第 31 卷 第 6 期 自 动 化 学 报 Vol. 31, No. 6
2005 年 11 月 ACTA AUTOMATICA SINICA November, 2005

一种适于异构环境的任务调度算法[1)]

支 青[1,2] 蒋昌俊[1,2]

[1](同济大学计算机科学与工程系 上海 200092)
[2](国家高性能计算机工程技术研究中心同济分中心 上海 200092)
(E-mail: zhi_qing_@hotmail.com)

摘 要 针对异构环境独立任务调度问题提出两个调度原则, 并基于 Min-min 算法提出优先级最小最早完成时间算法 (Priority min-min,PMM). 该算法将任务在各处理机上执行时间的标准误差作为任务的优先级. 选取最早完成时间较小的 k 个任务, 优先调度其中优先级最高的一个. 在实验基础上分析了参数 k 对 PMM 算法性能的影响. PMM 算法克服了 min-min 算法单纯追求局部最优的局限性, 更适合于异构环境. 实验数据表明 PMM 算法能有效地降低调度跨度, 其性能比 min-min 算法有明显提高.

关键词 调度, 最早完成时间, 最少执行时间, 调度跨度, 标准误差
中图分类号 TP301

A Scheduling Algorithm Suitable for Heterogeneous Computing Environment

ZHI Qing[1,2] JIANG Chang-Jun[1,2]

[1](*Department of Computer Science and Engineering, Tongji University, Shanghai* 200092)
[2](*Tongji Branch, National Engineering & Technology Center of High Performance Computer, Shanghai* 200092)
(E-mail: zhi_qing_@hotmail.com)

Abstract This paper presents two scheduling principles suitable to independent job scheduling in heterogeneous computing environment. A new algorithm named Priority Min-min (PMM) is presented, which computes jobs′ priorities based on their standard deviation of execution time. PMM chooses k jobs which have smaller earliest finish times, and assigns the job with the highest priority to the corresponding processor. This paper analyses how k influences the performance of PMM by experiment. PMM is more suitable for heterogeneous processor platforms by surmounting the limitation of Min-min. The experimental data show that PMM reduces the makespan effectively and its performance is much better than Min-min.

Key words Scheduling, EFT, MET, makespan, standard deviation

1) 国家高技术研究发展计划 (863)(2004AA104340), 国家重点基础研究发展规划 (973)(2003CB316902), 国家自然科学基金重大研究计划 (90412013), 国家杰出青年科学基金 (60125205), 上海科技攻关研究项目 (03DZ15029) 资助
Supported by the National High-Tech Research and Development Plan of P. R. China (2004AA104340), the National Grand Fundamental Research 973 Program of P. R. China (2003CB316902), the National Natural Science Foundation of P. R. China (90412013), the National Science Fund for Distinguished Youth Scholars of P. R. China (60125205), the Shanghai Science & Technology Research Plan of P. R. China (03DZ15029)
收稿日期 2004-11-10 收修改稿日期 2005-9-29
Received November 10, 2004; in revised form September 29, 2005

1 引言

异构环境中的独立任务调度研究多个任务在多个异构处理机上的执行过程, 可分为在线调度方式和批处理调度方式. 前者对任务调度一般不作优化, 来一个任务就调度一个; 后者强调对任务进行优化调度, 使任务与资源更好地匹配. 若任务的到达率较低, 多处理机系统的计算能力绰绰有余, 则可以采用在线调度方式; 若任务的到达率较高, 多处理机系统无法立即执行接收到的任务, 则应该采用批处理调度方式. 批处理方式比在线方式更能充分利用计算资源.

