基于 ARM11 MPCore的多核间通信机制研究

邢向磊　周　余　都思丹
(南京大学电子科学与工程系　江苏 南京 210093)

摘　要　嵌入式应用中采用 SMP(对称多处理)系统所面临的主要难题是多处理器内核之间的通信。对 ARM11 MPCore处理器的多核间通信机制进行研究,并结合 Linux-2.6.19对这一通信机制的具体实现作深入分析,并在 RealView Emulation Baseboard上面进行相应的验证。实验结果表明,多核间通信机制可以使多线程之间的交互时间减小为原来的 16.7%,从而提升并行计算系统的性能。

关键词　ARM11　MPCore　多核间通信　Linux　处理器间中断

ON INTER-PROCESSOR COMMUNICATION MECHANISM BASED ON ARM11 MPCORE

Xing Xianglei　Zhou Yu　Du Sidan
(Department of Electronic Science and Engineering, Nanjing University, Nanjing 210093, Jiangsu, China)

Abstract　Inter-processor communication is a main problem in embedded applications adopting SMP (Symmetrical multi-processing) system. Based on the study of the inter-processor communication mechanism of ARM11 MPCore processor, in this article it analyzes the implementation of this mechanism in depth with linux-2.6.19. The corresponding validation has been done on the RealView Emulation Baseboard. The result of the experiment indicates that the inter-processor communication mechanism can reduce the interactive time of multithreading down to 16.7% of the original sequentially this raises the performance of the parallel computation system.

Keywords　ARM11　MPCore　Inter-processor communication　Linux　Inter-processor interrupt

0 引　言

多处理器时代已经悄然到来。MPCore可以综合多处理器基于 ARMv6体系结构,可配置为 1到 4个处理器,性能可达 2600 Dhrystone MIPS,在 MPCore多处理器上实现了 Adaptive Shutdown技术和 ARM智能电源管理技术,降低了高达 85%的功耗[1]。如图 1所示。

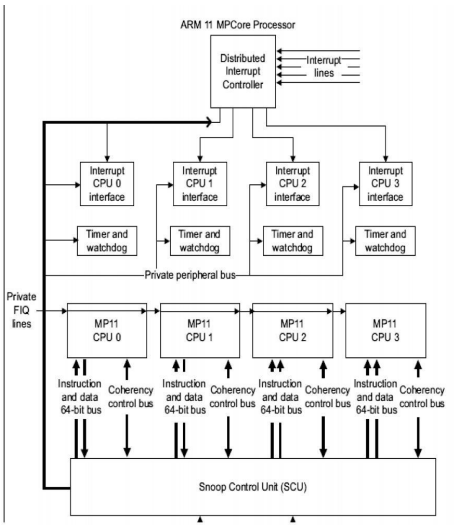

图 1　ARM 11 MPCore主要模块

构建一个并行计算系统,必须在系统不同部分之间建立通信通道。孤立的多个处理器不可能构建一个良好的并行计算方案。对于有效的并行操作而言,采用多个处理器共享专用资源的方法来实现通信(例如对称多处理器系统中采用的共享存储)通常是不够的,必须增加其他的方法来充分支持多个处理器之间便捷的交互。MPCore采用处理器间中断(IPI-inter processor interrupt)来实现多核间通信机制。通过中断系统来激活另一个处理器,由中断控制器触发其他的处理器,使他们根据中断 ID信息,进行各自的处理。MPCore多处理器核内集成了中断控制器,使访问中断系统更加高效。

1 MPCore分布式中断控制系统

MPCore在处理器内部引入了分布式中断控制系统,有效地解决了处理器间通信(inter-processor communication)问题。它由两部分组成:中断分发器(Interrupt Distributor)和 CPU接口单元(CPU Interfaces)。中断分发器收集所有的中断源,并把最高优先级的中断发送给相应的 CPU接口单元。CPU接口单元处理优先级屏蔽和中断嵌套,当一个中断获得处理后,将会在中断确认寄存器(Interrupt Acknowledge Register)中标记。CPU接口单元会记录下中断的优先级并在相应 MP11 CPU的中断分发器中将其标记为活动的。当中断处理完毕后,将在中断结束寄存器(End of Interrupt Register)中标记完成的中断 ID[2]。

