ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.4, April 2007, pp.1047−1055
DOI: 10.1360/jos181047

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于龙芯 CPU 的多核全系统模拟器 SimOS-Goodson*

高 翔[1,2+], 张福新[1], 汤 彦[1], 章隆兵[1], 胡伟武[1], 唐志敏[1]

[1](中国科学院 计算技术研究所,北京 100080)

[2](中国科学技术大学 计算机科学技术系,安徽 合肥 230027)

SimOS-Goodson: A Goodson-Processor Based Multi-Core Full-System Simulator

GAO Xiang[1,2+], ZHANG Fu-Xin[1], TANG Yan[1], ZHANG Long-Bing[1], HU Wei-Wu[1], TANG Zhi-Min[1]

[1](Institute of Computing Technology, The Chinese Academy of Sciences, Beijing 100080, China)

[2](Department of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China)

+ Corresponding author: Phn: +86-10-62565533 ext 9320, E-mail: gaoxiang@ict.ac.cn

Gao X, Zhang FX, Tang Y, Zhang LB, Hu WW, Tang ZM. SimOS-Goodson: A Goodson-processor based multi-core full-system simulator. *Journal of Software*, 2007,18(4):1047−1055. http://www.jos.org.cn/1000-9825/18/1047.htm

Abstract: As the Chip MultiProcessors (CMPs) have become the trend of high performance microprocessors, the target workloads become more and more diversified. The traditional user-level simulators cannot handle them, so new simulators are needed for the future architecture research. Based on the SimOS full-system environment, a new multi-core full-system simulator of Goodson processors, SimOS-Goodson, has been designed and implemented. The SimOS-Goodson decouples the simulation functionality and timing. It adopts a new value-prediction approach to implement memory consistency in the simulation environment. The credibility and accuracy of SimOS-Goodson are achieved by cross-validating the simulator with the actual hardware. The simulator inherits the benefits such as high speed and high flexibility from the traditional user-level simulators. It also has the new benefits such as accuracy, full-system support and easy to use. By porting the entire Linux OS, analysis and evaluation of the microarchitecture and workloads can be conducted easily in the SimOS-Goodson full-system environment. On a machine of Pentium4 3.0GHz, the speed of SimOS-Goodson exceeds 300K instructions per second. SimOS-Goodson will play a key role in the research of future Goodson multi-core architecture.

Key words: simulator; Goodson-2 processor; full-system; multi-core; SimOS

摘 要: 随着片上多核结构成为当前高性能微处理器发展的趋势,目标工作负载也变得多样化,传统的用户级

* Supported by the National Natural Foundation of China for Distinguished Young Scholars under Grant No.60325205 (国家杰出青年基金); the National High-Tech Research and Development Plan of China under Grant Nos.2005AA110010, 2005AA119020 (国家高技术研究发展计划(863)); the National Grand Fundamental Research 973 Program of China under Grant No.2005CB321600 (国家重点基础研究发展规划(973)); the Basic Research Foundation of the Institute of Computing Technology, the Chinese Academy of Sciences under Grant No.20056020 (中国科学院计算技术研究所基础研究基金); the Knowledge Innovation Program of the Institute of Computing Technology, the Chinese Academy of Sciences under Grant No.20056240 (中国科学院计算技术研究所知识创新课题)

Received 2005-12-19; Accepted 2006-04-03

模拟器已不能适应未来体系结构的研究需要.基于 SimOS 全系统模拟环境,设计并实现了龙芯 CPU 的片上多核全系统模拟器 SimOS-Goodson.在 SimOS-Goodson 的设计中运用了时序与功能分离的组织形式,并采用了一种新的值预测校验算法来解决模拟环境中的存储一致性问题.经过与真实硬件环境进行交叉校正,保证了模拟器的可信度与准确度.与用户级模拟器相比,SimOS-Goodson 保持了高速、灵活的优点,又具备精确、全系统和易使用的特征.通过对完整 Linux 操作系统的移植,可在 SimOS-Goodson 所模拟的全系统环境中进行各类微体系结构和应用负载的分析与评估.在 3.0GHz 的 Pentium4 微机上,SimOS-Goodson 的指令模拟速度超过 300K/秒.SimOS-Goodson 将会在基于龙芯 CPU 的片上多核体系结构研究中发挥重要作用.

