



(12)发明专利

(10)授权公告号 CN 106489255 B

(45)授权公告日 2019.11.01

(21)申请号 201580037369.2

(22)申请日 2015.07.09

(65)同一申请的已公布的文献号
申请公布号 CN 106489255 A

(43)申请公布日 2017.03.08

(30)优先权数据

62/023,321 2014.07.11 US

62/049,466 2014.09.12 US

14/792,070 2015.07.06 US

(85)PCT国际申请进入国家阶段日
2017.01.10

(86)PCT国际申请的申请数据
PCT/US2015/039767 2015.07.09

(87)PCT国际申请的公布数据
WO2016/007760 EN 2016.01.14

(73)专利权人 甲骨文国际公司

地址 美国加利福尼亚

(72)发明人 F·扎西德 E·G·格兰
B·博格丹斯基 B·D·约翰森

(74)专利代理机构 中国国际贸易促进委员会专
利商标事务所 11038

代理人 边海梅

(51)Int.Cl.
H04L 12/803(2013.01)

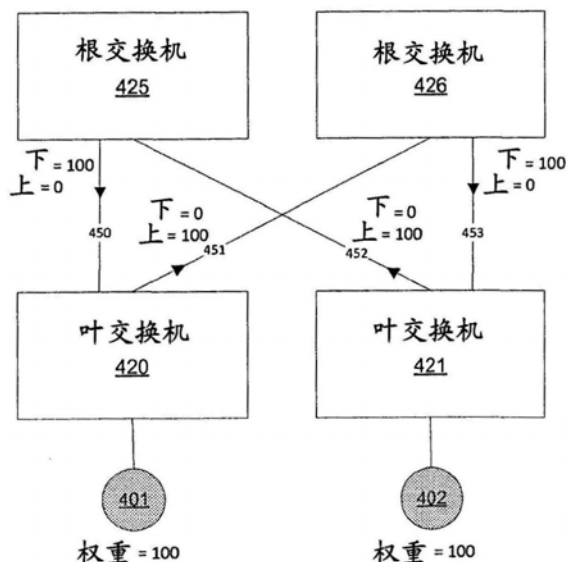
审查员 陈莹

(54)发明名称

用于在高性能计算(HPC)环境中支持高效负载均衡的系统和方法

(57)摘要

用于在网络环境中支持布置在树拓扑中的多个叶交换机和多个端节点之间的高效负载均衡的方法和系统。该系统和方法可以对多个端节点进行排序,其中该多个端节点是以接收权重的递减顺序排序的。该方法和系统还可以以接收权重的递减顺序路由多个端节点,其中该路由包括选择至少一个下行端口和至少一个上行端口。此外,该方法和系统可以以被路由的端节点的接收权重的幅度,增加每个被选择的下行端口上的累计向下权重,以及以被路由的端节点的接收权重的幅度,增加每个被选择的上行端口上的累计向上权重。



1. 一种用于在网络环境中支持布置在树拓扑中的多个交换机和多个端节点之间的高效负载均衡的方法,所述方法包括:

对所述多个端节点进行排序,所述多个端节点在所述多个交换机中的一个或多个交换机上,其中所述多个端节点是以接收权重的递减顺序排序的,其中所述接收权重指示所述多个端节点中的每一个端节点在所述网络环境中接收的业务量;

以接收权重的所述递减顺序路由所述多个端节点,其中所述路由包括选择至少一个下行端口和至少一个上行端口;

以被路由的端节点的接收权重的幅度,增加每个被选择的下行端口上的累计向下权重;以及

以所述被路由的端节点的接收权重的幅度,增加每个被选择的上行端口上的累计向上权重。

2. 如权利要求1所述的方法,其中选择所述至少一个上行端口包括比较多个上行端口以及选择具有最小累计向上权重的上行端口。

3. 如权利要求1或2所述的方法,还包括:

接收网络中的一个或多个叶交换机上的所述多个端节点中的每一个端节点的接收权重。

4. 如权利要求3所述的方法,其中网络中的一个或多个叶交换机上的所述多个端节点中的每一个端节点的接收权重是从来自管理员的输入接收的。

5. 如权利要求3所述的方法,其中网络中的一个或多个叶交换机上的所述多个端节点中的每一个端节点的接收权重是从输入接收的,所述输入分别与所述多个端节点中的每一个端节点上的被监控的业务相关。

6. 如权利要求1或2所述的方法,其中选择所述至少一个下行端口包括比较多个下行端口以及选择具有最小累计向下权重的下行端口。

7. 如权利要求1或2所述的方法,其中选择所述至少一个下行端口包括比较多个下行端口以及选择具有最小累计向上权重的下行端口。

8. 如权利要求1或2所述的方法,其中选择所述至少一个下行端口包括比较多个下行端口以及选择具有最小全局唯一标识符的下行端口。

9. 如权利要求1或2所述的方法,其中选择所述至少一个下行端口包括:

比较多个下行端口以及选择具有最小累计向下权重的下行端口;

响应于两个或更多下行端口具有最小累计向下权重,比较具有最小累计向下权重的所述两个或更多下行端口,以及从具有最小累计向下权重的所述两个或更多下行端口中选择具有最小累计向上权重的下行端口;以及

响应于所述两个或更多下行端口具有最小累计向下权重和最小累计向上权重,比较具有最小累计向下权重和最小累计向上权重的所述两个或更多下行端口,以及从具有最小累计向下权重和最小累计向上权重的所述两个或更多下行端口中选择具有最小全局唯一标识符的下行端口。

10. 一种用于在网络环境中支持布置在树拓扑中的多个交换机和多个端节点之间的高效负载均衡的系统,所述系统包括:

一个或多个微处理器;以及

处理器,所述处理器运行在所述一个或多个微处理器上,其中所述处理器操作以执行步骤,所述步骤包括:

对所述多个端节点进行排序,所述多个端节点在所述多个交换机中的一个或多个交换机上,其中所述多个端节点是以接收权重的递减顺序排序的,其中所述接收权重指示所述多个端节点中的每一个端节点在所述网络环境中接收的业务量;

以接收权重的所述递减顺序路由所述多个端节点,其中所述路由包括选择至少一个下行端口和至少一个上行端口;

以被路由的端节点的接收权重的幅度,增加每个被选择的下行端口上的累计向下权重;以及

以所述被路由的端节点的接收权重的幅度,增加每个被选择的上行端口上的累计向上权重。

11.如权利要求10所述的系统,其中所述多个交换机和所述多个端节点被布置在胖树拓扑中以用于在高性能计算(HPC)集群中的使用。

12.如权利要求10或11所述的系统,其中所述处理器操作以接收网络中的一个或多个叶交换机上的所述多个端节点中的每一个端节点的接收权重。

13.如权利要求10或11所述的系统,其中选择所述至少一个下行端口包括比较多个下行端口以及选择具有最小累计向下权重的下行端口。

14.如权利要求10或11所述的系统,其中选择所述至少一个下行端口包括比较多个下行端口以及选择具有最小累计向上权重的下行端口。

15.如权利要求10或11所述的系统,其中选择所述至少一个下行端口包括比较多个下行端口以及选择具有最小全局唯一标识符的下行端口。

16.如权利要求10或11所述的系统,其中选择所述至少一个下行端口包括:

比较多个下行端口以及选择具有最小累计向下权重的下行端口;

响应于两个或更多下行端口具有最小累计向下权重,比较具有最小累计向下权重的所述两个或更多下行端口,以及从具有最小累计向下权重的所述两个或更多下行端口中选择具有最小累计向上权重的下行端口;以及

响应于所述两个或更多下行端口具有最小累计向下权重和最小累计向上权重,比较具有最小累计向下权重和最小累计向上权重的所述两个或更多下行端口,以及从具有最小累计向下权重和最小累计向上权重的所述两个或更多下行端口中选择具有最小全局唯一标识符的下行端口。

17.一种具有存储于其上的指令的非暂态机器可读存储介质,所述指令用于在网络环境中支持布置在树拓扑中的多个交换机和多个端节点之间的高效负载均衡,所述指令被执行时使系统执行步骤,所述步骤包括:

对所述多个端节点进行排序,所述多个端节点进一步在所述树拓扑中的所述多个交换机中的一个或多个交换机上,其中所述多个端节点是以接收权重的递减顺序排序的,其中所述接收权重指示所述多个端节点中的每一个端节点在所述网络环境中接收的业务量;

以接收权重的所述递减顺序路由所述多个端节点,其中所述路由包括选择至少一个下行端口和至少一个上行端口;

