一种基于P2P的并行传输模型

姜春茂[1,2] 张国印[1] 姚爱红[1]
(哈尔滨工程大学计算机科学与技术学院 哈尔滨 150001)[1]
(哈尔滨师范大学计算机科学技术与信息工程学院 哈尔滨 150025)[2]

摘 要 文件共享是P2P的重要应用之一,如何提高基于P2P的快速文件传输是保证用户满意度的重要技术。基于以往的研究,提出了一种数据分块与公平存储策略,基于存储策略给出了一个并行传输算法。其中存储策略较完全副本部署具有极大的空间优势,而并行传输算法不但可以适应网络的动态变化,而且对最后一个数据的传输时间也做了优化。实验结果表明:与理论分析一致,它较先前提出的模型具有更强的适应性和实际应用价值。

关键词 P2P,并行传输,数据存储模型
中图法分类号 TP311 **文献标识码** A

Parallel Transmission Model Based on P2P

JIANG Chun-mao[1,2] ZHANG Guo-yin[1] YAO Ai-hong[1]
(School of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China)[1]
(Schoo of Computer Science and Technology, Harbin Normal University, Harbin 150001, China)[2]

Abstract File sharing is one of the important application based on P2P, and how to improve the fast transmission based on P2P is important technology which is the guarantee of users' satisfaction. Based on the previous studies, this paper put forward a data fragment and fair storage strategy, then gave a parallel transmission algorithm. This stored strategy has more great space advantage than the entity copy. And this algorithm not only can adapt to the dynamic changes of the network, but also increase data transmission rate. The experiment result in dicates that it is in accordance with the theory and has a strong suitability and practical application value than previous model.

Keywords P2P(Peer to Peer), Parallel data transmission, Storage model

1 引言

文件共享是P2P的重要应用之一,如BT,将节点的共享文件做成种子发布到服务器中,当有节点需要请求服务的时候,想服务器请求,然后连接到资源所在节点,而且共享是分布式地存储在众多节点中的,资源的请求与下载都由服务器加以完成。因而,如何保证从多个节点同时下载的文件可靠、快速地实现与各个节点的负载均衡是非常值得研究的问题[1,2]。

众多学者对该问题进行了研究,2004年Sudharshan等人[3,4]开创性地提出了协同数据获取(Co-Allocation)策略,利用GridFTP的部分文件获取功能,同时从多个副本站点传输部分数据,而后在进行合并。2005年Chao等人[5]在分析了Sudharshan基于分块的调度策略后认为,由于每次的调度分配以固定大小的块为单位,则最后一次调度结束后,最终的传输完成时间将受到最慢节点的制约。2006年,Huvaneswaran等人[6,7]在其研究中同样分析了Sudharshan等提出的保守和激进法中最后一个传输块对整体完成时间的影响,并且也考虑到了网络的不确定性对调度分配的影响。他们给出了一个环形队列分配方式,在这种情况下某些块可能会被分配到多个副本节点,因此该算法具备一定故障防范能力,即如果某个副本节点停止了传输,由于其他节点也分配到了这个数据块,则可以进行重新传送。但是由于各节点存储的是完全副本,因此无论任何一个节点出现问题,也都可以从其他节点进行传送。

2007年与2010年,同样是Chao等人[8,9]基于先前提出的基本思想,进一步丰富完善,开发了一个并行传输系统,在其系统里综合考虑了副本节点的CPU、IO、和网络负载情况,给出了一个副本节点选择模型,但是对其先前提出的并行传输算法本身的性能没有改进。

上述研究都充分利用了并行传输的优势,保障了传输的速度,但是都有一定的不足,具体为:(1)为了提升聚集带宽和支持并行传输,对原始数据文件进行完整的多节点副本部署,这在存储空间和网络流量上开销过大;(2)由于各节点存储容量的限制,当数据文件过大时,直接将完整的副本部署于一个节点是不可行的,Feng在实验时就遇到了这个问题,而在其他人的实验中,较大的文件也都没有超过4G。

在数据文件较大时进行完整的副本部署具有一定难度,

到稿日期:2011-06-20 返修日期:2011-09-21 本文受国家自然科学基金(61073042),黑龙江省自然科学基金(F201139)资助。
姜春茂(1972-),男,博士生,副教授,主要研究方向为P2P、嵌入式、多核,E-mail:hsdrose@126.com;张国印(1962-),教授,博士生导师,主要研究方向为嵌入式、网络安全等;姚爱红(1972-),女,副教授。

