计算机科学 2007Vol.34№.10

多核处理器体系结构软件仿真技术：研究综述

喻之斌　金　海
（华中科技大学计算机学院　武汉 430074）

摘　要　由于单核处理器的处理能力已经接近极限，很难再有提高，人们将目光投向了多核处理器体系结构。在处理器体系结构的设计中，体系结构软件仿真技术是最重要的一个方面。本文首先介绍处理器体系结构仿真技术的概念、分类、目的和意义，然后讨论多核处理器体系结构仿真技术的现状和面临的问题；分析了多核处理器软件仿真技术的复杂性；比较和分析了当前主流技术的优缺点。由于多核处理器体系结构的研究处于初期阶段，因此多核处理器体系结构仿真领域面临着诸多挑战和机遇。本文最后指出了多核处理器体系结构软件仿真技术今后的研究方向。

关键词　多核处理器，体系结构，仿真技术

Multi-core Architecture Simulation: A Survey

YU Zhi-Bin　JIN Hai
(School of Computer Science, Huazhong University of Science and Technology, Wuhan 430074)

Abstract　Since it is very difficult to promote the throughput of a single-core processor nowadays, the multi-core processor architecture is paid much attention to by computer architects. In processor architecture design, the most important activity is to use software to simulate the processor architecture. This paper firstly introduces the concepts, classification, purpose and advantages of architecture simulation. Then it tries to analyze the complexity of multi-core architecture simulation. After that, the advantages and disadvantages of some major popular simulation techniques are described in this paper. As the research of multi-core processor is still in its infant stage, there are a lot of challenges in multi-core simulation area. Therefore, the future promising research directions are also discussed in the last part of the paper.

Keywords　Multi-core processor, Architecture, Simulation technology

近年来，单核处理器的处理能力已经接近极限，摩尔定律似乎已经不再成立。由于能耗、散热特别是空间尺寸的限制，处理器设计师们很难设计出处理能力更高而又成本合理的单核处理器。目前，无论是工业界还是学术界都将目光转向了多核处理器体系结构[1]。多核处理器是指一个 CPU 芯片内含有两个或两个以上的“执行内核”。无论是单核还是多核处理器，在进行体系结构设计时，软件仿真技术是必不可少的。而且，多核处理器体系结构的软件仿真技术比起单核处理器体系结构来要面临更大的挑战。

本文先介绍处理器体系结构软件仿真技术的基本概念、分类、目的和意义，然后讨论处理器体系结构软件仿真技术的现状、面临的主要问题；分析多核处理器体系结构软件仿真的复杂性；比较和分析主要的仿真技术并指出今后的研究方向。

1　处理器体系结构软件仿真基本概念

1.1　定义

处理器体系结构软件仿真是指使用软件工具来模拟处理器的运行状态和行为。其实，除了处理器的运行状态和行为可以进行软件模拟外，一切数字设备都可以进行软件模拟。如图 1 所示[2]。

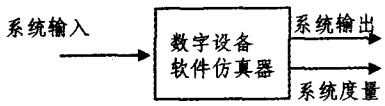

图 1　软件仿真示意图

根据需要，处理器体系结构软件仿真器可以模拟所有的处理器内部和与之关联的外部器件的运行。在实践中，一般将处理器体系结构仿真分为不同的级别，便于研究不同性质的问题，如时钟周期级的仿真、流水线级的仿真等。

1.2　体系结构软件仿真器的分类

体系结构软件仿真器的分类方法很多，如根据处理器的个数分类，分单处理器仿真器、多处理器仿真器和多核处理器仿真器；根据仿真执行的方式，可分为串行执行仿真器和并行仿真器等。

文[29]将计算机体系结构软件仿真器分为单处理器性能仿真器、完整仿真器、单处理器能耗仿真器、多处理器性能仿真器和模块化仿真器。其中完整仿真器是对整个计算机系统的完整仿真，包括对处理器系统、操作系统、内存访问、高速缓存、存储系统以及网络系统的仿真等。SIMICS 是这类仿真器的优秀代表[34]。模块化仿真器是使用软件中的一个函数来代表一个计算机部件。和函数的优点一样，这类仿真器可以容易地实现仿真程序的重用，并可以方便地开发并行仿真器。它的主要代表是普林斯顿大学开发的 LSE (Liberty Simulation Environment)[30~33]。

本文给出另一种典型的分类，如图 2 所示。

在图 2 所示的分类方法中，体系结构软件仿真器可以分为两大类，一类是体系结构仿真器，另外一类是微体系结构仿真器。体系结构仿真器仿真程序员可见的处理器体系结构的功能和行为，如处理器的通用寄存器、数据寄存器、地址寄存

