QoS约束下基于双向分层的网格工作流调度算法

姚 磊 戴冠中 张慧翔 任 帅
(西北工业大学自动化学院 西安 710072)

摘 要 为使网格工作流的执行满足用户QoS要求,应用有向无环图描述工作流,并分析其中的关键活动,把用户对工作流的整体QoS约束分割为对单个任务的QoS约束。以此为基础,提出了一种基于双向分层的网格工作流调度算法Q-TWS。该算法通过对工作流正向分层和逆向分层,可以方便并准确找到任务之间的并行关系。Q-TWS可最大程度放松对任务执行时间的约束,在增加调度灵活性的同时又满足用户的QoS要求。实验表明,Q-TWS算法与TL算法相比,在同样的截止时间约束下,工作流执行时间较短,且工作流执行费用较小。

关键词 网格计算,工作流调度,QoS约束,双向分层
中图法分类号 TP391 **文献标识码** A

QoS-constrained Workflow Scheduling Algorithm for Grid Computing Based on Two-way Stratified

YAO Lei DAI Guan-zhong ZHANG Hui-xiang REN Shuai
(College of Automation, Northwestern Polytechnical University, Xi'an 710072, China)

Abstract In order to meet user's QoS requirement for the implementation of grid workflow, the key tasks of the workflow were analyzed firstly, and the QoS of the whole workflow was divided into segments which are the QoS-constrained of a single task. Then, a grid workflow scheduling algorithm (Q-TWS) based on Two-Way Stratified was proposed. Through both positive layering and reverse layering, this algorithm can find the parallel relation between tasks easily and accurately. Q-TWS can relax the task execution time, increase flexibility scheduling and meet user QoS requirements. Simulation results show that Q-TWS has a shorted execution time and a less execution cost compared with BL when the two algorithms have the same deadline.

Keywords Grid computing, Workflow scheduling, QoS-constrained, Two-way stratified

1 引言

随着网格技术的不断发展,其处理问题的复杂度也逐渐增加。为了进一步研究网格序列任务调度算法[10,16,17],必须加强对网格工作流的管理。工作流管理联盟(Workflow Management Coalition, WfMC)对工作流的定义是[1]能够完全或部分自动执行的业务过程,根据一系列过程规则,文档、信息或任务能够在不同的执行者之间进行传递和执行。开放网格服务体系结构OGSA[2]提出在网格环境中把工作流中的任务分配到合适的Web服务上,完成整个网格工作流。GridAnt[3]和GridFlow[4]讨论了在网格环境中工作流管理的问题。

一般可以把网格工作流管理分成两类[5]:一类是尽力而为(Best-effort),典型的是Condor DAGMan[6]中实现的Myopic[7]算法,其基本原则是把任务分配到一个期望完成时间最早的资源上;另一类是QoS约束的工作流管理,此类方法还处于研究的初期阶段,文献[8,9]分别提出了两个不同的满足时间约束的网格工作流管理算法。基于QoS约束的工作流管理方法的目标是在网格服务提供者和网格用户之间建立服务水平协议(Service Level Agreements, SLA)。根据协商好的SLA,网格资源向用户提供一定水平的服务。文献[14,15]的研究说明了在大尺度的网格环境中,使用集中式工作流管理是不可行的,并提出了多Broker协作调度的思想。文献[18]采取由后向前方法将任务逆向分层(Bottom Level, BL),将工作流截止期转化为层截止时间,提出截止期约束的逆向分层费用优化算法DBL (Deadline Bottom Level)。

本文对用DAG描述的网格工作流进行分析,提出了一种QoS约束的网格工作流调度算法Q-TWS。算法支持网格任务合并、分层,满足用户的QoS要求。本文第2节描述一个典型的网格工作流,提出了网格工作流任务调度的问题。紧接着给出Q-TWS算法描述,分析了算法的可行性。最后对Q-TWS进行模拟,验证了算法的有效性。

2 网格工作流调度问题描述

网格工作流是一个包含了多个网格任务和任务间依赖关系,并作为一个整体提交给工作流管理器的集合。基于QoS

到稿日期:2008-10-31 返修日期:2009-01-09 本文受国防基础科研项目(C2720061361)资助。
姚 磊(1981-),男,博士生,主要研究方向为网格计算,E-mail:morningyao@gmail.com;戴冠中(1938-),男,博士生导师,主要研究方向为控制理论、复杂网络等;张慧翔(1981-),男,博士生,主要研究方向为Internet拥塞控制;任 帅(1983-),男,博士生,主要研究方向为计算机网络安全、数字水印。

