一种改进的分布强实时系统可调度性分析算法*

毛羽刚，张拥军，金士尧，胡华平

(国防科学技术大学 计算机学院，湖南 长沙 410073)

E-mail: maoyu@public.cs.hn.cn

http://www.nudt.edu.cn

摘要：Holistic算法是用于预测分布强实时系统可调度性的一种有用的方法.对该算法进行改进,扩展了原算法,使其更具普遍性.并且以一种高可靠、强实时、分布信息处理系统为研究背景,对有关应用实例进行了测试和分析.实验结果说明,改进后的算法更加准确、有效.

关键词：强实时;可调度性;时限;最大响应时间

中图法分类号：TP316 文献标识码：A

实时系统的正确性不仅依赖于任务计算结果的逻辑正确性,而且也依赖于任务的完成时间.强实时系统中的任务如果没有在规定的时限(deadline)内完成,则计算结果毫无意义,甚至引起严重后果(关键任务).实时调度为保证强实时任务满足时限要求起着重要作用.静态、固定优先级、可抢占调度算法由于具有简单、可预测等优点而获得了广泛的应用,常用于调度强实时周期任务,例如,单机系统中的RMS[1](rate-monotonic scheduling)算法、DMS[2](deadline-monotonic scheduling)算法、时间需求分析法[3]、忙周期分析法等[4].在分布环境下,实时任务通常由多个运行在不同节点上的子任务组成,由于子任务间存在着同步与通信关系而使得可调度性分析更加复杂.一个好的算法不但要满足任务的时限要求,而且任务的响应时间抖动要小.大的抖动通常需要大量的缓冲空间和处理机资源,会引起过度的系统设计.Holistic算法是由Tindell[5]提出的分布强实时系统的可调度性分析方法.其在线调度简单,开销低,但分析过程复杂,计算的最大响应时间抖动大.文献[6]引入时间偏移概念来减小或消除释放抖动,方法是当前一个子任务完成后,并不立刻释放后一个子任务,而是根据一定条件控制释放.文献[7]进一步给出了3种同步协议,用来管理子任务的释放和执行.使得各节点上的子任务均变成严格周期性的,从而消除抖动.但这需要增加相应的控制机制,调度复杂,并对系统时钟和网络要求高.本文从Holistic算法入手,从工程实际出发,引入了任务最小处理时间参数,拓展了原算法.通过用两种算法分析文献[8]中的系统实例,从而证明改进的算法合理、有效.对分布强实时系统的设计和验证具有一定意义.

1 Holistic算法

在讨论Holistic算法以前,首先给出有关的基本概念和定理.

* 收稿日期：1999-04-21；修改日期：1999-11-23

基金项目：国家"九五"国防预研基金资助项目

作者简介：毛羽刚(1964－)，男，上海人，博士，工程师，主要研究领域为实时及分布式计算机系统；张拥军(1971－)，男，湖南新邵人，博士，讲师，主要研究领域为实时，容错系统；金士尧(1937－)，男，江苏苏州人，教授，博士生导师，主要研究领域为计算机组织与系统结构，仿真，实时，分布式系统；胡华平(1966－)，男，江西临川人，博士，副研究员，主要研究领域为计算机系统性能评价，可靠性建模与分析.

定义 1. 一个端到端任务 T_i 由一系列串行执行的子动作$(a_{i,1},\dots,a_{i,j},\dots,a_{i,n})$组成,每个子动作可以是执行在某处理机上的子任务,也可以是网络子系统上的消息传递.每个子动作的最大响应时间 $R_{i,j}$是从第 1 个子动作开始到该子动作完成之间的时间间隔,最后一个子动作的最大响应时间就是该端到端任务的最大响应时间,如果该最大响应时间不超过端到端任务时限,则该任务就是可调度的.

定义 2. 对于端到端任务 $T_i=(a_{i,1},\dots,a_{i,j},\dots,a_{i,n})$,如果 $j<k$,则 $a_{i,j}$是 $a_{i,k}$的前趋子动作,如果 $k-j=1$,则 $a_{i,j}$是 $a_{i,k}$的直接前趋子动作.

定义 3. 在一个处理机的任务调度过程中,当且仅当在时刻 t 之前释放的优先级高于或等于 φ 的所有子任务实例(子任务的每次释放)都在时刻 t 之前完成,则时刻 t 为 φ 级空闲点.

定义 4. 处理机一直处于工作状态的最大时间间隔为系统忙周期;两个相邻 φ 级空闲点之间的非零时间间隔为 φ 级忙周期.

