

在 BCube 型拓扑中嵌入环结构 *

任方俊, 邓倩妮

(上海交通大学 计算机科学与工程系, 上海 200240)

摘 要: 在数据中心网络(DCN)中, 为了实现 BCube 拓扑与基于环的应用的对接, 利用互连网络与组合数学的知识, 研究了在 BCube 中嵌入环(ring)结构的问题, 提出了基于最小异维环组和递归化的算法。该算法找到了 $BCube(n, k)$ (n 为偶数且 $k \geq 1$) (简记为 $B(\text{even}, k \geq 1)$) 中的 Hamilton 圈, 能保证嵌入图的膨胀率是 1; 而且在 BCube 中的 switch 发生故障时, 相对其他环嵌入算法, 嵌入的膨胀率较小。针对 $BCube(n, k)$ (n 为奇数且 $k \geq 1$) (简记为 $B(\text{odd}, k \geq 1)$), 也提出了可供参考的环化算法。

关键词: 数据中心网络; BCube 拓扑; 环化; 图嵌入; 最小异维环组; 广义超立方体

中图分类号: TP393.1 **文献标志码:** A **文章编号:** 1001-3695(2011)06-2280-06

doi:10.3969/j.issn.1001-3695.2011.06.076

Embedding ring in BCube

REN Fang-jun, DENG Qian-ni

(Dept. of Computer Science & Engineering, Shanghai Jiaotong University, Shanghai 200240, China)

Abstract: In DCN with BCube topology, it's hard to implement applications based on ring topology. So this paper researched the problem of embedding ring into BCube. Referring to the knowledge of interconnection network and combinatorial mathematics, proposed the method which was based on the SDRG and recursion to support the ring embedding algorithm. The algorithm could find a Hamilton cycle in $BCube(n, k)$ where n was even and $k \geq 1$. The expansion of the embedded ring was 1. On the occasion that some servers or switches fail, the expansion of the embedded ring was relatively lower compared with the other ring embedding algorithms. Also proposed a certain ring embedding algorithm for $BCube(n, k)$, where n was odd and $k \geq 1$.

Key words: data center networking (DCN); BCube; ring embedding; graph embedding; smallest different-dimension ring group (SDRG); generalized hypercube

1 研究背景

近些年来, 大型数据中心正成为众多流行的在线应用服务的载体。数据中心网络(DCN)成为了研究热点。DCN 作为数据中心的网络架构, 通过高速链路(wire)和交换机(switch)将大量服务器(server)连接起来。为了获得更高的性价比, 目前的数据中心中, 商品级的计算机和 switch 逐渐取代了专用的高端 server 和互连结构^[1]。现在很多 DCN^[2]使用传统的树型结构(tree), 依靠交换机、核心交换机、核心路由器将服务器连接起来, 实现服务器之间、服务器与存储设备之间的并发通信。

模块化数据中心(modular data center, MDC)是构建和部署数据中心的新途径^[3,4]。模块化数据中心中, 为获得更高的渐进可扩展性、容错性和高聚集带宽, 近几年提出了一些新的 DCN 体系结构, 如 Fat-tree^[5]、DCell^[6]、BCube^[7]、VL2^[8]等。Fat-tree、VL2 都是基于树型结构的拓扑结构; 而 DCell、BCube 则是以服务器为中心的体系结构。BCube 具有高渐进可扩展性、容错性和高聚集带宽, 特别是它满足了 MDC 对容错性的高要求, 当出现部分 switch 或 server 失效时, BCube 的通信性能降低的速率很小。

MDC 支持各种类型的应用, 如大规模存储(Amazon 的 S3^[9]和 Dynamo^[10])、高性能计算(并行算法、MPI 消息传递程序^[11])和数据密集型计算(MapReduce^[12])。其中许多分布式

应用, 如分布式存储采用的 DHT 算法^[10,13]、MPI 集合通信, 其进程与进程间的通信关系可抽象成环, 或者是其他规则拓扑如 Mesh 或超立方体。当数据中心网络的拓扑结构与部署在其上的分布式进程的通信结构相匹配时, 可以大大降低网络资源的开销。所以研究如何将各种规则拓扑图嵌入到系统底层的物理拓扑图具有重要意义。

图嵌入问题(graph embedding problem)^[14]研究一个拓扑结构(逻辑的, 称为客图, 其连接称为 link)到另一个拓扑结构(物理的, 称为主图, 其连接称为 wire)的映射, 并保留某些被要求的性质。刻画嵌入优劣的参数有膨胀数(dilation)、膨胀率(expansion)、拥塞(congestion)、负载(load)。设 ϕ 是客图 G 到主图 H 的嵌入。在这个嵌入中, G 的边被“拉长”的最大长度称为 ϕ 的膨胀数, 记为 $dil(\phi)$ 。比率 $v(H)/v(G)$ 称为嵌入 G 到 H 的膨胀率。嵌入的拥塞是客图中用到主图某一条边的最大次数。负载是客图中用到主图某一顶点的最大次数。

本文研究了在 BCube 中嵌入环状结构(ring embedding)的问题, 即以环为客图, BCube 为主图。本文分别给出了适用于 $B(\text{even}, k)/B(\text{odd}, k)$ 的环化算法。这两个算法有两个特点: 一是实现了膨胀率较小的图嵌入(其中找出了 $B(\text{even}, k)$ 的 Hamilton 圈), 物理拓扑和虚拟拓扑的结合度非常好; 二是在环中每个节点与其前后两个节点所连的 switch 不相同(本文称之为“异维”, 2.2 节), 保证了比较好的容错性。

