第32卷 第16期 计 算 机 工 程 2006年8月
Vol.32 № 16 Computer Engineering August 2006
·网络与通信· 文章编号：1000—3428(2006)16—0109—03 文献标识码：A 中图分类号：TP393

基于Globus的有限资源再调度模型R2_Scheduler

苏 培，章 勇，胡云贵
(南京航空航天大学信息科学与技术学院，南京 210016)

摘 要：网格资源调度是网格管理技术中的关键组成部分，文章在研究 Globus 网格资源调度模型的基础上提出了一种基于资源动态性的再调度策略 R2_Scheduler。资源的动态特性以及跨越多个管理域使得网格管理策略必须根据其可用性作出相应调整，R2_Scheduler 通过一个启动因子 p 与用户进行协商，利用前一次调度的结果，在资源有限的情况下启动任务，可以避免因资源不足造成的请求失败；同时利用资源的局部性特点，把请求资源限制在一个协同调度器的作用域之内，从而减少网络通信负载。

关键词：网格计算；再调度；协同调度；启动因子

R2_Scheduler: A Rescheduling Model for Limited Resource Based on Globus Framework

SU Pei, ZHANG Yong, HU Yungui
(Information Science and Technology Institute, Nanjing University of Aeronautics and Astronautics, Nanjing 210016)

【Abstract】 Resource scheduling is the core administration component in grid computing. Based on the Globus framework and technologies, this paper proposes a resource rescheduling model named R2_Scheduler, with resource dynamic characteristics in mind. The scheduling strategies must be adapted according to the resources' dynamic and cross-administration-domain characteristics. According to the preceding scheduling results, R2_Scheduler negotiates with the user who invokes the request to determine whether to startup the task or not, in the condition that resource is lack. The limitation or availability of resource is reflected by a startup-factor. With the agreement of the user, the failure of the request is avoided, and the whole system network load is reduced by restricting the resource in the same co-allocator's management domain.

【Key words】 Grid computing; Rescheduling; Co-allocating scheduling; Startup-factor

网格计算通过将地理上分布的通信体系中的许多计算机组织成集群的方式，提供了一种解决大规模计算的模型，它区别于传统分布式计算的地方是支持跨越多个管理域的计算[1]。资源管理和调度是网格核心管理技术中的关键。目前，各种网格系统实现的调度策略在性能、动态交互和可扩展性方面各有侧重，被业界广泛采用的Globus在任务映射和资源分配方面的设计更多地考虑了通用性，可以作为研究开发某一具体应用的基础。本文对Globus调度模型进行了分析，提出了一种有限资源再调度模型——R2_Schduler(Rescheduling too.另一种形式的再调度模型)。

1 网格系统中的调度和再调度

1.1 一般调度问题

文献[4]中给出了调度的一个形式化定义：

一个在由 m 个处理单元和任务图 G=(T,E)之上的调度是一个函数 f，

$f: T \rightarrow \{1,2,\ldots,m\} \times [0,\infty]$

如果存在$v \in T$，$f(v)=(i, t)$，则表示任务v被调度到处理单元Pi上，且从时间t开始执行[4]。

1.2 网格调度和再调度

按照 GGF 对网格调度的定义，网格调度是一个软件框架，调度器负责收集资源状态信息，选择合适的资源，对候选调度进行性能预测并根据性能目标或者用户应用程序需求选择认为最好的调度执行策略。

资源的再调度的触发因素大致有如下几种：

(1)分配给任务的某些资源节点失效；
(2)任务调度后长期得不到执行；
(3)资源特性改变导致任务执行异常；
(4)用户请求发生变化；
(5)资源节点管理策略改变；
(6)现有调度算法实时改进。

在目前的调度系统中对再调度有不同的实现[2]，而且大多做了很大程度的简化。出现这种情况的原因，除了技术上的限制，一个重要因素还在于进行再调度所付出的计算和通信代价很高，因为需要在全局内进行资源信息的收集、性能预测和调度策略再选择以及任务的迁移、中间数据的整合等。本文提出的局部再调度策略试图在附加一些约束条件的前提下避免再调度的计算和通信开销。

