第35卷 第23期 计 算 机 工 程 2009年12月
Vol.35 No.23 Computer Engineering December 2009
·工程应用技术与实现· 文章编号：1000—3428(2009)23—0249—03 文献标识码：A 中图分类号：N945.12

支持多核架构的微内核操作系统设计

张荫芾，应忍冬，周玲玲
(上海交通大学电子信息与电气工程学院，上海 200240)

摘 要：针对多核架构开始在嵌入式领域普及的趋势，设计一个基于多核处理器架构的微内核操作系统，描述系统中内存管理、线程调度、锁和中断、线程间通信以及应用程序等各部分的设计方案。该设计充分利用多核架构和微内核操作系统的特点，不仅应用于与Intel公司合作的L4微内核操作系统研究项目，也为其他微内核操作系统设计提供了参考。

关键词：微内核；多核架构；操作系统；嵌入式系统

Multi-core Architecture Supported Micro-kernel Operating System Design

ZHANG Yin-fei, YING Ren-dong, ZHOU Ling-ling
(School of Electronic, Information and Electrical Engineering, Shanghai Jiaotong University, Shanghai 200240)

【Abstract】Based on the trend that multi-core architecture is more and more widely used in embedded-system field, this paper designs a micro-kernel operating system which supports multi-core architecture. It describes memory management, thread schedule, lock & interrupt, inter-thread communication as well as application design. The design fully utilizes the features of multi-core architecture and micro-kernel operating system. This paper not only provides the design for the L4 micro-kernel research project cooperated with Intel Co., but also benefits other developers as a reference design for the micro-kernel operating system.

【Key words】micro-kernel; multi-core architecture; operating system; embedded system

1 概述

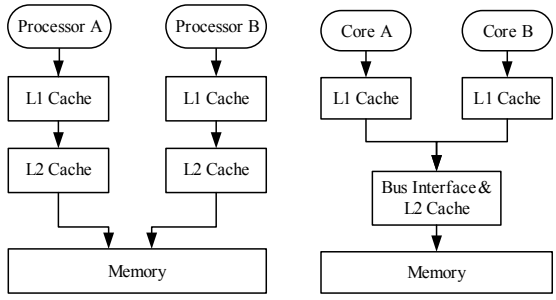
随着计算机技术的飞速发展，多核处理器已经在个人电脑上普及，并且开始进入嵌入式设备领域。有了硬件的支持，还必须有操作系统。传统的个人电脑上的Windows和Linux操作系统虽然可以通过裁剪移植到嵌入式设备上，但并非最佳选择。基于微内核架构的操作系统因其在实时性、安全性和可靠性方面的出色性能，非常适合嵌入式领域。微内核操作系统是指在内核中只保留最基本的进程调度、地址映射和进程间通信功能，而令其他功能都以用户态的服务程序的方式实现的操作系统。在微内核操作系统中，由Jochen Liedtke开发的L4微内核操作系统已经成为了第2代微内核的一个标准，在学术和工程领域都有广泛应用。当前已经有不少国外学者提出了L4在多处理器架构上的设计方案[1-3]，但尚无针对多核架构的设计。本文正是填补了这个缺口，提出一种设计方案，基于L4最广泛的一个实现Fiasco进行修改和增强，使之可以运行在多核架构上。

为避免混淆，对于多核的“核”(core)，本文使用处理器“核心”或者“核”这个称呼，对于微内核操作系统的“核”(kernel)，本文使用“内核”这个称呼。

2 多处理器架构和多核架构的比较

如前文所述，已经有不少的国外学者研究了支持多处理器架构的L4操作系统的设计，但由于多核架构和多处理器架构上的一些不同点，两者之上的L4操作系统的设计理念有非常大的区别。

如图1所示，多核架构的2个核共享一些电路，比如二级缓存、前端总线等。由于2个核在同一片芯片内部，互相之间的距离较近，这就允许缓存的同步电路运行在一个相当高的时钟频率上，这比多处理器架构需要信号在2片芯片之前传递的情况要快得多，而更高的时钟频率使单位时间里可以发送更多的数据[4-5]。



(a)多处理器架构 (b)多核架构

图1 多处理器架构和多核架构的比较

另一方面，多处理器架构也有自己的优点，正因为2个处理器不共享前端总线，当2个处理器同时访问内存中不同地址的时候不会相互影响，这就相当于多处理器架构在访问内存的时候带宽要比多核架构大得多。但多核架构在2个核之间高速通信的优点在大多数情况下都能弥补它由于共享前

基金项目：英特尔大学合作研究基金资助项目“微内核架构上的虚拟化技术研究”

