软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2017,28(8):2196−2213 [doi: 10.13328/j.cnki.jos.005119]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

一种构建超大规模数据中心的模块化网络结构*

陆菲菲[1,2], 谢向辉[1], 郭得科[3], 朱桂明[1]

[1](数学工程与先进计算国家重点实验室,江苏 无锡 214125)

[2](解放军信息工程大学 国家数字交换系统工程技术研究中心,河南 郑州 450002)

[3](国防科学技术大学 信息系统工程重点实验室,湖南 长沙 410073)

通讯作者: 陆菲菲, E-mail: lu.feifei@meac-skl.cn

摘 要: 模块化数据中心网络的模块间互联结构和路由负责模块的有效组织以及不同模块服务器间的高效通信,使得如何设计具有高带宽、高容错和高可扩展能力的互联结构以支持大规模、超大规模数据中心的构建,成为模块化数据中心网络需要解决的首要问题.提出了一种构建超大规模模块化数据中心的模块间互联结构 MDKautz.该结构通过模块内大量未被使用的交换机预留高速端口将模块以 Kautz 图互连,在无需额外增加任何高端交换设备的前提下,构造出具有高带宽、高容错和灵活可持续扩展性的超大规模数据中心网络.对 MDKautz 的构建方法、路由策略以及扩展方法进行了分析.数学分析和模拟实验结果证明了该新型网络结构具有良好的拓扑特性和通信性能,可有效支持数据中心高带宽、高容错的典型应用.

关键词: 超大规模;模块化数据中心;以服务器为中心;并行多路径;通信模式

中图法分类号: TP393

中文引用格式: 陆菲菲,谢向辉,郭得科,朱桂明.一种构建超大规模数据中心的模块化网络结构.软件学报,2017,28(8):2196−2213. http://www.jos.org.cn/1000-9825/5119.htm

英文引用格式: Lu FF, Xie XH, Guo DK, Zhu GM. Modular network structure for building mega-modular data center. Ruan Jian Xue Bao/Journal of Software, 2017,28(8):2196−2213 (in Chinese). http://www.jos.org.cn/1000-9825/5119.htm

Modular Network Structure for Building Mega-Modular Data Center

LU Fei-Fei[1,2], XIE Xiang-Hui[1], GUO De-Ke[3], ZHU Gui-Ming[1]

[1](State Key Laboratory for Mathematical Engineering and Advanced Computing, Wuxi 214125, China)

[2](National Digital Switching System Engineering & Technological Research Center, PLA Information Engineering University, Zhengzhou 450002, China)

[3](Science and Technology on Information System Engineering, National University of Defense Technology, Changsha 410073, China)

Abstract: The modular data center networks comprise inter- and intra-module networks. The structure and routing of inter-module networks take charge of organizing modules, and communicating among servers in different modules. Thus, it is the most important issue to design an inter-module structure with high bandwidth, high fault-tolerance, and flexible scalability to build the mega-modular data center. In this paper, an inter-module network structure called MDKautz is proposed for building the mega-modular data center. MDKautz uses Kautz as the basic interconnection topology among modules, and builds the mega-modular data center network with high bandwidth, high fault tolerance and good scalability without the need of any additional high end switches. The method for how to constructing, routing and expanding in MDKautz are analyzed. Mathematical analysis and experiment results show that MDKautz has a good topology characteristic and communication performance, and can effectively support the typical applications with high bandwidth and high fault tolerance in data center.

* 基金项目: 国家高技术研究发展计划(863)(2013AA01A21)
Foundation item: National High-Tech R&D Program of China (863) (2013AA01A21)
收稿时间: 2014-02-18; 修改时间: 2014-08-27, 2015-12-30; 采用时间: 2016-04-30; jos 在线出版时间: 2016-10-11
CNKI 网络优先出版: 2016-10-12 16:26:49, http://www.cnki.net/kcms/detail/11.2560.TP.20161012.1626.015.html

Key words: ultra-large scale; modular data center; server-centric; parallel multi-path; traffic pattern

随着 IT 建设成本和用户需求的增长,越来越多的大型网络服务业,如美国谷歌、微软、亚马逊等,均利用规模经济的优势建立起百万级的云计算数据中心[1].预计未来,单个数据中心的规模将扩大到更高的量级.采用模块化的方式构建超大规模数据中心,是当前的一个发展趋势.模块化网络的基本构建原理为:首先,将一定规模的服务器按照特定拓扑结构互连并置入模块内(如集装箱),形成基本的构建模块;然后,将大量这样的模块再进一步通过模块间的特定拓扑结构进行互连,构造出更大规模的数据中心网络.由于模块化数据中心(modular data center,简称 MDC)的模块内、外结构可以分开设计,因此极大地降低了超大规模数据中心构建和维护的复杂度.然而,当前对于数据中心网络结构的研究主要集中在单个数据中心网络的大规模构造以及 MDC 的模块内互连结构,而对于模块间应采用什么样的互连方式以支持超大规模数据中心构建的研究较少.

如何设计模块之间的互连结构是一个极具挑战性的课题,原因主要有以下 3 点.

- 第一,高带宽需求.数据中心需要支撑许多带宽密集型的应用,如分布式文件系统[2−4]、分布式执行引擎[5,6]以及一些在线数据密集型应用[7],这些典型应用往往需要在上千个模块中的成千上万台服务器上处理数据,因此,一个好的设计方案应确保在各种互连设置下模块间均具备较高的通信容量.
- 第二,平缓的性能下降(graceful performance degradation).目前提出的数据中心网络普遍采用廉价的商业硬件搭建而成,由于集装箱内的设备是工厂预制的,空间和操作上的限制使其出现故障不易在线维护,因此,模块间互连网络的设计不仅应具有高容错特性,还应确保网络设备出现故障时网络容量不会迅速下降.
- 第三,灵活、可持续的扩展性.随着数据中心网络承载的数据量不断增长,数据中心的规模持续扩大,不可避免地需要对数据中心网络进行增量部署.因此,不仅要求 MDC 网络具有可持续、可扩展能力,还要尽量减少扩展过程中对已有结构的影响.

针对上述问题,本文提出了一种适用于构建超大规模 MDC 的模块间互联结构 MDKautz(modularized data center Kautz network).MDKautz 借鉴 MDCube[8]结构的优点,利用模块中各交换机预留的高速上行端口直接连接所有模块.所不同的是,MDKautz 结构采用具有最优拓扑性质的常量度数网络结构 Kautz 图[9]将模块互连,实现模块内、外互连及路由解耦实现高带宽和高容错.MDKautz 结构很好地解决了上述 3 个挑战:第一,模块内外松耦合的互联设计和路由算法为模块之间提供了较高的聚合带宽;第二,多条模块间不相交平行多路径确保网络具有良好的容错能力;第三,采用增大网络的直径扩展网络以及增加单个模块中可用的 10Gbps 上行端口数扩展网络两种适用于不同应用场景的扩展方法,突破了传统 MDC 网络的扩展性受限于模块内可用上行端口数的限制,进一步满足构建大规模数据中心网络的需要.

本文第 1 节介绍相关工作.第 2 节介绍 MDKautz 网络的研究背景及后续分析中需要用到的重要结论.第 3 节详细阐述 MDKautz 结构的设计思路.第 4 节对 MDKautz 网络的静态性能与流量分布进行分析.第 5 节描述实验过程与结果分析.第 6 节对全文进行总结.

1 相关工作

作为数据中心与云计算的基础支撑技术,数据中心网络成为近年来学术界研究的热点.目前提出的数据中心网络结构按照服务器是否参与转发,大体可以分为两类:一类是以交换机为中心的结构,另一类是以服务器为中心的结构.在以交换机为中心的结构中,路由工作均由交换机完成,因此,交换机需要根据规模的扩展和应用需求不断升级.在以服务器为中心的结构中,路由工作部分或全部由服务器承担,交换机只提供简单的纵横式(cross- bar)交换功能.在这类结构中,服务器往往通过多个网口接入网络,而且存在大量冗余的网络链路和等价路由路径,这些因素为更好地支持数据中心内各种流量模式提供了有利条件.

以交换机为中心的结构其设计思路有两种:一种是采用全新的结构取代传统树形结构,另一种则是在原有树形结构的基础上增强其通信能力以提高二分带宽.二分带宽为将一个网络等分为两部分所需要切断的最小

链路的带宽和[10],它直接关系到拓扑结构的吞吐能力,二分带宽越大,则网络容量越高,对故障的容错能力越强.Fat-tree[11]结构实际上同构于 Clos 网络[12],能够为网络提供 1:1 的带宽收敛比(oversubscription)[11],即服务器之间实现无阻塞(non-blocking)通信,如图 1 所示.PortLand[13]和 VL2[14]都采用了普通商业交换机构建 3 层 Fat-tree 结构,不同之处在于,VL2 在 Clos 网络的高层使用了容量更高的 10GbE 交换机以减少布线复杂度.增强型树结构是在树形结构原有链路的基础上增加更快捷的链路如 60G 频段的无线链路[15]或光互连[16,17]提高网络的局部通信能力,因为对于百万级的大规模数据中心来说,全带宽通信即服务器间能够以网卡的最大速率进行通信往往很难实现也并非必须的,根据通信的局部性原则,加强通信密集区域的连接能力能够缓解热点和拥塞,从而提高网络整体吞吐量.