1.3 Globus 资源调度模型

Globus Toolkit 的核心组件旨在解决和安全、资源访问和管理、数据迁移和管理、以及资源发现等领域相关的基本问题。可以在 Globus 框架的基础上设计自己的应用框架。在文献[3]中，Karl Czajkowski 和 Ian Foster 等基于 Globus 体系提出了一个资源管理体系结构。在一个元计算系统中存在多个语义上不同的 Co-allocator，负责跨越多个域的资源协同，当需要再调度时，由资源代理负责在多个 Co-allocator 之间进行

作者简介：苏 培(1979—)，男，硕士生，主研方向：分布式计算，系统集成；章 勇，副教授；胡云贵，硕士生

收稿日期：2005-11-21 **E-mail：**supei.2003@163.com

资源映射和调度，即全局的再调度。

一个任务请求在 GRAM 中可能的状态以及状态转换由图 1 表示。

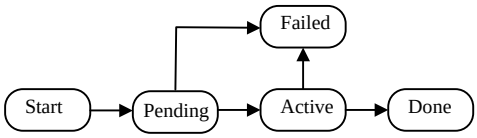

图 1 GRAM 中任务的状态转换

根据GRAM选择的本地调度算法的不同，对处于挂起状态的任务请求可能有不同的处理，但是，由于GRAM的上层Co-allocator对任务的调度是一个原子操作[5]，如果一个任务请求涉及多个GRAM中的请求长期处于挂起状态，则一方面会造成响应延迟，降低服务质量；另一方面也存在资源浪费，影响整个系统的吞吐量。

针对以上两方面问题，即再调度的全局性和 GRAM 调度的原子性，本文在第 2 节提出了一种有限资源再调度模型 R2_Schduler，旨在把再调度的作用范围尽量缩小，以减少网络负载；以及打破 GRAM 调度的原子性，及时提供可用资源信息供上层调度。

2 有限再调度模型 R2_Scheduler

2.1 R2_Scheduler 设计思想

R2_Scheduler 调度模型是一种再调度技术。和其他的再调度策略的根本不同在于，它利用了前一次资源调度的结果，当可用资源少于用户请求的数量时，经过和用户协商，可以在少量资源的情况下启动任务。下面是 R2_Scheduler 的一个形式化定义。

R2_Scheduler 调度是一个三元组：

R2S<C, p, G>

其中，C=<mr, loc, dl, rt, dy,hp>；p=L/R；G=<T,E>。

第 1 个元素 C 是 R2_Scheduler 资源再调度的约束条件。目前考虑了以下 6 个约束项：

(1)mr：multi-resource。假设用户的任务请求涉及多个资源节点。

(2)loc：locally。用户对资源的请求具有局部性，即资源节点的分布相对集中。

(3)rt：response time。用户请求更关注对任务请求的及时响应。

(4)dl：deadline。用户请求的任务对完成时间无特殊严格的限制。

(5)dy：dynamic。资源是有限的，并且其可用性可能变化频繁。

(6)hp：high pay。在多个管理域内进行资源映射和再调度的代价很大(在这种情况下 R2_Scheduler 可以称为轻量级的再调度策略)。

第 2 个元素 p 称为启动因子。表示当前可用资源 L 和用户请求资源 R 之间的比值。它反映了调度算法在资源有限的情况下响应用户请求的能力。

第 3 个元素 G 是一个二元组，表示一个资源调度模型中的网络拓扑。其中 T 是节点的集合(节点可以是用户、资源代理、协同分配器以及资源控制节点等)；E 是有向边的集合，表示节点之间的控制和交互。图 2 是 R2_Scheduler 再调度模型的网络拓扑图。其中虚线表示初始调度交互，实线表示再调度交互。假设用户任务请求 Ti 需要 N 个资源，经第 1 次调度完成资源映射后，资源的可用性发生变化，假设只有 K(K<N)个资源可用，这时需要再调度。R2_Scheduler 使用可用资源监控模块，同用户协商，确定是否可以在只有 K 个资源可用的情况下启动任务，如果用户同意，则不必抛弃前一次调度的结果，并且可以只在一个协同分配器管理域内进行任务分配。下面详细介绍其调度策略。

