并行分布计算中的任务调度及其分类*)

Taxonomy of Task Scheduling in Parallel and Distributed Computing

陈华平　黄刘生　安　虹　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 and Distributed Computing(PDC), task scheduling has great influence on the execution efficiency of PDC. In this paper, we first describe the concept of task scheduling, then point out the difference among job scheduling, task allocation and general task scheduling, and lastly give the taxonomy of task scheduling according to scheduling strategies which are commonly used in PDC.

Keywords　Parallel and distributed computing, Task scheduling, Taxonomy

1　并行分布计算中的任务调度问题

并行分布计算中的任务调度问题就是根据一定的调度规则和调度策略,把组成并行程序的一组任务或构成工作负载的一组作业,按照一定执行时序分配到并行分布系统的多个计算结点上,以期取得较好的系统执行性能[1,2]。目前许多基于并行分布处理的高性能计算中心的计算环境是由多种并行机或网络工作站机群系统构成的异构多应用系统,并且某些并行机的内部计算结点也可能是异构的(如 Paragon 系统中有些结点的内存为16M,有些结点的内存为32M;或者有些结点为单CPU,有些结点为双CPU),这时不同的应用层次对任务调度有不同的要求,下面就从作业调度、任务划分与一般任务调度概念的区别出发,讨论并行分布计算中的任务调度问题。

1.1　任务调度与作业调度

一般地,按照不同的应用层次,并行分布计算中的调度问题可以分为两大类,即任务层调度和作业层调度,虽然有时把它们两者统称为并行分布计算中的任务调度问题,但两者的调度着眼点(或调度对象)不同,前者主要针对某个用户构成单一并行应用程序的一组任务(子任务),而后者主要针对若干个用户的多个并行应用程序构成的一组作业,所以它们的调度策略及系统性能评价指标均有很大差别。我们把前者简称为任务调度,把后者简称为作业调度,一个并行应用程序相当于一个作业,一个作业由多个任务组成,整个系统负载由若干个用户作业组成。

任务调度所面向的一般是单应用程序系统,其调度实体是任务,调度目标一般就是求得某个作业最短的执行时间。任务调度必须了解该作业的具体结构,如以任务图形式表示的结构信息,它所考虑的问题也要具体些,如任务分解、任务间的优先关系及任务聚类等。作业调度所面向的一般是多应用程序系统,它的调度实体是传统批处理意义下的作业,即把每个作业作为一个调度单位,这时调度程序一般不太了解作业的具体结构信息,它的调度目标有很多[3,4],如最短的平均作业响应时间、最大的资源利用率或最大的系统吞吐率等,这些不同调度目标有时是相互冲突的[3]。

从用户角度出发,每个用户关心的是自己的作业能否在较短的时间内得以完成,他们比较注重的是构成自己作业的多个任务之间的任务调度策略,并且比较倾向于同构的计算环境,即每个计算结点的性质(包括处理速度、内存大小等)相同,从系统角度出发,它更关心的是如何通过有效的作业调度方案,使整个计算资源与不同用户作业的多种资源要求获得最佳匹配,从而提高整个系统的执行性能。

事实上,作业调度是任务调度的一种特殊形式,因为一个作业的所有进程可以组合成一个任务。另外,调度相互独立的一组任务也可视作为作业调度。并行分布计算中的任务调度问题着重研究任务调度方案中的各种调度规则及策略,因此,"调度系统"、"调度程序"和"调度算法"这几个术语的本质内容主要就是指这些

*)本文受国家863项目(863-306-ZD01-02-03)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,现主要从事分布算法方面的研究。安　虹　博士,主要从事高性能计算环境的研究。陈国良　博士生导师,国家高性能计算中心(合肥)主任,现主要从事并行分布计算和智能计算的研究。

调度规则和策略。

1.2 任务调度与任务划分

目前许多文献把"任务调度"与"任务划分"这两个术语互用，虽然在并行分布计算中任务划分是任务调度的一种特殊形式，但对两者不加区别有时会带来误解。

一般说来，在分布系统中一组任务相互间发生关系的方式有多种多样，其中有一种是"通讯关系"，在通讯关系情况下，如果两个任务被分配到不同的处理器上，那么就产生以通讯成本形式表示的通讯开销，反之，如果两个任务被分配到相同的处理器上，那么就不产生任何通讯开销，这种特殊的任务调度我们称作"任务划分"。

1.2.1 任务划分 对于任务划分，我们通常用任务作用图 TIG(Task Interaction Graph)来表示并行应用程序模型，其中 TIG 中的结点表示任务，当且仅当两个任务结点间相互作用时，TIG 中的这两个结点间存在一条无向边。因此，进行任务划分的一组任务不需要指定各任务之间的执行次序，也就是说，任务之间相互作用时不存在优先关系，而主要着重考虑如何把各个任务分配到合适的处理器上。另外，任务划分所面向的分布系统一般均是异构的，即各处理器的执行性能不同。下面给出的问题1是一般的任务划分问题定义。

