分离重新走向融合的十年：

- 通过云操作系统，将数据中心多厂家异构的计算、存储、网络资源进行水平融合，对外提供开放与标准化的IT服务接口，实现面向利用IT基础设施的“融合”；
- 通过超融合架构，单厂家计算、存储与网络资源进行垂直融合， 提供模块化、一站式、高性能、性价比最优、面向新建IT基础设施的交付模式。

无论IT架构如何螺旋式演进， 客户价值和驱动力都体现在：

- 更低的TCO；
- 更高的业务部署与生命周期管理效率；
- 更优的业务性能与用户体验。

IT基础设施架构从分离重新走向融合，并非简单的历史重复，而是在继承现有成果基础上的创新突破。无论水平融合，还是垂直融合，在核心技术支撑方面并未将现有已形成产业规模的x86 CPU及其服务器计算架构推倒重来，而是在最大限度地重用这些成熟产业组件的前提下，借助虚拟化及分布式云计算调度管理软件的作用，将多厂家异构，或者单厂家同构的计算、存储、网络整合为规模可大可小的“云计算机”，从而有效地解决传统IT架构所面临的业务上线周期长、TCO居高不下、企业关键应用性能低下的问题和挑战。

顾炯炯

目　录

第 1 章

云计算的商业动力与技术趋势

1.1 云计算基础概念与架构

云户端、服务器端并重的传统模式，向以“广泛的网络接入”、“计算、存储的集中资源池化”、“快捷的弹性伸缩”、“按需自助及可计量的服务”为典型特征的云计算模式的演进铺平了道路，并掀起了正在席卷全球的第三次IT变革浪潮。

在传统IT体系架构下，当前企业基础设施建设与运维所面临的核心痛点问题可以总结概括为如下几点。

1. 平均资源利用率及能耗效率低下

针对基础设施平台建设、扩容与更新换代，当前企业普遍采用的模式是服务器、网络交换与安全，以及存储设备的水平分层采购。各个IT基础设施单部件的选型、数量以及不同部件的组网连接方案均取决于企业IT收集的各业务部门对于IT核心业务处理量需求的预测和规划。同时所有企业IT应用软件、数据库以及中间件软件均采用独占计算、存储和网络资源的烟囱式部署。软件应用与硬件唯一捆绑，不同应用之间无法动态、高效共享相同的计算与存储资源。加之按照摩尔定律不断翻番增长的CPU计算能力已大大超出应用软件对计算资源利用率的同步能力，导致企业IT的平均资源利用率始终处于低于20%的水平。

2. 新业务上线测试周期长，效率低下

企业任何一项新业务上线，从最基础的硬件平台开始，向上逐层延伸至操作系统、中间件、数据库、CRM/ERP/HRM/PDM/Email/UC等各类业务关键软件堆栈，均需要投入IT专业化团队，进行软件安装、调试、功能与性能验证测试、网络配置及修改调整，然后经过若干轮测试、故障及性能稳定性测试定位及重配置和调整之后，才能最终达到期望正式上线运行的成熟度水准。这个过程一般需要长达2至3个月的时间。

资源储备及弹性伸缩能力不足，不具备应对企业IT突发业务高峰处理的能力

针对特定垂直行业短时间内突发性的高流量、高密度业务需求(比如节假日期间对视频网站的突发业务流程冲击)，企业内部物理基础设施资源往往无法满足短时间内迅速获取所需资源的需求，以及处置业务高峰过后的资源闲置问题。

3. 企业核心信息资产通过个人办公PC/便携外泄的安全风险，无法在个人智能终端(平板电脑、智能手机)方便地访问企业防火墙后的工作流及文档

部分企业核心信息资产通过员工个人PC电脑或便携设备外泄给竞争对手，对企业竞争力和商业利益带来负面影响。过分严格的信息安全管控措施又导致了工作效率的下降，企业管理层及员工无法便捷地通过无所不在的网络访问企业防火墙内部的信息资产。

4. 中小型企业希望通过宽带网络管道，从电信运营商或其他主机托管运营商的托管应用数据中心“按需获取”其所需的企业IT应用能力，从而实现日常运作中IT成本开销最小化

数量众多的中小企业，缺少IT领域专业经验，甚至没有财力和精力建设和维持自己专属的IT部门以及IT基础设施平台，普遍希望可以直接从托管运营商那里获取支撑其日常业务运作所需的SaaS服务。

针对解决上述企业IT系统建设和维护过程中遇到的普遍痛点问题，迫切呼唤业界IT软硬件解决方案提供商借助云计算技术，打造TCO、性价比与效率最优的“IT基础设施私有云及公有云”，具体包括：

- 面向大型企业和行业领域提供全自动化管理、一站式交付、支持与企业ITIL无缝集成融合、TCO最优化的端到端解决方案，实现企业传统IT基础设施及应用的改造、扩容和新建；
- 面向中小型企业(SMB)，提供支持多租户安全隔离与动态发放、超大规模资源池调度管理、可最大限度发挥规模经济效益的公有云托管解决方案。

无论上述哪一类形态，企业云计算IT基础设施平台均可定位于基础设施、中间层云平台服务、云计算业务发放与维护管理。针对云平台

服务层之上的多样化的内部IT软件及外部增值业务软件，企业(含运营商)可奉行“深淘滩、低做堰”的原则，广结各方ISV合作联盟，建设依托于云计算平台、繁荣的企业私有云及公有云生态系统。

通过上述私有云/公有云生态系统的建设，使得企业及运营商客户可以真正将“IT基础设施”、“开发部署平台”与“核心业务流程”及“对外服务”解耦，大幅精简企业及运营商的内部IT及对外业务的基础设施层建设部署、运营维护及生命周期管理成本，从而更好地聚焦“核心业务流程”及 “对外服务”的开发与定制，帮助企业和运营商在新形势下获得可持续发展。用一句话高度概括企业云计算IT基础设施平台的核心价值就是：“精简IT，敏捷商道”。

1.2 云计算的商业动力：企业ICT转型

1.2.1 互联网革命

基本上所有企业的第一目标都是追逐利益的最大化。政府与公共事业其实也是如此，只是政府和公共事业的利益是代表着国家利益和全民公共利益的最大化。即使是非营利组织，也会追求在有限投入的情况下，获得最高(数量或质量)的价值输出。所以“利益追求”是商业活动的一个最根本动力。企业对利益的追求，从时间上分为短期利益、中期利益和长期利益。在量化上分为谋基本生存、谋稳步发展、谋快速扩张。企业的商业转型的动力，自然也就来自于两个方面：外部竞争与环境变化导致的企业生存压力，以及更多利益的诱惑。

如今，环顾各行各业，每个企业所处环境和生存压力可能不尽相同。但是如果我们把当今企业所处的环境放到历史发展的画卷里，我们会发现，每个时代的企业所面临的主要挑战、压力与机会，大部分几乎都是相同的，那就是我们经常在教科书里面提到的一次又一次的商业革命、技术革命和政治革命。“革命不是请客吃饭”，而往往是“人头落地”，对应到企业，那就是每次革命都会潮起潮落般伴随着大量老企业的消亡和新企业的兴起。

蒸汽机时代的工业革命让大量的手工作坊消失，取而代之的是小型工厂。电力革命带来流水线作业模式的大规模工厂又同样淘汰了大量的小型工厂，并逐步诞生了行业垄断巨头。20世纪60年代起的第一波信息化革命及计算机革命，让第一代IT企业成为股票市场上耀眼的明星与全球首富的诞生地，很多传统企业紧跟这一轮信息化的浪潮，将计算机广泛应用到业务之中，但相比这些IT新秀，明显显得黯然失色。20世纪90年代起的第二波信息化革命——互联网革命，一大批一夜暴富的互联网新秀应运而生，第一代IT企业虽然没有立即被掀翻在地，但也被迅速地甩到了二流市场地位之中。IT行业之外的传统企业，除了IT行业的东家——金融行业外，很多企业在互联网大潮下已经显出了疲态，无所适从，因为他们虽然建立了互联网网站，却几乎没有从中受益。从2010年前后掀起的第三波信息化革命——移动互联网革命，同时伴随着源自互联网行业的商业模式革命，我们的商业世界正式进入了互联网寡头时代(也可以叫做“大数据时代”)，互联网(寡头)厂商不再是新秀，而已经是一个个的超级巨人，互联网厂商所从事的业务范畴也早已不再是互联网网站(或者某个手机APP)，而是向各行各业、全球各地快速渗透、影响、控制乃至颠覆各类传统企业。

在第三波信息化革命浪潮下，还没有向互联网转型成功的第一代IT企业，已经由二流市场地位，变得更加艰难，普遍出现业绩下滑，股价下跌，正在一路滑向被淘汰或兼并的火山口。IT行业的东家——金融行业也开始凌乱了，因为自己培养出的娃娃们——互联网巨头已经反过来快速渗入到金融行业之中，即互联网金融，使得传统金融行业不得不依仗制度保护延缓互联网金融带来的冲击，完全处于守势地位。

IT与金融以外的一部分传统垄断性行业，通

过非市场化的行业壁垒试图阻隔互联网巨头于行业主营业务之外(如矿山、石油、化工、电力、铁路、电信、政府与公共事业)，但回顾军事历史，早到特洛伊城，后到君士坦丁堡，再到“二战”马奇诺防线，短期可能有效的壁垒，没有一个能最终守住，商业也是一样。一些行业出于上方压力或纳入新技术的兴趣，试图开放部分非关键业务给互联网厂商运营，来拥抱互联网。其实，在大数据时代，一些传统行业甚至无法认识到哪些业务才是未来关键业务，现在开放给互联网行业去做的“非关键业务”往往是未来的关键业务，乃至核心业务，因为往往目前被认为“边缘”的终端用户类业务，才能收集到对未来最有价值的用户数据，而用户数据是未来大数据时代经营的核心。如今被互联网公司卡住用户数据的入口，等同于卡住了企业未来发展的咽喉(出租车行业所面临的互联网冲击就是一个微型的范例)。

而对于非垄断性的传统行业(如商业、服务业、制造业、物流业、饮食行业、非公共教育行业、医疗行业、房地产、影视娱乐业、体育、新闻业、个体农业、游戏业等)，在互联网大潮的冲击下基本上毫无招架之力，在商战上基本上逃脱不开要么“被杀”(企业倒闭)、要么“被俘”(迁移到互联网平台)的命运。互联网时代最先消失的就是渠道企业，之后是百货商场(百货商场现在基本上都转型到餐饮娱乐一条街了，10多年前北京中关村标志性的IT产品商场如今不得不关门歇业)。还没有消失的大量商家、中小企业则或出于服务便利与补贴的诱惑，或几乎别无选择地纳入到互联网运营平台之上(包括开店、日常运营、业务沟通、交易、店面管理等几乎所有经营活动)，成为互联网寡头主导控制下的生态一员，最终逐渐丧失交易权、定价权乃至经营权，但最终还要自负盈亏。为了保持互联网生态的活力，在互联网平台上，依旧会继续上演造富神话(但很难再是“首富”，因为首富是互联网平台的老板)，相反，对于在这个互联网平台上没有成功的企业和个人而言，他们的收益可能连一名互联网公司的员工都不如，保障更是无从谈起。传统行业中的大型企业，在互联网的压力(或诱惑)下，要么成为互联网平台的“VIP用户”被加入到互联网寡头的生态之中，要么靠自身实力进行转型，正面与互联网寡头展开竞争与合作。

在互联网大潮的冲击下，没有哪个行业、哪个企业或个人能够置身事外，其差异只会是受到冲击时间的先后，以及受到冲击的力度与波次。因为这是整个时代的发展与转型的动力。企业要想继续生存和发展，只有面对互联网大潮，进行积极转型，别无他路！

1.2.2 互联网企业的核心竞争力

在分析传统企业互联网转型之路之前，我们需要先分析一下互联网企业为何具有如此强大的杀伤力。

互联网企业和传统企业一样，也是由普通人组成的，他们并不是什么神童，他们的学识、精力、激情、工作效率与创新能力并不比传统企业的员工强大。如此有杀伤力的互联网企业，是竞争胜出的佼佼者。其竞争的原动力是互联网公司创立之初经营过程中外部恶劣环境所带来的生存压力，如2000年前后的互联网泡沫破裂，其导致大量互联网公司倒闭或被兼并(今天的O2O、P2P泡沫也是如此)，驱使互联网行业中生存下来的公司能够健康成长，实力强劲，并最终造就互联网公司让人畏惧的颠覆力。所谓的颠覆力并非一朝一夕养成的，而是经历了一个日积月累的成长过程。这场互联网公司与传统企业之间的竞争，早期有如龟兔赛跑。互联网公司一穷二白，传统企业资金雄厚，人才辈出。但赛跑的中局，乌龟却越跑越快，最终变成了忍者神龟，而兔子却还是那只兔子，兔子也可能没有睡觉，只是没有注意到乌龟在奔跑过程中的变化，如今忍者神龟正在变成智慧的狮子，不仅拥有了力量与速度，还有锋利的牙齿，食肉的性情，将要成为商业森林之王，这时兔子想不注意狮子(当初的乌龟)的动向已经不可能了。

所谓互联网公司颠覆性的竞争力，并非有什么神秘的武器，而是随便一本管理书籍中都

会提到的一些常识点，但是互联网公司通过日积月累，其在每个点上的竞争力与传统企业的差距已经不再是传统企业之间相互竞争的那种10%～20%的差距，而是至少10倍，多则百倍、千倍乃至数十万倍以上的差距，正是这种似乎遥不可及的差距，造成了互联网对传统行业如摧枯拉朽般的颠覆能力。

一、复杂盈利模式

与传统企业向消费者直接销售商品获得收入不同，绝大部分互联网公司从成立之初便向用户提供免费的服务。这给互联网公司带来极大的业务经营压力，一个是盈利模式，一个是经营成本，在这两方面竞争力构建上，互联网公司可谓绞尽了脑汁。

在盈利模式上，相比传统企业，互联网公司最终设计了更为复杂、先进的盈利模式，即我们通常所说的“羊毛出在狗身上，猪来买单”。相比传统企业，这种盈利模式的好处是，用户接受互联网公司的产品和服务几乎没有利益付出的负面障碍(要么无须付费，要么拥有极低的价格，消费者不仅没有觉得有付出，还感觉赚到了大便宜)。而最终为互联网提供免费服务的付费者也认为其每笔付出均有所值(这些付费者可能包括风险投资商、在互联网上做广告的企业，或者在互联网平台进行产品销售省去了渠道成本的企业)。这种复杂盈利模式让互联网公司的用户数量可以呈爆发式的增长，这是传统企业所不能企及的，从而形成对传统企业的一项巨大的竞争优势。如今互联网公司所设计的盈利模式更加复杂，已经不再是简单的羊、狗、猪这种三方的关系了，而是贯穿全产业链、全商业运作的一种生态模式。而传统企业如今还是以简单的商品买卖这种古老的交易模式为主。这种盈利模式的竞争力差异，可能会让传统企业最终落到“把自己卖了，还在替别人点钱”的尴尬境地。

二、极低成本

互联网公司的这种复杂盈利模式，必须在规模效应(需要时间培育)下才能获得回报，而放大规模，需要更高的经营成本。这就要求互联网公司，为了活到能够盈利，必须再绞尽脑汁考虑如何降低自身的运营成本。构成互联网公司的主要成本主要是经营互联网网站所需的IT设备成本、IT软件成本、机房租赁成本、网络成本、办公场地成本、人员成本、市场广告成本等。其中IT设备、软件机房、网络以及运维人力的成本是互联网公司的最大成本源(如今，新业务的现金补贴则成为初期规模化发展成本的大头，这需要和金融手段紧密结合)。对此，互联网公司不能再购买昂贵的商业IT产品，如产自知名IT厂商的小型机、数据库、操作系统等(除非一直拥有丰厚的资金与盈利，但这样的互联网公司很少)。他们只能寻求最便宜的解决方案，那就是我们现在看到的x86硬件和几乎全部基于开源软件一起构建的云计算平台。这个云计算平台相比传统企业使用的小型机、商业数据库、高端存储、商业应用软件等方式，成本至少下降了80%以上。而通过云计算的自动化运营技术，其大幅降低了运维人力的需求，一个运维人员可以管理数千台乃至上万台的IT设备。同时，基于云计算平台，其对机房基础设施也进行了优化改造，降低机房的能耗，即电力成本与场地成本。在业务上层，互联网公司千方百计地推进业务流程自动化的工作，使得大量传统企业人工流程在互联网平台上实现全自动化的处理，大幅降低了业务处理成本。

而传统企业，在丰厚的业务利润的滋养下，以及IT部门所处的企业非核心地位下，根本没有动力和条件向互联网公司那样拼命地降低IT成本。在大企业病的氛围下，传统企业所谓的降低成本，往往仅仅是为了一个漂亮的财务报表。实现每年10%～20%的成本下降幅度，即可以完美地达成当年业绩。在这种冰火两重天的环境下，让互联网公司在10年左右的时间里，大幅度地拉开了与传统企业的成本领先优势。如今，在传统金融行业每发放一笔贷款的成本竟然是互联网金融企业的1000倍。大家如果展开完全竞争，谁输谁赢，将一目了然。我们咋舌于如今互联网颠覆力的同时，不禁感叹，在互联网企业卧薪尝胆的

10年里，很多传统企业却蒙住了自己眼睛，似乎热衷于信息化的同时，却没有看清信息化(互联网)的真正威力。

三、极度的敏捷

早期的互联网公司进入门槛非常低，一台电脑、一台服务器，搭建个网站就可以成立一家互联网公司，没有什么技术专利、商业方法保护之类的障碍。同时，互联网行业又有大量风险投资商的追捧，更使得大量互联网初创企业遍地开花，一种互联网概念推出，立即一大群互联网企业跟进并争相模仿。同时，业务先行互联网公司所拥有的爆发式增长的用户数量会对后来跟进者形成天然屏障，这让互联网公司必须以最快的速度推出新型业务，获得足量用户，才有可能在互联网行业超级激烈(惨烈)的竞争中胜出。数百上千家同质服务的互联网公司中，最终活下来的只有一家到两家(有人经常拿互联网公司大量倒闭的例子，来印证互联网公司也不过如此，并质疑传统企业触网的价值，但其忽略了这正是互联网行业通过大量的优胜劣汰来获得强大竞争力的优势所在)。

为了快速推出业务，互联网公司也是无所不用其极。相比传统企业，互联网公司从人员组织架构、企业文化、经营模式、IT基础设施平台等方面都做了大幅改进。在组织架构上，即使是大型互联网企业，也放弃了传统企业那种逐层审批、大小领导签字画押的环节。业务研发团队自行进行业务决策，缩短内部业务研发外的时间。在企业文化上，为了业务快速上线，没日没夜的工作变得稀松平常，上线后再休息。在经营模式上，也完全打破传统企业把产品完美化再推向市场的策略，而是让位于业务上线时间，互联网公司让业务先上线，通过上线后用户反馈，再继续不断优化产品(规避了传统企业普遍存在的那种闭门造车的模式)。为了加快业务研发进度，在IT基础设施平台方面，互联网公司必须考虑用一个自动化的工具平台，利用这个平台，可以在最短的时间，开发出业务应用，并可以灰度发布，上线后还可以不断地继续完善。这就是我们已经熟知的云计算PaaS平台。经过这种为了加快业务上线速度进行的企业工作流程、组织架构、企业文化、IT平台的重构，互联网企业实现了新业务从研发立项到上线周期不会超过2周，最短只需不到2天的敏捷度。相比传统企业，特别是大型企业那种少则3～6个月，多则1～2年，甚至5年规划的项目，敏捷度提升了至少6倍，多则百倍。传统企业还在研究要不要推出一项互联网业务的时候，互联网企业已经通过这项业务赚到了白花花的银子，等传统企业推出了相同的业务，也基本上属于僵尸业务了(用户已经被先行者抢走而无人问津，微信推出很久之后，一些公司推出X信、Y信业务，最终均销声匿迹就是个鲜活的例子)。

四、真正的创新

在商业社会，“创新程度”与“失败风险”几乎是成正比的。比如，一个产品的新颖程度如果是80%，那么该产品(在市场上)失败的风险也是80%，甚至更高。对于大部分企业，特别是小企业而言，企业的决策者，从事高风险的创新，首先要考虑能否承受失败的风险。一个医药企业，研发并上市一款新药，投资可能是几十亿美元，这也使得传统企业在创新上非常谨慎，进行层层审批和审核。一个创新项目，可能要花费数月甚至数年的筹备过程。在传统行业，相对安全的生存模式是模仿已经成功企业的产品或服务，在地域进行差异化，相同地域则采用价格跟随策略来赚取一定的利润。但互联网行业的环境要比传统行业恶劣，因为互联网无所不在，除非有一些特殊的管制限制，否则很难有地域限制，即使有管制限制，只要遵从相关管制即可自由竞争，这样一家公司的互联网服务即可覆盖全球各个地域。鉴于互联网用户粘性的特点，先行者获得规模用户后，跟进者很难站稳脚跟。所以互联网公司必须通过创新，来寻求其他新的业务领域，才可能生存、发展和壮大。所以创新对互联网公司而言，相比传统企业会更加重要。

那么互联网公司是怎么克服创新风险问题的呢？他们主要从创新成本、创新范围和投资与组织模式几个方面来克服。其核心运作模式类似于风险投资的模式，同时并行投入大量的创新项目(为支撑大量并行创新项目，互联网公司也重构了其人力组织与决策架构)，即使单一项目的失败风险很高，当创新项目数量足够多时，总会有获得成功的项目，而有的项目一旦成功，其获得的回报，会高于对所有项目(包括失败项目)的总投入数倍乃至数十倍，最终通过创新获得收益。同时互联网公司将这个创新平台持续优化。首先这些创新项目所使用的资源最大化共享，比如办公场所、技术人员、IT基础设施(即云计算平台、实验设施等)。创新项目的资源共享降低了整体创新的成本。云计算平台让创新团队只需要几个人，且无须太高的知识水平即可完成快速创新工作，也同样降低了单项目人力成本和时间成本。互联网公司大量采用微创新的模式，也让创新成本获得了下降。如今，互联网公司还引入了生态运营模式，通过定向的开放平台的模式，让社会与产业力量广泛地参与到互联网平台的创新之中，这些基于互联网公司平台创新的企业(多为小企业)和个人，不仅无须互联网公司支付薪金和开发费用，而且在研发阶段还要向互联网公司交纳平台使用费，创新如果失败，互联网公司无须承担任何额外成本，创新成功，互联网公司则与这些创新者进行收益分成，前景看好的创新项目，互联网平台公司甚至可以直接收购。互联网公司这种生态创新的运营模式比公司内部VC风投型运营模式更上一层楼，成为了互联网平台公司一桩稳赚不赔的买卖，还因此披上了“大众创业、万众创新”的光鲜外衣。

目前互联网公司正在探索比生态创新更高级别的创新模式，那就是带有人工智能特性的自动化创新。未来个人商品和服务的需求收集、设计、制造、发货、使用、售后等全环节都不再人工参与，而是全自动化完成。目前这种自动化、智慧化创新模式已经在互联网网页(即人机界面)上在一定程度上实现。互联网公司已经能够为每个用户自动化、智慧化地定制不同的用户交互界面，提供不同定制化服务，而且这种服务在不断优化。目前在线上基础上，互联网公司积极探索线下的智能化创新产品，与工业4.0相结合，走向线上线下的智能一体化创新模式。

与传统企业相比，互联网公司在创新模式、创新速度、创新组织架构、创新生态架构、创新成本控制、智慧创新能力等方面均取得了大幅领先。大部分传统企业，至今还处于尴尬境地——为了一个创新项目内部争论不休，反复调研，评审，申报，层层领导审批背书，最终项目失败。双方差距之大，无须更多笔墨。

五、大数据寡头

如今的商业社会进入大数据时代已经是不争的事实，谁拥有数据，谁将拥有未来。相比传统企业，互联网公司最早意识到了数据的重要性，并几乎从不删除数据。至今，大型互联网公司拥有几千PB的数据已经稀松平常了，领先的互联网公司已经走向EB乃至ZB的数量级。而大型传统企业所拥有的数据量，也不过几PB到几十PB，拥有几百个PB数据的传统企业已经少之又少了。双方仅在数据量上就已经达到上百倍的差距。传统中小企业与互联网公司更没得比了。这还只是数据的数量，还不算质量，在数据质量方面，互联网公司对消费者(个人)信息的掌握更是拥有巨大的优势，大型互联网公司的用户数量都是以亿为单位。个人的几乎所有活动信息都会呈现在互联网之上，包括但不限于个人的姓名、电话、住址、社会家庭关系、活动轨迹、资金关系、资产数额、知识能力、个人喜好、照片、影像、银行账号、社会交流、商务交流等。除了个人信息，还有大量的企业信息，包括企业(特别是网上开店企业)的所有经营活动、资金活动、客户信息、市场状况、销售活动、广告活动等。当所有的个人信息和企业信息汇聚起来，又形成了整个经济数据。任何经济领域的风吹草动，都不会逃过互联网厂商大数据监测与分析系统的法眼，而且在信息获取时间方向上比传统企业或机构大幅领先

(比如某互联网公司公布的大企业景气指数曲线与国家统计局的指数曲线基本相同，但发布时间却比统计局的提前了5个月以上)。互联网企业获取的精准经济数据又可以反过来进行各种金融与市场商业活动。除了数据质量，在大数据处理技术上，互联网厂商也走在了前列。当大部分传统企业还在靠人工进行市场、经营与投资活动的时候，互联网公司已经开始进入了机器智能主导下的信息收集、分析、决策、处理的时代。阿尔法狗战胜围棋高手的案例清晰地反映了人工智能与人工决策之间的差距。当传统企业的决策指挥系统也全面落后于互联网公司的时候，双方在相同游戏规则下，对垒的胜算不会高于1：4，而且传统企业胜利的那一局，可能还是因为互联网公司为了要世界排名而已。其实1：4只是因为下了5局，如果下100局、1000局，数字估计会比1：4难看得多。

以上只是列出了一些互联网企业核心竞争力，还没有罗列互联网企业的其他优势，比如企业文化、技术人才构成、经营创意、薪酬待遇、员工的平均年龄与教育程度、政治地位以及国际化的视野等。互联网的优势明显，并非意味着传统企业毫无机会。因为互联网对传统行业的渗透才刚刚开始。传统行业还有时间(虽然留给我们的时间越来越少)，也有机会进行转身。那么传统企业在如今一波又一波信息革命的浪潮下，如何转型，才不会被拍倒在沙滩上呢？

1.2.3 传统企业的ICT转型

本书虽然是介绍云计算的书籍，但并不是想告诉读者，传统企业上了云计算就能让自己转型成功，能够跟互联网公司一决高下。如果真的这么简单，那么企业ICT转型只需要钱就可以了，这对很多企业都不是什么太大的问题，至少当前现金流都比较充沛。云计算只是企业转型过程中的一个必要的条件。

一、传统企业应先认清自己的落后点

早期互联网引起传统企业注意的时候，传统企业对互联网的认识多停留在互联网网站上面(如今是手机移动APP)，认为互联网就是一个网站或APP，传统企业自己也上马一个网站或手机APP就可以了，但大部分企业并没有在自己的互联网网站上获得什么收益，甚至很多企业的网站只是一个摆设。互联网公司发展到平台化阶段之后，传统的小企业开始大量地加入到互联网平台之中，甚至很多企业让互联网平台成为自己的业务窗口，并因此受益。也就是说，传统企业中的小企业最早认清并获取到了互联网的价值。但小企业无力于自己搭建互联网平台，而最终成为互联网平台生态中的成员。与小企业一样逐渐认知互联网价值的是生产消费品或服务类的大中型企业，比如手机、电脑、家电、服装、玩具等厂商，但这些厂商对互联网关注的焦点在于如何运用互联网，使其成为自己的一个新型产品销售渠道，很少有厂商将互联网定位为业务运营核心去建设和经营。而非直接面向消费者，或对消费者漠然的大中型企业单位，对互联网认知则非常有限，主要原因是互联网公司未切实触动自身利益的时候，几乎很少有企业远见到自己的差距和风险而提前布局。

对于传统企业而言，有个重要的疑问需要解决，那就是为何互联网等信息化平台会成为所有企业的业务核心，比如，一个拖拉机工厂，其核心业务是设计部门、装配生产线、测试与市场等部门，收益贡献来自机械和人，根本不是什么计算机，计算机与IT系统只是提供了一些辅助性工作，如财务、管理、计算等，并非业务核心。解决这个疑问的最佳答案是德国提出并已经局部实现的工业4.0概念。在工业4.0的场景下，一个制造企业，从产品的需求提出，到产品设计、原型生产、小批量试制、中等规模试制、测试验证，到大规模生产、物流仓储，再到市场销售的全环节、全流程，全部通过IT系统与互联网体系主导完成。人力工作只是在当前计算机设备能力有限的设计阶段和流程的规范性方面进行有限干预而已。在过去，一个新型号拖拉机从需求收集、设计、不同批量试制、生产到最终规模下线送到客户手里，可能需要1年的时间。而如今在工业4.0

场景下，可能只需要不到1天。某工业4.0场景下摩托车生产企业，从客户下达订单到定制化的摩托车交付，只需要6个小时。这种相比传统拖拉机厂的极度敏捷能力，就是IT与互联网核心平台支撑的结果。连一个拖拉机厂的未来都是完全信息化(互联网化)的，又有哪些企业不会走上完全的信息化路线呢？

再以与拖拉机厂这种制造企业运营模式似乎风牛马不相及的政府运营为例，政府未来的运营路径又是如何呢？政府作为公益、监管、执法机构，最高效率的运作就是利用互联网信息化的技术手段，打通与国家公民中所有企业、个人信息连接的壁垒，构建一个密集网状的信息化公共治理平台。比如：企业与个人的纳税可以在公共交易平台的交易与支付瞬间同时完成，税率也可以实现高度的定制化、个性化，可以针对不同企业经营与家庭状况收取不同税负并可调节。在这个公共信息化平台之上，所有的企业销售的商品与服务全程可追溯。而(企业)公民双方争议的调解和裁决大部分也可以在公共信息平台上解决，如此样例还有很多。政府达到这样一个公共信息化平台的信息化治理高度，其实不是一个幻想，这种治理模式实际上在一些互联网平台上已经完全或部分成为了现实，只是治理方是互联网公司而非政府而已。

在信息革命的时代背景下，所有传统企业、政府与公共事业都将走向更深度的信息化与互联网化。信息化平台也将成为企业单位的运营核心。而且在这个演进过程中，我们应该清醒地认识到，传统企业被互联网公司落在了后面。作为传统企业，应以最快的速度主动推进自身的ICT转型，让自己在被互联网企业冲击与淘汰之前早日成为互联网化的企业。

二、评估自己的防护壁垒与环境允许的转型窗口期

虽然互联网厂商在向各行各业快速渗透，但出于各行业独有的技术壁垒、监管资质壁垒、资源壁垒、垄断市场壁垒等限制，互联网厂商无法一下子在短时间内通吃一切，气愤之余，也因此给这些壁垒冠以“保护既得利益，阻碍改革深入发展”的帽子。无论如何，这些壁垒给各行业的传统企业一个难得的转型窗口期，一旦壁垒消失，己方的弓箭长矛就完全暴露在对方机枪、大炮与飞机的火力之下了，结果可想而知。不同行业的转型窗口期不尽相同，这跟互联网厂商的基础能力相关。当前阶段互联网公司的基础能力聚集在个人消费者与小企业及个人创业者层面。那么以个人和小企业为目标客户的传统企业，转型窗口期就更短。以大型企业或政府机构为目标客户的传统企业转型窗口期则相对较长。因为经济的运转最终要靠个人，所以任何传统企业都不会有太长的转型窗口期。按照IT更新换代的发展周期预计，短的窗口期也就3～5年，长的窗口期也难以超过10年。以生物技术行业为例，貌似生物学和IT没有特别直接的关系，而如今，最活跃的生物技术创业公司却是在IT行业的大本营——硅谷。这不仅是风投与创新氛围的原因，而且是IT信息化与生物技术深入融合的原因。还有我们看到的生物技术以外的航天技术领域和高铁技术领域，来自互联网行业的马斯克所创立的公司已经成功地部分完成了火箭回收实验和超高速高铁实验，其主导的特斯拉汽车早已商用就更不用说了，Google的无人驾驶汽车也已经可以商用上路。互联网公司还投入巨资，研究机器人技术，而机器人可以运用到几乎所有行业，替代大部分人工作业，从效率、成本、工作品质上均无法比拟。传统企业的决策者如果等看到互联网公司研发的机器人走到自己面前的时候，再考虑转型，可能连拿起电话通知总部的时间窗口都没有了。最具价值的机器人并不是人形的机器，而是机器人背后成千上万台服务器组成云计算平台支撑的人工智能。

三、将自己逐步打造成为立足本行业的互联网企业

很多企业谈到转型的时候，可能是为了给自己打气，增强信心，经常提到“后发优势”，

其实传统企业并没有什么后发优势，反而后发优势往往容易给人投机取巧的感觉。当初互联网公司发展起来，也不是因为什么后发优势，而是其所处的恶劣环境驱使的。现在环境恶劣的似乎不再是互联网公司了，而是传统企业了。对于和互联网公司一样，大部分都是由普通人组成的传统企业而言，转型可能有捷径，但不要指望什么捷径，先要从做强自己开始。要针对互联网公司所取得的那几项核心优势，需检讨自己如何改善。

1. 盈利模式探索

传统企业当前的经营状况良好，这使得传统企业没有动力重新构建新型盈利模式。但是从长远战略角度看，当前盈利模式是否长久，用户粘性是否牢固是个大问题。永远黏住客户的最佳方案，就是让用户没有感觉到付出，却一直能感觉到在获得回报。这是互联网免费的优势。为此传统企业也开始了多种探索，比如某轮胎生产企业，让客户可以选择不再购买轮胎，而是按照轮胎的使用里程付费，没有行使无须付费，让用户觉得更加划算。一个生产空气压缩机的企业，则可以让企业只按照空气供应量付费。那么一个服装企业是否可以免费地提供印上广告的服装给消费者呢，只要穿够多少天即可。诸如此类的盈利模式很多，各类传统企业均可以尝试探索。其实盈利模式是一种创新，当一个企业实现信息化支撑所有业务后，所有盈利模式都仅是一个APP的开发、上线与运营过程而已。但需要强调的是，盈利模式只是企业转型过程中的一环，而非全部，很多企业仅依靠商业模式，而没有强劲的底层技术平台支撑是很难长久的，因为商业模式容易被复制，几乎没有门槛。

2. 成本的竞争力构建

在成本竞争力构建方面，传统企业是最容易提升的，通过构建云计算平台，即可大幅降低IT信息化成本。但这还不够，传统企业需要一直保持信息化平台的成本竞争力。不仅是降低IT基础设施成本，而且需要通过应用的重构，实现业务流程的自动化来降低业务处理的时间成本与人力成本。同时整个IT基础设施平台，要在整个IT生态中保持一种成本的竞争力。传统企业需要具备IT信息化成本控制的主导权，并有能力压低成本，压低成本的方式不是通过议价，而是通过开放架构，并介入IT信息化平台的研发，也就是当无人帮自己降低成本的时候，自己也有能力降低平台成本。IT技术人才队伍的构建，是传统企业在转型过程中急需解决的一个重大问题。

3. 敏捷度提升

云计算平台可以帮助传统企业解决业务的敏捷度问题，包括业务的设计、开发、上线。但对传统企业挑战更大的是整体组织的敏捷度问题。处于转型期的企业，组织结构一直处于快速调整之中，并伴随着大量的优胜劣汰，触动着无数员工的利益。企业越大，组织敏捷度越成问题。企业信息化转型，貌似是IT部门的事情，实际上是企业整体的事情，是企业的关键战略。以目前传统企业IT部门的地位，仅能解决有限的敏捷度问题。IT部门的角色定位需要随着企业对信息化的重要性认识的深入，而不断提升。

4. 创新组织的架构

与组织敏捷遇到的问题类似，基于云计算平台的创新可以在IT部门内进行实验性的验证，若要最终发挥核心价值，需要企业做面向创新的组织架构转型。企业架构要走向扁平化，创新团队独立性更强，需要设立创新容错机制(项目失败是常态)。当云计算(即有限的企业信息化程度)还无法承载所有企业业务时，创新组织架构与创新活动则更是以人为核心的(甚至不是云平台支撑的)。目前一些家电制造企业和服装生产企业便实现了类似的创新型组织模式的转换，在大企业中设立大量小型团队完成从设计到生产交付的全流程工作，大量并行团队，满足了消费者非常个性化的需求，从而达到获取更高效益的目的。有人说这是工业3.5的状态，无论叫什么，这就是企业走向创新型组织的一步非常重要的尝试。当工业4.0条件具备的时候，这种小型创新型组织可以平滑地移植到全信息化处理平台之上(也就是未来的云计算产品创新平台之上)。

对于传统企业而言，组织的重构比IT系统的

重构更加关键，也更加困难。传统企业的ICT转型，必须要迈过组织重构这道门槛，才有可能成为创新型互联网企业。

5. 坚守行业大数据的主导权

从数据量上面看，传统企业无法与互联网公司相比拟，但很多传统企业在数据内容，也就是价值属性上会和互联网公司所拥有的数据区隔开来。因为传统企业具备大量的行业特性数据，比如电力行业拥有的电力数据、气象行业拥有的气象数据、农业拥有的农业数据、医疗行业拥有的医疗数据、石化行业拥有的勘探和化工数据等。另外，因为传统企业信息化深度的发展限制，使得很多传统企业的很多生产经营环节还没有实现完全的信息化，也就是没有变成IT数据，这部分也是传统企业未来数据量的一个来源。只要拥有独有的数据，就可以获得独有的价值，这也是企业能够长期存在的意义。企业需要用经营和长期发展的眼光看待数据对自身的价值和意义。企业或政府单位，可以开放自身数据，但不能丧失数据经营的主导权，丧失主导权，就可能丧失自己未来存在的意义。企业开放数据也是为了让更多外部数据为自己所用的一种数据交换。数据分析与数据经营团队也是企业转型过程中需要建立的一个新型组织，并需要与企业创新型组织相融合。

四、不是拥抱互联网，而是要拥抱未来

有人说互联网只是个工具，也有人说互联网只是虚拟经济。在这里我们要说的是，互联网行业在恶劣的环境下催生了一系列先进生产力与生产关系。而传统企业进行转型的目的，就是利用这一系列互联网行业所拥有的先进生产力与生产关系，去生产包括但不限于本行业的新产品，提供新服务，最终为企业获得更多的效益，并期望基业长青。互联网这些优秀的竞争力必须与企业的业务、企业自身优势资源结合在一起，才能发挥全新的生命力。

IT行业和互联网行业的诞生是信息化革命浪潮中的第一个与第二个波次，下一个波次信息化浪潮可能是延伸到各行各业自身的各个角落，而主导下一波次信息化浪潮的可能就来自各行各业的传统企业。

1.2.4 构建真正开放的新生态

在传统行业，要么是全行业垄断，要么就是全行业自由竞争，生态只是在这个自由竞争的环境下自然生成的产业链，生态链中的企业只要自扫门前雪就可以了，无须关注生态的动向，而在某些生态链条中的垄断厂商一般也只顾自身利益和同行的竞争，最多受限于反垄断法的威严而对利润率有所控制，很少关注和干预其他生态链中厂商的发展，因此传统行业的生态一直存在，但从来没有成为一个热词。但是在互联网及互联网控制下的行业，寡头企业已经能够控制影响整个产业生态链的每一个环节的每一个厂商，并最终发展成为对整个产业生态的经营。这就像一个“土豪”拥有超级的大牧场，牧场里种着各种草木、粮食，养着各种动物，除此之外，还有佃户、商户、工匠、寺庙、集市，所有生物的吃喝拉撒都在这个牧场里完成。牧场主人，那位“土豪”只需要收租子、收粮食，想吃肉了，就抓个动物杀了吃。

传统行业的生态链发展也非常有可能演变为一家独大的全生态经营模式，或者整个传统行业的生态链被纳入到现有的互联网生态平台之中。传统企业的转型发展，要格外注意自己在产业生态链中的地位、影响力与其他厂商的生态关系。最好的结局是做成牧场主，而不是一不小心成为别人生态牧场上的羔羊。

对于传统大型企业，特别是行业垄断性的企业，完全有能力发展一种行业生态经营模式。而对于中小企业，则需要考虑如何把自己的业务平衡地对接在不同的生态系统之间，如果能有块属于自己的独立自留地，自然是再好不过的了，但这个可能性随着不同生态圈之间的竞争会越来越小。总之，将来能成为牧场主来经营生态的传统企业可能少之又少。而对于大部分传统企业而言，加入哪个生态体系才显得尤为重要。

一个理想的产业生态体系该是什么样子的呢？

首先这个生态体系是要有生态聚合力的，也就是加入该生态的成员大部分能够获取相应的好处，并为这个好处而主动聚合在一起，共建这个生态，共同代表生态与其他生态系统进行竞争。

其次，这个生态体系要有内部的自由竞争，有了内部的竞争和流动，内部的优胜劣汰，才能保持生态成员的生命力，进而保持生态整体的竞争力。

第三，这个生态体系是要真正地完全开放，让生态成员自由地出入。这不能像如今一些企业所经营的生态体系那样，表面开放，实际上只是对生态成员(羔羊)加入的开放，而与其他竞争生态之间无法实现业务互通与成员流动，而且成员入驻后基本会逐渐形成业务锁定和业务依赖，而无法脱离该生态。一个开放的生态体系，是保障企业业务经营独立和自由的一个基本要素，如果一个企业丧失业务经营的主导权，那基本意味着企业的主体所有权的实质性被剥夺和转移，最终只能逐渐退化成躯壳乃至最终消失。

第四，这个生态应具备广泛的生态民主性。生态全体成员决定生态的未来，而不是某一家生态成员来独裁。只有这样，这个生态体系才能保障其生态成员利益的最大化。这种生态民主性并非凡事都搞全员公投，而是采用上市公司运作模式，全体成员作为股东选举出董事会、监事会，由董事会组织成立生态的行政经营与管理团队，对生态经营效果负责。

传统企业加入这样一个相对理想的生态体系，才能保障自身利益的最大化。目前这种相对理想的生态体系并非空中楼阁，而是在某些产业领域已经在实践。比如中国浙江某袜业基地的产业运作模式，以及云计算领域的OpenStack社区生态运作模式，就与这个理想生态体系非常接近。这两个生态圈均展现出了独到的生态活力与外部竞争力。

传统企业在转型过程中，要有意识地立足本行业、本专业领域建设类似的生态圈体系，聚合全行业的力量，搭建一个企业群自主经营的独立生态圈，与其他生态体系相互竞争、共同发展，而不是轻易地把企业自己的命运交给一个生态寡头打理。

在完全开放的跨行业生态圈框架下，传统企业可以获得全新的不亚于互联网企业的竞争力。这种竞争力的构建来自于跨产业链企业间的深度融合与协同。这种深度融合与协同，把产业链上下游的企业紧密地联合在一起，将各自的优势发挥到极致。上下游的企业不再是甲方、乙方的买卖关系，也不是各自心怀鬼胎，企图互相控制的关系，而是双方为了共同的目标联合运营，共同创新，共同面向市场，共同应对风险，共担损失，共享收益。为了一个新业务的上线不会再花去几个月的时间去进行采购和商务的谈判，不会再争执业务需求，不会再担心研发能力，甚至不会担心资金缺口。互联网公司所自豪的端到端研发与运营能力以及资金实力，在开放生态中的传统企业联合体同样可以实现。而且这个企业联合体会更具竞争力，因为这个生态更开放，更加敏捷，实力更强，各垂直链条的企业之间可以紧密合作却非紧耦合，横向链条之间的企业又可以充分竞争，在生态竞争维度又可以充分合作。充分的竞争可以让生态的每个环节更强壮，联合体的效率会更高效。充分的合作又会让生态整体实力更强大。而一个垄断生态平台，在高利润的保障下，因缺少改善动力，势必会走向更加封闭和低效。这种开放生态圈框架下企业联合体模式，也已经不是乌托邦了，而是有越来越多的企业在实践，比如IT厂商与电信运营商在全球范围内正在不断组建各种企业联合体进行公有云的运营便是一例。

企业间竞争像是星球的碰撞，而生态竞争像是两个星系的碰撞、融合，是一场更加复杂的竞争模式，孰优孰劣，谁输谁赢，或者最终是否能够共同发展，要看整个生态的经营与运作效果，以及最终的市场表现。生态成员之间以及生态圈与生态圈之间如何在竞争与合作中取得平衡，是生态建设与运营过程中需要不断探索与优化的过程。

1.3 企业云计算的发展趋势

随着云计算技术在各行各业日新月异的发展

与突破，以及云计算产学研生态链各环节持续不懈的“化云为雨”的努力，云计算的应用与价值挖掘已全面渗透到企业IT信息化以及电信网络转型变革的方方面面。各行业、各企业依据自身业务现状、竞争形势与信息化变革程度的不同，不断持续深化企业IT云化的程度，并从一个里程碑走向下一个里程碑。

一、云计算发展的里程碑

从云计算理念最初诞生至今，企业IT架构从传统非云架构，向目标云化架构的演进，可总结为经历了如下三大里程碑发展阶段。

1. 云计算1.0：面向数据中心管理员的IT基础设施资源虚拟化阶段

该阶段的关键特征体现为通过计算虚拟化技术的引入，将企业IT应用与底层的基础设施彻底分离解耦，将多个企业IT应用实例及运行环境(客户机操作系统)复用在相同的物理服务器上，并通过虚拟化集群调度软件，将更多的IT应用复用在更少的服务器节点上，从而实现资源利用效率的提升。

2. 云计算2.0：面向基础设施云租户和云用户的资源服务化与管理自动化阶段

该阶段的关键特征体现为通过管理平面的基础设施标准化服务与资源调度自动化软件的引入，以及数据平面的软件定义存储和软件定义网络技术，面向内部和外部的租户，将原本需要通过数据中心管理员人工干预的基础设施资源复杂低效的申请、释放与配置过程，转变为在必要的限定条件下(比如资源配额、权限审批等)的一键式全自动化资源发放服务过程。这个转变大幅提升了企业IT应用所需的基础设施资源的快速敏捷发放能力，缩短了企业IT应用上线所需的基础设施资源准备周期，将企业基础设施的静态滚动规划转变为动态按需资源的弹性按需供给过程。这个转变同时为企业IT支撑其核心业务走向敏捷，更好地应对瞬息万变的企业业务竞争与发展环境奠定了基础。云计算2.0阶段面向云租户的基础设施资源服务供给，可以是虚拟机形式，可以是容器(轻量化虚拟机)，也可以是物理机形式。该阶段的企业IT云化演进，暂时还不涉及基础设施层之上的企业IT应用与中间件、数据库软件架构的变化。

3. 云计算3.0：面向企业IT应用开发者及管理维护者的企业应用架构的分布式微服务化和企业数据架构的互联网化重构及大数据智能化阶段

该阶段的关键特征体现为：企业IT自身的应用架构逐步从(依托于传统商业数据库和中间件商业套件，为每个业务应用领域专门设计的、烟囱式的、高复杂度的、有状态的、规模庞大的)纵向扩展应用分层架构体系，走向(依托开源增强的、跨不同业务应用领域高度共享的)数据库、中间件平台服务层以及(功能更加轻量化解耦、数据与应用逻辑彻底分离的)分布式无状态化架构，从而使得企业IT在支撑企业业务敏捷化、智能化以及资源利用效率提升方面迈上一个新的高度和台阶，

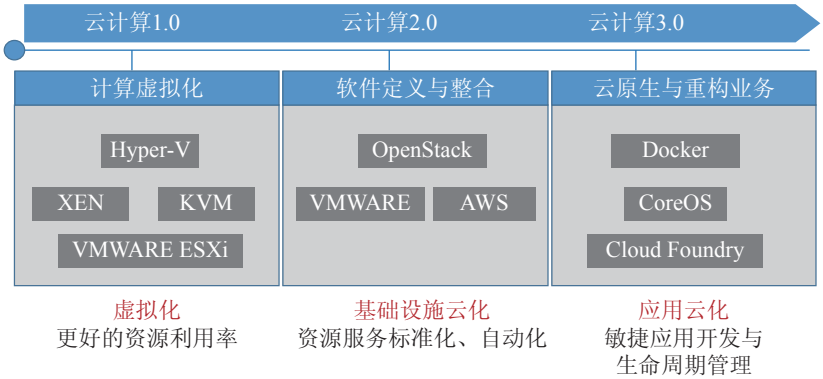


图1-1　云计算发展的三个阶段

并为企业创新业务的快速迭代开发铺平了道路。

针对上述三大云计算发展演进里程碑阶段而言，云计算1.0普遍已经是过去式，且一部分行业、企业客户已完成初步规模的云计算2.0建设商用，正在考虑该阶段的进一步扩容，以及面向云计算3.0的演进；而另一部分客户则正在从云计算1.0走向云计算2.0，甚至同步展开云计算2.0和3.0的演进评估与实施。

二、云计算各阶段间的主要差异

上述云计算里程碑阶段点之间，特别是云计算1.0与2.0/3.0阶段之间的主要差异体现为如下七点。

1. 差别1：从IT非关键应用走向电信网络应用和企业关键应用

站在云计算面向企业IT及电信网络的使用范围的视角来看，云计算发展初期，虚拟化技术主要局限于非关键应用，比如办公桌面云，开发测试云等。该阶段的应用往往对底层虚拟化带来的性能开销并不敏感。人们更加关注于资源池规模化集中之后资源利用效率的提升以及业务部署效率的提升。然而随着云计算的持续深入普及，企业IT云化的范围已从周边软件应用，逐步走向更加关键的企业应用，甚至企业的核心生产IT系统。由此，如何确保云平台可以更为高效、更为可靠地支撑好时延敏感的企业关键应用，就变得至关重要。

对于企业IT基础设施的核心资产而言，除去实实在在的计算、存储、网络资源等有形物理资产之外，最有价值的莫过于企业数据这些无形资产。在云计算的计算虚拟化技术发展初期阶段，Guest OS与Host OS之间的前后端I/O队列在I/O吞吐上的开销较大，而传统的结构化数据由于对I/O性能吞吐和时延要求很高，这两个原因导致很多事务关键型结构化数据在云化的初期阶段并未被纳入虚拟化改造的范畴，从而使得相关结构化数据的基础设施仍处于虚拟化乃至云计算资源池的管理范围之外。然而随着虚拟化XEN/KVM引擎在I/O性能上的不断优化提升(如采用SR-IOV直通、多队列优化技术)，使得处于企业核心应用的ERP等关系型关键数据库迁移到虚拟化平台上实现部署和运行已不是问题。

与此同时，云计算在最近2～3年内，已从概念发源地的互联网IT领域，渗透到电信运营商网络领域。互联网商业和技术模式的成功，启发电信运营商们通过引入云计算实现对现有电信网络和网元的重构来打破传统意义上电信厂家所采用的电信软件与电信硬件紧绑定的销售模式，同样享受到云计算为IT领域带来的红利，诸如：硬件TCO的降低，绿色节能，业务创新和部署效率的提升，对多国多子网的电信功能的快速软件定制化以及更强的对外能力开放。

2. 差别2：从计算虚拟化走向存储虚拟化和网络虚拟化

从支撑云计算按需、弹性分配资源，与硬件解耦的虚拟化技术的角度来看，云计算早期阶段主要聚焦在计算虚拟化领域。事实上，众所周知的计算虚拟化技术早在IBM 370时代就已经在其大型机操作系统上诞生了。技术原理是通过在OS与裸机硬件之间插入虚拟化层，来在裸机硬件指令系统之上仿真模拟出多个370大型机的“运行环境”，使得上层“误认为”自己运行在一个独占系统之上，实际上是由计算虚拟化引擎在多个虚拟机之间进行CPU分时调度，同时对内存、I/O、网络等访问也进行访问屏蔽。后来只不过当x86平台演进成为在IT领域硬件平台的主流之后，VMware ESX、XEN、KVM等依托于单机OS的计算虚拟化技术才将IBM 370的虚拟化机制在x86服务器的硬件体系架构下实现并且商品化，并且在单机/单服务器虚拟化的基础上引入了具备虚拟机动态迁移和HA调度能力的中小集群管理软件(如vCenter/vSphere、XEN Center、FusionSphere等)，从而形成当前的计算虚拟化主体。

随着数据和信息越来越成为企业IT中最为核心的资产，作为数据信息持久化载体的存储已经逐步从服务器计算中剥离出来成为了一个庞大的独立产业，与必不可少的CPU计算能力一样，在数据中心发挥着至关重要的作用。当企业对存储

的需求发生变化时该如何快速满足新的需求以及如何利用好已经存在的多厂家的存储，这些问题都需要存储虚拟化技术来解决。

与此同时，现代企业数据中心的IT硬件的主体已经不再是封闭的、主从式架构的大小型机一统天下的时代。客户端与服务器之间南北方向通信、服务器与服务器之间东西方向协作通信以及从企业内部网络访问远程网络和公众网络的通信均已走入了基于对等、开放为主要特征的以太互联和广域网互联时代。因此，网络也成为计算、存储之后，数据中心IT基础设施中不可或缺的“三要素”之一。

就企业数据中心端到端基础设施解决方案而言，服务器计算虚拟化已经远远不能满足用户在企业数据中心内对按需分配资源、弹性分配资源、与硬件解耦的分配资源的能力需求，由此存储虚拟化和网络虚拟化技术应运而生。

除去云管理和调度所完成的管理控制面的API与信息模型归一化处理之外，虚拟化的重要特征是通过在指令访问的数据面上，对所有原始的访问命令字进行截获，并实时执行“欺骗”式仿真动作，使得被访问的资源呈现出与其真正的物理资源不同的(软件无须关注硬件)、“按需获取”的颗粒度。对于普通x86服务器来说，CPU和内存资源虚拟化后再将其(以虚拟机CPU/内存规格)按需供给给资源消费者(上层业务用户)。计算能力的快速发展，以及软件通过负载均衡机制进行水平扩展的能力提升，计算虚拟化中仅存在资源池的“大分小”的问题。然而对于存储来说，由于最基本的硬盘(SATA/SAS)容量有限，而客户、租户对数据容量的需求越来越大，因此必须考虑对数据中心内跨越多个松耦合的分布式服务器单元内的存储资源(服务器内的存储资源、外置SAN/NAS在内的存储资源)进行“小聚大”的整合，组成存储资源池。这个存储资源池，可能是某一厂家提供的存储软硬件组成的同构资源池，也可以是被存储虚拟化层整合成为跨多厂家异构存储的统一资源池。各种存储资源池均能以统一的块存储、对象存储或者文件的数据面格式进行访问。

对于数据中心网络来说，其实网络的需求并不是凭空而来的，而是来源于业务应用，与作为网络端节点的计算和存储资源有着无法切断的内在关联性。然而，传统的网络交换功能都是在物理交换机和路由器设备上完成的，网络功能对上层业务应用而言仅仅体现为一个一个被通信链路连接起来的孤立的“盒子”，无法动态感知来自上层业务的网络功能需求，完全需要人工配置的方式来实现对业务层网络组网与安全隔离策略的需要。在多租户虚拟化的环境下，不同租户对于边缘的路由及网关设备的配置管理需求也存在极大的差异化，而物理路由器和防火墙自身的多实例能力也无法满足云环境下租户数量的要求，采用与租户数量等量的路由器与防火墙物理设备，成本上又无法被多数客户所接受。于是人们思考是否可能将网络自身的功能从专用封闭平台迁移到服务器通用x86平台上来。这样至少网络端节点的实例就可以由云操作系统来直接自动化地创建和销毁，并通过一次性建立起来的物理网络连接矩阵，进行任意两个网络端节点之间的虚拟通讯链路建立，以及必要的安全隔离保障，从而里程碑式地实现了业务驱动的网络自动化管理配置，大幅度降低数据中心网络管理的复杂度。从资源利用率的视角来看，任意两个虚拟网络节点之间的流量带宽，都需要通过物理网络来交换和承载，因此只要不超过物理网络的资源配额上限(缺省建议物理网络按照无阻塞的CLOS模式来设计实施)，只要虚拟节点被释放，其所对应的网络带宽占用也将被同步释放，因此也就相当于实现对物理网络资源的最大限度的“网络资源动态共享”。换句话说，网络虚拟化让多个盒子式的网络实体第一次以一个统一整合的“网络资源池”的形态，出现在业务应用层面前，同时与计算和存储资源之间，也有了统一协同机制。

3. 差别3：资源池从小规模的资源虚拟化整合走向更大规模的资源池构建，应用范围从企业内部走向多租户的基础设施服务乃至端到端IT服务

站在云计算提供像用水用电一样方便的服

务能力的技术实现角度来看，云计算发展早期，虚拟化技术(如VMware ESX、微软Hyper-V、基于Linux的XEN、KVM)被普遍采用，被用来实现以服务器为中心的虚拟化资源整合，在这个阶段，企业数据中心的服务器只是部分孤岛式的虚拟化以及资源池整合，还没有明确的多租户以及服务自动化的理念，服务器资源池整合的服务对象是数据中心的基础设施硬件以及应用软件的管理人员。在实施虚拟化之前，物理的服务器及存储、网络硬件是数据中心管理人员的管理对象，在实施虚拟化之后，管理对象从物理机转变为虚拟机及其对应的存储卷、软件虚拟交换机，甚至软件防火墙功能。目标是实现多应用实例和操作系统软件在硬件上最大限度共享服务器硬件，通过多应用负载的削峰错谷达到资源利用率提升的目的，同时为应用软件进一步提供额外的HA/FT(High Availability/Fault Tolerance，高可用性/容错)可靠性保护，以及通过轻载合并、重载分离的动态调度，对空载服务器进行下电控制，实现PUE功耗效率的优化提升。

然而，这些虚拟化资源池的构建，仅仅是从数据中心管理员视角实现了资源利用率和能效比的提升，与真正的面向多租户的自动化云服务模式仍然相差甚远。因为在云计算进一步走向普及深入的新阶段，通过虚拟化整合之后的资源池的服务对象，不能再仅仅局限于数据中心管理员本身，而是需要扩展到每个云租户。因此云平台必须在基础设施资源运维监控管理Portal的基础上，进一步面向每个内部或者外部的云租户提供按需定制基础设施资源，订购与日常维护管理的Portal或者API界面，并将虚拟化或者物理的基础设施资源的增、删、改、查等权限按照分权分域的原则赋予每个云租户，每个云租户仅被授权访问其自己申请创建的计算、存储以及与相应资源附着绑定的OS和应用软件资源，最终使得这些云租户可以在无须购买任何硬件IT设备的前提下，实现按需快速资源获取，以及高度自动化部署的IT业务敏捷能力的支撑，从而将云资源池的规模经济效益，以及弹性按需的快速资源服务的价值充分发掘出来。

4. 差别4：数据规模从小规模走向海量，数据形态从传统结构化走向非结构化和半结构化

站在云计算系统需要提供的处理能力角度看，随着智能终端的普及、社区网络的火热、物联网的逐步兴起，IT网络中的数据形态已经由传统的结构化、小规模数据，迅速发展成为有大量文本、大量图片、大量视频的非结构化和半结构化数据，数据量也是呈几何指数的方式增长。

对非结构化、半结构化大数据的处理而产生的数据计算和存储量的规模需求，已远远超出传统的Scale-Up硬件系统可以处理的，因此要求必须充分利用云计算提供的Scale-Out架构特征，按需获得大规模资源池来应对大数据的高效高容量分析处理的需求。

企业内日常事务交易过程中积累的大数据或者从关联客户社交网络以及网站服务中抓取的大数据，其加工处理往往并不需要实时处理，也不需要系统处于持续化的工作态，因此共享的海量存储平台，以及批量并行计算资源的动态申请与释放能力，将成为未来企业以最高效的方式支撑大数据资源需求的解决方案选择。

5. 差别5：企业和消费者应用的人机交互计算模式，也逐步从本地固定计算走向云端计算、移动智能终端及浸入式体验瘦终端接入的模式

随着企业和消费者应用云化演进的不断深入，用户近端计算、存储资源不断从近端计算剥离，并不断向远端的数据中心迁移和集中化部署，从而带来了企业用户如何通过企业内部局域网及外部固定、移动宽带广域网等多种不同途径，借助固定、移动，乃至浸入式体验等多种不同瘦终端或智能终端形态接入云端企业应用的问题。面对局域网及广域网连接在通信包转发与传输时延不稳定、丢包以及端到端QoS质量保障机制缺失等实际挑战，如何确保远程云接入的性能体验达到与本地计算相同或近似的水平，成为企业云计算IT基础设施平台面临的又一大挑战。

为应对云接入管道上不同业务类型对业务体验的不同诉求，业界通用的远程桌面接入协议在

满足本地计算体验方面已越来越无法满足当前人机交互模式发展所带来的挑战，需要重点聚焦解决面向IP多媒体音视频的端到端QoS/QoE优化，并针对不同业务类别加以动态识别并区别处理，使其满足如下场景需求。

- 普通办公业务响应时延小于100ms，带宽占用小于150Kbps：通过在服务器端截获GDI/DX/OpenGL绘图指令，结合对网络流量的实时监控和分析，从而选择最佳传输方式和压缩算法，将服务端绘图指令重定向到瘦客户端或软终端重放，从而实现时延与带宽占用的最小化。
- 针对虚拟桌面环境下VoIP质量普遍不佳的情况，缺省的桌面协议TCP连接不适合作为VoIP承载协议的特点：采用RTP/UDP代替TCP，并选择G.729/AMR等成熟的VoIP Codec；瘦客户端可以在支持VoIP/UC客户端的情况下，尽量引入VoIP虚拟机旁路方案，从而减少不必要的额外编解码处理带来的时延及话音质量上的开销。上述优化措施使得虚拟桌面环境下的话音业务MOS平均评估值从3.3提升到4.0。
- 针对远程云接入的高清(1080p/720p)视频播放场景：在云端桌面的多虚拟机并发且支持媒体流重定向的场景下，针对普通瘦终端高清视频解码处理能力不足的问题，桌面接入协议客户端软件应具备通过专用API调用具备瘦终端芯片多媒体硬解码处理能力；部分应用如Flash以及直接读写显卡硬件的视频软件，必须依赖GPU或硬件DSP的并发编解码能力，基于通用CPU的软件编解码将导致画面停滞、体验无法接受，此时就需要引入硬件GPU虚拟化或DSP加速卡来有效提升云端高清视频应用的访问体验，达到与本地视频播放相同的清晰与流畅度。桌面协议还能够智能识别并区分画面变化热度，仅对变化度高且绘图指令重定向无法覆盖部分才启动带宽消耗较高的显存数据压缩重定向。
- 针对工程机械制图、硬件PCB制图、3D游戏，以及最新近期兴起VR仿真等云端图形计算密集型类应用：同样需要大量的虚拟化GPU资源进行硬件辅助的渲染与压缩加速处理，同时对接入带宽(单路几十到上百M带宽，并发达到数10G/100G)提出了更高的要求，在云接入节点与集中式数据中心站点间的带宽有限的前提下，就需要考虑进一步将大集中式的数据中心改造为逻辑集中、物理分散的分布式数据中心，从而将VDI/VR等人机交互式重负载直接部署在靠近用户接入的Service PoP点的位置上。

另一方面，正当全球消费者IT步入方兴未艾的Post-PC时代大门之时，iOS及Android移动智能终端同样正在悄悄取代企业用户办公位上的PC甚至便携电脑，企业用户希望通过智能终端不仅可以方便地访问传统Windows桌面应用，同样期待可以从统一的“桌面工作空间”访问公司内部的Web SaaS应用、第三方的外部SaaS应用，以及其他Linux桌面系统里的应用，而且希望一套企业的云端应用可以不必针对每类智能终端OS平台开发多套程序，就能够提供覆盖所有智能终端形态的统一业务体验，针对此BYOD云接入的需求，企业云计算需在Windows桌面应用云接入的自研桌面协议基础上，进一步引入基于HTML5协议、支持跨多种桌面OS系统、支持统一认证及应用聚合、支持应用零安装升级维护，及异构智能终端多屏接入统一体验的云接入解决方案——Web Desktop。

6. 差别6：云资源服务从单一虚拟化，走向异构兼容虚拟化、轻量级容器化以及裸金属物理机服务器

在传统企业IT架构向目标架构演进的过程中，为了实现应用的快速批量可复制，以闭源VMware、Hyper-V及开源XEN、KVM为代表的虚拟化是最早成熟和广泛采纳的技术，使得应用安装与配置过程可基于最佳实践以虚拟机模板和镜像的形式固化下来，从而在后续的部署过程中大大简化可重复的复杂IT应用的安装发放与配置过程，使得软件部署周期缩短到以小时乃至以分钟计算的程度。然而，随着企业IT应用越来越多地从小规模、单体式的有状态应用走向大规模、分布式、数据与逻辑分离的无状态应用，人们开始意识到虚拟机虽然可以较好地解决大规模IT数

据中心内多实例应用的服务器主机资源共享的问题，但对于租户内部多个应用，特别是成百上千，甚至数以万计的并发应用实例而言，均需重复创建成百上千的操作系统实例，资源消耗大，同时虚拟机应用实例的创建、启动，以及生命周期升级效率也难以满足在线Web服务类、大数据分析计算类应用这种突发性业务对快速资源获取的需求。以Google、Facebook、Amazon等为代表的互联网企业，开始广泛引入Linux容器技术(namespace、cgroup等机制)，基于共享Linux内核，对应用实例的运行环境以容器为单位进行隔离部署，并将其配置信息与运行环境一同打包封装，并通过容器集群调度技术(如Kubernetes、MESOS、Swarm等)实现高并发、分布式的多容器实例的快速秒级发放及大规模容动态编排和管理，从而将大规模软件部署与生命周期管理，以及软件DevOps敏捷快速迭代开发与上线效率提升到了一个新的高度。尽管从长远趋势上来看，容器技术终将以其更为轻量化、敏捷化的优势取代虚拟化技术，但在短期内仍很难彻底解决跨租户的安全隔离和多容器共享主机超分配情况下的资源抢占保护问题，因此，容器仍将在可见的未来继续依赖跨虚拟机和物理机的隔离机制来实现不同租户之间的运行环境隔离与服务质量保障。

与此同时，对于多数企业用户来说，部分企业应用和中间件由于特殊的厂家支持策略限制，以及对企业级高性能保障与兼容性的诉求，特别是商用数据库类业务负载，如Oracle RAC集群数据和HANA内存计算数据库，并不适合运行在虚拟化上，但客户依然希望针对这部分应用负载可以在物理机环境下获得与虚拟化、容器化环境下相似的基础设施资源池化按需供给和配置自动化能力。这就要求云平台和云管理软件不仅仅要实现物理机资源自身的自动化操作系统与应用安装自动化，也需要进一步在保障多租户隔离安全的情况下实现与存储和网络资源池协同的管理与配置自动化能力。

7. 差别7：云平台和云管理软件从闭源、封闭走向开源、开放

从云计算平台的接口兼容能力角度看，云计算早期阶段，闭源VMware vSphere/vCenter、微软SystemCenter/Hyper-V云平台软件由于其虚拟化成熟度遥遥领先于开源云平台软件的成熟度，因此导致闭源的私有云平台成为业界主流的选择。然而，随着XEN/KVM虚拟化开源，以及OpenStack、CloudStack、Eucalyptus等云操作系统OS开源软件系统的崛起和快速进步，开源力量迅速发展壮大起来，迎头赶上并逐步成长为可以左右行业发展格局的重要决定性力量。仅以OpenStack为例，目前IBM、HP、SUSE、Redhat、Ubuntu等领先的软硬件公司都已成为OpenStack的白金会员，从2010年诞生第一个版本开始，平均每半年发布一个新版本，所有会员均积极投身到开源贡献中来，到目前为止已推出13个版本(A/B/C/D/E/F/G/H/I/J/K/L/M)，繁荣的社区发展驱动其功能不断完善，并稳步、快速地迭代演进。2014年上半年，OpenStack的成熟度已与vCloud/vSphere 5.0版本的水平相当，满足基本规模商用和部署要求。从目前的发展态势来看，OpenStack开源大有成为云计算领域的Linux开源之势。回想2001年前后，当Linux OS仍相当弱小、UNIX操作系统大行其道、占据企业IT系统主要生产平台的阶段，多数人不会想象到仅10年的时间，开源Linux已取代闭源UNIX，成为主导企业IT服务端的缺省操作系统的选择，小型机甚至大型机硬件也正在进行向通用x86服务器的演进。

本书下面的内容将重点围绕云计算出现的这些新变化来讲述云计算的架构技术。

第 2 章

云计算的架构内涵与关键技术

2.1 云计算的总体架构

从上述分析不难看出，云计算推动了IT领域自20世纪50年代以来的三次变革浪潮，对各行各业数据中心基础设施的架构演进及上层应用与中间件层软件的运营管理模式产生了深远的影响。在云计算发展早期，Google、Amazon、Facebook等互联网巨头们在其超大规模Web搜索、电子商务及社交等创新应用的牵引下，率先提炼出了云计算的技术和商业架构理念，并树立了云计算参考架构的标杆与典范，但那个时期，多数行业与企业IT的数据中心仍然采用传统的以硬件资源为中心的架构，即便是已进行了部分云化的探索，也多为新建的孤岛式虚拟化资源池(如基于VMware的服务器资源整合)，或者仅仅对原有软件系统的服务器进行虚拟化整合改造。随着近两年云计算技术与架构在各行各业信息化建设中和数据中心的演进变革，以及更加广泛和全面地落地部署与应用，企业数据中心IT架构正在面临一场前所未有的，以“基础设施软件定义与管理自动化”、“数据智能化与价值转换”以及“应用架构开源化及分布式无状态化”为特征的转化。

从架构视角来看，云计算正在推动全球IT的格局进入新一轮“分久必合、合久必分”的历史演进周期，我们通过分离回归融合的过程从三个层面进行表述，如图2-1所示。

1. 基础设施资源层融合

面向企业IT基础设施运维者的数据中心计算、存储、网络资源层，不再体现为彼此独立和割裂的服务器、网络、存储设备，以及小规模的虚拟化资源池，而是通过引入云操作系统，在数据中心将多个虚拟化集群资源池统一整合为规格更大的逻辑资源池，甚至进一步将地理上分散、但相互间通过MPLS/VPN专线或公网连接的多个数据中心以及多个异构云中的基础设施资源整合为统一的逻辑资源池，并对外抽象为标准化、面向外部租户(公有云)和内部租户(私有云)的基础设施服务，租户仅需制定其在软件定义的API参数中所需资源的数量、SLA/QoS及安全隔离需求，即可从底层基础设施服务中以全自动模式弹性、按需、敏捷地获取到上层应用所需的资源配备。

2. 数据层融合

面向企业日常业务经营管理者的数据信息资

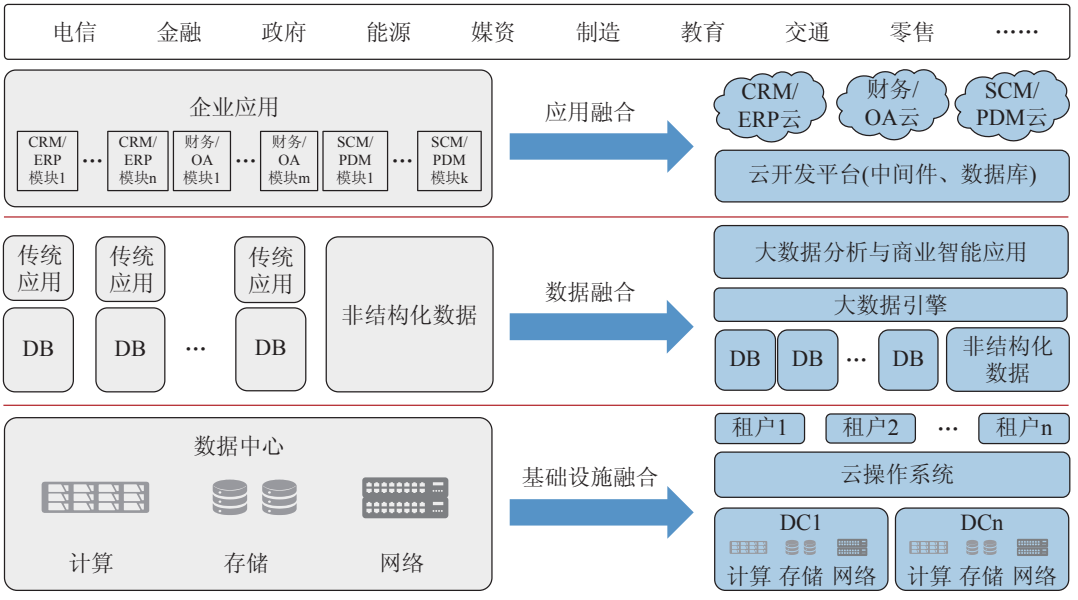


图2-1 企业IT架构的云化演进路径

产层，不再体现为散落在各个企业、消费者IT应用中，如多个看似关联不大的结构化事务处理记录(关系型数据库)数据孤岛，非结构化的文档、媒体以及日志数据信息片段，而是通过引入大数据引擎，将这些结构化与非结构化的信息进行统一汇总，汇聚存储和处理，基于多维度的挖掘分析与深度学习，从中迭代训练出对业务发展优化及客户满意度提升有关键价值的信息，从而将经营管理决策从纯粹依赖人员经验积累转变到更多依赖基于大数据信息内部蕴藏的智慧信息，来支撑更科学、更敏捷的商业决策。除大数据之外，数据层融合的另一个驱动力，来自于传统商业数据库在处理高并发在线处理及后分析处理扩展性方面所遭遇的不可逾越的架构与成本的瓶颈，从而驱动传统商业闭源数据库逐步被Scale Out架构的数据库分表分库及水平扩展的开源数据库所替代。

3. 应用平台层融合

面向企业IT业务开发者和供应者的应用平台层，在传统IT架构下，随具体业务应用领域的不同，呈现出条块化分割、各自为战的情况，各应用系统底层的基础中间件能力，以及可重用的业务中间件能力，尽管有众多可共享重用的机会点，但重复建设的情况非常普遍(比如ERP系统和SCM系统都涉及库存管理)，开发投入浪费相当严重。各业务应用领域之间由于具体技术实现平台选择的不同，也无法做到通畅的信息交互与集成；而企业IT应用开发本身，也面临着在传统瀑布式软件开发模型下开发流程笨重、测试验证上线周期长、客户需求响应慢等痛点。于是，人们开始积极探索基于云应用开发平台来实现跨应用领域基础公共开发平台与中间件能力去重整合，节省重复投入，同时通过在云开发平台中集成透明的开源中间件来替代封闭的商业中间件平台套件，特别通过引入面向云原生应用的容器化应用安装、监控、弹性伸缩及生命周期版本灰度升级管理的持续集成与部署流水线，来推动企业应用从面向高复杂度、厚重应用服务的瀑布式开发模式，逐步向基于分布式、轻量化微服务的敏捷迭代、持续集成的开发模式演进。以往复杂、费时的应用部署与配置，乃至自动化测试脚本，如今都可以按需地与应用软件打包，并可以将这些动作从生产环境的上线部署阶段，前移到持续开发集成与测试阶段。应用部署与环境依赖可以被固化在一起，在后续各阶段以及多个数据中心及应用上下文均可以批量复制，从而将企业应用的开发周期从数月降低数周，大大提升了企业应用相应客户需求的敏捷度。

综上所述，企业IT架构云计算演进中上述三个层次的融合演进，最终目的只有一个，那就是通过推动企业IT走向极致的敏捷化、智能化以及投入产出比的最优化，使得企业IT可以更好地支撑企业核心业务，进而带来企业业务敏捷性、核心生产力与竞争力的大幅提升，以更加从容地应对来自竞争对手的挑战，更轻松地应对客户需求的快速多变。

那么，云计算新发展阶段具体的架构形态究竟是怎样的呢？是否存在一个对于所有垂直行业的企业数据中心基础设施云化演进，以及无论对于公有云、私有云及混合云场景都普遍适用的一个标准化云平台架构呢？

答案无疑是肯定的。

尽管从外在表象上来看，私有云与公有云在商业模式、运营管理集成存在显著差别，然而从技术架构视角来看，我们宏观上不妨可将云计算整体架构划分为云运营(Cloud BSS)、云运维(Cloud OSS)以及云平台系统(IaaS/PaaS/SaaS)三大子系统，这三大子系统相互间毫无疑问是完全SOA解耦的关系，云平台和云运营支撑子系统很明显是可以实现在公有云和私有云场景下完全重用的，仅云运营子系统(Cloud BSS)部分，对于公有云和私有云/混合云存在一定差异。因此，我只需要将这部分进一步细分解耦打开，即可看到公有云、私有云可以共享的部分，如：基础计量计费，IAM认证鉴权，私有云所特有的ITIL流程对接与审批、多层级租户资源配额管理等，以及公有云所特有的批价、套餐促销和在线动态注册等。

由此我们可以看到，无论是公有云、私有云，还是混合云，其核心实质是完全相同的，都

是在基础设施层、数据层，以及应用平台层上，将分散的、独立的多个信息资产孤岛，依托相应层次的分布式软件实现逻辑上的统一整合，然后再基于此资源池，以Web Portal或者API为界面，向外部云租户或者内部云租户提供按需分配与释放的基础设施层、数据层以及应用平台服务，云租户可以通过Web Portal或者API界面给出其从业务应用的需求视角出发，向云计算平台提出自动化、动态、按需的服务能力消费需求，并得以满足。

综上所述，一套统一的云计算架构完全可以同时覆盖于公有云、私有云、混合云等所有典型应用场景。

2.1.1 云计算架构上下文

云计算架构应用上下文的相关角色包括：云租户/服务消费者、云应用开发者、云服务运营者/提供者、云设备提供者(见图2-2)。

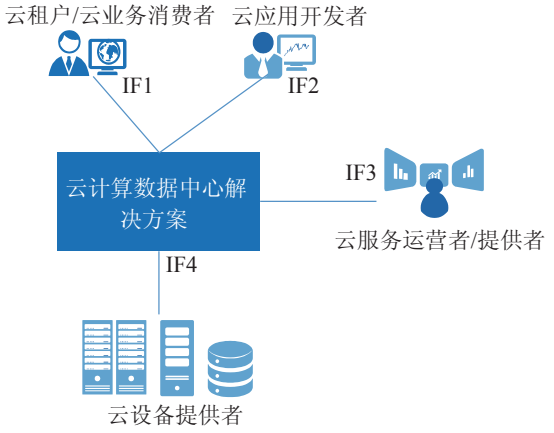


图2-2 云计算系统架构上下文

一、云租户/云服务消费者

云租户是指这样一类组织、个人或IT系统，该组织/个人/IT系统消费由云计算平台提供的业务服务(比如请求使用云资源配额，改变指配给虚拟机的CPU处理能力，增加Web网站的并发处理能力等)组成。该云租户/云业务消费者可能会因其与云业务的交互而被计费。

云租户也可被看做是一个云租户/业务消费者组织的授权代表。比如说一个企业使用到了云计算业务，该企业整体上相对云业务运营及提供者来说是业务消费者，但在该业务消费者内可能存在更多的细化角色，比如使得业务消费得以实施的技术人员，以及关注云业务消费财务方面的商务人员等。当然，在更为简化的公有云场景下，这些云业务消费者的角色关系将简化归并到一个角色。

云租户/业务消费者在自助Portal上浏览云服务货架上的服务目录，进行业务的初始化以及管理相关操作。

就多数云服务消费者而言，除从云服务提供者那里获取到的IT能力之外，也同时继续拥有其传统(非云计算模式)IT设施，这使得云服务与其内部既有的IT基础设施进行集成整合至关重要，因此特别需要在混合云的场景下引入云服务集成工具，以便实现既有IT设施与云服务之间的无缝集成、能力调用以及兼容互通。

二、云应用开发者

云应用开发者负责开发和创建一个云计算增值业务应用，该增值业务应用可以托管在云平台运营管理者环境内运行，或者由云租户(服务消费者)来运行。典型场景下云应用开发者依托于云平台的API能力进行增值业务的开发，但也可能会调用由BSS和OSS系统负责开放的云管理API能力(云应用开发者当然也可能选择独立构建其独立于云平台的增值业务应用系统的BSS/OSS系统，而不调用或重用底层的云管理API)。

云业务开发者全程负责云增值业务的设计、部署并维护运行时主体功能及其相关的管理功能。如同云租户/云业务消费者以及云业务运营提供者一样，云业务开发者也可以是一个组织或者个人，比如一个开发云业务的ISV开发商是一个云业务开发者，其内部可能包含了上百个担任不同细分技术或商业角色的雇员。另外，负责云业务管理的运维管理人员与负责开发云业务的开发组织紧密集成也是一种常见的角色组织模式(比如Google、Amazon、百度等自营加自研的互联网DevOps服务商)，这是提升云业务发放和上线效率的一种行之有效的措施，因为此类角色合一的

模式提供了更短的问题反馈路径，使得云业务的运营效率有了进一步实质性提升。

目前云计算业务开发者在公有云及私有云领域的典型应用包括：运营商虚拟主机出租与托管云、企业内部IT私有云或专有云、桌面私有云、运营商桌面云服务、企业网络存储与备份云、视频媒体处理云、IDC Web托管及CDN云以及大数据分析云等。

三、云服务运营者/提供者

云服务运营者/提供者承担着向云租户/服务消费者提供云服务的角色，云服务运营者/提供者概念的定义来源于其对OSS/BSS管理子系统拥有直接的或者虚拟的运营权。同时作为云服务运营者以及云服务消费者的个体，也可以成为其他对外转售云服务提供者的合作伙伴，消费其云服务，并在此基础上加入增值，并将增值后的云服务对外提供。当然，云服务运营者组织内部不排除有云业务开发者的可能性，这两类决策既可在同一组织内共存，也可相对独立进行。

四、云设备/物理基础设施提供者

云设备提供者提供各种物理设备，包括服务器、存储设备、网络设备、一体机设备，利用各种虚拟化平台，构筑成各种形式的云服务平台。这些云服务平台可能是某个地点的超大规模数据中心，也可能是由地理位置分布的区域数据中心组成的分布式云数据中心。

云设备提供者可能是云服务运营者/提供者，也可能就是一个纯粹的云设备提供者，他将云设备租用给云服务运营者/提供者。

在这里我们特别强调云设备/物理基础设施的提供者必须能够做到不与唯一的硬件设备厂家绑定，即在云计算系统平台南向接口上所谓的多厂家硬件的异构能力。

五、接口说明

从上述云计算的基础上下文描述，我们不难看出云平台和云运营与运维管理系统是介于上层多租户的IT应用、传统数据中心管理软件，以及下层数据中心物理基础设施层之间的一层软件。其中云平台可进一步被分解为面向基础设施整合的云操作系统，面向数据整合的大数据引擎，以及面向应用中间件整合的应用开放平台。而云运营与运维管理系统在云计算引入的初期，与传统数据中心管理系统是并存关系，最终将逐步取代传统数据中心管理。

云平台的南向接口IF4向下屏蔽底层千差万别的物理基础设施层硬件的厂家差异性。针对应用层软件以及管理软件所提出的基础设施资源、数据处理以及应用中间件服务诉求，云平台系统向上层多租户的云应用与传统数据中心管理软件屏蔽如何提供资源调度、数据分析处理，以及中间件实现的细节，并在北向接口IF1、IF2和IF3为上层软件及特定租户提供归一化、标准化的基础设施服务(IaaS)、数据处理及应用平台服务(PaaS) API服务接口。在云平台面向云运营与管理者(拥有全局云资源操作权限)的IF3接口，除了面向租户的基础设施资源生命周期管理API之外，还包括一些面向物理、虚拟设施资源及云服务软件日常OAM运行健康状态监控的操作运维管理API接口。

其中IF1/IF2/IF3接口中关于云租户感知的云平台服务API的典型形态为Web RESTful接口。IF4接口则为业务应用执行平面的x86指令，以及基础设施硬件特有的、运行在物理主机特定类型OS中的管理Agent，或者基于SSL承载的OS命令行管理连接。IF3接口中的OAM API则往往采用传统IT和电信网管中被广泛采用的Web RESTful、SNMP、CORBA等接口。

2.1.2 云计算的典型技术参考架构

基于上述分析，我们不难得出如下的云计算数据中心架构分层概要(见图2-3)。

一、云平台IT基础设施架构层

自20世纪50年代前第一台电子计算机诞生至今，IT基础设施先后经历了大规模集中式计算的大型机/小型机，到小规模分布式的个人计算PC机和小规模集中化的B/S、C/S客户端服务器架构，再到大规模集中化的云计算，从合并、分

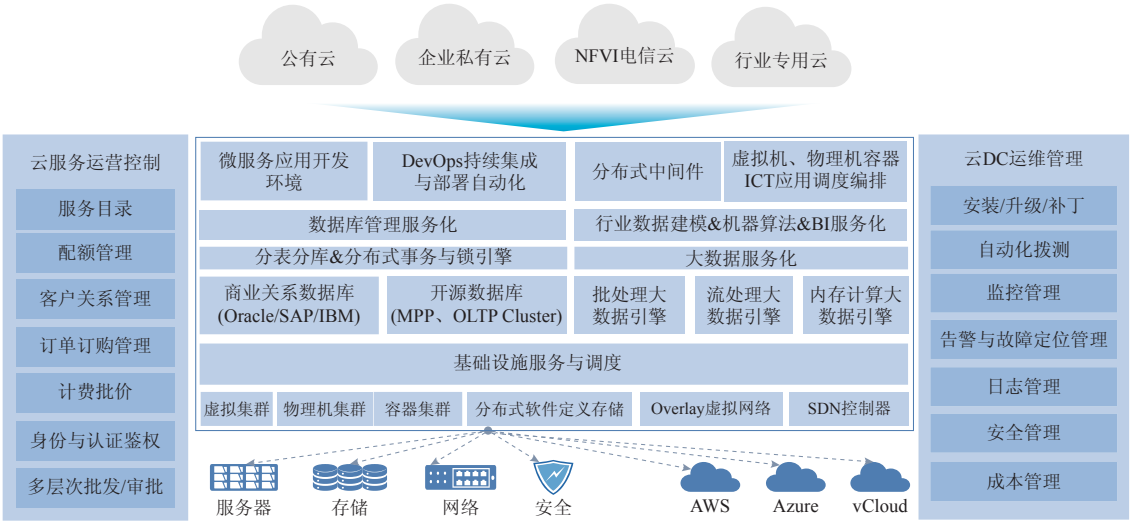


图2-3　云计算数据中心解决方案端到端总体分层架构

离，再重新走向融合。这个阶段的融合，借助虚拟化及分布式云计算调度管理软件，将IT基础设施整合成为一个规模超大的“云计算机”，相当于建成了一座“基础设施电厂”。多个租户可以从这座电厂中随时随地获取到其所需的资源，从而大大提升了业务敏捷度，降低了TCO消耗，甚至提供了更优的业务性能与用户体验。

然而，历史总是在螺旋式前进的，“云计算机”看似大型机，但绝非简单回到了大型机时代。

1. 区别1：架构不同，规模扩展能力不同

由“垂直扩展”到“横向扩展”；计算处理能力，存储容量，网络吞吐能力，租户/应用实例数量，均相差n个数量级以上，从硬件成本视角来看，TCO成本也更低。

2. 区别2：硬件依赖性不同，生态链、开放性不同

由“硬件定义”到“软件定义”；对于早期的IT系统，少数硬件厂家绑定OS和软件，IT只是少数用户的奢侈品。在新的时代，通过软件屏蔽异构硬件差异性，同一个硬件平台上，可以运行来自多个不同厂家的软件和OS。新时代的IT生态链更加繁荣，IT成为人人消费得起的日用品。

3. 区别3：可靠性保障方式不同

由“单机硬件器件级的冗余实现可靠性”发展到“依赖分布式软件和故障处理自动化实现可靠性，甚至支持地理级容灾”。

4. 区别4：资源接入方式不同

将IT基础设施能力比作“电力”，大型机只能专线接入，是只能服务于少数人群的 “发电机”，基于企业以太网或者互联网的开放接入，是可以为更多人群提供服务的“发电站”和“配电网”。

基础设施层又可进一步划分为物理资源层、虚拟资源层，以及资源服务与调度层。

1. 物理资源层

所有支撑IaaS层的IT基础设施硬件，其中包括服务器、存储(传统RAID架构垂直扩展的Scale Up存储和基于服务器的分布式水平扩展的Scale Out存储)，以及数据中心交换机(柜顶、汇聚以及核心交换)、防火墙、VPN网关、路由器等网络安全设备。

2. 虚拟资源层

虚拟资源层在云计算架构中处于最为关键与核心的位置。该层次与“资源服务与调度层”一道，通过对来自上层操作系统及应用程序对各类数据中心基础设施在业务执行和数据平面上的资源访问指令进行“截获”。指令和数据被截获后进行“小聚大”的分布式资源聚合处理，以及

“大分小”的虚拟化隔离处理，以及必要异构资源适配处理。这种处理可以实现在上层操作系统及应用程序基本无须感知的情况下，将分散在一个或多个数据中心的数据中心基础设施资源进行统一虚拟化与池化。

在某种程度上，虚拟资源层对于上层虚拟机(含操作系统及应用程序)的作用与操作系统对于应用软件的支撑关系是类似的，实质上都是在多道应用作业实例与底层的物理资源设备或者设备集群之间进行时分和空分的调度，从而让每道作业实例都“感觉”到自己是在独占相关资源，而实际上资源在多个作业实例之间的复杂、动态的复用调度机制完全由虚拟资源层屏蔽。技术实现的主要困难与挑战在于，操作系统的管理API是应用程序感知的，而虚拟资源层则必须做到上层操作系统与应用程序的“无感知”，同时如何对于频繁的指令级陷入和仿真调度助力，做到令上层应用及OS可接受的性能开销。

虚拟资源层又包括三个部分，具体如下。

(1) 计算虚拟化

所有计算应用(含OS)并非直接承载在硬件平台上，而是在上层软件与裸机硬件之间插入了一层弹性计算资源管理及虚拟化软件：弹性计算资源管理软件对外负责提供弹性计算资源服务管理API，对内负责根据用户请求调度分配具体物理机资源；虚拟化软件(Hypervisor)对来自所有的x86指令进行截获，并在不为上层软件(含OS)所知的多道执行环境并行执行“仿真操作”，使得从每个上层软件实例的视角，仍然是在独占底层的CPU、内存以及I/O资源(见图2-4)；而从虚拟化软件的视角，则是将裸机硬件在多个客户机(VM)之间进行时间和空间维度的穿插共享(时间片调度、页表划、I/O多队列模拟等)。由此可见，计算虚拟化引擎本身是一层介于OS与硬件平台的中间附加软件层，因此将不可避免地带来性能上的损耗。然而随着云计算规模商用阶段的到来，以及计算虚拟化的进一步广泛普及应用，越来越多的计算性能敏感型和事务型的应用逐步被从物理机平台迁移到虚拟化平台之上，因此对进一步降低计算虚拟化层的性能开销提出了更高的要求，典型的增强技术包括以下内容。

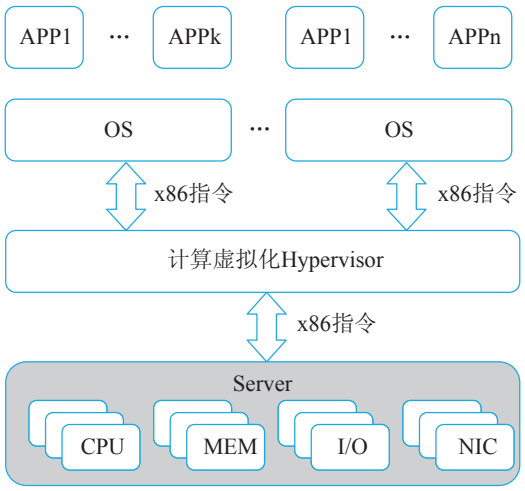


图2-4　计算虚拟化硬件接口

✍ 虚拟化环境下更高的内存访问效率：应用感知的大内存业务映射技术，通过该技术，可有效提升从虚拟机线性逻辑地址到最终物理地址的映射效率。

✍ 虚拟化环境下更高的CPU指令执行效率：通过对机器码指令执行的流程进行优化扫描，通过将相邻执行代码段中的“特权”指令所触发的“VM_Exit”虚拟化仿真操作进行基于等效操作的“合并”，从容达到在短时间内被频繁反复地执行。由于每次VM_Exit上下文进入和退出的过程都需要涉及系统运行队列调度以及运行环境的保存和恢复，即将多次上下文切换合并为一次切换，从而达到提升运行效率的目的。

✍ 虚拟化环境下更高的I/O和网络包收发处理效率：由于多个虚拟机在一个物理机内需要共享相同的物理网卡进行网络包收发处理，为有效减少中断处理带来的开销，在网络及I/O发包过程中，通过将小尺寸分组包合并为更大尺寸的分组包，可以减少网络收发接受端的中断次数，从而达到提升虚拟机之间网络吞吐率的目的。

✍ 更高的RAS可靠性保障：针对云计算所面临的电信领域网络及业务云化的场景，由于硬件故障被虚拟化层屏蔽了，使得物理硬件的故障

无法像在传统物理机运行环境那样直接被传送通知给上层业务软件，从而导致上层业务层无法对故障做出秒级以内的及时响应，比如业务层的倒换控制，从而降低了对整体可靠性水平。如何感知上层的业务要求，快速进行故障检测和故障恢复，保证业务不中断，这给计算虚拟化带来了新的挑战。

(2) 存储虚拟化

随着计算虚拟化在各行业数据中心的普遍采用，x86服务器利用效率提升中已获得普遍应用的同时，人们发现存储资源的多厂家异构管理复杂、平均资源利用效率低下，甚至在I/O吞吐性能方面无法有效支撑企业关键事务及分析应用对存储性能提出的挑战，通过对所有来自应用软件层的存储数据面的I/O读写操作指令进行“截获”，建立从业务应用视角覆盖不同厂家、不同版本的异构硬件资源的统一的API接口，进行统一的信息建模，使得上层应用软件可以采用规范一致的、与底层具体硬件内部实现细节解耦的方式访问底层存储资源。

除去带来硬件异构、应用软件与硬件平台解耦的价值之外，通过“存储虚拟化”层内对多个对等的分布式资源节点的聚合，实现该资源的“小聚大”。比如，将多个存储/硬盘整合成为一个容量可无限扩展的超大(EB级规模)的共享存储资源池。由此可以看到，存储虚拟化相对计算虚拟化最大的差别在于：其主要定位是进行资源的“小聚大”，而非“大分小”。原因在于，存储资源的 “大分小”在单机存储以及SAN/NAS独立存储系统，乃至文件系统中通过LUN划分及卷配置已经天然实现了，然而随着企业IT与业务数据的爆炸式增长，需要实现高度扁平化、归一化和连续空间，跨越多个厂家服务器及存储设备的数据中心级统一存储，即“小聚大”。存储“小聚大”的整合正在日益凸显出其不可替代的关键价值(见图2-5)。

- 高性能分布式存储引擎：伴随着云计算系统支撑的IT系统越来越大，覆盖范围从不同服务器存储节点，到分布在不同地理区域的数据中心，这就需要有一个分布式存储引擎。这个引擎，能满足高带宽、高I/O等各种场景要求，能很好地进行带宽的扩展。
- 存储异构能力：如何利旧，将不同厂家原有的独立SAN、NAS设备组合成一个大的存储资源池，也是软件定义存储中需要解决的问题。
- 存储卸载：传统的企业存储系统，在采用各种各样的存储软件，这些软件存储操作对存储I/O和CPU资源均有较大消耗，会影响到用户业务性能的发挥。因此，如何将存储操作标准化，然后将存储操作利用某些标准的硬件动作去代替，这就是存储卸载。

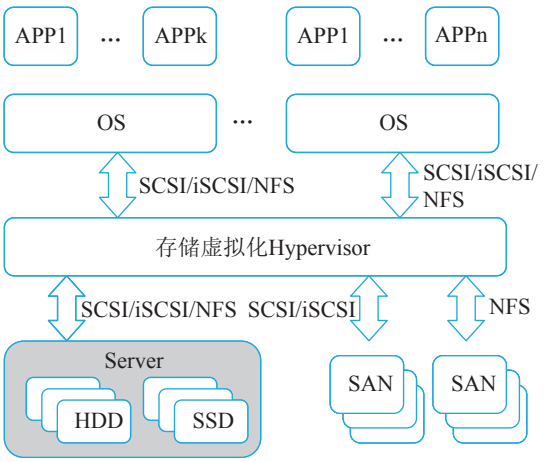


图2-5 存储虚拟化硬件接口

(3) 网络虚拟化

站在操作系统角度，OS管理的资源范畴仅仅是一台服务器，而Cloud OS管理的资源范畴扩展到了整个数据中心，甚至将跨越多个由广域网物理或者逻辑专线连接起来数据中心。在一台服务器内，核心CPU、内存计算单元与周边I/O单元的连接一般通过PCI总线以主从控制的方式来完成，多数管理细节被Intel CPU硬件及主板厂家的总线驱动所屏蔽，且PCI I/O设备的数量和种类有限，因此OS软件层面对于I/O设备的管理是比较简单的。相对而言，在一个具备一定规模的数据中心内，甚至多个数据中心内，各计算、存储单元之间以完全点对点的方式进行松耦合的网络

互联。云数据中心之上承载的业务种类众多，各业务类型对于不同计算单元(物理机、虚拟机)之间，计算单元与存储单元之间，乃至不同安全层次的计算单元与外部开放互联网网络和内部企业网络之间的安全隔离及防护机制要求动态实现不同云租户之间的安全隔离。云数据中心还要满足不同终端用户不同场景的业务组网要求以及他们的安全隔离要求。因此，云操作系统的复杂性将随着云租户及租户内物理机和虚拟机实例的数量增长呈现几何级数的增长，由业务应用驱动的数据中心网络虚拟化和自动化已变得势在必行和不可或缺。为了实现彻底与现有物理硬件网络解耦的网络虚拟化与自动化，唯一的途径与解决方案就是SDN(也即所谓软件定义的网络)，即构建出一个与物理网络完全独立的叠加式逻辑网络，其主要部件以及相关技术包括以下几方面。

- SDN控制器：这是软件定义网络的集中控制模块。负责云系统中网络资源的自动发现和池化、根据用户需求分配网络资源、控制云系统中网络资源的正常运行。
- 虚拟交换机：根据SDN控制器，创建出的虚拟交换机实例。可以对这个虚拟交换机进行组网的设计、参数的设置，一如对物理交换机的使用。
- 虚拟路由器：根据SDN控制器，创建出的虚拟路由器实例。可以对这个虚拟路由器进行组网的设计、参数的设置，一如对物理路由器的使用。
- 虚拟业务网关：根据用户业务的申请，由SDN控制器创建出虚拟业务网关实例，提供虚拟防火墙的功能。可以对这个虚拟业务网关进行组网的设计、参数的设置，一如对物理业务网关的使用。
- 虚拟网络建模：面对如此复杂多变的组网，如何保证网络的有效区分和管理，又能保证交换和路由的效率，一个有效的建模方法和评估模型是需要的。虚拟网络建模技术能提前预知一个虚拟网络的运行消耗、效率和安全性。虚拟网络建模可以做成一个独立功能库，在需要的时候启动，以减少对系统资源的占用。

3. 资源服务与调度层

相对虚拟化层在业务执行面和数据面上“资源聚合与分割仿真”，该层次主要体现为管理平面上的“逻辑资源调度”。

由于多个厂家已经投入到云计算的研发和实施中，不可避免地有多种实现方式。而要实现云计算真正的产业化并被广泛使用，各厂家的云计算平台必须要能够互相交互，即进行接口标准化。接口标准化后，主流的虚拟化平台，例如Hyper-V、KVM、UVP、ESX等之间能够互相兼容。各个硬件厂家或者中间件厂家可以自由选择虚拟化内核。

在云计算新的发展阶段中，面向公有云、面对国际化公司的分布式云系统将是重点。这样引发对超大资源的分配和调度。在整个云计算的实现架构上，计算、存储、网络资源的分配和使用将走向专业化。这是因为一个云应用业务，根据性质的不同，它对计算、存储、网络资源的需求可能是不一样的。例如：呼叫中心业务偏向于计算资源使用，而对于网盘业务则偏向于存储资源使用。在这种情况下，为了更有效地利用资源，给业务层提供基本资源调用API是最好的选择，将计算、存储、网络资源都作为基本资源单位，提供统一的资源调用接口，让云业务开发者自己选择如何高效地使用这些资源。这些API包括以下几个方面。

- 弹性计算资源调用API：计算资源包括CPU和内存，云计算平台根据云运营商的要求，已经将CPU和内存虚拟化和池化。系统提供资源的动态申请、释放、故障检测、隔离和自动切换功能，做到业务不感知。CPU资源又可以分为纯计算型、图像处理型等不同类型。不管是CPU还是内存，都提供瘦分配功能，资源的自动伸缩保证在低业务量时减少资源的消耗，高业务量时开启所有物理资源，确认业务的高效运行。计算资源API还需要提供集群能力。
- 弹性存储资源调用API：存储资源API提供文件或者卷接口，除了提供常见的资源申请、释

放、瘦分配等功能外，还涉及其他几个关键方面。

- 异构资源的池化：不同的厂家在将存储资源池化后，提供统一的API，一个厂家可以利用这些API，将不同厂家的存储资源池构成一个大的资源池，然后再封装出API供业务调用。
- 存储资源的分层分级存储：因业务性能要求的不同，分层存储是一个常用的技术，业务系统在申请存储资源的时候，可以选择是否使用这个特性。
- 内存存储资源的支持：未来的系统，内存一定会成为主存，所有的存储，除非一些特别重要的信息，基本上不再需要存入非易失性介质。而使用内存资源作为主存，可靠性是关键要求。在构造内存存储池的时候，可靠性必须贯彻始终，每个内存存储在其他地方有备份，或者确保内存存储有可靠的UPS保护。

弹性网络资源调用API：网络资源API的基本功能也包括资源的申请、释放、监控、故障隔离和恢复等，也需要考虑异构资源的统一化。

二、云平台大数据引擎层

数据服务层是叠加在基础设施服务之上，具备多租户感知能力的结构化、半结构化及非结构化数据服务的能力。

结构化数据服务子层提供对结构化数据的存储和处理功能，它通过叠加各种结构化数据库软件来实现，例如常见的Oracle\Sybase\HANA等。为提高处理效率，弹性存储资源调度层会针对不同的基于磁盘或者基于内存的数据库，提供更高效的存储资源调用API。例如面向HANA内存数据库，提供内存专用的存储资源调用API接口。

非结构化数据服务主要是叠加常见的NoSQL数据库的功能模块，例如Map-reduce、HBase等，提供弹性存储资源的特殊接口。

流数据服务更多地涉及对特殊CPU资源和专用芯片资源的使用。在弹性计算资源API中提供一些专用接口，来进行流数据的高效输入、压缩、解压缩、处理和转发。

传统IT系统中，软件业务处理逻辑总是位于端到端软件栈的核心，数据是作为业务处理逻辑可持久化的后端支撑，由此出现了数据库引擎技术及SQL标准化查询语言，用来进行可靠、高效的关系型数据创建、更新、存储和检索。由于数据库领域的算法难度大、专业性强，一直以来可商用数据库均由Oracle、IBM、Sybase等少数几家厂家所垄断。随着开源数据库技术的不断成熟和发展，特别是大数据与分布式数据库技术方兴未艾的发展，以及基于大量基础数据的机器学习算法不断取得突破，并在互联网搜索、电商和广告领域获得广泛应用，打破了数据总是作为独立应用的附属支撑的定位，使得之前被事务处理逻辑边界限制的数据孤岛，有可能通过ETL数据抽取和汇总机制，被大规模集中存储起来，并进行大规模的横向跨数据源、数据集(OLTP)，乃至跨越较长时间跨度的内生关联关系与价值信息的抽象分析提取与挖掘分析，原先需要通过昂贵的软件License及专业支持才能实现的数据汇总分析(OLAP)，通过开源软件即可实现。

三、云应用开发部署及中间件层

传统企业IT数据中心中，J2EE、微软.NET、IBM WebSphere、Oracle WebLogic等是被普遍采用企业应用开发平台及中间件(如消息队列、缓存、企业集成总线等)，然而随着全球互联网化、移动化等大趋势对业务创新能力以及快速响应客户需求的挑战不断加剧，传统企业中间件在开放互通性、水平扩展能力、支撑快速敏捷迭代开发等方面越来越无法满足业务支撑的需要，与此同时，以Kubernetes、Mesos、Coudify等为代表面向DevOps敏捷开发的开源应用与部署开发工具链与平台，具备分布式水平扩展能力的系列开源数据库和中间件，如MySQL/PostgreSQL/MangoDB、RabbitMQ/Kafka消息队列、Redis缓存，以及业务流行的Spring for Java、Ruby on Rails、Sinatra、Node.js、Grails、Scala on Lift、PHP及Phython等，在近年来获得快速成熟发展，且体现出相比

传统企业中间件的几个关键优势。

- 开放性、标准化：基于开源，应用开发平台的管理API更为开放透明，同时也引入了容器化技术进行应用部署，使得应用实例的部署不再与编程语言(如Java)绑定。
- 轻量化：相比闭源软件，Web中间件自身的资源消耗大幅减少，容器化应用部署相比虚拟机模式更加轻量化、敏捷化。
- 分布式及弹性扩展：与数据库层扩展能力配合，提供负载均衡及弹性伸缩控制基础框架机制的支撑，使Scale-Out应用架构可聚焦于应用逻辑本身，开发更加轻松高效。
- 敏捷开发与上线部署：支持从开发、集成、测试验证到生产上线的全流程自动化环境置备及测试自动化，配置随同应用一起发布，任何生产环境、任何开发集成部署节点皆可一键式快速重用，而无须烦琐部署配置流程。

四、云服务运营控制

云服务运营控制系统的主要服务对象是云服务产品定义、销售与运营人员，针对上述基础设施层、大数据层以及应用开发部署层的云服务产品，当然不是云服务运营者无偿提供给租户和用户使用的，需要引入“云服务运营控制”子系统来负责建立云服务产品在供应者和消费者之间的线上产品申请、受理及交付控制，以及完成与信息交付服务等值的货币交换。该系统以可订购的服务产品的形式在服务目录上呈现各层丰富多样的云服务产品，同时可基于云资源及云服务的实际使用量、使用计次及使用时长的消费记录，或简单按照包年包月的模式进行计量和计费，从而将企业获取IT产品和服务的商业和交付模式真正从买盒子、买上门人工服务支持(CAPEX)这样直接在线获取转变为按需在线购买(OPEX)的模式。一方面充分保障了面向云租户与消费者的经济性、敏捷性效益，另一方面也让公有云服务运营者从外部租户的规模经济效益中获利，并不断提升其服务质量；而私有云运营者也可有效依据计量计费信息来核算内部各租户、各部门对云服务产品的消费情况，并给出预算优化建议。

因此，对于公有云来说，“云服务运营控制”系统，进一步包含了租户身份认证鉴权管理、客户关系(CRM)管理、订购管理、产品定义与服务目录、促销与广告、费率管理、批价与信用控制、计费计量等一系列功能模块；而除了与公有云场景共享的租户身份认证鉴权、产品定义和服务目录、计量计费功能之外，与企业内部多层级部门组织结构相匹配的资源配额与服务封装和服务目录定制，服务申请审批流程，与ITIL系统的对接等，则是私有云服务运营过程需要解决的独特性问题。

无论是公有云还是私有云，为广泛引入第三方ISV的软件加入自己的云平台生态系统，将第三方软件与自建的云平台相结合，并进一步实现对自研和第三方云服务能力的封装组合及配置部署生命周期管理的工作流及资源编排自动化管理，引入了所谓Markeplace应用超市及XaaS业务上线系统，使得未来在IaaS/PaaS平台能力基础上引入的第三方业务上线部署、配置以及测试过程从原来多次重复的人工干预过程，转变为一键式触发的全自动化过程，从而大幅提升第三方应用软件在公有云、私有云平台上实现上线过程的自动化和敏捷化效率。

五、云DC运维管理

云DC运维管理子系统，目标是服务于云DC的运维管理人员的日常运维管理，包含云平台与应用软件的安装部署与升级补丁，虚拟及物理基础设施的监控与故障管理、日志管理、自动化仿真测试、安全管理，成本管理等功能模块和内部服务，通过管理工具支撑运维人员对系统运行的异常事件及健康状态进行快速、高效的响应处理，从而保障面向最终租户提供的云服务可靠性、可用性、性能体验等SLA属性达到甚至超出承诺的水平。

1. 面向DevOps敏捷开发部署的软件安装与升级自动化及业务连续性保障

在管理控制平面上，云DC相对于传统DC运维体系带来的一个显著变化，就是各云平台、云服务，以及云运维管理软件的架构，从原来厚重

的模块化、服务总线式架构，演进到被拆解分离的多个轻量化、运行态解耦的微服务架构，各微服务间仅有REST消息交互，没有任何数据库、平台组件的实例化共享依赖，从而实现了在线模式下各微服务的独立安装、灰度升级，数以百计的各个不同版本的云平台、云服务以及云管理运维微服务，只需保证升级时间窗内多个共存版本的服务接口契约语义级与功能级兼容性，以及各自上线前的预集成验证工作充分到位，无须做端到端系统测试，即可基于敏捷开发、集成与部署的支撑工具流水线，实现各自独立版本节奏的升级更新与上线发布，并在新上线发布的观测期期间一旦发现问题，具备快速升级回退的能力。

在数据平面上，由于租户的所有业务应用运行承载在每台服务器的虚拟化引擎及分布式存储和软件定义网络平台的基础上，因此为保障在对数以万计到百万计的资源池节点上的虚拟化引擎，分布式存储及软件定义网络，以及其管理代理节点进行软件升级的过程中，租户业务中断最小化，甚至是零中断，首先在工程上必须将此类资源池基础平台类软件的升级分批执行，其次也必须支持完善的虚拟化和分布式软件系统的热补丁机制，以及必要的跨物理节点热迁移机制，从而最大限度地降低升级过程中系统平台重启动带来的对云租户可感知的业务中断影响。此外，在热补丁无法完全覆盖、重启租户资源池服务器的场景下，则需要考虑引入对云租户可见的故障域的概念，引导用户将具备主备、负荷分担冗余能力的应用负载部署在不同的故障区域内，从而实现在不对故障域做并行升级的前提下，业务基本不中断，或业务中断时间仅取决于业务应用的主备切换或负荷分担切换的时延。

2. 智能化、自动化的故障与性能管理

与传统DC高度精细化的人工干预管控及治理模式不同，云DC运维管理最为显著的差异化特点是管理对象的数量、规模及复杂度均呈现指数级增长，因此对运维管理的自动化、智能化提出了更高的要求。从上述系统架构描述不难看出，无论是公有云还是私有云，一个功能完整的云DC软硬件栈系统是由基础设施层(进一步分解为虚拟化和软件定义存储与软件定义网络构成的数据平面，以及标准化云服务管理层)、数据层，以及应用开发部署平台与中间件层等多个SOA解耦的微服务、软件组件所构成的，具有很高的系统复杂度；同时云DC中的分布式、规模和数量庞大的基础设施(对于公有云及大型私有云，服务器数量往往可达数万到数十万、百万规模)及各类系统云服务及租户的业务应用负载数量，也达到了数以百万乃至千万级的程度，对网络与安全工程组网也提出了巨大挑战。因此，为应对上述挑战，将云DC运维管理员从传统DC烦琐低效的人工干预、保姆式管理监控与故障处理中解放出来，转向尽可能无人干预的自动化、智能化的运维管理模式，将人均维护管理效率从平均每人数十台服务器，提升到平均每人数千台服务器，需要解决的关键问题及其解决方案包括以下几点。

- 基于工作流的自动化人工故障修复机制：通过基于最佳运维实践预定义的工作流驱动，或者依据长期积累的故障模式库来驱动自动化运维工作流进行监测及无人干预或基于事件告警通知的一键式修复。
- 基于日志和监控信息的跨微服务、跨系统的制定业务流程和租户用户追踪与关联分析，用以解决客户报障和主动告警场景下的快速故障定位：传统数据中心中，各软硬件系统的日志监控信息往往相对零散孤立，没有实现与业务和用户的自动关联，因此难以适应复杂度更高的云数据中心故障管理的需要。
- 基于大数据机器学习引擎的大规模运维场景下的性能与故障规律分析、趋势预测及故障根因识别定位：传统数据中心的故障发现与修复建议的处理，主要依赖于运维管理系统收集监控日志信息，以及运维团队长期积累的历史经验总结出来的典型故障模式。但随着云数据中心管理维护对象的数量级增长，以及系统复杂度的持续迭代提升，导致基于人工经验及故障模式积累的维护效率终将难以跟上公有云及大型私有云业务规模快速扩张发展的步伐，因此需要引入大数据

机器学习机制，对历史积累的海量故障和监控信息进行围绕特定主题的关联分析，从而自动化地、智能化地挖掘出更多高价值的、运维人员认知范围外的故障模式与系统优化模式，从而进一步提升系统运维的效率。

3. 生命周期成本管理

除上述挑战之外，由于公有云、私有云面向云租户与云服务消费者的基础设施服务，是一个需要持续投入服务器和网络与安全硬件的重资产经营过程，因此如何保证运营过程中的硬件资产投入产出比与经济效益，如何进行精细化的成本管理，就成为一个至关重要的问题。在传统数据中心中，所有硬件资产都是以配置库的形式进行编号管理，用户与硬件基本是固定对应的关系，然而在这些硬件资产资源池化和多租户自动化之后，硬件通过虚拟化和跨节点调度机制在多个租户间动态共享，而云服务界面上甚至也会参照QoS/SLA的要求，面向租户提供超出硬件实际供给能力的资源配额，这些硬件资源是否在云资源池的共享环境下得到了高效的使用，硬件的平均空置率，或者说应用和租户对硬件资源的实际使用效率是否达到了理想水平(比如高于80%的实际使用率，或低于20%的空置率)，是否需要对资源分配算法、超分配比率做出及时调整，将对公有云的可持续运营利润水平，以及私有云、混合云的成本控制效率，将产生决定性的影响。

2.1.3 云计算的服务及管理分层分级架构

面向云租户、云服务消费者的服务分层分级视图，如图2-6所示。

一朵云划分为多个服务区域(Region)，每个服务区域对应一组共享的IaaS/PaaS/SaaS云服务实例，不同服务区域可能有不同的服务产品目录，以及体现云服务对区域本地化需求的考虑，如本地化的服务订购Portal，虚拟机及云服务的区域化定制选项等。

公有云面向全球的用户提供云服务，因此往往设计为跨多个服务区域，每个服务区域部署一套云服务，但多个服务区域共享同一套租户认证鉴权管理系统(SSO一次性登陆鉴权)，这样租户一旦成功登录鉴权后，即可在不同服务区域间随意按需切换，而无须重新鉴权。

一个服务区域由2个或2个以上可用性区域(AZ)组成，每个可用性区域面向租户呈现为一个地理上独立的可靠性保障区域，该区域一般依赖于相同的数据中心层1基础设施，比如数据中心电源供给、UPS非间断电源。

一般情况下，租户会指定将其申请的云资源及应用发放部署在哪个AZ内，对于分布式应用，则可选择应用跨AZ部署，以实现更高层次的地理容灾可靠性保障。

在租户签约虚拟机或容器服务的情况下，可以看到隶属于该租户的每台虚拟机或每个容器实例；在租户签约物理机的情况下，可以看到每台物理机实例，除此之外的资源池集群，物理数据中心等概念均对租户不可见。

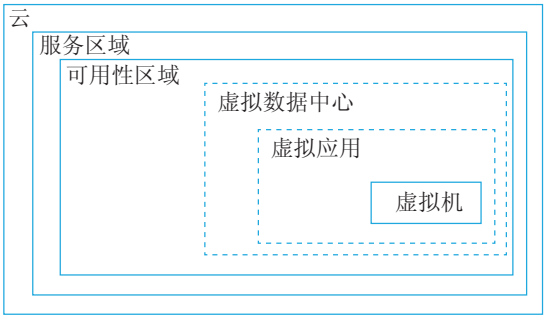


图2-6 面向云租户、云服务消费者的服务分层分级视图

面向云数据中心运维人员的管理分层视图，如图2-7所示。

每个服务区域(Region)下的可用性区域(AZ)设计部署对于云DC管理员直接可见，每个Region到租户/最终用户侧的接入时延一般推荐在100 ms的范围内。

服务区域(Region)内各可用性区域(AZ)间的网络传输时延迟一般控制在10ms的范围内，一个可用性区域一般由一个多个物理数据中心构成，物理数据中心层仅对运维管理员可见，对普通租户/用户不可见。

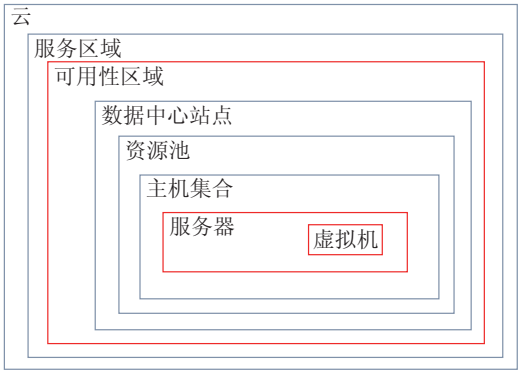


图2-7　面向云数据中心运维人员的管理分层视图

✍ 数据中心内物理网络传输时延一般在1ms范围以内，由一个或多个层2网络或层3子网构成。

✍ 每个物理数据中心内可以部署一个或多个资源池集群(POD)，每个资源池集群(POD)对应一个云资源池调度管理系统实例(如OpenStack)，每个资源池集群包含的服务器规模一般是数千到数万台，其中包含了用于承载租户业务应用负载的计算集群，以及用来承载租户数据的存储集群。

✍ 每台虚拟机缺省情况下直接接入到基础云资源调度管理系统，但对于异构的传统虚拟化集群，也可整体作为一台逻辑“大主机”接入到资源池调度管理系统。

✍ 在同一物理资源池集群范围内，考虑到云租户对服务器硬件需求的特殊性(比如对于GPU加速硬件、SR-IOV网卡、不同工作频率和数量的CPU、不同容量的内存、不同类型的虚拟化集群，乃至裸金属物理机集群等)，均需要对隶属于同一资源选择属性的服务器进行标签(Tagging)，隶属于同一资源属性标签的服务器构成一个“主机集合”，同一服务器主机可被标记为多个资源属性标签，即可以隶属于多个“主机集合”。由此可见，“主机集合”是一个物理资源池集群范围内用来划分与动态资源调度相关的，基于特定资源池属性维护来划分的“逻辑资源池”概念，该“主机集合”的标签条件，将与云资源池调度管理软件API入口制定的动态调度参数进行匹配，从而决定当前的资源发放申请被调度到哪些“主机集合”内。

分布式数据中心

对于新建的公有云数据中心，一般会选择尽量在选定的数据中心内集中式部署大规模资源池，即在云资源调度软件能力可以支撑的范围内，尽量扩大整合资源池的管控范围和规模。然而，由于部分大企业云租户对部分关键信息资产上公有云的安全顾虑，公有云也可选择将部分可用性区域及资源池集群建在远端的企业数据中心内，或者物理上与公共资源池完全隔离的托管区，此时该AZ/POD的规模往往规模较小，仅有几十到上百台服务器的规模，称之为“微型数据中心”(Micro DC)，但同样可通过与公有云大规模集中管理数据中心之间的VPN/MPLS广域网络连接实现统一的资源管控与调度，并保障该租户特有的敏感应用负载及数据仅存放于该租户专属的物理托管区域内。

对于新建的私有云数据中心，考虑到对现有分散部署的虚拟化及物理资源池的继承和利旧需求，以及部分终端用户接入体验敏感的应用负载就近接入的需求(比如虚拟桌面、视频交互应用等)，也会提出从大集中数据中心拉远的分布式多站点的可用性区域/资源池集群的需求。

2.1.4　拉通公有云与私有云的混合云架构

如上面几个章节所讨论，公有云服务因其大规模集约化的优势，在弹性、敏捷性以及无须固定硬件投资、按需资源及服务申请和计费的成本优势方面，获得了广大企业用户，特别是中小企业用户的青睐。公有云业务在国内乃至全球范围内的普及程度和渗透率也有了大幅度提升。然而与此同时，我们也不难发现，很多大企业及政府机构在面临云计算的建设使用模式的选择时，不可避免地将安全性问题放在了一个非常重要的位置上，甚至是作为首要考量的因素。目前，仅有在自建数据中心及自己维护管理组织的掌控范围内私有云模式才能保障企业敏感涉密的关键信息资产。这一事实决定了私有云仍将是很多大企业建设云计算首要选择的模式，私有云仍将与公有云在未来相当长一段时间内并存发展。只有拉通公有云

和私有云的混合云能够将线上的公有云弹性敏捷优势，与私有云的安全私密保障优势相结合，实现优势互补，才能成为企业的最佳选择(见图2-8)。

然而，考虑到大企业、政府机构等的业务负载的多样性，需要向云端迁移的应用并不仅仅包含核心涉密的信息资产，也包括业务突发性强，资源消耗量大，并且具备资源使用完毕之后可以立即释放的特征，比如开发测试应用、大数据分析计算应用、电商渠道的分布式Web前端应用等，均属于此类应用负载。这些应用当然也更适合采用公有云的方式来承载。但对于同一企业租户来说，如果一部分应用负载部署在公有云端，另一部分应用负载部署在私有云端，则仅仅跨云的身份认证、鉴权、拉通的统一发放及API适配是不够的，更重要的是必须实现拉通公有云和私有云的安全可信网络，实现自动化建立网络连接。

除基于企业应用负载的安全隐私级别分别跨公有云和私有云进行静态部署之外，对于已部署在私有云之上的应用而言，电商网站的三层Web架构、负载均衡、Web前端和数据库后端初始已部署在私有云内。当业务负载高峰到来后，企业用户希望可以在不对Web网站应用做任何修改与配置调整的情况下，实现Web前端到公有云的一键式敏捷弹性伸缩，并借用公有云端的弹性IP及其带宽资源，从而应对峰值业务负载对资源使用量及IP带宽资源的冲击。

为满足上述诉求，需要跨不同的公有云和私有云，构建一层统一的混合云编排调度及API开放层，实现跨不同异构云的统一信息模型，并通过适配层将不同异构私有云、公有云的云服务及API能力集，对齐到混合云的统一信息模型，并通过SDN与各公有云、私有云的网络控制功能相配合，最终完成跨异构云网络互联的自动化。

当然这个统一编排调度引擎，以及API开放层的实现架构，存在不同的可选路径。

1. 路径1

引入一个全新的编排调度层，逐一识别出跨不同异构云的公共服务能力，并以此公共能力及其信息建模为基础参照，进行到各公有云、私有云的计算，存储原生API能力的逐一适配。

该路径下的跨云网络互联方案，需要混合云SDN与各公有云、私有云的VPN网络服务进行紧密协同配合，由于不同异构云之间的网络服务语义及兼容性相比计算和存储服务差别更大，因此也必然给跨云的VPN网络连接适配处理带来了更大的复杂度与挑战。

2. 路径2

依托于业界开源事实标准的云服务与调度层(如OpenStack)，作为拉通各异构公有云、私有云的信息模型及API能力的基准，通过社区力量推动

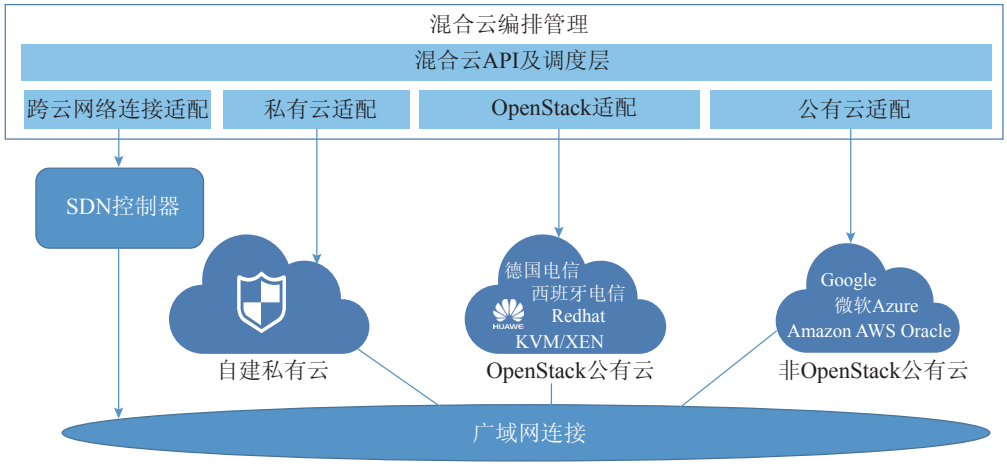


图2-8　混合云架构

各异构云主动提供与该事实标准兼容的适配驱动。

该路径下的跨云网络互联，采用叠加在所有异构云虚拟化之上的Overlay虚拟网络机制，无须进行跨异构云的网络模型适配转换，即可面向租户实现按需的跨云网络互联，从而大大降低了跨云网络互联处理的难度，为混合云的广泛普及奠定了坚实的基础。

2.2 云计算架构关键技术

相对于云计算初期阶段以探索和试用为特征的非互联网领域及行业的基础设施云资源池建设，新阶段云计算基础设施云化已步入大规建设，新阶段云计算基础设施化已步入大规模集中化建设的阶段，需要云操作系统(Cloud OS)必须具备对多地多数据中心内异构多厂家的计算、存储以及网络资源的全面整合能力，因此有如下一些关键技术和算法。

2.2.1 超大规模资源调度算法

我们说希望能像用水用电一样的方式去使用IT资源，那么IT资源的供给就需要许多类似于大大小小的水厂/电厂的IT资源工厂，这就是我们所说的IT数据中心。

以水厂为例，其实我们有各种大小的水库，有时候为了供给一个大城市，我们会通过复杂的管道，将某些江河或某些水库引入城市边上的大水库。就是说，这个供水系统是一个复杂的网络系统，需要有良好的预先设计。

可以看到，尽管我们家家户户使用的自来水没有什么差别，但实际上他们是来自于不同的水厂。每个水厂都可能遇到自己的枯水期，使用这些水厂水资源的客户可能面临缺水的问题，就是说，水的供应并不是无限制的。对应地，其实我们需要的IT资源也并不是无限制供给的，它是由后面大大小小的IT数据中心的能力决定的。当然，为了应对一些大企业的IT资源要求，我们需要将异地的IT数据中心进行联网设计，组成一个大的IT资源池来给大客户使用；此时，这个大资源池的组成技术、调度技术都是关键技术，包括以下三个方面。

一、计算资源调度算法

超大规模资源调度算法实现十万物理机、百万虚拟机的多级、分层调度。

在一个分布式的数据中心情况下，计算虚拟化部分负责L2～L5调度，以虚拟机(含OS及应用软件/中间件)为基本调度单元，完成指定虚拟机实例或者虚拟机集群到整个云数据中心计算资源池内最合适的物理机或者物理机集群的映射(见图2-9)。

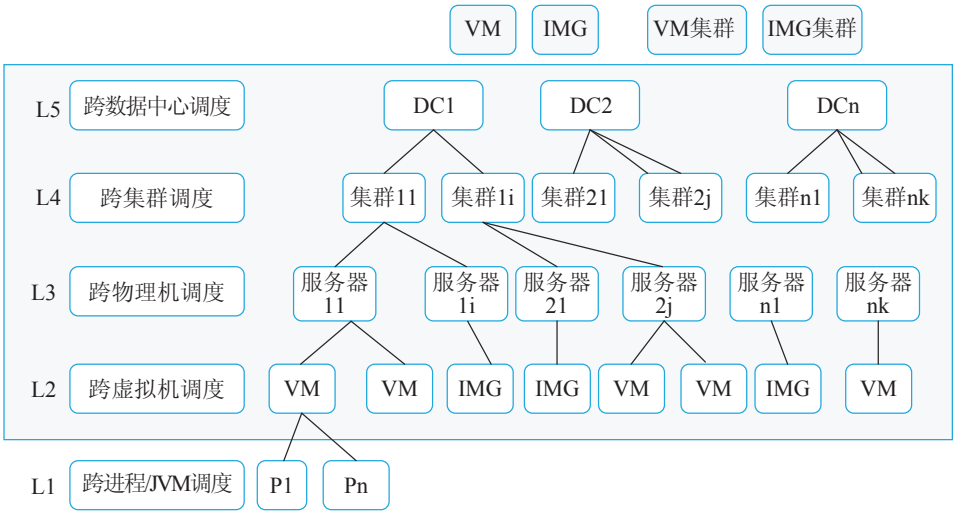


图2-9 超大规模数据中心调度

初始分配调度及动态规划

典型的调度算法：首次匹配、负载均衡、轮转指针等，可统一规划为运筹学的线性规划NP求最优解/次优解的问题，约束条件及目标函数均可按需进行策略配置。

资源弹性分配的问题域

资源弹性分配的限定条件有下面6个表达式，如图2-10所示。

$$\forall i,h \quad e_{ih}\in\{0,1\}, y_{ih}\in Q \qquad (1)$$
$$\forall i \quad \sum h e_{ih}=1 \qquad (2)$$
$$\forall i,h \quad 0\leqslant y_{ih}\leqslant e_{ih} \qquad (3)$$
$$\forall i \quad \sum h y_{ih}\geqslant \hat{y}_i \qquad (4)$$
$$\forall h,j \quad \sum i r_{ij}(y_{ih}(1-\delta_{ij})+e_{ih}\delta_{ij})\leqslant 1 \qquad (5)$$
$$\forall i \quad \sum h y_{ih}\geqslant \hat{y}_i+Y(1-\hat{y}_i) \qquad (6)$$

图2-10　资源弹性分配

其中各变量的含义如下。

- i=1，…，N，表示服务请求数量。
- h=1，…，H，表示每个集群中同质物理服务器的数量。
- j=1，…，d，表示每个服务器提供的资源类型数量(例如CPU、RAM、带宽等)。
- R_{ij}表示第i个服务请求对资源类型j的资源需求量，这个值在0和1之间，表示资源的满足程度。
- δ_{ij}表示R_{ij}是否为固定资源请求类型，取值0或者1。如果R_{ij}是固定资源请求(比如每个用户邮箱服务固定需要内存10G)，则δ_{ij}=1；如果R_{ij}是弹性资源请求(比如每个用户邮箱服务需要的内存可以在0～10G之间)，则δ_{ij}=0。
- $\hat{y}_i$，表示服务i的最小产出要求，取值在0和1之间。例如某个大型企业需要从云中获得邮箱服务1000个用户，并且客户要求无论如何，最差的情况下也需要保证服务200个用户，则此时取值0.2。
- e_{ih}表示资源请求i是否分配在物理服务器h上，取值0或者1。如果是，则取值1，否则取值0。
- y_{ih}表示服务i在服务器h上是否进行Scale方式的输出。如果服务i不在这个服务器上，则取值必须是0。

各个限定表达式的含义具体如下。

- 表达式(1)：表示服务请求i分配在物理服务器h上的状态，或者在，或者不在。
- 表达式(2)：表示无论某个服务器是否承载了服务请求i，所有服务器上满足服务请求i的总和肯定等于1。
- 表达式(3)：表示一个服务i可以在某个服务器h上得到部分运行资源的满足，这个满足程度肯定大于等于0，如果大于0而小于1，表示服务器i需要的资源是分配在多个服务器上的，此服务器只能满足部分资源需求；如果等于1，表示此服务此时无须进行Scale，它完全能在h服务器上得到全部的资源满足。
- 表达式(4)：表示一个服务在相关的服务器上能取得的产出必须大于它的最小产出需求。例如客户要求云系统满足最低200个邮箱用户的需求，而系统中有10万台服务器，不管此时有多少台服务器给这个企业客户提供邮箱服务，都必须保证200个邮箱用户的使用。
- 表达式(5)：表示资源类型j在服务器h上能分配各种服务使用的最大值是1。
- 表达式(6)：表示最小的服务产出Y不会大于任何服务的产出。

资源弹性分配的近似最优解有如下公式：

$$Y=\min\left(1,\min_{j\in NZ}\frac{H-\sum_i r_{ij}(\hat{y}_i(1-\delta_{ij})+\delta_{ij})}{\sum_i(1-\hat{y}_i)r_{ij}(1-\delta_{ij})}\right)$$

其中NZ表示不等于0的物理资源集合。

整个系统的求解就是获得Y值的最大值。一般来说，我们可以采用如下的条件来获得最优解：

$$e_{ih}=1/H \text{ and } y_{ih}=\frac{1}{H}(\hat{y}_i+Y(1-\hat{y}_i))$$

二、存储资源调度算法

存储资源的调度算法主要实现以下几点(见图2-11)。

- 将数据中心服务器(机架式服务器)直连存储(HDD/SSD)转换为高性能、低时延的共享存储资源，大幅提升可用存储空间，实现无SAN化的计算集群的虚拟化整合。

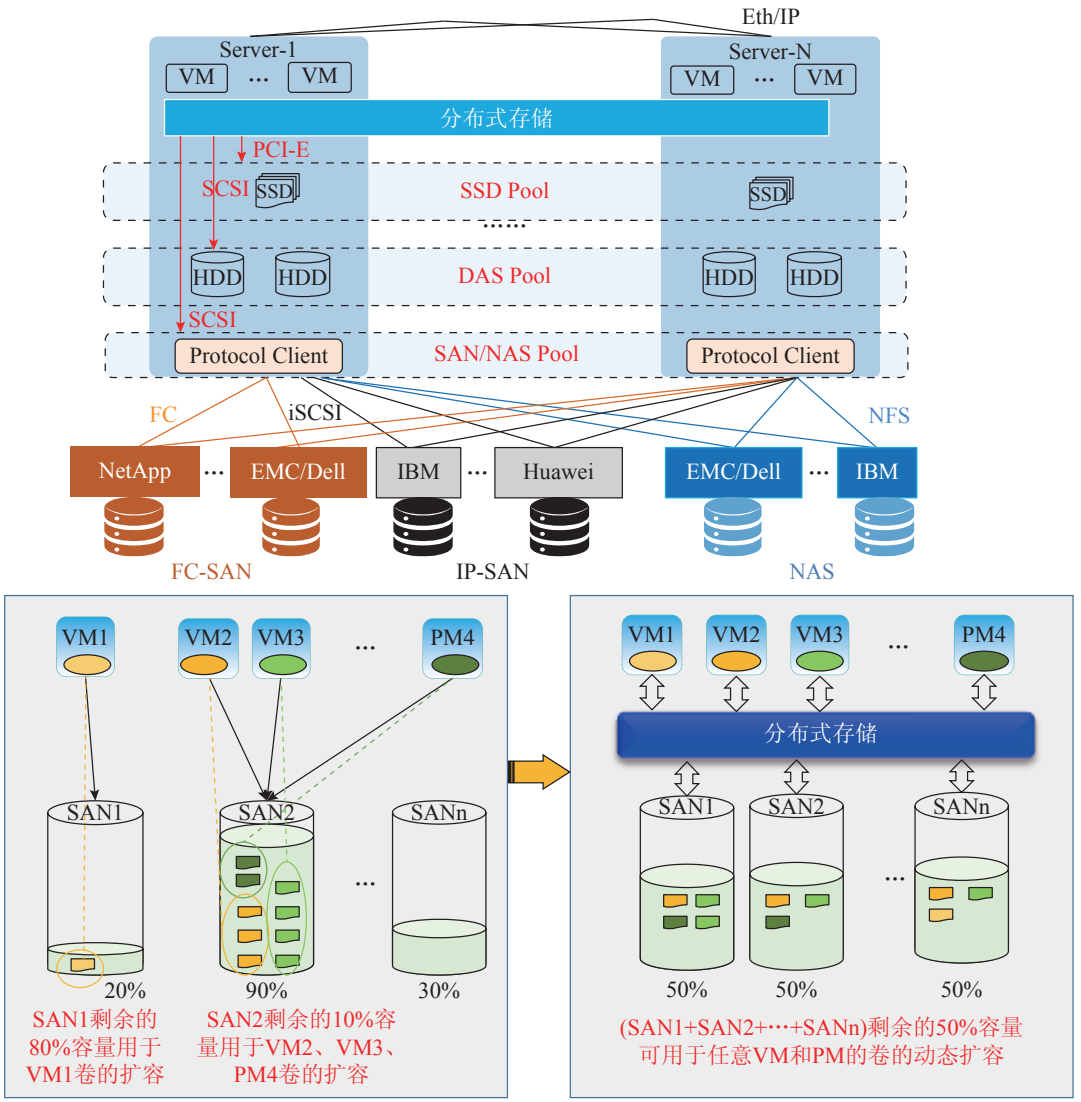


图2-11 异构大存储池

✍ 性能无损的瘦分配、为每VM/PM提供更大的“瘦分配弹性”：为不具备“瘦分配”能力的服务器内置DAS/SSD，以及外置SAN带来天然瘦分配能力，并解决多数外置SAN存储瘦分配带来的性能下降开销问题。

✍ 更大规模的跨SAN资源池，基于在线分布式去重实现更大范围的重复数据识别与删除(文件级/对象级/块级)，将资源利用率进一步提升40%。

✍ 更大规模的资源池，意味着可有更多共享空闲资源满足计算侧需求，避免独立SAN/NAS各自数据不均衡带来的资源浪费(30%)。

✍ 超大资源池下，将“跨SAN”数据热迁移的概率几乎降低为零。

✍ 物理机无Hypervisor，也需要引入“存储融合”层来解决数据的跨SAN热迁移能力(存储大资源池内的)。

三、能耗管理最优化算法

要降低PUE值，实现云计算绿色节能的理念，需要有一个好的能耗管理算法。能耗管理算法在云计算中是一个关键技术(见图2-12)。

- 计算部分是数据中心L1+L2功耗的主要矛盾和关键路径(60%～70%)；
- 数据中心内，基于“轻载合并”原则进行VM热迁移调度，使得更多的空闲服务器可以下电或处于节能运行态；
- 计算虚拟化部分与数据中心L1管理软件联动，尽量减少局部热点，从而允许L1管理软件控制空调提升平均工作温度，达到提升PUE效率的目的；
- 持续动态采集当前负载情况下服务器、UPS和空调、制冷设备的功耗及温度数据，得出PUE指标，并在管理界面上实时呈现。

处理过程具体如下。

1. 输入信息

(1) 物理机信息列表

- 静态规格：CPU主频和数量、内存大小、网卡速率。
- 负载信息：CPU利用率、内存利用率、网络I/O。
- 状态：上电、下电、异常状态。
- 功率信息(可选)：额定功耗、当前功率。
- 温度信息(可选)：当前CPU温度、物理机温度。
- 其他(可选)：物理机能耗效率评级、离冷风送风口距离或评级。

(2) 虚拟机信息列表

- 静态规格：虚拟机CPU(vCPU数量和主频)、内存大小、网卡速率。
- 负载信息：CPU占用、内存占用、网络I/O。
- 约束信息：互斥性约束、亲和性约束。
- 物理机和虚拟机的关联关系。
- 物理机对应的VM ID列表。

(3) 两个场景

- 轻载时，合并VM，物理机下电节能。
- 重载时，启动物理机，均衡VM，保证QoS。

(4) 三个子算法

- 轻载/重载检测算法。
- 上下电PM选择子算法。
- 负载均衡子算法。

(5) 算法设计时要考虑的问题

- 多维资源问题。
- 迁移成本-收益分析。
- 迁移震荡问题。
- What-if测试。
- 配电问题。

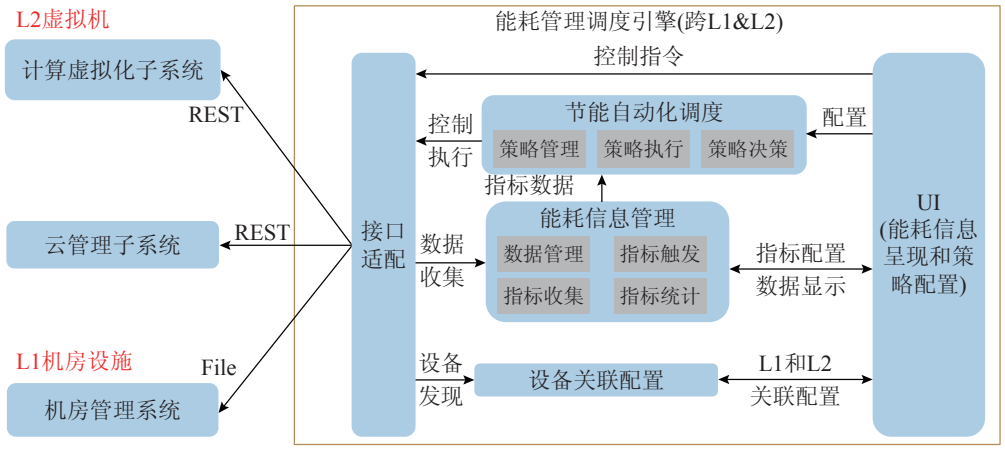


图2-12　能耗管理功能模块图

- 温度问题。
- 调度约束。

2. 输出信息

其主要有两种动作，即物理机上下电动作、VM迁移动作。

2.2.2 异构硬件集成管理能力

一、异构硬件管理集成技术

1. 异构的内容

异构的内容包括以下几点。

- 业务运行平面上，虚拟化引擎层(Hypervisor)天然实现了硬件异构：例如通过XEN及KVM虚拟化引擎的硬件指令仿真，并引入必要的半虚拟化驱动，则可对上层客户机操作系统完全屏蔽多数厂家x86服务器的差异。
- 管理维护平面上，云OAM维护管理软件通过采用灵活的插件机制对各类异构硬件通过有代理以及无代理模式的模式，从各类服务器硬件管理总线、以及操作系统内的Agent，甚至异构硬件自带的管理系统中收集，并适配到统一建模的CIM信息模型中来。
- 虚拟机、物理机统一建模：x86服务器虚拟机、物理机，以及ARM物理机的异构集群管理。

2. 异构实现原理

异构实现原理如图2-13所示。

二、异构Hypervisor简化管理集成技术

针对数据中心场景，企业IT系统中的Hypervisor选择往往不是唯一的，可能有VMware的ESX主机及vSphere集群，可能有Microsoft的Hyper-V及SystemCenter集群，也可能有从开源KVM/XEN衍生的Hypervisor(如华为UVP等)多种选择并存。此时云操作系统是否有能力对这些异构Hypervisor加以统一调度管理呢？答案是肯定的。可以依托OpenStack开源框架，通过Plug-in及Driver等扩展机制，将业界所有主流的Hypervisor主机或者主机集群管理接口统一适配到OpenStack的信息模型中来，并提供V2V/P2V虚拟机镜像的转换工具，在异构Hypervisor之间按需进行虚拟机镜像转换。这样即使不同Hypervisor也可共存于同一集群，共享相同存储及网络服务，甚至HA服务。

资源以统一集群方式管理(OpenStack目标)，屏蔽Hypervisor差异，简化云计算资源管理(见图2-14)。

三、异构存储管理集成的统一简化技术

异构存储管理继承的统一简化技术主要包括如下几点(见图2-15)。

- 10PB级存储大资源池，跨多厂家异构外置存储，以及服务器自带SSD/HDD的资源池化，将存储服务抽象为同时适用于虚拟机和物理机的“统一EBS”服务；
- 容量、IOPS、MBPS等SLA/QoS是EBS存

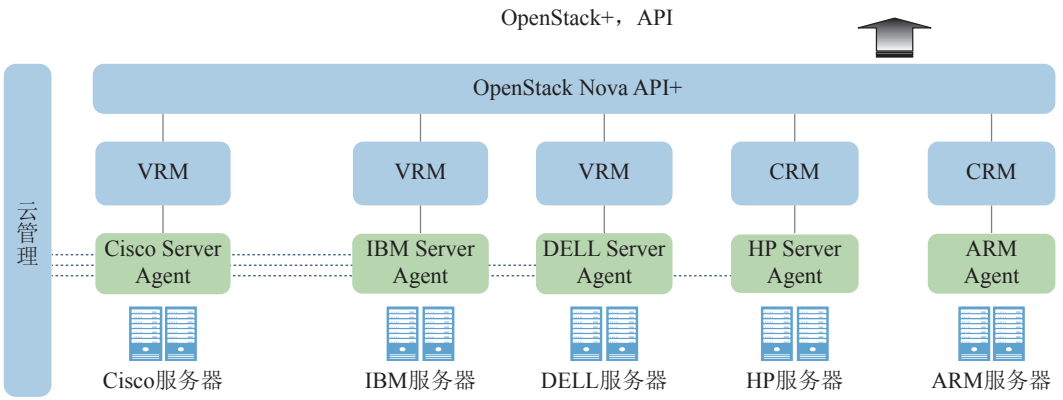


图2-13 硬件异构兼容原理

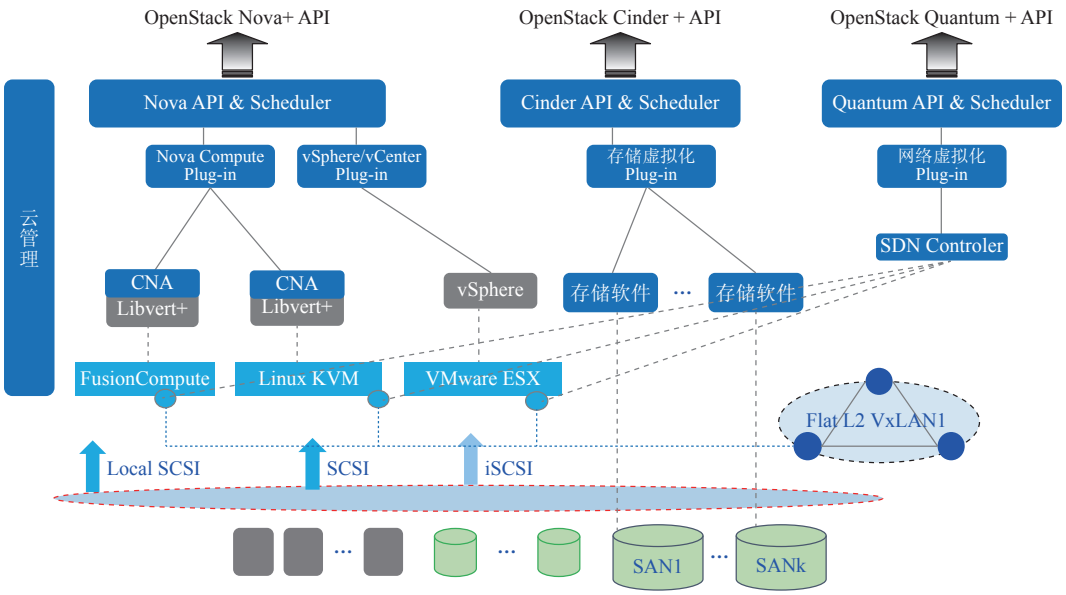


图2-14　Hypervisor异构统一管理原理

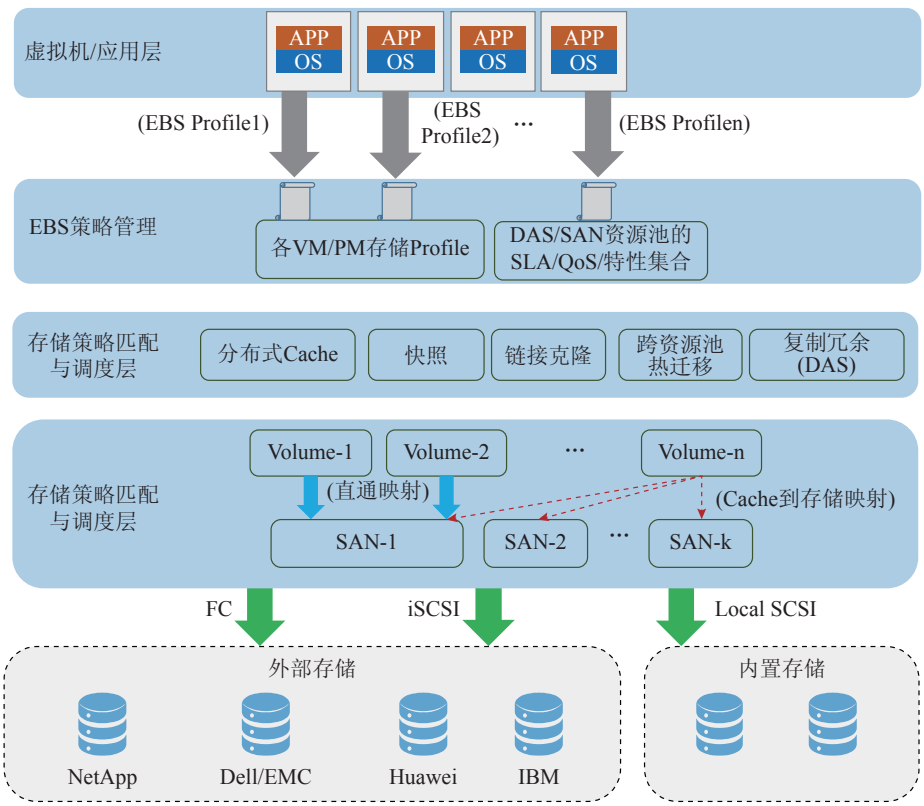


图2-15　存储异构统一管理原理

储服务界面的“统一语言”，与具体支撑该服务的存储形态及厂家无关；

- 可按需将部分存储高级功能(数据冗余保护、置0操作、内部LUN拷贝、链接克隆等)卸载到外置存储(类VVOL)；
- 针对DAS存储融合，应用层逻辑卷与存储LUN之间采用DHT分布式打散映射，以及一致的RAID保护；
- 针对SAN存储融合，应用层逻辑卷与存储LUN之间采用DHT分布式打散映射(新建卷)，或者直接映射(利旧并平滑迁移已有卷)，数据可靠性一般由SAN存储自身负责；
- 同一应用Volume的直接映射卷可“逐步”平滑迁移到DHT映射卷，实现业务中断。

2.2.3 应用无关的可靠性保障技术

一、数据中心内的可靠性保障技术

数据中心内的可靠性保障技术主要包括HA(High Availability)冷备份、FT(Fault Tolerance)热备份、轻量级FT。

1. HA(High Availability)冷备份

数据中心内基于共享存储的冷迁移，在由于软件或硬件原因引发主用VM/PM故障的情况下，触发应用在备用服务器上启动。其适用于不要求业务零中断或无状态应用的可靠性保障(见图2-16)。

