计算机科学1999Vol. 26№. 12

并行分布计算中随机任务负载的分配*)

Distribution of Random Workloads in Parallel and Distributed Computing

陈华平　石　琴　黄刘生　陈国良

（国家高性能计算中心　合肥 230027）（合肥工业大学　合肥 230009）

Abstract　As one of the hard problems in Parallel Distributed Computing(PDC), task distribution has great influence on the execution efficiency of PDC. In this paper, we first discuss the distribution model of random workloads, which is engaged to make trade-off between the balancing of the execution costs among processors and the reducing of communication and synchronization overheads. Finally we analyze the PDC meaning of the model and discuss the way of assigning these random workloads to processors.

Keywords　Parallel and distributed computing, Task distribution, Model

为了提高并行分布计算的执行效率，必须尽量减少处理结点间的通讯和平衡处理结点间的计算成本，这两个目标看上去是冲突的[1-3]。从减少通讯量角度出发，应把所有任务放在同一处理结点上，但很明显，这样会造成系统资源的极大浪费而且各处理结点计算成本极不平衡；从求得处理结点间的计算成本平衡及提高资源利用率出发，应把任务均匀分配到各处理结点上，但这样有可能大大增加处理结点间的通讯，因此，并行分布系统中任务分配的目标就是寻找一个任务分配方案，使得整个任务负载的执行时间最短。

目前分布系统下的大部分任务分配算法是基于网络流方法，其中最为著名的是 H. Stone[4]于1977年提出的由两个处理结点组成的并行分布系统上的最佳任务分配方法，以及 C. Lee[5]等于1992年提出的基于一维线性网络连接的并行分布系统上的最佳任务分配方法，其它绝大多数是启发式分配方法，其中最为典型的是 m-路网络流算法[6]。本文主要讨论如何通过数学方法，来获得随机任务负载在分布系统下的最佳分配。

1　问题模型

设系统负载由相互作用的 n 个任务 $\{a_1, a_2, \cdots, a_n\}$ 组成，并行分布计算系统由 m 个处理结点 $\{P_1, P_2, \cdots, P_m\}$ 组成。我们用一组随机函数变量来描述并行分布系统特性及任务负载特性。假定任务是随机产生的，用随机变量 t_i 来表示任务 a_i 在结点 P_j 上的执行时间，并且这些随机变量的数学期望值为 $t_i = E\{t_{ij}\}$。e_{ik} 是表示任务 a_i 与任务 a_k 之间是否存在数据交换的二进制随机变量，如果 a_i 与 a_k 之间存在数据交换，那么 $e_{ik}=1$，否则 $e_{ik}=0$，我们假定这随机变量的数学期望值为 $e = E\{e_{ik}\}$。

设 B_i 为分配给处理结点 P_i 的任务块子集，$L(i)$ 为 B_i 中任务块的下标集，即 $L(i)=\{j: a_j \in B_i\}$。用 x_i 表示分配给结点 P_i 的任务块数目，即 $x_i = |B_i| = |L(i)|$，考虑到实际情况下每个处理结点的计算能力会受到限制，并且不同处理结点的计算性能也有很大差异，我们假定每个结点 P_i 对其能运行的最大任务数有一上限 c_i。对于一个任务分配方案 $B=\{B_1, B_2, \cdots, B_m\}$，把 $x=(x_1, x_2, \cdots, x_m)$ 称为负载分配向量，它满足下列约束条件：$\sum x_i = n, 0 \leqslant x_i \leqslant h_i$，其中 $h_i = \min\{c_i, n\}$。我们所要解决的问题就是如何确定负载分配向量 x，使得整个任务负载具有最短的执行时间。

一般说来，在整个任务负载的执行过程中，每个结点 P_i 所处的可能状态有三种：计算状态、通讯状态和同步等待状态，这三个状态的交替反复构成了每个处理结点的运行过程。相应地，每个结点 P_i 执行完成 B_i 中的 x_i 个任务所需的整个执行时间 T_i 也由三部分 T_{i1}, T_{i2}, T_{i3} 构成。