本文对异构环境下批处理方式的任务调度进行研究, 对最小最早完成时间算法 (min-min)[1] 加以改进, 提出优先级最小最早完成时间算法 (Priority min-min, PMM). 在集群系统中, 任务在各处理机上的执行时间有较大的差异. 在异构计算环境中, 差异更为明显. 本文使用任务执行时间的标准误差 (Standard deviation of execution time, 以下简称 SDET) 反映计算环境的异构性. PMM 算法在综合权衡任务最早完成时间和 SDET 的基础上进行调度, 更适合于异构环境. 实验表明, PMM 算法性能比原先的 Min-min 算法有明显提高.

2 相关工作

目前国内外有多种针对独立任务的调度算法, 例如 MCT(Minimum completion time), MET(Minimum execution time)[2], SA(Switching algorithm), KPB(K-percent best)[3], OLB (Opportunistic load balancing)[3], Max-min[1], Min-min, DMSA(双最小平衡算法)[4], GDDS(全局动态分布调度算法)[5] 等. 这些算法通常权衡任务的最少执行时间 (Minimum execution time, MET) 和最早完成时间 (Earliest finish time, EFT) 两个方面, 各有其优缺点.

MCT 算法来源于 SmartNet[1] 中的启发式算法. 该算法以任务的最早完成时间为标准进行调度, 可以保证多处理机间的负载均衡, 但任务的实际执行时间并非最少.

MET 算法以任务的最少执行时间为标准进行调度. 该算法会导致大量的任务集中到某些性能较高的处理机上, 使得其它处理机空闲, 造成系统的负载不均衡.

SA 算法是 MCT 和 MET 的综合. 当系统负载均衡时, 使用 MET 算法; 当负载不均时, 使用 MCT 算法. 整个系统在负载均衡与不均衡间波动.

KPB 算法界于 MCT 和 MET 之间. 该算法按 $k\%$ 的比例选取任务执行时间较小的若干处理器, 再选取最早完成时间最小的处理器进行调度. 当 $k = 100/m(m$ 为处理机数) 时, 该算法退化为 MET 算法; 当 $k = 100$ 时, 该算法退化为 MCT 算法.

OLB 算法不考虑任务执行情况, 只是简单地将任务分配到下一个即将空闲的处理机上. 若有多个处理机均空闲, 则随机选取一个. 该算法可以实现负载均衡, 但由于没有考虑任务执行时间和完成时间, 难以实现任务与处理机的最佳匹配.

Max-min 算法是 MCT 的改进算法. 该算法优先选取最早完成时间最大的任务进行调度. 在短任务多、长任务少的情况下, 可以使长任务先被调度, 使之与多个短任务并行执行, 从而达到较好的性能. 在短任务和长任务数量相当的情况下, 从本文的测试结果可以看出该算法性能较低.

Min-min 算法也是 MCT 的改进算法. 该算法优先选取最早完成时间最小的任务进行调度. 在同类算法中, Min-min 算法具有较好的性能, 一般用作为调度算法的评测基准[3,6]. 本文对该算法进一步改进, 提出 PMM 算法.

DMSA 算法追求两个目标值最小, 一个是各处理机负荷的总和最小, 另一个是各处理机负荷的差值最小. 这两个目标相互制约, 难以同时满足.

GDDS 算法是基于异构计算系统的调度策略. 该算法以任务执行时间最小为目标, 但未考虑任务完成时间, 可能导致负载不均. 该算法的缺陷与 MET 类似.

3　PMM 调度算法

任务集 J 有 n 个任务为 $\{j_1, j_2, \cdots, j_n\}$, 任务间没有通信, 一个任务只在一个处理机上执行. 待调度的任务集 J' 是任务集的子集, 有 s 个任务 $\{j_{r_1}, j_{r_2}, \cdots, j_{r_s}\}$. 在调度开始之前, $J' = J$, $s = n$. 处理机集 P 有 m 个处理机为 $\{p_1, p_2, \cdots, p_m\}$, 每个处理机一次只能执行一个任务, 且任务不能被抢断. PMM 算法中预选任务数为 k.

