清华大学

综合论文训练

题目：数据中心的网络拓扑结构

系别：自动化系

专业：自动化

姓名：宋晓龙

指导教师：曹军威

2011年6月6日

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：学校有权保留学位论文的复印件，允许该论文被查阅和借阅；学校可以公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存该论文。

(涉密的学位论文在解密后应遵守此规定)

签　名：_____导师签名：_____日　期：_____

中文摘要

在计算机行业，有关云计算的内容发展越来越迅猛，而云计算的性能主要通过改善网络环境以及提高数据中心的运算能力来进行改善.本文旨在通过提出一种新型的计算网络的拓扑结构，来提高云计算网络中心中计算机之间的数据交换，从而提高整体的计算性能。

在经过了一系列的理论分析后，通过使用 Network Simulator2 软件进行网络仿真运算，来比较新的网络拓扑结构与以往拓扑结构之间的性能改善，并且针对特定状态下的应用，对整个结构进行优化配置。

关键词： 云计算；拓扑结构；网络仿真；优化

Abstract

In the industry of Internet, the development of Cloud computing is getting more and more rapid. The incrementally expanded efficiency of Data Center depend on the physical connections of severs. This paper proposed a new type of topology of Data Center network, to appeal with the scalable data exchange in Data Center network in distributed environment. The new topology provides a fault-tolerant routing and high capacity than traditional network.

With a series of theoretical analysis, we use Network Simulation 2 to simulate the topology in realistic states, to compare the new topology with other solutions before. And optimization of a certain environment is also considered.

Keywords: Cloud computing; Topology; Network Simulation; Optimization

目录

第1章 发展背景

1.1 云计算的概念及其发展

云计算的雏形概念来源于 Google 的工作模式和对应产品，现在云计算概念一般用来指我们所见到的能够按需配置获取资源和计算能力的，用户只接触平台，而不需要去了解平台的下层软硬件内容的网络计算方式，用户所使用的终端可以具有很少的计算能力，但是只要能够进行网络连接，就可以通过云端进行大量的数据远程计算，来实现对客户的服务。

在某种程度上说，云计算是行业内大量技术发展最终结合的产物，随着包括网格计算、虚拟化、Web Service 等技术在内的网络技术的发展，以及网络带宽的不断提高，使得直接通过广域网络进行共有的 SaaS 服务成为可能。同时，随着用户对隐私问题和安全问题的深入考量，以及对能耗等环境保护问题的逐渐重视，作为集中数据存储和计算能力的私有云也得到的发展。一般认为现在云计算能够提供三种方式的服务模式，包括 IaaS(Infrastructure-as-a-Service)，PaaS(Platform-as-a-Service)和 SaaS(Software-as-a-Service)。

IaaS(Infrastructure-as-a-Service)，基础设施即服务。用户可以通过 Internet 得到完善的计算机基础设施获得服务。一般认为 Amazon 的 EC2 服务就是这样的一类服务。

PaaS(Platform-as-a-Service)，PaaS 实际上是指将软件研发的平台作为一种服务，以 SaaS 的模式提交给用户。因此，从某种意义上说，PaaS 是 SaaS 模式的一种应用模式。但是，PaaS 的出现可以加快 SaaS 的发展，尤其是加快 SaaS 应用的开发速度。PaaS 的服务模式主要是提供了一类可以进行程序开发的平台，使得用户能够在这个平台上开发自己需求的应用，Google 的 App Engine 就是这个应用模式的先驱。

SaaS(Software-as-a-Service)：软件即服务。它是一种通过 Internet 提供软件的模式，用户无需购买软件，而是通过软件的租赁等方式来进行软件使用的服务。相对于传统的软件，SaaS 解决方案有明显的优势，包括较低的前期成本，便于维护，快速展开使用等。随着 Web Service 概念的发展完善，大量地网络应用已经开始出现，主要包括了各类网页应用和 RSS、微博、社交网络等产品，现

在已经普遍商业化的 SaaS 可以看做把所有东西都集中在浏览器中运行，而 Google 推出的 Chrome OS 也正是顺应了这样的一个潮流。

鉴于当前并没有比较权威的云计算的定义，行业内所提出的概念和定义基本上是各大企业对自身服务的一种描述性的定义，但是基本上公认的云计算（Cloud Computing）内容是网格计算（Grid Computing ）、分布式算（Distributed Computing）、并行计算（Parallel Computing）、效用计算（Utility Comuting）、网络存储（Network Storage Technologies）、虚拟化（Virtualization）、负载均衡（Load Balance）等传统计算机技术和网络技术发展融合的产物。它旨在通过网络把多个成本相对较低的计算实体整合成一个具有强大计算能力的大型系统，并借助 SaaS 等商业模式把这强大的计算能力推送以分布到终端用户手中。云计算的一个核心理念就是通过不断提高“云”的处理能力，进而减少用户终端的处理负担，最终使用户终端简化成一个单纯的输入输出设备，而将绝大部分计算在所谓的云端，也就是大型数据中心来完成。

现在比较普遍被接受的云计算的优势在于灵活的按需定制、较高的可扩展性、比较完善的安全性和隐私保密措施，以及较低的成本。通过虚拟化技术的完善，云计算服务的供应商能够保证计算机的计算能力的高度可扩展性和相对于传统计算机来说更快的配置速度。通过并行计算的发展和算法的不断改进，使得计算机的运算能力得到了很大的集中和高效的利用。并且，由于云计算服务一般通过设计比较完善的文件系统和备份方式来进行数据的存储，提高了数据的安全性和可达性，提高了整个网络的效率。

一般来说，行业内直接提出云计算概念并且开始进入产品化流程是在 2007 年左右，包括 Amazon、IBM、Google 等行业领先的企业，开始提出他们自己得云计算概念和相应的产品，其中最早将云计算的概念成功进行产品化并进行了商业运作的是 Amazon 的 EC2 平台，使得用户可以通过租用虚拟机，通过在上面运行的 Ubuntu 系统，在短期内获得大量的计算资源来完成需要的任务，而并不需要为了这些计算任务购置相应的大量计算机，并且 Amazon 的服务在一定程度上提供了虚拟机性能的垂直扩展，使得用户可以按需定制自己不能性能的虚拟机，或者通过租用大量虚拟机来完成原本多线程的任务。

其中最为著名的案例就是纽约时报公司在将 1851 年到 1922 年地纽约时报的影像资料转换为 PDF 文档时进行的工作，当时纽约时报集团的任务是将过去几十年间已经进行摄影存储的纽约时报文档进行数字化的存储，主要是将图片转化为

比较便于查阅的 PDF 文档。为了进行高速的转换，该公司在 Amazon 的 EC2 平台上购买了 100 个计算实例，并且使用 Hadoop 程序将超过 4TB 的数据进行了 PDF 转换，而时间仅仅用了 24 个小时，所用的费用不到 3000 美元（其中包括了带宽、租用 EC2 实例等全部费用），远远低于刚开始计划购买 100 台 PC 机进行转化所使用的费用。

而 Google 则是将云计算服务主要应用在了自己得搜索服务之内，最早就是由 Google 在 SIGCONN 上提出了自己云计算框架的三大内容，包括新的文件系统 Google File System，新的数据库存储方式 Bigtable[1]，以及对分布式资源利用率非常高的 Mapreduce[2]计算框架，使得之后的大量地云计算研究都离不开 Google 的影响，其中 Mapreduce 的框架经过了大量的发展，已经成功的绝大部分企业的云计算内容中出现，并且衍生出了诸如 Hadoop 这样的开源项目，来实现大量数据的并行计算。

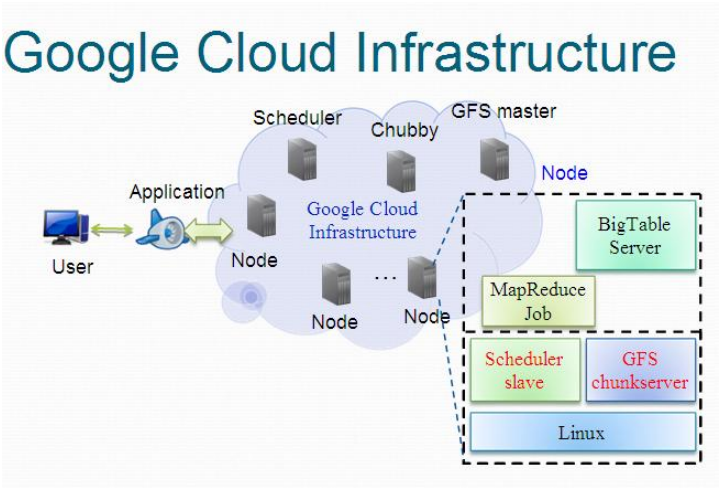

图 1.1Google 的云计算架构

Google 提出的三种技术，也分别解决了云计算在不同层面所面临的问题。但是比较不引人注目的是 Google 的服务器技术和相应的数据中心配置，这方面的内容一直是保密的，但是这并不妨碍我们从一些新闻或者介绍中得到蛛丝马迹。从现在我们了解到得情况来看，Google 的数据中心采用的是大量地廉价计算机组成的计算网格，通过在上面运行软件层面的 Mapreduce 来实现搜索和其他任务，而对于数据中心的硬件结构，也就是基础的计算设施，Google 的选择是自制了大

量廉价计算机，进行一次性的连接形成一个大的计算单位（有消息说是 1160 台计算机，装在一个集装箱中），而对每台个体计算机来说，一旦计算机出错，就会进行替换，而通过 GFS[3]和 Bigtable 的备份来实现无缝过渡。这种特性对数据中心网络内部的延时和带宽的要求非常大，从而对网络的拓扑结构和网络设备都提出了新的要求。据称，Google 和微软现在在其数据中心内使用的交换机和路由设备全部都是自行定制生产的。

