计算机科学1998Vol. 25 №5

动态负载平衡问题的一种分类与扩散算法*)

A Classification of Dynamic Load Balancing and Algorithm Based on Diffusion

秦忠国　姜弘道

(河海大学工程力学系　南京210024)

摘　要　This paper gives a classification of dynamic load balancing problems based on application characteristics. For the applications of the numerical analysing, such as adaptive FEM, a useful methods called diffusion method is described in more detail.

关键词　Parallel Computing, Dynamic Load Balancing, Diffusion Method

1　引言

在并行计算中，任务划分与处理机分配带来的负载平衡问题是并行计算的一个基本问题，直接影响到并行效率和加速比。一般来讲，负载平衡就是要尽量均匀地分配任务，并尽量减少节点机之间的通信；然而根据并行机体系结构与并行应用程序的不同，解决问题的方法也相应地不同。静态问题一般是对计算量与通信量进行先验估计，在此基础上采用一定的启发算法来划分任务与分配处理机；但有些问题事先无法预测系统的开销，需要在程序运行过程中动态地进行调度。总体上，负载平衡问题分为静态与动态两大类，静态问题只要进行一次任务划分与处理机分配，并一直保持这个划分与分配到程序运算结束。而动态问题根据应用程序不同的情况将可以分为好几个类型。

并行软件的开发需要满足可伸缩(Scability)的要求，而共享存储的体系结构从物理上讲是不可伸缩的，所以分布存储的并行体系结构受到越来越多的重视和应用。近年来发展较快的网络机群也可看成松耦合的分布存储并行机，在这类机器上通过网络采用消息传递结构来进行处理机之间的数据交换，与共享存储结构相比，旨在减少数据通信量的负载平衡算法显得更为重要，同时负载平衡算法本身不能增加过多的开销，并且本身也应是可并行且可伸缩的。

本文以分布存储并行机为背景，根据并行应用程序在进程迁移和数据依赖方面的不同，先对动态负载平衡问题进行分类。在此基础上引入一种称为"基于扩散的平衡算法[5]"，这是一个建立在静态平衡状态基础上的、可高度并行且可伸缩的迭代算法。

2　负载平衡问题分类

用 DMM(Distributed Memory Model)模型来描述一个并行机系统：用平面图上的一个点集 $U=\{u_1,\cdots,u_p\}$ 表示 p 个处理机，用这些点的可能的连线 $F=\{f_1,\cdots,f_k\}\subseteq U\times U$ 表示节点间的通信线路。仅考虑同构网络，即节点机的参数是相同的，假定处理机之间的消息传递能力与方向无关，因而可以略去节点的权重，也不必考虑边的方向性，这样，分布存储的并行机可以用无向图 $H=(U,F)$ 表示。

一个并行应用程序用赋权图 $G=(V,E,\rho,\sigma)$ 来描述，其中节点 $V=\{v_1,\cdots,v_n\}$，边 $E=\{e_1,\cdots,e_m\}\subseteq V\times V$，节点权重 $\rho:V\to R$，边的权重 $\sigma:E\to R$。即用某种方式对 V、E 进行量化，节点与边的具体含义可由并行应用的具体情况而定，对于本文讨论的数值分析问题，节点可代表并行程序的各进程的数据处理(如分裂子区域的单元数量)，边代表数据通信。当用赋权图描述一个并行应用程序时，其拓扑可由图的邻接矩阵 A(或关联矩阵 A_c)表示[6]。

负载平衡问题可以看成是一个图的嵌入问题。任务是找到一个映射 $\pi:G\to H$，并使某些判据得到

*) 中国博士后基金，水利科研基金课题资助。秦忠国　博士后，主要研究领域为结构分析、并行数值计算。姜弘道　教授，博士生导师。校长，研究领域为计算力学、复杂结构分析方法等。

极小化满足。根据并行应用的不同性质，图 G 可能是静态的或动态的，也就是说，程序生成的并行任务在运行期间所做的计算量是否变化；而图 H 对应于并行机结构，被认为是不变的。

由于静态的应用问题只需将并行应用图 G 向图 H 做一次映射，保持这个映射一直到程序运行结束。这样，静态问题的负载平衡问题退化成一个古典的映射问题，这个映射问题对应两个图的嵌入问题。对于绝大部分科学与工程计算问题，常常可以看成静态问题来研究，例如，非自适应的有限元计算。