喻之斌　博士研究生、讲师，主要从事并行编译技术、多核处理器技术研究；金　海　教授、博士生导师，主要从事计算机系统结构、网格计算和普适计算机研究。

器等的功能和行为。微体系结构仿真器对处理器系统（通常称为微结构）的内部状态和行为（如控制寄存器和状态寄存器）进行建模，并且仿真处理器系统内部部件与时间有关的特性，如每条指令需要多少时钟周期（CPI）等。下面对主要的仿真类型进行详细介绍。

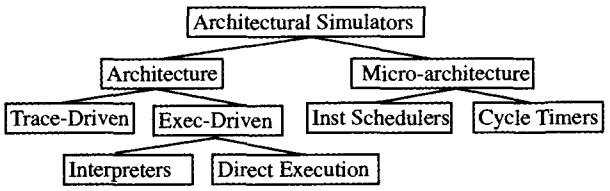

图 2　体系结构软件仿真器的一种分类

1.2.1　体系结构仿真

处理器体系结构软件仿真是用软件方法模拟 CPU 通用寄存器、数据寄存器等部件的功能和性能，这些部件的状态和数据可用汇编语言进行访问。体系结构仿真也被称为功能仿真。该类仿真器又通常分为如图 2 所示的两类：跟踪驱动仿真器（Trace-Driven）和执行驱动（Exec-Driven）仿真器。

对于执行驱动仿真，在体系结构研究者中有两种观点：一种观点认为，执行驱动仿真使用程序的可执行部分作为仿真器的输入。这种类型的仿真器使用实际可执行的指令输入来进行仿真，并不使用程序的跟踪状态进行仿真。因此，仿真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 3（a）所示。在学术界和工业界广泛使用的 SimpleScalar 就是执行驱动仿真的典型代表。另一种观点认为，执行驱动的仿真器依赖于在宿主机上实际运行一部分代码，在软件仿真器上运行另一部分代码。因此，这类执行驱动的仿真器不仿真执行一个运用程序的每一条指令。只有对问题研究有用的指令才会被仿真执行，其他的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仿真的处理器体系结构的指令集一样时才有意义。这种类型的仿真可以分为两个阶段：第一个阶段称为预处理阶段，主要是修改应用程序，在感兴趣的位置插入调用仿真子程序的语句。例如，对于一个内存管理系统仿真器，如果用户只对内存访问感兴趣，就只在应用程序中将进行内存访问时调用仿真程序，使用仿真程序来模拟内存访问行为。第二个阶段为仿真阶段，即在宿主机和仿真器上交替执行应用程序，如图 3（b）所示。Tango，Proteus 和 FAST 都是这类仿真器的代表[3,5]。

与执行驱动仿真不同，跟踪驱动仿真将每条指令顺序执行所产生的所有信息作为仿真器的输入，从而来模拟某种体系结构处理器的功能和性能[3]。跟踪驱动仿真技术采用了分治法（divide-and-conquer）策略。假设要仿真执行的程序负载为 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仿真。称跟踪信息生成的环境为 G，则当在 G 中执行 w 所发生的所有事件将被记录下来，这些记录被称为跟踪信息。然后，仿真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，如图 4 所示[4]。跟踪驱动仿真比较简单且容易理解；跟踪驱动仿真器也容易进行调试。实验数据可以重现，因为只要程序负载一样，在不同的仿真执行中，输入给仿真器的数据不会变化。但执行驱动的仿真器存在以下两个方面的问题：

