龙芯 2 号处理器的同时多线程设计

李祖松[1),2)] 许先超[1),2)] 胡伟武[1)] 唐志敏[1)]

1)(中国科学院计算技术研究所计算机系统结构重点实验室 北京 100190)

2)(中国科学院研究生院 北京 100039)

摘 要 提出了适合龙芯 2 号处理器的同时多线程处理器模型,并介绍了具体的微体系结构设计以及相应的 Linux 操作系统的实现方案.通过在设计的龙芯 2 号同时多线程处理器上启动 Linux 操作系统,并运行应用程序,例如 SPEC CPU2000,进行性能评测.结果表明,龙芯 2 号同时多线程处理器通过挖掘线程级并行性,将龙芯 2 号处理器的性能提高了 31.1%.

关键词 龙芯 2;同时多线程;微体系结构;Linux 操作系统

中图法分类号 TP302 **DOI 号**: 10.3724/SP.J.1016.2009.02265

Design of the Simultaneous Multithreading Godson-2 Processor

LI Zu-Song[1),2)] XU Xian-Chao[1),2)] HU Wei-Wu[1)] TANG Zhi-Min[1)]

1)(*Key Laboratory of Computer System and Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100190)

2)(*Graduate University of Chinese Academy of Sciences, Beijing* 100039)

Abstract A design model and microarchitecture implement scheme of simultaneous multithreading Godson-2 processor is proposed in this paper. The operating system on simultaneous multithreading Godson-2 processor is designed for operating multiple individual threads simultaneously and improving system level performance. Linux operating system is booted in simultaneous multithreading Godson-2 processor and application programs such as SPEC CPU2000 are executed in it to evaluate performance. It has been shown that simultaneous multithreading Godson-2 processor improves the performance of superscalar Godson-2 significantly by full exploitation of instruction-level parallelism. The average speedup is 31.1%.

Keywords Godson-2; simultaneous multithreading; microarchitecture; Linux operating system

1 引 言

龙芯 2 号处理器[1]是一个四发射 9 级流水线的超标量超流水线处理器.整条流水线分为取指、预译码、译码、寄存器重命名、分配队列项、发射、读寄存器、执行并写回和提交等 9 个流水级,微体系结构如图 1 所示.图中指令顺序提交队列有 32 项,允许 32 条指令在处理器内部乱序发射、乱序执行.龙芯 2 号通过寄存器重命名技术解决读后写(WAR)和写后写(WAW)两种假相关,使得更多数据准备好的指令可以先于数据没有准备好的指令执行.访存功能

收稿日期:2005-11-20;最终修改稿收到日期:2009-07-08.本课题得到国家"九七三"重点基础研究发展规划项目基金(2005CB321600)、国家自然科学基金杰出青年基金项目"计算机系统结构研究(60325205)"、国家"八六三"高技术研究发展计划基金项目"大规模片上多处理器高性能存储系统研究(2007AA01Z114)"、国家自然科学基金项目"共享二级 Cache 的片上多处理器 Cache 块分布技术研究(60703017)"、国家自然科学基金重点项目"高性能片上存储系统(60736012)"资助.**李祖松**,男,1977 年生,博士,副研究员,主要研究方向为高性能计算机体系结构和功能验证. E-mail: lisoon@ict.ac.cn. **许先超**,男,1979 年生,硕士,主要研究方向为高性能计算机体系结构和操作系统. **胡伟武**,男,1968 年生,博士,研究员,博士生导师,主要研究领域为高性能计算机体系结构、并行处理和 VLSI 设计等. **唐志敏**,男,1966 年生,博士,研究员,博士生导师,主要研究领域为高性能计算机体系结构和并行处理.

部件实现了非阻塞的高速缓存，允许访存指令乱序执行，避免长延迟的访存指令阻塞其它访存指令的执行.该处理器采用 load 指令猜测执行技术避免 store 与 load 之间的假相关阻塞 load 指令的执行.在预译码流水级，采用转移预测技术，在转移条件还没有确定的情况下，对转移的方向和转移地址作出预测，进行取指执行，避免在转移条件确定之前流水线处于空闲状态.龙芯 2 号处理器通过上述转移预测、寄存器重命名、乱序执行、非阻塞的高速缓存、访存猜测执行等关键技术，促使指令在流水线中尽量流动起来，充分挖掘了指令级并行性.

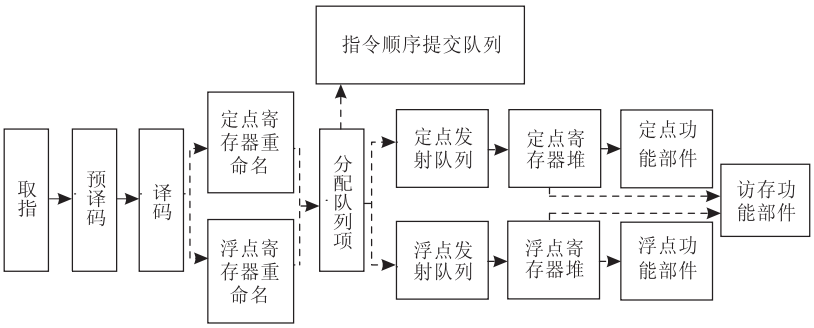

图 1 龙芯 2 号超标量处理器微体系结构

然而，这些超标量处理器的关键技术只能解决指令间的读后读(RAR)、读后写(WAR)和写后写(WAW)等假相关，单线程程序中指令间的写后读(RAW)真相关性依然存在.图 2 是完美的龙芯 2B 处理器(即在龙芯 2B 基础上分支预测永远正确和访存永远 Cache 命中)运行 SPEC CPU2000 的 IPC.从图中可以看出尽管分支预测永远正确和访存永远 Cache 命中，但 IPC 并不能达到四发射超标量处理器的理想值 4.这是由于单线程程序中指令间的真相关性，使得处理器无法每个时钟周期都能找到 4 条指令同时执行.

