一种在异构系统中实现负载平衡的方法*

金之雁
(中国气象科学研究院 ,北京 100081)
王鼎兴
(清华大学计算机科学系 ,北京 100080)

摘　要

提出了在异构系统实现负载平衡的区域分解算法和实现负载平衡的计算方法 ,利用它的负反馈性质解决了异构系统处理机计算速度测量误差造成的负载测量不准问题 ,并对处理机速度变化 ,速度测量误差 、处理机数量 、网格点计算量的分布等因素的影响进行了计算 ,结果表明本方法具有很强的平衡负载能力和较强的适应性 ;根据计算结果提出了解决模式网格点计算量不易测量问题的解决方案 ,并用扩散方程和模拟物理过程进行试验 ,试验表明这种方法是可行的 ,平衡负载的效果十分显著 。

关键词 : 并行计算　动态负载平衡　区域分解　数值天气预报

引　言

分布式并行处理技术已被越来越多的人所接受 ,机群系统以其低廉的价格和优异的性能得到众多用户的青睐 ,无论在国内还是国外 ,将机群系统用于数值预报研究 、开发的单位越来越多 ,有的甚至用于日常数值预报业务 ,取得了很好的效果 。但是分布式并行处理实现负载平衡比共享并行困难得多 。目前很多数值预报模式 ,如 MM5[1] , WRF[2] ,德国的 GME 模式[3]都未涉及该问题[4] ,这在很大程度上限制了模式的并行规模和应用范围 。

负载平衡可以分为动态与静态两种 ,如果在运行之前可以预知模式的计算负载分布情况 ,负载的划分是静态负载平衡问题 。反之 ,需要在模式运行时监测负载情况并根据监测信息动态调整模式负载的划分 ,是动态负载平衡问题 。数值预报的负载不平衡来源于多个方面 ,如积云对流参数化和云微物理过程等 ,这些都是随天气情况变化的计算负载 。另一个可能的负载不平衡来源是计算机 。由于采用机群系统的用户越来越多 ,很多用户需要用不同型号的微机或工作站和高速互联网组成机群系统 。如果并行系统中每个结点是相同的 ,称为同构系统 ,反之称为异构系统 。采用异构系统的数值预报模式必须有相应的负载平衡措施 ,否则 ,模式运算速度取决于系统中最慢的处理机的运算速度 。由于异构

* 本文得到国家自然科学基金项目 40245023 ,60273007 ,60121160743 和国家科技攻关计划 2001 DA607B 项目的资助 。
2002-02-26 收到 ,2002-06-25 收到修改稿 。

系统的构成变化很大,每个节点的计算速度不易测量,造成计算量估算不准,影响负载平衡的实现。本文提出了一种负载平衡方法,它具有负反馈性质,可以在对计算量估算不准的情况下平衡负载,解决数值预报模式的静态和动态负载平衡问题。

1　区域分解算法

首先我们定义负载平衡函数,设 P 个处理机完成一个时间步的时间为 T_l, $l=0,1,\cdots,P-1$。定义:

$$T_{av}=\frac{1}{P}\sum_{k=0}^{P-1}T_k \qquad T_{\max}=\max_{0\leqslant k\leqslant P-1}T_k$$

其中 T_{av} 为各处理机平均计算时间, $T_{\max}$ 为各处理机中最大的计算时间,也就是并行处理时间。负载平衡函数为:

$$I=\frac{T_{\max}-T_{av}}{T_{av}} \tag{1}$$

显然, I 为零时负载达到平衡,大于零表示负载不平衡。

区域分解是实现分布式并行处理的基础,作者曾提出在同构环境中的区域划分方法[5],本文将该方法扩展到异构系统中。定义预报区域 G 是矩形区域,东西、南北方向的网格点数各为 N_x、N_y。格点 (i,j) 的计算负载为 w_{ij},负载总和为 $W=\sum_{(i,j)\in G}w_{ij}$。参加运算的处理机数量为 P。如果,第 l 个处理机的计算速度是 s_l,总计算速度 $S=\sum_{l=1}^{P}s_l$。若第 l 个处理机的负载为 $\tilde{W}_l=\frac{W}{S}s_l$,则达到负载平衡。将处理机分成 $N=\left\lfloor\sqrt{P\frac{N_x}{N_y}}\right\rfloor$ 个组,第 k 组内有 $\hat{N}_k$ 个处理机:

$$\hat{N}_k=\begin{cases}\left\lfloor\frac{P}{N}\right\rfloor+1 & k\leqslant \mathrm{Mod}(P,N)\\ \left\lfloor\frac{P}{N}\right\rfloor & k>\mathrm{Mod}(P,N)\end{cases}\qquad k\in 1,N$$

