处理器值预测技术研究

黄立波[1,2],杨 凌[1,2],杨乾明[1,2],马 胜[1],王永文[1,2],隋兵才[1,2],沈 立[1],徐炜遐[1]
(1. 国防科技大学计算机学院,湖南长沙 410073;2. 先进微处理器芯片与系统重点实验室,湖南长沙 410073)

摘 要: 当今的处理器性能与存储器带宽和延迟严重失衡的问题限制了计算系统的整体性能,而存储器的性能对制程工艺不敏感,在后摩尔时代下很难再通过集成电路制造工艺的迭代获得处理器性能收益,因此人们更多地想通过体系结构的创新获得更高性能的计算系统. 处理器值预测技术是一种能在无需改变存储系统情况下有效缓解存储墙问题的解决方案,其通过预测性地打破数据真相关进而让更多的指令可以在乱序处理器中并行执行,而无需等待由于访存等操作造成的长周期指令执行. 近年来,值预测在各个方面都有了实质性的进步,但现如今还没有商用处理器使用这一技术,这主要是由于值预测技术的使用还面临许多挑战:现有的处理器的流水线架构不能直接使用值预测技术;值预测所需的预测值传递机制需要额外的硬件资源开销;值预测器巨大的存储开销让其很难在片上实现;由于值预测错误时的性能惩罚大,因此预测准确率较低的值预测器会降低处理器性能. 针对这些问题,本文以值预测技术为中心,围绕值预测技术相关的流水线架构、值预测器结构和错误恢复机制三个方面分别详细论述了国内外研究成果以及其对于各个问题挑战的解决策略. 最后,本文对当今的处理器值预测技术进行了总结并对未来的研究方向进行了展望.

关键词: 值预测;数据依赖;处理器;流水线;存储墙;超标量

基金项目: 国家重点研发计划(No.2021YFB0300300);国家自然科学基金(No.62272475,No.62172430);湖南省自然科学基金(No.2022JJ10064,No.2021JJ10052);湖南省科技创新计划项目(No.2022RC3065)

中图分类号: TP301 **文献标识码:** A **文章编号:** 0372-2112(2023)12-3591-28

电子学报 URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20230206

Research on Processor Value Prediction

HUANG Li-bo[1,2], YANG Ling[1,2], YANG Qian-ming[1,2], MA Sheng[1], WANG Yong-wen[1,2],
SUI Bing-cai[1,2], SHEN Li[1], XU Wei-xia[1]
(1. *College of Computer Science and Technology, National University of Defense Technology, Changsha, Hunan* 410073, *China*;
2. *Key Laboratory of Advanced Microprocessor Chips and Systems, Changsha, Hunan* 410073, *China*)

Abstract: The extreme imbalance between processor performance and memory bandwidth/latency limits the overall performance of computing systems. In the post-Moore era, it is challenging to obtain processor performance benefits through the iteration of the integrated circuit manufacturing process, and memory performance is not sensitive to the process. Therefore, people tend to obtain higher-performance computing systems through architectural innovation. Processor value prediction technology is a solution that can effectively alleviate the memory wall problem without changing the storage system. By speculatively breaking the true dependency of data, more instructions can be executed in parallel in an out-of-order processor. There is no need to wait for the execution of long-cycle instructions caused by memory access, etc. In recent years, value prediction has made significant progress in various aspects. However, no commercial processors are using this technology, mainly because the development of value prediction technology still faces many challenges: the pipeline architecture of existing processors cannot directly use value prediction techniques; the register file read and write ports required for value prediction are physically challenging to implement; the huge storage overhead of the value predictor makes it difficult to implement on-chip; due to the significant performance loss when the value prediction is wrong, the value predictors with low prediction accuracy will reduce processor performance. In response to these problems, this paper focuses on value prediction technology. It discusses in detail the research at home and abroad and its solutions to problems and challenges around the value prediction pipeline architecture, value predictor structure, and misprediction recovery mech-

收稿日期:2023-03-06;修回日期:2023-09-14;责任编辑:宋甲英

anism related to value prediction technology. Finally, this paper summarizes processor value prediction techniques and provides an outlook on future research directions.

Key words: value prediction; data dependency; processor; pipeline; memory wall; superscalar

Foundation Item(s): National Key R&D Program of China (No.2021YFB0300300); National Natural Science Foundation of China (No.62272475, No.62172430); Hunan Provincial Natural Science Foundation of China (No.2022JJ10064, No.2021JJ10052); The Science and Technology Innovation Program of Hunan Province (No.2022RC3065)

1 引言

存储墙(memory wall)[1~4]是当今计算机体系结构领域所共同面对的一个难题,存储器缓慢增长的带宽和延迟制约着计算机系统的整体性能[2]. 由于集成电路制造工艺的迭代难以获得更高性能的存储器,处理器性能与存储器带宽和延迟之间的失衡问题直接导致处理器中的访存指令占用大量时间,指令之间的数据相关性使后续指令必须等待访存指令执行完毕才能执行,这严重限制了处理器的性能.

为了提高指令级并行度,超标量处理器不断增加取指宽度并扩大指令队列,以寻找更多的数据无关指令来执行. 然而,相关性是计算机程序固有的属性,因此这种方式无法无限扩展处理器的性能[5]. 目前,包括英特尔的Skylake[6]、苹果的M1[7]在内的高性能超标量处理器大多采用四发射/八发射策略,即一个周期最多能同时发射执行四条/八条指令. 以四发射的龙芯2B处理器[8]为例,其是中国科学院计算技术研究所研发的高性能通用处理器,采用7级主流水线的设计,配备32 KB的一级指令缓存和数据缓存. 性能分析实验[9]说明,龙芯2B在SPEC 2000测试基准下,其平均每周期指令数(Instructions Per Cycle,IPC)性能仅为0.6,远低于其流水线的理想IPC=4.0的性能. 同时,在分支预测和访存都永远命中等一系列理想条件下,龙芯2B在整数SPEC 2000下的IPC性能没有超过2.5,浮点SPEC 2000的IPC性能也没有超过2.0. 此外,八发射的Sonic BOOM处理器[10],是一款加州伯克利大学基于RV64GC指令集架构设计实现的开源处理器,常用于科学研究,在配备先进的TAGE[11]分支预测器,指令缓存带宽16字节/周期,取指宽度为4,译码宽度为4的条件下,其Dhrystone测试性能(3.93 DMIPS/MHz)远超过了之前的BOOMv2(1.91 DMIPS/MHz),但其SPEC 2006的实际IPC性能仅为0.86. 而在SPEC 2017测试基准下的平均IPC为0.94,在相同的SPEC 2017测试条件下,六发射的基于x86架构的英特尔Skylake处理器[6]的平均IPC性能为1.32[10],仅为其流水线理想IPC性能的22%. 与之相比,十六发射及以上的策略在消耗指数级增长的硬件资源的同时,获得的额外性能收益越来越少. 例如,在2018年CVP(Championship Value Prediction)值预测竞赛[12]中,基于踪迹驱动的16取指处理器性能模型(32 KB一级缓存、1 MB二级缓存、8 MB三级缓存)在多种基准下测试的平均IPC性能为2.779,仅为流水线的理想IPC性能的17%. 因此,迫切需要新的计算原理和微体系结构来推动超标量处理器的进一步发展.

处理器值预测(Value Prediction,VP)技术是一种提高处理器指令级并行度的微体系结构技术. 20世纪90年代,Lipasti等人[13,14]和Gabbay等人[15]和Sazeides等人[16]同时提出了该技术,旨在进一步提高通用处理器的性能. 作为一种对软件透明的微体系结构技术,它可以预测性地打破指令之间的数据依赖关系,隐藏存储墙问题导致的访存延迟,从而增加乱序处理器的指令级并行度. 值预测技术的提出引起了广泛关注与研究[13~16]. 尽管经过长时间的发展,产生了许多学术研究成果,但目前还没有商用处理器采用这项技术. 主要原因有三点:一是值预测技术所需的硬件资源较高,值预测器本身需要大量的片上存储资源;二是需要修改现有的流水线架构,这将带来额外的资源开销以及研发成本,并对处理器的物理实现提出更大挑战;三是错误的值预测可能会降低处理器性能,因此在预测器性能较低的情况下,反而会减少处理器性能,这给值预测的应用带来了一定的不确定性.

近年来,针对值预测技术的研究逐渐陷入瓶颈,业界难以承担值预测带来的巨大资源开销,因此相关科学研究成果也逐渐减少. 然而,随着TAGE[11]预测技术的引入和CVP值预测竞赛的成功举办,值预测研究再度引起热潮,值预测器的性能被提升到了一个新的高度. CVP值预测竞赛于2018年在美国洛杉矶与ACM/IEEE ISCA会议同期举行,共包含三个赛道,分别是8 KB和32 KB的两个有限资源赛道以及一个无限资源赛道. CVP值预测竞赛采用统一的基于踪迹驱动的处理器性能评估框架,其面向值预测器的设计进行了功能抽象,参赛者仅需在特定的三个功能函数(预测函数、预测性更新函数、提交更新函数)内完成相应的算法设计即可提交作品,且所有源代码、文章和性能评估结果都将开源. 为了促进值预测技术的研究发展,CVP值预测竞赛平台一直处于在线状态(https://microarch.org/cvp1/cvp1online/contestants.html),在后续的几年中陆续有新的值预测算法刷新排行榜单. 以CVP值预测竞赛公布的数据为例,目前最佳表现的EVES(En-

hanced VTAGE Enhanced Stride)值预测器在32 KB资源配置下,平均IPC性能提升达到28.6%[17]. 值预测技术还有巨大的性能增长潜力,同样以CVP值预测竞赛公布的数据为例,在理想情况下,即值预测器能够完全准确地预测所有候选指令,单核处理器的IPC性能最大可提升107%. 这对学术界和工业界具有巨大吸引力,因为任何单核处理器的性能提升都能直接推动当今多核多线程高性能计算系统的整体性能增长. 在最近一段时间,Intel[18,19]、高通[20]、ARM[21]、华为[22]等业界巨头在值预测领域取得了突破,值预测技术再次成为业界的研究重点.

值预测对超标量处理器的发展具有重要意义,通过值预测能够充分利用超标量处理器的硬件计算资源,推动超标量处理器的发展,使处理器的实际性能接近理想性能. 目前国内针对值预测技术的研究较少[23~28],缺乏系统性的总结与分析. 为了准确理解值预测技术及其发展脉络和趋势,本文首先对值预测技术进行概述,然后结合国内外研究成果,从值预测流水线架构、值预测器结构和值预测错误恢复机制三个方面进行详细综述,并展望未来可能的研究方向.

2　值预测原理概述

2.1　数据真相关

值预测的首要问题是解决指令之间的数据依赖,这也被称为数据真相关或读后写相关(Read After Write,RAW)[29]. 当出现以下情况之一时,我们称指令j对指令i存在数据依赖:

(1) 指令i的数值结果会被指令j所使用;

(2) 指令j数据依赖指令k,而指令k数据依赖指令i,即数据依赖具有传递性.

当具有数据依赖关系的指令在流水线中执行时,假设指令j依赖指令i的结果,那么指令j必须等待指令i执行完毕后才能利用指令i的结果开始执行. 后续所有依赖指令i的指令都必须等待指令i的执行. 特别是在超标量处理器中,流水线的吞吐率较高,因此更容易遇到具有数据依赖关系的指令. 由于数据依赖具有传递性,因此存储墙问题导致的长时间访存指令,往往会导致后续所有指令都依赖访存指令的情况. 这将造成流水线停顿,严重影响处理器的运行性能. 此外,对于一些具有较长执行时间的指令,例如浮点计算,也具有类似的影响.

2.2　值预测原理

值预测通过预测将写入物理寄存器文件的指令的数值结果,以打破指令之间的数据依赖性. 后续具有数据依赖关系的指令可以利用预测值进行推测执行,而无需等待前面的指令执行. 值预测器通常通过保存指令在历史上出现的数值信息,并根据这些信息预测指令未来可能出现的数值结果.

例如当指令j依赖指令i的数据时,通过预测指令i的结果,使指令j具备执行条件并提前发射执行,而不需要等待指令i执行完毕. 如果值预测器正确地预测了指令i的结果,处理器就能获得性能收益,更多的指令可以提前发射执行. 但如果值预测器的预测结果与指令i实际的结果不符,则预测错误. 此时需要对流水线进行恢复,以避免使用了错误数据的指令对处理器的状态造成污染,并需要重新利用正确的数据执行后续指令.

图1展示了一个四发射的超标量流水线. 其中,指令D的执行依赖指令C,而指令C的执行又依赖指令A和B. 在没有使用值预测的情况下,假设每条指令都需要一个周期完成运算,那么所有指令共需3个周期才能完成全部计算. 在这种情况下,流水线的IPC为1.3. 然而,在图1左侧使用了值预测的情况下,对指令C和指令D所需的操作数进行了预测,使它们满足执行条件,无需等待指令A和指令B的执行. 这样,所有指令可以同时发射执行. 在预测正确的情况下,此时流水线的IPC可以达到理论设计的最大值4.

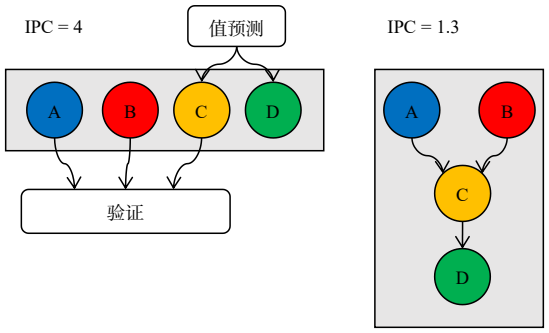

图1　值预测示例

值预测通过预测性地打破原有的数据依赖链,预测性地执行具有数据依赖关系的指令,可以让更多的指令在处理器中并行执行. 因此,值预测可以充分利用超标量处理器中的硬件计算资源,隐藏Load访存指令的访存延迟,提升超标量处理器的指令级并行度,提高处理器的性能和吞吐量. Load访存指令除了需要花费大量的时间等待访存行为的完成以外,还需要消耗大量的时间在Store缓存中进行内存消歧[30],因为Load访存指令之前的Store访存指令数据是写入Store缓存中的,可能还没有真正写入内存. 因此,需要首先计算Load访存指令的访存地址,并根据Load访存地址检查Store缓存中的Store地址,如果存在相同的地址,则Load指令需要使用Store缓存中的数据写回寄存器文件,这一机制也被称为STLF(Store-To-Load-Forwarding). 显然,值预测技术通过预测Load

指令的数值结果能够隐藏上述Load访存指令存在的访存延迟和内存消岐开销. 除了值预测技术以外,目前业界针对Store-Load的真相关问题已有的解决方案包括推测性存储绕过(Speculative Store Bypass,SSB)[31]和预测性存储转发(Predictive Store Forwarding,PSF)技术[32]. SSB技术的主要原理为通过推测Load访存指令的数据不会被Store缓存中的数据所覆盖,一方面预测性地使用Load访存的数据继续执行后续的指令,另一方面并行执行内存消岐过程. 在预测正确的情况下,SSB技术就可以隐藏内存消岐开销. 然而,SSB技术仍然不能解决由存储墙问题导致的Load访存延迟问题. 与SSB相对的PSF技术则是通过预测Store-Load指令对具有相同的访存地址,Store指令的数据将会直接转发给Load指令的目的寄存器文件,并继续执行后续的指令,从而隐藏访存地址的计算和比较等过程. PSF技术能够在Store指令数据还存在处理器中的情况进行预测性转发,然而,PSF技术无法对已经写回内存的Store指令数据进行转发. 与之不同的是,值预测器可以保存Load指令历史上的数据信息,无论这种具有相同访存地址的Store-Load指令对的距离有多远都能够实现预测. 此外,不仅针对Load访存指令,值预测技术还可以面向一些具有较长执行时间的指令,例如浮点计算,进行预测,这是SSB和PSF预测技术所完全不支持的.