各任务在各处理机上的执行时间用矩阵 $E_{n\times m}$ 表示, e_{ik} 记录任务 j_i 在处理机 p_k 上的执行时间. 各处理机最早空闲时间用矩阵 $C_{1\times m}$ 表示. 当任务 j_i 被调度到处理机 p_k 上执行时, $c_k(t+1) = c_k(t) + e_{ik}$. C 的初值为 0.

调度跨度 (makespan) 是分布式异构计算系统任务吞吐率的量度标准, 定义为 $M = \max\limits_{1\leqslant i\leqslant n}\{f_i\}$[7], 其中 f_i 是任务 j_i 的完成时间. 本文以调度跨度作为调度算法性能的评价标准.

3.1　Min-min 算法存在的问题

Min-min 算法本质上是一种贪心算法, 以任务的最早完成时间最小为目标, 做出在当前看来是最优的选择. 算法对任务的执行时间考虑不足, 为了达到任务最早完成时间最小的目标, 往往将任务调度到执行时间较长的处理机上去, 造成 makespan 增加.

现通过一个例子来反映 min-min 算法的这个缺陷. 假设处理机集 P 中有两个处理机 p_1, p_2. 调度完若干个任务后, p_1, p_2 的最早空闲时间分别为 30, 40. 待调度的任务集 J' 中还剩有两个任务 j_{r_1}, j_{r_2}. j_{r_1} 在 p_1, p_2 上的执行时间分别为 20, 15; j_{r_2} 在 p_1, p_2 上的执行时间分别为 25, 30.

Min-min 算法先将 j_{r_1} 调度到 p_1 上, 使得 j_{r_1} 的最早完成时间为 50. 然后将 j_{r_2} 调度到 p_2 上. 调度跨度为 70, 如图 1 中左图所示. Min-min 算法调度方案使得 j_{r_1}, j_{r_2} 两个任务恰好均在执行时间最长的处理机上运行, 增加了调度跨度.

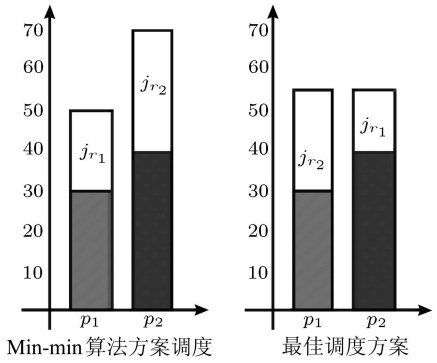

图 1　两种任务调度方案比较

Fig. 1　Comparison of two scheduling results

最佳调度方案应该是 j_{r_1} 调度到 p_2 上, j_{r_2} 调度到 p_1 上, 如图 1 中右图所示. 尽管 j_{r_1} 的最早完成时间为 55, 大于 min-min 算法中的 50, 但调度跨度仅为 55, 明显优于 min-min 算法.

从上述比较可以看出, Min-min 算法对任务执行时间考虑不足, 单纯追求当前局部最

优，给全局性能带来了负面影响．当任务在各处理机上的执行时间有较大的差异时，若将任务调度到执行时间较多的处理机上，更容易造成调度性能的下降．计算环境的异构性越大，Min-min 算法这一缺陷的负面影响就越大．

3.2 PMM 算法对 min-min 算法的改进

任务调度本质是以负载均衡和高吞吐率为目标，将任务与资源匹配，使资源得到最合理的利用．针对异构环境的任务调度问题，可以提出以下两个调度原则．

原则 1. 负载均衡原则．任务应该分配到空闲时间较早的处理机上，使得各处理机运行时间相当．

原则 2. 高吞吐率原则．任务应该分配到执行时间较少的处理机上，使任务尽快执行完．

定义 1. PMM 算法中，任务的优先级为任务在各处理机上执行时间的标准误差 (即 SDET). 第 i 个任务的优先级记作 $priority(j_i)$. $priority(j_i) = v_i$, 其中

