中国科学技术大学

博士学位论文

云计算数据中心结构及其调度机制研究

姓名：刘晓茜

申请学位级别：博士

专业：计算机系统结构

指导教师：杨寿保

2011-03-20

摘 要

自云计算出现以来，经过科学技术的不断发展，经过学术界与产业界的不断推进，云计算的应用正不断发展和深入，云计算也正在从理论走向实践。

随着云计算技术的不断成熟，数据中心也随之发展。今天的数据中心已经不只是一个简单的服务器统一托管、维护的场所，它已经衍变成一个集大数据量运算和存储为一体的高性能计算机的集中地。云计算数据中心涉及十万百万规模的服务器或 PC 机等，资源数量大，异构性强。其中，数据中心的网络结构是设计云计算数据中心必须考虑的重要因素，它为云计算数据中心的高扩展性和资源的高利用率提供充分保障。此外，数据中心引入的副本技术(Replica)虽然是一种提高可用性和性能的重要方法。它弥补了存储对象单点失效、容错性差、接入性能不高等问题；但是也引入以下几个方面的问题：副本一致性问题、负载均衡问题以及由副本产生的各种硬件和通信上的代价问题等等。

本文深入分析和总结云计算数据中心的新特征，从数据中心网络结构的扩展性与绿色节能问题、副本策略问题，以及调度机制出发，紧紧围绕上述三个问题展开研究。本文的主要研究工作如下：

首先，介绍了云计算的概念以及云计算数据中心的概念，针对传统数据中心的不足，着重介绍了云计算数据中心具备的特点。详细分析了由于云计算环境中的资源规模庞大，异构性强等特性产生的三个重要问题：数据中心网络结构的扩展性与绿色节能问题、副本策略问题、以及调度机制。

其次，针对云计算数据中心结构上的可扩展性及绿色节能问题展开研究。充分考虑了新型数据中心应具备的新特点，借鉴已有知名数据中心结构，依据著名科赫曲线，提出了一种新型数据中心网络结构——雪花结构。该结构充分考虑了数据中心的可扩展性，在保证交换机与服务器较低数量比例的前提下，可以在较短的平均路径内实现节点间路由机制，具有较小的网络开销，并降低能耗。

再次，针对云计算中的副本及基于副本的调度问题展开研究。提出了基于副本的调度模型；引入市场机制中的代价因素，通过综合考虑副本地理特性、网络状态、应用服务特点等因素，提出一种代价驱动的自适应副本策略，该策略针对不同应用的一致性与可用性的重要程度，以代价为驱动力来自适应地进行副本复制、副本销毁、副本迁移等操作，以达到负载均衡等目的；提出适用于该模型的，基于副本和数据中心网络结构的调度策略。实验表明，该调度策略可以有效缩短任务调度之前复制副本产生的传输时间，复制副本的平均最短路径仅仅是随机调度算法的一半。

最后，针对调度机制展开研究。为增大服务方的收益，本文从云计算资源提供方的角度出发，充分考虑优先级、时间调度底线、收益、资源风险等调度因素，分别提出云计算环境中的在线调度策略和批调度策略。模拟实验表明，提出的调度策略有效提高了任务完成的总数，增大了服务方收益，降低了服务方接受任务时承担的成本，促进了调度环境的和谐发展。

本文从云计算数据中心网络结构、副本机制、调度机制引出的问题出发，紧紧围绕这三个问题进行了深入研究，提出的数据中心网络结构，副本策略和调度机制不仅解决了相应问题，而且能够为云计算数据中心的后续研究提供借鉴和帮助。

关键词：云计算 数据中心 网络结构 副本 调度策略 在线调度 批调度

ABSTRACT

Since the emergence of Cloud Computing, with the continuous development of science and technology and advance by academia and industry, the applications of Cloud Computing are going on developing. Cloud Computing is moving from theory to practice.

With the development of Cloud Computing, data center is also improved. Nowaday, data center is not only a site which manages and repairs servers, but also a center of many computers with high performance which could compute and store huge data. Cloud Computing data center contains hundrands and southands, even million of servers or PCs. It has many hetergenious resources. How to design the network structure of Cloud Computing data center is a considerable problem. Data center is a key to promise high scalability and resource usage of Cloud Computing. In addition, replica is introduced into data center, which is an important method to improve availability and performance. It covers single point of failure, poor fault-tolerance, low access performance, and so on. However, replica also causes several problems as follows: consistency of replica, load balance, hardware and communication cost, and so on.

This dissertation analyzes and summarizes new characteristics of Cloud Computing data center in deep, and focuses on three main issues: the data center scalability issues with green energy, policy of replica, and scheduling mechanism. The main research contents of the dissertation are as follows:

First of all, the concepts of Cloud Computing and data center are introduced. Aiming at the lack of traditional data center, it focuses on the new features of cloud computing data center. It analyzes three important issues in deep: the scalability and green energy issues of the data center, policy of replica, and scheduling mechanism.

Second, aiming at scalability and green energy issues of Cloud Computing data center, it makes full consideration of new characteristics of Cloud Computing data center, referrs to well-known data center, and presentes a new data center network structure according to famous Koch Curve, that is the Snow structure. It makes full account of the data center’s scalability and low proportion of switches and servers, and can achieve routing within a shorter average path and smaller network overhead.

Again, aiming at replica issues in the Cloud Computing and scheduling problems based on replica, it proposes a scheduling model based on replica. Then, by introducing market mechanisms, it presents a price-driven adaptive replica strategy by

considering geographic features, network status, and application characteristics. The strategy is adaptive to copy, destruct and tansfer a replica for different applications in order to achieve load balance. Based on the model and data center network architecture, a scheduling strategy is proposed. The experiments show that, it could effectively reduce transfering time for copying replica, and the mean path length is just a half of that of random algorithm.

Finally, focusing on scheduling mechanism, to increase providers' revenue, from resource providers' view, this dissertation makes fully account of priority, deadline, profit, and risk, and proposes separately online and batch scheduling policis. Simulation results show that the proposed mechanisms effectively reduce cost of services for providers, increase the total number of completed tasks and profit of providers, and promote the harmonious development of scheduling environment.

This dissertation focuses on Cloud Computing data center, policy of replica, and scheduling mechanism, and makes deep research around such three issues. It not only proposes network structure of data center, policy of replica and scheduling algorithms, but also provides a strong foundation for other related researches on Cloud Computing in the future.

Key Words: Cloud Computing, data center, network structure, replica, scheduling strategy, online scheduling, batch scheduling

图表目录

中国科学技术大学学位论文原创性声明

本人声明所呈交的学位论文，是本人在导师指导下进行研究工作所取得的成果。除已特别加以标注和致谢的地方外，论文中不包含任何他人已经发表或撰写过的研究成果。与我一同工作的同志对本研究所做的贡献均已在论文中作了明确的说明。

作者签名：_____　　签字日期：_____

中国科学技术大学学位论文授权使用声明

作为申请学位的条件之一，学位论文著作权拥有者授权中国科学技术大学拥有学位论文的部分使用权，即：学校有权按有关规定向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅，可以将学位论文编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。本人提交的电子文档的内容和纸质论文的内容相一致。

保密的学位论文在解密后也遵守此规定。

□公开　□保密(____年)

作者签名：_____　　导师签名：_____

签字日期：_____　　签字日期：_____

第1章　绪　论

1.1 论文研究背景

1.1.1云计算的概念

传统模式下，企业建立一套 IT 系统不仅需要购买硬件等基础设施，还要购买软件的许可证，需要专门的维护人员。当企业的规模扩大时还要继续升级各种软硬件设施以满足需要。对于企业来说，计算机等硬件和软件本身并非他们真正需要的，它们仅仅是完成工作、提高效率的工具而已。对于个人来说，正常使用计算机需要安装许多软件，而这些软件有很多是收费的，对不经常使用的用户来说购买这些软件常常是不必要的。如果有这样一种服务，能够提供我们需要的所有软件并供租用，用户只需要在使用时支付少量租金，即可租用到这些软件服务，就能节省许多购买软硬件的资金。

基于这种思想，2006 年，Google 公司推出了“ Google 101 计划 ”，首次提出了云计算（Cloud Computing）概念和理论。这种网络应用模式是基于分布式处理(Distributed Computing)、并行处理(Parallel Computing)和网格计算(Grid Computing)发展而来，是由虚拟化(Virtualization)、效用计算(Utility Computing)、基础设施即服务（IaaS）、平台即服务（PaaS）、软件即服务（SaaS）等概念共同演进及发展的结果。

目前，云计算的概念仍未得到一致认可。通俗的理解是，“ 云 ” 是存储于互联网服务器集群上的资源，它包括硬件资源（服务器、存储器、CPU 等）和软件资源（应用软件、集成开发环境等），本地计算机只需要通过互联网发送一个需求信息，远端就会有成千上万的计算机为用户提供需要的资源并将结果返回到本地计算机；即，通过使计算分布在大量的分布式计算机上，而非本地计算机或远程服务器中，用户（企业或个人）数据的运行将更与互联网相似。这使得用户能够将资源切换到需要的应用上，根据需求访问计算机和存储系统。这样，本地计算机几乎不需要做什么，所有的处理由云计算提供商提供的集群来完成。

在云计算环境下，由于用户直接面对的不再是复杂的硬件和软件，而是最终的服务，因此使用观念会发生彻底变化：从“ 购买产品 ” 转变到“ 购买服务 ”。用户不需要拥有看得见、摸得着的硬件设施，也不需要为机房支付设备供电、空调制冷、专人维护等费用，并且不需要等待漫长的供货周期、项目实施等冗长的时间，只需支付相应费用，即可得到所需服务。

云计算可以按照多种维护方式分类，常见的分类维度包括：

(1) 按照运营和使用者来分类

按照云计算服务的运营和使用对象的不同，云计算可以分为公有云、私有云和混合云。公有云是指企业使用其他单位运营的云计算服务；私有云是企业自己运营并使用云计算服务；混合云是指在企业的云服务中，同时具有内部云服务和外部云服务。目前，Amazon 对外提供大量的公有云服务；Facebook、Yahoo、Google、Baidu、Tencent、淘宝等互联网公司均在积极构建企业内部的私有云系统，以降低企业运营成本；国内外的电信运营商如中国移动、中国电信和中国联通等在对外提供公有云服务的同时也在企业内部积极尝试实施私有云，这种情况即为混合云。

(2) 按照提供的服务类型分类

按照提供的服务类型，云计算可以分为基础架构即服务(Infrastructure as a Service, IaaS)、平台即服务(Platform as a Service, PaaS)和软件即服务(Software as a Service, SaaS)。其中，IaaS 是指以服务的形式提供虚拟基础资源；PaaS 是指提供应用服务引擎，如互联网应用编程接口和运行平台等，用户基于该应用服务引擎，可以构建该类应用；SaaS 是指用户通过 Internet 来按需租用软件。

1) IaaS（Infrastructure as a Service）：基础资源作为服务的模式。云计算服务商以服务的形式提供虚拟硬件资源，如虚拟主机、存储、网络、数据库管理等资源，用户无需购买服务器、网络设备、存储设备等，只需通过互联网租赁即可搭建自己的应用系统。目前 IaaS 是云计算的主要服务类型，Amazon 最早推出虚拟化主机托管(EC2)、对象存储(S3)和结构化数据存储(SimpleDB 和 RDB)的服务，IBM 紧跟其后，推出蓝云计划，并在中国无锡建立了云计算中心，电信运营商也开始逐步涉足云计算服务，如 AT&T、Verizon 相继推出 IaaS 的云计算服务。国内电信运营商中，中国移动也推出 BigCloud 大云计划，并开发了 BC-EC 弹性计算服务。

2) PaaS（Platform as a Service）：平台作为服务的模式。服务提供商提供应用服务引擎，如互联网应用编程接口和运行平台等，开发者基于该应用服务引擎进行应用开发和构建，利用服务提供商的资源向最终用户提供服务。Google AppEngine 是 Google 于 2008 年初发布的一个 PaaS 服务，基于其提供的应用开发接口用户可以利用 Python 和 Java 语言来构建 Web 服务，并在 Google AppEngine 上托管该 Web 服务，系统根据用户使用的资源量（如使用的存储空间、网络流量、CPU 时间等）来按需计费。其他系统，如 Baidu AppEngine、Sina AppEngine、Force.com 等均提供类似的服务和商务模式。

3) SaaS（Software as a Service）：软件即服务的模式。用户通过 Internet (如浏览器)来使用软件，用户不必购买软件，只要按需租用软件。SaaS 是一种广泛的面向应用的互联网服务模式，如 GoogleDocs 是较早推出的 SaaS 云计算服务，

其类似于微软的 Office 的在线办公软件，可以处理和搜索文档、表格、幻灯片，并可以设置共享权限通过网络和他人分享。同时，SaaS 在 CRM 和 ERP 方面也有相当多的服务提供商，如 Salesforce.com 和 Oracle CRM on Demand 等。

1.1.2 云计算数据中心的概念

“数据中心”是以外包方式让许多网上公司存放它们设备（主要是网站）或数据的地方，是场地出租概念在因特网领域的延伸。它是上世纪 IT 界的一大发明，标志着 IT 应用的规范化和组织化。

随着数据中心的发展，尤其是云计算技术的出现，数据中心已经不只是一个简单的服务器统一托管、维护的场所，它已经衍变成一个集大数据量运算和存储为一体的高性能计算机的集中地。各 IT 厂商将之前以单台为单位的服务器通过各种方式变成多台为群体的模式，在此基础上开发诸如虚拟化、云计算、云存储等一系列的功能，以提高单位数量内服务器的使用效率。目前，新一代数据中心（又称云计算数据中心、集装箱式数据中心或者绿色数据中心）的概念仍没有一个标准定义。普遍认为新一代数据中心（朱伟雄，2009）是：基于标准构建模块，通过模块化软件实现自动化 7×24 小时无人值守计算与管理，并以供应链方式提供共享的基础设施、信息与应用等 IT 服务。

新一代数据中心不是单一学科，它应是一个整合的、标准化的、最优化的、虚拟化的、自动化的适应性基础设施(Adaptive Infrastructure)和高可用计算环境。

1.1.3 云计算数据中心的特点

“数据中心”是上世纪 IT 界的一大发明，标志着 IT 应用的规范化和组织化。传统数据中心网络结构（Albert G.，2008）如图 1.1 所示。

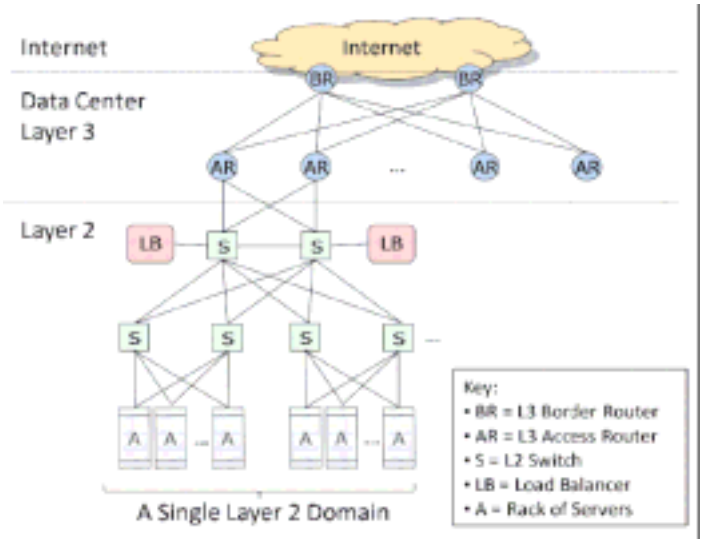


图 1.1　传统数据中心网络结构

来自 Internet 的请求经过第三层的边界路由器（Border Router，BR）和接入路由器（Access Router，AR）被转发到基于虚拟 IP 地址（Virtual IP，VIP）的第二层。处于第二层顶端的交换机连接了两个负载均衡器，若其中一个均衡器失效，另一个仍然可以维持工作。VIP 就配置在这两个负载均衡器中。此外，每个负载均衡器配置了一组目的 IP（Destination IP，DIP），因此一组 DIP 与一个 VIP 相对应，DIP 是机架上物理服务器的私有内部地址。这组 VIP 就定义了服务器资源池，用以解决发送给 VIP 的请求，而负载均衡器则负责在资源池中的 DIP 间传播请求。

然而，随着科技的不断进步，传统数据中心的局限性也逐步显现出来，它存在以下不足之处（牛宪龙，2010；Albert G.，2008）：

- 可扩展性差：服务器数量限制在 4000 台左右，若建立百万规模的数据中心则会导致高额的成本和低效的网络性能。
- 静态的网络分配：为支持数据中心的内部流量，独立的应用通常被映射到特定的物理设备。这种方式基于 VLAN 及其生成算法来满足服务器与特定应用映射。虽然直接映射服务与相关物理设备可以提供一定的安全性和独立性，但是在执行管理和安全性维护时容易造成 VLAN 策略超载。另外 VLAN 生成算法和大服务器池会将流量限制在树的顶端，易造成链路流量超载。
- 资源分片：主流的负载均衡技术通常要求同一个 VIP 资源池中的所有 DIP 在相同的两层中，因此应用需要更多服务器时，不能利用其它两层中的服务器，导致资源分片和低利用率。虽然全 NAT 负载均衡允许利用其它两层中的服务器，但基于 IP 的应用通常要求服务器知道客户端的 IP，而全 NAT 无法支持。
- 人力成本高：当服务需要在服务器间重分配时，传统数据中心网络的地址空间分片会导致巨大的人工配置成本，且人工操作出错的概率很高。
- 硬件成本高：传统数据中心网络使用专用的交换机，位于上层的交换机成本较高。此外，负载均衡器扩容时需要成对更新，成本较高。

因此，针对不断出现的新变化和新需求，云计算数据中心必须满足以下特点（牛宪龙，2010；朱伟雄，2009）：

（1）模块化的标准基础设施：在新一代数据中心中，为使 IT 基础设施简化和具有适应性与可扩展性，需要对服务器、存储设备、网络等基本组成按标准进行模块化配置设计，以使这种配置更易于针对数据中心的服务需求量身打造。基于标准的模块化系统能够简化数据中心的环境，加强对成本的控制，进而实现使用一套可扩展、灵活的 IT 系统和服务来构建更具适应性的基础设施环境，从而提高数据中心工作效率，降低复杂性和风险。

（2）虚拟化的资源和环境：在新一代数据中心中，广泛采用虚拟化技术将

物理资源集中在一起形成一个共享虚拟资源池，从而更加灵活和低成本的使用资源。通过服务器虚拟化、存储虚拟化、数据中心虚拟化等解决方案，不仅可以降低服务器数量，还可以优化资源利用率。虚拟化是新一代数据中心中使用最为广泛的技术，也是与传统数据中心的最大差异。

（3）良好的扩展性：数据中心的可扩展性包括三个方面。首先，物理结构必须是可扩展的。理想的结构必须支持十万甚至百万台服务器的低成本扩展，每个节点的链路数不宜过多或者不依赖于高端交换机。第二，物理结构必须支持增量扩展。当增加新的服务器时，不会影响已有服务器的运行。第三，通信协议设计必须是可扩展的，例如路由协议。

（4）良好的容错性：在当前的数据中心中，故障是非常普遍的。硬件、软件和能源等因素可能引起各种各样的服务器、链路、交换机和机架故障。当网络规模足够大时，单独的服务器和链路的故障甚至比异常发生的频率更高，因此新型数据中心必须具备足够的物理冗余和良好的容错性。

（5）良好的服务器间通信性能：部署在数据中心的许多应用在服务器间流量远大于与外部客户交互的流量，如网页检索、分布式文件系统、科学计算等。因此良好的服务器间通信性能是保障服务 QoS 的基础。

（6）位置无关的地址结构：服务需要采用与物理位置无关的地址结构来解决数据中心对服务器地址的限制问题。这样数据中心的任意服务器都可以成为任意资源池的一部分，既保证了服务的可扩展性又可以提高资源利用率，简化管理配置。

（7）节省能源和空间：传统数据中心设计追求的是性能，而新一代数据中心在当今能源紧缺与能源成本迅猛增涨的情况下必须综合考虑能源效率问题，提高数据中心空间利用率，解决传统数据中心的过量制冷和空间不足的问题。

1.2 研究问题的提出

自云计算出现以来，经过科学技术的不断发展，经过学术界与产业界的不断推进，云计算中的应用正不断发展和深入，云计算也正在从理论走向实践。在云计算的研究和应用过程中，出现的热点问题都在被广泛的讨论和研究中（Armbrust M.，2009；Foster I.，2008；Germain RC，2009；Erickson JS，2009；Leiba B.，2009；Jensen M.，2009）。其中，数据中心网络结构的扩展性与绿色节能问题、副本策略问题，以及调度机制是云计算环境中需要考虑的三个重要问题。

1.2.1 数据中心网络结构的扩展性与绿色节能问题

我们在 1.1.3 中已经讨论了云计算数据中心必须满足的新特点，其中良好的

扩展性和绿色节能是其必备的两大特征。

传统数据中心的服务器数量限制在 4000 台左右，如果建立百万规模的数据中心则会导致高额的成本和低效的网络性能。因此，传统数据中心的可扩展性较差。云计算数据中心包含十万百万规模的服务器或 PC 机等，其物理结构必须是可扩展的，并且支持增量扩展。这样不断新添的机器资源不会影响已有资源的运行，数据中心整体运行效率高，资源利用充分。理想的云计算数据中心结构可以支持十万，甚至百万级服务器，其成本低，扩展性高，并且每个节点的链路不会过多，也不依赖于高端交换机。因此，如何设计云计算数据中心的网络结构，使其具备高扩展性成为重要研究问题之一。

此外，过去两年里随着碳排放和电费的不断激增，使得绿色节能数据中心的研究发展成为热点。数据中心是提高能源效率最困难的商业建筑，因为安装在数据中心的计算机需要大量的电力并且排放出大量的热量，而数据中心消耗的一半电力都用于支持 IT 设备的电力系统和制冷空调等冷却基础设施。

美国联邦环境保护署（Environmental Protection Agency，EPA）于 2009 年 8 月提交了一份关于数据中心的报告。这份报告称，数据中心的能源消耗从 2000 年至 2006 年增长了一倍，预计到 2011 年再增长一倍。此外，全球权威机构 Gartner 调查也显示，IT 行业每年的二氧化碳排放量约为 3500 万吨，占全球总排放量的 2%，数据中心成碳排放大户。企业每年在用电成本上的花费已经大于当年硬件设备投资额。面对如此严峻的形势，节能已经成为建设绿色数据中心的重点。从机房用电分配看，服务器设备占电能总能耗的 52%，而制冷系统和电源系统各占 38%和 9%，照明系统仅占 1%。

1.2.2 数据中心中的副本问题

副本技术(Replica)是一种提高可用性和性能的重要方法（Andrew D.，1982）。副本弥补了存储对象单点失效、容错性差、接入性能不高等问题。但是，引入副本机制也必然带来以下几个方面的问题：副本一致性问题、负载均衡问题以及由副本产生的各种硬件和通信上的代价问题等等。同时，由于云计算自身的一些新特性，使得云中的副本问题存在一些新的挑战：

网络的广域性与动态性。云计算服务提供者通常在全球范围内的适宜地点建立大型数据中心来向广大客户群提供服务。一方面，数据中心的地理分布性即网络的广域性，使得副本策略不得不考虑地理特性带来网络延迟的影响，这其中也包括带宽所带来的负载均衡方面的影响。另一方面，由于数据中心大多使用廉价的 PC 机（Nicolas B.，2009；Pinheiro E.，2007），数据中心错误发生较为频繁，如电源错误、机架错误、网络错误、硬盘错误、系统过热、网络攻击、自然灾害等。网络变化较为频繁，从而使网络中单个副本失效成为一种常态。因此，对于

云计算环境中的副本策略来说，网络广域性和动态性使得副本策略需要考虑副本失效和网络延迟带来的影响。

商业利益追求。云计算服务提供商追求的目标是在保证服务质量的前提下实现利益最大化。保证服务质量的一个重要的前提是提高可用性和保证高一致性，而追求两者必然带来的是开销代价的增大。因此，如何在满足用户体验的前提下，减小开销使利益最大化，是云计算服务提供商面临的问题。

在数据中心海量资源中，如何放置副本，以及放置多少个副本是值得研究的热点问题，而基于副本的调度问题也随之而来。

1.2.3 数据中心基于资源性能的调度问题

云计算的目的就是实现协同工作和资源共享，而云计算环境中各式各样机器资源体现出来的异构性和动态性以及用户需求的多样性，使得云计算环境下的资源管理变得异常复杂。调度问题一直是资源管理中的热点研究问题。由于云计算的自身特性，云中的调度问题也出现了新挑战：

海量异构资源的利用率。云计算服务的成功与否很大程度上取决于数据中心资源利用率的高低，而数据中心海量异构资源利用率的高低与用户任务的调度机制又有很大关联。采用哪些高效的调度机制可以有效提高云计算中的资源利用率是云计算服务提供商需要解决的问题。

用户对“服务”类型的选择。云计算服务的用户众多，如何在保证资源利用率的前提下，满足用户的多种需求。例如，选择在线调度直接服务用户；还是选择批调度，将服务“打包”给用户完成任务；选择哪些在线调度或者批调度策略都是需要关注的问题。

此外，现有云计算环境中，调度策略主要着眼于资源的分配管理，往往以满足用户的各种资源请求为目的，从用户的 QoS 需求出发，以满足用户的各种资源请求为目的，而对于云计算服务提供方关注不够。因此，如何从云计算服务提供方的角度出发，充分考虑任务优先级、时间调度底线、服务方收益、资源风险等调度因素，将用户任务合理调度和执行是值得研究的问题。

1.3 论文的主要研究工作

1.3.1 论文研究内容

云计算数据中心涉及十万百万规模的服务器或PC机等，资源数量大，异构性强。本文围绕云计算数据中心的特点，研究新型的云计算数据中心网络结构、数据中心的副本策略，以及调度机制。本论文的主要研究内容包括以下几个方面：

第一，研究雪花结构的新型云计算数据中心网络结构。

网络结构是设计云计算数据中心必须考虑的重要因素，它为云计算数据中心的高扩展性和资源的高利用率提供充分保障。针对云计算数据中心结构上的可扩展性及绿色节能需求，充分考虑了新型数据中心应具备的新特点（1.1.2部分已介绍），借鉴已有知名数据中心结构，依据著名科赫曲线，本文研究和设计新型的数据中心网络结构，提出雪花型数据中心结构。该结构充分考虑数据中心的可扩展性，在保证交换机与服务器较低数量比例（0.125-0.333范围内）的前提下，可以在较短的平均路径内实现节点间路由机制，具有较小的网络开销。

第二，研究代价驱动的自适应副本策略。

现有的云计算环境中副本管理机制均相对简单，副本阶数、位置相对固定，对副本在整个生命周期中的代价和效率未做充分考虑。针对上述问题，借鉴分布式系统中的一些已有成果，研究并设计云计算环境中代价驱动的自适应副本策略。该副本策略通过引入市场机制中的代价，综合考虑负载均衡及一致性与可用性的平衡，对副本进行自适应的操作，达到最小化副本开销，最大化副本收益的目标。

第三，研究基于副本的调度模型及相关调度策略。

针对云计算环境引入的副本机制，研究基于副本的调度模型，以及基于副本和数据中心网络结构的调度策略。基于副本的调度模型综合要考虑云计算环境中的计算资源和存储资源，以SLA控制器管理用户需求，以资源信息控制器管理空闲资源，以副本信息控制器管理副本信息。由Cloud控制器负责综合资源信息控制器和副本信息控制器返回的信息，采用基于副本和数据中心网络结构的调度策略，调度执行任务。

第四，研究基于成本计算的在线调度策略。

现有云计算环境中，在线调度策略主要着眼于资源的分配管理，往往以满足用户的各种资源请求为目的，而对于服务方关注不够。为增大云计算服务提供方的收益，本文研究云计算环境中基于成本计算的在线任务调度策略。根据用户提交任务的相关信息，计算接受任务的沉没成本和机会成本以决定是否接受任务，使得服务提供方和服务请求方都实现自身的经济目标，促使云市场环境向健康稳定的方向发展。

第五，研究服务质量驱动的批调度策略和基于层次分析法（The Analytic Hierarchy Process，AHP）的批调度策略。

现有云计算环境中，批调度策略的研究主要着眼于资源的分配管理。针对任务优先级、时间调度底线、云服务提供方收益、资源风险等调度因素，研究服务质量驱动的批调度算法和基于 AHP 的动态级任务调度算法。这两个调度算法均从云计算服务提供方的角度出发，有效提高任务完成的总数，增大服务方收益。

本文工作受以下项目资助：

（1）《网络计算环境中信任感知的资源交易模型》

国家自然科学基金面上项目（编号：NSF60673172）

（2）《基于应用调度的网格服务环境及若干网格应用的研制》

国家 863 高技术研究与开发计划项目（编号：2006AA01A110）

（3）《校园云网络及其典型应用学伴平台的研究》

中国科学技术大学研究生创新基金项目（编号：KD0901110）

1.3.2 论文的组织结构

本论文组织结构如图 1.2 所示。

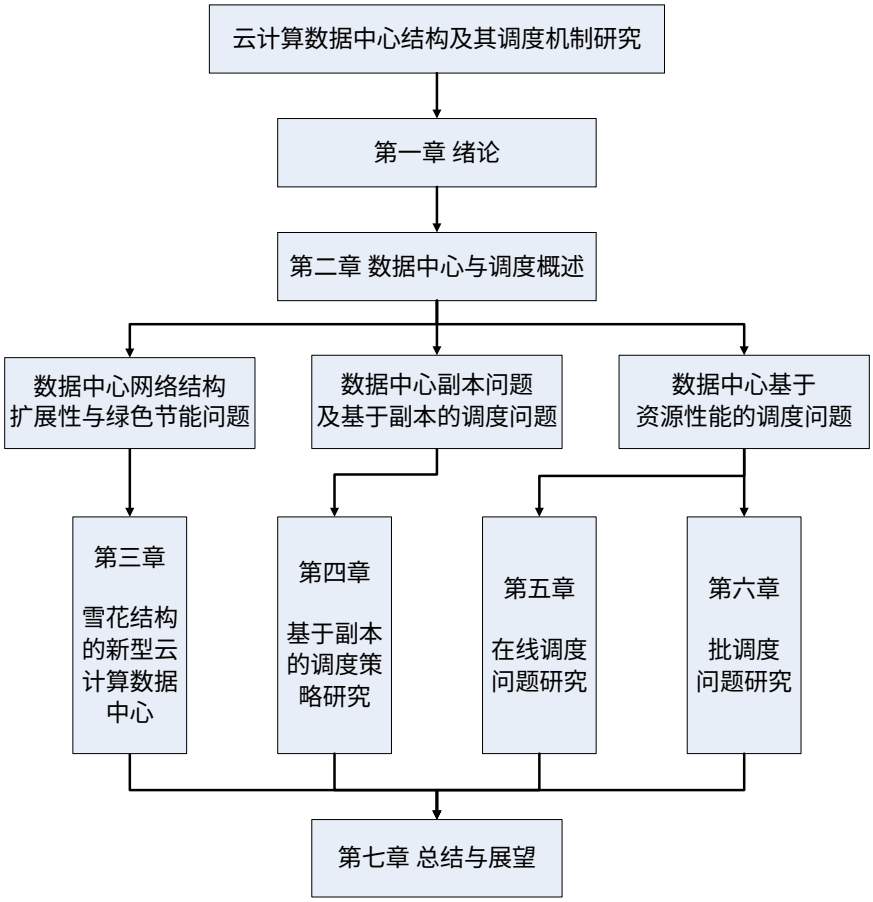


图 1.2　论文组织结构

除了第 1 章绪论之外，本文其他主要章节安排如下：

第2章　分为两大部分，首先介绍数据中心的演变、分类、分级情况，阐述新一代数据中心现状；接着介绍调度问题中的基本术语，以及云计算环境中的调度新特征和调度机制研究现状。论文的后续部分将围绕云计算数据中心和调度机制展开研究。

