异构系统中负载平衡扩散算法的加速方法*

金之雁[1+], 王鼎兴[2]

[1](中国气象科学研究院,北京 100081)

[2](清华大学 计算机科学与技术系,北京 100084)

An Optimal Method of Diffusion Algorithm for Hetergeneous System

JIN Zhi-Yan[1+], WANG Ding-Xing[2]

[1](Chinese Academy of Meteorological Science, Beijing 100081, China)

[2](Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

+ Corresponding author: Phn: 86-10-68409911 ext 5516, E-mail: jinzy@cma.gov.cn

http://www.cams.cma.gov.cn

Received 2002-04-04; Accepted 2002-09-02

Jin ZY, Wang DX. An optimal method of diffusion algorithm for hetergeneous system. *Journal of Software*, 2003,14(5):904~910.

http://www.jos.org.cn/1000-9825/14/904.htm

Abstract: Many organizations nowadays operate local area networks connecting hundreds of workstations and personal computers and use them as a cluster system. Dynamic load balancing is an important method to improve the performance on such heterogeneous system. Diffusion algorithm is a dynamic load balancing method for homogeneous system. In this paper, the diffusion algorithm is extended to heterogeneous system, the influence of the arrangement of different processors in the system on the convergence rate of the diffusion algorithm is studied, and an optimal method is proposed to improve the convergence rate. Primary result shows that it can find the better arrangement of the processors to accelerate the diffusion algorithm.

Key words: parallel computing; heterogeneous system; dynamic load balancing; diffusion algorithm; convergence rate; arrangement of processors

摘 要: 目前,很多单位与组织都有连接着数百台工作站和微机的局域网,并将它们作为一个机群系统使用.在这样的异构系统上动态负载平衡是提高性能的一个重要方法.扩散方法是同构系统的动态负载平衡算法.将散算法扩展到异构系统中,对异构系统中速度不同的处理机的位置与扩散收敛速度的关系进行了研究,提出了加速扩散算法的收敛速度的优化方法.初步实验证明,该方法能通过合理安排处理机,加快扩散算法的速度.

关键词: 并行计算;异构系统;动态负载平衡;扩散算法;收敛速度;处理机安排

中图法分类号: TP301 **文献标识码**: A

* Supported by the National Natural Science Foundation of China under Grant Nos.40245023, 60273007, 60131160743 (国家自然科学基金)

第一作者简介: 金之雁(1962－),男,江苏吴县人,正研级高级工程师,主要研究领域为数值天气预报,并行计算.

随着分布式计算技术的普及,在分布式并行系统上开展科学计算的学者越来越多.如果并行系统的所有节点完全相同,这种系统称为同构的,反之称为异构的.异构并行系统在工作站网络环境中是很常见的.

负载平衡问题是许多研究学者的研究焦点[1,2].如果计算负载可以在运行之前确定,可以事先将负载划分,这是静态负载平衡问题;若只能在运行时测量计算负载并动态确定负载划分,则是动态负载平衡问题.对于异构系统,不同体系结构的处理机的计算速度、利用率等等很难事先估算,静态负载平衡需要很多假定;动态负载平衡无须对处理机的计算速度作假定,可根据实际测量数据进行调整.

实现动态负载平衡的方法很多,一些学者采用由一个节点监视整个系统的负载状况,并控制整个系统的负载分配的方法,这称为直接方法,它的优点是简单、直观,在一些应用中取得了较好的效果.另一类方法是紧邻方法(nearest-neighbor approach),它假定并行处理系统是由多个节点按照一定的结构相互连接构成的并行处理网络,每个节点只能与其直接连接的节点通信,计算负载也只能通过这些连接移动.在负载平衡过程中,每个节点只能与其周围的节点进行负载交换,通过多次循环迭代实现系统的负载平衡[3].采用这种模式的负载平衡算法有:扩散方法(diffusion method)[4]、维交换法(dimension exchange method)[5]和基于梯度的方法(gradient based method)等.在维交换法中,每个节点在一个时间内只与它的一个相邻的节点进行负载平衡.在超立方体结构中,在每个节点与其所有相邻的节点进行负载平衡后,整个系统达到负载平衡.基于梯度的方法有多种,其共同特点是用负载梯度图来描述整个系统的负载情况,负载是向着梯度最陡方向的节点移动的.其中,梯度模式方法(gradient model method)[6]用计算负载压力面来表示负载的移动.在扩散算法中,每个节点向它周围计算负载较低的节点,在送出计算负载的同时从计算负载高的节点接受计算负载.一些学者还研究了构造切比雪夫多项式来加速收敛的方法[7].但是,上述算法都是针对同构系统的.Chi-Chung Hui[8]提出了针对异构系统的流体负载平衡(hydrodynamic load balancing)方法,该方法用连通器中水的流动来计算负载的移动,还应用水的位能在平衡时最低的原理计算负载的移动.本文将扩散方法扩展到异构系统,重点研究了不同速度的处理机安排在异构系统的不同位置对收敛速度的影响,并利用它来加速扩散算法收敛速度,提出了一种找到收敛速度较快的处理机安排的算法.

