混合关键度驱动的实时调度研究

张 奕[1,2]，程小辉[2]，蔡皖东[1]，詹 涛[1]

(1. 西北工业大学 计算机学院，陕西 西安 710072；2. 桂林理工大学 信息科学与工程学院，广西 桂林 541004)

摘 要：针对当前运行于不可预测开放环境下的嵌入式多使命复杂关键型系统，需要减少运行成本和处理不可预测工作负载情况的问题，文章提出一种混合关键度驱动的非对称式过载保护最小空闲调度策略。系统过载时，为共享同一处理器的不同关键度任务提供非对称式保护，禁止低关键度任务干扰高关键度任务，完全避免了传统的“关键度反转”问题。在恢复暂时阻塞的不同关键度任务时，在速率单调调度的基础上，引入关键度主导的截止期驱动动态调度策略，可使潜在处理器利用率达到 100%。实验结果表明，这种新算法的综合性能优于当前已有的混合关键度任务调度算法。

关 键 词：算法，关键度，计算机体系结构，分布式计算系统，反转问题，数学模型，实时系统，调度，混合关键度，速率单调调度，截止期驱动调度

中图分类号：TP311 **文献标识码**：A **文章编号**：1000-2758(2012)04-0617-05

关键型系统是指系统功能一旦失效即可能引起事故的系统[1]。随着关键型应用的日益广泛和微处理器技术的不断革新，关键型系统功能日趋复杂，开放环境下系统运行条件、输入工作负载具有高度的不确定性。同时，当前复杂关键型系统减少运行成本和所需物理资源的压力越来越大，各种不同关键度任务需要共享资源。传统方法需要提供最差情况下的资源需求(例如，Worst-Case Execution Time，WCET)。然而在不确定环境下获得 WCET 是一个挑战性的任务，并会导致两个问题：较差的平均处理器利用率和强制暂停运行时间超过 WCET 的任务。这种强制措施可能使高关键度任务不能抢占低关键度任务，而导致高关键度任务错过其截止期，出现“关键度反转”问题，危及关键型系统的安全。

传统解决“关键度反转”问题的方法是：按照任务关键度给任务分配优先级，称为“关键度作为优先级赋值”机制(Criticality As Priority Assignment，CAPA)[2]。当按这种简单关系赋值的优先级和最大化处理器利用率所需的任务优先级顺序相反时，就会引起优先级反转而导致更差的处理器利用率。本文提出一种混合关键度驱动的非对称式过载保护最小空闲调度(Asymmetric Overload Protected Shortest Slack Scheduling，AOLPSSS)算法，在系统过载时，为共享同一处理器的不同关键度任务提供非对称式保护，禁止低关键度任务干扰高关键度任务运行，完全避免了传统的“关键度反转”问题。在恢复暂时阻塞的任务时，在速率单调调度的基础上，引入关键度主导的截止期驱动动态调度策略，可使潜在处理器利用率达到 100%。

1 相关研究

当前传统实时调度算法分为 3 类：优先级驱动(Priority-Driven，PD)、共享驱动(Share-Driven，SD)和时间驱动(Time-Driven，TD)[3]。这 3 类调度方法分别适用不同的应用领域，分别针对特定的任务模型，都没有考虑复杂关键型系统中需要在同一调度器中集成调度各种不同关键度任务的需求问题。

基于速率的最早截止期优先(Rate-Based Earliest Deadline，RBED)[4]调度算法提出了一种动态优先级调度算法，利用松弛时间管理机制使不同类型应用并存于同一系统中。但此算法直接应用于关键

收稿日期：2011-09-20 基金项目：国家自然科学基金(61063001/F02020)及桂林理工大学科研启动费资助
作者简介：张 奕(1977—)，女，西北工业大学博士研究生，主要从事嵌入式关键系统及中间件研究。

型系统存在以下问题: ①对于非高关键度任务没有提供任何调度保证; ②在系统过载情况下不能保证按照任务的关键度进行 QoS 降级调整,这就极大地限制了系统的实时性能和应用范围。

RPDS(Rigorously Proportional Dispatching Server) [5] 是一种严格按比例派发服务器的混合实时调度算法。该算法将不同类型实时任务分开,由不同调度器调度,然后由 RPDS 统一派发时间片,在保证硬实时任务不受其它类型任务影响的基础上,使软实时任务的截止期错失率最小化。但是,在硬实时任务与非硬实时任务间是按比例共享的方式而不是按任务的重要性或时限要求调度,显然不适合关键型系统。

OCBP(Own Criticality-Based Priorities) [6] 是一种基于优先级调度的混合关键度调度算法,OCBP 考虑了最大化处理器利用率问题,但此算法有两个缺点: ①简单将任务关键度分为高低两种; ②不能完全避免系统过载情况下的关键度反转问题。

