网格多处理机的一种改进的子网分配算法*

张 艳, 孙世新, 彭文钦

(电子科技大学 计算机科学与工程学院,四川 成都 610054)

E-mail: zhangyan_hong@263.net

http://www.uestc.edu.cn

摘要: 子网分配问题是指识别并分配一个空闲的、满足指定大小要求的节点机. 首先,提出了网格结构中一种新的具有 $O(N_a^2 \cdot \log_2 N_c)$ 时间复杂度的空闲子网搜索算法,它优于现有的 $O(N_c^3)$ 时间复杂度的搜索算法. 然后,用该算法对基于保留因子的最佳匹配类子网分配算法——RF(reservation factor)算法进行了改进,得到了优于它的结果.

关键词: 子网分配;网格;空闲子网

中图法分类号: TP393 **文献标识码**: A

大规模并行处理机 MPP(massively parallel processors)被认为是构造高性能并行计算环境最有潜力的方法,而网格结构由于其简单、规则和易伸缩的特性成为 MPP 中最具吸引力的体系结构. 目前已有多种网格结构的商用并行机存在,如 Inter Touchston Delta, IntelParagon P/S, Fujitsu AP-100, PASM(partitionable SIMD/MIMD),曙光系列等.

网格多处理机是一种多用户、多任务环境,如 Intel Touchston Delta 和 Paragon XP/S 支持多任务环境,PASM 提供了一种操作一个或多个各种不同尺寸的独立子机的动态机制. 在多任务环境中,大量的任务可以同时分配给独立子网执行,由于各任务需要大小不同的子网,而网格结构应尽量容纳更多的任务,因而为获得它的高性能,我们需要有效的子网分配算法.

所谓的子网分配问题即是指对请求 $w \times h$ 个节点机的任务,分配满足要求的空闲子网 $a \times b$ ($a \geqslant w, b \geqslant h$ 或 $a \geqslant h, b \geqslant w$)给该任务. 而有效的子网分配算法则要求使系统的利用率最大,即使得系统由此产生的内片段和外片段最小,其中内片段是指给一个任务分配了一个超过它所需要的节点机数的大的空闲子网,从而造成资源的浪费;外片段是指网格结构中有足够的节点机可以被分配,但不能找到满足需要大小的一个空闲子网. 这样的一些片段严重降低了系统的利用率.

网格多处理机的子网分配算法有多种,大多属于首次匹配类或最佳匹配类,其中首次匹配类的分配算法(如 Adaptive Scan 算法[1]、QA 算法[2]等)是将找到的第 1 个足够大的空闲子网分配给所需任务;最佳匹配类的分配算法(如 Busy-List 算法[3]、Free-List 算法[4]等)首先找出可用于分配的所有候选子网,然后依据某种准则找到一个最佳候选子网,将它分配给请求节点机的任务. 首次匹配类的分配算法由于无须找出所有满足要求的空闲子网,且不存在比较,因而它的时间复杂度一般比最佳匹配类的分配算法要低,但它不能有效地降低外片段数,比后者的系统利用率低. 所以,在大

* 收稿日期: 2000-01-21; 修改日期: 2000-04-13

基金项目: 国家"九五"国防预研基金资助项目(16.1.4.1)

作者简介: 张艳(1973-),女,河北邯郸人,博士,主要研究领域为并行算法及其应用;孙世新(1941-),男,湖北孝感人,教授,博士生导师,主要研究领域为计算机科学理论,并行算法;彭文钦(1975-),男,四川自贡人,硕士,主要研究领域为并行算法及其应用.

多数情况下,最佳匹配类的子网分配算法是一种更好的选择.

现有的最佳匹配类子网分配算法有以下几种:

Buddy 算法[5]:该算法只适用于长度为 2 的幂的方形网格结构.对请求任意节点机数的任务,它将分配可容纳该请求的最小 2 的幂的方形子网.因此,该算法存在着严重的内片段数问题.

Best-Fit 算法(简称 BF 算法)[6]:BF 算法在最小的空闲区域分配子网.它选择具有最大边界值的子网作为最佳分配子网.

Busy-List 算法(简称 BL 算法)[3]:BL 算法是第 1 个具有完全识别能力的子网分配策略.它选择具有最大边界值的子网作为最佳子网,优于前两种算法.

