基于离散粒子群算法的数据中心网络流量调度研究

林智华[1,2],高 文[2],吴春明[2],李勇燕[3]
(1. 福建江夏学院电子信息科学学院,福建福州 350108;2. 浙江大学计算机学院,浙江杭州 310027;
3. 绍兴文理学院上虞分院,浙江上虞 312300)

摘 要: 数据中心网络利用多个并行路径为集群计算等网络服务提供高对分带宽.然而,现有的流量调度算法可能会引起链路负载不均衡,核心交换机冲突加剧,造成网络总体性能降低.本文将流调度问题转化成0-*K*背包问题求解,提出基于离散粒子群的流调度算法DPSOFS(Discrete Particle Swarm Optimization Flow Scheduling).该算法根据Fat-Tree结构特点定义了粒子速度、位置和运算规则,以两次迭代冲突流个数差值作为目标函数,并限定路径搜索范围,减少随机搜索的盲目性.仿真实验验证了该算法对减少流冲突快速有效,能提高网络对分带宽.

关键词: Fat-Tree; 数据中心网络; 离散粒子群; 流调度
中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112 (2016)09-2197-06
电子学报URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2016.09.026

Data Center Network Flow Scheduling Based on DPSO Algorithm

LIN Zhi-hua[1,2], GAO Wen[2], WU Chun-ming[2], LI Yong-yan[3]
(1. *College of Electronics Information Engineering, Fujian Jiangxia University, Fuzhou, Fujian* 350108, *China*;
2. *College of Computer Science, Zhejiang University, Hangzhou, Zhejiang* 310027, *China*;
3. *University of Shaoxing at Shangyu, Shangyu, Zhejiang* 312300, *China*)

Abstract: Data center networks leverage multiple parallel paths connecting end host pairs to offer high bisection bandwidth forcluster computing applications. However, state of the art flow scheduling algorithms may cause unfair link utilization and saturation of core switches, resulting in overall bandwidth loss. In the paper, we regard the flow scheduling problem as a 0-*K* knapsack problem and propose a new flow scheduling algorithm named DPSOFS based on DPSO. DPSOFS formulates the position, velocity and their operation rules of particles according to Fat-Tree topology structure, and defines objective function as the difference of the number of conflict flows between two iterations. Moreover, our proposed mechanism reduces random search blindness by limiting the range of the path search. The simulation suggests that it can improve overall network bisection efficiently.

Key words: Fat-Tree; data center network; DPSO; flow scheduling

1 引言

随着数据量指数级的增长和各种网络服务不断涌现,网络拓扑结构对现代云计算数据中心的扩展性、成本、容错、整体性能等方面产生重大影响[1].目前新型的数据中心网络拓扑结构主要有两类[2],一类是以交换机为中心的结构,如Fat-Tree[3],PortLand[4],SPAIN[5]等;另一类是以服务器为中心的结构,如DCell[6],BCube[7],Hyper-BCube[8],等.其中,Fat-Tree拓扑结构相对简单,容易部署,同时支持多条等价路径和完全的对分带宽[9],是新型的数据中心网络中最受重视的体系结构之一.

研究表明,网络流量由大部分短暂流和小部分长命流组成,长命流占据了大部分带宽.数据中心网络中流量也是由很多小型的、事务方式的RPC(Remote Procedure Call)流和小部分大流组成.这些大流在网络中存在的时间较长,对应的数据包数或字节数相对较多[10],占据大量的链路带宽.在Fat-Tree等树型拓扑结

收稿日期:2014-11-10;修回日期:2015-03-06;责任编辑:梅志强
基金项目:973计划(No. 2012CB315903);浙江省重点科技创新团队(No. 2011R50010-21);国家科技支撑计划(No. 2014BAH24F01);国家自然科学基金(No. 61379118)

构中(图1),主机之间存在多条等价的路径.因此,针对网络拓扑结构特点研究并设计大流调度算法,减少大流碰撞,保证链路负载均衡,对于提高网络传输性能至关重要.

数据中心网络流调度算法主要分成静态和动态两类,ECMP(Equal-CostMultipathRouting)[11],VLB(Valiant Load-Balancing)[12]等静态调度法会造成某条链路过载.动态调度算法中,GFF[11]对边界交换机收到的流在全网线性查找无冲突路径,实现局部优化.Hedera[11]采用集中控制调度策略,在全网视图下通过模拟退火算法得到流优化路径.DDFS[13]和Flowlet[14]通过分布式方式控制汇聚层交换机链路负载率来实现网络负载均衡,DiFS[15]通过交换机间控制命令来减少流量冲突,从而增加网络带宽利用率.