$$\bar{e}_i = \frac{\sum_{k=1}^{m} e_{ik}}{m} \tag{1}$$

$$v_i = \sqrt{\frac{\sum_{k=1}^{m}(e_{ik} - \bar{e}_i)^2}{m}} \tag{2}$$

SDET 是任务执行时间矩阵 E 的每行数值的标准误差，反映了任务在各处理机上执行时间的差异，即反映了计算环境的异构性．在调度中权衡 SDET, 可以避免 Min-min 算法在异构环境中的缺陷．此外，SDET 是任务在所有处理机上执行时间的全局性评定，可以在一定程度上克服 Min-min 算法追求局部最优的局限性．SDET 较小的任务在各处理机上的执行时间差距较小．这些任务无论调度到哪个处理机都相差无几，对调度跨度影响不大．SDET 较大的任务的执行时间差距较大．若这些任务被调度到执行时间较多的处理机上，容易造成调度跨度的增加．PMM 算法根据原则 2, 优先调度 SDET 较大的任务，使之尽可能调度到执行时间较少的处理机上．

在任务调度中，一般难以同时满足原则 1 和原则 2.PMM 算法可以通过调整参数实现两个原则之间权衡．PMM 算法首先计算各任务的最早完成时间，选出其中最早完成时间较小的 k 个任务作为备选任务．调度备选任务中任意一个都不会造成负载不均，满足原则 1. 然后考察备选任务的执行时间，优先调度 SDET 最大 (即优先级最高) 的一个任务，满足原则 2. PMM 算法中的 k 是一个可以调节的参数，取值范围为 $[1, s]$. k 越小，算法越偏向原则 1; k 越大，算法越偏向原则 2. 稍后将会对 k 的取值进行测试及分析．

3.3 PMM 算法的伪代码描述

1) Procedure PMM()
2) $J' = J$, $s = n$, $C = 0$;
3) for every $j_i (j_i \in J')$
4) $\quad \bar{e} = \frac{\sum_{k=1}^{m} e_{ik}}{m}$, $v = \sqrt{\frac{\sum_{k=1}^{m}(e_{ik} - \bar{e}_i)^2}{m}}$;
5) $\quad priority(j_i) = v_i$;
6) endfor
7) while $J' \neq \varnothing$

8)　if ($s > k$)
9)　　for every $j_i(j_i \in J')$
10)　　for every $p_q(p_q \in P)$
11)　　计算 j_i 在 p_q 上的完成时间;
12)　　endfor
13)　　计算 j_i 的 EFT;
14)　　endfor
15)　　选取 EFT 较小的 k 个任务 $(j_{s1}, p_{q1}), (p_{s2}, j_{q2}) \cdots (j_{sk}, p_{qk})$;
16)　在 k 个任务中选取优先级最高的一个任务 j_{si};
17)　else
18)　在 J' 中选取优先级最高的一个任务 j_{si};
19)　endif
20)　将 j_{si} 分配给 p_{qi};
21)　$J' = J' - \{j_{si}\}$, $s = s - 1$;
22)　$c_{qi} = c_{qi} + e_{si\,q_i}$;
23) endwhile
24) end procedure

3.4　PMM 算法的分析

性质 1. PMM 算法保持任务的优先级不变.

证明. 根据定义 1, 任务优先级 $priority(j_i) = v_i$. 各任务在各处理机上的执行时间不随实际调度方案变化, 由公式 (1) 、 (2) 可知任务的 SDET 是确定的, 所以任务的优先级不变. 证毕.

性质 2. 当 $k = 1$ 时, PMM 算法退化为 min-min 算法.

证明. 当 $k = 1$ 时, 每次只取出最早完成时间最小的一个任务进行调度. SDET 对调度不起作用, 即算法退化为 min-min 算法. 证毕.

性质 3. 当 $k = s$ 时, PMM 算法退化为静态优先级算法.

