

支持时空隔离机制的微内核设计与实现

王晓鹏, 张 凡, 苏卫宁

(西北工业大学 计算机学院, 陕西 西安 710072)

摘 要: 基于时空隔离的微内核设计思想能够满足安全关键嵌入式系统的非功能需求, 给出了基于时空隔离机制的微内核系统架构设计, 并对其时空隔离机制的实现给出了详细描述, 并对微内核在嵌入式领域的应用进行了展望.

关键词: 安全关键; 微内核; 时空隔离

中图分类号: TP391.8

文献标识码: A

文章编号: 1000-7180(2014)09-0011-04

Design and Implementation of Temporal-Spatial Isolation Based Microkernel

WANG Xiao-peng, ZHANG Fan, SU Wei-ning

(School of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China)

Abstract: The principle of Temporal-Spatial isolation based microkernel well satisfy the requirements, and has the important meaning to the development of the embedded application. This paper gives the design of Temporal-Spatial Isolation based microkernel, and the realization of temporal-spatial isolation mechanism is given in detail. Finally, the potential application of microkernel in embedded system are prospected.

Key words: safety-critical; microkernel; temporal-spatial iosolation

1 引言

解决微内核性能问题的其中一种思路是精简内核并对 IPC 进行直接优化, 这种思路导致了被称为第二代微内核的出现. L4^[1-4] 微内核和 Exokernel^[5] 微内核是其中的典型. 在性能方面, L4 的 IPC 比第一代微内核的 IPC 快 20 倍以上, 甚至比传统操作系统的 IPC 还要快. 同时, 由于 L4 微内核仅仅以 7 个系统调用接口就实现了以上功能, 所以只占用了 12 kB 内存. 相对于 L4 微内核, Exokernel 微内核是一个更加激进的设计, 其核心观点是: 内核应该支持一个最小的、高度优化的原语集. 从 Exokernel 操作系统原型的测试数据可知, 其内核的原语操作速率比 UNIX 操作系统要快 10~100 倍.

2 基于时空隔离微内核的架构设计

本文提出的基于时空隔离的微内核及基于微内核的操作系统框架分为两层: 核心层提供支持时空隔离机制的最小功能集, 包括分区管理、线程管理、线程调度及基本分区通信功能, 运行在内核态; 应用接口层, 包括运行时配置、安全策略管理、运行时监测、外设管理等功能, 运行在用户态, 为上层应用提供支撑. 基于时空隔离的微内核框架设计如图 1 所示.

2.1 核心层设计

(1) 分区管理由分区状态管理、分区空间管理和分区线程及调度策略管理等构成. 分区状态包括 INIT, NORMAL, IDLE, RESTART, STOP 五种状态, 系统初始化时根据运行时配置将物理内存分配

给各个分区,在此基础上进一步将对分区内的内存区域进行划分,确定分区中的数据段、代码段、堆栈和分区通信所需内存区域,如图2所示,并确定分区主线程的入口地址并将分区处于INIT状态;系统运行时,若分区被调度则启动分区的主线程并执行。