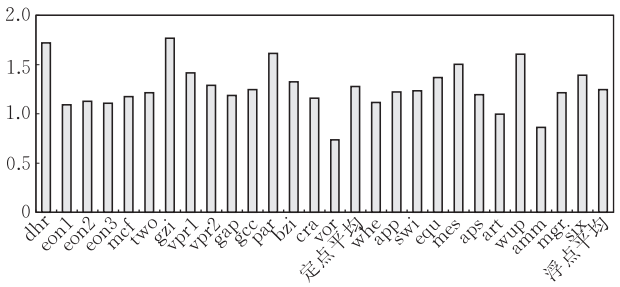

图 2 完美的龙芯 2B 处理器 SPEC CPU2000 的 IPC[2]

为了发掘更多可以并行执行的指令，一方面可以通过编译的循环优化、指令依赖检测及删除、指令调度、寄存器分配、软件流水等优化技术[3]，使进入硬件可见的指令窗口的不相关指令的数目增大.另一方面处理器设计一般可以采取如下两种方法.一种是对相关的值进行预测[4]，用预测的值先执行相关的指令，值确定后检查预测是否正确，预测错误就取消并重新执行.这种方法的硬件代价很大，目前仍然停留在研究阶段，还没有商用处理器实现.另一种方法是尽量找到更多不相关的指令.要找到更多不相关的指令也分为两种不同的方法，一种是扩大指令顺序提交队列，使得在处理器内部可并行执行的指令数增大，这样不相关的指令数也就增大，可以找到更多不相关的指令同时执行.但是不断扩大指令队列的效果却不断减小，最后当指令队列大到某一值后即使继续增大也无法再找到不相关的指令，而且增大指令队列也面对硬件代价大和复杂度高的问题.另一种方法是在一个处理器里执行多个线程的指令[5]，由于线程间的指令不相关，这样在相同大小的指令队列中就可以找到更多的不相关的指令来并行执行.这种方法也是目前许多通用处理器所采用的方法，也就是从传统的指令级并行向线程级并行发展.龙芯 2 号处理器要进一步提高处理能力，也需要引入多线程处理器技术.

本文主要介绍在龙芯 2 号处理器上实现同时多线程的设计方案.第 2 节介绍国内外关于多线程处理器技术的研究；第 3 节阐述龙芯 2 号同时多线程处理器的设计方案；第 4 节阐述龙芯 2 号同时多线程处理器如何处理存储一致性问题；第 5 节介绍操作系统及相应的软硬件接口；第 6 节介绍实验方法以及对测试结果进行分析；最后一节总结并介绍未来进一步的研究工作.

2 多线程处理器技术

当前流行的多线程处理器结构主要可分为两大类.一类是将原来的单进程程序划分成多个线程并

行执行,以达到对单进程程序加速的目的.这类技术根据硬件是否猜测执行划分出的线程而分为两个子类.进行线程猜测执行的处理器技术主要有多标量处理器(Multiscalar processors)[6-9],迹处理器(Trace processors)[10-11]、超线程处理器(Superthreaded architecture)[12]、动态多线程处理器(Dynamic multithreading processor)[13]、猜测执行多线程处理器(Speculative multithreaded processor)[14-15]等等.这类多线程处理器不仅要对每个执行线程提供一条独立的流水线,而且还要暂时保存猜测执行线程的执行结果,并且要进行复杂的数据相关的检查,这些都导致这类处理器需要消耗大量的硬件资源使得成本急剧增大,而且设计复杂度也大大增加,设计实现变得困难,设计正确性的保证也是困难重重.对单进程程序加速的多线程处理器的另一子类是不对线程进行猜测执行,也就是不由硬件自动划分出线程,而是由编译器划分多个线程.由编译器从单线程程序中划分出来的线程称为微线程.微线程技术也有两种,一种是同时辅助微线程(Simultaneous subordinate microthreading)[16],实现这种微线程技术的处理器在内部用一个 RAM 来保存微线程,执行的线程可以通过调用微线程来辅助原线程的执行.另一种是微小线程(Micro-threading)[17],由编译器生成多个微线程并行执行,微线程之间通过寄存器进行数据交换和同步.

多线程处理器技术的另一大类是以同时多线程和片上多处理器为代表的,通过在一个芯片上执行多个互不相关的线程,来增加整个芯片每一拍并行执行的指令数.这类技术根据多个线程是否共用同一条流水线又可以分为两个子类,其中一类是共用同一条流水线的细粒度多线程(Fine-grained Multithreading)[18]、粗粒度多线程(Coarse-grained Multithreading)和同时多线程(Simultaneous Multithreading,SMT)[19-21],另一类是每个线程各自拥有一条完整流水线的片上多处理器技术(Chip Multiprocessor,CMP)[22].

