ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.19, No.4, April 2008, pp.1051−1068
DOI: 10.3724/SP.J.1001.2008.01051

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

计算机体系结构软件模拟技术*

喻之斌[1+], 金 海[1], 邹南海[2]

[1](华中科技大学 计算机学院 服务计算技术与系统教育部重点实验室,湖北 武汉 430074)

[2](英特尔亚太研发有限公司,上海 200241)

Computer Architecture Software-Based Simulation

YU Zhi-Bin[1+], JIN Hai[1], ZOU Nan-Hai[2]

[1](Service Computing Technologies and Systems Laboratory, Computer School, Huazhong University of Science and Technology, Wuhan 430074, China)

[2](Intel Asia-Pacific Research & Development Ltd, Shanghai 200241, China)

+ Corresponding author: Phn: +86-27-87543529 ext 8015, E-mail: yuzhibin@mail.hust.edu.cn

Yu ZB, Jin H, Zou NH. Computer architecture software-based simulation. *Journal of Software*, 2008,19(4): 1051−1068. http://www.jos.org.cn/1000-9825/19/1051.htm

Abstract: Computer architecture simulation is an integral part of modern computer design process. Simulation can help architects reduce the time and cost of computer architecture design dramatically. However, difficulties of constructing simulators, long simulation time, and poor accuracy limit the effectiveness of simulation. Many researchers have proposed a wide range of approaches to resolving the three main problems, which are still open. Meanwhile, new challenges start to emerge for future computer architecture simulation. This paper investigates the history of computer architecture simulation, classifies and compares the existing methodologies and technologies, and analyzes the challenges to help architects select, develop, or research on simulators. A brief description of the simulator SimIPF design is provided as well.

Key words: computer architecture; simulation; modeling of computer architecture; modeling techniques; performance evaluation

摘 要: 在现代处理器或计算机系统设计中,体系结构软件模拟技术已成为一个不可缺少的环节.与不使用模拟技术的计算机系统或处理器设计方法相比,软件模拟技术可以极大地降低设计成本和缩短设计周期.然而,由于开发计算机体系结构软件模拟器通常十分困难,模拟器运行标准性能测试程序的时间很长以及模拟结果精度差等 3 个主要问题,限制了体系结构软件模拟技术在计算机系统设计中的有效性.许多研究人员已经提出了各种各样的方法和技术来解决这些问题,但是,到目前为止,这些问题还并未得到根本性解决.同时,未来体系结构模拟技术的新挑战已经开始显现.研究了体系结构软件模拟技术的由来和历史,对现有的技术和方法进行了分类和比较,对未来的挑战也进行了分析,指出了该领域今后的发展方向,以帮助计算机体系结构设计师或研究人员选择、开发体系结构模拟器或对该技术进行研究.基于这些调查分析,正在使用较为先进的技术开发一个适合于安腾系列架构的体系结构模拟器 SimIPF.

* Supported by the National Basic Research Program of China under Grant No.2007CB310900 (国家重点基础研究发展计划(973))
Received 2007-08-10; Accepted 2007-09-30

关键词: 计算机体系结构;模拟;计算机体系结构建模;建模技术;性能评估
中图法分类号: TP302 文献标识码: A

随着人们对计算能力需求的不断增长,计算机系统已经变得越来越复杂.处理器复杂程度按摩尔定律增长,到目前为止,单片处理器的晶体管数目已经达到了 10 亿.制造这样一个复杂的计算系统所需的时间成本和资金成本是非常高昂的.以处理器的设计制造为例,通常生产一款新的处理器需要经过体系结构设计、设计评估与验证、逻辑设计与验证、电路设计、电路验证、布局设计和生产制造等阶段,而且这些阶段都要重复多次以保证质量.从时间上看,该过程通常会持续4年~7年甚至更长的时间.从资金成本上看,该过程需要的投入也是非常巨大的.在一款处理器批量生产之前,最好的质量保证方式是实际制造出数个该款处理器,对其测试评估;发现缺陷后,进行修改,再制造出新的处理器,然后再测试评估;重复该过程直到达到设计要求为止.然而,这种方式并不现实,因为哪怕只生产一个新处理器,其所需的周期非常长,成本也非常高.为了克服这个缺陷,研究人员开发了精确到时钟周期级别的体系结构软件模拟技术.

体系结构软件模拟技术使用软件的方式来模拟计算机系统硬件在系统结构级别的功能和性能特性.在现代计算机系统的设计和制造中,使用体系结构软件模拟技术来对体系结构设计进行评估和验证已成为一个不可缺少的环节.在典型的处理器设计过程中,有超过 60%的人力、物力资源都用在了评估和验证上[1].体系结构软件模拟技术可以极大地降低计算机系统开发的成本并缩短开发周期:一方面,开发一套软件与开发一套相同复杂程度的硬件相比,其开发周期要短得多,成本也要低得多;另一方面,软件的修改比相同复杂程度硬件的修改也容易得多,即软件具有更大的灵活性.因此,使用体系结构软件模拟技术可以在相对短的时间内评估验证多种可能的体系结构设计方案,从中寻找最好的方案.然而,3 个方面的主要问题限制了体系结构软件模拟技术在计算机系统设计中的充分使用:1) 体系结构软件模拟器开发难度大;2) 当评估新的设计时,模拟器运行时间长;3) 模拟器模拟结果的精度较低.

计算机系统本身是一个非常复杂的系统,要使用软件来模拟每个晶体管或每个门电路各个方面的行为特征几乎是不可能的.人们简化系统复杂程度的常用办法是对系统按层次进行抽象,体系结构就是对计算机系统在结构层次上的简化.然而,体系结构层次上的计算机系统依然很复杂,开发其软件模拟器也因此而十分困难.另外,目前开发软件模拟器使用的主要编程语言是 C 或 C++语言,这些语言是串行结构化语言.使用这种语言的固有机制,如函数或类来模拟计算机系统部件的功能和行为,是一个非常耗时且容易出错的过程.因此,从零开始开发一个系统结构软件模拟器并不容易.通常的做法是,在已存在的模拟器基础上进行二次开发或改进,使其适应自己的要求.实践证明,这种方法也存在较大困难,且模拟器的模拟结果往往令人怀疑.为了使模拟器对体系结构设计的评估是可信的,对模拟器本身还要进行大量的、反复的验证,这又从另一个方面增加了模拟器开发的难度和开发周期.

体系结构软件模拟器第 2 个方面的问题是模拟运行时间很长.模拟器是一个在宿主机上运行的程序,在进行详细模拟时,该程序需在时钟周期级别上记录每条动态指令的运行结果以及处理器的状态.如寄存器状态、内存和Cache行为、分支预测器的状态等,其数据量巨大,从而导致详细模拟运行的速度缓慢.目前,加速最成功的软件模拟器SMARTS[2]的运行速度是 9MIPS,与其运行的宿主机硬件相比,其运行速度慢了 4~5 个数量级.为了评估通用处理器的性能,国际标准化组织SPEC随着处理器性能的提高不断地发布新的标准性能测试程序包.目前,最新的处理器性能测试程序包为SPEC CPU2006.标准性能测试程序包中包含多个负载极大的测试程序,从多个方面来评测处理器的性能.为使模拟的结果可信,使用标准性能测试程序在模拟器上运行是较为自然的方法.如此慢的模拟速度却要运行对硬件来说也是极大负载的性能测试程序,其运行时间必定会非常长.在使用参考输入参数情况下,根据模拟目的的不同,其模拟运行时间需要几个月、几年甚至几十年.另一方面,在体系结构层次可配置的参数较多,如Cache的大小、TLB(translation look aside buffer)的大小等,而且各参数对系统的影响并不独立.每修改一个配置参数,都需要重新模拟运行测试程序,观察该参数对整个系统的影响.所以,要获得较好的体系结构,需要模拟运行的总时间会很长.

体系结构软件模拟器第 3 个方面的问题是其模拟运行结果的精度较差.引起精度较差的原因是模拟器开发过程中存在的许多错误和使用模拟器时的不合理策略.模拟器的开发可以分为 3 个步骤:1) 理解拟设计的体系结构;2) 设计针对该体系结构的模拟器;3) 实现该模拟器.这 3 个步骤都有可能出现错误.第 1 个步骤实际上是软件模拟器的需求分析,不能正确理解需求分析是软件开发过程中的一个常见现象;第 2 个步骤对应着软件开发中的设计与分析阶段,在软件模拟器设计过程中,可能出现的错误有 3 类:① 虽然正确地理解了体系结构设计但在模拟器设计时出现了某些错误;② 由对体系结构设计的错误理解导致的模拟设计错误;③ 模拟器设计时故意忽略了体系结构设计的某些细节而产生了错误.实现模拟器对应着软件开发中的编码阶段,在编码过程中出现错误是显而易见的现象.引起模拟器模拟结果精度较差的第 2 个原因是模拟器运行时间长引起的.由于在软件模拟器上运行处理器标准性能测试程序包中的所有程序耗时非常长,研究人员一般只执行程序的部分指令以代表所有指令的运行结果.如何选取标准性能测试程序的部分指令模拟运行,目前是国际上的研究热点.对模拟运行的部分指令的选取不当,是造成模拟结果精度差的主要原因之一.

体系结构软件模拟器在计算机系统设计中占据着非常重要的地位.模拟器的设计和开发能力直接影响着处理器或计算机系统的设计能力.本文回顾了计算机体系结构软件模拟器的发展历史;比较了已存在的模拟器开发技术、方法和模拟器;分析了模拟器开发中所面临的问题及挑战,指出了该领域将来的研究方向;并简短描述了我们正在开发的模拟器 SimIPF.

1 简要历史回顾

在早期的处理器设计中并没有使用体系结构软件模拟技术,它随着处理器变得越来越复杂而逐渐发展起来.20 个世纪 80 年代前后,处理器体系结构设计中主要使用“数据驱动”设计技术和性能分析模型技术.例如,20 世纪 80 年代初,高性能计算机的设计中主要使用的是“数据驱动”技术[3].该技术的关键是大量收集程序在已有机器上运行的相关数据,如执行的指令数目、某条指令的执行次数或某一类指令执行的次数等.在此基础上分析某种体系结构处理器的性能特征和缺陷,以便在以后的设计中避免这些缺陷.“数据驱动”的设计技术又被称为“基于指令执行轮廓的静态建模”技术[4].对于数据的收集,可以使用以下 3 种方法之一:1) 使用处理器中的硬件计数器;2) 在程序中加入指令计数代码;3) 在指令集级别解释程序,并统计编辑各种指令的计数.但这种“数据驱动”设计技术毕竟是一种事后分析,在处理器设计时能够起到的作用十分有限.