(1)计算时间 T_{i1}，很明显，T_{i1} 为 B_i 中所有任务的计算时间和，即：

$$T_{i1} = \sum_{j \in L(i)} t_j \qquad (1)$$

(2)通讯时间 T_{i2}，如果任务 a_j 和 a_k 被分配在不同结点上，即 $a_j \in B_i, a_k \notin B_i$，那么它们之间通过外部网络进行通讯，我们用随机变量 D_{jk} 来表示这些外部通讯

*)本文受国家863重点项目(863-306-ZD01-02-3)和安徽省自然基金的资助。

开销，它们的数学期望值为 $D_i=E\{D_{ik}\}$。再根据任务 a_i 和 a_k 发生数据变换的随机变量 e_{ik}，T_{i2} 为：

$$T_{i2}=\Sigma_{i\in L(i)}\Sigma_{k\in L(i)}e_{ik}D_{ik} \quad (2)$$

如果任务 a_i 和 a_k 被分配在相同结点上，那么不管它们之间是否存在数据交换，我们均可忽略其内部通讯延迟。

(3)由于同步等待引起的延迟时间 T_{i3}。对于任务 a_i 和 a_k($a_i\in B_i$，$a_k\notin B_i$)的外部数据交换，我们可通过二进制随机变量 q_{ik} 来表示 P_i 是否因该次通讯而引起同步开销，并且假定如果存在延迟，用随机变量 Q_{ik} 来表示这延迟开销。设这两组随机变量的期望值分别为 $E\{q_{ik}\}=q$ 和 $E\{Q_{ik}\}=Q$，那么 T_{i3} 为：

$$T_{i3}=\Sigma_{i\in L(i)}\Sigma_{k\notin L(i)}e_{ik}q_{ik}Q_{ik} \quad (3)$$

在上面(1)(2)(3)式中，t_i，D_{ik}，e_{ik}，q_{ik} 和 Q_{ik} 这些随机变量均是相互独立的。另外，计算、通讯和同步等待这三种状态间的切换会带来一些额外开销，并且任务间的切换也会带来一些开销，我们假定已把这些开销合并到 t_i，D_{ik}，Q_{ik} 中去了。

根据以上讨论可知，处理结点 P_i 执行完成任务子集 B_i 所需的整个时间 T_i 为：

$$T_i=T_{i1}+T_{i2}+T_{i3}=\Sigma_{j\in L(i)}t_k+\Sigma_{j\in L(i)}\Sigma_{k\notin L(i)}e_{ik}D_{ik}+\Sigma_{i\in L(i)}\Sigma_{k\notin L(i)}e_{ik}q_{ik}Q_{ik} \quad (4)$$

对等式两边同取数学期望值：

$$T_i=E(T_i)=\Sigma_{i\in L(i)}E(t_k)+\Sigma_{j\in L(i)}\Sigma_{k\notin L(i)}E(e_{ik}D_{ik})+\Sigma_{j\in L(i)}\Sigma_{k\notin L(i)}E(e_{ik}q_{ik}Q_{ik})$$

进一步化简可得：

$$T_i=t_i\Sigma_{j\in L(i)}1+eD_i\Sigma_{j\in L(i)}\Sigma_{k\notin L(i)}1+eqQ\Sigma_{j\in L(i)}\Sigma_{k\notin L(i)}1$$

根据 $|L(i)|=x_i$，可把 T_i 定义成 x_i 的函数，上式化为：

$$T_i(x_i)=t_ix_i+eD_ix_i(n-x_i)+eqQx_i(n-x_i)$$

进一步化简，上式可化为：

$$T_i(x_i)=[-eD_i-eqQ]x_i^2-[-t_i-enD_i-eqQn]x_i \quad (5)$$

令 $$a_i=-eD_i-eqQ \quad (6)$$

$$b_i=-t_i-enD_i-enqQ \quad (7)$$

那么(5)式可化为：

$$T_i(x_i)=a_ix_i^2-b_ix_i \quad (8)$$

假定每个处理结点同时开始执行，那么最后执行完成的处理结点所需的执行时间即为整个系统任务负载的执行时间 $T(x)$，也就是说，

$$T(x)=MAX_i\{a_ix_i^2-b_ix_i\}=MAX_i\{T_i(x_i)\} \quad (9)$$

最佳任务分配问题就是如何确定一个负载分配向量 $x^*=(x_1^*,x_2^*,\cdots,x_m^*)$，使得 $T(x)$ 具有最小值，设最小值为 T^*，那么

$$T^*=T(x^*)=MIN_xT(x)=MIN_xMAX_i\{T_i(x_i)\}=MIN_xMAX_i\{a_ix_i^2-b_ix_i\} \quad (10)$$