第3章　针对云计算数据中心结构上的可扩展性及绿色节能需求，充分考虑

新型数据中心应具备的新特点，借鉴已有知名数据中心结构，依据著名科赫曲线，提出新型数据中心网络结构（雪花结构）。该结构充分考虑数据中心的可扩展性，在保证交换机与服务器较低数量比例（0.125-0.333范围内）的前提下，可以在较短的平均路径内实现节点间路由机制，具有较小的网络开销。

第4章　针对云计算中的副本及基于副本的调度问题，首先提出基于副本的调度模型；接着引入市场机制中的代价因素，通过综合考虑副本地理特性、网络状态、应用服务特点等因素，提出一种代价驱动的自适应副本策略，该策略针对不同应用的一致性与可用性的重要程度，以代价为驱动力来自适应地进行副本复制、副本销毁、副本迁移等操作，以达到负载均衡等目的；最后提出适用于该模型的，基于副本和数据中心网络结构的调度策略。

第 5 章　提出云计算环境中基于成本计算的在线任务调度策略。首先详细阐述服务方接受首个任务的调度策略，引入沉没成本，接着分析在计算沉没成本和机会成本的前提下，服务方接受非首个任务的调度策略，最后比较沉没成本与机会成本的关系，分析沉没成本和机会成本对服务方收益的影响。服务方收益最大化机制能有效降低服务方接受任务时承担的成本，促进云计算环境的和谐发展。

第 6 章　本章充分考虑优先级、时间调度底线、收益、资源风险等调度因素，提出服务质量驱动下的任务调度算法和基于 AHP 的动态级任务调度算法。这两个调度算法均从云计算资源提供方的角度出发，有效提高任务完成的总数，增大服务方收益。

第 7 章　给出总结与展望，总结本文的主要工作与贡献，并且对下一步有价值的研究点进行讨论。

1.4 本章小结

本章首先介绍了云计算的概念以及云计算数据中心的概念，然后针对传统数据中心的不足，着重介绍了云计算数据中心具备的特点。第二小节详细分析了由于云计算环境中的资源规模庞大，异构性强等特性产生的三个重要问题：数据中心网络结构的扩展性与绿色节能问题、副本策略问题、以及调度机制研究。接着，引出本文的研究内容，最后是本文的组织结构。

第2章　数据中心与调度概述

2.1 数据中心概述

2.1.1 数据中心的演变

随着科学技术的不断进步，数据中心也在不断地演变和发展。从功能的角度来看，可以将数据中心的演变和发展分为四个阶段。

数据中心经历的第一个阶段称为数据存储中心阶段。在这一阶段，数据中心承担了数据存储和管理的功能。因此，数据中心的主要特征仅仅是有助于数据的集中存放和管理，以及单向存储和应用。由于这一阶段的数据中心功能较为单一，因此其对整体可用性需求也较低。

数据中心发展的第二阶段称为数据处理中心阶段。在这一阶段，由于广域网、局域网技术的不断普及和应用，数据中心已经可以承担核心计算的功能。因此，这一阶段数据中心开始关注计算效率和运营效率，并且安排了专业工作人员维护数据中心。然而，这一阶段的数据中心整体可用性仍然较低。

接着，数据中心进入应用阶段，需求的变化和满足成为其主要特征。随着互联网应用的广泛普及，数据中心承担了核心计算和核心业务运营支撑功能。因此，这一阶段的数据中心又称为“信息中心”，人们对数据中心的可用性也有了较高的要求。

数据中心的第四阶段称为数据运营服务中心阶段。在这一阶段中，数据中心承担着组织的核心运营支撑、信息资源服务、核心计算，以及数据存储和备份功能等。业务运营对数据中心的要求将不仅仅是支持，而是提供持续可靠的服务。因此，这一阶段的数据中心必须具有高可用性。

2.1.2 数据中心的分类

各类数据中心的业务各异，其地位、规模、作用、配置和分类方法也有很大的不同，目前主要从以下两方面进行分类。

1．根据数据中心服务的规模分类

数据中心按照规模划分，可以划分为大、中、小型数据中心，但这也只是一个相对的概念，没有严格的量化标准。在我国，从规模上来分，省、部级以上级别（或相当级别）的企业与机构所建立的数据中心一般属于大型数据中心；省辖市级（或相当级别）的企业与机构所建立的数据中心一般属于中型数据中心；县辖级（或相当级别）的企业与机构及小型企业所建立的数据中心一般属于小型数据中心。

2．根据数据中心服务的对象和范围分类

根据数据中心服务的对象和范围，常常将数据中心分为企业数据中心和互联网数据中心。

（1）企业数据中心（CorporateDataCenter,CDC）。国内也称 EDC（EnterpriseDataCenter），泛指由企业与机构所有和使用的数据中心，他们的目的是为自己的组织、合作伙伴和客户提供数据处理和数据访问的支撑。企业内部的 IT 部门或合作方负责数据中心设备的运行维护。企业型数据中心是一个公司的内部网、互联网访问、电话服务的核心。

（2）互联网数据中心（InternetDataCenter，IDC）。指由服务提供商所有，并向多个客户提供有偿的数据互联网服务（如 Web 服务或 VPN 服务等）的数据中心。互联网数据中心是一种利用电信级机房设备向用户提供专业化和标准化的数据存放业务及其他相关服务的中心。用户可以享受数据中心的主机托管、整机租赁、虚拟主机等服务，也可以租用数据中心的技术力量来搭建自己的互联网平台。国际互联网设施包括传统的电话服务商和相关的商家，云计算数据中心即属于此种类型。

2.1.3 数据中心的等级

国际正常运行时间协会（The uptime institute，UI）将数据中心分为如下四个等级：

（1）　基础级。

这一级的数据中心有电力配送和制冷设备，也许有发电机或架空地板。然而，数据中心内设备属于单模块系统，因此具有多处单点故障。为了预防这些故障，需要手动关闭数据中心内的设施。一般来说，平均每年关闭一次；在特殊情况下，可能发生更频繁的关闭。此外，这一级数据中心的可用性只有 99.671%，自然故障以及对各个部件的错误操作都会导致整个数据中心运行中断。

（2）　冗余部件级。

顾名思义，这一级的数据中心包含一部分冗余部件，因此运行中断的可能性低于第一级数据中心，可用性为 99.741%。配备的发电机为单回路设计，因此仍具有单点故障的可能。此外，对基础设施和关键电路维护时仍需要关闭相关设备。

（3）　可并行维护级。

这一级的数据中心有了显著改善。自然故障或操作错误会引起数据中心运行中断；然而，保护性的和程序式的维护、维修和元件替换，增加或者减少与处理能力相关的部件，对部件和系统进行测试等活动已经不需要中断硬件设备。因此，数据中心的可用性得到提升，为 99.982%。此外，当客户的业务需求允许增加成本进行更高级保护时，第三级数据中心通常可以升级到第四级数据中心。

（4）　容错级。

这一级的数据中心要求所有计算机硬件具备双电源输入，并且任何活动均不会引起关键负载的中断。因此可用性得到大大提升，为 99.995%。此外，基础设施的容错能力也能够容错至少一次最糟糕情况，如设备故障等。根据消防和供电安全规范的要求，还会有由于火灾报警或启动了紧急停电程序而导致的停机事件的发生。云计算数据中心应属于第四级数据中心。

2.1.4 新一代数据中心现状

传统的数据中心往往让人们联想起宽敞的机房和排列整齐的机架，而近年来出现的新一代数据中心是将服务器、存储、网络设备等全部集成装箱处理。新型数据中心优点突出，其模块化设计可以实现系统的快速、灵活部署，不仅可以大幅降低建设成本，而且能够大大缩短数据中心的建设周期以及满足临时扩容的需求。此外，新一代数据中心对环境的要求不高，几乎可以用于任何环境，尤其是在一些比较恶劣的自然环境中，更能充分体现其易于部署和使用的特点。目前，众多企业，如 IBM、HP、Google、SGI、思科、微软等巨头纷纷涉足。

（1）Sun 的“ Blackbox ”数据中心

2006 年 10 月，Sun 在“ 黑盒子计划 ”中提出了集装箱数据中心的概念，即将构成数据中心的一些基本原件，包括计算机硬件、供电和冷却设备等全部装入一个 20 英尺长、8 英尺宽、8 英尺高的标准集装箱中。这样的设计带来很多优势：易搬运、低成本、建设速度快，并且不受场地限制，利用废弃的场地，可以在短期内快速构建起一个数据中心。

图 2.1 Sun 的“ Blackbox ”数据中心

图 2.2 IBM“ 便携式模块化 ”数据中心

Sun 的“ 黑盒子 ”可以容纳最多 250 台 Sun Fire T1000 或 x64 服务器，最高 7TB 内存，1.5PB 硬盘存储空间或 2PB 磁带存储空间，只要接通网络、供电和冷水管线就可以开始运作。“ 黑盒子 ”可以容纳近 10 吨的硬件设备，可以承受 9 倍地球重力的冲击。

“ 黑盒子 ”具备便携可搬运的特点，对于那些因为工作需要将计算设备转移

到灾难现场，或是因拓展业务需要将计算设备进行远距离搬运的企业来说是个好选择。第二，成本低，可快速搭建。传统数据中心的造价一般都很昂贵，从设计到建造要耗费很长时间，而采用“ 黑盒子 ”成本低，并且这种模块化的设计便于搭建和扩充，不受空间和地域限制。对于用户来说，这种一体化的数据中心也免去了自行建造中的不少麻烦。

（2）IBM“ 便携式模块化 ”数据中心

IBM 于 2008 年推出其模块化数据中心产品。2009 年 12 月，IBM 在拉斯维加斯举办的第 28 届 Gartner 数据中心大会上展示了名为“ Portable Modular Data Centre(PMDC，便携式模块化数据中心) ”的数据中心。

IBM 数据中心密封（现有 20 英尺和 40 英尺两种型号）而且隔热，适应于多种环境——无论有没有电、没有有冷却装置，甚至在冰原和沙漠。除了 IBM 自己的服务器机架、存储和网络设备，以及管理系统等，还密封了电源设备、冷却设备、紧急供电系统以及合作伙伴的设备。此外，还拥有与传统数据中心同样的控制功能，包括磁卡读卡器、生物扫描仪、数字小键盘，内部外部具有闭路电视监控。

据 IBM 称，和具有大量空调甚至活动地板和天花板的传统数据中心相比，便携式模块化数据中心在设计和建造上要便宜 30%。此外，利用便携式模块化数据中心可以在 8 到 12 周内建立 500 至 2500 平方英尺的数据中心，建造周期远远快于传统数据中心。并且建设的地点更加灵活，如果有需要，甚至可以在停车场建立。

（3）惠普 POD 数据中心

2008 年 7 月，惠普推出了名为“ Performance Optimized Data Center ”（POD）的数据中心。8×40 英尺的 POD 设备提供了 12TB 的存储空间。此外，POD 还预装了惠普 Insight Manager 或者 OpenView 服务器管理软件。

POD 内安装了用于优化散热和冷却的冷热通道，从一端排放的热空气被吸入顶部的热交换器，经过冷却之后作为冷空气被回送到另一端。在能源密度上，POD 要远远高于传统数据中心，据惠普介绍，传统数据中心每平方英尺功率为 200~300 瓦，而 POD 可以达到 1800 瓦。另外，POD 从下订单到供货只需要 6 周时间，可以让用户快速的搭建出一个数据中心。

2010 年初，惠普又推出了 20 英尺的数据中心小型版本，外形尺寸只有首款产品的一半大，不仅价格更低，而且更容易运输。此外，它可以为每个机柜提供 29KW，最高可达 34KW 功率，POD 的 PUE(每效率单位耗电量)比值为 1.25，而传统数据中心每个机柜的供电量只有 12KW 左右。小型 POD 装有 4 个可变速风扇，还配有测温感应器，风扇速度可根据每个机柜的实际温度进行调节。

（4）思科数据中心

2009 年，思科公司宣布推出统一计算系统（Unified Computing System，UCS），试图通过 UCS 进军数据中心市场。UCS 在单一的高能效系统中统一了计算、网络、存储访问和虚拟化资源，打包成一个整体的解决方案。以 UCS 为中心，思科与 IT 行业的领先企业建立了全面的协作和开发关系，包括 BMC Software、EMC、英特尔、NetApp 和 VMware 等，开始推出模块化的数据中心。

美国宇航局位于加州的 Ames 研发中心采购了思科的统一计算系统 UCS，用于 Nebula 云计算项目。

思科集装箱数据中心有 40x8 英尺和 20x8 英尺两种产品，一种是为船只服务，另一种服务于卡车运输。所有必要的服务器，存储和网络设备都被集成到集装箱中。据思科官网介绍，思科数据中心是其政府行业解决方案的一部分，可以在 12 到 16 周时间内完成部署，其 PUE 值在 1.05 到 1.25 之间，相比传统数据中心更低。

（5）Google 数据中心

Google 数据中心服务器采用 2U 结构，支持两颗 CPU、两块硬盘，具有 8 个内存插槽。事实上，从 2005 年开始，Google 的数据中心就开始采用标准的集装箱设计了。Goolge 的每个集装箱可以容纳 1160 个服务器，具有 250KW 的功率，每平方英尺最高具有超过 780W 的功率密度。

Google 数据中心的设计着重于“ 电源在上，水在下 ”，机架从集装箱的天花板悬挂下来，冷却设备在机架下面，让冷空气通过机架。冷却风扇速度可变，并可以精确管理，保证风扇在能够冷却机架的前提下运行在最低速度。

图 2.3 微软数据中心

图 2.4 SGI“ 冰立方 ”数据中心

（6）微软数据中心

芝加哥数据中心是微软最大的数据中心，占地面积 70 万平方英尺，一层就像停车场，停放着几辆拖车，上面放着集装箱。这些集装箱放置着微软云计算产品的重要组件，每个拖车的集装箱都存放了 1800 至 2500 台服务器，每台服务器都可以用来为微软的云计算操作系统 Windows Azure 收发电子邮件、管理即时通

信或运行应用程序等等。

微软在二层布置了 4 个传统的升降板服务器机房，平均每个机房约 1.2 万平方英尺，功率为 3 兆瓦特。这个数据中心的总面积为 70 万平方英尺(约合 6.502 万平方米)，成为全球最大的数据中心之一。即使只启用半数服务器，这个数据中心的能耗也将达到 30 兆瓦，是普通数据中心的数倍之多。

（7）SGI“ 冰立方 ”数据中心

SGI 的数据中心称为 Ice Cube（“ 冰立方 ”），有 20 英尺和 40 英尺两种规格。2010 年 5 月，SGI 对“ 冰立方 ”进行了改造，除了 SGI 硬件，还可以在集装箱内安装其他公司硬件。

“ 冰立方 ”采用模块化设计，能够容纳 SGI 所有的服务器和存储设备，包括其 Altix UV 向上扩展超级计算机和 Altix ICE 向外扩展超级计算机，允许客户对数据中心硬件进行混搭，从而增加灵活性。SGI 模块化数据中心可以达到超高密度，每个集装箱可以支持 46,080 个处理器核或 29.8PB 数据存储。这些优势使得 ICE Cube 成为建立新数据中心、改造旧数据中心或取代各种规模传统数据中心的理想选择。

从新一代数据中心概念的提出到现今的广泛应用，不过短短数年时间。这种低成本、易运输、容易搭建、周期短、易扩展、高能源效率、不苛求环境的一体化数据中心解决方案带来了一股新的热潮。

2.2 调度概述

调度是一个决策过程，研究的内容是在某个时间点或某段时间内将资源分配给不同的用户任务。调度优化的目标可以是成本、用户任务完成时间、任务优先级、收益等众多因素中的某一个或多个。

2.2.1 调度的基本术语

在这一节，我们介绍调度问题中常用的基本术语和符号。

（1） 任务数量 m：m 表示等待队列中任务数量，是常量。

（2） 机器资源数量 n：n 表示云计算环境中机器资源数量，也可以是空闲机器资源数量，n 也是常量。

（3） 任务完成的时间底线 deadline（或简写 d）：表示任务必须在该时间点之前完成；若服务方未能按时完成则减少收益，付出相应赔偿。

（4） 用户支付的费用 reward：任务在 deadline 之前完成时，用户向服务方支付的费用。

（5） 赔偿率 decay：服务方未能在 deadline 之前完成任务时，每超过一个单位

时间，服务方需支付的赔偿金。

（6） 权重 w：w 表示用户任务优先级的高低。w 的值越大表示任务优先级越高；相反，w 的值越小表示任务优先级越低。此外，w 也可以表示不同调度影响因素之间的权重比例。

2.2.2 云计算环境中的调度新特征

2.2.2.1 资源环境

云计算环境下，服务方对资源的配置差异较大，可能是高性能集群，可能是性能较高的服务器，也可能是普通 PC 机，以及各种硬件资源上的虚拟机环境。因此，云计算资源具有规模大、异构性强、可用性高等特征。同时，由于机器类型多种多样，资源又可能从属于不同的云服务提供商，资源的计算能力、存储能力、带宽等因素也具有较大差异，因此云计算环境下的调度问题就变得异常复杂。

2.2.2.2 约束条件

在传统的分布式计算环境中，系统整体性能最优往往是调度的最优化目标，资源均为无偿免费的。云计算环境中，云服务提供商提供资源提供服务，用户“ 按需付费 ”，只需要为使用的资源或服务付费。因此，云环境中的调度问题必须考虑任务执行的成本约束。此外，与成本相关的，任务完成时间底线（deadline）、赔偿率以及用户付费等因素，也是云计算调度问题中需要考虑的重要约束条件。

2.2.2.3 优化目标

传统分布式环境中，调度的优化目标均是以系统为中心，主要面向系统性能，如系统吞吐量、CPU 利用率等，而对用户的服务质量（QoS）需求考虑较少。云计算环境中不仅注重资源利用率及系统性能的提高，而且重视保证用户的 QoS 需求，以实现资源供给与资源消费的双赢局面。

在云计算环境中，由于用户可以依据自身需求（拥有的资金，对任务完成时间底线的要求等等），自主选择云中的资源。因此，在后文中，我们更注重从云服务提供方的角度优化任务调度目标。

2.2.3 云计算环境中的调度机制

目前，云计算环境中调度机制的相关研究不多，但是资源管理、任务调度和负载均衡问题仍然是云计算厂商关注的关键问题。各个云计算厂商均依据自己的基础设施架构发展云计算应用，因此云计算环境中的调度和资源管理模式呈现多样化，还未形成统一的标准和规范约束。

（1） IBM 的 Tivoli（G. Karjoth，2003）

IBM 的蓝云架构采用 Tivoli 系列产品来完成云计算的资源监测、分配、作业调度和系统负载均衡控制。Tivoli Provisioning Manager 使用 Websphere Application Server 呈现供应状态和数据中心资源的可用性、调度资源的供应及取消供应，并且可预订资源。由 IBM Tivoli Monitoring Server 监控 Tivoli Provisioning Manager 提供的服务器运行状况（CPU、磁盘和内存等）。这种方式体现了目前云计算的通用管理模式，即虚拟化技术的支持，简化了任务调度过程。

（2） Map Reduce 调度机制（Jeffrey Dean.，2008）

很多 IT 厂商提出的云环境中的编程模式都是基于 Map Reduce 的思想，它不仅是编程模型，同时也是一种高效的任务调度模型。在 Map Reduce 编程模式下，并发处理细节、容错细节、数据分布细节、负载均衡等细节被抽象到函数库 Lib 中，通过 Map Reduce 接口，用户可以把大尺度的计算自动的并发和分布执行。

Map Reduce 作为一种流行的云计算编程模型，在云计算系统中应用广阔，但是基于它的开发工具 Hadoop 并不完善，特别是其调度算法过于简单，判断需要进行推测执行的任务的算法造成了过多任务需要推测执行，降低了整个系统的性能。

云计算环境中资源数量庞大，并且是动态变化的，如何尽可能将所有资源（计算资源，存储资源等）集中起来，采用何种任务调度策略，对这些资源进行组织和调度，实现资源的自动调节和负载均衡，资源分配的灵活性和按需分配，对于充分利用资源，发挥云计算的特点具有重要意义。

2.3 本章小结

数据中心是云计算的重要组成部分，承担着云计算中存储、运行、容灾备份等重要任务。调度问题关系到资源合理分配及任务高效完成，一直是分布式计算、网格计算、云计算环境中的热点研究问题。本章首先介绍了数据中心的演变、分类、分级情况，阐述了新一代数据中心现状；接着介绍了调度问题中的基本术语，以及云计算环境中的调度新特征和调度机制研究现状。论文的后续部分将围绕云计算数据中心和调度机制展开研究。

第3章　雪花结构的新型云计算数据中心结构

网络结构是设计云计算数据中心必须考虑的重要因素，它为云计算数据中心的高扩展性和资源的高利用率提供充分保障。本章针对云计算数据中心结构上的可扩展性及绿色节能需求，充分考虑新型数据中心应具备的新特点，借鉴已有知名数据中心结构，依据科赫曲线，提出一种新型数据中心网络结构——雪花结构。该结构充分考虑了数据中心的可扩展性，在保证交换机与服务器较低数量比例（0.125-0.333 范围内）的前提下，可以在较短的平均路径内实现节点间路由机制，具有较小的网络开销。本章对雪花结构的云计算数据中心网络构建方法及其属性进行讨论，设计该结构中节点间的路由协议和算法，并进行实验模拟，以验证新型结构的性能。

3.1 引言

过去两年里随着碳排放和电费的不断激增，使得高能效数据中心的研究已经发展成为热点。数据中心是提高能源效率最困难的商业建筑，因为安装在数据中心的计算机需要大量的电力并且排放出大量的热量，而数据中心消耗的一半电力都用于支持 IT 设备的电力和制冷空调等冷却基础设施。

美国联邦环境保护署（Environmental Protection Agency，EPA）于 2009 年 8 月提交了一份关于数据中心的报告。这份报告称，数据中心的能源消耗从 2000 年至 2006 年增长了一倍，预计到 2011 年再增长一倍。此外，全球权威机构 Gartner 调查也显示，IT 行业每年的二氧化碳排放量约为 3500 万吨，占全球总排放量的 2%，数据中心成碳排放大户。企业每年在用电成本上的花费已经大于当年硬件设备投资额。面对如此严峻的形势，节能已经成为建设绿色数据中心的重点。从机房用电分配看，服务器设备占电能总能耗的 52%，而制冷系统和电源系统各占 38%和 9%，照明系统仅占 1%。因此从服务器的角度出发，在保证数据中心海量服务器的前提下尽可能降低交换机个数，提升服务器与交换机的数量比例，不仅可以有效降低能耗达到减排目的，同时还可以降低交换机的成本开销。

3.2 几种典型的数据中心网络结构

目前有代表性的数据中心网络结构有 Fat-Tree（Charles E.，1985），DCell（Chuanxiong Guo，2008），BCube（Chuangxiong Guo，2009）和 VL2（Albert G.，2008）等。我们将详细介绍这几种结构并分析其各自的特点。

Fat-Tree 结构将服务器分为 k 个子群，每个子群包含两层端口数为 k 的 k/2

个交换机，下层每个交换机的 k/2 个端口连接到 k/2 台主机，其余 k/2 个端口分别与每个聚和层交换机连接；核心层需要(k/2)2 个端口数为 k 的核心交换机，最多能支持 k3/4 台主机（如图 3.1 所示）。Fat-Tree 抛弃了传统数据中心采用专用交换机的模式，转而采用商业以太网交换机，较大提高了性价比。它能为包含上万台服务器的数据中心提供高聚合带宽，不需要对主机网络接口、操作系统进行修改便可构建，且与以太网、TCP/IP 等通信协议兼容良好。Fat-Tree 各层的链路数相等，使得所有服务器产生的最大流量和核心层的最大吞吐量相等，不存在网络瓶颈，而且它采用两张路由表进行两级路由，并采用一定的链路错误检测机制来实现容错路由。

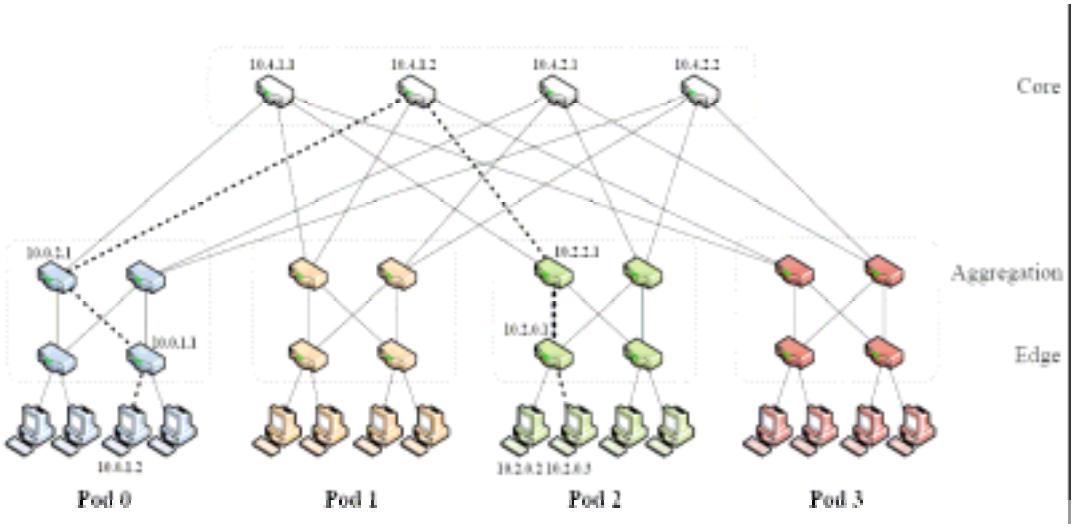


图 3.1 Fat-Tree 结构图

Fat-Tree 结构解除了树形结构上层链路对吞吐量的限制，并能为内部节点间通信提供多条并行链路。但是 Fat-Tree 的扩展性受限于核心交换机端口数量，目前比较常用的是 48 端口 10G 核心交换机，在三层树结构中能够支持 27,648 台主机。长远来讲，规模在十万以内的数据中心是无法满足应用需求的，因此 Fat-Tree 存在扩展性不足的缺点。Fat-Tree 的另一个缺点是容错性差，具体表现为处理交换机故障能力不足及路由协议容错性不强。MSRA 研究表明（A. Greenberg，2009）：Fat-Tree 对低层交换机故障非常敏感，严重影响系统性能。因为 Fat-Tree 仍然是树结构，本质上具有树结构的缺陷。

DCell 是一种递归定义的网络结构，使用位于第 i-1 层的 DCell 构建第 i 层的 DCell。当节点度增加时 DCell 的规模接近以 2 的指数次方扩展。通常 DCell_0 内包含常数服务器，一般为 3-8 台，并通过微型交换机互连（如图 3.2 所示）。DCell 容错性较好，没有单点故障并且能够在严重的链路和节点故障的情况下利用其分布式协议实现接近最短路径的路由。DCell 还能为各种各样的服务提供比传统树结构更高的网络容量。另外 DCell 可以增量扩展并且在不完全结构的情况

下表现出上述性能。虽然 DCell_0 很小，但 DCell 能支持的服务器数量是惊人的，例如当 DCell_0 包含 6 台主机时，DCell_3 可以支持 326 万台服务器。

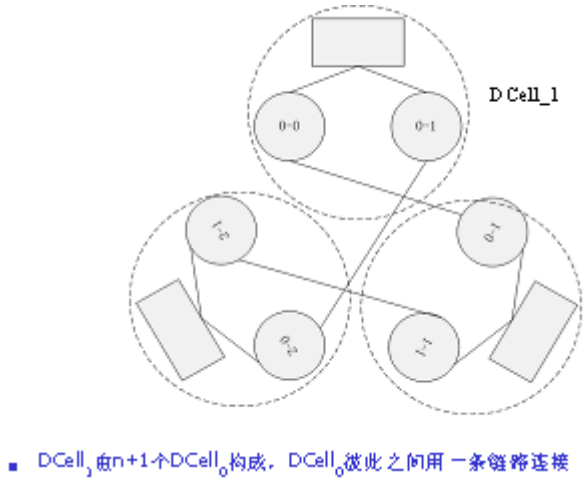


图 3.2 DCell_1 结构图

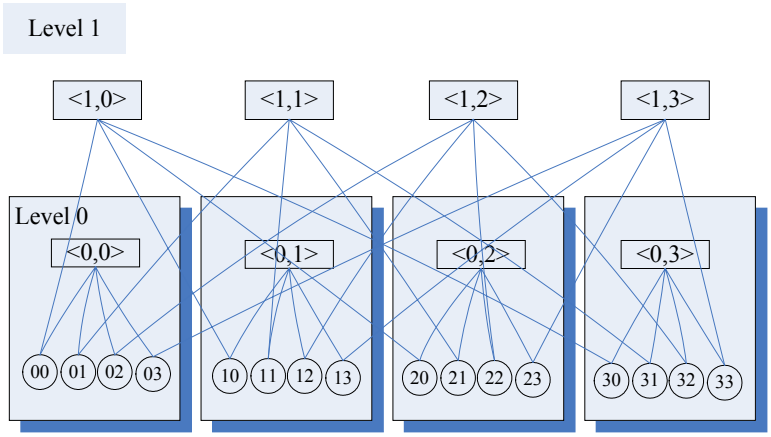


图 3.3 BCube_1 结构图

由于 DCell 连接方式接近完全图，并且 DCell 路由协议(DCell Fault-tolerant Routing，DFR)利用链路状态和贪心算法来实现容错路由，所以 DCell 可以在服务器、链路或交换机严重故障的情况下，实现性能较好的路由。然而，DCell 也有不足之处。首先，完全图的连接方式可能带来巨大花费，而且实际链路规模庞大，连接和维护困难。其次，DCell 中流量在不同层次分布不均匀，level0 承担了过多流量，严重影响吞吐量。最后，由于 DCell 使用服务器执行路由，增大了网络延迟，而且路由协议也不适于在链路故障时发现最短路径，网络延迟较大。

BCube 是 DCell 的模块化版本（J. Naous，2008；J. Hamilton，2009；J. Lockwood，2007），它的连接方式是在 BCube_0 以外的层次中采用微型交换机实现连接（如图 3.3 所示）。它在每层单元的数量上与 DCell 不同，如果 DCell 在第 k 层有 n 个 DCell_k-1，那么第 k+1 层则有 n+1 个 DCell_k；而 BCube 在任一级

都具有相同的单元数 n，易得出 BCube_k 拥有 nk+1 台服务器(n 为 BCube_0 中的服务器数)。交换机作为连接媒介使 BCube 具有很多冗余路径，这可以保证容错路由并方便模块化连接而且路由速度比 DCell 快。另外 BCube 应用 BSR(BCube Source Routing)选取路径，采用路径自适应协议，能够很好的开拓网络中的最短并行路径，并实现可靠数据传输。

BCube 能够高效无带宽限制的执行 One to One、One to All、One to Several、All to All 类型的通信，很好的支持 GFS，MapReduce 类应用。BCube 的连接方式和递归结构使得数据中心可以模块化建设，实现性好。在结构不完整的情况下 BCube 表现出比 DCell 具有更好的性能。实验表明（A. Greenberg，2009）给定 2048 台服务器，在 Fat-Tree、DCell 和 BCube 结构都不完整的情况下，BCube 的网络吞吐量和容错性能是最好的，这是因为 BCube 对不完整结构采用完整结构的交换机连接策略。BCube 的不足体现在可扩展性上，BCube 在 k=3，n=8 时(k 为 BCube 层次, n 为 BCube_0 中服务器数)，仅支持 4096 台服务器，与 DCell 差距较大。

