一种软硬件结合的大数据访存踪迹收集分析工具集

李作骏[1,2]，潘海洋[1,2]，陈明宇[1,2]，包云岗[1,2]
1. 中国科学院大学，北京 100049；2. 中国科学院计算技术研究所先进计算机系统研究中心，北京 100190

摘要
以Spark为代表的内存计算框架的兴起、新型非易失性内存研究的逐步深入以及数据安全形势的日益严峻，使得现有的访存行为分析工具无法满足对大数据应用访存行为进行分析的需求。提出了一种软硬件结合的大数据访存踪迹收集分析工具集，在由硬件收集的基本访存踪迹的基础上，结合软件信息同步及离线标注的方式，可以高速、准确、无失真地获取具备丰富语义信息的访存行为信息，且为大数据访存的实时安全监控提供了一种实现方式。最后，通过实验对一组真实的大数据应用进行了访存踪迹采集和分析。

关键词
访存踪迹；访存行为；大数据

中图分类号：TP391　　文献标识码：A　　doi: 10.11959/j.issn.2096-0271.2019031

A hybrid memory trace collection and analysis toolkit for big data applications

LI Zuojun[1,2], PAN Haiyang[1,2], CHEN Mingyu[1,2], BAO Yungang[1,2]
1. University of Chinese Academy of Sciences, Beijing 100049, China
2. Center for Advanced Computer Systems, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China

Abstract
The rise of in-memory computing framework represented by Spark, the gradual deepening of new non-volatile memory research and the increasingly severe data security situation made the existing memory behavior analysis tools unable to meet the demand for big data applications. A software-hardware hybrid memory trace collection and analysis toolkit for big data applications was proposed. Based on the basic memory trace collected by hardware, the memory behavior information with rich semantic information can be obtained quickly, accurately and undistorted by combining software information synchronization and offline annotation. It also provides an implementation method for real-time security monitoring of large data access. Finally, a group of real big data applications were analyzed by this toolkit.

Key words
memory trace, memory behavior, big data

1 引言

在现代计算机系统中，对于大数据、云计算这类以“数据”为核心的应用负载程序而言，“数据”本身已经取代传统的“计算”，成为影响系统性能的关键特征。这就使得有效地设计数据的组织和访问模式成为提升系统性能的关键，而数据的组织和访问模式在整个计算机系统结构中的关键点之一就是数据在中央处理器内部（高速缓存）以及外部存储器（内存）的存储和读写行为方式。因此，获取并分析大数据应用本身的访存行为特征已成为实现高效设计的前提。

以Spark[1]为代表的内存计算框架的兴起，导致大量的计算中间结果被缓存于内存中，现有的基于程序流的调试和分析工具显然已经无法对这类数据流与程序流分离、并包含大量随机与不规则数据访问的大数据应用进行有效的分析。此外，非易失性内存[2]在一些大数据系统研究中的逐步应用，需要对整个计算机的访存系统进行重新评估和设计，而目前普遍缺少基于大量应用访存行为模式并且能够对新型内存系统进行分析、预测和总结的工具。值得注意的是，大数据领域的信息安全对各大互联网企业以及政府机构的影响日渐突出，传统的基于等级保护和程序审查的方式已无法阻止数据通过病毒、木马或漏洞攻击被窃取或修改了，最终的“数据”安全必须能够实时监控到系统中的关键数据在何时以何种方式被访问、修改以及被何人修改等。这就要求有一种方式可以实时监控系统中数据被访问的全过程，并且及时做出安全防护动作。

