胖树中的分布式动态容错路由

胡农达[1),2),3)] 王达伟[1),2)] 孙凝晖[1),2)]

[1)](中国科学院计算技术研究所 北京 100190)

[2)](中国科学院高性能计算机研究中心 北京 100190)

[3)](中国科学院研究生院 北京 100039)

摘 要 面向云计算的超大规模互连网络增加了对网络容错的要求,容错已成为互连网络的重要问题.为了保证网络的高可用性和高性能,文中基于胖树网络拓扑提出了一种分布式的动态容错路由方法.该方法通过引入一套链路失效消息传播机制和一套基于链路失效信息的动态容错路由算法来实现胖树网络的分布式动态容错.相比已有方法,该方法不增加网络硬件和路由路径长度,并且具有高执行效率和高性能.实验结果表明,在 m 端口交换机构成的胖树中,该方法可以容忍任意 $m/2-1$ 条失效链路并以高概率容忍更多条失效链路的组合,同时保持网络的高性能.

关键词 容错;胖树;分布式路由;动态失效;链路失效消息

中图法分类号 TP393 **DOI 号**: 10.3724/SP.J.1016.2010.01799

Distributed Dynamic Fault-Tolerant Routing in Fat Tree

HU Nong-Da[1),2),3)] WANG Da-Wei[1),2)] SUN Ning-Hui[1),2)]

[1)](*Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100190)

[2)](*High Performance Computer Research Center, Chinese Academy of Sciences, Beijing* 100190)

[3)](*Graduate University of Chinese Academy of Sciences, Beijing* 100039)

Abstract Fault tolerance of the interconnection network becomes increasingly important, since Cloud Computing is now pushing the data center to adopt the very large scale interconnection network to connect up to tens of thousands of server nodes. In order to maintain high availability and high performance of the interconnection network, this paper proposes a fat-tree based distributed and dynamic fault-tolerant routing methodology. The methodology adopts a link fault message spreading mechanism and a dynamic fault-tolerant routing algorithm to achieve fault tolerance of the fat-tree network. Compared with previous proposals, it neither requires additional network hardware nor increases the length of routing paths. The results show that, in a m-port n-tree topology, the methodology is able to completely tolerate all the combinations of $m/2-1$ fault links. Moreover, it can tolerate the combination of more fault links with a high probability (99.3 percent of probability to tolerate the combination of ten fault links in an 8-port 3-tree fat tree). Meanwhile, it maintains the good performance of fault-tolerant network.

Keywords fault tolerance; fat tree; distributed routing; dynamic fault; link fault message

---

收稿日期:2010-08-22.本课题得到国家“八六三”高技术研究发展计划项目“曙光 6000 千万亿次高效能计算机系统研制”(2009AA01A129)资助.**胡农达**,男,1983 年生,博士研究生,主要研究方向为计算机体系结构、高性能互连网络等. E-mail: hunongda@ncic.ac.cn. **王达伟**,男,1980 年生,博士,助理研究员,主要研究方向为分布式计算、计算机体系结构、高性能互联网络等. **孙凝晖**,男,1968 年生,博士,研究员,博士生导师,主要研究领域为并行体系结构、分布式操作系统、高性能计算等.

1 引 言

采用低成本服务器的云计算模式使数据中心规模不断扩大.目前部分数据中心的规模达到上万个服务器节点[1-2],且规模还在迅速扩大.在超大规模网络中,虽然单个网络部件的失效概率很低,但整个网络的失效概率已不能忽略.网络失效可能导致网络应用运行失败甚至整个系统瘫痪.互连网络的容错性已经成为制约计算机系统性能的一个瓶颈.因此,提供能够有效容忍网络部件失效的方法,是超大规模互联网络设计所面临的重大挑战之一.

为了处理网络失效,有两类容错手段被使用:静态容错和动态容错.静态容错依赖于检查点技术离线地处理网络部件的失效.当网络失效出现时,网络应用被暂停,网络根据失效情况被重配,然后从检查点开始重启应用.静态容错需要暂停应用,浪费了应用执行时间,同时记录检查点也消耗很多资源,降低了系统的性能.与静态容错不同,动态容错要高效得多.动态容错可以在不暂停网络应用的情况下,实时动态地处理网络失效,保证整个系统的正常高效运行.

目前存在多种方法应对网络失效.第一种最简单的方法是替换失效的网络元件,但考虑到庞大的网络规模和高频率的失效,该方法将付出高昂的成本和性能代价.第二种方法是增加一层网络管理层对失效后的网络拓扑进行探测,并重新计算路由表.这种方法也被称为网络重配,它的优点是非常灵活,缺点是会降低网络的性能,因为它使用一种通用的路由算法来代替针对网络拓扑优化过的算法[3].同时,如果网络的重配通过静态的方式进行,网络应用的暂停将不可避免,这大幅度地降低了系统的性能.第三种方法是定义一套容错的路由算法动态地处理网络失效.这些算法在网络失效发生时,能够提供另一条健康的路径对数据进行路由以避开失效的网络部件.由于该方法动态地处理网络失效并且不需要暂停系统,可以获得很高的性能.本文将采用该方法.

胖树是一种典型的多级互连网络.由于具有等分带宽、低网络直径、良好的扩展性和丰富的路径多样性的特点,胖树被广泛应用于超级计算机和数据中心的商用网络中.本文基于胖树拓扑提出了一种分布式的动态容错路由方法.该方法充分利用胖树网络的拓扑特性,可以在不增加链路、交换机等网络硬件和数据包路由路径长度的情况下实现胖树网络的动态容错.并且,该容错路由方法是无死锁的.

本文第 2 节介绍网络容错的相关工作;第 3 节给出胖树的定义;第 4 节给出胖树网络的一般路由算法;第 5 节阐述本文提出的分布式动态容错路由方法;第 6 节评价本文提出的容错路由方法的性能;第 7 节总结全文.

2 相关工作

在多级互联网络中已有大量工作对网络容错进行研究.由于有些网络不能在源和目的节点间提供多条路径,如 Butterfly 网络,很多工作通过增加链路或交换机来为网络提供容错能力[4-5].这种方法实现简单,但增加了网络硬件开销.另一种常被使用的方法是通过误路由(misrouting)数据来实现容错[6-7].由于数据包需要被路由通过更长的路径,数据传输延迟会增加.文献[8]将以上两种容错方法相结合,提出了一种混合容错方法.该方法可以获得更好的容错性能,其代价是增加了网络硬件和数据传输延迟.文献[9]比较了多种容错的多级互连网络并给出了评价.文献[10]通过使用并行多个多级互连网络来实现容错.

