一种网格资源动态分配算法研究*

刘林东[1,2]，陈宏滨[3]
（1. 广东第二师范学院计算机科学系，广东 广州 510303；
2. 华南理工大学计算机科学与工程学院，广东 广州 510006；
3. 桂林电子科技大学信息与通信学院，广西 桂林 541004）

摘 要：在分布式集群环境中，如何对网格环境中的资源进行有效管理和合理调度至关重要。采用静态固定资源分配等策略不能适应资源和用户请求的动态变化，容易产生资源碎片，造成网格资源利用率低等问题。提出了一种基于分类挖掘的资源动态分配模型和算法，通过资源管理服务器中的守护进程，对集群中的任务动作进行分类挖掘，形成分类规则，用以指导资源的动态分配。实验证明，相比其他分配策略和算法，DRA 算法能较好地适应网格环境的变化，具有资源分配利用率高等优点。

关键词：分类挖掘；网格；资源；动态分配；集群
中图分类号：TP391.9 **文献标志码**：A **文章编号**：0529－6579（2013）02－0047－05

An Algorithm of Dynamic Resource Allocation in Grid Environment

LIU Lindong[1,2], *CHEN Hongbin*[3]
(1. Dept. of Computer Science, Guang Dong University of Education, Guangzhou 510303, China;
2. School of Computer Science & Engineering, South China University of Technology, Guangzhou 510006, China;
3. School of Information and Communication, Guilin University of Electronic Technology, Guilin 541004, China)

Abstract: It is important to manage and allocate the resources effective in a distributed cluster environment. Static resource allocation strategy can not meet the requirements of resources and requests' dynamic changes. It can produce some resource fragmentation in the resource pool, resulting in low utilization of grid resources. Classification mining models and algorithms about dynamic resources allocation are proposed. There is a process in every resource management server, mining the classification rules in the cluster based on history data and executing resource allocation based on these rules. Experiments show that DRA algorithm can better adapt to the changes in the grid environment compared to other allocation strategies and algorithms, and it can increase resources utilization in dynamic environment.

Key words: classification data mining; grid; resource; dynamic resource allocation; cluster

网格计算是利用分布式集群环境中一些闲置的计算能力来解决复杂问题的计算模式[1]。通过实现网络资源的全面共享与协同工作为用户提供服务。网格资源的分配是网格计算研究中的重要内容。网格资源分配是在满足用户需求的前提下，将网格任务分配到集群环境中合适的计算资源上，使任务分配的时间尽可能少且资源利用率尽可能高[2]。

在传统的静态资源分配过程中，无法根据请求任务的特征以及任务的动态变化灵活分配资源；当

* **收稿日期**：2012－09－28
基金项目：国家自然科学基金资助项目（61162008）；广西自然科学基金资助项目（2011GXNSFB018072）
作者简介：刘林东（1978 年生），男，讲师；E-mail：hongox@163.com

有用户任务请求以及资源加入或退出时，无法动态对资源分配进行调节。因此容易形成资源碎片、任务请求等待时间过长、资源被长期占用、资源闲置等问题，从而造成资源利用率低下，达不到网格资源分配的合理要求。针对以上问题，通过对资源动态调整的历史信息进行分析，得出基于网格用户的任务动态调整分类规则，抽取分类规则中任务动作标识，后续的任务根据动作类别完成相应的调度，在此基础上形成一种基于分类挖掘的资源动态分配模型和算法。

1 资源管理

资源管理是网格的核心，为资源提供者匹配相应的资源请求并控制他们存取和使用资源等方面发挥着重要的作用[3]。网格资源包括计算、资源、网络等各种类别，如：高性能计算机、工作站、数据库、存储器、CPU、网络带宽等资源。资源管理一般包括资源发现与选择、资源分配以及资源管理优化三个方面。文中主要阐述第二方面内容，并探讨一种资源动态分配模型与算法。