未来数据中心的网络设备将实现数据面和控制面的分离,在控制面上可以进行全局流量集中管控.本文针对Fat-Tree数据中心网络拓扑结构特点,借鉴离散粒子群DPSO思想,提出流调度算法DPSOFS,将流调度问题转化成0-K背包问题求解,在数据中心网络全局高度上进行流路由安排,得到最优网络对分带宽.实验结果表明,该算法使网络对分带宽较GFF和Hedera算法提高了20%以上,冲突流数量减少40%左右,网络性能得到明显改善.

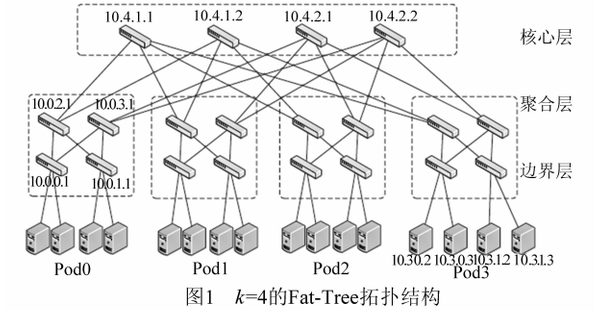


图1 k=4的Fat-Tree拓扑结构

2 相关工作

总体来讲,当前数据中心网络的流调度算法分成静态和动态两种,动态算法中又划分成集中式调度和分布式调度两种方式.

ECMP通过计算收到的数据包头的散列值所对应的等价路径的端口地址来转发数据流.但由于静态调度的特性,该算法在转发过程不考虑数据包或数据流大小,当某一主机流量较大时,依然会映射到同一转发端口,造成该端口负载增加,使得多条等价链路并不平衡.

VLB随机选择可用的转发端口实现负载均衡,但是依然存在多条数据流选择同一端口,导致流冲突和链路过载.

GFF流调度算法构建了中心调度器,对边界交换机收到的流线性查找一条无冲突路径.但是由于该算法在流分配上的先后次序不同,不能保证整个网络获得较优的对分带宽.

Hedera在全局网络视图下通过模拟退火算法得到流优化路径,使网络对分带宽有了一定的提高.但是它在路径选择上完全随机,在主机数量较大的情况下,核心交换机上可能产生流量拥塞,优化效果不明显.

DDFS分布式调度定义了链路负载率,通过控制聚合层交换机链路负载率来防止核心层交换机过载,实现负载均衡.Flowlet在DDFS的基础上利用Flow splitting技术,按不同链路负载率之间的比例关系,动态分割数据流到这些链路上,达到充分利用空闲链路带宽的目的.但这两种算法没有验证数据中心总体网络性能和吞吐量的改进.

DiFS算法在交换机上增加Explicit Adaption Request控制命令,通过该命令通告发送端远程流冲突,从而使发送端调整数据传输路径来避免冲突发生.但在数据中心网络较大的情况下,每台交换机分布式独立处理EAR控制命令,可能会增加通信开销,降低整体网络性能.

3 DPSOFS流调度算法

3.1 流调度问题数学模型

背包问题(Knapsack problem)是一种组合优化的NP完全问题,其相似问题经常出现在商业、组合数学,密码学和应用数学等领域中.由于背包问题与流调度问题本质是一致的,因此本文把该问题转换成0-K背包这种经典的问题进行求解.

在Fat-Tree拓扑结构中,主机之间有条路径.为某条流指定核心交换机的编号值,则唯一确定了主机间的通信路径.因此,流调度问题可以描述成一个n维背包问题:已知n条流$f_1,f_2,\cdots,f_n$,以及对应的流带宽属性值$b_i,(i=1,2,\cdots,n),b_i\in(0,1]$,$m$条链路$l_1,l_2,\cdots,l_m$,每条链路带宽上限值为1.用一个向量$\boldsymbol{c}_i(i=1,2,\cdots,n)$表示$n$条流对应的核心交换机的编号,$\boldsymbol{c}_i=(0,1,\cdots,(K/2)^2)$.当$\boldsymbol{c}_i$取值为0表示选择任何核心交换机都会造成流冲突,而非0值表示选择某一台核心交换机可行.另外,我们定义$\mathrm{map}(f_i,b_i,\boldsymbol{c}_i)$为映射函数,表示把流通过$\boldsymbol{c}_i$选择的路径映射到对应的若干条链路上,用$i\times j$的矩阵$\boldsymbol{a}$表示映射结果,对应的值是流$f_i$占用链路$l_j$的带宽$b_i$.该问题求解的目的是如何为所有的流$f_i$选择合适的$\boldsymbol{c}_i$,在保证所有链路$l_1,l_2,\cdots,l_m$,的带宽上限值的前提下,使$\boldsymbol{c}_i$取值为0的个数最少.