因此在分布式文件系统研究社区中提出了分块存储的机制，如 Google 为迅速增长的数据处理需求，设计并实现了 Google 文件系统(GFS，Google FileSystem)。GFS 系统中的文件大小与通常文件系统中的文件大小概念不一样，文件大小通常以 G 字节计算。GFS 存储的文件都被分割成固定大小的 Chunk，而后存储到不同节点上[10]。Hadoop 分布式系统基础架构由 Apache 基金会开发，实现了一个分布式文件系统(HadoopDistributed File System)，简称 HDFS，其与 GFS 具有很大的相似性，都采用数据分块存储策略，有着高容错性的特点，并且设计用来部署在低廉的(low-cost)硬件上[10]。

2006 年，Ruay-Shiung 等人[9]提出了基于分块存储策略进行并行传输的算法，即首先假定数据文件的多个块随机地存储于分布式文件系统中的不同节点，获取这些块文件信息后，给出了调度算法，但是该算法基于随机分块副本存储(即分块存储没有考虑到网络因素)，因此会出现较快节点等待较慢节点导致整体传输时间过长的问题。

相似地，2008 年 Gaurav 等人[11]针对“多源-多目标”的数据获取情形，基于并行传输机制，给出了一个传输优化模型，该模型力争达到整体完成时间的最小化。在其研究中多源并不是指相同的数据副本，可以是同一个文件的不同数据块。但是该研究与 Ruay-Shiung 具有同样不足，即没有考虑数据块的存储过程(位置、数量等)，这同样会导致较快节点等待较慢节点导致整体传输时间过长的问题。

为了解决上述问题，提出了一种数据分块与公平存储策略，基于存储策略给出了一个并行传输算法。其中存储策略较完全副本部署具有极大的空间优势，而并行传输算法不但可以适应网络的动态变化，而且对最后一个数据的传输时间也做了优化。

2 数据分块与存储模型

令数据文件表示为 f，其大小为 S，适合对当前文件进行存储的 k 个节点分别为 $N1-Nk$。

(1)文件分块　首先将文件进行 k 等分，每一份子文件表示为 $f_i(1\leqslant i\leqslant k)$，$f_i$ 的大小表示为 S_i，显然 $S_i=S/k$。然后对 f_i 进行 $k-1$ 等分，得到的每一份子文件表示为 $f_{ij}(1\leqslant i\leqslant k,1\leqslant j\leqslant k-1)$，$f_{ij}$ 的大小表示为 $S_{ij}(1\leqslant i\leqslant k,1\leqslant j\leqslant k-1)$。此时文件 f 包含的总数据块数为 $k(k-1)$，分割结果如图 1 所示。

(2)文件存储　分两个步骤：(a)首先将 f_i 存储到节点 N_i，(b)将 f_i 包含的 $k-1$ 个文件 $f_{ij}(1\leqslant i\leqslant k,1\leqslant j\leqslant k-1)$ 分别存储到其他 $k-1$ 个节点，每个节点只存储一个数据块，存储时按照节点编号和 f_{ij} 文件的编号 j 以逐一递增顺序存储(见图 2)。按照此规则处理完所有的 $f_i(1\leqslant i\leqslant k)$ 及其包含的子文件 f_{ij}。得到的数据存储结构如图 2 所示，这里假设 $k>3$。

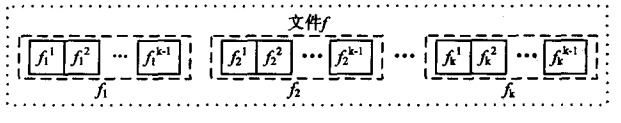

图 1　文件分块示意图

在图 2 中每个节上都存储了两种类型的数据：下层加粗的为步骤(a)分配的数据，这里称其为节点的 LND 数据；上层为步骤(b)分配的数据，这里称其为节点的 OND 数据。很显然 LND 数据量与 OND 数据量是相等的，因此存储的总数据量为 2S。

