文章编号:1000-5641(2009)01-0111-06

一种多 QoS 约束的网格资源调度算法

陈玉兰, 郑 骏, 胡文心
(华东师范大学 计算中心,上海 200062)

摘要: 针对网格计算中资源调度问题,提出一种多 QoS(Quality of Service)约束的贪婪推广算法. 调度时以调度驱动函数为贪婪准则. 调度驱动函数与网格用户提供的"deadline","budget","time weight"和"cost weight"等参数有关. 每次根据调度驱动函数选择最经济的资源进行调度. 通过在 Gridsim 环境下的分析与比较,该算法能够在满足用户 QoS 的同时,能够以最小的调度驱动函数值完成任务,最大化地实现用户的满意度.

关键词: 经济模型; 贪婪算法; Gridsim; 调度; 网格计算
中图分类号: TP399 **文献标识码**: A

Algorithm for resource scheduling based on multi-QoS constraints

CHEN Yu-lan, ZHENG Jun, HU Wen-xin
(*Computer Center Institute, East China Normal University, Shanghai* 200062, *China*)

Abstract: A new algorithm for resource scheduling in a grid environment based on multi-QoS constraints and greedy method was proposed. It takes a driver function which is determined by parameters"deadline", "budget", "time weight" and "cost weight" which the user provided as the greedy criterion. The performance of this new scheduling algorithm, choosing a most economic resource for scheduling each time and achieving the maximum of customer's satisfaction, is demonstrated by its implementation on Gridsim.

Key words: economic model; greedy algorithm; Gridsim; schedule; computational grid

0 引 言

网格计算[1,2]伴随着互联网技术而迅速发展起来,是专门针对复杂科学计算的新型计算模式:把整个网络整合成一台巨大的超级计算机,实现计算资源、存储资源、数据资源、信息资源、知识资源和专家资源的全面共享. 网格计算资源在现代社会已成为解决大型工业科学计算的趋势. 然而网格计算资源的分散性,决定基于跨节点通信的分布计算的代价远远高于局域网中的分布式计算,频繁的通信造成相当大的延时开销[3]. 提高并行性是减小通信开销的重要手段. 由于地理位置上分散的网络资源的执行速度和开销不同,调度时还要考虑用

收稿日期:2007-12
第一作者:陈玉兰,女,硕士,主要研究方向为 web 应用. E-mail:51071211003@ecnu. cn.
通讯作者:郑骏,男,高级工程师,主要研究方向为 web 应用. E-mail:jzheng@cc. ecnu. edu. cn.

户对调度任务的时间和开销等方面的需求.

1 用户 QoS 调度问题

由于网格资源的分散性,资源种类的差异,资源被不同个人和组织所拥有,资源的运行开销的差异,任务性质的差异等因素,资源调度是很复杂的一个过程[4].大多数现有的网格资源管理和调度系统采取的调度策略仅以提高系统的吞吐量和利用率以及在最早时间内完成任务为目标,并未将资源访问成本和用户对时间和开销的要求考虑在内,没有很好地考虑用户的需求.不同应用背景下的用户需求相差甚远.由于任务的轻重缓急不同,有些用户要求在费用不超过 budget 的前提下,任务越早完成越好.有些要求运行时间不超过 deadline 前提下,花费越少越好.还有的用户要求两者兼顾.

本文对贪婪算法加以推广,以调度驱动函数为贪婪准则,实现用户对该任务需求下的最经济的调度策略进行调度.

2 贪婪算法

贪婪算法[5,6](greedy algorithm)采用逐步构造最优解的方法.在每个阶段,都做出一个看上去最优的决策(在一定的标准下).决策一旦做出,就不可再更改.做出贪婪决策的依据称为贪婪准则(greedy criterion).

本文将贪婪算法加以推广,考虑用户的需求,做出网格资源调度决策.基于用户提交参数的成本函数为该算法的贪婪准则.