Free-List 算法(简称 FL 算法)[4]:FL 算法基于一个空闲列表.该空闲列表按现有空闲子网短边(相同短边,则按长边)的增序排列.对每个分配请求,该算法将空闲列表中第 1 个可容纳请求子网的空闲子网分配给它,若空闲子网大于请求子网,则将 4 个角生成的 8 个候选子网中具有最大边界值的子网作为最佳候选子网.它优于 BL 算法.

文献[7]所提出的算法(简称 RF 算法):基于保留因子的最佳匹配类分配算法——RF 算法每次在进行分配时,计算各候选子网相对于空闲子网的保留因子,将具有最大保留因子的候选子网作为最佳分配子网.

上述最佳匹配类子网分配算法依据的匹配准则不同.文献[7]分析并指出,该文中保留最大空闲子网的匹配准则是最佳选择,它有效地避免了文献[3,4]的匹配准则可能导致的在多个空闲子网的重叠区域分配子网这种情况的发生,从而将大的空闲子网用于后续任务的分配,使得系统的外片段数最小.但文献[7]中所提算法搜索所有空闲子网的算法的时间复杂度高,为 $O(N_a^3)$.为此,本文提出了一种新的具有 $O(N_a^2 \cdot \log_2 N_a)$ 时间复杂度的搜索空闲子网的算法,并据此改进了 RF 最佳匹配类子网分配算法.

1 一种新的搜索空闲子网的算法

文献[7]所提算法每次在其子网释放过程中重新计算网格结构中的空闲子网,它是一个不断地将每个分配子网与已得到的空闲子网比较,从中删除重叠部分的过程.本节提出了优于它的一种新的空闲子网搜索的算法,为便于描述新算法,首先给出一些基本概念.

1.1 基本概念

假设用(〈·,·〉,〈·,·〉)表示一个矩形子网,其中第 1 个〈·,·〉是该子网的左下角坐标,第 2 个〈·,·〉是该子网的右上角坐标.

设网格多处理机为 $L_x \times L_y$,记为 $M(\langle 0,0\rangle,\langle L_x-1,L_y-1\rangle)$,并记 MR 为$(\langle L_x,0\rangle,\langle L_x,L_y-1\rangle)$,$MB$ 为$(\langle 0,-1\rangle,\langle L_x-1,-1\rangle)$.

定义 1. 一个空闲子网是指该子网中所有的节点机均处于空闲状态,一个分配子网是指该子网中所有的节点机均分配给了一个任务.

定义 2. 一个空闲子网被称为控制子网,若它不能被其他空闲子网完全覆盖.控制子网之间可以互相重叠.

有了控制子网的定义,我们要说明一点是,文献[7]中的搜索方法得到的空闲子网是控制子网,它不包含在其他空闲子网之内.

定义 3. 一个空闲子网列(记为 FSL)是控制子网按其面积(也即节点机数)大小的非增序排列

的一个有序列.若两个或更多的控制子网是大小相等的,则按两边差的增序排列.

定义 4. 一个分配子网列(记为 BSL)是现有的分配子网 $S(\langle x_1,y_1\rangle,\langle x_2,y_2\rangle)$按其左下角的横坐标 x_1 的非减序排列的一个有序列.

定义 5. 一个未知空闲子网的起始位置列(记为 LSL)是由 n 个竖直线段 $l_i=(x^i:y_b^i,y_t^i)$按 y_b^i 的减序排列的一个有序列,且满足 $y_t^0=L_y-1$,$y_t^i=y_b^{i-1}-1$,$y_b^{n-1}=0$,$x^i\neq x^{i-1}$,其中的直线段 l_i 表示 $(\langle x^i-1,y_b^i\rangle,\langle x^i-1,y_t^i\rangle)$上的节点机已分配,而$(\langle x^i,y_b^i\rangle,\langle L_x-1,y_t^i\rangle)$子网内的节点机的分配状况是未知的,只能通过计算得到.

将 BSL 与 LSL 进行分析与比较就可以得到一种新的搜索最大区域的空闲子网,即控制子网的算法.

