基于分类挖掘的网格资源分配研究

刘林东[1,2]

(1. 广东第二师范学院 计算机科学系, 广州 510303; 2. 华南理工大学 计算机科学与工程学院, 广州 510006)

摘　要: 根据用户访问网格资源的历史信息,采用分类算法对此信息进行挖掘,得出用户使用集群资源的访问规则和模式,在此基础上构造一种基于分类挖掘的资源调度模型、用户调度 UA 算法以及资源调度 CDMRA 算法,分别将用户请求调度到各个集群中闲置的 CPU 资源。实验证明,采用基于分类挖掘的资源分配策略相比其他算法可以减少资源分配过程中对资源的重新分配次数,可以提高网格资源的利用率。

关键词: 分类挖掘; 网格计算; 资源; 资源分配; 集群

中图分类号: TP391.9　　**文献标志码**: A　　**文章编号**: 1001-3695(2013)02-0371-03

doi:10.3969/j.issn.1001-3695.2013.02.013

Research of resource allocation algorithm based on classification data mining

LIU Lin-dong[1,2]

(1. *Dept. of Computer Science, Guangdong University of Education, Guangzhou* 510303, *China*; 2. *School of Computer Science & Engineering, South China University of Technology, Guangzhou* 510006, *China*)

Abstract: Making use of classification data mining algorithm to analyze user's historical access information, this paper got user's classification access rules and patterns in cluster renvironment. Firstly, it constructed a classification mining-based resource scheduling model. Secondly, it designed a UA algorithm to allocate all of the users' tasks in each cluster. Finally, it applied a new resource algorithm CDMRA to assign users' tasks to idle CPU resources in every node. Experiments show that CDMAR algorithm can reduce the resource reallocation times and improve the efficiency and accuracy of resource allocation compared to other algorithms. It can increase the utilization of grid resources.

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

网格计算[1]是利用网络中一些闲置的计算能力来解决复杂问题的计算模式,适用于大型科学计算和应用研究。在网格计算环境中,计算任务的调度是一项非常复杂的工作,任务调度算法的策略直接影响网格计算的性能以及资源的利用率。资源调度是将用户提交的作业任务合理地分配到网格环境中相应的资源上进行计算。目前常用的资源分配策略主要包括如何对资源进行有效分配、资源分配的 QoS、资源分配的容错性以及资源分配的平台等方面的研究。资源分配策略有静态资源分配的 HEFT[2]、FCFS、EBF、FPFS 等算法;资源协同分配主要针对多集群分布式环境下,多用户多资源并发访问模式,协同分配[3]策略广泛应用于 GUR、HARC、KOALA、OAR 调度模型中,有 SCOAL 算法以及 SQ 模式等;资源再分配策略[4]主要是为了解决资源静态分配过程中的不合理现象,需要对资源分配进行重新调整,相关的研究较少,有弹性资源分配的 Flex-Co、Processor Remapping 模型等。

1　资源分配

在网格计算环境下,为了使集群服务器获取更好的计算性价比,需要对集群服务器中的各种资源进行合理的分配和管理。网格资源是由地理上分布、隶属不同机构、异构的各类计算资源、存储资源、网络带宽资源等组成,一般包括高性能计算机、工作站、数据库、存储器、CPU、网络带宽等资源。

在网格计算网络中,资源分配[5]的一般过程为:a)网格用户将各种计算任务的请求提交给元调度器,由元调度器对所有任务进行合并、分解、优先级设定等;b)元调度器将相应的子任务分配给网格环境中各本地调度器;c)本地调度器根据相应的调度算法对各个集群中的可用资源进行调度,即将请求任务与可用资源进行映射,每一个集群资源负责相应子任务的计算;d)计算任务完成后,将计算结果返回给元调度器,最后反馈给网格用户。

