

基于同时多线程的 TBHBP 分支预测器研究

李静梅 关海洋

(哈尔滨工程大学计算机科学与技术学院 哈尔滨 150001)

摘 要 针对传统处理器分支预测器存在分支预测信息混乱、分支指令别名冲突和容量冲突率高的缺点,提出基于同时多线程处理器的分支预测器 TBHBP。该分支预测器采取线程历史信息与基于地址索引的局部历史信息相结合的综合历史信息作为模式匹配表 PHT 的索引,并采取线程独立拥有线程历史寄存器和分支历史寄存器的方式,通过新增分支结果输出表来提高指令的分支预测执行速度。研究结果表明, TBHBP 分支预测器有效解决了分支信息过时、分支指令别名和容量冲突的问题。与 Gshare 分支预测器相比,其指令吞吐率提升了 12.5%,分支误预测率和误预测路径取指率分别下降了 0.5%和 2.1%。

关键词 同时多线程, TBHBP 分支预测器, SMTSIM 仿真器, 分支预测信息, 别名冲突, 容量冲突

中图分类号 TP302 **文献标识码** A

Branch Predictor with TBHBP Based on Simultaneous Multithreaded Processors

LI Jing-mei GUAN Hai-yang

(College of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China)

Abstract Aiming at the shortages of high conflict of alias and capacity as well as prediction information disorder existed in the branch instruction, a new branch predictor TBHBP which is based on simultaneous multi-threading processor was proposed. In this branch predictor, the comprehensive historical information of the historical thread information and an index based on address information of the local history are taken as the pattern-matching table index of PHT. Though the way of history register is owned by separate thread and branch history register, the instruction execution speed of branch prediction is improved by adding the branch result output table. The researches results show that branch predictor of TBHBP can effectively improve the problems of out-date branch information, branch instruction aliases and capacity conflicts. Comparing with the Gshare, instruction throughput rises up to 12.5%, and branch prediction error rate and error forecast path rate can separately drop down by 0.5% and 2.1%.

Keywords Simultaneous multithreading, Branch predictor of TBHBP, Simulator of SMTSIM, Branch prediction information, Alias conflicts, Capacity conflicts

随着计算机体系结构的发展,为顺应人们对高性能处理器的迫切需求,同时多线程(Simultaneous Multithreading, SMT)处理器应运而生,并成为目前主流的微处理器体系结构^[1]。同时多线程处理器摒弃传统的物理设计技术,在不增加芯片面积的前提下,通过物理核映射多个逻辑核的方式,充分挖掘和利用指令级和线程级的并行性,使得处理器性能大幅度提高^[2]。同时多线程处理器为处理器性能的提升提供了良好的底层硬件架构,但是传统处理器的分支预测器存在分支命中率低下、误预测路径取指率过高等缺点,已无法满足现行的高密度计算任务的需求,严重影响了同时多线程处理器性能的充分发挥。本文在对同时多线程技术和现有分支预测器深入研究的基础上,针对传统分支预测器存在的不足之处,通过综合两级自适应分支预测、分支预测信息拼接和分支指令结果重用等技术的优点提出一种基于线程分支历史的 SMT 分支预测器;最后通过设计合理的性能测试方案,来验

证新型分支预测器的高效性和优越性。

1 同时多线程处理器

同时多线程处理器是一种微处理器体系结构,它结合了超标量处理器和多线程处理器的优点,通过将每个物理核映射为多个逻辑核的方式,使得每个线程在逻辑核的运行相当于在自身的物理核上运行一样,保证在一个时钟周期内可以有多个线程的多条指令同时运行,并充分挖掘指令级和线程级的并行性,极大地提高了处理器的指令吞吐率。

同时多线程处理器的硬件结构如图 1 所示,该结构是基于指令乱序执行的超标量处理器实现的^[3]。本文的研究都是基于这种硬件结构展开的。图 1 简要地介绍了同时多线程处理器的工作流程:在每个时钟周期内,取指单元根据程序计算器提供的多个线程的 PC 值,并在分支预测器的帮助下,获取多个线程的多条指令放入指令 Cache,之后再根据一定的取