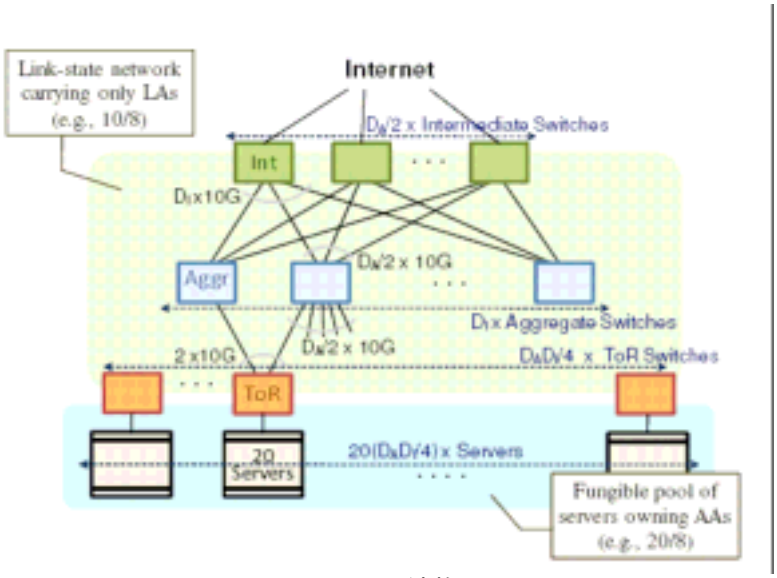


图 3.4 VL2 结构图

如图 3.4 所示，VL2 是一种可扩展的灵活的数据中心网络结构，它能够支持超大规模的数据中心，为服务器间提供均衡的高带宽通信性能，服务间的性能隔离及以太网第二层语义。第二层语义是指将第二层所有的域虚拟化为统一的域，在这个层面上所有主机都位于同一个域中。VL2 在结构上改变的是第三层交换机的连接方式，采用特殊协议实现虚拟第二层；而其它结构在物理连接方面的改变是整体。另外，地址表示和路由协议在 VL2 中更为重要，直接关系到虚拟第二层的实现。VL2 中采用 VLB(Valiant Load Balancing)进行路由，为均衡各路径流量，VL2 将各中间交换机设为相同 IP，采用随机的方式选择一个中间交换机实现路由。

3.3 雪花结构——新型云计算数据中心网络结构

本节提出数据中心雪花型网络结构的构建方法，以及其特有的一些属性。之所以称为雪花结构，是因为这种结构的构建依据科赫曲线，形似科赫雪花。

3.3.1 雪花结构及其构建方法

雪花结构包含两个组成部分：服务器和微型交换机。我们在 3.2 部分已经介绍了 DCell 和 BCube 结构，二者均采用了递归的方法来定义各自结构。在雪花结构中，我们同样采用递归定义的方法，在（n-1）级雪花结构的基础上添加若干个 0 级雪花结构，构成 n 级雪花结构。这里的“若干个”将在后面详细介绍。这样定义的好处在于，首先，用添加某种固定结构的方式尽可能的将结构模块化处理，有利于结构的模块化连接；其次，当 n 级结构没有扩展完全时，继续扩展（n+1）级结构也较容易。此时，若发现 n 级结构没有扩展完全，可以在不改变已有（n+1）级结构的情况下，继续补充不完全的 n 级结构，有利于结构的扩展。雪花结构的构建方法详细描述如下。

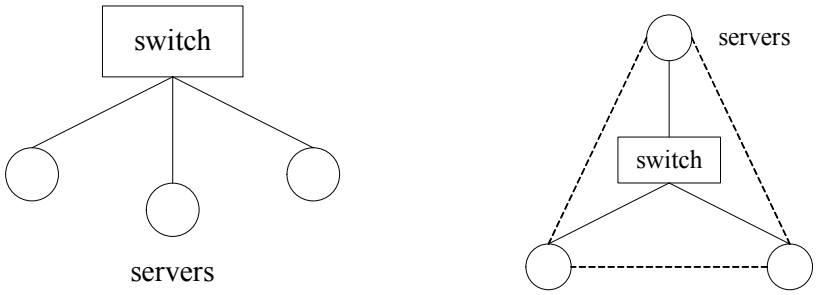


图 3.5 k=3 时的 Snow_0 结构

0 级雪花结构（Snow_0）由一个微型交换机和若干个服务器组成。借鉴已有数据中心结构，设置服务器的个数最少为 3 个，不超过 8 个（Chuanxiong Guo，2008；Chuanxiong Guo，2009）。如图 3.5（a）所示，我们以 3 个服务器为例（k=3），将 3 个服务器连接到一个 n 端口的微型交换机上。如图 3.5（b）所示，调整 3 个服务器的理论位置，为 3 个服务器的两两直接互联添加了三条虚线。这三条虚线并非实际的服务器互联，而是为了方便说明构造下一级雪花结构，我们称之为虚连接，即不存在的网络连接，只是为了方便说明结构的构建。

在 Snow_0 基础上，断开三条虚连接（如图 3.5（b）），每断开一条虚连接，即要添加一个 0 级雪花结构，将该新添 0 级结构中的微型交换机分别与虚连接的两端节点相连。相对于虚连接，这样的连接称为实连接，即断开虚连接重新构建的连接。这里的实连接是实际存在的连接，是新添加结构中的交换机与原有结构

中服务器的连接。这样得到一个 1 级（Snow_1）雪花结构，如图 3.6 所示。

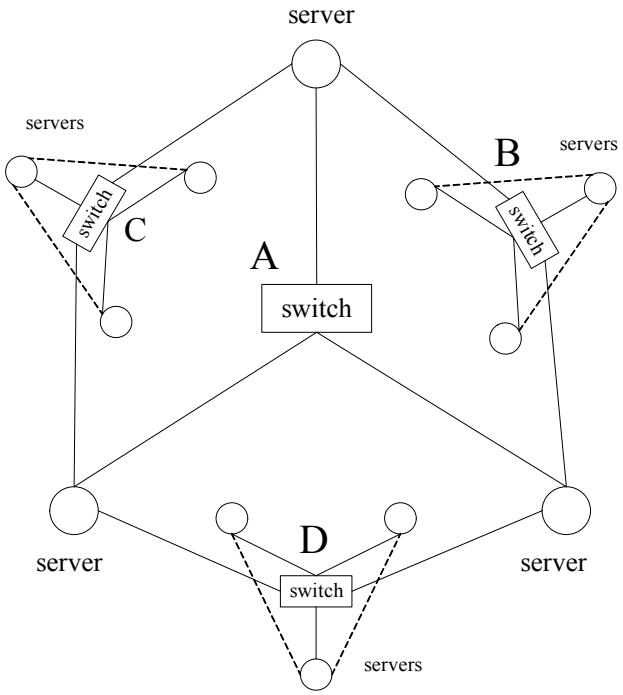


图 3.6 k=3 时的 Snow_1 结构

由于是在断开的虚连接处添加 0 级雪花，此时虚连接的状态已经变为实连接（1 条虚连接转化为 2 条实连接），链接状态从无到有，因此在此处不再添加虚连接。需要特别说明的是，实连接不只是实际存在的连接，更体现了由虚连接到实连接，这种连接从无到有的变化状态，因此在 Snow_0 中服务器与交换机相连的三处不看作是实连接，它虽然是实际存在的连接，但是并没有状态变化的体现。仔细观察可以发现，断开虚连接添加的 0 级结构与 Snow_0 是有区别的，它缺少了一条虚连接，并不是真正意义上的 Snow_0 结构。为了区分二者，这种后来断开虚连接、不断添加的、缺少一条虚连接的 0 级结构，我们称其为 Cell，以区分 Snow_0。

Snow_1 包含 6 条实连接和 6 条虚连接。在之后的每一级雪花结构中，总是断开上一级中包含的所有实连接和虚连接，添加 Cell，构成新的高级结构。图 3.7 显示了 k 为 3 时的 2 级（Snow_2）雪花结构。

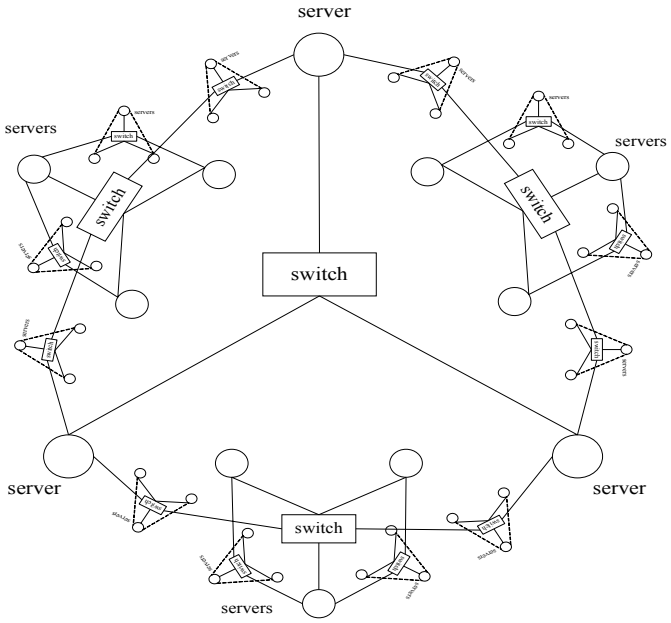


图 3.7 k=3 时的 Snow_2 结构

构建雪花结构的伪代码如算法 3.1 所示。

算法 3.1：*构建雪花结构的算法*

```
1)  Construct Snow(k,n)   //k 是 Snow_0 中包含的服务器个数，构建 Snow_n
2)  If （n==0）            //构建的是 Snow_0
3)     For(int i=0;i<k;i++)
4)        Connect ith server to switch;
5)        标记两两相邻的服务器；          //虚连接标记
6)        将标记的服务器成对放入队列；  //记录虚连接，将被虚连接的节点对加入队列
7)        记录队列长度 queue_len；        //队列长度即下一级要添加 Cell 的个数
8)     Return;
9)  Else while(n!=0)
10)       {while(queue_len!=0)
11)          {从队列中取出节点对，添加 Cell，Cell 中的交换机分别连接原节点对中的两个
节点，形成实连接；
12)           将添加 Cell 产生的 2 条新的虚连接和 2 条新的实连接节点对关系加入队列；
13)           记录新的队列长 queue_new_len；
14)           queue_len--；        //添加一次 Cell 队列长减一
15)              }
16)          n--;                  //当队列长为 0 时，表明新的一级雪花结构构建完成
17)          queue_len=queue_new_len;
18)    }
```

3.3.2 雪花结构的属性

在这一小节，我们以定理的形式给出并分析雪花结构的特有属性。

定理 3.1　假设第 0 级雪花结构中包含 k 个服务器(Cell 也包含 k 个服务器)，则第 n 级雪花结构包含 SV_n 个服务器，SW_n 个微型交换机。其中 $k \in \{3,4,5,6,7,8\}$，$n \geq 0$。SV_n 和 SW_n 分别满足下列条件：

$$SV\_n = SW\_n * k \tag{3.1}$$

$$SV\_n = k * (k+1)^n \tag{3.2}$$

证明：Snow_0 包含 k 个服务器，因此 Snow_0 中，SV_n：SW_n=k：1。

Cell 包含 k 个服务器，因此 Cell 中，SV_n：SW_n=k：1。

每一级雪花结构都是在前一级的基础上不断添加 Cell，服务器与交换机的比例没有改变，因此 SV_n：SW_n=k：1 是显而易见的。变换形式即得到（3.1）式结论。

下面，我们用数学归纳法来证明（3.2）式。

当 n=0 时，SV_0=k，$k*(k+1)^n$=k，此时，$SV\_n = k*(k+1)^n$。

当 n=1 时，SV_1=k+k*k=k(k+1)，$k*(k+1)^n$=k(k+1)，$SV\_n = k*(k+1)^n$。

假设，当 n=m 时，等式成立，即 $SV\_m = k*(k+1)^m$。

则当 n=m+1 时，SV_(m+1)=SV_m+k*断开 Snow_m 中虚连接和实连接添加的 Cell 个数。

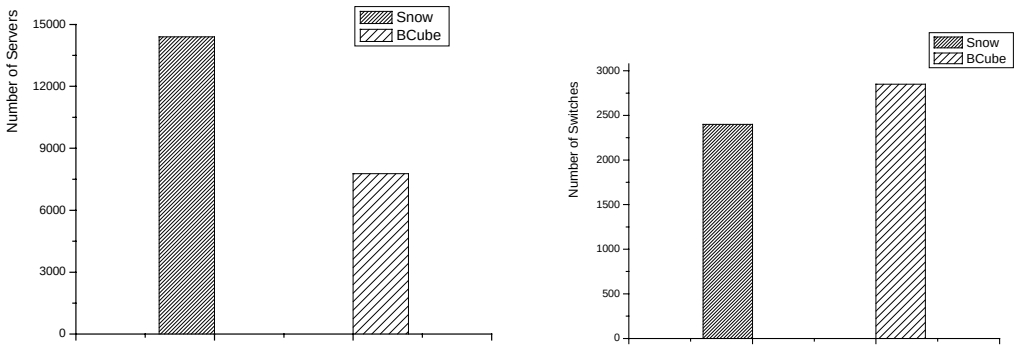
这里，断开 Snow_m 中虚连接和实连接添加的 Cell 个数=Snow_m 中虚连接和实连接个数；

Snow_m 中虚连接和实连接个数=(k+1)*[SV_m-SV_(m-1)]/k=$k*(k+1)^m$；

因此 SV_(m+1)=$k*(k+1)^{m+1}$。

所以，$SV\_n = k*(k+1)^n$，等式（3.2）成立。　　证毕。

从公式（3.1）和（3.2）可以得到，当 k=3，n=10 时，雪花结构可以达到 314 万个服务器。该结构在最初阶段扩展较慢，但是一旦形成即迅速增长。当 k=6，n=4 时，雪花结构可以达到 14406 个服务器，2401 个交换机，交换机与服务器的比例为 0.167。在 BCube 中，当 k=6，n=4 时，只能达到 7776 个服务器，却同时包含交换机 2850 个（如图 3.8(a)和(b)）。不仅服务器数量减半，扩展性较弱，而且交换机与服务器的比例为达到了 0.367，是雪花结构的 2 倍多。当 k=4，n=4 时，BCube 的交换机与服务器比例甚至达到 2.328，交换机的数量是服务器的 2 倍多。考虑到数据中心这种大规模结构的制冷开销以及交换机成本，这种交换机个数的较高比例甚至超过服务器同比例增长显然是不利的。此外，雪花结构可以保证交换机的低数量也是很有优势的，当 k=6 时，始终保持交换机与服务器的比例为 0.167，最坏情况 k=3 时，也可以达到 0.333 的低比例。



(a) 服务器总数的比较图　　(b) 交换机总数的比较图

图 3.8 k=6 n=4 时雪花结构与 BCube 的比较图

3.3.3 雪花结构的性能分析

在这一小节，我们从网络交换容量、瓶颈链路、延时、绿色节能四个方面来分析雪花结构的性能。

- 网络交换容量

我们假设交换机与交换机之间的带宽为 A Mb/s，交换机与服务器之间的带宽为 B Mb/s，一般情况下 A≥B。

当任意两个服务器之间需要发送数据时，这两个服务器之间的带宽受限于两者之间路径上的最低带宽值。

雪花结构在保证较好扩展性的前提下，同时维持较低的交换机数量。此外，在雪花结构中，服务器基本都是与交换机相连，几乎不存在服务器与服务器直接连接的情况，这样可以保证较高的数据传输速率。因此，雪花结构适用于数据中心内节点之间需要频繁通信的情况。

- 瓶颈链路

我们将瓶颈分为两种：节点瓶颈与路径瓶颈。

节点瓶颈是指，某个节点（服务器或者交换机）由于数据流量过大成为瓶颈，越是层级低的节点越容易成为瓶颈节点。因此我们在 3.5 节补充问题研究里将低层级中的一些服务器改进为交换机，其中的交换机也可以换成高吞吐量的，如万兆交换机，以减轻数据传输的压力。

路径瓶颈是指，两个子网之间有且只有一条路径，当该路径断开时，这两个子网即失去连接而形成瓶颈；这样的路径我们称为瓶颈路径。在定理 3.5 中我们将证明，Snow_n 中任意两个服务器包含并行路径，至少 2 条，不超过 2^{2n} 条。因此理论上雪花结构中不存在路径瓶颈的问题。

- 延时

RTT(Round-Trip Time)表示从发送端发送数据开始，到发送端收到来自接收

端的确认（接收端收到数据后便立即发送确认），总共经历的时延。当两点之间的 RTT 值较大时说明这两点之间的延时较大；反之，则说明两点之间延时较小。从 RTT 的定义可以看出，RTT 与两点之间的最短路径成正比。当两点之间距离较长时，RTT 较大；反之，则 RTT 较小。我们在定理 3.4 及推论 3.1 中可以得出最长最短路径为（2n+1）跳。因此，延时为 O（n）。

- 绿色节能

我们在 3.3.2 部分已经详细说明了雪花结构与 BCube 结构中服务器与交换机数值上的比较情况。如图 3.8 所示，当 k=6，n=4 时，雪花结构中交换机与服务器的比例为 0.167，BCube 中交换机与服务器的比例为 0.367。平均每台服务器占用的交换机个数减少了 54.5%，交换机产生的能耗减少了 54.5%。当 k=4，n=4 时，BCube 的交换机与服务器比例甚至达到 2.328，雪花结构仅为 0.25，平均每台服务器占用交换机个数减少了 89.3%，交换机产生的能耗减少了 89.3%。显然，雪花结构不仅实现了高扩展性，而且达到绿色节能、节约成本的目的。

3.4 雪花结构中的协议和路由策略

前面已经说到，当 k=3，n=10 时，雪花结构可以达到 314 万个服务器。由于该结构中的服务器个数随着 n 值的增加呈指数次方不断增长，它的目标是扩展连接十万百万量级的服务器。因此，基于雪花结构的数据中心不适合使用全局的链路路由机制，容易造成带宽瓶颈和单点失效的最短路径优先协议 OSPF（J. Moy，1998）自然不适合这种结构。

在这一部分，我们首先介绍无失效情况下节点路由情况；接着提出基于雪花结构的路由协议；最后，详细阐述依据路由协议的节点路由情况。

3.4.1 无失效情况节点路由

在这一小节，首先分析雪花结构中的无失效路由情况。假设源节点为 src，目的节点为 des。

定理 3.2　Snow_0 中任意两个服务器之间仅 1 跳。

证明：如图 3.5（a）和（b）所示，Snow_0 中任意两个服务器间通过交换机即可达，1 跳显然。　　证毕。

定理 3.3　k=3 时，Snow_1 中任意两个服务器之间不超过 2 跳。当 $k \in \{4,5,6,7,8\}$ 时，任意两个服务器之间不超过 3 跳。

证明：k=3 时，如图 3.6 所示，Snow_1 中节点路由可以分为以下五种情况：

Snow_1 中服务器—> Snow_0 中服务器：1 跳；

Snow_0 中服务器 —> Snow_1 中服务器：1 跳；

Snow_1 中服务器 —> Snow_0 中服务器 —> Snow_1 中服务器：2 跳；

Snow_1 中服务器 —> Snow_0 中服务器 —> Snow_0 中服务器：2 跳；Snow_0 中服务器 —> Snow_0 中服务器 —> Snow_1 中服务器：2 跳。不超过 2 跳的结论是很容易得到的。

同理，当 $k \in \{4,5,6,7,8\}$ 时，可得到 3 跳结论。　　证毕。

定理 3.4　Snow_n 中任意两个服务器之间最长最短路径为（2n+1）跳，其中最短路径是指两个服务器之间路由经过的最少服务器个数。

证明：假设源节点 src 位于 Snow_j 中，目的节点 des 位于 Snow_k 中。j 和 k 都小于等于 n。

此时，考虑最长最短路径情况。源节点 src 从 Snow_j 逐级向下经由 Snow_（j-1），Snow_（j-2）...路由到 Snow_0 中的服务器，最多需要 j 跳。Snow_0 中的服务器经由 Snow_1，Snow_2…路由到 Snow_k 中的服务器最多需要 k 跳。而 Snow_0 中的服务器彼此之间最多 1 跳。因此 Snow_j 中的源节点 src 路由到 Snow_k 中的目的节点 des 最长最短路径为（j+k+1）跳。当 j 和 k 都取最大值 n 时，src 与 des 的最长最短路径达到最大值（2n+1）跳。　　证毕。

由上述分析我们还可以得到，当 j 和 k 不同时取到 n 或者 j 和 k 不相等时，src 与 des 的最长最短路径必然小于等于（2n+1）跳。因此，我们得到推论 3.1。

推论 3.1　Snow_j 与 Snow_k（j 不等于 k，且 j 与 k 均小于等于 n）中任意 2 个服务器之间最长最短路径上限为（2n+1）跳。

定理 3.5　Snow_n 中，任意两个服务器包含并行路径，至少 2 条，不超过 2^{2n} 条。并行路径是指，两个服务器之间同时存在的路径，这些路径彼此之间独立无依赖关系。

证明：我们通过在前一级雪花结构的基础上断开所有的实连接和虚连接，添加 Cell 来构建下一级雪花结构，每一个添加的 Cell 都是通过交换机的左右两条连接才加入新结构中，因此，任意 2 个服务器之间至少包含 2 条并行路径。

当 Snow_j 中源节点 src 路由到 Snow_k 中的目的节点 des 时，src 从 Snow_j 开始经由 Snow_（j-1）Snow_（j-2）…路由到 Snow_0，每一次从 Snow_p 到 Snow_q 总是有 2 条并行路径，因此至多包含 2^j 条并行路径。同理，Snow_0 到 Snow_k 也至多包含 2^k 条路径。因此，从 Snow_j 的源节点 src 路由到 Snow_k 的目的节点 des 至多有 2^{j+k} 条并行路径。当 src 与 des 都处于 Snow_n 时，两节点至多包含 2^{2n} 条并行路径。　　证毕。

3.4.2 基于雪花结构的协议

(1) 节点唯一标识方法

我们对每一个服务器和交换机进行标记，标记结构为<级别，度数>。级别表明该服务器处于雪花结构的这个级别中，是在该级别被添加进雪花结构的。然而

由于在每个级别中并非只添加了一个节点，仅仅级别这一度量还不足以唯一标识雪花结构中的节点位置。因此，我们引入度数这个度量，结合级别来唯一标识雪花结构中的每一个服务器和交换机。下面详细解释节点标识的方法。

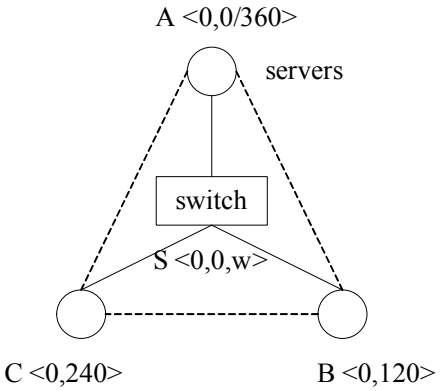


图 3.9 Snow_0 中的节点标识

如图 3.9 所示，标记 Snow_0 中的三个服务器分别为 A<0,0/360>，B<0,120>，C<0，240>。其中第 1 位表示级别，即这三个服务器均处于 Snow_0 中。第 2 位表示度数，这里的度数表明的是相对位置，均是以 Snow_0 中的交换机为基础参照物得到的相对位置。因此 Snow_0 中的交换机被标记为<0,0>，服务器 A 在交换机的正上方，相对度数为 0 度。依照顺时针方向，服务器 B 为 120 度，服务器 C 为 240 度，这样 Snow_0 中的三个服务器即被唯一标识。为了区分服务器与交换机，对交换机添加 w 标记，即 Snow_0 中的交换机标记为 S<0,0,w>。需要说明的是，对于特殊位置的服务器 A，同时标记其度数为 360 度，方便构造高级雪花结构的标识。

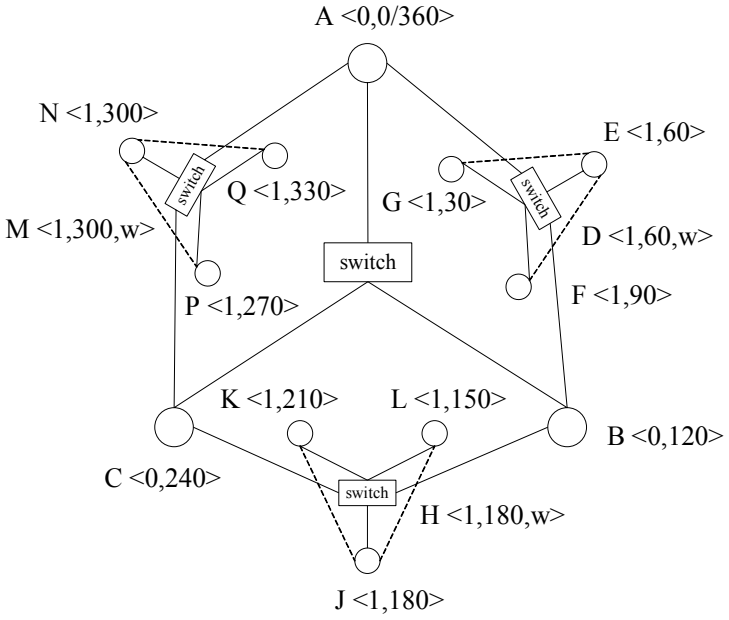


图 3.10 Snow_1 中的节点标识

我们在 3.1 节已经阐述了如何构建 Snow_1。这里，同样利用产生 Snow_1 的三个 Snow_0 服务器来产生 Snow_1 的标识。如图 3.10 所示，Snow_1 中的交换机 D 所处 Cell 添加在服务器 A,B 之间并与其直接相连，利用 A,B 来产生 D 的标识，记 D 的标识为<1,60,w>，即 Snow_0 构建产生 Snow_1，级别为 1，度数为 A,B 和的一半。用这一位置同时标记交换机 D 的正上方（相对于 D 的位置）服务器 E 为<1,60>。前面说到服务器 A 的度数同时标记为 360 度，当 A,C 用以标识交换机 M 时，采用 A 的 360 度构造 M 为<1,300,w>。即 A 与 180 度内的节点构造新标识时采用 0 度，与 180-360 度内的节点构造新标识时采用 360 度。这样做的好处是可以形成一个规范化的节点唯一标识方法。

(2) 请求数据包的结构定义

定义源节点 src 发送的请求数据包中，包含 src 的标识信息，数据包 ID 以及目的节点 des 的标识信息。这样一个请求数据包可以通过其自身 ID 与 src 的标识来唯一标记。当转发数据包的路由节点发现已经转发过某一数据包时，即放弃该数据包，不再转发以降低网络流量。当目标节点 des 收到请求数据包后，也可以根据包中携带的 src 的标识信息获得请求节点的相关信息。具体的路由机制阐述如下。

3.4.3 路由策略

我们定义源节点 src<j，a>，目的节点 des<k，b>。当 src 发送请求数据包时，包内携带的信息包含（ID，<j，a>，<k，b>）。

如图 3.11 所示，在该雪花结构中，蓝色虚线表示中间省略的若干级别。红色源节点标记为 src<j,a>，绿色目的节点标记为为 des<k,b>。当 src 要发送数据包给 des 时，首先查看 des 的度数 b 的所属范围。由于 b 属于 0-120 度范围，则通过服务器 A 或者 C 路由到 B，再经由中间若干交换器不断的接近 des，最后从服务器 G 到达 des。根据度数 b 可以确定路由的方向，根据级别 k 可以确定路由的大概路径长度。

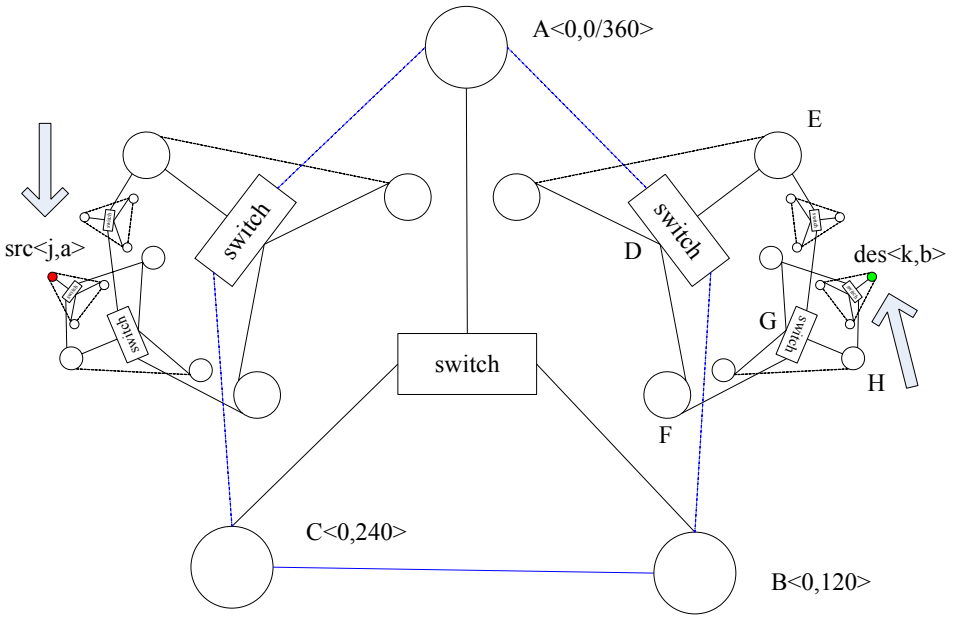


图 3.11　src 路由到 des 的示意图

以图 3.12 为例，src 需要发送请求包给 des 时，首先查看 des 的度数 b，这里为 45 度。src 第一步经由交换机 M<1,300,w>可以路由到 N 或者 C。由于 N 更接近服务器 A<0,0/360>，A 同时为 0 度较 C 更靠近 des，因此 src 选择 MNA 的路径。A 可达 H<2,30,w>，虽然与 des 同层级，但是度数较低，继续路由达交换机 D<1,60,w>。D 可达 K<2,90,w>，虽然 K 与 des 同级但是度数较大。进一步发现 D 连接的两个服务器 E<1,60>和 G<1,30>均属于 Snow_1，E 和 G 可构建 Snow_2 级，且 45 度属于 30-60 度之间，因此，D 经由 E 或者 G 到交换机 F<2,45,w>，最后到达 des。

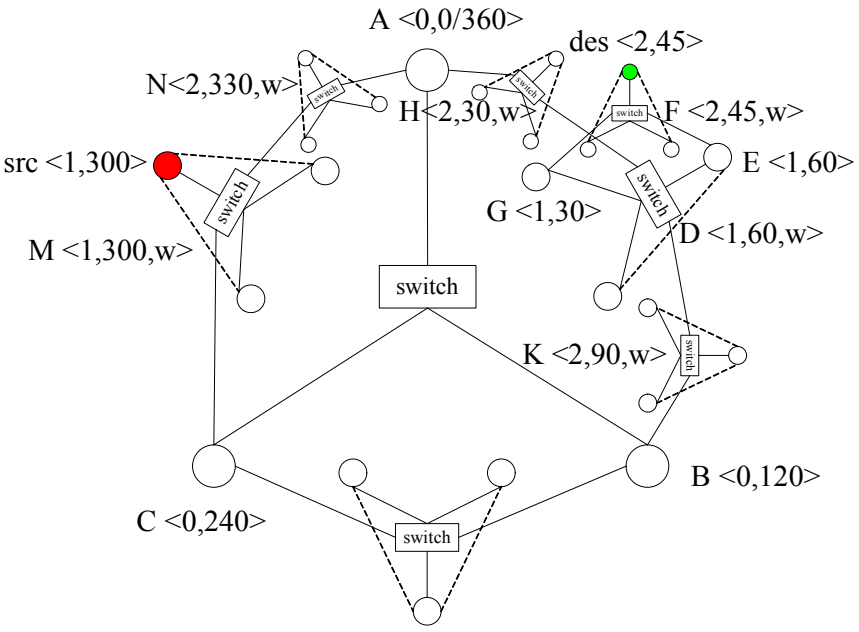


图 3.12　src 路由到 des 的示例

3.5 补充问题研究

当雪花级别不断扩大，结构中的服务器数目不断增多，达到百万级别时，路由的消息数量剧增，此时 Snow_1 中的 12 个服务器显然会担负较重的信息转发任务。因此，我们对上述雪花结构略加改进，将 Snow_1 中的 12 台服务器替换为交换机，如图 3.13 所示，以缓解低级雪花结构的负载强度，提高消息转发速度。