同时多线程是在超标量处理器的结构上,同时保持多个线程上下文处于活动状态,在同一个时钟周期内同时执行多个线程的指令.这样,同时多线程处理器就能够利用多线程间的指令不相关,找到更多可以并行执行的指令,而且当某一线程遇到长延迟的指令时,可以从其它的线程中寻找数据准备好的指令继续执行,避免了长延迟指令导致处理器长时间处于闲置状态.同时多线程处理器中的每一个线程除了保存有各自的上下文外,共用同一条流水线和所有的功能部件,因此同时多线程处理器与原来的超标量处理器相比,硬件资源增加不多,芯片面积也增加不大.同时多线程处理器最大的优点正在于通过增加很少的芯片面积,就能够获得很高的资源利用率,即用较少的成本换取较大的并行性能,充分地挖掘了线程级的并行.例如 Intel 的 P4 处理器[23]使用同时多线程技术仅增加 5%的芯片面积,而性能提高了 30%.

目前的商用处理器采用的技术主要是同时多线程和片上多处理器技术,例如 Intel 的 P4 处理器的 Hyper-thread[23]技术实现同时执行两个线程,IBM 的 Power5[24]处理器每个芯片有两个内核,每个内核可以同时执行两个线程 SUN 的 Niagara 处理器[25]每个芯片有 8 个内核,每个内核可以同时执行 4 个线程.

3　龙芯 2 号处理器的同时多线程设计方案

龙芯 2 号处理器的同时多线程设计方案支持两个线程,有如下两种运行模式:

(1)超标量模式

(2)同时多线程模式

超标量模式和原来龙芯 2 号处理器相同,处理器只运行一个线程,此线程占用所有的硬件资源,包括队列、流水线通路、物理寄存器堆、功能部件、Cache 等.

同时多线程模式是在处理器上同时运行两个线程,这两个线程来自不同的程序.每个线程拥有独立的程序计数器(PC)、逻辑寄存器、控制寄存器等,其它的资源包括队列、流水线通路、功能部件、Cache 等由两个线程共享.线程共享资源针对不同资源的特点采用 3 种不同的共享方式.第 1 种共享方式是各自占用一半,例如指令顺序提交队列等.第 2 种共享方式是不限制只用一半但限制不可以完全占用,例如定点发射队列等.第 3 种共享方式是轮流使用即分时占用,例如运算部件、流水线通路等.

龙芯 2 号同时多线程处理器的微体系结构如图 3 所示.由于要支持两个线程的执行,需要同时保存两个线程的上下文,所以同时多线程处理器比原来的超标量处理器增加了一些硬件资源.增加的硬件资源主要包括 PC、定点浮点寄存器重命名、定点浮点寄存器堆和控制寄存器等.其它资源则根据资源的特点不同选择不同的共享策略进行共享.下面按照流水线的顺序介绍龙芯 2 号同时多线程处理器的微体系结构.

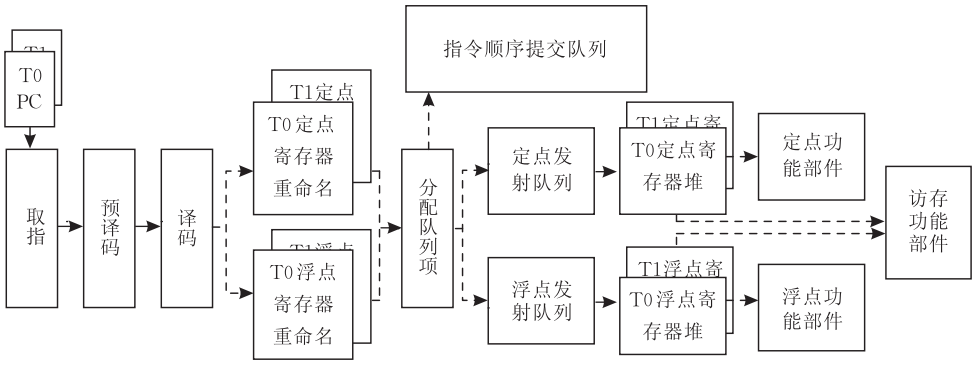

图 3 龙芯 2 号同时多线程处理器的微体系结构

龙芯 2 号同时多线程处理器取指结构如图 4 所示. 传统的同时多线程要从两个线程同时取指,需要一个双端口 RAM 来实现指令 Cache. 然而从表 1 可以看出,双端口 RAM 的面积是单端口的 2.8 倍,如果使用双端口 RAM 将导致指令 Cache 的面积急剧增大,延迟也有所增加. 因此龙芯 2 号同时多线程处理器仍然使用单端口 RAM 来实现指令 Cache,每一个时钟周期选择指令 Buffer 中指令数较少的线程进行取指. 指令 TLB 单线程时占用所有的 16 项,双线程时每个线程使用 8 项. 转移预测器部分,由于转移历史寄存器和地址返回栈是与每个线程上下文相关的,所以每个线程独立一套. 而用于预测寄存器跳转的 BTB 和转移模式表可以共用,也可以分用. 图 5 是分别实现共用转移预测装置和分用转移预测装置两种策略的龙芯 2 号同时多线程处理器的转移失效率的比较,从图中可以看出共用转移预测装置的转移失效率普遍低于分用转移预测装置的转移失效率,所以 BTB 和转移模式表两个线程共享. 取来的 4 条指令通过预译码的转移预测之后进入指令 Buffer,超标量处理器只有一个指令 Buffer,同时多线程处理器实现两个指令 Buffer,每个线程一个. 译码阶段,从处理器中正在运行的指令数较少的线程的指令 Buffer 中选择指令,最多不超过 4 条指令同时送往寄存器重命名流水线.