网格环境[6]中的资源分配策略一般包括静态资源分配、弹性资源分配、资源协同分配和资源再分配。各种分配策略主要从响应时间、资源利用率以及吞吐量三个指标来衡量。

静态资源分配也称为预先资源分配,根据用户的需求预先对任务进行资源分配。其优点是算法简单,缺点是容易产生碎片,资源利用率较低,主要用于 LSF、PBS Pro、Maui、Catalina 和 ANL/IBM SP 调度系统[7]中。弹性资源分配可以根据时间片大小、任务数量进行灵活分配,也可以通过调整任务的开始执行时间以及处理器重新映射来解决。目前各种网格平台均采用资源协同分配策略,可以最大程度地提高资源的利用率和减少碎片。资源再分配[8]一般是在弹性资源分配和资源协同分配的基础上实现的,可以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。

2　基于分类挖掘的资源分配

2.1　基于分类挖掘的资源分配模型

基于分类挖掘[9]的资源分配模型总体上分为用户层、任

收稿日期: 2012-06-24; **修回日期**: 2012-08-06

作者简介:刘林东(1978-),男,湖南邵阳人,讲师,博士研究生,主要研究方向为并行计算、分布式计算(hongox@163.com).

务调度层和资源层三层体系,如图 1 所示。其中用户层中的网格用户向任务调度层中的调度器发送资源请求,计算完成后接收计算调度器返回的计算结果;任务调度层包括元调度器(metascheduler)和本地调度器(local scheduler)。元调度器负责将用户请求的任务采用相应的策略分配到本地调度器中,再由本地调度器将任务分配给集群中的相应节点。元调度器包括分类数据挖掘(CDM)模块、用户管理(UM)模块以及资源管理(RM)模块。CDM 模块负责对整个网格进行分类挖掘;UM 模块负责管理用户以及根据 CDM 模块的结果将用户的请求分发到各个集群中去;RM 模块负责管理资源层的所有资源。本地调度器包括有资源调度(RA)模块,通过元调度器中的 CDM 模块分类挖掘所产生的分类挖掘模式,指导 RA 模块对当前集群中的闲置资源[10]进行合理的调度。

2.2 分类挖掘

网格用户向元调度器发送任务请求,元调度器中的 PM 模块将用户请求分配给相应的本地调度器。本地调度器中的 CDM 模块采取分类挖掘策略对用户请求的历史任务信息进行分析,从中找出对本地调度器调度任务有帮助的决策模式。

由于用户一般已对网格环境中的资源进行了访问,在各个集群的本地调度器中记录相应的历史任务信息。资源访问的任务信息主要包括用户请求任务的用户编号 u_i、时间片长度 d_{ij}、请求资源类型 r_i、集群编号 c_i 等。图 2 表示用户访问同一个节点的 CPU 资源时间片情况,其中 d_{ij} 表示第 i 个用户的第 j 个任务所占时间片长度。

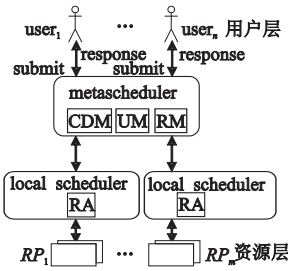

图1 基于分类挖掘的资源分配模型

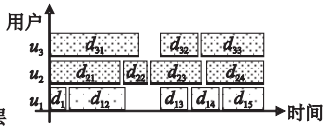

图2 用户资源时间片信息

为了方便分类挖掘,将用户访问资源的时间片划分为四个区间(单位:min),分别为 t_1:[0,5),t_2:[5,10),t_3:[10,15),t_4:[10,+∞)。在本地调度器中的守护程序记录用户每次访问处理器资源的时间片信息。表 1 中表示用户使用节点资源(r_1,r_2,r_3)的时间片信息,t_{ij} 表示请求资源 i 的时间片区间为 j,没有计算任务的时间片标记为 t_{i1},其中 i 为资源编号。

表 1 资源事务集 D

ID	r_1	r_2	r_3	userID	ID	r_1	r_2	r_3	userID
1	t_{11}	t_{22}	t_{31}	u_1	5	t_{12}	t_{22}	t_{33}	u_2
2	t_{12}	t_{21}	t_{32}	u_2	6	t_{13}	t_{22}	t_{34}	u_3
3	t_{14}	t_{21}	t_{31}	u_1	7	t_{12}	t_{22}	t_{32}	u_2
4	t_{11}	t_{21}	t_{31}	u_1					