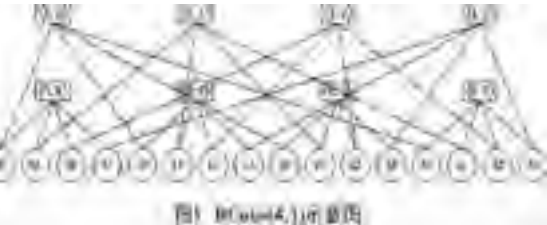
收稿日期: 2010-11-11; 修回日期: 2010-12-19 基金项目: 国家“863”计划资助项目(2009AA012201)

作者简介: 任方俊(1985-), 男, 山东淄博人, 硕士, 主要研究方向为云计算、数据中心等(renfangjun@gmail.com); 邓倩妮(1973-), 女, 副教授, 博士, 主要研究方向为云计算、数据中心、分布式计算等。

2 基本概念和问题表述

2.1 BCube 结构

定义 1 BCube BCube 中有两种设备,即多口 server(图 1 中以圆圈表示)和能连接常数个 server 的 switch(图 1 中以长方形表示)。BCube 是一种递归定义的结构^[7]。一个 BCube($n,0$)是连在一个 n 口 switch 上的 n 个 server(如图 1 中的 00,01,02,03 及 $\langle 0,0 \rangle$),其中 n 可以称为 BCube 的基数。一个 BCube($n,1$)是由 n 个 BCube0 和 n 个 n 口 switch 构成。一般地,一个 BCube($n,k \geq 1$)是由 n 个 BCube($n,k-1$)和 n 个 n 口 switch 构成的,共含有 $N = n^{k+1}$ 个节点。易得:一个 BCube(n,k)共有 $k+1$ 层 switch,其中每层中有 n^k 个 n 口 switch。



BCube 中的每个 server 都有一个 ID 号,这个 ID 号是一个 $k+1$ 位 n 进制的数,用 $ID = A_k A_{k-1} \cdots A_1 A_0$ 来表示。两个 server 编号有且只有一位不同时(设为第 L_i 位),这两个 server 相邻,即这两个 server 通过同一个 switch(在第 L_i 层)连接,两个 server 之间的连接称为第 L_i 维边。

在 BCube(n,k)中, $k+1$ 代表维度;物理拓扑中 switch 的层数是 $k+1$;每个 server 有 $k+1$ 个端口,各个端口与不同层(level_0 ~ level_k)的 switch 相连。因此,在 BCube 的 server 的 ID 号中,第 $i(0 \leq i \leq k)$ 位代表该 server 的第 i 个端口连接到第 i 层的 switch,第 i 位的值 A_i 代表该 server 会与 switch 的第 A_i 个端口相连。因为 ID 中的各位是对等的,所以各层是对等的^[7]。

2.2 本文提出的将环嵌入 BCube 的问题

BCube 的应用背景是模块化数据中心(MDC)。模块化方式构建数据中心可以减少构建部署时间,减少制造和冷却成本。但由于 MDC 一旦部署后就很难维护,所以对于模块化数据中心的设计要求有很好的容错性。而 BCube 以 server 为中心,switch 当做交叉结构来连接 server 迭代地构建,正好满足了 MDC 的容错要求。但是,由于 BCube 还没有得到广泛的研究和运用,MDC 支持的各种实际应用并不能很好地迁移到 BCube 拓扑上去。而环形则是在 MDC 中得到了广泛实际运用和验证的拓扑结构,绝大多数的应用程序可以支持环形拓扑。所以,为了使用 BCube 拓扑的 MDC 与相关应用程序的良好对接,本文研究了在 BCube 中嵌入环的问题。

为了保证良好的性能,嵌入的环应该满足以下条件:

- a) n^{k+1} 节点,每个节点的编号都是一个 $k+1$ 位 n 进制数,从第 0 位到第 k 位,每位的可取值为从 0 到 $n-1$ 。所有编号的集合恰好是所有 $k+1$ 位 n 进制数的集合。
- b) n^{k+1} 个节点构成一个环,ring[] 是这组节点的一个环序列。
- c) ring[] 中相邻的两个节点 ring[i] 和 ring[$i+1$] 的编号有且只有一位不同。这个不同的位,称为 ring[i] 的“变动位”。若 ring[i] 的“变动位”是 $L_i(0 \leq L_i \leq k)$,则可得 ring[i] 与 ring[$i+1$] 由一个 L_i -switch 连接。

d) ring[] 中相邻两个编号的“变动位”不同(本文根据 BCube 结构特点提出的附加条件,即“异维”)。

例如在表 1 中,节点编号序列 10-12-22 满足异维规则,序列 22-20 不满足异维规则。

表 1 异维与同维						
	...	$i-2$	$i-1$	i	$i+1$	$i+2$
ring[]	...	10	12	22	20	21
变动位	...	0	1	0	0	...

同维时,设一个节点 B 与它的两个相邻节点的连接 A-B、B-C 均由相同的 switch 管理,当这个 switch 失效时,A-B、B-C 之间的连接都需要通过其他的 switch 和 server 中转,这样增加了其他 switch 和 server 的负载压力。

例如在图 1 中,嵌入到 BCube(4,1)中的一个环 ring₁ 可以是:00-01-11-12-22-23-33-32-02-03-13-10-20-21-31-30-00,也可以是环 ring₂:00-01-02-03-33-32-31-30-20-21-22-23-13-12-11-10-00。ring₁ 满足异维条件,而 ring₂ 不满足异维条件。当 switch $\langle 0,0 \rangle$ 出现故障时,ring₁ 中 00-01 和 02-03 这两条边的膨胀数是 3,而在 ring₂ 中有三条边 00-01,01-02,02-03 的膨胀数为 3。