在此之后，网络计算的概念越来越深入人心，整个计算机业界和科研领域都开始关注与云计算有关的一系列内容。随着有关云计算的技术研究也越来越深入，而我的毕业课题就选择了这个比较跟进当前技术发展背景的题目，延续我在本科前面阶段进行的 SRT 研究，继续深入探讨一种可行性较高的数据中心网络拓扑设计，通过在本学期的研究来完成这个课题。

1.2 数据中心面临的问题

介绍完前面的应用背景，我们开始更深入的从技术层面的探讨我们所使用的当前的云计算架构。云计算的概念含义本身就包含云的意义，也即是说，对于用户来说，用户只能接触到直接使用的内容。举例来说，对于 Amazon 的 EC2 用户，他们接触到的是一个虚拟机的实例，对于网络登陆后，云计算的内容就是一个可用的 Ubuntu 操作系统，和在真实机器上安装一个这样的系统别无二致。而 Google 的 App Engine 则是提供了一个代码编译和执行的环境，用户只需要和代码打交道，然后将代码上传到 Google 的平台即可。

虽然对于用户来说，透明的云是一种可以简化工作流程，提高效率的做法，但是这对我们设计云的内容提出了更高的要求，一般可以认为，云计算的大致内容可以通过以下分层来实现。

其中，Datacenter 就是作为硬件架构的最底层，比如在 Google 的全球框架内，我们可以看到每个 Datacenter 都是一个独立的计算单位，在他们上面跑着无数的应用，但是同时，不同的数据中心之间也进行着大量的数据交换来完成数据的备份等工作，从而实现了数据的安全存储等一系列内容。

在其上的一层，也就是服务器，前文提到过，现在行业的发展趋势就在于用大量廉价的可替换计算机来代替以往的大型机进行计算，并且使用 Mapreduce 等框架将其计算能力进行整合，这里的 Server 一般都是使用比较简单的计算机，

比如 Google 使用的自制的简易计算机，使用了定制的硬件和特殊的电源来保证长期工作。之前曾经有过一些比较模糊的新闻对 Google 进行调侃，说 Google 的硬件管理员都是穿着轮滑鞋拖着机箱，一旦检测出哪个计算机出现了问题，二话不说就登上轮滑鞋跑去整台机器都给换掉，而新加入的机器可以在完善的网络环境和数据存储方式中，很快融入新的计算网络，补足故障计算机带来的损失。

再上面的一层就是平台层，主要的内容就是近些年来发展比较快的虚拟机，虚拟机是运行在物理机器上的一种虚拟设备，虽然依附于物理机器，但是可以在对外的表现上以一台真实计算机的状态出现，通过对虚拟机资源的分配的调度，云计算就可以在一定程度上实现计算资源的灵活配置，可以实现迅速高效的扩展和简化。同时也可以认为，类似 Google 的 App Engine 这样的产品，在编程上面实现的 PaaS 也是出于这一层。

最上层就是应用层，一般来说是各种基于网络的云计算应用，越往上层，用户能了解到的下层的架构就越少。通过这样的分层结构，我们就可以既能够使用户面对云地时候，不需要考虑底层软硬件的内容，又可以让开发人员高效的进行服务的开发。

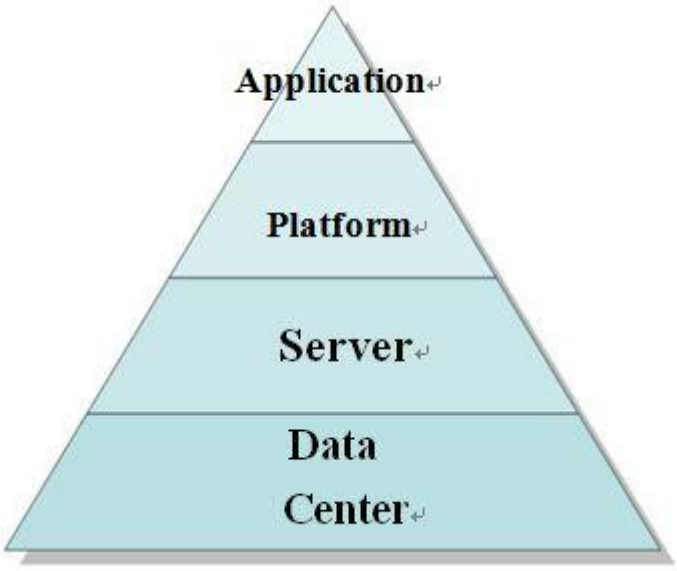

图 1.2 云计算的层级结构

当前我们所使用的云计算的所谓用户看不到的“云”无非就是上面的几个层

次，Mapreduce 的整个框架都是建立在 Application 这一层上，而相应的我们能使用到的服务，比如 Google App Engine 也是在最上面的 Application 层次，也即我们上传到服务器端的是一系列代码，而 Amazon 提供的 EC2 是直接提供了 Platform 层次的服务。

而其他几个企业的提供云计算的数据中心结构和和前文提到的 Google 的结构大同小异，

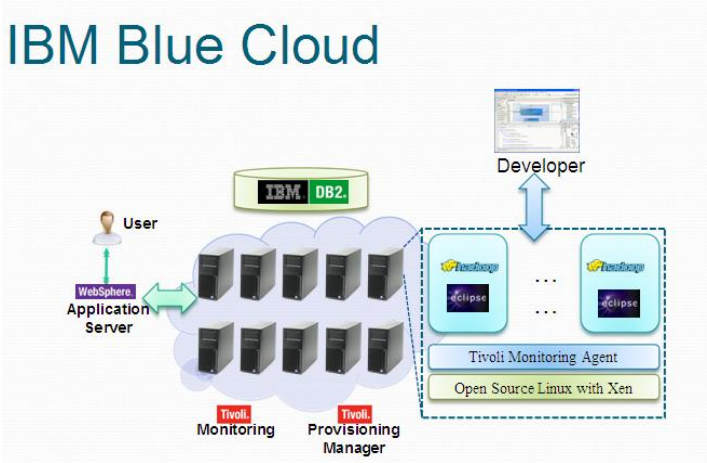

图 1.3IBM 的云计算架构

IBM 的蓝云通过 DB2 进行组织，在 Xen 虚拟机的基础上，整合的 Hadoop 的开源架构，并且利用 eclipse 的强大的开发能力，进行了整个结构的设计。

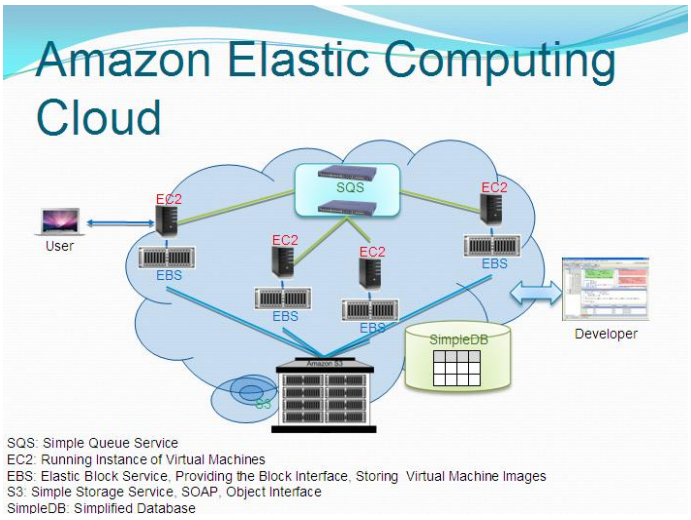

图 1.4 亚马逊的云计算架构

Amazon 的云计算结构是当前比较完善的一个结构，也是迄今为止唯一一个出于盈利状态的云计算商业应用。

但是，当我们把目光继续往下移，就会发现，包括服务器在内的数据中心的硬件架构，在多年以来一直没有重大改进。而几大企业的数据中心中，负责数据处理的计算机的连接方法并没有被过多的提及。

而所有的这些架构必须要依托硬件才能实现，以下要讨论的，就是从技术角度上来说，云计算需要用“云”的概念来隐藏的，具体的架构。

作为一个大型的数据中心，需要处理大量的数据，而这些计算机要进行的任务，绝大部分是简单的计算，与此同时，为了控制数据中心的成本，数据中心的计算机一般并不是高性能的服务器，而是大量的廉价计算机，随之而来的一个问题就是，当我们进行大量计算的时候，如何能保证整个数据中心内部的数据交换效率。显而易见的是，传统的总线式结构或者存在一个网络中心的结构中，总线或者网络中心节点是整个网络明显的性能瓶颈。相对而言，在服务架构虚拟化之后，物理网络会出现大量点对点或者一对多的并发网络通信，不难想到，耦合紧密、有着足够带宽和容错能力的网络结构会带来很大的性能提升。

这其实也是本文的内容所要解决的问题，随着云计算的概念和相应产品越来越成熟，包括 Dcell[4]结构在内的各种网络拓扑结构应运而生。

1.3 Dcell 网络结构和他的衍生结构

1.3.1 Dcell 的发展和超立方结构

在 2008 年地 SIGCONN 上，微软亚洲研究院的郭传雄博士和他带领的研究团队发表了一篇论文，题目是 DCell: A Scalable and Fault-Tolerant Network Structure for Data Centers， 让我们有机会了解了微软的云计算架构在硬件层面的分布。而且在此后的几年时间内，他们的团队又接连发表了关于这样一个网络拓扑结构的几种改进措施,陆续发表了包括 Bcube 和 Ficonn 两篇重量级的，在 Dcell 结构上又进行了发展换和改进的结构，让我们看到了一个越来越完善的应用在数据中心的网络拓扑结构。