其中 $\mathrm{Mod}(P,N)$ 是 P 被 N 除的余数,$\lfloor\ \rfloor$ 表示舍去小数部分取整。处理机 1 至 $\hat{N}_1$ 为第 1 组,$\hat{N}_1+1$ 至 $\hat{N}_2$ 为第 2 组,以此类推。记

$$\hat{W}_k=\sum_{i=1}^{k}\tilde{W}_i$$

$$\tilde{W}_l=\sum_{i=1}^{k}\tilde{W}_i,\quad k=\sum_{i=1}^{l}\hat{N}_i$$

首先按以下方法将区域 G 划分成 N 个区域:

令 $k=1$

对 j 从 N_y 至 1 增量为 -1 循环开始

　对 i 从 1 至 N_x 循环开始

　　将 (i,j) 点标为 k

计算 $\sum w_{ij}$,如果$\left|\sum w_{ij}-\widetilde{W}_k\right|$达到最小值, $k=k+1$

对 i 循环结束

对 j 循环结束

将 N 个区域的所有格点按照内循环 i 由小至大,中循环 j 由小至大,外循环是区域 k,由 1 至 N 的顺序排列成一维数组,即将所有格点从 1 至 N 区排列,区内按照 i 从小至大, j 从小至大的顺序排列成一维数组,设下标为 l。按照以下方法将 G 划分成 P 个子区域:

令 $SUM=0$

令 $k=1$

对 l 从 1 至 $N_x\cdot N_y$ 循环

格点 l 标记为 k

令 $SUM=SUM+w_l$

如果$\left|SUM-\widehat{W}_k\right|$达到最小,令 $k=k+1$

对 l 循环结束

设 G 内第 k 子区域的计算负载为 $\widehat{W}_k$,显然有

$$\left|\widehat{W}_k-\widetilde{W}\right|<w_{\max}$$

其中, $w_{\max}=\max(w_{ij})$, $(i,j)\in G$,所以对任意两个子区域有

$$\left|\widehat{W}_k-\widehat{W}_{k'}\right|<2w_{\max}$$

由于该方法允许将行或列断开使得子区域的形状出现小台阶,每一个台阶会增加一个格点的通信量,最多会增加 $P-1$ 个格点的通信量。所以,通信的增加量是很少的。

2 实现负载平衡的负反馈方法

如果可以精确测定每个处理机的处理速度和每个格点的计算负载,上述方法就可以近似达到负载平衡,但是测定处理机的计算速度本身就是一个复杂的问题,它与处理机硬件、编译器、编程、内存的使用等很多因素相关。其次,每个网格点的计算量的测定也是与处理机硬件、编译器、编程的优化程度等因素有关,精确测定每个网格点的计算量是不现实,甚至是不可能的。因此必须研究在不能精确测定处理机计算速度和每个网格点的计算负载条件下实现负载平衡的方法。

本文提出的方法如下:

步骤 1:利用一个速度估算子程序估算处理机的计算速度。该子程序只是进行一些计算,这些计算应能代表模式的计算特点,如加、减、乘、除、指数运算等,测出该子程序的计算时间 T'_l,令 $s'_l=\frac{1}{T'_l}$, $l=1$, P 得到处理机速度的估算值。

步骤 2:假定每个网格点的计算量为 $w'_{ij}=1$。

步骤 3:按照第一节的方法,对网格点进行分区。并将数据按照分区进行分配。

步骤 4:进行一个时间步长的计算,在计算过程中,测出每个网格点的计算时间 t_{ij}和

每个处理机的总计算时间 T_l

步骤 5:根据式(1)计算负载平衡函数 I

步骤 6:如果 I 小于阈值 ξ,则保持现有的分区,跳到步骤 4 进行下一个时间步长的计算;如果 I 大于阈值 ξ,令 $w'_{ij}=s'_l t_{ij}$,其中 l 为格点 i,j 所在的处理机编号。跳到步骤 3 重新进行分区。重复上述过程直至结束。

