计算机科学 1999Vol. 26№. 4

并行分布计算中的分布式动态任务调度*)

Distributed Dynamic Task Scheduling in Parallel and Distributed Computing

陈华平 黄刘生 陈国良

(国家高性能计算中心(合肥) 中国科学技术大学计算机系 合肥 230027)

Abstract As one of the most fundmental, critical and challengable problems in Parallel Distributed Computing(PDC), task scheduling has great influence on the execution efficiency of PDC. In this paper, we first describe the concept of dynamic task scheduling and the structure of dynamic scheduler in PDC, then propose the distributed dynamic task scheduling based on hybrid driving way, and finally give a set of rules used to design the algorithm of distributed dynamic task scheduling.

Keywords Parallel and distributed computing, Distributed dynamic task scheduling, Designing rules

1. 引言

并行分布计算中静态的启发式任务调度算法都要求并行分布程序任务在执行前是比较确定的。但一般情况下,实际并行应用程序并不满足这一限制条件。在执行前存在着许多不确定性因素,主要有:并行程序任务中的循环次数事先并不确定;条件分支语句到底执行哪个分支,在程序执行前不能完全了解;每个任务的工作负载大小事先不能确定;任务间的数据通讯量大小只有在运行时才能决定;有些任务是动态产生的。虽然能通过某些技术把这些不确定性转化为确定性,如对条件分支的归纳[1],但是,并行分布程序中存在的许多不确定性是不能在编译时予以解决的,这时只能采用动态调度的办法来有效处理这些不确定性因素。本文我们首先说明了动态任务调度的基本概念及动态调度程序的基本结构,并提出了分布式动态任务调度中基于混合驱动策略的方法,最后给出了设计分布式动态任务调度算法时应考虑的一组规则。

2. 动态任务调度程序的结构

按照动态调度程序是集中存储到一个处理器上,还是分布存储到各个处理器上,动态任务调度算法主要有集中式和分布式两种结构[2,3,4]。在集中式调度方法中,由一个叫作中心调度器的处理器来收集全局调度信息,其它处理器把它们的状态信息传送给中心调度器,并由中心调度器作出调度决定。而分布式调度则是由各自处理单元的调度程序根据局部范围内的一些调度信息来进行任务调度。

图1是集中式动态调度程序的一般结构。在集中式情形下,可将一个处理器专门用于调度(不妨设为P_k),它也称为调度处理器,其余每个处理器P_i($1\leqslant i\leqslant n$且$i\neq k$)有一个局部的任务等待队列WQ_i,WQ_i包含了所有分配到处理器P_i上的待执行任务。一个任务在执行时,有可能请求其它任务的服务,这些请求被传送到P_k上的调度队列中。调度处理器可按先来先服务的原则给其余处理器分发任务,按什么原则选择执行分发任务的处理器是这类动态调度

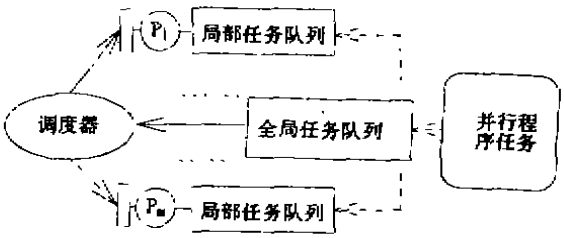

图1 集中式动态调度程序的一般结构

*)本文受中国科大青年基金和国家863重点项目(863-306-ZD-07)的资助。陈华平 博士,副教授,主要从事并行分布计算和网络计算的研究。黄刘生 副教授,副主任,现主要从事分布算法方面的研究。陈国良 博士生导师,主任,现主要从事并行分布计算和智能计算的研究。

算法的关键所在。常用的原则是按照等待队列中任务数目，或者按照等待队列中所有任务的总工作负载大小来选择处理器，当很难获得任务工作负载的精确估计时，一般采用第一种选择原则。

集中式调度的主要优点在于实现比较简单，比较适合于通讯延迟开销较低的共享存储并行计算模型。但在结点数较多的分布存储系统中，由于各结点与调度服务器的通讯成为瓶颈，所以调度开销比较大。因此，除非结点数目较少、或者在底层硬件系统中采取比如超级集线器这样的一些特殊实现措施[5]，否则在分布存储的并行系统中不太采用集中式调度方法。下面图 2 是分布式调度程序的一般结构。

