面向安全关键实时应用的分层防危调度算法研究

夏 平 周兴社
(西北工业大学计算机学院 西安 710072)

摘 要 针对现有防危调度算法在软硬件失效情况下防危能力不足的问题,具体进行了以下工作:构建了一种分层防危实时调度模型,该模型从功能组件和安全分区两方面描述了安全关键实时应用的防危性需求,并给出一种基于分层调度思想的三级防危调度器框架。以该模型和框架为基础,提出了一种新的分层防危调度算法(HSS),该算法对安全关键实时应用中不同关键度的功能组件采用空间隔离机制,对同一功能组件内的不同分区采用时间隔离机制,兼顾实现了时空隔离的防危效果。仿真实验结果表明,HSS算法与其他同类算法相比,在防危效果和应用负载承受能力方面具有较好的表现。

关键词 安全关键,实时,分层,调度,算法
中图法分类号 TP301 **文献标识码** A

Hierarchical Safeguard Scheduling Algorithm for Safety Critical Real-time Application

XIA Ping ZHOU Xing-she
(Department of Computer, Northwestern Polytechnical University, Xi'an 710072, China)

Abstract The paper solved the problem that current popular safeguard scheduling algorithm cannot achieve the safeguard function under the environment of software and hardware failure. It built a new hierarchical real-time scheduling model which describes the safety requirement of safety-critical real-time application from two aspects including function component and safe partition, and designed a three-level safeguard scheduler framework. Based on the model and framework, the paper proposed a new hierarchical safeguard scheduling algorithm (HSS) which achieves spacial separation effect by distributing function components with different critical degrees to different physical processor clusters, and attains temporal separation effect by activating various partitions running on the same processor in a fixed cycle. Empirical investigations show that the improvements in the safeguard performance and the endurance to different application loads can be achieved by choosing HSS than other similar algorithms.

Keywords Safety critical, Real time, Hierarchical, Scheduling, Algorithm

1 引言

随着微电子技术和网络技术的不断发展,安全关键实时系统(Safety Critical Real Time System)被广泛应用于航空航天、交通运输、国防军事等重要领域。这类系统不同于一般的实时系统,它一旦发生失效将会造成巨大的生命财产损失,甚至引发灾难,因此必须在系统设计中充分考虑防危性。传统的防危性设计分为两类:一类是空间隔离方式,即为每个功能组件提供专用计算和存储资源,彼此之间实现物理隔离。虽然这种方式实现了很好的防危效果,但成本耗费巨大,资源浪费较为严重。另一类是时间隔离方式,即多个功能组件共享系统计算和存储资源,在运行时间上彼此相互错开,每个功能组件只在固定的时间片内运行,以防止某个功能组件长时间占据系统资源。但这种时间隔离方式也存在缺陷,一方面某个功能组件上携带的病毒或恶意程序可能通过共享资源间接传播到安全关键高的功能组件,从而破坏整个系统的防危性;另一方面,硬件故障可能导致整个应用的运行崩溃。另外,随着多核技术和分布式技术的发展,大多数实时系统开始采用多处理机架构,由此所带来的计算优势无法在单机运行的时间隔离机制上得到体现。因此,克服以上两种防危设计方式的缺点,研究面向安全关键实时应用的新型防危调度算法,具有较强的实际意义。

目前关于防危调度算法的研究已经取得了许多成果。文献[1,2]最早提出了分层调度(hierarchical scheduling)方法,其基本思想是按照功能不同将实时应用分为多个分区(或子系统),并将每个分区视为一个独立任务,对其进行全局调度(global scheduling),而在分区内部对所有属于该分区的实时任务进行局部调度(local scheduling)。文献[3,4,8,9]对以上方法进行了不同程度的改进。文献[5]研究了硬实时系统在强分区约束下的双层分区调度问题,给出了一种与分区利用率匹配的分区设计方法。文献[6,7]以安全关键实时应用为对象,研究了两级时间隔离分区保护机制,并给出了分区参数