关键词: 模拟器;龙芯 2 号处理器;全系统;多核;SimOS
中图法分类号: TP302 文献标识码: A

模拟器是微体系结构研究者进行性能分析和设计空间探索所采用的重要支持工具,通过软件化的方法对硬件进行建模,研究者可以用参数化可控的形式来考察目标应用在被模拟系统上执行时的行为与特征.模拟器的设计需要权衡建模详细度、配置修改灵活度、执行速度 3 方面的需要,从具体实现形式上看,可分为用户态模拟和全系统模拟两大类.以 SimpleScalar[1]为代表的用户态模拟器一般针对某类特定的目标应用而设计,只进行处理器用户态的模拟.相对于全系统模拟而言,用户态模拟器实现更简单、修改也更灵活,因此,曾被广泛应用于超标量处理器时代的微体系结构研究.然而,随着处理器结构的发展和新应用的出现,模拟器设计面临一系列新的要求:一方面,目标工作负载日益多样化,新兴的各类应用,如 Web 服务、电子商务等,需要频繁进入核心态,只模拟用户态代码无法获取准确的结论;另一方面,片上多处理器结构成为当前高性能处理器的趋势,Intel,AMD, IBM 等各大处理器厂商都推出了商业的双核处理器,CMP(chip multiprocessor)结构特有的多核同步通信、片内线程调度等技术在很大程度上依赖于操作系统的支持.传统的用户态模拟已难以满足对新结构和新应用的研究需要,采用全系统模拟将是今后微体系结构研究的必然趋势.

由于多核全系统模拟不可避免地增大了系统的复杂度和模块的耦合度,如何协调详细、灵活、快速 3 方面的设计要求一直都是困扰研究者的难题.从国际上的研究现状来看,已有的全系统模拟器要么为了保持模拟的详细而降低了修改的灵活性,要么为了灵活性而影响了模拟速度[2].与用户级模拟器相比,全系统模拟器大都存在修改困难、执行速度较慢等问题.另外,模拟器的精度也是一个重要但常常为模拟器开发者所忽视的问题[3].国际上全系统模拟器的设计大多针对一个抽象的硬件进行建模,不可能对模拟器本身存在的误差进行分析校正,因此,难以获得准确、可信的结论.

本文基于 Stanford 大学开发的著名的 SimOS[4]全系统模拟环境,设计开发了龙芯 CPU[5]的片上多核全系统模拟器 SimOS-Goodson.本文的贡献包括:① 将用户级模拟器中常用的时序与功能分离的组织形式应用到全系统多核模拟器中,并提出一种新的值预测校验算法,解决了由此带来的存储一致性问题.SimOS-Goodson 具备了类似于用户级模拟器的详细、灵活、快速的特点,平均指令模拟速度超过 300K/秒;② SimOS-Goodson 采用真实的龙芯 2 号 CPU 作为基本的微处理器核,通过与真实硬件环境进行交叉校正,保证了模拟器的可信度和准确度.

本文第 1 节介绍研究背景.第 2 节是 SimOS-Goodson 的设计与实现.第 3 节介绍 SimOS-Goodson 的用户接口.第 4 节对 SimOS-Goodson 进行评测.第 5 节将 SimOS-Goodson 与国际相关工作进行比较.第 6 节是总结与未来工作展望.

1 背景介绍

1.1 龙芯2号CPU性能模拟环境

龙芯 2 号是中国科学院计算技术研究所研制的高性能通用处理器,采用了 4 发射超标量超流水结构,实现了先进的转移猜测、寄存器重命名、动态调度等乱序执行技术,以及非阻塞的高速缓存和取数操作猜测执行等

动态存储访问机制.在龙芯 2 号的研制过程中,最早开发的模拟器是信号级的 ICT-Goodson 模拟器,它模拟了处理器的所有细节,可以非常准确地给出处理器的行为数据.但是,这同时也造成它的速度和灵活性受到较大的限制,对于性能分析的一些重要任务,例如设计空间探索、硅前性能预测等,使用 ICT-Goodson 都显得很困难.Sim-Goodson 是 ICT-Goodson 之后设计的龙芯 2 号执行驱动(execution-driven)模拟器,它基于 SimpleScalar 工具集开发,采用功能和时序模拟分离的组织结构,经过与真实龙芯 2 号 CPU 硬件的校正,具备修改简单、执行迅速的特点.从目标工作负载是以 SPEC CPU2000 为代表的单线程计算密集性应用的角度来看,Sim-Goodson 可以较好地满足研究者对于传统单核微处理器结构进行性能分析和评估的要求,但是,Sim-Goodson 只能对单线程静态连接的程序进行用户级模拟,没有核心态和全系统组件的支持.

1.2 SimOS简介

SimOS 是由 Stanford 大学 DASH/FLASH 研究小组开发的一个开放源码多处理器全系统模拟环境,被学术界和工业界所广泛采用.SimOS 全系统模拟环境采用了一个类 MIPS R4000 的顺序单发射处理器模型,对包括硬盘、网卡、终端控制台等在内的计算机系统各个部件进行了精确模拟,可以运行完整的商用操作系统.SimOS 的主要缺点是:它采用的单发射处理器模型已经无法反映当前高性能微处理器的典型特征;另外,它运行的是一个经过修改的商业操作系统 IRIX,一般用户无法获得该操作系统源代码,不能进行涉及操作系统的改进(比如增加新设备).