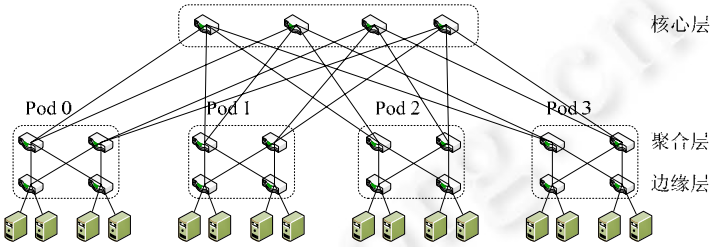

Fig.1 A Fat-tree structure

图 1 Fat-tree 结构

在以服务器为中心的结构中,服务器提供多个网络端口用于互连并参与包转发.DCell[18]是一个采用递归方式定义的多层网络结构.DCell 中的每台服务器提供多个网络端口,能够支持的服务器规模以接近 2 的指数次方扩展.DCell 结构的不足之处是网络中的流量分布不均衡,最底层网络承担了过多的流量,极大地限制了网络的最大聚合带宽.此外,DCell 的网络规模受限于服务器的网络端口数,不具备良好的持续扩展能力.FiConn[19]在 DCell 的基础上进一步将服务器的网络端口数固定为 2,提出服务器网络端口受限时的可扩展互连结构 FiConn.FiConn 与 DCell 采用相似的层次化方式构建,但每层仅使用服务器上的备用网络端口来连接,且无需增加服务器端的其他硬件成本.同时,与 DCell 结构相比,FiConn 结构极大地降低了配线成本,但节约的链接数是以损失部分网络容量为代价的,因此,FiConn 结构的链路利用率较低.

BCube[20]和 SCautz[21]是针对 MDC 设计的以服务器为中心的模块内互联结构.BCube 结构中的服务器以 Generalized hypercube[22]进行组织,具有多个网络端口的服务器连接到多个层次的小型交换机,任意两个服务器之间没有直接连接.该结构的最大优势是链路资源非常丰富且流量分布较均衡,如图 2 所示.但 BCube 必须分配给每个服务器更多的 NIC 端口,一般最多为 4 个,因此,BCube 网络规模的扩展也受限于服务器的网口数.SCautz 结构通过服务器的网络端口和交换机的少量冗余端口将网络连接成一个 Kautz 图,SCautz 具有丰富的并行路径,能够很好地支持数据中心的多模式通信.但 SCautz 对服务器网络端口数的需求量较大,从而其网络规模的扩展也受限于服务器的网口数.如,构建一个规模仅为 1280 的网络需要每台服务器提供 10 个网络端口.

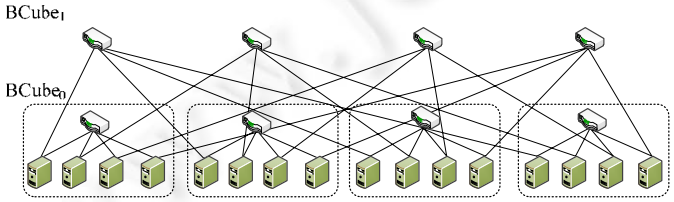

Fig.2 A $BCube_1$ structure

图 2 $BCube_1$ 结构

MDCube 和 uFix[23]是针对 MDC 网络模块间互连设计的结构.uFix 是针对异构集装箱容器互连提出的网络结构.它通过模块内服务器预留的网络端口,将采用不同互连结构的模块连接成 intensive mesh 结构,要求每个模块至少预留一定数量的服务器空闲端口进行模块之间的互连,但模块间的连线个数取决于模块内预留的网络端口数量,使得模块间的链路成为整个网络的带宽瓶颈,从而限制了整个网络的性能和可靠性.与本文工作最接近的是 MDCube 结构,如图 3 所示.MDCube 是一个多维的拓扑结构,用于连接采用 BCube 构建的数据中心集装箱.它可以连接的数据中心集装箱的个数是所有维度上可容纳的集装箱数量的乘积.但是,MDCube 是针对模块内结构 BCube 设计的模块间互连结构,并通过模块内、外连接和路由的紧耦合实现高带宽和高容错.且 MDCube 的相邻模块间仅靠一对交换机的高速端口连接,模块间可用的通信带宽有限且可靠性较差.同时,其扩展性也受限于模块中可用的高速端口数.

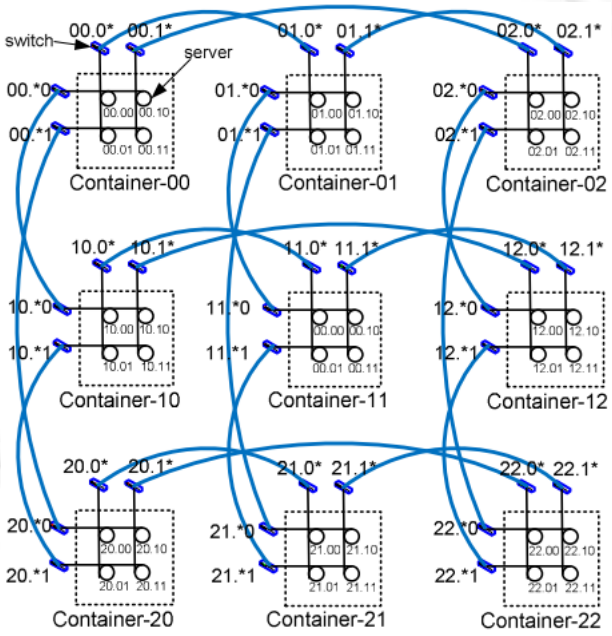

Fig.3 A 2-D MDCube structure

图 3 2-D MDCube 结构

2 研究背景

MDKautz 对模块内的拓扑结构不做特殊要求,广泛适用于各类新型结构,只要模块内具有充足的 10G 上行带宽.本文以 CH 结构[24]为例.CH 结构是我们提出的一种常量度数的、以服务器为中心的新型互连结构,由固定网络端口个数的服务器和廉价的商用交换机搭建.由于他具有良好的可扩展性、容错性和较高的网络容量,且随着网络故障率的升高,其性能下降非常平缓,因此,选择 CH 结构作为构建超大规模数据中心的模块内互连结构是合适且可行的.本节主要介绍 CH 结构及其拓扑特性并给出 Kautz 图的定义及相关结论.

2.1 CH结构

CH 结构采用递归定义的层次设计,其基本构件为具有固定网络端口的服务器和交换机.CH 结构定义为 $C(n,m)$,其中,$n(n\geq 1)$表示服务器端口个数,$m(m\geq 0)$表示 CH 结构的层次.$C(n,0)$由一台具有 n 个网络端口的服务器连接 n 台交换机构成;$C(n,m)(m\geq 1)$由 n^m 台具有 n 个网口的服务器连接 n 个 $C(n,m-1)$构成.定义 $C(n,m)$由编号为 $0\sim n-1$ 的 n 个 $C(n,m-1)$组成,每个 $C(n,m-1)$中包括 n^m 台交换机.则 $C(n,m)$的构建方法为:将第 m 层中第 $i(i\in[0,n^{m-1}])$台服务器的第 $j(j\in[0,n-1])$个网口与第 j 个 $C(n,m)$中的第 i 台交换机相连,如图 4(a)所示.

CH 网络中的节点(包括交换机和服务器)用唯一的 m 元组定义.服务器的编号定义为 $s_ms_{m-1}\dots s_{i+1}ns_{i-1}\dots s_1s_0(s_i\in[0,n],i\in[0,m])$,交换机的编号定义为 $x_mx_{m-1}\dots x_i\dots x_1x_0(x_i\in[0,n-1],i\in[0,m])$.其中,$n$ 所在的位标识了服务器所在的层次.编号为 $s_ms_{m-1}\dots s_{i+1}ns_{i-1}\dots s_1s_0$ 的服务器连接 n 台交换机,这 n 台交换机的编号仅在第 j 维不同,可表示为 $s_ms_{m-1}\dots s_{i+1}t_js_{i-1}\dots s_1s_0(t_j\in[0,n-1])$.例如,在 $n=2,k=3$ 的 $C(2,3)$结构中,服务器 2000 连接 2 台交换机,分别为 0000 和 1000,如图 4(b)所示.