证明. 当 $k = s$ 时, 在所有未调度的任务中选取优先级最高的任务进行调度. 任务最早完成时间对调度不起作用, 又由性质 1 可知任务优先级不变, 即算法退化为静态优先级算法. 证毕.

定理 1. PMM 算法的时间复杂度为 $O((k+m)n^2)$, 其中 k 为预选任务数, m 为处理机数, n 为任务数.

证明. 算法 2) 行, 赋初值. 算法 3)~6) 行, 循环 n 次, 计算各任务的优先级. 根据性质 1, 可以预先计算任务的优先级. 时间复杂度为 $O(n)$. 算法 7)~23) 行, 循环 n 次, 调度各任务, 时间复杂度为 $O((k+m)n^2)$. 其中 9)~14) 行, 计算各任务在各处理机上的完成时间, 并计算 EFT, 复杂度为 $O(mn)$. 15) 行, 在未调度的任务中, 选取 k 个最早完成时间较小的任务, 复杂度为 $O(kn)$. 16) 行, 在 k 个备选任务中, 选取优先级最高的任务, 复杂度为 $O(k)$. 18) 行, 在剩余任务中 (剩余任务数 $\leqslant k$), 选取优先级最高的任务, 复杂度为 $O(k)$. 综上所述, PMM 算法的时间复杂度为 $O((k+m)n^2)$. 证毕.

3.5 实例运行

现通过一个实例，具体说明 PMM 算法的调度过程. 假设 $n=4, m=4, k=3$, 各任务在各处理机上的执行时间如表 1 所示.

表 1 任务的执行时间
Table 1 Execution time of jobs

	p_1	p_2	p_3	p_4
j_1	5.3	6.7	8.6	9.8
j_2	6.2	7.5	8.4	9.2
j_3	5.2	6.3	8.0	8.9
j_4	5.7	6.4	7.2	8.3

PMM 算法先计算任务 j_1, j_2, j_3, j_4 的优先级，分别为 1.728,1.115,1.440,0.967. 最早完成时间较小的 3 个任务是 j_1, j_3, j_4, 其中任务 j_1 的优先级最高，所以先将 j_1 调度到 p_1 处理机上 $j_1 -> p_1$. 剩余的 3 个任务按照任务优先级由高到低依次调度. 调度方案为 $j_3 -> p_2, j_2 -> p_3, j_4 -> p_4$, 调度跨度 $M=8.4$. 若使用 Min-min 算法对本实例进行调度，调度方案为 $j_3 -> p_1, j_4 -> p_2, j_2 -> p_3, j_1 -> p_4$, 调度跨度 $M=9.8$. 在该实例中，PMM 算法的跨度小于 min-min 算法. 下面通过测试，比较 PMM 、min-min 、max-min 三种算法的性能.

4 PMM 算法测试及分析

4.1 模拟异构环境

异构性可分为处理机异构性和任务异构性[3]. 处理机异构性是指同一个任务在不同的处理机上的执行时间之间的差异，任务异构性是指同一个处理机上的不同任务的执行时间之间的差异. 任务执行时间矩阵 E 的每一行数值的标准误差 (即 SDET) 反映了处理机异构性，E 的每一列数值的标准误差反映了任务异构性.

在本文测试中, 任务执行时间矩阵 E 由计算机模拟生成, 算法如下所示. 算法中 Hp, Hj 分别为处理机异构系数和任务异构系数[3], 实际测试中均取值为 10.

for every $j_i (j_i \in J)$
  $Nj[i] \leftarrow Random[1, Hj]$;
endfor
for every $j_i (j_i \in J)$
  for every $p_q (p_q \in P)$
    $e_{iq} \leftarrow Nj[i]^* Random[1, Hp]$;
  endfor
endfor

4.2 参数选择

由 PMM 算法思想可知，参数 k 值较小时，算法较偏向原则 1; k 值较大时，算法较偏向原则 2. 所以，k 的不同取值会影响 PMM 算法的性能. 由性质 2 和性质 3 可知，k 取极值时， PMM 算法性质发生变化. 以下通过实验确定 k 的取值，使 PMM 算法达到较高的性能. 限于篇幅，本文在 n 分别为 100 、 200 、 500, $m=32$ 的前提下，对 k 值为 2 、 3 、 4 、 5 、 10 、 20 进行分析. PMM 算法在各 k 值下的调度跨度比较如图 2 所示.