2 SimOS-Goodson 的设计与实现

SimOS-Goodson 的设计目标包括修改配置灵活,执行迅速、准确.为了在全系统模拟环境中达到这些设计目标,SimOS-Goodson 采用了在用户级模拟器中常用的时序与功能分离的执行驱动方式.同时,为了减少非必要的工作量,我们借用 SimOS 模拟器的全系统组件支持,采用了将龙芯 2 号处理器核模型融入 SimOS 全系统环境中的设计思路.

SimOS-Goodson 的结构如图 1 所示,底层的硬件结构部分模拟了计算机系统中的所有组件,包括处理器核、存储子系统、硬盘、网卡、终端控制等,在此基础上运行完整的 Linux/MIPS 操作系统,使用者可以在一个真实的操作系统环境中进行各类应用程序的执行模拟.

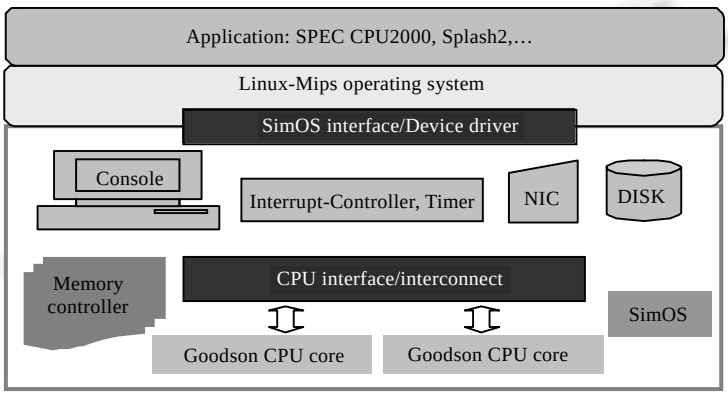

Fig.1 Diagram of SimOS-Goodson

图 1 SimOS-Goodson 结构图

在 SimOS 环境中实现对龙芯 2 号多核全系统模拟需要解决 3 个方面的关键问题:龙芯 2 号处理器核模型的实现、存储一致性问题的解决以及全系统环境支持.

2.1 处理器核模型

SimOS-Goodson 中的处理器模型直接在已有的龙芯 2 号用户级模拟器 Sim-Goodson 基础上进行实现.由于

用户级模拟器只实现了用户态的指令,采用称为“代理(proxy)”的方法来处理应用程序中的系统调用,例外处理也很简单,为此,需要在模拟器的功能模拟部分加入系统核心态指令的支持;在时序模拟部分完善异常处理机制.

由于采用了与用户级模拟器相似的功能和时序分离的组织形式,在全系统模拟环境中实现精确异常是SimOS-Goodson 设计中面临的一个难题.龙芯 2 号是一个多发射乱序执行的超标量处理器,在任一时刻,可能会有很多条指令处于流水线的不同级上,当某条指令发生异常时,实际硬件是在异常的指令位于重定序队列(reorder queue,简称 ROQ)头时才进行流水线排空、现场恢复等处理.而功能与时序分离的模拟器使用独立的执行驱动引擎,功能模型中指令的解释执行在译码阶段就完成,译码之后的流水级不再传递数据流,时序模型部分只维护指令时序所需要的微体系结构状态,不关心实际的数据.与硬件行为相比,模拟器对机器状态的修改发生在译码级,此时,模拟器尚不能检测出译码之后可能出现的异常,待到提交阶段检测出异常时,需要被取消的指令又已经修改了机器状态.为了正确实现精确异常,同时准确模拟异常指令后的指令执行对分支预测器和 Cache 等微结构状态的影响,我们设计实现了一种“备份滚回”的机制:在译码级修改机器状态时,记录下该指令所影响的机器状态单元的旧值,包括旧的寄存器状态和旧的内存数据;在译码级之后出现异常或者中断需要恢复现场时,对重排序队列采用从尾到头的逆序遍历,将指令所修改的单元滚回成该指令执行前的值,从而可以精确地恢复到异常之前的机器状态.

2.2 存储一致性问题

龙芯 2 号采用的是处理器一致性模型(processor consistency),允许写后读乱序,写指令在全局可见之前就会提交,但是,真正的机器状态修改是在写指令提交后若干拍才完成的.时序和功能分离的模拟器由于在译码阶段就模拟完成了实际的访存操作,从正确实现存储一致性的时序角度看,这种方式实际上表现为所有的访存操作都乱序,违反了一致性模型对访存次序的要求.为了解决由此带来的存储一致性问题,我们设计了一种“值预测检验”算法,描述如下:

① 译码阶段,访存指令依旧模拟内存数据的读写.