严格的数学描述为:

$$\begin{cases}\text{Min} \quad f(c_1,c_2,\ldots,c_n) = \sum_{i=0}^{n} \overline{c_i} \\ \text{s.t.} \quad \sum_{i=0}^{n} a_{ij} \leqslant 1, j = (1,2,\ldots,m)\end{cases} \tag{1}$$

式(1)中 $\text{Min} f(c_1,c_2,\cdots,c_n)$ 为目标函数,求所有 $\boldsymbol{c}_i=0$ 的个数总和最小值. 约束条件为占用任意 l_j 的流的带宽之和 $\boldsymbol{a}_{ij}$ 应限制在 1 以内.

3.2　流量带宽需求预测算法

对数据中心网络流量调度之前,必须先对每条大流的带宽需求 b_i 进行预测. 但流实际占用的带宽值无法反映其真实带宽需求. 另外,流的大小、长度以及字节数都是未知变量,即使事先已知其流量大小,对其进行带宽需求的预测也是 NP 难的优化问题. 考虑到任何流的传输带宽受限于发送端和接收端主机网卡的带宽,利用主机网卡带宽的限制来预测流的最大实际可用带宽是可行的. 流量带宽预测算法描述如下:

输入:$n\times n$ 矩阵 $\boldsymbol{I}$(元素值 I_{ij} 表示主机 i 向主机 j 发出大流数量),S_i 表示发送端主机 i 带宽限值,R_j 表示接收端主机 j 带宽限值,带宽初始限值为 1.

输出:$n\times n$ 矩阵 $\boldsymbol{O}$(元素值 O_{ij} 表示主机 i 向主机 j 发出流带宽需求预测值).

矩阵变量 $\boldsymbol{T}$,其元素值 T_{ij} 是由于受限于发送端的带宽,计算得到发送端 i 向接收端 j 发送流的预测带宽值.

矩阵变量 $\boldsymbol{A}$,其元素值 A_{ij} 是由于受限于接收端的带宽,计算得到发送端 i 向接收端 j 发送流的预测带宽值.

步骤 1:计算 $T_{ij}=S_i/\sum_{j=1}^{n} I_{ij}$;

步骤 2:计算 $A_{ij}=R_j/\sum_{i=1}^{n} I_{ij}$;

步骤 3:若存在任一元素 $T_{ij}\neq A_{ij}$ 则 $O_{ij}=\text{Min}(T_{ij},A_{ij})$;根据已确定的 O_{ij},更新矩阵 $\boldsymbol{I}$ 和剩余 S_i 值和 R_j 值;返回步骤 1;

步骤 4:若所有 $T_{ij}=A_{ij}$ 则 $O_{ij}=T_{ij}$;

步骤 5:返回矩阵 $\boldsymbol{O}$;

3.3　DPSOFS 算法设计

3.3.1　DPSOFS 算法模型

粒子群算法 PSO 是一种受到鸟群飞行集群活动的规律启发的群智能算法. 在 PSO 算法中,每一个粒子被认为问题的一个可行解,粒子以一定的速度在解空间运动,并向自身的历史最佳位置和全局最佳位置聚集,实现对候选解的进化[16],最终找到最优解. 与其他进化算法相比,该算法收敛速度更快,更加简单,易于实现并具有更强的全局优化能力. 近年来为了应对实际工程应用中的离散问题,研究者在原有算法基础上进行二进制空间的扩展,形成了离散粒子群算法 DPSO.