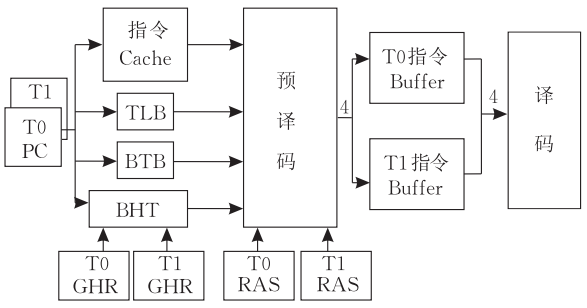

图 4 取指结构示意图

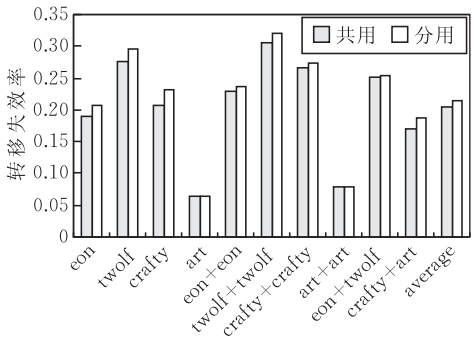

图 5 共用转移预测装置和分用转移预测装置的转移失效率比较

表 1 0.18μm cmos 工艺的 512×64 单双端口 RAM 比较

	面积/μm^2	延迟/ns
单端口 RAM	216258	1.384
双端口 RAM	611910	1.547

传统的同时多线程处理器中多个线程共用一个较大的物理寄存器器堆,大寄存器堆在物理实现上难度很大,而且对大寄存器堆及其重命名装置的访问延迟较大,会导致处理器时钟频率下降. 因此龙芯 2 号同时多线程处理器使用两个较小的寄存器堆来实现,每个线程独自占用一个寄存器堆. 虽然当某一个线程有空闲的物理寄存器时,另一个线程也无法利用这部分空闲资源,资源利用率不如整个的大寄存器堆,但是降低了物理实现的难度. 如图 3 所示,寄存器重命名逻辑就复制了一份,当执行双线程时每个线程独立使用一个寄存器重命名逻辑.

指令通过寄存器重命名之后进行队列项分配,也就是在该指令相关的队列中选择一个空项分配给此指令使用,包括定点发射队列、浮点发射队列、CP0 队列、转移指令队列和指令顺序提交队列. 这些队列是由两个线程共同使用的,它们的共享方式将影响流水线是否能够流畅地执行,特别是在某一线程遇到长延迟指令时,另一个线程能否不受影响继

续流畅执行,这也是发挥同时多线程优点的关键技术之一.

定点发射队列和浮点发射队列的特点相同,都是指令进入队列时可以放在任意的空项里,指令从队列中发射后,所占用的项就立即释放,指令也就在发射前的这段时间处在发射队列中.发射队列的共享方式是为每个线程至少保留 4 项,避免阻塞的线程过多的占用发射队列项,使得另一线程无法发射执行.

龙芯 2 号处理器的访存指令是乱序执行的,但为了保证程序的顺序性,也就是当两条指令的访存地址相同时,后执行的 load 指令获得的值是先执行的 store 指令写的新值,而不是数据 Cache 中原来保存的旧值,因此设计了 CP0 队列来保存指令的顺序,并且把程序中前面 store 指令的值传递给后面的 load 指令.CP0 队列的共享方式可以选择每个线程各占一半,各拥有一个 CP0 队列,各自独立的进行排序,不过要考虑两个队列之间访存地址相关指令如何执行,以及如何确定两个队列中的指令之间的先后顺序.这种方式的好处是当一个线程发生 Cache 不命中时,另一个线程仍然可以继续流畅地执行.另一种共享方式是两个线程按顺序都进入一个 CP0 队列,并且按顺序离开.这样两个线程的指令在 CP0 队列中就有了一定的顺序,不过这个顺序只是进入 CP0 队列的顺序,并不是指令真正执行的顺序.而且由于转移取消或者例外等原因,CP0 队列中有的项要取消,使得 CP0 队列中会有一些不连续的空项,为了保证 CP0 队列的顺序执行,新来的指令不能进入那些空槽.而且当一个线程发生 Cache 不命中时,它的指令将阻塞在 CP0 队列中,使得另一个线程会由于 CP0 队列已满而无法继续执行,因此本文采用第一种方案.

转移指令队列和指令顺序提交队列也和 CP0 队列一样,要求指令顺序进入,顺序离开,因此也将这两种队列设计成每个线程各有一套,避免相互影响.特别是其中的转移指令队列必须设计成各有一个,否则每个线程的基本块编号将变得混乱,转移取消机制也会有问题.每个线程有自己的转移指令队列,就使得每个线程的指令都有自己的基本块编号,当然两个线程的基本块编号会有相同的,转移取消时不仅要检查基本块号还要检查线程号是否一致才决定是否取消.

