第34卷 第23期 计 算 机 工 程 2008年12月
Vol.34 No.23 Computer Engineering December 2008
·工程应用技术与实现· 文章编号：1000—3428(2008)23—0252—03 文献标识码：A 中图分类号：N945

基于EFI和双核处理器的协处理器模型

张朝华，张申生
(上海交通大学计算机科学与工程系，上海 200240)

摘 要：为了提高双核处理器系统的性能，提出一种基于可扩展固件接口(EFI)、利用双核技术和IPI协议实现的协处理器模型。在EFI的DXE阶段，利用IPI协议引导启动一个与EFI系统并行的用户操作系统，将EFI系统作为一个后台服务系统(协处理器)处理用户系统的RPC请求。实验证明该分工协助的方式可以提高双核处理器系统的整体性能。

关键词：协处理器；可扩展固件接口；双核处理器；IPI协议

Coprocessor Model Based on EFI and Dual-core Processor

ZHANG Chao-hua, ZHANG Shen-sheng
(Department of Computer Science & Engineering, Shanghai Jiaotong University, Shanghai 200240)

【Abstract】 In order to improve the performance of dual-core computer system, this paper proposes a coprocessor model based on Extensible Firmware Interface(EFI) and utilizes dual-core technology and InterProcessor Interrupts(IPI) protocol. In the Driver Execution Environment (DXE)stage of EFI boot process, a user operating system is booted through IPI protocol, and the EFI system runs as a background service system, acting as a coprocessor handling RPC requests from the user operating system. Experimental result proves that the performance of dual-core computer system is improved compared with the one only running the user operating system.

【Key words】 coprocessor; Extensible Firmware Interface(EFI); dual-core processor; IPI protocol

1 概述

可扩展固件接口(Extensible Firmware Interface, EFI)[1]是Intel公司为解决传统BIOS无法启动Itanium架构平台的问题而设计的一种BIOS标准，并渐渐发展成为一个公认的下一代BIOS标准。Tiano是Intel公司基于EFI1.10 规范的一个开源代码实现。双核处理器[2]在基于单个半导体的一个处理器上拥有2个相同功能的处理器核心，即将2个物理处理器核心整合于一个核中。EFI/Tiano和双核处理器技术的出现使以双核处理器中的一个核作为通用协处理器使用成为可能，由它完成可能影响系统整体效率和某些有特殊需要的操作和 应用。

本文提出了一种协处理器模型，通过EFI引导系统进入双系统环境，然后通过RPC实现2个系统间的交互，并让其中一个系统作为另一个系统的服务提供者，即协处理器。为了验证本方法，实现了一个面向电子政务、安全计算等重要计算机系统的安全协处理器模型，使得原来需要通过专用设备解决的密码学和安全计算问题可以在普通的台式机上实现，降低了成本，方便了应用，且不影响原有系统的性能和效率，是一个比较好的综合解决方案。

2 相关技术

2.1 EFI/Tiano技术

EFI用于控制pre-Boot环境至进入操作系统的这段时间，在启动时，可以在操作系统和平台固件之间提供一个界面及一个初始化的支持。这个接口为 PC 硬件提供了与其操作系统互动的标准化方式。

EFI引导系统启动的过程如图1所示。其中，SEC是系统重启或开机后执行的第1个阶段，作为系统的安全自检阶段，检查固件的完整性。在PEI(Pre-EFI Initialization)阶段，系统初始化基本的处理器、芯片、主板、内存和其他资源，从而创建下一个阶段运行所需要的基本环境。在DXE(Driver Execution Environment)阶段，系统完成设备的枚举和初始化，实现EFI提供的服务、协议和驱动等。在BDS(Boot Device Selection)阶段，选择启动OS的设备。TSL(Transient System Load)是操作系统启动的过程，控制权交给用户操作系统。RT(Run Time)是指操作系统的运行期。在系统关机后，进入AL(After Life)阶段。

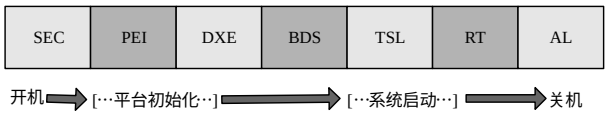

图1 EFI引导系统的启动过程

2.2 双核处理器技术

在双核处理器的构架中，每个处理器核拥有相当于未实现多线程功能的单处理器的独立微架构资源。每个核都有自己的APIC功能模块、PAT(Performance Acceleration Technology)、MCA(Machine Check Architecture)、调试寄存器和扩充寄存器。

