计算机科学 1999Vol.26№.1

多机系统的动态负载平衡

Dynamic Load Balancing in A Multicomputer System

刘振英 张 毅 胡铭曾 方滨兴

（哈尔滨工业大学计算机科学与工程系 哈尔滨 150001）

Abstract In a multicomputer system, a number of computers are connected by network. Balancing the workloads of these computers can improve the performance drastically. As to dynamic load balancing, this paper summarizes its algorithms, discusses its mechanism, and lists some important research directions.

Keywords Load balancing, Load sharing, Dynamic load balancing, Process migrating

在计算机硬件价格下降、计算机网络拓扑发展的情况下，分布式计算机系统给用户提供了一个丰富的资源集合。人们在研究分布式系统时，就注意到了这样一个问题：在一个由网络所连接起来的多计算机环境中，在某一时刻，一些计算机的负载极重，而另外一些计算机的负载极为空闲。平衡各计算机之间的负载是任务分配与调度的一个主要目标[1]，它能够提高整个系统的性能。

一、动态负载平衡算法

为了改善系统的性能，通过在多台计算机之间重新分配负载，把当前重载计算机的任务传送到轻载的计算机上执行，这种计算能力共享的形式，通常被称为负载平衡或负载共享。一般来说，"负载平衡"要达到的目标是使各台计算机之间的负载基本均衡，而"负载共享"意味着只是简单的负载的重新分配。

只是利用系统负载的平均信息，而忽视系统当前的负载状况的方法被称为静态负载平衡。根据系统当前的负载状况来调整任务划分的方法被称为动态负载平衡，或适应性负载平衡。以往的研究表明、动态策略比静态策略对系统性能的改进更大。因此本文着重研究动态负载平衡。下文中出现的负载平衡都指的是动态负载平衡。

目前，已经被普遍接受的负载平衡算法有两个广义分类：一是 Source-initiative 算法：由任务到达的计算机最先传送任务。二是 Server-initiative 算法：由能够和愿意接受被传送的任务的计算机，去寻找这样的任务。

一个负载平衡算法都包含以下三个组成部分：

a）信息策略：指定任务放置策略的制定者使用的负载和任务量，以及信息分配的方式。

b）传送策略：基于任务和计算机负载，判断是否要把一个任务传送到其它计算机上处理。

c）放置策略（Place policy）：对于适合传送到其它计算机处理的任务，选择任务将被传送的目的计算机。

负载平衡的上述三个部分之间是以不同的方式相互作用的。放置策略利用信息策略提供的负载信息，仅当任务被传送策略判断为适于传送之后才行动。

负载平衡的目标是：提供最短的平均任务响应时间；导致低负荷，因为负荷过大对改进工作负载不利；能适于变化的负载；是可靠的，应避免所有计算机花费它们所有的时间来传送任务。

二、动态负载平衡的机制

在这一部分，我们将主要研究负载平衡算法的三个组成部分——信息策略、传送策略和放置策略各自采用什么样的方法实现。

2.1 信息策略

在九十年代初的研究中，Kunz[2]用来描述负载

刘振英 博士生，研究方向：并行与分布计算，并行编译。张毅 学士。胡铭曾 博士导师。方滨兴 博士导师。

信息所采用的参数是：1)运行队列中的任务数；2)系统调用的速率；3)CPU 上下文切换率；4)空闲 CPU 时间百分比；5)空闲存储器的大小(K 字节)；6)1 分钟内的平均负载。对于这些单个的负载描述参数，第1)个，即采用运行队列中的任务数作为描述负载的参数被证明是最有效的，即它的平均任务响应时间最短，并且已经得到广泛应用。

针对不同的应用问题，人们还采用一些其它的负载参数，如文[3]采用了四大类模型化参数，分别是处理器参数、程序参数、网络参数和外部负载的参数。但是，如果为了使系统信息更全面而采集了更多的参数，则往往由于增加了额外开销，却得不到所希望的性能改善。例如，采用将前文中的六个参数中的某两个进行"AND"和"OR"组合，得到的平均响应时间反而比单个参数的平均响应时间还要差一些。