2.3 常用评估指标

在值预测的设计过程中,通常会使用多种评估指标来综合评估其性能和效果. 下面介绍三种常用的值预测性能评估指标:

2.3.1 每周期指令数

每周期指令数是一种直接反映值预测技术对处理器性能影响的评估指标. 它也是计算机体系结构领域中最常用的指标之一,可以直接反映处理器的性能变化. 每周期指令数可以使用式(1)计算,其中正确提交的指令数量在特定的测试场景下通常是固定的. 值预测技术通过预测性地让更多指令在处理器中并行执行,从而减少处理器执行的总周期数,进而提高IPC.

$$\mathrm{IPC}=\frac{\textit{正确提交的指令数量}}{\textit{处理器执行总周期}} \tag{1}$$

2.3.2 预测准确率

预测准确率描述了值预测器正确预测的指令数量占所有有效预测的指令数量的百分比. 有效预测指的是值预测器输出的预测值被写入物理寄存器文件,并作为后续指令的源操作数使用. 与之相对,无效预测指的是对于指令的值预测置信度还没有达到阈值,对应的预测值无效,不能被后续的指令所使用的情况. 而正确预测指的是有效预测的预测值与指令实际的数值结果相等. 预测准确率可以使用式(2)计算. 由于值预测错误会带来较大的性能惩罚,因此对值预测器的预测准确率要求较高. Perais等人[33]的研究表明,如果不是在理想的错误恢复机制下,95%~99%的预测准确率是不足以避免性能损失的. 因此通常要求预测准确率要大于99%,这对值预测器的设计提出了较高的挑战. 此外,业界常用的错误预测率指的是错误预测的指令数量占有效预测的指令数量的百分比,其与预测准确率之和恒为1.

$$\mathrm{accuracy}=\frac{\textit{正确预测指令数量}}{\textit{有效预测指令数量}}\times 100\% \tag{2}$$

2.3.3 预测覆盖率(coverage)

预测覆盖率用于描述值预测器有效预测的指令数量占所有目标预测指令数量的百分比. 目标预测指令是在值预测设计之初预先定义的需要进行值预测的指令. 例如,如果只针对Load指令进行值预测,则所有提交的Load指令都被视为目标预测指令. 预测覆盖率可以使用式(3)计算. 预测覆盖率在一定程度上反映了值预测器的性能和剩余性能收益的潜力. 预测覆盖率是衡量值预测器有效性的重要指标之一,较高的预测覆盖率意味着值预测器可以有效地预测更多的指令,提供更多的性能提升. 设计人员可以根据预测覆盖率来评估值预测器的性能,并进行进一步的优化和调整. 在本文中,除特别说明外,例如只面向Load访存指令进行预测,目标预测的指令都是所有将要写回寄存器文件的指令.

$$\mathrm{coverage}=\frac{\textit{有效预测指令数量}}{\textit{目标预测指令数量}}\times 100\% \tag{3}$$

2.4 流水线修改

在实际的流水线中运行值预测器时,通常情况如图2所示. 在流水线的前端,值预测器提供指令的预测值. 这些预测值一方面在重命名后写入物理寄存器,供后续指令使用;另一方面,如果当前周期存在数据依赖关系的指令,预测的数值将直接作为源操作数进行发射和执行. 所有经过值预测的指令仍然按正常流程执行. 在流水线的后端,当指令的真实结果获得后,将与预测值进行比较验证. 当预测正确时继续执行,当预测错误时,将启动相应的恢复机制.

因此,值预测技术对于流水线的修改主要体现在以下3个方面.

(1) 增加额外的预测值传递机制

为了利用处理器的预测值,需要将值预测器的有效预测输出写入乱序执行核心. 目前已经提出了3种主要方案. 方案一是在调度时将预测值写入物理寄存器文件PRF(Physical Register File),因此值预测器只需在写入物理寄存器之前提供预测值即可. 为了验证预测值的正确性,在指令执行完毕后需要进行预测值的

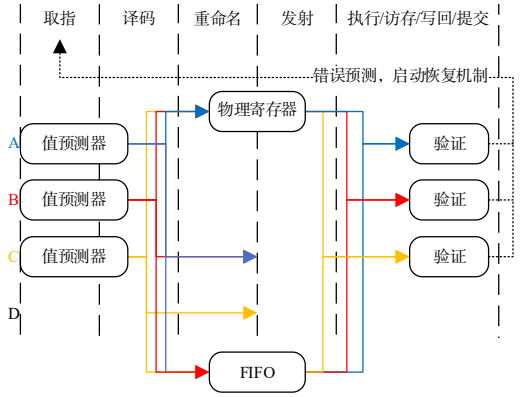

图2　处理器流水线示例

比较验证. 由于之前的预测值直接写入了PRF,所以需要额外的物理寄存器读端口,用于读取预测值并与指令执行的数值结果进行比较. 例如,在一个八发射的处理器中,如果需要对所有指令进行预测,则物理寄存器文件需要额外的八个读端口和八个写端口. 然而,增加这么多端口在物理上很难实现,因为随着PRF中端口数量的增加,功耗和面积也会呈二次方增长. 方案二是复用现有的物理寄存器读写端口,通过对执行指令的读写操作和值预测器的读写操作进行仲裁,复用现有资源. 然而,这种方法可能会增加关键路径上的延迟,并增加PRF读写的通信带宽压力,可能成为整个系统的瓶颈,甚至降低系统的整体性能. 显然,这样的设计对于高性能处理器而言并不适用. 方案三是使用专用的片上存储结构. 该结构仅用于存储预测值,后续指令只从这个专用存储结构获取预测值,因此不会对现有PRF结构造成影响. 总的来说,方案一和方案三都会引入额外的开销,区别在于开销是在原有的PRF还是专用存储结构上产生的. 专用存储结构所需的硬件资源开销较少,但由于引入了仲裁功能,可能会在关键路径上增加额外的延迟开销,从而影响处理器的整体时序约束.

(2) 新增值预测器和值预测验证模块

值预测器和值预测验证模块对乱序执行引擎的影响较小. 值预测器通常位于流水线的前端,负责提供指令的预测数值. 值预测器通常保存指令在历史上产生的数值信息,这将消耗大量的片上存储资源,影响处理器流水线的时序验证,并对后端的物理实现提出了较大挑战. 值预测器的详细结构将在第4节中详细阐述.

值预测验证模块主要负责比较有效预测值与指令的执行结果是否相等,并在二者不相同时启动相应的错误恢复机制. 一种比较验证模块设计如图2所示,其中值预测器的输出一方面写入指令的目的寄存器,另一方面写入FIFO(First In First Out)缓存. 在预测的指令执行完毕后,其数值结果将重新写入指令的目的寄存器. 在指令提交时,按顺序读出对应的目的寄存器和FIFO缓存中的数值结果进行比较,以验证值预测结果,并在发现二者数值不一致时启动相应的错误恢复机制. 此外,在指令执行完毕提交时,通常会将指令的真实执行结果返回给值预测器,用于更新值预测器.

(3) 新增涉及整个流水线的错误恢复机制

流水线的值预测错误恢复机制需要在值预测错误的情况下对处理器状态进行恢复. 由于在错误预测的情况下,处理器可能已经在错误预测的路径上执行了一段指令,因此通常流水线的前端需要重新定位到错误预测指令的后一条指令位置,对于流水线的后端,需要清除所有受错误值预测影响的指令以及对处理器状态的影响. 常用的错误恢复机制将在第5节中介绍.

2.5　综合分析

通过以上的介绍可知,值预测技术通过打破原有的数据依赖进而可以提升处理器的指令级并行度. 目前常用处理器的每周期指令数、值预测的预测准确率、值预测的预测覆盖率三个指标评估值预测的性能. 为了使用值预测技术,还需要对现有流水线架构进行修改. 主要修改包括以下3个方面:其一是流水线所支持的预测值使用,包括后续依赖指令的推测执行和执行结果同预测值之间的对比验证;其二是增加值预测器模块,值预测器除了本身的预测工作而外还需要不断接收流水线后端指令提交的信息用于更新值预测器;其三是增加值预测错误时的流水线恢复机制. 本文也以此3个方面分别用三节进行介绍.

3　值预测流水线架构

值预测器可以根据指令的地址和处理器的上下文给出对应的预测值,而现有的流水线架构却不能直接使用这一预测值,需要进行适当的修改以支持值预测. 值预测器所提供的预测值通常需要通过预测值传递机制给后续的指令以提供预测性的源操作数,例如通过写入PRF或者是写入专用存储结构的方式. 从流水线设计的角度来看,额外的PRF读写端口、额外的片上存储结构、预测值的验证以及恢复工作都会带来额外的性能、功耗、面积(Performance Power Area,PPA)开销. 在本节中,我们主要介绍近年来提出的几种支持值预测的流水线架构设计.

3.1　EOLE架构

Perais等人[34~37]提出了一种名为EOLE的架构,EOLE即早-乱序-晚执行(Early/OoO/Late Execution,EOLE)的缩写. 它主要瞄准支持值预测的流水线中的两个问题,分别是复杂的乱序执行引擎以及通过增加PRF端口的方式传递预测值机制所需的大量硬件资源问题. 图3展示了EOLE架构的流水线图和对流水线的

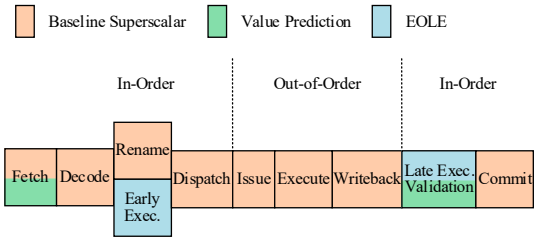

图3 EOLE流水线图[34~36]

主要修改.

首先值预测器将在取指阶段对指令的结果进行预测. 在重命名阶段,部分指令的源操作数已经准备就绪,这些操作数可能是立即数,也可能是预测器给出的预测值. 这部分指令将在重命名阶段进行早执行. 对于预测器所预测的指令以及早执行阶段所执行的指令,会把数值结果在分发阶段写入物理寄存器文件,而不再进入乱序执行部分,同时预测值也会写入一个类似FIFO的队列. 最后在比较验证阶段,已经满足执行条件的预测指令将会在这一阶段进行晚执行,并完成FIFO中的预测值和实际执行结果的比较验证. 通过早-晚执行,部分指令已经不需要进入乱序执行引擎,进而降低了乱序执行部分的硬件性能需求. 因此可以适当降低发射宽度以及相应的物理寄存器文件写端口,从而更加容易进行乱序执行部分的设计.

3.2 BeBoP架构

Perais等人基于EOLE的思想进一步地提出了BeBoP的流水线架构[38,39],即基于块的值预测(Block-Based Value Prediction,BBP或BeBoP). BeBoP主要针对EOLE的架构中的两个问题尝试进行解决:其一是对于超标量流水线而言,需要值预测器同时给出多个指令的预测值,而值预测器所需的多端口的存储结构在成本上是非常昂贵的;其二是值预测器本身的硬件资源开销较大,因此在降低值预测器硬件资源开销的同时还必须要保证一定的预测覆盖率条件下才会被考虑实现.

针对第一个问题,他们提出了BeBoP的架构,将一个取指块中的多条指令同时放在值预测器中的一个表项之中,并利用取指块的块地址进行值预测器的索引. 这样就可以通过一个读端口的一次读操作实现对于一个取指块内多条指令的同时预测. 针对第二个问题,差分VTAGE(Differential VTAGE,D-VTAGE)预测器被提出. 这一预测器通过预测步幅信息降低了预测器本身的硬件资源需求,因为相比指令的数值结果,数值结果之间的差值即步幅,是比较小的,因此可以适当降低保存步幅信息的数据位宽进而降低预测器的硬件资源开销.

3.3 DLVP架构

不同于对指令直接进行值预测,Load地址预测通过预测Load指令的访存地址,并利用该地址访存数据缓存,将读取的数值作为预测值使用. 如图4所示为Sheikh等人[40]提出的DLVP地址预测架构,从缓存中取回的预测值将保存在专用的片上存储中. 地址预测是一种间接实现值预测的方式,从缓存的角度来看,地址预测同值预测的差别在于Load地址预测会对数据缓存带来额外的带宽压力,在数据缓存带宽较小的情况下,这甚至会影响处理器的性能[41]. 从预测覆盖率的角度来看,Load地址预测只是针对Load指令进行的预测,程序中大约只有25%的指令是Load指令,而值预测可以针对更多的ALU指令实现预测,因此Load地址预测的预测覆盖率是低于值预测的. 从局部性的角度来看,二者利用了不同的局部性. Load地址预测利用的是地址的局部性,而值预测针对的是数值的局部性.

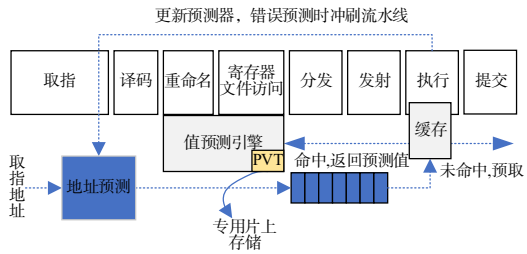

图4 DLVP地址预测的架构[40]

3.4 混合地址预测和值预测的架构

为了获得更高的处理器性能,Sheikh等人[20]基于DLVP架构进一步针对现有的4种不同类型的Load预测器进行了分析,包括2种Load地址预测器和2种Load值预测器. 他们发现其预测空间是部分正交的,换言之地址和数值具有不同的局部性,因此提出混合地址预测和值预测的混合预测器,从而实现了更高的处理器性能加速比. 图5是其设计的流水线架构图,其在DLVP架构的基础上增加了值预测结构和过滤机制,值预测和地址预测的工作在取指阶段完成. 对于地址预测的部分,首先会通过地址查看Cache缓存中的数据. 倘若Cache命中,则将Cache中的数据作为预测值;倘若Cache未命中,则对更低级的Cache发出预取请求. 虽然Cache未命中的情况下不能及时提供值预测引擎所需的预测数值,但是这可以让指令真正开始执行时更快地访问到数据. 而当值预测错误时,则对流水线进行冲刷,恢复处理器状态. 流水线冲刷的错误恢复机制将在第5节进行详细介绍. 地址预测能够很好地预测部分地址较为固定,但是数据经常变化的访存指令,例如访问某些外设接口的指令,而这类访存指令是值预测器很难预测正确的. 为了提高预测准确率,地址预测和

值预测的结果都将通过一个过滤机制过滤掉在历史上错误预测率较大的预测结果,这一机制基于历史上指令的错误预测统计情况实现.而在地址预测和值预测同时有效且没有被过滤的情况下,该架构优先选用值预测的结果.这是因为二者同时预测有效且预测结果不同的情况非常少(<0.03%),而地址预测还会给缓存带来额外的带宽开销.

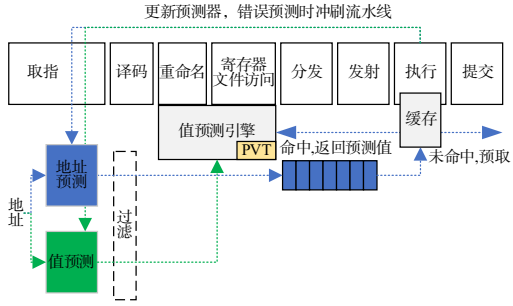

图5　支持Load地址预测的架构[20]

3.5　AVPP架构

不同于地址预测架构中只面向当前指令的一个访存地址进行预测的策略,Lois等人[42]提出的AVPP(Address-first Value-next Predictor with Value Prefetching)架构中的地址预测器将会面向Load指令的两个地址进行预测,分别是预测地址和预取地址,其架构如图6所示.同DLVP架构类似,AVPP的预取地址将用于从缓存中预取数据,此预取数据是该条静态指令未来的某个实例将要使用的数据.由于预取的数据并不是当前指令所使用的,因此并没有直接写入值预测引擎,而是将其保存在值缓存表中.当前指令则是利用预测地址访问值缓存表获得预测值,在该缓存表命中的情况下,预测值将直接写入物理寄存器文件.AVPP在本质上是将值预测器当作L0缓存使用.此外,Ricardo等人[43]则是将Load地址作为PRF的标签并利用其探索了PRF之间的重用性,通过PRF之间的数据重用,可以降低数据缓存的访问压力.这类支持Load地址预测的架构创新性地将值预测、地址预测和数据预取进行了结合,使用Load地址路径作为上下文实现了很好的处理器性能.更为重要的是,Sheikh等人[40]提出使用专用的片上存储结构用于存储预测值的方式,这有助于降低由于引入PRF额外读写端口所带来的开销.

3.6　综合分析

目前值预测的流水线架构设计中的主要问题在于值预测器以及PRF等的开销问题,此外在流水线设计过程中还需考虑如下3个方面的问题:(1)值预测器的选择;(2)值预测验证点的选择;(3)错误预测恢复机制.本节将所介绍的近年来提出的支持值预测的流水线架构总结如表1所示.EOLE/BeBoP都选择了一种基

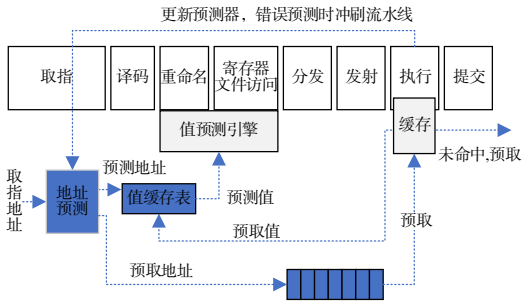

图6　AVPP架构[20]

于TAGE[11]的值预测器,而Sheikh等人[20]则是使用了多种值预测器混合的方式,实现了比单一值预测器更高的处理器性能.EOLE主要用于降低执行段的压力,BeBoP则主要是在此基础上降低值预测器的开销,而Sheikh等人[20,40]的工作在结合了地址预测和值预测的基础上通过增设专用片上缓存避免了PRF的额外读写端口需求.而AVPP架构则是在地址预测的基础上通过L0缓存预取的方式实现预测.总的来说,目前优先考虑低成本高性能的值预测器,由于Load地址拥有Load数值类似的局部性,因此通过利用这部分局部性,流水线的性能也逐渐被挖掘出来.对于值预测的验证点,通常选择执行段或是单独加一级流水线实现,值预测的验证点越靠后,则错误预测时的惩罚就越大,在设计选择时需要进行各方面的权衡.另外得益于较高的预测准确率,目前通常采用流水线冲刷机制进行错误恢复.值预测器将会在第4节进行介绍,错误预测恢复机制将在第5节进行介绍.