测试数据表明，在 $m=32, n \leqslant 500$ 时，k 的最佳取值为 3 或 4. 当 $k=2$ 时，由于备选任务不足，PMM 算法没有很好地发挥其优势，性能较低. 当 k 等于 3 或 4 时，PMM 算法

可以达到较好的性能. 任务数较多时, k 取 4 性能更好些. 当 k 值在 [5,20] 范围内逐渐增大时, PMM 算法过于偏离原则 1, 性能逐渐下降.

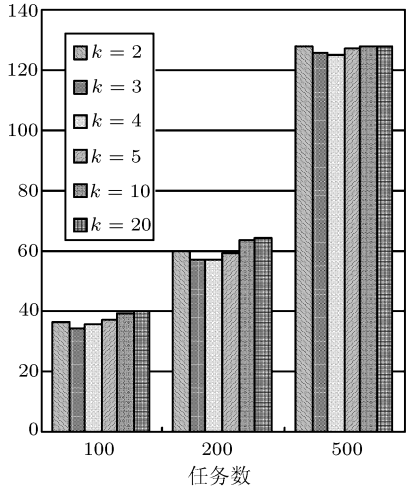

图 2　k 值对 PMM 算法的影响

Fig. 2　k influences the performance of PMM

4.3　PMM 算法测试

本测试比较 PMM, Min-min, Max-min 三个调度算法的 makespan. 这三种算法作为启发式算法, 并不能保证每次都能找到最优解决方案, 所以本文比较各算法多次运行的 makespan 的平均值. 由上文可知, k 等于 3 或 4 时, PMM 算法性能较高. 在以下测试中, PMM 算法中参数 k 取值为 4. 测试按处理机数分为 3 组, 每组测试再按任务数分为 3 小组.

第 1 组测试, $m = 16$, n 分别为 50 、 100 、 200; 第 2 组测试, $m = 32$, n 分别为 100 、 200 、 500; 第 3 组测试, $m = 64$, n 分别为 200 、 500 、 800. 三组测试的结果见图 3∼ 图 5.

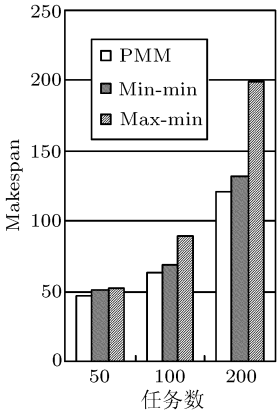

图3 调度跨度比较 (第 1 组测试)

Fig. 3 Comparison of makespan of algorithms (Test 1)

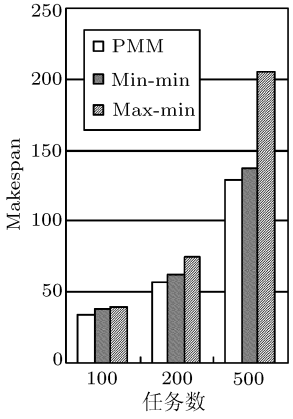

图4 调度跨度比较 (第 2 组测试)

Fig. 4 Comparison of makespan of algorithms(Test 2)

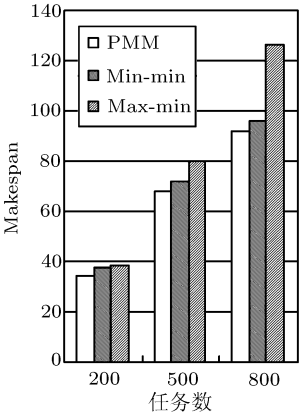

图5 调度跨度比较 (第 3 组测试)