以被路由的端节点的接收权重的幅度,增加每个被选择的下行端口上的累计向下权

重;以及

以所述被路由的端节点的接收权重的幅度,增加每个被选择的上行端口上的累计向上权重。

18.如权利要求17所述的非暂态机器可读存储介质,其中选择所述至少一个下行端口包括比较多个下行端口以及选择具有最小累计向下权重的下行端口。

19.如权利要求17所述的非暂态机器可读存储介质,其中选择所述至少一个下行端口包括比较多个下行端口以及选择具有最小累计向上权重的下行端口。

20.如权利要求17所述的非暂态机器可读存储介质,其中选择所述至少一个下行端口包括比较多个下行端口以及选择具有最小全局唯一标识符的下行端口。

21.一种包括用于执行如权利要求1至9中任一项所述的方法的部件的装置。

用于在高性能计算 (HPC) 环境中支持高效负载均衡的系统和 方法

[0001] 版权声明

[0002] 本专利文档的公开内容的一部分包含受版权保护的材料。版权所有人反对任何人对专利文档或专利公开内容按照在专利商标局的专利文件或记录中出现的那样进行传真复制,但是除此之外在任何情况下都保留所有版权。

技术领域

[0003] 本发明一般地涉及计算机系统,并且具体地涉及网络环境。

背景技术

[0004] 胖树拓扑 (fat-tree topology) 被用于高性能计算 (HPC) 集群以及用于基于 InfiniBand™ (IB) 技术的集群。例如,胖树拓扑被用在诸如天河2号之类的最快的超级计算机中。此外,胖树IB系统包括诸如Stampede、TGCC Curie和SuperMUC之类的大型设施。

[0005] 这些是本发明的实施例旨在解决的一般领域。

发明内容

[0006] 本文描述的是用于在网络环境中支持布置于树拓扑中的多个交换机和多个端节点之间的高效负载均衡的系统和方法。该系统和方法可以对多个端节点进行排序,多个端节点在树拓扑中的一个或多个叶交换机上,其中多个端节点是以接收权重的递减顺序排序的。该系统和方法可以以接收权重的递减顺序路由多个端节点,其中该路由包括选择至少一个下行端口和至少一个上行端口。该系统和方法可以以被路由的端节点的接收权重的幅度增加每个被选择的下行端口上的累计向下权重。最后,该系统和方法可以以被路由的端节点的接收权重的幅度增加每个被选择的上行端口上的累计向上权重。

[0007] 在实施例中,本文所述的系统和方法可以允许多个交换机和多个端节点位于胖树拓扑中以用于在高性能计算 (HPC) 集群中使用。附加地,在实施例中,一个或多个叶交换机上的多个端节点的接收权重可以通过系统接收。

[0008] 在实施例中,由该方法和系统执行的对至少一个下行端口的选择可以包括比较多个下行端口以及选择具有最小累计向下权重的下行端口。

[0009] 在实施例中,由该方法和系统执行的对至少一个下行端口的选择包括比较多个下行端口以及选择具有最小累计向上权重的下行端口。

[0010] 在实施例中,由该方法和系统执行的对至少一个下行端口的选择包括比较多个下行端口以及选择具有最小全局唯一标识符的下行端口。

[0011] 在实施例中,由该方法和系统执行的对至少一个下行端口的选择包括比较多个下行端口以及选择具有最小累计向下权重的下行端口。当两个或更多下行端口具有最小累计向下权重时,该系统和方法可以进一步比较具有最小累计向下权重的两个或更多下行端口,以及从具有最小累计向下权重的两个或更多下行端口中选择具有最小累计向上权重的

下行端口。当两个或更多下行端口具有最小累计向下权重和最小累计向上权重时,该方法
和系统可以比较具有最小累计向下权重和最小累计向上权重的两个或更多下行端口,以及
从具有最小累计向下权重和最小累计向上权重的两个或更多下行端口中选择具有最小全
局唯一标识符的下行端口。

附图说明

- [0012] 图1是网络环境中的胖树路由的框图,在其中可以实践本发明的实施例。
- [0013] 图2是网络环境中的胖树路由的框图,在其中可以实践本发明的实施例。
- [0014] 图3是网络环境中的胖树路由的框图,在其中可以实践本发明的实施例。
- [0015] 图4是示出根据本发明实施例的在网络环境内的示例性端口选择的框图。
- [0016] 图5是示出根据本发明实施例的在网络环境内的示例性端口选择的框图。
- [0017] 图6是示出根据本发明实施例的在网络环境内的示例性端口选择的框图。
- [0018] 图7是示出根据本发明实施例的在网络环境内的示例性端口选择的框图。
- [0019] 图8是示出根据本发明实施例的在网络环境内的示例性端口选择的框图。
- [0020] 图9是示出根据本发明实施例的在网络环境内的示例性端口选择的框图。
- [0021] 图10是示出根据本发明实施例的方法的流程图,该方法用于在网络环境中支持布
置在树拓扑中的多个交换机和多个端节点之间的高效负载均衡。

具体实施方式

[0022] 在以下详细描述中,本发明将以示例的方式而非限制的方式在附图的图中示出。
应当注意的是,本公开中对“一”或者“一个”或者“一些”实施例的引用并不必然针对同一实
施例,并且这样的引用意味着至少有一个。尽管讨论了特定的实施方案,但是要理解的是,
特定的实施方案只是为了示意性的目的而提供的。相关领域中的技术人员将认识到,在不
脱离本发明的范围和精神的情况下也可以使用其他部件和配置。

[0023] 在整个附图和详细描述中,共同的参考标号被用来指示同样的元件;因此,如果元
件在其他地方被描述,则在图中使用的参考标号可以或者可以不在特定于这样的图的详细
描述中被引用。

[0024] 本发明的以下描述使用InfiniBand™ (IB) 网络作为高性能网络的示例。对本领域
技术人员清晰的是,可以使用其他类型的高性能网络而不受限制。以下描述也使用胖树拓
扑作为组构拓扑(fabric topology)的示例。对本领域技术人员清晰的是,可以使用其他类
型的组构拓扑而不受限制。

[0025] InfiniBand™

[0026] InfiniBand™ (IB) 是由InfiniBand™贸易组织开发的开放标准无损网络技术。该
技术基于提供高吞吐量和低延迟通信的串行点对点全双工互连结构,特别适用于HPC应用
和数据中心。

[0027] InfiniBand™架构(IBA)支持两层拓扑划分。在较低层处,IB 网络被称为子网,其
中子网可以包括使用交换机和点对点链路互连的一组主机。在较高层处,IB组构成一个
或多个子网,其可以使用路由器互连。

[0028] 在子网内,使用交换机和点对点链路连接主机。附加地,存在一个主管理实体——

子网管理器 (SM)，子网管理器驻留在子网中指定的子网设备上。子网管理器负责配置、激活和维护IB子网。附加地，子网管理器 (SM) 可以负责在IB组构中执行路由表计算。在此，例如，IB网络的路由目的在于本地子网中全部源和目的地对之间的适当的负载均衡。

[0029] 通过子网管理接口，子网管理器与子网管理代理 (SMA) 交换控制包，该控制包被称为子网管理包 (SMP)。子网管理代理驻留在每个IB子网设备上。通过使用SMP，子网管理器能够发现组构、配置端节点和交换机、以及从SMA接收通知。

[0030] 一般地，除主子网管理器外的全部其他子网管理器都工作于待机模式中以用于容错。然而，在主子网管理器失效的情况下，由待机的子网管理器协商得出新的主子网管理器。主子网管理器也执行对子网的定期扫描以检测任何拓扑改变以及相应地重新配置网络。

[0031] 此外，可以使用本地标识符 (LID) 为子网内的主机和交换机寻址，并且单个子网可以被限制为49151个LID。除作为在子网内有效的本地地址的LID以外，每个IB设备还可以具有被烧入其非易失性存储器的64位全局唯一标识符 (GUID)。GUID可以被用来形成全局标识符 (GID)，该全局标识符是IB层3 (L3) 地址。可以通过将 64位子网标识符 (ID) 与64位GUID联接以形成与IPv6类似的128 位地址而创建GID。例如，不同端口GUID可以被分配给连接到IB组构的端口。

[0032] SM可以在网络初始化时计算路由表 (即，树内每个节点对之间的连接/路由)。此外，无论何时拓扑改变都可以更新路由表，以便确保最优的性能。在正常操作期间，SM可以执行对网络的定期轻扫描以检查拓扑改变。如果在轻扫描期间发现改变，或者如果由SM接收到表明网络改变的消息 (陷阱 (trap))，则SM可以根据所发现的改变重新配置网络。

[0033] 例如，当网络拓扑改变时 (诸如当链路关闭时、当添加设备时或者当移除链路时)，SM可以重新配置网络。重新配置步骤可以包括在网络初始化期间执行的步骤。此外，重新配置可以具有本地范围，该本地范围限于在其中发生了网络改变的子网。此外，利用路由器对大型组构进行分段可以限制重新配置范围。

[0034] 胖树路由

[0035] 胖树拓扑是通用网络拓扑的可扩展类。胖树拓扑背后的初始想法是将网络拓扑布置为交换机的分层的、多根树结构，其中端节点驻留在叶交换机处。通过使用朝向胖树的根越来越胖的链路，可以维护完全平分 (bisection) 带宽并且潜在地避免拥塞。这可以附加地提供使用任何可用带宽的好处。