Dcell 的基本目标就是在一个处理海量数据的数据中心中，通过不同于以往的连接方式，来实现不同节点之间的高度耦合，从而减少数据交流之间的延时，提高带宽和容错性。Dcell 通过一种分层架构、使用特殊的小型交换机来实现这样一种功能，从广义上讲，不管是 Dcell 也好，还是 Dcell 及其衍生架构，或者是我们即将提出的对于数据中心的新的网络拓扑结构而言，他们都是最早的超立方网络的一个结构变种。

超立方结构是在我们可见世界的立方结构的无限扩展，我们可以认为，一维的“方形”是一个点，二位的“方形”是一个平面的正方形，而到了三维，“方形”的概念就变成了一个八面体，也就是我们说的立方体。如果我们用纯数学的方式进行扩展，那么我们就可以在欧式空间中进行四维、五维甚至更高纬度的“方形”的构造。为了方便研究，我们是可以把高维度的立方体投影到低维度来研究的。一般来说，我们都会将这些结构投影在二维，以便进行数据的仿真和网络链路的连接。

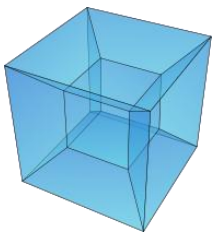

图 1.5 四维立方体的三维投影

而 Dcell 就是这样的超立方结构的一个具体实现。

1.3.2 Dcell 的物理结构和命名空间

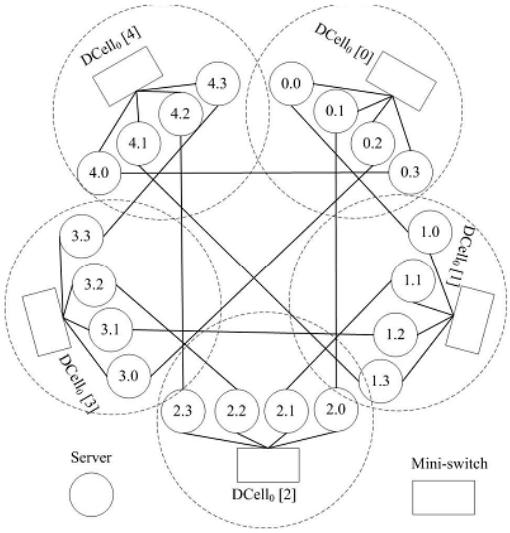

图 1.6Dcell 的结构图

Dcell 的物理结构是一个分层连接，相互耦合的立方网状结构，在构造的过程中，首先建立一个结构最小的 Dcell0 单元，这样的一个单元由一个 Dcell 结构专门设计的小型交换机和 4 台计算机组成，当这样一个单元连接完成后，一个 Dcell0 单元可以被看成一个虚拟的节点，在这样的结构中充当下一层结构的基本单元，在这样的一个连接过程中，最终保证整个 Dcell 结构是一个完整图。

而 Dcell 的命名空间是一个比较固定的做法，对于最基本的 Dcell 单元的 Dcell0[k]来说，我们让每个 Dcell 的最小结构中包含 n 个计算机节点，让 Dcell 结构的层数为 k，其中所包含的计算机节点的命名都为(k,n)，理论上说，当 Dcell 结构进行无限扩展的时候，这个有序的数组就会无限增长，从而完整的表示整个结构中的所有计算机。但是这里有一个比较容易忽略的事实，对于 Dcell 结构来说，当结构越扩展越大，使得整个数据中心的结构增加到一定程度的时候，交换机节点的数目也随之增长，对于没有完整命名空间的交换机节点来说，非常不易于从软件或者网络的角度直接判断哪一台交换机节点出现了问题，而这个弊端在

后来的发展以及此次提出的新结构中都得到了一定的改善。

对于初始的 Dcell 结构来说，它的命名空间和连接方式几乎完全无关，仅仅是在一个最小的 Dcell 结构中选择一个计算机节点，和另外结构的计算机节点相连，从数学上看，这是没有什么问题的，但是把这样的一个连接方式延续到实际应用中就会产生很多意想不到的困难，这也就是我在进行计算机模拟的时候，最开始遇到的问题。对于 Dcell 结构来说，如果不能用最简单的方法来表示任何两台计算机之间的连接方式，那么当我们进行实际的数据中心构建时候，就需要不停地进行 Dcell 结构的抽象思维，来判断到底哪两台计算机需要被连接起来。而当整个数据中心的计算机节点数目迅速增长的时候，仅仅是判断哪两台计算机节点需要被互相连接起来，就需要很长的时间，更不要说连接时候的效率问题了，之后的改进结构从一定程度上改善了这个问题，但是还没有达到一个比较好的效果。理论上说，最好的预期效果是，我们对每台新进入数据中心的计算机进行一个编号，只要有了这个编号，我们就可以知道这台计算机的哪几个端口需要连接到另外哪几个编号的计算机或者交换机上，从这个角度来看，交换机进行编号，进入命名空间也是很有必要的。

1.3.3 Dcell 结构中的边和计算机的数目

Dcell 结构是一个超立方结构的具体实现，当维度不断增加时，我们的计算机数目实际上并不是线性或者指数增加的，而是比指数增加的速度还要快。

其中 Dcell 结构中边的数目 tk 值如下：

$$(n+\frac{1}{2})^{2^k}-\frac{1}{2}<t_k<(n+1)^{2^k}-1$$

而在最早的 Dcell 结构中,计算机节点的总数目可以由完全图的递推来得出，当一层的节点数目是 tk-1 的时候，那么我们可以在其上的一层构建 gk 个对应的 Dcell 更高一层的结构，那么其中计算机的总数为 tk，递推方法如下

$$\begin{aligned} g_k &= t_{k-1}+1 \\ t_k &= g_k \times t_{k-1} \end{aligned}$$

通过这样的递推我们可以知道，Dcell 结构是一个可以高速增长，并且包含大量计算节点的数据中心的拓扑结构，举例来说，当 k=3，n=6 的时候，整个 Decll

结构中将有 326 万个计算机节点，这也就意味着，只要很简单的基层结构，就可以将现有的数据中心中所有的计算节点包含在内。

同时我们也可以知道。Dcell 结构中的对剖平面数目为

$$\frac{t_k}{4\log_n t_k} \; for \; k > 0.$$

1.3.4 路由和其容错算法

Dcell 结构中的路由是依靠在不同的 Dcell 单元之间进行的，其中最简单的路由算法是计算两个 Dcell[k]节点直接的路径，通过分层来找到对应的路径。举例来说，路由的起始和目标节点在同一个大的 Dcell 结构中，那么我们就开始进行递归计算。首先，找到这两个计算机节点所对应的最小的 Dcell 单元，也就是 Dcell[0]的单元，查看这两个单元是否在同一个 Dcell[1]的结构中，如果不是，那么就继续向上寻找,这两个对应的 Dcell[1]虚拟节点,是不是同在一个 Dcell[2]的结构中，如果不是那那么继续寻找，如果是，则在这个 Dcell[2]结构中，计算出两个对应 Dcell[1]节点之间用来连接的实际计算机节点，然后计算出路由的起点和终点是怎样通过交换机连接到这两个用于沟通的计算机节点的。

通过这样一个思路，我们也可以很轻松的了解 Dcell 结构是如何进行容错路由的。首先我们假设在整个计算机网络结构中，存在一个链路中断，那么对于我们来说，这个链路中断必将导致两个出于某一层的 Dcell 结构之间无法连接。但是 Dcell 的结构本身就从一定程度上规避了这样的链路失效，因为同一层中的几个 Dcell 节点是两两互联的，也即是说，即使有两个结构之前发生链路失效，那么我们还可以通过其他的路径，通过另外的一个 Dcell 结构来进行中间连接，从而达到规避失效链路的问题，容错路由的示意图如下，

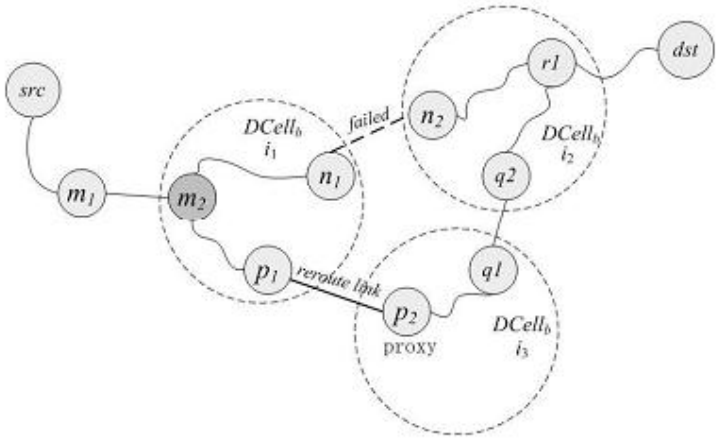

图 1.7　Dcell 的容错路由

1.3.5 作为改进型结构的 Bcube

在 Dcell 结构提出不久，郭传雄博士的团队就开始对这个结构进行了一定程度的改进，这就是之后发布的 Bcube[5]结构，Bcube 结构在很大程度上保持了 Dcell 原有的很多优点，并且在很大程度上改进了原来 Dcell 结构一些并不理想的特性。