（1）仿真时间可能会非常长，用于存储仿真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被满足。

（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。

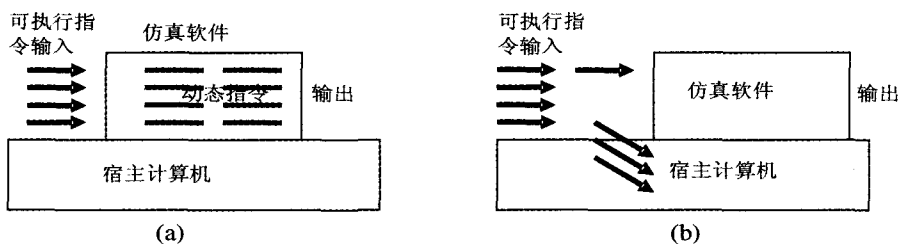

图 3　两种执行驱动仿真

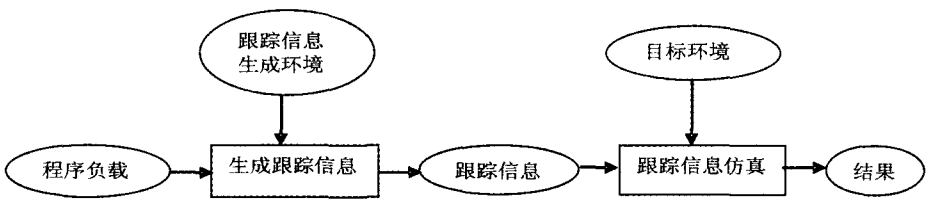

图 4　跟踪驱动仿真过程

1.2.2　微体系结构仿真

微体系结构仿真主要对处理器系统的内部结构和行为进行仿真，通常对时间有关的处理器内部特性进行模拟和分析。微体系结构仿真过去也被称为性能仿真。如图 2 所示，微体系结构仿真器可以分为两类：一类为基于约束的指令调度仿真器，另一类为 CPU 时钟周期级的仿真器。基于约束的指令调度仿真器根据当前状态下可以获得的系统资源对指令进行调度，并形成指令执行路线图。在这种类型的仿真器中，每次仿真执行只能有一条指令被处理，并且指令是按顺序执行的。这类仿真器的优点是对其修改（如指令调度策略）比较容易，缺点是仿真信息不够详细。时钟周期级的仿真器跟踪记录每一个时钟周期的详细系统状态，这样仿真器的状态就是所仿

真的微系统的状态。从这类仿真器所模拟的内容来看，它适合于进行详细微体系结构仿真，并且该类仿真器的仿真结果是最为可靠的。

2 处理器体系结构软件仿真的意义

在处理器设计和计算机体系结构研究中，最重要的工具就是软件仿真器。使用软件仿真器可以使处理器架构师方便、快捷地评估不同结构或资源配置下的处理器的功能和性能，从而节省处理器的研发成本和上市时间。另外，使用软件仿真器对处理器体系结构进行评估比制造一个处理器样品来进行评估具有更大的灵活性，因为软件仿真器可以精确地评估一款新的处理器设计而不用去做电路级的设计验证。制造一款新的处理器至少要花几个月甚至几年的时间，其费用是非常昂贵的。如果这款新的处理器制造出来仅仅是为了设计的验证，若有问题，得进行修改，重新制造，再进行验证，这种方式所花的成本和时间是人们不能接受的。因此，没有软件仿真器，处理器的设计和研究不仅昂贵得让人们无法接受，而且通常会产生非常糟糕的设计结果[6~9]。目前，处理器体系结构仿真已经是处理器设计和制造中不可缺少的一个组成部分了。如图 5 所示。

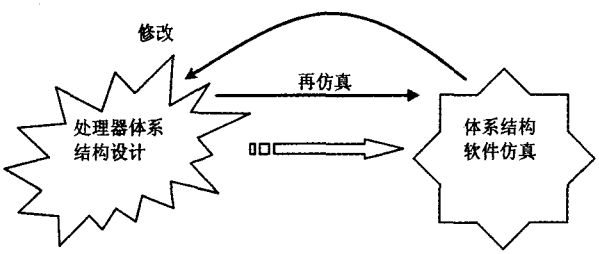

图 5 处理器设计过程

总的来说，处理器体系结构仿真具有以下几个方面的意义：

1）缩短处理器的设计和开发周期，节约开发成本。

2）帮助设计人员找出处理器中具有更多设计空间的部分，以便进一步研究或设计。

3）可在没有硬件实物的条件下对新的处理器体系结构进行验证。

4）有助于对处理器系统进行测量。

5）帮助学术界对处理器结构进行学术研究。

6）有利于教育机构对计算机系统结构进行形象生动的教学。

3 主要的体系结构软件仿真器

由于软件仿真器在处理器设计中如此地重要，因此世界各国都在处理器体系结构软件仿真技术和仿真器的构造方面进行了研究。实际上，使用软件仿真器来对处理器结构或计算机系统进行研究已经有很长的历史了。目前，已有了上百种处理器体系结构仿真器，表 1 中列举了目前较为流行的主要体系结构仿真器。文[10]和[11]中列出了更为详尽的资料。

在这些软件仿真器中，可用于仿真多处理器体系结构的仿真器有：Augmint、SIMCA、SimOS、SimPoint、SimFlex 和 GMES。目前，可以用于多核处理器的最优秀的软件仿真器是 SimPoint 和 SMARTS。SimpleScalar 是学术界和工业界最为流行的体系结构仿真器，其他许多仿真器是在它的基础上开发的，例如 TurboSmart，SimPoint 和 SIMCA。SimpleScalar 提供了一整套模拟工具，如编译器、汇编器、连接器、仿真和可视化工具等[12]，构成了基于执行驱动的体系结构仿真的基础。

表 1 主要的体系结构仿真器

仿真器名称	类型	拟模拟的体系结构或处理器平台	研制单位
SimpleScalar	执行驱动	Alpha，PowerPC，x86，SPARC，RS6000，PA-RISC	Wisconsin-Madison
GEMS	执行驱动	x86，SPARC	Wisconsin-Madison
SimOS	完整仿真	MIPS4k，MIPS10k，Alpha	Stanford U
Simics	完整仿真	x86，SPARC	Wisconsin-Madison
bochs	完整仿真	x86	SourceForge. net
ML-RSIM	执行驱动	SPARC V-8，Pentium III	Notre Dame U Utah U
Dinero IV	跟踪驱动	x86，Alpha，SGI，RS6000，SPARC	Wisconsin-Madison
WWT-II	事件驱动	SPARC	Wisconsin-Madison
RSIM	执行驱动	SPARC，SGI，Convex Exemplar	Illinois U
SIMCA	执行驱动	SPARC，x86	Minnesota U
HASE	执行驱动	x86，MC88000	Edinburgh U
Shade		SPARC	SUN Microsystems
Augmint	执行驱动	x86	Illinois U
TurboSmart	微结构仿真	x86	Carnegie Mellon U
SimFlex	完整仿真	x86	Carnegie Mellon U
SimPoint	执行驱动	x86，SPARC	California U，San Diego