1.1 资源分配

在网格环境下，资源分配是个关键问题，也是个 NP 问题。资源分配的基本因素主要包括网格用户的任务、网格资源以及资源分配策略。在分布式集群环境中，网格资源分配策略一般包括静态资源分配、资源协同分配和资源再分配等几种策略，各种分配策略主要从响应时间、资源利用率以及吞吐量等三个指标来衡量，其中典型的有 FCFS、SCOAL、FlexCo 等算法[4-5]。好的资源分配策略对于提高网格计算的性能和充分利用闲置计算能力是非常重要。

资源动态分配是指根据资源池中的资源以及用户请求任务的情况实时调整资源分配策略，从而保证资源利用率尽可能高。是在弹性资源分配和资源协同分配的基础上实现的，实现资源的动态分配，主要为了解决资源分配过程中所产生的碎片以及提高资源的利用率[6]。常见的动态资源分配策略包括：GARA、FLexCo、HARC、SQ 等[7]。

1.2 资源分配产生的问题

在资源分配过程中，考虑单一集群的情况，设初始状态包括有 3 个 CPU 资源 r_1，r_2，r_3 以及 3 个网格用户 u_1，u_2，u_3，每个用户包括一个任务请求 J_{11}，J_{21}，J_{31}，资源分配过程中容易产生如图 1 所示的几种情况。图 1（a）为 t_1 时刻资源分配初始情况；图 1（b）表示在 t_2 时刻 3 个用户的 3 个新的任务请求，显然 t_1 和 t_2 时刻间的资源是空闲的；图 1（c）表示 t_1 时刻任务完成后，u_2 没有新的任务请求；图 1（d）表示 t_1 时刻任务完成后新用户 u_4 的 J_{41} 任务加入到请求队列中。同时 CPU 资源的数量也可能动态的变化。因此在资源分配过程中必须要充分考虑各种可能出现的情况，采取相应的策略对资源进行分配，使资源利用率达到最大。

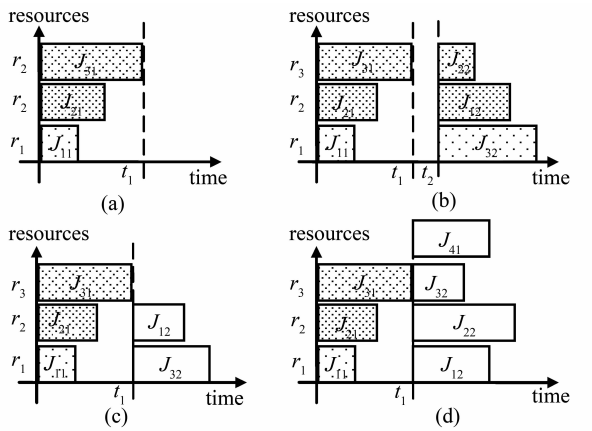

图 1 动态资源分配任务图

Fig. 1 Task graphs of dynamic resources allocation

2 资源动态分配策略

2.1 基于分类挖掘的资源动态分配模型

基于分类挖掘的资源动态分配模型如图 2 所示，包括用户层、资源管理层以及资源层三层体系[8]。用户层中的网格用户向网格服务提供者发送任务请求，由网格中的主节点接收用户请求，任务完成后，发送相应的消息给用户；资源管理层负责资源的管理和调度，包括集群中的主节点以及各个集群中的一台资源管理服务器 S_1，…，S_n，其中主结点负责将用户请求根据相应的策略分配给相应的集群和节点，资源管理服务器负责当前集群中的分类挖掘以及与主节点、其它资源管理服务器交换消息；资源层由 n 个集群 Cluster_1，…，Cluster_n 构成，每个集群中包括具有相应计算能力的节点。

在资源管理层中构造两层分类挖掘。其中主节点通过对用户请求任务的历史信息进行分类挖掘，得出相应的经验模式，将任务分配至相应的集群；资源管理服务器中的分类模块负责对当前集群中的资源动态分配进行挖掘。