目前，国内外可用的测量和分析方法有两类：一类是纯软件的“植入式”工具或模拟平台，另一类是纯硬件的物理信号采集。软件方法（如DRAMSim2[3]、Simics[4]、gem5[5]、CMP$IM[6]、MARSSx86[7]等软件模拟器）虽然可以获取含有丰富语义信息的访存踪迹信息，但是其模拟速度通常低于实际运行速度的千分之一，如Pin[8]、Valgrind[9]、ATOM[10]和PEBIL[11]等使用代码插入的方法，虽然执行速度较软件模拟的方式稍快，但还是会无法避免地造成访存干扰和较大的性能开销。硬件方法虽然大多能够非常快速地获取访存信息，但是使用硬件计数器的方法（如VTune[12]、DTrace[13]、OProfile等）只能获取有限公开的统计信息，并不能获取完整的访存踪迹，如RAMP[14]、BEECube-BEE[15]、Palladium XP、Dini Group、HAPS等硬件仿真平台往往受限于其平台规模、设计开发工作量，不能完整地仿真主流处理器平台和一些大规模商用软件；MemorIES[16]、PHA$E[17]、ACE[18]、RACFs[19]等硬件侦听工具及具备内存监听和分析功能的示波器、逻辑分析仪（如Lecroy、Nexus等产品）的价格非常昂贵，存储容量也非常有限，且无法一次性采集大数据应用TB级以上的访存踪迹。

这就使得大数据领域迫切需要一种高效的监测手段，细致地“观察”数据是如何在内存中被使用的。为此，笔者提出并研制了一种新型的软硬件结合、集访存踪迹收集和分析为一体的工具集——软硬件结合的访存踪迹工具集（hybrid memory trace toolkit，HMTT），该工具集可以满足人们对大数据应用访存行为进行监测和分析的需求，其特点见**表1**。

2 软硬件结合的访存踪迹收集分析工具集

笔者提出的HMTT是为了解决上述

各类方法的不足而实现的专用内存数据分析工具集。HMTT主要是由访存踪迹采集卡、被测系统软件信息同步模块、访存踪迹接收和存储设备以及离线标注工具包4个部分组成的。

访存踪迹采集卡主要由一块现场可编程门阵列（field-programmable gate array，FPGA）芯片、双倍数据速率（double data rate，DDR）总线分支、高速外设部件互联总线标准（peripheral component interconnect express，PCIe）传输模块及其他功能电路组成。访存踪迹采集卡插在测试机的内存插槽上，通过监听DDR总线的形式将原有的访存指令信息转化为包含访存物理地址、访存时延以及读写特征等的访存踪迹信息，并通过PCIe连接线传输给访存踪迹接收和存储设备，如**图1**所示。此外，人们可以在FPGA上设置定制的过滤逻辑，以减少回传的数据量。

被测系统软件信息同步模块运行于测试机的计算机系统中，在程序启动或特定信息修改（如缺页异常等）时，记录应用元信息的变化，并在合适的时机将其传输给接收机进行处理。此外，研究人员还可以通过访问特定物理地址的形式告知访存踪迹采集卡及离线标注工具包一些特定的信息，这些特定的信息可以被用于控制访存踪迹采集卡或给离线标注工具包提供一些程序关键行为信息。

访存踪迹接收和存储设备主要是由一块PCIe转接卡和一个TB级的存储阵列通过一台接收机组成的。这部分主要负责接收由访存踪迹采集卡通过PCIe连接线传输来的访存踪迹信息，并将其无丢失地存储到存储阵列中。此外，接收机还可以通过对应的软件配置模块经由PCIe链路对访存踪迹的收集进行控制和过滤。

离线标注工具包主要由访存踪迹解析模块和访存行为分析模块两部分组成。访

表1 访存踪迹获取方法的对比

方法	完整	快速	无失真	丰富语义信息	低成本
软件模拟	√-	×	√	√	√
代码插入	√-	×	×	√-	√
硬件计数器	×	√	√	×	√
硬件仿真	√	√	×	√	×
硬件侦听	√	√	√	×	√-
HMTT	√	√	√	√	√