4 体系结构软件仿真器的现状及面临的问题

软件仿真器在计算机系统的设计、研究和发展中起着不可替代的作用。据统计，在设计一款新的体系结构的计算机时，超过 60%的人力和计算机资源都用于设计方案的仿真和验证上，而这其中的 80%又用在使用软件进行的功能仿真上[13]。由此可见，体系结构软件仿真在过去的时间里得到了成功的运用，并成为了体系结构设计过程中不可缺少的一个环节。在现代计算机体系结构研究中，理解处理器运行一个应用程序时在时钟周期级别的行为是至关重要的。而要理解时钟周期级别的处理器行为，通常只能使用软件对处理器在时钟周期级别的行为细节进行仿真[6,7]。另外，由于研究问题的不同，人们也使用不同级别的软件仿真，例如指令级别的仿真、流水线级别的仿真等。

很显然，体系结构仿真软件的仿真执行速度比所仿真的实际硬件设备的执行速度要慢许多。目前，世界上最快的时钟周期级别的微体系结构性能仿真器一般比其所仿真的硬件设备慢 5 个数量级以上。例如在 Pentium 4 2GHz 的机器上运行仿真器，其所达到的最高仿真速度也只有 0.5 个 MIPS。对于更详细的仿真和寄存器数据迁移级别的仿真，其仿真速度比其对应的硬件运行速度更是要慢 6 个数量级以上。即使是目前在加快仿真器速度方面研究得最成功的卡内基-梅隆大学，其仿真器也仅能在现有基础上将仿真速度提高 35～60 倍[14]。

另外，要求仿真器仿真的信息越详细，仿真器本身的设计和开发环节就越多，从而加大仿真器本身的出错机率。目前，各类仿真器的仿真错误率也是一个非常严重的问题，只有少数几个仿真器如 SMART，SimPoint 将出错率控制在一个比较低的范围内。

目前，人们为了追求更高的计算性能，处理器设计的复杂程度和相应硬件的复杂程度呈指数级增长，而为了验证和评估这类处理器设计，其软件仿真的过程将会更加复杂。尤其是在多核的体系结构下，其仿真的复杂度更是复杂得难以想象。总之，目前体系结构详细仿真的仿真速度非常慢，仿真错误也是一个不容忽视的问题。实际上，在计算机体系结构软件仿真领域，仿真速度和仿真结果的精确性是一对矛盾。仿真速度提高了，仿真结果的出错率就会加大。进行足够详细的仿真，可以保证仿真结果的精确性，但仿真速度却会大大降低，直到难以忍受的程度。因此，在多核体系结构这样的复杂环境下，如何更进一步提高软件仿真的速度而又不牺牲精度是体系结构仿真器所面临的重大挑战。

5 多核体系结构下的软件仿真复杂性分析

5.1 多核体系结构处理器的基本概念

多核处理器体系结构是在一个芯片或一个处理器封装内含有两个或两个以上的“可执行内核”或计算引擎，并且可以实现真正意义上的线程并行处理[16]。这些“可执行内核”共享系统内的某些资源（如内存或者高速缓存）[15]。图 6 示意了两种可能的多核处理器体系结构[15]。