2.2 资源动态分类

为了在资源管理服务器中对当前集群中的资源进行合理分配，需要借助分类挖掘模块对资源和用

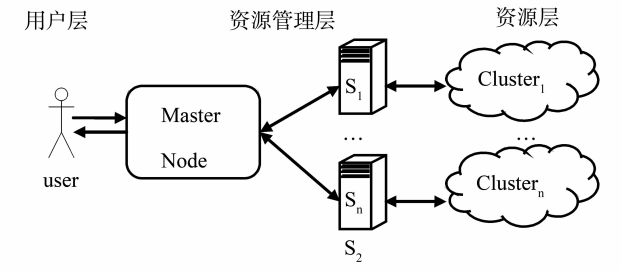

图 2　基于分类挖掘的资源动态分配模型

Fig. 2　Dynamic resources allocation model based on classification mining

户请求所发生的变化进行挖掘，产生指导资源分配的分类规则。每个资源信息由 r（id，r_s，r_e，$state$，in，out）构成，其中 id 表示资源编号，r_s 表示资源开始时间，r_e 表示资源结束时间，$state$ 表示资源当前的是否占用，in 表示资源是否为新增资源，out 表示资源是否为退出资源；每个任务请求由七元组 T（id，s，d，a_s，a_c，a，r）构成，其中 id 表示任务对应的用户编号，s 表示任务开始时间，d 表示任务需要执行的时间片长，a_s 表示任务实际开始时间，a_c 表示任务实际完成时间，a 表示该任务调整的状态，r 表示任务所占用的资源。以图 1（c）有用户退出任务请求为例，为简化分析，设每个网格用户的并发任务数为 1，任务动态变化 action 包括 3 种动作，a_0：当前任务不变化，即 $s=a_s$；a_1：任务在当前资源中前移 5 秒，即 $s=a_s+5$；a_2：任务在当前资源中前移 10 秒，即 $s=a_s+10$。表 1 所示为各个用户请求资源以及所产生的动作事务集。其中 r_i = “ – ”（$i=1$，2，3）表示资源 r_i 闲置，有任务退出资源请求。

表 1　任务资源分配事务集

Table1　Datasets of resources allocation tasks

ID	r_1	r_2	r_3	action
1	–	u_2	u_3	a_0
2	u_1	u_2	–	a_1
3	u_2	u_1	u_3	a_0
4	u_1	–	u_2	a_1
5	–	–	u_2	a_2
6	u_1	u_3	u_2	a_0
7	u_1	–	u_2	a_1
8	u_2	–	–	a_2

根据改进的 *FP-Growth* 分类算法对表 1 中的事务进行分类挖掘[9]。设最小支持度计数 $min\_sup=3$。算法首先产生候选 1 项集 C_1 以及频繁 1 项集 L_1，通过频繁项集中项之间连接运算 $L_1 \rhd\lhd L_1$ 生成下一个候选集 C_2，直到候选项集为空为止。产生的各个频繁项集分别如下：L_1：{u_1:5，u_2:8，u_3:3，a_0:3，a_1:3}，L_2：{$<u_1,u_2>$:5，$<u_1,a_1>$:3，$<u_2,u_3>$:3，$<u_2,a_0>$:3，$<u_2,a_1>$:3，$<u_3,a_0>$:3}，L_3：{$<u_1,u_2,a_1>$:3，$<u_2,u_3,a_0>$:3}。