根据 Apriori、改进的 FP-Growth 分类算法[11]对表 1 中的事务进行挖掘,设最小支持度计数 min_sup = 2,得出频繁模式为 $\{\langle t_{11},t_{31},u_1\rangle,\langle t_{12},t_{22},u_2\rangle,\langle t_{12},t_{32},u_2\rangle,\langle t_{21},t_{31},u_1\rangle\}$;设置最小置信度 min_conf = 70%,产生分类规则:$t_{11}\wedge t_{31}\Rightarrow u_1$,$t_{11}\wedge u_1\Rightarrow t_{31}$,$t_{11}\Rightarrow t_{31}\wedge u_1$,$t_{21}\wedge t_{31}\Rightarrow u_1$,$t_{21}\wedge u_1\Rightarrow t_{31}$ 和 $t_{12}\wedge t_{22}\Rightarrow u_2$,$t_{22}\wedge u_2\Rightarrow t_{12}$,$t_{12}\wedge t_{32}\Rightarrow u_2$,$t_{32}\wedge u_2\Rightarrow t_{12}$,$t_{32}\Rightarrow u_2\wedge t_{12}$。根据 t_{ij} 与 r_i 之间的对应关系,得出 u_1 使用 r_1 和 r_2 的时间片为 t_1 或使用 r_2 和 r_3 的时间片为 t_1;u_2 使用 r_1 和 r_3 的时间片为 t_2 或使用 r_1 和 r_2 的时间片为 t_2。

同时可以针对访问周期内用户对资源访问的情况进行分类挖掘,产生基于用户的分类规则,得出分类规则:$t_2\wedge u_1\Rightarrow t_1$,$t_1\wedge u_2\Rightarrow t_2$,$t_2\wedge u_2\Rightarrow t_1$,$u_2\Rightarrow t_1\wedge t_2$。即如果 u_1 使用 t_2 任务请求的话,u_1 也会使用 t_1 类任务;u_2 在请求资源时,会同时使用 t_1 和 t_2 两种类型的时间片。

2.3 资源分配算法

设网格环境中有 n 个用户,分别为 $u_1,\cdots,u_n$,每个用户有 w_i 个任务请求,任务请求的时间长度表示为 T_{ij}(i 为用户编号,j 为任务编号,其中 $1\leqslant i\leqslant n$,$1\leqslant j\leqslant w_i$)。由 n、w_i 以及 T_{ij} 可以得出所有用户所有任务请求的时间片总长 T_{sum} 为

$$T_{\text{sum}}=T_{11}+T_{12}+\cdots+T_{1w_1}+\cdots+T_{n1}+T_{n2}+\cdots+T_{nw_n}\quad(1)$$

网格中有 m 个集群,负责提供相应的处理器资源,分别为 $C_1,\cdots,C_m$,每个集群包含 N_i 个节点,每个节点中均包括有 c 个同构的 CPU 资源,资源的编号为 $r_1,r_2,\cdots,r_s$,整个网格环境上的处理器资源总数 R_{sum} 如式(2)。其中 $1\leqslant i\leqslant m$,$1\leqslant s\leqslant R_{\text{sum}}$。

$$R_{\text{sum}}=c\times(N_1+N_2+\cdots+N_m)\quad(2)$$

通过分类挖掘,可以将访问同一集群的用户分类在一起,产生形如 $u_i\wedge\cdots\wedge u_j\Rightarrow C_k$ 的规则,其中 $1\leqslant i\leqslant n$,$1\leqslant j\leqslant n$,$1\leqslant k\leqslant m$。

在分类挖掘算法得出的分类规则及挖掘模式的基础上,先通过元调度器中的 UM 模块利用 UA 算法将用户请求分发至各个集群中,再由本地调度器中的 RA 模块利用 CDMRA 算法对用户的新任务请求进行调度。UA 和 CDMRA 算法分别如下:

UA 算法——用户分配算法

输入:Classification_rules_set,$u_1,\cdots,u_n$,m,$C_1,\cdots,C_m$,T_{sum},R_{sum}。

```
initialize c[m][n] = 0, u_count = 0;
if count_rule = m{ //规则数即为集群数
for (i = 1; i < = count_rule; i + + ){
  C_k ∈ P_i; //获取当前规则的集群编号
  for every u_i ∈ P_i{//对集中的每一个规则 P_i
    c[k,j] = 1; u_count + + ;
    assign u_j's tasks to C_k;//将 u_j 的任务分配给集群 k;
  }
}
if u_count < n{
  T_additional = T_new/m; /* T_additional表示每个集群额外增加的计算任务,T_new表示新用户请求的任务时长 */
  update c[m][n];
}
else{
  T_random/(m-count_rule);
  //T_random表示随机从集群中选择用户的任务时长
  update c[m][n];
}
```

输出:$c[m][n]$。

UA 算法将分类挖掘产生的分类模式、用户信息、集群信息等作为输入,从中找出哪些用户属于同一集群,将用户任务分发给相应的集群。如果某些用户不在分类规则中,说明这些用户之前没有访问网格资源,属于新用户,UA 算法将这些用户的任务平均分配到各个集群中;如果分类规则中的集群数少于网格中集群数,说明部分集群在之前没有被这些用户访问,根据闲置集群的数量,从每个集群中随机选择一个用户,并将选出的用户任务平分到各个闲置集群中。

CDMRA——基于分类挖掘的资源分配算法

输入:Classification_rules_set,$c[m][n]$,C_k,$u_1,\cdots,u_j$,$r_1,r_2,\cdots,r_i$。

```
initialize r[j][i] = 0, s[j] = 0;
for (q = 1; q < = count_rule; q + + ){
  u_l ∈ P_q; //获取当前规则的用户编号
  for every r_s ∈ P_q{
    s[l] + + ;
    r[l,s] = 1 or 2 or 3 or 4;
    //根据分类规则中四种时间类型赋予不同的值
```

```
    }
  }
  for (q = 1; q <= j; q++) {
    T_task = T_q / s[q];
    //将 u_q 的请求任务平均分配到相应的资源中;
    for (u = 1; u <= s[q]; u++)
      assign T_task to r_{d_u}; //d_u 为 r[q,s] 中值为非 0 的下标;
  }
  输出:r[j][i]。
```

CDMRA 算法在 UA 算法的基础上,在每个集群内部对当前集群中的资源进行调度。根据所产生的分类规则,判断每个用户的资源访问主要集中于哪几个资源以及资源访问的时间特征,由此对请求任务进行直接调度。

3　实验结果及分析

3.1　实验过程

在 UA 和 CDMRA 算法的实验过程中,通过 GridSim 模拟环境创建资源、集群、用户和任务对算法进行仿真。主要参数设置如下:网格中集群数 $m=3$,每个集群中的节点数 $N_i=5$,每个节点的处理器数 $c=8$,因此资源总数 $R_{sum}=120$;每个集群中的资源数为 40 个,其中每个节点的 CPU 类型和数量均相同;用户数 $n=10$,每个用户的请求任务数 $w_i=3$,每个任务请求的时间片在 0 ~ 30 min 内。

先采用 UA 算法将用户分配给各个集群,算法执行的结果是 $C_1:\{u_1,u_2,u_3,u_9\}$,$C_2:\{u_4,u_6,u_{10}\}$,$C_3:\{u_5,u_7,u_8\}$。在 UA 算法的基础上,最后采用 CDMRA 算法分别在各个集群中实现资源调度。在实验过程中,对该算法共执行 50 次,CDMRA 算法的执行次数与集群中任务完成时间关系如图 3 所示。对比其他资源分配算法(FCFS、EBF),得到在处理器资源利用率上的对比关系,如图 4 所示。

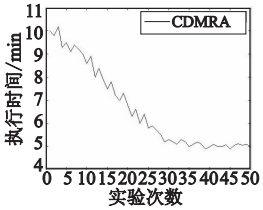

图3　任务完成时间与算法次数关系

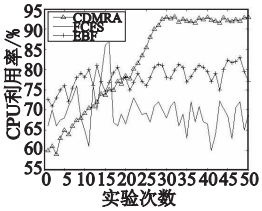

图4　几种算法的资源利用率对比

3.2　实验分析

从图 3 的实验结果可知,在采用 CDMRA 算法进行资源调度时,由于需要基于用户访问资源的历史信息,在算法执行的前期,因没有足够的信息用以支撑分类挖掘,所以得不到有效的分类规则,用户任务在执行过程中所花费的时间较长。随着执行次数的增加,分类挖掘模块有足够的历史信息进行分析,从而可以有效地指导后续的任务,使资源在分配的过程中更有效,在执行到 30 次以后,任务的执行时间围绕在 5 min 上下波动,并最终趋于稳定。