[0036] 例如在HPC环境内，胖树拓扑可以为支持高性能互连提供各种好处。这些好处可以包括无死锁 (deadlock freedom)、固有容错以及完全平分带宽。无死锁表示树结构的使用使得在不特别考虑死锁避免的情况下路由胖树成为可能。由于在各个源目的地对之间存在多个路径而导致的固有容错允许网络故障的高效处置。完全平分带宽允许网络支撑网络的两半之间的全速通信。

[0037] 胖树路由算法可以旨在生成线性转发表 (LFT)，线性转发表跨网络组构中的链路均匀地分布最短路径路由。该算法可以以索引顺序遍历组构，以及将端节点的目标LID以及因此对应的路由分配给每个交换机端口。

[0038] 此外，胖树路由算法可以被用来支持底层胖树拓扑的高效使用。下面的算法1是示例性的胖树路由算法。

算法 1 *route_to_cns()* 函数**要求:** 寻址完成**确保:** 所有 hca_ports 都被路由

```

1: for swleaf = 0 to max_leaf_sw do
[0039] 2:   for swleaf.port = 0 to max_ports do
3:     hca_lid = swleaf.port->remote_lid
4:     swleaf.routing_table[hca_lid] = swleaf.port
5:     route_downgoing_by_going_up()
6:   end for
7: end for

```

[0040] 如以上所示,路由函数route_to_cns()可以在叶交换机的数组上迭代(行1-7)。例如在端口编号指令序列中(行2-6),对于每个被选择的叶交换机,路由函数可以路由连接到被选择的叶交换机的每个端节点端口。

[0041] 此外,当路由与特定LID关联的端节点端口时,路由函数可以在网络拓扑中上行一层以路由下行路径,而当路由每个交换机端口时,路由函数可以下行以路由上行路径。此过程可以被重复,直到到达根交换机级。在此之后,朝向全部节点的路径可以被路由和被插入到组构中全部交换机的线性转发表(LFT)中。

[0042] 例如,route_downgoing_by_going_up()函数(行5)可以是能够均衡路径并调用route_upgoing_by_going_down()函数的递归函数,route_upgoing_by_going_down()函数通过从其调用route_downgoing_by_going_up()函数的交换机朝向目的地路由胖树中的向上路径。

[0043] 可能存在与route_to_cns()函数关联的若干潜在的缺陷。首先,route_to_cns()是不经意(oblivious)的并且在不考虑端端口(end-port)属于哪个端节点的情况下路由由端端口。其次,route_to_cns()函数依赖物理端口编号来路由。

[0044] 图1示出在网络环境中的胖树路由的示意图,在其中可以实践本公开的实施例。如图1所示,在网络组构100中可以连接一个或多个端节点101-104。网络组构100可以是基于胖树拓扑的,网络组构100包括多个叶交换机111-114以及多个脊交换机或者根交换机131-134。附加地,网络组构100可以包括一个或多个中间交换机,诸如交换机121-124。

[0045] 还如图1中所示,端节点101-104中的每个端节点可以是多宿主节点,即通过多个端口连接到网络组构100的两个或更多部分的单个节点。例如,节点101可以包括端口H1和H2,节点102可以包括端口H3和H4,节点103可以包括端口H5和H6,而节点104可以包括端口H7和H8。

[0046] 附加地,每个交换机可以具有多个交换机端口。例如,根交换机131可以具有交换机端口1-2,根交换机132可以具有交换机端口3-4,根交换机133可以具有交换机端口5-6,而根交换机134可以具有交换机端口7-8。

[0047] 图2示出在网络环境200中的胖树路由的示意图,在其中可以实践本公开的实施例。图2描绘k元n级树(k-ary-n-tree),k元n级树是具有 k^n 个端节点以及 $n \times k^{n-1}$ 个交换机、每个交换机具有2k个端口的n级树。更具体地,图2描绘了4元2级树,即该胖树拓扑具有2级、16个端节点(201-216)以及8个交换机(4个叶交换机220-223以及4个根交换机225-

228), 其中每个交换机具有8个端口。

[0048] 传统胖树路由算法(在本文各处称为FTree)旨在生成跨网络组构中的链路均匀分布最短路径路由的LFT。该算法一般以索引顺序遍历组构并且将端节点的目标LID以及因此对应的路由分配给每个交换机端口。对连接到同一叶交换机的端节点,索引顺序取决于端节点所连接到的交换机端口(端口编号次序)。对每个端口,该算法可以维护端口使用计数器并且在每次添加路由时用它来选择最少使用的端口(如果有多于一个选择是可用的)。如果存在连接到相同两个交换机的多个端口,则这样的端口形成端口组。在此情况下,最小负载的端口组中最少使用的端口被选择来添加到新的路由。

[0049] 一般地,到LID的端口分配是在两个阶段中递归执行的,其从叶交换机处开始。在第一阶段中,算法从每个端节点向上朝向树根下遍历,从而将下行端口分派到LID。在下行端口被设置之后,算法通过下树(descending down)将向上端口分配到全部已连接的向下交换机上的LID。随后可以通过向上移动到树的下一级递归地重复该过程。

[0050] 存在与用于胖树拓扑的传统路由机制(即,FTree算法)关联的两个缺点。

[0051] 首先,被用于胖树拓扑的标准算法所使用的负载均衡技术寻求在不考虑任何节点的业务特性的情况下跨拓扑中的链路均衡负载。换句话说,传统胖树算法为网络中的全部节点假定相同的权重。然而,在HPC集群中,不同节点经常具有确定其业务概况的预先分配的角色。例如,存储节点或者I/O网关可能比其他节点消耗更多业务。因此,在网络中朝向这些高业务节点的路由可能更加拥塞并且需要优先。当在不考虑特定节点的业务需求的情况下进行路由时,它可能导致次优的网络吞吐量,因为某些链路被超额预订(oversubscribe)而备选的链路未被充分使用。

[0052] 其次,用于胖树拓扑的传统算法是不合需要的,因为它可能导致不可预测的性能。此不可预测的性能产生是因为该算法根据索引顺序将路由分配到链路。然而,索引顺序是不可配置的并且取决于端节点所连接的叶交换机的端口数量。因此,以相同方式连线的胖树系统可能显现出不同且不可预测的性能。作为示例,在两级胖树中,如果在不同叶交换机处的两个端节点共享相同的索引位置,则朝向这两个节点的业务将被路由通过同一根节点。作为结果,前往这两个节点但来源于其他叶交换机处的端节点的全部业务也将争夺对连接到单个根交换机的向上链路的公共集合的访问,即使可能存在通过备选根交换机的若干更少负载的路径。

[0053] 为了更好地示出传统胖树路由算法的问题,考虑图2中的示例的路由是有帮助的。在图2中,节点201、206、210和213被阴影化,以便表示4个接收者节点或者已知接收总网络业务的大部分的节点。4个叶交换机220-223中的每个叶交换机被连接到4个根交换机225-228。假定节点从左向右符合索引顺序(即,节点201具有索引顺序1、节点206具有索引顺序2、节点210具有索引顺序2、而节点213具有索引顺序1),这意味着节点206和节点213共享相同的索引顺序(即,1)并且节点206和节点210同样地共享相同的索引顺序(即,2)。因此,胖树路由算法将只使用两个最左边的根交换机225和226来路由朝向这4个端节点的业务。这导致上行方向上的4个潜在超额预订的链路,如图2中虚线所示。在图2中虚线被标签为“上{a,b}”以指示朝向接收者节点a和b的向上流将争夺链路上的带宽。

[0054] 作为示例,即使在拓扑中存在足够多的可用链路而能够避免前往接收者节点201和213的业务流之间的干扰,传统胖树算法还是会使得到节点206和210的两个独立流共享从

最左边的叶交换机220起的同一向上链路。

[0055] k 元 n 级树的索引冲突概率

[0056] 如上面讨论的,如果在网络中作为承担系统内大部分业务的节点的接收者节点在其各自叶交换机处共享索引位置,则FTree的性能可能恶化。例如,在图2中,接收者节点201与接收者节点213共享索引位置1,而接收者节点206和接收者节点210共享索引位置2。因此,当涉及负载均衡时,确定这种索引冲突(也就是何时接收者节点在不同叶交换机处共享相同的索引位置)的概率以便评估FTree的可行性是重要的。

[0057] 回想到 k 元 n 级树是具有 k^n 个端节点以及 $n \times k^{n-1}$ 个交换机、每个交换机具有 $2k$ 个端口的 n 级树。假定用端节点完全填充的树,并且级 $l=n$:

[0058] 每个端节点由 n 元组 $\{0, 1, \dots, k-1\}^n$ 表示,并且每个交换机由顺序对 $\langle s, l \rangle$ 表示,其中 $s \in \{0, 1, \dots, k-1\}^{n-1}$,而级 $l \in \{0, 1, \dots, n-1\}$ 。