该方法可以在不精确测定处理机计算速度的情况下,利用它的负反馈性质逐渐逼近负载平衡。具体过程是,如果 s'_l 大于真实计算速度,分配在处理机 l 上的计算负载会多一些,在第 2 次对格点的计算负载进行估算时,由于 t_{ij} 是实测值,所以 $w'_{ij}=s'_l t_{ij}$,就会比实际计算负载大一些,如果第 2 次循环时区域划分变化不大,分配在该处理机上的部分网格点就会移到其它处理机上,降低该处理机的总计算负载。可以看出,该方法不能精确测定处理机计算速度和网格点的计算负载。其负反馈机制是通过调整每个网格点的"计算负载"实现的,很显然,这种负反馈机制只有在区域划分变化不大,多数网格点仍然处于同一个处理机的情况下才有作用。如果 s'_l 与实际相差太大,就会使同一个网格点的计算量 $w'_{ij}=s'_l t_{ij}$ 在不同处理机上差别很大,循环的收敛速度就会很慢,甚至不收敛。

3　负载平衡方法试验

为了考察该方法平衡负载的能力,我们设计了针对数值预报的并行处理的试验。设预报区域的网格点数量 $N_x\times N_y$ 为 320×160,在区域中部,有半径为 10 个格距的区域计算负载是周围其它格点的 c 倍,即

$$w_{ij}\begin{cases} c, & (i-\frac{N_x}{2})^2+(j-\frac{N_y}{2})^2 \leqslant 100 \\ 1, & (i-\frac{N_x}{2})^2+(j-\frac{N_y}{2})^2 > 100 \end{cases}$$

用来模拟格点之间负载不均匀的情况。分别用 32、64、128、256 个处理机进行试验。假定处理机速度为:

$$s_l = 1 + \mathrm{random}(r)$$

其中,random(r)为 0 ~ r 之间的随机数。这样,最快与最慢处理速度之比约为 r:1,"测定"的处理机速度为:

$$s'_l = (1+\mathrm{rand}(a))\,s_l$$

其中 rand(a)是($-a$,a)之间的随机数,a 分别取为 10 %、20 %、30 %,相当于测量误差。试验方法如下:

步骤 1:用随机数发生器计算出处理机的"实际"计算速度 s_l 和"测量"计算速度 s'_l

步骤 2:令 $w'_{ij}=1$,用 s'_l、w'_{ij} 和第 1 节介绍的方法进行区域划分,并令 $m=0$。

步骤 3:每个格点的计算时间用 w'_{ij} 和 s_l 求出,令 $w'_{ij}=\frac{w_{ij}}{s_l}s'_l$,用 s'_l 和 w'_{ij} 再次进行区域划分。令 $m=m+1$。

步骤 4:将每个区域的 w_{ij} 求和并除以 s_l,求出每个处理机的计算时间 T_l,用式(1)计

算负载平衡函数 I,如果 $I > \xi$,跳到步骤 3,反之结束,考察 m 的值。

为了减少试验偶然性的影响,重复上述试验 100 次,m 取 100 次试验中的最大值。

我们分别取 $\xi = 5\%$, $a = 0.1$、0.2、0.3, $r = 2.0$、4.0、6.0、8.0, $c = 2$、4、8, $P = 16$, 32, 64, 128, 256 进行试验,表 1 是试验得到的 m 的值。

表 1 测定每个网格点计算时间条件下,P 取为 16, 32, 64, 128, 256 达到负载平衡的循环次数 m

		a														
		0.1					0.2					0.3				
r	c	16	32	64	128	256	16	32	64	128	256	16	32	64	128	256
2.0	2	1	1	1	1	2	1	1	1	2	2	2	2	2	3	5
	4	1	1	1	2	2	1	1	2	2	3	2	2	2	3	5
	8	1	1	2	2	2	2	2	2	2	4	2	2	3	3	6
4.0	2	1	1	1	1	2	2	2	2	2	3	2	2	2	3	5
	4	1	1	1	2	2	2	2	2	2	3	2	2	2	3	6
	8	1	2	2	2	3	2	2	2	3	5	2	2	3	4	7
6.0	2	1	1	1	2	2	2	2	2	2	3	2	2	2	4	6
	4	1	1	1	2	2	2	2	2	3	3	2	2	2	4	6
	8	1	2	2	2	8	2	2	2	3	4	2	2	3	4	N
8.0	2	1	1	1	2	2	2	2	2	3	3	2	2	2	3	6
	4	1	1	2	2	2	2	2	2	3	3	2	2	2	4	6
	8	1	2	2	2	N	2	2	2	3	N	2	2	3	4	N