注：√表示满足；×表示不满足；√-表示部分满足

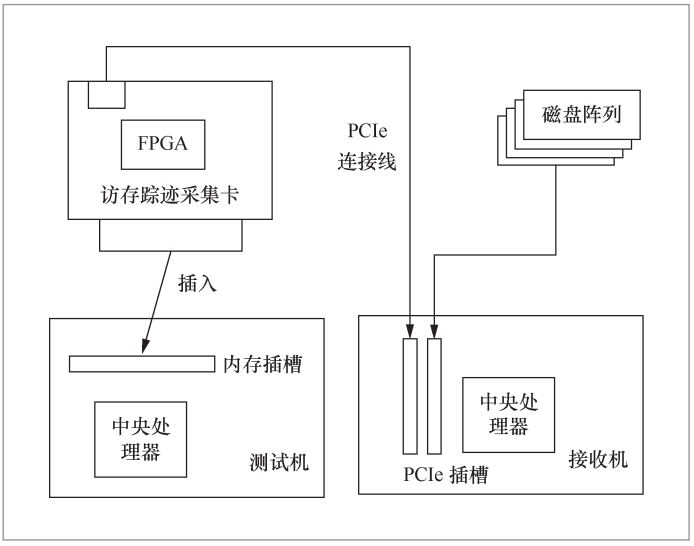


图1 HMTT4 测试平台结构

存踪迹解析模块将接收到的访存踪迹信息进行解析，验证访存踪迹信息的完整性，并提取出其中的物理地址、访存时延、读写特征等各类基本访存信息。访存行为分析模块可以将访存踪迹解析模块获得的各类基本访存信息与在被测系统软件信息同步模块中获取到的页表、锁、函数调用、系统调用等信息结合，得到最终需要的操作系统和应用层面的高级语义信息，进一步分析程序的访存行为特征。

HMTT很好地解决了之前各种工具的问题：在访存踪迹完整性方面，HMTT的离线存储可以连续存储以小时计、TB级的访

存踪迹数据；在采集快速性方面，HMTT采用硬件旁路侦听的方式，使得软件的执行速度几乎不受影响；在无失真性方面，除了少量插桩，软件几乎不用修改，也不会干扰指令流、高速缓存替换；在语义信息方面，HMTT开发了多种元信息采集和标注方法，将物理的踪迹数据还原成对应语义信息的数据，为各种应用提供了基础；在成本方面，HMTT不依赖昂贵的测量仪器，成本不到具有相似功能的内存协议分析仪的20%。

经历十余年的不断研究，HMTT已经从第1代发展到最新的HMTT 4[20-23]，目前基本能够满足大数据应用访存行为分析的需求，4代HMTT的基本特征见**表2**。

3 应用场景

HMTT实验平台至今已应用于多个体系结构分析领域，包括直接内存访问（direct memory access，DMA）缓冲区、最后一级缓存（last level cache，LLC）、转换检测缓冲区（translation lookaside buffer，TLB）、多线程锁、程序对象、虚拟地址、功耗评估及非易失性内存等，并有可能进一步应用在大型程序的分析、优化和调试、数据访问安全分析等方面。

对于DMA缓冲区而言，可以将预先定义好的HMTT软件应用程序编程接口（application programming interface，API）插入Linux操作系统中的网卡或者硬盘等设备驱动中，然后通过检测HMTT收集的访存踪迹中的标记信息，区分应用执行过程中的DMA访存行为和处理器访存行为，为后续的DMA访存行为优化提供有效的数据支撑[24]。

对于LLC而言，HMTT可以捕捉到全部LLC缺失的访存操作时间和地址，因此可以通过对HMTT收集的访存踪迹信息的时延特征进行分析，获取系统访问不同物理地址序列对LLC缺失的影响，从而指导应用使用更为合理的内存分配技术，如页着色、大页面等，提升应用对LLC的利用率[25]。

对于TLB而言，通过将内核特定进程的页表设置成不可缓存的页面，可以使所有TLB缺失后访问页表的操作都进入DDR总线，从而被HMTT完整地捕捉并进行分析，得到整个程序执行过程中造成TLB缺失的页面序列，进而对程序和TLB自身算法的优化进行指导[26]。

对于多线程锁而言，研究人员可以通过在Linux操作系统中调用与多线程锁相关的函数，插入HMTT软件API，生成一个包含当前锁相关信息的针对特定物理地址的内存访问，从而降低开销，并精确地监测到多线程应用的锁冲突[27]。

对于程序对象而言，研究人员可以通过动态库重写的方式替换Linux中原有的malloc( )函数，配合HMTT收集的访存踪迹信息，使得在不修改源代码的情况下得到虚拟地址范围与各个对象的对应关系，从而区分各个对象的访存行为信息[25]。

