文章编号:1001 - 9081(2009)10 - 2635 - 03

负载控制的网格资源调度

毕可军[1,2],段富海[2],马满福[1,2]
(1. 西北工业大学 自动化学院,西安 710072; 2. 兰州飞行控制有限责任公司,兰州 730070)
(mamanfu@ nwnu. edu. cn)

摘 要:针对网格环境下的系统负载管理,将网格系统模型描述为节点和通信链路组成的无向图。在此基础上,提出了节点和通信负载的度量方法,给出了相应的阈值定义以及网格负载水平的计算方法。由此构成了资源选择中对节点、通信链路的选择条件,实现节点、通信的负载控制;网格负载水平则是判定拒绝服务的条件,控制系统任务总量。针对该方法,设计了实现管理的体系结构,讨论了模型计算的时间周期,给出了实现控制的调度算法。实验结果表明,该方法在提高系统吞吐量、控制调度失败率等方面表现出良好的性能,特别是在重载情况下,提高了系统的运行效率,增强了系统的健壮性。

关键词:网格;负载控制;负载阈值;控制算法
中图分类号: TP302 **文献标志码**:A

Grid resource scheduling based on load control

BI Ke-jun[1,2], DUAN Fu-hai[2], MA Man-fu[1,2]
(1. *School of Automation, Northwestern Polytechnical University, Xi' an Shaanxi* 710072, *China*;
2. *Lanzhou Flight Control Company Limited, Lanzhou Gansu* 730070, *China*)

Abstract: Concerning the load system management of grid, this paper described grid model as an undirected graph (i-DG) by computing nodes connected by a set of communicating edges. Therefore, the authors presented the load measurement methods of node and communication, and brought forward how to get the load threshold and grid system load grade. By the load threshold, scheduler selected the resource whose node load was under the threshold and that communication load allowed; at the same time, the grid system load grade decided a resource request to be refused or accepted. As the requirement, the authors proposed the load management architecture, analyzed refresh periods of load threshold and grid system load grade. At last, the load management scheduling algorithm was described based on the load threshold and grid system load grade. Finally simulations were performed to compare the performance of load management scheduling algorithm with that of the None Minimisation. The experiments show that the load management scheduling algorithm is efficient and robust in controlling failed scheduling ratio and improving the throughput within grid environments, thus it is especially fit for overload.

Key words: grid; load control; load threshold; control algorithm

0 引言

目前网格资源管理中,调度主要依据用户提出的 QoS 和信任度的判定两个方面[1-2],从整个系统层面的分析、管理以及在资源调度和系统性能关系上的研究缺乏。而以信任度为依据的调度会加剧负载的失衡,信任度高的节点所提供的资源处在重载下,而信任度相对较低的系统所提供的资源则出现"饥饿"。由此,在重载下的系统不能很好地完成任务,出现时间上的延迟、调度失败率的提高等现象,同时由于这种失衡使得信任模型本身不稳定。另一方面,由于对任务的接入量不加限制,在整个网格中容易形成过载。由此导致:1)在网格系统层面,调度失衡,整体资源的效率不高,系统稳定性不好;2)在资源层面,重载使得原来高信任度的被动下降,影响信任系统的稳定性。

针对上述问题,本文在网格负载管理层面,对节点和通信负载进行度量,由此评判是否允许建立调度关系,从而防止节点和通信在资源调度中的过载,实现调度均衡;同时,通过对网格负载水平的衡量来判定当前网格的负载量,防止出现整个系统的过载。在负载管理前提下,按照用户策略,进行优化调度。

1 网格资源模型

网格可以描述为由节点和通信链路构成的无向图 $G = (N,E)$,其中,$N = \{n_1, n_2, \cdots, n_m\}$ 为网格中的节点,节点是向网格提供资源的自制系统,每个节点可向网格提供多种类型的多个资源,这些资源称为系统资源。E 为图中的边,表示网格中的通信链路,即网格通信资源。资源可以用节点作为顶点,通信资源作为边的无向图表示其状态 θ,则整个网格的资源体现为[3]:

$$R = \{\langle S_i(\theta), E_{ij}(\theta)\rangle \mid S_i = n_i, E_{ij} = n_i \times n_j, i \neq j\}$$