到稿日期:2011-10-24 返修日期:2012-03-02 本文受国家自然科学基金(60873138, 61003036),黑龙江省自然科学基金(F201124)资助。

李静梅(1964—),女,博士,教授,主要研究方向为计算机系统结构、嵌入式系统、并行计算;关海洋(1987—),男,硕士,主要研究方向为多线程技术、并行计算, E-mail: haiyang_guan@163.com。

指策略从指令 Cache 读取一定数目的指令放入指令队列后进行译码和重命名操作,将处理过的指令分别放入定点/浮点发射队列进行等待,将队列中满足发射条件的指令乱序发射到对应的功能部件去执行,最后将执行完毕的指令再经过指令重排序部件后顺序将结果写回到对应的定点/浮点寄存器组^[4]。

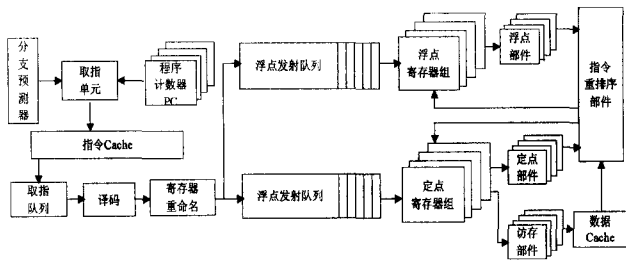


图1 同时多线程处理器的硬件结构图

2 现有分支预测器分析

两级自适应分支预测器是目前使用较为广泛的分支预测器。现行的很多分支预测器都是在其基础上进行改进的,它取代了通过剖析程序积累统计数据的方式,通过更新历史寄存器的内容及依据分支结果模式表目项中的模式历史位,来收集作为分支预测基础的执行历史信息^[5]。在两级自适应分支预测器中有两个最主要的部件:分支历史寄存器 BHR (Branch History Register)和模式历史表 PHT (Pattern History Table)。BHR 用于记录最近 n 个分支的分支结果,并可通过移位方式完成对分支历史信息的更新操作;模式历史表 PHT 用于存储分支历史寄存器中特定模式的最近分支结果,大都是通过两位饱和计数器实现的。两级自适应分支预测实现的基本原理:通过利用分支历史在时间上的局部性,应用一个 n 位的分支历史寄存器 BHR 去索引模式历史表 PHT,最终获得分支指令的预测结果,并根据预测结果更新分支历史表用于后继分支指令的预测。

此外,根据历史寄存器中存储内容的不同,两级自适应分支预测器还可分为全局历史分支预测器和局部历史分支预测器^[6]。在全局历史分支预测器中,分支历史寄存器记录前 n 次的历史预测分支。而在局部分支预测器中,分支历史寄存器则储存每个分支指令的目标地址。对于含有大量 if-else 分支的负载程序而言,全局分支预测效果较好。而对于测试程序中含有大量 loop 分支的情况,局部分支预测的效果表现较好,但由于每次循环分支预测都需一个 BHT (Branch History Table),因此分支预测表的硬件开销也较大。

本文的 SMT 分支预测器基于两级自适应分支预测器进行了改进,并综合了全局分支预测和局部分支预测的优点。设计的具体流程将在第3节进行详细介绍。

目前,基于分支历史的分支预测器还有很多种类型。Gshare 和 Gselect 是传统处理器应用较多的预测器。Gshare 预测器通过将分支地址高位和分支历史低位做异或运算,并将运算结果通过哈希函数散列寻址的方式完成模式匹配表 PHT 的索引操作,最终实现对分支指令的预测^[7]。Gselect 预测器通过分支地址高位和分支历史低位的拼接索引模式匹配表 PHT 进行分支的预测^[8]。两者在一定程度上都有效减