上面提到的定点发射队列、浮点发射队列、CP0 队列和指令顺序提交队列在超标量模式下,将所有的项都给一个线程使用,资源不会闲置.不过转移指令队列在多线程模式下每个线程有 8 项,在超标量模式下单个线程也只使用 8 项,因为转移指令还有一些关于寄存器的信息要保存用于取消时恢复,转移取消的逻辑比较复杂,把转移队列做到 16 项会导致延迟较长,不能在一个时钟周期内完成寄存器信息的恢复.

在同时多线程模式下,每个线程拥有自己的一套控制寄存器,但共享 64 项 DTLB.DTLB 的项不作划分,两个线程可以使用其中的任意一项,只是不可以同时访问.这个控制由操作系统完成.

其它的各个流水级都保留有指令的线程号.当发生例外时,指令不一定被取消,还需要比较发生例外的线程号,相同才取消.转移取消也是一样.

4　存储一致性模型

存储一致性模型多种多样,本节主要对弱一致性模型[26]、顺序一致性模型[27]和处理机一致性模型[28]进行分析.

龙芯 2 号同时多线程处理器的两个线程共享数据 Cache,当数据写入数据 Cache 后就不存在一致性问题了,但对于 CP0 队列中的访存指令就必须考虑一致性问题.最简单的做法是实现弱一致性模型.原来的超标量设计就支持同步指令顺序执行,并且保证同步指令执行时 CP0 队列中在同步指令之前的指令都写入数据 Cache,同步指令之后的指令都没有执行.因此原来的设计不作改动就可以支持弱一致性.又由于龙芯 2 号中的访存管理队列是顺序提交的,其中取数操作在提交前执行,存数操作在提交后顺序执行,这种执行方式满足处理机一致性模型的要求.所以龙芯 2 号同时多线程处理器也符合处理机一致性模型.

由于龙芯 2 号同时多线程处理器依然被当作一颗处理器,在上边运行的程序仍然认为是在单机上运行,因此龙芯 2 号同时多线程处理器还要求符合顺序一致性模型.龙芯 2 号同时多线程处理器的顺序一致性要求任意一个线程都严格按照访存指令在程序中出现的次序执行访存指令,且在当前访存指令彻底完成之前不能开始执行下一条访存指令.其中,一个存数操作“彻底完成”是指它所引起的值的变化已被所有的线程所接受,一个取数操作“彻底完成”是指它取回的值已确定且写此值的存数操作已“彻底完成”.下面我们举一个简单的例子来说明在

什么情况下执行会出错，并给出解决的方案，来保证顺序一致性.

如图 6 所示的程序段，这是保证只有一个进程进入临界区的一种同步机制. 当 T0 和 T1 分别执行完相应的指令时，R_1 和 R_2 的值的正确组合是(0,1)、(1,0)和(1,1)，只有 $R_1=R_2=0$ 的结果是错误的，它将导致 T0 和 T1 同时进入临界区.

Thread T0	Thread T1
L_{11}: store a, 1	L_{21}: store b, 1
L_{12}: load R_1, b	L_{22}: load R_2, a

图 6 程序段 PRG(初始值 $R_1=R_2=a=b=0$)

原来的龙芯 2 号超标量处理器中存数指令在提交之后才写数据 Cache，如果发生数据 Cache 不命中，它将暂时停留在 CP0 队列中，此时另一个线程还没有接受该存数指令的值，而存数指令之后的取数指令可以提交了，这就不满足顺序一致性. 如果两个线程都发生这种情况，就会得到错误结果 $R_1=R_2=0$.

为了解决上面提到的问题，龙芯 2 号同时多线程处理器对原来的 CP0 队列进行了必要的修改. 当一条取数指令进入 CP0 队列时，查找另一个线程的 CP0 队列中是否有地址相关的存数指令，如果有则把该条取数指令置例外重新执行. 同样的，当一条存数指令进入 CP0 队列时，查找另一个线程的 CP0 队列中是否有地址相关的取数指令，如果有则把该条存数指令置例外重新执行. 这样，通过 CP0 队列给两个线程间发生读写相关的指令置例外的机制，保证处理器不会同时处理两个线程间读写相关的指令，以此来保证一个线程写的值能够被另一个线程所读取. 至于写写相关，由于存数指令是顺序执行的，在提交时确定一个顺序就不会出错了，所以本方案不检查写写之间的地址. 而对于读读相关，不会引起错误，所以也不做检查.

总之，通过 CP0 队列间互相查找读写相关的指令置例外的机制保证处理器不同时执行读写相关的指令，以此使得同时多线程处理器满足了顺序一致性模型.

5 操作系统以及软硬件接口

龙芯 2 号同时多线程操作系统的设计目的就是利用龙芯 2 号同时多线程处理器可以同时运行两个独立的线程的功能，来提高整个软件系统的性能. 龙芯 2 号同时多线程处理器在同时多线程模式下可以同时支持两个完全独立的程序执行流，每个执行流都有自己一套完整的定点浮点通用寄存器和控制寄存器，虽然这两个执行流共享功能部件，但是数据处理是完全独立的，每个执行流都可以互不影响地进入内核态或切换回用户态. 从操作系统的角度看，同时多线程模式下的龙芯 2 号同时多线程处理器就像是含有两个完全独立 CPU 的 SMP 处理器，从而内核可以像对待 SMP 处理器一样来对它进行支持和利用. 因此，本文设计的龙芯 2 号同时多线程操作系统实际上就是一个龙芯 2 号处理器的 SMP 操作系统，但是这个操作系统在硬件相关方面的代码必须采用龙芯 2 号同时多线程处理器提供的接口.