其中:$i,j = 1,\cdots,m$,m 表示节点数,S_i 表示计算节点的系统资源,N_{ij} 表示从节点 i 到 j 的通信资源。

一个应用同样可以被描述成任务和通信关系的无向图。应用 A 由 p 个任务组成,任务为一个调度的单位,应用对资源

收稿日期:2009 - 03 - 31;**修回日期**:2009 - 05 - 25。
基金项目: 教育部科学技术研究重点项目(208148);甘肃省科技攻关项目(2GS064-A52-035-03)。
作者简介:毕可军(1963 -),男,山东荣成人,高级工程师,博士研究生,主要研究方向:自动控制系统、分布计算; 段富海(1965 -),男,甘肃酒泉人,研究员,主要研究方向:飞控系统、嵌入式系统; 马满福(1968 -),男,甘肃甘谷人,副教授,博士,主要研究方向:计算机系统结构、移动计算。

的需求体现为：

$$A = \{t^1, t^2, \cdots, t^p\} = \{\langle t^k, e^{kl}\rangle \mid t^k = \alpha^k, e^{kl} = \alpha^k \times \alpha^l, k \neq l\}; \quad k, l = 1, \cdots, p$$

这里，α^k 为应用的任务，e^{kl} 为执行任务的通信需求，l 为所有和 α^k 相关的任务之间的通信。

在上述模型下，资源的调度就体现为一个从任务图到资源图的映射函数：

$$F(A,R) = \{\langle S_i^k, N_{ij}^{kl}\rangle\} = \{\langle t^k, e^{kl}\rangle \rightarrow \langle S_i, N_{ij}\rangle\}$$

其中，能满足任务执行需求的候选资源为：$S_k = \{\langle S_i, E_{ij}\rangle \subseteq R \mid S_i \geqslant t^k, E_{ij} \geqslant b^{kl}, i \neq j\}$。

2 负载控制模型

2.1 节点资源占空比

设节点 n_i 在时刻 t 向网格 G 注册的有效资源（即在有效的调度时间窗口内）为 $n_i^r = \{r_i^1, r_i^2, \cdots, r_i^q\}$，在该时刻，$n_i$ 上被调度的资源构成的集合为 $load_n^i = \{r_i^1, r_i^2, \cdots, r_i^l\}$，$l \leqslant q$，则定义节点 n_i 在时刻 t 的占空比为：

$$p_t^i = \frac{\mid load_n^i \mid}{\mid n_i \mid}$$

显然，占空比标志着在 t 时刻节点 n_i 上从网格角度看到的资源的负载情况，进而体现了 n_i 在该时刻负载情况。$p_t^i \in [0,1]$，其值越大，表示对应节点负载越重。

2.2 通信资源负载

设任务 T_i 到节点 n_j 的通信路径为 $e^{i,j} = (e^{i,1}, e^{1,2}, \cdots, e^{q,j})$，在一个实际的网格环境中，由于网络跳数被限制为15跳，则存在 $q < 16$。对 $e^{r,s} \in \{e^{i,1}, e^{1,2}, \cdots, e^{q,j}\}$，$1 \leqslant r < q$，$1 < s \leqslant q$，其固有带宽为 $B^{r,s}$，则 $e^{i,j}$ 对应的固有带宽为 $B^{i,j} = \min\{B^{r,s}\}$。同样，在时刻 t，$e^{r,s}$ 上的可用带宽为 $B_{load}^{r,s}$，则 $e^{i,j}$ 对应的可用带宽为 $B_{load}^{i,j} = \min\{B^{r,s}\}$。定义 $e^{i,j}$ 上通信资源的负载为：

$$p_B^{i,j} = \frac{B_{\text{load}}^{i,j}}{B^{i,j}}$$

该值收敛于[0,1]，同时能够体现 $e^{i,j}$ 上的负载情况。其中可用带宽的计算可用 Humble-Burst[4] 等方法，固有带宽可从网络参数中直接获取。

2.3 节点负载阈值和通信阈值

依据节点上的资源占空比和通信负载，为每个节点设定负载阈值，当负载超过该值时，反映出节点因为过载而导致了服务质量的下降，从负载管理的角度，不再允许建立调度关系。同理，对通信负载设定对应阈值，根据网络通信特性，当超过该值时，网络性能急剧下降。