在 DPSOFS 算法模型中,我们用 $\boldsymbol{X}_i^k$ 表示粒子群中的第 i 个粒子在进化的第 k 代的位置值. 每一个 $\boldsymbol{X}_i^k$ 表示流调度问题中 f_i 对应的 $\boldsymbol{c}_i$ 的一个解向量,其中,$\boldsymbol{X}_i^k=[X_{i1}^k,X_{i2}^k,\cdots,X_{in}^k]$,$n$ 是粒子群中粒子具有的维数,即有 n 条流选择对应的 c_i 值. $\boldsymbol{V}_i^k$ 表示粒子群中的第 i 个粒子在进化的第 k 代的速度值. 粒子 i 在经过 k 代运算后得到的最优值为 $\boldsymbol{Pb}_i^k$,则

$$\boldsymbol{Pb}_i^k=\begin{cases}\boldsymbol{Pb}_i^{k-1}, & f(\boldsymbol{X}_i^k)>\boldsymbol{Pb}_i^{k-1} \\ \boldsymbol{X}_i^k, & \text{else}\end{cases} \tag{2}$$

整个群体得到的全局最优值记为 $\boldsymbol{Gb}^k$,则

$$\boldsymbol{Gb}^k=\text{Min}(\boldsymbol{Pb}_i^k) \tag{3}$$

每个粒子的速度更新公式为:

$$\boldsymbol{V}_i^{k+1}=\omega(f(\boldsymbol{X}_i^k)-f(\boldsymbol{X}_i^{k-1})) \tag{4}$$

位置更新公式为:

$$\boldsymbol{X}_i^{k+1}=\begin{cases}\boldsymbol{X}_i^k, & r1\leqslant\text{sig}(\boldsymbol{V}_i^{k+1}) \\ \boldsymbol{Pb}_i^{k-1}, & r1>\text{sig}(\boldsymbol{V}_i^{k+1})\text{ and }r2\leqslant c \\ \boldsymbol{Gb}^k, & r1>\text{sig}(\boldsymbol{V}_i^{k+1})\text{ and }r2>c\end{cases} \tag{5}$$

以上公式中,$f(\boldsymbol{X}_i^k)$ 函数计算第 i 个粒子进化到 k 代,位置值为 $\boldsymbol{X}_i^k$ 的冲突流个数. 式(4)中 ω 为调节系数,把冲突流差值限制在一定范围内. 当差值变化较大时,ω 取值较小. 式(5)中 $r1$ 和 $r2$ 为两个独立的随机数,c 为一非负常数,其值是 $\boldsymbol{X}_i^{k+1}$ 选择局部最优解和全局最优解的判断依据. sig 函数定义为:

$$\text{sig}(\boldsymbol{V}_i^{k+1})=1/(1+\exp(\boldsymbol{V}_i^{k+1})) \tag{6}$$

该函数把 $\boldsymbol{V}_i^{k+1}$ 值的大小映射到(0,1)的概率范围内. 当 $\boldsymbol{V}_i^{k+1}<0$ 时,则 $\boldsymbol{X}_i^{k+1}$ 以大概率选取原来的 $\boldsymbol{X}_i^k$ 值. 当 $\text{sig}(\boldsymbol{V}_i^{k+1})$ 趋向 0 时,$\boldsymbol{X}_i^{k+1}$ 则根据 $r2$ 与常数 c 的关系,按一定概率比例选择 $\boldsymbol{Pb}_i^{k-1}$ 或 $\boldsymbol{Gb}^k$,作为下一步的迭代的初始值.

3.3.2　$f(\boldsymbol{X}_i^k)$ 函数算法实现描述

$f(\boldsymbol{X}_i^k)$ 函数把 GFF 流调度算法的结果作为 $f(\boldsymbol{X}_i^k)$ 函数初始值,即 $\boldsymbol{X}_i^0=\boldsymbol{c}_i$.

$f(\boldsymbol{X}_i^k)$ 函数对每个 $\boldsymbol{c}_i=\boldsymbol{0}$ 的流在 $[1,2,\cdots,(k/2)^2]$ 范围内随机选择一个值,通过 $\text{map}(f_i,b_i,\boldsymbol{c}_i)$ 函数将该流写入矩阵 $\boldsymbol{a}$ 中,并用 $\text{replace}(b_i,\boldsymbol{c}_i)$ 函数根据流带宽值替换出与原有已安排好的流,以保证新插入的流在每条链路带宽上限值小于 1. replace 函数返回结果为替换出的流集合 R. 对该集合中的流采用 $\text{GFF}(f_i)$ 为其搜索其他可行路径,对无法分配的流同样置对应 $\boldsymbol{c}_i$ 为 $\boldsymbol{0}$. 当一次迭代结束后,统计 $\boldsymbol{c}_i=\boldsymbol{0}$ 的个数,并返回结果.