2.2 传送策略

为了简单起见，在选用传送策略时，多选用阈值策略。例如，Eager 等人[4]的方法是：在判断是否要在本地处理一个任务时，无须交换计算机之间的状态信息，一旦服务队列或等待服务队列的长度大于阈值时，就传送这个任务，而且传送的是刚刚接受的任务。本文后面提到的进程迁移，能够迁移一个正在执行的任务，是对这种只能传送刚刚接受的任务的一种改进。

Zhou[5]在模拟研究七个负载平衡算法时，其传送策略都采用阈值策略。他的阈值策略基于两个阈值：计算机的负载阈值 T1 和任务执行时间阈值 Tcpu。如果计算机的负载超过 T1 并且任务的执行时间超过 Tcpu 时，就把此任务传送到其它计算机执行。需要说明的是，虽然任务的执行时间难以预测，但是可以简单地通过察看任务的名字来成功地估计，例如，一个文字处理任务将几乎肯定地超过1秒 CPU 时间，而且目录察看任务的 Tcpu 太小，确实不值得负载平衡考虑。

2.3 放置策略

经过总结，共有以下八种放置策略。

A)全局策略。每隔 P 秒，其中一个计算机被指定为"负载信息中心"(LIC)，接受所有其它负载的变更值，并把它们汇集到一个"负载向量"中，然后把负载向量广播给所有其它的计算机。如果一台计算机的负载和最后要发送时刻的负载相同，就无须给 LIC 发送负载变更值。

B)局部策略。Zaki 等人的研究中，是把多台计算机分成大小为 K 的不同组。局部策略与全局策略的区别是，局部策略仅在组内交换信息。

C)分布策略。不象全局策略那样，把局部的信息都报告给一个 LIC。而分布策略是每台计算机定期地把它的负载信息广播给其他计算机，去更新那些局部维护的负载向量。

D)集中策略。LIC 除了定期从其他计算机接受信息外，还作为所有计算机的一个中心调度器。当一台计算机认为一个任务适于传送到其它计算机上执行时，它就给 LIC 发送一个请求，并告知当前负载的值。LIC 选一台具有最短队列长度的计算机，并且通知任务所在的计算机把任务发送给它。同时，它把目的主机负载值增加 1。

E)随机策略。当外来的任务到来之后，如果任务队列≤阈值，则接受；否则，再随机传送至其它计算机。此策略可能会产生颠簸。解决颠簸的简单方法是：用一个静态传送极限 Lt，来限制任务传送的次数。不管任务的第 Lt 次传送的目的计算机如何，该计算机必须处理该任务。加上 Lt 之后，会极大地改进系统的响应时间。

F)阈值策略。随机选择一台计算机，并探测以判断：若把任务传送到那台计算机后，判断那台计算机的任务队列长度是否会超过阈值。如果不超过阈值，就传送此任务；否则，随机选择另一台计算机，并以同样方式探测，继续这样作直到找到一台合适的目的计算机，或探测次数超过一个静态探针限制 Lp，当任务真正到达计算机后，不管状态如何，必须处理该任务。

G)最短时间策略。随机选择 Lp 台不同的计算机，察看每台计算机的任务队列长度，任务被传送到具有最短任务队列长度的计算机。当任务真正到达计算机、无论状态如何，目的计算机必须处理该任务。对此策略的一个简单改进是，无论何时，遇到一台队列长度为 0 的计算机时，不再继续探测，因为可以确定此计算机是一台可以接受的目的计算机。

Zhou 的论文中，比较了 E，F 和 G 三种策略，结论是：以简单(计算不昂贵)的方式，利用少量状态信息，会取得较好的效果。但是，复杂的方法，必须用性能的改善来补偿额外花费，取得的效果会稍差一些。本文将它整理成下表所示。