[0059] 叶交换机被定义为具有到在级 n 的端节点的边 $c_0, c_1, \dots, c_{n-1}$ 的级 $n-1$ 交换机 $\langle l_0, l_1, \dots, l_{n-2}, n-1 \rangle$ 。

[0060] 在存在 y 个节点、其中在每个叶交换机处的 k 个端节点中这 y 个节点中每个节点具有网络中业务接收的较高比例(例如,接收者节点)的情况下,接收者节点在叶交换机处的任意索引位置 i 处被找到的概率由以下公式给出:

$$[0061] \quad p_i = \frac{y}{k}$$

[0062] 因为胖树具有 $N=k^{n-1}$ 个叶交换机,所以二项分布可以被用来得到接收者节点在其对应的交换机处共享相同的索引位置的概率。在概率为 p_i 的情况下,在任意索引位置 i 处找到恰好 r 个接收者节点的概率由以下给出:

$$[0063] \quad f(r, N, p_i) = \binom{N}{r} p_i^r (1 - p_i)^{N-r}$$

[0064] 为了计算在位置 i 处获得至少 x 个索引冲突的概率,对全部对应的概率求和,如这里所示:

$$[0065] \quad f'(x, N, p_i) = \sum_{j=x+1}^N f(j, N, p_i), x < N$$

[0066] 注意,对于在每个叶交换机处具有 R 个连接的端节点的胖树, R 个位置中的任意位置($i \in \{1, 2, 3, \dots, R\}$)处的索引冲突可能导致网络争用加剧。

[0067] 加权的胖树路由算法

[0068] 根据本公开的实施例,加权的胖树路由算法(通篇各处称为 wFatTree)被使用,以便克服上面的FTree的不足。在wFatTree内,每个端节点被分配新的参数receive_weight,当计算系统内的路由时,该参数可以被用来考虑已知的或学习得到的业务特性。

[0069] 在实施例中,每个端节点的receive_weight参数的值反映在计算路由表时朝向节点的流所接受的优先程度。作为示例,配置可以是在范围 $[1, 100]$ 中将权重分配给端节点。每个节点将接受权重,该权重取决于已知节点在网络中接收多少业务。在此示例中,端节点可以被分配receive_weight为1。这将表示接收非常少业务的节点(业务生成节点)。附加

地,接收接近链路容量的业务的端节点可以被分配 `receive_weight` 为100。在这种情况下,在1和100之间的`receive_weight` 的值于是将表示在网络中节点接收的业务的比例。

[0070] 在另一实施例中,节点可以接受`receive_weight`为500,同时网络中的全部其他节点被赋予`receive_weight`为1。这将指示具有 `receive_weight`为500的端节点是关键节点,并且流向关键节点的业务应当被优先。

[0071] 在实施例中,wFatTree路由算法(下面算法2中所示)在三个阶段中递归地工作。在此实施例中,所有路由是后向计算的,其从目的地节点处开始并且以反方向工作。以下的算法2是示例性的 wFatTree路由算法。

算法 2 wFatTree 路由算法

```

1: procedure ROUTETOENDNODES
2:   for all  $s \in \text{leafSwitches}[]$  do
3:     sort end nodes connected to  $s$  by  $\text{rcv\_weight}$ 
4:     for all  $c \in \text{endNodes}[]$  do
5:        $s.\text{LFT}[c.\text{LID}] \leftarrow c.\text{port}$ 
6:       ROUTEDOWNGOINGBYASC( $s, c$ )
7:     end for
8:   end for
[0072] 9: end procedure

10: procedure ROUTEDOWNGOINGBYASC( $s, c$ )
11:    $p \leftarrow \text{GETLEASTLOADEDPORT}(s, \text{UpGroups}[])$ 
12:    $r\text{Switch} \leftarrow p.\text{Switch}$ 
13:    $r\text{Switch}.\text{LFT}[c.\text{LID}] \leftarrow p$ 
14:    $p.\text{Dwn} += c.\text{rcv\_weight}$ 
15:   ROUTEUPGOINGBYDESC( $s, c$ )
16:   ROUTEDOWNGOINGBYASC( $r\text{Switch}, c$ )
17: end procedure

18: procedure ROUTEUPGOINGBYDESC( $s, c$ )
19:   for all  $g \in \text{DownGroups}[]$  do
20:     skip  $g$  if the  $\text{LFT}(c.\text{LID})$  is part of this group
21:      $p \leftarrow \text{GETLEASTLOADEDPORT}(g)$ 
22:      $r\text{Switch} \leftarrow p.\text{Switch}$ 
[0073] 23:      $r\text{Switch}.\text{LFT}[c.\text{LID}] \leftarrow p$ 
24:      $p.\text{Up} += c.\text{rcv\_weight}$ 
25:     ROUTEUPGOINGBYDESC( $r\text{Switch}, c$ )
26:   end for
27: end procedure

```

[0074] 在实施例中,在示例性算法2的第一阶段期间,每个叶交换机处的端节点根据递减的`receive_weight`被排序(行3)。(注意,在算法2中,`receive_weight`被缩写为“`rcv_weight`”)。如先前提到的, `receive_weight`可以或者由管理员提供,或者它们也可以被计算得出。此话题的进一步讨论稍后出现。

[0075] 在实施例中,在示例性算法2的第二阶段期间,wFatTree从每个端节点(例如,目的地端节点或者路由的目的地)向上遍历树,从而为当前节点分派在下一级中的被选择的交换机处的下行端口(ROUTEDOWGOINGBYASC,示例性的算法2的行6)。当下行端口被选择时,该算法以被路由的端节点的receive_weight的幅度,增加对应端口的累计向下权重(行14)。这标志新的加权的路由已经被添加到对应的端口。

[0076] 在实施例中,在设置了下行端口之后,在示例性算法2的第三阶段中,该算法通过下树在全部已连接的向下交换机上为朝向端节点的路由分配上行端口(ROUTEUPGOIGNBYDESC)(以及通过加上被路由的端节点的receive_weight来更新端口的对应的向上权重)。随后通过上移到树中的下一级来重复整个3阶段过程(行16)。

[0077] 在实施例中,作为wFatTree算法的算法为每个路由计算选择最小负载的端口。选择标准首先基于向下权重。在两个端口具有相等的向下权重的情况下,选择具有最小向上权重的端口。附加地,在向下权重和向上权重两者都相等的情况下,算法选择具有最小GUID的端口以保持该过程是确定性的。以下示例性算法3示出wFatTree如何为每个路由计算选择最小负载的端口:

算法 3 获取全部端口组中最小负载的端口

```

1: procedure GETLEASLOADEDPORT(PortGroups[])
2:   min  $\leftarrow$  null
3:   for all g  $\in$  PortGroups[] do
4:     for all port  $\in$  g.Ports[] do
5:       if port.Dwn < min.Dwn then
6:         min  $\leftarrow$  port
7:       else if port.Dwn = min.Dwn then
8:         if port.Up < min.Up then
[0078] 9:           min  $\leftarrow$  port
10:        else if port.Up = min.Up
11:          if port.GUID < min.GUID then
12:            min  $\leftarrow$  port
13:          end if
14:        end if
15:      end if
16:    end for
17:  end for
18: end procedure

```

[0079] 在实施例中,wFatTree以若干方式改进传统FTree路由算法。首先,不像如上面提到的基于网络中节点的索引(其中叶交换机处的每个节点被索引)的FTree,wFatTree以递减的接收权重的顺序路由节点。这允许例如是接收者节点(例如,具有系统内业务的大比例的节点)的节点首先被路由。附加地,在交换机处的向下端口被分配到端节点的情况下,wFatTree更新与该节点关联的其他本地链路上的向上权重。这允许在选择链路时考虑向上权重,因为向上链路潜在地携带朝向该节点的业务。最后,在选择了最少使用的向下端口的情况下,wFatTree在检查了向下权重之后也检查所分配的向上权重,以选择最小竞争的端

口。相对于只检查向下方向上被路由的链路的数量的传统 FTree,这提供改进的负载均衡的好处。当发现向下链路数量相同时,传统FTree回到索引顺序来确定路由。

[0080] 现在参考图3,图3描绘了在网络环境中的胖树路由,在其中可以实践本公开的实施例。网络环境300(被描绘为 k 元 n 级树,该 k 元 n 级树是具有 k^n 个端节点以及 $n \times k^{n-1}$ 个交换机、每个交换机具有 $2k$ 个端口的 n 级胖树)。与图2类似,高亮的节点301、306、310和313被指定为接收者节点,因为例如这些节点携带总网络业务中的大部分。在图3所描绘的实施例中,使用wFatTree路由算法。作为wFatTree路由算法的结果,网络环境现在在计算路由时考虑每个节点的receive_weight。

[0081] 在实施例中,如图3中所示,流向接收者节点301的向上业务(该向上业务由具有指示方向的实箭头的虚线表示)全部通过根交换机325。随后,到接收者节点301的向下路由(由具有实箭头的实线表示)从根交换机325流经叶交换机320。类似地,目的地为接收者节点313的向上业务(该向上业务由具有指示方向的空箭头的虚线表示)在被下路由到叶交换机323之前全部通过根交换机328。在实施例中,也为流向接收者节点306和310的业务呈现类似的业务模式。