表 2 4代 HMTT 的基本特征

版本	内存速率	传输协议	传输带宽	其他特性
HMTT 1	DDR-200	千兆以太网	125 MB/s	旁路监听、行为标记
HMTT 2	DDR2-667	千兆以太网	375 MB/s	功耗监控、踪迹压缩、页表信息
HMTT 3	DDR3-800	PCIe Gen1	1 GB/s	内存集成、多板同步、函数、对象及I/O信息
HMTT 4	DDR4-1600	PCIe Gen2	2 GB/s	稳定性、可移植性提升

对于虚拟地址而言，研究人员可以通过修改Linux内核记录每次页表信息的更新，然后再将HMTT获取的访存物理地址信息通过反查页表的形式转化为所需的虚拟地址信息[28]。

对于功耗评估而言， HMTT有一个定制版本在访存踪迹采集卡上集成了功耗检测组件，可以真实地获取到毫秒级的内存模组的功耗[29]。

在非易失性内存领域，HMTT已为非易失性内存的数据放置、动态随机存取存储器缓存的预取算法研究提供了前期的数据支撑[30] 。

此外，HMTT还可能用于对系统软件的分析和调试。包括Firmware、OS启动代码、设备驱动程序等在内的系统软件的调试一直是软件调试的难题，因为在这些操作系统启动的初始阶段很难建立一个稳定可靠、支持调试的子环境。采用HMTT工具，研究人员可以从第一条指令的执行就开始分析，而无须等待软件内部建立支持调试的子环境。

HMTT还可能用于大型程序分析和优化：对于复杂的大型程序，HMTT可以低干扰地进行长时间、细粒度的访存特征分析，为定位系统瓶颈、划分存储特征区域、进一步优化程序设计等提供支持。

HMTT也可能用于数据安全领域。恶意软件一般可以对基于软件的调试工具进行感知和防范。而使用HMTT可以在完全不影响原有程序的情况下，全面监控敏感数据区域何时被访问、何时被修改，从而协助定位系统中不安全的因素，包括软件难以发现的侧信道攻击等。因此HMTT为大数据的安全研究提供了一种新的解决方案。

4 实验评测

本文基于Intel E5 2620 V2处理器搭建了一套HMTT3访存踪迹收集分析平台，选用大数据内存相关测试基准——大内存测试集（big memory bench），对其中的部分应用进行了行缺失率的分析。实验平台参数见**表3**。

大内存测试集是一种面向大容量内存访问的测试基准，它不同于传统的低内存容量占用的测试基准SPEC 2006[31]、PARSEC[32]，也不同于大多只扩展了自身数据规模的大数据领域测试基准Big Data Bench[33]，其工作集往往会占用较大的内存容量，且在各自领域都被广泛使用。笔者在试验中选取了大内存测试集中的6个应用，见**表4**，除了SAP HANA是专用的内存数据库外，其余5个都是基于Spark的应用，完整的访存踪迹数据有2 TB左右。

为了说明HMTT的功能，笔者选择了行缺失率的统计分析功能。行缺失率是评价一个应用访存性能以及功耗的重要指标之一，在现代内存芯片中采用行缓冲区这

表 3 实验平台参数

实验平台	组件	参数
发送机	中央处理器	两个Intel E5-2620V2,，6核心12线程，主频2.1 GHz，三级缓存15 MB
	内存	8条32 GB内存，共256 GB，DDR3-800，8路交错
	操作系统	CentOS 7
接收机	磁盘阵列	4个1 TB固态硬盘组成4 TB大小的磁盘阵列，最高写入带宽为2 000 MB/s

表 4 实验使用的 Big Memory Bench 工作集

工作集	使用内存大小/GB	访存踪迹大小/GB	描述
Quicksort	247.9	195	排序
SAP HANA	111.9	362	内存计算数据库
PageRank	253.4	553	搜索引擎
TeraSort	253.2	235	排序
Bayes	206.8	96	分类
Kmeans	95.2	425	聚类

一架构，通过缓存一整行的内存数据信息减少打开内存行地址的次数，从而降低对同一行的访存时延。如果行缺失率较高，说明当前应用的访存随机性较大，访存的并发度不高，这会导致在内存芯片中频繁出现打开行和关闭行的操作，提高访存时延和访存功耗。但是传统的CPU的性能计数器一般不提供行缺失的信息统计，而模拟器方式速度太慢，无法获得完整的统计信息。HMTT可以收集到完整的访存踪迹，并对其进行离线分析，能够非常简单地获取行缺失率的信息。