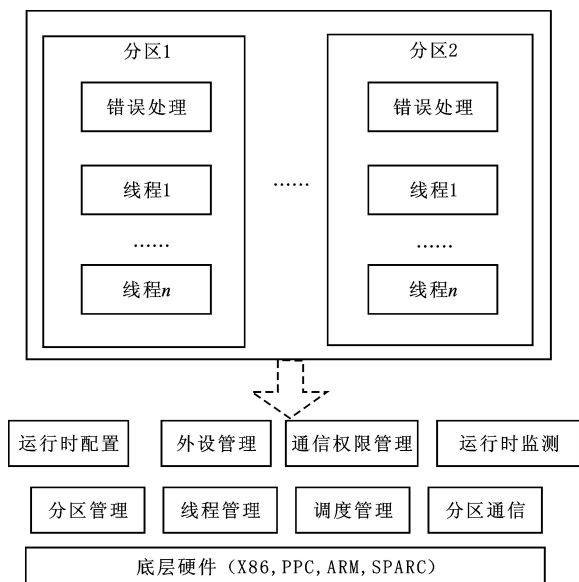


图1 基于时空隔离微内核的系统架构及其应用

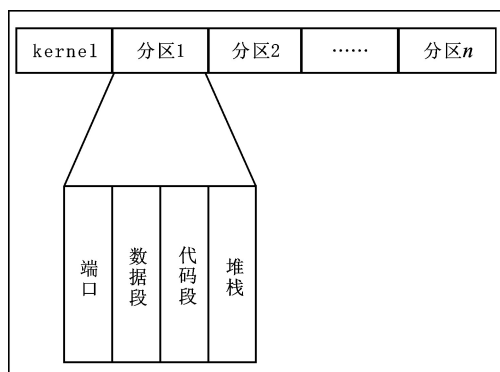


图2 系统分区结构

(2)线程管理由线程初始化、线程资源管理、线程状态管理等构成。线程的状态包括ACTIVE,EXECUTE,BLOCK,STOP四种状态,系统初始化时根据系统默认值对线程属性进行初始化,为线程分配资源,例如线程所属的分区、执行时间、入口地址和堆栈地址等。当线程因某种原因阻塞时,要进行线程间的切换,以及终止某个不能继续运行的线程等。另外,线程管理还提供了两个特殊的线程,即内核线程(kernel-thread)和空闲线程(idle-thread)。内核线程运行于内核态,执行特权指令;空闲线程在系统中没有线程执行时会被调用。

(3)调度管理,提供分区调度和分区内调度。分

区调度采用轮转调度算法(RR)对分区进行调度,微内核初始化时根据系统参数配置为每个分区分配时间片和调度序列,运行时按照调度序列依次调度执行。分区内调度用于对分区内所有线程进行调度,系统初始化时根据运行时配置会对每个分区的调度器进行初始化,设置其调度策略。分区被调动执行时,调度管理会启动分区内的调度器。分区内调度可以选择不同的调度策略,如轮转调度、短作业优先(SJF),最早截止期优先调度(EDF)等。

(4)基本分区通信,提供分区间通信(Inter-partitions communication)和分区内通信(Intra-partition communication)的必要支持。分区间通信实现分区间的数据交换机制,为每个分区建立了数量不等的的数据交换端口,数据通过端口与其他分区进行交换。分区内通信管理实现分区内线程之间数据交换,分区内通信管理与其他微内核中线程之间的通信类似,提供不同的通信方式,如消息、事件,信号量等。为了提高分区通信的性能,微内核提供了一些优化策略^[1-2]来提高数据交换的效率。例如,对于短的传输数据可以直接将其写入寄存器进行传输,而对于长的传输数据可以通过直接调度接收该数据的线程来减少数据传输所需时间。

2.2 应用接口层

(1)运行时配置,为用户使用微内核提供配置接口,系统在初始化时,用户根据需求对微内核运行参数进行设置,如系统的分区数、总的线程数、每个分区中的线程数,以及分区间通信端口及数据传输控制策略等,部分配置参数如表1所示。例如,在微内核中我们需要建立4个分区,分区1和2中各有2个线程,分区3有4个线程,分区4有5个线程。每个分区执行时间均为100 ms,并按照分区顺序进行调度,则配置文件如下:

```
#defineSAFEBOX_THREADS13
#defineSAFEBOX_PARTITIONS4
#defineSAFEBOX_PARTITIONS_THREADS{2,2,4,5}
#defineSAFEBOX_NBSLOTS4
#defineSAFEBOX_SLOTS{100,100,100,100}
#defineSAFEBOX_SLOTS_ALLOCATION{1,2,3,4}
```

(2)通信权限管理,对分区间通信过程进行认证。在微内核中,不同的分区可设置不同的安全级别。系统在初始化时根据运行时配置会为每个分区设置不同的安全级别,在此基础上对分区间通信的数据流进行安全控制,通过对分区间线程通信进行

表1 微内核主要运行时配置参数

SAFEBOX_THREADS	定义微内核中总的线程数
SAFEBOX_PARTITIONS;	定义微内核所拥有的分区数.
SAFEBOX_PARTITIONS_THREADS	定义微内核每个分区中所包含的线程数. 该参数为数组类型, 数组中的每个元素的值对应分区中所包含的线程数.
SAFEBOX_NBSLOTS	分配给分区的时间片个数.
SAFEBOX_SLOTS	定义时间片的大小.
SAFEBOX_SLOTS_ALLOCATION	定义时间片的分配方法, 即在哪个时间片调用哪个分区. 这个声明是一个数组, 数组中的值表示了使用这个时间片的分区号.