为了提高体系结构设计的质量、减少计算机系统研发成本和失败的风险,在“数据驱动”技术的基础上,发明了性能分析模型技术.性能分析模型主要是对计算机系统的资源及资源之间的连接使用数学工具进行建模,从而通过估计资源利用率以及时间方面的特性来预测计算机系统的性能[5].从静态方面来看,计算机系统是由一系列的硬件和软件资源组成的;从动态方面来看,多个任务或作业访问或竞争这些软、硬资源的使用权.因此,使用以排队论为基础的队列网络来对计算机系统进行建模是十分自然的.为了准确刻画计算机系统特征,数理统计方法、马尔可夫模型和Petri网也是必不可少的数学工具[6].实际上,计算机系统性能分析模型应用也极大地刺激了排队论及相关数学理论的发展.在 20 世纪 70 年代和 80 年代初,计算机系统分析模型曾被广泛地应用于计算机系统的性能评估.因为它基于数学方程的高效解题方法,所以,分析模型是一种低成本的性能评估和预测工具.然而,为了使分析模型中的数学方程易于求解和理解,人们不得不对队列模型进行简化,所以,分析模型不能精确地反映计算机系统的所有细节.但对许多系统来说,分析模型所提供的关于关键资源和负载需求的评估精度足以用来对系统的性能瓶颈和影响性能的关键参数进行分析.一般说来,性能分析模型对作业吞吐率和资源利用率的评估精度可达 5%~10%,对平均响应时间的评估精度可达 10%~30%[7].这样的精度水平在以下 3 个方面是足够使用的:1) 计算能力规划;2) I/O子系统性能评估;3) 新型计算机系统的初期辅助设计.

然而,如上所述,分析模型对计算机系统性能评估的精度很低,能适用的范围也很窄.而在计算机系统的设计后期,需要做出更多的设计决策.在计算机市场竞争日益激烈的时代,一个小小的决策或许就决定了一个产品的成败.体系结构师需要更精确且适用范围更广的技术来辅助设计.体系结构软件模拟正是这样的一种技术,它使用C或Java这样的高级编程语言将新的设计实现为一个虚拟的计算机系统,在已有的计算机系统上运行,从而

对新设计系统的性能进行分析和预测.体系结构模拟技术具有极大的灵活性,它可以在任意详细程度的级别上对计算机系统进行模拟,包括极度复杂的级别,如时钟周期级别,这是分析模型难以胜任的.另外,分析模型只能提供性能预测的平均值,对计算机系统中的动态交互行为无法进行分析.而体系结构模拟技术不仅可以用来研究计算过程中的动态行为及特性,而且可以进行程序执行的全过程研究,包括初始化阶段、稳定阶段和终止阶段.因此,从 20 世纪 80 年代中期以来,体系结构模拟技术开始变得越来越流行.该技术可分为功能模拟和性能模拟两部分.功能模拟器模拟目标体系结构的功能,用于预测和验证新的体系结构是否满足功能要求.它主要关注数据的正确性.例如,模拟器中某个寄存器的值与被模拟计算机的相应寄存器的值要求一样[6].性能模拟器以评估计算机系统与时间有关的特性为目标.这类模拟器并不主要模拟计算机系统中的数据,但每个操作所要花费的时钟周期数和与数据地址信息却是要准确模拟的.当然,一个体系结构模拟器中可以同时以性能和功能模拟为目标,但这种模拟器的性能比单目标模拟器的性能要低.早期的体系结构模拟器使用的是跟踪驱动模拟技术.在 20 世纪 90 年代,这种技术是主流的模拟技术.但随着计算机系统复杂程度的提高以及跟踪驱动技术本身的缺陷,更有优势但成本也更高的执行驱动模拟技术逐渐流行起来.到目前为止,执行驱动模拟依然是处理器体系结构模拟领域最为主流的技术.

2 与体系结构模拟技术相关的几个概念

体系结构软件模拟技术经过几十年的发展,不仅产生了许多不同的模拟技术,而且也创造了许多概念.其中有些概念被人们广泛使用,但也经常混淆,不利于进行学术交流.本文力图区分两个最难区分的概念:模拟(simulation)和仿真(emulation).无论是从汉语还是从英语的角度看,这两个概念的意义都很相近,甚至可以等同.因此,计算机领域的研究人员经常混合使用这两个概念,也经常造成交流上的困难.实际上,体系结构软件模拟过程中存在许多不同的情况,只使用一个概念来指代是不够的.这里尝试给出计算机体系结构模拟和计算机体系结构仿真概念的定义,并在后面的论述中按照这里的定义使用这两个概念.

定义 1. 计算机体系结构模拟.计算机体系结构模拟是使用软件或硬件原型的方法来模拟某种体系结构计算机系统的计算过程,其目标是对未来计算机体系结构的功能或性能进行预测和分析,其重点是研究计算机执行计算任务的过程.例如,计算机在运行程序时,缓存的命中率、访存的平均延迟等程序运行结果不是主要关注目标.

定义 2. 计算机体系结构仿真.计算机体系结构仿真是使用软件或硬件原型的方法来模仿某种体系结构计算机的计算结果,其目标是在另一平台上重现一个已有计算机系统的功能,重点是确保某个计算任务在另一平台上得到与原系统一样的计算结果.计算任务执行的过程不是主要的关注目标.

一般来说,在计算机体系结构模拟中包含了部分计算机体系结构仿真.体系结构模拟中首先要得到正确的输出结果,在此基础上研究得到输出结果的过程.例如,在计算机体系结构模拟中,在模拟器上运行使用某种指令集的程序,而模拟器运行的宿主机却往往使用另一种指令集.因此,需要将模拟器使用的指令集翻译为宿主机的指令集.如果只关注指令集翻译得是否正确,那么,为此开发的转换工具可以被称为指令集仿真器.如果在此基础上增加功能用于仔细研究转换的过程,如转换过程的性能瓶颈,则该指令集仿真器就演变成了一个指令集模拟器.

3 体系结构模拟的分类

由于体系结构软件模拟技术的研究与开发已有相当长的历史,因此,该技术呈现了多样化的局面.目前,使用各种技术的体系结构模拟器已经有了上百种之多,其分类也较为复杂.本节首先介绍体系结构模拟器分类的总体情况,然后详细讨论几种主流的分类方法.

3.1 体系结构模拟器分类概览

根据处理器的个数,可以将性能模拟分为单处理器系统模拟和多处理器系统模拟.例如,SimpleScalar[8,9]是

一个单处理器系统模拟器开发工具集;Rsim[10]是一个指令级并行、共享内存多处理器系统体系结构模拟器的典型代表.根据模拟目标的异同,可以分为功能模拟、性能模拟、能耗模拟、发热模拟等.Ravel[11]即是一个体系结构功能模拟器的例子.对于性能模拟,目前已有了许多模拟器.在这些体系结构性能模拟器中,SimpleScalar是最为流行的.随着计算机系统的日趋复杂,系统的能耗和冷却问题变得越来越重要,甚至成为计算机系统设计的关键之一.SimplePower[12]和Wattch[13]就是为研究处理器系统能耗问题而设计的模拟器,而HotSpot[14]是研究处理器系统发热情况的模拟器.也有将能耗和发热问题放在一个模拟器中研究的方法,文献[15]研究了与内存相关的能耗与发热问题.根据模拟器开发和执行的模式,体系结构模拟器可以被分为两类:串行结构化模拟器和并行结构化模拟器.所谓串行结构化模拟器是指使用类似于C或C++这样的串行编程语言来开发的模拟器,这种模拟器在运行时是通过串行调用函数来实现功能或性能模拟的.目前,绝大多数体系结构模拟器都属于这一类,如SimpleScalar,SMARTS[16]和我国中国科学院计算技术研究所开发的Sim-Godson[17]等.并行结构化模拟器是用特殊编程语言开发的,它们不同于C或C++这类串行执行的编程语言.LSS[18,19]和SystemC[20]是这种模拟器开发语言的代表.使用这种编程语言可以直接描述计算机系统中的硬件组件,生成的模拟器可以像硬件组件一样并发执行.并行结构化体系结构模拟器或模拟器开发环境有HASE[21],Ptolemy II[22]和LSE[18,19,23]等.从对多线程的支持来看,体系结构模拟器可以分为细粒度多线程体系结构模拟器和粗粒度多进程体系结构模拟器.例如,Joshua[24]等人在SMARTS的基础上开发了细粒度多线程体系结构的模拟器,Rsim实际上也是多进程体系结构模拟器.根据模拟计算系统中组件的不同,可以分为流水线模拟、分支预测模拟、Cache模拟、内存系统模拟、网卡模拟等.在SimpleScalar模拟器开发工具集中,提供了对主要单个处理器部件的模拟工具.根据模拟的范围是否包括最终的应用程序、操作系统和I/O设备,可以分为用户态模拟和全系统模拟.用户态模拟不包括对操作系统的模拟,如SimpleScalar和我国的Sim-Godson.全系统模拟包括对操作系统行为的模拟,其开发难度更大,典型的例子是Simics[25],SimOS[26]和中国科学院计算技术研究所开发的SimOS-Goodson[27].根据模拟器模拟的“可执行内核”数量,可以分为单核系统模拟和多核系统模拟,SimOS-Goodson和GEMS[28]的多核版就是多核模拟器的例子.总体来看,这些类别都是从某一个角度来分类的,类别之间存在大量的交叉现象.随着时间的推移,体系结构模拟必将出现更多的种类.下面详细介绍两种主要分类:跟踪驱动模拟和执行驱动模拟.