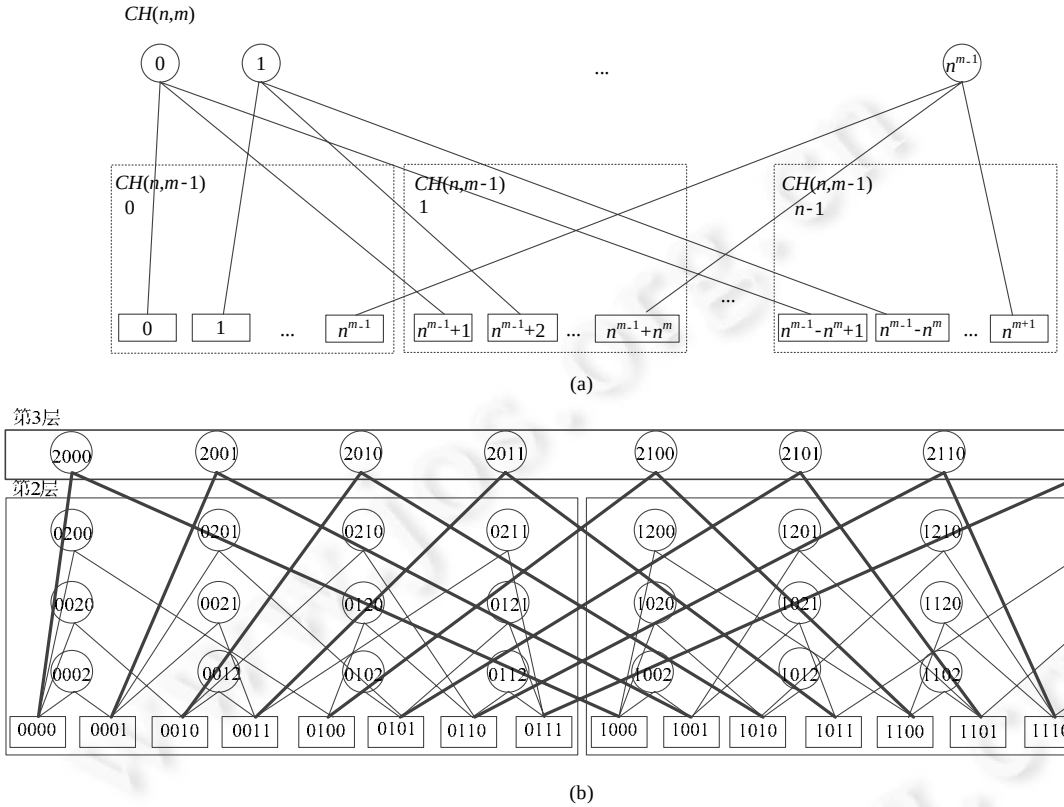

Fig.4 A design method and an example for $C(n,m)$

图 4 $C(n,m)$的构建方法与实例

CH 结构具有以下几个已经被文献[22]证明的性质.需要特别说明的是,本文中令服务器到交换机为 1 跳,因此,CH 的直径为原文中描述的 2 倍.

(1) CH 结构的直径为 $2(m+1)$.

(2) CH 结构中任意两台服务器之间有 n 条边不相交并行路径.

(3) CH 结构的网络规模为 $N=(m+1)n^m$,交换机规模为 n^{m+1}.

(4) CH 结构的二分带宽为 $\begin{cases}n^{m+1}, & n\text{为偶数}\\ n^m(n-1), & n\text{为奇数}\end{cases}$.

2.2 CH结构的其他性质

本节将论证 CH 结构的其他几个属性,这些结论主要用于后续设计 MDKautz 结构模块间的路由算法和分析 MDKautz 结构的特性.

(1) CH 结构中,任意两台交换机间的最长最短路径为 $2(m+1)$.

证明:根据 CH 结构的编号规则,编号差 1 位的任意两台交换机连接到同一台服务器,则它们之间的最短路

径为 2.由于网络中交换机的编号最多差 m+1 位,路由时通过逐位校正上一跳交换机编号中的一位建立一系列中间交换机,因此,交换机间的最长最短路径为 2(m+1). □

(2) CH 结构中,任意一台服务器与交换机之间的最长最短路径为 2m+1.

证明:从上述证明过程中很容易得出该结论. □

(3) CH 结构中,任意两台交换机之间有 m+1 条边不相交并行路径.

证明:已知 CH 结构中任意两台服务器之间有 m+1 条边不相交并行路径,根据 CHRouting 路由算法,分别连接到任意两台服务器的两台交换机之间也有 m+1 条边不相交并行路径,得证. □

(4) CH 结构中,任意一台服务器与交换机之间有 m+1 条边不相交并行路径.

证明:同上. □

2.3 Kautz图

Kautz 图由于其良好的特性被广泛应用于并行计算和 P2P 网络[25]中,因此,本文基于 Kautz 图设计 MDKautz 结构.Kautz 图记为 $K(d,k)$,其中,$d(d\geq 1)$表示节点出/入度,$k(k\geq 1)$表示直径.图 5 所示为度和直径均为 2 的 Kautz 图.Kautz 图的节点集记为 $V=\{x_k\ldots x_i\ldots x_1|x_i\in[0,d],i\in[1,k]\}$,对 $K(d,k)$中每个标识为 $x_k\ldots x_i\ldots x_1$ 的节点 V 都有 d 条出边,即对任意$\alpha\in\{0,1,2,\ldots,d\}-\{x_1\}$,节点 V 都有一条到节点 $V'=\{x_{k-1}\ldots x_i\ldots x_1\alpha\}$的出边.

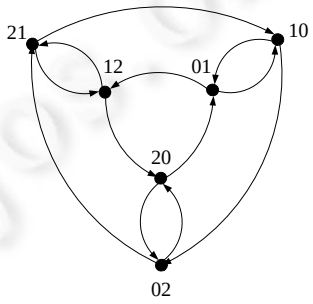

Fig.5 A Kautz with d=2, k=2

图 5 Kautz 图 K(2,2)

Kautz 图具有如下已被证明的性质.

(1) $K(d,k)$的出、入度均为 d.

(2) $K(d,k)$的节点规模为 d^k+d^{k-1},边的数量为 $d(d^k+d^{k-1})$.

(3) $K(d,k)$的二分宽度为 $\frac{(d+1)d^k}{2k}$.

3 MDKautz 结构

MDKautz 采用 CH 结构作为模块内互连结构,并采用常量度数的网络结构 Kautz 图作为模块间的互连结构以支持超大规模数据中心.然而,将数据中心网络中的数百个模块连接起来,不仅需要设计合适的网络结构,还要设计合理的连接方式和路由策略.首先,MDC 网络将模块看作黑盒以适应各类模块内互连结构,因此,要求模块间的互连和路由与模块内部无关;其次,应确保模块内部能够提供足够的空闲端口用于更大规模的网络互连,即提供可持续扩展能力;再次,模块之间应该有足够的链路,以提高网络的容量和可靠性;最后,应合理分配模块提供的端口,尽量使端口的使用率较为平均,确保当通信的源端和目的端过于密集的情况下,流量能够被分流至不同的端口上,从而达到均衡流量、减少热点的目的.从本节开始,将逐一解决上述问题.

3.1 构建方法

MDKautz 利用模块内大量未被使用的交换机预留的高速端口、采用 Kautz 图将 CH 网络互连起来.

MDKautz网络中,交换机仅作为交叉结构连接模块内部和模块之间的服务器,因此,MDKautz是以服务器为中心的网络结构.每个交换机提供 1 个高速端口作为 CH 模块对外连接的端口,因此,模块内交换机的数量即为该模块对外提供的端口的数量.根据CH结构的性质(3)和Kautz图的性质(1)得出 $d=\frac{1}{2}n^{m+1}$,故将MDKautz网络记为 $M(n,m,k)$.

在MDKautz网络中有两种类型的链路,即,模块内交换机与服务器之间互连的本地普通链路和模块间交换机之间互连的远程高速链路,因此,每个交换机由其所属 CH 模块的 ID 及其在模块中的 ID 共同标识,即交换机节点(x,y)表示为$\{(x,y)|x=x_k\dots x_i\dots x_1,y=y_m\dots y_j\dots y_0,i\in[1,k],j\in[0,m]\}$,其中,$x$ 表示 CH 模块的 ID,y 表示交换机在 CH 网络中的 ID.$M(n,m,k)$中,远程链路的起点和终点分别连接到不同 CH 模块内相应的交换机节点对上,连接规则为:首先,将 CH 模块中的所有参与互连的交换机节点随机划分成数量相等的两部分,分别作为远程链路的出、入节点,并将他们的ID分别按升序排列;然后,对远程出、入链路分别进行排序,根据Kautz图的定义,将模块 $x_k\dots x_i\dots x_1$ 到模块 $x_{k-1}\dots x_i\dots x_1\alpha$的远程出链路定义为第 i 条出链路,其中,$i=\begin{cases}(\alpha-x_k+d+1)\bmod(d+1), & x_k\neq x_1\\(\alpha-x_k+d)\bmod(d+1), & x_k=x_1\end{cases}$,模块 $x_k\dots x_i\dots x_1$ 的第 i 条出链路也是相应模块 $x_k\dots x_i\dots x_1\alpha$的第 i 条入链路,因此,每个模块的 $\frac{1}{2}n^{m+1}$ 条出链路和入链路也可以分别按升序排列;最后,将任意模块的第 i 条出链路替换为两个模块间的远程连接,即一个模块的第 i 个出节点到相应模块的第 i 个入节点之间的远程连接.

MDKautz 的构建算法如算法 1 所示.

算法 **1**. *BuildMDKautz*($x,x',swid_{out},swid_{in}$).

输入:节点 $x=x_kx_{k-1}\dots x_i\dots x_1$,数组 $swid_{out}[d]$和 $swid_{in}[d]$分别存放按升序排列的输出、输入交换机节点编号.

