软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2014,25(6):1339−1351 [doi: 10.13328/j.cnki.jos.004444]

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

一种高吞吐量、高可扩展数据中心网络结构*

朱桂明[1], 谢向辉[1], 郭得科[2], 陆菲菲[1,3], 陶志荣[1]

[1](数学工程与先进计算国家重点实验室,江苏 无锡 214125)

[2](国防科学技术大学 信息系统与管理学院 信息系统工程国防科技重点实验室,湖南 长沙 410073)

[3](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)

通讯作者: 朱桂明, E-mail: guiming.zhu@gmail.com

摘 要: 虽然以服务器为中心的数据中心网络互连结构部分程度地解决了树型结构面临的性能瓶颈和可扩展性难题,但如何使数据中心网络同时兼具高吞吐量和高可扩展能力,仍然是一个颇具挑战性的问题.为此,提出了具有高吞吐量和高可扩展能力的常量度数数据中心网络互连结构 XDCent.XDCent 在各服务器网络端口个数为常量的情况下,确保数据中心网络在保持高吞吐量的前提下能够进行系统规模的无损和持续扩展.

关键词: 数据中心网络;常量度数;高吞吐量;高可扩展

中图法分类号: TP393

中文引用格式: 朱桂明,谢向辉,郭得科,陆菲菲,陶志荣.一种高吞吐量、高可扩展数据中心网络结构.软件学报,2014,25(6):1339−1351. http://www.jos.org.cn/1000-9825/4444.htm

英文引用格式: Zhu GM, Xie XH, Guo DK, Lu FF, Tao ZR. High performance expandable data center networking structure. Ruan Jian Xue Bao/Journal of Software, 2014,25(6):1339−1351 (in Chinese). http://www.jos.org.cn/1000-9825/4444.htm

High Performance Expandable Data Center Networking Structure

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

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

[2](State Key Laboratory for Information System Engineering, College of Information System and Management, National University of Defense Technology, Changsha 410073, China)

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

Corresponding author: ZHU Gui-Ming, E-mail: guiming.zhu@gmail.com

Abstract: While server-centric data center networking schemes partly resolve the performance bottleneck and extensibility problems in existing tree-based networking schemes, designing a high extensible data center networking scheme with high performance is still a challenging problem. This paper proposes a high extensible and high performance data center networking scheme, called XDCent, where servers have fixed number of network interface card (NIC) ports. In the case that each server has the fixed number of NICs, XDCent can not only result in a high performance for data centers but also extend data centers continually without affecting the running applications.

Key words: data center networking; fixed number of degree; high throughput; high extensibility

日益增长的大规模互联网在线应用、企业级基础服务和对云计算应用的需求,促使诞生了十万量级、甚至百万量级服务器的数据中心.现有的数据中心网络主要是通过交换机、核心交换机、核心路由器将服务器连接成树型结构,这对位于树型结构高层的核心路由器、核心交换机的性能要求极高,且其容易成为网络流量的瓶颈[1].扩展系统规模时,往往需要性能更高、价格更昂贵的核心路由器和核心交换机,系统规模扩展的代价十分

* 基金项目: 国家自然科学基金(61170284); 国家高技术研究发展计划(863)(2011AA01A203)
收稿时间: 2012-08-24; 修改时间: 2012-12-03; 定稿时间: 2013-06-21

巨大.此外,树型结构容错性较差,容易出现单点失效问题.这种结构很难满足数据中心网络追求的高聚集带宽、高可扩展性、容错性好的设计目标.近年来,研究者对树型结构进行了拓展,提出了 Fat-Tree[2],VL2[3]这类以交换机为中心的多层树型互连结构;同时,从以服务器为中心的数据中心网络设计角度,提出了 DCell[4],FiConn[5],BCube[6],HCN[7]和 BCN[7]等互连结构,这类互连结构的路由功能由各个服务器分担,使得网络中不存在核心路由器和核心交换机,可扩展性得以大大增强,且由于服务器的编程能力强,因配置特别灵活.

目前提出的以服务器为中心的数据中心网络互连结构,为构建大规模数据中心网络奠定了基础,但是数据中心网络的以下几个重要问题仍然未得到解决:

(1) 数据中心网络规模的扩展对各类应用不产生影响.DCell 和 BCube 每扩展一层,要求每台服务器额外增加一个 NIC 端口,并为各服务器添加新的连线,从而影响了各类应用,无法进行无损扩展;

(2) 在服务器网络端口个数为常量的情况下,保证数据中心网络的可持续扩展.DCell 和 BCube 为非常量度数的结构,数据中心网络的规模受限于服务器 NIC 端口的个数;

(3) 在服务器网络端口个数为常量、保证数据中心网络的高吞吐量的前提下,保持数据中心网络的高可扩展.FiConn,HCN 以及 BCN 是 NIC 端口个数为 2 的常量度数数据中心网络,HCN 和 BCN 能够使得数据中心网络进行无损和持续可扩展,但是这些结构在追求可扩展性的同时,降低了对网络性能的要求,因为不同层次结构的链路容量一般大小,但是所承担的流量却可能差异很大.例如,在 all-to-all 的流量模型下,高层结构的链路流量更大.

为此,本文提出了一种高吞吐量、高可扩展数据中心网络互连结构 XDCent.XDCent 在各服务器网络端口个数为常量的情况下,各服务器预留 1 个网络端口,其余网络端口个数首先依照 BCube 的互连方式构建数据中心网络,直到这些网络端口被使用完.然后,利用各服务器预留的 1 个网络端口,使得数据中心网络的规模能够进行持续不断的扩展.XDCent 既能保证数据中心局部网络的高性能特性,兼顾整个网络具有较高的性能,又能保证网络的高可扩展性.

1 相关工作

现有的数据中心网络互连结构主要依靠交换机、核心交换机、核心路由器将服务器连接起来构成树型结构,树型结构的固有弊端使得其远远不能达到大型数据中心所追求的高可扩展、容错性好、高聚集带宽等目标.为了更贴合数据中心网络需求,Al-Fares 提出使用廉价的交换机构建胖树结构 Fat-Tree,如图 1 所示.

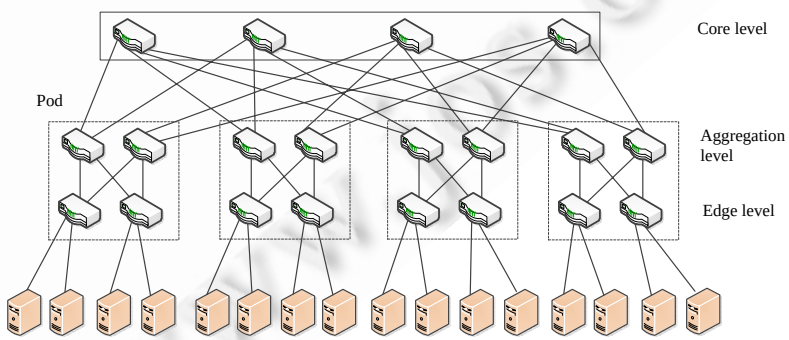


Fig.1 Fat-Tree structure

图 1 Fat-Tree 结构

该结构实现了系统的大规模互连以及服务器之间的高通信带宽.Fat-tree 共有核心层(core level)、聚集层(aggregation level)、边界层(edge level)这 3 层交换机;有 n 个 Pod,每个 Pod 含有聚集层和边界层各 $n/2$ 个交换机;边界层的每个交换机使用 $n/2$ 个端口连接服务器,其余 $n/2$ 端口连接聚集层的 $n/2$ 个交换机;核心层有$(n/2)^2$ 个 n 口交换机,每个交换机有一个端口连接一个 Pod.胖树结构相对于传统多根树结构的优点是:聚集层的节点之间

具有更高的数据传输性能和容错性,且各层的链路数相等,屏蔽了根节点瓶颈问题.但它的设计思想是“横向扩展”模式,其扩展性受限于交换机端口数量,存在扩展性不足的缺点;它的另一个缺点是处理交换机故障能力不足及路由协议容错性不强;而它的树形特征也决定了其不能很好地支持 one-to-many(一对多)和 several-to-several(多对多)的网络通信服务.