- 对于写指令,在已有的多重误预测[1]基础上添加了一层用于暂存非推测写的数据.对这些非推测写数据采用了译码后局部可见和提交后全局可见的策略,即可被本处理器后续的取数指令访问,但对其余处理器暂时不可见.在写指令正常提交后,非推测写数据才会被用于机器状态的修改,此后的值为所有处理器全局可见.
- 对于读指令,需要先查找非推测写暂存数据,没有找到才访问实际的体系结构状态.

② 提交阶段,对访存指令进行机器状态校验和修改.

- 对于读指令,重新访问实际的体系结构状态进行该内存单元的数据校验,若出现值不一致的情况,则根据所采用的存储一致性模型决定是否需要进行错误恢复.恢复过程与精确异常的恢复过程类似.
- 对于写指令,在时序模型中修改机器状态时功能模型才真正进行机器状态的修改,同时删除原有的暂存非推测写数据.由于龙芯 2 号所具有的结构特性,此时的修改可能发生在写指令已经提交后的若干拍后.

从时序角度看,采用值预测检验算法后,读指令可认为是在译码阶段采用了值预测的机制.文献[6]中曾对值预测的存储一致性问题进行过研究,指出乱序访存造成对一致性模型的违反可通过在提交时进行值校验来解决.由于龙芯 2 号处理器放松了对写后读同一单元的次序要求,写指令的值可以先本地可见,读指令可获取前面尚未提交的写操作结果.在同一单元的写指令译码后和读指令提交前的这段时间内,可能会有其余处理器修改了该单元,对于此类读指令在提交校验时出现的数据不一致的情况,是龙芯 2 号所采用的一致性模型可以接受的正确结果,因此不需要进行错误恢复.

2.3 全系统环境支持

全系统模拟环境需要对包括硬盘、网卡、终端控制台等在内的计算机系统各个部件进行精确模拟,以满足

正常操作系统运行的要求.SimOS-Goodson 借用了 SimOS 中已有的框架结构和系统组件建模,在此基础上实现对全系统模拟的支持.全系统模拟器可以视为一个虚拟机,要在其上运行新的操作系统有两种方法:在模拟器端中增加操作系统所需要的硬件;或者在操作系统里增加现有虚拟硬件的驱动支持.我们综合使用这两种方法:

- 在模拟器端增加了标准 8255 串口硬件支持,这是 Linux/MIPS 内核支持的控制台设备,使用者可以通过模拟控制台来实现与被模拟系统的交互与控制.另外,SimOS-Goodson 运行时需要装载操作系统内核,Linux/MIPS 内核是一个 ELF(executable and linkable format)格式的文件,为此,在 SimOS-Goodson 中实现了 ELF 文件装载的支持.
- 操作系统级的支持主要包含两个方面:一方面是针对虚拟多核平台特点,对已有龙芯 2 号的 Linux 操作系统进行多处理器版本的移植,主要工作包括多处理器启动时的初始化、处理器间中断驱动的支持以及系统全局和本地时钟中断的处理;另一方面,在 Linux 内核中加入了对 SimOS-Goodson 虚拟机的识别和虚拟 SCSI(small computer system interface)磁盘、虚拟网卡等设备的驱动支持.

目前,我们在 SimOS-Goodson 上完成了多处理器版 Linux 操作系统(内核版本 2.4.20)的移植.由于 SimOS-Goodson 实现了龙芯处理器全部的硬件指令和必要的基本外设的模拟,龙芯处理器支持的 VxWorks 等嵌入式操作系统也可以在经过简单的修改后移植到 SimOS-Goodson 模拟器上.SimOS-Goodson 提供给用户的是一个完整、真实的多处理器操作系统环境,实际环境中的各类应用都可以直接运行在模拟器上.

3 用户接口

3.1 调试支持

模拟器的使用者经常需要对模拟器结构进行源代码级的修改,调试过程将会占据相当大的工作量,必须在设计之初就考虑到提供方便的程序调试接口.SimOS-Goodson 为使用者提供了 3 种调试手段,包括踪迹交叉比对、GDB(the GNU project debugger)调试插桩(stub)和检查点(checkpoint)支持.