少了分支别名干扰现象的发生,其分支预测性能基本相当。此外,还有一些分支预测性能较好但硬件实现复杂度过高的分支预测器被提出,如 Filter、YAGS 等。

同时,针对现行处理器结构的特点,还有一些新型的基于值预测器和基于神经网络的分支预测器被提出,如 VBBP、O-GEHL 等^[9,10]。

3 TBHBP 分支预测器设计

本节在对上述分支预测器深入分析的基础上,综合考虑同时多线程处理结构的特点,提出一种基于线程分支历史信息的同时多线程分支预测器,即 TBHBP (Thread Branch History Branch Predict)。该分支预测器通过采用线程分支历史和分支指令目标地址信息拼接的方式完成对 PHT 的索引操作,并采用分支指令结果的重用技术,提高了分支指令的预测效率。同时,线程还采用独立的分支历史寄存器和分支指令目标地址寄存器,缓解线程对分支预测表等系统资源的竞争和分支别名冲突现象的发生,有效地提高了处理器的分支预测率和系统资源的利用率。

3.1 TBHBP 分支预测器硬件结构

TBHBP 分支预测器以两级自适应分支预测器为基础,线程间采用独立的分支历史寄存器和地址历史寄存器,对于模式匹配表则采取线程共享的方式,并应用分支结果输出表对分支指令结果进行存储。其具体的硬件结构如图2所示。

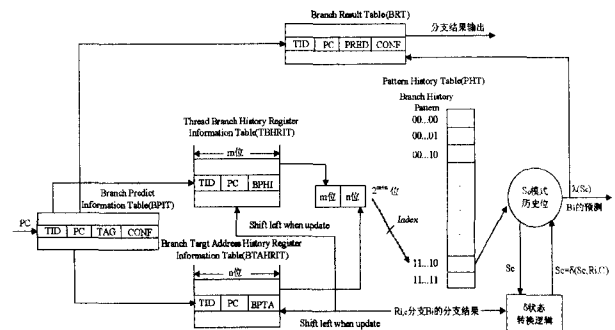


图2 TBHBP 分支预测器的硬件结构图

由图2可见,TBHBP 分支预测器主要包含6个部分:分支预测信息表 BPIT (Branch Predict Information Table)、线程分支历史寄存器信息表 TBHRIT (Thread Branch History Register Information Table)、分支目标地址历史寄存器信息表 BTAHRIT (Branch Target Address History Register Information Table)、模式匹配表 PHT (Pattern History Table)、分支结果输出表 BRT (Branch Result Table),以及除分支预测信息表和 PHT 表外,其他3个表的逻辑更新电路。

分支预测信息表 BPIT 是根据分支指令的 PC 值进行索引的,每个线程对应一组独立的表目项,每一个表目项包含4个字段:TID 字段为线程的索引号用于索引 TBHRIT 表;PC 字段用于 BTAHRIT 表的索引;TAG 字段用于与分支结果表的 PC 值进行比较,以确定该分支指令是否需要分支预测;CONF 字段作为分支预测的阈值,以判断是否应用 BRT 表中的分支预测结果。当线程的某一分支指令进入流水线后,分支预测电路用其 PC 值索引 BPIT 表的某一表目项。

线程分支历史寄存器信息表 TBHRIT 是通过 BPIT 表中

的 TAG 字段进行索引的,每个线程对应一组独立的表目项,每一个表目项包含 3 个字段:TID 字段为线程的索引号;PC 字段用于对分支指令的索引;BPFI 字段为预测的分支历史信息,用于分支历史信息位的拼接,并在分支指令提交时进行更新。当线程的某一指令进入流水线的译码阶段时,分支预测电路用其 PC 值索引 TBHRIT 表的某一表目项。

分支目标地址历史寄存器信息表 BTAHRIT 是通过 BPIT 表中的 PC 字段进行索引的,每个线程对应一组独立的表目项,每一个表目项包含 3 个字段:TID 字段为线程的索引号;PC 字段用于对分支指令的索引;BPTA 字段为分支指令的目标地址信息,用于读取每个分支指令目标地址的局部分支历史信息,之后完成分支历史信息位的拼接,并在分支指令提交时进行更新。当线程的某一分支指令进入流水线的译码阶段时,分支预测电路用其 PC 值索引 BTAHRIT 表的某一表目项。