节点负载阈值是一个动态的量，体现了节点系统对网格任务的承载能力。可以借助 TCP 协议中的慢启动策略来确定其值。首先为节点 n_i 当前资源占空比负载阈值设置一个经验型初始值 p_T^i，以信任评价的反馈为自变量进行修改。设定变量因子 $p = \text{int}\left(\frac{p_{T-1}^i}{p_{T-1}^i - p_t^i}\right)$，$p_T^i$ 的递增量定义为 $inc = \frac{(C_T^i - C_{T-1}^i)}{p}$，其中 C_T^i 为评价周期 T 内节点 n_i 所有资源信任评价的平均值，C_{T-1}^i 为周期 $T-1$ 的评价平均值。则阈值 p_b^i 的修改策略为：

$$p_T^i = p_{T-1}^i + inc;\ p_T^i \in [0,1]。$$

显然，比例因子 p 的取值范围为 $[1, p_{T-1}^i]$。当 inc 增长到 $p_{T-1}^i/2$ 之前，它以指数方式增长；此后，随着 p_t^i 的增长，逐渐减缓增长速度，平滑地接近饱和负载量。且当信任评价值递增时，负载逐步增加，否则逐步减小，使负载导致的信任评价达到最大值。

对于通信阈值，由于理论带宽和实际可用带宽往往不一致，可依据网络饱和流量的计算得到[5]。在网络达到饱和流量时，网络处在最佳的工作状态，且要求所有的流量是稳定有序的。实际网络中往往有突发流量等干扰，达不到饱和流量的条件，因此可按饱和流量的某一比例来确定通信阈值。设时刻 t 在路径 $e^{r,s}$ 上的饱和流量为 $f_s^{i,j}$，考虑突发干扰，限制允许的最大饱和程度为 $\alpha \cdot f_s^{i,j}$，$0 < \alpha < 1$，则通信负载阈值为：

$$p_T^{i,j} = \frac{\alpha \cdot f_s^{i,j}}{B^{i,j}}$$

其中，$B^{i,j}$ 为 $e^{i,j}$ 对应的固有带宽。

2.4 网格负载水平

网格依据所有节点和通信负载状态，为用户提供服务的能力和质量在一个动态的变化中。对单个节点 n_i，负载接近阈值 p_T^i 时出现性能和质量下降；同样，对通信链路 $e^{r,s}$，负载接近 $p_T^{i,j}$ 时通信可能受阻。在网格中，当有多个节点或通信链路接近其负载阈值时，整个网格的服务能力和接通能力下降，出现过载现象。据此，网格节点负载水平定义如下：

$$N_{\text{load}}^T = \begin{cases} 1, \lambda \leqslant \dfrac{\sum p_T^i}{m}; i = 1, \cdots, m \\ 0, \text{其他} \end{cases}$$

其中：m 为网格中的节点数，λ 为节点饱和时的负载，是一个经验值。类似地，网格通信负载水平定义为：

$$E_{\text{load}}^T = \begin{cases} 1, \mu \leqslant \dfrac{\sum p_T^{i,j}}{2m^2}; i = 1, \cdots, m, j = 1, \cdots, m-1 \\ 0, \text{其他} \end{cases}$$

其中 μ 为通信饱和时的网络负载。

网格节点和通信的综合负载定义为：

$$NE_{\text{load}}^T = \alpha \cdot N_{\text{load}}^T + \beta \cdot E_{\text{load}}^T;\ 0 < \alpha, \beta < 1, \alpha + \beta = 1$$

由上述定义，网格负载定义为一个二值函数：

$$G_{\text{load}}^T = \begin{cases} 1, N_{\text{load}}^T \text{ or } E_{\text{load}}^T \text{ or } NE_{\text{load}}^T = 1 \\ 0, \text{其他} \end{cases}$$

显然，网格负载有重载和轻载两种状态。上述定义中，将节点负载、通信负载和综合负载分别定义，是因为网格资源提供能力和通信能力不一定匹配，且任务类型对通信和处理能力的要求是动态变化的，但任何一种类型的过载都会导致系统性能和服务质量的下降。

2.5 时间周期

阈值计算、节点、通信链路以及整个网格负载的计算是按照各自的时间周期 T 进行的，T 的定义应当能够反映当前节点、网格等负载的动态变化，同时也不宜太小，以免带来太大的计算负载。T 大小一般可根据系统中平均任务执行时间的整数倍来计算，其倍数则与网格的吞吐能力相关，因为吞吐能力强的网格改变其负载状态需要的任务也多。如网格负载周期可用如下方法来确定：设网格 $G = (N, E)$ 的吞吐能力为 A，任务平均执行时间为 P，考虑资源调度的多样性，则 T 可定义为 $T = l \times A \times P$，其中 $l = 5, \cdots, 10$ 为宜。一般地，由于阈值反映的是更为局部的负载，与网格负载水平比较，其计算周期较小。