1.2 一种新的搜索空闲子网的算法

假设分别用 free_list,busy_list,line_list 表示 FSL,BSL,LSL.假设 busy list 中的前 $k-1$ 个分配子网已与 line_list 进行了比较,line_list 中的直线段也因而不断地变化.现在分析当前 line_list 中的一条直线段 $l_i=(x^i:y_b^i,y_t^i)$与 busy list 中的第 k 个分配子网 $S(\langle x_1,y_1\rangle,\langle x_2,y_2\rangle)$之间的位置分布情况以及它们对控制子网和 l_i 变化的影响.

显然,由于分配子网互不重叠,所以 S 与 l_i 之间只存在以下几种位置分布情况:

(1) S 与 l_i 在竖直方向不相交,即满足 $y_1>y_t^i$ 或 $y_2<y_b^i$;

(2) S 与 l_i 在竖直方向相交,且满足 $x^i=x_1$;

(3) S 与 l_i 在竖直方向相交,且满足 $x^i<x_1$.

它们对控制子网和 l_i 变化的影响如下:

(1) 由于 x^i 是第 y_b^i 行至第 y_t^i 行之间未知空闲节点机的最左边界,所以上述情况(1)中的 S 对 l_i 不产生影响,即不能得到控制子网,且 l_i 保持不变;

(2) 对于上述第(2)种情况来讲,假设交集为$[y_m,y_n]$($y_m=\max\{y_b^i,y_1\}$,$y_n=\min\{y_t^i,y_2\}$),由于 $x^i=x_1$,因此,子网 $S'(\langle x^i,y_m\rangle,\langle x_2,y_n\rangle)$是分配子网,因而$[y_m,y_n]$这些行之间未知空闲节点机的最左边界将变为 x_2+1,而$[y_b^i,y_m-1]$和$[y_n+1,y_t^i]$这些行之间未知空闲节点机的最左边界仍是 x_i.这种情况不能得到控制子网.注意,在得到新的 l_i 之后,可以和它前面的直线段进行合并,即可将线段$(x:y_b,y_t)$和$(x:y_b{}',y_b-1)$合并为$(x:y_b{}',y_t)$,这样可减少下一个分配子网比较的次数.

(3) 对于上述第 3 种情况,由于 busy_list 是排序的,第 k 个子网之后的左边界的位置均大于或等于 x_1,所以存在一个空闲子网$(\langle x^i,y_b^i\rangle,\langle x_1-1,y_t^i\rangle)$,且用(2)的分析方法同理可分析 l_i 的改变:交集部分的最左边界变为 x_2+1,其他保持不变.

上述(3)中得到的空闲子网不一定是控制子网.这是因为,若 l_{i-1} 满足 $x^{i-1}\leqslant x^i$,显然,$(\langle x^i,y_b^i\rangle,\langle x_1-1,y_t^{i-1}\rangle)$是包含它的一个更大的空闲子网.为此,我们首先根据 line_list 中的直线列求得满足下列条件的 p,q:

$$p=\{m\mid 0\leqslant m\leqslant i,x^j\leqslant x^i,\ i\geqslant j\geqslant m \text{ and } x^{m-1}>x^i\},\tag{1}$$

$$q=\{m\mid n-1\geqslant m\geqslant i,x^j\leqslant x^i,\ i\leqslant j\leqslant m \text{ and } m^{m+1}>x^i\}.\tag{2}$$

其中 n 为当前 line_list 中的直线段数,并令 $x^{-1}=L_x$,$x^n=L_x$.

由 p,q 可得新的控制子网$(\langle x^i,y_b^i\rangle,\langle x_1-1,y_t^p\rangle)$.

根据上述思想,初始时令 line list=$\{(0:0,L_y-1)\}$,然后用上述方法依次比较 busy_list 中的每个分配子网和 line_list 中的每条直线段,最后将 MR 看做是分配子网与 line_list 进行比较就可

以得到一种求网格多处理机中所有控制子网的算法.该算法中每次得到空闲子网时,通过计算式(1)和式(2)来得到控制子网是它的一种最直接的计算方法,但可能会重复比较,因而得到重复控制子网.为此,我们用下面的方法来实现分配子网与line_list中直线段的比较.