Holistic 算法在计算各节点子任务最大响应时间的时候,采用单机忙周期分析法[4],但必须考虑释放抖动的影响,因为除了第 1 个子动作以外,其他子动作的释放时间不会是严格按周期的,依赖于其直接前趋子动作的完成时间.由于 Holistic 算法认为所有子动作的最小处理时间可能为 0,所以一个子动作的最大释放抖动应该等于其直接前趋子动作的最大响应时间,$J_{i,j}=R_{i,j-1}$.因此,释放抖动的计算又依赖于响应时间.例如,接收主机上子任务的释放抖动依赖于消息的到达,消息的响应时间依赖于高优先级消息的响应时间,高优先级消息的响应时间又依赖于发送主机子任务的释放抖动.所以每个节点上的子任务的最大响应时间不可能从单机调度算法中直接求出,但可以通过一个递归过程联合求解.在每一步中,计算各子系统(包括处理机和网络)中的所有子动作的最大响应时间,所有释放抖动值在下一步计算前更新,其初始值可设为 0.当递归过程稳定时,计算停止,所有子任务的最大响应时间和释放抖动不再变化,或者至少发现一个子系统不可调度.

2 算法的改进

由于 Holistic 算法假定各子动作的最小处理时间为 0,使得子动作的最大释放抖动是其直接前趋子动作的最大响应时间.这会大大地增加后续子动作及低优先级子动作的最大响应时间.但在实际应用中,每个动作的执行时间不但存在最大值,也存在最小值,这可以通过分析实时程序的最好和最坏执行路径以及有关延迟因素来确定,不可能任意小.因此,每个子动作存在非零的最小响应时间,最大释放抖动应该等于其直接前趋子动作的最大响应时间与最小响应时间之差,最小释放抖动则可能为 0.设 $R_i(R'_i)$和 J_i 分别表示子系统 i 上所有子任务的最大(最小)响应时间和释放抖动的集合.IEERT(intermediate end-to-end response time)表示各节点最大响应时间的计算算法,则整个递归过程可表示为

$$\begin{cases} R_1^{(m+1)}=\mathrm{IEERT}(J_1^{(m)}) \\ R_2^{(m+1)}=\mathrm{IEERT}(J_2^{(m)}) \\ \quad\vdots \\ R_n^{(m+1)}=\mathrm{IEERT}(J_n^{(m)}) \end{cases}, \qquad \begin{cases} J_1^{(m+1)}=R_0^{(m+1)}-R'^{(m+1)}_0 \\ J_2^{(m+1)}=R_1^{(m+1)}-R'^{(m+1)}_1 \\ \quad\vdots \\ J_n^{(m+1)}=R_{n-1}^{(m+1)}-R'^{(m+1)}_{n-1} \end{cases}.$$

计算各节点子任务的忙周期分析法也应该作相应修改.根据文献[4]的有关思想和定理,下面重新给出在分布环境下,计算各单机子任务 $T_{i,j}$的最大响应时间 $R_{i,j}$的算法 IEERT:

输入:一组端到端周期任务及其子任务的属性,包括周期 p_i、最大执行时间 $\tau_{i,j}$和最小执行时间 $\tau'_{i,j}$,释放抖动 $J_{i,j}$.

输出:子任务的最大响应时间 $R_{i,j}$和最小响应时间 $R'_{i,j}$.

算法:对任意给定的子任务 $T_{i,j}$:

(1) 计算 $\Phi_{i,j}$级忙周期.

$$D_{i,j}=\min\left\{t>0\mid t=\sum_{T_{u,v}\in H_{i,j}}\lceil(t+J_{u,v})/p_u\rceil\tau_{u,v}\right\},$$

$$D'_{i,j}=\min\left\{t>0\mid t=\sum_{T_{u,v}\in H_{i,j}}\lceil t/p_u\rceil\tau'_{u,v}\right\}.$$

(2) 计算 $\Phi_{i,j}$级忙周期内任务 $T_{i,j}$释放的实例数.

$$M_{i,j}=\lceil(D_{i,j}+J_{i,j})/p_i\rceil,$$

$$M'_{i,j}=\lceil D'_{i,j}/p_i\rceil.$$

(3) m 从 1 到 $M_{i,j}$,m'从 1 到 $M'_{i,j}$.

(a) 计算实例 m,m'的完成时间.

$$c_{i,j}(m)=\min\left\{t>0\mid t=m\tau_{i,j}+\sum_{T_{u,v}\in H_{i,j}-\{T_{i,j}\}}\lceil(t+J_{u,v})/p_u\rceil\tau_{u,v}\right\},$$

$$c'_{i,j}(m')=\min\left\{t>0\mid t=m'\tau'_{i,j}+\sum_{T_{u,v}\in H_{i,j}-\{T_{i,j}\}}\lceil(t/p_u\rceil\tau'_{u,v}\right\}.$$

(b) 计算实例 m,m'的响应时间.