我们在 Snow_（n-1）的基础上添加若干个 Cell 形成 Snow_n。此时，若新添加的 Cell 中的交换机发生故障时，会导致该交换机连接的三台服务器完全脱离雪花结构。当出现这种情况且较为严重时，Snow_n 即退化为 Snow_（n-1）级结构，可用服务器数量显著受到影响，这种影响源于交换机而非服务器本身造成。为了避免这种情况的出现，我们补充规定，在新添加的 Cell 中，三台服务器轮流每隔 20 秒发送一个查询交换机是否存活的消息，这样，每个服务器平均 1 分钟发送一次消息。当三台服务器不能连通时表明交换机出现故障，服务器已经脱离雪花结构，这时需要检查该交换机，确保其稳定工作。

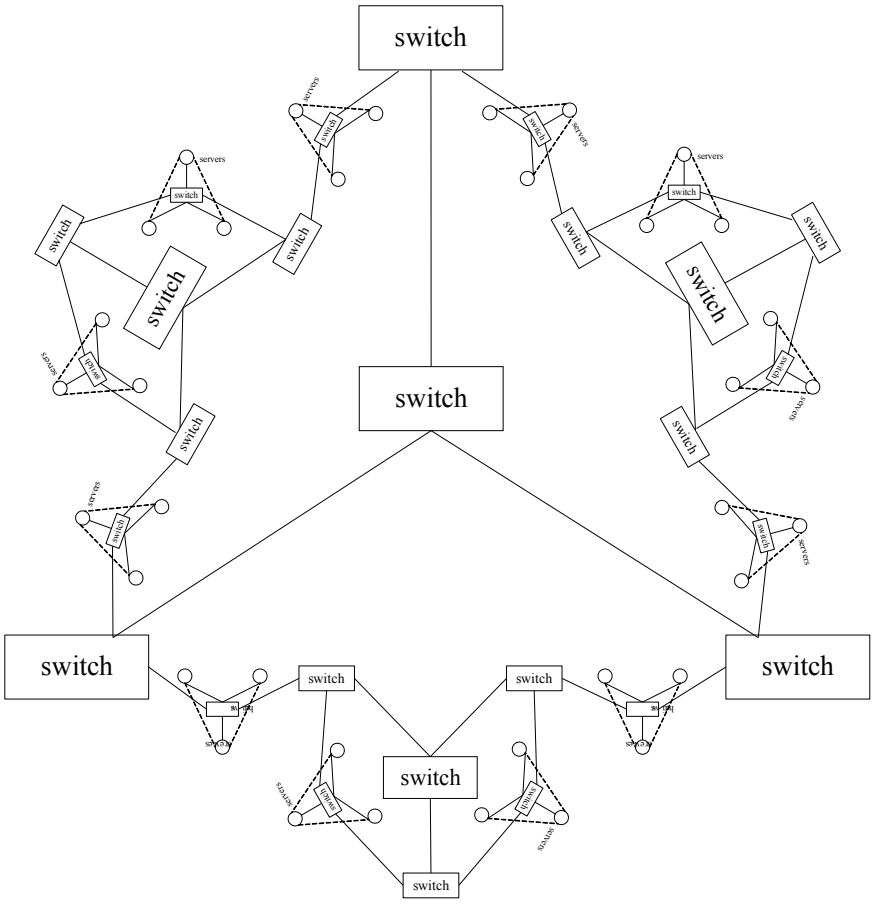


图 3.13 改进 Snow_1 后的雪花结构

3.6 实验模拟及结果分析

在这一部分，我们采用模拟的方法来评估雪花结构的性能。模拟程序用 Java 编写，开发工具为 Eclipse 3.5.0，在一台联想 X200 笔记本上运行。

场景 1：无节点失效时，不同层级节点总数及平均最短路径长度。

表 3.1 无节点失效时各节点到 Snow_0 中三个服务器的平均最短路径情况

雪花级别	Snow_2	Snow_3	Snow_4
服务器总数	48	192	768
交换机总数	16	64	256
可达节点数	47	191	767
不可达节点数	0	0	0
最短路径总和	80	416	2048
平均最短路径长度	1.702	2.178	2.670

首先来看无节点失效情况。表 3.1 显示了当 k=3 时，Snow_0 中的服务器到其余节点的平均最短路径情况。由于未考虑节点失效情况，因此不可达节点数均为 0。由平均最短路径长度可以发现，在 Snow_4 的范围内，平均路径可以保持在 3 跳内，与 3.4.1 中的定理 3.4 相符，但是实验结果优于定理 3.4 的理论推导结果。

场景 2：有节点失效时，平均最短路径长度与路径失效率。这里的节点失效是指服务器故障。

当 k=3 时，我们详细设置了节点失效比例，测试了 Snow_4 中，随着节点失效比例的增加，平均最短路径长度变化情况。从图 3.14 可以看出，当节点失效保持 0.05-0.30 时，平均路径长度波动不大，基本处于 2.6-2.7 之间。

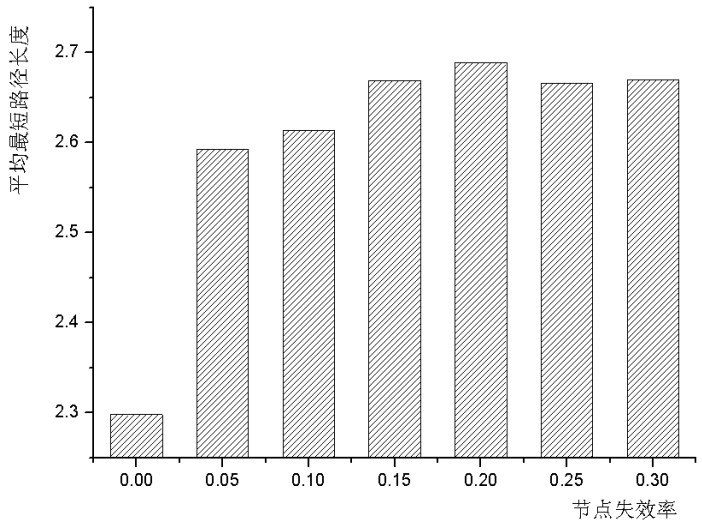


图 3.14 随着节点失效率的增加 Snow_4 平均最短路径长度变化情况

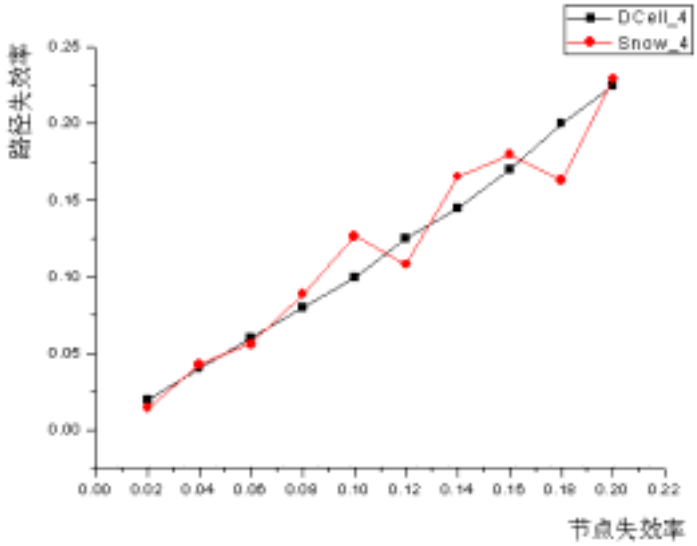


图 3.15 随着节点失效率的增加 Snow_4 与 DCell_4 的路径失效率比较

当 k=3 时，我们测试在了 Snow_4 中随着节点失效率的增加，路径失效率的情况，并与 DCell（Chuanxiong Guo，2008）中的实验结果做了对比。图 3.15 所示为 Snow_4 与 DCell_4 的路径失效对比图。从图中可以看出，Snow 的路径失效率围绕 DCell 上下波动，差别不大。

场景 3：有链路失效时，平均最短路径长度与路径失效率。这里的链路失效是指交换机与服务器之间的链接断开，或者交换机故障，并非服务器本身故障。

从图 3.16 可以看出，随着链路失效率的增加，平均最短路径长度变化幅度不大。链路失效率在 0.2 以内时，平均最短路径长度始终在 2.6-2.9 之间。

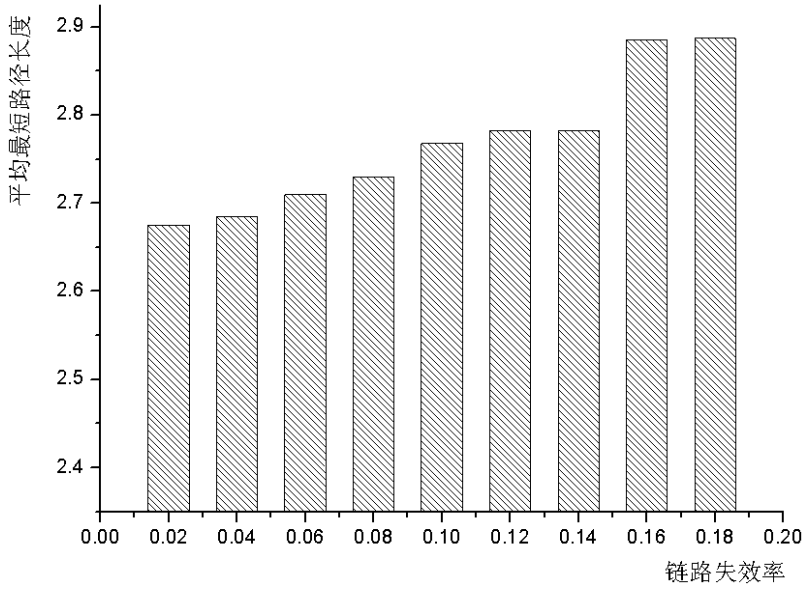


图 3.16 随着链路失效率的增加 Snow_4 与平均最短路径长度变化情况

当 k=3 时，我们测试在了 Snow_4 中随着链路失效率的增加，路径失效率的变化情况，同样与 DCell（Chuanxiong Guo，2008）中的实验结果做了对比。图

3.17 所示为 Snow_4 与 DCell_4 的路径失效对比图。从图中可以看出，Snow 的路径失效率比较稳定，始终保持 0.001-0.004 之间。

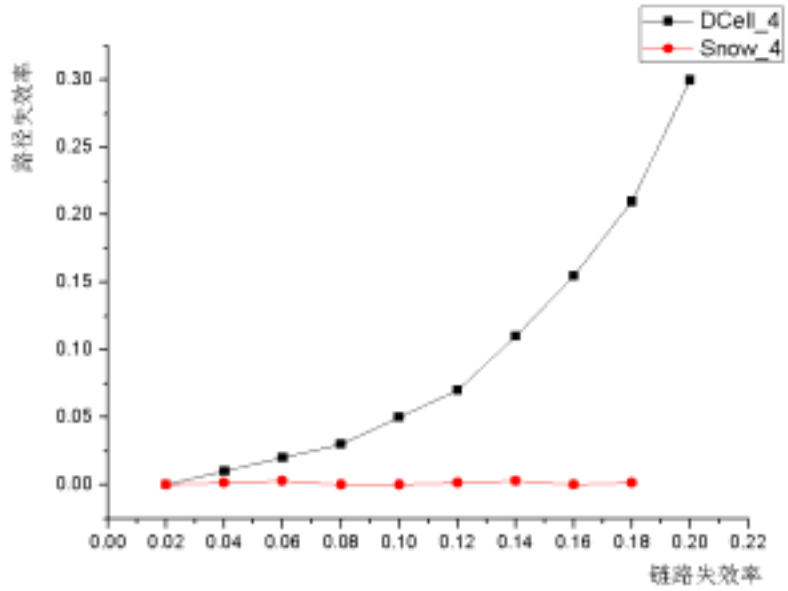


图 3.17 随着链路失效率的增加 Snow_4 与 DCell_4 的路径失效率比较

最后，我们综合比较了 DCell，BCube 以及 Snow，结果如表 3.2 所示。

表 3.2 DCell，BCube 和 Snow 的比较

网络结构	DCell	BCube	Snow
交换机与服务器的比例	高	高	低（优点）
节点失效与路径失效的关系	较好	较好	与 DCell 效果相当
链路失效与路径失效的关系	较好	较好	较 DCell 失效率更低更稳定
网络吞吐量	网络扩展性较好，有较高的网络吞吐量	与 DCell 相比较差	低级别的服务器和交换机可能会成为系统瓶颈（缺点）

3.7 本章小结

为了满足云计算数据中心结构上的可扩展性及绿色节能需求，本章借鉴已有数据中心结构，依据著名科赫曲线，提出了新型数据中心网络结构——雪花结构。该结构充分考虑了数据中心的可扩展性，在保证交换机与服务器较低数量比例（0.125-0.333 范围内）的前提下，可以在较短的平均路径内实现节点间路由机制，具有较小的网络开销。

第4章　基于副本的调度策略研究

副本问题一直是云计算中的热点问题，它弥补了存储对象单点失效、容错性差、接入性能不高等问题。但是副本机制也带来一致性、负载均衡以及由副本产生的各种硬件和通信上的代价问题等等。本章针对云计算中的副本及基于副本的调度问题，提出基于副本的调度模型。接着引入市场机制中的代价因素，通过综合考虑副本地理特性、网络状态、应用服务特点等因素，提出一种代价驱动的自适应副本策略，该策略针对不同应用的一致性与可用性的重要程度，以代价为驱动力来自适应地进行副本复制、副本销毁、副本迁移等操作，以达到负载均衡等目的。最后提出适用于该调度模型的基于副本和数据中心网络结构的调度策略。

4.1 引言

根据 IDC 的数据，到 2013 年，云存储服务的增长率预计将超过所有其他 IT 云服务。在未来四年内，云计算的服务市场规模将从现在的 174 亿美元增长到 442 亿美元，其中，云存储的市场比例将从目前的 9%增长到 14%，也就是说云存储的市场规模将接近 62 亿美元（云存储内部结构深度解析，2010）。

可见，云存储有着广阔的发展前景。然而，如何保证云存储系统的各方面要求，如扩展性、可用性、可靠性、安全性、效率等，是我们需要解决的问题（Sun 公司，2009）。

在分布式系统中，副本是一种提高可用性和性能的重要方法（Andrew D.，1982）。副本弥补了存储对象单点失效、容错性差、接入性能不高等问题。但同时，引入副本机制也必然带来以下几个方面的问题：副本一致性问题、负载均衡问题以及由副本产生的各种硬件和通信上的代价问题等等。同时，由于云存储自身的一些商业特性，使得云中的副本问题存在一些新的挑战：

网络的广域性与动态性。云计算服务提供者通常在全球范围内的适宜地点建立大型数据中心来向广大客户群提供服务。一方面，数据中心的地理分布性即网络的广域性，使得副本策略不得不考虑地理特性带来网络延迟的影响，其中也包括带宽所带来的负载均衡方面的影响。另一方面，由于数据中心大多使用廉价的 PC 机（Nicolas B.，2009；E. Pinheiro，2007），数据中心错误发生较为频繁，如电源错误、机架错误、网络错误、硬盘错误、系统过热、网络攻击、自然灾害等。因此，网络变化较为频繁，从而使网络中单个副本失效成为一种常态。对于云存储环境中的副本策略来说，网络广域性和动态性使得副本策略需要考虑副本失效和网络延迟所带来的影响。

商业利益追求。云存储作为一种商业行为，服务提供者追求的目标是在保证服务质量的前提下实现利益最大化。保证服务质量的一个重要的前提是提高可用性和保证高一致性，而追求两者必然带来开销代价的增大。如何在满足用户体验的前提下，减小开销使利益最大化，是云存储服务提供者面临的问题。

一致性与可用性的权衡。Brewer 于 2000 年提出了分布式系统中的 CAP（Consistency Availability Partition-tolerance）特性（Gilbert S.，2000；Vogels W.，2009），即一致性、可用性、分区容错性三者在系统实现中无法兼顾。其中，一致性是指数据在进行更新时，保证各副本之间的一致状态；可用性是指数据对象的响应性能；分区容错性是指网络的可靠性。因此，在网络分区是给定的情况下不能同时满足一致性和可用性的要求，这就带来了一致性与可用性的权衡问题。在云存储服务中，应用类型与用户群体的多样性带来了一致性与可用性需求的多样性。一部分应用要求高一致性而可以忍受在可用性上的一些牺牲，如航班订票业务；另一部分应用则对一致性的要求相对不高，而对用户能在任何时间接入要求苛刻，如电子商务业务。同时，对于不同的应用服务，用户请求会呈现各种不同的特点，如写集中、读集中等。如何在一致性和可用性不可同时达到的前提下，权衡两者使得用户得到更好的体验，是一个值得研究的问题。

与此同时，副本作为云存储系统中不可或缺的一部分是存储系统的价值所在。类比到社会环境中，云存储中的副本即为社会环境中有着自身价值的商品。这些遍布世界各地的商品通过一个全球的市场环境进行流通，并且在市场这只“看不见的手”的指引下达到一定程度上的资源利用率的提高和分布的优化配置。

因此，本章针对云存储中的副本及基于副本的调度问题，提出基于副本的调度模型。引入市场机制中的代价因素，通过综合考虑副本地理特性、网络状态、应用服务特点等因素，提出一种代价驱动的自适应副本策略，以代价为驱动力来自适应地进行副本复制、副本销毁、副本迁移等操作，以达到负载均衡等目的。最后提出适用于该调度模型的基于副本和数据中心网络结构的调度策略。

4.2 基于副本的调度模型

在这一节，我们提出基于副本的调度模型，并说明该模型中的调度流程。

4.2.1 基于副本的调度模型

基于副本的调度模型如图 4.1 所示。

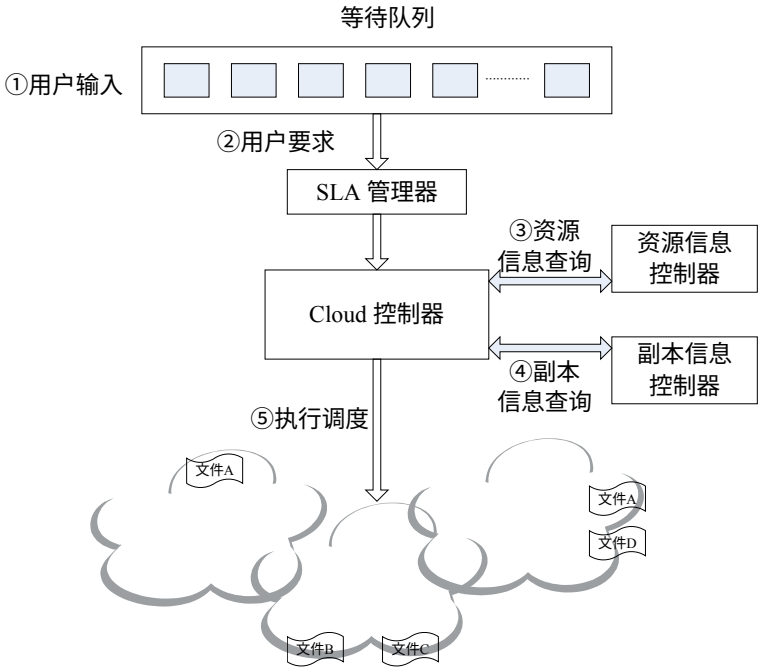


图 4.1　基于副本的调度模型

该模型的各组成部件分别说明如下：

等待队列：负责存储用户提交的任务，并将任务排队等待。

SLA 管理器：负责归类并记录用户提交任务的相关需求（如 QoS 需求等）。

资源信息控制器：负责收集并记录当前空闲机器资源的相关信息列表。

副本信息控制器：负责收集并记录副本的相关信息列表。

Cloud 控制器：负责收集资源信息控制器和副本信息控制器的信息，将用户提交的任务调度到某机器资源，完成调度任务。

由于该模型中的资源信息控制器、副本信息控制器，以及 Cloud 控制器均为全局控制器，为防止出现“ 热点 ”和单点失效问题，我们可以适当增加这三个控制器的个数，或者采用 P2P 中半分布式的结构，使得这三个控制器的负载均衡。

如图 4.2 所示，根据数据中心网络结构的不同，可以将 Cloud 控制器扩展为 3-8 个（或若干个），每个控制器在功能上都是一个单独的 Cloud 控制器，将一个 Cloud 控制器的负载均衡处理。

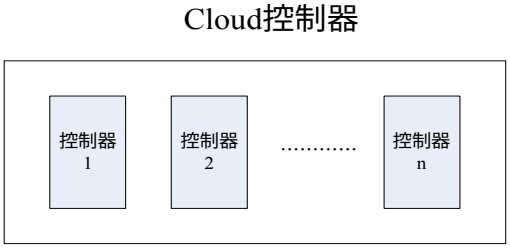


图 4.2　包含 3-8 个 Cloud 控制器的结构

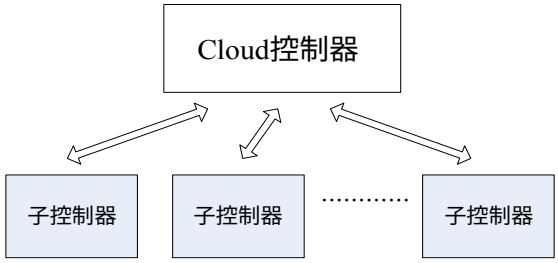


图4.3　半分布式Cloud控制器的结构

或者采用 P2P 中半分布式结构，在 Cloud 控制器中设置子控制器，如图 4.3 所示，每个子控制器的功能相同，负责处理 Cloud 控制器的一部分工作。各个子控制器组成了 Cloud 控制器，实现了 Cloud 控制器的功能。

资源信息控制器和副本信息控制器可以采用同样的方法解决“ 热点 ”和单点失效问题。

4.2.2 调度流程

基于副本的调度模型的调度流程如下：

（1） 用户提交任务，由等待队列负责存储排队用户提交的任务。

（2） SLA 管理器归类并记录用户提交任务的相关需求，将任务提交给 Cloud 控制器。

（3） Cloud 控制器在调度任务之前，首先查询资源信息控制器，查找当前空闲资源，并且空闲资源需符合 SLA 管理器中提交的用户要求。

（4） Cloud 控制器查询副本信息控制器，查找当前用户任务所需副本所在的目标机器。

（5） Cloud 控制器综合（3）和（4）中资源信息控制器和副本信息控制器提供的信息，采用相应调度算法，调度完成任务。

在整个调度模型中，副本放置策略（阶数及位置的解决）以及 Cloud 控制器采用的调度策略（调度流程第 5 步）我们在 4.3 和 4.4 部分分别介绍。

4.3 代价驱动的自适应副本策略研究

在这一节，我们介绍副本相关策略，提出代价驱动的自适应副本策略。

4.3.1 副本策略相关工作

研究云存储服务中的副本策略在很大程度上可以借鉴分布式系统中的副本策略。作为提高可用性与性能重要手段的副本策略，其主要研究点包括：复制方式、副本创建及调整策略等。同时，副本策略可分为静态和动态两大类。静态的副本策略基于已知的访问方式，副本阶数与放置位置保持不变；动态的副本策略则是根据访问方式的变化，动态改变副本阶数与放置位置。

副本复制方面，先后出现了传统悲观复制机制与乐观复制机制两类（Yasushi S.，2005）。悲观复制机制保证一个高一致性的副本，仅在副本被证明是最新副本时，才可进行存取操作。该复制机制效率低，开销大。而乐观复制机制采用并发控制，允许副本暂时出现用户不可见的不一致状态。该复制机制具有高可用性，高效率的优点，是目前较为普遍的复制机制。

副本创建及调整策略方面，前期的研究主要从性能方面考虑（Madhukar K.，1999；B. Li，1999）。其后由于硬件性能的飞升，出现了从可用性等方面出发的研究。文献（Haifeng Yu，2002）针对网络服务效用限制主要在可用性而非性能上的问题，提出了从可用性角度出发的副本放置策略。该策略通过动态进行副本的创建与删除操作，达到在可用性得到保障的前提下最小化代价的目标。但该文并未考虑相关的应用特性以及副本的一致性与可用性的权衡问题。文献（Rashedur M.，2008）在考虑到网格的动态特性后，提出了三种模型 p-median, p-center, multi-objective，分别针对用户请求、网络延迟、综合用户请求与网络延迟。实验表明，考虑用户请求的模型与综合考虑的模型优于只考虑网络延迟的模型。但该文只考虑了用户请求与网络延迟两个方面，并未考虑副本代价与副本地理特性等方面。文献（Nicolas B.，2009）提出了一种在保证一定可用性前提下，通过动态调整副本达到副本代价最小化的目标。但该文并未考虑一致性与可用性的权衡等问题。文献（Zhou Xu，2004）提出了一种基于访问统计预测的副本模型 FDRM。该模型通过预测下一阶段访问情况，对副本进行自适应的操作，达到最小化系统开销的目标，但该文未考虑到负载均衡的问题。

随着云存储的广泛认同，现已有很多公司拥有自己的云存储系统，其中有 Amazon 公司的 Dynamo（DeCandia G.，2007），Google 公司的 GFS（Chen CT，2009）。

Dynamo 采用 key-value 存储方式，将系统中的副本映射到虚拟节点组成的结构化的环上。该系统采用结构化的分布式结构，不存在单点失效及瓶颈问题，具有高可用性、高可扩展性的特点。在副本方面，通过版本控制、类 Quorum 技术以及分布式副本同步协议等，解决副本一致性问题。采用基于 Gossip 的方式进行成员发现和错误检测。

GFS 则由单个 master 和多个 chunk-server 组成，为避免单点失效及瓶颈问题，单个 master 还配备有多个辅助 master。系统中的元数据被划分成多个大小相等的块存储于 chunk-servers 上。master 作为管理节点，负责处理 chunk-server 的任务分配、状态监控、故障监控及恢复等。在副本方面，master 负责控制副本机制与负载均衡。

综上所述，现有的云存储系统中的副本管理机制均相对简单，副本阶数、位置相对固定，对副本在整个生命周期中的代价和效率未做充分考虑。基于以上问题，借鉴分布式系统中的一些已有成果，我们提出了一种云存储系统中代价驱动的自适应副本策略 CDRS，来更好地迎合云存储服务的新要求。

4.3.2 云存储系统结构

4.3.2.1 云存储系统结构模型

与传统的存储设备相比，云存储不仅仅是一个硬件，而是一个由网络设备、存储设备、服务器、应用软件、公用访问接口、接入网和客户端程序等多个部分组成的复杂系统。各部分以存储设备为核心，通过应用软件来对外提供数据存储和业务访问服务（Nicolas B.，2009）。

目前，业界广泛认为云存储系统结构模型从下往上共由 4 层组成：存储层，基础管理层，应用接口层，访问层，如图 4.4 所示。

我们讨论的副本策略问题处于基础管理层。该层通过集群、分布式文件系统和网格计算等技术，实现云存储中多个存储设备之间的协同工作，使多个的存储设备可以对外提供同一种服务，并提供更大、更强、更好的数据访问性能。

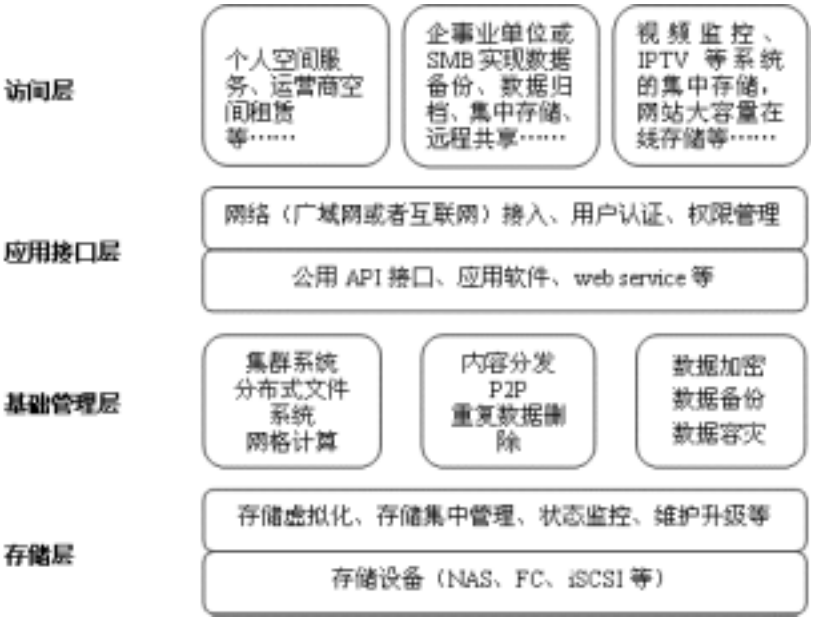


图 4.4 云存储系统结构图

具体到当前的一些云存储系统平台，Google 的 GFS 和 Amazon 的 Dynamo 等，均由分布在世界各地的大型数据中心组成。因此本章讨论的云存储环境，其实际分布结构 MSCS (Master-Slave Cloud Storage)如图 4.5 所示。该结构由物理节点 master 和 slave 组成。master 的作用是实现 slave 的任务分配、状态监控、副本策略、负载均衡等。为了避免单个 master 出现单点失效故障的情况，系统还配备了多个辅助 master。在 master 出现单点失效的情况下，自动通过选举算法产生出新的 master。而 slave 是数据存储和读写的载体，完成 master 分配的任务并定期向 master 报告自身状态，并且可以根据自身性能和当前应用需求虚拟出一个或多个虚拟节点 vnode。

图 4.5 云存储分布示意图

本章的讨论基于无单点失效的理想情况。在实际情况中，系统包含 1 个主 master 和多个辅助 master，以防止 master 单点失效的情况。我们还规定，读请求由离请求端最近的副本提供并只涉及到一个副本，在一致性协议许可条件下的任意节点上，副本的写更新均需被传播到其他所有副本上。

4.3.2.2 物理节点

每个 slave 即为一个物理节点，表示的是某个云数据中心中的一台物理机。与虚拟节点不同之处在于，两个物理节点之间的网络接入及负载等情况之间是相互独立的，而隶属于同一物理节点的两个虚拟节点在这些方面是相同的且相互影响。

由于副本所在节点的地理特性因素与数据的一致性和可用性有着密切的联系，因此我们在此将地理特性作为 slave 的固有属性加以考虑，将地理特性定义为向量 P=（洲编号，国家编号，地区编号，数据中心编号，机架编号，物理机编号）（Nicolas B.，2009）。

编号规则为：亚洲，欧洲，北美洲，南美洲，非洲，大洋洲分别为 0~6；国家、地区编号采用电话区号区分；数据中心编号采用数据中心名称缩写进行区分，机架编号与物理机编号均为数据中心内部统一编号。

如（0，86，0551，USTCNIC，01，01）表示的是亚洲中国合肥中国科学技术大学网络中心数据中心第 01 号机架上的 01 号物理机。而（0，86，0551，USTCNIC，01，03）与（0，86，0551，USTCNIC，01，01）则为同一个机架上的两台不同的物理机，即两个不同的物理节点。

4.3.2.3 虚拟节点 vnode

由于虚拟节点的地理特性依赖于它所在的物理节点的地理特性，因此我们规定虚拟节点的地理特性与其所在的物理节点的地理特性相同。在此基础上，定义虚拟节点距离表征两个虚拟节点间耦合的紧密程度，如定义 4.1 所示。

定义 4.1　设云存储系统中有两个虚拟节点 j，k，这两个虚拟节点所在的物理节点的地理位置分别为 P_j、P_k，则两虚拟节点所在物理节点的距离向量为 $Pdistance_{j,k}$，两虚拟节点之间距离为 $Vdistance_{j,k}$。计算方式如公式（4.1-4.2）所示。

$$Pdistance_{j,k} = (j \oplus k) \tag{4.1}$$

$$Vdistance_{i,j} = \frac{Pdistance_{i,j} \cdot band^*}{min\{band_i, band_j\}} \tag{4.2}$$

