一种网格环境下作业混合调度的策略

吴 昊[1], 冯小辉[1], 李长刚[2], 高承东[3]

(1.九江职业技术学院 电气工程系, 江西 九江 332007;
2.桂林工学院 电子与计算机系, 广西 桂林 541004;
3.解放军理工大学 气象学院, 江苏 南京 211101)

[摘要] 网格作业调度是网格平台以尽可能高效的方式将应用程序提交的计算任务分配到恰当的计算资源上执行的过程. 为此分析了网格作业的调度目标与资源分配策略, 并在此基础上给出了一个综合的调度算法, 以及一个仿真实验结果.

[关键词] 网格, 作业调度, 资源分配
[中图分类号] TP 301.6 [文献标识码] A [文章编号] 1672-1292(2008)04-0035-04

Research on a Mixed Job Scheduling Strategy in Grid Environment

Wu Hao[1], Feng Xiaohui[1], Li Changgang[2], Gao Chengdong[3]

(1. Electrical Engineering Department of Jiujiang Vocational and Technical College, Jiujiang 332007, China;
2. Electronics and Computer Department of Guilin University of Technology, Guilin 541004, China;
3. Institute of Meteorology, PLA University of Science and Technology, Nanjing 211101, China)

Abstract Grid job scheduling is the process where the platform more efficiently distributes computing tasks of applications to proper computing resources. The paper analyzes the goals and strategies of grid job scheduling, and baded on it puts forward a comprehensive job scheduling algorithm, and also gives a simulation result.

Key words grid; job schedule; resource distribute

网格[1]是构筑在 Internet基础上的一组新型技术, 对于由地理分布、组织独立的计算资源、存储资源、数据资源、信息资源、知识资源、专家资源等组成的动态虚拟组织的资源实现共享和协作问题的求解. 作业调度和数据管理技术上是网格资源管理的重要部分[2]. 资源管理主要解决资源的定位、分配、认证、进程创建以及其它使用资源的准备活动; 作业调度主要解决任务分解、指派和安排任务执行顺序的问题.

作业方式是使用网格资源的一种形式, 它根据用户确定的流程, 为用户提供使用资源的功能. 网格作业一般都在远端节点上进行, 作业提交者对远端设备的控制能力是非常有限的, 为了有效管理作业的运行, 就需要网格作业管理机制, 管理整个作业的运行过程. 在网格计算平台下, 作业管理机构还要具备作业迁移管理、作业任务分解等一些特殊的功能.

Globus Toolkit与开放网格服务体系结构(OGSA)[3]提供了基于标准且功能强大的网格服务框架. 在GT3提供的服务中[2], 主受控作业工厂服务(MasterManaged Job Factory Service, MMJFS)称为作业管理器. MMJFS提供了单一的接口, 用它来请求和使用远程系统的资源执行作业. 这一组接口称为 Globus资源分配管理器(Globus Resource Allocation Manager, GRAM), 它的设计目标是为调度系统提供灵活的接口. 可以将它想像成具有某些特性的远程 shell.

作业调度程序是一种网络化的系统, 用于提交、控制和监视一台或多台计算机上的批处理作业的工作量. 子系统根据可用的策略和资源的可用性, 安排在某一选定时间执行作业或任务.

1 作业调度

在计算机操作系统中, 我们已经熟悉了作业调度的概念, 它的调度机制是通过时间片轮转. 这种轮流

收稿日期: 2008-06-18
基金项目: 国家自然科学基金(60403043)资助项目.
通讯联系人: 李长刚, 硕士研究生, 研究方向: 网格计算. E-mail: licg0115@163.com

调度的本质实际上就是轮流获得使用处理器资源的权利, 但在网格环境中, 作业和资源的数目都比较大, 作业和资源之间的匹配也比较复杂. 由于网格资源具有自治特性, 网格作业调度不能剥夺资源本地对作业的管理权利, 所以它与资源本地的作业调度要协同工作, 共同完成网格用户提交的网格作业的调度, 为网格用户充分利用计算资源提供方便.

1.1　作业调度目标

网格作业调度的目标与分布式计算系统、集群系统的作业调度目标是相似的, 在这些系统中, 衡量调度性能的指标包括资源利用率、算法复杂度、完成时间等. 在网格系统中, 高性能仍然是人们追求的目标之一.