3.2 跟踪驱动模拟

跟踪驱动模拟技术最早被用于页面置换算法[29,30]以及高速缓存管理算法中.跟踪驱动模拟将每条指令顺序执行所产生的所有信息作为模拟器的输入,从而模拟某种体系结构处理器的功能和性能[6].跟踪驱动模拟技术采用了分治法(divide-and-conquer)策略.如图1所示[26],假设要模拟执行的程序负载为w,该程序负载运行的目标环境为T,则在环境T中模拟执行w的问题可以分为两个子问题:跟踪信息的生成和基于跟踪信息的模拟.称跟踪信息生成的环境为G,则当在G中执行w时所发生的所有事件均被记录下来,这些记录被称为跟踪信息.然后,模拟器将跟踪信息作为输入,从而模拟在目标环境T中执行w的行为.

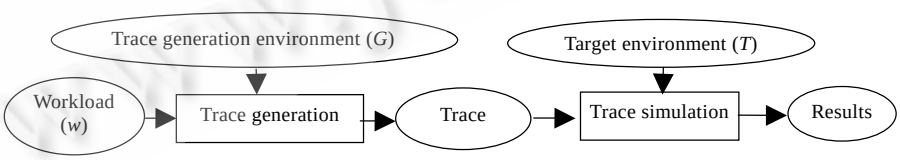

Fig.1 Trace-Driven simulation process

图1 跟踪驱动模拟过程

跟踪驱动模拟比较简单且容易理解;跟踪驱动模拟器也容易进行调试;实验数据可以重现,因为使用同一跟踪信息,在不同的模拟执行中,模拟器的输入、输出数据不会发生变化.但是,跟踪驱动的模拟器存在以下两个方面的问题:

(1) 模拟运行时间可能会非常长,用于存储跟踪信息和模拟信息的存储设备的容量要求非常大.现实情

况下,这个时间和空间的巨大要求往往难以满足;

(2) 跟踪信息在有程序分支预测的情况下不能表示真实的处理器指令流.

对于第(1)个问题,可以使用跟踪信息采样或减少跟踪信息的方法来尝试解决;对于第(2)个问题,可以使用重建错误分支预测路径的办法加以解决.

3.3 执行驱动模拟

对于执行驱动模拟,在体系结构研究者中有两种观点:一种观点认为,执行驱动模拟使用程序的可执行指令作为模拟器的输入,并不使用程序的跟踪状态信息作为输入进行模拟.因此,模拟器的输入数据量只与程序的静态指令数目成比例,而不与动态指令数目成比例.如图 2(a)所示,程序的所有可执行指令由软件模拟器执行.在学术界和工业界广泛使用的SimpleScalar就是执行驱动模拟的典型代表.另一种观点认为,执行驱动的模拟器依赖于在宿主机上实际运行一部分代码,在软件模拟器上运行另一部分代码,如图 2(b)所示.因此,这类模拟器不模拟执行一个应用程序的所有指令.只有对问题研究有用的指令才会被模拟执行,其他指令直接在宿主机硬件上加速执行.这样做有一个限制,就是只有当宿主机的指令集和被模拟的处理器体系结构的指令集一样时才有意义.这种类型的模拟可以分为两个阶段:第 1 个阶段被称为预处理阶段,主要是修改应用程序,在感兴趣的位置插入调用模拟子程序的语句.例如,对于一个内存管理系统模拟器,如果用户只对内存访问感兴趣,就只在应用程序中进行内存访问时调用模拟程序;第 2 个阶段为模拟阶段,即在宿主机和模拟器上交替执行应用程序.Tango, Proteus和FAST都是这类模拟器的代表[6,31].

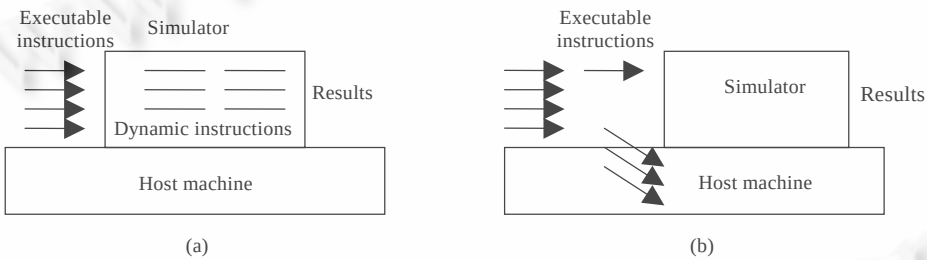

Fig.2 Two execution-driven simulation strategies

图 2 两种执行驱动模拟策略

与跟踪驱动模拟技术相比,执行驱动更加接近于计算机硬件运行程序的真实情况.即用高级语言编写应用程序、编译和连接,得到应用程序的可执行指令,然后在计算机上运行.执行驱动模拟在模拟的过程中会模拟动态指令的生成和分支预测等过程.因此,执行驱动模拟的过程是更真实的.然而,执行驱动模拟的成本却比跟踪驱动模拟的成本要高,执行速度更慢,因为在模拟的过程中需要使用软件的方法来生成动态指令.从 20 世纪 90 年代末到目前,执行驱动模拟是最主流的体系结构模拟技术.

4 知名模拟器或模拟器工具集介绍

4.1 SimpleScalar

SimpleScalar最初是由Todd Austin在其攻读博士期间开发的一个单处理器架构的性能模拟器工具集.它是目前学术界最为流行的体系结构模拟器.据统计[32],第29 届Annual International Symposium on Computer Architecture (ISCA 2002)中,超过一半的文章使用了SimpleScalar来模拟研究人员的体系结构设计.SimpleScalar体系结构模拟器开发工具集包含了非常丰富的工具,可以配置成许多不同体系结构的模拟器.在SimpleScalar 4.0 版本[33]中,除了上述的基础工具之外,还集成了微体系结构模拟环境MASE、ARM体系结构模拟器、流水线可视化工具GPV、嵌入式性能测试程序包MiBench、能耗分析器Power Analyzer、Alpha 21264 处理器体系结构验证模型、PowerPC体系结构性能模拟器SS-PPC、全系统模拟器SS-OS和SimpleScalar可视化工具SS-VIZ等

多种工具.

SimpleScalar 体系结构如图 3 所示.目标应用程序,如处理器标准性能测试程序 SPEC CPU2000 和操作系统可以运行在使用 SimpleScalar 工具集开发的处理器性能模拟器上.图 3 中,第 1 层是 SimpleScalar 抽象出的硬件模型,如取指令器、流水线、分支预测器、高速缓存和其他核心模型等.这些模型既可以单独使用,也可以集成到一个完整的体系结构模拟器中;第 2 层是模拟内核,主要提供事件模拟服务;第 3 层主要提供目标指令集体系结构(即要模拟的指令集)和目标 I/O 接口.目前,可以支持的目标指令集体系结构有 PISA,Alpha,PowerPC 和 x86 等.

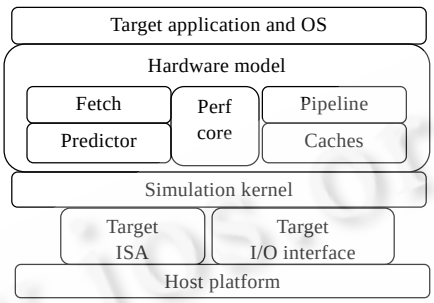

Fig.3 SimpleScalar architecture

图 3 SimpleScalar 体系结构

在 SimpleScalar 中,不同硬件部件间信息交换的模拟是通过函数参数传递和全局变量实现的.这种方法容易造成信息不加限制地流动而产生许多副作用.另一个缺点是,由于大量使用全局变量,使得程序结构变得比较混乱,使程序维护成了一个沉重的负担.所以,要理解和维护 SimpleScalar,特别是更改它以适应多种不同的、新的设计需要花费较多的时间和精力.

4.2 Asim

Asim[34]是一个模拟器开发环境,提供了大量的模块和程序库,由Intel公司的Joel Emer及其同事以及Compaq公司的一些工程师共同开发.Asim针对SimpleScalar程序难以管理和难以重用的缺点,采用面向对象技术很好地重新设计了体系结构模拟器开发环境.

在 Asim 中,一个计算机系统硬件模块的结构使用 C++的类机制进行描述,其动态操作用类的方法来进行模拟.由类生成的实例对象是软件模拟器的组件,比较自然地反映了计算机系统是由一个个模块化的硬件组件构成的特性.因此,Asim 从很大程度上降低了体系结构模拟器的开发及维护难度.具体来讲,Asim 的特点主要表现在以下几个方面:

1) 标准端口互连.计算机硬件之间是通过端口相互连接的,Asim 相应地定义了标准的类接口来模拟组件的端口.在模拟器的开发过程中,只要某个环境中需要某种类型的端口连接,则所有具有这种类型的类的对象就可以在这个位置使用.这样做非常有利于代码重用和提高程序的可扩展性.除了用于模拟硬件模块之间的连接和信息传送之外,Asim 还使用端口来模拟硬件模块之间和模块本身在时间上的延迟.

2) 设计方案管理.一个处理器或完整的计算机系统是由上百个部件组成的.在设计时每个部件都可能有多个可选方案,每个部件之间又相互依赖,没有完善的机制来管理这些组件及其之间的依赖关系是难以完成模拟器开发任务的.Asim 针对这个问题提供了两套机制:(1) 模块类型;(2) 使用单独的模块依赖信息文件.awb 文件.在 Asim 中定义的每个模块都通过 Asim 模块类型来描述该模块的功能和接口,同时还描述了能与之相连的模块或模块类型.另外一个机制是将模块及模块之间的依赖关系使用特定的格式进行描述,并把它们存在一个单独的.awb 文件中,Asim 处理引擎会自动地对这些文件进行处理,分析出模块之间的依赖关系.

3) 信息输入器(feeder).在体系结构模拟领域,模拟执行速度一直是一个瓶颈问题.为了提高模拟器的性能,在短时间内得出模拟结果,人们往往只对影响性能的操作进行详细模拟.对于其他操作,研究人员使用多种技术

来避免详细的模拟执行以加快速度.Asim 为此利用一个特殊的模块 Feeder 来提供驱动性能模拟的信息输入.Feeder 模块要完成的关键任务是提供程序(可以是应用程序或标准性能测试程序)指令.针对不同问题,Asim 提供了 3 种类型的指令输入器(instruction feeder):静态指令跟踪信息输入器(static instruction trace feeder)、动态指令跟踪信息输入器(dynamic instruction trace feeder)和 Aint Feeder.