$f(\boldsymbol{X}_i^k)$ 函数算法步骤如下:

输入:由 GFF 流调度算法得到的 $\boldsymbol{c}_i(i=1,2,\cdots,n)$;

输出:算法运算后的 $\boldsymbol{c}_i(i=1,2,\cdots,n)$ 结果以及 $\boldsymbol{c}_i=\boldsymbol{0}$ 的个数值;

步骤1:当 $\boldsymbol{c}_i=\boldsymbol{0}$ 且 $i<n$ 时,执行步骤2~5;

步骤2:对每一条冲突的流随机分配一台核心交换机,即 $\boldsymbol{c}_i=\mathrm{Rand}(1,\cdots,(k/2)^2)$;

步骤3:将步骤2中的流对应的链路写入矩阵 $\boldsymbol{a}$ 中,即 $\mathrm{map}(f_i,b_i,\boldsymbol{c}_i)$;

步骤4:替换出原先已安排好的流,保证步骤2中的冲突流符合带宽要求,$R=\mathrm{replace}(b_i,\boldsymbol{c}_i)$;

步骤5:对 R 集合中的流 f 执行 $\mathrm{GFF}(f)$;

步骤6:计算 $\boldsymbol{c}_i=\boldsymbol{0}$ 的个数值并返回结果.

在Fat-Tree数据中心网络的 k 值较大($k>=16$),大流数较多时,采用随机产生核心交换机编号的方式会导致replace函数产生的替换流集合数过大,收敛效果不佳.为了加快收敛速度,我们对随机产生的 $\boldsymbol{c}_i$ 值限定范围.由于流从源主机到目的主机需经过4段可能产生冲突的链路,通过定义冲突链路阈值 t,使冲突流 f_i 随机选择的路径造成链路冲突的个数小于 t,即 $\{\sum_{j=0}^{m}|\boldsymbol{a}_{Iij}|\leqslant t\,|\,\sum_{i=0}^{m}\boldsymbol{a}_{ij}>1\}$,replace函数替换出的流集合数较少,函数返回的值较小,收敛加快,可取得较好的流调度效果.

3.3.3 $f(\boldsymbol{X}_i^k)$ 函数算法复杂度分析

$f(\boldsymbol{X}_i^k)$ 函数涉及到map函数,replace函数和GFF函数.map根据随机产生的核心交换机的值找到对应的链路,时间复杂度为 $O(1)$.replace函数根据对应的链路替换出链路上冲突的流,其个数小于某常数.GFF函数线性搜索空闲路径,路径最大值为核心交换机数量,其时间复杂度为 $O((k/2)^2)$.因此 $f(\boldsymbol{X}_i^k)$ 函数算法的时间复杂度为 $O(|f|*(k/2)^2)$,其中 $|f|$ 为冲突流的个数.另外,该函数调用map函数时需要维护网络中所有链路状态表,Fat-Tree数据中心网络双向链路总个数为 $2\times k\times(k/2)\times k=k^3$(不需要维护主机到边界交换机的链路),因此该函数的空间复杂度为 $O(k^3)$.

3.3.4 DPSOFS 算法流程

DPSOFS流调度算法步骤如下:

输入:n 条流 $(f_1,f_2,\cdots,f_n)$;

输出:$\boldsymbol{Gb}^k$;

步骤1:设定最大迭代次数,粒子个数,获得流量带宽预测 b_i 值,GFF算法 $\boldsymbol{c}_i$ 值;

步骤2:$\boldsymbol{X}_i^0=\boldsymbol{c}_i$;$\boldsymbol{Pb}_i^k=\boldsymbol{X}_i^0$;$\boldsymbol{Gb}^k=\boldsymbol{Pb}_i^k$;并计算初始条件下 $\boldsymbol{c}_i=\boldsymbol{0}$ 值个数,作为 $f(\boldsymbol{X}_i^0)$ 的初始值;

步骤3:令 $k=1$;

步骤4:对每个粒子分别并行 $f(\boldsymbol{X}_i^k)$ 值;

步骤5:如果粒子 i 的适应度目标函数 $f(\boldsymbol{X}_i^k)$ 值优于 $\boldsymbol{Pb}_i^k$ 和 $\boldsymbol{Gb}^k$,则根据公式(2)和公式(3)更新其值;

步骤6:若 $f(\boldsymbol{X}_i^k)=0$,说明所有流无任何冲突,输出 $\boldsymbol{X}_i^k$,算法结束;

步骤7:根据粒子速度更新公式(4)和位置更新公式(5)分别计算 $\boldsymbol{V}_i^{k+1}$ 和 $\boldsymbol{X}_i^{k+1}$;

步骤8:$k=k+1$;

步骤9:若 k 达到最大迭代次数,则输出 $\boldsymbol{Gb}^k$,算法结束,否则返回步骤4.