假设在分配子网$S(\langle x_1,y_1\rangle,\langle x_2,y_2\rangle)$与line_list中的直线段进行比较的过程中,我们用XL记录已得到的空闲子网按其左边界的降序排列的一个有序列,且XL中的空闲子网只明确记录了该子网的上边界,其下边界在S与后续直线段的比较中得到确定.

现在假设S与l_i直线段进行比较,首先将x'与XL中的空闲子网左下角的x坐标进行比较,找到它的按降序插入位置$f(\langle x_{f1},y_{f1}\rangle,\langle x_{f2},y_{f2}\rangle),(x_{f1}<x')$,则对于XL中的空闲子网和$l_i$得到的空闲子网来讲,它们有以下几种情况:

(1) 对排在f之前的XL中的空闲子网$(\langle x'_1,y'_1\rangle,\langle x'_2,y'_2\rangle)$来讲,仍不能确定它的$q$,$(\langle x'_1,y_h^i\rangle,\langle x'_2,y'_2\rangle)$是包含它的一个更大的空闲子网,所以保持不变;

(2) 若根据上述分析存在对应于l_i的空闲子网$(\langle x^i,y_h^i\rangle,\langle x_1-1,y_l^i\rangle)$且不存在XL中的空闲子网$(\langle x'_1,y'_1\rangle,\langle x'_2,y'_2\rangle)$满足$x'_1=x'$,则将该子网$y_l^i$修改为$y_{f2}$(即确定了该空闲子网对应的控制子网的上界),将它插入f之前;

(3) 显然,对于包括f在内的XL中后面的空闲子网$(\langle x'_1,y'_1\rangle,\langle x'_2,y'_2\rangle)$来讲,可以确定它的$q$为$i-1$,它所对应的控制子网为$(\langle x'_1,y_i+1\rangle,\langle x'_2,y'_2\rangle)$,将它按序插入free_list中,并将该空闲子网从XL中删除.

假设用xl_list表示XL,初始时令xl_list$=\varnothing$,然后将S与line_list按上述方法进行比较,最后再与MB进行比较,就是根据S与line_list计算控制子网的过程.将它与前面的分析结合,即得到求网格结构中所有控制子网的一种新算法.它的具体描述见算法1.

算法1. 搜索网格多处理机的空闲子网的算法

```
1  search_submesh( )
2  {
3    free_list=∅;
4    line_list={(0,0,L_y-1)};
5    for all S∈(busy_list∪MR) do{
6      xl_list=∅;
7      for all l_i∈(line_list∪MB) do{
8        用二分法查找 x' 在 xl_list 中的插入位置 f;
9          if(l_i 与 S 在竖直方向有交集){
10           if(x'<x_1 且 xl_list 不存在子网,满足它的左下角的 x=x')
11             将由上述(2)计算的空闲子网插入 xl_list 中 f 之前;
12           分析并计算 l_i 的变化;
13           合并邻接项;
14         }
15         按上述(3)将 xl_list 中排在 f 之后(包括 f)的子网对应的控制子网按序
           插入 free_list 中,并将这些子网从 xl_list 中删除;
16       }
17     }
```

下面举一个简单的例子.假设子网分配情况如图1所示(图中阴影节点为分配节点),求网格结构中所有的控制子网,其计算步骤如下:

(1) 初始line_list$=\{(0,0,7)\}$;

(2) 对分配子网$S(\langle 1,3\rangle,\langle 2,5\rangle)$与$(0,0,7)$进行比较,得到xl_list$=\{(\langle 0,0\rangle,\langle 0,7\rangle)$,且有

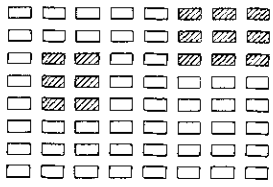

Fig.1 An example of submesh allocation
图1 一个子多分配的例子

line_list={(0:6,7),(3:3,5),(0:0,2)};与 MB 进行比较,得到控制子网(⟨0,0⟩,⟨0,7⟩).