配置,从而决定分区的哪一个端口可以发生数据,其发送的数据可以由哪些端口接收等.每次进行分区间数据交换时,通信权限管理根据配置信息判断此次数据交换是否被允许,如果允许,则进行数据传输;否则,返回通信错误消息.

(3)外部设备管理,由外部设备数据的输入输出管理和外部设备的访问控制管理构成.在系统进行初始化时系统会为每个外部设备分配一块内存区域,外部设备通过这块内存区域进行数据的输入输出,并且根据运行时配置信息,每个外部设备的内存区域只能被一个特定的分区访问(和外部设备绑定的分区).其他分区要访问外部设备的内存区域,则必须通过与这个特定的分区进行分区间数据交换来实现.

(4)运行时监测,对分区的运行状态和系统的状态进行监测.通过对系统运行时可能出现的异常制定相应的处理措施,运行时监测模块可实时监测分区状态和异常情况,并依据策略执行相应动作,如当分区因为某些原因而意外停止时微内核可对该分区进行重启,从而能够提高系统的可靠性和容错能力.

3 时空隔离机制的实现

基于时空隔离微内核中,系统的功能模块被隔离在各个分区中,并通过一个专用的内核对分区隔离进行监督,该内核通过对系统进行时空隔离来确保分区的隔离.

时间隔离:为每个分区分配一个固定的时间片,在规定的分区时间片内分区利用自己的调度机制来调度执行其内的线程.

空间隔离:每个分区被固定在某个具体的内存区域,并且这些区域之间不互相重叠,这样可以保证分区不会对其他分区的内存区域进行访问.但是分

区之间可以通过专门的通道进行数据通信.

因此,微内核的时空隔离通过空间(分区管理)和时间(任务调度)隔离实现.下面就实现机制做简要表述.

3.1 空间隔离

分区是基于时空隔离微内核中的资源管理单位.在系统编译之前根据分区内用户程序占用空间及堆栈、数据空间,由使用者通过配置接口配置好各个分区的大小.系统运行之后,微内核应该根据配置的大小为分区分配相应大小的一块连续内存空间,然后把用户程序装载到分区的起始地址处,用户堆栈从分区的最底端向上生长,当然分区内会有很多线程,各个线程有自己独立的堆栈,每个线程的用户级堆栈大小都是相同的因此很容易界定它们,如图3所示.

基于时空隔离微内核的分区应该被隔离起来受到保护,因此每个分区应该分配两个段描述符,它们的基地址是分区起始地址,段长度是分区长度,一个用作界定代码段,一个用于界定数据段,这样分区内的程序可以访问分区内的所有资源但是无法访问到分区外部资源,若程序出错产生了错误的地址指向了分区外部,那么立刻会产生段保护异常.因此,分区即受到了保护又不会因为出错干扰到其他分区或者内核.

微内核初始化时,根据运行时配置信息对分区管理数据结构中的参数进行赋值,通过对分区分配不重叠的物理内存区域,从而达到空间隔离的目的.在此基础上,系统进一步在分区物理内存中划分出数据段、代码段、堆栈等区域,从而保证运行时分区间的空间独立.分区初始化函数部分伪代如下:

1. i 表示分区号
2. for 每一个分区 i do
3. 对分区 i 的快大小进行赋值
4. 对分区 i 的物理初始地址进行赋值

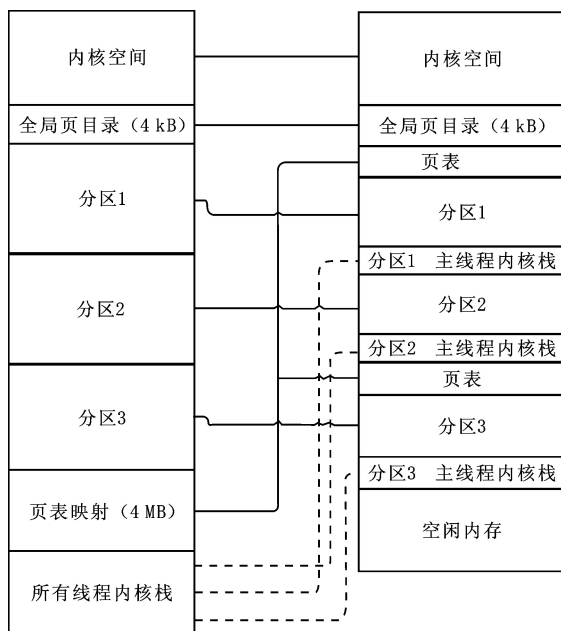


图3 时空隔离微内核分区结构和内存划分

5. 对分区 i 内所得最大线程数进行赋值
6. 初始化分区 i 主线程入口地址
7. 分区 i 中当前已创建的地址初始化为 0
8. 对分区 i 的其他属性进行赋值
9. end

3.2 时间隔离

基于时空隔离微内核,为了满足时间隔离策略,需要一个可以预知的方法来对时间片分配的执行进行保证. 所以其使用 RR 算法协议对分区进行调度: 每个分区固定的执行一个固定的时间. 同时微内核采用了分层调度机制: 即分区调度和分区内调度. 图 4 是一个具有四个分区微内核的调度策略: 分区 1 和分区 3 内部实现的是 RMS(Rate Monotonic Scheduling)调度算法, 而分区 2 和分区 4 则使用的是 RR 调度算法, 每个分区执行 100 ms, 各分区在其时间片内, 使用其实现的调度算法调度它的线程:

在基于时空隔离微内核中,调度主要是对线程根据任务和分区的时间需要进行调度服务. 分区主要是使用循环调度算法进行调度, 而分区内调度可基于应用需求选择不同的调度策略, 如轮转调度、短作业优先(SJF), 最早截止期优先调度(EDF)等. 分区调度的伪代码如下:

(1) 分区间调度

1. 选择分区

```
if 当前时间片 < 总的时间片个数
    选择下一个时间片执行的分区
else
```

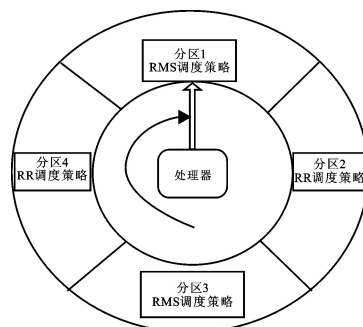


图4 基于时空隔离微内核的调度策略

重置时间片

选择在第一个时间片执行的分区

(2) 调度分区

启动分区调度器

调度主线程

2. 分区内调度

if 当前分区内已经创建的线程 < 分区可包含的最大线程数

根据分区调度器执行的调度算法, 选择分区内处于可运行状态的线程.

调度所选择的线程

```
return true
```

else

```
return false
```

3. end

可以看出,分区隔离的特点: 从时间和空间上都进行了隔离, 这样做不仅有利于系统的设计, 将各个模块的依赖关系降到了最低, 而且也避免了结构上的错误, 有利于系统将来的验证, 从极大的程度上降低了系统验证的难度. 因此, 时空隔离微内核开发目标是减小高可靠性系统验证的难度, 把系统中安全机制分散成诸多可管理的组件, 进程或者线程被分散到不同分区, 分区定义了代码, 数据和系统相关资源. 不同安全级的功能运行在不同的分区, 这样一个分区功能发生错误就不会影响其他功能分区的运行, 阻止了错误的传播, 使得系统更加安全.

4 结束语

本文给出了一种基于时空隔离机制的微内核设计方法, 给出了微内核的设计框架, 并且详细地描述了各模块的功能, 最后给出了其核心时空隔离机制的实现方法. 随着对微内核研究的不断深入, 微内核的性能和安全性会不断地提高, 相信在不久的将来, 越来越多的微内核会在嵌入式系统中得到应用.

