异构环境下基于松弛标记法的任务调度

杜晓丽 [1,2] 王俊丽 [1,2] 蒋昌俊 [1,2]

摘 要 提出了一种基于松弛标记法的任务调度算法 (Relaxation labeling based task scheduling, RLBTS), 将任务映射到异构资源 (处理器计算能力和链路的通信能力不同) 上. 松弛标记法善于处理大量的约束条件, 其核心思想是结点的标签分配通常受该结点的邻居结点某些属性的影响. 依据邻居约束关系, 可以逐渐排除不相关因素, 迅速缩小搜索空间. 该算法统筹兼顾了任务执行的计算需求和通信需求问题, 实验结果表明对于通信和计算需求都很高的任务和通信密集型任务, RLBTS 不失为一种有效的调度算法.

关键词 任务调度, 松弛标记法, 异构环境
中图分类号 TP393.01

Relaxation Labeling Based Task Scheduling in Heterogeneous Environments

DU Xiao-Li[1,2] WANG Jun-Li[1,2] JIANG Chang-Jun[1,2]

Abstract On the base of relaxation labeling, an algorithm of task scheduling in heterogeneous computing environments is presented. This new method maps data-processing tasks onto heterogeneous resources (*i.e.*, processors and links of various capacities), and takes both computing requirement and communication needs into account. Relaxation labeling can handle a broad range of constraints and its key idea is that the label of a node is typically influenced by the features of the node′s neighborhood in the graph. According to neighborhood restrictions, it can gradually get rid of uncorrelated factors and rapidly shrink the searching space. Experimental results show that it performs very well for applications that have high computing and communication requirements or communication-intensive requirements.

Key words Task scheduling, relaxation labeling, heterogeneous environments

1 引言

CPU 计算能力和网络带宽都以惊人的速度发展, 然而, 最近的发展趋势表明网络带宽的发展速度远远超过 CPU 的发展速度. 这使得以前只能运行在超级计算机或机群上的通信密集型并行任务, 有可能通过网络利用分布在实验室甚至家庭中的台式机构成分布式计算环境来执行. 从而, 资源管理问题成为核心问题. 资源管理中最基本的问题之一就是给定包含一组任务的应用程序, 如何选择合适的资源分配给其任务. 然而, 实际应用中情况更加复杂, 资源通常来自不同的厂家, 处理器性能有很大差异. 能处理相同任务的处理器所花费的计算代价可能不同, 各种处理器之间的通信速度也有很大差异. 任务执行过程中需要与相邻任务进行通信, 而且某些任务只能由特定的处理器来执行. 例如由图 1 和图 2 可以看到, 图 1 中的任务 T_0 只能由图 2 中的处理器 M_0, M_2, M_4 执行; 任务 T_1 只能由 M_3 来执行; 任务 T_2 可由 M_1, M_2, M_6 来执行; 任务 T_3 可由 M_0, M_1, M_2, M_5 执行; 任务 T_4 可由 M_3, M_5 执行. 以往大多数调度算法, 往往假设任务可以被任何处理器执行, 而且要么忽略资源的异构性[1], 要么把任务看作是相互独立的, 忽略任务间的通信关系, 如算法 MCT, MET[2], SA, Max-min[3], PMM[4], BM[5] 等.

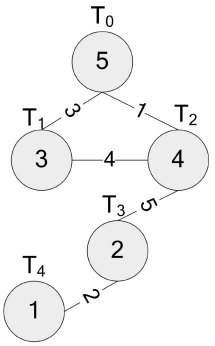


图 1 任务图 G
Fig. 1 Task graph G

收稿日期 2005-8-26 收修改稿日期 2006-6-30
Received August 26, 2005; in revised form June 30, 2006
国家自然科学基金 (60534060, 90412013), 国家重点基础研究发展计划 973 (2003CB316902), 上海市科委 2006 年度 "登山行动计划"(06JC14065), 上海市优秀学科带头人计划 (04XD14016) 资助
Supported by National Natural Science Foundation of P. R. China (60534060, 90412013), the National Base Research Program of P. R. China–973 Program (2003CB316902), the 2006 Mountaineering Program of Science and Technology Commission of Shanghai Municipality (06JC14065), the Program of Shanghai Subject Chief Scientist (04XD14016)
1. 同济大学电子与信息工程学院 上海 201804 2. 国家高性能计算机工程技术中心同济分中心 上海 201804
1. Electronics and Information Engineering School, Tongji University, Shanghai 201804 2. Tongji Branch, National Engineering & Technology Center of High Performance Computer, Shanghai 201804
DOI: 10.1360/aas-007-0615