输出:MDKautz 拓扑.

```
1.  for (i=0; i≤d^k+d^(k−1); i++)
2.      visitedcid=0;                          //设置所有模块未访问
3.  cid_1=x_k x_(k−1)…x_i…x_1;
4.  EnQueue(Q,cid_1);
5.  While (!QueueEmpty(Q))
6.      DeQueue(Q,cid_1);
7.      if (!visited cid_1)                    //cid_1 未访问
8.          visited cid_1=1;                   //设置当前模块访问过
9.          for (i=0; i≤d; i++)
10.             if (i≠x_k)
11.                 cid_2=cid_1 左移一位最右补 i;
12.                 if (!visited cid_2)        //cid_2 未访问
13.                     EnQueue(Q,cid_2);
14.                 if (x_k≠x_1)
15.                     j=(i−x_k+d+1) mod (d+1);   //cid_1 的第 j 条出边
16.                 else
17.                     j=(i−x_k+d) mod (d+1);     //cid_1 的第 j 条出边
18.                 swid_1=swid_out[j];        //cid_1 的第 j 个输出交换机
19.                 swid_2=swid_in[j];         //cid_2 的第 j 个输入交换机
20.                 connect (cid_1,swid_1) and (cid_2,swid_2);
21. return;
```

以图 6 为例,实线和虚线分别表示每个模块的第 0 条和第 1 条出边,对于模块 21 而言,它到 12 之间的远程

出链路为第 0 条出链路,则交换机节点(21,00)为模块 21 的第 0 个出交换机,交换机节点(12,01)为模块 12 的第 0 个入交换机;模块 21 到 10 之间的远程出链路为第 1 条出链路,则交换机节点(21,11)为模块 21 的第 1 个出交换机,交换机节点(10,10)为模块 10 的第 1 个入交换机.

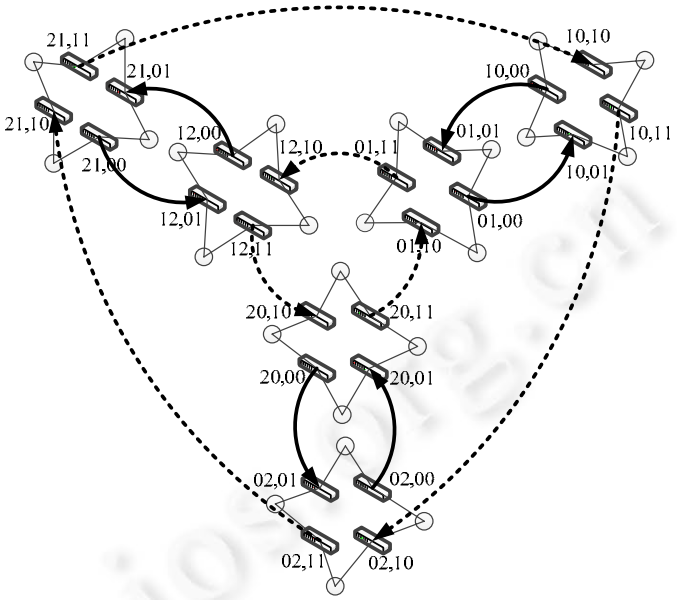

Fig.6 A MDKautz with m=2, n=1, k=2

图 6 MDKautz 结构 M(2,1,2)

3.2 单路径路由

根据 $M(n,m,k)$结构的构建规则,建立分级路由策略,即模块内部路由采用 CH 结构的路由算法,模块间的路由采用 Kautz 图的最短路径路由算法,从而得到 $M(n,m,k)$结构的单路径路由算法,如算法 2 所示.

算法 **2**. *MDKautzRouting*(*src*,*dst*).

输入:源节点 $src=x_kx_{k-1}\ldots x_1$,目的节点 $dst=y_ky_{k-1}\ldots y_1$,二者均定义为{*cid*,*sid*};

输出:路径 *path*.

```
1.  c1=src; c2=dst; path=()
2.  pref=CommSuffixPreffix(c1.cid,c2.cid);          //c1 的后缀与 c2 的前缀的共同部分
3.  l=Length(pref);
4.  if (l==k)                                       //源节点和目的节点在一个模块内
5.     path=CHRouting(c1,c2);                       //CH 单路径路由算法
6.  else
7.     for (i=0; i<k–l; i++)
8.        c2.cid=c1.cid 左移一位最右补 y_{k–l–i};
9.        (sw1,sw2)=GetLink(c1.cid,c2.cid);        //连接两个模块的交换机对
10.       path1=CHRouting(c1,sw1);                 //CH 中服务器与交换机之间的路由
11.       path=path+path1+(sw1,sw2);
12.       c1=c2;
13. if (c2.cid=dst.cid)                             //同属于一个模块
14.    path1=CHRouting(c2,dst);
```

15. *path*=*path*+*path*1;

16. return *path*;

函数 GetLink 返回算法 1 定义的连接到两个不同模块 *c*1.*cid* 和 *c*2.*cid* 的交换机对,具体路由方法通过 *c*1.*cid* 的第 *i* 个出节点到 *c*2.*cid* 的第 *i* 个入节点之间的链路实现,其中, $i=\begin{cases}(\alpha-x_k+d+1)\bmod(d+1), & x_k\neq x_1\\(\alpha-x_k+d)\bmod(d+1), & x_k=x_1\end{cases}$.模块内服务器间或交换机间的路由采用 CH 结构的路由算法,尽管 CH 结构仅定义了服务器之间的路由算法,交换机之间的路由仍可以通过与其直接相连的服务器采用相同的算法实现.

仍以图 6 为例,假设源节点 *src*=(21,21),目的节点 *des*=(20,02).源节点所在模块 ID *src*.*cid*=21,目的节点所在模块 ID *des*.*cid*=20,共同前/后缀长度为 0,因此,二者不在同一模块内,得出源节点的下一跳模块为 *tmp*.*cid*=12,计算出连接两个模块的交换机对为(21,00)和(12,01),从而得出路由路径为

path={(21,21),(21,00),(12,01),(12,11),(20,10),(20,02)}.

3.3 平行多径路由

当网络流量均衡时,单路径路由能够满足通信的需要.但数据中心的流量往往并不均衡,常常伴随突发流量.在突发流量模式下,由于单路径路由算法总是选择最短路径可能导致低效的网络带宽,从而降低数据中心网络的性能.因此,我们通过设计并行多径路由平衡模块间流量,加速端到端的传输,提高路由的容错能力,从而使整个网络达到高吞吐量.

定理 **1**. MDKautz 网络中,任意两台服务器之间存在 *d* 条中间模块不相交路径,中间模块不相交路径指一条路径上的中间模块不重复出现在另一条路径上.

证明:设源服务器所在模块的 ID 为 $a=a_ka_{k-1}\dots a_{l+1}a_l\dots a_1$,目的服务器所在模块的 ID 为 $b=b_kb_{k-1}\dots b_{k-l+1}b_{k-l}\dots b_1$,他们 ID 重叠部分的长度为 *l*.要证明从源服务器到目的服务器之间存在 *d* 条中间模块不相交路径,只需证明从 *a* 到 *b* 存在 *d* 条中间模块不相交路径.根据 Kautz 图的定义,节点 *a* 有 *d* 条出边,则 *a* 有 *d* 个不相同的下一跳节点,从这些节点到 *b* 可生成 *d* 条路径.若从 *a* 到 *b* 的一条路径 $a=a_k\dots a_{l+1}a_l\dots a_1\to a_{k-1}\dots a_{l+1}a_l\dots a_1b_{k-l}\to\dots\to a_l\dots a_1b_{k-l}\dots b_1=b$ 用$\langle a_k\dots a_{l+1}a_l\dots a_1b_{k-l}\dots b_1\rangle$表示,则这 *d* 条路径可以表示为以下 3 种形式.

I. $\langle a_k\dots a_{l+1}a_l\dots a_1b_{k-l}\dots b_1\rangle$;

II. $\langle a_k\dots a_{l+1}a_l\dots a_1\alpha b_{k-l}\dots b_1\rangle$;

III. $\langle a_k\dots a_{l+1}a_l\dots a_1\beta\chi b_{k-l}\dots b_1\rangle$,

其中,$\alpha\neq\beta\neq\chi\cap\alpha\neq a_1\cap\alpha\neq b_k\cap\alpha\neq b_{k-l}\cap\beta\neq a_1\cap\chi\neq b_k\cap\chi\neq b_{k-l}\cap\chi\neq a_{l+1}$.

下面证明这 *d* 条路径为边不相交路径.因为 $\alpha\neq\beta\neq b_{k-l}$,所以这 3 条路径从节点 *a* 出发的下一跳节点均不相同;同理,节点 *b* 的上一跳节点也都不相同.令 $n_{\mathrm{I}},n_{\mathrm{II}},n_{\mathrm{III}}$ 分别为路径 I~路径 III 上的任意一个中间节点,则有:

$$n_{\mathrm{I}}=a_z\dots a_{l+1}a_l\dots a_1b_{k-l}\dots b_{z-l+1},$$
$$n_{\mathrm{II}}=a_m\dots a_{l+1}a_l\dots a_1\alpha b_{k-l}\dots b_{m-l+2},$$
$$n_{\mathrm{III}}=a_p\dots a_{l+1}a_l\dots a_1\beta\chi b_{k-l}\dots b_{p-l+3},$$

其中,$z,m,p\in[1,k]$.

假设路径 I 和路径 II 不是边不相交的,那么 $n_{\mathrm{I}}=n_{\mathrm{II}}$,得出 $a_z=a_m\cap a_1=a_2\cap\alpha=a_1$,与原结论矛盾.因此,路径 I 和路径 II 是边不相交的.同理可以证明路径 I 和路径 III、路径 II 和路径 III 都是边不相交的,原命题得证. □

注意到,定理 1 中描述的中间模块不相交路径并非真正的边不相交(edge-disjoint)路径,即一条路径上的中间节点(服务器和交换机)不重复出现在另一条路径上.因为源模块中从服务器到 *d* 条输出交换机的路径上节点一定有重复,目的模块中,从 *d* 条输入交换机到服务器的路径上节点也有重复,因此,源模块和目的模块有可能成为整个网络流量的瓶颈,在第 4.2 节中有详细论证.