部分工作对胖树网络的容错进行了研究.文献[11]提出了一种由两棵并行胖树构成并且对应交换机由链路相连的拓扑以实现容错.该方法具有良好的静态容错性能,但在动态容错的情况下只能容忍一条失效链路.文献[12]通过局部误路由的方式实现胖树的动态容错.该方法需要在数据包头增加一个特殊的域并在容错路由过程中对其修改.同时,与其它误路由的容错方法相同,由于增加了数据的路由路径长度,该方法会增加数据传输延迟.文献[13]采用包含排除区间的区间路由方法来实现胖树的动态容错.该方法的一个缺点是在合并排除区间时健康路径可能被牺牲.文献[14]采用集中式管理织网来维护网络配置、监测链路状态、传播网络失效和路由信息,以实现胖树中的动态容错.该方法具有很好的灵活性,但由于管理节点需要和网络中的所有交换机相连并通信,其系统的扩展性受到很大的限制.

3 胖树拓扑

本文的工作基于 m-port n-tree 胖树,该胖树的

定义如下.

定义 1. m-port n-tree 胖树由两种类型的顶点组成：$2\times(m/2)^n$ 个处理节点和 $(2n-1)\times(m/2)^{n-1}$ 个 m 端口交换机. 处理节点标记为 $P(p=p_{n-1}p_{n-2}\cdots p_1p_0)$，其中 $p\in\{0,1,\cdots,m-1\}\times\{0,1,\cdots,(m/2)-1\}^{n-1}$，交换机标记为 $SW\langle l,c=c_{n-2}c_{n-3}\cdots c_1c_0\rangle$，其中 l 为交换机所在层号，$l\in\{0,1,\cdots,n-1\}$；c 的取值范围是

$$c\in\begin{cases}\{0,1,\cdots,(m/2)-1\}^{n-1}, & l=n-1\\ \{0,1,\cdots,m-1\}\times\{0,1,\cdots,(m/2)-1\}^{n-2}, & l\in\{0,1,\cdots,n-2\}\end{cases}$$

交换机 $SW\langle l,c\rangle$ 的第 k 个端口标记为 $SW\langle l,c\rangle_k$，其中 $k\in\{0,1,\cdots,m-1\}$.

交换机端口 $SW\langle l,c\rangle_k$ 和 $SW\langle l',c'\rangle_{k'}$ 相连，当且仅当

$$\begin{cases}l'=l+1\\ c_i'=c_i\ \text{ for all }\ i\neq l\\ k'=c_l\\ k=c_l'+m/2\end{cases}.$$

交换机端口 $SW\langle l,c\rangle_k$ 和节点 $P(p)$ 相连，当且仅当

$$\begin{cases}l=0\\ c_i=p_{i+1}\ \text{ for all }\ i\in\{0,1,\cdots,n-2\}\\ k=p_0\end{cases}.$$

图 1 是一个 4-port 3-tree 胖树网络.

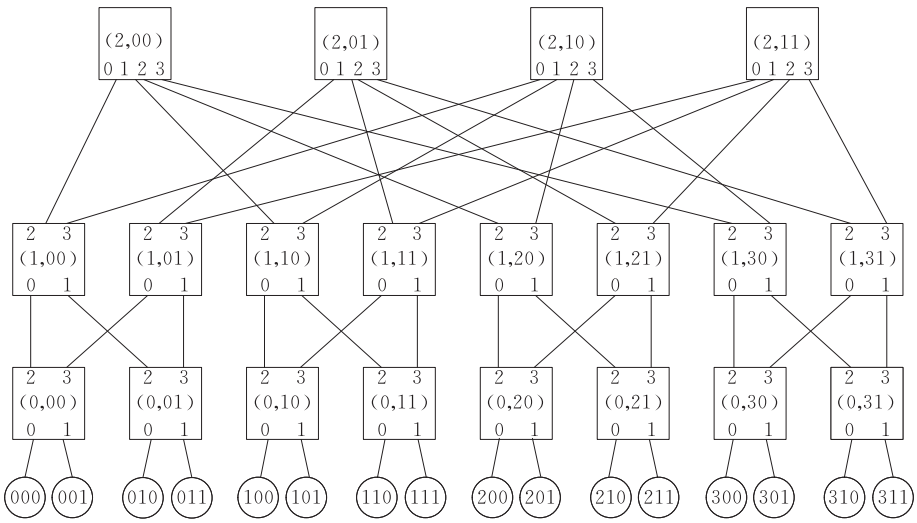

图 1　4-port 3-tree 胖树

4　胖树中的一般路由算法

在 m-port n-tree 胖树中，实现路由是十分简单的. 将一个数据包从源处理节点 $P(p=p_{n-1}p_{n-2}\cdots p_1p_0)$ 发送到目的处理节点 $P'(p'=p'_{n-1}p'_{n-2}\cdots p'_1p'_0)$ 的最短路由是：先将数据包向上转发到源和目的处理节点的最近公共祖先，然后向下一直转发到目的处理节点. 在上行阶段，交换机的任意一个上端口都可被选择，因此包含多条路由路径. 在下行阶段，路由路径是被唯一确定的：若数据包处于交换机 $SW\langle l,c_{n-2}c_{n-3}\cdots c_1c_0\rangle$，则该包必须通过交换机的端口 p_l 进行转发. 源和目的处理节点的最近公共祖先可以通过比较源和目的处理节点得到：对于源处理节点 $P(p=p_{n-1}p_{n-2}\cdots p_1p_0)$ 和目的处理节点 $P'(p'=p'_{n-1}p'_{n-2}\cdots p'_1p'_0)$，如果对于所有 $i\in\{i\mid j+1\leq i\leq n-1\}$，有 $p_i=p'_i$，但 $p_j\neq p'_j$，则向上所能到达的层号为 j 的交换机是源和目的处理节点的最近公共祖先. 沿不同上行路径，可以到达不同的最近公共祖先交换机. 一对源和目的处理节点共有 $(m/2)^j$ 个最近公共祖先交换机.

5　胖树中的分布式动态容错路由

在大规模网络中，链路会因为交换机或端口异常等原因而失效，这时数据包将不能通过该链路转发. 在胖树网络中，将一个数据包从源处理节点发送到目的处理节点，上行路径有多条可以选择，但下行路径是由上行路径和目标处理节点地址唯一确定的. 因此当网络中存在失效链路时，上行路径必须被合理地选择，以避免在下行过程中通过失效链路. 为了达到这样的目的，上行路径中的交换机必须预先得知链路失效信息以做出合理的路由选择. 考虑图 2 中的例子. 交换机端口 $SW\langle 1,21\rangle_1$ 和 $SW\langle 0,21\rangle_3$ 间

的链路失效了,一个数据包要从源处理节点 $P(p=100)$ 发送到目的处理节点 $P'(p'=211)$. 当该数据包到达交换机 $SW\langle 0,10\rangle$ 时,有两条上行路径可供选择,通过端口 2 或通过端口 3. 但是,如果选择了端口 3 上行,之后无论如何选择路由,在下行时都将通过失效链路,从而引起数据路由失败. 因此为了成功路由,在交换机 $SW\langle 0,10\rangle$ 中,足够的信息必须被提供,以避免通过端口 3 进行路由.