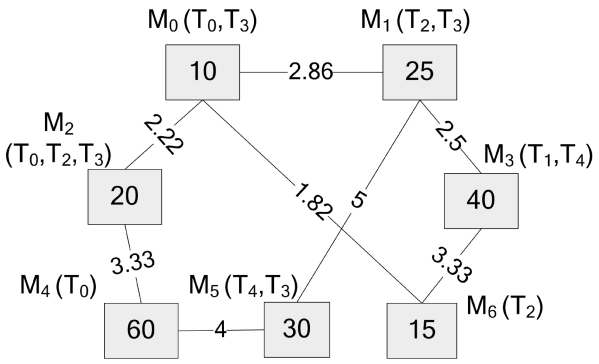


图 2 资源图 P

Fig. 2 Resource graph P

松弛标记法[6] 是解决将标签分配到图中结点的一种非常有效的技术, 善于处理大量的约束条件, 核心思想是结点的标签分配通常受该结点的邻居结点某些属性的影响. 松弛算法本质上是并行的, 它以迭代形式进行, 整个过程与人的猜测推理过程相类似, 利用各种关系逐步缩小搜索范围, 最终求出正确结果. 针对任务执行过程中既有计算需求, 又有通信需求, 而且某些任务只能由特定处理器来执行这类问题, 本文基于松弛标记法提出了一种异构环境下任务调度算法 (Relaxation labeling based task scheduling, RLBTS), 任务被看作是目标对象, 资源被看作是待分配标记. 目标是根据对象 (任务) 间所有可能的关系 (如所需计算能力, 任务间的通信等) 及关系间的相容性, 计算对象接受一个标记 (处理器) 的置信度, 以找到最合适的资源分配方案及最小的完成时间. 本算法以任务图和资源图作为输入, 输出为任务到处理器的匹配情况.

本文的其他部分组织如下：第 2 节介绍了一些基本的概念和符号定义, 第 3 节介绍了算法的构造过程, 第 4 节给出实验结果与性能分析, 第 5 节总结了本文的工作.

2 问题的定义

2.1 任务图

任务图是无向图 $G = (V_G, E_G)$, 其结点表示任务, 结点权值 $W_g(T_i)$ 表示任务 T_i 执行一个单位任务步所需要的计算量, 边表示相邻任务间有通信关系, 但并不是严格意义上的时间偏序关系, 其权值 $W_g(T_i, T_j)$ 表示一个单位任务步内 T_i 与 T_j 之间的通信量. 其中 V_G 是任务集合, E_G 是边集合. 注意这里的任务图并不是一般意义上的 DAG 图, 在 DAG 图中, 有向边 $s \to t$ 表示任务 t 当且仅当任务 s 完成之后才能开始. 然而, 本文的任务图是一个数据处理管道, 所有任务不断地接收数据、处理数据, 然后发送数据, 并没有严格意义上的时间偏序关系.

2.2 资源图

资源图是无向图 $P = (V_p, E_p)$, 其结点表示处理器, 权值 $W_p(M_i)$ 表示单位时间内处理器 M_i 可执行的计算量. 边表示连接处理器的网络, 其权值 $W_p(M_i, M_j)$ 表示链路 (M_i, M_j) 单位时间内传输的消息量. V_p 是处理器集合, E_p 是边集合.