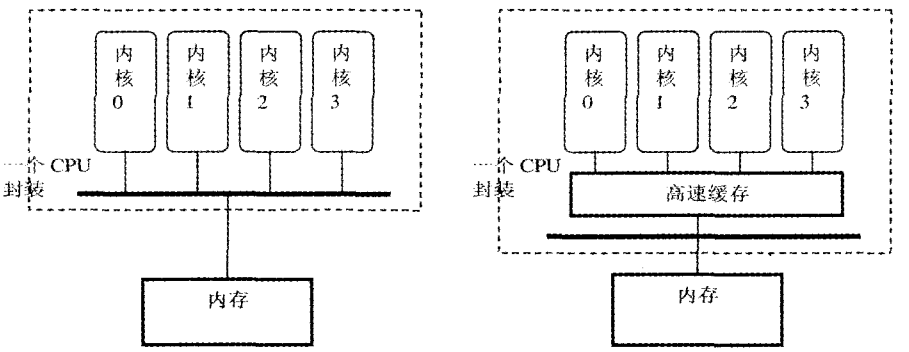

图 6 两种可能的多核体系结构

5.2 多核体系结构下软件仿真复杂性分析

多核体系结构处理器本身的复杂性决定了利用软件技术对其进行仿真的复杂性。从图 6 中可以看出，多核处理器中的多个核需要与其他的核共享一些资源。当多个核并行运行的时候，资源共享势必引起资源的竞争，资源的竞争会大大影响每个核的性能，从而大大影响整个多核处理器的性能。线程内工作负载的变化和相互影响、资源的竞争都将导致处理器体系结构的设计选择和程序活动的数量急剧增加，而这些都需要使用软件仿真的办法来进行详细的研究。无疑，对多核处理器体系结构的软件仿真是复杂的。从上面的分析可以看出，这种复杂性主要来源于两个方面：(1) 选择哪些资源在多个线程中共享；(2)理解多线程在这些共享资源基础上的交互行为[6]。

并发多线程之间的资源共享策略可处在共享所有的处理器资源到只共享非常少的资源如低级高速缓存和总线这个范围之间。共享资源的确定需要使用软件仿真技术进行详细的评估，并在评估结果中进行折衷选择。在当前的仿真技术条件下，这是一项非常繁琐和耗时的工作。

精确刻画一个多线程系统的性能需要准确理解多线程在它们共享的资源基础上的交互行为。当处理器中的执行内核的数量增加时，用于测量处理器性能的测试程序（benchmark）的运行组合数量会呈指数增长。另外，根据文[17]的研究，每个应用程序都是由一些有唯一行为特性的程序阶段组成的。处理器性能的刻画实际就是发现这些程序阶段或这些程序阶段组合的行为特性。在多核处理器的环境下，程序阶段的总数量就是各线程内各程序阶段的组合数量，而这个组合数量也会随着线程数量（“执行内核”数量）的增加而呈指数级增长。

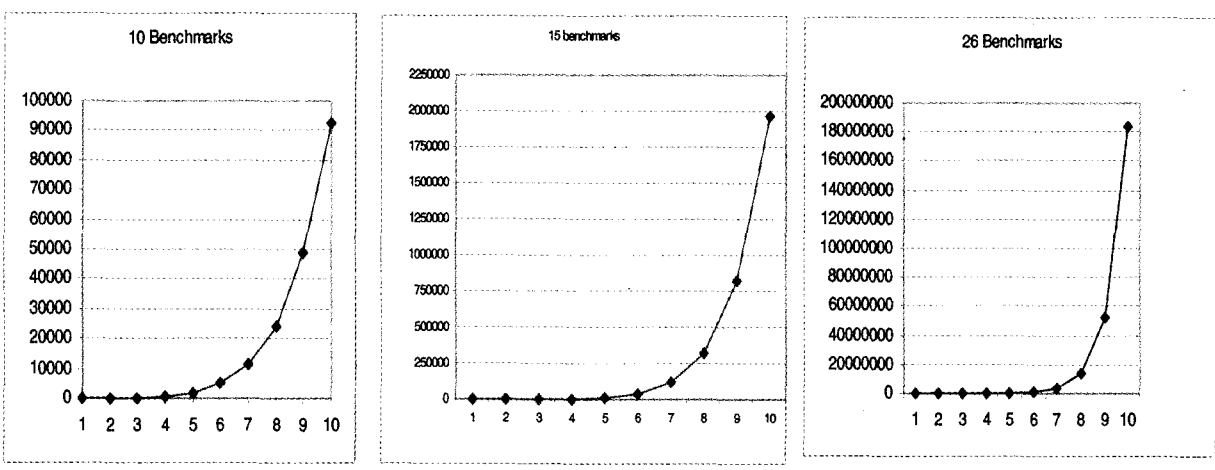

图 7 测试次数的变化情况

共享资源的选择和程序阶段及其组合数量在多核体系结构下的增加使得软件仿真器执行测试程序(benchmark)的次数非常庞大。例如 SPEC CPU2000[18]性能测试包中有 26 个测试程序,其中 12 个用于测试整数运算性能,14 个用于测试浮点运算性能。刻画单核处理器的 SPEC 性能,只需将 26 个测试程序全部执行一遍,获得性能数据即可,即运行 26 次测试程序。对于双核处理器,如果两个核是同构的,为刻画整个双核处理器的性能,需要执行测试程序 351 次;如果两个核是异构的,则需要执行测试程序 676 次。图 7 展示了 10 个、15 个和 26 个测试程序在从 1 个核到 10 个核的情况下需要执行的次数情况。

从图中可以看出,无论测试程序的多少,在"可执行内核"增加的情况下,为刻画整个多核处理器的性能,需要执行测试程序的次数会呈指数级增加,从而使软件仿真的难度也急剧加大。对于完整测试处理器的性能,需要使用 SPEC CPU2000 中的所有测试程序,即 26 个测试程序。假设要仿真和评估含 10 个可执行内核的处理器的性能,由图 7 可知,其测试次数在 1.8 亿以上。准确地说,要测试 183579396 次。SEPC CPU2000 中的 26 个测试程序,每个测试程序的指令数都在 10 亿以上[19]。又假设某软件仿真器的频率为 100kHz,平均每时钟周期 3 条指令,则要完成 26 个测试程序在 10 核处理器上的仿真所需的时间为 19404 个 CPU 年,这是没有意义的。