的工作流管理器首先要解析出工作流中的任务及其之间的依赖关系，并按照一定的调度规则把所有的任务映射到资源上，工作流的执行性能要达到用户的 QoS 要求。

一个网格工作流至少要包含两类信息：任务信息和任务依赖关系信息。设工作流中所有任务的集合是 $T=\{t_0,t_1,\cdots,t_n\}$，$|T|=n+1$。E 表示任务之间依赖关系的集合。用 (t_i,t_j) 表示 t_i 和 t_j 之间的依赖关系 $(t_i,t_j)\in E$。其中 t_i 是 t_j 的紧前任务，t_j 是 t_i 的紧后任务。当 t_j 的所有紧前任务执行完毕后，t_j 才能开始执行。用 $\alpha(t_i,R_u)$ 表示 t_i 在资源 R_u 上的执行时间，用 $\beta(t_i)$ 表示执行 t_i 的花费。$W=(T,E)$ 表示一个网格工作流。为使网格工作流调度支持用户的 QoS 要求，需加入对 QoS 的描述。把 QoS 约束分成两个：完成时间约束 D 和预算约束 B。用 $Q_W=(D,B)$ 表示用户对工作流 W 的 QoS 要求。因此 QoS 约束下的网格工作流调度可以形式化地描述如下：

$$\alpha(t_i,R_u)|_{i=n}\leqslant D,R_u\in R \quad s.t.\ \min\sum_{i=1}^{n}\beta(t_i)\leqslant B \tag{1}$$

R 表示全体可用资源。上式说明了一个截止期约束下的最小费用问题，此种问题一般情况下是 NP 难问题，采用启发式方法具有较好的效果。

本文提出了一种双向分层算法 Q-TWS。算法试图把整个工作流的 QoS 约束分解到对单个任务的 QoS 约束，并在此约束下进行最小费用调度。用 $ES(t_i)$ 和 $LF(t_i)$ 表示 t_i 的最早可能执行时刻和最晚必须完成时刻，t_i 必须在时间段 $[ES(t_i),LF(t_i)]$ 内完成并使得执行费用最少。显然，区间 $[ES(t_i),LF(t_i)]$ 越大，对 t_i 的约束就越宽松，t_i 就可以选择速度较慢但费用较低的资源，使得局部的任务执行费用最小，最终满足式(1)的约束条件。

3 算法描述与分析

Q_W 对整个工作流的约束，即整个工作流完成的最晚时间和整个工作流的费用进行预算。Q_W 并不针对工作流中的单个任务，但是网格工作流的任务调度是对单个任务进行管理。如何把用户对工作流整体的 QoS 要求，映射到单个任务的 QoS 要求是网格工作流管理系统首要考虑的问题。这里首先给出工作流双向分层方法，然后给出 Q-TWS 算法的描述。

3.1 双向分层及任务约束时间计算

首先合并工作流中具有顺序执行关系的任务，得到网格工作流 W，采用双向分层方法对所有任务进行分层，尽量放大一个节点的时间约束。下面给出正向层次和反向层次的定义。

定义 1（正向层次，Top Level，TL） 给定工作流 $W=(T,E)$，TL 是从源点到任务 t_i 经过的最多依赖关系数，t_i 的 TL 值计算公式为：

$$TL(t_i)=\begin{cases}0, & t_i=\text{source}\\ \max\limits_{t_p\in pre(t_i)}\{TL(t_p)\}+1, & t_i\neq\text{source}\end{cases} \tag{2}$$

定义 2（反向层次，Bottom Level，BL） 给定工作流 $W=(T,E)$，BL 是最大 TL 值减去从终点到任务 t_i 经过的最多依赖关系数，t_i 的 BL 值计算公式为：

$$BL(t_i)=\begin{cases}\max\limits_{t\in T}\{TL(t)\}, & t_i=\text{destination}\\ \min\limits_{t_s\in succ(t_i)}\{BL(t_s)\}-1, & t_i\neq\text{destination}\end{cases} \tag{3}$$