到稿日期:2012-02-27 返修日期:2012-07-16 本文受国家自然科学基金(60736017)资助。
夏 平(1982—),男,博士生,工程师,CCF学生会员,主要研究方向为高性能计算、实时计算,E-mail:sharkping@gmail.com;周兴社(1955—),男,教授,博士生导师,CCF高级会员,主要研究方向为高性能计算、嵌入式计算、普适计算。

的设计方法。以上文献均只针对如何实现合理的分区方式展开研究，调度对象仅限于在单处理机上运行的硬实时周期任务，而前文所提出的各种防危方式存在的缺陷仍然没有得到有效解决。

针对以上算法存在的不足，本文提出了一种分层防危调度算法 HSS(Hierarchical Safeguard Scheduling Algorithm)，该算法基于分层调度的思想，兼顾采用时空隔离机制，克服传统单种防危方式的缺陷，并充分利用多处理机优势来提高调度性能。仿真实验结果表明，HSS 算法在多种失效情况下的防危效果以及对于不同应用负载的承受能力方面有较好表现。

2 分层防危调度模型

2.1 问题描述及假设

本文的研究对象是由多个功能组件(或子系统)组成的安全关键实时应用，其中每个功能组件包含多个实时任务，这类应用既要满足实时性需求，还必须满足防危性需求，即不同关键度的功能组件以及每个功能组件内不同关键度的实时任务必须实现防危隔离，其中关键度是指实体运行失效可能对系统造成的影响程度。发生失效实体的关键度越高，对系统造成的影响也越大。基于以上因素考虑，这类实时应用的调度问题可以归结为：如何调度各功能组件及其包含的实时任务，使之运行满足实时性和防危性双重需求。

2.2 分层防危调度模型

本文提出的分层防危调度模型如图 1 所示，该模型具体包含 3 个部分：实时防危组件模型、处理机群集模型和分层防危调度器，其细节介绍如下。

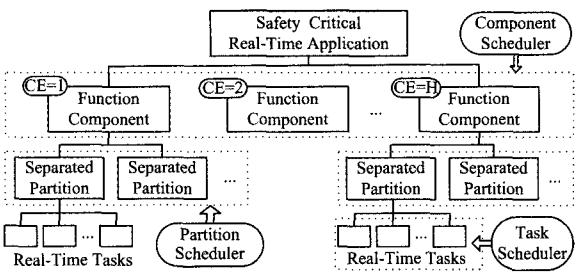


图 1 分层防危调度模型

2.2.1 实时防危组件

假设安全关键实时应用按照关键度的不同分为 N 个功能组件(function component)，记为 $CP=\{cp_1, cp_2, cp_N\}$。每个功能组件 cp_k 用多元组(CE_k)来表示，其中 CE_k 表示 cp_k 的关键度。对于不同 CE_k 的组件，其运行时需采取相应的防危措施。

对于任意功能组件 $cp_k \in CP$，其内部所包含的实时任务按照关键度的不同划分为多个独立的安全分区，用 $SP(cp_k)=\{sp_1^k, sp_2^k, \cdots\}$ 来表示，其中 sp_i^k 表示组件 cp_k 的某个安全分区。为了满足防危性需求，这些安全分区在运行时也必须采取相应的防危措施。

假设安全分区 $sp_i^k \in SP(cp_k)$ 所包含的任务集合记为 $\Gamma(sp_i^k)=\{\tau_{i,1}^k, \tau_{i,2}^k, \cdots\}$。对于任意实时任务 $\tau_{i,i}^k \in \Gamma(sp_i^k)$，用多元组($C(\tau_{i,i}^k)$, $D(\tau_{i,i}^k)$, $R(\tau_{i,i}^k)$, $B(\tau_{i,i}^k)$, $P(\tau_{i,i}^k)$)来表示，其中 $C(\tau_{i,i}^k)$ 表示任务的最大运行时间，$D(\tau_{i,i}^k)$ 表示任务的运行时限(deadline)，$R(\tau_{i,i}^k)$ 表示任务的最大响应时间，即从该任务被释放直至运行完成的时间间隔，$B(\tau_{i,i}^k)$ 表示任务被高优先级任务抢占的时间，$P(\tau_{i,i}^k)$ 表示任务运行所在的处理机。

