第40卷 第4期 西 安 交 通 大 学 学 报 Vol. 40 №4
2006年4月 JOURNAL OF XI′AN JIAOTONG UNIVERSITY Apr. 2006

网格工作流中基于优先级的调度方法研究

刘 洋, 桂小林, 徐玉文
(西安交通大学计算机科学与技术系, 710049, 西安)

摘要: 根据网格用户的身份、信誉,以及网格工作流中应用程序的时间紧迫度和依赖关系,提出了网格工作流中应用程序的优先级计算方法.其中,完全优先级调度算法根据网格工作流中的应用程序优先级向量生成调度序列,而且每次只将队列中优先级最高的应用程序提交给网格,而非完全优先级调度算法可同时调度若干无依赖的网格应用程序进入网格,以弥补完全优先级调度算法的并行化问题.实验表明,当网格中的资源较少且资源的性能差异较大时,使用完全优先级调度算法可以保证优先级较高的工作流的应用程序优先使用优势资源;当网格中的资源性能差异不大时,非完全优先级调度算法可解决因资源未充分利用而造成工作流完成时间大幅增加的问题.

关键词: 工作流;网格;优先级;调度算法
中图分类号: TP393 文献标识码: A 文章编号: 0253-987X(2006)04-0411-04

Study on Scheduling Methods Based on Priorities in Grid Workflow

Liu Yang, Gui Xiaolin, Xu Yuwen
(Department of Computer Science and Technology, Xi′an Jiaotong University, Xi′an 710049, China)

Abstract: A priority calculating method was proposed, in which the grid workflow applications' priority vector was calculated by the users' identity, credit, urgency and dependency of grid applications in the workflow. The full priority scheduling algorithm generates the scheduling list according to the priority vectors, and only the application which has the highest priority can be submitted while the non-absolute priority scheduling algorithm schedules several independent grid applications so as to compensate the parallelism problem of the full priority scheduling algorithm. Experiments show that, when the grid has fewer available resources or the performance discrepancy of the resources is much higher, the full priority scheduling algorithm ensures the workflow applications that have high priority to be scheduled firstly; and when the grid contains more resources that have the same performance, the non-absolute priority scheduling algorithm avoids largely increasing of the workflow's makespan caused by resources waste.

Keywords: workflow; grid; priority; scheduling algorithm

用户在网格中使用工作流技术[1],可以根据自己的工作流程把简单、独立的网格应用程序组合起来,实现大规模、复杂的业务层网格应用.网格工作流系统 WaderFlow 是建立在校园级网格中间件 GridWader[2] 基础之上的,可以自行调度并运行网格工作流中定义的应用程序.

目前,已有的网格工作流系统大多都是根据高能物理或者生物信息学等专用网格系统中的工作流程复用等需求而设计的[3-6],它们更偏重于工作流程建模,而其用户群比较单纯,所以没有划分用户和应用程序的优先级别,相应的调度策略也无优先性.对于用户群体比较复杂的一般网格而言,由于资源的

收稿日期: 2005-07-15. 作者简介: 刘 洋(1982~),男,硕士生;桂小林(联系人),男,教授,博士生导师. 基金项目: 国家自然科学基金资助项目(60273085);国家高技术研究发展计划资助项目(2001AA111081,2002AA104310);教育部新世纪优秀人才支持计划资助项目.

有限性,级别较高的用户提交的紧迫性应用程序需要一定的机制来保证优先执行. 本文首先提出了网格工作流中应用程序的优先级计算方法,然后在此基础上提出了事件驱动的完全优先级调度算法和非完全优先级调度算法.

1 网格工作流系统 WaderFlow

WaderFlow 包括用户接口和工作流管理系统,它与网格中间件 GridWader 结合可形成一种分层的结构系统,如图 1 所示.

用户接口主要用于网格用户的身份验证、建立会话和网格工作流系统与用户的交互. 工作流管理系统负责网格工作流的解析、调度,将程序交予网格中间件运行,取回运行结果并返馈给用户接口. 在网格工作流中应用程序优先级的计算和基于优先级的调度均在工作流管理系统中完成.

