异构环境下独立任务调度算法的研究*)

周 洋[1,2,3] 蒋昌俊[1,2,3] 方 钰[1,2,3]

(同济大学计算机科学与工程系 上海 201804)[1]

(国家高性能计算机工程技术研究中心同济分中心 上海 201804)[2]

(同济大学嵌入式系统与服务计算教育部重点实验室 上海 201804)[3]

摘 要 本文基于 Min-min 算法和 Sufferage 算法提出了基于任务调度损失的最小最早完成时间算法(Sufferage Min-min,SMM)。该算法将任务调度损失引入 Min-min 算法,选取最早完成时间较小的 k 个任务,再优先对其中任务调度损失最大的一个进行调度。SMM 算法克服了 Min-min 算法单纯追求局部最优而缺少全局意识的缺点。测试表明,SMM 算法可以做到调度跨度低与平均等待时间小的统一,在综合性能上较 Min-min 算法有所提高。

关键词 调度算法,Min-min 算法,Sufferage 算法,调度跨度,平均等待时间

Research of Scheduling of Independent Tasks onto Heterogeneous Computing Systems

ZHOU Yang[1,2,3] JIANG Chang-jun[1,2,3] FANG Yu[1,2,3]

(Department of Computer Science and Engineering, Tongji University, Shanghai 201804, China)[1]

(Tongji Branch, National Engineering & Technology Center of High Performance Computer, Shanghai 201804, China)[2]

(The Key Laboratory of Embedded System and Service Computing, Ministry of Education, Shanghai 201804, China)[3]

Abstract This paper presents a new heuristic called Sufferage Min-min (SMM), which is based on both Min-min heuristic and Sufferage heuristic. Combining task sufferage with Min-min heuristic, SMM chooses k tasks which have smaller earliest finish times, and assigns the task, which will be suffered most if it isn't assigned, to the corresponding processor. SMM is more suitable for heterogeneous processors platforms by surmounting the limitation of Min-min heuristic. The statistic from the experiments proves that SMM can reach lower makespan and less average waiting time at the same time and the performance of it is better than the one of Min-min heuristic.

Keywords Scheduling, Min-min heuristic, Sufferage heuristic, Makespan, Average waiting time

1 引言

目前国内外有关独立任务的调度算法大体上可分为在线模式(on-line mode)和批处理模式(batch-mode)两类。

在线模式在任务到达调度器的第一时间就加以映射。经典的在线模式调度算法有:MCT(Minimum completion time),MET(Minimum execution time)[1],SA(Switching algorithm),KPB(K-percent best)[2]和 OLB(Opportunistic load balancing)[2]。MCT 算法以任务的最早完成时间作为标准进行调度,可以保证多处理机的负载平衡,但任务的实际执行时间并非最少。MET 算法则以任务最少执行时间为标准调度任务,但该算法容易造成系统的负载不均衡。SA 算法是 MCT 和 MET 的综合,当负载均衡时使用 MET,当负载不均时使用 MCT,但该算法会导致整个系统在负载均衡与不均衡间不停波动。KPB 算法介于 MCT 和 MET 之间,该算法按 $k\%$ 的比例选取任务执行时间较小的处理机,再选取最早完成时间最小的处理机进行调度。OLB 算法不考虑任务执行情况,仅是将任务分配到下一个即将空闲的处理机上,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,因此很难实现任务与处理机的最佳匹配。

批处理模式调度算法会把任务收集起来,等映射事件发生后才集中映射所有任务。经典的批处理模式调度算法有:Min-min[3],Max-min[3] 和 Sufferage[4]。Min-min 算法是 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,一般用作调度算法的评测基准[2,5]。Max-min 算法也是 MCT 的改进算法,与 Min-min 算法不同,该算法选取最早完成时间最大的任务进行调度。Sufferage 算法认为最好的映射算法应该是将机器分配给一个任务,如果该机器不分配给此任务,则此任务在期望完成时间上遭受的损失是所有任务中最大的。该算法在减小任务调度跨度上的性能优于其他批处理算法。

本文对异构环境下批处理方式的任务调度算法进行了研究。结合 Min-min 算法和 Sufferage 算法的思想,提出了基于任务调度损失的最小最早完成时间算法(Sufferage Min-min,SMM)。本文将任务调度损失引入了 Min-min 算法中,在综合权衡任务最早完成时间与任务调度损失的基础上进行调度,使算法更适合于异构环境。测试表明,SMM 算法具有较好的综合性能。

