实时系统 调度 硬实时系统 操作系统(13)

计算机科学1999Vol.26№.9

并行与分布硬实时系统的调度*)

Scheduling of Parallel and Distributed Hard Real-Time System

毛羽刚 金士尧 张拥军

(国防科技大学计算机学院并行与分布国家重点实验室 长沙 410073)

Abstract This paper discusses some research issues on scheduling of parallel and distributed real-time system, including priority assignment; synchronization and real-time communication; scheduling of independent tasks; scheduling of end-to-end system. It provides their models and algorithms. Some future research directions are also introduced.

Keywords Hard real-time system, Scheduling, Deadline, Precedence constraint, End-to-end system

一、前言

实时系统是工作在时间约束下的系统,与一般计算机系统的主要区别是引入了时间概念,这大大地影响了系统的设计、验证和实现。实时系统不但要保证计算结果的逻辑正确性,而且要在规定的时间内完成计算。如果某个实时任务没有按时完成,则可能导致整个系统失败,甚至引起灾难性后果。这类实时系统被称为硬实时(hard real-time)系统。例如,核电厂或导弹控制系统等。因此硬实时系统的调度理论需要严格和有效的可调度性分析工具去表述和验证任务执行的时间行为。

近十年来,计算机速度、处理能力等性能得到飞速发展,但实时应用对计算能力的需求似乎增长更快。复杂和安全关键的系统越来越多,需要并行和分布式硬实时系统,它们具有良好的并发性、伸缩性和容错性,但并发会引起任务间的同步和通信开销以及资源约束增加,系统的可预测性降低,使得任务调度更加复杂。虽然在某些情况下,存在多项式的解决方案,但一般来说,是NP难问题,因为并行和分布环境中的实时任务大多非独立且相互作用,形成一定的通信和同步模型,共同响应外部事件。所有任务可在不同的处理器上执行,使得任务组调度很难有一个通用的形式。一般将其大致分为三种:独立任务组的调度;具有前趋约束关系任务组的调度;具有任意同步关系任务组的调度,其算法复杂性依次增加。所以一般都寻求启发式算法解决这类具有复杂需求包括资源和前趋限制的约束问题。本文将探讨该领域当前主要思想和理论并给出调度分析模型。

二、调度算法

2.1 优先级分配

并行与分布实时系统的调度可分为三个子问题:优先级分配;任务间同步和通信;可调度性分析和验证。最后一个是指:给定优先级分配方法和任务间同步及通信模型,我们如何验证一个端端任务系统是可调度的。在硬实时系统中,任务往往是周期性的,优先级固定分配,一般使用优先级驱动的调度方法,因为它实现简单,可调度性分析容易,瞬时过载的行为可预测。任何时候,具有最高优先级的任务首先执行,一个正在执行的任务实例(周期任务的每次执行),除非自动放弃处理机或被高优先级任务抢占将连续执行直到结束。

并行与分布环境下的任务优先级分配基本上是个组合优化问题,是NP难的,一般采用启发式分配方法,例如,基于应用要求;基于RM(rate-monotonic)的分配;基于时限(deadline)的分配;和基于最优的分配算法。单机系统基于RM的优先级分配方法可直接应用于并行与分布系统,每个任务的优先级与其周期成反比,但已不是最优分配。基于时限的方法是指每个任务的优先级与它的相对时限成反比。相对时限短,优先级高。在最优分配方面,Tindell

*) 本研究得到95国防预研项目资助

等[1]使用模拟退火技术寻找可行的优先级分配算法，同时给出了可行的负载划分方法。后来，Garcia和Harbour[3]根据效率和寻找可行方案的概率进行了改进，开发出HOPA算法。这两个算法的初始优先级随机分配，或根据基本的分配方法，然后连续调整直到找到可行答案为止。在调整优先级分配过程中使用一个确定的指导标准，一个好的标准将使调整过程迅速而有效。基于最优的方法一般比其他方法性能好，但开销大。

2.2 同步

分布环境下的同步关系可分为：时钟同步、资源同步和任务同步：

1)时钟同步：时钟同步对分布式实时系统来说极为重要，必须使用确定性的同步技术始终保持各单机系统的时钟一致(误差不超过一定的阈值)。可通过硬件或软件实现。硬件方法代价较高，一般使用软件方法，例如取均值法或取中值法等。其同步精确度与实时通信延迟有关。