算法 3 描述了构造 *d* 条模块不相交路径的过程.该算法生成的路径最多经过 *k*+1 个中间不相交模块,仅比最短路径算法多经过 2 个模块.

算法 **3**. *MDisjointRouting*(*src*,*dst*).

输入:源节点 *src*,目的节点 *dst*,二者均定义为{*cid*,*sid*}.

输出:路径集 *pathSet*.

1. *c*1=*src*; *c*2=*dst*; *path*={}; *counter*=0;
2. *pref*=*CommSuffixPreffix*(*c*1.*cid*,*c*2.*cid*);
3. *l*=*Length*(*pref*);
4. if (*l*==*k*)
5. *pathSet*=*CHParallelRouting*(*c*1,*c*2); //CH 平行多径路由算法
6. else
7. P_1=*MDKautzRouting*(*src*,*dst*);
8. add P_1 to *pathSet*; //一条长度不大于 *k* 的最短路径
9. for (*i*=0; *i*≤*d*; *i*++) //*counter* 条长度为 *k*+1 的路径
10. if ($i\neq x_1$ && $i\neq y_k$ && $i\neq x_{l+1}$ && $i\neq y_{k-l}$)
11. out_{src}.*cid*=*c*1.*cid* 左移一位最右补 *i*;
12. in_{dst}.*cid*=*c*2.*cid* 右移一位最左补 *i*;
13. if (out_{src}.*cid*≠$uout_{src}$.*cid* && in_{dst}.*cid*≠uin_{dst}.*cid*)
14. P_2=*MDKautzRouting*(*src*,out_{src})+*MDKautzRouting*(out_{src},in_{dst})+*MDKautzRouting*(in_{dst},*dst*);
15. add P_2 to *pathSet*;
16. out_{src}.*cid*=$uout_{src}$.*cid*;
17. uin_{dst}.*cid*=in_{dst}.*cid*;
18. *counter*++;
19. if (*counter*+1≠*d*) //(*d*−*counter*−1)条长度为 *k*+2 的路径
20. for (*i*=0; *i*≤*d*; *i*++)
21. for (*j*=0; *j*≤*d*; *j*++)
22. if ($i\neq j$ && $i\neq x_1$ && $j\neq y_k$ && $i\neq y_{k-l}$ && $j\neq x_{l+1}$)
23. out_{src}.*cid*=*c*1.*cid* 左移一位最右补 *i*;
24. in_{dst}.*cid*=*c*2.*cid* 右移一位最左补 *j*;
25. if (out_{src}.*cid*≠$uout_{src}$.*cid* && in_{dst}.*cid*≠uin_{dst}.*cid*)
26. P_3=*MDKautzRouting*(*src*,out_{src})+*MDKautzRouting*(out_{src},in_{dst})+*MDKautzRouting*(in_{dst},*dst*);
27. add P_3 to *pathSet*;
28. $uout_{src}$.*cid*=out_{src}.*cid*;
29. uin_{dst}.*cid*=in_{dst}.*cid*;
30. return *pathSet*;

3.4 容错路由

单路径路由策略无法应对路由过程中节点和链路出现故障的情况,容错路由算法则能够在处理链路故障的同时,有效利用网络中的并行路径获得高聚合吞吐量.利用 MDKautz 结构具有多条模块不相交路径的特性,设计了一种分级的容错路由算法,将模块内和模块间的路由解耦,模块内由 CH 容错路由算法维护,利用 CH 容错路由算法处理模块内部的路由失效;模块之间的路由路径由源节点决定,利用 MDKautz 平行多径路由算法处理模块间的路由失效并均衡整个网络的流量.

该容错路由算法采用松散源路由的方式实现,如算法 4 所示.其基本思想是:源节点在并行路径中任选一条作为路由路径,并将经过的中间模块的 ID 存储到数据包头中.中间节点不参与路由选择,只基于包头中的 ID 转发相应的数据包,并判断下一跳是否可达.如果模块间的链路或连接模块的交换机节点失效,则向源节点发送路

由错误消息,源节点收到消息后,利用算法 3 重新选择一条并行路径,算法 3 中的模块不相交并行路径恰好能够确保新选择的路径不经过失效链路.模块内部失效的节点或链路直接由 CH 容错路由算法处理,而无需通知源节点.

算法 **4**. *MDFTRouting*(*src*,*dst*).

输入:源节点 $src=x_kx_{k-1}\ldots x_1$,目的节点 $dst=y_ky_{k-1}\ldots y_1$,二者均定义为{*cid*,*sid*}.

输出:路径 *SelectPath*.

源节点:

```
when a pkt arrives:
1.    SelectPath={};
2.    pathSet=MDisjointRouting(src,dst);
3.    path=pathSet.remove();                    //任意选择一条路由路径
4.    SelectPath.add(path);
5.  when a path failure msg received:
6.    path=BFS(pathSet,SelectPath);             //宽度优先搜索另一条并行路径
7.    if (path exists)
8.       SelectPath.add(path);
9.    else
10.      return error;
11.   return SelectPath(path);
```

中间节点:

```
when a pkt is received:
1.    if (next hop is not available)
2.       Send path failure msg to src;
3.       return;
4.    Forward(pkt);
```

注意到,模块内的路由完全由 CH 的容错路由算法决定,因此,如果有多个数据流经过同一个模块,不对流加以标识则可能产生包乱序;再者,模块间的链路可能被多个数据流占用,也可能发生包乱序.因此,我们在源节点中用 *flow_id* 标识数据包,后续的路径选择则基于源、目的节点 ID 和 *flow_id* 这 3 个参数进行.

3.5 扩展性分析

MDKautz 网络模块间的互连结构为 Kautz 图,利用 Kautz 图的特性,如果将构成 MDKautz 网络的每个模块看作一个虚拟节点,则 MDKautz 网络具备两种适用于不同应用场景的扩展方式.

- 第 1 种扩展方式通过固定单个模块的度数 *d*,增大网络的直径 *k* 扩展网络.

这种扩展方式利用 Kautz 图能够不改变节点度数进行扩展的特性,使网络具备无限可持续扩展的能力.但这种扩展方式的不足之处在于其渐进扩展的粒度较大,如果用Δ表示每次扩展至少需要增加的模块个数,则$\Delta=d^{k+1}+d^k-(d^k+d^{k-1})$.当 *d* 较大时,$\Delta\approx d^{k+1}$,即随着 *k* 的增加,其渐进扩展的粒度呈指数增长,且增大网络直径将导致节点编号和互连关系的改变,而重新部署网络将影响到数据中心的各类应用.针对这种情况,我们可以利用非正则 Kautz 图 Quasi-Kautz 结构[23]构建具有任意规模和常量度数的不完全 Kautz 图,使网络具备渐进扩展的能力,且仅需要少量地调整现有网络结构,具体实现可参考文献[23],本文不再赘述.

- 第 2 种扩展方式通过固定网络的直径 *k*,增加单个模块的度数 *d* 扩展网络.

这种扩展方法利用 Kautz 图的另一个特性,即 $K(d,k)$是 $K(d+1,k)$的导出子图[24],可以在不改变已有节点编号和互连关系的情况下扩展原网络结构,以满足数据中心网络的无损扩展需求.通常,数据中心网络的构建过程并非一蹴而就,初始构建网络时往往不需要连接很大规模的集装箱模块,因此可选择模块内部分可用的高速上行

端口将一定数量的模块互连起来,空闲的那部分高速端口即可用于支持后续扩展.目前,数据中心单个集装箱容器内的服务器数量为 1K~4K[1,8],对于 CH 结构,当网络规模为 1 024 时,其交换机数量为 256,MDKautz 网络至少能够支持约 16 908 288 个集装箱互连;对于 BCube 结构,当网络规模为 2 304 时,其交换机数量为 96,MDKautz 网络至少能够支持约 5 419 008 个集装箱互连.在网络建设初期,可根据实际需要确定 k 值,通过增加 d 实现对网络的持续扩展,因此,采用此扩展方法完全能够满足当前一段较长时期内的实际需求,即能够满足数据中心网络的实际可持续扩展需求.但这种扩展方式同样存在渐进扩展粒度较大的问题,仍然用Δ表示每次扩展至少需要增加的模块个数,则$\Delta=(d+1)^k+(d+1)^{k-1}-(d^k+d^{k-1})$,当 k 取 2 时,$\Delta=4d+2$,即随着模块度数的增加,其渐进扩展的粒度呈线性增长.针对这种情况,我们仍可以在扩展时采用非正则 Quasi-Kautz 结构构建不完全 MDKautz 网络,通过少量调整现有网络结构实现对模块的渐进部署.

综上所述,两种扩展方法各有利弊,部署时,可根据实际需要选择不同的扩展策略.当模块内没有多余可用上行端口时,可采用第 1 种方法扩展网络;第 2 种扩展方法则更适用于模块内尚有足够可用的上行端口的情况.

4 性能评估

4.1 MDKautz的静态性能

本节主要分析 MDKautz 的静态性能,包括网络规模、直径和二分宽度.数据中心网络的规模表示他所支持的服务器数量的最大值,反映了网络的扩展性;数据中心网络的直径表示网络中任意两台服务器之间最短路径的最大值,直径越短,网络中点对点延迟就越小,节点间的数据交换也越便捷;而二分宽度是衡量数据中心网络带宽利用率的重要指标,二分宽度越大网络容量越高,对故障的容错能力越强.