4　值预测器结构

Sazeides等人[16]首先将值预测按照预测方式分为了基于计算的和基于上下文的,现在几乎所有的值预测器基本都满足这一分类方式.在本节中,同样沿用这一分类方式对值预测器的发展进行介绍.

4.1　基于计算的值预测器

4.1.1　传统步幅预测器

基于计算的值预测器是将先前的数值利用一个函数计算得到预测值的[16].目前常见的基于计算的值预测器是步幅预测器,该预测器在先前值的基础上通过加上一个步幅值得到新的预测值,主要针对程序中具有步幅特征的指令进行预测.步幅(stride)特性是程序中不可忽略的一部分指令,例如循环变量的计算、地址预测[45~47]等都经常呈现这一特性,因此步幅预测也常用于地址预测[20]和数据预取[48,49]等领域.以下展示了2种常见的具有步幅特征的序列.

第一,常量序列,例如:“100,100,100,…”当某条指令所产生的数值结果是一个常量时,其会表现出这

表1 支持值预测的流水线架构

架构	会议/期刊,提出时间	预测器	值预测验证点	错误预测恢复机制	预测值传递机制	特点
EOLE[34]	ISCA,2014	VTAGE	晚执行段	流水线冲刷	增加PRF读写端口 增设值预测队列	降低执行段的压力
BeBoP[38]	HPCA,2015	D-VTAEG	晚执行段	流水线冲刷	增加PRF读写端口 增设值预测队列	降低值预测器的开销
DLVP[40]	MICRO,2017	PAP	执行段	流水线冲刷	增设专用片上存储	基于路径地址实现地址预测,避免了PRF的额外读写端口需求
Load地址预测架构[20]	HPCA,2019	LVP[14]+CVP[33] SAP[44]+CAP[40]	执行段	流水线冲刷	增设专用片上存储	结合地址预测与值预测,避免了PRF的额外读写端口需求
AVPP[42]	TACO,2018	AVPP-DVTAGE AVPP-Stride	执行段	流水线冲刷	增加PRF读写端口	根据预测地址实现L0预取

种特征. 例如对高级语言中的类进行实例化时,会对部分属性进行初始化操作,这部分初始化数值通常较为固定,在编程之初就已经确定. 此外对于高级语言中声明的静态变量而言,其变量地址在整个程序运行期间,也都是固定不变的,对于这类变量的地址预测也能表现出较好的预测性能.

第二,步幅序列,例如:“100,200,300,400,…”. 循环变量的数值变化是一种较为常见的步幅序列,例如以下展示的循环变量i将表现出步幅为1的序列. 同样的特征还会出现在例如计算卷积时矩阵的地址计算过程之中.

```
for(int i=0;i<100;i++){
a[i]=b[i] + 1;
}
```

最初设计的步幅预测器的结构如图7所示,其利用指令地址对预测表进行索引,预测表是一个类似cache的结构,其利用指令地址的高位当作tag信息,当tag信息匹配时输出对应的预测值. 这一预测器针对的是某些固定的指令表现出的步幅特征,但是由于没有置信度机制,因此预测准确率较差. Gabbay等人[15]也将这一预测方式扩展成了寄存器文件值预测器,其利用指令的目的寄存器地址对预测表进行索引,并利用相似的步幅计算方式得到预测值. 类似的,他们还将步幅预测的机制引入浮点指令,将原有的数值和步幅信息分别用浮点的符号、指数和分数部分来表示,并分别对指数部分和分数部分进行步幅计算. 这些最初设计的预测方式都比较简单,且没有全面的性能评估实验. 但是利用步幅信息进行值预测的原理一直沿用至今,许多步幅预测器在预测值的生成方式上都是类似的,例如以下将要介绍的几种步幅预测器.

图7显示的是一种较为简单的预测机制,任何匹配的指令都会输出对应的预测值,而没有任何置信度的计算机制. 基于此,Sazeides等人[16,50,51]便提出的两级步幅(2-delta)预测器,它是步幅预测器的一种变体. 两级步幅预测器将维护2个步幅信息:一个步幅(s1)始终通过两个最近值之间的差异进行更新,这和图7的更新机制类似;而另一个步幅(s2)是用于计算预测的步幅. 当步幅s1连续出现两次时,则会将其更新到预测步幅s2. 因此可以将其视为一种饱和机制,只能使用重复2次出现的相同步幅来进行预测. 这样的机制能够避免许多错误预测的出现,大大提高预测准确率.

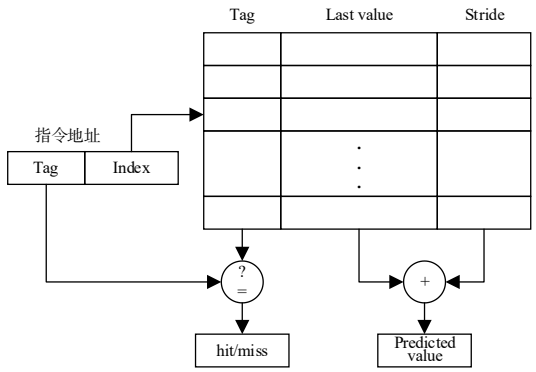

图7 步幅预测机制[15]

Rychlik等人[52]提出的Stride+值预测器和2-delta预测器的机制很相似. 他们基于图7的机制,在表项中额外保存了匹配的步幅和一个饱和计数器,其结构如图8所示. Stride+预测器利用指令地址索引预测表,在表项命中时利用上一次的数值和匹配的步幅计算得到预测值. 表项中匹配的步幅只有在连续出现两次相同的步幅的情况下才会被更新. 另外饱和计数器的存在也提高了预测器的输出阈值,以预测覆盖率降低为代价提高了预测器的预测精度. 这些机制使Stride+预测器获得了平均5%的预测准确率提升.

Shimomura等人[53]基于Stride步幅预测器进行了更细致的表项管理,对新表项的分配、替换等设置了较高

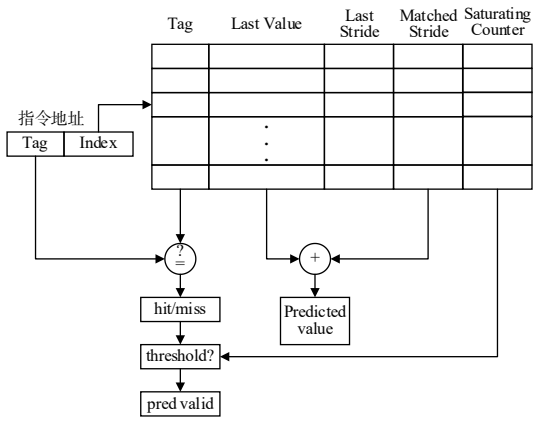

图8　Stride+步幅预测机制[52]

的阈值,进而也获得了性能收益. 总的来说,传统的步幅预测器可以通过提高输出阈值,以牺牲预测覆盖率为代价获得较高的预测精度.

4.1.2　最后一次值预测器

除了针对具有步幅变化特性的值预测器外,还有一类特殊的基于计算的值预测器,即最后值预测器LVP(Last Value Predictor). LVP根据指令最后生成的一个值来预测当前的指令结果,换言之即是步幅为0的步幅预测器. Lipast等人[13,14,54]首先提出了这种预测器,其利用数值预测表专门保存指令最后产生的结果,然后再利用一个分类表决定是否输出当前的预测值. LVP预测器结构如图9所示,其中的分类表和预测表均通过指令地址索引,只有2个表均命中且在分类表中的计数器饱和的情况下才能得到有效的预测值.

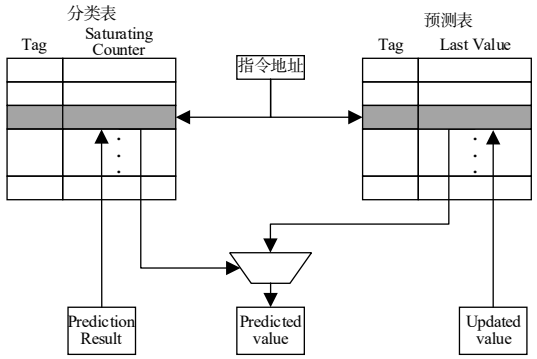

图9　LVP预测机制[13]

Loh[55]为了降低LVP的存储容量,对不同的指令采用不同的数据宽度进行保存. 为了增加最后值预测器的预测能力,Wang等人[56]提出了最后4个值预测器(Last 4 Value,L4V). L4V的结构如图10所示,其主要由2个表组成,分别是值历史表和模式历史表. 值历史表最多会记录4个不同的值历史,并使用编码后的值历史模式来索引相应的模式历史表,用于确定这4个值中的哪一个是预测值. 这种预测器显然只能针对常量数值和一些周期同指令宽度相同的周期重复序列进行预测. 基于L4V,Chang等人[57]发现部分指令需要更长的值历史才能实现预测,因此他们探索了记录最后8个值的方式进行预测,并证明了该方式的有效性. 这类LnV(Last n Values)预测的方式显然在资源利用率方面没有优势:一是不能无限地增加预测器中保存的值历史二是保存的值历史中至多只有一个正确的数值能够被使用. 此外,由于这类预测器利用局部值历史,即某条指令提交的数值结果序列信息,进行预测表的索引,因此这类保存最后n个局部值历史的LnV预测器在一定程度上也可以被归类到基于上下文的类别. 由于LnV是基于LVP改进形成的,而LVP是一种特殊的步幅为0的步幅预测器,因此,为方便起见,本文也将这类LnV预测器归类于基于计算的值预测器.

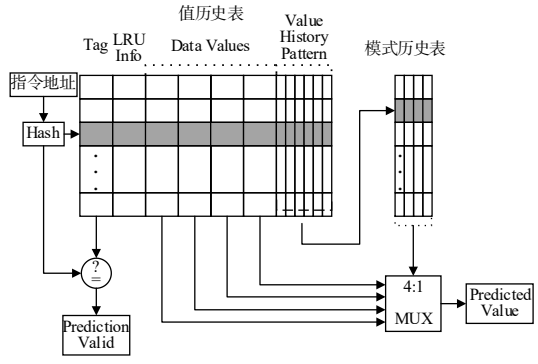

图10　L4V预测机制[56]

4.1.3　基于全局历史预测

全局历史信息主要包括全局值历史和全局分支历史. 全局值历史即是在处理器中提交的写回物理寄存器类指令的数值结果序列信息,而全局分支历史指的是在处理器中的提交的分支指令的指令地址或目标地址序列信息. 不同于前述利用局部值历史进行预测的方式,Bodine等人[58~60]首先尝试探索全局值历史的局部性,并提出了gDiff值预测器,gDiff同样是通过Stride的方式保存全局的步幅信息并通过相加方式计算得到预测值的. 不同于全局值历史,Nakra等人[61]提出基于路径的步幅(Per-Path Stride,PS)预测以及基于路径的最后值(Path-based Last Value,PLV)预测,预测器结构分别如图11和图12所示,其利用全局分支历史区分不同的路径,通过利用全局分支历史和指令地址的哈希值索引值预测表的方式对处在不同路径上的指令进行步幅和最后值预测,步幅预测基于传统的2-delta机制实现. 最终的性能评估结果也表明,PS和PLV分别比传统的Stride和LVP预测器的预测准确率高8.4%和6.4%. 利用全局信息的方式在分支预测器,数据预取器等设计时也有较多的案例[62~64],因此从现

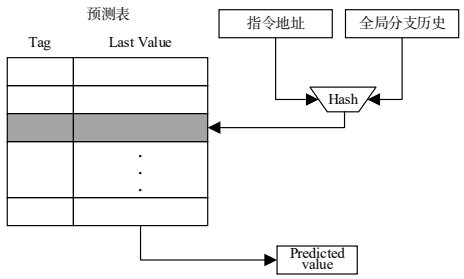

图11 PLV预测机制[61]

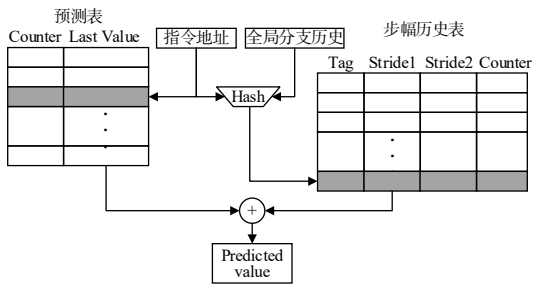

图12 PS预测机制[61]

有的研究来看全局值历史中确实存在着部分局部性可供探索.

4.1.4 特定步幅模式预测

Yong等人通过在步幅预测器表项中增加额外的控制位,提出了修正的步幅预测器RSVP[65]. RSVP通过保存最后两个数值信息,不仅可以预测传统的步幅指令,而且可以支持部分只能由基于上下文的值预测器预测的指令(例如:1 -5 1 -5…). 但是RSVP自身的资源开销较大,保存2个数值信息所带来的存储开销几乎是原来的两倍.

表2 基于计算的值预测器

预测器	会议/期刊,提出时间	饱和计数器	概率计数器[66]	全局历史	局部历史	预测器开销	测试环境	测试程序	性能收益
LVP[14]	ASPLOS 1996	√	×	×	√	16 KB	VMW[67]	SPEC 92/95	缓存冲突时间平均减少6%
2-delta[16]	MICRO 1997	×	×	×	√	∞	—	SPEC95	56%预测准确率
L4V[56]	MICRO 1998	√	×	×	√	>79 KB	—	SPEC92 int	≈50%预测覆盖率
Stride[15]	1998	×	×	×	√	∞	自研模拟器	SPEC95 int	57.66%预测准确率
Stride+[52]	1998	√	×	×	√	∞	PSIM	SPEC95 int	预测准确率比Stride提升5%
PLV[61]	HPCA 1999	√	×	√	√	92 KB	SHADE[68]	SPEC95	预测准确率比LVP提升6.4%
PS[61]	HPCA 1999	√	×	√	√	136 KB	SHADE[68]	SPEC95	预测准确率比Stride提升8.4%
gDiff[58]	2002	×	×	√	×	Not given	Simple Scalar[69]	SPEC2k int	73%预测准确率 55%预测覆盖率
ES[17]	CVP 2018	×	√	×	√	0.62 KB	CVPv6[12]	SPEC06/17 数据库类 加密程序	16.1%处理器IPC性能提升 3.6%预测覆盖率 99.4%预测准确率

Chen等人[70]为了更好地预测重复波峰(Repeated Peak, RP)和重复局部步幅(Shifting Locality Stride, SLS)数值模式,分别提出了重复波峰预测器(Repeated Peak Predictor, RPP)和重复局部步幅预测器(Shifting Locality Stride Predictor, SLSP). RSVP, RPP, SLSP这类预测器都是针对特定的数值模式所设计的,在特定情况下的表现较好,但是在资源利用率和通用性方面则表现不佳.

4.1.5 最先进的步幅预测器

目前较为先进的步幅预测器是Seznec在第一届CVP值预测竞赛中提出的增强型步幅预测器(Enhanced Stride, ES)[17],目前CVP值预测竞赛各个赛道上的预测器均包含增强型步幅预测器[71,72]. ES的结构如图13所示,其中inflight表示当前指令在指令窗口中的实例数量. 预测值由上一次提交的值加上指令窗口中的相同指令数量加一倍步幅得到. 通过这种预测方式能够解决指令窗口中存在多条相同指令时的预测问题,同时ES采用的概率计数器、细粒度的置信度管理和表项分配等策略进一步降低了错误预测率.

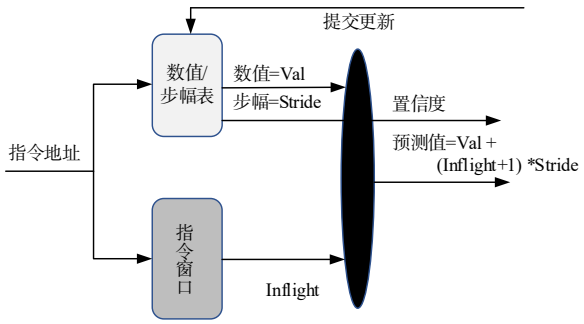

图 13　增强型步幅值预测器[17]

4.1.6　步幅预测器小结

基于计算的值预测器的几种具有代表性的预测器的总结在表 2 之中. 从中可以看出 4 个特点:一是值预测经历了长时间的发展,步幅预测器仍然是主要的基于计算的值预测器,这主要是由于步幅特征是程序中主要存在的一类数值计算特征,而通过探索其他模式的预测器则在处理器性能收益或性价比方面表现不佳;二是概率计数器因其较高的预测准确率和资源利用率也被引入到步幅预测器中,通过提高步幅预测器的输出阈值,以牺牲预测覆盖率为代价可以提高步幅预测的准确率;三是局部值历史是步幅预测器主要利用的历史信息,但同时全局值历史中仍然存在着部分可预测性;四是 21 世纪以来,基于计算的值预测器的发展十分缓慢,鲜有研究在该方向上获得突破性成果.