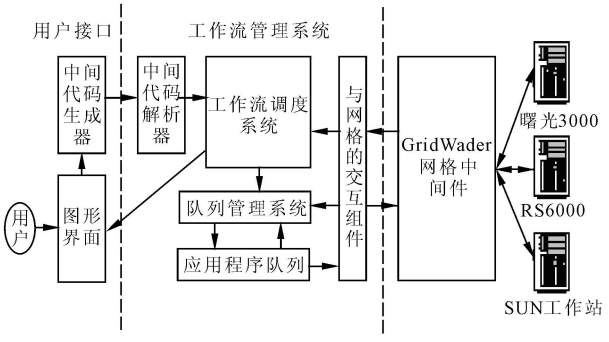


图 1 WaderFlow、GridWader 结合下的分层结构系统

2 网格工作流中应用程序的优先级定义

可用信誉[7] 来评估网格用户的行为,信誉良好的用户应该得到较高的优先级别. 从发展的角度来看,网格的使用是有偿的,对于不同身份的用户(普通用户、VIP)应分别提供不同的优先级服务. 在用户提交的应用程序中,往往有些应用程序对时间和资源的要求较高,紧迫性较强,这些"紧急"的应用程序应该具有更高的优先级别.

在 GridWader 系统中,用户的身份可以表示为 $r^I \in M$, $M=\{$普通用户, VIP$\}$,用户的信誉可以表示为 $r^C \in N$, $N=\{$较差, 一般, 良好$\}$,则用户的特权等级可以表示为 $r^u \in M \times N$. 对集合 M 指定的量化函数是 $f(M_i)=i(M_i \in M, i=[1, 2])$,对集合 N 指定的量化函数是 $\phi(N_i)=i(N_i \in N, i=[1, 2, 3])$,则用户的特权等级可以量化为 $r^u=f(r^I)\phi(r^C)$. 在提交应用程序时,可根据用户指定的时间紧迫度为应用程序设定时间紧迫系数 $r^T \in U$, $U=\{$缓慢, 普通, 紧急$\}$,为集合 U 指定的量化函数是 $\omega(U_i)=i(U_i \in U, i=[1, 2, 3])$,则应用程序的紧迫度可以表示为 $r^t=\omega(r^T)$.

定义 1 由提交工作流的用户特权等级 r^u 和工作流中应用程序 P_i 的紧迫度 r_i^t 产生的 P_i 的优先级作为 P_i 的直接优先级 R_i^1,计算式为

$$\left.\begin{aligned} R_i^1 &= \alpha r^u + \beta r_i^t \\ \alpha+\beta &= 1 \end{aligned}\right\} \tag{1}$$

在工作流中应用程序的直接依赖是指用户在定义工作流时为其指定的依赖应用程序,工作流中应用程序的优先级包含直接优先级和间接优先级,它们与工作流中的位置和依赖关系有关.

定义 2 在工作流中应用程序的依赖关系所产生的优先级为间接优先级,设应用程序间接优先级的递增量为 τ,工作流 W 中应用程序 P_j 的间接优先级为 R_j^2,其直接依赖的应用程序 P_i 的间接优先级为 R_i^2,则间接优先级的赋值方法为

$$\left.\begin{aligned} R_j^2 &= 0, \quad \text{没有程序直接依赖于 } P_j \\ R_i^2 &= R_i^2 + R_j^2 + \tau \end{aligned}\right\} \tag{2}$$

定义 3 在工作流中应用程序 P_i 的优先级为二维向量 $\pi_i=\langle R_i^1, R_i^2\rangle$.

定理 1 用户直接优先级的排序不能违背间接优先级的排序.

证明 应用程序的直接优先级反映了调度的优先权,间接优先级反映了工作流应用程序之间的依赖关系. 间接优先级规定了应用程序必须在依赖的应用程序完成后才能执行,所以被依赖的应用程序必须优先调度和运行,必须保证被依赖的应用程序的直接优先级别更高.

推论 1 设应用程序 P_j 的直接优先级为 R_j^1,其直接依赖的应用程序 P_i 的直接优先级为 R_i^1,则

$$R_i^1 = \max(R_j^1, R_i^1) \tag{3}$$

定理 2 在应用程序队列中,保证等待的应用程序不会因为初始优先级较低而"饿死"的方法是:直接优先级的增长率为 $\frac{\nabla R}{\nabla t}$,队列中应用程序 P_i 的原始直接优先级为 $R_i^{1'}$,在队列中等待的时间为 t_i,当新的应用程序到来时,队列管理器可从

$$R_i^1 = R_i^{1'} + t_i \frac{\nabla R}{\nabla t} \tag{4}$$

重新为其计算直接优先级.

证明　应用程序的直接优先级反映了调度的优先权, 它决定了应用程序在队列中的位置, 并且随着等待时间的增加不断增大. 当新的应用程序到来时, 原有的应用程序就以高优先级排在新的应用程序之前, 从而获得被提交和执行的机会, 因此不会出现"饿死".