为了能更好地达到数据中心网络所追求的高性能目标,越来越多的以服务器为中心的数据中心网络结构在近几年被提出.借助服务器的多网络 NIC 端口以及数据转发功能,可以利用大量服务器之间的连线以及低端交换机来实现大型数据中心网络的高效互连,其中,基于复合图(compound graph)并采用层次网络的设计思想成为当前以服务器为中心的数据中心网络互连的主流方法.

DCell 提供了可扩展的结构来构建大型数据中心网络,其利用低端交换机以及具有多网络 NIC 端口的服务器迭代地构建服务器之间的互连结构,利用分布式容错性路由协议 DFR(DCell fault-tolerant routing)实现近似的最短路路由,并将流量均匀分散到各条链路上消除瓶颈问题.*DCell* 使用迭代的方式构建,每个高层的 *DCell* 通过连接一定数量的低层 DCell 来构建,多个同层 DCell 之间彼此全连通.$DCell_0$ 是最基本的构建模块,它由 n 个服务器与一个 n 口微型交换机连接构成;$DCell_1$ 由 n+1 个 $DCell_0$ 构成,图 2 给出了 n=4 的 $DCell_1$ 拓扑结构.*DCell* 采用的是完全复合图(complete compound graph)结构[7],而 FiConn 和 HCN 采用的是不完全复合图(incomplete compound graph)结构.对 FiConn 而言,每台服务器只需要配备两个 NIC 端口就能构建任意层次的 FiConn 结构,如图 3 所示.

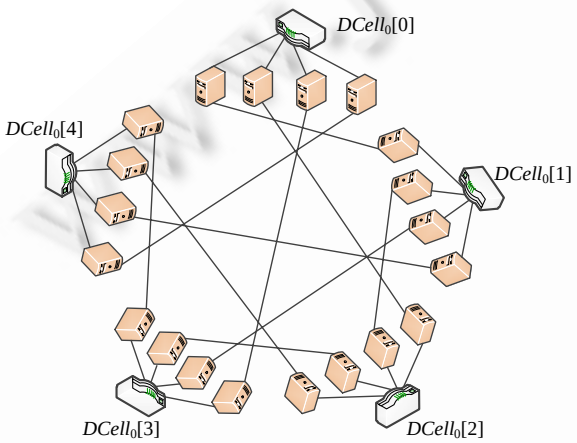


Fig.2 $DCell_1$ structure

图 2 $DCell_1$ 结构

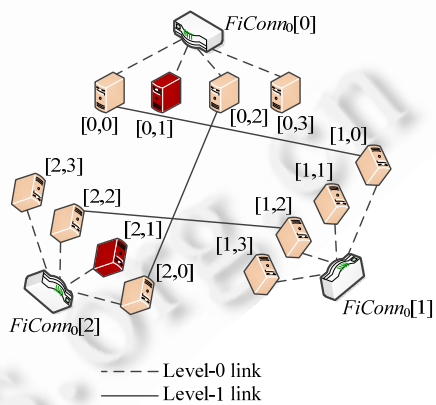


Fig.3 $FiConn_1$ structure

图 3 $FiConn_1$ 结构

BCube 是一种依层次化构建的、具有高吞吐量的新型数据中心网络结构.$BCube_0$ 由 n 个服务器连接一个 n 口交换机构成;$BCube_k$ 由 n 个 $BCube_{k-1}$ 和 n^k 个 n 口交换机构成.

图 4 给出了 $BCube_1$ 互连结构,与 DCell 中服务器直接与其他 cell 中的服务器相连不同的是,BCube 中不同 cube 之间通过服务器-交换机-服务器的方式连接,当出现部分交换机或服务器失效时,BCube 的性能降低的速率比较缓慢.HCN 和 BCN 与 FiConn 有着相似的设计理念,即网络的规模不受限于服务器的网络 NIC 端口个数;同时,HCN 结构具有高度的规则性、灵活性和系统性.但是,HCN 和 BCN 专注于解决数据中心网络的无损可扩展性、持续可扩展性和渐进可扩展性问题,却降低了对数据中心网络性能的要求.

图 5 给出了 *HCN*(4,2)结构.CamCube[8]结构中服务器之间通过多个网卡直接相连,构成一个 3D torus 拓扑结构,数据包的转发工作完全由服务器承担.CamCube 结构的路由机制需要占用大量服务器的计算资源,对服务器的负载压力较大.其网络规模受限于服务器的网络端口个数.

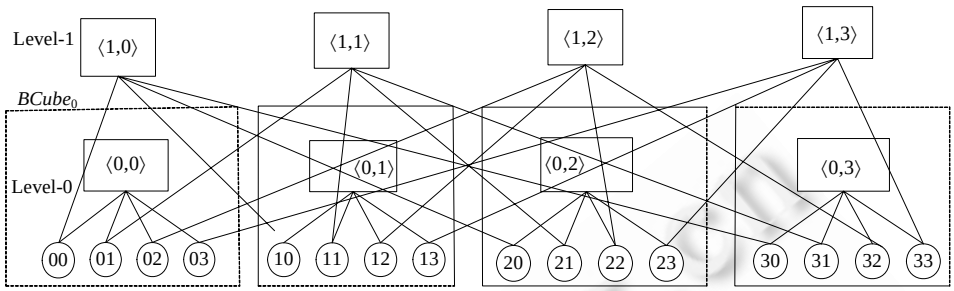


Fig.4 $BCube_1$ structure

图 4 $BCube_1$ 结构

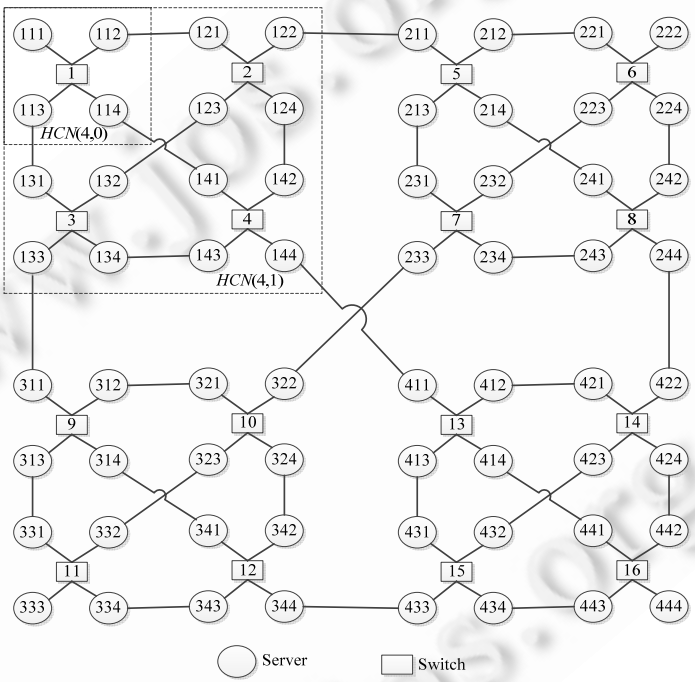


Fig.5 *HCN*(4,2) structure

图 5 *HCN*(4,2)结构

Jellyfish[9]采用随机图的方法将大量的 ToR 交换机互连成随机网络,服务器直接与交换机相连.这种结构能够很好地支持数据中心网络的渐进式扩展,且扩展时只需新增连线,对网络改动较小.然而,Jellyfish 是一种不规则的互连结构,其路由和布线的设计都是具有挑战性的问题.SWDC[10]和 Scafida[11]在数据中心拓扑设计中分别引入了小世界模型和无尺度网络模型:前者首先采用规则网络拓扑互联服务器,然后根据小世界分布模型为服务器增加了一些随机边,这些随机边起到了缩短网络直径的作用;后者依据无尺度网络模型约定服务器之间的互联关系,并获得较小的网络直径和很高的容错能力.