问题1 给定一组任务 $A=\{a_1,a_2,\cdots,a_n\}$，一组处理器 $P=\{P_1,P_2,\cdots,P_m\}$，一个通讯成本矩阵$[c_{ij}]$(其中 c_{ij}表示任务 a_i与任务 a_j 被分配到不同处理器上时的通讯成本)，一个执行成本矩阵$[t_{ij}]$(其中 t_{ij}表示任务 a_i在处理器 P_j 上执行时的成本开销)，及一个正数 H，是否存在一个任务划分方案，使得 $EC+CC\leqslant H$，其中 $EC=\sum_{i=1}^{n}t_{ik}(K=P(a_i))$，$CC=\sum_{p(a_i)\neq p(a_j)}c_{ij}$，$p(a_i)$表示分配给任务 a_i 的处理器号。

通过采用先把"图划分问题"转换成"完全分割图的团图划分问题"，然后再把后者转换成问题1的方法，Hesham H. Ali 等人[5]证明了问题1的 NP 完全性，而"图划分问题"的 NP 完全性由 M. Garey 和 D. Johnson 等人[6]所证明。

求解任务划分问题的最有效方法是首先把任务划分问题转换成求解二端网络中的最大流量-最小分割问题，然后利用网络流量算法获得最优的任务划分结果，其中最为著名的网络流量算法要数 H. Stone[7]于1997年提出的。更有意义的是，不管是最优任务划分还是启发任务划分，有关任务划分问题的绝大多数著名结果均是通过这种转换方法获得的，有一个例外就是上面问题1的 NP 完全性证明，它是通过把任务划分问题转化为"完全分割图的团图划分"模型来求解的。

1.2.2 任务调度 对于任务调度，任务间相互作用所发生的关系方式是"优先关系"，调度这些任务有时也称为"优先约束调度"。通常用带权有向图 $G=(V,E)$来表示并行程序模型，其中顶点集 V 为表示所有任务的有权结点，任务的工作负载为该结点的权重；E 为表示任务间通讯关系的带权有向边集，任务之间的通讯量为这些有向边的权。任务调度问题除了考虑每个任务的执行处理器外，还必须考虑每个任务的执行时序，因为每个任务只有当比它优先级高的所有任务执行完毕后，才有可能开始执行。下面给出的问题2是最为一般的任务调度问题定义。

问题2 给定一组任务 $A=\{a_1,a_2,\cdots,a_n\}$，这组任务相互之间满足偏序关系"$<$"，每个任务对应的权重为 $w(a_i)$，另外给定 m 个处理器及一时间限制 h，是否存在一个从 A 到$\{0,1,\cdots,h-1\}$的全函数 f，该函数必须满足三个条件：①如果 $a_i<a_j$，那么 $f(a_i)+w(a_i)\leqslant f(a_j)$；②对 A 中的每个任务 a_i，满足 $f(a_i)+w(a_i)\leqslant h$；③对区间$[0,h)$上的任何一点 t，至多存在 m 个 i 值满足 $f(a_i)\leqslant t<f(a_i)+w(a_i)$。

问题2的条件①规定了各任务在时序上必须满足的约束条件；条件②规定了每个任务必须不迟于时刻 h 完成，即限制了调度长度不超过 h；条件③限制了某一时刻最多只能有 m 个任务处于执行状态。问题2的 NP 完全性由 Richard M. Karp 于1972年给出了证明。

问题2是一般性的任务调度问题描述，还有其它许多具有 NP 难度的受限任务调度问题，它们是在问题2的基础上，通过对任务模型或处理器模型附加一些限制条件而产生的。

2 并行分布计算中任务调度的分类

任务调度是并行分布计算中最为基本的一个问题，涉及到并行分布计算环境及实际并行应用程序的各个方面，因此，它从各个不同侧面影响着并行程序的设计及并行分布计算环境的整个系统执行性能。从不同的角度出发，并行分布计算中的任务调度问题有不同的分类方法[8]，下面所述的各种分类方法同时也比较全面地反映了设计一个任务调度算法时需要考虑的各种因素。

2.1 按照何时执行处理器分配操作划分

按照何时决定每个任务的执行处理器号，并行分布计算中的任务调度方法主要可分为静态调度、动态调度和混合调度这三类。静态调度是指在并行程序编译时，就决定每个任务的执行处理器及执行时序，经常用于任务图比较确定的情况下。而动态调度则是在并行程序运行过程中，根据当前任务调度及系统执行情况，临时决定每个任务的执行处理器及起始执行时刻。