在 SimpleScalar 的基础上,从简化模拟器开发过程方面,Asim 做了许多开创性的工作,但它仍然还存在许多问题,如模拟速度和模拟精度的问题.在 Asim 中,还存在目前不支持并行化、支持的指令集也非常少、对系统时钟的建模还不够灵活等缺点.

4.3 SimOS

SimOS是由Stanford大学于 1992 年开始研发的用于研究操作系统行为的软件模拟环境[35].事实上,SimOS是一个全系统模拟器,它不仅可以用来研究在某种体系结构上运行的操作系统,也可以用于研究计算机体系结构.它最早用于模拟基于MIPS(microprocessor without interlocked piped stages)指令集的多处理器计算机系统,并可足够详细地模拟在其上运行的IRIX操作系统的启动和运行过程.SimOS模拟环境的总体结构如图 4 所示.

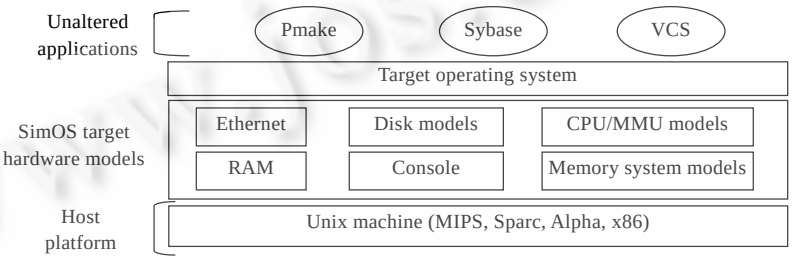

Fig.4 SimOS simulation environment

图 4 SimOS 模拟环境

从图 4 中可以看出,SimOS模拟环境的主要特点表现在两个方面:(1) 全系统模拟;(2) 可互换的模拟模型.SimOS包含了现代计算机系统中所有硬件部件的软件模拟.例如处理器、内存管理单元(memory management unit,简称MMU)、高速缓存、内存、I/O设备、网卡、硬件时钟和控制台等.对计算机系统中的每个硬件部件,SimOS提供了多个与之对应的软件模拟模型.与每个硬件部件对应的多个模拟模型在功能上是一致的,其功能模拟的详细程度足以正确地运行操作系统和应用程序.但这些模型在时间精度上却存在着很大的差异,如模拟速度、统计采样的时间间隔和多处理器执行的时间间隔等.每个模拟模型都支持转换到另一个与同一个硬件对应的模拟模型,这就使得SimOS允许用户在模拟过程中可以动态地选择用来模拟某个硬件部件的模型.为了支持对MIPS处理器和内存系统的高速仿真,SimOS包含了一个称为Embra[36]的模拟器.Embra主要借用了Shade[37]系统的动态二进制块翻译方法.这样,模拟器就可以运行与宿主机不同指令集的操作系统和应用程序.二进制翻译器实际上是一种指令集仿真器,其速度比详细模拟过程细节的模拟器的速度要高很多.因此,严格地说,SimOS在提高模拟速度方面使用了仿真技术.另一方面,为了正确而详细地对计算过程进行模拟,SimOS也在牺牲模拟速度的情况下,为同一硬件部件提供了详细的模拟模型.在模拟器上完整而详细地模拟运行一个大的应用程序,如处理器标准性能测试程序,在时间上往往是不允许的.解决的办法之一是部分指令仿真运行,对计算机系统性能有重要影响的指令详细模拟运行,SimOS为一个硬件部件提供多种类型的模拟模型正是为了达到这个目的.

SimOS作为一个全系统模拟环境,还包含许多其他技术,详细信息可参考文献[35,38,39].SimOS在 20 世纪 90 年代是一个非常成功的模拟器,并产生了深远的影响,被广泛地应用到许多研究中.Stanford大学的并行编译器SUIF[40]研究就使用了SimOS,中国科学院计算技术研究所的SimOS-Goodson[27]全系统模拟器也是在SimOS的基础上开发的,但SimOS的运行速度已难以满足目前对模拟器的速度要求.

4.4 SMARTS

SMARTS[16]是由Carnegie Mellon大学开发的模拟器开发环境.如前所述,在研究计算机体系结构时,研究人员通常使用处理器标准性能测试程序在体系结构软件模拟器上运行,从而评估体系结构设计.然而,在体系结构软件模拟器上完整地执行测试程序的所有指令,需要极长的模拟运行时间,研究人员通常只能选取部分指令进行详细模拟执行.如何在测试程序的完整指令中选取部分指令详细模拟执行,而又能代表完整指令执行的结果是一个非常关键的问题.许多研究人员对这个问题进行了研究,SMARTS是目前最好的解决方案之一.

SMARTS模拟器开发环境的基础是SimpleScalar.其主要贡献是实现了系统采样模块.即采用系统采样的方法从标准性能测试程序的动态指令中选取少部分指令进行详细模拟执行.SMARTS 中采用的系统采样技术如图5所示.

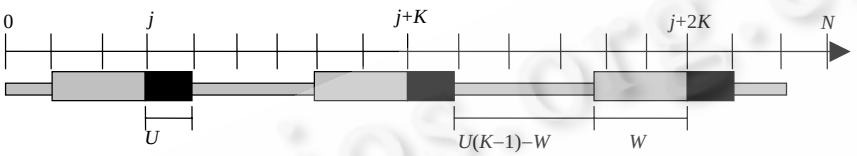

Fig.5 System sampling in SMARTS

图5 SMARTS中的系统采样

假设一个标准性能测试程序的动态指令数为 L,采样单元是动态指令中连续的 U 条指令,则SMARTS中采样的样本空间大小为 $N=L/U$.为了与时钟周期边界对齐,采样单元中 U 的数目可能会有轻微的变化,但这无关紧要.在系统采样中,模拟执行区间是固定大小的.即在采样前选好模拟执行区间,在采样过程中区间长度不变.假设执行区间是 K 个连续采样单元(U)的动态指令,并从某标准性能测试程序动态指令的第 j 个采样单元处开始采样,如图5所示,则SMARTS将详细模拟执行 U 条连续的指令,然后以快速向前的方式模拟执行 $U(K-1)-W$ 条指令,再以“预热”的方式模拟执行 W 条指令.然后再开始详细模拟执行 U 条连续指令.如此周期性地重复,直至执行完性能测试程序的所有指令.快速向前执行的 $U(K-1)-W$ 条连续的指令是详细程度很低的模拟执行,在此过程中,模拟器忽略了许多内部状态的记录,因此可以快速向前执行.快速向前执行的 $U(K-1)-W$ 条指令的模拟详细程度可以由研究人员决定.如果快速向前执行一段指令后,立即开始详细模拟执行,在进行详细模拟执行时模拟器中许多状态还没有数值,这显然不符合硬件部件执行中的真实情况.为了克服这个缺点,SMARTS 使用了“预热”执行 W 条指令的方式.它实际上是按照详细模拟的方式执行的,虽然这 W 条指令执行的过程状态被记录下来,但并不作为计算模拟结果的依据.

从SMARTS模拟执行的过程来看,U,K 和 W 这几个参数是需要确定下来的.这些参数的确定,可以根据要达到的模拟精度和置信区间的大小,使用数理统计和概率论的方法确定,并使用模拟执行的结果来进行检验.从效果上看,SMARTS 是目前执行速度最快、模拟精度最高的体系结构软件模拟器,但其速度与当前的需求仍有较大差距,更糟糕的是 W 往往难以确定.

4.5 LSE

LSE的全称是Liberty Simulation Environment.它是由Princeton大学开发的一套用于计算机系统建模的开发环境.该开发环境旨在对计算机系统的并行结构化部件进行建模,然后自动生成模拟器.在模拟器的开发过程中最大限度地重用已有的组件,从而降低开发成本和难度.利用LSE的模拟器开发过程如图6[23]所示.

在图6中,利用LSE开发体系结构软件模拟器的第1步是使用LSS(liberty structural specification language)语言对拟模拟的体系结构进行描述.LSS是LSE中定义的一种硬件描述语言,主要对计算机系统中的并行结构化部件进行静态描述[18].如处理器中的算术逻辑部件(arithmetic logic unit,简称ALU)、高速缓存及地址转换缓冲等.该描述以*.lss文件保存,一个lss文件就是一个某种抽象级别的处理器部件.获得了lss文件以后,需要使用LSE提供的编译器对其进行解释.在解释的过程中,LSE组件库中的模块描述库(LSS module description)将被调

用.LSE中的模块描述库有两种:一种是LSE提供的核心模块库[41],另一种是模拟器开发者利用LSE提供的模块定义机制自定义的模块库.两种模块库中的每一个模块都可以使用两种类型的参数:变量型参数和userpoint参数.变量型参数使用一个变量作为参数,用于规定模块在静态结构方面的某些属性;userpoint参数是LSE设计的一种特有的参数类型,它允许用户编写一段代码作为参数的内容,用于规定模块在运行时的动态行为.经过LSE编译器编译的LSS文件只刻画了计算机系统各部件纯粹意义上的连接,还没有对这些连接进行合法性、一致性检查.因此,下一步就是利用LSE提供的静态结构分析引擎(static analysis engine)给连接加上约束,并检查这些连接和约束的合法性、一致性.如果lss文件成功通过了静态结构分析引擎的分析,就得到了具有正确连接约束的系统结构模型.最后,LSE代码生成器(code generator)通过调用组件运行时行为库(component runtime behavior)生成可执行的模拟器.

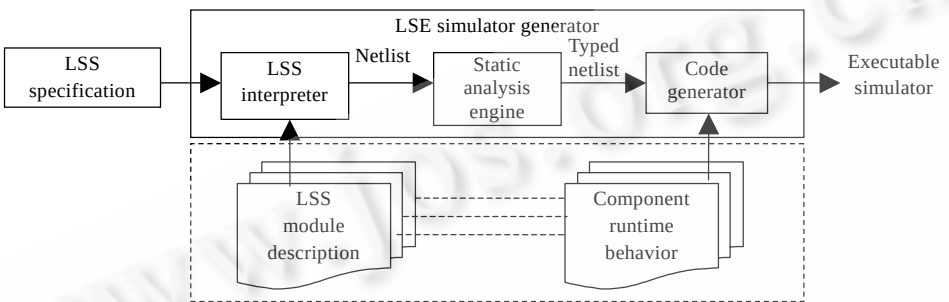

Fig.6 LSE simulator generating process

图 6 利用 LSE 的模拟器生成过程

LSE 是目前最优秀的处理器体系结构模拟器开发工具之一.它的特点主要体现在以下几个方面:(1) 提供了并行结构化建模机制;(2) 提供了基于模块的重用机制;(3) 提供了静态模型分析机制;(4) LSE 可以作为一个处理器通用模拟器构建框架.LSE 在较大程度上简化了模拟器的开发过程,但其速度和精度还有待提高.