收稿日期:2007-11-29。国家自然科学基金项目(60472026)。
刑向磊,硕士,主研领域:嵌入式系统,Linux系统。

处理器间中断(IPI)是中断源的一种,被标记为每个MPCore CPU的私有中断,中断ID为ID0—ID15,并且只能被软件触发。

在中断分发寄存器组中与处理器间中断(IPI)密切相关的是软中断寄存器(Software Interrupt Register),触发IPI中断需要向这个寄存器写入目标处理器(CPU target)和中断ID(interrupt ID)。

在CPU接口单元寄存器组中,与处理器间中断(IPI)密切相关的是中断确认寄存器(Interrupt Acknowledge Register)和中断结束寄存器(End of Interrupt Register)。响应IPI时从中断确认寄存器获取被请求的CPU和中断ID,测试中断ID是否为IPI(ID在0—15之间)。

确认是IPI后,跳转到do_ipi处理函数进行后续处理,并立即通知中断结束寄存器(End of Interrupt Register)。

2 多核间通信机制的作用分析

在并行计算系统中,采用IPI方案的一个最重要的原因是为了对多处理器上的多线程程序执行进行有效调度,减少多线程程序间交互的延迟时间。

操作系统在发生计时器事件或者类似的周期性外部中断的时候,都会重新对线程的执行进行调度。但是,如果某个线程触发了一个同步对象(这种情况会不规律的发生,而且不是周期性发生的),那么其他等待该对象的线程就必须执行。这些线程可以通过排队的方式在下一个周期性的中断信号来临时执行,但从处理器的角度来看,这种方法会相当浪费时间[3]。举例说明,有4个线程thread1、thread2 thread3、thread4分别运行在4个不同的处理器核上,thread2、thread3、thread4都阻塞在同一个信号量上,这是thread1触发了这个信号量(V操作)。这将唤醒thread2、thread3、thread4并使他们从等待队列转移到各自处理器的运行队列当中。但是thread2 thread3 thread4通常不会立即执行,而是等到下一个时钟中断到来时由调度程序选择执行,这就是重新调度方案产生的延迟。

另外一种方法是在程序设计的时候允许线程检测与其相关的其他线程的状态,但是这种程序设计方法有时甚至不能成为并行。由于所有线程都是在抢占环境中执行,线程只有在已经抢占了执行权且状态变量也发生了改变的条件下,才能真正开始执行。两个条件缺一不可,所以轮询存储器中的某个状态变量的模型不会总是有效。与IPI方案的比较如图2与图3所示[3]。

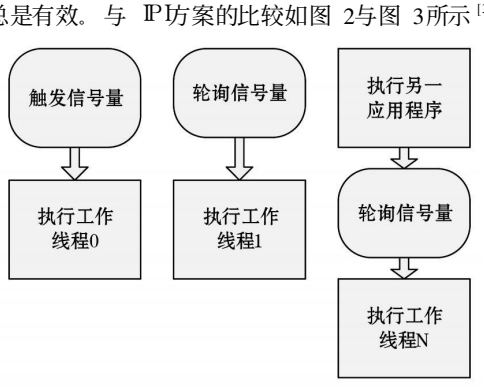

图2 程序设计中采用轮询同步线程的情形

通过采用IPI方案,系统就能够确保在同步对象被触发之后,所有等待该同步对象的线程都能够立即执行,并且能够在可预知的延迟(IPI延迟)之后开始执行。IPI延迟比一般的采用重新调度的方案所产生的延迟要小得多。

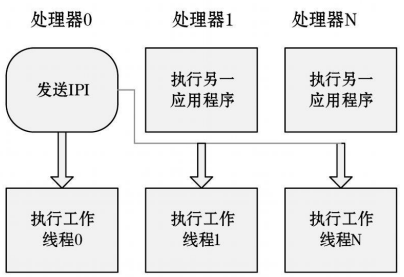

图3 并行程序设计中采用多处理器间中断的情形

3 多核间通信机制在linux内核中的实现

Linux内核在2.6.15版本后对MPCore进行了全面的支持。本文以Linux-2.6.19为基础对多核间通信机制的具体实现进行深入分析。多核间通信可分为通信发起部分和目标处理器响应部分。