2.3 IPI通信协议

IPI(InterProcessor Interrupts)[2]使一个CPU能够发送中断信号给系统中任何一个CPU，包括它自己。IPI不是通过IRQ线路发送的，而是作为一个连接到所有CPU的本地APIC总线

作者简介：张朝华(1978－)，男，硕士研究生，主研方向：嵌入式系统；张申生，教授、博士、博士生导师
收稿日期：2008-01-20 **E-mail：**leowa@sjtu.edu.cn

上的信号传递的。

2.4 RPC 通信技术

RPC(Remote Procedure Call)技术允许一个计算机程序通过调用另一个地址空间的子例程或过程来完成某个特定的功能，编程人员不需要在编程时给出这个远程交互的详细过程，也不用关心这个远程调用是在同一个计算机程序中还是在远程的某个计算机程序中运行的。

3 协处理器模型的设计

3.1 双系统的启动过程

这里的双系统是指 EFI 系统和用户应用操作系统，如 Windows, Linux。

在EFI引导系统启动的过程中，BDS阶段以后，EFI系统只保留Runtime Services[1]，其他接口都不再存在，由于Runtime Services并非运行于一个独立的系统上，而只是一些能够被用户操作系统调用的函数集，因此Runtime Services不能作为协处理器使用。

在多处理器系统中，多处理器初始化协议[2-3]描述了系统启动和初始化的过程。因为多处理器初始化协议只针对逻辑处理器，而双核处理器或多核处理器中的每个核都是系统中的一个逻辑处理器，所以多处理器的启动过程完全适用于双核或多核处理器系统。

在双核或多核处理器的环境下，EFI的启动过程通常会初始化所有的核，但是除了作为BSP[2]的逻辑处理器，其他逻辑处理器均被置于HALTED状态。

进入 DXE 阶段后，EFI 系统拥有完整的应用程序的开发环境，支持用户通过编写新的协议、驱动或 Shell 应用程序来扩充 DXE 阶段的功能。在 EFI/Tiano 下已经实现了一个名为 MPService 的多处理器协议，该协议遵循多处理器初始化协议和 IPI 协议的规范，封装了多处理器之间进行通信的 IPI 通信协议。

在 MPService 协议的基础上，通过编写一个 Shell 应用程序(以下称为 Boot 引导程序)来启动一个用户操作系统内核镜像文件，并让这个内核运行于双核处理器系统的另一个内核上，这部分工作已经由 Intel SSG 部门的 Tiano 组完成，并已集成到本协处理器模型中。

在引导用户操作系统启动完成以后，进入 EFI 和用户操作系统并行的双系统环境，整个启动过程如图 2 所示。

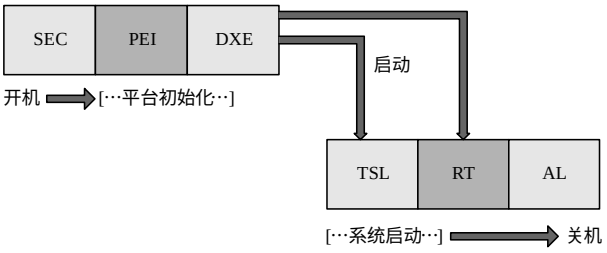

图 2 协处理器模型的启动过程

图 2 与图 1 的不同之处是没有经过 BDS 阶段，而是通过 Boot 引导程序直接开始操作系统的启动过程，EFI 则依然处于 DXE 阶段，从而可以作为后台服务系统继续运行。

3.2 四层架构的体系设计

在双系统启动之后，为了实现协处理器模型，必须让 2 个系统能够相互通信。由于运行的 2 个系统都能够访问系统的物理内存，因此可以通过共享内存的方式实现这 2 个系统之间的通信。