4.6 Sim-Godson

Sim-Godson[17]是我国自主研发的第一个通用高性能处理器软件模拟器,由中国科学院计算技术研究所为研究与开发龙芯 2 号通用处理器而开发完成.Sim-Godson是在SimpleScalar模拟器工具集的基础上开发的,但由于龙芯 2 号处理器在指令集和体系结构方面与SimpleScalar所支持的指令集与体系结构存在较大差异,Sim-Godson做了许多独创性的工作.主要包括两个大的方面:(1) 实现了对MIPS指令集体系结构的支持;(2) 实现了龙芯 2 号处理器微体系结构.

SimpleScalar 没有包括对龙芯 2 号处理器所采用的类 MIPS 指令集的支持.Sim-Godson 做了 3 个方面的工作,实现了对 MIPS 指令集的支持:(1) 在 PISA(portable instruction set architecture)指令集的基础上实现了目标机器定义.包括对 MIPS 指令的译码和定义,处理 MIPS 的转移指令延迟槽和 likely 类的转移指令;(2) 实现了目标二进制文件的装载;(3) 实现了对目标操作系统调用的代理.

利用 SimpleScalar 的基础模块,Sim-Godson 实现了龙芯 2 号通用处理器的微体系结构.包括流水逆向模拟执行、多重误预测支持、寄存器重命名、功能部件、龙芯 2 号多指令发射、优化访存这 6 个方面的关键技术.SimpleScalar 中 Sim-outorder 只能支持一个层次的错误猜测,即在猜测模式下再次猜错时,Sim-outorder 不会再更正了.Sim-Godson 引入了错误层次,增强了指令执行引擎的体系结构状态读写接口,从而实现了多重误预测.寄存器重命名是现代处理器为解决数据相关而普遍采用的提高流水线吞吐率的技术,Sim-Godson 中也模拟实现了这项技术,并使用了基于全相联查找的寄存器重命名方式.在模拟功能部件方面,虽然 Sim-Godson 采用了与 Sim-outorder 类似的方式,但还是针对龙芯 2 号通用处理器的特点,模拟了可变延迟的功能部件和端口竞争的情况.对指令发射,Sim-Godson 使用复杂度较低的算法精确模拟了龙芯 2 号的多指令发射情况.最后,Sim-Godson 优化了 Load/Store 队列,简化了模拟访存的逻辑,从而在一定程度上提高了模拟器模拟执行的速度.

Sim-Godson是一款优秀的国产通用处理器模拟器,用电路级的硬件模拟器ICT-Godson作参考,其模拟精度达到了 95%.Sim-Godson 的模拟运行速度也较高,达到了 0.5MIPS,达到了处理器软件模拟器领域的国际先进水平.

5 体系结构软件模拟领域面临的问题及解决方案比较

如前所述,在体系结构软件模拟技术领域存在3大问题:即体系结构模拟器的开发难度大,开发周期长;详细和完整模拟情况下的模拟执行时间非常长,现阶段还无法接受如此长的模拟时间;模拟结果的精度还较低,还无法通过单一的模拟方式来辅助计算机系统体系结构的设计.多年来,研究人员为解决这 3 个问题进行了大量的研究,但大部分的研究集中在如何解决后两个问题上.

在现有的技术条件下,提高模拟速度和模拟结果的精度实际上是一对矛盾.因为目前在体系结构软件模拟器上详细模拟运行计算机系统或处理器标准性能测试程序的全部动态指令所耗费的时间是无法让人接受的.人们只能选取部分指令详细模拟执行来代替模拟执行完整动态指令的模拟结果.选取的指令越多,越逼近完整指令的模拟结果,其模拟精度越高;反之,其模拟精度越低.在处理器性能模拟研究领域,研究所采用的事实上的标准性能测试程序是SPEC组织的SPEC CPU2000或SPEC CPU2006.因此,为减少模拟运行的时间,主要的研究方法可以分为两类:(1) 减少标准性能测试程序的输入参数集;(2) 使用各种技术减少在模拟器上运行的动态指令数.在SPEC CPU2000中,一共有26个性能测试程序,从不同的方面测试处理器的性能.其中,有12个整数性能测试程序和14个浮点数性能测试程序,每个性能测试程序都有3套输入参数,被称为测试输入参数集、训练输入参数集和参考输入参数集.参考输入参数集是每个性能测试程序最大的输入参数集合.通常情况下,在测试处理器性能时,每个测试程序只有在使用各自的参考输入参数集时,才能更准确地测试处理器的真实行为,但在进行处理器性能模拟研究时却很少使用参考输入参数集,因为使用参考输入参数集时需要耗费太长的模拟运行时间[42].

对于性能模拟,无论采用哪种方法或技术,都需要对模拟的结果进行衡量以确定这些方法或技术的有效性.目前,主要有3种指标来衡量模拟技术的效果:1) 计算系统或处理器的性能瓶颈;2) 程序执行轮廓;3) 体系结构级的特征.对于第 1 项指标,主要是看模拟器分析出的性能瓶颈与实际计算系统的性能瓶颈是否一致.性能瓶颈分析可以使用Yi等人开发的P&B Design[43]工具进行,它可以分析出哪些因素对系统的影响最大;第2项指标是分析模拟器执行性能测试程序所产生的指令跟踪信息的分布轮廓与真实系统执行性能测试程序所产生的指令跟踪信息分布轮廓的相似度.为了刻画一个程序的执行轮廓,通常会将一个程序的指令划分成程序基本块[42]或程序阶段[44,45],程序基本块指的是两个分支指令之间所含的指令.刻画执行轮廓时,考察每个基本块的执行频率(basic block execution frequency,简称BBEF)和每个基本块内部的指令数(basic block vector,简称BBV).通过比较参考输入参数集和其他输入参数集的BBEF以及不同技术的BBEF,就可以从代码覆盖率方面来获得某种模拟技术的精度;第 3 项指标在于衡量模拟器是否能够准确刻画计算系统体系结构级的特征.体系结构级特征的刻画方法是首先向量化一组测量指标,如IPC、分支预测精度、一级高速缓存命中率、二级高速缓存命中率等,然后对每个测量指标的测量值进行归一化处理,最后计算向量之间(模拟器得出的指标向量和真实机器得出的指标向量)的欧拉距离.该欧拉距离越小,则模拟器刻画的体系结构级特征的精度越高.对于这 3 项指标,在性能模拟领域应该从3个方面来加以分析:(1) 性能瓶颈保真度,即模拟器所发现的性能瓶颈与真实系统性能瓶颈之间的吻合程度.现实中,模拟器分析出的性能瓶颈与真实情况之间往往会存在偏差;(2) 配置依赖特性,为使用模拟器评估不同结构计算系统的性能,通常模拟器会设计许多配置参数,如果在众多的配置参数组合下,模拟器所模拟的结果的精度能够保持一个常数,则该模拟器是一个好的模拟器.这也是模拟器研究应达到的关键目标,因为模拟器模拟的主要目标是评估还不存在的计算系统或处理器的性能;(3) 速度与精度的折衷程度,即希望模拟执行速度快,同时所能达到的模拟精度也高,虽然这是一对现实的矛盾.

5.1 减少标准性能测试程序的输入参数集

减少标准性能测试程序输入参数集的基本思想是,修改性能测试程序的参考输入参数集,从而减少性能测试程序在软件模拟器上的运行时间.这种方法的一个特点是依然运行标准性能测试程序的所有指令,但使用减少了的输入参数集来模拟运行,并使用该模拟运行结果来代表性能测试程序使用参考输入参数集的模拟运行结果[46].这种方法由Minnesota大学提出,并进行了大量的研究,开发了针对性能模拟器的性能测试程序输入参数集程序包MinneSPEC.与SPEC CPU2000中每个测试程序的3套输入参数集对应,MinneSPEC为每个性能测试程序也提供了3套减少了的输入参数集合,分别称为small,medium和large输入参数集合.使用MinneSPEC中提供的输入参数集和使用SPEC CPU2000中原有的输入参数集运行测试程序,其运行结果之间的差异非常大[42],只有少部分的结果吻合得较好[46].因此,该方法不能较好地解决速度与精度的矛盾.

5.2 减少模拟运行的指令数

减少模拟运行的指令数是在使用标准性能测试程序参考输入参数集的情况下,选取部分指令模拟运行,并使用这部分指令模拟运行的结果来代替整个测试程序指令模拟运行的结果.如何选取部分指令进行模拟运行是影响模拟执行速度和模拟结果精度的关键因素之一.根据选取部分指令的方式,减少模拟运行指令数方法可以分为两大类:(1) 直接截取一段连续的指令;(2) 使用统计采样方法选取模拟运行的指令.