作者简介：张荫芾(1984－)，男，硕士研究生，主研方向：嵌入式系统；应忍冬，讲师；周玲玲，副教授

收稿日期：2009-06-30 **E-mail：**zhangyinfei@gmail.com

端总线而造成的访问内存的性能上的损失。所以当前多核架构更加流行一些。

综上所述，本文在设计支持多核架构的 L4 操作系统时，会放宽对多核间 IPC 的限制，因为系统可以利用共享的 L2 Cache 存放消息来实现高速的多核间 IPC。但考虑到共享前端总线，在设计时会避免频繁地访问内存。

3 系统设计

3.1 总体设计

对于多核上的操作系统设计，最容易想到的设计方案是在某一个核(一般称为 Bootstrap Core)上运行操作系统，然后这个操作系统管理所有的核。但这种方案需要大量修改原有的 L4 代码，并且系统结构复杂，难以保证微内核操作系统应有的效率。正因为 L4 是微内核架构的操作系统，本设计采用另一种选择，就是如图 2 所示的在每一个处理器核心(core)上运行一个 L4 操作系统的内核(kernel)。各个内核之间互相独立，又有一定的同步机制。而之上运行的服务程序如内存分配程序 sigma0，资源管理器 roottask，以及其他程序也都是运行在各个独立的内核上，通过多核间 IPC 来通信。本文称之为离散模型。

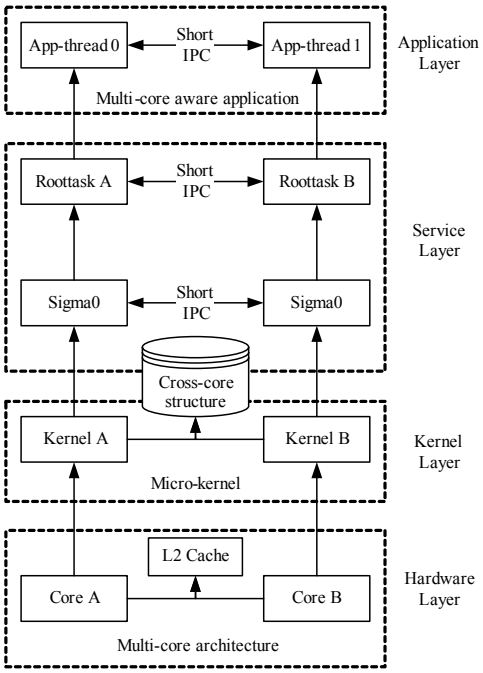


图 2 总体架构设计的离散模型

基于离散模型的架构设计的优点不仅是可以大量复用原有的 L4 代码，它还非常适用于嵌入式场景。比如嵌入式设备需要高可靠性[6]，本设计可以使用一个核运行所有的应用程序，而另一个核作为热备份。由于 2 个核上都运行了操作系统，就算操作系统崩溃，也可以较为平滑地过渡到热备份的核上。此外，L4 是一个实时的操作系统，本设计可以指定一个核的资源完全用于实时的应用程序，而另一个核的资源留给非实时的程序，这样的隔离架构非常清晰，而实现方式只要指定一个操作系统内核运行在实时模式，而另一个运行在非实时即可[7]。不仅如此，离散模型在安全性上也有应用，L4 的一个著名的应用是在微内核上运行多个 L4 Linux，密钥放在一个 L4 Linux 中，而执行程序运行在另一个中。现在本设计把密钥放在某个核上运行的 L4 内，而程序执行在另一个核的 L4 之上，实现更加彻底的隔离[8]。

3.2 内存管理

对内存管理的设计，其实也就是对地址空间共享程度的设计。按照 L4 的设计，一个任务(相当于一般操作系统中的进程)中所有的线程共享一个地址空间。如果想实现并行计算，即一个任务中的多个线程运行在不同的核上的话，必须让地址空间对所有的核都是可见的。也就是说更改地址映射等工作必须通知其他核，这无疑增加了系统的复杂性。当然，也可以让每个核上的线程对应一个地址空间，并对所有的地址空间同步，但这无疑成倍地增加了内存的开销，并且同步所有的页表开销也很大。此外设计时还会遇到 L4 中存放内存分配映射关系的数据库 MapDB 是多个核共享还是每个核拥有一个的问题。由于 MapDB 中保存了内存分配和映射的层次关系，而这种关系难以从一个核复制到另一个核上。除非系统完全不允许线程的迁移，即线程永远都运行在同一个内核上，否则分离的 MapDB 架构难以实现。

