高可信赖实时操作系统的防危调度机制

杨仕平，桑 楠，熊光泽，刘校矢

(电子科技大学计算机科学与工程学院 成都 610054)

【摘要】为增强实时操作系统的防危性，在分析现有调度机制的基础上，探讨了最大关键度优先的调度算法，该算法是一种混合型的优先级实时调度算法，由静态优先级、动态子优先级和静态子优先级3部分组成，综合了固定优先级调度算法和动态优先级调度算法的优点，既可充分利用处理器资源，又能在发生瞬时过载时保证关键任务不受非关键任务的影响，从而增强了实时操作系统的防危性。

关 键 词 高可信赖； 关键度； 防危性； 调度算法； 实时操作系统
中图分类号 TP302.8 文献标识码 A

A Safety Scheduling Mechanism of High Assurance Real Time Operating System

YANG Shi-ping，SANG Nan，XIONG Guang-ze，LIU Xiao-shi

(School of Computer Science and Engineering, UEST of China Chengdu 610054)

Abstract To buildup the safety of real time operating system, after status quo of existing scheduling mechanism analyzed, brought forward a kind of novel safety scheduling mechanism based on Maximum Criticality First (MCF). MCF is a hybrid priorities real time scheduling algorithm which consisting of three parts, the first two parts are the assignment of the importance and user priority, which is done statically, the second part is the assignment of the dynamic priority based on minimum laxity first scheduling algorithm. MCF synthesizes the strongpoint of the fixed priority scheduling algorithm and the dynamic priority scheduling algorithm. By this way, MCF not only takes full advantage of resources such as CPU, also make critical task not suffer from other non critical tasks at transient overload. The safety of real time operating system is improved with MCF.

Key words high assurance; criticality; safety; scheduling algorithm; real time operating system

实时操作系统作为一种重要的系统支撑软件，主要功能是采用适当的调度机制使尽可能多的任务在死限(deadline)到达之前完成计算，其可靠性是整个安全关键系统(Safety Critical System，SCS)可靠性的决定性因素。由于SCS一般包含多个任务，任务的个数在整个系统的运行期间随着应用环境的变化而变化。同时，由于处理器的处理能力有限，因而在某一时刻会出现任务量瞬时过载的情况，会有一些任务不可避免地错过其死限。但是，由于SCS中的各任务并不都具有相同的关键度，即一些任务的重要程度会大于另外一些任务的重要程度，因而过载时的首要任务是保证关键任务不错过死限，其次才是保证尽可能多的任务不错过死限。然而，现有的实时调度机制没有考虑各任务的关键度，也没有相应的过载处理机制。本文将在研究现有实时调度机制的基础上，探讨适合于安全关键实时操作系统的防危(safety)调度新机制。

1 现有的实时调度机制

实时调度一般可分为静态调度机制与动态调度机制两类。典型的静态调度算法为速率单调调度(Rate Monotonic Scheduling，RMS)算法、最早死限优先(Earliest Deadline First，EDF)调度算法和最小松弛度优先(Minimum Laxity First，MLF)调度算法。本文首先对以上3种实时调度算法作简单介绍。

1.1 速率单调调度(RMS)算法

RMS算法的缺点是处理器的利用率较低。文献[1]计算出了使用RMS算法调度n个周期任务的最坏可调度

收稿日期：2003 – 03 – 10
基金项目：国家十五项目(41315040106)；国防科研基金
作者简介：杨仕平(1974–)，男，博士，主要从事实时操作系统防危核机制与实现方面的研究.

