数据中心中路由编码的可行性研究

丁炳辰[1],李卫忠[1],唐永康[2]
(1. 空军工程大学防空反导学院,陕西西安 710051; 2. 国防科学技术大学计算机学院,湖南长沙 410073)

摘 要: 修复带宽最优并不代表修复通信量也是最优的,后者与物理网络拓扑有着密切联系. 本文基于路由编码的思想减少修复通信量. 首先,基于信息流图对物理网络中数据的传递过程进行建模,证明得出了满足路由编码可行的充要条件,并发现路由编码可以基于再生码实现. 然后,针对数据中心网络设计的特点,为 Fat-tree 设计了一个工作在应用层的协议来生成修复树,为 CamCube 设计了一个启发式算法来生成修复树. 关于最小存储再生码的数据修复过程的仿真实验表明,路由编码可以有效地降低修复通信量,2 种修复树生成方案在各自适合的网络中均有较好性能:在帮助节点数较小时,Fat-tree 和 CamCube 中的修复通信量分别降低了大约 50% 和 30%.

关键词: 数据中心; 物理网络; 修复带宽; 修复通信量; 再生码
中图分类号: TP302.8 **文献标识码:** A **文章编号:** 0372-2112 (2017)11-2742-12
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2017.11.023

Feasibility Study of Routing Codes in Datacenters

DING Bing-chen[1], LI Wei-zhong[1], TANG Yong-kang[2]
(1. *Air and Missile Defense College, Air Force Engineering University, Xi'an, Shaanxi* 710051, *China*;
2. *College of Computer, National University of Defense Technology, Changsha, Hunan* 410073, *China*)

Abstract: Repair traffic is not always optimal when repair bandwidth is optimal. The former is relative to physical network topology. This paper aimed at reducing repair traffic based on routing codes. First, we modeled data transmission in physical networks based on information flow graph so that we could get the necessary and sufficient condition to feasibility of routing codes. And we found that routing codes could be realized based on regenerating codes. Then, we designed a protocol working on application layers to generate repair trees in Fat-tree, and a heuristic algorithm to generate repair trees in CamCube, which were both in agreement with their own design features of datacenter networks. Simulations about data-repair processes in systems using minimum-storage regenerating codes show that routing codes can reduce repair traffic efficiently, and performance of the two generation schemes of repair trees are both good in their own adapted networks. In fact, repair traffic had about 50% and 30% reductions in Fat-tree and CamCube respectively when the number of providers was small.

Key words: datacenter; physical networks; repair bandwidth; repair traffic; regenerating codes

1 引言

数据中心可以形成规模化效应,即较大的数据中心有更低的单位成本,所以数据中心通常包含上千台到上百万台的服务器,对于这种规模的服务器集群,由软件或硬件故障引起的存储节点离线、数据丢失等(统称为节点失效)已成为一种常态,而大型分布式存储系统的可靠性是数据中心的基础,因此数据中心需要存储冗余数据来提高可靠性.

随着数据中心的数据量以指数的趋势增长,简单的复制方式已难以适应数据中心对磁盘利用率和容错能力的需求[1]. 纠删码(erasure codes)[2]以其高存储利用率和容错能力的优势得到越来越广泛的关注与应用[3]. 但是传统的纠删码有较高的修复代价[4,5]. 例如,Facebook Analytics Hadoop cluster 对所存储的总数据的 8% 使用了 Reed-solomon 编码,而这 8% 的数据产生的修复通信量占了总网络通信量的将近 20%[6]. 这也成为了传统纠删码没有被存储系统广泛采用的主要原因.

收稿日期:2016-08-01;修回日期:2016-12-07;责任编辑:马兰英

Dimakis 等人[7]提出的再生码(regenerating codes)[8]达到了单节点存储容量与单节点修复带宽(repair bandwidth)的最优折衷(optimal tradeoff),通过连接更多的帮助节点降低了修复带宽.虽然再生码有最优的修复带宽,但并不意味着修复通信量(repair traffic)也是最优的.文献[7]基于星型逻辑网络进行分析,每个帮助节点与新节点直接相连,距离为 1 跳,此时的修复带宽等于修复通信量,但这是理想化的,实际中帮助节点与新节点之间的距离往往不止 1 跳.Zeng 等人[9]考虑了物理网络拓扑和路由器对修复时间和修复通信量的影响,提出了路由器执行编码操作的思想,使得路由器输出的数据量小于输入的数据量,但是没有讨论路由器中编码方式的可行性.Zhang 等人[10]也考虑了物理网络拓扑对修复通信量的影响,但是仅基于最大距离可分(maximum distance separable,MDS)编码,没有考虑帮助节点的增加对编码的可行性及修复通信量的影响.

本文围绕降低修复通信量的问题,指出了修复带宽与修复通信量的差别:修复带宽仅仅是修复过程中帮助节点生成的修复数据总量,它是修复通信量的下界,后者受数据中心的物理网络拓扑的影响.本文贡献有以下几点:

(1)将信息流图(information flow graph)模型拓展到更一般的情形,即物理网络中有中间节点的树状结构,通过分析证明得出了满足路由编码可行的充要条件,即,中间节点输出数据量的下界以及存储量与修复带宽的最优折衷.

(2)为两种设计方式不同的数据中心网络——Fat-tree 与 CamCube——分别设计了修复树的生成方案.

2　准备工作

相比容易发生节点暂时离线的对等分布式存储系统,数据中心中节点的规划都是良好的,发生单节点失效的概率远高于发生多节点失效的概率,例如在 Facebook warehouse cluster 中,单节点修复通常占 98% 以上[3,11].因此本文仅考虑单节点失效情形.

定义 1

(1)新节点:替代失效节点的可用的存储节点.失效节点上的数据需要在该节点上得以恢复.

(2)帮助节点:利用本地存储的数据帮助新节点恢复丢失数据的存储节点.

(3)修复数据:修复过程中帮助节点利用本地相关数据生成的用于修复的数据.

(4)修复带宽:所有修复数据的数据总量,记为 γ.

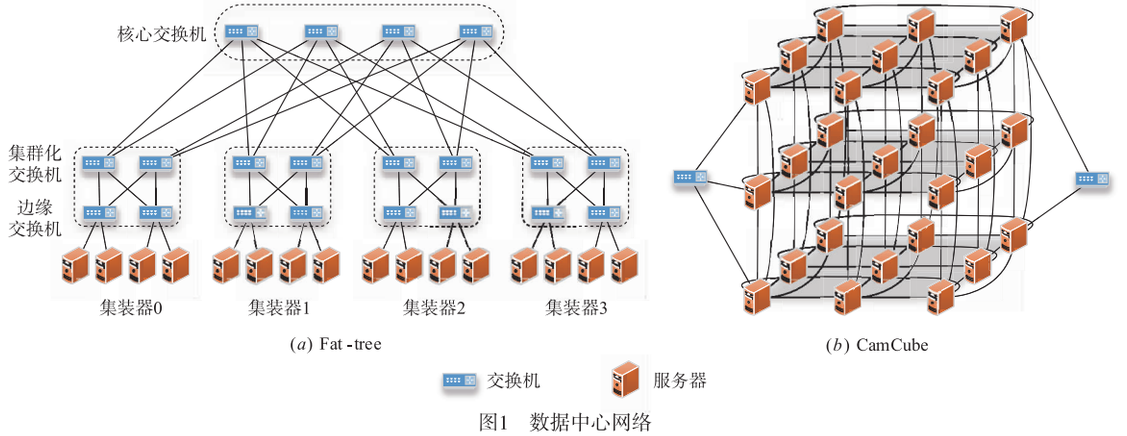
(5)修复通信量:所有修复数据经过网络中实际的链路最终到达新节点时产生的通信总量,记为 γ_t.

根据定义 1,当帮助节点与新节点间距离均为 1 跳时,$\gamma_t = \gamma$.但这是理想化的,物理网络中帮助节点与新节点之间的距离往往不止 1 跳.因此,γ 是 γ_t 的下界.