定理 **2**. $M(n,m,k)$的网络规模为 $\frac{(m+1)n^{k(m+1)-1}}{2^k}(n^{m+1}+2)$.

证明:由 $M(n,m,k)$的定义可得,CH 模块总数为 d^k+d^{k-1},每个模块包含的服务器个数为 $n^m(m+1)$,且 $d=\frac{1}{2}n^{m+1}$.

因此,网络规模为 $n^m(m+1)\times(d^k+d^{k-1})=\frac{(m+1)n^{k(m+1)-1}}{2^k}(n^{m+1}+2)$. □

分别给定 MDKautz 的模块内结构参数 n=2,m=7,MDCube 的模块内结构参数 n=32,m=1,使得两个网络中单个模块的规模相同.图 7 比较了两种网络的模块个数随相应参数的变化情况.不难看出,MDKautz 能够支持的模块个数随 k 的增加呈指数增长,其规模的扩展不受限制;而 MDCube 支持的模块,其增长率随维度的增加而降低,说明 MDCube 的规模受模块内可用上行端口的限制.相比之下,MDKautz 更具有规模和可持续扩展优势.

一般情况下,当 k=2 时,即可支持规模逾百万甚至千万的超大规模数据中心.例如,由 $C(2,7)$构建的 $M(2,7,2)$网络中,单个模块的规模为 1 024,共 16 512 个模块,则总网络规模为 16 908 288.

定理 **3**. $M(n,m,k)$网络的直径为 $2m(k+1)+3k$.

证明:由 Kautz 图的定义可知:在 $M(n,m,k)$网络中,任意两台服务器节点之间最多经过包括源、目的模块在内的 k+1 个模块,因此,模块间的远程链路最多有 k 条.根据第 2.2 节的属性(1)和属性(2)可得:源模块中,源服务器节点与任意交换机节点之间的最长最短路径为 $2m$+1;中间模块中,任意交换机节点之间的最长最短路径为 $2(m+1)$;目的模块中,任意交换机节点与目的服务器节点之间的最长最短路径为 $2m$+1.因此,任意两台服务器之间的最长最短路径即直径为 $2(2m+1)+2(m+1)(k-1)+k=2m(k+1)+3k$. □

由于 MDKautz 的模块内结构 CH 的直径大于 BCube,为了更公平地对比模块间的互联结构,同样采用 BCube 作为 MDKautz 的模块内结构,给定 BCube 的参数为 n=32,k=1,图 8 给出了其直径随网络规模变化的情况.可以看出,MDKautz 的直径的增长率小于 MDCube,当网络规模超过百万后,MDKautz 的直径更优.当 k=2 时,由 $C(2,7)$构建的 MDKautz 网络的最长最短路径为 54.实际通信过程中,服务器间的路径长度是小于这个上限值的,因为网络中任意两台服务器间都有多条并行路径,采用最短路径路由算法极大地降低了走最长路径的概率.

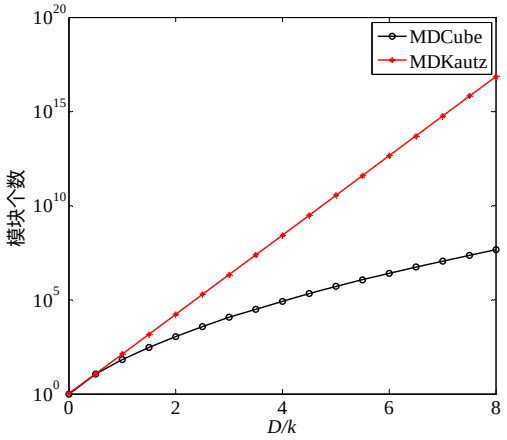

Fig.7 Number of modules is a function of each parameter

图 7 模块个数随相应参数的变化关系

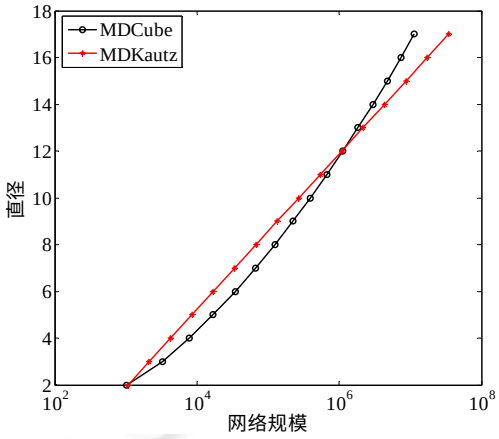

Fig.8 Diameter is a function of network size

图 8 网络直径随网络规模的变化关系

定理 **4**. $M(n,m,k)$网络的二分宽度为 $\frac{n^{k(m+1)}}{k2^{k+1}}\left(\frac{1}{2}n^{m+1}+1\right)$.

证明:由 Kautz 图的性质 3 和 $d=\frac{1}{2}n^{m+1}$,容易得证.

在便于分析,同时不影响分析结论的前提下,假设 MDCube 在各维度上互连的模块数量相等,则其二分宽度可以表示为 $\begin{cases}\frac{N^2-1}{4}N^{D-1}, & N\text{为奇数}\\ \frac{N^2}{4}(1+N^{D-1}), & N\text{为偶数}\end{cases}$,其中, $N=\frac{n^m(m+1)}{D}+1$ 表示每个维度包含的模块个数. □

图 9 给出了单个模块的规模相同时,两种结构的二分宽度随网络规模的变化情况.可以看出,两种结构的二分宽度均随网络规模的增大而增大,MDKautz 的二分宽度明显优于 MDCube.

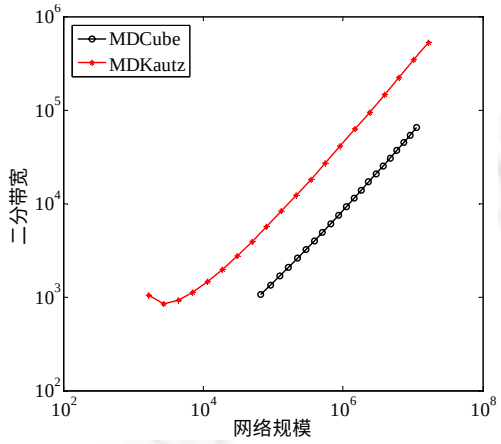

Fig.9 Bisection bandwidth is a function of network size

图 9 二分宽度随网络规模的变化关系

4.2 流量分布与ABT

All-to-All 通信模式广泛应用于 MapReduce 等应用中.ABT(aggregate bottleneck throughput)是数据中心网络中衡量 all-to-all 通信的一个评价指标,表示网络内各数据流获得的最小带宽与网络内所有数据流总和的乘

积,反映了数据中心网络在 all-to-all 通信模式下的网络容量.ABT 较大,也意味着 all-to-all 通信的时长更短.在 all-to-all 通信模式下,数据中心网络中的流量由模块内数据流和模块间数据流两部分组成,模块内数据流指每台服务器与同一模块内的其他所有服务器进行通信时产生的数据流;模块间数据流指每台服务器与其他所有不同模块内的所有服务器进行通信产生的网络流.由此,MDKautz 网络和 MDCent 网络中数据流的分布情况可用如下定理表示.

定理 **5**. 假设整个 MDKautz 网络都处于 all-to-all 的流量模式下,即网络中任意两台服务器之间存在一对数据流.若采用 MDKautzRouting 算法,则每条普通链路上承载的数据流的数量为

$$\left(2m+1-\frac{2m}{n}+\frac{n-1}{n}(k-1)(m+1)\right)\frac{(N-t)}{n}.$$

每条高速链路上承载的数据流的数量为 $(k-1)(m+1)\frac{(N-t)}{n}$,其中,$t$ 和 N 分别为 CH 模块和 MDKautz 网络中服务器的数量.

证明:根据 CH_m 结构的性质 3 可知,单个 CH_m 模块的服务器个数 $t=n^m(m+1)$,交换机个数 $g=n^{m+1}$,由 $M(n,m,k)$ 结构的定义可知:网络包含的模块总数为 M,网络规模 $N=tM$.有向普通链路的个数为 $2nN$,有向高速链路的个数为 $\frac{1}{2}gM$.在 all-to-all 流量模式下,MDKautz 网络中所有模块间数据流的数量为 $M(M-1)t^2$.

在 CH_m 网络中,所有服务器到任意交换机 A 的路径长度可以分成 i 组,第 G_i 组包括所有距离 A 为 $2i-1$ ($i\in[1,m+1]$)跳的服务器.每组中的服务器数量可表示为 $(m+1)C_m^i(n-1)^i$,则服务器到交换机的平均路径长度为

$$h_1=\frac{1}{(m+1)n^m}\sum_{i=0}^{m}[(2i+1)(m+1)C_m^i(n-1)^i]=2m+1-\frac{2m}{n}.$$

同理,计算出任意两台交换机之间的平均路径长度 $h_2=\frac{1}{n^{m+1}-1}\sum_{i=1}^{m+1}[2iC_{m+1}^i(n-1)^i]=\frac{2(n-1)t}{n(t-1)}(m+1)$.