2. FT(Fault Tolerance)热备份

其指指令、内存、所有状态数据同步。该方式的优势是状态完全同步，完全保证一致性，且支持SMP。劣势是性能开销大，会带来40%左右的性能降低(见图2-17)。

3. 轻量级FT

其是基于I/O同步的FT热备机制。优势是CPU/网络性能损耗10%以内，支持单核和多核。劣势是适合于网络I/O为主服务的场景(见图2-18)。

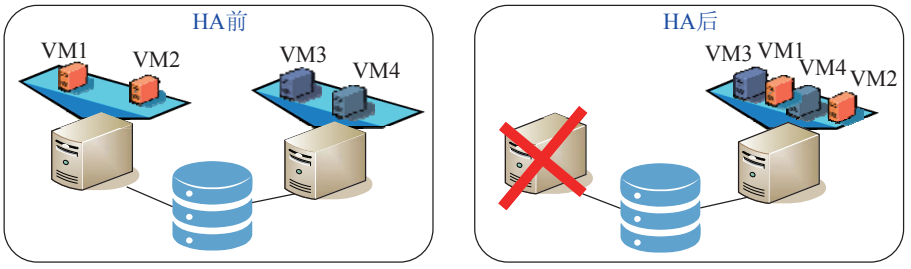


图2-16 冷备份原理

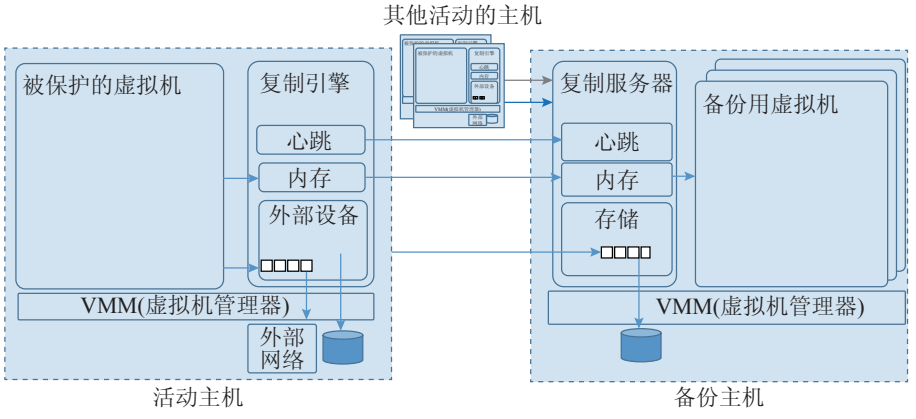


图2-17 热备份原理

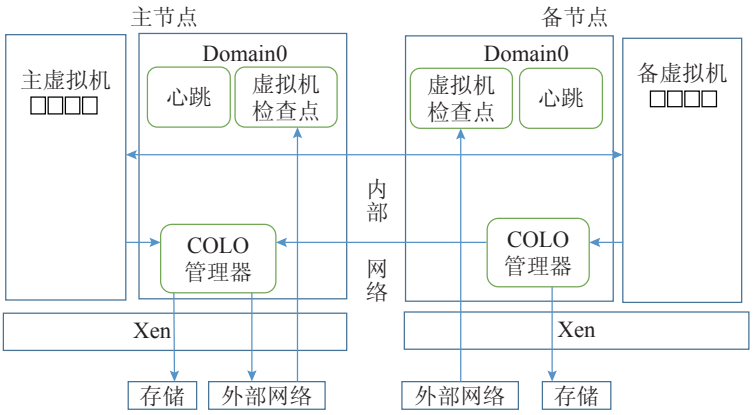


图2-18　轻量级FT原理

二、跨数据中心的可靠性保障技术

跨数据中心的可靠性保障技术，主要是基于存储虚拟化层I/O复制的同步和异步容灾两种。

基于存储虚拟化层I/O复制的同步容灾，采用生产和容灾中心同城(<100KM)部署，时延小于5ms，DC间带宽充裕，并且对RPO(恢复点目标)要求较高，一般RPO接近或者等于0秒。分布式块存储提供更高效的I/O同步复制效率(见图2-19)。

基于存储虚拟化层I/O复制的异步容灾采用生产和容灾中心异地(大于100KM)部署，带宽受限，时延大于5ms，同时对RPO有一定的容忍度，如RPO大于5分钟。I/O复制及快照对性能的影响趋近于零(见图2-20)。

2.2.4　单VM及多VM的弹性伸缩技术

单VM及多VM的弹性伸缩技术包括基本资源部件级别、虚拟机级别、云系统级别三个层次的伸缩技术。

基本资源部件级别：精细化的Hypervisor资源调度，对指定虚拟机实例的CPU、内存及存储

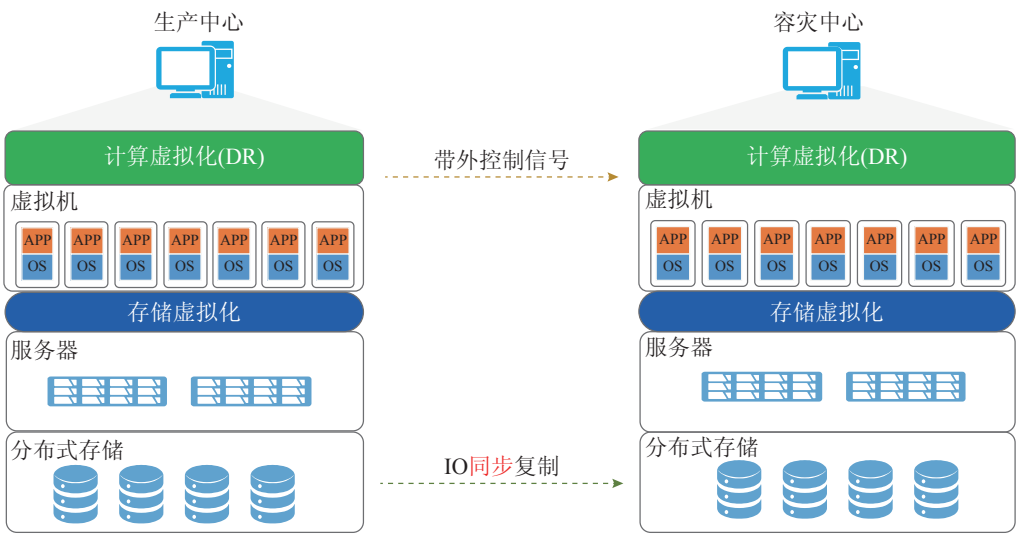


图2-19　基于应用层的容灾复制原理

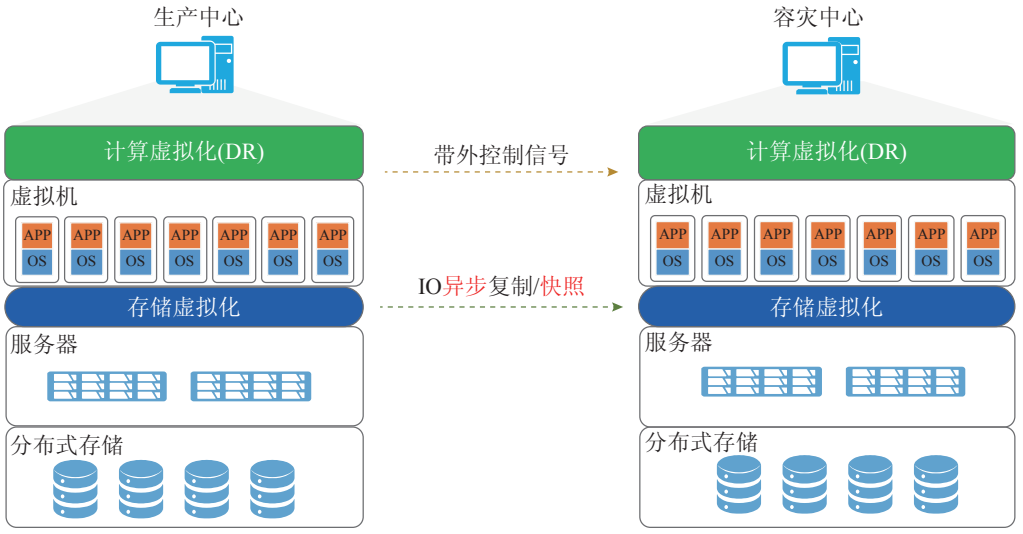


图2-20　基于存储层的容灾复制原理

规格进行弹性伸缩，并可对伸缩上下限进行配额限制。

虚拟机级别：指虚拟机集群的自动扩展与收缩，基于CloudWatch机制对集群资源忙闲程度的监控，对业务集群进行集群伸缩与扩展的Auto-Scaling控制。

云系统级别：在内部私有云资源不足的情况下，自动向外部公有云或其他私有云(计算及存储资源池)“租借”及“释放”资源。

上述弹性伸缩机制，使得在大规模共享资源池前提下，流控及因流控引发的业务损失被完全规避(见图2-21)。

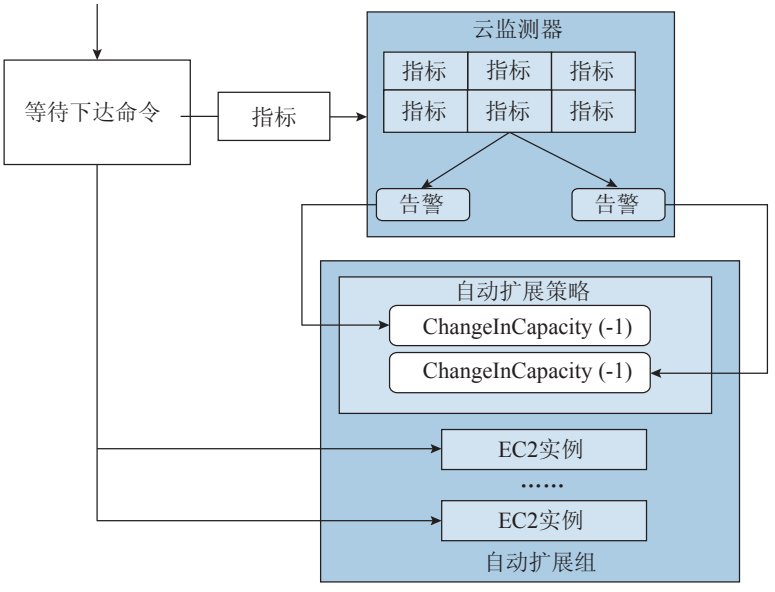


图2-21　弹性伸缩

2.2.5 计算近端I/O性能加速技术

原则上，针对在线处理应用，I/O加速应发生在最靠近计算的位置上，因此作为提高I/O性能的分布式Cache应该运行在计算侧(见图2-22)：

- 远端Cache的I/O效率，高出本地IOPS/MBPS效率1个数量级；
- 通过分布式内存、SSD Cache，实现对内部和外部HDD硬盘介质资源的I/O性能提升2～3倍；
- NVDIMM/NVRAM和SSD Cache保证在全局掉电(或多于2个节点故障情况下)情况下计算近端的写Cache数据无丢失；
- 分布式Cache可提供更大的单VM(单应用)的磁盘并发MBPS，效率可提升3～5倍。

2.2.6 网络虚拟化技术

一、业务应用驱动的边缘虚拟网络自动化

分散在交换机、防火墙、路由器的L2转发表及L3路由表集中到SDN控制器，使得跨多节点的集中拓扑管控及快速重定义成为可能。基于x86的软件交换机和VxLAN隧道封装的Overlay叠加网可以实现业务驱动且与物理网络彻底解耦的逻辑网络自动化，支持跨数据中心的大二层组网。基于业务模板驱动网络自动化配置，如图2-23所示。

二、更强大、更灵活的网络安全智能策略

在云计算早期阶段，一般采用下面的方法进行网络安全的部署，但存在一些不足。

- 公有云、多租户共享子网场景下，静态配置安全组规则仅在目的端进行过滤，无法规避DOS攻击(见图2-24)。
- 采用外置防火墙方法控制子网间安全，但存在流量迂回(见图2-25)。

为解决云计算早期阶段技术安全隐患，新阶段云计算架构通过软件定义网络的实施，解决这些问题(见图2-26)。

- 按业务需求统一定义任意目标——源组合的安全策略定义下发到Controller；
- 子网内互访，首包上送Controller，动态下发安全过滤规则，源头扼杀攻击；
- 子网间互访，动态下发快转流表，避免迂回。

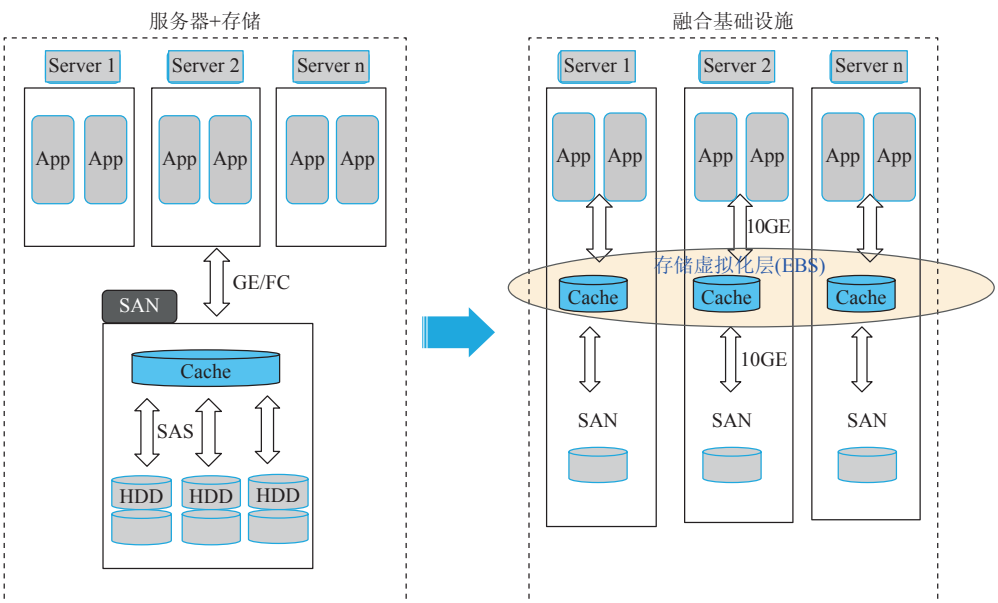


图2-22 存储缓存加速功能

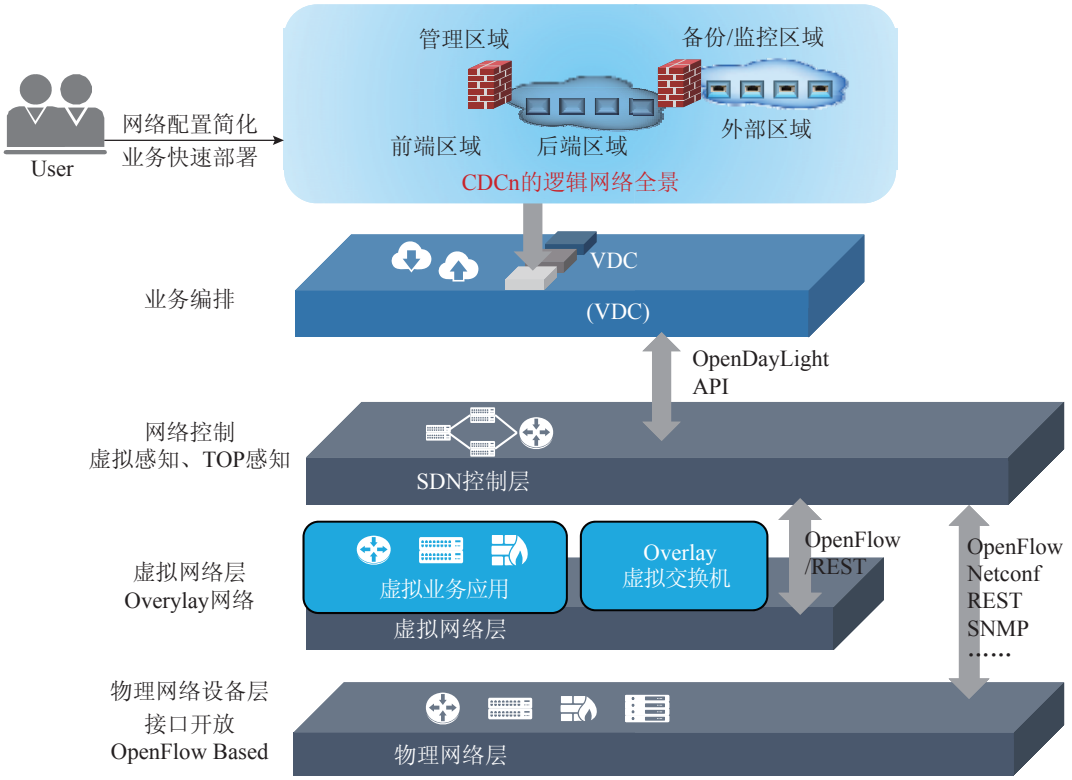


图2-23　网络功能虚拟化层次架构

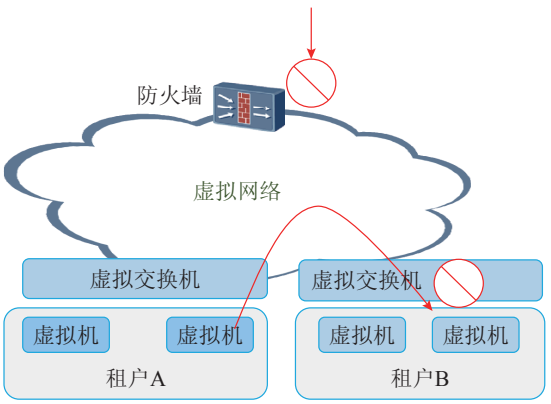


图2-24　虚拟机网络安全原理

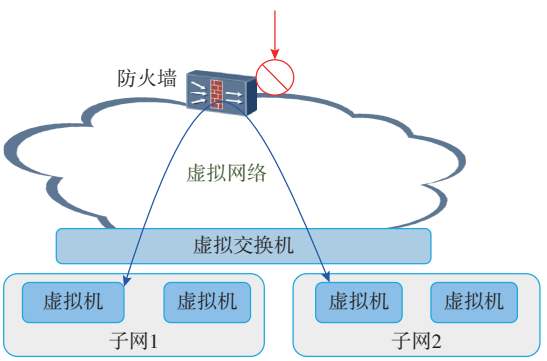


图2-25　采用外置防火墙方法控制子网间安全

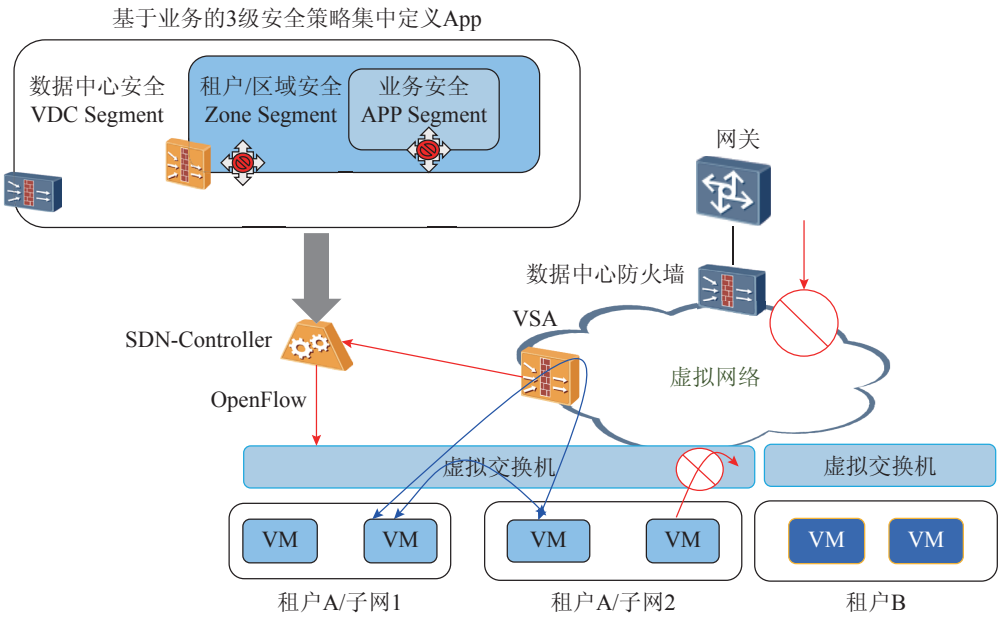


图2-26　通过软件定义网络的实施解决问题

2.2.7　应用模块以及工作流技术

对于目标架构的基础设施层的管理功能定位，仅仅做好物理和虚拟机资源的调度是远远不够的，其应当涵盖独立于具体业务应用逻辑的普遍适用的弹性基础设施之上的应用的全生命周期管理功能，涵盖从应用模板、应用资源部署、配置变更、业务应用上线运行之后基于应用资源占用监控的动态弹性伸缩、故障自愈以及应用销毁的功能。整个应用的生命周期管理应遵循如图2-27所示的流程。

图2-27　全生命周期管理流程

各部分包括以下内容。

(1) 图形化的应用模板设计方式：采用基于图形的可嵌套式重用模板；采用拖拽和粘贴拷贝式的方式来定义分布式应用模板；使得模板设计简单高效(见图2-28、图2-29)。

(2) 提前准备的丰富模板库和自动部署：为物理机、容灾、SDN、LB、防火墙等准备好应用模板；当有应用需求时，系统直接从模板库中选取相应的模板进行自动部署(见图2-30)。

图2-28　图形化的应用模板

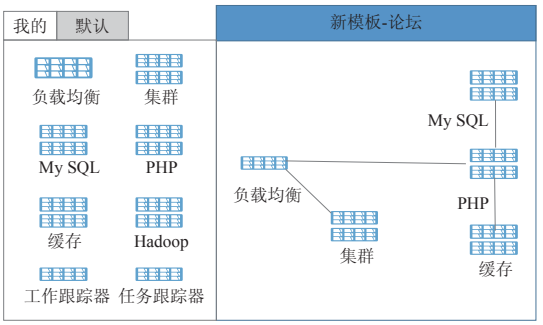


图2-29 图形化的应用模板设计

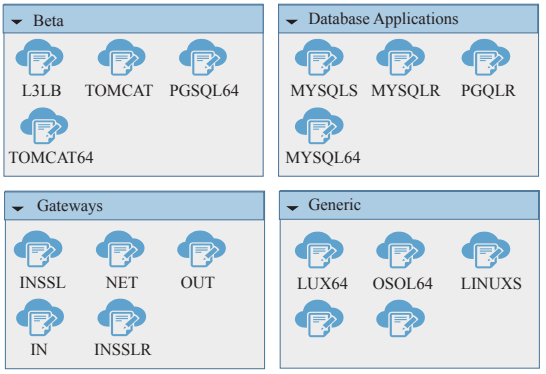


图2-30 从模板库中选取相应的模板

(3) 基于SLA的应用监控：面向不同的应用(数据库、HPC、基于LAMP的Web网站等)，定义不同的SLA指标集，对这些指标进行监控，采用静态阈值和动态基线相结合的方法进行故障告警和性能预警，使应用监控更自动化和精细化，满足客户业务运行的要求。

(4) 基于工作流的应用故障自愈：采用基于工作流的管理方式，通过对应的设计工具来设计用户自定义事件，当监控到应用故障、事件触发和工作流引擎的运转时，系统支持应用的自动修复，达到故障自愈的目的。

2.2.8 容器调度与编排机制

Docker 2013年发布正式开源版本后，容器技术成为云计算领域一个新的热点。进入容器的世界后，你会发现Docker生态系统十分庞大，Docker公司发布的容器引擎只是单节点上管理容器的守护进程，而企业数据中心或者公有云管理的节点规模十分庞大，因而一个成熟的容器管理平台还需要至少下述两个层面的能力(见图2-31)。

1. 容器集群资源管理和调度

收集被管理节点的资源状态，完成数以万计规模的节点的资源管理；同时根据特定的调度策略和算法，处理用户的容器资源申请请求。

2. 应用编排和管理

针对数据中心不同类型的应用，比如Web服务，批量任务处理等，抽象不同类型应用常用的应用管理的基础能力，并通过API暴露给用户使用，从而用户开发和部署应用时可以利用容器管理平台的上述API能力实现对应用的自动化管理。用户通过应用编排和管理还可以实现应用模版定义，以及一键式自动化部署。应用模版包括组成应用的各个组件的定义，以及组件间逻辑关系定义，用户可以从应用模版一键部署应用，实现应用灰度升级等，大大简化了应用的管理和部署。

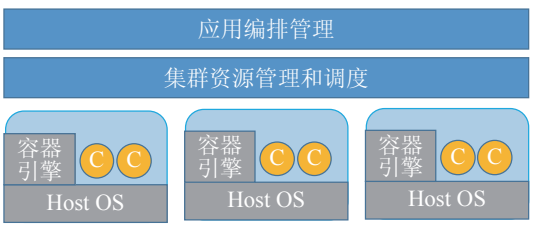


图2-31 容器调度与编排机制

本书第五章对容器调度和编排机制进行了详细的阐述。

2.2.9 混合云适配连接机制

越来越多的企业使用混合云，将公有云资源作为企业侧的私有云的延伸进行统一管理。混合云适配连接机制需要提供的功能包括：

- 统一对象模型；
- 统一北向API；
- 云间网络互联互通。

典型的混合云架构，如图2-32所示。

典型的混合云架构，分为如下三层。

1. Cloud Gateway

Cloud Gateway部署在最底层，对接异构云的

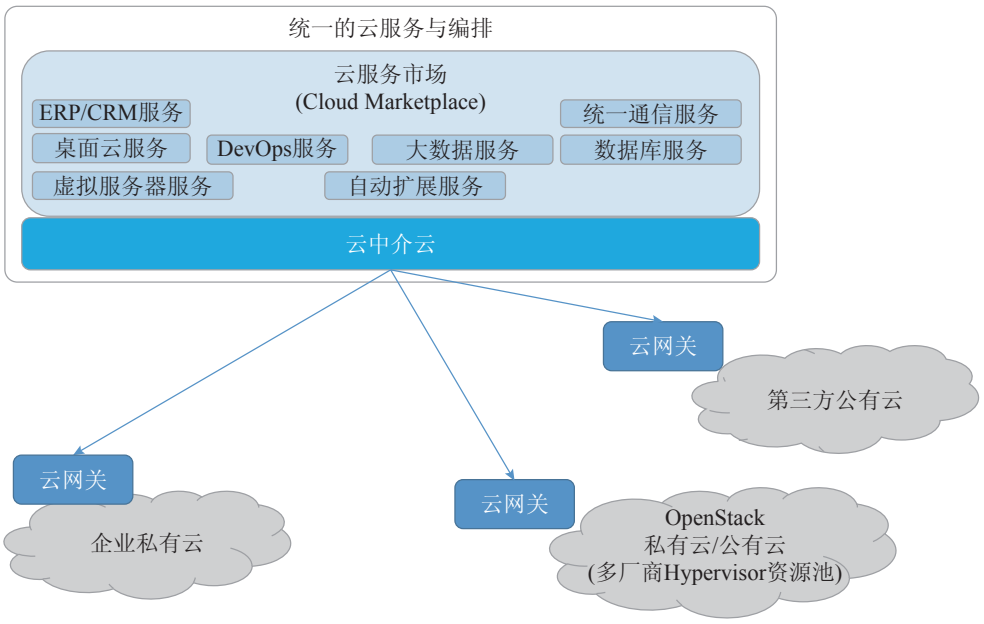


图2-32　混合云架构

资源管理服务。以AWS为例，Cloud Gateway对接适配AWS的弹性计算服务(EC2)、弹性块存储服务(EBS)和虚拟网络服务(VPC)，用于异构云对象模型的转换和API适配，屏蔽云间功能差异性。

2. Cloud Broker

Cloud Broker提供加云管理、加云配置(用于接入供应者云进行发放资源的账号、密码、Flavor映射关联等云配置)、统一租户管理、云间资源管理和云间互联互通功能。Cloud Broker通过Cloud Gateway将供应者云作为计算、存储资源池使用，在此之上提供统一的租户管理，实现云间资源的统一管理和资源跨云统一编排，对外提供统一的API接口；Cloud Broker通过Cloud Gateway使用VPN隧道或者云间专线互联的方式实现跨云网络的互联互通。

3. Cloud Market

Cloud Market在Cloud Broker提供的云间资源统一管理和云间网络互联互通的功能基础之上，通过调用Cloud Broker北向统一的云管理API实现服务的跨云统一部署和编排。

2.3　云计算核心架构竞争力衡量维度

从将云计算技术引入传统数据中心所带来的独特商业价值角度看，重点可以从开源与节流两个方面来衡量云计算的核心竞争力。

一、节流(Cost Saving)方面

在业务系统搭建过程中，云计算和虚拟化使得企业及运营商的烟囱式软件应用可以突破应用边界的束缚，充分共享企业范围内、行业范围内甚至全球范围内公用的“IT资源池”，无须采购和安装实际物理形态的服务器、交换机以及存储硬件，而是依赖于向集中的“IT资源池”动态申请所需的虚拟IT资源(或资源集合)，就可以完成相关应用的自动化安装部署，从而达到快速搭建支撑自身核心业务的IT系统与基础平台的目的。这种模式可以减少系统搭建的人力和资源投入，降低系统初始构筑成本。

在业务应用执行过程中，依托节能减排及资源利用率最大化原则，实现必要的智能资源动态

调度，以完成既定的业务处理或计算任务，并在特性业务处理或计算任务完成后即时地释放相关IT资源供其他企业、行业进一步动态共享，从而实现IT建设与运维成本的大幅度优化与降低。

另外，针对涉及海量数据处理及科学计算的特殊行业，以往依托于造价昂贵小型机、大型机甚至巨型机、高端存储阵列，或者采用通用处理设备需要数月甚至数年才能完成的复杂计算与分析任务，有可能在云计算数据中心基于通用服务器集群，以更为低廉的成本并花费更短的时间就可以轻松应对。

二、开源(Revenue Generation)方面

针对公有云数据中心运营商的价值：将SaaS等早在云计算概念出现就已普及的资源服务的概念进一步扩展到IaaS与PaaS层，云计算数据中心运营商可以在IaaS/PaaS上建设自营增值业务服务于云用户，也可引入众多第三方应用运行在IaaS/PaaS云平台之上，实现相比传统数据中心托管服务具备更高附加值的虚拟机、虚拟桌面及虚拟数据中心租赁业务，或者在第三方应用开发/提供商、云运营商(IaaS/PaaS云平台提供者)以及云租户/云用户之间分享丰富的SaaS应用以带来增值利润。

针对企业私有云数据中心建设的价值：云计算使得IT基础架构可以对与企业、行业业务紧密绑定的业务软件形成更为高效和敏捷的集成融合，从而大大提升企业IT资源灵活适应并支撑企业核心业务流程与业务模式快速变化的能力，有效地优化企业业务的运作效率。

云计算的海量数据分析与挖掘能力的价值：使得企业、行业有能力依托其海量存储及并行分析与处理框架的能力，从其企业IT系统所产生的海量的历史数据中提炼并萃取出对其有价值的独特信息与价值，从而为其市场及业务战略的及时优化调整提供智能化决策引擎，从而有效提升企业的竞争力。

基于以上云计算数据中心解决方案商业价值，可以从下面六大架构质量属性指标来衡量云计算数据中心解决方案的竞争力(见图2-33)。

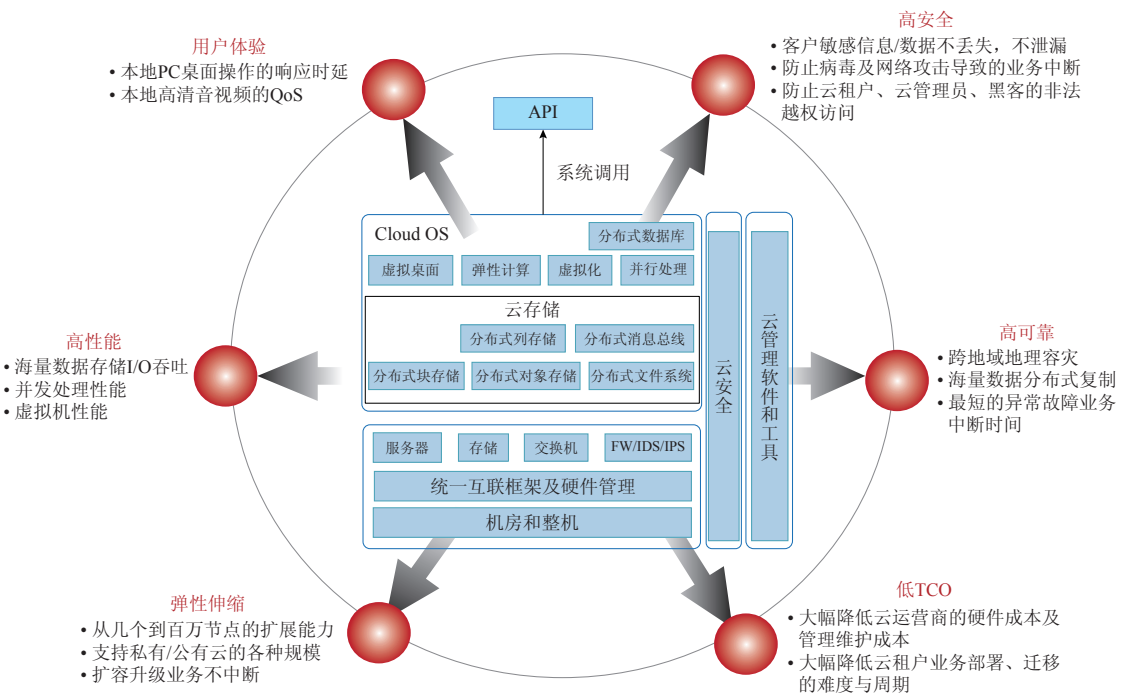


图2-33　云计算架构核心竞争力

2.3.1 低TCO

低TCO能力的构建包括降低和优化云计算数据中心的设备投资成本以及运维成本两个方面。

设备投资成本的优化与节能主要考虑的架构策略涉及如下策略。

一、计算资源的成本优化与节省

站在整体数据中心资源集群成本的视角，针对由以太交换网连接起来的计算密集型服务器构成的计算集群，云计算资源集群初始调度分配以及运行态动态算法的优劣，决定了通过资源占用的削峰错谷可以带来的资源利用率提升，以及相应的成本下降比例。

除调度算法之外，支撑更大规模的集群是实现计算资源CAPEX成本最优化的有效手段，云计算的云资源调度软件通过支持大甚至是超大规模的HA集群(例如：总集群服务器容量达到128服务器/集群)，实现对多种大小不同颗粒度的客户服务器集群的容纳，从而降低资源碎片概率，提升资源利用率。

考虑将成本颗粒度从服务器集群细化层面向下细化到单服务器层面，虚拟机VMM引擎在一个服务器范围内的CPU及内存资源调度能否实现跨虚拟机的充分动态共享，则决定了服务器颗粒度内的多虚拟机资源利用率的高低，以及对应的成本竞争力。虚拟化引擎通过支持实时应用调度优化、小包数据中断调度优化，以及内存气泡、内存交换与共享等优化措施来提高服务器级资源利用率。

二、存储资源的成本优化与节省

在普通云计算数据中心环境下，存储容量一般均在几十TB以上，在满足相同容量及IOPS存储吞吐能力需求的基础之上，需从成本角度出发做出权衡抉择，即采用基于集中RAID控制器带一系列存储磁盘的垂直扩展(Scale-up)模式，还是采用基于全分布式及普通服务器附带硬盘存储的水平扩展(Scale-out)模式。通过引入全分布式存储，有望通过差异化架构规避RAID双控制机头随存储容量与处理能力的上升所带来的成本指数级增长的矛盾，从而实现云存储成本的大幅降低及性价比的提升。

三、网络资源的成本优化与节省

在可能的情况下，考虑取消独立硬件形态的汇聚网络交换机及防火墙网关设备，在通用x86平台上支持Load Balancer、防火墙等设备，从而有效降低网络资源的成本占用。由于部分云计算虚拟网络特性(如ACL、安全组等)可能大量消耗CPU资源，需要考虑将相关功能卸载到智能网卡上。

四、维护成本的优化与节省

为实现数据中心大规模计算、存储集群依据多层网络交换设备的维护成本最优化，要求云管理OSS支持最大限度的智能化管理，实现系统在故障状态下对DC内部服务器、网络及存储资源垂直整合的融合架构，一站式交付将大大降低硬件安装维护复杂度。

五、节能减排等生命周期维护成本的节省

为达到数据中心整个运行服务周期中节能减排效率的不断提升，包括在完成相同工作负荷的前提下更为有效地降低服务器、存储及交换设备自身的耗电量，可采用以下几项关键措施：

- 在云管理层面引入更为优秀的资源调度算法，通过热迁移机制实现将轻载应用尽量合并到数量更少的服务器上，其他服务器则直接下电，从而提升整体资源利用率；
- 在服务器颗粒度内，引入多级节能控制机制，在轻工作负载时自动调整CPU工作于节能状态；
- 在硬件选型方面尽可能选择低功耗CPU以及器件、组件以构筑低成本优势，不断改善服务器单板散热布局；
- 引入分布式电池或者电容，减少由于UPS在空载或轻载情况下的电源效率损失；
- 在数据中心基础设施层引入更为智能的热管理软件及监测手段，并实现充分的冷热风道

隔离，以及热耗散的自动补偿，甚至通过直接拉通来实现整体PUE效率最佳。

2.3.2 弹性伸缩

弹性伸缩要求以相同架构，支撑从最少几个计算与存储节点，到最大10万甚至是100万级的计算与存储节点集群规模，且保证数据中心容量扩展过程中的业务连续性及业务服务不中断，或中断时延最短。

这里的弹性伸缩扩展能力应该体现在：

- 管理节点的弹性伸缩能力；
- 数据中心资源的弹性伸缩能力；
- 所承载云租户业务的计算集群弹性伸缩能力；
- 承载用户数据信息及系统卷镜像的存储集群的弹性伸缩能力；
- 连接计算与存储集群资源的网络弹性伸缩能力。

为了支持该能力，数据中心交换枢纽需要支持大二层虚拟化网络，采用CLOS无阻塞以太网连接模型；存储资源需要支持全分布式存储架构；计算资源需要支持大集群规模；管理节点需要支持基于共享存储、无状态机制实现的可无级扩展的管理能力。这些要求是支撑该强弹性伸缩能力的根本保障。

2.3.3 高性能

整体云计算的性能，重点体现在以下几个维度。

- 虚拟化云平台上运行普通颗粒度托管应用场景：I/O吞吐性能、CPU调度效率等相比同等处理能力物理机平台的下降比例越小，性能竞争力越强。
- 并发云平台上运行超大规模数据分析与应用处理场景：完成既定任务所耗费的时间越少，或计算、存储资源越少，性能竞争力越强。
- 频繁数据库操作或媒体类存储信息的应用场景：云存储的IOPS吞吐率最大，可共享的存储容量越大，性能竞争力越强。
- 云计算数据中心内依赖于网络总线的分布式B/S、C/S应用(如网站)场景：网络时延越短，性能竞争力越强。
- 批量虚拟机发放、虚拟机系统加载等涉及大流量、大尺寸数据流及文件处理的场景：需要通过包括P2P加载、链接克隆等有效架构手段加以优化，或通过Cache机制减少跨节点性能压力。其依托弹性计算及分布式存储与中间件的并行数据分析引擎，支持批处理及流式海量数据分析与挖掘，从而提升性能。

2.3.4 领先的用户体验

以桌面云应用场景为例，通过局域网络或者远程网络连接的桌面业务体验，包括基础桌面应用操作、音视频播放、VoIP、高清视频以及3D加速图形处理密集型应用，达到与本地桌面体验持平，并在时延、抖动、带宽占用等方面有优势，这决定了直接面向企业及个人家庭最终用户的云业务的质量保障的评价。

2.3.5 高安全

安全性无疑是云计算技术在数据中心建设部署与扩容中被采纳的首要障碍性因素，尤其是公有云场景，该方面的问题更为突出，因而其也是架构竞争力衡量的关键维度。对安全质量属性的需求实际上贯穿于云计算架构的自低向上的各个层面，包括：

- 物理层的数据中心安全防护，实现硬件层与软件层关联可信度管理的TPM机制；
- 虚拟化层公共的事件检测、防病毒及安全管理机制；
- 操作系统安全加固，去除无用服务及安全隐患；
- 面向云租户/云用户的云资源管理接入认证与加密管理；
- 面向云资源管理维护者的分权分域及认证鉴权管理，以及面向虚拟私有云的分级资源管理授权能力；
- 面向云租户/云用户的数据传输加密、解

密及网络层安全隔离机制；

✍ 面向云租户/云用户，以及云管理员的数据中心边界安全网关、防火墙或者地址隐藏转换机制(集中硬件或分布式软件)；

✍ 面向云租户/云用户的云存储持久化数据加密、解密及其密钥管理机制；

✍ 作为云管理维护及云用户交互界面的Web Portal的应用层安全防攻击机制；

✍ 面向云管理者及监管机构的，基于数据中心系统管理与业务日志的安全合规性分析；

✍ 托管应用的安全网关(如Email、其他Web应用等，可选)。

2.3.6 高可靠

更高级别的可靠性一直被公认为是集约式的数据中心计算模式，相比传统全分布PC计算模式，其可以提供货架式的、具备量化服务水平保障的增值特性。由于云计算技术的引入，使得数据中心系统的动态计算负载可以进一步以与上层应用无关的形态进行跨越硬件服务器边界的调度，同时数据信息也可通过网络在数据中心内不同持久化存储和计算节点内存之间，甚至是跨地理区域的多个不同数据中心之间进行容灾同步，使得运行于云平台之上的应用相比传统方式，可将更多的精力聚焦于核心业务，而将可靠性保障留给云计算数据中心IaaS服务层来提供。

高可靠性的架构属性保障涵盖如下方面。

✍ 云管理节点自身的可靠性保障机制。

✍ 承载用户计算负载的计算节点的故障恢复机制：计算节点本地重启故障，以及不可本地重启类的异地恢复类故障发生时，如何在无须维护干预以及应用层特殊处理的前提下，保持业务提供的连续性。

✍ 云计算数据中心整体网络的可靠性保障机制。

✍ 云存储数据连续服务与数据防丢失保障机制，如硬盘故障、服务器故障、机柜/机框，乃至整个数据中心意外电源及网络故障的容错与恢复能力。

2.4 云计算解决方案的典型服务与落地架构

2.4.1 弹性计算云服务

基于云计算总体架构下的弹性云计算服务解决方案如图2-34所示。

弹性计算业务与传统的主机出租业务相比，

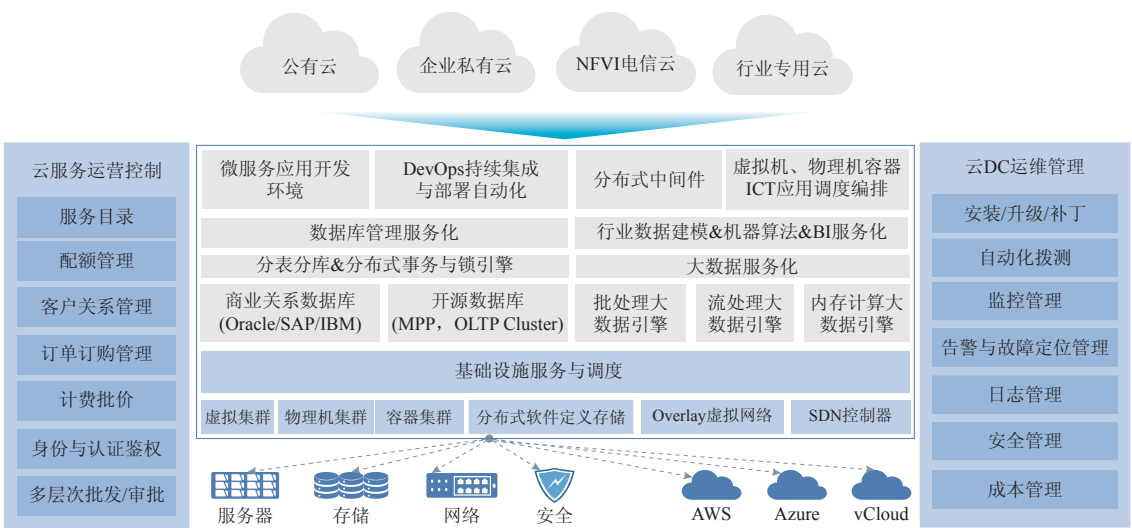


图2-34 弹性计算云服务解决方案架构子系统组合

具有快速部署、按照使用付费的特点，整个业务过程不需要服务提供商人工参与资源的分配。

弹性计算业务提供了一系列不同规格的计算资源，这些规格参数包括CPU性能、内存、操作系统、磁盘、网络。不同规格的计算资源有不同的价格，用户可以根据需要申请不同规格的计算资源。

弹性计算业务实现了对计算、存储、网络资源的打包销售，对计算资源按照使用计算资源的性能和使用时间收费；对于存储资源按照存储容量、存储时间、存储的IOPS和数据量收费；对网络资源按照流量和IP地址的资源的租用时间收费。

云主机类型分为虚拟机和裸金属两种，其发放的主要业务流程包括以下几点。

- 用户操作企业IT系统或运营商BOSS系统进行云主机发放，选择云主机的Flavor (CPU性能、内存)，虚拟磁盘数量、性能规格(IOPS)和大小，虚拟网口的数量，连接的虚拟网络和虚拟网口的带宽要求，以及选择云主机运行镜像文件。
- 企业IT系统或运营商BOSS系统通过弹性计算云服务的SOAP/RESTful API接口下发云主机发放命令，命令中包含IAM系统为用户分配的操作TOKEN。
- 弹性计算云服务的API模块收到命令后首先进行TOKEN鉴权，确认用户是否具有云主机资源发放权限，然后进行用户配额核查，确认用户发放的云主机、虚拟磁盘、虚拟网口资源数量没有超过配额。
- 弹性计算云服务的API模块将云主机信息写入数据库，生成一个系统唯一的云主机UUID返回给用户。
- API模块根据用户配置的Flavor信息选择满足SLA规格的计算资源池，向调度器模块发出物理主机调度的请求消息。
- 调度器在计算资源池中筛选满足CPU、内存资源要求的可用物理主机，向运行在该物理主机侧的虚拟机发放代理模块或者物理机发放代理发送创建云主机的请求消息。
- 代理模块根据运行镜像信息调用云存储服务SOAP/RESTful API，创建以该镜像为内容的Bootable卷，根据卷物理链接信息将卷挂接到物理主机上。
- 代理模块根据用户配置的虚拟网口信息调用云网络服务的SOAP/RESTful API创建虚拟网口对象，根据云网络服务返回的虚拟网口信息通过在物理主机上运行的虚拟网桥或者虚拟交换机设备创建对应的虚拟端口。
- 代理模块使用物理主机上云主机对应的虚拟磁盘ID和虚拟端口ID以及CPU、内存信息生成的云主机启动配置文件，调用虚拟机管理器创建云主机。
- 裸金属云主机场景下不需要上述两步虚拟端口、虚拟机设备的创建，可依赖云网络服务完成裸金属的物理网络配置。

2.4.2 云网络服务

云网络服务通常包括以下几方面内容。

一、虚拟层2网络连接服务

虚拟层2网络连接服务对应于传统网络中的VLAN。云化之后，为了实现和硬件物理网络解耦，业界引入了Overlay网络的概念，因此，云网络中的L2网络服务除了VLAN，还会有VxLAN、NVGRE、STT等多种类型的二层网络。从用户的角度来看，其不关注具体的虚拟网络实现，因此，在有的云服务界面上看到的L2网络有时候会以子网Subnet来代替，创建子网时可以指定是否开启DHCP、可分配的IP地址、网关、DNS Server等信息。

二、虚拟层3网络服务

对应于传统网络中的网关和路由功能实体，比如路由器，完成跨子网之间的IP路由转发。云网络中，通常会划分两种类型的层3网络服务：(1)扁平网络，也就是不同的租户/用户共享一个公共的L3网络，他们之间路由可达，网络互通；(2)多租户网络，不同租户/用户之间的网络互相隔离，无法直接互访，类似传统网络中的

VRF(Virtual Route Forward)。L3网络服务还包括NAT服务，在某些云服务系统中也称之为弹性IP服务或者Floating IP服务。

三、虚拟专用网络(VPN)服务

VPN服务和传统网络中的VPN互联功能基本一致，通常包括IPSec VPN、SSL VPN、PPTP VPN等。

四、负载均衡服务

负载均衡服务也称为弹性负载均衡(Elastic Load Balance)服务或者软件负载均衡服务(Service Load Balance)。根据前端虚拟IP(Virtual IP)是否使用公网，可以区分为面向公网的外部ELB服务和面向私网的内部ELB服务。有的云还会提供自动伸缩组(Auto Scaling Group)服务，通常也基于ELB服务实现：把负载均衡作为前端Server入口，自动伸缩的负载单元作为后端Member，通过后端Member数量的扩减容，实现按负载的自动化伸缩。

五、虚拟防火墙及安全组隔离服务

该服务所提供的安全隔离与策略控制能力与传统防火墙相似，主要完成不同安全等级的网络隔离和访问控制功能。在实现上，有不同的业务呈现方式和部署形态，通常包括集中式的虚拟防火墙安全接入控制列表(ACL)和分布式安全组(Security Group)两种较为典型的业务方式。

从用户视角来看，典型的网络服务场景如图2-35所示。

通常，一个云计算系统的网络服务在架构实现上会包含5个层次的功能，如图2-36所示。

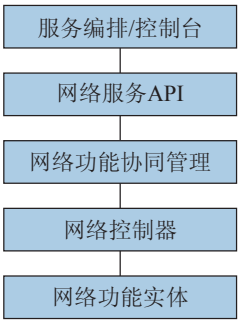


图2-36　云计算系统中的网络服务架构定义

不同的云服务系统因为具体实现方式的不同，在解决方案落地时会有所调整。比如，开源软件OpenStack的网络服务Neutron就同时具备服务API、协同管理、网络控制器三个功能。下面以Neutron为例，看看这个虚拟的多租户网络架构是如何实现的。

首先，云网络服务需要和具体的网络对象相对应，以便最终映射到物理网络功能实体中。

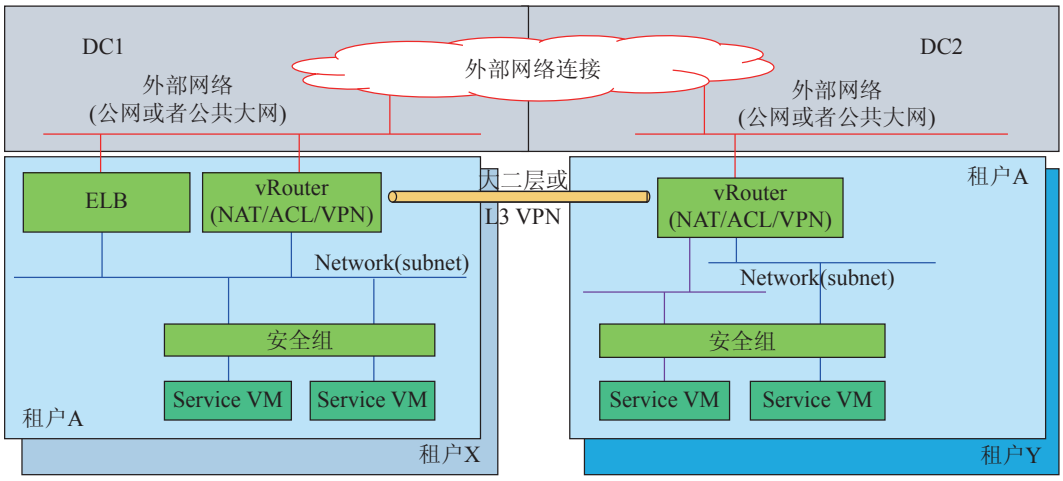


图2-35　租户网络服务典型视图

Neutron的网络对象模型如图2-37所示。

服务编排功能体首先将用户的编排结果拆解成对应的Neutron的网络对象，然后调用Neutron的API，由Neutron将这些网络对象映射到具体的物理网络实体上。比如，把Subnet的网关映射到相应的vRouter实例去实现路由转发，把firewall_policy映射到相应的vFW实例去实现访问控制，把Floating IP映射到相应的vRouter实例或者vFW实例去实现NAT地址转换。

Neutron的整体架构，如图2-38所示。

在数据面的实现方案和部署形态上，Neutron社区提供了多种实现参考。比较典型的解决方案是：OVS部署在计算节点上，VM通过Neutron port对象连接OVS，然后再通过OVS上联到vRouter。vRouter物理部署放在网络节点NetworkNode上，通过Namespace对不同租户的vRouter实例进行隔离。虚拟负载均衡器vLB也部署在网络节点NetworkNode上，同样采用Namespace对不同租户

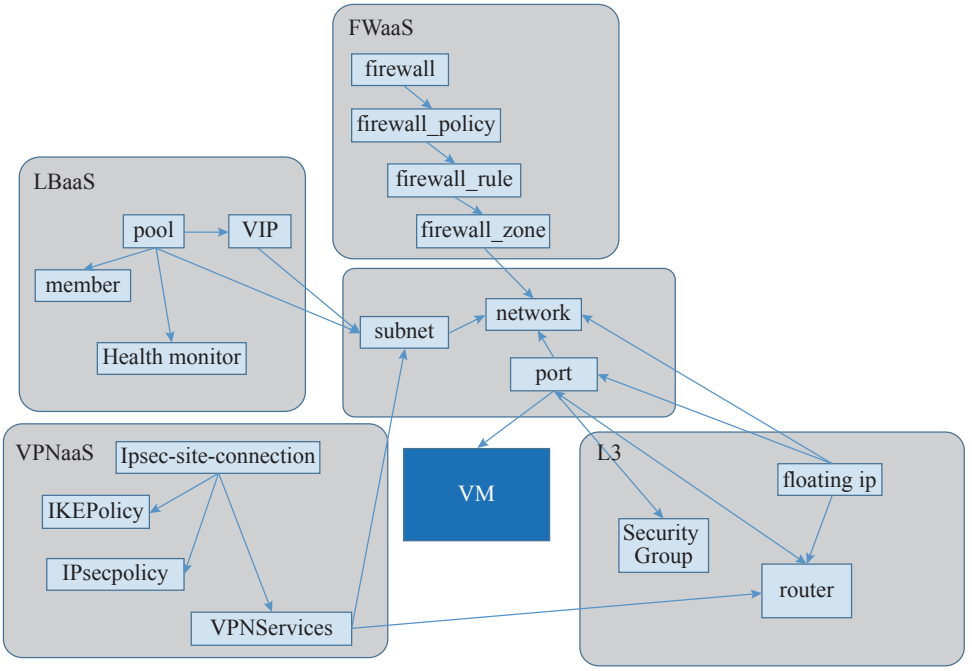


图2-37　Neutron的对象模型

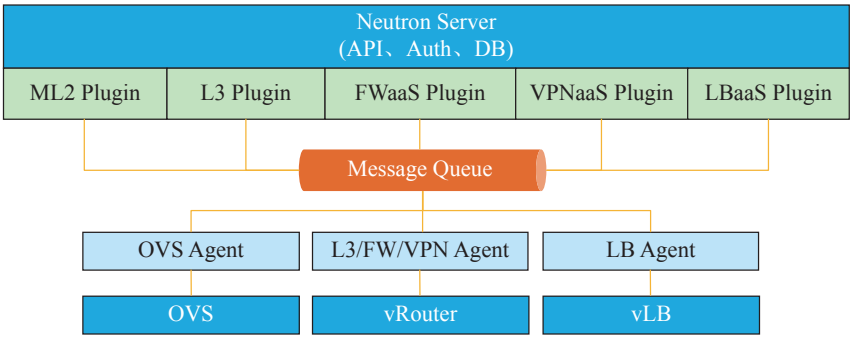


图2-38　Neutron架构

的vLB实例进行隔离(见图2-39)。

在具体的云计算系统中，即使网络管理采用Neutron，数据面的实现上也可以采用不同的网络设备，比如硬件交换机做vRouter、硬件防火墙做vFW、硬件负载均衡器做vLB。这些网络设备都可以通过Neutron的Plugin机制接入到Neutron并且协同工作。

2.4.3 块存储、对象存储及文件存储服务

基于云计算总体架构下的存储云解决方案，如图2-40所示。

云存储解决方案依托云计算平台的弹性存储、分布式对象存储，以及操作运维及业务发放管理系统等功能，通过集成第三方的企业在线备份软件、个人网盘、个人媒体上载及共享类软

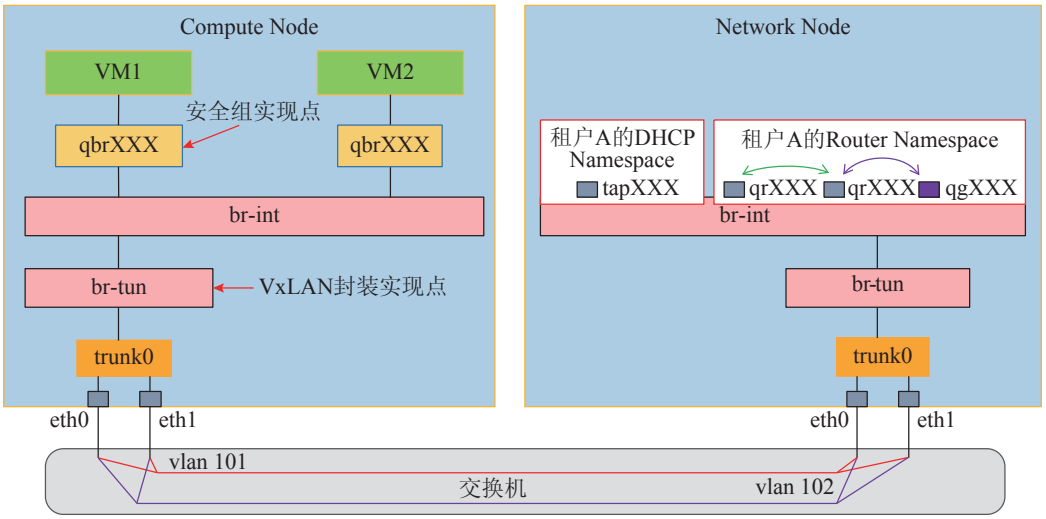


图2-39　OpenStack的计算节点和网络节点网络参考架构

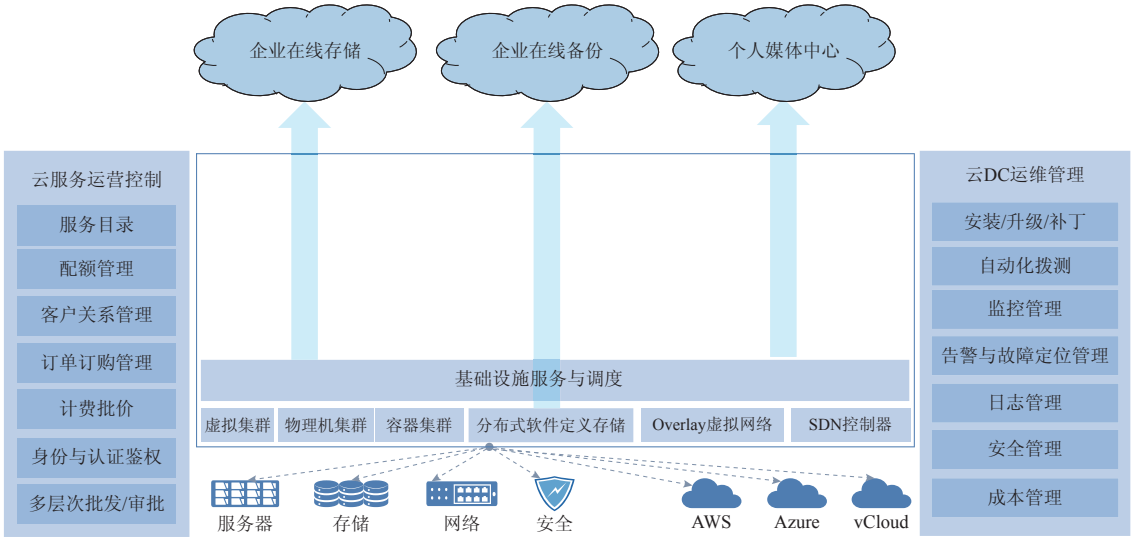


图2-40　存储云解决方案架构子系统组合

件，允许云存储运营商提供面向个人消费者用户的廉价/高性能网络存储服务，以及面向企业用户的在线备份及恢复类服务。

在云存储平台与企业备份/恢复类应用软件以及个人网盘、个人媒体上载/共享类软件绑定部署销售的场景下，与云存储应用用户的Portal、UI交互界面及其与应用相关的核心业务功能(如权限管理、断点续传等)由第三方合作的应用软件支撑，同时从服务器端或直接从客户端调用“弹性存储”服务的对象存储或分布式文件系统API (OBS/POSIX)，实现对云存储用户的高吞吐量、超大容量存储内容的读写及其元数据管理。该模式下，用户的计费主要由第三方软件负责。

运营商的云存储平台以IaaS形式提供与第三方合作的企业备份/恢复类应用软件，以及个人网盘、个人媒体上载/共享类软件的后端支撑，此时“弹性存储”提供对第三方云存储应用软件的多租户隔离以及存储空间和IOPS/MBPS访问流量的精确计量，以便为云存储服务商与第三方增值服务提供商之间的计费结算与商业分成提供支撑。

“弹性存储”提供以通用服务器及其硬盘为基础的全分布式平台，具备水平无级扩展、超大容量等特点，并通过瘦分配、跨用户的重复数据删除、数据压缩等大幅降低云存储的设备及运维成本，实现超高性价比存储方案，提升云存储类业务的利润空间。

“弹性存储”所提供的块存储跨服务器、跨机柜的数据可靠性冗余机制(2份或3份拷贝)可以为存储云功能提供超越PC本地存储的数据可靠性保障，同时基于对象存储的快照机制和异地容灾机制，使得存储云数据在更大灾难发生时也有机会还原为最近一次快照时刻的存储数据内容。

2.4.4 桌面云服务

基于云计算总体架构下的桌面云解决方案如图2-41所示。

桌面云解决方案主要基于云计算平台的弹性计算、弹性存储、操作运维及业务发放管理系统功能，通过集成桌面云会话控制管理及远程虚拟桌面控制代理等模块，提供针对企业内部应用的呼叫中心桌面云、营业厅终端桌面云、Office办公桌面云解决方案，以及面向公众网用户的VDI出租业务。

其主要业务流程包括如下几点。

✍ 来自企业IT系统或运营商BOSS系统的桌面云发放命令，通过与云计算云管理平台之间的

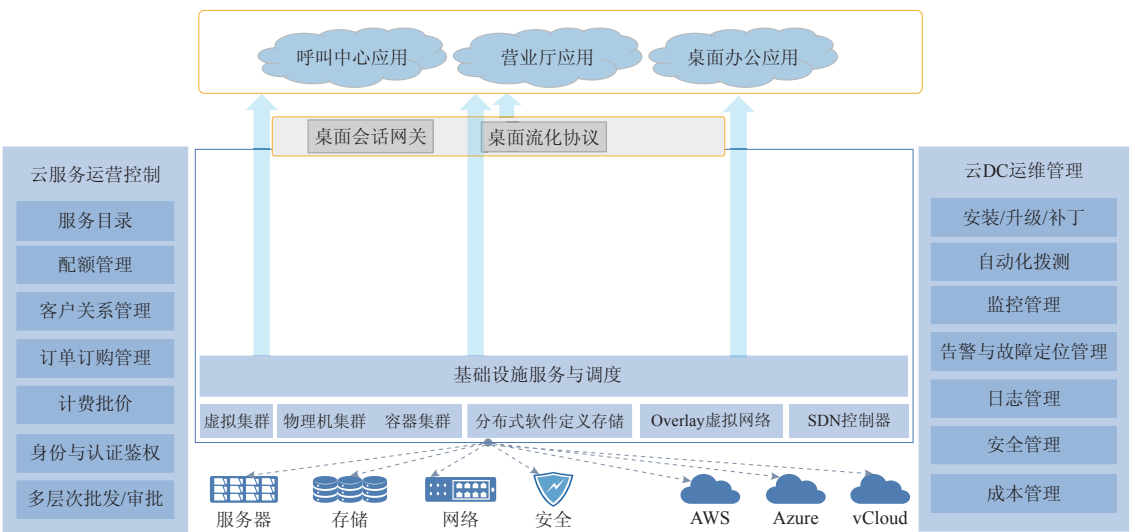


图2-41 桌面云解决方案架构子系统组合

SOAP、RESTful API接口，交互包含标准桌面配置规格定义的VDI业务发放/撤销封状式命令。

✍ 云管理平台的BSS系统对VDI业务发放命令进行解析，将该命令分解为指向“桌面会话网关”的VDI账户发放命令，以及指向“弹性计算API及集群调度”的VDI虚拟机实例发放命令。

✍ “桌面会话网关”接受来自云管理平台的命令创建桌面账户。

✍ “弹性计算”服务部分接受来自云管理平台BSS部分通过的EC2兼容接口创建符合原始发放需求规格、包含虚拟桌面服务器端代理的虚拟机镜像；桌面服务端的EC2 IP地址还将反馈给“桌面会话网关”系统。

✍ “桌面会话网关”接收来自VDI瘦终端的HTTP/HTTPS桌面会话登录请求，通过云计算API与AAA服务器交互，或者与企业LDAP/AD服务目录交互进行用户身份鉴权，随后进一步通过EC2 API通知弹性计算服务启动虚拟机的运行实例。

✍ “弹性计算”服务通过与“弹性存储”以SOAP消息交互，完成指定业务发放命令中指定的虚拟存储卷的挂载，其中包括了为该虚拟分配的系统卷及数据卷，虚拟机从其系统卷引导启动，完成系统的初始化启动。

✍ 用户认证通过后，将用户所用VDI瘦终端通过远程桌面协议与数据中心服务器相连接，建立桌面会话。

✍ 来自瘦终端的客户操作通过“桌面流化协议”实现与其后端服务的虚拟机进行交互，完成实际的操作动作。

✍ 构成桌面云解决方案的弹性计算管理服务器、块存储及对象存储设备，VDI虚拟机、桌面云会话控制网关、数据中心各层交换机(接入/汇聚/骨干)、防火墙以及桌面云特有的负载均衡及Cache加速设备(LBS)相关的监控、告警、性能、配置、拓扑以及安全管理等，均通过符合SOA原则的Web OM对象化API接受云管理子系统的端到端管理。

✍ “弹性计算”的智能调度算法，可以为有效提升桌面云VDI VM在整体服务器集群内的资源利用效率，减少负载不均衡导致的资源浪费并提升节能减排效率，以及实现桌面云工作负载与其他非桌面类工作负载的动态调度能力提供支撑。

✍ 由于“弹性计算”内所有服务器及虚拟机共享相同的“弹性存储”实例，使得支持业界最大规模的虚拟机HA及在线热迁移的集群成为可能，在集群内的服务器故障可以快速恢复，从而实现了软硬件故障导致的VDI服务故障影响最小化。

✍ 由于“弹性存储”所提供的块存储跨服务器、跨机柜的数据可靠性冗余机制(2份或3份拷贝)，可以为桌面云功能提供超越PC本地存储的数据可靠性保障，同时基于对象存储的快照机制，以及异地容灾机制，使得桌面云数据在更大灾难发生时也有机会还原为最近一次快照时刻的存储数据内容。

✍ 针对桌面办公类应用，采用软件预安装模式，除基本客户机操作系统外，进一步增加终端安全管理、Office系列、UC通讯、Email、CRM、ERP等预装软件，并可根据不同目标市场，制作不同的虚拟机模板，可以在云管理平台中指定不同虚拟机、存储及网络带宽规格，甚至不同的计费规则(针对公有云桌面出租的场景)。

2.4.5 IDC托管云

基于云计算总体架构下的IDC托管云解决方案，如图2-42所示。

IDC托管云解决方案依托云计算平台的弹性计算集群、弹性存储集群、分布式结构化存储服务以及分布式消息队列服务，为IDC(Internet数据中心)运营商提供ISP/ICP多租户的计算与存储资源的托管服务。

业务应用与IT子系统运行于IDC托管云中，隶属于不同的第三方ISP/ICP以及企业。IDC托管云依托于云平台的自动化、虚拟化基础能力，实现多租户的分权分域的安全隔离及资源共享，相比传统的物理服务器独占式的IDC解决方案，其可提供高出3～5倍的IDC出租资源利用效率，从而有效提升IDC托管类业务的利润率。

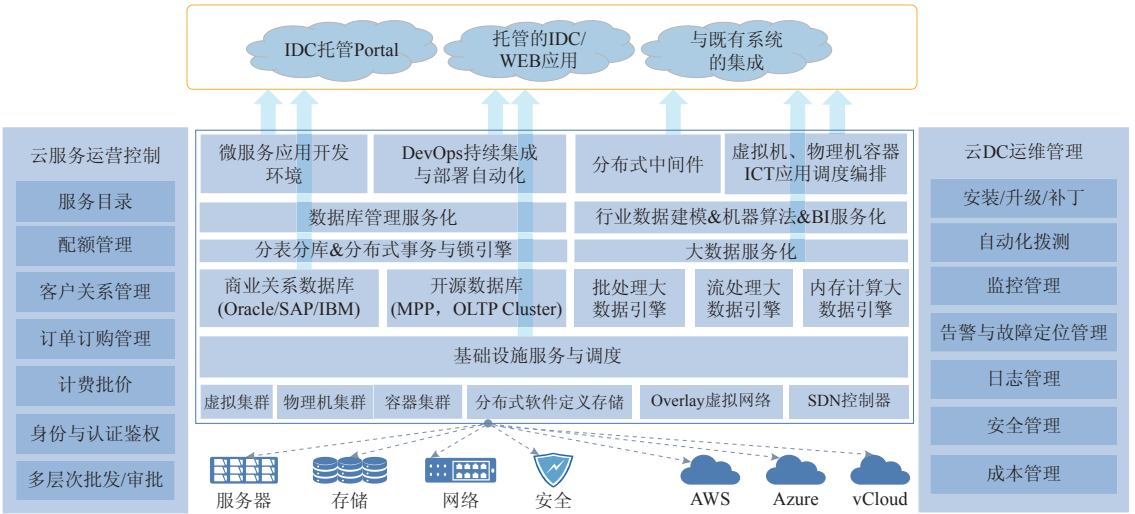


图2-42　IDC托管云解决方案架构子系统组合

云计算中云管理平台的BSS子系统为IDC业务的运营发放、资费定价、后计费、实时付费(按资源规格、按时长、按流量，以及上述各类维度的综合)提供了强大的后台支撑。如果用户(运营商)已有BSS系统，可通过云计算API(EC2/S3等的兼容API)与用户已有的后台BSS系统进行对接。

除虚拟化计算集群资源(含虚拟CPU与内存资源，以及挂载于该虚拟机实例之下的系统卷及数据卷块存储资源)之外，云计算平台还提供独立的分布式对象结构化存储，分布式消息队列等超越单机物理处理能力范畴的分布式中间件服务(针对运行于云平台之上的软件)，以及远程接入的服务能力“虚拟桌面”(针对云平台业务的直接消费者)。

云计算平台对IDC托管的业务应用的管理是通过业务应用底层的操作系统来完成的。这些操作系统一般为x86架构，如Windows、Linux以及Unix。IDC托管的业务应用也可以与操作系统一起打包作为一体化的虚拟机镜像使用。只要IDC托管应用本身的颗粒度不超过一个物理服务器的场景，则不存在软件兼容性问题，但需要IDC托管应用的软件管理系统实现与云计算平台API的集成，实现软件安装部署及监控维护从硬件平台到云平台的迁移，并可能依赖“运营维管”子系统的自动化应用部署引擎，实现跨越多个应用虚拟机镜像的复杂拓扑连接的默认模板化自动部署，从而有效提升大型分布软件的部署效率，这是目前云计算IDC托管的主流形态。

针对分布式对象存储、分布式消息队列、分布式列存储数据库等场景，则需要IDC托管应用针对其业务层API进行适配改造，相对难度较高，IDC托管应用一般是云计算平台生态战略联盟内的云应用合作伙伴。

2.4.6　企业私有云

基于云计算总体架构下的企业私有云解决方案，如图2-43所示。

伴随IT与网络技术的飞速发展，IT信息系统对于企业运作效率、核心竞争力，以及企业透明化治理正在起着越来越重要和无可替代的作用，而企业信息集中化、企业核心信息资产与商业逻辑的规模越来越庞大，跨不同厂家IT软硬件产品的集成复杂度不断增加。企业IT系统的架构正在从传统的与特定厂家硬件平台及管理系统绑定的客户端/服务器(B/S、C/S)架构向更为集中化的统一整合平台架构的方向演进。云计算平台与企业

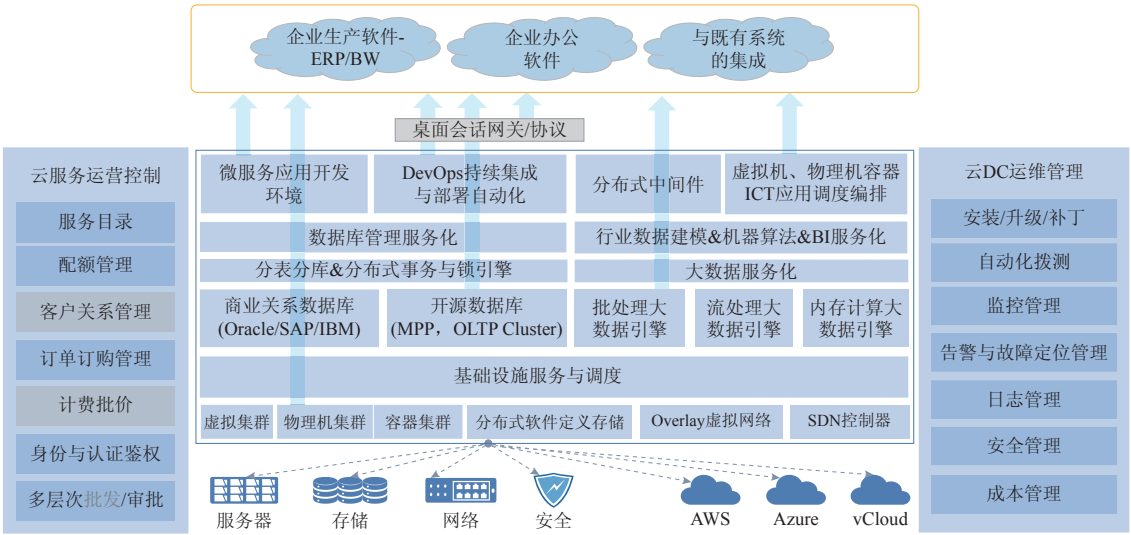


图2-43　企业私有云解决方案架构子系统组合

IT应用层软件的结合，尤其是基于虚拟机的弹性计算服务、虚拟网络服务、虚拟桌面服务、分布式块存储、对象存储服务、文件系统服务以及与之配套的自动化运维管控的能力，使得企业IT系统可以更高效地支撑企业核心业务的敏捷运作，大幅提升IT及机房基础设施利用效率并实现节能减排。

在保障业务运行效率与性能不下降的前提下(计算、存储资源配额，业务访问时延等)，通过将原有直接运行在x86服务器硬件平台之上的企业IT软件迁移到虚拟化平台，将企业IT软件相关的存储数据(数据库/文件格式)迁移到分布式块存储或者传统IP-SAN存储，可以充分利用弹性计算平台的跨服务器边界的资源分配与热迁移能力，实现多个相对独立的IT软件应用在虚拟机资源池内动态共享，以及削峰错谷的负载均衡调度，并实现不同应用的分级QoS(硬件资源下限)策略保障，实现IT资源利用效率从平均20%～30%到60%～70%的提升。同时在系统轻载的情况下，通过将轻载虚拟机迁移到少数物理服务器，可实现更多空闲服务器硬件的自动休眠，来最大限度地提升数据中心及IT资源池的节能减排效率。

借助云计算平台的虚拟桌面(即桌面云)能力，可以实现企业员工PC办公的计算与存储能力向数据中心的集中化迁移，实现核心信息资产与用户接入访问终端的解耦和剥离。虚拟桌面具有绿色、节能、安全隔离及移动接入能力方面的优势。除了对办公PC的改造之外，虚拟桌面也是最终企业员工接入到后端IT应用业务的必由界面和通道。

借助面向大型分布式应用软件的云计算自动化、模板化部署，通过故障自动修复管理能力，运行态自动伸缩管理工具，弹性计算、虚拟网络、虚拟桌面与企业IT管理系统(含可选的ITIL子系统)的无缝集成，可以实现IT应用软件与底层IT硬件和网络基础设施的彻底解耦，利用标准化的虚拟应用部署模板(描述格式如OVF)大幅度(70%)缩短IT软件应用的上线部署效率，以及降低业务在线运营的容量规划与故障维护的复杂度，有效提升IT服务支持企业核心业务的SLA水平和效率，从而促进企业生产率的同步提升。

云计算的分布式对象存储、半结构化存储(列存储数据库)以及消息队列能力，对于企业私有云来说，是可选的高层云平台能力。其适用于企业定制开发新型应用，比如：企业/行业搜索引擎，基于企业IT系统海量日志或统计类数据仓库的商

业智能挖掘与分析，以便指导企业的业务规划策略的调整优化等以大数据集作为输入和输出的软件，是性价比最优的选择。但这部分云平台能力在企业私有云中一般无法适用于面向实时在线事务及交易类的应用形态。原因是这些云平台的API与单机通用操作系统(Windows、Linux、Unix等)下的文件系统、进程间通信以及数据库访问API都是不兼容的，而业界大多数企业IT应用软件、商业操作系统以及数据库(如Oracle)软件是运行在通用操作系统之上的。

2.4.7 大数据分析云服务

基于云计算总体架构下的大数据分析云解决方案，如图2-44所示。

大数据分析云解决方案为海量静态数据批处理以及大流量动态流数据处理为关键特征的企业及行业应用场景提供支撑，通过自动化提取与归纳价值信息实现业务增值。大数据分析云由云计算的并行数据分析与挖掘平台所支撑，可充分利用云计算底层能力创造最大价值。

在海量静态数据批处理的场景下，大数据分析平台需要充分分析经过相当长一段时间积累的、存储容量庞大的历史数据(如话单、日志、话统信息等)。大数据分析平台的并行数据处理引擎进一步依赖于弹性计算集群、弹性存储服务、分布式结构化存储服务以及分布式消息队列服务，为诸如互联网电子商务网站用户、电信运营商的BSS/OSS系统、视频娱乐类网站、搜索类网站等提供服务。大数据分析平台所提供的服务类型包括：信息库精细化搜索、用户消费行为日志分析、系统运行日志分析以及集中监控信令信息的智能分析和挖掘。这些大数据分析服务为精确定位广告推送、网络运维优化、基于用户消费趋势分析的销售策略等商业运营提供策略决策性支撑。

在流量动态数据流处理的场景下，其关键特征在于对来自于数量庞大的信息源所产生的动态事件与动态数据(比如来自电信网络实时检测的信令信息、来自于众多车辆GPS的定位信息、来自物联网终端实时采集的信息等)，在一个相对较短的时间窗口内，进行动态数据的流水线方式自动关联分析与处理，并给出及时、准确和智能的执行策略决策，为特定业务目标服务(如大规模智能交通云、物流云的构建)。与上节数据批处理在分割、合并与混排等中间步骤所涉及的大量持久化存储I/O交互方面的特征相比，其最大差异处在

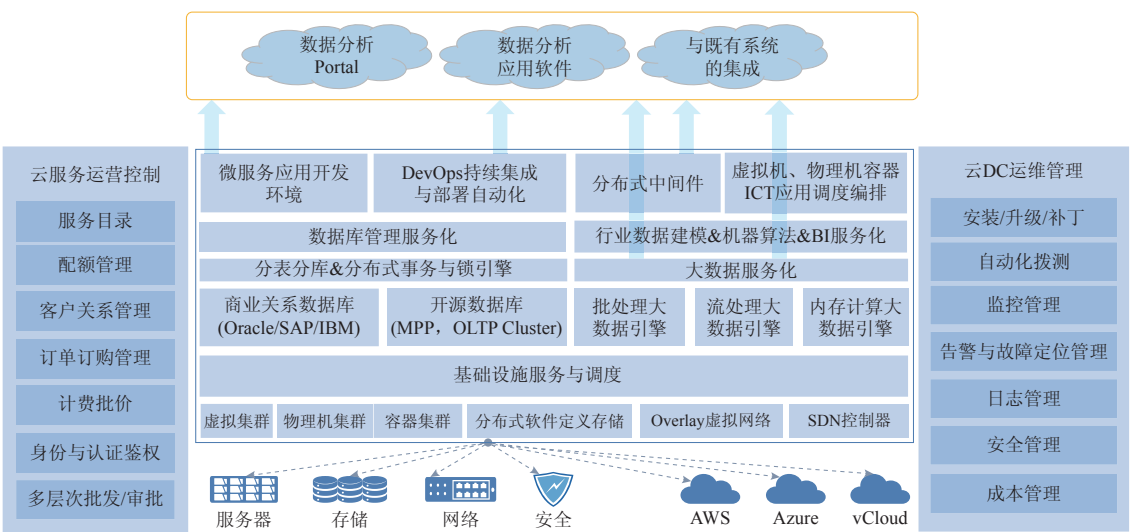


图2-44 大数据分析云解决方案架构子系统组合

于，数据流处理过程更讲究处理的及时性与敏捷控制能力，因此处理过程主要在内存中完成。流处理与批处理可以统一在相同的框架引擎之下。

为便于广大第三方应用开发编程人员以及云计算平台生态系统的合作伙伴充分独立于海量数据批处理以及流处理业务的内部实现架构细节，可在并行数据分析引擎与并发应用之间设置SQL/类SQL适配与翻译层，提供开发人员所熟知的SQL或类SQL规范语言进行海量数据的操作。

2.4.8 数据库云服务

基于云计算总体架构下的数据库云解决方案，如图2-45所示。

数据库云主要指基于云平台构建的结构化/半结构化数据库处理系统。

数据库云可以基于虚拟化平台，也可以基于物理平台直接构建。

在性能要求高的场景下，一般基于物理平台构建，系统无须弹性计算部分(图中虚线表现)；而在要求容量很大、应用用户很多的情况下，则可采用基于虚拟化平台构建的形式。

数据库云一般以数据库一体机的形态出现，会在以下方面做一些增强。

- 数据库加速：为取得更好的数据库性能，会在硬件层、弹性存储层做垂直层面的深入调优，例如采用读写更快的SSD卡，采用面向数据库独特的读写算法。
- 数据库加固：为保证数据库数据不丢失、不损坏，会在中间件服务层增加数据库的备份/恢复、容灾、定期校验等服务，提高数据的可用性。

2.4.9 媒体云服务

基于云计算总体架构下的媒体云解决方案，如图2-46所示。

媒体云解决方案依托云计算平台的弹性计算集群、弹性存储集群、分布式结构化存储服务以及分布式消息队列服务，为广电系统电视台企业提供高效率的媒体存储、编辑及缓存加速的计算与存储资源托管服务。

媒体云采用云平台的关键驱动力，来源于面向公共媒体传播的广播电视行业(包括国家及地方电视台、广播电台)从模拟传统卡带式存储向全面的数字媒体化方向的发展演进。在此过程中，数字化内容与云数据信息需进行大规模集约化存储、处理以及分发。这些需求包括媒体的采集、

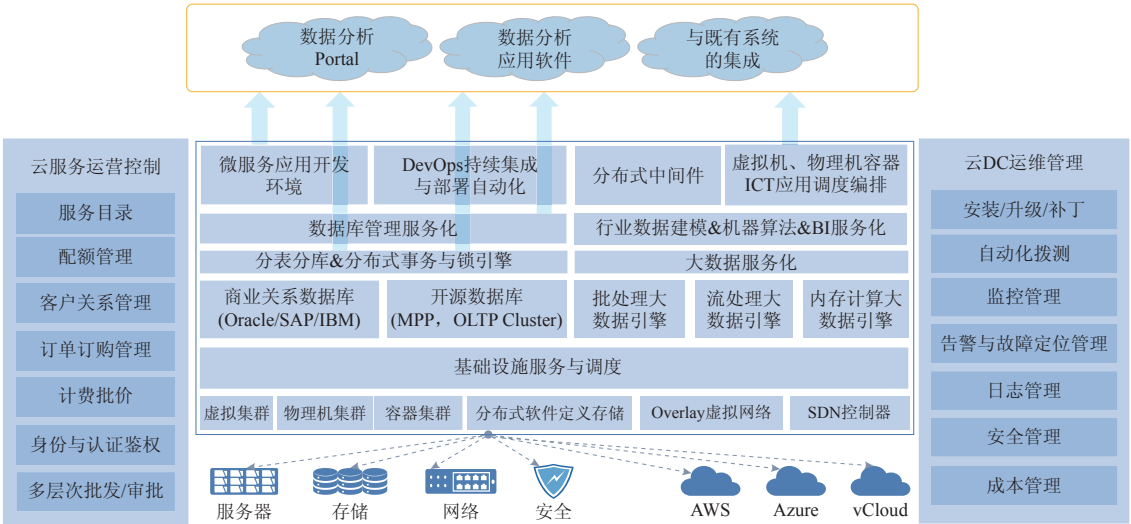


图2-45 数据库云解决方案架构子系统组合

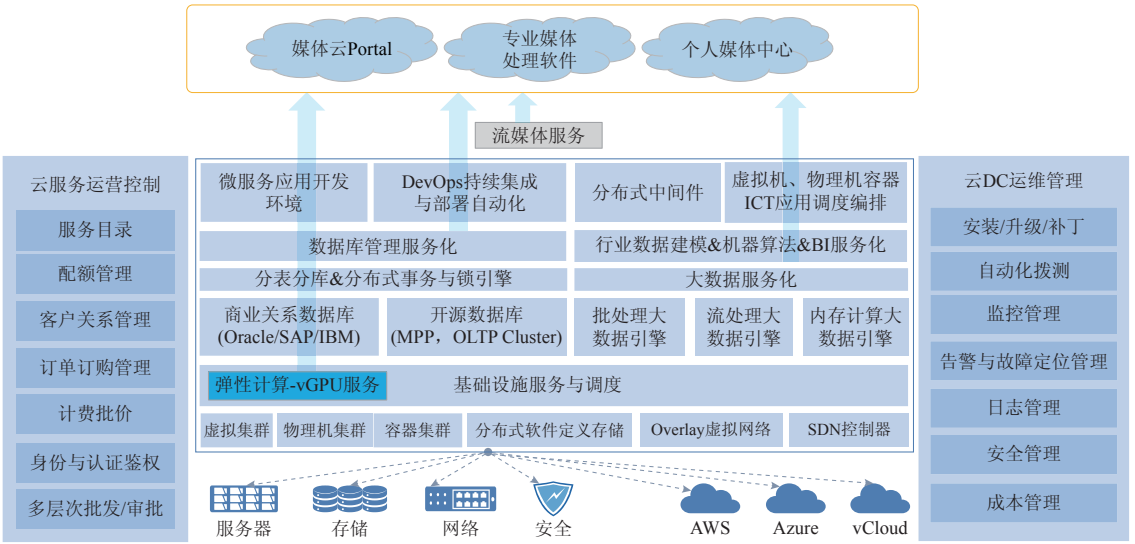


图2-46　媒体云解决方案架构子系统组合

编辑、播放控制、音视频媒体转码、管理等各方面的软件处理需求。目前国内外业界(如中科大洋、索贝等)已纷纷推出针对媒体采集编播的专业化软件，这些软件重点聚焦于解决媒体内容资源的应用层的业务处理，但在对底层的服务器及存储硬件基础能力提升方面仍然缺乏积累，如增强服务器存储硬件的使用效率，如何以更低的成本提供更大的存储空间、处理能力和I/O吞吐带宽，可靠性容灾能力，以及按照业务需求分配弹性可伸缩的资源池等。只有上述专业化的媒体管理软件与云平台能力紧密结合，才能实现媒体数据中心基础设施硬件资源利用率的提升，实现节能减排，获得最优的较硬件整体性价比。

云计算平台对于媒体云数据中心的核心价值在于以下几点。

✍ 海量存储能力：来源于各种片源的音视频媒体内容信息本身的持久化存储，由于数据量庞大、存储周期长(3个月)，采用传统的Scale-up模式的RAID控制方式存储在性价比方面已越来越难以满足媒体云规模化运营的需求。同时，媒体云的多项业务，尤其是视频/音频媒体的编辑制作以及在线播放，都对存储与计算资源之间的高I/O吞吐率(IOPS/MBPS)提出了更高的要求。满足这些需求依赖于引入支持无级水平扩展的Scale-out存储。

✍ 跨业务共享的计算、存储资源共享与均衡：针对媒体云应用层软件的不同业务类别(采、编、播、存、管等)在同一时间段的资源占用情况差别巨大的特点，通过引入云平台，采用统一的资源池平台支撑资源的自动伸缩、动态错峰削谷与负载均衡，通过SOA Web Service/REST接口与应用层软件交互实现自动化的资源管理能力，平均资源占用率可以从20%提升到70%以上。

✍ 并行计算与海量处理能力：针对媒体云普遍所需的不同的视频格式之间的动态转换(TS、H.264、MP4、AVI、RMVB等)需求，由于媒体文件尺寸庞大，计算能力需求密度高，通过最大限度提高编解码的并行度，可充分利用可获得计算资源对大媒体文件分而治之，有效缩短编码处理所需时间，以资源换取时间，大幅度提升业务处理效率。其他诸如广告推送等增值业务，也可依赖于并行数据分析与处理平台实现基于用户消费行为历史数据分析的智能化与自动化的广告策略制定与发布。

分布式缓存与加速：为缓解广大互联网用户点播热点多媒体内容引发巨量带宽需求与有限互联网广域连接带宽的矛盾，需要在媒体云数据中心对媒体内容进行分片，并自动识别用户访问热点。热点内容被自动推送到分布式网络节点缓存，供用户就近访问，从而缓解集中访问的带宽压力。缓存的内容与云数据中心的源内容会定期同步，以保证用户看到的内容为最新内容。

2.4.10 电信NFV云

基于云计算总体架构下的电信NFV云解决方案，如图2-47所示。

NFV(Network Function Virtualization网络功能虚拟化)旨在通过研究标准IT虚拟化技术，使得电信网络设备的功能能够以软件方式运行在符合行业标准的大容量通用的服务器、交换机和存储设备中去。这里的软件可以根据需要在网络中不同位置的硬件上安装和卸载，不需要安装新的硬件设备。简单地说，NFV就是在平台层引入云计算平台，实现电信网元纯软件化和硬件设备解耦。

电信NFV云通过引入云计算平台，主要要解决运营商如下几大问题。

一、提高硬件投资收益比

简单来说，采用通用硬件一方面可以大大降低成本，另一方面开通新的业务时也不需要更换硬件，只要升级相应的软件即可。

二、提高业务部署的灵活性

由于NFV使得软件和硬件解耦，也就是说运营商的业务可以灵活部署在不同机框和不同机房，在不同地域的硬件上部署，那么相比传统业务部署来说自然是极大地提高了灵活性。

三、快速部署业务

软硬件解耦带来的最大好处就是设备商可以专注于纯软件层面的研发，在已有的软硬件部署下，新的业务研发周期会大大缩短，运营商也能尝到快速部署业务的甜头。

四、自动扩容和节能减排

由于虚拟化技术的支撑，网络智能调度资源的能力大幅提升。在业务压力增加时网络智能调度系统可以通过自动增加网络资源来缓解业务压力；在业务闲暇时可以通过自动地删减网络资源实现节能减排。

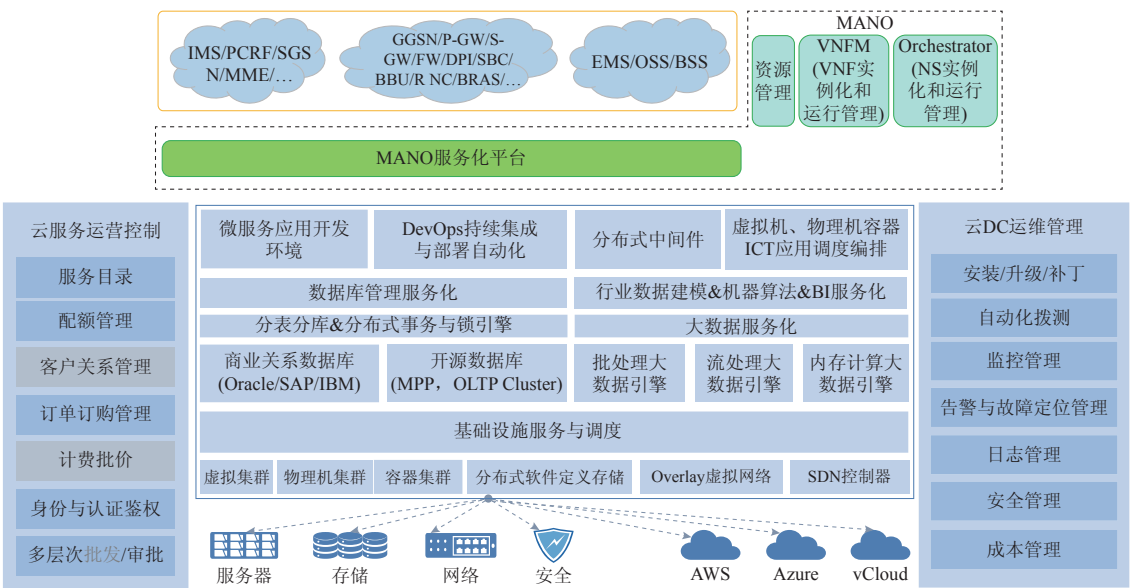


图2-47　电信NFV云解决方案架构子系统组合

五、降低设备商的准入门槛

硬件设备通用了，软件接口也通用了，毫无疑问，设备商的准入门槛也降低了，这给电信服务引入了更多的竞争，这些对于运营商来说，都是最想看到的。

NFV希望电信网元纯软件化，同时又需要提供电信级5个9的高可靠性服务，因此对云计算平台需要有更高的要求。

云计算平台对于电信NFV云的核心价值在于以下几点。

- 电信应用由于可靠性要求，对应用的虚拟机之间的亲和性关系有着不同的约束，云计算平台支持电信应用的各种亲和性调度需求。
- 电信级服务意味着5个9的高可靠性，对于服务中断时间有着严苛的要求，通过硬件故障快速检测和故障快速通知技术，电信应用能够快速感知故障，并及时进行自愈机制的启动，减少业务中断时间。
- 作为NFV基础设施，提供与应用无关的高可靠性特性：HA(High Availability)特性主要在服务器出故障时提供虚拟机冷备机制，轻量级FT(Fault Tolerance)为面向网络I/O的应用提供热备机制以及跨数据中心的虚拟机容灾机制。
- 针对NFV进行云计算转发面的优化，使得VM到VM之间的转发性能可提升到可以满足电信网元的要求。
- 云计算平台利用存储虚拟化技术把所有服务器本地HDD、SAN、NAS组合起来构建超大规模存储资源池，利用服务器的内存/Flash/SSD构建分布式高性能大容量近端cache网格，服务器机头可同步横向扩展，加速I/O读写能力。同时，云存储资源池可提供基于SLA的分级存储服务。
- 云计算的自动化和模板化(TOSCA、HOT、CloudFormation、OVF等)特性可大幅度提高电信应用和IT软件应用的上线效率。云计算的故障自动修复和自动伸缩管理能力可大幅度降低业务在线运营与故障维护的复杂度。

电信NFV是大规模复杂系统，从电信应用到IT的各种应用，包罗万象，服务海量用户，在线生成海量数据。云计算的结构化数据库、分布式对象存储、半结构化存储(列存储数据库)以及消息队列等中间件能力，为电信NFV带来广泛的运营基础能力，满足日益增长的新老应用的基础能力需求，加快新业务和新运营应用的上线能力。

第 3 章

云计算及大数据开源软件概览

3.1 OpenStack概述

OpenStack是目前最为流行的开源云操作系统框架。自2010年6月首次发布以来，经过数以千计的开发者和数以万计的使用者的共同努力，OpenStack不断成长，日渐成熟。目前，OpenStack的功能强大而丰富，已经在私有云、公有云、NFV等多个领域得到了日益广泛的生产应用。

与此同时，OpenStack已经受到了IT业界几乎所有主流厂商的关注与支持，并催生出大量提供相关产品和服务的创业企业，在事实上成为了开源云计算领域的主流标准。时至今日，围绕OpenStack已经形成了一个繁荣而影响深远的生态系统，OpenStack已经是云计算时代一个无法回避的关键话题。可以说，不了解OpenStack，就无法理解当今云计算技术的发展，也无法把握云计算产业的脉搏。因此，本章将对OpenStack的相关要点进行概括介绍。

3.1.1 OpenStack概念辨析

目前，与OpenStack相关的书籍与文章俯拾皆是，但其中的内容质量不一。对于读者而言，也很容易受到一些错误信息的误导。为此，本节将对OpenStack相关的一些核心概念进行介绍与澄清。

一、什么是OpenStack

OpenStack，是目前最为流行的开源云操作系统框架。想要深入理解OpenStack是什么，则需要围绕开源、云、操作系统、框架这几个关键词展开说明。

1. 云

针对什么是云，业界已有充分的论述，此处不再深入展开。读者只需明确，OpenStack是用来构建云计算系统的核心软件组件即可。

2. 云操作系统，是面向云计算的操作系统

云操作系统这个概念，对于许多读者来说，可能还比较陌生。在此，我们将通过与操作系统的类比，来帮助读者理解何谓云操作系统。

操作系统，是计算机系统领域里一个至关重要的概念。有了操作系统，我们才能将计算机系统中的各类软硬件整合起来，形成一个能够完成各类处理任务的完整系统，为用户提供服务。这个描述较为抽象，但结合到日常生活与工作中的实例，就清楚易懂多了。无论是服务器和个人电脑上的Linux、Windows，还是手机上的Android、iOS，都是操作系统的常见实例。

无论是服务器、个人电脑还是手机上的操作系统，本质上，其核心功能都可以概括为五个方面，即资源接入与抽象、资源分配与调度、应用生命周期管理、系统管理维护和人机交互支持。换言之，只有具备了以上这五个方面的主要功能，一个操作系统才能够实现各类软硬件的整合，让系统具备为用户提供服务的能力。

具体而言：(1)资源接入与抽象，是指将各类硬件设备，如CPU、内存、本地硬盘、网卡等，接入到系统中，并将其抽象为操作系统可以识别的逻辑资源，以此作为操作系统对各类硬件资源实施管理的基础；(2)资源分配与调度，是指利用操作系统的资源管理能力，将前述的不同硬件资源，按照需求的类型和数量，分配给不同的系统软件或应用软件，供其使用；(3)应用生命周期管理，是指协助用户实现各类应用软件在操作系统上的安装、升级、启动、停止、卸载等管理操作；(4)系统管理维护，是指协助系统管理员实现对系统自身的各类配置、监控、升级等管理操作；(5)人机交互支持，指提供必要的人机界面，支持系统管理员和用户对系统实施各类操作。

与之对应，一个完整的云操作系统，同样应该能够具备上述五个方面的主要功能。其核心区别只是在于，云操作系统需要管理的，是一个由大量软硬件组成的分布式的云计算系统，而一个普通操作系统需要管理的，则是一台服务器、一台个人电脑，或者一部手机。

针对云操作系统，上述五项主要功能的内容应该是：(1)资源接入与抽象，是指将各类服务器、存储、网络设备等硬件资源，通过虚拟化的或者可软件定义的方式，接入到云计算系统中，并将其抽象为云操作系统可以识别的计算、存

储、网络等资源池，以此作为云操作系统对各类硬件资源实施管理的基础；(2)资源分配与调度，是指利用云操作系统的资源管理能力，将前述的不同资源，按照不同的云租户对于资源类型与数量的不同需求，将资源分配给各个租户，以及不同租户的不同应用；(3)应用生命周期管理，是指协助租户实现各类云应用在云操作系统上的安装、启动、停止、卸载等管理操作；(4)系统管理维护，是指协助系统管理员实现对于云计算系统的各类管理与运维操作；(5)人机交互支持，指提供必要的人机界面，支持系统管理员和普通租户对系统实施各类操作。

由上述介绍可以看出，虽然云操作系统比我们日常接触的操作系统复杂很多，但其最为关键的五项主要功能，其实是可以一一对应的。通过这种对应，我们可以更为直观地理解云操作系统这个概念。而OpenStack，则是实现云操作系统的关键组件，或者说，是构建一个完整的云操作系统的框架。

3. 云操作系统框架，不等于云操作系统

要构建一个完整的云操作系统，需要对大量软件组件进行有机整合，让它们协同工作，共同提供系统管理员和租户所需的功能与服务。而OpenStack本身，尚且不能独立具备一个完整云操作系统所需的全部能力。举例而言：在上面提到的云操作系统的五项主要功能中，OpenStack不能独立实现资源接入与抽象，而需要和底层的虚拟化软件、软件定义存储、软件定义网络等软件相配合；OpenStack不能独立提供完善的应用生命周期管理能力，而需要在上层集成各类管理软件平台；OpenStack自身不具备完整的系统管理维护能力，在投入生产实用时，还需要集成各类管理软件与维护工具；OpenStack自身提供的人机界面，其功能也还不够丰富强大，等等。

由此不难看出，想在OpenStack基础上构建一个完整的云操作系统，需要将OpenStack与其他一些软件组件进行集成，以实现OpenStack自身并不提供的能力。因此，OpenStack自身的准确定位，是一个云操作系统框架。基于这个框架，可以集成不同的各类组件，实现满足不同场景需要的云操作系统，并在此基础上，最终构建完整的云计算系统。

4. 开源

开源，是OpenStack的一个重要属性。应该说，不理解开源，就不能真正理解OpenStack的发展历程与未来趋势。

与简单地在网络上公开源代码不同，OpenStack社区遵循的，是一种更为深入、更为彻底的开源理念。在OpenStack社区中，对于每一个组件，每一个特性，乃至每一行代码，其需求提出、场景分析、方案设计、代码提交、测试执行、代码合入的整个流程，都总体遵循开放原则，对公众可见，并且在最大程度上保证了社区贡献者的监督与参与。

正是这种监督与参与的机制，保证了OpenStack社区总体上处于一种开放与均衡的状态，避免了少数人或者少数公司、组织的绝对控制，由此保障了社区生态的健康与繁荣。

同时，OpenStack遵循了对商业最为友好的Apache 2.0许可，也保障了企业参与社区的商业利益，从而推动了OpenStack的产品落地与商业成功。

通过以上介绍，可以看出，OpenStack是一个以开源方式开发与发布的，用于构建不同场景下的云操作系统的框架性软件。深入理解这个本质，对于深入学习和掌握OpenStack，有着非常关键的意义。

二、OpenStack与云计算系统的关系

基于前面的介绍，不难看出，OpenStack与云计算系统之间，既紧密联系，又相互区别。

OpenStack是构建云操作系统的框架。使用云操作系统，集成并管理各类硬件设备，并承载各类上层应用与服务，才能最终形成一个完整的云计算系统。由此可见，OpenStack是云计算系统的核心软件组件，是构建云计算系统的基础框架，但OpenStack和云计算系统并不能直接等同。

三、OpenStack与计算虚拟化的关系

计算虚拟化，是很多读者非常熟悉的概念。其

对应的软件实现，就是平常所说的Hypervisor， 如开源的KVM、Xen，以及VMware的vSphere、 华为的FusionCompute、微软的Hyper-V等。OpenStack与计算虚拟化之间的关系，是目前仍然被频繁混淆的一个问题。理解这二者之间的联系与区别，也是理解OpenStack的关键之一。

OpenStack是一个云操作系统的框架。为构建完整的云操作系统，特别是，为实现资源接入与抽象的功能，OpenStack需要与虚拟化软件实施集成，从而实现对服务器的计算资源的池化。应该指出的是，在资源池化的过程中，物理资源虚拟化的功能，仍然由虚拟化软件完成。举例而言，在使用KVM作为OpenStack的虚拟化软件时，仍然由KVM完成将一台物理服务器虚拟为多台虚拟机的功能，而OpenStack负责记录与维护资源池的状态。例如，系统中一共有多少台服务器，每台服务器的资源共有多少，其中已经向用户分配了多少，还有多少资源空闲。在此基础上，OpenStack负责根据用户的要求，向KVM下发各类控制命令，执行相应的虚拟机生命周期管理操作，如虚拟机的创建、删除、启动、关机等。

由此可见，两相对比，OpenStack更像是系统的控制中枢，是云操作系统的“大脑”；计算虚拟化软件则更像是系统的执行机构，是云操作系统的“肢体”。二者分工合作，共同完成对云计算系统中的计算资源池的管理，但绝不能认为OpenStack等同于计算虚拟化软件。

3.1.2 OpenStack的设计与开发

一、OpenStack的设计思想

OpenStack之所以能够取得快速的发展，除了有云计算技术和产业快速发展的大背景之外，其自身设计思想的独到之处，也起到了有力的促进作用。OpenStack的设计思想，在总体上可以被概括为“开放、灵活、可扩展”。本节将对此展开扼要分析。

1. 开放

OpenStack的开放，根源于其开源模式本身。

前已述及，OpenStack的开源，不仅体现在简单的源代码开放，更体现在其设计、开发、测试、发布的全流程中。这种开源模式，总体上可以保证OpenStack不被个别人或个别企业所控制，在技术上不会走向封闭架构、封闭体系，从而始终呈现出良好的开放性。无论是北向的API标准开放，还是南向的各类软件、硬件自由接入，都是OpenStack开放性的充分体现。

与此同时，OpenStack也秉持了开源社区中“不重复发明轮子”的一贯理念，在设计中持续引入并充分重用各相关技术领域中的优秀开源软件，从而提升了设计与开发效率，并为软件质量提供了基本保证。

2. 灵活

OpenStack的灵活，首先体现在其大量使用插件化、可配置的方式进行设计。最为突出的体现，就在于OpenStack采用插件化的方式实现不同类型计算、存储、网络资源的接入，由此实现OpenStack对于不同类型资源的灵活接入与管理，用一套架构实现了对于不同厂商、不同类型设备的资源池化，例如，在计算领域，可以以插件化的形式接入KVM、Xen、vCenter、FusionCompute等不同的Hypervisor；在存储领域，可以以插件化的形式实现对不同厂商的存储设备，以及Ceph、FusionStorage、vSAN等不同的软件定义存储的管理；在网络领域，可以实现对不同的网络硬件设备，OVS、Liunx-bridge、HAProxy等开源网络组件，以及多种SDN控制器的接入。并且，这些接入都是通过可配置的方式加以选择。当在不同的资源之间进行选择时，OpenStack自身并不需要重新打包发布，只需通过配置项选择不同的接入插件即可，非常方便。

在此基础上，OpenStack的灵活还体现在不依赖于任何特定的商用软硬件。换言之，任何商用软硬件产品在OpenStack中一定是可选、可替换的，从而严格保证用户可以使用完全开源、开放的方案来构建基于OpenStack的云计算系统，而完全不必担心被锁定在某些特定厂商的产品之上。

3. 可扩展

OpenStack的架构高度可扩展。具体而言，其

扩展性体现在功能和系统规模两个方面。

从功能视角看，OpenStack由多个相互解耦的项目组成。不同的项目分别完成云计算系统中的不同功能，如身份认证与授权服务、计算服务、块存储服务、网络服务、镜像服务、对象存储服务等。对于一个特定场景下的云计算系统，系统设计人员可以根据实际需要决定使用OpenStack中的若干个项目，也可以在系统上线后，根据需求继续引入新的OpenStack项目。OpenStack的一些项目自身也具有功能可扩展性。系统设计人员可以在这些项目中引入新的功能模块，在不影响项目既有功能使用的前提下，对其功能进行扩展。

从系统规模视角看，OpenStack总体上遵循了无中心、无状态的架构设计思想。其主要项目，均可实现规模水平扩展，以应对不同规模的云计算系统建设需求。在系统建成后，可根据应用负载规模的实际增长，通过增加系统管理节点和资源节点的方式，逐渐扩展系统规模。这种架构可以有效避免高额的初始建设投资，也降低了系统初始规划的难度，为云计算系统的建设者和运营者提供了充分的扩展空间。

二、OpenStack的开发模式

前已述及，OpenStack采用了完全开放的开发模式，由数以千计的社区贡献者通过互联网协作的方式，共同完成各个项目的设计、开发、测试和发布。

具体而言，OpenStack社区以每6个月为一个版本开发与发布周期，分别于每年4月和10月发布新的OpenStack版本。每个新版本发布之后约三周，社区会举行一次OpenStack设计峰会，以便开发者集中讨论新版本应优先引入的特性，或应集中解决的问题。其后，社区将进入为期约5个月的开发和测试阶段，直至新的版本发布。截至本书撰写之时，最新的OpenStack发行版是2016年4月初发布的Mitaka版本，也是OpenStack的第13个发行版。

OpenStack各个项目统一遵循Apache 2.0开源许可证，对于商业应用非常友好。OpenStack各项目以Python为首选开发语言，各个项目的核心代码均使用Python语言实现。

三、OpenStack社区发展现状

自2010年成立以来，OpenStack社区始终保持了高速发展的态势，目前已经成为了仅次于Linux的世界第二大开源软件社区。而这一切仅仅用了不到六年的时间，不得不让人惊叹开放的云计算技术所具有的强大魅力。

在过去的五年多时间里，OpenStack社区的各项主要贡献指标，都呈现出快速上升的总体趋势。这种趋势，从OpenStack峰会参会人数的爆炸式增长就可以看出。2010年OpenStack首届峰会举办时，仅有75人参与。而2016年4月举办峰会时，参会人数高达7500人以上。短短六年不到的时间，人数激增100倍，由此不难看出OpenStack社区巨大的影响力与凝聚力。

3.1.3 OpenStack架构与组成

在2010年OpenStack社区首次发布其第一个发行版——Austin时，OpenStack仅包含两个项目Nova和Swift，仅能实现非常简单和基础的功能。时至今日，OpenStack已经日渐成熟和强大，其组成项目也已经大大增多，仅包含在Mitaka版本release notes中的服务项目就多达29个。各个项目各司其责，分工合作，共同形成了一个架构灵活、功能丰富、扩展性强的云操作系统框架。

为便于读者快速了解OpenStack的概貌，但又不致淹没在众多的信息当中，本节将优先选择OpenStack中最为关键和有代表性的部分项目，进行扼要介绍，以便帮助读者更为直观地了解OpenStack。

1. Keystone：身份认证与授权服务

将计算、存储、网络等各种资源，以及基于上述资源构建的各类IaaS、PaaS、SaaS层服务，在不同的用户间共享，让众多用户安全地访问和使用同一个云计算系统，是一个云操作系统的基本能力。而实现这个能力的基础，就是一个安全

可靠的身份认证与授权服务。而Keystone就是OpenStack的身份认证与授权服务项目。

Keystone负责对用户进行身份认证，并向被认定为合法的用户发放令牌(token)。用户持Keystone发放的令牌访问OpenStack的其他项目，以使用其提供的服务。而各个组件中内嵌的令牌校验和权限控制机制，将与Keystone配合实现对用户身份的识别和权限级别的控制，保证只有恰当的用户能够对恰当的资源实施恰当的操作，以此保证对不同用户资源的隔离与保护。

2. Nova：计算服务

向用户按需提供不同规格的虚拟机，是任何一个云操作系统最为基础的功能。而Nova就是OpenStack中负责提供此类计算服务的项目。

Nova的核心功能，是将大量部署了计算虚拟化软件(即Hypervisor)的物理服务器统一纳入管理之下，组成一个具有完整资源视图的逻辑资源池。在此基础上，Nova通过接收不同用户发起的请求，对资源池中的资源进行生命周期管理操作。其中最为核心的，就是虚拟机的创建、删除、启动、停止等操作。通过执行客户发起的虚拟机创建操作，Nova将逻辑资源池中的CPU、内存、本地存储、IO设备等资源，组装成不同规格的虚拟机，再安装上不同类型的操作系统，最终提供给用户进行使用，由此满足用户对于计算资源的需求。

除了虚拟机资源管理服务能力之外，Nova还通过与Ironic项目配合，共同为用户提供裸机资源管理服务能力。具体而言，Nova可以接收用户发起的裸机资源申请，然后调用Ironic项目的对应功能，实现对裸机的自动化选择、分配与操作系统安装部署，从而使得用户可以获得与虚拟机资源使用体验相当的物理机资源使用体验。

3. Ironice：裸机管理

Ironic通过与Nova相配合，共同为用户提供裸机服务能力。

在实际工作时，Ironic直接负责对物理服务器的管理操作。一方面，在物理服务器被纳入到资源池之中时，Ironic负责记录物理服务器的硬件规格信息，并向Nova上报；另一方面，在用户发起裸机管理操作时，Ironic负责根据Nova的指令，对相应的物理服务器执行具体的管理操作动作。例如，当用户发起一个创建裸机操作时，Ironic需要根据Nova调度的结果，对选定的物理服务器执行硬件初始化配置、操作系统安装等一系列具体操作，以完成裸机创建动作。

4. Glance：镜像服务

通常而言，在虚拟机被创建之后，都需要为其安装一个操作系统，以便用户使用。为此，云计算系统中往往需要预置若干不同种类、不同版本的操作系统镜像，以便用户选用。此外，在一些应用场景下，为进一步方便用户，镜像中还需要预装一些常用的应用软件，这将进一步增加镜像的种类与数量。为此，云操作系统必须具备镜像管理服务能力。Glance就是OpenStack中的镜像服务项目。

Glance主要负责对系统中提供的各类镜像的元数据进行管理，并提供镜像的创建、删除、查询、上传、下载等能力。但在正常的生产环境下，Glance本身并不直接负责镜像文件的存储，而是仅负责保管镜像文件的元数据，本质上是一个管理前端。Glance需要与真正的对象存储后端对接，才能共同提供完整的镜像管理与存储服务能力。

5. Swift：对象存储服务

对象存储服务，是云计算领域中一种常见的数据存储服务，通常用于存储单文件数据量较大、访问不甚频繁、对数据访问延迟要求不高、对数据存储成本较为敏感的场景。Swift就是OpenStack中用于提供对象存储服务的项目。

与OpenStack中大部分只实现控制功能、并不直接承载用户业务的项目不同，Swift本身实现了完整的对象存储系统功能，甚至可以独立于OpenStack，被单独作为一个对象存储系统加以应用。

此外，在OpenStack系统中，Swift也可以被用做Glance项目的后端存储，负责存储镜像文件。

6. Cinder：块存储服务

在典型的、基于KVM虚拟化技术的OpenStack部署方案下，Nova创建的虚拟机默认使用各个计算节点的本地文件系统作为数据存储。这种数据存储的生命周期与虚拟机本身的生命周期相同，即当虚拟机被删除时，数据存储也随之被删除。如果用户希望获得生命周期独立于虚拟机自身的、能够持久存在的块存储介质，则需要使用Cinder提供的块存储服务，也称为卷服务。

Cinder负责将不同的后端存储设备或软件定义存储集群提供的存储能力，统一抽象为块存储资源池，然后根据不同需求划分为大小各异的卷，分配给用户使用。

用户在使用Cinder提供的卷时，需要使用Nova提供的能力，将卷挂载在指定的虚拟机上。此时，用户可以在虚拟机操作系统内看到该卷对应的块设备，并加以访问。

7. Neutron：网络服务

网络服务，是任意云操作系统IaaS层能力的关键组成部分。只有基于稳定、易用、高性能的云上虚拟网络，用户才能将云计算系统提供的各类资源和服务能力连接成真正满足需求的应用系统，以解决自身的实际业务需求。

Neutron是OpenStack中的网络服务项目。Neutron及其自身孵化出来的一系列子项目，共同为用户提供了从Layer 2到Layer 7上不同层次的多种网络服务功能，包括Layer 2组网、Layer 3组网、内网DHCP管理、Internet浮动IP管理、内外网防火墙、负载均衡、VPN等。整体而言，Neutron的Layer 2、Layer 3服务能力已经较为成熟。时至今日，Neutron已经取代了早期的nova-network，成为了OpenStack中Layer 2、Layer3的主流虚拟网络服务实现方式。与之对应，Neutron的Layer 4至Layer 7服务能力仍在迅速发展中，目前已具备初步应用能力。

需要说明的是，OpenStack的DNS即服务能力，并未包含在Neutron项目的功能范围当中，而是由另一个单独的项目Designate负责实现。

8. Heat：资源编配服务

云计算的核心价值之一，即在于IT资源与服务管理和使用的自动化。换言之，在引入云计算技术之后，大量在传统IT领域中需要依靠管理人员或用户通过手工操作实现的复杂管理操作，应当可以通过调用云操作系统提供的API，以程序化的方式自动完成，从而显著提高IT系统管理的效率。

在上述提及的IT领域复杂管理操作中，用户业务应用系统的生命周期管理操作，即应用系统的安装、配置、扩容、撤除等，可谓是具有代表性的一类。这类操作的复杂与耗时耗力，与当前不断凸现的业务快速上线、弹性部署诉求，已经表现出明显的不适应性。

Heat项目的出现，就是为了在OpenStack中提供自动化的应用系统生命周期管理能力。具体而言，Heat能够解析用户提交的，描述应用系统对资源类型、数量、连接关系要求的定义模板，并根据模板要求，调用Nova、Cinder、Neutron等项目提供的API，自动实现应用系统的部署工作。这一过程高度自动化，高度程序化。同样的模板，可以在相同或不同的基于OpenStack的云计算系统上重复使用，从而大大提升了应用系统的部署效率。

在此基础上，Heat还可以与OpenStack Ceilometer项目的Aodh子项目相配合，共同实现对于应用系统的自动伸缩能力。这更进一步简化了部分采用无状态、可水平扩展架构的应用系统的管理，具有典型的云计算服务特征。

9. Ceilometer：监控与计量

在云计算系统中，各类资源均以服务化的形式向用户提供，用户也需要按照所使用资源的类型和数量缴费。这种基本业务形态，就要求云操作系统必须能够提供资源用量的监控与计量能力。这正是OpenStack引入Ceilometer项目的根本动机。

Ceilometer项目的核心功能，是以轮询的方式，收集不同用户所使用的资源类型与数量信息，以此作为计费的依据。

在此基础上，Ceilometer可以利用收集的信息，通过Aodh子项目发送告警信号，触发Heat项目执行弹性伸缩功能。

需要说明的是，Ceilometer项目自身并不提供计费能力。系统设计者需要将其与适当的计费模块相对接，才能实现完整的用户计费功能。目前，OpenStack社区已经创建了CloudKitty项目，作为OpenStack社区原生的计费组件。但该项目当前尚处于较为初期的阶段，难以直接商用。

10. Horizon：图形界面

Horizon项目是OpenStack社区提供的图形化人机界面。经过社区长期的开发完善，Horizon界面简洁美观，功能丰富易用，可以满足云计算系统管理员和普通用户的基本需求，适于作为基于OpenStack的云计算系统的基本管理界面使用。

此外，Horizon的架构高度插件化，灵活而易于扩展，也便于有定制化需求的系统设计人员针对具体场景进行增量开发。

11. Sahara：数据处理服务

应当说，大数据和云计算，是两个天然紧密联系的技术领域。云计算技术的出现，为大数据处理提供了廉价、易用、易扩展的计算支撑平台。而大数据处理业务，由于其可并行、高弹性、自身可容错的特征，也是云计算平台上的一种理想业务。

针对这一背景，OpenStack社区推出了Sahara项目，以实现Hadoop、Spark等主流大数据处理集群软件的云化。使用Sahara项目，即便是没有任何大数据处理集群软件安装部署和管理应用经验的用户，也可以以图形化的方式，极其简便地安装部署属于自己的、规模适当的大数据集群，并以简明易懂的方式对自己的数据集进行指定算法的处理，以获取处理结果。Sahara的出现，极大地简化了普通用户使用大数据处理软件的过程，将大数据和云计算两项技术紧密地结合在了一起。

12. Magnum：容器服务

容器是当今无可回避的热门技术话题。容器技术的出现、发展与繁荣，极大地提升了软件的开发与部署效率，也极大地改变了软件生命周期管理的既有模式。围绕着以Docker为代表的容器化软件生命周期管理技术体系，已经逐渐发展出Kubernetes、Mesos、Swarm等容器集群管理系统，以便在服务器集群上实施应用软件生命周期管理和集群资源调度。

在这种情况下，OpenStack社区自然也不会无动于衷。Magnum项目就是OpenStack社区为实现容器集群管理系统的服务化而推出的新项目。使用Magnum，用户可以在基于OpenStack的云计算系统上，实现容器集群管理系统的生命周期管理自动化。具体而言，利用Magnum，用户可以完全通过API调用的方式，实现Kubernetes集群在OpenStack之上的自动化安装部署，并通过Magnum的API对Kubernetes实施管理操作，非常简单便捷。

3.2 容器开源软件：Kubernetes / Mesos / Docker

Docker技术的出现和迅猛发展，已成为云计算产业的新的热点。容器使用范围也由互联网厂商快速向传统企业扩展，大量传统企业开始测试和尝试部署容器云。

相比于企业对容器技术的逐步接受与认同，在如何使用容器上却并不统一，存在多种思路和诉求。容器技术开发者和社区倡导云原生应用场景，这一理念被业界普遍所认可，但在实际使用中发生分化。部分企业基于容器技术，尝试新应用开发和对传统应用进行改造；更多的企业在实际使用中，面临应用改造困难和人员技能变更，认为不可一蹴而就，希望先以轻量级虚拟机的方式使用容器。

容器的下述技术特点，决定了容器所能发挥真正价值的应用场景。

✍ 轻量化：容器相比于虚拟机提供了更小的镜像，更快的部署速度。容器轻量化特性非常适合需要批量快速上线应用，或需要快速规模弹

缩应用，如互联网web应用。

✍ 性能高、资源省：相比虚拟化，接近物理机的性能，系统开销大幅降低，资源利用率高。容器的高性能特性非常适合对计算资源要求较高的应用，如大数据和高性能计算应用。

✍ 跨平台：容器技术实现了OS解耦，应用一次打包，可到处运行。容器的跨平台能力，非常适合作为DevOps的下层封装平台，实现应用的CI/CD流水线；容器应用可跨异构环境在不同云平台、公有云和私有云上部署，也非常适合作为混合云的平台。

✍ 细粒度：容器本身的轻和小，非常匹配细粒度服务对资源的诉求，与微服务化技术的发展相辅相成，可作为分布式微服务应用的最佳载体。

企业关注下一代内部IT架构变革，希望将服务作为IT核心，提升业务敏捷性，大幅降低TCO。容器成为企业应用转型很好的承载平台，针对企业的业务痛点，使用上述一种或多种特点，优化业务场景。

业界当前主要有三种容器集群资源管理调度和应用编排的不同选择。

✍ Mesos生态：核心组件包括Mesos容器集群资源管理调度以及不同的应用管理框架。典型的应用管理框架包括Marathon和Chronos，其中Marathon用来管理长期运行服务，如Web服务；Chronos用来管理批量任务。Mesos生态主要由Mesosphere、Twitter等公司主力推动。

✍ Kubernetes生态：Google公司发起的社区项目，涵盖容器集群资源管理调度，以及不同类型应用的应用管理组件。例如副本可靠性管理，服务发现和负载均衡，灰度升级，配置管理等组件。

✍ Docker生态：Docker公司希望向容器生态系统上层发展，推出了Swarm容器资源管理调度组件，以及Compose应用编排组件。

本书第五章将对容器生态与技术进行详细的阐述。

3.3 大数据开源软件：Hadoop/Spark

3.3.1 Hadoop简介

大数据开启了一次重大的时代转型。就像望远镜让我们感受宇宙，显微镜让我们能够看到微生物一样，大数据正在改变我们的生活和理解世界的方式——《大数据时代》。我们正处于这样一个数据指数爆发的时代，海量数据来自于智能终端、物联网、社交媒体，电子商务等。如何收集、存储、分析海量数据，进而支持科学预测、商业决策，提升医疗健康服务水平，提升能源效率，防范金融欺诈风险，降低犯罪率和提升案件侦破效率？应对上述问题和挑战，开源社区与产业界给出了答案：

Hadoop！

一、Hadoop项目

Hadoop来自Apache社区，是一个可水平扩展、高可用、容错的海量数据分布式处理框架，提供了简单分布式编程模型map-reduce。Hadoop设计的假设是底层硬件不可靠，由Hadoop检测和处理底层硬件失效(见表3-1)。

表3-1 Hadoop核心组件

组件	描述
Hadoop Common	支持其他模块的通用基础库
HDFS	高吞吐、高可靠的分布式文件系统
Hadoop YARN	集群资源管理和任务调度框架
Hadoop MapReduce	基于yarn的并行处理系统

二、Hadoop生态层次

基于Hadoop提供的基础分布式存储及分布式并行处理能力，Apache社区围绕Hadoop衍生出大量开源项目，图3-1展示了当前Hadoop开源生态包含的主要开源项目，表3-2展示了Hadoop开源生态层次。

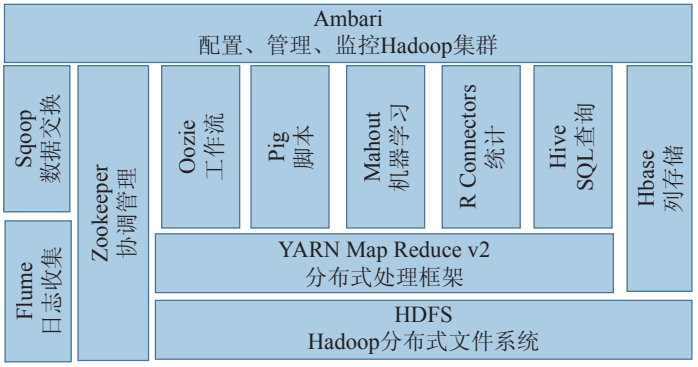


图3-1 Hadoop生态架构

表3-2 Hadoop开源生态层次

层次	描述	组件名称
数据存储层	提供海量数据分布式存储	HDFS，HBASE
数据处理层	提供集群资源管理、调度框架、并行计算框架	Yarn，MapReduce
数据接入访问层	提供海量数据访问高层次接口	Hive，Pig，Mahout，R Connectors
管理层	分布式任务工作流引擎、数据导入导出工具，协调服务，集群部署监控与集群管理	Oozie，Flume，Sqoop，Zookeeper，Ambari

三、Hadoop开源生态发展历程

Hadoop开源生态系统在应对大数据收集、存储、分析等难题的挑战中，逐渐成长壮大。

表3-3展示了大数据问题与对应的Hadoop开源组件。

表3-3 大数据问题与对应的Hadoop开源组件

大数据问题	组件名称
海量数据的存储问题	HDFS
海量数据并行处理问题	MapReduce
非结构化数据收集、导入、导出问题	Chukwa，Flume
结构化数据收集、导入、导出问题	Sqoop
支持OLTP，海量数据表随机读写快速查询问题	Hbase
MapReduce任务的工作流管理问题	Oozie
MapReduce更高层次的访问接口问题	Pig，Hive
数据挖掘提供更高层次接口问题	Mahout

Hadoop 开源生态系统的快速成长得益于产业界长久以来在大数据方面的实践积累和持续贡献。

表3-4展示了Hadoop发展的关键事件。

表3-4 Hadoop发展的关键事件

年	月	事件	开源组件
2003	10月	Google GFS论文发表	HDFS
2004	12月	Google MapReduce论文发表	MapReduce
2006	4月	Hadoop 0.1.0版本发布	
2006	11月	Google发表Bigtable论文	Hbase
2006	11月	Google发表Chubby论文	Zookeeper
2007	6月	HBase(Hadoop Database)发布	
2007	10月	Yahoo实验室将Pig贡献给Apache社区	Pig
2008	8月	Facebook将Hive贡献给Apache社区	Hive
2012	1月	Yarn取代MapReduce	Yarn
2012	11月	Apache Hadoop 1.0 发布	
2015	6月	Apache Hadoop 2.7 发布	

四、Hadoop最新进展

1. HDFS

HDFS发展相对比较迅速，目前已得到广泛应用。但随着业务的发展，用户对于HDFS产生了许多新的需求。目前开源社区已根据这些需求提供许多新特性，如SnapShot、Inotify等。

(1) SnapShot

Hadoop从2.1.0版开始提供了HDFS SnapShot的功能。一个SnapShot可以是某个被设置为snapshottable的目录在某一时刻的镜像。

一个snapshottable目录可以同时容纳65536个快照。snapshottable目录没有个数上限，管理员可以设置任意个snapshottable。如果一个snapshottable中存在快照，那么这个目录在删除所有快照之前，不能删除或改名。SnapShot适应于如下场景。

- 错误操作恢复：如果对于某个目录设置了SnapShot，当用户意外地删除了一个文件，就可以使用该SnapShot进行文件的恢复。
- 备份：管理员可以根据业务需求周期性地对文件目录进行备份。

(2) Inotify

类似于Linux Inotify。NameNode对目录的每次操作，都会产生一个新的操作编号。Client通过RPC机制向NameNode发送某个操作编号，NameNode读取Edits将文件系统的大于该编号的所有Event操作返回给客户端(见图3-2)。

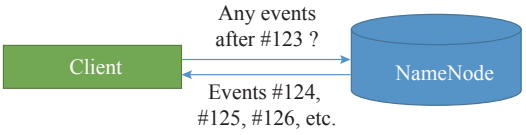


图3-2　Inotify逻辑

通过HDFS Inotify机制，用户能够及时地得知某个目录发生了什么操作。该机制可以广泛用于感知目录操作，获取频繁修改文件，以及由于安全原因需要对特定目录或文件进行操作监控等场景，避免扫描整个目录，从而为用户提供更好的服务。

2. YARN

(1) 多维资源调度

YARN目前支持Memory、CPU的两维资源，利用DRF(Dominant Resource Fairness)算法对Memory、CPU进行调度。但是随着大数据应用场景的极大丰富，用户需要申请、调度和控制集群中的更多的资源类型，如硬盘、网络等。开源社区已经考虑支持Disk(YARN-2139)、Network(YARN-2140)和HDFS Bandwidth(YARN-2681)。

1) Resource Define

对于新资源类型的支持，首先应该分析如何定义资源。例如对于Network资源来说，应用可能关注于网络bandwidth和每秒ops；而对于Disk资源，应该关注于I/O bandwidth。

2) Resource isolation

除了如何定义资源外，另一个更关键的部分在于如何隔离资源以保证应用使用，而不受其他应用影响。Linux的CGroup已经为我们提供了资源隔离的技术方案，利用CGroup可以实现Disk资源vdisk的定义，实现Network资源net_cls的定义。

3) Resource Model & Extend

YARN重新定义资源类型Resource Type Information替换原来仅支持Memory和CPU的资源定义，在ResourceManager和NodeManager新增了资源类型的配置文件，定义包括资源名、单位、类型等信息；同时也新增了资源规格模板配置文件的定义，用户可以根据需要定义不同的资源规格。当用户提交应用时可以直接选取模板中的适合的规格。这样也可以避免当引入新的资源类型时，需要修改以前提交应用的代码以兼容。现在只需要重新定义资源规格模板即可。

(2) 基于已分配资源利用率的调度

YARN分配资源基于整个集群资源的可用资源，可用资源来自于所有节点总容量减去已分配资源量。对于一个具体Container来说，其分配的资源是其真实使用量的上限。而实际中，往往大部分时间Container的资源使用量小于这个分配值，

这就导致了整个集群的实际资源利用率低下。

因此YARN可以通过获取Container实际资源率信息，对未真正使用的资源进行调度分配，以提高整个集群的资源使用率。

1) OPPORTUNISTIC Container

引入新OPPORTUNISTIC类型的Container后，这种Container可以利用节点上已分配但未真正使用的资源。原有Container类型定义为GUARANTEED类型。相对于GUARANTEED类型Container，OPPORTUNISTIC类型的Container优先级更低。

2) Monitor Utilization

监控单个Container及节点上所有Container的实际资源使用量，并通过节点心跳上报给ResourceManager。

3) Preempt & Promote

为节点设置资源利用率的阈值，当资源利用率超过该阈值后，抢占之前分配出去的OPPORTUNISTIC类型的Container，以保证GUARANTEED类型的Container资源使用。当GUARANTEED类型的Container完成任务释放资源后，该节点上的OPPORTUNISTIC类型的Container可以升级为GUARANTEED类型。

3. HBASE

(1) Hbase 多租户

1) Region Server Groups

Hbase集群中，多个Region Server组成一组，通过将特定租户的表Region分配到租户所属的Region Server Groups，为租户提供Region Server级别的资源隔离能力。

同时，Region Server Group之间可能在硬件、配置、性能上存在差异，基于此可以为不同的工作负载特征表，分配不同的Region Server，实现不同工作负载彼此资源隔离，性能上互不干扰。

LoadBanlancer负责表Region的分配，在分配Region的过程中，LoadBanlancer通过候选的Region Server选择一个Region Server。Region Server组实现方案是通过是Region Server组过滤器，实现将特定租户表的Region分配到特定的Region Server组中(见图3-3、图3-4)。

2) Namespace

Namespace是Hbase的一等公民，提供了Hbase多租户抽象，用于创建管理表和Region Server组，通过Quota限制Namespace下表的数量和Region的数量。尽管通过Region Server组可以实现RegionServer级别的隔离，集群中仍有一部分资源是共用的，比如Hmaster、meta表等，可以通过ACL实现管理域资源的隔离访问(见图3-5)。

4. Hbase多数据中心数据复制方案

单主(Single Master)：只有一个主数据中心写入，并向其他数据中心同步数据，副本数据中心提供只读服务(见图3-6)。

- 保证数据复制不影响写入性能，通常采用异步复制防止方式，在主数据中心数

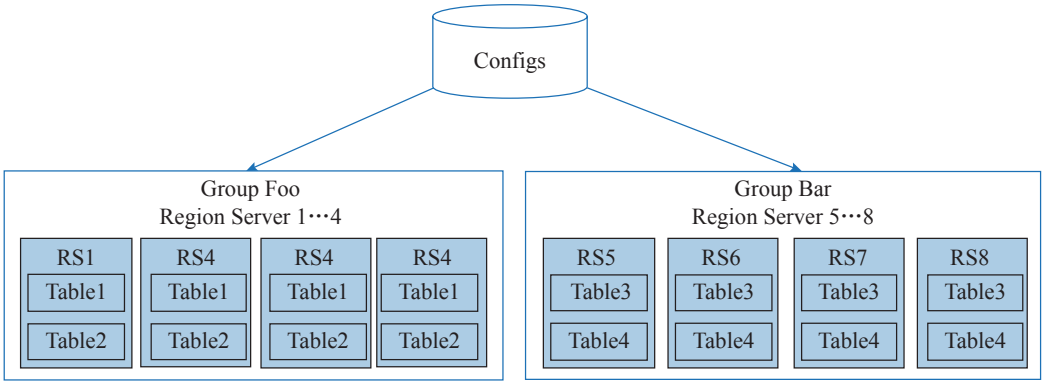


图3-3　Region Server Group示意图

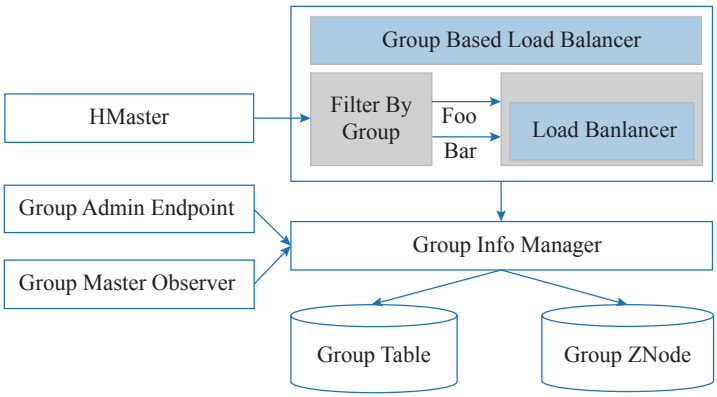


图3-4　Region Server Group实现

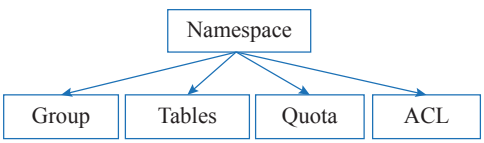


图3-5　Hbase Namespace

据完全丢失后，会存在同步窗口内少量数据丢失。

- 副本数据中心故障，副本数据中心数据不一致。
- 由于采用跨数据中心读取，数据一致性需要额外保障
- 写依然被限制在一个数据中心。

主主(MM，Master-Master)：所有数据中心都支持读写，所有数据都一直支持事务，但很难做到。

- 主主需要解决多写合并问题，首先需要解决写请求的顺序问题，由于缺乏全局一致的时钟，可以使用的是本地时间戳、分布式一致性。

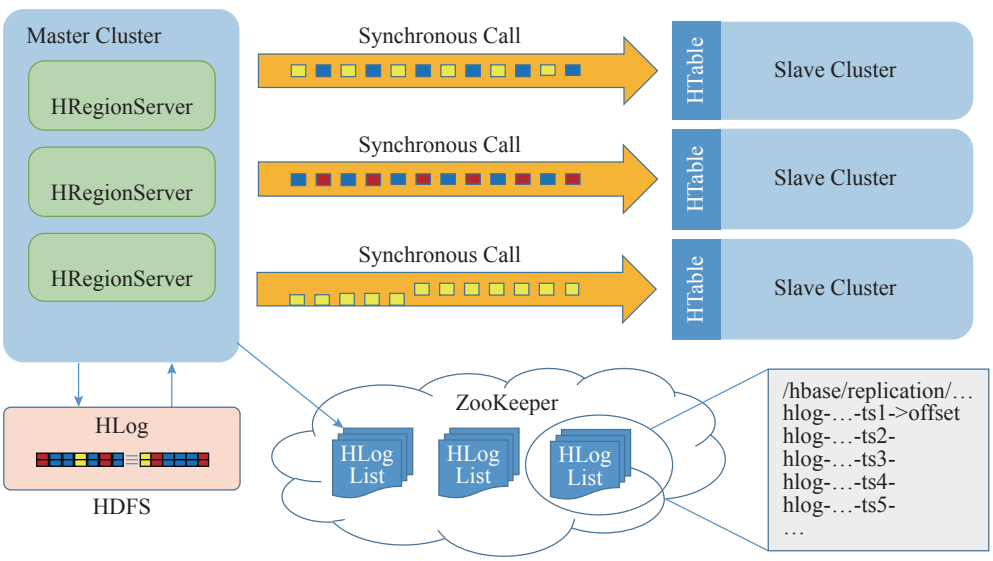


图3-6　Hbase 主备异步复制方案

- 一般选在两个地理足够接近的数据中心，时延严格保障，通过两阶段提交保证事务。两地三中心方案是此方案的衍生方案。
- 多于两数据中心，一般认为是十分困难的。考虑到传输，数据中心间的交互，时延的代价将非常大。

表3-5为多数据中心复制方案的比较。

表3-5 多数据中心复制方案比较

	M/S	MM	2PC	Paxos
一致性	最终一致	最终一致	强一致性	强一致性
事务处理	全部	本地	全部	全部
时延	低	低	高	高
吞吐量	高	高	低	中等
数据丢失	一些	一些	无	无
故障切换	只读	读/写	读/写	读/写

五、Hadoop开源生态展望

Hadoop生态已经日臻完善，Hadoop平台也是当下最受欢迎的大数据平台之一，非常适合网页、日志等非结构化或半结构化类文本数据的分析处理。那么下面Hadoop的可能发展方向有哪些？我们大胆的想象一下。

批流合一的计算引擎

Hadoop架构是面向批处理的，在需要实时处理秒级甚至毫秒级响应场景下，需要流处理平台，例如Storm。统一平台进行批处理和流处理是当前和未来研究热点。

相关开源项目包括Flink、Apex、Nifi、DataFlow等。

1. SQL On Hadoop

SQL是通用的数据操作语言，很多开源工具开发目标是能够在Hadoop上使用SQL。这些工具有些是对MapReduce的封装，有些是在HDFS上实现完整的数据仓库，当然也存在介于两者之间的方案，查询速度与查询性能存在差异。

相关开源项目包括Hive、Impala、Shark、Drill、HAWQ、Calcite、Phoenix等。

2. Spark

很多人认为Hadoop的未来是Spark。下一小节我们会对Spark开源生态进行介绍。粗略地看，Spark与Hadoop的典型差异在于通过内存计算大幅提升数据处理，尤其是迭代运算的处理速度。当然Spark不仅仅是MapReduce的替代品，Spark包含了内存计算引擎、内存文件系统、流处理平台Spark Streaming、数据挖掘库Mlab。Spark会取代Hadoop吗？Hadoop社区的100多名commiter正在谋划未来，让我们拭目以待。

3. YARN

随着数据规模的不停快速增长，处理数据的集群规模也在不断变大，且由于应用类型的不断丰富，集群中将会存在各种不同类型的物理节点，如高内存、GPU等。而对于不同类型的节点，不同的应用也期望不同的资源分配策略。目前YARN通过label对集群进行分组分区，同时通过label匹配应用指定运行的机器。而正是由于label兼具了分区和匹配的两重身份，导致目前YARN很难保证异构资源采用不同的调度策略(因为调度策略是基于租户，而整个集群只有一套租户策略)。

因此如果将label的双重身份进行拆解，引入资源池的概念独立进行集群资源的分组分区，也许可以很好地解决当前问题。不同的资源池具有独立不同的一套租户策略，而原有的label只负责定位具体应用到相应类型的节点。这样将面对越来越巨大的集群规模，资源池可以使管理者可以灵活对集群进行分组，针对不同的场景和资源需求，设定不同的策略，以保证资源的最优化最有效配置(见图3-7)。

4. 数据挖掘理论与技术

从海量大数据中挖掘隐含的信息或者知识是大数据分析的终极目标。当前社区已经实现了基于大数据的机器学习平台或库(Mahout)。然而这部分工作才刚刚开始，数据挖掘、机器学习、深度机器学习技术与大数据结合解决特定场景问题目前尚不成熟，仍有很大的提升空间。

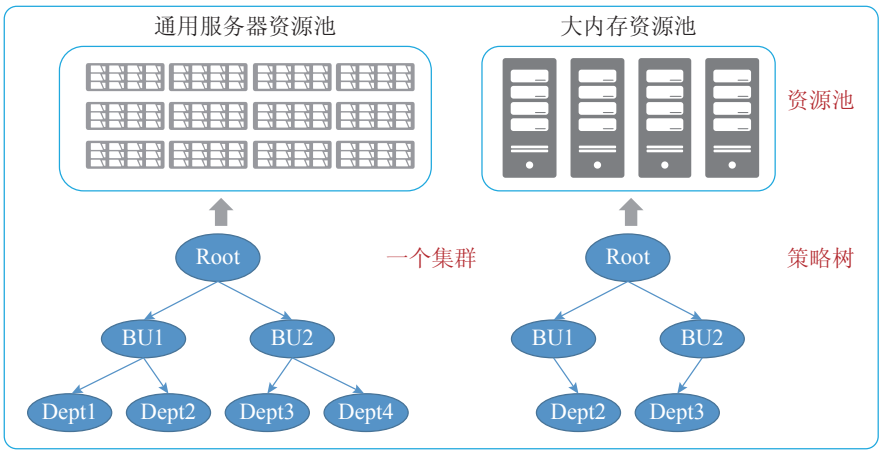


图3-7　Yarn工作原理

3.3.2　Apache Spark简介

一、Spark架构

Apache Spark是一种用Scala语言编写的通用并行计算框架，最早由UC Berkeley AMP Lab在2009年开发，于2010年将其开源，随后捐赠给Apache软件基金会，在2014年2月成为了Apache的顶级项目。

Apache Spark完全兼容Apache Hadoop，Apache Spark除了支持Map和Reduce操作之外，还自持了SQL查询、流数据处理、机器学习和图计算。开发者可以在应用中单独使用Apache Spark的某一特性，或者将这些特性结合起来一起使用。Apache Spark总体架构，如图3-8所示。

二、Apache Spark核心组件

1. Spark Core

Spark Core是Spark整个项目的基础，作为其他组件的计算引擎，提供了分布式计算任务调度，分发和存储管理能力，对外通过RDD (Resilient Distributed Dataset，弹性分布式数据集)的概念暴露API接口。

RDD是Spark中的重要概念，可以理解为一个跨机器的分布式缓存，RDD一旦被生成，存储在其中的数据就不能被改变，直到生成一个新的RDD。

RDD支持两种类型的操作具体如下。

- Transformation：Transformation包括常用的map、filter、flatMap、group ByKey、reduceByKey、aggregateByKey 等算子，其操作结果产生新的RDD。
- Action：Action会触发RDD的计算，计算结果将返回到Spark的驱动程序或者将结果持久化到特定的存储中。

由于RDD充分利用内存来存储数据，整个计算过程中避免了Hadoop MapReduce在执行工作任务时必须通过磁盘来缓存中间结果，大幅度地提高计算性能，提高了Spark的速度。Java的内存管理往往给Spark带来问题，于是Project Tungsten计

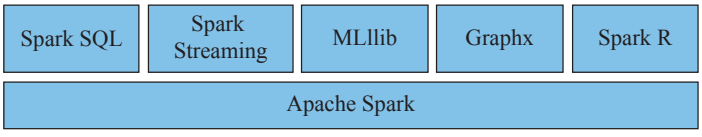


图3-8　Apache Spark 架构

划避开JVM的内存和垃圾收集子系统，以此提高内存效率。

Apache Spark支持多种运行模式，可以在单机和集群中运行，同时还支持在Hadoop YARN集群和Mesos集群中运行。

Spark主要是用Scala编写的，所以Spark的主要API长期以来也支持Scala。不过另外三种使用广泛得多的语言同样得到支持：Java(Spark也依赖它)、Python和R。

2. Spark SQL

Spark SQL是构建在Spark Core上面的一个模块，主要用来处理结构化数据，用户可以通过SQL、DataFrames API以及Datasets API和Spark SQL交互。

另外Spark SQL可以通过JDBC API将Spark数据集暴露出去，而且还可以用传统的BI和可视化工具在Spark数据上执行类似SQL的查询。用户还可以用Spark SQL对不同格式的数据(如JSON、Parquet以及数据库等)执行ETL，将其转化，然后暴露给特定的查询。

Spark SQL其实不支持更新数据，因为那与Spark的整个意义相悖。可以将查询操作生成的数据写回成新的Spark数据源(比如新的Parquet表)，但是UPDATE查询并不得到支持。

3. Spark Streaming

Spark Streaming是对Spark Core的扩充，是一种可扩展的、高吞吐、容错的流处理计算框架，目前支持的数据源包括 Kafka、Flume、Twitter、ZeroMQ、Kinesis、TCP sockets等，数据处理完成后，可以被存放到文件系统、数据库等。

Spark Streaming并不会像Storm那样一次一个地处理数据流，而是在处理前按时间间隔预先将其切分为一段一段的批处理作业，即先汇聚批量数据，然后提交到Spark Core去运行，所以在数据延迟，其方面相对Storm会大一些(见图3-9、图3-10)。

4. MLlib Machine Learning Library

MLlib是一个可扩展的Spark机器学习库，由通用的学习算法和工具组成，包括分类、线性回归、聚类、协同过滤、梯度下降以及底层优化原语(见图3-11)。

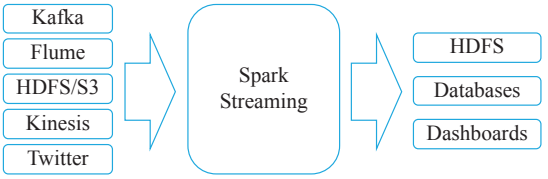


图3-9 Apache Spark Streaming

图3-10 Apache Spark Streaming原理图

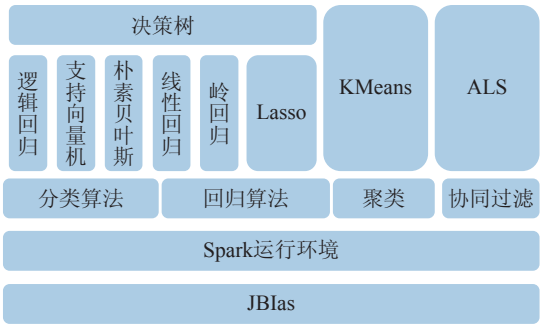


图3-11 Apache Spark MLlib支持的算法

5. GraphX

GraphX是用于图计算和并行图计算的Spark API。通过引入弹性分布式属性图(Resilient Distributed Property Graph)，顶点和边都带有属性的有向多重图，扩展了Spark RDD。为了支持图计算，GraphX暴露了一个基础操作符集合(如subgraph、joinVertices和aggregate Messages)和一个经过优化的Pregel API变体。此外，GraphX还包括一个持续增长的用于简化图分析任务的图算法和构建器集合。

6. SparkR

R语言为进行统计数值分析和机器学习工作提供了一种环境。Spark在2015年6月添加了支持R的功能，以匹配其支持Python和Scala的功能。

除了为潜在的Spark开发人员多提供一种语言外，SparkR还让R程序员们可以做之前做不了的

许多事情，比如访问超过单一机器的内存容量的数据集，或者同时轻松地使用多个进程或在多个机器上运行分析。

三、Apache Spark未来展望

根据Spark官网信息，截至目前至少有500家大型组织在自己的生产系统中部署和使用Spark，包括Amazon、Autodesk、IBM、Yahoo、百度、腾讯等。

当下Spark最重要的核心组件仍然是Spark SQL。而在未来的几次发布中，除了性能上的更加优化外(包括代码生成和快速Join操作)，还要提供对SQL语句的扩展和更好的集成。未来发展的重点将是数据科学化和平台API化，除了传统的统计算法外，还包括学习算法，使得SparkR得到长足的发展，同时也会使Spark的生态系统越来越完善。此外，Tungsten项目和DAG可视化、调试工具等同样是持续的重点发展方向。

3.4 开源还是闭源

软件领域开源和闭源之争已经持续了几十年，自软件与硬件逐渐解耦的20世纪70年代就已经开始。著作权法和专利法对软件提供了法律保护，通过软件的著作权和专利权，来保障企业或个人的商业利益，这样可以推动企业和个人创新的积极性，进而推动整个产业与经济的良性发展，这也是早期软件厂商强调闭源的原因。开源软件诞生之初，并非出于商业利益，而是部分软件才子对自由、共享的一种理想化的追求。但真正让开源软件蓬勃发展的，还是其背后的商业利益的驱动(由早期的个人组成的社区，变成大量商业公司的加入，而且这些商业公司最终成为开源社区的主体)。也就是说，当开源软件为企业和个人真正带来商业利益的时候，它才获得了广泛的支持，也带来了广泛的创新动力，并让开源软件发展的理想与现实获得了相对完美的结合。著作权法和专利法同样为开源软件提供了相关保护，并非是闭源软件的专享法律。