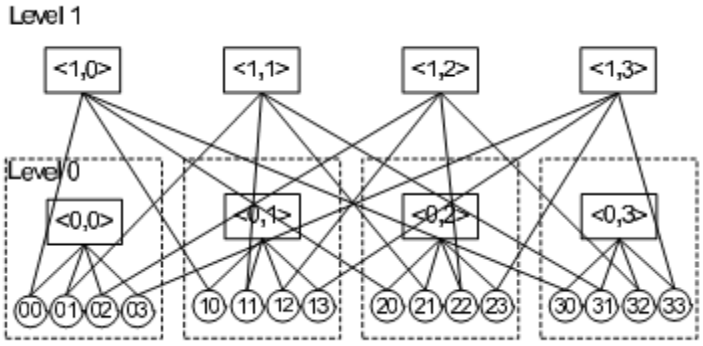

图 1.8Bcube 的结构

首先，Bcube 在 Dcell 的基础上，对命名空间进行了一定的修改，如前文所提到的，交换机节点本身在 Dcell 结构中是没有编号的，这就给整个系统的连接和排错带了了很大的压力。如上图所示，在 Bcube 结构中，一台小型交换机连接 n 台计算机，而这 n 台计算机分别命名为 00,01,02……，而在同一层结构中的另

外的交换机所连接的计算机命名 10,11,12……，并且最终计算机节点的总的层数为 k。为而对应连接的每台交换机分别命名为(1,0),(1,1)……这从某种程度上大大改善了原来 Dcell 结构中的交换机没有命名空间而导致的一些列问题。

这样的命名改进可以使得对整个计算机网络中节点数目和交换机数目的计算大为简化，对于一个最小节点中连接 n 个节点，并且有 k 层地 Bcube 结构来说，总的计算机数目为 n*2^k，而总的交换机数目为 k*n。

同时这样的命名空间可以使得整体的路由方案大大简化，原来的寻找同一层的节点，可以直接转化为寻找命名空间中第一个数字不同的其他节点，排除了原来在同一层寻找不同的 Dcell 结构的问题，而是直接转化成寻找同层可以连接的节点，与此同时，路由的整个路径可以用数字表示出来，减少了分析和实际应用中的计算失误。

之后该团队又发布了主要用于增加备份来保证数据安全的结构 Ficonn，这里就不展开说了。

第2章 一种新的网络拓扑结构

2.1 初创思路和结构设计

当我们仔细研究 Dcell 和它的一系列衍生结构后，我们发现，这一系列结构都是超立方结构的一些不同形态的转化，但是这些结构存在很多的弊病。首先，不管对于 Dcell 还是 Bcube 来说，整个结构中的计算机节点需要被多次连接，并且随着数据中心结构的扩大，需要的端口数目也越来越多。虽然对于较大的企业来说，自行制作特殊的迷你交换机，或者是能够在初始情况下定制多端口的计算机是一件很简单的事情，但是这并不能改变这样一个事实：如果我们能够减少计算机节点所需要的端口，并且将大量的接口工作转移到已经发展成熟的通用交换机上面，从直观上来说我们的成本将大大降低。同时，如果这样的一个结构拥有不低于、甚至高于当前 Dcell 结构及其衍生结构的性能的话，带来的成本和效率的提升将是空前的。

基于这样的一种思路，在我本科 SRT 阶段，和曹老师的博士生张帆学长就开始了这方面的基础研究。

首先是整个结构中的节点选择，我们比较倾向于在整个结构中使用定制完整的固定端口的计算机。从定制的角度来看，在一台服务器上面定制 3-4 个端口，远远比定制十几个端口但是将很多空闲不用来的简单的多。同时，在整个结构中放弃了不管是微软也好，Google 也好，他们所采用的定制小型交换机，而是换用了已经比较成熟的通用交换机来完成交换工作。在这期间我们仔细了解了行业内各大公司的基本思路，在实际应用中，对于比较完善的高度耦合的立方结构网络，当前的交换机能力大大超出了所需要的数据交换能力，于是很多企业就转向了特殊定制的小型交换机来完成数据交换的任务，而我们的结构则可以通过使用比较常见的中型交换机来完成数据交换问题，又不需要进行额外的定制。

从这样的角度，我们开始进行新的结构的设计，在构造方式上，我们的新结构比较接近 Bcube 的构建方式，主要是通过固定的命名空间和确定的数字关系来完成构造和连接工作。但是我们将原来结构中的连接方式进行了彻底的颠覆，我们并不是按照以往，将多台计算机连接到同一个交换机这样的连接方式，而是将思路一转，给同一台计算机连接多个交换出口，并且通过我们最终的研究，任何一个交换出口理论上来说都可以达到目的地，在这同时就大大增加了系统的容错

性能，保证了整个网络的数据交换能力不受突发的事件的影响而造成无法进行数据交换的问题。

同时我们也要意识到，仅仅在超立方的结构中并不能从根本上改善一个结构的性能，于是我们引入了新的机制，尝试做出一些不同于以往的超立方结构的事情，这些内容都将在下面叙述。

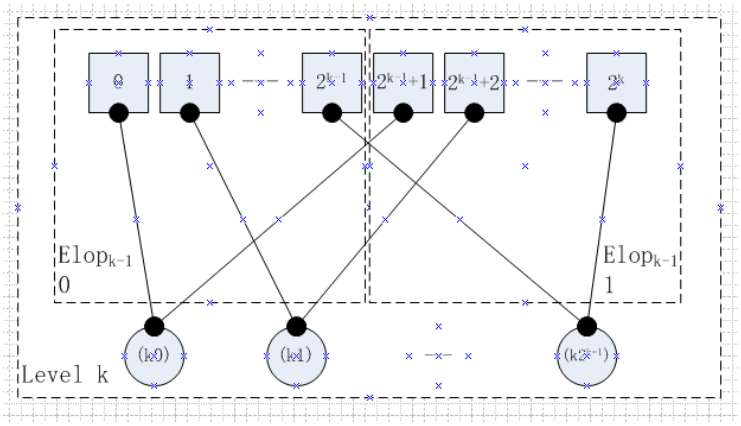

图 2.1 新结构的扩展方式

在以上的结构图中，方框表示交换机，而圆圈表示计算机节点。为了保证构造描述的完整性，我们引入了以下三个参数：

最小连接数 N，N 代表了每台计算机节点连接的交换机的数目，在某种程度上，n=2 的时候，对于维度的扩展，就相当于我们原来印象中超立方结构，因为我们的连接方式完全颠覆了原来的 Dcell 结构，所以当 n 增加的时候，并不是像原来的直接扩充超立方网格，而是变成超级 n 方体，从某种意义上来说，当我们选择比较合适的 n 的值的时候，我们的网络结构可以扩充出类似碳-60 的一种结构，或者其他更加复杂的空间结构。同时，对于大部分立方网络没有探讨的一个问题是，当不同的 n 的取值时候，整个网络的性能是否会有很大的差别，对于 n 来说，这是这个网络的最基础的属性，n 的改变带来的结构和性能的改变差异并不会小到可以忽略，那么就可能存在一个比较可能的 n 的最优化数值。

K 层数（维度），K 的取值对于我们来说就是原来的超立方网络进行扩展时候的维度，同时也可以单纯的在二维平面上认为，这是一个二维结构扩展之后的层数，层数这个参数的主要用途是配合 n 来组成整个网络的设计，如果我们不考虑

不完全图的情况下，一个固定 n 和固定 k 的网络总是一定的，但是当我们引入一定的不完全性的时候，会存在另一些情况的性能波动，这方面的问题留待后面讨论。

m 最小单元数目$m = n^{k-1}$ 时，结构是完全的。是否是一个完全结构是从实践的角度来考虑的。首先，对于一个正在建设中的数据中心，我们前面讨论过，不同的 n 值可能带来的完全迥异的数据中心性能。那么我们同时可以考虑，如果对于一个固定成本的数据中心，也就是说，当我们只能购买一定数量的计算机节点的时候，我们是否能够通过调节 n 的数值，同时在整个结构不是完全图的情况下，保证整个数据网络性能的提升，这一系列问题我们将尽量在本文以及后续的研究工作中解答。

2.2 构造方式和命名空间

当我们引入了 n，k 和 m 的数值后，我们可以开始构造一个实际的结构了。让我们以 n=2，k=3 为例子，来解释一下整体的构造方式和命名空间。

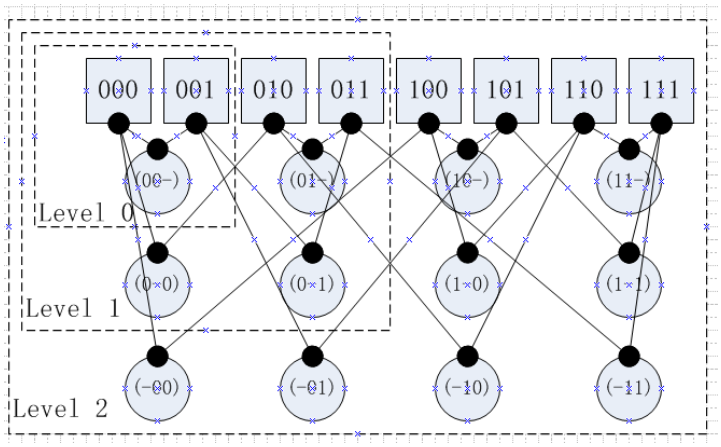

图 2.2n=2，k=4 时的新结构样例

首先，我们建立一个最小的单元，也就是一个计算机节点连接两个交换机节点的情况，我们将对应的交换机节点从 0 开始进行命名，交换机节点命名的进制数目就是 n，在这样的情况下，对整个结构中所有的交换机节点进行命名。

然后，我们回头来看计算机节点，由于每个交换机节点都有了名字，而且是

两个 n 进制的数字，而我们可以看到，在最小的一个单元中，两个交换机节点只有最后的一个数字不同，那么我们就把计算机节点的命名最前面的两个交换机节点设置为一样的，而最后一个数字设置为 n+1.