2 SMM 算法(Sufferage Min-min algorithm)

2.1 算法相关概念

异构环境中的独立任务调度问题实际上是研究 n 个独立的任务在 m 台处理机上的执行过程。为了方便说明本文的

*)基金项目:国家发改委项目(CNGI-04-15-5A);上海市科委重大项目(05DZ15007);上海市科委重大项目(05DZ15004)。周 洋 硕士生,研究领域为并行计算、网格计算;蒋昌俊 博士,教授,博士生导师,研究领域为并行处理、并发理论、Petri 网理论、网格计算;方 钰 博士,讲师,研究领域为嵌入式系统、移动计算。

算法，现给出如下概念。

设任务集 T 中包含 n 个任务$\{t_1,t_2,\cdots,t_n\}$，任务间彼此没有通信，且一个任务只在一台处理机上执行。处理机集 P 中包含 m 台处理机$\{p_1,p_2,\cdots,p_m\}$，每台处理机一次只能执行一个任务，且任务不可以被中断或者转移。各任务在各台处理机上的执行时间用执行时间矩阵(Expected execution time matrix)$E_{n\times m}$表示，e_{ij} 表示任务 t_i 在处理机 p_j 上的执行时间。各处理机的最早空闲时间用矩阵 $K_{1\times m}$表示，其中 K_j 表示处理机 p_j 的最早空闲时间。各任务在各处理机上的最早完成时间用最早完成时间矩阵(Expected completion time matrix)$C_{n\times m}$表示，c_{ij} 表示处理机 p_j 处理完任务 t_i 及之前分配到 p_j 上的所有任务所需的时间，$c_{ij}=K_j+e_{ij}$。

调度方案的调度跨度(makespan)定义为$\max\limits_{1<i<n}(F(t_i))$，其中 $F(t_i)$表示任务 t_i 的完成时间。调度跨度是衡量异构计算系统任务吞吐率的重要指标，但是该指标不能反应映单个任务的服务质量。

任务的等待时间(waiting time)被定义为任务的完成时间与任务在该处理机上的执行时间的差。设任务 i 被分配至处理机 j，且任务 i 在处理机 j 上的完成时间为 c_{ij}，则任务 i 的等待时间 $w_i=c_{ij}-e_{ij}$。任务的平均等待时间是任务调度方案中所有任务的等待时间的平均，即$(\sum_{t_i\in T}w_i)/|T|$。调度方案的平均等待时间反映的是调度算法减小任务调度对具体任务影响程度的能力。在本文后续章节中将给出对于SMM算法在调度跨度和平均等待时间上的性能分析。

2.2 Min-min 算法的不足

传统的Min-min算法在不同的执行时间矩阵下具有较好的性能[5]，也是目前网格调度算法的基础之一。其算法的思路是映射尽可能多的任务到能最快完成该任务的处理机上从而使每次任务调度对处理机的最早空闲时间影响最小。在这个思路下任务被分配到执行时间最短的处理机上的概率也是其他各类传统启发式调度算法中最高的。

但是Min-min算法本质上是一种贪心算法，其片面追求完成时间最早的局部最优解，而影响到了算法的负载均衡(load balancing)性能，造成了调度跨度值的增加。考虑表1中所示的执行时间矩阵，若使用Min-min算法，t_1 会被调度到 p_1 上，然后 t_2 又被调度到 p_1 上，这种调度方法使得调度跨度为60，同时造成 p_2 计算能力的浪费，如图1(a)所示。而最优调度方案则是 t_1 调度到 p_2，t_2 调度到 p_1，调度跨度为40，如图1(b)。

表1 执行时间矩阵

	p1	p2
t1	20	40
t2	40	70

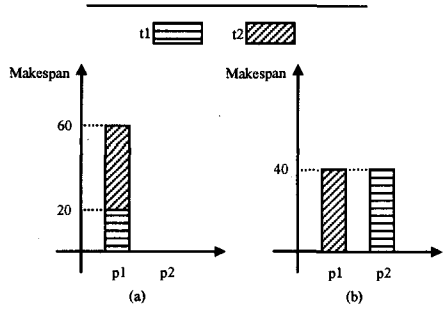

图1 Min-min算法调度与最优调度的调度跨度

从上面的实例可以发现，Min-min算法单纯追求局部当前最优，而对全局性能考虑不足。当任务在各处理机上执行时间存在较大差异的时候，Min-min的这一负面影响就会越明显。SMM算法所要做的工作就是寻找一种在保留Min-min算法较高吞吐率的基础上提高其负载均衡的能力方法。