3　工作流中应用程序的优先级生成算法

设在 W 中 P_i 的优先级为 π_i, 直接优先级和间接优先级分别为 R_i^1、R_i^2, 若为每个 P_i 设置计算标志 z_i 来表示优先级是否已经计算, 则 W 中应用程序的优先级的计算算法描述如下.

步骤 1: 初始化. 它针对的是 W 中每个 P_i, 令 $R_i^1=R_i^2=0$, $z_i=0$.

步骤 2: 对 W 中每个 P_i 用式(1)计算直接优先级 R_i^1.

步骤 3: 选择工作流中 $z_i=0$ 且位置最后(位置最后是指不被任何一个 $z_j=0$ 的 P_j 所依赖)的一个 P_i, 按照式(2)和式(3)依次计算其直接依赖的间接优先级和直接优先级, 并令 $z_i=1$; 如果 P_i 不直接依赖于任何应用程序, 令 $z_i=1$.

步骤 4: 重复步骤 3, 直到对 W 中任意 P_i 都有 $z_i=1$.

步骤 5: 对 W 中的每个 P_i, 有优先级向量 $\pi_i=\langle R_i^1, R_i^2\rangle$.

4　事件驱动的完全优先级调度算法和非完全优先级调度算法

当网格工作流系统已提交的应用程序在网格中执行完毕时, 会触发队列管理器的提交事件, 使队列中优先级最高的应用程序进入网格运行. 采用事件触发的方式可以使应用程序的运行能够适应网格的动态性, 并且使网格工作流系统和网格中间件处于一种松散耦合的结构, 这样可扩展性更好.

完全优先级调度算法是完全按照应用程序的优先级进行调度的方法, 每次只将队列中优先级最高的应用程序提交给网格, 以使其获得网格中最多、最好的资源, 取得最佳的执行效果.

完全优先级调度算法描述如下.

(1)工作流调度系统.

步骤 1: 查询工作流中间代码解析器是否有新的工作流, 如果有, 转步骤 2, 否则上转步骤 1.

步骤 2: 对用户提交的 W^k, 由工作流调度系统按照第 3 节中的算法, 为其中的每个 P_i^k 计算优先级向量 π_i^k.

步骤 3: 对应用程序队列中等待的所有应用程序用式(4)重新计算优先级.

步骤 4: 将 W^k 中的每个 P_i^k 按照优先级向量 π_i^k 的直接优先级插入到优先级队列中, 如果 2 个应用程序的直接优先级相同, 则比较它们的间接优先级.

步骤 5: 转步骤 1.

(2)队列管理器.

步骤 1: 等待来自网格的应用程序完成事件, 如果有, 转步骤 2, 否则继续等待.

步骤 2: 将队列中位于队首的优先级最高的应用程序提交给网格中间件, 并将其从队列中删除.

步骤 3: 转步骤 1.

完全优先级调度算法保证了具有高优先级的应用程序可以获得最佳的执行环境和最短的等待时间, 但是由于缺少并行化, 网格资源的利用率较低. 针对完全优先级调度算法存在资源浪费的缺陷, 本文提出一种改进算法——事件驱动的非完全优先级调度算法. 在非完全优先级调度算法中, 应用程序队列只按照直接优先级的顺序放置工作流中无依赖的或者依赖的应用程序都已经执行完毕的应用程序. 在工作流调度系统中, 该算法为每个工作流建立一个临时队列, 队列按照工作流的应用程序的间接优先级排序.

非完全优先级调度算法描述如下.

(1)工作流调度系统.

步骤 1: 对用户提交的 W^k, 由工作流调度器按照第 3 节中的算法, 为其中的每个 P_i^k 计算优先级向量 π_i^k.

步骤 2: 为 W^k 建立临时工作队列 Q^k, 将 W^k 中的每个 P_i^k 按照优先级向量 π_i^k 中的间接优先级大小插入到 Q^k 中.

步骤 3: 在应用程序队列中等待的应用程序用式(4)重新计算其直接优先级.

步骤 4: 如果 Q^k 中仍有应用程序, 则将 Q^k 中具有最高间接优先级的应用程序取出, 并按照其优先级向量中的直接优先级大小插入到应用程序队列中, 然后将其从 Q^k 中删除; 否则, 删除 Q^k.

步骤 5: 等待来自网格的应用程序完成事件和新的工作流到达事件, 如果前者到达, 转步骤 6; 如果后者到达, 上转步骤 1; 否则, 继续等待.

步骤 6: 在已完成的 P_i^k 所对应的 Q^k 中, 如果

与 P_i^k 的间接优先级相同的应用程序都已经执行完毕, 则上转步骤 3; 否则, 上转步骤 5.

(2)队列管理系统.