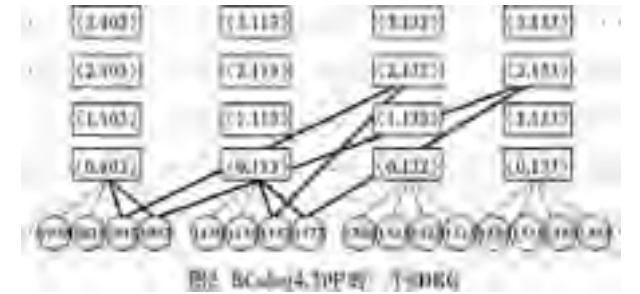
从上述分析得:满足异维的环化路径在容错性能上要优于同维环化路径。所以,本文提出的环化算法以异维为必要条件。

3 B(ev en, k) 的环化问题

3.1 定义

定义 2 最小异维环组(SDRG) 在 BCube(n,k)中,取第 s 层和第 t 层(不妨设 $s > t$), A_s 表示 server 编号的第 s 位, A_t 表示 server 编号的第 t 位。SDRG 中包含且仅包含四个节点,它们的 ID 号满足下列条件:a)除了 A_s 和 A_t 以外,其余各位对应相同;b) A_s 取值有两种(每种出现于两个 ID 号中),如 i,j,A_t 取值也有两种(每种出现于两个 ID 号中),如 $p,q(0 \leq i,j,p,q < n)$ 。

SDRG 中的四个节点(如 $a_k a_{k-1} \cdots a_{s+1} i a_{s-1} \cdots a_{t+1} p a_{t-1} \cdots a_1 a_0$ 、 $a_k a_{k-1} \cdots a_{s+1} i a_{s-1} \cdots a_{t+1} q a_{t-1} \cdots a_1 a_0$ 、 $a_k a_{k-1} \cdots a_{s+1} j a_{s-1} \cdots a_{t+1} p a_{t-1} \cdots a_1 a_0$ 、 $a_k a_{k-1} \cdots a_{s+1} j a_{s-1} \cdots a_{t+1} q a_{t-1} \cdots a_1 a_0$)组成一个以 switch 为连接中介的 4-圈。SDRG 满足异维规则,是下文算法部分构建 ring[] 的基本单元。如图 2 所示,图中加粗部分为 BCube(4,3)中的一个 SDRG:1 032,1 033,1 132,1 133。



定义 3 邻接点集 在 BCube 中,如果一个 SDRG(设为 SDRG₁)中的两个节点与另一个 SDRG(设为 SDRG₂)中的两个节点与同一 switch(不妨设为第 L_i 层, $0 \leq L_i \leq k$)相连,则称两个 SDRG 相邻于第 L_i 层,这四个节点称为 SDRG₁ 和 SDRG₂ 的邻接点集,记为 Interface(SDRG₁,SDRG₂)。

例如 *BCube(4,3)中,设 SDRG₁ = (1 032,1 033,1 132,1 133),SDRG₂ = (1 232,1 332,1 312,1 212),则 SDRG₁ 和 SDRG₂ 相邻于第 2 层,邻接点集为(1 032,1 132,1 232,1 332)。

在下文的环化算法中,邻接点集用于为两个 SDRG/环的合并提供接口。

定义 4 subBCube BCube(n,k)的一个 d 维的 subBCube 包含且仅包含 ID 号($A_k A_{k-1} \cdots A_1 A_0$)满足下列条件的节点:a)有 d 个 A_i 为 * (其中“*”表示无关位);b)其余各位对应相同,均为从 0 到 $n-1$ 中的一个确定值。

例如 $n=2$ 时,*01*0 表示 BCube(2,5)中的一个二维的 subBCube,它包含的节点有 00100,00110,10100,10110。subBCube 是子图^[15,16]的一种。

定义 5 最小环化单元 BCube(n,k)的所有 subBCube 中,可以完成异维环化的且维度数最小的 subBCube 称为 BCube(n,k)的最小环化单元。

本文中 $B(\text{even},k)$ 的最小环化单元取 $B(n,1)$ 。另外,根据 2.1 节,在 ID 的 $k+1$ 位中任意选取的两位是对等的,所以 BCube($n,1$)与二维 subBCube 同构(如无特别需要,下文不注明“同构体”),因而二维 subBCube 也可以作为最小环化单元。

3.2 环化算法

本节提出的算法基于 SDRG,算法分为最小环化单元的环化和递归完成多维 BCube 环化两部分。本算法适用于 $B(\text{even},k)$ 。

3.2.1 最小环化单元 $B(n,1)$ 的环化

1)划分最小异维环组 SDRG。因为 n 为偶数,所以设 $n=2m$,则 BCube($n,1$)中共含 $n^2=4m^2$ 个节点;由 BCube($n,1$)的结构特性可得,所有的节点可以分成 m^2 个 SDRG。在一个分别以 A_1 和 A_0 为列号和行号的表格中,一个 SDRG 中的四个节点呈“井”字分布。一个对 $B(\text{even},1)$ 必然可行的划分方法如表 2 所示(以 BCube(6,1)为例),其中:SDRG[0]=(00,01,11,10), $\cdots$,SDRG[7]=(42,43,53,52), $\cdots$ 。

表 2 BCube(6,1)划分 SDRG 的一种方法

SDRG[i]	A_1, level_1					
	0	1	2	3	4	5
A_0, level_0	0	0	0	3	3	6
	1	0	0	3	3	6
	2	1	1	4	4	7
	3	1	1	4	4	7
	4	2	2	5	5	8
	5	2	2	5	5	8

上述划分方法并不是唯一的。另一个可行的划分方法举例如表 3 所示,其中:SDRG[0]=(00,01,41,40),SDRG[1]=(02,04,24,22), $\cdots$,SDRG[7]=(21,23,33,31), $\cdots$ 。