在 $M(n,m,k)$中,任意两台服务器节点之间最多经过包括源、目的模块在内的 $k+1$ 个模块,由于中间模块通常为几十到数百不等,所以本文忽略目的模块中服务间的数据流,并认为经过的中间模块的平均数量为 $k-1$,则平均每条模块间数据流所要经过的普通链路的数量近似可表示为 $2h_1+(k-1)h_2$.由于 CH 中所有链路的使用都是平等和均衡的,因此,每条普通链路上承载的流的数量为

$$\frac{(2h_1+(k-1)h_2)M(M-1)t^2}{2nN}=\left(2m+1-\frac{2m}{n}+\frac{n-1}{n}(k-1)(m+1)\right)\frac{(N-t)}{n}.$$

同理,每条高速链路上的流的数量为 $\frac{kM(M-1)t^2}{\frac{1}{2}gM}=2k(m+1)\frac{(N-t)}{n}$. □

由定理 5 可知,普通链路和高速链路传输的数据流的数量不同.因此,要实现无阻塞地 all-to-all 通信,二者所需提供的链路带宽也不尽相同.高速链路和普通链路所承载的数据流数量之比,决定了 all-to-all 通信时模块间互联结构与模块内互联结构所需提供的聚合带宽,也决定了整个模块化数据中心网络是否存在性能瓶颈.由定理 5 可得出,这一比率为 $r=\frac{m+1}{2m+1-\frac{2m}{n}+k\frac{n-1}{n}(m+1)}$.

因此,对于一个由 16 512 个 CH 模块构成的 $M(2,7,2)$网络,假设普通链路的容量为 1,那么 all-to-all 流量模式下其所需的高速链路容量为 2.7Gbps,即高速链路只需提供 2.7Gbps 的带宽即可避免成为瓶颈,实现无阻塞的 all-to-all 传输.而目前普通商用交换机单个高速端口的带宽为 10Gbps,因此,高速链路不会成为整个 MDKautz 网络的性能瓶颈.

从定理 5 的证明过程中还可以得出以下推论.

推论 **1**. MDKautz 网络的瓶颈链路为模块内普通链路,故其 ABT 为 $N\left(\frac{2m+1}{n}+\frac{(k-1)(n-1)(m+1)-2m}{n^2}\right)^{-1}$.

图 10 给出了 MDKautz(k=2)与 MDCube(D=2)的 ABT 随网络规模的变化规律,二者的 ABT 均随网络规模的增大呈线性增长趋势,当 n>6 时,MDKautz 的 ABT 大于 MDCube 的 ABT.

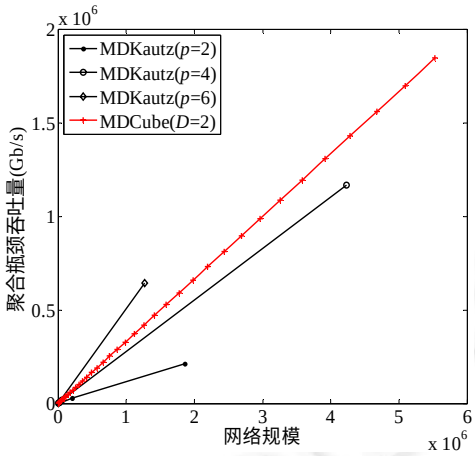

Fig.10 ABT is a function of network size

图 10 ABT 随网络规模的变化关系

5 模拟实验

在超大规模数据中心中,服务器和交换机都不可避免地遇到各种故障,而且故障发生时往往无法即刻定位并修复.因此,我们更关注存在网络失效的情况下模块内、外的容错路由策略是否能够确保 MDKautz 网络的性能呈现平缓下降的趋势.在本节中,我们在 Eclipse6.0 平台下编写了模拟程序,选取 k=2 时的 MDKautz 网络作为评估对象,选取 D=2 时的典型 MDCube 网络作为比较对象.令模块内部普通链路的速率为 1Gbps,而不同模块间高速链路的速率为 10Gbps.

第 1 个实验测试不同规模的 MDKautz 网络中 2 个模块间通信的 ABT.该实验模拟 MapReduce 的 reduce 过程,即每个 reducer 从所有 mapper 中取回数据,产生一个 all-to-all 的流量模式.在该流量模式下,当服务器/交换机的失效率从 0%达到 20%时,观察 2 个模块间通信时网络的 ABT 的变化情况.假设模块内部普通链路的速率为 1Gbps,而不同模块间高速链路的速率为 10Gbps(如图 11 所示).

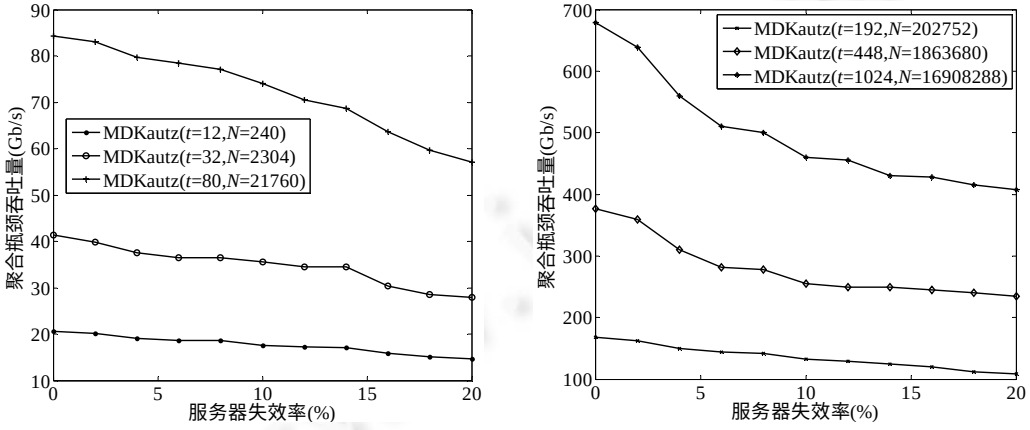

Fig.11 ABT of two modules in MDKautz under server failures

图 11 MDKautz 中 2 个模块间通信的 ABT 随服务器失效率的变化曲线

从图 11 中可以看出:单个模块的规模 t 越大,MDKautz 网络中 2 个模块之间的 ABT 越大.因为当服务器的网络端口数一定时,单个模块中交换机的数量随规模的增大而增加,因此能够提供给模块间的通信链路也随之增多.与此同时,随着链路故障率的上升,MDKautz 网络的性能始终是平缓下降的,这是因为 MDKautz 网络的模块内部和模块之间均存在多条并行路径,因此任意一对服务器间存在多条富连接,在网络失效的情况下,冗余的通信路径确保任意两台服务器间总有一条连通的链路.根据 ABT 还可以算出每台服务器实际获得的吞吐量,以规模为 2 304 为例,当链路失效率为 0 时,2 个模块间(共 64 台服务器)的最大可达聚合吞吐量为 45.304.因此,在网卡线速为 1Gbps 的条件下,每台服务器实际获得的吞吐量为 0.71Gbps.

在目前提出的模块间互连结构中,MDCube 结构拥有最优网络容量和容错性能,因此,第 2 个实验比较单个模块规模相同、网络规模不同的 MDKautz 和 MDCube 网络中 2 个模块间通信的 ABT.结果表明,图 12(a)、图 12(b)中,MDKautz 的 ABT 大于 MDCube,而图 12(c)、图 12(d)正好相反.这是因为单个模块的规模相同时,模块内的交换机数量决定了模块间互连的链路数,从而决定了网络容量的大小.在图 12(a)的参数配置下,MDKautz 中单个模块对外能够提供的交换机端口数均大于 MDCube,所以其相应的 ABT 也大于 MDCube,而图 12(b)中,MDKautz 的单个模块对外能够提供的交换机端口数均小于 MDCube,所以其相应的 ABT 也比 MDCube 小.同时,当 20%的链路失效率发生时,MDCube 的 ABT 性能平均下降 50%,而 MDKautz 只有 38%.因此,随着网络失效率的增加,MDKautz 的性能下降更为平缓.