2.3 处理器的性能估计

处理器的综合性能是其计算、存储、通信以及其他能力的函数. 为了简单起见, 仅考虑其计算和通信能力, 则处理器 M_i 的性能见式 (1).

$$Perf(M_i) = \alpha_1 * W_p(M_i) + \alpha_2 * E(Comm(M_i)) \tag{1}$$

其中, $E(Comm(M_i))$ 为处理器 M_i 通信能力的期望, 其计算式见式 (2). 在式 (2) 中, n 为结点 M_i 的度. α_1, α_2 为处理器计算和通信能力的权重, 满足 $\alpha_1 + \alpha_2 = 1$. 若 α_1 远远大于 α_2 表示任务为计算密集型, 反之则为通信密集型. α_1, α_2 需根据经验确定, 因处理器的性能估计仅影响处理器分配的初始置信度, 受邻居任务处理器分配的影响, 算法的每次迭代都会修改任务处理器分配的置信度, 所以 α_1, α_2 值估计有所偏差, 但影响不是很大.

$$E(Comm(M_i)) = \frac{1}{n} Comm(M_i) = \frac{1}{n} \sum_{(M_i, M_j) \in E_p} W_p(M_i, M_j) \tag{2}$$

2.4 处理器分配的初始置信度

令 C_i 为能够处理任务 $T_i \in V_G$ 的处理器集合, 如果 $M_j \in V_P$ 且 $M_j \in C_i$, 则 M_i 处理 T_i 的初始置信度为

$$P^0(T_i = M_j) = \frac{Perf(M_j)}{\sum\limits_{M_k \in C_i} Perf(M_k)} \tag{3}$$

如果 $M_j \notin C_i$, 则 $P^0(T_i = M_j) = 0$. (其中 $T_i = M_j$, 表示将 M_j 分配给 T_i, 即由 M_j 来执行 T_i). 图 1 和图 2 的处理器分配的初始置信度见表 1.

表 1　图 1 和图 2 的初始置信度

Table 1　The initial confidence level of Fig. 1 and Fig. 2

	M_0	M_1	M_2	M_3	M_4	M_5	M_6
T_0	0.12	0.00	0.22	0.00	0.66	0.00	0.00
T_1	0.00	0.00	0.00	1.00	0.00	0.00	0.00
T_2	0.00	0.36	0.36	0.00	0.00	0.00	0.28
T_3	0.13	0.25	0.25	0.00	0.00	0.37	0.00
T_4	0.00	0.00	0.00	0.57	0.00	0.43	0.00

2.5　处理器所花费的计算时间估计

$$Tcp(M_i)=\frac{\sum\limits_{T_j=M_i} WG(T_j)}{W_p(M_i)} \tag{4}$$

若存在 $T_m=M_i$, $T_n=M_i(m\neq n)$, 表示 M_i 可执行 T_m, T_n 两种类型的任务.

2.6　处理器所花费的通信时间估计

$$T_{cm}(M_i)=\sum_{T_u=M_i,T_v=M_j,(T_u,T_v)\in E_G} W_G(T_u,T_v)\cdot ShortPath(M_i,M_j) \tag{5}$$

2.7　一组任务执行单位任务步的时间估计

$$T=\max_{M_i\in V_m}\{\alpha_1\cdot Tcp(M_i)+\alpha_2\cdot Tcm(M_i)\} \tag{6}$$

因此, 我们的目标是通过合适资源分配获得最小 T.

3　算法

3.1　松弛标记法和启发性知识

对任务图中所有任务尝试分配处理器的操作称为一次匹配, 记作 m. 如图 1 和图 2, 有分配 $m=\{(T_0,M_4),(T_1,M_3),(T_2,M_1),(T_3,M_2),(T_4,M_5)\}$, 表示 T_0 由 M_4 来执行, T_1 由 M_3 来执行, 依次类推. 由图 1 和图 2 可以看出 T_2 要和 T_3 通信可经过 $M_1\rightarrow M_0\rightarrow M_2$, $M_1\rightarrow M_5\rightarrow M_4\rightarrow M_2$ 或 $M_1\rightarrow M_3\rightarrow M_6\rightarrow M_0\rightarrow M_2$ 三条路径. 取其所有路径中通信时间最小的路径作为两个任务的通信路径, 记作两个结点之间的最短路径, 用式 (7) 来计算. 那么, T_2 和 T_3 的通信路径为 $M_1\rightarrow M_0\rightarrow M_2$.