2)资源同步：协调任务对共享资源的竞争。常用的资源同步算法有集中式算法，分布式算法，令牌环算法等。如果系统结构和通信延迟确定，这些算法开销都是可预测的。资源同步过程中最大的问题是优先级倒置，即由于资源共享而引起的低优先级任务不可预测地阻塞高优先级任务的执行。不过目前已有许多方法解决该问题，例如：优先级继承协议、优先级顶级(Ceiling)协议、堆资源协议(SRP)等。利用这些协议可限定阻塞时间。

3)任务同步：任务间有协作或约束关系，需要同步协议。任务同步协议实际上是一种调度协议，它控制着任务的释放或执行，决定着系统可调度性分析方法和结果。

2.3 实时通信

分布式实时系统中，通信频带是系统最重要的资源之一。同步和数据交换一般要经过消息传递，实时通信为保证消息传递时限起着关键作用。它最重要的性质是需要确定的和有界的消息传递延迟。实时通信网络可分为两类：共享介质网络和点到点网络。在共享介质网络中，处理器共享信息载体。包括实时以太网、实时总线(TDMA)、令牌环、FDDI等。这些网络的通信协议实际上是一种分布式消息调度算法，优点是广播性能好，最坏消息延迟有界，但容错性不好，各节点网络带宽有限，并且与网络上连接的节点数有关。点到点网络的优点是速度快，容错性好但不具有广播机制、结构复杂。目前，大多数分布实时系统采用共享介质网络。但由于点到点网络频带宽、容错和可扩展性好，正越来越多地用于实时系统，被认为是今后实时通信研究和发展的一个方向。

实时通信也可分为面向连接和面向无连接的通信。前者在实际通信开始之前就先建立好源和目的之间的通信链路，通信分为建立连接和消息传递两个阶段。面向连接的实时通信比较适合静态脱机的任务调度，最好是一次建立、长时通信的任务模型，面向无连接的通信虽然没有连接建立开销，但要保证其传输时间有界则比较难，要考虑路由、拥塞以及通信节点的消息调度问题。

为了保证分布式实时系统的各任务满足时限，必须限定实时通信的开销，预测通信延迟的上限。对于共享介质网络，可将网络看作"链路处理机"。例如，对于定时令牌协议，各节点之间的同步传输带宽必须满足如下关系：$\sum_{i=1}^{n} H_i + \tau \leqslant \mathrm{TTRT}$(target token rotation time)。其中，H_i为节点i的同步传输带宽，τ是令牌沿网一周的传播延迟。因此，对于本地消息来说，其他节点占用的网络频宽可定义为周期等于TTRT，执行时间为$(\mathrm{TTRT}-H_i)$的本地高优先级任务发送消息所需的开销。此时，网络适配器中的消息传递延迟可根据单机调度理论分析和预测。对于由点到点网络互连的硬实时系统，任务及其消息都具有周期性，通常采用面向连接的通信，实时通道一旦建立，长期有效。在通道建立过程中，某个中间节点能否接受连接请求主要应考虑两点：一是它不能影响已经建立的实时连接，可使用类似于下节将要阐述的实时调度理论分析和判定；二是所有中间节点的最大传输延迟之和不能超过点到点消息传递时限，判断方法类似于第2.5节阐述的端端系统任务调度。如果一个中间节点能满足这两个要求就可接受连接请求，否则拒绝，另谋出路。

2.4 独立周期任务组的调度

独立周期任务组的调度相对比较简单。一般将全局调度策略划分为任务分配算法和局部节点调度算法。单机调度算法已很成熟，只需研究任务的分配策略。任务分配类似于背包问题，可使用已有的许多启发式算法将任务分配到各处理机上。但与背包问题不同的是：这里的包是指处理机的利用率，是变量。例如，Dhall和liu在文[3]提出如下基于RMS[4]调度的分配方法：

·首先适应法(RMFF)：将任务首先分配给能满足时限要求且处理机序号最小的节点。

·下一适应法(RMNF):将任务首先在当前处理机(P_i)上分配,如果不能满足时限要求,则选择下一处理机(P_{i+1})。

·最好适应法(BF):按处理机号从小到大的顺序,将任务分配给能满足时限要求且利用率最高的处理机。

每次给处理机分配新任务时应不影响原来已分配任务的可调度性,必须考虑如下调度条件:

条件 WC(worse case):对一组 n 个任务 $\tau_i=\{(C_i,T_i)|i=1,\cdots,n\}$,其中,$C_i$ 是任务执行时间,T_i 为周期,如果满足条件 $\sum_{i=1}^{n}C_i/T_i\leqslant n(2^n-1)$,则该任务组是可调度的。

该条件由 Liu 和 Layland[4]提出,是最坏情形下的条件,充分但不必要。为此,Dhll 和 liu[1]进行了改进,提出如下条件:

条件 IP(increase period):假定 $\tau_1,\tau_2,\cdots,\tau_m$ 是 m 个任务,其周期分别为:$T_1\leqslant T_2\leqslant\cdots\leqslant T_m$,设:$u=\sum_{i=1}^{m-1}C_i/T_i\leqslant(m-1)(2^{1/(m-1)}-1)$,如果 $C_m/T_m\leqslant 2(1+u/(m-1))^{-(m-1)}-1$,则该任务集可由 RMS 算法调度,任务 τ_m 可以分配到该处理机上。

当 m 趋于无穷时,τ_m 的最小利用率为$2e^{-u}-1$。该条件仍然是充分但不必要条件,以后,Lehoczky 等[5]又开发了一个充分必要条件:

条件 IFF(if and only if):给定周期任务组$\{\tau_1,\cdots,\tau_n\}$,如果使用 RMS 算法调度,则有:1)对于任务 τ_i,当且仅当 $L_i=\min_{t\in S_i}(W_i(t)/t)\leqslant 1$时,任务 τ_i 是可调度的。2)对于所有任务,当且仅当 $L=\max_{1\leqslant i\leqslant m}L_i\leqslant 1$时,任务集是可调度的。其中 $S_i=\{jT_k|k=1,\cdots,i,j=1,\cdots,\lceil T_i/T_k\rceil\}$,$W_i(t)=\sum_{k=1}^{i}C_k\lceil t/T_k\rceil$,$L_i(t)=W_i(t)/t$,$L_i=\min_{t\in S_i}L_i(t)$。

以上任务分配主要是从提高处理机利用率的角度来考虑的,如果需负载平衡,则在任务分配时应尽量保持各处理机利用率相等。另外,任务分配时还要充分考虑任务的特性。对于硬实时任务通常采取100%的资源预分配策略,并静态地将任务分配到处理机上。

2.5 具有简单前趋约束关系的实时任务组的调度

对于具有简单前趋约束关系的系统,目前研究比较多的是端端系统,又叫工作流(flow-shop)系统。一方面由于它的同步关系简单,便于研究,是开发复杂任务同步模型调度算法的基础;另一方面,端端系统在航空、航天、国防、工业过程控制系统等领域有广泛的应用。系统中的每个端端任务为具有端端时限的周期任务,优先级固定分配,任务由若干子任务组成,第一个子任务周期性地释放,以后子任务在不同的处理机上顺序执行,只有当前一个子任务完成后,下一个子任务才可释放,子任务之间的同步可采用相位修改协议或直接同步协议,前者的方法是:在充分估计各子任务最坏完成时间的基础上,确定各子任务的第一次释放时间 $f_j=f_1+\sum_{k=1}^{j-1}R_{i,k}$,其中 f_1 是每个任务的第一个子任务的首次释放时间,$R_{i,k}$ 是第 j 个子任务之前的第 k 个子任务的最坏执行时间。然后各子任务按照端端任务的周期独立运行。其优点是可使用单机忙周期分析法计算各子任务的最坏响应时间,这些响应时间之和就是整个端端任务的最大响应时间 EER。而后者采用的方法是:当前一个子任务完成后,立刻释放后一个子任务。该协议简单,但每个节点上的子任务周期性不好,可调度性分析复杂。要考虑释放抖动问题,即每个子任务(第一个子任务除外)的释放时间与其前趋子任务的完成时刻有关。在可调度性分析和验证时要考虑该因素。我们以任务 T_i 的第 j 个子任务 $T_{i,j}$ 为例,以单机系统忙周期分析法[5,6]为基础,给出可调度性分析算法如下:

1)计算忙周期 $D_{i,j}$:

$$D_{i,j}=\min\left\{t>0\,|\,t=\sum_{\tau_{u,v}\in(H_{i,j}\cup T_{i,j})}\left\lceil\frac{t+R_{u,v-1}}{p_u}\right\rceil\tau_{u,v}\right\}$$

其中,$H_{i,j}$ 是高优先级子任务集;$R_{u,v-1}$ 是前趋节点机的高优先级子任务的 IEER 时间(从第一个子任务到中间节点机子任务的端端响应时间)。它影响着本节点机的高优先级子任务的释放时间,从而也影响 $T_{i,j}$ 的完成时间。p_u 是高优先级任务的周期,$T_{u,v}$ 是高优先级子任务,$\tau_{u,v}$ 是其执行时间。t 从大于0的一个小数开始递归迭代计算。一般 t 的初始值可设为子任务的执行时间 $C_{i,j}$,即最小响应时间。可以证明:如果 $\sum_{\tau_{u,v}\in(H_{i,j}\cup T_{i,j})}\tau_{u,v}/p_u\geqslant 1$,则 t 收敛。