3　体系结构

负载管理是网格域管理的组成部分,其结构如图 1 所示。

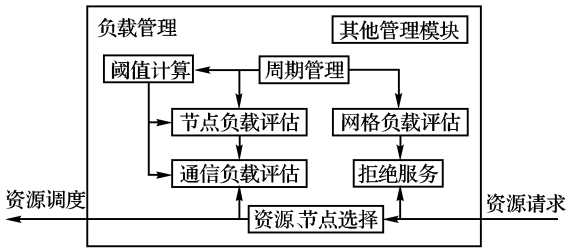

图 1　负载管理结构

周期管理按照设定的大小产生时间周期,以驱动对网格负载和节点以及通信负载的评价;阈值计算按照其周期计算网格中所有节点和通信链路的当前阈值,在周期上和其他评估是异步进行的;网格负载评估按照周期评估当前网格负载水平,并判定当前是否进入重载状态,当进入重载时,驱动拒绝服务拒绝当前的所有请求,从而使系统恢复到轻载状态;节点负载和通信负载的评估各自周期性地完成所有节点和通信链路的负载评估,判定是否到达各自允许的阈值,以备调度时查询。当一个资源请求到来时,首先检测网格负载水平是否为重载,如为重载,则拒绝请求,否则按照请求 QoS 选择资源,针对选择资源,检测对应节点和通信链路是否达到负载阈值,然后判定是否允许调度。

4　负载控制算法

节点、通信阈值和网格负载水平的定义为资源调度从管理和系统角度设定了要求,即只有网格轻载时接受请求,只有目标资源所在的系统和对应通信链路没达到既定阈值时方可建立调度关系。在该前提下,按照系统的优化策略进行优化调度,管理控制算法如下:

```
Manager( T_i , G^T_load )
  { S = ∅ ;
  If G^T_load = 1 then
    { refuse to T_i by G ;        //网格当前状态为重载,拒绝服务
    abort("the task has been refused by system for overload") }
    Else { selecting r_QoS ≥ T_i,QoS from GRIS content
      S = S + r;                  // 产生满足任务 T_i 的候选资源集合
    If S ≠ ∅ then {
      { for every r_j ∈ S do
      If ( p^i_i ≥ p^i_T or p^ij_B ≥ p^ij_T ) then delete r_j from S }
      //删除系统或通信过载的资源
      if S = ∅ then break;
        selecting r_i ∈ S to task T_i as policy
        //按照系统优化策略进行调度
    }
  }
}
```

上述算法中,GRIS(Grid Resource Information Services)为网格资源信息服务机构。

5　实验验证

5.1　比较算法

目前比较典型的调度算法是文献[6]作者提出的时间最小化、成本最小化和非最小化算法,这里,由于负载控制发生在算法执行之前,采用非最小化算法,在实验测试平台 Nimrod/G[6] 上进行实验。非最小化算法描述为:

1)按照时间和预算的约束来产生候选资源;

2)对于费用更低的资源,按照任务完成时间的反比分配任务;

3)对于费用更高的资源,重复上述步骤,直到所有任务得到资源。

5.2　实验设计

基于 Nimrod/G 平台实现了原型系统,系统由 40 个节点组成,节点既是资源的提供者,也是任务的宿主者。为了选择资源,系统提供一个中心节点来实现资源的注册、发布以及部署 GRIS,另外提供一个节点部署负载管理部件,来执行负载控制。网格上提供了 5 类资源,资源的登陆以及退出采用符合泊松分布的随机函数进行调度。网格负载水平定义中,λ 和 μ 取值为 0.7,α,β 均为 0.5。任务的平均执行周期为 50 ms,阈值计算、节点、通信链路以及整个网格负载的计算时间周期均取 T = 4 s,由负载管理负责执行。在各个节点上,任务按照线性增长的方式向网格提供,以获得饱和的负载量。

5.3　实验及结果分析

实验针对两个算法在每个节点的平均吞吐量、调度失败率、拒绝服务的数量等参数,分别在轻载和重载情况下取得了实验结果。为了使网格工作在轻载状态,任务的提供在线性增长的方式下,一旦出现拒绝服务,将任务注入频率减半,再次循环增长。而重载状态则出现拒绝服务后,对任务注入频率不减半、不增长。实验结果如表 1 和 2 所示。

