一种混合型值关联间接跳转预测机制

谭明星[1,2,3],刘先华[1,2],张吉豫[1,2],佟 冬[1,2],程 旭[1,2,3]
(1.北京大学信息科学技术学院,北京 100871;2.微处理器及系统教育部工程研究中心,北京 100871;
3.北京大学深圳研究生院,广东深圳 518055)

摘 要: 准确的间接跳转预测对现代处理器的性能和能耗有效性都具有重要意义.本文提出了一种混合型值关联间接跳转预测机制,通过混合使用多种关联信息以降低间接跳转误预测率.该机制一方面依赖于编译器根据高层次数据流信息识别间接跳转指令所对应的初始关联数据值.另一方面,该机制针对间接跳转预测的不同场景分别设计了两类关联信息:单一数据值和值历史,并实现了一种低开销的硬件结构,该硬件结构在运行时刻根据不同应用场景动态选择最佳关联信息引导间接跳转预测.实验结果表明,相对于传统的 BTB 预测器和最新的 VBBI 预测器,本文机制能够有效降低误预测率,提高程序性能并降低系统能耗.

关键词: 转移预测; 间接跳转; 值关联; 混合型预测器
中图分类号: TP302.7 **文献标识码:** A **文章编号:** 0372-2112 (2012)11-2298-05
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2012.11.024

A Hybrid Value Correlation Based Indirect Jump Prediction

TAN Ming-xing[1,2,3], LIU Xian-hua[1,2], ZHANG Ji-yu[1,2], TONG Dong[1,2], CHENG Xu[1,2,3]
(1. *School of Electronics Engineering and Computer Science, Peking University, Beijing* 100871, *China*;
2. *Engineering Research Center of Microprocessor & System, Ministry of Education, Beijing* 100871, *China*;
3. *Shenzhen Graduate School of Peking University, Shenzhen, Guangdong* 518055, *China*)

Abstract: Accurate indirect jump prediction is critical for the performance and energy efficiency of modern high-performance processors. This paper proposes the Hybrid Value Correlation (HVC) based indirect jump prediction, which combines various types of correlated information to reduce indirect jump mispredictions. First of all, our mechanism relies on the compiler to identify the correlated data values based on high-level dataflow information. Second, our mechanism maintains two kinds of correlated information: the single data value and the value history. Our mechanism makes use of a low-cost hardware structure, which dynamically chooses the best correlated information for indirect jump prediction according to different processor states. Experimental results show that HVC prediction can significantly reduce the misprediction rate over the baseline-BTB prediction and the state-of-the-art VBBI prediction. The low misprediction rate of HVC prediction leads to better performance and lower power consumption over previous predictors.

Key words: branch prediction; indirect jump; value correlation; hybrid prediction

1 引言

准确的转移预测对现代处理器的性能和能耗有效性都具有重要意义[1~4].转移指令按照转移条件和转移目标地址寻址方式不同可分为 4 类:无条件直接转移指令、条件直接转移指令、无条件间接转移指令和条件间接转移指令.间接转移指令的频率低于条件转移指令,主要应用于 Switch-case 语句、函数指针调用、虚函数调用、函数返回等结构中.然而,间接转移指令因具有多个动态目标而难以预测,除函数返回指令可以通过返回地址栈(RAS)[5]获得较低误预测率以外,其它间接转移指令的误预测率通常比较高[1,2].本文将除函数返回指令以外的其它间接转移指令简称为"间接跳转指令".随着面向对象程序、动态链接库和虚拟机的广泛应用,间接跳转指令的预测失效开销已成为影响处理器性能和能耗的重要因素[1,2].

本文面向现代超标量处理器提出了一种混合型值关联间接跳转预测机制(Hybrid Value Correlation based indirect jump prediction, HVC).与传统间接跳转预测技术不同,本文将编译优化和处理器设计相结合,通过混合使用多种关联信息来降低间接跳转指令的误预测率.首先,由编译器根据高层次数据流信息识别关联数据值,并通过引导指令进行标记;然后,处理器在运行时刻根据不同场景,动态选择最优关联信息引导间接跳转指令预测.本文机制使用一个 520 比特小容量缓冲区保存值关联信息,硬件开销小.实验结果表明,本文机制只需

收稿日期:2011-05-05;修回日期:2012-01-10
基金项目:国家核高基重大专项(No.2009ZX01029-001-002;No.2009ZX01036-001-003);北京市自然科学基金(No.4123098)

要添加很少的硬件支持,就可以有效降低间接跳转误预测率,提高处理器性能并降低系统能耗.