最后得到频繁模式为 {$<u_1,u_2,a_1>$，$<u_2,u_3,a_0>$}，虽然项集 {u_2，a_2} 的支持度小于 2，考虑到在频繁模式中没有出现 a_2 动作，因此将其加入到频繁模式集中，即最终的频繁模式为 {$<u_1,u_2,a_1>$，$<u_2,u_3,a_0>$，$<u_2,a_2>$}。设最小置信度 $min\_conf=60\%$，得出的分类规则有：$u_1 \wedge u_2 => a_1$，$u_1 \wedge a_1 => u_2$，$a_1 \wedge u_2 => u_1$，$u_1 => u_2 \wedge a_1$，$a_1 => u_1 \wedge u_2$；$u_2 \wedge u_3 => a_0$，$u_2 \wedge a_0 => u_3$，$a_0 \wedge u_3 => u_2$，$u_3 => u_2 \wedge a_0$，$a_0 => u_2 \wedge u_3$；$u_2 => a_2$，$a_2 => u_2$。由于主要考虑任务请求的动作，因此重点关注 $u_1 \wedge u_2 => a_1$，$u_2 \wedge u_3 => a_0$ 以及 $u_2 => a_2$ 三个分类规则。

从分类挖掘得出的规则可知，当请求队列中只有 u_1 和 u_2 两个用户时，执行 a_1 动作，任务的实际开始时间 a_s 向前移 5 秒；当请求队列中只有 u_2 和 u_3 两个用户时，执行 a_0 动作，任务的实际开始时间 a_s 不变；当请求队列中只有 u_2 一个用户时，执行 a_2 动作，任务的实际开始时间 a_s 向前移 10 秒。

图 1 中其它几种资源分配情况的分类规则产生的过程与上述方法类似，只需针对不同的问题对任务调整动作进一步进行细化和扩展即可。

2.3　动态分配算法

设集群中有 m 个节点，每个节点包括有同构的 c 个 CPU 资源，则当前集群中的资源总数 $r_{sum}=m*c$，标识为 r_1，…，r_{m*c}，资源集 R 由 r_{sum} 个资源 r（id，r_s，r_e，$state$，in，out）构成；初始状态包括有 n 个用户，标识为 u_1，…，u_n，用户数 n 随着用户的加入或退出动态变化，每个用户有 1 个任务请求。分类挖掘模块所生成的分类规则集 P 由 t 个分类规则构成，分别为 p_1，…，p_t。任务动态调整动作集 A 由 z 个动作 a_0，…，a_z 构成，分别表示前移、后移、上移、下移、交换等动作。

资源管理服务器中的基于分类挖掘的资源动态分配策略由 *DRA* 算法实现，具体算法描述如下：

RA 算法 – 静态资源分配算法

输入：R，T，$a[z]$；

int $start=s$；//获取任务的开始时间；

while（r_i. state = True）{ //直到资源闲置止；

start + + ; //如果资源被占用，则开始时间加 1;

}

for ($i = 1; i < = n; i + +$) { //修改每个任务的属性;

$t_i.a_s = start$;

$t_i.a_c = start + d$;

$t_i.r = i$; // 当前任务分配至编号为 i 的资源;

$t_i.a = a[0]$; //不发生动态变化;

}

输出：任务的 a_s，a_c，a，r 等信息。

DRA 算法 – 动态资源分配算法

输入：$a[z]$，P，R，T，r_{count}，u_{count}; //T 由 id，s，d，a_s，a_c，a 构成，r_{count} 表示当前资源数量，u_{count} 表示当前用户数量;

Initialize $a[z]$，$r._{in}$ = True，$r._{out}$ = True;

when a new resource r enter in {//当有新资源加入到资源池，修改资源的相关属性; r_{count} + +;

$r[r_{count}]._{in} = False$;

$r[r_{count}]._{state} = True$;

$r[r_{count}]._{out} = False$;

}

if ($n = m * c$) and ($r_{count} = m * c$) and ($u_{count} = n$) //没有资源、用户的动态变化，调用静态分配算法;

call RA algorithm;

else {

for each p_i in P { //每个分类挖掘规则

get a_x from p_i//获取 p_i 规则中的动作;

if $x < > 0$ //有资源动作调整;

for each user in p_i { //分类规则 p_i 每个用户;

modify $t.a_s$; //根据调整动作修改任务的实际开始时间;

$t.a_c = t.a_s + t.d$; //修改任务的实际完成时间;