获取行缺失率信息的关键在于准确获取当前环境下的物理地址与内存行列地址的具体映射关系。这一信息很难通过传统的软件或者硬件方法单独获取，而使用HMTT通过固定序列访存踪迹模式分析可以非常简单地得到。在获取当前环境下的物理地址与内存行列地址的具体映射关系之后，研究人员可以通过离线分析准确获取应用的访存行缺失率。**图2**为上述6个工作集的访存行缺失率统计信息，其中SAP HANA的行缺失率高达54.7%，而Bayes仅有24.24%。

5 结束语

本文提出的软硬件结合的大数据访存踪迹收集分析工具集能够完整、快速、无失真地获取具备丰富语义信息的大数据应用访存踪迹，对大数据系统访存行为特征分析、非易失性内存在大数据领域的应用以及实时大数据安全监控提供了有效支撑。

目前HMTT支持DDR3-800和DDR4-1600速率下的访存地址相关踪迹收集和分析，对如何完整获取数据总线的研究还在进行当中。此外，如何优化访存踪迹采集卡的设计，实现小型化和多卡联动，也是笔者后续的关注点之一。

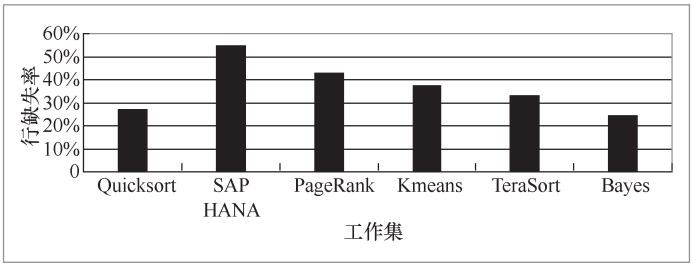


图 2 访存行缺失率

参考文献:

[1] ZAHARIA M, CHOWDHURY M, FRANKLIN M J, et al. Spark: cluster computing with working sets[C]//The 2nd USENIX Conference on Hot Topics in Cloud Computing, June 22-25, 2010, Boston, USA. Berkeley: USENIX Association, 2010: 10.

[2] BEZ R, PIROVANO A. Non-volatile memory technologies: emerging concepts and new materials[J]. Materials Science in Semiconductor Processing, 2004, 7(4-6): 349-355.

[3] ROSENFELD P, COOPERBALIS E, JACOB B. DRAMSim2: a cycle accurate memory system simulator[J]. IEEE Computer Architecture Letters, 2011, 10(1): 16-19.

[4] MAGNUSSON P, CHRISTENSSON M, ESKILSON J, et al. Simics: a full system simulation platform[J]. Computer, 2002, 35(2): 50-58.

[5] BINKERT N, BECKMANN B, BLACK G, et al. The Gem5 simulator[J]. ACM Sigarch Computer Architecture News, 2011, 39(2): 1-7.

[6] JALEEL A, COHN R, LUK C K, et al. CMP$im: a pin-based on-the-fly multi-core cache simulator[C]//The 4th Annual

Workshop on Modeling, Benchmarking and Simulation, June 22, 2008, Beijing, China. [S.l.:s.n.], 2008: 28-36.

[7] PATEL A, AFRAM F, CHEN S F, et al. MARSSx86: a full system simulator for multicore x86 CPUs[C]//The 48th ACM/EDAC/IEEE Design Automation Conference, June 5-9, 2011, New York, USA. Piscataway: IEEE Press, 2011: 1050-1055.

[8] LUK C K, COHN R, MUTH R, et al. Pin: building customized program analysis tools with dynamic instrumentation[C]// The 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation, June 12-15, 2005, Chicago, USA. New York: ACM Press, 2005: 190-200.

[9] NETHERCOTE N, SEWARD J. Valgrind: a framework for heavyweight dynamic binary instrumentation[C]//The 28th ACM SIGPLAN Conference on Programming Language Design and Implementation, June 10-13, 2007, San Diego, USA. New York: ACM Press, 2007: 89-100.