3 基于用户 QoS 的网格资源调度算法

3.1 调度成本函数

用户提交任务时设置完成该任务所能接受的时间限制和开销限制,分别用参数 deadline 和 budget 表示.系统资源 R_i 每一单位时间运行的行数(以 MI 为单位)和每秒执行代码的开销分别用参数 V_i 和 C_i 表示.假设任务 A_j 的行数为 L_j,把任务 A_j 调度到资源 R_i 执行,则时间开销 $=L_j/V_i$,执行代码花费 $=(L_j/V_i)\times C_i$.对不同的应用程序,用户对时间和开销的要求是不一样的:

(1) 有些用户要求在费用不超过 budget 的前提下,任务越早完成越好;

(2) 有些要求运行时间不超过 deadline 前提下,花费越少越好;

(3) 还有的用户要求两者兼顾.

所以为了更好地满足用户的需求,还须设两个参数 timeWeight,costWeight 作为用户赋给完成任务所需时间和花费的权重,为了方便推导,简单设为 α,β,且满足 $\alpha+\beta=1$.例如,对上述(1):$\alpha=1,\beta=0$;对上述(2):$\alpha=0,\beta=1$;对上述(3):$0<\alpha,\beta<1$ 且满足 $\alpha+\beta=1$.

上述带权重的总开销 $=(L_j/V_i)\times(\alpha+\beta\times C_i)$.任务 A_j 调度到资源 R_i 执行的总开销 $F(V_i,C_i,L_j)=(L_j/V_i)\times(\alpha+\beta\times C_i)$.称 $F(V_i,C_i,L_j)$ 为任务 A_j 调度到资源 R_i 上执行的成本函数.

3.2 调度驱动函数

假设某一时刻网格用户向系统提交了 n 个任务,现将任务按照长度递减排序得到任务列表 A,如图 1 所示.

$$A:A_1,A_2,\cdots,A_n$$
$$L:L_1,L_2,\cdots,L_n$$

图 1　任务列表

Fig. 1　List of tasks

系统中有 m 个可用资源列表，将资源按照资源执行速度递减排序得到系统资源列表 R，如图 2 所示. 资源速度与该资源上次执行任务速度有关，且充分考虑资源的繁忙程度.

$$R:R_1,R_2,\cdots,R_m$$
$$V:V_1,V_2,\cdots,V_m$$
$$C:C_1,C_2,\cdots,C_m$$

图 2　系统资源列表

Fig. 2　List of system resources

资源 R_i 执行任务的总时间用 T_i 表示，所有资源执行任务的总开销用 C_{total} 表示. 令 $G(V_i,C_i,T_i,L_j)=F(V_i,C_i,L_j)+\alpha\times T_i+\beta\times C_{\text{total}}$，其中 $1\leqslant i\leqslant m$，$1\leqslant j\leqslant n$. 则当任务 A_j 调度到资源上执行后，$G(V_i,C_i,T_i,L_j)=F(V_i,C_i,L_j)+\times\alpha\times T_i+\beta\times C_{\text{total}}$，其中 $F(V_i,C_i,L_j)$ 为第 j 次的调度开销，$\alpha\times T_i$ 为 $j-1$ 次调度后资源 i 的总运行时间的折合开销，则 $G(V_i,C_i,T_i,L_j)$ 为前 j 次的调度后资源 i 的总运行时间的折合开销.

3.3　网格资源调度算法

每次选择调度时，查找 $G(V_i,C_i,T_i,L_j)$ 值最小的资源进行调度，并修改相应资源的运行时间 T_i.

提交任务时：

(1) 初始时 $T_i=0$，其中 $1\leqslant i\leqslant m$，$G(V_i,C_i,T_i,L_1)=F(V_i,C_i,L_1)+\alpha\times T_i+\beta\times C_{\text{total}}=F(V_i,C_i,L_1)$，函数 $G(V_i,C_i,T_i,L_1)$ 的值最小，即 $F(V_i,C_i,L_1)$ 的值最小. 把 A_1 调度给 $F(V_i,C_i,L_1)$ 值最小的资源执行. 比较调度的时间和花费是否超出了用户可以承受的 deadline 和 budget. 如果不超出，则分配；否则调度失败. 若调度成功，假设该资源为 R_k，则本次调度结束后，$T_k=L_1/V_k$.