作业完成时间是作业在特定节点上执行完成的时刻. 作业执行时间是作业从开始执行第一条指令到执行完最后一条指令所经过的时间. 用表达式来表示有: $T_{\text{exec}} = T_{\text{finished}} - T_{\text{start}}$

其中, T_{exec} 是作业执行时间, T_{finished} 是作业执行结束的时刻, T_{start} 是作业开始执行的时刻.

假设有 m 个作业 $J = \{j_0, j_1, \cdots, j_{m-1}\}$, 需要调度到 n 个资源 $R = \{r_0, r_1, \cdots, r_{n-1}\}$ 上运行, 作业 j_t 在 r_q 上的执行时间为 e_{tq}, 在 j_t 开始执行之前的等待时间为 f_q, f_q 是在映射 j_t 之前所有已经映射到 r_q 的任务的最早完成时间. 一种调度策略总是对应一种资源匹配模式和一种作业执行次序, 任务调度的目标就是求得这样一种资源匹配模式和一种作业执行次序, 使得所有作业的总完成时间最短, 即有

$$\min\left\{\sum_{\substack{t\in\{0,1,\cdots,m-1\}\\ q\in\{0,1,\cdots,m-1\}}}(e_{tq}+f_q)\right\}$$

由于网格资源可提供的本地作业管理机制不同, 网格作业调度器向本地作业管理器提交作业的形式可以分为如图 1所示的 3种形式.

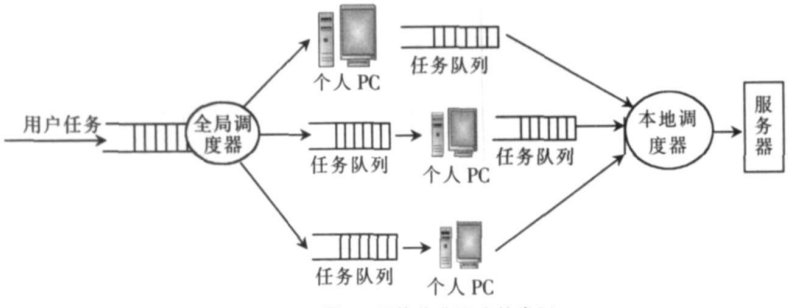

图 1　网格作业调度的类型

Fig.1　Type of grid job scheduling

①一个资源可接收的作业数目不受限制, 作业本地的作业管理器需要维护一个足够长的作业队列, 需要容纳下提交给该资源的所有作业, 网格作业管理器不需要为该资源维护作业队列.

②一个资源可以最多接收不超过规定数目的 n 个网格作业. 在这种情况下, 资源本地要维护一个长度不超过 n 的网格作业队列, 如果分配给该资源的作业数目不超过规定的数目 n, 所有分配给该资源的作业都进入本地维护的队列.

③一个资源一次只接收一个网格作业, 该作业接收的一个作业运行完成之后, 再从网格作业管理器那里接收新的作业.

用户把作业提交给网格作业管理器之后, 作业进入网格作业管理机构维护的作业队列. 网格作业调度模块从作业队列中选择合适的作业分配到合适的资源上运行. 在网格作业的整个生命周期中, 作业要经历不同的作业状态. 网格作业可能离开一个队列, 然后进入另外一个队列, 由于某个事件的发生, 作业状态从一种状态变为另外一种状态, 作业调度模块负责管理作业从一种状态改变为另一种状态的过程. 其状态转换图如图 2所示.

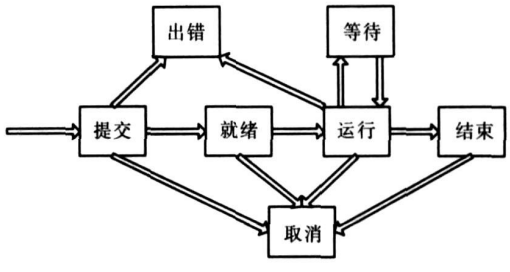

图 2　网格作业状态转换图

Fig.2　Conversion chart of grid job state

1.2　资源分配策略

为使数据通过网格作业管理器在系统中协调流动, 首先要对数据资源进行合理分配, 防止某个结点瓶颈的发生. 资源分配需要借助于合理的资源描述来实现. 它有很多策略, 比较常用的有最先匹配、随机匹配、最优匹配等[4]. 但它们都是从多个资源中选择一个合适的资源策略, 所以比较简单.