2)计算 $D_{i,j}$ 时间内任务 $T_{i,j}$ 的实例数:

$$M_{i,j}=\left\lceil\frac{D_{i,j}+R_{i,j-1}}{p_i}\right\rceil$$

3)m 从1到 $M_{i,j}$:

(a)计算 $T_{i,j}$ 各实例的完成时间:

$$C_{i,j}(m)=\min\{t>0\,|\,t=m\tau_{i,j}+$$

$$\sum_{T_{v,o}\in H_{i,j}}\left\lceil\frac{t+R_{v,o-1}}{p_v}\right\rceil\tau_{v,r}\Bigg\}$$

(b)计算第 m 个实例的 IEER 时间：

$R_{i,j}(m)=C_{i,j}(m)+R_{i,j-1}-(m-1)p_i$

4)计算最大 IEER 时间：$R_{i,j}=\max\{R_{i,j}(m)\}$，$1\leqslant m\leqslant M_{i,j}$

5)最后一个子任务的 IEER 时间就是端端任务 $T_{i,j}$的最大响应时间 EER。

当任务响应时间小于端端时限时，该任务可调度。如果系统中所有端端任务都是可调度的，则该系统是可调度系统。否则需重新划分子任务或分配优先级或增加处理机计算能力，直到满足要求。

三、展望

本文对并行与分布式实时系统的调度理论进行了研究。目前单机系统的实时调度理论比较成熟，已经提出静态和动态的最优调度算法，并得到了广泛应用。原则上，多机和分布式实时系统的调度是NP难的，所以本文的研究主要集中在静态、固定优先级的独立或具有简单约束关系的实时任务集调度，今后可以进一步深入的领域有：

1)优先级分配：可利用当前一些先进的组合优化问题的解决思想、算法开发有效的优先级分配算法，减少系统的响应时间。

2)增加任务同步模型的复杂性：例如，一个子任务可以具有两个或多个前趋子任务；一个子任务的完成也可以释放两个或多个子任务；多个具有前趋关系的相邻或不相邻子任务可以在同一台处理机上执行等等。

3)各种实时网络的可调度性：包括共享介质网络和点到点网络，并与主机的调度有机结合起来进行研究。调度的粒度可以从消息减小到报文，使分析更加准确。

4)动态调度：包括优先级的动态分配。研究基于EDF(earliest deadline first，最早期限优先)调度理论的并行与分布式实时系统的调度。这对于硬实时系统来说很困难，需要限制系统模型的复杂度。

5)针对具体的系统模型研究调度算法：面向实际应用，具体问题具体分析。

主要参考文献

1 Tindell K, et al. Allocating real-time tasks An NP-hard problem made easy, Real-Time Systems Journal, 1992, 4(2)

2 Garcia J, Harbour M. Optimized priority assignment for tasks and messages in distributed hard real-time systems. In. The Third Workshop on Parallel and Distributed Real-Time Systems April 1995. 124～132

3 Dhall S K, Liu C L. On a Real-time Scheduling Problem. in Operations Research, 1978, 26: 127～140

4 Liu C L, Layland J. Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment J. Assoc. Comput. Machinery, 1973, 10(1): 174～189

5 Lehoczky J, Sha L, Ding Y. The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior. IEEE Real-Time Symposium 1989. 166～171

6 Lehoczky J. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In. 11th IEEE Real-Time Systems Symposium. 1990. 201～209

7 Stankovic J A, Spuri M. Implications of Classical Scheduling Results For Real-Time Systems, June 23, 1994

8 Sun J, Liu J W S. Synchronization protocols in distributed real-time systems. In. The 16th Intl. Conf. on Distributed Computing Systems. Hong Kong, May 1996

(上接第89页)

模型可被用来评估所要做的变动对系统性能产生的影响，如果所要做的变动完全是为了增强系统的性能，那么所做变动的损/益比将可以精确地确定。

·**系统配置**。作为面向市场的产品，系统经常要包含许多选项，使用户能够根据自己的需要来配置系统，比如，磁盘的数量、处理器的能力、通信线路的传输速率。性能模型可以作为一种工具，帮助用户配置新系统。

参考文献

1 Tanenbaum A S. Distributed Operating System. Prentice Hall Intl., Inc., 1996

2 汤子瀛，哲凤群，汤小丹. 计算机操作系统. 西安电子科技大学出版社，1998

3 Highlevman W H. Performance Analysis of Transaction Processing System. Prentice Hall, Englewood Cliffs, New Jersey, 1992