4.2　基于上下文的值预测器

4.2.1　传统有限上下文方法预测器

这里的有限上下文定义为一个有限且有序的值序列,值则是来源于指令产生的结果. 而基于上下文的预测器则是在相同上下文时预测其中一个值. 这使其能够预测一些重复序列[16]. 这一定义同来源于文本压缩[73]的有限上下文方法(Finite Context Method,FCM)预测器的定义一致. 一个 k 阶的有限上下文方法预测器使用 k 个先前值作为上下文. 预测器由计数器构成,这些计数器对特定上下文出现的特定值进行计数. 如图 14 所示,阶数越高的有限上下文预测器,通常具有越多的计数器个数,在某一特定上下文情况下,计数器越高的数值具有越高的输出权限. 要想获得更准确的预测,那么上下文的阶数就必须越高,上下文的范围也必须越广. 显然,这些变化带来的硬件资源开销增长是指数级的.

FCM 的提出者也讨论了其预测模型的实现方式[75],他们通常使用两级预测表的结构实现预测. 主要包括两个表,分别是值历史表(Value History Table,VHT)和值预测表(Value Prediction Table,VPT). 通常可以利用指令地址等处理器状态索引 VHT 获得对应的值历史上下文,然后再通过上下文和处理器状态共同索引 VPT 获得对应的预测值. 而当指令提交时,则更新对应的 VHT 和 VPT.

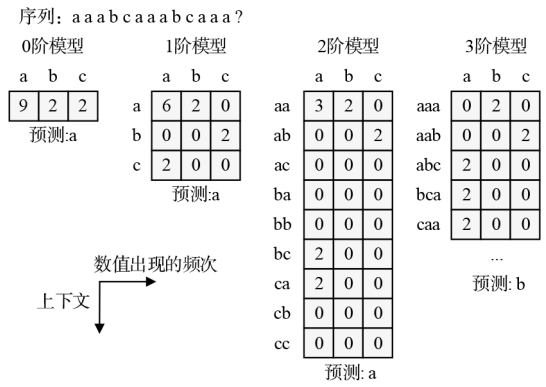

图 14　有限上下文模型[16]

在高级语言中,这类 FCM 预测器能够针对多次循环过程中的许多内层指令实现预测,因为通常内层循环过程中的许多计算结果都会随着外层循环而不断重复. 以下展示了 2 种嵌套循环中常见的具有周期性重复序列:

(1) 重复步幅序列

例如:在如下所示的两层循环中,内层的循环变量 j 表现出“0,1,2,0,1,2,…”,是周期性重复的步幅序列.

```
for(int i=0;i<100;i++){
for(int j=0;j<2;j++){
…
}
}
```

(2) 重复非步幅序列

例如:“1,-13,-99,1,-13,-99,…”是周期性重复的非步幅序列.

使用 FCM 预测器能够对这类重复的数值序列实现很好的预测,而传统的步幅预测器往往不能很好地预测这类序列. 这类 FCM 预测器也是当时最为先进的值预测器.

4.2.2　基于 FCM 的差分技术优化

基于 FCM 有限上下文方法,Goeman 等人[74]提出了差分有限上下文方法(Differential FCM,DFCM)预测器,其结构如图 15 所示. DFCM 利用步幅历史索引下一级预测表得到相应的步幅,并结合上一次的数值得到最终的预测值. 由于 DFCM 是预测步幅而不是完整的数值,通过降低步幅信息的数据位宽可以在预测器内部节省更多的硬件资源,因此,在相同的硬件资源情况下,DFCM 的预测准确率将会更高,在相同开销的情况下 DFCM 的预测准确率最高提升了 15%. 而后,Martin[76]在针对(D)FCM 的研究过程中发现在许多情况下 VPT 表的索引范围比较窄,并不能很好地覆盖 VPT 的

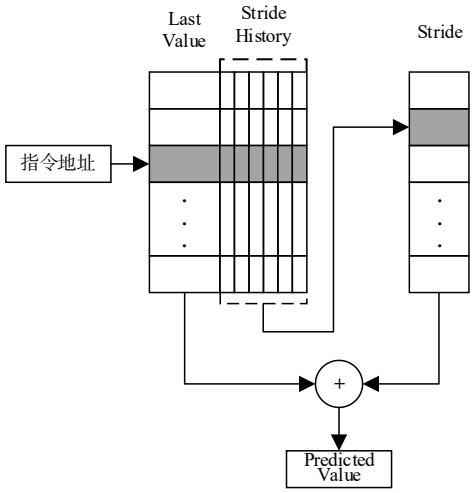

图 15 DFCM 预测机制[74]

存储空间,因此对于 VPT 表而言其资源利用率较低. 为此,Martin 等人设计了一种改进的索引哈希函数. 相比于原有的(D)FCM,改进后的索引哈希函数能够生成覆盖范围较广的索引,可以充分利用 VPT 表的存储空间,最终帮助预测器提升了 5% 左右的预测覆盖率.

基于 DFCM,Deshmukh 等人[77]在第一届 CVP 值预测竞赛中提出了 DFCM++预测器. 相比原有的 DFCM,DFCM++做出了 4 个方面的增强,分别是(1)提前更新策略;(2)数据依赖驱动的值估计器;(3)PC 黑名单策略;(4)动态上下文长度策略. 提前更新策略即是针对在指令窗口中具有多个实例的指令,在预测时会用预测值更新对应的表项,这样有利于对指令窗口中存在多个实例的指令的情况进行预测. 数据依赖驱动的值估计器则是通过指令过去的数据情况推断出其操作类型,然后一旦给出源寄存器的值,便可以推断出这一指令的结果. PC 黑名单策略则是将一些短时间内多次预测错误的指令地址进行记录,并在后续的一段时间内屏蔽针对这些指令的预测,这有助于降低值预测的错误预测率. 动态上下文长度驱动策略则是动态地决定上下文的长度,这一策略来源于某些数值模式只在短时间内重复出现,而太短的上下文在全局上的表现却很差. 他们分别实现了 32 和 64 阶的 DFCM,然后利用额外的性能计数器在两个 DFCM 之间选择输出,这是一种混合值预测器中常用的设计方法. 相比 DFCM,DFCM++获得了 40.2% 的处理器 IPC 性能提升. 但相比于没有使用值预测,DFCM++的处理器 IPC 性能提升为 28.1%,即 DFCM 在这一测试条件下甚至获得了负的性能收益.

4.2.3 基于 FCM 的上下文优化

基于 FCM,Gupta 等人[72]在第一届 CVP 值预测竞赛中提出了 Sliding FCM 预测器,因为其参加的是无限资源赛道,因此没有采用差分的方式. 其在预测时假设指令窗口中的预测值是正确的,并以此匹配最近的值历史序列进行预测. 不同于原有的 FCM 机制,Sliding FCM 没有计数器进行置信度计算,其结构如图 16 所示. Sliding FCM 将上下文的长度作为了一种静态的置信度,越长的上下文配置具有越高的预测置信度. 在验证值序列匹配值历史的情况下,Sliding FCM 在匹配的历史序列前指令窗口处取得预测值. 最终 Sliding FCM 通过混合 EVES 值预测器获得了 37.2% 的处理器 IPC 性能提升.

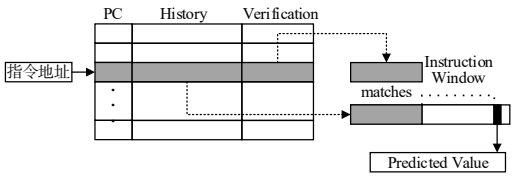

图 16 Sliding FCM 预测机制[72]

Koizumi 等人[78]也在第一届 CVP 值预测竞赛中提出了基于历史的高可靠混合值预测器(History Based Highly Reliable Hybrid Value Predictor,H3VP). H3VP 结合了 FCM 以及步幅预测器的预测方式,支持步幅、2 周期以及 3 周期重复的数值预测. H3VP 包含 3 个不同的预测表,分别对以上 3 种数值模式进行预测. 这里将其分类为基于上下文的值预测器是因为 H3VP 的 3 个预测表均是通过指令的局部值历史进行索引的,将局部值历史作为上下文进行预测. 同 Sliding FCM 一样,H3VP 同样参加的是无限资源赛道,但是其获得的处理器 IPC 性能提升是 10.0%,远低于 H3VP. 这主要是由于 EVES 值预测器能够很好地覆盖基于上下文的数值,而 H3VP 所支持的数值模式太少.

Sakhuja 等人[79~81]也在 CVP 值预测竞赛中通过结合分支历史和值历史信息,提出了基于异构上下文(Heterogeneous Context Value Predictor,HCVP)的值预测器. HCVP 预测器结构如图 17 所示,其通过获取指令在相同分支历史情况下最后一次提交的值,再通过分支历史进行 DFCM 的计算得到相应的步幅,最后通过相加得到最终的预测值. HCVP 通过和 ES 进行混合,最终 39.6% 的 IPC 性能提升获得了 2019 年 CVP 值预测竞赛无限资源赛道的第一名. HCVP 的成功说明更加准确的上下文信息能够帮助这类基于上下文的值预测器获得更好的性能.

4.2.4 基于上下文的预测器小结

基于上下文的值预测器总结在表 3 中,其中基于 TAGE[11]的值预测器是一类属于基于上下文的值预测器,将会在下一节详细展开描述. 从表 3 中可以总结出如下 4 点信息. 一是局部值历史是基于上下文的值预测器所主要利用的上下文信息,大部分值预测器均是

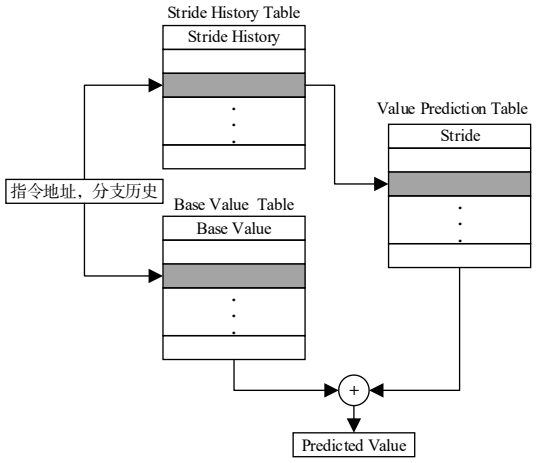

图 17　HCVP 预测机制[79]

在针对某条指令或者某一特定上下文情况下利用相应的局部值历史给出预测值的. 二是CVP值预测竞赛的举办极大地推动了相关预测器的发展,在CVP值预测竞赛至今的一段时间内,基于上下文的值预测器相关成果迅速增加. 这主要得益于CVP值预测竞赛中开放了论文成果以及开源了相应的值预测器源码,有助于研究人员快速验证和对比相应的想法. 三是该类预测器的研究还主要停留在理论阶段,许多预测器均是在无限资源条件下实现的,而并没有考虑值预测器实际的硬件执行时序和资源开销问题. 四是该类预测器所能带来的性能收益较大,许多值预测器带来的处理器IPC性能都超过了20%,这样的性能收益是不可忽略的,如何更加高效地利用数值的局部性,设计实现高效的值预测器结构,是亟待解决的问题.

表 3　基于上下文的值预测器

预测器	会议/期刊,提出时间	上下文	差分技术	饱和计数器	概率计数器[66]	特点	预测器开销	测试环境	测试程序	性能收益
FCM[16]	MICRO 1997	局部值历史	×	√	×	首次提出	∞	—	SPEC95	78% 预测准确率
DFCM[74]	HPCA 2001	局部值历史	√	√	×	使用差分技术	>12.5 KB	Simple Scalar[69]	SPEC95 int	比FCM的预测准确率提高6%~9%
DFCM++[77]	CVP 2018	局部值历史	√	√	×	提前更新策略,动态上下文长度	∞	CVP v6[12]	SPEC06/17数据库类加密程序	25.6% 处理器IPC性能提升
H3VP[78]	CVP 2018	局部值历史	√	√	×	混合预测器	32 KB	CVP v6[12]	SPEC06/17数据库类加密程序	4.3% 处理器IPC性能提升
Sliding FCM[72]	CVP 2019	局部值历史	×	×	×	匹配最近的上下文	∞	CVP v6[12]	SPEC06/17数据库类加密程序	>30% 预测覆盖率
STEVES[72] (Sliding FCM+EVES)	CVP 2019	局部值历史	×	×	√	匹配最近的上下文	∞	CVP v6[12]	SPEC06/17数据库类加密程序	37.2% 处理器IPC性能提升
HCVP[79]	CVP 2019	局部步幅历史,全局分支历史	×	√	×	结合分支历史和值历史作为上下文	∞	CVP v6[12]	SPEC06/17数据库类加密程序	39.6% 处理器IPC性能提升 59% 预测覆盖率 >99.9% 预测准确率
基于TAGE[11]的值预测器	(4.3 节)	全局分支历史,局部值历史	—	×	√	使用几何长度历史	—	—	—	—

4.3　基于TAGE的值预测器

4.3.1　传统TAGE预测器

常用于分支预测领域的TAGE预测器[11]因极其高效的资源利用率,得到了业界广泛的关注[82~84]. TAGE分支预测器的结构如图18所示,其通常使用几何历史长度的全局分支历史信息(图18 hist)对分支预测表Ti进行索引,使用的全局分支历史信息长度越长的分支预测表具有越高的输出权限.

ITTAGE(Indirect Target TAGE)[85]间接分支预测器也是基于TAGE的预测方法,表项结构如图19所示,其将间接跳转的目标地址,即Target,保存在预测器的表项之中,在预测时输出对应表项中的分支目标地址. 这

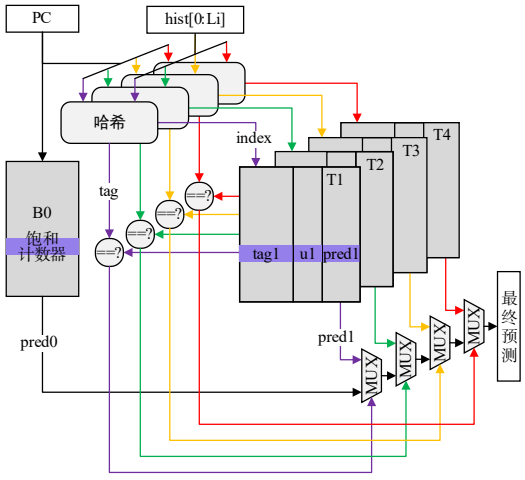

图 18 TAGE 预测器[11]

Target	Tag	Ctr	U

图 19 ITTAGE 表项结构[85]

一思想非常契合值预测器的目标功能. 值预测器是需要在特定的条件下输出特定的数值,而这里的间接分支目标地址可以看作间接分支指令的执行结果. 基于TAGE的值预测器是一类基于上下文的预测器,不同于FCM的上下文定义,TAGE中的上下文通常包含全局分支历史.

基于ITTAGE分支预测器,Perais等人提出了VTAGE(Value TAGE)值预测器[33,86,87]. VTAGE值预测器的提出具有重要意义,后续提出的很多值预测器都是基于VTAGE实现的. VTAGE本质上是一种基于上下文的LVP值预测器[72],其在具有相同上下文的情况下输出上一次相同上下文情况下产生的数值结果,本文根据上下文的索引方式将其归于本类.

4.3.2 VTAGE预测器存储优化

为了降低值预测器的硬件资源开销,Toshinori等人[88]很早就提出了通过降低数据位宽来减少预测器硬件开销的方式. 类似的,由于VTAGE中是直接将完整的32位或者64位数值保存在表项之中的,因此这部分的硬件资源消耗极其严重. 因此为了降低VTAGE的硬件成本,Perais等人同时参考了DFCM[74,77]的设计思想,尝试利用VTAGE预测步幅信息,而步幅的宽度通常就比较小,因此可以适当减少这部分步幅的数据宽度. 基于此,Perais等人[38]提出了D-VTAGE预测器. D-VTAGE的表项结构如图20所示,此外,还设计了一个利用PC索引的上一次值表,这个表专门负责记录每条指令上一次的数值结果,然后在预测时将其中的值和D-VTAGE预测的步幅,即Stride,进行相加得到最终的预测值. D-VTAGE的预测方式可以涵盖VTAGE和步幅预测器预测的指令,差分的预测技术极大地降低了预测器的存储开销,其硬件成本和复杂性也和现代分支预测器类似.