综上所述，考虑到多核架构共享 L2 Cache、多核之间通信较快的特点，本设计采用同一任务的多个线程运行在不同核上时共享同一地址空间和 MapDB 的设计。这个架构允许并行计算，给整个系统带来了很大的灵活性。

此外由于本设计采用了离散的架构设计，并且多个操作系统内核之间共享 MapDB，设计时需要对操作系统内核本身的地址空间做一下处理。对于内核地址空间中非共享的部分，多个内核会分配到不同的物理地址。而对于共享的部分，会使用相同的物理地址。

3.3 线程调度

对于线程调度，最理想的情况自然是完全自由的全局调度，内核把线程分配到空闲的核上，并进行全局的负载平衡。但全局的调度有诸多问题。首先，L4 的调度策略是最高优先级的线程占有 CPU，如果多个线程有相同的优先级，则轮流被执行。如果在多核架构下，这些最高优先级的线程数少于处理器核心的数量，则就算有核空闲也只能空跑，造成了资源的浪费。其次，L4 有所谓“捐赠”的机制，被阻塞的线程可以把自己的时间片贡献给依赖的线程，使自己需求的资源尽快释放或者自己想要的服务尽快被执行。但对于多核架构，线程所拥有的时间片只是针对自己所在的核心，如果要把依赖的线程迁移到自己所在的核运行自己的时间片，然后再迁移回去，情况就变得很复杂。此外，由于本设计使用了离散模型，同步全局的等待队列也是件开销很大的事情。

因此，本设计退而求其次，对于细粒度的调度，系统只支持在操作系统内核内部，即同一个处理器核心中进行，系统也支持同一个核内部的时间片捐赠。但系统会提供一种称为线程迁移的方法，把线程从一个核转移到另一个核。具体操作就是保存线程的状态并在当前的核上删除，然后在另一个核上重新创建线程。对于多核之间的负载平衡，系统采用用户态的应用程序通过线程迁移实现。

3.4 锁和中断

L4 中提供的关中断和 Helping Lock 这 2 种机制都难以适应多核架构的环境。关中断只能屏蔽来自当前处理器核心的中断，而 Helping Lock 是指被阻塞的线程贡献时间片给拥有锁的线程来加快资源释放，这在多核架构下需要线程在 2 个核间来回迁移，实现机制复杂。综上所述，关中断和 Helping Lock 依旧会被保留，但只能用在处理器核心内部的同步，还需要寻找其他方法来实现多核之间的锁和同步。

Intel 架构的多核处理器提供一种叫总线锁的机制。系统

可以使用 LOCK#信号或者 LOCK 指令前缀来在执行访问内存命令时锁定总线，其他处理器核心对总线的访问会被屏蔽。但正如 Intel 的开发手册[5]所述，总线锁的作用是保证对内存的 byte 和 word 级别的读写是原子操作，如果访问的是大块的内存区域，总线锁有可能被系统中断。

对于多核间的同步，本设计采用自旋锁或者处理器间中断的方式实现。自旋锁是最简单的同步机制，线程在等待循环中不断查看自旋锁的状态，直到锁被释放。对于时间较短并且发生频率较低的同步，系统可以采用自旋锁的方式。Intel 64 以及 IA-32 架构提供处理器间中断机制(Inter-Processor Interrupts, IPI)。一个核可以通过 IPI 使另一个核中断。操作系统可以通过发送 IPI 给目标核，使之执行中断向量中指定的线程。

基于多核架构的特点，利用类似 mailbox(邮箱)的系统来同步也是一个很好的方法。mailbox 就是内存中一些所有核都知道的固定地址，一个核将信息放到 mailbox 中并通过 IPI 来通知其他核来读取。由于 mailbox 的内存地址会被频繁使用，这些内存地址很可能被放到 L2 Cache 中，而又因为多核架构共享 L2 Cache，对 mailbox 的访问速度会很快。此外还可以在实现 mailbox 时，通过 PREFETCHh 指令来暗示处理器核心把 mailbox 对应的内存读入 L2 Cache，更加保证了 mailbox 的读写速度。

3.5 多核间的 IPC

当处于不同核上的 2 个线程要进行进程间通信(Inter Process Communication, IPC，对于 L4 来说其实是线程间通信，但本文保留惯用的称呼)的时候，这就是一个多核间的 IPC，显然 L4 原来的 IPC 机制无法满足要求[9]。一种解决方法是改造原有的 IPC 机制，使之适应跨核的情况，但首先这种修改非常复杂，代码结构难以兼顾 2 种情况；此外本地(核内部)IPC 所采用的寄存器和内存的传递方式对于多核间 IPC 不是很有效率；最重要的是多核间 IPC 相对于本地 IPC 仍是少数情况，如果因为修改原来的 IPC 机制造成本地 IPC 效率下降就得不偿失了。因此，本设计会尽量保留原有的 IPC 机制，并用于本地的 IPC。