以上各种调度方法都不是完全适用于需要保证高可靠性的关键型系统。复杂关键型系统包含不同关键度的各种类型应用程序,要求具有混合调度各种不同任务模型的能力。

2 调度对象的模型定义及相关术语

本文所涉及的复杂关键型系统调度对象及其相关术语如下:

定义 1 任务(task) 是指完成某一特定功能的软件实体,是实时调度中的基本单位。任务在其生命期中的某一次执行称为该任务的一个作业(job) ,作业不允许在不同的处理器上执行。

定义 2 任务关键度(Criticality) 。任务 τ_i 的关键度 K_i 由以下两个参数确定: 平均请求处理成本和 QoS 参数。平均请求处理成本代表第 i 级 QoS 级的请求处理时间,用变量 O_i 表示。QoS 参数代表每个 QoS 级必须保证的平均延迟时间,用变量 M_i 表示,则 $K_i = \frac{O_i}{M_i}$。

从整个系统框架的角度来看,复杂关键型系统的调度对象具有以下的特性:

· 每个非周期性任务 $\tau_i(A_i, E_i, E_i^o, D_i, K_i)$ 具有 5 个时间属性: 其中 A_i 为任务的到达时间; E_i 为任务最坏情况下的计算时间,是任务在某节点上运行时计算时间的上限; E_i^o 为任务过载情况下所需的执行时间; 而 D_i 为任务的相对截止期限; K_i 为任务的关键度,K_i 值越小表示任务的关键度越高; 任务必须在($A_i + D_i$) 之前完成。

· 每个周期性任务 $\tau_i(A_i, E_i, E_i^o, P_i, K_i)$,其中 P_i 为任务的周期。

· 各任务之间是相互独立的,调度对象的每个任务对共享资源的访问方式为互斥访问。

· 任务是可抢占的,当较高关键度任务达到时,能抢占较低关键度任务的处理器控制权。

· 周期性任务的绝对截止期限与其每个周期结束点相同。

3 混合关键度驱动的非对称过载保护最小空闲调度策略

3.1 任务的接纳原则

复杂关键型系统的任务调度策略与资源分区管理思想相结合,以实现最佳的系统行为 [2] 。在多任务分配阶段,任务根据一定的标准划分成组,每个处理器维护一个任务调度队列。在单一处理器内部调度阶段,如果调度器接纳某一任务 τ_i,则当没有比 τ_i 关键度更高的任务时,调度器必须保证 τ_i 在 E_i^o 时间期限内完成。

基于此任务接纳控制原则,AOLPSSS 算法调度策略分成 2 种情况: ①当系统没有过载情况出现时(Normal Mode, NMode) ,按最大化处理器利用率的原则调度任务,尽量延迟高关键度任务的执行时间; ②当系统出现过载情况时(Overload Mode, OLMode) ,保证最高关键度的任务在其过载执行预算 E_i^o 的时间期限内完成。

3.2 AOLPSSS 算法

1) 进入 OLMode 最小空闲计算方法

在系统过载条件下,为了避免关键度反转,必须保证高关键度任务在其截止期之前完成,因此,高关键度任务可能需要抢占低关键度任务的处理器运行时间片。假设任务 NMode 和 OLMode 下的运行时间分别用 E_i^n 和 E_i^{ol} 表示,任务进入 OLMode 的最晚时间用 SL_i 表示。在最小空闲运行时间的前提下,利用启发式方法计算 E_i^n、E_i^{ol} 和 SL_i。具体算法如下

Algorithm1 GetLastSlackInstant($i, E_i, E_i^o, D_i, K_i, SL_i$) : Enter OLMode Last Instant Calculate Algorithm

Input: task set $\Gamma = \{\tau_1, \tau_2, \cdots, \tau_n\}$, E_i, E_i^o, D_i, K_i.

Output: SL_i.

1. /* 假设任务按 K_i 值降序排列,且计算范围在任务集合之内 * /
2. $E_i^n = 0$; $E_i^{ol} = E_i^o$;
3. while($E_i^n \leqslant E_i^0$ && $E_{i-1}^o + E_i^n + E_{i+1} \leqslant D_{i-1}$) {
4. $SL_i = D_i - E_i^{ol}$;
5. $I_i = max((SL_i - E_{i-1}^o - E_{i+1}), 0)$;
6. $E_i^n = I_i$;
7. $E_i^{ol} = E_i^o - E_i^n$;
8. }
9. if(K_i is the highest criticality)
10. $SL_i = D_i - E_i^{ol}$;
11. else
12. $SL_i = \min(E_{i-1}^o + E_i^n, D_i - E_i^{ol})$;
13. return SL_i;