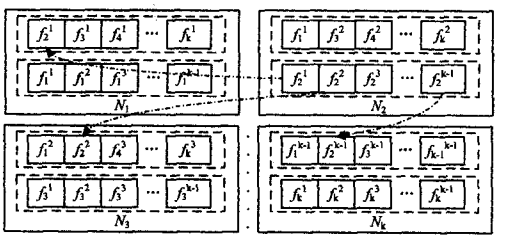

图 2　数据块存储示意图

3 负载调整算法

定义 1(本地数据剩余量，S_i)　节点 N_i 理想的下载数据量与本地数据 LNDi 的差值定义为本地数据剩余量，即 $S_i=V_i/(\sum_{j=0}^{k-1}V_j)-LND_i$。

该指标反映出节点下载任务的负载情况，S_i 趋于或均等于 0 时说明各节点可以大致同时完成下载。

如果 S_i 为正，说明该节点下载速度较快，可以分担 S_i 为负的节点的下载任务。

算法 1　负载调整算法

目标：优化各节点的 S_i，使其接近或趋于 0，达到各节点负载的均衡。

说明：对于任意两节点 ZN、FN，如果 ZN 的 S_i 为正，并且 FN 的 LND 数据中仍有没有被 ZN 的 OND 数据分担的部分，那么 ZN 可以分担 FN 的下载任务，每分担一块则 ZN. S_i－－，FN. S_i＋＋。

如果数据被分担，则对该数据块(包括存储在其他节点中的)进行下载位置标记，其他节点中的该数据块不再进行处理。

输入：k，p，V_i，满足存储模型的数据。

输出：从各节点下载数据块的调度方案。

(1) 构建 S_i 为正的节点链表 ZS_i_list，对 ZS_i_list 进行降序排列。

(2) 构建 S_i 为负的节点链表 FS_i_list，对 FS_i_list 进行升序排列。

(3) For every node FN in FS_i_list deal from first //依次处理 FS_i_list 中的每个节点

For every node ZN in ZS_i_list deal from first //让 ZS_i_list 中的节点分担 FS_i_list 中节点的下载任务

IF FN. $S_i\geqslant 0$ THEN FS_i_list. Move_to_Next，go to(3)

(4)IF ZN. $S_i>0$ THEN

令当前节点 ZN 的 OND 数据分担节点 FN 中 LND 的数据，并根据分担的块数更新 ZN. S_i，一次分担一块，并时刻判断以下两个条件：

(a) IF FN. $S_i\geqslant 0$ THEN FS_i_list. Move_to_Next，go to(3)

(b) IF ZN. $S_i\leqslant 0$ THEN ZS_i_list. Move_to_Next，go to(4)

(5) 在上述(1)－(4)的处理中，让速度较快的节点分担较慢节点的下载任务，只要两个链表有一个走到尾则跳出。

(6) 如果此时 FS_i_list 中仍然有 S_i 为负的节点 FN，那么从 ZS_i_list 头开始遍历，如果 ZS_i_list 中节点 ZN 的 OND 数据中没有被处理的数据块与 FN 中 LND 的没有被处理的数据块有交集，则令 ZN 分担 FN 的下载任务，直到没有交集为止，而不管 ZN 的 S_i 是否已经为负。

(7) 此时 FS_i_list 中的节点有两种情况，S_i 分别为正或非正。那么让 S_i 为正的节点的 OND 数据分担 S_i 为负的节点的 LND 数据，直到不能分但为止。

(8) 此时 ZS_i_list 中的节点存在两种情况，S_i 分别为正或非正。针对 ZS_i_list 中的节点，重复(1)－(4)步处理。完成后根据各节点数

据块的下载标记进行下载。

算法 1 的缺陷是它只能完成类似“历史法”的功能，即每次传输只进行一次调度，那么由于网络传输速度的动态变化，它必须进行多次调度来调整从各节点下载的数据块。

4 并行传输算法

客户端由下载单元和调度器单元组成，在每一个 Δt 结束时，下载单元将各节点最近一段时间的平均速度 V_i 以及各节点没有完成的的数据量 m_i 传递给调度器，调度器算出 x_i，计算输出应从各节点下载的数据块（B_i 中的各数据块），并实时观测是否有新 Δt 生成，循环处理，如图 3 所示。

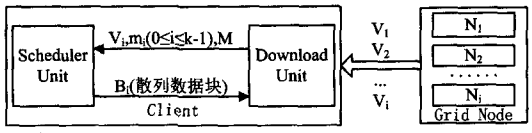

图 3 并行传输原理图

存储策略采用先将原始数据分割为若干块，对于每一块应用上述模型，假设数据块的数据量为 MM，分割的块数用 h 表示，每一块用 B_i 表示。由于任意一个满足存储模型的数据块 B_i 为非完整副本，假设从 4 个节点同时下载 B_i，那么必须满足：①不间断地从 4 个节点下载数据；②不进行重复数据块下载；③各节点下载完成后，从各节点下载的数据的并集等于 B_i。