$$ShortPath(M_i,M_j)=\min\{\sum_{Path^{w=1}(M_i,M_j)}^{n}\sum_{(M_x,M_y)\in Path^w}\frac{1}{W_p(M_x,M_y)}\} \tag{7}$$

在一次分配中, M_j 为此次分配中分配给 T_i 的标记, 令 T_i 为其邻居任务标记分配的一种情况. 对象 T_i 分配标记 M_j 的置信度由与它直接作用的邻居任务的标记分配决定. $R(T_i=M_j,T_i)$ 表示对象 T_i 和其邻居对象之间的相容度, 即是否分配 M_j 给任务 T_i 与该分配对其他任务的执行情况影响有关, 反映为 M_j 分配给 T_i 是否有利于整体上获得较小的执行时间 T. 对 T_i 分配 M_j 的影响主要从计算和通信两个方面来考虑. 函数 $\sigma(x)=1/(1+\exp(x))$ 广泛应用与结合各种来源的证据[7], 因此 T_i 和 T_i 之间的相容度的量化, 采用式 (8) 计算.

$$R(T_i=M_j,\Delta T_i)=1/(1+\exp(\max_{M_u\in\Delta T_i}(\alpha_1\cdot Tcp(M_u)+\alpha_2\cdot Tcm(M_u)))) \tag{8}$$

在松弛过程中, 通过迭代来更新估计的标记分配置信度, 更新算子的计算见式 (9). 式 (9) 中 C_i 为可处理 T_i 的处理器集合, $P^{k+1}(T_i=M_l)$ 表示第 $k+1$ 步, T_i 分配 M_l 的置信度.

$$P^{k+1}(T_i=M_j)=\frac{P^k(T_i=M_j)Q^k(T_i=M_j)}{\sum\limits_{M_k\in C_i}P^k(T_i=M_k)Q^k(T_i=M_k)} \tag{9}$$

式 (10) 为第 k 次迭代中 T_i 的邻居任务对 T_i 分配 M_j 的支持度, 即为得到 $P^{k+1}(T_i=M_j)$, 在第 $k+1$ 次迭代中作用于 $P^k(T_i=M_j)$ 的校正量.

$$Q^k(T_i=M_j)=\sum_{\Delta T_i}(R(T_i=M_j,\Delta T_i)\prod_{T_j\in\Delta T_i}P^k(T_j=M_k)) \tag{10}$$

3.2　RLBTS 算法

1) 根据式 (3) 计算各个任务分配各个处理器单元的初始置信度;

2) Repeat

3) 由式 (7)∼(9) 计算受邻居关系约束的任务的处理器单元分配置信度;

4) 根据第 3 步计算得出的处理器单元分配置信度, 更新原来的分配置信度;

5) Until (任意 T_i 分配处理器单元 M_j 的置信度连续三次上升或达到指定的迭代次数).

4 实验与性能分析

4.1 实验环境

我们设计了一个简单的随机任务图与资源图生成器, 根据用户定制的参数, 产生相应的任务图和资源图. 任务图和资源图的参数及含义如表 2 所示.

4.2 性能分析

由图 1 和图 2 可看出各个任务对处理器计算能力和通信能力都有很高要求, 因此令 $\alpha_1 = \alpha_2 = 0.5$, 迭代次数为 5, 根据我们的算法得出结果如表 3 所示, 其中 "↑" 表示为该分配的置信度连续三次上升. 所以最终分配结果为: $\{(T_0, M_0), (T_1, M_3), (T_2, M_1), (T_3, M_1), (T_4, M_5)\}$, 在迭代过程中 T_1 分配 M_3 的置信度始终为 1, 因为 T_1 只能由处理器 M_3 来执行. 与根据穷举算法得出的最优结果相同. 但是 RLBTS 算法的运行时间仅为 20, 而穷举算法的运行时间则为 50. 这是因为穷举算法要得到最优解, 需搜索整个解空间 $\prod_{T_i \in E_G} Num(C_i)$, 而 RLBTS 算法则能根据任务分配处理器单元置信度的上升趋势迅速收敛, 得出比较理想的匹配结果.