如图 3 所示，对于多核间 IPC，本设计采用 mailbox 加上 IPI 的机制来实现。这主要是要充分利用 2 个核之间共享 L2 Cache 的特点，实现高速的多核间 IPC 通信。但是由于新的 IPC 机制和原有的大不相同，没法透明地应用到那些 L4 中原有的没有为多核环境做特别设计的应用程序之上，还需要一个代理层来为这些应用程序提供多核间的 IPC 服务。本设计采用的是每个核都有一个代理线程的方式，线程想要进行多核间 IPC 通信，首先通过原本的本地 IPC 机制发送给自己所在核的代理线程，然后代理线程把需要传递的消息放到 mailbox，并通过 IPI 通知需要通信的核的代理线程读取，最后目标核的代理线程再通过本地 IPC 把消息发送到真正的 IPC 接收者线程。

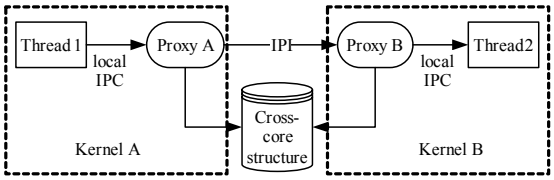


图 3 多核间的 IPC 示意图

3.6 应用程序

对于 L4 之上运行的应用程序，如果应用程序的作用范围仅仅是本地内核，在现有的架构下基本不需要修改，可以直接使用。如果应用程序作为服务程序需要接受来自其他核的请求，可以有主从模式和离散模式 2 种实现方案。主从模式就是指应用程序的主服务程序运行在某个核之上，而其他核上运行的是一个小的从服务程序，负责接收所在核上的请求并转发给主服务程序。离散模式是指每个核上的服务程序处理自己核上的请求，并通过多核间 IPC 和 mailbox 系统同步。

4 结束语

本文提供了一种基于双核架构的 L4 微内核操作系统的设计方案。该方案在兼顾兼容性和代码复杂性的基础上，充分利用多核架构共享 L2 Cache 的特点，实现了较为高效的微内核操作系统，更好地发挥了微内核架构在嵌入式环境中的优越性。

参考文献

[1] Völp M. Prototypical Design and Implementation of L4-SMP Micro-kernel Mechanisms[D]. Karlsruhe, Germany: Universitat Karlsruhe, 2002.

[2] Potts D, Winwood S, Heiser G. Design and Implementation of the L4 Micro-kernel for Alpha Multiprocessors[D]. Sydney, Australia: University of New South Wales, 2002.

[3] Schneider S. Multiprocessor Support for the Fiasco Micro-kernel[D]. Chemnitz, Germany: Technical University of Chemnitz, 2006.

[4] Intel Corp.. Intel Architecture Software Developer's Manual[Z]. 2005.

[5] 何 军, 王 飙. 多核处理器的结构设计研究[J]. 计算机工程, 2007, 33(16): 208-210.

[6] 王 峥. 微机应用系统中的可靠性剖析[J]. 微电子学与计算机, 2006, 23(11): 74-77.

[7] 赵立业, 张 激, 游 夏. 实时操作系统的性能分析和评估[J]. 计算机工程, 2008, 34(8): 283-285.

[8] 陈幼雷, 黄 强, 沈昌祥. 操作系统可信增强框架研究与实现[J]. 计算机工程, 2007, 33(6): 12-14.

[9] Schierboom E G H. Verification of Fiasco's IPC Implementation[D]. Nijmegen, Holland: Radboud University, 2007.

编辑 任吉慧

(上接第 248 页)

参考文献

[1] 巩向信, 葛益军. IEC61131-3 编程语言的现状与趋势[J]. 控制工程, 2007, 14(1): 99-101.

[2] 张海峰, 郑 明, 翟长远. PLC 编程软件中梯形图检查模块的设计[J]. 仪器仪表用户, 2007, 14(1): 72-73.

[3] 崔小乐, 周卓岑. 可编程控制器的梯形图语言与语句表语言的互换算法[J]. 微电子学与计算机, 2000, 16(1): 26-30.

[4] 葛 芬, 吴 宁. 基于 AOV 图及二叉树的梯形图与指令表互换算法[J]. 南京航空航天大学学报, 2006, 38(6): 754-758.

编辑 陈 文