注:其中 N 表示在 30 次循环之内未能达到 $I < \xi$,下同。

由表 1 可以看出,在处理机数量不多的情况下,处理机速度的误差影响并不大,对于处理机数量较多时,应尽量将处理机速度测量准确。系统内处理机速度相差越大,循环次数越多,尤其是在处理机数量比较多的时候,所以淘汰一些处理速度过慢的节点有利于提高平衡负载的速度。格点间计算量相差越大,实现负载平衡越困难。但在绝大多数情况下,循环 2 ~ 3 次均可实现负载的均衡。只有个别情况下循环次数多一些或者不能在 30 次内达到负载平衡函数小于 5%。所以本方法平衡负载的效果是比较好的。

在上述方法中我们假定可以测量每个网格点的计算时间,但有时测量每个网格点的计算时间是很难的,因此,我们还试验了在不能测量每个网格点的计算时间时的处理方法。我们用每个处理机上每个网格点的平均处理时间来替代每个网格点的计算时间。具体方法是将步骤 3 改为:将各处理机上的网格点的 w_{ij} 求和并算出每个处理机上的网格点数量 N_l,令 $w'_{ij} = \frac{\sum_{(i,j)\in \text{处理机}l} w_{ij}}{s_l N_l} s'_l$,用 s'_l 和 w'_{ij} 再次进行区域划分。令 $m = m + 1$。其余步骤与原来相同(表 2)。

可以看出,如果网格点的计算量相差在一倍左右时($c = 2$),两种方法的速度差不多,如果网格点计算量相差在 4 倍,平均负载与实际负载相差很大,收敛速度就会很慢,若相差 8 倍以上基本不收敛。

在数值预报模式中,动力框架部分的计算负载是比较均匀的,但是测定每一个网格点的计算量比较困难,可以用平均网格点计算量,即便有些误差(一般误差很少能超出 1 倍)影响也不大。对于物理过程部分,网格点之间负载相差可以很大,但测定每个网格点的计

算时间却比较方便。模式中可将两种方法结合使用。

表 2　不能测定每个网格点计算时间条件下,P 取为 16,32,64,128,256 达到负载平衡的循环次数 m

		α														
		0.1					0.2					0.3				
r	c	16	32	64	128	256	16	32	64	128	256	16	32	64	128	256
2.0	2	1	1	2	1	2	1	2	2	2	3	2	2	2	3	5
	4	2	2	4	3	4	2	3	5	3	6	2	2	5	4	12
	8	3	4	12	N	N	5	N	N	N	N	7	N	N	N	N
4.0	2	1	1	2	2	3	2	2	2	2	3	2	2	3	4	5
	4	2	2	3	3	4	3	3	5	4	7	3	3	5	4	10
	8	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
6.0	2	1	1	1	2	2	2	2	2	2	3	2	2	3	4	6
	4	2	2	3	3	7	2	2	7	4	5	3	3	7	5	9
	8	N	6	N	9	N	N	5	N	N	N	N	4	N	10	N
8.0	2	1	1	2	2	2	2	2	2	3	3	2	2	3	4	6
	4	2	2	5	3	N	2	2	5	4	6	3	3	5	5	9
	8	6	4	N	N	N	N	6	N	7	N	N	6	N	N	N

4　动态负载平衡试验

数值天气预报的计算负载是随计算过程而变化的,需要在计算过程中,根据测定计算负载的分布情况动态地调整每个节点的计算量,是一种动态的负载平衡过程。根据 R. W. Ford[6]的研究,数值预报模式中每个网格点的计算量随时间积分过程是一个缓慢的变化过程,上一个时间步与下一个时间步的计算时间相差不大。所以,调整一次负载,可在一段时间内起作用。很显然,如果可在 1 个循环步内将负载调整好,本方法也可用对负载进行动态调整。对于需要多个时间步循环才能实现负载平衡,同时负载还在变化,效果就比较差了。

数值预报程序量庞大,直接试验困难比较大。因此,我们利用一个简化的模型进行试验。数值预报计算主要集中在动力部分和物理过程部分。随着模式的细化,物理过程的计算量越来越大,可以达到动力部分的几倍到十几倍,且分辨率越高,所占比重越大。它也是负载不平衡的主要来源。