其中 x_i 为整数，并且 $\Sigma x_i=n$，$0\leqslant x_i\leqslant h_i\leqslant n$。

2 问题模型的并行分布计算意义

下面对函数 $T_i(x_i)=a_ix_i^2-b_ix_i$($x_i\in[0,h_i]$)进行分析，其中设 $y_i=T_i(h_i)=a_ih_i^2-b_ih_i$。对 $T_i(x_i)$ 分别求一阶导数和二阶导数可得：$T_i'(x_i)=2a_ix_i-b_i$，$T_i''(x_i)=2a_i$。很明显，当 $x_i<b_i/2a_i$ 时，$T_i'(x_i)<0$，$T_i(X_i)$ 是单调上升的，$x_i>b_i/2a_i$ 时，$T_i'(x_i)<0$，$T_i(x_i)$ 是单调下降的。所以我们必须根据 h_i 的大小分别讨论。

(Ⅰ)($h_i<b_i/2a_i$)：这时 $x_i\leqslant h_i<b_i/2a_i$，$T_i'(x_i)>0$，$T_i(x_i)$ 在$[0,h_i]$上恒单调上升。

(Ⅱ)($h_i>b_i/2a_i$)：这时当 $x_i\in[0,b_i/2a_i)$ 时，$T_i'(x_i)>0$，$T_i(x_i)$ 是单调上升的，当 $x_i\in(b_i/2a_i,h_i]$ 时，$T_i'(x_i)<0$，$T_i(x_i)$ 是单调下降的，所以这时 $T_i(x_i)$ 在$[0,h_i]$上是非单调的。

$T_i(x_i)=a_ix_i^2-b_ix_i=0$ 有两个根，即 $x_{i1}=0$，$x_{i2}=b_i/a_i$，虽然情形(Ⅱ)中 h_i 可能大于 $b_i/2a_i$，但 $h_i<b_i/a_i$，这是因为根据(6)(7)式，$b_i/a_i=t_i/(eD_i+eqQ)+n>n>h_i$。

从并行分布计算意义上我们也可以这样理解，如果给 P_i 上多分配些任务，那么由此可以减少任务间的外部通讯及同步开销，从减少整个通讯开销和同步开销的角度出发，这样是有利的，但由于新增任务本身计算时间的加入，使得 $T_i(x_i)$ 一开始还是随任务数目的增加而增长，但 $T_i'(x_i)=2a_i<0$，也就是说 $T_i(x_i)$ 的增加速度逐步减小，即 $T_i(x_i)$ 随 x_i 的增加而呈亚线性增长。如果继续不断增加 P_i 上的任务数目，那么由此减少的外部通讯及同步开销所获得的收益，逐渐逼近或超过由此带来的新增计算时间，任务数 $x_i=b_i/2a_i$ 就是它们之间的平衡点。如果 $h_i<b_i/2a_i$，说明由于处理结点 P_i 受到计算能力的限制，其上所分配的任务数目到达不了这个平衡点；如果 $h_i>b_i/2a_i$，那么 P_i 上的任务数目超过 $b_i/2a_i$ 时，由于外部通讯及同步开销的大大减小，使得 $T_i(x_i)$ 反而随任务数的增加而减小。