2.1　数据中心网络

数据中心的关键设计是集群中所有服务器之间的互联网络,其网络设计必须满足 5 个特殊要求:低延迟、高带宽、低成本、消息传递接口通信支持和容错[12].在数据中心中的任何两个服务器节点间应该建立多个路径,并通过在冗余服务器之间"复制"数据和计算来实现服务器容错,不应该存在会将整个系统拖垮的单点路径或单点故障.数据中心的网络主要有 2 种设计方式:

(1)以交换机为中心,交换机用于连接服务器,服务器不需要做任何修改.比如典型的 Fat-tree[13],如图 1(*a*)所示,服务器节点都在底层,边缘交换机用于连接服务器节点,集群化交换机、边缘交换机和它们的叶节点服务器构成一个集装器,核心交换机提供不同集装器间的路径.胖树结构在任何两个服务器之间提供了多条路径,相应的,路由表也提供了额外的路由路径.



(*a*) Fat-tree　　(*b*) CamCube

图1　数据中心网络

(2)以服务器为中心,该设计会修改运行在服务器上的操作系统,使用特殊的驱动程序来转发网络数据包,仍需要组织交换机来实现互连,如 BCube[14,15]. Costa 等人[16]进一步强化服务器在网络中的角色,提出了一种基于直接相连的 3D 环形网络结构 CamCube. 图 1(*b*)为 3×3×3CamCube(共 27 个服务器节点),每个服务器节点与其他 6 个服务器节点直接相连($n\times n\times n$ CamCube 的拓扑结构可以形式化为如图 1(*b*)所示的立方体,包括水平或垂直的 $3n$ 个平面,每个平面包含水平或垂直的 $2n$ 条边,同一条边上的节点直接相连). CamCube 内部的通信全由服务器来传送,交换机仅用来连接 CamCube 与外部网络.

2.2 信息流图

存储网络中数据的传递过程可以基于信息流图进行建模[7,17]. 信息流图是一个有向无环图,至少包含 3 类节点:数据源点 S,数据汇点 DC 以及存储节点. 数据源点 S 把经过编码的原数据分发给多个存储节点,数据汇点 DC 是希望读取原数据的节点. 由于存储节点的失效、修复以及 DC 的不唯一,对任意一种编码方案,都有相应的一系列的信息流图.

定义 2 任意的信息流图 G 的一条割$(U,\bar{U})$,$S\in U$,$DC\in\bar{U}$,其容量为从 U 指向 $\bar{U}$ 的所有边的容量的和,其中容量最小的割称为最小割. 编码方案对应的所有信息流图中所有割的容量的最小值用 C 表示.

图 2 是某一信息流图的示意图,边上的数字表示边的容量,图中的割包含边 $V_1\to V_2$、$V_4\to V_7$、$V_6\to V_7$,$U=\{S,V_1,V_3,V_4,V_5,V_6\}$,$\bar{U}=\{V_2,V_7,V_8,DC\}$,该割的容量为 3,该割也是此图的最小割.

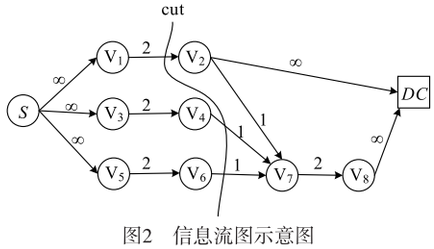


图2 信息流图示意图

引理 1[7,17] 对任意的信息流图 G,如果其最小割的容量小于原数据大小 M,则 DC 无法重构出原数据.

引理 2[7,17] 如果 C 不小于原数据大小 M,则存在线性网络编码使得任意的 DC 可以重构出原数据. 并且当有限域足够大时,随机线性编码保证 DC 以高概率重构出原数据.

由上述 2 个引理不难得到下列定理 1,并由定理 1 给出定义 3.

定理 1 任意的 DC 可以重构出原数据的充要条件是 C 不小于原数据大小 M.

定义 3 当任意 DC 可以重构出原数据时,称该编码方案是可行的.

2.3 (n,k,d)再生码

大小为 M 的原数据经由线性编码生成 n 个编码块,每个编码块大小为 α,分别放置在不同的存储节点,其中任意 k 个编码块足以重构(解码)出原数据. 任意一个节点失效后,新节点从其他可用节点中选择任意 d $(k\leqslant d\leqslant n-1)$个作为帮助节点. 每个帮助节点生成 β 大小的修复数据并传送给新节点,新节点利用修复数据在本地修复(解码)出新的编码块. 修复带宽 $\gamma=d\beta$.

可以求得再生码的函数 C 如下[7]:

$$C=\sum_{i=1}^{k}\min\{\alpha,(d-i+1)\beta\} \tag{1}$$

根据定理 1 和定义 3,有下列定理 2 成立:

定理 2[7] 对任意的 $\alpha\geqslant\alpha^*(n,k,d,\gamma)$,五元组$(n,k,d,\alpha,\gamma)$都是可行的,且线性网络编码可以满足下界. 理论上,当 $\alpha<\alpha^*(n,k,d,\gamma)$时,$(n,k,d,\alpha,\gamma)$是不可行的. 阈值函数 $\alpha^*(n,k,d,\gamma)$如下:

$$\alpha^*(n,k,d,\gamma)=\begin{cases}\dfrac{M}{k}, & \gamma\in[f(0),+\infty)\\ \dfrac{M-g(i)\gamma}{k-i}, & \gamma\in[f(i),f(i-1))\end{cases} \tag{2}$$

式中,$i=1,\cdots,k-1$,$k\leqslant d\leqslant n-1$

$$f(i)\triangleq\frac{2Md}{(2k-i-1)i+2k(d-k+1)}$$

$$g(i)\triangleq\frac{(2d-2k+i+1)i}{2d}$$

给定(n,k,d),最小修复带宽是

$$\gamma_{\min}=f(k-1)=\frac{2Md}{2kd-k^2+k}$$

$\alpha^*(n,k,d,\gamma)$的函数图像上有 2 个特殊点:

最小存储量点:

$$(\alpha_{\mathrm{MSR}},\gamma_{\mathrm{MSR}})=\left(\frac{M}{k},\frac{M}{k}*\frac{d}{d-k+1}\right) \tag{3}$$

最小修复带宽点:

$$(\alpha_{\mathrm{MBR}},\gamma_{\mathrm{MBR}})=\left(\frac{2Md}{2kd-k^2+k},\frac{2Md}{2kd-k^2+k}\right) \tag{4}$$

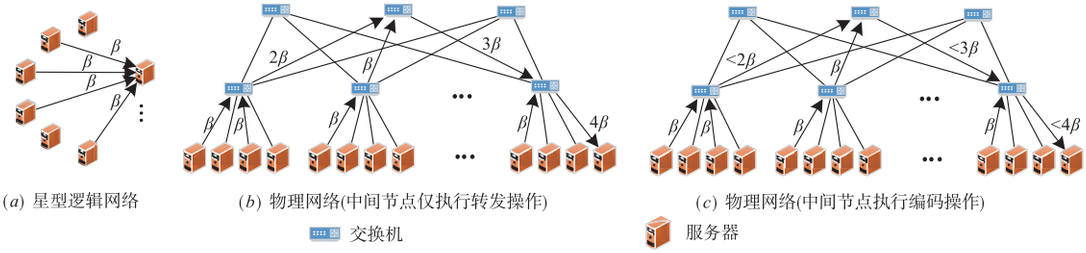
相应的编码方案称为最小存储再生码(minimum-storage regenerating codes,MSR codes)和最小带宽再生码(minimum-bandwidth regenerating codes,MBR codes).

2.4 路由编码

图 3(*b*)、图 3(*c*)是某一物理网络,图 3(*a*)是其对应的星型逻辑网络[18]. 大部分文献都是基于星型逻辑网络来分析修复过程(见图 3(*a*)),此时帮助节点与新节点直接相连,距离为 1 跳,修复带宽等于修复通信量;而在其对应的物理网络中(见图 3(*b*)),帮助节点

与新节点间的距离最小为 2 跳,最大可以是 4 跳,$\gamma_t=\beta+\beta+2\beta+\beta+\beta+3\beta+\beta+4\beta=14\beta,\gamma=4\beta$,前者是后者的 3.5 倍,可见物理网络拓扑对修复通信量的影响很大. 事实上,物理网络中帮助节点与新节点间的距离越大,修复通信量与修复带宽间的差距就越大.