动态调度主要用于任务图不确定的情况下，但它不可避免地会带来额外开销[7,8]。混合调度是介于静态调度和动态调度两者之间的调度方法，在编译时先静态调度部分任务，而剩余部分则采用动态调度方法在系统运行过程中来给它们分配处理器。

2.2 按照调度目标的实现要求划分

按照调度目标的实现要求来划分，并行分布计算中的任务调度可分为最优调度和近似最优调度，后者也经常称为启发式任务调度。最优调度一般是指静态调度，如果一个调度算法能在多项式复杂度的时间内获得最佳调度结果，那么称之为有效的最优调度算法。但是，只有在极少数情况下才存在有效的最优调度算法[1,6]，大部分最优调度算法所需的执行时间随任务数或处理器数目的增加而呈指数增长。因此，经常采用启发式任务调度方法[11]来把各任务调度分配到各处理器上，这虽然不能确保获得最优解，但可以获得最优调度的近似解。

2.3 按照并行分布计算模型划分

按照并行分布计算模型的存储器结构，主要可分为基于共享存储的任务调度和基于分布存储的任务调度这两大类。在共享存储的计算模型上进行任务调度时，一般不需要考虑通讯延迟，这时任务调度的着重点在于如何最大限度地获得并行程序任务间的并行性。而在分布存储的计算模型上进行任务调度时，通讯延迟的存在使得任务调度更为复杂，如果无视通讯延迟的存在，而只考虑尽量增大并行执行时间，那么有可能在较多处理器上的执行效率还不如在较少处理器上的执行效率；同样，如果没有完全开发任务之间的并行性，那么并行处理器就没得到充分利用，因此，调度程序必须在尽可能利用各任务之间的并行性和尽量减少通讯开销之间进行折衷[12,13,14]。随着目前网络工作站机群系统的广泛应用，尤其是网络带宽的不断提高，基于分布存储的任务调度问题显得越来越重要。

2.4 按照调度程序结构或调度信息范围划分

按照调度程序的结构或调度程序所收集调度信息的范围，并行分布计算中的任务调度方法可划分为集中式调度和分布式调度两大类。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，其最大优点在于具有良好的可扩放性[14]。集中式调度的主要优点在于实现比较简单，但在结点数较多的大规模并行分布系统中，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，分布式调度一般都是动态的。

2.5 按照调度程序所执行的调度操作划分

按照调度程序执行调度操作时是否允许任务抢占执行，可分为抢占式调度和非抢占式调度。如果一个任务一旦占有处理单元，那么它就不能被中断执行，而是必须一直运行完毕后才释放该处理器，这种情形即为非抢占执行方式；相反，如果允许任务被中断执行而把它所在的处理单元让给其它任务执行，这种情形即为抢占执行方式，当然这必须假定被抢占任务最终还能获得它所要求的执行时间。由于抢占执行方式所引起的环境切换会增加系统开销，所以一般情况下不常采用，用得较多的地方还是以分时方式实现的群调度（gang scheduling，有时也称为联合调度，即 co-scheduling）[15,16]方案中。与抢占执行调度操作类似的还有任务是否允许迁移执行，一般地，编译时对任务进行的处理器分配我们有时也称为初始分配，如果应用程序任务启动执行以后，某些任务可以被重新移到其它处理器上执行，这也是可抢占执行方式的一种形式。

2.6 按照调度策略是否动态变化划分

按照调度程序的调度策略是否随系统状态信息变化而自动调整，并行分布计算中的任务调度可分为自适应调度和非自适应调度这两大类。自适应调度的基本思想就是根据调度程序在以前一段时间内的执行效果，以及当前系统状态信息（主要包括系统资源和任务负载情况）自动修正调度程序的执行机制。由于它们是通过收集当前系统状态信息来动态修正调度策略，所以一般都是动态的。而非自适应调度方法则一旦确定任务调度算法的调度策略，那么在并行程序执行过程中就固定地按照这些调度策略来进行任务调度，即使某些不确定因素（如动态产生的任务）使得调度效率比较低，那么也必须等到非执行状态时，再来修正调度策略。

结束语 本文主要从作业调度、任务划分这两个具有一些特殊性的任务调度问题，说明了并行分布计算中最为基本、也是具有相当难度的关键问题—任务调度问题，并通过一些常用的调度策略、并行分布计算模型和调度目标，给出了并行分布计算中任务调度的分类方法，这些分类涉及了并行分布计算中任务调度问题的诸多方面。随着基于网络工作站机群的大规模并行分布处理的日益普及，作业调度和任务划分这两个问题将变得越来越重要。

参考文献

1 El-Rewini H, Lewis T G, Ali H H. Task scheduling, PTR Prentice Hall, Englewood Cliffs, New Jersey, 1994, 07632

2 Lewis T, El-Rewini H. Introduction to parallel computing. Prentice Hall, 1992

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, Apr. 1996 1～9

PVM 任务迁移协议的研究*)

The Study of PVM Task Migration Protocol

李 毅 周明天 虞厥邦

(电子科技大学570研究室 电子科技大学计算机学院 成都610054)

Abstract The work load could not be floated in PVM system, this fact limits seriously its functions of the balancing of work and dynamic scalabillity, thereby limits the improving of the parallel resource utilization and parallel running performance. Based on PVM system architecture and the scheme of message driving, this paper proposes a new protocol and its implementation of transparent and independent PVM task migration for supporting system load balancing in dynamic PVM system.