(下转第 19 页)

系统中大部分数据类型,具有非常强的可扩展性和应用性,能够为用户服务、订单处理等实时任务提供高速的响应,并且支持大数据处理,能够在后续工作中为数据挖掘等科学研究提供有效的数据处理方式。虽然航天数据自身结构非常复杂而且实时性要求较高,目前在高速处理方面还有所欠缺,但是在将来,实时流计算概念一定会逐步成为航天地面系统数据处理的主要方式。

表1 并行计算处理结果 Mb/s

	单机处 理速率	并行处理 速率(4- 1)	并行处理 速率(4-2- 1)	并行处理 速率(4-3- 1)
单帧事务数据流	30	120	120	120
单帧事务集比重 75%	30	60	82	91
单帧事务集比重 50%	30	42	61	76
随机长度数据帧	30	44	62	80

参考文献:

- [1] 李博权. 空间多载荷协同工作信息可视化研究[J]. 微电子学与计算机, 2012, 29(8): 56-58.
- [2] Tom White. Hadoop 权威指南[M]. 2版. 北京:清华大学出版社, 2011.
- [3] Apache Incubator. Storm-wiki [EB/OL]. (2013-12-01). [2014-06-07]. <http://storm-project.net>.

(上接第14页)

参考文献:

- [1] Liedtke J. Improving IPC by kernel design[C] // ACM SIGOPS Operating Systems Review. Germany: Dagtuhl Csstle, ACM, 1994; 27(5): 175-188.
- [2] Blackham B, Shi Y, Heiser G. Improving interrupt response time in a verifiable protected microkernel[C] // Proceedings of the 7th ACM european conference on Computer Systems. Swizerland, Berne, ACM, 2012: 323-336.
- [3] Dannowski U, LeVasseur J, Skoglund E, et al. I4 experimental kernel reference manual [R]. Germany: University of Karlsruhe, Version X, 2011.
- [4] Blackham B, Heiser G. Correct, fast, maintainable; choose any three! [C] // Proceedings of the Asia-Pacific Workshop on Systems. China: Beijing, ACM, 2012: 13.
- [5] Marzi H, Yang Y. Microkernel based real-time embedded operating systems efficiency improvement [C] // Electrical & Computer Engineering (CCECE), 2012 25th IEEE Canadian Conference on. Montreal, IEEE, 2012: 1-4.

- [4] 左莉华. 高级在轨系统(AOS)数据传送与分路简介[J]. 遥测遥控, 1993(3): 47-49.
- [5] 量子恒道. Storm 系统框架研究[EB/OL]. (2013-12-10). [2014-06-07]. <http://blog.linezing.com>.
- [6] Ghemawat S, Gobioff H. The google file system[C] // Proceeding of the 19th ACM Sympoium on Operating Systems Principles. New York: ACM Press, 2003.
- [7] 刘伟峰, 王智广. 细粒度并行计算编程模型[J]. 微电子学与计算机, 2008, 25(10): 72-74.
- [8] David karger, Eric lehman. Consistent hashing and random trees[C] // STOC'97 Proceedings of the twenty-ninth annual ACM symposium on Theory of Computing. New York, 1997.
- [9] Frank Dehne. Scalable parallel computing geometry [C] // Michel Cosnard, Afonso Ferreira, Parallel and Distributed Computing. USA: Afonso Ferreira, 1994.

作者简介:

李一辰 男, (1987-), 硕士研究生. 研究方向为实时流计算.
李绪志 男, (1968-), 研究员, 博士生导师. 研究方向为地面应用系统总体设计、综合信息系统、智能信息处理.
阎镇 男, (1974-), 硕士, 高级工程师. 研究方向为空间科学与应用科学数据可视化.

作者简介:



张凡 男, (1979-), 博士, 副教授. 研究方向为嵌入式计算技术、形式化方法.



王晓鹏 男, (1988-), 硕士研究生. 研究方向为嵌入式计算技术、形式化方法.



苏卫宁 男, (1990-), 硕士研究生. 研究方向为嵌入式计算技术、形式化方法.