3.1 通信发起

当一个CPU核准备与其它处理器核通信时,它将通过send_ipi_message(cpumask_t call_map enum ipi_msg_type msg)发送处理器间中断信息。输入参数包括目标处理器(全部在线处理器核或某些特定的处理器核)和消息类型。Linux-2.6.19下定义了四种消息IPI_TIMER、IPI_RESCHEDULE、IPI_CALL_FUNC、IPI_CPU_STOP,send_ipi_message函数的主要工作为:

(1)将消息以特定的形式保存成相应的目标处理器的私用数据。将IPI指针关联到目标处理器的私用数据结构,以"位结构"保存信息类型。Linux-2.6.19下定义了四种消息,所以传递的信息转换为具有0x0001、0x0010、0x1110、0x1111…等16种可能的ipi->bits位组合结构形式。并且这个私有信息将由目标处理器取出。

(2)触发相应目标处理器核的IPI中断:

```
static inline void smp_cross_call(cpumask_t call_map)
{   gic_raise_softirq(call_map, 1);
}
```

我们可以看到Linux-2.6.19将中断ID1用作IPI标记。其实在MPCore中每个MPCPU有自己的私有中断,其中ID0-ID15被用作IPI中断标记,并且被软中断触发。这里仅使用了ID1,其余留作扩展。对gic_raise_softirq进行分析:

```
void gic_raise_softirq(cpumask_t cpumask, unsigned int irq)
{
unsigned long map = *cpus_addr(cpumask);
writel(map << 16 | irq, gic_dist_base + GIC_DIST_SOFTINT);
}
```

可以得出,writel(map<<16|irq,gic_dist_base+GIC_DIST_SOFTINT)完成了写入目标处理器(CPU target)和中断ID(interrupt ID)到软中断寄存器(software interrupt register)中。由此触发了IPI中断。

3.2 目标处理器响应

目标处理器核被触发中断后,自动跳转(硬件执行)到中断异常向量表中相应的入口点,并进行如下工作:从CPU接口寄存器(CPU interface register)中的中断确认寄存器(Interrupt Acknowledge Register)中获取当前最高优先级的中断ID,并验证是否为IPI中断,如果是则调用do_ipi进行后续处理。

(下转第110页)

pfnOrig = GetProcAddress(m_hModule, pszFuncName)

(3) 使用上面构建的机器码，用 WriteProcessMemory函数重写这个函数的前 8字节：

WriteProcessMemory(..., GetCurrentProcess(), (void *) pfnOrig, bt-NewBytes, sizeof(DWORD) * 2, NULL);

现在，当一个线程调用这个 API函数(由 pszFuncName指定)时，JUMP指令会将函数调用重定向到自定义函数(由 pfnHook指定)，而在这个函数中可以执行任何代码。

参 考 文 献

[1] 郑阿奇. Visual C++ 实用教程[M]. 北京：电子工业出版社，2005.

[2] 王正军. Visual C++ 6.0程序设计从入门到精通[M]. 北京：人民邮电出版社，2006.

[3] 王艳平. Windows程序设计[M]. 北京：人民邮电出版社，2005.

[4] 牛力，等. Visual C++. Net编程宝典[M]. 北京：电子工业出版社，2006.

[5] 导向科技. 注册表应用一点通[M]. 北京：人民邮电出版社，2005.

(上接第 10页)

3.3 ipi中断响应函数 do_ipi

asmlinkage void do_IPI(struct pt_regs * regs)这个处理器间中断处理函数中：

(1) 获得当前的处理器 ID；

(2) 获取当前处理器保护的 IP数据结构；

(3) 中断处理前保存寄存器组；

(4) 增加 IPI计数；

(5) 从 ipi->bits中提取信息；

(6) 根据消息的类型进行相应的中断处理过程。

4 IPC机制在 Linux内核中的实验应用

IPC的一个主要作用是为了对多处理器上的多线程程序执行进行有效调度，减小多线程程序间交互的延迟时间。当一个线程触发信号量唤醒阻塞在该信号量上的其他线程时将调用 try_to_wake_up而 try_to_wake_up内部调用 smp_send_reschedule最终由 send_ipi_message实现 IPC机制，经处理器内部中断控制系统向目标处理器发中断信号，使目标处理器立即处理被唤醒的线程。