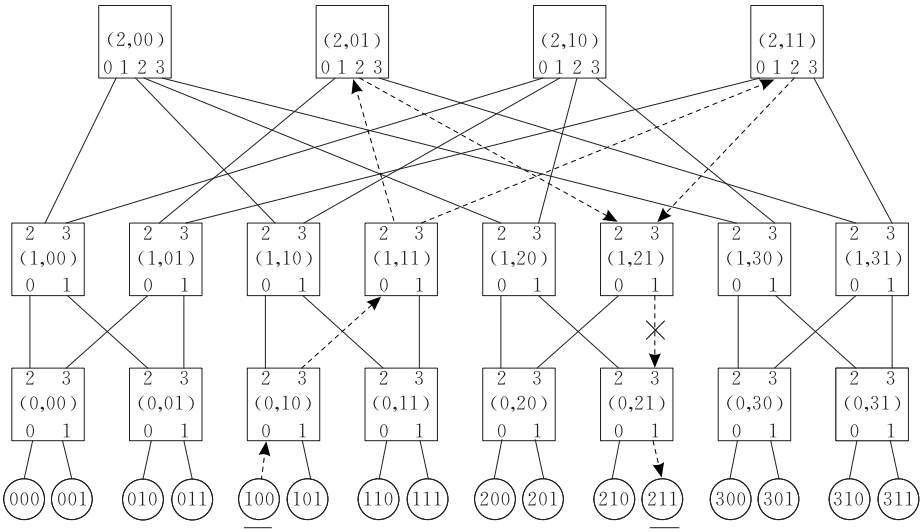

图 2 链路失效的胖树中路由失败的例子

本文对 m-port n-tree 胖树网络内的链路(不包括处理节点与 0 层交换机之间的链路)失效的容错进行研究,提出一套分布式的动态容错路由方法. 通过该方法,数据包在路由过程中可以避开包含失效链路的路径. 该方法包括链路失效的侦测、链路失效信息的传播和容错路由算法 3 部分. 下面分别对它们进行阐述.

5.1 链路失效的侦测

m-port n-tree 胖树中的交换机周期性地向与其相连交换机发送侦测包对其链路进行侦测. 其邻居交换机收到侦测包后,会回应一个应答包,表示链路正常. 如果从某端口发出的侦测包在一定时间内没有得到应答,交换机会向该端口重发侦测包,重发次数上限为固定值. 如果重发次数达到上限时,仍旧没有收到应答包,则该端口的链路被认为已经失效,相应的端口状态被设置. 在此,交换机的端口被分为上端口和下端口两类. 对于非根交换机,上端口是指端口 $m/2,m/2+1,\cdots,m-1$,下端口是指端口 $0,1,\cdots,m/2-1$;对于根交换机,所有端口都是下端口. 上端口状态用二元组 $PS\langle l,f\rangle$ 表示,其中 l 是与失效链路相关的层号,f 是失效标识. 上端口 $m/2,m/2+1,\cdots,m-1$ 的状态可分别记为 $PS_{m/2-1}\langle l,f\rangle,\cdots,PS_{m-1}\langle l,f\rangle$. 当交换机上端口 $SW\langle l',c'_{n-2}c'_{n-3}\cdots c'_1c'_0\rangle_{k'}$ 所连链路被侦测到失效时,该端口状态被设置为 $PS_{k'}\langle l',1\rangle$,其中 l' 是交换机层号. 在 5.2 节可以看到,当交换机上端口收到下行链路失效消息时,端口状态也会被设置. 下端口状态仅用失效标识 f 表示. 对于非根交换机,下端口 $0,1,\cdots,m/2-1$ 分别对应状态标识 $f_0,f_1,\cdots,f_{m/2-1}$;对于根交换机,下端口 $0,1,\cdots,m-1$ 分别对应状态标识 $f_0,f_1,\cdots,f_{m-1}$. 当交换机下端口 k 被侦测到失效时,标识 f_k 被置 1. 根据该侦测机制,交换机的端口失效或整个交换机的失效,最终也表现为链路失效.

5.2 链路失效信息的传播

链路的表示. m-port n-tree 胖树中链路由唯一标识 $LK\langle l,c_{n-2}c_{n-3}\cdots c_1c_0\rangle_k$ 表示,其中 $SW\langle l,c_{n-2}c_{n-3}\cdots c_1c_0\rangle_k$ 是与链路相连的交换机的上端口. 一般化地,交换机端口 $SW\langle l'=l+1,c'_{n-2}c'_{n-3}\cdots c'_1c'_0\rangle_{k'}$ 和 $SW\langle l,c_{n-2}c_{n-3}\cdots c_1c_0\rangle_k$ 之间的链路可以表示为 $LK\langle l'-1,c'_{n-2}c'_{n-3}\cdots c'_{l'k'}c'_{l'-2}\cdots c'_1c'_0\rangle_t$,$t=m/2+c'_{l'-1}$ 或 $LK\langle l,c_{n-2}c_{n-3}\cdots c_1c_0\rangle_k$. 因为根据胖树定义,总有 $\langle l'-1,c'_{n-2}c'_{n-3}\cdots c'_{l'k'}c'_{l'-2}\cdots c'_1c'_0\rangle_t=\langle l,c_{n-2}c_{n-3}\cdots c_1c_0\rangle_k$,所以链路标识是唯一的. 比如,在图 2 中,交换机端口 $SW\langle 1,21\rangle_1$ 和 $SW\langle 0,21\rangle_3$ 之间的链路可唯一表示为 $LK\langle 0,21\rangle_3$.

链路失效消息. 交换机周期性地查看端口状态,当发现某端口失效时,向其它交换机发送链路失效消息. 有两类链路失效消息被定义,它们分别是上行链路失效消息(Upward Link Fault Message,ULFM)和下行链路失效消息(Downward Link Fault

Message，DLFM).

上行链路失效消息用 $ULFM\langle l, c_{n-2}\,c_{n-3}\cdots c_1 c_0, k\rangle$ 表示，其中 $LK\langle l, c_{n-2}c_{n-3}\cdots c_1 c_0\rangle_k$ 是失效链路. 在 3 种情况下，交换机会发送上行链路失效消息.

第 1 种情况是交换机侦测到其某下端口失效，即其失效标识为 1. 交换机向失效端口以外的所有端口发送上行链路失效消息 $ULFM\langle l, c_{n-2}\,c_{n-3}\cdots c_1 c_0, k\rangle$. 收到 ULFM 的交换机按以下规则对 ULFM 进行转发或处理.

(1) 如果当前交换机的层号不等于 l 且 ULFM 来自上端口，则交换机向其所有下端口转发 ULFM.

(2) 如果当前交换机的层号不等于 l 且 ULFM 来自下端口，则交换机向除来源端口以外的所有端口转发 ULFM.