踪迹交叉比对可以通过在线与离线两种方式进行.在线方式通过利用进程间通信,直接比对 SimOS-Goodson 与 SimOS-Fast 的指令踪迹来进行,其中,SimOS-Fast 是我们最早在 SimOS 上实现的一个纯功能级模拟器,指令模拟速度可达 2M 指令/秒以上.经过大量严格的测试,我们将其作为功能级的基准模拟器纳入到调试工具集中.在线踪迹交叉比过程如图 2 所示,两个模拟器同时运行,SimOS-Fast 将指令踪迹和指令输出状态写入到共享的数据通信缓冲区中,SimOS-Goodson 读取并比对缓冲区中的指令序列来判断程序执行是否正确;离线方式直接比对静态的指令流,先由 SimOS-Fast 将典型程序的指令序列与状态写入到踪迹文件中,在 SimOS-Goodson 执行时,通过与踪迹文件中的指令流比较来判断执行正确与否.

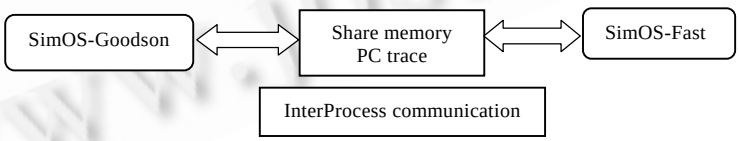

Fig.3 On-Line cross validating

图 3 在线交叉验证

SimOS 支持利用 GDB 调试插桩交叉调试被模拟系统,可以对被模拟系统进行源代码级调试.这一功能对于不适合采用踪迹比对的并行程序调试非常有用.但是,可获得的 SimOS 软件包中的 GDB 版本很低(4.16),也不能支持现代 Linux/MIPS 系统的程序.我们在新版本的 GDB(5.3 和 6.0)上重新实现了 SimOS-Goodson 的调试插桩,并针对龙芯 2 号体系结构和多核环境特性进行了修改扩充,使之能够用于交叉调试 SimOS-Goodson 所模拟的多核 Linux/MIPS 系统.

检查点支持是在 SimOS-Goodson 中集成的另一种有效的调试手段.由于多核全系统模拟的复杂性,错误往往发生在程序执行很长时间以后.通过加入检查点的支持,可以在很短时间内恢复到错误现场,从而提高调试工

作的效率.检查点的另一作用是可以减少模拟器的预热(warm-up)和启动时间.例如,在程序进入核心循环或其他使用者感兴趣的区域之前建立检查点,以后每次模拟就可以直接从检查点开始,从而省掉系统启动耗时.

3.2 数据统计与设计指导

SimOS-Goodson 的一大特色是可以用 Tcl 语言脚本来控制模拟器的运行并进行性能数据的统计.由于脚本的解释独立于系统模拟,脚本的添加与修改无须重新编译模拟器源代码,也不会对模拟器的正常运行造成影响.原始的 SimOS 版本不能解析 ELF 文件的调试符号,我们对这部分代码进行了重新实现,解决了 dwarf 格式调试符号的识别问题,这样,在 Tcl 脚本中可以直接使用符号名称来引用函数和变量,方便了脚本编写.

在多处理器全系统环境中,由于进程会在各个处理器核上迁移,若采用不公开源码的商业操作系统,模拟器将很难精确统计每个进程自己的性能数据.SimOS 中原有的性能分析脚本是针对 IRIX 操作系统编写的,针对 Linux 开放源码的特性,我们对其进行了重新实现.通过对 Linux 内核进程切换过程的分析,我们在 Tcl 脚本中实现了对进程切换行为的识别与跟踪功能,可以按进程号来精确统计被模拟系统中每个进程的性能数据.

在全系统模拟器的帮助下,通过对统计得到的性能数据进行分析,可以更加精细地理解和描述实际硬件的行为,从而提早暴露和精确定位设计中的问题.例如,在龙芯 2 号的设计中,采用一个 Load/Store 队列(称为 cp0queue)来解决存储相关[5],cp0queue 是一个有序队列,它能够检测前后的 Load/Store 指令是否发生相关.早期设计中不支持猜测写回,即只要有更早的地址未知的 Store,则无论 CACHE 是否命中,Load 操作都不能立刻写回,而是要等待它之前所有的 Store 被发射到 cp0queue.虽然硬件队列可以同时容纳多个 LW/SW 操作,但所有的 LW/SW 必须串行执行.在模拟器上统计到的实验数据表明,Load 由于等待前面的 Store 地址解决而被延迟写回的情况相当普遍(平均约 30%),这样的设计加大了访存的延迟.解决的方法是,让准备好的 Load 无论前面有没有地址未知的 Store 都写回,如果后来检测到发生相关,则利用例外机制取消这个 Load.根据修改后的运行结果,所有 SPEC CPU 程序都从中受益,定点程序性能平均提高达 11%,浮点程序性能平均提高 7%.