[10] SRIVASTAVA A, EUSTACE A. ATOM: a system for building customized program analysis tools[C]//The ACM SIGPLAN 1994 Conference on Programming Language Design and Implementation, June 20-24, 1994, Orlando, USA. New York: ACM Press, 1994: 196-205.

[11] LAURENZANO M, TIKIR M, CARRINGTON L, et al. PEBIL: efficient static binary instrumentation for Linux[C]//IEEE International Symposium on Performance Analysis of Systems & Software, March 28-30, 2010, White Plains, USA. Piscataway: IEEE Press, 2010: 175-183.

[12] REINDERS J. VTune performance analyzer essentials: measurement and tuning techniques for software developers[M]. Santa Clara: Intel Press, 2004: 1-500.

[13] GREGG B, MAURO J. DTrace: dynamic tracing in Oracle Solaris, Mac OS X, and free BSD[M]. Upper Saddle River: Prentice Hall Professional, 2011: 1-1152.

[14] TAN Z X, WATERMAN A, AVIZIENIS R, et al. RAMP gold: an FPGA-based architecture simulator for multiprocessors[C]//The 47th Design Automation Conference, June 13-18, 2010, Anaheim, USA. New York: ACM Press, 2010: 463-468.

[15] ROTHMAN J, CHEN C. BEE technology overview[C]//International Conference on Embedded Computer Systems, July 16-19, 2012, Samos, Greece. Piscataway: IEEE Press, 2012: 277.

[16] NANDA A, MAK K K, SUGAVANAM K, et al. MemorIES: a programmable, real-time hardware emulation tool for multiprocessor server design[J]. ACM SIGPLAN Notices, 2000, 35(11): 37-48.

[17] CHALAINANONT N, NURVITADHI E, MORRISON R, et al. Real-time L3 cache simulations using the programmable hardware-assisted cache emulator (PHA$E)[C]//IEEE International Conference on Communications, October 27, 2003, Austin, USA. Piscataway: IEEE Press, 2003: 86-95.

[18] HONG J, NURVITADHI E, LU S L. Design, implementation, and verification of active cache emulator (ACE)[C]//The 2006 ACM/SIGDA 14th International Symposium on Field Programmable Gate Arrays, February 22-24, 2006, Monterey, USA. New York: ACM Press, 2006: 63-72.

[19] YOON H M, PARK G H, LEE K W, et al. Reconfigurable address collector and flying cache simulator[C]//The High-Performance Computing on the Information Superhighway, April 28-May 2, 1997, Seoul, Korea. Piscataway: IEEE Press, 1997: 552.

[20] 阮元, 陈明宇, 包云岗, 等. 基于硬件的内存trace工具——MTT的设计与实现[J]. 电子学报, 2008, 36(8): 1519-1525.

RUAN Y, CHEN M Y, BAO Y G, et al. The design and implementation of MIT a hardware-based memory trace tool[J]. Acta Electronica Sinica, 2008, 36(8): 1519-1525.

[21] 朱晏. 内存行为、功耗监控平台的硬件设计[D]. 北京: 中国科学院大学, 2011.
ZHU Y. Hardware design of memory behavior and power monitoring platform[D]. Beijing: University of Chinese Academy of Sciences, 2011.

[22] 李作骏. 访存踪迹收集工具的逻辑设计与实现[D]. 北京: 中国科学院大学, 2016.
LI Z J. Logical design and implementation of memory trace collection tool[D]. Beijing: University of Chinese Academy of Sciences, 2016.

[23] 李作骏, 陈明宇. 一种支持DDR4的软硬件结合的访存踪迹收集分析工具集[C]//全国高性能计算学术年会, 2018年10月18-20日, 青岛, 中国. 北京: 中国计算机学会, 2018: 224-231.
LI Z J, CHEN M Y. Hybrid memory trace collection and analysis toolkit for DDR4[C]//National Annual Conference on High Performance Computing, October 18-20, 2018, Qingdao, China. Beijing: China Computer Federation, 2018: 224-231.