模式匹配表 PHT 是通过线程的综合分支历史进行索引的,综合历史信息是由 TBHRIT 表记录的线程分支历史信息和 BTAHRIT 表中依据分支指令目标地址读取的局部分支历史信息拼接而成的,对 PHT 表的应用采用多线程共享的方式。

分支结果输出表 BRT 是通过 BPIT 表中的 PC 字段进行索引的,每个线程对应一组独立的表目项,每一个表目项包含 4 个字段:TID 字段为线程的索引号;PC 字段用于对分支指令的索引;PRED 用于存储分支指令的预测结果;CONF 作为分支预测的阈值。当线程的某一分支指令进入流水线的写回阶段时,分支结果更新电路用其 PC 值索引 BRT 表的所有表目项,完成对其的更新操作。同时,对 TBHRIT 表和 BTAHRIT 表也将应用相关的更新电路完成表目项的更新操作。

3.2 TBHBP 分支预测器实现流程

本节针对图 2 所示的 TBHBP 分支预测器硬件结构,对其具体实现的工作流程进行详细介绍。

假设在处理器系统初启动时,分支预测器需对一分支指令 B_i 进行分支预测。具体的工作流程如下:

步骤 1 根据分支指令 B_i 的 PC 值索引分支预测信息表 BPIT,读取分支指令所属的线程索引号 TID。

步骤 2 通过获取的 TID 编号索引线程分支历史寄存器信息表 TBHRIT,读取线程预测的分支历史信息 BPFI,将其作为分支预测的全局历史信息。同时,通过获取的指令 PC 值索引分支目标地址历史寄存器信息表 BTAHRIT,读取分支指令的目标地址 BPTA,并根据指令地址读取用于分支预测的局部历史信息。

步骤 3 每个线程的分支历史信息 BHR 和根据目标地址读取的历史信息 BHT 通过哈希函数结合在一起,作为二级模式匹配表 PHT 的索引。在这里,分别对 $BHR \oplus BHT$ 、 $BHR+BHT$ 和 $BHT+BHR$ 3 种历史信息的连接方式进行分支预测性能的测试,以确定两种历史信息最佳的连接方式。实验通过运行 art-perlbnk、craft-mcf 和 bip2-lucas 3 组两线程工作负载程序,分析在不同的连接方式下分支误预测率和分支误预测路径取指数。具体分析结果如表 1 所列。

表 1 不同历史信息连接方式的分支预测性能对比图

基准测试程序	信息连接方式	分支误预测率	分支误预测路径取指数
art-perlbnk	BHR+BHT	2.8%	14.3%
	BHT+BHR	3.0%	18.5%
	$BHR \oplus BHT$	3.3%	20.6%
craft-mcf	BHR+BHT	3.2%	7.5%
	BHT+BHR	3.5%	10.2%
	$BHR \oplus BHT$	3.7%	12.3%
bip2-lucas	BHR+BHT	1.8%	2.4%
	BHT+BHR	2.0%	2.9%
	$BHR \oplus BHT$	2.4%	3.5%

由表 1 分析可知,BHR+BHT 相比于其他两种历史信息的连接方式,在分支预测性能上有一定的优势。因此,对于二级模式匹配表 PHT 的索引,本文采取线程历史信息和地址历史信息拼接的方式