同理，当我们将整个结构向着 K 维度扩展的时候，我们可以发现一个有趣的事实，在外层连接交换机节点的计算机节点，对应的两个交换机节点命名必然只有一个数字不同，那么我们就将这个数字改为 n+1 来命名计算机节点。这样就形成了一个完整的 n+1 进制的命名空间，包含了所有的计算机节点和交换机节点，而其中含有数字 n 的节点就是计算机节点。

这样的命名空间在某种程度上比 Dcell 和 Bcube 的结构要大大增加了适用性和方便性，首先我们可以看到，对于 Dcell 中交换机节点不进行编号来说，将交换机和计算机节点统一纳入一个命名空间，对于我们的连接和研究整个结构都方便了很多。而相对于 Bcube 来说，虽然 Bcube 进行了交换机的编号，但是交换机编号和计算机节点的编号是完全不同的两个命名空间，而且如果我们尝试进行统一研究的时候，如果使用同样的数据结构来保存两种节点的信息，可能会产生命名类似而无法查错的情况。

而我们的新的命名空间，既可以将所有的节点都包含在里面，又能够直接通过比较判断节点的性质，而且通过两个命名数字的比较，我们可以直接看出两个节点之间的连接关系：

连接特性定理 1：当同一个位置只有一个数字不同，分别是 0 和 1 的两个节点一定是连接到同一台计算机节点的两个交换机节点

连接特性定理 2：而含有 n+1，但是位置不同的两个节点，一定是通过同一个交换机节点连接到一起的计算机节点。

由此可以看出，这样的一个命名空间优于 Dcell 系列的命名空间，当我们尝试用新的程序来处理整个网络的性能问题时候，会带来很大的便利。

整个构造方式的伪代码如下所示：

```
Build Construct(pref, n, k)
Part I:
if (l == 0) /*build Construct0 */
for (int i = 0; i < n; i++)
connect node [pref,k] to its switch[pref,i];
 return;
```

```
Part II:
for (int i = 0, i < k ; i++) /*build Construct S*/
BuildConstuct([pref,i], n, k-1);
connect n and n ;
return;
```

2.3 寻路和路由方式

新的命名空间必将带来新的寻路方式，为了简单叙述我们的新寻路方式，我们继续用 n=2，k=3 的结构来进行说明。

首先，我们通过命名空间一部分最后的说明，可以知道任何一个节点和与它互联的节点之间的数字关系，那么我们可以直观的想象到，我们可以通过数字的改变来直接得到整个路由过程中，经过的计算机节点和交换机节点。

举例来说，当我们需要从 200 号计算机节点，寻路到达 121 号计算机节点，那么我们可以在纯数学的结构上来找到这样一个完整的路径：

第一步，由连接特性定理 1，我们可以知道 200 节点连接到 000 和 100 编号的交换机节点，从数字上来说，100 交换机节点距离 121 相对来说要比较近，那么我们就选择首先达到 100 交换机。

第二步，我们要选择到达一个离 121 计算机节点更近的节点，我们可通过连接特性定理 2，得出，我们可以通过一个计算机节点作为中介，到达 101 交换机节点，而中间节点，我们可以通过连接特性 1 知道，编号应该为 201。

第三部，我们可以发现，101 交换机节点本身就和 121 计算机节点是相连的，那么我们的计算路径的目的就达到了。

这样我们就可以获得这样的一个路径 200-100-101-121，通过类似的计算我们就可以得到所有的理论上可以到达的路径。

同理，我们可以直接通过这样的寻路方式获得其他的路径，当我们遍历所有的数字解的过程中，我们也就得到了任何两个点之间的所有路径。

而对于同样重要的容错路由，我们也可以通过这样的过程来规避故障路径，如果在任何一个路径中存在不能通过的链路，那么我们可以回溯到上面一步，在两个交换机节点或者两个计算机节点之间选择另外一条路线。这样就避免了 Dcell 中会出现的一种情况，就是两个节点本身在物理连接上非常近，比如就是

两个比较大的 Dcell 结构之间的连接点，但是当我们需要寻找他们之间的链路的时候,本身存在一个非常接近的捷径,但是不停地回溯到高一级的 Dcell 结构中，会使得我们浪费了大量的计算，从而导致计算资源的浪费。

2.4 网络容量和路径容量

从上面的一些数据中，我们可以很轻松的计算，当结构是完全的，每个系统中有 N^K 个交换机，K * N^（K-1） 个计算机节点。节点当结构不是完全的，每个系统中有 N*M 个交换机，K*M 个节点。而对于整个网络结构来说，网络直径为 k+1，也就是说，在一个完全的网络中，相距最远的两个点之间通过 k+1 跳的路由即可到达目的地。

同时，作为一个有环网络网络，它的对剖平面个数也就是说通过切断最小的链路数目来使得整个网络分为两个互不相连的自制网络，这个数目为 n（k-1）。

对于使用相同的 n 和 k 的值来构造的 Dcell 结构，当 n=2,的时候，也就是两种结构同时为超立方结构的的一个子集的情况下，两种结构的增长方式是类似的，但是当 n 选择增加的时候,新结构的计算机节点容量带来更大的增加,也就是说，整个网络的计算能力得到了提高，对于不同的 n、k，新结构的计算机节点容量如下表：

表 2.1 新结构在不同参数下的计算机容量

k / n	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9
2	4	12	32	80	192	448	1024	2304
3	6	27	108	405	1458	5103	17496	59049
4	8	48	256	1280	6144	28672	131072	589824
5	10	75	500	3125	18750	109375	625000	3515625
6	12	108	864	6480	46656	326592	2239488	15116544
7	14	147	1372	12005	100842	823543	6588344	51883209
8	16	192	2048	20480	196608	1835008	16777216	150994944
9	18	243	2916	32805	354294	3720087	38263752	387420489

我们可以看到,当我们使用性能相近的计算机的时候,我们可以直观的认为,

在同样的结构复杂度的情况下，新的结构带来可定制性和计算机节点的计算能力远远超出了 Dcell 结构。

但是另一方面，在数据中心的各个计算机节点之间，我们要处理很多数据的交换，这也就对数据链路的容量带来了很大的要求，一个可以遇见的情况是，如果我们只增加计算机节点的数目，而网络链路的容量并没有增加，那么整体的网络性能和计算性能其实并不会得到什么提高，那么我们下面就看看，新的结构是不是能带来网络链路容量的提高。

首先，我们要了解，当我们把超立方结构看成一个单纯的图结构的时候，我们就可以看到，在整个 Dcell 就是一个完整的图，而当我们连接的时候，会发现，新的结构相当于 Dcell 结构中交换机和计算机节点之间的地位的对换，这在图论中相当于两个图之间的对偶，下面我们看一下在这种对偶过程中带来了什么变化。

在对偶的过程中，点和连线之间的关系发生了改变，我们会尝试用点来代替连线，而在原来的连线中间保证有一个节点。

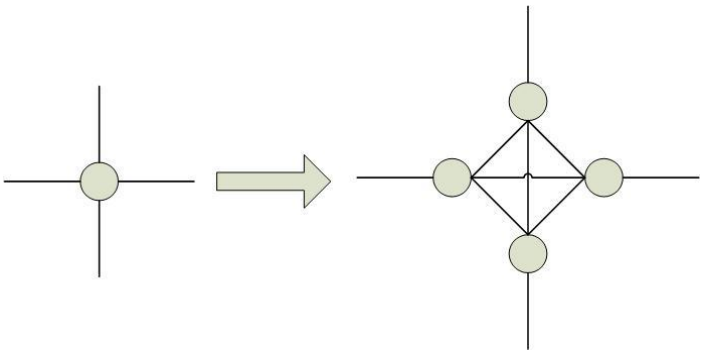

图 2.3 对偶图时的链路增长

我们来看最常见的一个节点连接了四个链路的情况，当我们把节点和连线之间的进行对偶，我们可以看到，为了保证节点之间连接关系不变，每个链路都会变成一个节点，但是另一方面，原来的一个节点并不是变成了对应的一个链路，而是一次性的变成了六条链路。从这个方面，我们可以看出，我们的对偶操作，在某种程度上，大大增加了同样的网络中的链路数目，当我们使用同样的连接宽带的时候，也就意味着整个网络的链路容量大大的扩充了。

第3章 基于 NS2 的网络模拟实验

3.1 NS2 软件

NS2 是 Network Simulator version 2，NS（Network Simulator）是一款开源的网络仿真软件，能够完成一系列的网络仿真工作。由于 NS2 的内容涉及到了基本上涉及到了网络信息交换过程中的所有过程，我们可以通过编写 NS2 代码，进行 NS2 实验来完成网络拓扑特性和其他性能的测试和仿真。所以，NS 成了当前进行大规模实际测试之前，或者不能进行大规模实际测试的时候的比较完善的方针替代品。但是另一方面，NS2 的开放源代码和整个计划没有一个强有力的统一组织者，使得 NS2 的易用性有所下降。