由于 Linux 操作系统对 SMP 处理器有很好的支持，在处理器间同步、通信、内核资源同步和共享上已经比较成熟，所以我们利用它的 SMP 相关处理来实现对龙芯 2 号同时多线程处理器的支持，实现龙芯 2 号同时多线程 Linux 操作系统.

Linux 操作系统在对 SMP 处理器的处理中，主要对以下 4 个方面作了特别考虑：

(1)启动过程. Linux 先从主 CPU 上开始执行，初始化完主 CPU 后，再依次启动其它次 CPU，让每个 CPU 都处于一个正确的状态并准备好执行分配给它的进程；

(2)进程调度. Linux 将会为每个 CPU 调度一个需要运行的进程，充分利用每个 CPU；

(3)资源同步共享. Linux 中对不可重入代码采取了同步措施，并为每个可能出现竞争使用的资源增加了同步锁；

(4)CPU 间消息传递. 当一个 CPU 修改了内核的数据结构如页表，或处理了一个中断而这个中断的处理需要向另一个 CPU 运行的进程发送信号时，Linux 通过 CPU 间消息传递来通知另一个 CPU.

在这些处理中，进程调度与硬件没有任何关系，而且 Linux 针对 SMP 处理器的进程调度也比较完善，所以这部分不需要做修改. 其它 3 个方面则都与硬件相关，龙芯 2 号同时多线程处理器必须从硬件实现上对操作系统提供支持，操作系统也必须根据龙芯 2 号同时多线程处理器提供的接口进行修改. 启动过程方面，硬件将提供由主 CPU 初始化次 CPU 的控制寄存器的指令和启动次 CPU 的指令，Linux 通过这些指令来实现初始化和启动次 CPU 的函数 smp_boot_cpus(). 资源同步共享方面，Linux 中 MIPS 体系结构的处理器基本上都是通过

ll/sc 来实现 down 和 spin_lock 等同步函数,龙芯 2 号同时多线程处理器对 ll/sc 两条指令提供了完整的支持,这方面可以不作修改.另外,龙芯 2 号同时多线程处理器提供了 wait 指令,这条指令会使一个逻辑 CPU 停止取指并等待一定的时间,让另外一个逻辑 CPU 获得更多的硬件资源,从而提高另一个逻辑 CPU 上程序运行的速度.同步函数在尝试失败后,可以通过 wait 指令去停止一段时间,而不是不停地尝试,通过这种方法提高处理器的性能.在 CPU 间消息传递机制方面,龙芯 2 号同时多线程处理器通过 CPU 间中断来向另一个逻辑 CPU 来发送消息,硬件上增加了 CPU 间中断寄存器和 CPU 间中断屏蔽寄存器,并提供新的指令处理这两个寄存器.对 Linux 的中断响应函数作修改,以支持新的 CPU 间中断,并根据不同的 CPU 间中断号来确定传递的是什么消息.同时,Linux 的消息发送函数也作了相应的修改,将要发送的消息转化为 CPU 间中断发送给目标 CPU.

本文通过硬件提供支持以及对 Linux 相应接口的修改,实现了龙芯 2 号同时多线程处理器的操作系统.

6　实验方法及性能分析

龙芯 2 号的微体系结构在不断的改进中,我们在指令顺序提交队列 64 项、CP0 队列 32 项的龙芯 2 号处理器版本上实现了同时多线程处理器,处理器配置如表 2 所示.我们设计了龙芯 2 号同时多线程处理器的代码,并使用 Mentor Graphics 公司的 VStationPro 仿真加速器对 RTL 代码的仿真进行加速.龙芯 2 号同时多线程处理器的超标量模式与原来的龙芯 2 号超标量处理器结构是一样的,所以我们通过在龙芯 2 号同时多线程处理器的超标量模式和同时多线程模式下运行程序进行性能比较.同时我们在 Linux2.4.20 的代码上实现了对龙芯 2 号同时多线程的支持,通过在 Linux 操作系统平台上运行基准程序进行性能分析.

表 2　龙芯 2 号处理器配置

Fetch width	Decode width	ALU	FP Unit	Load/Store Unit	Roqueue	Fixqueue	Ftqueue	Cp0queue	DTLB	ITLB	BHT	DCACHE	ICACHE	片上二级 CACHE
4	4	2	2	1	64	16	16	32	64	16	4K	64Kbytes	64Kbytes	无

由于 VStationPro 上的内存资源有限,我们只能选择内存需求较小的程序运行,因此我们选择如下 3 类测试程序进行性能比较.第 1 类测试程序是每条指令都相关的程序,在超标量模式下运行两遍,而在同时多线程模式下两个线程同时执行这个程序.第 2 类测试程序是多线程程序 chat.第 3 类测试程序是 SPEC CPU2000 的 TEST 规模中的 eon、twolf、art、crafty,从中选择两个相同或不同的程序作为一组进行测试.我们通过在超标量模式和同时多线程模式下分别运行以上测试程序,统计各自执行的指令数和运行周期数,然后以每个周期完成的指令数(即 IPC)来评价其性能.