2) AOLPSSS 算法实现

速率单调调度(Rate Monotonic, RM) 算法被证明是固定优先级调度方案中的最优算法[7],本文提出的 AOLPSSS 算法作为元调度器,在 RM 调度器的基础上给任务的优先级赋予初始值。由任务之间处理器时间抢占原则可以看出,尽量延长任务进入 OLMode 的时间,可以最大化处理器利用率。

AOLPSSS 算法与文献[2]调度思想不同的是:当重新调度被暂时阻塞的任务时,不是按栈方式,而是引入截止期驱动[8](Deadline Driven, DD) 动态调度的思想,按关键度高低顺序,选择剩余时间足够任务执行完的作业重新恢复执行,可使潜在的处理器利用率达到 100%。具体算法如下所示。

Algorithm1 AOLPSSS Algorithm

1. /* 假设任务按 K_i 值降序排列,且计算范围在任务集合之内* /
2. for ($i = 1$ to n) {
3. GetLastSlackInstant($i, E_i, E_i^o, D_i, K_i, SL_i$);
4. }
5. for($i = 1$ to n) {
6. if(task is suspended task) {
7. for($j = i - 1$ to 1) {
8. if($E_j^0 - E_j^n < D_j - (SL_i + E_i^{ol})$)
9. resumption τ_j;
10. }
11. resumes other tasks in the order of RM;
12. }
13. else{
14. if(task state is NMode && $\text{Time}_{\text{current}} < SL_i$) {
15. excute task E_i^n time in the order of RM priority;
16. }
17. if(task state is OLMode) {
18. excute task E_i^{ol} time in the order of criticality;
19. }
20. }
21. }

4　性能评价

4.1　形式化评价方法

当关键型系统出现过载情况时,需要保证最关键应用能满足其截止期要求。为了证明 AOLPSSS 算法能满足此项重要要求,引入任务负载描述矩阵 W(Workload Matrix) 和任务截止期保障矩阵 D(Deadline Guarantee Matrix) 评价 AOLPSSS 算法性能。不同关键度任务的各种负载情况用 W 表示,W 的元素等于 0 表示任务正常执行,W 的元素等于 1 表示任务过载。W 的不同列表示不同关键度任务的负载情况。假设系统任务不同关键度个数为 k,则系统不同关键度任务所有可能负载情况有 2^k 种。D 中不同行元素与 W 相对应,表示不同关键度任务在 W 负载情况下,任务满足其截止期的情况。D 中元素的值用一个分数表示,分子值表示评价时间内满足其截止期的任务执行的作业数,分母值表示评价时间内任务需要执行的作业总数。

不失一般性,假设任务按关键度值降序排列,即 $\forall i < j, K_i > K_j$。为了利用 D 对算法性能进行定量地评价,引入映射函数 P_d 将 D 转化为标量计算[9]

$$P_d(\boldsymbol{D}) = \sum_{c=1}^{k} \left\{ \frac{1}{2^{k-c}} \frac{\sum_{r=0}^{2^k-1} d_{r,c}}{2^k} \right\} \quad (1)$$

在映射函数 P_d 引入权重因子 $\frac{1}{2^{k-c}}$，区别表示 D 中不同列任务的重要程度。当在所有可能任务负载情况下，任务的所有作业均满足其截止期，则可得到 P_d 的理论最大值

$$\max(P_d) = \sum_{c=1}^{k} \frac{1}{2^{k-c}} = 2 - \frac{1}{2^{k-1}} \quad (2)$$

因此，为了清晰比较不同调度算法的性能，引入 P_d 规格化表示因子 γ，$\gamma = \frac{P_d(D)}{2 - \frac{1}{2^{k-1}}}$，公平比较调度算法在不同关键度任务集下的 γ 值，γ 值越大表示该调度算法综合性能越好。

4.2 算法性能比较结果

当前针对混合关键度任务调度的研究成果不多，本次试验采用表 1 的示例任务集，在相同环境下将 AOLPSSS 算法和 OCBP 调度算法[6] 进行比较。改变任务集关键度向量值（$< K_{\tau_1}, K_{\tau_2}, K_{\tau_3} >$），观察 AOLPSSS 算法和 OCBP 算法在不同关键度向量值下的调度结果。为使本次试验结果更具参考价值，同时将传统避免关键度反转的 CAPA 算法[2] 调度结果参与比较。

表 1 AOLPSSS 调度任务集

Task	E	E^o	D	K	RM Priority PRI	E_i^n	E_i^{ol}	SL
τ_1	1	5	10	2	0	3	2	8
τ_2	2	10	20	1	1	1	9	6
τ_3	4	20	40	0	2	5	15	25