表 3 BCube(6,1)划分 SDRG 的一种可行的方法

SDRG[i]	A_1, level_1					
	0	1	2	3	4	5
A_0, level_0	0	0	4	6	4	0
	1	0	3	7	7	0
	2	1	5	1	5	8
	3	2	3	7	7	2
	4	1	4	1	4	8
	5	2	5	6	5	2

但 SDRG 的划分方法也不是随意的。表 4 为不可行的划分方法举例,其剩余部分无法进行井字填充。

表 4 BCube(6,1)中不可行的 SDRG 划分方法

SDRG[i]	A_1, level_1					
	0	1	2	3	4	5
A_0, level_0	0	0	4	6	0	6
	1	0	3		0	3
	2	1	5	1	5	
	3	2	3		2	3
	4	1	4	1	4	
	5	2	5	6	5	6

2)利用 SDRG 成环,如算法 1 所示。

算法 1 ($B(\text{even},1)$ 的环化算法)

```
a) 初始化,将任意一个 SDRG 放入 ring;  
while(BCube( $n,1$ )划分所得的 SDRG 没有全部进入 ring)  
{  
    b) 找到 ring 中的 SDRG 的所有相邻 SDRG,从中选择一个(假设选  
       择的是 SDRG_u 的邻居 SDRG_v);  
    c) 在 Interface(SDRG_u,SDRG_v)中的节点处取断开点,将 SDRG_v  
       嵌入 ring;  
}
```

算法 1 实现了如下的环化性能:(a)可成环;(b)完整性,所有节点都嵌入 ring 而没有遗漏;(c)无复用,根据 BCube 的结构特性,异维处理保证了 server 和 wire 无复用;(d)膨胀数为 1,膨胀率为 1,拥塞为 1,负载为 1;(e)虚拟环对应于 $B(\text{even},1)$ 的一个 Hamilton 圈。所以,利用基于 SDRG 的算法,一定可以将 $B(\text{even},1)$ 环化。

以 BCube(4,1)为例,其环化过程如下:

- a)分为四个 SDRG:SDRG[0]=(00,02,22,20),SDRG[1]=(01,03,13,11),SDRG[2]=(10,12,32,30),SDRG[3]=(21,23,33,31)。
- b)BCube(4,1)_Ring=SDRG[0]=00-02-22-20-00。
- c)取 SDRG[0]的邻居 SDRG[3]嵌入,BCube(4,1)_Ring=00-02-22-23-33-31-21-20-00。
- d)取 SDRG[0]的邻居 SDRG[1]嵌入,BCube(4,1)_Ring=00-03-13-11-01-02-22-23-33-31-21-20-00。
- e)取 SDRG[1]的邻居 SDRG[2]嵌入,BCube(4,1)_Ring=00-03-13-12-32-30-10-11-01-02-22-23-33-31-21-20-00。〈完成〉

3.2.2 递归完成 $B(\text{even},k \geq 2)$ 的环化

当几个小的环形拓扑结构要合并成一个大的环形拓扑结构时,只要在重组点对处断开,并重新连接即可(图 3)。所谓重组点对,是指在一个逻辑环参与合并时,适合断开的边所连接的两个 server(它们物理相邻,也逻辑相邻,设它们在环中的逻辑序号为 l_id 和 l_id+1)。



BCube 是一种递归构成的结构,所以在最小环化单元 $B(n,1)$ 采用了相同的环化算法完成环化的基础上,可以通过如下递归环化的方式,完成 $B(\text{even},k \geq 2)$ 的环化。

1)为 $B(n,L_i-1)[j]$ 和 $B(n,L_i-1)[(j+1) \bmod(n)]$ (下文描述中类似情况默认取余)选取重组点对)方法如下:

a)从 $B(n,L_i-1)[j]$ 选取两个节点 $\text{ring}[l_id]$ 和 $\text{ring}[l_id+1]$,使得它们满足如下条件:(a)它们之间的物理边是非 0 维边(对应变动位不是第 0 位);(b)还未被当前逻辑环用做重组点对;(c)取满足(a)(b)的重组点对中 l_id 最小的。不妨设 $\text{ring}[l_id] = a_k a_{k-1} \cdots a_{li} i a_{li-2} \cdots a_{i+1} p a_{i-1} \cdots a_0$, $\text{ring}[l_id+1] = a_k a_{k-1} \cdots a_{li} i a_{li-2} \cdots a_{i+1} q a_{i-1} \cdots a_0$ 。

b)从 $B(n,L_i-1)[j+1]$ 选取两个节点 $\text{ring}'[l_id]$ 和 $\text{ring}'[l_id+1]$,使得与 $\text{ring}[l_id]$ 和 $\text{ring}[l_id+1]$ 相对应,即 $\text{ring}'[l_id] = a_k a_{k-1} \cdots a_{li} (i+1) a_{li-2} \cdots a_{i+1} p a_{i-1} \cdots a_0$, $\text{ring}'[l_id+1] = a_k a_{k-1} \cdots a_{li} (j+1) a_{li-2} \cdots a_{i+1} q a_{i-1} \cdots a_0$,可得, $\text{ring}[l_id]$ 和 $\text{ring}'[l_id]$ 在 $\text{level}-L_i$ 物理相邻, $\text{ring}[l_id+1]$ 和 $\text{ring}'[l_id+1]$ 在 $\text{level}-L_i$ 物理相邻。