步骤 1: 等待来自网格的应用程序完成事件, 如果该事件到达, 转步骤 2; 否则, 继续等待.

步骤 2: 将队列中位于队首位置的应用程序提交给网格中间件, 同时将其从队列中删除, 并从网格获取资源信息.

步骤 3: 如果网格中还有可用资源, 上转步骤 2; 否则, 上转步骤 1.

在调度的过程中, 非完全优先级调度算法依然是从优先级最高的应用程序开始的, 从而保证了高优先级的应用程序可以获得较好的资源和最短的执行时间. 同时, 它允许其他无依赖关系的应用程序并行运行, 由此提高了资源的利用率.

5 基于优先级调度的一个示例

如图 2 所示, 2 个等级不同的用户提交的工作流分别为 W^1 和 W^2, 初始状态和权重系数分别为 α、β, 则用第 3 节中的算法计算 W^1 和 W^2 中各程序的优先级向量, 结果如表 1 所示.

表 1 W^1 和 W^2 中应用程序的优先级向量

应用程序	A^1	B^1	C^1	A^2	B^2	C^2	D^2
优先级向量	⟨3, 6, 2⟩	⟨3, 6, 1⟩	⟨3, 6, 0⟩	⟨2, 4, 4⟩	⟨2, 4, 1⟩	⟨2, 1⟩	⟨2, 0⟩

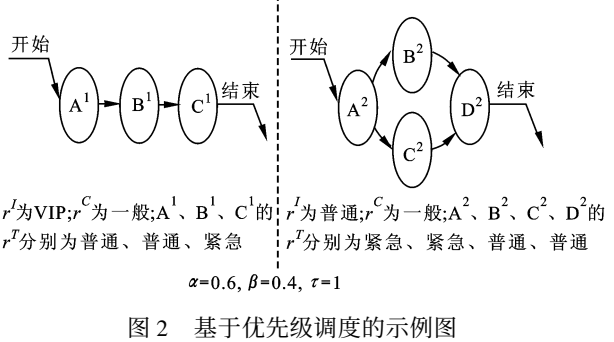
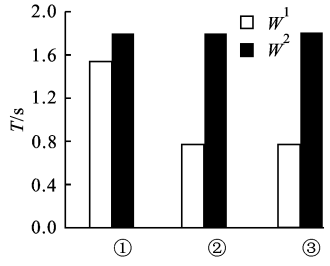
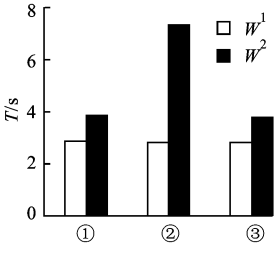


图 2 基于优先级调度的示例图

为了考察文中所述 2 种调度算法在不同网格资源状况下的性能, 可进行如下实验: 设当前网格系统中有 2 个可用资源, 其性能比分别为 4 ∶1 和 1 ∶1; 又设 W^1 和 W^2 中每个应用程序的运行时间均为 T, 分别按照非优先级的 Fast Greedy 算法[8]、完全优先级算法、非完全优先级算法, 在 2 种性能比之下对这 2 个工作流进行调度, 实验结果如图 3 所示. 由图 3a 可以看出: 当网格中的资源较少且资源的性能差异较大时, 完全优先级调度算法可保证在 W^1 中优先级较高的应用程序优先使用优势资源; Fast Greedy 算法则使 W^2 中的应用程序和 W^1 中的应用程序争夺优势资源, 造成 W^1 的完成时间比较长; 非完全优先级算法的效果和完全优先级算法是一样的. 由图 3b 可以看出: 当网格中的资源性能差异不大时, 完全优先级调度算法虽然可保证高级别用户提交的 W^1 的完成时间较短, 但是会因资源浪费造成普通用户提交的 W^2 的完成时间大幅增加; 非完全优先级调度算法既保证了 W^1 的完成时间较短, 又与 Fast Greedy 算法一样, 充分利用了网格中的资源, 使得普通用户提交的 W^2 的完成时间有所减少.



(a) 资源较少且资源性能差异较大



(b) 资源性能差异不大

①: Fast Greedy 算法; ②: 完全优先级算法; ③: 非完全优先级算法

图 3 3 种算法在 2 种性能比下的工作流调度结果

6 结 论

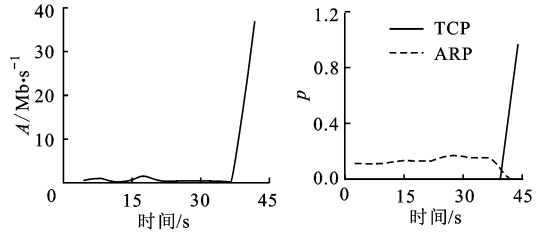
通过对网格工作流中应用程序优先级的定义, 可以使用优先级调度算法来保证优先级较高的网格工作流应用程序能够被优先调度和运行. 在提出的 2 种优先级算法中, 非完全优先级算法不仅可以保证高优先级的工作流应用程序的完成时间较短, 又能够充分利用网格中的资源, 具有较好的性能.