步骤 4 通过拼接的历史信息索引 PHT 表获取分支指令的模式历史位 S_c ,用于实际的分支预测操作。

步骤 5 将获取的模式历史位 S_c 输入到预测决定函数,即式(1)中完成分支预测结果的计算操作。同时,通过状态转换函数 δ 完成模式历史位的更新操作,更新后的模式历史位将由原来的 $R_{i,c-k}, R_{i,c-k+1}, \dots, R_{i,c-1}$ 变为 $R_{i,c-k+1}, R_{i,c-k+2}, \dots, R_{i,c}$ 。

$$Z_c = \lambda(S_c) \quad (1)$$

步骤 6 将分支指令 B_i 的预测结果写入分支结果输出表 BRT 中。当下次有相同的分支指令被预测时,若预测结果与 BRT 表中的 PRED 值相同,则 CONF 加 1;反之,CONF 做减 1 操作。

步骤 7 通过 TBHRIT 表的更新电路,将获取的分支输出结果 $R_{i,c}$ 左移入线程历史寄存器中的最末位上,并将预测的历史信息更新为分支指令提交的历史信息。

步骤 8 通过 BTAHRIT 表的更新电路,将获取的分支输出结果 $R_{i,c}$ 所对应的目标地址历史信息左移入地址历史寄存器中的最末位上,并将预测的分支指令目标地址更新为分支指令提交时的实际地址信息。

步骤 9 当分支预测器对下一分支指令 B_{i+1} 进行分支预测时,首先根据其 PC 值索引 BRT 表中的 CONF 字段。若 CONF 大于等于 2,则 BPIT 表中的 TAG 字段记为 1,分支预测电路将不对指令 B_{i+1} 进行分支预测操作,而是直接将存储的分支结果输出。反之,若 CONF 小于 2,则 BPIT 表中的 TAG 字段记为 0,分支指令将按照上述的 8 个步骤重新执行分支预测操作,并将预测结果与 BRT 表中的数据进行比对,完成 CONF 字段和 PRED 字段的更新操作。

此外,若在分支预测的整个过程中出现分支误预测现象,则将随时停止剩余操作,并取消误预测分支指令后所属同一线程的正在流水线中运行的全部指令,线程的 PC 值调整为分支后的正确目标指令地址,然后从新的地址重新开始读取指令执行。同时,根据分支的实际执行结果调整分支结果输出表 BRT 中相应表目项的 CONF 字段和 PRED 字段,以供该分支指令再次执行时使用。

3.3 TBHBP 分支预测器功能分析

在同时多线程处理器中,由于在一个时钟周期内有多个线程并行地乱序取指和执行,使得线程中指令的读取是乱序进行的。而处理器的分支预测器也在这一阶段工作,那么对指令的分支预测执行也将是乱序的,预测使用的历史信息将不会按照程序记录顺序地去读取,导致 SMT 处理器中分支预测信息混乱、不完整等问题的出现,直接影响处理器分支预

测准确率提高。可见,分支预测信息的组织方式对 SMT 处理器分支预测性能的提升将产生直接的影响。

TBHP 分支预测器通过采取线程历史信息并根据地址索引的局部历史信息相结合的综合历史信息作为模式匹配表 PHT 的索引,有效地减少了 SMT 处理器中分支信息过时、混乱现象的发生,进而提升了处理器的分支预测性能。

同时,由于 SMT 处理器一个时钟周期同时从多个线程读取多条指令,使得取到的分支指令数目也相应地增多,加剧了分支指令对分支预测表等系统资源的竞争,使得分支别名冲突和容量冲突的现象更为严重。

在 TBHP 分支预测器中,线程独立拥有线程历史寄存器和分支历史寄存器,减小了分支指令发生别名冲突和容量冲突的可能性,而模式匹配表 PHT 则采取线程共享的方式,减小了处理器的硬件开销。同时,通过增加分支结果输出表,加速分支指令的预测执行速度,避免分支指令在指令队列中出现堆积,影响后继指令的顺利执行,促进处理器指令吞吐率性能的提升。

传统分支预测器根据线程号查询分支指令的全局历史信息,通过指令的目标地址检索分支指令的局部历史信息,之后对采集的两种历史信息分别进行分支预测,再通过一定的选择机制确定最后的预测结果。TBHP 预测器通过两种历史信息拼接的方式进行分支预测,保证分支指令所对应历史模式的唯一性。在整个分支预测过程中仅需一个 PHT 表,同时不增加选择器等物理部件,有效地降低了分支预测算法的复杂度。