表 1　轻载实验结果

算法	吞吐量	调度失败率/%	拒绝服务数
非最小化算法	48 720	4.60	—
负载控制算法	46 917	3.12	1 538

从表 1 看出,在吞吐量上,非最小化算法比负载控制算法高 2.37%,这一结果体现了轻载控制出现瞬时饱和时拒绝的请求、资源枯竭等;调度失败率由 4.60% 下降到 3.12%。但总体上看,二者在上述参数上很接近,拒绝服务的数量也很少,占请求总量的 3.17%。

表 2 重载实验结果

算法	吞吐量	调度失败率/%	拒绝服务数
非最小化算法	58 766	12.56	—
负载控制算法	61 438	4.32	11 462

由表 2 可见,负载控制算法在吞吐量上超过了非最小化算法,提高了 4.54%,而调度失败率则明显小于非最小化算法,由 12.56% 下降的 4.32%,同时,拒绝服务数为 11 462,占到请求总数的 15.72%。

按照线性增长的方式给网格连续注入任务,两种算法在拒绝服务数与调度失败率上体现出的特性如图 2、3 所示。图 2 中,当节点中的任务数每小时低于 5 万个时,系统处于轻载,两个算法产生的拒绝服务数基本一致,拒绝服务是由资源类型的匹配等原因造成;当任务数每小时超过 5 万个后,系统进入重载状态,非最小化算法对此不予处理,而负载控制算法则大量拒绝请求,以保证系统的稳定性和有效性。类似地,图 2 显示,当节点中的任务数每小时大于 5 万个时,非最小化算法的调度失败率急剧增加,而负载控制算法则保持了相对的稳定。

比较轻载和重载,可以看到,吞吐量在重载时变化明显,且负载控制算法优于非最小化算法;而调度失败率重载和

(下转第 2651 页)

6　结语

本文构造了一种基于 P2P 的计算资源共享与聚集平台，该平台的高效性得益于建立在 JXTA 协议上的节点组织机制,易用性得益于建立在 JXTA 协议上的并行编程库,通过作业管理子系统即可以把 Internet 环境下的志愿机组织起来,实现大规模计算资源共享与聚集。通过对 PCP 平台的通信延迟、吞吐量及加速比进行测试得知,该平台取得了比较好的性能。我们在开发 PCP 平台的过程中都假设主控节点是比较稳定的,所以提高平台的容错性能是本文的下一步工作。

参考文献:

[1] ANDERSON D P. BOINC: A system for public-resource computing and storage[C]// GRID 2004: Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing. New York: ACM Press, 2004: 4 - 10.

[2] NEARY M O, PHIPPS A, RICHMAN S, *et al*. Javelin 2.0: Java-based parallel computing on the Internet[C]//Proceedings from the 6th International Euro-Par Conference on Parallel Processing, LNCS 1900. London: Springer-Verlag, 2000: 1231 - 1238.

[3] FEDAK G, GERMAIN C, NERI V, *et al*. XtremWeb: A generic global computing system[C]// Proceedings of the 1st IEEE/ACM International Symposium on Cluster Computing and the Grid. New York: ACM Press, 2001: 582 - 587.

[4] Distributed. net[EB/OL]. [2009 - 01 - 01]. http://www.distributed.net/.

[5] 章勤, 鄢娟, 金海, 等. 吴宇网络计算平台体系结构研究[J]. 计算机研究与发展, 2003, 40(12): 1725 - 1730.

[6] KORPELA E, WERTHIMER D, ANDERSON D, *et al*. SETI@home: Massively distributed computing for SETI[J]. Computing in Science and Engineering, 2001, 3(1): 78 - 83.

[7] GONG LI. JXTA: A network programming environment[J]. IEEE Internet Computing, 2001, 5(3): 88 - 95.

[8] SARMENTA L F G. Sabotage-tolerance mechanisms for volunteer computing systems[C]// CCGRID: Proceedings of the 1st International Symposium on Cluster Computing and the Grid. Washington, DC: IEEE Computer Society, 2002: 561 - 572.

[9] HEWGILL G. RC5 and Java Toys[EB/OL]. [2009 - 02 - 02]. http://www.hewgill.com/rc5/index.html.