2.3 SMM 算法对 Min-min 算法的改进

为了更好地说明SMM算法对传统Min-min算法的改进，先给出如下定义。

定义1 设在异构环境中，待调度任务集为 $T\{t_1,t_2,\cdots,t_n\}$，处理机集为 $P\{p_1,p_2,\cdots,p_m\}$。定义任务集 T 的完成时间最小匹配集合 $\min CT(T,P)$为：

$$\min CT(T,P)=\{(t_i,p_j)\mid K_j+e_{ij}=\min_{p_q\in P}(K_q+e_{iq}),t_i\in T,p_j\in P\}$$

其中元素(t_i,p_j)表示任务 t_i 在处理机 p_j 上的完成时间早于 t_i 在其他任何处理机上的完成时间，则称 t_i 和 p_j 实现了完成时间最小匹配，记作(t_i,p_j)。一个任务至少与一个处理机实现完成时间最小匹配。集合 $\min CT(T,P)$表示待调度任务集 T 中各任务的完成时间最小匹配的集合。

定义2 在SMM算法中，第 i 个任务 t_i 的损失度(sufferage)被定义为，任务 t_i 的次优最早完成时间(不失一般性，设任务 t_i 被调度到处理机 p_x)与最优最早完成时间(不失一般性，设任务 t_i 被调度到处理机 p_y)的差。即：

$$\text{sufferage}(t_i)=c_{ix}-c_{iy}$$

任务的损失度反映的是任务在较理想处理机上执行完成时间上所存在的差异，利用损失度可以反映出计算环境的异构性。同时损失度是在最优解及次优解的基础上计算得出的综合评定，也可以在一定程度上克服Min-min算法单纯追求局部当前最优，而对全局性能考虑不足的毛病。任务调度损失较小的任务，无论调度到最优解或者次优解处理机在性能上都相差无几，也不会对调度跨度带来较大影响。而任务调度损失较大的任务，由于次优解在性能上远远落后于最优解，若该任务被调至次优处理机上，对全局调度跨度影响也就较损失度小的任务大，因此SMM算法选择优先调度任务调度损失较大的任务。

SMM算法的基本思想是：首先计算待调度任务集 $T\{t_1,t_2,\cdots,t_n\}$中各任务的最早完成时间，得到集合 $\min CT(T,P)$，并选出其中最小完成时间较小的前 k 个元素作为备选集。再从备选集中挑出 sufferage(t)最大的元素作为优先调度的任务。该算法中参数 k 是一个可调节的参数，其范围在$[1,n]$间，k 的取值影响到算法的性能，在下面各节中将给出对参数 k 的取值分析与测试。

SMM算法将任务损失度引入Min-min算法同时利用参数 k 作为调节变量，做到调度跨度低与平均等待时间小的统一，提高了算法对异构环境的适应能力，同时也能满足不同任务调度的需求。

2.4 SMM 算法伪码实现

```
(1)FOR every ti(ti∈T)
(2)   FOR every pj(pj∈P)
(3)     cij=Kj+eij;
(4)   ENDFOR
(5)ENDFOR
(6)DO UNTIL 待调度任务集 T=ϕ
(7)IF T 的元素个数大于 k
(8)   minCT(T,P)=ϕ;
(9)     FOR every ti(ti∈T)
(10)      计算 sufferage(ti)及 ti 的完成时间最小匹配(ti,pj),并将其
      加入集合 minCT(T,P);
(11)  ENDFOR
(12)  选取 minCT(T,P)中最早完成时间较小的 k 个匹配;
(13)  在 k 个匹配中选取 sufferage(t)最大的一个匹配(tp,pq);
```

```
(14)ELSE
(15)   FOR every t_i(t_i∈T)
(16)     计算 sufferage(t_i);
(17)   ENDFOR
(18)   在 T 中选取 sufferage(t)最大的一个匹配(t_p,p_q);
(19)ENDIF
(20)   将 t_p 分配给处理机 p_q;
(21)   T=T-{t_p};
(22)   K_q=K_q+e_pq;
(23)   FOR every r in c_rq
(24)     c_rq=c_rq+e_rq;
(25)   ENDFOR
(26)ENDDO
```