4 实验仿真平台与结果

为了验证DPSOFS算法的效果,我们通过Mininet[17]实验平台来搭建Fat-Tree网络.另外,为了验证该算法收敛性能以及在大型数据中心网络的扩展性能,我们也模拟了算法在 k 值较大情况下的运行结果.

4.1 Mininet 仿真平台

在Mininet平台上,我们配置了 $k=8$-ary 的Fat-Tree数据中心结构,每台主机产生不同长度的流,长度服从指数分布,流产生的时间服从泊松分布.

每台主机根据某种通信模式来选取目的主机,并且每台主机只接收网络中某一台主机的流量数据.通信模式[3]有以下三种:间隔方式 Stride(x)、交错方式 Staggered(pEdge,pPod)和随机方式 Random.

调度算法每隔5秒查询所有边界交换机大流(流传输数据超过100KB)的统计值,并计算最优路径.在流调度算法中,我们设定DPSOFS迭代次数为10,粒子数为3.在与Hedera算法对比时,由于Hedera循环不考虑流数量,因此我们把Hedera循环次数设为 $10\times3\times$ 需调度的流数量.

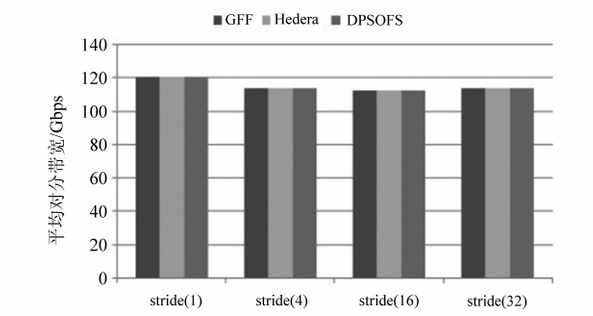
在实验中,我们使用对分带宽作为测试的性能指标.对分带宽越大,说明网络的传输性能越强.

4.2 实验结果

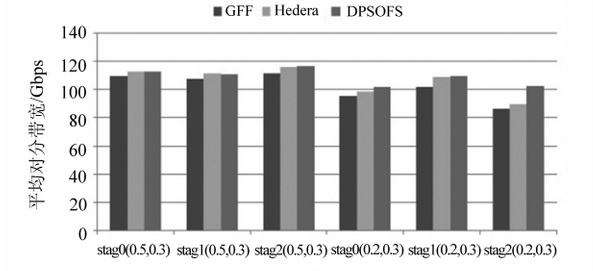
图2是GFF、Hedera和DPSOFS三种算法在 $k=8$ 的Fat-Tree拓扑下网络对分带宽测试的结果.我们对三种通信模式分别进行了多次测试,每次测试运行60秒,在第10秒到50秒这段时间内测量对分带宽.

在间隔模式下(一台主机与距离其位置 x 的其它主机通信),GFF总能找到一条无冲突的路径,得到最优的对分带宽值.这是由于Fat-Tree拓扑结构的特点,每台交换机上联和下联的带宽相同,即网络过载率为1:1.按该模式通信的主机可以并行无冲突地传输.

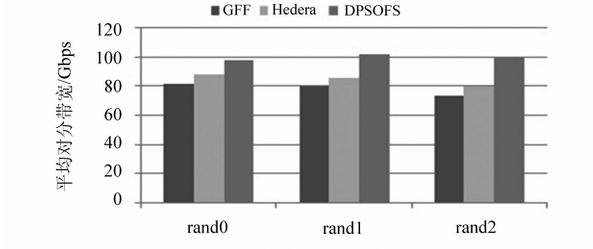
在stag(0.5,0.3)交错模式下,大多数流来自同一子网,这些流通过边界交换机直接进行全双工通信.由于不同Pod间通信的流数量较少,造成冲突的流也较少.三种算法得到的对分带宽相差不大.在stag(0.2,0.3)交错模式下,随着Pod间通信流数量增加,冲突流也增多,因此,GFF得到的对分带宽值有所降低.Hedera和DPSOFS在流数量较少的情况下都采用随机调度的方式,但是DPSOFS利用多粒子并行搜索,并且粒子间



(*a*) 三种算法在间隔模式下对分带宽的比较（k=8）



(*b*) 三种算法在交错模式下对分带宽的比较（k=8）



(*c*)三种算法在随机模式下对分带宽的比较（k=8）

图2　三种算法在不同模式下的比较

共享全局最优解,得到的结果好于 Hedera,网络对分带宽亦高于 Hedera.