2 相关工作

现有处理器主要使用目标地址缓冲区(Branch Target Buffer,BTB[6])进行间接跳转预测.它利用间接跳转局部性特征,总是预测目标地址为该指令最近发生的跳转地址,预测准确率较低.为减小误预测率,先后有历史关联、数据值关联等多种预测器被提出.

Chang 等人提出了基于转移历史信息的 TTC 预测器[7],使用额外硬件缓冲区保存目标地址,并使用全局历史作为关联信息.由于需要额外的大容量硬件缓冲区,该预测器仅用于 Pentium M[8] 等少量处理器.文献[9]分析了 TTC 预测器中误预测的产生原因,通过改进索引、压缩存储等方法减小硬件开销并提高性能.Driesen 等人[10]提出了一种使用路径历史信息的 Cascade 预测器,将间接跳转指令动态划分为不同类型,分别使用不同预测方法以避免相互干扰.Seznec[11]提出了一种 ITTAGE 预测器,使用非常长的转移历史和部分匹配算法进行预测,硬件容量大且控制逻辑较为复杂.最近,Kim 等人提出了一种避免大容量硬件缓冲区的 VPC 预测器[1],将间接跳转预测转化为多个条件转移预测.然而,VPC 预测器需要多次迭代访问条件转移预测器,增加了预测延时开销并降低了性能.

Heil 等人在文献[12]中最早指出,使用数据值作为关联信息,可降低部分转移指令的误预测率.Chen[13]提出了一种使用专有硬件动态跟踪数据依赖关系的方法,并将该方法应用于转移预测以降低误预测率.然而,该方法需要复杂硬件逻辑,硬件开销大.最近,Farooq 等人提出了一种编译指导的 VBBI 预测器[2],借助于剖视(profiling)手段识别关联信息.然而,由于现代处理器流水线延时和访存延时会引起关联数据值经常不可用,VBBI 预测器也难以取得理想效果.

此外,Roth 等人[14]提出了一种针对虚函数调用的预计算方法,通过特殊部件提前产生目标地址从而避免预测失效.Joao 等人提出了一种 DIP 机制[15],通过断言执行间接跳转指令的多条路径,简化目标地址预测.然而,它们都需要复杂硬件设计,难以实用.

3 混合型预测机制的提出

3.1 问题分析

近期研究[1,2]表明,数据值关联信息对间接跳转预测更为有效.值关联预测器将间接跳转指令目标地址与变量数据值建立关联关系,在关联数据值产生时将其保存在缓冲区中,然后在间接跳转预测时使用已保存的关联数据值引导间接跳转预测.由于关联数据值是在处理器后端产生(如 write-back 阶段),而转移预测在前端发生(如 fetch 阶段),因此最新关联数据值在间接跳转预测时刻可能还没有产生,即:预测时刻最新关联数据值不可用.此时,传统值关联预测器都只能使用前一次预测时所保存的数据值作为关联信息引导当前转移预测,从而引起误预测率上升和性能下降.关联数据值在预测时刻不可用问题,是制约值关联间接跳转预测器的重要因素.在宽发射、深度流水的高性能处理器中,该问题更加突出.本文基于 4 发射、16 级流水线的处理器,模拟评测了关联数据值不可用情况的比例.结果表明,平均情况下有超过 70% 的动态间接跳转指令在预测时刻关联数据值是不可用的.

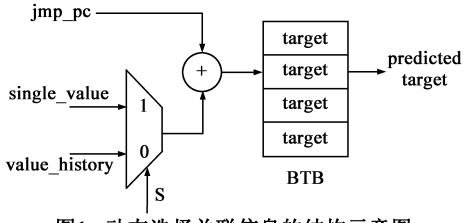

图1 动态选择关联信息的结构示意图

3.2 混合型值关联预测机制

针对关联数据值在预测时刻不可用问题,本文提出了一种混合型值关联间接跳转预测机制,其基本特点是根据不同应用场景分别使用不同关联信息引导间接跳转预测.图 1 给出了动态选择关联信息的示例图,处理器同时维护两类关联信息:单一数据值(single_value)和值历史(value_history).当最新关联数据值可用时($S=1$),使用当前可用的单一数据值引导间接跳转预测.当最新关联数据值不可用时($S=0$),使用值历史引导间接跳转预测.值历史是一种由多个关联数据值组合形成的复合体,它反映了关联数据值的规律.表 1 给出了值历史的示例,假设间接跳转指令的关联数据值按照时间先后顺序依次为 $a,a,b,b,a,a,b,b,\cdots$,通过将多个关联数据值组合形成值历史,则可根据值历史完全确定当前关联数据值.例如,包含两个关联数据值的值历史如表 1 所示,值历史(a,a)总是对应 b,而值历史(b,b)总是对应 a.值历史反映了关联数据值出现的规律,当最新关联数据值不可用时,可使用值历史信息引导间接跳转预测.