在实验中，改造了使用的 Linux 内核，避免 Linux 内核启动代码覆盖 EFI 使用的内存空间。

在此基础上，通过分配EFI系统和用户操作系统都能访问到的物理内存来实现跨系统的共享内存访问。首先在EFI端调用Boot Service 的AllocatePages[1]方法，分配得到一定的物理内存，并把得到的物理内存的地址和大小写到一个固定的物理内存位置处，在Linux系统启动完成后，协处理器设备驱动只须读取这个物理地址上的值就可以得到共享内存的信息。

在实现 2 个系统间基本通信的基础上，本文实现了如图 3 所示的系统架构。

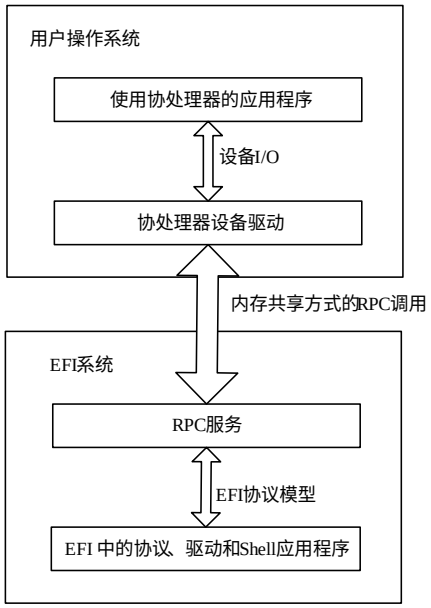

图 3 协处理器模型的 4 层架构

对于用户操作系统的应用程序，EFI 协处理器系统相当于一个虚拟的硬件设备，应用程序通过设备 I/O 通信的方式与 EFI 协处理器系统进行通信，让 EFI 协处理器完成它所支持的任务。

协处理器设备驱动实现了一个用户操作系统下的设备驱动，负责处理应用程序的设备 I/O 请求，将请求封装成相应的 RPC 请求发送给 EFI 系统，等待 RPC 请求的回复，然后将结果返回给发起 I/O 请求的应用程序。

EFI 系统中的 RPC 服务是作为 DXE 阶段的一个协议运行的，它接收来自用户操作系统中协处理器设备驱动的 RPC 请求，根据 RPC 请求的内容调用相应的 DXE 协议、驱动或 Shell 应用程序来完成任务，并将任务执行的结果封装成 RPC 包返回给用户操作系统的协处理器设备驱动。

在EFI系统中，协议、驱动和Shell应用程序是EFI提供的接口的组成部分，它们遵循EFI的驱动-协议规范[1,4]，因此，可以很方便地被RPC服务驱动调用，完成用户操作系统的调用请求。

3.3 模型的优点

协处理器设备和 RPC 服务驱动隐藏了 2 个系统间通信和 RPC 调用的复杂细节，使得本模型两端的用户系统上的应用和 EFI 系统中的扩展应用不需要关注中间的通信过程。

对于应用软件的开发者而言，不论是开发用户操作系统上的应用程序还是 EFI 上的扩展应用，本模型都不需要额外

的学习。在用户操作系统端，协处理器设备驱动和其他 I/O 设备并无区别，开发者可以像使用其他 I/O 设备一样使用协处理器设备，完成 EFI 系统支持的功能；而在 EFI 系统端，开发者只需要遵循 EFI 系统 DXE 阶段协议、驱动或应用程序开发的规范即可。

协处理器模型的另一个突出优点是减轻了用户操作系统进程调度的负担。通过将那些计算复杂度高的任务交给协处理器处理，减少了系统中需要经常调度的进程数或线程数，有助于用户系统整体响应速度和性能的提高。

4 实验设计和结果

为了验证协处理器模型能有效提高系统的整体性能，设计了下面的实验。

整个实验分为 2 个部分：

(1)Linux 系统中的一个 SHA-256 散列算法测试程序；

(2)EFI 系统上实现 SHA-256 算法的一个 EFI 协议。

实验目的是验证能够通过协处理器生成 SHA-256 报文摘要，然后将结果返回给 Linux 系统，并且协处理器模型在整体性能上优于单系统环境。

实验流程如下：

(1)测试程序通过发送一条生成 SHA-256 报文摘要的请求给协处理器设备；协处理器设备接收到请求后，找到生成 SHA-256 报文摘要对应的 EFI 接口的 GUID，根据接口要求封装成 RPC 请求发送给 RPC 服务驱动。