在随机模式下,同一个子网以及同一个 pod 之间的通信流减少,大多数为不同 Pod 之间的数据流. 由于 GFF 调度先后次序造成较多的流无法找到无冲突路径. 因此其对分带宽较低. Hedera 对流的调度优化不太明显. 但是从图 2(*c*)中可以看出,冲突流越多,DPSOFS 多粒子并行运算获得的网络对分带宽就越优,全局调度效果就越好.

4.3　大型数据中心网络实验仿真结果

由于大型数据中心网络中的主机数量庞大,用 Mininet 无法实现仿真. 为了验证 DPSOFS 算法在收敛速度上的优势以及在大型数据中心网络流调度性能,我们模拟了调度过程,得到了调度后三种算法产生的冲突流个数,并进行了比较.

(1)三种算法的收敛效果

在 k = 16 的 Fat-Tree 拓扑结构下,我们测试了 GFF、Hedera 和 DPSOFS 在随机通信模式下迭代 20 次后获得的冲突流的个数.

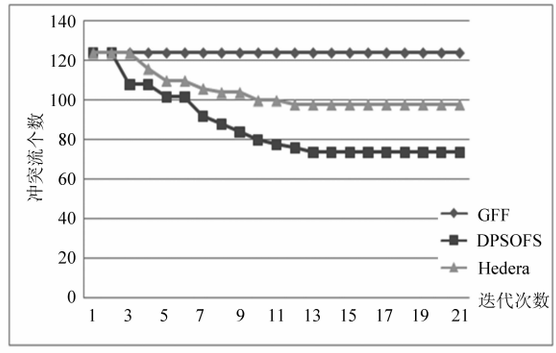


图3　三种算法在k=16的Fat-Tree拓扑结构下的收敛曲线

从图 3 可以看出,经过 GFF 调度后有 124 条流产生冲突. 而 DPSOFS 在第三次迭代后时就获得 Hedera 第五次迭代得到的较优结果(优化结果为 108). DPSOFS 在第七次迭代时就获得超出 Hedera 最终的优化结果(Hedera 优化结果为 92). 在迭代 11 次左右,DPSOFS 算法基本完成了收敛,获得全局较优的结果,体现了该算法较强的收敛特性和全局优化能力.

(2)三种算法在 k 值较大情况下的流调度效果

在算法测试过程中,我们选择 k = 16、24 和 32 的 Fat-Tree 拓扑结构. 测试结果为在随机模式下,三种算法得到的冲突流个数值和对应的占比关系,见表 1,表 2 和表 3.

表 1　三种算法在随机通信模式下的流冲突比较(k = 16)

随机次数	冲突个数			对 GFF 占比	
	GFF	Hedera	DPSOFS	Hedera/GFF	DPSOFS/GFF
Rand0	122	98	70	80.33%	57.38%
Rand1	116	102	72	87.93%	62.07%
Rand2	106	96	58	90.57%	54.72%

表 2　三种算法在随机通信模式下的流冲突比较(k = 24)

随机次数	冲突个数			对 GFF 占比	
	GFF	Hedera	DPSOFS	Hedera/GFF	DPSOFS/GFF
Rand0	312	298	220	95.51%	70.51%
Rand1	348	332	224	95.40%	64.37%
Rand2	348	326	242	93.68%	69.54%

表 3　三种算法在随机通信模式下的流冲突比较(k = 32)

随机次数	冲突个数			对 GFF 占比	
	GFF	Hedera	DPSOFS	Hedera/GFF	DPSOFS/GFF
Rand0	740	698	460	94.32%	62.16%
Rand1	750	738	502	98.40%	66.93%
Rand2	668	640	438	95.81%	65.57%

从表 1,2,3 中可以看到,DPSOFS 在 k 值较大的情况下比 Hedera 表现出更优的运算结果. 随着主机数的

增加,Hedera 优化效果不明显,在 $k=32$ 时,多次测试 Hedera/GFF 比值保持在 95% 左右,而 DPSOFS/GFF 比值基本都在 65% 左右,体现了 DPSOFS 较好的流调度效果. 这主要源于 DPSOFS 对随机选择核心交换机的改进上. Hedera 只是通过随机选择两条流并交换各自的核心交换机位置来判断两次结果的优劣;DPSOFS 一方面通过多粒子并行计算并共享全局最优解,另一方面充分利用空闲链路,避免选择与其他流出现链路冲突的路径,使得算法迭代次数减少,加快收敛,获得更优的调度结果.