2) 利用重组点对将 $B(n, L_i - 1)[j]$ 和 $B(n, L_i - 1)[j + 1]$ 合并。

环化算法如算法 2 所示。

算法 2 (一组 $B(n, k - 1)$ _Ring 组成一个 $B(n, k)$ _Ring 环的算法)

```
a)  $B(n, k)$ _Ring  $\leftarrow B(n, k - 1)[0]$ _Ring;  
b) 设定下层单元数 CELL_COUNT; // 一般置为  $n$   
for ( $i = 0; i < \text{CELL\_COUNT}; i++$ )  
{  
c) 为  $B(n, k - 1)[i]$  和  $B(n, k - 1)[i + 1]$  选取重组点对, 设为 ring  
[l_id] 和 ring[l_id + 1], ring'[l_id] 和 ring'[l_id + 1];  
d)  $B(n, k)$ _Ring 中, 断开 ring[l_id] 和 ring[l_id + 1] 间的 link;  
 $B(n, k - 1)[i + 1]$  中, 断开 ring'[l_id] 和 ring'[l_id + 1] 间的 link;  
e) 建立 ring[l_id] 和 ring'[l_id] 间的 link; 建立 ring[l_id + 1] 和  
ring'[l_id + 1] 间的 link;  
}  
〈完成〉
```

由算法 2 得到的嵌入图, 膨胀数为 1, 膨胀率为 1, 拥塞为 1, 负载为 1。可见算法 2 充分利用了 BCube 的资源配置和结构特点。综上可得, 任意的完整 $B(\text{even}, k \geq 2)$ 都可以环化。

图 4 为 BCube(2, 2) 的环化。其中单箭头虚线部分为 BCube(2, 1) 环化中使用但 BCube(2, 2) 环化中不使用的 wire; 双箭头实线部分反之。

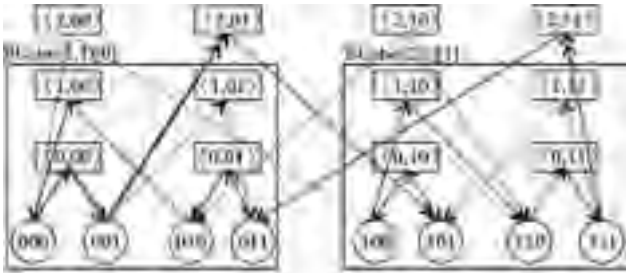


图 4 由 $B(\text{even}, k = 2)$ 的环化示例 (即 $B(\text{even}, k = 2)$) 的环化嵌入图

- 从 BCube(2, 1) 到 BCube(2, 2) 的递归环化步骤为:
- a) $\text{BCube}(2, 1)[0]$ _Ring = 000 - 010 - 011 - 001 - 000, $\text{BCube}(2, 1)[1]$ _Ring = 100 - 110 - 111 - 101 - 100;
 - b) $\text{BCube}(2, 1)[0]$ 和 $\text{BCube}(2, 1)[1]$ 的重组点集: ring[l_id] = 000, ring[l_id + 1] = 010, ring'[l_id] = 100, ring'[l_id + 1] = 110;
 - c) $\text{BCube}(2, 2)$ _Ring = $\text{BCube}(2, 1)[0]$ _Ring = 000 - 010 - 011 - 001 - 000;
 - d) $\text{BCube}(2, 2)$ _Ring = 000 - 〈100 - 101 - 111 - 110〉 - 010 - 011 - 001 - 000. 〈完成〉

3.3 $B(\text{even}, k \geq 1)$ 环化的模拟实验

3.3.1 实验环境

本实验使用网络模拟工具 NS2 来模拟数据中心网络的运行。首先, 用 NS2 模拟多个 server 与 switch 之间通过 BCube 网络进行互连, 形成一个具有 $B(\text{even}, k)$ 拓扑的数据中心; 然后, 在 $B(\text{even}, k)$ 上进行环嵌入, 分别采用 SDRG 算法、随机环化法来建立虚拟环; 最后, 在上面部署一个基于环的分布式存储系统, 并模拟系统中节点间的备份操作: 每个 server 向逻辑环上的后续两个 server 发送数据 (数据的备份)。

此模拟实验中网络参数的设置如下: 包的大小为 1 000 Byte, 发送速度为 1 Mbps, 网络延迟时间为 5 ms。

3.3.2 实验原理

本实验通过测量备份操作对物理网络造成的通信压力 (以 Byte 为单位), 对 SDRG 算法和随机环化法进行性能上的

比较。SDRG 算法即本文的算法 1。随机环化法生成环化序列的过程是: 将 $\text{BCube}(n, k)$ 中的 $n \wedge (k + 1)$ 个节点, 组合成一个随机序列并首尾相连, 即得到虚拟环。

本实验分为两个部分。实验 1: 在无 switch 失效的条件下, 在不同的 $\text{BCube}(n, k)$ 结构上进行两种方法的对比。实验 2: 在有 switch 失效的条件下, 对两种方法在不同 switch 失效率下的 $B(4, 3)$ 上的表现进行对比。

3.3.3 实验结果

实验 1 的数据如表 5 所示, 实验 2 的数据如表 6 所示。

表 5 完整 $B(\text{even}, k)$ 上两种环化方法的对比		
BCube 结构	SDRG 算法	随机环化法
BCube(2, 1)	256 000	358 400
BCube(4, 1)	880 000	1 449 600
BCube(6, 1)	1 920 000	3 331 200
BCube(2, 3)	880 000	1 913 600
BCube(4, 3)	13 362 000	40 273 600
BCube(6, 3)	67 440 000	224 095 600

表 6 失效条件下, $B(4, 3)$ 上两种环化方法的对比		
失效数/失效率/%	SDRG 算法	随机环化法
10/3.9	14 422 800	40 407 600
20/7.8	15 607 600	40 461 200
30/11.7	16 834 800	40 741 200
40/15.6	18 104 400	40 845 200