3 负载分配向量的确定

下面我们先按 x_i 是连续变量这个假定求解，然后在此基础上求整数解，根据以上讨论，我们可把所有的 $T_i(x_i)(1\leqslant i\leqslant m)$ 组成的函数集合 R 分为两部分，一部分为单调递增函数，记为 R_1，对应于上一节的第(Ⅰ)种情形；另一部分为非单调递增函数，记为 R_2($R_2=R-R_1$)，对应于上一节的第(Ⅱ)种情形。如果 R_2 非空，那么设 u 是 R_2 中使 $T_i(h_i)$ 取得最小值的下标，即定义：

$$y_u=T_u(h_u)=\min_{T_i(x_i)\in R_2}\{T_i(h_i)\}=\min_{T_i(x_i)\in P_2}\{y_i\} \quad (11)$$

如果 R_2 中取得最小值 y_u 的函数多于一个，那么任取其中一个即可。如果 $R_2-\{T_u(x_u)\}$ 非空，那么设 v

是$R_u-\{T_u(x_u)\}$中使$T_i(h_i)$取得最小值的下标，即定义：

$$y=T_v(h)=MIN_{T_i(x_i)\in R_2-T_u(x_u)}\{T_i(h_i)\}$$
$$=MIN_{T_j(x_j)\in R_2-T_u(x_u)}\{y_j\} \quad (12)$$

同样，如果$R_2-\{T_u(x_u)\}$中取得最小值y_i的函数多于一个，那么任取其中一个即可。这样，y_u和y_v是非单调递增函数集R_2中最小的两个值，并且$y_u\leqslant y_v$。

给R中的每个$T_i(x_i)$定义如下反函数：

$$x_i=G_i(y)=\begin{cases}T_i^{-1}(y) & 0\leqslant y\leqslant y_i\\ h_i & y>y_i\end{cases} \quad (13)$$

根据上一节对二次曲线$T_i(x_i)$的系数a_i和b_i的分析可知，R_1中的$T_i(x_i)$在区间$y\in[0,y_i]$上是一一对应的，所以它们都存在严格定义的反函数$T_i^{-1}(y)=(b_i+\sqrt{b_i^2+4a_iy})/2a_i$。对于$R_2$中的$T_i(x_i)$，在点$y=y_i$上，$T_i^{-1}(y_i)$存在两个值，一个是$x_{i1}=h_i$，另一个是$a_ix_i^2-b_ix_i=y_i$的小根，由于$a_i<0$，$b_i<0$，所以小根值为$x_{i2}=(b_i+\sqrt{b_i^2+4a_iy_i})/2a_i$。如果我们规定$x_{i2}$作为$T_i^{-1}(y_i)$的唯一值，那么这时，$T_i^{-1}(y)$在$y\in[0,y_i]$上也是严格定义的。

很明显，对于R_1中的函数，上面定义的$T_i^{-1}(y)$（即$G_i(y)$）是连续的。而对于R_2中的函数$G_i(y)$在$y=y_i$处有一跳跃，即：$G_i(y_i)=(b_i+\sqrt{b_i^2+4a_iy_i})/2a_i$，$G_i(y_i^+)=h_i$。

我们定义$S(y)=\Sigma G_i(y)(y\geqslant 0)$，由于$G_i(y)$是y的非减函数，所以S(y)也是y的非减函数。对于R_2中函数$T_i(x_i)$的反函数$G_i(y)$，S(y)在其对应每个y_i处出现一个跳跃性的不连续，它们对应了$G_i(y)$函数的不连续，另外，在两个不连续点的区间内函数S(y)是连续且单调递增的，具有严格定义的反函数$S^{-1}(*)$，其中$S^{-1}(*)$的自变量为构成系统负载的任务总数。