2.5 SMM 算法分析

根据上节所给出的 SMM 算法伪码可以发现 SMM 算法具有如下性质。

性质 1 算法中各任务的损失度 sufferage 随调度的过程动态改变。

根据定义 2 可知任务的损失度(sufferage)是次优最早完成时间与最优最早完成时间的差所决定的。而根据算法伪码(23)-(25)可知任务的最早完成时间是随着调度的过程动态改变的,因此算法中的任务调度损失也随算法的过程动态改变。算法的这个性质有利于算法动态的适应不同的调度状态,更好地克服 Min-min 算法在负载均衡上的不足。

性质 2 $k=1$ 时,SMM 算法退化为传统 Min-min 算法。

当 $k=1$ 时算法每次只取最早完成时间最小的一个任务进行调度,任务调度损失对调度的结果不造成影响。算法的结果和传统 Min-min 算法一致,即退化为 Min-min 算法。

性质 3 $k=n$ 时,SMM 算法退化为 Sufferage 算法。

当 $k=n$ 时算法每次都对所有待调度任务中任务调度损失最大的一个进行调度,任务的最早完成时间对调度任务的选择不起作用,算法结果和传统 Sufferage 算法一致,即算法退化为 Sufferage 算法。

根据上节给出的算法伪码也可以计算出在待调度任务集 T 中包含 n 个待调度任务,处理机集 P 中包含 m 台处理机的条件下,算法的时间复杂度为 $O(n^2(m+k))$。因为,代码(1)-(5)行初始化最早完成时间矩阵 C_{rm},时间复杂度为 $O(nm)$。代码(6)-(26)行,循环 n 次调度各任务,时间复杂度为 $O(n^2(m+k))$。其中(9)-(11)行计算各任务的完成时间最小匹配及任务调度损失,复杂度为 $O(nm)$,代码(12)计算最早完成时间较小的 k 个匹配,复杂度为 $O(kn)$,代码(13)寻找 k 个匹配中最大的一个,复杂度为 $O(k)$。代码(15)-(18)是在待调度集元素个数小于 k 时的调度过程,复杂度为 $O(k(m+1))$。综上所述,SMM 算法的时间复杂度为 $O(n^2(m+k))$。

2.6 实例说明

为了具体说明 SMM 算法的调度过程,现给出一个具体实例。其各任务在各处理机上的执行时间如表 2 所示。

表 2 任务的执行时间

	p_1	p_2	p_3	p_4
t_1	53	66	86	98
t_2	62	76	84	97
t_3	52	63	80	89
t_4	57	64	72	83

假设 $k=2$,根据 SMM 算法首先计算出各任务的完成时间最小匹配集合$\{(t_1,p_1),(t_2,p_1),(t_3,p_1),(t_4,p_1)\}$,再计算出最早完成时间较小的前 2 个匹配$(t_1,p_1)$和$(t_3,p_1)$。然后选出 t_1 与 t_3 中任务调度损失最大的匹配(t_1,p_1),最后将任务 t_1 分配给处理机 p_1。使用相同方法可以得到 t_3 的调度方案 $t_3\rightarrow p_2$。此时待分配任务集中的元素个数等于 k,于是利用 sufferage 算法计算出余下分配方案为 $t_2\rightarrow p_3$ 和 $t_4\rightarrow p_4$,该任务分配方案调度跨度为 84。若对与上例使用 Min-min 算法所得出的调度方案为:$t_3\rightarrow p_1$,$t_4\rightarrow p_2$,$t_2\rightarrow p_3$,$t_1\rightarrow p_4$,调度跨度为 98。而使用 Sufferage 算法可得到调度跨度为 89 的调度方案:$t_2\rightarrow p_1$,$t_1\rightarrow p_2$,$t_4\rightarrow p_3$,$t_3\rightarrow p_4$。在该实例中,SMM 算法的跨度小于 Min-min 算法和 sufferage 算法。下面将通过测试,具体比较 SMM 算法与 Min-min 算法、Max-min 算法以及 Sufferage 算法的性能。