当闭源软件和开源软件都在以共同的商业利益为发展驱动的时候，闭源与开源之间便不再是一对水火不相容的冤家了。如今，大部分IT公司，甚至过去以闭源著称的微软，如今所提供的IT产品都是开源与闭源相结合的。微软与昔日的老冤家Linux开源公司也结成战略合作关系。作为非IT行业的企业用户而言，对开源和闭源的选择也是相互融合的，以为企业获得利益的最大化为目标。

企业(或个人)用户选择开源或闭源产品的基本原则是：首先满足业务基本功能需求；其次保持业务的敏捷高效、安全可靠及易用性；第三保持业务运营的低成本(最高的ROI)；第四要保证业务的自主定制(业务本身灵活性)；第五是需要和企业的生态环境无缝融合。无论是开源软件还是闭源软件，哪个能够满足企业的这些需求，哪个就可以获得广泛的市场。ORACLE、DB2的闭源数据库以其企业级的安全性、可靠性一直占据传统企业核心业务的主导地位，而开源的SQL/noSQL数据库，特别是非关系型数据库，随着自身在企业级特性上不断走向成熟，市场空间也在不断壮大。开源的Android操作系统在市场空间上已经完全超越了iOS，当然也不是意味着iOS彻底失败，苹果依旧占据着整个手机市场的最多利润空间，而消费者则在Android操作系统上获得了更多的实惠。

在公有云领域，对开源与闭源选择的判断也是基于企业(个人)用户需求出发的。全球大型的互联网公有云厂商，几乎都是基于开源软件建设的公有云平台，只是基于开源，却没有很好地回馈开源，没有把自己后续对开源软件的改进完全地贡献出来(或者贡献率很低)，这使得这些公有云平台，虽然基于开源软件，但经过大量改动，已经趋向闭源平台了。如果他们的平台所提供的服务功能够完善，安全性、稳定性够高，成本够低，平台够开放，便不影响其客户与市场的规模。但是事实往往不是如此。当市场垄断性或领先企业追求利益最大化的时候，其对平台的开放性与服务的价格降低方面便会失去积极性，同时

随着平台规模的扩大，其平台安全性与稳定性的挑战也会越高。另外一个方面，与开源社区保持同步(基于开源、回馈开源)的公有云平台，正在快速成长和壮大，其平台提供公有云服务的丰富性、稳定性、安全性都在不断增强，而其平台所具有的独特开放性更是领先于其他公有云厂商，这些独特优势都让与开源社区同步的公有云平台不断获得大量的新客户。至今，在公有云市场，闭源与开源之争并无胜负之分，而且闭源与开源的界限也比较模糊。企业或个人用户，则是根据自己需求的优先级来选择开源或闭源的公有云服务。

在私有云领域，闭源软件与开源软件的竞争则显得更加激烈。随着开源虚拟化软件已经走向成熟，以及开源容器技术的快速发展，闭源软件面临的压力在不断加大(包括价格压力、产品功能性压力、容器对虚拟化的替代性压力等)，虚拟化毕竟是闭源云软件厂商的主要收入来源。而在虚拟化之上的云管理平台，闭源软件厂商在产品成熟度及功能丰富性方面并没有什么优势。OpenStack开源软件快速发展，使得闭源软件厂商在私有云市场空间上发展缓慢甚至有微缩的态势。作为企业用户而言，选择基于开源软件的产品已经没有任何技术性障碍，作为拥有技术研发实力的一些企业用户，甚至可以直接从开源社区获取代码使用，自己进行定制化的增强。

在大数据领域，整个发展态势有些向开源一边倒了。因为大数据技术的广泛发展就是基于开源技术与开源社区的。开源大数据平台本身天然的容错性、基础架构的优越性与开放性，也为其快速发展提供了技术保障，活跃的开源社区生态让开源社区技术一直走在业界大数据技术的发展的前列。大数据技术的发展，与企业用户在大数据领域的探索相结合，让技术与应用相辅相成，协同发展，从而让开源成为大数据市场的主流。在大数据领域的闭源软件，则更多地侧重于各行业或特定企业的具体上层应用，很少有闭源软件自立门户，搭建一个与众不同的大数据技术基础架构。

开源和闭源软件作为一种竞争性的存在，通过相互竞争促进了相互发展，不断提升自身的功能、性能与价格竞争力。如今开源与闭源在竞争的同时又存在很多的互补、互融与互相转换。双方的竞争给企业用户带来了质美价廉的产品和服务，而双方的互补又给IT厂商带来了差异化的收入来源。双方共同组成了一个充满活力的软件生态体系。

第 10 章

大数据平台核心技术与架构

10.1 大数据特点与支撑技术

从技术角度来看，大数据所涵盖的范围往往比人们想象的窄很多。因为无论是大数据的存取，还是大数据的处理，大数据的分析要受限于当代IT技术发展的制约。正因为IT技术的制约，传统IT技术存储处理能力受限，所以从IT技术角度才感觉数据太“大”了。这个“大”表现在数据量的大，比如大部分存储设备都是TB级，几十TB到数百TB以及对传统存储而言是很高的数据量了。那么几十PB、成百上千PB的数据量自然对当前存储系统而言是“大数据”了。存储数据量的“大”很有时效性，因为大家都清楚，20世纪90年代初，1GB的存储空间都觉得“大”得不得了。现在说1GB，连PC机内存配置都觉得不够。数据量级之外的大数据，涵指数据格式的多样性，特别是大家已经熟知的图片、音频、视频、网页、日志等半结构化、非结构化的数据。除数据格式多样性之外，其还有数据实时性要求、数据价值密度等维度的指代。这就是通常大家比较认可的大数据4V特性。

- 体量(Volume)：非结构化数据超大规模地增长，占总数据量的80%～90%，比结构化数据增长快10倍到50倍，是传统数据仓库的10倍到50倍。
- 多样性(Variety)：大数据具有异构的多样性，拥有许多不同形式(文本、图像、视频、机器数据)，无模式或者模式不明显，拥有不连贯的语法或句义。
- 价值密度(Value)：有大量的不相关信息，对未来趋势与模式可预测分析，可进行深度复杂分析(机器学习、人工智能等)。
- 速度(Velocity)：实时分析而非批量式分析，对于实时的数据输入、处理与丢弃，分析结果立竿见影而非事后见效。

首先是对Volume的理解，非结构化数据一直存在，随着处理成本的降低和竞争的加剧，非结构化数据受到越来越多的重视。非结构化数据并不是完全没有“结构”的数据，而是包括类似图像、视频等比较难以计算的数据，以及日志等“结构”变化相对随意、不严格受控的半结构化数据。

第二，无论是结构化大数据，还是非结构化数据，处理的挑战都非常大，由于竞争的加剧，商业过程要求从这些数据中进行提取有效信息的时间大大缩短。数据量变大的同时，还要求提取过程缩短，并且提取过程本身也越来越复杂化，甚至对同一个业务问题的解答，要求用不同模型或者同一模型的不同参数加以对比印证。

不同性质、不同来源但是与同一个商业对象(比如客户)相关的各种数据(比如流数据、块数据和全局数据)都必须在一个系统中进行有效整合，形成对这个商业对象的有效认知，从而驱动商业流程。

大数据环境下，数据来源非常丰富且数据类型多样，存储和分析挖掘的数据量庞大，对数据展现的要求较高，并且很看重数据处理的高效性和可用性。传统的数据采集来源单一，且存储、管理和分析数据量也相对较小，大多采用关系型数据库和并行数据仓库即可处理。对依靠并行计算提升数据处理速度方面而言，传统的并行数据库技术追求事务写的一致性和容错性，难以保证其可用性和扩展性。传统的数据处理方法是以处理器为中心，而大数据环境下，需要采取以数据为中心的模式，减少数据移动带来的开销。因此，传统的数据处理方法已经不能适应大数据的需求。针对非结构化数据，利用分布式文件系统和MapReduce技术进行处理的Hadoop技术；针对实时类数据，采用流处理技术，如S4、Esper、Storm技术；针对结构化数据，采用已相对比较成熟的关系型数据库技术，如MPP RDB技术、SMP OLTP、MPP OLAP技术等。但是，任何技术的发展都是以业务为根本驱动的(见图10-1)。

然而，新的平台与技术的发展，也在逐步地改变图10-1中大数据处理技术的划分。以Spark为典型例子，依赖于其更先进的架构，不断优化的性能，以及日益成熟的生态，Spark已有完全取

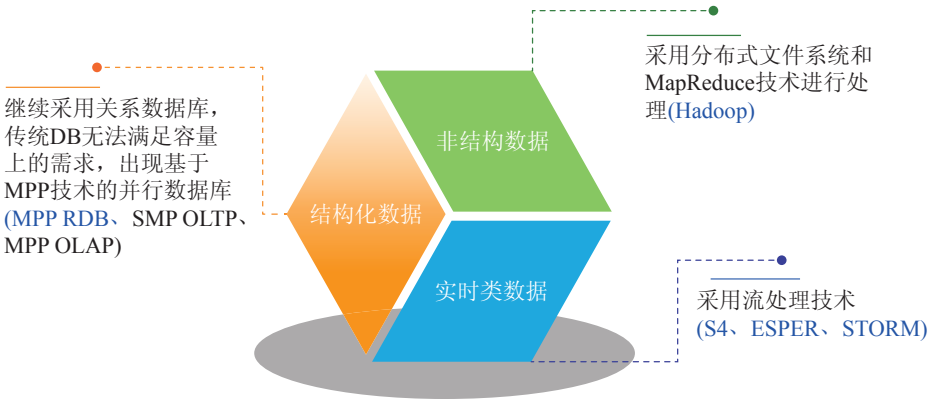


图10-1　大数据处理技术

代MapReduce成为Hadoop默认执行引擎之势；同时借助于统一混合型框架，Spark提供Spark SQL和Spark Streaming能力，使得其在结构化关系查询、实时类数据处理方面也能占据一席之地。

10.1.1　数据采集技术

数据是指通过传感器网络、社交网络及移动互联网等方式获得的各种类型的结构化、半结构化及非结构化的海量数据。要重点突破分布式高速、高可靠数据的抽取或采集、高速数据全映像等大数据收集技术；突破高速数据解析、转换与装载等大数据整合技术；设计质量评估模型，开发数据质量技术。

大数据采集一般分为大数据智能感知层：主要包括数据传感体系、网络通信体系、传感适配体系、智能识别体系及软硬件资源接入系统，实现对结构化、半结构化、非结构化的海量数据的智能化识别、定位、跟踪、接入、传输、信号转换、监控、初步处理和管理等。必须着重攻克针对大数据源的智能识别、感知、适配、传输、接入等技术。重点攻克分布式虚拟存储技术，大数据获取、存储、组织、分析和决策操作的可视化接口技术，大数据的网络传输与压缩技术，大数据隐私保护技术，等等。

10.1.2　数据预处理技术

本环节主要完成对已接收数据的辨析、抽取、清洗等操作。因获取的数据可能具有多种结构和类型，数据抽取过程可以帮助我们将这些复杂的数据转化为单一的或者便于处理的类型，以达到快速分析处理的目的。所有的大数据并不全是有价值的，有些数据并不是我们所关心的内容，而另一些数据则是完全错误的干扰项，因此要对数据进行过滤“去噪”，从而提取出有效的数据。

10.1.3　数据存储及管理技术

数据存储与管理要用存储器把采集到的数据存储起来，建立相应的数据库，并进行管理和调用。其重点解决复杂结构化、半结构化和非结构化大数据管理与处理技术，包括大数据的可存储、可表示、可处理、可靠性及有效传输等几个关键问题。大数据存储与管理还需要可靠的分布式文件系统、能效优化的存储、计算融入存储、大数据的去冗余及高效低成本的大数据存储技术、分布式非关系型大数据管理与处理技术、异构数据的数据融合技术，数据组织技术、大数据建模技术、大数据索引技术、大数据移动/备份/复制等技术、开发大数据可视化技术等。

10.1.4　数据分析及挖掘技术

大数据分析技术最近几年获得了很大的进展，包括改进已有数据挖掘算法和机器学习技术；开发数据网络挖掘、特异群组挖掘、图挖掘

等新型数据挖掘技术；突破基于对象的数据连接、相似性连接等大数据融合技术；突破用户兴趣分析、网络行为分析、情感语义分析等面向领域的大数据挖掘技术。

数据挖掘就是从大量的、不完全的、有噪声的、模糊的、随机的实际应用数据中，提取隐含在其中的、人们事先不知道的但又是潜在有用的信息和知识的过程。数据挖掘涉及的技术方法很多，有多种分类方法。

根据挖掘任务可分为分类或预测模型发现、数据总结、聚类、关联规则发现、序列模式发现、依赖关系或依赖模型发现、异常和趋势发现等。

根据挖掘对象可分为关系数据库、面向对象数据库、空间数据库、时态数据库、文本数据源、多媒体数据库、异质数据库、遗产数据库以及环球网Web等。

根据挖掘方法分，可粗分为机器学习方法、统计方法、神经网络方法和数据库方法。机器学习中，可细分为归纳学习方法(决策树、规则归纳等)、基于范例学习、遗传算法等。统计方法中，可细分为回归分析(多元回归、自回归等)、判别分析(贝叶斯判别、费歇尔判别、非参数判别等)、聚类分析(系统聚类、动态聚类等)、探索性分析(主元分析法、相关分析法等)等。神经网络方法中，可细分为前向神经网络(BP算法等)、自组织神经网络(自组织特征映射、竞争学习等)等。数据库方法主要是多维数据分析或OLAP方法，另外还有面向属性的归纳方法。

我们可从挖掘任务和挖掘方法的角度，着重突破。

- 可视化分析。数据可视化无论对于普通用户或是数据分析专家，都是最基本的功能。数据图像化可以让数据自己说话，让用户直观地感受到结果。
- 数据挖掘算法。图像化是将机器语言翻译给人看，而数据挖掘就是机器的母语。分割、集群、孤立点分析还有五花八门的算法供我们精炼数据，挖掘价值。这些算法一定要能够应付大数据的量，同时具有很高的处理速度。
- 预测性分析。预测性分析可以让分析师根据图像化分析和数据挖掘的结果做出一些前瞻性判断。
- 语义引擎。语义引擎需要有足够的人工智能以从数据中主动地提取信息。语言处理技术包括机器翻译、情感分析、舆情分析、智能输入、问答系统等。
- 数据质量和数据管理。数据质量与管理是管理的最佳实践，透过标准化流程和机器对数据进行处理可以确保获得一个预设质量的分析结果。

10.1.5 数据展现与应用技术

大数据技术能够将隐藏于海量数据中的信息和知识挖掘出来，为人类的社会经济活动提供依据，从而提高各个领域的运行效率，大大提高整个社会经济的集约化程度。在我国，大数据将重点应用于以下几大领域：金融、电商、公共服务，例如商业智能技术、政府决策技术、电信数据信息处理与挖掘技术、电网数据信息处理与挖掘技术、气象信息分析技术、环境监测技术、大规模基因序列分析比对技术、Web信息挖掘技术、多媒体数据并行化处理技术、影视制作渲染技术以及其他各种行业的云计算和海量数据处理应用技术等。

10.2 企业级Hadoop

10.2.1 Apache Hadoop起源

Hadoop由Apache基金会于2005年秋天作为Lucene的子项目Nutch的一部分正式引入，该技术受到最先由Google开发的 Map/Reduce和Google File System(GFS)的启发。2006 年 3 月，Map/Reduce和HDFS(GFS的开源实现)分别被纳入称为Hadoop的项目中。Facebook向Apache基金会贡献了Hive后，Hadoop系统具备了以类SQL方式处理结构化数据的能力。2010年9月，Hive脱离Hadoop，成为Apache顶级项目。HDFS、

MapReduce、Hive分别对应分布式系统的存储、计算框架、结构化分析能力。

MapReduce是Google在2004年提出的一个编程模型，用于大规模数据集(大于1TB)的并行运算。概念“Map(映射)”、“Reduce(化简)”以及其主要思想都是从函数式编程语言里借来的，并从矢量编程语言里借来了特性。其极大地方便了编程人员在不会分布式并行编程的情况下，将自己的程序运行在分布式系统上。

GFS(Google File System)是Google在2003年发表的文章，但现在仍被广泛讨论，其对后来的分布式文件系统设计具有指导意义。GFS的主要假设为GFS的服务器都是普通的商用计算机，并不可靠，集群出现节点故障是常态。系统存储适当数量的大文件，理想的负载是几百万个文件，文件一般都超过100MB，GB级别以上的文件是很常见的，必须进行有效管理。负载通常包含两种读，即大型的流式读(顺序读)和小型的随机读。前者通常一次读数百KB以上，后者通常在随机位置读几个KB。从这些假设基本可以看出GFS期望的应用场景应该是大文件、连续读、不修改、高并发。HDFS是GFS的开源实现。

Hive是Hadoop项目中的一个子项目，由Facebook向Apache基金会贡献。Hive被视为一个数据仓库工具，可以将结构化的数据文件映射为一张数据库表，并可以将SQL语句转换为 MapReduce 任务进行运行。其优点是学习成本低，可以通过类SQL语句快速实现简单的MapReduce任务，不必开发专门的MapReduce应用，十分适合数据仓库的统计分析。

从方案的架构来看，Hadoop已形成一整套完整的生态环境，是当前最重要的大数据平台之一，具有良好的并行处理能力、可扩展性和伸缩能力，非常适合处理半结构化、非结构化的类文本数据，如网页、日志等。

10.2.2 企业级Hadoop总体框架

企业级Hadoop可系统化地对Apache Hadoop进行增强，让对Hadoop缺乏了解的专业人员或者只有少量Hadoop专业人员的企业能够利用Hadoop的处理能力，解决本企业面临的与大数据相关业务挑战。

企业级Hadoop对外提供大容量数据分析和查询能力，解决各大企业的以下需求(见图10-2)：

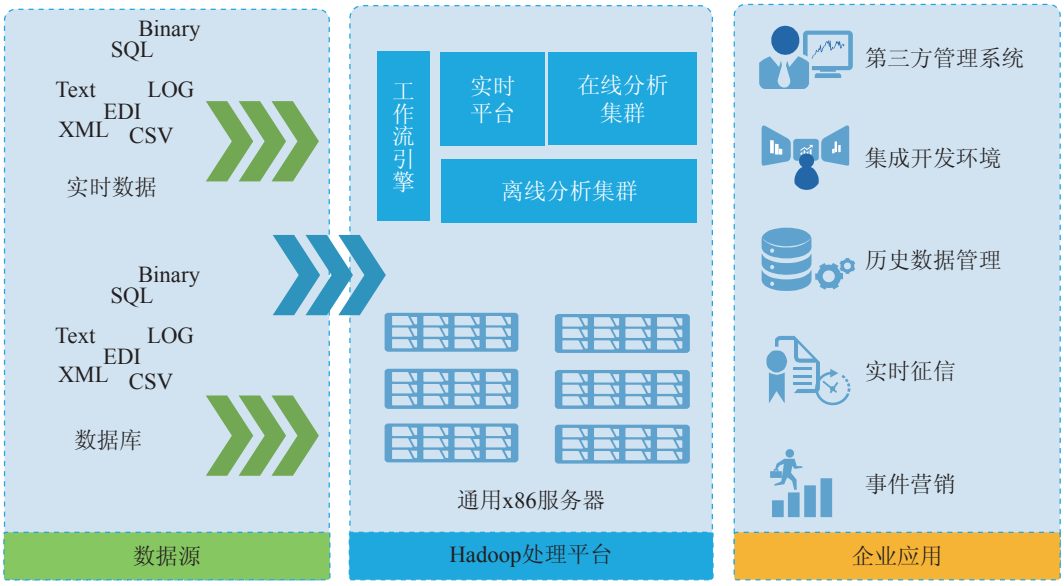


图10-2 Hadoop企业产品定位

- 快速地整合和管理不同类型的大容量数据；
- 对原生形式的信息采用高级分析；
- 可视化所有的可用数据，供特殊分析使用；
- 为构建新的分析应用程序提供开发环境；
- 工作负荷的优化和调度。

产品化的Hadoop需要在开源Hadoop版本的基础上对HBase、HDFS和MapReduce等组件增加HA、查询和分析功能，并进行性能优化。

Hadoop需要支持多种服务器的硬件平台，供企业根据自身需求灵活选择。用户可以通过简单地叠加相应服务的内存要求来计算所需要的内存总和。

产品化Hadoop软件系统整体结构如图10-3所示。

企业级Hadoop产品，需要对开源组件进行封装和增强，对外提供稳定的数据分布式存储和分析能力，包括数据的访问、存储、处理和保护功能，分为HDFS、HBase、MapReduce和ZooKeeper。

- HDFS：Hadoop分布式文件系统(Hadoop Distributed File System)能提供高吞吐量的数据访问，适合大规模数据集方面的应用。
- HBase：提供海量数据存储功能，是一种构建在HDFS之上的分布式、面向列的存储系统。
- MapReduce：提供快速并行处理大量数据的能力，是一种分布式数据处理模式和执行环境。
- ZooKeeper：提供分布式、高可用性的协调服务能力，帮助系统避免单点故障，从而建立可靠的应用程序。

10.2.3 HDFS

HDFS，即Hadoop分布式文件系统(Hadoop Distributed File System)，能提供高吞吐量的数据访问，适合大规模数据集方面的应用。HDFS包含主、备NameNode和多个DataNode。HDFS是一个Master/Slave的结构，在Master上运行NameNode，而在每一个Slave上运行DataNode。NameNode和DataNode之间的通信都是建立在TCP/IP的基础之上的。NameNode和DataNode均被设计为可以部署在Linux服务器上(见图10-4、表10-1)。

HDFS原理如图10-5所示。

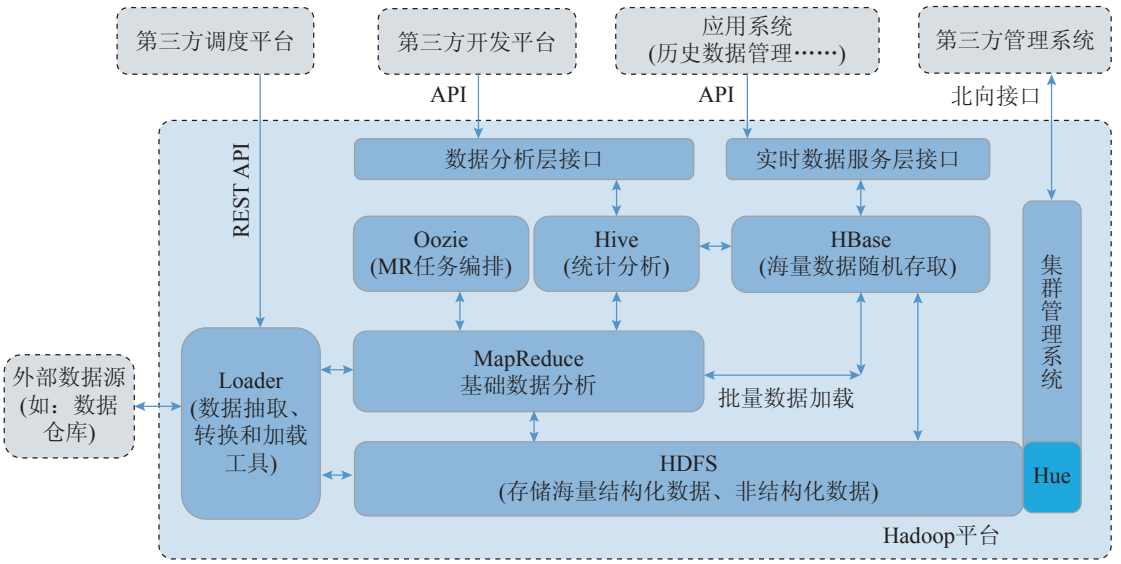


图10.3 Hadoop软件系统架构

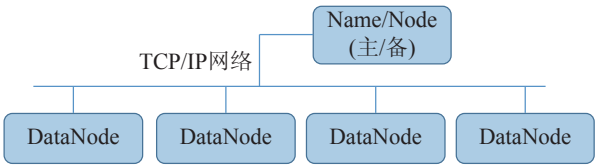


图10-4　HDFS结构

表10-1　HDFS各功能模块说明表

名称	描述
NameNode	用于管理文件系统的命名空间、目录结构、元数据信息以及提供备份机制等，分为： • Active NameNode：管理文件系统的命名空间、维护文件系统的目录结构树以及元数据信息，记录写入的每个“数据块”与其归属文件的对应关系 • Secondary NameNode：对Active NameNode进行监控，对Active NameNode中的数据进行备份，随时准备在Active NameNode出现异常时接管其服务
DataNode	用于存储每个文件的“数据块”数据，并且会周期性地向NameNode报告该DataNode的存放情况

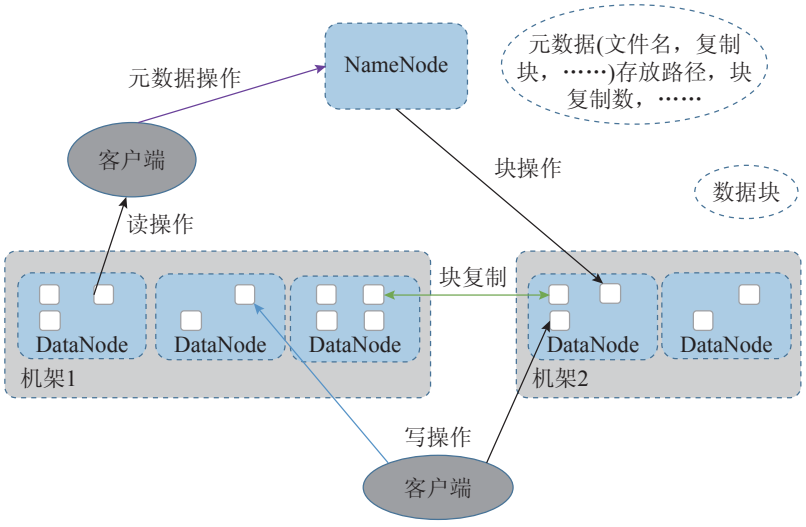


图10-5　HDFS工作原理图

在HDFS内部，一个文件可分为一个或多个“数据块”，这些“数据块”存储在DataNode集合里。客户端连接到NameNode，执行文件系统的“命名空间”操作，例如打开、关闭、重命名文件和目录，同时决定“数据块”到具体DataNode节点的映射。DataNode在NameNode的指挥下进行“数据块”的创建、删除和复制。

因为NameNode负责保管和管理所有的HDFS元数据，所以用户数据的读写就不需要通过NameNode，而是直接在DataNode上进行。

10.2.4　MapReduce

MapReduce是一种简化并行计算的编程模型，名字源于该模型中的两项核心操作：Map和Reduce。Map将一个任务分解成为多个任务，Reduce将分解后多任务处理的结果汇总起来，得出最终的分析结果。MapReduce模型主要由ResourceManager、ApplicationMaster和NodeManager组成，如图10-6所示。

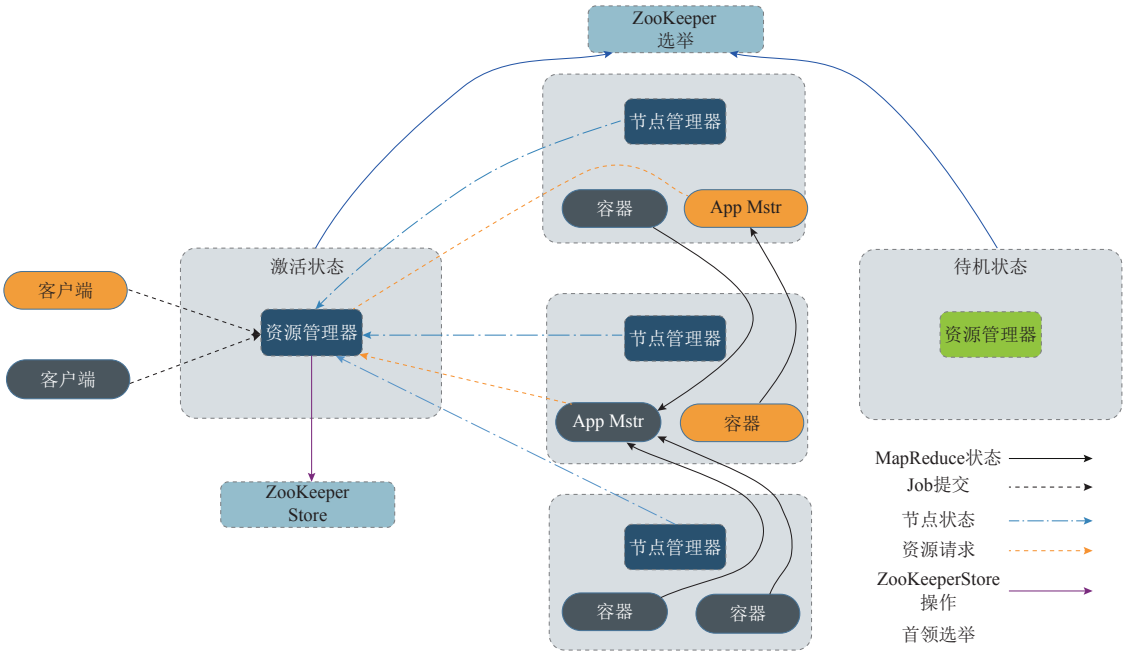


图10-6　MapReduce框架(基于Yarn)架构

MapReduce各部分的功能说明如表10-2所示。

表10-2　结构图说明

名称	描述
Client	MapReduce的客户端，可以通过客户端向ResourceManager发起MapReduce操作
ResourceManager	MapReduce的资源管理器，基于应用程序对资源的需求进行调度；由于每一个应用程序需要不同类型的资源，因此需要不同的容器；同时，资源管理器提供一个调度策略的插件，它负责将集群资源分配给多个队列和应用程序；调度插件可以基于现有的能力调度和公平调度模型进行调度
NodeManager	负责执行应用程序的容器，同时监控应用程序的资源使用情况(CPU、内存、硬盘、网络)，并向ResourceManager汇报
ApplicationMaster	负责相应的调度和协调，结合从ResourceManager获得的资源和NodeManager协同工作来运行和监控任务
Container	作为资源隔离，当前仅仅提供java虚拟机CPU、内存的隔离

新的Hadoop MapReduce框架命名为MapReduceV2或者Yarn。Hadoop MapReduce新框架主要分为ResourceManager、ApplicationMaster与NodeManager三个部分。

✍ ResourceManager核心服务，负责调度、启动每一个Job所属的ApplicationMaster、同时监控ApplicationMaster的运行情况。ResourceManager负责作业与资源的调度，并接收JobSubmitter提交的作业，按照作业的上下文(Context)信息，以及从NodeManager收集来的状态信息，启动调度过程，分配一个Container作为ApplicationMaster。

✍ NodeManager功能比较专一，就是负责Container状态的维护，并向ResourceManager保持

心跳。

✍ ApplicationMaster负责一个Job生命周期内的所有工作，类似老框架中的JobTracker。但注意每一个Job(不是每一种)都有一个ApplicationMaster，它可以运行在ResourceManager以外的机器上。

10.2.5 ZooKeeper

ZooKeeper是一个分布式、高可用性的协调服务。产品化Hadoop系统中主要提供两个功能：

✍ 帮助系统避免单点故障，建立可靠的应用程序；

✍ 提供分布式协作服务和维护配置信息。

ZooKeeper集群中的节点分为三种角色，即Leader、Follower和Observer，其结构和相互关系如图10-7所示。

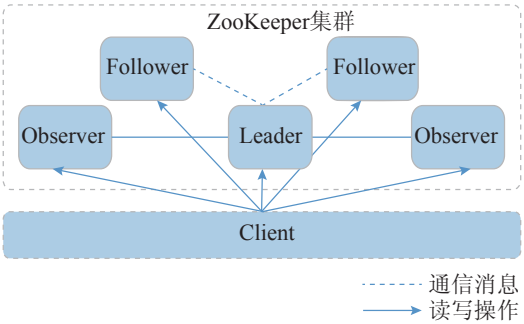


图10-7 ZooKeeper结构图

ZooKeeper各部分的功能说明如表10-3所示。

表10-3 ZooKeeper各部分的功能说明

名称	描述
Leader	在Zookeeper集群中只有一个节点作为集群的领导者，由各Follower通过Paxos算法选举产生，主要负责接受和协调所有写请求，并把写入的信息同步到Follower和Observer
Follower	Follower的功能有两个： • 每个Follower都作为Leader的储备，当Leader故障时重新选举Leader，避免单点故障； • 配合Leader一起进行写请求处理
Observer	Observer不参与选举，负责接受写请求、处理读请求，避免系统处理能力浪费
Client	Zookeeper集群的客户端，对Zookeeper集群进行读写操作。例如HBase可以作为Zookeeper集群的客户端，利用Zookeeper集群的仲裁功能，控制其HMaster的“Active”和“Standby”状态

ZooKeeper写请求原理具体如下：

✍ Follower或Observer接收到写请求后，转发给Leader；

✍ Leader协调各Follower，通过投票机制决定是否接受该写请求；

✍ 投票结果为接受，则由Leader处理写请求；

✍ 数据写入完成后，Leader向Follower和Observer同步，保证所有节点的数据一致性。

ZooKeeper只读请求原理：客户端直接向Leader、Follower或Observer读取数据。

10.2.6 HBase

HBase是一种构建在HDFS(Hadoop Distributed File System)之上的分布式、面向列的存储系统，它具有高可靠、高性能、面向列和可伸缩的特性。HBase适合于存储大表数据(表的规模可以达到数十亿行以及数百万列)，并且对大表数据的读、写访问可以达到实时级别。

HBase集群由主备HMaster进程和多个RegionServer进程组成，如图10-8所示。

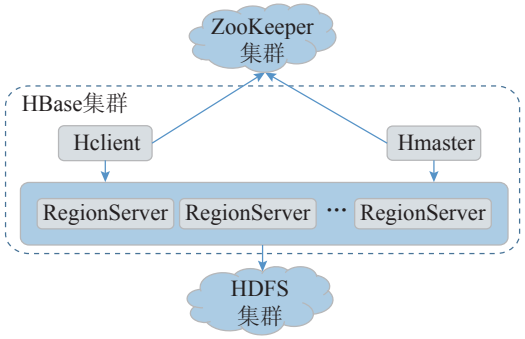


图10-8 HBase结构

HBase各部分的功能说明如表10-4所示。

表10-4 HBase功能说明

名称	描述
主用HMaster	主用HMaster负责管理RegionServer节点，同时维护集群的网络拓扑以及集群的负载均衡
备用HMaster	当主用HMaster故障时，备用HMaster将取代主用HMaster对外提供服务；故障恢复后，原主用HMaster降为备用
Region-Server	RegionServer负责提供表数据读写等服务，是HBase的数据处理和计算单元；RegionServer一般与HDFS集群的DataNode合设，实现数据的存储功能
ZooKeeper集群	ZooKeeper为HBase集群中各进程提供分布式协作服务；各RegionServer将自己的信息注册到Zookeeper中，HMaster据此感知各个RegionServer的健康状态
HDFS集群	HDFS为HBase提供高可靠的文件存储服务，HBase的数据全部存储在HDFS中

HBase以表的形式存储数据，数据模型如图10-9所示。表中的数据划分为多个Region，并由HMaster分配给对应的RegionServer进行管理，每个Region包含了表中一段Row Key区间范围内的数据，HBase的一张数据表开始只包含一个Region，随着表中数据的增多，当一个Region的大小达到容量上限后会分裂成两个Region。

HBase数据模型中各列的说明如表10-5所示。

表10-5 数据模型列说明表

名称	描述
Row Key	表的主键，数据存储时表中的记录按照Row Key的字符序进行排序
Timestamp	插入数据时对应的时间戳，HBase支持相同Row Key的多版本数据存储
Column Family	列族，HBase中表在水平方向上由一个或者多个Column Family组成，一个Column Family由任意多个Column组成
Column	列，与传统的数据库类似，HBase的表中也有列的概念，列用于表示相同类型的数据

RegionServer数据存储：RegionServer主要负责管理由HMaster分配的Region，RegionServer的数据存储结构如图10-10所示。

RowKey	Timestamp	Column Family 1		Column Family N		
		URL	Content	Column1	Column2	
Row1	t2	www.huawei.com	“<html>”			Region
	t1	www.huawei.com	“<html>”			
…	…	…	…	…	…	
RowM	…	…	…	…	…	
RowM+1	…	…	…	…	…	Region
RowM+2	t1	…	…	…	…	
	t3	…	…	…	…	
	t2	…	…	…	…	
	t1	…	…	…	…	
…	…	…	…	…	…	Region
RowN	t1	…	…	…	…	
…	…	…	…	…	…	

图10-9 HBase数据模型

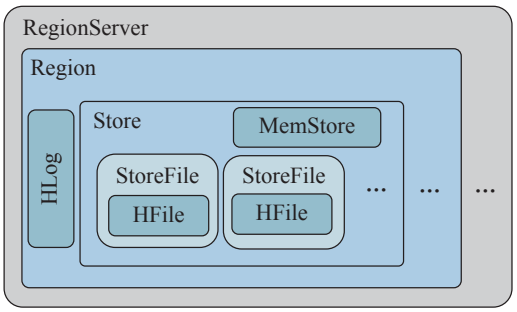


图10-10　RegionServer的数据存储结构

Region的各部分的说明如表10-6所示。

表10-6　Region结构说明

名称	描述
Store	一个Region由一个或多个Store组成，每个Store对应图10-10中的一个Column Family
MemStore	一个Store包含一个MemStore，MemStore缓存客户端向Region插入的数据，当RegionServer中的MemStore大小达到配置的容量上限时，RegionServer会将MemStore中的数据“flush”到HDFS中
StoreFile	MemStore的数据“flush”到HDFS后成为StoreFile，随着数据的插入，一个Store会产生多个StoreFile，当StoreFile的个数达到配置的最大值时，RegionServer会将多个StoreFile合并为一个大的StoreFile
HFile	HFile定义了StoreFile在文件系统中的存储格式，它是当前HBase系统中StoreFile的具体实现
HLog	HLog日志保证了当RegionServer发生故障时用户写入的数据不丢失，RegionServer的多个Region共享一个相同的HLog

元数据表是HBase中一种特殊的表，用来帮助Client定位到具体的Region，包括“.META.”表和“-ROOT-”表。

- “.META.”表：记录用户表的Region信息，例如Region位置、起始Row Key及结束Row Key等信息。
- “-ROOT-”表：记录“.META.”表的Region信息。“-ROOT-”表不会分裂，因此只有一个Region包含“-ROOT-”表。

元数据表和用户表的映射关系如图10-11所示。

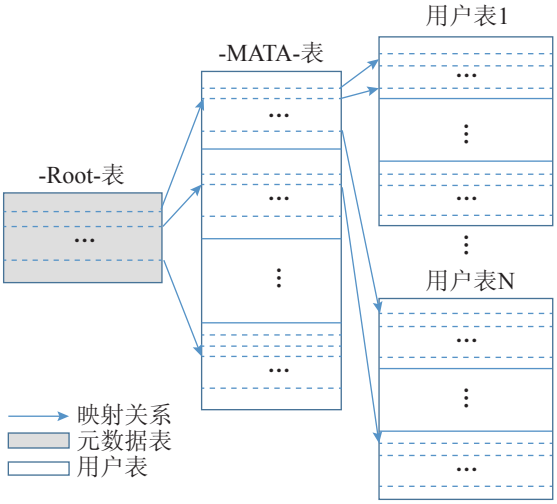


图10-11　元数据表和用户表的映射关系

数据操作流程

HBase数据操作流程如图10-12所示。

(1) 当Hadoop对HBase进行增、删、改、查数据操作时，HBase Client首先连接ZooKeeper获得“-ROOT-”表所在的RegionServer的信息。

(2) HBase Client连接到“-ROOT-”表的Region所在的RegionServer，并获得“.META.”表的Region的信息。

(3) HBase Client连接到包含对应的“.META.”表的Region所在的RegionServer，并获得相应的用户表的Region所在的RegionServer位置信息。

(4) HBase Client连接到对应的用户表Region所在的RegionServer，并将数据操作命令发送给该RegionServer，RegionServer接收并执行该命令，从而完成本次数据操作。

为了提升数据操作的效率，HBase Client会在内存中缓存“-ROOT-”、“.META.”和用户表Region的信息，当应用程序发起下一次数据操作

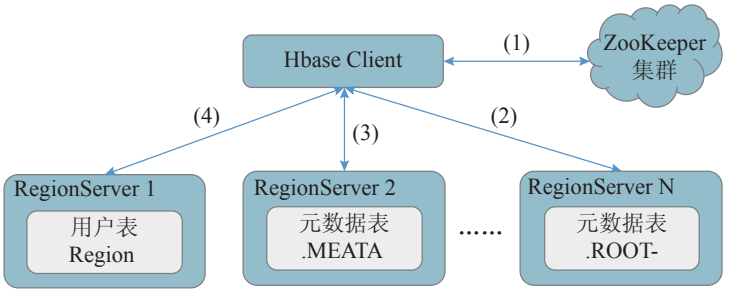


图10-12　HBase数据操作流程

时，HBase Client会首先从内存中获取这些信息；当内存中缓存的数据信息与系统中的实际信息不一致时，HBase Client会重复上述操作。

10.2.7　Hive

Hive是一个基于Hadoop的开源的数据仓库平台，通过Hive，可以方便地进行数据提取转化加载(ETL)的工作，提供类似SQL的HQL语言，操作结构化数据存储服务和基本的数据分析服务。HQL能够将用户编写的SQL转化为相应的MapReduce程序。当然，用户也可以自定义Mapper和Reducer来完成复杂的分析工作。基于MapReduce的Hive具有良好的扩展性和容错性。不过由于MapReduce缺乏结构化数据分析中有价值的特性，以及Hive缺乏对执行计划的充分优化，导致Hive在很多场景下比并行数据仓库慢(在几十台机器的小规模下可能相差更大)。

Hive主要特点如下：

- 海量结构化数据分析汇总；
- 将复杂的MapReduce编写任务简化为SQL语句；
- 灵活的数据存储格式，支持JSON、CSV、TEXTFILE、RCFILE、SEQUENCEFILE这几种存储格式。

为保证Hive服务的高可用性、用户数据的安全及访问服务的可控制性，在开源社区的Hive-0.9.0版本上新增双机特性和安全特性。

开源社区的Hive特性，请参见https://cwiki.apache.org/confluence/display/Hive/DesignDocs。

Hive双机特性采用基于双机设备，即基于主备切换方式的服务器设备来保证Hive服务的高可用性。在同一时间内只有一个Hive Server运行，当主Hive Server出现故障无法继续提供服务时，备Hive Server会被激活，保证业务在短时间内完全恢复正常使用。

通过Zookeeper的Master Election机制保证总有一个Hive Server正常运行并提供服务。主备两个HiveServer以Ephemeral方式分别注册到Zookeeper中，使得其中一个Hive Server出现故障时，能够及时选举并恢复服务。同时，Zookeeper中存储主Hive Server的IP地址。Hive客户端可通过Zookeeper获取主Hive Server的IP地址来连接Hive Server以获得服务。

在双机环境下，需要知道Zookeeper的IP地址和端口才能获取主Hive Server的地址；在安全环境下，除了需要Zookeeper的IP地址和端口信息外，还需要连接Zookeeper的相关认证信息。

Hive用户认证采用的是基于Kerberos的认证。Kerberos是一种适用于在公共网络上进行分布计算的工业标准的安全认证系统。用户通过Hive客户端和Hive Server建立连接时，需要进行双向认证。当用户是Kerberos KDC的合法用户时，其才能够通过认证，访问Hive的服务。认证方法为用户使用用户名(Principal)及Keytab文件登录KDC，登录成功，则表示认证通过。

Hive使用user、group、role对权限进行管理。在Hive中，进行数据定义、加载和查询都需要相应的操作权限。

Hive作为强大的数据仓库和数据分析平台至少需要具备以下几点特性：

- 灵活的存储引擎；
- 高效的执行引擎；
- 性能良好的索引机制；
- 良好的可扩展性；
- 强大的容错机制；
- 多样化的可视化。

先看看Hive是否完全具备了以上几点，以及与传统的并行数据仓库对比优劣如何。

一、存储引擎

Hive没有自己专门的数据存储格式，也没有为数据建立索引，用户可以非常自由地组织Hive中的表，只要在创建表时告诉Hive数据中的列分隔符和行分隔符，Hive就可以解析数据。Hive的元数据存储在RDBMS中，所有数据都基于HDFS存储。Hive包含Table、External Table、Partition和Bucket等数据模型。

并行数据仓库需要先把数据装载到数据库中，按特定的格式存储，然后才能执行查询。每天需要花费几个小时将数据导入并行数据库中，而且随着数据量的增长和新的数据源加入，导入时间会越来越长。导入时大量的写I/O与用户查询的读I/O产生竞争，会导致查询的性能很差。

Hive执行查询前无须导入数据，直接执行计划。Hive支持默认的多种文件格式，同时可以通过实现MapReduce的InputFormat或OutputFormat类，由用户定制格式。因为公司的数据种类很多，存储于不同的数据源系统，如MySQL、HDFS或者Hypertable等，很多时候Hive的分析过程会用到各种数据源的数据。当然使用多个存储数据源，除了功能上要能够支持导入/导出之外，如何根据各种存储源的能力和执行流获得最优执行计划也是一件麻烦的事情。

二、执行引擎

MR对于Map和Reduce job间的数据传输处理方式具有潜在的性能问题。假设有N个Map instance，每个产生M个输出文件，每个输出文件指向不同的Reduce instance。这些文件会被写到执行Map instance节点的本地磁盘。如果N是1000，M是500，那么Map阶段将会产生500 000个本地文件。当Reduce阶段开始时，500个Reduce instance中每个都需要读取1000个输入文件，同时必须使用一个文件传输协议从每个Map instance所运行的节点处拉取输入文件。假设同时有100个Reduce instance并行执行，不可避免地将会有两个或者更多的Reduce instance同时在同一个节点上试图读取文件，这就会产生大量的磁盘seek操作，从而降低磁盘传输效率。这也是为什么并行数据库系统没有将它们的中间数据保存为文件，并且采用了一种推模式而不是拉模式进行数据传输的原因。

并行数据仓库使用优化器进行性能优化。在生成执行计划时，利用元数据信息估算执行流上各个算子要处理的数据量和处理开销，进而选取最优的执行计划。并行数据仓库实现了各种执行算子(Sort、GroupBy、Union和Filter等)。优化器可以灵活地选择这多个算子以实现不同类别的性能优化。此外，并行数据仓库还拥有完备的索引机制，包括磁盘布局、缓存管理和I/O管理等多个层面的优化，这些都对查询性能至关重要，而这恰恰是Hive的不足之处。

Hive的编译器负责编译源代码并生成最终的执行计划，包括语法分析、语义分析、目标代码生成，所做的优化并不多。Hive基于MapReduce，Hive的Sort和GroupBy都依赖MapReduce。而MapReduce相当于固化了执行算子，Map的MergeSort必须执行，GroupBy算子也只有一种模式，Reduce的Merge-Sort也必须可选。另外Hive对Join算子的支持也较少。另外，内存复制和数据预处理也会影响Hive的执行效率。当然，数据预处理可能会影响数据的导入效率，这需要根据应用特点进行权衡。

三、索引机制

所有的现代DBMS都使用hash或者B树索引来加速数据访问。如果某人要查找一个记录子集(比如工资大于100 000美元的雇员)，那么使用合适的索引可以明显缩小查询的范围。大部分数据库

都允许单个表格具有多个索引。因此，查询优化器可以决定为用户查询使用哪个索引或者是简单地采用一个暴力的顺序搜索。

Hive在加载数据的过程中不会对数据进行任何处理，甚至不会对数据进行扫描，因此也没有对数据中的某些Key建立索引。Hive要访问数据中满足条件的特定值时，需要暴力扫描整个数据，因此访问延迟较高。由于MapReduce的引入，Hive可以并行访问数据，即使没有索引，对于大数据量的访问，Hive仍然可以体现出优势。由于数据的访问延迟较高，决定了Hive不适合在线数据查询。

四、扩展性

并行数据仓库可以很好地扩展到几十或上百个节点的集群，并且达到接近线性的加速比。然而，今天的大数据分析需要的可扩展性远远超过这个数量，经常需要达到数百甚至上千节点。目前，几乎没有哪个并行数据仓库运行在这么大规模的集群上，这涉及多个方面的原因。并行数据仓库假设底层集群节点完全同构；并行数据仓库认为节点故障是很少出现的；并行数据仓库设计和实现基于的数据量并未达到PB级或者EB级。

与并行数据仓库不同的是，Hive更加关注水平扩展性。简单来讲，水平扩展性指系统可以通过简单地增加资源来支持更大的数据量和负载。

Hive处理的数据量是PB级的，而且每小时每天都在增长，这就使得水平扩展性成为一个非常重要的指标。Hadoop系统的水平扩展性是非常好的，基于MapReduce框架，Hive能够很自然地利用这一点。

五、容错性

Hive具有较好的容错性。Hive的执行计划在MapReduce框架上以作业的方式执行，每个作业的中间结果文件写到本地磁盘，最终输出文件写到HDFS文件系统，利用HDFS的多副本机制来保证数据的可靠性，从而达到作业的容错性。如果在作业执行过程中某节点出现故障，那么Hive执行计划基本不会受到影响。因此，基于Hive实现的数据仓库可以部署在由普通机器构建的分布式集群之上。

当某个执行计划在并行数据仓库上运行时，若某节点发生故障，则必须重新执行该计划。所以，当集群中的单点故障(可能是磁盘故障等)发生率较高时，并行数据仓库的性能就会下降。在实际生产环境中，假设每个节点故障发生率是0.01%，那么1000个节点的集群中单点故障发生率则为10%。这个数字并非耸人听闻，处理海量数据的I/O密集型应用集群，平均每月的机器故障率达到1%～10%。当然这些机器可能是2万～3万元的普通机型。

六、可视化

Hive的可视化界面基本属于字符终端，用户的技术水平一般比较高。面向不同的应用和用户，提供个性化的可视化展现，是Hive改进的一个重要方向。个性化的可视化也可以理解为用户群体的分层，例如图形界面方式提供初级用户，简单语言方式提供中级用户，复杂程序方式提供高级用户。

10.2.8 Spark

Spark是新一代的大数据分布式处理平台，旨在提供快速、易用以及通用的大数据处理和分析能力。Spark基于内存计算，相较MapReduce可以显著提升任务执行效率，同时，其提供易用的高级APIs，支持Java、Scala、Python和R语言。在这之上，Spark还提供一系列更高级的工具，例如：针对SQL和结构化数据处理场景，Spark SQL提供关系型查询能力；针对机器学习和图计算，MLib和GraphX分别提供高级分析能力以及丰富的图算法库；针对流处理场景，Spark Streaming提供高吞吐、准实时的流处理能力。如图10-13所示。

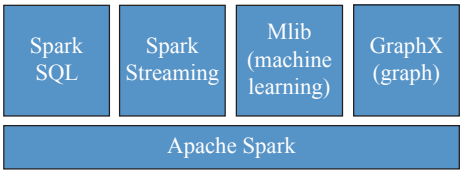


图10-13 Spark通用框架

Spark有如下两个核心抽象：

- 弹性分布式数据集(RDD，Resilient Distributed Dataset)，是Spark的基本数据结构，是一个只读的、分布式的对象集合，可全部或部分缓存在内存中，在多次计算间重用，同时提供容错性；
- 有向无环图(DAG，Direct Acyclic Graph)，施加于数据之上的运算拓扑，节点为RDD，边为变化操作。

Spark的主要特点具体如下。

- 高性能：依赖于先进的DAG执行引擎与内存计算，Spark相较于MapReduce可提升100倍的性能。
- 易用性：Spark提供多语言支持的高级API，用户可以快速、方便地利用这些API构建分布式并行应用，采用提交程序或交互式shell工具等方式。
- 通用性：Spark提供SQL、机器学习、图算法和流处理能力，用户可以在同一个应用内无缝地使用这些库能力。
- 高适配性：Spark可以通过standalone方式部署，也可以部署于Yarn、Mesos或是云上(例如EC2)；同时，Spark可以访问存储于不同平台的多种数据，包括HDFS、Cassandra、HBase、Hive、Alluxio等，以及任意Hadoop数据源。

Spark的主要结构，如图10-14所示。

结构图说明如表10-7所示。

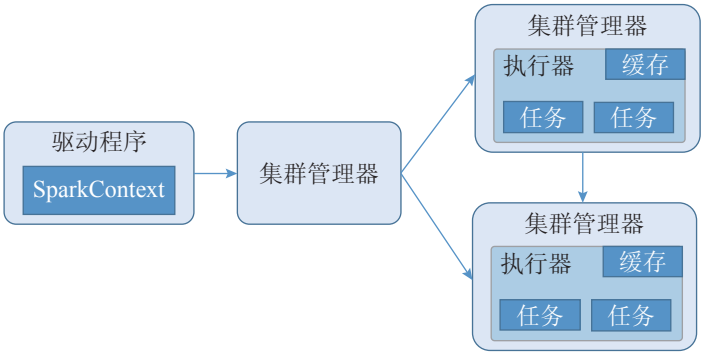


图10-14　Spark模块图

表10-7　结构图说明

名称	描述
Cluster Manager	集群管理器，Spark支持多种集群管理器，Spark自带的standalone集群管理器、Mesos或YARN
Driver Program	Spark应用程序运行时包含一个Driver进程，也是应用程序的主进程，负责应用程序的解析、生成Stage，并调度Task到Executor上
Executor	真正执行应用程序的地方，一个集群一般包含多个Executor，每个Executor接收Driver的命令Launch Task，一个Executor可以执行一到多个Task
Master Node	集群的主节点，负责接收客户端提交的作业，管理Worker，并命令Worker启动Driver和Executor
SparkContext	面向用户的Spark程序入口，是Spark应用程序的总控，负责实行分布执行计划和Task的调度；是用户基于业务逻辑定义的类，里面包含DAG(无回路有向图)；SparkContaxt会基于用户的业务逻辑，划分stage，并生成Task
Task	承载业务逻辑的运算单元，是Spark平台中可执行的最小工作单元；一个应用根据执行计划以及计算量分为多个Task
Cache	分布式缓存，每个Task可将结果放置于Cache，供多个后续Task读取

10.3 流处理技术

10.3.1 流处理的应用场景

对大数据技术不熟悉的人而言，流处理技术最容易让人联想到的是视频和音频这种流媒体实时处理，比如视频监控，一边拍着视频，一边对视频内容做实时的分析处理，比如在城市建设里，通过摄像头能动态查找犯罪嫌疑人。在反恐动作片里，利用大数据分析技术自动定位、查找出恐怖分子。很遗憾，目前大数据分析技术中，流处理技术还没有这么先进。之所以没有达到这么理想的效果，也不都是大数据分析处理的问题，除了大数据分析处理还不够强大外，配套设施也不是很先进，比如我们的摄像头还无法把在摄像范围内的人脸拍得那么清晰，特别是现在雾霾比较严重的情况下，摄像头的表现并没有比人眼出色多少。模糊处理技术也还不够智能，当前的数据分析最多能把活动目标进行锁定(而且还是在夜深人静、干扰很少的情况下)，并判断其活动轨迹和活跃度(速度)，至于这个活动目标是不是嫌疑人，还得求助警察通过进一步的工作来完成。

当前的流处理技术，更多地侧重于结构化、半结构化的文本类文件，如日志类文件的处理，其特征是：事件驱动、实时处理(数据先计算，不存储或后存储)、毫秒/秒级(分析响应速度要求)、实时监控、高级事件触发、立即事件浅加工。

流处理技术能解决哪些问题呢？下面举几个例子：

- 对于电信运营商：当前网络质量是什么状态？有什么异常点发生？
- 对于券商：当前是否可以买入或者卖出一支股票？
- 对于信用卡银行：当前是否发生了异常、欺诈交易？可以给用户提供什么推荐服务？
- 对于电商：客户正在购物，可以给其推荐什么商品？
- 对于社交媒体：用户当前的访问模型是什么？当前热点话题是什么？用户的类型、兴趣是什么？
- 对于城市道路交通系统：当前各道路流量、车速情况有什么异常，需要采取什么措施？
- 对于物流企业：当前货物库存、周转、运输状态是什么？有没有异常？

……

10.3.2 流处理技术关键概念

流处理技术利用时间“窗口”概念来让“流”中的数据有了“边界”。窗口中的数据等于“数据库中的一张静态表”(见图10-15)。

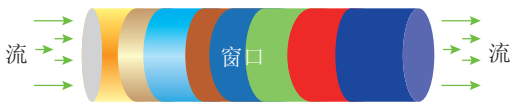


图10-15 窗口概念

“窗口”又可分为跳动窗口和滑动窗口，用于不同场景的分析处理(见图10-16、图10-17)。

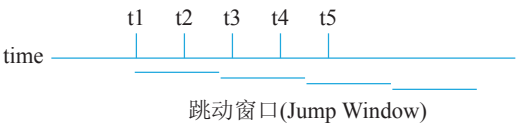


图10-16 跳动窗口

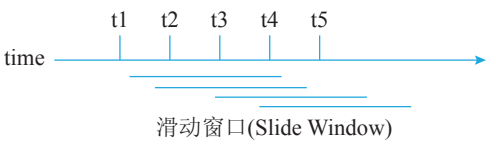


图10-17 滑动窗口

例如：在电信业务中，其可以利用窗口的概念来进行数据流分析。电信信令监控的某业务是否是恶意呼叫，需寻找在连续1分钟内发起大于5次呼叫的手机用户(见图10-18)，语句如下：

```
SELECT * FROM CallEvent.win:time(60 sec)
GROUP BY strImsi HAVING count(*)>5
```

在流处理中，这个查询语句是持续作用的，流入的事件满足条件即被触发持续查询，这一特性叫“持续查询”。

通过查询处理，分析并发现数据反映出的关系和问题，叫“模式发现”，可以采用以下分析方式“(模式)发现”(见图10-19)：

- 跟随模式，A事件发生后总会发生B事件；
- 非事件，不发生某个事件；
- 周期性发生，A事件总是在某一时间段内发生。

通过以上的文本搜索，获得以下分析结果，事件A发生后3秒内B或C跟随发生，在10秒时间内D不会发生，从而发现其中的问题(见图10-20)。

10.3.3 流处理技术辨析

目前流处理技术主要有两种：CEP(Complex Event Processing)用于复杂事件处理，Stream Processing用于流处理。两种技术的对比如图10-21所示。

随着流处理技术的不断发展，CEP技术与Stream Processing技术已有逐渐融合到统一平台之中的趋势。例如Apache Flink，一个开源的流批混合数据处理平台，在其最新发布的1.0版本中，支持CEP并提供了一套复杂事件处理库，无缝地整合了两种流处理技术所针对的场景需求。

CEP技术又分为基于查询和基于规则两种框架。基于查询(Query-Based)的CEP技术适用于高

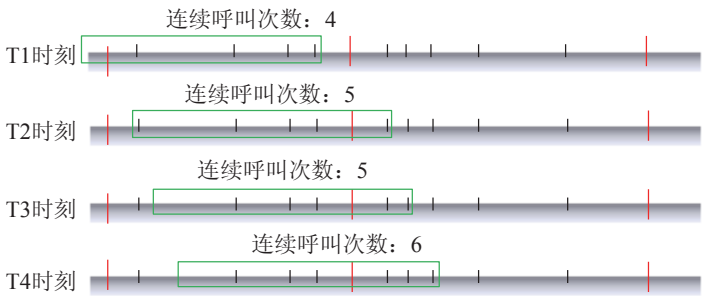


图10-18 恶意呼叫查询

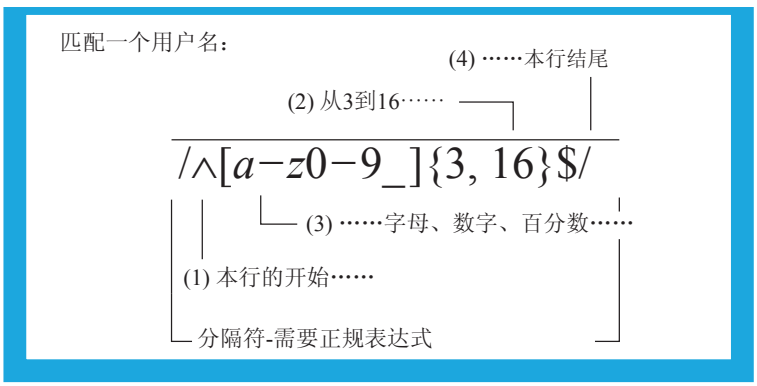


图10-19 文本搜索中的正则表达式

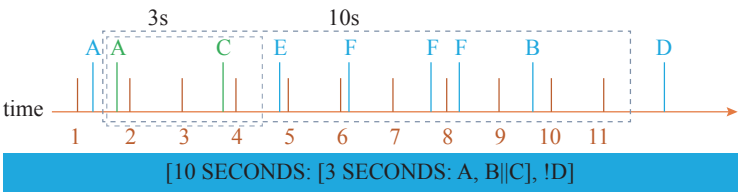


图10-20 异常发现

速数据流分析，基于规则的CEP技术则是“条件-规则-行动”的模式。

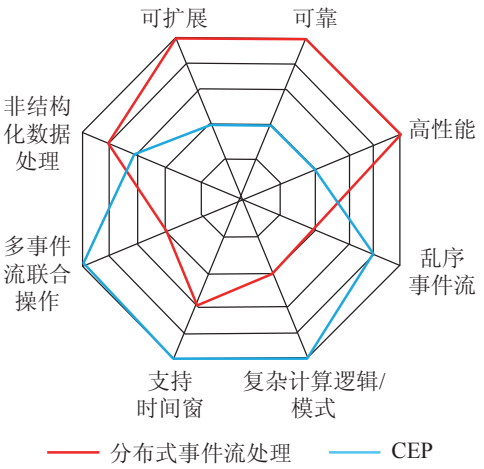


图10-21　流处理技术对比

基于查询(Query-Based)的CEP技术框架如图10-22所示。

代码样例如图10-23所示。

基于查询(Query-Based)的CEP技术基本特征：

- 支持基于类SQL查询语言；
- Antlr解析查询语句；
- 根据解析结果生成数据结构；
- 计算事件是否满足触发条件；
- 可以支持较高数据量。

基于查询(Query-Based)的CEP技术适用场景：

- 数据量非巨量，使用单节点可以满足需求；
- 较复杂类SQL业务，如KPI统计、定时输出结果、多流Join处理等。

基于规则(Rule-based)的CEP技术基本特征是：实时决策支持、数据吞吐量不大、基于“条

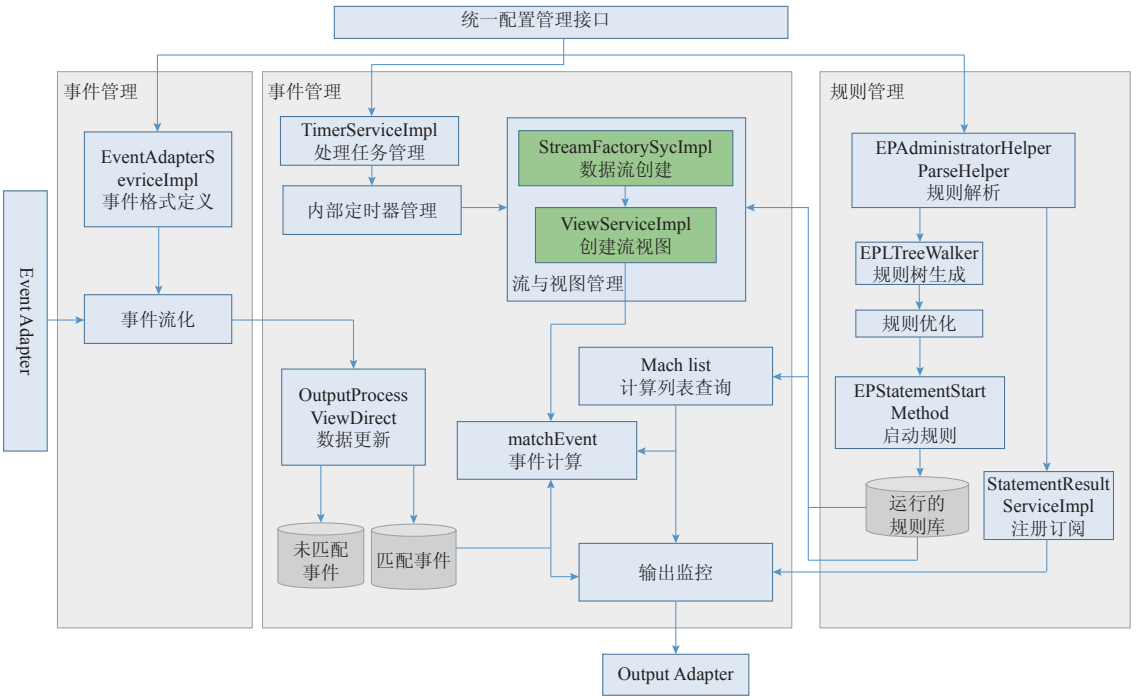


图10-22　基于查询的CEP技术框架

```
select sum(price) from StockTickEvent(symbol='GE')win:length(5)
```

图10-23　代码样例

件—规则—行动”、常用于BPM，以及可作为规则引擎、决策引擎。其技术框架如图10-24所示。

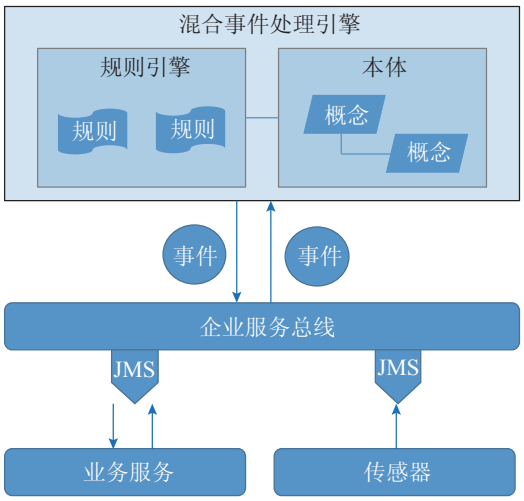


图10-24　基于规则的CEP技术框架

CEP技术发展有十几年的历史，近年借助大数据有升温趋势，特别是在金融证券、网络安全、物流、交通、物联网领域应用。CEP技术分类有Query-Based、Rule-Based、State-Based等，主流是Query-Based，也称为ESP(Event Stream Processing)。CEP在尝试标准化流、窗口、关系等概念已写入SQL99规范，CQL(Continuous Query Language)将成为未来标准。但CEP未发展成SQL一样严谨、完备的理论体系，各家实现标准不一，开放性差。对于复杂业务，基于CEP进行开发很困难。而且CPE的Scale-out扩展能力也很受限制。

10.3.4　流处理技术的最新发展

在流处理技术领域，主要有两类技术趋势。一类是基于MapReduce框架进行流处理技术改进，以Spark Streaming为代表，通过微批(Micro-batch)的方式模拟流处理，降低响应时延至秒级；另一类是基于真实的事件处理，通过架构和引擎的改进，降低时延并提升吞吐，典型代表为Storm和Flink。

Spark Streaming是基于Spark实现的流处理，实现秒级延迟，支持流、窗口、各种聚合操作，提供Exactly-once的容错保障，支持高级API与流式SQL查询。其具体实现是通过将实时流划分为一系列微批(例如，每1秒数据为1个微批)，每个微批被视为一个RDD，Spark对RDD实施流操作，结果也是分批次输出的(见图10-25)。

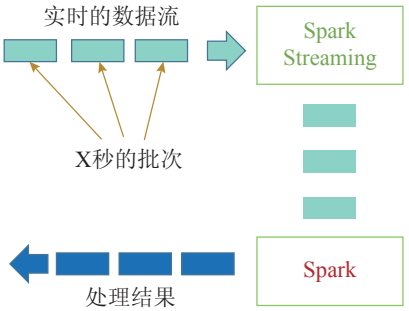


图10-25　Spark Streaming框架

Spark Streaming在处理流任务时，可以提供相当高的吞吐；其另外一个显著优势是可以将实时工作负载与批次工作负载融合于一个统一的处理平台之中，降低平台复杂性，被广泛应用于Lambda架构中(比如Cloudera Oryx)。然而，由于处理单元为微批，而非单条记录，Spark Streaming仅能达到秒级的时延，较真正流处理的毫秒级有一定差距；同时，在功能性方面，微批也为窗口操作带来了最小窗口尺寸等限制。

Storm是由Twitter主导开发的分布式实时计算系统，并将其开源，目前拥护者众多，版本演进迅速(见图10-26)。

Storm具有分布式处理框架，由中心管理节点采用多种形式的事件路由分发机制。工作任务由事件驱动(Event-Driven)。Storm具有DAG图形式Topology，并有机制保障事件处理的可靠性。

Flink是一个分布式的、可扩展的数据处理平台，基于其核心的流计算框架，Flink提供低时延、高吞吐的实时流处理能力。与Storm相同，Flink引擎也是由事件驱动的，并以管线方式优化并发运行流任务拓扑。比Storm更进一步的是，Flink将批次任务抽象为流任务的一个特例，因此也能完全支持批处理任务，使其成为一个流批混合型处理平台。

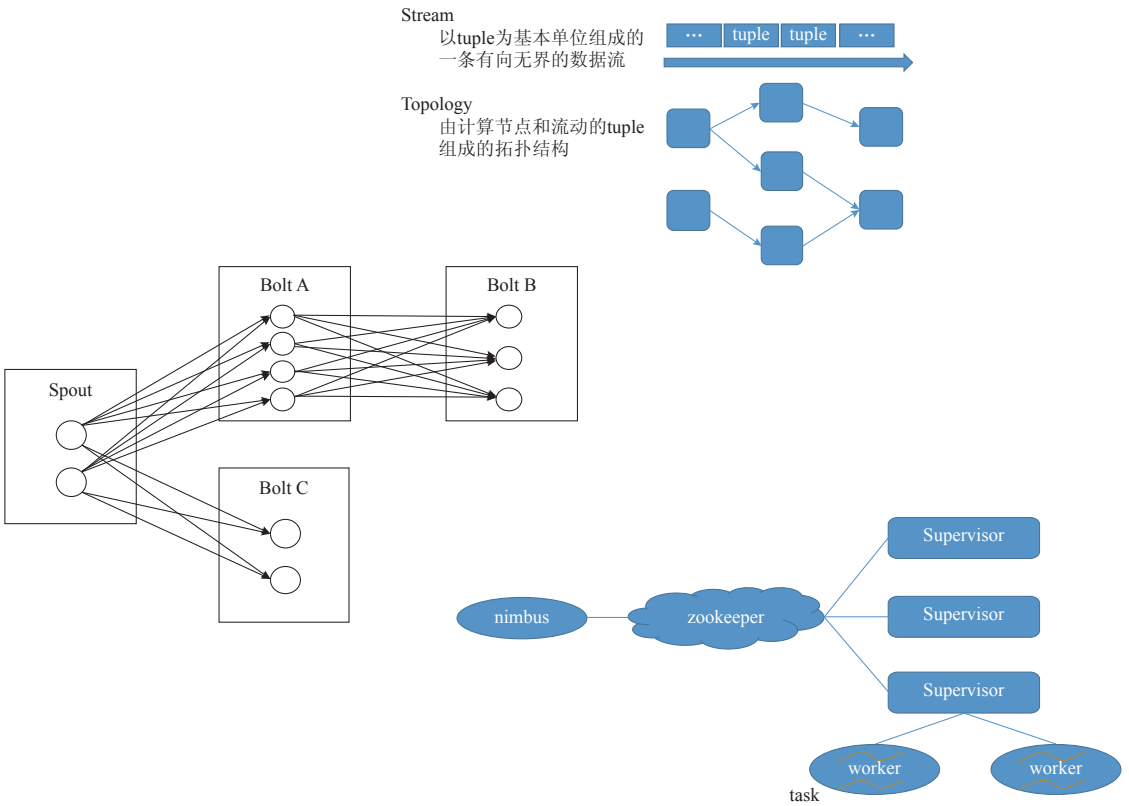


图10-26 Storm流处理框架

和Spark类似，Flink提供高级的、易用的API，涵盖批处理、实时流处理和关系查询等场景。Flink也提供了特定领域的库，如机器学习库FlinkML，图算法库Gelly，以及复杂事件处理CEP库(见图10-27)。

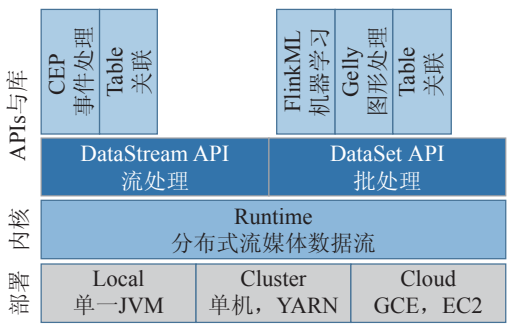


图10-27 Flink软件栈

在流处理方面，Flink主要具有以下特点。

低时延，高吞吐：得益于先进的架构和Pipelined处理引擎，Flink可以达到毫秒级的时延，以及百万级别的吞吐。

轻量级分布式快照技术，提供exactly-once的容错性保障：Flink的容错性机制提供最高级别的容错性，exactly-once，即数据不丢失、不重算；另一方面，由于快照轻量可控，开启容错机制对吞吐性能的影响较小。

乱序事件处理：借鉴自Google Dataflow的Watermark概念，Flink实现了较先进的时间机制，从而可处理乱序事件、延迟事件等在真实网络场景下常见的问题。

高灵活度定制窗口：Flink提供多种内置的窗口API，可以基于时间、事件个数、会话来划分时间；同时，用户可以利用高灵活度、高可扩展性的API，高度定制窗口的划分方式、运算

的触发机制等。

✍ 复杂事件处理：Flink提供CEP库，使得复杂事件处理与流处理无缝地融合于同一个平台。

当前，随着大数据发展的日益加剧，大数据处理技术面临着更大的挑战。一方面是大数据的4V特性都在显著增长，数据更快、更大、更多样，不可能像传统的MapReduce那样将数据先存储下来，然后进行处理和分析；另一方面，企业对大数据处理有了更高诉求，要求更快、更精准地捕获数据价值。高性能的流处理将是解决这些问题的关键之一，在大数据处理中将扮演越来越重要的角色。以Google的Dataflow、Flink为代表数据处理平台，将流批统一起来，并在其上提供数据处理、高级分析、关系查询等能力，将是接下来大数据的重要方向之一。

10.4 大数据在金融领域的探索与实践

大数据在信息时代占据很高的地位，分析大数据对于众多行业都具有很重要的战略意义。大数据技术的主要任务是从内部和外部数据源中快速地发现商业机会并挖掘其价值，还对这些数据进行高效快捷的评估，最终提供决策支撑。大数据有利于从各类数据中快速获得信息，物联网的快速发展也为大数据提供了广泛的数据来源。智能手机的普及使数据量呈指数级增长。廉价的存储和高速的带宽也为大数据的诞生提供了必备条件。云计算为大数据的诞生提供了物质基础。目前，数据量的增长在大数据应用中处于主导性的地位，从TB级升至PB级，并且仍在持续爆炸式增长，其增长速度远超摩尔定律增长速度。大数据的分析需求由抽样分析转向全量分析，成为企业发展必不可少的支撑点。由于数据量不断迅速增加，数据规模也随之增大，导致成本上升。从成本角度考虑，大数据的硬件平台由专用硬件服务器转向标准开放硬件构成的大规模机群平台。大数据遍布在许多领域，物联网、云计算、移动互联网、车联网、手机以及各式传感器，无一不是数据来源或者承载的方式。全球对大数据技术和服务的投资在不断增长，依靠大数据可以提供足够有利的资源。

大数据之于金融，其根本驱动是降低资本融通的成本、提高资金服务的效率(见图10-28)。

10.4.1 银行业现状和大数据的潜在机会

由于金融服务的业务转型的影响，银行业服务及管理模式都将发生根本性的改变。统计显示，以ATM、网上银行、手机银行为代表的电子银行在我国当前已经逐渐成为重要交易渠道，对传统银行渠道的替代率超过了60%。特别是互

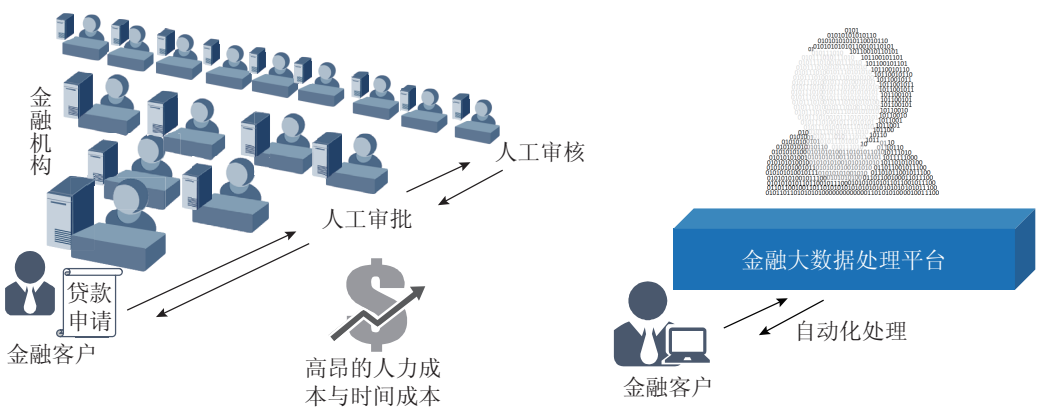


图10-28 大数据在金融行业

联网金融可能会对银行的观念和经营模式加以颠覆，银行业应如何主动变革、变挑战为机遇是一个值得探讨和深刻思考的问题。银行是经营信用的企业，数据的力量尤为关键和重要。在“大数据”时代，以互联网为代表的现代信息科技，特别是门户网站、社区论坛、微博、微信等新型传播方式的蓬勃发展，移动支付、搜索引擎和云计算的广泛应用，构建起了全新的数字化客户信息体系，并将改变现代金融运营模式。数据海量化、多样化、传输快速化和价值化等特征，将给商业银行市场竞争带来全新的挑战和机会。

中国银行业现阶段的大数据应用需求大致可以分为以下四类。

- 客户分析：基于各种数据源的客户数据和客户行为数据分析，用于客户分类、客户差异化服务、客户推荐系统、客户流失预测等。
- 风险分析：基于银行交易和客户交互数据进行建模，借助大数据平台快速分析和预测在此发生或者新的市场风险、操作风险等。
- 运营分析：基于企业内外部运营、管理和交互数据分析，借助大数据平台，三百六十度统计和预测企业经营和管理绩效。
- 行业监管：基于企业内外部交易和历史数据，实时或准实时预测和分析欺诈、洗钱等非法行为，遵从法规和监管要求。

10.4.2 大数据时代的银行业发展

大数据的高速发展，使银行业的客户数据、交易数据、管理数据等均呈现爆炸式增长，海量数据席卷而来，机遇和挑战也随之而来，为银行创造变革性价值创造了条件，银行业服务及管理模式都将发生根本性改变。大数据在银行业的应用范围包括客户定价、产品营销、风险管理等(见图10-29)。

一、机遇

大数据时代的银行业发展面临以下机遇。

1. 业务发展空间

我国商业银行所提供的业务服务和产品都具有很强的同质性，但是竞争关系要求银行实施差异化战略。互联网的兴起使得社交媒体成为银行新的接触客户渠道，银行可以从各个网点、PC、移动终端、传感器网络等端口收到结构化、非结构化的海量数据，可深入挖掘客户，强化交叉销售，同时加快产品的创新空间。

2. 决策判断能力

在信息时代，人类社会面临的中心问题将从

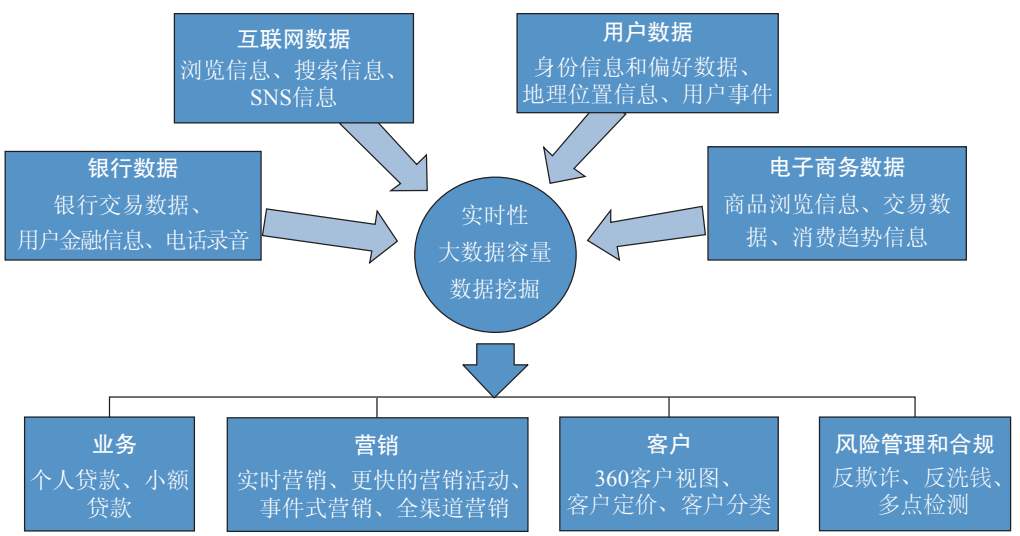


图10-29 中国银行业的大数据应用场景

如何提高生产率转变为如何更好地利用信息来辅助决策。对于银行而言，“大数据”将使银行决策从“经验依赖”向“数据依据”转化，将在深入了解和把握银行自身乃至市场状况的基础上，更加科学地评价经营业绩、评估业务风险、配置全行资源，引导银行业务科学健康发展。

3. 经营管理能力

大数据将掀起银行业的精细化管理革命和竞争。关于资产、负债、客户、交易对手及业务过程中产生的各种数据资产，在风险控制、成本核算、资本管理、绩效考核等方面发挥着重要作用，充分利用数据分析技术将是银行制胜的关键。“数据—信息—商业智能”将逐步成为银行定量化、精细化管理的发展路线，为有效提升服务能力提供强大支撑。

二、挑战

大数据时代的银行发展面临以下挑战。

1. 数据驾驭能力

“大数据”时代首先对银行的数据驾驭能力提出了全新的挑战。在数据收集方面，银行不仅要收集来自网点、信贷等传统渠道的结构化数据，还要收集来自物联网、互联网、机构系统的各类非结构化数据，甚至还要与历史数据对照，非结构化数据收集模式将彻底颠覆银行数据收集理念。在数据存储方面，要达到低成本、低能耗、高可靠性目标，通常要用到冗余配置、分布化和云计算技术，这正是银行所欠缺的。在数据处理方面，有的数据涉及上百个参数，难以用传统的方法描述与度量，处理的复杂度相当大，如客服录音数据等。利用“大数据”的能力将成为决定银行竞争力的关键因素。

2. 生存发展能力

银行的生存发展能力受到挑战。大量的数据来源和强大的数据分析工具正催生出很多新的金融业态来直接瓜分银行的信贷市场。与传统银行相比，互联网金融在信息收集、信息处理、产品交付以及风险防范等方面都有优势，其提供的金融服务已经从简单支付渗透到了转账汇款、小额信贷、现金管理、资产管理、供应链金融、基金和保险代销等银行核心业务领域。

3. 商业运营模式

随着数据化和网络化的全面深入发展，金融服务虚拟化将成为大势所趋。一是产品虚拟化，金融IC卡的推广应用正在逐步提升银行的电子化发展进度，银行资金将越来越多地呈现为各类数据信号的交换，电子货币将与实物货币并驾齐驱。二是服务虚拟化，“善融商务”、“交博汇”以及网络金融商城等银行电子商务平台不断发展，鼠标银行、电子银行成为未来趋势。三是管理虚拟化，银行业务中的各种单据、凭证等将以数字文件的形式出现，网络成为重要的管理通道，电子化、数据化的管理模式更加方便快捷。传统的商业银行运营模式将逐渐消融在数据化的洪流里，借助“大数据”手段，实现跨越发展，成为未来商业银行可持续发展的唯一选择。

10.4.3 大数据在银行业的发展趋势

目前，大数据相关的技术和工具非常多，给企业提供了更多的选择。在未来，其还会继续出现新的技术和工具，如Hadoop、下一代数据仓库等，这也是大数据领域的创新热点。企业越来越希望能将自己的各类应用程序及基础设施转移到云平台上。就像其他IT系统那样，大数据的分析工具和数据库也将走向云计算。首先云计算为大数据提供了可以弹性扩展、相对便宜的存储空间和计算资源，使得中小企业可以像跨国大企业一样通过云计算来完成大数据分析。其次，云计算IT资源庞大、分布较为广泛，是异构系统较多的企业及时准确处理数据的有力方式，甚至是唯一的方式。随着数据分析集的扩大，以前部门层级的数据集市将不能满足大数据分析的需求，它们将成为企业级数据库的一个子集。

对于银行业来说，“大数据”革命必将颠覆银行传统观念和经营模式。要强化“数据治行”理念，建立分析数据的习惯，重视“大数据”开发利用，提升全行的质量管理、数据管理。同时，其要营造“数据治行”文化，倡导用数据说话，准确描述事实，反映逻辑理性，将现有数据

转化为信息资源，为高层管理和决策提供强有力的依据，让决策更加针对目标，让发展更加贴近真实市场。其着眼于“大数据”挖掘和分析，对海量数据的持续实时处理，建设数据仓库项目，为服务质量改善、经营效率提升、服务模式创新提供支撑，全面提升运营管理水平。在项目建设中，其通过梳理、整合经营管理关键数据，建立数据管控体系，搭建基础数据平台。通过数据仓库建设，运用数据挖掘和分析，全方位调整管理模式、产品结构、营销模式、信息战略，从根本上提高风险管理、成本绩效管理、资产负债管理和客户关系管理水平，实现多系统数据的业务逻辑整合，形成全行级客户、产品、协议等主题数据。其积极推动传统业务渠道与移动通信、云计算等新兴业态纵向整合、横向渗透，促进信息集中、整合、共享、挖掘。一方面，要“走出去”，与移动网络、电子商务、社交网络等“大数据平台”完美融合，开展“大数据”分析，为客户提供开放服务平台。另一方面，要“请进来”，与数据分析专业厂商合作，对数据存量进行综合处理与分析。建立完善内容涵盖全面、功能丰富齐全，集网上贸易服务、网上保理、电子商业汇票、票据池、应收账款池融资、在线融资等为一体的综合供应链金融服务体系，为客户提供触手可及的全方位贴身服务。

10.4.4 大数据在金融行业的实践

一、问题和需求

随着互联网金融的快速兴起，银行业务竞争激烈，A金融机构急需以金融大数据查询、分析、挖掘为基础，进行产品创新、预测和风险评估，改善服务质量，提升竞争力。但A金融机构的IT数据处理平台架构还是以传统数据库与数据仓库为主的模式(见图10-30)。

传统面向结构化数据的数据平台已经无法满足大数据数量、种类的快速增长，无法支撑A金融机构未来对数据处理的要求。具体表现在以下几个方面。

- 系统无法支撑在线查询5年历史明细：业务需要在线查询5年15TB的交易历史明细，但现状是容量受限，数据库仅能在线查询1年3TB的交易历史明细，1年以上历史明细需要数据仓库离线人工查询，客户等待时间长。
- 系统无法支撑全量多维客户行为分析：业务需要全量多维度分析，改进分析机制，提高分析结果转化率，但现状是受限于处理能力制约，ETL抽取到传统数据库的数据维度较少，且现有SAS分析工具采用专家经验机制，导致分析结果转化率低。
- 系统无法支撑实时征信：征信数据分散在业务数据库、数据仓库，信用卡开户、透支服务征信数据需要分时分散查询，且流程人工干预多，导致业务办理需要3周左右，无法实现实时征信数据查询客户满意度低。
- 数据库问题：原有数据系统仅适合实时交易类业务，不适合分析类业务，存储结构化数据，容量有限，无法存储过多的历史数据。
- 数据仓库问题：原有数据仓库系统仅适

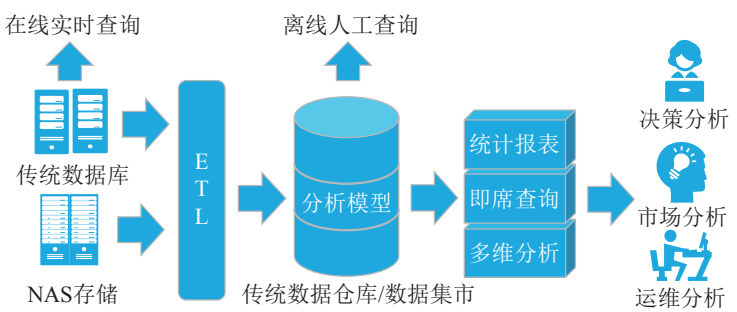


图10-30 传统数据库+数据仓库为主的银行数据处理平台

合结构化数据分析类业务，非结构化数据需前端ETL工具抽取，转换成结构化数据存储，但数据维度减少，无法线性扩展，数据量大处理缓慢，银行一般多业务分时复用使用，无法多业务并发使用。

基于以上诸多挑战和问题，A金融机构寻求采用大数据技术来完善金融数据处理解决方案，来应对日益突出的业务问题。在大数据解决方案选择上，A金融机构对大数据平台有三个核心的要求。

- 安全性：满足金融等要求；
- 可靠性：所有组件支持HA，支持容灾；
- 易用性：与应用无缝衔接，适合现有编程习惯。

二、架构方案及优点

A金融机构对多家知名IT厂商的大数据解决方案进行了详细的考察和测试验证，搭建了一个全新的金融大数据处理平台，其架构如图10-31所示。

在这个架构中，A金融机构将数据分为第一数据平面和第二数据平面。第一数据平面主要基于原有的金融IT平台，以交易为核心，支撑传统的金融数据处理与分析业务。第二数据平面则是以大数据平台为核心的新建数据平面，主要处理金融数据分析业务，实现诸如社交情绪指数、或有金融资产、金融脉络关系、在线征信、精准推荐、在线历史明细等业务。第一平面与第二平面的数据实时共享与交互，实现相互间业务支撑。

该架构除了实现基于Hadoop的基础大数据分析能力外，还实现了A金融机构所需的安全、可靠、易用三大特性。

- 高安全性：业界第一家支持金融等级保护、第一家支持RBAC用户组权限管理和消除HDFS明文存放隐患的大数据平台。
- 高可靠性：大数据平台全组件支持HA，业界第一个通过1000+KM异地容灾验证的厂家。
- 易用性：丰富的行业全量建模和二次开发能力，让大数据与客户应用无缝衔接，全自动化在线运维、自动化的应用开发助手，轻松管理大数据系统。

在线历史明细查询解决方案中，由IT供应商提供完整的大数据分布式业务平台和Hadoop大数据平台解决方案，A金融机构的技术团队只需专注历史明细查询业务的编写。分布式业务平台支持多业务系统并发访问，实现实时历史明细查询能力。方案同时支持Socket、Web业务请求接入和分发，与A金融机构业务系统无缝衔接。创新的CTBase方案、独有的表聚簇和多级索引支持HBase多表关联查询的能力。HBase同时支持SQL、Java API编程接口，适应客户的编程习惯。

在全量多维客户行为分析解决方案中，由IT供应商提供完整的数据挖掘工具大数据洞察平

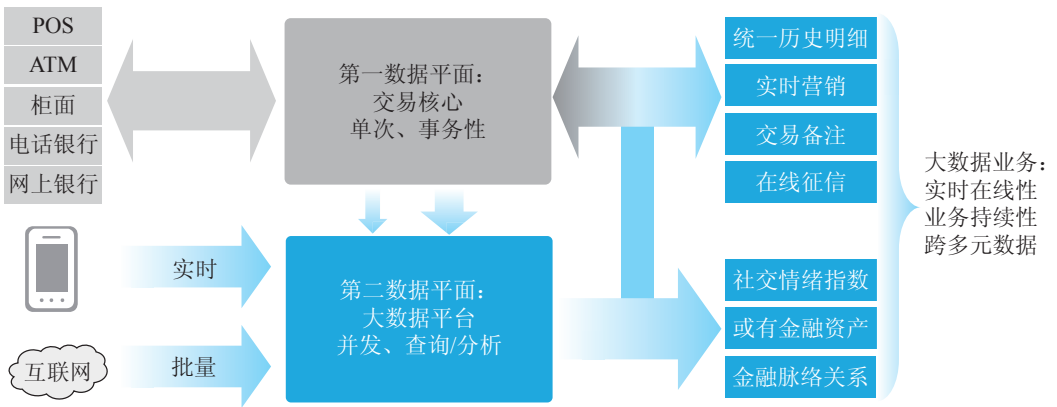


图10-31　A金融机构全新的金融数据架构

台和Hadoop大数据平台解决方案，A金融机构技术团队只需专注客户行为分析业务的编写。IT供应商同时提供了金融通用的客户行为分析业务，即用户特征刻画、小微贷倾向分析。大数据洞察平台基于大数据全量建模分析，可以挖掘出14 000位客户特征，实现多维并发分析。方案采用Hadoop机器自动学习机制，大大提高分析准确度。客户行为分析结果存储在HBase，供业务查询使用。

在实时征信解决方案中，IT供应商提供完整的大数据分布式业务平台(DAP)和Hadoop大数据平台解决方案，A金融机构技术团队只需专注实时征信业务组件的编写。分布式业务平台提供征信流程业务系统并发处理能力。分布式业务平台提供实时工作流引擎，可以根据预先设置的顺序自动化执行征信的每一个业务流程。HBase中存储所有与实时征信相关的表，包括资料信息表、内部资信表、客户公共profile表、风险信息表、征信主表，提供征信数据实时查询能力。HBase支持SQL调用接口，与实时征信已有的业务组件无缝衔接。

三、系统收益

A金融机构建设的第二数据平面大数据平台为A金融机构带来非常显著的价值收益，具体表现在以下几个方面(见图10-32)。

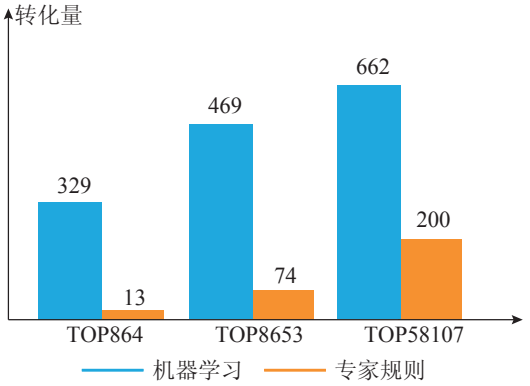


图10-32　小微贷大数据与传统数据仓库分析效果对比

- 历史明细查询：实现统一集中存储5年15TB交易历史明细数据，便于管理和扩展，多业务系统并发实时查询5年交易历史明细数据，提升金融最终用户的使用体验。
- 实时征信：实现了2～5秒自动化征信，提升金融信贷客户的满意度；实时工作流引擎简化客户部署实时自动化征信业务的难度。
- 客户行为分析：小微贷倾向分析实现TOP10 000客户推荐成功转化率，相比传统数据仓库，四维度分析模式转化率将提高6倍。

10.5　未来大数据应用畅想

10.5.1　身边的大数据

似乎一夜之间，大数据(Big Data)变成一个IT行业中最时髦的词汇。

首先，大数据不是什么完完全全的新生事物，Google的搜索服务就是一个典型的大数据运用，根据客户的需求，Google实时从全球海量的数字资产(或数字垃圾)中快速找出最可能的答案，并呈现给你，这就是一个最典型的大数据服务。只不过过去这样规模的数据量处理和有商业价值的应用太少，在IT行业没有形成成型的概念。现在随着全球数字化、网络宽带化、互联网应用于各行各业，累积的数据量越来越大，越来越多企业、行业和国家发现，可以利用类似的技术更好地服务客户、发现新商业机会、扩大新市场以及提升效率，从而逐步形成大数据这个概念。

从运营商行业的一个真实故事讲起：某运营商客户有一个困扰，即一些手机厂商和客户内部的某些人内外勾结，采用俗称“洗码”的方式来非法获利，具体来说，就是这些手机厂商的部分手机进入运营商渠道销售、加上资费包出库后，并没有进入最终消费者手中，而是进入相关利益者手中，相关利益者把手机资费拆包卖掉后，手机又通过渠道重新进入运营商的销售环节。对于运营商来说，这样没有发展新用户，只是循环徒增成本。单纯从运营商的流程和业务系统是找不出问题在哪里的。后来，客户技术团队利用技术手段从网络、运营系统等抓取了很多的数据，对

各品牌、各款式、各批次手机建模分析，发现一系列线索，找到了问题点和相关责任人，运营商通过处罚及威慑基本杜绝了这种情况的发生。这件事当时做的时候，大数据概念还没有变得很热，但毫无疑问，这样的做法就是大数据方法的有效实践。

另外一个有趣的故事是关于奢侈品营销的。PRADA在纽约的旗舰店中每件衣服上都有RFID码。每当一个顾客拿起一件PRADA进试衣间，RFID会被自动识别。同时，数据会传至PRADA总部。每一件衣服在哪个城市哪个旗舰店什么时间被拿进试衣间停留多长时间，数据都被存储起来加以分析。如果有一件衣服销量很低，以往的做法是直接下架。但如果RFID传回的数据显示这件衣服虽然销量低，但进试衣间的次数多，那就能另外说明一些问题。也许这件衣服的下场就会截然不同，也许在某个细节的微小改变就会重新创造出一件非常流行的产品。

还有一个是关于中国粮食统计的故事。中国的粮食统计是一个老大难的问题。中国的统计，虽然有组织、有流程、有法律，但中央的统计人员依靠省统计人员，省靠市，市靠县，县靠镇，镇靠村，最后真正干活或上报的是基层兼职的调查人员。在前两年北京的一个会议上，原国家统计局总经济师姚景源讲述了他们是如何提升数据的精准性的。他们采用遥感卫星，通过图像识别，把中国所有的耕地标识计算出来，然后把中国的耕地网格化，对每个网格的耕地抽样进行跟踪、调查和统计，然后按照统计学的原理，计算(或者说估算)出中国整体的粮食数据。这种做法是典型采用大数据建模的方法，打破传统流程和组织，直接获得最终的结果，或者获得不同渠道来源的数据，对同一个事实从不同侧面加以验证。

最后是一个炒股的故事。这个故事来自2011年好莱坞的一部高智商电影《永无止境》，讲述一位落魄的作家库珀，服用了一种可以迅速提升智力的神奇蓝色药物，然后他将这种高智商用于炒股。库珀是怎样炒股的呢？他能在短时间掌握无数公司的资料和背景，也就是将世界上已经存在的海量数据(包括公司财报、电视、几十年前的报纸、互联网、小道消息等)挖掘出来，串联起来，通过海量信息的挖掘、分析，使一切内幕都不是内幕，使一切趋势都在眼前，结果在10天内他就赢得了200万美元，神奇的表现让身边的职业投资者目瞪口呆。这部电影展现了大数据魔力，我们推荐没有看过的IT人士看一看。

从这些案例来看，大数据并不是很神奇的事情。就如同电影《永无止境》提出的问题：人类通常只使用了20%的大脑，如果剩余80%的大脑潜能被激发出来，世界会变得怎样？在企业、行业和国家的管理中，通常只有效使用了不到20%的数据(甚至更少)，如果剩余80%的数据价值激发起来，世界会变得怎么样呢？特别是随着数据爆发式增长，并且得到更有效应用，世界会怎么样呢？

单个数据并没有价值，但越来越多的数据累加，量变就会引起质变，就好像一个人的意见并不重要，但1千人、1万人的意见就比较重要，上百万人就足以掀起巨大的波澜，上亿人足以改变一切。

数据未被整合和挖掘，其价值无法呈现出来。《永无止境》中的库珀如果不能把海量信息围绕某个公司的股价整合起来、串联起来，这些信息就没有价值。

因此，海量数据的产生、获取、挖掘及整合，使之展现出巨大的商业价值，数据驱动经营，这就是我们所理解的大数据。在互联网对一切重构的今天，这些问题都不是问题。因为，大数据是互联网深入发展的下一波应用，是互联网发展的自然延伸。目前，可以说大数据的发展到了一个临界点，才会成为IT行业中最热门的词汇之一。

10.5.2 大数据将重构很多行业的商业思维和商业模式

我们以对未来汽车行业的狂野想象来感受大数据可能带来的变化。

在人的一生中，汽车是一项巨大的投资。以一部30万元的车、7年换车周期来算，每年折旧费4万多元(这里还不算资金成本)，加上停车、保险、油、维修、保养等各项费用，每年花费应在6万元左右。汽车产业也是一个拥有很长产业链的龙头产业，这个方面只有房地产可以媲美。

但同时，汽车产业链是一个低效率、变化慢的产业。汽车一直以来就是四个轮子、一个方向盘、两排沙发(李书福语)。这么一个昂贵的东西，围绕车产生的数据却少得可怜，行业产业链之间几乎无任何数据传递。

我们在这里狂野地想象一番，如果将汽车全面数字化，都大数据了，会产生什么结果？

有些人说，汽车数字化，不就是加个MBB模块吗？不，这太小儿科了。深入地来看，数字化意味着汽车可以随时连上互联网，意味着汽车是一个大型计算系统加上传统的轮子、方向盘和沙发，意味着可以数字化导航、自动驾驶，意味着你和汽车相关的每一个行动都数字化，包括每一次维修、每一次驾驶路线、每一次事故的录像、每一天汽车关键部件的状态，甚至你的每一个驾驶习惯(如每一次的刹车和加速)都记录在案。这样，你的车每月甚至每周都可能产生1TB以上的数据。

好了，我们假设这些数据都可以存储并分享给相关的政府、行业和企业。这里不讨论隐私问题带来的影响，假设在隐私保护的前提下，数据可以自由分享。

那么，保险公司会怎么做呢？保险公司把你的所有数据拿过去建模分析，发现几个重要的事实：一是你开车主要只是上下班，A地到B地这条线路是非繁华路线，红绿灯很少，这条路线过去一年统计的事故率很低；你的车况(车的使用年限、车型)好，此车型在全市也是车祸率发生较低的车型；甚至可以统计你的驾驶习惯，加油平均，临时刹车少，超车少，和周围车保持了应有的车距，驾驶习惯好。最后结论是，你的车型好，车况好，驾驶习惯好，常走的线路事故率低，过去一年也没有出过车祸，因此可以给予更大幅度的优惠折扣。这样保险公司就完全重构了它的商业模式了。在没有大数据支撑之前，保险公司只把车险客户做了简单的分类，一共分为四种客户，第一种是连续两年没有出车祸的，第二种是过去一年没有出车祸的，第三种是过去一年出了一次车祸的，第四种是过去一年出了两次及以上车祸的。这种简单粗略的分类，就好像女人找老公，仅把男人分为没有结过婚的、结过一次婚的、结过二次婚的、结过三次及以上婚的四种男人，就敢嫁人一样。在大数据的支持下，保险公司可以真正以客户为中心，把客户分为成千上万种，每个客户都有个性化的解决方案，这样保险公司的经营将完全不同，对于风险低的客户敢于大胆地给予折扣，对于风险高的客户报高价甚至拒绝。如此一来，一般的保险公司将难以和这样的保险公司竞争。拥有大数据并使用大数据的保险公司比传统公司将拥有压倒性的竞争优势，大数据将成为保险公司最核心的竞争力，因为保险就是一个基于概率评估的生意，对于准确评估概率，大数据毫无疑问是最有利的武器，而且简直是量身定做的武器。

在大数据的支持下，4S店的服务也将完全不同。车况信息会定期传递到4S店，4S店会根据情况及时提醒车主及时保养和维修，特别是对于可能危及安全的问题，在客户同意下甚至会采取远程干预措施，同时还可以提前备货，车主一到4S店就可以维修而不用等待。

对于驾驶者来说，不想开车的时候，在大数据和人工智能的支持下，车辆可以自动驾驶，并且对于你经常开的线路可以自学习、自优化。Google的自动驾驶汽车，为了对周围环境做出预测，每秒钟要收集差不多1GB的数据，没有大数据的支持，自动驾驶是不可想象的；在与周围车辆距离过近的时候，会及时提醒车主避让；上下班的时候，会根据实时大数据情况，对于你经常开车的线路予以提醒，绕开拥堵点，帮你选择最合适的线路；在出现紧急状况的时候，比如爆胎，自动驾驶系统将自动接管，提高安全性(人一辈子可能难以碰到一次爆胎，人在紧急时的反应

往往是灾难性的，只会更糟)；到城市中心，寻找车位是一件很麻烦的事情，但未来你可以到了商场门口后，让汽车自己去找停车位，等想要回程的时候，提前通知汽车，让汽车自己开过来接你。

车辆是城市最大、最活跃的移动物体，是拥堵的来源，也是最大的污染来源之一。数字化的车辆、大数据应用将带来很多的改变。红绿灯可以自动优化，根据不同道路的拥堵情况自动进行调整，甚至在很多地方可以取消红绿灯；城市停车场也可以大幅度优化，根据大数据的情况优化城市停车位的设计，如果配合车辆的自动驾驶功能，停车场可以进行革命性的演变，设计专门为自动驾驶车辆服务的停车楼，地下、地上楼层可以高达几十层，停车楼层可以更矮，只要高于车高度即可(或者把车竖起来停)，这样将会对城市规划产生巨大的影响；在出现紧急情况时，如前方塌方的时候，可以第一时间通知周围车辆(尤其是开往塌方道路的车辆)；现在的燃油税也可以发生革命性变化，可以真正根据车辆的行驶路程，甚至根据汽车的排污量来收费，排污量少的车甚至可以搞碳交易，将排放量卖给高油耗的车；政府还可以每年公布各类车型的实际排污量、税款、安全性等指标，鼓励民众买更节能、更安全的车。

电子商务和快递业也可能发生巨大的变化。运快递的车可以自动驾驶，不用在白天行驶在拥堵的道路，而在晚上行驶。你可以在家门口设计自动接收箱，通过开启密码自动投递进去，就好像过去报童投报一样。

这么想象下来，我们看到，汽车数字化、互联网化、大数据应用、人工智能等将对汽车业及相关的长产业链产生难以想象的巨大变化和产业革命，具有无限的想象空间，可能完全被重构。当然，要实现我们所描述的场景，还有很长的路要走。

下面，我们总结一下大数据将会带来的改变。

第一，大数据使企业真正有能力从以自我为中心改变为以客户为中心。企业是为客户而生的，目的是为股东获得利润。只有服务好客户，才能获得利润。但过去，很多企业是没有能力做到以客户为中心的，原因就是相应客户的信息量不大，挖掘不够，系统也不支持，目前的保险业就是一个典型。大数据的使用能够使对企业的经营对象从客户的粗略归纳(提炼归纳的“客户群”)还原成一个个活生生的客户，这样经营就有针对性，对客户的服务就更好，投资效率就更高。

第二，大数据在一定程度上将颠覆了企业的传统管理方式。现代企业的管理方式来源于对军队的模仿，依赖于层层级级的组织和严格的流程，依赖信息的层层汇集、收敛来制定正确的决策，再通过决策在组织的传递与分解以及流程的规范，确保决策得到贯彻，确保每一次经营活动都有质量保证，也确保一定程度上对风险的规避。过去这是一种有用而笨拙的方式。在大数据时代，我们可能重构企业的管理方式，通过大数据的分析与挖掘，大量的业务本身就可以自决策，不必要依靠膨大的组织和复杂的流程。大家都是基于大数据来决策，都是依赖于既定的规则来决策，是高高在上的首席执行官决策，还是一线人员决策，本身并无大的区别，那么企业是否还需要如此多层级的组织和复杂的流程呢？

第三，大数据另外一个重大的作用是改变了商业逻辑，提供了从其他视角直达答案的可能性。现在人的思考或者企业的决策，事实上都是逻辑的力量在主导起作用。我们去调研、收集数据、进行归纳总结，最后形成自己的推断和决策意见，这是一个观察、思考、推理、决策的商业逻辑过程。人和组织的逻辑形成需要大量的学习、培训与实践，代价是非常巨大的。但这是否是唯一的道路呢？大数据给了我们其他的选择，就是利用数据的力量，直接获得答案。就好像我们学习数学，小时候学九九乘法表，中学学几何，大学学微积分，碰到一道难题，我们是利用了多年学习沉淀的经验来努力求解，但我们还有一种方法，在网上直接搜索是不是有这样的题目，如果有，直接抄答案就好了。很多人就会批

评说，这是抄袭，是作弊。但我们为什么要学习啊？不就是为了解决问题嘛。如果我任何时候都可以搜索到答案，可以用最省力的方法找到最佳答案，这样的搜索难道不可以是一条光明大道吗？换句话说，为了得到“是什么”，我们不一定要理解“为什么”。我们不是否定逻辑的力量，但是至少我们有一种新的巨大力量可以依赖，这就是未来大数据的力量。

第四，通过大数据，我们将以全新的视角来发现新的商业机会和重构新的商业模式。我们现在看这个世界，比如分析家中食品腐败，主要就是依赖于我们的眼睛再加上我们的经验，但如果我们有一台显微镜，我们一下就看到坏细菌，那么分析起来就完全不一样了。大数据就是我们的显微镜，它可以让我们从全新视角来发现新的商业机会，并可能重构商业模型。我们的产品设计可能不一样了，很多事情不用猜了，客户的习惯和偏好一目了然，我们的设计能轻易命中客户的心窝；我们的营销也完全不同了，我们知道客户喜欢什么、讨厌什么，更有针对性。特别是显微镜再加上广角镜，我们将有更多全新的视野了。这个广角镜就是跨行业的数据流动，使我们过去看不到的东西都能看到了，比如前面所述的汽车案例，开车是开车，保险是保险，本来不相关，但当我们把开车的大数据传递到保险公司以后，整个保险公司的商业模式就完全重构了。

第 11 章

企业桌面云接入的关键技术架构与应用

我们知道，一个发电厂建设好后，若每家每户使用电的话，还需要架设长途电缆、建设变电站、在居民楼设置变压器、在每家每户铺设电线、安置插座和开关，设置电表，这样才能够将电力传送到千家万户。同样地，第4章、第5章和第6章分别讲述了计算虚拟化、网络虚拟化和存储虚拟化，利用这些技术，我们已经将分布在不同地方的数据中心进行虚拟化，构筑一个可以提供云服务的、很大的IT资源池，就是说，“IT电厂”已经建成。但是，对于最终的用户，要获得并使用IT资源，就要解决如何高效访问云计算系统以获取所需的IT资源，并获得满意的访问体验的问题，这就是云接入。

11.1 桌面云接入概述

11.1.1 什么是桌面云接入

云接入是指从单一平台实现桌面和应用虚拟化，提供固定和移动终端融合接入的统一工作空间，帮助客户对固定办公和移动办公环境下的桌面、应用和数据进行统一管理、发布和聚合。它是一种基于云计算的终端用户计算模式。在这种模式中，所有的应用程序都在云数据中心运行，应用程序无须在终端上安装。用户通过终端云接入协议连接到云数据中心，并运行在云数据中心的程序，以获取程序运行结果。

随着企业信息化进程的不断深入，企业中增加了各种各样的电子设备。由于传统IT的束缚，企业IT团队依然要维护大量的传统PC，这不仅需要大量的人力物力，而且在进行外网接入以及异地登录的时候无法很好地保障企业数据安全。全球可连接互联网设备的出货情况显示，PC所占份额越来越小。2015年智能手机出货量接近12.93亿部。据调查，美国人每天在智能手机上花1个小时，每天在平板电脑上花半个小时。移动带来了娱乐、通信、媒体和商务新方式。企业IT基础架构要不断适应这种新变化。

云接入很好地解决了这些问题，不仅可以快速地搭建企业IT基础架构，还可以快速对员工账户进行管理，实现跨平台作业。

桌面云是对云接入桌面这一侧重点的专门阐述。

11.1.2 云接入的作用和意义

云接入的业务价值很多，除了上面所提到的随时随地访问桌面以外，还有下面一些重要的业务价值。

一、数据上移，信息安全

传统桌面环境下，由于用户数据都保存在本地PC，因此内部泄密途径众多，且容易受到各种网络攻击，从而导致数据丢失。云接入桌面环境下，终端与数据分离，本地终端只显示设备，无本地存储，所有的桌面数据都集中存储在企业数据中心，无须担心企业的智力资产泄露。除此之外，TC的认证接入、加密传输等安全机制，保证了云接入桌面系统的安全可靠。

二、高效维护，自动管控

传统桌面系统故障率高，据统计，平均每400台PC机就需要一名专职IT人员进行管理维护，且每台PC维护流程(故障申报->安排人员维护->故障定位->进行维护)需要2～4个小时。

在云接入桌面环境下，可实现资源自动管控，维护方便简单，节省IT投资。

- 维护效率提升：云接入桌面云不需要前端维护，强大的一键式维护工具让自助维护更加方便，提高企业运营效率。使用云接入桌面后，每位IT人员可管理超过2000台虚拟桌面，维护效率提高4倍以上。
- 资源自动管控：白天可自动监控资源负载情况，保证物理服务器负载均衡；夜间可根据虚拟机资源占用情况，关闭不使用的物理服务器，节能降耗。

三、应用上移，业务可靠

在传统桌面环境下，所有的业务和应用都在本地PC上进行处理，稳定性仅99.5%，年宕机

时间约21个小时。在云接入桌面方案中，所有的业务和应用都在数据中心进行处理，强大的机房保障系统能确保全局业务年度平均可用度达99.9%，充分保障业务的连续性。各类应用的稳定运行，有效降低了办公环境的管理维护成本。

四、无缝切换，移动办公

在传统桌面环境下，用户只能通过单一的专用设备访问其个性化桌面，这极大地限制了用户办公地的灵活性。采用云接入桌面，由于数据和桌面都集中运行和保存在数据中心，用户可以不中断应用运行，实现无缝切换办公地点。

五、降温去噪，绿色办公

节能、无噪的TC部署，有效地解决了密集办公环境的温度和噪音问题。TC让办公室噪音从50分贝降低到10分贝，办公环境变得更加安静。TC和液晶显示器的总功耗大约60W左右，终端低能耗可以有效地减少降温费用。

六、资源弹性，复用共享

- 资源弹性：在云接入桌面环境下，所有资源都集中在数据中心，可实现资源的集中管控，弹性调度。
- 资源利用率提高：资源的集中共享，提高了资源利用率。传统PC的CPU平均利用率为5%～20%，在云接入桌面环境下，云数据中心的CPU利用率可控制在60%左右，提升了整体资源利用率。

七、安装便捷，部署快速

云接入桌面解决方案具有安装便捷、部署快速的特点。到客户现场后，只需服务器上电，进行云接入桌面软件的向导式安装，接通网络并进行相关业务配置即可进行业务发放，大幅度提高了部署效率。

11.1.3 桌面云接入的挑战和需求

云接入的挑战和需求，主要集中在如何应用虚拟化技术为终端用户提供资源访问的便利性、安全性以及用户体验上，通过分析这些典型技术的特点，可以发现他们仍然存在如下一些难以解决的问题。

一、外设兼容性

在云接入桌面虚拟化项目中对外设的支持是非常普遍的，绝大多数虚拟桌面基础架构项目中都会遇到用户对外设的需求，但有时也非常棘手。众所周知，外设在云终端上接入，在后端做桌面识别，这就涉及将具有电器特性的硬件设备通过网络传输到后端的桌面中，并且设备本身的驱动是在前端还是后端，都需要桌面云厂家考虑，加上国内外设的多样性和不标准性，要在桌面云中支持具有多样性复杂性的外设，需要厂家有独特的外设支持技术。

二、视频体验

云接入桌面的计算和存储全都在数据中心，终端只负责键盘鼠标的I/O和显示的输出，此时云桌面的传输协议就显得尤为重要。普通办公桌面的传输没有什么问题，但是实际上用户可能有各种各样的业务需求，例如视频，这类业务在终端桌面就可能出现画面不流畅，终端画面出现马赛克，更别提播放三维动画了，尤其随着桌面互联网的发展，很多桌面虚拟化方案需要基于互联网部署，而给予互联网的传输效果更是大打折扣。这要求桌面云厂家在桌面传输协议上有独特的通道设计，通过不同的通道来处理不同的桌面显示，并且在带宽上能优化处理。

三、3D应用

虚拟化桌面固然有其诱人之处，但是目前主流的桌面虚拟化技术在3D图形设计方面很难满足客户的需求，这也使得传统虚拟化方案在制造行业、数字内容创作等行业遇到了难以逾越的瓶颈，再好的解决方案如果不能满足用户的实际需求也是空谈。

四、网络负载压力

局域网一般不会存在太大的问题，但是如果通过互联网就会出现很多技术难题，由于桌面虚拟化技术的实时性很强，如何降低这些传输压

力，是很重要的一环；虽然千兆以太网对数据中心来说是一项标准，但还没有广泛部署到桌面，目前还达不到虚拟化桌面对高带宽的要求。而且如果用户使用的网络出现问题，桌面虚拟化发布的应用程序不能运行，则会直接影响应用程序的使用，其对用户的影响也是无法估计的。

五、安全、部署效率和用户体验是移动办公的主要挑战

移动办公的主要挑战如图11-1所示。

六、云接入的关键需求

随着企业的发展，分支机构、办事处、连锁店等企业扩大经营造成员工分布广，需要一种便捷、灵活和具有跨地域性的办公方案，使员工无论身在何处，都能实现员工与员工之间、企业与业务伙伴之间的相互交流和沟通。各级政府机构服务观念在不断提高，也希望通过移动化的方式提高办公效率，降低管理成本，提升服务质量。

市场人员遍布全国各地，没有固定的办公场所，他们每次访问内部系统的终端和网络的地方都不一样，没有局域网环境，他们无法使用CRM来更新客户信息，也无法利用OA系统来实现办公自动化。

公司领导经常出差办公，在火车站、飞机场等地方随时需要查看、调用、审批内部的资料文档，并知道业务进展及生产线的进度，需要随时随地都能访问内部办公系统及生产管理系统的解决方案。

政府工作人员驻点调查路况信息、各分局等的数据采集等，需要有一种方式可以将采集到的重要信息及时传达给内部系统。

突发和意外情况，能在事件发生的最短时间内上报、传达给企业内部的相关人员，相关人员和领导层能不受地点的限制，快速、及时地对突发和意外情况做出指示和决定。

随时随地办公，通过公网访问企业内部核心信息资源，就面临着非法访问、信息窃取等外部的安全威胁，就必须有相应的信息安全策略，在严格防止企业信息资源被非法窃取的同时，对合法的访问要提供方便。

移动性对后PC时代的成功至关重要。在IT组织希望满足终端用户对使用各种设备在家、旅途中和办公室的一致体验要求的同时，IT基础设施必须确保业务计算环境安全、易于管理并具备持续合规性。

云接入解决方案既要能提高终端用户的自

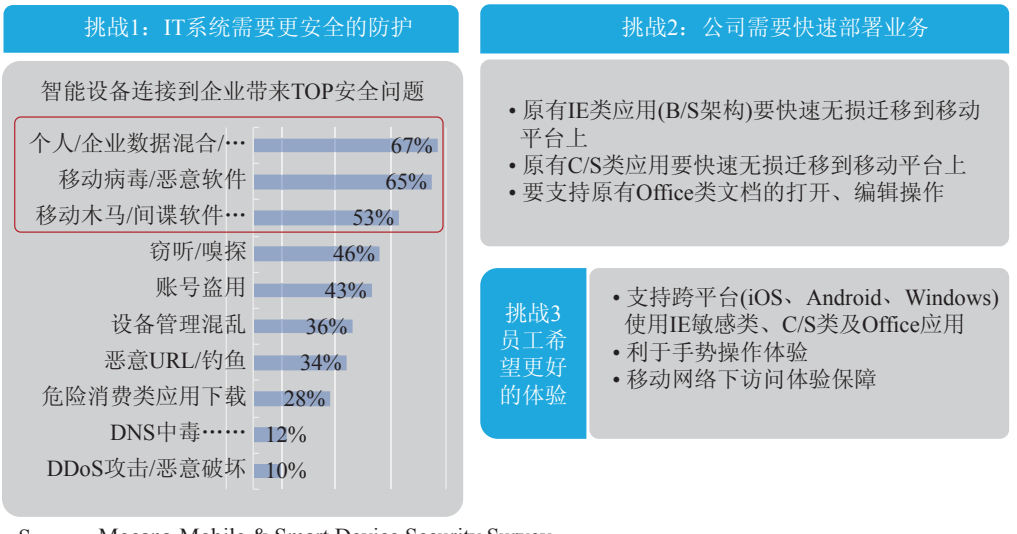


图11-1　移动办公的主要挑战

由，又不能削弱IT系统控制力，它必须以下面三条关键原则为基础。

- 简化：将终端用户资产(包括操作系统、应用和数据)从计算小环境转变为集中式IT托管服务。
- 管理：为IT创建一个中心点，用于跨公有云和私有云实现终端用户对 IT服务的访问，并能够控制终端用户具有访问权限及相应的安全级别。
- 连接：改善终端用户与IT服务及其他终端用户的连接性，且终端用户能够为手上的任务自由选择最合适的设备和应用。

11.2 桌面云接入的架构

云接入架构如图11-2所示。

- 终端侧：运行各种终端，在任何设备上随时随地访问用户应用、桌面、数据。
- 接入侧：云接入协议，实现从终端到云端的安全接入、加密传输、负载均衡、流量控制。
- 云数据中心侧：云接入统一策略管理，提供用户、应用、桌面、数据及策略管理及分发，并包含后台资源和软件的管理和配置。
- 云数据中心侧：云接入网关，提供云接入安全接入控制，协议加速。
- 办公应用：常用的办公应用如Windows办公桌面，统一通信协作软件，Web浏览器。
- 企业后台应用：支持企业日常业务运行的后台应用，如CRM/ERP/数据库。

11.3 桌面云接入的典型应用

11.3.1 桌面云的概念和价值

云接入的典型应用就是我们最常见到的桌面云。

什么是桌面云，桌面云的定义是：“可以通过瘦客户端或者其他任何与网络相连的设备来访问跨平台的应用程序以及整个客户桌面。”也就是说我们只需要一个瘦客户端设备，或者其他任何可以连接网络的设备，通过专用程序或者浏览

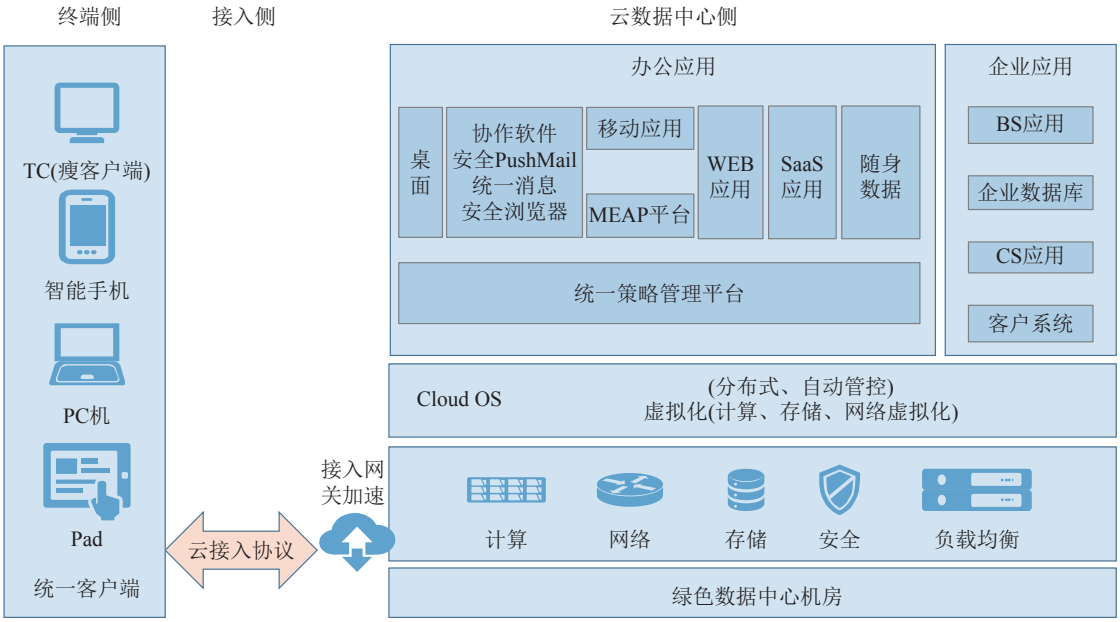


图11-2 云接入端对端架构图

器，就可以访问驻留在服务器端的个人桌面以及各种应用，并且用户体验和我们使用传统的个人电脑是一模一样的。

桌面云的业务价值很多，除了上面提到的随时随地访问桌面以外，还有下面一些重要的业务价值。

一、集中化管理

在使用传统桌面的整体成本中，管理维护成本在其整个生命周期中占很大的一部分，管理成本包括操作系统安装配置、升级、修复的成本，硬件安装配置、升级、维修的成本，数据恢复、备份的成本，以及各种应用程序安装配置、升级、维修的成本。在传统桌面应用中，这些工作基本上都需要在每个桌面上做一次，工作量非常大。对于那些需要频繁替换、更新桌面的行业来说，工作量就更大了。例如对于培训行业来说，他们经常需要配置不同的操作系统和运行程序来满足不同培训课程的需要，对于有上百台机器来说，这个工作量已经非常大了，而且这种工作还要经常变化内容。

在桌面云解决方案里，管理是集中化的，IT工程师通过控制中心管理成百上千的虚拟桌面，所有的更新、打补丁都只需要更新一个“基础镜像”。对于上面所提到的培训中心来说，管理维护就非常简单了：我们只需要根据课程的不同配置几个基础的镜像，然后不同的培训课程的学员就可以分别连接到这些不同的基础镜像，而且我们只需在这几个基础镜像上进行修改，只要重启虚拟桌面，学员就可以看到所有的更新，这样就大大节约了管理成本。

二、安全性提高

安全是IT工作中一个非常重要的方面，一方面各单位对自己有安全要求，另一方面政府对安全也有一些强制要求，一旦违反，后果非常严重。对于企业来说，数据、知识产权就是他们的生命，例如银行系统中的客户的信用卡账号，保险系统中的用户详细信息，软件企业中的源代码等。如何保护这些机密数据不被外泄是许多公司IT部门经常面临的一个挑战。为此他们采用了各种安全措施来保证数据不被非法使用，例如禁止使用USB设备，禁止使用外面的电子邮件等。对于政府部门来说，数据安全也是非常重要的，英国不久前就发生了某政府官员的笔记本丢失，结果保密文件被记者得到，这个官员不得不引咎辞职。

在桌面云解决方案里，首先，所有的数据以及运算都在服务器端进行，客户端只显示其变化的影像，所以在不需要担心在客户端出现非法窃取资料行为，我们在电影里面看到的商业间谍拿着U盘疯狂地拷贝公司商业机密的情况再也不会出现了。其次，IT 部门根据安全挑战制作出各种各样的新规则，这些新规则可以迅速地作用于每个桌面。

三、应用更环保

如何保护我们的有限资源，怎样才能消耗更少的能源，这是现在各国科学家在不断探索的问题。因为在我们地球上的资源是有限的，不加以保护的话很快会陷入无资源可用之困境。现在全世界都在想办法减少碳排放量，为之也采取了很多措施，例如利用风能等更清洁的能源等。但是传统个人计算机的耗电量是非常惊人的，一般来说，每台传统个人计算机的功耗在200W左右，即使处于空闲状态时，耗电量也至少在100W左右，按照每天10个小时，每年240天工作来计算，每台计算机桌面的耗电量在480度左右，一个具有1万桌面的中型企业，仅PC年耗电量就会达到480万度。除此之外，为了冷却这些计算机在使用中产生的热量，我们还必须使用一定的空调设备，这些能量的消耗也是非常大的。

采用桌面云解决方案以后，每个瘦客户端的电量消耗在16W左右，只有原来传统个人桌面的8%，所产生的热量也将大大减少。

四、总拥有成本减少

IT 资产的成本包括很多方面，初期购买成本只是其中的一小部分，其他还包括整个生命周期里的管理、维护、能量消耗等方面的成本，硬件更新升级的成本。从上面的描述中我们可以看

到相比传统个人桌面而言，桌面云在整个生命周期里的管理、维护、能量消耗等方面的成本大大降低了，那么硬件成本又是怎样的呢？桌面云在初期硬件上的投资是比较大的，因为我们要购买新的服务器来运行云服务，但是由于传统桌面的更新周期是3年，而服务器的更新周期是5年，所以硬件上的成本基本相当。由于软成本的大大降低，而且软成本在TCO中占有非常大的比重，所以采用云桌面方案总体TCO大大减少了。根据Gartner公司的预计，云桌面的TCO相比传统桌面可以减少40%。

11.3.2 桌面云的逻辑架构

桌面云解决方案包括7个逻辑部分：云终端、接入控制、桌面会话管理、云资源管理及调度、虚拟化平台、运维管理系统和硬件，如图11-3所示。

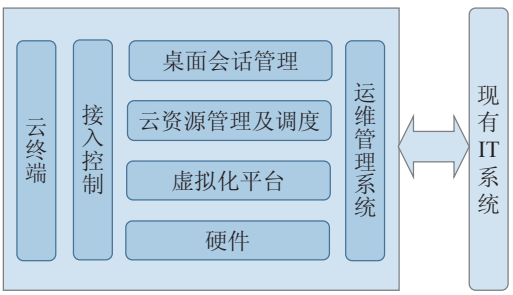


图11-3 桌面云解决方案逻辑图

一、云终端

云终端是在远端用于访问桌面云中虚拟桌面的特定的终端设备，包括瘦终端、软终端软件和各种手持终端等。

二、接入控制

接入控制用于对终端的接入访问进行有效控制，包括接入网关、防火墙、负载均衡器等设备。接入控制设备不是桌面云解决方案所必需的组成部分，可以根据客户的实际需求进行裁减。

三、桌面会话管理

桌面会话管理负责对虚拟桌面使用者的权限进行认证，保证虚拟桌面的使用安全，并对系统中所有虚拟桌面的会话进行管理。

四、云资源管理及调度

云资源管理是指根据虚拟桌面的要求，把桌面云中各种资源分配给申请资源的虚拟桌面，分配的资源包括计算资源、存储资源和网络资源等。

云资源调度是指根据桌面云系统的运行情况，把虚拟桌面从负载比较高的物理资源迁移到负载比较低的物理资源上，保证整个系统物理资源的均衡使用。

五、虚拟化平台

虚拟化平台是指根据虚拟桌面对资源的需求，把桌面云中各种物理资源虚拟化成多种虚拟资源的过程，这些虚拟资源可以供虚拟桌面使用，这些资源包括计算资源、存储资源和网络资源等。

六、硬件

硬件是指组成桌面云系统相关的硬件基础设施，包括服务器、存储设备、交换设备、机架、安全设备、防火墙、配电设备等。

七、运维管理系统

运维管理系统包括桌面云的业务运营管理和系统维护管理两部分，其中业务运营管理完成桌面云的开户、销户等业务发放过程，系统维护管理完成对桌面云系统各种资源的操作维护功能。

八、现有IT系统

现有IT系统指已经部署在现有网络中对桌面云有集成需求的企业IT系统，包括AD(Active Directory)、DHCP、DNS等。

11.3.3 桌面云典型应用场景

一、办公桌面云解决方案

办公桌面云是指企业使用桌面云来进行正常的办公活动(如处理邮件、编辑文档等)，同时提供多种安全方案，保证办公环境的信息安全。办公桌面云解决方案如图11-4所示。

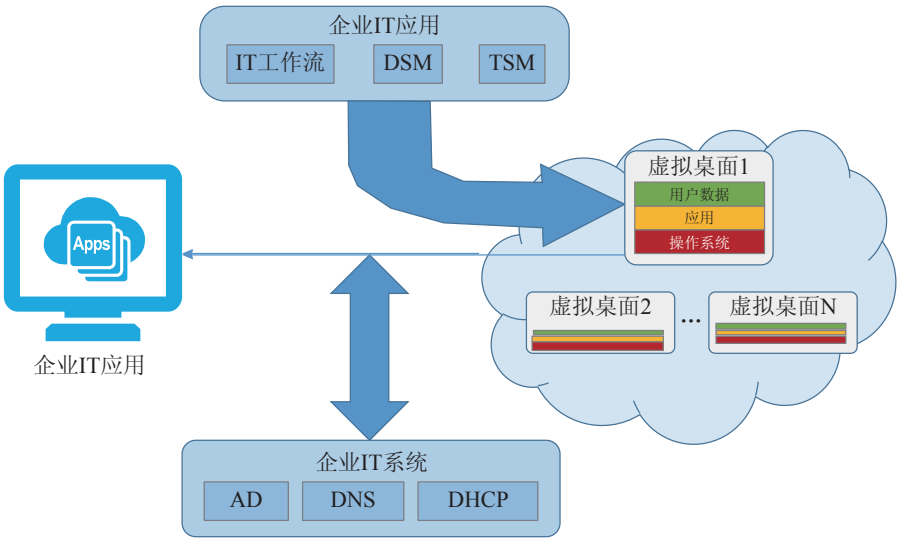


图11-4　办公桌面云解决方案

特点

桌面云支持与企业已有的IT系统对接，充分利用已有的IT应用。比如利用已有的AD系统进行桌面云用户鉴权；在桌面云上使用已有的IT工作流；通过DHCP给虚拟桌面分配IP地址；通过企业的DNS来进行桌面云的域名解析等。

优势

减少投资，平滑过渡：充分利用已有的IT系统设备与IT应用，减少重复投资，做到平滑过渡。

可靠的信息安全机制：桌面云提供多种认证鉴权与管理机制，保证办公环境的信息安全。

二、绿色座席解决方案

绿色座席解决方案如图11-5所示。

特点

多数企业用户部署的呼叫中心越来越多地由TDM方式的语音解决方案演进到采用IP语音解决方案。

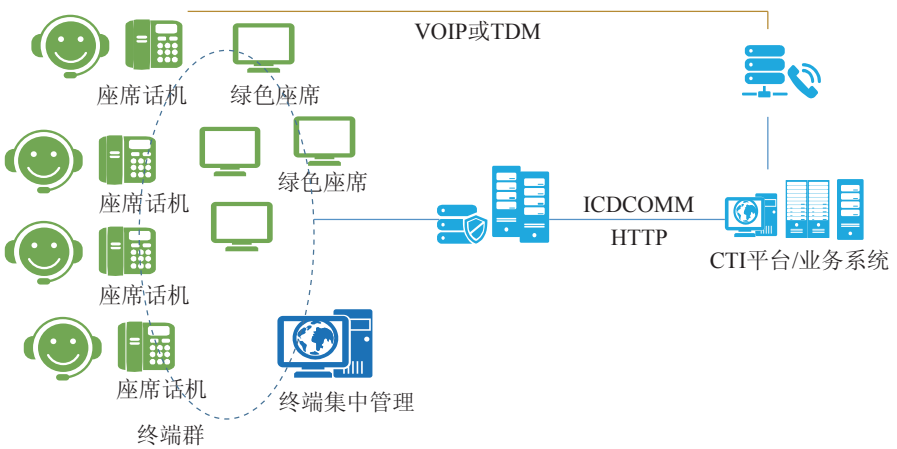


图11-5　绿色座席解决方案

优势

✍ 支持平滑迁移：完善的呼叫中心平台和桌面云的集成方案，平滑迁移客户原有呼叫中心。

✍ 快速应用，优质语音：提供桌面应用的快速响应特点和优质的语音体验。

✍ 成本优化：同类应用的共享部署模式，大大节省了虚拟桌面实例的资源占用，方便维护、升级。采用TC终端替代传统PC，降低呼叫中心的噪音、电力消耗，为客户打造绿色呼叫中心。

三、营业厅解决方案

企业营业厅系统划分为服务人员使用的桌面系统、业务办理的客户使用的自助系统。营业厅解决方案如图11-6所示。

特点

营业厅解决方案针对营业厅分布地域广泛、网络连接质量差异大的特点，改进了桌面云系统的调度和连接优化配置，可以有效地克服网络闪断、网络速率抖动等恶劣条件，保证企业营业厅系统的高质量服务。

营业厅解决方案提供即插即用的外设终端接入方案，并通过预置的具备广泛兼容性的驱动插件支持常见的串口、并口、USB口外设，极大地降低了企业客户部署的难度。

根据企业营业厅的业务特点，其支持多种桌面系统认证方式。对于客户自助系统的桌面，其还可以支持免认证登录桌面系统、即时打印服务清单等功能。

优势

营业厅解决方案是针对各种营业厅推出的解决方案，营业厅解决方案具有如下优点。

✍ 利旧原有IT外设：无须采购新的IT外设，兼容常见接口外设，并可对于外设驱动统一部署和管理，保证即插即用的客户体验。

✍ 快速软件安装部署：运营软件通过云平台集中推送，做到大规模快速软件安装部署，便于企业统一新业务上线。

✍ 支持客户自助系统：支持客户自助系统在桌面云的部署，可免认证使用企业为客户提供的系统，可即时打印服务清单等。

四、网管维护解决方案

网管维护解决方案如图11-7所示。

特点

桌面云网管维护解决方案针对网络管理的特点，定制了多种接入终端的接入程序，方便随时随地地接入进行网络状态分析与网络故障定位，对于重大问题，充分发挥企业网管专家的经验优势。

桌面云网管维护解决方案集成多种网管适配

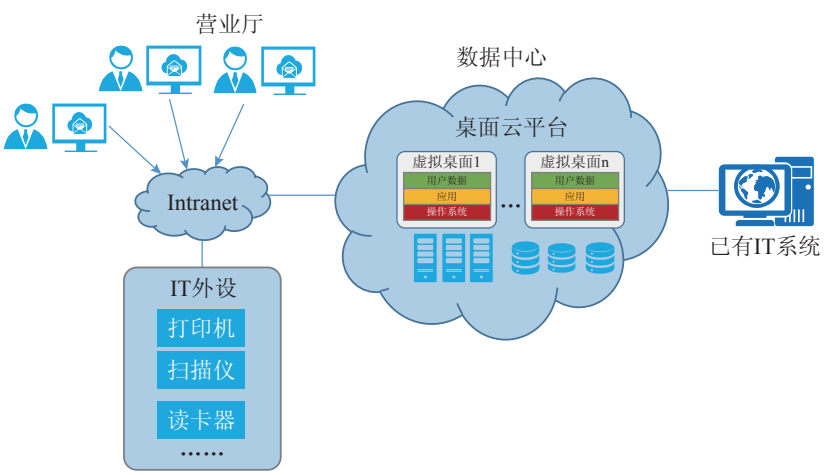


图11-6　营业厅解决方案

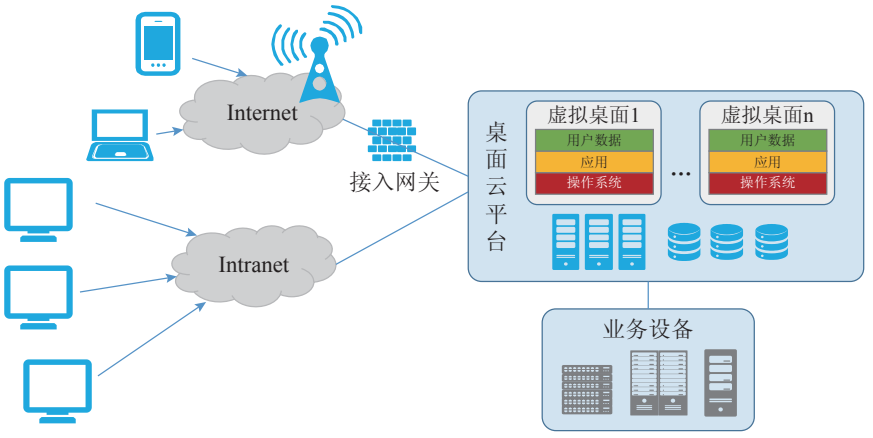


图11-7 网管维护解决方案

解决方案，无须既有网管系统进行改造，即可实现统一管理。

优势

网管维护解决方案是针对各类网管推出的解决方案，网管维护解决具有如下优点。

- 无缝接入：支持各种接入终端，包括多种手持终端(Android类、Windows Mobile类，iPhone OS类，iPad OS类，Embedded Linux类终端)，可以实现无缝地、随时随地地接入以及远程维护和监控，有利于企业发挥维护专家的优势。
- 广泛支持多种类型的网管系统：支持远程维护非C/S、B/S架构的网管系统，使用近端观测程序，极大地减少现场维护需求。
- 整合零散的网管系统：企业现有网管系统无需改造，通过桌面云系统即可以实现网络的全集中管理，提高网管维护效率。

11.4 桌面云接入的关键技术

11.4.1 桌面云协议简介

在目前云接入领域的关键技术主要是桌面云的接入协议技术，目前业界知名的有微软的RDP协议、思杰的ICA、红帽的Spice协议、华为HDP协议(我们指的接入协议关键技术并非仅指通信协议本身，还包含协议服务器端的实现与客户端的实现)。桌面协议包括具体的远程显示、远程控制、远程音频、远程外设等关键技术，而这些技术的实现具有很大的难度，所以我们认为桌面云协议是云接入最为关键的技术。

11.4.2 桌面云协议关键技术：高效远程显示

从表面上来看，桌面云高效远程显示为一个较为简单的技术，通过操作系统接口来抓取屏幕内容，再经过一定的压缩处理即可在客户端显示服务器端的屏幕内容。例如我们常用的VNC(Virtual Network Computing)就属于这一类型的实现，VNC也具有一些手段降低带宽，但是我们发现，如果与Citrix ICA、Microsoft RDP进行比较，VNC在带宽方面的劣势非常明显。

这些高性能的桌面云协议中屏幕显示基于什么原理？他们的实现有什么不同？事实上他们在实现的架构上比较类似，都是基于普通计算机的显示原理。

注：由于目前桌面云主要应用的操作系统基本都是微软的Windows操作系统，本文将直接描述Windows平台的实现(见图11-8)。

我们需要远程看到显示的内容，就好像将机器的显示器拉到远端一样。从图11-9中我们可以看到操作系统的软件层次通过操作系统，可以完全获取到显示内容以及和硬件交互的为“Windows显示驱动程序”。如果将“Windows

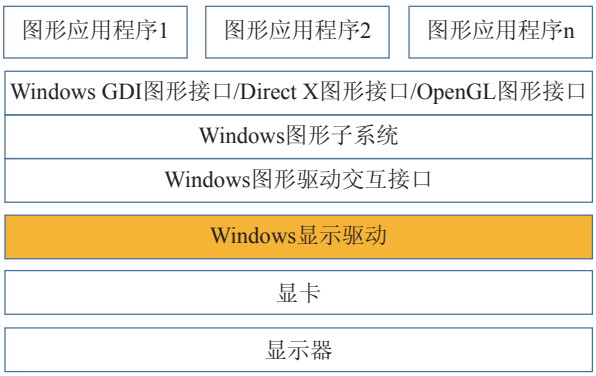


图11-8　计算机屏幕显示原理

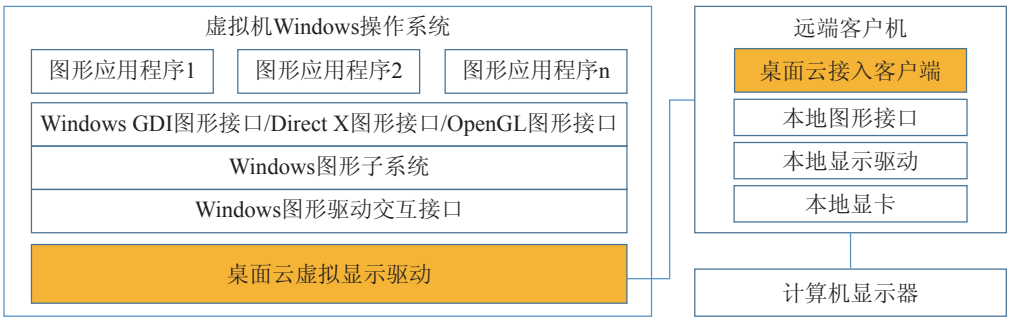


图11-9　桌面云内容在远端客户端显示的原理

显示驱动程序”发往显卡的数据传输至远程瘦终端的显卡上，即可以达到远程显示的效果。

目前，业界的实现通常会为运行在虚拟化平台中的虚拟机安装一个远程虚拟显示驱动，通过虚拟显示驱动来高性能地获取显示的图形指令数据，并将这些数据传送到远程客户机进行显示。

通常应用程序会通过Windows平台提供接口来绘图，这些图形接口调用会通过Windows图形子系统的转换来调用到虚拟显示驱动中(这些图形接口调用我们暂且称为图形指令)，图形指令内部的参数描述了图形程序的具体显示，这些数据可以被传输到远程客户端进行重新绘制显示。

这里有两个问题。

为什么不使用Windows平台的接口来直接获取整个屏幕数据，压缩后传输到客户端进行显示，而是要实现一个难度更高、更复杂的虚拟显示驱动来获取图形指令来进行处理？

通常情况下，显卡与计算机上的接口为PCIE，PCIE的带宽非常之高，16X的显卡可以拥有16GB/s巨大带宽，而我们客户端的网卡一般就100MBPS～1000MBPS，实际带宽更少，如果远程客户机与服务器端的距离较远的话，平均每用户的带宽可能不到1MBPS，这样的网络情况与原始的显卡PCIE差距很大，怎样实现远程显示？

这两个问题的答案会在下面进行解答(见图11-10)。

(1) 2D基本图形显示桌面

大多数的图形应用程序都是2D基本图形显示程序，如Word、Excel、Outlook、Notepad、杀毒软件等，通常我们普通办公场景下的程序都为2D图形程序。所以目前桌面云主要的场景为2D显示场景，如果显示驱动仅仅支持2D显示，遵从微软的显示驱动架构，可以实现微软定义的XPDM(Windows XP display driver model)显示驱动

来满足需要。

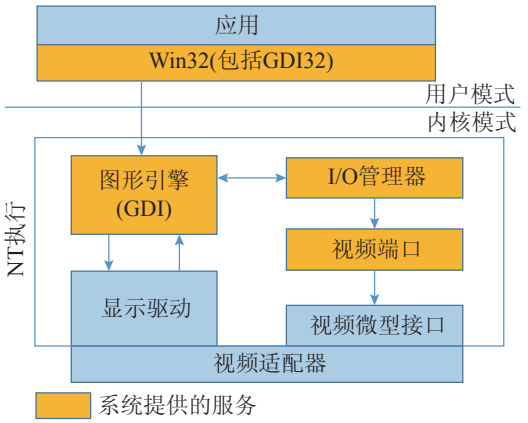


图11-10 Windows 2000/XP显示基本架构

XPDM架构是微软为Windows Vista之前的Windows版本定义的显示驱动(Vista、WIN7可兼容)，目前在桌面云场景下协议服务器端一般也会实现一个XPDM驱动，即图中显示驱动与视频微型端口驱动两部分，该驱动并非用来驱动本地显卡，而是获取Windows图形引擎向显卡发送的图形指令数据，并传输到客户端进行重新渲染显示。通常在桌面协议里面我们仅用XPDM驱动方式来支持2D应用，对于需要3D加速的应用无能为力(可以用软件实现的3D渲染来辅助支持3D应用，但是性能有限)。服务器端获取的这些图形指令需要的数据量非常大，100M网卡是不够的，所以需要加入许多优化的处理，通常的优化手段具体如下。

- 图像数据压缩：利用各种图像压缩算法对图像内容进行有损或者无损的压缩来降低带宽。
- 指令合并：图形指令的数量有时候会非常之多，通过合并技术可以显著降低指令数量与总体数据量。
- 缓存：通过缓存技术减少服务器与客户端之间的冗余数据交互。

当然实际厂商的技术可能会有不同，且优化手段更多，通过这些类型的优化手段，可以将网络带宽降低百倍甚至更多。

(2) 高性能图形支持

高性能图形指的是需要显卡辅助来支持的程序，通常是DirectX或者OpenGL程序。在目前的桌面云场景下一般仅支持普通办公，没有大量部署高性能图形的能力，主要原因是桌面云的部署还处在初始阶段，企业未大量更新到桌面云，高性能图形的桌面云虚拟机成本也更高，导致桌面云总体的技术研发投入有限，在普通办公场景下的技术基本成熟，但是高性能图形方面的高级技术还有待进一步发展，最近各厂商加大了研发投入，如VMware的vSGA (Virtual Shared Graphics Acceleration)、Citrix的OpenGL加速组件、NVidia的VGX等GPU虚拟化技术都已经推出。当然目前在高性能图形方面，直通方式更加通用，拥有更好的兼容性与性能，但是成本较高。

GPU直通实现方式一般如图11-11所示。

通过虚拟化平台的直通技术可以将显卡直接给虚拟机使用，与物理机接入显卡的效果基本一致，在虚拟机上只要安装了对应显卡的显示驱动，显卡就可以为这个虚拟机提供高性能图形的能力。桌面云服务器端程序将捕获桌面图像数据，来支持远程客户端的显示。这个方式的桌面图像处理方式与前面介绍的2D桌面处理方式有些不同，具体细节不进行介绍。

简单来说，GPU虚拟化/共享能够将一个物理存在的显卡分享给多个虚拟机使用，每个虚拟机将获得高性能图形处理的能力。

前面简单地介绍了一下2D桌面支持时经常使用XPDM方式显示驱动架构来实现桌面云图像的处理，但是实际上微软在Windows Vista以后采用了新的显示驱动架构WDDM(Windows Display Driver Model)，如图11-12所示。

WDDM显示驱动架构有三部分，为DirectX服务的用户态模式驱动、为OpenGL接口服务的OpenGL ICD驱动(由于微软主推DirectX接口，OpenGL只是可选组件)以及Display Miniport Driver(工作在内核态负责与硬件交互)。

如果需要支持GPU虚拟化技术，通常需要实现一个虚拟的WDDM显示驱动，来获取Windows对显示驱动产生的接口调用数据，并将这些接口调用重定向到一个GPU共享组件上来进行处理，