(3) 如果当前交换机的层号等于 l，则当前交换机对 ULFM 消息进行处理. ULFM 的信息被记录在交换机中的链路失效表(Link Fault Table，LFT)的链路失效条目(Link Fault Entry，LFE)中. 链路失效条目用 $LFE\langle c_{n-2}\,c_{n-3}\cdots c_1 c_0, f_{m/2-1}\,f_{m/2-2}\cdots f_1 f_0\rangle$ 表示，坐标信息 $c_{n-2}c_{n-3}\cdots c_1 c_0$ 相同的链路失效信息被记录在同一个链路失效条目中. 因此，同一交换机的所有上端口的链路的失效信息被保存在同一条目中. 当 l 层交换机处理上行链路失效消息 $ULFM\langle l, c_{n-2}\,c_{n-3}\cdots c_1 c_0, k\rangle$ 时，它首先检查是否存在坐标为 $c_{n-2}c_{n-3}\cdots c_1 c_0$ 的条目：如果存在，则将该条目的 $f_{k-m/2}$ 置为 1；否则，新增条目 $LFE\langle c_{n-2}\,c_{n-3}\cdots c_1 c_0, f_{m/2-1}\,f_{m/2-2}\cdots f_1 f_0\rangle$，其中 $f_{k-m/2}=1$，$f_i=0$，$i\neq k-m/2$.

由于 m-port n-tree 胖树网络的拓扑特点，$ULFM\langle l, c_{n-2}\,c_{n-3}\cdots c_1 c_0, k\rangle$ 只会被记录在交换机 $SW\langle l, xx\cdots x c_{l-1}\,c_{l-2}\cdots c_1 c_0\rangle$($x$ 为任意值)中. 同时，从上面的规则可知，ULFM 一旦向下转发就不再会被向上转发，因此 ULFM 的转发不会形成环，所以 ULFM 的传播是无死锁的. 图 3 给出了链路 $LK\langle 0, 21\rangle_3$ 失效时上行链路失效消息 $ULFM\langle 0, 21, 3\rangle$ 的传播过程，并最终在底层交换机中生成链路失效条目 $LFE\langle 21, 10\rangle$.

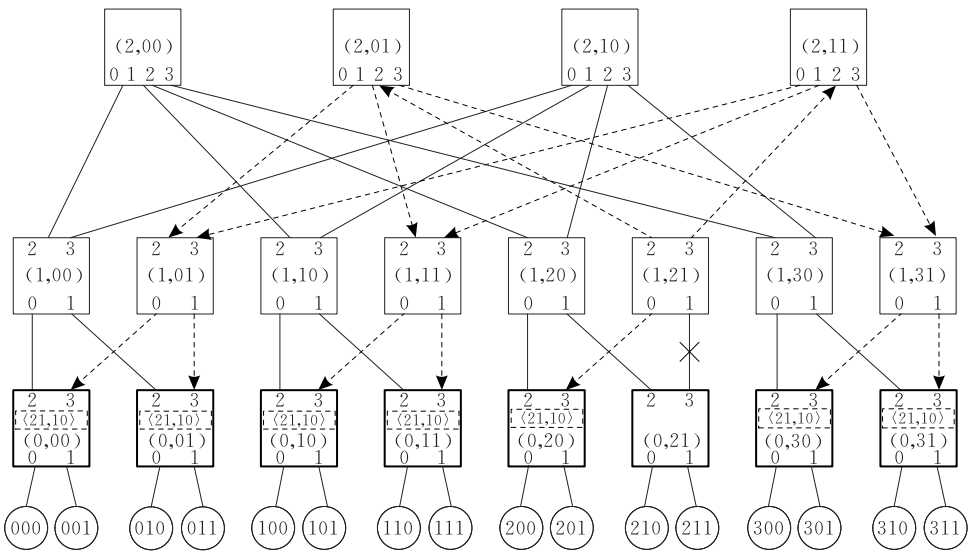

图 3　上行链路失效消息的传播

第 2 种情况是交换机的链路失效表的某一链路失效条目 $LFE\langle c_{n-2}\,c_{n-3}\cdots c_1\,c_0, f_{m/2-1}\,f_{m/2-2}\cdots f_1 f_0\rangle$ 的 $f_{m/2-1}f_{m/2-2}\cdots f_1 f_0$ 为全 1. 此时，交换机向其所有下端口发送上行链路失效消息 $ULFM\langle l-1, c_{n-2}c_{n-3}\cdots c_l k c_{l-2}\cdots c_1 c_0, m/2+c_{l-1}\rangle$，其中 l 是交换机所在层号，$k=0,1,\cdots,m/2-1$. 收到上行链路失效消息的交换机的行为与第 1 种情况相同.

第 3 种情况是交换机的上端口链路存在失效，且其链路失效表的某一链路失效条目 $LFE\langle c_{n-2}c_{n-3}\cdots c_1 c_0, f_{m/2-1}f_{m/2-2}\cdots f_1 f_0\rangle$ 中 $f_{m/2-1}f_{m/2-2}\cdots f_1 f_0$ 不为全 1，但分别与对应的上端口状态二元组中失效标识 $PS_{m-1}.fPS_{m-2}.f\cdots PS_{m/2}.f$ 按位或后为全 1(即 $f_{m/2-1}f_{m/2-2}\cdots f_1 f_0 \mid PS_{m-1}.fPS_{m-2}.f\cdots PS_{m/2}.f$ 为全 1)，并且当前交换机与条目所指示的交换机 $SW\langle l, c_{n-2}c_{n-3}\cdots c_1 c_0\rangle$ 的最近公共祖先所在层号大于所有失效上端口的状态二元组中的层号的最大值($\max\{PS_k.l \mid PS_k.f$ 为 $1, k=m/2, m/2+1, \cdots, m-1\}$). 此时，交换机向其所有下端口发送上行链路失效消息 $ULFM\langle l-1, c_{n-2}c_{n-3}\cdots c_l k c_{l-2}\cdots c_1 c_0, m/2+c_{l-1}\rangle$，其中 l 是交换机所在层号，$k=0,1,\cdots,$

$m/2-1$. 收到上行链路失效消息的交换机的行为与第 1 种情况相同.

下行链路失效消息用 $DLFM\langle l\rangle$ 表示,其中 l 是交换机的层号. 当一个交换机的所有上端口的状态二元组中的失效标志为 1 时,交换机向其所有下端口发送下行链路失效消息 $DLFM\langle l\rangle$,其中 l 是所有上端口状态二元组中层号的最大值($\max\{PS_k.l|PS_k.f$ 为 $1,k=m/2,m/2+1,\cdots,m-1\}$). 收到 DLFM 的交换机对 DLFM 进行处理:将收到该消息的上端口 k 的状态设为 $PS_k\langle l,1\rangle$,其中 l 是下行链路失效消息中的层号. $PS_k\langle l,1\rangle$ 向交换机提供这样的信息:通过上端口 k 向上路由时最高到达层 l,因为所能到达的 l 层的交换机的所有上端口都已经失效. 接收到下行链路失效消息的交换机,如果其所有上端口的状态二元组中的失效标志已变为全 1,则同样要向下端口发送下行链路失效消息. 由于下行链路失效消息只往下传输,因此下行链路失效消息的传播不会形成环,所以也是无死锁的.