(2) 在进行第 j 次调度时，其中 $1<j<n$，$G(V_i,C_i,T_i,L_j)=F(V_i,C_i,L_j)+\alpha\times T_i+\beta\times C_{\text{total}}$，把 A_j 调度给 $G(V_i,C_i,T_i,L_j)$ 值最小的资源执行. 比较调度的时间和花费是否超出了用户可以承受的 deadline 和 budget. 如果不超出，则分配；否则调度失败. 若调度成功，假设该资源为 R_p，则本次调度结束后，$T_p=T_p+L_j/V_p$.

(3) 在进行第 n 次调度时，$G(V_i,C_i,T_i,L_n)=F(V_i,C_i,L_n)+\alpha\times T_i+\beta\times C_{\text{total}}$，把 A_n 调度给 $G(V_i,C_i,T_i,L_n)$ 值最小的资源执行. 比较调度的时间和花费是否超出了用户可以承受的 deadline 和 budget. 如果不超出，则分配；否则调度失败.

3.4　算法优越性证明

采用本算法进行调度比任何其他算法获得的调度驱动函数值都小，即用户的时间和开销折合值最小，能最好地满足用户对时间和开销的要求.

证明　假设采用本算法的调度为 D，采用其他算法的调度为 B，n 个任务按长度从大到小排列为：$L_1,L_2,\cdots,L_n$；m 个资源按运行速度排列为：$R_1,R_2,\cdots,R_m$. T^D 和 C^D_{total} 为调度 D 下的运行总时间和总开销，初始时为 0；T^B 和 C^B_{total} 为调度 B 下的运行总时间和总开销，初始时为 0.

现采用第二数学归纳法来证明解 D 是比 B 较优的一个解：

(1) 当进行第一次调度时，根据本调度算法，D 每次调度选取一个 G 函数值最小的资源，所以 $G(V_i^D,C_i^D,T_i^D,L_1)\leqslant G(V_j^B,\ C_j^BT_j^B,L_1)$，即 $\alpha\times T_i^D+\beta\times C_{\text{total}}^D\leqslant\alpha\times T_i^B+\beta\times C_{\text{total}}^B$，进行第一次调度后 D 是比 B 较优的一个解；

(2) 假设当进行完第 $k-1$ 次调度后，D 是比 B 较优的一个解，即满足 $\max(\alpha\times T_i^D+\beta\times C_{\text{total}}^D)\leqslant\max(\alpha\times T_j^B+\beta\times C_{\text{total}}^B)$，其中 $1\leqslant i\leqslant k-1,1\leqslant j\leqslant k-1$；

(3) 则进行第 k 次调度，若第 k 次调度完后 $\max(\alpha\times T_i^{D'}+\beta\times C_{\text{total}}^{D'})>\max(\alpha\times T_j^{B'}+\beta\times C_{\text{total}}^{B'})$，其中 $1\leqslant i\leqslant k,1\leqslant j\leqslant k$. 现用反正法证明不可能：若第 k 次调度后，D 把 A_k 调度给资源 R_p^D 执行，B 把 A_k 调度给资源 R_q^B 执行，由

$$\begin{cases}\max(\alpha\times T_i^D+\beta\times C_{\text{total}}^D)\leqslant\max(\alpha\times T_j^B+\beta\times C_{\text{total}}^B),\\ \text{其中 } 1\leqslant i\leqslant k-1,1\leqslant j\leqslant k-1;\end{cases}\tag{1}$$

$$\begin{cases}\max(\alpha\times T_i^{D'}+\beta\times C_{\text{total}}^{D'})>\max(\alpha\times T^{B'}{}_j+\beta\times C_{\text{total}}^{B'}),\\ \text{其中 } 1\leqslant i\leqslant k,1\leqslant j\leqslant k\end{cases}\tag{2}$$

可知：

$$\begin{aligned}\max(\alpha\times T_i^{D'}+\beta\times C_{\text{total}}^{D'})&=G(V_p^D,C_p^D,T_p^D,L_k)\\&>\max(\alpha\times T_j^{B'}+\beta\times C_{\text{total}}^{B'})\\&\geqslant G(V_q^B,C_q^B,T_q^B,L_k),\end{aligned}$$

即 $G(V_p^D,C_p^D,T_p^D,L_k)>G(V_q^B,C_q^B,T_q^B,L_k)$.