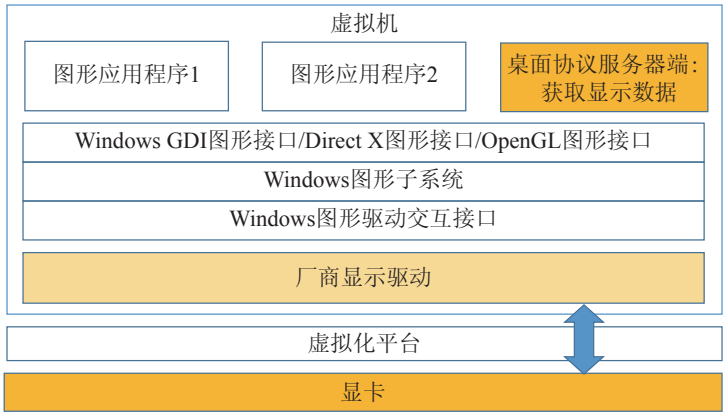


图11-11　GPU直通实现原理

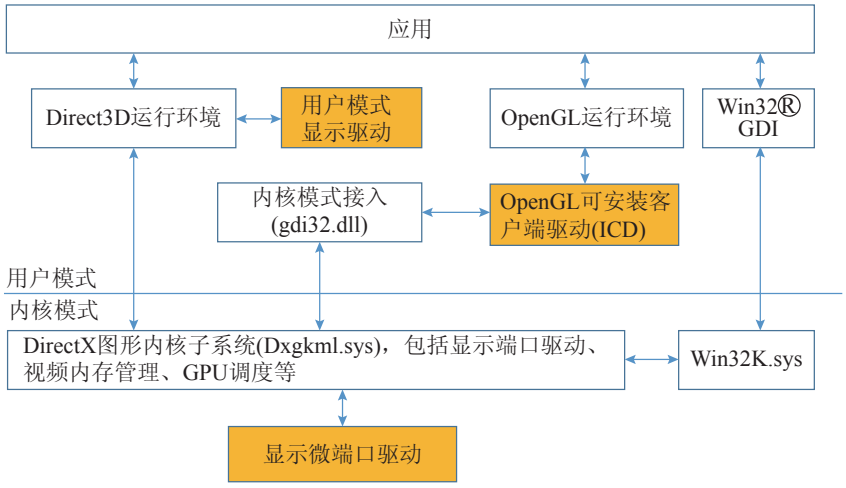


图11-12　Windows Vista/7 显示架构

该GPU共享组件一般会存在于虚拟化平台内部(也有可能是存在于其他处)。处理后将得到渲染后的桌面图像数据，这些数据将被桌面服务器端通过处理发送给客户端，可能是直接将图像编码为H.264码流，也可能是其他图像编码方式(见图11-13)。

为什么不采用与2D图形处理方式一致的图形指令重定向方式将图形指令重定向到客户端进行渲染显示，仅在客户端安装一个相对低性能的显卡？因为2D图形指令进行一系列的优化处理，带宽可以呈100倍以上地降低，仅仅1MBPS以内的带宽即可以满足普通办公的场景需求，但是采用3D应用去支持则难度很大，目前还没有技术可以将3D图形指令重定向带宽降低到如此低的程度。

11.4.3　桌面云协议关键技术–低资源消耗的多媒体视频

多媒体视频播放器对桌面云来说是一个挑战，如用户体验、音视频同步、带宽等。目前在桌面云支持多媒体视频一般有两种方式，一种是将服务器端的多媒体视频播放器的图像进行重新视频编码，将视频编码传输到客户端进行解码显

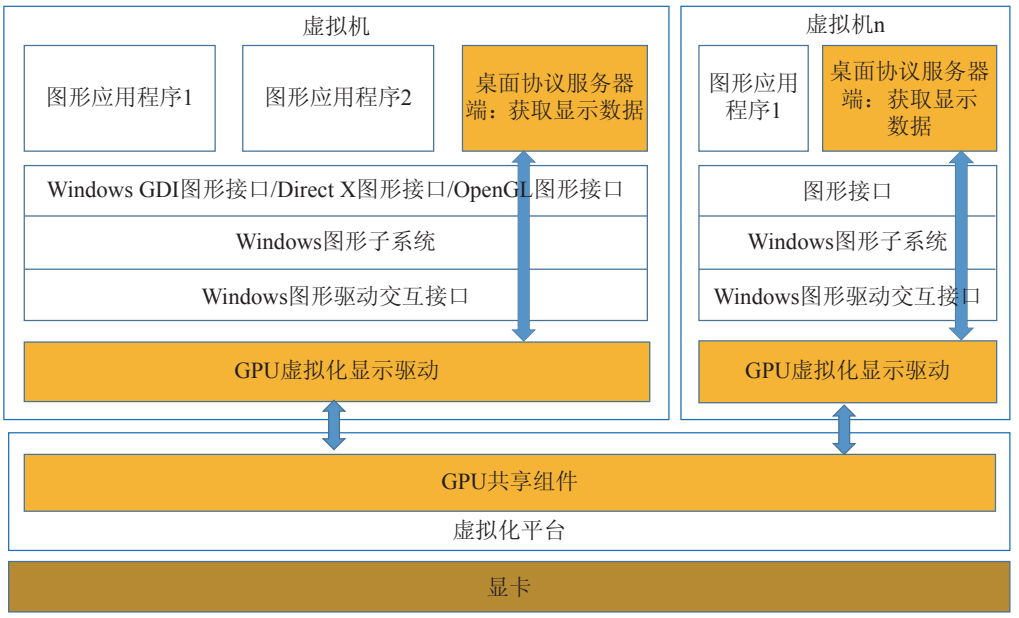


图11-13　GPU技术

示；另外一种为视频重定向方式，一般通过捕获播放器需要播放的视频编码流，直接将视频编码流发送到客户端进行解码显示。很明显，第二种视频重定向方式看上去效率更高，服务器少了视频解码与重新编码的资源消耗，但是实际上这种方式非常受限，无法得到广泛的支持(见图11-14)。

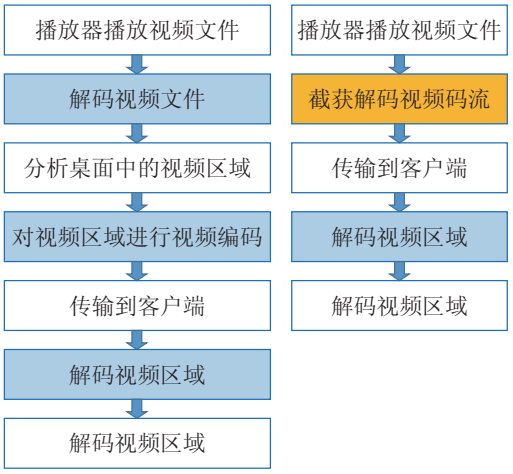


图11-14　服务器端解码视频播放(左)与多媒体重定向(右)原理

第一种方式由于在桌面虚拟机中的播放器将视频进行了解码，这里会有较大的解码CPU资源消耗，在对视频区域进行编码时消耗更大，这样将降低一个服务器能够支持虚拟机数量的密度。另外，对视频区域的识别也是一个重要的技术点，通常会通过识别刷新频率超过一定帧率的图像变更区域来识别。

第二种方式由于仅仅在服务器端截获待解码的视频码流并传输到客户端进行解码显示，对服务器端的开销较小，目前比较流行的技术为针对MediaPlayer支持的多媒体重定向技术。但是该技术在国内的实用性并不高，毕竟国内很少使用MediaPlayer播放多媒体文件，所以第二种方式实际比较受限。当然技术也在发展，对其他播放器能够支持的多媒体重定向技术相信后续也会出现，这将降低对服务器端的资源消耗。

11.4.4　桌面云协议关键技术–低时延音频

桌面云协议对音频的支持与前面介绍的2D图形支持实现比较类似(见图11-15)。

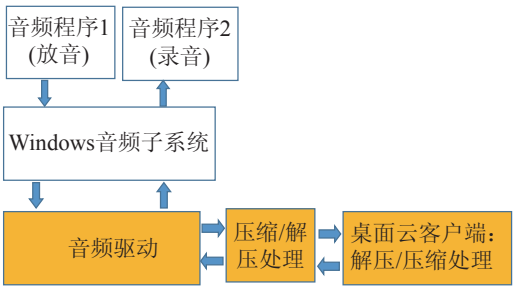


图11-15　桌面云音频输入输出实现原理

通常桌面协议服务器端可以在虚拟机里面实现一个音频驱动，音频驱动会与Windows的音频子系统(音频引擎)进行交互。在放音阶段，音频驱动将收到Windows音频子系统发送过来的音频数据，经过压缩处理后传输到桌面云客户端，客户端进行解码并进行放音。在录音阶段，客户端将获取客户端本地的录音数据，并将数据进行压缩后传输到服务器端，服务器端进行解码后由音频驱动返回给Windows音频子系统。由于音频对延时非常敏感，整个过程要关注对延时的控制。

11.4.5　桌面云协议关键技术–兼容多种外设

在通用的系统上，常用的外设种类有USB外设、并口外设、串口外设等，目前来看USB外设占据主流，解决USB外设的支持即可满足目前最为流行的外设硬件支持。

实现该部分关键技术需要先认识目前传统USB外设工作的原理(见图11-16)。

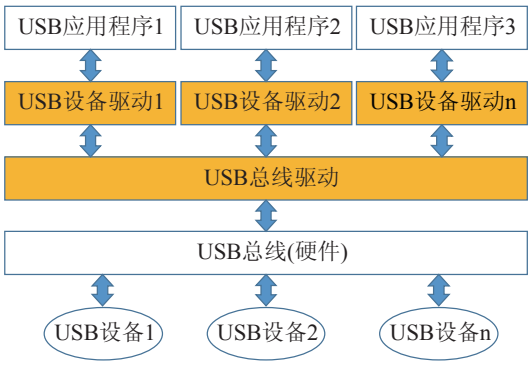


图11-16　USB实现原理示意图

从图11-16可以了解到，所有USB外设的正常工作在软件层面依赖的是USB总线驱动。一个应用需要使用USB外设必须与USB设备驱动进行交互，而设备驱动的工作完全依赖USB总线驱动来交互USB设备数据，与硬件的交互都是由总线驱动来代理完成的。从我们的理解来看，从USB总线驱动入手是软件层面最合适的方式，将USB总线驱动与本地硬件的交互远程化，转化为本地USB总线驱动与远程客户机USB硬件总线的交互(见图11-17)。

通过在虚拟机内部实现虚拟USB总线驱动可以与客户端的硬件设备进行通信交互，交互也不是直接的通信，而需要在客户端上开发一个虚拟USB设备驱动，通过虚拟化USB设备驱动与客户机的USB总线驱动进行交互。当有一个设备插入时，客户机的USB总线会发现一个新设备插入，此时将启动一份虚拟化USB设备驱动的实例，如

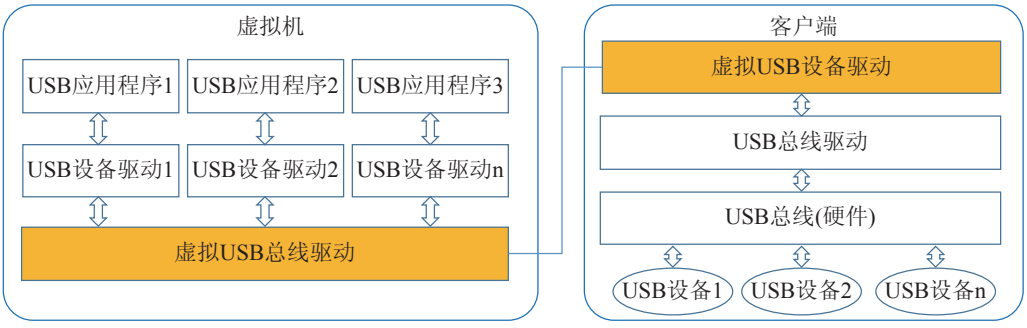


图11-17　桌面云USB外设支持实现原理图

果有多个设备需要同时被重定向时，需要多份虚拟USB设备驱动实例运行在客户端上。而设备对应的真实USB设备驱动安装并运行在虚拟机中，与虚拟机USB总线驱动进行交互，这样对虚拟机中的USB设备驱动来说并没有太大感知，对应用程序也没有太大感知，原因是，远程的这种数据交互会带来延时，有些设备驱动在设计的时候考虑了一些超时的处理。

这种方式表面上来说能够很好地支持各种USB外设，但是实际上来说也有可能存在一些问题，一是很难非常好地做到对设备的兼容，二是一些设备重定向后带宽非常之大而无法被使用，所以一些设备无法正常地使用USB总线方式来实现设备的重定向，比如摄像头，如果走总线重定向的方式，带宽有数十兆，甚至更多，这基本无法实际部署。针对这一类型的设备，一般会单独为它优化来使其可以满足实际商用。如针对摄像头，我们可以采用如下方式实现优化，如图11-18所示。

我们可以在客户端通过应用程序级别的接口来获取摄像头的数据(一般为位图数据或者YUV数据)，再将数据通过视频压缩算法(如H.264)进行压缩处理，发送到服务器端，服务器端解码摄像头视频数据后通过虚拟摄像头提供给应用程序使用。有了视频压缩技术支持，这种基于摄像头的重定向技术比基于USB总线的重定向技术带宽下降数十倍。除了摄像头类型的设备，其他类型的设备也有可能进行特定的重定向处理，这取决于单独实现针对这类型设备重定向的价值。

11.4.6 桌面云协议总结与其他实现

前面介绍了桌面云协议的一些重要的关键技术实现，这些技术主要运行于虚拟机内部，包括显示(2D、3D、多媒体)、音频、USB外设、键鼠，除了这些还存在其他的一些关键技术，另外目前桌面云的技术也在发展，肯定会推出新的技术(见图11-19)。

目前的关键技术实现都是运行于虚拟机内部的，也就是说这些技术与虚拟化平台没有关系，通常桌面云协议服务器端也可以安装于物理机器上(例如RDP就可以远程连接虚拟机或者物理机)。这个实现架构有与虚拟化平台或者硬件平台无关的优势，但是它也有缺点，如整体实现完全需要基于操作系统来实现，支持Windows的实现与支持Linux的实现将存在很大的不同。当然目前Windows是绝对主流，所以各桌面云商业厂商主要都在支持Windows操作系统。红帽的Spice采用了另外一套实现架构，这种实现架构能够做到绝大部分与操作系统无关，也就是说它可以更好地支持多种操作系统(见图11-20)。

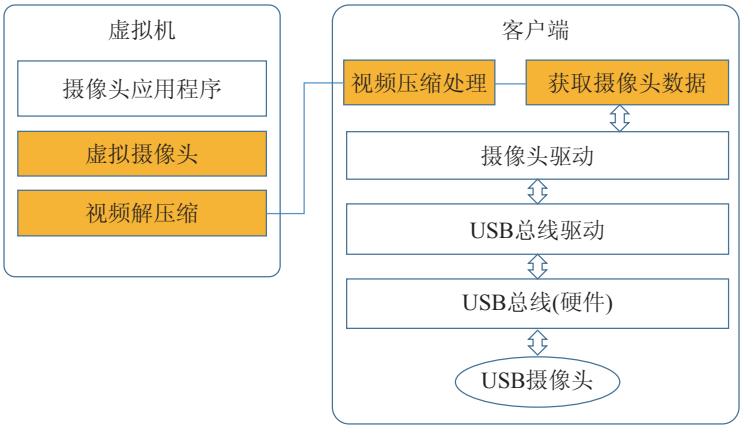


图11-18 摄像头外设支持原理

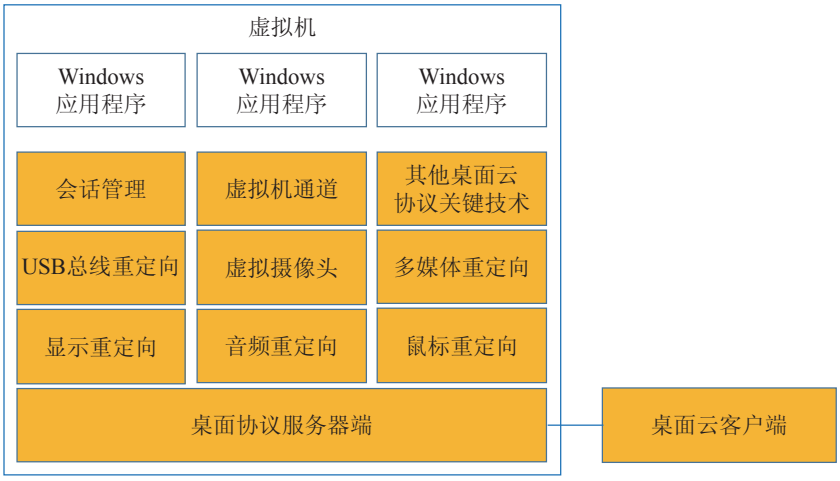


图11-19　通常桌面云技术实现原理示意图

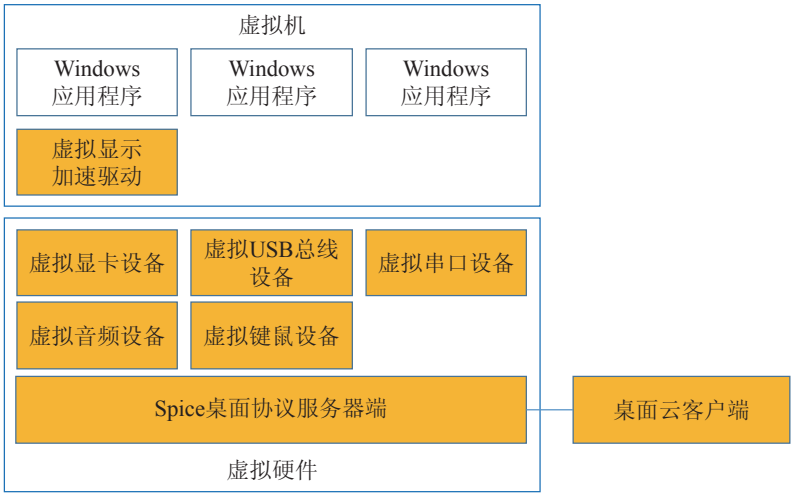


图11-20　Spice实现原理示意图

Spice的架构中大部分的实现都在虚拟化层，而不是在虚拟机中，所以Spice提供了对Linux桌面云的支持。为什么Spice在虚拟机中还提供了一个虚拟显示驱动？原因是如果不提供显示驱动，它的性能会很差，就好像如果你有一台电脑安装了显卡，但是不安装显卡对应的驱动，实际上无法使用这个显卡，同样如果没有显示驱动的加速Spice虚拟显卡，仅仅工作在VGA的模式下，则无法获取上文中提到的图形指令，无法高性能地处理重定向图形显示。当然Spice的架构还有其他独特的优势，由于工作在虚拟硬件上，不同于操作系统内部，它还可以不依赖虚拟机内部的网络与客户端进行通信，这样可以防止很多用户自己对网络进行更改，造成桌面云主机无法使用的情况，它还可以在开机启动过程中看到整个系统启动的过程(虚拟机内部的桌面协议要等待操作系统启动后才能连接)，这些都是Spice的优势。当然Spice的方式也有一些明显的缺点，它和KVM虚

拟化平台强绑定，目前官方无法支持其他虚拟化平台，无法支持物理主机接入，由于它的主要实现实体在虚拟硬件层，一些针对Windows的桌面协议优化无法支持(或者支持付出的代价更大)，如多媒体重定向、摄像头重定向等，也无法像RDP一样可以支持Windows Server操作系统多用户的接入。两种实现方式各有优缺点，目前来看主流实现以基于Windows操作系统层实现的桌面协议为主。

无论是基于Windows操作系统层实现各种驱动，还是基于Spice在虚拟硬件层次实现各种虚拟硬件，其实都是希望在底层将原本属于本地的交互转换为远程的交互，而屏幕显示的效果与本地交互相同。这种转换涉及显示重定向-用户视觉、音频重定向-用户听觉、键鼠重定向-用户触觉、外设重定向-各种外设使用，目的是利用一个简单的桌面云终端替代PC物理机为用户服务，并且需要非常高的用户体验，包括用户操作延时、桌面云显示质量、视频质量与流畅度、音频延时与质量、传输带宽大小等。整个桌面云的实现技术涉及类别非常多，包括各种虚拟驱动/虚拟硬件(显示、音频、键鼠、USB)、各种图形算法、音频算法、视频算法、带宽优化、数据去重等，所以桌面云协议的技术难度较大、门槛较高，可提供商用解决方案的也仅仅几家公司。

11.5 面向多租户的企业桌面公有云服务

11.5.1 DAAS

DaaS是什么？其指将Windows桌面和应用以云服务的形式向用户交付。它具有如下特性，如表7-1所示。

表7.1 DaaS特性

特性	解释
高可用性	7×24的服务保障
可移动性	用户可以通过互联网随时随地接入桌面和数据
安全	定时数据备份 集中管理IT架构保证企业数据安全 重要数据零丢失保证
灵活升级扩容	无须考虑软件升级 新用户扩容短时间完成
IT费用节省	IT预算可预期 无软硬件投资费用 SP提供专业服务和支持

IT服务商部署DaaS的主要驱动力为以下几点：

- 巨大的潜在市场，提供差异化服务，新的盈利点；
- 增强用户粘性；
- 带动其他服务 (备份/防病毒/企业应用托管/文件存储/数据迁移)；
- 现有的PC外包市场大约300亿美元(数据来源：Gartner)。

为什么需要DaaS？

- 更佳的桌面体验
  - 不同的设备，一样的桌面体验。
  - 无须关机，随时随地进入工作状态。
  - 统一部署和升级应用软件，更快捷。
- 简单易用的虚拟桌面
  - 自助申请，自动部署。
  - 在线支付，立即可用。
  - 专业级的运营服务。
- 投资可以预期
  - 初始投资成本低：按需申请，用完即退。
  - 运维成本低：无须自己运维。
  - 终端成本低：TC更换周期长，故障率低。
- 数据集中管控更安全
  - 数据集中管控，接入终端无保密数据。
  - 企业租户安全隔离，确保网络安全。
  - 高效快捷的外设管控策略和审计机制。

11.5.2 DAAS的典型场景

托管(hosting)模式：政府/园区(见图11-21)

混合云模式：大型企业/垂直行业 (见图11-22)
公有云服务模式 ： 中小企业/分支机构 (见图11-23)

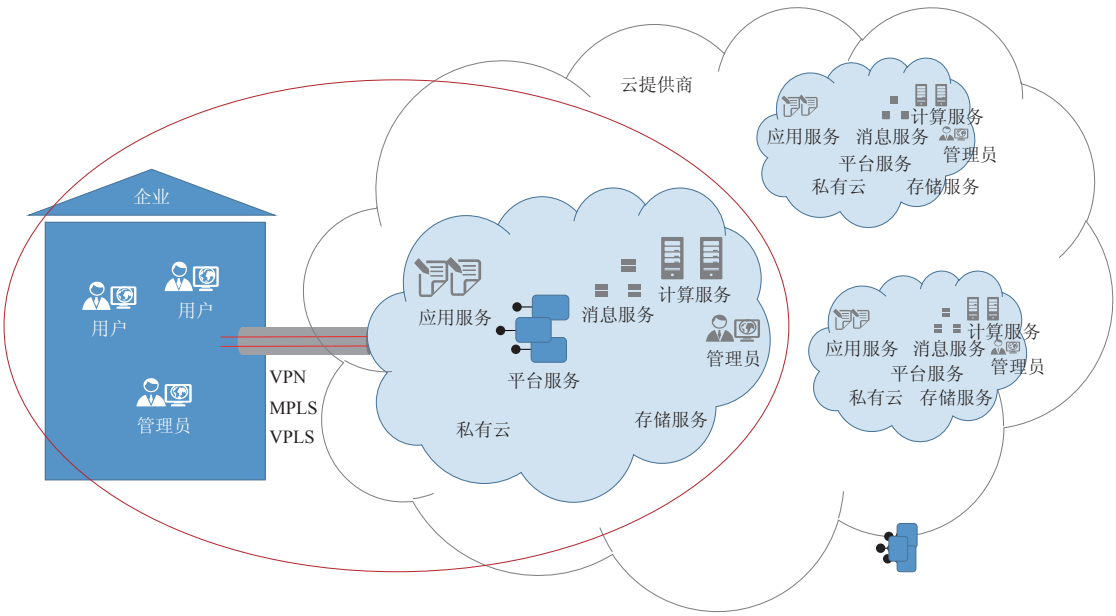


图11-21　托管模式

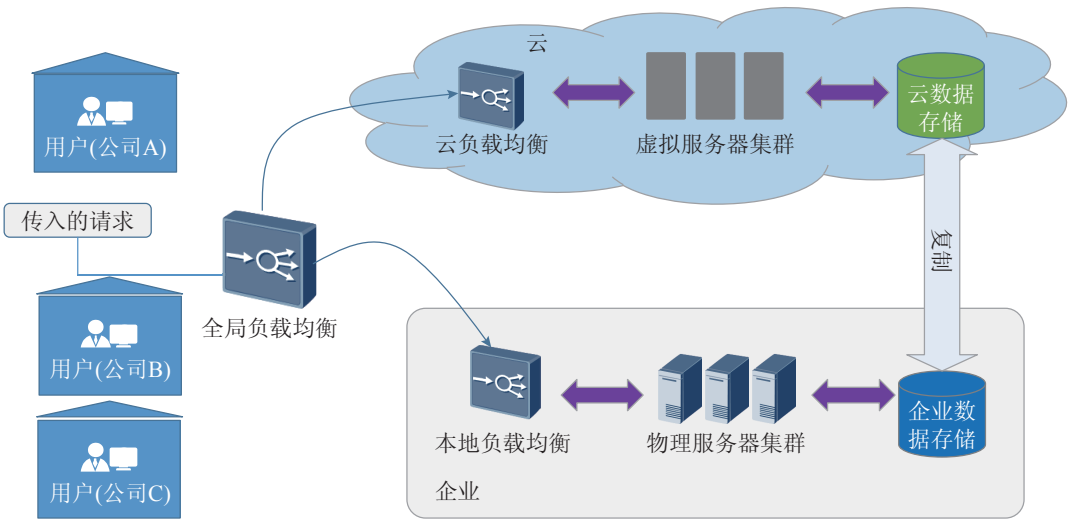


图11-22　混合云模式

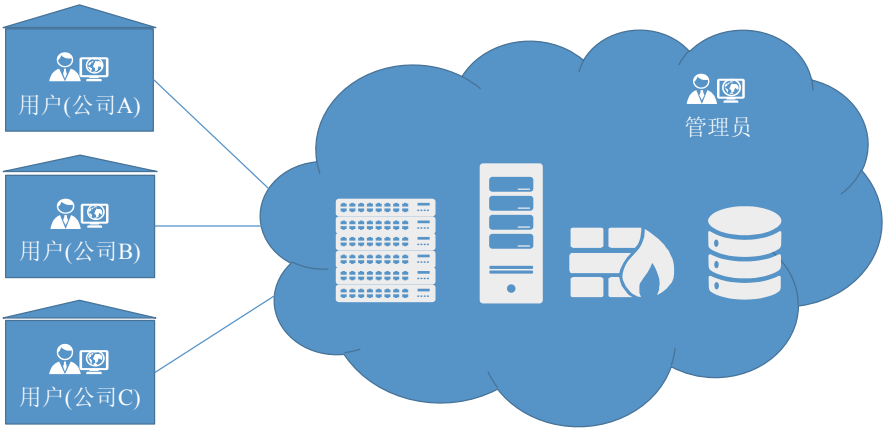


图11-23　公有云服务模式

11.6　终端无关的移动办公接入

伴随着智能设备以及3G、4G网络的蓬勃发展，企业移动办公方式向3A(Anyone Anywhere through Any Network Use Any Device to Do Anything)发展的趋势已经十分明显，大量企业纷纷升级IT基础设置，提升办公效率。各种技术方案层出不穷，各领风骚。

据某公司提供的一项调查报告显示，预计到2020年，全球89%的企业都将采用移动办公的工作方式，而目前有24%的受访者表示已经全面部署了移动办公，另有38%将扩大移动办公的部署规模，还有21%的受访者表示预计在未来两年内实现移动办公。这也就意味着移动办公市场将从2012年的24%迅猛上升到2014年的83%，年复合增长率高达86%。

据这份报告显示，北美地区的移动办公市场较为领先，90%的企业表示已经部署或正在扩大部署移动办公解决方案，紧随其后的是中国(85%)，之后则是巴西、印度、英国、法国和德国，其百分比都超过了70%。

这些数据显示移动互联网时代的到来。移动办公为何受到这么多国家或地区用户的青睐呢？此报告也罗列了原因，73%的受调查者表示移动办公方式能够使工作方式更加灵活，并有53%的受调查者认为这让出差或者旅游更加方便，48%的受访者表示能够降低办公场所的费用。吸引(47%)和留住(44%)人才也是企业部署移动办公解决方案的驱动因素之一。

移动办公一直被认为是最理想的工作方式，对企业对员工能够提供诸多方便：第一，移动办公能够提供比较弹性的办公环境；第二，能够降低企业员工的出行、生活成本；第三，能降低企业办公场所成本；第四，还能吸引更多、更好的人才。过去人们常说的一句话——21世纪最缺的是什么？人才。如果企业实现了移动办公，在员工不方便到办公室上班的时候可以允许员工在家移动办公，既保证了业务的连续性，又能提供人文关怀，这样的公司谁不想留下呢？

那么，移动办公什么时候才能实现呢？某公司授权一家市场调查公司独立做了一项针对移动办公未来趋势影响的调研，涉及19个国家、2000名IT高管。根据这项移动办公调查报告显示，移动办公已经成为全球企业办公的重要趋势，调查中大约24%的企业已经部署移动办公方案，21%的企业计划在两年内部署移动办公方案，预计到2014年，将有86%的企业实现移动办公。预计到2020年，全球企业办公空间将减少17%，中国企业的办公空间将减少21%。预计每10人平均需要7张办公桌，在新加坡、荷兰、美国、英国平均需要6张办公桌，而在中国，这个数字是7.69。

通过被访者对可能会进行移动办公的地点的回答，可以一窥未来工作场所的特点。在被访者回答中，95%的中国企业用户会选择在本地办公室接入移动办公平台，亚太地区平均数据为73%；90%的中国企业用户会选择从家里接入移动办公平台，亚太地区平均数据为62%；84%的中国企业用户会选择从公司的项目所在地或其他现场接入移动办公平台，亚太地区平均数据为64%；81%的中国企业用户会选择从客户、伙伴、供应商那里或活动现场接入移动办公平台，亚太地区平均数据为53%；48%的中国企业用户会选择从咖啡厅、餐厅、图书馆以及类似公共区域接入移动办公平台，亚太地区平均数据则为30%；其余地点还包括在各种交通工具上。如此看来，移动办公使得办公并不限于办公室或者家中，而是真正实现了随时随地办公。

在调查中，中国企业用户对于移动办公的热情远高于其他国家和地区，考虑到中国地缘广大、东西部发展速度不均衡，在沿海地区和一、二级城市的企业用户的移动办公水平相当乐观。44%的中国被访企业表示有专门的团队进行移动办公解决方案的开发和管理，28%的企业表示将在2012年底之前成立这样的团队。之所以引入移动办公，其主要驱动力是来自员工的主动要求。而在中国，IT高管表示第一驱动力是因为终端设备种类繁多，其次是出于安全和风险控制的考虑。调查显示，全球人均拥有6台不同的计算设备，而在中国，这一数字为3.06台。

11.6.1 移动办公的概念和价值

移动办公是云接入的另一种常见的典型应用。

一、移动办公与移动办公系统

移动办公也称移动OA，是指利用手机、PDA或笔记本电脑等移动终端通过无线网实现移动办公的移动信息化手段，移动办公软件是实现移动办公的软件应用系统，它使得移动办公人员摆脱时间和场所局限，随时进行随身化的信息管理和沟通，从而有效提高管理效率，推动政府和企业效益增长。

移动办公是当今高速发展的通信业与IT业交融的产物，它将通信业在沟通上的便捷、在用户上的规模，与IT业在软件应用上的成熟、在业务内容上的丰富，完美结合到了一起，使之成为继电脑无纸化办公、互联网远程化办公之后的新一代办公模式。这种最新潮的办公模式，通过在手机、Pad上安装企业信息化软件，使得手机、Pad也具备了和电脑一样的办公功能，而且它还摆脱了必须在固定场所固定设备上进行办公的限制，对企业管理者和商务人士提供了极大的便利，为企业和政府的信息化建设提供了全新的思路和方向。它不仅使得办公变得随心、轻松，而且借助手机通信的便利性，使得使用者无论身处何种紧急情况下，都能高效、迅捷地开展工作，对于突发性事件的处理、应急性事件的部署有极为重要的意义。

移动办公系统，是一套建立以手机等便携终端为载体实现的移动信息化系统，系统将智能手机、无线网络、OA系统三者有机结合，实现与任何办公地点和办公时间的无缝接入，提高了办公效率。 它可以连接客户原有的各种IT系统，包括OA、邮件、ERP以及其他各类个性业务系统，可通过手机操作、浏览、管理公司的全部工作事务，同时提供一些无线环境下的新功能。其设计目标是帮助用户摆脱时间和空间的限制，随时、随地、随意地处理工作，提高效率、增强协作。

二、移动办公的意义

移动办公是一种新型的低碳办公模式，能为企业和社会节约资源，减少废气排放。

移动办公是一种无纸化低碳办公模式。移动办公系统通常能支持pdf、jpg、doc、xls等文件格式，这些文件格式基本覆盖了大多数企业内的文件审批格式。现在，领导使用手机等移动终端即可打开各种待审核和待审批公文，远程进行批复。业务人员在与客户会谈前总是需要先打印出各种准备资料，而在使用装有移动办公系统的手

机端之后，只需在会谈期间根据需要随时进入公司系统进行查阅，同时运用PPMEET等相关视频会议软件进行会议的召开。

移动办公是一种电能消耗极少的低碳办公模式。与大约每小时消耗0.2～0.3度电的PC相比，手机消耗的电量几乎可以忽略不计。企业是电能消耗大户，对于移动办公任务较多的企业来说，通过移动办公每月可节约的电量将是一个不小的数字，同时可为社会节约电力资源。

移动办公大大减少了用户在交通工具所需的汽柴油等燃料方面的消耗，同时减少含有大量二氧化碳气体的尾气排放。这是一种典型的污染物排放少的低碳办公方式。业务型员工和领导移动办公任务重，在外忙碌了一天之后往往还需要返回公司，处理一些收尾的工作，例如将业务信息录入系统，查询最新的通知公告等。现在，移动办公系统省去了用户返回公司的必要，大大减少了乘坐交通工具所需的燃料方面的消耗。

如今，移动办公已经不仅是一种节约能源、减少二氧化碳污染的低碳办公模式，还是提高员工办公效率、提高企业综合竞争力、提升公众形象的一种手段。

11.6.2 移动办公的逻辑架构

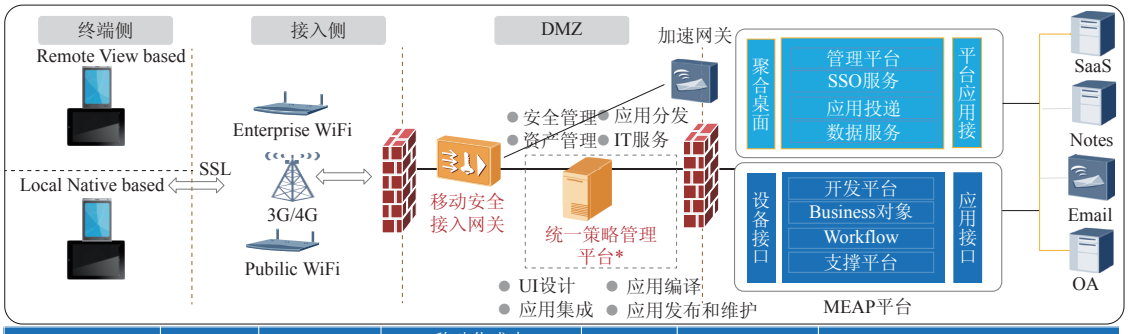
一、主流解决方案：原生应用与远程应用

图11-24对比了两种主流的移动办公解决方案，我们可以从中看出，远程应用在安全、维护成本、上线周期等方面具有明显优势，在离线使用、移动办公体验方面稍显逊色。由此可见，远程应用模式的移动办公更加适用于高安全要求、快速迁移等场景。

图11-25为移动办公方案全景图，图中应用虚拟化(SBC)被公认为移动办公最佳解决方案之一。

二、移动办公解决方案(远程应用模式SBC)

图11-26为SBC移动办公解决方案，向企业用户提供一个应用与数据集中管理与发布平台，管理员用户可以分别通过管理员Portal进行应用管理、数据管理、用户管理、服务管理。同时，集成商或企业IT部门可以通过开放API进行二次开发。



解决方案类型	安全	运维成本	移动化成本 (上线周期)	可离线	移动办公体验	应用场景
远程应用 (SBC)	非常高	低	低	否	中	1、高安全 本地无数据，网络上传输的只是客户端的键盘、鼠标动作以及显示界面的刷新部分 2、快速无损部署移动应用
原生应用	中	中	高	是	中高	1、操作体验要求高 2、满足网络条件差的场景

图11-24　主流移动办公方案对比

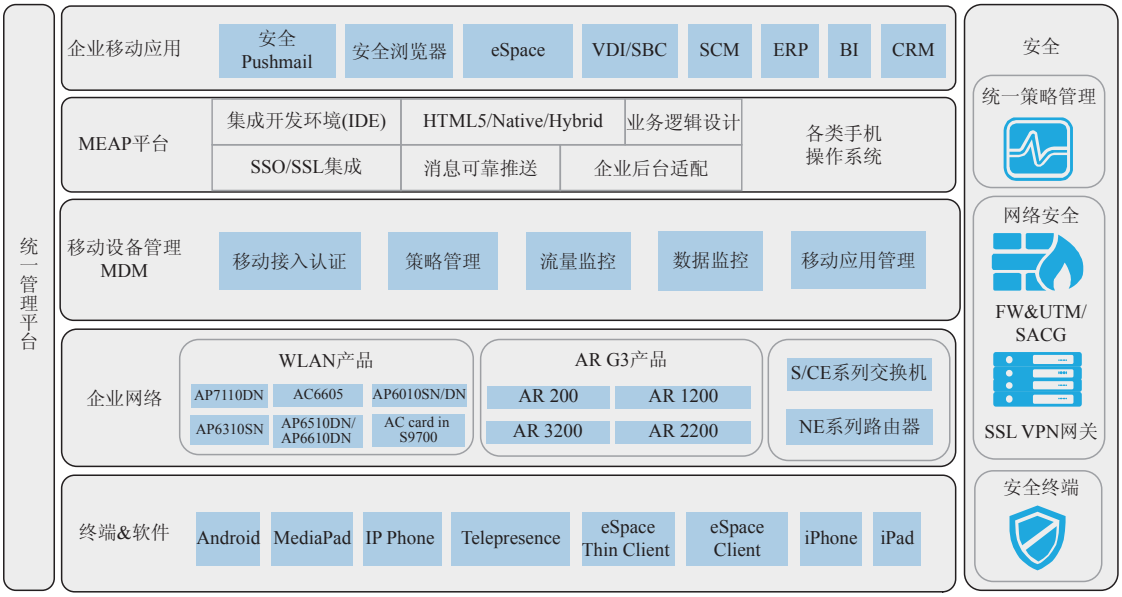


图11-25　移动办公方案全景图

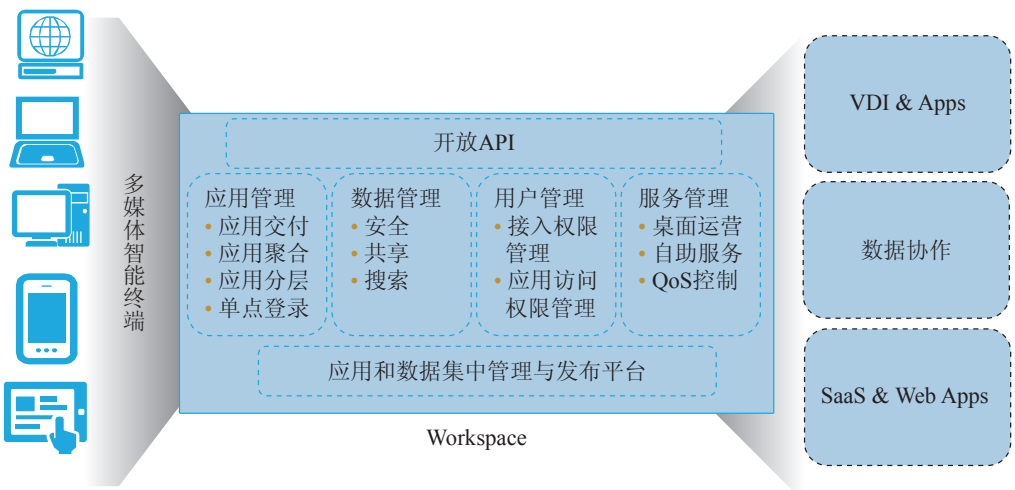


图11-26　移动办公应用集成平台

11.6.3　移动办公解决方案的特点

移动办公解决方案具有的特点如下所示。

端到端数据安全防护，使企业更聚集于业务发展

- 文档在线编辑，无法保存到客户端本地；若操作超时，则会锁屏。
- 链路层端到端加密，保障传输安全。
- 用户鉴权。
- 企业资源访问控制。
- 文档与应用分离。
- 应用进程间隔离。
- 应用层软件网关。

一键式安装部署

- 无人值守安装，深度定制Linux，界面友

好，多节点复用统一镜像。

- 配置简单，自动时间同步，支持远程配置、远程定位。
- 移动应用快速上线，后台无须修改。
- 移动应用快速升级，客户端无须升级。

出众的用户体验

- 定制化的快捷工具栏。
- 拍照插入文档。
- 触摸可编辑放大镜。
- 自动弹出键盘、自动滚屏。
- 分区域滚屏。
- 便捷的任务管理。

资源调度最优化

- 网关负载均衡。
- 应用服务器负载均衡。

通过广域网加速技术，提升应用访问速度

- TCP协议优化。
- 实时流压缩。
- 绘图指令重定向。
- 数据缓存。
- 云模式+广域网加速技术大幅提升应用访问速度。

第 12 章

第三方云应用生态Marketplace及应用编排自动化

12.1 基于开放云平台的云生态系统构建

整个云系统包括L1层基础设施、物理资源、虚拟化系统、云管理系统和上层应用，还包括租户、设备商、系统提供商、服务运营商、集成商、应用软件厂商等组成的一个复杂系统，需要各个厂商和系统一起配合，构建生态，才能建立比较完整的产业链。本章重点阐述云应用生态系统构建相关探索和实践。

12.1.1 为什么需要构建云应用生态系统

一、应用是生产系统，应用是最终目标

传统模式下我们将应用装在一个一个独立的物理设备上，云模式下我们将应用装在可共享的物理设备或者虚拟设备上。不管是传统模式还是云模式，变化和改变的是资源共享和资源提供方式。这些技术手段最终都是为了满足应用需求，承载各种应用，给企业和社会带来价值。应用是和业务紧密相连的，是生产系统，是应该关注的核心。

所以光有云平台是不行的，需要有周边各种生态，特别是应用厂商的配合才能让整个云系统生机勃发。可以简单类比，云平台就像一栋刚装修好的商场，应用就像商场中的专柜，商场装修很好，没有专柜品牌，或者品牌不丰富，商场是不能吸引客户的。

因而我们谈云平台的开放性，更多的是要关注该云平台是否有机制和能力吸引大家，有利于各厂商相互配合来构建一个完整易用的系统。是否有利于整个云应用生态系统的构建。

应用的安装、部署复杂，需要专家经验

传统模式下，一些企业的复杂应用(如数据库应用等的部署和配置)需要相关领域专家几周甚至更长的时间才能完成。对于没有专家经验储备的部门，则需要花费更长的时间。在云环境下，底层物理和虚拟资源的标准化以及它们和应用实现了解耦，为应用部署和配置的自动化提供了可能，此时如果能有一种平台能够将专家经验调测好的应用部署配置模板共享出来，这样专家的经验和能力就能实现极大化共享，那么即使对应用不熟悉的部门，也能够毫无困难地完成一个复杂应用的安装和部署。

二、应用需要更快地投入生产、产生效益

传统的复杂应用在安装配置上需要几周时间，再加上设备申请、采购、网络规划等，可能需要半年甚至更久才能正式在生产系统使用。 在云平台上，借助应用市场，仅花费分钟级的时间，租户就可以自助获得一套完整的应用环境，包括资源申请、网络配置、应用安装、配置、调优等涉及应用的各方面全部可以完成。

三、满足企业复杂的应用使用场景需要

一般企业的一套应用涉及在开发环境、测试环境和生产环境的镜像和同步过程，如何保证开发环境开发的应用能够传递到测试环境，如何保证测试环境经过测试调优后的应用直接拷贝到生产环境，生产环境的应用数据如何脱敏后反向传递到测试环境用于模拟生产环境的验证使用，这些应用市场都可以成为一个很好的中介串联作用。

12.1.2 云应用生态系统组成

云应用生态系统涉及各个方面，一般云应用生态系统如图12-1所示。

云生态系统组成包括设备厂商、虚拟化厂商、云服务/云平台厂商、云服务运营商、应用软件厂商、集成商和租户等。

- 设备厂商：如华为、思科、IBM、HP、EMC等，提供计算、存储和网络设备。
- 虚拟化厂商：如华为、VMware、红帽，微软等，提供各种虚拟化系统。
- 云服务/云平台厂商：如华为、VMware、IBM、HP等，其提供云服务和云平台，当前主流云平台基于OpenStack架构，同时提供Marketplace。
- 云服务运营商：提供云服务给租户使用并运营。
- 应用软件厂商：开发和提供应用软件，

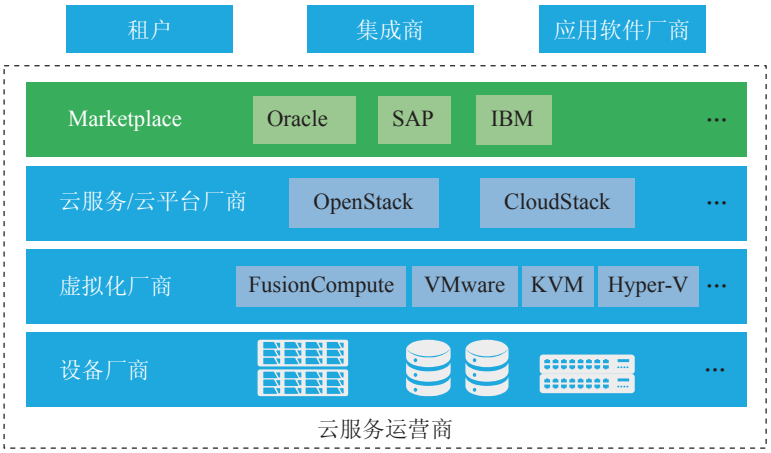


图12-1　云应用生态系统

有时应用软件厂商也可能自己制作应用模板并发布到应用市场中。

✍ 集成商：将某个应用软件或者多个应用软件的组合制作成应用模板并发布到应用市场中。

✍ 租户：申请和使用应用，并支付费用。

12.1.3　怎样构建云应用生态系统

图12-2为云应用生态系统示意图。

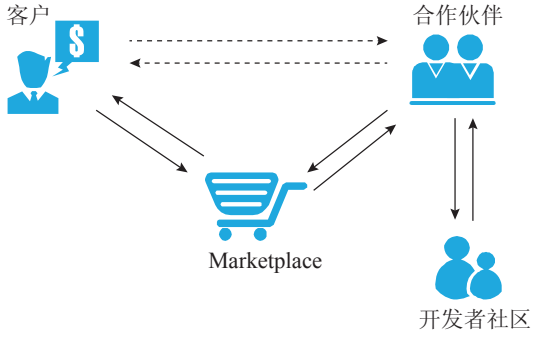


图12-2　云应用生态系统关系

一、提供平台能力，保障各角色专注于自身擅长领域

以Marketplace为基础，围绕Marketplace提供一个合作伙伴产品营销、推广和变现的平台，整合了上游的应用，结合开发者社区共同建设开发者生态圈，最终吸引租户使用云应用。

合作伙伴：应用Marketplace提供销售平台和底层云计算资源和应用，并且提供支付和处理账单的能力，合作伙伴只需要聚焦自己的应用，在此平台便捷地销售自己的产品。

租户：在Marketplace通过一键开通的简单操作，就能获得一套云主机资源和已经部署好的应用软件，系统会自动启动预配置的软件，用户只需要聚焦自己的业务。

二、提供开放能力，保证各角色方便地使用系统

开放能力涵盖多个方面，具体如下所示。

- 针对开发者的开放：提供开发接口，基础中间件服务支持，开发集成测试发布环境等。
- 针对合作伙伴的开放：产品管理能力的开放，含产品管理、上下线、营销推广、计费结算等。
- 针对租户的开放：即租户购买、使用、监控应用的开放接口。

✍ 社区建设，实现快速的经验传递和共享。

其包括开发者社区、合作伙伴社区、租户社区的建设。

重点是开发者社区建设，开发者社区要为开发者提供易集成的多语言SDK、开发工具、互动论坛，全方位支持开发者进行二次开发，并基于Marketplace进行营销，最终吸引开发者/租户使用云应用。

12.2 Marketplace系统架构

Marketplace的一般架构如图12-3所示。

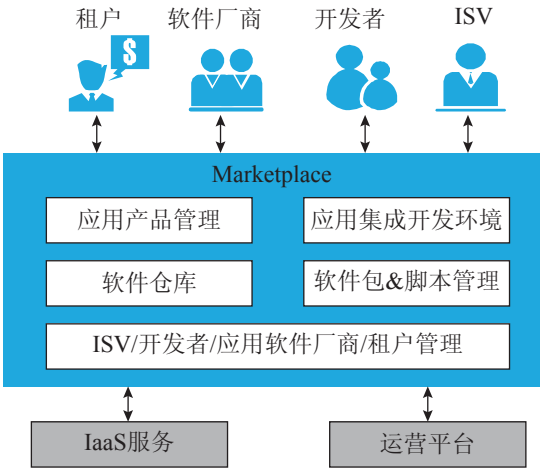


图12-3 Marketplace的系统架构

一、周边系统

系统周边和租户、软件厂商、开发者、ISV、IaaS服务和运营平台有集成配合关系，具体如表12-1所示。

表12-1 生态配合关系

周边系统	配合关系	备注
租户	提供Marketplace中的产品给租户使用，记录租户的使用情况	可以是租户或租户应用系统，下面的软件厂商、开发者和ISV类似
软件厂商	向Marketplace中发布应用软件包 推广应用软件包 了解应用软件包的使用情况和获取受益	
ISV	向Marketplace发布应用软件包、应用配套脚本、应用部署模板等 应用产品管理，含上下线等 了解应用软件包、应用配套脚本、应用部署模板的使用情况和获取受益	
开发者	同ISV，主要是小应用和应用配套脚本等。	
IaaS服务	租户申请应用产品实例时需要IaaS服务支撑完成IaaS层资源提供和应用的自动化部署	
运营平台	完成统一用户、鉴权、订单、计费等功能	

二、系统内部

系统内部主要功能如表12-2所示。

表12-2

系统组件	配合关系	备注
应用产品管理	应用产品的发布，下线管理 应用产品搜索查询，分类管理 应用产品的购买使用	
应用集成开发环境	可视化应用蓝图设计 应用开发工具：含应用开发环境、源码控制、调试、监控和测试等 软件基础设施支撑：含各类中间件软件 应用蓝图验证环境	
软件仓库	各类应用软件包、脚本、镜像等的存储中心 软件资产基于用户的分权分域管理	
软件包，脚本管理	各种软件包脚本的管理，包括上传、下载、在线查看、发布、下线、删除等	
ISV/开发者/应用软件厂商/租户管理	针对各角色的分权分域管理功能	

12.3 面向电信网络和业务云化的CT编排自动化——MANO

随着云计算技术的成熟和普及，电信运营商也开始考虑如何把电信设备转移到云计算上，从

而降低电信网络的建设成本和运营成本。这种电信网络演进技术被称为NFV (Network Function Virtualization)，如图12-4所示。

传统的电信设备，每种设备都采用不同的专用硬件，在上面运行针对硬件定制的专用软件，很多这样的设备放在一起就像一个个烟囱，运维人员需要管理很多种硬件。旧的设备淘汰后也不能被重利用。在引入NFV技术之后，所有的电信设备都将采用通用的硬件，管理起来更加方便；通过云计算技术将电信软件和硬件解耦，旧的设备业务量减少或者退网后可以把资源共享给其他设备使用，从而降低了运营成本。

在引入NFV技术后，如何将不同供应商提供的电信软件、CloudOS、硬件集成，实现快速便捷的部署和管理，就成为一个重要的问题。这些部署和管理功能被称作MANO (Management and Organization)。

12.3.1 MANO的需求

IT业界已经有了一些编排自动化的工具，如Cloudify、OpenStack Heat，为什么还要再提出一个MANO的概念呢？我们先来看一看VNF的部署和生命周期管理到底有哪些特殊的需求。

(1) 高性能

VNF往往需要同时支持高计算性能和高网络转发性能。一般情况下，高计算性能可以使用通用服务器硬件架构解决，高网络转发性能可以通过路由器/交换机硬件架构解决，两者同时有需求时，传统上是通过定制专用硬件解决的。传统的虚拟化技术还不能完全满足VNF的性能要求，需要在通用云计算硬件的基础上进行增强，如支持SRIOV转发、CPU亲和性、NUMA分区感知等，相应地也需要对云管理和编排系统进行增强。

(2) 高可用性

电信应用由于关系国计民生，可用性要求一般要达到99.999%，一般的云系统往往只能提供99.9%~99.99%的可用性。因此用于NFV的云系统要在可用性设计(如硬件故障快速检测、跨层故障定位、反亲和性部署等方面)上进行增强。

(3) 多层级的网络业务编排

一般的IT应用可以简单地看做是一个服务，为很多客户端提供点到点的服务，部署时在IT应用内部对虚拟资源和服务组件进行编排。而电信业则是由多个VNF通过一定的拓扑关系形成网络服务，除了在一个VNF内部对资源和服务进行编排外，还包括对多个VNF进行编排形成端到端的服务。

以图12-5为例，图中的每一个方框，如MGCF、BGCF等，都是一个VNF，通过多个虚拟机和虚拟网络编排而成。多个VNF按一定规则互联起来，可以形成一个网络服务(NS： Network Service)，即图中的IMS，可以为语音或其他多媒体业务提供核心网服务。而IMS又和其他的VNF/NS，如PCRF、EPC等一起组成了更大的网络服务VoLTE。这种多层级的业务编排功能是一般IT编排工具不具备的。

(4) 对网络拓扑的编排能力要求更高

IT应用一般是提供点到点的服务，网络结构简单。而NFV则会存在复杂的网络拓扑，对网络拓扑的编排能力要求更高。

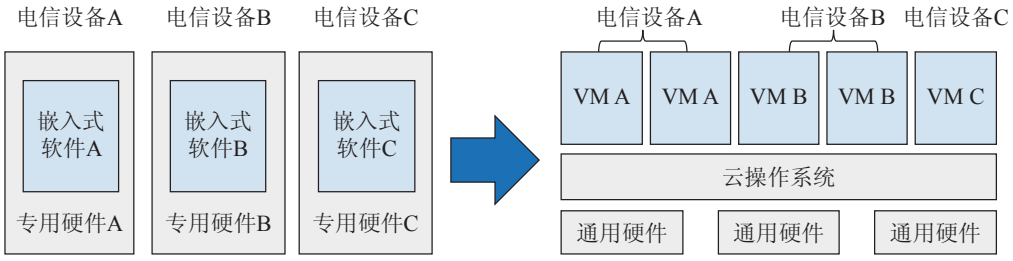


图12-4　电信设备从烟囱式向NFV演进

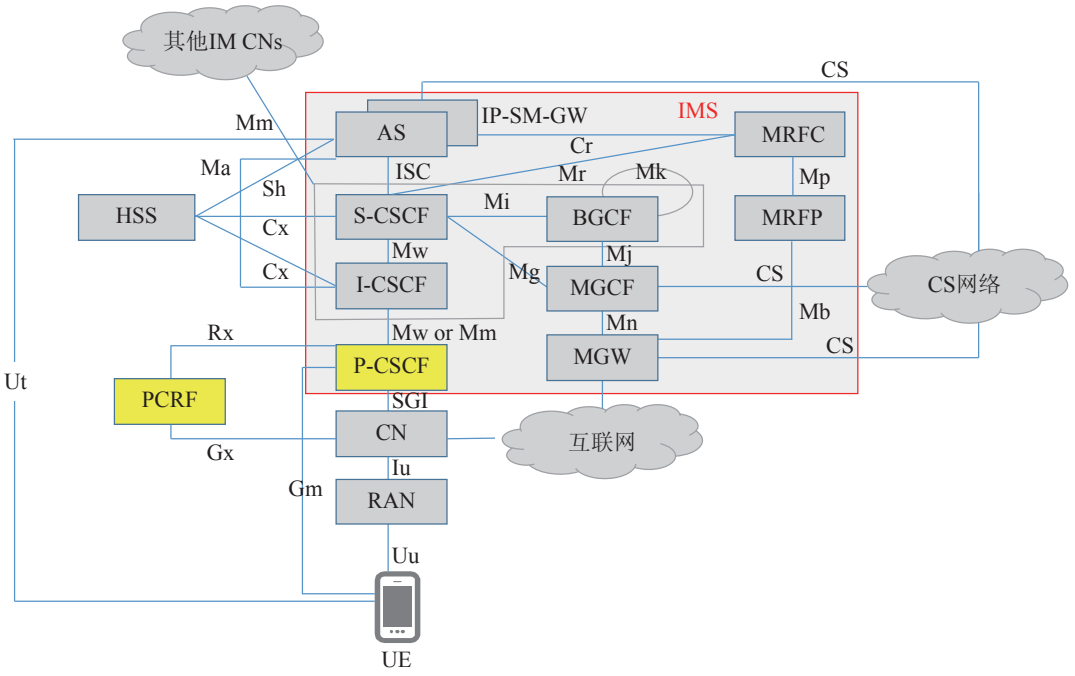


图12-5　电信业务多层次的网络拓扑

(5) 复杂的业务配置管理

IT应用部署后一般通过脚本方式完成部署，而电信业务的配置一般比较复杂，往往涉及数百条命令和参数，如果依赖手工书写脚本进行配置，那么配置脚本的写作就会成为部署效率的瓶颈。

(6) 存量资源的利用

为了保护电信运营商已有的投资，传统的物理设备(PNF)需要能和VNF共存。VNF部署时需要考虑如何和PNF互通。

(7) 可运营

电信运营商为了经营，已经花费了很大的成本开发了运营系统(BSS/OSS)。引入NFV技术后，已有的投资需要能尽量得到沿用，NFV系统需要能接入已有的BSS/OSS系统。

12.3.2　MANO的参考架构

NFV的标准化研究由ETSI ISG NFV负责。其中针对MANO的部分有一个专门的工作组。相关的标准建议已经在2014年12月发布了第一阶段的成果，在本书写作期间该工作组在进行第二阶段的讨论。图12-6为NFV的架构图。

在这个参考架构中，左侧部分是NFV功能部分，包括如下几点。

- NFVI(NFV Infrastructure)：提供NFV的运行环境，包括硬件和Cloud OS。
- VNF(Virtualized Network Function)：虚拟化后的电信设备。
- EM(Element Management)：电信设备的管理系统，通常不同的电信设备供应商都会提供自己的EM。
- OSS(Operations Support Systems)：电信运营的管理系统，通常每个电信运营商会有自己定制的OSS。

参考架构的右侧是对NFV进行操作维护和管理的部分，也就是MANO，包括如下几点。

- VIM：管理NFVI中的计算资源、存储资源和网络资源。通常，一个VIM管理一个基础设施域(例如一个数据中心)中的资源，既包括虚拟

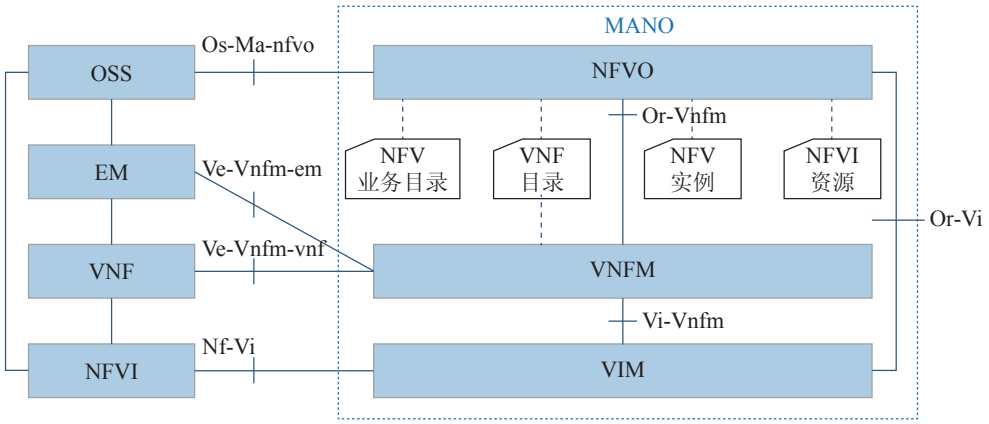


图12-6　ETSI NFV参考架构

资源，也包括硬件设备。

✍ VNFM：对VNF的生命周期进行管理，包括对VNF的实例化、修改、终止、升级、弹性扩缩容、自愈、故障管理和性能管理等。

✍ NFVO：包括两个主要功能。一个是对多个VIM管理的资源进行编排；另一个是对多个VNF编排成网络服务(NS：Network Service)，并对NS的生命周期进行管理。

✍ NS Catalog：存储网络业务的模板，描述网络业务是如何编排的，以及其生命周期是如何管理的。

✍ VNF Catalog：保存VNF的版本包，包括VNF软件和VNF模板。VNF模板用来描述VNF是如何编排的，以及其生命周期如何管理。

✍ NFV Instances：保存所有的VNF实例信息和所有的网络业务实例信息。

✍ NFVI Resources：保存NFVI资源信息，包括可用的、预留的和已分配的资源。

12.3.3　MANO的现状

一、VIM

电信运营商在VIM上的选择已经形成了事实上的标准，绝大部分运营商的选择可以分成两大类：VMware的管理系统VMware vCloud Director，基于OpenStack开发的商用版本，而且更多的运营商选择了OpenStack。

但是在VIM的选择上仍然存在较多的变数，具体如下。

✍ OpenStack本身仍不稳定，新版本和新特性频繁发布。

✍ OpenStack提供的功能与ETSI MANO标准中描述的VIM需求仍有一定的差距，多数厂商都对此进行了增强开发，但是这些增强互不相同，导致了版本的碎片化。

✍ OpenStack只提供虚拟资源的管理，VIM中的硬件管理没有统一的方案。

二、VNFM

VNFM的选择则出现了更多的碎片化，目前可以分成四大类，如图12-7所示。

a) IaaS类的部署工具。一些已经成熟的部署工具，如OpenStack Heat成为部分运营商的选择，它们可以对基本的虚拟资源进行编排和部署，这些工具都已经商用，比较成熟，因此很多简单的应用都可以直接被这些工具管理，减少了集成成本。但是这类工具离MANO的需求和ETSI标准都差距较大，对复杂电信服务的很多商用诉求无法满足。

b) PaaS类的部署工具。一些已经商用的工具可以对服务资源进行编排，构建SOA软件。电信软件的服务化是一个趋势，也是NFV的演进目标，因此一些激进的运营商会尝试使用这类工具。但是很多存量的电信设备还没有完成SOA

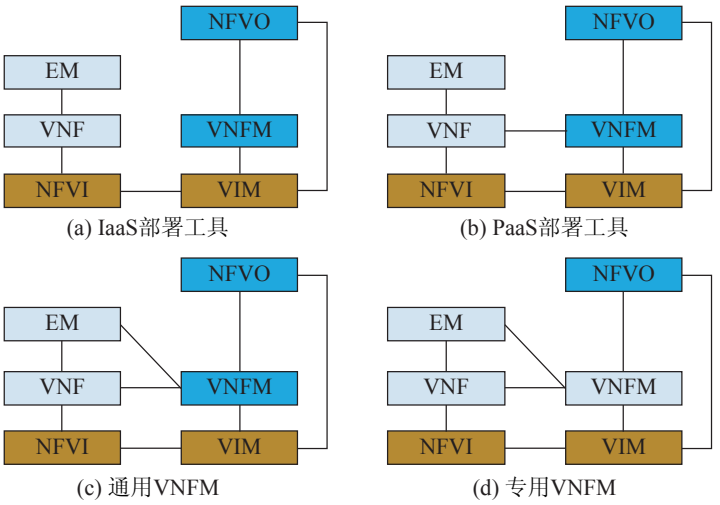


图12-7 VNFM分类

化，对数百万行代码规模的软件进行SOA化架构改造并不是件容易的事，可能需要花费数年时间才能完成，因此这些软件无法通过PaaS方式编排。除此之外，这类工具离MANO的需求和ETSI标准的差距也很大，因此部署复杂的电信服务时也遇到很多困难。

c) 通用VNFM。这类工具参照ETSI标准开发，试图管理所有类型的VNF。OpenStack Taker就是这类工具的代表，很多厂商也声称开发了通用VNFM。这类工具目前遇到的最大问题就是ETSI MANO标准并没有完全确定下来，因此每个厂商都会有自己的理解和扩展。例如VNF和VNFM之间的消息接口，几乎每个厂商的VNFM的定义都不相同。这一事实导致通用VNFM的碎片化最厉害，跨厂商的VNFM和VNF集成需要花费大量的时间和成本。大部分运营商期待在ETSI MANO标准正式发布后这一问题能得到解决，因此看好通用VNFM的未来。

d) 专用VNFM。由于一个厂商的VNF被其他厂商的VNFM进行管理很困难，因此很多电信设备厂商声称，在ETSI MANO标准完全确定前，他们提供的VNF只能被自己提供的VNFM管理，这样可以大大节省集成的成本。但是对电信运营商来说，就意味着他们需要同时使用多个供应商提供的不同VNFM，导致维护成本上升。因此电信运营商普遍不期望使用专用VNFM，或者仅把它们当作是短期的过渡产品。

三、NFVO

大多数运营商引入NFV不仅仅是期待用通用硬件取代专用硬件降低成本，更期待在电信设备虚拟化以后能提高运营商面对市场的反应速度，提高竞争力，以应对OTT的挑战。这些期待是承载在NFVO中的。目前对NFVO的研究很多，其覆盖的范围和能提供的业务能力也不断在扩展。因此不同厂商对NFVO的理解也存在较多差异，具体如下所示。

1. NFVO是一个产品，还是多个产品

NFVO包含了两类主要的功能：对资源的管理和对NS的生命周期管理。因此一些厂商建议将NFVO拆分成两部分，一个叫NSO(NS Orchestrator)管理NS的生命周期，另一个叫RO(Resource Orchestrator)管理资源。这样MANO部分的架构会变成如图12-8所示，每个产品的功能更加明确。

这样的架构也会更适应部分电信运营商的组织结构。在电信运营商中，在每个地区或国家有一个本地的基础设施维护部门，管理本地的RO和

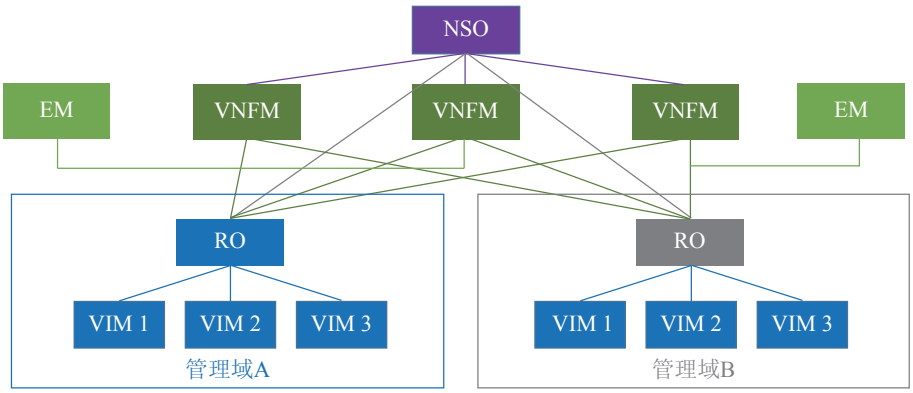


图12-8　NFVO被分拆成NSO和RO以后的架构

VIM；而其他的业务部门则管理NSO和VNFM。

但这样带来的问题是由于功能比较分散，一些集成性的功能难以实现，例如故障的关联性分析、提升业务SLA的资源分配算法分析等。

2. 根据业务特性的部署

每种业务类型对业务服务水平的要求不一样，因此对于部署的区域要求也不一样。例如：对于在线游戏，时延是一个非常敏感的因素，因此服务器必须部署在消费者所在的区域，在每个国家都会部署本地的服务；而企业应用为了给跨国企业提供多个分部之间的交流功能，需要跨国家进行部署(见图12-9)。

3. 业务的多级管理

由于NS业务存在级联的情况，因此部分厂商建议MANO也可以划分成多个管理域进行级联，每个管理域中的设施给对应的管理部门使用。例如图12-10中管理域A中提供了IMS业务，而上级管理域可以提供VoLTE或者CPM业务。

也有部分厂商认为这样的多级管理可以在MANO中提供多租户能力支持，并不需要建设多级服务。

四、数据仓库

NS Catalog、VNF Catalog、NFV Instances和NFVI Resources都是数据库。它们所处的位置在ETSI标准中并没有明确规定，可能是独立的数据库，也可能作为NFVO的一部分。它们可能集中部署，也可能是分布式部署，例如VNF Catalog就可能同时部署在NFVO和VNFM中。

VNF Catalog中存储的VNFD(VNF Descriptor)是VNFM和VNF之间耦合的重要因素，它的定义是跨厂商的VNFM和VNF之间能否集成的关键。因

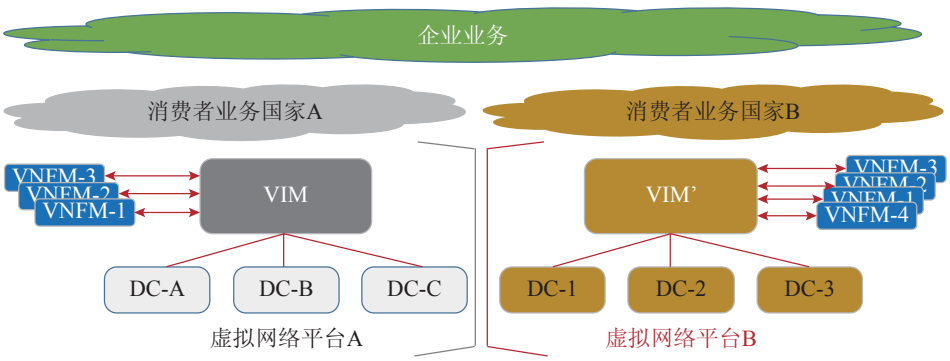


图12-9　不同业务的部署区域

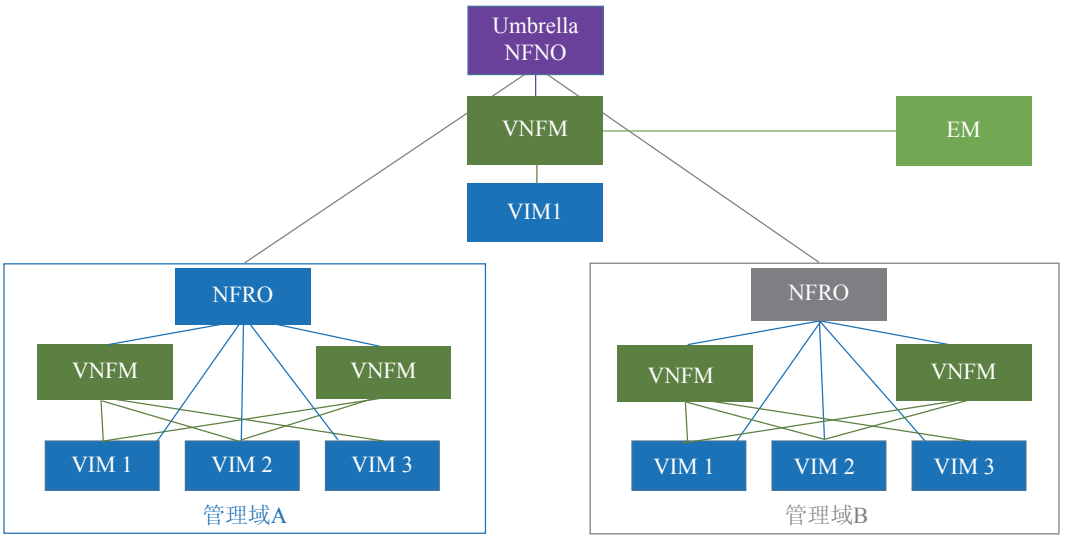


图12-10　多个管理域的网络业务

此VNFD的定义也是当前标准定义和各厂商关注的重点。

12.3.4　MANO的演进中的问题

尽管电信运营商已经开始对NFV和MANO系统进行商用或试商用，但是MANO系统仍有不少问题未完全解决，需要更深入的研究。

一、MANO和OSS的关系

在传统的电信系统里，对各种电信设备的运维是通过OSS进行的。NFV引入云计算技术后，为了尽量减少对已经存在的OSS/EM系统的变更，在架构中新引入了MANO的概念，用于处理云计算新产生的运维功能。

但是端到端的运维体系必然涉及传统业务操作和云计算新运维操作之间的综合，例如：

- 电信业务话务量增加引起NS资源扩容；
- 自动扩缩容时NFVO的自动策略和OSS体系中的业务自动配置之间的关联；
- 硬件故障、虚拟化资源故障和业务故障之间的跨层故障处理；
- 资源性能统计指标和业务性能统计指标之间的关联处理。

这些综合分析必然引起OSS和MANO，尤其是NFVO之间的融合，以至于部分电信运营商认为未来OSS和NFVO会成为一体。

二、MANO与SDN的关系

在很多场景中NFV和SDN之间存在关联(见图12-11)。

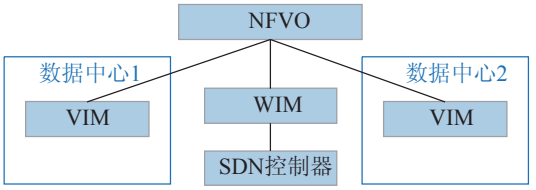


图12-11　跨数据中心的资源编排

1. 跨DC的资源编排

NFVO可以对多个数据中心的资源进行编排，而VIM通常只管理一个数据中心的资源。当一个业务部署在多个数据中心时，NFVO除了通过每个数据中心的VIM对该数据中心的资源进行编排外，还需要通过WIM对数据中心之间的网络资源进行编排，使得部署在各数据中心的业务部分之间能相互连通。

2. 通过SDN实现业务流转发

ETSI的MANO中还定义了一个VNFFG (VNF Forward Graph)的概念。如图12-12所示，在一个NS中有一个作为业务流分发的VNF，它完成两项

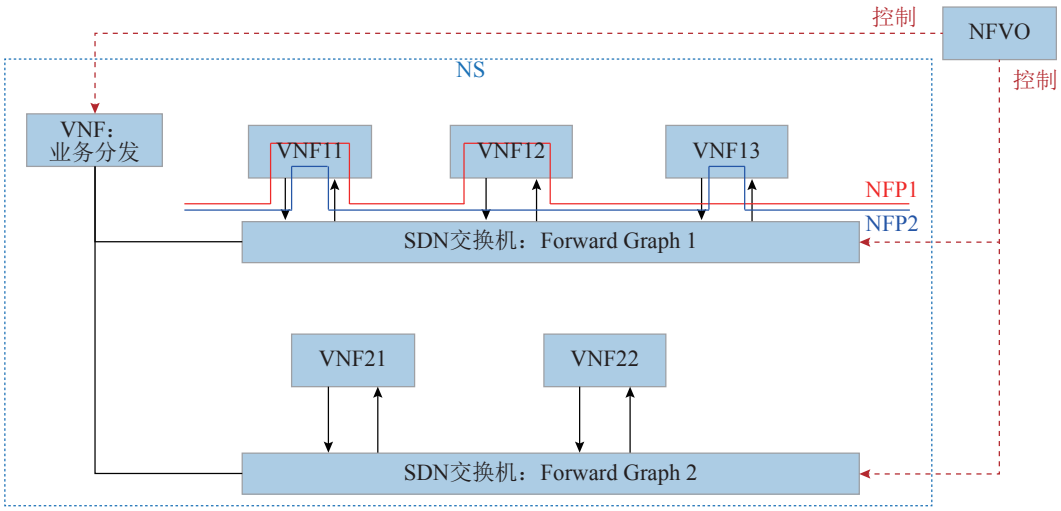


图12-12　VNFFG

任务。

(1) 区分业务流的类型，例如HTTP业务发送到SDN交换机1，视频业务发送到SDN交换机2。

(2) 识别使用该业务流的用户类型，例如VIP用户在数据流中打标签1， 普通用户在数据流中打标签2。每一种标签称为一种网络转发路径(NFP： Network Forwarding Path)。

SDN交换机根据标签将数据流转发到不同的VNF处理，例如SDN1将标签1的数据流转发到VNF11和VNF12处理，将标签2的数据流转发到VNF11和VNF13处理。

每种VNF进行一类功能处理，例如VNF11进行数据缓存，VNF12进行数据加密，等等。

在传统的电信设备中，每个VNF要知道它发出的数据要转发到哪里。这样新增一种业务(FG)或者新增一类用户(NFP)，相关的VNF都要进行升级才能支持。

如果通过SDN控制，那么VNF就不需要关心它自己处在哪种数据流中了。无论是业务(FG)变更还是用户(NFP)变更，都只需要在NFVO上进行编排，而无须任何VNF升级。

在这些场景中，NFVO需要控制SDN路由器/交换机。但是SDN控制器目前只有南向接口比较成熟，北向接口一直没有形成标准。这就对NFVO功能的实现造成了较大的障碍。

三、PNF与VNF的统一管理

尽管电信运营商都对NFV的引入抱有很大的期待，但是在网络上已经存在的大量物理设备如何得到保护也是他们关注的重要内容。因此如下的问题也是值得研究的：

- PNF和VNF是否能被同一个EM所管理？
- VNF在新引入了虚拟化资源的概念以后，其管理对象模型，尤其是告警管理模型是否仍能和PNF很好地兼容？
- NFVO在业务编排时是否能同时编排VNF和PNF？
- VNF和PNF之间的网络如何连接？
- PNF在动态扩缩容上的劣势能否被一起编排的VNF所弥补？

四、MANO对电信运营方式的改变

NFV和MANO的引入不仅仅是简单的设备成本降低和运维成本降低，还很可能对电信运营方式造成很大的影响。

- 传统的电信设备采购由于涉及硬件设备的运输、安装，周期往往长达数月。在引入NFV

以后，数据中心硬件设备作为资源池已经事先采购和安装，电信设备的采购变成软件采购，购买和安装的周期可以缩短成几天甚至几小时。这样电信运营商的运维计划和市场营销计划就可能从每个季度一次，变成几周一次，甚至几天一次，以便更好地响应市场变化，提高竞争力。这不仅仅涉及电信设备和技术的变化，更需要企业管理和运营流程的变革。为了支撑这种变革，就需要能将企业的网络规划/优化系统和MANO无缝地对接起来。

✍ 引入MANO后，不只是新部署一个电信业务更容易，而且删除一个电信业务也更容易。传统的电信业务退网，涉及很多硬件的报废和企业会计成本的调整，一个不合适的业务上线后，如果由于市场需求不多很快退出，可能导致巨额的浪费，因此每个电信业务的上线都需经过严密的分析。引入NFV后，如果按使用时间对采购的VNF业务付费，发现新上线的业务达不到预期而尽快终止的话，成本会很低。这样电信运营商可以采用更激进的方式，通过更频繁的试错挖掘有潜力的业务，从而提高竞争力。这样就需要MANO系统能支持在线采购、在线安装、按时计费、及时退网等一系列功能来端到端支撑运营商的流程变革。

12.4 面向IT应用的IT编排自动化——Heat & TOSCA

12.4.1 OpenStack Heat 介绍

Heat是一个基于预先定义的模板编排和运行复合云应用的服务。Heat目前支持自有的HOT(Heat Orchestration Template)模板格式，也支持亚马逊的CloudFormation模板格式。通过模板，可以简化部署和管理各种基础设施、服务和应用等的复杂度。

Heat模板支持丰富的资源类型，不仅涵盖了常见的各种资源，例如虚拟主机(Server)、虚拟磁盘(Volume)、镜像(Image)、网络(Subnet)、安全组(Security Group)、弹性IP(Floating IP)、账户(User)等，还支持弹性伸缩(AutoScaling)、Sahara集群、Trove实例等高级资源。另外，Heat支持资源插件扩展机制(Resource Plugins)，用户可以根据应用需要提供自己的资源插件，并根据需要实现对应的资源管理。

12.4.2 Heat编排

OpenStack最初就提供了API、命令行和Horizon来供用户管理资源。然而，依靠一行行的命令执行或者在浏览器上点击来管理资源，相当费时费力；即使把命令行保存在脚本执行，或者编写程序直接调用REST API，在输入输出、相互依赖之间都需要大量的脚本、代码来协调控制，引入了额外的复杂性，也不易于扩展。

Heat在这种情况下应运而生。Heat采用了业界流行使用的模板方式来设计或者定义编排。用户只需要通过文本编辑器，编写一段基于 Key-Value 的模板，就能够方便地定义编排。为了方便用户的使用，Heat 还提供了大量的模板样例，大多数的时候用户只需要选择想要的模板样例片段，通过“拷贝-粘贴”的方式就可以完成模板的编写。

Heat从四个方面来支持编排。

(1) OpenStack 自己提供基础架构资源，包括计算、网络和存储等资源。通过编排这些资源，用户可以得到最基本的VM。在编排VM的过程中，用户可以通过定义一些简单的脚本，对VM做一些简单的配置(例如修改Hostname、修改密码等)。

(2) 用户可以通过Heat提供的Software Configuration和Software Deployment等对VM进行复杂的配置，比如安装和配置软件。

(3) 如果用户有高级的功能需求，例如需要能够根据负荷自动伸缩的 VM 组，或者需要一组负载均衡的 VM，Heat 提供了 AutoScaling 和 Load Balance 等进行支持。

(4) 如果用户的应用足够复杂，或者说用户的应用已经有了一些基于流行配置管理工具的部署，比如说已经基于Chef的Cookbook或者基于Puppet的配置文件，那么可以通过集成Chef或者

Puppet 来复用这些资源，这样就可以节省大量的开发时间或者是迁移时间。

12.4.3 Heat架构

Heat服务包括以下重要组件。

1. Heat-api

Heat-api组件提供了OpenStack原生的REST-API。该组件会把API请求通过RPC发送给heat-engine处理。

2. Heat-api-cfn

Heat-api-cfn组件提供了兼容亚马逊CloudFormation的QueryAPI，并把API请求通过RPC发送给heat-engine处理。

3. Heat-engine

Heat-engine是Heat的大脑，提供Heat最主要的编排功能。Heat-engine负责解析模板，建立构成应用的各资源的依赖关系，被按照依赖关系创建和运行应用。

用户可以在Horizon(OpenStack提供的页面服务)或者命令行中提交包含模板和参数的请求，Horizon或者命令行工具(Heat-cli)会把请求转化为REST风格的API调用，然后调用Heat-api。Heat-api验证模板通过后，通过RPC将请求传递给Heat-engine处理。或者，用户通过工具发送Query请求给Heat-api-cfn，Heat-api-cfn验证通过后将请求转递给Heat-engine处理

Heat-engine拿到请求后，根据模板中定义的各种类型的资源和请求中指定的参数，调用OpenStack其他服务的API，将资源请求发给OpenStack其他服务。通过和OpenStack其他服务的协作，最终完成请求的处理。

该次请求成功创建的对象和资源在OpenStack heat中被封装为一个Stack实例进行管理，如图12-13所示。

Heat-engine在这里的作用分为三层： 第一层处理Heat层面的请求，就是根据模板和输入参数来创建Stack，这里的Stack是由各种资源组合而成。第二层解析Stack里各种资源的依赖关系，Stack和嵌套Stack的关系。第三层就是根据解析出来的关系，依次调用各种服务的API来创建各种资源。

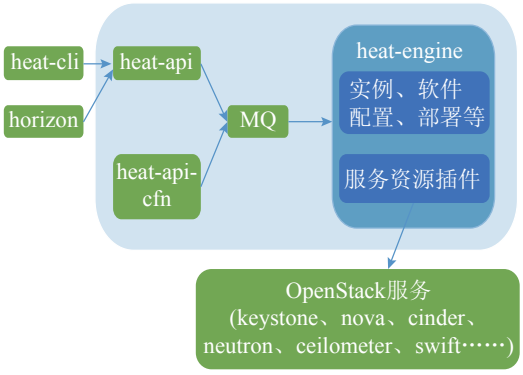


图12-13 Heat架构

一个简单Heat模板如图12-14所示。

```
heat_template_version: 2015-10-15

description: Sample Nova Host Aggregate template

parameters:
  host_aggregate_name:
    type: string
    description: Nova host aggregate name
  availability_zone_name:
    type: string
    description: Nova availability zone name
  hosts:
    type: comma_delimited_list
    description: Nova host name list
  metadata:
    type: json
    description: Arbitrary key/value metadata

resources:
  sample_host_aggregate:
    type: OS::Nova::HostAggregate
    properties:
      name: {get_param: host_aggregate_name}
      availability_zone: {get_param: availability_zone_name}
      hosts: {get_param: hosts}
      metadata: {get_param: metadata}

outputs:
  sample_host_aggregate_id:
    value: {get_resource: sample_host_aggregate}
    description: Sample nova host_aggregate
```

图12-14 Heat模板

✍ Stack(栈)： 在Heat，Stack是由Heat创建的多个对象或者资源的集合。它包含实例(虚拟

机)、网络、子网、路由、端口、路由端口、安全组(Security Group)、全组规则、自动伸缩等。

- Template(模板)：Heat使用Template的概念来定义一个Stack。例如，上面的示例模板就定义了通过指定参数(主机集合名以及主机列表)创建一个主机集合(Host Aggregate)。
- Parameters(参数)：参数包含一些基本信息，比如具体的镜像ID，或者特定网络ID。他们将由用户输入给Template。这种参数机制允许用户创建一个一般的Template，它可能潜在使用不同的具体资源。
- Resources(资源)：Resource就是由Heat创建或者修改的具体的资源。
- Output(输出)：模板被执行后的结果(输出)。它是通过OpenStack Dashboard或者Heat stack-list/stack-show命令来显示给用户。在最新的OpenStack版本，还可以通过output-list/ output-show查询输出结果。
- HOT：Heat Orchestration Template的缩写，是Heat Template使用的两种格式的一种。HOT并不与AWS CloudFormation Template格式兼容，只能被OpenStack使用。HOT格式的Template，可以使用YAML和JSON格式。
- CFN：AWS CloudFormation 模板格式；Heat兼容亚马逊CloudFormation格式的模板，方便用户执行亚马逊上已经存在的模板。CFN格式的Template通常使用JSON。

12.5 TOSCA(云应用的拓扑编排标准)

TOSCA(云应用程序的拓扑结构和业务流程规范，英文为Topology and Orchestration Specification for Cloud Applications) 是OASIS组织(XML标准的制定者)管理的标准。TOSCA标准的目标是为了提高云计算应用程序和云服务的可移植性。它确保云应用程序的描述和基础设施云服务的互操作性，以及云服务各个部分之间的关系和这些服务的操作行为(例如部署、补丁、关闭)与具体的云计算服务的提供商或托管技术无关。

TOSCA规范定义了针对IT服务的元模型(metamodel)。元模型定义了服务的结构以及如何管理服务。拓扑模板(有时也被叫做服务拓扑模型)定义了服务的结构；计划定义了如何创建、终止服务，以及在整个生命周期中如何管理服务的流程模型，如图12-15所示。

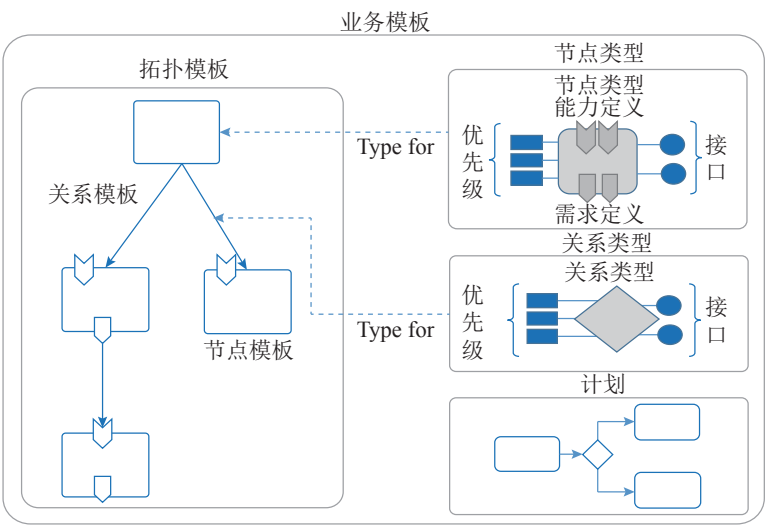


图12-15 TOSCA业务模板

TOSCA主要有两个基本概念：节点(Node)和关系(Relationship)。节点是定义的资源和对象模型，可以是物理资源(例如服务器、交换机等)，可以是虚拟资源(例如虚拟机、虚拟网络等)，或者是应用或者服务等软件。关系定义了这些节点是如何连接的，例如：一个Web服务应用(Node)部署在(关系)多个虚拟机(Node)，这些虚拟机使用(关系)同一个子网(Node)。节点可以定义属性和操作；节点和关系都支持扩展定义。

TOSCA模板目前支持YAML和XML两种格式。

目前，TOSCA的实现包括OpenStack Heat、OpenStack Tacker、Cloudify等。另外，OpenStack社区还发起了Heat-Translator项目，希望能够把TOCSA模板转成Heat的HOT模板。

例如，如果我们想定义一个有一个磁盘(BlockStorage)的计算节点(Compute)，如图12-16所示。

其对应的TOSCA模板为：

```
tosca_definitions_version： tosca_simple_yaml_1_0
description： >
 TOSCA simple profile with server and attached block storage using the normative AttachesTo Relationship Type.
topology_template：
  inputs：
    cpus：
      type： integer
       description： Number of CPUs for the server.
      constraints：
       - valid_values： [ 1， 2， 4， 8 ]
    storage_size：
      type： scalar-unit.size
       description： Size of the storage to be created.
      default： 1 GB
    storage_snapshot_id：
      type： string
      description： >
          Optional identifier for an existing snapshot to use when creating storage.
    storage_location：
      type： string
          description： Block storage mount point (filesystem path).
  node_templates：
```

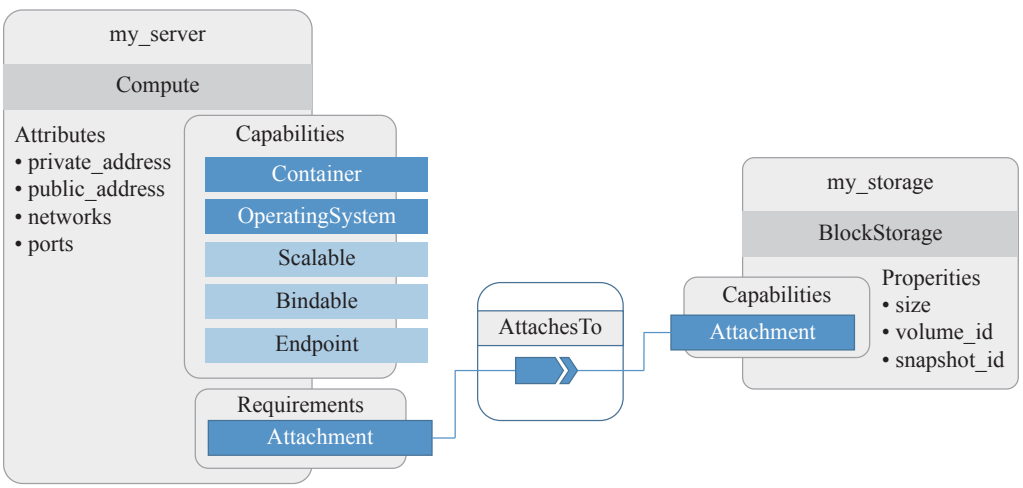


图12-16　有一个磁盘的计算节点

```
        my_server:
          type: Compute
            capabilities:
              host:
                properties:
                  disk_size: 10 GB
                      num_cpus: { get_input:
cpus }
                  mem_size: 1 GB
            os:
              properties:
                architecture: x86_64
                type: linux
                distribution: fedora
                version: 18.0
        requirements:
          - local_storage:
            node: my_storage
            relationship:
              type: AttachesTo
                properties:
                  location: { get_input: storage_
location }
        my_storage:
                type: BlockStorage
                properties:
                  size: { get_input: storage_size }
                  snapshot_id: { get_input: storage_
snapshot_id }
      outputs:
        private_ip:
          description: The private IP address of the
newly created compute instance.
              value: { get_attribute: [my_server,
private_address] }
        volume_id:
          description: The volume id of the block
storage instance.
              value: { get_attribute: [my_storage,
volume_id] }
```

input和output分别定义了输入和输出。例如，输入的CPU的值只能是1、2、4 或者8；而输出的private_ip为创建的计算节点的IP地址。“my_server”代表需要生成一个类型为Compute的资源； requirement标识这个资源需要关联一个名为“my_storage”的节点，且关联的方式为“attach”。

第 4 章

面向计算资源共享最大化和管理自动化的软件定义计算

4.1 XEN/KVM虚拟化引擎

4.1.1 虚拟化架构分类

计算虚拟化技术的实现形式是在系统中加入一个虚拟化层，将下层的资源抽象成另一种形式的资源，供上层使用。

计算虚拟化技术的通用实现方案是将软件和硬件相互分离，在操作系统与硬件之间加入一个虚拟化软件层，通过空间上的分割、时间上的分时以及模拟，将服务器物理资源抽象成逻辑资源，向上层操作系统提供一个与它原先期待一致的服务器硬件环境，使得上层操作系统可以直接运行在虚拟环境上，并允许具有不同操作系统的多个虚拟机相互隔离，并发运行在同一台物理机上，从而提供更高的IT资源利用率和灵活性。

计算虚拟化的虚拟化软件层需要模拟出来的逻辑功能主要为高效、独立的虚拟计算机系统，我们称之为虚拟机，在虚拟机中运行的操作系统软件，我们称之为Guest OS。

计算虚拟化技术可以将单个CPU模拟为多个CPU，允许一个平台同时运行多个操作系统，并且应用程序可以在相互独立的空间内运行而互不影响。简单地说，计算虚拟化技术实现了计算单元的模拟和模拟出来的计算单元间的隔离。

虚拟化软件层模拟出来的每台虚拟机都是一个完整的系统，它具有处理器、内存、网络设备、存储设备和BIOS，因此虚拟机中运行的操作系统和应用程序与在物理服务器上运行的操作系统和应用程序并没有本质的区别。

计算虚拟化的这个软件层，也就是虚拟机监控器(Virtual Machine Monitor，VMM)，通常被称为Hypervisor。常见的Hypervisor软件栈架构方案分为两类，即Type-I型和Type-II型。

Type-I型(裸金属型)指VMM直接运行在裸机上，使用和管理底层的硬件资源，GuestOS对真实硬件资源的访问都要通过VMM来完成，作为底层硬件的直接操作者，VMM拥有硬件的驱动程序。Type-II型(宿主型)指VMM之下还有一层宿主操作系统，由于Guest OS对硬件的访问必须经过宿主操作系统，因而带来了额外的性能开销，但可充分利用宿主操作系统提供的设备驱动和底层服务来进行内存管理、进程调度和资源管理等。

我们进一步分析Hypervisor对于CPU指令的模拟和虚拟实例的隔离方式，计算虚拟化技术可以细分为如下几个子类。

一、全虚拟化(Full Virtualization)

全虚拟化是指虚拟机模拟了完整的底层硬件，包括处理器、物理内存、时钟、外设等，使得为原始硬件设计的操作系统或其他系统软件完全不做任何修改就可以在虚拟机中运行。操作系统与真实硬件之间的交互可以看成是通过一个预先规定的硬件接口进行的。全虚拟化VMM以完整模拟硬件的方式提供全部接口(同时还必须模拟特权指令的执行过程)。举例而言，x86体系结构中，对于操作系统切换进程页表的操作，真实硬件通过提供一个特权CR3寄存器来实现该接口，操作系统只需执行“mov pgtable，%%cr3”汇编指令即可。全虚拟化 VMM 必须完整地模拟该接口执行的全过程。如果硬件不提供虚拟化的特殊支持，那么这个模拟过程将会十分复杂。一般而言，VMM必须运行在最高优先级来完全控制主机系统，而Guest OS需要降级运行，从而不能执行特权操作。当Guest OS执行前面的特权汇编指令时，主机系统会产生异常(General Protection Exception)，执行控制权将重新从Guest OS转到VMM手中。VMM事先分配一个变量作为影子CR3寄存器给Guest OS，将pgtable代表的客户机物理地址(Guest Physical Address)填入影子CR3寄存器，然后VMM需要将pgtable翻译成主机物理地址(Host Physical Address)并填入物理CR3寄存器，最后返回到Guest OS中。随后VMM还将处理复杂的Guest OS缺页异常(Page Fault)。比较著名的全虚拟化VMM有Microsoft Virtual PC、VMware Workstation、Sun Virtual Box、Parallels Desktop for Mac和QEMU。

二、超虚拟化(Paravirtualization)

这是一种修改Guest OS部分访问特权状态的代码以便直接与VMM交互的技术。在超虚拟化虚拟机中，部分硬件接口以软件的形式提供给客户机操作系统，这可以通过Hypercall(VMM 提供给Guest OS直接调用，与系统调用类似)的方式来提供。例如，GuestOS把切换页表的代码修改为调用Hypercall来直接完成修改影子CR3寄存器和翻译地址的工作。由于不会产生额外的异常和模拟部分硬件执行流程，超虚拟化可以大幅度提高性能，比较著名的 VMM 有 Denali、Xen。

三、硬件辅助虚拟化(Hardware-Assisted Virtualization)

硬件辅助虚拟化是指借助硬件(主要是主机处理器)的支持来实现高效的全虚拟化。例如有了Intel-VT技术的支持，Guest OS和VMM的执行环境自动地完全隔离开来，Guest OS有自己的“全套寄存器”，可以直接运行在最高级别。因此在上面的例子中，Guest OS能够执行修改页表的汇编指令。Intel-VT和AMD-V是目前x86体系结构上可用的两种硬件辅助虚拟化技术。

四、部分虚拟化(Partial Virtualization)

VMM只模拟部分底层硬件，因此客户机操作系统不做修改是无法在虚拟机中运行的，其他程序可能也需要进行修改。在历史上，部分虚拟化是通往全虚拟化道路上的重要里程碑，最早出现在第一代的分时系统CTSS和IBM M44/44X实验性的分页系统中。

五、操作系统级虚拟化(Operating System Level Virtualization)

在传统操作系统中，所有用户的进程本质上是在同一个操作系统的实例中运行的，因此内核或应用程序的缺陷可能会影响其他进程。操作系统级虚拟化是一种在服务器操作系统中使用的轻量级的虚拟化技术，内核通过创建多个虚拟的操作系统实例(内核和库)来隔离不同的进程，不同实例中的进程完全不了解对方的存在。比较著名的虚拟化技术有Solaris Container、FreeBSD Jail和OpenVZ等。

接下来我们对当前比较热门的Xen和KVM这两种虚拟化方案进行详细的介绍。

4.1.2 Xen虚拟化技术

Xen是由剑桥大学计算机实验室开发的一个开源项目，可以直接运行在计算机硬件上，并且可以并发地支持多个客户操作系统。Xen能够支持多种处理器，如x86、x86-64、Power PC和ARM等，所以其可以运行在较多种类的设备上。目前Xen支持的客户操作系统有Linux、NetBSD、FreeBSD、Solaris、Windows和其他一些常用的操作系统。

Xen 包含三种基本组件，分别是Hypervisor、Domain 0和Domain U。

Hypervisor运行在物理硬件之上，承载所有的客户操作系统，主要负责向物理硬件设备上的所有的操作系统提供CPU调度和内存分配等功能。Hypervisor隔离了物理硬件和操作系统，从而提高客户操作系统的安全性。

Domain 0运行在Hypervisor之上，是Xen的管理员，具有直接访问硬件和管理其他客户操作系统的权限。在Domain 0中，存在两个基本的驱动，分别是网络驱动程序和块存储驱动程序。网络驱动程序能够直接与本地的网络硬件进行通信，从而处理来自于客户操作系统的网络请求。类似地，块存储驱动程序能够与本地的存储设备通信，处理客户操作系统的数据读写请求。

Domain U运行于Hypervisor之上，是Xen虚拟环境中的客户虚拟机。Domain U上运行的客户操作系统有两类，一种是半虚拟化客户机，另一种是完全虚拟化客户机。半虚拟化客户机上运行的操作系统，是经过修改之后的。而完全虚拟化客户机上的操作系统，则是未被修改的标准的操作系统。Xen虚拟环境中可以同时运行多个Domain U。

目前Xen支持三种虚拟化方案，分别是超虚拟化、完全虚拟化和IO半虚拟CPU完全虚拟化

方案。

在超虚拟化方案中，客户虚拟机操作系统可以感知自己的运行环境不是物理硬件，而是在Hypervisror之中，同样也可以感知到其他的客户虚拟机。客户虚拟机的操作系统，为了能够调用Hypersivor，需要对操作系统进行专门的修改。把标准的操作系统移植到Xen架构中，才能够运行到客户虚拟机上。

在完全虚拟化中，客户虚拟机操作系统所感知的运行环境则始终是物理硬件，并且无法感知到在相同环境下的其他正在运行的客户虚拟机。所有的客户虚拟机上运行的，都是标准的不需要修改的操作系统。

而第三种方案，则对前两种方案结合起来。在完全虚拟化的客户虚拟机上安装使用特殊的超虚拟化设备驱动。因为超虚拟化的设备驱动具有更好的性能，所以本方案能够使完全虚拟化的客户虚拟机具备更好的性能。

4.1.3 KVM虚拟化技术

KVM(Kernel-based Virtual Machine)是基于Linux内核的开源的虚拟化解决方案。KVM从2.6.20版本开始被合入kernel主分支维护，成为Linux的重要模块之一，截止到目前已经实现了对x86、S390和PowerPC等体系结构的支持。

KVM本身只能够提供CPU虚拟化和内存虚拟化等部分功能，而其他设备的虚拟和虚拟机的管理工作，则需要依靠QEMU来完成。在KVM虚拟化环境中，一个虚拟机就是一个传统的Linux进程，运行在Qemu-KVM进程的地址空间。KVM和QEMU相结合，一起向用户提供完整的平台虚拟化。在KVM虚拟化方案中，通过在Linux内核中增加虚拟机管理模块，直接使用Linux非常成熟和完善的模块和机制，例如内存管理和进程调度等，从而使Linux内核成为能够支持虚拟机运行的Hypervisor。

KVM虚拟化方案中使用了VT-x的VMX(Virtual Machine eXtension)模式。在这种模式下，CPU具有根模式和非根模式两种操作模式，而每种操作模式又分别具有独立的Ring0和Ring3。在KVM虚拟化场景中，KVM主机在根模式下运行，主机的kernel处于Ring0级别，而用户态程序则处于Ring3级别。客户虚拟机运行在非根模式下，其中，kernel运行在非根模式中的Ring0，而其他用户态程序则在Ring3上运行。当处于非根模式中的主机(即客户虚拟机)执行敏感指令时候，会触发VM-Exit，CPU会从非根模式切换到根模式，也就是KVM主机会进行接管，对敏感指令进行一系列的处理。同样也存在从根模式到非根模式的VM-Entry切换，这种情况主要发生在Hypersivor调度启动客户虚拟机的时候。

当在KVM虚拟化环境中创建虚拟机时，首先运行在KVM主机用户态的Qemu-kvm会调用ioctl通过/dev/kvm设备创建虚拟机和虚拟CPU。Linux内核的KVM模块会创建和初始化相关的数据。之后会运行用户态的qemu-kvm，通过ioctl启动运行vCPU，之后内核会启动虚拟机，并且通过VM-Entry进入到客户虚拟机操作系统。在客户虚拟机中，操作系统会执行虚拟机的代码，如果是非敏感指令，可以直接在物理CPU上运行。如果执行到了敏感指令或发生异常时，就会触发VM-Exit并记录相关的信息，从而使CPU切换到根模式下，由Hypervisor来进行进一步的处理，处理完成后会触发VM-Entry进入到客户虚拟机操作系统中继续执行其他指令。

Xen和KVM都是当前非常热门的虚拟化技术，且具有各自的特点。Xen是较早出现的虚拟化技术，具有广泛的管理工具和较好的性能，同时支持半虚拟化和完全虚拟化，实现了对大多数CPU的支持。KVM是一个轻量级的虚拟化管理程序模块，来自于Linux内核，具有较好的性能和实施的简易性，以及对Linux的持续支持，但是目前只支持具有虚拟化功能的CPU。

4.2 基于OpenStack Nova的计算资源池调度算法

在OpenStack Nova中关于计算资源的调度

问题，是在Nova-Scheduler模块中实现的。在OpenStack中，Filter Scheduler模块的作用就是决策虚拟机创建在哪个主机上，调度仅支持计算节点。通过分析源代码，我们可以看到调度器有以下定制的方法(见图4-1)。

第一种扩展方法是不用默认的filter_scheduler.py的class FilterScheduler，自己完全可以从driver.py/class Scheduler继承并实现自己的类。

第二种扩展方法是不使用默认的物理主机状态管理类host_manager.py/ classHostManager，自己从HostManager继承或者重载实现xxxHostManager类进行物理主机状态的管理。

第三种方法是提供自己的调度过滤器或者权重过滤器插件，参考affinity_filter.py或ram.py实现自己的插件，在部署的时候配置为自己的插件即可。除了上面的三种方法，调度器还在各种地方提供了扩展性。

例如第四种方法，其在物理主机中定义一个主机组Group，然后在Scheduler Hint中带上这个Group，也可以实现可定制的调度策略。

第五种方法是在Scheduler Hint中增加自定义的调度条件，在调度的时候通过自己的Filter插件增加调度能力。

第六种方法是在创建虚拟机请求的AZ参数中通过“AZ：host”的方式指定主机进行虚拟机的创建。

第七种方法是调度器始终监控一个调度配置json文件，如果这个文件发生变化，则动态加载这个调度配置文件，改变和调整调度策略，因此可以动态修改这个json文件修改调度策略(见图4-2)。

基于Scheduler的调度机制分为两部分，主机过滤(Filtering)和权值计算(Weights)。

4.2.1 主机过滤

如图4-3所示，在主机过滤过程中，过滤调度器首先生成一个字典包含没有过滤的主机集，经过若干过滤属性机制生成主机集，再经过权重机制，选择开销最小的若干主机，最终得到请求的实例数主机集。对于特定的过滤机制，会产生特

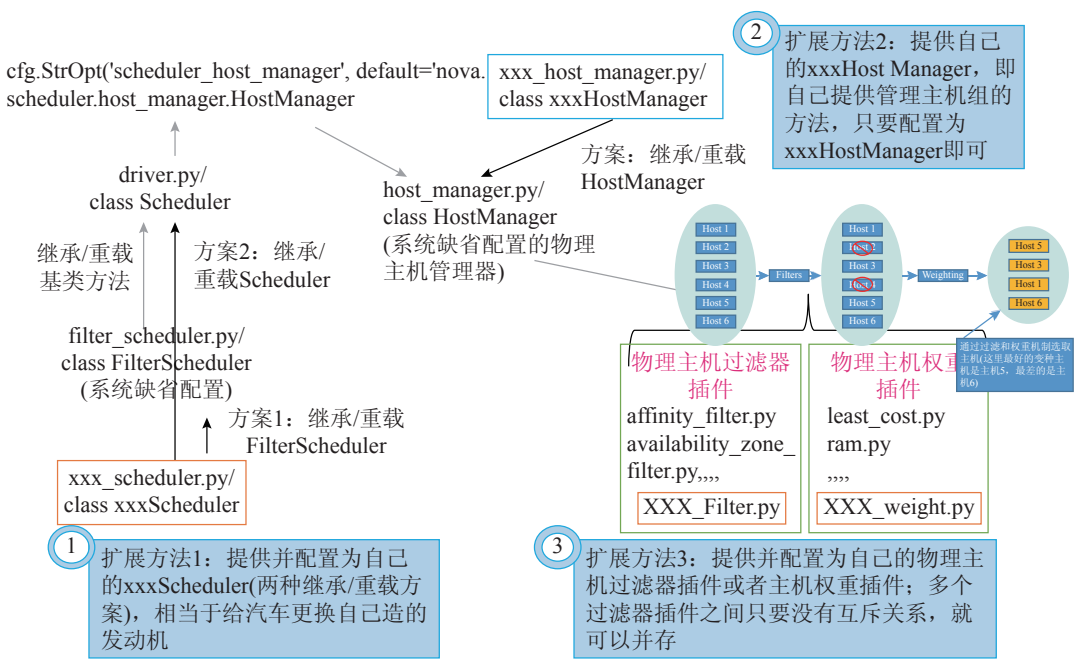


图4-1 Nova Scheduler的可扩展机制1

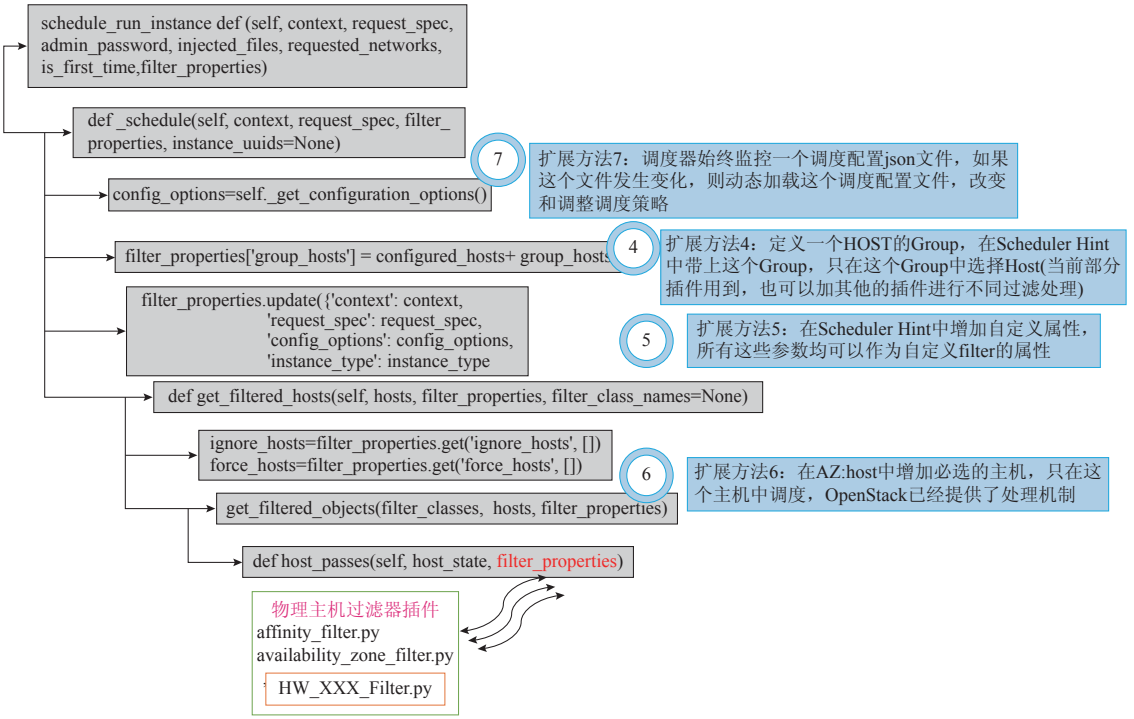


图4-2　Nova Scheduler的可扩展机制2

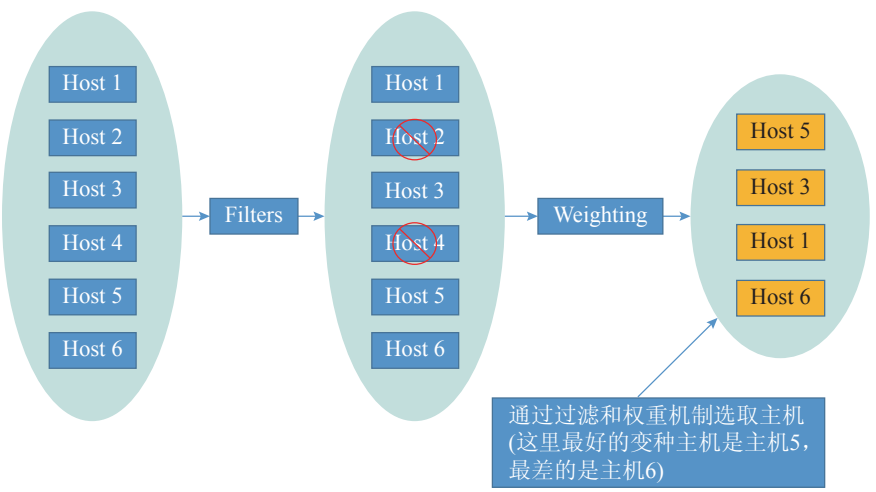


图4-3　Nova Scheduler机制

定的主机集。

如果经过上述调度机制没有选择出主机，那么意味着没有合适的主机可供调度。过滤机制非常灵活的支持一系列过滤函数以及权重策略。若认为过滤机制，实现不够有效率，还可以实现自己的过滤算法。

在nova.scheduler.filters中，有一些标准的过滤函数，如表4-1所示。

表4-1　各插件的功能和使用方法

插件名称	功能	使用方法
Core Filter	设置主机能够分配虚拟机的vCPU/pCPU的比例，超过比例就不能分配虚拟机了；允许设置为超分配	全局配置数据
Aggregate Core Filter	针对某个Host Aggregate单独设置主机能够分配虚拟机的vCPU/pCPU的比例，超过比例就不能分配虚拟机了；没有设置，默认就用全局的CoreFilter配置数据	针对HostAggregate设置一个metadata(Key，Value)，比如设置cpu_allocation_ratio=10
Aggregate Instance ExtraSpecs Filter	在指定的HostAggregate中选定一个主机分配虚拟机	需要在HostAggregate打上flavor的对应标签，就是HostAggreagate和flavor要设置相同的metadata(key，value)，比如两个都增加storage = DSWare的key-valuemetadata，则表明只在存储为Dsware的主机分配虚拟机规格为flavor的虚拟机
Aggregate Multi Tenancy Isolation	在指定的HostAggregate中只允许给某个租户分配虚拟机	HostAggregate增加一个metadata(key，value)，filter_tenant_id = xxx
RamFilter	和CoreFilter类似，只不过针对的是内存	参考CoreFilter
Aggregate Ram Filter	和AggregateCoreFilter类似，只不过针对的是内存	参考AggregateCoreFilter
AllHosts Filter	允许所有的HOST参加分配	
Availability Zone Filter	只允许在创建虚拟机参数中带的Availability Zone内的那些Host上分配虚拟机	
Compute Capabilities Filter	用于过滤满足Flavor的extra specs条件的主机	如AggregateInstanceExtraSpecsFilter就要用到extra_specs
Compute Filter	所有能用能操作的主机都可以分配虚拟机	默认需要配置该过滤器
Disk Filter	根据磁盘分配比例过滤能够分配的虚拟机，超过比例就不能分配了(允许配置为超分配)	在nova.conf中配置disk_alloc ation_ratio=1.0
Different Host Filter	指定不要和某些虚拟机实例共享主机，可以用于互斥的分配场景	在Scheduler hint中增加一个hint‘os：scheduler_hints’:{‘different_host’：[‘vm1’，‘vm2’]}
Group Anti Affinity Filter	指定不要在某个组内的所有HOST上分配虚拟机，可以用于互斥的分配场景	在Scheduler hint中增加一个hint ‘os：scheduler_hints’：{‘group’：[‘host1’，‘host2’]

(续表)

插件名称	功能	使用方法
Image Properties Filter	根据镜像属性过滤主机，主要的属性包括CPU架构、Hypervisor的类型、VM_Mode[Hypervisorapplication binary interface (ABI)]等	glance image-update img-uuid——propertyarchitecture=arm——property Hypervisor_type=qemu
Isolated Hosts Filter	在管理员指定的一些孤立的主机中创建特定镜像的虚拟机	需要在nova.conf配置isolated_hosts=server1，server2isolated_images=342b492c-128f-4a42-8d3ac5088cf27d13，ebd267a6-ca86-4d6c-9a0ebd132d6b7d09
Json Filter	在Schedluer hint中带上json脚本，选出满足脚本运算规则的主机分配虚拟机	如设置这样的运算条件os：scheduler_hints'：{ 'query'：'[">="，"$free_ram_mb"，1024]
Retry Filter	不再对已经尝试分配虚拟机但是失败的主机再次进行选择，只有当nova.conf中允许调度失败、重新调度的配置有效时，这个指标才起作用	scheduler_max_attempts>0
Same Host Filter	在指定虚拟机所在主机上分配虚拟机	scheduler hint中加上same_host的设置'os：scheduler_hints'：{ 'same_host'：['a0cf03a5-d921-4877-bb5c-86d26cf818e1'，'8c19174f-4220-44f0-824a-cd1eeef10287']，}
Simple CIDR Affinity Filter	根据某个子网IP地址范围内的主机进行虚拟机分配	如下为在192.18.1.1的子网分配虚拟机'os：scheduler_hints'：{'build_near_host_ip'：'192.168.1.1'，'cidr'：'24'}

很多过滤机制使用的数据都是通过scheduler_hints，其通常定义在用户初始建立实例的过程中，唯独不包括JsonFilter过滤器。

对于自定义的过滤器，必须继承于BaseHost Filter类，实现host_passes方法，如果通过这个方法的逻辑，则返回true。

举一个例子，若nova.conf文件中包含下面配置：

- scheduler_driver=nova.scheduler.Filter Scheduler
- scheduler_available_filters=nova.scheduler.filters.all_filters
- scheduler_available_filters=myfilter.MyFilter
- scheduler_default_filters=RamFilter，ComputeFilter，MyFilter

对于上面配置，nova会使用FilterScheduler作为调度器的驱动。标准的过滤函数以及自定义的MyFilter函数可作为可用的过滤器，RamFilter、ComputeFilter、MyFilter可作为默认的过滤器。

不同的过滤函数会有不同的实现方式，那么也决定了不同的开销。在scheduler_default_filters中的顺序，会影响调度器的性能。一般建议是尽快过滤掉无效主机，以避免不必要的开销。我们可以通过其开销倒序排序scheduler_default_filters中的顺序。

在大中型环境中，通常利用Availability ZoneFilter在其他资源计算型过滤函数前，会非常有用。

4.2.2 权值计算

权值计算，是一种能够在过滤集中找出最佳的主机的一种机制。为了能够控制不同的权值因素，使用权值系数。因此，最终的公式可以定义为：

weight = w1_multiplier * norm(w1) + w2_

multiplier * norm(w2) + ...

权值计算是weights.BaseHostWeigher类的子类，可以实现weight_multiplier 和 _weight_object方法，也可以只实现_weight_object方法。

对于权值计算部分，在配置文件nova.scheduler.weights.all_weighers中，默认的基于scheduler_weight_classes类，分别包括下列权值计算函数。

- RAMWeight：通过可用的RAM大小决定计算节点。
- DiskWeight：通过可用的磁盘大小进行权值分配，最大的优先级最高。
- MetricsWeigher：通过主机的不同参数度量决定主机的权值。在配置文件中指定相关参数：

metrics_weight_setting = name1=1.0，name2=-1.0

- IoOpsWeigher：通过主机的负载进行权重计算，默认选择负载最小的主机(见图4-4)。
- Server Group Soft Affinity Weigher：通过同一个主机组中运行的实例数进行权重计算。

Filter Scheduler模块通过一系列的主机过滤以及权值计算，获得一个可以接受的主机集合。

最后被选择的主机集合，通过最终的权值计算进行排序，并提供主机集与准备运行的实例。

4.3 计算高可靠性保障

在云计算的条件下，业务运行所需要的资源是通过软件模拟或者软件管理分配的方式提供的，也就是说在业务运行负载和物理硬件之间引入了Hypervisor作为管理层，保证业务运行的可靠性。

在通过虚拟化技术保障云计算条件下的业务运行可靠性方面，当前业界发展出来的技术主要有以下两种。

- 基于冷备机制的虚拟化HA保护，这种方式主要提供了在物理硬件故障的条件下，选择资源池中的其他健康物理主机重新部署虚拟机，并在原有数据不丢失的条件下，尽快恢复虚拟机业务。
- 基于虚拟机热备机制的虚拟机容灾方案，这种方式主要是在不同的物理主机上，提供虚拟机业务运行的容灾，在一台物理主机发生故障的时候，自动由另外一台主机上的虚拟机业务运行镜像接管以进行业务处理，从而保证虚拟机的业务不发生中断。

上述两种方式实现的技术难度和工程化部署的要求限制不一样。其中冷备机制实现技术难度较低，对工程化部署的限制和要求较少，在业界

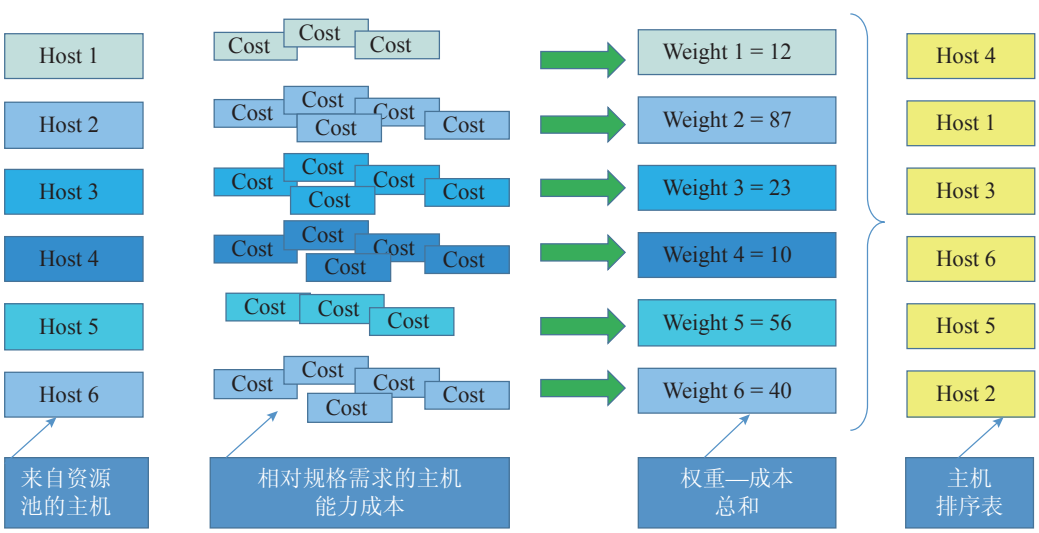


图4-4　权值计算机制

得到广泛的应用。热备机制的实现难度较大，同时，因需要提供虚拟机运行业务镜像冗余，虚拟机的性能会有额外的占用，同时也需要在工程化部署时保证虚拟机的业务组网能够在切换的条件下平滑过渡，对于业务组网的要求较高，所以现阶段的应用还处于逐步走向成熟的过程。

4.3.1 基于冷备机制的虚拟机HA保护

当物理服务器宕机或者重启时，系统可以将具有HA属性的故障虚拟机迁移到其他物理服务器，保证虚拟机快速恢复。

由于单个集群内可以运行上千个虚拟机，当某个或某些服务器宕机后，为避免大量虚拟机迁移造成网络拥塞和目的服务器过载，系统会根据网络流量、目的服务器负荷选择将虚拟机迁移到不同的目的服务器。

当虚拟资源管理器与物理服务器上的计算代理心跳中断超过30秒时，会触发虚拟机HA，当一个虚拟机由运行状态突然异常消失时，也会触发HA在其他正常的计算节点上快速恢复业务。

通过存储层面的锁机制可防止同一个虚拟机实例在多个物理机器上同时启动。

当一个物理服务器节点掉电恢复后，业务进程开机自启动恢复，之前运行的虚拟机全部故障迁移至其他物理节点。

4.3.2 基于热备机制的虚拟机运行业务容灾方案

在虚拟环境下设置主备虚拟机，在备节点上创建主虚拟机的完整拷贝。主节点上虚拟机的CPU状态、内存、磁盘操作、QEMU等与备节点虚拟机保持低延迟的定时同步。备节点虚拟机定时检测主节点虚拟机心跳，在指定时间内收不到心跳即认为异常发生，备虚拟机切换到正常运行状态。这个方案的优势是主备节点可以保持状态完全同步，数据完全一致，缺点是会带来一些性能开销。当一处系统因意外(如火灾、地震等)停止工作时，整个应用系统可以切换到另一处，使得该系统功能可以继续正常工作。

基于热备机制的虚拟机容灾方案可以实现主机层复制容灾，该方案主要通过在虚拟化平台主机层进行虚拟机卷I/O的实时捕获与复制，实现数据的远程复制和容灾管理，实现容灾保护策略制定、容灾计划制定、容灾切换、容灾回切及有计划性的虚拟机迁移等。

基于热备机制的虚拟机容灾方案是基于I/O分流技术，实现捕获生产端VM的实时I/O数据并异步复制到容灾端VM卷中。VM的整个I/O数据流向为：生产端捕获，代理转发至容灾站点。通过在源端捕获虚拟机I/O，使用代理进程分别部署在生产站点和容灾站点负责I/O转发至容灾端，再分发到指定主机的写入代理进程上，由写入代理进程写入VM的卷中，完成虚拟机I/O的远程异步复制。

4.4 针对企业关键应用云化的虚拟化调优

为了满足电信和企业关键应用，计算虚拟化需要满足计算高性能低时延的要求、可靠性的要求、资源占用效率的要求。为了满足计算高性能低时延的要求，主要的关键性技术如下。

4.4.1 虚拟化调度优化

一、精细化CPU调度技术

为了保证电信和关键企业应用运行的性能，就要求同一台物理机上的多个虚拟化运行实例所获取的资源既能满足其运行的需要，同时不互相产生干扰，精细化的CPU调度技术应运而生。

精细化CPU调度技术主要指的是CPU上下限配额及优先级调度技术。

现代计算机体系结构一般至少有两个特权级(即用户态和核心态)用来分隔系统软件和应用软件。那些只能在处理器的最高特权级(内核态)执行的指令称之为特权指令，一般可读写系统关键资源的指令(即敏感指令)绝大多数都是特权指令(x86存在若干敏感指令，这些指令是非特权指令)。如果执行特权指令时处理器的状态不在内核

态，通常会引发一个异常，从而交由系统软件来处理这个非法访问(陷入)。经典的虚拟化方法就是使用“特权解除”和“陷入-模拟”的方式，即将Guest OS运行在非特权级，而将VMM运行于最高特权级(完全控制系统资源)。解除了Guest OS的特权级后，Guest OS的大部分指令仍可以在硬件上直接运行，只有执行到特权指令时，才会陷入到VMM模拟执行(陷入-模拟)。“陷入-模拟”的本质是保证可能影响VMM正确运行的指令由VMM模拟执行，大部分的非敏感指令还是照常运行。

因为x86指令中有若干条指令是需要被VMM捕获的敏感指令，但是却不是特权指令(称为临界指令)，因此“特权解除”并不能导致它们发生陷入模拟，从而阻碍指令的虚拟化。x86下的敏感指令大致分类如下。

(1) 访问或修改机器状态或虚拟机状态的指令。

(2) 访问或修改敏感寄存器或存储单元的指令，比如访问时钟寄存器和中断寄存器。

(3) 访问存储保护系统或内存、地址分配系统的指令(段页之类)。

(4) 所有I/O指令。

其中的(1)和(4)都是特权指令，在内核态下执行时会自动产生陷阱被VMM捕获，但是(2)和(3)不是特权指令，而是临界指令。部分临界指令会因为Guest OS的权限解除执行失败，但是却不会抛出异常，所以不能被捕获，如(3)中的VERW指令。

基于x86计算架构的指令处理原理，CPU的精细化调度技术采用了vCPU调度分配机制来实现精细化管控，通常也称之为CPU QoS功能(见图2-6)。

从虚拟机系统的结构与功能划分可以看出，客户操作系统与虚拟机监视器共同构成了虚拟机系统的两级调度框架，图4-5是一个多核环境下虚拟机系统的两级调度框架。客户操作系统负责第2级调度，即线程或进程在vCPU上的调度(将核心线程映射到相应的虚拟CPU上)。虚拟机监视器负责第1级调度，即vCPU在物理处理单元上的调度。两级调度的调度策略和机制不存在依赖关系。vCPU调度器负责物理处理器资源在各个虚拟机之间的分配与调度，本质上把各个虚拟机中的

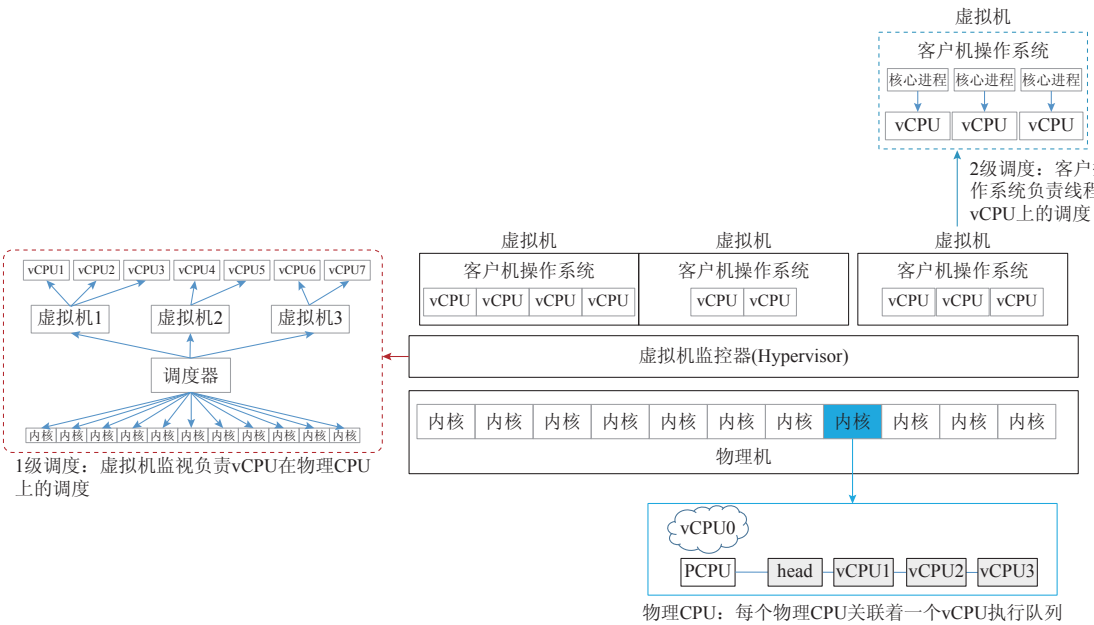


图4-5　vCPU调度分配机制

vCPU按照一定的策略和机制调度在物理处理单元上，可以采用任意的策略来分配物理资源，满足虚拟机的不同需求。vCPU可以调度在一个或多个物理处理单元执行(分时复用或空间复用物理处理单元)，也可以与物理处理单元建立一对一固定的映射关系(限制访问指定的物理处理单元)。

所以，CPU QoS功能实现的最终形态包括以下几种。

- 资源上限限制：适用资源严格隔离场景。
- 资源下限预留：适用资源衡量场景，结合上限设置实现资源可销售。
- 资源份额分配：适用资源复用场景。
- 上限限制：保证虚拟机隔离资源竞争，保证用户体验。
- 下限预留：最低资源保障，保证服务质量。
- 份额分配：针对用户划分不同等级，实现资源竞争。

CPU QoS的价值在于为应用提供计算服务质量的保障，确保针对资源的分配是确定的、可衡量的。

二、NUMA架构感知的调度技术

随着x86架构体系的发展，大部分用于提供计算资源的物理器件，也就是x86服务器，都按照SMP的系统架构来进行处理能力的增强。这就引入了NUMA的问题。

通过虚拟化软件的Host NUMA技术，可以显著提高虚拟机的性能，降低处理时延，以下针对NUMA和虚拟化软件的Host NUMA技术进行详细描述。

NUMA是非一致性内存架构(Non-uniform Memory Architecture)，解决了SMP系统中的可扩展性问题。NUMA将几个CPU通过内存总线与一块内存相连构成一个组，整个系统就被分为若干个Node(见图4-6)。

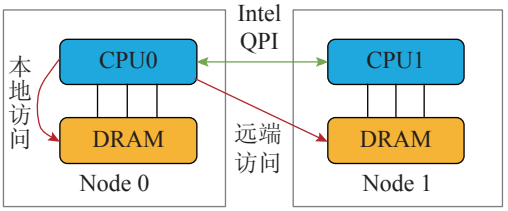


图4-6 CPU Node

一个Node服务器内包含若干CPU和一块内存，CPU访问其所在Node的本地内存的速度最快，访问其他Node的内存性能较差。

操作系统根据SRAT和SLIT表识别NUMA拓扑(见图4-7)。

虚拟化软件实现的Host NUMA主要提供CPU

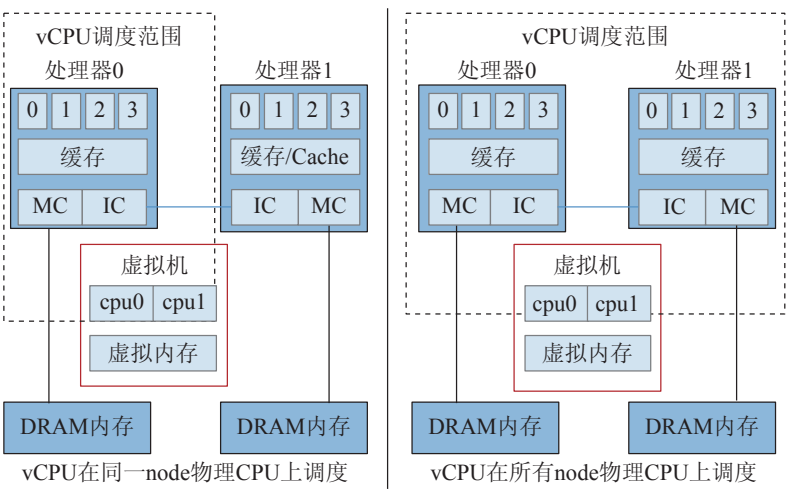


图4-7 Host NUMA原理图

负载均衡机制，解决CPU资源分配不平衡引起的VM性能瓶颈问题，当启动VM时，Host NUMA根据当时主机内存和CPU负载，选择一个负载较轻的node放置该VM，使VM的CPU和内存资源分配在同一个node上。如图4-7左边所示，Host NUMA把VM的物理内存放置在一个node上，对VM的vCPU调度范围限制在同一个node的物理CPU上，并将VM的vCPU亲和性绑定在该node的物理CPU上。考虑到VM的CPU负载是动态变化，在初始放置的node上，node的CPU资源负载也会随之变化，这会导致某个node的CPU资源不足，而另一个node的CPU资源充足，在此情况下，Host NUMA会从CPU资源不足的node上选择VM，把VM的CPU资源分配在CPU资源充足的node上，从而动态实现node间的CPU负载均衡。

对于VM的vCPU个数超过node中CPU的核数的VM，如图4-7右边所示，Host NUMA把该VM的内存均匀地放置在每个node上，vCPU的调度范围为所有node的CPU。用户绑定了VM的vCPU亲和性，Host NUMA特性根据用户的vCPU亲和性设置决定VM的放置，若绑定在一个node的CPU上，Host NUMA把VM的内存和CPU放置在一个node上，若绑定在多个node的CPU上，Host NUMA把VM的内存均匀分布在多个node上，VM的vCPU在多个node的CPU上均衡调度。

虚拟化软件提供复杂的NUMA调度程序来动态平衡处理器负载，根据当时主机内存和CPU负载优先把VM的CPU和内存资源分配在同一个node上，并随着资源负载的动态变化对主机node间的CPU资源做负载均衡。

此特性为虚拟化软件基本特性，不需要管理员明确处理节点之间的虚拟机平衡，在各种场景都可使用。

Host NUMA保证VM访问本地物理内存，减少了内存访问延迟，可以提升VM性能，性能提升的幅度与VM虚拟机访问内存大小和频率相关。

三、内存复用技术

前文介绍了计算虚拟化技术中CPU分配的技术，本节针对提高资源利用率的内存管理和复用技术进行说明(见图4-8)。

计算虚拟化软件中，VMM (Virtual Machine Monitor) 掌控所有系统资源，因此VMM 握有整个内存资源，其负责页式内存管理，维护虚拟地址到机器地址的映射关系。因Guest OS 本身亦有页式内存管理机制，则有VMM的整个系统就比正常系统多了一层映射。

映射关系如下：Guest OS：PA = f(VA)、VMM：MA = g(PA)。

VMM 维护一套页表，负责PA到MA的映射。Guest OS维护一套页表，负责VA 到PA的映射。实际运行时，用户程序访问VA1，经Guest OS的页表转换得到PA1，再由VMM 介入，使用VMM的页表将PA1 转换为MA1。

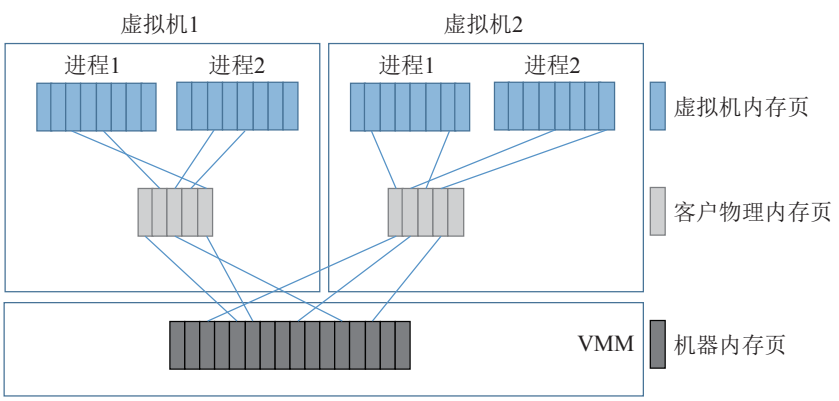


图4-8　内存虚拟化三层模型

四、页表虚拟化技术原理

普通MMU只能完成一次虚拟地址到物理地址的映射，在虚拟机环境下，经过 MMU 转换所得到的“物理地址”并不是真正的机器地址。若需得到真正的机器地址，必须由VMM 介入，再经过一次映射才能得到总线上使用的机器地址。如果虚拟机的每个内存访问都需要VMM 介入，则由软件模拟地址转换的效率是很低下的，几乎不具有实际可用性，为实现虚拟地址到机器地址的高效转换，现普遍采用的思想是：由VMM根据映射f和g生成复合的映射fg，并直接将这个映射关系写入 MMU。当前采用的页表虚拟化方法主要是MMU类虚拟化(MMU Paravirtualization)和影子页表，后者已被内存的芯片辅助虚拟化技术所替代。

MMU Paravirtualization的基本原理是当Guest OS创建一个新的页表时，会从它所维护的空闲内存中分配一个页面，并向Xen注册该页面，Xen会剥夺Guest OS对该页表的写权限，之后Guest OS对该页表的写操作会陷入到Xen加以验证和转换。Xen会检查页表中的每一项，确保他们只映射了属于该虚拟机的机器页面，而且不得包含对页表页面的可写映射。然后Xen会根据自己所维护的映射关系，将页表项中的物理地址替换为相应的机器地址，最后再把修改过的页表载入MMU。如此，MMU就可以根据修改过的页表直接完成虚拟地址到机器地址的转换。

五、内存芯片辅助虚拟化

内存的芯片辅助虚拟化技术是用于替代虚拟化技术中软件实现的“影子页表”的一种芯片辅助虚拟化技术。其基本原理是：GVA(客户操作系统的虚拟地址)转换到 GPA(客户操作系统的物理地址)，然后再转换到 HPA(宿主操作系统的物理地址)。两次地址转换都由CPU硬件自动完成(软件实现内存开销大、性能差)。我们以VT-x技术的页表扩充技术Extended Page Table(EPT)为例。首先，VMM预先把客户机物理地址转换到机器地址的EPT页表设置到CPU中；其次，客户机修改客户机页表无须VMM干预；最后，地址转换时，CPU自动查找两张页表完成客户机虚拟地址到机器地址的转换。使用内存的芯片辅助虚拟化技术，客户机运行过程中无须VMM干预，去除了大量软件开销，内存访问性能接近物理机。

虚拟化软件平台提供多种内存复用技术和灵活自动的内存复用策略。对于某些物理内存资源比较紧张的场景，如果用户希望运行超过物理内存能力的虚拟机，以达到节省成本的目的，就需要有内存复用策略来动态地对内存资源进行分配和复用。内存复用策略通过内存复用技术，提升物理内存利用率的同时，尽可能减少对虚拟机性能的影响。客户无须关心何时调用和怎么调用几种复用技术，只需简单配置和开启复用策略后就能达到提升虚拟机密度的目的。

虚拟化软件的内存复用技术有以下三种：内存气泡、内存零页共享和内存交换技术。

1. 内存气泡技术(Ballooning)

内存气泡技术是一种VMM通过“诱导”客户机操作系统来回收或分配客户机所拥有的宿主机物理内存的技术。当客户机物理内存足够时，客户机操作系统从其闲置客户机物理机内存链表中返回客户机物理内存给气球；当客户机物理内存资源稀缺时，客户机操作系统必须回收一部分客户机物理内存，以满足气球申请客户机物理内存的需要。通过Balloon Driver模块，从源虚拟机申请可用内存页面，通过Grant Table授权给目标虚拟机，并更新虚拟机物理地址和机器地址映射关系表。

通过使用Ballooning技术，可以提升内存使用效率。

2. 零页共享技术

内存零页共享技术作为内存复用技术的一种，能有效地识别和释放虚拟机内未分配使用的零页，以达到提高内存复用率的目的。客户开启零页共享技术后，能实时从虚拟机内部把零页进行共享，从而把其占用的内存资源释放出来给其他虚拟机使用，以创建更多的虚拟机，实现提高虚拟机密度的目的。与内存气泡技术不同，零页

共享后的内存页对于虚拟机来说还是可用的，虚拟机可以随时根据需要收回这部分内存，用户体验相对来说更加友好。

用户进程定时扫描虚拟机的内存数据，如果发现其数据内容全为零，则通过修改P2M映射的形式把其指向一个特定的零页。从而做到在物理内存中仅保留一份零页拷贝，虚拟机的所有零页均指向该页，从而达到节省内存资源的目的。当零页数据发生变动时，由Xen动态地分配一页内存出来给虚拟机，使修改后的数据有内存页进行存放，因此对于GuestOS来说，整个零页共享过程是完全不感知的。

3. 内存交换技术

内存交换技术作为内存复用技术的一种，能通过Xen把虚拟机内存数据换出到存储介质上的交换文件中，从而释放内存资源，以达到提高内存复用率的目的。由于内存气泡和零页共享的数量与虚拟机本身的内存使用情况强相关，因此其效果不是很稳定，用户使用内存交换技术，可以弥补上述不足，即可以保证释放出一定量的内存空间出来(理论上所有虚拟机内存都能交换出来)，但同时也会带来一定程度的虚拟机性能下降。

内存交换触发时，根据用户需要告知Xen需要向某个虚拟机交换出一定量的内存页出来，Xen按一定的选页策略从虚拟机中选择相应数量的页后，把页数据保存到存储介质上的交换文件中，同时释放原先存放数据的那些页供其他虚拟机使用。当虚拟机读写的页正好是被换出的页时，在缺页处理时Xen会重新为其分配一页内存，然后从存储介质上的交换文件中把相应的页交换回新分配的内存页中，同时再选择另外一页内存交换出去，从而保证虚拟机对页的正常读写的同时，稳定交换页的数量。这个过程与零页共享一样，对GuestOS都是不可感知的。

六、I/O调度中断优化技术

影响电信和企业关键应用的运行时响应时延的因素很多，其中，中断处理就是一个在虚拟化条件下的关键因素。

CPU在处理I/O访问请求出现中断时，其实际处于等待的状态，在虚拟化的条件下，因为单个物理机上运行的虚拟机实例比较多，因此所产生的I/O中断请求也会变多，这样实际上造成了物理CPU的等待，从而影响了应用运行的响应时延和性能。

VMM 通过I/O虚拟化来复用有限的外设资源，其通过截获 Guest OS对I/O设备的访问请求，然后通过软件模拟真实的硬件来响应这些截获的请求。目前I/O设备的虚拟化方式主要有三种：设备接口完全模拟、前端/后端模拟、直接划分。

七、设备接口完全模拟

设备接口完全模拟即软件精确模拟与物理设备完全一样的接口，Guest OS驱动无须修改就能驱动这个虚拟设备。优点是没有额外的硬件开销，可重用现有驱动程序，缺点是为完成一次操作要涉及多个寄存器的操作，使得VMM要截获每个寄存器访问并进行相应的模拟，导致多次上下文切换，性能较低。

八、前端/后端模拟

VMM 提供一个简化的驱动程序(后端，Back-End)，Guest OS中的驱动程序为前端驱动 (Front-End，FE)，前端驱动将来自其他模块的请求通过与Guest OS间的特殊通信机制直接发送给Guest OS的后端驱动，后端驱动在处理完请求后再发回通知给前端驱动(Xen即采用该方法)。优点是基于事务的通信机制能在很大程度上减少上下文切换开销，没有额外的硬件开销，缺点是需要VMM实现前端驱动，后端驱动可能成为瓶颈。

九、直接划分

直接划分即直接将物理设备分配给某个Guest OS，由Guest OS直接访问I/O设备(不经 VMM)，目前与此相关的技术有AMD IOMMU、Intel VT-d、PCI-SIG之SR-IOV 等，旨在建立高效的I/O虚拟化直通道。优点是直接访问减少了虚拟化开销，缺点是需要购买额外的硬件。

十、网络直通VMDq技术

在虚拟化的条件下，网络访问报文需要从软件层的虚拟网卡经过物理网卡才能发出。软件模拟虚拟网卡会造成网络访问时延的增加和抖动。为了解决这个问题，先后发展出VMDq的技术和SR-IOV的技术。

在虚拟环境中，Hypervisor管理网络I/O活动，随着平台中的虚拟机和传输量增加，Hypervisor需要更多的CPU周期来进行数据包分类操作，并需要将数据包路由对应到相应的虚拟机中，这些操作会因对CPU的占用，而影响上层应用软件对CPU的使用。Hypervisor利用VMDq(Virtual Machine Device Queues虚拟机设备队列)技术，针对虚拟机网络性能有极高要求的场景，在支持VMDq的网卡上，用硬件实现Layer 2分类/排序器，根据MAC地址和VLAN信息将数据包发送到指定的网卡队列中。这样虚拟机收发包时就不需要Dom0的参与。这种模式极大地提升了虚拟化网络效率。

Intel VMDq(Virtual Machine Device Queue 虚拟机设备队列)技术，是专门用于提升网卡的虚拟化I/O性能的硬件辅助I/O虚拟化技术，主要解决I/O设备上频繁地VMM切换以及对中断的处理问题，其可以减轻Hypervisor的负担，同时提高虚拟化平台网络I/O性能。

VMDq技术可以将网络I/O管理负担从Hypervisor上卸载掉，多个队列和芯片中的分类智能性支持虚拟环境中增强的网络传输流，从应用任务中释放处理器周期，提高向虚拟机的数据处理效率及整体系统性能。VMDq为虚拟机提供接近物理机的网络通信性能，兼容部分虚拟化高级特性，比如在线迁移、虚拟机快照等。

十一、网络直通SR-IOV技术

与VMDq类似，SR-IOV也是采用类直通的方式来避免软件层对于网络转发的时延、抖动的影响，从而满足电信与企业关键应用对于高性能低时延的要求。

服务器虚拟化技术是通过软件模拟多个网络适配器的方式来共享一个物理网络适配器端口，来满足虚拟机的I/O需求。虚拟化软件在多个层面控制和影响虚拟机的I/O操作，因此导致环境中出现瓶颈并影响I/O性能。SR-IOV是一种不需要软件模拟就可以共享I/O设备I/O端口的物理功能的方法，主要利用iNIC实现网桥卸载虚拟网卡，允许将物理网络适配器的SR-IOV虚拟功能直接分配给虚拟机，可以提高网络吞吐量，并缩短网络延迟，同时减少处理网络流量所需的主机CPU开销。

SR-IOV(Single Root I/O Virtualization)是PCI-SIG推出的一项标准，是虚拟通道(在物理网卡上对上层软件系统虚拟出多个物理通道，每个通道具备独立的I/O功能)的一个技术实现，用于将一个PCIe设备虚拟成多个PCIe设备，每个虚拟PCIe设备如同物理PCIe设备一样向上层软件提供服务。通过SR-IOV，一个PCIe设备不仅可以导出多个PCI物理功能，还可以导出共享该I/O设备上的资源的一组虚拟功能，每个虚拟功能都可以被直接分配到一个虚拟机，能够让网络传输绕过软件模拟层，直接分配到虚拟机，实现将PCI功能分配到多个虚拟接口以在虚拟化环境中共享一个PCI设备的目的，并且降低了软件模拟层中的I/O开销，因此实现了接近本机的性能。在这个模型中，不需要任何传输，因为虚拟化在终端设备上发生，允许管理程序简单地将虚拟功能映射到VM上以实现本机设备性能和隔离安全。SR-IOV虚拟出的通道分为两个类型。

- PF(Physical Function) 是完整的PCIe设备，包含了全面的管理、配置功能，Hypervisor通过PF来管理和配置网卡的所有I/O资源。
- VF(Virtual Funciton)是一个简化的PCIe设备，仅仅包含了I/O功能，通过PF衍生而来，就如物理网卡硬件资源的一个切片。对于Hypervisor来说，这个VF同一块普通的PCIe网卡一模一样，可满足高网络I/O应用要求，无须特别安装驱动，且可以实现无损热迁移、内存复用、虚拟机网络管控等虚拟化特性。

十二、GPU直通调度

在虚拟化的环境中，会有需要在桌面环境下进行图形渲染，显示增强等类型的用户应用，要求虚拟机能够使用GPU进行图形处理运算，并且一个GPU能够为一个或多个虚拟机提供vGPU资源。