其中 $band^*$ 为一个估计定值，是系统中带宽的一个平均值。

$Pdistance_{j,k}$ 为一个六位二进制数，由物理节点编号的各元素异或所得。

其主要的影响因素有：

地理距离：地理距离的增大会直接导致延迟的增加以及数据传输可靠性的降低；

传输带宽：传输带宽决定了数据的传输速率，进而影响到两个虚拟节点间紧密程度。

如表 4.1 所示，节点 i 表示亚洲中国合肥中国科学技术大学网络中心数据中心第 01 号机架上的 01 号物理机（0，86，0551，USTCNIC，01，01）。节点 j 表示亚洲中国合肥中国科学技术大学网络中心数据中心第 01 号机架上的 03 号物理机（0，86，0551，USTCNIC，01，30）。则节点 i，j 的地理距离为 000001=1，同理，节点 k，j 的地理距离为 001111=15。虚拟节点之间的距离为地理距离与带宽之积。

表 4.1　虚拟节点间距离向量

节点 i	0	86	0551	USTCNIC	01	03
节点 j	0	86	0551	USTCNIC	01	01
节点 k	0	86	010	TSUNIC	03	05
$Pdistance_{i,j}$	0	0	0	0	0	1
$Pdistance_{k,j}$	0	0	1	1	1	1

4.3.2.4 数据划分

我们采用固定大小的方式对数据对象进行划分。云存储中的每个数据对象划分为 M 个大小固定的块，每个数据块的大小均为 64MB。其中的块 i (i∈M)的 r_i 个副本分别分布在虚拟节点集合 $S = (s_1, s_1, \cdots, s_{r_i})$ 上。这些虚拟节点遍布世界各地的数据中心，既可以在不同的大洲之间，也可以在同一数据中心不同的机架上。

4.3.2.5 副本覆盖范围指标

对于一个读集中的应用，其目标是尽可能地增加本地或近距离读所占的比例，因此需要将副本分布在较为广阔的节点集合 S 上来提高读操作的性能和可用

性；而对于一个写集中的应用，因为在每个副本上触发的写更新均需要被传播到其他副本上，广阔的副本分布会增加更新的通讯开销和不一致性的风险，所以需要副本节点集合 S 比较紧密的分布来减小副本更新的通讯开销。因此，在同等情况下，地理分布越稀疏，其可用性越高，而进行一致性维护的代价越大，数据一致性也就越低。

我们通过定义一个指标 t_i 来表示副本节点集合 S 分布的广阔性与紧密性，并以此为依据控制系统中副本增加、删除、迁移等操作，进而调整一致性与可用性之间的平衡。副本覆盖范围指标 t_i 为副本对象 i 所分布的平均距离，如公式 4.3 所示。

$$t_i = \frac{\sum_{k,j \in S} Vdistance_{k,j}}{r_i} \tag{4.3}$$

当副本覆盖范围指标 t_i 大于疏密上界 t_u 时，表示副本覆盖范围比较稀疏。相对可用性来说，一致性保证降低；而当副本覆盖范围指标 t_i 小于疏密下界 t_d ，表示副本覆盖范围比较紧密。相对一致性来说，可用性降低。因此，对于不同的应用和不同的用户群体访问，最优的 t_u ， t_d 取值有所不同。

4.3.3 问题定义

4.3.3.1 可用性

云存储服务的广域性与动态性给可用性带来了更多的要求。同时，随着机器性能的攀升，服务质量变得相对更受可用性的限制。一份最近研究显示，网络问题致使用户在 1.5%-2%的时间内无法接入集中式服务（Nicolas B.，2009）。导致这一现象的原因有：网络错误、网速过慢、一致性要求无法得到满足等。云存储作为一种商业产品，强调服务的质量与信誉。尤其那些对一致性要求不高的应用服务，如电子商务服务等，可用性直接影响着用户的使用体验。

目前对于可用性的定义尚无统一的标准。我们从用户的角度将单个被划分的数据对象副本块 i 的可用性定义为用户提交并得到服务的接入占总接入的比例。对于某个数据对象 i 在虚拟节点 $k(k \in S)$ 上副本的可用性定义为公式（4.4）：

$$avail_{i,k} = \frac{acceptedaccesses_{i,k}}{submittedaccesses_{i,k}} \tag{4.4}$$

则对于网络中某一数据对象 i 而言，其可用性为公式（4.5）：

$$pa_i = 1 - \prod_{k=1}^{r_i} avail_{i,k} \tag{4.5}$$

这种由用户角度定义的可用性的影响因素有副本规模、用户总量、网络可靠性、一致性级别及维护协议等。由公式(4.4)可知，可通过增加副本数量 r_i 以及提高单个数据对象块的可用性 $avail_{i,k}$ 来提高可用性。一般情况下，$avail_{i,k}$ 和用户网

络状况、一致性级别及维护协议等因素有关，较为复杂，故不做讨论，这里主要讨论通过增加副本数的方法提高副本可用性的情况。

4.3.3.2 一致性

高一致性是所有分布式系统追求的目标，是应用服务正确性的保证。同时，不同的应用服务对于一致性方面的要求也不尽相同。我们针对应用一致性要求的多样性，并以维护副本一致性的代价作为衡量一致性水平的标准，定义了拥有 r_i 个副本的数据对象 i 的维护副本一致性代价为公式（4.6）：

$$Ccost_i = w \cdot \sum_{k,j \in S} Vdistance_{k,j} \tag{4.6}$$

其中 w 表示更新一个副本的代价。

4.3.3.3 负载均衡

云存储系统由于规模较大且分布较为广泛，必然会有负载不均衡的现象。这样的现象会引发节点失效、服务性能降低等问题。通过引入市场机制，我们利用节点的虚拟价格对各节点的负载进行动态调整，同时达到最小化副本代价的目的。

定义 4.2　对于云存储系统的虚拟节点 i，其虚拟价格 $Vrent_i$ 为该虚拟节点所在物理节点的单位物理硬件使用价格 $Prent_i$（包括内存费用，外存费用，带宽费用，处理费用）以及该物理节点的负载情况 $Pload_i$。

$$Vrent_i = (1 + \theta \cdot Pload_i) \cdot Prent_i \tag{4.7}$$

公式（4.7）中，θ 为负载所占权重。

4.3.4 代价驱动的自适应副本策略

4.3.4.1 副本代价

引入副本机制必然带来副本的相关开销。文献（Haifeng Yu，2002）将副本开销 cost 定义为副本创建、销毁、使用的代价之和，这个定义并未考虑到副本所在虚拟节点的负载情况和维护副本一致性的代价。

我们定义的副本代价与两个因素有关：节点的虚拟价格和维护副本一致性代价，其中前者用来负载均衡，后者用来平衡一致性与可用性。

定义拥有 r_i 个副本的数据对象 i 的副本代价为副本所租用的虚拟节点的虚拟价格与维护一致性所需代价之和，即公式（4.8）

$$cost_i = \rho \cdot Ccost_i + \sum_{j \in S} Vrent_j \tag{4.8}$$

其中，ρ 是维护一致性代价所占的比例，ρ 的取值与应用服务读写特性有关。

4.3.4.2 副本收益

为了表征副本对象的利用程度来优化副本机制，迎合云存储服务盈利最大化的特点，我们引入副本收益的概念，其定义如下：

定义 4.3　设云存储系统中数据对象 i 有 r_i 个副本分布在虚拟节点集合 $S=(s_1,s_1,\cdots,s_{r_i})$ 上，在 Δt 时间内该数据对象 i 的第 j 个副本的访问次数为 $popularity_i^j$，则该数据对象 i 的副本收益为公式（4.9）：

$$profit_i = stdrent \cdot popularity_i - cost_i \tag{4.9}$$

$$popularity_i = \sum_{j \in S} popularity_i^j \tag{4.10}$$

其中 $stdrent$ 是标准价格，用来将访问次数转换为货币单位。

4.3.4.3 副本策略

针对云存储服务的一些新的特点，代价驱动的自适应副本策略可在保证云服务用户良好使用体验的前提下达到均衡负载、降低开销的目的。我们考虑的副本策略包括的操作有副本复制、副本删除、副本迁移，其算法逻辑如图 4.6 所示。

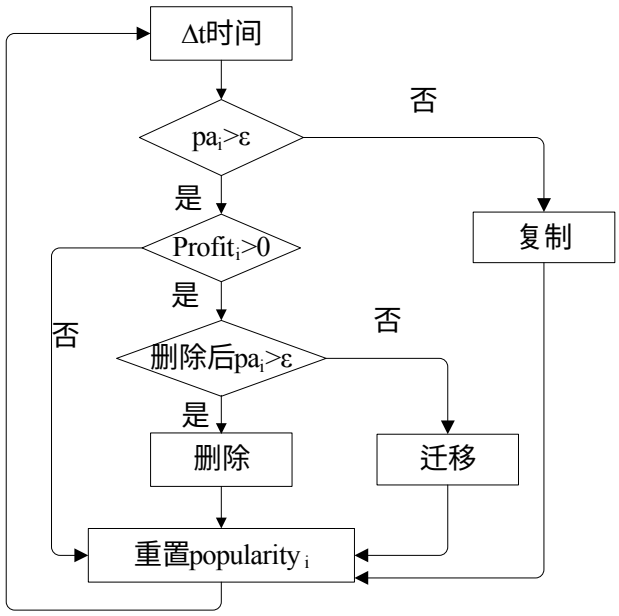


图 4.6 自适应的副本策略流程图

- 副本复制

当云存储系统中数据对象 i 的可用性无法达到要求，即小于阈值 ε 时，通过增加副本数来提高数据对象 i 的可用性。新增加的副本的放置位置是进一步需要考虑的问题。

新增副本节点 i 的放置位置可根据副本覆盖范围指标 t_i 以及副本代价最小化的原则来选择，如算法 4.1 所示：

算法 4.1：副本复制算法

1)　任意副本 $k(k \in r_i)$

*2)　if　*$(t_i \le t_d)$　*/*提高副本覆盖范围指标*/*

3)　　找到与副本 k 距离大于 t_i *的节点范围* G *；*

*4)　else　　/*降低副本覆盖范围指标*/*

*　　找到与副本 k 距离小于等于* t_i *的节点范围* G *；*

5)　$j = \{ j \in G | min\, cost'_i \}$ *；*

- 删除副本

当云存储系统中数据对象 i 的可用性得到满足时，可以考虑降低副本代价。降低副本代价可通过删除副本或者迁移副本到低成本的虚拟机上来实现。若该数据对象 i 的副本收益为盈利状态，则通过删除副本达到降低副本代价的目的。删除的副本应为副本收益最小的副本，即删除的副本 j 如公式（4.11）：

$$j = \left\{ j \in S \middle| min\left\{ popularity_i^j - Vrent_j \right\} \right\} \tag{4.11}$$

- 迁移副本

当云存储系统中数据对象 i 的可用性得到满足但副本收益为亏损状态时，考虑将收益低的副本迁移到代价低的虚拟节点上，以达到降低副本代价，提高副本收益的目的。首先依据公式(4.11)删除副本 j，再依据增加副本的算法复制一个副本，完成迁移的操作。

4.3.5 实验模拟与分析

4.3.5.1 实验环境

我们讨论的云存储系统由 master 和 slave 组成。master 中保存着每个 slave 的相关信息，包括 Δt 时间内各 slave 的访问量、响应量和 slave 的路由信息。master 负责实现任务分配、状态监控、副本策略等工作。各 slave 保存着各自的地理特性、负载状态、带宽等信息。

模拟程序采用 C++编写。系统中有 60 个 slave，每个 slave 可根据自身性能虚拟出 g 个 vnode，$g \in [1,5]$。初始状态时系统中共有 M 个大小为 64MB 的数据对象，每个数据对象均拥有 3 个分布在不同 slave 上的副本。各副本访问概率服从泊松分布，访问量和访问类型为随机产生。由于模拟试验中不存在网络拥塞等情况，以及副本的接入数等于总请求接入，因此设置副本可用性 $avail_{i,k}$ 为 0.9。模拟实验中的参数取值如下：$\Delta t = 600s$，$\theta = 0.3$，$\rho = 0.5$，$t_u = 31$，$t_d = 15$，$\varepsilon = 99.9\%$。

实验的对比策略为传统静态副本策略即始终保持 3 个副本且副本位置不变。

4.3.5.2 实验结果与分析

- 负载均衡

本实验通过各 slave 的负载方差来衡量该算法在负载均衡上的性能。

图 4.7 为 Δt 时间间隔下副本对象数目保持 M=50 不变情况下 CDRS 策略的负载方差变化。从图 4.7 可以看出，在副本访问不断变化的过程中，该策略在负载上一直处于相对平缓的变化，且负载方差值较低，这表明了该方法在负载均衡方面的表现有一定的普遍性。

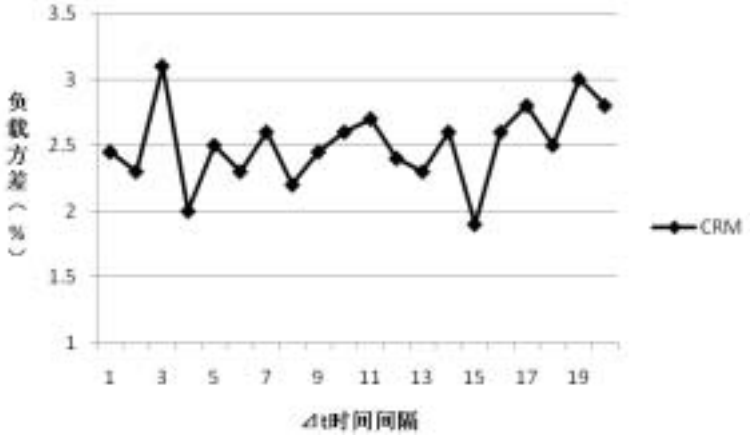


图 4.7　Δt 时间间隔下 CDRS 策略负载方差变化(M=50)

在副本对象个数呈线性增长的情况下，与传统副本策略相比，CDRS 策略在初始状态时与传统副本策略负载均衡相当。在副本数量明显小于节点个数时，各节点的负载均衡有一定的随机性。但当副本数量增大到一定范围，CDRS 策略明显优于传统副本策略，如图 4.8 所示。这是因为，静态传统策略的副本放置是一种简单随机的，当副本数量达到与节点个数呈现一定数量规模时，会无法保证负载均衡方面的性能。而 CDRS 通过副本代价和副本收益，进而考虑到节点的负载均衡，因而在负载方面比静态传统策略表现良好。

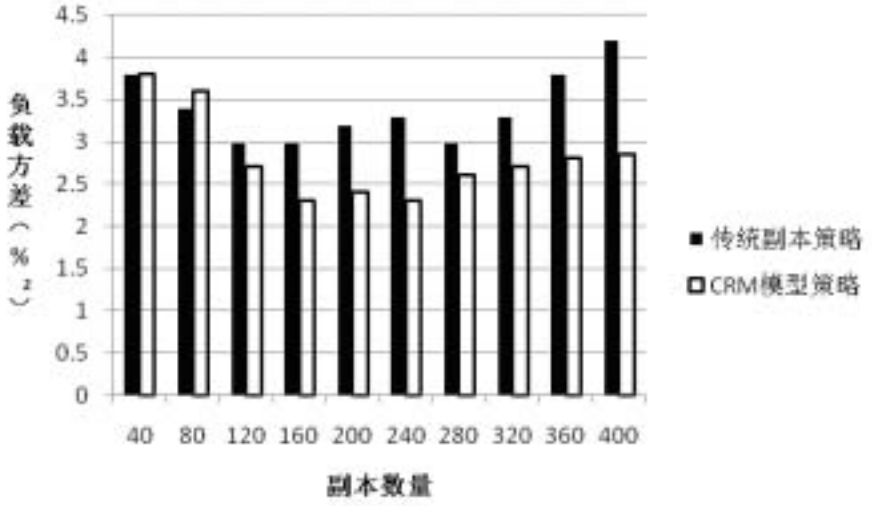


图 4.8 负载方差变化随副本数量变化情况

- 副本收益比较

副本收益可以用来从一方面衡量云存储服务商的盈利程度，因为它从一个方面体现了副本的利用程度。我们采用 Δt 时间间隔下平均副本收益来比较 CDRS 策略与静态传统在副本数目保持 M=50 不变情况下副本收益上差异。从图 4.9 可以看出，通过副本代价驱动的特性，CDRS 策略可以得到比静态传统策略更高的副本收益，并且在副本收益为亏损的情况下，可以通过自适应的副本操作使副本收益扭亏为盈。

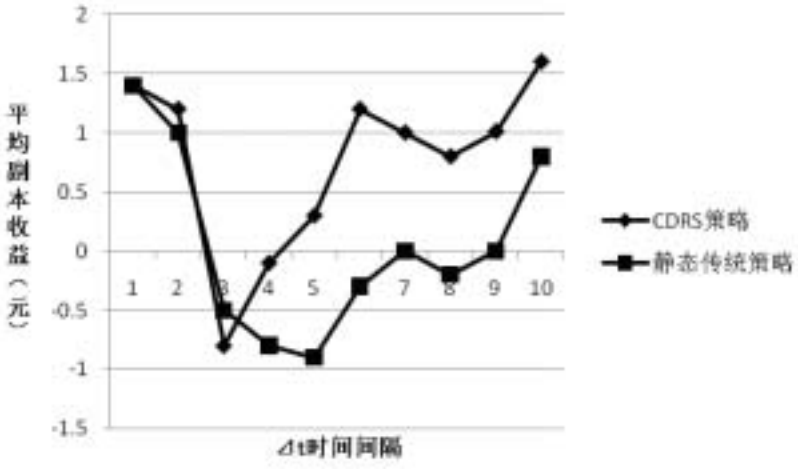


图 4.9　Δt 时间间隔下平均副本收益（M=50）

4.4 基于副本和数据中心网络结构的调度策略研究

在这一节，我们详细介绍在 4.2 节基于副本的调度模型中，Cloud 控制器采用的任务调度算法。

由于我们考虑了数据中心网络结构，因此引入 K-means 算法（Beckmann N.，1990），将符合用户需求的所有空闲资源按数据中心网络结构中距离的远近划分为 K 个聚类（集群）。当 K 个集群中仅有一个集群满足用户需求时，Cloud 控制器调度任务到该集群中完成任务；当 K 个集群中有多个集群都满足用户需求时，我们参考副本的位置信息，进一步选取合适的集群。

4.4.1 K-means 算法确定集群聚类

K-means 算法（于翔，2007；陆云，2007）是指输入聚类个数 K，以及包含 n 个数据对象的数据集，将这 n 个数据对象划分为 K 个聚类输出，这 K 个聚类满足：同一聚类中的对象相似度较高，而不同聚类中的对象相似度较小。

K-means 算法是一种典型的逐点修改迭代的动态聚类算法，其要点是以误差平方和为准则函数。逐点修改聚类中心：一个象元样本按某一原则，归属于某一组类后，就要重新计算这个组类的均值，并且以新的均值作为凝聚中心点进行下

一次象元素聚类；逐批修改聚类中心：在全部象元样本按某一组的类中心分类之后，再计算修改各类的均值，作为下一次分类的凝聚中心点。

由于 K-means 算法的计算结果和迭代性能依赖于初始 K 个对象（聚类中心的初始 K 个位置）的选择，这 K 个对象选择位置不同，可能导致聚类结果不同，更可能产生无解情况。因此，后来提出了很多的优化算法（Higgs R.，1997；Snarey M.，1997；Kanfman L.，1990；钱线，2007）。考虑到数据中心网络结构，尤其是第三章提出的雪花结构，我们采用 Millan（Milligan G.,1980）提出的 K-means 改进算法。该算法中，K-means 的 K 个初始值是由 Ward（Ward J.，1963）的层次聚类方法得到的。层次聚类方法认为，每个数据点都是一个独立的类，每次迭代时都从已有类中选出最相似的两个类合并，直至合并到仅有一个类。在用 K-means 算法求解聚类之前，首先用层次聚类方法做聚类，当类的个数合并为 K 时，停止聚类，求出这 K 个类的中心作为 K-means 算法的 K 个初始位置。算法流程如图 4.10 所示：

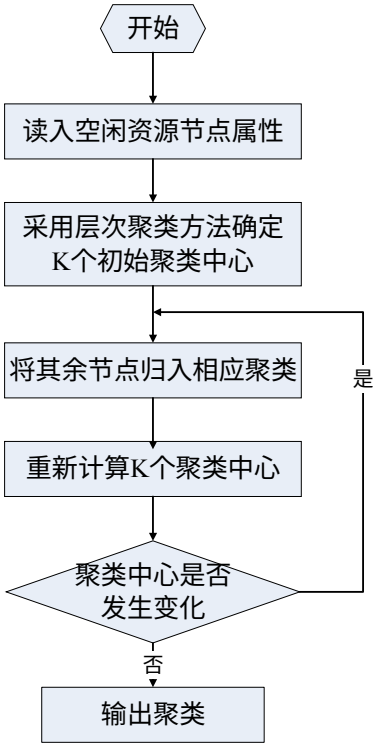


图 4.10 改进的 K-means 算法流程图

算法的基本思想如下（李东琦，2007；梁循，2005；范明，2004）：

1. 首先查找出所有符合用户需求的空闲资源（又称数据对象）；
2. 采用层次聚类方法，依据距离的远近，将同样标记的数据对象放在一起，并对数据对象个数计数，得到层次聚类分析后 K 个类的均值，将这个 K 个对象作为初始聚类中心；
3. 根据每个聚类对象的均值，计算每个对象与这些聚类中心的距离，并将

该对象归入离它最近的聚类；

4. 重新计算每个聚类中心；

5. 重复步骤 3 和 4 直到聚类不再发生变化。

由于对 K 个初始值的选取有所改进，不仅克服了 K-means 算法选择初始 K 个聚类中心的随机性带来的结果不确定缺点，而且使得算法迭代次数明显减少，提高了算法的运行速度（段明秀，2009）。总的来说，该算法不仅简单，而且收敛速度快，能扩展用于大规模的数据集。因此，该算法适用于大规模云计算数据中心网络结构，我们采用该算法对任何网络结构的数据中心求聚类。

补充说明如下：

假设空闲资源数是 N，用户需求机器台数是 M。

当 N 小于 M 时，任务等待。

当 N 等于 M 时，空闲资源组成集群，全部分配给用户，调度任务。

当 N 大于 M 时，通过控制聚类次数即可以满足用户需求。

4.4.2 副本信息确定集群聚类

当满足用户需求的集群聚类有多个时，我们使用副本的位置信息进一步确定集群的选择。

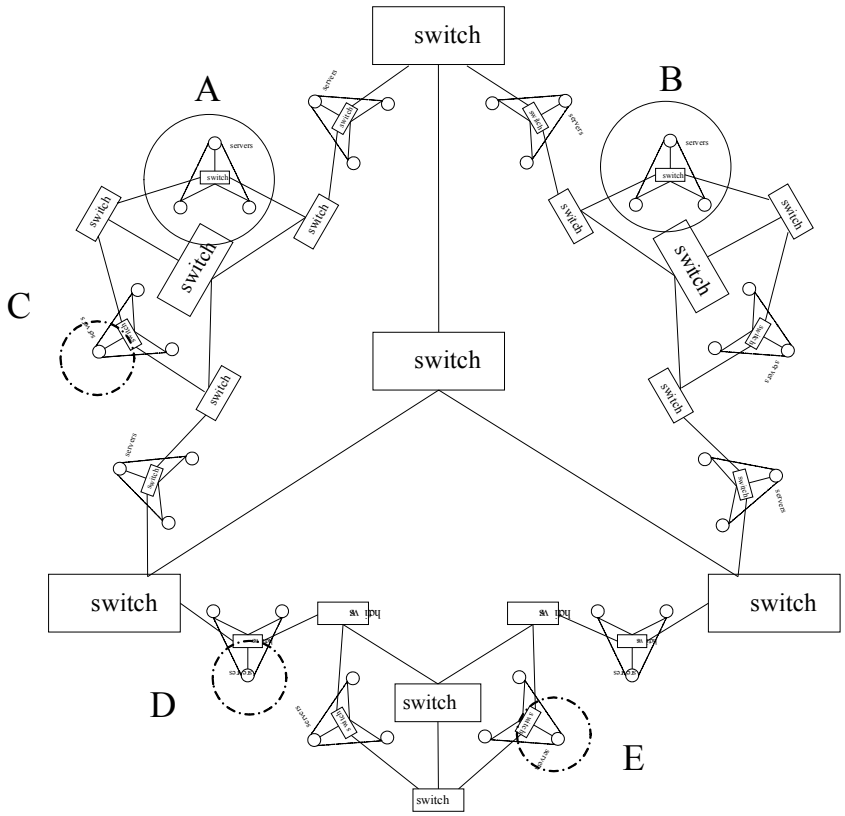


图 4.11 副本信息确定集群聚类实例

以第三章的雪花结构为例，如图 4.11 所示，实线圆圈标记的聚类 A 和 B 表示满足用户需求的集群聚类。点画线圆圈标记的 C、D 和 E 表示包含任务所需的副本信息（采用 3 个副本）。由于我们使用<级别，度数>的结构标记数据中心节点，因此，通过度数做差的方法可以很快确定 A 中的集群聚类与 C 中的副本源较近，可以选择将 C 中的副本信息复制到 A 集群聚类中，以减小副本复制的时间开销。

4.4.3 实验模拟

为了验证提出的算法，我们将基于副本和数据中心网络结构的调度策略与随机调度策略相比较。由于该调度策略的目的，是在综合考虑数据中心网络结构和副本这两个因素时，缩短任务调度之前复制副本产生的传输时间，不涉及任务在资源上的实际处理时间和服务方收益、用户付费等问题，因此采用与随机调度策略相比较的方法。

我们在第三章雪花结构的数据中心网络结构上实验验证，采用 n 为 4 时的雪花结构，空闲资源占总资源的 40%，用户申请机器数为 10。假设资源被分配给用户调度任务后，即长期使用，实验中不再考虑再分配问题。由于实验结果数据起伏变化不大，因此我们取空闲资源为 60 倍数时的实验结果做柱状图比较，如图 4.12 所示。

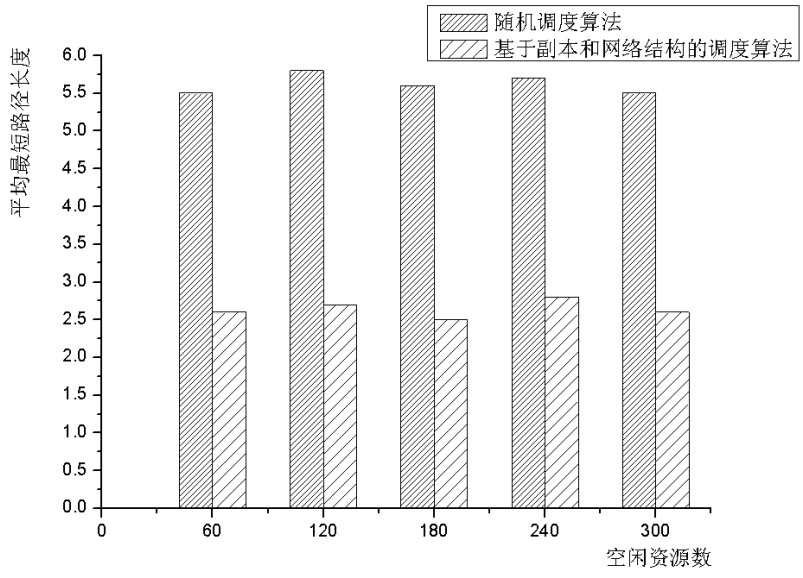


图 4.12 基于副本和网络结构的调度算法与随机调度算法的比较

由图 4.12 可以看出，我们提出的基于副本和数据中心网络结构的调度算法，其复制副本的平均最短路径大约是随机调度算法的一半，可以大大节约副本在传输复制过程中的时间开销。

4.5 本章小结

本章针对云计算中的副本及基于副本的调度问题，首先提出了基于副本的调度模型；接着引入市场机制中的代价因素，通过综合考虑副本地理特性、网络状态、应用服务特点等因素，提出了一种代价驱动的自适应副本策略，该策略针对不同应用的一致性与可用性的重要程度，以代价为驱动力来自适应地进行副本复制、副本销毁、副本迁移等操作，以达到负载均衡等目的；最后提出了适用于该模型的，基于副本和数据中心网络结构的调度策略。实验表明，该调度策略可以有效缩短任务调度之前复制副本产生的传输时间，复制副本的平均最短路径仅仅是随机调度算法的一半。

第5章　在线调度问题研究

任务调度依据调度时间的不同，可以分为批调度和在线调度两种模式。在在线调度中，一个任务只要到达就立刻被调度到某个空闲资源上执行，如果资源忙则需等待。现有云计算环境中，在线调度策略主要着眼于资源的分配管理，往往以满足用户的各种资源请求为目的，而对于云计算服务提供方关注不够。为增大服务方的收益，本章提出云计算环境中基于成本计算的在线任务调度策略，根据用户提交任务的相关信息，计算接受任务的沉没成本和机会成本，以决定是否接受任务，使得服务提供者和服务请求者实现双赢，促使云计算市场环境向健康稳定的方向发展。

5.1 引言

云计算的目的就是实现协同工作和资源共享，而云计算环境中各式各样机器资源体现出来的异构性和动态性以及用户需求的多样性，使得云计算环境下的资源管理变得异常复杂。

经济学中的市场机制能够充分激励和调动个体的积极性，在商品生产和交换中实现合理的社会资源配置，有效提高社会生产率（平新乔，2001），而由各个高异构性和动态性节点构成的云计算环境与现实经济社会具有极大的相似性，借助经济学中相关概念和方法解决云计算环境中的资源管理具有可行性和优越性（Subramoniam K.，2002）。

云计算环境中的任务调度涉及两种参与者，即服务的提供者和服务的请求者。现有在线任务调度策略（B. Chun，2001；B.N.Chun，2002；David E.，2004）往往以满足请求者的各种服务请求为目的，对于提供方的收益问题关注不够（David E.，2004）。本章着眼于云计算环境中在线任务调度策略，从提高服务方收益出发，在参考机会成本（平新乔，2001）的基础之上，引入沉没成本（王德敏，2004）的概念，研究服务方的任务接受策略。本章论述如何通过计算沉没成本和机会成本，以及协调沉没成本和机会成本的关系，实现服务方的收益更大化。与文献（David E.，2004）的任务调度策略相比，本章提出的策略能降低服务方的成本，用户得到更满意的服务，促进调度环境健康、和谐地发展。

5.2 在线任务调度的相关研究

对于在线任务调度算法，已有的研究主要是基于任务的优先级，执行任务需求的资源等等，基本都是从用户的角度出发，对于服务方考虑不充分。

文献（Cheng Chiyu，1997）提出一个在线任务调度算法，对多个具有不同优先级的任务进行动态调度。对于优先级高的任务尽量优先完成，对于优先级低的任务尽量在优先级较高的任务之后完成，以期望在到达时间底线时，优先级越高的任务被完成的可能性越大。文章提出三种启发式算法来选择动态环境中优先级最高的可以抢占其他任务的任务；而对于低优先级的任务，采取分配部分资源的策略，使得低优先级的任务在被高优先级任务抢占的情况下，也有可能在时间底线之前完成。该文提出的在线任务调度算法很好地考虑了用户提交任务的优先级，从用户的角度出发，尽量满足用户请求的优先级别。

对于在线非抢占式调度，总是假设任务是单一的，或者假设任务只需要一种资源即可执行完成，这些假设对于模拟实际情况是不合适的。文献（Kamhing Ho，1990）针对在线非抢占调度模式提出一种新的任务调度算法，考虑到任务可由一个或者多个部分组成，可能需求一种或者多种资源来完成，可以在不同的队列中等待。文章提出松散分布策略，对上述情况进行分析，实验模拟也证实算法的有效性。但是，文献（Kamhing Ho，1990）同样对于服务方的考虑并不充分。