4 TBHP 分支预测器性能测试

本节在对同时多线程处理器结构和 TBHP 分支预测器的特点深入研究的基础上,设计合理的性能测试方案,并深入分析 TBHP 预测器的性能测试结果,最终在测试结果数据分析的基础上,验证了 TBHP 预测器在分支预测性能上的高效性和合理性。

4.1 性能测试方案设计

TBHP 分支预测器的性能测试方案主要包括以下 4 个方面:模拟平台搭建、基准测试程序的选择、性能参照对象的选择和性能测试参数设计。本节分别从这 4 个方面对分支预测器的性能测试方案进行具体介绍。

4.1.1 模拟平台搭建

表 2 SMTSIM 模拟器参数的基本配置表

参数	值
功能单元	浮点:3 定点:6(包含 4 个访存部件)
流水线	9 级
取指/译码带宽	8 条指令/时钟周期
指令队列	定点:64 浮点:64
重命名寄存器	定点:32 浮点:32
L1 I-Cache	64kB 2 路组相连 64 字节/行
L1D-Cache	64kB 2 路组相连 64 字节/行
L2 Cache	512kB 2 路组相连 64 字节/行
L3 Cache	4MB 2 路组相连 64 字节/行
ITLB	48 项
DTLB	128 项
CPU 访问延迟	L1I-Cache:1,L1D-Cache:1,L2 Cache:10,L3 Cache:20 主存:100
分支预测器	2k gshare 误预测开销为 3 个时钟周期
BTB	256kB 2 路组相连

本文采用西部实验室 Dean. M. Tullsen 等人研发的 SMTSIM 模拟器进行实验研究。SMTSIM 模拟器是基于 James Lames 编写的 SPIM 模拟器进行开发的,可同时运行 8 个线程,每个线程运行的指令可以达到 300M。同时,SMTSIM 模拟器还支持 Alpha 可执行程序运行,且运行速度也是目前 SMT 模拟器中最快的^[11]。模拟器参数的基本配置如表 2 所列。

4.1.2 基准测试程序选择

实验选取 SPEC 2000 测试集中的 7 个定点程序和 5 个浮点程序,并将其随机组合为 6 个双线程负载集进行性能测评。同时,在实验中完整的模拟测试程序需要花费大量的时间,有时甚至不可能完成,因此针对不同测试程序的运行指令数也进行具体的配置。具体的测试程序参数及运行指令数配置如表 3 所列,运行指令数的单位为十亿。

表 3 TBHP 分支预测器性能测试基准程序参数配置表

测试程序	数据类型	负载类型	运行指令数
gzip, swim	int, float	ILP, Mem	1.5
bzip2, lucas	int, float	Mem, Mem	3.3
mcf, crafty	int, int	Mem, ILP	1
parser, twolf	int, int	Mem, Mem	3.3
perlbnk, art	int, float	ILP, Mem	0.3
appl, sixtrack	float, float	Mem, Mem	2

4.1.3 性能参照对象

性能参照采用目前分支预测性能较好的 Gshare 分支预测器与 TBHP 分支预测器进行性能比对。通过与高性能的预测器进行性能对比,更能显现 TBHP 分支预测器在分支预测性能上的优越性和可用性。

4.1.4 性能测试参数

性能测试实验采用的评估参数包括:处理器和线程 IPC、分支误预测率及分支误预测路径取指率。

处理器的 IPC 值是指处理器在每个时钟周期内执行的指令数,是衡量处理器指令吞吐率和加速比的重要性能指标。

分支误预测率是指误预测的分支指令数与分支指令总数的比值。分支误预测路径的取指率是指误预测路径读取的指令数与读取指令总数的比值。

4.2 性能测试结果分析

本节通过采取与 Gshare 分支预测器进行性能比对的方式,分别从处理器 IPC、线程 IPC、分支误预测率和误预测路径取指率 4 个方面对 TBHP 分支预测器进行性能测试;并将获得的结果数据进行统计形成性能结果对比图,对分支预测器在各方面的性能进行分析和评估。