物理网络中,修复数据在到达新节点之前几乎都会经过一个或多个交换机或服务器(除非是服务器直接相连的拓扑结构),称为中间节点. 若中间节点仅将接收到的数据转发出去而不作任何处理,此时修复通信量的最小值 γ_{t0} 在如图 3(*b*)所示的情形下(各个帮助节点生成的修复数据经过各自的最短路径到达新节点)达到. 如果中间节点可以执行编码操作使得输出数据量小于输入数据量(如图 3(*c*)),在路径合适的情况下,修复通信量会小于 γ_{t0},甚至取得 γ(γ 是 γ_t 的下界). 这种中间节点执行"转发 + 编码"操作的方案,称之为路由编码.



(*a*) 星型逻辑网络　(*b*) 物理网络(中间节点仅执行转发操作)　(*c*) 物理网络(中间节点执行编码操作)

图3　星型逻辑网络与物理网络

3　路由编码模型及其可行性

定义 4

(1)修复树:一个存储节点的修复过程的信息流图中,各个节点及其之间的边构成的一个以新节点为根的树.

(2)叶子节点:修复树中的叶子节点. 它们都是帮助节点.

(3)中间节点:修复树中叶子节点与根节点(即新节点)之间的存储节点. 其中,属于帮助节点的称为中间帮助节点,否则称为普通中间节点.

3.1　模型

大小为 M 的原数据经由线性编码生成 n 个编码块,每个编码块大小为 α,分别放置在不同的存储节点,其中任意 k 个编码块足以重构出原数据. 任意一个节点失效后,新节点可以从其他可用节点中选择任意 $d(k\leqslant d\leqslant n-1)$ 个作为帮助节点. 帮助节点 u 生成 β_u 大小的修复数据,如果 u 是叶子节点,则将修复数据直接输出;中间节点 w 利用接收到的数据(如果本地存在相关数据,则也要被利用,比如中间帮助节点)生成大小为 η_w 的数据并输出.(不同的 u(或 w),β_u(或 η_w)可以不同.)新节点利用修复数据在本地修复出新的编码块.

3.2　信息流图的构造

信息流图被 Dimakis 等人[7]首次用于分析编码方案是否可行的充分必要条件,其模型是星型的逻辑网络拓扑;Wang 等人[19]将信息流图扩展到一般的树状结构,允许帮助节点作为中间节点,但仍然是基于逻辑网络;我们需要将信息流图继续拓展到更一般的情形,即物理网络中有中间节点的树状结构.

图 4 为物理网络的信息流图的示意图(图中忽略了不同节点的 β_u、η_w 的差异,统一用 β、η 表示,后文信息流图构造方式的描述中也是如此),构造方式如下:

(1)叶子节点和新节点用 2 个节点 in、out 和 1 条容量为 α 的有向边 $\text{in}\xrightarrow{\alpha}\text{out}$ 来表示;当中间帮助节点生成 β 大小的修复数据时,用 3 个节点 in、out、ext 和 2 条容量分别为 α、β 的有向边 $\text{in}\xrightarrow{\alpha}\text{out}$、$\text{out}\xrightarrow{\beta}\text{ext}$ 来表示(可以看作在叶子节点的基础上增加了一个虚拟的中间节点 ext);普通中间节点仅用一个节点 ext 表示.

(2)当叶子节点 u 传输 β 大小的数据给中间节点 w 时,在 u^{out} 与 w^{ext} 之间加一条容量为 β 的有向边 $u^{\text{out}}\xrightarrow{\beta}w^{\text{ext}}$.

(3)当叶子节点 u 传输 β 大小的数据给新节点 v 时,在 u^{out} 与 v^{in} 之间加一条容量为 β 的有向边 $u^{\text{out}}\xrightarrow{\beta}v^{\text{in}}$.

(4)当中间节点 w_1 传输 η 大小的数据给中间节点 w_2 时,在 w_1^{ext} 与 w_2^{ext} 之间加一条容量为 η 的有向边 $w_1^{\text{ext}}\xrightarrow{\eta}w_2^{\text{ext}}$.

(5)当中间节点 w 传输 η 大小的数据给新节点 v 时,在 w^{ext} 与 v^{in} 之间加一条容量为 η 的有向边 $w^{\text{ext}}\xrightarrow{\eta}v^{\text{in}}$.

(6)在源点 S 和所有的初始存储节点 u_i 之间各加一条容量为无穷的有向边 $s\xrightarrow{\infty}u_i^{\text{in}}$,在汇点 DC 和其所连的 k 个存储节点 v_j 之间各加一条容量为无穷的有向边 $v_j^{\text{out}}\xrightarrow{\infty}DC$.

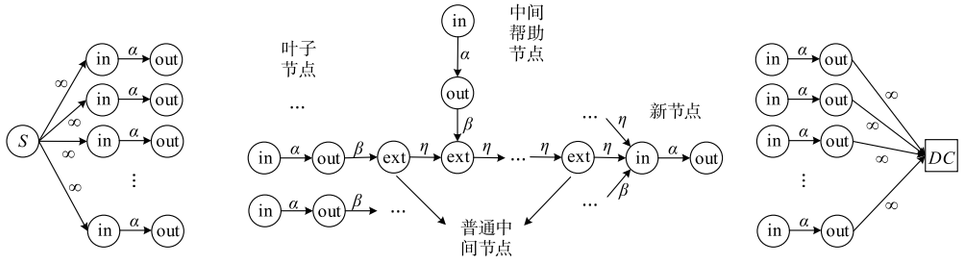


图4 物理网络的信息流图

3.3 路由编码可行的条件

对任意的信息流图 G,其最小割 mincut $(U,\overline{U})$,$S\in U$,$DC\in\overline{U}$,满足 $v_j^{\text{out}}\in\overline{U}$,$j\in I$,其中 v_j^{out},$j\in I$ 是 DC 所连接的 k 个 out 节点,所以 $\overline{U}$ 中至少存在 k 个 out 节点. 把 G 中的节点拓扑排序后(有向无回路图 $G=(V,E)$ 中,若存在边 $v_i\rightarrow v_j$,v_i、$v_j\in V$,则拓扑序列中 v_i 位于 v_j 之前),再把 $\overline{U}$ 中处于拓扑序列的前 k 个 out 节点按顺序重新编号为 v_i^{out},$i=1,\cdots,k$. 以 v_i^{in} 为根的修复树 T_i 中,一共有 d 个 out 节点,其中不属于 $\{v_1^{\text{out}},\cdots,v_{i-1}^{\text{out}}\}$ 的 out 节点构成集合 D,$d-i+1\leqslant|D|\leqslant d$. 设 W 是 w^{ext} 所在的修复树中以 w^{ext} 为根的子树中包含的 out 节点的集合.

定理 3 路由编码方案可行的充要条件是下列 2 个不等式同时成立:

$$\forall\, w^{\text{ext}}\qquad \eta_w\geqslant\min\left\{\alpha,\sum_{u^{\text{out}}\in W}\beta_u\right\}\tag{5}$$

$$\forall\, D\qquad \sum_{i=1}^{k}\min\left\{\alpha,\sum_{u^{\text{out}}\in D}\beta_u\right\}\geqslant M\tag{6}$$

证明 对于 T_i,考虑 3 种情形:

(1)T_i 中的点均属于 U;

(2)v_i^{in} 以及 T_i 中所有 ext 节点均属于 $\overline{U}$;

(3)$v_i^{\text{in}}\in\overline{U}$,并存在 T_i 中的 ext 节点属于 U.

第(1)种情况 v_i^{out}、T_i 对 mincut$(U,\overline{U})$ 的贡献为 α;第(2)种情况 v_i^{out}、T_i 对 mincut$(U,\overline{U})$ 的贡献至少为 $\min\limits_{D}\left(\sum\limits_{u^{\text{out}}\in D}\beta_u\right)$.