二阶线性扩散方程虽只是数值预报模式的一小部分,但是它与模式动力过程的计算特点以及数据通信方式是一样的,因此用它模拟模式动力过程部分的计算可以说明问题,其方程是:

$$\frac{\partial F}{\partial t} = K_x \frac{\partial^2 F}{\partial x^2} + K_y \frac{\partial^2 F}{\partial y^2} + K_z \frac{\partial^2 F}{\partial z^2}$$

离散化形式为:

$$\frac{F^{t+\Delta t}_{i,j,k} - F^{t}_{i,j,k}}{\Delta t} = K_x \frac{F^{t}_{i+1,j,k} + F^{t}_{i-1,j,k} - 2F^{t}_{i,j,k}}{\Delta x^2} + K_x \frac{F^{t}_{i,j+1,k} + F^{t}_{i,j-1,k} - 2F^{t}_{i,j,k}}{\Delta y^2} + K_x \frac{F^{t}_{i,j,k+1} + F^{t}_{i,j,k-1} - 2F^{t}_{i,j,k}}{\Delta z^2}$$

其中 F 是扩散变量，K_x，K_y，K_z 分别是 x，y，z 三方向上的扩散系数，Δt，Δx，Δy，Δz 是时间步长和三个方向上的格距。离散化后的网格点是（N_x，N_y，N_z）阶三维矩阵，取固定边条件。

物理过程每个格点的计算彼此相互独立，计算量由该点的天气、地理、时间等因素决定。我们采用在格点（i，j）上计算 sin 和 cos 函数 Z_{ij} 次来模拟物理过程。Z_{ij} 定义为：

$$Z_{ij} = \begin{cases} 5000 & (i - i_c)^2 + (j - j_c)^2 \leqslant 100 \\ 500 & (i - i_c)^2 + (j - j_c)^2 > 100 \end{cases} \tag{14}$$

表示在格点（i，j）上要计算 Z_{ij} 次 sin 和 cos 函数。其中（i_c，j_c）是“天气系统”的中心，该中心以 20 个时间步一个格距的速度自西北向东南方向移动，预报区域是 101 × 101 的正方形区域，垂直层次有 100 层（图 1）。

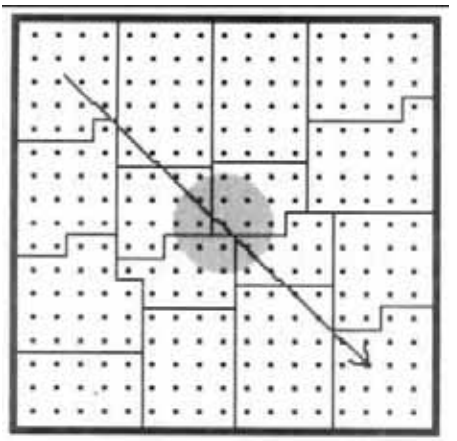

图 1 预报区域示意图，阴影区为计算负载较大区域，沿箭头方向移动

计算过程如下：

假定每个网格点的计算相同，计算分区并将数据分布到每个处理机上；

时间积分循环开始：

每个处理机从周围处理机上取得所需要的数据；

每个处理机在相应的区域内进行本时间步的计算并对每个网格的计算时间计时，对本处理机的总计算时间计时； 计算负载平衡函数；

如果负载平衡函数连续大于给定阈值 5 次，则根据测得的每个网格点的计算量重新计算分区并按照新的分区将数据重分布。

时间积分循环结束。

试验研究在 IBM 的 SP 计算机上进行，它有两种类型的结点，POWER3 和 POWER PC，在试验中，我们使用了两个 POWER3 节点和两个 POWER PC 节点。在试验中我们发现，系统有时会出现随机扰动，对负载平衡影响很大。我们用连续大于给定阈值 5 次作为触发数据重分布的条件以避免随机扰动的影响。试验中我们取阈值为 0.1 并与不进行数据重分布进行对比。图 2 是试验结果。可以看出，采用了动态负载平衡的效果比未采

用该技术的效果要好得多。通过几次调整,负载平衡函数维持在比较低的水平;阈值的选取对调整的次数有很大影响,较小的阈值虽然可以使负载更加平衡,但是调整次数过多本身也会有很大的开销。应选取适合的阈值。