Stride	Tag	Ctr	U

图 20 D-VTAGE 表项结构[85]

而后,Seznec[17]在第一届CVP值预测竞赛中基于VTAGE和步幅预测器提出了EVES(Enhanced VTAGE and Enhanced Stride)预测器. EVTAGE值预测器结构如图21所示,VTAGE预测的指针将用于访问值表(Value TABLE,VTABLE)获得预测值. 相比于原有的VTAGE以及ITTAGE,EVTAGE主要从4个方面进行了增强. 其一是对所有的预测表都增加了tag表项,并且将预测表组织成了两路组相联的cache结构. 其二是降低了在预测器中保存的数据量. 其利用EVTAGE预测一个指针,这个指针指向一个专门负责保存数值的表,使用特定的算法对此表进行管理. 因此,一些原来在预测器中相同的数值就可以共用数值表的表项,从而实现硬件存储资源的降低. 其三是使用了概率计数器. 不同于饱和计数器,概率计数器可以在相同的位宽情况下提供更高的预测精度,在预测精度相同时消耗更少的硬件资源. 其四是根据指令类型概率性地为错误预测分配新表项,这有助于避免表项访问时出现冲突. Seznec提出的EVES值预测器也在当时获得了所有赛道的第一名,至今仍然保持着8 KB和32 KB赛道的最高IPC性能加速比的名次,是当今开源的最先进的值预测器. 这主要归结于其细粒度的存储空间管理算法,极为高效地利用了有限的存储空间实现了值预测. 而后续Sheikh等人[20]提出的混合Load预测器通过混合4种预测能力相互正交的值/地址预测器实现了比EVES更好的IPC性能加速效果,这主要是其通过地址预测的方式覆盖了部分值预测器所不能覆盖的范围实现的.

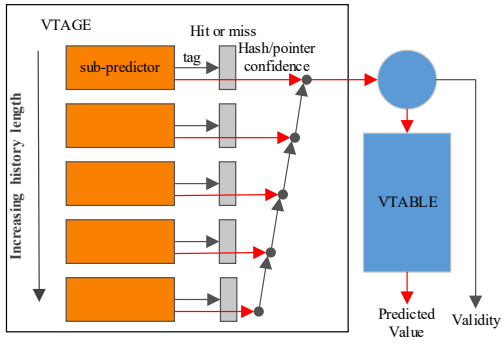

图 21 EVTAGE 预测机制[17]

4.3.3 VTAGE预测机制优化

VTAGE作为一种数值预测方法,可以更加巧妙地应用于其他数值模式预测. Ishii[21]在第一届CVP值预

测竞赛中尝试利用TAGE的方式实现对步幅数值模式的预测,进而提出了基于上下文和计算的(Context-Base Computational Value Predictor,CBC-VTAGE)值预测器.CBC-VTAGE的一种典型表项结构如图22所示,其中的指针可用于访问类似图21中VTABLE的值压缩缓存获得数值,对于较小的基值,也可以直接保存在表项中的Base部分.这种基于上下文同时预测数值和步幅的方式可以发现一些在全局上没有明显步长特性,但是在特定分支上下文情况下展现出的步幅特性,这种特性是VTAGE很难捕获的.同时,其为了降低硬件成本,其并不在VTAGE内部直接保存完整的数值,类似于EVTAGE,其保存一个指向另一个缓存的指针.这一缓存拥有专门的算法进行数据管理,能够有效降低值预测器的硬件资源开销.基于以上的策略,CBC-VTAGE在8 KB赛道上一直占据着第二名的成绩.

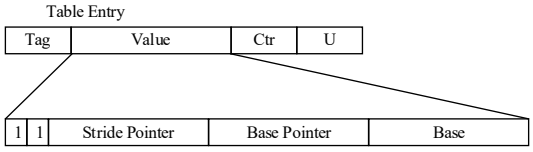

图22　CBC-VTAGE表项结构[21]

基于CBC-VTAGE,隋兵才[24]提出了利用历史上的预测情况反馈控制预测置信度的策略,以提高预测器的预测准确率,进而提出了名为RH-VTAGE的预测器.RH-VTAGE的相关反馈控制机制主要是进一步地利用历史上错误预测的信息降低类似情况下的错误预测率实现的,其在135个基准测试程序上实现了17.5%的几何平均IPC加速比,超过了CBC-VTAGE的15.5%.因此可以看出,通过进一步降低现有值预测的错误预测率的方式是可以进一步获得性能收益的.

同样基于VTAGE,Perais[89,90]通过只预测固定几个特殊数值进而实现预测性的硬件强度降低.强度降低(strength reduction)[89,91]是编译器设计中特别使用的一个术语,它包括选择能够实现程序员指定的语义的成本最低的操作.因此,强度降低通常被认为是编译时的优化,也可以将其称为静态强度降低.但是得益于值预测技术,硬件强度降低技术通过预测某些指令所需的特殊操作数,例如0或1,使指令在流水线前端就可快速得到执行结果,从而无需发送到流水线后端执行,即硬件层面的强度降低.例如一个具有操作数0的加法指令就可以通过寄存器重命名将目的寄存器直接设置成另一个源寄存器即可完成计算,无需其他额外计算操作.其提出的最小值预测(Minimal Value Prediction,MVP)只对0和1进行预测,最终使1.73%的动态指令在重命名阶段就执行结束.MVP成功实现的关键在于0和1这类特殊的数值在程序中出现的频率较高,同时这类特殊的数值又可以实现绝大部分指令的硬件强度降低.

4.3.4　VTAGE值相等机制优化

Kalaitzidis等人[19,22]首先提出了值相等预测的概念,即指令的结果一直或者周期性地表现出常量的特性,即周期性数值模式.为了更好地预测周期性数值模式,他们基于TAGE设计实现了VSEP(Value Speculation through Equality Prediction)预测器,VSEP由TAGE预测器[11]和最近提交值表(Last Committed Value Table,LCVT)组成.TAGE预测器[11]根据全局分支历史对值相等进行预测,并在预测相等时通过LCVT给出指令最近提交的数值作为预测值.VSEP也是一种基于上下文的LVP,其只保留了指令最后一次提交的数值,因此其不同于VTAGE预测器之处在于VSEP的预测值是指令上一次产生的结果,而VTAGE是上一次相同上下文情况下产生的结果.VSEP通过与其他预测器混合,最终在2020年的CVP值预测竞赛平台上获得了无限资源赛道第一名的成绩[71].

Ling等人[92]基于VSEP的工作,提出了步幅相等的概念.VSEP所预测的值相等其实是步幅等于0的步幅相等的特殊场景.基于此,他们提出了步幅相等预测器(Stride Equality Value Predictor,SEVP).SEVP预测器的结构如图23所示,其中,SE-TAGE负责给出预测步幅相等预测信号SEP(Stride Equality Prediction),LCVST负责保存指令上一次提交的数值及其步幅信息.SEVP在对周期性步幅数值模式实现更准确的预测的同时也保留了VSEP所关注的周期性数值模式的预测能力,因此在预测能力上SEVP预测器应当是可以完全覆盖VSEP的.

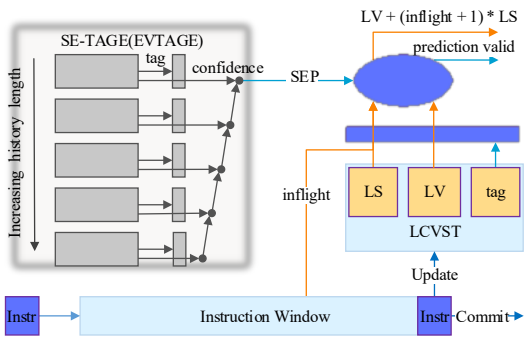

图23　SEVP预测机制[92]

Perais等人[93]则是根据当前指令的结果可能由之前的某条指令产生提出了寄存器相等预测.不同于预测指令的数值结果,寄存器相等预测通过距离预测器预测当前指令同指令窗口中的另一条指令的距离并利用寄存器共享完成指令的推测执行.由于指令窗口的

范围相比于数值的范围变化更小,因此相同表项下的TAGE距离预测器要比VTAGE值预测器小很多.以Store-Load指令对为例,传统的STLF转发技术可以将停留在Store队列中的Store数据转发给具有相同访存地址的Load指令.但是大多数情况下,Store指令所需要保存的数据往往来自更早的一条指令,这条指令的写回PRF的数值结果与Load指令写回PRF的数值结果是相等的.寄存器相等预测即通过预测两条指令写回目的寄存器中的数值相等,进而可以在重命名阶段通过共享物理寄存器实现Load访存开销的隐藏.这同Tullsen等人[94]提出的寄存器值预测RVP不同,RVP预测当前指令的结果已经存在其目的寄存器之中,更多的需要依赖编译器的支持.Ricardo等人[43]在此基础上为寄存器中添加了额外的地址标签,地址预测器通过更加准确的地址上下文信息实现了性能更好的寄存器共享与数据预取.

4.3.5 VTAGE预测器小结

几种具有代表性的基于TAGE的值预测器总结在表4之中.从表4中可以总结出如下2点信息:一是随着TAGE技术的引入,基于TAGE的值预测器在近十年内得到了迅速的发展,尤其自CVP值预测竞赛以来,基于VTAGE的相关研究一直在延续和发展;二是基于TAGE的值预测器的硬件资源开销正在逐渐降低,从数百KB逐渐降低至几十KB甚至是几KB,同时其还保留着一定的处理器IPC性能收益,即目前基于值预测研究的一个显著方向是如何更加高效地实现值预测器.

表4 基于TAGE的值预测器

预测器	会议/期刊,提出时间	主要特点	预测器开销	测试环境	测试程序	性能收益
VTAGE[33]	HPCA 2014	首次引入TAGE技术	132.7 KB	Gem5[95]	SPEC2k/06	>99.5%预测准确率 ≈50%预测覆盖率
D-VTAGE[38]	ISCA 2014	差分技术降低开销	256 KB	Gem5[95]	SPEC2k/06	>10%处理器IPC性能提升
EVTAGE[17]	CVP 2018	采用数值表压缩存储开销	31.36 KB	CVPv6[12]	SPEC06/17 数据库类 加密程序	11.2%处理器IPC性能提升 18.7%预测覆盖率
CBC-VTAGE[21]	CVP 2018	采用值压缩缓存降低开销	8 KB	CVPv6[12]	SPEC06/17 数据库类 加密程序	15.5%处理器IPC性能提升 >99.2%预测准确率
VSEP[19]	ICCD 2019	针对周期性相等数值	64 KB	Gem5[95]	SPEC06/17	5.8%处理器IPC性能提升 ≈54%预测覆盖率 (面向周期性相等) >99%预测准确率
MVP[90]	MICRO 2021	硬件强度降低[89]	7.9 KB	Gem5[95]	SPEC17	0.54%处理器IPC性能提升 ≈5.3%预测覆盖率 >99.9%预测准确率
TVP[90]	MICRO 2021	硬件强度降低[89]	13.9 KB	Gem5[95]	SPEC17	1.11%处理器IPC性能提升 ≈12.6%预测覆盖率 >99.9%预测准确率
SEVP[92]	CAL 2022	针对周期性相等步幅	47 KB	CVP2 v2.2[96]	SPEC06/17 数据库类 加密程序	处理器IPC性能比ES高5.3% >99%预测准确率

4.4 综合分析

根据以上的介绍可知,值预测器可以分为基于计算的和基于上下文的.目前基于计算的值预测器研究仍然以步幅预测器为主,通过结合TAGE值预测技术,其可预测空间进一步扩展,进一步提高处理器的性能.基于上下文的值预测器同样由于TAGE技术的引入在近年来得到了迅速的发展,通过充分利用全局分支历史信息,预测准确率有了较大的提升.另外由于预测准确率的提升,因此在近十年的值预测器研究中,处理器的IPC性能指标被常用于比较和评估值预测器的性能,例如2018年CVP值预测竞赛便是采用了这一指标作为排名依据.CVP值预测竞赛的举办极大地刺激了值预测器的发展,目前许多研究都是基于竞赛中所提出的值预测器展开的,这主要归结于比赛中开放的研究成

果使研究者可以快速复现和比较相应的值预测器.另外,通过结合以压缩数值宽度和值压缩缓存为代表的数值压缩技术,值预测器在硬件成本上得到了明显的降低.

5 值预测错误恢复机制

值预测器可以通过处理器的上下文给出对应的预测值,而现有的流水线架构却不能直接使用这一预测值.现有的流水线架构需要进行适当的修改以支持值预测,值预测正确时无需额外的操作,而预测错误时,则需要将处理器状态回滚到之前的状态.由于很难兼顾值预测较高的预测准确率和预测覆盖率,同时由于错误预测导致的性能惩罚较大,因此如何高效地恢复处理器状态一直是值预测研究的一个重要方向.在本节中,我们主要介绍支持值预测错误时的3种恢复机制:选择性重发射、流水线冲刷和软件值预测修补代码.

5.1 选择性重发射

Lipasti等人[13]首先在值预测研究中说明了重发射(reissue)的机制,即当值预测器对于指令的结果预测错误时,只利用正确的数据重新发射执行后续所依赖的指令,而对于其他没有依赖关系的指令,则没有任何额外操作,继续正常执行.这一策略也被后续的研究人员称为选择性重发射(selective reissue)[97,33,98]、重执行(re-execute)[99]或者选择性重放(selective replay)[34~36]策略.选择性重发射是一种处理器性能损失最低的值预测恢复机制,可以在发现错误预测后最快弥补性能损失.但是这一机制的实现也存在许多问题.首先是对于流水线控制逻辑设计的复杂性,不仅需要硬件动态监控指令的数据依赖链,而且需要发射单元保留已发射指令数据.此外,还需要备份每次值预测时的体系结构寄存器和物理寄存器之间的重命名映射表.一旦检测到错误预测的发生,相应的依赖指令将会被重新发射执行.总的来说,选择性重发射机制造成的性能损失最小,但是会带来额外的资源开销以及复杂的逻辑设计,因此这一策略主要面向的应当是高端的桌面级和服务级处理器.

5.2 流水线冲刷

Lipasti等人[14]同样在值预测研究中首先说明了流水线冲刷的机制,其在名为Alpha AXP 21164的顺序处理器模型中使用了这一机制,当发现值预测错误时,流水线中后续的指令将会被全部冲刷并重新发射执行.显然,流水线冲刷是一种同分支预测错误恢复机制相似的机制,因此在现有流水线中比较容易实现,这也是这一机制的优点.但是这一机制的缺点也很明显,即错误预测的惩罚比选择性重发射机制要大很多.错误预测指令之后的所有指令都会被重新发射执行,流水线需要更多的时间实现填充.同时Balkan等人[100]还发现值预测会对分支预测造成影响,进而造成错误的流水线冲刷.Rami等人[40]的研究也发现,将流水线冲刷机制替换为选择性重发射后,CAP[46],DLVP[40],VTAGE[33]值预测器带来的处理器性能收益分别提高了1.9%,0.7%,0.7%,CAP的预测准确率较低,因此其所获得的性能提升也最大,但是随着值预测器预测准确率的不断提高,例如DLVP和VTAGE,流水线冲刷机制造成的性能损失也在逐渐降低.

值预测错误时,还会存在一种不进行任何恢复操作的特殊情况,即所预测的值没有被任何后续依赖指令所使用的情况.虽然值预测的目标是让预测值作为后续具有数据依赖关系的指令,提前满足执行条件,但是现有值预测器还并不能对其预测值是否会被后续指令所使用实现预测.在后续指令均没有使用预测值的情况下,错误的预测值并没有对更多的处理器状态造成污染,因此只需将正确的值写回原来的目的寄存器即可.这种情况下值预测错误时不会造成性能的惩罚,但是在预测正确时也不会获得性能收益.

这一特殊的处理操作可以看作选择性重发射机制的一种特殊情况,可同时用于优化流水线冲刷机制.当错误预测的结果并没有被后续任何指令所使用的情况下,不需要进行任何的流水线冲刷操作,继续正常执行即可;而当后续指令使用了预测值的情况下,则需要进行相应的错误恢复机制.这一策略也被现在的一些研究所采用[33],为方便起见,本文将此策略称为无依赖条件下的零惩罚策略.无依赖条件下的零惩罚策略是在设计错误恢复机制时可选的一种优化机制,这一机制需要在硬件上支持对于所预测的指令的数据依赖检测功能,这一检测功能没有选择性重发射机制那么复杂,不需要明确哪一条指令的哪一个操作数使用了预测值,而只需要检测是否有后续指令使用了预测值.只有在检测到后续没有指令使用预测值时,才能发挥这一机制的作用.因此对于流水线冲刷机制而言,需要一些额外的硬件开销,才能支持这一优化策略的实现.

虽然无依赖条件下的零惩罚策略可以用于优化流水线冲刷机制,避免一些无用的冲刷操作,但是就这一机制的额外资源开销以及所能获得的具体性能收益而言,现有的文献都没有给出明确的说明.对此,也期望后续的科研工作者能够对此进行进一步的实验论证,以支持和推动值预测错误恢复机制的进一步发展.