由于各节点传输速度的动态变化，那么 Δt_i 就为最先完成 B_{i-1} 和 B_i 下载任务的节点对应的时刻之间的间隔。比如 N1 最先完成 B_{i-1} 下载时刻为 t_{i-1}，N3 先完成 B_i 下载时刻为 t_i，那么时间间隔 $\Delta t_i = t_{i-1}$。各节点完成数据 B_i 下载数量确定方式如下：①以 $t_{i-1}(0 \leqslant i \leqslant k-1)$ 时刻各节点速度 V_i 为调器输入速度；②在时刻 t_{i-1} 时各节点先前没有下载完的数据量为 m_i；③各节点承担数据 B_i 下载量为 $x_i(0 \leqslant i \leqslant k-1)$；④理论上必须满足式(1)，即尽量保证各节点同时完成下载任务。

$$\frac{V_0}{m_0+x_0}=\frac{V_1}{m_1+x_1}=\cdots=\frac{V_{k-1}}{m_{k-1}+x_{k-1}} \tag{1}$$

对式(1)应用等式恒等定理，推出式(2)，由式(2)推出 x_i 求解方法式(3)。以 x_i 为调度器 S_i 计算输入值。

$$\left(\frac{V_i}{m_i+x_i}\right)_{i=0}^{k-1}=\frac{\sum_{i=0}^{k-1}V_i}{\sum_{i=0}^{k-1}(m_i+x_i)}=\frac{\sum_{i=0}^{k-1}V_i}{\sum_{i=0}^{k-1}m_i+\sum_{i=0}^{k-1}x_i}=\frac{\sum_{i=0}^{k-1}V_i}{\sum_{i=0}^{k-1}m_i+M}=\lambda \tag{2}$$

$$x_i=\frac{V_i}{\lambda}-m_i(0 \leqslant i \leqslant k-1) \tag{3}$$

根据上述公式和约束，给出并行传输的原理图，如图 3 所示。客户端由下载单元和调度器单元组成，在每一个 Δt 结束时，下载单元将各节点最近一段时间的平均速度 V_i 以及各节点没有完成的数据量 m_i 传递给调度器，调度器算出 x_i，计算输出应从各节点下载的数据块（B_i 中的数据块），并传递给下载单元。下载单元根据调度输出结果从各节点下载 B_i 中的各数据块，并实时观测是否有新 Δt 生成，循环处理。

5 算法的测试

本文并行传输算法为 PTBM（Parallel Transmission Based on Model），令保守调度算法为 CLBA（Conservative Load Blancing Algorithm）。选择 4～6 个普通 PC 机节点安装 IIS 服务器，并按照存储模型部署数据，客户端为教育网内普通 PC。数据为“java14. wmv”，文件大小为 142M，设定存储模型 $p=1$。

定义 2（最小极限传输时间 MLTT：Minimum Limited Transmission Time） 如果总数据量为 M，节点数为 k，节点 i 的实时速度为 V_i，最小极限传输时间为 ζ，则有 $M=\sum_{i=1}^{k}(\int_0^{\zeta}V_i\,dv)$。

时间 ζ 的确定方法，从副本节点向客户端传输数据流，只记录各节点传输的数据量 m_i（不考虑任何的调度和文件合并问题），当 $\sum m_i = M$ 时，停止，即可得到 ζ。

实验：测试在相同个数的节点情况下，本文算法与其他算法性能的差异。

在以往的文献中，其主要贡献是缩短了最后一块数据的传输时间，但是当数据块为几十兆时，其性能与 Conservative 相差不大，而当数据块大小为几兆或十几兆时，该算法几乎没有多大优势。由于本文的存储模型可以控制数据块大小，因此本文直接与 Conservative 方法比较性能。令 $B_i=4$，数据文件“java14. wmv”经模型处理后分割为 48 块数据。每个数据块大小约为 2. 96M。对于 Conservative 并行传输，同样分成 48 块，与存储模型的划分方法相同。并且观测 MLTT 的时间，与其他两种算法性能作比对。