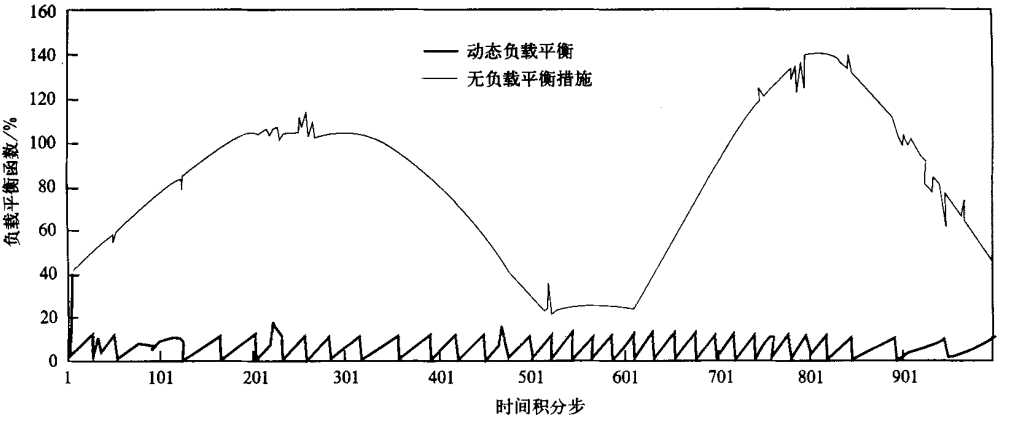

图 2　采用和不采用动态负载平衡方法的对比

5　结　论

本文设计了一种在异构系统中实现负载平衡的区域分解算法,并提出了在不能精确测定处理机计算速度时实现负载平衡的方法,并针对处理机速度高低分布、处理机速度误差、处理机数量、网格点计算量分布、网格点计算量确定方法等因素的影响进行计算,结果表明本方法具有很强的平衡负载能力和较强的适应性,在处理机速度对比,网格点计算量对比、处理机速度误差都比较大的情况下仍然可以比较好地平衡负载;在网格点计算量比较均匀(格点计算量相差约 1 倍)的情况下,可以用平均计算量代替网格点计算量,但是如果计算量相差比较悬殊,就不能用平均网格点计算量替代;提出了针对数值模式动力框架和物理过程计算量分布的不同特点将二者结合的方式解决模式网格点计算量的测量问题的方法,并用扩散方程和模拟物理过程进行试验,试验表明这种方法是可行的,平衡负载的效果十分显著。

参 考 文 献

1　Michalakes J, Canfield T, Nanjundiah R, et al. Parallel implementation, validation and performance of MM5, coming of age. Proceedings of the Sixth ECMWF Workshop on the Use of Parallel Processors in Meteorology. World Scientific, River Edge, New Jersey, 1995. 266 ~ 276.

2　Michalakes J, Dudhia J, Gill D, et al. Design of a next-generation regional weather research and forecast model, towards teracomputing. Proceedings of the Eighth ECMWF Workshop on the Use of Parallel Processors in Meteorology. World Scientific, River Edge, New Jersey, 1999. 117 ~ 123.

3　Schattler U. Model development for parallel computers at DWD, making its mark. Proceedings of the Seventh ECMWF

Workshop on the Use of parallel Processors in Meteorology. World Scientific, River Edge, New Jersey, 1997. 83~99.

4 舒继武,郑纬民,沈美明,等,大规模问题数据并行性能的分析.软件学报,2000,**11**(5):628~633.

5 Ford R W, Burton P M. Load balancing physics routines, towards teracomputing. Proceedings of the Eighth ECMWF Workshop on the Use of parallel Processors in Meteorology. World Scientific, River Edge, New Jersey, 1999. 147~159.

6 金之雁,王鼎兴.一种有限差分格式负载平衡区域分解方法.气象学报,2002,**60**(2):188~193.

AN ALGORITHM FOR LOAD BALANCING IN A HETEROGENEOUS SYSTEM

Jin Zhiyan

(*Chinese Academy of Meteorological Sciences, Beijing* 100081)

Wang Dingxing

(*Department of Computer Science, Tsinghua University, Beijing* 100084)

Abstract

Load balancing is a crucial problem in a heterogeneous system such as PC or workstation clusters, which has been widely used in the research and development of numerical weather prediction, A load balancing algorithm, based on feedback, is presented to eliminate the influence of uncertainty of processor speed, which is difficult to measure precisely. The influences of variation of processor speed, load distribution, errors of tested processor speed, etc., have been calculated. The results show that this method is quite robust. The diffusion equation and the simulated physical processes are used to test the algorithm, which shows that it is feasible and can balance the load quite well.

Key words: Parallel computing Dynamic load balancing Area decomposition Numerical weather prediction