表 3 图 1 的结果

Table 3 The results of Fig. 1

	M_0	M_1	M_2	M_3	M_4	M_5	M_6
T_0	0.25 ↑	0.00	0.02	0.00	0.73	0.00	0.00
T_1	0.00	0.00	0.000	1.00	0.00	0.00	0.00
T_2	0.00	0.50 ↑	0.00	0.00	0.00	0.00	0.50
T_3	0.12	0.55 ↑	0.01	0.00	0.00	0.32	0.00
T_4	0.00	0.00	0.00	0.34	0.00	0.66 ↑	0.00

对图 1 中的每个任务 T_i 随机生成其处理器集合 C_i, 然后根据 RLBTS 算法进行 50 次实验, 将实验结果和穷举法得出的最优分配情况进行比较, 如图 3 所示, 横坐标表示实验次数, 纵坐标表示估计完成时间. 其中, 本算法有 28 次 (54%) 与穷举算法得出的最优结果相同, 有 11 次 (22%) 与穷举算法得出的最优结果的估计完成时间相差不超过 0.3 时间单位, 7 次 (18%) 不超过 0.8 时间单位, 仅有 4 次 RLBTS 算法产生的结果与最优结果估计完成时间相差超过 1 个时间单位. 然而, 在这 50 次实验中, 穷举算法的运行时间平均比 RLBTS 算法高约 6.30 倍, 如图 4 所示.

表 2 任务图和资源图参数表

Table 2 Parameters of task graph and resource graph

类型	参数	含义
Task graph	V_t	节点数目: 一定程度上反映了图的规模
	mxNd_weiTA	节点最大权重
	mnNd_weiTA	节点最小权重: 节点权重在最大、最小权重之间随机生成
	mxEd_weiTA	边的最大权重
	mnEd_weiTA	边的最小权重: 边的权重在最大、最小权重之间随机生成
	deepth	图的深度
Resource graph	type_RE	资源图类型: complete-connected 和 arbitrary-connected, 本文为后者
	V_p	节点的数目
	mxNd_weiRE	节点的最大权重
	mnNd_weiRE	节点的最小权重: 节点的权值为这两个值之间得一个随机数
	mxEd_weiRE	边的最大权重
	mnEd_weiRE	边的最小权重: 节点的权值为这两个值之间得一个随机数
	λ	节点异构率: $\lambda \in [0, 1]$, $\lambda = 1$ 时, 处理单元完全异构; $\lambda = 0$ 时, 为处理单元同构的系统;
	δ	边的异构率: $\delta \in [0, 1]$, $\delta = 1$ 时, 链路完全异构; $\delta = 0$ 时, 为链路同构的系统
	mxOut_Deg	节点的最大出度: 节点的出度在 [0, mxOut_Deg] 之间随机生成

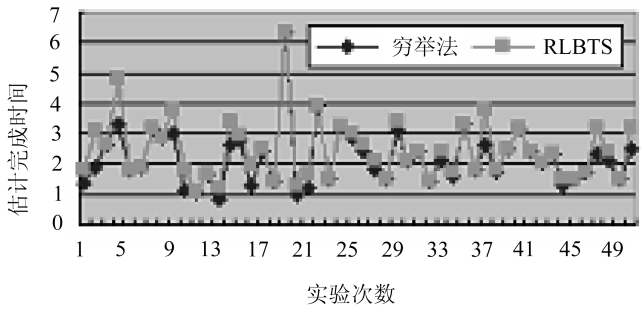


图 3 RLBTS 结果与穷举算法最优结果的估计完成时间比较

Fig. 3 The estimate complete time comparison between RLBTS and the optimal results from exhausted algorithm