GPU虚拟化功能通过GPU硬件虚拟化功能，使得一个物理GPU设备可虚拟为多个虚拟GPU设备供虚拟机使用，每个虚拟机通过绑定的vGPU可以直接访问物理GPU的部分硬件资源(所有vGPU都能够分时共享访问物理GPU的3D图形引擎和视频编解码引擎，并拥有独立的显存)。物理GPU通过DMA的方式能够直接获取图形应用下发给vGPU Driver的3D指令，渲染后将结果放到在各自vGPU的显存内，虚拟机中的vGPU驱动可以直接从物理显存中抓取渲染数据。因此上层应用可以直接获取物理GPU的硬件能力，对上层应用来说就像直接使用物理GPU一样，因而具有强大的图形性能和良好的程序兼容性。

4.4.2 跨服务器的计算资源调度

当物理服务器宕机或者重启时，系统可以将具有HA属性的故障虚拟机迁移到其他物理服务器，保证虚拟机快速恢复。

一、高性能、低时延的虚拟机热迁移机制

虚拟机是弹性计算服务的资源实体，为保证虚拟资源池中对于虚拟机资源的灵活分配，需提供在资源池内的虚拟机热迁移能力，即虚拟机在不中断业务的情况下实现在不同物理机上的迁移。虚拟机迁移时，管理系统会在迁移的目的端创建该虚拟机的完整镜像，并在源端和目的端进行同步。同步的内容包括内存、寄存器状态、堆栈状态、虚拟CPU状态、存储以及所有虚拟硬件的动态信息。在迁移过程中，为保证内存的同步，虚拟机管理器(Hypervisor)提供了内存数据的快速复制技术，从而保证在不中断业务的情况下将虚拟机迁移到目标主机。同时，通过共享存储保证了虚拟机迁移前后持久化数据不变。

虚拟机热迁移的作用，具体如下。

- 降低客户的业务运行成本：根据时间段的不同，客户的服务器会在一定时间内处于相对空闲的状态，此时若将多台物理机上的业务迁移到少量或者一台物理机上运行，而将没有运行业务的物理机关闭，就可以降低客户的业务运行成本，同时达到了节能减排的作用。
- 保证客户系统的高可靠性：如果某台物理机运行状态出现异常，在进一步恶化之前将该物理机上运行的业务迁移到正常运行的物理机上，就可以为客户提供高可用性的系统。
- 硬件在线升级：当客户需要对物理机硬件进行升级时，可先将该物理机上的所有虚拟机迁移出去，之后对物理机进行升级，升级完成再将所有虚拟机迁移回来，从而实现在不中断业务运行的情况下对硬件进行升级，保证服务的持续可用性。

一个虚拟化系统，至少需要在下列场景下支持虚拟机热迁移功能：

- 根据需要按照迁移目的手动把虚拟机迁移到空闲的物理服务器；
- 根据资源利用情况将虚拟机批量迁移到空闲的物理服务器。

虚拟机应用于电信和企业关键应用领域，且虚拟网元在硬件平台间无缝迁移时，面临切换时延带来虚拟机内部业务中断的问题，通信领域虚拟机迁移切换时间确保在1s以内，同时电信和企业关键应用业务压力较大，对热迁移构成挑战。一般可以采用如下技术提升热迁移的效果。

混合拷贝热迁移，具体如下。

- Pre-copy和Post-copy按需自适应切换，避免重载业务无限制迭代拷贝。
- 前台和后台数据传输带宽管控，充分利用带宽完成后台传输。
- 识别热点内存，减少缺页概率。

RDMA/RoCE热迁移加速，具体方法如下。

- 通过RDMA硬件能力加速，提升迁移效率。
- vMotion与RDMA相结合的方式进行加速。

- 采用RoCE网卡直通技术。
- 在Host中集成RoCE协议栈，改造热迁移等服务采用的协议栈。

二、计算资源池的动态资源调度管理和动态能耗管理

在云化资源池的环境下，虚拟机的负载是动态变化的，因此计算资源池的管理系统需要实现对于虚拟机部署位置的自动化调整，以保障虚拟机能够获得所需要的资源，同时也保障资源池内的负载是均衡的，从而保障资源池资源的高效利用。

实现该技术的方式，通常称为动态资源调度管理，简称为DRS(Dynamic Resource Schedule)。

DRS周期性监控集群下物理主机和虚拟机负载(CPU和内存)，如果主机负载的不均衡度(标准方差)超过集群配置的阈值，则触发虚拟机迁移，使得集群范围内的负载区域平衡。作为DRS的扩展功能，DPM(Distributed Power Management 分布式电源管理)支持在集群负载低(CPU和内存负载小于集群配置的阈值)时，主动对某些主机进行下电，达到节能减排的目的，同时在集群负载高(负载大于集群配置的阈值)时，重新启动一些主机，满足业务需要。

以下是DRS的几种基本的应用场景。

- 场景1：集群内主机间的负载不均衡，调度后将部分虚拟机从负载高的主机迁移至负载低的主机，达到负载均衡。
- 场景2：集群内主机间的负载不均衡，负载也较低，调度后负载达到平衡，同时对经过评估可以下电的主机(物理主机3)进行下电。
- 场景3：集群内主机的负载均很高，调度后将处于已下电状态的主机(物理主机3)上电，并将部分虚拟机迁移至负载低的主机上，达到负载均衡。

实现DRS和DPM的调度算法很多，通常衡量一个资源池内部的负载不平衡的程度，可以采用每个计算服务器节点的负载与资源池平均负载差异值的标准方差来衡量。差值越大，表示该节点与资源池平均负载的差异越大，越需要调整其负载，以保证尽快地将集群的负载进行均衡。

进行资源平衡调整的方法，业界有很多的实现方式，针对不同的负载波动水平，不同的业务可靠性要求，都会在集群负载收集的时间和门限、计算节点负载调整的门限和时间间隔等处理上进行不同的调整。比较优良的算法，还会保证对集群内的负载根据历史的运行状况进行负载预测，以避免负载调整的震荡。

通常情况下，动态资源调整主要考虑CPU和内存的负载变化情况，但网络和存储的负载在面对电信和企业关键应用中也是重要的且必须考虑的因素，针对CPU/内存/网络/存储等多维资源的负载预测可以采用以上类同算法。

真正进行负载调整时，触发进行业务部署调整的原则为：负载方差大于阈值，或物理机利用率超过阈值。

需要进行部署位置调整的源VM选择原则为：选择离均值最远的物理主机之上的“最有效”VM(即迁移后使该物理主机的负载最靠近均值)。

目标物理主机选择的原则：选择接受源VM后使方差下降最多的物理主机。

通过重复步骤计算，直到方差低于阈值或者达到VM迁移次数限制。

需要说明的是，动态能耗管理对于虚拟机和物理机的负载计算方法，实际上跟动态资源调度管理的方法是一致的，主要的区别在于动态能耗管理是需要尽量地合并物理主机上的虚拟机业务，以尽量少的物理机占用达成节省能源消耗的目的。

三、亲和性和反亲和性调度

在一些企业应用中，为了达到虚拟机间的资源优化目的，需要将一部分虚拟机放置在一台主机上。该操作可以通过应用亲和性组规则来实现：在虚拟机创建/启动/HA调度时，会为其选择同一亲和组的其他虚拟机所在的主机，达到亲和性调度目的。

在有些场景下，需要将某一类虚拟机分开

放置在不同的主机上，以提供更好的资源优化和服务器故障时的冗余，可以应用反亲和性调度规则：在发放虚拟机时进行调度控制，可以将同一个反亲和性组的虚拟机放置在不同Hypervisor上，包括虚拟机启动、HA、迁移等生命周期管理，都将遵从反亲和性调度。

4.5 基于OpenStack Ironic的裸金属服务

Ironic作为OpenStack Kilo版本正式引入的一个新服务，提供了对裸金属服务器的管理，使用户可以像使用虚拟机一样使用和管理裸金属服务器：支持通过OpenStack Nova接口进行裸金属服务器的指定镜像和规格的创建、查询、修改、启动、停止、重启等生命周期管理操作；还可以将cinder创建的共享存储类型的卷通过和虚拟机相同的接口挂载给裸金属服务器；并且可以通过与Neutron的交互自动配置租户网络，以实现和虚拟机之间的网络互通。

Ironic提供了一些通用的Driver，例如PXE和IPMI的Driver，可以管理大量业界通用的裸机服务器；另外，Ironic的Driver机制具有较好的扩展性，允许针对有特殊诉求的服务器进行定制特有的Driver进行管理。

4.5.1 应用场景

Ironic提供的裸金属服务器管理主要适用于一些不适合部署于虚拟机中的应用，主要如下：

- 需要直接访问不能虚拟化的服务器硬件设备的一些应用；
- 在虚拟机中运行时性能比较差的数据库应用、大数据处理应用等；
- 其他一些对性能、安全等专属硬件有诉求的应用。

4.5.2 基于OpenStack的架构

图4-9描述了Ironic服务与其他OpenStack服务的交互关系。

- Ironic通过与Nova交互，提供裸机的生命周期管理功能；裸机的生命周期管理API由Nova提供，使用同虚拟机相同的API接口进行管理，使用户可以像管理虚拟机一样来管理裸机。
- Ironic通过与Neutron交互，提供裸机的网络管理功能(当前Kilo版本的Neutron不具有自动控制裸机所使用的TOR的配置修改的网络能力，需要自研或对接第三方SDN控制器来实现TOR的自动化配置等网络特性)；裸机使用的Network同虚拟机使用的Network一样，都是Neutron提供，并且裸机和虚拟机可以接入同一个Network来实现二层互通。
- Ironic通过与Glance交互，提供裸机的镜像加载功能，Glance通过对接Swift来存储镜像文件；Ironic可以使用标准的Qcow2、raw格式的镜像文件作为裸机的镜像，镜像的管理与虚拟机镜像的管理方式一样，可以通过Glance的标准镜像管理接口进行管理。
- Cinder服务为裸机提供共享存储的能力(当前Kilo版本的Ironic不具备与Cinder对接为裸机提供共享存储的能力，可以自研扩展对应的能力)，使裸机可以完全像虚拟机一样使用共享存储卷，而且同一个共享存储卷可以挂载给虚拟机，也可以挂载给裸机。
- Ceilometer可以通过与Nova的对接，监控到裸机实例的信息，采用与监控虚拟机实例一样的方式；另外，也可以通过与Ironic对接，监控裸机的带外硬件信息。
- 与其他OpenStack服务一样，Ironic服务也是通过Keystone服务来提供鉴权能力。
- Horizon可以提供裸机实例的管理界面，Horizon通过Nova服务提供的生命周期管理接口来管理裸机实例，Ironic本身提供的API接口并不对EndUser提供，仅仅是管理员操作接口。
- Heat可以实现裸机的资源编排，也可以实现裸机与虚拟机的混合编排。

4.5.3 逻辑架构

图4-10描述了裸机服务所包含的内部组件。

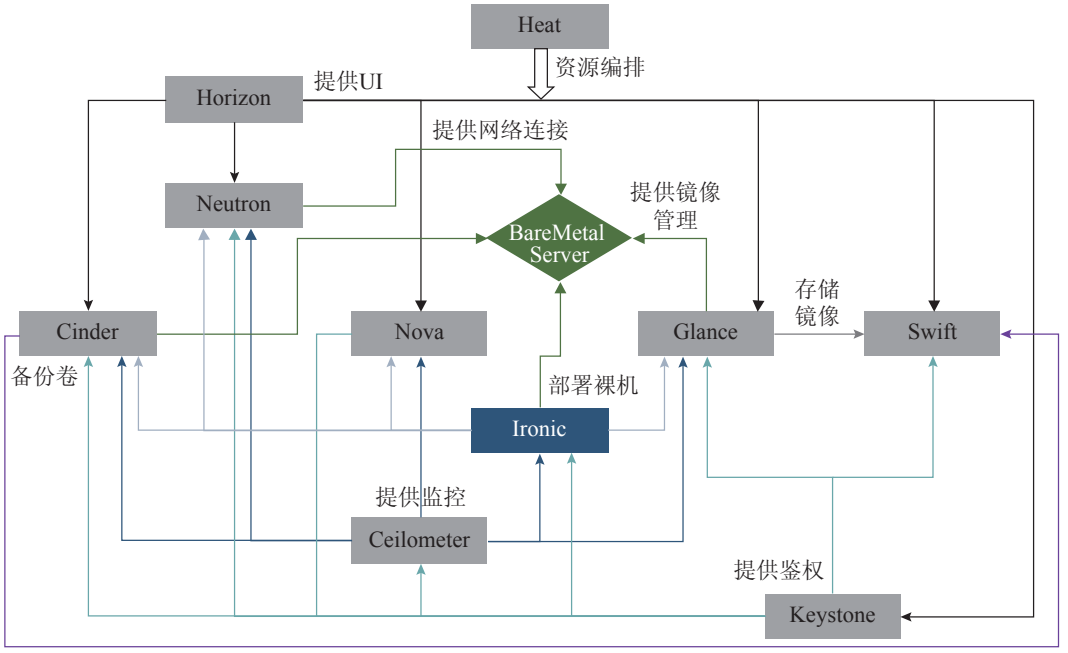


图4-9　Ironic和其他OpenStack组件的交互

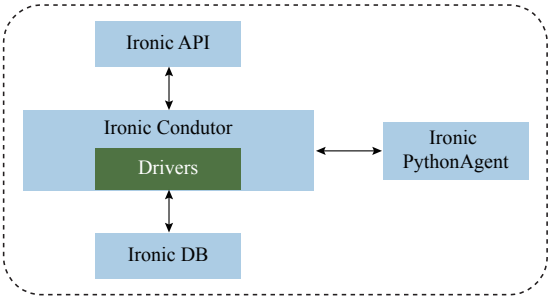


图4-10　Ironic组件构成

Ironic服务由以下组件组成。

✍ Ironic API：处理外部API请求的组件；Ironic的外部API请求主要有两部分，一部分是管理员直接调用Ironic的API用于裸机服务器的配置管理、维护管理等，另一部分是通过Nova Compute组件的Ironic Driver下发的裸机实例生命周期管理的API操作。

✍ Ironic conductor：裸机管理的核心组件，接受Ironic API请求的后端处理组件，通过挂载不同的Driver来完成对裸机服务器的部署及管理。

✍ Drivers：Ironic对于硬件的异构问题主要是通过对接不同的后端驱动来解决的，主要的Driver有PXE、IPMI、amt、drac、iLO、irmc等，也可以以plugin形式扩展特有硬件的Driver。

✍ Ironic-python-agent：集成于deploy ramdisk中，通过与ironic管理服务的交互完成裸金属OS加载等操作，临时存在于裸金属服务器加载启动的内存中，裸金属OS启动后失效。

4.5.4　主机管理

Ironic服务对已注册的裸金属服务器提供集群管理、硬件规格检测、主机GuestOS的自动化安装、主机清理等功能，这些功能通常以带内或带外的方式进行，带外管理通过服务器的BMC控制网络进行，带内管理则通过Ironic-python-agent进行。

一、主机注册

使用Ironic服务管理裸金属服务器前，需要先把主机的信息注册到Ironic中。在主机注册时，需

要指定管理主机使用的Driver以及对应Driver需要的必要信息。以pxe_ipmitool为例，注册时还需要指定服务器的IPMI地址、用户名、密码等信息。为了实现主机GuestOS的自动化安装，还需要指定用于安装引导的deploy_kernel 和 ramdisk。

二、集群管理

图4-11描述了Ironic服务管理裸金属服务器集群的结构。

在OpenStack架构模型中，Ironic的控制服务在网络上要求与服务器的带外管理平面互通，以方便Ironic以带外的方式实现对服务器的管理和控制。Ironic支持通过不同类型的后端驱动对服务器进行电源管理和控制。对于通用的计算服务器，Ironic的IPMI Driver通过IPMI(Intelligent Platform Management Interface)标准协议进行控制和监控，通过IPMI协议，Ironic可以对主机进行上电、下电、重启、设置启动方式等控制。对于不支持IPMI协议的服务器类型，Ironic可以通过与服务器类型匹配的Driver进行管理(如iLO、drac等)。

三、硬件检测

Ironic的可选组件ironic-inspector提供了自动硬件检测的能力，它允许Ironic服务在主机的Driver信息注册完成后自动发现已注册节点的硬件属性信息和硬件网卡信息。

Ironic服务支持两种类型的硬件检测。

(1) 带外检测(Out-of-band)，针对HP的服务器，支持通过iLO驱动进行检测，可以自动检测到memory_mb、cpus、cpu_arch、local_gb等用于调度的基本属性，同时还可以检测到ilo_firmware_version、rom_firmware_version、secure_boot、server_model、pci_gpu_devices、nic_capacity等信息

(2) 带内检测(In-band inspection)：通过ironic-inspector服务(OpenStack kilo版本，该服务名称为ironic-discoverd)，可以实现对通用服务器的自动检测。带内检测的原理是通过在目标服务器上PXE引导启动一个OS(即主机注册时指定的deploy ramdisk和deploy kernel)，通过该OS内运行的ironic-python-agent服务直接发现服务器的硬件信息，如memory_mb、cpus、cpu_arch、local_gb等。

四、主机GuestOS自动化安装

Ironic服务最重要的服务是可以提供裸金属服

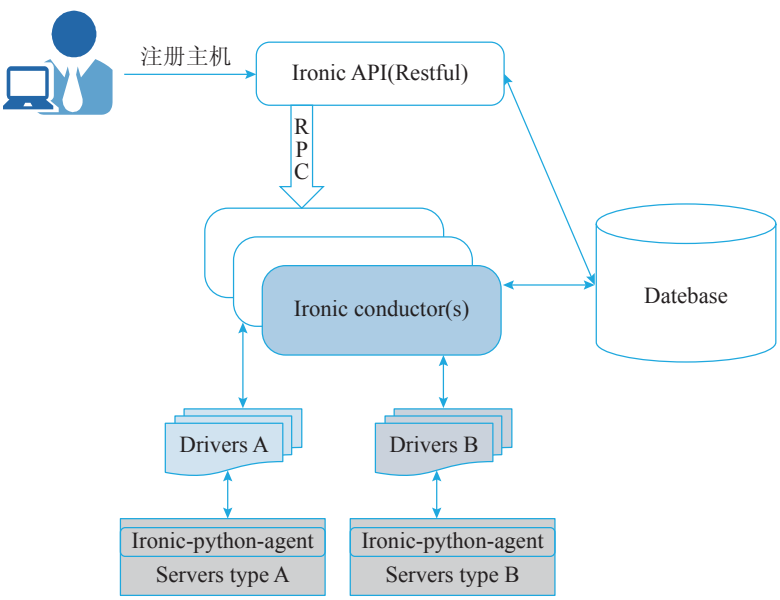


图4-11　Ironic服务集群结构

务器GuestOS的自动化安装。如图4-12所示，Ironic服务通过IPMI网络对裸机的上下电、重启和启动方式进行控制，在安装GuestOS时，设置裸金属服务器从网络启动，通过PXE启动，自动从管理网络的TFTP服务器加载引导用的minios，然后从注册时指定的GuestOS位置或从OpenStack Glance服务下载GuestOS后写入裸金属服务器的本地磁盘。最后设置裸金属服务器从硬盘启动，完成GuestOS的自动安装。

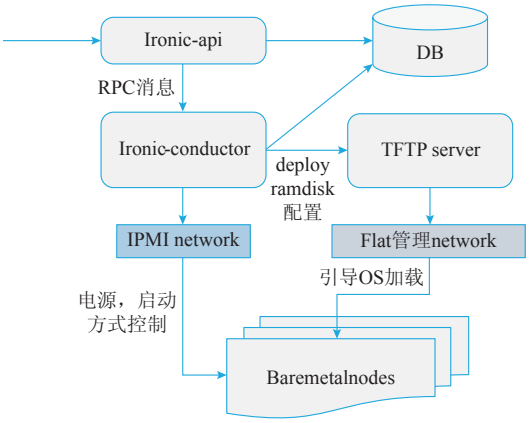


图4-12　主机自动化安装过程

五、主机清理

基于安全方面的考虑，Ironic支持对已注册的主机进行清理，Ironic服务提供了多种主机清理操作，另外也预留了接口给不同的厂商定制自己的主机清理方法。

Ironic提供了两种不同的主机清理方法：自动清理和手工清理。

(1) 自Kilo版本开始，Ironic服务提供自动清理功能，裸金属服务器每次发放给租户后都需要触发自动清理流程，以保证租户每次获取到的主机都是一致的。启用自动清理后，每次清理都会根据Driver中预定义的清理步骤进行清理。

(2) Mitaka版本，Ironic服务将会支持手工清理。手工清理需要管理员通过API触发后执行。

4.5.5　发放管理

图4-13是OpenStack系统中租户申请裸机的流程。

在OpenStack中，裸金属服务器和虚拟机的使用体验完全相同。租户可以通过和虚拟机完全相同的方式使用裸机，包括上传镜像、规格、申请裸金属计算实例等。

裸金属实例的发放管理具体如下。

(1) 管理员通过Ironic API注册裸金属服务器节点信息到Ironic。

(2) Nova-compute通过Ironic API接口查询裸金属服务器资源，记录到Nova数据库。

(3) 租户通过Nova API申请裸金属服务器，Nova API将消息转发给Nova Scheduler。

(4) Nova Scheduler服务通过用户指定的规格，调度到合适的裸金属服务器。

(5) Nova-compute 通过Ironic API接口发送请求到Ironic API，触发Ironic服务开始GuestOS的自动化安装部署。

(6) Nova-compute周期性通过Ironic API检测裸金属服务器的状态，等待Ironic服务完成安装过程。

(7) Ironic-conductor在Neutron中创建DHCP端口，用于后续裸金属服务器网络启动，并配置PXE/TFTP server，准备裸金属服务器启动时的PXE配置。

(8) Ironic-conductor通过IPMI驱动设置裸金属服务器从网络启动，上电裸金属服务器。

(9) 裸金属服务器通过DHCP加载引导OS(deploy_kernel和deploy_ramdisk)，然后根据后端管理驱动的不同完成GuestOS的安装。

(10) GuestOS安装完成后，ironicconductor通过IPMI驱动设置裸金属服务器从硬盘启动，并重启裸金属服务器进入GuestOS，并设置Ironic的状态为安装完成。

(11) Nova-compute检测到裸金属服务器安装完成后，更新计算实例的状态为可用。

4.5.6　网络管理

在OpenStack架构模型中，Ironic服务涉及的网络有以下四类。

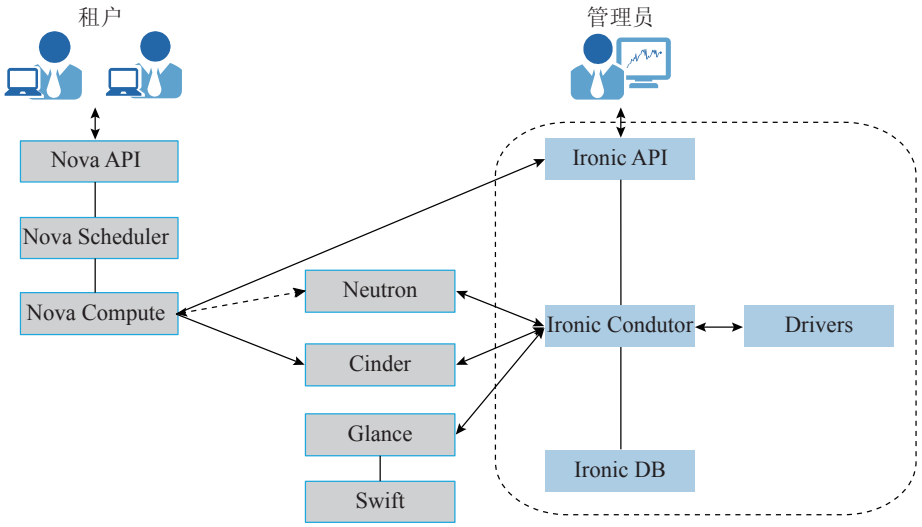


图4-13　用户申请裸金属服务器过程

(1) OpenStack管理网络：用于Ironic服务接受外部API请求，并和 OpenStack Keystone、Nova、Neutron、Glance等服务进行通信。

(2) IPMI网络：Ironic服务通过IPMI协议控制管理裸机需要的网络，在使用Ironic服务的时候需要保证ironic-conductor组件能与裸金属服务器的IPMI网络互通。

(3) PXE网络：Ironic服务完成GuestOS部署所用的网络，PXE/TFTP服务器都在这个网络中。服务器通过Neutron在该网络中提供的DHCP server完成PXE启动，并进行后续的GuestOS镜像下载。在当前的OpenStack版本中，这个网络是一个没有VLAN的Flat类型的网络。

(4) 租户网络：在当前的Ironic服务提供的能力中，无法支持多租户网络的能力，所有裸金属服务器的租户网络都与PXE网络相同。作为一个缺陷，目前OpenStack社区已经开始规划对多租户网络的支持。为了支持对多VLAN/VxLAN网络的自动化配置，基于目前的OpenStack版本，可以自行扩展Neutron对接SDN等插件来支持对TOR的动态自动化配置，满足裸机租户网络的动态、按需自动化配置，以及提供FireWall、VPN等高级网络特性。

4.5.7　存储管理

截止OpenStack Mitaka版本的Ironic服务为止，在租户申请裸金属服务器时，只能使用服务器的本地存储，尚无法支持共享存储卷设备的使用。针对共享存储卷设备，在OpenStack社区Ironic服务的特性规划中，已经有Sanboot等方面的规划。

在Minata版本中，Ironic服务新增对裸金属服务器本地硬盘组RAID的功能，允许管理员通过Ironic的API对服务器进行RAID配置。租户在申请裸金属服务器时可以通过在Flavor中指定RAID级别进行调度。

4.5.8　裸机与虚拟机的交互模型

图4-14描述了OpenStack内部，裸机与虚拟机之间交互的模型。

在OpenStack架构中，租户的网络资源包括网络(Network)、子网(Subnet)、路由(Router)等。在同一网络的子网内，虚拟机和虚拟机之间是二层互通的，同样在同一子网内裸金属服务器和虚拟机也是二层互通的；不同网络的虚拟机直接通过路由三层互通，而裸金属服务器和其他网络内的虚拟机也是通过路由三层互通。

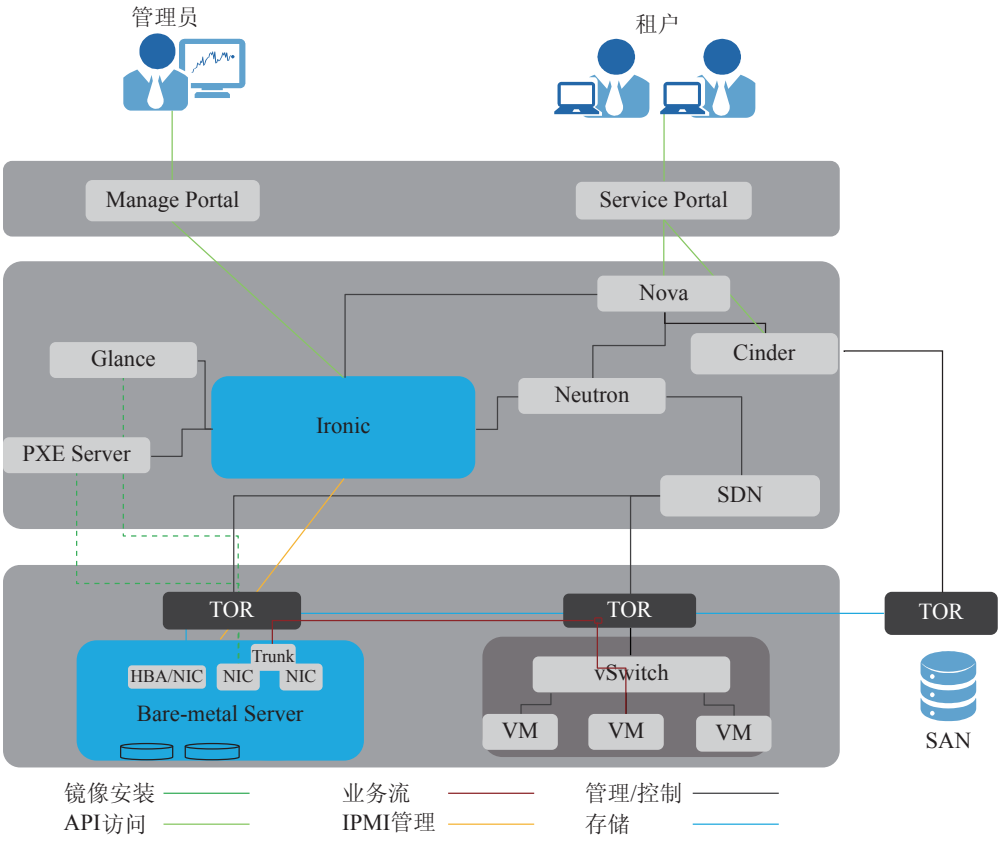


图4-14 裸金属服务器和虚拟机交互过程

4.6 异构适配多种Hypervisor类型

当前存在有多种Hypervisor类型(Xen、KVM、Ironic等)，对于用户来说，单一采用某种虚拟化类型已经不能满足需求，同一个用户可能在不同的场景使用不同的虚拟化类型，甚至在一种场景中使用多种虚拟化类型，这就涉及在同一套业务发放管理平台中支持接入多种类型的Hypervisor。这样不但可以保护用户当前已有的虚拟化技术的平台继续运行，也可以通过构建统一管理的异构资源池，达到业务类型通过不同资源池隔离，而配置管理统一的目的。

由于OpenStack的架构支持不同Hypervisor类型的虚拟机/物理机同时发放。因此可以通过配置实现在同一个OpenStack中进行各个不同类型的Hypervisor的虚拟机/物理机混合发放。其也可以用于异构不同虚拟化厂商的虚拟化软件，前提是该虚拟化软件需要在OpenStack中有对应的Driver。

对应每一种虚拟化类型，需要在OpenStack启动独立的一组nova-compute进程进行管理对接。如图4-15所示，在每个nova-compute中配置对应虚拟化设备的南向驱动，用于对接指定的虚拟化类型。

1. 异构资源池创建

在OpenStack中注册不同类型的nova-compute主机用于对接不同类型的Hypervisor，并为这些不同类型的主机创建不同的HostAggregate，并分别将主机加入到对应的HostAggregate中，通过设

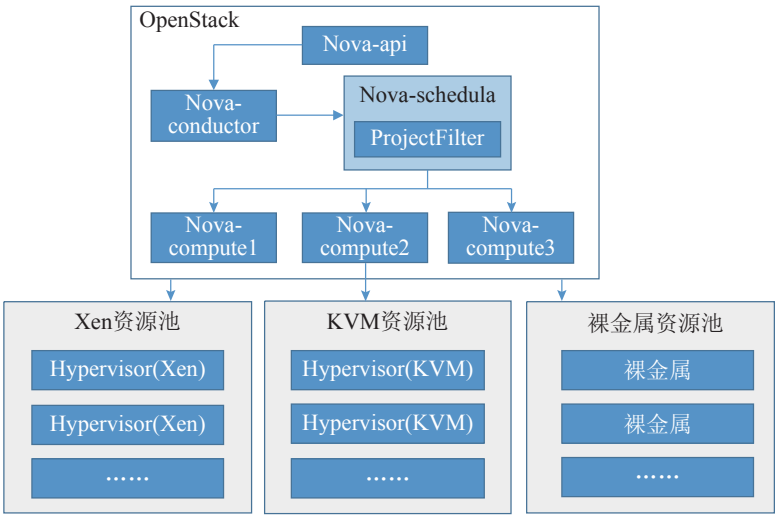


图4-15　各种类型Hypervisor的接入方法

置HostAggregate的metadata(Key，Value)来标记该HostAggregate，例如设置hypervisor_type=xen。

2. Flavor创建

在OpenStack中为不同的Hypervisor类型创建不同的Flavor，并在Flavor中加入指定的extra_specs，该参数的配置需要和Flavor对应类型的HostAggregate的Metadata(Key，Value)保持一致，以确保发放时能够调度到指定HostAggregate。

3. 过滤器配置

开启Nova scheduler的AggregateInstanceExtraSpecsFilter过滤器，这样在创建虚拟机调度时会按照Flavor的extra_specs属性分配满足要求的主机组。

4. 镜像设置

每个不同的虚拟化类型需要使用不同的镜像，通过在镜像中加入Metadata信息，来区分该镜像属于哪种虚拟化类型，再分发到相应HostAggregate进行虚拟机创建。

虚拟机发放时，用户选择特定的Flavor，意味着选择了相应的Hypervisor类型。也就只能选择对应的镜像和计算资源。nova-scheduler会将创建虚拟机请求调度到指定的nova-compute，再由各个不同Hypervisor类型的Driver进行创建虚拟机处理。

目前OpenStack能够接入的Hypervisor类型除了KVM之外，还有VMware、XenServer、Baremetal和Docker等。

第 5 章

面向应用敏捷化部署的Docker容器及其调度

5.1 容器典型应用场景

Docker技术的出现和迅猛发展，已成为云计算产业的新的热点。容器使用范围也由互联网厂商快速向传统企业扩展，大量传统企业开始测试和尝试部署容器云。

相比于企业对容器技术的逐步接受与认同，在如何使用容器上却并不统一，存在多种思路和诉求。容器技术开发者和社区倡导云原生应用场景，这一理念被业界普遍认可，但在实际使用中发生分化。部分企业基于容器技术，尝试新应用开发和对传统应用的改造；而有的企业在实际使用中，面临应用改造困难和人员技能变更的问题，认为不可一蹴而就，希望先以轻量级虚拟机的方式使用容器。

容器的下述技术特点，决定了容器所能发挥真正价值的应用场景。

- 轻量化：容器相比于虚拟机提供了更小的镜像，更快的部署速度。容器轻量化特性非常适合需要批量快速上线的应用或快速规模弹缩应用，如互联网web应用。
- 性能高、资源省：相比虚拟化，接近物理机的性能，系统开销大幅降低，资源利用率高。容器的高性能特性非常适合对计算资源要求较高的应用，如大数据和高性能计算应用。
- 跨平台：容器技术实现了OS解耦，应用一次打包，可到处运行。容器的跨平台能力非常适合作为DevOps的下层封装平台，实现应用的CI/CD流水线；容器应用可跨异构环境在不同云平台、公有云和私有云上部署，也非常适合作为混合云的平台。
- 细粒度：容器本身的“轻”和“小”的特性非常匹配细粒度服务对资源的诉求，与微服务化技术的发展相辅相成，可作为分布式微服务应用的最佳载体。

企业关注下一代内部IT架构变革，希望将服务作为IT核心，提升业务敏捷性，大幅降低TCO。容器成为企业应用转型很好的承载平台，针对企业的业务痛点，使用上述一种或多种特点，优化业务场景。

下面内容详细描述了容器支持的典型应用场景。

5.1.1 互联网web类应用

这是容器技术最广泛使用的场景。Web类应用通常是三层架构(见图5-1)。系统面临大量用户突发业务访问时，对于无状态的Web前端，非常适合使用容器部署。快速规模弹性伸缩Web服务节点实例数，结合ELB的分发调度，适配业务的负载变化。无状态的App服务节点，或通过无状态化改造，也可以打包为容器，提供快速弹缩能力。

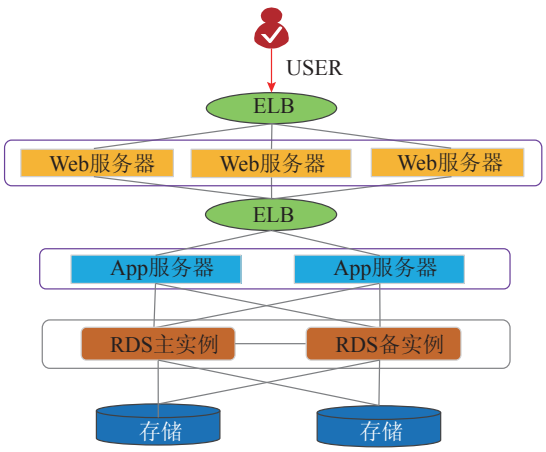


图5-1 Web类应用典型部署

5.1.2 CI/CD开发测试云

Docker开源后，在开发人员和运维人员之间迅速流行起来，成为第一款获得共同认可的DevOps工具，继而成为容器的事实标准。Docker为实现DevOps的四个技术基础技术提供了完善的解决方案，分别是：分布式的开发环境、标准化的运行环境、丰富的应用镜像仓库以及持续的自动化部署(见图5-2)。

- 分布式的开发环境：Docker的分层文件系统机制，使不同的开发人员完全独立地进行开发，并最终进行文件挂载的方式搭建应用开发程序，使开发人员之间的影响最小，实现开发敏捷。
- 标准化的运行环境：Docker Image可以在

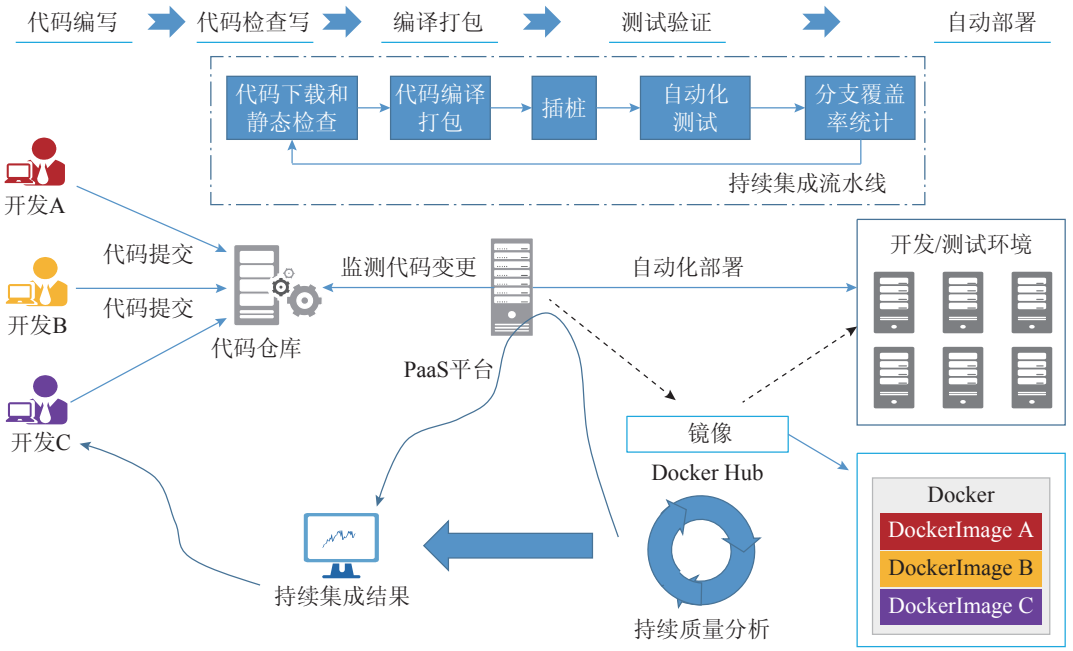


图5-2 CI/CD开发测试流水线

各种支持Docker的开发、测试和生产环境中运行，而屏蔽不同环境间软硬件的差异。

✍ 丰富的应用仓库：在Docker Hub和私有镜像仓库中存储着多种类型的Docker Image，利用仓库来存储Docker镜像，快速搭建应用所需的标准化环境。

✍ 持续的自动化部署：各种Docker的编排工具，如Mesos、Kubernates工具能够支持应用生命周期管理，支持服务发现、负载均衡和灰度升级等，满足运维的应用不停机升级。

5.1.3 微服务管理平台

微服务是一种软件架构模式，此模式下应用被分解为一系列相互独立、边界明确、自主完成单一的任务的服务，服务之间解耦，可独立替换、升级和伸缩，服务间通过语言无关的轻量级接口，如网络通信(RPC、HTTP等)、消息队列等进行协同。

微服务架构将应用解耦分拆为小粒度服务模块，容器的轻量化可为微服务提供更细粒度的资源供给，有效地利用资源。服务启动快和弹缩快，也能更好地应对单服务和系统突发式的业务访问，如图5-3所示。

5.1.4 容器主机

容器不同于虚拟化的实现方式，占用更少的系统资源，有效地提升了数据中心的资源利用率，同时利用容器快速弹性等特性，使业务系统可以灵活扩展，架构演进至微服务架构。

对于轻量级虚拟机，虚拟机管理程序对硬件设备进行抽象处理，而容器只对操作系统进行抽象处理，容器有自己的文件系统、CPU和内存，意味着容器能像虚拟机一样独立运行，却占用更少的资源，极大地提高了资源利用率。

5.2 Docker容器关键技术

5.2.1 Docker Daemon

技术人员一般会将Docker的技术堆栈分成好

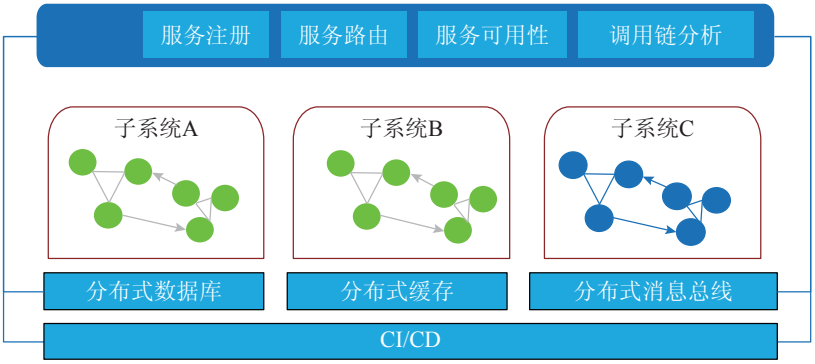


图5-3　微服务管理平台

几层，用来进一步阐明Docker技术本身。相对于整个技术堆栈而言，Docker容器OS层的关键技术显得尤为重要。要了解Docker，首先看看Docker总架构图，如图5-4所示。

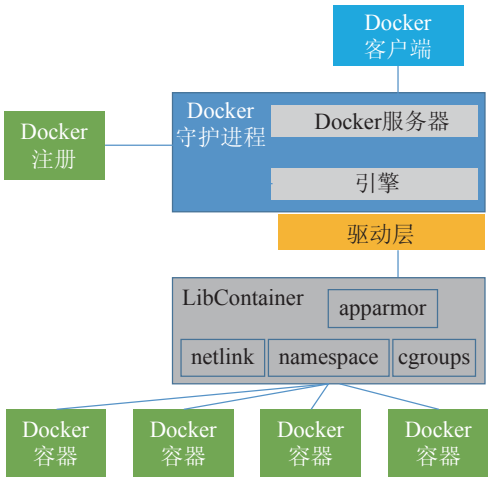


图5-4　Docker总架构图

由图可知，用户可以从Docker Client建立与Docker Daemon的通信连接，发送容器的管理请求，这些请求最终被Docker Server接受，而请求处理则交给Engine处理。

而在与系统调用方面，Docker则是通过更底层的工具LibContainer与内核交互。Libcontainer是真正的容器引擎，是容器管理的解决方案，涉及大量的Linux内核方面的特性，如namespace、cgroups、apparmor等。LibContainer很好地抽象了这些特性，提供接口给Docker daemon。

用户执行启动容器的命令后，一个Docker Container实例就运行起来了，这个实例拥有隔离的运行环境、网络空间以及配置好的受限资源。

Docker Daemon是一个常驻在后台的系统进程，实际上就是驱动整个Docker功能的核心引擎。

简单地说，Docker Daemon实现的功能就是接收客户端发来的请求，并实现请求所要求的功能，同时针对请求返回相应的结果。在功能的实现上，因为涉及容器、镜像、存储等多方面的内容，实现复杂，涉及多个模块的交互。

5.2.2　Docker容器

Container(容器)是整个Docker技术堆栈的核心内容，相对传统虚拟化，这项基础技术在性能上给Docker带来了极大优势。

Docker Daemon通过LibContainer对容器运行实例实现了生命周期的管理，配置信息的设置、查询、监控以及通信等丰富的功能。概念上来说，容器诠释了Docker提倡的集装箱的概念：存放任何货物，通过货轮运输到各个不同的码头。而运输、装载、卸载集装箱的码头都不用关心集装箱中的货物是什么，这种标准的应用封装形式真正意义上打破了原有云计算世界中虚拟机镜像格式的束缚，极大地方便开发人员的工作。Docker容器作为一个集装箱，可以安装指定的软件和库，以及任意环境配置。当开发和运维人员在部署和管理应用的时候，可以直接把应用容器运行起来，不用关心容器里是什么。容器技术不

是一个新的技术，但是通过Docker的神奇封装，与集装箱概念联系起来，开创了云计算领域的新的时代，并迅速推广到全世界。

5.2.3 Docker镜像

运行中的容器实例，提供了一个完整的、隔离状态的运行环境，与之相对应的Docker Image技术，则是整个运行环境的静态体现。相比较传统虚拟机繁杂、多样的镜像格式而言，Docker的镜像要轻量化很多，并且简单很多。从技术层面俯视，这就是一个可以定制的RootFS。

Docker公司在镜像上的另外一个创新就是分层复用。其在一些实际场景中非常有用，大部分的镜像需要相同的OS发行版环境，而许多文件的内容都是一致的，因此复用这些基础文件将会减少很多制作镜像时候的工作。Docker Image做到了这点，采用UnionFS的特性，通过一个基础版本的分层机制，不断开发新的版本堆叠在上面即可，减少磁盘和内存的开销。

开发人员通常会使用Dockerfile来创建Docker Image，这是一个定制镜像内容的配置文件，同时也能在内容上体现层级关系的建立关系。可以使用docker commit这样的命令行方式来手动修改镜像内容并提交生成新的镜像。

5.2.4 Docker Registry

有了Docker Image之后，就需要考虑镜像仓库了，Docker Registry就是这样的功能设定。在Docker的世界中，开发人员可以很容易地从这里下载镜像，Registry通常被部署在互联网服务器或者云端。在Docker Image的传输过程，Registry就是中转站。

例如，在公司里，我们可以把一个软件的运行环境制作成为镜像，上传到Registry中，然后就可以很方便地在可以接入到网络的地方从Registry上把镜像下载下来并运行，当然这些操作都是和Docker结合在一起的，使用者甚至不会感知到Registry的存在，简单的命令行就可以显示所有的操作了。

Docker公司提供的官方镜像Registry叫做Docker Hub，提供了大部分的常用软件和OS发行版的官方镜像，同时也存有无数的个人用户提供的镜像文件。Registry本身也是一个开源项目，任何开发人员可以在下载该项目后，自己部署一个Registry，许多企业都会选择独立部署Docker Registry并且做二次开发，或者购买更强大的企业版Docker Hub。

5.3 容器操作系统

Docker发布以来，对传统的操作系统厂商产生了巨大的冲击，出现了很多容器操作系统，包括CoreOS、Ubuntu Snappy、RancherOS、Red Hat的Atomic等。这些操作系统支持容器技术，作为主要卖点，构成了新的轻量级容器操作系统的生态圈。

传统Linux的操作系统，其发行版本出于通用性考虑，会附带大量的软件包，而很多运行中的应用并不需要这些外围包。例如在容器中运行Java程序，容器中安装了JRE，而容器外的环境不会产生任何依赖，除系统支持Docker运行时的环境之外，无用的外围包可以省略掉。这可以减少一些磁盘空间开销。同样地，运行在后台的服务，如果没有封装到Docker 容器中，也可以认为是不需要的，多余的。减少这样的服务，也可以减少内存的开销，因此全面面向容器的操作系统就这样诞生了，“Container OS”就是最好的诠释，与其他OS相比，这些操作系统更小巧，占用资源更少，相应的运行速度更快。图5-5是容器操作系统与其他OS在软件栈中的对比。

顺应市场需求，容器操作系统应运而生，我们接下来盘点一些业界常见的“Container OS”。

5.3.1 CoreOS

CoreOS是第一个容器操作系统，它是为大规模数据中心和云计算而生，立足于云端生态系统的分布式部署、大规模伸缩扩展的需求，具备在生产环境中运用的能力。

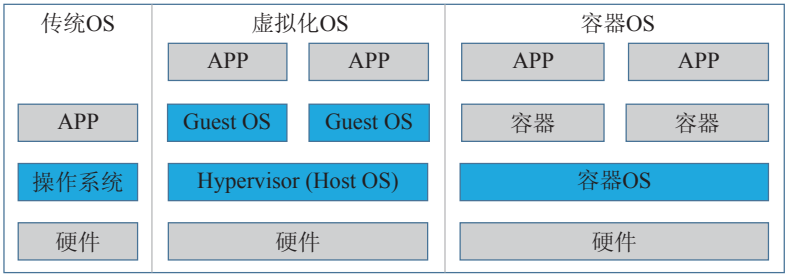


图5-5　几种操作系统在各自应用场景中的位置对比

相比于其他瘦客户操作系统主要是针对特殊需求进行技术定制，CoreOS通过容器技术支持大众通用的选择。CoreOS官方支持很多部署方案，甚至一些有特殊需求的方案也得到支持。

CoreOS是基于Chrome OS再定制的轻量级Linux发行版，将操作系统和应用程序分离，所有用户服务和应用都在容器内运行。鉴于CoreOS源于Chrome OS，因此它的系统升级方法和Chrome OS也很类似，比如，有新的组件发布就会自动更新。而且它支持双系统分区，这样可以通过滚动更新的方式在任何时候直接将系统升级成最新的版本。另外，CoreOS还支持秘钥签名，能有效保证更新的有效性和整个系统的完整性。

CoreOS还同时支持Docker和Rkt，也预装了Etcd、Fleet、Flannel和Cloud-init等容器集群管理配置工具。

5.3.2　RancherOS

RancherOS只由内核和Docker构成。相比于VMware的Photon大约300MB的体积来说，RancherOS是最小的容器操作系统之一，它只有22MB左右。

为了开发Docker化的容器操作系统，它采用了一种与众不同的方式来引导系统，也就是：去掉了内嵌在Linux发行版中的服务管理系统Systemd，改用Docker来引导系统，这个Docker叫“系统Docker”，负责系统服务的初始化，并将所有的系统服务作为Docker容器进行管理，包括Systemd。另外系统Docker会创建一个特殊的容器服务Docker，称为“用户Docker”，主要负责应用容器的管理。

在RancherOS开发的早期阶段，“系统Docker”创建的Systemd容器经常会和“用户Docker”直接冲突。因为“用户Docker”创建的应用容器是在Systemd外创建的，Systemd总是会去杀掉它们。RancherOS花了很长时间才找到一种解决方案，且目前不确定是否还有其他方法可以解决这个问题。

RancherOS具有如下特点。

- 与Docker的开发速度相匹配，提供最新版本的Docker。
- 不再需要复杂的初始化系统，使用一个简单的配置文件，管理人员就能很容易地把系统服务配置成Docker容器。
- 容易扩展，用户很容易通过配置使RancherOS启动一个自定义的控制台容器，提供Ubuntu、CentOS或者Fedora发行版的体验。
- 资源占用小、启动速度快、容易移植、安全性更好，升级、回滚简单。
- 可以使用像Rancher这样的集群管理平台，容易维护。

5.3.3　Snappy Ubuntu Core

Ubuntu是Docker容器技术最流行的Linux发行版，Canonical引以为傲。从Docker的周边生态来看，在Ubuntu上运行Docker容器的数量远远超过其他操作系统。

Canonical为移动设备创建“短小精悍”的操作系统付出了很多努力，也从其中吸取了很多经验教训，而Snappy Ubuntu Core就是运用这些经

验教训开发出来的，其体积只有200MB左右。为了支持用户系统可靠和应用更新的需求，Snappy Ubuntu Core对系统和应用采取“基于业务和镜像delta”的更新，只传输镜像delta以保证小数据量下载，并且可以回滚。

Snappy Ubuntu Core从安全性考虑，引入AppArmpr来隔离应用的运行环境，并且它将会与Canonical自己基于LXC开发的轻量化Hypervisor(LXD)整合，使得VM化的容器成为可能。

此外，Snappy Ubuntu Core还支持Canonical自己的编排部署工具Juju Charms，提供对容器整个生命周期的管理。

可以说，Snappy Ubuntu Core是目前业界唯一支持ARM的容器操作系统。

5.3.4 VMware Photon

Docker使用容器这种轻量级虚拟化技术部署应用，随着Docker的越来越流行，对VMware将会造成潜在威胁。这是因为容器可以不需要虚拟机而直接运行在裸机上，虽然目前由于安全性还有不足，公有云上的容器仍然跑在虚拟机里。

虽然VMware在2014年8月宣布了与Docker的合作伙伴关系，且宣称会帮助Docker构建一个真正可扩展的系统，后续也在其产品中对与Docker相关的网络和存储等做了优化和增强。但是，应用在操作系统之上的容器中运行，是VMware无论如何也无法面对的痛点，因为VMware没有操作系统，或者就VMware目前的技术软件栈来说，需要在Hypervisor之上的系统中提供管理，也就是在操作系统中实施管理。当然，VMware也不可能不允许CoreOS、Red Hat或者其他操作系统在Hypervisor上运行，而独自在Hypervisor中提供对容器的支持。这就需要一个特别的操作系统来帮助把容器加到VMware产品里，并且需要扩展Hypervisor到操作系统里。于是，在2015年4月VMware推出了容器操作系统Photon，它紧密地与vSphere生态集成，并做了相应的优化，定位于VMware产品线。它具备以下特性。

- 支持多种主流容器——Docker、Rkt和Garden。
- 通过虚拟机技术或者集成Lightwave的授权与鉴权机制来提高容器运行的安全隔离性。
- 支持全新的、开源的兼容Yum的包管理器。

5.3.5 Red Hat Atomic Host

RHEL 7 Atomic Host在2015年3月发布，相比RHEL 7完全发行版的6500个外围包，它只有不到200个，总大小约为400MB。

它比传统操作系统更轻量，但体积并没有其他同类产品小。其主要考量的是，尽管RancherOS能做最小化的构建，但是可能会带来更多的复杂度，因为除了应用容器之外，还需要运行额外的系统容器，所以核心系统服务需要在主机里。Red Hat针对系统行为的数据收集、系统日志和身份验证等也开发了容器化版本，但是坚持认为这些组件必须放在主机上。

Atomic Host是从经过Red Hat工程团队严格验证的企业级操作系统派生而来的精简系统。Red Hat对Atomic Host也在做ISV(Independent Software Vendors)的工作，认证容器是基于知名的、测试过的、安全的基础镜像构建的。

Atomic Host采用Selinux提供强大的多租户环境下的安全防护。此外，Atomic Host支持Docker，也预装了Kubernetes等集群管理配置工具。

5.3.6 Microsoft Nano Server

Nano Server是微软的容器操作系统，约400MB大小，还处于早期阶段。它定位在两个场景上，一是聚焦Hypervisor集群和存储集群等云基础设施，二是着重于原生云应用的支持。对于第二种场景，Nano Server适配新类型的应用，这些应用在全新的基于Azure的开发环境中开发，并部署在Azure上面，而不是在传统的基于客户端的Visual Studio中。这些新的应用为Windows开发者通往容器领域提供了入门通道。

开发者为Nano Server开发的程序，可完全与已有的Windows服务器安装版本兼容，因为Nano Server实际上是Windows服务器版的子集。

给Nano Server写的应用可以运行在物理机上、虚拟机上或者容器里。有两类容器可在Windows服务器和Nao Server上工作，一类是为Linux开发的Docker容器，另一类是微软为自己的Hypervisor平台Hyper-V开发的容器。这些容器提供了额外的隔离，能真正用于像多租户服务或者多租户PaaS等场景。

Nano Server支持多种编程语言和运行时，如C#、Java、Node.js和Python，也支持Windows服务器容器和Hyper-V容器，预装了Chef等集群管理配置工具。

微软从Nano Server入手，拥抱容器，给开发者提出了挑战。开发者可能需要一个比较长的转变期来接受并习惯微服务理念。

5.4 Docker容器资源管理调度和应用编排

业界当前主要有三种容器集群资源管理调度和应用编排的不同选择。

- Mesos生态：核心组件包括Mesos容器集群资源管理调度以及不同的应用管理框架。典型的应用管理框架包括Marathon和Chronos，其中Marathon用来管理长期运行服务，如Web服务；Chronos用来管理批量任务。Mesos生态主要由Mesosphere、Twitter等公司主力推动。
- Kubernetes生态：Google公司发起的社区项目，涵盖容器集群资源管理调度，以及不同类型应用的应用管理组件。例如副本可靠性管理，服务发现和负载均衡，灰度升级，配置管理等组件。
- Docker生态：Docker公司希望向容器生态系统上层发展，推出了Swarm容器资源管理调度组件，以及Compose应用编排组件。

下面章节进一步详细介绍了上述三种容器资源管理调度和应用编排的不同选择。

5.4.1 Mesos生态

现代企业的数据中心运行不同类型的分布式服务，如Web服务、NoSQL数据库、大数据分析任务等，这些应用往往运行在彼此隔离的物理集群上，造成资源彼此无法共享，资源利用率下降。

Mesos通过对数据中心资源的统一管理和分配，把整个数据中心抽象为一台大的计算机，使开发者不用关心资源的管理，只聚焦应用的开发，从而提高了应用的开发效率。同时，其打破了数据中心不同应用彼此独立的资源烟囱，实现了多应用框架共享Mesos统一管理的资源池，提升了资源利用率。

Mesos生态的核心由Mesos资源管理分配框架，以及运行其上的不同的应用框架组成。

Mesos两阶段资源分配

Mesos资源管理和分配框架采用主-从(Master-Slave)模式，其中Master节点(控制节点)负责集群资源信息的收集和分配，Slave节点(工作节点)负责上报资源状态，并执行具体的计算任务。

Mesos资源管理和分配过程如下(见图5-6)。

(1) 工作节点1向控制节点上报空闲资源状态，例如<s1，4cpu，4GB>。

(2) 控制节点根据资源分配策略，决策应该向哪个应用框架(Framework)提供资源，以及提供多少。例如，控制节点决策向应用框架1提供工作节点1上报的全部资源。Mesos中该消息成为资源提供(Resource Offer)，如<s1，4cpu，4GB>。

(3) 应用框架的调度器决策是否接受控制节点发送的资源提供，应用框架同时负责接收和调度具体的工作任务。假设应用框架1接收资源提供，并把两个任务调度到工作节点1上。例如应用框架1返回的响应如下：<任务1，s1，2cpu，1GB>，<任务2，s1，1cpu，2GB>。

(4) 最后，Mesos控制节点把上述响应发送给工作节点1，工作节点1为应用框架1的执行器分配所需资源，执行器启动工作任务。控制节点发送给工作节点1的响应如：<FW1，任务1，2cpu，

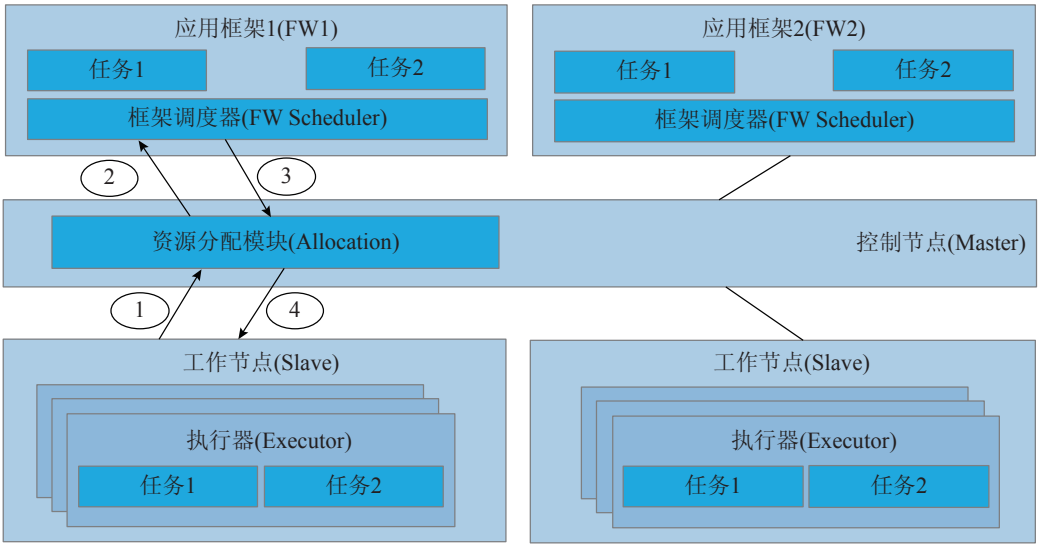


图5-6 Mesos工作原理

1GB>，<FW1，任务2，1cpu，2GB>。

经过上述资源分配，控制节点1发送的资源提供还有<1cpu，1GB>资源剩余，控制节点可以继续把该剩余资源提供给应用框架1或者应用框架2。

Mesos采用两阶段资源调度。Mesos控制节点负责根据不同的资源分配策略和算法，通过资源提供(Resource Offer)给已经注册的不同应用框架分配资源，完成第一阶段资源分配。应用框架的调度器负责第二阶段资源调度：把本框架的不同任务调度到不同的资源提供上。Mesos的两阶段资源调度实现了集群调度规模的可扩展性，以及应用框架资源调度策略和算法的灵活性。

Marathon容器应用管理平台

Marathon是MesosSphere公司开发的一个容器应用管理平台，面向Long-Running应用的部署和管理。

Marathon容器应用管理平台的核心能力具体如下。

- 架构HA：Marathon通过ZooKeeper支持Active-Passive集群部署，保证高可用。
- 支持不同类型的容器运行时(runtime)：包括Mesos容器(采用cgroups)和Docker容器。
- 支持有状态应用：支持给应用绑定持久化存储，从而可以用来部署数据库等类型的应用。
- 调度器：支持调度约束，比如限制一个节点智能部署一个应用实例副本。
- 健康检查：支持采用HTTP或者TCP检查应用实例的健康状态。
- 事件订阅：支持向一个HTTP端点进行事件通知，比如和一个外置的LB集成。
- 监控：支持向监控系统，如Graphite、Datadog上报监控指标。
- 灰度升级：支持应用部件灰度升级。
- 负载均衡和服务发现：支持不同类型的负载均衡和服务发现机制。

5.4.2 Kubernetes生态

Kubernetes是Google公司在2014年6月宣布开源的容器资源管理和应用编排引擎。Google公司内部大量使用了容器承载数据中心不同类型的应用负载，如搜索业务、Gmail、大数据等，积累了丰富的容器使用经验。Kubernetes是基于Google内部的容器集群管理系统Borg演变而来，感兴趣的可以阅读Google发表的Borg、Omega以及三者关系比较的相关论文，了解其功能定位和

演进历史。

一、Kubernetes介绍

本节先介绍Kubernetes系统架构，常见对象模型，然后对Kubernetes容器调度过程，以及引用管理能力进行分析说明。

二、Kubernetes架构

Kubernetes是Google开源的容器集群管理系统，提供应用部署、维护、扩展机制等功能，利用Kubernetes能方便地管理跨机器运行容器化的应用，主要功能如下：

- 使用Docker对应用程序打包(Package)、实例化(Instantiate)、运行(Run)；
- 以集群的方式运行、管理跨机器的容器；
- 解决Docker跨机器容器之间的通讯问题；
- Kubernetes的自我修复机制使得容器集群总是运行在用户期望的状态。

架构设计上Kubernetes采用典型的主-从结构，希望构建为一个可插拔组件和层的集合，具有可替换的调度器、控制器、存储系统。

Kubernetes架构核心组件功能，如表5-1所示。

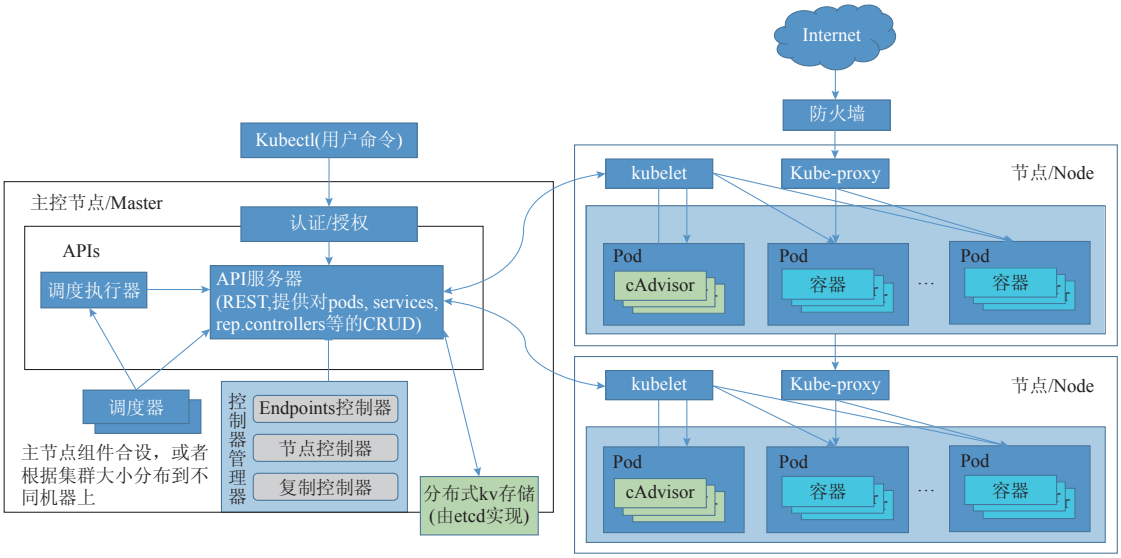


图5-7　Kubernetes架构

表5-1　Kubernetes架构核心组件功能

平面	组件	功能描述
控制平面(Master)	API服务器	主要提供Kubernetes API，提供对Pods、Services、ReplicationController等对象的生命周期管理； 处理REST操作，验证它们，在etcd中更新相应的对象； API不仅仅是面向最终用户的，也是面向工具和扩展开发者的，是开放生态系统的基础
	调度器	Scheduler负责Pods在各个节点上的分配；Scheduler是插件式的，Kubernetes将来可以支持用户自定义的Scheduler
	控制器管理器	所有其他的集群级别的功能目前由Controller Manager提供； Endpoints对象由端点控制器创建和更新，节点控制器发现、管理和监控节点； Kubernetes将来可以把这些控制器拆分并提供可插拔的组件
	分布式kv存储(ETCD)	所有的持久性状态都保存在etcd；etcd支持watch，这样组件很容易得到系统状态的变化，从而快速响应和协调工作

(续表)

平面	组件	功能描述
数据平面(Node)	Kubelet	节点代理，Kubelet管理Pod生命周期，以及Pod的容器、镜像、卷等
	Kube-Proxy	Kube-Proxy是一个简单的网络代理和负载均衡器； 它具体实现Service模型，每个Service都会在所有的Kube-Proxy节点上体现； 根据Service的Selector所覆盖的Pods，对这些Pods做负载均衡来服务于Service的访问者

三、Kubernetes对象模型

理解Kubernetes中如下几个常见概念，有助于我们深刻理解整个系统。

✍ 集群(Cluster)：物理机或者VM的集合，是应用运行的载体。

✍ 节点(Node)：可以用来创建容器集(Pod)的一个特定的物理机或者VM。

✍ 容器集(Pod)：Kubernetes中的最小资源分配单位，一个Pod是一组共生容器的集合。共生指一个Pod中的容器只能在同一个节点上。

✍ 服务(Service)：一组Pod集合的抽象，比如一组Web服务器；服务具有一个固定的IP或者DNS，从而使得服务的访问者不用关心服务后面具体Pod的IP地址。

✍ 复制控制器(RC，ReplicationController)：Kubernetes通过RC确保一个Pod在任何时候都维持在期望的副本数。当Pod期望的副本数和实际运行的副本数不符时，RC调用Kubernetes接口创建或者删除Pod。

✍ 标签(Lable)，标签选择器(LabelSelector)：即与一个资源关联的键值对，方便用户管理和选择资源。资源可以是集群、节点、Pod、RC等。

四、Kubernetes容器调度

Kubernetes的调度器是Kubernetes众多组件的一部分，独立于API服务器之外。调度器本身是可插拔的，任何理解调度器和API服务器之间调用关系的工程师都可以编写定制的调度器。本文后面的介绍将聚焦Kubernetes的默认调度器。

如前所述，Kubernetes的调度器和API服务器是异步工作的，他们之间通过HTTP通讯。调度器通过和API服务器建立List&Watch连接来获取调度过程中需要使用的集群状态信息，例如节点的状态、Service的状态(用于Service内Pod的反亲和)、Controller的状态、所有未调度和已经被调度的Pod的状态等。调度器工作步骤具体如下。

✍ 从待调度的Pod队列中取出一个Pod。

✍ 依次执行调度算法中配置的过滤函数(Predicate)，得到一组符合Pod基本部署条件的节点的列表。过滤函数是一些“硬约束”，例如资源是否足够，Pod要求的Label是否满足等。

✍ 对上一步骤中得到的节点列表中的节点依次执行打分函数(Prioritizer)，为各个节点进行打分。每个打分函数输出一个0～10之间的分数，最终一个节点的得分是各个打分函数输出分数的加权和(每个打分函数都有一个权值)。

✍ 对所有节点的得分由高到低排序，把排名第一的节点作为Pod的部署节点(如果不唯一则在所有得分最高的节点中随机选择一个)，创建一个名为Binding的API对象，通知API服务器将被调度Pod的部署节点改为计算得到的节点。

目前Kubernetes的调度器支持多种维度的过滤和打分函数，考虑的因素包括但不限于：各个节点的Label(Pod可以通过LabelSelector指定自己希望部署在具有哪些Label的节点上)；基于Service的反亲和；Pod对指定节点的反亲和；持久化硬盘的挂载情况检查；节点的端口使用情况；指定节点名字的部署等。调度过程中还会考虑资源使用情况，注意这里资源使用情况不是实时的资源使用情况，而是Pod中的各个Container的Request字段所指定的资源数量之和，调度器考虑候选节点能否满足该Pod的Request资源请

求。关于Pod或Container的Request资源，请参考Kubernetes中关于Pod的QoS(Quality of Service)的介绍，此处不再赘述。

Kubernetes支持用户自定义调度算法，即可以通过模板配置使用哪些过滤和打分函数。用户也可以根据自己的需求编写相应的过滤或者打分函数作为调度函数库的一部分，并放到自定义调度算法中。除了自定义/编写调度算法，Kubernetes还支持Extender机制来进一步扩展调度逻辑，用户可以在系统中另外启动一个Scheduler Extender，其中可以包含其他自定义的过滤或者打分函数，每当默认调度器的过滤和打分函数执行之后，调度器可以分别调用(HTTP调用)Extender中的过滤和打分函数形成最终的调度结果。图5-8简单描述了调度器的调度过程。

Kubernetes调度器在调度过程中还会搜集调度的延时数据，为工程师提供数据支持，统计的延时数据包括以下几点。

- 端到端调度延时：从待调度队列中取出到Binding生效的间隔。
- 调度算法延时：从开始执行第一个过滤函数到计算得到最终部署节点的间隔。
- Binding生效延时：从调度器向API服务器发送Binding请求到收到回复成功(即Binding生效)的间隔。

除了上面提到的在调度算法库中添加新的函数和使用Extender外，Kubernetes还支持同时使用多个调度器来对不同类型的Pod进行调度。用户只需要在Pod的Annotation的中填写“scheduler.alpha.kubernetes.io/name：my-scheduler”便可以指定该Pod仅可以被名为my-scheduler的调度器调度，默认调度器或其他名字的调度器不会为Pod进行调度。

为了提高调度器的吞吐量，社区贡献者让调度器缓存一些集群信息来提高调度决策的速度，缓存的信息包括节点的资源信息，已部署Pod的信息等，另外充分利用Go语言的特性对过滤和打分过程进行并行处理。目前调度器可以达到至少支持数百个Pod每秒的调度吞吐量，具体数值和集群的规模和Pod数目有关。

未来，Kubernetes还会添加Re-scheduler来进一步强化Kubernetes集群资源配置的运行时优

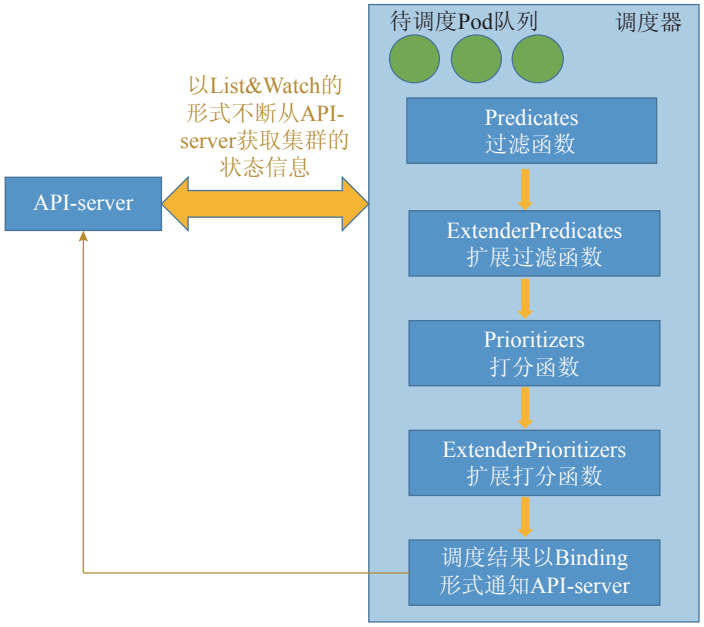


图5-8　Kubernetes调度器的结构和调度Pod的流程

化，与Kubernetes的调度器、QoS分类等一起，实现更加高效容器集群资源管理。

五、Kubernetes应用管理

除了容器资源管理和调度，Kubernetes另外一个核心价值是提供了针对不同类型应用管理的API接口集合，这些API集合把针对不同类型应用的管理能力分别到Kubernetes平台中。以Web业务(Long-Running类型应用)为例，提供了应用组件可靠性管理能力以及多副本管理能力、多副本之间的负载均衡能力、不同应用组件之间的服务发现能力、配置管理能力、灰度升级能力等。从而使得应用开发者直接使用上述能力开发应用时十分简单快捷，从而聚焦业务核心逻辑的开发。

Kubernetes提供了针对如下不同类型应用的管理能力。

- Long-Running应用：一旦应用启动，会长时间运行，如Web业务。提供应用组件可靠性保障、副本数保障、灰度升级、多组件间负载均衡等能力。
- 批量任务：提供任务创建、删除、更新、查询、状态跟踪等能力。
- DaemonSet类型应用：当用户部署一个Daemonset类型的应用时，Kubernetes在集群的每个节点上都部署一个Pod。典型的例子如日志、监控的代理程序的部署。
- PetSet类型应用：用来支持用状态应用，比如一个MySQL集群。具体管理能力如允许PetSet类型应用的不同组件独立挂载容器存储卷，提供不同组件间通信机制等。

上述不同类型的应用，对应一个不同类型的控制器管理器(Controller Manager)。用户可以根据自己的需求，开发特定类型的自定义控制器管理器。

5.4.3 Docker生态

一、Swarm介绍

Swarm项目是Docker公司发布三剑客中的一员，用来提供容器集群服务，目的是更好地帮助用户管理多个Docker Engine，方便用户使用，像使用Docker Engine一样使用容器集群服务。

Swarm这个项目名称特别贴切。在Wiki的解释中，Swarm behavior是指动物的群集行为。比如我们常见的蜂群、鱼群、秋天往南飞的雁群，都可以称作Swarm behavior。Swarm项目正是这样，通过把多个Docker Engine聚集在一起，形成一个大的Docker Engine，对外提供容器的集群服务。同时这个集群对外提供Swarm API，用户可以像使用Docker Engine一样使用Docker集群。

Swarm具有如下特点。

(1) 对外以Docker API接口呈现，这样带来的好处是，如果现有系统使用Docker Engine，则可以平滑地将Docker Engine切到Swarm上，无须改动现有系统。

(2) Swarm对用户来说，之前使用Docker的经验可以继承过来，非常容易上手，学习成本和二次开发成本都比较低。同时Swarm本身专注于Docker集群管理，非常轻量，占用资源也非常少。

(3) 插件化机制，Swarm中的各个模块都抽象出了API，可以根据自己一些特点进行定制实现。

(4) Swarm自身对Docker命令参数支持的比较完善，Swarm目前与Docker是同步发布的。Docker的新功能，都会第一时间在Swarm中体现。

1. Swarm架构

Swarm容器集群由两部分组成，分别是Manager和Agent，Swarm框架如图5-9所示，图的上半部分为Swarm Manager，在每一个节点会运行一个Swarm Agent。各个模块的具体作用如下。

(1) API：Swarm对外提供两种API，一种是Docker API，用于负责容器镜像的生命周期管理，另外一种是Swarm集群管理CLI，用于集群管理。

(2) Scheduler模块：主要实现调度功能，在通过Swarm创建容器时，会经过Scheduler模块选择出一个最优节点，里面包含了两个子模块，分别是Filter和Strategy，Filter用来过滤节点，找出满足条件的节点，Strategy用来在过滤出的节点中根据策略选择一个最优的节点，当然用户可以定制

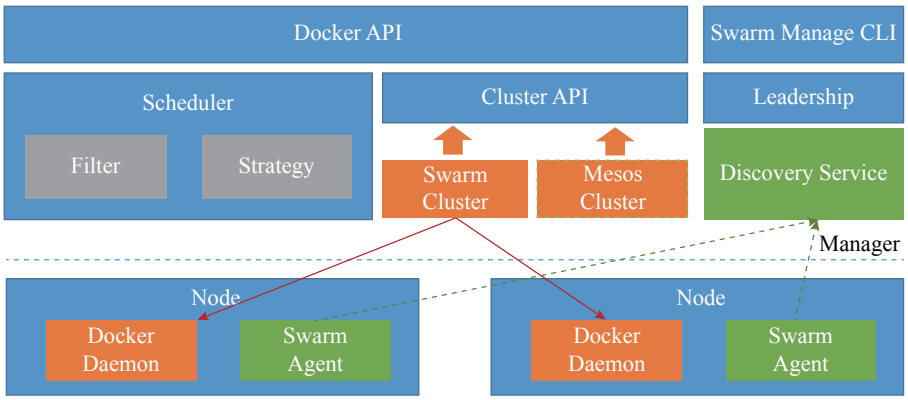


图5-9　Swarm架构

Filter/Strategy。

(3) Cluster API：Swarm对集群进行了抽象，抽象出了Cluster API，Swarm支持两种集群，一种是Swarm自身的集群，另外一种基于Mesos的集群。

(4) LeaderShip模块：用于Swarm Manager自身的HA，通过主备方式实现。

(5) Discovery Service 服务发现模块：用来提供节点发现功能。

(6) Swarm Agent：在每一个节点上，都会有一个Agent，用于连接Discovery Service，上报Docker Daemon的IP端口信息，Swarm Manager会直接从服务发现模块中读取节点信息。

2. Swarm集群管理

Swarm Manager CLI用于集群管理，通过如下三步就可将集群创建起来。创建完集群后，就可以使用Docker命令进行容器的创建。

- 通过swarm create命令创建一个集群ID。
- 通过swarm join命令将节点加入到集群。
- 通过swarm manage命令启动swarm manage。

3. Swarm容器调度

用户容器创建时，会经过调度模块选择一个最优节点。在选择最优节点过程中，分为两个阶段，即过滤和策略。

(1) 过滤

调度的第一个阶段是过滤，根据条件过滤出符合要求的节点，过滤器有以下五种。

- Constraints，约束过滤器，可以根据当前操作系统类型、内核版本、存储类型等条件进行过滤，当然也可以自定义约束，在启动Daemon的时候，通过Label来指定当前主机所具有的特点。
- Affnity，亲和性过滤器，支持容器亲和性和镜像亲和性，比如一个web应用，我想将DB容器和Web容器放在一起，就可以通过这个过滤器来实现。
- Dependency，依赖过滤器。如果在创建容器的时候使用了某个容器，则创建的容器会和依赖的容器在同一个节点上。
- Health filter，会根据节点状态进行过滤，去除故障节点。
- Ports filter，会根据端口的使用情况过滤。

(2) 策略

调度的第二个阶段是根据策略选择一个最优节点。其有以下三种策略。

- Binpack，在同等条件下，选择资源使用最多的节点，通过这个策略，可以将容器聚集起来。
- Spread，在同等条件下，选择资源使用最少的节点，通过这一个策略，可以将容器均匀分布在每一个节点上。
- Random，随机选择一个节点。

4. Swarm服务发现

在Swarm中，服务发现主要用于节点发现，

每一个节点上的Agent会将Docker Engine的IP端口注册到服务发现系统中。Manager会从服务发现模块中读取节点信息。Swarm中服务发现支持以下三种类型的后端。

- 第一种，是hosted discovery service，是Docker Hub提供的服务发现服务，需要连接外网访问。
- 第二种，是KV分布式存储系统，现在已支持etcd、ZooKeeper、Consul三种。
- 第三种，是静态IP。可以使用本地文件或者直接指定节点IP，这种方式不需要额外使用其他组件，一般在调试中会使用到。

5. Swarm HA机制

Swarm HA主要由Leadership模块实现，为了防止Swarm Manager单点故障，引入了HA机制，Swarm Manager自身是无状态的，所以很容易实现HA。实现过程中采用主备方式，当主节点故障以后，会从新选主提供服务，选主过程中采用分布式锁实现，现在支持etcd、ZooKeeper、Consul三种类型的分布式存储，用来提供分布式锁。当备节点收到消息后，会将消息转发给主节点。

二、Compose介绍

Docker Compose的前身是Fig项目，Fig是Orchard的一个产品，并很快成为Docker容器编排工具，采用Python编写，通过Apache2.0协议开源。Compose 2014年被Docker公司收购并成为官方支持的解决方案。

Docker Compose是Docker编排服务的三驾马车之一，Machine负责安装部署Docker，Swarm负责容器集群的管理，Compose负责容器应用的编排组织。Compos是支持定义由多个容器组成的应用并运行启动的工具。应用所涉及的容器规格以及容器之间网络、存储的配置由Compose规范进行描述，以YAML文件编写。通过简单的命令行，用户可以创建并启动由该YAML描述的应用。

1. Compose架构

Compose的代码实现中包括三种对象：container，service和project。其中，container指的就是容器；service是一组相同业务的container；project是对外提供服务的，由多个service通过link等关键字关联而成的功能单元。Compose的使用方式非常简单，如图5-10所示。首先，在docker-compose.yml中，定义构成该应用的容器运行时是哪些，这些容器之间的依赖关系是什么。接着，通过执行Docker-compose up 启动运行。之后通过compose命令行或者API可以对该应用的生命周期进行管理，包括应用的起、停、删、查等操作。

Compose代码的核心是一个YAML模板的解析器，基本流程如下。

- 运行docker-compose命令，Compose解析输入的YAML文件，并生成一系列的Docker指令。
- 通过调用下层Docker或者Swarm API，完成服务的创建以及关联，进而完成APP的部署。
- Compose提供一系列命令用来控制应用的整个生命周期管理，包括：启、停、重启service，查看service的运行状态等。

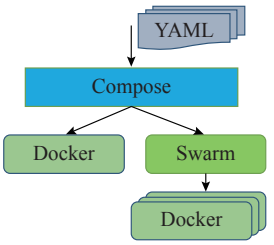


图5-10　Compose应用架构图

2. Compose示例

上面的YAML文件定义了一个包含4个容器的应用：第一个容器是一个redis容器，使用“redis：v1”镜像，并通过volumes参数将主机上的“/var/lib/docker/volumes/magento-redis”目录挂载到容器内部的“/data”目录下；第二个容器是一个mysql容器，使用“magento_mysql：5v19”镜像，并通过environment参数给该容器中注入了“BASE_URL：http：//192.168.103.218：8000/”这个环境变量，这个环境变量最后会被上层web应用读取，并用于拼接url；第三个容器是一个php容器，负责处理动态请求，该容器使用redis容器和mysql容器作为后端存储，因此该容器

的启动顺序应该位于redis和mysql容器之后，这种启动顺序的关系通过depends_on参数来描述，同时因为该容器需要防伪redis和mysql容器提供的服务，因此需要打通它们之间的网络，这种网络关联关系通过“links”参数来描述；第四个容器是一个nginx容器，负责处理静态请求，同时可以将动态请求转发到后端的php容器中，因此也使用了“depends_on”和“links”这两个参数来声明其与php容器之间的依赖关系，除此之外，由于该容器还需要对外提供服务，因此通过“ports”参数将主机的8000端口映射到容器的80端口，从而使得外部请求可以通过访问主机的8000端口来实现对本应用的访问(见图5-11)。

```
version: '2'
services:
    redis:
        image:redis:v1
            restart: always
            volumes:
             -"/var/lib/ docker/volumes/ magento -redis:/data"
    mysql:
        image:magento_mysql:5v19
        restart:always
        environment:
          BASE_URL:http://192.168.103.218:8000/
    php:
        image:magento_php:v4.8
          restart: always
          depends_on:
            - redis
            -mysql
          links:
            -redis
            -mysql
ngx:
  image: magento_nginx:v10
        restart:always
  depends_on:
      - php
  links:
      - php
  ports:
      - "8000:80"
```

图5-11　Compose示例

5.5　Docker容器与软件定义计算的集成

容器出现后，OpenStack社区积极提供和容器集成不同的服务能力，一方面让用户基于OpenStack更加方便、自动化地使用容器服务，同时提供容器和虚拟机统一管理的存储和网络方案。本节介绍了OpenStack开源社区中与容器相关的几个重点项目。

5.5.1　Magnum介绍

Magnum是OpenStack社区推出的用于部署和管理容器集群的项目。用户可以很方便地通过Magnum来部署和管理Kubernetes、Swarm和Mesos集群(见图5-12)。

Magnum由Magnum API和Magnum Conductor两个部件构成。Magnum API负责处理用户的请求。一些繁重的任务，比如与Heat进行交互创建集群、则由Magnum Conductor来执行。Magnum API和Magnum Conductor之间通过消息队列进行通信。数据库中保存着Bay和BayModel的状态信息。Magnum创建集群的过程是通过Heat模板来完成的，不同的集群对应着不同的模板。

Magnum包括两个对象，即BayModel和Bay。

- BayModel：BayModel主要定义了容器集群的一些规格，例如这个集群有多少个控制节点，多少个计算节点，以及控制节点和计算节点使用的镜像和资源规格等。
- Bay：Bay代表一个容器集群。目前可以通过Magnum创建Kubernetes、Swarm、Mesos三种类型的Bay。每一个Bay关联着一个BayModel。

Magnum希望将OpenStack提供的I层能力与容器进行深度结合，为容器提供网络、存储等能力。Magnum功能还在持续开发中，当前支持的主要功能如下：

- Magnum目前能部署Kubernetes、Swarm、Mesos三种类型的容器集群；
- 支持容器集群的自动伸缩；
- 支持本地Docker镜像仓库；
- 支持在容器集群中使用Cinder的卷和neutron的LBaaS。

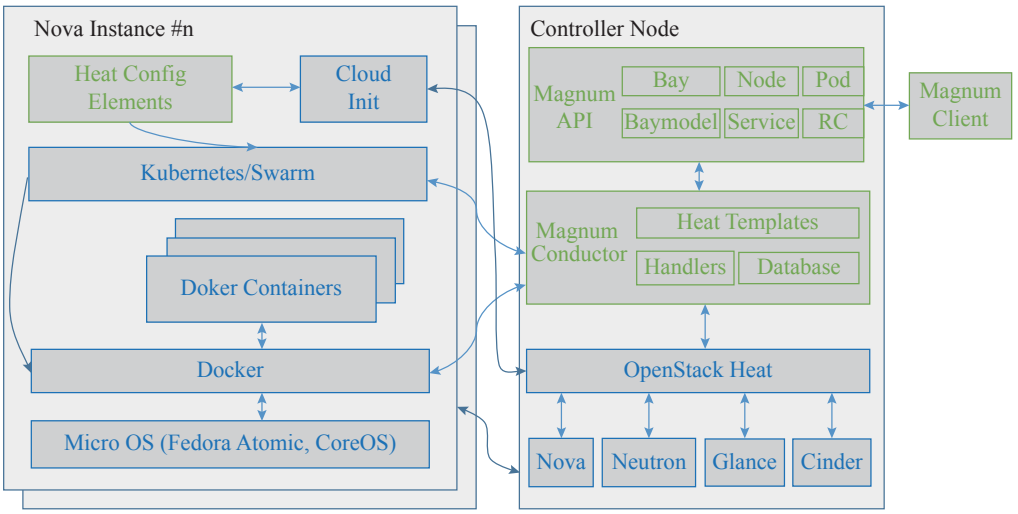


图5-12 Magnum架构

5.5.2 Murano介绍

Murano是OpenStack的Application Catalog服务，推崇AaaS(Anything-as-a-Service)的概念，通过统一的框架和API实现应用程序快速部署和应用程序生命周期管理的功能，降低应用程序对底层平台(OpenStack层和虚拟化层)的依赖，其架构如图5-13所示。

Murano包括以下几个部分。

- murano核心服务：包含了Murano API server、Murano engine和MuranoPL。
- murano-agent：运行在客户虚拟机并执行部署计划。
- murano-dashboard：Murano UI提供了OpenStack的仪表板插件。
- murano-client CLI：即Murano的客户端库和命令行。

Murano使用OpenStack的服务，使用REST API和OpenStack服务交互。

Murano通过AMQP队列，远程操作部署在用户服务器上的Murano agent，确保基础设施和服务器被隔离。

Murano的主要特性概况如下。

(1) 应用程序目录

- 以图标的方式显示应用程序，并支持拖放选择和部署；
- 应用可以分类，可以定制配置界面；
- 自动生成应用的网络拓扑图。

(2) 应用程序目录管理

- 上传应用提供了UI和命令行方式，支持本地zip文件、URL和包名称多种方式；
- 分类组织应用，可以更新应用的名称、描述和标签。

(3) 应用程序生命周期管理

- 简化配置和应用集成，可以定义应用之间的依赖关系，新的应用可以使用已经存在的服务；
- 支持HA和自动缩放；
- 相同环境的应用之间可以交互，不同租户之间的应用隔离。

5.5.3 Kuryr介绍

Kuryr实现了Docker网络组件Libnetwork的一个远程网络插件，Kuryr通过把Libnetwork的调用映射到OpenStack Neutron网络上，实现了Libnetowrk CNM(容器网络模型)和Neutron网络模型之间的转换，从而让容器可以使用OpenStack

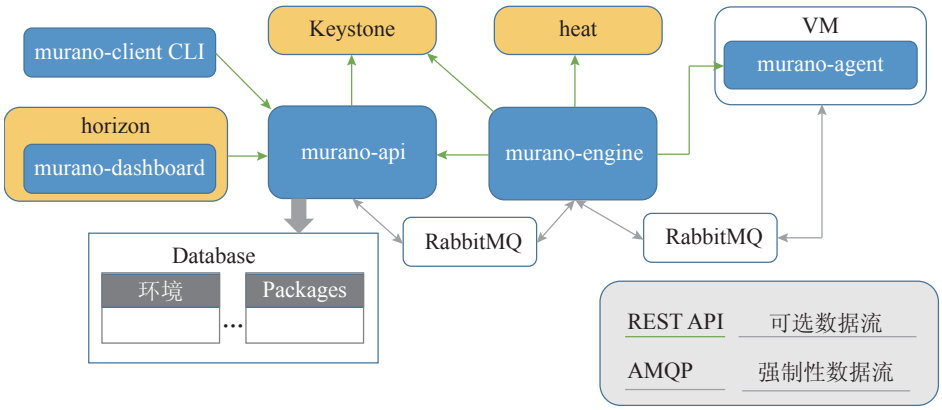


图5-13　Murano架构

Neutron网络。

Kuryr一词来自捷克语，意思是“信使”，旨在成为连接Docker和Neutron两个社区的信使和桥梁，从而弥补当前容器网络方案不成熟，快速利用Neutron能力提供容器网络方案。

当前容器生态中，Docker和Kubernetes分别抽象了不同的网络模型，即CNM(Container Network Model)网络模型和CNI(Container Network Interface)网络模型。Kuryr分别提供给了接入两种网络模型的能力。如图5-14所示，Kuryr当前实现了Libnetwork网络插件和IPAM(IP地址管理)插件，从而可以实现用Kuryr支持Docker或者Docker Swarm的网络能力。

Kuryr正在开发接入Kubernetes网络的能力。如图15-4所示，Kuryr Raven通过向Kubernetes中API服务订阅事件变化，获取网络相关的状态变化，并把这些状态变化转化成对Neutron API的调用，从而构建基于Neutron的虚拟网络。当获取Neutron相应后，Raven负责把返回的信息添加到Kubernetes对象上，如把IP、端口信息添加到Kubernetes Pod对象上。

Kuryr同时实现了Kubernetes CNI驱动，从而实现为每个Kubernetes工作节点上面的Pod设置网络信息。

Kuryr通过Neutron Client实现对Neutron API的调用。

图5-14　Kuryr核心组件

(注：摘自Gal Sagie技术博客，Gal是Kuryr核心贡献者，华为以色列研究所网络专家)

Kuryr通过Keystone Client调用Keystone实现请求认证。

5.5.4 Fuxi介绍

Fuxi项目旨在将OpenStack Cinder卷存储、Manila文件存储、Swift/S3提供给Docker容器使用，作为容器的持久化存储，使Docker可以重用Cinder和Manila提供的高级功能(如快照、备份)和丰富的第三方厂商设备接入。

Fuxi实现标准Docker remote volume plugin API，便于Docker deamon/Swarm接入，同时提供Fuxi Client插件接入Kubernetes(见图5-15、图5-16)。

Fuxi包括以下几个部分。

(1) API

API提供标准Docker remote volume plugin api，包括：

- /VolumeDriver.Create
- /VolumeDriver.Remove
- /VolumeDriver.Mount
- /VolumeDriver.Path
- /VolumeDriver.Unmount
- /VolumeDriver.List
- /VolumeDriver.Get
- /Plugin.Activate(handshake)。
- Controller：接受来自API的请求，执行存储的创卷、删除、mount和unmount等能力，并调用用户设定的Storage provider plugin和

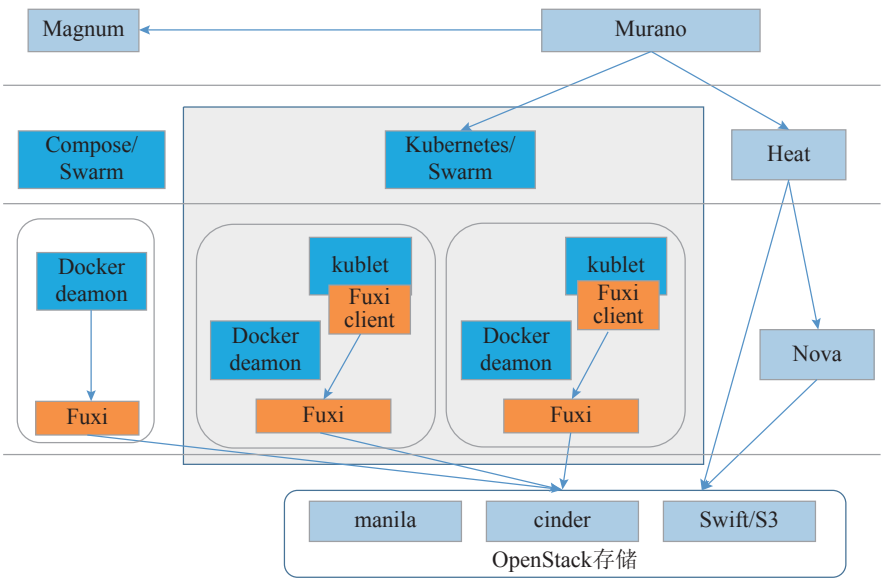


图5-15 Fuxi架构

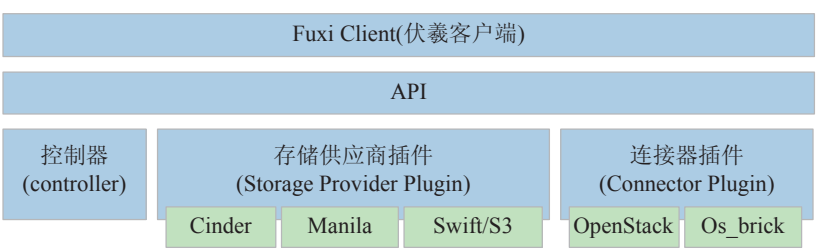


图5-16 Fuxi API

Connectors plugin执行具体的动作。

(2) Storage provider plugin

其提供标准API供用户接入插件，当前提供Cinder、Manila、Swift/S3插件，与OpenStack对象组件交互，为容器提供卷、文件、对象存储。

(3) Connectors plugin

其提供存储挂载到Docker Deamon节点插件，提供Cloud挂载方式和裸机挂载方式。

第 6 章

分布式软件定义存储概述

6.1 分布式软件定义存储

6.1.1 分布式软件定义存储驱动力

企业级存储中使用的传统SAN、NAS设备，在云计算中面临了很多的问题，主要问题如下。

(1) 存储弹性问题

企业级存储无法满足多业务不同负载、动态的资源变化需求，在不同租户和不同应用对资源有不同要求的时候，很难方便地做出调整，包括性能和容量资源的弹性调配，而云计算中多租户多业务负载下资源的弹性是极其重要的核心要素。

(2) 存储扩展问题

传统存储的扩展性面临了多个瓶颈，如机头、前后端网络、磁盘与CPU/MEM资源不同步扩展等，都是传统存储无法做到线性扩展的几个关键因素。

(3) 形态和实施的成本、复杂性问题

传统存储在部署的时候，需要独立的存储网络，用于多主机互联，特别是针对性能较高的FC网络，在实施的时候成本较高、组网实施复杂，不利于大规模集群的简化部署实施。

(4) 大规模集群下的容错和可靠性问题

在规模很大的云计算环境下，需要具备跨机房、跨机柜、跨服务器的数据保护机制，即使在机柜故障等场景下，数据仍然不丢失，仍然可访问。

(5) 灵活的软件定义策略问题

在云计算环境下，不同的租户、不同的业务应用对存储有着不同的要求，需要底层存储具备灵活的软件定义策略支持，允许用户按需进行存储的策略配置(如定义多大的容量、多少IOPS、多大的SSD Cache缓存、什么样的数据冗余和可靠性要求等)，底层存储可以根据这些软件定义策略进行资源的调配，按需自动地满足上层业务和应用的需求。

云计算的存储虚拟化概述如图6-1所示，其中包含了传统存储的虚拟化、分布式存储的池化和加速以及软件定义的存储策略控制三个部分。

6.1.2 软件定义存储概述

软件定义存储(Software Defined Storage，简称SDS)，业界不同的组织都有各自的定义，目前还没有一个统一的标准。IDC给出软件定义存储

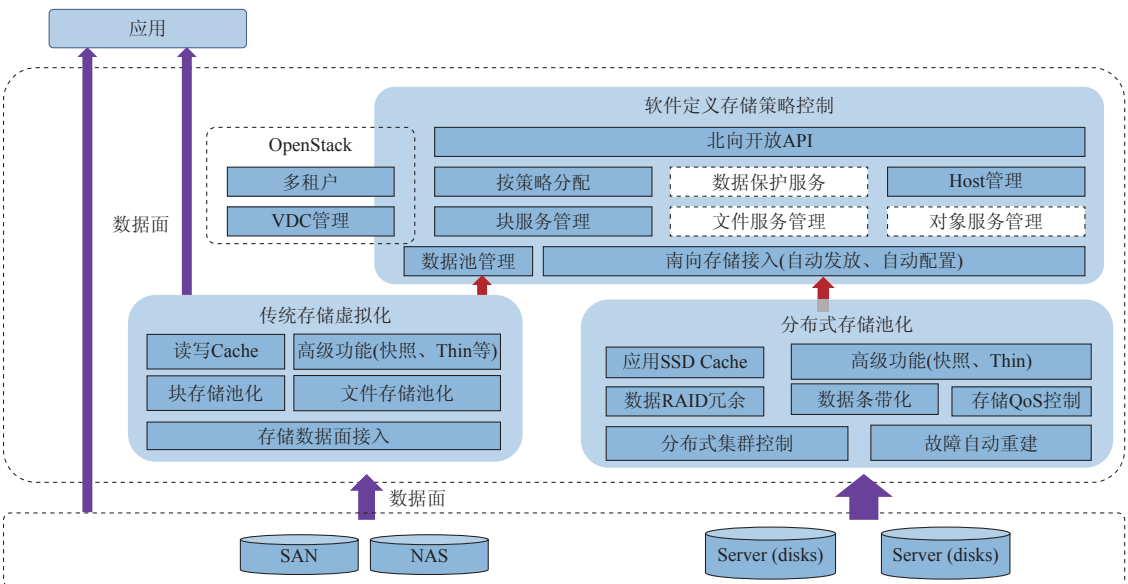


图6-1 存储虚拟框架

(SDS)的定义：可以安装在商用资源(x86硬件、虚拟机监控程序或者云)和/或者现有计算硬件上的任何存储软件堆栈。此外，为了取得资格，基于软件的存储堆栈应该提供一套完整的存储服务，还有在基础的持续数据配置资源之间的联邦，这使其租户的数据可以在这些资源之间流动。

软件定义存储的需求模型如图6-2所示。

从租户视角，其典型操作包括：

- 根据服务目录操作高级服务；
- 查询卷、文件、对象存储的容量；
- 卷、文件、对象存储的增删改查；
- 主机挂载、卸载卷和文件目录。

从管理员视角，其典型操作包括：

- 查询TOPO接连关系；
- 分配物理设备到租户视角的Cell以及Zone；
- 分配物理空间给数据池；
- 存储设备与数据池OM；
- 主机安装Agent；
- 服务目录管理；
- 租户权限管理。

在功能上，软件定义存储对租户屏蔽物理存储设备、对管理员提供将物理设备映射到逻辑概念的手段。在架构上，对上提供补充存储类服务操作到数据中心服务目录，对下集成设备管理信息。

软件定义存储有以下的价值。

- 完全通过软件实现存储的高级特性：快照、克隆、瘦分配、高速缓存等都不依赖于存储设备。
- 策略驱动的设计：传统存储无法与应用配合，性能低下，基于策略的存储，能够为不同的应用提供不同的QoS。
- 简化存储管理：可以实现一次配置，多次使用，计算管理员只需在系统初始配置或者扩容时需要存储管理员的参与，后续应用需要存储资源时，能够做到及时分配。

6.1.3 分布式存储概述

随着企业面临的竞争环境越来越激烈、新业

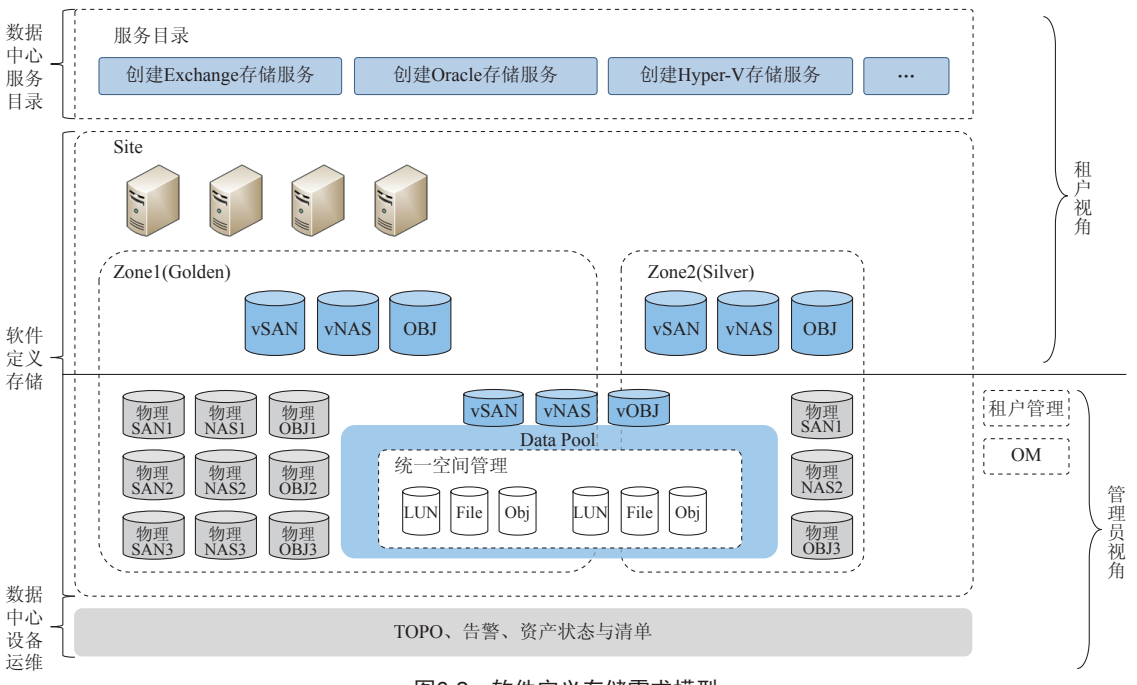


图6-2 软件定义存储需求模型

务上线时间要求越来越短，其IT系统需要从传统的成本中心转变为提升企业竞争力的利器，帮助企业提升竞争力并实现商业成功。作为存放企业数据资产的存储系统，不但要满足业务所需要的高性能、高可靠等基本诉求，更要满足未来业务的发展、提升业务的敏捷性，帮助业务更快更好地适应竞争环境的需要。

计算与存储在过去20年一直在非均衡地发展。摩尔定律设想单位面积晶体管数量每18月增加1倍，对应单位价格的计算性能将翻2倍以上。回顾过去20年：处理器和网络带宽分别提升了3000倍和1000倍，但磁盘和内存带宽仅提升120倍，远落后于摩尔定律。阿姆达尔定律认为，计算系统中对某一部件采用更快执行方式所能获得的系统性能改进程度，取决于这种执行方式被使用的频率，或所占总执行时间的比例。对于多数应用而言，基本均属于CPU计算与内/外部存储(Mem/Disk)的串联模型。换言之，系统中最慢部分(存储)的效率将决定和制约整个系统的效率。

在云计算集中化数据中心资源池环境中，由于GE以太网络的延伸作用，远端RAM/SSD的容量与本地存储相差不超过1个数量级，数据访问时延则比本地HDD减少30倍，带宽降低1倍。

从20世纪80年代到21世纪的前10年，计算与存储经历了一次分离的变革。这次分离是由计算与存储的性能发展差距导致的。基于晶体管的计算与基于机械硬盘介质的处理性能差距越来越大，以及存储数据的重要性不断增加，为便于提升资源利用率，终于导致了计算、存储的架构分离，各自进行资源最优配置(见图6-3)。

下面我们介绍发生在世纪之交的这次计算、存储分离带来的其他优势。

- 应用计算的高可靠HA迁移能力：通过多个计算实例共享相同存储，在计算节点发生故障后，无须耗时的外存储数据迁移，即可快速在其他计算节点上恢复故障应用。
- 计算处理逻辑与应用数据分离，使得计算节点的更新(软硬件升级)不影响数据的可获得性。
- 计算与存储各自资源利用率最大化及技术独立发展：计算与存储各自独立组成水平资源池，存储资源池按需向计算应用实例供给存储资源。

随着企业业务的不断发展，特别是互联网突飞猛进的发展，这种计算/存储分离架构面临的挑战和问题越来越突出，具体表现在如下几个方面。

- 计算与存储物理架构上的人为分割，导致系统成本居高不下：目前业界主流的磁盘阵列软硬件普遍价格昂贵，且磁盘阵列自身的策略管理配置非常复杂，维护成本居高不下，尤其是在缺乏IT专业人员的场景(如行业分支机构、SME等)下，由人工误操作导致的业务中断，其风险较高。
- SAN机头成为可能制约系统扩展性的单点瓶颈：随着计算集群规模的不断扩展，由于存储资源池集中式控制机头的存在，使得共享存储系统的可扩展性及可靠性受到制约；如采用多个SAN系统，则会导致各独立SAN系统之间存储无法共享。
- SSD存储介质的引入，使得SAN控制机头可能成为系统性能瓶颈，SSD与归属计算节点CPU/MEM之间的最高效连接方式应为PCI-E，如果将SSD按照传统存储模式集中部署，则控制机头有可能成为CPU与SSD间高吞吐I/O带宽的瓶颈，并且系统复杂度更高。
- 集中式SAN控制机头可能成为影响系统整体可靠性的单点故障风险点：在虚拟化服务器整合环境下，成百上千VM共享同一存储资源池，一旦磁盘阵列控制器发生故障，将导致整体存储资源池不可用。尽管SAN控制机头自身具有主备机制，但依然存在异常条件下主备同时故障的可能性。
- 集群组网环境下，各计算节点的内存/SSD作为分层存储的缓存彼此孤立，只能依赖集中存储机头内的缓存实现I/O加速：共享存储的集群内各节点Cache容量有限，但不同节点Cache无法协同，且存在可靠性问题，导致本可作为集群共享缓存资源的容量被白白浪费。
- 虚拟化技术迅猛发展，虚拟机技术给服务器带来更高的利用率、给业务带来更便捷的部

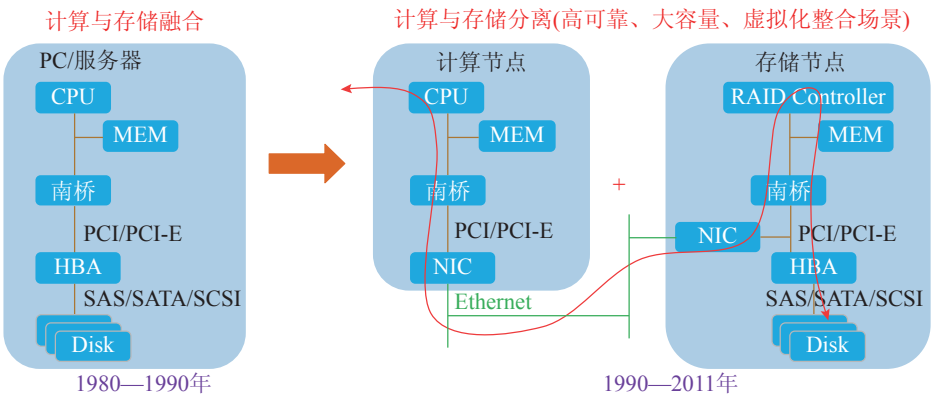


图6-3 存储由合到分的历史

署，降低了TCO，因而在众多行业得到了广泛的应用。与此同时，虚拟机应用给存储带来以下挑战：第一，相比传统的物理服务器方式，单个存储系统承载了更多的业务，存储系统需要更强劲的性能来支撑；第二，采用共享存储方式部署虚拟机，单个卷上可能承载几十或上百的虚拟机，导致卷I/O呈现更多的随机特征，这对传统的Cache技术提出挑战；第三，单个卷承载多个虚拟机业务，要求存储系统可协调虚拟机访问竞争，保证对QoS要求高的虚拟机获取到资源实现性能目标；第四，单个卷上承载较多的虚拟机，需要卷具有很高的I/O性能，这对传统受限于固定硬盘的RAID技术提出挑战；第五，虚拟机的广泛使用，需要更加高效的技术来提高虚拟机的部署效率，加快新业务的上线时间。

面对这些挑战，正所谓“合久必分、分久必合”的哲学规律在IT领域同样上演。针对大多数企业事务型IT应用而言，关注核心在于信息数据的“即时处理”而非“存储/归档”，因此存储向计算的融合再次符合企业业务应用的根本诉求。通过引入Scale-out存储机制，可实现服务器集群环境下DAS直连硬盘的资源池化和虚拟化，推动计算与存储从“物理分离”架构向“物理”融合与“逻辑”分离相结合架构的演进，实现以大一统融合架构形态实现对典型企业IT应用整合及性价比最优化支撑(见图6-4)。

在这种Scale-out架构与计算和网络融合后，便形成了一种更加高效的一体化分布式存储与分布式计算架构(见图6-5)。

一体化分布式存储架构具有以下鲜明特点。

一体化系统内，设置Layer1～Layer3共三层信息存储，基于分布式存储软件引擎完全水平拉通，且支持基于强一致的跨服务器数据可靠型。

- Layer1 Storage(内存)：时延100 ns (本地)～100 us (远地)
- Layer2 Storage(SSD)：时延10 us (本地)～300 us (远地)
- Layer3 Storage(DAS)：时延 5ms (本地)～10 ms (远地)
- Layer4 Storage(SAN)：时延 5ms (本地)～10 ms (远地)

通过上述各层Storage的热点数据读写推至更上一层Storage，实现数据I/O吞吐及整个系统性能的大幅提升。

Layer1 Storage(内存) 尽管吞吐率及时延优势明显，但容量和功耗成本过高，因此对于多数应用只能作为Cache，必须基于Cache算法(FIFO、LRU、LFU等)管理内存与下一级存储(SSD/DAS/SAN)之间的读写数据刷新。

Layer2 Storage(SSD)既可以作为Cache(针对数据量超出分布式Cache容量的场景)，也可以作为最终存储(纯SSD，针对数据容量不大的场

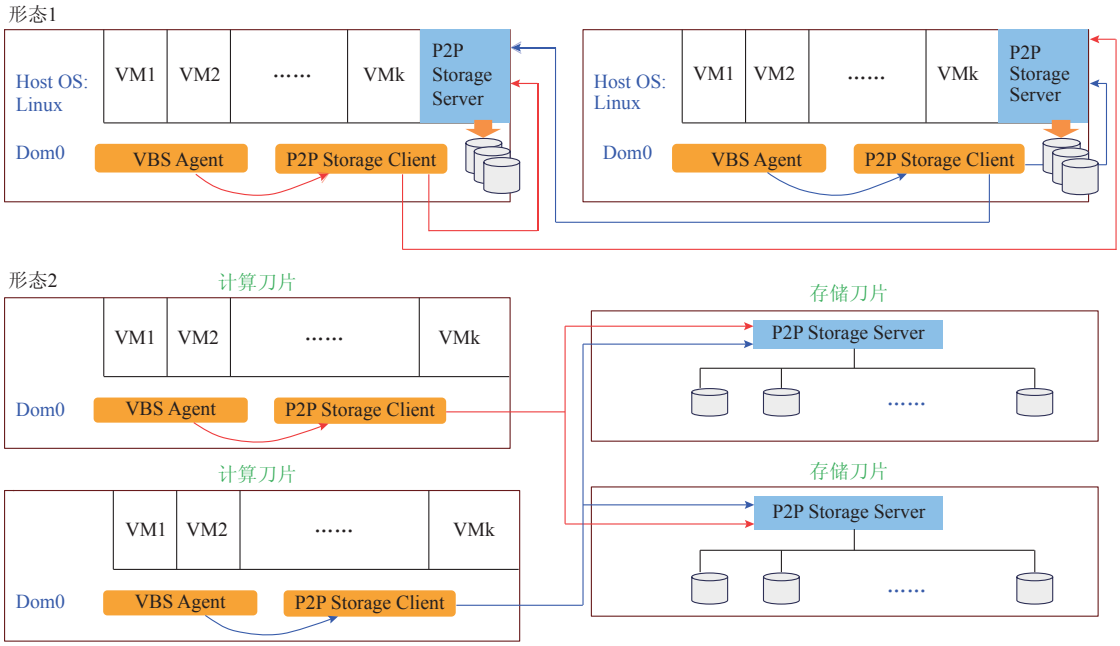


图6-4　计算、存储合一Scale-out架构

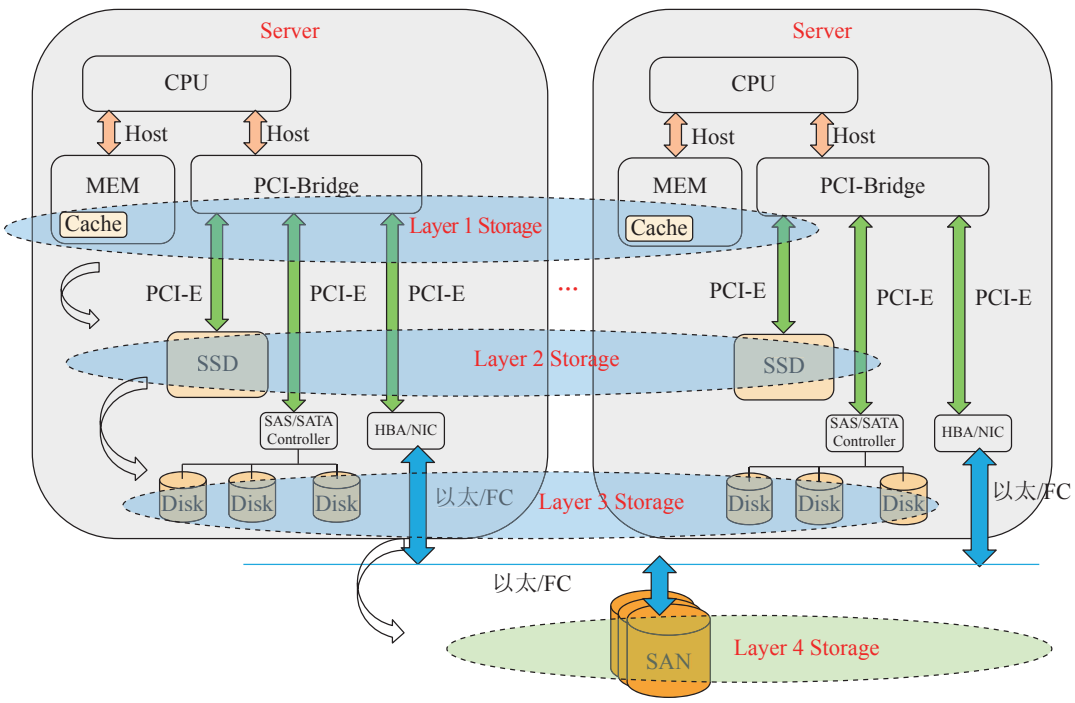


图6-5　融合一体化架构

景)，为提升容量效率，未来需考虑进一步引入硬加速的块级去重引擎。

✍ Layer3 Storage(DAS)相比Layer 4 Storage，时延基本相同，因此作为Cache，意义不大，但可以作为Layer 4 Storage(SAN)的替代或补充(部分数据放置在外置SAN，部分数据放置在Layer 3 Storage)，从而达到降低用户购置及维护外置存储的TCO。

✍ Layer 2 Storage(SSD)作为最终存储介质，因SSD不存在机械损坏故障风险，因此关键在于如何通过优化的Cache算法，将随机写I/O串行化，从而有效降低SSD写放大率，提升SSD寿命，用于中高端数据库等业务。

✍ Layer 3 Storage(DAS)作为最终存储介质，与Layer 4 Storage一样，需面对硬盘机械振动、磨损、环境影响等特殊因素的制约，因此充分借鉴和共享Layer 4 Storage在硬盘故障检测与修复方面的长期经验与成果积累。

✍ 针对新建私有云/公有云的场景推荐采用Layer 3 Storage；针对IT平台替换或改造项目，可考虑借助存储虚拟化充分重用现有的外置Layer 4 Storage，同时将新产生的业务数据部署在Layer 3 Storage上，也可在外置存储退网前将其数据向Layer 3逐步无缝平滑迁移。

✍ 基于内存/SSD网格的计算近端I/O加速：SSD应用加速需要靠近服务器和应用侧，这已经成为业界共识。业界最具代表性的SSD加速产品大多是单机版，不支持分布式缓存一致性，很多应用场景下无法使用，比如：无法支持共享磁盘环境的active/active集群，如双机、数据库集群；无法支持虚拟机集群的动态资源调度和虚拟机迁移功能。而分布式存储引擎利用有最先进的分布式集群技术，可以很好地解决传统架构中热点容量不足的问题。同时，其也可与IPSAN配合，形成高速缓存和外置低速的互补。

✍ 基于Scale-out计算、存储融合架构的I/O性能提升：针对随机IOPS读写，基于分布式存储软件，各服务器内存Cache总容量相比集中式SAN机头的Cache容量增加可达5倍以上，从而使热点数据访问命中率与读写效率提升3～5倍；分布式存储可以采用大容量低成本SATA硬盘提供与SAS/FC硬盘持平的性能，而且有效容量更大；在针对大文件对象的顺序读写方面，分布式存储为App实例或VM提供并发读写服务，使得突发MBPS提升3～5倍以上。

一体化分布式存储架构与Google 那种“Data Center as a Computer”的区别是，后者是面向海量搜索业务的计算/存储垂直整合数据中心，前者是面向企业IT核心业务及电信业务的计算存储垂直整合的高性能、高可扩展的IT平台。

业界典型的分布式存储技术主要有分布式文件系统存储、分布式对象存储和分布式块设备存储等几种形式。分布式存储技术及其软件产品已经日趋成熟，并在IT行业得到了广泛的使用和验证，例如互联网搜索引擎中使用的分布式文件存储，商业化公有云中使用的分布式块存储等。分布式存储软件系统具有以下特点。

✍ 高性能：分布式哈希数据路由，数据分散存放，实现全局负载均衡，不存在集中的数据热点和大容量分布式缓存。

✍ 高可靠：采用集群管理方式，不存在单点故障，灵活配置多数据副本，不同数据副本存放在不同的机架、服务器和硬盘上，单个物理设备故障不影响业务的使用，系统检测到设备故障后可以自动重建数据副本。

✍ 高扩展：没有集中式机头，支持平滑扩容，容量几乎不受限制。

✍ 易管理：存储软件直接部署在服务器上，没有单独的存储专用硬件设备，通过Web UI的方式进行软件管理，配置简单。

6.1.4 分布式软件定义存储多种形态

分布式软件定义存储基于通用的x86服务器，通过分布式的存储软件来构建存储系统，开放丰富灵活的软件定义策略和接口，允许管理员和租户进行自动化的系统管理和资源调度发放。

业界的分布式软件定义存储，有多种形态，包括块存储、文件存储、对象存储。

一、分布式块存储

分布式存储包括以下内容。

✍ 企业应用存储资源池：一般用于运营商、金融、石油、各大中型企业建立自己的私有云资源池；其典型特点是性能线性扩展、容量共享精简分配、多应用性能共享分时复用、成本性价比要求高。

✍ 公有云存储服务：一般用于运营商建立公有云系统，作为公有云中的弹性块存储系统(如Amazon EBS)；其典型特点是、多租户不同SLA等级要求、多应用混合负载、性能和扩展性强、成本性价比要求高。

二、分布式文件存储

分布式文件存储包括以下内容。

✍ 企业应用文件存储：分布式文件存储的行业应用十分广泛，包括媒资、大数据、HPC、企业办公等；一般要求海量大规模，性能线性扩展。

✍ 公有云文件服务：一般用于公有云中的弹性文件服务(如Amazon EFS)，提供给各个租户按需收费的文件存储服务；其典型特点是高扩展能力、跨AZ/DC的共享访问、按需弹性。

三、分布式对象存储

企业中使用分布式对象存储的不是很多，因为对象的访问接口随着公有云的兴起而逐步成为事实标准的，一般在企业中也主要用做海量文件、资料的归档备份；在公有云中，分布式对象存储一般作为一个典型的存储服务(如Amazon S3)，可以用做虚机的镜像存储、租户应用的数据存储等。其典型特点是低成本、跨AZ容灾、无限扩展。

6.2 支持企业关键应用的软件定义块存储

6.2.1 分布式存储池的概念

分布式存储系统把所有服务器的本地硬盘组织成若干个资源池，基于资源池提供创建/删除应用卷(Volume)、创建/删除快照等接口，为上层软件提供卷设备功能。

分布式存储系统资源池具有如下特点(见图6-6)：

✍ 每块硬盘分为若干个数据分片(Partition)，每个Partition只属于一个资源池，Partition是数据多副本的基本单位，也就是说多个数据副本指的是多个Partition。

✍ 系统自动保证多个数据副本尽可能分布在不同的服务器上(服务器数大于数据副本数时)。

✍ 系统自动保证多个数据副本之间的数据强一致性。

✍ Partition中的数据以Key-Value的方式存储。

✍ 对上层应用提供卷设备(Volume)，没有LUN的概念，使用简单。

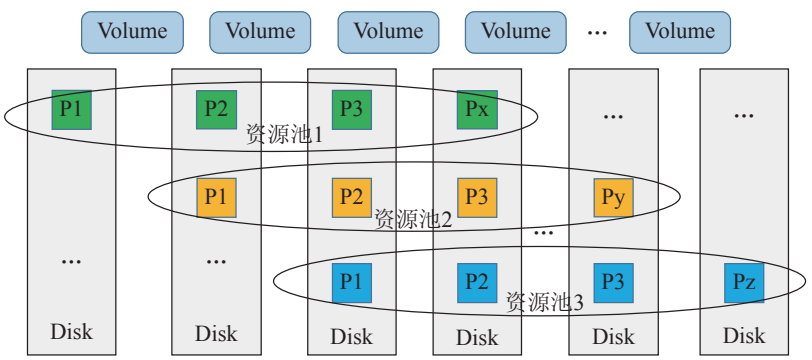


图6-6 分布式存储系统存储资源池

✍ 系统自动保证每个硬盘上的主用Partition和备用Partition数量是相当的，避免出现集中的热点。

✍ 所有硬盘都可以用做资源池的热备盘，单个资源池最大支持数百上千块硬盘。

6.2.2 分布式存储系统的功能框架

分布式存储系统采用分布式集群控制技术和分布式Hash数据路由技术，提供分布式存储功能特性。分布式存储系统功能架构图如图6-7所示。

分布式存储系统功能模块具体如下。

✍ 存储接口层：通过SCSI驱动接口向操作系统、数据库提供卷设备。

✍ 存储服务层：提供各种存储高级特性，如快照、链接克隆、精简配置、分布式Cache、容灾备份等。

✍ 存储引擎层：分布式存储系统存储基本功能，包括管理状态控制、分布式数据路由、强一致性复制技术、集群故障自愈与并行数据重建子系统等。

✍ 存储管理层：实现分布式存储系统软件的安装部署、自动化配置、在线升级、告警、监控和日志等OM功能，同时对用户提供Portal界面。

6.2.3 分布式存储系统的应用场景

分布式存储系统尤其适合计算和存储融合一体化系统。传统的虚拟化方式是在相互分离的计算、存储和网络设备上叠加了一层虚拟化软件。这种方式虽然可以提升资源利用率，但是由于系统的复杂性，并不能简化各类基础设施的运维成本。融合一体化系统真正实现了计算、存储和网络设备的深度融合，硬件设备与虚拟化软件平台的一体化。一体化IT系统采用分布式存储系统把计算服务器的本地硬盘组织成一个类似SAN设备的虚拟存储池，对上层应用提供存储功能。

在IT平台中，分布式存储系统替代了传统的外置存储设备，适合使用分布式存储系统的应用场景。

✍ VDI、OA应用。其典型特点是：容量共享精简分配，性能共享分时复用，计算和存储配比相对均衡，成本性价比要求高。

✍ 虚拟化环境混合应用。其典型特点是：容量共享需求明显，多应用混合负载，线性扩展。

✍ OLAP应用。其典型特点是：大并发吞吐量，计算和存储带宽要求高。

✍ OLTP应用。其典型特点是：IOPS并发度高。

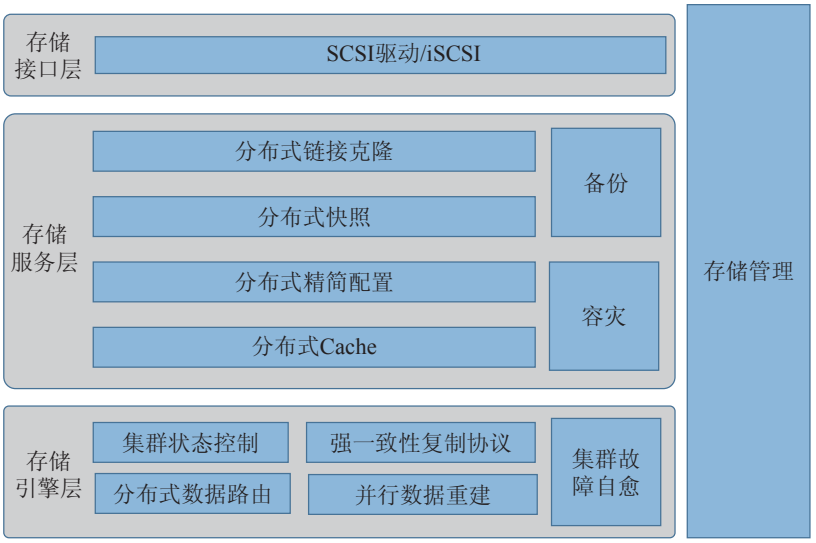


图6-7 分布式存储系统功能架构

6.2.4 分布式存储关键技术：性能提升技术

一、性能卓越

分布式存储系统通过创新的架构把分散的、低速的SATA/SAS机械硬盘组织成一个高效的类SAN存储池设备，提供比SAN设备更高的I/O，把性能发挥到了极致。

分布式存储系统一般支持使用SSD替代HDD作为高速存储设备，支持使用InfiniBand网络替代GE/10GE网络提供更高的带宽，为对性能要求极高的大数据量实时处理场景提供完美的支持。

分布式存储系统采用无状态的分布式软件机头，机头部署在各个服务器上，无集中式机头的性能瓶颈。单个服务器上软件机头只占用较少的CPU资源，却能提供比集中式机头更高的IOPS。其实现了计算和存储的融合，缓存和带宽都均匀分布到各个服务器节点上。

分布式存储系统集群内各服务器节点的硬盘使用独立的I/O带宽，不存在独立存储系统中大量磁盘共享计算设备和存储设备之间有限带宽的问题。其将服务器部分内存用做读缓存，NVDIMM用做写缓存，数据缓存均匀分布到各个节点上，所有服务器的缓存总容量远大于采用外置独立存储的方案。即使采用大容量低成本的SATA硬盘，分布式存储系统仍然可以发挥很高的I/O性能，整体性能提升1～3倍，同时提供更大的有效容量。

二、全局负载均衡

分布式存储系统的实现机制保证了上层应用对数据的I/O操作均匀分布在不同服务器的不同硬盘上，不会出现局部的热点，实现全局负载均衡。

第一，系统自动将数据块打散存储在不同服务器的不同硬盘上，冷热不均的数据会均匀分布在不同的服务器上，不会出现集中的热点。

第二，数据分片分配算法保证了主用副本和备用副本在不同服务器和不同硬盘上的均匀分布，换句话说，每块硬盘上的主用副本和备副本数量是均匀的。

第三，扩容节点或者故障减容节点时，数据恢复重建算法保证了重建后系统中各节点负载的均衡性。

三、分布式SSD存储

分布式存储系统通过支持高性能应用设计的SSD存储系统，可以拥有比传统的机械硬盘(SATA/SAS)更高的读写性能。特别是PCIe卡形式的SSD，会带来更高的带宽和I/O，采用PCIe 2.0 x8的接口，可以提供高达3.0GB的读/写带宽。SSD I/O性能可以达到4KB数据块，100%随机，提供高达600K的持续随机读IOPS和220K的持续随机写IOPS。

SSD存在一个普遍的问题，就是写寿命问题，在采用SSD的时候，分布式SSD存储系统通过以下措施增强了可靠性(见图6-8)。

- 内嵌的ECC检错/纠错引擎和RAID5引擎，数据通道间形成二维的检错/纠错机制；
- 内置DATA Scrubbing引擎定时检测存储数据，提前预防数据错误的产生；
- 通道间使用Dynamic RAID算法，实现通道间的资源共享，确保在芯片坏块过多甚至是多个芯片故障的情况下均能正常工作；
- 内部实现冷热数据分类与管理，配合先进的磨损算法，最大程度地提升回收效率，降低写磨损，从而提升SSD的使用寿命。

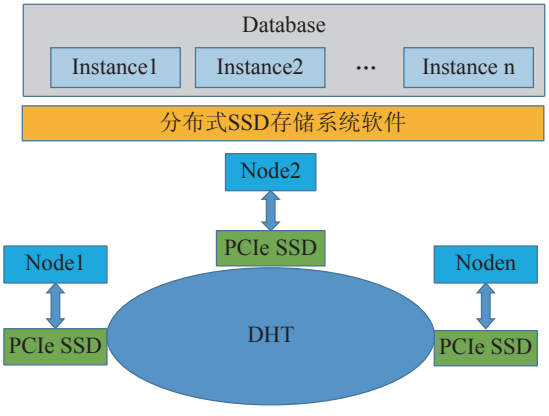


图6-8 分布式存储系统支持分布式SSD存储系统

四、高性能快照

分布式存储系统提供了快照机制，将用户的逻辑卷数据在某个时间点的状态保存下来，后续可以作为导出数据、恢复数据之用。

分布式存储系统快照数据基于DHT机制，快照不会引起原卷性能下降。针对一块容量为2TB的硬盘，完全在内存中构建索引需要几十MB空间，通过一次Hash查找即可判断有没有做过快照，以及最新快照的存储位置，因此效率很高(见图6-9)。

五、高性能链接克隆

分布式存储系统可以基于增量快照提供链接克隆机制，基于一个快照创建出多个克隆卷，各个克隆卷刚创建出来时的数据内容与快照中的数据内容一致，后续对于克隆卷的修改不会影响原始的快照和其他克隆卷。分布式存储系统通过支持批量进行虚拟机卷部署，可以在秒级批量创建上百个虚拟机卷。克隆卷可支持创建快照、从快照恢复以及再次作为母卷进行克隆操作(见图6-10)。

六、高速InfiniBand网络

为消除分布式存储环境中存储交换瓶颈，分布式存储系统可以部署为高带宽应用设计的InfiniBand网络。InfiniBand网络可以为分布式存储系统带来以下特性：

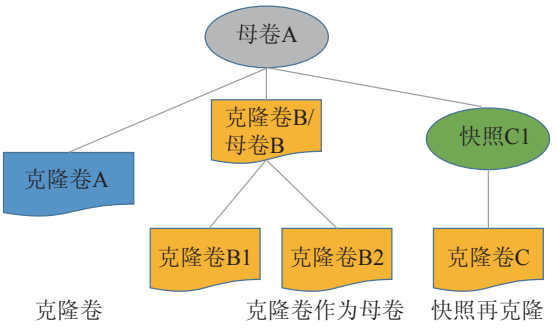


图6-10　分布式存储系统链接克隆

- 56Gbps FDR InfiniBand，超高速互联；
- 标准成熟多级胖树组网，平滑容量扩容；
- 近似无阻塞通信网络，数据交换无瓶颈；
- 纳秒级通信时延，计算存储信息及时传递；
- 无损网络QoS，数据传送无丢失；
- 主备端口多平面通信，冗余通信无忧；
- 单口56Gbps带宽，完美配合极速SSD存储吞吐，性能无限。

6.2.5　分布式存储关键技术：简化管理技术

分布式存储系统采用的分布式集群架构，天然支持无性能损耗的弹性扩展。

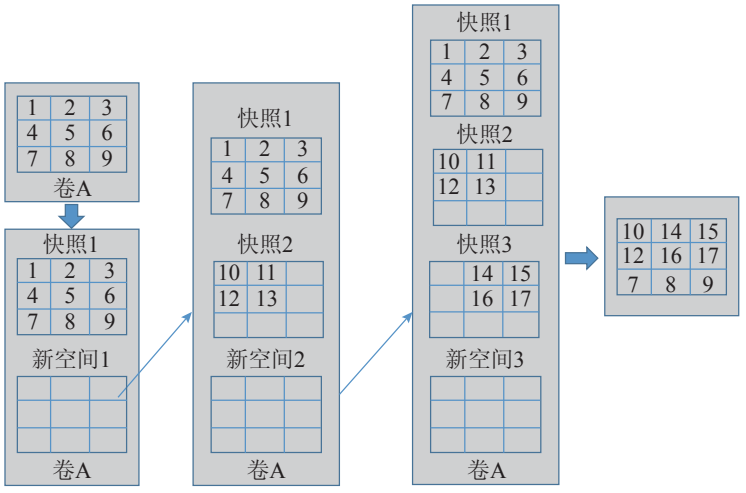


图6-9　分布式存储系统快照

DHT数据路由：分布式存储系统采用DHT(Distribute Hash Table，分布式哈希表)路由数据算法。每个存储节点负责存储一小部分数据，基于DHT实现整个系统的寻址和存储。

DHT算法具有以下特点。

- 均衡性(Balance)：数据能尽可能地分布到所有的节点中，这样可以使得所有节点负载均衡。
- 单调性(Monotonicity)：当有新节点加入到系统中时，系统重新做数据分配，原来的数据存储位置不需要很大的调整。

由于分布式存储系统存储路由采用分布式哈希算法，使得存储系统具有如下特点(见图6-11)。

- 快速达到负载均衡：新加入节点只需要搬移很少部分数据分片即可达到负载均衡。
- 数据高可靠：灵活配置的分区分配算法，避免多个数据副本位于同一个服务器、同一个磁盘上。

一、平滑扩容节点

分布式存储系统的分布式架构具有良好的可扩展性，支持超大容量的存储(见图6-12)。

- DHT算法保证了扩容后不需要做大量的数据搬迁，可以快速达到负载均衡状态。
- 扩展计算节点可以同步扩容存储空间，新扩展节点和原有节点可构成统一的资源池进行使用。
- 分布式存储系统分布式系统的带宽和Cache均匀分布在各个节点上，带宽和Cache不会随着节点的扩容而减少。

二、资源按需使用

分布式存储系统提供了精简配置机制，为用户提供比实际物理存储更多的虚拟存储资源。相比直接分配物理存储资源，可以显著提高存储空

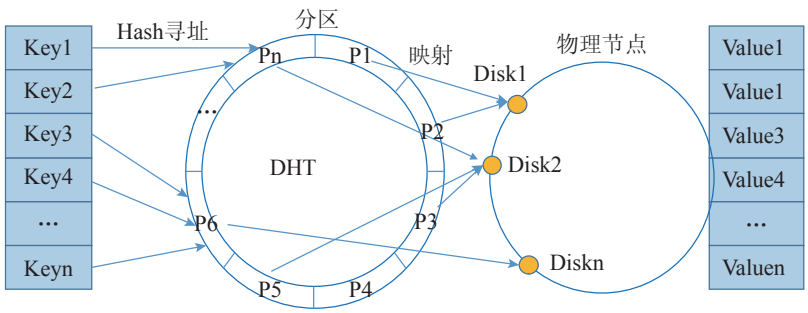


图6-11　分布式存储系统DHT数据路由

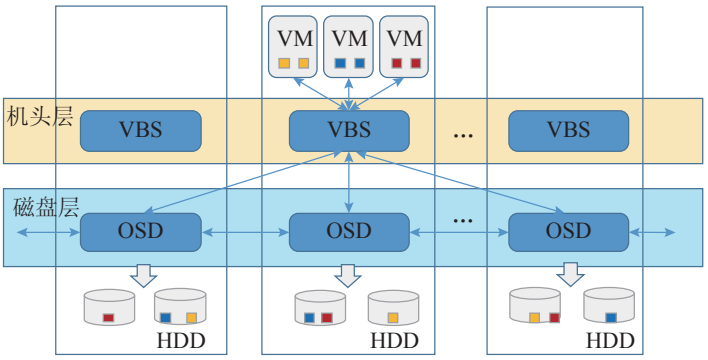


图6-12　分布式存储系统平滑扩容节点

间利用率。

采用分布式Hash技术，天然支持分布式自动精简配置(Thin Provisioning)，无须预先分配空间。

精简配置(Thin Provisioning)无任何性能下降(IPSAN扩展空间时需要耗费额外的性能)，如图6-13所示。

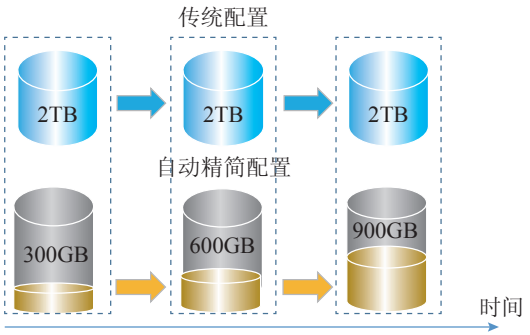


图6-13　分布式存储系统自动精简配置

三、统一的Web管理界面

分布式存储环境，在大规模场景下涉及设备众多，各组件的运行维护需要通过自动化形式完成，尽量减少人工干预。用户从Portal界面可以查看系统监控(KPI指标)、告警事件和存储池状态等，操作维护简单，有力帮助分布式系统在传统企业落地部署(见图6-14)。

四、广泛兼容能力

其一般要求分布式存储系统采用通用x86服务器平台，在软件上支持通用的操作系统、数据库系统及虚拟化软件。

6.2.6　分布式存储关键技术：安全可靠性增强技术

一、集群管理

分布式存储系统的分布式存储软件采用集群管理方式，规避单点故障，一个节点或者一块硬盘故障自动从集群内隔离出来，不影响整个系统业务的使用。

集群内选举进程Leader，Leader负责数据存储逻辑的处理，当Leader出现故障，系统自动选举其他进程成为新的Leader。

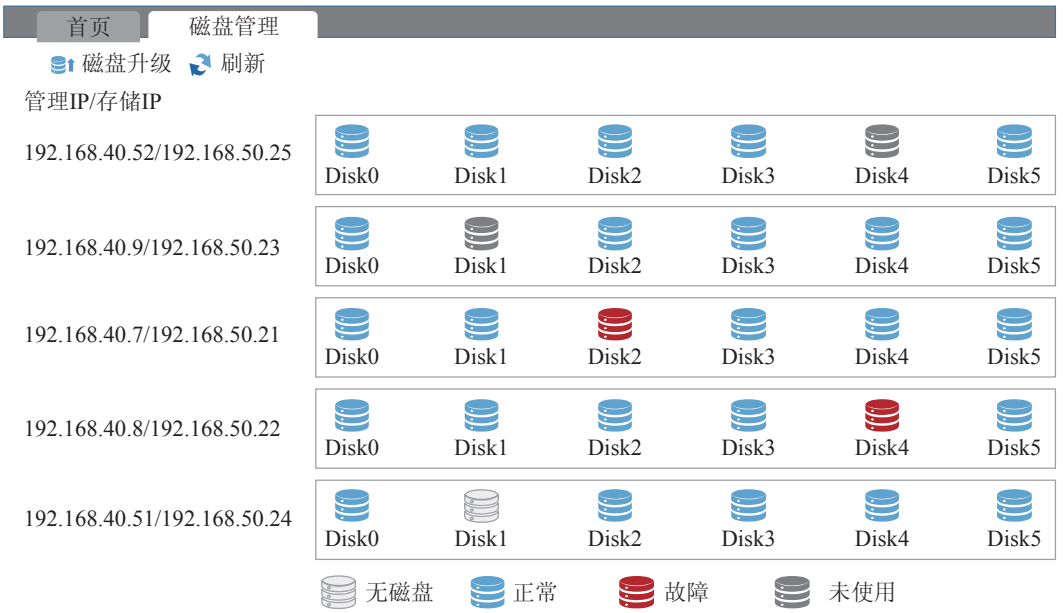


图6-14　分布式存储系统Web UI界面样例

二、多数据副本

分布式存储系统存储系统中没有使用传统的RAID模式来保证数据的可靠性，而是采用了多副本备份机制，即同一份数据可以复制保存多个副本。在数据存储前，对数据进行分片，分片后的数据按照一定的规则保存集群节点上。

如图6-15所示，对于服务器Server1的磁盘Disk1上的数据块P1，它的数据备份为服务器Server2的磁盘Disk2上P1'，P1和P1'构成了同一个数据块的两个副本。

三、数据一致性

数据一致性的要求是：当应用程序成功写入一份数据时，后端的几个数据副本必然是一致的，当应用程序再次读取时，无论在哪个副本上读取，都是之前写入的数据，这种方式也是绝大部分应用程序所希望的。

保证多个数据副本之间的数据一致性是分布式存储系统的重要技术点，分布式存储系统采用强一致性复制技术确保各个数据副本的一致性，一个副本写入，多个副本读取。

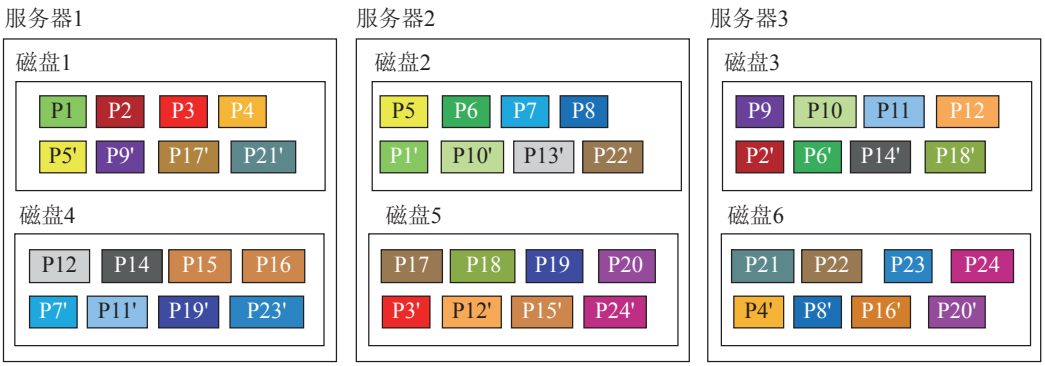


图6-15　分布式存储系统多数据副本

分布式存储系统还支持Read Repair机制。Read Repair机制是指在读数据失败时，会判断错误类型，如果是磁盘扇区读取错误，可以通过从其他副本读取数据，然后重新写入该副本的方法进行恢复，从而保证数据副本总数不减少。

四、快速数据重建

分布式存储系统内部需要具备强大的数据保护机制。数据存储时被分片打散到多个节点上，这些分片数据支持分布在不同的存储节点、不同的机柜之间，同时数据存储时采用多副本技术，数据会自动保存多份，每一个分片的不同副本也被分散保存到不同的存储节点上。在硬件发生故障导致数据不一致时，分布式存储系统通过内部的自检机制，通过比较不同节点上的副本分片，自动发现数据故障。发现故障后启动数据修复机制，在后台修复数据。由于数据被分散到多个不同的存储节点上保存，数据修复时，在不同的节点上同时启动修复，每个节点上只需修复一小部分数据，多个节点并行工作，有效避免单个节点修复大量数据所产生的性能瓶颈，对上层业务的影响做到最小化。数据故障自动恢复流程如图6-16所示。

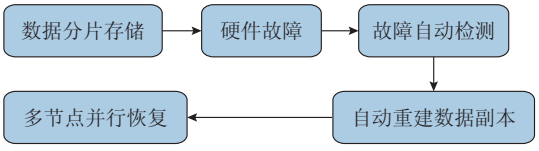


图6-16　分布式存储系统数据重建流程

分布式存储系统需要支持并行、快速故障处理和重建：

- 数据分片在资源池内被打散，硬盘出现故障后，可在资源池范围内自动并行重建；
- 数据分布上支持跨服务器或跨机柜，不会因某个服务器故障而导致数据不可访问；

- 扩容时可以自动进行负载均衡，应用无需调整即可获得更大的容量和性能。

五、掉电保护

分布式存储系统运行过程中可能会出现服务器突然下电的情况，分布式存储系统在内存中的元数据和写缓存数据会随着掉电而丢失，需要使用NVDIMM非易失内存来保存和恢复元数据和缓存数据。

部署分布式存储系统软件的每一台服务器要求配备NVDIMM内存条，服务器掉电时会把元数据和缓存数据写入NVDIMM的Flash中，上电后又会把Flash中的数据还原到内存中。

分布式存储系统能够识别出系统中的NVDIMM内存，并把需要保护的数据按照内部规则存放在NVDIMM中，以便提供掉电保护功能。

6.3 传统存储SAN/NAS的管理整合及性能加速

存储虚拟化概述

当仅需要单个主机服务器(或单个集群)访问多个磁盘阵列时，可以使用基于主机的存储虚拟化技术。该技术又称为逻辑卷管理，通常由主机操作系统下的逻辑卷管理软件实现。逻辑卷管理软件把多个不同的物理磁盘映射成一个虚拟的逻辑块空间。当存储需求增加时，逻辑管理软件能把部分逻辑空间映射到新增的磁盘阵列，因此可以在不中断运行的情况下增加或减少物理存储设备。

基于主机的存储虚拟化示意图如图6-17所示。

主机1可以使用磁盘阵列1和阵列2上的存储空间，主机2可以使用磁盘阵列2上的存储空间，主机3和主机4均可使用磁盘阵列3和阵列4上的存储空间。

该技术使主机经过虚拟化的存储空间跨越多个异构的磁盘阵列，因此常用于在不同磁盘阵列之间做数据镜像保护。

该技术的优点：

- 支持异构的存储系统；
- 容易实现，不需要额外的特殊硬件；
- 开销低，不需要硬件支持，不修改现有系统架构。

该技术的缺点：

- 占用主机资源，降低应用性能；
- 存在操作系统和应用的兼容性问题；
- 导致主机升级、维护、扩展复杂，容易造成系统不稳定；
- 需要复杂的数据迁移过程，影响业务连

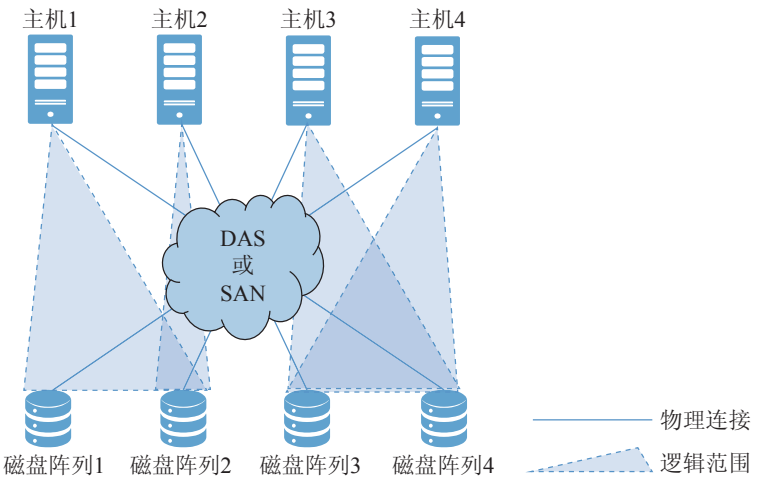


图6-17 基于主机的存储虚拟化

续性。

如果仅针对传统中低端存储设备整合的软件定义存储方案，其架构要简单很多，架构框架如图6-18所示。

中低端存储的存储性能差，LUN数量有限，无NVRAM、SSD加速，无快照/克隆，瘦分配等功能，可通过主机侧的Flash硬件和软件定义存储软件能力，整合现有中低端SAN、NAS等存储资源，提供性能高、功能强的软件定义存储解决方案。

面向中低端存储整合的软件定义存储系统主要是一套存储虚拟化软件，运行在Host OS上，它具有强大的异构能力，底层能够兼容块、文件或者对象。软件定义存储系统具备线性扩展能力，具有高速的分布式Flash Cache，能够实现性能无损的快照和瘦分配，能够实现VM粒度的策略驱动，拥有丰富的对外接口，能够对外提供块、文件或对象接口。系统主要提供卷管理服务、I/O服务、元数据服务。各种服务可以融合部署，也可分离部署。软件定义存储的软件逻辑如图6-19所示。