对于第(3)种情况,设 w^{ext} 是 T_i 中的任意一个 ext 节点,c 是包含 w^{ext} 的出边的任意一条割,c 还包含 B_c 中 ext 节点(设为 $w_1^{\text{ext}},\cdots,w_z^{\text{ext}}$)的出边和 D_c 中 out 节点的出边. B_c 是 T_i 中除 w^{ext} 以外的 ext 节点的集合的子集,D_c 是 D 的子集,B_c、D_c 均可能为 ϕ. v_i^{out}、T_i 对割 c 的贡献至少为 $\eta_w+\delta_c$,其中

$$\delta_c=\sum_{j=1}^{z}\eta_{w_j}+\sum_{u^{\text{out}}\in D_c}\beta_u\tag{7}$$

注意 $D\subset W\cup W_1\cup\cdots\cup W_z\cup D_c$,其中 W_j,$j=1,\cdots,z$ 是 T_i 中以 w_j^{ext} 为根的子树中包含的 out 节点的集合. 所以 v_i^{out}、T_i 对 mincut$(U,\overline{U})$ 的贡献至少为 $\min\limits_{w^{\text{ext}},c,D}(\eta_w+\delta_c)$.

因此,v_i^{out}、T_i 对 mincut $(U,\overline{U})$ 的贡献至少为 $\min\limits_{w^{\text{ext}},c,D}\left\{\alpha,\sum\limits_{u^{\text{out}}\in D}\beta_u,\eta_w+\delta_c\right\}$. (注意,上文是在 T_i 中定义的变量:w^{ext},c,D,所以它们隐含着变量 i;并且,由于 i 具有一般性,所以 w^{ext},c,D 隐含地代表了所有 T_i 下的相应变量.)所以

$$C\geqslant\sum_{i=1}^{k}\min_{w^{\text{ext}},c,D}\left\{\alpha,\sum_{u^{\text{out}}\in D}\beta_u,\eta_w+\delta_c\right\}\tag{8}$$

因为式(8)不等式右侧的值是通过上述步骤构造得出,因此式(8)可以取等号,所以

$$C=\sum_{i=1}^{k}\min_{w^{\text{ext}},c,D}\left\{\alpha,\sum_{u^{\text{out}}\in D}\beta_u,\eta_w+\delta_c\right\}\tag{9}$$

由定理 1 和定义 3 可知路由编码方案可行的充要条件是 $C\geqslant M$,即

$$\forall\, w^{\text{ext}},c,D\qquad \sum_{i=1}^{k}\min\left\{\alpha,\sum_{u^{\text{out}}\in D}\beta_u,\eta_w+\delta_c\right\}\geqslant M\tag{10}$$

下面证明:式(10) $\Leftrightarrow\begin{cases}\text{式(5)}\\ \text{式(6)}\end{cases}$.

$\Leftarrow$:当式(5)成立时,即

$$\forall\, w^{\text{ext}}\qquad \eta_w\geqslant\min\left\{\alpha,\sum_{u^{\text{out}}\in W}\beta_u\right\}$$

由式(7)可得

$$\delta_c\geqslant\sum_{j=1}^{z}\min\left\{\alpha,\sum_{u^{\text{out}}\in W_j}\beta_u\right\}+\sum_{u^{\text{out}}\in D_c}\beta_u$$

所以

$$\eta_w+\delta_c\geqslant\min\left\{\alpha,\sum_{u^{\text{out}}\in W}\beta_u\right\}+\sum_{j=1}^{z}\min\left\{\alpha,\sum_{u^{\text{out}}\in W_j}\beta_u\right\}+\sum_{u^{\text{out}}\in D_c}\beta_u$$

当上式至少有一个 min 函数取 α 时,有 $\eta_w+\delta_c\geqslant\alpha$,当 min 函数均不取 α 时,

$$\eta_w+\delta_c\geqslant\sum_{u^{\text{out}}\in W}\beta_u+\sum_{j=1}^{z}\sum_{u^{\text{out}}\in W_j}\beta_u+\sum_{u^{\text{out}}\in D_c}\beta_u$$

因为 $D\subset W\cup W_1\cup\cdots\cup W_z\cup D_c$,所以

$$\sum_{u^{\text{out}}\in W}\beta_u+\sum_{j=1}^{z}\sum_{u^{\text{out}}\in W_j}\beta_u+\sum_{u^{\text{out}}\in D_c}\beta_u\geqslant\sum_{u^{\text{out}}\in D}\beta_u$$

所以

$$\eta_w+\delta_w\geqslant\sum_{u^{\text{ext}}\in D}\beta_u.$$

即对 $\forall w^{ext},c,D$,有 $\eta_w+\delta_c\geqslant\alpha$ 或 $\sum_{u^{\text{ext}}\in D}\beta_u$,所以

$$\forall w^{\text{ext}},c,D\quad\sum_{i=1}^{k}\min\left\{\alpha,\sum_{u^{\text{ext}}\in D}\beta_u,\eta_w+\delta_c\right\}=\sum_{i=1}^{k}\min\left\{\alpha,\sum_{u^{\text{ext}}\in D}\beta_u\right\}\tag{11}$$

因为式(6)成立,所以

$$\forall w^{\text{ext}},c,D\quad\sum_{i=1}^{k}\min\left\{\alpha,\sum_{u^{\text{ext}}\in D}\beta_u,\eta_w+\delta_c\right\}\geqslant M,$$

即式(10)成立. ⇐得证.

⇒:采用反证法.

(1)当式(5)不成立,式(6)成立时.

$$\exists w^{\text{ext}}\quad\eta_w<\min\left\{\alpha,\sum_{u^{\text{ext}}\in W}\beta_u\right\},$$

则

$$\exists w^{\text{ext}}\quad\eta_w+\delta_c<\min\left\{\alpha,\sum_{u^{\text{ext}}\in W}\beta_u\right\}+\sum_{j=1}^{z}\eta_{w_j}+\sum_{u^{\text{ext}}\in D_c}\beta_u,$$

所以

$$\exists w^{\text{ext}}\quad\eta_w+\delta_c<\sum_{u^{\text{ext}}\in W}\beta_u+\sum_{j=1}^{z}\eta_{w_j}+\sum_{u^{\text{ext}}\in D_c}\beta_u.$$

因为 $\exists c$,例如 $W+D_c=D,B_c=\varnothing$,使得

$$\sum_{u^{\text{ext}}\in W}\beta_u+\sum_{j=1}^{z}\eta_{w_j}+\sum_{u^{\text{ext}}\in D_c}\beta_u=\sum_{u^{\text{ext}}\in D}\beta_u,$$

所以 $\exists w^{\text{ext}},c\quad\eta_w+\delta_c<\sum_{u^{\text{ext}}\in D}\beta_u$.

所以

$$\forall D,\exists w^{\text{ext}},c\quad\sum_{i=1}^{k}\min\left\{\alpha,\sum_{u^{\text{ext}}\in D}\beta_u,\eta_w+\delta_c\right\}<\sum_{i=1}^{k}\min\left\{\alpha,\sum_{u^{\text{ext}}\in D}\beta_u\right\}\tag{12}$$

又因为 $\exists D$,使得 $\sum_{i=1}^{k}\min\left\{\alpha,\sum_{u^{\text{ext}}\in D}\beta_u\right\}=M$,

所以

$$\exists w^{\text{ext}},c,D\quad\sum_{i=1}^{k}\min\left\{\alpha,\sum_{u^{\text{ext}}\in D}\beta_u,\eta_w+\delta_c\right\}<M,$$

与式(10)矛盾.

(2)当式(6)不成立,式(5)成立时.

$$\exists D\quad\sum_{i=1}^{k}\min\left\{\alpha,\sum_{u^{\text{ext}}\in D}\beta_u\right\}<M\tag{13}$$

因为当式(5)成立时,式(11)成立,所以

$$\exists D\quad\sum_{i=1}^{k}\min\left\{\alpha,\sum_{u^{\text{ext}}\in D}\beta_u,\eta_w+\delta_c\right\}<M.$$

与式(10)矛盾.

(3)当式(5)、(6)均不成立时.

由式(12)、(13)可知,

$$\exists w^{\text{ext}},c,D\quad\sum_{i=1}^{k}\min\left\{\alpha,\sum_{u^{\text{ext}}\in D}\beta_u,\eta_w+\delta_c\right\}<M$$

与式(10)矛盾.

⇒得证.

综上,路由编码方案可行的充要条件是式(5)、(6)同时成立.