测试时间，分别为上午 9－10 点（分别观测 PTBM，CLBA，MLTT），下午 2－3 点（分别观测 PTBM，CLBA，MLTT），晚上 5－6 点（分别观测 PTBM，CLBA，MLTT）。由于在不同时段进行测试，考虑到网络的动态性，因此对每种实验观测 3 次（3 个时间段），取得的数值如表 1 所列。

表 1 传输时间

类别	PTBM			CLBA			MLTT		
次数	1	2	3	1	2	3	1	2	3
传输时间/(秒)	2306	2226	1765	2250	2259	1666	2180	2152	1580

对表中的数据进行处理，形成图 4，从中可以看到，在同一时间观测的 PTBM 和 CLBA 性能相差甚微，并且由于网络不稳定，在第一、三次观测试时，CLBA 传输时间略小于 PTBM，而在第二次观测时，PTBM 传输时间略小于 CLBA。

采用 5 次多项式对 3 类传输时间进行曲线拟合，进一步观测走势和性能差异。如图 5 所示，MLTT 曲线位于最下方，PTBM 与 CLBA 两条曲线基本吻合，走势相同，中间存在交线，说明二者性能近似一致，而都以 MLTT 曲线为下限。

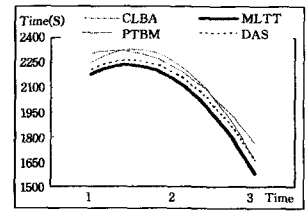

图 4 传输时间比较

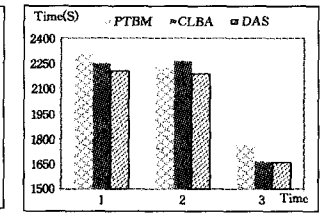

图 5 传输时间多项式曲线拟合

结束语 在数据量较小的时候，提出的算法与保守调度具有相似的性能，但是，本文的模型可以控制数据块的大小，通过优化最后一块数据的传输速度，对于共享数据量大的情

况具有极大的性能提升。还要进一步研究数据块大小选择的实际影响、多线程参数的选择以及节点失效情况等。

参 考 文 献

[1] 周文莉，雷振明. BitTorrent 文件共享系统的流量模型与文件评估方法[J]. 计算机工程，2006，32(13)：15-17

[2] 王杨，王汝传，徐小龙，等. 资源共享 P2P 网络的进化博弈激励模型[J]. 计算机工程，2011，37(11)

[3] Vazhkudai S. Enabling the co-allocation of grid data transfers [C]// Grid Computing，2003. 2003：44-51

[4] Vazhkudai S. Distributed downloads of bulk，replicated grid data [J]. Journal of Grid Computing，2004，2(1)：31-42

[5] Yang C T，Yang I H，Li K C，et al. A recursive-adjustment Co-allocation scheme in data grid environments[J]. Distributed and Parallel Computing，2005，3719：40-49

[6] Bhuvaneswaran R S，Katayama Y，Takahashi N. Dynamic co-allocation scheme for parallel data transfer in grid environment [C]// Semactics，Knowledge and Grid，2005. 2005：17

[7] Bhuvaneswaran R，Katayama Y，Takahashi N. Coordinated Co-allocator Model for Data Grid in Multi-sender Environment[C]// Service-Oriented Computing—ICSOC 2006. 2006：66-77

[8] Feng J，Cui L，Wasson G，et al. Toward seamless grid data access：Design and implementation of gridftp on. net[C] // Grid Computing，2005. 2005：8

[9] Chang R S，Chen P H. Complete and fragmented replica selection and retrieval in Data Grids[J]. Future Generation Computer Systems，2007，23(4)：536-546

[10] Ghemawat S，Gobioff H，Leung S T. The Google file system[C]// SOSP'03 Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles. 2003

[11] Yang C T，Yang I H，Li K C，et al. Improvements on dynamic adjustment mechanism in co-allocation data grid environments [J]. The Journal of Supercomputing，2007，40(3)：269-280

[12] 刘道群，孙庆和，刘君. 一种基于不同角色和反馈可信度的 P2P 信誉模型[J]. 重庆邮电大学学报：自然科学版，2010，22(6)：834-839

(上接第 130 页)