5.3 软件修补代码

软件修补代码(patch-up code)通过编译器实现[101,102].这一错误恢复机制通过软件实现,无须硬件

恢复模块的支持. 但需要预先定义好需要进行值预测的指令,然后首先通过专用预测指令获取预测器的预测值并进行后续指令的软件调度,最后通过比较原有指令与专用预测指令的结果来判断是否实现正确预测. 在值预测错误的情况下跳转到相应的软件修补代码执行以实现处理器状态的恢复,执行完修补代码后再跳转回正常的程序执行流. 图24所示为修补代码示例,其中图24(*a*)为原始代码,图24(*b*)为实现了软件值预测的代码. I8指令(LDPRED指令)将I3指令的预测结果写入了R8,是一条专用预测指令. 指令I8对I3指令的执行结果同预测的结果进行比较,如果二者不一致,则表示值预测错误,此时跳转至修补代码块进行处理器状态恢复. 可以看到,Patch-up代码块重新执行了错误预测的指令后的所有指令,并在执行完毕后返回了原有的指令流.

```
(a) Original code
I1:     ADD     R1 ← R2, 5
I2:     SHL     R3 ← R1, 2
I3:     LW      R4 ← O(R3)
I4:     ADD     R5 ← R4, 1
I5:     OR      R6 ← R5, R7
I6:     SW      O(R3) ← R6
Next:   ...

(b) New code after value speculation of R4 (predicted instruction I3)
I1:     ADD                 R1 ← R2, 5
I2:     SHL                 R3 ← R1, 2
I3:     LW                  R4 ← O(R3)
I7:     LDPRED              R8 ← index // load prediction
into R8
I4:     ADD                 R5 ← R8, 1
I5:     OR                  R6 ← R5, R7
I6:     SW                  O(R3) ← R6
I8:     BNE Patchup  R8, R4 // verify prediction
Next:   ...

Patchup:
I9:     UDPRED              R4, Index     // update
predictor with R4
I4:     ADD                 R5 ← R4, 1
I5:     OR                  R6 ← R5, R7
I6:     SW                  O(R3) ← R6
I10:    JMP Next
```

图24 软件修补代码示例[101]

软件修补代码的方式需要对现有程序重新编译. 预先定义好具体哪些指令可以进行值预测,并且需要增加额外的代码,增加的代码长短取决于所预测的指令延迟大小. 同时由于编译器需要通过专用指令同硬件值预测器交互,软硬件的设计复杂度都较高,因此该方式目前使用较少.

5.4 综合分析

值预测的错误恢复机制总结在表5之中. 从表5中可以总结出如下2点信息:一是业界比较认可现有的错误恢复机制,针对值预测错误恢复的相关研究比较少;二是各种错误恢复机制各有优缺点,要么成本高,要么设计复杂,要么需要软件重新编译,在选择错误恢复机制时需要综合考量性价比.

表5 值预测错误恢复机制

预测器	会议/期刊,提出时间	恢复时间	设计复杂度	所需资源	处理器性能收益比较[40]	应用场景
选择性重发射[13]	MICRO 1996	低	高	高	CAP:4.2% DLVP:5.5% VTAGE:2.8%	性能需求高
流水线冲刷[14]	ASPLOS 1996	高	低	低	CAP:2.3% DLVP:4.8% VTAGE:2.1%	预测精度高
可选的无依赖条件下的零惩罚策略[36]	MICRO 2015	最低	较低	较低	—	可优化流水线冲刷
软件修复代码[101]	ASPLOS 1998	较高	较高	无	—	流水线修改少

目前常用的流水线冲刷机制设计简单,容易实现,但是其造成的性能损失也是最大的,而性能损失最小的选择性重发射机制的成本开销很大且设计复杂度较大. 为了加快推进值预测技术的研究发展以及其在商用处理器中落地实现,业界更多地采用了设计简单的流水线冲刷机制. 而面向流水线冲刷机制下较大的值预测错误性能损失问题,业界更倾向于具有更低值预测错误预测率的发展方向. 例如CVP值预测竞赛第一名[71]的错误预测率已经达到0.061%. 错误预测率几乎可以忽略,因此流水线冲刷所带来的性能损失也逐渐减少. Perais等人[33]的研究也表明,在错误预测率很低的情况下,即使是完美的0周期选择性错误恢复机制,即在发现错误的同时重新发射后续使用了错误预测值的指令,也并没有明显的性能提升. 另外由于可以采用无依赖条件下的零惩罚策略来优化整体的错误预测惩罚,因此流水线冲刷这一简洁的恢复机制也逐渐被人们所接受[103]. 总的来说,复杂的选择性重发射机制难以实现,因此,现有的工作[38,20,18,22,19]更多地采用了流水线冲刷的机制. 在流水线冲刷机制下,值预测器的准确率不断提高,流水线冲刷机制造成的性能损失也在

逐渐降低. 但是极低的错误预测率是以牺牲预测覆盖率为代价的,通过设置较高的值预测输出阈值,能够屏蔽大部分预测准确率较低的值预测输出,这限制着值预测技术所能带来的最大性能收益. 因此,业界亟需一种恢复时间较短、设计复杂度较低且易于实现的高效值预测错误恢复机制. 这样的机制能够在适当降低当前值预测输出阈值的情况下提高值预测的预测覆盖率,从而进一步提高处理器的性能.

6　未来研究展望

基于TAGE的分支预测技术目前通过结合基于计算的和基于上下文的预测方式,已经取得了阶段性的成果,未来如何更进一步地开发值预测技术,仍然是一个重要的主题,这里对值预测未来可能的研究方向做出几点展望.

6.1　实用化值预测技术研究

目前值预测器还没有在处理器中实现,主要还是由于预测器的成本较高并且相关设计复杂,需要对流水线的设计进行修改. 许多值预测相关研究都在关注这一问题. 早期降低值预测器硬件资源开销的策略主要包括预测值宽度降低[88]和比特共享(bit sharing)[104],即降低数值在预测器中的存储开销,这些其实都属于通过数值压缩[21]降低成本的方式. 此外,由于值预测需要对流水线进行多处修改,因此,在处理器设计的早期阶段就需要确定值预测的方案,并进行相关的设计和验证工作. 低成本的值预测方案可以确保值预测器能够与处理器的其他部分协同工作,并在实际应用中发挥作用.

除了硬件上降低成本的方式,Martin[105]等人还提出的在编译器上对Load指令进行分类的方法,其只让预测器预测那些可能发生Cache未命中,并且预测器容易预测正确的Load指令. Cache未命中的Load指令将会从更低一级的存储中获取数据,及其占用处理器时间. 因此这一策略可以避免许多的值预测器存储空间浪费,最终获得了最高8%的会造成Cache未命中的Load指令预测精度提升. 类似的还有Gellert等人[106]的研究,但是他们是从功耗降低的角度去分析的. Bandishte等人[18]只针对关键路径上的Load指令进行预测,而对于其他指令,都不进行预测,因此也不需要过多的存储资源开销. 最终其在1.2 KB的硬件成本下获得了3.3%的处理器平均IPC性能提升. Perais[89,90]也通过只预测固定几个特殊数值(例如0,1)进而实现预测性的硬件强度降低. 类似的还有Islam等人[107]的研究,他们通过跟踪记录访问过的0的地址,在后续某些访问指令进入存储系统前取消这些访问请求. 这样的方式可以缩短这部分访存指令的执行时间,同时也可以降低存储系统的执行压力.

此外,值预测器本身的存储资源以及值预测在流水线中引入的额外操作,尤其是错误恢复操作也给处理器带来了不可忽略的功率开销[108]. 业界也较早地意识到了这个问题,提出了一些高能效的设计,主要通过降低值预测器的复杂度与减少错误恢复的次数[109]实现动态功耗的降低,抑或是通过自适应地禁用预测表中未使用的条目实现静态功耗的降低[110~112]. Martin等人[113]则是通过预测器之间的状态共享与简单的数值压缩降低混合值预测器的大小.

值预测器实用化过程中除了低成本、轻量化、功耗等问题外,还存在着一些潜在的实用化问题等待探索,例如,在真实处理器中多线程切换时应当适当提高值预测器的输出阈值,甚至关闭值预测器的预测功能,以避免上下文切换导致大量错误值预测的情况. 因为不同进程操作的数据通常不同,上下文切换会导致值预测器索引过程中出现大量冲突. 换言之,值预测器在使用其他进程或较旧的历史信息进行值预测时通常会导致大量的错误预测,严重影响计算机的整体性能. 因此,需要在线程切换时采用一定的方法策略限制值预测器的预测功能,并采用一定的算法在新线程切换一段时间后重新启用值预测器的预测功能,例如,在固定的值预测器训练时间后启用.

总的来说,值预测技术的实用化研究将一直是值预测未来发展的一个重点方向,这一方面可以推动值预测技术的最终落地,另一方面也会有利于其在嵌入式等真实应用场景下的广泛应用.

6.2　值预测空间探索研究

由于值预测错误恢复机制的影响,目前值预测器的实际预测覆盖率还不算很高,根据CVP值预测竞赛无限资源赛道第一名[71]的数据分析可知,在针对SPEC'06/'17等通用测试程序的场景下,目前预测能力最强的值预测器在针对所有类型的指令进行预测时预测覆盖率也只达到了63.9%,因此还有很大一部分预测空间等待探索. 这部分预测空间是目前的预测算法所不能覆盖的,未来还需要提出一些全新的预测方式以实现预测能力质变的提升,这些预测方式应该是和现有值预测技术所能覆盖的预测空间是互补的或者是正交的,抑或在现有的预测空间基础上覆盖更宽广的区域.

例如,在CVP值预测竞赛的无限资源赛道上,Arpit等人[72]通过混合Sliding FCM和EVES值预测器获得了比EVES预测器更高的处理器IPC性能加速效果. Kleovoulos等人[19]提出的VSEP值预测器的覆盖范围和现有的VTAGE是部分正交,因此其通过混合两种值预测器获得了平均19%的加速效果. 而后André等人[71]通过

混合ES/HCVP/VTAGE/VSEP值预测器更是获得了45.0%的处理器IPC性能加速.

总的来说,值预测的预测空间探索工作对值预测技术的发展具有重要意义.通过研究分析不同类型的值预测空间占比,可以设计出更加高效的值预测器,同时也能促进其他相关领域的发展,例如地址预测、数据预取等.

6.3 多领域的异构融合研究

多领域融合发展是未来可能的发展方向之一.值预测可以作为一种提高处理器性能的手段,同时也可以作为一种工具或者方法.因此将值预测同其他领域进行异构融合也许能创新性地解决许多难题.

从存储墙的角度来看,数据预取与值预测的功能极其相似,都是为了尽快提供指令所需的源操作数.但是从超标量流水线架构的角度,值预测可以解决数据依赖造成的调度问题,可以让更多的指令同时发射执行.因此,值预测与数据预取所能提供的性能收益应当是部分正交的,可以通过混合这两种技术实现更高的处理器性能,这也是支持Load地址预测的流水线架构所常用的方式.在Sheikh等人[20]的数据中也可以看出34%的Load指令预测只能通过用于预取的地址预测或是值预测所完成,而剩余的指令则是二者均可完成.此外Orosa等人[41,42]也将值预测同数据预取相结合,将预测的地址作为值预测的上下文,最终获得了11.2%的处理器性能加速.

Wang等人[114]则是通过结合近似计算[115]与值预测,在通用GPU中利用地址步幅辅助值预测,最终也获得了可观的预测覆盖率.Walid等人[116]通过结合基于强度的动态信息流分析判断出值预测器何时的预测更准确.Perais[89,90]则是将硬件强度降低同值预测结合,旨在利用预测值降低硬件计算的复杂性,进而获得性能收益并能节省功耗开销.

因此,通过将值预测技术同其他领域相结合,能够创新性地解决一些现有的问题,同时也能开辟一些新领域(例如Perais[89]提出的预测性硬件强度降低),最终推动微体系结构的不断发展.例如,当今研究较热的提前执行技术[117~119]已经在分支预测领域[120]获得了突破,那么同样可以尝试利用提前执行模式下的部分数据实现更准确的值预测.

6.4 基于机器学习的值预测技术研究

机器学习相关预测技术已经在分支预测等微体系结构领域逐渐发展起来[121~123],例如Sweety等人[124]利用神经网络构建的分支预测器有效地提高了分支预测的精度,类似的预测方式也可以直接牵引到类似值相等预测[22]等方法之中.Manel等人[125]在数据预取配置中引入深度学习方法,在多个程序运行时利用该方法尝试寻找最优的预取配置,最终也成功在多个应用负载下获得了明显的性能提升.

不可否认的是机器学习方法的有效性,该方法逐渐在微体系结构领域发展起来.由于巨大的神经网络模型在实际的流水线中很难部署,因此在值预测领域的相关研究目前还比较少,只有少数研究尝试利用神经网络方法对预测器的置信度进行管理[126~128],但这些研究都比较初步,未来还具有很大的研究空间.例如,不仅仅是针对值预测的置信度进行神经网络方法预测,对预测值也可以尝试采用类似的方法.另外,如何降低预测模型的规模大小以及如何在硬件上实现快速的模型推理计算等都是非常具有研究价值的问题.

6.5 值预测流水线架构研究

支持值预测的低成本流水线架构.值预测的支持对于原有流水线的修改以及成本的增加较大,因此,如何在低成本的情况下快速地实现对值预测的支持是让值预测技术落地的一个重要因素.目前所提出的EOLE,BeBoP等架构[37,38]均在降低原有流水线成本方面做了很多的工作,主要的研究点在于如何降低值预测器带来的物理寄存器文件的资源开销以及值预测器自身的存储开销.而后,Sheikh等人[20]提出使用专用的片上存储保存预测值的方式避免了额外的PRF读写端口需求,但同时由于仲裁机制的引入,因此会在关键路径上引起延迟的增加.另外,Ravi等人[129]提出的值预测器延迟问题也是一大挑战.由于值预测器的存储资源较大,那么如何在高频处理器流水线中设计实现值预测器对高性能处理器而言就具有重要的意义.

此外,如何将值预测技术同现有的其他指令级优化技术相结合也是流水线架构研究的一个方向,例如Sheikh等人[20]提出的同时支持地址预测与值预测的流水线架构很好地结合了这两种预测技术;Sudhanshu等人[41]将物理寄存器文件预取同值预测相结合获得了超过它们任意一种技术所带来的性能收益.因此对于多种指令级优化技术而言,可以考虑的一个方向是设计高效的流水线架构将它们整合在一起.在此过程中还需要考虑多种技术之间存在的重叠优化问题,例如数据预取与值预测均能对部分Load指令实现优化,但同时也存在各自所独有的性能收益.因此需要在多种流水线架构优化技术之间做好权衡,从而可以获得更好的处理器性能、更高的存储资源利用率和更低的功率消耗.

6.6 值预测安全问题研究

在过去几年中,瞬态执行攻击[130],也称为推测执行攻击,引起了人们的极大兴趣,因为它们会导致关键

数据泄露. 处理器的推测执行不会改变处理器的体系结构状态,任何可能影响体系结构状态的事件都将通过错误预测恢复机制被消除,例如流水线冲刷. 然而,推测执行却会改变处理器的微体系结构状态,例如Cache中的数据. 处理器在沿着错误的路径推测执行时会被立即恢复,因此称该种情况为瞬态执行. 瞬态执行攻击由两个主要组成部分组成:瞬态执行本身和用于实际过滤信息的隐蔽通道. 隐蔽通道是微架构组件的时间和空间共享而导致的非故意信息泄漏通道. 在处理器的实际运行过程中,处理器可能处于不同的状态,如特权状态和用户状态. 特权状态通常由操作系统维护,该状态的程序可以访问所有用户数据和受保护的特权数据. 值预测是一种新型的推测执行技术,如果值预测器学习到了这种特权状态下的访存数据,那么用户状态下的程序就可能会利用类似的程序上下文获取到受保护的数据. 为了解决这个问题,一种简单的解决方案是在关键的处理器状态关闭值预测器的预测和更新功能,然后在恢复为用户状态后重新开启值预测器的预测和更新功能.

研究值预测技术在推测执行过程中造成的处理器微体系结构状态变化对处理器的数据安全具有重要意义. 在后续的研究过程中,人们提出了许多的解决方案. 例如,Yan等人[131]提出的一种称为InvisiSpec的方法,可以隐藏由于推测执行导致的微结构状态的变化. Zhao等人[132]提出的InvastSpec则是使用了一种结合编译器和硬件的方案. 编译器分析程序的控制流和数据流,为每条指令识别一组安全指令,用以保证指令的安全执行. 但即使如此,其也会带来性能损失和功耗开销[133]. Christos等人[134]则是通过利用值预测技术来避免由于预测性的访存造成的Cache数据变化,而后利用正确预测下的延迟访存[135]来保证程序的正确性.

除了值预测技术本身可能引入的安全问题以外,一种新颖的想法是利用值预测技术解决某些其他安全问题. 故障攻击是一种常见的低成本攻击手段[136],其通过利用故障注入可以绕过安全机制. Sheikh等人[137~139]首次利用值预测技术解决故障攻击问题,他们将故障检测逻辑和反应逻辑同值预测器结合,通过对预测值的安全性进行自我验证,可以减轻对数据路径和值预测器本身的故障攻击.

总的来说,安全领域对于计算机领域的发展至关重要,而值预测器技术的使用给安全领域带来了新的挑战的同时也带来了新的机遇. 未来如何避免值预测技术泄露数据以及如何在安全防御中利用值预测技术都具有非常重要的研究与应用价值.