5 结论

本文对当前常用的 Fat-Tree 数据中心网络流调度算法进行了分析,针对集中控制的两种动态调度算法 GFF 和 Hedera 进行研究和改进,提出 DPSOFS 算法,将流调度问题转化成 0-K 背包问题,以两次迭代冲突流个数差值作为目标调节函数,并限定路径搜索范围. 在仿真和验证过程中,该算法体现出更快的收敛速度和更优的调度结果.

参考文献

[1] Hammadi A, Mhamdi L. A survey on architectures and energy efficiency in data center networks[J]. Computer Communications, 2014, 40(1): 1 – 21.

[2] Zhang Y, Ansari N. On architecture design, congestion notification, TCP incast and power consumption in data centers [J]. Communications Surveys & Tutorials, IEEE, 2013, 15 (1): 39 – 64.

[3] Al-Fares M, Loukissas A, Vahdat A. A scalable, commodity data center network architecture [J]. ACM SIGCOMM Computer Communication Review, 2008, 38(4): 63 – 74.

[4] Mysore R N, Pamboris A, Farrington N, et al. PortLand: a scalable fault-tolerant layer 2 data center network fabric [J]. ACM SIGCOMM Computer Communication Review, 2009, 39(4): 39 – 50.

[5] Mudigonda J, Yalagandula P, Al-Fares M, et al. SPAIN: COTS data-center ethernet for multipathing over arbitrary topologies[A]. Proceedings of Networked System Design and Implementation[C]. San Jose: NSDI, 2010. 265 – 280.

[6] Guo C, Wu H, Tan K, et al. DCell: a scalable and fault-tolerant network structure for data centers [J]. ACM SIGCOMM Computer Communication Review, 2008, 38 (4): 75 – 86.

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

[8] Lin D, Liu Y, Hamdi M, et al. Hyper-bcube: A scalable data center network[A]. Proceedings of International Conferenceon Communications (ICC)[C]. Ottawa: IEEE, 2012. 2918 – 2923.

[9] Escudero-Sahuquillo J, Garcia P J, Quiles F J, et al. A new proposal to deal with congestion in InfiniBand-based fat-trees[J]. Journal of Parallel and Distributed Computing, 2014, 74(1): 1802 – 1819.

[10] Tso F P, Hamilton G, Weber R, et al. Longer is better: exploiting path diversity in data center networks[A]. Proceedings of Distributed Computing Systems[C]. Philadelphia: IEEE, 2013. 430 – 439.

[11] Al-Fares M, Radhakrishnan S, Raghavan B, et al. Hedera: dynamic flow scheduling for data center networks[A]. Proceedings of Networked System Design and Implementation[C]. San Jose: NSDI, 2010. 19 – 19.

[12] Zhang-Shen R, McKeown N. Designing a predictable internet backbone with valiant load-balancing[A]. Proceedings of Quality of Service – IWQoS [C]. Passau: Springer, 2005. 178 – 192.

[13] Bharti S, Pattanaik K K. Dynamic distributed flow scheduling with load balancing for data center networks[J]. Procedia Computer Science, 2013, 19(1): 124 – 130.

[14] Bharti S, Pattanaik K K. Dynamic distributed flow scheduling for effective link utilization in data center networks [J]. Journal of High Speed Networks, 2014, 20 (1): 1 – 10.

[15] Cui W, Qian C. Difs: Distributed flow scheduling for adaptive routing in hierarchical data center networks[J]. Measurement, 2014, 5(1): 17.

[16] 程祥,张忠宝. 基于粒子群优化的虚拟网络映射算法[J]. 电子学报,2011,39(10):2240 – 2244.
Cheng Xiang, Zhang Zhong-bao. Virutal network embedding based on particle swarm optimization[J]. Acta Electronica Sinica, 2011, 39(10): 240 – 2244. (in Chinese)

[17] Lantz B, Heller B, McKeown N. A network in a laptop: rapid prototyping for software-defined networks[A]. Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks[C]. New Delhi: ACM, 2010.

作者简介

林智华 男,1972 年 12 月生于福建福州,现为福建江夏学院电子信息科学学院副教授,主要研究方向为新一代网络技术.
E-mail: lindiva@126.com