界限为$W_n = n(2^{1/n} - 1)$，且有$W_\infty = 69\%$。设U_n为n个周期性任务的总处理器利用率[2]，当$U_n \le W_n$时，所有的任务都将满足其死限。如果$U_n > W_n$，则存在一个最高优先级任务子集S，当满足$U_s \le W_s$时，任务子集S中的各任务都将满足其死限，便可形成由关键任务组成的“关键集合”[2]。

1.2 最早死限优先调度(EDF)算法

EDF调度算法属于动态调度算法，能很好地支持周期动态变化的任务[3]，但也存在如下一些缺点：

(1) 不能预测任务计算的成功性

可能某任务i不能在其死限内完成其计算，但在执行前并不能预测该任务的计算即将失败，只有任务i的死限真正到达时才能发现。

(2) 瞬时过载时的非确定性

在瞬时过载的情况下，不能形成由关键任务组成的“关键集合”，即没有任何措施保障关键任务的成功运行。

1.3 最小松弛度优先调度(MLF)算法

MLF调度算法不仅考虑任务的死限D_i[4]，而且要考虑任务的运行时间C_i。松弛度定义为t_{laxity}=死限时间D_i－运行时间C_i。然而，在任务量瞬时过载的情况下，MLF调度算法不能控制任务的失败与成功，因而也不能保证关键任务成功地完成计算[5]。

由上可知，RMS不适合SCS状态变化的需要，而EDF，MLF在瞬时过载的情况下不能预测性地控制任务的失败或成功，都缺乏防危性。另外，纯动态调度策略将导致过多的CPU开销，存在使所有任务错过死限的危险。

综上所述，可见现有的实时调度算法都不能很好地应用于安全关键实时操作系统，研究新的防危调度机制是十分必要的。

2 基于MCF的防危调度机制

研究具体的防危调度机制之前，首先定义任务i在其死限D_i到达之前完成该任务计算的价值为关键度(criticality)。如在核电安全关键系统中，负责紧急停堆的任务τ_{shutdown}的关键度总是大于负责控制电能输出任务τ_{output}的关键度，且任务τ_{shutdown}的周期远大于任务τ_{output}的周期。当使用静态调度RMS算法时，在需要紧急停堆的情况下，由于任务τ_{shutdown}的优先级较低，所以任务τ_{shutdown}并不能立即抢先系统中的其他任务而优先执行。在任务量瞬时过载的情况下[6]，当需要紧急停堆时，由于任务τ_{shutdown}的优先级较低，不能进入“关键集合”，任务τ_{shutdown}运行的成功性得不到保障，因而可能导致核电关键系统酿成灾难性事故。动态调度EDF、MLF算法因过载时的非确定性，致使这两种调度机制也缺乏防危性。本文所研讨的最大关键度优先(Maximum Criticality First，MCF)调度机制可以很好地解决以上调度算法的不足之处。

MCF是一种混合性的优先级调度机制，关键度由静态优先级和动态优先级两部分组成。静态优先级分为两种，一种称为重要度，其优先权较任务的动态优先级高；另一种称为用户优先级，其优先权较任务的动态优先级低，如图1所示。动态优先级反比于任务的松弛度t_{laxity}。任务的关键度由n bit值组成，其中i bit为任务的重要度，d bit为动态优先级，u bit为用户优先级，且i、d和u的值都大于或等于$\ln_2 N_{\text{task}}$，N_{task}为最大任务数，则重要度、动态优先级和用户优先级的范围分别为0～$(2^i - 1)$、0～$(2^d - 1)$与0～$(2^u - 1)$。

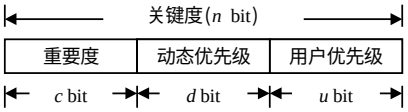

图1 MCF调度算法的组成

由前述可知，MCF调度算法由两部分组成，第一部分分配重要度与用户优先级，该部分可事先确定；第二部分根据任务的松弛度计算动态优先级，然后选择具有最大关键度的任务执行。重要度与用户优先级的确定步骤为：