若 D 选择其它资源 R_i^D 进行调度，则 $G(V_i^D,C_i^D,T_i^D,L_k)\geqslant G(V_p^D,C_p^D,T_p^D,L_k)>G(V_q^B,C_q^B,T_q^B,L_1)$，因为 D 选择 R_p^D 调度是在 D 方案下的最优解，其他解都比它开销大，即 $G(V_q^B,C_q^B,T_q^B,L_k)<\min(G(V_i^D,C_i^D,T_i^D,L_k))$.

而又由 $\max(\alpha\times T_i^D+\beta\times C_{\text{total}}^D)\leqslant\max(\alpha\times T_i^B+\beta\times C_{\text{total}}^B)$，其中 $1\leqslant i\leqslant k-1,1\leqslant j\leqslant k-1$，可知在 D 的调度方案中不同资源的执行时间相对 B 比较平均，D 的并行性比较好. 且被调度的任务按照代码长度递减的顺序依次被调度，所以若存在 q 使得 $G(V_q^B,C_q^B,T_q^B,L_k)<\min(G(V_i^D,\ C_i^D,T_i^D,L_k))$，必定存在 j 使得 $\max(\alpha\times T_i^{B'}+\beta\times C_{\text{total}}^{B'})-G(V_j^D,C_j^D,T_j^D,L_k-1)\geqslant F(V_j,C_j,L_k)$ 成立. 把 A_k 调度给资源 R_j^D 执行，调度完成后有

$$\begin{aligned}\max(\alpha\times T_i^{D'}+\beta\times C_{\text{total}}^{D'})&\leqslant G(V_j^D,C_j^D,T_j^D,L_k-1)+F(V_j,C_j,L_k)\\&\leqslant\max(\alpha\times T_i^{B'}+\beta\times C_{\text{total}}^{B'}),\end{aligned}$$

这与(2)式矛盾.

综上所述，由数学归纳法得，利用推广的贪婪算法得到的调度方案 D 优于任意其它可行调度方案 B，实现了约定条件下的资源调度的最优化.

4 算法实现

为了验证算法的性能，采用 Gridsim[7,8] 仿真环境模拟调度过程.

假设一个网格用户有 8 个任务要执行，长度分别为 8000，7000，6000，5000，4000，3000，2000，1000，单位为 MI，用户设置的 deadline 和 budget 均为 1000，系统此时有 3 个可用资源 R_1,R_2,R_3，执行速度分别为 300 MIPS，500 MIPS，800 MIPS，开销分别为3.0，5.0，8.0(单位 cent/s). 在 Gridsim 下实现该算法。

当时间权重 timeWeight = 1(即 $\alpha = 1$)，开销权重 costWeight = 0(即 $\beta = 0$)时，在 Gridsim 中调度过程如表 1 所示.

表 1　调度过程

Tab. 1　Schedule process

任务 ID	任务长度/MI	开始执行时刻/s	执行完时刻/s	开销/cent	可用资源
0	8000	19.32	77.32	80.0	Resourse_0
1	7000	24.76	81.51	43.75	Resourse_1
2	6000	30.2	85.7	22.5	Resourse_2
3	5000	77.32	131.57	50.0	Resourse_0
4	4000	81.51	134.51	25.0	Resourse_1
5	3000	131.57	183.32	30.0	Resourse_0
6	2000	85.7	136.2	7.5	Resourse_2
7	1000	134.51	183.76	6.25	Resourse_1

8 个任务分别被分配到资源 $R_0, R_1, R_2, R_0, R_1, R_0, R_2, R_1$ 上执行，总运行时间为183.76 s，总开销为 256 cent.

timeWeight 依次设为 0，0.1，0.2，0.3，0.4，0.5，0.6，0.7，0.8，0.9，1 时，运行时间和开销如图 3 所示.