[0082] 在图3中采用的wFatTree路由算法展示对网络环境300内的可用链路的改进的分配。相对于使用传统FTree的网络,这允许网络中的改进的性能。

[0083] 在实施例中,当交换机处的向下端口被选择用于朝向端节点的路由时,到该交换机的前往该端节点的全部入业务都被路由通过所选择的端口。具体而言,如果全部链路都是全双工的,则连接到交换机的所有其它向上链路潜在地在向上方向上携带朝向所讨论的端节点的业务。在设置了所选择的端口的向下权重之后,wFatTree用被路由到的节点的receive_weight标志全部可用的向上链路。在具有相同向下负载的多个向下端口可用的情况下,当为路由选择下一个向下部分时,选择具有最小向上权重的端口。将选择基于向下权重和向上权重两者确保网络中的链路根据端节点的receive_weight而被均衡。

[0084] 图4描绘根据实施例的在网络环境内的示例性端口选择。图4中描绘的网络环境包括叶交换机420和421、根交换机425和426、以及端节点401和402。在整个图4-图9的讨论中,将假定网络环境使用wFatTree算法,并且端节点401和端节点402两者在它们各自的叶交换机处具有同样的索引位置以及同样的接收权重100(即对于端节点401和端节点402两者receive_weight=100)。

[0085] 图5描绘根据实施例的在网络环境内的示例性端口选择。图5中描绘的网络环境包括叶交换机420和421、根交换机425和426、以及端节点401和402。图5示出当计算朝向端节点401的路由时,在两个不同的根交换机425和426上,具有相同向下权重的两个上游端口、链路450和451是可用的。在所描绘的实施例中,在所描绘的向上方向上尚无权重,并且在向下方向上也尚无权重。因此,链路450上的权重为:上=0,下=0;同样地,链路451上的权重为:上=0,下=0。

[0086] 图6描绘根据实施例的在网络环境内的示例性端口选择。图6中描绘的网络环境包括叶交换机420和421、根交换机425和426、以及端节点401和402。如图6中所描绘的,因为在向下方向和向上方向上尚无权重,所以最左边的根交换机425上的端口被选择,因为它携带有较小的GUID。因此,链路450因而在下方向上携带有端节点401的receive_weight。因此,链路450上的权重为:上=0,下=100。附加地,链路452在向上方向上携带有端节点401的

receive_weight。因此,链路452上的权重为:上=100,下=0。

[0087] 图7描绘根据实施例的在网络环境内的示例性端口选择。图7中描绘的网络环境包括叶交换机420和421、根交换机425和426、以及端节点401和402。如类似于图5的图7中所描绘的,计算到端节点 402的路由。如图5中所示,存在具有相同向下权重的两个上游端口。两条链路452和453具有相等的向下权重,也就是0。然而,链路452 具有向上权重100,而链路453具有向上权重0。这意味着453的向上权重小于452的向上权重。如上面讨论的,当两条链路具有相同的向下权重时,如果一条链路相比另一条链路具有更大的向上权重,则算法将会选择具有较小向上权重的端口来下路由。

[0088] 图8描绘根据实施例的在网络环境内的示例性端口选择。图8中描绘的网络环境包括叶交换机420和421、根交换机425和426、以及端节点401和402。如图8中所描绘的,因为链路452(见图7)上的向上权重大于链路453上的向上权重,所以链路453被选择来下行(carry down)到节点402。作为结果,链路453因而在下方向上携带有端节点402的receive_weight。因此,链路453上的权重为:上= 0,下=100。附加地,如上面讨论的,链路451在向上方向上携带有端节点402的receive_weight。因此,链路451上的权重为:上=100, 下=0。

[0089] 图9描绘根据实施例的在网络环境内的示例性端口选择。图9中描绘的网络环境包括叶交换机420和421、根交换机425和426、以及端节点401和402。图9示出在全部链路权重被更新之后的最终路由。具体而言,将注意到,使用拓扑中可用的链路,到端节点401和402的路由被很好地均衡,尽管两个接收者节点在它们各自的叶交换机处共享相同的索引位置。

[0090] 图10经由流程图描绘用于在网络环境中支持布置在树拓扑中的多个交换机和多个端节点之间的高效负载均衡的示例性方法1000。在步骤1001处,示例性方法1000以对多个端节点进行排序而开始,该多个端节点在多个交换机中的一个或多个交换机上,其中该多个端节点以接收权重的递减顺序排序。在实施例中,端节点的接收权重可以是1和100或者另一个合适的范围之间的值,其中越大的接收权重的值表示相应的节点相对具有较小接收权重的节点在比例上具有网络中业务的越大份额。

[0091] 在步骤1002处,示例性方法1000通过以接收权重的递减顺序路由多个端节点而继续,其中该路由包括选择至少一个下行端口和至少一个上行端口。通过以递减顺序路由,这允许网络将到接收较多业务的端节点的业务优先以及降低端口冲突的可能性。在一些实施例中,选择是基于各自的接收权重的。

[0092] 在步骤1003处,示例性方法1000可以通过以被路由的端节点的接收权重的幅度增加每个被选择的下行端口上的累计向下权重而继续。

[0093] 在步骤1004处,示例性方法1000可以通过以被路由的端节点的接收权重的幅度增加每个被选择的上行端口上的累计向上权重而继续。

[0094] 计算receive_weight

[0095] 在实施例中,当关于节点的管理性信息不可用时,或者更具体地,当节点的receive_weight未被提供或者未以其他方式可用时,可以计算receive_weight。在使用OFED(OpenFabrics企业分布)的实施例中,称为ibdatacounts的实用程序被提供以用于读取数据计数器。在设立网络并且给每个节点赋予相等的receive_weight之后,可以在指定

时间段之后计算得出或学习得到新的权重。

[0096] 在实施例中,如果B是一段时间上测量得到的全部节点的接收带宽的集合,则通过使用线性变换可以在范围[a,b]中分配每个节点的权重,该线性变换如此示例性等式中给出的:

$$[0097] \quad W(x) = (x - a) \frac{b - a}{\max(B) - \min(B)} + a, \forall x \in B$$

[0098] 在实施例中,一旦已经从数据计数器获得新的权重集合,就可以使用优化后的路由表重新配置网络。然而,注意在实施例中可以执行均衡测试,该测试对重新配置要被优化的路由表的好处和这种重新配置将要求的停机时间进行均衡。在实施例中,路由表的重新配置可以被推迟到由诸如拓扑改变之类的外部因素引发重新配置的这种时候。

[0099] 在一个实施例中,可以提供用于在中间件机器环境中支持布置在树拓扑中的多个交换机和多个端节点之间的高效负载均衡的系统,该系统包括:与多个交换机和多个端节点通信的子网管理接口;以及耦合到子网管理接口并且被配置为对多个端节点进行排序的子网管理器,多个端节点在多个交换机中的一个或多个交换机上,其中多个端节点以接收权重的递减顺序排序。该系统可以以接收权重的递减顺序路由多个端节点,其中该路由包括选择至少一个下行端口和至少一个上行端口;以被路由的端节点的接收权重的幅度增加每个被选择的下行端口上的累计向下权重;以及以被路由的端节点的接收权重的幅度增加每个被选择的上行端口上的累计向上权重。

[0100] 对本领域技术人员清晰的是,子网管理器可以结合子网管理接口实现如以上陈述的任何胖树路由算法或者胖树路由算法的步骤中的一步或多步。对本领域技术人员清晰的是,对于诸如上述子网管理接口和子网管理器之类的单元/模块的具体操作过程,可以参考相关方法实施例中共享相同概念的对应步骤,并且该参考也被认为是相关单元/模块的公开。并且因此,为了方便和描述的简洁,将不再重复或详细地描述一些特定的操作过程。应当理解的是,诸如子网管理接口、子网管理器和/或系统之类的单元、装置和设备可以以已知的或未来开发的软件、硬件和/或这样的软件和硬件组合的形式实现。

[0101] 在一个实施例中,多个交换机和多个端节点被布置在胖树拓扑中以用于在高性能计算(HPC)集群中使用。

[0102] 在一个实施例中,子网管理器还可以被配置为接收网络中的一个或多个叶交换机上的多个端节点中的每一个端节点的接收权重。

[0103] 在一个实施例中,选择至少一个下行端口包括比较多个下行端口和选择具有最小累计向下权重的下行端口。

[0104] 在一个实施例中,选择至少一个下行端口包括比较多个下行端口和选择具有最小累计向上权重的下行端口。

[0105] 在一个实施例中,选择至少一个下行端口包括比较多个下行端口和选择具有最小全局唯一标识符的下行端口。

[0106] 对本领域技术人员清晰的是,至少一些步骤也可以由各种硬件来实现,包括但不限于DSP、FPGA、ASIC等。例如,一些实施例中的“操作”可以由在CPU中运行的指令或者由实现“操作”的功能的诸如DSP、FPGA、ASIC之类的专用处理器来实现。