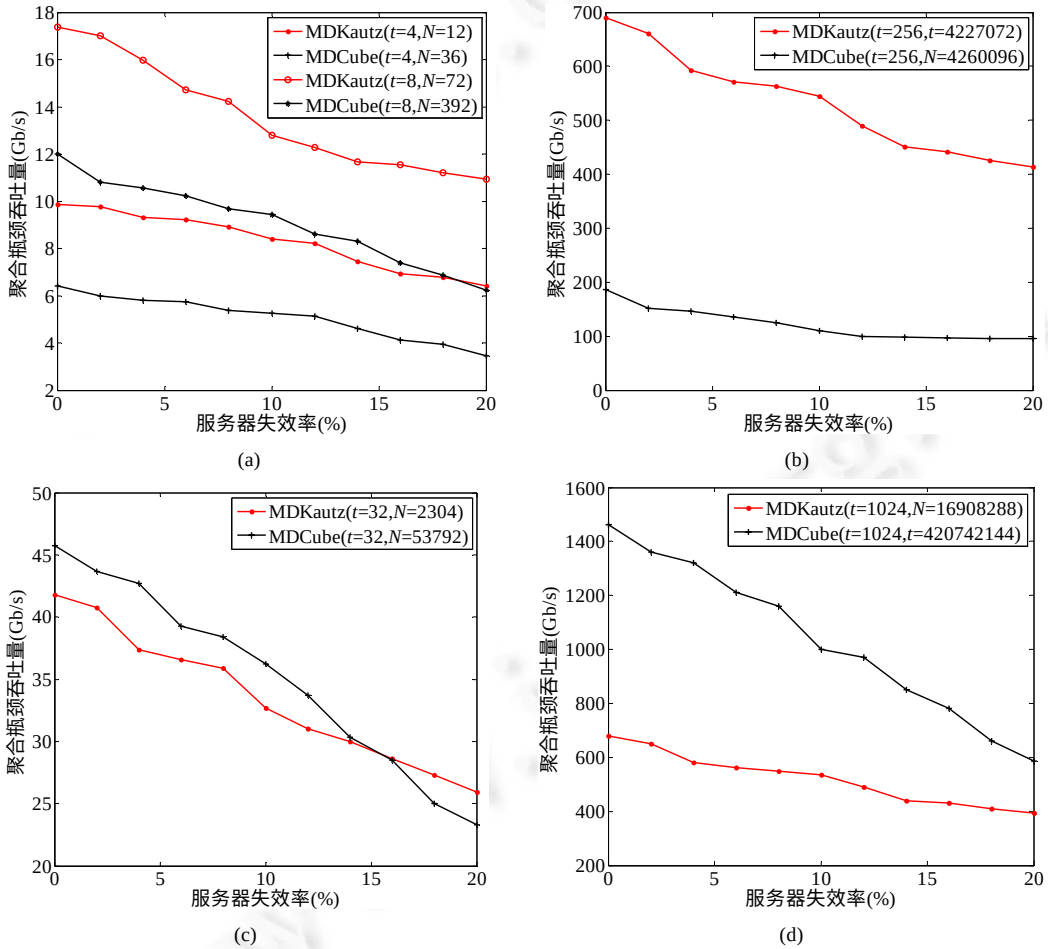

Fig.12 Comparison between MDKautz and MDCube in the ABT of two modules

图 12 MDKautz 与 MDCube 中 2 个模块间通信的 ABT 对比

6 结 论

本文针对超大规模MDC网络的互连问题提出了一种新型网络结构MDKautz.MDKautz利用模块内大量未被使用的交换机预留高速端口将模块以Kautz图互连,构造出具有高带宽、高容错和灵活可持续扩展性的超大规模数据中心网络.模块内外松耦合的互连设计和路由算法为模块之间提供了较高的聚合带宽;多条模块间不相交平行多路径确保网络具有良好的容错能力;且能够通过少量地调整现有网络结构,使网络的扩展不受限于模块内的可用高速端口数,从而进一步满足大规模数据中心对无损扩展和持续扩展的需求.MDKautz 能够广泛适用于连接各种模块内的互联结构,且均能达到较理想的网络性能.数学分析和模拟实验结果表明:MDKautz结构具有良好的拓扑特性和通信性能,能够以较小的直径构建出较大规模的网络,同时具有更大的二分带宽,对构建超大规模MDC网络是可行的.

References:

[1] Katz RH. Tech titans building boom. IEEE Spectrum, 2009,46(2):40−54. [doi: 10.1109/MSPEC.2009.4768855]

[2] Ghemawat S, Gobioff H, Leung S. The google file system. In: Proc. of the SOSP 2003. New York: ACM Press, 2003. 29−43. [doi: 10.1145/945445.945450]

[3] Shvachko K, Kuang H, Radia S, Chansler R. The hadoop distributed file system. In: Proc. of the MSST 2010. Washington: IEEE Computer Society, 2010. 1−10. [doi: 10.1109/MSST.2010.5496972]

[4] Weil SA, Brandt SA, Miller EL, Long DDE, Maltzahn C. Ceph: A scalable, high-performance distributed file system. In: Proc. of the OSDI 2006. Berkeley: USENIX Association, 2006. 307−320.

[5] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. In: Proc. of the OSDI 2004. Berkeley: USENIX Association, 2004. 27−39. [doi: 10.1145/1327452.1327492]

[6] Isard M, Budiu M, Yu Y, Birrell A, Fetterly D. Dryad: Distributed data-parallel programs from sequential building blocks. In: Proc. of the EuroSys 2007. New York: ACM Press, 2007. 59−72. [doi: 10.1145/1272996.1273005]

[7] Meisner D, Sadler CM, Barroso LA, Weber WD, Wenish TF. Power management of online data-intensive service. SIGARCH Computer Architecture News, 2011,39(3):319−330. [doi: 10.1145/2000064.2000103]

[8] Wu H, Lu G, Li D, Guo C, Zhang Y. MDCube: A high performance network structure for modular data center interconnection. In: Proc. of the CoNEXT 2009. New York: ACM Press, 2009. 25−36. [doi: 10.1145/1658939.1658943]

[9] Bhuyan LN, Agrawal DP. Design and performance of generalized interconnection networks. IEEE Trans. on Computers, 1983, c-32(12):1081−1090. [doi: 10.1109/TC.1983.1676168]

[10] Dally WJ, Towles B. Principles and Practices of Interconnection Networks. San Francisco: Morgan Kaufmann Publishers Inc., 2003.

[11] Al-Fares M, Loukissas A, Vahdat A. A scalable, commodity data center network architecture. In: Proc. of the SIGCOMM 2008. New York: ACM Press, 2008. 63−74. [doi: 10.1145/1402958.1402967]

[12] Massively Scalable Data Center (MSDC) Design and Implementation Guide. Cisco Systems, Inc., 2014.

[13] Niranjan MR, Pamboris A, Farrington N, Huang N, Miri P, Radhakrishnan S, Subramanya V, Vahdat A. PortLand: A scalable fault-tolerant layer 2 data center network fabric. SIGCOMM Computer Communication Review, 2009,39(4):39−50. [doi: 10.1145/1594977.1592575]

[14] Greenberg A, Hamilton JR, Jain N, Kandula S, Kim C, Lahiri P, Maltz DA, Patel P, Sengupta S. VL2: A scalable and flexible data center network. Communications of the ACM, 2011,54(3):95−104. [doi: 10.1145/1897852.1897877]

[15] Ramachandran K, Kokku R, Mahindra R, Rangarajan S. 60GHz Data-Center networking: Wirless⇒Worry less? Technical Report, Princeton: NEC Laboratories America, 2008.

[16] Wang G, Andersen DG, Kaminsky M, Papagiannaki K, Ng TSE, Kozuch M, Ryan M. c-Through: Part-Time optics in data centers. SIGCOMM Computer Communication Review, 2010,40(4):327−338. [doi: 10.1145/1851275.1851222]

[17] Farrington N, Porter G, Radhakrishnan S, Bazzaz HH, Subramanya V, Fainman Y, Papen G, Vahdat A. Helios: A hybrid electrical/optical switch architecture for modular data centers. In: Proc. of the SIGCOMM 2010. New York: ACM Press, 2010. 339−350. [doi: 10.1145/1851182.1851223]

[18] Guo C, Wu H, Tan K, Shi L, Zhang Y, Lu S. DCell: A scalable and fault-tolerant network structure for data centers. In: Proc. of the SIGCOMM 2008. New York: ACM Press, 2008. 75−86. [doi: 10.1145/1402958.1402968]

[19] Li D, Guo C, Wu H, Tan K, Zhang Y, Lu S, Wu J. Scalable and cost-effective interconnection of data-center servers using dual server ports. IEEE/ACM Trans. on Networking, 2011,19(1):102−114. [doi: 10.1109/TNET.2010.2053718]

[20] Guo C, Lu G, Li D, Wu H, Zhang X, Shi Y, Tian C, Zhang Y, Lu S. BCube: A high performance, server-centric network architecture for modular data centers. SIGCOMM Computer Communication Review, 2009,39(4):63−74. [doi: 10.1145/1594977.1592577]

[21] Huang F, Lu X, Li D, Zhang Y. A fault-tolerant network architecture for modular datacenter. Int'l Journal of Software Engineering and Its Applications, 2012,6(2):93−106.

[22] Bhuyan LN, Agrawal DP. Generalized hypercube and hyperbus structures for a computer network. IEEE Trans. on Computers, 1984,c-33(4):323−333. [doi: 10.1109/TC.1984.1676437]

[23] Li D, Xu M, Zhao H, Fu X. Building mega data center from heterogeneous containers. In: Proc. of the ICNP 2011. Washington: IEEE Computer Society, 2011. 256−265. [doi: 10.1109/ICNP.2011.6089059]

[24] Lu FF, Luo XG, Xie XH, Zhu GM, Pu XC. Researching on constant degree network for massively data center. Journal of Computer Research and Development, 2014,51(11):2437−2447 (in Chinese with English abstract). [doi: 10.7544/issn1000-1239.2014.20130165]

[25] Zhang Y, Lu X, Li D. SKY: Efficient peer-to-peer networks based on distributed Kautz graphs. Science in China Series F: Information Sciences, 2009,52(4):588−601. [doi: 10.1007/s11432-009-0016-x]

附中文参考文献:

[24] 陆菲菲,罗兴国,谢向辉,朱桂明,濮小川.面向大规模数据中心的常量度数互连网络研究.计算机研究与发展,2014,51(11):2437−2447. [doi: 10.7544/issn1000-1239.2014.20130165]

陆菲菲(1981－),女,江苏无锡人,博士,工程师,CCF 专业会员,主要研究领域为分布式计算,数据中心网络.

郭得科(1980－),男,博士,教授,CCF 高级会员,主要研究领域为分布式系统,软件定义网络,数据中心网络.

谢向辉(1958－),男,博士,研究员,博士生导师,CCF 专业会员,主要研究领域为计算机系统结构,分布式计算.

朱桂明(1980－),男,博士,工程师,主要研究领域为分布式计算,数据中心网络.