定理 1 具有相同 TL 值的任务之间没有依赖关系；具有相同 BL 值的任务之间没有依赖关系。

证明：令 $TL(t_i)=TL(t_j)$，并存在依赖关系 (t_i,t_j)，t_i 为 t_j 紧前任务。由定义 2 得 $TL(t_j)-TL(t_i)\geqslant 1$，与定理条件冲突，具有相同 TL 值的任务之间没有依赖关系得证；同理可证，具有相同 BL 值的任务之间没有依赖关系。证毕。

对所有的任务节点，从源点开始依次计算其 TL 值，计算完毕后保存最大的 TL 值。然后，从终点开始依次计算所有节点的 BL 值。经过两轮计算后，每个节点得到对应的一对层次属性：

$$Layer(t_i)=(TL(t_i),BL(t_i)) \tag{4}$$

把 TL 值相同的任务分到同一层，然后在同一层中找到符合下列条件的任务：

$$Length(t_i)=\max_{t\in Lay(TL(t_i))}\{Length(t)\},s.t.\ TL(t_i)=BL(t_i) \tag{5}$$

其中，$Length(t_i)$ 表示任务 t_i 的长度，$Lay(TL(t_i))$ 表示具有相同 $TL(t_i)$ 值的所有任务的集合。这样的节点记作 t^k，上标 k 为此层的 TL 值。记 $MCP_w=\{t^0,t^1,\cdots,t^p\}$，其中 $p=\max\limits_{t\in T}\{TL(t)\}$。$MCP_w$ 就是一个改进的关键路径，其中的任务就是关键任务。由定理 1 知，t^k 与其同层的任务相互独立，故同层的任务可以同步执行。一层任务全部完成的时间早晚取决于具有最长执行时间的任务。由于工作流在未调度时没有为任务指定相应的资源，因此任务的执行时间无法估计。但在同一资源上执行时，长度较大的任务执行时间一定较长，因此可以近似地认为一层任务全部完成时间早晚取决于具有最大长度的任务。t^k 便是符合这一条件的任务。MCP_W 中的所有任务构成关键任务，把所有关键任务作为一个整体，调度到符合下面条件的资源上：

$$\min\frac{\sum\limits_{t^k\in MCP_W}Length(t^k)}{V(R_u)}*C(R_u)$$
$$s.t.\ \frac{\sum\limits_{t^k\in MCP_W}Length(t^k)}{V(R_u)}\leqslant D \tag{6}$$

其中，$V(R_u)$ 和 $C(R_u)$ 分别表示资源 R_u 的计算速度和使用费率。

把用户对 $W=(T,E)$ 的时间约束分割到 MCP_W 中的各个任务上，任务 t^k 的时间约束为：

$$d(t^k)=\frac{\sum_{i=0}^{k}Length(t^i)}{\sum_{t\in MCP_W}Length(t)}*D \tag{7}$$

因此可以得到 MCP_W 中任务 t^k 的最早开始时间 ES、最早可能结束时间 EF、最迟必须完成时间 LF、最迟必须开始时间 LS 如下：

$$\begin{cases}ES(t^k)=\begin{cases}0,k=0\\ EF(t^{k-1}),k\neq 0\end{cases}\\ EF(t^k)=ES(t^k)+\alpha(t^k,R_u)\\ LF(t^k)=d(t^k)\\ LS(t^k)=LF(t^k)-\alpha(t^k,R_u)\end{cases} \tag{8}$$

接下来讨论与 t^k 同一层的其余任务的时间约束。

令 t_i^k 是与 t^k 同一层的一个节点，分两种情况讨论：

(1) $TL(t_i^k)=BL(t_i^k)$

此时，表示任务 t_i^k 与 $K+1$ 层的任务有依赖关系，t_i^k 被限制到在第 K 层内必须完成。故 t_i^k 的时间约束与 t^k 相同，即

在时间段$[ES(t_i^k),LF(t_i^k)]$内。

(2) $TL(t_i^k)\neq BL(t_i^k)$

考虑正向分层。由定理1可知t_i^k与第K层内的节点没有依赖关系;考虑反向分层。设$L=BL(t_i^k)$,由定理1可知t_i^k与第L层的节点无依赖关系。综合以上两点考虑,可知t_i^k与第K到第L层之间的任务也无依赖关系,所以t_i^k的最早开始时间与第K层的最早开始时间相同,t_i^k的最晚完成时间与第L层的最晚时间相同,即t_i^k的执行可以跨越第K到第L层,放松了对t_i^k的时间约束。