首先我们通过在超标量模式下和同时多线程模式下分别运行单个单进程的测试程序进行比较,也即是单进程负载下的性能比较.测试结果如图 7 所示.从测试结果可以看出,由于指令顺序提交队列和 CP0 队列在同时多线程模式下是分用的,也就是这两个队列中的一半闲置,没有得到利用,使得同时多线程模式下的性能不如超标量模式.当然由于处理器中的大部分资源是共享的,此时由一个线程独占,所以同时多线程模式下的性能与超标量模式相差不大.实际上,龙芯 2 号同时多线程处理器支持两种模式,允许实时地进行模式切换,操作系统可以在发现只有单进程负载时,从同时多线程模式切换到超标量模式,更好地发挥处理器的性能.

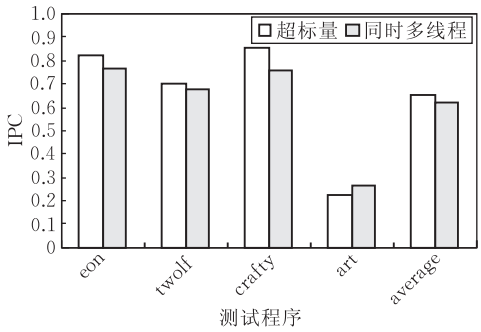

图 7　同时多线程与超标量的单进程负载性能比较

我们通过在超标量模式和同时多线程模式下分别运行前面提到的 3 类测试程序进行比较,也即是多进程负载下的性能比较.图 8 为测试结果,其中的平均 IPC 是不包括指令相关程序的其它应用的平均值.从测试结果中可以看出,类似第 1 类测试程序

这样相关性较强的程序，同时多线程性能提高99.8%，与理想情况仅差0.2%，说明了同时多线程结构充分提高了功能部件的利用率，在一个线程等待计算结果的时候，利用空闲的功能部件为另一个线程进行计算，使得运行两个程序的时间与运行一个的时间基本相同. 对于一般的应用，例如多线程程序chat，SPEC CPU2000的测试程序等，龙芯2号同时多线程处理器对其性能也有显著的提高，最多提高了54.5%，平均提高31.1%. 测试结果表明龙芯2号同时多线程处理器在多进程负载的情况下，利用多个线程之间互不相关的特性，在相同的指令窗口大小的情况下，寻找到更多的可并行的指令，充分挖掘功能部件的利用率，显著提高了龙芯2号处理器的性能.

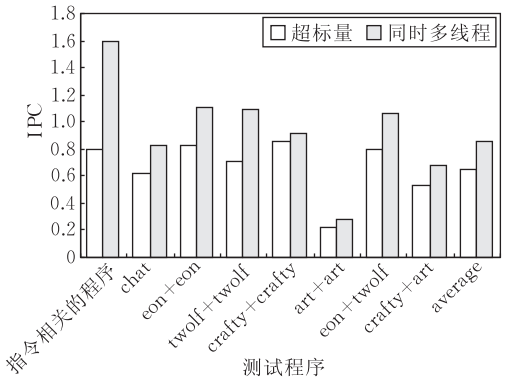

图8 同时多线程与超标量的多进程负载性能比较

7 总 结

龙芯2号同时多线程处理器结合龙芯2号处理器的结构特点，以挖掘程序的线程级并行能力、提高功能部件的利用率为切入，以提高龙芯2号处理器性能为目的. 本文详细描述了龙芯2号同时多线程处理器的结构设计以及相应的操作系统平台的实现，并且通过在操作系统平台上运行应用程序进行性能分析，证明龙芯2同时多线程处理器增强了龙芯2号处理器的处理能力. 相信经过对Linux操作系统的进一步改进，充分发挥其在两个逻辑CPU上调度不同的进程的能力，能够更大地提高龙芯2号同时多线程处理器的性能.

参 考 文 献

[1] Hu Wei-Wu, Zhang Fu-Xin, Li Zu-Song. Microarchitecture of the Godson-2 processor. Journal of Computer Science and Technology, 2005, 20(2): 243-249

[2] Zhang Fu-Xin. Performance analysis and optimizations of microprocessors [Ph. D. dissertation]. Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 2005(in Chinese)
(张福新. 微处理器性能分析与优化[博士学位论文]. 中国科学院计算技术研究所，北京，2005)

[3] Muchnick Steven S. Advanced Compiler Design and Implementation. San Francisco, California: Morgan Kaufmann Publishers, 1997

[4] Lipasti Mikko H, Shen John Paul. Exceeding the dataflow limit via value prediction//Proceedings of the 29th Annual IEEE/ACM International Symposium on Microarchitecture. 1996: 226-237

[5] Ungerer Theo, Robic Borut, Silc Jurij. Multithreaded processors. The Computer Journal, 2002, 45(3): 320-348

[6] Franklin M. The multiscalar architecture. University of Wisconsin-Madison, WI: Computer Science Technical Report No. 1196, 1993

[7] Sohi G S. Multiscalar: Another fourth-generation processor. Computer. IEEE Computer, 1997, 30(9): 72

[8] Sohi G S, Breach S E, Vijaykumar T N. Multiscalar processors//Proceedings of the 22nd ISCA. Santa Margherita Ligure, Italy, 1995: 414-425

[9] Vijaykumar T N, Sohi G S. Task selection for a multiscalar processor//Proceedings of the 31st International Symposium on MICRO, Dallas, TX, 1998: 81-92