文献（R Abbott，1992）在单处理器的前提下提出一种新的任务调度算法，目标是到达任务规定的时间底线之前，尽量多完成用户提交的任务数量。

文献（David E.，2004）在研究基于市场机制的任务调度研究中，引入了经济学中机会成本的概念，详细计算机会成本对任务调度产生的影响，以决定是否接受任务，从而降低了服务方承担的风险，使服务方获得更多的收益。但是，文献（David E.，2004）只考虑了机会成本，未详细考虑服务方其他方面的利益损失。

5.3 任务调度模型

云计算环境中的任务调度涉及两种参与者，即服务的提供者和服务的请求者。服务的提供者（简称服务方）提供用户所需服务，并负责调度完成任务。对于用户的 QoS 需求，用户可以依据自身条件（如资金、时间等）自行选择。如果用户拥有较多资金，即可以选择更好的服务（相对也更昂贵）。如果服务方未能在有效时间内（如 deadline 之前）完成调度任务，则需要对用户支付赔偿金。支付赔偿金的方法可以有效约束服务方，降低用户的付费风险。

本章的任务调度模型基于文献（David E.，2004），沿用其价值函数(value function)的概念。但是，由于本章较之文献（David E.，2004）提出了更全面的调度算法，故相应地扩展该价值函数的内涵，如图 5.1 所示。

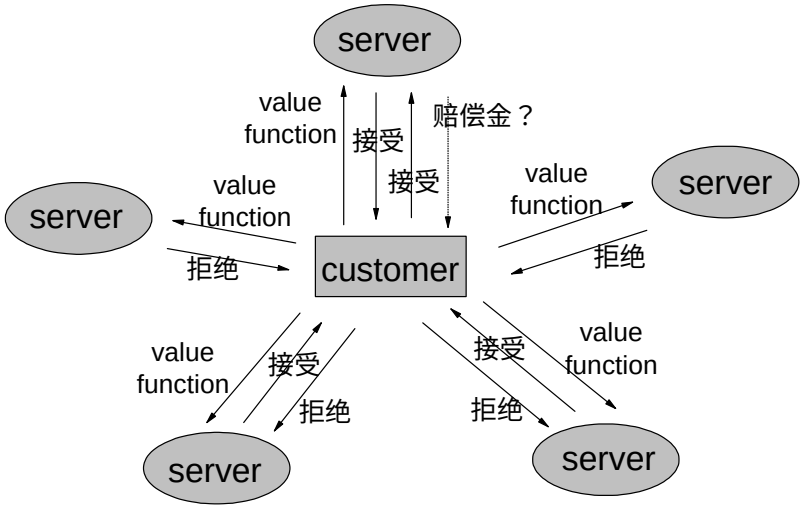


图 5.1 任务调度模型

本章涉及的任务相关的价值函数是一个五元组（deadline, reward, decay, bottomline, penalty），具体解释如下：

(1) deadline：表示任务的时间底线，反映出用户期望在此之前完成任务的时间需求，当服务方未能满足用户时间需求时，支付用户赔偿金；

(2) reward：表示用户支付服务方的费用，当服务方满足用户时间需求，不需支付赔偿金时，获得的费用；

(3) decay：表示服务方支付用户赔偿金的比率，当服务方未能满足用户时间需求时，单位时间内支付用户的赔偿金；

(4) bottomline：表示服务方支付最高赔偿金对应的时间点，当服务方未能满足用户时间需求时，随着时间的增加，支付用户的赔偿金不断增多，在此时刻，赔偿金达到最大值，且不再增加，显然， bottomline>deadline ；

(5) penalty：表示服务方支付的最高赔偿金，当服务方未能满足用户时间需求时，随着时间的增多，赔偿金也不断增多，当赔偿金数额达到 penalty 时，则不再增加。

5.4 基于成本计算的在线任务调度策略

通常，在线调度模式对于每一个到达的任务都立即分配资源，加以执行。这种调度方式的好处是可以简化调度，有广泛的实际应用（Sgall J.，1998）。

本节研究基于成本计算的在线任务调度接受策略，研究基于如下假设条件：

假设 1：在线调度中，正在执行的任务可以被其他任务抢占，且一旦被抢占，则在再次获取资源后将从头开始执行。

假设 2：在线调度中，服务方只有完成全部任务，才能得到用户提供的任务报酬。

假设 3：服务方完成的任务均满足用户 QoS 需求。

在在线调度中，可以对任务设置优先级，当服务方接收到新的任务时，如果执行新的任务将获得更高的收益，则可暂时终止对当前任务的执行，在完成新任务后重新唤醒该任务以继续执行。假设符合在线调度的特征，且有利于服务方获取较高的收益，因此假设合理。

5.4.1 首个任务的接收策略

处在空闲状态的服务方接收到第一个任务价值函数时，由于服务方的资源正在空闲，不存在任务间资源的抢占。此时，服务方根据任务的价值函数，计算接收该任务的收益，并以此判断是否接收该任务。

价值函数五元组中的 penalty 可以设置为 0，即该任务不存在最高赔偿金，赔偿金随着任务延迟时间的增加而增加，直至任务被完成，此时 bottomline 为+∞。图 5.2 给出了不存在最高赔偿金时服务方收益随时间变化的情况。

当任务不存在最高赔偿金时，服务方接受任务 i 的收益($yield_i$)见公式(5.1)和公式(5.2)，其中 $reward_i$ 为任务 i 的服务付费， $delay_i$ 为服务方完成任务 i 时超过 *deadline* 的时间， $decay_i$ 为任务 i 的超时赔偿率。

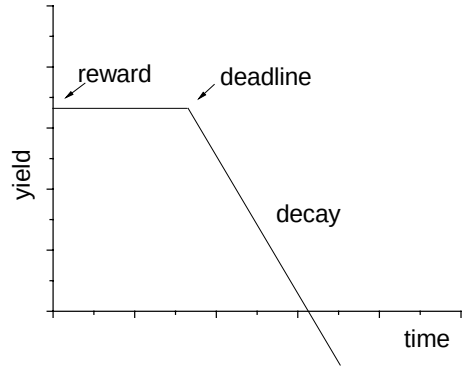


图 5.2 不存在最高赔偿金时服务方收益随时间变化的情况

$yield_i = reward_i$，服务方预计的任务运行时间 ≤ deadline　　（5.1）

$yield_i = reward_i - (delay_i * decay_i)$，服务方预计的任务运行时间>deadline　　（5.2）

如果任务设置了最高赔偿金，即 penalty ≠ 0 时，服务方收益随时间变化的情况见图 5.3。

此时服务方收益($yield_i$)的计算如公式(5.3)、(5.4)和(5.5)所示，其中 $penalty_i$ 为任务 i 的超时最高赔偿金。

$yield_i = reward_i$，服务方预计任务运行时间 ≤ deadline　　（5.3）

$yield_i = reward_i - (delay_i * decay_i)$, deadline < 服务方预计任务运行时间 < bottomline　　（5.4）

$yield_i = -penalty_i$，服务方预计任务运行时间 ≥ bottomline　　（5.5）

因此，首个任务的接收策略为：当服务方接收到用户提交的价值函数五元组时，首先估计任务运行时间，之后根据公式(5.1)和(5.2)或者公式(5.3)、(5.4)和(5.5)计算运行任务的收益，当收益 $yield_i$>0 时，即可接受该任务。

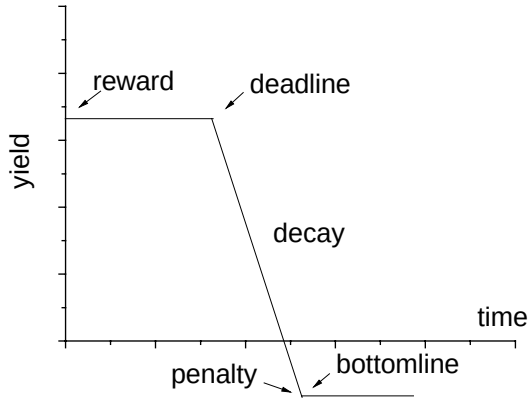


图 5.3 设置最高赔偿金时服务方收益随时间变化的情况

5.4.2 非首个任务的接收策略

“非首个任务”是指，当前正在执行任务的服务方又接收到的新任务。在此情况下，服务方分别计算当前任务的沉没成本以及新任务抢占当前任务时造成的机会成本，并以此判断是否接收新的任务。

（1）基于沉没成本的任务接受策略

沉没成本(sunk cost)指已经付出且不可收回的成本。当服务方正在执行第 i-1 个任务时，因为服务方已经投入了计算、存储或带宽资源，所以对该任务已经投入了沉没成本。定义任务被抢占前，服务方投入的任务执行时间为沉没时间(sunk time)。当服务方正在执行的第 i-1 个任务没有设置最高赔偿金时，若服务方接收任务 i 且抢占任务 i-1 时，其成本如公式(5.6)所示，其中 sc_{i-1} 为服务方为第 i-1 个任务投入的沉没成本。

$$\text{cost}_\text{i} = sc_{i-1} + delay_{i-1} * decay_{i-1} \qquad (5.6)$$

当服务方正在处理的第 i-1 个任务设置了最高赔偿金时，接收任务 i 且抢占任务 i-1 时，其成本如公式(5.7)和公式(5.8)所示，其中 s_time_{i-1} 为任务 i-1 的沉没时间。

$\text{cost}_\text{i} = sc_{i-1} + delay_{i-1} * decay_{i-1}$，

当 $deadline_i + deadline_{i-1} + s\_time_{i-1} <= bottomline_{i-1}$　　（5.7）

$\text{cost}_\text{i} = sc_{i-1} + penalty_{i-1}$，$deadline_i + deadline_{i-1} + s\_time_{i-1} > bottomline_{i-1}$　　（5.8）

若 $yield_i$-cost_i>0 表明服务方接受第 i 个任务仍可获得收益；

若 $yield_i$-cost_i=0 表明服务方不获益也不赔本，相当于没有执行任务 i，但是

浪费了执行任务 i 所花的时间；

若 $yield_i$-cost$_i$<0 表明服务方赔本；

$yield_i + yield_{i-1}$-cost$_i$>0 表明服务方执行第 i-1 个任务的收益补贴了第 i 个任务的成本，还有剩余收益；

$yield_i + yield_{i-1}$-cost$_i$<0 表明任务方执行第 i-1 个任务的收益完全补贴第 i 个任务的成本，仍然赔本；

在接收到非首个任务时，若服务方不考虑服务执行效率，只追求显性收益时，服务方计算 $yield_i$-cost$_i$ 的值，若 $yield_i$-cost$_i$>0，则可接受第 i 个任务并抢占第 i-1 个任务。

（2）基于沉没成本和机会成本的任务接受策略

基于沉没成本的任务接受策略仅仅考虑了接受新任务的显性成本，但是当服务方接受任务 i 并抢占任务 i-1 的资源时，服务方需要为此次任务接受决策消耗隐形成本，即机会成本(opportunity cost)。

机会成本是为了获取某种机会而消耗的人力、财力和物力，即把一种资源投入某一特定用途之后，所放弃的在其他用途中所能得到的最大利益。当第 i 个任务抢占了其他的任务时，服务方消耗的机会成本(oc_i)的计算如公式(5.9)和公式(5.10)所示。

$$oc_i = \sum_{j=0,i\neq j}^{n} decay_j * (bottomline_j - deadline_j)，\ 0 < deadline_i <= deadline_j \qquad (5.9)$$

$$oc_i = \sum_{j=0,i\neq j} decay_j * (bottomline_j - deadline_i)，\ deadline_j < deadline_i <= bottomline_j \qquad (5.10)$$

公式（5.9）和（5.10）中若任务 j 不存在 bottomline，则以估计任务 i 完成时的当时时间来代替 bottomline。

当服务方正在执行任务时又接收到多个任务请求，则可根据沉没成本和机会成本进行综合考虑，判断是否接受新任务。首先，服务方计算接受新任务 i 所需的显性成本，即计算每个任务的 $yield_i$-cost$_i$。然后，服务方对每个 $yield_i$-cost$_i$ > 0 的任务计算机会成本，选择沉没成本和机会成本均较低的任务抢占当前正在执行的任务。

机会成本可以用于任务紧急度的计算。当任务紧急时，可分别计算其机会成本，确定最紧急的任务优先执行（David E.，2004）。

5.4.3 收益最大化的任务接受机制

在线调度中，任务间的抢占会造成服务方的沉没成本和机会成本，服务方可根据 5.4.1 节和 5.4.2 节介绍的策略对新任务进行接受控制。但是，当多个新任务造成的沉没成本和机会成本总和相当时，服务方无法判断接受哪个新任务可以获取最大收益。例如，服务方接收到两个新任务 A 与 B 的请求，当执行 A 造成的

沉没成本高于执行 B 造成的沉没成本，然而执行 A 造成的机会成本低于执行 B 造成的机会成本时，服务方执行任务 A 或 B 均能获益，此时，依据 5.4.2 节的任务接受策略，服务方无法判断执行哪个新任务可以获取最大收益。

针对该问题，本节提出一种服务方收益最大化的任务接受机制，用于在沉没成本和机会成本总和相当或者差别可忽略不计的情况下，如何选择多个任务中的一个，实现服务方收益最大化。

定义 $result_i$ 为接受任务 i 时，服务方不同权重成本下的收益。定义平衡因子 α 和 β，分别表示收益和沉没成本所占比例，其中收益、沉没成本和机会成本所占比例总和为 1。则 $result_i$ 的计算如公式(5.11)所示，其中 $\cos t_i$ 指任务 i 抢占其他任务时造成的沉没成本开销；oc_i 指任务 i 优先执行时造成的机会成本开销；$\exp ect_i$ 指任务 i 执行时间。

$$result_i = \{\alpha * yield_i - \beta * \cos t_i - (1-\alpha-\beta) * oc_i\} / \exp ect_i \qquad (5.11)$$

其中，平衡因子 α 和 β 的获取如下：

1) 服务方执行首个任务后，根据公式（5.11），不断调整平衡因子 α 和 β，以获得最大的 $result_1$，记录此时的平衡因子 α 和 β；

2) 之后，服务方再接收到多个任务时，根据以上得到的平衡因子 α 和 β，估计这些任务的 $\cos t_i$ 和 oc_i，并计算其对应的 $result_i$，选择 $result_i$ 最大的任务执行。当该任务完成后，可获得该任务对应的实际 $\cos t_i$ 和 oc_i。重新调整 α 和 β，计算在 $result_i$ 值最大的情况下对应的 α 和 β 值，与原有的 α 和 β 值进行平均，作为之后任务接受策略所需的参数。

5.5 实验模拟和结果分析

本章使用模拟器 CloudSim（Rodrigo N.，2009）对基于市场的在线调度中服务方的任务接受策略进行模拟，采用 Java 语言，开发平台为 Eclipse 3.2.1。实验中输入的任务实例均采用文献（A. Downey，1999；V. Lo，1998；H. Shan，2003）中的任务数据，任务的到达时间和任务的执行时间均呈指数分布。我们采用与（David E.，2004）相似的模拟方法来计算机会成本，假定了用户的任务付费总体比例，20%的任务付费较高，80%的任务付费较低。至于具体任务的付费多少，均在[5,5000]内均匀随机选取。服务方对任务完成与否的赔偿率采用双峰分布来模拟。最后，定义实验模拟中涉及的任务沉没成本均服从[5,5000]之间的均匀随机分布。

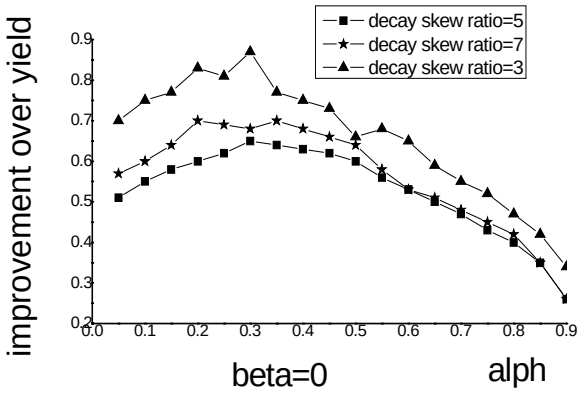


图 5.4 收益与机会成本关系图

设置 β 为 0，研究收益与机会成本的关系（由于机会成本较沉没成本动态变化更为剧烈，对收益的影响程度高于沉没成本，故不再研究收益与沉没成本的关系，只以机会成本为例来研究）。图 5.4 表明取不同的赔偿率时，机会成本所占比例与收益增加的关系。当赔偿率为 3，α 为 0.3 时收益为最大。可见，赔偿率越低，收益越大；α 较小时，机会成本占比例较大，即考虑的机会成本较多，此时收益较大。

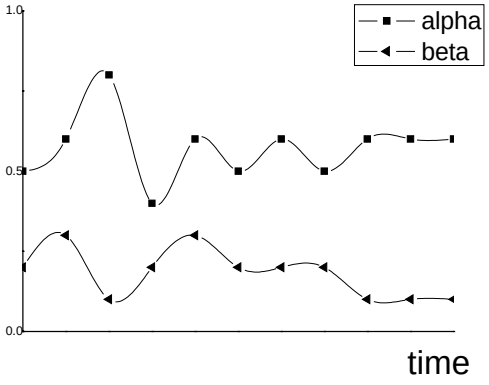


图 5.5 α 和 β 变化曲线图

图 5.5 为根据实验结果所做出的平衡因子 α 和 β 变化曲线图。刚开始时，随着任务的不断执行，新任务不断抢占原来执行的任务，服务方不断调整自身的平衡因子 α 和 β 值，所以 α 和 β 上下变动。随着时间的延长，服务方执行的任务不断增多，α 和 β 不断调整，直至达到一个平衡的状态，则基本不再变化。此时，服务方已经基本实现在此平衡因子 α 和 β 的作用下，任务的 $result_i$ 在其最大的 $result_i$ 状态，有效降低了服务方的成本，实现收益最大化。

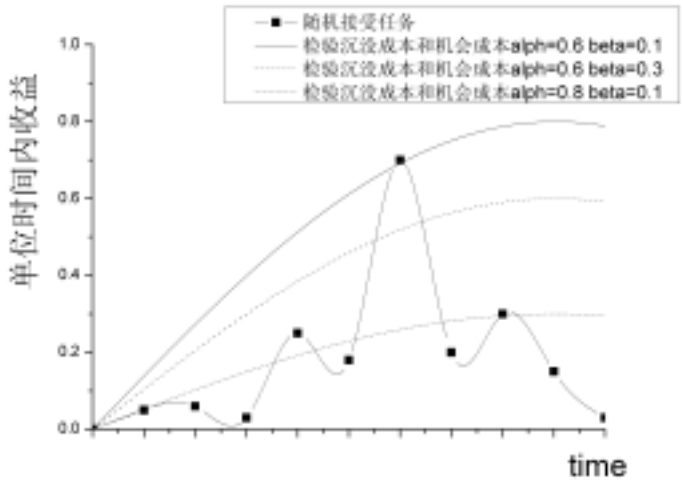


图 5.6 随机接受任务和不同 α 和 β 调节下单位时间内收益的比较

由于在整个任务接受策略中，平衡因子 α 和 β 是不断变化的，不能直观地看出调节平衡因子对服务方收益的影响。故实验取两个特殊的中间状态，一是 α=0.8，β=0.1；另一个是 α=0.6，β=0.3。在这两个中间状态时，假设 α 和 β 的值不再改变，已经成为服务方最终达到的平衡状态时对应的平衡因子。分别重新模拟之后的实验，计算各个任务的 $result_i$，按照上述的任务调度策略重新进行任务的调度。同时，取出服务方最后趋于平衡状态时对应的 α 和 β 值，α=0.6，β=0.1。以上三种状态下服务方的收益情况见图 5.6。为了更直观地进行比较，在服务方随机接受任务的情况下，也模拟出对应的单位时间收益，以做出对比，体现 α 和 β 重要的平衡作用。

5.6 本章小结

本章提出了云计算环境中基于成本计算的在线任务调度策略，首先详细阐述了服务方接受首个任务的调度策略，引入沉没成本，接着分析了在计算沉没成本和机会成本的前提下，服务方接受非首个任务的调度策略，最后比较了沉没成本与机会成本的关系，分析了沉没成本和机会成本对服务方收益的影响。模拟实验结果表明，提出的服务方收益最大化机制，有效降低了服务方接受任务时承担的成本，促进调度环境的和谐发展。

第6章　批调度问题研究

任务调度依据调度时间的不同，可以分为批调度和在线调度两种模式。在批调度中，任务到达后并非立即得到服务，而是被收集成一个任务集合等到某个时间或者事件后一起调度到资源。现有云计算环境中，批调度策略的研究主要着眼于资源的分配管理。本章充分考虑优先级、时间调度底线、收益、资源风险等调度因素，提出服务质量驱动下的任务调度算法和基于 AHP 的动态级任务调度算法。这两个调度算法均从云计算资源提供方的角度出发，有效提高任务完成的总数，增大服务方收益。

6.1 引言

市场模型（B. Ravi，2007；G. Stuer，2007；R. Buyya，2002）能够更有效地管理和评估云计算中的资源分配问题。这种模型有以下四个方面的好处：1，使用户公平竞争和使用计算资源；2，可以调节云计算中资源的供需平衡。当供不应求时（基本不会出现这种情况），通过提高资源价格来减少用户和任务数量；当供过于求时，可以降低资源价格，以吸引更多用户参与使用；3，给用户提供了服务质量（QoS）需求，比如任务的截止时间，完成任务的费用，服务资源的安全性等；4，提供了有效的资源管理和分配机制。我们将市场模型引入云计算中，研究批调度相关问题。

6.2 批调度的相关研究

任务调度依据调度时间的不同，可以分为批调度和在线调度两种模式。在批调度中，任务到达后并非立即得到服务，而是被收集成一个任务集合等到某个时间或者事件后一起调度到资源。Min-min（R. Armstrong，1998；R. Freund，1998；O. Ibarra，1977）是经典的批调度算法。它的算法思想是将最好的资源分配给计算长度最小的任务，这样提高了系统吞吐率，使得任务集合获得最小的完成时间。

随着市场模型的提出，许多研究开始关注用户服务质量的要求。文献（R. Buyya，2000）提出了在预算和任务截止时间限制下的时间优化算法（DBCT）。DBCT 算法主要思想是将任务调度到最早完成它的资源上，并且要求这个资源的执行费用不超过用户预算，完成时间不超过任务截止时间。虽然该算法在用户任务预算充足和截止时间长的应用环境下可以得到较短的任务完成时间，但是当用户任务预算相对较低或任务截止时间较短的情况下，算法完成的任务总数较少。文献（S. Kavitha，2004）比较了基于 QoS 的五种算法在用户满意度、任务完成

周期和元任务效用三个方面的性能，指出在不同的用户服务质量需求条件下，每种算法的性能效果是不同的，所以应该根据具体的应用环境，采用不同的任务调度算法。文献（K. Vanmechelen，2008）提出采用经济管理方法来解决基于截止时间的 CPU 绑定应用任务。文献（V. Sundaram，2008）讨论了在截止时间限制下任务调度吞吐量和公平性的关系，指出，如果增加调度吞吐量，就会给一些任务带来不公平调度；相反，如果注重公平性调度，则会降低系统吞吐量。该文通过设定一个简单参数来调整两者的平衡关系。

6.3 服务质量驱动的任务批调度算法

本小节借助云计算市场模型，针对某些用户的任务预算较少或者用户对任务的完成时间的要求较高，资源提供方不能完成用户的所有任务的情况，提出一种服务质量驱动下的任务调度算法。该算法结合了 Min-min 算法吞吐量较高和线性规划全局优化的优点，不仅考虑了用户的所有任务，同时还考虑了优先级较高的任务。实验结果表明，该算法在任务完成总数方面比两个经典算法 Min-min 算法和 DBCT 算法（预算和截止时间限制下的最大任务完成数调度算法）分别提高了约 10.6%和 22.0%，在优先级高的任务完成总数方面也有大幅度提高，分别约为 20%和 40%。

6.3.1 云计算调度模型

云计算调度模型主要由用户（Client），代理（Broker），资源（Resources），云计算资源提供者（Resources supporter）和信息服务器（Information Service）组成，其体系结构如图 6.1 所示（R. Buyya，2000）。

用户需要执行的任务通常可以分为串行应用、并行应用、参数扫描应用、协同应用等。系统允许用户设定资源需求和参数偏好；用户通过支付费用使用资源。

代理是用户和资源之间的中间接口，用于发现资源、选择资源、接收任务、返回调度结果、交换用户与资源的信息。代理支持不同的调度策略，能够根据用户的需求来发现资源和调度任务。代理主要由作业控制代理（Job Control Agent）、调度顾问（Schedule Advisor）、搜索（Explorer），交易管理器（Trade Manager）和部署代理（Deployment Agent）组成。

云计算环境中有多种计算资源，资源有计算性能和计算价格属性。计算性能用 MIPS（Million Instructions Per Second）表示，价格属性用单位时间 G$来表示。一般来说，计算资源的计算性能越好，则价格越高。

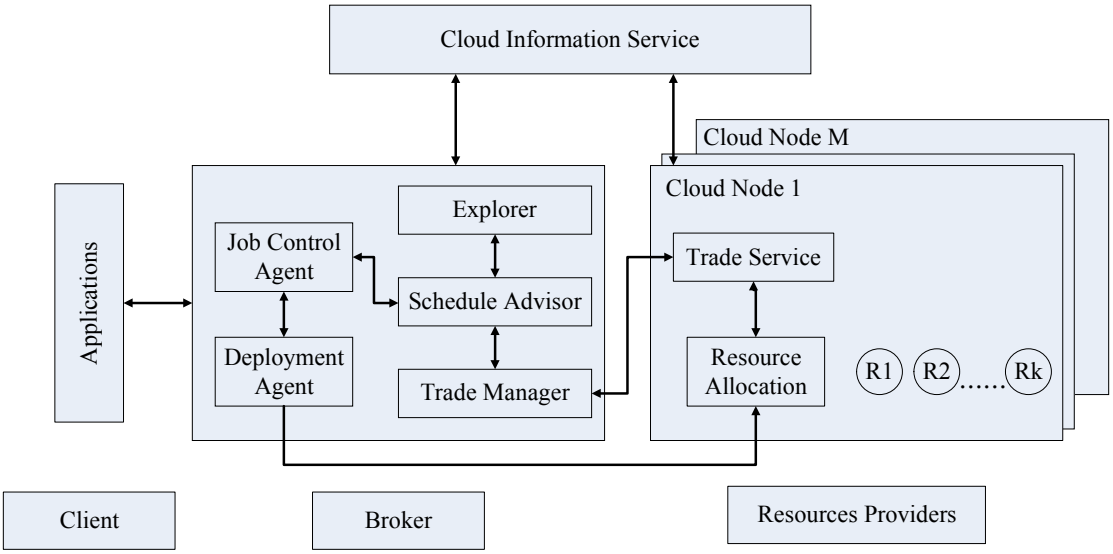


图 6.1　云计算调度模型

信息服务器主要记录可用资源信息。代理要寻找合适的资源，必须先在信息服务器中查询，获得适合用户条件的资源信息，然后才能和资源提供者进行交互。如果资源提供者有新资源需要租赁，也必须先向信息服务器注册，代理才能够找到这个资源。

在用户与服务提供者交易的过程中，服务提供者首先在信息服务器中注册资源信息。当用户把任务提交给代理后，代理在信息服务器中查找资源，然后根据相应的调度算法，将任务调度到合适资源。在任务被执行之前，代理会估计任务的完成时间和成本。如果时间超过截止时间或者计算成本高于用户预算，则代理拒绝接受该任务。如果任务执行成功，代理则返回给用户调度结果并获得相应收益；否则返回错误信息。

6.3.2 调度问题描述

在云计算环境下，服务提供者可以通过租赁出计算资源获得利益，而用户可以通过购买资源（实际上是资源的使用权）完成一些在自己的资源（或者根本没有自己的资源）上没有能力按服务质量需求完成的任务。我们假设任务预算和截止时间是有限的，用户希望总的服务费用在预算之内，同时任务完成时间在截止时间之前的前提下，计算资源能够尽可能多的完成任务。假设用户定义足够多的预算和足够长的截止时间，那么用户任务一定会在预算和截止时间限制内全部完成。

由于用户的预算较少或者任务本身较为紧急，可能造成云计算资源在预算和截止时间限制下无法全部完成任务。所以，我们主要讨论在上述情况下如何最大化任务完成数的问题。同时在所有的任务当中，我们还考虑了在特殊情况下，某些任务较为紧急重要，用户希望资源可以在较短截止时间之前确保完成这些任

务，而其余任务相对次重要，用户可以接受该类任务延迟完成。

现在，用户从所有任务中分出一类任务，该类任务优先级较高，务必在较短截止时间内完成，我们用 HS 来表示该类任务；剩余的用户任务优先级略低，可以延迟一段时间直到最后的截止时间之前完成，用 LS 表示该类任务。图 6.2 所示是某个用户定义的两类任务截止时间。从图 6.2 可以看到，HS 类任务必须在 $T = T_{HS}$ 之前完成，LS 类任务在 $T = T_{LS}$ 之前完成。我们将用户任务形式化表示为（HS，T_{HS}），（LS，T_{LS}）。如果任务类型为（HS，0），（LS，T_{LS}），说明用户没有定义 HS 类任务，所有任务相互对立，没有优先顺序，即所有任务只要在预算和截止时间 T_{LS} 限制下尽可能多的完成即可。

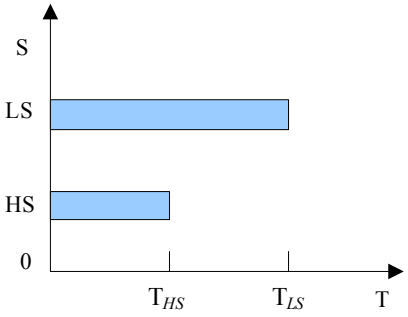


图 6.2 HS 和 LS 两类任务的截止时间

6.3.3 服务质量驱动下的任务调度算法

6.3.3.1 预算和截止时间限制下的最大任务完成数调度算法（DBCN）

假设在云计算环境中，有 m 个机器 $\{r_1, r_2, ..., r_m\}$，n 个任务 $\{e_1, e_2, ..., e_n\}$ 和一些如表 6.1 所示的变量。

表 6.1 参数定义

变 量	定　义
n	任务数目
m	计算资源数
L_i	第 i 个任务的长度
B	任务的预算
Re_j	第 j 个资源的计算性能
P_j	第 j 个资源的计算价格

根据用户的要求，HS 任务应该在 T_{HS} 之前尽可能完成，LS 任务则尽可能在 T_{LS} 之前完成。为了确保优先级高的 HS 任务可以按需求完成，在调度该类任务时，应该将好的资源分配给计算长度小的任务，这样可以在一段时间内得到较多的任务完成数。所以在 DBCN 算法中，根据 HS 和 LS 两类优先级不同的任务集合，

通过 Min-min 算法优先把好资源分配给 HS 任务。当执行完 HS 任务时，由于每个机器性能不同，同时分配的任务大小和数目也不一样，所以每个机器的完成时间和计算成本也会不同。若用 t_j 和 b_j 来表示第 j 个机器完成 HS 任务的时间和成本，用 b 来表示执行完所有 HS 任务所用的计算成本，则 $b=\sum_{j=1}^{m} b_j$ 。

但是，若有限的预算或者 HS 任务截止时间紧迫，无法全部完成 HS 类型的任务，则应该将该类型任务加入到 LS 任务集合中。在执行 LS 任务时，我们通过一个线性规划驱动下的 LPM 计算模块来得到 LS 任务与资源的映射关系。我们定义每个任务结构体中有一个属性 sort，当 sort=1 时，表示该任务为 HS 任务；当 sort=2 时，表示该任务为 LS 任务。