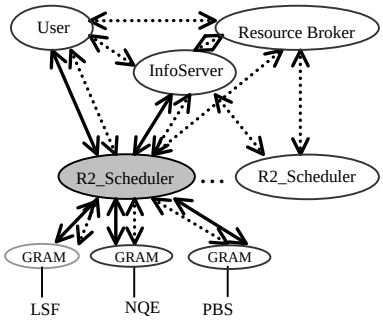

图 2 R2_Scheduler 体系结构

2.2 R2_Scheduler 的调度过程

完整的资源调度过程分两个阶段：初始调度和再调度。通过算法 *Algorithm_R2* 描述这一过程。

有限资源再调度算法 *Algorithm_R2*：

(1) query-info_server
(2) send-request-to-resource_broker
(3) if (request multi-resource)
MapOnMultiR2Scheduler(loc, hp);
(4) r2scheduler-MapGRAMs(dl, rt)
(5) if (当前可用资源个数 L= =请求资源个数 R)
转 (7);
else{
r2scheduler 把当前 L 个资源的状态置为 reservered(占有);
构造一个协商信息表 nig-inof-list ; }
(6) r2scheduler 与 user 协商
if(user-agree= =true){
转 (7);
}else{
report-request-fail;
return-failed;}
(7) r2scheduler 在 L 个资源上启动任务并监视其运行情况
report-to-InformationServer;
while(true){
if(task-state = =done){
result-collection;
report-log;
return-done;} }

步骤(1)～步骤(4)，用户通过查询和发送任务请求，由资源代理 resource_broker 根据用户对响应时间 rt 和完成时间 dt 的要求先在多个调度器 R2Scheduler 之间进行初始调度。在步骤(5)中判断资源可用情况，若有足够的资源，则转步骤(7)进行资源分配和任务执行；否则 R2Scheduler 先对当前 L 个可用资源进行先占，然后在步骤(6)中与用户进行协商，若用户同意在 L 个资源上启动任务，则转步骤(7)进行任务执行和结果收集等；否则本次请求失败，整个调度过程结束。

3 R2_Scheduler 模拟实现

根据以上设计思想，参考 Globus Toolkit 的体系结构，在网格调度模拟器 GridSim 环境下对以上算法进行了模拟实现。GridSim 模拟器本身使用 Java 语言开发。提供了一些基本的调度接口，用户可以在此基础上编写自己的调度算法并进行模拟和结果分析。

实现中设计了以下关键类：

UserNigotiation(int R2SchedulerID,

ResInfoList resList, int resNumber)：R2_Scheduler 与用户进行协商的类，提供参数分别为调度器 ID，当前资源信息列表，资源个数。

GetCurrentAvailableResource(int timeout)：获得当前可用资源信

息。超时值根据各个 GRAM 使用的本地调度算法和用户对响应时间的接受程度进行综合评估来确定。

GRAMAttemptScheduling(int timeout)：本地 GRAM 进行可用资源的预分配，由 GetCurrentAvailableResource()调度。

CalculateDelay(int R2SchedulerID)：计算使用 ID 为 R2SchedulerID 的再调度策略时任务完成时间的延迟量。

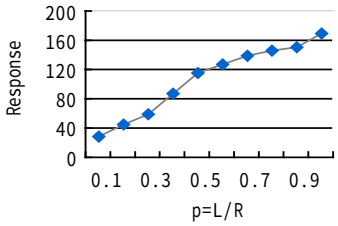

图 3 响应时间随资源变化趋势

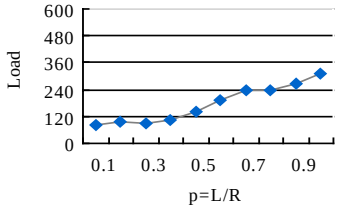

图 4 网络负载随资源变化趋势

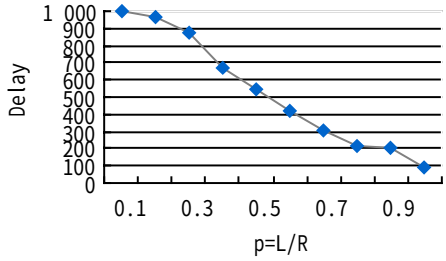

图 5 任务完成时间延迟随资源变化趋势

通过重载 GridSim 相应模块中的接口，对 R2_Scheduler 各个调度模块进行封装，并把结果数据重定向。以启动因子 p 为自变量，对响应时间、网络负载、以及任务完成时间延迟 3 个因素进行数据过滤，结果分别如图 3、图 4 和图 5 所示。