1 异构系统的动态负载平衡方法

我们假设并行处理系统是由用某种拓扑结构的网络相连的处理速度不同的节点构成.该系统可以用节点图 $G=(V,E)$来表示,其中 V 是节点,P 是节点数量,$|V|=P$,E 是边,代表节点之间的通信连接.我们将节点从 1~P 顺序编号,使每个节点有惟一的序号与之对应,节点用 n_i 表示.同样,我们将边从 1~$|E|$编号,使每个边也有惟一的序号与之对应,用 e_i 表示,也可用 e_{ij} 表示连接节点 n_i,n_j 的边,$|E|$是边的数量.第 i 个节点的计算速度是 s_i,计算负载是 w_i,计算时间是 $l_i=w_i/s_i$,节点的最缺计算时间是

$$\bar{l}=\frac{\sum_{i=1}^{P}w_i}{\sum_{i=1}^{P}s_i}=\frac{\sum_{i=1}^{P}l_is_i}{\sum_{i=1}^{P}s_i}. \tag{1}$$

负载平衡问题可以归结为通过调整计算时间 w_i,使该节点的时间为 $\bar{l}$,则系统达到负载平衡,节点计算时间变化向量是

$$b=(\bar{l}-l_1,\bar{l}-l_2,\bar{l}-l_3,...,\bar{l}-l_P)^T, \tag{2}$$

其中$()^T$为()的转置矩阵.我们规定,节点间的每一条边都有一个方向,从序号大的处理节点指向序号小的节点.设沿 e_i 边计算负载的转移量是δ_i,计算负载转移向量 $x=(\delta_1,\delta_2,\delta_3,...,\delta_{|E|})^T$,沿边 e_{ij} 的负载转移量也可成δ_{ij},对于节点 i,它的计算负载总变化量是 $\sum_{j\leftrightarrow i}\delta_{ij}$,其中 $j\leftrightarrow i$ 表示节点 j 与节点 i 之间有边直接相连,称为节点 j 与节点 i 相邻.如果整个系统达到负载平衡,对于节点 i 有

$$\frac{\sum_{j\leftrightarrow i}\delta_{ij}}{s_i}=\bar{l}-l_i,\quad i=1,...,P,$$

写成矩阵的形式为

$$S^{-1}Ax=b, \tag{3}$$

其中 S 为对角矩阵, $S=diag(s_1,s_2,s_3,...,s_P)$, A 为图 G 的邻接矩阵,其维数为$|V|\times|E|$,其元素 a_{ij} 为

$$a_{ij}=\begin{cases}1, & i\leftrightarrow j且i是起点\\ -1, & i\leftrightarrow j且i是终点\\ 0, & 其他\end{cases}.$$

因此,负载平衡算法归结为求出每一条边的计算负载转移量,即向量 x,使计算时间相同.

对于给定的系统,已知速度对角矩阵 S 和邻接矩阵 A,向量 b 可以通过测量处理节点的处理时间 l_i 并通过式(1)和式(2)得出,因此,动态负载平衡问题归结于求解代数方程式(3).它有$|E|$个未知变量,有 $P=|V|$个方程,一般地,$|E|>P$,所以式(3)的解不惟一,存在多种方法来求解该方程,各种方法的结果也可以各不相同.