表 1 值历史示意图

关联数据值	a	a	b	b	a	a	b	…
值历史	$(-,-)$	$(-,a)$	(a,a)	(a,b)	(b,b)	(b,a)	(a,a)	…

4 混合型值关联间接跳转预测器设计

4.1 总体设计

本文机制首先基于指令分类思想[10]将间接跳转指令分为“难预测指令”和“普通跳转指令”.对每一个难

预测指令,编译器根据高层次数据流信息识别初始关联数据值,并插入引导指令以标记关联数据值.在运行时刻,当引导指令发射时,处理器根据该引导指令所标记的关联数据值,同时更新单一数据值和值历史.当取指阶段遇到间接跳转指令时,处理器动态选择最优关联信息:若最新关联数据值可用,则选择单一数据值作为关联信息,否则选择值历史作为关联信息.然后,处理器使用关联信息与间接跳转指令地址的异或运算结果作为索引,访问 BTB 以获得预测目标地址.在指令提交阶段,处理器使用实际跳转目标更新 BTB.

4.2 初始关联数据值的识别与标记

本文机制依赖于编译器根据高层次数据流信息识别初始关联数据值.在高级语言源程序级别,间接跳转指令主要用于 Switch-case 语句、函数指针调用、虚函数调用、函数返回等程序结构中,其中函数返回指令可使用返回地址栈(RAS)[5]进行预测,本文预测机制主要针对除函数返回结构的其它间接跳转指令.由于间接跳转指令的目标地址完全由跳转寄存器的值所决定的,因此编译器需要找到定义该跳转寄存器的地方并进行标记.具体而言包括识别和标记两个过程,首先由编译器通过函数内部的数据流分析,获得跳转寄存器"Use-Define"链,并将所有 Define 点作为关联数据产生点.然后,编译器在关联数据产生点插入引导指令以标记关联数据值.实现引导指令的方法主要有复用已有指令和新添加指令两种方式.现有 VBBI 技术采用指令复用方式,但是这会导致关联信息缓冲区需要较多的读写口,并且每个间接跳转指令最多只能对应一个关联数据.为降低硬件设计复杂度并扩大引导指令的适用范围,本文通过扩展指令系统以添加引导指令,使每个间接跳转指令可以对应多个关联数据值.图 2 给出了引导指令的格式,其中"opcode"为预留指令编码位,"ra"为关联数据值所对应的寄存器编号,"s_offset"表示引导指令与相应间接跳转指令之间的距离,最高位为符号位.对每个间接跳转指令,编译器通过插入一个或多个引导指令,以标记关联数据值.

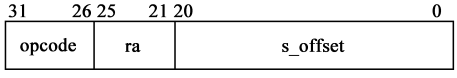

图2 引导指令格式

4.3 预测器硬件结构设计

图 3 展示了 HVC 预测器的硬件结构设计.关联信息缓冲区(Correlated Information Buffer,CIB)是关键部件,它同时保存了两类关联信息:单一数据值和值历史.CIB 中每个表项包含四个域:pc_tag 为标签域,*S* 为状态位,single_value 为单一数据域,value_history 为值历史域.处理器在引导指令发射阶段更新 CIB,在间接跳转指令取指阶段读取 CIB.对一组关联的引导指令和间接跳转指令,处理器使用相同的索引访问 CIB,以保证 CIB 同一表项的更新和读取过程是相互匹配的.