(3) 对分配子网 S(⟨5,5⟩,⟨7,7⟩)与(0:6,7)进行比较,得到 xl list=(⟨0,6⟩,⟨4,7⟩),且(0:6,7)分解为(8:6,7);与(3:3,5)进行比较,得到控制子网(⟨0,6⟩,⟨4,7⟩),且有 xl_list={(3,3>,⟨4,7⟩)},且(3:3,5)分解为(8:5,5)和(3:3,4)两段;与(0:0,2)进行比较,由于 0<3,xl_list 保持不变,且(0:0,2)保持不变;与 MB 进行比较,得到控制了网(⟨3,0⟩,⟨4,7⟩.

合并 line list 中的邻接项后有 line list={(8:5,7),(0:0,2)};

(4) 将 MR 看做是分配子网.

同理,将它依次与 linc_list 中的 3 段和 MB 进行比较,得到控制子网(⟨3,0⟩,⟨7,4⟩)和(⟨0,0⟩⟨7,2⟩).

至此,我们已搜索到图 1 中所有的控制子网,它们分别是(⟨0,0⟩,⟨0,7⟩)、(⟨0,6⟩,⟨4,7⟩)、(⟨3,0⟩,⟨4,7⟩)、(⟨3,0⟩,⟨7,4⟩)、(⟨0,0⟩,⟨7,2⟩).

2 改进的最佳匹配类子网分配算法

为方便算法描述,首先给出 RF 算法中的几个相关概念.

2.1 基本概念

定义 7. 对一个大小为 $a\times b$ 或 $b\times a$ 子网的分配请求用(a,b)来表示.

定义 8. 一个子网 $S(\langle x_1,y_1\rangle,\langle x_2,y_2\rangle)$ 相对于一个子网 $S'(\langle x'_1,y'_1\rangle,\langle x'_2,y'_2\rangle)$的保留因子用 $RF(S,S')$表示,定义为 S'剩余子网中最大子网的面积:

$$RF(S,S')=\begin{cases}\max\{(x_1-x'_1)\cdot l'_y,(x'_2-x_2)\cdot l'_y,l'_x\cdot(y_1-y'_1),l'_x\cdot(y'_2-y_2)\} & \text{若 } S \text{ 与 } S' \text{ 重叠}\\ l'_x\cdot l'_y(=S'\text{的大小}) & \text{其他}\end{cases}$$

其中 $l'_x=x'_2-x'_1+1$,$l'_y=y'_2-y'_1+1$,max{ * }返回最大值.

定义 8 中保留因子的概念是文献[7]中的最佳匹配类子网分配算法的依据,它选择相对于控制子网的保留因了最大的候选了网进行分配,从而尽可能地保留了最大面积的控制了网,可用于后续任务的分配请求.

定义 9. 对一个请求子网(a,b),一个候选子网是一个大小为(a,b)或(b,a)的空闭子网,它由 free_list 中的控制子网产生.

2.2 改进的最佳匹配类子网分配算法

假设 cand_list 是请求子网(a,b)的候选子网列.

用第 1 节中的搜索控制子网的算法代替文献[7]中的计算控制子网的算法,并对算法进行部分改进,得到了以下改进的最佳匹配子网分配算法及相应的子网释放算法.

算法 2. 改进的最佳匹配子网分配算法

```
1  Alloc(a,b)
2  {
3     cand_list=∅;
4     while(cand_list=∅){
```

```
5       cand_list=cand(a,b);
6       if (cand_list==∅)
7         等待,直到一个子网被释放;
8     }
9     for all S∈free_list
10      计算 cand_list 中每一个候选子网对 S 的保留因子;
11      只保留 cand_list 中保留因子最大的候选子网;
12      if (cand_list 中只有一个候选子网)
13          将它赋给所需任务,并将该子网按左下角 x 坐标的增序加入 busy_list 中;
14        return;
15    }
16    选择相对于((0,0),(L_x-1,L_y-1))的保留因子最大的候选子网,将它赋给所需任务,并将该子网按左
      下角 x 坐标的增序加入 busy_list 中;
17  }
```

算法 3. 子网释放算法

```
1  dealloc(R)
2  {
3    从 busy_list 中删除 R
4  }
```

其中 $cand(a,b)$ 用于计算请求子网 (a,b) 的候选子网,只需将算法 1 稍加修改(在每次得到新的控制子网时,计算它是否可分配空闲子网给请求子网 (a,b),若可以,则在它的 4 个顶点处生成候选子网,加入 cand_list 中)即可得到,故这里省略它的具体算法描述.