2 异构系统的动态负载平衡扩散算法

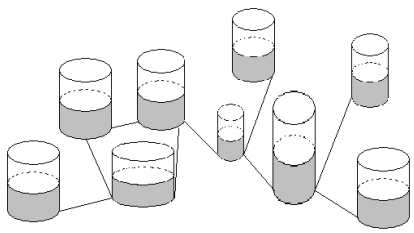

Fig.1 The diffusion algorithm for heterogeneous system

图 1 异构系统中的扩散算法

如图 1 所示,用一组相互连接的连通器建立异构扩散算法的模型.设有 P 个水平截面积不同的容器,在底部有管道相互连接,容器内有水,可以通过连接管在容器之间流动.容器的底面都在同一水平面上.如果在初始时刻液面有高有低,水就会在容器之间流动,直到所有容器的液面高度相同为止.在此模型中,水的体积与计算量相对应,水位与节点的计算时间相对应,容器的横截面积与节点的处理速度相对应,连接管与节点之间的通信线路对应,如果计算时间(水位)不相同,一些计算负载(水)会通过处理节点间的通信连接(管道)从一个节点转移到另一个节点.如果计算时间相等,整个系统达到负载均衡,则计算负载不再移动.

设节点 i,j 相邻,一个时间步内通过 e_{ij} 到达 n_i 的流量正比于节点 i,j 的水位差.设在第 t 次迭代时,各节点的水位(运算时间)是 $l_i^{(t)}$,扩散算法的公式是

$$l_i^{(t+1)}=l_i^{(t)}+\frac{1}{s_i}\sum_{j\leftrightarrow i}\tau_{ij}(l_j^{(t)}-l_i^{(t)}),$$

其中 π_{ij} 是非负的常量.S_t 是节点 i 的横截面积(计算速度).此式表示每一步,节点 i,j 之间交换的计算负载为 $\tau_{ij}(l_j^t-l_i^t)$,引起的计算时间的变化为 $\frac{1}{s_i}\tau_{ij}(l_j^t-l_i^t)$.我们考虑的通信线路是双向的,因此 $\tau_{ji}=\tau_{ij}$.公式可改写为

$$l_i^{t+1}=\left(1-\frac{1}{s_i}\sum_{j\leftrightarrow i}\tau_{ij}\right)l_i^t+\frac{1}{s_i}\sum_{j\leftrightarrow i}\tau_{ij}l_j^t,$$

其中 $j\leftrightarrow i$ 表示所有与节点 i 相邻的节点.如果与节点 i 相邻的节点在初始时刻的计算时间为 0,在一个迭代步内,处理节点 i 上的计算时间不能为负,所以对于常数 π_{ij} 的两个约束条件是

$$\begin{cases}\tau_{ij}>0\\ \left(1-\frac{1}{s_i}\sum_{j}\tau_{ij}\right)\geq 0\end{cases}. \tag{4}$$

令 $l^{(t)}=(l_1^{(t)},l_2^{(t)},...,l_P^{(t)})^T$,扩散公式写为矩阵形式为

$$l^{(t+1)}=Ml^{(t)}, \tag{5}$$

其中, $M=(m_{ij})$ 为 $|V|$ 阶方阵,其元素

$$m_{ij}=\begin{cases}\dfrac{\tau_{ij}}{s_i}, & \text{如果} i\leftrightarrow j\\ 0, & \text{其他}\\ 1-\dfrac{1}{s_i}\sum_k \tau_{ik}, & \text{如果} i=j\end{cases}. \tag{6}$$

显然,对于其中的每一行元素都有 $\sum_j m_{ij}=1$,容易验证 $M=I-S^{-1}AWA^T$,其中 $W=diag(\tau_1,\tau_2,\tau_3,...,\tau_{|E|})$, A 为图 G 的邻接矩阵, $S=diag(s_1,s_2,s_3,...,s_P)$,令

$$y^{(t+1)}=WA^T l^{(t)}, \tag{7}$$

则方程(3)的解为 $x=\sum_{t=1}^{\infty} y^{(t)}$.在实际计算中,只取前面一些项.显然,扩散矩阵 M 不是对称矩阵.