[0107] 如本领域普通技术人员将理解的,由框图表示的功能可以由软件和/或硬件来执

行。取决于特定处理策略,例如事件驱动、中断驱动等,可以以除图中所示的顺序或次序来执行各种功能。类似地,尽管未明确示出,但是可以重复地执行一个或多个步骤或功能。同样地,取决于特定实施方案,可以省略各种功能。本领域技术人员已知的各种功能可以不被明确地示出或描述,而是由所示的框或模块暗示。在一个实施例中,所示的功能主要由存储在计算机可读存储介质中并由基于微处理器的控制器执行以控制系统的操作的软件、指令或代码实现的控制逻辑执行。尽管相对于磁带驱动器一般地示出和描述,但是本领域普通技术人员将认识到,各种功能可以应用于各种其他类型的外围存储设备。

[0108] 可以使用一个或多个常规通用或专用数字计算机、计算设备、机器或微处理器(包括根据本公开的教导编程的一个或多个处理器、存储器和/或计算机可读存储介质)来方便地实现本发明。正如软件领域的技术人员将清晰的,基于本公开的教导,熟练的程序员可以容易地准备适当的软件编码。

[0109] 在一些实施例中,本发明包括计算机程序产品,其是具有存储在其上/中的指令的非暂态存储介质或计算机可读介质,其可以被用于对计算机进行编程以执行本发明的任何过程。存储介质可以包括但不限于任何类型的盘,包括软盘、光盘、DVD、CD-ROM、微型驱动器和磁光盘、ROM、RAM、EPROM、EEPROM、DRAM、VRAM、闪存存储器装置、磁卡或光卡、纳米系统(包括分子存储器IC)或适于存储指令和/或数据的任何类型的媒体或装置。

[0110] 为了说明和描述的目的提供了本发明的前述描述。它并不旨在穷举或将本发明限制为所公开的精确形式。许多修改和变化对于本领域技术人员将是清晰的。选择和描述实施例是为了最好地解释本发明的原理及其实际应用,从而使得本领域的其他技术人员能够理解本发明的各种实施例以及具有适合于预期的特定用途的各种修改。本发明的范围旨在由所附权利要求及其等同物限定。