[10] DJILALI S. P2P-RPC: Programming scientific applications on peer-to-peer systems with remote procedure call[C]// CCGRID: Proceedings of the 3st International Symposium on Cluster Computing and the Grid. Washington, DC: IEEE Computer Society, 2003: 406 - 413.

[11] 窦文, 贾焰, 王怀民, 等. 基于对端重叠网络的通用大规模计算资源共享环境的构造[J]. 计算机学报, 2004, 27(1): 21 - 31.

[12] VERBEKE J, NADGIR N, RUETSCH G, *et al*. Framework for peer-to-peer distributed computing in a heterogenuous and decentralized environment[C]// Grid 2002: Proceedings of the third International workshop on Grid Computing, LNCS 2536. Heidelberg: Springer, 2002: 1 - 12.

(上接第 2637 页)

轻载变化不大,而随着负载量的不断注入,拒绝服务依据系统的负载,将当前系统的负载控制在一个系统承载接近饱和的状态。分析原因:1)调度失败率体现了节点在轻载时系统固有的稳定性和可靠性,同时也体现了通信负载的限制使得丢包率减少;2)拒绝服务则一定程度上实现了网格对负载饱和状态的界定;3)吞吐量的提高则体现了系统良好的性能,工作的更加有序,冲突少。

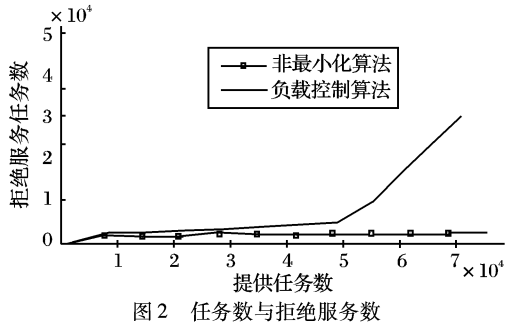

图 2　任务数与拒绝服务数

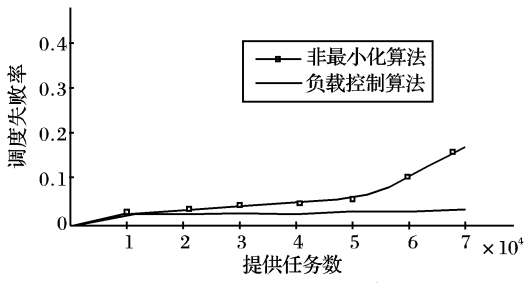

图 3　任务数与调度失败率

显然,负载控制算法通过控制节点的负载量、通信的承载量以及整个网格系统的总负载,在资源的调度和任务的总量控制上实现了主动管理,使系统更为健壮,尤其在系统进入重载时,效果更加明显。

6　结语

文章通过对网格节点负载、任务和资源间通信链路负载和整个系统负载的评价,在节点和通信上平衡负载,按照系统的吞吐能力,在系统级设计了任务量的控制,由此使得系统在负载平衡和总负载量上实现了主动控制,提高了系统的运行效率,增强了系统的健壮性。实验表明,所提出的方法在系统吞吐量、调度失败率等方面表现出良好的性能,是有效的。

参考文献:

[1] BRADLEY A, CURRANN K, PARRZ G. Resource discovery and management in computational GRID environments[J]. International Journal of Communication Systems, 2007, 19(6): 639 - 657.

[2] ALOISIO G, CAFARO M, EPICOCO I, *et al*. Resource and service discovery in the igrid information service[C]//Proceedings of International Conference on Computational Science and Applications, LNCS 3482. Berlin: Springer-Verlag, 2005: 1 - 9.

[3] MA MAN-FU, WU JIAN, LI SHU-YU, *et al*. . A grid-distance based scheduling for grid resource management[C]// Proceedings of HPC. Los Alamitos: IEEE Computer Society, 2005: 576 - 581.

[4] 刘敏, 李忠诚, 石晶林, 等. IPv6 网络中基于优先级的可用带宽测量方法[J]. 计算机研究与发展, 2004, 41(8): 1361 - 1367.

[5] MUU L D. DC optimization methods for solving minimum maximal network flow problem[EB/OL]. [2009 - 02 - 01]. http://www.mmm.muroran-it.ac.jp/~shi/muushi.pdf.

[6] BUYYA R, ABRAMSON D, GIDDY J. Nimrod/G: An architecture or a resource management and scheduling system in a global computational grid[C/OL]. [2009 - 03 - 01]. http://www.gridbus.org/papers/GridEconomy.pdf.