$$\begin{cases} ES(t_i^k)=\begin{cases}0, & k=0\\ EF(t^{k-1}), & k\neq 0\end{cases}\\ EF(t_i^k)=ES(t_i^k)+\alpha(t_i^k,R_u)\\ LF(t_i^k)=d(t^l)\\ LS(t_i^k)=LF(t_i^k)-\alpha(t_i^k,R_u)\end{cases} \tag{9}$$

t_i^k的执行时间段约束在区间$[ES(t^k),LF(t^l)]$内。由于$k<l$,并且MCP_W中的任务是顺序执行的,故有$LF(t^k)<LF(t^l)$,因此可得结论:

$$[ES(t^k),LF(t^k)]\subset[ES(t^k),LF(t^l)] \tag{10}$$

即通过双向分层后,对$TL(t_i^k)\neq BL(t_i^k)$的任务,其执行时间段的约束放宽,有利于任务在不影响整个工作流执行的情况下,选择能力较弱但价格较低的资源。

3.2 Q-TWS算法描述

经过以上分析,可把Q-TWS算法描述如下:

输入:用DAG图描述的网格工作流$W=(T,E)$,及其$Q_W=(D,B)$。

输出:对所有任务形成调度策略。

1)合并所有能合并的任务,直到无任务可合并,得到W';

2)对W'进行正向分层,得到所有任务的TL值;

3)对W'进行反向分层,得到所有任务的BL值;

4)对所有节点,具有相同TL值k的节点,加入到$Lay(k)$中,其中$0\leqslant k\leqslant\max_{t\in T}\{TL(t)\}$;

5)对所有$Lay(k)$,根据式(5)找出t^k,得到MCP_W;

6)把MCP_W作为整体调度到式(6)约束下的资源R_u上;

7)把时间约束D分割到t^k,得到$d(t^k)$;

8)对MCP_W中所有任务,根据式(8)计算t^k的时间约束;

9)对所有层次:

对$Lay(k)$中除t^k外的所有任务:

如果$TL(t_i^k)=BL(t_i^k)$,使用式(8)计算其时间约束;

如果$TL(t_i^k)\neq BL(t_i^k)$,使用式(9)计算其时间约束;

10)从源点开始,对所有任务分配一个满足其时间约束且最廉价的资源,分配一个任务后,更新被分配资源的最早就绪时间;

11)所有任务都分配完毕。算法结束。

算法的第2步和第3步的时间复杂度相同,都是$O(n)$。第4步和第5步的时间复杂度相同,都是$O((p+1)n)$,其中p是所有任务的最大TL值。第7步的时间复杂度为$O(p+1)$。第8到10步的时间复杂度为$O(2n)$,故整个算法的时间复杂度为$O((2p+6)n+p+1)$。

4 实验结果

为验证本文提出的Q-TWS网格工作流调度算法,使用GridSim[11]建立表1描述的一个网格系统。要调度的网格工作流使用London e-Science Centre[12]开发,并被文献[13]使用的网格工作流应用,如图1所示。

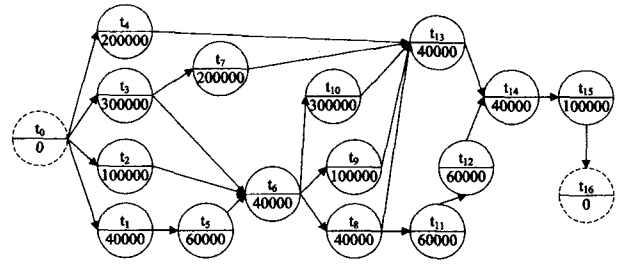

图1 网格工作流应用

网格系统共有5个资源,这5个资源都具备执行工作流中任务的能力。所有资源都是独占式访问的,即同时只能有一任务在资源上执行,并且任务使用资源全部的计算能力。各个资源计算能力的不同,开销的费用也不同,具体参数如表1所列。网格工作流中包含15个实任务,每个任务都用一个圆表示,上半部分是任务编号,下半部分是任务长度,单位是MI,如图1所示。图中的t_0和t_{16}是虚任务,表示工作流的开始和结束。虚任务的长度为0,不占用调度时间和执行时间。