6.7　值预测编译器优化技术研究

通过编译器优化值预测的性能也是未来发展的一个方向. Zhao等人[140]很早就提出了利用编译器优化值预测的想法,其根据不同的程序行为将指令划分成多个指令块,通过专用指令与特定的值预测器交互,利用特定的方式生成预测值. 这样的方式可以提高值预测器的利用率,避免多个值预测器保存相同的冗余预测信息. 类似的,Nana等人[141]通过硬件计数器为每条Load指令分配预测组件的方式提高预测器的效率. 而后,Fernando等人[37]曾在EOLE架构上对编译器的相关优化手段进行了分析,其发现某些传统编译器的优化手段会降低值预测的性能,一些编译器优化较少的程序(例如指令调度)反而能从值预测中获得更多的性能收益. 因此值预测技术的引入会对传统的编译器优化技术提出新的要求.

综上,如何量化传统编译优化技术的收益与值预测带来的收益和错误预测惩罚,并以此指导编译器的工作,以获得最大化的处理器性能收益,是一个极具潜力的发展方向. 值预测编译器优化技术将在值预测技术普及的未来提供额外的性能增长空间.

6.8　值预测错误恢复机制研究

在当前的值预测研究中,大多数值预测器的预测精度非常高. 因此,许多研究都采用了流水线冲刷的错误恢复机制. 然而,从另一个角度来看,这种易于实现的错误恢复机制会带来较大的性能惩罚. 值预测器通过牺牲预测覆盖率为代价来提高预测精度,以获得整体处理器性能的收益. 如果能够设计一种极其高效的值预测错误恢复机制,其设计复杂度和资源需求与流水线冲刷机制类似,并且错误恢复时间类似于选择性重发射机制,那么值预测器就可以适当降低输出阈值,从而缓解预测覆盖率与预测准确率之间的冲突,进一步提高处理器性能.

值预测错误恢复机制与值预测器相比,具有较高的解耦性,可以进行独立的研究. 在错误恢复机制方面获得的性能收益可以直接应用于几乎所有的值预测器上,全面提升当前值预测技术对处理器带来的性能收益,因此具有重要的研究意义. 目前,一个可探索的方向是研究无需进行错误恢复操作的机制,例如Zhou等人[142]的研究中提到的一种名为“无需恢复的值预测”. 该方式利用值预测加强存储级并行,对于Load访存指令,能够提前将所需的数据取到缓存中,并对所有推测执行的指令进行非推测执行,以消除原有推测执行对处理器状态所造成的影响. 然而,从本质上来看,这实际上是一种默认都进行选择性重发射恢复的策略,它不仅会造成计算资源竞争压力的增加,还会对缓存的带宽造成较大的压力.

综上所述,值预测的错误恢复机制研究对值预测技术的发展具有重要意义.业界应该在流水线冲刷和选择性重发射机制的基础上,研究资源开销小、错误预测惩罚低的恢复机制.

7 总结

通过对处理器值预测技术的系统性讨论,可以发现处理器值预测技术经过了二十多年的发展,目前基本已经形成了一套较为完整的技术体系:值预测流水线架构为值预测器提供了一个完整的预测和更新条件,基于上下文的和基于计算的值预测器负责提供预测值,错误恢复机制负责消除错误路径上的指令对处理器状态的影响.

虽然在21世纪初,由于巨大的成本因素,值预测技术迟迟不能在现实的处理器中实现,同时基于计算的值预测器研究在预测性能方面迟迟没有突破性进展导致人们对值预测技术的预期降低.但是各种高效数值压缩技术、TAGE预测技术的提出以及CVP值预测竞赛的成功举办,为值预测技术的发展提供了新的动力,分别在基于上下文的和基于计算的等各个值预测领域有了突破.基于TAGE的值预测技术由于其极其高效的空间使用率以及准确的预测性能得到了迅速的发展.另外,值预测作为一种数值预测工作,在与数据预取、硬件强度降低等领域的融合过程中,也出现了一些极为有效的流水线架构,极大地刺激了值预测技术的应用以及发展.

最后,本文还分别从实用化、预测安全、编译器优化等8个方面对处理器值预测技术未来的发展做出了展望.值预测领域还有许多空白研究等待填补,还存在一些实用化问题等待发现,期望本文针对该技术的研究能够为关注处理器值预测相关技术的研究者提供思路和借鉴.

参考文献

[1] RAJBHANDARI S, RUWASE O, RASLEY J, et al. Zero-infinity: Breaking the GPU memory wall for extreme scale deep learning[C]//Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. New York: ACM, 2021: 1-14.

[2] VAVOULIOTIS G, CHACON G, ALVAREZ L, et al. Page size aware cache prefetching[C]//2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO). Piscataway: IEEE, 2022: 956-974.

[3] LEE S, KANG S H, LEE J, et al. Hardware architecture and software stack for PIM based on commercial DRAM technology: Industrial product[C]//Proceedings of the 48th Annual International Symposium on Computer Architecture. New York: IEEE, 2021: 43-56.

[4] KE L, ZHANG X, SO J, et al. Near-memory processing in action: Accelerating personalized recommendation with AxDIMM[J]. IEEE Micro, 2022, 42(1): 116-127.

[5] JEONG I, LEE J, YOON M K, et al. Reconstructing out-of-order issue queue[C]//Proceedings of the 55th Annual IEEE/ACM International Symposium on Microarchitecture. New York: ACM, 2022: 144-161.

[6] DOWECK J, KAO W F, LU A K Y, et al. Inside 6th-generation Intel core: New microarchitecture code-named skylake[J]. IEEE Micro, 2017, 37(2): 52-62.

[7] MATTIOLI M. Meet the FaM1ly[J]. IEEE Micro, 2022, 42(3): 78-84.

[8] HU W W, ZHANG F X, LI Z S. Microarchitecture of the godson-2 processor[J]. Journal of Computer Science and Technology, 2005, 20(2): 243-249.

[9] 张福新. 微处理器性能分析与优化[D]. 北京: 中国科学院研究生院(计算技术研究所), 2005.
ZHANG F X. Performance Analysis and Optimizations of Microprocessors[D]. Beijing: Graduate School of Chinese Academy of Sciences (Institute of Computing Technology), 2005. (in Chinese)

[10] ZHAO J, KORPAN B, GONZALEZ A, et al. Sonicboom: The 3rd generation berkeley out-of-order machine [EB/OL]. (2020-05-29) [2023-03-01]. https://carrv. github. io/2020/papers/CARRV2020_paper_15_Zhao.pdf.

[11] SEZNEC A, MICHAUD P. A case for (partially) TAgged GEometric history length branch prediction[J]. The Journal of Instruction-Level Parallelism, 2006, 8: 23.

[12] PERAIS A. First championship value prediction (CVP-1)[EB/OL]. (2018-06-03)[2023-03-01]. https://microarch.org/cvp1/cvp1online/contestants.html.

[13] LIPASTI M H, SHEN J P. Exceeding the dataflow limit via value prediction[C]//Proceedings of the 29th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). New York: ACM, 1996: 226-237.

[14] LIPASTI M H, WILKERSON C B, SHEN J P. Value locality and load value prediction[C]//Proceedings of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 1996: 138-147.

[15] GABBAY F, MENDELSON A. Speculative Execution Based on Value Prediction[R]. Haifa: Technion-Israel Institute of Technology, Department of Electrical Engineering, 1996.

[16] SAZEIDES Y, SMITH J E. The predictability of data values [C]//Proceedings of the 30th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). New York: ACM, 1997: 248-258.

[17] SEZNEC A. Exploring value prediction with the EVES predictor[C]//CVP-1 2018-1st Championship Value Prediction. Piscataway: IEEE, 2018: 1-6.

[18] BANDISHTE S, GAUR J, SPERBER Z, et al. Focused value prediction: Concepts, techniques and implementations presented in this paper are subject matter of pending patent applications, which have been filed by Intel Corporation[C]// 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA). Piscataway: IEEE, 2020: 79-91.

[19] KALAITZIDIS K, SEZNEC A. Value speculation through equality prediction[C]//2019 IEEE 37th International Conference on Computer Design (ICCD). Piscataway: IEEE, 2019: 694-697.

[20] SHEIKH R, HOWER D. Efficient load value prediction using multiple predictors and filters[C]//2019 IEEE International Symposium on High Performance Computer Architecture (HPCA). Piscataway: IEEE, 2019: 454-465.

[21] ISHII Y. Context-base computational value prediction with value compression[C]//CVP-1 2018-1st Championship Value Prediction. Piscataway: IEEE, 2018: 1.

[22] KALAITZIDIS K, SEZNEC A. Leveraging value equality prediction for value speculation[J]. ACM Transactions on Architecture and Code Optimization, 2020, 18(1): 1-20.

[23] 肖勇. 值预测技术研究[D]. 长沙: 国防科学技术大学, 2005.
XIAO Y. Research on Data Value Prediction[D]. Changsha: National University of Defense Technology, 2005. (in Chinese)

[24] 隋兵才. 基于真实历史反馈的自适应值预测器的设计与优化[J]. 计算机工程与科学, 2021, 43(2): 274-279.
SUI B C. Design and improvement of self-adaptive value predictor based on real history feedback[J]. Computer Engineering & Science, 2021, 43(2): 274-279. (in Chinese)

[25] 冀蓉, 周宏伟, 张民选, 等. 推测执行中值预测与指令重用技术的研究与分析[J]. 计算机工程与科学, 2005, 27(11): 98-101.
JI R, ZHOU H W, ZHANG M X, et al. Research and analysis of the value prediction and instruction reuse techniques in speculated execution[J]. Computer Engineering & Science, 2005, 27(11): 98-101. (in Chinese)

[26] 李笑天, 郭德源, 何虎. 分支预测与值预测在VLIW处理器中的实现[J]. 微电子学与计算机, 2015, 32(1): 54-59.
LI X T, GUO D Y, HE H. Realization of branch prediction and value prediction in VLIW[J]. Microelectronics & Computer, 2015, 32(1): 54-59. (in Chinese)

[27] 冀蓉, 张民选, 陈怒兴. 值预测技术中基本值预测模型的功耗分析[J]. 计算机工程与科学, 2006, 28(4): 126-129.
JI R, ZHANG M X, CHEN N X. A power consumption analysis of the basic value prediction models in value prediction techniques[J]. Computer Engineering & Science, 2006, 28(4): 126-129. (in Chinese)

[28] 党向磊, 王箫音, 佟冬, 等. 一种基于值预测和指令复用的按序处理器预执行机制[J]. 电子学报, 2011, 39(12): 2880-2883.
DANG X L, WANG X Y, TONG D, et al. A pre-execution mechanism based on value prediction and instruction reuse for in-order processors[J]. Acta Electonica Sinica, 2011, 39 (12): 2880. (in Chinese)

[29] HENNESSY J L, PATTERSON D A. Computer Architecture: A Quantitative Approach[M]. Cambridge: Elsevier, 2011.

[30] GALLAGHER D M, CHEN W Y, MAHLKE S A, et al. Dynamic memory disambiguation using the memory conflict buffer[J]. ACM SIGPLAN Notices, 1994, 29(11): 183-193.

[31] KIRIANSKY V , WALDSPURGER C .Speculative buffer overflows: Attacks and defenses [EB/OL]. (2018-07-10) [2023-03-01]. https://arxiv.org/pdf/1807.03757.pdf.

[32] PONCE-DE-LEON H, KINDER J. Cats vs. spectre: An axiomatic approach to modeling speculative execution attacks[C]//2022 IEEE Symposium on Security and Privacy (SP). Piscataway: IEEE, 2022: 235-248.

[33] PERAIS A, SEZNEC A. Practical data value speculation for future high-end processors[C]//2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA). Piscataway: IEEE, 2014: 428-439.

[34] PERAIS A, SEZNEC A. EOLE: Paving the way for an effective implementation of value prediction[C]//2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA). Piscataway: IEEE, 2014: 481-492.

[35] PERAIS A, SEZNEC A. EOLE: Combining static and dynamic scheduling through value prediction to reduce complexity and increase performance[J]. ACM Transactions on Computer Systems, 2016, 34(2): 4.

[36] PERAIS A, SEZNEC A. EOLE: Toward a practical implementation of value prediction[J]. IEEE Micro, 2015, 35(3): 114-124.

[37] ENDO F A, PERAIS A, SEZNEC A. On the interactions between value prediction and compiler optimizations in the

context of EOLE[J]. ACM Transactions on Architecture and Code Optimization, 2017, 14(2): 18.

[38] PERAIS A, SEZNEC A. BeBoP: A cost effective predictor infrastructure for superscalar value prediction[C]//2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA). Piscataway: IEEE, 2015: 13-25.

[39] PERAIS A. Increasing the Performance of Superscalar Processors Through Value Prediction[D]. Rennes: Université de Rennes, 2015.

[40] SHEIKH R, CAIN H W, DAMODARAN R. Load value prediction via path-based address prediction: Avoiding mispredictions due to conflicting stores[C]//Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). New York: ACM, 2017: 423-435.

[41] SHUKLA S, BANDISHTE S, GAUR J, et al. Register file prefetching[C]//Proceedings of the 49th Annual International Symposium on Computer Architecture. New York: ACM, 2022: 410-423.

[42] OROSA L, AZEVEDO R, MUTLU O. AVPP: Address-first value-next predictor with value prefetching for improving the efficiency of load value prediction[J]. ACM Transactions on Architecture and Code Optimization, 2018, 15(4): 49.

[43] ALVES R, KAXIRAS S, BLACK-SCHAFFER D. Early address prediction: Efficient pipeline prefetch and reuse[J]. ACM Transactions on Architecture and Code Optimization, 2021, 18(3): 39.

[44] GONZÁLEZ J, GONZÁLEZ A. Speculative execution via address prediction and data prefetching[C]//Proceedings of the 11th International Conference on Supercomputing. New York: ACM, 1997: 196-203.

[45] ZHANG J S. The predictability of load address[J]. ACM SIGARCH Computer Architecture News, 2001, 29(4): 19-28.

[46] BEKERMAN M, JOURDAN S, RONEN R, et al. Correlated load-address predictors[J]. ACM SIGARCH Computer Architecture News, 1999, 27(2): 54-63.

[47] CHUNG B K, ZHANG J, PEIR J K, et al. Direct load: Dependence-linked dataflow resolution of load address and cache coordinate[C]//Proceedings of the 34th ACM/IEEE International Symposium on Microarchitecture (MICRO). New York: ACM, 2001: 76-87.

[48] BAER J L, CHEN T F. Effective hardware-based data prefetching for high-performance processors[J]. IEEE Transactions on Computers, 1995, 44(5): 609-623.

[49] HARRISON L. Examination of a memory access classification scheme for pointer-intensive and numeric programs [C]//Proceedings of the 10th International Conference on Supercomputing - ICS'96. New York: ACM 1996: 133-140.

[50] EICKEMEYER R J, VASSILIADIS S. A load-instruction unit for pipelined processors[J]. IBM Journal of Research and Development, 1993, 37(4): 547-564.

[51] BURTSCHER M, ZORN B G. Hybrid load-value predictors [J]. IEEE Transactions on Computers, 2002, 51(7): 759-774.

[52] RYCHLIK B, FAISTL J, KRUG B, et al. Efficient and Accurate Value Prediction Using Dynamic Classification[R]. Pittsburgh: Carneige Mellon University, 1998.

[53] SHIMOMURA Y, KOBAYASHI R. A stride value predictor suppressing conflicts focusing on predictability[J]. IEEE Transactions on Electronics, Information and Systems, 2011, 131(6): 1260-1270.

[54] LIPASTI M H. Value Locality and Speculative Execution [D]. Pittsburgh: Carnegie Mellon University, 1997.

[55] LOH G H. Width prediction for reducing value predictor size and power[C]//Proceedings of the First Value-Prediction Workshop. New York: ACM, 2003: 86-93.

[56] WANG K, FRANKLIN M. Highly accurate data value prediction using hybrid predictors[C]//Proceedings of the 30th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). Piscataway: IEEE, 1997: 281-290.

[57] CHANG S, TIAN F. An Efficient Implementation of Hybrid Data Value Predictors with Confidence Estimation[R]. Berkeley: University of California, 1998.

[58] BODINE J T. Exploiting Computational Locality in Global Value Histories[D]. Raleigh: North Carolina State University, 2002.

[59] ZHOU H Y, FLANAGAN J, CONTE T M. Detecting global stride locality in value streams[C]//Proceedings of the 30th annual international symposium on Computer architecture. Piscataway: IEEE, 2003: 324-335.

[60] GUNAL U. The Effectiveness of Global Difference Value Prediction And Memory Bus Priority Schemes for Speculative Prefetch[D]. Raleigh: North Carolina State University, 2003.

[61] NAKRA T, GUPTA R, SOFFA M L. Global context-based value prediction[C]//Proceedings of the Fifth International Symposium on High-Performance Computer Architecture. Piscataway: IEEE, 1999: 4-12.

[62] SEZNEC A. Tage-sc-l branch predictors again[EB/OL]. (2016-06-18)[2023-03-01]. https://jilp.org/cbp2016/paper/AndreSeznecLimited.pdf.