对于链路失效消息的传播,消息的最长传播路径是由 1 层交换机发出的上行链路失效消息通过根交换机转发到底层交换机. 因此,在 m-port n-tree 胖树中链路失效消息的传播只需经过 $2n-3$ 跳的转发,它的复杂度是 $O(n)$,其中 n 是胖树的层数. 由于链路失效消息很短并且交换机向各个端口发送或转发消息可以并行,链路失效消息的转发延迟很小. 在交换机的设计中,通过为链路失效消息的转发设置独立的虚通道并且设置比普通数据包更高的转发优先级,即使在网络饱和的情况下链路失效信息也能快速地传播到整个网络.

5.3 容错路由算法

本小节描述容错路由算法,该算法是根据链路失效表的信息和上端口状态信息 $PS\langle l,f\rangle$ 对胖树网络的一般路由算法的增强. 交换机利用链路失效条目和上端口状态信息合理地选择路由端口,以避开失效链路,从而实现动态容错路由. 下面对该算法进行描述. 假设一个数据包从源处理节点 $P(p=p_{n-1}p_{n-2}\cdots p_1p_0)$ 发送到目的处理节点 $P'(p'=p'_{n-1}p'_{n-2}\cdots p'_1p'_0)$. 某一时刻该数据包到达交换机 $SW\langle l,c_{n-2}c_{n-3}\cdots c_1c_0\rangle$,则按以下步骤进行路由:

1. 如果数据包来自上端口或 $l\geq l_{anc}$(l_{anc} 是源处理节点和目的处理节点的最近公共祖先交换机的层号,$l_{anc}=\max\{i|p_i\neq p'_i,i=0,1,\cdots,n-1\}$),则通过 $SW\langle l,c_{n-2}c_{n-3}\cdots c_1c_0\rangle$ 的下端口 p_l 向下路由数据包,并完成路由. 否则,需要选择合适的上端口对数据包进行向上路由,执行步 2.

2. 检查每个上端口 k 的状态 $PS_k\langle l,f\rangle$,如果 $PS_k.f$ 为 1 且 $PS_k.l<l_{anc}$,则将该端口从待选择的上端口的集合中排除. $V=V_{m/2-1}V_{m/2-2}\cdots V_0$ 被用于表示上端口是否在待选的上端口的集合中. $V_{m/2-1},V_{m/2-2},\cdots,V_0$ 分别对应上端口 $m-1,m-2,\cdots,m/2$,它们的初始值都为 1. 当上端口 k 被排除时,$V_{k-m/2}$ 被置为 0. 对所有上端口的检查被同时执行.

3. 检查链路失效表中的链路失效条目 $LFE\langle c'_{n-2}c'_{n-3}\cdots c'_1c'_0,f'_{m/2-1}f'_{m/2-2}\cdots f'_1f'_0\rangle$,并判断条目所指示的交换机 $SW\langle l,c'_{n-2}c'_{n-3}\cdots c'_1c'_0\rangle$(其中,$l$ 为当前交换机层号,$c'_{n-2}c'_{n-3}\cdots c'_1c'_0$ 为条目中信息)是否为 $P'(p'=p'_{n-1}p'_{n-2}\cdots p'_1p'_0)$ 的祖先. 如果 $SW\langle l,c'_{n-2}c'_{n-3}\cdots c'_1c'_0\rangle$ 是 $P'(p'=p'_{n-1}p'_{n-2}\cdots p'_1p'_0)$ 的祖先(即 $c'_{n-2}c'_{n-3}\cdots c'_{l+1}c'_{l'}=p'_{n-1}p'_{n-2}\cdots p'_{l'}+2p'_{l'+1}$),则检查 $f'_i(i=0,1,\cdots,m/2-1)$ 是否为 1. 若 f'_k 为 1,则将上端口 $k+m/2$ 从待选择的上端口的集合中排除,即 $V_{k+m/2}$ 被置为 0. 所有条目的检查可被同时执行.

4. 检查上端口 p'_l 是否在待选择的上端口的集合中,即判断 V_k(其中 $k=p'_l$)是否为 1. 如果是,则从上端口 p'_l 对数据包进行路由;否则,选择下一个在待选择的上端口的集合中的端口进行路由. 如果待选择的上端口的集合为空,则表明不存在健康的路径能够路由该数据包,该数据包被丢弃.

对于以上容错路由算法,在交换机的设计中,各步骤并非顺序执行的. 当交换机获得数据包的源和目的处理节点地址后,步 1~3 就可以由独立的硬件模块并行执行,并最终通过组合逻辑获得路由端口号. 理论上,整个路由过程可以在一个时钟周期内完成,但考虑到实际设计中的时钟周期约束,路由过程往往被设计成在常数个时钟周期内完成. 但是,无论如何,路由算法复杂度总可以保持为 $O(1)$,因此是十分高效的.

图 4 给出了容错路由的一个例子. 数据包从源处理节点 $P(p=100)$ 发送到目的处理节点 $P'(p'=211)$. 当数据包通过交换机 $SW\langle 0,10\rangle$ 向上路由时,交换机通过检查链路失效条目 $LFE\langle 21,10\rangle$ 将上端口 3 从待选择的上端口的集合中排除,从而避免了在路由过程中通过失效链路 $LK\langle 0,21\rangle_3$.

根据以上 3 小节对胖树中的分布式动态容错路由方法的阐述,可以发现该容错方法具有以下特点:

(1) 该容错方法不需要在胖树中增加额外的链路或交换机等网络硬件,而只需对原有胖树路由算法进行增强,因此硬件代价小.

(2) 该容错方法不增加数据在胖树网络中的路由路径长度,因此不增加相应的数据传输延迟.