1) 与RMS调度算法一样，从最短周期到最长周期对任务进行排序。

2) 定义已排序的前N个任务为“关键集合”，并使其总的最坏处理器利用率不超过100%。如果某关键任务由于具有较长的周期而没有进入该关键集合，则应该：(1) 设该关键任务的周期为$T_{\mathrm{CriticalTask}}$，把该周期为$T_{\mathrm{CriticalTask}}$的关键任务分割为周期都为$T_{\mathrm{CriticalTask}}/2$的两个子任务；(2) 经过步骤2)的(1)后，如果关键任务已进入关键集合，则进行步骤3)，否则把该关键任务分割为3个或3个以上的周期相同的子任务，重复该步骤直到该关键任务进入关键集合为止。

3) 关键集合中的任务分配较高的重要度，而关键集合之外的任务分配较低的重要度。

4) 对系统中的各任务分配唯一的用户优先级。无论何时某任务进入就绪队列，都要进行重新调度操作。

MCF调度器选择即将执行的任务的原则为：(1) 选择具有最大重要度的任务；(2) 如果两个或多个任务的重要度相同，则选择具有最高动态优先级(最小松弛度)的任务；(3) 如果两个或多个任务的重要度相同，且具有相同的动态优先级，则选择具有最高用户优先级的任务；(4) 如果两个或多个任务的重要度、动态优先级、用户优先级都相同，则按先来先服务的方式选择任务。

3　MCF调度算法的防危性

由上可知，MCF调度算法是一种混合性的优先级调度算法，任务的关键度由静态优先级、动态子优先级和静态子优先级3部分构成，各部分在调度中的特权程度依次递减，静态优先级按重要度分配，动态子优先级按MLF算法(或EDF算法)分配，静态子优先级应事先指定。MCF算法综合了固定优先级调度算法和动态优先级调度算法的优点，既可充分利用处理器资源，又能在发生瞬时过载时保证关键任务不受非关键任务的影响。为了证明MCF调度算法的防危性，假设有由4个周期性任务组成的任务集，且其中每个任务的死限等于其周期，如图2所示。

任务	表示	关键度	周期	死限	计算时间	利用率%
T_1		高	6	6	2	33
T_2		高	10	10	4	40
T_3		高	12	12	3	25
T_4		低	15	15	4	27

图2　4个任务组成的任务集及各任务的属性

由图2可知，4个任务总的最坏利用率为125%=33%+40%+25%+27%，已超过CPU的最大利用率100%，这种情况即为过载。此时，某些任务将错过其死限。当使用RMS算法调度此任务集时，只有任务T_1和T_2在关键集合中，这意味着将保证任务T_1和T_2不会错过其死限，但任务T_3和T_4将错过其死限，如图3所示。

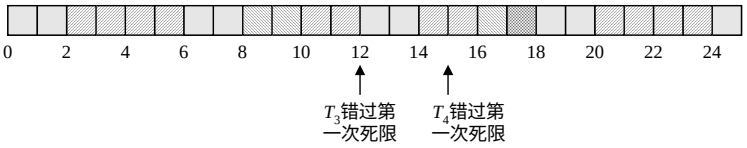

图3　过载时使用RMS调度算法调度任务集的情况

当使用EDF算法调度此任务集时，任务T_1将错过第4次死限，而任务T_2将错过第2次死限，如图4所示。可见，EDF调度算法不能在任务量瞬间过载时预测任务的失败与成功。

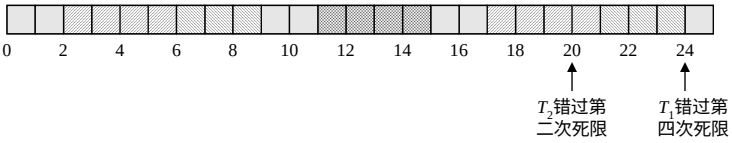

图4　过载时使用EDF调度算法调度任务集的情况

由图2可知，4个任务中的任意3个的总的最坏处理器利用率都小于100%，尽管EDF调度算法的最大CPU利用率可达100%，还是有任务T_1与T_2错过其死限，如图4所示。这是由于EDF调度算法缺乏预测性所造成的。当使用最大关键度优先调度MCF算法时，按最小周期到最大周期对任务排序后，由于任务T_1, T_2和T_3总