表 4 实验数据表

Table 4 Test data

Task Graph		Resource Graph	
V_t	$\{5, 10, 15, 30, 50\}$[1]	type_RE	arbitrary-connected
mxNd_weiTA	100	V_p	$\{7, 9, 15, 37, 20\}$
mnNd_weiTA	10	mxNd_weiRE	200
mxEd_weiTA	10	mnNd_weiRE	50
mnEd_weiTA	1	mxEd_weiRE	50
mxOut_Deg	[3,10][2]	mnEd_weiRE	5
deepth	[3,10]	λ	[0,1]
		δ	[0,1]
		mxOut_Deg	[3,10]

[1] 表示五组实验中的依次取值; [2] 表示随机选择 [a,b] 区间内一个值.

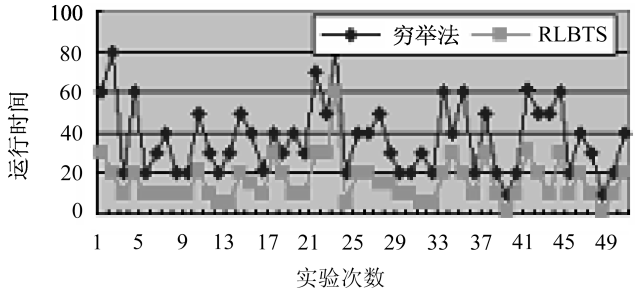


图 4 RLBTS 与穷举算法的运行时间比较

Fig. 4 The runtime comparison between RLBTS and exhausted algorithm

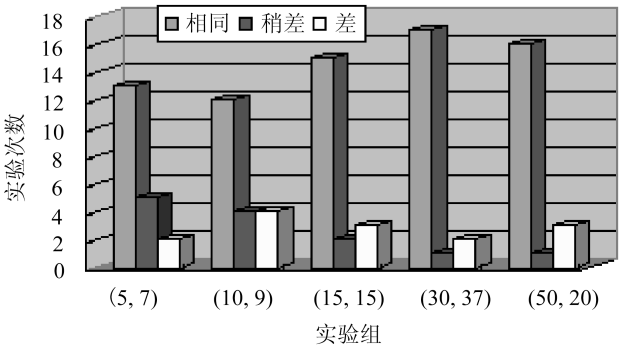


图 5 RLBTS 产生的结果分析

Fig. 5 RLBTS′ results analysis

根据表 4 的数据又进行了五组实验, 每组进行 20 次实验, 每次试验中随机生成任务图、资源图以及任务图的处理器集合. 将每组实验结果分为 “相同”、“稍差” 和 “差” 三种, 其中 “相同” 为 RLBTS 算法产生与穷举法得出的最优结果相同的结果, “稍差” 为与最优结果的估计完成时间不超过 0.3 个时间单位的结果, 其他情况为 “差”. 图 5 中横坐标为五组实验中随机生成任务图和资源图的结点数目. 可以看出在这五组实验中, RLBTS 算法产生与最优结果相同的情况均超过 60%, 产生相对 “稍差” 和 “差” 结果的情况平均约为 13% 和 14%. 对于 α_1 和 α_2 的估计值, 通过实验我们得出如表 5 所示的结论.

表 5 α_1 和 α_2 的取值

Table 5 Results of α_1 and α_2

CCR	α_1	α_2
$0.90 \le CCR < 1.10$	0.50	0.50
$0.60 \le CCR < 0.90$	0.60	0.40
$0.10 \le CCR < 0.60$	0.70	0.30
$0.09 \le CCR < 0.10$	0.90	0.10
$CCR < 0.09$	0.99	0.01
$1.10 \le CCR < 1.50$	0.40	0.60
$1.50 \le CCR < 2.00$	0.30	0.70
$2.00 \le CCR < 10.00$	0.10	0.90
$10.00 \le CCR$	0.01	0.99