6 主要仿真技术

对现代计算机体系结构的研究,一般采用 SPEC CPU2000 中的测试程序使用软件仿真的方法进行。SPEC CPU2000 中的每个测试程序都有测试输入参数集合,其中每个测试程序的最大的测试输入参数集合称为该测试程序的参考输入参数。从上述的分析中可以看出,进行完整仿真所需的时间是极长的,人类不可能完成这样的仿真。因此,目前处理器体系结构软件仿真领域的主要技术也大都集中在如何减少仿真时间上。主要的技术有 3 种:(1)减少仿真输入参数;(2)只仿真执行 SPEC CPU2000 测试程序的某些片断来代表仿真整个测试程序的结果;(3)在仿真中加入统计采样技术。

6.1 减少仿真输入参数

减少仿真输入参数技术的基本思想是用某种方式减少测试程序的参考输入参数,从而减少仿真时间。这种技术的期望是缩减的输入参数集合能像参考输入参数集合一样刻画处理器某个方面的性能,但减少了仿真时间。该方法的主要优点是用缩减的输入参数集合详细仿真完整测试程序的行为,包括程序的初始化、程序执行主体和程序的清除阶段。它的主要缺点是仿真的结果可能和使用参考输入参数的仿真结果相差很远。另外,选择和制定缩减的输入参数集合是一项非常繁琐和耗时的工作。这项技术的典型代表是明尼苏达大学的 SIMCA 仿真器[22],开发了三个减少了的输入参数集合,称为 MinneSPEC small,medium 和 large[21]。

6.2 仿真执行程序片段

仿真执行程序片段的主要思想是仿真器只执行 SPEC CPU2000 中测试程序一定数目的指令而不执行所有的指令,使用的输入参数为参考输入参数。这种技术期望任意选取一定数目的指令在仿真器上执行,从而代表整个测试程序的行为。根据选取执行指令方式的不同,该技术主要三种形式,称为 Run Z,FF X + Run Z,FF X + WU Y + Run Z。Run Z 是仿真执行一个测试程序最初的 Z 百万条指令代表仿真执行整个测试程序的行为,仿真时间由 Z 的值决定。使用最初的 Z 百万条指令代表整个测试程序的行为显然有些不合理,于是出现了 FF X + Run Z。这种方式是先快速向前仿真执行 X 百万条指令,再详细仿真执行 Z 百万条指令,依然使用这 Z 百万条指令的仿真结果代表整个测试程序的行为。所谓快速向前仿真执行是指仿真器在仿真时忽略许多细节状态的产生和记录,如处理器和内存的状态,从而可以快速向前执行。但它根本不能提供研究人员所需要的数据。该方式是对 Run Z 的改进,但它又引来了另一个问题。在快速向前执行 X 百万条指令后,仿真器中还没有处理器和内存状态的记录,然而却要开始详细仿真,这并不符合实际情况。因此,人们考虑在快速向前仿真执行 X 百万条指令以后,仿真执行 Y 百万条指令作为仿真"预热",开始产生和记录某些细节状态,然后再详细仿真执行 Z 百万条指令。仿真执行测试程序的部分指令最大的优点是减少了仿真执行的时间。然而,它最大的缺点是仿真结果往往不能代表执行完整测试程序的仿真结果。

6.3 统计采样技术

从第 5 节的分析中可以看出,完整仿真执行 SPEC CPU2000 中的测试程序所需的时间是极长的,甚至是不可能的。上面介绍的两种方法又有难以克服的缺陷,统计采样的方法便被运用到了仿真技术中。根据统计采样理论,通过采样方式选择的部分指令,可以较好地代表整个测试程序的行为。所以,这种仿真技术一般分两步:(1)使用采样理论选择指令片段;(2)对所选择的指令进行详细仿真执行。根据采样方式的不同,统计采样仿真技术又可分为三类: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仿真执行点,用仿真执行点的结果来代表整个测试程序的行为。这个技术的典型代表是加州大学伯克利分校的 SimPoint[23]。SimPoint 仿真分三步进行。首先,分析测试程序,找出候选的仿真点;然后使用基于统计的聚类办法(clustering)选择一组仿真点进行仿真,代表整个测试程序的仿真。最后,将所选的每个仿真点的仿真结果进行加权,得出最后的仿真结果。所选的仿真点的个数和每个所选仿真点的长度决定仿真时间。与代表性采样仿真不同,周期性采样仿真是在测试程序动态指令中以固定间隔周期性地选取部分指令进行仿真。显然,可以通过控制采样频率和每个样本的长度来控制仿真时间。这类技术的典型代表是卡内基梅隆大学的 SMARTS[14]。随机采样是先从测试程序的动态指令中随机抽取 N 段指令进行仿真,然后将这些仿真结果以某种方式进行合并,形成最后的仿真结果。这种方法的代表人物是 M. C. Thomas 等人[24]。需要指出的是,统计采样技术是基于仿真执行程序片段技术之上的。例如 SMARTS 中就有 FF X 和 WU Y 部分;M. C. Thomas 等人曾建议为了减少随机采样仿真的错误,增加 WU Y 中的 Y 值。