证毕

定理3 给出了中间节点输出数据量的下界(式(5))以及存储量与修复带宽的最优折衷(式(6)). 由式(5)式可知,当 $\sum_{u^{\text{ext}}\in W}\beta_u\leqslant\alpha$ 时,$\min(\eta_w)=\sum_{u^{\text{ext}}\in W}\beta_u$,$w^{\text{ext}}$ 没有减少输出数据量,仅执行转发操作就可以满足;当 $\sum_{u^{\text{ext}}\in W}\beta_u>\alpha$ 时,$\min(\eta_w)=\alpha$,意味着只要 w^{ext} 的输入数据量大于 α,w^{ext} 就可以执行编码操作并仅输出 α 的数据量. 所以仅当 $\sum_{u^{\text{ext}}\in W}\beta_u>\alpha$ 时,中间节点 w 才有执行编码操作的必要.

当所有帮助节点输出的数据量相同时,即 $\forall u\ \beta_u=\beta$,有下列推论成立.

推论1 当 $\forall u\quad\beta_u=\beta$ 时,路由编码方案可行的充要条件是下式与式(2)同时成立,

$$\forall w^{\text{ext}}\quad\eta_w\geqslant\min\left\{|W|\beta,\alpha\right\}\tag{14}$$

证明 当 $\forall u\quad\beta_u=\beta$ 时,式(5)、(6)分别等价于:

$$\begin{aligned}&\forall w^{\text{ext}}\quad\eta_w\geqslant\min\left\{|W|\beta,\alpha\right\}\\&\sum_{i=1}^{k}\min\{\alpha,(d-i+1)\beta\}\geqslant M\end{aligned}\tag{15}$$

式(15)中不等式左边等于式(1),推得式(2)的过程见文献[7].

证毕

由推论1可知,可以基于再生码来实现所有帮助节点生成等量的修复数据的情形下的路由编码.

推论2 对于 MSR 编码,只有当 $|W|>d-k+1$ 时中间节点才有执行编码操作的必要;对于 MBR 编码,所有的中间节点均没有执行编码操作的必要.

证明 对于 MSR 编码有 $\alpha_{\text{MSR}}=(d-k+1)\beta_{\text{MSR}}$,所以 $|W|\beta_{\text{MSR}}>\alpha_{\text{MSR}}\Leftrightarrow|W|>d-k+1$;对于 MBR 编码有 $\gamma_{\text{MBR}}=\alpha_{\text{MBR}}$,因为 $\forall w\quad|W|\beta_{\text{MBR}}\leqslant\gamma_{\text{MBR}}$,所以 $\forall w\quad|W|\beta_{\text{MBR}}\leqslant\alpha_{\text{MBR}}$.

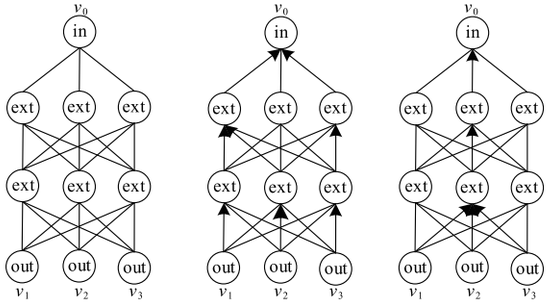
证毕

事实上,对于任意满足 $\gamma=\alpha$ 的编码,所有中间节点均没有执行编码操作的必要.

4 修复树的建立

图5(a)是网络中4个存储节点间的拓扑结构,原数据采用(n,3,3) MSR 编码,v_1、v_2、v_3 是3个帮助节点(省去了 in 节点),v_0 是新节点(省去了 out 节点). 图5

(*b*)、图5(*c*)是2个不同的修复树,由推论1可知v_1、v_2、v_3输出的数据量为α,中间节点输出的数据量也为α,两图中3个帮助节点的修复数据均是通过各自的最短路径到达新节点,但是,图5(*b*)中的修复通信量为8α,而图5(*c*)中的修复通信量为5α. 可以看出,不同的修复树会影响路由编码的效果. 对于不同的数据中心网络设计方式,建立修复树所受的限制也会不同,下面分别对Fat-tree与CamCube两种网络中基于MSR编码实现的路由编码进行分析.



(*a*) 拓扑结构 (*b*) 修复树一 (*c*) 修复树二

图5 修复树对修复通信量的影响

4.1 Fat-tree

如果交换机不能执行编码操作,编码操作就只能在服务器节点进行,会发生“数据流往返”的情形,如图6(*a*):服务器A先将数据传送给服务器B,服务器B把接收到的数据与本地数据编码后再发送给服务器D,服务器B与交换机C之间出现了“数据流往返”,产生了额外的通信量. 因此引入可以执行编码操作的路由器,例如Cisco 1900系列路由器具有数据包处理和计算功能. 如图6(*b*):服务器A、B同时将数据传送给路由器C,路由器C执行编码操作后再发送给服务器D,消除了“数据流往返”,节省了带宽. 下文中统一用路由器来表述,除了特别说明不能执行编码操作外,均默认可以执行编码操作.

4.1.1 简单再生协议(simple regeneration protocol, SRP)

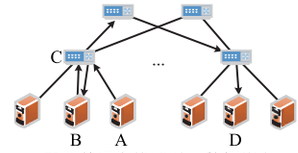
Fat-tree中,路由算法内置于内部路由器中,系统的物理拓扑和路由算法对服务是透明的,无法自定义路由路径. 因此,我们设计了SRP来构造修复树,SRP工作在应用层,图7是SRP报文的头部格式,最小为24字节,各个字段介绍如下:

类型:指明了应当如何处理该报文,有NOTIFICATION、ACK、DETECTION、START、DATA、DONE六种类型;

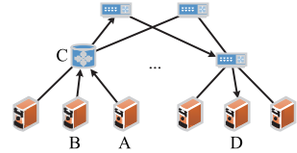
阈值:指明了路由器执行编码的条件,即$d-k+1$;

FID:文件的唯一标识;

源IP:为路由器提供修复过程中的子节点的IP地址;



(*a*) 配置普通交换机时的数据流向



(*b*) 配置具有计算功能的路由器时的数据流向

普通交换机 具有计算功能的路由器 服务器

图6 交换机的计算能力对修复通信量的影响

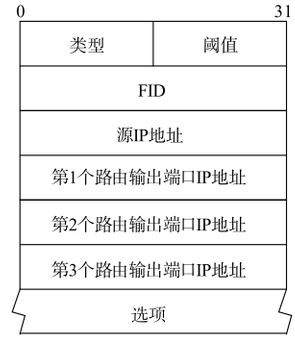


图7 SRP报文的头部格式

第1~3个路由输出端口IP地址:为新节点提供物理拓扑结构信息;

选项:当集群化路由器有多层或者修复过程涉及多个集装器时,修复数据经过的路由器会超过3个,需要增加路由输出端口IP地址.

4.1.2 修复过程

修复过程包括2个阶段:准备阶段和数据传输阶段. 新节点需要根据获得的拓扑结构选取帮助节点;每个路由器维护一个编码表,编码表包含修复过程中经

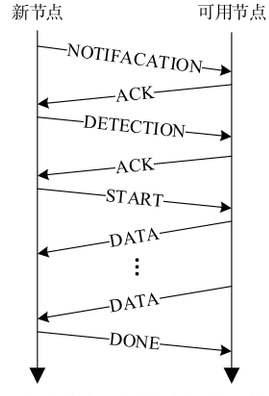


图8 新节点与可用节点间的通信过程

过该路由器的修复数据流的数量及来源,路由器需要根据 SRP 头部提供的信息来填充该编码表. 图 8 是新节点与可用节点间的通信过程.

准备阶段:

(1)当检测到一个存储节点失效后,一个新节点被选中并接受到一个数据报,包含失效数据对应的 FID、所有可用的存储节点的 IP 地址以及使用的编码参数 n、k.

(2)新节点向所有可用节点发送 NOTIFACATION,包含 FID.

(3)所有可用节点在收到 NOTIFICATION 后,返回一个 ACK,阈值字段为 0,包含 FID.

(4)当路由器接收到一个阈值字段为 0 的 ACK 后,把输出该 ACK 的端口 IP 地址依序添加到 SRP 头部中路由输出端口 IP 地址字段,并传送该 ACK. 假设此阶段后路由器不再更新路由表,保证传输路径固定.