关于扩散矩阵 $M=I-S^{-1}AWA^T$,可以证明在迭代过程式(7)中,总计算量不变,即 $W^{(t)}=\sum_{i\in V} s_i l_i^{(t)}$ 为常量; M 所有特征值均为实数; M 的所有特征值的绝对值小于等于 1;–1 不是 M 的特征值.证明从略.

如果所有权重系数相同 $\tau_{ij}=\tau$,矩阵 M 可以写为 $M=I-\tau S^{-1}M'$, $M'=AA^T$.设 M 的特征值为 η, $S^{-1}M'$ 的特征值为 λ. M 与 $S^{-1}M'$ 特征值的关系为 $\eta=1-\tau\lambda$.因为 $-1<\eta\leq 1$,所以可以得出 $S^{-1}M'$ 的特征值 $\lambda\geq 0$,设它们是 $0=\lambda_1<\lambda_2\leq\lambda_3\ldots\leq\lambda_P$,则 M 的特征值 $\eta_1=1-\tau\lambda_1=1$, $\eta_2=1-\tau\lambda_2,\ldots,\eta_P=1-\tau\lambda_P$,相应特征的向量为 $\alpha_1,\alpha_2,\ldots,\alpha_P$.其中 $\alpha_1=(1,1,\ldots,1)^T$.向量 l_0 可以表示为它们的线性组合

$$l_0=a_1\alpha_1+a_2\alpha_2+\ldots+a_P\alpha_P,$$

$$l^{(t+1)}=Ml^{(t)}=M^{t+1}l_0=a_1\eta_1^{t+1}\alpha_1+a_2\eta_2^{t+1}\alpha_2+\ldots+a_P\eta_P^{t+1}\alpha_P,$$

其中 $a_1\alpha_1$ 是负载平衡项.所以,循环(5)的收敛速度取决于 η_2 和 η_P 中绝对值较大者.由 $\eta_2=1-\tau\lambda_2$, $\eta_P=1-\tau\lambda_P$ 可得,当 $\tau=\dfrac{2}{\eta_2+\eta_P}$ 时, η_2, η_P 的绝对值相等并达到最小 $\dfrac{\lambda_P-\lambda_2}{\lambda_P+\lambda_2}$,若令 $p=\dfrac{\lambda_P}{\lambda_2}$,则收敛速度取决于 $\dfrac{p-1}{p+1}$.所以, p 越接近 1,收敛速度越快, p 越大,收敛速度越慢.

3 提高扩散收敛速度的方法

我们手工随机画出一个连接图,共有 9 个处理节点,如图 1 所示,假定节点的速度分别是 1,2,3,…,9,我们遍历了所有 362 880 种不同的可能,得到 p 的最小值是 4.2,最大值是 146.1,因此研究比如找到 p 达到最小或比较小的节点的安排方式对加快扩散算法的收敛速度是很有意义的.

加快扩散算法的收敛速度的方法是找到 p 值比较小的节点速度安排的方法.它是特征值问题的反问题,要求特征值满足一定的要求,即求矩阵的问题.这类问题一般来讲是比较困难的.最容易的方法是遍历方法,即计算所有可能的安排的 p 值,从中挑选出 p 最小的情况.这种方法只能在节点数量比较少(P<10)的情况下采用,由于需要试验的数量是 P!,计算量随着节点数量增加很快,节点数量在 10 个以上时这种方法基本不可用.为了找出计算量更少、速度更快的方法,本文提出了一种将节点逐次加入连接图的方法,需要实验的数量比遍历方法要少得多,但是该方法不能求出最小的 p 值的节点安排.事实上,我们没有必要求出 p 达到最小值时的最佳安排,只需找出一种 p 比较接近最小值的情况,能够加速扩散的收敛速度即可.具体方法如下:

- 测量节点速度 s_i,将所得到的节点速度除以速度最慢的节点的速度.这样,最慢节点速度为 1,其他节点速度就是与它的相对速度.按照速度由大至小逆序排列.
- 根据图建立矩阵 M'.
- 令速度矩阵 S 为单位矩阵.
- 将所有节点位置置标志 0.
- do,对节点 i 从 1~P 循环

  do,对节点位置 j 从 1~P 循环

  if (位置 j 标志为 0) then

令速度矩阵 S 的第 j 行对角元素为 s_i,计算矩阵 $S^{-1}M'$的特征值λ_2 和λ_n

如果 $p=\lambda_n/\lambda_2<p_{\min}$,令 $k=j$

将速度矩阵 j 行对角元素还原成 1

end if

end do

将位置 k 记标记为 1,令速度矩阵 S 的第 k 个对角元素为 s_i

end do.