参考文献:

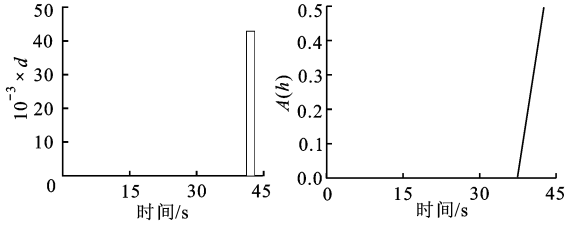
[1] 史美林, 杨光信, 向 勇, 等. WfMS: 工作流管理系统[J]. 计算机学报, 1999, 22(3): 325-334.

(下转第 419 页)

度量值如图 3 所示, 从图 3d 可以看出, 从时刻 8 开始异常度值骤然升高, 这说明本文方法可以及时准确地评估流量异常.



(a) 数据传输速率指标　　(b) TCP、ARP 协议比重



(c) SYN 报文平均数　　(d) 流量评估异常度曲线

图 3　一种新攻击的评估结果

3　结论与展望

网络流量的变化与网络行为存在着一定的联系. 本文通过分析用户正常行为引发的流量变化与恶意攻击造成流量突变之间的差别, 提出了一种主机实时流量的评估方法, 并且证明这种方法对评估 DDoS、Dos、worm 等网络攻击的有效性.

本文提出的网络流量评估方法可以结合主机的漏洞、开启的服务等信息, 对系统的安全态势进行评估, 也可作为短期评估的补充, 亦可以结合抽样理论对整个局域网系统的网络流量状况进行分析.

参考文献:

[1] Hariri S, Qu Guangzhi, Dharmagadda T, et al. Impact analysis of faults and attacks in large scale networks [J]. IEEE Security and Privacy, 2003, 1(5): 49-54.

[2] Denning D E. An intrusion detection model [J]. IEEE Transactions on Software Engineering, 1987, 13(2): 222-232.

[3] Lan Kunchan, Hussain A, Dutta D. Effect of malicious traffic on the network [A]. Passive and Active Measurement Conference, San Diego, USA, 2003.

[4] 程　光, 龚　俭, 丁　伟. 基于抽样检测的高速网络实时异常检测模型 [J]. 软件学报, 2003, 14(3): 594-599.

[5] Hu Hanping, Guo Wenxuan. A method of security measurement of network data transmission [A]. 19th IEEE International Parallel and Distributed Processing Symposium, Nice, France, 2005.

[6] Ham J, Kamber M. 数据挖掘概念与技术 [M]. 范　明, 孟小峰, 译. 北京: 机械工业出版社, 2001.

(编辑　苗　凌)

(上接第 414 页)

[2] 桂小林, 钱德沛, 何　戈. 基于校园网络的元计算实验系统 WADE 的设计与实现 [J]. 计算机研究与发展, 2002, 39(7): 888-894.

[3] Graham G E, Evans D, Bertram I. McRunjob: a high energy physics workflow planner for grid [A]. Conference on Computing in High Energy and Nuclear Physics, La Jolla, USA, 2003.

[4] von Laszewski G, Zaluzec N, Hategan M, et al. Gridant: client side workflow management in grids with application onto position resolved diffraction [A]. Midwest Software Engineering Conference, Chicago, USA, 2003.

[5] Wroe C, Goble C, Greenwood M, et al. Automating experiments using semantic data on a bioinformatics grid [J]. IEEE Intelligent Systems, 2004, 19(1): 48-55.

[6] Cao J, Jarvis S A, Saini S, et al. Gridflow: work flow management for grid computing [A]. 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid, Tokyo, Japan, 2003.

[7] Gui Xiaolin, Xie Bing, Li Yinan, et al. A grid security infrastructure based on behaviors and trusts [A]. 3rd International Conference on Grid and Cooperative Computing, Wuhan, China, 2004.

[8] Armstrong R, Hensgen D, Kidd T. The relative performance of various mapping algorithms is independent of sizable variances in run time predictions [A]. 7th IEEE Heterogeneous Computing Workshop, Orlando, USA, 1998.

(编辑　苗　凌)