在调度过程中我们假设：

●任务在执行过程中是不可抢占的，一旦机器开始执行，就不能中断该任务转而执行其它任务，必须完成它之后再继续执行其余任务。

●对于相同类型任务，没有优先顺序，用户关心的是资源在截止时间之前是否完成全部的任务。

如上所述，我们可以得到预算和截止时间限制下的最大任务完成数调度算法（DBCN），如算法 6.1 所示。

算法 6.1：预算和截止时间限制下的最大任务完成数调度算法（DBCN）

1)　S ←所有任务的集合；

2)　S_{HS} ←sort=1 任务集合；S_{LS} ←sort=2 任务集合；

3)　初始化 T_{HS}，T_{LS}，t_j，b_j，B，b；

4)　if（$S_{HS}=\varnothing$）转到 6；else

5)　采用 Min-min 算法，预计 HS 任务完成情况，返回 b，t_j；

6)　$S_{LS}=S_{LS}\cup\{e_i \mid e_i$ 未执行 $\wedge\, e_i \in S_{HS}\}$；

7)　调用 LPM（S_{LS}，T_{LS}，t_j，B，b）模块，返回 LS 任务完成情况和任务与资源的映射关系；

8)　根据前面计算产生的任务与资源的映射关系，把任务调度到相应资源进行执行计算；

9)　接受云资源服务方的调度结果；

10)　结束；

6.3.3.2 LPM 模块

首先，我们定义矩阵 A_{nm} 如公式（6.1）所示，

$$A_{nm}=\begin{pmatrix} a_{11},a_{12},...,a_{1m} \\ a_{21},a_{22},...,a_{2m} \\ ... \\ a_{n1},a_{n2},...,a_{nm} \end{pmatrix} \tag{6.1}$$

$$\text{其中}a_{ij}=\begin{cases}1,\text{如果第i个任务调度到第j个资源；}\\ 0,\text{否则；}\end{cases}$$

- 因为每个任务最多在一个资源上执行，所以有公式（6.2），

$$\sum_{j=1}^{m} a_{ij} \le 1, i = 1,\ldots,n, e_i \in S_{LS} \tag{6.2}$$

- 当将要执行 LS 类型任务时，由于在执行 HS 类型的任务时，每个资源已经执行了一段时间t_j，同时所有资源也获得了服务成本 b，根据截止时间T_{LS}和剩下的资源可用预算 B-b，有公式（6.3）和（6.4），

$$\sum_{i=1}^{n} a_{ij} * L_i / \mathrm{Re}_j \le T_{LS} - t_j, j = 1,...,m, e_i \in S_{LS} \tag{6.3}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij} * L_i * P_j / \mathrm{Re}_j \le B - b, e_i \in S_{LS} \tag{6.4}$$

- 在执行 LS 任务时，调度目标就是最大化任务数，所以有公式（6.5），

$$\max = \sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij}, e_i \in S_{LS} \tag{6.5}$$

根据公式（6.2-6.5）可以建立一个如下的线性规划模型，我们采用 LPM 算法解决在算法 6.1 中的 LPM 模块，如算法 6.2 所示。

$$\max = \sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij}$$

$$s.t. \sum_{j=1}^{m} a_{ij} \le 1, i = 1,2,...,n$$

$$\sum_{i=1}^{n} a_{ij} * L_i / \mathrm{Re}_j \le T_{LS} - t_j, j = 1,...,m$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij} * L_i * P_j / \mathrm{Re}_j \le B - b$$

$$a_{ij} = 0 or 1, i = 1,2,...,n, j = 1,2,...,m$$

$$e_i \in S_{LS}$$

算法 6.2：LPM

1)　M←所有机器集合；

2)　初始化 A_{nm}，R_{e_i}，P_j，L_i，T_{LS}，t_j，B，b；

3)　建立和计算线性规划模型，并返回 A_{nm}；

4)　for(i=1; i<=n; i++)

5)　　for(j=1; j<=m;j ++)

{

6)　　5.1 if(a_{ij} = =1&& $e_i \in S_{LS}$) 第 i 个任务映射到第 j 个机器上;

7)　　5.2 else continue;

}

8)　返回资源的预计完成时间和映射结果；

9)　结束；

6.3.4 实验模拟和结果分析

为了验证我们提出的算法，本节与两个经典算法 Min-min 和 DBCT 算法进行比较：(1) 当总任务中没有 HS 任务集合时，对于不同任务预算和截止时间组合的应用条件下，算法完成的总任务数情况；(2) 当 HS 任务数占总任务数从 10%变化到 80%时，算法在 HS 任务完成率和总任务完成率两个方面的情况。其中 HS 任务完成率是指完成的 HS 任务数占 HS 任务总数的百分比，总任务完成率是指完成的任务数（包括完成的 HS 任务）占用户任务总数的百分比。

6.3.4.1 实验方法和结果

实验的模拟平台是 Cloudsim（R. Buyya，2009）。Cloudsim 是一个基于 Java 的事件驱动的云计算仿真工具包，它的主要目标是通过模拟云计算环境来研究基于计算经济模型的有效资源分配方法。设有 10 个计算节点，每个节点有 50 个计算资源，共有 500 个计算资源。每个节点里计算资源的计算速率和单位时间价格相同。所有节点中的计算资源性能（每秒钟执行指令数）和单位时间价格（相对价格）如表 6.2 所示。

表 6.2　计算资源每秒执行指令数和单位时间价格

节点号	0	1	2	3	4	5	6	7	8	9
指令数/s	10	12	14	16	18	20	22	24	26	28
价格	1	2	3	4	5	6	7	8	9	10

我们设计了两个模拟实验。第一个模拟实验设置如下：

- 1000 个任务，任务的大小随机分布在 600MI（Million Instruction）到

12600MI 之间。

- 用户任务预算和截止时间参数的组合分别为：第一组（450000，5000），第二组（700000，7000），第三组（850000，9000），第四组（1000000，10000），第五组（1150000，13000），第六组（1300000，16000），第七组（1450000，19000）。
- 用户没有定义 HS 类任务，HS 任务集合为空，这样 LS 任务集合有 1000 个任务。

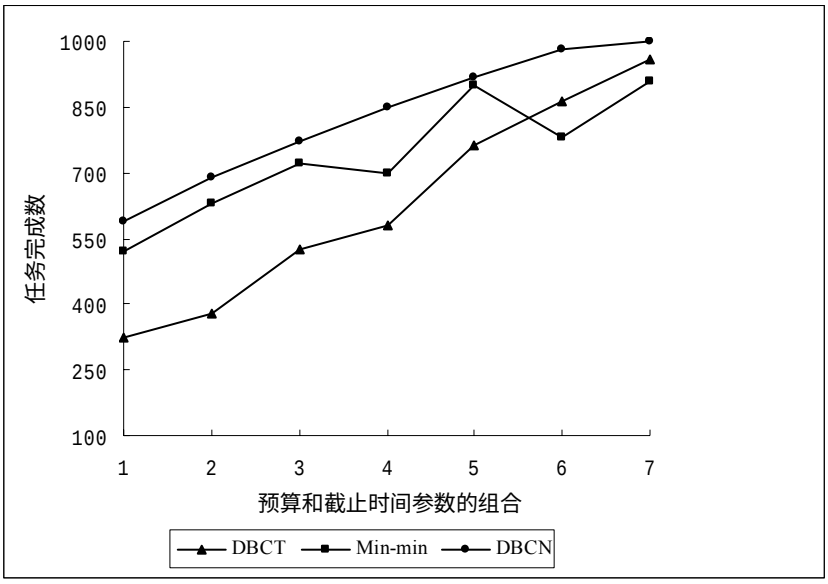


图 6.3 每组预算和截止时间对应的任务完成数

模拟实验的结果如图 6.3 所示。随着预算和截止时间增加，Min-min 算法的任务完成数总体是增加的，但不是绝对递增。从图 6.3 中可以看到，第 4 组比第 3 组完成的任务少，第 6 组比第 5 组完成的任务少，出现了预算和截止时间增加，任务完成数反而减少的“ 颠簸 ”现象。这主要是由于在无法全部完成任务的情况下，预算和截止时间的限制会对选择调度任务产生影响，以致整个调度完成后会产生任务完成数目减少的情况出现。比较看来，DBCN，DBCT 算法随着预算和截止时间的增加，完成的任务总数逐渐递增。对于每组预算和截止时间参数，DBCN 算法的性能要好于 DBCT 和 Min-min 两个算法，约有 22.0%和 10.6%的性能提高。

在第二个模拟实验中，HS 集合的任务数占总的任务数从 10%变化到 80%；总的任务数目随机从 500 到 1000 分布，通过模拟实验我们得到 HS 任务完成率的平均值和总任务完成率的平均值。从图 6.4 中可以看到，对于每个 HS 占有比例来看，DBCN 算法的 HS 任务完成率明显比其他两个算法要高，这主要因为在 DBCN 算法中，我们优先考虑了 HS 任务，而其他两个算法没有考虑 HS 任务的紧急性和重要性。

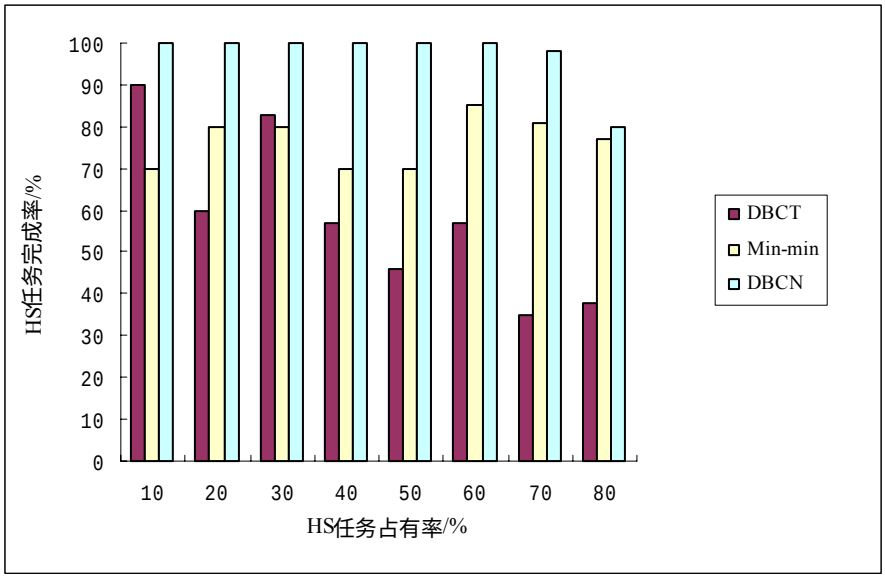


图 6.4 不同 HS 任务占有率所对应的 HS 任务完成率

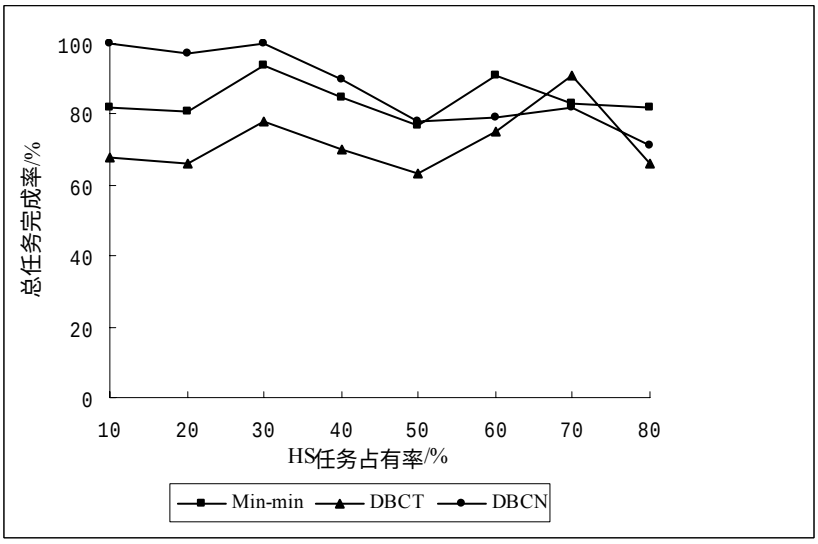


图 6.5 不同 HS 任务占有率所对应的总任务完成率

同时在图 6.5 中可以看到，随着 HS 任务占有率的提高，Min-min 和 DBCT 算法的总的任务完成率比较平稳，DBCN 算法的总的任务完成率在 HS 任务占有率为 50%左右之前要高于其他两个算法，在 50%之后算法性能不如 DBCT 和 Min-min 算法。这主要是因为 HS 任务占有较好的资源，随着 HS 任务占有率增加，影响了总的任务的完成情况。所以在实际的系统设计中，要限制 HS 任务数目，使得 HS 任务完成率和总的任务完成率性能都可以较好。

6.3.4.1 算法性能分析

在 LPM 算法中我们采用了分支定界法（A. Land，1960）来计算线性规划模型，整个算法的时间和空间开销也主要在求解线性规划模型上。从表 6.3 和 6.4

可以看到，当任务数或者资源数增加时，算法时间开销和空间开销也增加。

表 6.3 不同任务数对应的算法时间和空间开销

任务数目	50	100	500	1000
时间开销（ms）	10	10	20	40
空间开销（K）	101	192	922	1825

表 6.4 不同资源数对应的算法时间和空间开销

资源数目	50	100	300	500
时间开销（ms）	10	20	20	40
空间开销（K）	866	1704	5100	8421

从时间开销角度上看，算法基本上处于毫秒级别，远远小于任务的执行时间，所以通常算法调度的时间是可以忽略的。从空间开销角度上看，算法开销呈现出线性递增，所以根据机器硬件条件，我们可以限制算法应用的任务数或者资源数。

虽然线性规划模型的求解是个 NP 问题，但是它已经作为一种成熟的方法应用到其它调度算法和实际调度生产中。现在有大量研究成果可以使得串行算法时间复杂度控制在多项式级别上，同时随着并行技术的提高，线性规划的串行算法已经可以很容易并行化，这也大大减少了时间和空间开销。

6.4 基于 AHP 的动态级任务调度算法

动态级调度算法是一种快速有效的算法。在调度的每一步中，通过计算得到预备任务 N_i 和空闲机器 P_j 相匹配的动态级别最高的任务 N_i，将任务 N_i 调度到 P_j 上执行。我们充分考虑了云计算环境中机器资源的异构性，以及资源承担的风险，从资源服务费角度出发，提出基于 AHP 的动态级调度算法。该算法利用 1-9 尺度充分考虑了资源承担的风险，有效提高了服务方在时间底线内的任务完成率，以及单位时间内的收益。

6.4.1 调度算法中的要素

调度算法中涉及的要素很多，我们引用时间底线（deadline）、赔偿率和收益这三个要素，体现最简单最直接的用户需求和服务费收益。Deadline 是指任务调度的时间底线。每一个任务都有自己的 deadline，服务方的调度目的就是在任务的 deadline 之前尽可能完成任务，并且尽可能的最大化完成任务所带来的收益。为了充分考虑资源节点承担的风险，我们还引入赔偿率和收益这两个要素，来体现节点可能承担的赔偿以及可能获得的利润。

6.4.2 AHP 及 1-9 尺度

AHP（The Analytic Hierarchy Process）（TL. Saaty，2008）又称为层次分析法。它是一种通过参考设定的优先级尺度（又称相对尺度），两两成对的比较已有因素来获得结果的测量方法。我们利用 AHP 方法，从用户的角度出发，对用户的任务要素进行计算，即得到体现用户需求的权值参数。此外，由于参数中包含了赔偿率与收益的信息，即隐形的资源节点承担风险的信息，我们可以更可靠有效的进行资源的调度，具体的调度算法见 6.4.5 部分。

对于任务要素之间的比较问题，已经提出的 1-9 尺度（TL. Saaty，2008）可以解决该问题。1-9 尺度说明如表 6.5 所示。

表 6.5 1-9 尺度(TL. Saaty, 2008)

Intensity of Importance	Definition	Explanation
1	Equal Importance	Two activities contribute equally to the objective
2	Weak or slight	
3	Moderate importance	Experience and judgement slightly favor one activity over another
4	Moderate plus	
5	Strong importance	Experience and judgement strongly favor one activity over another
6	Strong plus	
7	Very strong or demonstrated importance	An activity is favored very strongly over another; its dominance demonstrated in practice
8	Very, very strong	
9	Extreme importance	The evidence favoring one activity over another is of the highest possible order of affirmation
Reciprocals of above	If activity i has one of the above non-zero numbers assigned to it when compared with activity j, then j has the reciprocal value when compared with i	A reasonable assumption
1.1-1.9	If the activities are very close	May be difficult to assign the best value but when compared with other contrasting activities the size of the small numbers would not be too noticeable, yet they can still indicate the relative importance of the activities.

6.4.3 风险权值的计算

以上我们已经简单介绍了 AHP 以及 1-9 尺度，现在我们来详细说明如何利用 AHP 方法计算得到任务的权值。

为了计算得到任务的相关权值，结合任务调度问题，我们设置如下（如图 6.6 所示）：目标层为 Goal，即最佳的调度任务；准则层包含三个要素，分别是：时间底线 Deadline，赔偿率 Reparation Duty，以及收益 Profit。我们用时间底线体现了用户需求，赔偿率及收益包含了资源节点承担的风险；方案层为所有的资

源。综上所述，我们构成了层次结构图，如图 6.6 所示。

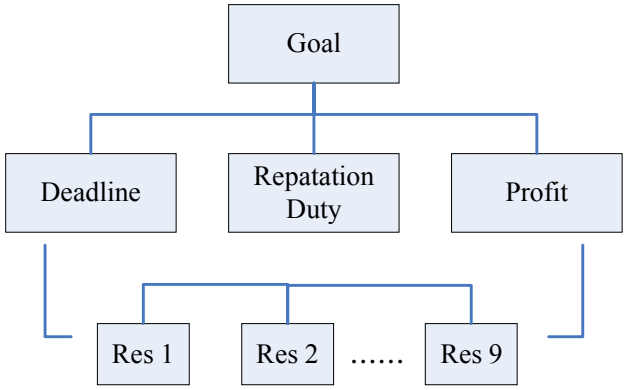


图 6.6 层次结构图

根据该层次结构图，我们计算任务的权值。例如，对于一个给定任务，按照 Saaty 的 1-9 尺度等级，设置为时间底线：赔偿率：收益为 7：9：5。即赔偿率最为重要，其次是时间底线，最后为收益，则我们构建该任务的成对比较阵如公式（6.6）所示：

$$T=\begin{pmatrix} 1 & 3 & 3 \\ 1/3 & 1 & 1 \\ 1/3 & 1 & 1 \end{pmatrix} \tag{6.6}$$

计算得到该矩阵的特征向量为：(0.75, 0.125, 0.125)，即为 deadline，赔偿率以及 Profit 相对于任务分布的各个权值。

表 6.6 与目标相关联的成对比较阵

	Deadline	Reparation Duty	Profit	Priorities
Deadline	1	3	3	0.75
Reparation Duty	1/3	1	1	0.125
Profit	1/3	1	1	0.125

若只有 5 个资源节点，且在提交该任务的用户看来，当时间底线为参照物时这些资源节点的权重比如公式（6.7）所示；赔偿率为参照物时这些资源的权重比如公式（6.8）所示；而当收益为参照物时资源节点的权重比如公式（6.9）所示。这些资源构成的成对比较阵如下：

$$R_D=\begin{pmatrix} 1 & 1/2 & 2 & 3 & 3 \\ 2 & 1 & 4 & 5 & 5 \\ 1/2 & 1/5 & 1 & 1 & 1 \\ 1/3 & 1/5 & 1 & 1 & 1 \\ 1/3 & 1/5 & 1 & 1 & 1 \end{pmatrix} \tag{6.7}$$

计算得到该矩阵的特征向量为：(0.222, 0.611, 0.055, 0.055, 0.055)。

表 6.7 与 Deadline 相关联的成对比较阵

	Resource 1	Resource 2	Resource 3	Resource 4	Resource 5	Priorities
Resource 1	1	1/2	2	3	3	0.222
Resource 2	2	1	4	5	5	0.611
Resource 3	1/2	1/5	1	1	1	0.055
Resource 4	1/3	1/5	1	1	1	0.055
Resource 5	1/3	1/5	1	1	1	0.055

$$R_R=\begin{pmatrix} 1 & 1/3 & 3 & 2 & 2 \\ 3 & 1 & 8 & 5 & 5 \\ 1/3 & 1/9 & 1 & 1 & 1 \\ 1/2 & 1/5 & 1 & 1 & 1 \\ 1/2 & 1/5 & 1 & 1 & 1 \end{pmatrix} \tag{6.8}$$

计算得到该矩阵的特征向量为：(0.169, 0.687, 0.048, 0.048, 0.048)。

表 6.8 与 Reparation Duty 相关联的成对比较阵

	Resource 1	Resource 2	Resource 3	Resource 4	Resource 5	Priorities
Resource 1	1	1/3	3	2	2	0.169
Resource 2	3	1	8	5	5	0.687
Resource 3	1/3	1/9	1	1	1	0.048
Resource 4	1/2	1/5	1	1	1	0. 048
Resource 5	1/2	1/5	1	1	1	0. 048

$$R_P=\begin{pmatrix} 1 & 1/3 & 3 & 2 & 3 \\ 3 & 1 & 8 & 5 & 5 \\ 1/3 & 1/9 & 1 & 1 & 1 \\ 1/2 & 1/5 & 1 & 1 & 1 \\ 1/3 & 1/5 & 1 & 1 & 1 \end{pmatrix} \tag{6.9}$$

计算得到该矩阵的特征向量为：(0.182, 0.682, 0.045, 0.045, 0.045)。

表 6.9 与 Profit 相关联的成对比较阵

	Resource 1	Resource 2	Resource 3	Resource 4	Resource 5	Priorities
Resource 1	1	1/3	3	2	2	0.182
Resource 2	3	1	8	5	5	0.682
Resource 3	1/3	1/9	1	1	1	0.045
Resource 4	1/2	1/5	1	1	1	0.045
Resource 5	1/2	1/5	1	1	1	0.045

综合上述 T 和 R 的矩阵，我们可以获得最终结果如表 6.10 所示。

由表 6.10，我们获知，对于该任务，5 个资源节点承担的风险权值分别为：(0.210, 0.629, 0.053, 0.053, 0.053)。

表 6.10 综合 T 和 R 获得的最终结果

	Deadline	Reparation Duty	Profit	Priorities
Resource 1	0.75	0.125	0.125	0.210
Resource 2	0.75	0.125	0.125	0.629
Resource 3	0.75	0.125	0.125	0.053
Resource 4	0.75	0.125	0.125	0.053
Resource 5	0.75	0.125	0.125	0.053

6.4.4 一致性检查

如果一个成对比较阵 A 满足公式（6.10），则称 A 为一致性矩阵。

$$a_{ij} * a_{jk} = a_{ik}\text{，i，j，k=1，2……n} \tag{6.10}$$

由于成对比较阵是以时间底线，赔偿率以及收益为参照物，从我们主观角度出发评价的各个资源的等级，所以成对比较阵可能不是一致阵，比如公式（6.7）。当这种不一致在一定的允许范围内时，我们也接受这样的不一致。

定理 6.1　n 阶成对比较阵 A的最大特征根$\lambda \geq$ n，而当λ=n时，A是一致阵。（TL. Saaty，2008）.

证明：详见参考文献（TL. Saaty，2008）。

由定理 6.1 可知，λ与 n 的值差别越大，A 的不一致性程度越严重，用特征向量作为权向量引起的判断误差越大。因此，可以用$\lambda - n$的大小来衡量 A 的不一致程度。文献（TL. Saaty，2008）定义一致性指标如公式（6.11）所示。CI=0 时，A 为一致阵；CI 越大，则 A 的不一致程度越严重。

$$CI = \frac{\lambda - n}{n - 1} \tag{6.11}$$

文献（TL. Saaty，2008）引入随机一致性指标 RI，来找出一致性指标 CI 的标准。文献（TL. Saaty，2008）对于不同的 n（n 为 1-11 的整数），用 100500 个样本算出的随机一致性指标 RI 数值如表 6.11 所示。

表 6.11 随机一致性指标 RI 的数值

n	1	2	3	4	5	6	7	8	9	10	11
RI	0	0	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49	1.51

表 6.11 中$n = 1,2$时，$RI = 0$是因为 1，2 阶的成对比较阵总是一致阵。

对于$n \geq 3$的成对比较阵 A，文献（TL. Saaty，2008）规定一致性比率如公式（6.12）所示。

$$CR = \frac{CI}{RI} \tag{6.12}$$

若$CR < 0.1$，则认为成对比较阵 A 的不一致程度在允许的范围之内。

对于任意一个成对比较阵 A，若 A 是一致阵，则我们接受由成对比较阵计算得到的权值。然而，若 A 不是一致阵，则要对 A 进行上述的一致性检查，仅当 $CR < 0.1$ 时，我们认为可以接受 A 的权值；若 $CR \geq 0.1$，我们认为 A 不一致，需要重新调整成对比较阵的元素值，直到 A 通过一致性检查。

6.4.5 基于 AHP 的动态级调度算法

动态级调度算法是一种快速有效的算法。在调度的每一步中，通过计算得到预备任务 N_i 和空闲机器 P_j 相匹配的动态级别最高的任务 N_i，将任务 N_i 调度到 P_j 上执行。

我们充分考虑了云计算环境中机器资源的异构性，以及资源承担的风险。当任务调度到目标节点机器上执行时，信任度就反映了目标节点承担的风险，即可能提供赔偿或者获得收益的程度。我们将基于 AHP 的动态级调度算法公式定义如公式（6.13）所示：

$$\mathrm{DL}(\mathrm{N_i},\mathrm{P_j},\sum) = SL^*(N_i) - \max[DA(N_i,P_j,\sum),TF(P_j,\sum)] + n*w(N_i,P_j)*\Delta(N_i,P_j) \quad (6.13)$$

其中，$SL^*(N_i)$ 是指任务 N_i 的动态优先级。$\max[DA(N_i,P_j,\sum),TF(P_j,\sum)]$ 表示任务 N_i 在资源节点 P_j 上可以开始执行的最早时间。$\Delta(N_i,P_j)$ 反映了不同机器资源的处理速度。$\Delta(N_i,P_j)$ 为正值时，表示机器资源 P_j 执行任务 N_i 较之其他机器资源的速度都要快，数值越大，则相对速度越快；相反，当 $\Delta(N_i,P_j)$ 为负值时，则表示机器资源 P_j 执行任务 N_i 较之其他机器资源的速度都要慢，绝对值越大，则相对速度越慢。$w(N_i,P_j)$ 为第一节中利用 AHP 方法计算得到的资源节点 P_j 相对于任务 N_i 的权重值，即 P_j 承担的风险（或者信任度）。n 为资源节点数。我们设置 n，是因为利用 AHP 方法计算得到的权值可能较小，不能充分体现资源节点承担的风险，故利用节点数目，将资源节点承担风险的权重因子合理的放大。

6.4.6 实验模拟与结果分析

6.4.6.1 实验参数设置

我们设置资源节点如表 6.12 所示。处理能力表示资源节点处理任务的能力权值，单位价格指资源节点运行任务时单位收费权值，偏差表示资源节点运行任务时承担的风险比例，比如，可能速度有所降低，不能按预期完成任务等等。Percentage 表示各个资源的分布比例。

表 6.12 资源节点的各个状态及分布比例

处理能力	单位价格	偏差	Percentage
0-3	1-2	25%-30%	50%
3-6	4-5	15%-20%	30%
6-9	8-9	5%-10%	20%

我们设置七种类型的任务。对于任务的时间底线，赔偿率以及收益的权值，我们设置如表 6.13 所示，Percentage 为各个任务在总的调度任务中所占比例。

表 6.13 各个任务的权值及分布比例

	Deadline	Reparation Duty	Profit	Percentage
Task 1	0.6	0.2	0.2	15%
Task 2	0.45	0.45	0.1	10%
Task 3	0.45	0.1	0.45	10%
Task 4	0.2	0.6	0.2	15%
Task 5	0.2	0.2	0.6	15%
Task 6	0.1	0.45	0.45	10%
Task 7	0.33	0.33	0.33	25%

当用户任务需要执行时，任务的基本信息采用 5.3 中定义。

- deadline－任务执行时间：任务在该时间之前完成时，服务方不需支付赔偿金；
- reward－服务付费：任务在 deadline 之前完成时，用户向服务方支付的费用；
- decay－任务超时赔偿率：当服务方未在 deadline 完成任务时，每超过一个单位时间，服务方需支付的赔偿金；
- bottomline－任务最高赔偿金的时间底线：当服务方未在 deadline 完成任务，到达 bottomline 时刻时，赔偿金不再增加，其中 bottomline>deadline；
- Penalty－任务超时最高赔偿金：服务方执行任务的时间达到 bottomline 时，所支付的赔偿金。此时，任务可能完成了一部分，也可能完全未执行。

此时，以资源节点 1 和 2 来举例说明如下，如图 6.7 所示。

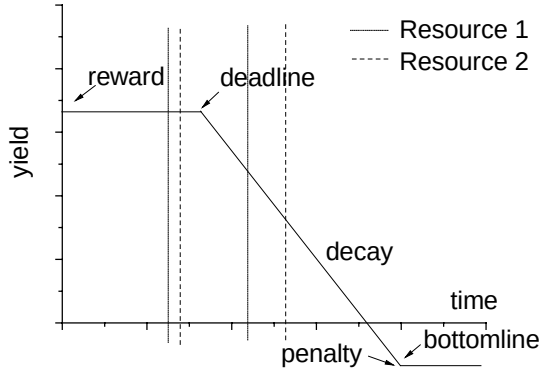


图 6.7 任务的状态图

资源节点 1 存在的偏差较小，在所示短虚线内可以完成任务。资源节点 2 的

偏差较大，在图示点线内可以完成该任务。则用户可以根据这两个资源节点的基本信息（如表 6.12 中所示），根据第一节中介绍的 AHP 方法，分别计算得到这两个资源节点对于时间底线，赔偿率以及收益的权重值（如表 6.7-6.9）。再结合任务自身权重值（如表 6.6），即可以计算出这两个资源节点最终的风险值（如表 6.10 所示）。

6.4.6.2 实验结果与分析

我们将基于 AHP 的动态级调度算法与动态级调度算法进行比较，实验模拟结果如图 6.8，图 6.9 所示。

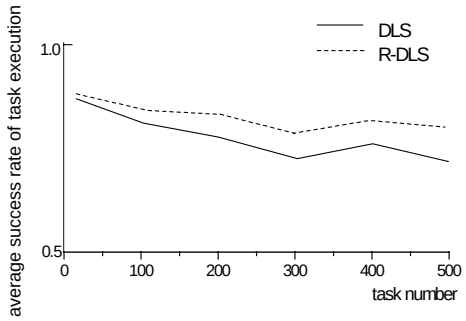


图 6.8 任务平均成功率变化情况

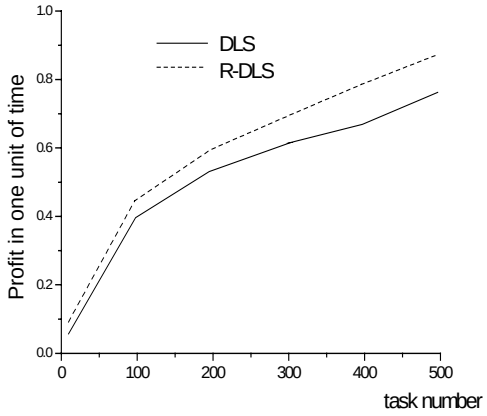


图 6.9 单位时间内的收益变化情况

由图 6.8 可以看出，随着任务数量的增加，基于 AHP 的动态级调度算法执行任务的成功率比动态级调度算法执行任务的成功率略高。这是因为我们利用 1-9 尺度计算了机器资源可能承担的风险程度，有效减少了超出时间底线完成的任务数量。由图 6.9 可以看出，由于基于 AHP 的动态级调度算法计算了资源节点承担的风险 ，因此资源节点承担的风险能够得到有效降低，单位时间内收益比动态级调度算法中资源节点的单位收益要高。