同时,为了使得测试环境更为接近实际程序运行状态,本节实验采用 12 个工作负载程序两两随机组合的方式,最终形成 6 个复合型测试程序对分支预测器进行性能测试。具体的性能测试结果如图 3 所示。

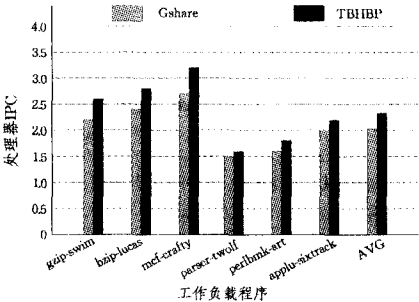


图 3 处理器 IPC 性能测试对比

由图 3 分析可知,相比于 Gshare 预测器,在 TBHBP 预测器下处理器的指令吞吐率均呈现不同程度的涨幅。在两线程负载运行的条件下处理器的 IPC 性能达到 2.33,而在 Gshare 预测器下处理器 IPC 的性能仅为 2.07,工作负载程序性能的平均加速比约为 12.5%。由本文第 3.3 节对 TBHBP 预测器的功能分析可知,处理器指令吞吐率性能的提升主要是由于在 TBHBP 预测器中,线程历史寄存器和分支历史寄存器为线程独立拥有,有效减少了分支别名冲突和容量冲突,提高了处理器的取指质量。同时,通过增加分支结果输出表有效地提高了分支指令的预测执行速度。在两种因素共同的推动下,处理器的指令吞吐率性能提升显著。

同时,由于单线程独立拥有分支预测表等系统资源,有效地减弱了线程对资源的竞争程度。因此,在处理器整体性能提升的同时,各单线程指令吞吐率的性能也呈现上升趋势。下面针对上一实验 6 个两线程工作负载中的 12 个单线程的 IPC 性能进行测试,具体的性能测试结果如图 4 所示。

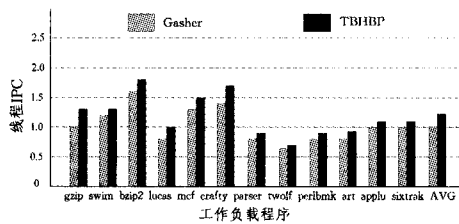


图 4 单线程 IPC 性能测试对比

分析图 4 可知,与 Gshare 预测器相比,在 TBHBP 预测器下的 12 个工作负载程序的指令吞吐率均有一定的提升。性能测试结果经过统计后显示,TBHBP 预测器下单线程的平均 IPC 值为 1.16,而 Gshare 预测器仅为 1.03。可见,TBHBP 预测器具有一定的公平性,对于单线程指令吞吐率性能的提升起到了一定的促进作用。

TBHBP 预测器的 PHT 表索引采取线程历史信息 and 地址历史信息相结合的方式,有效地缓解了同时多线程处理器中因分支指令的乱序执行导致分支历史信息混乱、不完整等现象的发生,提高了处理器分支历史信息的准确性。同时,线程独立拥有分支预测资源的模式,降低了分支别名冲突和容量冲突发生的概率,增强了分支预测结果的正确性和可用性。分支结果输出表 BRT 通过记录常用分支指令的分支结果,加速了处理器分支预测的执行速度。综合以上因素,在 TBHBP 预测器下处理器的分支误预测率性能较传统的 Gshare 应当有所下降。具体的性能测试结果如图 5 所示。