对于动态问题，并行应用图在运行过程中是变化的，它的节点或边可能会增加或减少。也有可能节点或边的权重会改变。负载平衡问题可以根据应用图的权重函数来进行分类，看节点（进程）和边（通信）哪一个是程序的主要方面。此外，并行应用程序的粒度对负载平衡的分类也很重要。细粒度并行包含大量的轻度进程与较少数据量，而粗粒度并行需要的进程数较少，但每个进程都有较重的数据处理任务。进程的轻重之分可以决定是否有进程迁移。将重进程在程序运行中从一个处理机迁移到另一个处理机一般是较为困难的，相对来讲，数据项的传输却较为容易。根据并行应用程序的通信要求与进程迁移的可能性，可以把动态问题的负载平衡问题分成四种情况，如图1所示。

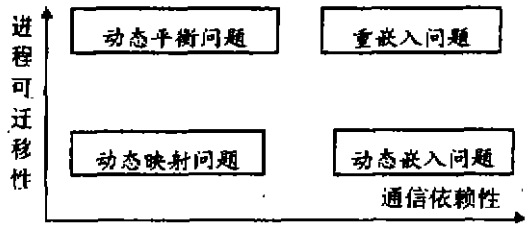

图1　动态平衡问题的一种分类

(1)动态映射问题。进程在某个处理机上动态地创建，一旦创建，必须在某个处理机上一直运行到最后，与其它进程的通信量可以忽略。在这种情况下，负载平衡问题必须对平衡的总开销（例如由于考虑负载平衡必须首先收集系统状态信息）与负载平衡质量作出权衡。常见于某些客户/服务器应用。

(2)动态嵌入问题。进程间需要较多的通信并且进程不可以迁移，可见于一些树搜索问题。

(3)动态平衡问题。面对的是一些可迁移，但无通信的进程，目前对这一类型的问题研究较多，有许多有效的负载平衡算法。

(4)重嵌入问题。进程可迁移，进程之间需要通信，一个典型的例子就是数值计算中的自适应问题，经过细化后的单元可以在处理机上重新分配，各处理机在求解中具有较强数据依赖性。

3　动态负载平衡的算法

本文只考虑重嵌入问题。假设应用程序的运行具有分阶段变化的特性，这样，图 $G=(V,E,\beta,\sigma)$ 的变化并不是随机的，在数值计算领域，这个假设是符合实际的，例如，自适应有限元方法，网格的划分常常是建立在后验误差估计的基础上，程序的下一步运行要等本阶段程序运行结束后才开始，在各阶段中间没有新的负载产生或减少，负载变化的次数则是由应用本身所决定的。

程序运行中负载的多次变化将引起多次的负载平衡问题，程序开始运行的时候，可按静态负载平衡方法进行任务划分与处理机分配，当程序进入下一阶段时，这种平衡状态将遭到破坏，动态负载平衡的任务是及时对此作出调整，原则上可以把这类问题当成一系列的静态问题来解决，即多次应用静态负载平衡算法，但当问题的规模较大时，这需要很多计算时间，假如使用常用的谱方法，每次都要计算 G 的拉普拉斯矩阵的特征值[2]。为此可以从另一角度考虑问题：当出现负载不平衡时，不是整体上重新分配负载，而是在局部对负载进行调整。考虑分两步来做，第一步计算"有多少"负载必须在处理机间进行转移，以及往"哪里"转移；第二步决定转移"哪些"负载。

在进行负载转移时，为不改变程序的通信结构和增加额外的通信开销，不应改变处理机之间原有的数据依赖关系，即必须考虑数据的位置，要保证 G 中相邻的节点不转移到不相邻的处理机上去。

为了表现处理机之间的这种数据依赖关系，可以计算图 G 对应于映射 π 的商图 Q，如果把 G 看成一个拓扑，R 是 G 上对应于 π 的一个等价关系（即把同一个处理机上的节点看成同一），商图 Q 表示的是 G 的商拓扑空间 $Q=G/R$，它的节点对应于处理机，边指明了程序需要的通信要求（图2）。如果在商图 Q 上考虑负载平衡问题，则显然程序数据的连续性是能得到保证的。

解决负载平衡的第一步是"多少负载"的问题，这个问题可以看成是一个"流"的问题，为了达到负载平衡，必须计算出通过 H 的每一条边转移的负载，虽然对这一问题，可以用一些图论的网络流的算

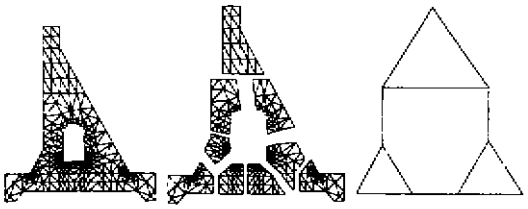

图2 求图 G 的商 Q

法予以解决，但通常这些算法适应于较为复杂的问题，并且非常费时，也不易并行化。这里讨论一种较为简单易行，易于并行化的算法，这个算法甚至不需要提供整体控制，它的每一步运行只是相邻的少数几个处理器之间的相互协调。