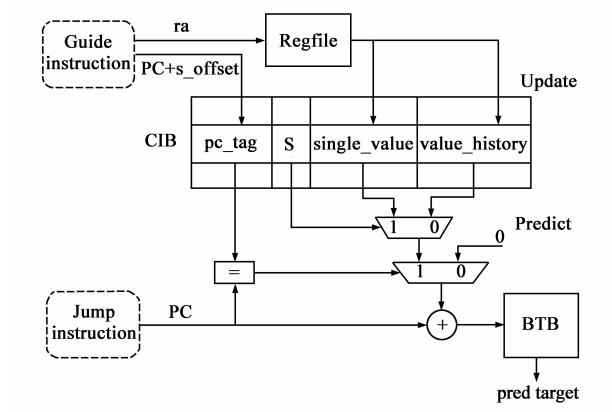

图3 HVC预测器硬件结构图

在引导指令发射阶段,它所标志的关联数据值已经就绪,处理器从寄存器文件中读取'ra'寄存器作为初始关联数据值,同时以引导指令 PC(program Counter)与 s_offset 相加的结果作为索引查找 CIB.若无法找到匹配项,则使用先入先出(First-In-First-Out,简称 FIFO)策略为该间接跳转指令分配一项,并将该项所有内容初始化为 0.接下来,处理器使用初始关联数据值分别更新两类关联数据值,其中单一数据值(single_value)的更新通过赋值操作完成,而值历史(value_history)的更新则是将已有的值历史与当前初始关联数据值进行拼接,即:将 value_history 左移之后再加上初始关联数据值的部分位域.不同的拼接方式会影响到值历史的有效性,本文实验中采用统一的经验值,但在微处理器具体设计中可将其设计为可配置的部件,使其能够根据实际应用进行调整.

在取指阶段当一个跳转指令被取出时,处理器首先以该间接跳转指令 PC 为索引访问 CIB.若无法找到匹配项,则说明当前指令是一个普通跳转指令,此时处理器直接以当前指令 PC 为索引访问 BTB 获得预测目标地址,即此时是以 0 作为关联信息.若 CIB 中存在与当前跳转指令相匹配的项,则说明当前指令是一个难预测间接跳转指令,此时处理器根据 *S* 位动态选择的关联信息:若 *S* 位为 1 则使用单一数据值(single_value)作为关联信息,若 *S* 位为 0 则使用值历史(value_history)作为关联信息.接下来,处理器使用关联信息与指令 PC 的哈希结果作为索引访问 BTB 以获得最终的预测目标地址.当该引导指令提交时,处理器使用实际目标地址更新对应的 BTB 项.

HVC 预测器通过 CIB 表项中 S 位实现"动态选择最佳关联信息"的机制.对一组关联的引导指令和间接跳转指令,处理器设置 *S* 位的方法为:

(1)在引导指令发射阶段设置 S 位为 1,

(2)在间接跳转指令的提交阶段设置 S 位为 0.

图 4 给出了一组引导指令和跳转指令的两次运行示例.通常情况下 S 位始终保持为 0,在引导指令发射(guide issue)后 S 位才会被设置为 1.在间接跳转指令取指(jmp fetch)时刻,若 S 位为 1(如 predict_1 所示),则说明该引导指令已经发射,此时选择单一数据值作为关联信息;若 S 位为 0(如 predict_2 所示),则说明引导指令尚未发射,最新关联数据值是不可用的,此时选择值历史作为关联信息.由于引导指令和间接跳转指令是成对出现且通常是按序提交,因此本文选择在间接跳转指令提交阶段(jmp commit)清除 S 位.

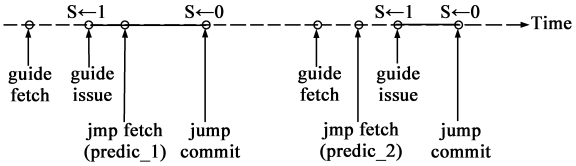

图4　使用S位实现"动态选择关联信息"的示例

4.4　硬件开销

HVC 预测器复用常规 BTB,避免了额外的大容量缓冲区,只需额外添加 CIB 结构和控制逻辑.CIB 结构包含 8 个表项,每表项由 32 比特 pc_tage、16 比特 singe_value、16 比特 value_history 和 1 比特 S 位组成,每项 65 比特,因此 CIB 总容量为 520 比特.HVC 预测器的控制逻辑主要包括 1 个加法器,2 个多路选择器,2 个比较器和 1 个异或器,如图 3 所示.总之,HVC 预测器所需额外硬件开销较小,易于在现代处理器中实现.