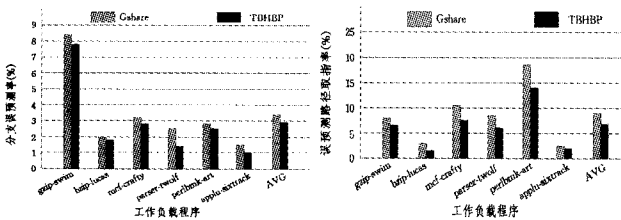


图 5 分支误预测率性能测试对比

由图 5 分析可知,Gshare 预测器下的分支误预测率为 3.42%,而 TBHBP 预测器下的分支误预测率仅为 2.93%,降低了近 0.5 个百分点。尽管误预测率的下降幅度并不是很大,但处理器的总体指令吞吐率性能涨幅约为 12.5 个百分点,可见处理器分支预测性能的改善对于指令吞吐率性能的

提升显得尤为重要。

同时,分支误预测率的下降有效地减少了取指部件在误预测路径上的取指数,处理器在误预测路径上的取指率也相应地下降。具体的性能测试结果如图 6 所示。

由图 6 分析可知,Gshare 预测器下的误预测路径取指率为 8.41%,而 TBHBP 预测器下的误预测路径取指率仅为 6.28%,降低了近 2.1 个百分点。分支误预测率和误预测路径取指率的下降有效地提高了处理器的分支预测性能,并促进了处理器指令吞吐率性能的提升。

结束语 TBHBP 分支预测器通过采取线程历史信息和根据地址索引的局部历史信息相结合的综合历史信息作为模式匹配表 PHT 的索引,有效地减少了 SMT 处理器中分支信息过时、混乱现象的发生。线程独立拥有线程历史寄存器和分支历史寄存器,减小了分支指令发生别名冲突和容量冲突的可能性,而模式匹配表 PHT 则采取线程共享的方式,减小了处理器的硬件开销。分支结果输出表的增加,避免了分支指令在指令队列中出现堆积,加速了指令的分支预测执行速度。

实验结果表明,TBHBP 分支预测器有效地提高了处理器的分支预测性能,且在一定程度上促进了处理器指令吞吐率性能的提升。由此可见,TBHBP 分支预测器的应用对同时多线程处理器的研究具有重要的指导意义和研究价值。

参考文献

- [1] 李祖松,许先强,胡伟武. 龙芯 2 号处理器的同时多线程设计[J]. 计算机学报,2009,32(11):2265-2273
- [2] 路放,安虹,梁博,等. OpenSMT:一个同时多线程处理器模拟器的设计和实现[J]. 计算机科学,2006,33(1):158-163
- [3] Tullsen D M, Eggers S J, Emer J. Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor[C]//23rd Annual International Symposium on Computer Architecture. Philadelphia, American, 1996:191-202
- [4] 刘权胜,杨洪斌,吴悦. 同时多线程技术[J]. 计算机工程与设计,2008,29(04):963-967
- [5] Yeh T-Y, Patt Y N. Alternative Implementations of Two-Level Adaptive Branch Prediction[C]//The 1992 ACM. Michigan, American, 1992:124-134
- [6] 陈恺冬,史江一,郝跃. 一种基于双模结构的预测器[J]. 微电子学与计算机,2008,25(5):110-113
- [7] Sendag R, Yi J J, Chuang Peng-fei. Low Power/Area Prediction Using Complimentary Branch Predictors [C]// Proceedings of the 2008 IEEE international parallel & distributed processing symposium. Minneapolis, American, 2008:1-12
- [8] Fast Design Space Exploration Using Local Regression Modeling With Application to ASIPs[J]. IEEE transactions on computer-aided design of integrated circuits and systems, 2009, 27(3):508-515
- [9] 何立强,刘志勇. 一种有效的同时多线程处理器取指控制机制[J]. 计算机学报,2006,29(4):535-543
- [10] Hallschmid P, British C U, et al. Seznec A. Analysis of the O-GEometric History Length branch predictor[C]// Proceedings of the 32nd International Symposium on Computer Architecture (ISCA'05). Washington, DC, 2005:1-12
- [11] 潘治. 同时多线程模拟研究现状[C]//高性能计算应用大会. ShangHai, China, 2005:68-72