6.4 分布式对象存储

6.4.1 分布式对象存储的概念

分布式对象存储，通常指基于通用的x86/ARM服务器的本地硬盘，通过分布式的存储软件构建起来，一般采用HTTP/REST接口进行访问操作的存储系统。如Amazon的S3，OpenStack Swift等。

对象存储有两个基本概念，桶(Bucket)和对象(Object)。桶是对象的容器，每个用户可以创建很多个桶，桶名不能重复。桶可以用于计费、权限控制等功能。对象是要存储的数据内容，每个对象可以指定一个唯一标识，应用可以很方便地根据标识进行对象的读、写、删，对象的数据内容长度可以从几个字节到几个TB，对象存储在桶中。

6.4.2 分布式对象存储系统的技术特点

相比文件存储而言，对象比较简单方便，结构层次扁平，只有桶和对象两层；操作也简单，只有put、get、delete、list等基本操作。这种扁平的层次和简单的操作，更容易构建高扩展的集群存储系统。一般的分布式对象存储系统都具备以

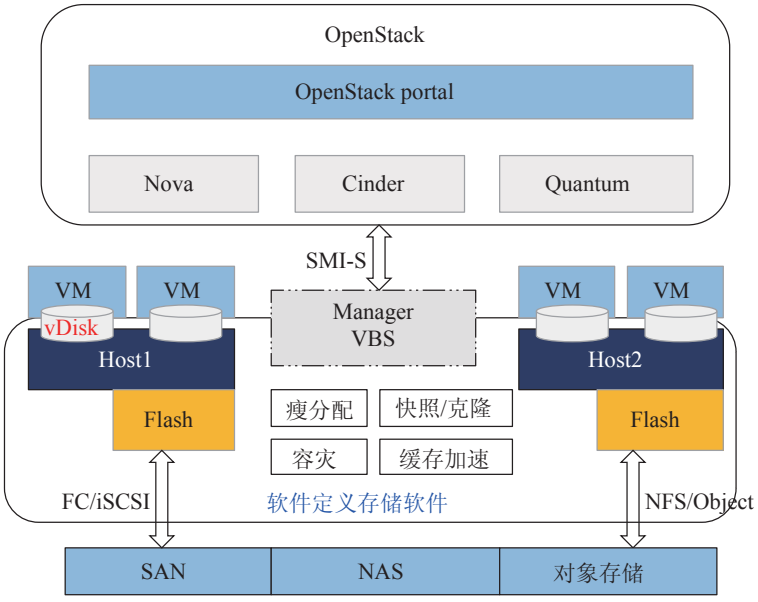


图6-18　中低端存储整合软件定义存储系统框架

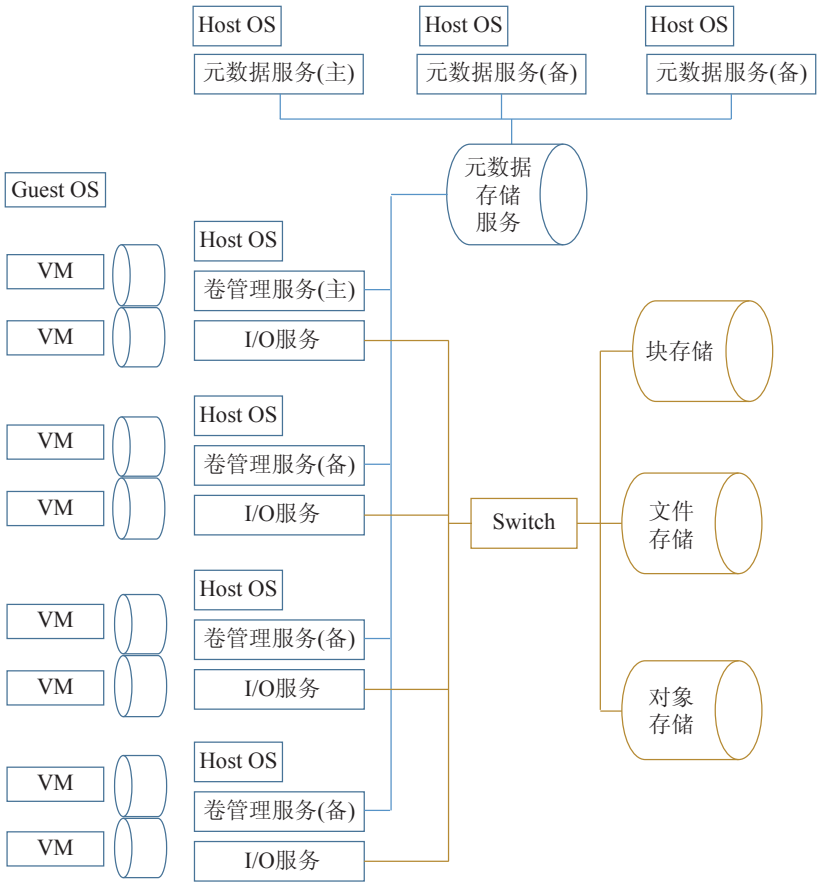


图6-19　面向中低端存储设备软件定义存储的软件逻辑

下技术特点。

一、大规模线性扩展

OpenStack Swift，Amazon S3底层存储引擎，都基于DHT(Distributed Hash Table)算法来构建，扩展能力强，可以从几台服务器扩展到成千上万的服务器规模。DHT的示意图如图6-20所示。

在具体应用DHT算法过程中，不同的系统会有不同的扩展实现机制，有的是按照物理节点进行DHT，有的是按照虚拟逻辑节点(也可称为Partition)进行DHT。按照物理节点进行DHT的，在节点扩容的时候，会对邻近节点有较大的数据重均衡影响；用虚拟节点进行DHT的，可以预先设置固定的虚拟节点数量，在物理节点扩容的

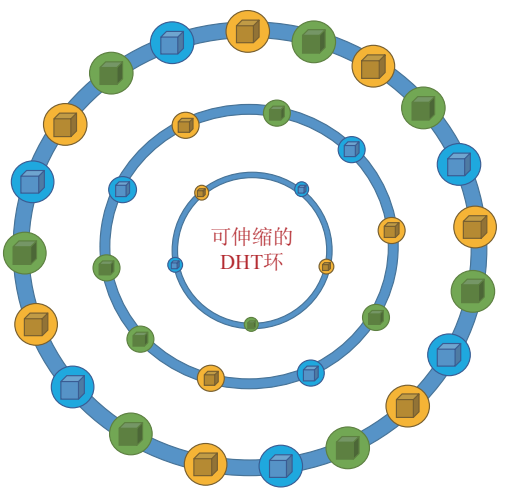


图6-20　可伸缩DHT环

时候，对各个物理节点的数据重均衡影响比较均匀。但是不管哪种具体的实现方法，采用DHT技术，根据对象的标识计算出该对象所在的存储节点，直接进行对象的存取，可以避免空间管理上元数据的查找，也不会随着系统存储规模的越来越大，而导致空间管理的元数据越来越多，可以做到线性扩展。

二、无状态的接入控制节点，处理海量用户请求

对象存储系统是基于互联网HTTP/REST的对象接口协议，海量的存储节点，需要能够处理大量HTTP/REST读写请求的前端接入控制节点，一般在这类大规模系统构建的时候，前端都会采用无状态服务的方式，服务请求可以通过负荷分担机制由任何一个接入控制节点提供服务，任何一个接入控制节点都可以直接访问到后端任何存储节点的对象数据，如图6-21所示。

三、跨多站点的数据同步和访问

分布式对象存储系统，需要具备跨站点的数据同步能力，一方面是为了随时随地的数据访问，用户在A站点上传的数据可以在B站点下载；另一方面是为了数据的持久度，即便一个站点故障后，数据仍然是安全的。其不管在公有云还是私有云中都是比较普遍的需求(见图6-22)。

四、低成本

分布式对象存储，主要用做海量内容的存储，对成本比较敏感。对象存储系统一般都会采用Erasure Code(N+M)的方式进行数据冗余保护，有效提升空间利用率。Erasure Code(N+M)是指把一个文件切分成N个数据块，计算出M个校验块，总共N+M个数据块，空间有效利用率为N/(N+M)，比如图6-23所示的4+2，4个数据块，2个校验块，空间利用率为66.67%。

当有数据块故障时，如n4和m2，系统会自动在其他节点或硬盘上把n4、m2的数据块重建出来，n1、n2、n3、n4、m1、m2又称为一个EC条带。对于持久度要求比较高的系统，可以增多校验块数量，但是会让重建时的开销增大。

随着工业界硬盘技术的发展，更大容量的硬盘和新型的硬盘(如SMR盘)也会陆续出现，也可以在一定程度上降低系统构建的成本。

五、安全加密

对象存储基于HTTP/REST协议进行访问，在数据传输过程中需要支持采用SSL(Secure Sockets Layer)加密，另外也需要支持ACL(Access Control List)、Bucket Policy、IAM(Identity and Access Management) Policy等多种方式管理权限，提升数据的安全性(见图6-24)。

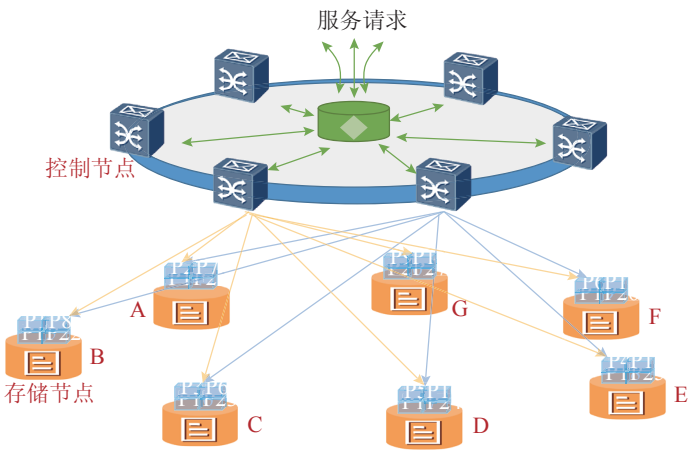


图6-21　去中心化架构实现全对等点对点访问

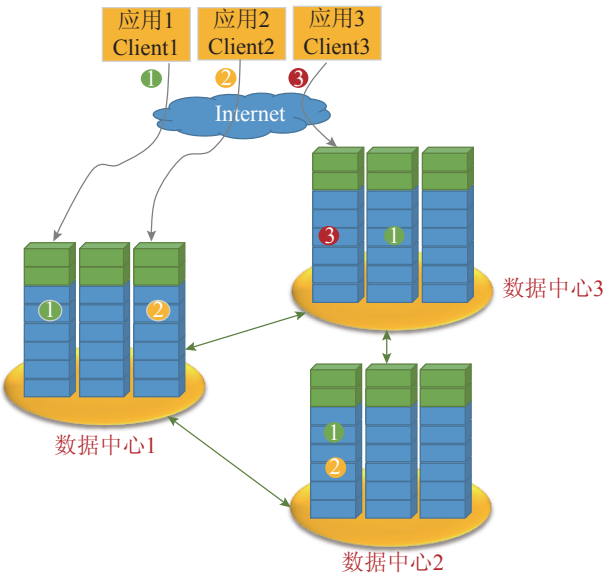


图6-22　多数据中心跨地域部署并支持统一调度与灵活管理

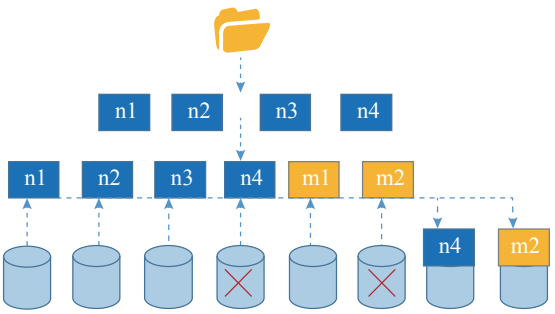


图6-23　N+M的Erasure Code冗余

图 6-24　数据安全加密过程

6.4.3　分布式对象存储的应用场景

分布式对象存储有着广泛的应用场景，如备份归档，内容存储，大数据存储和分析等，也可以作为大型企业内多种应用的对象存储池。

一、海量对象资源池

分布式对象存储，可以提供海量对象资源池解决方案，面向种类繁多的上层业务，为客户提供存储多种类型业务数据的需求，帮助客户解决海量数据存储、匹配高可靠的问题，同时帮助客户避免维护多套不同的存储设备和网络，降低TCO，摆脱复杂的管理工作，聚焦在业务本身，创造更大的价值(见图6-25)。

二、备份归档

备份，一直是数据保护的一种重要手段，被广泛应用于各种应用场景和多个垂直行业，备份介质形态也各有不同，从传统的磁带、虚拟带库到光盘、磁盘阵列，人们在享受备份对数据高可靠性保障的同时，也深受现有备份产品的困扰(见图6-26)。

✍ 恢复数据不方便：恢复数据只能够更新

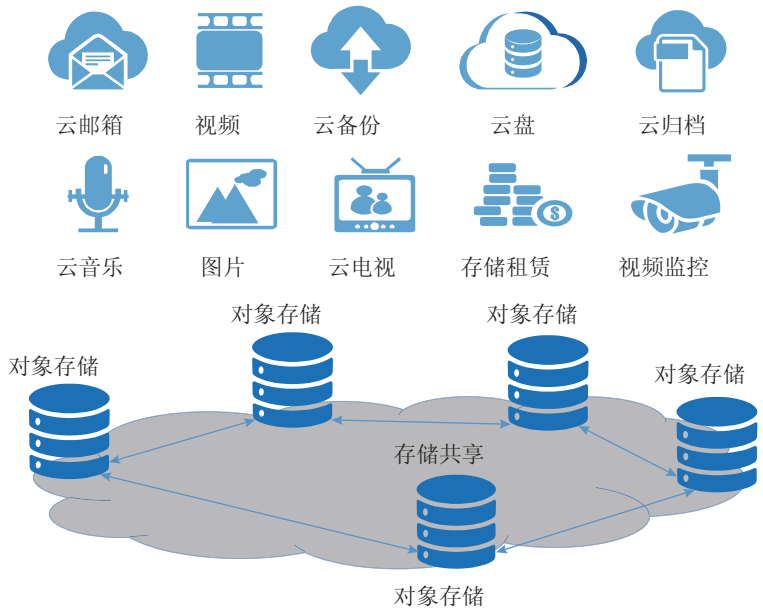


图6-25　海量对象资源池

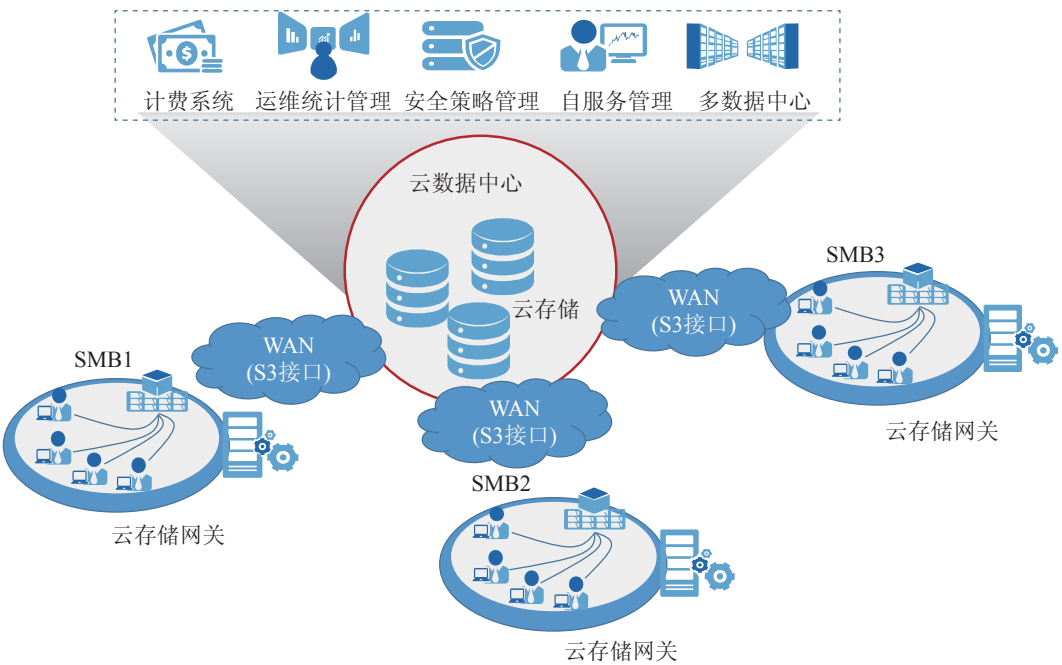


图6-26　云存储备份归档

到上次磁带备份的程度。

✍ 容量低：磁带库容量有限，虚拟磁带库该问题更明显。

✍ 扩展难：磁带库和虚拟磁带库扩展有上限，除非新购买设备。

采用分布式对象存储作为备份存储，可以很好地处理上述问题。

归档面向数据中心应用，能够把主存储中不常用的数据迁移到安全、廉价的二级存储平台，从而充分发挥存储效率，降低TCO。

三、个人/企业网盘

网盘解决方案基于IP 网络利用对象存储技术为终端用户提供在线存储服务。该方案满足海量低成本、弹性扩展、大范围整合能力等需求，通过丰富的在线存储业务打造以用户为中心安全、方便、高速、大容量的个人数据中心，为企业用户提供安全可靠、经济易用、快速部署的生产、办公协同的网盘服务(见图6-27)。

6.5 面向云存储服务的QoS/SLA管理

在公有云中，弹性块存储服务和对象存储服务是被广泛使用的，租户和应用在使用的过程中无须关心底层的存储系统是基于什么技术和硬件构建，只需要按需指定所需存储服务的QoS/SLA能力。

对弹性块存储服务而言，QoS一般指的是IOPS、MBPS、时延。以Amazon的EBS(摘自Amazon官方网站)为例，根据不同的IOPS、MBPS、时延把块存储分为几个档次，如cold HDD、吞吐量优化HDD、通用SSD、预配置SSD；这些不同的块存储服务的IOPS、MBPS、时延都会有所差别，租户可以根据自己应用的需要，选择不同档位的块存储服务(见表6-1)。

QoS的主要控制机制包括以下几种。

✍ IOPS上限控制、IOPS下限控制、Burst IOPS控制、每GB IOPS：公有云存储服务上每个等级的IOPS控制基准一般是以每GB来衡量，卷越大，IOPS就会越大，但是也存在一个IOPS上限。Burst IOPS主要用于在突发(比如业务短暂突增)情况下需要持续一段时间的高IOPS； 带IOPS下限控制的存储类型，是能够基本保证(比如99.9%的时间) 达到标定的IOPS值，典型的如Amazon的预配置SSD。

✍ MBPS上限控制、MBPS下限控制、Burst MBPS控制、每TB MBPS：对于带宽类的卷类型，一般基准是按照每TB MBPS来控制，卷越

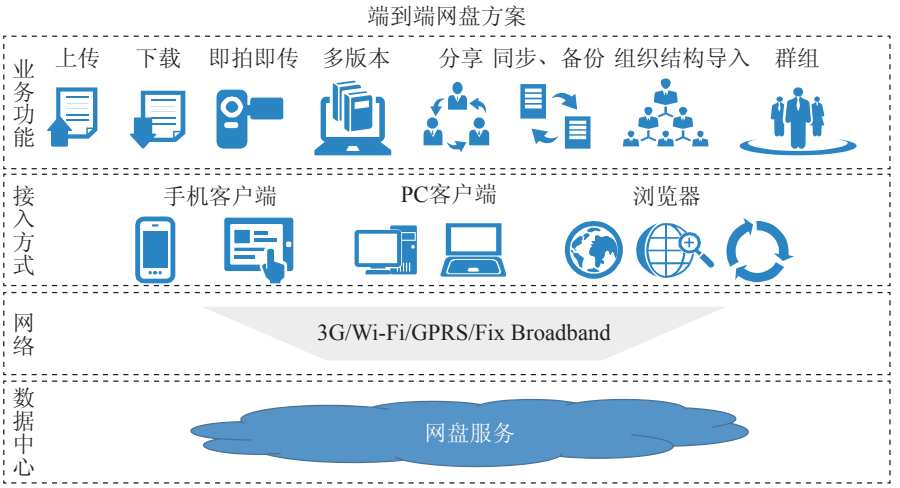


图6-27 个人/企业网盘

表6-1　块存储服务的划分

	固态硬盘(SSD)		普通硬盘(HDD)	
卷类型	EBS预配置IOPS SSD(io1)	EBS通用型SSD(gp2)	吞吐量优化型HDD(st1)	Cold HDD(sc1)
简短描述	专用于对延迟敏感的交易型工作负载的最高性能SSD卷	用于均衡各种交易型工作负载的价格和性能的通用型SSD卷	专用于频繁访问的吞吐量密集型工作负载的低成本HDD卷	专用于非频繁访问的工作负载的最低成本HDD卷
使用案例	I/O密集型NoSQL数据库和关系型数据库	启动卷、低延迟交互式应用程序、开发和测试	大数据、数据仓库、日志处理	每天需要较少扫描的冷数据
API名称	io1	gp2	st1	sc1
卷大小	4GB～16TB	1GB～16TB	500GB～16TB	500GB～16TB
自大IOPS**/卷	20000	10000	500	250
最大吞吐量/卷	320MB/s	160MB/s	500MB/s	250MB/s
最大IOPS/实例	48000	48000	48000	48000
最大吞吐量/实例	800MB/s	800MB/s	800MB/s	800MB/s
价格	每月每GB0.125USD 每个预配置IOPS0.065USD	每月每GB0.10USD	每月每GB0.045USD	每月每GB0.025USD
主要性能属性	IOPS	IOPS	MB/s	MB/s

大，MBPS就越高，但是也存在一个MBPS上限。同理，Burst MBPS主要用于在突发情况下持续一段高MBPS。

✍ 对于时延，不同类型的存储会有一个大概的等级，比如HDD的存储大概10ms左右，SSD大概1~3ms。

对于公有云中的对象存储服务而言，因为经过了互联网HTTP/REST，时延上受到网络的影响很大，服务提供上一般不会严格承诺QoS指标，或者承诺一些相对宽泛的QoS指标。

对于存储服务而言，有两个指标是服务提供商无法回避的SLA，一个是数据持久度，一个是数据可用性。

数据持久度，指的是数据丢失的概率；数据可用性，指的是数据可访问的概率。以Amazon S3对象存储服务举例：S3的数据会跨3个AZ存储，数据的持久度是年度为对象提供99.999999999%(11个9)的持久性，意味着概率上讲1000亿的对象才可能会存在一个对象数据丢失，很重要的一点是该对象存储服务能够承受两个AZ的数据丢失；数据的可用性为99.99%，意味着每年小于50分钟的数据不可访问时间。

6.6 分布式软件定义存储的Erasure Code，分布式重删压缩

6.6.1 分布式低时延Erasure Code

首先，我们介绍一个用来衡量分布式存储系统整体性能和时延的要素，其可以分解为如下几个方面。

✍ Roud delay：网络上消息往返次数，即应用层的一次读或写请求，需要多少次底层分布

式存储多少次的网络消息往返才能完成。每次跨网络消息交互都会增加响应时延，降低性能，所以对于分布式存储系统而言，越少的网络消息往返，就意味着越好的时延表现。

✍ Messages：网络上消息的总数，即应用层的一次读或写请求，总共需要分布式存储层多少条消息交互才可以完成。其计算的是消息总条数，和roud delay的消息往返次数有区别。每种网络都有一定的pps，如果上层的一个读写请求消耗的网络pps越多，也就意味着整个系统最终可以达到的IOPS越少。

✍ Disk reads：读磁盘次数，即应用层的一次读或写请求，需要读多少次磁盘才能完成。应用层的一次读写请求，消耗的磁盘读越多，也就意味着整个系统的IOPS越低。

✍ Disk writes：写硬盘次数，即应用层的一次读或写请求，需要写多少次磁盘才能完成。应用层的一次读写请求，消耗的磁盘写越多，也就意味着整个系统的IOPS越低。

✍ NVM reads：NVM cache介质读的次数，即应用层的一次读或写请求，需要读多少次NVM Cache介质才能完成。此处的NVM cache介质并不特指SSD cache、NVDIMM cache或DRAM。应用层的一次读写请求，消耗的NVM cache介质读越多，也就意味着整个系统的IOPS越低。

✍ NVM writes：NVM cache介质写的次数，即应用层的一次读或写请求，需要写多少次NVM Cache介质才能完成。应用层的一次读写请求，消耗的NVM cache介质写越多，也就意味着整个系统的IOPS越低。

✍ Network b/w：网络带宽消耗，即应用层的一次读或写请求(如4K，8K等)，需要消耗掉多少网络带宽。消耗带宽越多，意味着整个系统的MBPS越低。

上述七点，基本上可以决定一个分布式存储系统的性能和时延到底可以达到什么程度，下面以Amazon Dynamo作为例子，分别给出上面七点涉及的内容，如表6-2所示。

目前，业界主流的分布式存储一般采用副本策略，特别是分布式的块存储，但是这种副本冗余的方式，造成资源利用率低，例如采用3副本的情况下，存储池的有效利用率只有33%，成本高。传统的SAN可以采用RAID10和RAID5/6来降低成本，分布式存储系统中类似的情况一般采用Erasure Code的方式，Erasure Code可以支持灵活的N+M条带(N个数据块，M个校验块)。但是在分布式块存储系统中实现低时延的Erasure Code，存在两个技术难点。

表6-2 Amazon Dynamo

1. 3副本 2. nKB读或写	Amazon Dynamo (NRW-DHT)	
	Read	Write
Round delay	1	2
Messages	2	5
Disk reads	2	2
Disk writes	0	3
NVM reads	2	2
NVM writes	0	0
Network b/w	2nKB	5nKB

(1) 和RAID5/6一样，Erausre Code的N+M个条带存在跨节点的条带一致性问题。在修改一个条带的某个数据分片的时候，必须保证校验块也一起被修改，保证整个条带仍然是一致的。

(2) Erasure Code条带在承担多客户端并发访问的时候，如何确保多个客户端的数据是一致的。多个客户端访问并发某个EC条带的时候，需要通过一种机制对多个客户端进行互斥并发访问。

上述两点，都会造成分布式块存储系统因为实现Erasure Code而带来时延增加和性能下降。

为了解决低时延小块随机IO的条带惩罚，分布式低时延Erasure Code的架构设计是这样的：一般在EC的前端加一层采用副本机制的分布式的Cache，用来做write back，使得IO快速返回给上层应用，降低读写时延，特别是写的时延。

具体到后端EC的架构设计上，大体上可以分为两大类，我们暂且使用2pc erasure code和full-stripe erasure code这两个名称来代表这样两类分布式低时延Erasure Code的设计机制。

一、2pc erasure code

采用由条带主节点控制的二阶段事务，来解决单客户端访问时条带一致性问题。多客户端的并发读写操作，通过一定的数据路由策略，都会路由到主节点进行串行化，解决多客户端并发互斥的问题。(注：此处仅从条带数据一致性角度描述如何解决多客户单并发冲突，不包含上层应用语义的互斥访问问题。)图6-28中，D1、D2、D3、D4代表的是EC条带的数据块，P1、P2代表的是EC条带的校验块，D1、D2、D3、D4、P1、P2构成了一个完整的EC条带，C1、C2代表同时访问该条带的两个客户端。

对应的衡量表格如表6-3所示，按照上图：m=4，n=6，k=2，1<=x<=4。

表6-3　衡量表格

	满条带读写		X Block访问	
	Read	Write	read	Write
Round delay	4	4	4	4
Messages	2m + 2	2n + 2	2x + 2	2x + 4k + 2
Disk reads	m	0	x	k
Disk writes	0	n	0	x + k
NVM reads	0	0	0	0
NVM writes	0	n	0	n
Network b/w	(2m−1)B	(2n−1)B	2xB	(2x+4k)B

当然，为了保证多客户端并发访问时条带数据的一致性，我们还可以采用分布式锁的方式。如图6-29所示，C1、C2在同时访问某个条带的时候需要先加锁，先抢到锁的客户端具备对该条带的读写权限，条带的一致性仍然需要通过2PC来保证。

对应的衡量表格如表6-4所示，按照上图：m=4，n=6，k=2，1<=x<=4。

表6-4　衡量表格

	满条带访问		X Block访问	
	Read	Write	read	Write
Round delay	2	4	2	4
Messages	2n	4n	2n	4n
Disk reads	m	0	x	x + k
Disk writes	0	n	0	x + k
NVM reads	n	0	n	n
NVM writes	0	n	0	n
Network b/w	mB	nB	xB	2(x+k)B

上述两种方法的适用场景也会略有不同，第一种方法比较适合于随机小I/O且条带冲突比较大的场景；第二种方法适合于大块吞吐冲突且条带冲突比较小的场景。

二、full-stripe erasure code

full-stripe erasure code与2PC的最大的区别在于条带的一致性，无须通过二阶段事务来保证，对于条带而言永远都是满条带的写，这个就需要上层的cache中具备满条带的数据。当上层发生小I/O的时候，首先小块I/O缓存到Cache层，然后给

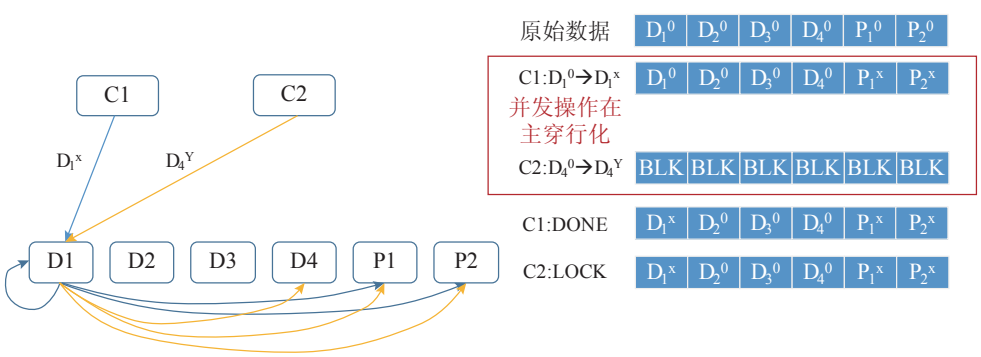


图6-28　2pc erasure code

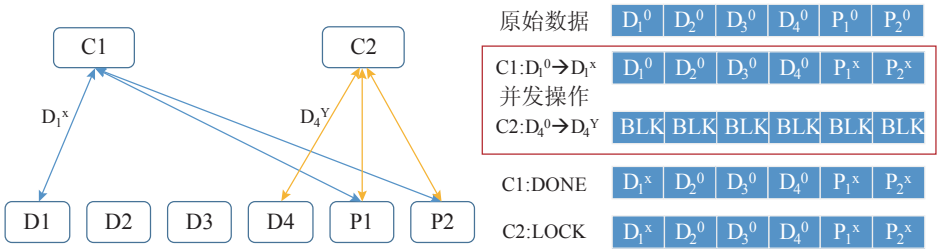


图6-29　分布式锁

应用返回I/O写成功；当积累到一定程度的时候，如果cache中已经积累了满条带，则直接满条带写到后面的存储节点上；如果cache中积累的数据尚未达到满条带，则在后台将EC条带读到Cache中，修改掉变更部分，在cache中保存成一个完整的满条带，然后再以满条带的方式写到后端存储节点上。因为Cache中已经存在满条带了(Cache使用多副本技术)，所以在写EC条带到存储节点的过程中，无须担心因为存储节点的故障带来的条带不一致问题。

2pc erasure code和full-stripe erasure code这两类技术，都有相应的应用。另外，还有一些厂家的存储存储系统实现了离线的Erasure Code方式，与full-stripe erasure code的方式比较类似，数据先存成多副本主存(非Cache)，然后后台逐步地转换成EC，如HDFS RAID。

具体针对不同的场景，应该采用哪种技术来实现Erasure Code，大家可以根据性能和时延的要求，采用上述表格中的几个维度，对不同的设计技术进行权衡，找到最适合自己业务场景的实现方法。

6.6.2　分布式重删压缩

重删压缩，针对某些应用场景，可以有效地提高存储利用率，是现在很多存储系统都会实现的增值特性。

分布式对象存储中的重删压缩，可以节省重复的视频、文件等内容，对于企业同一个部门的文档、资料等具有很高的重复度，重删压缩可以起到比较好的效果。分布式对象存储系统的重删压缩，可以以对象为粒度，也可以以对象的定长数据片或变长数据片为粒度。

分布式块存储中的重删压缩一般用于全SSD场景，这样可以有效节省SSD的写入量，提升SSD的寿命和可用容量。分布式块存储系统中SSD的重删压缩，一般以定长的4K、8K、16K等数据片为粒度，计算每个数据片的指纹，针对重复的数据仅存储一个索引。

分布式块存储中针对SSD的重删压缩，主要的技术难度是低时延，因为针对数据块计算指纹后，在读写过程中，在指纹的查找和匹配方面会存在问题，特别是基于以太网TCP/IP的分布式存储系统中，指纹查找和匹配可能需要跨网络进行访问，会增加I/O时延。目前不同的系统和产品也有各自的实现方法，一般而言分为如下两种。

一种是基于指纹或数据进行Hash散列，实现资源池层面全局的重删压缩，限于篇幅，此处不展开介绍具体的架构设计技术。这种方式可以实现全局资源池的重删压缩，但是在读写过程中存在多次跨网络的分布式交互，对网络pps是一个考验。有些厂家会采用Infiniband或RoCE的网络，一般在多控制器阵列内通过PCIE网络等来实现，这样能够保证一个相对较好的时延。

另一种方式是基于重删服务的方式来实现，特别在云数据中心大规模存储池中，有些应用比较适合重删压缩的，就增加一个重删压缩服务；有些应用不适合重删压缩的，还是走原先的I/O流程完成。服务化的重删压缩设计方法，可以很方便地进行scale-out，每个应用或租户都可以有自己的重删压缩服务实例。

第 7 章

面向自动化、多租户的软件定义网络

7.1 网络虚拟化的驱动力与关键需求

7.1.1 网络虚拟化的驱动力

随着服务器虚拟化和存储虚拟化技术的迅速发展，服务器内开始集成各种虚拟化软件Hypervisor，一台物理服务器内可以同时运行多个虚拟机实例，以最大限度地利用计算资源。同时，随着云服务被越来越广泛地使用，用户可以方便地从云服务提供者直接获取虚拟机的租用服务，大大节省了自建和运维成本。用户可以不用关注虚拟机所在物理服务器的具体型号、具体位置，甚至如何实现，而转为专注于用户自身业务的快速部署和上线应用。

传统的网络虚拟化技术，已经越来越难以满足云时代下用户的需求。例如广泛被使用的VLAN(Virutal Local Area Network)技术，虽然可以在物理交换机上通过划分多个VLAN来隔离，并虚拟出多个逻辑网络，但是其设计和配置，通常基于固定的规划，以及网络和服务器的位置不会频繁变更的假设为前提。而面对云化的数据中心，大量虚拟机的动态的生命周期变化，以及弹性漂移和伸缩的特点，对网络提出了更高的按需配置和随动的需求。通过传统的静态方式的VLAN规划和配置，已经越来越难以满足诉求。网络虚拟化在全系统范围内，能够和硬件相解耦，同时灵活、自动、弹性地适应虚拟化业务，提升对网络资源的利用率，成为云时代下对网络虚拟化的强烈诉求。

为什么传统的网络难以适应云时代的需求？

首先，传统的网络得以迅速地发展，主要源于网络所具备的两个特点。

✍ 无所不在的互联性，能够穿越几乎所有类型的异构网络设备。

✍ 面向无连接的网络特征。把所有和业务状态相关的信息都留在主机(Hosts)，网络中只有独立的数据包，没有任何业务状态，这个使其成功规避了传统数据网络。这个无连接的特点，也使得网络和网络业务彻底解耦，为互联网业务的快速创新提供了可能。

然而，传统网络的这种特质，也造成网络与业务的割裂(见图7-1)。

✍ 网络无法响应业务的快速变化：由于网络和业务割裂，业务的商用速度远远快于网络设备的商用速度，网络需要数年的特性和架构调整，引入新设备，才能满足一个新的业务，网络已经限制了新业务的发展。如：业务的开放周期为几个月，而满足新业务需求的芯片开发通常需要1～2年的时间，至新设备开发出来，到业务部署上线一般需要3年左右的时间。同时，对于云数据中心的虚拟机和虚拟网络运营业务，传统二层的VLAN只支持4096个，无法满足虚拟化的隔离需求，云数据中心要求虚拟机能够按需自动迁移，这些对交换机设备提出了新承载协议的要求，

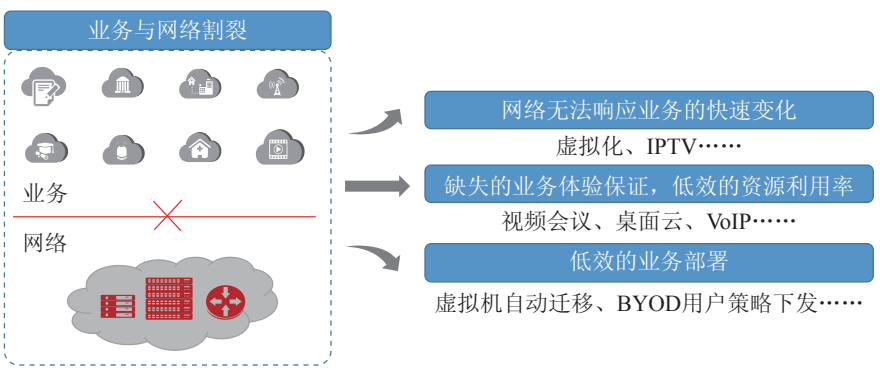


图7-1 传统网络和业务的割裂

此时传统的物理网络设备已经无法及时适应。

✍ 缺失的业务体验保证，低效的资源利用率：网络中只有数据包的流入和流出，并没有任何业务连接的信息，针对业务的质量保证和监控措施很难实施。无法保证端到端的业务体验，例如：在看视频、开视频会议时，会遇到较短时间的卡顿；在语音通话时，出现暂时的模糊音、单通；在用远程云桌面时，特别是进行图形化操作时，响应很慢。同时，大多数公司能利用到网络资源的30%～40%，很多情况下存在大量资源闲置，而某些时候由于数据量周期性的猛增，却发现网络资源不够。除了网络扩容之外，更重要的是提高目前已有网络的利用率。

✍ 低效的业务部署：很多情况下向业务测提供网络资源时都需要手工配置，包括网络交换机端口配置、ACL配置、路由配置等。网络和应用安全策略的部署仍然艰难地通过手工配置，自动化程度很低。网络变更困难，每次人工配置都需要小心翼翼，需要网络专家花费大量的时间来帮忙连接不通的设备，任何小的疏忽都可能酿成网络故障。应用的部署也不得不拖延几天、几周甚至更长时间，直到网络资源最终准备好了为止。网络缺少移动性，网络的配置绑定于硬件。网络配置的状态遍及大量独立的网络设备(物理的和虚拟的)。底层VLAN、网关、防火墙等物理资源的部署限制了计算资源的部署与自由迁移。

7.1.2 对网络虚拟化的关键需求

通过网络能力的软件化，满足业务系统的敏捷、自动化、效率提升，这成为虚拟化和云时代下对网络虚拟化的关键诉求。

首先是敏捷和业务快速上线的诉求，网络不能成为阻碍IT业务的绊脚石，要支撑业务系统的快速创新和上线；其次，网络业务的下发和配置，要改变基于手工或静态配置的方式，支撑业务系统实时、按需、动态化的部署网络业务，从静态网络演进为动态网络，从单点部署演进为整体部署，从人机接口演进为机机接口；最后，要提升网络资源的利用率，从设备和连接为中心的模式，转化为以应用和服务为中心的模式，最大化利用网络资源，同时细分用户的业务流量，提供不同SLA的保障。

具体来说，自下而上，对网络的需求包括如下几点。

✍ 与物理层解耦：网络虚拟化的目标是接管所有的网络服务、特性和应用的虚拟网络必要的配置(VLANs、VRFs、防火墙规则、负载均衡池、IPAM、路由、隔离、多租户等)。从复杂的物理网络中抽取出简化的逻辑网络设备和服务，将这些逻辑对象映射给分布式虚拟化层，通过网络控制器和云管理平台的接口来消费这些虚拟网络服务。应用只需和虚拟化网络层打交道，复杂的网络硬件本身作为底层实现，对用户的网络控制面屏蔽。

✍ 共享物理网络，支持多租户平面及安全隔离：计算虚拟化使多种业务或不同租户资源共享同一个数据中心资源，网络资源也同样需要共享，在同一个物理网络平面，需要为多租户提供逻辑的、安全隔离的网络平面。

✍ 网络按需自动化配置：通过API自动化部署，一个完整的、功能丰富的虚拟网络可以自由定义从任何约束在物理交换基础上的设施功能、拓扑或资源。通过网络虚拟化，每个应用的虚拟网络和安全拓扑就拥有了移动性，同时实现了和流动的计算层的绑定，并且可通过API自动部署，又确保了和专有物理硬件解耦。

✍ 网络服务抽象：虚拟网络层可以提供逻辑端口、逻辑交换机和路由器、分布式虚拟防火墙、虚拟负载均衡器等，并可同时确保这些网络设备和服务的监控、QoS和安全。这些逻辑网络对象就像服务器虚拟化虚拟出来的vCPU和内存一样，可以提供给用户，实现任意转发策略，安全策略的自由组合，构筑任意拓扑的虚拟网络。

7.1.3 主流SDN框架

解决网络虚拟化的问题，存在不同的方式，SDN(Software Define Network)是其中之一。SDN并不是具体的技术，而是一种全新的网络设计框

架，SDN的核心理念是改变传统网络对数据流的控制方式。在传统网络中，网络采用分布式控制面，报文从源到目的的转发行为由各个网络节点自己独立控制和完成，每个网络节点都需要独立的配置。 SDN框架中的网络，控制面与转发面是分离的，转发面与具体协议无关(见图7-2、图7-3)。

(1) 控制面与转发面分离：控制面更灵活，转发面抽象与标准化。

(2) 集中化的网络控制：控制软件具备全局网络视图，策略转发规则集中。

(3) 网络开放可编程。

(4) 网络业务的自动化应用程序控制。

对于SDN业界并无标准的理解，具体含义与其运行的网络领域和其使用的策略和协议相关。以下是几种SDN思路，如图7-4所示。

一、OpenFlow SDN

OpenFlow起源于斯坦福大学的Clean Slate项目组。该项目的最终目的是重新发明互联网，旨在改变设计已略显不合时宜，且难以进化发展的现有网络基础架构。在2006年，斯坦福的学生

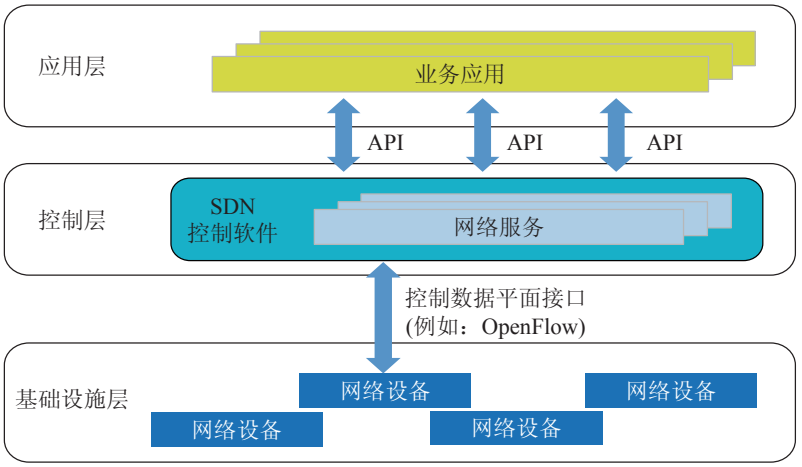


图7-2 SDN 架构

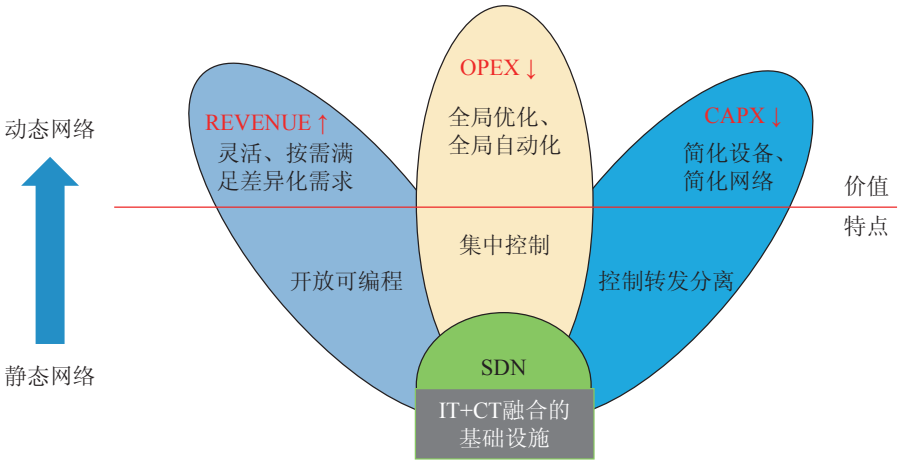


图7-3 SDN 特点与价值

Openflow模式	Overlay模式	设备API模式

单点设备内部可编程
• 网络维度
  • 网络设备底层转发平面的可编程
  • 控制灵活性很大
• 用户使用维度
  • 使用难度很大
	DC虚拟网络的可编程
	• 网络维度
	  • 基于服务器或网络设备的可编程，提供虚拟化网络的可编程
	  • 控制灵活性较大
	• 用户使用维度
	  • 使用难度较小
		单点设备外部接口可编程
		• 网络维度
		  • 设备的可编程，而非网络可编程，基于局部视角解决问题，而非整体
		  • 控制灵活性适中
		• 用户使用维度
		  • 使用难度适中

图7-4　SDN 路线

Martin Casado领导了一个关于网络安全与管理的项目Ethane，该项目试图通过一个集中式的控制器，让网络管理员可以方便地定义基于网络流的安全控制策略，并将这些安全策略应用到各种网络设备中，从而实现对整个网络通信的安全控制。受此项目(及Ethane的前续项目Sane)启发，Martin和他的导师Nick McKeown教授(时任Clean Slate项目的Faculty Director)发现，如果将Ethane的设计更一般化，将传统网络设备的数据转发(Data Plane)和路由控制(Control Plane)两个功能模块相分离，通过集中式的控制器(Controller)以标准化的接口对各种网络设备进行管理和配置，那么这将为网络资源的设计、管理和使用提供更多的可能性，从而更容易推动网络的革新与发展。于是，他们便提出了OpenFlow的概念，并且Nick McKeown等人于2008年在ACM SIGCOMM(美国计算机协会数据通信专业组会议)发表了题为“OpenFlow: Enabling Innovation in Campus Networks”的论文，首次详细地介绍了OpenFlow的概念。

SDN的设计理念是将网络的控制面与数据转发面进行分离，并实现可编程化控制。SDN的典型架构共分三层，最上层为应用层，包括各种不同的业务和应用；中间的控制层主要负责处理数据平面资源的编排，维护网络拓扑、状态信息等，在OpenFlow环境中，控制器会使用OpenFlow协议和Netconf协议与交换机联系(OpenFlow是将流数据发送到交换机的API，而NETCONF是网络配置API)。最底层的基础设施层负责进行基于流表的数据处理、转发和状态收集(见图7-5)。

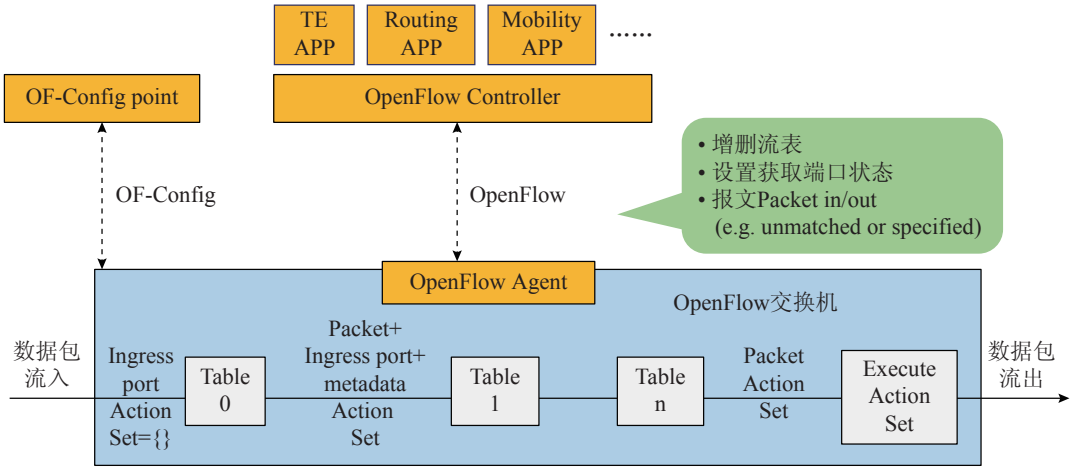


图7-5　基于OpenFlow的控制与转发分离体系结构

SDN本质上具有“控制和转发分离”、“设备资源虚拟化”和“通用硬件及软件可编程”三大特性，这带来了一系列的好处。

(1) 设备硬件归一化，硬件只关注转发和存储能力，与业务特性解耦，可以采用相对廉价的商用架构来实现。

(2) 网络的智能性全部由软件实现，网络设备的种类及功能由软件配置而定，对网络的操作控制和运行由服务器作为网络操作系统(NOS)来完成。

(3) 对业务响应相对更快，可以定制各种网络参数，如路由、安全、策略、QoS、流量工程等，并实时配置到网络中，开通具体业务的时间将缩短。

ONF SDN的三大要素，具体如下。

(1) 转发与控制分离，这使得网络交换机的数据转发变得更加简单、快速；同时，控制变成了网络操作系统中一个相对集中的逻辑功能。

(2) OpenFlow协议，它向交换机传送转发表，交换机依此转发报文。这种做法与传统网络完全不同。在传统网络架构中，交换机和路由器需要自己决定报文的转发路径，这可能会给网络运营商带来一些不可预知的负面影响，包括成本增加、性能降低、上市时间延缓等。有了SDN，控制软件决定报文的转发路径，使得运营商可以“随心所欲”地控制网络。

(3) 具有一致性的、全系统范围的网络操作系统可编程接口，它能让网络实现真正意义上的可编程或者软件定义。如果不能实现转发与控制分离，那么几乎所有SDN所能带来的好处都无法体现；如果能实现转发和控制分离，但没有OpenFlow协议，那么就需要通过其他途径，将所需要的流量表信息传递给交换机。OpenFlow就是实现这一功能的行业标准。

OpenFlow/SDN吸引了业界越来越多的关注，成为近年来名副其实的热门技术。目前，包括HP、IBM、Cisco、NEC以及国内的华为和中兴等在内的传统网络设备制造商都已纷纷加入OpenFlow的阵营，同时有一些支持OpenFlow的网络硬件设备已经面世。2011年，开放网络基金会(Open Networking Foundation)在Nick等人的推动下成立，专门负责OpenFlow标准和规范的维护和发展；同年，第一届开放网络峰会(OpenNetworking Summit)召开，为OpenFlow和SDN在学术界和工业界都做了很好的介绍和推广。2013年召开的第二届峰会上，来自Google的Urs Hölzle在以“OpenFlow@Google”为题的Keynote演讲中宣布Google已经在其全球各地的数据中心骨干网络中大规模地使用OpenFlow/SDN，从而证明了OpenFlow不再是仅仅停留在学术界的一个研究模型，而是已经完全具备了可以在产品环境中应用的技术成熟度。最近，Facebook也宣布其数据中心中使用了OpenFlow/SDN的技术。

二、Open Flow协议介绍

自2010年初发布第一个版本(v1.0)以来，OpenFlow规范已经经历了1.1、1.2、1.3、1.4等版本。同时，OpenFlow管理和配置协议也发布了第一个版本(OF-Config 1.0 & 1.1)。

1. OF规范

OF规范主要分为如下四大部分。

(1) OpenFlow的端口(Port)

OpenFlow规范将Switch上的端口分为三种类别：

- 物理端口，即设备上物理可见的端口；
- 逻辑端口，在物理端口基础上由Switch设备抽象出来的逻辑端口，如为tunnel或者聚合等功能而实现的逻辑端口；
- OpenFlow目前总共定义了ALL、CONTROLLER、TABLE、IN_PORT、ANY、LOCAL、NORMAL和FLOOD这8种端口，其中后三种为非必需的端口，只在混合型的OpenFlow Switch(OpenFlow-hybrid Switch，即同时支持传统网络协议栈和OpenFlow协议的Switch设备，相对于OpenFlow-only Switch而言)中存在。

(2) OpenFlow的FlowTable(国内直译为“流表”)

OpenFlow通过用户定义的或者预设的规则

来匹配和处理网络包。一条OpenFlow的规则由匹配域(Match Fields)、优先级(Priority)、处理指令(Instructions)、统计数据(如Counters)、超时时间(Timeout)、附属属性(Cookie)等字段组成，如图7-6所示。

在一条规则中，可以根据网络包在L2、L3或者L4等网络报文头的任意字段进行匹配，比如以太网帧的源MAC地址，IP包的协议类型和IP地址，或者TCP/UDP的端口号等。目前OpenFlow的规范中还规定了Switch设备厂商可以选择性地支持通配符进行匹配。据说，OpenFlow在未来还计划支持对整个数据包的任意字段进行匹配。

所有OpenFlow的规则都被组织在不同的FlowTable中，在同一个FlowTable中按规则的优先级进行先后匹配(见图7-6)。一个OpenFlow的Switch可以包含一个或者多个FlowTable，从0依次编号排列。OpenFlow规范中定义了流水线式的处理流程，如图7-7所示。当数据包进入Switch后，必须从FlowTable 0开始依次匹配；FlowTable可以按次序从小到大越级跳转，但不能从某一FlowTable向前跳转至编号更小的FlowTable。当数据包成功匹配一条规则后，将首先更新该规则对应的统计数据(如成功匹配数据包总数目和总字节数等)，然后根据规则中的指令进行相应操作——比如跳转至后续某一FlowTable继续处理，修改或者立即执行该数据包对应的Action Set等。当数据包已经处于最后一个FlowTable时，其对应的Action Set中的所有Action将被执行，包括转发至某一端口，修改数据包某一字段，丢弃数据包等。OpenFlow规范中对目前所支持的Instructions和Actions进行了完整详细的说明和定义。

另外，OpenFlow规范中还定义了很多其他功能和行为，比如OpenFlow对于QoS的支持(即MeterTable和Meter Bands的定义等)，对于GroupTable的定义，以及规则的超时处理等。

(3) OpenFlow的通信通道

OpenFlow规范定义了一个OpenFlow Switch如何与Controller建立连接、通信以及相关消息类型等。

OpenFlow规范中定义了三种消息类型。

- Controller/Switch消息，是指由Controller

Mach Fields	Priority	Counters	Instructions	Timeouts	Cookie

图7-6　OpenFlow规则

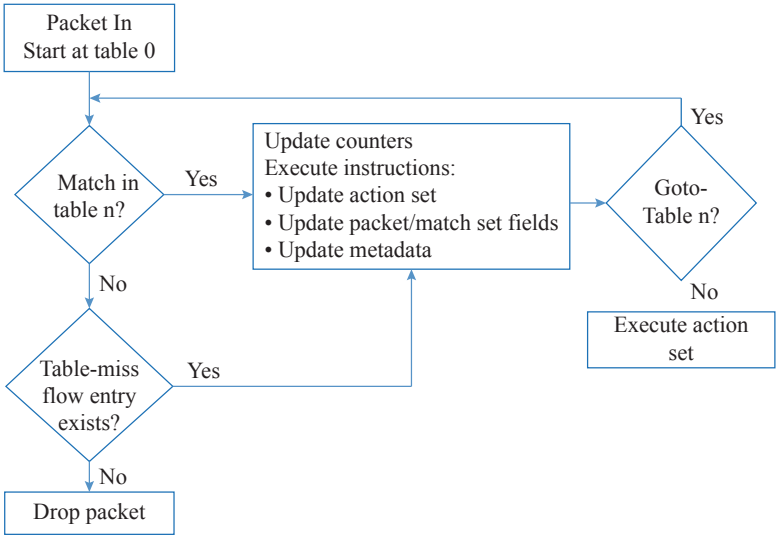


图7-7　OpenFlow流表匹配过程

发起、Switch接收并处理的消息，主要包括Features、Configuration、Modify-State、Read-State、Packet-out、Barrier和Role-Request等消息。这些消息主要由Controller用来对Switch进行状态查询和修改配置等操作。

✍ 异步(Asynchronous)消息，是由Switch发送给Controller、用来通知Switch上发生的某些异步事件的消息，主要包括Packet-in、Flow-Removed、Port-status和Error等。例如，当某一条规则因为超时而被删除时，Switch将自动发送一条Flow-Removed消息通知Controller，以方便Controller做出相应的操作，如重新设置相关规则等。

✍ 对称(Symmetric)消息，顾名思义，这些都是双向对称的消息，主要用来建立连接、检测对方是否在线等，包括Hello、Echo和Experimenter三种消息。

图7-8展示了OpenFlow和Switch之间一次典型的消息交换过程，出于安全和高可用性等方面的考虑，OpenFlow的规范还规定了如何为Controller和Switch之间的信道加密、如何建立多连接等(主连接和辅助连接)。

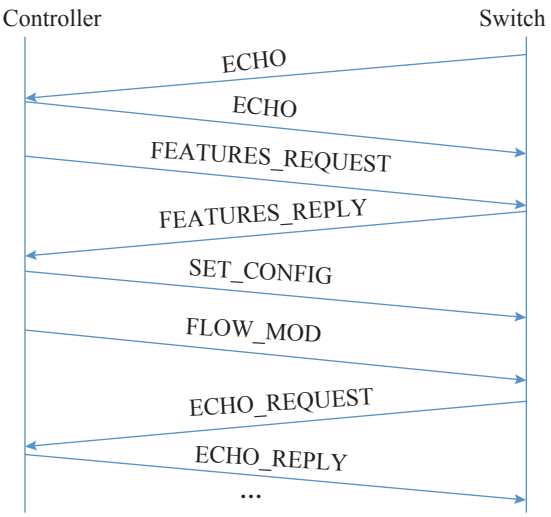


图7-8 控制器与Switch交互过程

(4) OpenFlow协议及相关数据结构

在OpenFlow规范的最后一部分，主要详细定义了各种OpenFlow消息的数据结构，包括OpenFlow消息的消息头等。

2. OF-Config

OF-Config是OpenFlow的伴侣协议。OpenFlow实现了Flow的match-action相关行为，但其他Flow依赖的资源都依赖OF-Config进行管理，包括：配置OpenFlow Controller地址、队列和物理端口、逻辑端口、通信信道、交换机能力发现等。对转发面的配置与管理，也有定义私有协议方案，如Open vSwitch使用自定义的OVSDB。

三、API SDN

IETF作为传统网络架构的制定者，其核心思路是重用当前的技术而不是OpenFlow，并关注重点设备控制面的功能与开放API。

XML-based SDN(Software-Defined Networking)，使用Netconf等设备存在的接口对现有设备进行配置，对当前设备不修改(见图7-9)。

IETF I2RS(Interface to Routing System：路由系统接口)工作组希望将路由协议中的策略配置运行在集中的控制器上，控制器通过设备反馈的事件、路径拓扑和网络流量信息来动态下发路由状态、策略到设备上，具体的路由计算还是由各个网络设备分布式完成。

四、ONF及OpenDayLight标准联盟

开放网络基金会(ONF，Open Networking Foundation)是一个组织机构，致力于软件定义网络(SDN)的发展和标准化。ONF的主要任务是培养一个网络环境，这种环境能够支持OpenFlow，OpenFlow是一个允许服务器通知交换器往哪里发送数据包的协议。ONF的董事会成员包括微软、Google和Verizon。普通会员有几十个，包括思科、富士通、IBM、NEC、三星和惠普。

2013年4月，SDN行业组织OpenDaylight宣告成立。其成员包括Arista、Big Switch、博科、思科、思杰、戴尔、爱立信、富士通、IBM、英特尔、瞻博网络、微软、华为、NEC、Nuage Networks、PLUMgrid、红帽、VMware。在OpenDaylight项目中，网络行业将采取相同的方

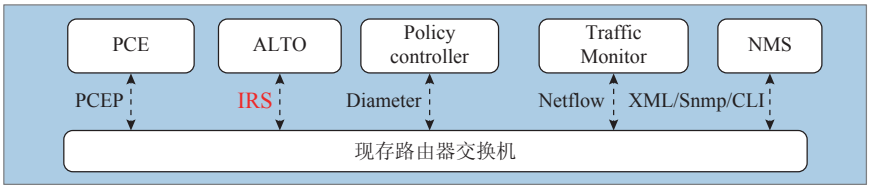


图7-9　I2RS与其他现存设备接口

案研发他们的下一代技术，整体情况类似于大数据领域中利用Hadoop或带有WebKit的Web浏览器进行研发一样。OpenDaylight是一个研发实体，未来将与作为标准化实体的ONF形成互补。

OpenDaylight是一套以社区为主导的开源框架，旨在推动创新实施以及软件定义网络(简称SDN)透明化。面对SDN型网络，大家需要合适的工具帮助自己管理基础设施，这正是OpenDaylight的专长。作为项目核心，OpenDaylight拥有一套模块化、可插拔且极为灵活的控制器，这使其能够被部署在任何支持Java的平台之上。这款控制器中还包含一套模块合集，能够执行需要快速完成的网络任务(见图7-10)。

OpenDaylight项目将研发一系列技术，包括在SDN中为网络设备提供集中控制的控制器。该控制器为云管理工具等具有网络感知功能的应用、交换机和其他网络中的硬件提供接口。该项目还将创建网络应用、网络虚拟化软件和其他组件。

如此众多的重量级厂商走到一起，说明SDN已成为一股不可忽视的潮流。不过，显而易见的问题是，种种不同的技术、见解如何真正聚合成一个切实可行的解决方案，面临的困难想来不少。因此，其发展前景尚待观察。这些重量级厂商汇聚在一起，一方面堵死了SDN新兴企业的发展壮大之路，另一方面也形成了和VMware生态系统抗衡的态势。

五、Overlay SDN

在数据中心网络中，“区域”对应于VLAN

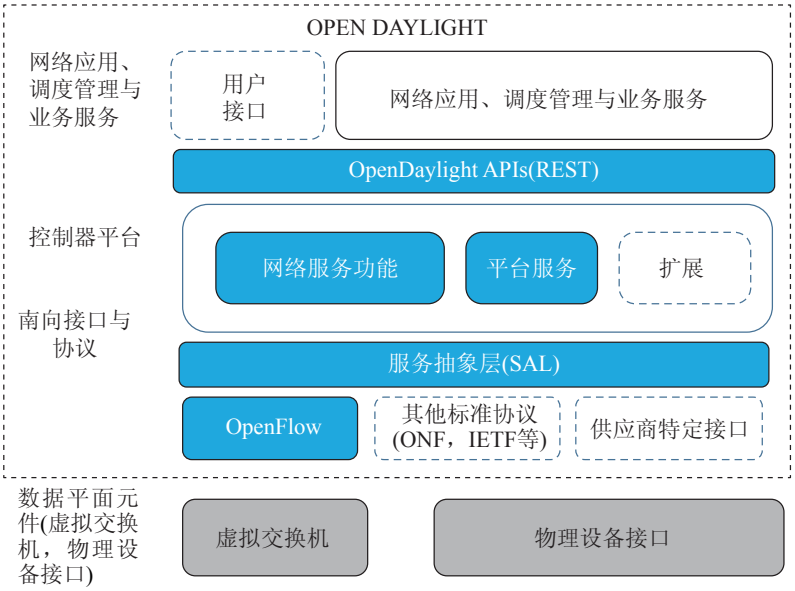


图7-10　OpenDaylight功能逻辑架构

的划分。相同VLAN内的终端属于同一广播域，具有一致的VLAN-ID，二层连通；不同VLAN内的终端需要通过网关互相访问，二层隔离，三层连通。传统的数据中心主要是依据功能进行区域划分，例如Web、APP、DB、办公区、业务区、内联区、外联区等。不同区域之间通过网关和安全设备互访，保证不同区域的可靠性、安全性。同时，不同区域由于具有不同的功能，因此需要相互访问数据时，只需终端之间能够通信，并不一定要求通信双方处于同一VLAN或二层网络。

Overlay SDN，是将网络报文封装后，在更大的范围内进行扩展的方案，可以很好地解决虚拟机灵活迁移的问题。

Overlay在网络技术领域，指的是一种网络架构上叠加的虚拟化技术模式，其大体框架是对基础网络不进行大规模修改的条件下，实现应用在网络上的承载，并能与其他网络业务分离，并且以基于IP的基础网络技术为主。Overlay采用全新方式的解决以下问题。

(1) 虚拟机迁移范围受到网络架构限制

Overlay是一种封装在IP报文之上的新的数据格式，因此这种数据可以通过路由的方式在网络中分发，而路由网络本身并无特殊网络结构限制，具备良性大规模扩展能力，并且对设备本身无特殊要求，以高性能路由转发为佳，且路由网络本身具备很强的故障自愈能力、负载均衡能力。采用Overlay技术后，企业部署的现有网络便可用于支撑新的云计算业务，改造难度极低(除性能可能是考量因素外，技术上对于承载网络并无新的要求)。

(2) 虚拟机规模受网络规格限制

虚拟机数据封装在IP数据包中后，对网络只表现为封装后的网络参数，即隧道端点的地址，因此，对于承载网络(特别是接入交换机)，MAC地址规格需求极大降低，最低规格也就是几十个(每个端口一台物理服务器的隧道端点MAC)。

(3) 网络隔离/分离能力限制

针对VLAN数量4000以内的限制，在Overlay技术中引入了类似12比特VLAN ID的用户标识，支持千万级以上的用户标识，并且在Overlay中沿袭了云计算“租户”的概念，称之为Tenant ID(租户标识)，用24或64比特表示。针对VLAN技术下网络的TRUANK ALL(VLAN穿透所有设备)的问题，Overlay对网络的VLAN配置无要求，可以避免网络本身的无效流量带宽浪费，同时Overlay的二层连通基于虚拟主机业务需求创建，在云的环境中全局可控。

IETF在Overlay技术领域有如下三大技术路线。

(1) VxLAN

VxLAN与NVGRE类似，只是外层IP头和内层IP头之间加的不是GRE头，而是其他自定义的头部。

VxLAN的报文封装格式如图7-11所示。

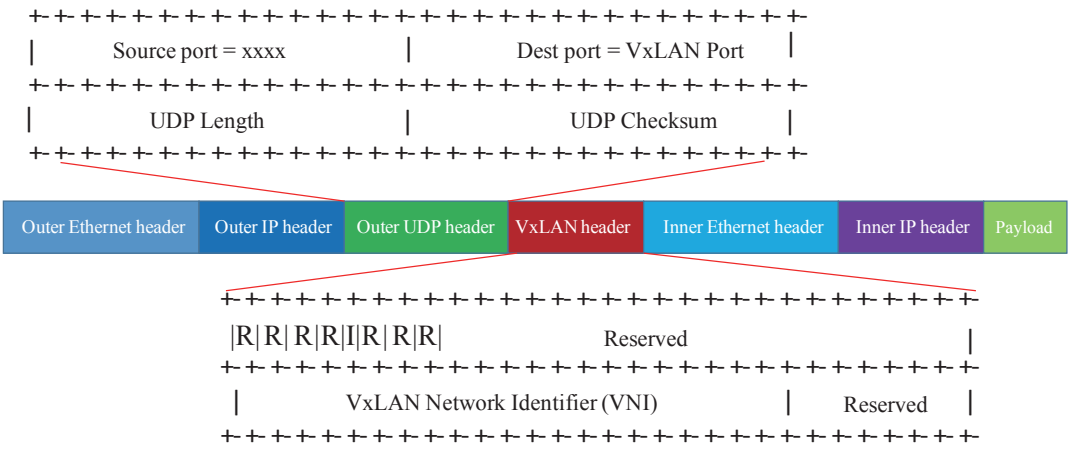


图7-11 VxLAN报文封装格式

VxLAN和NVGRE的最大不同之处在于在外层IP头后面增加了UDP头，这样可以利用现在很多网卡支持的一些加速特性，从而在使用服务器实现VxLAN的封装和解封装时获得更高的性能。

VxLAN是由VMware主导的，现在也在IETF提了标准草案。

(2) NVGRE

NVGRE是由微软主导的，目前已经在IETF提了标准草案。该技术就是在原始IP报文的基础上添加GRE封装和外层IP头，并将租户信息存放在GRE头中，这样虚拟机可以在网络内任意迁移，只要访问它的虚拟机或者主机，就知道它位于哪个物理服务器了。NVGRE报文封装如图7-12所示。

图7-12中，GRE头中的VSID就是用来标识一个租户或者租户的一个子网。该字段为24个比特，可以支持多达16M的租户，在可预见的将来应该是足够用的。外层IP头中的源和目的IP用来标识进行NVGRE封装和解封装的网络节点，这种节点在标准中称为网络虚拟化元素(NVE：Network Virtulization Element)。如果NVE在服务器内，则外层IP头中的源和目的IP就分别用来标识源虚拟机和目的虚拟机所在的服务器。

(3) STT

STT的报文封装格式如图7-13所示。

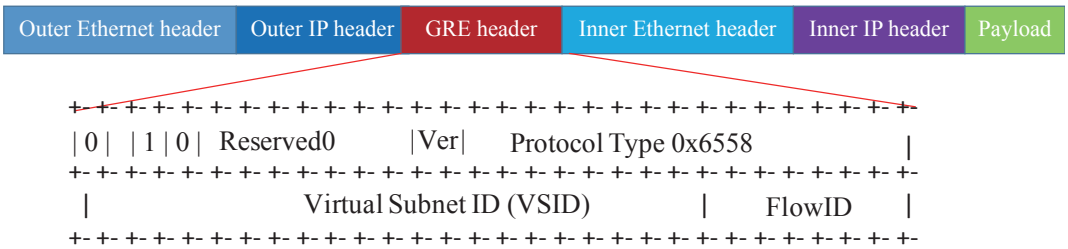


图7-12　NVGRE报文封装格式

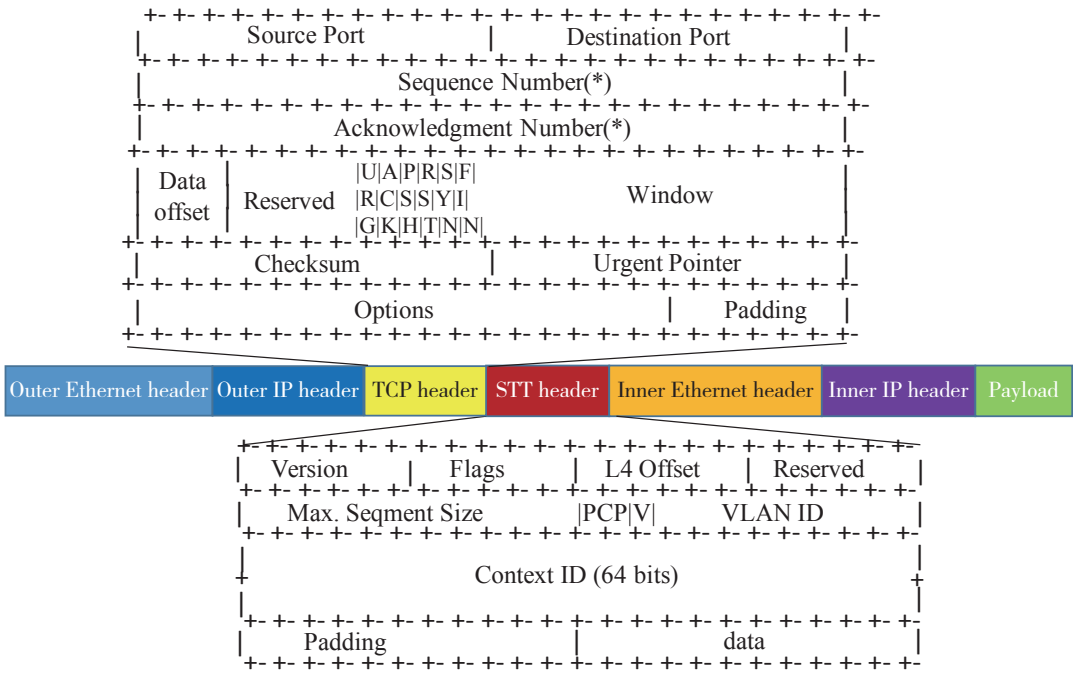


图7-13　STT报文封装格式

STT报文的封装和VxLAN的更为相似一些，只是STT在外层IP头后面采用的是类似TCP头而不是UDP头，但也可以利用网卡的加速特性，从而在采用服务器完成STT报文的封装和解封装时获得更高的性能。还有一个不同之处在于STT封装报文中的Context ID为64bit，而不像NVGRE和VxLAN中的24bit租户标识，它可以用做更细粒度的标识。

这三种二层Overlay技术，大体思路均是将以太网报文承载到某种隧道层面，差异性在于选择和构造隧道的不同，而底层均是IP转发。VxLAN和STT对于现网设备对流量均衡要求较低，即负载链路负载分担适应性好，一般的网络设备都能对L2～L4的数据内容参数进行链路聚合或等价路由的流量均衡，而NVGRE则需要网络设备对GRE扩展头感知并对flow ID进行HASH，并且需要硬件升级；STT对于TCP有较大修改，隧道模式接近UDP性质，隧道构造技术属于革新性，且复杂度较高，而VxLAN利用了现有通用的UDP传输，成熟性极高。总体比较，VxLAN技术相对具有优势。

7.2 软件Overlay SDN网络，L2/L3网络

Overlay SDN由于其可以在现有网络的架构上叠加虚拟化技术，因而可以对基础网络不进行大规模的修改下，实现应用在网络上的承载，并能与其他网络业务分离。Overlay的网络是物理网络向云和虚拟化的延伸，使云资源池化能力可以摆脱物理网络的重重限制，是实现云网融合的关键。

由于软件的灵活性，以及现在CPU的能力不断增强，加上服务器虚拟化技术的助力，现在业界网络设备由软件实现的趋势越来越明显，从最基本的交换机，到防火墙和负载均衡器等，无不如此。

vSwitch的出现最早是因为VM迁移后配置在网络设备上的相关策略无法随之迁移，因此服务器虚拟化厂商就在Hypervisor上内嵌了vSwitch功能，由vSwitch取代原来的物理接入交换机执行基本的二层转发和接入策略执行功能。后来发现，接入策略等本来就是通过IT系统下发的，vSwitch和虚拟机等也都是IT系统进行管理的，由vSwitch来执行策略，对整个策略管理和部署带来了简化。另外，采用软件实现的vSwitch可以快速实现各种新的转发技术，满足当前解决数据中心网络遇到的各种问题的要求。因此，渐渐地，vSwitch成为了Hypervisor的一个重要部件，也成为了IT和CT场上的争夺焦点之一。

vSwitch的争夺最典型的就是VMware的DVS和Cisco的Nexus 1000V。VMware最早采用Cisco的Nexus 1000V作为其虚拟化平台的vSwitch，但是它自己开发的DVS也逐渐成熟，逐渐用DVS取代了Nexus 1000V，使两家原本亲密无间的合作关系蒙上了一层阴影。另外从业界可获得的vSwitch数量，特别是能够运行在VMware和微软的Hypervisor上的vSwich的数量非常有限这一点也可以看出vSwich的重要性。

业界其他有名的vSwich还包括由Nicira开发并开源的Open vSwitch、IBM的DOVE等。微软也在其Hyper-V中内嵌了vSwitch。

目前的虚拟化主机软件在vSwitch内支持VxLAN，使用VTEP(VxLAN Tunnel End Point)封装和终结VxLAN的隧道。为了使得VxLAN Overlay网络更加简化运行管理，便于云的服务提供，各厂家使用集中控制的模型，将分散在多个物理服务器上的vSwitch构成一个大型的、虚拟化的分布式Overlay vSwitch。只要在分布式vSwitch范围内，虚拟机在不同物理服务器上的迁移，便被视为在一个虚拟的设备上迁移，如此大大降低了云中资源的调度难度和复杂度。

7.2.1 Open vSwitch

Open vSwitch是一个开源软件，是利用虚拟平台，通过软件的方式形成的交换机部件。跟传统的物理交换机相比，Open vSwitch配置更加灵活，一台普通的服务器可以配置出数十台甚至上百台虚拟交换机，且端口数目可以灵活选择。由

于大部分的代码是使用平台独立的C写成，所以可移植性非常好。

图7-14显示了Open vSwitch的架构。网卡收到数据包后，会交给datapath内核模块处理，当匹配到对应的datapath会直接输出，如果没有匹配到，会交给用户态的ovs-vswitchd查询flow，用户态处理后，会把处理完的数据包输出到正确的端口，并且设置新的datapath规则，后续数据包可以通过新的datapath规则实现快速转发。

Open vSwitch的主要模块如图7-15所示。这些组件在Linux上表现为各自独立的程序，它们之间的交互通过图中的方式进行。

- ovsdb-server：轻量级的数据库服务，主要保存了整个OVS的配置信息，包括接口、交换内容、VLAN等。ovs-vswitchd会根据数据库中的配置信息工作。
- ovs-vswitchd：守护程序，实现交换功能，和Linux内核兼容模块一起，实现基于流的交换。
- ovs-dpctl：用来配置交换机内核模块的工具，可以控制转发规则。
- ovs-vsctl：用于获取或者修改ovs-vswitchd的配置信息，操作的时候会更新ovsdb-server中的数据库。
- ovs-appctl：主要用于向OVS守护进程发

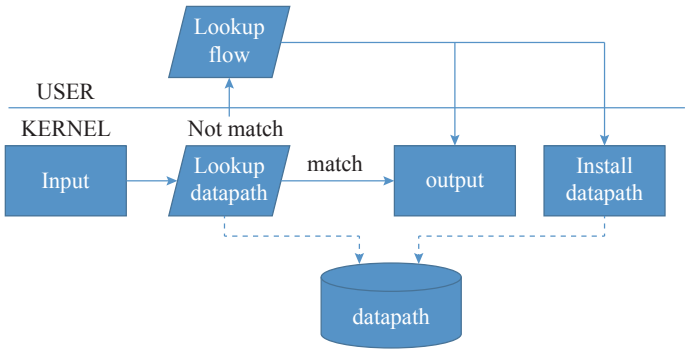


图7-14　Open vSwitch架构

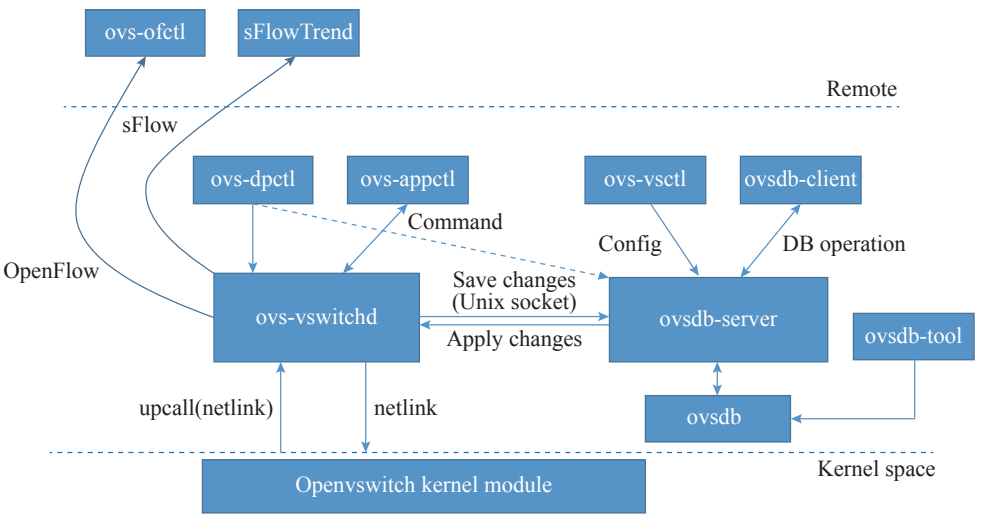


图7-15　Open vSwitch模块

送命令。

✍ ovsdbmonitor：GUI工具，用于显示ovsdb-server中数据信息。

✍ ovs-ofctl：用于控制OVS作为OpenFlow交换机工作时候的流表内容。

7.2.2 软件Overlay SDN的转发面

一、Overlay L2数据流

图7-16描述了基于VxLAN的跨主机L2数据流量：

✍ VM1发送同网段流量至VM2，经过br-int匹配流表，打上VM1子网本地的Vlan Tag后，流量转发至br-tun；

✍ br-tun匹配流表，剥离VM1子网本地Vlan Tag后直接进行VxLAN封装，通过承载网转发至对端主机节点；

✍ VM2所在主机节点收到流量后进行VxLAN解封装，并且加上VM2子网本地Vlan Tag，将流量转发至br-int；

✍ br-int匹配流表，剥离本地VM2子网本地Vlan Tag，将流量转发至VM2。

二、Overlay L3数据流

对于跨子网的L3通信，在OpenStack中，传统的集中式L3 Router不论东西向还是南北向的流量都需要经由网络节点的虚拟路由器，会造成流量瓶颈的问题。为了解决流量瓶颈的问题，产生了DVR(分布式路由器)技术。

DVR的核心作用是，通过东西向横向流量扩展，将vRouter的路由功能同时分布于计算节点和网络节点上，减轻网络节点的性能瓶颈/流量集中。当某个租户建立了一个vRouter的时候，如果某台主机上存在其路由所连接子网的虚机，那么这台主机上就会建立一个分布式vRouter。所有的东西向流量都会由这个路由进行转发而不是通过网络节点进行转发。

通过使用 DVR，三层的转发功能都会被分布到计算节点上，这意味着计算节点也有了网络节点的功能。

图7-17描述了基于VxLAN的跨主机L3数据流量。

✍ 从vm1中发出带有vm2 ip目的IP的数据流，首先发往本网段网络的默认网关mac，上行至br-int，匹配流表，发送数据流直接发送至本地DVR VM1网关。

✍ 本地DVR路由器VM1网关接口接受这个数据帧，然后路由这个数据帧。

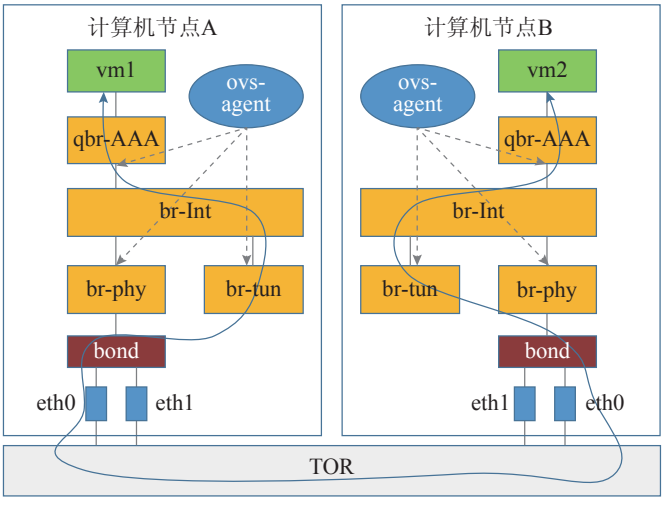


图7-16 Overlay L2数据流

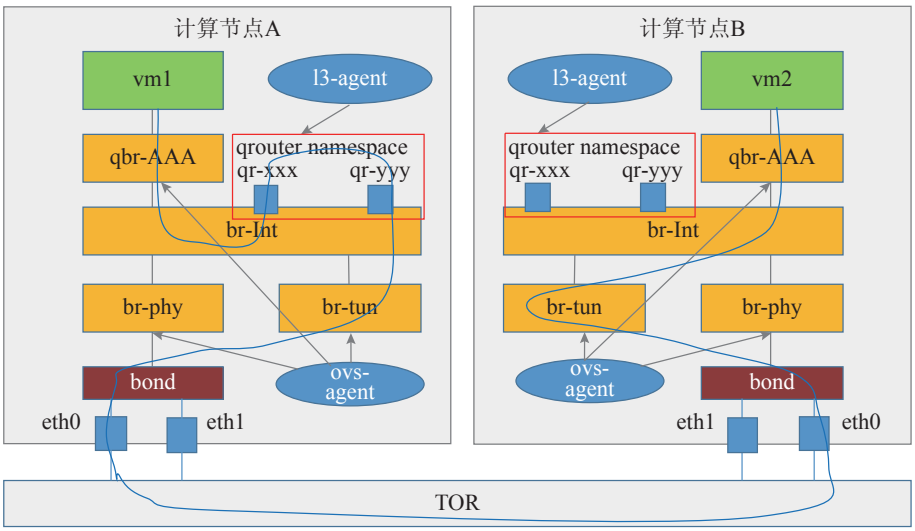


图7-17　Overlay L3数据流

✍ 路由之后，本地DVR将这个数据帧发送到VM2网关接口，这个数据帧被br-int交换到br-tun，并且打上VM2子网的本地VLANtag。

✍ 在计算节点A上的br-tun用节点上唯一的DVR mac地址代替帧的源mac地址。更改后的数据帧通过br-tun发送到计算节点B，在发送前他也去除了VM2子网本地VLANtag，并打上隧道vni VxLAN id。

✍ 计算节点B上br-tun收到这个隧道数据帧，去除VM2 对应的vni标签。随后加上本地Vm2子网本地VLAN tag，然后发送这个帧到br-int。

✍ 计算节点B上的br-int识别到数据帧的源mac地址是一个独特的DVR mac地址之后，将这个mac地址替换成VM2子网的mac地址，然后发送这个数据帧给vm2。

7.2.3　OpenStack Neutron L2/L3控制面

Neutron是OpenStack提供网络服务的专用组件，随着Neutron逐渐成熟，越来越多的云计算厂家开始选用Neutron组件，并且有大量网络设备商和云计算方案厂商为Neutron提供网络设备和网络方案，使得Neutron得到迅速发展，并逐渐成为云平台网络的事实接口标准。

类似于计算、存储节点被虚拟化为计算、存储资源池，OpenStack所在的整个物理网络在Neutron中也被虚拟化为网络资源池。通过对网络资源的划分和可扩展性，Neutron能够为每个租户提供独立的虚拟网络环境。

Neutron的L2/L3控制面，分别提供了二层vSwitch交换和三层vRouter路由的抽象的功能，对应于物理网络环境中的交换机和路由器实现。其具体实现了如下功能。

✍ Network：对应于一个真实物理网络中的二层局域网(VLAN)，从租户的角度而言，其是租户私有的。

✍ Subnet：为网络中的三层概念，指定一段IPV4或IPV6地址并描述其相关的配置信息。它附加在一个二层Network上，指明属于这个Network的虚拟机可使用的IP地址范围。

✍ Router：为租户提供路由、NAT等服务。

Neutron包含Neutron Server、Plugin以及Agent三部分组件，用以支持通过插件形式，结合后端的各种网络转发面实体，构建OpenStack的网络服务，其逻辑架构如图7-18所示。

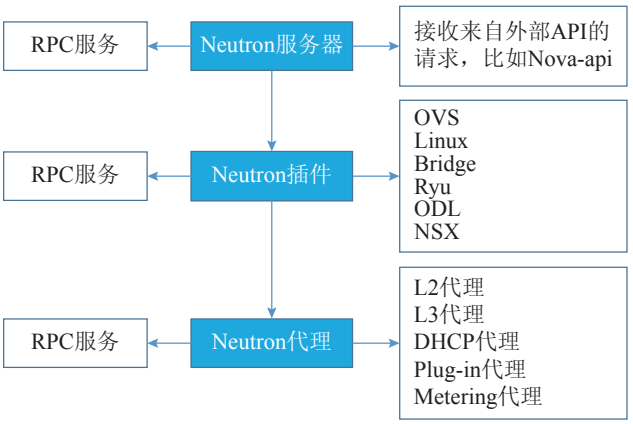


图7-18　Neutron逻辑架构

一、Neutron Server

Neutron Server位于最上部，负责接受来自外部服务的API请求，如管理平台的网络配置，以及计算服务Nova创建端口的请求。

Neutron提供的众多API资源对应了各种虚拟网络资源。其中L2的抽象Network/Subnet/Port可以被认为是核心资源API(Core API)，其他层次的抽象，包括Router以及众多的高层次服务则是扩展资源API(Extension API)。

二、Neutron Plugin

为了支持扩展架构，Neutron项目利用Plugin的方式组织代码，每一个Plugin支持一组API资源并完成特定的操作，这些操作最终由Plugin通过RPC调用相应的Agent来完成。

这些Plugin又被做了一些区分，一些提供基础二层虚拟网络支持的Plugin称为Core Plugin。而Core Plugin之外的其他Plugin则被称为Service Plugin，比如提供防火墙服务的FWaaS等。

因为各种Core Plugin的实现之间存在很多重复的代码，比如对数据库的CRUD等操作。自OpenStack H 版起，Neutron实现了一个ML2 Core Plugin，它采用了更加灵活的结构进行实现，通过Driver的形式对现有的各种Core Plugin提供支持。

ML2作为L2的总控，其实现包括Type和Mechanism两部分，每部分又分为Manager和Driver。Type指的是L2网络的类型(如Flat、VLAN、VxLAN等)，与厂家实现无关。Mechanism则是各个厂家自己设备机制的实现(见图7-19)。

同时，为了解决采用Overlay隧道后的广播报文(ARP、DHCP等)和单播MAC地址位置学习问题，业界采用了一些创新的技术手段。例如在开源OpenStack中的ML2 Plugin采用l2-Population机制在虚拟机创建时预下发本子网的所有虚拟机

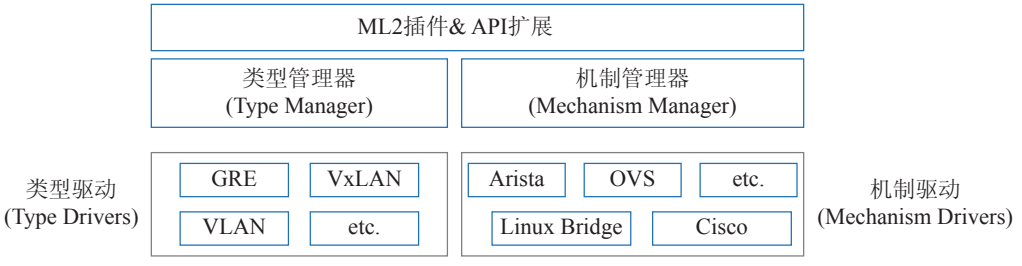


图7-19　ML2 Plugin架构

MAC地址和隧道端点信息。

虚拟机在通信前，会发送ARP请求去解析目的MAC与目的IP间的映射关系，这一过程需要发送二层的广播包，这会导致隧道上的泛洪。而传统的网络中ARP依赖于广播泛洪的原因在于没有一个集中式的控制平面，而Neutron中的数据库存有所有虚拟机MAC地址与IP地址间的映射，可以说是一个天然原生的控制平面。因此l2_population将该映射关系注入到OVS本地，在本地处理ARP广播，避免了隧道上的泛洪。

三、Neutron Agent

Agent一般专属于某个功能，用于使用网络设备或虚拟化技术来完成实际的网络配置操作，在L2/L3控制面，主要的Agent是OVS Agent和L3 Agent。

7.2.4 分布式控制面

原生Neutron的L2/L3控制面，关键是把网络节点的业务功能下沉到计算节点(分布式的虚拟化网络)，然而由于依赖于传统Linux网络协议栈的网络功能，导致耦合性高，资源消耗大。同时，控制面的RPC、DB访问机制也限制了Neutron在大规模主机站点时的能力。

因此Neutron的发展趋势，也在逐渐将精力集中在微内核，重点提供北向API，建立生态和外围，像Service Framework都交给第三方去做。

对于L2/L3的控制面，典型的分布式控制架构有OVN项目和Dragonflow项目。

一、OVN

OVN是VMware的项目，基于Open vSwitch社区，通过Plugin接入Neutron。由于是基于Open vSwitch的项目，所以和Open vSwitch由同一个团队维护(见图7-20)。

其中ovn-controller相当于Neutron中的l2/l3 agents，它运行在每一个Hypervisor上，直接通过OpenFlow+OVSDB协议在南向和ovs-vswitched交互。

由于OVN目前只支持OVSDB，所以暂时还没有分布式集群。OVN的DB本身又细分为了Northbound DB和Southbound DB，两者之间是通过守护进程OVN-Northd来联系起来的。Northbound DB收集业务数据，负责把Neutron中的数据结构(如network、subnet、port、securitygroup)转换为OVN的数据结构(如logical switches、logical ports、ACL)，而Southbound DB收集底层网络的实时数据，把logical flows和port bindings(VIF的信息，如MAC地址、接入位置等)同步给各个Local OVN Controller。最后Local

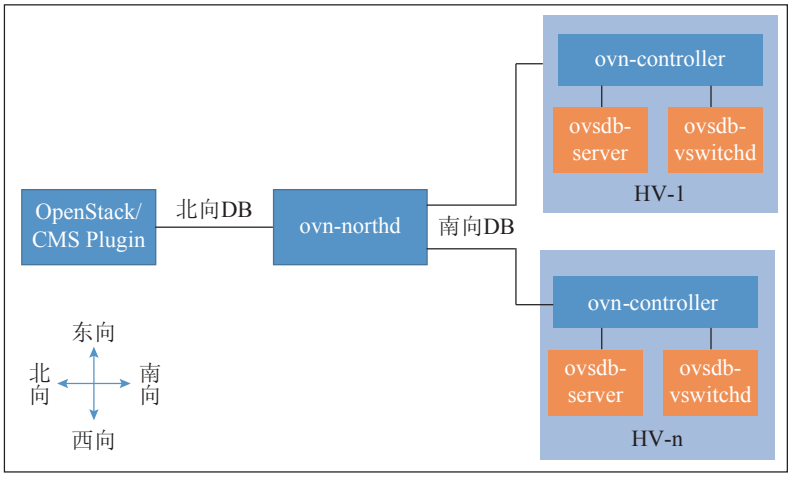


图7-20 OVN架构

OVN Controller将logical flows映射为physical flows，通过OpenFlow+OVSDB部署本地的OVS。

二、Dragonflow

Dragonflow作为Neutron的子项目之一，由华为以色列技术团队提出，Dragonflow的设计符合一个标准的Neutron扩展，通过Plugin来实现API，通过L2 Agent和L3 Agent分别实现功能。DragonFlow最初的目标是通过可插拔、无状态、轻量级的SDN控制器来实现分布式路由，它的提出是为了解决DVR中存在的一些问题，主要是DVR会造成计算节点上资源和性能的一些损耗(见图7-21)。

Dragonflow和OVN一样是全分布式的架构，并可以通过插件集成多种DB，由于是基于Phython的实现，与Neutron更易集成，代码量少，也更容易维护。目前Dragonflow已经支持的功能包括：L2交换，分布式DHCP(控制器做DHCPServer)，DVR(流表)，多种隧道封装。正在合入的功能包括分布式DB支持Reactive或Selective模式、安全组、Service Chain、分布式SNAT/DNAT等。

从设计理念上讲，Dragonflow使用的是可插入式无状态化轻量级软件分布式SDN控制器，实现了租户子网间(东-西)流量的完全分布化，避开了网络节点，减小了故障域，避免单点故障。它可以提升OpenStack Neutron L3的可扩展性、性能和可靠性。它可以支持数千个计算节点，为实现动态增长而保持控制器的无状态化，没有中央瓶颈，通过避免使用IPTables和命名空间减少计算节点开销。

7.3 硬件Underlay SDN网络

硬件Underlay SDN网络，包含云平台和硬件SDN控制器，通过控制器插件集成OpenStack Neutron，与Neutron协同完成网络的编排和控制(见图7-22)。

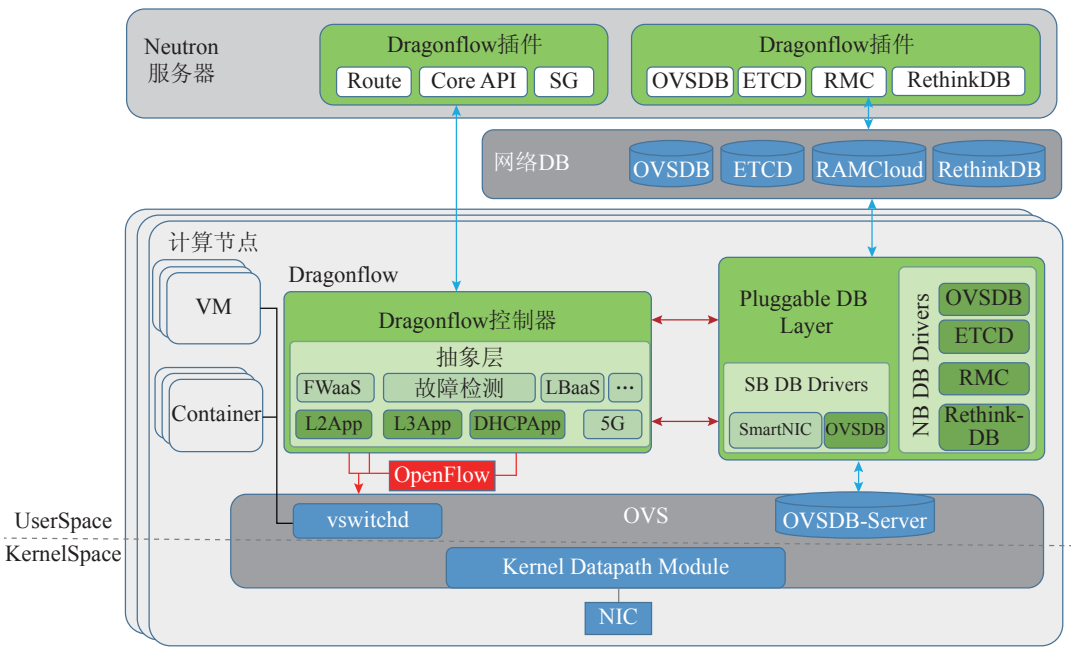


图7-21 Dragonflow架构

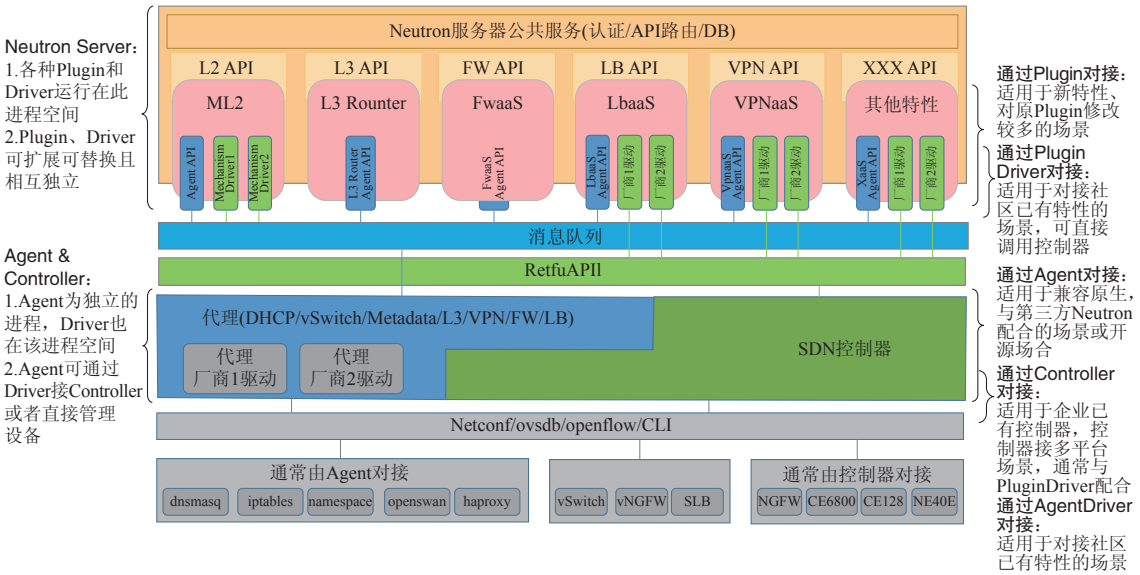


图7-22　Neutron的插件机制

如图7-22所示， Neutron框架提供了多种灵活的插件接入方式，包括Plugin、Plugin Driver、Agent和AgentDriver，甚至它们的组合也可以实现设备接入到Neutron框架，实现统一建模和业务编排。

各厂商硬件设备及控制器，可以按需接入Neutron管理硬件设备，而vSwitch虚拟网元由Neutron负责管理，实现物理和虚拟网络的解耦。这种组网环境下，可以支持提供由OVS提供VLAN，并由ToR和L3Gateway提供VxLAN的硬件VxLAN网络，也可以支持提供由OVS提供VxLAN，并由L3Gateway提供VxLAN的硬件+软件VxLAN网络。

同时，其支持由控制器通过L3/FW/LB/VPN的插件方式，与OpenStack协同，完成由硬件提供的Router/FW/LB等网络服务。

整体的逻辑架构分为以下几层(见图7-23)。

一、业务呈现/协同层

业务呈现功能主要面向数据中心用户，例如运营商与企业用户，向业务/网络管理员、租户管理员提供运维管理界面，实现服务管理，业务自动化发放，资源和服务保障等功能。业务协同层主要包括OpenStack云平台中的Nova、Neutron、Cinder等组件；通过各种组件实现对应资源的控制与管理，实现数据中心内的计算、存储、网络资源的虚拟化与资源池化，通过不同组件间交互实现各资源间的协同。

二、网络控制层

完成网络建模和网络实例化，协同虚拟与物理网络，提供网络资源池化与自动化；同时构建全网络视图，对业务流表实现集中控制与下发，这是实现SDN网络控制与转发分离的关键部件。

三、Fabric网络层

数据中心网络的基础设施，提供业务承载的高速通道。采用VxLAN技术，使用MAC-in-UDP封装来延伸L2层网络，实现业务和资源与物理位置解耦，为数据中心构建一个大二层逻辑网络，同时VxLAN 使用24bit的VNI字段标识二层网络，可支持16M的网络分段，解决了VLAN标签(4096)日益不足的缺陷。

四、服务器层

提供虚拟化服务器或Baremetal服务器接入。其中虚拟化服务器是指将一台物理服务器使用虚拟化技术虚拟成多台虚拟机和虚拟网络交换机，

虚拟机通过虚拟交换机接入到Fabric网络。方案兼容当前主流的服务器虚拟化产品：VMware、Hyper-V以及开源KVM/XEN。Baremetal服务器是OpenStack G版本支持的特性，通过将一个物理机看做一个实例，使OpenStack同服务器的硬件直接进行交互，实现OpenStack云平台对物理机的直接管理。

根据Overlay的封装/解封装的位置，我们可以划分为两种实现方式。

- 完全基于硬件实现VxLAN的SDN网络
  - 通过VxLAN网络实现大规模的二层网络、业务资源的大范围互通，计算的部署与网络物理位置无关；
  - 将TOR交换机作为VxLAN边缘交换机，实现了网络和计算的解耦，VxLAN网关性能较高；
  - OpenStack控制vSwitch实现VLAN、安全组、QoS等网络配置；
  - 基于云平台的计算和网络资源统一发放，网络业务可以独立发放。
- 基于硬件+vSwitch实现VxLAN的SDN网络；
  - 通过VxLAN网络实现大规模的二层网络、业务资源的大范围互通，计算的部署与网络物理位置无关；
  - 将vSwitch作为VxLAN边缘交换机，可以对TOR设备异构，还可利旧数据中心的大量接入交换机。同时保持了VxLAN网关的较高性能；
  - OpenStack控制vSwitch实现VLAN/VxLAN，安全组，QoS等网络配置；
  - 基于云平台的计算和网络资源统一发放，网络业务可以独立发放。

7.4 软件化L4～L7网络功能

除了vSwitch，很多厂商已经在提供纯软件的防火墙和负载均衡器等。这些L4层以上的网络设

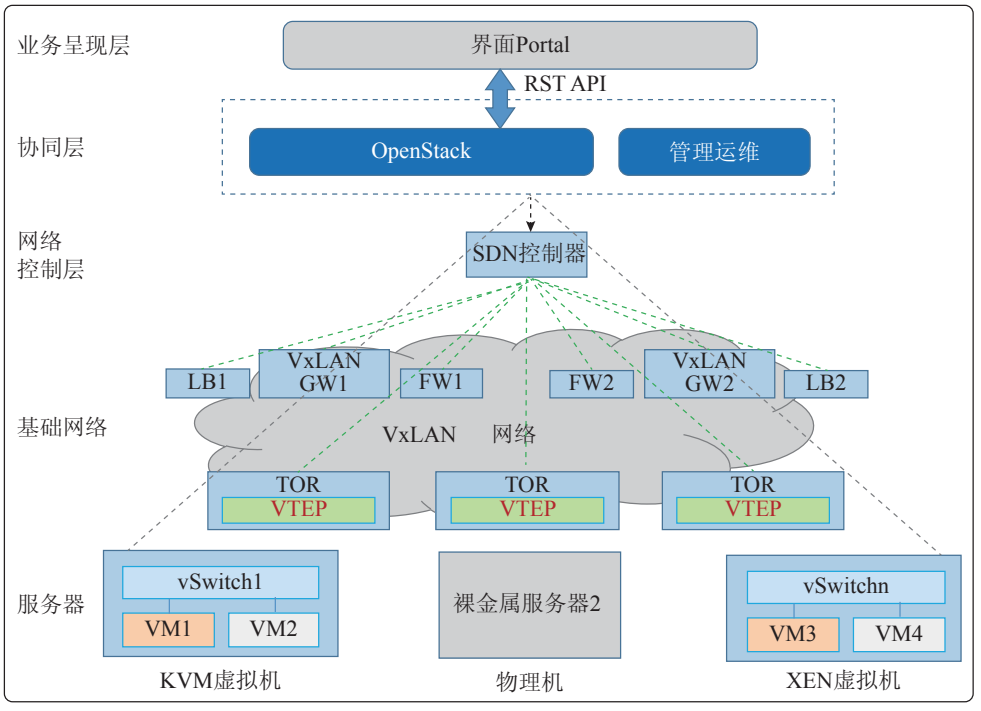


图7-23 硬件VxLAN

备本来就大多是通过软件实现的，但毕竟传统的交付形态是一个专有硬件，其采用的CPU也大多不是x86这种通用的计算平台，而且往往还有一些硬件加速引擎等。此类专用的硬件设备原来的优势是单设备性能相对较高，但最大的缺点在于多租户支持能力不足，而且缺乏弹性扩展能力。在网络中增加一个该种设备，会涉及大量网络设备的配置更改。

由于不是每个设备都可支持Overlay隧道技术，通常将隧道终结在vSwitch上，设备与Overlay网络的对接需要通过VLAN到VxLAN Mapping来实现。

随着x86 CPU计算能力的增强，L4层以上的网络设备的性能已经不是最大的问题，在云计算环境下，业务能力的弹性扩展和多租户支持能力成为关键。采用虚拟机实现的虚拟网络设备，可以方便地在需要时增加，不需要时减少，弹性很强；通过定义私有封装，可以很方便地携带租户信息，因此可以有很强的多租户支持能力(见图7-24)。

7.4.1 NFV

NFV(Network Function Virsualization网络功能虚拟化)旨在通过研究标准IT虚拟化技术，使得电信网络设备的功能能够以软件方式运行在符合行业标准的大容量通用的服务器、交换机和存储设备中去。这里的软件可以根据需要在网络中不同位置的硬件上安装和卸载，不需要安装新的硬件设备。简单地说，NFV就是想在平台层引入云计算平台，实现网元纯软件化和硬件设备解耦。其主要解决的几大问题包括如下几点。

一、提高硬件投资收益比

简单来说，采用通用硬件一方面可以大大降低成本，另一方面开通新的业务时也不需要更换硬件，只要升级相应的软件即可。

二、提高业务部署的灵活性

由于NFV使得软件和硬件解耦，也就是说运营商的业务可以灵活部署在不同机框和不同机

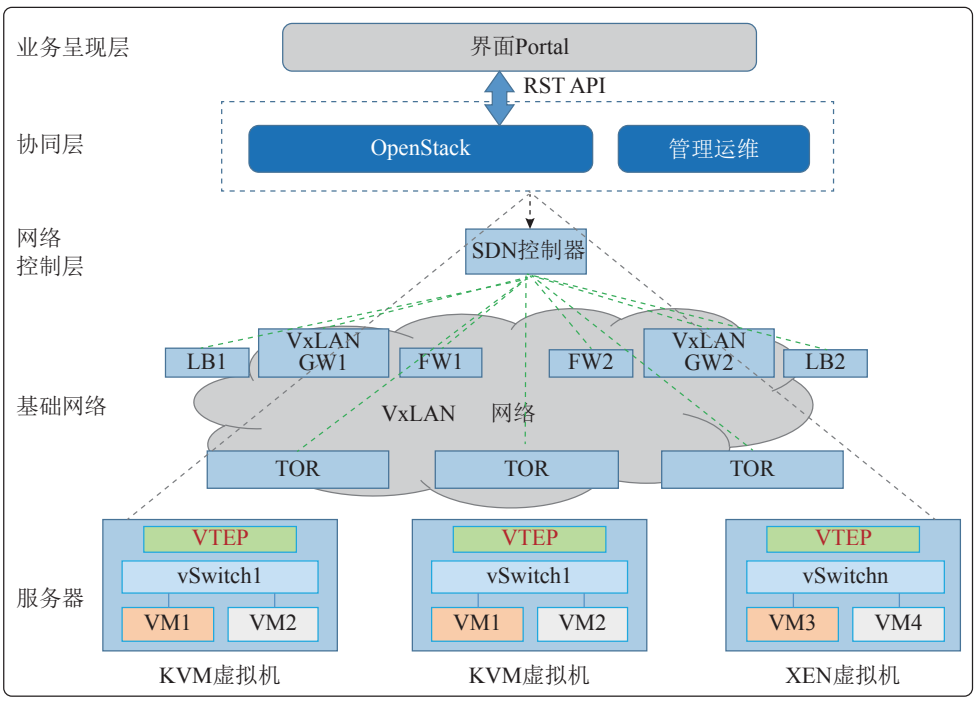


图7-24 硬件+软件VxLAN

房，在不同地域的硬件上部署，那么相比传统业务部署来说自然是极大地提高了灵活性。

三、快速部署业务

软硬件解耦带来的最大好处，就是设备商可以专注于纯软件层面的研发，在已有的软硬件部署下，新的业务研发周期会大大缩短，运营商也能尝到快速部署业务的甜头。

四、自动扩容和节能减排

由于虚拟化技术的支撑，网络智能调度资源的能力大幅提升。在业务压力增加时网络智能调度系统可以通过自动增加网路资源来缓解业务压力；在业务闲暇时可以通过自动地删减网络资源，实现节能减排。

五、降低设备商的准入门槛

硬件设备通用了，软件接口也通用了，毫无疑问设备商的准入门槛降低，也给电信服务引入了更多的竞争，这些对于运营商来说，都是最想看到的。

NFV网元纯软件化，同时又需要提供电信级5个9的高可靠性服务，因此对云计算平台需要有更高的要求。

云计算平台对于NFV云的核心价值在于以下几点。

- 由于可靠性要求，对应用的虚拟机之间的亲和性关系有着不同的约束，云计算平台支持应用的各种亲和性调度需求。
- 电信级服务意味着5个9的高可靠性，对于服务中断时间有着严苛的要求，通过硬件故障快速检测和故障快速通知技术，电信应用能够快速感知故障，并及时进行自愈机制的启动，减少业务中断时间。
- 作为NFV基础设施，提供与应用无关的高可靠性特性：HA(High Availability)特性主要面向服务器故障时虚拟机冷备机制，轻量级FT(Fault Torrance)为面向网络I/O的应用提供热备机制，以及跨数据中心的虚拟机容灾机制。
- 针对NFV进行云计算转发面的优化，使得VM到VM之间的转发性能提升到可以满足电信网元的要求。
- 云计算自动化和模板化特性可大幅度缩短电信应用和IT软件应用的上线效率。云计算的故障自动修复和自动伸缩管理能力可大幅度降低业务在线运营与故障维护的复杂度。

7.4.2 OpenStack Neutron的L4～L7控制面

除了L2/L3网络，OpenStack Neutron的插件机制，也涵盖了L4～L7的网络业务，包括VPN、Firewall、LoadBalaner(见图7-25)。

L4～L7设备，可以支持通过Neutron的L4～L7的Plugin/Agent/Driver机制直接接入OpenStack，提供对L4～L7的网络业务配置下发和管理。(如Cisco，Brocade通过 Firewall/VPN的Plugin/AgentDriver 方式接入自己的防火墙和VPN设备； A10，F5通过LBPlugin/Driver方式接入自己的负载均衡设备。)

同时，为了更灵活地统一编排，L4～L7设备也可以通过SDN控制器接入Neutron，提供对L4～L7的网络业务的统一编排，配置下发和调度管理。

对于Plugin/Driver/Agent的接入方式，各厂商的实现有所差异。

7.4.3 L4～L7的业务编排

NFV业务的编排，负责整个网络通信的生命周期管理，资源使用以及采用策略用于总资源的使用、优先级等多VNF实例的共存，其包括以下几点。

- 网络服务的实例化和网络服务实例生命周期管理，如更新，内扩/外扩，性能测定，事件收集和关联，终结。
- 分布的管理，给通信网络实例和整个运营商域的VNF实例进行NFVI资源的预留和分配。
- 在其生命周期中对通信网络的完整性和可见性的管理，通信网络实例和VNF实例之间的关系的管理。
- 通信网络实例的拓扑管理。
- 策略管理、通信网络实例和VNF实例的

执行(如NFVI资源接入控制，预留和/或分配策略，基于亲和和/或非亲和性规则及地理和/或调整规则的安置优化等)。

结合云平台和网络控制器，配置网络服务虚拟设备，并通过操控来编排与这些设备的连接，从而通过操控服务本身实现网络服务的功能。

同时，SDN控制器对网络流量转发可以进行编程控制，以所需的可用性和可扩展性等属性，或结合ServiceChain技术，无缝交付网络服务，如图7-26所示。

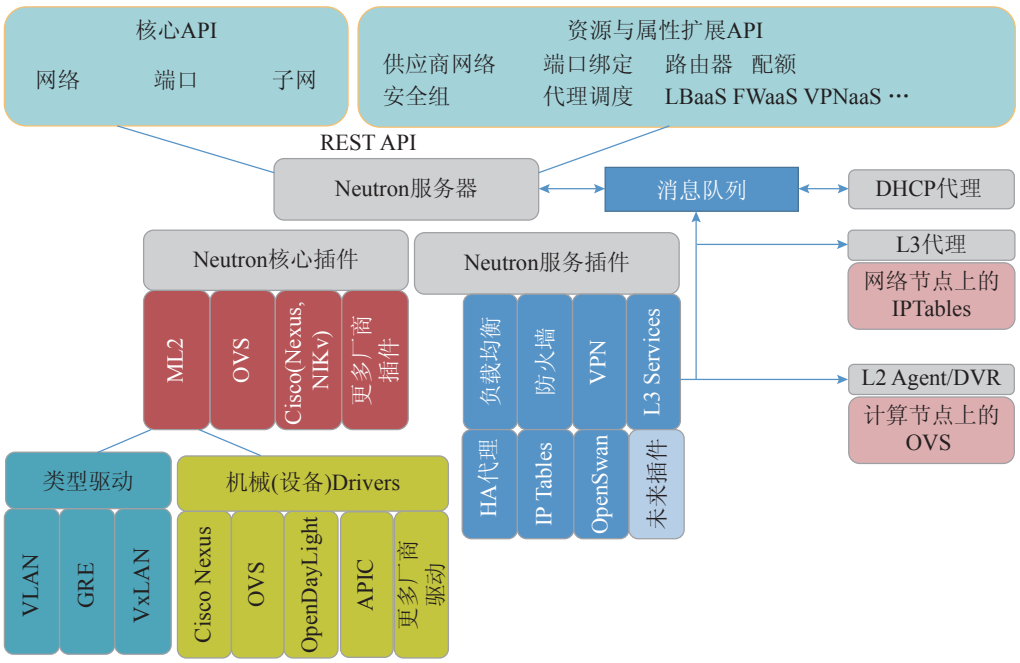


图7-25　L4～L7插件的对接

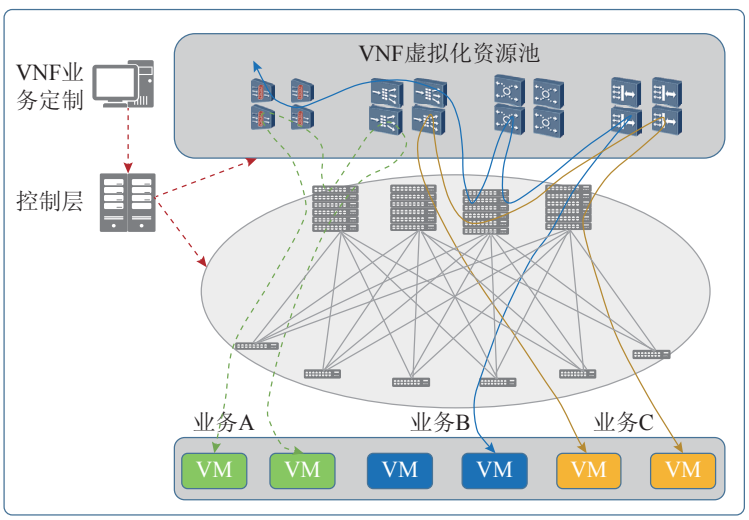


图7-26　VNF的业务编排

7.5 网络虚拟化端到端解决方案

软件Overlay SDN提供了智能、集中控制的全局优化，开放可编程、端到端QoS、网络快速修复的能力。典型的案例包括以下几个(见图7-27)。

一、基于业务和应用驱动的网络管理优化

提供了一个简化的网络自动化配置流程。系统具备统一标准的数据库，所有设备提供基于OpenFlow和Restful的接口。使用标准化的API提供自动化的服务创建过程，从而消除了使用CLI和其他人工服务的创建过程。标准OpenFlow也消除了需要分别使用厂商定制的EMS来逐个配置各个网络服务的复杂过程。

二、网络虚拟化

网络虚拟化需要能抽象出底层网络的物理拓扑，在逻辑上对网络资源进行分片或者整合，从而满足各种应用对于网络的不同需求。虚拟网络也可支持IP地址重叠，通过VxLAN/VRF进行隔离。因此，不同用户或应用可以使用自己的逻辑租户来定义不同的网络拓扑，同时又可以保证这些资源之间能够互相隔离而互不影响。

三、绿色节能的网络服务

在公有云和数据中心的云计算环境中，如何降低运营成本是一个重要的研究课题。软件Overlay技术可根据工作负荷按需分配、动态规划，不仅可以提高资源的利用率，还可以在网络负载不高的情况下选择性地关闭或者挂起部分网络设备，使其进入节电模式，达到节能环保、降低运营成本的目的。

四、动态插入安全与策略

使用ServiceChain实现自动流量导流，从而提供防火墙和入侵检测系统(IDS)服务。当前大多数的安全策略受到VLAN或接口的限制，并且使用静态的配置，无法感知具体应用上下文信息。尽管可通过使用802.1.x 进行动态策略与身份标识管理的增强，但还是无法提供足够灵活的安全策略管理能力。在虚拟化网络环境中，云平台可理解流的上下文信息(用户、时间、应用和其他外部参数)，让管理员可以配置更好颗粒度的策略并应用到交换中。

7.5.1 整体方案

一、物理网络架构设计

- 采用两层扁平化架构，实现大带宽、灵活扩展。
- Leaf层接入计算(物理服务/虚拟服务器)、存储、增值业务、管理控制节点。
- VxLAN完全由软件实现。POD区实现L2/L3 VxLAN互通，L4/L7区实现南北互联。
- VxLAN网络纯软实现，同硬件网络分离组网。

二、VxLAN Overlay网络架构

- VxLAN VTEP在虚拟化平台vSwitch上实现，东西流量/DVR通过VxLAN互通，南北流量终结在Vrouter服务器集群。
- Overlay层面网络业务由分布式控制架构控制和发放。

7.5.2 虚拟网络管理

一、网络业务模型

对于公有云或者企业数据中心，用户的业务模型一般可以抽象为图7-28所示的业务模型。

对应到逻辑网络模型可以为图7-29的逻辑模型。

二、VDC概念及规划建议

VDC(Virtual DataCenter)是指虚拟数据中心，对应到OpenStack里的Domain概念。VDC是面向最终组织的资源容器，企业用户可以通过VDC对企业内的不同应用可以使用的资源进行配额管理，包括vCPU、内存、存储、EIP、VPC、安全组、虚拟机数量等参数。

在进行VDC设计时，可根据实际情况按照部门来划分，每个部门一个VDC，彼此间资源隔离；也可按照业务来划分，将规模大、部署复杂

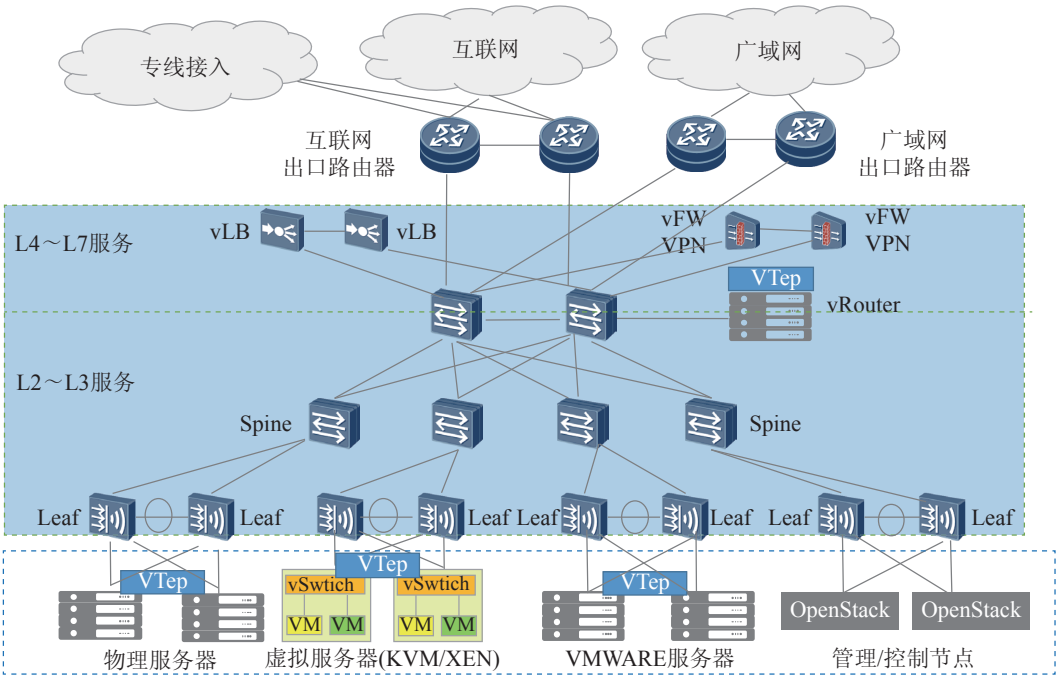


图7-27　网络虚拟化的整体方案

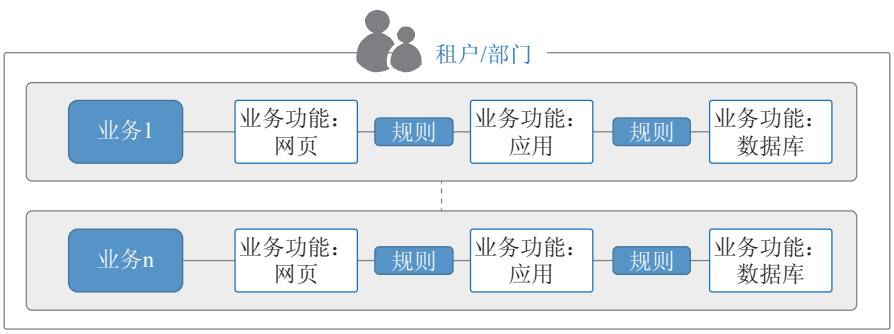


图7-28　业务模型

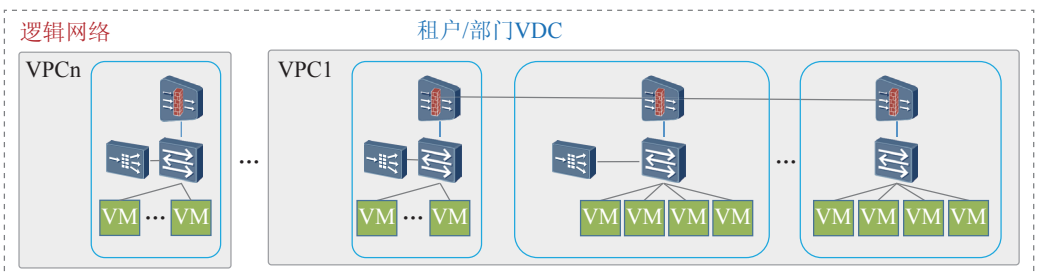


图7-29　逻辑模型

的业务规划为一个VDC。

对于公有云场景，可以给每个租户分配一个VDC，也可以多个租户共享一个VDC，只有该VDC的租户才可以管理该VDC下的虚拟机。

VDC模型如图7-30所示，并具有如下特点：

- 将资源虚拟化成VDC，VDC管理员只需要关心VDC内多少资源可用，而不用关心应用使用的资源的具体部署。
- VDC的资源可以来自一个数据中心，也可以来自多个数据中心。
- 按应用分配资源，保证应用间资源使用不冲突。

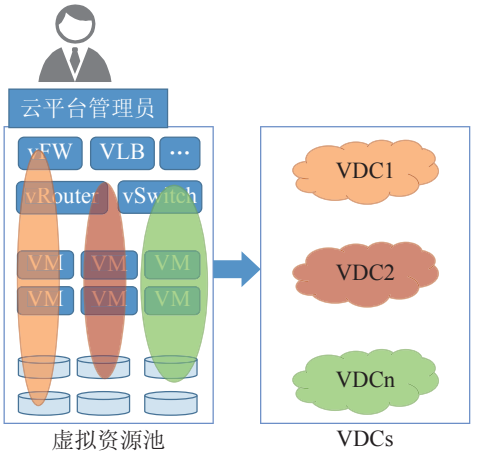


图7-30 VDC逻辑模型

- 按硬件配置划分给不同的VDC，提供不同质量的资源给对应的应用。

对于多租户公有云场景，对于每个VDC，内部网络可由租户自行规划，租户间IP地址可重叠(租户流量进入外部网络前必须做NAT转换)。但此时VDC设计需遵循：一个物理分区可部署一个或多个VDC；单个VDC内的资源必须被分配在一个物理分区内，相应的计算及存储资源需要被分配到一个AZ内。

使用VDC进行资源配额配置，可以方便地对系统内的资源进行整体的规划，提升运维能力，并达到对资源进行灵活部署的目的。

三、VPC概念及规划建议

VPC(Virtual Private Cloud)即虚拟私有云，对应OpenStack里的Project。VPC是使用VDC资源创建的一种虚拟安全域，提供安全的网络边界防护，不受限制的完整IP地址空间，以及基于VPC提供一系列增强的特性，例如：弹性IP、安全组、防火墙ACL及VPN等(见图7-31)。

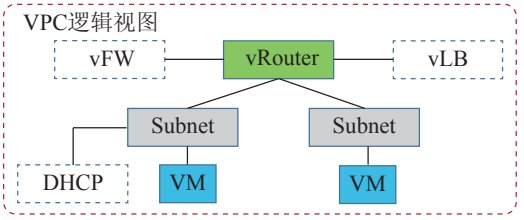


图7-31 VPC逻辑模型

对于企业用户而言，VPC的设计主要起以下作用。

- 资源隔离：VPC(Virtual Private Cloud)提供隔离的计算和网络环境，满足不同部门/业务网络隔离要求。
- 增值业务接入：每个VPC可以提供独立的虚拟防火墙、弹性IP、安全组、IPSec VPN、NAT网关等增值业务，这类业务通过防火墙组件提供，表7-1是防火墙部件提供的增值服务列表。

表7-1 防火墙部件提供的增值服务列表

业务特性	特性描述
vFW	用于基于五元组安全策略控制保护vFW的untrust域到trust域之间的业务数据流
ASPF	提供基于虚拟防火墙安全域之间的应用安全防护 (控制与数据分离，例如FTP：控制21/数据20)
EIP	弹性IP业务是指将公网IP地址和VPC内的虚拟机绑定，以实现VPC内各种资源通过固定的公网访问地址对外提供业务
DNAT/SNAT	共用公网IP，提供端口级的外网访问，只能从外网主动访问内网，用户设置公网地址与私网地址的映射关系，在防火墙上配置一对多NAT；另外部分地址重叠的VPC之间相互访问同样需要使用NAT技术将重叠地址进行映射后才能互访

(续表)

业务特性	特性描述
IPSec VPN	可以为IP网络提供安全性的协议和服务的集合，建立企业分支网络和数据中心VPC之间的安全访问通道， 实现传输加密
ACL	防火墙对进入自己的流量基于报文五元组等元素进行转发控制。

✍ 提供多种网络类型：基于VPC用户网络可以是直连网络、路由网络和内部网络(见图7-32、表7-2)。

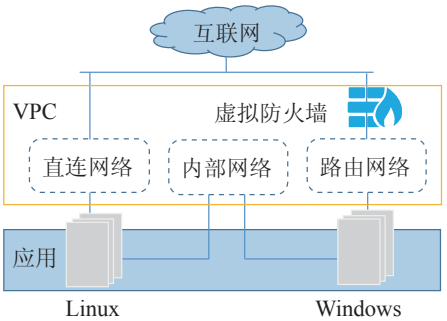


图7-32 VPC网络类型

表7-2 VPC网络类型

网络类型	特性描述
内部网络	不含网关的二层网络，可选支持IP地址管理(DHCP)，这种网络仅有二层，不提供三层访问的能力；这种网络一般用于内部使用，不需要与外部路由互通，例如产品开发过程中测试网络；内部网络VM间可以通过安全组进行隔离
路由网络	在二层网络基础上提供IP地址管理和三层网关服务，路由网络支持多种业务功能和灵活的互通能力，通常包含至少一个网段，基于VPC中的虚拟路由器提供以弹性IP或NAT的方式与公网进行通信，或者与VPC中的其他路由网络互通；该组网模型还可选防火墙、LB为业务提供ACL、弹性IP、VPN、负载均衡等增值网络服务；主要适用于业务模型复杂、内部间互访需要精细控制、对外服务需要提供高性能、高可靠性的场景
直连网络	直连网络直接与外部网络相连，其自身不包含任何网络资源，在直连网络中创建虚拟机实际获取的是外部网络中的IP地址资源；外部网络可以是公司现有网络或者公网；主要适用于用于其他对外网提供访问的业务应用或者视频点播业务；直连型网络的安全通过安全组控制或者边界防火墙的安全过滤控制

VPC的业务包括以下几点。

1. 弹性IP

弹性IP地址是一个公网IP地址，该IP地址可以与VPC内任何一个路由网络中的内部IP地址绑定。这个内部地址可以是虚拟机的IP地址、VLB的虚拟IP地址或浮动IP地址。例如，为VPC内的Web服务器绑定弹性IP后，公网用户通过访问弹性IP地址使用Web服务，如图7-33所示。

2. DNAT

当VPC内部需要提供对外服务时，公网用户发起连接请求，由防火墙上的网关接收这个连接，然后将连接转换到内部，此过程是由带有公网IP的网关替代内部服务来接收外部的连接，然后在内部做地址转换，此转换称为DNAT，主要用于内部服务对外发布(见图7-34)。

3. SNAT

内部地址单向发起请求访问公网上的服务时(如Web访问)，内部地址会主动发起连接，由防火墙上的网关对内部地址做地址转换，将内部IP地址转换为公网IP地址。这个由网关完成的地址转换称为SNAT，主要用于内部共享IP访问外部，如图7-35所示。

4. VPN

VPN业务用于在公网之上建立一条安全、稳定的通信隧道，将分布于不同地域的企业或个人连接起来，并保证通信隧道内发送和接收数据的安全性。云管理需支持IPsec和L2TP两种类型的VPN，将处于公网中的用户接入到VPC的路由网络，使用户与路由网络中的服务器互通。

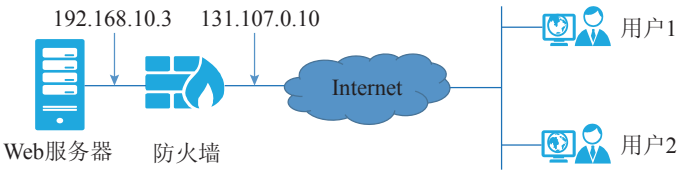


图7-33 弹性IP服务

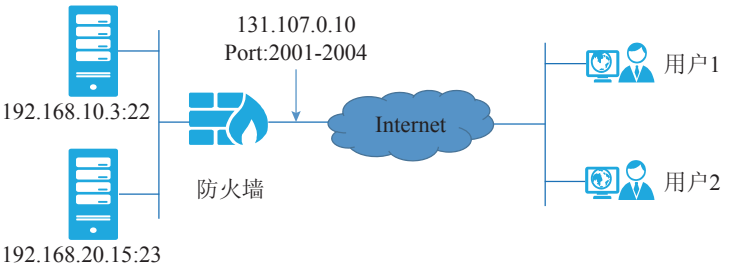


图7-34 DNAT服务

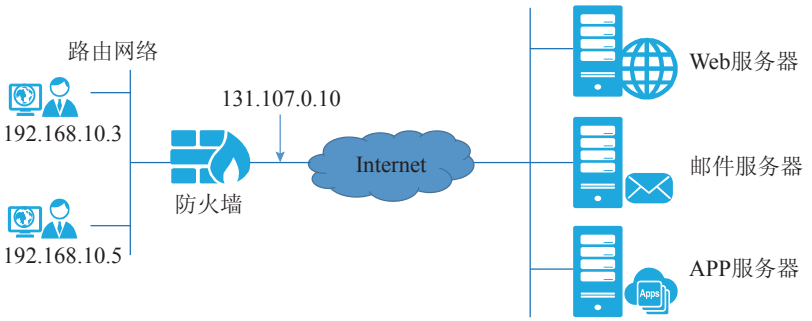


图7-35 SNAT服务

7.5.3 虚拟化环境中的网络安全

虚拟化平台的网络通信平面划分为业务平面、存储平面和管理平面，且三个平面之间是隔离的。存储平面与业务平面、管理平面间是物理隔离；管理平面与业务平面间是逻辑隔离。通过网络平面隔离保证管理平台操作不影响业务运行，最终用户不能破坏基础平台管理。

网络平面隔离如图7-36所示。

- 业务平面：为用户提供业务通道，为虚拟机虚拟网卡的通信平面，对外提供业务应用。
- 存储平面：为iSCSI存储设备提供通信平面，并为虚拟机提供存储资源，但不直接与虚拟机通信，而通过虚拟化平台转化。
- 管理平面：负责整个云计算系统的管理、业务部署、系统加载等流量的通信。

一、IP/MAC防欺骗功能设计

弹性EVS实现的一个重要网络安全功能是IP/MAC的防欺骗功能。

其功能包括：

- 截获DHCP报文，进行解析；
- 对开启IP/MAC防欺骗功能的虚拟网卡侧过来的报文进行非法DHCP报文过滤；
- 根据开启IP/MAC防欺骗功能的虚拟网卡的DHCP ACK报文生成对应IP/MAC防欺骗 DB条目，用于IP报文源地址防欺骗和ARP报文防欺骗。

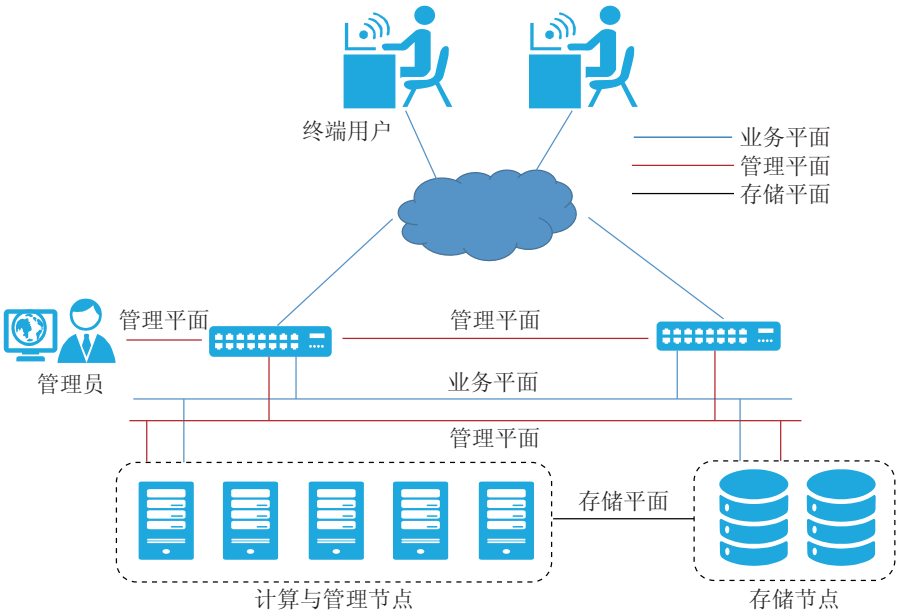


图7-36　网络平面隔离

二、VLAN隔离

通过虚拟网桥实现虚拟交换功能，虚拟网桥支持VLAN Tagging功能，实现VLAN隔离，确保虚拟机之间的安全隔离。

虚拟网桥的作用是桥接一个物理机上的虚拟机实例。虚拟机的网卡eth0、eth1等称为前端接口(front-end)。后端(back-end)接口为vif，连接到Bridge。这样，虚拟机的上下行流量将直接经过Bridge转发。Bridge根据mac地址与vif接口的映射关系做数据包转发。

Bridge支持VLAN Tagging功能，这样分布在多个物理机上的同一个虚拟机安全组的虚拟机实例可以通过VLAN Tagging对数据帧进行标识。网络中的交换机和路由器可以根据VLAN标识决定是否对数据帧路由和转发，也可以依据VLAN标识提供虚拟网络的隔离功能。

如图7-37所示，处于不同物理服务器上的虚拟机通过VLAN技术可以划分在同一个局域网内，同一个服务器上的同一个VLAN内的虚拟机之间通过虚拟交换机进行通信，而不同服务器上的同一VLAN内的虚拟机之间通过交换机进行通信，确保不同局域网的虚拟机之间的网络是隔离的，不能进行数据交换。

三、公有云、私有云的安全组

虚拟化带来的最大威胁是虚拟机间资源未完全隔离，云计算提供了同一物理机上不同虚拟机之间的资源隔离，避免虚拟机之间的数据窃取或恶意攻击，保证虚拟机的资源使用不受周边虚拟机的影响。终端用户使用虚拟机时，仅能访问属于自己的虚拟机的资源(如硬件、软件和数据)，不能访问其他虚拟机的资源，保证虚拟机隔离安全。

实现原理如图7-38所示。

用户可以根据虚拟机的属性灵活定义安全组，包括IP、MAC、VLAN、Subnet等元数据。安全组是具有同等安全要求的一组虚拟机。安全组可以由一个或多个VM，也可以由一个或多个VLAN组成。安全组的策略可以细化到VLAN甚至每台VM。每个安全组对应一个安全域，可以基于安全组定义自己的访问控制规则，实现域内和域外的安全隔离。安全组间的访问控制在Hypervisor层实现，每个host上部署一个，不占用额外的虚拟机资源，不存在安全检测引流导致的

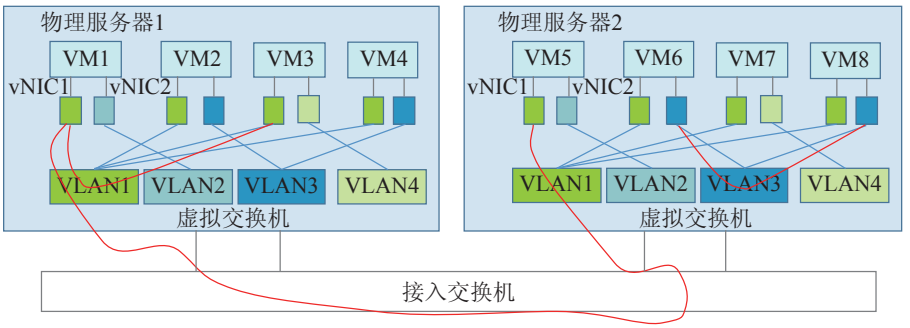


图7-37　VLAN平面隔离

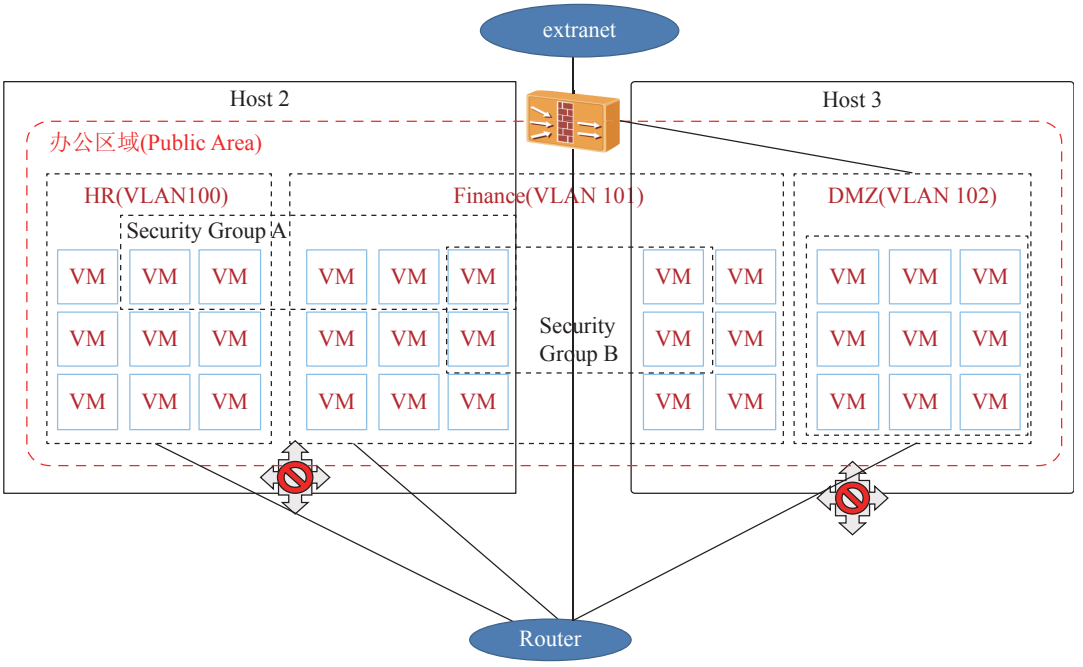


图7-38　安全组隔离

流量迂回问题。

安全网关控制器(Security Gateway Controller，SGC)是实现VM隔离功能的核心组件，它部署在Hypervisor，主要功能如下：

- 当SGC上不存在从源VM到目的VM的安全组规则时，它和目的端SGC进行安全组规则协商，以决定是否允许VM之间的通信；
- 当SGC本地的安全组成员表、安全组规则表发生变化时，需要通知相关SGC做相应的更新；
- SGC接收安全组规则协商请求消息，根据本地的安全组成员表和安全组规则表，动态生成安全组会话表；
- SGC接收安全组成员、安全组规则表(增、删、改)通知消息，以及VM迁移事件，并做相应处理；
- SGC接收Hypervisor发来的VM在线、离线事件，并做相应处理。

7.5.4 跨站点、大规模的网络虚拟化方案

云的核心动力，将隔离的资源、纵向的烟囱结构，通过资源共享、资源池化，提升资源的利用率，同时在网络层面，通过对各个资源孤岛的网络互连和拉通，满足云的弹性、伸缩的需求，从而给客户带来敏捷、简单、资源整合，以及业务的自动化和迅速上线的价值。

然而，在现实中，其仍然存在各种各种的问题。

首先，是多DC带来的问题，如何拉通不同DC的件的资源池，统一地编排和自动化，同时资源可以在网络上互通。由于各个DC的虚拟化平台的版本可能不同，以及独自运维、独自故障隔离的需求，在采用OpenStack云平台的场景下，既需要保持OpenStack的开放生态，而又不将范围扩展到所有DC，可以独立自治。

其次，即使在一个站点内，当站点资源无法满足业务需求，资源急迫需求扩容的时候，是机械地在主机层面的扩容，还是可以通过与集成的云管理平台和资源池，实现可插拔的扩容和升级。

最后，在跨DC的情况下，不同的场景有着不同的需求，例如需要跨DC的二层互通，这就要求跨DC的管理业务面完全拉通。对于虚拟机的端口网络，计算资源发放到哪里，网络资源就跟随到哪里，但是对于租户的网络配置，则需要全局拉通，包括统一的IP/MAC地址管理，租户地址隔离，统一的安全QoS策略，以及弹性漂移时的策略随动。

用传统的OpenStack来解决这些问题的时候，存在的障碍包括以下几点。

- 单个OpenStack难以管理大规模集群，特别是由于Neutron的限制。
- 故障隔离，RPC域的范围隔离。
- 单个OpenStack所带来的运维，故障定位恢复，升级维护的复杂性和难度。
- 难以支持多版本共存，例如juno/icehouse/vcenter...
- 通过新的API接入点，全局管理配置多个OpenStack，问题是API的生态被破坏，二层拉通困难，同时，客户的管理平台对接将变得不标准化。

基于这些问题和驱动力，华为在社区提出了OpenStack级联的架构和方案，社区正式的名称叫Tricircle(见图7-39)。

上层级联层暴露标准OpenStackAPI，上层级联层通过调用被级联层OpenStack的API，管理被级

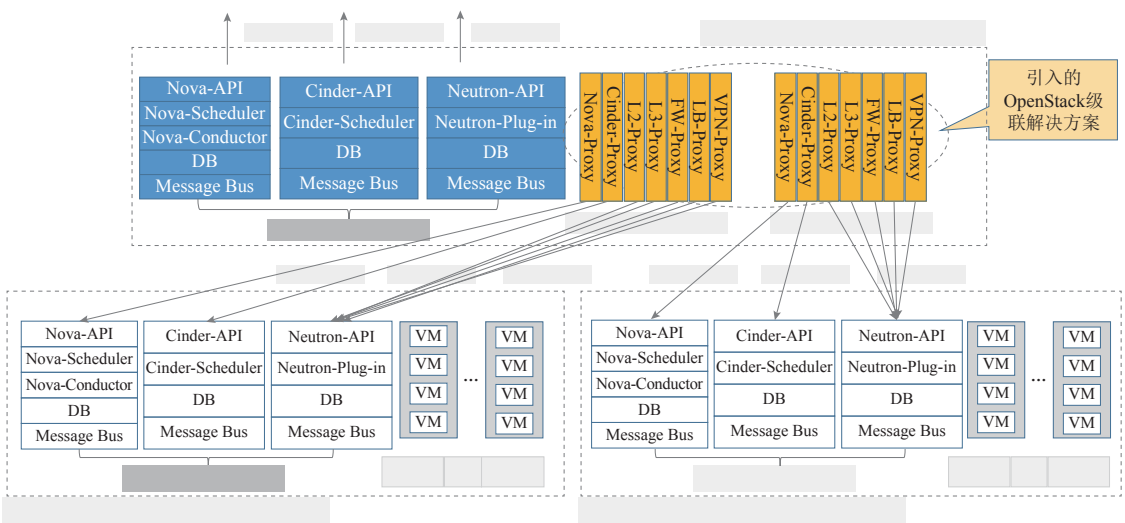


图7-39 OpenStack级联架构

联的下层OpenStack。由于Neutron本身是一个插件的架构，所以这种扩展也是一个标准的扩展。

以Neutron的二层为例，级联层的L2-Neutron-Proxy 作用类似一个普通的L2-Agent，以OVS作为后端。但实际上，通过在Proxy中扩展实现，由原来的针对OVS br-int，br-tun的流表配置，改变为调用被级联层的Neutron模块，从而同步下发L2配置。

级联方案带来的价值包括以下几点。

- 这是一个开放的架构，拥有两级标准OpenStack API，以及完全开放的生态。
- 即插即用的快速集成：由于被级联的OpenStack只需要提供标准OpenStack API即可，因此架构具备即插即用、快速集成第三方基础设施的能力，采用标准OpenStack API可实现多厂家异构快速集成。
- 故障隔离的高可靠系统：单个被级联的OpenStack站点可以达到管理1024台服务器的规模，系统故障的影响范围局限在1024的小规模下，不会影响其他的被级联的OpenStack的资源池。同时，即使级联OpenStack出现故障，被级联OpenStack仍然可管理。系统故障容忍度高。
- 升级隔离：单个被级联OpenStack升级不影响其他系统，系统天然具备多版本并存能力，不会因为局部升级导致整个百万系统规模的升级。
- 具备水平扩展的能力，在被级联的OpenStack站点内的服务器可以大规模扩展被级联OpenStack站点的数量也可以大规模水平扩展，这样的OpenStack云平台可以从极小的几台服务器到百万级别服务器的规模。

7.5.5 物理和虚拟化资源的统一管控

云管理平台，通过对各种物理资源、虚拟化资源数据统一建模，将资源以用户可见的资源池形式提供给上层应用，在接入不同的物理设备和虚拟化资源环境时，上层应用不感知。

统一资源管理，支持发现其管辖范围内的物理设备(包括机框、服务器、存储设备、交换机、防火墙和负载均衡器等)以及它们的组网关系。支持将这些物理设备进行池化和集中管理，提供给上层应用管理使用，实现资源高效共享。

虚拟化资源管理可以统一管理不同厂商的虚拟化平台系统，如OpenStack、华为FusionSphere、VMware vSphere、微软Hyper-V、思杰Xen Server等虚拟化平台，提供不同资源的生命周期管理功能，包括虚拟机资源、虚拟网络资源、虚拟存储资源管理等。

通过资源池管理，其可提高基础设施资源的利用率和灵活性，提供统一的虚拟化资源管理能力，对上层应用发放屏蔽差异，实现虚拟资源集中管理，提升管理效率，降低运维成本。

采用南向插件机制，使云管理可以快速、便捷、可定制地实现不同硬件和虚拟化系统的对接。

网络资源基础管理， 提供对路由器、交换机、负载均衡器、防火墙、IP地址、虚拟网络设备等网络设备和资源的查询和配置管理，网络资源包括：资源编号、对应网络设备的基本信息，网络设备的管理和配置接口信息。将多个网络资源整合为一个整体，对外提供统一的网络资源分配和集中式管理。网络资源池应包含下面信息：网络资源池编号、网络资源类型(比如IP地址、交换机、路由器、负载均衡器、防火墙等)、网络资源池组成信息、网络资源池容量信息、资源池操作方式、资源池访问接口、系统域ID等。