针对数据中心网络尚未解决的在保持高吞吐量前提下的高可扩展性问题,本文提出了具有高吞吐量和高可扩展的常量度数数据中心网络互连结构 XDCent.对于配置任意固定数目的网络 NIC 端口服务器,XDCent 要求各服务器预留 1 个网络端口,其余网络端口个数首先依照 BCube 结构的互连方式构建数据中心网络,直到这些网络端口被使用完;然后,XDCent 利用各服务器预留的 1 个网络端口,确保了数据中心网络的规模能够进行持续扩展.XDCent 既能保证数据中心局部网络的高性能特性,兼顾整个网络具有较高的性能,又能保证网络的高

可扩展性.

2 XDCent 结构

2.1 XDCent构建方法

XDCent 的主要构件为带有 k+1 个网络端口的服务器和 n 口交换机,通过迭代方式构建.$XDCent_0$ 由 n 个服务器与一个 n 口交换机连接而成.$XDCent_1$ 由 n 个 $XDCent_0$ 和 n 或 n–1 个 n 口交换机构成.一般地,$XDCent_i$ 由 n 个 $XDCent_{i-1}$ 和 $n^i(i\leqslant k-1)$或 $n^k-n^{k-1}(i\geqslant k)$个交换机构成.

构建 $XDCent_i$ 的方法为:

- 将 n 个 $XDCent_{i-1}$ 编号为 0~n–1;
- 将 $XDCent_i$ 中的服务器编号为 0~n^{i+1}–1,并以 $a_ia_{i-1}\ldots a_0$ 表示,其中,$a_j\in[0,n-1],j\in[0,i]$;
- 交换机表示为$\langle l,s_{i-1}\ldots s_0\rangle$,其中,$l$ 为交换机的级别,$s_j\in[0,n-1],j\in[0,i-1]$.

基于这 n 个 $XDCent_{i-1}$,我们通过如下规则构建 $XDCent_i$:

(1) 当 $i\leqslant k-1$ 时,将第 j 个 $XDCent_{i-1}$ 中的服务器 $ja_{i-1}a_{i-2}\ldots a_0$ 的第 i 级网络端口与第 i 级的交换机$\langle i,a_{i-1}\ldots a_0\rangle$的第 j 个网络端口相连;

(2) 当$i\geqslant k$时,将第j个$XDCent_{i-1}$中的服务器$ja_{i-1}a_{i-2}\ldots a_0$的第$k$级网络端口与第$i$级的交换机$\langle i,a_{i-1}\ldots a_{i-k}\rangle$的第 j 端口相连,其中,$a_{i-1}a_{i-2}\ldots a_0$ 必须满足 $a_r=n-1$ for $0\leqslant r\leqslant i-k-1,a_{i-k}\neq n-1$.

XDCent 结构中,交换机只连接服务器,交换机之间没有直接连接.对服务器而言,交换机仅起到数据转发的作用,从而,普通商用交换机即可满足 *XDCent* 结构的要求.

- 当 $i\leqslant k-1$ 时,$XDCent_i$ 构建方法等同 *BCube*,由 n 个 $XDCent_{i-1}$ 通过 n^i 个交换机连接而构成;
- 当 $i\geqslant k$ 时,$XDCent_i$ 由 n 个 $XDCent_{i-1}$ 通过 n^k-n^{k-1} 个交换机连接各自的 n^k-n^{k-1} 服务器而构成,$XDCent_i$ 中剩余 n^k 个第 k 级网络端口处于空闲状态的服务器用于构建 $XDCent_{i+1}$ 结构.

图 6 给出了 n=4,k=2 时 $XDCent_2$ 的结构.$XDCent_2$ 由 4 个 $XDCent_1$ 通过各自服务器节点 00,01,02,10,11,12,20,21,22,30,31,32 分别与交换机$\langle 2,00\rangle$,$\langle 2,01\rangle$,$\langle 2,02\rangle$,$\langle 2,10\rangle$,$\langle 2,11\rangle$,$\langle 2,12\rangle$,$\langle 2,20\rangle$,$\langle 2,21\rangle$,$\langle 2,22\rangle$,$\langle 3,30\rangle$,$\langle 3,31\rangle$,$\langle 3,32\rangle$连接而构成.$XDCent_2$中的服务器节点003,013,023,033,103,113,123,133,203,213,223,233,303,313,323,333仍有一个网络端口用于扩展 $XDCent_3$ 结构.

定理 1. 当 $i\geqslant k$ 时,$XDCent_{i-1}$ 始终有 n^k 个第 k 级网络端口处于空闲状态的服务器用于构建 $XDCent_i$ 结构.

证明:当 i=k 时,$XDCent_{k-1}$ 的服务器个数为 n^k,且每个服务器的第 0 级~第 k–1 级共 k 个网络端口均被使用;又因为每个服务器的网卡个数为 k+1,即每个服务器的第 k 级网络端口处于空闲状态,从而 $XDCent_{k-1}$ 具有 n^k 个第 k 级网络端口处于空闲状态的服务器可用于构建 $XDCent_k$ 结构.当 $j\geqslant k$ 时,如果 $XDCent_{j-1}$ 始终有 n^k 个第 k 级网络端口处于空闲状态的服务器用于构建 $XDCent_j$ 结构;又由于 $XDCent_j$ 由 n 个 $XDCent_{j-1}$ 通过 n^k-n^{k-1} 个交换机连接各自的 n^k-n^{k-1} 服务器而构成,从而,$XDCent_j$ 中第 k 级网络端口处于空闲状态的服务器个数为 $n\times n^{k-1}$ 即 n^k,即 $XDCent_j$ 有 n^k 个第 k 级网络端口处于空闲状态的服务器用于构建 $XDCent_{j+1}$ 结构.因此,当 $i\geqslant k$ 时,$XDCent_{i-1}$ 始终有 n^k 个第 k 级网络端口处于空闲状态的服务器用于构建 $XDCent_i$ 结构.定理 1 得证. □

对于给定网络端口个数为常量 k+1 的条件下,定理 1 保证了 XDCent 结构具有持续可扩展性,从而保证了 XDCent 建立大规模数据中心的可行性.

2.2 最短路路由算法

当任意源服务器 *src* 需要将报文发送给目的服务器 *dst* 时,可仅仅根据源和目的服务器的标识符 *src* 和 *dst*,即可快速计算出整条路径.