5　实验评估

本文采用 SimpleScalar[16] 模拟器,并集成 Wattch[17] 功耗模型,基准处理器的配置如参数如表 2 所示.评测程序集包括 8 个 SPEC INT 2000/2006 和 2 个 C++ 典型程序.在 SPEC INT 2000/2006 评测程序集中,本文仅选择间接跳转误预测性能损失大于 5%的典型程序.另外两个 C++ 典型程序为 Richards 和 ixx,其中 Richards 是一个模拟的操作系统内核任务调度器,ixx 是一个将 IDL 源程序转化为C++ 程序的转化器,它们反映了面向对象程序的间接跳转指令行为,被广泛用于间接跳转评测[1,2,15].对每个评测程序,使用 SimPoint[18] 工具选取由 100M 条指令构成的代表程序片段来实际运行.

表 2　基准处理器配置参数

参数	配置值
流水线	16 级,4 发射乱序执行
分支预测器	12KB混合预测器(8K 双模 PHT 项,8K 选择项,32K gshare PHT 项),4K 项 4-way BTB
执行逻辑	取指/译码/发射提交窗口宽度为 4, 512 项 ROB,128 项 LSQ
L1 I/D Cache	16KB,4-way,每行 64 字节,1 周期访问延时
L2 Cache	512KB,8-way,每行 64 字节,10 周期访问延时
Memory	150 周期平均访问延时

本文基于 GCC-4.2 添加了 HVC 预测器的编译支持,并基于基准处理器完成"难预测指令"的剖视分析.

5.1　性能

本文选取 Baseline-BTB 预测器[6] 和 VBBI 预测器[2] 作为对比方法.Baseline-BTB 预测器总是预测间接跳转指令的目标地址为该指令最近发生的跳转地址.VBBI[2] 预测器是最近提出的间接跳转预测器,它使用单一关联数据值作为关联信息引导间接跳转预测.同时,本文还评测了 perfect 预测器的性能,即完全没有间接跳转误预测情况下的理想性能.

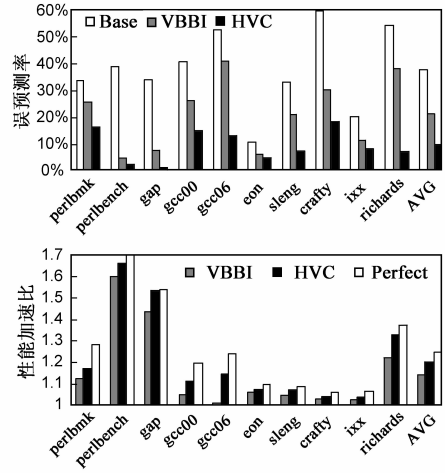

图5　各预测器的误预测率和性能

图 5 给出了各预测器的误预测率和相对于基准处理器的性能加速比.平均情况下,HVC 预测器的误预测率比 Baseline-BTB 预测器和 VBBI 预测器分别减少了 28%和 12%.更低的误预测率使得 HVC 预测器获得了显著的性能提升,平均比 Baseline-BTB 预测器性能提高 19%,比 VBBI 预测器性能提高 4%.评测结果表明,本文的混合预测机制通过在不同场景使用不同关联信息,能够有效提高关联信息的有效性,从而降低间接跳转误预测率并提高性能.

5.2　能耗

HVC 预测器通过降低误预测率,能够减少因预测失效而引起的流水线清空和无效执行,从而减小系统能耗.本文分别评测了 VBBI 预测器、HVC 预测器和 perfect 预测器节省系统能耗的情况,结果如图 6 所示.平均情况下,本文所提出的 HVC 预测器比基础处理器降低能耗 5.7%.比 VBBI 预测器降低能耗 2.4%.

6　结论

本文提出了一种根据不同应用场景动选择不同关

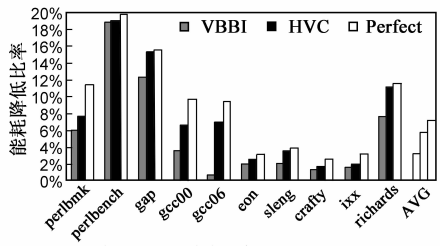

图6 各预测器降低系统能耗的比例

联信息,以引导间接跳转预测的混合型值关联间接跳转预测机制.该机制依赖于编译器根据高层次数据流信息识别关联数据值,并在运行时刻使用小容量缓冲区同时维护两种关联信息:单一数据值和值历史.在间接跳转指令取指阶段,处理器根据最新关联数据值的不同状态,从多种关联信息中动态选择最优关联信息来引导间接跳转预测.相对于传统的 BTB 预测器和最新的 VBBI 预测器,本文机制能够有效降低误预测率,提高处理器性能并降低能耗.