假如请求者提出的资源需求用一个 m 元组表示 $(g_0, g_1, \cdots, g_{m-1})$, 系统中节点 k 的资源也用 m 元组表示 $(g_0^k, g_1^k, \cdots, g_{m-1}^k)$, 资源分配过程实际就是寻找一个节点 n, 使得 $g_i^n \geqslant g_i\ (0 \leqslant i \leqslant m-1)$ 成立.

这里的"$\geqslant$"表示资源之间的一种比较关系, 它是一种偏序关系. 如果有"$A \geqslant B$"则说明如果 B 满足某个请求, A 一定能满足该请求. 反过来就不一定成立了. i 是资源的一个编号, 不管资源采用什么样的实际编号, 对一个有限资源集合中的所有资源总可以用从 0开始的正整数对其进行编号, 集合中的 m 个资源的编号分别就是 0, 1, …, $m-1$.

请求寻找合适的匹配资源时, 可以每次都从 0号资源开始匹配, 也可以从任意的一个随机编号开始, 还可以从上次查找的结束点开始, 不同开始点的选择会影响资源匹配的效果[5].

① 随机匹配. 在查找到的满足请求的众多资源中, 随机挑选一个分配给请求者. 这种策略的优点是可以部分避免不均衡的情况发生, 缺点是不能有效地匹配用户需求和资源.

② 最先匹配. 在查找资源的过程中, 把最先找到的资源分配给请求者. 这种策略的优点是处理简单, 不需要等到所有可用资源都调查清楚就可以快速响应请求. 缺点是可导致一些资源负担过重, 而另一些资源可能无事可做, 并且不能有效地匹配用户需求和资源, 容易造成资源浪费.

③ 最优匹配. 找到集合中所有满足请求的资源, 从中找出浪费最少的资源分配给请求者. 也就是找到一个资源 m, 使得 $\begin{cases} g_i^n \geqslant g_i\ (0 \leqslant i \leqslant m-1) \\ g = \sum_{i=0}^{m-1} (g_i^n \geqslant g_i) \text{ 为最小} \end{cases}$ 成立.

1.3　作业调度策略

将资源通过以上 3种资源分匹策略之一分配给网格作业管理器之后, 接下来还要选择一种合适的作业调度策略, 来对这些网格作业进行合理调度, 提高作业执行效率. 在网格作业调度中, 经常用到以下几种调度策略, 下面简单的介绍两种比较重要的调度策略算法.

① Min-min算法: Min-min算法首先将需要调度的作业组成一个作业集合, 然后计算出作业集合中每个任务的最短完成时间. 调度时, 从作业集合中选择有最短执行时间的那个作业分配到相应的资源上, 同时从作业集合中删除这个作业. 然后开始新的调度, 重复该过程直到作业集合为空.

② Sufferage算法: Sufferage算法的原理是, 一个资源将被分配给这样的一个作业, 如果该作业不分配到该节点上, 将会蒙受最大的损失. 该算法中每个作业都有一个 sufferage值, 定义在该任务的最好完成时间和它的次好完成时间. Sufferage值高的作业有优先权, 每个作业的最好完成时间是针对给定的主机计算的.

本文根据以上两种算法的工作原理, 并结合实验过程的探究提出了一种新的调度算法, 我们称之为 Le算法. 其原理为: 根据 Min-min算法, 计算出所有作业的平均完成时间 E_t, 与根据 Sufferage算法得出的 Sufferage值 S_t 相比较, 即如果 $S_t > E_t$, 则选择 Min-min算法, 反之选择 Sufferage算法.