XDCent 的路由算法见算法 1.算法 1 接收源和目的服务器编号 *src*,*dst* 和[i,i–1,…,0]的一个混洗序列Π,计算出 *src* 到 *dst* 一条最短路径.算法 1 通过 *CommPrefix* 函数计算出 *src* 和 *dst* 的共同前缀,通过 *Length* 函数计算共同前缀的长度,从而确定 *src* 和 *dst* 共同最低层次 $XDCent_{i-m}$.如果 $XDCent_{i-m}$ 的构造方法等同 *BCube*,则采用

BCube 的路由算法进行路由;否则,计算出一条连接 *src* 和 *dst* 各自隶属 $XDCent_{i-m-1}$ 的路径(*src*′,*dst*′),并将计算 *src* 到 *dst* 的路径问题转化为计算 *src* 到 *src*′和 *dst*′到 *dst* 的问题,从而通过迭代调用算法 1 即可计算出 *src* 到 *dst* 的路径.*src*′和 *dst*′的计算中,$a_{i-m-1}\dots a_{i-m-k+1}\in[0,n^{k-1}]$, $a_{i-m-k-1}...a_0=\overbrace{(n-1)...(n-1)}^{i-m-k个}$,$a_{i-m-k}\neq n-1$,从而,*src*′和 *dst*′间具有 n^k-n^{k-1} 种可能性,即从 *src* 和 *dst* 各自隶属 $XDCent_{i-m-1}$ 之间的 n^k-n^{k-1} 条路径中选择一条路径进行路由.

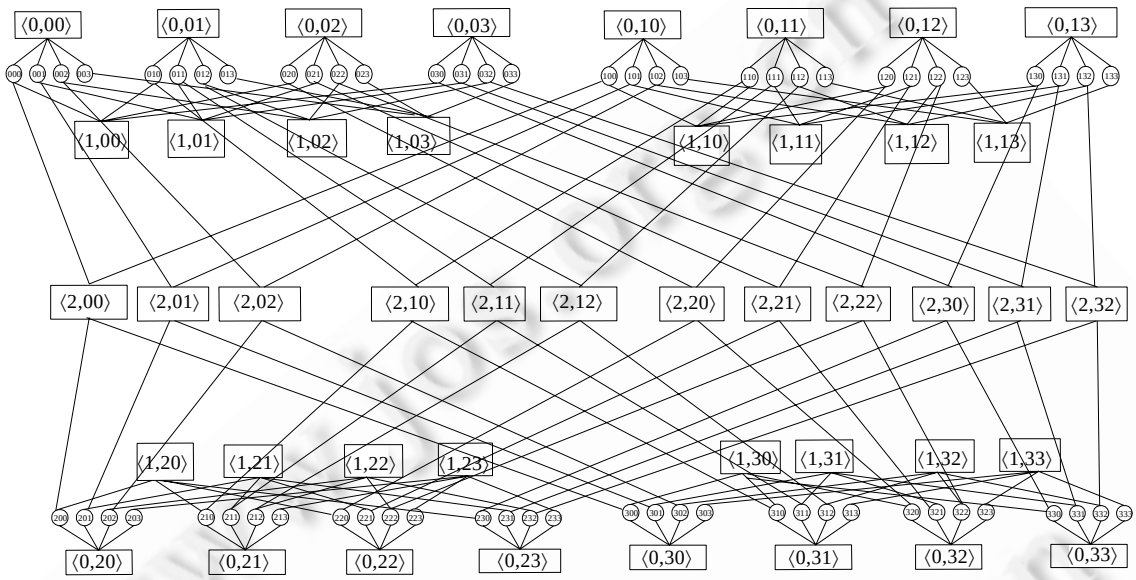


Fig.6 $XDCent_2$ (n=4, k=2) structure

图 6 $XDCent_2$(n=4,k=2)结构

算法 1. *Routing*(*src*,*dst*,*Π*).

要求:*src* 和 *dst* 是 $XDCent_i$ 中的两个服务器,$src=s_is_{i-1}\dots s_0$,$dst=d_id_{i-1}\dots d_0$,

Π=[$\pi_i,\pi_{i-1},\dots,\pi_0$]是[*i*,*i*−1,…,0]的一个混洗序列

1. *pref*=*CommPrefix*(*src*,*dst*);
2. *m*=*Length*(*pref*);
3. if (*i*−*m*<*k*) return *BCubeRouting*(*src*,*dst*,*Π*)
4. if (*i*−*m*≥*k*)
5. 计算 $a_{i-m-1}\dots a_0$,其中, $a_{i-m-k-1}...a_0=\overbrace{(n-1)...(n-1)}^{i-m-k个}$,$a_{i-m-k}\neq n-1$;
6. $src'=s_is_{i-1}\dots s_{i-m+1}s_{i-m}a_{i-m-1}\dots a_{i-m-k+1}a_{i-m-k}\dots a_0$;
7. $dst'=d_id_{i-1}\dots d_{i-m+1}d_{i-m}a_{i-m-1}\dots a_{i-m-k+1}a_{i-m-k}\dots a_0$;
8. return *Routing*(*src*,*src*′,*Π*)+(*src*′,*dst*′)+*Routing*(*dst*′,*dst*,*Π*)

算法 1 通过迭代的方法计算出 *src* 到 *dst* 的一条完整路径,迭代的次数取决于 *src* 和 *dst* 的标识符的长度 *i*+1、其共同前缀的长度 *m* 和网络端口个数 *k*+1.算法 1 的复杂度为 $O((k+1)2^{i-k+1})$.

认定连接到同一交换机的两台服务器之间的路径长度为 1,定理 2 给出了 $XDCent_i$ 网络直径的上限.

定理 2. $XDCent_i$ 的直径,即 $XDCent_i$ 中任意两台服务器之间最短路径的最大值,小于或等于 $2^{i-k+1}(k+1)-1$.

证明:从 $XDCent_i$ 的路由算法可知,对于任意两台距离最大的服务器,其路径长度$|XDCent_i|$满足:

$$|XDCent_i|\leqslant 2|XDCent_{i-1}|+1;$$

又因为 $XDCent_{k-1}$ 的构建方法等同 *BCube*,从而$|XDCent_{k-1}|=k$,因此,$|XDCent_i|\leqslant 2^{i-k+1}(k+1)-1$. □

实际应用中,$XDCent_i$ 的直径可能会极大地小于上限值 $2^{i-k+1}(k+1)-1$.XDCent 是低直径网络.

2.3 平行多径路由

如果从源服务器 *src* 到目的服务器 *dst* 的一条路径上的中间服务器、交换机不出现在从服务器 *src* 到 *dst* 的另一条路径上,那么这两条路径是从 *src* 到 *dst* 的平行路径.

定义 1. $XDCent_i$ 中用于 $XDCent_{i-1}$ 之间的连接节点称为 $XDCent_i$ 或 *i* 层连接点.

定理 3. 任给两台服务器 $src,dst \in XDCent_i$:

① 若 $i \leqslant k-1$,*src* 和 *dst* 之间具有 $i+1$ 条并行路径;

② 若 $i \geqslant k$,*src* 和 *dst* 之间至少有 *k* 条并行路径.