(5)当新节点接收到所有可用节点的 ACK 后,利用 SRP 头部信息建立拓扑结构,并选出使得修复通信量最小的 d 个帮助节点.

(6)新节点向 d 个帮助节点发送 DETECTION,包含 FID 和赋值为 $d-k+1$ 的阈值.

(7)帮助节点接收到 DETECTION 后,返回一个 ACK,包含 FID、赋值为 $d-k+1$ 的阈值以及赋值为自身 IP 地址的源 IP 地址.

(8)当路由器接收到一个阈值字段不为 0 的 ACK 后,如果编码表中没有相关 FID 和源 IP 地址的表项,则在编码表中添加一个表项:FID、源 IP 地址、阈值、EffNum(初始值为 1);否则仅把对应表项中的 EffNum 加 1,但不超过阈值. 然后把源 IP 地址字段修改为输出该 ACK 的端口 IP 地址并把该 ACK 传送出去.

(9)新节点接收到所有帮助节点的 ACK 后,给每个帮助节点发送一个 START,包含 FID.

数据传输阶段:

(*a*)当一个帮助节点接收到 START 后,开始传输数据,SRP 类型为 DATA.

(*b*)只有当一个编码表中,同 FID、不同源 IP 地址的表项的 EffNum 的和大于阈值时,该路由器才可以执行编码操作. 如果执行编码操作,则需要等待所有源 IP 地址的数据均到达后才能执行编码并传送;如果仅执行转发操作,当一个数据报到达后可以直接传送而不需要等待.

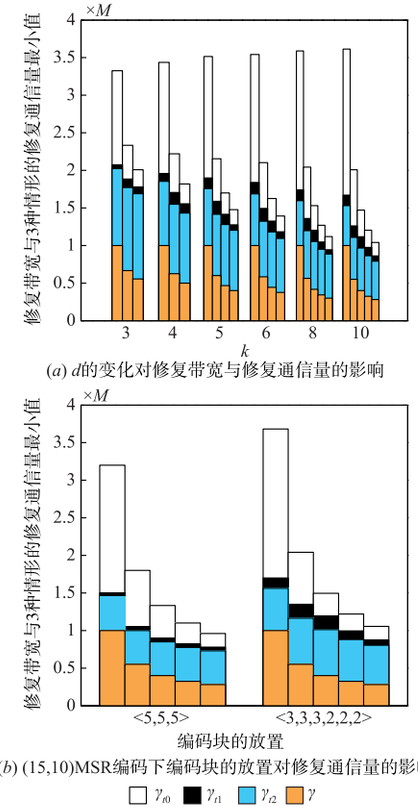
(*c*)当新节点接收到其相连的路由器传送的所有数据并确认修复成功后,向所有帮助节点和相关路由器(准备阶段的 4、5 步可得到相应的 IP 地址)发送 DONE,包含 FID.

(*d*)路由器接收到 DONE 后,清空编码表中相应的表项,当编码表为空时,可以更新路由表.

4.1.3　仿真与评估

对一个集装器内单节点失效的修复过程进行仿真,对比修复带宽 γ 与 3 种情形下的修复通信量最小值:不采取路由编码(γ_{t0})、使用 SRP 并且仅边缘路由器可以执行编码操作(γ_{t1})、使用 SRP 并且边缘路由器与集群化路由器均可以执行编码操作(γ_{t2}).

假设有 6 个边缘路由器和处于同一层的 2 个集群化路由器(防止发生单点失效的最低层次的路由器冗余),每个边缘路由器连接 10 个服务器,每个集群化路由器连接所有的边缘路由器. 同一个边缘路由器连接的服务器间的距离是 2 跳,不同边缘路由器连接的服务器间的距离是 4 跳,并且有 2 条不同的路径. 对大小为 M 的原数据采用($n=k+5$,k) MSR 编码,为了避免单个边缘服务器的失效导致原数据不可用,每个边缘服务器所连接的存储有编码块的服务器不超过 5 个. 图 9 给出了仿真结果,所有的结果都是 1000 次运行的平均值,每次运行中编码块的放置及失效节点都是随机的.



(*a*) *d*的变化对修复带宽与修复通信量的影响

(*b*) (15,10)MSR编码下编码块的放置对修复通信量的影响

图9　Fat-tree的仿真结果.每一组紧挨的条形图是*d*取不同值的结果，从左往右依次加1，初值为*k*.

图 9(*a*)展示了不同 k 值下 d 的增加对修复带宽与修复通信量的影响. 可以看到,对于不同的 k 值,d 的增加均会带来修复带宽与修复通信量的降低,但是由于实际中链路数的影响,修复通信量的降低速度不如修

复带宽的降低速度. 对于 3 种情形下的修复通信量,始终有 $\gamma_{t0}>\gamma_{t1}>\gamma_{t2}$,并且 d 较小时的 γ_{t1}、γ_{t2}可能小于 d 较大时的 γ_{t0},可见编码操作有利于修复通信量的降低. γ_{t1}与 γ_{t2}非常接近,说明在上述 Fat-tree 的拓扑结构下边缘路由器的编码操作更易降低修复通信量,这也与实际相符:路由算法内置于内部路由器的设计使得服务无法控制路由路径,而集群化路由器在给边缘路由器提供多条路经的同时,也会减少多个边缘路由器的数据流经过同一集群化路由器的机会(负载均衡). 可以想象,集群化路由器越多,在它上面执行编码的机会越低. 因此,在实际应用中,可以仅把边缘交换机换成具有计算能力的路由器,既可以节省成本,使用 SRP 的路由器也可以正常更新路由表,而不用在修复过程中停止更新. 另外,随着 d 的增加,γ_{t0}与 γ_{t1}、γ_{t2}越来越接近,说明编码的效果越来越差,这是因为编码的条件 $|W|>d-k+1$ 越来越苛刻,需要更多的帮助节点的数据汇合才能执行编码.

图 9(b)对比了(15,10)MSR 编码下两种编码块的放置方式对修复通信量的影响. 〈5,5,5〉表示把 15 个编码块放置在 3 个边缘路由器下的服务器中,每个边缘路由器分配 5 个编码块;〈3,3,3,2,2,2〉表示把 15 个编码块放置在 6 个边缘路由器下的服务器中,每个边缘路由器分配 3 或 2 个编码块. 可以看出,〈5,5,5〉的 3 种情形的修复通信量最小值均小于〈3,3,3,2,2,2〉的对应值,并且〈5,5,5〉的 γ_{t1}与 γ_{t2}更加接近. 〈5,5,5〉与〈3,3,3,2,2,2〉分别代表了集中放置与分散放置,图 9(b)说明不论执行编码操作与否,集中放置都有利于修复通信量的降低,而且采取集中放置时,相关的边缘路由器的数量减少,在集群化路由器层更难执行编码操作,γ_{t1}与 γ_{t2}也更加接近.

4.2 CamCube

通过使用服务器直接相连的拓扑结构,并让服务器处理数据包的传送,CamCube 完全消除了逻辑网络和物理网络的差异,将物理网络拓扑直接展现给服务,这就使得服务可以利用 CamCube 提供的 API 配置定制的路由协议,并且在每一跳都可以对数据包进行截获和修改[20]. 因此,可以自定义修复过程中的路由,尽可能地减少修复通信量.