典型的云管理平台，对网络资源能力包括以下几点。

- 支持对网络设备的自动发现。
- 支持设备拓扑图。
- 支持对网络设备(包括虚拟网络设备)的配置和管理，包括网络带宽容量、VLAN等资源的配置、查询、导出功能，支持通过全网资源统计，观测全网网络资源使用的状况。
- 支持网络资源(池)的容量管理，包括总容量、已使用的资源容量、可用资源容量等信息的管理和查询，对网络设备实时监测。
- 提供对网络资源(池)的生命周期管理(包括创建、修改、查询和删除)。

- 提供对网络资源(池)的操作和配置接口。
- 网络设备管理，查看交换机信息，查看交换机的名称，管理IP地址、型号、类型和状态等信息；查看交换机端口连接状态，查看交换机每个端口的编号、状态、发送速率、接收速率、发送丢包率、接收丢包率、发送错误率和接收错误率等信息；网络配置管理，对系统网络进行配置和管理，包括外部网络、组织网络和服务器BMC IP池等。

云管理平台对虚拟化网络资源、VPC业务的管理包括子网管理、VLAN池管理、VPC管理和虚拟防火墙。云平台的子网管理，支持子网下虚拟机的二层隔离。当组网模式采用二层、三层时均可根据用户需要配置VLAN池，在组网模式为三层组网时添加的VLAN池只用于隔离二层网络。VPC为应用发放提供了一个独占并且完全隔离的网络容器，可以在VPC内添加虚拟防火墙和各种类型的网络。

7.5.6 现阶段的网络云化

与计算虚拟化相比，网络虚拟化以及网络云化的历史并不算久，世界各地大规模商用的案例还比较少。在现阶段，以虚拟交换机、分布式虚拟交换机、虚拟路由器、虚拟防火墙、虚拟负载均衡器为代表的软件Overlay SDN方案，对现有网络硬件设备基本没有改动，而是通过服务器虚拟化层运行的虚拟化与云计算软件完成网络虚拟化功能。这种方案在公有云和数据中心私有云中具有成本低、推广阻力小、对网络硬件要求低等特点，会随着公有云和私有云的建设而变得普遍化。

第 8 章

无边界计算的混合云

8.1 混合云的驱动力与背景

业界在云计算的应用上，已经从最初简单的计算虚拟化，延展到了整个数据中心的云化，将数据中心内的计算、网络和存储资源都采用软件定义的模式实现自动化、规模化的管理。拥有多个数据中心的大型企业，则在多个云数据中心之间实现了多AZ拉通的资源共享模式。如今，已经有越来越多的企业把不同提供商的云拉通来使用，以解决不同的业务问题。

从传统IT向云演进的过程中，诞生了大量不同的云技术(见图8-1)。最基本的分类包括传统IT环境、企业自建的私有云、服务商面向特定企业建立的托管云和服务商面向大众建立的公有云。每一种生产环境又存在大大小小的不同的技术选择和特性选择，构筑了今天异常复杂的应用部署关系。从中短期来看，不同的企业、不同的应用选择不同的执行环境有其内在的原因，不能简单地选择其一而抛弃其他，然而这又导致了大量企业在这个过程中要同时面对多种不同类型的云，如何能管理不同类型的云并能够做大限度地实现这些云间的资源互助便是混合云产生的动因。

8.1.1 混合云对企业的商业价值

一、混合云是解决企业向云平滑演进的有效手段

企业都存在大量的历史遗留资产，包括软件系统和硬件资源，这些资产让企业可以正常运作，稳定且变更缓慢。另一方面，企业为了应对越来越加速的新业务上线挑战，不得不引入新的技术和新的面向云的基础设施架构，特别是非核心数据系统的前端应用，只有采用最新的云化技术才能跟得上竞争的节奏。反过来说，承载核心数据系统的传统基础设施和软件很难一蹴而就地迁移，通过混合云来打通传统IT和云应用之间的网络连接，让两者各司其职，演进过程逐步将传统系统替换或者迁移到云系统，这是解决企业向云平滑演进的有效手段。

二、混合云是解决企业安全使用云技术的有效方式

公有云无论从规模成本还是使用的便捷性方面都有着无可比拟的优势，对于企业而言，未来必将有大量的应用会部署到公有云上运行。但很多企业都会存在一些应用和数据在安全方面的考虑，担心公有云上的数据存在泄密的可能。而这也恰恰是公有云难以克服的短板，采用复杂的技术来提升公有云的数据安全性，总是不如把重要数据存储在物理隔离的安全区域，然后通过暴露特定的、可控的接口供外部应用访问。这是混合云会长期存在的一个重要原因，通过混合云来拉通私有云和公有云的统一管理，按需将不同安全等级的应用部署到不同的云中，两者通过对外可控的API来相互访问。

三、混合云可以帮助企业以最佳成本协同云资源

企业的基础设施资源往往不能按照最极端的情况来规划，而一旦出现某种特殊的峰值场景，

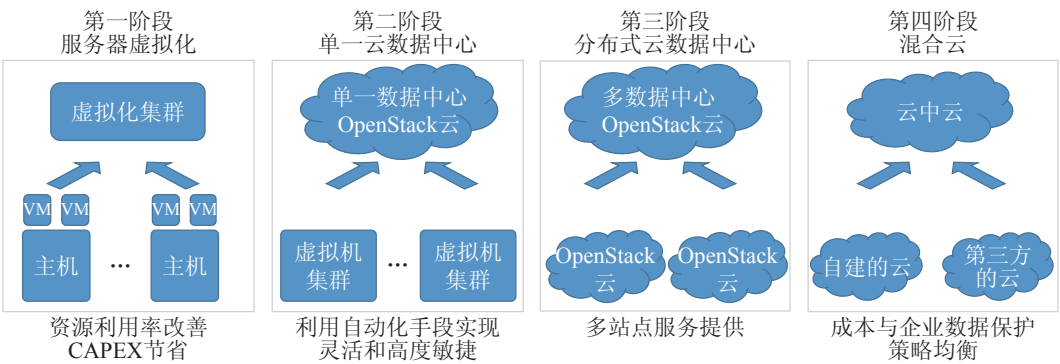


图8-1 云计算发展历程

往往会导致部分业务不能正常服务或者降低服务等级。规划和增加硬件资源往往是一个漫长的过程，特别是针对峰值带宽瓶颈导致的问题，不仅仅是简单增加数据中心的设备就能解决问题，还需要考虑从最终用户到服务端的整体带宽如何调整，这些涉及工程改造的工作会消耗大量的资金和时间，而通过混合云来临时借调外部资源却是一种高效而且廉价的解决方案。

四、混合云可以防止企业被单一的公有云Lock-in

关于企业未来如何使用基础设施资源，从云计算的角度，常常会采用发电站的例子来做类比，认为企业使用公有云的资源就好比今天我们使用电的资源一样，由集中的发电站来提供电，企业按需获取即可。但实际情况往往比这个类比要复杂得多，企业使用公有云的资源并不能像使用电一样灵活，最大的问题是一旦企业选择了某个公有云的服务商，往往就意味着后续的业务将绑定到这个服务商，很难切换到另一家服务商。混合云是解除这个绑定的有效手段，企业可以通过混合云来管理多个不同服务商的公有云资源，可以实现灵活的切换和按需组合使用，防止被一家服务商Lock-in。

8.1.2 混合云的典型业务用例

一、业务跨云部署和伸缩

企业A是一家航空公司，其服务的客户遍及全球，长期以来，所有的业务都是远程连接到其位于德国的总部来进行处理，总部的业务运行在自建的一个私有云数据中心之上，随着业务量的增加，总部的处理能力和接入带宽明显成为了瓶颈，基于这样的现状，企业A使用了混合云的解决方案，通过混合云统一管理总部的私有云和分布在全球各地的两家知名的公有云，将业务流程中的前端处理分散到全球几个不同的公有云地点进行处理，前端服务的部署位置和其所服务的最终用户尽量接近，前端处理完成后，只需要少量的前后端交互访问即可完成整个业务处理。通过这样的混合云跨云协同部署，大幅提升了系统的服务能力和用户体验。混合云带来的另一好处是，企业A可以按照更贴近实际需要的方式购买硬件资源，业务量低的时候，可以依旧采用传统模式集中到私有云上处理，当某个区域的业务量上升时，可临时租用相应的公有云资源，业务量下降后即可以释放相应的资源(见图8-2)。

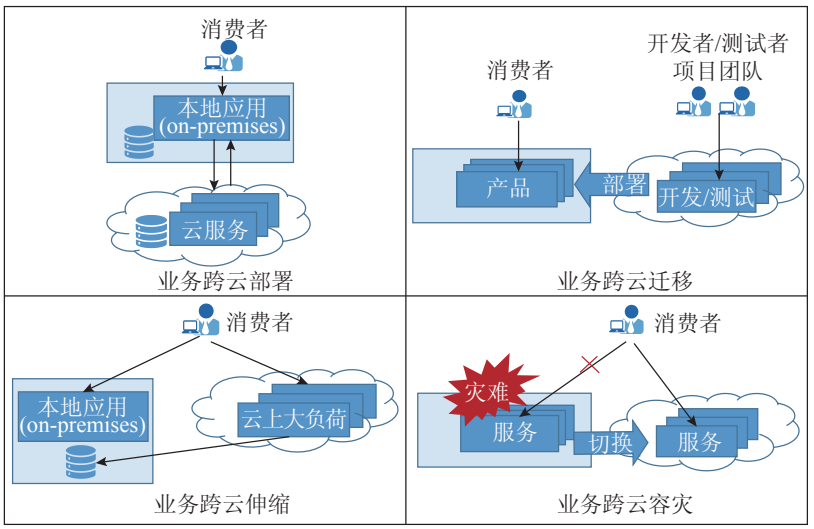


图8-2 混合云的典型业务用例

二、业务跨云复制和迁移

企业V是一家电商公司，其业务在中国开展多年，已经非常成功，企业V的业务在中国运行在三个自建的私有云数据中心之上。目前，企业V正在大规模将业务拓展到全球海外市场，为了满足较好的用户体验，也为了满足部分国家的数据安全法规要求，企业V需要将业务从中国复制到其他相关国家和区域。如果采用类似中国区的做法，到各个国家和区域构建自己的私有云，成本和时间开销都会很大。企业V选择了混合云的解决方案，直接借用海外知名地区和国家的公有云资源，通过混合云来拉通管理自己的私有云资源和第三方的公有云资源。业务统一开发和测试，然后部署到不同的私有云和公有云上，为不同的地区提供服务。企业还可以按需通过混合云提供的能力将特定地区的业务从公有云迁移到私有云，或者从一个公有云迁移到另一个公有云。这种方式给企业带来了最大的灵活性和后续的扩展能力。

三、业务跨云灾备

企业N是一家小公司，公司发展多年，已经建立了大量的内部IT和业务系统，这些系统运转良好，但一直都只在一个单独的数据中心运转。随着公司的规模扩大，业务系统的容灾能力提上了日程，如果按照两地三中心的方式构建容灾能力，对于企业N而言成本太高，不可承受。企业N最终选择了通过混合云的方式来构建业务的容灾能力，通过选择合适的公有云来作为灾备点，一方面可以降低成本，另外地点选择也比较灵活。通过混合云把企业的私有云和第三方公有云拉通管理，把主业务部署到私有云，把备用业务部署到公有云上，通过最低的成本就实现了跨站点的容灾能力，而且灾难恢复的时间还能够控制在很短的时间内。

8.2 典型的混合云架构模式

8.2.1 简单的三层网络互通

要想实现私有云和公有云的融合使用，最基本也是最简单的方式就是通过VPN或者专线打通私有云和公有云之间的三层网络。多数公有云都提供了一种叫做虚拟私有云(Virtual Private Cloud)的特性，这个能力允许企业在公有云上创建自己专属的虚拟网络，VPC内的网络和其他网络完全隔离，包括独立的子网、独立的地址池、独立的路由配置和独立的网关出口。通过VPC的出口网关配置，企业可以在公有云内的虚拟网络和企业本地数据中心的网络之间创建虚拟专有网络(Virtual Private Network)连接，使两者可以三层联通起来。

虚拟专有网络有三种典型的连接形式，一种是采用硬件专线连接，这种方式带宽大且时延有保证，但对于跨云互通带宽消耗不大的情况下成本会很高。另外，这种方式涉及对传输网络的硬件配置，往往需要以工单的形式单独申请，申请周期较长，不能完全自动化处理，而且对企业的接入地点有要求，只有公有云事先已经部署专线网络的地域才可以选择。另一种方式是采用硬件的VPN设备进行连接，这种方式就是公有云广泛提供的VPNaaS服务。企业创建VPC后，再调用公有云的VPNaaS服务创建VPN网关，关联到指定的VPC，然后通过配置典型的VPN协议，例如IPSec VPN，打通公有云的虚拟VPN网关和企业本地的VPN网关，这里的虚拟VPN网关是由公有云运营者通过硬件设备或者软件虚拟的网络设备来提供的。还有一种方式是由企业自己开发或者从第三方购买的软件VPN设备来提供，企业在自己的VPC内创建一个或者多个以虚拟机运行的VPN设备，通过路由将VPC内需要路由到企业本地网络的网段配置到软件VPN设备，软件VPN设备直接通过Internet网关连接到企业本地网络的VPN网关，然后通过企业自定义的VPN设置配通两端的VPN链路，这种方式相对于前面两种方式在性能上肯定是最差的，但这种方式的最大优势是灵活定制能力，企业可以更加自主地选择专属的协议来配置两端的VPN链路。

采用简单的三层网络互通的模式来混合私有云和公有云最大的优势是方案简单，适用范围广

泛。但其缺点是没有真正做到两端云的资源拉通使用，企业需要独立采用不同的模型来管理两朵独立的云，还需要规划和配置好两边的地址和路由信息，对于单一的应用而言，这个工作可以算作一次性的工作，但对于存在很多不断持续演讲的应用的企业而言，手工维护两个或者多个云的账户信息、镜像信息、虚拟机信息、数据信息、网络信息、路由配置等信息是一项繁重的工作。

8.2.2 云中介使能多云管理

对于经常需要使用多个云来部署应用的企业而言，通过云中介(Cloud Broker)来管理多个不同的云是一种更进一步的混合云使用模式。Cloud Broker提供统一的API和Console界面，底层通过适配不同云的API接口实现资源的统一管理。

一套云中介可以同时挂接多个云的适配模块，不同的云中介功能差异会比较大，首先是云中介支持的底层云的类型和数量，例如同时管理AWS、Azure、OpenStack公有云、Google Cloud Computing、OpenStack私有云、VMware私有云。其次，云中介所提供的服务能力会有差异，简单的云中介只提供虚拟机的统一管理功能，有些云中介会提供虚拟网络的管理、虚拟卷的管理，功能更全面的云中介会提供LB、VPN、对象存储、数据库等更高级的服务。

由于不同的云的API能力存在差异，而且往往越是高级的功能差异会越大，一般而言，虚拟机和卷的管理比较接近，但网络和其他增值服务在不同云上会存在较大的差异。采用API适配模式来实现Broker的最大限制也是这个原因导致的，很多功能在不同的云上很难统一成一套模型，所支持的云的类型越多，提供的服务越多，体系出来的不一致性就越大。最终的效果往往让使用者非常头疼，云中介本意是为了提供一套统一的模型和API来屏蔽各个云的差异，结果新定义的API不能完全屏蔽底层云的差异，不得不让用户感知到这些差异，通过不同的API或者不同的参数来调用底层不同云的功能，最终用户需要同时学习云中间的概念和底层云的原生概念。

8.2.3 云网关使能多云资源无缝共享

一种更为先进的混合云技术就是利用云网关来拉通不同云的资源，形成一种逻辑上可以共享使用的混合云资源。前面介绍的简单的云中介方式存在高级服务无法统一模型，导致用户不得不使用不同的模型来管理不同的云。采用云网关来配合云中介则不会存在这样的问题。通过注入到各个不同底层云内部的云网关抽象统一各个底层云最基本的资源接口，最后把不同的底层云转化成混合云云中介下面的不同的资源池，云服务通过统一的云中介模型提供。这种模式能做到租户业务可以平滑部署到完全不同的几种云上，就好似同一个云内部的不同的资源池。如果业务时延上允许，也可以轻易实现跨云的弹性伸缩、跨云迁移、跨云容灾等特性。

8.3 基于OpenStack级联的开放异构混合云

为了统一不同类型的云的管理，势必需要寻找合适的模型来统一不同云的API接口，重新定义一套私有的混合云API是业界传统的做法，这种方式对用户而言，会增加学习一种新的API的成本。一种更为优异的做法是选择一种现有的标准的云接口为基线，OpenStack的API很适合来作为这样的基线。首先，OpenStack是一种被广泛接受的云类型，功能完善；其次，OpenStack API的定义完全是开放的社区运作模式，容易被客户接受，新的特性也可以通过社区讨论以公平的方式进行添加。采用标准OpenStack API来作为混合云模型的最大好处是生态对接能力，也就是说只要是基于OpenStack之上开发的管理软件，就可以轻松对接到混合云上，也就可以管理AWS、Azure、VMware等大量的异构云了。

OpenStack级联是一个开放的社区项目，项目名称是Tricircle。基于OpenStack级联的混合云采用的是云中介加上云网关的架构模式，这个

OpenStack级联就是云中介的核心服务，它起到了拉通底层各个不同云的资源池的作用。云网关则是注入到底层不同的云中来拉通各个云的资源，包括AWS公有云、Azure公有云、VMware私有云、第三方OpenStack公有云、华为OpenStack公有云和私有云等。

云中介内部是标准的OpenStack模块，包括负责计算的Nova，负责存储的Cinder，负责网络的Neutron，负责资源编排的Heat，负责认证的Keystone等。这里的计算、存储、网络虽然完全兼容OpenStack的标准API，但云中介内部其实并没有任何真实的资源，只是通过简单的调度算法找到合适的Proxy，通过Proxy找到合适的底层云来进行真正的资源分配，资源分配完成后，云中介要负责登记注册这些资源信息，为后续的跨云网络互通、跨云迁移和跨云容灾提供管理(见图8-3、图8-4)。

云网关是注入到底层云内部的模块，它以虚拟机的形态运行在AWS、Azure、VMware等不同的云内。它的内部主要由四个独立的网关组成，一个叫做Jacket，其作用主要是把底层云的核心资源类API模型转换成OpenStack模型来进行统一的资源申请，例如针对AWS云的就有一个AWS Jacket，针对Azure云的，就有一个AZURE Jacket。第二个网关是V2V GW，它的作用是提供云间虚拟机和卷相互迁移的网关，通过V2V GW来实现跨不同Hypervisor和不同底层存储的相互迁移能力。第三个网关是Border GW，它的作用是提供跨云互联的虚拟网络能力，通过Border GW来创建一个跨多个不同云的逻辑网络，租户

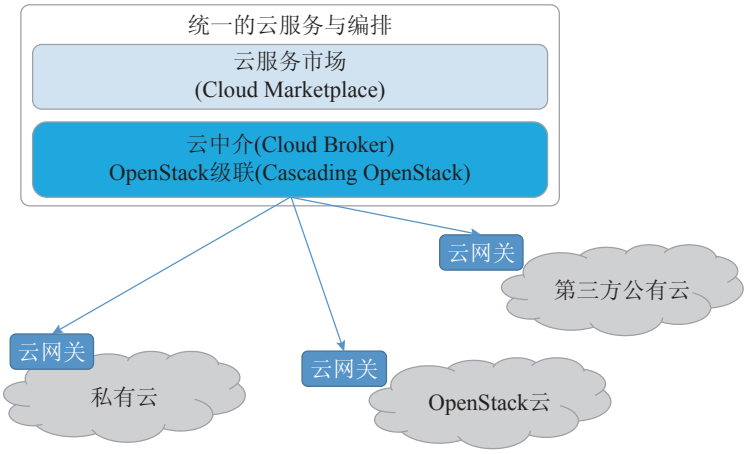


图8-3　基于OpenStack级联混合云架构

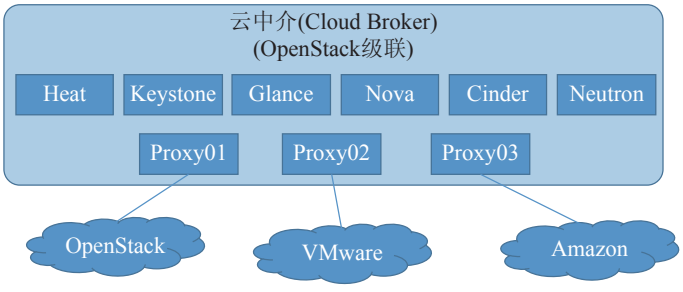


图8-4　云中介(Cloud Brocker)架构

就好似管理同一个网络那样简便。第四个网关是Storage GW，它的作用是提供跨云容灾的能力，通过Storage GW来实现数据卷在两个云之间进行增量备份，当一个云发生故障后，数据卷可以直接从另一个云恢复出来。这些网关能力可以根据企业对混合云的实际需求来进行按需部署(见图8-5)。

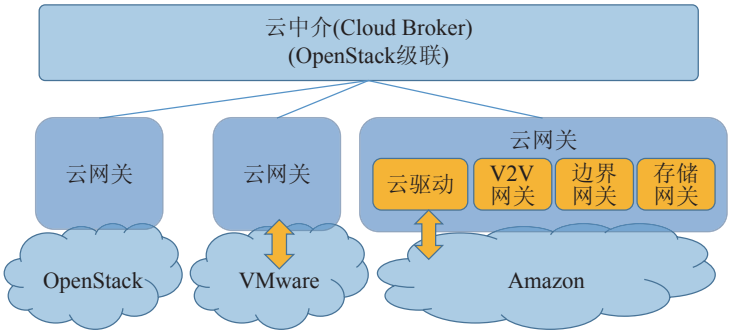


图8-5　云网关架构

第 9 章

PaaS应用开发平台

9.1 PaaS简介

9.1.1 PaaS概述

PaaS是云计算中重要的一类服务，为用户提供应用的全生命周期管理和相关的资源服务。通过PaaS，用户可以完成应用的构建、部署、运维管理，而不需要自己去搭建计算环境，如安装服务器、操作系统、中间件和数据库等。

云计算中的IaaS系统提供给用户的是虚拟机资源，PaaS提供的是一个无服务器(Serverless)的计算环境， 用户只需专注于应用的开发，PaaS负责应用的部署和运维，实现应用的弹性伸缩和高可用等功能。

9.1.2 传统PaaS构架和功能

传统的PaaS系统系统主要由管理、计算和服务三个部分组成。管理部分主要负责应用部署、运维监控、认证授权等。应用实际运行在计算节点上，计算节点提供应用所需要的运行环境，包含语言环境和应用框架等，一般采用Cgroup和Namespace为应用提供资源隔离和限制，也有PaaS系统采用沙箱(Sandbox)机制来隔离应用。服务节点通过代理或接口为应用提供数据库、缓存和存储等服务。

图9-1描述了传统的PaaS架构，其具有如下的功能。

(1) 应用部署：PaaS中有代码仓库(SVN/Git)或者应用仓库，这些用来保存用户上传的代码或者编译后的应用，系统根据应用的开发语言，将其和所依赖的中间件和框架打包，按照用户设置的应用提供所需的CPU、内存和磁盘资源，系统的调度模块根据调度算法选出合适的计算节点，该节点上的管理程序将应用下载到本地后启动。

(2) 应用日志：用来查看用户的应用日志信息，方便测试和调试(Debug)。

(3) 应用伸缩：用户可以手工地增减应用的实例数量，以应对负载的变化。有的PaaS提供应用的自动伸缩功能，可以根据用户需要设置CPU/内存的负载阈值或者根据应用访问量自动地增减应用实例数量。为了实现应用的自由伸缩，要求应用必须是无状态应用，Session信息、数据库都要放在服务节点上的资源池中。

(4) 负载均衡：系统设置有负载均衡(Router)模块，其上注册了所有的应用和位置，是应用的访问入口。

(5) 资源管理：通常情况下，用户的数据和状态信息都不保存在计算节点上，而是存放在服务节点上的数据库和缓存集群中，用户可以设置所需资源的额度和配置信息。应用通过接口或者代理来访问这些资源。

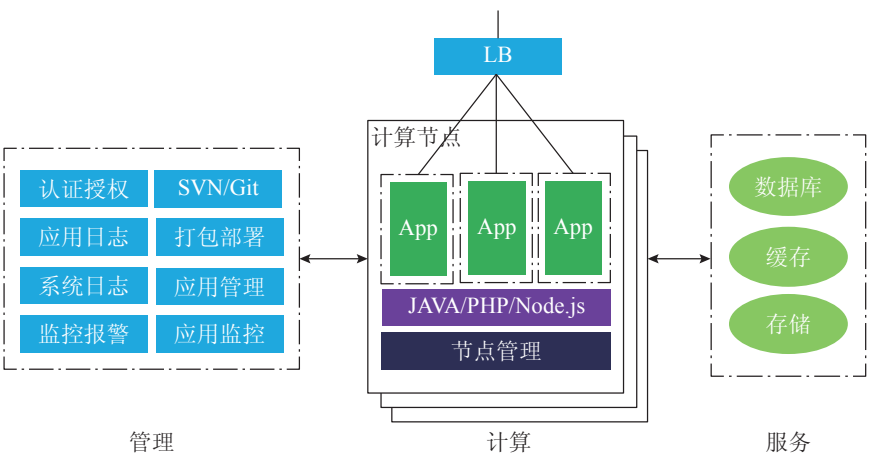


图 9-1 传统PaaS架构

(6) 应用商店：PaaS公有云提供商大都会设置应用商店，提供各种第三方应用，用户只需一键就可以将应用部署在系统中。

(7) 认证授权：系统中所有的访问都会通过授权认证模块的验证，保证系统的安全。

(8) 系统运维：用来监控系统中的各个模块的状态和信息；PaaS实时地监控系统中所有应用的状态和CPU、内存信息，发现应用意外停止时，系统会将其再次启动。用户可以手动启动、停止和升级应用。

9.1.3 传统PaaS的局限

PaaS平台只能提供有限的开发语言、框架和中间件支持，例如只支持JAVA、PHP、NodeJs和Ruby等，用户开发应用时所能选择的技术比较受限。

传统PaaS一般只支持Web+中间件+Database的单体(Monolithic)应用，因为缺少服务治理功能，无法支持复杂的分布式应用，对其他非Web类应用的支持也有限。

用户的应用与PaaS平台耦合强，PaaS平台为了管理或者安全的考虑，都会提供各种专属的SDK，用户的应用必须依赖这些SDK，并且要使用PaaS平台定制的框架和中间件来重新开发自己的应用。

由于其对开发者不够友好，开发者丧失了对环境和底层的控制力，且存在较多限制，使得开发效率不高。

PaaS标准不统一，使得跨不同供应商PaaS移植非常困难，直接导致了供应商锁定。

不能很好地保护现有IT系统上的投资，现有IT系统很难甚至是无法迁移到PaaS平台中，除非对应用进行大量重写。

9.2 基于Docker的新型PaaS

随着Docker容器技术的兴起和企业IT系统架构的发展，PaaS迎来了新的挑战和发展。

✍ Docker将应用和依赖的框架中间件等运行环境都打包到了镜像，用户应用进程在Docker容器中运行，不再依赖宿主机提供开发语言等计算环境支持。用户提交到PaaS平台的不再是代码而是应用的Docker镜像，将应用和PaaS平台做了解耦，不仅给用户开发应用带来了便利，PaaS也无须再准备各种语言的支持和复杂的应用打包过程。

✍ 近年来微服务框架越来越受到关注，之前通用的软件设计模式是使用单体式(Monolithic)架构，应用程序在开发、测试、打包和部署阶段都是作为一个整体存在。这种架构使得持续交付变得充满挑战，因为哪怕是应用程序的最小改变也需要整个应用重新编译和测试。微服务是一种将应用分解成小的自治服务的软件架构，每个服务被独立地开发、测试和部署，服务间使用约定的API进行通信，所有的服务组合在一起，通过API Gateway向外提供服务。微服务提高了应用的灵活性、扩展性和高可用性，在PaaS平台上部署微服务框架，需要系统提供完整的服务治理功能，包含服务的注册、发现、管理、授权、分布式事务、调用链分析等功能。

✍ 随着企业IT系统的发展，不仅仅需要PaaS提供关系型数据库、缓存和存储的服务，还需要大数据、NoSQL、搜索和机器学习等服务。

✍ 云计算技术逐渐被广泛地接受，用户的应用可能部署在企业内部的私有云上，也有可能部署到公有云上，所以PaaS应该可以有跨云部署的能力，可以将应用调度到不同的云计算环境中，提供给用户无缝连接的计算环境。

✍ 以应用的Docker镜像为软件的交付标准，集成开发测试流水线，实现应用的快速迭代。

9.2.1 新型PaaS的架构

新PaaS以Docker容器为基础，面向未来云化、微服务场景，对接大数据、ML/DL等多种计算服务，集成开发测试部署流水线，成为一个一站式的应用开发运行平台(见图9-2)。

新PaaS平台在构架上与传统的PaaS变化比较大，在每个计算节点上通常会部署负载均衡模

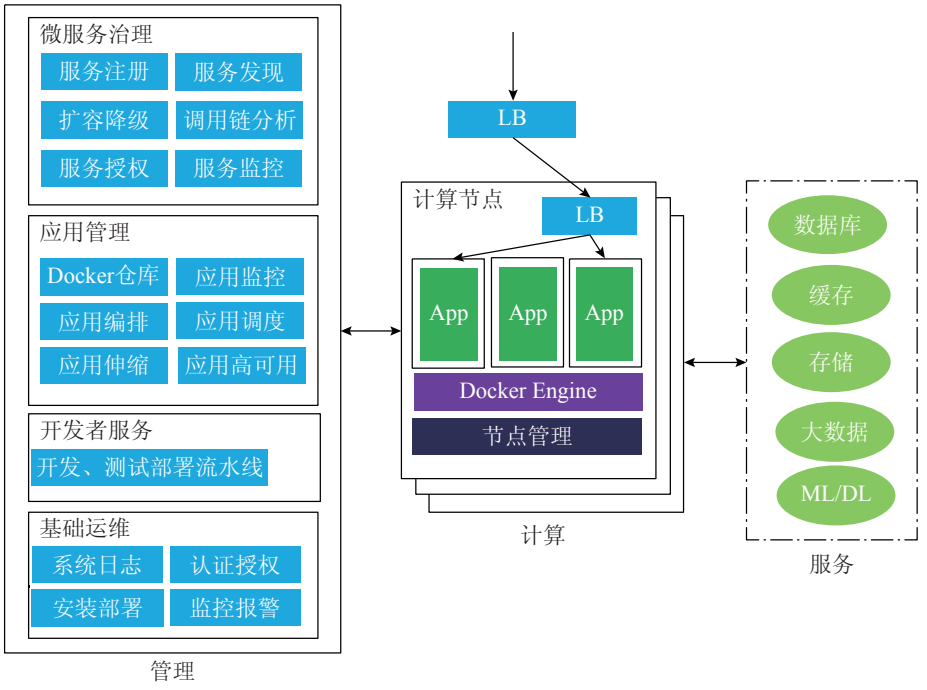


图9-2 新PaaS平台架构

块，为多实例应用或者微服务提供访问服务，实现了服务治理中的负载均衡功能。

管理节点上会安装etcd/zookeeper/consul，用于服务注册和服务发现，同时也是整个系统的配置中心。其还会部署DNS模块(如Skydns)，为系统中的应用提供名字解析服务。

在计算节点上，容器网络一般采用Bridge模式，采用Flannel、weaver或Calico实现跨节点的容器间互联互通，图9-3是采用Flannel方案的网络示意图。

容器的监控可以采用Cadvisor。Cadvisor部署在计算节点上，用于采集节点和之上的所有容器的运行信息，包括CPU/内存/磁盘I/O等。Cadvisor采集的信息可以保存到InfluxDB中，由Grafana来展示，也可以通过Heapster收集汇总后由Kafka转发至其他系统。

9.2.2 新型PaaS的功能

新PaaS平台需要至少提供如下的功能。

1. 应用编排

应用编排帮助用户构建和管理分布式应用，一般采用ymal或者json格式的模板文件定义应用各个模块，比如依赖的Docker镜像、环境变量、运行端口、健康检查机制与其他模块关系等，编排引擎调用容器调度模块在PaaS上构建出整个应用。

2. 容器调度

根据调度算法，在系统中创建应用编排模板中定义的容器应用。

3. 应用模块(容器)自动伸缩

为了高可用性，分布式应用的每个模块会在系统中部署多份容器，用户可以手工修改容器的数量，也可以定义多种策略，如：CPU/内存阈值、定期、周期以及访问量让系统自动增减容器数量。

4. 应用滚动升级

应用升级时，系统会用新版本的镜像创建容器，逐步增加数量，同步减少旧版本容器数量，

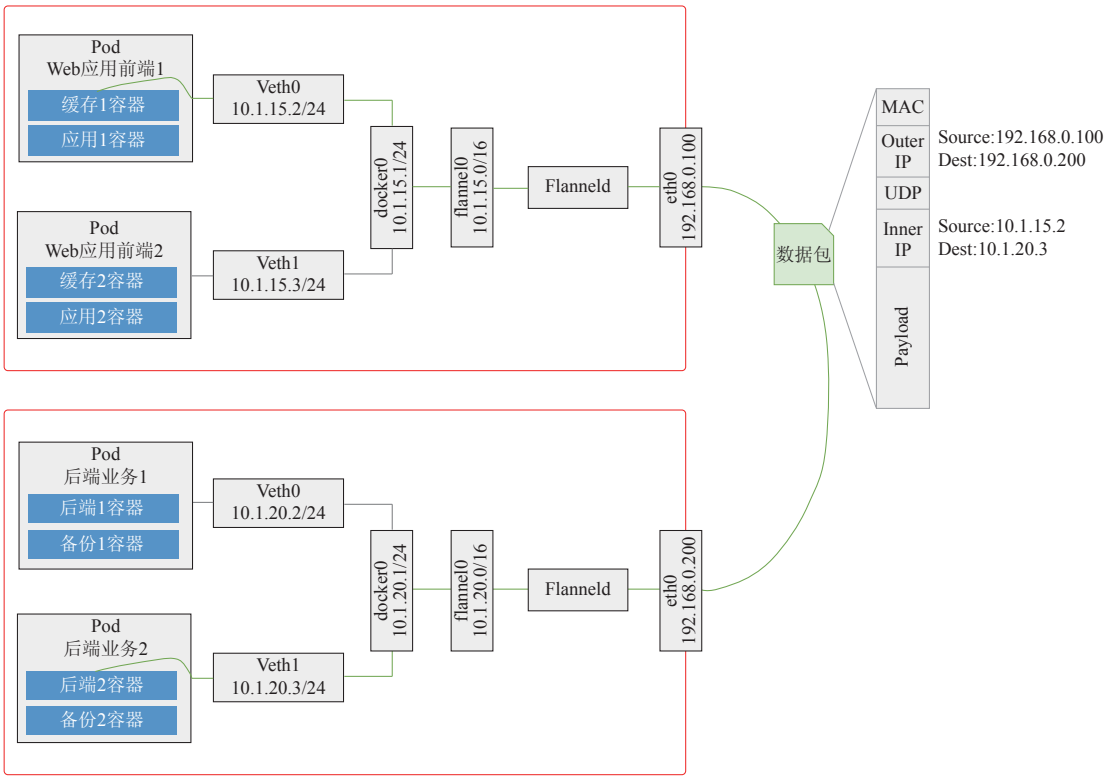


图9-3　Flannel方案网络示意图

实现对外服务不中断，如果应用升级失败，会回滚到旧版本。

5. 跨云部署应用

PaaS通过CloudAPI等接口可以部署在物理机群、IaaS系统和公有云上，实现跨云调度部署。

6. 对微服务架构的支持

微服务架构的核心是将一个大的单体应用分解为众多独立的服务，一个微服务在系统中可能有多个实例在运行，实例的数量可以根据负载进行调整。PaaS对微服务框架的支持体现在如下几个方面。

✍ 以Docker容器来封装微服务，因为每个微服务采用的开发语言都不一样，用Docker封装微服务，实现了应用与底层运行平台的解耦，提高了应用的灵活性。

✍ PasS平台实时监控每个容器的状态，保证每个微服务运行的实例数量，实现了微服务的高可用性。

✍ 用户可以定义基于CPU/内存的负载、定时或者周期等策略，自动地调整微服务的实例数量，实现应对负载变化的弹性伸缩。

✍ PaaS可以滚动升级容器应用，保证升级期间业务不掉线，与开发测试部署流水线集成，可以实现微服务的快速迭代升级，必要时可以回滚。

✍ PaaS的应用编排和调度功能使部署大规模的微服务成为可能。

✍ PaaS内部的负载均衡/代理模块用来将同一个微服务的多个实例集合起来对外提供服务。

✍ 服务治理模块提供了服务注册、发现和监控以及调用链分析等功能，能够快速将各个微服务集成在一起，对外提供高可靠、按需取用的云服务。

7. 服务治理

在分布式应用特别是微服务架构中服务众多，之间的调用关系又比较复杂，就需要一个统一的框架来管理这些服务，PaaS中的服务治理模块包含如下的部分。

- 服务注册，服务提供方将服务注册在中央的注册系统中。
- 服务发现，管理和更新服务的状态和信息。
- 调用链分析，记录每个服务调用日志信息，包含服务ID、调用方、参数、响应时间等，用于做debug分析。
- 服务降级或自动扩容，负载上升时可以增加服务的实例数量，过高时启动服务降级保护机制。
- 服务授权，根据认证授权模块，验证每次调用的权限信息。
- 服务监控，监控服务的状态。

8. 集成持续集成/持续部署系统(CI/CD)系统

图9-4是CI/CD与PaaS集成后，一次应用从源码提交到线上部署的自动化流程。

开发人员提交代码后，代码仓库(git)里的钩子(hook)触发CI系统的应用构建测试和发布流程，将通过测试的应用打包成Docker镜像上传到Docker镜像仓库中，调用管理节点上的应用部署接口，发起部署，整个过程无须人工干预，自动完成创建、打包和部署到计算节点上。

9.3 消息中间件服务

目前越来越多的分布式应用采用消息中间件来构建，消息中间件一般有两种传递模型：点对点模型(PTP)和发布-订阅模型(Pub/Sub)。点对点模型用于消息生产者和消息消费者之间点到点的通信。消息生产者将消息发动到由某个名字标识的特定消费者。这个名字实际上对应于消息服务中的一个队列(Queue)，在消息传动给消费者之前，它被存储在这个队列中。队列可以是持久的，以保证在消息服务出现故障时仍然能够传递消息。

发布-订阅模型用称为主题(Topic)的内容分层结构代替了PTP模型中的唯一目的地，发送应用程序，发布自己的消息，指出消息描述的是有关分层结构中的一个主题的信息，希望接收这些消息的应用程序订阅了这个主题，订阅了包含子主题的分层结构中的主题的订阅者可以接收该主题和其子主题发表的所有消息。图9-5展示了发布和订阅模型：

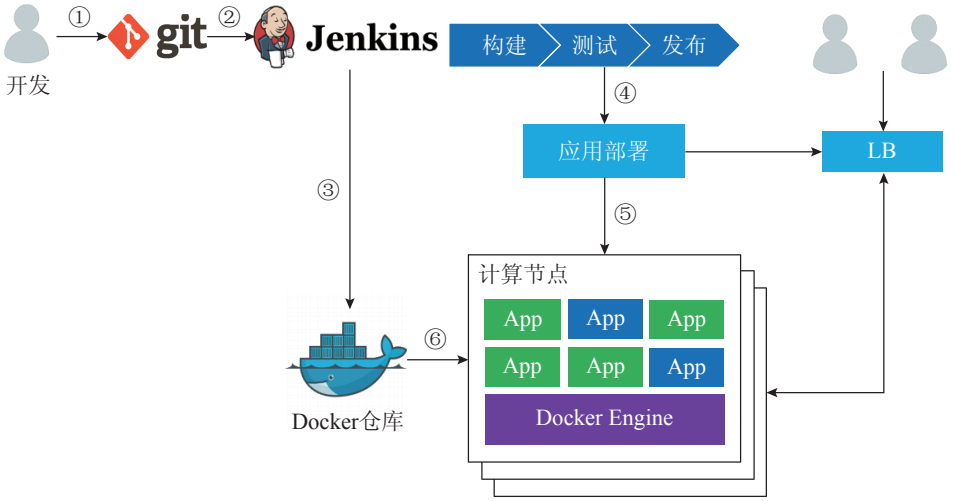


图9-4 PaaS开发自动化流程

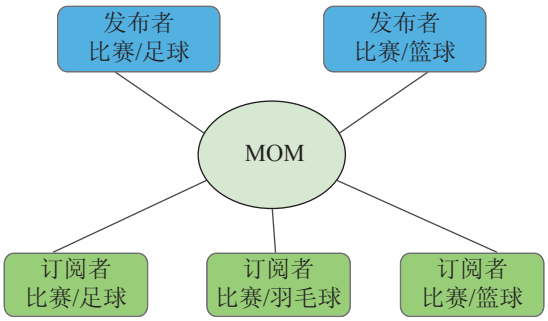


图9-5　发布和订阅模型

基于分布式架构的使用场景

在传统应用中，各个模块之间的调用是通过编程语言级别的方法或者函数来实现的。但是在整个的分布式应用中，需要将应用拆分成很多个微服务，因此一个基于微服务的分布式应用是运行在多台机器上的。一般来说，每个服务实例都是一个进程。因此，服务之间的交互必须通过进程间通信(IPC)来实现。当为某一个服务选择IPC时，首先需要考虑服务之间如何交互。通常有很多的交互模式，我们可以从两个维度进行归类。

第一个维度是一对一还是一对多。一对一：每个请求端有一个服务实例来响应。一对多：每个请求端有多个服务实例来响应。

第二个维度是这些交互是同步还是异步。同步模式：请求端请求需要服务端即时响应，甚至可能由于等待而阻塞。异步模式：请求端请求不会阻塞进程，服务端的响应可以是非即时的。

一对一的交互模式有以下几种方式。

- 请求/响应：一个请求端向服务器端发起请求，等待响应。请求端期望此响应即时到达。在一个基于线程的应用中，等待过程可能造成线程阻塞。
- 通知(单向请求)：一个请求端请求发送到服务端，但是并不期望服务端响应。
- 请求/异步响应：请求端发送请求到服务端，服务端异步响应请求。客户端不会阻塞，而且被设计成默认响应不会立刻到达。

一对多的交互模式有以下几种方式。

- 发布/ 订阅模式：请求端发布通知消息，被零个或者多个感兴趣的服务消费。
- 发布/异步响应模式：请求端发布请求消息，然后等待从感兴趣服务发回的响应。

每个服务都是以上这些模式的组合，对某些服务，一个IPC机制就足够了；而对另外一些服务，则需要多种IPC机制组合。

因此基于消息通信的异步模式就是使用基于异步交换消息的进程通信方式时，一个请求端通过向服务端发送消息提交请求。如果服务端需要回复，则会发送另外一个独立的消息给客户端。因为通信是异步的，请求端不会因为等待而阻塞，相反，请求端可以认为响应不会立刻接收到。图9-6展示了一个送餐服务请求中，分布式服务是如何通信的。

在图9-6中的服务使用了通知、请求/响应、发布/订阅等方式。例如，消费者通过手机给用餐服务发送通知，希望下一个午餐订单。用餐服务给消费者服务发请求/响应消息给消费者服务以确认有效性。当中午大家都在订餐，大量的服务

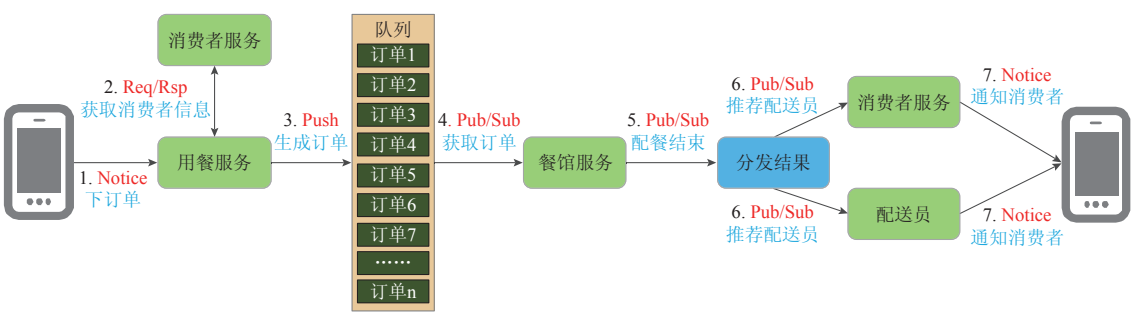


图9-6　分布式服务通信原理

请求到来时，正常情况下用餐服务的流量需要直接传导到餐馆服务。但是餐馆服务可能接受不了这么大的服务请求，会直接拒绝服务或者响应很慢，这样将导致非常差的用户体验。所以，可以在用餐和餐馆服务之间加入一个队列，第一可以达到负载均衡，第二点可以充当一个大的缓冲，这样可以把所有的流量缓存到队列里面。把用餐服务生产的消息放到队列里面，哪个餐馆订阅了相关消息就把消息拿走，这样给系统带来一个好处就是首先对于后面的这些应用而言，它们不会因为前面像海啸一样的流量而被压垮。最后餐馆服务到队列中取到订阅信息，在配餐结束后，生成配餐信息，并用订阅/发布模式通知其他服务，包括定位最近可用的配送员。

使用消息机制有很多优点。

(1) 解耦客户端和服务端：客户端只需要将消息发送到正确的channel。请求端完全不需要了解具体的服务实例，更不需要一个发现机制来确定服务实例的位置。

(2) Message Buffering：在一个同步请求/响应协议中，例如HTTP，所有的客户端和服务端必须在交互期间保持可用。而在消息模式中，消息系统将所有写入channel的消息按照队列方式管理，直到被消费者处理。如上例子，用餐服务接受消费者订单，由于在订餐高峰时，存在餐馆系统下单很慢或者不可用，只要保持下单消息进入队列就好了。

(3) 弹性请求端-服务端交互：消息机制支持以上说的所有交互模式。

(4) 直接进程间通信：基于RPC机制，试图唤醒远程服务看起来跟唤醒本地服务一样。然而，因为物理定律和部分失败可能性，他们实际上非常不同。消息使得这些不同非常明确，开发者不会出现问题。

然而，消息机制也有自己的缺点。

(1) 额外的操作复杂性：消息系统需要单独安装、配置和部署。消息broker必须高可用，否则系统可靠性将会受到影响。

(2) 实现基于请求/响应交互模式的复杂性：请求/响应交互模式需要完成额外的工作。每个请求消息必须包含一个回复渠道ID和相关ID。服务端发送一个包含相关ID的响应消息到channel中，使用相关ID来将响应对应到发出请求的客户端，或许直接使用一个请求/响应的IPC机制会更容易些。

9.4 数据库和缓存服务

RDS(Relational Database Service)是一种基于云计算平台的可即开即用、稳定可靠、弹性伸缩、便捷管理的在线关系型数据库服务。RDS具有完善的性能监控体系和多重安全防护措施，并提供专业的数据库管理平台，让用户能够在云中轻松设置、操作和扩展关系型数据库，简化运营流程，减少日常运维工作量，从而能够专注于应用开发和业务发展。

RDS通过垂直扩容(CPU、内存、连接数、IOPS、存储空间等) 支持读写分离，只读多备，扩展并超越单实例的性能限制，减轻主库数据读取负载，支持自定义数据库参数，优化数据库引擎性能；通过统一身份认证服务，实现权限访问控制；通过多只读副本和多重备份，实现安全存储，数据可靠性高；支持自动故障转移，从主库实例快速转移到备库实例上，保障数据库服务快速恢复可用，支持多只读副本、多可用区部署，以获得增强的服务可用性和可靠性。

表9-1描述了RDS与其他数据库解决方案对比有什么差异。

表9-1 RDS与其他数据库解决方案对比

功能	RDS	自购服务器搭建数据库服务
服务可用性	高	需自行保障，自行搭建主从复制，自建RAID等
数据可靠性	高	自行搭建主从复制，自建RAID等
系统安全性	防DDoS攻击，及时修复各种数据库安全漏洞	自行部署，价格高昂；自行修复数据库安全漏洞

(续表)

功能	RDS	自购服务器搭建数据库服务
数据库备份	自动备份	自行实现，但需要寻找备份存放空间以及定期验证备份是否可恢复
软硬件投入	无软硬件投入，按需付费	数据库服务器成本相对较高
系统托管	无托管费用	每台2U服务器每年超过5000元
维护成本	无须运维	需招聘专职DBA来维护，花费大量人力成本
部署扩容	即时开通，快速部署，弹性扩容，按需开通	需硬件采购、机房托管、部署机器等工作，周期较长
资源利用率	按实际结算，100%利用率	考虑峰值，资源利用率很低

RDS适合于需要关系型数据库全部特性与能力的开发者或企业，也适合于希望用关系型数据库来迁移现有的应用和工具的情况。RDS主要适用场景包括以下几个。

(1) 互联网网站：在线游戏、电子商务、电子政务、企业门户、社交平台、社区论坛等网站可以迁移到云平台上，使用RDS来快速获得低成本、高性能、易使用、安全可靠的数据库支撑服务。

(2) 企业应用系统：企业办公应用、SaaS化应用等业务系统可以迁移到云平台，由RDS来支撑业务数据管理需求，减少IT建设投入成本和人力维护工作量，可随时随地办公或使用SaaS服务。

(3) 软件开发测试：软件开发者可以在云平台搭建开发测试环境，无须花费大量时间和成本自建数据库，直接使用稳定可靠的、不同性能规格的RDS来联调测试，从而能够聚焦应用开发，缩短软件发布时间。

(4) 云存储Redis：兼容开源Redis协议中定义的所有数据类型，如String、Hash、List、Set、SortedSet等，支持多种数据操作，充分满足业务需求，适用于缓存和Key-Value数据库等多种场景。

(5) 云数据库MongoDB：对于初创型的业务非常适用。MongoDB采用No schema的方式，免去变更表结构的痛苦，将模式固定的结构化数据存储在RDS中，模式灵活的业务存储在MongoDB，并将高热数据存储在Memcache中。MongoDB的高并发读写能力性能优异，适用于高并发场景下的数据读写；Mongo内部支持MapReduce框架，可以将大量数据进行聚合分析，在MongoDB引擎内完成数据内部加工。

(6) 云缓存MemCached：主要用于为热数据提供高性能缓存，配合数据库使用能极大提高系统性能，解决了内存数据可靠性、分布式及一致性上的问题，让海量访问业务的开发变得简单快捷。

9.5 大数据服务

随着互联网的飞速发展，企业产生的数据与日俱增，如何让这些复杂无序的数据产生价值，如何让企业轻松驾驭这些海量数据信息来创新、快速洞察商机，这是现今企业的迫切需求。传统数据的处理分析能力已经不能满足要求，大数据时代的到来解决了这些迫在眉睫的需求。

9.5.1 弹性大数据

通过弹性大数据来打造高可靠、高安全、易使用的运行维护平台，对外提供大容量的数据存储和分析能力，可解决各企业的数据存储和处理需求。用户可以独立申请和使用托管Hadoop、Spark服务，用于快速在主机上创建Hadoop、Spark集群，提供海量数据的实时性要求不高的批量数据存储和计算能力。图9-7是弹性大数据的架构图，提供具有以下特性。

一、海量数据的分析和计算

基于分布式系统基础架构Hadoop，采用MapReduce实现对大数据集(大于1TB)的并行运算。

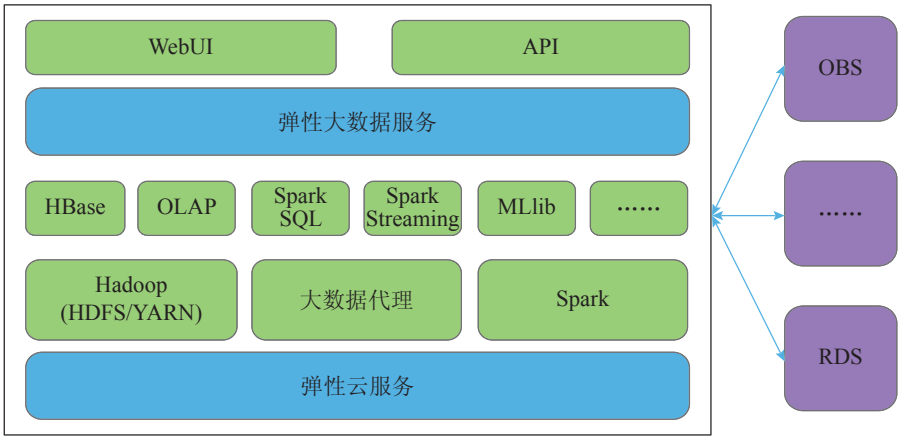


图9-7　弹性大数据架构图

Spark是分布式批处理框架，提供分析挖掘与迭代式内存计算能力，支持多种语言(Scala/Java/Python)的应用开发。其还提供了Spark SQL特性，即SQL on Hadoop方案，通过SQL语句可直接进行数据查询和分析。另外，其还支持使用Spark SQL提交Carbon的DDL和DML语句，对数据提供秒级响应查询。Carbon也支持使用SQL语句进行复杂的数据互动分析。

HBase是一个高可靠性、高性能、面向列、可伸缩的分布式存储系统。其设计目标是用来解决关系型数据库在处理海量数据时的局限性。其还提供了Spark SQL on HBase特性，即Astro。Astro提供了一种在Spark中直接通过SQL方式访问HBase的方法，方便用户对HBase中数据的交互处理，同时它对row key中各维数据做了编码处理，保证了各维数据的有序性。

二、海量数据的存储

Hadoop分布式文件系统(Hadoop Distributed File System)能提供高吞吐量的数据访问，具有高容错性的特点，适合大规模数据集方面的应用。利用HDFS的高吞吐性能读取大规模的数据进行计算，数据处理分析完成后，采用SSL加密传输数据至OBS系统，也可将数据存储在HDFS中。

三、日志分析

日志分析包括：存储用户日志并对日志进行格式化；对用户建模，提取用户特征。

弹性大数据具有以下两大特点。

- 易于使用。弹性大数据使用简单，用户只需几分钟就可以启动大数据集群，用户不必花时间关注节点调配、集群配置和集群调试。大数据服务会自动处理这些任务，用户只需集中精力分析。其提供SQL和OpenAPI等多种易用接口，大幅度降低开发人员编程难度。
- 成本低廉。按需使用弹性大数据服务，在作业结束后可以释放计算资源，提升资源使用效率。

9.5.2　流计算

流计算服务是基于流式计算模型打造的具有高吞吐低时延、动态扩展、高并发数据处理、丰富的公共算子特点的分布式高性能实时计算框架。目标是提供突破性的功能，实现从海量且多样的数据中提取出有价值的信息。基于先进的分布式实时计算框架、CEP规则引擎，大幅度提高了信息处理的时效性和可靠性。

流计算主要应用于对海量数据的实时分析(仅存在微小的延迟)。流计算不像Hadoop一样搜集大量数据、批量处理数据、将数据存储到磁盘上，然后进行分析(换句话说，是指静止数据分析)。在流计算中，数据将会流过有能力操控数据流的运算符，然后对这些数据执行动态分析。这

样可使企业利用即时的智能分析实时采取行动，最终改善业务成果。

流计算平台具有以下特点。

(1) 流计算平台的可扩展性很高，支持的数据流量也比传统实时计算系统多得多。数据流可以看做由一系列连结运算符组成，每个连结运算符都具有各自的运算功能。流计算作为一个平台，用户能够以任何方式构建或自定义流计算，从而提供应用程序来解决各种业务问题，同时，这些运算符中的每一个均可在集群中的独立服务器上运行，只需增加额外的服务器和分配这些服务器上运行的运算符，即可增加执行数据流分析的服务器数量，从而提高可用性、可扩展性和性能。

(2) 流计算平台可与Hadoop平台一样，采用分布式调度管理机制，在运行时根据集群类负载平衡和可用性指标自主确定处理元素 (PE) 的运行位置，从而使其能够重新配置运算符在其他服务器上运行，确保一旦服务器或软件发生故障，数据仍能持续流动。同时，其还能够以编程的方式指定在哪些服务器上运行哪些运算符，并可在特定的服务器上运行流逻辑。

流计算不仅极其适用于结构化数据，而且适用于其他 80% 的数据(包括传感器数据、语音、文本、视频、财务以及许多其他来源生成的非传统半结构化数据或非结构化数据)。

9.5.3 机器学习

当前，数据挖掘在应用场景不断丰富，数据日益膨胀，数据种类繁多，数据分析领域面临重重挑战，如：如何在大数据中发现海量的小数据特征；如何让数据分析人员更加聚焦价值发现，降低在编码和基础数据处理的投入；如何构建数据挖掘领域的下一代数据分析工具平台。

机器学习，即通过大数据分析挖掘平台，构建在当前流行的Hadoop/Spark大数据技术之上，采用分布式存储和并行计算技术，从海量数据中挖掘出价值信息的平台；同时，其兼容通用的数据分析Python/R环境，可以使用自己擅长的语言进行数据分析，并且提供了分布式算法(Java/R/Python)以及可视化能力，帮助用户有效快捷地完成数据分析，通过对各类海量数据信息进行实时和非实时的分析和挖掘，帮助企业从海量数据信息中获取到真正的价值，及时洞察和决策新的机会与风险。

机器学习的上下游关系解释如图9-8所示。

(1) 与用户(数据科学家)的关系：数据科学家

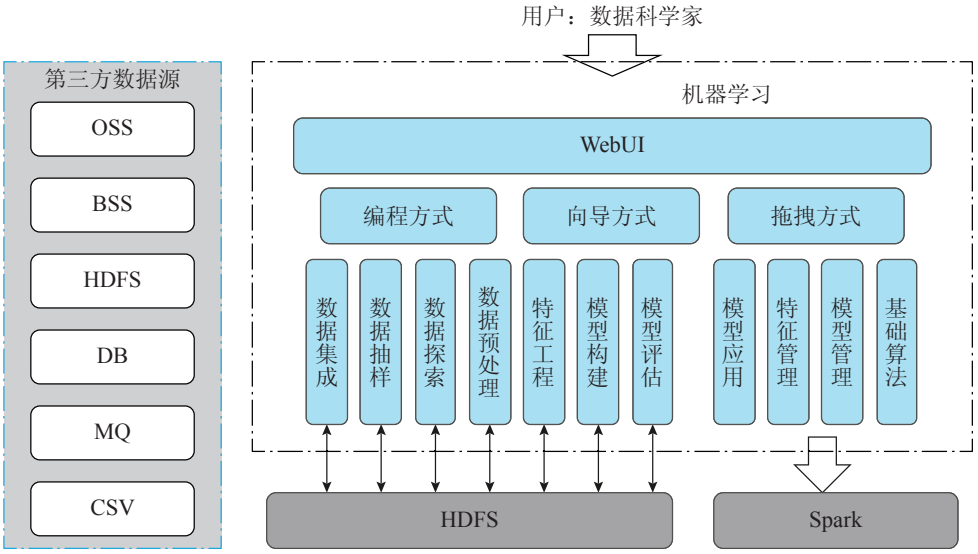


图9-8　机器学习的上下游关系

可以通过机器学习平台，进行数据分析。数据科学家是机器学习面向的一类用户，这类用户拥有一定的编程技能，可以使用编程方式进行数据分析动作。

(2) 与第三方数据源的关系：通过数据集成将第三方数据源的数据统一存放在HDFS上存储。

(3) 与底层HDFS和Spark的关系：机器学习直接从HDFS上获取数据，通过Spark的分布式内存计算能力，对数据进行分析、处理和挖掘。

表9-2为机器学习模块的说明。

表9-2 机器学习模块说明

模 块	说 明
数据集成	将来自不同数据源的数据集成到统一平台中来
数据抽样	使用抽样技术，将大数据总量降低，支持快速分析
数据探索	探索数据，发现数据特征和价值
数据预处理	对数据进行规整、去空等预处理，总之就是保留有价值的，去除无价值的
特征工程	依照原始特征数据，进行特征工程，例如特征降维，特征衍生等
模型构建	特定特征和数据输入给算法，然后构建出模型的过程
模型评估	将测试数据构建在特定模型上，得出这个模型的评估指标，例如混淆矩阵、F1Score等
模型应用	将构建好的模型，进行应用发布，可以在生成环境上按照一定规则运行
特征管理	特征工程的管理，包括查看、共享、删除等操作
模型管理	模型的管理，包括查看、共享、删除等操作
基础算法	基本的算法，包含了业界/学术界通用的算法，例如决策树、SVM、随机森林等

第 13 章

云微服务敏捷治理架构与组织流程

13.1 从瀑布式到敏捷式，从服务到微服务

13.1.1 云化软件服务化架构产生的背景

云计算技术和云服务产品的发展，使得软件的架构和开发模式发生了重大的变化。最近几年，传统企业的IT系统在逐步地向基于云计算基础设施的IT系统演变，软件产品的运行环境已经发生了深刻的变化，由此伴随着软件本身的架构设计与实现也在向云化和服务化的方向转型。

云化软件是为了支撑基于云计算基础设施的服务化的开发、构建和部署维护应运而生的一类软件的统称，随着云计算的发展，基于云计算的基础设施开发的提供各种功能和服务的诸多软件，也被称之为云化软件。基于云计算的基础服务具备五大基本特征：(1)按需获得的自助服务；(2)广泛的网络接入；(3)资源池化；(4)快捷的弹性伸缩；(5)可计量的服务。

以上的五大基本特征决定了云化软件的架构具备面向服务、弹性伸缩、灵活扩展的特点，要求能够灰度发布和灰度升级。云化软件也大量采用DevOps模式来开发，开发效率较高，软件的服务化能力较强。

13.1.2 微服务架构模式的定义

关于云化软件的架构实现，业界有很多的探索和实践。微服务化架构是近两年业界在实现云化软件架构中总结出来的一种架构模式。微服务本身没有一个严格的定义，业界对于微服务的架构模式在实践中给出了概括性的描述。

微服务架构是一种架构模式，它提倡将单一应用程序划分成一组小的服务，服务之间相互协调、相互配合，为用户提供最终价值。每个服务运行在其独立的进程中，服务与服务间采用轻量级的通信机制互相沟通(通常是基于HTTP的RESTful API)。每个服务都围绕着具体业务进行构建，并且能够被独立地部署到生产环境、类生产环境等。另外，应尽量避免统一的、集中式的服务管理机制，对于具体的服务，应根据业务上下文，选择合适的语言、工具对其进行构建。

微服务架构模式，核心是将复杂应用划分成小颗粒度、轻量化的自治服务，并围绕微服务开展服务的开发和服务的治理，这是实现云化软件的一种架构模式，也是近两年最热门的架构模式。

实践中，微服务架构模式中的微服务具有如下几个主要特点。

1. 小

微服务架构通过对于特定的业务领域进行分析和业务建模，将复杂的业务逻辑剥离成为小而专一、耦合度低并且高度自治的一组服务，每个服务都是很小的应用。

虽然强调每个服务的小，但是每个微服务本身还是完整的应用，这与我们通常说的组件、插件、共享库是有区别的。

微服务的规模也没有严格的定义，通常一个微服务的开发团队在6~8人左右，一个微服务的架构重构可以在2周时间内完成这样的描述来确定一个大致的规模。这样的开发团队能够完成的开发规模就是一个微服务规模的上限。

2. 独

独，指的是微服务的独立性。这里的独立性主要是针对一个微服务应用的交付过程而言的，也就是开发、测试以及部署升级的独立。

在微服务架构中，每个服务都是一个独立的业务单元。这个业务单元在部署形态上，是独立的业务进程。对某个微服务进行改变，不会影响其他的服务。

对于每个微服务，都有独立的代码库。某个微服务的代码修改，不会影响其他的微服务。

对于每个微服务，都有独立的测试验证机制，不必担心破坏完整功能而开展大范围的回归测试(这往往是现有大集成全覆盖测试研发模式中消耗很大但测试结果却不让人放心的地方)。

3. 轻

微服务强调服务自治，因此服务之间的交互，必须采用消息通信的方式予以开展。从效率的角度，应当选择轻量级的通信机制。在软件实

现的实践上，RESTful API的方式被广泛采用。这种通信机制的优点是语言无关、平台无关，并且十分便于制定通信协议，保证接口的前向兼容性。

在从传统软件架构向微服务化架构演进的过程中，业界实践部分也保留了RPC的通信机制，但要求基于RPC进行通信协议的制定，以保证接口的前向兼容性，实际是为了支撑服务间的独立和服务的松耦合态。

4. 松

松是指微服务间松耦合，每个微服务可独立部署，互相之间没有部署先后顺序的依赖，微服务的接口前向兼容，单独微服务的上线，对于其他服务而言不产生关联，可以独立地灰度发布和灰度升级。

实现微服务间松耦合还有一点需要注意，就是一个微服务完成且最好仅完成一件事，业务逻辑的独立是微服务间解耦的关键。

13.1.3 微服务架构模式与传统架构模式的对比

微服务架构模式成熟之前，软件领域讨论的比较多的是SOA的架构模式。SOA早在1996年就由Gartner提出，作为面向服务的架构模式，SOA的理念是对于复杂的企业IT系统，按照不同的、可重用的粒度划分，将功能相关的一组功能提供者组织在一起为消费者提供服务。SOA在实际的发展过程中并不顺利，随着ESB(Enterprise Service Bus)、Web Service、SOAP等技术出现，SOA才渐渐落地。但其主要解决的是企业内部不同IT资源之间的信息共享、互通等问题，相当长时间内的发展依赖于企业级软件总线的进展，而这类软件总线总体上厚重、大而全，但相互之间由不同的软件厂家提供，自成体系无法互通，从而又制约了其发展。

从SOA的面向服务而言，其与微服务架构模式中的服务化这类思想是基本一致的。落地实践的过程中，微服务与SOA又有着明显的区别，如表13-1所示。

表13-1 SOA软件架构模式与微服务软件架构模式的区别

微服务架构模式	SOA架构模式
服务自治，服务间通过轻量级的通信机制通信，没有集中统一的总线架构	企业级服务总线，统一规范，集中式的服务架构
服务单独部署，灰度发布和灰度升级，服务独立上线，无相互依赖和相互影响	通常是单块系统架构，相互依赖，需要联动升级，部署复杂
基于微服务的应用集成相对简单(HTTP/REST/JSON)	基于企业级通用平台的集成，比较厚重(ESB/WS/SOAP)
服务的划分粒度细，6～8人负责一个微服务构建	服务由多个子系统构成，通常还共用组件，服务粒度划分粗

13.1.4 微服务架构关键概念说明

软件领域有比较多的概念使用，为了更好地体现本章节所引用概念所指的确切含义，表13-2对关键概念名称进行澄清。

表13-2 关键概念澄清

关键概念名称	澄清和说明
组件	软件模块的统称，通常是一个独立功能的软件实体，本文中所指软件组件，指的是构成微服务的软件模块，比微服务的粒度更小
插件	软件插件对应一种软件架构设计模式，即在相互交互的软件模块间定义好接口(这类接口有函数性接口、RPC接口等)，可以通过适配的方式，灵活地更换对接方的软件模块，统称为插件；部分插件支持部署后的动态加载和卸载；本文中所指的插件，指的是微服务中可以支撑部署后动态加载、卸载的软件模块
服务	对外提供服务化API调用的软件实体，本文中的服务特指微服务；通常软件领域所指的服务，按照不同的服务消费者，划分的粒度也有很大的不同；例如面向公有云租户的服务，通常是租户可见的大颗粒度的消费服务，往往由多个微服务组成的一个应用来承载
云化软件	本章中的云化软件，指为了支撑云服务构建，采用分布式技术进行设计开发的所有软件的统称

13.2 微服务的治理架构

传统软件的分层架构、模块化、平台化，都强调治理模式统一，归一化，这种治理架构在面向大规模软件开发的过程中，越来越制约着软件开发效率的进一步提升。例如希望在架构设计阶段进行理想的模块抽象和拆分，并且保持良好的演进能力，这是一个破坏效率提升的因素，大量的时间花费在架构的抽象和细化上，且缺乏足够的手段来验证架构的合理性；软件在开发阶段，因为多个模块的实际开发量不同，互相依赖，造成开发环节的进度协调困难，这是第二个破坏效率提升的因素；在软件的测试阶段，互相依赖的各个模块实际上需要一起测试，大量的问题发现集中在软件集成之后，严重地拖慢了整个的测试进度，这是第三个破坏效率的因素。

微服务是将复杂的软件系统微观化敏捷地变革，虽然微观上每个微服务自治了，可以独立地开发测试部署了，但是给整个软件系统的治理带来了新的调整：(1)微服务如何划分才能达到效果；(2)微服务的数量众多，服务部署、上线、升级的操作众多，自动化的诉求迫切；(3)系统出现异常时的故障界定困难。

13.2.1 微服务的划分

微服务的划分业界并没有严格的规定，基于业务的建模分析是开展微服务划分的主要方式，微服务业务建模需要开展的活动为以下几种。

(1) 云化软件系统的服务能力分析：基于满足服务消费者视角的服务API定义，决定了云化软件系统的对外服务能力。通常服务能力不是由设计者来确定的，而是由客户，也就是服务消费者来决定的。

(2) 云化软件系统的部署架构分析：云化软件主要是分布式架构，控制逻辑单元、管理逻辑单元、代理逻辑单元通常是分布式部署的。在微服务化的架构模式下，微服务之间是隔离的，微服务不共享数据库，微服务间通过API进行消息交互。云化软件原有的部署架构需要根据微服务的隔离形态进行分析，并且结合微服务提供的负载均衡器的能力进行分析，通过部署来进一步验证微服务划分的合理性。

(3) 云化软件系统的软件组件分析：划分微服务时，需要分析单个微服务运行所包含的软件组件、数据库、消息通信组件等。微服务需要整理独立完整运行的软件组件列表，在进行微服务拆分时，需要保证软件组件的完整性。

(4) 云化软件系统的逻辑分层分析：通常的云化软件系统，除了以微服务为粒度的服务化划分方式进行的软件架构分解分析之外，还应开展分层软件架构视角的分析。通常的软件逻辑平面，有数据面、控制面、管理面的划分。数据面对于的是网络、存储、计算等数据业务的转发，有各自的领域通信协议，例如iSCSI、FC、NFS等，这部分无法以微服务的轻量化通信机制进行规范，保持原有即可，但数据使用上可以尽量采用类似于对象存储和非关系型数据库的数据访问方式进行改造。对于控制面和管理面，这部分需要采用微服务间的轻量化通信协议进行规范化。

(5) 微服务负载均衡器选型分析：云化软件的分布式系统，以及对于灰度发布和升级的要求，都需要负载均衡器进行单一微服务多个实例之间的负载可控分发。业界采用的一般为Haproxy，或者Ngix+LVS，也可以根据需求，通过自行开发的API G网关来提供负载均衡。通常选择自行开发的API网关，还能够提供一部分业务编排的能力。

云化软件的开发实践过程中，大量使用了开源软件。开源软件在云化软件的实现中，存在三种使用方式：

- 开源软件的部分代码被摘取，作为云化软件的一部分代码片段被集成；
- 开源软件的软件包以一个完整软件组件的方式，作为云化软件中一个微服务的一部分被集成到微服务中；
- 开源软件的软件包以一个独立的微服务的方式，作为云化软件的一个微服务集成到整个软件系统中。

由于开源软件本身的架构设计不完全是微服

务化/服务化的，在开展云化软件的微服务化设计时，应以开源软件的独立软件包粒度进行微服务的划分和集成。对于开源软件已经提供的独立软件包，不要进一步拆分为多个微服务，除非开源软件社区的架构设计已经微服务化了。

开源软件包如果在作为微服务时，未能满足微服务化架构要求的，应考虑提供与微服务框架的集成，实现微服务的架构原则。

13.2.2 从单块架构向微服务化架构的演进

单块架构向微服务化架构的调整，是一个逐步的过程。

微服务化架构给研发模式带来的好处是研发效率的提升，软件的快速迭代，缩短上市周期。为了达成这样的效果，在微服务的研发过程中引入工具，实现自动化，也是一个逐步迭代的渐进的过程。

这个过程分为三个阶段。

(1) 单块架构的服务化调整：对现有系统架构的一个服务化调整，调整后的架构划分颗粒度较粗，不满足微服务的要求，但服务实体对外提供契约化的服务接口，可以独立部署。

(2) 服务到微服务的调整：选取典型的服务实体进行进一步微服务化的调整，粒度进一步细化，微服务的研发工具需要引入并在团队中贯通。

(3) 全软件系统微服务化：全软件产品系统转变到微服务化的研发模式上来，需要贯通从需求到上市的端到端环节(不仅仅是研发)。

在阶段二，可以选取典型的服务实体，进一步微服务化，这个过程需要引入微服务的研发工具作为支撑，并需要匹配微服务的CI/CD工具——Pipeline工具。针对已经微服务化的服务开发团队，需要调整到面向微服务的一个完整团队去交付。

阶段三，最终的全软件系统微服务化，需要具备微服务治理系统、微服务的CI/CD工具、Pipeline工具、微服务的测试系统、微服务的维系统、微服务接口管理、依赖关系管理系统。只有以上这些系统与工具对接并平稳运行，这样才能保证研发效率的提升。这个阶段的全体研发团队的组织需要调整，面向微服务的开发模式组织团队。

13.2.3 微服务开发框架

微服务是一个独立完整的服务化实体单元，在云化软件系统中，完成一个应用，需要多个微服务的协同配合，同时也引入了对于微服务的管理和微服务的维护等一系列的要求。实践中，一般通过提供统一的微服务开发框架，来实现这些要求。

微服务开发框架应包含的内容为微服务注册、微服务发现、微服务代码框架模板、微服务日志、微服务监控、微服务告警、微服务安装部署升级、微服务测试、微服务的HA、微服务负载均衡、微服务消息队列、微服务缓存、微服务关系型数据库访问等。具体如图13-1所示。

目前与微服务开发框架相关的项目有不少，但是多数的开源项目比较零散，比如提供注册、发现能力，而不是整个微服务框架，而Spring Cloud提供了比较全面的微服务开发支持，虽然存在一定的编程语言限制(仅Java)，但是不得不说对微服务开发快速上手提供了很大的帮助。

Spring Cloud提供的能力包括Distributed/versioned configuration、Service registration and discovery、Routing、Service-to-service calls、Load balancing、Circuit Breakers、Global locks、Leadership election and cluster state、Distributed messaging等。

具体Spring Cloud的编程可以直接访问官方网站学习，下面以一个简单微服务来了解具体的微服务工作流程，如图13-2所示。

Movie Service为新开发的一个微服务。Eureka(Netflix开源) 提供配置注册、发现能力，其中Eureka Serer为注册发现服务器，Eureka Client为微服务提供了注册接口，以及发现其他微服务访问地址的能力(这种客户端模式发现，通过与Ribbon配合具备了Load Blance能力)。Ribbon是

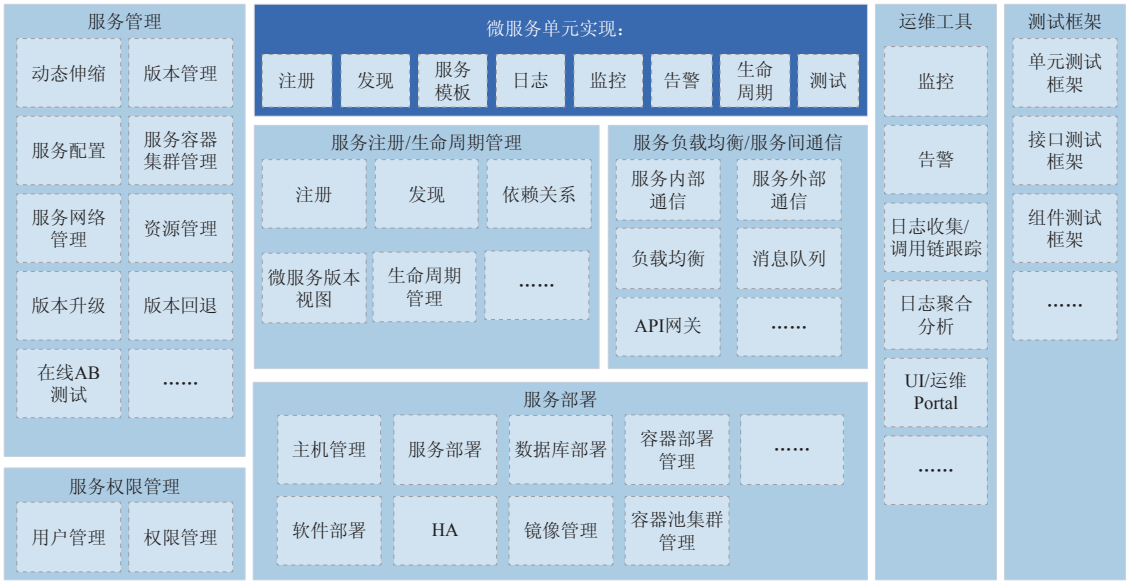


图13-1　微服务框架

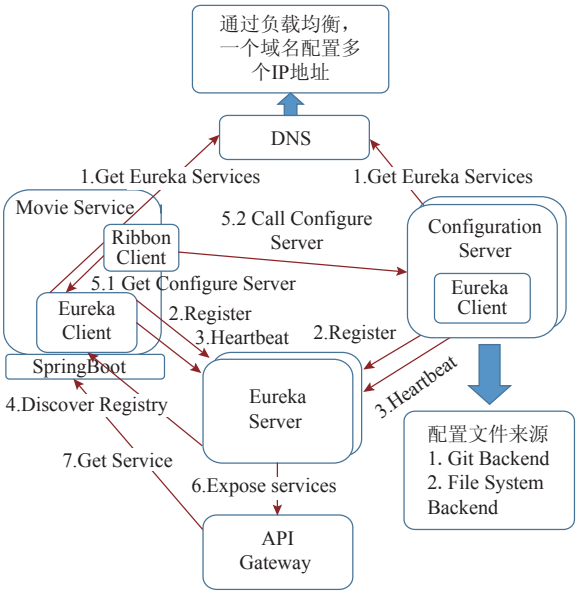


图13-2　Spring-Cloud微服务框架样例

客户端的负载均衡器。Configuration Server提供配置服务，配置数据可来自于Git、SVN、文件系统等。

API Gateway提供外部系统访问的能力，其具体交互流程如下。

(1) 微服务Movie Service通过DNS服务器获得Eureka服务的地址(在这里也可以直接将Eureka服务的地址写入到Movie Service的配置文件)。

(2) Movie将自己的服务信息注册到Eureka。

(3) Movie与Eureka维持心跳信息。

(4) Movie微服务会保存Eureka收集到的集群中各个服务的相关信息，并在维持心跳的过程中确保信息是最新的。

(5) 应用需要获取配置数据，Ribbon负载均衡器选择可用的配置服务节点；访问Configuration Server获取配置数据，Configuration Server访问实际数据源获取数据。

(6) API Gateway从Eureka中获取集群中微服务的地址。

(7) 通过API Gateway访问集群中的微服务(外部访问触发)。

注：Eureka可以维持多个实例，每个实例会互相维持信息的一致性。

Spring-Cloud提供的微服务支持是一堆松散组合的框架，如配置框架、注册发现框架、微服务单元的开发框架，并需要外部做一些支撑工作整个系统才能运转。

13.2.4 微服务的服务发现和服务注册

微服务的服务注册由两种方式来完成。

一种是由服务部署系统来完成，由服务部署系统根据服务的部署位置和服务参数，统一设置到服务注册中心。

一种是由微服务自身来完成，在微服务部署完成后，微服务自主发起向注册中心的注册。

微服务需要注册的信息主要有以下几点。

- 服务的Endpoint，基于RESTful的API的服务，提供就是一个HTTP的访问地址(也可以基于其他接口方式提供，但是RESTful方式支持多语言更加方便)；
- 服务的SLA信息；
- 服务的版本信息；
- 服务能力列表(可选，推荐)；
- 服务的名称和标识。

服务发现是由服务消费者完成的，通过服务消费者向服务注册中心查询，获取需要访问的服务的信息。

服务注册中心的功能，主要包含：

- 接受服务节点上线注册；
- 服务节点下线注销；
- 提供服务发现订阅方式，返回消费者所要调用的服务节点列表；
- 提供服务订阅取消方式，返回消费者所要调用的服务节点列表；
- 提供注册中心与服务的心跳机制；
- 获取服务消费者所依赖的服务和可用情况；
- 提供服务访问的基本权限控制；
- 查询服务的所有接口和SLA；
- 查询消费者所依赖的服务。

13.2.5 微服务接口管理

微服务的接口反映了每个微服务的服务能力，为了在微服务化架构下对于每个微服务做到独立、自治、灰度发布和灰度升级，微服务的接口需要保持前向兼容，并且需要纳入到接口变更管理和接口的自动化测试验证去保证。

一、接口版本管理

微服务的接口是演进的，也会有变化，但必须保证严格的前向兼容。所以接口需要引入版本管理。接口的能力发布对应具体的版本。

二、接口的工具化发布

如图13-3所示，我们可以获知如下信息。

(1) 微服务提供服务接口的定义，对应API的设计过程和发布到接口管控平台。

(2) 根据服务接口定义，完成测试接口的服务端，包括接口打桩的过程。

(3) 通过接口定义，生成接口代码框架(可以借助工具自动生成，或者采用手动编写的方式)。

(4) 服务消费者通过UI进行接口定义的查看(接口管理工具提供基于UI的查询方式)。

(5) 根据接口定义，生成服务消费者的调用实现。

(6) 服务消费者根据接口调用实现，完成与接口服务端的对接。

(7) 根据接口代码框架实现微服务的开发工作。

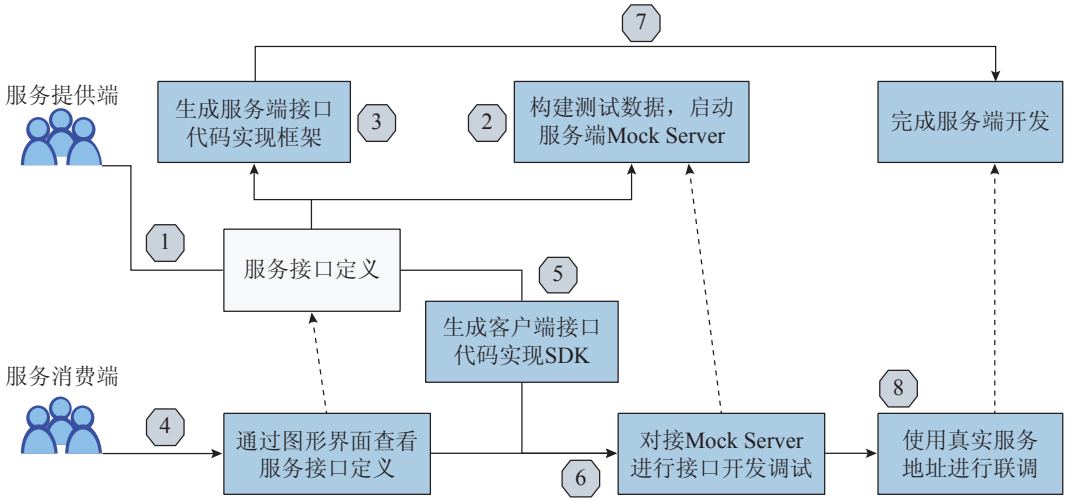


图13-3　微服务接口管理

(8) 基于真实部署的微服务进行服务消费者和提供者测试。

推荐的接口管理工具是Swagger。

13.2.6　微服务依赖关系管理

微服务的依赖关系，严格意义上不是指微服务间的耦合关系，而是从业务实现角度出发，通过各个微服务之间的服务能力组合，从而支撑一个完整应用的组合关系。比如A服务调用B服务的服务接口。A+B服务实现了一个应用。微服务的依赖关系管理，在于采用自动化的方式实现服务间的业务管理呈现，从而实现以下的目标：

- 可视化呈现服务依赖关系；
- 提前识别特性依赖的服务；
- 结合依赖关系，指导集成验证、现网服务特性打开时机；
- 可以导入、呈现服务调用热度情况，识别循环调用；
- 通过动态注册和服务发现，使服务位置透明，实现软负载均衡和故障转移；
- 依据服务的调用数据和响应时间等指标，作为服务容量规划的参考；
- 服务依赖关系管理通过自动化的工具从服务实现的代码中提取依赖关系；或者依赖于微服务框架的调用链分析功能实现。业界当前可选型的工具为PACT。

13.2.7　微服务的测试

服务创建阶段主要进行服务划分、接口定义、设计规格和服务管理信息录入等。

服务开发根据服务设计和接口定义设计用例，进行测试代码开发。服务编码完成本地调测后，提交到配置库(Git配置库)。Pipeline启动自动化任务，生成服务Alpha验证环境(含测试桩)，编译生成服务并部署，最后执行自动化用例并反馈结果。

服务由服务Owner决策发布Beta环境，进入服务验证阶段。首先选择新版本的服务实例升级或安装部署到哪些节点，再由Pipeline启动自动化任务，执行服务级和特性级测试用例，版本稳定后进行专项测试及需求场景验收。

服务通过灰度发布到类生产环境，执行Sanity测试用例及解决方案测试，生成测试报告，决策服务化/产品化发布(见图13-4)。

13.2.8　微服务的运维

微服务运维主要的挑战是分布式系统所引入的复杂性。从全系统的视角来看，完成某一功能的

微服务有多个运行实例，而且同一时间，这些实例的软件版本也不一定相同，于是服务能力也有差异。另外，由于系统是分布式系统，在多个运行实例之间，数据的一致性的要求也会带来复杂性。

微服务的运维基于分布式系统的运维演进，可以划分为业务上线、例行运维、故障发现和处理几个环节的运维子系统。

根据微服务的特点，特别需要注意的运维子系统还有微服务的配置变更管理、微服务的可用性监控、微服务的调用链和依赖关系管理、微服务的部署上线升级管理、微服务的拨测等。图13-5针对上述的运维组织了信息流和控制流，便于说明微服务运维整体系统的流转。

微服务的上线和变更是比较频繁的，为了避免各种可能的出错和配置数据的错误及丢失，需要匹配相应的配置管理方案，表13-3中描述了必

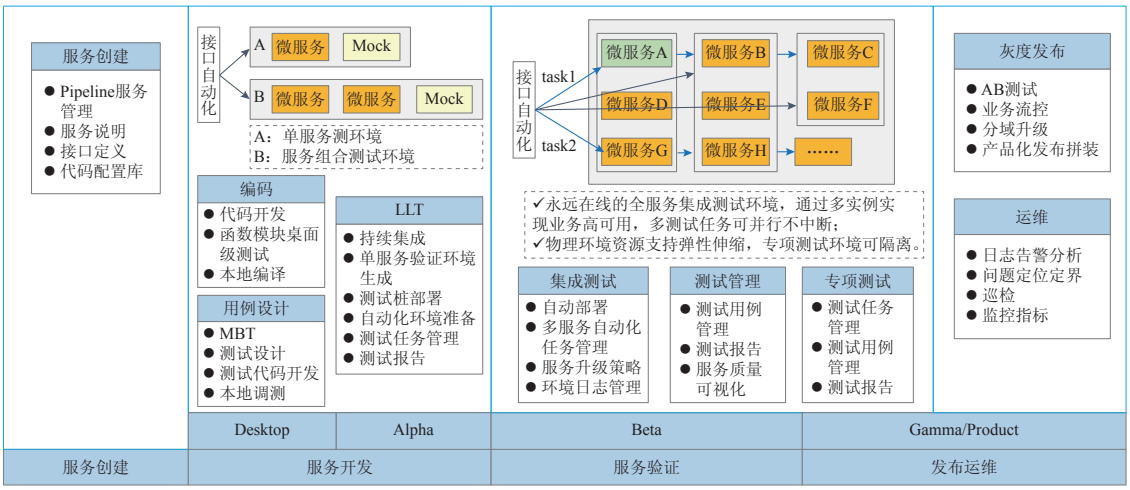


图13-4　微服务的测试

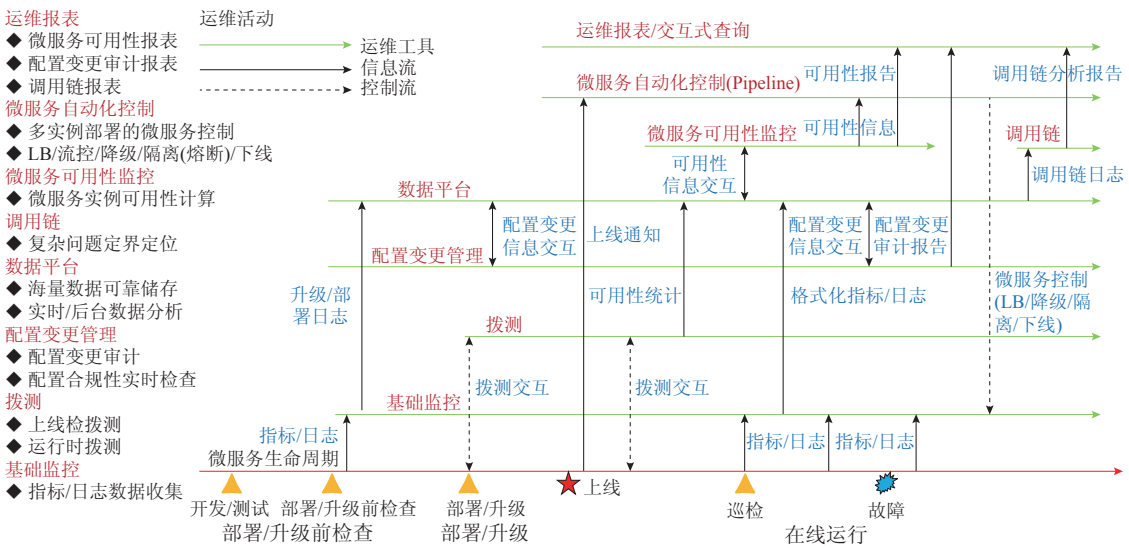


图13-5　微服务的运维

要的配置变更管理场景和方案。

表13-3 必要的配置变更管理场景和方案

场景	场景描述	变更管理方案	效果
软件Bug	软件Bug导致配置错误	跟踪资源的配置变更事件，进行实时的配置合规性检测	跟踪资源的配置变更事件，进行实时的配置合规性检测
软件Bug导致配置错误	人为差错引入的配置错误		
升级	升级导致配置不一致，进而引起组件/微服务间协作故障	配置合规性规则中引入关联关系	及时发现升级带来的配置一致性问题
静默损坏	软件Bug、误操作、故障等原因导致的配置静默损坏	后台周期性收集配置快照，并进行周期性的配置合规性检测	及时发现配置的静默损坏，避免发生故障
故障定位	复杂流程中，某环节的资源配置错误导致故障，难于定位	对关键资源定义时间窗，在时间窗内记录配置变更历史序列，并提供友好的查询界面	将指标劣化/事故等异常事件与配置变更进行时间轴上的关联分析，快速发现配置变更导致的故障
资源关联	资源间存在各种关联关系，导致配置变更引入的问题传播到其他组件/微服务，难于定位根因	配置合规性规则中引入关联关系	及时发现配置依赖导致的复杂问题根因

13.2.9 微服务监控管理

微服务的监控，是对于微服务的服务状态，也就是对外服务能力SLA指标达成度的管理方式。

针对微服务的监控主要包含以下监控的内容：

- 微服务的实例运行状态；
- 微服务的实例资源占用量周期性统计；
- 微服务的服务能力SLA指标周期性统计；
- 微服务的API调用统计。

其中对于微服务的服务能力SLA指标的监控主要是：服务消费者在请求服务的过程中会监控服务提供者是否可用，并将状态反馈给监控程序；自动部署程序也会将部署成功、失败、启动、停止等信息反馈给监控程序，资源消耗等情况也会送达监控程序，监控程序判断各项指标是否达到SLA中约定的阈值，当达到阈值时，触发自动报警。

对于微服务的API调用统计，主要是识别API使用的频繁度，并且可以对于常用的API等由研发人员进一步进行性能调优，以优化整体的系统性能。

监控的工具在开源业界有相当多成熟的工具可以直接选用，例如Nagios或者Zabbix，都是应用非常广泛的开源工具。

13.2.10 微服务的故障定位

在微服务的软件架构中，系统的业务逻辑是由多个微服务组合来实现的，这是一个去中心化的分布式软件架构。网络因素(带宽、时延、抖动、丢包)会带来数据的一致性的实现问题。在系统功能发生故障时，如何快速地进行问题的定界和定位，对于系统的运维能力而言，是一个巨大的挑战。

基于业务流程的微服务调用链以及日志分析，是进行微服务的故障定位和定界常用的技术手段。

调用链，又称为调用树，是业务调用的一次过程，这个过程包含了多次的服务接口调用。只要针对接口调用进行采样和打点，并将所有的打点数据连接为一个链条，就会产生一个调用链。通过对调用链中的数据进行分析，可以确定发生

故障的阶段和流程点，通过结合系统的监控和日志记录，可以确定故障发生的原因。

调用链的生成是由多个交互过程组织起来的，如图13-6所示。

调用链指一整个调用链，也就是一组交互的集合，一个交互树。交互代表单个客户的一次请求+服务的响应，跨两个实体，是调用链的一部分。一次交互中涉及的实体操作包含下面四个过程：CS(客户端发送)、SR(服务端接收)、SS(服务端发送)、CR(客户端接收)。打点记录的数据，就在这四个过程中生成，并通过交互标识、父标识、跟踪标识进行关联。故障发生的时候，采用关联信息进行分析以确定故障原因。

13.2.11 微服务的日志管理

云化软件系统微服务化架构模式本质是基于分布式系统之上的应用架构模式。随着微服务的数量增加，系统运行节点数量或者微服务运行实例的增加，对于分散在各个分布式节点的日志提取、采集、分析，或者将日志用于故障定位、调用链分析等，都存在非常大的困难。

日志是软件开发领域常用的信息记录和留存手段，本文不再累述。需要补充说明的是，在微服务化条件下，系统的分布式特点会造成日志生成源比较分散，为了便于问题的定位，对于日志的管理必须实现微服务的日志聚合，以及基于日志聚合的日志后分析。日志中记录的信息，主要包括所需要的标识、日志生成点的故障信息、日志生成的时间、生成的实体等。

开源领域比较好的日志聚合和后分析工具为Splunk或者ELK。

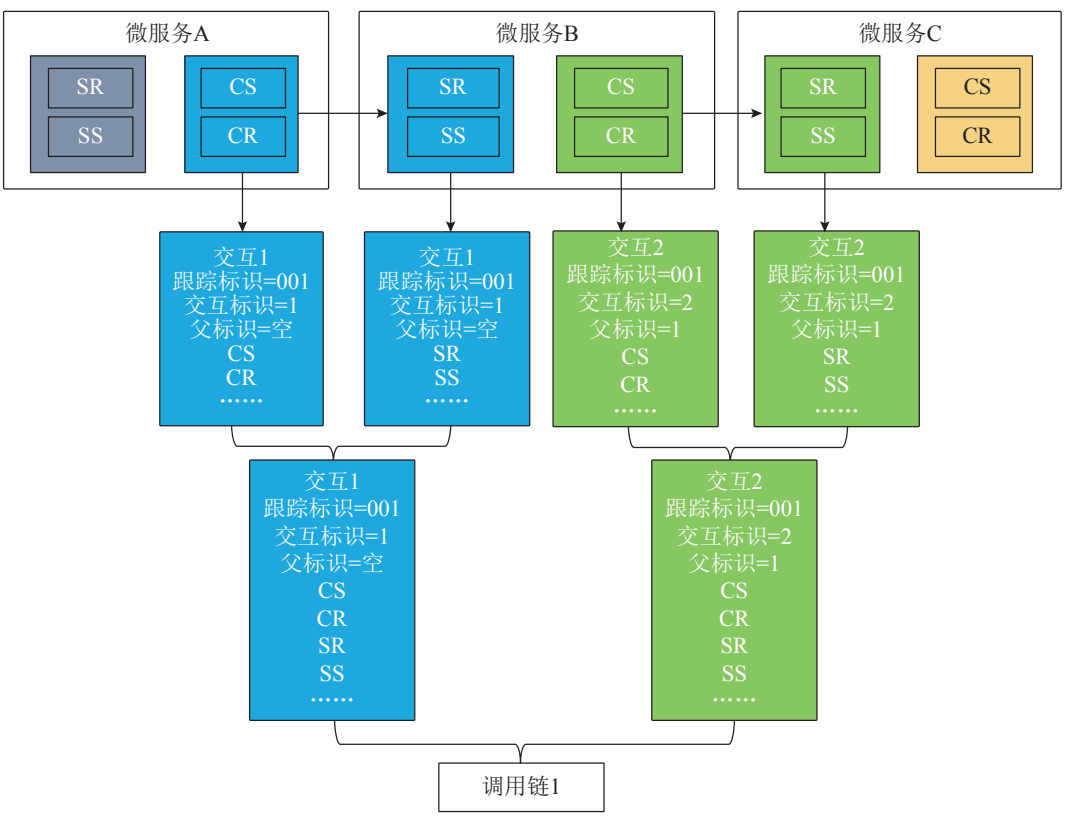


图13-6 微服务调用链

13.3 支撑敏捷开发与上线的微服务CI/CD工具链

13.3.1 微服务的持续集成

传统的软件架构将系统划分为多个模块，但并不注重模块间接口的契约化，同时产品集成的间隔时间很长。于是每次产品的集成都好像是在苦难地行军，集成阶段经常会出现系统功能不可用，既有特性丢失等严重问题，也有修复问题，即一个模块的问题修复，需要其他的模块开发团队停下来等待。

为了解决这类问题，软件开发中引入了持续集成的概念，也就是用小步快跑的方式频繁地开展软件系统的集成，避免开发时间跨度过大造成集成难度的累积。对于微服务架构的产品，往往需要数量众多的微服务一起配合完成一项复杂的特性和功能，在这种条件下，持续集成的必要性就更高。

微服务的每个开发单元颗粒度较小，虽然服务的数量众多，但是每个开发单元业务功能单一，接口契约化，这些条件便于验证测试，服务间松耦合，支持独立部署和升级，有利于开展持续集成，而且能够将整个开发验证流程通过工具的支撑实现自动化，从而极大地提升研发的效率。这也是微服务化架构在复杂软件系统情况下具有强大生命力的原因。图13-7说明了微服务的开发团队以持续集成为核心，构筑的端到端的流水线开发模式。

需要注意的是，一个服务的端到端的环节，虽然通过微服务化架构实现了独立自治的开发，但是对每一个开发团队而言，持续集成实际上加快了软件开发测试修复上线的循环速度。我们也可以看到，从服务的诞生到服务的发布和上线，都是一个相当复杂和完整的业务流程，这个业务流程与传统单一软件产品的开发流程相比，复杂度并没有本质的降低，为了能够保证每个服务团队的开发活动标准化，开发活动能够高效开展，这就需要工具来支撑作业体系的自动化。

13.3.2 微服务研发的工具链

微服务的架构模式真正落实到软件的生产流程中获得软件研发效率和质量的收益，需要针对软件的开发环节，持续地开展工具、流程和自动化的建设。一个复杂的软件系统，按照业务逻辑实现了软件架构的解耦，并保证每个微服务具备了可以独立开发上线部署，仅仅是微服务架构模式的第一步。这就好比人类社会每个独立的个体一样，要驱动如此复杂的研发模型，还需要制定

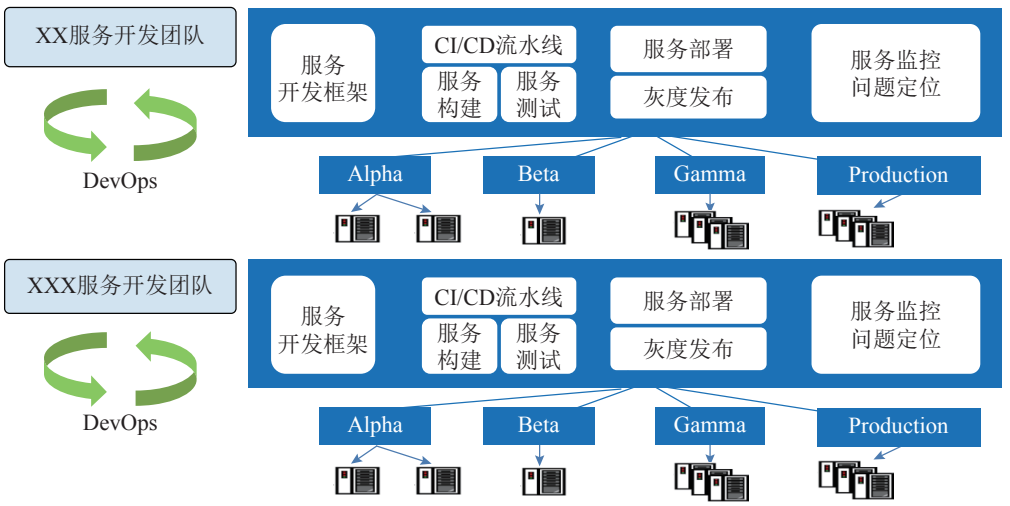


图13-7 微服务持续集成流水线

详细的分场景和阶段的游戏规则，而工具链就是游戏规则的承载。

从微观上看，微服务的开发模式，就是我们建议的持续集成、持续发布的模式；从宏观上看，需要链接端到端的需求到产品上市，并且对于大型的商业化软件系统，还需要有项目管理、配置管理等支撑系统予以辅助。

根据业界的软件开发实践，微服务的工具集按照领域分为三个大的部分，如图13-8所示。

1. 微服务开发流水线

其主要是微服务研发团队进行日常软件开发所使用的工具集，包括了构建、测试、部署、验证的环节。我们可以理解为一个程序员日常会使用的研发工具集。比较成熟的微服务开发团队还会根据自身的业务需求，将微服务的通用性的代码整理为开发框架，进一步与流水线集成起来，让微服务的程序员聚焦于微服务业务逻辑的实现。

2. 研发作业的管理支撑工具

其主要是指进行大型软件项目管理的项目管理工具，包括项目计划、任务分配、任务跟踪，软件的配置管理、版本发布支撑、集成测试(例如测试用例的管理系统)等支撑类的工具集。

3. 客户界面的运维管理工具

其主要是指软件交付到客户局点之后的运维工具(例如监控、告警、日志、服务拨测、服务状态管理等)、客户的问题管理回馈、客户的需求管理工具。

对于微服务研发的工具链的获取方式，一般推荐优先选用开源已有的工具集，例如下文介绍了云计算领域OpenStack的开源社区所采用的持续集成和持续发布的工具集；对于开源工具无法支持的工具需求，可以考虑购买一些商业公司的微服务开发工具集，或者自己进行开发。

13.3.3 典型开源社区的工具集介绍

OpenStack社区是一个大型的开源合作项目，除了OpenStack本身是开源软件之外，社区也大量采用了其他的开源软件来构筑了一个高效的协作开发和持续集成的平台。我们注意观察系统中的代码的流向：从贡献者提交代码的补丁(Pacth)到Gerrit，然后是准备测试环境资源，准备测试系统并运行Tempest测试套(OpenStack社区规定的门槛用例集)，最终报告测试结果并归档测试结果(Test Artifacts)。

当贡献者提交补丁给OpenStack的一个项目，

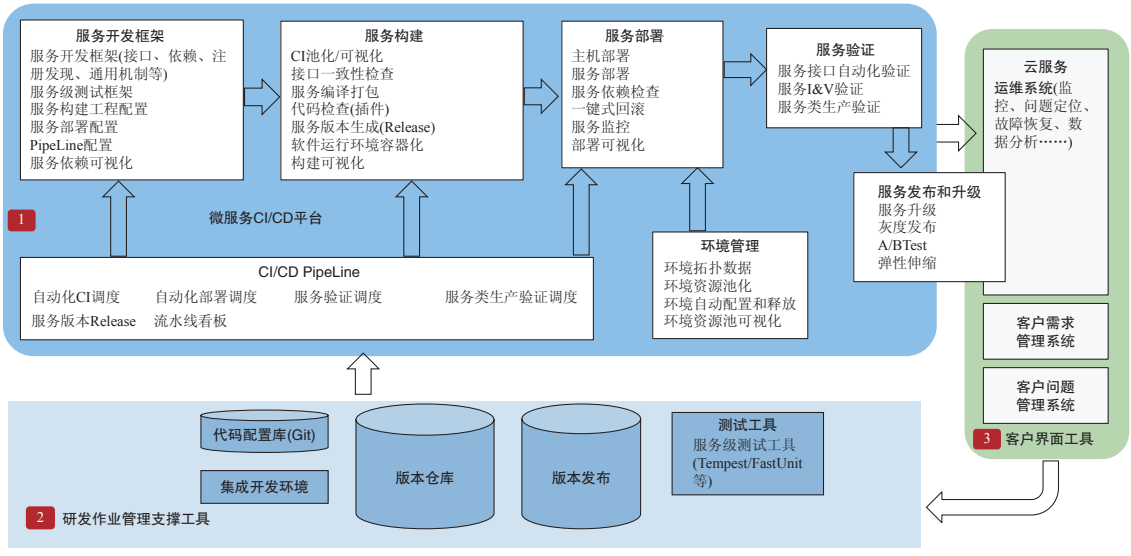


图13-8　微服务研发工具链

他将推送代码给Git服务器，该服务器被Gerrit管理。代码要进入到OpenStack的社区主干，需要经过代码检视和测试的活动。

通常情况下，贡献者使用一个叫Git-review的Git插件提交检视。Gerrit控制哪些用户或群组可以提交代码、合并代码和管理代码库。当贡献者推送代码到review.OpenStack.org，Gerrit会创建一个变更设施并放入所提交的补丁。补丁提交者和其他贡献者都可以向变更设施添加新的修改，Gerrit收集并记录补丁所有的变化。

当代码被推送给Gerrit的时候，将会触发一系列任务来测试所提交的代码。Jenkins是执行和管理这些任务的服务器，它是一个Java应用程序，具有可扩展架构，可通过插件在其基础服务上增加功能。

Jenkins里的每个任务是单独配置的。在后台，Jenkins将配置信息存储在其数据目录的xml文件里。你可以用Jenkins管理员身份手动编辑Jenkins任务，然而在OpenStack CI系统这样大的测试平台，手工操作几乎是不可能的，并且很容易配错。幸运的是，有个叫Jenkins Job Builder(JJB)的辅助工具，它可以通过一组YAML文件和作业模板来生成XML配置文件。

一般做系统集成的时候需要针对一个完整的项目真实环境来安装软件和运行测试。这种类型的测试称为集成测试，它确保了对一个组件的修改不会导致其他组件的失败。这对于如OpenStack这种子系统间存在重要依赖的、复杂的、多项目系统尤其重要。

对这些复杂的Jenkins任务来说，另外一组工具被添加进来提供用于测试的资源管理。Nodepool为Jenkins master提供虚拟机实例，以用来运行复杂的、独立的Jenkins 任务。

Nodepool创建的实例运行Jenkins slave，所以这些实例能够与Jenkins CI master服务器进行通信，每个nodepool创建的虚拟机实例上都运行一个prepare_node.sh的脚本，这个脚本仅仅是把OpenStack Infra配置项目Git clone到节点上，然后安装Puppet，运行一系列的Puppet清单——根据节点的类型设置节点。节点类型有裸节点、虚拟机等。

图13-9为OpeanStack社区工具集。

13.4 面向微服务的DevOps研发运维组织变革

任何的研发组织都需要完成从需求到产品的转换，传统的基于瀑布模式的开发组织形态，通

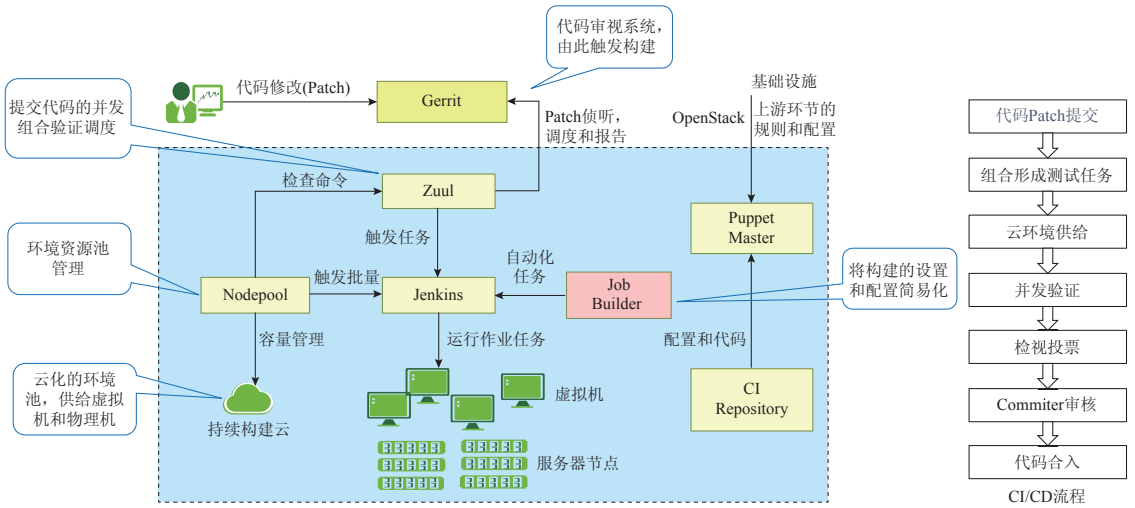


图13-9　OpenStack社区工具集

过将架构、设计、开发、测试等专业化，并且以专业化组建团队，例如架构师在一个小组，设计师在一个小组，开发人员一个小组，测试人员也是独立的团队，通过这样的方式来强调专业技能的共享，以追求单业务做到极致的方式来提升组织的效率。在微服务的条件下，这种模式需要打破。

业界开展微服务研发的实践非常多，相信每个公司，乃至于每个微服务团队，都会有自己的组织形态，根本无法完全对比其中的优劣。针对微服务的研发模式和团队组建，以当前的业界实践的成熟度，也仅仅满足推荐和讨论一些注意事项的水平。

微服务通常会设置产品经理的角色，用于管理微服务的研发项目，进而会考虑对于微服务产品的竞争力规划、演进等。

微服务的开发团队是一个自治型的团队，需要的是一个全功能的完整团队，包括了微服务的需求、设计、开发、测试、资料等软件研发环节涉及的几乎所有的角色(见图13-10)。这是与传统模式有很大不同的地方。

微服务的开发团队同时负责服务上线后的运维活动，这就回答了针对服务质量改进的原生动力的来源问题。这同时也让服务本身功能是否满足客户需求得到快速回馈。

综合以上，我们谈到面向微服务的组织变革的时候，核心的观点还是给微服务的研发组织决定自己服务开发的权利，也让其对于最终的开发结果能够负责。减少决策链条，让最终客户来评价功能的优劣和质量的好坏，这是微服务组织调整能够提升软件的开发效率的原因之一。

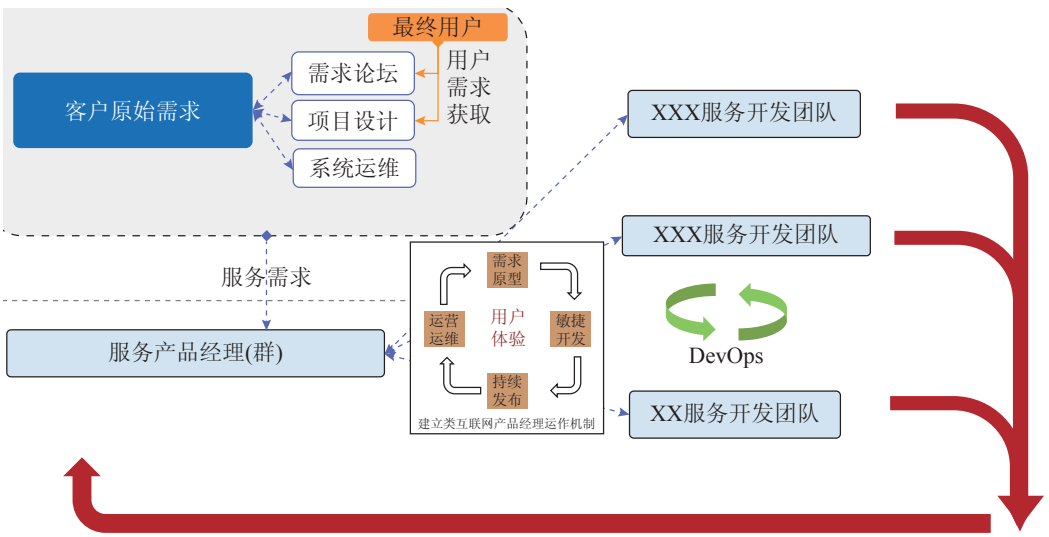


图13-10　微服务研发组织的设置示意图

第 14 章

云安全架构与应用实践

云计算的“安全”实际上与我们通常意义上的“安全”概念上是基本一样的。

以我们最关注的个人安全为例进行类比可以看到，个人安全威胁可能会涉及个人财产损失、个人人身伤害、个人隐私侵犯等。针对这些可能的个人安全威胁，人类很早就开始穿上衣服来遮挡隐私；然后盖上房子修建围栏来防止野兽的侵扰；然后房子又加上锁来防止同类来偷窃；到了现代又发明了保险柜，防止小偷撬锁；后来又把钱存到银行防止小偷连保险柜一起偷走……而进入云计算时代，个人安全威胁以新的形式以及更大的广度在扩大，例如将照片、视频、通讯录、私人日记放在了网盘的个人空间里(云)，一些担心就会随之而来，如网盘信息会不会被人贩卖？托管在公有云上的客户关系管理系统的客户数据信息会不会被竞争对手拿到？业务系统信息是否可能被篡改？可以看到，在云计算时代，为了保障个人安全，我们只盖个房子、加个锁是远远不够的，而且房子怎么盖，锁加在哪里都成了问题。这时，我们就需要使用系统工程的方法来构筑个人安全的防范问题。

对应地，在云计算领域，使用系统工程的方法来建立和完善云计算体系的安全，这便是“端到端云安全架构”。

14.1 端到端云安全架构

14.1.1 云计算中主要安全威胁

概括地讲，云计算的主要安全威胁仅4个字，即“天灾人祸”：

- “天灾”泛指各种不可抗力，例如地震、火灾、水灾等；
- “人祸”指某些个人或者组织为了实现其利益而对其他人或其他组织尽其所能地进行入侵、攻击、窃取、破坏等行为。“人祸”是云计算的主要安全威胁。

云计算体系可能遭受的威胁来自多个层次。

一、网络层次

网络层次的威胁具体如下。

- 数据传输过程中的数据私密性与完整性存在威胁：目前多数用户仍使用HTTP方式(未加密)而非HTTPS(加密)访问云资源。一些敏感信息如密码，可能被窃取。
- 更容易遭受网络攻击：云计算必须基于随时可以接入的网络，便于用户通过网络接入，方便地使用云计算资源。云计算资源的分布式部署使路由、域名配置更加复杂，更容易遭受网络攻击，比如DNS攻击和DDoS攻击。对于IaaS，DDoS攻击不仅来自外部网络，也容易来自内部网络。
- 资源共享风险：云计算的共享计算资源带来的更大的风险，包括隔离措施不当造成的用户数据泄漏、用户遭受相同物理环境下的其他恶意用户攻击；网络防火墙、IPS虚拟化能力不足，导致已建立的静态网络分区与隔离模型不能满足动态资源共享需求。

二、虚拟化层次

虚拟化层次的威胁具体如下。

- Hypervisor的安全威胁：Hypervisor为虚拟化的核心，可以捕获CPU指令，为指令访问硬件控制器和外设充当中介，协调所有的资源分配，运行在比操作系统特权还高的最高优先级上。一旦Hypervisor被攻击破解，在Hypervisor上的所有虚拟机将无任何安全保障，直接处于攻击之下。
- 虚拟机的安全威胁：虚拟机动态地被创建、被迁移，虚拟机的安全措施必须相应地自动创建、自动迁移。虚拟机没有安全措施或安全措施没有自动创建时，容易导致接入和管理虚拟机的密钥被盗、未及时打补丁的服务(FTP、SSH等)遭受攻击、弱密码或者无密码的账号被盗用、没有主机防火墙保护的系统遭受攻击。

三、数据与存储威胁

数据与存储威胁具体如下。

- 静态数据的安全威胁：静态数据可以加密保存，如简单对象存储业务，用户通过客户端加密数据，然后将数据存储到公有云中，用户的数据加密密钥保存在客户端，云端无法获取密钥

并对数据进行解密。

✍ 数据处理过程的安全威胁：数据在云中处理，数据是不加密的，可能被其他用户、管理员或者操作员获取。

✍ 剩余数据保护：用户退租虚拟机后，该用户的数据就变成剩余数据，存放剩余数据的空间可以被释放给其他用户，如果数据没有经过处理，其他用户可能获取到原来用户的私密信息。

四、身份认证与接入管理

云计算支持海量的用户认证与接入，对用户的身份认证和接入管理必须完全自动化，为提高认证接入管理的体验，需要云简化用户的认证过程，比如提供云内所有业务统一的单点登录与权限管理。

表14-1描述了云计算管理员、用户和黑客对云管端所造成的威胁。

表14-1 不同客户的云安全威胁

模块	威胁源	管理员	用户	黑客
端	TC	非法操作：如利用TCM与TC间正常的升级通道，植入木马控制TC 伪造非法TCM：控制TC 权限滥用：对TC USB端口管理不合理	恶意用户：仿冒其他用户登录，破解密码 非法TC：非法TC具有获取VM数据的能力	伪造TCM管理员 TCM漏洞攻击 TC被非法破坏：植入木马，非法获取VM数据
	SC	权限滥用	数据泄露到本地，如截屏	SC漏洞攻击
管	网络		截获其他用户密码 PC等设备绕过安全网关	常见网络攻击
云	虚拟机	非法重置用户密码 误挂卷 利用虚拟机备份文件非法恢复用户数据 虚拟机自然损坏	用户非法登录：弱口令或口令保管不善 用户虚拟机被篡改 攻击相邻虚拟机，如ARP攻击 非授权访问相邻虚拟机 攻击虚拟化平台 利用虚拟化资源从事非法活动，如攻击外网 虚拟机迁移过程中安全策略失效	类似PC的常见攻击
	虚拟化层	管理员非法登录：利用弱口令或口令保管不善 权限滥用：在缺少三权分立场景下易发生关键操作无法回溯 破坏镜像文件，植入木马 管理员权限扩大化：如节点间采用互信，则获取单节点权限即可控制整朵云 非法获取敏感信息，如数据库口令 非法监视用户虚拟机流量 非法获取用户密码	还原出前一用户硬盘数据 还原出前一用户内存数据	利用租用的虚拟机攻击虚拟化平台 利用租用的虚拟机攻击虚拟化管理平台，如利用OS/Web漏洞 虚拟机迁移中截获用户数据

以上列出的仅仅是来自某个个人的可能威胁，还未列出来自某个组织(叫“团伙”可能更容易理解)的威胁，比如某个大型公司内部的某个小团伙对某大型公司进行破坏活动，那么这个破坏力会成倍增长，因为这个团伙成员可能来自管理层、IT、内外部用户和黑客。

14.1.2 端对端的安全架构

上面分析了云计算环境中所面临的各种威胁，不同级别的威胁，其相应的安全保障措施是不同的。企业或组织对安全级别要求越高，这个端到端安全架构的价值也就越大，安全架构之下的具体技术保障手段也更严谨和丰富，如图14-1所示。

结合云计算的风险，端对端云计算安全架构有以下几个核心部分。

一、“云终端”安全

采取USB端口策略管控、禁止直接访问内置存储、补丁和升级关联等。

其实质是采用专用的TC(Thin Client，瘦客户端)终端，这种TC终端用于显示云中心的图像，是一个专用的嵌入式系统。TC这种瘦客户端很像家里安装的机顶盒，看到的节目(TC对应的内容)都来自电视台(TC对应的云平台)。和机顶盒不同的是，机顶盒使用遥控器进行操作，TC可以直

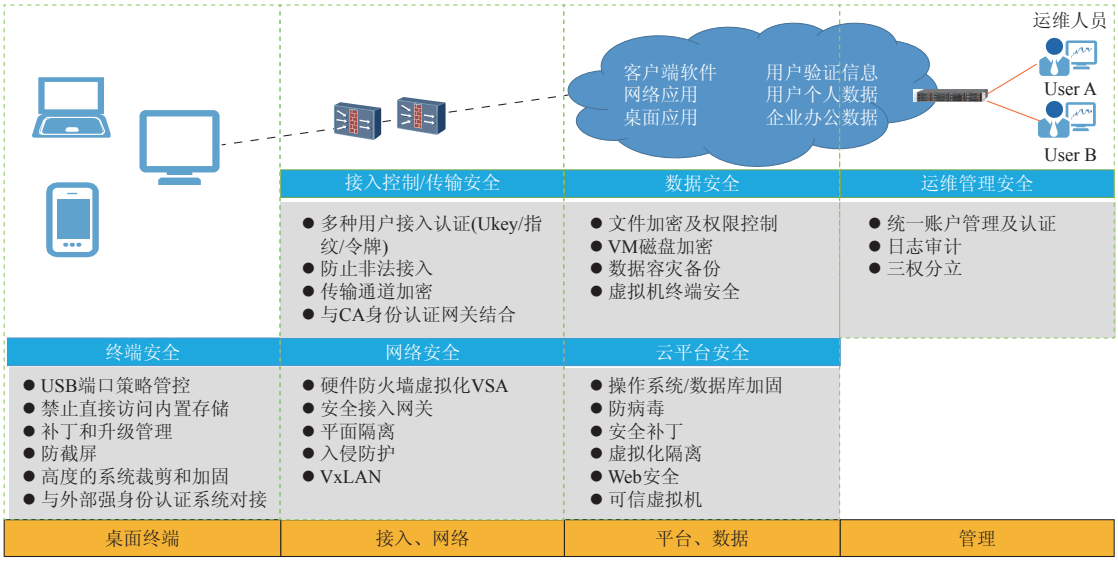


图14-1　云计算E2E安全架构

接连接键盘鼠标，像以前使用PC那样(跟PC使用体验基本一致，只是受到了更严格的安全管制)。采用这种TC终端能够实现较好的安全性，包括以下几个方面。

1. BIOS仅从内置存储引导

TC的BIOS只保留从内置存储引导的方式，没有保留其他的引导方式，如USB引导、PXE(Preboot Execute Environment，预启动执行环境，网络远程启动)引导。

2. OS安全

TC是一个精简的、封闭的Linux/WES7/XPe操作系统。

✍ 禁止存储类设备使用：TC的操作系统从驱动层可禁止USB存储设备的使用，不包含任何USB存储的驱动。包括U盘、USB光驱等在内的USB存储设备无法在TC本地使用。

✍ 禁止直接访问内置存储：操作系统的TC在本地没有暴露内置存储访问接口，用户只能通过系统提供的程序间接访问，这样能有效地避免系统文件被破坏。

✍ 禁止任意安装程序：操作系统的TC在本地不提供安装软件的接口，用户或者其他第三方人员无法自行安装任何软件。如果需要在TC上安装软件，则只能通过TCM(专门的TC管理系统)将软件包下发到TC上，然后在TC上进行安装。

简单理解“云终端安全”的目标，那就是，对安全可能构成威胁的操作基本都被禁止。

二、“管道”安全

管道安全包括接入安全和网络安全。

接入安全包括多种方式的接入认证(如USB KEY/指纹/令牌等)、防非法接入(基于网络IP参数和用户ID)、传输通道加密、与CA系统对接等。

网络安全包括防火墙的访问控制(这里的防火墙除传统防火墙以外，还包括虚拟化的防火墙等多种类型)、安全接入网关(用于与TC进行认证，建立安全通道，甚至提供负载均衡的功能)、网络平面隔离(提供业务网、管理网和存储网的三网隔离)、网络入侵防护(如提供防拒绝服务攻击功能)、基于VxLAN的网络VLAN划分等。

理解“管道安全”的重要性，有一个形象的例子。换位思考一下，一伙劫匪，他们为了钱，是抢银行容易呢，还是抢运钞车容易？答案肯定是运钞车，因为运钞车在路上，再怎么防护，也没有银行防护得严密。在银行内部，通往金库是最薄弱的环节，也是通路。那么为了安全防护，

在外部，要考虑运钞车本身的坚固性、押解人员以及行进路线的安全性；在银行内部，通往金库的道路上要层层设卡，设立各种监控和密码设施。

三、“云”安全

云安全包括云端数据安全，这是云计算重点需要考虑的安全内容。云计算中的数据太过集中会造成用户的担忧，信息资产的集中存储也是网络攻击的重点所在。“云”安全包括云数据安全和云平台安全两部分。

云数据安全的解决方案包括文件加密及权限控制、VM的磁盘加密、数据的容灾备份、虚拟机终端安全。

云平台安全包括云操作系统和数据库的安全加固、防病毒、安全补丁、虚拟化隔离、Web安全、可信虚拟机。

其中的虚拟化隔离和可信虚拟机(借助可信芯片)是下面重点介绍的内容。

四、“管理”安全

管理安全包括统一账号管理及认证，日志审计，三权分立等内容。

下面对云计算中的重要安全技术及实现方案进行介绍。

14.2 可信计算TPM/vTPM

TPM英文全称为Trusted Platform Module，即可信赖平台模块；vTPM的英文全称为Virtualizing the Trusted Platform Module，即虚拟化可信赖平台模块。

在云计算环境中，用户失去了对虚拟机的完全控制，导致用户无法信任虚拟机环境，这成为了用户部署云计算的一个重要障碍。因此采用可信计算和虚拟化技术的结合成为热点。基于TPM/TCM/TXT可信硬件技术，在云平台上开发可信虚拟机原型系统，可实现虚拟机的可信启动、可信运行、虚拟机可信迁移等，从而为虚拟机构建一个可信的计算环境，提升用户对虚拟机环境的信任度。

虚拟机的可信是高等级信息系统中的关键点，主要包括：

- 防止节点的软件配置、虚拟机的OS被黑客攻击篡改；
- 虚拟机防止被其他虚拟机攻击、嗅探；
- 防止虚拟机被恶意的人员启动和利用；
- 防止虚拟机迁移到不被认可的非安全Host主机上；
- 防止Dom0的恶意管理员利用特权账户发起对Dom0的监控与攻击(Dom全称Domain，是CPU虚拟化内核调度中的一个名词，简单理解，CPU中有很多Domain，每个Domain中存放一个操作系统软件，Domain0中放的是虚拟化软件内核操作系统软件，Domain0中的操作系统软件有比其他Domain更高的CPU指令执行特权，恶意管理员可能利用这个特权去监控其他Domain中的操作系统)。

虚拟机可信计算技术是当前虚拟化环境中的技术发展热点，也是技术难点，一般通过可信芯片技术来实现。技术实现方案如图14-2所示。

图14-3是实现虚拟机的可信启动原理。

用户的VM初始内存的数据(不变的部分)、软件安装/卸载/运行信息、OS信息、磁盘数据以Hash的方式做了运算，成为度量的基数。该基数保持在TPM芯片中，后续的度量值与之比较。(简单理解，Hash是一种算法，能够对每个目标状态以唯一数值的方式进行标注，相当于你家养的牲畜都被它盖上戳，多了少了都可以看出来，有狼披着羊皮混在你的牲畜圈也能看出来，或根本就混不进去)。

其实现的功能是：

- 可信VM的OS状态受到TPM的保护，防止OS被修改；
- 可信VM中软件的安装/卸载/运行需要得到用户授权，并可防止软件被修改；
- 监控Dom0和管理员对可信VM的内存访问，防止VM内存数据被获取。

用于云计算当中的TPM芯片可能有几种安装

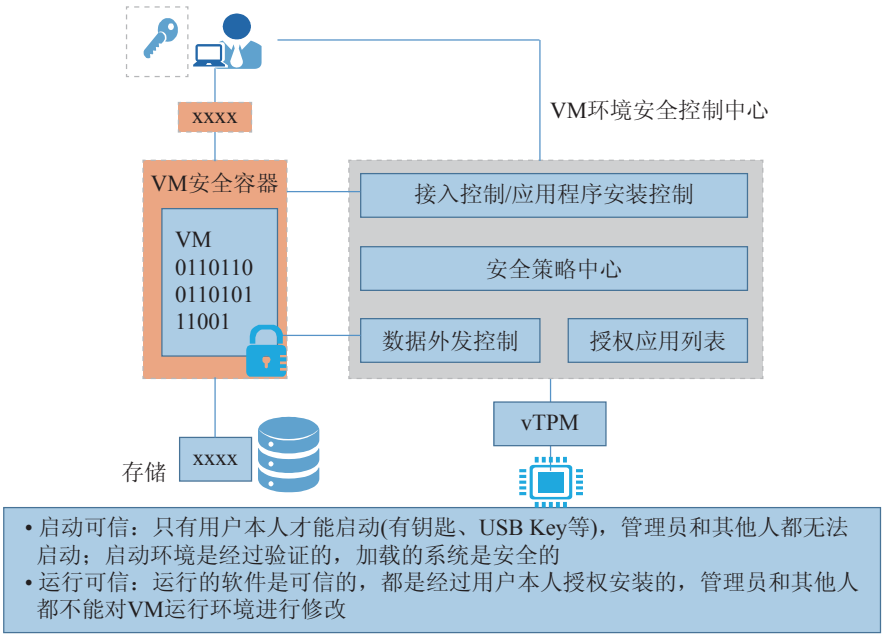


图14-2　可信芯片技术原理

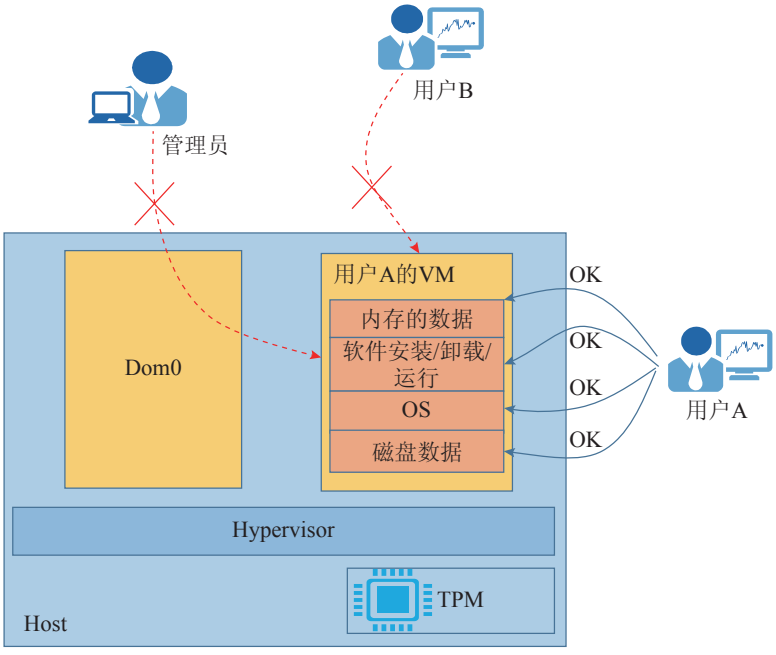


图14-3　可信启动原理

形式：内嵌到主板之上，或以插卡的形式安装到主板上的预留插槽。比如华为的Tecal系列服务器，全部采用TPM插卡形式，TPM插卡作为服务器的选配件提供。

下面几个小节介绍一下TPM可以实现的主要功能。

14.2.1 TPM功能1：主机启动/静态度量

基于上述可信技术可以实现许多非常有用的功能，举例来讲，云计算环境中主机的安全是云计算安全的基础，如果主机操作系统被篡改或植入恶意代码，那么在主机之上运行的云操作系统都变得不安全。基于TPM技术，可以把合法的主机信息保存起来，在系统启动过程中进行有效判断(见图14-4)。

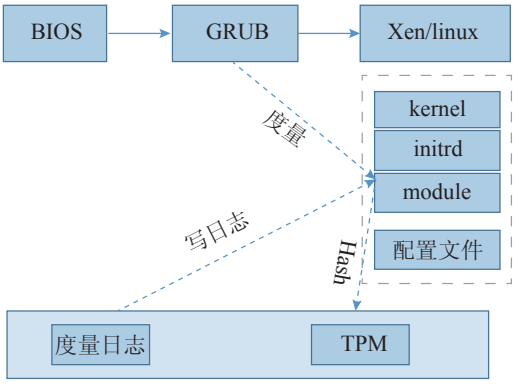


图14-4 主机静态度量流程

我们将实现这一功能的过程描述如下：

- BIOS将控制权交给GRUB；
- 在kernel、initrd、module命令中加载操作系统内核镜像及其他模块，GRUB引导操作系统启动，同时进行度量，将hash值扩展到TPM PCR10(PCR、Platform Configuration Register、平台配置寄存器)，将记录写入度量日志；
- 度量配置文件列表中的文件，将hash值扩展到TPM PCR10，将记录写入度量日志；
- 执行boot命令，操作系统启动；
- 在操作系统启动后，传值模块自启动，将主机静态度量信息传出。

14.2.2 TPM功能2：虚拟机的静态度量

同样，基于TPM，可以对虚拟机的合法性进行保护。主要原理是在虚拟机启动前，依据云管理服务器下发的静态度量配置文件构造度量文件列表，然后将虚拟机镜像mount放在指定目录，mount的镜像分为Windows和Linux两种操作系统。

根据度量文件列表依次对查找到的文件进行SHA-1度量操作，得出160位hash值，同时将每一个文件的hash值进行迭代操作，得到最终的度量结果，将该度量结果与各个文件的hash值构成的日志文件上传给上层，为虚拟机镜像是否遭到篡改提供依据。

14.2.3 TPM功能3：主机动态度量

TPM除了用于对操作系统及其虚拟机系统的静态信息的保护以外，在系统运行中，还能根据系统的运行状态信息进行动态保护。主机动态度量分为两个模块：Xen度量模块和Dom0度量模块(见图14-5)。

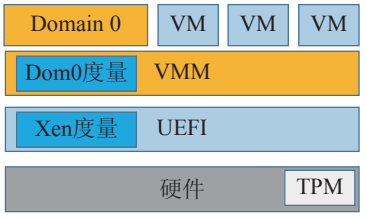


图14-5 主机动态度量原理

Xen度量模块部署在UEFI中，利用SMM机制进行实时度量保护，主要在UEFI(新型UEFI，全称“统一的可扩展固件接口”，英文为Unified Extensible Firmware Interface，是一种详细描述类型接口的标准)中进行实现，借助CMOS来传递度量的地址和范围，可以进行手动和定时两种触发方式，当度量程序被触发后，会对CMOS中指定的数据进行度量，并将度量值扩展到TPM(Trusted Platform Module)的PCR(Platform Configuration Register)中。手动触发是通过管理员在Domain0中输入触发指令来触发SMI Measurement Handler，而定时触发是由UEFI中的定时协议提供的功能，到达指定时间会自动执行SMI Measurement

Handler；SMI Measurement Handler会从CMOS中读取指定的度量范围和地址并完成度量操作，最后将度量结果生成日志信息。

Dom0度量模块部署在Xen中，主要对上层Domain0中的GDT(Global Descriptor Table)、IDT(Interrupt Descriptor Table)和模块及驱动进行度量，并负责主动获取在内存中Xen的度量值，然后生成度量日志文件。

14.2.4 TPM功能4：VM动态度量

如图14-6所示，本部分为动态度量结构图，分为调度模块和度量模块。度量模块负责对虚拟域进行度量，调度模块负责Domain-0与度量模块的通信。

通信包括两个方面：第一，将度量模块度量得到的度量值传送到Domain-0的Client端；第二，接收Client端发来的命令请求，并将请求发给度量模块，对度量模块的行为进行触发。度量分为定时度量和手动度量两种。手动度量由Client触发，度量模块接收到手动度量请求后，完成度量行为，并将度量值回发给Client端。定时度量的触发由度量模块中的timer完成。每一个虚拟域对应一个timer，Client端能够设置每个timer的时间片以及是否进行度量。对于不同的虚拟域，其有不同的度量点，如表14-2所示。

表14-2 虚拟域的不同度量点

虚拟机类型	度量点
Windows	GDT IDT SSDT驱动
Linux-hvm	GDT IDT KERNEL内核模块

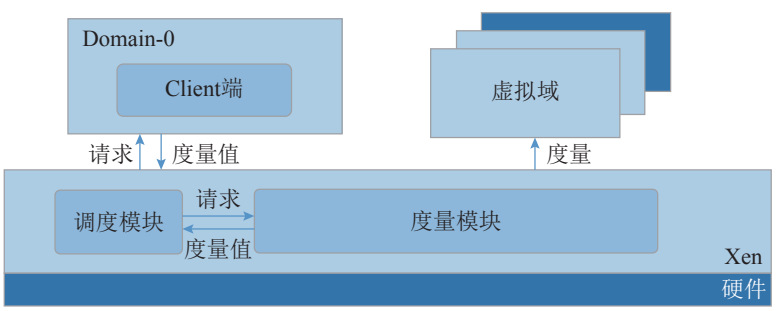


图14-6 VM动态度量流程

14.2.5 TPM功能5：远程证明

远程证明是对平台做全面的度量，向远程通信方证明自身运行环境是可信的。远程证明是一个综合完整性校验和身份鉴别的过程，同时向验证者提供一份可信的平台状态报告。TPM是报告的可信根，能够保证对当前完整性度量值做可信的报告。

远程证明是通过远程证明协议来实现的，如图14-7所示。

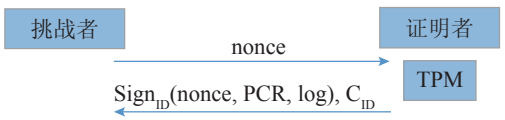


图14-7 远程证明协议

注：PCR英文全称为Platform Configuration Register，即平台配置寄存器PCR。

我们基于远程证明协议(见图14-8)，根据不同的场景，收集客户端相应的度量值和度量日志，开始进行完整性检查。服务器产生一个随机数，发送给客户端。配备TPM的客户端对度量值用SHA1算法求出哈希值，再用生成的签名私钥对其进行签名，形成完整性报告，报告包括主机名、随机数、度量值、哈希值、签名及一些平台相关信息等。最后，将完整性报告和度量日志一起发送给服务器。服务器检查随机数、验证哈希、验证签名，通过比较签名值和基线值判断平台是否可信。

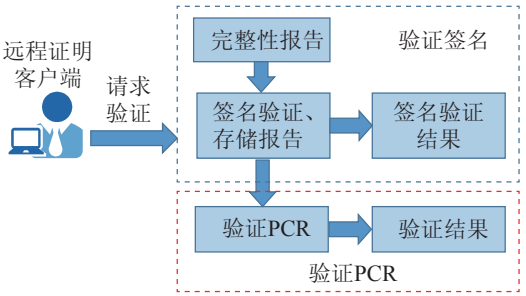


图14-8　远程证明流程

14.3　虚拟机的安全隔离

14.3.1　vCPU调度隔离安全

Hypervisor普遍采用了硬件辅助虚拟化技术(以下以服务器最常用的Intel CPU硬件虚拟化技术VT-x介绍)，VT-x将CPU的操作扩展为两个forms(窗体)：VMX Root Operation(根虚拟化操作)和VMX Non-root Operation(非根虚拟化操作)，VMX Root Operation设计供Hypervisor使用，其行为与传统的IA32并无特别不同，而VMX Non-root Operation则是另一个处在VMM(Virtual Machine Monitor)控制之下的IA32环境。所有的forms都能支持所有的Ring0~Ring3共4个特权级，这样在VMX Non-root Operation环境下运行的虚拟机就能完全地利用Ring 0等级(见图14-9)。

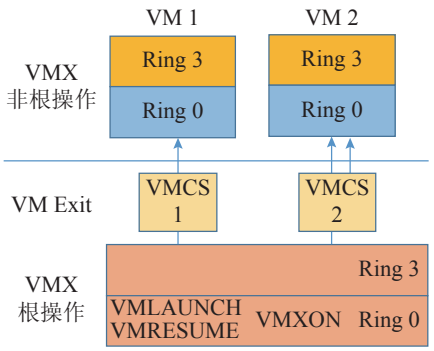


图14-9　CPU的隔离原理

Root Operation环境和Non-root Operation环境之间可以切换。如果虚拟机要进行一些特权操作，比如I/O访问、对控制寄存器的操作、MSR的读写指令等，此时就进入了Root Operation环境，当处理完这些特权操作后，将重新进入Non-root Operation环境继续虚拟机的执行。从Root Operation环境到Non-root Operation环境叫VM Entry，反之称为VM Exit。同时，VMM可以通过执行VMXON 和VMXOFF 指令打开和关闭VT-x。

在Root Operation环境和Non-root Operation环境之间进行切换时，有一个虚拟机控制结构VMCS(Virtual Machine Control Structure)进行控制和管理，当虚拟机被创建时，VMM就同时为每一个vCPU创建一个VMCS，这个数据结构可以决定哪些操作会触发VMExit进入Root Operation。进入到Root Operation后，Hypervisor取得控制权，通过读取VMCS中的VM Exit Information Fields得到发生VM Exit的原因，在vmx-vmexit-handler函数中开始执行相应处理。Hypervisor通过选用几种调度算法(如credit、BVT等)，把物理CPU合理地分配给虚拟机使用。虚拟机获得一个时间片后，在这个时间片内连续运行它的逻辑CPU，时间片消耗完后，Hypervisor会调度下一个虚拟机运行。Hypervisor正是通过VMCS结构灵活的监控虚拟机的运行，环境切换时硬件自动保存、恢复各自的状态，做到了CPU的完全隔离。

14.3.2　内存隔离

虚拟机通过内存虚拟化来实现不同虚拟机之间的内存隔离。内存虚拟化技术在客户机已有地址映射(虚拟地址和机器地址)的基础上，引入一层新的地址——“物理地址”。在虚拟化场景下，客户机OS将“虚拟地址”映射为“物理地址”；Hypervisor负责将客户机的“物理地址”映射成“机器地址”后，再交由物理处理器来执行(见图14-10)。

内存虚拟化共涉及三个内存地址。

- 机器内存地址：真实的机器地址，即地址总线上出现的地址信号。
- 虚拟机物理内存地址：经Hypervisor抽象、虚拟机看到的伪物理地址。
- 应用程序内存地址：客户机OS提供给其

应用程序使用的线性地址空间。

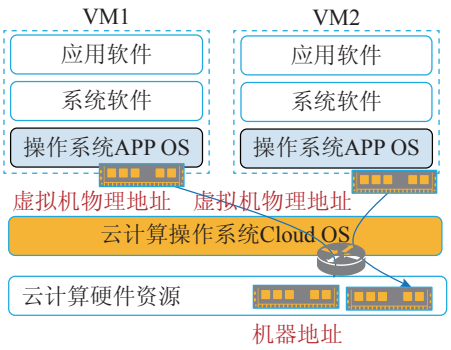


图14-10　内存隔离原理

虚拟机物理内存地址与应用程序使用的内存地址之间的映射关系是由APP OS维护的，即虚拟机的OS维护虚拟机上的应用程序之间的内存分配和调度。

Cloud OS(Hypervisor)管理虚拟机使用的虚拟物理内存(Physical Memory)和真实机器内存(Machine Memory)之间的对应关系，即P2M转换，这种对应关系是一一对应的，每个虚拟机都有一张P2M表。虚拟机访问内存时，通过P2M表转换，只能访问到分配给它的内存，不能访问没有分配给它的内存，从而实现各虚拟机之间的内存隔离。

普通MMU(Memory Management Unit)只能完成一次虚拟地址到物理地址的映射，在虚拟机环境下，经过 MMU转换所得到的“物理地址”并不是真正的机器地址。若需得到真正的机器地址，必须由VMM介入，再经过一次映射才能得到总线上使用的机器地址。如果虚拟机的每个内存访问都需要VMM介入，且由软件模拟地址转换的效率是很低下的，其几乎不具有实际可用性，为实现虚拟地址到机器地址的高效转换，现普遍采用的思想是：由VMM根据映射f和g生成复合的映射fg，并直接将这个映射关系写入MMU。

Hypervisor采用的内存硬件辅助虚拟化技术是用于替代虚拟化技术中软件实现的“影子页表”的一种硬件辅助虚拟化技术，其基本原理是：VA-> PA-> MA，两次地址转换都由CPU硬件自动完成(软件实现内存开销大、性能差)。Hypervisor采用VT-x技术的页表扩充技术Extended Page Table(EPT)，首先VMM预先把客户机物理地址转换到机器地址的EPT页表设置到CPU中；其次客户机修改客户机页表无须VMM干预；最后，地址转换时，CPU自动查找两张页表完成客户机虚拟地址到机器地址的转换。通过使用内存的硬件辅助虚拟化技术，在客户机运行过程中无须VMM干预，去除了大量软件开销，内存访问性能接近物理机。

14.3.3　内部网络隔离

Hypervisor提供虚拟防火墙——路由器(VFR，Virtual Firewall - Router)的抽象，每个客户虚拟机都有一个或者多个在逻辑上附属于VFR的网络接口VIF(Virtual Interface)。从一个虚拟机上发出的数据包，先到达Domain 0，由Domain 0来实现数据过滤和完整性检查，并插入和删除规则；经过认证后携带许可证，由Domain 0转发给目的虚拟机；目的虚拟机检查许可证，以决定是否接收数据包。

14.3.4　磁盘I/O隔离

虚拟机所有的I/O操作都由Hypervisor截获处理，Hypervisor保证虚拟机只能访问分配给该虚拟机的物理磁盘，实现不同虚拟机硬盘的隔离。

14.3.5　用户数据隔离

Hypervisor采用分离设备驱动模型实现I/O的虚拟化。该模型将设备驱动划分为前端驱动程序、后端驱动程序和原生驱动三个部分，其中前端驱动在DomainU中运行，后端驱动和原生驱动则在Domain0中运行。前端驱动负责将DomainU的I/O请求传递到Domain0中的后端驱动，后端驱动解析I/O请求并映射到物理设备，提交给相应的设备驱动程序控制硬件完成I/O操作。换言之，虚拟机所有的I/O操作都会由VMM截获处理；同时，系统对每个卷定义不同的访问策略，没有访问该卷权限的用户不能访问该卷，只有卷的真正

使用者(或者有该卷访问权限的用户)才可以访问该卷，VMM保证虚拟机只能访问分配给它的物理磁盘空间，从而实现不同虚拟机硬盘空间的安全隔离。

14.4 虚拟化环境中的网络安全

虚拟化平台的网络通信平面划分为业务平面、存储平面和管理平面，且三个平面之间是隔离的。存储平面与业务平面、管理平面间是物理隔离；管理平面与业务平面间是逻辑隔离。通过网络平面隔离保证管理平台操作不影响业务运行，最终用户不能破坏基础平台管理。

网络平面隔离如图14-11所示。

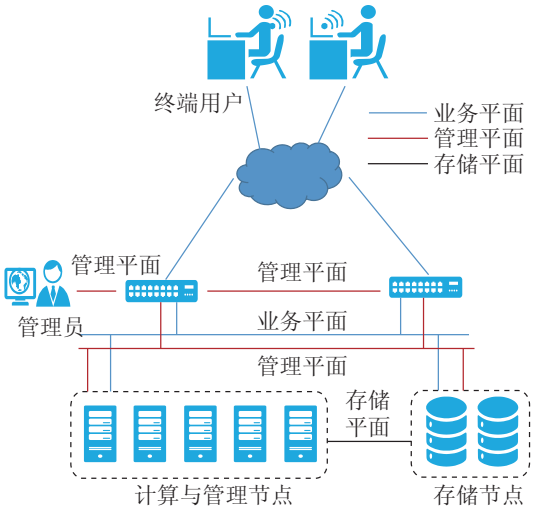


图14-11 网络平面隔离

✍ 业务平面：为用户提供业务通道，为虚拟机虚拟网卡的通信平面，对外提供业务应用。

✍ 存储平面：为iSCSI存储设备提供通信平面，并为虚拟机提供存储资源，但不直接与虚拟机通信，而通过虚拟化平台转化。

✍ 管理平面：负责整个云计算系统的管理、业务部署、系统加载等流量的通信。

14.4.1 虚拟交换机及防Arp攻击

ARP(Address Resolution Protocol，地址解析协议)，负责将某个IP地址解析成对应的MAC地址。ARP攻击就是通过伪造IP地址和MAC地址实现ARP欺骗，通过在网络中产生大量的ARP通信量使网络阻塞，该攻击主要存在于局域网网络中，通过木马程序发起攻击。

在ARP的防攻击中，虚拟交换机EVS(Elastic Virtual Switch)起到了重要作用。那么什么是虚拟交换机呢？图14-12是EVS的架构图。

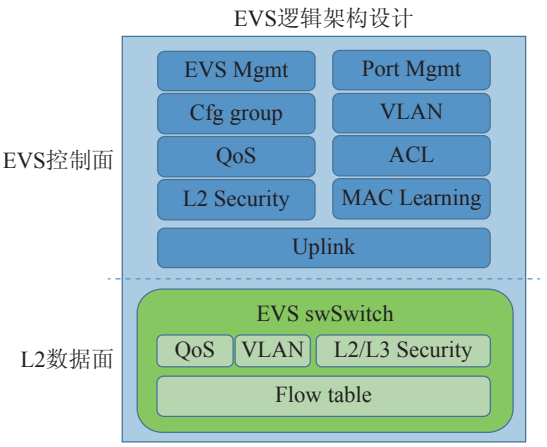


图14-12 单机EVS架构图

单机EVS功能模块职责表如表14-3所示。

表14-3 单机EVS功能模块职责表

功能模块名称	职责
EVS Mgmt	EVS对象管理，支持创建多个EVS对象
Port Mgmt	管理虚拟交换机的端口资源，根据VM申请给其分配虚拟交换机端口资源，该端口最终和VM的vNIC对应
Cfg group	提供配置组能力，方便用户对端口做批量配置
VLAN	提供标准的802.1Q能力
Qos	提供基本的Qos能力，实现对虚拟机的带宽保护、流量限速、简单的流量整形等
ACL	基于五元组，根据ACL规则对包转发引擎进行规则配置，对转发的报文根据规则进行接收/丢弃处理
L2 Security	提供二层的IP/MAC防欺骗的安全能力
Mac Learning	提供用户态的Maclearning学习能力

(续表)