(3) 该容错方法可以容忍任意 $m/2-1$ 条失效链路. 由容错路由算法可知,数据包仅在没有上端口可以路由时才被丢弃,而一条失效链路只能引起交

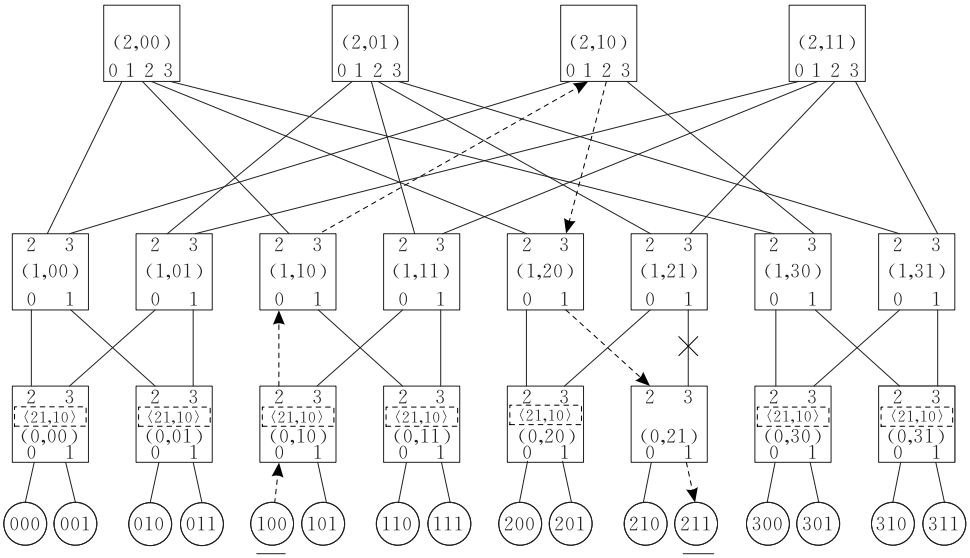

图 4　容错路由的一个例子

换机一个上端口不可路由(表现为路由算法根据链路失效表的信息或上端口状态信息将该上端口从待选择的上端口的集合中排除). 因此 $m/2-1$ 条失效链路最多引起交换机的 $m/2-1$ 个上端口不可路由，至少还存在一个上端口可以提供路由. 所以，该容错方法可以容忍任意 $m/2-1$ 条失效链路.

(4) 该容错方法是无死锁的. 由 5.2 节的讨论可知，ULFM 和 DLFM 的传播是无死锁的. 同时，由 5.3 节阐述的容错路由算法的步 1 可知，数据包在胖树中路由时，一旦进入下行路由阶段就不再向上路由，因此数据包在胖树中的路由不会形成环，所以该路由算法也是无死锁的. 综合以上两方面，可知该容错方法是无死锁的.

6　性能评价

本节对文中提出的分布式动态容错路由方法进行评估. 一个时钟精确的模拟环境被建立，对容错方法的性能进行测试. 在该环境中，m 端口的交换机采用虚切入的数据转发方式，XON/XOFF 流控方式，交换机中链路失效表的容量除特殊说明外都为 32 个条目. 数据流均匀注入，每个节点的注入率相同，目标地址随机均匀分布. 数据包的长度为 64 拍. 失效链路的注入采用随机均匀分布. 不考虑处理节点和底层交换机间的链路失效的情况.

本节首先对采用了本文提出的容错方法的 4-port 3-tree (4P3T)，4-port 5-tree (4P5T)，8-port 3-tree (8P3T)胖树在不同失效链路数目下的容错能力进行分析；然后分析 3 种胖树容错后的吞吐率和数据传输延迟，以评价容错对网络性能的影响；最后，分析容错能力与交换机中链路失效表容量的关系，对该容错方法的可扩展性进行讨论.

6.1　容错能力

本小节对本文提出的容错路由方法的容错能力进行分析. 首先对几个概念进行说明：可容忍 k 条失效链路的某种组合是指在该失效链路的组合下容错方法能够为每一对源和目的节点提供一条健康的路径；可容忍 k 条失效链路是指能够容忍任意一种 k 条失效链路的组合；k 条失效链路的不可容错概率是指不可容忍的 k 条失效链路的组合数与 k 条失效链路的总组合数的比值；k 条失效链路的容错概率是指可容忍的 k 条失效链路的组合数与 k 条失效链路的总组合数的比值. 由于在网络规模较大且失效链路数目较多时，要穷尽所有的失效链路的组合是非常困难甚至是不可能的，因此在给定失效链路数目的情况下，如果失效链路的组合少于 1000 种，则穷尽所有的失效链路组合，否则，测试 1000 种随机的失效链路组合.

图 5 给出了 4-port 3-tree，4-port 5-tree，8-port 3-tree 胖树在 0～20 条链失效路的情况下的不可容错概率. 与第 5 节的分析结果一致，该图显示，对于 m-port n-tree 胖树，本文提出的容错方法可以容忍任意 $m/2-1$ 条失效链路. 比较 3 种胖树：随着失效链路数的增加，4-port 3-tree的不可容错概率增长迅速，4-port 5-tree 的增长则要慢一些. 这是因为 4-port 5-tree 的链路数比 4-port 3-tree 多，在相同失效链路数的情况下，它的失效链路数比率更低. 8-port 3-tree 和 4-port 5-tree 虽然有相同链路数和

相同的失效链路数的比率，但 8-port 3-tree 的不可容错概率比 4-port 5-tree 低得多且增长十分缓慢. 这是因为较大端口的交换机构成的胖树与较小端口的交换机构成的胖树相比，在存在链路失效的情况下，前者所有上端口都不可路由的概率更小，这有利于降低一对源和目标节点间不存在健康通路的概率. 同时，可以注意到本文提出的容错方法对 8-port 3-tree 已具有很好的容错能力：(1) 与理论分析一致，少于等于 3 条失效链路时网络失效被完全容忍；(2) 失效链路数为 4～6 条时，在 1000 中随机的失效链路组合中也不存在不能容忍的组合；(3) 当失效链路数远大于 3 条的容错能力时，网络依旧保持极低的不可容错概率，比如失效链路为 10 条时，不可容错概率仅为 0.007，即可以容忍 99.3% 的失效链路的组合. 通过 3 种胖树容错性能的比较，可以发现，对于由更大端口交换机构成的胖树，本文提出的容错方法有更好的容错能力.

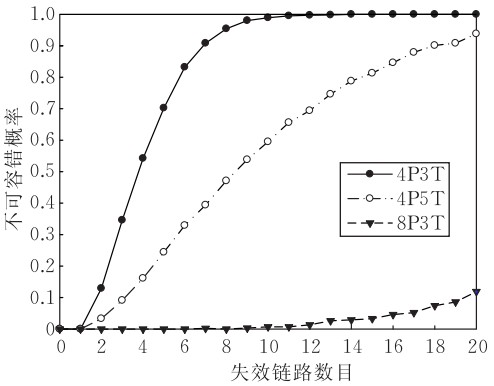

图 5 不同失效链路数目下的容错能力

6.2 容错网络的性能

对于容错网络，除了容错能力之外，容错后的网络性能也是评价容错方法的重要方面. 本小节分析容错后网络的吞吐率和延迟的变化. 不同规模的胖树在 0～10 条失效链路的情况下的容错后性能被测试. 由于分析不可容错的失效链路组合下的网络性能没有意义，此处只对可容错的失效链路组合下的网络性能进行分析.