表 1 三种放置策略的比较

	随机策略	阈值策略	最短时间策略
利用系统信息	无	较少	完全
性能改进	很大	比随机策略大	和阈值策略类似
采用的分析模型	最简单	简单	复杂

H)保留策略。上述放置策略都是基于 Source-initiative 算法，此保留策略基于 Server-initiative 算法。当一个任务从一台计算机离开时，该计算机检查本地负载。如果负载小于阈值 T1，就探测其它计算机，并在 R 个负载大于 T1 的计算机中登记该计算机的名字，并把登记的内容保留到一个栈中。当一个任务到达一台超载的计算机时，就把这个任务传送到此台计算机栈顶上记载的计算机上。如果一个计算机的负载低于 T1，就清空栈里保留的所有的计算机名。

三、负载平衡的研究热点

负载平衡的研究至今，至少有二十年的历史。近几年，人们对负载平衡的研究主要集中于以下几个方面。

3.1 进程迁移

当一个进程突然大量地申请某种资源，可能破坏已达到的负载平衡。如果能动态地访问系统负载，并能在进程的生命期内，重新合理分布进程，那么尽管移动进程需要一定的开销，但系统的吞吐量仍能获得较大的提高。进程迁移还可以用于故障恢复。

进程迁移是一种在进程生命期内将它从一台计算机传送到另外一台计算机上，并使进程从其"继点"继续进行下去的方法。进程迁移包括系统级和用户级。系统级进程迁移的方法是要在分布式系统中引入进程迁移的机制，必须对系统进行多方面的扩充，包括文件系统、通信机制以及操作系统的内核。用户级进程迁移[6]多是在 PVM 的基础上进行研究，并且只能对用户透明地迁移 PVM 生成的进程。

进程迁移还必须实现寻找空闲计算机。

3.2 在不规则问题中的应用

如果出现对一个数组的间接引用，如 A[B[i]]，一般称之为不规则问题。科学计算中的流体力学和求解偏微分方程等都属于这一类问题。在多机系统中并行求解这类问题时，常常把数组 A 划分到多台计算机上。程序运行时，拥有数组 A 的元素的计算机执行计算。由于 B[i]的值不定，这就造成了运行时每台计算机的计算量不均匀，所以也需要负载平衡。解决这一问题的办法是：定期重分配，或重映射计算机[7]。

3.3 在规则问题中的应用

规则问题的数组下标在编译时可以确定。但是，如果这个应用问题的并行程序运行于一个多用户的异构网络工作站时，由于每一台工作站都可能有不同的拥有者和速度，这便得工作站的负载情况难以预测。为了提高加速比，人们在并行程序中加入负载平衡策略[3]，使它在运行时，自动调整各工作站的负载，即 DO 循环的迭代个数。目前，人们在实现负载平衡时，仅支持 Doall 和 Doacross 循环的并行程序。我们也试图解决并行程序运行时的负载平衡，尤其是当该并行程序的 DO 循环包含数据依赖和控制依赖时。

参 考 文 献

1 Casarant T L, Kuhl J G. A taxonomy of scheduling in general-purpose distributed computing systems. IEEE Tran. On Software Engineering, 1988, SE-14(2): 141～154

2 Kunz T. The Influence of different workload descriptions on a heuristic load balancing scheme. IEEE Tran. On Software Engineering, 1991, SE-17(7)

3 Zaki M J, et al. Customized dynamic load balancing for a network of workstations. Journal of Paraller Computing, 1997, 43: 156～162

4 Eager D L, et al. Adaptive load sharing in homogeneous distributed systems. IEEE Tran On Software Engineering, 1986, SE-12(5)

5 Zhou S. A trace-driven simulation study of dynamic load balancing. IEEE Tran. On Software Engineering, 1988, SE-14(9): 1327～1341

6 Konuru R B, et al. A migratable user-level process package for PVM. Journal of Parallel and Distributed Computing, 1997, 40: 81～102

7 Nicol D M, Reynolds P F Optimal dynamic remapping of data parallel computations. 1990, 39(2)