显然,本方法需要试验 $\frac{P(P+1)}{2}$ 种情况,比遍历方法的 $P!$要少得多.我们仍然使用第 1 节的例子.在该例中,用本方法得到的 p 值为 15.55,可见,用本方法求出的 p 值是比较小的.

4 数值实验

为了进一步验证提出方法的有效性和可靠性,我们设计了一些实验,对能够遍历的规模较小的系统的所有情况进行统计,给出比较详细的分析;对于规模较大的系统,用蒙特卡洛方法进行分析实验.表 1 是实验的结果.

实验 1.

我们用节点数量比较小的系统进行实验,以便用遍历方法求出最佳方案进行对比.实验方法如下(见表 1):

- 假定系统由 9 个节点组成,每个节点的速度分别为 1,2,3,…,9;
- 用手工方法确定 9 个节点间的连接图,其中 8 个连接图随机确定,另一个是 3×3 的二维格栅网;
- 我们用本文介绍的方法求出它们的 p 值,用 p_1 表示;
- 采用遍历方法,求出 p 的最小值 $p_{\min}$,最大值 $p_{\max}$.

Table 1 Results of experiment 1

表 1 实验 1 的结果

Connection No.	1	2	3	4	5	6	7	8	9
Connection	3	2,4,6	2,7,6	6,2	6,2	2 4 5	2,3	2	2,4
	3	1,3,7	1,3	1,8,3	1,8,3	1 4 3	1,4,6,8	1,3	1,3,5
	1,2,4	2,4,8	2,4	2,5	2,5	2 6 7	1,5,9,7	2,4,9	2,6
	3,5,7	1,3,5	3,9,5	6,5,8	6,5,8	2 1 6 5	2	3	1,5,7
	4,6	4,9	4,6	4,9,3	4,9,3	1 4 6 8	3	6	2,4,6,8
	5	1,9	5,1	4,1,9,7	4,1,9,7	3 4 5 7 8	2	5,7,9	3,5,9
	4,8,9	2,9	1,8	6,8	6,8	3 6 8 9	3	6,8	4,8
	7	3,9	7,9	4,2,7	4,2,7	5 6 7 9	2	7	5,7,9
	7	5,6,7,8	8,4	6,5	6,5	7 8	3	6,3	6,8
P_1	15.3	8.0	13.5	7.9	23.9	7.0	27.1	13.0	9.3
$p_{\min}$	9.2	5.4	7.7	6.6	7.1	5.4	7.5	7.4	7.4
$p_{\max}$	88.2	25.1	33.4	29.1	144.8	38.2	131.9	40.9	32.6
P_r (%)	7.7	13.2	25.6	5.8	12.2	4.9	15.8	16.7	7.5
$N_{p<p1}$	278	472	1831	36	159	90	1720	655	184
$N_{p<p1}$	362 601	362 407	361 049	362 843	362 720	362 790	361 159	362 224	362 694
pct (%)	0.07	0.13	0.5	0.01	0.04	0.02	0.5	0.2	0.05

在表 1 中,第 1 行是连接图的标号,第 2 行表示图的连接情况,第 i 行的数字表示节点 i 与标号为该行数字的节点相连,p_1 是用本方法求出的 p 值,$p_{\max}$,$p_{\min}$ 是用遍历方法找到的最大和最小 p 值,

$$p_r=(p_1-p_{\min})/(p_{\max}-p_{\min}),$$

p_r 越接近 0,表示 p_1 越接近最小值;越接近 1,表示 p_1 越接近最大值,$N_{p<p1}$,$N_{p>p1}$ 是 p 值小于、大于 p_1 的数量,

$$pct=N_{p<p_1}/(N_{p<p_1}+N_{p>p_1}+1),$$

是 p 值小于 p_1 占总数的百分比.可见,用本方法找出的 p 值是比较小的,一般 p_r<30%,比较接近最小值,$pct_{p<p1}$<1%.