Fig. 5 Comparison of makespan of algorithms(Test 3)

4.4 测试结果分析

测试数据表明 PMM 和 min-min 算法的跨度远小于 max-min 算法的跨度，PMM 的跨度比 Min-min 更小．PMM 算法在遵守两个调度原则的基础上进行任务调度，有效地降低了调度跨度．在不同的处理机数量规模 (16∼64) 和任务数量规模 (50∼800) 情况下，当算法中参数 k 取值为 4 时，PMM 算法的性能比 min-min 算法提高了 6% 左右．

5 结论

本文对 min-min 算法加以改进，提出了 PMM 调度算法．PMM 算法以 SDET 作为任务优先级，对最早完成时间较小的 k 个备选任务按优先级进行调度．SDET 既反映计算环境的异构性，又是对任务在所有处理机上执行情况的全局性评价．PMM 算法以任务最早完成时间和 SDET 为任务选择的约束条件，既克服 min-min 算法在异构性环境中的不足，又克服 min-min 算法片面追求局部最优的局限性．此外，通过调整 PMM 算法中参数 k 的值，可以使 PMM 算法在两个调度原则之间取得平衡．实验表明，在异构计算环境中 PMM 算法比 min-min 算法具有更好的性能．

References

1 Freund R F, Gherrity M, Ambrosius S, Campbell M, Halderman M, Hensgen D, Keith E, Kidd T, Kussow M, Lima J D, Mirabile F, Moore L, Rust B, Siegel H J. Scheduling resources in multi-user, heterogeneous, computing environments with smartNet. In: Proceedings of the Seventh IEEE Heterogeneous Computing Workshop. Washington, DC: IEEE Computer Society, 1998, 184∼199

2 Armstrong R, Hensgen D, Kidd T. The relative performance of various mapping algorithms is independent of sizable variances in run-time predications. In: Proceedings of the Seventh IEEE Heterogeneous Computing Workshop. Washington, DC: IEEE Computer Society, 1998. 79∼87

3 Muthucumaru M, Shoukat A, Howard J S, Debra H, Richard F F. Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. In: Proceedings of the Eighth IEEE Heterogeneous Computing Workshop. Washington, DC: IEEE Computer Society, 1999, 30∼44

4 Jiang S J, Xu X F. Double minimum-balance scheduling algorithm for a class of workload balance problems. *High Technology Letters*, 2002, **12**(7): 53∼57 (in Chinese)

5 Wang D Q, Wei J B. Research on dynamic scheduling policy on distributed computer system. *Journal of Huazhong University of Science and Technology*, 2001, **29**(6): 87∼89 (in Chinese)

6 Ibarra O H, Kim C E. Heuristic algorithms for scheduling independent tasks on nonidentical processors. *Journal of the Association for Computing Machinery*, 1977, **24**(2): 280∼289

7 Pinedo M. Scheduling: Theory, Algorithms, and Systems. New Jersey: Prentice Hall. 1995

支 青 同济大学计算机系硕士研究生，研究领域为并行计算，网格计算．

(**ZHI Qing** Master student in Department of Computer Science and Engineering at Tongji University. His research interests include parallel computing and grid computing.)

蒋昌俊 1995 年于中国科学院自动化研究所获得博士学位，现为同济大学电信学院院长，教授，博士生导师．研究领域为并发理论，Petri 网理论与应用，网格计算．

(**JIANG Chang-Jun** Received his Ph. D. degree from Institute of Automation, Chinese Academy of Sciences in 1995. Now he is professor and chairman in School of Electric and Information Engineering at Tongji University. His research interests include theory of concurrency, theory and application of Petri nets, and grid computing.)