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

$$R'_{i,j}(m')=c'_{i,j}(m')+R'_{i,j-1}-(m'-1)p_i.$$

(4) 计算最大及最小响应时间.

$$R_{i,j}=\max\{R_{i,j}(1),\ldots,R_{i,j}(m)\},\quad 1\leqslant m\leqslant M_{i,j},$$

$$R'_{i,j}=\min\{R'_{i,j}(1),\ldots,R'_{i,j}(m')\},\quad 1\leqslant m'\leqslant M'_{i,j}.$$

$H_{i,j}$是优先级高于或等于并包括 $T_{i,j}$的子任务集.$T_{u,v}$是其中的子任务.$D_{i,j}$实际上是在$[0,t]$内完成 $T_{i,j}$释放的实例所需的时间,t 可从一个大于 0 的小数开始递归迭代,如果处理机利用率小于 1,则 t 收敛,$D_{i,j}$有界.$M_{i,j}$是$[0,t]$内 $T_{i,j}$释放的实例数.$R_{i,j}$是 $M_{i,j}$个实例响应时间中最大的一个.根据文献[1]中的临界带定理可知,该最大响应时间也是以后忙周期中释放实例的最大响应时间.$D'_{i,j}$,$M'_{i,j}$和 $R'_{i,j}$对应于子任务最小处理时间的计算.通过以上分析可知,Holistic 算法实际上是本文的算法在 $\tau'_{i,j}=0$ 时的一个特例.改进算法需要计算最小响应时间,所以稍为复杂一些,但算法是静态分析,并不影响系统的实际工作.

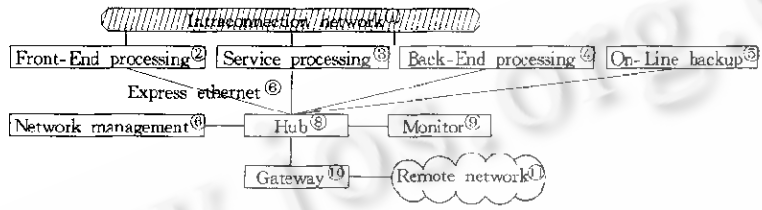
由于在网络子系统中,消息继承其发送子任务的优先级,因此,网络中的消息调度及最大响应时间的计算过程与处理机调度相似,不同的是需要加上消息因等待网络资源而引起的延迟时间.不同的网络采用的通信协议不同,由此引起的网络访问时间也不同.Tindell 在文献[5]中给出了在TDMA协议下消息最大响应时间的计算方法.本文在这方面不作具体讨论.

3 实例分析

本文用改进前后的 Holistic 算法对文献[8]中的分布强实时系统有关应用实例进行了可调度性分析.系统强实时部分由 3 台处理机组成,前端、服务和后端处理机,通过内部互连网络紧密耦合,主要处理外部强实时请求.由快速以太网连接的节点为弱实时部分,主要实现系统管理、业务管理、网络管理及监控、显示等工作.系统结构如图 1 所示.

系统中有 8 个端到端任务,每个端到端任务又由 3 个子任务组成,它们串行执行.由于内部互

连网络通信延迟是微秒量级,远远小于子任务响应时间,为研究方便起见,实验过程中忽略了通信时间.表1和表2是各节点上的子任务属性(子任务执行时间包括最大和最小执行时间)以及用Holistic及其改进算法求得的端到端任务的最大响应时间,其中任务优先级是根据实时请求的重要性而定的.本文同时还根据任务属性要求,通过操作系统提供的实时机制,用C语言实现了各任务程序,并运行和实测,测量值(测量10^6次后的平均值)见表2.实验数据说明,用改进后的算法计算出来的最大响应时间接近实测值,特别是对于低优先级任务改进的效果更好,证明了前面的分析的正确性.这有利于正确估计系统响应时间和系统设计.



①内部互连网络,②前端处理,③服务处理,④后端处理,⑤联机备份,⑥快速以太网,⑦网络管理,⑧集中器,⑨监控,⑩网关,⑪远程网.