令 $g(L)=\left\lfloor \frac{L}{T'} \right\rfloor$，则由式(3)得：

$g(nT')=\left\lfloor \frac{oT'}{T'} \right\rfloor=n$

$\therefore g(L_{n+1})=n+1, g(L_{n+1})-g(L_n)=1$。

$\because L\in[L_n, L_{n+1})$，

$\therefore$ 可令 $L=L_n+x$，其中 $0\leqslant x<T'$。

按照取整函数的特性，有 $g(L)=g(L_n)=n$。

$\therefore f(Q,H,L)=\frac{H}{L_n+x}\times g(L)=\frac{nH}{L_n+x'}$。

由于参数 H、n 和 L_n 均为已知，因此该函数显然是 x 的单调递减函数。

即 $\exists T'=\frac{H}{2Q}$，当 $L\in[nT', (n+1)T')(n\in Z^+)$ 时，函数 $y=f(Q,H,L)$ 单调递减。

故得证。

命题 6 $\forall Q\in[1,255), \forall H\in[1,255)$，若 $L\in[T,\infty)$，则当 $L=2T'-1$ 时，有 $Y_L=Y_{\min}$。

证明：

由命题 2 可知，函数 $y=f(Q,H,L)$ 在每个 $[nT', (n+1)T')$ 区间内 $(n\in Z^+)$ 单调递减，即在每个 $[nT', (n+1)T')$ 区间内，当 $L=(n+1)T'-1$ 时，Y_L 获得区间最小值。下面来证明：当 $L=2T'-1$(即 $n=1$)时，Y_L 也是区间 $[L_n,\infty)$ 的最小值。

令 $L_x=(n+1)T'-1$，由命题 2 可知：

$$Y_{Lx}==\frac{H}{(n+1)T'-1}\times n=\frac{H}{T'+\frac{T'-1}{n}}$$

$\because$ 参数 H 和 T' 均已确定，

$\therefore Y_{Lx}$ 显然是 n 的单调递增函数。

也就是说，当 $n=1$，即 $L=2T'-1$ 时，有 $Y_L=Y_{\min}$。

故得证。

结束语 本文采用实验测试与理论推导相结合的办法来研究 IB QoS 机制的高优先级虚通道与高优先级虚通道的带宽分配量化关系，该项工作成果可用于更好地指导利用 IB QoS 机制进行多并发应用程序下的流量控制，从而更有效地发挥 HPC 平台上 IB 网络资源的利用率。虽然本文的结论只是基于 Mellanox 的 IB 设备的实验平台，但研究方法同样适用于研究其它厂商的设备特性。

参 考 文 献

[1] InfiniBand Trade Association. InfiniBand architecture specification volume 1[s]. Release 1. 2. 1，Nov. 2007

[2] Mellanox Technologies. Virtual Machine Migration Acceleration using Mellanox ConnectX®-2 EN 40Gb/s IO Adapter[R]. www. mellanox. com，Dec. 2010

[3] Mellanox Technologies，MTS3600 36-port InfiniBand Switch Product Development Platform[R]，www. mellanox. com，June 2008

[4] Reinemo S-A，Skeie T，Sodring T，et al. An Overview of QoS Capabilities in InfiniBand，Advanced Switching Interconnect，and Ethernet[J]. IEEE Communications Magazine，2006，44(7)：32-38

[5] Joslfaro F J S，Mendui M，Josuato J. A Formal Model to Manage the InfiniBand Arbitration Tables Providing QoS[J]. IEEE Trans. Computers，2007，56(8)

[6] Martz-Vicente A，Apostolopoulos G，Joslfaro F，et al. Efficient Deadline-Based QoS Algorithms for High-Performance Networks[J]. IEEE Trans. Computers，2008，57(7)

[7] Grant A A R R，Rashti M J. An Analysis of QoS Provisioning for Sockets Direct Protocol vs IPoIB over Modern InfiniBand Networks[C]// P2S2 Workshop，in conjunction with ICPP. 2008

[8] Subramoni H，Lai P，Panda D K. Designing QoS Aware MPI for InfiniBand[R]. Jan. 2009

[9] Grant R E，Rashti M J，Afsahi A. An Analysis of QoS Provisioning for Sockets Direct Protocol vs. IPoIB over Modern InfiniBand Networks[R]. June 2008

[10] Alfaro F J，S'anchez J L，Duato J. A Strategy to Compute the InfiniBand Arbitration Tables[R]. Jan. 2002

[11] www. mellanox. com

[12] www. openfabrics. org

[13] 吴志兵，陈忠平. InfiniBand 网络的 QoS 管理技术研究[J]. 高性能计算技术，2010(1)：33-37