3.3.4 分析与结论

由表 5 可知, 由 SDRG 算法得到的虚拟环, 在实际运用中充分利用了 BCube 结构的特点, 减小了通信数据对物理网络造成的压力, 明显优于随机环化法。由表 6 可知, 由 SDRG 算法得到的虚拟环, 在有一定数目的 switch 失效的情况下, 依然可以确保通信压力比较小。这些结果与 BCube 的结构特点是一致的, 也验证了前文理论分析的正确性。

4 用于 $B(\text{odd}, k)$ 环化的附加算法

上文提出的基于 SDRG 的算法 1 不适用于 $B(\text{odd}, k)$ 的环化, 因为无法为 $B(\text{odd}, k)$ 划分 SDRG。例如 $\text{BCube}(3, 1)$ 有 9 个节点, 不是 4 的倍数, 无法井字填充, 如表 7 所示。本文猜想: $B(\text{odd}, k)$ 无法实现本文上述标准 (包含所有节点, wire 不复用) 的环化。

表 7 $B(\text{odd}, k)$ 无法划分 SDRG			
SDRG[i]	A_1, level_1		
	0	1	2
A_0, level_0	0	0	0
	1	0	0
	2		

所以, 本章将提出算法 2, 可以在算法 1 的基础上, 通过复用 wire 的方式, 将 $B(\text{odd}, k \geq 1)$ 环化。

4.1 定义和性质

定义 6 $L_i\text{-}B(n, 1)$ 对于一个 $B(n, 1)$ 的同构体, 如果其两层 switch 中的较高层 switch 在第 L_i 层, 较低层 switch 为第 0 层 switch, 则称之为 $L_i\text{-}B(n, 1)$ 。

性质 1 $B(n, 2)$ 中, 设 s 为一个特定值, 且 $0 \leq s < n$ 。从 $B(n, 2)$ 中的每个 $1\text{-}B(n, 1)$ 中, 取其第 s 个 $B(n, 0)$ (记为 $B(n, 0)[s]$, 称为“剥离 $B(n, 0)$ ”), 则得到的 n 个 $B(n, 0)[s]$ (共 $n \wedge 2$ 个 server) 及连接它们的 switch 和 wire (下文中如无需要, 不特别指出 switch 和 wire), 可构成一个 $2\text{-}B(n, 1)$ 。而且这个 $2\text{-}B(n, 1)$ 在 level_0 和 level_2 与 $B(n, 2)$ 中的其他元素互相

独立。

把 BCube 的 ID 的第 0 位和第 2 位遍历,其余位取确定值,即可证明上述性质。

图 5 为性质 1 的图示,其中加粗部分为所得的 $2_B(n,1)$ 。



4.2 环化算法

4.2.1 $B(\text{odd},1)$ 的切分

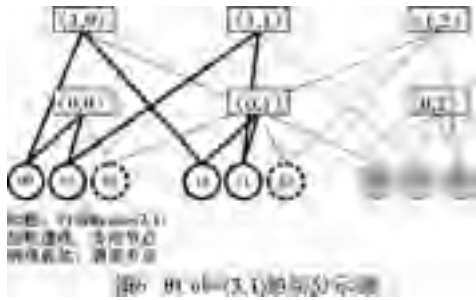
利用性质 1,取 $s = n - 1$,则 $B(n,0)[n - 1]$ 就是剥离 $B(n,0)$ 。根据性质 1,可以将每个 $1_B(n,1)$ 切分成一个剥离 $B(n,0)$ 和 $n - 1$ 个非剥离 $B(n,0)$ 。其中,剥离 $B(n,0)$ 中,对应 server 的 ID 为 $a_k a_{k-1} \cdots a_2 a_1 *$, $a_1 = n - 1$ 。非剥离 $B(n,0)$,即 $B(n,0)[i], 0 \leq i \leq n - 1$,共有 $n - 1$ 个 $B(n,0)$, $n * (n - 1)$ 个 server。对应 server 的 ID 为 $a_k a_{k-1} \cdots a_2 a_1 *$,其中 $0 \leq a_1 < n - 1$ 。

4.2.2 $B(\text{odd},1)$ 中非剥离 $B(n,0)$ 的环化

对每个非剥离 $B(n,0)$,将物理 ID 最大的一个 server 作为备用节点,即 $a_k a_{k-1} \cdots a_2 a_1 (n - 1)$,其中 $0 \leq a_1 < n - 1$ 。 $B(\text{odd},1)$ 中的 $n - 1$ 个非剥离 $B(n,0)$,除 $n - 1$ 个备用节点外的所有 server,构成一个内嵌 $B(\text{odd} - 1,1)$ 。内嵌 $B(\text{odd} - 1,1)$ 属于 $B(\text{even},1)$,所以可以用算法 1 来环化。并可以用算法 2 (其中,CELL_COUNT = odd) 来实现相同 level 的内嵌 $B(\text{odd} - 1,1)$ 所对应环的递归合并。

非剥离 $B(n,0)$ 中的其他节点,也就是备用节点,在必要时可以通过 wire 复用来环化。

图 6 表示了 BCube(3,1) 中的非剥离节点 (含内嵌 $B(2,1)$ 与备用节点) 与剥离节点。



4.2.3 剥离 $B(n,0)$ 的处理及层次化

根据性质 1,如上对 $1_B(n,1)$ 进行切分得到 $2_B(n,1)$ 后,可以对 $2_B(n,1)$ 进行切分得到 $3_B(n,1)$,及 $4_B(n,1) \cdots k_B(n,1)$,并分别切分。

根据 3.3.3 节,对任一 $L_i (1 \leq L_i \leq k)$,所有 $L_i_B(n,1)$ 切分得到的非剥离 $B(n,0)$ 都可以环化 (所得环记为 $L_i_B(n,1)_Ring$),并根据算法 2 最终合并成一个 L_i_Ring 。所以,共得到大小不同的 k 个环。最大的是 1_Ring ,最小的是 k_Ring 。