为了公平比较 AOLPSSS、OCBP 和 CAPA 三种算法对于关键度反转问题的解决程度，将调度结果转化为标量 γ 显示，具体的测试结果如图 1 所示。

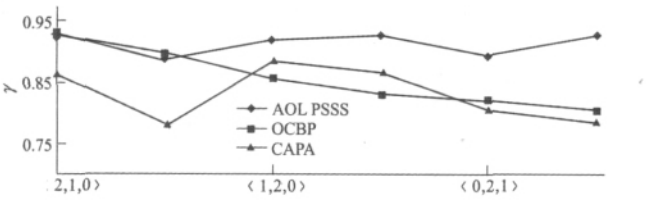

图 1 AOLPSSS、OCBP 和 CAPA 算法 γ 值比较

图 1 的调度结果清晰地显示 3 种算法的性能区别，在任务不同的关键度向量值下，本文提出的 AOLPSSS 算法综合性能要优于 OCBP 和 CAPA 算法。

5 结 论

本文重点研究了在单一处理器上集成调度不同关键度任务的 AOLPSSS 调度策略，尽量延迟高关键度任务进入过载运行模式的时间，并在恢复阻塞任务时结合 DD 动态调度的思想，最大化降低了调度成本，并避免了关键度反转问题。下一阶段的研究任务是，在多任务分配的全局调度阶段，结合装箱算法思想，提出一种关键度可感知的多处理器负载均衡算法，尽量减少复杂关键型系统所需处理器个数，进一步减少复杂关键型系统的运行成本。

参考文献:

[1] Erl T. Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall, 2005

[2] Niz D D, Lakshmanan K, Rajkumar R R. On the Scheduling of Mixed-Criticality Real-Time Task Sets. Proceedings of the Real-Time Systems Symposium, 2009, 291-300

[3] 谭朋柳，金 海，张明虎. 用于开放式系统的二维优先级实时调度. 电子学报，2006，34(10)：1773-1777
Tan Pengliu, Jin Hai, Zhang Minghu. Two-Dimensional Priority Real-Time Scheduling for Open Systems. Acta Electronica Sinica, 2006, 34(10): 1773-1777 (in Chinese)

[4] Brandt S A, Banachowski S, Lin C, Bisson T. Dynamic Integrated Scheduling of Hard Real-Time, Soft Real-Time and Non Re-

al-Time Processes. Proceedings of the 24th IEEE RTSS, 2003

[5] 龚育昌, 王立刚, 陈香兰等. 一种严格按比例派发服务的混合实时调度算法. 软件学报, 2006, 17(3): 611-619
Gong Yuchang, Wang Ligang, Chen Xianglan, et al. A Hybrid Real-Time Scheduling Algorithm Based on Rigorously Proportional Dispatching of Serving. Journal of Software, 2006, 17(3): 611-619 (in Chinese)

[6] Li H, Baruah S. Load-Based Schedulability Analysis of Certifiable Mixed-Criticality Systems. http://www.cs.unc.edu/~baruah/Pubs.shtml, 2010

[7] Liu C, Layland J. Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. Journal of the ACM, 1973, 20(1): 46-61

[8] Lakshmanan K, Niz D, Rajkumar R, et al. Resource Allocation in Distributed Mixed-Criticality Cyber-Physical Systems. Proceedings of the 30th International Conference on Distributed Computing Systems (ICDCS), 2010, 169-178

An Effective Algorithm of Mixed-Criticality Driven Real-Time Scheduling

Zhang Yi[1,2], Cheng Xiaohui[2], Cai Wandong[1], Zhan Tao[1]

(1. Department of Computer Science and Technology, Northwestern Polytechnical University, Xi'an, 710072 China
2. College of Information Science and Engineering, Guilin University of Technology, Guilin, 541004, China)

Abstract: In many cases unexpected workload spikes are likely to occur due to unpredictable changes in the physical environment. In this paper we present a mixed-criticality driven asymmetric overload-protected shortest slack scheduling algorithm that implements an alternative protection scheme to avoid the criticality inversion problem. Sections 1 through 4 of the full paper explain our algorithm mentioned in the title, which we believe is new and effective and whose core consists of: "This algorithm can be used with rate monotonic based preemptive scheduler with deadline driven scheme to resume the blocked tasks. The potential processor utilization of the new algorithm can reach 100%. Section 1 briefs relevant past research. Section 2 deals with the model of the objects to be scheduled and relevant semantics. Section 3 deals with the problem of criticality inversion. Section 4 deals with our algorithm mentioned in the title." The experimental results, presented In Figs. 2 through 5 and Table 2, show preliminarily that our new algorithm provides performance higher than those of two existing algorithms for mix-criticality task scheduling.

Key words: algorithms, computer architecture, distributed computing systems, inverse problems, mathematical models, real time systems, scheduling; mixed-criticality, rate monotonic scheduling, deadline driven scheduling