2.2.2 多处理机群集

假设实时系统提供了 M 个处理机，记为 $P=\{p_1, p_2, \cdots, p_M\}$。按照不同关键度将处理机集合 P 划分为多个群集(processor cluster)，记为 $PC=\{pc_1, pc_2, \cdots\}$，其中每个群集 $pc_s \in PC$ 由若干个处理机组成，其对应的关键度记为 $CE(pc_s)$。按照防危性需求，只有关键度等于 $CE(pc_s)$ 的功能组件才能被分配给 pc_s。每个处理机上运行着来自不同组件的分区，对于处理机 p_j，其上运行的分区集合记为 $SP(p_j)=\{sp_{j,1}, sp_{j,2}, \cdots\}$，而分区 $sp_{j,t} \in SP(p_j)$ 所包含的任务集记为 $\Gamma(sp_{j,t})$。各群集所包含的处理机数量可依照当前应用负载而动态变化。当应用负载较低时，空闲处理机将被回收到备份群集 pc_b。随着应用负载的增加，pc_b 将向其他群集提供可用的处理机，如图 2 所示。

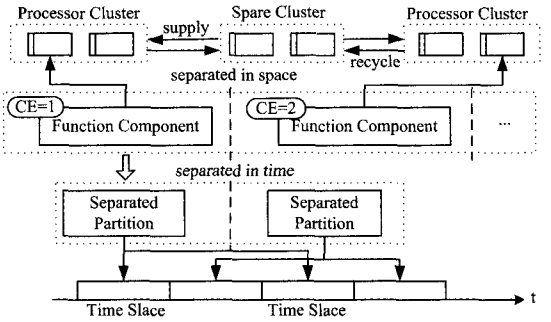


图 2 分层防危调度原理

2.2.3 分层防危调度器

分层防危调度器分为全局层(global level)、本地层(local level)和任务层(task level) 3 个层次，如图 1 所示。

在全局层，对安全关键应用的多个功能组件及所属分区进行全局调度，具体的做法是将每个功能组件分配到相同关键度的群集，并在群集中为该组件所属分区确定合适运行的处理机，如图 2 所示。由于这些组件运行在不同群集上，彼此实现物理隔离，因此调度实现了空间隔离防危效果。

在本地层，对每个处理机上运行的多个分区进行本地调度，具体的做法是采取时间片轮询方式定期激活各个分区，如图 2 所示。当分区被激活后，该分区所包含的实时任务开始执行，一旦时间片耗尽，该分区将被阻塞，等待下次运行。由于分区在不同时间片内独立运行，因此这种调度方式实现了时间隔离防危效果。

在任务层，对每个分区所包含的实时任务进行任务调度，而调度策略的选取将依照各分区内实时任务的特点，如任务是否周期到达、任务对实时性约束是否严格等因素来合理安排。为了便于叙述，本文仅考虑固定优先级调度策略。

3 分层防危调度算法 HSS

基于上述模型，本文提出一种分层防危调度算法 HSS (Hierarchical Safeguard Scheduling Algorithm)，本节首先介绍算法的相关概念，然后给出详细的算法设计步骤。

3.1 相关概念

为了更好地描述算法内容，本节参照文献[5-7]给出相关定义与定理。

定义 1 如果某分区中所有的实时任务在某种调度策略

下都能在时限前完成，则称该分区在该调度策略下是可调度的。

定义 2 对于处理机 p_j 上运行的分区 $sp_{j,t}$，每次轮询周期为其分配的运行时间片记为 $T(sp_{j,t})$，其在轮询总周期 $T(p_j)$ 中所占的比例称为该分区的运行占比，记为 $\alpha_{j,t}$。如果该分区采用 RM 调度策略，那么其最小运行占比 $\mathrm{MIN}(\alpha_{j,t})$ 可通过式(1)计算，其中 $U_{j,t}$ 表示分区利用率，$N(sp_{j,t})$ 表示分区包含的任务的数量。

$$\mathrm{MIN}(\alpha_{j,t})=\frac{U_{j,t}}{(N(sp_{j,t})\times(2_{1/N(sp_{j,t})}-1))} \tag{1}$$