4 评 测

SimOS-Goodson 可运行在任意体系结构的 Linux 或类 Unix 平台上,本文的硬件测试平台配置为 3.0G Pentium 4,内存 2GB,硬盘 4GB,软件环境为 Red Hat Linux 7.3,采用的编译器为 GCC 2.96.被模拟系统上运行 Linux/MIPS,内核版本 2.4.20.评测分为考察误差的准确度评测与考察执行速度的性能评测两部分.

4.1 准确度测试

由于多核系列的龙芯处理器尚在前期研究中,暂时只有龙芯单处理器硬件环境可供 SimOS-Goodson 进行交叉验证,因此,准确度测试采用了单进程的 SPEC CPU 2000[7]与 LMbench[8]两类测试程序,多线程并行程序的误差验证将在未来多核龙芯处理器流片后进行.典型的 SPEC CPU 2000 程序用于检测 CPU 模型由用户级扩充为全系统时可能导入的错误,评测指标为程序平均每拍执行的指令 IPC(instructons per cycle).SimOS-Goodson 的 CPU 模型实现了全系统模拟和用户级模拟的同时支持.作为系统中最复杂的部件,处理器模型已经在用户态环境下进行了严格的验证,但由于系统环境发生了变化,一定程度上的偏差也将必然存在.

表 1 列出了在龙芯 2 号单 CPU 的典型配置下,SPEC CPU2000 程序使用 test 输入集,每个程序最多运行 10 亿条指令时,龙芯 2 号 3 个模拟器 IPC 的比较.对表 1 中的数据进行分析可以看出,与最准确的 ICT-Goodson 模拟器相比,大部分程序 SimOS-Goodson 的误差都小于 10%,说明 SimOS-Goodson 具有较为可信的精度;与用户级模拟器 Sim-Goodson 相比,SimOS-Goodson 的 IPC 数据大多略微偏低,可能的误差主要来源于全系统环境中内核线程、进程切换等操作系统服务等对分支预测器的干扰和对 Cache 的污染.

LMbench 是一组操作系统核心态测试程序包,主要测试操作系统服务性能,评测指标为程序执行耗时(微秒).表 2 是 300MHZ 主频时,真实的龙芯 2 号机器与 SimOS-Goodson 模拟器的比较.大部分评测项的误差都在 15%以内,说明 SimOS-Goodson 对操作系统核心态的模拟比较接近真实的硬件环境.

Table 1 Validating with SPEC CPU 2000

表 1 SPEC CPU 2000 程序验证结果

Prog (SPECint)	ICT-Goodson	Sim-Goodson	SimOS-Goodson	IPC-Err.
Mcf	0.41	0.43	0.41	0.00
Parser	0.75	0.76	0.72	–0.04
Perlbmk	0.57	0.55	0.52	–0.09
Crafty	1.05	1.06	0.96	–0.09
Twolf	0.84	0.89	0.81	–0.04
Vortex	0.66	0.64	0.60	–0.09
Vpr	0.97	0.97	0.91	–0.06
Gap	0.83	0.84	0.82	–0.01
Gcc	0.71	0.71	0.67	–0.06
Gzip	0.81	0.82	0.77	–0.05
Eon	0.97	0.98	0.95	–0.02
Applu	0.61	0.62	0.65	0.07
Apsi	0.51	0.52	0.50	–0.02
Art	0.15	0.15	0.16	0.07
Swim	0.45	0.46	0.46	0.02
Equake	0.82	0.83	0.89	0.09
Mesa	1.09	1.12	1.06	–0.03
Wupwise	1.17	1.14	1.09	–0.07
Sixtrack	1.04	1.03	0.92	–0.12

Table 2 Validating with LMbench

表 2 LMbench 程序验证结果

Application	Real	SimOS-Goodson	Err.	Description
Syscall null	0.81	0.81	0.00	Simple syscall
Syscall read	1.43	1.72	0.20	Read syscall
Pipe	13.18	13.44	0.02	Interprocess communication through pipes
Sig catch	5.23	5.59	0.07	Signal handler
Unix	22.87	23.53	0.03	Unix stream socket
Fifo	13.2	13.93	0.06	Pipe operation
Ctx	3.15	3.63	0.15	Context switching

4.2 性能评测

性能评测采用了两种模拟器的速度指标,包括指令模拟速度 IPS(instructions per second)与主频模拟器速度 CPS(cycles per second).为了能够更准确地反映出模拟器的性能,我们对单核、双核和 4 核的 3 种配置分别进行评测,采用的测试程序包括 Linux 操作系统启动、SPEC CPU 2000、并行科学计算程序 SPLASH2[9]以及商业应用 Apache Benchmark 等,其中,SPLASH2 采用的进程数等同于核的数目.表 3 列出了 SimOS-Goodson 对不同程序的模拟速度.