的最坏利用率为98%，小于MCF调度算法可能达到的最大CPU利用率100%，因此任务 T_1, T_2 和 T_3 将形成关键集合，此时只有任务 T_4 在关键集合之外，任务 T_4 将错过其死限，如图5所示。

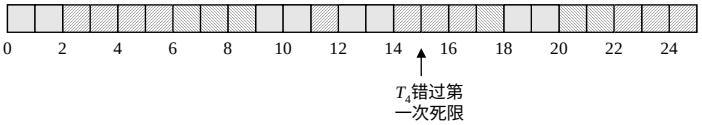

图5 过载时使用MCF调度算法调度任务集的情况

由前可知，当使用RMS调度算法时，只有任务 T_1 和 T_2 处于关键集合中；而使用MCF调度算法，除了任务 T_1 和 T_2 处于关键集合中外，任务 T_3 也处于关键集合中，将保证任务 T_3 不会错过其死限，即MCF调度算法增大了关键集合的可调度边界。通过比较图5与图4又可以看出，使用MCF调度算法时，能更加准确地预测系统的行为。另外，当使用EDF调度算法时，在任务量瞬时过载的情况下，任务 T_1 与 T_2 都有可能错过其死限，而使用MCF调度算法时，则只有任务 T_4 错过其死限，可在增大可调度任务数的同时使系统的行为更可预测。通过以上比较可以说明，MCF调度算法确实具有一定的防危性。

4 结 束 语

在现有实时调度算法中，无论是RMS、EDF，还是MLF，都未考虑各任务在整个SCS中的重要程度。在任务量瞬时过载的情况下，RMS算法虽然能形成由高优先级任务组成的“关键集合”，但关键集合的可调度边界在最坏情况下只有69%，平均情况下也只有88%，处理器的利用率较低。此外，关键任务不一定能进入关键集合。而EDF和MLF调度算法尽管都大大提高了处理器的利用率，但在任务量瞬时过载的情况下，都缺乏一定的可预测性[7-8]，关键任务的成功计算得不到保障。因此，RMS、EDF和MLF调度算法都不能很好地用于安全关键实时操作系统中。本文所研究的最大关键度优先调度MCF算法是一种混合性的优先级调度算法，它综合了固定优先级调度算法和动态优先级调度算法的优点，既可充分利用处理器资源，又能在发生瞬时过载时保证关键任务不受影响。

参 考 文 献

[1] Liu C L, Layland J W. Scheduling algorithms for multiprogramming in a hard real time environment[J]. Journal of the Association for Computing Machinery, 1973, 20(1): 44-61.

[2] Lehoczky J, Sha L, Ding Y. The rate monotonic scheduling algorithm: exact characterization and average case behavior[C]. Proceedings 10th IEEE Real-Time Systems Symposium, Santa Monica, CA, 1989.166-171.

[3] Sprunt B, Sha L, Lehoczky J. Aperiodic task scheduling for hard real-time systems[J]. Journal of Real-Time Systems, 1989, 1(1): 27-60.

[4] Stewart D B, Schmitz D E, Khosla P K. Implementing real-time robotic systems using CHIMERA II[C]. Proceedings of 1990 IEEE International Conference on Robotics and Automation, Cincinnatti, OH, 1992.

[5] Stewart D B, Khosla P K. Real-time scheduling of sensor-based control systems[A].Real-Time Programming[M]. NY: Pergamon Press, 1992.

[6] Sanjoy K B, Jayant R H. Scheduling for overload in real time system[J]. IEEE Transaction, 1997, 46(9): 1 034 -1 039.

[7] Alvarez P M, Melhem R, Daniel M. An incremental approach to scheduling during overloads in real time systems[J]. IEEE Real-Time Systems, 2000, 10(1): 283-293.

[8] Hansson J, Thuresson M, Son S H. Imprecise task scheduling and overload management using OR-ULD[J]. IEEE Real-Time Systems, 2000, 12(2): 307-314.

编 辑 熊思亮