定理 1 在处理机 p_j 上运行着分区集合 $SP(p_j)=\{sp_{j,1},sp_{j,2},\cdots\}$，分区内采用 RM 调度策略。如果分区集合 $SP(p_j)$ 满足式(2)，则称该分区集合在 p_j 上可被调度。

$$\sum_{sp_{j,t}\in SP(p_j)}\mathrm{MIN}(\alpha_{j,t})\leqslant 1 \tag{2}$$

定义 3 在处理机 p_j 上运行着分区集合 $SP(p_j)=\{sp_{j,1},sp_{j,2},\cdots\}$，各分区内采用 RM 调度策略。如果该分区集合可被调度，则分区集合的轮询总周期 $T(p_j)$ 可通过式(3)来计算，而每个分区 $sp_{j,t}\in SP(p_j)$ 的运行时间片可通过式(5)来计算。

$$T(p_j)=\min_{sp_{j,t}\in SP(p_j)}(B_{\min}(\mathrm{MIN}(\alpha_{j,t}))/(1-\mathrm{MIN}(\alpha_{j,t}))) \tag{3}$$

$$B_{\min}(\alpha_{j,t})=\min_{\tau_i\in\Gamma(sp_{j,t})}(\max_{l\in[0,D(\tau_i)]}(l-\sum_{\tau_k\in hep(i)}(\frac{C(\tau_k)}{\alpha_{j,t}}\times\left\lceil\frac{l}{T(\tau_k)}\right\rceil))) \tag{4}$$

$$T(sp_{j,t})=T(p_j)\times\mathrm{MIN}(\alpha_{j,t}) \tag{5}$$

3.2 算法设计

本文提出的 HSSA 算法分为 3 个部分，即群集分配算法 CAS、本地分区调度算法 LSS 和分区任务调度算法 STS，下面分别具体介绍。

3.2.1 群集分配调度算法

群集分配调度算法主要用于调度安全关键实时应用的多个功能组件，调度分为两个步骤：首先为每个功能组件选择关键度匹配的群集，然后将功能组件中多个分区分配到可调度的处理机上运行，如果无法找到可调度的处理机，则从备份群集中分配新的处理机，并将其加入到当前集群。算法基本步骤如下所示。

Algorithm 1 CAS (Cluster Assigning Scheduling)

```
for all cp_k ∈ CP do
  assign cp_k to pc_s when CE(pc_s)=CE_k;
  foreach sp_t^k in SP(cp_k) do
    compute MIN(α_{j,t}) by equation 1;
    if ∃ p_j ∈ pc_s can satisfiy that {sp_t^k} ∪ SP(p_j) is schedulable on p_j
      by theorem 1 then
      assign sp_t^k to processor p_j;
    else
      assign cp_k to new processor p_new from pc_b and join p_new to pc_s;
    end if
  end for
end for.
```

3.2.2 本地分区调度算法

本地分区调度算法主要用于调度单个处理机上的分区集合，该调度分为两个部分：(1)计算分区参数，即分区集合的轮询总周期以及各分区的运行时间片；(2)采用时间片轮询的方式依次激活各个分区运行，当分区时间片耗尽后，算法阻塞当前分区，并激活下一个分区。算法基本步骤如下所示。

Algorithm 2 CPD (Cycle Parameter Designing)

```
foreach p_j ∈ P do
  compute the great cycle T(p_j) by equation 3;
  foreach sp_{j,t} in SP(p_j) do
    compute acticvation time T(sp_{j,t}) by equation 5;
  end for
end for.
```

Algorithm 3 CSS (Cycling Separation Scheduling)

```
while (true) do
  foreach sp_{j,t} in SP(p_j) do
    suspend current running tasks;
    activate the running of tasks in sp_{j,t} during the interval of T
    (sp_{j,t});
  end for
end while.
```