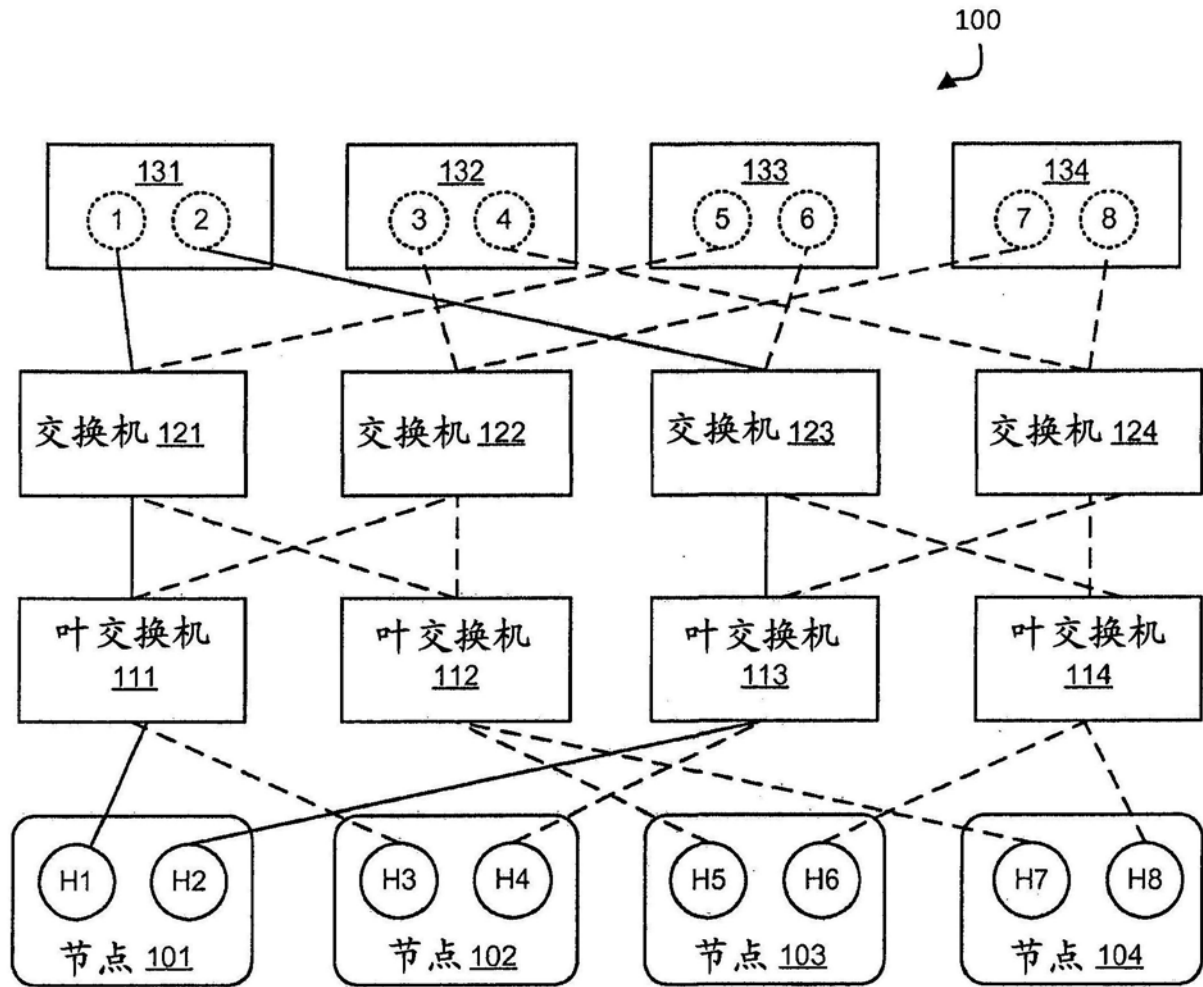


图1

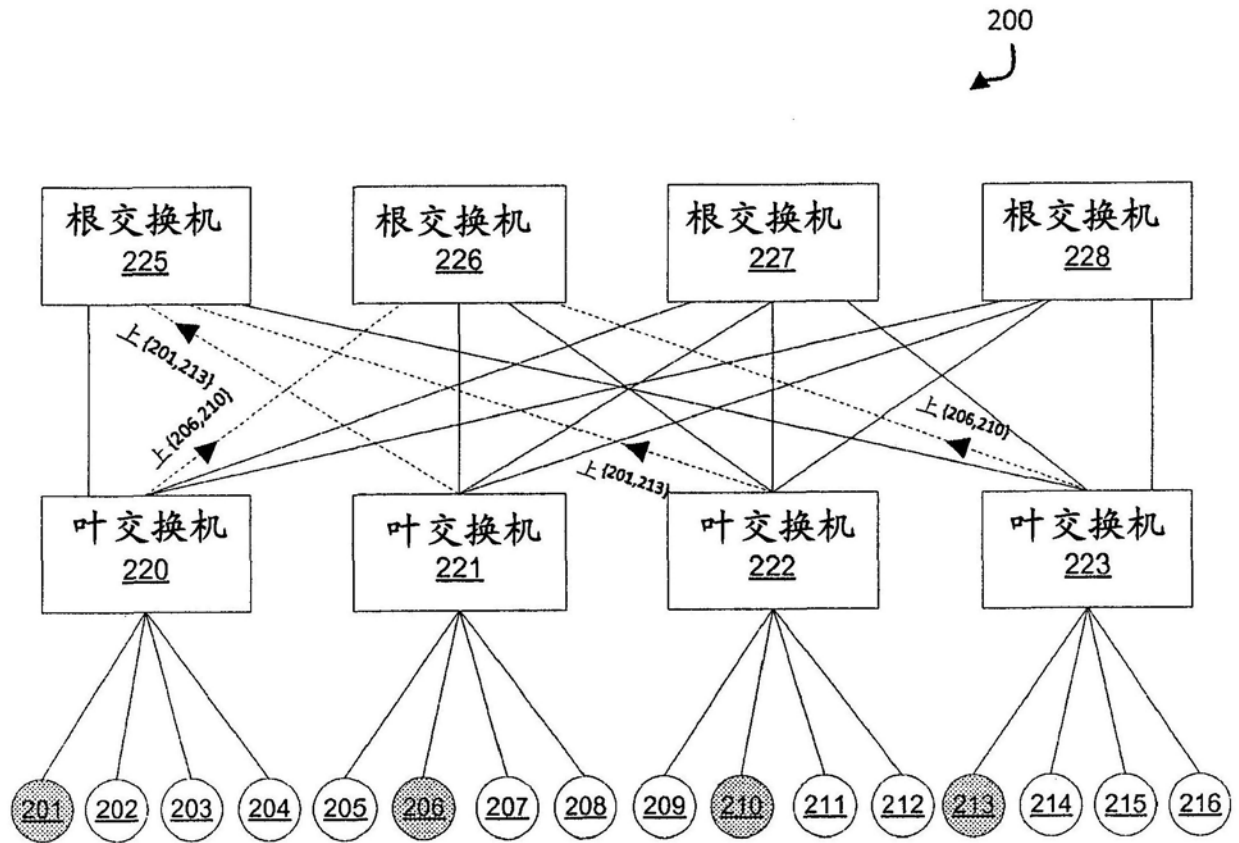


图2

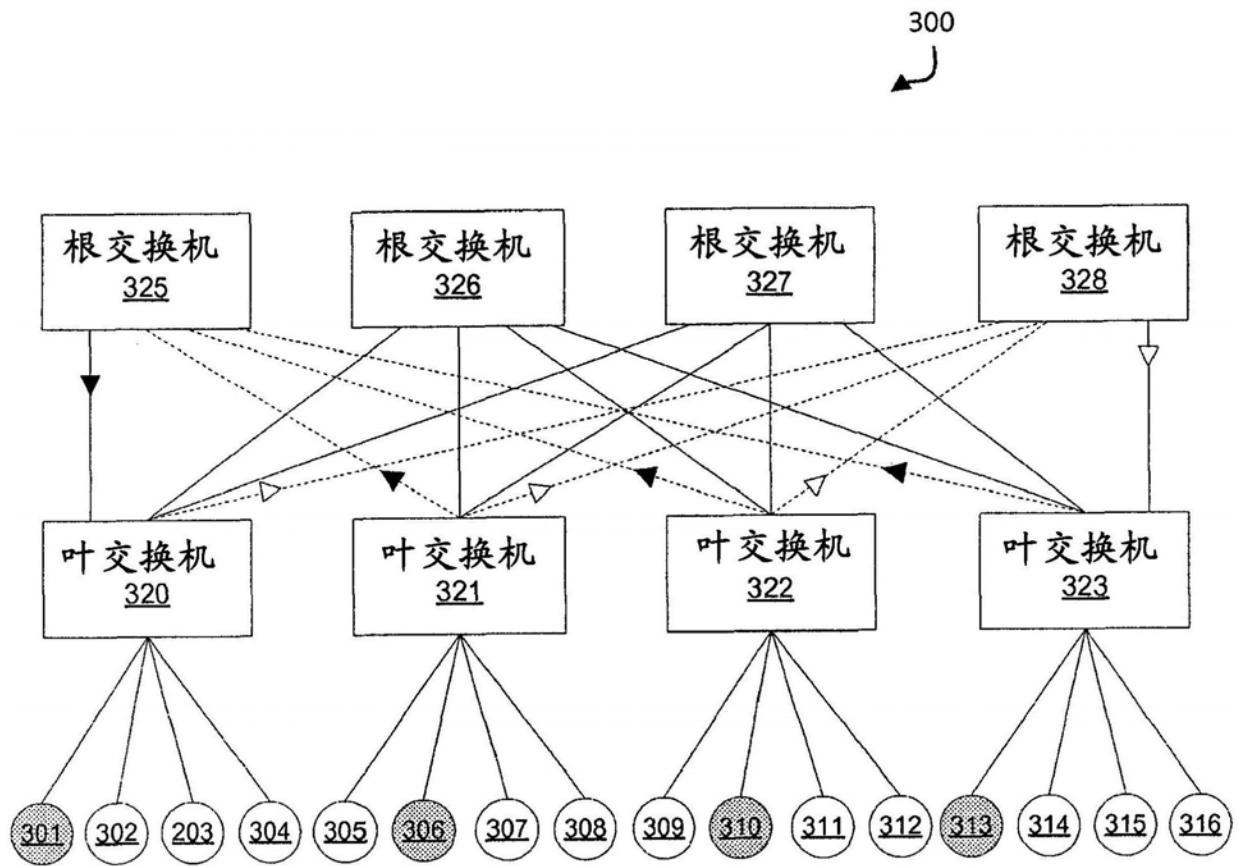


图3

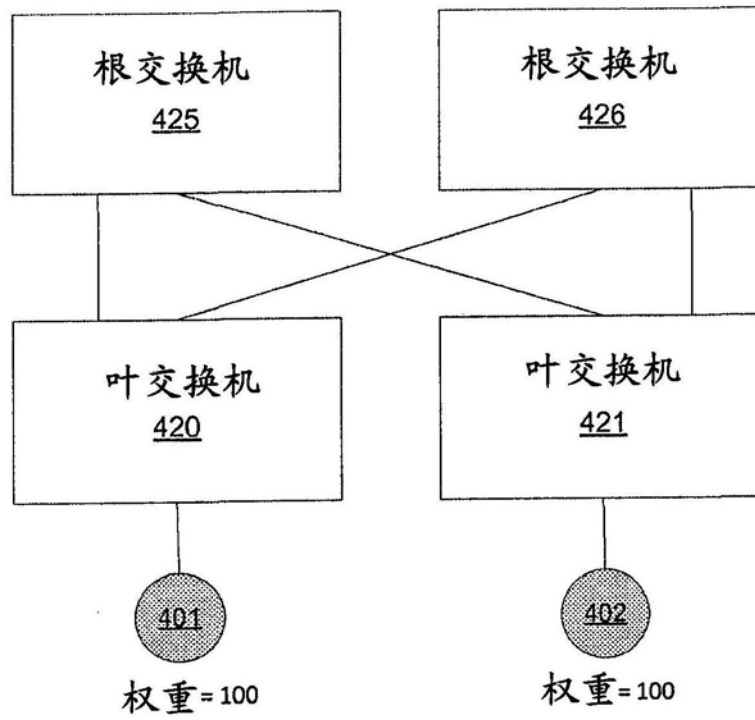


图4

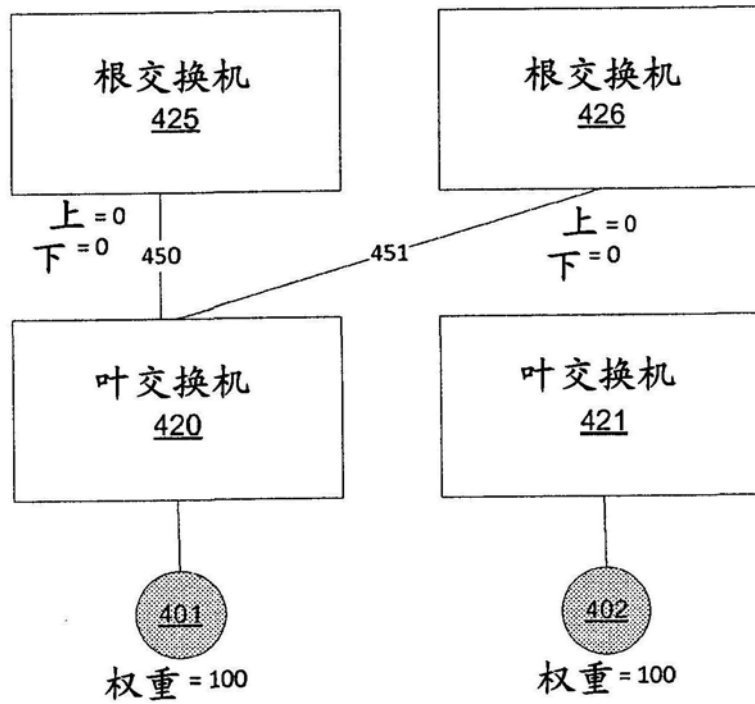


图5