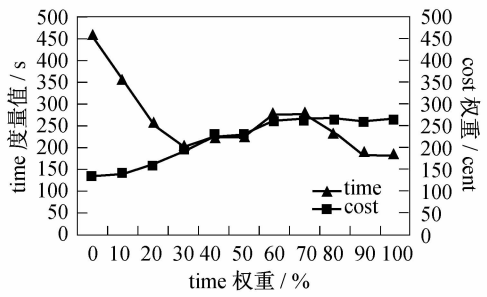

图 3　timeWeight 取不同值的仿真结果

Fig. 3　The simulation of different timeWeight value

从图 3 可知，timeWeight 逐渐变大，总运行时间有变小的趋势，costWeight 逐渐变小，总开销有变大的趋势. 该算法能够根据用户对时间和开销要求的不同作出相应的调度策略，且该调度策略能够较好地满足客户需求.

Gridsim 系统本身也提供了时间优化算法，在相同条件下，与本文所叙的算法运行时间相比较，结果如图 4 所示. 本算法适合于任务数不是特别多，且任务长度有明显差异的情况.

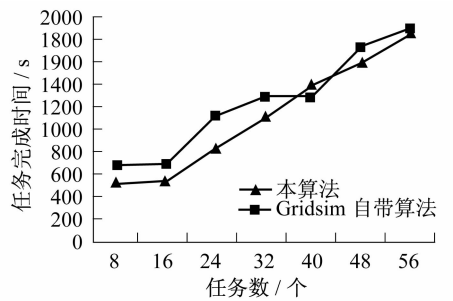

图 4　改进的贪婪算法与 Gridsim 自带算法的比较

Fig. 4 Comparison of improved greedy algorithm and the algorithm in Gridsim

5 总 结

采用贪婪算法,根据网格用户提供的“deadline”,“budget”,“timeWeight”和“costWeight”等参数,以一种最适合用户对该任务需求下的最经济的调度策略,实现尽可能小的调度驱动函数值完成任务,最好地满足客户在不同环境下的需求.当然还可以考虑系统中的其他参数,如线路带宽、线路长短和任务输出长度等.当选择资源进行调度时,选择线路带宽大和线路短的资源进行调度,并把任务输出长度考虑在内.

[参 考 文 献]

[1] LUIS F, VIKTORS B, JONATHAN A. Introduction to grid computing with globus[EB/OL]. [2008-03-30]. http://www.redbooks.ibm.com/redbooks/pdfs/sg246895.pdf.

[2] RAJKUMAR B, DAVID A, JONATHAN G. An economy driven resource management architecture for global computational power Grids [J/OL]. [2008-03-30]. http://www.buyya.com/ecogrid/.

[3] 孔晓红,叶宾,须文波.多目标蚁群优化网格调度算法[J].计算机工程与应用,2007,43(30):88-90.
KONG X H, YE B, XU W B. Ant colony optimization for multi-objective grid scheduling algorithm[J]. Computer Engineering and Applications, 2007,43(30):88-90.

[4] RAJKUMAR B, JONATHAN G, DAVID A. An evaluation of economy-based resource trading and scheduling on computational power grids for parameter sweep applications[J/OL]. [2008-03-30]. http://www.buyya.com/ecogrid/.

[5] SHIE M, RON M, TONG Z. Greedy algorithms for classification-consistency, convergence rates, and adaptivity. [EB/OL]. [2008-03-30]. http://www.ee.technion.ac.il/~rmeir/Publications/Mannor-Meir-Zhang-Jmlr03.pdf.

[6] SARTAJ S. 数据结构、算法与应用[M/OL]. [2008-05-08]. http://www.tjnu.edu.cn/ini/arithetics/No11.htm.
SARTAJ S. Data Structures, Algorithms and Applications[M/OL].[2008-05-08]. http://www.tjnu.edu.cn/ini/arithetics/No 11.htm.

[7] RAJKUMAR B. Grid computing and distributed systems laboratory and the gridbus project. [J/OL]. [2008-03-30]. http://www.gridbus.org/reports/GRIDS-Lab-AnnualReport2007.pdf.

[8] MARK B, RAJKUMAR B, DOMENICO L. Grids and grid technologies for wide-area distributed computing. [J/OL].[2008-05-08]. http://www.gridbus.org/~raj/papers/gridtech.pdf.