证明:

① 若 $i \leqslant k-1$,$XDCent_i$ 的构建方法等同 $BCube_i$,由 *BCube* 结构的性质可知,*src* 和 *dst* 之间具有 $i+1$ 条并行路径;

② 若 $i \geqslant k$,$XDCent_i$ 是由多个 $BCube_{k-1}$ 通过层次化构建的:

- 若 *src* 和 *dst* 同属一个 $BCube_{k-1}$,由 *BCube* 结构的性质可知,*src* 和 *dst* 之间具有 *k* 条并行路径;
- 若 *src* 和 *dst* 分属不同的 $BCube_{k-1}$,不失一般性,令 *src* 和 *dst* 分属不同的 $XDCent_{i-1}$,从 $XDCent_i$ 的构建方式可知,两个 $XDCent_{i-1}$ 之间通过 n^k-n^{k-1} 条并行链路进行连接.由路由算法 1 可知,*src* 和 *dst* 之间的并行路径由 $XDCent_{i-1}$ 之间的并行路径和 *src* 与 *dst* 到各自 $XDCent_{i-1}$ 之间的连接节点的并行路径构成.$XDCent_{i-1}$ 之间的连接节点均匀分布在 *n* 个 $XDCent_{i-2}$ 中,$n-1$ 个 $XDCent_{i-2}$ 到包含 *src* 或 *dst* 节点的 $XDCent_{i-2}$ 之间共具有 n^k-n^{k-1} 条并行路径;从 $XDCent_i$ 的构建方式可知,当 $i \geqslant k$ 时,对于任何一个 $XDCent_j$,$0 \leqslant j<i$,其用于连接相邻级别 $XDCent_{j+k}$ 和 $XDCent_{j+k+1}$ 的连接节点的个数分别为 $n-1$ 和 1,从 $XDCent_{j+k+1}$ 层的连接节点到 $XDCent_{j+k}$ 层的连接节点的路径能够限定在各个局部 $XDCent_i$ 中,为此,$XDCent_{i-1}$ 之间的 n^k-n^{k-1} 个连接节点与 $XDCent_{i-2}$ 之间的 n^k-n^{k-1} 个连接节点之间存在着并行路径.以此类推,$XDCent_{i-1}$ 之间的 n^k-n^{k-1} 个连接节点与 $XDCent_{k-1}$ 即 $BCube_{k-1}$ 的 n^k-n^{k-1} 个节点存在着并行路径.而由 *BCube* 结构的性质可知,存在着 *src* 和 *dst* 节点到各自 $BCube_{k-1}$ 中的任意 *k* 个节点的并行路径,又因为 $n^k-n^{k-1}>k$,所以 *src* 和 *dst* 之间至少有 *k* 条并行路径. □

定理 3 给出了 $XDCent_i$ 结构中任意两个服务器 *src* 和 *dst* 之间的并行路径条数.当 $XDCent_i$ 的层次 $i \leqslant k-1$ 时,即 $XDCent_i$ 的构建方法等同 $BCube_i$,*src* 和 *dst* 之间的并行路径条数为 $i+1$;当 $i \geqslant k$ 时,*src* 和 *dst* 之间的并行路径条数至少为 *k*,最多为 $k+1$.

算法 2 给出了构建 *src* 和 *dst* 之间所有并行路径的算法 BuildParallelPathSet.BuildParallelPathSet 首先通过 CommPrefix 函数计算出 *src* 和 *dst* 的共同前缀,通过 Length 函数计算共同前缀的长度,从而确定 *src* 和 *dst* 的共同最低层次 $XDCent_{i-m}$.如果 $i-m<k$,则通过 *BCube* 的并行路径构造算法 BuildPathSet 构建;否则,在 $XDCent_{i-1}$ 之间构建 *k* 条并行路径.从 $(n^k-n^{k-1})/(n-1)$ 即 n^{k-1} 组交换机中选择 *k* 个($n^{k-1} \geqslant k$),以构建 $XDCent_{i-1}$ 之间的 *k* 条并行路径,并将每条路径对应的交换机编号 $a_{i-m-1}\ldots a_{i-m-k}$ 作为参数传递给递归算法 XDCRouting.算法 XDCRouting 负责在包括 *src*,*dst* 的各级 *XDCent* 之间使用编号为 $a_{i-m-1}\ldots a_{i-m-k}$ 的交换机,直到 $i-m<k$.当 $i-m<k$ 时,XDCRouting 算法直接将由参数给定的源和目的节点作为路径返回.最后,由算法 addBCubePath 对并行路径集 *pathSet* 中各条路径添加 BCube 结构中对应的路径,最终完成 XDCent 结构 *k* 条并行路径的构建过程.

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

要求:*src* 和 *dst* 是 $XDCent_i$ 中的两个服务器,$src=s_is_{i-1}\ldots s_0$,$dst=d_id_{i-1}\ldots d_0$

```
1   pref=CommPrefix(src,dst);
2   m=Length(pref);
3   if (i–m<k) return BuildPathSet(src,dst);
4   pathSet={};
5   exclusiveSet={};
```

```
6   if (i−m≥k)
7     for (r=0; r<k; r++)
8       while true
9         choose random value a_{i−m−1}…a_{i−m−k+1}∈[0,n^{k−1}−1];
10        if a_{i−m−1}…a_{i−m−k+1}∉exclusiveSet
11          add a_{i−m−1}…a_{i−m−k+1} to exclusiveSet;
12          break;
13      choose random value a_{i−m−k}∈[0,n−1];
14      a_{i−m−k−1}...a_0 = (n−1)...(n−1) (i−m−k个) ;
15      src′=s_i s_{i−1}…s_{i−m+1} s_{i−m} a_{i−m−1}…a_0;
16      dst′=d_i d_{i−1}…d_{i−m+1} d_{i−m} a_{i−m−1}…a_0;
17    p_r=XDCRouting(src,src′,a_{i−m−1}…a_{i−m−k})+(src′,dst′)+XDCRouting(dst′,dst,a_{i−m−1}…a_{i−m−k});
18      add p_r to pathSet;
19    addBCubePath(pathSet);
20    return pathSet;
XDCRouting(src,dst,w_{k−1}…w_0)
1   pref=CommPrefix(src,dst);
2   m=Length(pref);
3   if (i−m<k) return (src,dst);
4   if (i−m≥k)
5     a_{i−m−1−k}...a_0 = (n−1)...(n−1) (i−m−k个) ;
6     a_{i−m−1}…a_{i−m−k}=w_{k−1}…w_0;
7     src′=s_i s_{i−1}…s_{i−m+1} s_{i−m} a_{i−m−1}…a_0;
8     dst′=d_i d_{i−1}…d_{i−m+1} d_{i−m} a_{i−m−1}…a_0;
9     return XDCRouting(src,src′,w_{k−1}…w_0)+(src′,dst′)
10          +XDCRouting(dst′,dst,w_{k−1}…w_0);
```

2.4 容错路由

将链路(*src*1,*dst*1)失败定义为 *src*1 有效但其无法与 *dst*1 通信,*src*1 和 *dst*1 连接到同一台交换机上.容错路由的基本思想是:为失败的链路寻找可替代链路,从而完成源服务器 *src* 和目的服务器 *dst* 之间的路由过程.

XDCent 的容错路由采取源路由的方式实现,其基本思想是:源节点通过算法 2 建立到目的节点的并行路径集,对并行路径集中的各条路径进行探测,当某个流对应的路径出现故障的时候,源节点从探测到的路径中重新为该流选择新的路径,继续流的传递过程,从而实现容错路由的目的.