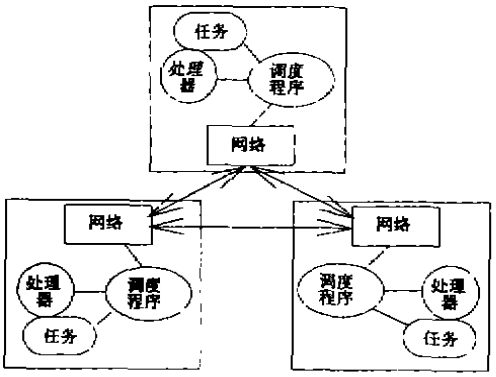

图 2　分布式动态调度程序的一般结构

3. 分布式动态任务调度的驱动策略

3.1　调度约束条件

分布式调度是根据局部范围内的一些系统信息来进行任务调度操作，调度约束条件也就是这些系统调度信息的一个目标函数，是指在什么样的情况下才开始进行任务调度。最为基本的一种调度约束条件是平衡约束条件[6]，该条件主要用来局部平衡各处理器的工作负载，每个处理器根据系统的拓扑连接结构及平衡约束条件，把部分过重任务负载迁移到负荷较轻的相邻处理器上。由于每个处理器与其相邻局部范围内的其它处理器协同执行调度操作，因此，这些操作会把负载调整传播到整个系统中。

一般的成本约束条件不仅考虑处理器之间的负载平衡，而且还针对不同的具体并行分布系统，进一步根据某些成本指标进行任务调度，例如，在基于网络环境的并行分布系统中，通讯延迟开销较大，虽然通过负载平衡调度能获得一些收益，但有可能这些收益还远抵不上由于任务迁移所带来的通讯开销。因此，成本约束调度方法首先根据平衡约束条件确定需要迁移的候选任务，但至于该候选任务到底迁移与否，还要进一步通过检查任务迁移所引起的某些成本开销来确定。除了通讯成本指标外，还有内存开销及磁盘操作成本等指标，有时为了使某项成本最低，甚至不惜以负载不平衡为代价。例如，有时我们不管每个处理器的工作负载如何，而尽量要求每个处理器的 I/O 操作数量比较平均。

在下面的讨论中，我们假定每个结点上的任务是动态产生的，每个结点的负载大小是动态变化的，为了简单起见，我们采用负载平衡作为调度约束条件。

3.2　任务调度的驱动策略

(1)**接收者驱动策略**。其主要思想是由空闲结点逐个向邻接结点请求任务，如果请求到任务，那么就中止请求，否则就继续询问下一个相邻结点。也有可能所有相邻结点都没有满足请求，那么请求结点就等待，过一段时间后再向相邻结点发出任务请求。很明显，这里等待多长的一个时间段 λ 后再发出任务请求很关键，如果 λ 过短，那么就会加重忙结点的负担；如果 λ 过长，那么有可能相邻结点早就有过重负载的任务，这样就浪费了空闲结点的计算资源。

接收者驱动主要有下列几个优点：(1)不需要相互交换负载信息；(2)对于大规模并行计算问题，当每个结点均处于忙状态时，几乎不需要额外调度开销；(3)负载平衡的许多工作由空闲结点来完成，没有给忙结点增加许多额外负担。接收者驱动的主要缺点是，在开始和结束阶段时任务数相对较少，许多任务请求会延迟忙结点的执行。

(2)**发送者驱动策略**。其主要思想是结点间的任务调度分配由创建任务的结点来执行，至于分配给哪个邻接结点，主要取决于邻接结点的负载状态，因此，该策略需要交换处理器的负载信息。一个结点有多种方法向邻接结点通知它的负载情况，如定期询问，每当任务数发生变化，接收到执行任务请求、响应请求或者当任务数超过一定阈值时。

发送者驱动的主要优点是，没有过重负载的忙结点不会被空闲邻接结点所打扰，这一点在系统整个负载较低时尤其重要。发送者驱动的主要缺点是，负载过重的忙结点还要额外增加处理负载平衡调度的负担。

(3)**混合驱动策略**。其主要思想是结合接收者驱动和发送者驱动这两者的优点[7]，只有当负载状态发生变化时，才交换负载状态信息，因而减少网络竞

争。另外，具有空闲信息的负载消息也被作为是任务请求，如果不能满足该请求，那么就记录下请求任务的结点号，以后产生新任务时，就把该任务发送到该空闲的邻接结点上，这样就避免了接收者驱动策略中的反复请求，从而减少通讯开销。同时，为了减少消息传递数量，可在向邻接结点传递任务消息和结果消息时，把自身当前的负载状态消息也附上。因此，每个结点都具有记录邻接结点负载状态信息及相互间任务转递情况的一组属性。