NS2 是一种时间驱动的离散事件网络仿真工具，由 UC Berkeley 开发，旨在通过软件仿真工具来模拟现实网络环境中的一系列问题，达到工程预研和学术研究的目的。目前，NS2 作为一个比较完善的软件工具，已经能够从很多方面仿真现实网络的真实情况。能够对网络各层的应用和流量进行贴近实际的模拟，包括 TCP、UDP 等网络协议和，FTP 等网络应用，以及路由算法比如 Dijkstra 等路由算法。并且在当前无线网络发展迅猛的情况下，NS2 也能进行一系列的无线网络仿真，包括 Adhoc 和各种 3G 无线网络都能通过 NS2 进行完善的仿真模拟。而对于网络更加底层的 Mac 协议，NS2 也能够完善的处理和仿真。

NS2 的使用和开发主要通过 C++和 Otcl 语言来进行。Otcl 语言是一个脚本语言，主要通过描述 NS2 模拟网络中的仿真事件来控制仿真过程的进行，通过设定一个完善的仿真时钟和队列行进过程，来激活和执行 NS2 仿真框架中的事件和组件。通过前面的介绍我们可以看出，NS2 是一个事件驱动的仿真程序，也即在 NS2 的仿真过程中，所有的执行和分组通信并不消耗仿真时间，而是通过每个事件的调度来进行仿真钟的推进，从而进行下一步的工作。

NS 的底层文件是通过 C++来编写的，对于 NS2 中已经有的一系列可用的节点、事件调度器等工具，可以通过 C++高效的完成数据的分发和路由的管理，而使用者也可以通过 C++编程来对网络底层进行更改，从而实现更多的网络功能的方针，在我们下一步的工作中，也可以通过 NS2 的 C++编程，对新结构的路由算法和寻路方式进行改进，从而对新型网络结构的容错路由和寻路效率进行优化研究。

当仿真过程进行完毕后，NS 将会一个用来记录整个仿真过程的 LOG 文件。其中会包含整个仿真过程的时钟和事件调度的内容，这些数据可以通过其他的程序进行分析，来对整个网络的性能问题进行研究探讨，本文的下面部分就是用这样一个过程来进行研究的，另一方面，我们也可以使用 NAM 将整个仿真过程展示出来。

3.2 网络拓扑结构的构造

为了试验方便，首先我构造了一个 n=3，k=3 的 Dcell 结构和新的网络拓扑结构来进行实验，实验的主要内容主要是在整个网络中，通过设定 UDP 或者 TCP 连接来进行随机数据传输，在为了方便起见，实验中主要使用了 UDP 连接来进行数据传说。采用的数据流使用 NS2 自带的泊松分布的数据发生程序进行随机数据输出，并且在所有的实验中，随机数据包的分组大小全部设置为 1000 字节，发送速率设置为 40Mbps，并且在数据发送过程中加入了随机扰动，来保证整个实验的均一性。

对于所有在实验中的数据连接，全部使用了 100Mbps 带宽，链路物理延时为 10ms 的实验环境，并且对所有的计算机节点绑定 UDP 的数据发送和接受的上层代理。

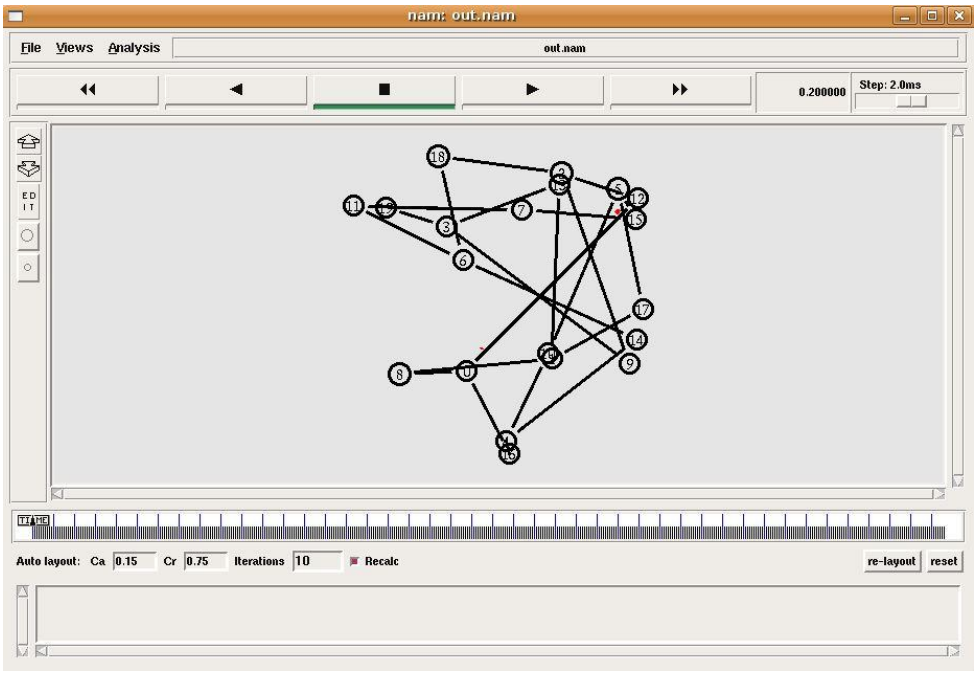

图 3.1NS2 的界面

以其中一次的实验示意图为例，在未进行具体设置的时候，NS2 会将设定的所有节点进行随机排序，图中示意的节点名称是按照设定的 n+1 进制的节点名称进行了 10 进制的转化，当模拟软件进入实验状态时，会产生一个记录文件，用来记录整个模拟过程中每个数据节点之间发生的事件。

```
+ 64.13017 0 16 ack 40 ------- 1 12.0 16.0 3585 9233
- 64.13017 0 16 ack 40 ------- 1 12.0 16.0 3585 9233
+ 64.135714 16 0 tcp 1040 ------- 1 16.0 12.0 3587 9234
- 64.135714 16 0 tcp 1040 ------- 1 16.0 12.0 3587 9234
r 64.138024 0 12 tcp 1040 ------- 1 16.0 12.0 3586 9229
+ 64.138024 12 0 ack 40 ------- 1 12.0 16.0 3586 9235
- 64.138024 12 0 ack 40 ------- 1 12.0 16.0 3586 9235
r 64.140173 0 16 ack 40 ------- 1 12.0 16.0 3585 9233
r 64.145797 16 0 tcp 1040 ------- 1 16.0 12.0 3587 9234
+ 64.145797 0 12 tcp 1040 ------- 1 16.0 12.0 3587 9234
- 64.145797 0 12 tcp 1040 ------- 1 16.0 12.0 3587 9234
r 64.148027 12 0 ack 40 ------- 1 12.0 16.0 3586 9235
+ 64.148027 0 16 ack 40 ------- 1 12.0 16.0 3586 9235
```

图 3.2NS2 的数据文件

在节点的记录文件中，第一个字符的含义是节点的动作，其中 r 表示接收到数据包，+表示数据包生成并且进入链路的队列，-表示数据包从链路队列中退出，

d 该数据包被丢掉。

第二个字符的含义是该动作发生的时间，第三第四个字符是该数据包发送的源地址和目的地的，下一个字符是数据包的类型，再其后是数据包大小。

在数据记录的后半段则是数据包的内容。

得到了数据记录文件之后，就可以通过 NS2 组件中的 gawk 语言，来进行数据分析。

3.3 网络比较实验的结果和分析

3.3.1 丢包率

对于一个网络来说，任意两点之间的数据传输的丢包率是一个比较重要的数据，为了具体测试新的网络结构和 Dcell 之间的性能提升和区别，我先设计了两个实验。

实验一是一个基本的对比实验，对于整个网络来说，数据生成程序本身将产生随机的数据，但是对于整个网络来说，丢包可能产生在任何节点，丢包的原因也有很多，为了保证两个网络之间的性能比较不被数据的随机性影响，并且确定整个实验中，不会因为除了队列过长而产生的丢包以外的不确定性，首先我将整个网络的所有数据节点都关闭，然后打开其中距离最远的两个节点，各进行了长时间的随机数据发送。由于设计的链路带宽比较大，基本保证了所有的数据包都没有丢失。

表 3.1 低负载情况下的丢包情况

	发送数目	丢失数目	丢包率
Dcell	76407	0	0
新结构	78044	0	0

同时，这个实验也带来了一些问题，当数据生成数据发送流的设定值过大时，网络必然产生大量的丢包，并且在开启多个节点之后，整个网络都会陷入拥塞，经过第一个实验之后的一些测试，我确定了将数据生成的数据流量设置在 40Mbps 左右，保证了整个网络的可用性。

同时为了保证整个网络的负载均衡情况是随机的，我设计了这样一个节点连接的方式，通过生成两个随机数表，然后将对应的节点进行连接，并且在这两个节点直接使用数据生成程序进行数据发送，经过测试，40Mbps 的数据发送速率可

以在有限时间内不产生大量的拥塞导致整个网络崩溃的情况，并且能够产生一定的数据交换的速率浮动和拥塞。以后不加说明的话，就是在整个网络中采用了这样的数据设定和连接方式。