(1) 直接截取一段连续的指令

直接截取一段连续的指令主要是在标准性能测试程序指令中任意选取一段连续的指令在模拟器上执行,并由此结果来代表性能测试程序所有指令模拟执行的结果.值得注意的是,这种方法只选一段指令而不是多段指令模拟执行的结果作为最终的模拟结果,但每个测试程序模拟执行时使用参考输入参数集作为输入参数.这种方法的主要代表是Run *Z*和它的两个改进版FFX+Run *Z*,FFX+WU Y+Run *Z*.Run *Z*详细模拟执行一个性能测试程序最初的 *Z* 百万条指令,模拟执行的时间由 *Z* 的值决定.使用最初的 *Z* 百万条指令来代替所有指令模拟执行的结果显然不合理,因为最初的指令中包含了性能测试程序在模拟器上运行的初始化工作,而初始化时模拟器还没有达到稳定状态.为了克服这个缺陷,出现了FFX+Run *Z*技术.这种技术先快速向前执行 *X* 百万条指令,再详细模拟执行 *Z* 百万条指令,并以此结果作为最后模拟的结果.所谓快速向前执行是相对详细模拟执行而言的,模拟器在模拟执行时忽略了许多细节状态的产生和记录,如指令发射、内存状态等,从而快速向前执行.FFX+Run *Z*是对Run *Z*的改进,但它立即引来另外一个问题:快速向前执行 *X* 百万条指令以后,模拟器中还不存在与实际硬件对应的许多信息,如控制寄存器和状态寄存器的信息,它们将直接影响后续指令的执行,这时突然开始详细模拟执行实际上不符合实际情况,因此模拟执行的结果并不可信.为了解决这个问题,FFX+WUY+Run *Z*被提了出来.这项技术在快速向前执行 *X* 百万条指令后,开始详细执行 *Y* 百万条指令作为"预热"执行,最后详细执行 *Z* 百万条指令.实际上,SMARTS中的"预热"执行方式来源于这里.

直接截取一段连续的指令模拟执行(Run *Z*,FFX+Run *Z* 和 FFX+WU Y+Run *Z*)的最大优点是可以极大地减少模拟执行的时间,并且模拟执行的时间可由 *Z*,*X* 和 *Y* 的值来加以调整.然而,这种技术往往不能代表完整指令模拟执行的结果.因此,这种技术只能应用在非常有限的情况下,不能作为一种成功的技术使用在多数模拟环境中.

(2) 使用统计采样方法选取模拟运行的指令

统计采样方法是利用统计学的方法从详细观察的子集的特点来推知全集的特征.该技术的关键是如何确定详细模拟执行的指令来保证模拟结果正确地代表所有指令模拟执行的结果.目前,在计算机体系结构研究领域,主要的统计采样方法有 3 种:1) 代表性采样;2) 周期性采样;3) 随机采样.代表性采样的基本思想是在SPEC CPU2000测试程序的动态指令中选择模拟执行点(一段指令),用模拟执行点的模拟结果来代表整个测试程序的行为.这项技术的典型代表是加州大学圣地亚哥分校的SimPoint[44,45,47].SimPoint模拟分3步进行:首先分析测试程序,找出候选的模拟点;然后,使用基于统计的聚类(clustering)办法选择一组模拟点进行模拟执行,代表整个测试程序的模拟;最后,将所选的每个模拟点的模拟结果进行加权,得出最后的模拟结果.所选模拟点的个数和每

个所选模拟点的长度(即指令数)决定模拟时间.与代表性采样模拟不同,周期性采样模拟是在测试程序的动态指令中以固定间隔周期性地选取部分指令进行详细模拟.显然,可以通过控制采样频率和每个样本的长度来控制模拟执行的时间.这类技术的典型代表是Carnegie Mellon大学的SMARTS[16].随机采样先从测试程序的动态指令中随机抽取*N*段指令进行模拟执行,然后将这些模拟结果以某种方式加以合并,形成最后的模拟结果.这种方法的代表人物是Thomas等人[48].需要指出的是,统计采样技术也采用了直接截取一段连续指令技术中的部分技术.例如,SMARTS中也有FF *X*和WU *Y*部分;Thomas等人曾建议为了减少随机采样模拟的错误,增加WU *Y*中的*Y*值.

5.3 技术比较

根据文献[42]的研究和我们对各种模拟技术的研究与理解,下面将从性能瓶颈分析、程序执行轮廓、体系结构级特征刻画与分析以及速度与精度的折衷和模拟结果配置依赖这 4 个方面来对上述 6 种提高模拟速度的技术进行比较.

(1) 性能瓶颈分析

Run *Z*,FFX+Run *Z* 以及 FFX+WU Y+Run *Z* 在性能瓶颈分析方面的精度非常差.因此,不能使用这 3 种方法来指导计算系统或处理器设计中的性能瓶颈分析.其原因是非常显然的:第一,这 3 种技术中,*X*,*Y* 和 *Z* 值的选择是任意的,很可能所选择的详细模拟执行的指令段是对性能瓶颈分析没有帮助的,并且所选择的这部分详细模拟执行的指令的执行结果不能代表测试程序所有指令模拟执行的结果;第二,对于具有复杂行为特性的性能测试程序,如 gcc,即使快速向前执行了几十亿条指令,再详细模拟执行 *Z* 百万条指令,还是不能分析出性能瓶颈.为了能够分析出性能瓶颈,只有加大详细模拟执行的指令数,然而,这种技术无法预知还应详细模拟执行多少条指令才能分析出性能瓶颈.同时,加大了详细模拟执行的指令数会延长模拟执行的时间,从而让这 3 种技术失去吸引力.而 SMARTS 和 SimPoint 能够较为逼真地分析系统的性能瓶颈.从不同的性能测试程序在 SMARTS 和 SimPoint 上模拟执行的结果来看,某些测试程序在 SimPoint 上执行的结果略好于在 SMARTS 上的执行结果,另外一些测试程序在 SMARTS 上的执行结果则略好于在 SimPoint 上的执行结果.但从总体来看,SMARTS 在性能瓶颈分析方面的精度略高于 SimPoint.

(2) 程序执行轮廓、体系结构级特征刻画与分析

使用减少输入参数集的方法和 Run *Z* 及其改进版所分析出来的程序执行轮廓和体系结构级特征,与使用参考输入参数集的测试程序在同样配置下的实际机器上运行分析出的程序执行轮廓和体系结构特征差别很大.其原因是 Run *Z* 及其改进版只执行了测试程序的一小部分指令,而在减少输入参数集方法中,同一程序基本块执行的频率与使用参考输入参数集时的执行频率是不同的.使用 SMARTS 和 SimPoint 模拟执行性能测试程序和性能测试程序使用参考输入参数集在真实机器上运行所得出的结果非常相似.其中,SMARTS 得出的执行轮廓比 SimPoint 所得出的执行轮廓与真实执行轮廓更相似一些.在体系结构级特征刻画方面,各种技术的结果和程序执行轮廓方面得出的结果是一样的.

(3) 速度与精度的折衷

在不能模拟执行性能测试程序所有指令的前提下,提高模拟执行速度总是以牺牲模拟精度为代价.但体系结构模拟技术的研究目标是,在牺牲最少精度的情况下最大限度地提高模拟执行的速度.虽然在体系结构模拟技术领域模拟结果的精度是最重要的,但在几种模拟技术具有相近精度的情况下比较技术的优劣,模拟速度就成为最重要的一个方面.况且,目前的模拟速度还非常慢,与研究人员的需求尚有相当的距离.因此,模拟速度的有效提高,一直是研究人员追求的最重要的目标之一.

如果定义模拟速度为 *S*,模拟结果的精度为 *P*,则速度与精度的折衷定义为 *Tsp*=*S*/*P*.随着 *S* 的变化,*Tsp* 越接近 *S* 的线性函数越好,即 *S* 变化时,*P* 基本保持一个常量.在速度和精度的折衷方面,减少输入参数集和 Run *Z* 及其变种这些技术的结果非常差,即 *S* 增加时,*P* 会急剧下降.特别地,对于 gcc 这个性能测试程序,Run *Z* 及其变种在增加模拟指令段长度,即增加 *Z* 值时,其模拟结果的精度却并没有增加.SMARTS 和 SimPoint 都有着较好的速度和精度折衷,但 SimPoint 有着更好的速度和精度折衷性.

(4) 模拟结果配置依赖

模拟器的主要目的是评估和预测未来的处理器或计算机系统体系结构设计,如何才能准确地预测未来处理器或计算机系统的性能,是体系结构模拟器研究的关键.为了达到正确预测的目的,可以有两类方法:1) 建立足够精确的分析模型;2) 设计的模拟器基本没有配置依赖.分析模型主要是用数学的方法来计算某种设计的性能,如果数学模型是精确的,那么这种方法是最理想的.但如前所述,在计算系统越来越复杂的今天,要建立整个计算系统的精确分析模型非常困难,使设计的模拟器的配置依赖尽可能少却是有可能的.模拟器包括两种类型的配置:所模拟的目标体系结构的配置,如高速缓存的大小等;模拟器本身的配置,如 SMARTS 中 U 的大小等.

减少模拟输入参数集技术和 Run Z 及其变种的配置依赖非常强,即配置的不同,其所模拟的结果的精度差别非常大.SMARTS 技术基本上没有配置依赖,SimPoint 技术在少数情况下有很强的配置依赖,在大多数情况下基本没有配置依赖.因此在这方面,SMARTS 的表现是最好的.

5.4 提高模拟速度和精度模拟技术小结

目前,体系结构软件模拟器的执行速度非常低,比真实的硬件机器慢 4~5 个数量级,研究人员还不可能在可以接受的时间内详细模拟执行一个标准性能测试程序包中所有程序的所有指令.如前所述,许多研究人员提出了多种提高模拟速度的方法.其总的思想是减少在模拟器详细模拟执行的指令数.从前面的比较可以看出,减少输入参数集、Run Z、FFX+Run Z以及FFX+WU Y+Run Z总体上基本不能解决提高模拟速度同时不降低模拟精度的问题.SMARTS和SimPoint能够比较有效地解决模拟速度和模拟精度的矛盾.SMARTS采用了系统采样技术,周期性地选择部分指令详细模拟执行;SimPoint从另一个方向来选择指令模拟执行,即先分析标准性能测试程序的指令分布特性,找出其中影响性能的关键指令段,然后选择这些指令段详细模拟执行.然而,SMARTS和SimPoint并没有完全解决体系结构模拟领域的所有问题,例如模拟速度,最快的SMARTS模拟器的速度是9MIPS,该速度在现代处理器设计中特别是多核处理器的设计中还是远远不够的.而且这些技术本身的有效性需要进一步的研究证明.Daniel Citron[49]对现有的体系结构模拟技术提出了批评,其中主要的一点是,目前的模拟技术并没有运行标准性能测试程序包中的所有程序,而只选择了部分测试程序模拟执行.对于为什么只选择某几个程序,几乎所有的文章都没有给出合理的解释.Daniel Citron认为,只使用部分测试程序来评估计算系统或处理器体系结构设计会违背Amdal定律.因此,在SMARTS和SimPoints的基础上,还需要做大量的研究工作.