参考文献

[1] H Kim, et al. VPC prediction: reducing the cost of indirect branches via hardware-based dynamic devirtualization[A]. Int'l Symposium on Computer Architecture[C]. IEEE Computer Society, 2007. 424 – 435.

[2] M U Farooq, et al. Value based BTB indexing for indirect jump prediction[A]. Int'l symposium on High-Performance Computer Architecture[C]. New York: ACM, 2010. 1 – 11.

[3] 朱德新,程旭,慎辉. UNICORE 体系结构中动态转移预测机制的研究与设计[J]. 电子学报, 2004, 32(8): 1351 – 1355.
Zhu D X, Cheng X, Shen H. Dynamic branch prediction research and design for UNICORE architecture [J]. Acta Electronica Sinic, 2004, 32(8): 1351 – 1355. (in Chinese)

[4] M X Tan, et al. Energy-efficient branch prediction with compiler-guided history stack[A]. Int'l Conference on Design, Automation and Test in Europe[C]. 2012. 449 – 454.

[5] D R Kaeli and P G Emma. Branch history table prediction of moving target branches due to subroutine returns[A]. Int'l Symposium on Computer Architecture[C]. IEEE Computer Society, 1991. 34 – 42.

[6] J K F Lee and A J Smith. Branch prediction strategies and branch target buffer design[J]. IEEE Computer, 1984, 17(1): 6 – 22.

[7] Y Chang, et al. Target prediction for indirect jumps[A]. Int'l Symposium on Computer Architecture[C]. 1997. 274 – 283.

[8] S Gochman, et al. The Intel Pentium M processor: microarchitecture and performance[J]. Intel Technology Journal, 2003, 7(2): 21 – 36.

[9] 袁楠,范东睿. 高性能代价比的两层关联间接转移预测器设计[J]. 计算机学报, 2008, 31(11): 1898 – 1906.
Yuan N, Fan D R. Design of cost-effective 2-level correlation indirect branch prediction[J]. Chinese Journal of Computers, 2008, 31(11): 1898 – 1906. (in Chinese)

[10] K Driesen and U Hölzle. The cascaded predictor: economical and adaptive branch target prediction[A]. Int'l symposium on Microarchitecture[C]. IEEE Computer Society, 1998. 249 – 258.

[11] A Seznec and P Michaud, A case for (partially) TAgged GEometric history length predictors[J]. Journal of Instruction-Level Parallelism, 2006.

[12] T H Heil, et al. Improving branch predictors by correlating on data values[A]. Int'l symposium on Microarchitecture[C]. IEEE Computer Society, 1999. 28 – 37.

[13] L Chen, et al. Dynamic data dependence tracking and its application to branch prediction[A]. Int'l symposium on High-Performance Computer Architecture[C]. New York: ACM, 2003. 65 – 76.

[14] A Roth, et al. Improving virtual function call target prediction via dependence-based pre-computation[A]. Int'l Conference on Supercomputing[C]. New York: ACM, 1999. 356 – 364.

[15] J A Joao, et al. Improving the performance of object-oriented languages with dynamic predication of indirect jumps[A]. Int'l conference on Architectural Support for Programming Languages and Operating Systems[C]. New York: ACM, 2008. 80 – 90.

[16] T Austin, et al., SimpleScalar: an infrastructure for computer system modeling[J]. IEEE Computer, 2002. 35(2): 59 – 67.

[17] D Brooks, et al. Wattch: a framework for architectural-level power analysis and optimizations[A]. Int'l Symposium on Computer Architecture[C]. IEEE, 2000. 83 – 94.

[18] E Perelman, et al. Using SimPoint for accurate and efficient simulation[A]. Int'l conference on Measurement and Modeling of Computer Systems[C]. New York: ACM, 2003. 318 – 319.

作者简介

谭明星 男,1985 年生于湖南茶陵,北京大学信息科学技术学院博士研究生,主要研究方向为微处理器设计、软硬件协同设计和编译优化.
E-mail: tanmingxing@mprc.pku.edu.cn

刘先华(通讯作者) 男,1978 年生于湖北孝感,北京大学信息科学技术学院讲师,主要研究方向为计算机系统结构、编译优化.
E-mail: liuxianhua@mprc.pku.edu.cn