图 7 表示了 BCube(3,2) 中的 1_Ring 和 2_Ring 。



根据以上的讨论,可以得到算法 3。

算法 3 (各层 $L_i_B(n,1)$ 的环化)

```
a) for ( $L_i = 0; L_i < = k; L_i++$ )
{
    b) 若  $L_i = 0$ ,则取所有  $0\_B(n,1)$ ;若  $i > 0$ ,则取来自  $(L_i - 1)\_B(n,1)$  的  $L_i\_B(n,1)$ 。
    c) 对  $L_i\_B(n,1)$ ,将其中的  $B(n,0)$  或同构体切分为剥离节点/非剥离节点。
    d) 对非剥离节点,调用算法 1、算法 2;如有需要,则复用 wire 把备用节点加入环中。
    e) if ( $L_i < k$ ) 对剥离节点,构造  $(L_i + 1)\_B(n,1)$ 。
    f) if ( $L_i = k$ ) 对剥离节点,当做备用节点处理。
}
<完成>
```

4.2.4 各个环的合并及收尾处理

根据性质 1, $(L_i + 1)_B(n,1)$ 来自于 $L_i_B(n,1)$, $(L_i + 1)_Ring$ 中的 $B(n,0)$ 通过 level1_switch 与 L_i_Ring 中的 $B(n,0)$ 物理相连。而且上文的算法中,并没有第 0 维的边断开。所以,可以在这些 $B(n,0)$ 中选取第 0 维边所连的点对作为重组点对,并通过 level1_switch 重组。

选取重组点对的方法如下:

- a) 从 $(L_i + 1)_Ring$ 中选取第一个由第 0 维边连接的点对,作为重组点对,设为 $(L_i + 1)_Ring[l_id]$ 和 $(L_i + 1)_Ring[l_id + 1]$ 。
- b) 从 $(L_i + 1)_Ring[l_id]$ 和 $(L_i + 1)_Ring[l_id + 1]$ 的所有第 1 维物理邻居中,选取一个点对,使得:(a) 这个点对未作过重组点对;(b) 这个点对与点对 $(L_i + 1)_Ring[l_id]$ 、 $(L_i + 1)_Ring[l_id + 1]$ 之间的 wire 的已使用次数最少。将这个点对记为 $L_i_Ring[l_id']$ 和 $L_i_Ring[l_id' + 1]$ 。

图 8 表示了 BCube(3,2) 中的 1_Ring 和 2_Ring 的合并。



根据以上讨论,可得算法 4。

算法 4 (各个 L_i_Ring 的合并)

```
a) for ( $L_i = 1; L_i < = k; L_i++$ )
{
    b) 在  $(L_i + 1)\_Ring$  和  $L_i\_Ring$  中,选取重组点对,分别为  $(L_i + 1)\_Ring[l\_id]$  和  $(L_i + 1)\_Ring[l\_id + 1]$ 、 $L_i\_Ring[l\_id']$  和  $L_i\_Ring[l\_id' + 1]$ 。
    c)  $(L_i + 1)\_Ring$  中,断开  $(L_i + 1)\_Ring[l\_id]$  和  $(L_i + 1)\_Ring[l\_id + 1]$  之间的 link;  $L_i\_Ring$  中,断开  $L_i\_Ring[l\_id']$  和  $L_i\_Ring[l\_id' + 1]$  之间的 link。
    d) 建立  $(L_i + 1)\_Ring[l\_id]$  和  $L_i\_Ring[l\_id']$  间的 link;建立  $(L_i + 1)\_Ring[l\_id + 1]$  和  $L_i\_Ring[l\_id' + 1]$  间的 link。
}
<完成>
```

由算法 4 得到的嵌入图,膨胀数为很小,膨胀率为 1,拥塞为 1,负载为 1,也较充分地利用了 BCube 的资源配置和结构特点,使得 $B(\text{odd},k)$ 也可以在实际应用中进行环化处理。综上可得,任意的完整 $B(n,k \geq 2)$ 都可以通过本文算法实现异维的、低嵌入参数的环化。

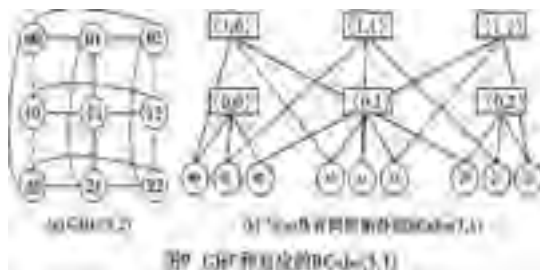
对比可得,从环化的角度看,构建 BCube 物理拓扑时,推荐采用 $B(\text{even},k)$ 。

5 相关工作

5.1 广义超立方体

广义超立方体(generalized hypercube, GHC)是BCube结构在图形几何中的拓扑原型。GHC($base, dim$)中, $base$ 是GHC的基数,表示一个维度中含有的节点数目,这些节点是两两直接相连的;节点编号中每位数字可取值的范围是 $0 \sim (base - 1)$; dim 是GHC的维度总数;节点编号共有 dim 位。

BCube作为一种适用于网络环境的拓扑结构,其与GHC的最大不同点是使用廉价switch代替了直接相连。如果GHC中一个维度下的 n 个节点间的连接方式,从全互连变换为用一个switch连接(例如图9(a)中00、01、02间的连接从全互连变换为(b)中用switch(0,0)连接),即可得到BCube结构。因为BCube和GHC在拓扑上是同质的,所以BCube具有GHC所具有的多数优良性质。BCube的直径是 $k+1$,在单播、多播等应用中性能良好;在容错方面,BCube网络的性能受个别server或switch失效的影响很小;在图嵌入方面,因为GHC是泛圈的^[15],所以BCube在wire可以复用 $n-1$ 次的假设下也是泛圈的。