Keywords PVM, Load balance, Task migration, Scalability

1 引言

被作为MPI"事实标准"的PVM[1,2]是一个优秀的异构多机分布计算环境支持软件。出于最初系统设计的考虑,它具极好的静态可伸缩性,但是它不支持负载在系统内部浮动,严重影响了系统资源利用率和并行效率的提高。因此,为PVM增加任务迁移功能,使它成为可负载均衡的动态PVM系统是近年来的研究热点。

文[3,4]主要讨论了PVM任务调度问题,文[5]研究了SPMD计算模型的PVM任务调度,改进了PVM原来任务在主机集上平均分配的模式。但是,一个PVM任务一经启动,仍必须在原主机上一直运行到结束。文[6,7]修改了PVM任务间接通信的协议,每个pvmd用一张转发消息的"路由表"(oldtid-newtid映射表),将消息发送到目标任务的新位置,主pvmd有一张全局的路由表,用于从pvmd发送消息失败后,查找目标任务的新位置,这样可解决任务迁移期间发送给被迁移任务消息的转发问题,以及向已经迁出的任务发送消息失败的问题。文[8]沿用了文[6,7]的思想(包括路由表方法),但做了两点改变:(1)修改通信库函数,使PVM任务在进行socket I/O时,阻塞任务迁移信号,以保证通信信息的原子性;(2)每个主机(host)引入了专用于通信和(与任务迁移有关的)报文

*)本文的工作得到九五国防预研基金(DJ8.1.1)和四川省科技基金(JS05671)的资助。李 毅 博士生,主要研究方向为计算机操作系统,分布并行处理及分布对象技术。周明天 教授,博士生导师,主要研究方向为计算机网络,分布并行处理,分布对象技术和网络与信息安全。虞厥邦 教授,博士生导师,主要研究方向为电路CAD及EDA,并行分布系统及神经网络计算机。

4 Wan M, et al. A batch scheduler for the Intel Paragon with a non-contiguous nodes allocation algorithm. In: IPPS '96 Workshop on Job Scheduling Strategies for Parallel Processing. Hawaii, Apr, 1996. 29～39
5 Ali H, El-Rewini H. Task allocation in distributed systems. Journal of Combinatorial Mathematics and Combinatorial Compruting, 1993, 14. 15～32
6 Garey M, Johnson D. Computers and intractability. A guide to the theory of NP-completeness, W H, Freeman and Company, San Francisco, 1979
7 Stone H. Multiprocessor scheduling with the aid of network flow algorithms. IEEE Trans. Software Eng., 1977. 85～93
8 Casavant T, Kuhl J. A taxonomy of scheduling in general purpose distributed computing systems. IEEE Transaction on Software Engineering, 1988, 14(2)
9 Saletore V. A distributed and adaptive dynamic load balancing scheme for parallel processing of medium-grain tasks. In: Proceedings of DMCC. Portland, Oregon, 1990. 994～999
10 陈华平,计永昶,等. 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
11 陈华平,林洪,陈国良. 并行分布计算中的启发式任务调度. 计算机研究与发展, 1997, 34(suppl.): 74～78
12 Sarkar V. Partitioning and scheduling parallel programs for execution on multiprocessors. MIT Press, 1989
13 Kim S, Browne J. A general approach to mapping of parallel computation upon multiprocessor architectures. In: Proc. of the Intl. Conf on Parallel Processing. 1988. 1～8
14 Joosen W, Pollet J. The efficient management of task clusters in a dynamic load balancer. In: Proc. of the 1994 Intl. Conf. on Parallel Distributed Systems. Hsinchu, Taiwan, ROC, 1994. 19～21
15 Wang Fang, et al. A gang scheduling design for multiprogrammed parallel computing environments. In: IPPS '96 Workshop on Job Scheduling Strategies for Parallel Processing. Hawaii, Apr. 1996. 67～73
16 Hori A, Tezuka H, Ishikawa Y. Implementation of gang-scheduling on workstation cluster. In: IPPS '96 Workshop on Job Scheduling Strategies for Parallel Processing. Hawaii, Apr. 1996. 76～82