算法 3 给出了源路由的路径选择过程.对源节点 *Source* 而言,当新的数据流到达或者探测定时器超时,则重新探测所有并行路径.由 BuildParallelPathSet 构建并行路径集合 *pathSet*,从 *pathSet* 中依次取出各条路径进行探测.如果某条路径可达,则将该路径加入到 *goodPathSet* 中;否则,通过宽度优先搜索算法为该不可达路径重新计算并行路径.计算方法是:将已有的并行路径、失效路径的某条链路从 $XDCent_i$ 结构中删除,然后使宽度优先搜索算法为该失效路径重新查找新的路径.因为新发现的路径是在将原有并行路径删除之后计算而来的,因此,其将与已有路径构成并行路径.最后,*Source* 从 *goodPathSet* 中选择一条路径.中间节点 *Intermediate_server* 接收到一个探测包时,如果包的下一跳不可达,则向源节点发送失败消息;否则,将探测包转发给下一跳节点.目的节点 *Destination* 收到探测包时,表明该条路径可达,目的节点向源节点发送路径可达应答包.

算法 3. *PathSelection*(*src*,*dst*).

要求:*src* 和 *dst* 是 $XDCent_i$ 中的两个服务器,$src=s_is_{i-1}\ldots s_0$,$dst=d_id_{i-1}\ldots d_0$

Source:

when a flow arrives or probing timer timeouts;

goodPathSet={};

pathSet=BuildParallelPathSet(*src*,*dst*);

while (*pathSet* not empty)

path=pahtSet.remove();

If (*ProbePath*(*path*)*succeeds*)

goodPathSet.add(*path*);

Else

altPath=BFS(*pathSet*,*goodPathSet*);

If (*altPath* exists) *pathSet.add*(*altPath*);

Return *SelectBestPath*(*goodPathSet*);

Intermediate_server:

when a path probe *pkt* is received:

if (next hop not available)

Send *path* failure msg to *src*; return

Forward (*pkt*);

Destination:

when a path probe *pkt* is received:

reverse the *path* in *pkt*;

Route *pkt* back to *src*;

2.5 非正则*XDCent*结构

某些情况下,受场地、经费等因素的制约,很难或者不需要一次性构建完整的(即正则)XDCent.构建非正则结构(即部分结构)XDCent 的主要思想是:遵从正则 XDCent 结构的构建方法,顶层结构使用全部个数的交换机进行互连,同时保留其中未被使用的空缺位置.这种构建方式保证了源路由算法的有效性.具体而言,对于构建非正则结构 $XDCent_i$,首先构建所需个数的 $XDCent_{i-1}$ 结构,然后使用全部个数的第 i 级交换机连接这些 $XDCent_{i-1}$ 结构,构成非正则 $XDCent_i$ 结构.第 i 级的交换机部分端口未被使用,这造成了一定程度的浪费,然而,这种构建方法很好地保证了源路由算法的有效性,且这些未被使用的端口可以作为扩展使用.

3 性能分析

3.1 网络规模

定理 4 给出了 XDCent 结构的网络规模与层次之间的关系,对于层次为 i 的 XDCent 结构,其网络规模为 n^{i+1},这与 BCube 结构的规模与层次之间的关系等同.

定理 4. $XDCent_i$ 中服务器个数为 n^{i+1}.

证明:由 *XDCent* 结构的定义可知,$XDCent_0$ 中服务器的个数为 n,$XDCent_1$ 由 n 个 $XDCent_0$ 构成,$XDCent_i$ 由 n 个 $XDCent_{i-1}$ 构成,从而,$XDCent_i$ 服务器的个数为 n^{i+1}. □

图 7 给出了 XDCent 结构的网络规模随层次的变化情况.XDCent 结构与 Bcube,HCN 结构服从同样的函数变化情况,随着层次的增加,网络规模呈指数增长,且交换机的端口个数越大,增长越快.

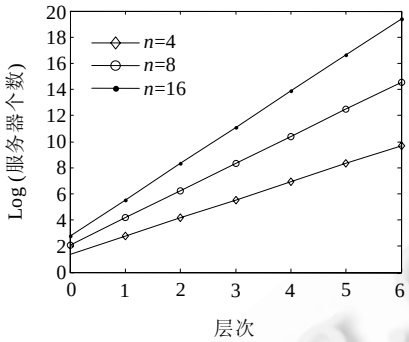


Fig.7 Number of servers is a function of layers

图 7 服务器个数随层次变化关系

定理 5. $XDCent_i$ 中交换机个数为: $a_i=\begin{cases} n\cdot a_{i-1}+n^i, a_0=1, & i<k \\ n\cdot a_{i-1}+n^k-n^{k-1}, & i\geqslant k \end{cases}$.

证明:由 XDCent 结构的定义可知,$XDCent_0$ 由 n 个服务器连接一个交换机构成,为此,a_0=1;$XDCent_i$ 由 n 个 $XDCent_{i-1}$ 构成和 $n^i(i<k)$或 $n^k-n^{k-1}(i\geqslant k)$个交换机构成,为此, $a_i=\begin{cases} n\cdot a_{i-1}+n^i, a_0=1, & i<k \\ n\cdot a_{i-1}+n^k-n^{k-1}, & i\geqslant k \end{cases}$. □

定理 5 给出了 XDCent 结构中交换机个数与层次的关系.当扩展的层次 $i<k$ 时,每扩展一层结构,顶层交换机个数为 n^i,与 BCube 结构相同.当扩展的层次 $i\geqslant k$ 时,每扩展一层结构,顶层交换机的个数始终为 n^k-n^{k-1},远小于 BCube 结构所需的交换机个数.

图 8 给出了 XDCent,HCN 和 BCube 结构的交换机个数随层次变化情况.

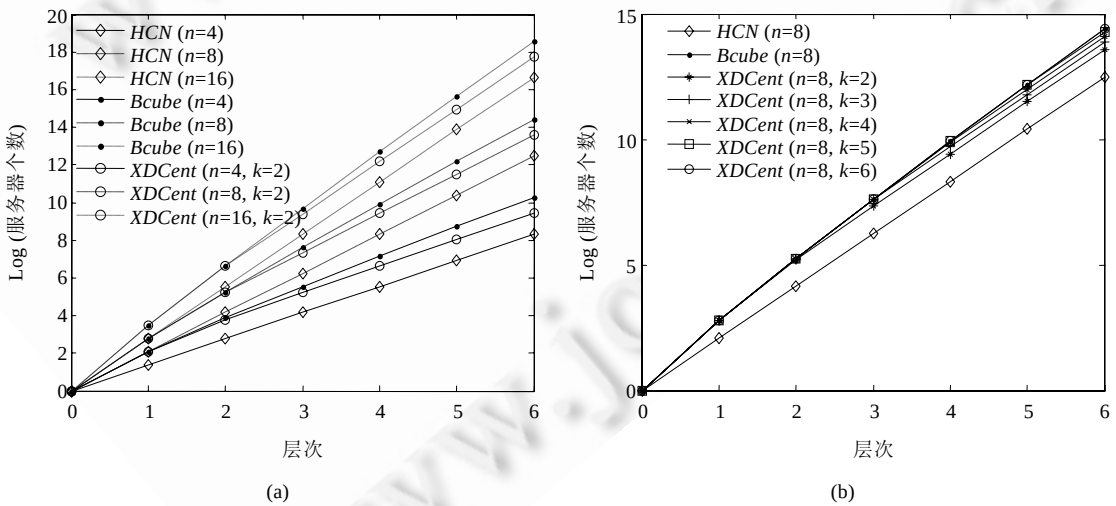


Fig.8 Number of switches is a function of layers

图 8 交换机个数随层次的变化关系

图 8(a)表明:随着层次的增加,XDCent,HCN 和 BCube 结构的交换机个数均呈指数增长,且交换机端口个数越大,增长越快;而对于相同的交换机端口个数,BCube 结构对应的交换机个数最多,HCN 最少,XDCent 介于两者之间.由图 8(b)可知:对于相同的交换机端口个数,k 值越大,XDCent 结构对应的交换机个数越多,且越接近 BCube 结构对应的交换机个数.