[10] Rotenberg E et al. Trace processors//Proceedings of the 30th International Symposium on MICRO. Research Triangle Park, NC, 1997: 138-148

[11] Smith J E, Vajapeyam S. Trace processors: Moving to fourth-generation microarchitectures. IEEE Computer, 1997, 30(9): 68-74

[12] Tsai J-Y, Yew P-C. The superthreaded architecture: Thread pipelining with run-time data dependence checking and control speculation//Proceedings of the Conference PACT. Boston, MA, 1996: 35-46

[13] Akkary H, Driscoll M A. A dynamic multithreading processor//Proceedings of the 31st International Symposium on MICRO. Dallas, TX, 1998: 226-236

[14] Marcuello P, Gonzales A, Tubella J. Speculative multithreaded processors//Proceedings of the International Conference on Supercomp. Melbourne, Australia, 1998: 77-84

[15] Tubella J, Gonzalez A. Control speculation in multithreaded processors through dynamic loop detection//Proceedings of the 4th International Symposium on HPCA. Las Vegas, NE, 1998: 14-23

[16] Chappell R S et al. Simultaneous subordinate microthreading (SSMT)//Proceedings of the 26th ISCA. Atlanta, GA, 1999: 186-195

[17] Jesshope C R, Luo B. Micro-threading: A new approach to future RISC//Proceedings of the ACAC 2000. Canberra, 2000: 34-41

[18] Agarwal Anant, Lim Beng-Hong, Kranz David, Kubiatowicz John. APRIL: A processor architecture for multiprocessing//Proceedings of the 17th Annual International Symposium on Computer Architecture. Seattle, WA: IEEE Computer Society Press, 1990: 104-114

[19] Tullsen D M, Eggers S J, Levy H M. Simultaneous multithreading: Maximizing on-chip parallelism//Proceedings of the 22nd Annual International Symposium on Computer Architecture. Santa Margherita Ligure, Italy, 1995: 392-403

[20] Tullsen D M, Eggers S J, Emer J S, Levy H M, Lo J L, Stamm R L. Exploiting choice: Instruction fetch and issue on an implementable simultaneous multithreading processor//Proceedings of the 23nd Annual International Symposium on Computer Architecture. Philadelphia, PA, 1996: 191-202

[21] Eggers S J, Emer J S, Levy H M, Lo J L, Stamm R L, Tullsen D M. Simultaneous multithreading: A platform for next-generation processors. IEEE Micro, 1997, 17(5): 12-19

[22] Hammond Lance et al. The stanford hydra CMP. IEEE Micro, 2000, 20(2): 71-84

[23] Marr Deborah T, Binns Frank, Hill David L, Hinton Glenn, Koufaty David A, Miller J Alan, Upton Michael. Hyper-threading technology architecture and microarchitecture. Intel Technology Journal Q1, 2002, 6(1): 4-15

[24] Kalla Ron, Sinbaroy Balaram, Tendler Joel M. IBM Power5 chip: A dual-core multithreaded processor. IEEE Micro, 2004, 24(2): 40-47

[25] Kongetira Poonacha, Aingaran Kathirgamar, Olukotun Kunle. Niagara: A 32-way multithreaded sparc processor. IEEE Micro, 2005, 25(2): 21-29

[26] Dubios M, Scheurich C, Briggs F. Memory access buffering in multiprocessors//Proceedings of the 13th International Symposium on Computer Architecture, 1986

[27] Lamport L. How to make a multiprocessor computer that correctly executes multiprocessor programs. IEEE Transactions on Computers, 1979, C-28(9): 690-691

[28] Goodman J. Cache consistency and sequential consistency. SCI Committee: Technical Report No. 61, 1989

LI Zu-Song, born in 1977, Ph. D., associate researcher. His research interests include high performance computer architecture and verification.

XU Xian-Chao, born in 1979, M. S.. His research interests include computer architecture and operating system.

HU Wei-Wu, born in 1968, Ph. D., professor, Ph. D. supervisor. His main research interests include high performance computer architecture, parallel processing and VLSI design.

TANG Zhi-Min, born in 1966, Ph. D., professor, Ph. D. supervisor. His main research interests include high performance computer architecture and parallel processing.

Background

With Moore's law furnishing chip designers with billions of transistors, architects are increasingly moving towards simultaneous multithreading (SMT) and chip multiprocessors (CMP) as an effective way of dealing with escalating design complexity and power constraints and exploiting thread level parallelism to improve the performance using available on-chip transistor resources. General purpose processor Godson-2 is developed by the Institute of Computing Technology, Chinese Academy of Sciences. Superscalar Godson-2 processor can only exploits instruction-level parallelism and data-level parallelism. Developing from conventional instruction-level parallelism and data-level parallelism to thread-level parallelism by implementing multithread technique can improve performance of Godson-2 significantly. Nowadays, there is a clear thread in industry towards CMP+SMT processors, like the Intel Core 2 Duo, IBM POWER5, and POWER6, and SUN T1 and T2 Niagara processors. Godson-3 Processor implements CMP technique. And the work demonstrated in this paper of Godson-2 multithreading processor has been implemented to exploit thread-level parallelism fully by SMT technique. The work is supported by the National Natural Science Foundation of China (60325205, 60703017 and 60736012), the National High Technology Research and Development Program (863 Program) of China (2007AA01Z114), the National Basic Research Program (973 Program) of China (2005CB321600).