[24] TANG D, BAO Y G, HU W W, et al. DMA cache: using on-chip storage to architecturally separate I/O data from CPU data for improving I/O performance[C]//The 16th International Symposium on High-Performance Computer Architecture, January 9-14, 2010, Bangalore, India. Piscataway: IEEE Press, 2010.

[25] CHEN L C, CUI Z H, BAO Y G, et al. A lightweight hybrid hardware/software approach for object-relative memory profiling[C]//The 2012 IEEE International Symposium on Performance Analysis of Systems & Software, April 1-3, 2012, New Brunswick, USA. Piscataway: IEEE Press, 2012: 46-57.

[26] ZHANG J T, LIU Y H, LI H F, et al. PTAT: an efficient and precise tool for tracing and profiling detailed TLB misses[C]//ACM Transactions on Embedded Computing Systems (TECS), April 24-25, 2017, Santa Rosa, USA. Piscataway: IEEE Press, 2017.

[27] HUANG Y B, CUI Z H, CHEN L C, et al. HaLock: hardware-assisted lock contention detection in multithreaded applications[C]//The 21st International Conference on Parallel Architectures and Compilation Techniques, September 19-23, 2012, Minneapolis, USA. New York: ACM Press, 2012.

[28] BAO Y G, CHEN M Y, RUAN Y, et al. HMTT: a platform independent full-system memory trace monitoring system[C]//The 2008 ACM International Conference on Measurement and Modeling of Computer Systems, June 2-6, 2008, Annapolis, USA. New York: ACM Press, 2008: 229-240.

[29] CUI Z H, ZHU Y, BAO Y G, et al. A fine-grained component-level power measurement method[C]//2011 International Green Computing Conference and Workshops, July 25-28, 2011, Orlando, USA. Piscataway: IEEE Press, 2011.

[30] LU T Y, LIU Y H, PAN H Y, et al. TDV cache: organizing off-chip DRAM cache of NVMM from a fusion perspective[C]//The 35th IEEE International Conference on Computer Design (ICCD), November 5-8, 2017, Boston, USA. Piscataway: IEEE Press, 2017.

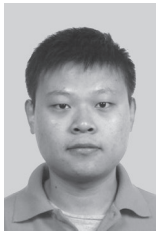
[31] HENNING J. SPEC CPU2006 benchmark descriptions[J]. ACM SIGARCH Computer Architecture News, 2006, 34(4): 1-17.

[32] BIENIA C, KUMAR S, SINGH J, et al. The PARSEC benchmark suite: characterization and architectural implications[C]//The 17th International Conference on Parallel Architectures and Compilation Techniques, October 25-29, 2008, Toronto, Canada.

New York: ACM Press, 2008.
[33] WANG L, ZHAN J F, LUO C J, et al. BigDataBench: a big data benchmark suite from Internet services[C]//IEEE 20th International Symposium on High Performance Computer Architecture, February 15-19, 2014, Orlando, USA. Piscataway: IEEE Press, 2014.

作者简介

李作骏(1990-),男,中国科学院大学博士生,主要研究方向为计算机系统结构。



潘海洋(1990-),男,中国科学院大学博士生,主要研究方向为计算机系统结构。

陈明宇(1972-),男,博士,中国科学院计算技术研究所研究员,主要研究方向为高性能计算机体系结构、操作系统及并行算法优化。

包云岗(1980-),男,博士,中国科学院计算技术研究所研究员,计算机体系结构国家重点实验室教授,先进计算机系统研究中心主任,中国科学院大学岗位教授。担任中国计算机学会理事、普及工作委员会主任,中国科学院青年创新促进会理事。主要研究方向为计算机系统结构,主持研制多款达到国际先进水平的系统,在国际会议期刊发表了40余篇论文,相关技术在华为技术有限公司、阿里巴巴集团、英特尔公司等国内外企业应用,多次受邀担任ASPLOS、ISCA、MICRO、SC等国际会议的程序委员会委员。

收稿日期:2019-05-03
基金项目:"十三五"国家重点研发计划基金资助项目(No.2017YFB1001602)
Foundation Item: The National Key Research and Development Program During the Thirteenth Five-Year Plan Period (No.2017YFB1001602)