Table 3 Simlating speed of SimOS-Goodson

表 3 SimOS-Goodson 的模拟速度

Core count	Workload	Cycles (K)	Instructions (K)	Time (s)	CPS (K)	IPS (K)
1	Linux OS boot	21 235	13 111	42	506	312
1	Spec2k (MESA)	500 000	439 558	1 435	348	306
1	Splash2 (radix)	196 054	146 663	420	466	349
1	Apache benchmark	75 711	50 646	167	453	303
2	Linux OS boot	21 572	25 196	78	277	323
2	Splash2 (radix)	91 838	132 241	428	215	309
2	Spec2k (MESA+MESA)	500 000	794 916	2 620	190	303
2	Apache benchmark	38 798	76 544	238	163	322
4	Linux OS boot	24 217	66 122	163	148	405
4	Splash2 (radix)	64 440	198 351	623	104	318
4	Apache benchmark	39 237	180 063	510	77	353

对表 3 中数据加以分析可以看到,模拟速度受到应用程序和所模拟的体系结构双重影响.主频模拟速度主要与所模拟的结构相关,处理器核的数目每增加 1 倍,主频模拟速度约下降一半.最快的主频模拟速度出现在单

核 Linux 内核启动时,速度可达 506K CPS.而指令模拟速度对应用程序的 IPC 比较敏感,一般而言,程序的 IPC 越高,指令模拟速度越快.对于各类程序,SimOS-Goodson 指令模拟速度都在 300K IPS 以上,最快的指令模拟速度 405K IPS 出现在 4 核的 Linux 启动过程中,由于这期间各处理器核需要互相进行频繁的同步,每个核都有相当长的时间处于空闲等待状态而循环执行 Linux 内核的 cpu_idle 函数,此时的 Cache 命中率和分支预测率都接近于 1,程序有很高的 IPC 值,从而加快了指令的模拟.

与龙芯 2 号 CPU 的其余模拟器相比,ICT-Goodson 的速度约为 50K 指令/秒;Sim-Goodson 的速度约为 500K 指令/秒;SimOS-Goodson 速度平均在 300K 指令/秒以上,最高可超过 400K 指令/秒.ICT-Goodson 与硬件行为完全一致,模拟最多的细节,具有最高准确性,因此模拟速度较慢;Sim-Goodson 抽象层次较高,模拟细节较小,准确性较高,速度较快;SimOS-Goodson 是全系统模拟,抽象层次、模拟细节和速度都非常接近用户级模拟器 Sim-Goodson,但由于加入对更多组件的模拟支持,在一定程度上影响了模拟速度.

5 相关工作比较

DEC 和 IBM 两大处理器厂商曾分别在 SimOS 的基础上开发出基于 Alpha 和 PowerPC 处理器的全系统模拟平台 SimOS-Alpha 和 SimOS-PPC.与 SimOS-Goodson 一样,这些模拟器都借助了 SimOS 环境来降低开发难度.但是,由于采用了时序和功能模拟紧耦合的实现方式,造成修改灵活性和模拟速度的降低;而且,这两种模拟器都是针对商业操作系统而设计的,在原始设计者停止软件维护后,后续的操作系统版本在模拟器上将面临严重的兼容性问题.此外,其采用的商业操作系统不公开源代码,因而增大了普通使用者的调试与统计难度.

GEMS(general execution model simulator)[10]是 Wisconsin 大学开发的全系统模拟器,它借助商业模拟器 Simics[11]作为功能模拟基础,通过加载时序模型来驱动功能模拟器 Simics 的执行.与 SimOS-Goodson 相比,这是另外一种功能和模拟分离的实现形式.GEMS 的主要优点在于它的灵活和简洁,用户可以专注于时序模型的设计,而全系统环境下复杂的功能模拟则交由商业模拟器 Simics 来实现.但正是因为功能模型受时序驱动,每条指令提交时都需要在时序模型和功能模型之间校验结果,使得时序模拟和功能模拟的部分工作重叠冗余而影响模拟执行的速度.根据文献[2]中公布的数据,这种实现方式的指令模拟速度仅达到 120K IPS.另外,GEMS 的功能模型采用的是一个扁平(flat)的内存映象,只能模拟严格的顺序一致性,而无法满足弱一致性模型的模拟要求.

由于所基于的体系结构和硬件模型的差异,无法将上述模拟器与 SimOS-Goodson 进行直接的性能比较,但从速度量级的角度来看,SimOS-Goodson 具有超过 300 K IPS 的执行速度,已经达到了目前国际上全系统模拟的最高性能.另外,从模拟器的准确度来看,SimOS-Goodson 通过和实际的龙芯处理器系统进行交叉校正,可以准确给出反映模拟器精度的误差分析,这种硅后验证的方式一般只能应用在工业界模拟器上,而学术界开发的模拟器只是对抽象机器进行建模,不具备硅后验证的条件,无法对模拟器的精确度作出分析与说明.