3.2.3 分区任务调度算法

分区任务调度算法采用固定优先级抢占式 RM 策略来调度分区的实时任务。算法只在分区被激活的前提下进行，其触发时机发生在激活分区内有新任务 τ_{new} 到达或本地处理机已完成当前任务。对于第一种情况，算法从激活分区 sp_a 的等待队列中选择最短周期(最高优先级)任务来运行；对于第二种情况，算法将依据任务 τ_{new} 的优先级来区别对待，即如果新任务的优先级高于当前运行任务的，则抢占其运行，否则添加到等待队列。算法关键步骤如下所示。

Algorithm 4 STS (Separation Task Scheduling)

```
while τ_now ∈ Γ(sp_a) completed or new task τ_new ∈ Γ(sp_a) released do
  if τ_now ∈ Γ(sp_a) completed then
    find task τ_k with the shortest period in waited queue;
    start to run τ_k on p_j;
  end if
  if new task τ_new ∈ Γ(sp_a) released then
    if τ_new has the shortest period in waited queue then
      suspend τ(p_j) and start to run τ_new on p_j;
    else
      add τ_new to waited queue;
    end if
  end if
end while.
```

4 仿真实验

本文通过一组仿真实验来评估 HSS 算法的性能。为了便于比较，我们基于文献[5-7]的思想设计了分区调度算法 PS(Partition Scheduling)和分隔调度算法 SS(Separated Scheduling)作为本次试验的参照算法，其中 PS 算法将实时应用的各个组件集中调度到一个处理机上运行，并采用时间隔离方式实现防危性，而 SS 算法将每个组件分配到独立的处理机上运行，以物理隔离实现防危性。

实验主要采用以下 3 个性能评价指标：(1)调度算法防危性，即实时应用中低关键度部分(组件或分区)的运行失效对高关键度部分(组件或分区)的影响程度；(2)调度处理机数量 $NP(Algo)$，即采用某种调度算法进行成功调度所需的处理

机数量;(3)调度处理机平均负载率,即所有可调度处理机负载率的平均值。

仿真实验代码采用C++语言编写,参与实验的实时应用通过随机方式产生,其参数设置如下:实时应用包含的组件个数为5,设立两级关键度,每个组件按照防危性需求划分为3个分区,分区负载率在区间[0.01,0.8]内取值。每个分区包含20个实时任务,各实时任务的周期随机生成,其时限与周期相等,任务负载率随机生成,其总和等于分区负载率,任务运行时间通过任务负载率与周期的乘积得出。

为了测试调度算法的防危性,实验采用软件方式模拟实时系统可能发生的4种失效,它们分别是分区的软、硬失效和组件的软、硬失效,其中软失效表示分区(或组件)因故障导致自身运行中断或终止,而硬失效表示分区(或组件)因故障导致所在的处理机崩溃。实验主要考察以下几种情况:(1)低关键度分区失效对同一组件内其他分区的运行影响;(2)低关键度分区失效对高关键度组件的影响;(3)低关键度组件失效对高关键度组件的影响。

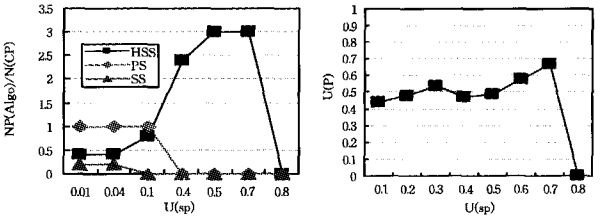
表1给出了3种调度算法在4种失效情况下的防危性表现,其中$H(P)$表示与失效分区处于同一组件的其他分区,$H(C)$表示高关键度组件,Y表示运行成功,N表示运行失败,U表示可部分运行。实验结果表明,HSS算法在4种失效情况下的防危性表现优于PS算法和SS算法,这是因为HSS将不同关键度组件分配给不同集群,彼此实现了物理隔离,因而能够抵御组件硬失效的影响。另外,HSS将每个组件拆分为不同分区,并将其分配到集群的不同处理机上运行,在相同处理机上的分区之间实现了时间隔离,因而能够抵御分区软失效的影响。而在不同处理机上运行的分区之间实现了物理隔离,当某个分区发生硬失效时,只有与该失效分区处于不同处理机的分区才能抵御这种影响,因此这种情况下的实验结果为可部分运行。综上所述,HSS算法在软硬件失效情况下的防危性优于其他两种算法。