$t.a = x$; //标识任务的动作;

$r._{state}$ = False; //资源已被占用;

}

else //无动作发生时，调用静态资源分配算法;

call *RA* algorithm;

}

}

输出：资源 R 的 *state*，*in*，*out* 等信息，任务 T 的 a_s，a_c，a，r 等信息。

当资源和用户请求没有动态变化时，调用 RA 算法；否则，调用 DRA 算法。DRA 算法根据分类挖掘产生的分类规则，提取分类规则中的资源动作以及相关用户，然后根据资源动作类型对相应用户的任务信息进行调整，从而实现资源的动态分配。

3 仿真结果及分析

3.1 实验过程

实验过程中，通过数据表记录每个任务的历史动作信息，通过 FP-Growth 分类挖掘算法实施分类挖掘，形成分类规则，在此基础上，利用 GridSim 模拟环境对 DRA 算法的进行仿真实验，得出资源动态分配结果以及相应的资源利用率等数据输出。在 GridSim 仿真实验中，创建 CPU 资源、集群、用户和任务对 DRA 算法进行仿真，主要参数设置如下：网格中集群数为 3，分别为 Cluster1，Cluster2，Cluster3，3 个集群的初始节点数 m 分别为 3，4，5，每个节点的处理器数 $c = 4$，因此资源总数 r_{sum} 分别为 12，16，20。其中每个节点的 CPU 类型相同；初始用户数 n 分别为 12，16，20，每个用户的请求任务数为 1。

采用 DRA 算法对集群中的 CPU 资源进行动态资源分配。针对 CPU 资源变化从 – 5 ~ 5，用户请求数不变；用户请求变化从 – 5 ~ 5，资源数不变的情况，得到资源动态分配后利用率如图 3 所示。

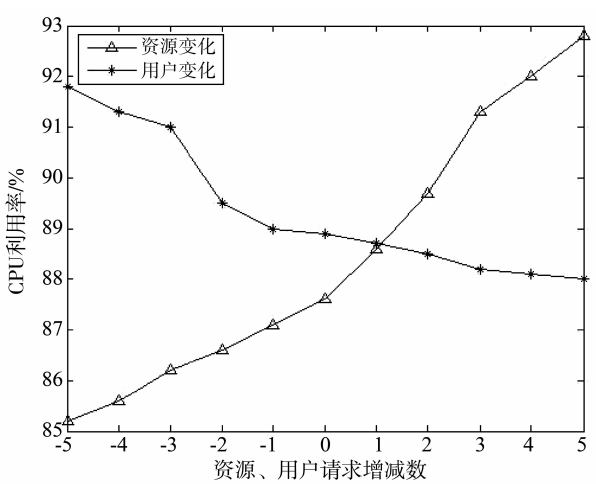

图 3 资源、用户请求增减对资源利用率影响

Fig. 3 Effect of resources utilization with the number of resources and users' change

在同样的仿真实验环境中，基于各个任务的历史动作信息，采用 FP-Growth 分类挖掘算法以及

DRA 算法对集群环境中的 CPU 资源进行动态资源分配，对比其它动态资源分配算法（FLexCo、SQ）在 CPU 资源利用率上的关系，进行 50 次实验，得到 CPU 资源利用率的对比关系如图 4 所示。