Fig. 1 System structure
图1 系统结构

Table 1 Attribute of hard real-time tasks (ms)
表1 强实时任务属性 (ms)

Real-Time task attribute①				Execution time②		
Name③	Priority④	Period⑤	End-to-End⑥ deadline	Front-to-End processing⑦	Service processing⑧	Back-End processing⑨
Clock set⑩	1	17	100	1~2	1~3	1~2
First-Class request⑪	2	50	500	3~5	3~5	3~5
Second-Class request⑫	3	50	370	3~5	5~7	2~3
Third-Class request⑬	4	50	110	3~5	5~7	1~3
Fourth-Class request⑭	5	50	137	3~5	4~7	2~3
Fifth-Class request⑮	6	100	340	3~5	5~8	2~3
Inquiry⑯	7	200	500	3~5	3~5	2~3
Monitor⑰	8	200	500	3~5	3~5	2~3

①实时任务特性,②执行时间,③名称,④优先级,⑤周期,⑥端到端时限,⑦前端处理,⑧服务处理,⑨后端处理,⑩时钟校准,⑪一级请求,⑫二级请求,⑬三级请求,⑭四级请求,⑮五级请求,⑯查询,⑰监控.

Table 2 Maximal response time of end-to-end task (ms)
表2 端到端任务的最大响应时间 (ms)

Real-Time task①	Holistic	Improved Holistic②	Measured value③
Clock set④	7	7	6.4
First-Class request⑤	22	22	17.8
Second-Class request⑥	40	37	32.2
Third-Class request⑦	59	57	49.6
Fourth-Class request⑧	105	74	69
Fifth Class request⑨	144	125	123.8
Inquiry⑩	207	140	123.8
Monitor⑪	255	204	185.2

①实时任务,②改进的Holistic,③实测值,④时钟校准,⑤一级请求,⑥二级请求,⑦三级请求,⑧四级请求,⑨五级请求,⑩查询,⑪监控.

4 结 论

本文在深入研究分布强实时系统Holistic算法的基础上,对其不足之处进行了改进.通过对文献[8]中的实时应用端到端任务的最大响应时间的计算和测试,本文比较了两种算法的性能,证明改进的算法是准确、可靠的,对于分布强实时系统的合理设计具有一定的参考价值.

References:

[1] Liu, C.L., Layland, J. Scheduling algorithms for multiprogramming in a hard real-time environment. Journal of the ACM, 1973,20(1):46~61.

[2] Leung, J., Whitehead, J. On the complexity of fixed-priority scheduling of periodic real-time tasks. Performance Evaluation, 1982,2(4):237~250.

[3] Lehoczky, J., Sha, L., Ding, Y. The rate monotonic scheduling algorithm: exact characterization and average case behavior. In: Locke, D. ed. Proceedings of the 10th IEEE Real-Time Systems Symposium. Los Alamitos, CA: IEEE Computer Society Press, 1989. 166~171.

[4] Lehoczky, J. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In: Vista, L.B. ed. Proceedings of the 11th IEEE Real-Time Systems Symposium. Los Alamitos, CA: IEEE Computer Society Press, 1990. 201~209.

[5] Tindell, K., Clark, J. Holistic schedulability analysis for distributed hard real-time systems. Microprocessing and Microprogramming, 1994,50(2):117~134.

[6] Tindell, K. Using offset information to analysis static priority pre-emptively scheduled task sets. Technical Report, Department of Computer Science, University of York, 1992.

[7] Sun, Jun. Fixed Priority end-to-end scheduling in distributed real-time systems [Ph.D. Thesis]. University of Illinois, 1997.

[8] Hu, Hua-ping, Jin, Shi-yao, Wang, Wei. High reliability study and realization of distributed real-time systems. Computer Research and Development, 1998,35(9):841~845 (in Chinese).

附中文参考文献:

[8] 胡华平,金士尧,王维.分布式实时系统的高可靠性研究与实现.计算机研究与发展,1998,35(9):841~845.

An Improved Schedulability Analysis Algorithm of Hard Real-Time Distributed System*

MAO Yu-gang, ZHANG Yong-jun, JIN Shi-yao, HU Hua-ping

(*School of Computer, National University of Defence Technology, Changsha* 410073, *China*)

E-mail: maoyu@public.cs.hn.cn

http://www.nudt.edu.cn

Abstract: The holistic theory is a useful approach for assessing the schedulability of hard real-time distributed systems. In this paper, an improvement on the holistic algorithm is proposed. Original holistic analysis has been extended to make it more generalized. Some real-time applications are also tested and analyzed based on research background of an hard real-time distributed information processing system. The experimental result illustrates that the worst-case response time of each task generated by new algorithm is more accurate and valid.

Key words: hard real-time; schedulability; deadline; worst-case response time

* Received April 21, 1999; accepted November 23, 1999

Supported by the Defence Pre-Research Project of the National 'Ninth Five-Year-Plan' of China