另外，上面三种策略在定位每次平衡调度的源结点或目的结点时，可选择首次适应算法或最佳适应算法，前者只要找到满足平衡调度条件的某个结点即可，不需检查所有邻接结点；而后者每次总是检查比较所有邻接结点，挑选一个最佳的结点来完成这次负载平衡调度。

4. 分布式动态任务调度算法的设计规则

根据图 2 可知，一个分布式任务调度程序由一组相互协作、共同执行来完成分布式任务调度功能的局部任务调度器来组成。一般地，设计一个分布式任务调度算法时，需要考虑下列几个规则。

(1)**启动规则**：决定由谁来启动任务调度过程。一般地，按一次局部任务调度操作是由任务发送处理器来启动，还是由任务接收处理器来启动，主要可分为接收者驱动和发送者驱动这两种启动策略。在这方面 D. L. Eager[8]等做了较多的工作，他们的模拟结果表明，当整个任务负载较重时，接收者驱动策略显得效率好一些。

(2)**传送规则**：决定何时启动任务调度。动态任务调度操作会给系统带来额外开销，如果任务调度操作频度过高，那么它所获得的收益可能还抵不上系统额外开销。以负载平衡调度为例，有的策略是当两个计算结点的任务负载差额到达一定阈值时进行负载平衡调度，而有的策略则是当某一计算结点任务负载为空时才与相邻处理器进行局部负载平衡调度。

(3)**选择规则**：选择哪个(些)任务进行调度。在非抢占任务调度方案中，可以从就绪任务队列中选择任务；而在抢占式情形下，处理器上的所有任务都在选择范围之内。另外，一般尽量选择与本处理器上其它任务相关性较小的任务集进行局部调度。

(4)**定位规则**：选择一组处理器单元，轮流访问它们以从中选择一个来执行拟调度的任务。一般说来，选择范围越大，那么带来的额外开销也越大，选择相邻处理器集进行局部负载信息交流，是设计分布式动态负载平衡调度算法的一种常用方法。

(5)**接受规则**：该规则用于决定任务到底放在哪个处理单元上比较合适。在异构型的并行分布计算环境下，这与计算结点本身的执行性能、其上的工作负荷，以及被调度任务所在计算结点与选中计算结点间的通讯性能(包括传送速率、通讯中继和通讯竞争等)有关。另外，对由定位规则选择的处理器集可采用首次适应法或最佳适应法来选择处理单元。

(6)**信息规则**：收集保留哪些系统信息，用以决定任务调度策略。每个处理器为了进行局部任务调度，必须要了解局部范围内的任务负载情况及底层系统特性，与定位规则相似，通常采用相邻处理器作为局部系统信息范围。另外，局部调度信息有比较简单的，如在负载平衡调度中主要以处理器负载作为调度信息；但也有比较复杂些的，如在成本约束调度方案中，还必须以 I/O 开销或内存开销等一些成本指标作为调度信息。

有些分布式任务调度算法的设计可能并不需要考虑上述全部规则，并且在不同的调度系统中，这些规则有许多种不同的实现方式。并且，一个功能强大的分布式调度系统必须具有一定的自适应性，即它们能根据系统状态的变化，或者以前调度获取的一定知识来调整某些调度策略。

参 考 文 献

1 陈华平，李京，陈国良．并行分布计算中条件分支的静态调度．小型微型计算机系统，1997，18(1)：6～12

2 Casey L M. Decentralized scheduling. Australian Computing Journal，13：58～63

3 Casavant T L，Kuhl J G A Taxonomy of scheduling in general-purpose distributed computer systems. IEEE Trans. Software Engineering，1988，14(2)：141～153

4 Bal H E. Programming distributed systems. Silicon Press，1990

5 Jan G E，Lin Ming-Bo. Effective load balancing on highly parallel multicomputers based on superconcentrators. In：Proceedings of the 1994 Intl. Conf. on Parallel Distributed Systems. Hsinchu，Taiwan，ROC. 1994. 19～21

6 Feng M D，Yuen C K. Dynamic load balancing on a distributed system. In：Proceedings of the 6th Symposium on Parallel and Distributed Processing. DALLAS，TEXAS，oct. 1994. 26～29

7 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

8 Eager D L，et al. A comparison of receiver-initiated and send-initiated adaptive load sharing. Performance Evaluation，1986，6：53～68