表1 4种失效对系统运行的影响

Scheduling Algorithm		SS		PS		HSS	
Scheduled Object		H(P)	H(C)	H(P)	H(C)	H(P)	H(C)
Partition Failure	hard	N	Y	N	N	U	Y
	soft	N	Y	N	Y	Y	Y
Component Failure	hard		Y		N		Y
	soft		Y		Y		Y

图3(a)表示分区负载率对3种算法的调度处理机数量的影响,图中横轴表示分区负载率,取值范围从0.01到0.8,而纵轴表示运行每个组件所需的平均处理机数量,即调度所需处理机数量与实时应用中组件总数的比值。当分区负载率逐渐增加时,PS和SS算法所需的处理机数量固定不变,而HSS算法所需的处理机数量逐步上升。从图中可知,利用硬件优势,HSS算法能够承担比其他两种算法更高的分区负载率,因此在应用负载承受能力上优于其他两种算法。

图3(b)表示分区负载对HSS算法的平均处理机负载率的影响。当分区负载率增加时,HSS的平均处理机负载率呈现波段式变化,这是因为当分区负载率超过当前处理机数量承受能力时,HSS通过增加处理机来实现调度,而新增加的处理机分担了其他处理机的负载,从而降低了平均处理机负载率,但随着分区负载的增加,新增处理机负载能力将逐渐被耗尽,从而导致平均处理机负载率的升高。因此,当处理机数量固定时,通过合理设置分区负载率可以最大限度地利用处理机资源。



(a) U(sp)对NP(Algo)的影响　(b) U(sp)对U(P)的影响

图3 仿真实验结果

结束语 在安全关键实时应用中,低关键度部分(组件或分区)发生失效时,系统必须采用防危措施来保护高关键度的部分(组件或分区)不受其影响,而现有防危调度算法在软硬件失效情况下防危能力表现不佳。针对这个问题,本文的主要贡献有:(1)构建了一种分层防危实时调度模型;(2)基于上述模型,提出了一种新的分层防危调度算法HSS;(3)将HSS算法与两类参照算法在不同条件下进行了仿真实验对比,实验结果分别反映出HSS算法在防危效果和应用负载承受能力方面的优势。下一步工作包括:在调度算法中考虑不同的分区任务调度策略,以及研究更优化的分区设计方式。

参 考 文 献

[1] Kuo Tei-wei, Li Ching-hui. A Fixed Priority Driven Open Environment for Real-Time Applications[C]// Proceedings of the IEEE Real-time Systems Symposium. 1998:256-267

[2] Deng Z, Liu J W-S. Scheduling Real-Time Applications in an Open Environment[C]// Proceedings of the IEEE Real-Time Systems Symposium. 1997:308-219

[3] Saewong S, Rajkumar R, Lehoczky J P. Analysis of hierarchical fixed-priority scheduling[C]// Proceedings of the 14th Euromicro Conference on Real-Time Systems. 2002:173-181

[4] Davis R, Burns A. An investigation into server parameter selection for hierarchical fixed priority pre-emptive systems[C]// Proceedings of the 16th International Conference on Real-time and Network Systems. 2008:19-28

[5] 李昕颖,顾健,何锋,等. 硬实时系统在强分区约束下的双层分区调度[J]. 计算机学报,2010,33(6):1032-1039

[6] 李全军,张安,张耀中. 航电系统综合处理机时间隔离保护机制研究[J]. 计算机工程与应用,2009,45(3):167-169

[7] 杨仕平,熊光泽,桑楠. 安全关键实时系统高可信集成技术的研究[J]. 电子学报,2003,31(8):1237-1241

[8] Davis R, Burns A. Hierarchical fixed priority preemptive scheduling[C]// Proceedings of the 26th IEEE International Real-Time Systems Symposium. 2005:398-408

[9] Almeida L, Pedreiras P. Scheduling within temporal partitions: response time analysis and server design[C]// Proceedings of the 4th ACM International Conference on Embedded Software. 2004:95-103