图 4 的实验结果是分析了文中提出的 CDMRA 算法与其他两种算法在资源利用率上的对比。容易得知,在算法执行的早期,由于分类挖掘对历史信息的敏感,所以造成资源利用率较低,但随着执行次数的增加,处理器资源利用率最终趋向于稳定,处于 90% 较高的水平;而 FCFS 算法由于只是采用了简单的先来先分配的策略,其算法效率处于 70% 的低位水平,该算法不受执行次数的影响,在执行过程中,由于受执行任务的影响,实验结果会出现某些异样;EBF 算法采用了回滚策略,其算法效率介于 FCFS 和 CDMRA 之间。

4　结束语

本文提出了 UA 和 CDMRA 算法。在分类挖掘的资源分配模型中,分类挖掘模块主要存在于元调度器中,可以视集群的规模情况,在资源分配模型中采取多层架构,调度器间可实现信息共享和相互调度,可以在本地调度器中增加分类挖掘守护程序以提高分类挖掘的效率。

文中只针对处理器资源的分配策略和算法进行了研究,当网格网络中存在处理器资源、存储资源以及带宽资源多类资源[12]时,需要对多种资源进行协调处理[13],使调度算法可以对各种资源的利用率达到整体优化。在 CDMRA 算法的设计中,由于分类规则中记录了用户访问资源的时间特性,因此可以将 t_1、t_2 类任务在适当的时候合并为 t_3、t_4 类任务,减少任务分配过程中所造成的碎片问题。

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] DECKER J, SCHNIDER J. Heuristic scheduling of grid workflows supporting co-allocation and advance reservation[C]//Proc of the 7th CCGrid. Washington DC: IEEE Computer Society,2007:335-342.

[3] BUCUR A I D, EPEMA D H J. Scheduling policies for processor co-allocation in multicluster system[J]. IEEE Trans on Parallel and Distributed Systems,2007,18(7):958-972.

[4] NETTO M A S, BUYYA R. Rescheduling co-allocation requests based on flexible advance reservations and processor remapping[C]//Proc of the 9th Grid Computing Conference. [S. l.]: IEEE Press, 2008:144-150.

[5] NETTO M A S, De ROSE C A F. CRONO: a configurable and easy to maintain resource manager optimized for small and mid-size GNU/Linux cluster[C]//Proc of International Conference on Parallel Processing. Washington DC: IEEE Computer Society,2003:555-562.

[6] 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.

[7] KAUSHIK N R, FIGUEIRA S M, CHIAPPARI S A. Flexible time-windows for advance reservation scheduling[C]//Proc of the 14th MASCOTS. Washington DC: IEEE Computer Society,2006:218-225.

[8] NETTO M A S, BUYYA R. Offer-based scheduling of deadline-constrained bag-of-tasks applications for utility computing systems[C]//Proc of IEEE International Symposium on Parallel & Distributed Processing. Washington DC: IEEE Computer Society,2009:1-11.

[9] 方金城. 分类挖掘算法综述[J]. 沈阳工程学院学报:自然科学版,2006,2(1):73-76.

[10] NETTO M A S, CALHEIROS R N, SILVA R K S, *et al*. Transparent resource allocation to exploit idle cluster nodes in computational grids[C]//Proc of the 1st International Conference on e-Science and Grid Computing. Washington DC: IEEE Computer Society,2005:238-245.

[11] YE Yan-bin, CHIANG C C. A parallel Apriori algorithm for frequent itemset mining[C]//Proc of the 4th International Conference on Software Engineering Research, Management and Applications. Washington DC: IEEE Computer Society,2006: 87-94.

[12] FERRETO T C, NETTO M A S, CALHEIROS R N, *et al*. Server consolidation with migration control for virtualized data centers[J]. Future Generation Computer Systems,2011,27(4):1027-1034.

[13] 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 Distrib uted Computing,2011,71(5):1388-1399.