实验 2.

主要考察节点速度对本方法效果的影响.用白噪声随机数发生器在区域[1,10]内产生 9 组节点速度,采用与实验 1 相同的连接方式,分别进行实验,结果见表 2 和表 3.其中,表 2 为 9 组节点速度,表 3 为实验结果.

可见,节点速度的不同对结果有一些影响,但是并不严重.影响最大的还是连接图.选择比较好的连接图是

重要的.

Table 2 Speed of the nodes in experiment 2

表 2 实验 2 中节点的连接

Speed No.	1	2	3	4	5	6	7	8	9
Processor speed	8.67	9.13	9.53	8.86	8.11	9.68	8.95	9.93	9.51
	4.08	6.07	4.00	5.05	4.99	8.88	6.57	3.64	6.03
	3.27	5.96	3.51	4.29	4.46	8.64	5.60	3.57	5.92
	3.24	5.53	2.89	3.55	4.14	5.53	5.48	3.26	5.92
	2.97	5.34	2.63	2.54	3.04	3.37	3.52	2.21	4.91
	2.08	3.90	1.56	2.37	2.47	2.51	3.23	1.85	3.53
	1.87	2.83	1.39	2.07	1.68	1.61	2.58	1.73	2.56
	1.27	1.45	1.29	1.14	1.52	1.53	1.05	1.28	2.26
	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table 3 Results of experiment 2

表 3 实验 2 的结果

Speed No.	Connection No.	1	2	3	4	5	6	7	8	9
1	P_r	7.6	8.3	19.2	6.7	11.4	7.1	15.2	17.0	4.5
	Pct (%)	0.22	0.02	0.34	0.07	0.08	0.04	0.23	0.07	0.005
2	P_r	11.6	9.5	21.4	4.5	14.0	7.5	15.8	16.4	6.3
	Pct (%)	0.61	0.12	1.34	0.006	0.51	0.01	0.40	0.20	0.03
3	P_r	8.0	12.5	14.1	5.6	11.2	3.2	13.8	15.8	0.1
	Pct (%)	0.11	0.21	0.14	0.08	0.10	0.02	0.32	0.10	0.0003
4	P_r	5.5	7.5	15.5	5.8	9.6	2.4	13.4	10.1	1.1
	Pct (%)	0.05	0.05	0.11	0.008	0.10	0.008	0.221	0.005	0.002
5	P_r	7.7	9.1	25.2	0.8	11.4	1.6	13.5	14.8	3.6
	Pct (%)	0.26	0.02	2.03	0.002	0.67	0.005	0.30	0.06	0.006
6	P_r	6.6	4.1	19.0	2.4	8.1	4.0	9.1	13.5	4.1
	Pct (%)	0.06	0.09	0.21	0.006	0.42	0.09	0.27	0.11	0.03
7	P_r	8.1	10.8	18.6	3.9	11.9	6.4	13.0	14.3	6.0
	Pct (%)	0.11	0.10	0.24	0.004	0.32	0.01	0.28	0.08	0.02
8	P_r	4.2	9.8	16.5	7.4	11.3	0.4	14.5	11.4	6.0
	Pct (%)	0.03	0.01	0.17	0.01	0.10	0.001	0.14	0.15	0.03
9	P_r	10.1	12.3	18.9	5.7	14.2	7.5	16.0	15.3	13.0
	Pct (%)	0.25	0.20	0.40	0.01	0.49	0.04	0.42	0.10	0.86

实验 3.

为了考察本方法对更大规模系统的有效性,我们采用更大规模的系统进行实验.规模扩大以后,遍历方法的计算量太大,因此我们采用蒙特卡洛方法对 p 的分布情况进行估计.由于二维格栅网是经常采用的一种连接方式,因此我们以它为例进行实验.所采用的方法是:

- 假定网络是 8×8 二维格栅网,对格栅网节点按行标号,算出矩阵 M'.
- 随机数发生器在区间[1,10],产生 64 个随机数,并除以最小数得到节点的运算速度,将所有节点按照速度快慢按逆序排列,最小节点速度为 1.
- 用本文的方法计算出 p_1.
- 用白噪声发生器产生随机整数 m,将序号为 mod(m,P)+1 的节点放在 1 号节点位置,将该节点以后的节点序号依次前移.产生第 2 个随机数 m,将序号为 mod(m,P−1)+1 的节点放在 2 号节点位置,依此类推,得到一个节点的随机排列.
- 计算该排列的 p 值.
- 重复上述过程 100 000 次,统计 p 的最小值、最大值、p 小于 p_1 的次数等.用得到的统计值得出 $p_{\min}$, $p_{\max}$, P_r, $pct_{p<p1}$ 等的近似值.