在这五组实验中, 第四组实验中随机产生的任务图的通信计算比 (Communication computation ratio, CCR) 都小于 0.1, 即此组实验为计算相对密集型任务; 第一、二、三组实验中随机产生的任务图的 $CCR \in [0.5, 1.5]$, 即通信、计算需求都很高的任务; 第五组为 $CCR > 8.5$. RLBTS 算法的运行时间与穷举法比较如图 6~10 所示. 由图 6~8 可以看出, 在第一、二、三组实验中穷举算法的运行时间平均比 RLBTS 算法高出 6~7 倍, 而对于计算相对密集型任务 (图 9 情况), RLBTS 算法的优势并不明显, 原因是计算密集型任务的邻居约束关系较弱, 松弛标记法优势难以充分发挥, 尤其不适合不考虑通信的独立任务调度. 对于通信密集型任务情况, 由图 10 可以看出穷举算法的运行时间比 RLBTS 算法高出 10 倍以上.

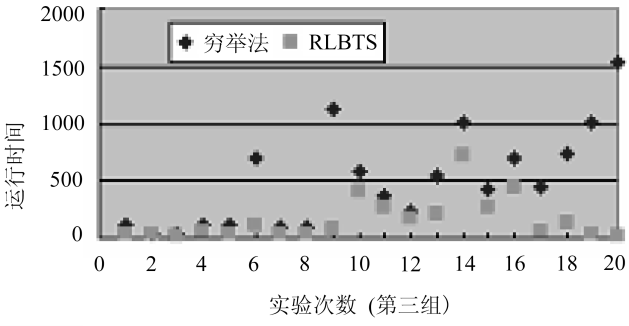


图 8 第三组实验中穷举算法和 RLBTS 的运行时间比较

Fig. 8 The runtime comparisons between exhausted algorithm and RLBTS in third group of tests

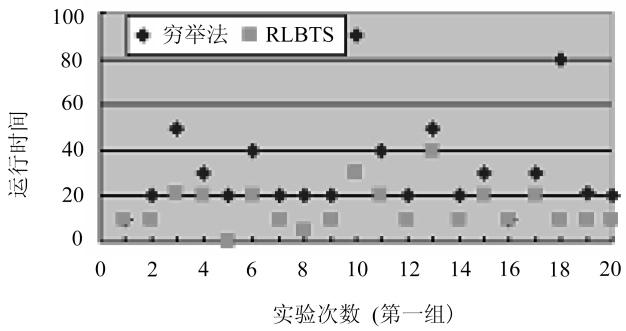


图 6 第一组实验中穷举算法和 RLBTS 的运行时间比较

Fig. 6 The runtime comparisons between exhausted algorithm and RLBTS in first group of tests

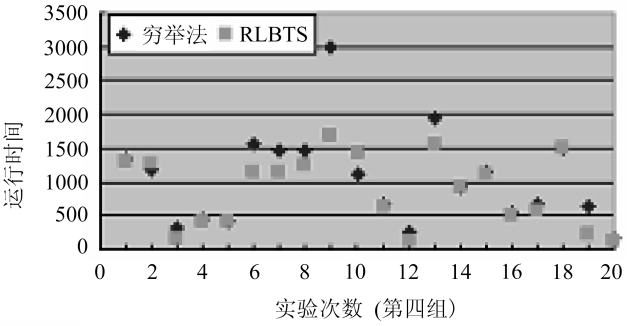


图 9 第四组实验中穷举算法和 RLBTS 的运行时间比较

Fig. 9 The runtime comparisons between exhausted algorithm and RLBTS in fourth group of tests

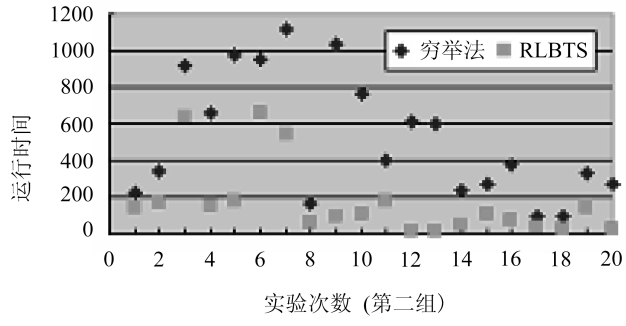


图 7 第二组实验中穷举算法和 RLBTS 的运行时间比较

Fig. 7 The runtime comparisons between exhausted algorithm and RLBTS in second group of tests