图 6 和图 7 给出了 4-port 3-tree，4-port 5-tree，8-port 3-tree 胖树在 20% 的注入率的情况下吞吐率和延迟随失效链路数的变化. 从图中可以看出：容错后网络始终保持与没有失效链路的情况下相同的吞吐率；容错后网络的延迟仅略有上升. 比较 3 种胖树，具有更大端口的交换机构成的 8-port 3-tree 胖树受到的影响最小. 总体来看，在网络非饱和的情况下，容错方法几乎不降低网络的性能.

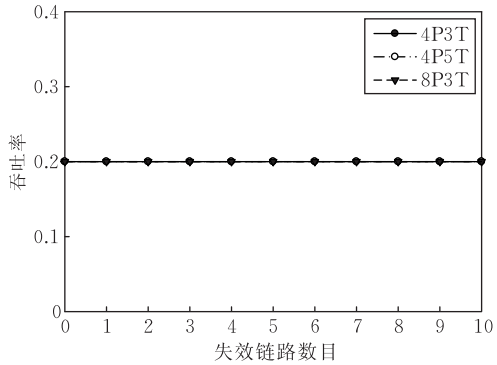

图 6 网络非饱和时，不同失效链路数目下的吞吐率

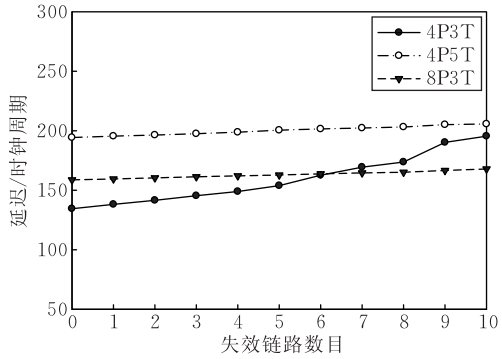

图 7 网络非饱和时，不同失效链路数目下的数据包传输延迟

图 8 和图 9 给出了 4-port 3-tree，4-port 5-tree，8-port 3-tree 胖树在 100% 的注入率的情况下，饱和吞吐率和延迟随失效链路数的变化. 在该情况下，网络已经达到饱和. 从吞吐率的角度看，随着失效链路数的增加，饱和吞吐率有所下降. 由于有更大比率的链路失效，4-port 3-tree 的吞吐率的下降更加明显. 8-port 3-tree 与 4-port 5-tree 虽然有相同的失效链路比率，但前者的性能损失更小. 从网络延迟的角度来看，可以得到相似的结论. 从总体来看，容错后网络依旧保持良好的性能. 以 8-port 3-tree 为例，当失效链路数为 10 条(远大于 3 条的容错能力)时，网络的归一化饱和吞吐率仍能达到 0.4.

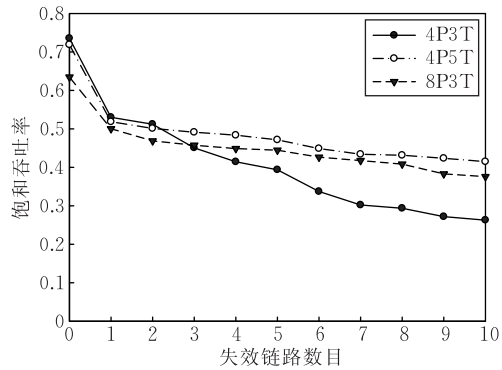

图 8 网络饱和时，不同失效链路数目下的饱和吞吐率

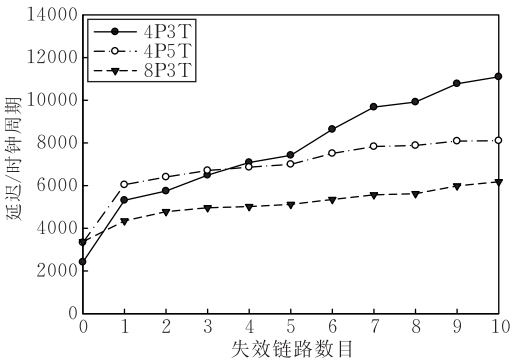

图 9　网络饱和时，不同失效链路数目下的数据包传输延迟

由以上讨论可知，在网络非饱和的情况下，容错对网络的性能几乎没有影响；在网络饱和的情况下，容错网络的性能与没有失效链路的情况下相比有不同程度的下降，但依旧能保持良好的性能. 并且，由更大端口交换机构成的胖树有更好的容错性能.

6.3　链路失效表容量对容错的影响

在构建容错胖树的交换机中，链路失效表(LFT)中链路失效条目(LFE)数会随着失效链路数的增加而增加. 当 LFT 没有空间容纳新的 LFE 时，新条目将不被记录，链路失效信息会丢失，从而引起网络容错能力的下降. 因此，随着网络规模增大和失效链路数增多，LFT 容量(LFT 中可容纳的的最大 LFE 数目)可能会成为制约容错能力的关键因素. 本节分析容错能力与 LFT 容量的关系.

由第 5 节可知，当链路 $LK\langle l, c_{n-2}c_{n-3}\cdots c_1c_0\rangle_k$ 失效时，在交换机 $SW\langle l, xx\cdots xc_{l-1}c_{l-2}\cdots c_1c_0\rangle$ 中会建立一个条目，并且同一交换机的所有上端口的链路的失效信息被记录在同一条目中. 因此最坏的情况是：失效链路发生在 0 层交换机的上端口，并且每个交换机只有一条链路出错. 这样当有 k 条链路出错时，0 层交换机的 LFT 中就要保存 k 条 LFE 记录. 所以，要保证最大的容错能力，LFT 容量最小为 k. 图 10 给出了 8-port 3-tree 在存在 12 条失效链路的情况下，不可容错概率与 LFT 容量的关系. 不可容错概率由随机 12 条失效链路的 1000 种组合进行实验得到，不同的 LFT 容量下的 1000 种随机组合是相同. 从图中可以看出，与前面分析的一致，当 LFT 的容量大于等于 12 个 LFE 条目数时，网络已经达到最大的容错能力. 并且，由于在 1000 次随机中最坏的情况并没出现，当 LFT 的容量为 10 个 LFE 条目时，网络已达到最大的容错能力.

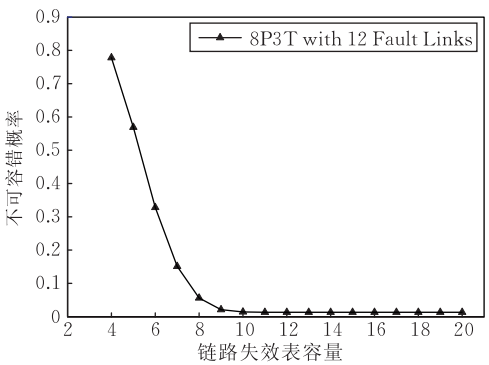

图 10　不可容错概率与 LFT 容量的关系

由上面的讨论可知，为了使网络要达到最大的容错能力，当胖树中存在 k 条链路失效时，在最坏的情况下 LFT 需要的容量为 k 个 LFE 条目. 但在现实中，由于大多数失效链路仅与少数交换机相关，且在我们的容错方法中同一交换机的所有上端口的链路的失效信息被保存在同一 LFE 中，LFT 需要的容量将远远小于失效链路数目，而仅与失效的交换机数目相关. 文献[15]在分析了 8 个由数十万台服务器构成的商品化数据中心的日志后指出 95% 的网络失效集中发生在少于 20 个网络设备中. 因此，LFT 容量不会成为制约网络容错能力的因素.