如果R_2至少包含两个函数，那么$S(y_v^+)=\Sigma G_i(y_v^+)=G_v(y_v^+)+G_u(y_v^+)+\Sigma_{i\neq u,i\neq v}G_i(y_v^+)$，由于$y_v^+\geqslant y_v\geqslant y_u^+$，根据(13)的定义，$G_u(y_v^+)=h_u$，从而上式可化为：$S(y_v^+)=h_v+h_u+\Sigma_{i\neq u,i\neq v}G_i(y_v^+)$。由于$h_u>(b_u/2a_u)$，$h_v>(b_v/2a_v)$，所以$S(y_v^+)>(b_u/2a_u)+(b_v/2a_v)$，根据上一节说明，$b_i/a_i>n$，因此$S(y_v^+)>n$。这样，在$R_2$非空的一般情况下，S(y)最多只可能包含$y_u$和$y_v$处两个断点，函数S(y)在区间$[0,y_u]$及$(y_u^+,y_v]$上连续且单调递增，相应地$S^{-1}(n)$在$[0,S(y_u)]$及$(S(y_u^+),S(y_v)]$上连续且单调递增，并且$S^{-1}(n)$与任务总数n构成了一一对应关系。而在$(S(y_u),S(y_u^+)]$及$(S(y_v),S(y_v^+)]$这两个区间上，$S^{-1}(n)$与任务总数n不满足一一对应条件。

如果点$(n,S^{-1}(n))$，落在$(S(y_u),S(y_u^+)]$或者$(S(y_v),S(y_v^+)]$非一一对应区间上时，可分别令$x_u^*=h_u$或者$x_v^*=h_v$。对于其它的x_i^*，直接选择满足$(T_i(x_i^*)\leqslant y_u)$或者$(T_i(x_i^*)\leqslant y_v)$，并且$\Sigma x_i^*=n$的整数解即可，这时T(x)取得最小值，其值为$T^*=y_u$或$T^*=y_v$。如果点$(n,S^{-1}(n))$落在$[0,S(y_u)]$或$(S(y_u^+),S(y_v)]$一一对应区间上，或者$R_2$为空（这时S(y)均由$R_1$中的单调函数构成），S(y)和$S^{-1}(n)$在任何点上均是一一对应并且单调递增的，这时$T^*=S^{-1}(n)$是其最小值。令$x_i^*=G_i(T^*)=G_i(S^{-1}(n))(1\leqslant i\leqslant m)$，这时T(x)取得最小值，其最小值为$(T^*)=S^{-1}(n)$，但$x_i^*$并不一定是整数，可对非整数解$x^*=(x_1^*,x_2^*,\cdots,x_m^*)$进一步调整，使其每个元素$x_i^*$均为整数。

设调整后的整数解为$r=\{r_1,r_2,\cdots,r_m\}$。一开始r_i取x_i^*的整数部分，因此$r_i\leqslant x_i^*$，结果$\Sigma_i r_i\leqslant\Sigma_i x_i^*=n$，并且$\Sigma_i r_i$一般都要比n小，除非初解$x_i^*(1\leqslant i\leqslant m)$刚好均为整数。然后每次选择r中的一个元素$r_z$进行加1操作，选择的元素满足条件$T_z(r_z+1)\leqslant\forall_i(i\neq z)(T_i(r_i+1))$，经过$n-\Sigma_{i=1}^m r_i$次加1操作后，$\Sigma_{i=1}^m r_i=n$，下面是具体的调整算法。

```
procedure integer_adjust(x*,r)
  begin
    for(1≤i≤m) do r_i=round(x_i*);endfor
    k=n-Σ_{i=1}^m r_i;
    for (1≤j≤k)do
            选择满足 T_z(r_z+1)≤∀_i(i≠z)(T_i(r_i+
            1))的r向量下标z
            r_z=r_z+1;
            endfor
          end
```

参考文献

1 Hesham El-Rewini, et al. Task scheduling, PTR Prentice Hall, Englewood Cliffs, New Jersey, 1994, 07632

2 陈华平，计永昶，陈国良. 分布式动态负载平衡调度的一个通用模型. 软件学报，1998, 9(1): 25～29

3 Feitelson D G, Rudolph L. Toward Convergence in job schedulers for parallel super-computers. In: IPPS '96 Workshop on Job Scheduling Strategies for Parallel Processing. Hawaii, 1996. 1～9

4 Stone H. Multiprocessor scheduling with the aid of network flow algorithms. IEEE Trans. Software Eng., 1977

5 Lee C, et al. Optimal task assignment in linear array networks. IEEE Trans. on Computers

6 Lo V. Heuristic algorithms for task assignment in distributed systems. IEEE Trans. on Computers, 1988(11)