图 2 是我们得到的结果,其中纵坐标是 $p \in (n-1,n]$ 的次数,横坐标是 n,箭头是 $p_1 = 93.9$ 的位置. p 基本呈现出一种近似正态分布的情况,最大值和最小值的情况比较少,绝大多数情况的 p 值是在其平均值附近. p 的最小值是 63.8,最大值是 225.2,p_r≈18.6%,共有 23 次 $p<p_1$,$pct_{p<p1}$≈0.023%.可见,本实验的结论与前两个是相同的,本方法对规模较大的系统是适用的.

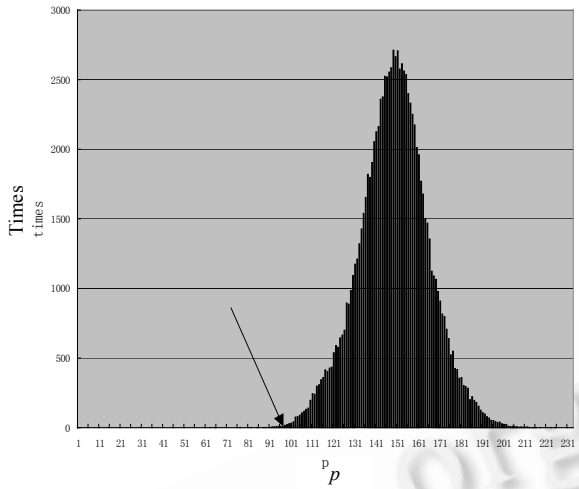

Fig.2 Result of experiment 3

图 2 实验 3 的结果

5 结 论

本文将动态负载平衡的扩散方法扩展到异构系统,并提出了在异构系统中的扩散方法,分析了提高扩散算法的原理,并指出,在给定的节点连接图和节点速度的条件下,将不同节点安排在不同的位置会对扩散方法的计算收敛速度产生很大影响.提出了找到较优的处理机位置安排的算法.该算法能用比较少的次数得到 p 值比较少的节点的处理机安排,从而加快扩散方法的收敛速度.实验证明,该算法的 p_r 一般小于 30%,大多数情况下小于 10%,pct 小于 1%,找到的节点安排的 p 值在所有可能的安排中是较小的.

References:

[1] Casavant TL, G.Kuhl J. A taxonomy of scheduling in general-purpose distributed computing systems. IEEE Transactions on Software Engineering, 1988,14(2):141~154.

[2] Willebeek-LeMair MH, Reeves AP. Strategies for dynamic load balancing on highly parallel computer. IEEE Transactions on Parallel and Distributed System, 1993,4(9):979~993.

[3] Xu CZ, Lau FCM. Iterative dynamic load balancing in multicomputers. Journal of Operational Research Society, 1994,45(7):786~796.

[4] Cybenko G. Dynamic load balancing for distributed memory multiprocessors. Journal of Parallel and Distributed Computing, 1989, 7(2):279~301.

[5] Hosseini SH, Litow B, Malkawa M, McPherson J, Vairavan K. Dynamic load balancing for distributed memory multiprocessors. Journal of Parallel and Distributed Computing, 1989,7(2):279~301.

[6] Lin CH, Keller RM. The gradient model load distribution methods. IEEE Transactions on Software Engineering, 1987,13,(1): 32~38.

[7] Xu CZ, Lau FCM. Optimal parameters for load balancing with the diffusion method in mesh networks. Parallel Processing Letters, 1994,l(4):139~147.

[8] Hui CC, Chanson ST. Hydrodynamic load balancing. IEEE Transactions on Parallel and Distributed System, 1999,10(11):1118~1137.