算法 2 中的第 4 步保证了该候选子网相对于整个网格多处理机的保留因子最大,从而保证了只有在该任务存在时,才有大的子网可用于分配.

2.3 算法分析

下面分析算法 1~3 在最坏情况下的时间复杂度.

假设 busy_list 中的分配子网数为 N_a,计算后,free_list 中的控制子网数为 N_f.

由于 busy_list 是排序的,所以,算法 3 中可以首先用二分法查找到该子网,然后释放,故它的时间复杂度为 $O(\log_2 N_a)$.

由于最坏情况下 line_list 中的直线段数为 $2N_a+1$ 条,所以算法 1 中语句 5 和语句 7 的执行次数均为 $O(N_a)$ 次,语句 8 的时间复杂度为 $O(\log_2 N_a)$,语句 15 在两个循环体内的执行次数的总和为 $O(N_f)$,而插入操作的时间复杂度为 $O(\log_a N_f)$,其他语句的执行时间为 $O(1)$. 根据文献[7]中定理 1(控制子网的最大数小于或等于 $4N_a$)可知,算法 1 的时间复杂度为 $O(N_a^2 \cdot \log_2 N_a)$,这也是本文提出的新的搜索控制子网算法的时间复杂度,它优于文献[7]中 $O(N_a^3)$ 的搜索算法.

算法 2 中语句 5 的时间复杂度为 $O(N_a^2 \cdot \log_2 N_a)$. 由于候选子网在控制子网的 4 个角生成,所以最多有 $4N_f$ 个候选子网,算法 2 中语句 9~11 的执行次数均为 $O(N_f)$,语句 9~16 的时间复杂度为 $O(N_f^3)$. 故算法 2 的时间复杂度为 $O(N_a^2 \cdot \log_2 N_a)$.

将已有的几种最佳匹配类子网分配算法的时间复杂度与本文的改进算法的时间复杂度进行比较,见表 1.

从表 1 可以看出,本文的改进算法在时间复杂度上优于 Buddy,BF,BL 和 RF 分配算法,且根据文章开始部分中指出的 RF 的匹配准则优于 BL 和 FL 的匹配准则,所以,基于它的匹配准则的改进算法也优于 BL 和 FL 算法.

Table 1 Time complexity of best-fit submesh allocation strategies

表 1 几种最佳匹配类子网分配算法的时间复杂度

Strategies①	Allocation complexity②	Deallocation complexity③	Complete recognition④	Internal fragmentation⑤	Type⑥
Buddy	$O(N)$	$O(N)$	No	Yes	Best-Fit⑦
BF	$O(N)$	$O(N)$	No	No	Best-Fit
BL	$O(N_a^3)$	$O(1)$	Yes	No	Best-Fit
FL	$O(N_f^2)$	$O(N_f^2)$	Yes	No	Best-Fit
RF	$O(N_f^2)$	$O(N_a^3)$	Yes	No	Best-Fit
Improved⑧	$O(N_a^2 \cdot \log_2 N_a)$	$O(\log_2 N_a)$	Yes	No	Best-Fit

$N=L_x * L_y$ is the number of nodes in a mesh, N_a is number of allocated submeshs in busy_list, N_f is the number of free submeshes in the free_list.⑨

①算法,②分配复杂度,③释放复杂度,④完全识别,⑤内部片段,⑥类型,⑦最佳匹配,⑧改进,

⑨$N=L_x * L_y$ 是网格多处理机的节点数,N_a 是 busy_list 中的分配子网数,N_f 是 free_list 中的空闲子网数.

3 算法模拟

我们在 UNIX 环境下用 client-server 结构对改进算法和文献[7]中的子网分配算法进行了模拟.其中,server 的作用相当于一个子网分配和释放器,它接受来自 client 的节点机分配或释放请求,然后执行相应的算法,返回结果给 client. client 的作用相当于一个任务生成器,它调用随机函数生成任务执行所需的处理机数 $w \times h$ 及其假设的执行时间 seconds,然后发送处理机分配请求给 server,当接受到分配结果后,调用 fork()函数生成子进程对该任务进行管理,由子进程负责向 server 发送该任务的节点机释放请求.