7　结束语

胖树由于具有等分带宽、低网络直径、良好的扩展性和丰富的路径多样性的特点被广泛应用于超级计算机和大型数据中心的商用网络. 胖树的路径多样性使得它能够为源和目的处理节点对提供多条不相交的路径，这使得胖树本身具有良好的静态容错能力. 但是，数据包在胖树中路由时，虽然在上行路由阶段有多条路径可以选择，但一旦数据包上行到达源和目的处理节点的最近公共祖先，它的下行路径就被唯一确定了，使得数据包在下行路由阶段可能需要通过失效链路，从而导致路由失败. 这使得胖树自身不能提供动态容错. 本文通过引入一套链路失效消息传播机制和一套基于链路失效信息的动态容错路由算法，使得数据包在胖树中传输时，能够在上行路由阶段合理地选择路径，以在下行阶段避开失效的链路，从而实现胖树的动态容错. 实验结果显示，本文提出的容错路由方法在保证网络的高可用性和高性能方面十分有效. 在 m-port n-tree 胖树中，本文提出的容错方法能够容忍任意 $m/2-1$ 条失效链路，并且以高概率容忍更多条失效链路的组合(在 8-port 3-tree 胖树中，以 99.3% 的概率容忍 10 条失效链路的组合). 同时，实验表明，在不饱和的网络中，在存在大量失效链路的情况下，容错后的网络仍能达到与无失效链路情况下相似的性能；在饱和的网络中，容错后网络的性能虽随失效链路数的增加而有所降低，但仍能保持良好的网络性能.

参 考 文 献

[1] Greenberg A, Lahiri P, Maltz D et al. Towards a next generation data center architecture: Scalability and commoditization//Proceedings of the ACM SIGCOMM Workshop on Programmable Routers for Extensible Services of Tomorrow. Seattle, WA, USA, 2008: 57-62

[2] Al-Fares M, Loukissas A, Vahdat A. A scalable, commodity data center network architecture//Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication. Seattle, WA, USA, 2008: 63-74

[3] Scott S L, Thorson G M. The Cray T3E network: Adaptive routing in a high performance 3D torus//Proceedings of the IEEE Symposium on High Performance Interconnects IV. Stanford University, 1996

[4] Kamiura N, Kodera T, Matsui N. Design of a fault-tolerant multistage interconnection network with parallel duplicated switches//Proceedings of the 15th IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems. Yamanashi, Japan, 2000: 143

[5] Konstantinidou S. The selective extra stage butterfly. IEEE Transactions on Very Large Scale Integration Systems, 1993, 1(2): 167-171

[6] Chalsani S, Raghavendra C, Varma A. Fault-tolerant routing in MIN-based supercomputers//Proceedings of the 1990 ACM/IEEE conference on Supercomputing. New York, NY, USA, 1990: 244-253

[7] Lee T H, Chou J J. Some directed graph theorems for testing the dynamic full access property of multistage interconnection networks//Proceedings of the IEEE Region 10 Conference on Computer, Communication, Control and Power Engineering. Beijing, China, 1993, 1: 217-220

[8] Sharma N K. Fault-tolerance of a MIN using hybrid redundancy//Proceedings of the 27th Annual Simulation Symposium. La Jolla, CA, 1994: 142-149

[9] Mun Y, Youn H Y. On performance evaluation of fault-tolerant multistage interconnection networks//Proceedings of the 1992 ACM/SIGAPP Symposium on Applied Computing. Kansas, Missouri, USA, 1992: 1-10

[10] Sengupta J, Bansal P. Fault-tolerant routing in irregular MINs//Proceedings of the IEEE Region 10 International Conference on Global Connectivity in Energy, Computer, Communication and Control. New Delhi, 1998, 2: 638-641

[11] Sem-Jacobsen F O, Skeie T, Lysne O et al. Siamese-Twin: A dynamically fault-tolerant fat-tree//Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium. Denver, Colorado, 2005: 100

[12] Sem-Jacobsen F O, Skeie T, Lysne O, Duato J. Dynamic fault tolerance with misrouting in fat trees//Proceedings of the 2006 International Conference on Parallel Processing. Columbus, OH, 2006: 33-44

[13] Gomez C, Gomez M E, Lopez P, Duato J. FT^{2}EI: A dynamic fault-tolerant routing methodology for fat tree with exclusion interval. IEEE Transactions on Parallel and Distributed Systems, 2009, 20(6): 802-817

[14] Mysore R N, Pamboris A, Frrington N et al. PortLand: A scalable fault-tolerant layer 2 data center network fabric//Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication. Barcelona, Spain, 2009: 39-50

[15] Greenberg A, Hamilton J, Jain N et al. VL2: A scalable and flexible data center network//Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication. Barcelona, Spain, 2009: 51-62

HU Nong-Da, born in 1983, Ph. D. candidate. His main research interests include computer architecture, high performance interconnection networks.

WANG Da-Wei, born in 1980, Ph. D., assistant professor. His main research interests include computer architecture, high performance interconnection networks, ASIC design, chip verification.

SUN Ning-Hui, born in 1968, Ph. D., professor. His main research interests include architecture of parallel computer, high performance computing, distributed OS, performance evaluation.

Background

This paper is supported by the National High Technology Research and Development Program (863 Program) of China under grant No. 2009AA01A129.

As the size and complexity of the interconnection network increase, fault tolerance of the interconnection network becomes of growing importance. Fault-tolerant methods need to be adopted to maintain high availability and high performance of the network. A fat tree as a type of Multistage Interconnection Networks is commonly used in commercial machines because of its good performance. Since the fat tree provides several alternative paths between every source and destination pair, it has good static fault tolerance abilities. However, fat tree is not able to provide dynamic fault tolerance in its original form. By introducing a link fault message spreading mechanism and a dynamic fault-tolerant routing algorithm, the authors make fat tree suitable for use with dynamic fault tolerance. Compared with previous proposals, the fault-tolerant method in this paper neither requires additional network hardware nor increases the length of routing paths. Simulation results show that, in a m-port n-tree topology, the method is able to completely tolerate all the combinations of $m/2-1$ fault links. Moreover, it can tolerate the combination of more fault links with a high probability (99.3 percent of probability to tolerate the combination of ten fault links in an 8-port 3-tree fat tree). Meanwhile, it maintains the good performance of fault-tolerant network.