这个方法称为"基于扩散的负载平衡方法[5]"，其原理为：每个处理器与它周围的其它数个处理器平均分配所承担的负载，并且重复这一过程，直至达到整个系统的负载平衡，设第 t 步迭代时，p 处理机的负载为 w_p^t，则

$$w_p^{t+1}=w_p^t-\sum_{q\in N(P)}\frac{1}{a_{p,q}}(w_p^t-w_q^t)$$

其中 $N(p)$ 是与 p 处理机相邻的处理机的集合，$1/a_{p,q}$ 为平衡因子，可取 $a_{p,q}=deg_q$，deg_q 为节点 q 的度（相邻的处理机数量），上式表明：在第 t 步迭代中，数量为 $(w_p^t-w_q^t)/a_{p,q}$ 的负载通过 Q 的边 $\{p,q\}$ 进行传递，上式也可以用矩阵表示，设 $w^t=(w_p^t)$ 为负载向量，$M=(m_{p,q})$ 为扩散矩阵，其中 $m_{p,q}=1/a_{p,q}$，$\forall\{p,q\}\in Q$，且 $m_{p,p}=1-\sum_{q\neq p}m_{p,q}$，则

$$w^{t+1}=Mw^t$$

由于 M 满足 $|m_{r,q}|<1$，且有 $\sum_p|m_{p,q}|\leq 1$，所以这一迭代过程是收敛的，最终可以保证 $w^{t+1}\to w^t$。为加速迭代的过程，还可以使用超松弛迭代法，这可以对通过每个边传输的负载数量进行监控，对于第 t+1步迭代，不仅仅取决于当前的负载不平衡状态，而且取决于前一次迭代（第 t-1步）时的情况，迭代式为：

$$\begin{cases}w^1=Mw^0\\ w^{t+1}=\beta Mw^t+(1-\beta)w^{t-1}\end{cases}$$

设 $l_{p,q}^t$ 为第 t 次迭代过程通过 $\{p,q\}$ 传输的负载，则

$$l_{p,q}^{t+1}=\begin{cases}m_{p,q}(w_p^t-w_q^t) & (t=0)\\ (\beta-1)l_{p,q}^t+\beta m_{p,q}(w_p^t-w_q^t) & (t>0)\end{cases}$$

设 ε 为负载平衡需要的精度，可以证明[1]，如果将 β 置于 $2/(1+\sqrt{1-r^2})$，则需要迭代次数 $T_\varepsilon\sim O\left(\log(\frac{1}{\varepsilon})/\sqrt{1-r^2}\right)$，其中 r 为 M 的第二大特征值。

讨论 数值分析中的负载平衡问题是由计算区域上数据密度的变化而引起的，动态平衡算法的任务是要调整系统的平衡状态，其主要问题并不是计算每个处理机应该承担多少负载，而是要计算处理机怎样传递负载才能达到较好的平衡状态。为此，先求出处理机的数据依赖关系，这是通过计算商图 Q 得到的，再计算平衡负载时 Q 的每个边上传输负载的流量，这可以通过扩散算法来进行。

扩散算法解决"转移多少"和"往哪里"转移的问题，但转移"哪些"负载的问题，目前尚研究得较少。判别标准仍然应该是程序的通信量，这与静态问题的判别标准是一致的，一些基于先验估计的技术常常使用一些几何上的判据来决定迁移的对象，如有限元计算的区域分裂问题，选择能使子区域的边界为极小的单元做为平衡负载，但目前对这些类似形状优化的问题仅仅有一些较为简单的启发式算法可用[3,4]。这方面有大量的问题要研究，以便使这些常用的数值方法的负载平衡问题能得到实际而有效的解决。

参 考 文 献

[1] B. Ghosh et al., First and Second Order Diffusive Methods for Rapid, Coarse, Distributed Load Balancing. 8th ACM SPAA, 1996

[2] B. Hendrickson, R. Leland, An Improved Spectral Graph Partitioning Algorithm for Mapping Parallel Computations, SIAM J. Scientific Computing, 16(2) 1995

[3] R. Diekmann et al., Parallel Decomposition of Unstructured FEM-Meshes, IRREGULAR, Spinger LNCS980, 1995

[4] N. Chrisochoides et al., Automatic Load Balanced Partitioning Strategies for PDE Computations, Proc. ACM Int. Conf. on Supercomputing, 1989

[5] G. Cybenko, Load Balancing for Distributed Memory Multiprocessors, J. of Parallel and Distributed Computing, (7), 1989

[6] 秦忠国、姜弘道，静态负载平衡问题的图论表示与算法，计算机科学. №2, 1998