4.2.1 启发式算法

CamCube 中修复通信量的最优化问题可以简化为一个 NP 完全问题:Steiner Tree 问题[10]. 因此,我们使用启发式算法来构造修复树. 首先,修改广度优先搜索(breadth-first search,BFS)算法[21]得到 newBFS 算法,以获取每个可用节点到新节点的所有可能的无权最短路径:修改了对后继顶点的处理过程(newBFS 中 14-21 行). 因为 BFS 仅能得到每个顶点的一条无权最短路径,我们需要所有可能的无权最短路径,从而根据每条边在路径中的重复次数来确定边的重要程度. newBFS 算法并没有增加计算复杂度,仍是 BFS 算法的 $O(V+E)$[21].

newBFS:新广度优先搜索算法

输入:源点 s,图 $G(V,E)$
输出:广度优先树
1. for each vertex $u\in V-\{s\}$
2. color[u] ← WHITE;//利用顶点的 3 种颜色(白色、灰色、黑色)来记录搜索轨迹
3. $d[u]\leftarrow\infty$;//变量 $d[u]$用于存储源点 s 与顶点 u 之间的距离
4. $\pi(u)\leftarrow\varnothing$;//队列 $\pi(u)$用于保存顶点 u 的父母,可能有多个,分别对应不同的路径
5. end
6. color[s] ← GRAY;
7. $d[s]\leftarrow 0$;
8. $\pi(s)\leftarrow\varnothing$;
9. $Q\leftarrow\varnothing$;//先进先出队列 Q 用于管理所有灰色顶点
10. ENQUEUE(Q,s);//入队列
11. while $Q\neq\varnothing$
12. $u\leftarrow$ DEQUEUE(Q);//出队列
13. for each $v\in$ Adj[u] //Adj[u]包含所有与顶点 u 相邻的顶点
14. if color[v] = WHITE
15. color[v] ← GRAY;
16. $d[v]\leftarrow d[u]+1$;
17. ENQUEUE(Q,v);
18. end
19. if $d[v]=d[u]+1$
20. ENQUEUE($\pi(v)$,u);//已经在同层遍历过,只添加父母,不再添加到队列 Q
21. end
22. end
23. color[u] ← BLACK;
24. end

算法 1 的核心思想是:首先获取启发式信息,根据每条边在所有无权最短路径中的重复次数来确定边的重要程度(即权值),以及有机会执行编码的节点(即所在边的重复次数达到阈值 $d-k+1$ 的节点);然后根据启发式信息在赋权值之后的图上构建修复树. 1~17 行利用蚁群算法给图中的边赋权值:一些“蚂蚁”沿着这些路径从可用节点出发前往新节点,当一条边被越多的“蚂蚁”经过时,该条边的权值越小,被选进修复树的机会越大. 当一条边上经过的“蚂蚁”数量达到阈值 $d-k+1$ 时,该边上的节点有机会执行编码操作. 18~27 行利用赋权后的图生成修复树:一共选取 d 个可用节点作为帮助节点,每次均选取距离修复树中新节点或有机会执行编码操作的节点最近(加权距离)的节点. 另外,我们给出一个简单的算法 2 作为比较. 算法 2 直接选取距离新节点最近(无权距离)的 d 个可用节点作为

帮助节点.

BFS 算法的计算复杂度是 $O(V+E)$,所以算法 2 的计算复杂度是 $O(V+E)$. 算法 1 中,newBFS 的运行时间是 $O(V+E)$,3 ~ 5 行的运行时间是 $O(l)$,6 ~ 17 行的运行时间是 $O(lE)$,20 ~ 26 行如果采用 Dijkstra 算法的话,运行时间是 $O[d(V\lg V+E)]$[21](CamCube 的拓扑结构是稀疏图,$|E| << |V|^2$),其中 $l=|R|$,l、d 均是与图规模无关的常数,所以算法 1 的计算复杂度是 $O(V\lg V+E)$.

算法 1　(n,k,d) MSR 编码的修复树启发式生成算法

输入:n,k,d,新节点 t,可用节点集合 A,图 $G(V,E)$;
输出:修复树 T
1. $R \leftarrow$ newBFS(G,t);//每个可用节点到新节点的所有最短路径集合
2. destiNodes $\leftarrow t$;//保存新节点与有机会执行编码的节点
3. for each path $p \in R$
4. Edge(p) $\leftarrow p$ 上的边;
5. end
6. for each edge $(i,j) \in E$
7. weight(i,j) $\leftarrow 0$;
8. for each path $p \in R$
9. if $(i,j) \in$ Edge(p)
10. weight(i,j) + +;
11. end
12. end
13. if weight(i,j) $\geqslant d-k+1$
14. destiNodes $\leftarrow$ destiNodes + i + j;//达到阈值的边上的节点有机会执行编码
15. end
16. weight(i,j) $\leftarrow$ Max-weight(i,j);//权重越小,选中的机会越大,定值 Max 保证 weight(i,j) 为正
17. end
18. $T \leftarrow t$;//修复树
19. TNodes $\leftarrow t$;//修复树中的节点
20. for $i=1:d$ //共选 d 个可用节点;
21. provider $\leftarrow A$ 中距离 TNodes ∩ destiNodes 最近的节点;//Dijkstra 算法
22. path $\leftarrow$ provider 到 TNodes ∩ destiNodes 的最短路径;
23. TNodes $\leftarrow$ TNodes + path 上的节点;
24. $T \leftarrow T$ + path;
25. $A \leftarrow A$-provider;
26. end
27. return T;

算法 2　(n,k,d) MSR 编码的修复树简单生成算法

输入:n,k,d,新节点 t,可用节点集合 A,图 $G(V,E)$;
输出:修复树 T
1. $R \leftarrow$BFS(G,t);//每个可用节点到新节点的最短路径集合
2. $T \leftarrow t$;
3. for $i=1:d$ //共选 d 个可用节点
4. provider $\leftarrow A$ 中距离 t 最近的节点;
5. $T \leftarrow T$ + provider 到 t 的最短路径;
6. $A \leftarrow A$-provider;
7. end
8. return T;

4.2.2　仿真与评估

对图 1(b)所示的 3 × 3 × 3CamCube 中单节点失效的修复过程进行仿真,任意两节点间距离有 3 种情况:同一条边上的节点间距离为 1 跳,同一个平面不同边上的节点间距离为 2 跳,不同平面上的节点间距离为 3 跳. 大小为 M 的原数据采用($n=k+5,k$) MSR 编码. 图 10 给出了仿真结果,所有的结果都是 1000 次运行的平均值,每次运行中编码块的放置及失效节点都是随机的.

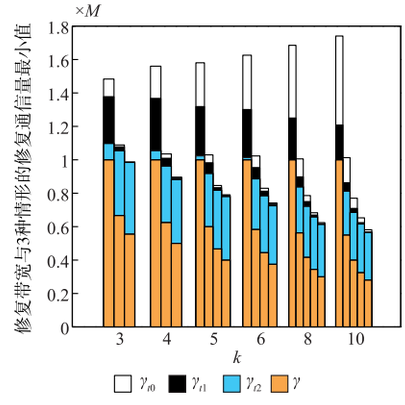


图10　CamCube的仿真结果.每一组紧挨的条形图是d取不同值的结果，从左往右依次加1，初值为k.

图 10 对比了修复带宽 γ 与 3 种情形下的修复通信量最小值:不采取路由编码(γ_{t0})、利用算法 2 生成修复树并采取路由编码(γ_{t1})、利用算法 1 生成修复树并采取路由编码(γ_{t2}). 注意 γ_{t0} 与 γ_{t1} 对应的修复树均由算法 2 生成(因为 γ_{t0} 在各个帮助节点生成的修复数据经过各自的最短路径到达新节点时达到最小),与 γ_{t2} 对应的由算法 1 生成的修复树一般不相同. 可以看出,对于不同的 k 值,修复带宽与修复通信量均随着 d 的增加而减小,并且由于实际中链路数的影响,修复通信量的降低速度不如修复带宽的降低速度(与 Fat-tree 中情形类似). 算法 1 对修复通信量的优化程度优于算法 2;算法 1 和算法 2 均在 d 较小时的优化程度更好,尤其是 $d=k$ 时,γ_{t2} 非常接近 γ,因为此时任意汇合 2 个帮助节点的数据就可以执行编码操作;但是随着 d 的增加,算法 1 和算法 2 的优化程度急剧下降,当 $d=n-1$ 时,γ_{t0} 与 γ_{t1}、γ_{t2} 几乎相等,算法 1 和算法 2 几乎都没有优化效果.

造成这种现象的原因与 CamCube 特殊的网络结构有关. 当 2 个帮助节点处于不同平面时,两者之间的距离为 3 跳,修复数据至少要经过 2 跳才能汇合,而修复数据直接传送到新节点最多经过 3 跳,也可能只需要 1 跳(该帮助节点与新节点在同一条边上). 也就是说,处于不同平面的帮助节点的修复数据在汇合之前经过的链路数一般会多于汇合之后到达新节点所经过的链路数. 而 3 个处于不同平面的帮助节点的修复数据的汇合更加困难. 所以 CamCube 中修复通信量降低的难度较大,而 d 的增大会加剧降低的难度.