在过去的 10 年里,计算机体系结构软件仿真技术大部分集中在仿真执行程序片段技术和减少仿真输入参数技术上。根据 J. Y. Joshua[20]等人的统计,FF X + Run Z 占 27.3%,Run Z 占 23.1%,减少仿真输入参数占 18.5%,这两种技术占了将近 70%的比例。另外,完全仿真技术占了 17.8%。统计采样仿真技术虽然很早被提出,但在最近才得到重视。特别是多核处理器仿真的需要,统计采样仿真技术将成为主流的体系结构软件仿真技术。

当前，在多核处理器体系结构软件仿真领域，还有一些正在发展的技术。由于多核处理器架构下，各个核之间本身存在并行性，可以发展并行仿真技术来加快仿真速度。普林斯顿大学和科罗拉多大学[25,26]已经开始了这方面的工作。A. Z. David 等人将面向对象技术引入体系结构软件仿真技术中，以加快处理器体系结构软件仿真时间[27]。H. Greg 等学者使用机器学习的方法来指导体系结构的软件仿真[19]。佐治亚理工大学和英特尔公司联合，研究软件仿真和 FPGA 仿真联合使用的办法来加快体系结构的仿真[28]。

总结与展望 随着单核处理器性能极限的接近，多核处理器体系结构势必成为高性能处理器设计的首选和主流。多核处理器体系结构的出现将引起计算机工业的一场革命，它将引起程序设计语言，编译器，操作系统等一系列技术的变革。而在多核处理器体系结构的研究中，软件仿真技术是必不可少的。在单核处理器时代，体系结构软件仿真器发挥了不可磨灭的作用。在正在来临的多核处理器时代，体系结构软件仿真器将发挥更为重要的作用。在学术界，美国的卡内基梅隆大学、加州大学伯克利分校、普林斯顿大学、斯坦福大学、明尼苏达大学、佐治亚理工学院等众多著名大学纷纷开始了多核处理器软件仿真技术和仿真器的研究与开发工作。在工业界，因特尔等处理器制造巨头也展开了试验和研究。多核处理器体系结构下有更多的问题需要使用软件仿真的方法来进行研究，如能耗问题，性能问题，容错问题等。本文从处理器体系结构软件仿真的基本概念，分类，已经存在的各种软件仿真器到处理器体系结构软件仿真技术的现状及所面临的主要问题进行了概述。分析了多核体系结构下软件仿真技术面临的复杂性。比较了目前主要的仿真技术，并指出了可能的，值得研究的体系结构软件仿真技术的研究方向。

就如何在多核处理器体系结构下减少软件仿真的时间，同时又不降低仿真结果的真实程度，依然是一个极具挑战的问题。使用 SPEC CPU2000 中的测试程序进行详细仿真运行，由于数量巨大的动态指令数和仿真器本身的低速度，导致了极长的仿真时间。要解决这个问题，我们认为可从三个方面进行：(1)在统计采样技术的基础上，继续寻找使用测试程序片段技术进行仿真的可行方案，如寻找新的采样技术；(2)探寻新的高性能计算技术，使体系结构软件仿真的执行速度得到极大的提高；(3)研究和设计新的处理器性能测试程序，使其不但能准确刻画处理器的性能，同时其指令数少，减少软件仿真的时间。目前，我们正在进行第一个和第三个方面的工作。