在用户请求可以调整的情况下，由分析数据可以得出以下 3 个重要结论：

(1)采用有限资源再调度策略可以获得较快的响应时间。当前可用资源越少，响应越快；而多数基于 Globus 的调度只有在 p=1 时，即有足够的资源的情况下才响应用户请求。

(2)可以避免在全局内进行通信，从而减轻了网络负载。资源的分散程度越高，则跨越多个管理域通信的可能性也越大，因此资源个数与通信量大致成反比关系，启动因子与通信量成正比关系。

(3)任务完成的时间被延迟。延迟的程度与 p 成反比关系，p 越小，延迟越严重。

4 总结与展望

本文针对网格系统中资源的动态特性，基于 Globus 提出了一种有限资源再调度模型 R2_Scheduler。通过调整调度请求，在启动因子 p<1 的情况下响应请求。作为已有调度系统的可选方案，适用于对用户请求快速响应的动态资源网格环境。对于并行性良好的任务和资源，更加完善的资源再调度机制，可考虑支持资源的动态参与，以缩短任务的运行周期。

参考文献

1 Foster I, Kesselman C. The Grid: Blueprint for a New Computing Infrastructure[M]. Morgan Kaufmann, 1998-07: 2-3.

2 Braun T D. A Taxonomy for Describing Matching and Scheduling Heuristics for Mixed-machine Heterogeneous Computing Systems[C]. Proc. of IEEE Symposium on Reliable Distributed System, 1998: 330-335.

3 Czajkowski K, Foster I. A Resource Management Architecture for Metacomputing Systems[C]. Proc. of IPPS/SPDP'98 Workshop on Job Scheduling Strategies for Parallel Processing, 1998: 62-82.

4 朱福喜，何炎祥. 并行分布计算中的调度算法理论与设计[M]. 武汉：武汉大学出版社, 2003.

5 都志辉，陈　渝，刘　鹏. 网格计算[M]. 北京：清华大学出版社, 2002.

(上接第 104 页)

3.2 发送数据聚类中心分析

3 月 13 日是个休息日，网络在 22~4 点和 12~16 点出现了使用高峰期，符合学生的网络使用习惯。3 月 15 日是个工作日，网络在 10~13 点和 20~23 点出现了使用高峰期，也符合学生的网络使用习惯。从以上 2 张图看，各聚类中心间的距离很小，反映在图形上，即曲线出现了很多交叠。

表 4 3 月 13 日发送数据样本的聚类结果

类别	1	2	3	4	5	6
样本数	2	208	5	19	3	1

表 5 3 月 15 日发送数据样本的聚类结果

类别	1	2	3	4	5	6
样本数	9	25	8	3	238	6

从表 4 和表 5 的聚类结果看：分别有 87.39%和 82.35%的样本都聚类入了同一类，即它们有相同的网络使用模式；其余样本对网络的使用习惯不同，这说明使用该 IP 的学生有不同的使用习惯，这就是该日网络使用模式的“奇点”。将每日网络使用模式的“奇点”看作一个事务，采用关联规则算法，可以得到个别 IP 的集合是一定支持度下的频繁集（Frequent Item Set），即他们的网络使用习惯“长期”与众不同。由此可见，上述的聚类算法还可用于“奇点”的自动检测。

参考文献

1 Kohonen T. Self-organized Formation of Topologically Correct Feature Maps[J]. Biological Cybernetics, 1982, 43 (1): 59-69.

2 Kohonen T. Self-organized and Associative Memory (3nd Edition)[M]. New York: Springer-Verlag, 1998

3 Kononen T. The Self-organizing Map (2nd Edition)[M]. Berlin: Springer-Verlag, 1997.

4 Kononen T. Things You Haven't Heard About the Self-organizing Map[C]. Proceedings of the IEEE International Conference on Neural Networks, San Francisco, 1993: 1147-1156.

5 Simon H. Neural Networks, A Comprehensive Foundation (2nd Edition)[M]. Prentice Hall, 1998: 443-483.