相应的,如果编码块的放置密度较大时,会有更多的编码块处于同一条边或同一个平面,距离较近,汇合也较为容易. 如图 10 所示,随着 k 的增加(系统规模不变,编码块越密集),γ_{t1}、γ_{t2} 相比 γ_{t0} 的降低比例也在增加.

5 总结

本文围绕降低修复通信量的问题,通过分析证明得出了满足路由编码可行的充要条件,即,中间节点输出数据量的下界以及存储量与修复带宽的最优折衷. 然后针对两种设计方式不同的数据中心网络——Fat-tree 与 CamCube——设计了不同的修复树生成方案. 对于 Fat-tree,由于路由算法内置于内部交换机中,所以我们设计了 SRP 来构建较优的修复树;CamCube 为服务提供了 API 来自定义路由协议,但是 CamCube 中修复通信量的最优化是 NP 完全问题,因此我们设计了启发式算法来构建次优的修复树. 仿真结果显示,路由编码可以有效地降低修复通信量,2 种修复树生成方案在各自适合的网络中均有较好性能.

本文仅考虑了修复通信量这一衡量标准,还有许多其他的重要衡量标准,如修复时间. 路由编码及本文设计的 2 种修复树生成方案在实际系统中这些重要的衡量标准上的具体表现是下一步需要做的工作. 此外,虽然数据中心中单节点失效的概率远远高于多个节点失效的概率,但是由于修复过程中会导致级联失效[22],所以同时修复多个节点的能力也是必需的,因此多节点同时修复情形[17,23]下的路由编码也是下一步的研究内容. 另外,从定理 3 可以看出,帮助节点可以生成不等量的修复数据,此时对修复通信量和修复时间的影响目前仍然是一个开放性问题.

参考文献

[1] 罗象宏,舒继武. 存储系统中的纠删码研究综述[J]. 计算机研究与发展,2012,49(1):1 - 11.
Luo X H, Shu J W. Summary of research for erasure code in storage system[J]. Journal of Compute Research and Development, 2012, 49(1): 1 - 11. (in Chinese)

[2] 王意洁,许方亮,裴晓强. 分布式存储中的纠删码容错技术研究[J/OL]. 计算机学报,2016, http://www.cnki.net/kcms/detail/11.1826.TP.20160919.0024.002.html.
Wang Y J, Xu F L, Pei X Q. Research on Erasure Code-Based Fault-Tolerant Technology for Distributed Storage [J/OL]. CHINESE J OF COMPUTERS, 2016 http://www.cnki.net/kcms/detail/11.1826.TP.20160919.0024.002.html. (in Chinese)

[3] Huang C, Simitci H, Xu Y, et al. Erasure coding in windows azure storage[A]. USENIX Conference on Technical Conference [C]. Berkeley, CA, USA: USENIX, 2012. 2 - 2.

[4] Rodrigues R, Liskov B. High availability in DHTs: erasure coding vs. replication[A]. Proc. of Peer-to-Peer Systems [C]. Ithaca, NY, USA: Springer Berlin Heidelberg, 2005. 226 - 239.

[5] Rashmi K V, Nakkiran P, Wang J Y, et al. Having your cake and eating it too: jointly optimal erasure codes for I/O, storage, and network-bandwidth[A]. USENIX Conference on File and Storage Technologies[C]. Santa Clare, CA, USA: USENIX, 2015. 81 - 94.

[6] Sathiamoorthy M, Asteris M, Papailiopoulos D, et al. XORing elephants: novel erasure codes for big data[A]. Proc. of Very Large Database Endowment[C]. Riva del Garda, Italy: IEEE, 2013. 325 - 336.

[7] Dimakis A G, Godfrey P B, Wu Y N, et al. Network coding for distributed storage systems[J]. IEEE Trans. on Information Theory, 2010, 56(9): 4539 - 4551.

[8] 郝杰,逯彦博,刘鑫吉,等. 分布式存储中的再生码综述[J]. 重庆邮电大学学报(自然科学版),2013,25(1):30 - 38.
Hao J, Lu Y B, Liu X J, et al. Survey for regenerating codes for distributed storage[J]. Journal of Chongqing University of Posts and Telecommunications (Natural Science Edition), 2013, 25(1): 30 - 38. (in Chinese)

[9] Zeng T G, Liu L, Zhao J, et al. Router supported data regenerating protocols in distributed storage systems[A]. International Conference on Ubiquitous & Future Networks [C]. Dalian, China: IEEE, 2011. 315 - 320.

[10] Zhang J, Liao X K, Li S S, et al. Aggrecode: constructing route intersection for data reconstruction in erasure coded storage[A]. IEEE Conference on Computer Communications [C]. Toronto, ON, Canada: IEEE, 2014. 2139 - 2147.

[11] Rashmi K V, Shan N B, Gu D, et al. A solution to the network challenges of data recovery in erasure-coded distributed storage systems: a study on the Facebook warehouse cluster[A]. USENIX Workshop on Hot Topic in Storage

and File Systems[C]. San Jose, USA: USENIX, 2013. 8 – 8.

[12] Hwang K, Fox G C, Dongarra J J. 云计算与分布式系统:从并行处理到物联网[M]. 武永卫,等译. 北京:机械工业出版社,2013. 142 – 147.

[13] Vahdat A, Al-fares M, Loukissas A. A scalable, commodity datacenter network architecture[J]. ACM Sigcomm Computer Communication Review, 2008, 38(4): 63 – 74.

[14] Guo C, Lu G, Li D, et al. BCube: a high-performance server-centric network architecture for modular datacenters[J]. ACM Sigcomm Computer Communication Review, 2009, 39(4): 63 – 74.

[15] Wu H, Lu G, Li D, et al. MDCube: a high performance network structure for modular datacenter interconnection[A]. ACM Conference on Emerging Networking Experiments & Technology[C]. New York, NY, USA: ACM, 2009. 25 – 36.

[16] Costa P, Donnelly A, Oshea G, et al. CamCubeOS: A key-based network stack for 3D torus cluster topologies[A]. International Symposium on High-performance Parallel& Distributed Computing[C]. New York, NY, USA: ACM, 2013. 73 – 84.

[17] Kermarrec A M, Scouarnec N L, Straub G. Repairing multiple failures with coordinated and adaptive regenerating codes[A]. International Symposium on Network Coding[C]. Beijing, China: IEEE, 2011. 1 – 6.

[18] Li J, Yang S, Wang X, et al. Tree-structured data regeneration in distributed storage systems with regenerating codes[A]. IEEE Conference on Information Communications[C]. Piscataway, NJ, USA: IEEE, 2010. 2892 – 2900.

[19] Wang Y, Wei D S, Yin X R, et al. Heterogeneity-aware data regeneration in distributed storage systems[A]. IEEE Conference on Computer Communications[C]. Toronto, ON, Canada: IEEE, 2014. 1878 – 1886.

[20] Abu-libdeh H, Costa P, Rowstron A, et al. Symbiotic routing in future datacenters[J]. ACM Sigcomm Computer Communication Review, 2010, 40(4): 51 – 62.

[21] Cormen T H, Leiserson C E, Rivest R L, et al. 算法导论[M]. 潘金贵,等译. 北京:机械工业出版社,2006. 321 – 395.

[22] Datta A, Pamies-Juarez L, Oggier F. A study of the performance of novel storage-centric repairable codes[J]. Computing, 2016, 98(3): 1 – 23.

[23] 谢显中,黄倩,王柳苏,等. 一种云存储中基于干扰对齐的多节点精确修复方法[J]. 电子学报,2014,42(10):1873 – 1881.
Xie X Z, Huang Q, Wang L S, et al. A multi-node exact repair method in cloud storage based on interference alignment[J]. Acta Electronica Sinica, 2014, 42(10): 1873 – 1881. (in Chinese)

作者简介

丁炳辰　男,1992 年出生,湖北随州人,现为空军工程大学防空反导学院硕士研究生,主要研究方向为分布式存储与纠删码.
E-mail: xiaoding16@aliyun.com

李卫忠(通信作者)　男,1968 年出生,河南焦作人. 现为空军工程大学防空反导学院副教授. 主要研究方向为云计算.