3 SMM 算法测试(Experiments of SMM algorithm)

3.1 测试环境

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。任务的异构性是指同一个处理机上不同任务的执行时间的差异。在本文的测试中,任务执行时间矩阵 E 由计算机模拟生成,为了同时考虑任务异构性与处理机异构性,在矩阵 E 产生的过程中引入了任务异构系数 H_t 和处理机异构系数 H_p[3]。模拟任务执行时间矩阵 E 的产生算法如下:

```
FOR every t_i(t_i∈T)
    N_t[i]=random[1,H_t];
ENDFOR
  FOR every t_i(t_i∈T)
    FOR every p_j(p_j∈P)
    E[i][j]= N_t[i] * random[1,H_p];
  ENDFOR
ENDFOR
```

3.2 测试参数的选择

从 SMM 算法思想可以看出,参数 k 的取值影响到整个算法性能。由性质 2 和性质 3 可知,当参数 k 取 1 或 n 时,算法都会退化为其他算法而影响算法性能。通过试验可以发现当参数 k 从 1 至 n 的变化过程中,算法调度跨度会减少,而任务的平均等待时间则会增加。图 2 是在 $n=100$,$m=16$ 的前提下,对 k 值为 2,3,4,5,10,20,…,50 的情况进行分析所得到的调度跨度和任务平均等待时间的比较图。

测试数据表明,当 k 过小时,由于缺少备选任务,SMM 算法无法得到充分的全局信息,在调度跨度方面性能较低,但任务平均等待时间较少。当 k 过大时,情况则刚好相反,算法在调度跨度方面明显减少,但任务平均等待时间也会相应上升。算法的这个性质决定了参数 k 需要结合不同的情况动态确定。当对调度跨度要求较高时,选取比较大的 k 值,当要追求任务等待时间较少时,选取较小的 k 值。为统一期间本文在接下去的测试中,参数 k 一律确定为 20。

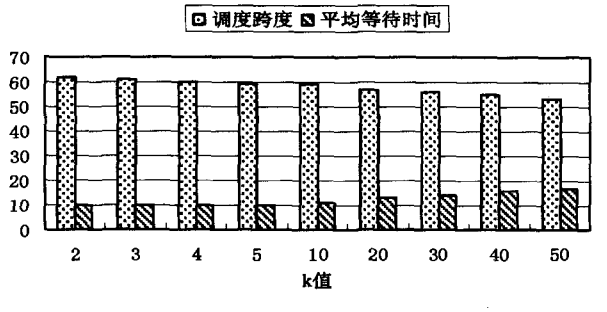

图 2 参数 k 与调度跨度和平均等待时间的关系

3.3 算法测试

本文在测试中,分别对 SMM 算法、Min-min 算法、Max-

(下转第 97 页)

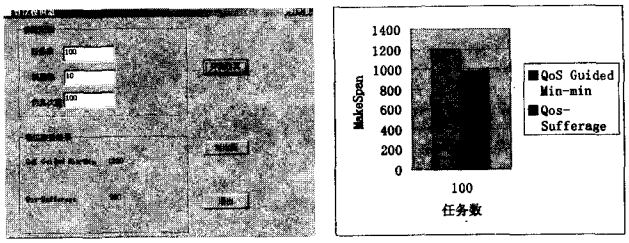

图 3 任务数为 100 时，QoS Guided Min-min 与 Qos-Sufferage 算法的对比

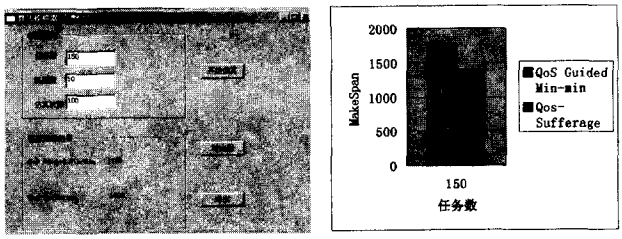

图 4 任务数为 150 时，QoS Guided Min-min 与 Qos-Sufferage 算法的对比

从图 2,3,4 可以看出 Qos-Sufferage 算法与 QoS Guided Min-min 算法相比，降低了平均完成时间，提高了资源调度的性能。

结束语 网格资源的调度是网格计算中的核心问题。本文通过对 GridSim 模拟器的深入探讨和研究，对 Minmin 算法和 QoS Guided Min-min 算法进行了研究，提出一些不足，并对 Minmin 算法和 QoS Guided Min-min 算法进行改进，在改进的算法中融入了 QoS 机制和 Sufferage 算法思想。实验结果表明：改进后的算法，在平均完成时间上有很大的改进，提高了资源的调度性能。