然后为了测试整个网络在数据负载比较重的情况下的丢包情况，我开启了随机节点列表中的所有节点，并且开始了泊松分布数据流，经过一定时间的计算，得到了以下数据。

表 3.2 高负载情况下的丢包情况

	发送数目	丢失数目	丢包率
Dcell	281495	76	0.027%
新结构	286934	60	0.021%

从这里我们可以看出，新的结构比 Dcell 的丢包率要略低，证明了新的结构在数据包的丢包问题上要略好于 Dcell 结构。

3.3.2 吞吐量测试

吞吐量是衡量一个网络性能的重要参数，在一个网络中，吞吐量越大意味着网络内的数据交流越多，信息量的交换也越快。这里吞吐量的计算在本质上很难设立一个标准来评价 Dcell 结构和我们的新结构，最简单的方式是按照前面的做法，对所有的数据进行随机化的处理，然后选取其中的一个节点，对该节点的吞吐量进行对比测试。但是鉴于这两个网络在本质上并不存在能够相互比较的节点，所以在实验过程中进行了专门比对，在具体的实验论述中将一一涉及。总体来说，新的结构在绝大部分情况下还是优于 Dcell 结构的，而在特定场合中，Dcell 结构的优势，也可以通过改变新结构的 n 和 k 的值来达到接近或者超过 Dcell 结构的性能。但是另一方面，这样的结构改变本身也要根据整个网络的计算机容量来进行特异化的处理。具体的结论将在第四章进行总结。

为了测试信息的吞吐量，我设计了多种情况下的测试。

首先是，对于整个网络中，有大量的计算机之间进行并行通信，这里我测试了整个结构中的一个计算机节点的网络数据吞吐量。

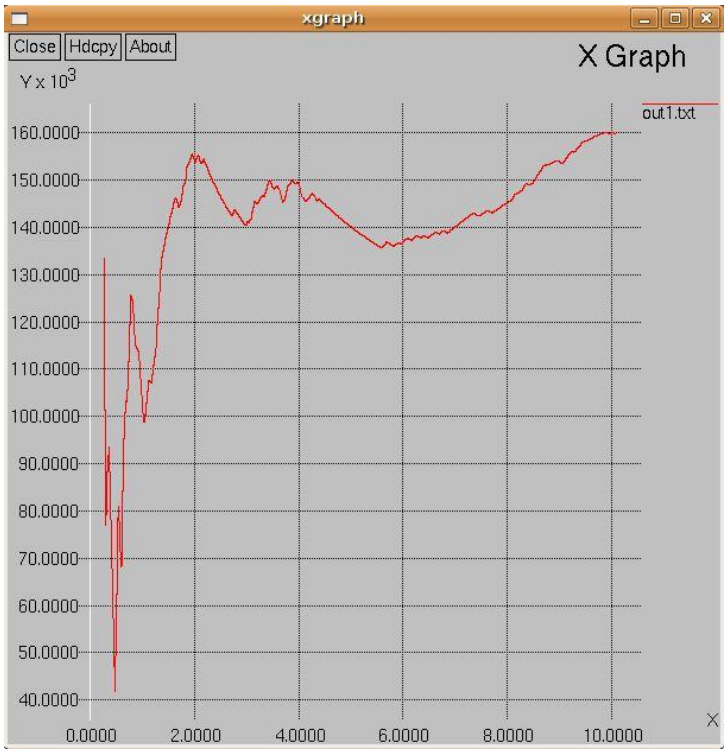

图 3.3Dcell 结构的计算机节点数据吞吐量

图 3.4 新结构的计算机节点数据吞吐量

从数据上我们可以明显看出，在整个网络中大量节点进行随机数据交换的情况下，新结构的网络吞吐量要强于 Dcell 结构，并且当 n=2 的情况下，在测试数据下，对整体的带宽利用率较高。

第二个实验主要是测量当网路中大量节点集中访问一个节点时候，网络的数据。对于我们预设的 n=2，k=3 的网络，这里的性能要比 Dcell 结构要差。

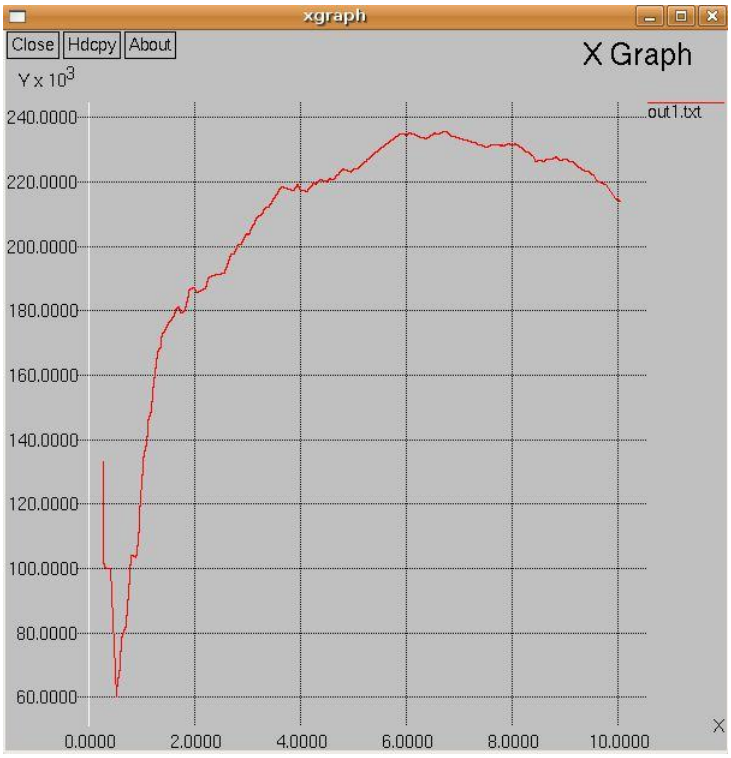

图 3.5Dcell 结构的网络吞吐量

图 3.6 新结构的网络吞吐量

这里的数据测试带来的一个比较极端的影响是，在 n=2 的情况下，网络中大量节点同时访问一个节点，对于新结构来说，如果 n 的取值比较小，那么一个计算节点对外连接的交换机数目有限，当大量并发数据到来的时候，会产生比较严重的数据拥塞现象，造成网络性能的急剧下滑。经过多次比较发现，Dcell 结构进行扩展的过程中，本身计算机节点的大量连接对于并发数据处理的一个很好的保证。

于是我开始对新结构的计算特点进行改进，经过一定的比较发现，本身新结构在扩展性和灵活性远远高于 Dcell 结构的，对于同样在 1000 台左右计算机节点的数据中心来说，Dcell 结构中，每台计算机的链路连接数大约为 4。同理，对于新结构的扩展方式来说，我们也可以选择 n=4 来进行整体架构的构造，得到了和 Dcell 基本持平并且略微有优势的数据吞吐量结构。

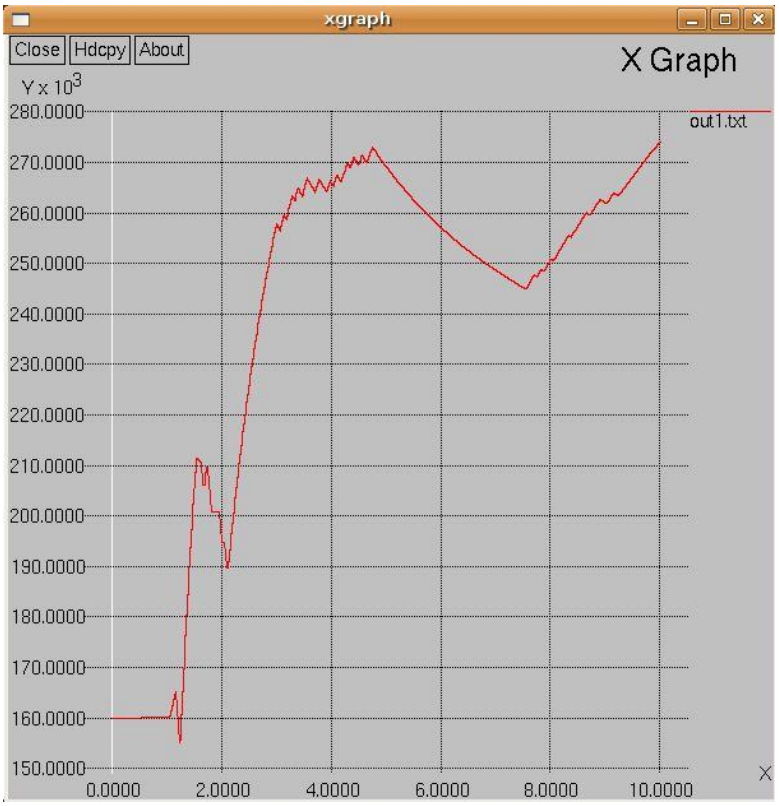

图 3.7 新结构在 n=4 时候的数据吞吐量

总体来说，新结构在进行合理调整后的性能依然优于 Dcell 结构。同时我们也可以看到，对于新结构进行调整之后，优于整体的层数减少了，在初始的大量数据交换后出现了一定程度的性能下滑，但是依旧远远超出 Dcell 结构的表现。同时我们也可以知道，在更换构造方式之后，网络的整体连接方式并没有发生质的改变，

第4章 结论

通过前面的一系列研究，我们可以看到新的网络拓扑结构在不同情况下的表现，也得到了很多结论。

对于新结构来说,从命名和连接的角度来看,便利性要远远超出 Dcell 结构，从数据中心的建立和维护的角度来说，这是一个比较具有优势的特性，在实际操作中，对于成本的控制和网络的更新和维修都带来了很大的便利。

而对于网络的基本性能，我们还是需要结合很多其他方面的因素进行考量，对于首次搜索可达步数的分析表明，对于整个网络的选择，在某种程度上来说还是要依赖数据库或者其他更上一层的应用的影响。

更进一步说，当我们直接和 Dcell 结构来进行比较的时候就会发现，不同的设计方式和不同的网络容量都会对网络的性能产生不可估量的影响，仅就一般应用而言，新的拓扑结构比 Dcell 的优势比较明显，而对于数据中心比较需要的大量数据并发操作的时候，新结构也能够过自身的连接方式的多样性而达到对比 Dcell 结构的优势。

比较针对性的分析在于，当我们使用大约 1000 台左右的计算机来组成一个更大的计算单元的时候，并且在数据库本身有一定的备份的情况下，新结构可以通过适当的连接方式，获得完全优于 Dcell 结构的性能。

插图索引

表格索引

参考文献

[1] J. Dean and S. Ghemawat, Mapreduce: Simplified dataprocessing on large clusters, ACM Commun., vol. 51,Jan. 2008, pp. 107-113.