IPC减小多线程交互延迟的作用可以通过如下方案验证。

(1) 利用 CPU的亲缘性(CPU affinity)，将 4个线程指定在 4个不同的 CPU核上。

(2) 使 thread2、thread3、thread4阻塞在同一个条件变量上，再由 thread1广播该条件变量，唤醒其他线程。

(3) 分别在采取 IPC机制与常规调度机制情况下，测量线程被唤醒的平均响应时间。

在 RealView Emulation Baseboard上的测试结果如表 1所示。

表 1 IPC与常规调度的多线程交互时间对比

响应时间(μs)	CPU1	CPU2	CPU3	平均时间
常规调度	2632	7945	5228	5268
采取 ipi	313	811	1509	878

从测试数据得：采取 IPC机制后平均响应时间由 5268μs减少到 878μs即多线程间的交互时间减少为原来的 16.7%，这说明采取 IPC机制可以提升多线程间的交互性能。

IPC机制在内核中的另一个主要应用是完成时钟中断的处理。在单 CPU系统中，时钟中断用来更新 jiffies，更新系统运行时间，更新资源消耗和处理器统计值等。在多核系统中每个活动的 CPU核也会维护一个 time_tick，而时钟中断却并非一定要触发每一个 CPU核。在 MPCore中，时钟中断仅触发 CPU0，而其它 CPU的系统时钟则是由 CPU0通过 IPC机 send_ipi_message(mask, IPI_TIMER)来维护的。

我们在 RealView Emulation Baseboard上对这一机制加以验证。查看 /proc/interrupt文件，如图 4所示。

```
~ # cat /proc/interrupts
           CPU0       CPU1       CPU2       CPU3
 33:     149184          0          0          0   GIC  RealView Timer Tick
 36:       1013          0          0          0   GIC  uart-pl011
 39:        116          0          0          0   GIC  kmi-pl050
 40:         10          0          0          0   GIC  kmi-pl050
 41:       8285          0          0          0   GIC  eth0
IPI:        447        917        373        205
LOC:     149631     149700     149681     149659
Err:          0
~ #
```

图 4 IPC机制在 MPCore时钟中断中的应用

可以看到：RealView Timer Tick在各个 CPU核上的中断计数。其中只有 CPU0有很大的计数，而 CPU1、CPU2、CPU3上的计数均为 0。这说明 MPCore采取 IPC机制简化了 SMP系统时钟中断的处理。

5 结 语

本文阐明了并行系统需要采用多核间通信机制的原因，以及如何实现这一机制。在硬件实现上选取以 ARM11 MPCore的处理器间中断实现方案为代表；在系统软件的实现上以 Linux-2.6.19内核为代表对这一机制进行了深入分析。并且分析了多核间通信机制在并行系统中的典型应用。

在多处理器或多核系统中，多核间通信机制扮演着一个基础而又重要的地位。理解和发展多核间通信机制，对于构建良好的并行计算方案，减小多线程之间的同步交互时间，提升多线程程序的性能等有着重要意义。

嵌入式系统设计者，一直努力使下一代产品具有更高的性能，同时又有更低的功耗。ARM11 MPCore通过在多个处理器核(可同时支持 4个)上实现应用的任务级并行提高了处理器的性能并且具有动态 CPU使能以及智能电源管理技术。这使得它非常适用于下一代高性能的无线、多媒体、移动消费产品。目前，Linux2.6内核已经对 MPCore进行了全面的支持。可以预见，基于 Linux操作系统和 MPCore的下一代嵌入式产品将会很快出现在广大消费者面前。

参 考 文 献

[1] ARM发布首个可综合多处理器内核[J]. 电子产品与技术，2004,6：103.

[2] ARM11 MPCore Processor Technical Reference Manual[R].

[3] Shameem Akhter, Jason Roberts. 多核程序设计技术-通过软件多线程提升性能[M]. 李宝峰，等译. 北京：电子工业出版社，2007.

[4] Bovet D, Cesati M. Understanding the linux kernel[M]. Oreilly November 2005.

[5] Robert Love. Linux内核设计与实现[M]. 陈莉君，等译. 北京：机械工业出版社.

[6] Claudia Salzberg Rodriguez. Linux内核编程[M]. 陈莉君，等译. 北京：机械工业出版社.