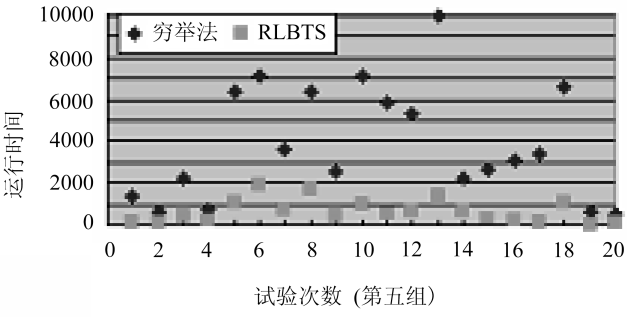


图 10 第五组实验中穷举算法和 RLBTS 的运行时间比较

Fig. 10 The runtime comparisons between exhausted algorithm and RLBTS in fifth group of tests

5 结论

本文利用松弛标记法善于处理大量的约束条件, 结点的标签分配通常受该结点的邻居结点某些属性的影响等特点, 提出了一种基于松弛标记法的任务调度算法, 该算法统筹兼顾了任务执行的并行度和通信容量问题. 实验结果表明对于通信和计算需求都很高的任务和通信密集型任务, RLBTS 算法不失为一种有效的调度算法.

References

1 Ritchie G, Levine J. A fast effective local search for scheduling independent jobs in heterogeneous computing environments. In: Proceedings of the 22nd Workshop of the UK Planning and Scheduling Special Interest Group. Glasgow, UK, 2003. 178∼183

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, USA, IEEE Computer Society, 1998. 79∼87

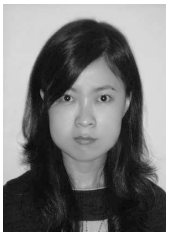
3 Freund R F, Gherrity M, Ambrosius S. Scheduling resources in multi-user heterogeneous computing environments with smartnet. In: Proceedings of the Seventh IEEE Heterogeneous Computing Workshop. Washington DC, USA, IEEE Computer Society, 1998. 184∼199

4 Zhi Qing, Jiang Chang-Jun. A scheduling algorithm suitable for heterogeneous computing environment. *Acta Automatica Sinica*, 2005, **31**(6): 865∼872
(支青，蒋昌俊. 一种适于异构环境的任务调度算法. 自动化学报, 2005, **31**(6): 865∼872)

5 Zhi Qing, Jiang Chang-Jun. A both matched dynamic scheduling algorithm. *Information and Control*, 2005, **34**(5): 532∼538
(支青，蒋昌俊. 一种双匹配动态调度算法. 信息与控制, 2005, **34**(5): 532∼538)

6 Hummel R, Zucker S. On the foundations of relaxation labeling processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1983, **5**(3): 267∼287

7 Agresti A. *Categorical Data Analysis*. New York: Wiley, 1990



杜晓丽 同济大学计算机软件与理论专业博士研究生. 主要研究方向为网络计算及并行算法. 本文通信作者.
E-mail: du_xiaoli@163.com
(**DU Xiao-Li** Ph. D. candidate in computer software and theory at Tongji University. Her research interest covers network computing and parallel algorithm. Corresponding author of this paper.)

王俊丽 同济大学计算机应用技术专业博士研究生. 主要研究方向为语义网格.
E-mail: wangjunli_1029@hotmail.com
(**WANG Jun-Li** Ph. D. candidate in computer application technology at Tongji University. Her research interest covers semantic grid.)

蒋昌俊 同济大学电子与信息工程学院教授. 主要研究方向为网格计算、语义网格和 Petri 网.
E-mail: cjjiang@online.sh.cn
(**JIANG Chang-Jun** Professor in School of Electronic and Information Engineering at Tongji University. His research interest covers network computing, semantic grid, and Petri nets.)