参 考 文 献

1 Neil V, Matthew I, Chinmay A, et al. Chip Multi-Processor Scalability for Single-Threaded Applications. ACM SIGARCH Computer Architecture News, 2005, 33(4): 44～53
2 Todd M A. A User's and Hacker's Guide to the SimpleScalar Architectural Research Tool Set. Intel MicroComputer Research Labs, 1997
3 Lizy K J. Performance Evaluation: Techniques, Tools and Benchmarks. http://lca.ece.utexas.edu/pubs-by-type.html # article
4 Stephen R G, John L H. The Accuracy of Trace-Driven Simulations of Multiprocessors: [Technical Report]. CSL-TR-92-546. Computer Systems Laboratory, Department of Electrical Engineering and Computer Science, Stanford University, 1992
5 Dwarkadas S, Jump R J, Sinclair B J. Execution-Driven Simulation of Multiprocessors: Address and Timing Analysis. ACM Transactions on Modeling and Computer Simulation, 1994, 4(4): 314～338
6 Joshua L K, Daniel A C. Statistical Simulation of Multithreaded Architectures. In: Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'2005)
7 Erez P, Greg H, et al. Using SimPoint for Accurate and Efficient Simulation. In: ACM SIGMETRICS'03, San Diego, California, USA, June 2003
8 Matthew C C, Alan D G. Parallel Simulation of Chip-Multiprocessor Architectures. In: High-Performance Computing and Simulation (HCS) Research Laboratory, Department of Electrical and Computer Engineering, University of Florida, 2002
9 Joshua J Y, David J L, et al. Improving Computer Architecture Simulation Methodology by Adding Statistical Rigor. IEEE Transactions on Computers, 2005, 54(11)
10 http://www.cs.wisc.edu/arch/www/tools.html
11 Irina C. A Processor Based Classification of the Instrumentation and Simulation Tools. http://www.cs.inf.ethz.ch/～chihaia/instr-sim.html
12 http://www.cs.wisc.edu/～mscalar/simplescalar.html
13 Srihari C, Chandra S M, Pranav N A. A Fast, Inexpensive and Scalable Hardware Acceleration Technique for Functional Simulation. In: DAC 2002 New Orleans, Louisiana, USA, 2002
14 Roland E W, Thomas F W, et al. SAMRTS: Accelerating Microarchitecture Simulation via Rigorous Statistical Sampling. In: Proceedings of the 30th Annual International Symposium on Computer Architecture (ISCA'03), 2003
15 Shobhit K, Irma E P, et al. FastMP: A Multi-core Simulation Methodology. In: MoBs 2006 Advanced Program, 2006
16 http://www.intel.com/technology/computing/multi-core/
17 Michael V B, Timothy S, et al. A Co-Phase Matrix to Guide Simultaneous Multithreading Simulation. In: IEEE International Symposium on Performance Analysis of Systems and Software, 2004.3
18 http://www.spec.org/osg/cpu2000/press/release.html
19 Greg H, Erez P, et al. Using Machine Learning to Guide Architecture Simulation. Journal of Machine Learning Research, 2006, 7: 343～378
20 Joshua J Y, Sreekumar V K, et al. Characterizing and Comparing Prevailing Simulation Techniques. In: Proceedings of the 11th international Symposium on High-Performance Computer Architecture (HPCA-11 2005)
21 Osowski AJ K, David J L. MinneSPEC: A new SPEC Benchmark Workload for Simulation-based Computer Architecture Research. In: Computer Architecture Letters, 2002
22 Jian H. The Simulator for Multithreaded Computer Architecture, Release 1.2: [Technical Report ARCTiC-00-05]. Laboratory for Advanced Research in Computing Technology and Compilers, University of Minnesota, 2000
23 Timothy S, Erez P, et al. Automatically Characterizing Large Scale Program Behavior. In: International Conference on Architecture Support for Programming Languages and Operating Systems, 2002
24 Thomas M C, Mary A H, et al. Reducing State Loss for Effective Trace Sampling of Superscalar Processors. In: International Conference on Computer Design, 1996
25 David P, Dan F, et al. Exploiting Parallelism and Structure to Accelerate the Simulation of Chip Multi-Processors. In: 12th International Symposium on High-Performance Computer Architecture, 2006
26 James D, Margaret M. An Efficient, Practical Parallelism Methodology for Multi-core Architecture Simulation. Computer Architecture Letters, 2006, 5
27 David A Z, Jarrod A N, et al. NetSim: An Object-Oriented Architectural Simulator Suite. In: The 2005 International Conference on Computer Design, 2005
28 Taeweon S, Shih-Lien L, et al. Initial Observations of Hardware/Software Co-Simulation Using FPGA in Architecture Research. In: Workshop on Architecture Research Using FPGA Platforms in Conjunction with International Symposium on High-Performance Computer Architecture, 2006
29 Joshua J Y, David J L. Simulation of Computer Architectures: Simulators, Benchmarks, Methodologies, and Recommendations. IEEE Transactions on Computers, 2006, 55(3)
30 Penry D, August D. Optimizations for a Simulator Construction System Supporting Reusable Components. In: Proc. Design Automation Conf., 2003
31 Vachharajani M, Vachharajani N, et al. Microarchitecture Exploration with Liberty. In: Proc. Int'l Symp Microarchitecture, 2002
32 Vachharajani M, Vachharajani N, et al. The Liberty Simulation Environment, version 1.0. Performance Evaluation Review: Special Issue on Tools for Architecture Research, 2004, 31(4)
33 Vachharajani M, Vachharajani N, et al. The Liberty Structural Specification Language: A High-Level Modeling Language for Component Reuse. In: Proc. Conf. Programming Language Design and Implementation, 2004
34 Magnusson P, Christensson M. Simics: A Full System Simulation Platform. Computer, 2002, 35(2)