表1 网格资源描述

Resource ID	MIPS Rating	Cost(G$/Sec)	Policy
0	500	2	SPACE_SHARE
1	1000	4	SPACE_SHARE
2	1500	8	SPACE_SHARE
3	2000	16	SPACE_SHARE
4	2500	32	SPACE_SHARE

首先合并工作流中的可合并任务,简化工作流。t_1和t_5可合并,记作$ct_{1,5}$。t_{11}和t_{12}合并后记作$ct_{11,12}$,t_{14}和t_{15}合并后记作$ct_{14,15}$。对所有任务求其TL和BL值并分层后,找到每层的关键任务,结果在表2中列出。

表2 分层结果(任务后的值为其TL,BL值)

k	t^k	$Lay(k)-t^k$
0	$t_0(0,0)$	Ø
1	$t_3(1,1)$	$\{ct_{1,5}(1,1),t_2(1,1),t_4(1,3)\}$
2	$t_6(2,2)$	$\{t_7(2,3)\}$
3	$t_{10}(3,3)$	$\{t_8(3,3),t_9(3,3)\}$
4	$t_{13}(4,4)$	$\{ct_{11,12}(4,4)\}$
5	$ct_{13,15}(5,5)$	Ø
6	$t_{16}(6,6)$	Ø

由分层结果可知,t_4和t_7可跨层执行。t_4的执行时间约束为$[ES(t^1),LF(t^3)]$,t_7的执行时间约束为$[ES(t^2),LF(t^3)]$,其余的任务都被限制到本层中执行。图2显示在$D=500s$和$D=2000s$约束情况下任务在资源上的调度情况。由于t_4长度较本层关键任务t_3短,故这两种情况下t_4均在t_3完成之前完成。但是t_7的长度大于t_6,原有算法中,如果不把t_7分配到一个计算能力很强的资源上,在t_6完成前t_7无法完成。在使用Q-TWS算法后,t_7可以与第3层的任务同时执行,放松了对t_7的时间约束。

图3对比了在不同Deadline约束下TL和Q-TWS的执行时间和执行费用。从图中可以看出,在不同的时间约束下,Q-TWS的执行时间总是小于TL的执行时间,同时Q-TWS的执行费用也小于TL的执行费用。即在同样的约束下,Q-TWS调度算法可以获得较TL算法更好的效果。这是由于Q-TWS算法可以尽可能放松对并行任务的时间约束,使得调

度器可灵活地把此类任务放置到一个满足时间约束但又较廉价的资源执行，获得更好的性价比。

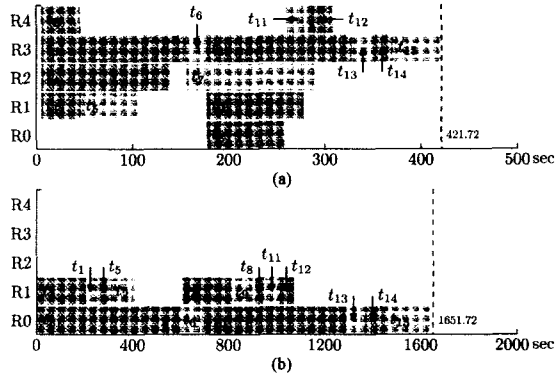

图 2 (a)D=500s,(b)D=2000s 时的任务调度情况

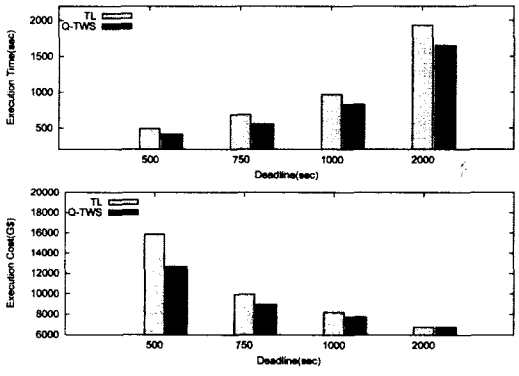

图 3 不同时间约束下 TL 和 Q-TWS 的执行时间和费用比较

结束语 本文提出了一种基于双向分层的网格工作流调度算法 Q-TWS。算法可以找到工作流中的时间约束较松的任务，进而可以对任务进行更灵活的调度，可以提高工作流执行的并行性。实验结果显示了在 Q-TWS 算法调度下，可以有效放宽对特定任务的执行时间约束，提高调度灵活性，使得网格工作流可以在 Deadline 约束下完成并且费用较小。由于本算法使用 DAG 对网格工作流进行描述，DAG 本身不支持任务循环、条件，因此本算法不能直接支持循环和条件。但是对含有循环和条件的工作流，通过放大任务的粒度，把循环放到一个任务中，把条件看作多个任务的并行执行，就可以应用 Q-TWS 算法。