该算法的描述如下:

for作业集合 T中所有的作业 job_k
　for所有网格节点 $node_j$
　　$c_{ij} = e_{ij} + r_j$
　　do until T 为空
　　计算 T 中作业的平均完成时间 E_t,
　　找到具有最早完成时间的作业 job_k, 其完成时间用 job_{kt} 表示
　　和次早完成时间 job_q, 其完成时间用 job_{qt} 表示
　　Sufferage- value $S_t = job_{kt} - job_{qt}$
　　if $(S_t > E_t)$
　　分配作业 job_k 到节点 n_l

T 中删除作业 job_k
if $(S_t < E_t)$
for 寻找具有最早完成时间的网格节点 n_j
if n_j 没有指派
指派 job_k 给 n_j, 从 T 中删除 job_k, 标记 n_j 为已经指派
else
if 已经指派 n_j 的 job_i 的 sufferage value 小于 job_k 的 sufferage value
取消 job_i 的指派, 把 job_i 放回 T 中, 指派 job_k 给 n_j, 从 T 中删除 job_k
endfor
更新 r_l
更新所有的 c_{il}
enddo

算法中的 e_{ij} 是期望执行时间, 表示作业 job_i 在节点 $node_j$ 没有负载的情况下分配 $node_j$ 执行 job_i 的时间. c_{ij} 是期望完成时间, 表示 $node_j$ 执行完 job_i 的时间. r_j 表示当前时刻节点 $node_j$ 执行任务的期望时间.

2 数值结果

我们使用 SPNP 软件进行 SHLPN [5] 模型性能分析. 数值分析中系统参数值根据实际的计算网格系统进行选取. 为了简化模型求解, 不失一般性, 这里给出的数值例子仅考虑了具有 2个节点的计算网格系统.

图 3显示了处理任务的吞吐量 (单位为: 任务 /s) 随着全部任务生成速率 (其任务分别以调用不同的算法来生成) 的增加而变化的情况.

结果分析: 当任务生成速率较低时, 应用于 Min-min 算法, 有比较高的吞吐量. 当任务生成速率很高时应用于 Sufferage 算法较好, 但在一般情况下采用 Le 算法, 是最好的折中方式. 这时即有着较快的任务生成速度, 而又能实现很好的处理任务的能力.

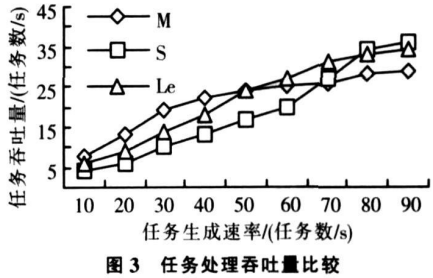

图 3 任务处理吞吐量比较

Fig.3 Comparison of handled task throughput

3 结语

论文讨论了几种调度策略和算法, 并在网格仿真软件 SPNP 中通过建立数据网格模型对它们的性能进行了测试. 结果表明, 不同策略的选择会影响网格资源的利用率以及网格作业的吞吐量! 而且在不同的数据网格场景中 (采用不同的资源分配策略) 其性能也有所变化. 但不同资源的分配策略与不同算法相结合的具体细节影响还有待于进一步研究.

[参考文献] (References)

[1] 都志辉, 陈渝, 刘鹏. 网格计算 [M]. 北京: 清华大学出版社, 2002: 3-5.
Du Zhihui, Chen Yu, Liu Peng. Grid Computing[M]. Beijing: Tsinghua University Press, 2002: 3-5. (in Chinese)

[2] 徐志伟, 冯百明, 李伟. 网格计算技术 [M]. 北京: 电子工业出版社, 2004: 119-120, 231-241.
Xu Zhiwei, Feng Baiming, Li Wei. Grid Computing Technology[M]. Beijing: Publishing House of Electronics Industry, 2004: 119-120, 231-241. (in Chinese)

[3] Forster C. Kesselman. The anatomy of the grid: enabling scalable virtual organizations international [J]. Supercomputer Applications, 2001: 23-25.

[4] 杨永健, 孙永雄, 李树秋, 等. 网格计算中一种负载均衡聚类匹配迁移算法 [J]. 微电子学与计算机, 2006, 23(10): 121-123.
Yang Yongjian, Sun Yongxiong, Li Shuqiu, et al. Migrating algorithm matched with hierachical of load balancing in grid computing[J]. Microelectronics & Computer, 2006, 23(10): 121-123. (in Chinese)

[5] 单志广, 林闯. 计算网格任务调度的随机高级 Petri网模型与分析 [J]. 系统仿真学报 (增刊), 2007, 19: 200-202.
Shan Zhiguang, Lin Chuang. Modeling and analysis of task scheduling in computational grids using stochastic high-level petri nets[J]. Journal of System Simulation(), 2007, 10: 200-202. (in Chinese)

[责任编辑: 孙德泉]