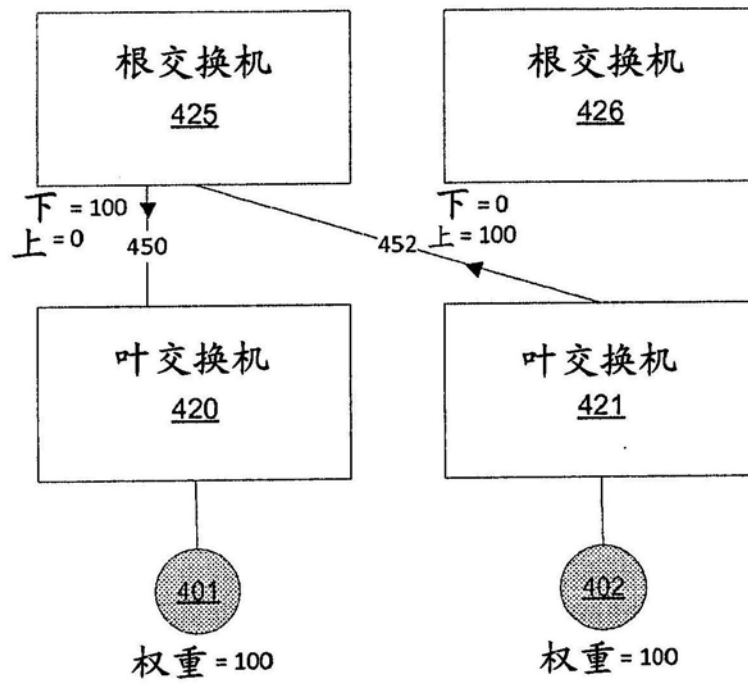


图6

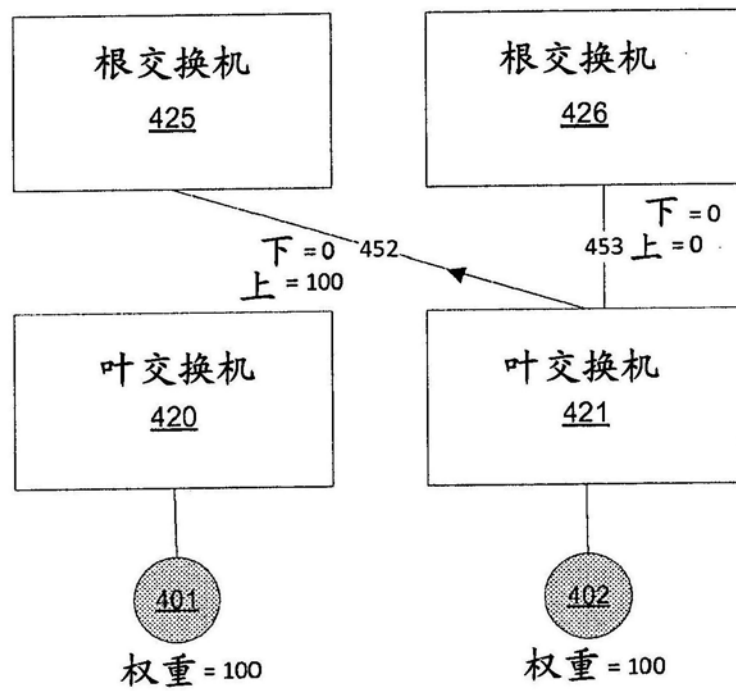


图7

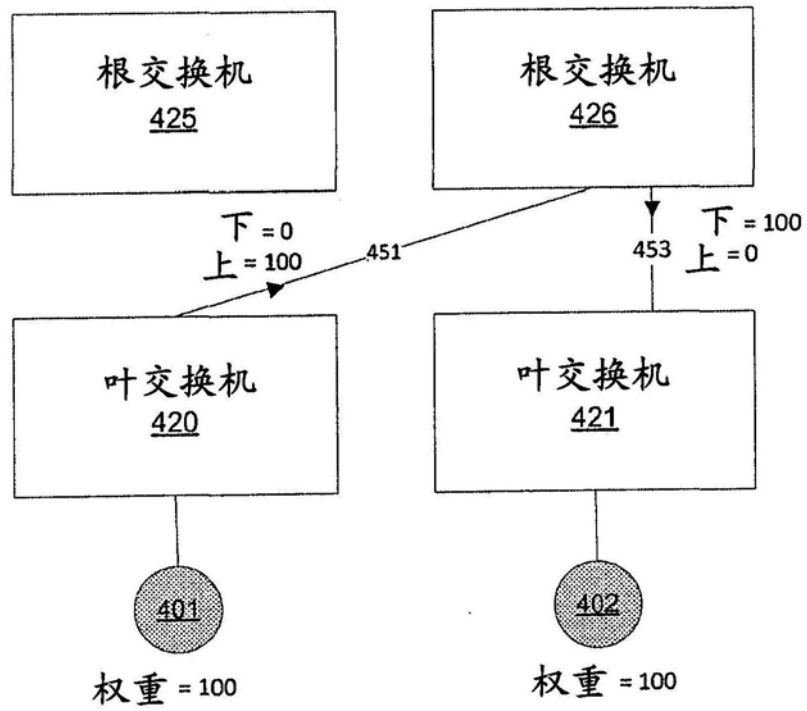


图8

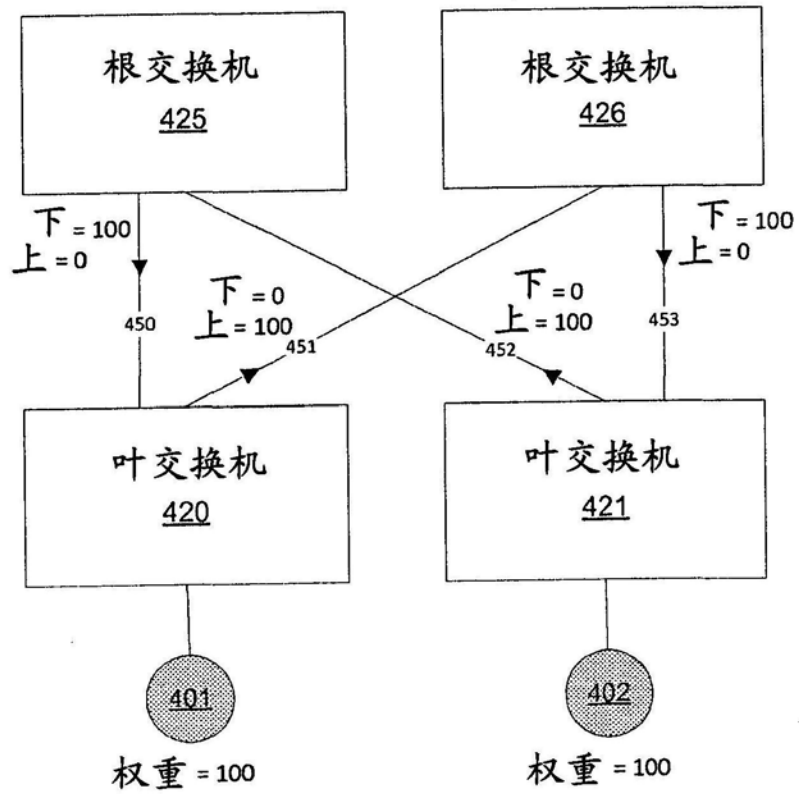


图9

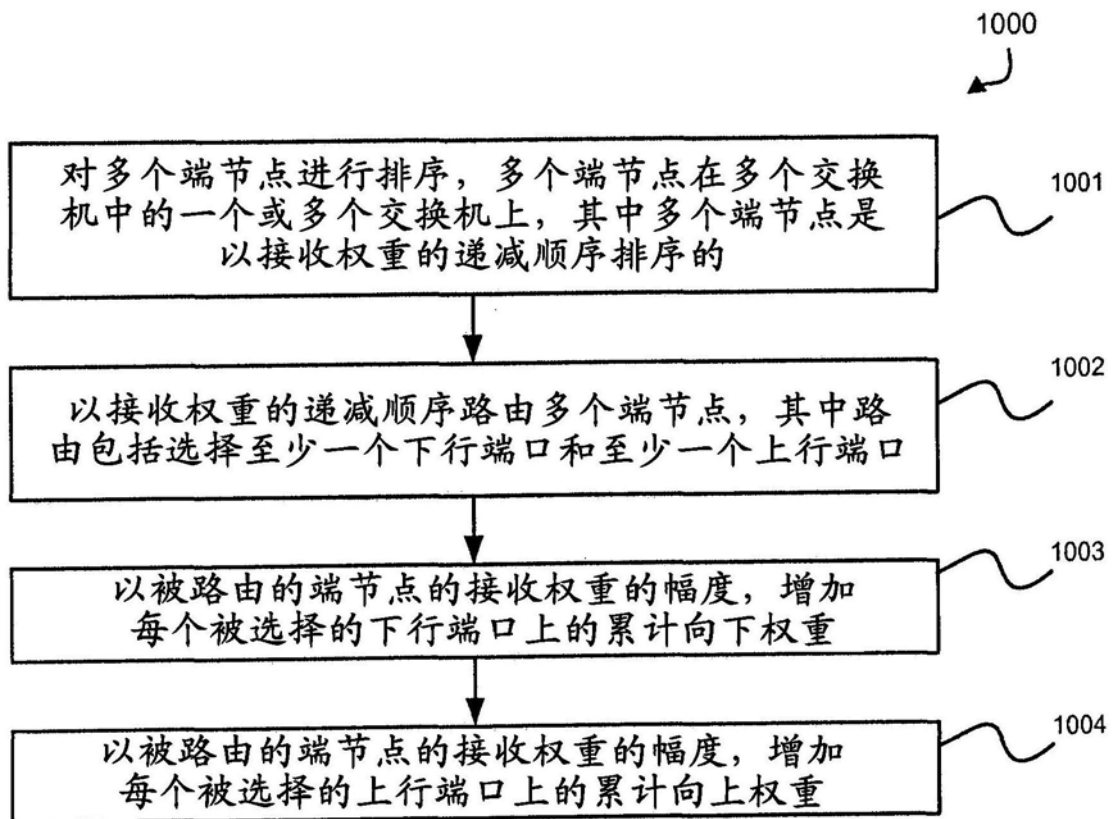


图10