XDCent 结构中的链路只存在于服务器与交换机之间,为此,$XDCent_i$ 中最大链路个数为服务器个数与网络端口个数的乘积,即当 $i<k$ 时,$(i+1)n^{i+1}$ 或当 $i\geqslant k$ 时,$(k+1)n^{i+1}$.

3.2 网络直径和度

根据定理 2,$XDCent_i$ 的直径小于或等于 $2^{i-k+1}(k+1)-1$,其度数为 $k+1$.图 9 给出了 XDCent,HCN 和 BCube 结构网络直径与层次的关系.由图 9 可知,XDCent 结构的网络直径介于 HCN 和 BCube 之间,且随着 k 的增加,XDCent 结构的网络直径越来越小,并越来越接近 BCube 的直径.

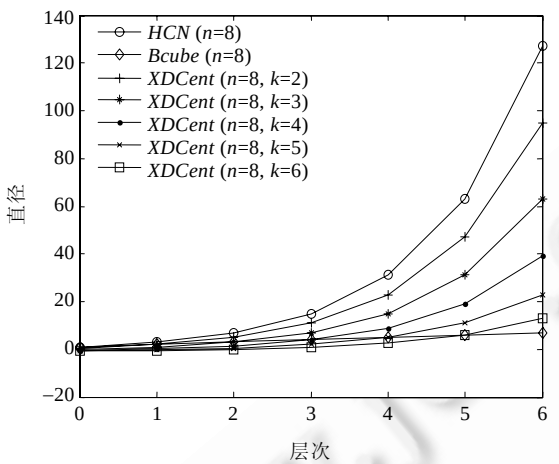


Fig.9 Network diameter is a function of layers

图 9 网络直径随层次的变化关系

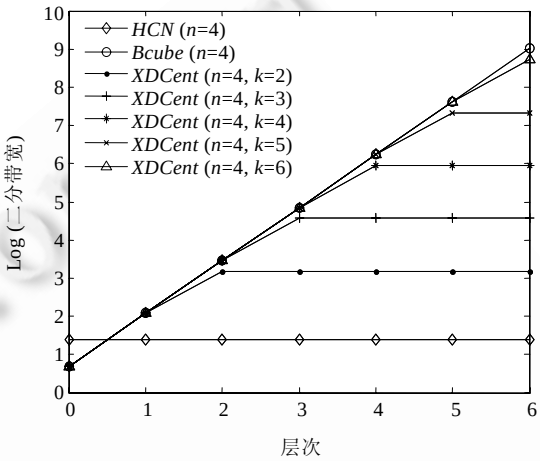


Fig.10 Bisection bandwidth is a function of layers

图 10 二分带宽随层次的变化关系

3.3 连通性和二分带宽

定理 6. $XDCent_i$ 中至少需移除 $n^j(j<k)$或 $n^k-n^{k-1}(j\geqslant k)$条链路,才能将 $XDCent_i$ 中的 $XDCent_j(0\leqslant j<i)$分离出去.

证明:从 $XDCent_i$ 构造规则可知,$XDCent_i$ 有 $n^j(j<k)$或 $n^k-n^{k-1}(j\geqslant k)$链路与 $XDCent_j$ 中其他同等结构互连,因此,至少需要移除 $n^j(j<k)$或 $n^k-n^{k-1}(j\geqslant k)$条链路才能将 $XDCent_i$ 中的 $XDCent_j(0\leqslant j<i)$分离出去. □

定理 6 描述了 XDCent 结构的连通性.由定理 6 可知,将 $XDCent_j(0\leqslant j<i)$从 $XDCent_i$ 中移除,至少需要移除 $n^j(j<k)$或 $n^k-n^{k-1}(j\geqslant k)$条链路.定理 7 给出了 XDCent 结构的二分带宽.

定理 7. $XDCent_i$ 的二分带宽,当 $i<k$ 时为 $\begin{cases} n^{i+1}/2, & n\text{为偶数} \\ (n-1)n^i/2, & n\text{为奇数} \end{cases}$,当 $i\geqslant k$ 时为 $\begin{cases} n(n^k-n^{k-1})/2, & n\text{为偶数} \\ (n-1)(n^k-n^{k-1})/2, & n\text{为奇数} \end{cases}$.

证明:$XDCent_i$ 由 n 个 $XDCent_{i-1}$ 构成和 $n^i(i<k)$或 $n^k-n^{k-1}(i\geqslant k)$个交换机构成,$XDCent_i$ 的二分带宽由 n 个 $XDCent_{i-1}$ 之间的连接决定.从而可得:

$XDCent_i$ 的二分带宽,当 $i<k$ 时为 $\begin{cases} n^{i+1}/2, & n\text{为偶数} \\ (n-1)n^i/2, & n\text{为奇数} \end{cases}$,当 $i\geqslant k$ 时为 $\begin{cases} n(n^k-n^{k-1})/2, & n\text{为偶数} \\ (n-1)(n^k-n^{k-1})/2, & n\text{为奇数} \end{cases}$. □

图 10 给出了 XDCent,HCN 和 BCube 结构的二分带宽与层次的关系.

由图 10 可知:XDCent 结构的二分带宽介于 HCN 和 BCube 之间;且随着 k 的增加,XDCent 结构的二分带宽越来越大,并越来越接近 BCube 的二分带宽.

4 模拟实验

本节通过模拟实验比较 XDCent 和 BCube 结构在服务器随机失效情况下的 ABT(aggregate bottleneck throughput)[6]值.ABT 值是在 all-to-all 通信模式下由最小流的流量与流的总数的乘积计算得到的.ABT 反映了 all-to-all 模式下的网络容量的大小.实验中,假定所有链路的容量为 1Gb/s,交换机的端口个数 n 均为 4.

图 11(a)给出了 k=2(即网络端口个数为 3)时的 $XDCent_2$ 结构、k=3 时(即网络端口个数为 4)的 $XDCent_3$ 结构、$BCube_2$ 结构和 $BCube_3$ 结构的 ABT 值随着服务器失效率的变化情况.

图 11(b)给出了 k 分别取 1~3 时,$XDCent_3$ 结构和 $BCube_3$ 结构的 ABT 值随着服务器失效率的变化情况.

图 11(c)给出了 k=2 时,$XDCent_2$ 和 $XDCent_3$ 结构、$BCube_2$ 结构的 ABT 值随着服务器失效率的变化情况.其中,$XDCent_2$ 和 $BCube_2$ 规模相同,$XDCent_3$ 和 $BCube_3$ 规模相同.