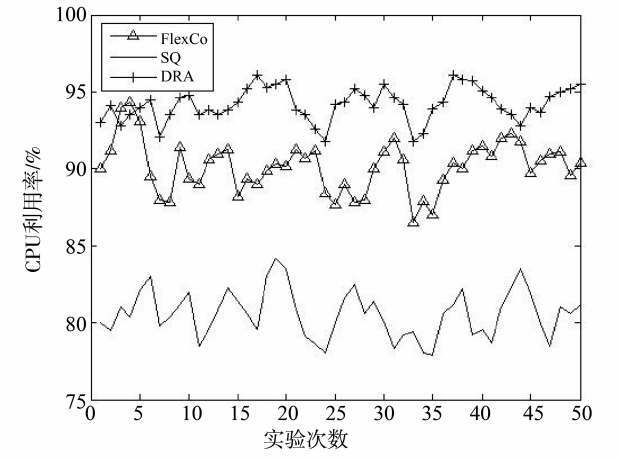

图 4　三种算法的资源利用率对比

Fig. 4　Contrast of CPU resource utilization between three kinds of algorithms

3.2　实验分析

从图 3 的结果可知，资源和用户请求的变化均会对资源利用率造成影响。其中，节点资源利用率 =（任务占用该节点时间片/节点时间片）* 100%。随着资源数量的减少，资源的利用率会降低，相反，当资源池中的资源增加时，资源的利用率会逐渐提高；相应的，随着用户请求数的减少，资源的利用率会提高，而当用户请求数增加时，资源利用率会逐渐降低，根据节点资源利用率的定义可知资源和用户的变化是相反的。当资源数减少或用户请求数增加时，资源的利用率并不会急速下降。

图 4 对比了 DRA 算法和 FLexCo、SQ 在资源动态分配时的效率。通过多次实验，DRA 算法绝大部分实验的 CPU 资源利用率优于其他两种算法，其中有几次实验的 CPU 利用率要低于 FlexCo 算法，但高于 SQ 算法；FLexCo 效率其次，SQ 算法效率最差。

4　结　语

文中对介绍了资源管理以及资源动态分配，提出了资源分配中所产生的几个问题，为了提高网格资源的利用率，合理地对集群环境中的资源进行分配非常重要。针对单个集群环境中的资源、用户任务请求以及分配过程中的动态变化，对资源动态变化的历史信息进行分类挖掘，形成相应的分类规则，从而指导资源的动态分配和再分配。实验证明，DRA 算法能有效提高资源的利用率。

在算法的实现过程中，可以在任务信息 T 中加入优先级、经济利益以及社会关系等属性[10]，从而使算法的实现与真实环境更接近。另外，针对资源动态分配的动作类别，根据不同情况，可以将类别进行细化，从而进一步提高资源的利用率。

参考文献：

[1]　王观玉. 网格计算中任务调度算法的研究和改进[J]. 计算机工程与科学, 2011, 33(10):186 - 187.

[2]　郑志蕴, 赵甜, 张勇涛. 网格环境下改进 PSO 算法的资源分配研究[J]. 计算机工程, 2011, 37(1):178 - 180.

[3]　严大鹏, 杜学东. 网格资源分配算法的研究[J]. 计算机工程与应用, 2008,44(29):135 - 137.

[4]　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 - 107.

[5]　NETTO M A S, VECCHIOLA C, KIRLEY M, et al. Use of run time predictions for automatic co-allocation of multi-cluster resources for iterative parallel applications [J]. Journal of Parallel and Distributed Computing, 2011,71(5):1388 - 1399.

[6]　NETTO M A S, BUYYA B. Offer-based scheduling of deadline-constrained bag-of-tasks applications for utility computing systems [C]//Parallel & Distributed Processing, 2009:1 - 11.

[7]　NETTO M A S, BUYYA B. Rescheduling co-allocation requests based on flexible advance reservations and processor remapping [C]//9th Grid Computing Conference, 2008:144 - 150.

[8]　LI J Y, QIU M K, MING Z, et al. Online optimization for scheduling preemptable tasks on IaaS cloud systems [J]. Journal of Parallel and Distributed Computing, 2012, 72(2): 666 - 677.

[9]　YE Y, CHIANG C C. A parallel apriori algorithm for frequent itemset mining [C]//Fourth International Conference on Software Engineering Research, Management and Applications,2006:87 - 94.

[10]　LEE Y C, WANG C, ZOMAYA A Y, et al. Profit-driven scheduling for cloud services with data access awareness[J]. Journal of Parallel and Distributed Computing, 2011,71(12):591 - 602.