6 结论和未来工作

本文在全系统模拟环境 SimOS 的基础上设计开发了多核全系统模拟器 SimOS-Goodson.通过采用所设计的备份滚回、值预测检验等算法,解决了由功能与时序分离的组织形式所带来的精确异常、存储一致性等关键难题,较好地协调了速度、灵活、精确 3 方面的关系.SPEC CPU 2000 和 LMbench 的测试结果表明,SimOS-Goodson 与真实硬件环境的误差大都在 15%以内,具有相当高的可信度.SimOS-Goodson 中还实现了方便、高效的调试支持,同时,针对龙芯 2 号 CPU 和 Linux 操作系统的特点设计了灵活的控制与统计功能.尽管因为模拟了更多的组件和实现了更多的功能,SimOS-Goodson 的速度比用户级模拟器略慢约 30%,但其 300K 指令/秒的全系统模拟速度仍然具有较高的可用性.

SimOS-Goodson 将作为未来多核龙芯处理器的一个研究平台进行各种微结构的设计与评估,以指导多核结构处理器的研究.由于当前多核系列的龙芯处理器版本尚在前期研究中,暂时还无法对 SimOS-Goodson 模拟器的 Cache 一致性协议、互连结构等部分进行与实际硬件的误差校正工作,我们计划在多核系列的龙芯处理器流片后进行模拟器的硅后验证,以进一步提高精确度.在代码文档完善和条件成熟时,我们将在互联网上公布

SimOS-Goodson 模拟器,供国内外研究者使用与参考.

References:

[1] Burger DC, Austin TM. The simplescalar tool set, version 2.0. Technical Report, CS-TR-97-1342, Madison: University of Wisconsin, 1997.

[2] Mauer CJ, Hill MD, Wood DA. Full system timing-first simulation. In: Proc. of the 2002 ACM Sigmetrics Conf. on Measurement and Modeling of Computer Systems. ACM Press, 2002. 108−116.

[3] Gibson J, Kunz R, Ofelt D, Horowitz M, Hennessy J, Heinrich M. FLASH vs. (simulated) FLASH: Closing the simulation loop. In: Proc. of the 9th Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS). IEEE Computer Society, 2000. 49−58.

[4] Rosenblum M, Herrod SA, Witchel E, Gupta A. Complete computer system simulation: The SimOS approach. IEEE Parallel and Distributed Technology: Systems and Applications, 1995,3(4):34−43.

[5] Hu WW, Zhang FX, Li ZS. Microarchitecture of the Goodson-2 processor. Journal of Computer Science and Technology, 2005, 20(2):243−249.

[6] Martin MMK, Sorin DJ, Cain HW, Hill MD, Lipasti MH. Correctly implementing value prediction in microprocessors that support multithreading or multiprocessing. In: Proc. of the 34th Int'l Symp. on Microarchitecture. IEEE Computer Society, 2001. 328−337.

[7] Henning J. SPEC CPU2000: Measuring CPU performance in the new millennium. IEEE Computer, 2000,33(7):28−35.

[8] McVoy L, Staelin C. LMbench: Portable tools for performance analysis. In: USENIX Annual Technical Conf. San Diego, 1996. 279−294.

[9] Woo SC, Ohara M, Torrie E, Singh JP, Gupta A. The SPLASH-2 programs: Characterization and methodological considerations. In: Proc. of the 22nd Annual Int'l Symp. on Computer Architecture. IEEE Computer Society, 1995. 24−36.

[10] Martin MM, Sorin DJ, Beckmann BM, Marty MR, Xu M, Alameldeen AR, Moore KE, Hill MD, Wood DA. Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset, submitted to computer architecture news (CAN). 2005. http://www.cs.wisc.edu/gems

[11] Magnusson PS, Cbristensson M, Eskilson J, Forsgren D, Hällberg G, Högberg J, Larsson F, Moestedt A, Werner B. Simics: A full system simulation platform. IEEE Computer, 2002,35(2):50−58.

高翔(1982－),男,湖北公安人,博士生,主要研究领域为微处理器系统结构设计,性能评估,操作系统.

章隆兵(1974－),男,博士,副研究员,主要研究领域为微处理器设计,机群计算.

张福新(1976－),男,博士,副研究员,主要研究领域为微处理器系统结构设计,性能评估,机群计算.

胡伟武(1968－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为高性能计算机体系结构,并行处理,VLSI 设计.

汤彦(1981－),男,硕士生,主要研究领域为性能评估,操作系统,微处理器结构设计.

唐志敏(1966－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为高性能计算机体系结构,并行计算,芯片设计.