功能模块名称	职责
UpLink	管理上行链路和上行端口，连接上行链路和物理网卡，用户申请上行端口时必须连接到指定的Uplink链路上，通过VLAN能力的属性设置保证VM的流量从该上行端口(上行链路)上收发。Uplink还会管理普通nic、inic、enic等各种资源，并提供inic或者enic能力下的网络直通对应的资源分配能力
EVS.swSwitch	Host的二层虚拟交换能力，保证二层报文的跨VM、跨host的交换能力；提供QoS、VLAN、L2 /L3 Security等功能，并针对每条数据流建立对应的flow table表项，后续数据流基于对应的flow table表项进行快速交换

14.4.2 IP/MAC防欺骗功能设计

弹性EVS实现的一个重要网络安全功能是IP/MAC的防欺骗功能。

其功能包括：

- 截获DHCP报文，进行解析；
- 对开启IP/MAC防欺骗功能的虚拟网卡侧过来的报文进行非法DHCP报文过滤；
- 根据开启IP/MAC防欺骗功能的虚拟网卡的DHCP ack报文生成对应IP/MAC防欺骗 DB条目，用于IP报文源地址防欺骗和ARP报文防欺骗。

14.4.3 VLAN

通过虚拟网桥实现虚拟交换功能，虚拟网桥支持VLAN Tagging功能，实现VLAN隔离，确保虚拟机之间的安全隔离。

虚拟网桥的作用是桥接一个物理机上的虚拟机实例。虚拟机的网卡eth0、eth1等称为前端接口(front-end)。后端(back-end)接口为vif，连接到Bridge。这样，虚拟机的上下行流量将直接经过Bridge转发。Bridge根据mac地址与vif接口的映射关系做数据包转发。

Bridge支持VLAN Tagging功能，这样分布在多个物理机上的同一个虚拟机安全组的虚拟机实例可以通过VLAN Tagging对数据帧进行标识。网络中的交换机和路由器可以根据VLAN标识决定是否对数据帧路由和转发，也可以依据VLAN标识提供虚拟网络的隔离功能。

如图14-13所示，处于不同物理服务器上的虚拟机通过VLAN技术可以划分在同一个局域网内，同一个服务器上的同一个VLAN内的虚拟机之间通过虚拟交换机进行通信，而不同服务器上的同一VLAN内的虚拟机之间通过交换机进行通信，确保不同局域网的虚拟机之间的网络是隔离的，不能进行数据交换。

14.5 云数据安全

数据是云计算的核心，数据的安全包括数据的机密性保护(张三的数据不能被李四读取)、完整性保护(数据不能被篡改)、数据操作审计等内容。下面介绍云数据安全的一些解决方案。

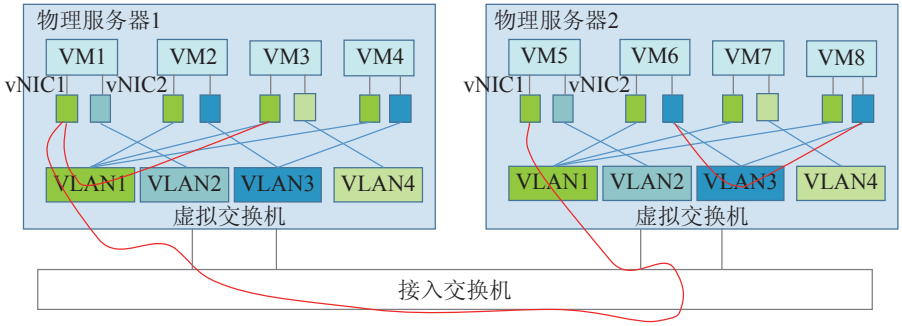


图14-13 VLAN组网图

14.5.1 云存储加密与用户数据安全

一、云计算虚拟磁盘加密方案

云计算用户最大的担心就是个人数据安全，由于云计算将数据集中管理，数据不再由用户自己管控，而是被云计算运营企业、云计算管理员控制，如何保证这些用户的数据不被偷看、泄露？如何保证用户数据不会在不同用户间交叉，造成泄露？数据加密是其中最可信的解决方案，从密码理论上讲，只要用户的密钥没暴露，即使数据丢失也能保障信息不外泄(见图14-14)。

二、云计算数据安全解决方案

对虚拟机数据盘进行全盘加密，加密过程通过降低CPU消耗来避免影响系统效率。采用加密机、PKI证书、对称密钥三级密钥机制提供对用户虚拟磁盘的高强度、高安全性加密。 数据安全防护系统与第三方CA无缝对接，提供高安全、高可靠的密钥管理服务。通过屏蔽云平台差异，兼容所有类型的Hypervisor，实现与云业务管理系统松耦合关系以及数据安全防护系统的独立部署。

14.5.2 用户数据安全有效保护

数据安全是保障数据中心安全的重点。为了保障用户的数据安全，需要从数据隔离、访问控制等多个方面采取措施。

一、用户卷访问控制

系统对每个卷定义不同的访问策略，没有访问该卷权限的用户不能访问该卷，只有卷的真正使用者(或者有该卷的访问权限)才可以访问该卷，每个卷之间是互相隔离的。

二、存储节点接入认证

存储节点采用标准的iSCSI进行访问，并且支持询问握手认证协议(CHAP，Challenge Handshake Authentication Protocol)认证功能。CHAP协议可通过三次握手，周期性校验对端的身份。CHAP认证可在初始链路建立时、完成时以及在链路建立之后重复进行。通过递增改变的标识符和可变的询问值，可防止来自端点的重放攻击，限制暴露于单个攻击的时间。CHAP认证功能可以提高应用服务器访问存储系统的安全性。

存储系统启用CHAP认证以后，应用服务器侧也必须启用CHAP认证，同时在存储系统中把应用服务器的信息加入到存储系统的合法CHAP用户，只有经过CHAP认证通过以后，才能连接到存储系统并存取数据。

三、剩余数据删除

当用户把卷卸载释放后，系统在把该卷进行重新分配之前，会对该卷进行数据格式化，

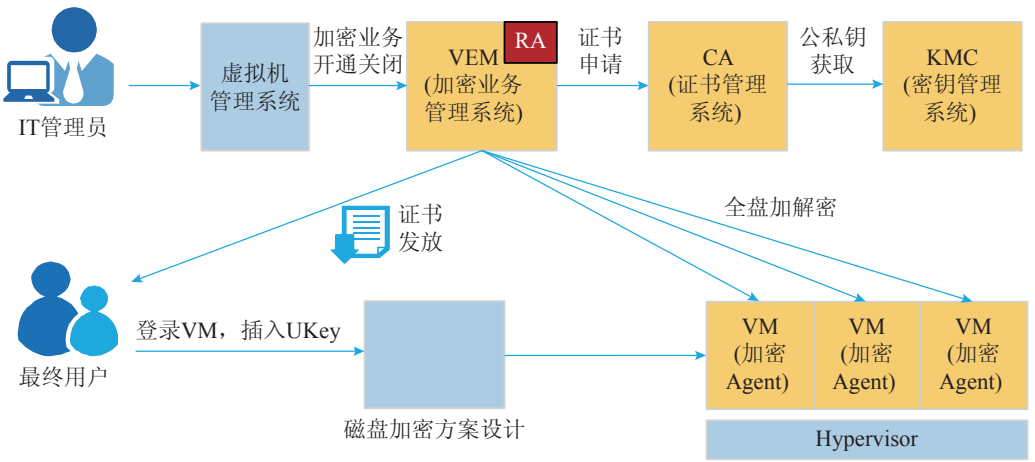


图14-14 云计算虚拟磁盘加密系统(VES)

以保证该卷上的用户数据的安全性。

✍ 存储的用户文件/对象删除后，对应的存储区进行完整的数据擦除，并标识为只写(只能被新的数据覆写)，保证不被非法恢复。

四、数据备份

✍ 云数据中心的数据存储采用多重备份机制，每一份数据都可以有一个或者多个备份，当数据因存储载体(如硬盘)出现故障的时候，不会引起数据的丢失，也不会影响系统的正常使用。

✍ 系统对存储数据按位或字节的方式进行数据校验，并把校验的信息均匀地分散到阵列的各个磁盘上。阵列的磁盘上既有数据，也有数据校验信息，数据块和对应的校验信息会存储于不同的磁盘上，当一个数据盘损坏时，系统可以根据同一带区的其他数据块和对应的校验信息来重构损坏的数据。

五、IPSAN保险箱技术

✍ 存储系统遭遇意外掉电时，采用数据保险箱技术保证数据的安全性和完整性。数据保险箱技术即从系统中的某几块硬盘上划分出一定区域，用来专门存放因突然掉电而尚未及时写入硬盘的Cache数据和一些系统配置信息。当系统外部供电中断时，则通过内置电池或外置UPS供电，使得Cache中的数据能够写入数据保险箱。当外部电力恢复时，控制器再将数据从数据保险箱中读回到Cache，继续完成对数据的处理。

14.6 公有云、私有云的安全组

虚拟化带来的最大威胁是虚拟机间资源未完全隔离，云计算提供了同一物理机上不同虚拟机之间的资源隔离，避免虚拟机之间的数据窃取或恶意攻击，保证虚拟机的资源使用不受周边虚拟机的影响。终端用户使用虚拟机时，仅能访问属于自己的虚拟机的资源(如硬件、软件和数据)，不能访问其他虚拟机的资源，保证虚拟机隔离安全。

实现原理如图14-15所示。

用户可以根据虚拟机的属性灵活定义安全

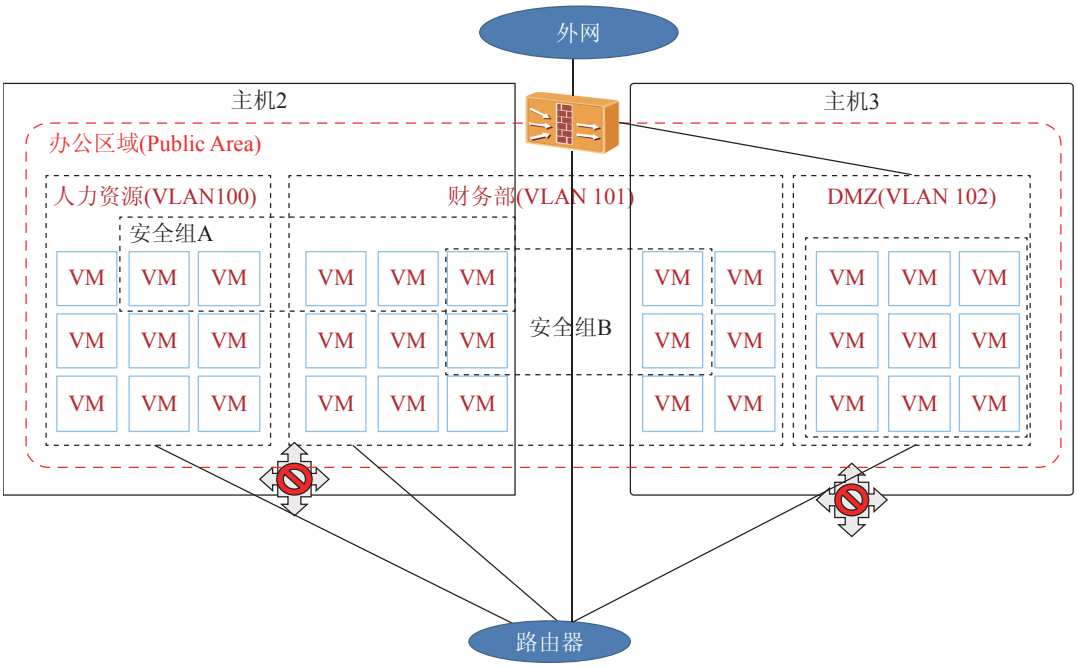


图14-15 安全组原理

组，包括IP、MAC、VLAN、Subnet等元数据。安全组是具有同等安全要求的一组虚拟机。安全组可以由一个或多个VM，也可以由一个或多个VLAN组成。安全组的策略可以细化到VLAN甚至每台VM。每个安全组对应一个安全域，可以基于安全组定义自己的访问控制规则，实现域内和域外的安全隔离。安全组间的访问控制在Hypervisor层实现，每个host上部署一个，不占用额外的虚拟机资源，不存在安全检测引流导致的流量迂回问题。

安全网关控制器(Security Gateway Controller，SGC)是实现VM隔离功能的核心组件，它部署在Hypervisor，主要功能如下：

- 当SGC上不存在从源VM到目的VM的安全组规则时，它和目的端SGC进行安全组规则协商，以决定是否允许VM之间的通信；
- 当SGC本地的安全组成员表、安全组规则表发生变化时，需要通知相关SGC做相应的更新；
- SGC接收安全组规则协商请求消息，根据本地的安全组成员表和安全组规则表，动态生成安全组会话表；
- SGC接收安全组成员、安全组规则表(增、删、改)通知消息，以及VM迁移事件，并做相应处理；
- SGC接收Hypervisor发来的VM在线、离线事件，并做相应处理。

14.7 云安全管理

14.7.1 日志管理

云计算带来了成本降低、效率提高等一系列好处的同时，由于计算、存储的集中，对管理维护提出了更高的安全要求，以保障基础设施的安全运行。

系统支持集中的日志收集和存储，满足各种审计要求，如分级保护。

一般的云计算平台均支持以下三类日志。

(1) 操作日志

操作日志记录操作维护人员在管理节点进行的管理维护操作，包括用户、操作类型、客户端IP、关键参数、操作时间、操作结果等内容，存放在管理节点的数据库中。审计人员可通过OMS Portal导出和查看操作日志，定期审计操作维护人员在管理节点进行操作，及时发现不当或恶意的操作。操作日志也可作为抗抵赖的证据。

(2) 运行日志

运行日志记录各节点的运行情况，分为debug、info、warning、error 4个级别，优先级依次递增，可由日志级别来控制日志的输出。

各节点的运行日志通过日志管理组件统一汇总、过滤成高级别日志(warning、error级别)和完整日志(包括所有已设置输出级别的日志包)。高级别日志定期汇总到日志服务器统一存放。完整日志在本地存放，并支持通过脚本方式上载指定节点、时间段的完整日志到日志服务器。

运行日志包括级别、线程名称、运行信息等内容，维护人员可通过查看运行日志，了解和分析系统的运行状况，及时发现和处理异常情况。

(3) 黑匣子日志

黑匣子日志记录系统严重故障时的定位信息，主要用于故障定位和故障处理，便于快速恢复业务。其中计算节点产生的黑匣子日志汇总到日志服务器统一存放，而管理节点、存储节点产生的黑匣子日志在本地存放。

云计算系统支持集中的日志收集和存储，如图14-16所示。

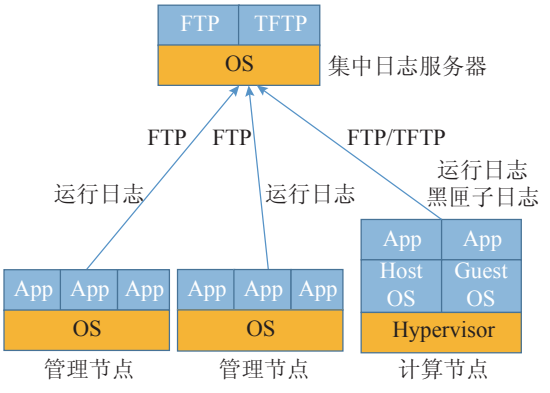


图14-16 集中日志管理

在各节点部署日志收集客户端，实时收集本地产生的运行日志、黑匣子日志，通过配置日志收集服务器，实现将日志数据过滤成高级别日志和完整日志。高级别日志定期汇总到集中日志服务器。完整日志通常存在本地节点，可通过脚本把指定节点、指定时间段的完整日志汇总到日志服务器进行集中管理，主要管理方法如下。

一、安全告警管理

安全告警是指当系统侦测到违背安全策略的事件行为时，将与安全事件相关的一些信息上报给安全告警管理，管理员根据这些信息对违背安全策略的行为进行及时处理，排除安全隐患。安全告警上报的内容包含告警的来源、告警产生的时间、告警产生的原因、服务提供者、服务使用者、告警级别、事件类型等信息。

二、日志分类管理

通过查看日志可以了解系统的运行情况和操作记录，用于用户行为审计和问题定位。日志分为操作日志和运行日志，操作日志与系统安全相关。操作日志记录了用户对系统所做的操作以及操作的结果，用于跟踪和审计。记录操作日志，可以快速定位系统是否遭受恶意的操作和攻击。

三、日志备份

系统需要定期备份操作日志，并提供“定时周期性备份方式”备份操作日志；日志备份成功后，系统会自动删除已备份的日志。

在界面上查看日志时，采用Https方式进行传输。查看日志时采取的安全措施如下。

✍ 根据管理员的权限定义日志查看范围。超级管理员能查看所有日志，但普通管理员仅能查看自己的操作日志。任何人员不能在界面上修改或删除日志。有查询权限的人才能导出日志。

✍ 日志格式：各系统针对操作日志提供多个字段，详细记录外界用户在系统上执行的具体操作、用户信息、操作时间等信息，便于管理员根据操作日志识别有风险的操作或已导致问题的危险操作，并采取合理的防卫措施，提高系统的安全性。

14.7.2 账户和密码安全

硬件设备及系统存在初始默认的账号和密码，主要是用于维护硬件设备及系统。建议管理员首次登录就修改默认密码，且修改时需满足密码复杂度要求，同时建议管理员定期修改密码，确保密码不泄漏。

系统中各类账户的密码加密、设置和修改原则如表14-4所示。

表14-4 各类账户的密码加密、设置和修改原则

项目	原则
密码首次设置原则	首次登录系统时，需要修改系统默认密码，密码修改方法请参见账户密码维护； 密码设置需满足密码策略表要求
密码加密原则	所有密码加密存放； 密码不以明文形式在界面显示
密码修改原则	只有通过认证的用户，才能修改密码； 修改密码时，必须通过旧密码校验； 密码达到最长有效期后，用户再次登录时，系统强制用户更改密码； 密码需要定期修改，对于管理员密码，建议最长180天修改一次
密码策略表修改原则	系统安装时生成默认的密码策略表； 密码策略表只有系统管理员有权修改； 修改密码策略表后，根据原有密码策略表设置的密码能够正常登录

14.7.3 分权分域管理

通过对管理员区分权限，对被访问的数据区分权限，限制管理员访问系统的范围，保证系统的安全。管理员分权分域管理模型遵循美国国家标准与技术研究院(NIST，The National Institute of Standards and Technology)标准的基于角色的访问控制(RBAC，Role Based Access Control)模型。

一、三权分立

底层采用强制访问控制措施，在系统启动的时候就自动产生系统管理员、审计员和操作员三类角色，而且每类角色可以延伸新的角色，系统

管理员不能删除其他角色，其行为将会被审计员所审计。

二、分权

“分权” 指区分“操作权限”，它由“角色”进行控制。一个“角色”可拥有一个或多个不同的“操作权限”，一个“用户”可拥有一个或多个不同的“角色”。通过绑定“用户”和“角色”，实现“用户”和“操作权限”的绑定。如果一个“用户”拥有多个“角色”，其拥有的“操作权限”是多个“角色”拥有的“操作权限”的并集。产品支持灵活的设置角色，并灵活赋予角色拥有的权限。

三、分域

“分域” 指区分“数据管理范围”。

四、基础设施及虚拟桌面分权分域管理

创建多个管理员用户，并支持对各管理员用户进行操作权限和数据权限的管理。“超级管理员”可根据企业自身的业务需要，在“桌面云业务维护系统”系统中添加“业务管理员”，通过“业务管理员”管理和维护企业的桌面云资源。“超级管理员”可赋予“业务管理员”不同的角色，用以定义用户可执行的操作权限。“超级管理员”可配置“业务管理员”，用以定义用户可执行操作的范围。这里提到的“超级管理员” 不一定是一个人，出于安全需要，可能是以多个人联合完成的，比如5个人，每个人拥有一段独立的密码，5个人分别输入密码才能让超级管理员权限生效，而超级管理员的操作则在5个人的监督下完成。

14.8 安全即服务

云计算在公有云的场景下，安全不但是要保证云平台自身的安全，同时安全也需要作为一种服务能力提供给租户。 公有云场景下，云服务商和租户之间在安全方面首先需要划分安全责任界面，业界通用的公有云责任界面如图14-17所示：

云服务的核心在于服务。公有云服务商提供可感知的租户级安全特性，帮助租户提高租户资产的安全性。租户可以方便地在界面完成与安全相关的配置操作和防护报告。

公有云服务商通过两个界面提供安全服务：租户控制台和超市。租户控制台通常提供与云平台基础设施相关的安全服务，比如Anti-DdoS即服务、云主机防护即服务、 与负载均衡一起提供的Web应用防火墙、对象存储加密等。 超市的定位提供更多的第三方安全产品，满足不同租户的安全需求。

14.9 云安全应用实施案例

桌面云与身份认证系统的对接案例

某单位以前采用传统PC办公，终端数量众多，维护工作量大，分级保护终端软件安装和维护成本巨大，使用和维护极其不便；即使在此情况下，泄密风险也时刻存在。因此应重视桌面云技术解决安全问题，实现安全和方便地办公。

云桌面管理包括桌面管理和虚拟化管理两个部分。虚拟桌面管理软件提供高性能且可靠的桌面

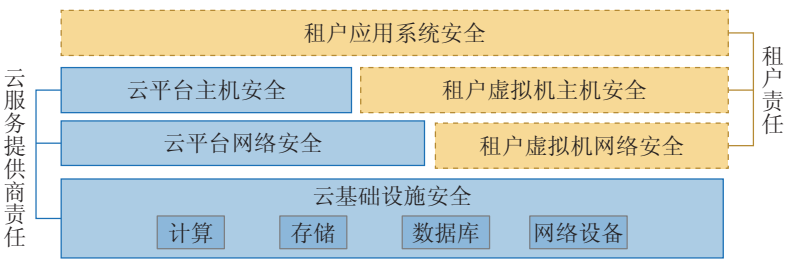


图14-17 “共担责任”模型

投送。虚拟化管理对硬件设备(服务器、存储、交换机)、虚拟资源进行集中管理，采用B/S架构，可以远程统一管理本项目中VDI桌面、应用虚拟化、服务器虚拟化三个资源池。虚拟化管理系统可管理、监控硬件资源、虚拟资源，支持虚拟机的快速部署、定制化策略调度。

桌面云登录认证采用身份认证网关和AD认证共存的解决方案，对最终用户体现为USBkey登录认证，桌面云内部实际认证利用AD的域用户名/密码，由身份认证网关/管控平台实现用户证书与用户名/密码的映射转换。

身份认证网关部署在安全接入网关前，登录WI的会话请求被网关客户端截获，送到身份认证网关(由网关客户端与身份认证网关建立安全加密通道)，经过解析处理(如域用户密码代填)后转发到WI(见图14-18)，可登录WI(网页界面，Web Interface)或通过虚拟机实现单点登录(只输入一次PIN码)。

一、管理平台

有关管理平台，具有以下内容。

- 接受管控平台客户端的USBkey注册。
- USBkey认证的用户管理，包括用户与证书的对应关系，以及维护证书与用户名/密码的映射，供认证网关通过登录模块查询域用户名和密码。
- 定期自动修改用户的域密码。

二、认证网关

有关认证网关，具有以下内容。

- 用户USBkey的登录认证：认证通过后代填用户名/密码，然后登录到WI。
- 网关客户端模块：拦截TC到Wi的登录请求，代理转发到认证网关。
- 管控平台客户端：USBkey使用前注册到管控平台，第一次注册需要输入用户的域用户名和密码；截获PIN码到管控平台中(为保证PIN码、密码的安全，需经过认证网关)。
- 登录模块：从管控平台获取PIN码、用户名和密码，然后登录虚拟机。

14.10 云计算安全的其他考虑

一、安全与用户体验

云计算体系采用的安全措施越多，该云体系的投入和运营成本相对没有安全措施云计算体系的投入与成本会越高，出于安全的需要，运营维护效率也会相对降低。举个最简单的例子，大家在使用互联网时最头疼的一个问题就是密码，不同的网站对密码的格式要求不同，导致我们需要记下很多用户名密码，密码忘记是常事，找回密码往往还需另外一个密码。提升体系安全性的同时，怎样改善用户体验，还需要持续不断的改进。利用桌面云安全方案来替换以前的PC安全方案便是一个很好的例子。桌面云安全性不仅比PC时代提升了，而且终端用户体验以及运维管理效率还能得到大幅提升，这无疑是涉密企业与机构的极佳选择。

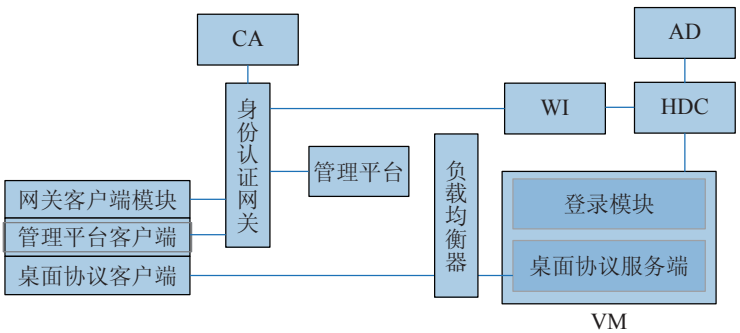


图14-18　桌面云系统安全架构

二、安全与效益

作为企业而言，要综合地考虑安全问题，因为企业需要效率和赢利。比如，对于当今已经竞争白热化的企业电子商务，安全问题很重要，因为每个安全漏洞可能都意味着企业的损失，但出于竞争的需要，企业需要不断创新交易模式，比如交易模式由PC的互联网点对点固定交易过渡到手机的移动交易模式，以此来抢占更多的市场空间。创新本身就会有安全风险(比如手机更容易丢失，更容易被植入恶意软件)，企业若想抢占先机，必须去承担这个风险，不能因此退缩不前。被竞争对手超越所带来的损失，可能比黑客窃取一部分资金的损失大得多。可以由此衍生出保险模式，比如一些企业以“你敢付、我就敢赔”这种承担安全风险的前卫模式，大力拓展其市场空间。

14.11 云计算服务法律风险及其应对

云计算技术从诞生之日起就伴随着法律上的争议，虽然某些国家已计划在法律上对其加以规范，但更多的国家对于该事物还处于理解、观望和踟蹰的状态。诚然，法律的滞后和技术的前瞻之间的张力是普遍存在。而税法则是一个例外，因其直接关系财政收入，税务机关有动力对新事物和新技术保持密切关注。但云计算技术从诞生到规模化商用已有一段时日，各国税务机关还未对其形成明确的征税规范，足见云计算为各国法律体系带来的挑战。此种规范的严重滞后或空白的一种可能解释是：云计算概念本身的模糊。这一方面源于技术层面的百家争鸣，虽然美国国家标准与技术学会对云计算的定义被广泛接受，但其未能兼容不同主体的视角和利益。这种模糊的另一方面体现为法律概念上的空白，目前还未见哪一国在法律上明确界定云计算的概念。此种概念的模糊和空白自然导致法律关系的不明确，最直接的就是对云计算服务法律性质的争议，其究竟是租赁关系、服务关系或其他法律关系？云计算中存在的三种不同的服务模式在此问题上也起到了推波助澜的作用。虽然云计算通过互联网整合成了一个整体，但其内部仍是按照分层次模式管理。因不同的层次均有自己的技术规则，其对应的法律规则也存在差异。

但即便如此，技术的巨轮不会因法律的空白而停滞。相反，由于没有法律的双黄线和红绿灯，云计算技术的演进和商用处于一种自由但“又无处不受枷锁的状态”。规范的真空带来的自由显而易见，但由于云计算服务提供商通常是实力雄厚的科技巨擘，其对此种无规则的状态反而无可适从。因为他们在法律或合规上的任何错步或踉跄都有可能招致执法机关或公众的关注。正因如此，国际市场上云计算服务巨头都制定了事无巨细的合规政策。但更多的本地云计算服务提供商还尚未意识或认识到此种监管真空的挑战。故本文拟在参考国际市场上云计算服务巨头的合规经验的基础上，结合国际先进立法实践和我国的立法动态，站在云计算服务提供商的角度，为其拨开云雾，梳理其提供服务过程中可能面临的法律风险及其对策。当然，云服务提供商通常是该产业的市场优势地位一方，这些风险并不会完全由云服务提供商承担，其有能力将风险转嫁或传递给设备、服务或软件供应商。但转嫁风险的能力并不能改变其面对风险首当其冲的事实。因此下文着重从云计算服务提供商的法律性质、司法管辖、数据保护及云安全、知识产权、消费者保护等方面剖析云计算服务提供商的风险和策略，而与云计算服务提供商存在唇亡齿寒关系的该行业其他参与者也可参照下述风险和策略制定符合自身实际情况的合规政策。

14.11.1 云计算服务提供商的法律性质

云计算服务提供商究竟属于网络服务提供商亦或电信业务提供商？这看似是一个显而易见但又不是那么容易回答的问题。大部分消费者对电信业务提供商的印象还停留在运营商的阶段，认为只有获得许可或牌照经营电信基础业务或增值业务的运营商才能被视为电信业务提供商。但现

实是，云计算服务提供商正在从事与语音、短信业务相似的业务。反之亦然，传统的电信业务提供商也在提供与云计算服务相同或相似的业务。在这两者界限越来越模糊的趋势下，传统电信业务提供商进入云服务市场并不改变其运营商的性质，但非传统电信业务提供商提供云服务究竟属于网络服务亦或是电信业务就不那么好回答了。

云计算服务提供商并非由单一种类的主体组成，云服务市场充斥着提供不同层次的技术的不同种类的公司。这些公司之间存在着竞争、合作、依存的复杂而微妙的关系。例如，苹果公司面向其最终用户的iCloud云服务就托管在亚马逊的IaaS云服务之上。这种复杂的关系虽一定程度加大了识别云计算服务提供商的难度，但我们还是可以在大体上将云计算服务提供商分为三类，即云计算服务提供商、云计算基础设施提供商和网络连接服务提供商。

在这三类服务提供者之间，很明显网络联接服务提供商受电信法律的管辖。但云计算服务提供商和云计算基础设施提供商是否也要受电信法的管辖就不那么明确了，这取决于他们所提供的服务的性质。虽有少数互联网巨头在电信监管宽松的市场中尝试建立自己的网络，但是在绝大多数情况下，云服务提供商都依赖电信网络和服务与其客户联接。此时，这些云服务提供商不会被轻易视为电信网络和电信服务提供商。但当云服务提供商可能提供类似于呼叫或短信的软件功能时，该云服务提供商就很可能受电信法的管辖。此时，云服务提供商将很有可能被要求去获取经营许可或牌照并承担电信法下设置的严格的法律遵从义务。

对于上述担忧的现实的例子有很多，VoIP技术即是典型。该技术最初出现时并不被视为通信服务。但当其用户量和使用频率与日俱增后，就出现对VoIP监管的不确定性可能会导致对消费者的保护不足的担忧。传统电信业务提供商也认为由于缺乏相应的监管，放任该技术野蛮发展，抢占了其市场，损害了传统网络运营商投资的价值。为应对此种担忧和呼吁，某些监管机构发展出了针对该新问题的、统一的规范。例如要求在号码、竞争规则和技术中性原则方面适用现有的传统电信业务法律法规，这也再次印证了上文提到的电信业务边界的模糊和可拓展性的特点。[①] 类似地，某些云计算服务提供商提供的SaaS服务可以实现某些网络功能，这就有可能将其纳入电信法的管辖范围。虽然电信法是否有必要拓展其监管范围尚有争议，但是从技术特征、最终消费者的体验以及合理期待的角度，如果接受的是类似于传统电信业务的服务，其理应受到传统电信业务服务的监督带来的便利性和可靠性。

中国法和相关监管机关对此问题也有所关注，但是其传递出的信息颇值得玩味。一方面，在工信部相关负责人在解读《电信业务分类目录(2015年版)》中提到：近年来，随着云计算等技术的快速发展，基于数据中心设施、利用互联网实现资源灵活调配、共享、协作的相关业务不断创新业务形态。该《目录》在第一类增值电信业务互联网数据中心业务定义中增加互联网资源协作服务业务的表述，进一步明确了上述业务属性范围。在此，工信部尝试将云计算服务纳入电信业务中的互联网资源协助服务业务。但是此种做法并未获得立法条文的认可或其他监管机关的印证，即工信部虽然在新发布的《电信业务分类目录》的解读中提到云计算和互联网资源协助服务业务，但是其并未明确在该新目录中规定云计算服务和定义互联网资源协助服务业务。电信监管机构的犹豫和纠结体现了其试图扩展传统电信法适用范围的野心，以防云计算产业的巨大发展前景被其他监督机构把持。目前云计算产业尚未发展到足被认定为通用的、普遍接入的服务的阶段，没有足够正当的理由将其纳入自己的管辖范围。监管机构的此种态度对于云服务提供商并不利，因为一旦云服务提供商的规模做大，其就需考虑潜在的来自监管机构的要求，而此种要求在其发展和壮大的过程中并不存在。因此云服务提供商需要提前做好应对监管机构用传统电信业务

① Demystifying Regulation in the Cloud: Opportunities and Challenges for Cloud Computing, GSR 2012 Discussion Paper

规范来规制云服务的可能，以防止在毫无防备的情况下被迫适用新的行业规则。

14.11.2 司法管辖的担忧

与法律适用相关的另一个的问题是司法管辖。此问题源于云计算服务的技术特征，即分布式计算和存储。由于存储在云上的数据，时刻都在各个服务器间转移。若此时发生纠纷，纠纷涉及的数据位置难以确定，因此产生管辖的困难。而且，云服务提供商通常会将某些云服务分包给第三方，这进一步加剧了此种困难。该第三方仅受分包合同的约束，而与数据主体并无直接合同关系，其无须消费者知晓和同意即可按照其分包合同的要求将数据主体的数据存储于多地。[①]这对消费者和云服务提供商都是潜在的危险，因为国际司法管辖适用的一个基本原则为适用密切关系地的法律或管辖。若云服务提供商或用户的关键数据所在地被认定为密切关系地甚至是经营场所所在地受该地法律管辖，而上述主体对该地法律的特殊要求并不知晓，就可能导致因无知而违反法律规定。此外，云环境下的数据可能受多地法律管辖，多个管辖地的法律可能不仅与消费者或云服务提供商住所地或经营地提供不同，而且其间就可能存在难以调和的矛盾和差异，这些矛盾和差异进一步加剧了管辖的不确定性。

因此云服务和消费者在评估云计算服务方案的风险时应考虑管辖和适用法律的不确定性。尤其需要考虑的是，是否会因仅仅数据临时在某地传输或存储即受到该地法律管辖，而此种法律又施加了相对于其住所地更严格的法律遵从的要求。例如，美国在911事件之后制定的《爱国者法案》就要求云服务提供商在某些情况下允许政府访问存储在云上的个人数据，而这种配合执法机构的义务在其他国家的法律中并不常见。

此外，云计算服务合同通常以格式合同的形式存在，云计算服务提供商应将上述因司法管辖不确定给用户带来的合规和法律遵从的不确定性的影响以明确的方式告知用户。或者也可考虑在合同中明确云服务提供商仅能将用户的数据存储在特定的司法管辖区或在此间转移，云服务市场的领跑者亚马逊和微软公司即是如此操作。这样做虽然以限制用户的使用便利性为代价，但是其相当于直接让用户面对云计算服务中的司法管辖的不确定性来做决策并承担相应的风险。

当然如果云服务提供商的技术能力还不能让其用户指定数据存储或传输的区域，也可以退而求其次，直接通过格式合同的方式来转移风险。绝大部分用户都是通过点鼠标以与云服务提供商订立云服务合同，其本身并不具备对合同条款谈判的机会和能力。因此云服务提供商也可以通过排除云服务提供商责任的方式将司法管辖不确定而带来的风险转移到消费者。若此种操作有可能构成限制用户法定权利或排除云服务提供商法定义务之嫌，云服务提供商也可通过明确责任上限和排除间接损失的方式将自身的风险降至最低水平。即在用户因司法管辖的不确定性遭受损失向云服务提供商索赔时仅承担其直接损失，而对于营业损失、商誉损失不承担责任。

对于司法管辖的问题，中国法院的实践中也积累了一些经验和原则，最高人民法院民一庭负责人就《关于审理利用信息网络侵害人身权益民事纠纷案件适用法律若干问题的规定》答记者问中提到：移动互联网的普及，对网络侵权案件尤其是侵害人身权益案件的最直接影响是，管辖地变得几乎无处不在。我国民事诉讼法及其司法解释，规定侵权案件由侵权行为地或者被告住所地人民法院管辖。利用互联网尤其是利用移动互联网发布侵权信息侵害他人人身权益，基于移动互联网本身的特征，会导致管辖法院变得更加广泛和不确定。例如，在理论上，侵权结果发生地可以是任何地方。但是，我们认为，在管辖法院确定问题上，仍然要坚持民事诉讼法所确定的“方便当事人诉讼、方便人民法院审理”的“两便”原则，同时要考虑互联网的技术特征。所以，司法解释规定，侵权行为实施地包括实施被诉侵权

① Stuart D. Levi, Skadden, Arps, Slate, Meagher & Flom LLP and Kelly C. Riedel Cloud Computing: Understanding the Business and Legal Issues

行为的终端设备所在地，侵权结果发生地包括被侵权人住所地。在此，中国法院对于技术带来的管辖无处不在的问题总结出了原则性的规则，但此种原则性的规则在云计算的复杂性面前显得过于简陋。最高人民法院民一庭负责人也不得不承认：司法解释未将实施被诉侵权行为的网络服务器所在地作为管辖地，一个非常重要的原因是云计算技术的发展、分布式服务器技术的采用等，导致以此作为管辖地具有某些不确定性，并不符合“两便”原则。由此可见，云计算分布式存储和服务带来的司法管辖不确定性的困境不仅是现实的，而且是对此前既有规则具有破坏性的。新的规则尚未建立，而旧的规则已在坍塌，这大概就是云计算服务提供商在面对司法管辖问题时最大的困难。

14.11.3 数据保护

云计算被普遍认为将成为IT产业继PC、互联网之后的第三次革命浪潮，其将不仅根本性地变革着信息生产的过程和生态环境，也显著改变着信息传播与利用的基本模式和具体方式。[①]诚然，在云计算技术的推动下，数据存储、处理和传输的成本大幅下降。但数据和隐私保护的成本和难度却因数据共享硬件资源以及数据物理位置难以确定而直线上升。各种数据丢失或数据泄露的案例屡见不鲜。例如，2011年，亚马逊公司在北弗吉尼亚州的云计算中心宕机，导致托管在其云上的多个网站受到影响。同年，谷歌的Gmail电子邮箱爆发大规模用户数据泄露事件，近15万Gmail用户的所有邮箱和聊天记录被删除，部分用户的账户被重置而无法登录。[②]此类事件在近年来层出不穷。再加之各国数据保护立法态度和实践千差万别，缺乏对于云计算环境下的数据保护的统一和确定的规则，因此作为云服务提供商需要特别注意其云服务面向的市场地的差异性的规定，在遵从数据保护相关法律和规章制度的前提下部署云业务和提供云服务。

虽然各国具体层面的数据保护法律差异性比较大，但是数据保护的基本理念和原则是类似的。特别是在法治越发达地区，其数据保护的规则同质化程度越高，即越倾向于赋予最终消费者相对严格的数据和隐私保护。这其中的典型代表是欧盟的数据保护法，在欧盟，数据保护规则经历了从指导性规则向强制性规则演变的一个历程。此前数据保护规则在某种程度上是统一在《欧盟数据保护指令》中的。尽管该指令已经为欧盟28个成员国采纳和转化，并且适用基本相同的原则。但各国依然存在立法的差异，这就要求云服务提供商在某些情况下去审视不同国家的隐私和实践。[③]但近期欧盟推出了一项新的立法，即《一般数据保护法》以增强对个人数据和隐私的保护。该法扩大了数据主体的权利，增加了义务主体的义务，也强化了执法机关的执法和处罚力度，且该立法将无须转化直接适用于所有成员国。[④]因此下文将重点介绍欧盟《一般数据保护法》对于数据保护确定的原则和规则，因为在全球范围内，欧盟立法设计的个人数据保护制度是最为完善和严格的，因此若云服务提供商能够满足该法提出的诸多合规要求，其在其他地区除需关注某些特定的规定外，可以基本以该法的要求作为当地合规实践的框架和基础。

虽然上文提到了云计算领域法律适用存在大量空白和模糊地带，但是欧盟本地或面向欧盟的云计算服务因其本身的属性必然会受到欧盟数据保护法的管辖。欧盟数据保护法适用的两个基本的前提条件为数据处理和个人数据。而云计算服务不可避免地涉及数据的处理。因而云计算服务也就自然落入了欧盟数据保护法的管辖范围。欧盟数据保护法是建立在个人数据、处理、数据控制者、数据处理者这几个基本概念之上的，要理

① 王太平. 云计算环境下的著作权制度：挑战、机遇与未来展望. 知识产权，2013(12)：18。

② 徐慧丽：《云计算环境中的法律风险-以数据安全为视角》，载《科技与法律》2013年第6期，第1页。

③ 参见《欧盟数据保护指令》The Data Protection Directive (Directive 95/46/EC)。

④ 参见《一般数据保护法》*The General Data Protection Regulation* (*GDPR*)。

解该法对云计算服务适用的影响，首先要明确云计算服务以及云服务提供商在这些概念所构建起的框架间的位置和关系。欧盟数据保护法的起点是对个人数据的界定，如前文所述，云计算服务不可避免地涉及个人数据。欧盟数据保护法将个人数据定义为任何关于具体或可识别的个人的信息。因此云服务提供商很有可能在如下情况下触发欧盟数据保护法的适用：例如当收集个人数据用以提供云服务；当云服务用以处理个人数据；当云服务产生的某些数据而被认定为个人数据的控制者。

针对云服务提供商在面向不同的客户提供云计算服务过程中扮演的不同角色，欧盟数据保护法设置了不同程度的数据保护规则。在欧盟数据保护法的框架下，云服务提供商可能构成数据控制者、数据处理者或第三方。针对上述三种不同的角色，欧盟数据保护法为其设置了不同的数据和隐私保护的责任和义务。处理个人数据的安全保障义务以及法律责任主要由数据控制者承担，且数据控制者在选择数据处理者时应承担必要的审慎和注意的义务，并选择适格的数据处理者从事数据处理，而数据处理者需选择具体合适的技术和组织上的安全保障措施。

上文界定了云服务中不同处理可能扮演的角色，云服务提供商可能被认定为数据控制者，也可能被认定为数据处理者，但是无论它被认定为何种角色，都要确保其数据处理行为符合法律的要求，而此种要求在欧盟数据保护法框架下并非体现在具体的条文中，更多的是体现在其设定的一系列原则中。而此种设计也是符合数据保护法这类受瞬息万变的科技影响的法律的本质要求的。若规定得太过于具体，则可能造成法律的滞后性与技术的前瞻性之间的严重脱节。因此通过设定原则性规则可以保留一定的解释空间和灵活性以应对规制对象变化太快带来的挑战。因此云服务提供商必须关注如下几个数据处理的原则，并将其作为遵从数据保护义务的主线。

第一个主要的原则是关于数据处理的公平合法原则：公平是指数据主体必须知道数据处理的存在和信息。合法则是处理过程中必须遵守数据保护的原则、法律义务以及其他相关的法律要求。第二个主要的原则是目的明确且有限范围原则：目的明确是指数据控制者仅可在特定且合法的目的下处理个人数据。若不符合最初特定的目的，则不能处理数据。有限范围即处理行为必须充分并相关，且不能超出其收集数据的目的。换言之，应以一种尽可能不侵权的方式处理个人数据。第三个主要的原则是数据质量原则，即数据控制者必须确保其收集数据的准确性。第四个主要的原则为透明原则：透明原则体现在数据保护立法的诸多方面，主要有：在未事先通知数据主体数据处理的存在以及处理的目的和方式的情况下，不能收集存储和处理数据。此外，在通知时，控制者还要考虑数据主体的预期并提供任何可能影响数据主体同意的信息。此外还有诸多其他数据保护原则：例如，数据最小化原则，即数据控制者仅能采集直接与完成特定目的相关且必要的数据。而且，其仅能按照其实现目的的要求保持必要的记录。《一般数据保护法》较之《数据保护指令》已强化此种原则，其规定只有当其他方式不能实现目的时才能收集和保存数据。对于互用性和便携性原则，此原则旨在促进跨系统的兼容。这就要求云服务提供商在系统开发和部署业务时要满足可靠性和互用性的行业标准。最后《通用数据保护法》还强调责任原则，要求数据控制者记录每一处理行为，并于监管机关要求时提供该记录。[①]

14.11.4 云安全

云安全的问题与数据和保护问题相关但不完全相同。可以从两个维度来理解云安全问题：首先用户会担心他们的数据和资源能否随时使用。其次，他们希望数据不会被未授权获取或接触。前者直接受云服务提供商的控制，后者则通常不在云服务提供商的范围，尤其是云服务提供商通过网络连接用户时，网络的可靠性和安全性就决

① 参见《一般数据保护法》*The General Data Protection Regulation* (*GDPR*)。

定了云服务的安全性。此外，理论上用户和云服务提供商均应对云安全负有责任。虽然云服务是一系列服务的集成者，但用户也有义务采取必要的措施确保其传输给云服务提供商的数据处于必要的安全保障措施之下。例如，用户可以在传输数据过程中对关键业务数据进行加密处理。这样做虽然会伴随一定的成本和数据传输处理效率的贬损，但其能从源头上保障数据的安全。遗憾的是，现实中这些用户并不热衷这项实践，而是将自己应该承担的数据安全保障义务一并交给云计算服务提供商。

尽管如前所述，云安全一部分取决于公共网络的安全性和可靠性，一部分有赖于用户自身的安全保障措施，但是监管机构还是倾向于从单一维度，即只看云服务提供商是否遵从相应安全保障义务来确定云安全的责任承担。例如，《杭州市计算机信息网络安全保护管理条例》第十八条原则性规定：互联网接入服务提供者及主机托管、租赁和虚拟空间租用等互联网数据中心服务提供者，应当建立并落实以下安全保护制度和安全保护技术措施。然后在其第三十九条进一步具体规定：互联网接入服务和数据中心服务提供者、互联网信息服务提供者或者互联网上网服务提供单位违反本条例第十八条、第十九条或者第二十一条规定的，由公安机关给予警告，责令其限期改正，并可处以一千元以上一万五千元以下的罚款；情节严重的，给予六个月以内停机整顿的处罚；对单位直接负责的主管人员和直接责任人员可处以五百元以上五千元以下的罚款。此处，监管机构对于违反云安全保障义务设置的直接的罚款数额并不大。但其若施加六个月以内停机整顿的处罚将对云服务提供商造成巨大的间接损失和商誉损失。

现状如此，云服务提供商只能承担理应由网络提供商和用户承担的安全保障义务。而在云服务领域，对于此种安全保障义务主要源于两个方面的要求，一为行业标准，二为特定法律规范的要求。行业标准主要有 ISO/IEC的一系列标准等，云计算服务商可利用对此种标准的遵从来为其外部审计和安全认证提供依据和基础。此外，一些行业标准化组织，例如云安全联盟，也在尝试制定业界的最佳实践和安全隐私的指南来统一目前云计算服务行业的安全实践。而在特定法律规范要求方面，面向金融行业开放的云服务无疑是监管最严格的区域。因此云服务提供商可参照金融行业对数据中心和云计算服务的相关要求制定自身的合规策略以应对潜在的风险。例如，中国支付清算协会网络支付应用工作委员会发布的《支付机构互联网支付业务风险防范指引》中要求对于金融机构的数据中心：任何人员进出数据中心机房需登记，涉及重要数据的区域需提前申请并由相关责任人陪同方可进入，参观人员仅可访问指定授权区域。又如，中国保险监督管理委员会发布的《保险机构信息化监管规定》公开征求意见稿中要求保险行业的数据中心规划应当报中国保监会备案，并且对数据来源于中华人民共和国境内的，数据中心的物理位置应当位于境内。再如，《广东省计算机信息系统安全保护条例》要求第三级以上计算机信息系统以及政府投资的计算机信息系统的数据中心必须设在境内。此外，其还要求对于发现有害信息后应及时采取删除、停止传输、关停账号或者服务等技术措施。这些规范在数据中心管理、数据中心位置以及非法信息处理等方面为云服务提供商的合规实践提供了有参考价值的指引。

此外，云安全不仅关乎数据机密性、完整性和真实性，它也关乎信息所有权。虽然在大多数法律体系下，数据本身的财产属性并不明确，但法律也赋予了数据某些权利属性，这些权利属性涉及个人数据、专利或版权的某些内容，因此云服务提供商需关注这些内容并确保其对数据的传输或处理不侵犯此种权利。在云环境中，数据信息所有权的另外一个方面则是云服务中衍生的数据的归属。对于云服务提供商而言，其完全有能力从这些数据中实现价值，甚至据此为用户提供免费的云服务。但是，在许多国家的电信法下，都限制运营商直接使用用户在使用通信服务过程

中产生的数据(例如流量数据)。[①]如上述第一个问题所述，电信法的此种要求是否会直接适用到云计算服务中是存疑的。但是，若云服务越来越普遍，监管机构也完全有可能考虑对云服务中衍生数据的使用采取严格监管的态度。

14.11.5 知识产权

为实现资源高效利用与灵活供给而诞生的云计算，一直是高科技与新理念的象征，而维持整个系统运转的操作系统和管理模块都是以软件的方式定义的。与此同时，云计算平台上有大量的数据在奔驰，这些数据承载的可能是涉及著作权、专利、商标以及专有技术等各类知识产权的信息。云计算架构和运作中的这些特点，使得知识产权问题在整个法律分析中显得十分基础而重要。

总体而言，云计算服务中可能存在的知识产权问题主要体现在两个方面：一是与云计算服务提供商相关的知识产权问题；二是与使用云计算解决方案相关或与存储在云计算平台的硬件上的数据相关的知识产权问题。云计算服务提供商向用户所提供的服务是打包的解决方案，可能包括存储、计算能力、开发环境及工具、业务应用等，需要硬件和软件层面的无缝配合。其中软件层面就又包括了系统软件和SaaS层的应用软件。系统软件是维持整个云计算平稳高效运转的血液，具体体现为云平台操作系统(Cloud OS)以及运作在该系统之上的管理软件。这些基础性的软件一般由云计算服务提供商自行研发或从第三方采购并集成后提供给云平台用户。为避免后续可能出现的知识产权问题，云计算服务提供商应当在合同层面与其软件供应商或用户规定清楚下列问题。

首先，对于之前已有的背景知识产权的保护问题，应在与供应商及用户的合同中表明，背景知识产权的归属不因平台运营或云服务提供而发生变更，原权利方仍完整地享有其受知识产权法律规定的全部权益，但供应商应授予云服务提供商充分的许可权以使用该软件，以商业目的向第三方许可和转让的，云服务提供商应授予用户平台应用软件一般的使用许可，该许可是有期限的、不可转让的、非排他并可撤回的。另外，云服务提供商应要求供应商提供充分的权利说明以保证其权利不存在瑕疵，同时应当要求供应商提供权利保证，以应对可能的纠纷和赔偿。

其次，对于在平台运营过程中新产生的知识产权，也应在相关合同中清楚规定其归属和许可问题。如果是云服务提供商与供应商共同联合开发的原件，原则上应归双方共同所有，并约定共有的原则和使用方法，供应商应提供全部权利材料。如果是用户利用PaaS层的开发环境新开发出来的软件，一般而言知识产权属于用户所有，但平台可通过制定开发规则，约定平台享有部分知识产权。

除了以上问题外，开源软件所引发的法律问题也日益引起人们的注意。同样作为新生事物，开源软件在为开发者降低研发和维护成本的同时，也因为其适用的开源许可证不同而带来了新的遵循义务。因为开源许可证类型多样，这里不展开赘述，仅简述开源软件的法律风险应对。若平台IaaS、PaaS及SaaS中含有第三方提供的开源软件，则云服务提供商应在其软件供应商协议中明确要求该第三方提供其软件中所使用的开源内容列表及开源使用声明，并承诺承担因任何开源软件问题所导致的赔偿责任；若为云服务提供商的自研软件，则应根据实际情况合理选择开源许可证类型，并履行相关的开源遵从义务。

第二部分是与使用云计算解决方案相关或与存储在云计算平台的硬件上的数据相关的知识产权问题。用户在使用云计算提供商的解决方案时也可能会产生一些知识产权，例如，用户为使用该解决方案可能会在IaaS和PaaS层创造新的用户接口(user interface)。除此之外，用户在云计算平台上存储和传输非法内容，尤其是侵犯他人知识产权的内容，也会引发法律风险。通常而言，应当由内容的上传者承担第一位的侵权责任，但对

① Demystifying Regulation in the Cloud: Opportunities and Challenges for Cloud Computing, GSR 2012 Discussion Paper

于中间方，如宿主(hosts)，应承担何种责任，各国法律并没有一致的规定和看法。

欧盟的电子商务指令针对中间方的二级责任(secondary liability)规定了较为具体的制度。该指令规定，各成员国禁止为企业设立一般性的审查义务，同时根据企业具体从事的活动规定了三项责任免除情形，即纯管道(mere conduit)、缓存(caching)和宿主(hosting)。具体到云计算平台上，应考虑宿主情形的适用，即避风港规则(the safe harbor regime)。指令的第十四条规定了云计算服务提供商得以适用该种免责情形的条件：(1)中间方对该非法活动及信息并不知情，如果被要求损害赔偿，中间方应没有意识到表明非法活动及信息明显的事实或环境；(2)中间方一旦获知或意识到上述情形，应立即采取行动删除或禁止访问该信息。[①]欧盟法院的判例法认为，云计算服务提供商在提供存储服务时，其角色定位一定是被动的、技术性的和自动的，这样才能使用避风港规则。相反地，如果云计算服务提供商的行为较为主动，并对在线的内容实施了一些控制、筛选和决定动作，则基本不可能再适用该责任免除规定了。

14.11.6 消费者保护问题

消费者保护的问题在各个领域都存在，之所以在云计算领域也值得一提的主要原因在于，人们对于技术能力的乐观往往超出技术本身的潜能。这种盲目乐观已经导致许多云服务提供商遭受消费者投诉或相关监管机构的调查。例如，在英国，广告标准委员会认定一个云托管服务提供商存在虚假宣传，误导消费者的陈述。其向用户宣传其能够为用户提供无限流量和能力的套餐，从实际上只要随时随地地按照服务合同约定的水平为用户提供了服务，在用户的角度来看确实享受了无限量的资源和能力。但从理论上看，监管机关认为服务器物理和硬件资源所代表的计算和存储的能力是有限的，这就意味着总有些客户不能无限量地使用该服务，故而云服务提供商的宣传构成了不实陈述。同样地，一家网站托管公司的5个九的服务水平的承诺也被认定为误导性宣传。虽然向用户承诺5个九的服务水平是云计算服务行业的通行操作和实践，但该网站托管公司在短短几个月内就出现了三次重大网络问题。虽然其有可能在漫长的服务合同期限内控制事故次数而使得服务水平最终达到5个9的水平，但是监管机构在面对用户的投诉时并不会给云服务提供商太多的时间去美化他们的承诺的兑现状况。[②]

消费者保护领域另外一个相关的问题则是锁定，即用户在使用一段时间服务后难以以合适的格式回收数据并迁移到另一个云计算服务提供商的云服务中。对于这个问题，目前市场上不同的云服务提供商有着不同的做法。有些云服务提供商会在合同中约定其可以授予数据主体一定的宽限期取回数据；有些则约定直接删除。虽然云服务提供商可以合同约定或技术水平的限制拒绝辅助用户实现迁移，但此种约定或辩解是存在一定法律风险的，因为这可能引发竞争法和消费者保护法的双重问题。如果这些限制并非为提供服务所必须，可能被认定为违反竞争法，因为这样做相当于为其他云服务提供商设置了准入门槛。

格式合同也是消费者保护领域值得关注的问题。云计算服务提供商的商业模式(标准化服务模式)的本质要求在于限制定制化合同条款。许多中小企业只能通过用鼠标单击我同意的方式订立格式合同获得云服务。在这种情况下，客户无法协商合同条款或对云服务提供商进行必要的尽职调查，因此云服务提供商应注意其服务所在地关于格式合同生效和解释的原则。例如在中国《合同法》和《消费者保护法》下就确立了格式条款的解释应采取三项特殊的原则：即应当按照通常理解予以解释、对条款制作人做不利的解释，以及格式条款和非格式条款不一致的，应采用非格式条款的原则。此外，提供格式条款一方免除其责任、加重对方责任、排除对方主要权利的，该条款无效。因此云服务提供商需注意在其格式合同中将可能的免责条款通过加粗或下划线的方式提

① 参见《欧盟电子商务指令》(Directive 2000/31/EC)。

② Demystifying Regulation in the Cloud: Opportunities and Challenges for Cloud Computing, GSR 2012 Discussion Paper

示用户注意，或者在用户提出疑问时通过网上客服向其明确说明，使其充分理解该条款对其潜在的影响。这样才能确保其设置的合同条款不至于因格式合同解释规则而被认定为无效。

14.11.7 其他问题

除上述问题外，云计算服务提供商还有可能面临竞争法和环境法的约束，以及执法机关的不请自来的造访。由于这些风险与前述章节之间的关联性不强，故在此分别论述。

如前文所述，云计算服务缺乏相关行业标准。此种表述并不完全准确，其实云计算行业并不缺乏相应的标准。在这个以规模论成败的行业，行业领导者产生或推行的标准就是行业标准。这一方面赋予了行业领导者制定有利于自身的标准的自由和权利，但另一方面也很容易使云计算服务行业领导者陷入不正当竞争或垄断的争议。例如，在2010年苹果公司在其与应用开发者的许可条款中加入限制条款，要求开发者使用苹果原生编程工具和语言开发应用。该举措招致了欧盟委员会的调查，其认为苹果的此种限制会损害那些与苹果应用平台竞争的平台。而苹果在欧盟委员会介入调查后的几个月内便宣布取消此项限制。此外，云计算领域的反不正当竞争和反垄断法的风险也可能直接出现在云基础设施提供商一侧。例如，在2010年欧盟委员会对IBM发起了一项关于IBM大型机的调查。IBM被指控涉嫌捆绑销售其大型机操作系统且对与其有竞争关系供应商实行差别对待。欧盟委员会认为IBM此举旨在利用其在大型机操作系统的垄断地位来提升其硬件的市场占有率。因为云计算设施将直接影响到云服务的可迁移性，因此云计算基础设施提供商也应注意遵循相关反不正当竞争法和反垄断法对于捆绑销售、协议定价或价格歧视的限制。[①]

执法机关的不请自来是另一个让云服务提供商非常头痛的问题。据云计算服务行业的领跑者亚马逊和微软公司的云安全白皮书披露，其每年都面临近百起执法机关未告知数据主体调取数据的请求。云上的数据一方面关系个人隐私，另一方面可能涉及巨大的商业价值，若云服务提供商面对执法机构的请求不加选择地接受，将招致其用户的背弃甚至是诉讼。但若其不配合，法治不健全国家的执法机构又可以通过各种方式干扰云服务提供商的正常运营。这对于云服务提供商而言确实是两难，虽然业界也有许多硬脖子的云服务提供商强硬地面对执法机关的调取数据的请求。如苹果公司与FBI的对峙就闹得满城风雨，而WhatsApp更因此在巴西被要求全面停止运营。但是并非每个云服务提供商都有资本或能力去效仿此种行为以博取用户的信任和支持。绝大部分云计算服务提供商还是要评估当地法治环境和自身状况决定是否配合执法机关的请求。如果其不得不接受此种请求，也应该制定相应的指引来将执法机关对涉案数据主体或其他数据主体的影响降至最低。例如，注意对前台接待人员或保安人员的培训，因为他们通常最先接触到执法或调查人员。因此应要求上述人员能做到言谈举止得体专业，知晓应当核查哪些官方证件，清楚如何有条不紊地采取下一步行动，并且明确应当通知公司的哪位负责代表。再者，还应及时指定代表，负责回答调查人员的问题，并在调查人员进入经营场所调查后，全程陪同调查人员。此外，在应对执法机构调取数据的要求时，需要执法机构出具符合法律规定的完整书面材料，并针对主动提供的以及被执法人员带走的所有文件，准备一份书面记录。最后，在法律允许的情况下，于执法机构调取数据后及时将调取数据情况通知数据主体或用户。[②]

最后一个潜在的风险是来自环境保护方面的压力。大型数据中心消耗大量的能量，这也就是为什么微软提出要在北极或海下建立数据中心的设想。这种环境方面的担忧并非杞人忧天，近期有报道称，仅YouTube每年就会消耗全球0.1%的能量。正因如此，欧盟委员会在2009年颁布了

① Demystifying Regulation in the Cloud: Opportunities and Challenges for Cloud Computing, GSR 2012 Discussion Paper

② 参见牟晨，史昭君所撰写的《中国执法突查所必需知道的几件事》。

《数据中心节能规范》。它设定了一系列自愿采纳的方案建议，让云服务提供商承诺其数据中心的设计符合一定的能效标准。这些标准最终可能会被立法吸纳。再如，我国《国务院关于促进云计算创新发展培育信息产业新业态的意见》就规定：新建大型云计算数据中心能源利用效率(PUE)值优于1.5。引导大型云计算数据中心优先在能源充足、气候适宜、自然灾害较少的地区部署，以实时应用为主的中小型数据中心在靠近用户所在地、电力保障稳定的地区灵活部署。此外，《广州市绿色建筑和建筑节能管理规定》也明确，绿色数据中心是指机房、IT系统、机电设备、数据应用管理等的设计取得能源效率最大化和环境影响最小化的数据中心。因此云计算服务提供商以及构建数据中心的云基础设施提供商应注意遵从上述规范对于云数据中心能效和其他环境标准的遵从。

以上为云服务提供商在提供服务过程中可能面临的主要风险。这些风险本质上源于云计算按需服务、广泛接入、弹性拓展等特征。当然，上述风险也不可能是云雾缭绕的云计算服务市场可能面临的风险的全貌。而且在这个波谲云诡的市场，任何此前的经验随时都有可能不再适用。即便如此，云服务提供商和与其有千丝万缕联系的其他云服务市场参与主体也不应该在这个监管的模糊领域盲目乐观和自由。而是应该在看到行业的大象们都开始带着镣铐跳舞的时候，也目测其锁链的长度，结合自身的情况用合规或其他措施为自己划定安全的、自由的边界。

缩略语

缩略语	全称	说明
AAA	Authentication、Authorization、Accounting	验证、授权和记账
ACL	Access Control List	接入控制列表
ACM SIGCOMM	ACM：Association for Computing Machine，美国计算机协会 SIGCOMM：Special Interest Group on Data Communication，数据通信专业组	SIGCOMM是ACM组织在通信网络领域的旗舰型会议，也是目前国际通信网络领域的顶尖会议，由ACM SIGCOMM组织举办
AD	Active Directory	活动目录
Aero	Authentic(真实)、Energetic(动感)、Reflective(反射)及Open(开阔)	Windows Aero是从Windows Vista开始使用的新型用户界面，透明玻璃感让用户一眼贯穿
API	Application Programming Interface	应用程序接口
ARM	Advanced RISC Machine	一种处理器架构
ARP	Address Resolution Protocol	地址解析协议
ASM	Any-source multicast	任意源组播
ASPF	Application Specific Packet Filter	基于应用的包过滤
ATOM	Intel® Atom™	英特尔®凌动™处理器，是Intel的一个超低电压处理器系列
AZ	Availability Zone	可用区域
BGCF	Breakout Gateway Control Function	出口网关控制功能
BIOS	Basic Input/Output System	基础输入输出系统
BOSS	Business Operating Support System	业务运营支撑系统
BSS	Business Support System	业务支撑系统
C.R.S.A	Centralization， Routine， Standardization，Automation	集中化，常规化，标准化，自动化
CBT	Changed Block Tracking	改变数据块跟踪
CDC	Cloud Data Center	云数据中心
CEP	Complex Event Processing	复杂事件处理
CHAP	Challenge Handshake Authentication Protocol	询问握手认证协议

(续表)

缩略语	全称	说明
CI/CD	Continuous Integration and Continuous Delivery	持续集成和持续交付(软件研发术语)
CIM	Common Information Model	通用信息模型
CLOS	英文姓氏	一种无阻塞交换架构的名称，源于发明者Charles Clos博士
CNI	Container Network Interface	容器网络接口
CNM	Container Network Model	容器网络模型
CORBA	Common Object Request Broker Architecture	通用对象请求代理架构是软件构建的一个标准
CPM	Communications Processor Module	通信处理模块
CQL	Continuous Query Language	连续查询语言
CR	Client Receive	客户端接收
CRM	Customer Relationship Management	客户关系管理
CS	Client Send	客户端发送
DAAS	Desktop as a Service	桌面即服务
DAG	Direct Acyclic Graph	有向无环图
DAS	Direct-attached Storage	直接附加存储
DB	DataBase	数据库
DC	Data Center	数据中心
DDoS，DOS	Distributed Denial of Service	分布式拒绝服务攻击
DevOps	英文Development和Operations的组合	DevOps是一组过程、方法与系统的统称，用于促进开发(应用程序/软件工程)、技术运营和质量保障(QA)部门之间的沟通、协作与整合；它的出现是由于软件行业日益清晰地认识到，为了按时交付软件产品和服务，开发和运营工作必须紧密合作
DHCP	Dynamic Host Configuration Protocol	动态主机配置协议
DHT	Distributed Hash Table	分布式哈希表
DMA	Direct Memory Access	直接内存访问
DNAT	Destination Network Address Translation	目标地址转换
DPM	Distributed Power Management	分布式电源管理
DRS	Distributed Resource Scheduler	动态资源调度
DSP	Digital Signal Process	数字信号处理器
DX	DirectX (Direct eXtension)	微软公司建立的一系列专为多媒体以及游戏开发的应用程序编程接口
EBS	Elastic Block Store	弹性块存储
EC2	Elastic Compute Cloud	弹性计算云
ECC	Error Checking & Correction	错误检查与纠正
ECMP	Equal-cost multi-path routing	等价多路径路由
EDC	Enterprise Data Center	企业数据中心

(续表)

缩略语	全称	说明
EIP	Elastic IP Address	弹性IP地址
ELB	Elastic Load Balance	弹性负载均衡
EM	Element Management	电信设备的管理系统
EOI	End of interrupt	中断结束
EPC	Evolved Packet Core	演进型分组核心网
EPT	Extended Page Table	页表扩充技术
ERP	Enterprise resource planning	企业资源计划
ESB	Enterprise Service Bus	企业业务总线
ESP	Event Stream Processing	事件流处理
ETL	Extract-Transform-Load	用来描述将数据从来源端经过萃取(extract)、转置(transform)、加载(load)至目的端的过程
ETSI ISG NFV	The European Telecommunication Standards Institute (ETSI)，Industry Specification Group (ISG) for Network Functions Virtualization (NFV)	欧盟电信标准机构网络功能虚拟化工业标准组
EVS	Elastic Virtual Switch	虚拟交换机
FC	Fibre Channel	光纤通道
FCAPS	Fault，Configuration，Accounting，Performance，Security	表示网络管理的五种基本功能的缩写：故障、配置、计费、性能、安全管理
FSFO	Fast-Start Failover	快速启动故障切换
FW	FireWall	防火墙
GDI	Graphics Device Interface	图形设备接口
GDT	Global Descriptor Table	全局描述符表
GFS	Google File System	谷歌文件系统
GPA	Guest OS Physical Address	客户操作系统的物理地址
GPS	Global Positioning System	全球定位系统
GPU	Graphics Processing Unit	图形处理器
GRE	Generic Routing Encapsulation	通用路由封装协议
GUI	Graphical User Interface	图形用户界面
GVA	Guest OS Virtual Address	客户操作系统的虚拟地址
HA/FT	High Availability/Fault Tolerance	高可用性/容错
HAE	Huawei App Engine	华为APP引擎
HDD	Hard Disk Drive	硬盘
HDFS	Hadoop Distributed File System	Hadoop分布式文件系统
Hop	Hadoop Online Prototype	Hadoop在线原型
HOT	Heat Orchestration Template	Heat编排模板
HPA	Host OS Physical Address	宿主操作系统的物理地址
HPC	High Performance Computing	高性能计算
HRM	Human Resource Management	人力资源管理
HTML	HyperText Markup Language	超文本标记语言

(续表)

缩略语	全称	说明
HTTP/HTTPS	HyperText Transfer Protocol Secure	超文本传输(安全)协议
HWS	Huawei Web Services	华为企业云服务(公有云)
I/O	Input and Output	输入输出
I2RS	Interface to Routing System	路由系统接口
IaaS	Infrastructure as a Service	基础设施即服务
IAM	identity and access management(简称：IAM)身份和接入管理	统一身份认证(Identity and Access Management)是面向企业租户的安全管理服务，包括对OTC系统服务的访问控制、权限分配和访问策略管理，用于对用户的权限控制
ICA	Independent Computing Architecture	独立计算架构，思杰公司开发的远程桌面协议
ICP	Internet Content Provider	互联网内容提供商
IDS	Intrusion Detection Systems	入侵检测系统
IDT	Interrupt Descriptor Table	中断描述符表
IETF	Internet Engineering Task Force	互联网工程任务组
IF	Interface， IF1 IF2 IF3…	接口缩写代号
IMS	IP Multimedia Subsystem	IP多媒体子系统
iNIC	Intelligent Network Interface Card	智能网卡
IOMMU	input/output memory management unit	输入输出内存管理单元
IOPS	Input/Output Operations Per Second	每秒读写(I/O)操作的次数
IPAM	IP Address Management	IP地址管理
IPC	Interprocess Communication	进程间通信
IPMI	Intelligent Platform Management Interface	智能平台管理接口(定义：Intel、HP、NEC、Dell等厂商为提高服务器的可用性指标而推出的智能化平台管理接口标准；IPMI可为服务器提供设备管理、传感器和事件管理、用户管理、风扇框和电源框管理、远程维护等功能。)
IPMI	Intelligent Platform Management Interface	智能平台管理接口
IRF	Intelligent Redundant Framework	智能冗余框架
IS-IS	Intermediate System-to-Intermediate System	中间系统到中间系统
ISP	Internet Service Provider	互联网服务提供商
ISV	Independent Software Vendor	独立软件开发商
ITIL	Information Technology Infrastructure Library	信息技术基础设施库
ITSM	IT Service Management	IT服务管理
KPI	Key Performance Indicators	关键绩效指标
KSM	Kernel Samepage Merging	内核同页合并
KV	key/value	key/value (分布式存储系统)
KVM	Kernel-based Virtual Machine	基于Linux内核的开源的虚拟化解决方案

(续表)

缩略语	全称	说明
LAMP	Linux， Apache， MySQL， PHP	LAMP是指一组通常一起使用来运行动态网站或者服务器的自由软件名称首字母缩写：Linux，操作系统；Apache，网页服务器；MariaDB或MySQL，数据库管理系统(或者数据库服务器)；PHP、Perl或Python，脚本语言PHP用于编辑网页，网页运行在Apache软件上，网页的数据来自MySQL Apache和MySQL，安装在Linux操作系统上
LB	Load Balance	负载均衡
LBS	Load Balance Server	负载均衡服务器
LDAP	Lightweight Directory Access Protocol	轻型目录访问协议
LUN	Logical Unit Number	逻辑单元号
LXC	Linux Container	Linux Container容器
MANO	Management and Organization	管理与组织
MBPS	Megabyte Per Second	兆字节每秒
MGCF	Media Gateway Control Function	媒体网关控制功能
MME	Multi Media Extension	多媒体扩展
MMU	Memory Management Unit	内存管理单元
MPLS	Multiprotocol Label Switching(简称MPLS)	多协议标记交换，在IP路由和控制协议的基础上，向网络层提供面向连接的交换技术，它采用短而定长的标记封装各种链路层分组
MPP	Massively Parallel Processing	大规模并行处理
NAS	Network Attached Storage	网络附属存储
NFP	Network Forwarding Path	网络转发路径
NFS	Network File System	网络文件系统
NFV	Network Function Virsualization	网络功能虚拟化
NFVO	Network Functions Virtualization Orchestrator	网络功能虚拟化编排器
NOS	Network Operation System	网络操作系统
NS	Network Service	网络服务
NSO	NS Orchestrator	网络服务编排
NUMA	Non-uniform Memory Architecture	非一致性内存架构
NVDIMM	Non-Volatile Dual In-line Memory Module	NVDIMM是在一种集成了DRAM加非易失性内存芯片的内存条规格，能够在完全断电的时候依然保存完整内存数据
NVGRE	Network Virtualization Using Generic Routing Encapsulation	采用通用路由封装的网络虚拟化
NVRAM	Non-Volatile Random Access Memory	非易失性随机访问存储器
OA	Office Automation	办公自动化
OAM	Operations， Administration， and Maintenance	运营、管理、维护

(续表)

缩略语	全称	说明
OASIS	Organization for the Advancement of Structured Information Standards	结构化信息标准促进组织
OBS	Object-Based Storage	对象存储
OLAP	On-Line Analytical Processing	在线分析处理
OLTP	On-Line Transaction Processing	联机事务处理
OM	Operation Management	运营管理
OMS	Operation Management System	运营管理系统
ONF	Open Networking Foundation	开放网络基金会
OpenGL	Open Graphics Library	开放图形库定义跨程序语言、跨平台的应用程序接口(API)的规范，用于生成二维、三维图像
OS	Operation System	操作系统
OSS	Operations Support System	运营支撑系统
OVF	Open Virtualization Format	开放虚拟化格式
OVN	Open Virtual Network	开放虚拟网络
OVS	Open vSwitch	开放虚拟交换标准，开放虚拟交换器，开放虚拟化软件交换机
OVSDB	Open vSwitch Database Management Protocol	开放虚拟交换机数据库管理协议
PaaS	Platform as a Service	平台即服务
PCB	Printed circuit board	印刷电路板
PCI	Peripheral Component Interconnect	外设互联标准或个人电脑接口
PCR	Platform Configuration Register	平台配置寄存器
PCRF	policy and charging rules function (PCRF)	策略和计费规则功能
PDA	Personal Digital Assistant	个人数码助理
PDM	Product Data Management	产品数据管理
PE	Processing Elements	处理元素
PF	Physical Function	物理功能
PIN	Personal Identification Number	个人鉴别码
PM	Physical Machine	物理主机
POD	Pod	密集的小群，指代IT物理或逻辑(虚拟)设备集群
POSIX	Portable Operating System Interface	可移植操作系统接口
PTP	Point to Point	点对点模型
PUE	Power Usage Effectiveness	能源使用效率
PXE	Preboot Execution Environment	预启动执行环境
QEMU	Quick EMUlator	一款开源免费的硬件虚拟化软件
QEMU	Quick EMUlator	一款仿真处理器的自由软件
QoE	Quality of Experience	用户体验质量

(续表)

缩略语	全称	说明
QoS	Quality of Service	服务质量
RAID	Redundant Array of Independent Disks	独立硬盘冗余阵列
RAS	Reliability， Availability and Serviceability	可靠性、可用性、可维护性
RBAC	Role Based Access Control	角色的访问控制
RC	Replication Controller	复制控制器
RDB	Relational Database	关系型数据库
RDC	Regional Data Center	区域数据中心
RDD	Resilient Distributed Dataset	弹性分布式数据集
RDMA	Remote Direct Memory Access	远程直接数据存取
RDP	Remote Desktop Protocol	远程桌面协议
RDS	Relational Database Service	关系型数据服务
RESTful	REST，Representational State Transfer	含状态传输的 Web 服务(也称为 RESTful Web API)是一个使用HTTP并遵循REST原则的Web服务
RO	Resource Orchestrator	资源编排
RoCE	RDMA over Converged Ethernet	基于融合的以太网的远程直接数据存取
RPC	Remote Procedure Call	远程过程调用
RPO	Recovery Point Objective	恢复点目标
RTP	Real-time Transport Protocol	实时传输协议
S4	Simple Scalable Streaming System	简单可扩展的流处理系统
SaaS	Software as a Service	软件即服务
SAN	Storage Area Network	存储区域网络
SAS	Serial Attached SCSI	一个用于数据存储的点对点串行协议
SATA	Serial ATA	一种计算机总线接口，一般用于硬盘
SBC	Server-based Computing	应用虚拟化
SDI	Software Defined Infrastructure	软件定义基础设施
SDN	Software-defined Detworking	软件定义网络
SDS	Software-defined Storage	软件定义存储
SGC	Security Gateway Controller	安全网关控制器
SGSN	Serving GPRS Support Node	GRPS服务节点
SLA	Service-level Agreement	服务等级协议
SLB	Service Load Balance	业务负载均衡
SMB	Small and Medium-sized Business	中小企业
SMI-S	Storage Management Initiative Specification	存储管理初始化配置
SMP	Symmetric Multiprocessing	对称多处理
SNAT	Source Network Address Translation	源地址转换
SNAT	Source Network Address Translation	源地址转换
SNMP	Simple Network Management Protocol	简单网络管理协议

(续表)

缩略语	全称	说明
SOA	Service-oriented Architecture	面向服务的架构
SOAP	Simple Object Access Protocol	简单对象访问协议
SPB	Shortest Path Bridging	最短路径桥接
SQL	Structured Query Language	结构化查询语言
SR	Server Receive	服务端接收
SR	Server Room	机房
SR-IOV	The Single Root I/O Virtualization	一种基于硬件的虚拟化解决方案
SS	Server Send	服务端发送
SSD	Solid State Disk、Solid State Drive	固态硬盘
SSH	Secure Shell	安全命令解释器
SSO	Single Sign-on	单点登录
STT	Stateless Transport Tunneling Protocol	无状态的传输隧道协议
TC	Thin Client	瘦客户端
TCM	Thin Client Management	瘦客户端管理
TCO	Total Cost of Ownership	总拥有成本
TCP	Transmission Control Protocol	传输控制协议
TLV	Type-Length-Value	类型 - 长度 - 值
TOR	Top of Rack	机架交换机
TOSCA	Topology and Orchestration Specification for Cloud Applications	云应用程序的拓扑结构和业务流程规范
TPM	Trusted Platform Module	可信平台模块
TRILL	Transparent Interconnection of Lots of Links	多链接透明互联
TXT	Trusted Execute Technology	可信执行技术
UADP	Unified Analysis and Design Platform	统一分析设计平台
UC	Unified Communications	统一通信
UDP	User Datagram Protocol	用户数据报协议
UEFI	Unified Extensible Firmware Interface	统一的可扩展固件接口
UPS	uninterruptible power supply	不间断电源
UVP	Universal Virtualization Platform	华为公司的虚拟化平台
vCPU	Virtual CPU	虚拟CPU
VDC	Virtual DataCenter	虚拟数据中心
VDI	Virtual Desktop Infrastructure	虚拟桌面基础设施(桌面云)
VF	Virtual Funciton	虚拟功能
vFW	Virtual Firewall	虚拟防火墙
VGA	Video Graphics Array	视频图形阵列
VIF	Virtual Interface	虚拟接口
VIM	Virtualized Infrastructure Management	虚拟基础设施管理
VIP	Virtual IP	虚拟IP

(续表)

缩略语	全称	说明
VM	Virtual Machine	虚拟机
VMCS	Virtual Machine Control Structure	虚拟机控制架构
VMDq	Virtual Machine Device Queue	虚拟设备队列
VMM	Virtual Machine Manager	虚拟机管理器
VMX	Virtual Machine eXtension	虚拟机扩展
VNC	Virtual Network Computing	虚拟网络计算
VNF	Virtualized Network Function	虚拟网络功能
VNFFG	VNF Forward Graph	VNF转发图
VNFM	Virtualized Network Function Manager	虚拟网络功能管理器
VNI	VxLAN Network Identifier	VxLAN网络ID
VoIP	Voice over Internet Protocol	IP电话
VoLTE	voice over Long Term Evolution	LTE网络语音业务
vPC	Virtual Port-Channel	虚拟端口通道
VPC	Virtual Private Cloud	虚拟私有云
VPLS	Virtual Private LAN Service	虚拟专用局域网业务
VPN	Virtual Private Network	虚拟专用网
VR	Virtual Reality	虚拟现实
VRF	Virutal Routing Forwading	虚拟路由转发
VRM	Virtual Resource Manager	虚拟资源管理器
vRouter	Virtual Router	虚拟路由器
vSAN	Virtual SAN	VMware公司推出的分布式存储软件
vSGA	Virtual Shared Graphics Acceleration	虚拟共享图形加速
VSI	Virtual Subnet Identifier	虚拟子网标识符
VTEP	VxLAN Tunnel End Point	虚拟隧道端点
vTPM	Virtualizing the Trusted Platform Module	虚拟化可信赖平台模块
VVOL	VMware Virtual Volume	VMware虚拟机卷
VxLAN	Virtual Extensible LAN	虚拟扩展局域网
WDDM	Windows Display Driver Model	桌面显示驱动模型
WI	Web Interface	网页界面
WS	Web Service	Web服务
XPDM	Windows XP Display Driver Model	Windows XP显示驱动模型
YAML		一个可读性高、用来表达数据序列的格式；参考了其他多种语言，包括C语言、Python、Perl，并从XML、电子邮件的数据格式(RFC 2822)中获得灵感
YUV	Y”表示明亮度(Luminance、Luma)，“U”和“V”则是色度、浓度(Chrominance、Chroma)	一种颜色编码方法

后　记

本书在撰写过程中参考和引用了来自互联网和第三方公司或机构的内容，其中包括但不限于：美国国家标准与技术协会(NIST)关于云计算的相关文件、Gartner相关报告、IDC相关报告、Nick McKeown等人于2008年在ACM SIGCOMM发表的题为*OpenFlow： Enabling Innovation in Campus Networks*的论文、Google公司发表的*Data Center as a Computer*一文、微软公司的WINDDK文档、中国移动云计算相关资料、埃森哲的《埃森哲 2012年技术展望》白皮书、蒋清野对开源社区的跟踪研究(http：//www.qyjohn.net/?p=3399)、Apache社区网站发布的文献资料(特别是关于CloudStack、OpenStack、Hadoop、Docker以及流处理相关的文档)、VMware公司网站公开文档、Amazon公司网站公开文档、Cisco公司网站公开文档、H3C公司网站公开文档、Xen开源社区网站公开文档、Linux KVM社区网站公开文档、Intel公司网站公开文档、微软公司网站公开文档、OpenFlow网站文档、Oracle公司网站公开文档、VCE公司网站公开文档、Nutanix公司网站公开文档、HP公司网站公开文档、华为公司网站公开文档、Resource Allocation Algorithms for Virtualized Service Hosting Platforms(Mark Stillwell，2010)、OpenDaylight网站公开文档、Wikipedia网站公开文档(特别是http：//zh.wikipedia.org/wiki/MapReduce)、清华大学出版社出版的《分布式云数据中心的建设与管理》等。同时，我们在撰写过程中采用了互联网搜索工具(如Baidu和Google)搜索查阅了一些文章并引用了部分文字，已难以确定具体出处，无法一一列举。如读者发现未列明出处的引用，可通过出版社告知作者，在本书下一版中会补充到引用说明中或做其他修订处理。

在书中以及本书描述的产品中，出现的商标、产品名称、服务名称以及公司名称，由其各自的所有人拥有。本书内容不构成任何形式的承诺。除非适法要求，作者及出版社对本书所有内容不提供任何明示或暗示的保证。

在法律允许的范围内，本书作者及出版社在任何情况下都不对因使用本书相关内容而产生的任何特殊的、附带的、间接的、继发性的损害进行赔偿，也不对任何利润、数据、商誉或预期的损失进行赔偿。

第 1 章

云计算的商业动力与技术趋势

1.1 云计算基础概念与架构

云户端、服务器端并重的传统模式，向以“广泛的网络接入”、“计算、存储的集中资源池化”、“快捷的弹性伸缩”、“按需自助及可计量的服务”为典型特征的云计算模式的演进铺平了道路，并掀起了正在席卷全球的第三次IT变革浪潮。

在传统IT体系架构下，当前企业基础设施建设与运维所面临的核心痛点问题可以总结概括为如下几点。

1. 平均资源利用率及能耗效率低下

针对基础设施平台建设、扩容与更新换代，当前企业普遍采用的模式是服务器、网络交换与安全，以及存储设备的水平分层采购。各个IT基础设施单部件的选型、数量以及不同部件的组网连接方案均取决于企业IT收集的各业务部门对于IT核心业务处理量需求的预测和规划。同时所有企业IT应用软件、数据库以及中间件软件均采用独占计算、存储和网络资源的烟囱式部署。软件应用与硬件唯一捆绑，不同应用之间无法动态、高效共享相同的计算与存储资源。加之按照摩尔定律不断翻番增长的CPU计算能力已大大超出应用软件对计算资源利用率的同步能力，导致企业IT的平均资源利用率始终处于低于20%的水平。

2. 新业务上线测试周期长，效率低下

企业任何一项新业务上线，从最基础的硬件平台开始，向上逐层延伸至操作系统、中间件、数据库、CRM/ERP/HRM/PDM/Email/UC等各类业务关键软件堆栈，均需要投入IT专业化团队，进行软件安装、调试、功能与性能验证测试、网络配置及修改调整，然后经过若干轮测试、故障及性能稳定性测试定位及重配置和调整之后，才能最终达到期望正式上线运行的成熟度水准。这个过程一般需要长达2至3个月的时间。

资源储备及弹性伸缩能力不足，不具备应对企业IT突发业务高峰处理的能力

针对特定垂直行业短时间内突发性的高流量、高密度业务需求(比如节假日期间对视频网站的突发业务流程冲击)，企业内部物理基础设施资源往往无法满足短时间内迅速获取所需资源的需求，以及处置业务高峰过后的资源闲置问题。

3. 企业核心信息资产通过个人办公PC/便携外泄的安全风险，无法在个人智能终端(平板电脑、智能手机)方便地访问企业防火墙后的工作流及文档

部分企业核心信息资产通过员工个人PC电脑或便携设备外泄给竞争对手，对企业竞争力和商业利益带来负面影响。过分严格的信息安全管控措施又导致了工作效率的下降，企业管理层及员工无法便捷地通过无所不在的网络访问企业防火墙内部的信息资产。

4. 中小型企业希望通过宽带网络管道，从电信运营商或其他主机托管运营商的托管应用数据中心“按需获取”其所需的企业IT应用能力，从而实现日常运作中IT成本开销最小化

数量众多的中小企业，缺少IT领域专业经验，甚至没有财力和精力建设和维持自己专属的IT部门以及IT基础设施平台，普遍希望可以直接从托管运营商那里获取支撑其日常业务运作所需的SaaS服务。

针对解决上述企业IT系统建设和维护过程中遇到的普遍痛点问题，迫切呼唤业界IT软硬件解决方案提供商借助云计算技术，打造TCO、性价比与效率最优的“IT基础设施私有云及公有云”，具体包括：

- 面向大型企业和行业领域提供全自动化管理、一站式交付、支持与企业ITIL无缝集成融合、TCO最优化的端到端解决方案，实现企业传统IT基础设施及应用的改造、扩容和新建；
- 面向中小型企业(SMB)，提供支持多租户安全隔离与动态发放、超大规模资源池调度管理、可最大限度发挥规模经济效益的公有云托管解决方案。

无论上述哪一类形态，企业云计算IT基础设施平台均可定位于基础设施、中间层云平台服务、云计算业务发放与维护管理。针对云平台

服务层之上的多样化的内部IT软件及外部增值业务软件，企业(含运营商)可奉行“深淘滩、低做堰”的原则，广结各方ISV合作联盟，建设依托于云计算平台、繁荣的企业私有云及公有云生态系统。

通过上述私有云/公有云生态系统的建设，使得企业及运营商客户可以真正将“IT基础设施”、“开发部署平台”与“核心业务流程”及“对外服务”解耦，大幅精简企业及运营商的内部IT及对外业务的基础设施层建设部署、运营维护及生命周期管理成本，从而更好地聚焦“核心业务流程”及 “对外服务”的开发与定制，帮助企业和运营商在新形势下获得可持续发展。用一句话高度概括企业云计算IT基础设施平台的核心价值就是：“精简IT，敏捷商道”。

1.2 云计算的商业动力：企业ICT转型

1.2.1 互联网革命

基本上所有企业的第一目标都是追逐利益的最大化。政府与公共事业其实也是如此，只是政府和公共事业的利益是代表着国家利益和全民公共利益的最大化。即使是非营利组织，也会追求在有限投入的情况下，获得最高(数量或质量)的价值输出。所以“利益追求”是商业活动的一个最根本动力。企业对利益的追求，从时间上分为短期利益、中期利益和长期利益。在量化上分为谋基本生存、谋稳步发展、谋快速扩张。企业的商业转型的动力，自然也就来自于两个方面：外部竞争与环境变化导致的企业生存压力，以及更多利益的诱惑。

如今，环顾各行各业，每个企业所处环境和生存压力可能不尽相同。但是如果我们把当今企业所处的环境放到历史发展的画卷里，我们会发现，每个时代的企业所面临的主要挑战、压力与机会，大部分几乎都是相同的，那就是我们经常在教科书里面提到的一次又一次的商业革命、技术革命和政治革命。“革命不是请客吃饭”，而往往是“人头落地”，对应到企业，那就是每次革命都会潮起潮落般伴随着大量老企业的消亡和新企业的兴起。

蒸汽机时代的工业革命让大量的手工作坊消失，取而代之的是小型工厂。电力革命带来流水线作业模式的大规模工厂又同样淘汰了大量的小型工厂，并逐步诞生了行业垄断巨头。20世纪60年代起的第一波信息化革命及计算机革命，让第一代IT企业成为股票市场上耀眼的明星与全球首富的诞生地，很多传统企业紧跟这一轮信息化的浪潮，将计算机广泛应用到业务之中，但相比这些IT新秀，明显显得黯然失色。20世纪90年代起的第二波信息化革命——互联网革命，一大批一夜暴富的互联网新秀应运而生，第一代IT企业虽然没有立即被掀翻在地，但也被迅速地甩到了二流市场地位之中。IT行业之外的传统企业，除了IT行业的东家——金融行业外，很多企业在互联网大潮下已经显出了疲态，无所适从，因为他们虽然建立了互联网网站，却几乎没有从中受益。从2010年前后掀起的第三波信息化革命——移动互联网革命，同时伴随着源自互联网行业的商业模式革命，我们的商业世界正式进入了互联网寡头时代(也可以叫做“大数据时代”)，互联网(寡头)厂商不再是新秀，而已经是一个个的超级巨人，互联网厂商所从事的业务范畴也早已不再是互联网网站(或者某个手机APP)，而是向各行各业、全球各地快速渗透、影响、控制乃至颠覆各类传统企业。

在第三波信息化革命浪潮下，还没有向互联网转型成功的第一代IT企业，已经由二流市场地位，变得更加艰难，普遍出现业绩下滑，股价下跌，正在一路滑向被淘汰或兼并的火山口。IT行业的东家——金融行业也开始凌乱了，因为自己培养出的娃娃们——互联网巨头已经反过来快速渗入到金融行业之中，即互联网金融，使得传统金融行业不得不依仗制度保护延缓互联网金融带来的冲击，完全处于守势地位。

IT与金融以外的一部分传统垄断性行业，通

过非市场化的行业壁垒试图阻隔互联网巨头于行业主营业务之外(如矿山、石油、化工、电力、铁路、电信、政府与公共事业)，但回顾军事历史，早到特洛伊城，后到君士坦丁堡，再到“二战”马奇诺防线，短期可能有效的壁垒，没有一个能最终守住，商业也是一样。一些行业出于上方压力或纳入新技术的兴趣，试图开放部分非关键业务给互联网厂商运营，来拥抱互联网。其实，在大数据时代，一些传统行业甚至无法认识到哪些业务才是未来关键业务，现在开放给互联网行业去做的“非关键业务”往往是未来的关键业务，乃至核心业务，因为往往目前被认为“边缘”的终端用户类业务，才能收集到对未来最有价值的用户数据，而用户数据是未来大数据时代经营的核心。如今被互联网公司卡住用户数据的入口，等同于卡住了企业未来发展的咽喉(出租车行业所面临的互联网冲击就是一个微型的范例)。

而对于非垄断性的传统行业(如商业、服务业、制造业、物流业、饮食行业、非公共教育行业、医疗行业、房地产、影视娱乐业、体育、新闻业、个体农业、游戏业等)，在互联网大潮的冲击下基本上毫无招架之力，在商战上基本上逃脱不开要么“被杀”(企业倒闭)、要么“被俘”(迁移到互联网平台)的命运。互联网时代最先消失的就是渠道企业，之后是百货商场(百货商场现在基本上都转型到餐饮娱乐一条街了，10多年前北京中关村标志性的IT产品商场如今不得不关门歇业)。还没有消失的大量商家、中小企业则或出于服务便利与补贴的诱惑，或几乎别无选择地纳入到互联网运营平台之上(包括开店、日常运营、业务沟通、交易、店面管理等几乎所有经营活动)，成为互联网寡头主导控制下的生态一员，最终逐渐丧失交易权、定价权乃至经营权，但最终还要自负盈亏。为了保持互联网生态的活力，在互联网平台上，依旧会继续上演造富神话(但很难再是“首富”，因为首富是互联网平台的老板)，相反，对于在这个互联网平台上没有成功的企业和个人而言，他们的收益可能连一名互联网公司的员工都不如，保障更是无从谈起。传统行业中的大型企业，在互联网的压力(或诱惑)下，要么成为互联网平台的“VIP用户”被加入到互联网寡头的生态之中，要么靠自身实力进行转型，正面与互联网寡头展开竞争与合作。

在互联网大潮的冲击下，没有哪个行业、哪个企业或个人能够置身事外，其差异只会是受到冲击时间的先后，以及受到冲击的力度与波次。因为这是整个时代的发展与转型的动力。企业要想继续生存和发展，只有面对互联网大潮，进行积极转型，别无他路！

1.2.2 互联网企业的核心竞争力

在分析传统企业互联网转型之路之前，我们需要先分析一下互联网企业为何具有如此强大的杀伤力。

互联网企业和传统企业一样，也是由普通人组成的，他们并不是什么神童，他们的学识、精力、激情、工作效率与创新能力并不比传统企业的员工强大。如此有杀伤力的互联网企业，是竞争胜出的佼佼者。其竞争的原动力是互联网公司创立之初经营过程中外部恶劣环境所带来的生存压力，如2000年前后的互联网泡沫破裂，其导致大量互联网公司倒闭或被兼并(今天的O2O、P2P泡沫也是如此)，驱使互联网行业中生存下来的公司能够健康成长，实力强劲，并最终造就互联网公司让人畏惧的颠覆力。所谓的颠覆力并非一朝一夕养成的，而是经历了一个日积月累的成长过程。这场互联网公司与传统企业之间的竞争，早期有如龟兔赛跑。互联网公司一穷二白，传统企业资金雄厚，人才辈出。但赛跑的中局，乌龟却越跑越快，最终变成了忍者神龟，而兔子却还是那只兔子，兔子也可能没有睡觉，只是没有注意到乌龟在奔跑过程中的变化，如今忍者神龟正在变成智慧的狮子，不仅拥有了力量与速度，还有锋利的牙齿，食肉的性情，将要成为商业森林之王，这时兔子想不注意狮子(当初的乌龟)的动向已经不可能了。

所谓互联网公司颠覆性的竞争力，并非有什么神秘的武器，而是随便一本管理书籍中都

会提到的一些常识点，但是互联网公司通过日积月累，其在每个点上的竞争力与传统企业的差距已经不再是传统企业之间相互竞争的那种10%～20%的差距，而是至少10倍，多则百倍、千倍乃至数十万倍以上的差距，正是这种似乎遥不可及的差距，造成了互联网对传统行业如摧枯拉朽般的颠覆能力。

一、复杂盈利模式

与传统企业向消费者直接销售商品获得收入不同，绝大部分互联网公司从成立之初便向用户提供免费的服务。这给互联网公司带来极大的业务经营压力，一个是盈利模式，一个是经营成本，在这两方面竞争力构建上，互联网公司可谓绞尽了脑汁。

在盈利模式上，相比传统企业，互联网公司最终设计了更为复杂、先进的盈利模式，即我们通常所说的“羊毛出在狗身上，猪来买单”。相比传统企业，这种盈利模式的好处是，用户接受互联网公司的产品和服务几乎没有利益付出的负面障碍(要么无须付费，要么拥有极低的价格，消费者不仅没有觉得有付出，还感觉赚到了大便宜)。而最终为互联网提供免费服务的付费者也认为其每笔付出均有所值(这些付费者可能包括风险投资商、在互联网上做广告的企业，或者在互联网平台进行产品销售省去了渠道成本的企业)。这种复杂盈利模式让互联网公司的用户数量可以呈爆发式的增长，这是传统企业所不能企及的，从而形成对传统企业的一项巨大的竞争优势。如今互联网公司所设计的盈利模式更加复杂，已经不再是简单的羊、狗、猪这种三方的关系了，而是贯穿全产业链、全商业运作的一种生态模式。而传统企业如今还是以简单的商品买卖这种古老的交易模式为主。这种盈利模式的竞争力差异，可能会让传统企业最终落到“把自己卖了，还在替别人点钱”的尴尬境地。

二、极低成本

互联网公司的这种复杂盈利模式，必须在规模效应(需要时间培育)下才能获得回报，而放大规模，需要更高的经营成本。这就要求互联网公司，为了活到能够盈利，必须再绞尽脑汁考虑如何降低自身的运营成本。构成互联网公司的主要成本主要是经营互联网网站所需的IT设备成本、IT软件成本、机房租赁成本、网络成本、办公场地成本、人员成本、市场广告成本等。其中IT设备、软件机房、网络以及运维人力的成本是互联网公司的最大成本源(如今，新业务的现金补贴则成为初期规模化发展成本的大头，这需要和金融手段紧密结合)。对此，互联网公司不能再购买昂贵的商业IT产品，如产自知名IT厂商的小型机、数据库、操作系统等(除非一直拥有丰厚的资金与盈利，但这样的互联网公司很少)。他们只能寻求最便宜的解决方案，那就是我们现在看到的x86硬件和几乎全部基于开源软件一起构建的云计算平台。这个云计算平台相比传统企业使用的小型机、商业数据库、高端存储、商业应用软件等方式，成本至少下降了80%以上。而通过云计算的自动化运营技术，其大幅降低了运维人力的需求，一个运维人员可以管理数千台乃至上万台的IT设备。同时，基于云计算平台，其对机房基础设施也进行了优化改造，降低机房的能耗，即电力成本与场地成本。在业务上层，互联网公司千方百计地推进业务流程自动化的工作，使得大量传统企业人工流程在互联网平台上实现全自动化的处理，大幅降低了业务处理成本。

而传统企业，在丰厚的业务利润的滋养下，以及IT部门所处的企业非核心地位下，根本没有动力和条件向互联网公司那样拼命地降低IT成本。在大企业病的氛围下，传统企业所谓的降低成本，往往仅仅是为了一个漂亮的财务报表。实现每年10%～20%的成本下降幅度，即可以完美地达成当年业绩。在这种冰火两重天的环境下，让互联网公司在10年左右的时间里，大幅度地拉开了与传统企业的成本领先优势。如今，在传统金融行业每发放一笔贷款的成本竟然是互联网金融企业的1000倍。大家如果展开完全竞争，谁输谁赢，将一目了然。我们咋舌于如今互联网颠覆力的同时，不禁感叹，在互联网企业卧薪尝胆的

10年里，很多传统企业却蒙住了自己眼睛，似乎热衷于信息化的同时，却没有看清信息化(互联网)的真正威力。

三、极度的敏捷

早期的互联网公司进入门槛非常低，一台电脑、一台服务器，搭建个网站就可以成立一家互联网公司，没有什么技术专利、商业方法保护之类的障碍。同时，互联网行业又有大量风险投资商的追捧，更使得大量互联网初创企业遍地开花，一种互联网概念推出，立即一大群互联网企业跟进并争相模仿。同时，业务先行互联网公司所拥有的爆发式增长的用户数量会对后来跟进者形成天然屏障，这让互联网公司必须以最快的速度推出新型业务，获得足量用户，才有可能在互联网行业超级激烈(惨烈)的竞争中胜出。数百上千家同质服务的互联网公司中，最终活下来的只有一家到两家(有人经常拿互联网公司大量倒闭的例子，来印证互联网公司也不过如此，并质疑传统企业触网的价值，但其忽略了这正是互联网行业通过大量的优胜劣汰来获得强大竞争力的优势所在)。

为了快速推出业务，互联网公司也是无所不用其极。相比传统企业，互联网公司从人员组织架构、企业文化、经营模式、IT基础设施平台等方面都做了大幅改进。在组织架构上，即使是大型互联网企业，也放弃了传统企业那种逐层审批、大小领导签字画押的环节。业务研发团队自行进行业务决策，缩短内部业务研发外的时间。在企业文化上，为了业务快速上线，没日没夜的工作变得稀松平常，上线后再休息。在经营模式上，也完全打破传统企业把产品完美化再推向市场的策略，而是让位于业务上线时间，互联网公司让业务先上线，通过上线后用户反馈，再继续不断优化产品(规避了传统企业普遍存在的那种闭门造车的模式)。为了加快业务研发进度，在IT基础设施平台方面，互联网公司必须考虑用一个自动化的工具平台，利用这个平台，可以在最短的时间，开发出业务应用，并可以灰度发布，上线后还可以不断地继续完善。这就是我们已经熟知的云计算PaaS平台。经过这种为了加快业务上线速度进行的企业工作流程、组织架构、企业文化、IT平台的重构，互联网企业实现了新业务从研发立项到上线周期不会超过2周，最短只需不到2天的敏捷度。相比传统企业，特别是大型企业那种少则3～6个月，多则1～2年，甚至5年规划的项目，敏捷度提升了至少6倍，多则百倍。传统企业还在研究要不要推出一项互联网业务的时候，互联网企业已经通过这项业务赚到了白花花的银子，等传统企业推出了相同的业务，也基本上属于僵尸业务了(用户已经被先行者抢走而无人问津，微信推出很久之后，一些公司推出X信、Y信业务，最终均销声匿迹就是个鲜活的例子)。

四、真正的创新

在商业社会，“创新程度”与“失败风险”几乎是成正比的。比如，一个产品的新颖程度如果是80%，那么该产品(在市场上)失败的风险也是80%，甚至更高。对于大部分企业，特别是小企业而言，企业的决策者，从事高风险的创新，首先要考虑能否承受失败的风险。一个医药企业，研发并上市一款新药，投资可能是几十亿美元，这也使得传统企业在创新上非常谨慎，进行层层审批和审核。一个创新项目，可能要花费数月甚至数年的筹备过程。在传统行业，相对安全的生存模式是模仿已经成功企业的产品或服务，在地域进行差异化，相同地域则采用价格跟随策略来赚取一定的利润。但互联网行业的环境要比传统行业恶劣，因为互联网无所不在，除非有一些特殊的管制限制，否则很难有地域限制，即使有管制限制，只要遵从相关管制即可自由竞争，这样一家公司的互联网服务即可覆盖全球各个地域。鉴于互联网用户粘性的特点，先行者获得规模用户后，跟进者很难站稳脚跟。所以互联网公司必须通过创新，来寻求其他新的业务领域，才可能生存、发展和壮大。所以创新对互联网公司而言，相比传统企业会更加重要。

那么互联网公司是怎么克服创新风险问题的呢？他们主要从创新成本、创新范围和投资与组织模式几个方面来克服。其核心运作模式类似于风险投资的模式，同时并行投入大量的创新项目(为支撑大量并行创新项目，互联网公司也重构了其人力组织与决策架构)，即使单一项目的失败风险很高，当创新项目数量足够多时，总会有获得成功的项目，而有的项目一旦成功，其获得的回报，会高于对所有项目(包括失败项目)的总投入数倍乃至数十倍，最终通过创新获得收益。同时互联网公司将这个创新平台持续优化。首先这些创新项目所使用的资源最大化共享，比如办公场所、技术人员、IT基础设施(即云计算平台、实验设施等)。创新项目的资源共享降低了整体创新的成本。云计算平台让创新团队只需要几个人，且无须太高的知识水平即可完成快速创新工作，也同样降低了单项目人力成本和时间成本。互联网公司大量采用微创新的模式，也让创新成本获得了下降。如今，互联网公司还引入了生态运营模式，通过定向的开放平台的模式，让社会与产业力量广泛地参与到互联网平台的创新之中，这些基于互联网公司平台创新的企业(多为小企业)和个人，不仅无须互联网公司支付薪金和开发费用，而且在研发阶段还要向互联网公司交纳平台使用费，创新如果失败，互联网公司无须承担任何额外成本，创新成功，互联网公司则与这些创新者进行收益分成，前景看好的创新项目，互联网平台公司甚至可以直接收购。互联网公司这种生态创新的运营模式比公司内部VC风投型运营模式更上一层楼，成为了互联网平台公司一桩稳赚不赔的买卖，还因此披上了“大众创业、万众创新”的光鲜外衣。

目前互联网公司正在探索比生态创新更高级别的创新模式，那就是带有人工智能特性的自动化创新。未来个人商品和服务的需求收集、设计、制造、发货、使用、售后等全环节都不再人工参与，而是全自动化完成。目前这种自动化、智慧化创新模式已经在互联网网页(即人机界面)上在一定程度上实现。互联网公司已经能够为每个用户自动化、智慧化地定制不同的用户交互界面，提供不同定制化服务，而且这种服务在不断优化。目前在线上基础上，互联网公司积极探索线下的智能化创新产品，与工业4.0相结合，走向线上线下的智能一体化创新模式。

与传统企业相比，互联网公司在创新模式、创新速度、创新组织架构、创新生态架构、创新成本控制、智慧创新能力等方面均取得了大幅领先。大部分传统企业，至今还处于尴尬境地——为了一个创新项目内部争论不休，反复调研，评审，申报，层层领导审批背书，最终项目失败。双方差距之大，无须更多笔墨。

五、大数据寡头

如今的商业社会进入大数据时代已经是不争的事实，谁拥有数据，谁将拥有未来。相比传统企业，互联网公司最早意识到了数据的重要性，并几乎从不删除数据。至今，大型互联网公司拥有几千PB的数据已经稀松平常了，领先的互联网公司已经走向EB乃至ZB的数量级。而大型传统企业所拥有的数据量，也不过几PB到几十PB，拥有几百个PB数据的传统企业已经少之又少了。双方仅在数据量上就已经达到上百倍的差距。传统中小企业与互联网公司更没得比了。这还只是数据的数量，还不算质量，在数据质量方面，互联网公司对消费者(个人)信息的掌握更是拥有巨大的优势，大型互联网公司的用户数量都是以亿为单位。个人的几乎所有活动信息都会呈现在互联网之上，包括但不限于个人的姓名、电话、住址、社会家庭关系、活动轨迹、资金关系、资产数额、知识能力、个人喜好、照片、影像、银行账号、社会交流、商务交流等。除了个人信息，还有大量的企业信息，包括企业(特别是网上开店企业)的所有经营活动、资金活动、客户信息、市场状况、销售活动、广告活动等。当所有的个人信息和企业信息汇聚起来，又形成了整个经济数据。任何经济领域的风吹草动，都不会逃过互联网厂商大数据监测与分析系统的法眼，而且在信息获取时间方向上比传统企业或机构大幅领先

(比如某互联网公司公布的大企业景气指数曲线与国家统计局的指数曲线基本相同，但发布时间却比统计局的提前了5个月以上)。互联网企业获取的精准经济数据又可以反过来进行各种金融与市场商业活动。除了数据质量，在大数据处理技术上，互联网厂商也走在了前列。当大部分传统企业还在靠人工进行市场、经营与投资活动的时候，互联网公司已经开始进入了机器智能主导下的信息收集、分析、决策、处理的时代。阿尔法狗战胜围棋高手的案例清晰地反映了人工智能与人工决策之间的差距。当传统企业的决策指挥系统也全面落后于互联网公司的时候，双方在相同游戏规则下，对垒的胜算不会高于1：4，而且传统企业胜利的那一局，可能还是因为互联网公司为了要世界排名而已。其实1：4只是因为下了5局，如果下100局、1000局，数字估计会比1：4难看得多。

以上只是列出了一些互联网企业核心竞争力，还没有罗列互联网企业的其他优势，比如企业文化、技术人才构成、经营创意、薪酬待遇、员工的平均年龄与教育程度、政治地位以及国际化的视野等。互联网的优势明显，并非意味着传统企业毫无机会。因为互联网对传统行业的渗透才刚刚开始。传统行业还有时间(虽然留给我们的时间越来越少)，也有机会进行转身。那么传统企业在如今一波又一波信息革命的浪潮下，如何转型，才不会被拍倒在沙滩上呢？

1.2.3 传统企业的ICT转型

本书虽然是介绍云计算的书籍，但并不是想告诉读者，传统企业上了云计算就能让自己转型成功，能够跟互联网公司一决高下。如果真的这么简单，那么企业ICT转型只需要钱就可以了，这对很多企业都不是什么太大的问题，至少当前现金流都比较充沛。云计算只是企业转型过程中的一个必要的条件。

一、传统企业应先认清自己的落后点

早期互联网引起传统企业注意的时候，传统企业对互联网的认识多停留在互联网网站上面(如今是手机移动APP)，认为互联网就是一个网站或APP，传统企业自己也上马一个网站或手机APP就可以了，但大部分企业并没有在自己的互联网网站上获得什么收益，甚至很多企业的网站只是一个摆设。互联网公司发展到平台化阶段之后，传统的小企业开始大量地加入到互联网平台之中，甚至很多企业让互联网平台成为自己的业务窗口，并因此受益。也就是说，传统企业中的小企业最早认清并获取到了互联网的价值。但小企业无力于自己搭建互联网平台，而最终成为互联网平台生态中的成员。与小企业一样逐渐认知互联网价值的是生产消费品或服务类的大中型企业，比如手机、电脑、家电、服装、玩具等厂商，但这些厂商对互联网关注的焦点在于如何运用互联网，使其成为自己的一个新型产品销售渠道，很少有厂商将互联网定位为业务运营核心去建设和经营。而非直接面向消费者，或对消费者漠然的大中型企业单位，对互联网认知则非常有限，主要原因是互联网公司未切实触动自身利益的时候，几乎很少有企业远见到自己的差距和风险而提前布局。

对于传统企业而言，有个重要的疑问需要解决，那就是为何互联网等信息化平台会成为所有企业的业务核心，比如，一个拖拉机工厂，其核心业务是设计部门、装配生产线、测试与市场等部门，收益贡献来自机械和人，根本不是什么计算机，计算机与IT系统只是提供了一些辅助性工作，如财务、管理、计算等，并非业务核心。解决这个疑问的最佳答案是德国提出并已经局部实现的工业4.0概念。在工业4.0的场景下，一个制造企业，从产品的需求提出，到产品设计、原型生产、小批量试制、中等规模试制、测试验证，到大规模生产、物流仓储，再到市场销售的全环节、全流程，全部通过IT系统与互联网体系主导完成。人力工作只是在当前计算机设备能力有限的设计阶段和流程的规范性方面进行有限干预而已。在过去，一个新型号拖拉机从需求收集、设计、不同批量试制、生产到最终规模下线送到客户手里，可能需要1年的时间。而如今在工业4.0

场景下，可能只需要不到1天。某工业4.0场景下摩托车生产企业，从客户下达订单到定制化的摩托车交付，只需要6个小时。这种相比传统拖拉机厂的极度敏捷能力，就是IT与互联网核心平台支撑的结果。连一个拖拉机厂的未来都是完全信息化(互联网化)的，又有哪些企业不会走上完全的信息化路线呢？

再以与拖拉机厂这种制造企业运营模式似乎风牛马不相及的政府运营为例，政府未来的运营路径又是如何呢？政府作为公益、监管、执法机构，最高效率的运作就是利用互联网信息化的技术手段，打通与国家公民中所有企业、个人信息连接的壁垒，构建一个密集网状的信息化公共治理平台。比如：企业与个人的纳税可以在公共交易平台的交易与支付瞬间同时完成，税率也可以实现高度的定制化、个性化，可以针对不同企业经营与家庭状况收取不同税负并可调节。在这个公共信息化平台之上，所有的企业销售的商品与服务全程可追溯。而(企业)公民双方争议的调解和裁决大部分也可以在公共信息平台上解决，如此样例还有很多。政府达到这样一个公共信息化平台的信息化治理高度，其实不是一个幻想，这种治理模式实际上在一些互联网平台上已经完全或部分成为了现实，只是治理方是互联网公司而非政府而已。

在信息革命的时代背景下，所有传统企业、政府与公共事业都将走向更深度的信息化与互联网化。信息化平台也将成为企业单位的运营核心。而且在这个演进过程中，我们应该清醒地认识到，传统企业被互联网公司落在了后面。作为传统企业，应以最快的速度主动推进自身的ICT转型，让自己在被互联网企业冲击与淘汰之前早日成为互联网化的企业。

二、评估自己的防护壁垒与环境允许的转型窗口期

虽然互联网厂商在向各行各业快速渗透，但出于各行业独有的技术壁垒、监管资质壁垒、资源壁垒、垄断市场壁垒等限制，互联网厂商无法一下子在短时间内通吃一切，气愤之余，也因此给这些壁垒冠以“保护既得利益，阻碍改革深入发展”的帽子。无论如何，这些壁垒给各行业的传统企业一个难得的转型窗口期，一旦壁垒消失，己方的弓箭长矛就完全暴露在对方机枪、大炮与飞机的火力之下了，结果可想而知。不同行业的转型窗口期不尽相同，这跟互联网厂商的基础能力相关。当前阶段互联网公司的基础能力聚集在个人消费者与小企业及个人创业者层面。那么以个人和小企业为目标客户的传统企业，转型窗口期就更短。以大型企业或政府机构为目标客户的传统企业转型窗口期则相对较长。因为经济的运转最终要靠个人，所以任何传统企业都不会有太长的转型窗口期。按照IT更新换代的发展周期预计，短的窗口期也就3～5年，长的窗口期也难以超过10年。以生物技术行业为例，貌似生物学和IT没有特别直接的关系，而如今，最活跃的生物技术创业公司却是在IT行业的大本营——硅谷。这不仅是风投与创新氛围的原因，而且是IT信息化与生物技术深入融合的原因。还有我们看到的生物技术以外的航天技术领域和高铁技术领域，来自互联网行业的马斯克所创立的公司已经成功地部分完成了火箭回收实验和超高速高铁实验，其主导的特斯拉汽车早已商用就更不用说了，Google的无人驾驶汽车也已经可以商用上路。互联网公司还投入巨资，研究机器人技术，而机器人可以运用到几乎所有行业，替代大部分人工作业，从效率、成本、工作品质上均无法比拟。传统企业的决策者如果等看到互联网公司研发的机器人走到自己面前的时候，再考虑转型，可能连拿起电话通知总部的时间窗口都没有了。最具价值的机器人并不是人形的机器，而是机器人背后成千上万台服务器组成云计算平台支撑的人工智能。

三、将自己逐步打造成为立足本行业的互联网企业

很多企业谈到转型的时候，可能是为了给自己打气，增强信心，经常提到“后发优势”，

其实传统企业并没有什么后发优势，反而后发优势往往容易给人投机取巧的感觉。当初互联网公司发展起来，也不是因为什么后发优势，而是其所处的恶劣环境驱使的。现在环境恶劣的似乎不再是互联网公司了，而是传统企业了。对于和互联网公司一样，大部分都是由普通人组成的传统企业而言，转型可能有捷径，但不要指望什么捷径，先要从做强自己开始。要针对互联网公司所取得的那几项核心优势，需检讨自己如何改善。

1. 盈利模式探索

传统企业当前的经营状况良好，这使得传统企业没有动力重新构建新型盈利模式。但是从长远战略角度看，当前盈利模式是否长久，用户粘性是否牢固是个大问题。永远黏住客户的最佳方案，就是让用户没有感觉到付出，却一直能感觉到在获得回报。这是互联网免费的优势。为此传统企业也开始了多种探索，比如某轮胎生产企业，让客户可以选择不再购买轮胎，而是按照轮胎的使用里程付费，没有行使无须付费，让用户觉得更加划算。一个生产空气压缩机的企业，则可以让企业只按照空气供应量付费。那么一个服装企业是否可以免费地提供印上广告的服装给消费者呢，只要穿够多少天即可。诸如此类的盈利模式很多，各类传统企业均可以尝试探索。其实盈利模式是一种创新，当一个企业实现信息化支撑所有业务后，所有盈利模式都仅是一个APP的开发、上线与运营过程而已。但需要强调的是，盈利模式只是企业转型过程中的一环，而非全部，很多企业仅依靠商业模式，而没有强劲的底层技术平台支撑是很难长久的，因为商业模式容易被复制，几乎没有门槛。

2. 成本的竞争力构建

在成本竞争力构建方面，传统企业是最容易提升的，通过构建云计算平台，即可大幅降低IT信息化成本。但这还不够，传统企业需要一直保持信息化平台的成本竞争力。不仅是降低IT基础设施成本，而且需要通过应用的重构，实现业务流程的自动化来降低业务处理的时间成本与人力成本。同时整个IT基础设施平台，要在整个IT生态中保持一种成本的竞争力。传统企业需要具备IT信息化成本控制的主导权，并有能力压低成本，压低成本的方式不是通过议价，而是通过开放架构，并介入IT信息化平台的研发，也就是当无人帮自己降低成本的时候，自己也有能力降低平台成本。IT技术人才队伍的构建，是传统企业在转型过程中急需解决的一个重大问题。

3. 敏捷度提升

云计算平台可以帮助传统企业解决业务的敏捷度问题，包括业务的设计、开发、上线。但对传统企业挑战更大的是整体组织的敏捷度问题。处于转型期的企业，组织结构一直处于快速调整之中，并伴随着大量的优胜劣汰，触动着无数员工的利益。企业越大，组织敏捷度越成问题。企业信息化转型，貌似是IT部门的事情，实际上是企业整体的事情，是企业的关键战略。以目前传统企业IT部门的地位，仅能解决有限的敏捷度问题。IT部门的角色定位需要随着企业对信息化的重要性认识的深入，而不断提升。

4. 创新组织的架构

与组织敏捷遇到的问题类似，基于云计算平台的创新可以在IT部门内进行实验性的验证，若要最终发挥核心价值，需要企业做面向创新的组织架构转型。企业架构要走向扁平化，创新团队独立性更强，需要设立创新容错机制(项目失败是常态)。当云计算(即有限的企业信息化程度)还无法承载所有企业业务时，创新组织架构与创新活动则更是以人为核心的(甚至不是云平台支撑的)。目前一些家电制造企业和服装生产企业便实现了类似的创新型组织模式的转换，在大企业中设立大量小型团队完成从设计到生产交付的全流程工作，大量并行团队，满足了消费者非常个性化的需求，从而达到获取更高效益的目的。有人说这是工业3.5的状态，无论叫什么，这就是企业走向创新型组织的一步非常重要的尝试。当工业4.0条件具备的时候，这种小型创新型组织可以平滑地移植到全信息化处理平台之上(也就是未来的云计算产品创新平台之上)。

对于传统企业而言，组织的重构比IT系统的

重构更加关键，也更加困难。传统企业的ICT转型，必须要迈过组织重构这道门槛，才有可能成为创新型互联网企业。

5. 坚守行业大数据的主导权

从数据量上面看，传统企业无法与互联网公司相比拟，但很多传统企业在数据内容，也就是价值属性上会和互联网公司所拥有的数据区隔开来。因为传统企业具备大量的行业特性数据，比如电力行业拥有的电力数据、气象行业拥有的气象数据、农业拥有的农业数据、医疗行业拥有的医疗数据、石化行业拥有的勘探和化工数据等。另外，因为传统企业信息化深度的发展限制，使得很多传统企业的很多生产经营环节还没有实现完全的信息化，也就是没有变成IT数据，这部分也是传统企业未来数据量的一个来源。只要拥有独有的数据，就可以获得独有的价值，这也是企业能够长期存在的意义。企业需要用经营和长期发展的眼光看待数据对自身的价值和意义。企业或政府单位，可以开放自身数据，但不能丧失数据经营的主导权，丧失主导权，就可能丧失自己未来存在的意义。企业开放数据也是为了让更多外部数据为自己所用的一种数据交换。数据分析与数据经营团队也是企业转型过程中需要建立的一个新型组织，并需要与企业创新型组织相融合。

四、不是拥抱互联网，而是要拥抱未来

有人说互联网只是个工具，也有人说互联网只是虚拟经济。在这里我们要说的是，互联网行业在恶劣的环境下催生了一系列先进生产力与生产关系。而传统企业进行转型的目的，就是利用这一系列互联网行业所拥有的先进生产力与生产关系，去生产包括但不限于本行业的新产品，提供新服务，最终为企业获得更多的效益，并期望基业长青。互联网这些优秀的竞争力必须与企业的业务、企业自身优势资源结合在一起，才能发挥全新的生命力。

IT行业和互联网行业的诞生是信息化革命浪潮中的第一个与第二个波次，下一个波次信息化浪潮可能是延伸到各行各业自身的各个角落，而主导下一波次信息化浪潮的可能就来自各行各业的传统企业。

1.2.4 构建真正开放的新生态

在传统行业，要么是全行业垄断，要么就是全行业自由竞争，生态只是在这个自由竞争的环境下自然生成的产业链，生态链中的企业只要自扫门前雪就可以了，无须关注生态的动向，而在某些生态链条中的垄断厂商一般也只顾自身利益和同行的竞争，最多受限于反垄断法的威严而对利润率有所控制，很少关注和干预其他生态链中厂商的发展，因此传统行业的生态一直存在，但从来没有成为一个热词。但是在互联网及互联网控制下的行业，寡头企业已经能够控制影响整个产业生态链的每一个环节的每一个厂商，并最终发展成为对整个产业生态的经营。这就像一个“土豪”拥有超级的大牧场，牧场里种着各种草木、粮食，养着各种动物，除此之外，还有佃户、商户、工匠、寺庙、集市，所有生物的吃喝拉撒都在这个牧场里完成。牧场主人，那位“土豪”只需要收租子、收粮食，想吃肉了，就抓个动物杀了吃。

传统行业的生态链发展也非常有可能演变为一家独大的全生态经营模式，或者整个传统行业的生态链被纳入到现有的互联网生态平台之中。传统企业的转型发展，要格外注意自己在产业生态链中的地位、影响力与其他厂商的生态关系。最好的结局是做成牧场主，而不是一不小心成为别人生态牧场上的羔羊。

对于传统大型企业，特别是行业垄断性的企业，完全有能力发展一种行业生态经营模式。而对于中小企业，则需要考虑如何把自己的业务平衡地对接在不同的生态系统之间，如果能有块属于自己的独立自留地，自然是再好不过的了，但这个可能性随着不同生态圈之间的竞争会越来越小。总之，将来能成为牧场主来经营生态的传统企业可能少之又少。而对于大部分传统企业而言，加入哪个生态体系才显得尤为重要。

一个理想的产业生态体系该是什么样子的呢？

首先这个生态体系是要有生态聚合力的，也就是加入该生态的成员大部分能够获取相应的好处，并为这个好处而主动聚合在一起，共建这个生态，共同代表生态与其他生态系统进行竞争。

其次，这个生态体系要有内部的自由竞争，有了内部的竞争和流动，内部的优胜劣汰，才能保持生态成员的生命力，进而保持生态整体的竞争力。

第三，这个生态体系是要真正地完全开放，让生态成员自由地出入。这不能像如今一些企业所经营的生态体系那样，表面开放，实际上只是对生态成员(羔羊)加入的开放，而与其他竞争生态之间无法实现业务互通与成员流动，而且成员入驻后基本会逐渐形成业务锁定和业务依赖，而无法脱离该生态。一个开放的生态体系，是保障企业业务经营独立和自由的一个基本要素，如果一个企业丧失业务经营的主导权，那基本意味着企业的主体所有权的实质性被剥夺和转移，最终只能逐渐退化成躯壳乃至最终消失。

第四，这个生态应具备广泛的生态民主性。生态全体成员决定生态的未来，而不是某一家生态成员来独裁。只有这样，这个生态体系才能保障其生态成员利益的最大化。这种生态民主性并非凡事都搞全员公投，而是采用上市公司运作模式，全体成员作为股东选举出董事会、监事会，由董事会组织成立生态的行政经营与管理团队，对生态经营效果负责。

传统企业加入这样一个相对理想的生态体系，才能保障自身利益的最大化。目前这种相对理想的生态体系并非空中楼阁，而是在某些产业领域已经在实践。比如中国浙江某袜业基地的产业运作模式，以及云计算领域的OpenStack社区生态运作模式，就与这个理想生态体系非常接近。这两个生态圈均展现出了独到的生态活力与外部竞争力。

传统企业在转型过程中，要有意识地立足本行业、本专业领域建设类似的生态圈体系，聚合全行业的力量，搭建一个企业群自主经营的独立生态圈，与其他生态体系相互竞争、共同发展，而不是轻易地把企业自己的命运交给一个生态寡头打理。

在完全开放的跨行业生态圈框架下，传统企业可以获得全新的不亚于互联网企业的竞争力。这种竞争力的构建来自于跨产业链企业间的深度融合与协同。这种深度融合与协同，把产业链上下游的企业紧密地联合在一起，将各自的优势发挥到极致。上下游的企业不再是甲方、乙方的买卖关系，也不是各自心怀鬼胎，企图互相控制的关系，而是双方为了共同的目标联合运营，共同创新，共同面向市场，共同应对风险，共担损失，共享收益。为了一个新业务的上线不会再花去几个月的时间去进行采购和商务的谈判，不会再争执业务需求，不会再担心研发能力，甚至不会担心资金缺口。互联网公司所自豪的端到端研发与运营能力以及资金实力，在开放生态中的传统企业联合体同样可以实现。而且这个企业联合体会更具竞争力，因为这个生态更开放，更加敏捷，实力更强，各垂直链条的企业之间可以紧密合作却非紧耦合，横向链条之间的企业又可以充分竞争，在生态竞争维度又可以充分合作。充分的竞争可以让生态的每个环节更强壮，联合体的效率会更高效。充分的合作又会让生态整体实力更强大。而一个垄断生态平台，在高利润的保障下，因缺少改善动力，势必会走向更加封闭和低效。这种开放生态圈框架下企业联合体模式，也已经不是乌托邦了，而是有越来越多的企业在实践，比如IT厂商与电信运营商在全球范围内正在不断组建各种企业联合体进行公有云的运营便是一例。

企业间竞争像是星球的碰撞，而生态竞争像是两个星系的碰撞、融合，是一场更加复杂的竞争模式，孰优孰劣，谁输谁赢，或者最终是否能够共同发展，要看整个生态的经营与运作效果，以及最终的市场表现。生态成员之间以及生态圈与生态圈之间如何在竞争与合作中取得平衡，是生态建设与运营过程中需要不断探索与优化的过程。

1.3 企业云计算的发展趋势

随着云计算技术在各行各业日新月异的发展

与突破，以及云计算产学研生态链各环节持续不懈的“化云为雨”的努力，云计算的应用与价值挖掘已全面渗透到企业IT信息化以及电信网络转型变革的方方面面。各行业、各企业依据自身业务现状、竞争形势与信息化变革程度的不同，不断持续深化企业IT云化的程度，并从一个里程碑走向下一个里程碑。

一、云计算发展的里程碑

从云计算理念最初诞生至今，企业IT架构从传统非云架构，向目标云化架构的演进，可总结为经历了如下三大里程碑发展阶段。

1. 云计算1.0：面向数据中心管理员的IT基础设施资源虚拟化阶段

该阶段的关键特征体现为通过计算虚拟化技术的引入，将企业IT应用与底层的基础设施彻底分离解耦，将多个企业IT应用实例及运行环境(客户机操作系统)复用在相同的物理服务器上，并通过虚拟化集群调度软件，将更多的IT应用复用在更少的服务器节点上，从而实现资源利用效率的提升。

2. 云计算2.0：面向基础设施云租户和云用户的资源服务化与管理自动化阶段

该阶段的关键特征体现为通过管理平面的基础设施标准化服务与资源调度自动化软件的引入，以及数据平面的软件定义存储和软件定义网络技术，面向内部和外部的租户，将原本需要通过数据中心管理员人工干预的基础设施资源复杂低效的申请、释放与配置过程，转变为在必要的限定条件下(比如资源配额、权限审批等)的一键式全自动化资源发放服务过程。这个转变大幅提升了企业IT应用所需的基础设施资源的快速敏捷发放能力，缩短了企业IT应用上线所需的基础设施资源准备周期，将企业基础设施的静态滚动规划转变为动态按需资源的弹性按需供给过程。这个转变同时为企业IT支撑其核心业务走向敏捷，更好地应对瞬息万变的企业业务竞争与发展环境奠定了基础。云计算2.0阶段面向云租户的基础设施资源服务供给，可以是虚拟机形式，可以是容器(轻量化虚拟机)，也可以是物理机形式。该阶段的企业IT云化演进，暂时还不涉及基础设施层之上的企业IT应用与中间件、数据库软件架构的变化。

3. 云计算3.0：面向企业IT应用开发者及管理维护者的企业应用架构的分布式微服务化和企业数据架构的互联网化重构及大数据智能化阶段

该阶段的关键特征体现为：企业IT自身的应用架构逐步从(依托于传统商业数据库和中间件商业套件，为每个业务应用领域专门设计的、烟囱式的、高复杂度的、有状态的、规模庞大的)纵向扩展应用分层架构体系，走向(依托开源增强的、跨不同业务应用领域高度共享的)数据库、中间件平台服务层以及(功能更加轻量化解耦、数据与应用逻辑彻底分离的)分布式无状态化架构，从而使得企业IT在支撑企业业务敏捷化、智能化以及资源利用效率提升方面迈上一个新的高度和台阶，

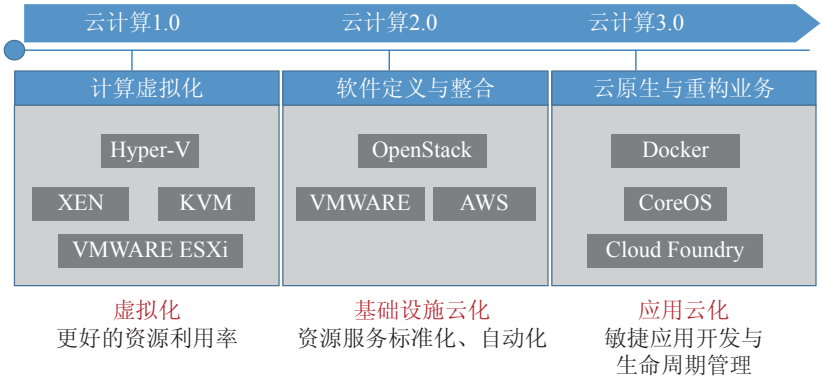


图1-1　云计算发展的三个阶段

并为企业创新业务的快速迭代开发铺平了道路。

针对上述三大云计算发展演进里程碑阶段而言，云计算1.0普遍已经是过去式，且一部分行业、企业客户已完成初步规模的云计算2.0建设商用，正在考虑该阶段的进一步扩容，以及面向云计算3.0的演进；而另一部分客户则正在从云计算1.0走向云计算2.0，甚至同步展开云计算2.0和3.0的演进评估与实施。

二、云计算各阶段间的主要差异

上述云计算里程碑阶段点之间，特别是云计算1.0与2.0/3.0阶段之间的主要差异体现为如下七点。

1. 差别1：从IT非关键应用走向电信网络应用和企业关键应用

站在云计算面向企业IT及电信网络的使用范围的视角来看，云计算发展初期，虚拟化技术主要局限于非关键应用，比如办公桌面云，开发测试云等。该阶段的应用往往对底层虚拟化带来的性能开销并不敏感。人们更加关注于资源池规模化集中之后资源利用效率的提升以及业务部署效率的提升。然而随着云计算的持续深入普及，企业IT云化的范围已从周边软件应用，逐步走向更加关键的企业应用，甚至企业的核心生产IT系统。由此，如何确保云平台可以更为高效、更为可靠地支撑好时延敏感的企业关键应用，就变得至关重要。

对于企业IT基础设施的核心资产而言，除去实实在在的计算、存储、网络资源等有形物理资产之外，最有价值的莫过于企业数据这些无形资产。在云计算的计算虚拟化技术发展初期阶段，Guest OS与Host OS之间的前后端I/O队列在I/O吞吐上的开销较大，而传统的结构化数据由于对I/O性能吞吐和时延要求很高，这两个原因导致很多事务关键型结构化数据在云化的初期阶段并未被纳入虚拟化改造的范畴，从而使得相关结构化数据的基础设施仍处于虚拟化乃至云计算资源池的管理范围之外。然而随着虚拟化XEN/KVM引擎在I/O性能上的不断优化提升(如采用SR-IOV直通、多队列优化技术)，使得处于企业核心应用的ERP等关系型关键数据库迁移到虚拟化平台上实现部署和运行已不是问题。

与此同时，云计算在最近2～3年内，已从概念发源地的互联网IT领域，渗透到电信运营商网络领域。互联网商业和技术模式的成功，启发电信运营商们通过引入云计算实现对现有电信网络和网元的重构来打破传统意义上电信厂家所采用的电信软件与电信硬件紧绑定的销售模式，同样享受到云计算为IT领域带来的红利，诸如：硬件TCO的降低，绿色节能，业务创新和部署效率的提升，对多国多子网的电信功能的快速软件定制化以及更强的对外能力开放。

2. 差别2：从计算虚拟化走向存储虚拟化和网络虚拟化

从支撑云计算按需、弹性分配资源，与硬件解耦的虚拟化技术的角度来看，云计算早期阶段主要聚焦在计算虚拟化领域。事实上，众所周知的计算虚拟化技术早在IBM 370时代就已经在其大型机操作系统上诞生了。技术原理是通过在OS与裸机硬件之间插入虚拟化层，来在裸机硬件指令系统之上仿真模拟出多个370大型机的“运行环境”，使得上层“误认为”自己运行在一个独占系统之上，实际上是由计算虚拟化引擎在多个虚拟机之间进行CPU分时调度，同时对内存、I/O、网络等访问也进行访问屏蔽。后来只不过当x86平台演进成为在IT领域硬件平台的主流之后，VMware ESX、XEN、KVM等依托于单机OS的计算虚拟化技术才将IBM 370的虚拟化机制在x86服务器的硬件体系架构下实现并且商品化，并且在单机/单服务器虚拟化的基础上引入了具备虚拟机动态迁移和HA调度能力的中小集群管理软件(如vCenter/vSphere、XEN Center、FusionSphere等)，从而形成当前的计算虚拟化主体。

随着数据和信息越来越成为企业IT中最为核心的资产，作为数据信息持久化载体的存储已经逐步从服务器计算中剥离出来成为了一个庞大的独立产业，与必不可少的CPU计算能力一样，在数据中心发挥着至关重要的作用。当企业对存储

的需求发生变化时该如何快速满足新的需求以及如何利用好已经存在的多厂家的存储，这些问题都需要存储虚拟化技术来解决。

与此同时，现代企业数据中心的IT硬件的主体已经不再是封闭的、主从式架构的大小型机一统天下的时代。客户端与服务器之间南北方向通信、服务器与服务器之间东西方向协作通信以及从企业内部网络访问远程网络和公众网络的通信均已走入了基于对等、开放为主要特征的以太互联和广域网互联时代。因此，网络也成为计算、存储之后，数据中心IT基础设施中不可或缺的“三要素”之一。

就企业数据中心端到端基础设施解决方案而言，服务器计算虚拟化已经远远不能满足用户在企业数据中心内对按需分配资源、弹性分配资源、与硬件解耦的分配资源的能力需求，由此存储虚拟化和网络虚拟化技术应运而生。

除去云管理和调度所完成的管理控制面的API与信息模型归一化处理之外，虚拟化的重要特征是通过在指令访问的数据面上，对所有原始的访问命令字进行截获，并实时执行“欺骗”式仿真动作，使得被访问的资源呈现出与其真正的物理资源不同的(软件无须关注硬件)、“按需获取”的颗粒度。对于普通x86服务器来说，CPU和内存资源虚拟化后再将其(以虚拟机CPU/内存规格)按需供给给资源消费者(上层业务用户)。计算能力的快速发展，以及软件通过负载均衡机制进行水平扩展的能力提升，计算虚拟化中仅存在资源池的“大分小”的问题。然而对于存储来说，由于最基本的硬盘(SATA/SAS)容量有限，而客户、租户对数据容量的需求越来越大，因此必须考虑对数据中心内跨越多个松耦合的分布式服务器单元内的存储资源(服务器内的存储资源、外置SAN/NAS在内的存储资源)进行“小聚大”的整合，组成存储资源池。这个存储资源池，可能是某一厂家提供的存储软硬件组成的同构资源池，也可以是被存储虚拟化层整合成为跨多厂家异构存储的统一资源池。各种存储资源池均能以统一的块存储、对象存储或者文件的数据面格式进行访问。

对于数据中心网络来说，其实网络的需求并不是凭空而来的，而是来源于业务应用，与作为网络端节点的计算和存储资源有着无法切断的内在关联性。然而，传统的网络交换功能都是在物理交换机和路由器设备上完成的，网络功能对上层业务应用而言仅仅体现为一个一个被通信链路连接起来的孤立的“盒子”，无法动态感知来自上层业务的网络功能需求，完全需要人工配置的方式来实现对业务层网络组网与安全隔离策略的需要。在多租户虚拟化的环境下，不同租户对于边缘的路由及网关设备的配置管理需求也存在极大的差异化，而物理路由器和防火墙自身的多实例能力也无法满足云环境下租户数量的要求，采用与租户数量等量的路由器与防火墙物理设备，成本上又无法被多数客户所接受。于是人们思考是否可能将网络自身的功能从专用封闭平台迁移到服务器通用x86平台上来。这样至少网络端节点的实例就可以由云操作系统来直接自动化地创建和销毁，并通过一次性建立起来的物理网络连接矩阵，进行任意两个网络端节点之间的虚拟通讯链路建立，以及必要的安全隔离保障，从而里程碑式地实现了业务驱动的网络自动化管理配置，大幅度降低数据中心网络管理的复杂度。从资源利用率的视角来看，任意两个虚拟网络节点之间的流量带宽，都需要通过物理网络来交换和承载，因此只要不超过物理网络的资源配额上限(缺省建议物理网络按照无阻塞的CLOS模式来设计实施)，只要虚拟节点被释放，其所对应的网络带宽占用也将被同步释放，因此也就相当于实现对物理网络资源的最大限度的“网络资源动态共享”。换句话说，网络虚拟化让多个盒子式的网络实体第一次以一个统一整合的“网络资源池”的形态，出现在业务应用层面前，同时与计算和存储资源之间，也有了统一协同机制。

3. 差别3：资源池从小规模的资源虚拟化整合走向更大规模的资源池构建，应用范围从企业内部走向多租户的基础设施服务乃至端到端IT服务

站在云计算提供像用水用电一样方便的服

务能力的技术实现角度来看，云计算发展早期，虚拟化技术(如VMware ESX、微软Hyper-V、基于Linux的XEN、KVM)被普遍采用，被用来实现以服务器为中心的虚拟化资源整合，在这个阶段，企业数据中心的服务器只是部分孤岛式的虚拟化以及资源池整合，还没有明确的多租户以及服务自动化的理念，服务器资源池整合的服务对象是数据中心的基础设施硬件以及应用软件的管理人员。在实施虚拟化之前，物理的服务器及存储、网络硬件是数据中心管理人员的管理对象，在实施虚拟化之后，管理对象从物理机转变为虚拟机及其对应的存储卷、软件虚拟交换机，甚至软件防火墙功能。目标是实现多应用实例和操作系统软件在硬件上最大限度共享服务器硬件，通过多应用负载的削峰错谷达到资源利用率提升的目的，同时为应用软件进一步提供额外的HA/FT(High Availability/Fault Tolerance，高可用性/容错)可靠性保护，以及通过轻载合并、重载分离的动态调度，对空载服务器进行下电控制，实现PUE功耗效率的优化提升。

然而，这些虚拟化资源池的构建，仅仅是从数据中心管理员视角实现了资源利用率和能效比的提升，与真正的面向多租户的自动化云服务模式仍然相差甚远。因为在云计算进一步走向普及深入的新阶段，通过虚拟化整合之后的资源池的服务对象，不能再仅仅局限于数据中心管理员本身，而是需要扩展到每个云租户。因此云平台必须在基础设施资源运维监控管理Portal的基础上，进一步面向每个内部或者外部的云租户提供按需定制基础设施资源，订购与日常维护管理的Portal或者API界面，并将虚拟化或者物理的基础设施资源的增、删、改、查等权限按照分权分域的原则赋予每个云租户，每个云租户仅被授权访问其自己申请创建的计算、存储以及与相应资源附着绑定的OS和应用软件资源，最终使得这些云租户可以在无须购买任何硬件IT设备的前提下，实现按需快速资源获取，以及高度自动化部署的IT业务敏捷能力的支撑，从而将云资源池的规模经济效益，以及弹性按需的快速资源服务的价值充分发掘出来。

4. 差别4：数据规模从小规模走向海量，数据形态从传统结构化走向非结构化和半结构化

站在云计算系统需要提供的处理能力角度看，随着智能终端的普及、社区网络的火热、物联网的逐步兴起，IT网络中的数据形态已经由传统的结构化、小规模数据，迅速发展成为有大量文本、大量图片、大量视频的非结构化和半结构化数据，数据量也是呈几何指数的方式增长。

对非结构化、半结构化大数据的处理而产生的数据计算和存储量的规模需求，已远远超出传统的Scale-Up硬件系统可以处理的，因此要求必须充分利用云计算提供的Scale-Out架构特征，按需获得大规模资源池来应对大数据的高效高容量分析处理的需求。

企业内日常事务交易过程中积累的大数据或者从关联客户社交网络以及网站服务中抓取的大数据，其加工处理往往并不需要实时处理，也不需要系统处于持续化的工作态，因此共享的海量存储平台，以及批量并行计算资源的动态申请与释放能力，将成为未来企业以最高效的方式支撑大数据资源需求的解决方案选择。

5. 差别5：企业和消费者应用的人机交互计算模式，也逐步从本地固定计算走向云端计算、移动智能终端及浸入式体验瘦终端接入的模式

随着企业和消费者应用云化演进的不断深入，用户近端计算、存储资源不断从近端计算剥离，并不断向远端的数据中心迁移和集中化部署，从而带来了企业用户如何通过企业内部局域网及外部固定、移动宽带广域网等多种不同途径，借助固定、移动，乃至浸入式体验等多种不同瘦终端或智能终端形态接入云端企业应用的问题。面对局域网及广域网连接在通信包转发与传输时延不稳定、丢包以及端到端QoS质量保障机制缺失等实际挑战，如何确保远程云接入的性能体验达到与本地计算相同或近似的水平，成为企业云计算IT基础设施平台面临的又一大挑战。

为应对云接入管道上不同业务类型对业务体验的不同诉求，业界通用的远程桌面接入协议在

满足本地计算体验方面已越来越无法满足当前人机交互模式发展所带来的挑战，需要重点聚焦解决面向IP多媒体音视频的端到端QoS/QoE优化，并针对不同业务类别加以动态识别并区别处理，使其满足如下场景需求。

- 普通办公业务响应时延小于100ms，带宽占用小于150Kbps：通过在服务器端截获GDI/DX/OpenGL绘图指令，结合对网络流量的实时监控和分析，从而选择最佳传输方式和压缩算法，将服务端绘图指令重定向到瘦客户端或软终端重放，从而实现时延与带宽占用的最小化。
- 针对虚拟桌面环境下VoIP质量普遍不佳的情况，缺省的桌面协议TCP连接不适合作为VoIP承载协议的特点：采用RTP/UDP代替TCP，并选择G.729/AMR等成熟的VoIP Codec；瘦客户端可以在支持VoIP/UC客户端的情况下，尽量引入VoIP虚拟机旁路方案，从而减少不必要的额外编解码处理带来的时延及话音质量上的开销。上述优化措施使得虚拟桌面环境下的话音业务MOS平均评估值从3.3提升到4.0。
- 针对远程云接入的高清(1080p/720p)视频播放场景：在云端桌面的多虚拟机并发且支持媒体流重定向的场景下，针对普通瘦终端高清视频解码处理能力不足的问题，桌面接入协议客户端软件应具备通过专用API调用具备瘦终端芯片多媒体硬解码处理能力；部分应用如Flash以及直接读写显卡硬件的视频软件，必须依赖GPU或硬件DSP的并发编解码能力，基于通用CPU的软件编解码将导致画面停滞、体验无法接受，此时就需要引入硬件GPU虚拟化或DSP加速卡来有效提升云端高清视频应用的访问体验，达到与本地视频播放相同的清晰与流畅度。桌面协议还能够智能识别并区分画面变化热度，仅对变化度高且绘图指令重定向无法覆盖部分才启动带宽消耗较高的显存数据压缩重定向。
- 针对工程机械制图、硬件PCB制图、3D游戏，以及最新近期兴起VR仿真等云端图形计算密集型类应用：同样需要大量的虚拟化GPU资源进行硬件辅助的渲染与压缩加速处理，同时对接入带宽(单路几十到上百M带宽，并发达到数10G/100G)提出了更高的要求，在云接入节点与集中式数据中心站点间的带宽有限的前提下，就需要考虑进一步将大集中式的数据中心改造为逻辑集中、物理分散的分布式数据中心，从而将VDI/VR等人机交互式重负载直接部署在靠近用户接入的Service PoP点的位置上。

另一方面，正当全球消费者IT步入方兴未艾的Post-PC时代大门之时，iOS及Android移动智能终端同样正在悄悄取代企业用户办公位上的PC甚至便携电脑，企业用户希望通过智能终端不仅可以方便地访问传统Windows桌面应用，同样期待可以从统一的“桌面工作空间”访问公司内部的Web SaaS应用、第三方的外部SaaS应用，以及其他Linux桌面系统里的应用，而且希望一套企业的云端应用可以不必针对每类智能终端OS平台开发多套程序，就能够提供覆盖所有智能终端形态的统一业务体验，针对此BYOD云接入的需求，企业云计算需在Windows桌面应用云接入的自研桌面协议基础上，进一步引入基于HTML5协议、支持跨多种桌面OS系统、支持统一认证及应用聚合、支持应用零安装升级维护，及异构智能终端多屏接入统一体验的云接入解决方案——Web Desktop。

6. 差别6：云资源服务从单一虚拟化，走向异构兼容虚拟化、轻量级容器化以及裸金属物理机服务器

在传统企业IT架构向目标架构演进的过程中，为了实现应用的快速批量可复制，以闭源VMware、Hyper-V及开源XEN、KVM为代表的虚拟化是最早成熟和广泛采纳的技术，使得应用安装与配置过程可基于最佳实践以虚拟机模板和镜像的形式固化下来，从而在后续的部署过程中大大简化可重复的复杂IT应用的安装发放与配置过程，使得软件部署周期缩短到以小时乃至以分钟计算的程度。然而，随着企业IT应用越来越多地从小规模、单体式的有状态应用走向大规模、分布式、数据与逻辑分离的无状态应用，人们开始意识到虚拟机虽然可以较好地解决大规模IT数

据中心内多实例应用的服务器主机资源共享的问题，但对于租户内部多个应用，特别是成百上千，甚至数以万计的并发应用实例而言，均需重复创建成百上千的操作系统实例，资源消耗大，同时虚拟机应用实例的创建、启动，以及生命周期升级效率也难以满足在线Web服务类、大数据分析计算类应用这种突发性业务对快速资源获取的需求。以Google、Facebook、Amazon等为代表的互联网企业，开始广泛引入Linux容器技术(namespace、cgroup等机制)，基于共享Linux内核，对应用实例的运行环境以容器为单位进行隔离部署，并将其配置信息与运行环境一同打包封装，并通过容器集群调度技术(如Kubernetes、MESOS、Swarm等)实现高并发、分布式的多容器实例的快速秒级发放及大规模容动态编排和管理，从而将大规模软件部署与生命周期管理，以及软件DevOps敏捷快速迭代开发与上线效率提升到了一个新的高度。尽管从长远趋势上来看，容器技术终将以其更为轻量化、敏捷化的优势取代虚拟化技术，但在短期内仍很难彻底解决跨租户的安全隔离和多容器共享主机超分配情况下的资源抢占保护问题，因此，容器仍将在可见的未来继续依赖跨虚拟机和物理机的隔离机制来实现不同租户之间的运行环境隔离与服务质量保障。

与此同时，对于多数企业用户来说，部分企业应用和中间件由于特殊的厂家支持策略限制，以及对企业级高性能保障与兼容性的诉求，特别是商用数据库类业务负载，如Oracle RAC集群数据和HANA内存计算数据库，并不适合运行在虚拟化上，但客户依然希望针对这部分应用负载可以在物理机环境下获得与虚拟化、容器化环境下相似的基础设施资源池化按需供给和配置自动化能力。这就要求云平台和云管理软件不仅仅要实现物理机资源自身的自动化操作系统与应用安装自动化，也需要进一步在保障多租户隔离安全的情况下实现与存储和网络资源池协同的管理与配置自动化能力。

7. 差别7：云平台和云管理软件从闭源、封闭走向开源、开放

从云计算平台的接口兼容能力角度看，云计算早期阶段，闭源VMware vSphere/vCenter、微软SystemCenter/Hyper-V云平台软件由于其虚拟化成熟度遥遥领先于开源云平台软件的成熟度，因此导致闭源的私有云平台成为业界主流的选择。然而，随着XEN/KVM虚拟化开源，以及OpenStack、CloudStack、Eucalyptus等云操作系统OS开源软件系统的崛起和快速进步，开源力量迅速发展壮大起来，迎头赶上并逐步成长为可以左右行业发展格局的重要决定性力量。仅以OpenStack为例，目前IBM、HP、SUSE、Redhat、Ubuntu等领先的软硬件公司都已成为OpenStack的白金会员，从2010年诞生第一个版本开始，平均每半年发布一个新版本，所有会员均积极投身到开源贡献中来，到目前为止已推出13个版本(A/B/C/D/E/F/G/H/I/J/K/L/M)，繁荣的社区发展驱动其功能不断完善，并稳步、快速地迭代演进。2014年上半年，OpenStack的成熟度已与vCloud/vSphere 5.0版本的水平相当，满足基本规模商用和部署要求。从目前的发展态势来看，OpenStack开源大有成为云计算领域的Linux开源之势。回想2001年前后，当Linux OS仍相当弱小、UNIX操作系统大行其道、占据企业IT系统主要生产平台的阶段，多数人不会想象到仅10年的时间，开源Linux已取代闭源UNIX，成为主导企业IT服务端的缺省操作系统的选择，小型机甚至大型机硬件也正在进行向通用x86服务器的演进。

本书下面的内容将重点围绕云计算出现的这些新变化来讲述云计算的架构技术。