参 考 文 献

[1] Murshed M, Buyya R. Using the GridSim Toolkit for Enabling Grid Computing Education // International Conference on Communication Networks and Distributed Systems Modeling and Simulation (CNDS2002). San Antonio, Texas, USA, January 2002

[2] Braun T D, Siegel H J, Beck N, et al. A comparison study of staticmapping heuristics for a class of meta-tasks on heterogeneous computing systems // Proeeedings of the 8th IEEE Heterogeneous Computing Workshop(HCW,99). IEEE Computer Society Press, 1999: 15-29

[3] Bruna T D, Siegel H J, Beck N. A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems. Journal of Parallel and Distributed Computnig, 2001, 61

[4] He Xiaoshan, Sun Xian-he, von Laszewski G. A QoS Guided SchedulingAlgorithm For Grid Computing[J]. Journal of Computer science and Technology, 2003, 18(4)

(上接第 92 页)

min 算法以及 Sufferage 算法在性能上做了比较。由于这四类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，因此测试给出的结果均是多次计算结果的平均值。

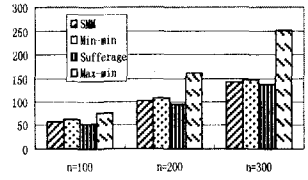

图 3 调度跨度比较 $m=16$

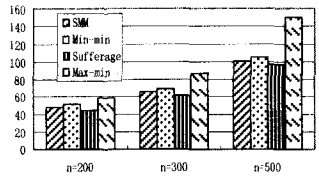

图 4 调度跨度比较 $m=32$

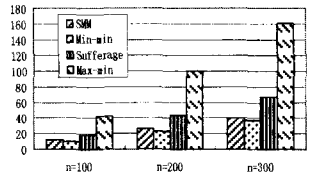

图 5 平均等待时间 $m=16$

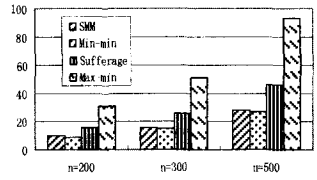

图 6 平均等待时间 $m=32$

测试一共进行了 2 组，图 3 至图 6 分别给出了各自有关调度跨度和平均等待时间的比较结果。第一组测试条件为：$m=16, n=100,200,300$；第二组测试条件为：$m=32, n=200,300,500$。

3.4 测试结果分析

从图 3，图 4 中可以发现，在不同的处理机规模和任务数量下 SMM 算法、Min-min 算法和 Sufferage 算法的调度跨度远优于 Max-min 算法，且 SMM 算法调度跨度低于 Min-min 算法，接近于 Sufferage 算法。同样图 5，图 6 说明了在平均等待时间上，SMM 算法远低于 Sufferage 算和 Max-min 算法。总的来说，SMM 算法做到了调度跨度低与平均等待时间小的统一，因此综合性能略优于其他三个算法。

结束语 本文提出了适用于异构环境的独立任务调度算法：基于任务调度损失的最小最早完成时间算法（SMM 算法）。该算法将任务调度损失引入 Min-min 算法中，在综合权衡任务最早完成时间与任务调度损失的基础上进行调度，克服了 Min-min 算法片面追求局部最优的局限性，使算法更适合于异构环境。经测试表明，该算法在任务调度跨度和平均等待时间这两项指标中均有较优性能。

参 考 文 献

[1] Armstrong R, Hensgen D, Kidd T. The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions[A] // Proceedings of 7th IEEE Heterogeneous Computing Workshop(HCW '98)[C]. 1998: 79-87

[2] Maheswaran M, Ali S, Siegel H J, et al. Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems // Proceedings of the Eighth IEEE Heterogeneous Computing Workshop: 30-44

[3] Freund R F, Gherrity M, Ambrosius S, et al. Scheduling resourcesin multi-user, heterogeneous, computing environments with SmartNet // Proceedings of Heterogeneous Computing Workshop. 1998: 184-199

[4] Braun T D, Siegel H J, Beck N. A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing System. Journal of Parallel and Distributed Computing, 2001: 810-837

[5] Ibarra O, Kim C. Heuristic algorithms for scheduling independent tasks on nonidentical processors. Journal of the ACM, 1977, 77(2): 280-289