6.5 本章小结

本章充分考虑优先级、时间调度底线、收益、资源风险等调度因素，提出了服务质量驱动下的任务调度算法和基于 AHP 的动态级任务调度算法。这两个调度算法均从云计算资源提供方的角度出发，有效提高了任务完成的总数，增大了服务方收益。

第7章　总结与展望

自云计算出现以来，经过科学技术的不断发展，经过学术界与产业界的不断推进，云计算中的应用正不断发展不断深入，云计算也正在从理论走向实践。在云计算的研究和应用过程中，出现了一些重要的热点问题，都在被广泛的讨论和研究。其中，数据中心网络结构的扩展性与绿色节能问题、副本策略问题、以及调度机制研究都是云计算环境中需要考虑的三个重要问题。

7.1 主要工作总结

本文围绕云计算环境中的资源规模庞大，异构性强等特性产生的三个重要问题：数据中心网络结构的扩展性与绿色节能问题、副本策略问题、以及调度机制，开展了以下研究工作：

本文首先介绍了云计算的概念以及云计算数据中心的概念，然后针对传统数据中心的不足，着重介绍了云计算数据中心具备的特点；详细分析了由于云计算环境中的资源规模庞大，异构性强等特性产生的三个重要问题：数据中心网络结构的扩展性与绿色节能问题、副本策略问题、以及调度机制研究，引出本文的研究内容，给出本文的组织结构和贡献。

其次，针对云计算数据中心结构上的可扩展性及绿色节能需求，充分考虑了新型数据中心应具备的新特点，借鉴已有知名数据中心结构，依据著名科赫曲线，提出了新型数据中心网络结构（雪花结构）。该结构充分考虑了数据中心的可扩展性，在保证交换机与服务器较低数量比例（0.125-0.333范围内）的前提下，可以在较短的平均路径内实现节点间路由机制，具有较小的网络开销。

接着，针对云计算环境中的副本问题，提出了基于副本的调度模型；引入市场机制中的代价因素，通过综合考虑副本地理特性、网络状态、应用服务特点等因素，提出了一种代价驱动的自适应副本策略，该策略针对不同应用的一致性与可用性的重要程度，以代价为驱动力来自适应地进行副本复制、副本销毁、副本迁移等操作，以达到负载均衡等目的；提出了适用于该模型的，基于副本和数据中心网络结构的调度策略。

然后，为增大服务方的收益，提出云计算环境中基于成本计算的在线任务调度策略。首先详细阐述了服务方接受首个任务的调度策略，引入沉没成本，接着分析了在计算沉没成本和机会成本的前提下，服务方接受非首个任务的调度策略，最后比较了沉没成本与机会成本的关系，分析了沉没成本和机会成本对服务方收益的影响。模拟实验结果表明，提出的服务方收益最大化机制，有效降低了

服务方接受任务时承担的成本，促进调度环境的和谐发展。

最后，充分考虑优先级、时间调度底线、收益、资源风险等调度因素，提出了服务质量驱动下的任务调度算法和基于 AHP 的动态级任务调度算法。这两个调度算法均从云计算资源提供方的角度出发，有效提高了任务完成的总数，增大了服务方收益。

7.2 主要贡献与创新点

本文的主要贡献与创新点如下：

- 提出了一种雪花结构的新型云计算数据中心网络结构，实现了高扩展性和绿色节能、降低成本的目标。随着节点失效率的增加，雪花结构的路径失效率与 DCell 相比，变化不大；随着链路失效率的增加，DCell 的路径失效率达到了 30%，雪花结构几乎没有路径失效率。

在设计云计算数据中心网络结构时，扩展性是首要考虑的设计因素。高扩展性可以保证数据中心的资源利用率及添加新资源的便捷性。此外，绿色节能也是当前设计云计算数据中心的重要考虑因素，以降低二氧化碳排放量，减少能源损耗。本文提出的雪花结构云计算数据中心，在设计结构时充分考虑了资源的模块化扩展，并针对数据中心海量服务器的高扩展性，有针对性的尽可能减少交换机的使用，在保证高扩展性的前提下同时减少交换机的能耗和成本开销。

主要的创新点在于：借鉴科赫曲线，设计了雪花结构；数学证明及实验模拟证实，雪花结构具备高扩展性和绿色节能的特性，在较短的平均路径内实现节点间路由机制，具有较小的网络开销。

- 提出了代价驱动的自适应副本策略，实现了副本的动态复制、删除和迁移操作。模拟实验表明，与传统副本策略相比，当副本数量增大到 200 以上时，提出的副本策略将节点负载均衡性能提高约 30%。此外，与静态传统策略相比，不仅提高了平均副本收益，而且还可以扭亏为盈。

分布式系统中的CAP特性，即一致性、可用性、分区容错性三者在系统实现中无法兼顾。在网络分区给定情况下不能同时满足一致性和可用性的要求，这就带来了一致性与可用性的权衡问题。在云存储服务中，应用类型与用户群体的多样性带来了一致性与可用性需求的多样性。如何在一致性和可用性不可同时达到的前提下，权衡两者使得用户得到更好的体验。本文提出的代价驱动的自适应副本策略，借鉴分布式系统中的一些已有成果，以价格为出发点，充分考虑了副本的复制、删除和迁移操作及带来的开销。

主要的创新点在于：将市场机制中的价格因素引入副本策略中，综合考虑负载均衡及一致性与可用性的平衡，对副本进行自适应的操作，达到最小化副本开

销，最大化副本收益的目标。

- 提出了基于副本的调度模型及基于副本和数据中心网络结构的调度策略，区分未考虑副本情况的调度模型，避免随机调度或一般调度策略中未考虑副本阶数和位置的情况。与随机调度算法相比，该调度策略复制副本的平均最短路径大约减少 50%，可以有效节约副本在传输复制过程中的时间开销。

云计算环境中副本问题是核心问题，引入副本策略弥补了存储对象单点失效、容错性差、接入性能不高等问题。针对现有调度模型鲜有兼顾副本因素，提出了基于副本的调度模型。该模型中，以SLA控制器控制管理实现用户需求，以资源信息控制器管理空闲资源，以副本信息控制器管理副本信息，由Cloud控制器负责综合资源信息控制器和副本信息控制器返回的信息。在该模型的基础上，综合考虑副本和数据中心网络结构两个因素，引入改进的K-means算法，依据副本信息进一步确定聚类，得到相应的调度策略。

主要的创新点在于：调度模型中综合考虑了副本因素；在该模型的基础上，引入改进的K-means算法，兼顾副本信息确定集群聚类，获得基于副本和数据中心网络结构的调度算法。

- 提出了基于成本计算的在线调度策略，服务质量驱动的批调度策略和基于 AHP 的批调度策略。三种调度策略均从云计算服务提供方出发。模拟实验结果表明，提出的服务方收益最大化机制，有效降低了服务方接受任务时承担的成本，提高了任务完成总数，增大了服务方收益。

针对云计算环境，引入经济学中沉没成本的概念，分析了沉没成本和机会成本对服务方收益的影响。其次，针对任务优先级、时间调度底线、云服务提供方收益、资源风险等调度因素，引入层次分析法和1-9尺度，分别分析了服务质量和风险对调度的影响。模拟实验证明，三种调度策略不仅优化了策略目标，而且使得服务提供方和请求方都实现了自身的经济目标，促使交易环境向健康稳定的方向发展。

主要的创新点在于：将经济学中的沉没成本概念引入云计算环境，提出了基于成本计算的在线调度策略；兼顾调度中优先级、风险等众多因素，引入层次分析法和1-9尺度，提出了服务质量驱动的批调度策略和基于AHP的批调度策略。

7.3 进一步的工作

本文紧紧围绕云计算环境中的资源规模庞大，异构性强等特性产生的三个重要问题：数据中心网络结构的扩展性与绿色节能问题、副本策略问题、以及调度机制，进行了深入的研究，提出了雪花结构的新型云计算数据中心网络结构，基

于副本的调度策略，以及相关调度机制。基于本文的研究内容，还有如下一些问题，值得进一步研究：

- **完善资源定价机制**

云计算环境中，用户依据资源价格选择符合自身QoS需求的集群资源，因此资源价格是影响用户QoS需求及任务调度分配的重要因素，如何为云计算环境中的海量资源定价，如何利用价格机制促进海量资源的合理分配，更好更优更加高效的完成任务调度是值得深入研究的问题。

- **设计基于虚拟机动态迁移的调度策略**

云计算数据中心内，广泛采用虚拟化技术将物理资源集中在一起形成一个共享虚拟资源池，从而更加灵活和低成本的使用资源。通过服务器虚拟化、存储虚拟化、数据中心虚拟化等解决方案，不仅可以可降低服务器数量，还可以优化资源利用率。虚拟化是新一代数据中心中使用最为广泛的技术，也是与传统数据中心的最大差异，如何基于虚拟机的动态迁移，设计高效的任务调度策略是我们需要进一步研究的问题。

- **解决可以进一步分解的复杂任务调度问题**

本文研究的调度问题针对的是元任务调度，并未考虑可以进一步分解的复杂任务或无环有向图任务等。这些任务的调度完成需要考虑先后顺序的约束条件，以及网络通信开销和工序等待问题，比元任务调度复杂的多。如何解决这些复杂任务的调度需要进一步的研究。

参考文献

段明秀. 2009. 层次聚类算法的研究及应用[硕士学位论文]. 长沙：中南大学.

范明，孟小峰等. 2004. 数据挖掘概念与技术[M]. 北京：机械工业出版社.

牛宪龙，陈华平，周文煜，武斌，刘晓茜. 2010. 新型数据中心网络结构研究进展综述[J]. CNGrid 2010.

陆云. 2007. 聚类分析数据挖掘方法的研究与应用[硕士学位论文]. 安徽：安徽大学.

李东琦. 2007. 聚类算法的研究[硕士学位论文]. 成都：西南交通大学.

梁循. 2003. 数据挖掘算法与应用. 北京：北京大学出版社.

平新乔. 2001. 平新乔讲义系列：微观经济学十八讲[M]. 北京：北京大学出版社.

钱线，黄萱菁，吴立德. 2007. 初始化 K-means 的谱方法[J]. 自动化学报.

Sun 公司. 2009. 云计算架构介绍白皮书[M].

王德敏. 2004. 财务管理（第 1 版）[M]. 中国言实出版社.

维基百科：科赫曲线. 2011. http://zh.wikipedia.org/zh-cn/.

云存储内部结构深度解析. 2010. http://storage.chinabyte.com/337/11452837_3.shtml

于翔. 2007. 聚类分析中 k-均值方法的研究：[硕士学位论文]. 黑龙江：哈尔滨工程大学.

朱伟雄，王德安，蔡建华. 2009. 新一代数据中心建设理论与实践[M]. 人民邮电出版社出版.

A. Downey and D. Feitelson. 1999. The elusive goal of work-load characterization. In Performance Evaluation Review, pages 14–29.

Albert Greenber, Parantap Lahiri, David A. Maltz, Parveen Patel, Sudipta Sengupta. 2008. Towards a next generation data architecture: scalability and commoditization. Proceedings of the ACM workshop on programmable routers for extensible services of tomorrow. 57-62, Seattle, WA, USA.

Albert Greenberg. 2008. VL2: a scalable and flexible data center network. 51-62, SIGCOMM.

A. Land and A. Doig. 1960. An automatic method of solving discrete programming problems, Econometrica. pp. 497-520.

A.Greenberg, J.Hamilton, D.A.Maltz, P.Patel. 2009. Cost of Cloud. ACM SIGCOMM Computer Communication Review, Volume 39, Number1.

Andrew D. Birrell, Roy Levin, Roger M. Needham, and Michael D. Schroeder. 1982. Grapevine: An Exercise in Distributed Computing. Communications of the ACM, 25(4):260-274.

Armbrust M, FOX A, GRIFFITH R, et al. 2009. Above the Clouds: A Berkeley View of Cloud Computing [R]. Berkeley, CA,USA: Distributed Systems Lab, University of California.

Beckmann N., Kriegel H.. 1990. The R* tree: An efficient and robust access method for points and

rectangles.

B. Chun. Market-based Cluster Resource Managemen. 2001. PhD thesis, University of California at Berkeley.

B. Li, M. Golin, G. Italiano, and A. K. Sohraby. 1999. On The Optimal Placement of Web Proxies in the Internet. In Proceedings of IEEE INFOCOM'99.

B. N. Chun and D. E. Culler. 2002. User-centric performance analysis of market-based cluster batch schedulers. In 2nd IEEE International Symposiumon Cluster Computingand the Grid.

B. Ravi, D. Sanjukta, G. Robert, and J. Stallaert. 2007. A market design for grid computing. INFORMS Journal on Computing, Forthcoming, 20(1).

Charles E. Leiserson. 1985. Fat-trees: universal neworks for hardware-efficient supercomputing. IEEE Transactions on Computers, 892-901.

Chuanxiong Guo. 2008. DCell: a scalable and fault-tolerant network structure for data centers. 75-86, SIGCOMM.

Chuanxiong Guo. 2009. BCube: a high performance, server-centric network architecture for modular data center. 63-74, SIGCOMM.

Chen CT, Hsu CC, Wu JJ, et al. 2009. GFS: A Distributed File System with Multi-source Data Access and Replication for Grid Computing, 4th International Conference on Grid and Pervasive Computing, Geneva.

Cheng Chiyu, Sourav Bhattacharya. 1997. Dynamic Scheduling of Real-Time Messages over an Optical Network, Computer Science and Engineering Department Arizona State University, In 6th IEEE International Conference Computer Communications and Networks.

DeCandia G, Hastorun D, Jampani M, Kakulapati G, Lakshman A, Pilchin A, Sivasubramanian S, Vosshall P, Vogels W. 2007. Dynamo: Amazon's highly available key-value store. In Proceedings of the 21st ACM Symposium on Operating Systems Principles.

DavidE.Irwin, Laura E.Grit, and JeffreyS.Chase. 2004. Balancing Risk and Reward in a Market-based Task Service，Departmentof Computer Science Duke University，Box 90129,Durham,NC 27708,U.S.A.

Dogan A, Ozguner F. 2002. Matching and scheduling algorithms for minimizing execution time and failure probability of applications in heterogeneous computing. IEEE Transactions on Parallel and Distributed Systems, 13(3): 308-323.

Erickson J S, SPENCE S, RHODES M, et al. 2009. Content-Centered Collaboration Spaces in the Cloud [J]. IEEE Internet Computing, 13(5):34-42.

E. Pinheiro, W.-D.Weber, and L.A. Barroso. 2007. Failure trends in a large disk drive population. In Proc. of 5th USENIX Conference on File and Storage Technologies, SanJose, CA, USA.

Foster I, ZHAO Yong, RAICU I, et al. 2008. Cloud Computing and Grid Computing 360-degree Compared [C]//Proceedings of the IEEE Grid Computing Environments Workshop(GCE ’08), Nov 12-16, 2008, Austin, TX,USA. Piscataway, NJ,USA.

Germain R C, RANA O F. 2009. The Convergence of Clouds, Grids, and Autonomics [J]. IEEE Internet Computing, 13(6):9.

Gilbert S, Lynch N. 2000. Brewer‘s Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services, ACM SIGACT News 33(2).

G. Karjoth. 2003. Access control with IBM Tivoli access manager. ACM Transactions on Information and System Security.

G. Stuer, K. Vanmechelen and J. Broeckhove. 2007. A commodity market algorithm for pricing substitutable grid resources, Future Generation Computer Systems, 23(5):688–701.

Haifeng Yu, Amin Vahdat. 2002. Minimal Replication Cost for Availability, PODC, Monterey, California, USA.

Higgs R., Bemis K., Watson I. 1997. Experimental designs for selecting molecules from large chemical databases[J]. Journal of Chemical Information and Computer Sciences, 37(5): 861-870.

H.Shan, L.Oliker, and R.Biswas. 2003. Job superscheduler architecture and performance in computational grid environments. In Proceedings of Supercomputing.

I. Foster and C. Kesselman. 1997. Globus: A Metacomputing Infrastructure Toolkit, International Journal of Supercomputer Applications, 11(2): 115-128.

I. Foster and C. Kesselman. 1997. Globus: A Metacomputing Infrastructure Toolkit, International Journal of Supercomputer Applications, 11(2): 115-128.

Jensen M, SCHWENK J, GRUSCHKA N, et al. 2009. On Technical Security Issues in Cloud Computing [C]//Proceedings of the 2009 IEEE International Conference on Cloud Computing(CLOUD ’09), Sep 21-25, 2009, Bangalore, India. Los Alamitos, CA,USA: IEEE Computer Society, 109-116.

J. Naous, G. Gibb, S. Bolouki, and N. McKeown. 2008. NetFPGA: Reusable Router Architecture for Experimental Research.

J. Hamilton. 2009. Cooperative Expandable Micro Slice Servers (CEMS). In 4th CIDR

J. Lockwood, N. McKeown, G. Watson, G. Gibb, P. Hartke, J. Naous, R. Raghuraman, and J. Luo. 2007. NetFPGA–An Open Platform for Gigabit-rate Network Switching and Routing. In IEEE International Conference on Microelectronic Systems Education.

J.Moy. 1998. OSPF Version 2. RFC 2328.

Jeffrey Dean, Sanjay Ghemawat. 2008. MapReduce: simplified data processing on large clusters.

Communications of the ACM - 50th anniversary. Vol 51.

J Sgall. 1997. On-Line Scheduling—A Survey. In A. Fiat and G. Woeginger, editors, On-Line Algorithms, Lecture Notes in Computer Science. Springer-Verlag, Berlin.

Kanfman L., Rousseeuw P. 1990. Finding groups in data: An introduction to cluster analysis[M]. Canada: John & Wiley & Sons, Inc.

Kamhing Ho, James H.Rice, Jaideep Srivastava. 1990. Real-Time Scheduling of Multiple Segment Tasks, Department of Computer Science University of Minnesota Minneapolis, In 14th IEEE Annual International Computer Software and Applications Conference.

K. Vanmechelen, W. Depoorter and J. Broeckhove. 2008. Economic grid resource management for CPU Bound Applications with hard deadlines，In CCGrid' 08: The 8th International Conference on Cluster Computing and the Grid, Lyon, France.

Leiba B. 2009. Having One ' s Head in the Cloud [J]. IEEE Internet Computing, 13(5):4-6.

Liu XiaoQian. 2008. A cost-computing based task scheduling in grid market. Journal of Graduate School of the Chinese Academy of Sciences, 25(3): 397-385.

Madhukar Korupolu, Greg Plaxton, and Rajmohan Rajaraman. 1999. Placement Algorithms for Hierarchical Cooperative Caching. In Proceedings of the lOth Annual Symposium on Discrete Algorithms.

Milligan G. 1980. An examination of the effect of six types of error perturbation on fifteen clustering algorithms[J]. Psychometrika, 45(3): 325-342.

M. Maheswaran, S. Ali, H.J. Siegel, D. Hensgen and R.F. Freund. 1999. Dynamic mapping of a class of independent tasks noto heterogeneous computing systems [J]. Journal of Parallel and Distributed Computing. 59(2), pp. 107-131.

Nicolas Bonvin, Thanasis G. Papaioannou and Karl Aberer. 2009. Dynamic Cost-Efficient Replication in Data Clouds, ACDC'09, Barcelona, Spain.

Nicolas Bonvin, Thanasis G. Papaioannou and Karl Aberer. 2009. The Costs and Limits of Availability for Replicated Services, ACDC'09, June 19, Barcelona, Spain.

O. Ibarra and C. Kim. 1977. Heuristic algorithms for scheduling independent tasks on nonidentical processors. Journal of the ACM. 77(2), pp. 280-289.

Qiyuan Jiang. 1993. Mathematics Model [Second Edition]. Bei Jing: Higher Education Press.

Rashedur M. Rahman, Ken Barker, Reda Alhajj, 2008. Replica Placement Strategies in Data Grid, J Grid Computing: 103-123.

R Abbott and H Garcia-Molina. 1992. Scheduling real-time transactions: A performance evaluation. ACM Transactions on Database Systems, vol. 17, no. 3, 513-560.

Rodrigo N. 2009. CloudSim: A Novel Framework for the Modeling and Simulation of Cloud

Computing Infrastructures and Services.

R. Buyya, D. Abramson and J. Giddy. 2000. Economy driven resource management architecture for computational power Grids. In: PDPTA ’00: Proceedings of the 7th International Conference on Parallel and Distributed Processing Techniques and Applications.

R. Buyya, M. Murshe and D. Abramson. 2002. A deadline and budget constrained cost-time optimization algorithm for scheduling task farming applications on global grids, In: ICPDP ’02: The 2002 International Conference on Parallel and Distributed Processing Techniques and Applications, Las Vegas, Nevada, USA.

R. Buyya, D. Abramson and J. Giddy. 2000. Nimrod/G: An architecture for a resource management and scheduling System in a Global Computational Grid, In: HPC ASIA 2000: 4th International Conference and Exhibition on High Performance Computing in Asia-Pacific Region, Beijing, China.

R. Armstrong, D. Hensgen, and T. Kidd. 1998. The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions, In: HCW ’98: the 7th IEEE Heterogeneous Computing Workshop, pp. 79-87.

R. Freund, M. Gherrity, S. Ambrosius, M. Campbell, M. Halderman, D. Hensgen, E. Keith, T. Kidd, M. Kussow, J. Lima, F. Mirabile, L. Moore, B. Rust, and H. Siegel. 1998. Scheduling resources in multi-user, heterogeneous, computing environments with SmartNet, In: HCW ’98: the 7th IEEE Heterogeneous Computing Workshop, pp. 184-199.

R. Buyya, J. Giddy and D. Abramson. 2000. An evaluation of economy-based resource trading and scheduling on computational power grids for parameter sweep applications, In: AMS ’00: The Second Workshop on Active Middleware Services, Pittsburgh, USA.

R. Buyya and M. Murshed. 2002. GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing, The Journal of Concurrency and Computation: Practice and Experience, Wiley Press.

Snarey M., Terrett N., Willet P. 1997. Comparison of algorithms for dissimilarity-based compound selection[J]. Journal of Molecular Graphits & Modeling, 15(6): 372-385.

Subramoniam K, Maheswaran M, Toulouse M. 2002. Towards a Micro-Economic Model for Resource Allocation in Grid Computing Systems, In: Proc. of the 2002 IEEE Canadian Conf. on Electrical & Computer Engineering, Manitoba: IEEE press, 782-785.

Sgall, J. 1998. On-line scheduling－a survey, in Online Algorithms: The State of the Art, A.W. Fiat, G. J., Editor. Lecture Notes in Computer Science. 196－231.

S. Kavitha, Golconda and F. Ozguner. 2004. A Comparison of static QoS-based scheduling heuristics for a meta-task with multiple QoS dimensions in heterogeneous computing,

Proceedings of IPDPS ’04.

Sih GC, Lee EA. 1990. Dynamic level scheduling for heterogeneous processor networks. Proceedings of the 2nd IEEE Symposium on Parallel and Distributed Systems.

Sih GC, Lee EA. 1990. Scheduling to Account for Inter-processor Communication Within Interconnection-Constrained Processor Networks. Proceedings of the International Conference on Parallel Processing: 9-16.

Sih GC, Lee EA. 1993. A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures. IEEE Transactions on Parallel and Distributed Systems, 4(2): 175-187.

TL Saaty. 2008. Decision making with the analytic hierarchy process. International Journal of Services Sciences.

V. Lo, J. Mache, and K. Windisch. 1998. A comparative study of realworkload tracesand synthetic workload models for parallel job scheduling. In Proceedings of the 4th Work-shoponJobScheduling StrategiesforParallelProcessing.

Vogels W, Eventually consistent. 2009. COMMUNICATIONS OF THE ACM 52.1:40-44.

V. Sundaram, A. Chandra and J. Weissman. 2008. Exploring the throughput-fairness tradeoff of deadline scheduling in heterogeneous computing environments, In ACM SIGMETRICS’ 08: The 2008 International Conference on Measurement and Modeling of Computer Systems, Annapolis, Maryland, USA.

Wenying Zeng, Yuelong Zhao, Kairi Ou, Wei Song. 2009. Research on Cloud Storage Architecture and Key Technologies, ICIS 2009, November 24-26, Seoul, Korea

Ward J. 1963. Hierarchical grouping to optimize an objective function[J]. Journal of American Statistical Association, 58: 236-244.

Yasushi Saito, Marc Shapiro. 2005. Optimistic Replication, ACM Computing Surveys, Vol.37, No.1, pp.42-81.

Zhou Xu, Lu Xianliang, Hou Mengshu Wu Jin. 2004. A Dynamic Distributed Replica Management Mechanism Based on Accessing Frequency Detecting, Operating systems review, Vol.38 (No.3).

致 谢

二零零六年四月，我第一次走进中国科学技术大学，带着忐忑的心参加研究生入学复试考试。几个月后，当我第一次走进教室时，没有想到会在科大停留五年。这五年里，我度过了生命中最精彩最重要的时光，遇到了平易近人严谨求实的杨老师，遇到了重要的另一半，遇到了一群志同道合的兄弟姐妹。匆匆的五年，当提笔写致谢的时候，才发现身边的一切如此美好，有这么多值得珍惜的人、值得珍惜的物、值得珍惜的回忆。心里满溢的是无边的幸福感，是说不出的感动，更是无尽的感激。

特别感谢我的导师杨寿保教授。在科大的五年里，杨老师给予了我宽松的学习环境和悉心的指导。杨老师严谨求实的科研态度，幽默开朗、积极乐观、热心助人的生活态度和个人品质都值得我学习终生。记得周末的时候，杨老师耐心的帮我修改毕业论文；记得给灾区捐款的时候，杨老师毫不犹豫的捐了五千；记得遇到求学的考研学生时，杨老师热心帮助；更记得四月一日的上午，杨老师兴冲冲的愚弄我们：本.拉登被活捉啦……回忆太多太美好。片片回忆留下的是终生难忘的师生情！

感谢 863 项目的负责人陈华平教授！感谢帮我找实习的李京教授！

感谢网络中心的钱老师、夏老师、封老师、汝老师，感谢运动和宇宙，对我的关心、帮助和指导。

感谢几年来一起科研和生活的网络中心的兄弟姐妹：王绍林、郭磊涛、王菁、孙伟峰、张蕾、申凯、董阔、王大鹏、赵叶红、韦冬、纪雯、路卫娜、张瑞、姜燕、徐钽、史凌志、张然美、郭良敏、刘玉枚、滕达、胡云、武斌、胡玲玲、陈为、程伟、张鹏、陈波、赫卫卿、王淑玲、刘福丽、谢丽华、侯冠博、赵元韬、郭晓雷、牛宪龙、宋浒、黄彦彬、徐婧、是丽俐、方君、周文煜、王喜妹，真心的祝福你们一切顺利。同时感谢室友方方、小袁、云云、何超，祝你们幸福快乐。

感谢我的父母，对我默默的支持！

感谢评阅论文及参加答辩的老师们！

刘晓茜

2011-3-22于中国科学技术大学

网络信息中心实验室(USTC-NIC)

攻读博士学位期间发表的学术论文

1. **刘晓茜,** 杨寿保, 雪花结构：一种新型数据中心网络结构，计算机学报，已录用。
2. **刘晓茜**, 杨寿保, 郭磊涛, 网格市场中基于成本计算的任务调度研究. 中科院研究生院学报. Vol.25 No.3, 379-385, 2008 5.
3. **Xiaoqian Liu**, Shoubao Yang, Dong Wei, "A trust evaluation model based on resistive network", Proceeding of 5th ChinaGrid Annual Conference (ChinaGrid 2010 Workshop).
4. **Xiaoqian Liu**, S.B. Yang, Shuling Wang, "R-DLS: An Improved DLS Algorithm", Proceeding of 7th International Conference on Grid and Cooperative Computing (GCC 2008), October 24, Shenzhen.
5. **Xiaoqian Liu**, S.B. Yang, Leitao Guo, "Profit Sharing Based Routing Trust Evaluation Algorithm in P2P Network", Proceeding of 3rd ChinaGrid Annual Conference (ChinaGrid 2008), August 20, Dunhuang.
6. **Xiaoqian Liu**, S.B. Yang, Kai Shen, "Trust-Aware Resource Transaction Model in P2P Networks", Journal of Huazhong University of Science and Technology, Nature Science, October 2007.
7. 武斌, 杨寿保, 徐婧, **刘晓茜**. 网格市场中一种模糊决策的多维 QoS 批调度方法. 小型微型计算机系统. 2009 年 12 月 VOL30, NO12. 2428-2432.
8. 宋浒, 杨寿保, **刘晓茜**, 郭良敏. 网格市场中服务质量驱动下的任务调度算法. 中国科学院研究生院学报. 2011, 28(1) 86-93.
9. 牛宪龙, 周文煜, 武斌, **刘晓茜，**杨寿保. 新型数据中心网络结构研究进展综述. 第二届中国国家网格学术年会. 2010 年 1 月, 69-75.
10. 徐婧, 杨寿保, 王淑玲, **刘晓茜**. CDRS：一种云存储中代价驱动的自适应副本策略, 中科院研究生院学报, 已录用.
11. 武斌, 杨寿保, **刘晓茜**. 一种动态网格资源可用性风险评价方法. 中国科学技术大学学报. 已录用.
12. Xianlong Niu, Shoubao, Yang, Bin Wu; **Xiaoqian Liu**, Liangmin, Guo. A cache scheduling scheme based on layered coding VOD system. 8th International Conference on Grid and Cooperative Computing, GCC 2009, 238-243
13. Dong Wei, Shoubao Yang, **Xiaoqian Liu**. Artificial Immunology Based Anti-Pollution P2P File Sharing System. 2007 International Conference on Grid and Cooperative Computing, 中国 乌鲁木齐, 2007.08.16-18, 81-87.
14. Shen Kai, Shoubao Yang, Wei Chen, **Xiaoqian Liu**. Balancing risk and price: An

opportunity-cost approach for job scheduling in the grid market. Sixth International Conference on Grid and Cooperative Computing, Proceedings, 2007: 521-527.

15. Bin Wu, Shoubao Yang, **XiaoQian Liu**. A resource service risk based scheduling algorithm in grid market. NCM 2009 - 5th International Joint Conference on INC, IMS, and IDC(INC2009), p 25-30, 2009. (EI Accession number: 20100212622300).
16. Bin Wu, Shoubao Yang, **XiaoQian Liu**. A Resources Composing Based Batch Scheduling Algorithm in grid market. 2009 International Conference on Computational Intelligence and Software Engineering (CiSE 2009), 2009. (EI Accession number: 20101212799814).
17. Hu Song, Shoubao Yang, Bin Wu, **Xiaoqian Liu**, Liangmin Guo. An Optimal Algorithm for Scheduling Tasks within Deadline and Budget Constraints. 2009 Fifth International Joint Conference on INC, IMS and IDC(NCM2009), 2009. pp.62-65.

攻读博士学位期间参与申请的专利

- **国家发明专利：基于基尔霍夫定理的节点可信度评估算法（已通过审核）**
  - 申请号：200910079036.2
  - 申请日：2009.3.3
- **国家发明专利：网格市场中基于信任过滤的计算节点选择算法**
  - 申请号：200810242775.4
  - 申请日：2008.12.16

攻读博士学位期间参加的科研项目

- **国家自然科学基金项目**
  - 时间：2006.2～2010.12
  - 项目名称：**网络计算环境中信任感知的资源交易模型**
  - 工作职责：项目成员
- **国家 863 项目**
  - 时间：2006.9～至今
  - 项目名称：**基于应用调度的网格服务环境及若干网格应用研制**
  - 工作职责：项目负责人
- **中国科学技术大学研究生创新基金项目**
  - 时间：2009.7～2010.7
  - 项目名称：**校园云网络及其典型应用学伴平台的研究**
  - 工作职责：项目负责人

云计算数据中心结构及其调度机制研究

作者： 刘晓茜
学位授予单位： 中国科学技术大学