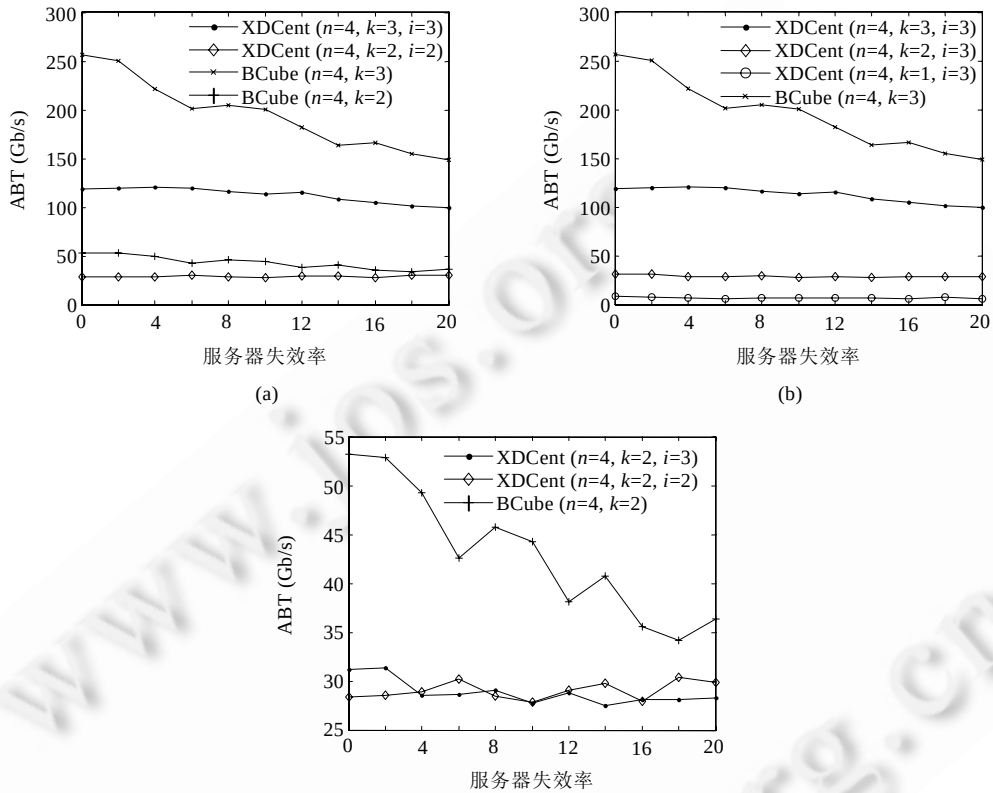


Fig.11 ABT is a function of server failure

图 11 ABT 值随着服务器失效的变化情况

由图 11(a)可知:XDCent 结构的 ABT 值随着服务器失效率的增大而缓慢降低,且降低的幅度低于 BCube 结构.网络端口个数相同、规模相同的情况下,XDCent 结构的 ABT 值比 BCube 结构低.其原因是:网络端口个数相同、规模相同的情况下,XDCent 结构预留了一部分网络端口用于扩展,从而引起 ABT 值的降低.此外,图 11(a)还表明:相同失效率的情况下,服务器网络端口个数越多、网络规模越大,XDCent 的 ABT 值越大,但均小于相应 BCube 结构的 ABT 值.

由图 11(b)可知:对于相同规模的 XDCent 结构,服务器的网络端口个数越多,ABT 值也越大,但小于对应 BCube 结构的 ABT 值.

由图 11(c)可知:$XDCent_2$ 和 $XDCent_3$ 结构的 ABT 值随着服务器失效率的变化情况基本相同,均小于 $BCube_2$ 结构的 ABT 值.但是对于配置相同网络端口个数的 $XDCent_3$ 结构和 $BCube_2$ 结构,前者的网络规模为 256,远大于后者的 64.

由实验可知,XDCent 结构的 ABT 值随着服务器失效率的增长而缓慢下降;对于相同条件下的 XDCent 和 BCube 结构,前者的 ABT 值比后者小;对于服务器的网络端口个数相同的条件下,XDCent 结构的网络规模远大于 BCube 结构.为此,XDCent 结构更适合对于扩展性要求比较高的应用.

5 结 论

本文针对数据中心网络在保持高吞吐量前提下的高可扩展性问题,提出了具有高吞吐量和高可扩展的常量度数数据中心网络互连结构 XDCent.XDCent 在各服务器网络端口个数为常量的情况下,首先依照 BCube 结构的互连方式构建数据中心网络,直到各个服务器只剩余 1 个网络端口.然后,利用各服务器剩余的 1 个网络端口,采用Incomplete compound graph方式进行连接,确保数据中心网络的规模能够进行持续扩展.XDCent既能保证数据中心局部网络的高性能特性,兼顾整个网络具有较高的性能,又能保证网络的高可扩展性.

References:

[1] Chen GH, Wu P, Yang PL. Data center network. Communications of the CCF, 2011,7(7):21−26 (in Chinese with English abstract).

[2] Fares MA, Loukissas A, Vahdat A. A scalable, commodity data center network architecture. In: Proc. of the SIGCOMM. 2008. 63−74. [doi: 10.1145/1402958.1402967]

[3] Greenberg A, Jain N, Kandula S, Kim C, Lahiri P, Maltz DA. Vl2: A scalable and flexible data center network. In: Proc. of the SIGCOMM. 2009. 51−62. [doi: 10.1145/1592568.1592576]

[4] 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. 39−50.

[5] Li D, Guo C, Wu H, Zhang Y, Lu S. Ficonn: Using backup port for server interconnection in data centers. In: Proc. of the IEEE INFOCOM. 2009. 2276−2285. [doi: 10.1109/INFCOM.2009.5062153]

[6] 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. In: Proc. of the SIGCOMM. 2009. 63−74. [doi: 10.1145/1592568.1592577]

[7] Guo D, Chen T, Li D, Liu Y, Chen G. Expansible and cost-effective network structures for data centers using dual-port servers. IEEE Trans. on Computers, 2013,62(7):1303−1317.

[8] Abu-Libdeh H, Costa P, Rowstron A, Shea G, Donnelly A. Symbiotic routing in future data centers. In: Proc. of the SIGCOMM 2010. New York: ACM Press, 2010. 51−62. [doi: 10.1145/1851182.1851191]

[9] Singla A, Hong C, Popa L, Godfrey P. Jellyfish: Networking data centers randomly. In: Proc. of the NSDI 2012. Berkeley: USENIX Association, 2012. http://dl.acm.org/citation.cfm?id=2228322

[10] Shin I, Wong B, Sirer E. Small-World data centers. In: Proc. of the ACM SOCC 2011. New York: ACM Press, 2011. http://dl.acm.org/citation.cfm?id=2038918 [doi: 10.1145/2038916.2038918]

[11] Gyarmati L, Trinh T. Scafida: A scale-free network inspired data center architecture. ACM SIGCOMM Computer Communication Review, 2010,40(5):5−12. [doi: 10.1145/1880153.1880155]

附中文参考文献:

[1] 陈贵海,吴盼,杨盘隆.数据中心网络.中国计算机学会通讯,2011,7(7):21−26.

朱桂明(1981−),男,江苏兴化人,博士,工程师,主要研究领域为数据中心网络,对等计算.

E-mail: guiming.zhu@gmail.com

陆菲菲(1981−),女,博士生,主要研究领域为数据中心网络.

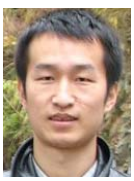
E-mail: Lu.feifei@meac-skl.cn

谢向辉(1958−),男,博士,研究员,博士生导师,主要研究领域为计算机体系结构.

E-mail: xie.xianghui@meac-skl.cn

陶志荣(1969−),女,高级工程师,主要研究领域为信息处理.

E-mail: tao.zhirong@meac-skl.cn



郭得科(1980−),男,博士,副教授,主要研究领域为对等计算,数据中心网络.

E-mail: guodeke@gmail.com