6 SimIPF 模拟器设计

在较为充分的调研基础上,我们正在利用 LSE 的技术和思想开发一个适用于安腾系列体系结构的模拟器SimIPF.选择 LSE 技术主要基于以下几个方面的原因:1) LSE 充分利用了软件模块化和重用的思想,同时有效地降低了由重用带来的额外开销;2) LSE 具有开放的架构,开发人员可以开发自己的模块并作为后续开发的基础.更为重要的是,LSE 技术还可以用来建立处理器的能耗模型、发热模型等;3) 利用 LSE 提供的 LSS 硬件描述语言可以方便地描述硬件组件.另外,由于与其他类型模拟器开发技术所具有的优越性,文献[50]也指出该技术可能是模拟器开发应该努力的方向.开发该模拟器的主要目的有 4 个:(1) 验证 LSE 技术的有效性和先进性,特别是在快速、准确构造模拟器方面的有效性;(2) 利用 LSE 技术开发更多的复合模块,方便以后的模拟器构建;(3) 进一步在LSE的执行速度方面进行研究以提高其构建的模拟器的执行速度;(4) 以LSE为基础框架,集成其他模拟技术,如统计采样技术,构建一个功能更为丰富的模拟器开发环境.因此,开发模拟器 SimIPF 只是我们研究体系结构软件模拟器开发技术的一个实例.

在 SimIPF 模拟器中,我们将较为详细地模拟安腾 2 的体系结构.包括取指令、指令预取、指令缓冲IB(instruction buffer)(在第 2 级流水线和第 3 级流水线之间)、指令发射逻辑(每个时钟周期发射指令数可达 6 条)、分支预测、数据投机和控制投机执行、3 级缓存体系结构和 TLB 等.多数安腾 2 体系结构的新特征将被模拟出来,如 Prevalidated-tag 高速缓存设计、ALAT(advanced-load address table)等.LSE 体系结构模拟器开发环境提供了 16 个核心模块库,SimIPF 将尽量使用这些模块来进行开发,以缩短开发周期.例如,目前我们直接使用了 mqueue 模块来模拟指令缓冲器(instruction buffer)的行为.但有些功能却不能使用核心模块进行模拟,需要使

用核心模块构建复合模块进行模拟,如指令发射逻辑.其实现如图 7 所示.

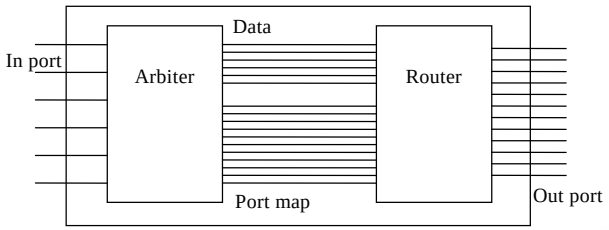

Fig.7 Dispersal logic simulation model

图 7 指令发射逻辑模拟模块

在安腾 2 中,指令缓冲器每个时钟周期可提供多达 6 条指令给指令发射单元.指令发射单元将根据不同的指令类型以及指令之间的组合(3 条指令 1 个组合)将指令发射到 11 个端口上去.在我们实现的指令发射逻辑模拟模块中,使用了如图 7 所示的 Arbiter 和 Router 两个核心模块.其中,Aibiter 模块负责根据安腾 2 的指令发射逻辑对指令进行排序,Router 模块负责指令的发射.对指令发射逻辑的实现,是在 Arbiter 模块的 userpoint (comparison_func)中完成的,在这个 userpoint 中,需要编写大量的代码来模拟安腾 2 复杂的指令发射逻辑.从该复合模块的输入端口输入的数据直接传给 Arbiter 模块的输入端 Inport,经它处理后会产生两种类型的输出:(1) 排序后的数据 data;(2) 输出端口对应的输入端口号 Portmap;输出(1)直接输入给 Router 模块的数据输入,输出(2)输入给 Router 的输出信息选择端口,如图 7 中的 Port map 连接所示.

选择安腾 2 作为模拟器开发案例,主要是由于安腾 2 具有多种配置,如单核、4 核等,高速缓存的大小也有多种配置.这非常有利于使用一种模拟器开发技术开发具有灵活配置的模拟器,特别是在单核和多核处理器模拟器后,使用各种配置的实际硬件机器来进行模拟器验证,得出各种配置模拟器的模拟精度.如果所得的多个模拟精度趋近于一个常数,则证明该模拟技术是较为理想的,使用该技术对拟设计的体系结构的性能进行预测是可信的;如果多个模拟精度的值较为发散,则需要进一步研究,从而提高使用该技术的模拟器的预测精度.

7 研究展望

随着时间的推移,人们对计算能力不断提出新的需求,研究人员和产品设计人员根据这些需求不断研究和设计新的处理器或计算系统体系结构.而在体系结构研究与设计中,模拟器是不可缺少的工具,且耗费了大量的人力、物力资源和时间.因此,体系结构软件模拟技术在今后相当长的时期内都将是十分重要的研究内容.

对模拟器的构造还很困难,需要研究新的模拟器开发技术,以减少模拟器建造的难度并缩短开发周期.目前,国际上这方面的研究已经开始,但还不是很多.LSE就是解决这一问题的成功典范,但它还只是解决了模拟器领域的第一个问题,其模拟运行的速度还有待进一步提高.另外,Asim,NetSim[51]使用面向对象技术开发了便于构造模拟器的库,库中提供了与硬件对应的模块.归纳起来,这些技术都力图提供与硬件相对应的软件组件模块,但LSE做得更好一些,它在并行化和减少面向对象带来的额外开销方面做了许多突破性的工作.因此,LSE可以被称作真正意义上的并行结构化模拟器开发方法.使用LSE的思想和技术去解决其速度方面的问题或开发新的模拟器开发工具或框架,是今后值得大力研究的方向之一.另外,开发针对某个特别体系结构的处理器模拟器,不是体系结构模拟技术领域应该研究的重点内容,其重点应该放在开发一个方便的、具有开放式架构的、提供了丰富而灵活的库模块的模拟器开发环境上.

在体系结构研究与设计领域,经过几十年的发展,出现了基于“数据驱动”的静态模型分析、分析模型、模拟技术等多种辅助设计技术.模拟技术中又发展了跟踪驱动模拟和执行驱动模拟技术.这些技术的流行顺序为基于“数据驱动”的静态模型分析、分析模型、跟踪驱动模拟和执行驱动模拟.虽然目前在学术界最为流行的是执行驱动模拟技术,但这并不表示其他模拟技术没有存在的必要了.事实上,许多以前流行的技术目前在工业界还发挥着重要的作用.例如,Intel公司还在继续使用跟踪驱动模拟技术研究与开发其处理器产品.在计算系统或处

理器变得越来越复杂的今天,特别是多核处理器架构的出现,执行驱动的模拟技术很难在时钟周期级别并在有限的时间内进行有效的模拟.因此,我们应重新回过头去再研究分析模型、跟踪驱动模拟技术[50],并设法使它们能与执行驱动模拟技术结合起来开发更有效的模拟技术.

鉴于处理器体系结构向多核乃至众核方向发展,计算机系统向着多处理器架构方向发展,将来的模拟技术研究应重点放在多核处理器模拟和多处理器计算机系统模拟技术上.目前,处理器内部核的数量发展非常快,2007 年 2 月Intel公司宣布,它在一个研究项目中已经成功制造了 80 核的处理器[52],其他公司也在大力推进多核技术的发展,并向 512 核冲击.如何充分发挥多核的性能,需要使用模拟技术来评估和分析核间的交互行为.然而,使用时钟周期级别的模拟来分析预测如此众多核的处理器的性能,在近几年内是不可能的[53].因此,研究非时钟周期级别但又能准确预测分析多核体系结构设计的模拟技术,也是今后的主要研究方向之一.

由于目前评估和测试处理器性能或计算系统性能的标准性能测试程序指令数巨大,导致在使用软件模拟器分析、预测新的设计时会耗费极长的模拟执行时间.解决该问题的较好方法是使用统计采样技术选择部分指令模拟执行.SMARTS 是最成功的方法之一,但 SMARTS 的执行速度依然不能满足处理器体系结构设计新的需求.因此,研究新的统计采样方法,进一步提高模拟速度同时又不降低模拟精度,仍将是模拟技术研究的主要方向.另一方面,研究新的标准性能测试程序,减少其中的冗余指令,使其既能准确地评估处理器或计算系统的性能,又便于使用模拟器分析预测未来设计的性能,也应是一个值得研究的方向.

8 总 结

计算机体系结构软件模拟技术是现代处理器和计算机系统设计中的必备工具,该技术的水平将直接影响处理器和计算系统的设计质量乃至处理器的制造水平.无论过去、现在,还是将来,体系结构软件模拟技术都会是研究的热点.本文从回顾历史开始,全面综述了该技术的走势,详细而深入地分析和比较了目前该领域取得的研究成果,并在此基础上指出了将来应该着重研究的若干方向.同时,我们也正在以开发一个模拟器 SimIPF 为实例,研究先进的模拟器开发技术.在我国,还只有中国科学院计算技术研究所和复旦大学等少数几个单位在进行这方面的研究.为了提高我国计算机技术的核心竞争力,我们认为,应该有更多的研究人员投入到该项技术的研究中来.

References:

[1] Assn SI. International technology roadmap for semiconductors. ITRS, 1999. http://public.itrs.net

[2] Wenish T, Wunderlich R: Applying SMARTS to SPEC CPU2000. Technical Report, Computer Architecture Laboratory at Carnegie Mellon (CALCM), 2003.

[3] Mukherjee S, Adve. S. Performance simulation tools. IEEE Computer, 2002,35(2):38−39.

[4] Hennessy J, Patterson D. Computer Architecture: A Quantitative Approach. 3rd ed., Elsevier Science Pte Ltd., 2003. 46.

[5] Heidelberger P, Lavenberg S. Computer performance evaluation methodology. IEEE Trans. on Computers, 1984,c-33(12): 1195−1220.