在模拟过程中假设任务数为 90,任务队列的调度模型为先来先服务,并假设任务的执行时间为 6～10 秒.

在模拟过程中用 90 个任务的完成时间和任务的平均等待时间来衡量算法的性能,其中任务的平均等待时间定义为各任务到达队首和到它得到处理机这段时间的平均值.

实验结果见表 2.

Table 2 The completion time and mean waiting time of ninety tasks (s)

表 2 90 个任务的完成时间和平均等待时间 (秒)

L_x, L_y		256	128	62	32	16
Completion time①	RF	245.08	244.06	249.03	237.20	220.89
	Improved②	215.83	213.84	225.80	222.73	199.71
Mean waiting time③	RF	2.64	2.63	2.69	2.56	2.35
	Improved	2.31	2.30	2.43	2.39	2.12

①完成时间,②改进,③平均等待时间.

从表 2 中可以看出,本文的改进算法优于文献[7]中的 RF 算法.

4 结 论

本文首先提出了一种新的搜索网格多处理机中控制子网的算法,它具有 $O(N_a^2 \cdot \log_2 N_a)$ 的时间复杂度,优于文献[7]中 $O(N_a^3)$ 的搜索算法.然后将其与文献[7]中的最佳匹配准则结合,对文献[7]中的 RF 分配算法进行了改进.算法分析及算法模拟均表明,改进算法优于原有算法.

References:

[1] Ding, J., Bhuyan, L.N. An adaptive submesh allocation strategy for two-dimensional mesh connected systems. In: Choudhary, A.N., Berra, P.B., eds. Proceedings of the 1993 International Conference on Parallel Processing, Vol II Software.

Boca Raton: CRC Press, 1993. 193~200.

[2] Yoo, Seong-Moo, Yong, Hee Youn. An efficient task allocation scheme for 2D mesh architectures. IEEE Transactions on Parallel and Distributed Systems, 1997,8(9):934~942.

[3] Sharma, D. D., Pradhan, D. K. A fast and efficient strategy for submesh allocation in mesh-connected parallel computers. In: Jurezuk, M., ed. Proceedings of the 5th IEEE Symposium on Parallel and Distributed Processing. Los Alamitos: Soc. Press, 1993. 682~689.

[4] Liu, T., Huang, W-K., Lombardi, F., *et al*. A submesh allocation scheme for mesh-connected multiprocessor systems. In: Banerjee, P., ed. Proceedings of the 24th International Conference on Parallel Processing, Vol II Software. Boca Raton: CRC Press, 1995. 159~163.

[5] Li, K., Cheng, K. H. A two dimensional Buddy system for dynamic resource allocation in a partitionable mesh connected system. Journal of Parallel and Distributed Computing, 1991,12(5):79~83.

[6] Zhu, Y. Efficient processors allocation strategies for mesh-connected parallel computers. Journal of Parallel and Distributed Computing, 1992,16(12):328~337.

[7] Geunmo, Kim, Hyunsoo, Yoon. On submesh allocation for mesh multicomputers: a best-fit allocation and a virtual submesh allocation for faulty meshes. IEEE Transactions on Parallel and Distributed Systems, 1998,9(2):175~185.

An Improved Submesh Allocation Scheme for Mesh Multicomputers*

ZHANG Yan, SUN Shi-xin, PENG Wen-qin

(*College of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu* 610054, *China*)

E-mail: zhangyan_hong@263.net

http://www.uestc.edu.cn

Abstract: The submesh allocation problem is to recognize and allocate a free submesh that can accommodate a request for a submesh of a specified size. Firstly, a new algorithm of searching free submesh for mesh multicomputers is proposed, which time complexity is $O(N_a^2 \cdot \log_2 N_a)$, better than those existed algorithms whose time complexity are $O(N_a^3)$. Then this new algorithm is used to improve the best-fit allocation scheme which is based on reservation factor——RF scheme, and the result is better than it.

Key words: words submesh allocation; mesh; free submesh

* Received January 21, 2000; accepted April 13, 2000

Supported by the Defence Pre-Research Project of the 'Ninth Five-Year-Plan' of China under Grant No. 16.1.4.1