[2] J. Ekanayake, S. Pallickara and G. Fox, MapReduce forData Intensive Scientific Analyses, DOI10.1109/eScience.2008.5

[3] S. Ghemawat, H. Gobioff, and S. Leung, The Google File System, SOSP'03, October 19–22, 2003, BoltonLanding, New York, USA.

[4] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang and S. Lu, DCell: A Scalable and Fault-Tolerant Network Structure for Data Centers, SIGCOMM'08, August 17–22, 2008,Seattle, Washington, USA.

[5] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C.Tian, Y. Zhang and S. Lu, BCube: A High Performance, Server-centric Network Architecture forModular Data Centers, SIGCOMM'09, August 17–21,2009, Barcelona, Spain.

[6] C. Jin, C. Vecchiola and R. Buyya, MRPGA: AnExtension of MapReduce for Parallelizing GeneticAlgorithms, Fourth IEEE International Conference oneScience, 2008, Indiana University, USA.

[7] A. Matsunaga, M. Tsugawa and J. Fortes,CloudBLAST: Combining MapReduce andVirtualization on Distributed Resources forBioinformatics Applications, Fourth IEEE International Conference on eScience, 2008, IndianaUniversity, USA

[8] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A.Wallach, M. Burrows, T. Chandra, A. Fikes and R. E.Gruber, Bigtable: A Distributed Storage System forStructured Data, OSDI 2006: Seattle, WA, USA.

致谢

感谢我的指导老师曹老师在论文写作过程中给予的指导，感谢张帆学长在之前对我的一系列启发和引导，感谢实验室的各位老师对我在论文写作过程中的帮助，感谢王靖远同学在NS2软件上面对我的帮助。

声明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签　名：_____　　日　期：_____

附录 A 外文资料的调研书面翻译

Dcell-一种新型数据中心可扩展的容错网络结构

摘要

在数据中心的的建设过程中有一个基础问题，那就是如何高效容错的在大量增长的计算机中完成高效的数据交换。本论文主要内容是提出以 Dcell 结构，这是一种新型的可扩展结构，有着许多数据中心网络所急需的性能特点。Dcell 是个完善的多层网络拓扑结构，其中的高层结构是由多个更低一层的小型结构所构造成的，并且在 Dcell 结构中的每个小的底层结构都能完整的和其他的底层机构进行连接。Dcell 结构随着节点的连接方式变化，总体的容量将会成倍的增长。我们的实验显示，在一个 20 服务器的 Dcell 结构中，Dcell 能够提供大约两倍于传统树形连接方式所带来的带宽，这在 Mapreduce 等应用的数据类型中体现的更明显。

第一章 综述

总的来说，我们提出了一个新的物理连接方式，满足了数据中心网络所需要的三个最重要的性能特点。本文的主要贡献已经被模拟仿真和实验所确认了，在我们结构中的一个潜在缺陷就在于，当 Dcell 的链路流量主要从高层流向结构低层，所使用交换机和路由器的资源要比原来的树形结构要多。然而我们可以从另一个角度看待这个问题，这些资源的使用其实从另一个方面看是多路适应和容错路由所必须的。

论文的剩余部分大致如下，第二章提出了数据中心网络所需要的一系列特点。第三章描述了 Dcell 结构的基本内容。第四第五章描述了 Dcell 结构的路由和当网络增长的情况下所反应的性能优势。第六第七章用了模拟和仿真来提出了 Dcell 结构的改善。第八章的时候将 Dcell 结构和其他的相似工作进行了比较。最后一章总结了整个论文的内容。

第二章 数据中心的网络结构

当前的数据中心使用廉价计算机和简易交换机来代替原有的高性能服务器和特殊设的路由连接，从而实现了非常好的性能价格比。现有的数据中心网络实践是使用树形连接来实现数据中心中大量计算节点之间的连接，在属性结构的最底层，服务器节点分布在同一个机柜中（一般是 20-80 台计算机），并且连接到一个机柜交换机。再下一层中，机柜交换机连接到了一个核心交换机中，每个核心交换机都连接到几百个机柜的机柜交换机中。一个两层的树形结构足够支持近千台服务器。为了维护数据节点的数量的大量增长，更多的树形层需要被加入进来，

需要使用更多更贵的高速交换机。

属性网络结构的缺点有两个，首先，服务器一般上都是两层结构。其次，核心交换机，当然也包括机柜交换机，是整个网络非常明显的平静。在树形结构中，同时也会有一个单链路的错误会导致很大的问题，那就是，核心交换机一旦崩溃，将会导致几千台计算机的链路被切断。虽然增加交换机的数目可以改善这样的问题，但是整体的结构性特点使得这个问题是无法避免的。

为了解决数据中心的网络问题，看上去，我们需要使用一些并行计算中已经使用过的常见结构。这些结构将不同的 CPU 通过并行连接连接成为一个超级计算机，这包括 Mesh 结构，Torus 结构，超立方结构，Fat Tree 结构和蝴蝶型结构。然而，这些结构在某种程度上说，不但是并行计算中比较有效的办法，而且能够满足数据中心网络结构的一系列需求，我们在第八章将论述这些问题。

第三章　Dcell 结构

Dcell 结构有四个特点来满足数据中心网络结构的一系列要求，Dcell 通过四个不同的角度来完成这样的要求。Dcell 结构的可扩展性、有效性和分布式的路由算法，使得 Dcell 结构的效率非常高，同时也避免了链路、服务器或者是机柜路由器这样的链路失效问题带来的整个网络的崩溃，而且我们可以通过很简单的方式来拓展 Dcell 的结构。

3.1 Dcell 的物理链路

Dcell 结构使用多端口的计算机节点和迷你交换机来组成整个结构，在 Dcell 中，每个服务器节点连接到另外几个服务器节点和一个迷你交换机上，并且这中间的连接是双向链路，高层的 Dcell 结构是由几个相对低层的 Dcell 结构来组成的。我们用 Dcellk（k>=0）来组成一个 k 层的 Dcell 结构，在下面的例子中我们将具体解释 Dcell 结构的构造方式。

Dcell 的最基层结构是用来构建更高层结构的基础，其中含有 n 个服务器节点和一个迷你交换机节点。在最底层的结构中，所有的服务器节点都连接到同一个交换机节点上。在我们的设计中，n 是一个自然数，从这个角度来考虑，我们一般设计 n 不大于 8，并且交换机是一个 8 口交换机，链路速率保持在 1G/s 到 10G/s 之间。

一个更高层的 Dcell 结构是由 n+1 个 Dcell 的基层结构来构建的，每个 Dcell0 结构都通过双向链路和其他的 Dcell0 结构连接到一起，在实例 1 中，我们取 n=4，在每个 Dcell1 结构中，都包含了 5 个 Dcell0 结构。

3.2 Dcell 的结构特点和构造

特性 1 当我们取 n=4 的时候，每个 Dcell1 节点中含有 5 个 Dcell0 结构，如

果我们把 Dcell0 结构看做一个虚拟节点,那么每个 Dcell1 结构都是一个完全图。

特性 2 Dcell 的构建结构是一个可控的递推过程

构建 Dcell 的伪代码如下：

```
/* pref is the network pre¯x of DCelll
l stands for the level of DCelll
n is the number of nodes in a DCell0*/
BuildDCells(pref, n, l)
Part I:
if (l == 0) /*build DCell0*/
for (int i = 0; i < n; i + +)
connect node [pref; i] to its switch;
return;
Part II:
for (int i = 0, i < gl; i + +) /*build the DCelll¡1s*/
BuildDCells([pref; i], n, l ¡ 1);
Part III:
for (int i = 0, i < tl¡1; i + +) /*connect the DCelll¡1s*/
for (int j = i + 1, j < gl; j + +)
uid 1 = j ¡ 1; uid 2 = i;
n1 = [pref; i; uid 1]; n2 = [pref; j; uid 2];
connect n1 and n2;
return;
```

3.3 Dcell 的数学特性

Dcell 结构是一个超立方结构的具体实现，当维度不断增加时，我们的计算机数目实际上并不是线性或者指数增加的，而是比指数增加的速度还要快。

其中 Dcell 结构中边的数目 tk 值如下：

$$(n+\frac{1}{2})^{} \quad -\frac{1}{2} < t_k < (n+1)^{} \quad -1$$

而在最早的 Dcell 结构中,计算机节点的总数目可以由完全图的递推来得出，当一层的节点数目是 tk-1 的时候，那么我们可以在其上的一层构建 gk 个对应的 Dcell 更高一层的结构，那么其中计算机的总数为 tk，递推方法如下

$$t_k = \quad g_k \times t_{k-1}$$

通过这样的递推我们可以知道，Dcell 结构是一个可以高速增长，并且包含大量计算节点的数据中心的拓扑结构，举例来说，当 k=3，n=6 的时候，整个 Decll 结构中将有 326 万个计算机节点，这也就意味着，只要很简单的基层结构，就可以将现有的数据中心中所有的计算节点包含在内。

同时我们也可以知道。Dcell 结构中的对剖平面数目为

$$\frac{t_k}{4\log_n t_k} \; for \; k > 0.$$