[6] John L. Performance evaluation: Techniques, tools and benchmarks. 2002. http://lca.ece.utexas.edu/pubs/john_perfeval.pdf

[7] Lazowska E, Zahorjan J. Quantitative System Performance-Computer System Analysis Using Queueing Network Models. Prentice Hall, 1984. 127−151.

[8] Austin T, Larson E. SimpleScalar: An infrastructure for computer system modeling. IEEE Computer, 2002,35(2):56−67.

[9] The SimpleScalar Architectural Research Tool Set, Version 2.0. 1998. http://www.cs.wisc.edu/~mscalar/simplescalar.html

[10] Hughes C, Pai V. Rsim: Simulating shared-memory multiprocessors with ILP processors. IEEE Computer, 2002,35(2):40−49.

[11] Shriver E, Shakallah K. Ravel: Assigned-Delay compiled-code logic simulation. In: Proc. of the 1992 Int'l Conf. on Computer-Aided Design. Santa Clara: IEEE Computer Society Press, 1992. 364−368.

[12] Ye W, Vijaykrishnan N. The design and use of SimplePower: A cycle-accurate energy estimation tool. In: Proc. of the DAC. 2000. http://www.sigda.org/Archives/ProceedingArchives/Dac/Dac2000/papers/2000/dac00/pdffiles/21_1.pdf

[13] Brooks D, Tiwari V. Wattch: A framework for architecture-level power analysis and optimizations. In: Proc. of the ISCA. 2000. http://ieeexplore.ieee.org/iel5/6892/18551/00854380.pdf?arnumber=854380

[14] Skadron K, Stan M. Temperature-Aware microarchitecture: Modeling and implementation. ACM Trans. on Architecture and Code Optimization, 2004,1(1):94−125.

[15] Li Y, Hempstead M. Power and thermal effect of SRAM vs. latch-mux design styles and clock gating choices. In: Proc. of the ISLPED 2005. San Diego, 2005. http://www.cs.virginia.edu/~skadron/Papers/islped05_li.pdf

[16] Roland W, Thomas W. SAMRTS: Accelerating microarchitecture simulation via rigorous statistical sampling. In: Proc. of the 30th Annual Int'l Symp. on Computer Architecture (ISCA 2003). 2003. http://www.ece.cmu.edu/~simflex/publ/smarts2003isca.pdf

[17] Zhang FX, Zhang LB. Sim-Godson: A Godson processor simulator based on SimpleScalar. Chinese Journal of Computers, 2007, 30(1):68−73 (in Chinese with English abstract).

[18] Vachharajani M, Vachharajani N. The liberty structural specification language: A high-level modeling language for component reuse. In: Proc. of the Conf. Programming Language Design and Implementation. 2004. http://portal.acm.org/citation.cfm?id=996893.996865

[19] Vachharajani M, Vachharajani N. Microarchitecture exploration with liberty. In: Proc. of the Int'l Symp. Microarchitecture. 2002. http://ieeexplore.ieee.org/iel5/8385/26403/01176256.pdf?arnumber=1176256

[20] http://www.systemc.org/

[21] Coe P, Howell F. A hierarchical computer architecture design and simulation environment. ACM Trans. on Modeling Computer Simulation, 1998,8(4):431−446.

[22] Janneck J, Lee W. Taming heterogeneity—The ptolemy approach. In: Proc. of the ACM SIGPLAN 2001 Workshop on Languages, Compilers and Tools for Embedded Systems (LCTES). 2001. http://ieeexplore.ieee.org/iel5/5/26369/01173203.pdf?tp=&isnumber=&arnumber=1173203

[23] Vachharajani M, Vachharajani N. The liberty simulation environment: A deliberate approach to high-level system modeling. ACM Trans. on Computer Systems, 2006,24(3):211−249.

[24] Joshua K, Daniel C. Statistical simulation of multithreaded architectures. In: Proc. of the 13th IEEE Int'l Symp. on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2005). 2005. http://ieeexplore.ieee.org/iel5/10197/32540/01521119.pdf?tp=&isnumber=&arnumber=1521119

[25] Magnusson P, Christensson M. Simics: A full system simulation platform. IEEE Computer, 2002,35(2):211−249.

[26] Stephen G, John H. The accuracy of trace-driven simulations of multiprocessors. Technical Report, CSL-TR-92-546, Stanford University, 1992.

[27] Gao X, Zhang FX. SimOS-Goodson: A Goodson-processor based multi-core full-system simulator. Journal of Software,2007,18(4): 1047−1055 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/18/1047.htm

[28] Martin M, Sorin D. Multifacet's general execution-driven multiprocessor simulator (gems) tool set. Computer Architecture News, 2005,22(4).

[29] Babaoglu O, Ferrari D. Two-Level replacement decisions in paging stores. IEEE Trans. on Computers, 1983,C-32(12):1151−1159.

[30] Belady L. A study of replacement algorithms for a virtual storage computer. IBM System Journal, 1966,5(2):78−101.

[31] Dwarkadas S, Jump J. Execution-Driven simulation of multiprocessors: Address and timing analysis. ACM Trans. on Modeling and Computer Simulation, 1994,4(4):314−338.

[32] http://www.simplescalar.com/

[33] Todd A, Dan E. SimpleScalar tutorial (for release 4.0). In: Proc. of the 34th Annual Int'l Symp. on Microarchitecture (MICRO-34). Austin, 2002. http://www.simplescalar.com/tutorial.html

[34] Emer J, Ahuja P. Asim: A performance model framework. IEEE Computer, 2002,35(2):68−76.

[35] Rosenblum M, Bugnion E. Using the SimOS machine simulator to study complex computer systems. ACM Trans. on Modeling and Computer Simulation, 1997,7(1):78−103.

[36] Witchel E, Rosenblum M. Embra: Fast and flexible machine simulation. In: Proc. of the ACM SIGMETRICS'96: Conf. on Measurement and Modeling of Computer Systems. Philadelphia, 1996. http://portal.acm.org/citation.cfm?id=233013.233025&coll=portal&dl=ACM&type=series&idx=SERIES328&part=series&WantType=Proceedings&title=SIGMETRICS&CFID=15151515&CFTOKEN=6184618

[37] Cmelik F, Keppel D. Shade: A fast instruction-set simulator for execution profiling. In: Proc. of the ACM SIGMETRICS'94. 1994. 128−137. http://portal.acm.org/citation.cfm?doid=183018.183032

[38] Rosenblum M, Stephen A. Complete computer system simulation: The SimOS approach. IEEE Parallel and Distributed Technology, 1995,3(4):34−43.

[39] Stephen A. Using complete machine simulation to understand computer system behavior [Ph.D. Thesis]. Stanford University, 1998.

[40] Wilson R, French R. The SUIF compiler system: A parallelizing and optimizing research compiler. Technical Report, CSL-TR-94-620, Stanford University, 1994.

[41] Liberty Research Group. Liberty simulation environment core module library reference manual. 2003. http://liberty.princeton.edu/

[42] Yi J, Kodakara S. Characterizing and comparing prevailing simulation techniques. In: Proc. of the 11th int'l Symp. on High-Performance Computer Architecture (HPCA-11 2005). 2005. http://ieeexplore.ieee.org/iel5/9519/30167/01385950.pdf?tp=&isnumber=&arnumber=1385950

[43] Yi J. A statically-rigorous approach for improving simulation methodology. In: Proc. of the Int'l Symp. on High Performance Computer Architecture. 2003. http://ieeexplore.ieee.org/iel5/8433/26557/01183546.pdf

[44] Biesbrouck M, Sherwood T. A co-phase matrix to guide simultaneous multithreading simulations. In: Proc. of the IEEE Int'l Symp. on Performance Analysis of System and Software. 2004.3. http://www.cs.ucsd.edu/~calder/papers/ISPASS-04-CoPhaseMatrix.pdf

[45] Perelman E, Hamely G. Using SimPoint for accurate and efficient simulation. In: Proc. of the ACM SIGMETRICS 2003. San Diego, 2003. http://www.cs.ucsd.edu/~calder/papers/SIGMETRICS-03-SimPoint.pdf

[46] KleinOsowski A, Ljia D. MinneSPEC: A new SPEC benchmark workload for simulation-based computer architecture research.2002. http://www.arctic.umn.edu/~Liljia/papers/minnespec-cal-v2.pdf

[47] Sherwood T, Perelman E. Automatically characterizing large scale program behavior. In: Proc. of the Int'l Conf. on Architecture Support for Programming Languages and Operating Systems. 2002. http://cseweb.ucsd.edu/users/calder/papers/ASPLOS-02-SimPoint.pdf

[48] Conte T, Hirsch M. Reducing state loss for effective trace sampling of superscalar processors. In: Proc. of the Int'l Conf. on Computer Design. 1996. http://www.tinker.ncsu.edu/symposia/iccd96.pdf

[49] Citron D. MisSpeculation: Partial and misleading use of SPEC CPU2000 in computer architectural conferences. In: Proc. of the ISCA 2003. 2003. http://ieeexplore.ieee.org/iel5/8578/27165/01206988.pdf?arnumber=1206988

[50] Yi J, Eechout L. The future of simulation: A fields of dreams? IEEE Computer, 2006,39(11):22−29.

[51] Zier D, Nelsen J. NetSim: An object-oriented architectural simulator suite. In: Proc. of the 2005 Int'l Conf. on Computer Design (CDES 2005). 2005. http://web.engr.oregonstate.edu/~benl/Publications/Conferences/CDES05.pdf

[52] Krazit T. Intel shows off 80-core processor. 2007. http://news.com.com/2100-1006_3-6158181.html

[53] http://www.arctic.umn.edu/~jjyi/MoBS/2006/advance_program.html

附中文参考文献:

[17] 张福新,章隆兵.基于 SimpleScalar 的龙芯 CPU 模拟器 Sim-Godson.计算机学报,2007,30(1):68−73.

[27] 高翔,张福新.基于龙芯 CPU 的多核全系统模拟器 SimOS-Goodson.软件学报,2007,18(4):1047−1055. http://www.jos.org.cn/1000-9825/18/1047.htm

喻之斌(1972－),男,湖北宜昌人,博士生,讲师,CCF 会员,主要研究领域为计算机系统结构模拟.

金 海(1966－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为网格计算,服务计算,计算系统虚拟化.

邹南海(1975－),男,工程师,主要研究领域为操作系统性能优化.