Q-TWS 算法没有考虑资源失效，即没有考虑到资源的动态性。进而由此引起的任务执行失败也没有考虑，这是下一步工作中要重点考虑的问题。另外，如何减少由于任务执行而在资源上产生的时间碎片也是一个需要解决的问题。

参 考 文 献

[1] WfMC. WorkFlow Management Coalition: Terminology&Glossary[S]. WfMC-TC-1011 (Issue 3.0). Feb. 1999

[2] Foster I, Kesselman C, Nick H, et al. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration[C]// Open Grid Service Infrastructure WG, Global Grid Forum, June 2002

[3] Amin K, Hategan M, von Laszewski G, et al. GridAnt: A Client-Controllable Grid Workflow System[C]// 23 37th Hawai'i International Conference on System Science. Island of Hawaii, Big Island, January 2004

[4] Cao Junwei, et al. GridFlow: Workflow Management for Grid Computing[C]//3rd International Symposium on Cluster Computing and the Grid. Tokyo, Japan, May 2003

[5] Yu Jia, Buyya R, Ramamohanarao K. Workflow Schdeduling Algorithms for Grid Computing[M]. Metaheuristics for Scheduling in Distributed Computing Environments. Berlin, Germany: Springer, 2008

[6] Tannenbaum T, Wright D, Miller K, et al. Condor--a distributed job scheduler[M]. Beowulf Cluster Computing with Linux. Cambridge, MA: The MIT Press, 2002

[7] Wieczorek M, Prodan R, Fahringer T. Scheduling of Scientific Workflows in the ASKALON Grid Enviornment[J]. ACM SIGMOD Record, 2005, 34(3): 56-62

[8] Menascμe D A, Casalicchio E. A Framework for Resource Allocation in Grid Computing[C]// The 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'04). Volendam, The Netherlands, Oct. 2004

[9] Yu J, Buyya R, Tham C K. A Cost-based Scheduling of Scientific Workflow Applications on Utility Grids[C]// The First IEEE International Conference on e-Science and Grid Computing. Melbourne, Australia, Dec, 2005

[10] Yao L, Dai G, Zhang H, et al. Guarantee the Victorious Probability of Grid Resources in the Competition for Finite Tasks[C]// The 3rd International Conference on Grid and Pervasive. Kunming, China, May 2008

[11] Buyya R, Murshed M. GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing[J]. Concurrency and Computation: Practice and Experience, 2002, 14: 1175-1220

[12] O'Brien A, Newhouse S, Darlington J. Mapping of Scientific Workflow within the e-Protein project to Distributed Resources [C]//UK e-Science All Hands Meeting. Nottingham, UK, Sep. 2004

[13] Yu J, Buyya R, Tham C K. QoS-based Scheduling of Workflow Applications on Service Grids[C]//Proc. of the 1st IEEE International Conference on e-Science and Grid Computing (e-Science'05). Melbourne, Australia, December 2005

[14] Ranjan R, Harwood A, Buyya R. A case for cooperative and incentive based coupling of distributed clusters[M]. Future Generation Computer Systems, Elsevier Science. The Netherlands, April 2008

[15] Ranjan R, Rahman M, Buyya R. A Decentralized and Cooperative Workflow Scheduling Algorithm[C]// Proceedings of 8th IEEE International Symposium on Cluster Computing and the Grid. France, May 2008

[16] 王璞，彭玲. 一种新的经济网格计算任务调度控制模型[J]. 计算机科学，2008，35(3)：106-108

[17] 陈庆奎. 基于强化学习的多机群网格资源调度模型[J]. 计算机科学，2007，34(11)：67-70

[18] 苑迎春，李小平，王茜，等. 基于逆向分层的网格工作流调度算法[J]. 计算机学报，2008，31(2)：282-290