(2)RPC 服务驱动接收到这个 RPC 请求后，对其进行解析，根据接口的 GUID 加载 EFI 系统中的 SHA-256 协议，调用它的接口函数生成 SHA-256 报文摘要，然后将生成的摘要封装成 RPC 回复返回给协处理器设备。

(3)协处理器设备读出 RPC 回复包，解析以后，将摘要返回给测试程序，测试程序对返回的结果进行验证。

为了模拟真实的操作系统运行环境，本文利用 Linux 下的负载软件 Stress 模拟系统负载的增加，然后在相同的系统负载下比较采用协处理器模型和不用协处理器模型的环境下，使用 SHA-256 算法生成 SHA-256 报文摘要的效率。实验数据采用了从 http://csrc.nist.gov/cryptval/shs.htm 上下载的 sha256longmsg.txt.数据文件。测试机的配置是英特尔酷睿 2.8 GHz 双核 CPU，1GB 内存。2 种方法生成 SHA-256 报文摘要的效率比较如表 1 所示。

表 1　2 种方法的实验结果

Stress	Linux + EFI/s	Linux only/s
1	10.23	30.76
2	10.64	46.00

第 1 行数据运行的压力测试为

stress --cpu 100 --timeout 30 --backoff 2500

第 2 行数据运行的压力测试为

stress --cpu 100 --timeout 45 --backoff 2500

相对于第 1 个压力测试，第 2 个压力测试的负载更大，因为负载软件 Stress 运行的时间变长了。

需要说明的是，上面的测试都是在协处理器模型的系统中运行的，不同的是，Linux + EFI 使用协处理器设备来完成 SHA-256 的测试，而 Linux only 没有利用协处理器设备，直接在 Linux 系统中同时运行 Stress 负载软件和 SHA-256 的测试。

实验结果说明：

(1)完成相同计算量的任务，协处理器设备所需要的时间明显少于 Linux 系统所需的时间；

(2)随着 Linux 系统负载的增加，Linux 系统处理相同计算量任务所需时间的增长比例远远大于协处理器设备。

5 应用前景

由于协处理器设备适合处理那些运算量比较大的工作，并且实际的工作是在与用户操作系统隔离的 EFI 系统中完成的，具有更高的安全性，因此本模型在计算机安全领域具有广泛的应用前景。

例如，在 EFI 系统中增加加密和解密协议，将协处理器设备作为加解密设备来使用；让 EFI 管理密钥，增加数字签名协议，将协处理器设备作为 USB 智能卡来进行用户身份的认证。

6 结束语

本文提出了一种基于 EFI/Tiano 和双核技术的协处理器模型，并通过实验验证了其可行性和对系统整体性能的改善。基于 RPC 的 4 层架构使进一步扩充应用变得较为容易和直接。

进一步的研究方向是：

(1)实现基于处理器之间通信的 IPI 协议双系统通信的同步原语。在 EFI 和用户系统中都增加一个 IPI 中断处理例程，通过这个例程调用本系统内部的同步原语，从而实现跨系统的同步原语，然后通过这种同步原语实现跨系统 RPC 调用的同步性。

(2)在 EFI 系统中实现一个动态部署协议，这样就可以通过在用户操作系统环境中传给 EFI 系统一个新的 EFI 协议模块来动态扩展协处理器设备的功能。

参考文献

[1] Intel Corporation. Extensible Firmware Interface Specification(Ver. 1.10)[DB]. (2002-12-20). http://download.intel.com/echnology/efi/docs/EFI_110.zip.

[2] Intel Corporation. Intel Architecture Software Developer’s Manual: System Programming[DB]. (2006-01-15). http://download.intel.com/design/PentiumII/manuals/24319202.pdf.

[3] Intel Corporation. Multiprocessor Specification(Ver. 1.4)[DB]. (1997-05-10). http://download.intel.com/design/archives/processors/pro/docs/24201606.pdf.

[4] Intel Corporation. EFI 1.10 Driver Writer's Guide(Ver. 0.9)[DB]. (2004-07-20). http://download.intel.com/technology/efi/docs/pdfs/EFI1.10_DWG_0-9.pdf.