5.2 在主图中嵌入环形拓扑

环形拓扑是一种大家熟知的互连网络拓扑结构。许多应用都是基于环形拓扑或在环形拓扑中得到了良好的实现,如分布式文件系统、基于工作流(workflow)的算法等。

Yu等人^[17]总结和提出了在一维、二维、三维的网格和tori结构中嵌入环形结构的算法,其实用环境为Blue Gene/L超级计算机,而不是数据中心。Tseng^[16]提出了一种容错算法,可以在二进制超立方体中嵌入环,环化算法基于逐级将 i 维展开为 $i-1$ 维。这种算法的本质是对维度进行递归和分类讨论。二进制超立方体仅相当于BCube(2, k)。Sengupta^[18]提出了在完整和存在失效的超立方体中嵌入环的算法,与文献^[16]的适用范围相近。Huang等人^[15]用递归和分类讨论的方法证明了基数大于3的GHC图是泛圈的。这个结论中的泛圈是强于可以找出Hamilton圈的,但其证明过程中允许相邻节点的变动位相同,不是异维的。

本文提出的算法针对的主图是BCube,其基数一般比超立方体大,而可利用wire数一般比GHC小。本文算法基于SDRG,使得最小环化单元可以有序、完全进入环化序列,而且本文在处理高维度BCube时所提出的递归环化过程也是简单递归,容易理解和使用。

6 结束语

本文给出了基于SDRG和递归环化的 $B(even, k)$ 环化算法(算法1),而且从膨胀率、拥塞等嵌入参数可以看出其嵌入性能非常优异。其中,对于环化步骤中的“SDRG的划分”,本文给出了一种对 $B(even, k)$ 一定可以成功的划分方法。下一步工作中将研究还有哪些可行的划分方法。

笔者认为: $B(odd, k)$ 无法实现异维Hamilton标准(包含所有节点, wire不复用)的环化。具体的证明将在下一步工作中实现。本文给出了可以将 $B(odd, k)$ 环化的参考算法(算法2),下一步将对算法2进行优化,以降低膨胀率。

物理拓扑的不完整将引入不完全BCube的研究。资源分配给多任务的情况则将引入在BCube中嵌入指定长度的环的研究。这些内容将在下一步工作中实现。

参考文献:

- [1] BARROSO L A, DEAN J, HÖLZLE U. Web search for a planet; the Google cluster architecture[J]. *IEEE Micro*, 2003, 23(2): 22-28.
- [2] Cisco data center infrastructure 2.5 design guide[EB/OL]. [2010-06-08]. <http://www.cisco.com/univercd/cc/td/doc/solution/dcldg21.pdf>.
- [3] HAMILTON J. An architecture for modular data centers[C]//Proc of Conference on Innovative Data Systems Research. 2007.
- [4] IBM. Scalable modular data center[EB/OL]. [2010-06-08]. <http://www-935.ibm.com/services/us/its/pdf/smdc-eb-sfe03001-us-en-00-022708.pdf>.
- [5] AL-FARES M, LOUKISSAS A, VAHDAT A. A scalable, commodity data center network architecture[C]//Proc of ACM SIGCOMM Conference on Data Communication. New York: ACM Press, 2008: 63-74.
- [6] GUO Chuan-xiong, WU Hai-tao, TAN Kun, et al. DCell: a scalable and fault-tolerant network structure for data centers[C]//Proc of ACM SIGCOMM Conference on Data Communication. New York: ACM Press, 2008: 75-86.
- [7] GUO Chuan-xiong, LU Guo-han, LI Dan, et al. BCube: a high performance, server-centric network architecture for modular data centers[C]//Proc of ACM SIGCOMM Conference on Data Communication. New York: ACM Press, 2009: 63-74.
- [8] GREENBERG A, HAMILTON J R, JAIN N, et al. VL2: a scalable and flexible data center network[C]//Proc of ACM SIGCOMM Conference on Data Communication. New York: ACM Press, 2009: 51-62.
- [9] Amazon. Amazon simple storage service (Amazon S3)[EB/OL]. [2010-06-08]. <http://s3.amazonaws.com>.
- [10] DeCANDIA G, HASTORUN D, JAMPANI M, et al. Dynamo: amazon's highly available key-value store[C]//Proc of 21st ACM SOSP'07. New York: ACM Press, 2007: 205-220.
- [11] Message Passing Interface Forum. MPI: a message-passing interface standard[J]. *International Journal of Supercomputer Applications*, 1994, 8(3/4): 159-416.
- [12] DEAN J, GHEMAWAT S. MapReduce: simplified data processing on large clusters[C]//Proc of Communications of the ACM. New York: ACM Press, 2008: 107-113.
- [13] RHEA S. The bamboo distributed hash table[EB/OL]. [2010-06-08]. <http://bamboo-dht.org/index.html>.
- [14] 徐俊明. 组合网络理论[M]. 北京: 科学出版社, 2007: 12-15.
- [15] HUANG C H, FANG J F. The pancconnectivity and the pancycle-connectivity of the generalized base-b hypercube[J]. *The Journal of Supercomputing*, 2009, 50(2): 162-176.
- [16] TSENG Y C. Embedding a ring in a hypercube with both faulty links and faulty nodes[J]. *Information Processing Letters*, 1996, 59(4): 217-222.
- [17] YU Hao, CHUNG I H, MOREIRA J. Topology mapping for Blue Gene/L supercomputer[C]//Proc of ACM IEEE Conference on Supercomputing. New York: ACM Press, 2006: 116.
- [18] SENGUPTA A. On ring embedding in hypercubes with faulty nodes and links[J]. *Information Processing Letters*, 1998, 68(4): 207-214.