[63] SEZNEC A. Tage-sc-l branch predictors[EB/OL]. (2014-06-

15) [2023-03-01]. https://jilp. org/cbp2014/paper/Andre-Seznec.pdf.

[64] NADERAN-TAHAN M, SARBAZI-AZAD H. Adaptive prefetching using global history buffer in multicore processors[J]. The Journal of Supercomputing, 2014, 68(3): 1302-1320.

[65] XIAO Y, YANG Y P, ZHOU X M. Revised stride data value predictor design[C]//Eighth International Conference on High-Performance Computing in Asia-Pacific Region (HPCASIA'05). Piscataway: IEEE, 2005: 526-531.

[66] RILEY N, ZILLES C. Probabilistic counter updates for predictor hysteresis and stratification[C]//The Twelfth International Symposium on High-Performance Computer Architecture. Piscataway: IEEE, 2006: 110-120.

[67] DIEP T A, SHEN J P. VMW: A visualization-based microarchitecture workbench[J]. Computer, 1995, 28(12): 57-64.

[68] CMELIK B, KEPPEL D. Shade: A fast instruction-set simulator for execution profiling[C]//Proceedings of the 1994 ACM SIGMETRICS conference on Measurement and modeling of computer systems. New York: ACM, 1994: 128-137.

[69] BURGER D, AUSTIN T M. The SimpleScalar tool set, version 2.0[J]. ACM SIGARCH Computer Architecture News, 1997, 25(3): 13-25.

[70] CHEN R, QIN Z, YANG F. Data Value Prediction Methods and Performance[R]. Madison: University of Wisconsin, 1998.

[71] SEZNEC A, KALAITZIDIS K. Exploring value prediction limits[C]//CVP 2020-Championship Value Prediction. New York: ACM, 2020: 1-5.

[72] GUPTA A, MOR P, TANEJA H, et al. Steves: Pushing the limits of value predictors with sliding fcm and eves[EB/OL]. (2019-05-01)[2023-03-01]. https://www.microarch.org/cvp1/papers/Gupta_Mor_Taneja_Panda.pdf.

[73] BELL T C, CLEARY J G, WITTEN I H. Text Compression [M]. Englewood Cliffs: Prentice Hall, 1990.

[74] GOEMAN B, VANDIERENDONCK H, DE BOSSCHERE K. Differential FCM: Increasing value prediction accuracy by improving table usage efficiency[C]//Proceedings of the 7th International Symposium on High-Performance Computer Architecture. New York: ACM, 2001: 207-216.

[75] SAZEIDES Y, SMITH J E. Implementations of Context Based Value Predictors[R]. Madison: University of Wisconsin, 1997.

[76] BURTSCHER M. An improved index function for (D) FCM predictors[J]. ACM SIGARCH Computer Architecture News, 2002, 30(3): 19-24.

[77] DESHMUKH N, VERMA S, AGRAWAL P, et al. Dfcm++: Augmenting dfcm with early update and data dependence-driven value estimation[EB/OL]. (2018-06-03)[2023-03-01]. https://www.microarch.org/cvp1/papers/Deshmukh.pdf.

[78] KOIZUMI K, HIRAKI K, INABA M. H3VP: History based highly reliable hybrid value predictor[EB/OL]. (2018-06-03)[2023-03-01]. https://www. microarch. org/cvp1/papers/Koizumi.pdf.

[79] SAKHUJA C, SUBRAMANIAN A, JOSHI P, et al. Combining branch history and value history for improved value prediction[EB/OL]. (2019-11-01)[2023-03-01]. https://www.microarch.org/cvp1/papers/ChiragSakhuja.pdf.

[80] JOSHI P B. Techniques for Advancing Value Prediction[D]. Austin: The University of Texas, 2019.

[81] SUBRAMANIAN A. Advancing Value Prediction[D]. Austin: The University of Texas, 2019.

[82] ZHOU C B, HUANG L B, LI Z S, et al. Design space exploration of TAGE branch predictor with ultra-small RAM [C]//Proceedings of the on Great Lakes Symposium on VLSI 2017. New York: ACM, 2017: 281-286.

[83] MICHAUD P. An alternative TAGE-like conditional branch predictor[J]. ACM Transactions on Architecture and Code Optimization, 2018, 15(3): 30.

[84] SUGGS D, SUBRAMONY M, BOUVIER D. The AMD "zen 2" processor[J]. IEEE Micro, 2020, 40(2): 45-52.

[85] SEZNEC A. A 64-Kbytes ITTAGE indirect branch predictor [EB/OL]. (2011-06-04)[2023-03-01]. https://jilp.org/jwac-2/program/cbp3_07_seznec.pdf.

[86] PERAIS A. Exploiting Value Prediction With Quasi-Unlimited Resources[R]. Rennes: INRIA-IRISA Rennes Bretagne Atlantique, équipe ALF, 2012.

[87] PERAIS A, SEZNEC A. Revisiting Value Prediction[R]. Rennes: INRIA-IRISA Rennes Bretagne Atlantique, équipe ALF, 2012.

[88] SATO T, ARITA I. Table size reduction for data value predictors by exploiting narrow width values[C]//Proceedings of the 14th international conference on Supercomputing. New York: ACM, 2000: 196-205.

[89] PERAIS A. A Case for Speculative Strength Reduction[J]. IEEE Computer Architecture Letters, 2021, 20(1): 22-25.

[90] PERAIS A. Leveraging targeted value prediction to unlock new hardware strength reduction potential[C]//MICRO-54: Proceedings of the 54th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). New York:

ACM, 2021: 792-803.
[91] COOPER K D, SIMPSON L T, VICK C A. Operator strength reduction[J]. ACM Transactions on Programming Languages and Systems, 2001, 23(5): 603-625.
[92] YANG L, HUANG L B, YAN R, et al. Stride equality prediction for value speculation[J]. IEEE Computer Architecture Letters, 2022, 21(2): 57-60.
[93] PERAIS A, ENDO F A, SEZNEC A. Register sharing for equality prediction[C]//Proceedings of the 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). Piscataway: IEEE, 2016: 1-12.
[94] TULLSEN D M, SENG J S. Storageless value prediction using prior register values[C]//Proceedings of the 26th Annual International Symposium on Computer Architecture. New York: ACM, 1999: 270-279.
[95] BINKERT N, BECKMANN B, BLACK G, et al. The gem5 simulator[J]. ACM SIGARCH Computer Architecture News, 2011, 39(2): 1-7.
[96] Release CVP2v2.2 Infrastructure eric-rotenberg[EB/OL]. (2022-12-04)[2023-08-01]. https://github.com/eric-rotenberg/CVP/releases/tag/cvp2v2.2.
[97] ZHOU H, FU C, ROTENBERG E, et al. A Study of Value Speculative Execution and Misspeculation Recovery in Superscalar Microprocessors[R]. Raleigh: North Carolina State University, 2000.
[98] VINŢAN L N. Value prediction and speculation into the next microprocessors generation[J]//Proceedings of the Romanian Academy, 2004, 5(3): 1-8.
[99] REINMAN G, CALDER B. Predictive techniques for aggressive load speculation[C]//Proceedings of the 31st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). Piscataway: IEEE, 1998: 127-137.
[100] BALKAN D, KALAMATIANOS J, KAELI D. A study of errant pipeline flushes caused by value misspeculation[C]//Proceedings of the 16th Symposium on Computer Architecture and High Performance Computing. Piscataway: IEEE, 2004: 32-39.
[101] FU C Y, JENNINGS M D, LARIN S Y, et al. Value speculation scheduling for high performance processors[C]//Proceedings of the eighth international conference on Architectural support for programming languages and operating systems. New York: ACM, 1998: 262-271.
[102] FU C, JENNINGS M D, LARIN S Y, et al. Software-only Value Speculation Scheduling[R]. Raleigh: Department of Electrical and Computer Engineering, North Carolina State University, 1998.
[103] SAZEIDES Y. Modeling value speculation[C]//Proceedings Eighth International Symposium on High Performance Computer Architecture. New York: IEEE, 2002: 211-222.
[104] SALEHI M, Baniasadi A. Storage-aware value prediction [C]//2010 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools. Washington: IEEE, 2010: 722-728.
[105] BURTSCHER M, DIWAN A, HAUSWIRTH M. Static load classification for improving the value predictability of data-cache misses[J]. ACM SIGPLAN Notices, 2002, 37(5): 222-233.
[106] GELLERT A, PALERMO G, ZACCARIA V, et al. Energy-performance design space exploration in SMT architectures exploiting selective load value predictions[C]//2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010). Piscataway: IEEE, 2010: 271-274.
[107] ISLAM M M, STENSTROM P. Zero loads: Canceling load requests by tracking zero values[C]//Proceedings of the 9th workshop on MEmory performance: DEaling with Applications, Systems and Architecture. New York: ACM, 2008: 16-23.
[108] MORENO R, PINUEL L, DEL PINO S, et al. A power perspective of value speculation for superscalar microprocessors[C]//Proceedings of the 2000 International Conference on Computer Design. Piscataway: IEEE, 2000: 147-154.
[109] MORENO R, PINUEL L, DEL PINO S, et al. Power-efficient value speculation for high-performance microprocessors[C]//Proceedings of the 26th Euromicro Conference. EUROMICRO 2000, Informatics: Inventing the Future. Piscataway: IEEE, 2000: 292-299.
[110] CEBRIAN J M, ARAGON J L, GARCIA J M, et al. Adaptive VP decay: Making value predictors leakage-efficient designs for high performance processors[C]//Proceedings of the 4th International Conference on Computing Frontiers. New York: ACM, 2007: 113-122.
[111] CEBRIAN J M, ARAGON J L, GARCIA J M. Leakage energy reduction in value predictors through static decay [C]//2007 IEEE International Parallel and Distributed Processing Symposium. Piscataway: IEEE, 2007: 1-7.
[112] CEBRIÁN J M, ARAGÓN J L, GARCÍA J M, et al. Leakage-efficient design of value predictors through state and non-state preserving techniques[J]. The Journal of Supercomputing, 2011, 55(1): 28-50.
[113] BURTSCHER M, ZORN B G. Hybridizing and coalescing load value predictors[C]//Proceedings of the 2000 Inter-

national Conference on Computer Design. Piscataway: IEEE, 2000: 81-92.

[114] WANG H N, IBRAHIM M, MITTAL S, et al. Address-stride assisted approximate load value prediction in GPUs [C]//Proceedings of the ACM International Conference on Supercomputing. New York: ACM, 2019: 184-194.

[115] YAZDANBAKHSH A, PEKHIMENKO G, THWAITES B, et al. RFVP: Rollback-free value prediction with safe-to-approximate loads[J]. ACM Transactions on Architecture and Code Optimization, 2016, 12(4): 62.

[116] GHANDOUR W J, AKKARY H, MASRI W. Leveraging strength-based dynamic information flow analysis to enhance data value prediction[J]. ACM Transactions on Architecture and Code Optimization, 2012, 9(1): 1-33.

[117] NAITHANI A, FELIU J, ADILEH A, et al. Precise runahead execution[J]. IEEE Computer Architecture Letters, 2019, 18(1): 71-74.

[118] NAITHANI A, AINSWORTH S, JONES T M, et al. Vector runahead[C]//2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA). Piscataway: IEEE, 2021: 195-208.

[119] NAITHANI A, EECKHOUT L. Reliability-aware runahead [C]//2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA). Piscataway: IEEE, 2022: 772-785.

[120] PRUETT S, PATT Y. Branch runahead: An alternative to branch prediction for impossible to predict branches[C]// MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture. New York: ACM, 2021: 804-815.

[121] 杨智杰, 王蕾, 石伟, 等. 类脑处理器异步片上网络架构[J]. 计算机研究与发展, 2023, 60(1): 17-29.
YANG Z J, WANG L, SHI W, et al. Asynchronous network-on-chip architecture for neuromorphic processor [J]. Journal of Computer Research and Development, 2023, 60(1): 17-29. (in Chinese)

[122] GOPE D, LIPASTI M H. Bias-free branch predictor[C]// Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). Piscataway: IEEE, 2014: 521-532.

[123] JJIMÉNEZ D A. Fast path-based neural branch prediction [C]//Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). New York: ACM, 2003: 243-252.

[124] NAIN S, CHAUDHARY P. A neural network-based approach for the performance evaluation of branch prediction in instruction-level parallelism processors[J]. The Journal of Supercomputing, 2022, 78(4): 4960-4976.

[125] LURBE M, FELIU J, PETIT S, et al. DeepP: Deep learning multi-program prefetch configuration for the IBM POWER 8[J]. IEEE Transactions on Computers, 2022, 71(10): 2646-2658.

[126] BLACK M, FRANKLIN M. Neural Confidence Estimation for More Accurate Value Prediction[M]//Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2005: 376-385.

[127] MOHAPATRA S M, MISHRA P K. A novel approach for confidence estimation using support vector machines for more accurate value prediction[J]. International Journal of Computer Applications ETCC-2014, 2014(1): 51-58

[128] MISHRA P K, MOHAPATRA S M. More accurate value prediction using neural methods[J]. Elixir International Journal. 2014, 74: 26657-26663.

[129] BHARGAVA R, JOHN L K. Latency and energy aware value prediction for high-frequency processors[C]//Proceedings of the 16th international conference on Supercomputing. New York: ACM, 2002: 45-56.

[130] XIONG W J, SZEFER J. Survey of transient execution attacks and their mitigations[J]. ACM Computing Surveys, 2021, 54(3): 54.

[131] YAN M J, CHOI J, SKARLATOS D, et al. InvisiSpec: Making speculative execution invisible in the cache hierarchy[C]//2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). Piscataway: IEEE, 2018: 428-441.

[132] ZHAO Z N, JI H X, YAN M J, et al. Speculation invariance (InvarSpec): Faster safe execution through program analysis [C]//2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). Piscataway: IEEE, 2020: 1138-1152.

[133] SAKALIS C, ALIPOUR M, ROS A, et al. Ghost loads: What is the cost of invisible speculation?[C]//Proceedings of the 16th ACM International Conference on Computing Frontiers. New York: ACM, 2019: 153-163.

[134] SAKALIS C, KAXIRAS S, ROS A, et al. Efficient invisible speculative execution through selective delay and value prediction[C]//Proceedings of the 46th International Symposium on Computer Architecture. New York: ACM, 2019: 723-735.

[135] SAKALIS C, KAXIRAS S, ROS A, et al. Understanding selective delay as a method for efficient secure speculative execution[J]. IEEE Transactions on Computers, 2020, 69

(11): 1584-1595.
[136] BAR-EL H, CHOUKRI H, NACCACHE D, et al. The sorcerer's apprentice guide to fault attacks[J]. Proceedings of the IEEE, 2006, 94(2): 370-382.
[137] SHEIKH R, CAMMAROTA R, RUAN W J. Value prediction for security (VPsec): Countering fault attacks in modern microprocessors[C]//2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST). Piscataway: IEEE, 2018: 235-238.
[138] CAMMAROTA R, SHEIKH R. VPsec: Countering fault attacks in general purpose microprocessors with value prediction[C]//Proceedings of the 15th ACM International Conference on Computing Frontiers. New York: ACM, 2018: 191-199.
[139] SHEIKH R, CAMMAROTA R. Improving performance and mitigating fault attacks using value prediction[J]. Cryptography, 2018, 2(4): 27.
[140] ZHAO Q, LILJA D J. Static classification of value predictability using compiler hints[J]. IEEE Transactions on Computers, 2004, 53(8): 929-944.
[141] SAM N B, BURTSCHER M. Improving memory system performance with energy-efficient value speculation[J]. ACM SIGARCH Computer Architecture News, 2005, 33(4): 121-127.
[142] ZHOU H, CONTE T M. Enhancing memory level parallelism via recovery-free value prediction[C]//Proceedings of the 17th Annual International Conference on Supercomputing. New York: ACM, 2003: 326-335.

作者简介

黄立波 男,1983年出生于湖南省邵阳市. 现为国防科技大学计算机学院研究员. 主要研究方向为计算机体系结构.
E-mail: libohuang@nudt.edu.cn

杨 凌(通讯作者) 男,1996年出生于四川省雷波县. 现为国防科技大学计算机学院博士研究生. 主要研究方向为计算机体系结构.
E-mail: yanglingnudt@nudt.edu.cn

杨乾明 男,1984年出生. 现为国防科技大学计算机学院助理研究员. 主要研究方向为计算机微体系结构、芯片存储系统设计.
E-mail: yqm21249@nudt.edu.cn

马 胜 男,1986年出生于湖南省永州市. 现为国防科技大学计算机学院副研究员. 主要研究方向为计算机体系结构.
E-mail: masheng@nudt.edu.cn

王永文 男,1977年出生于山东省泰安市. 现为国防科技大学计算机学院研究员. 主要研究方向为微处理器体系结构.
E-mail: yongwen@nudt.edu.cn

隋兵才 男,1981年出生于山东省烟台市. 现为国防科技大学计算机学院副研究员. 主要研究方向为微处理器体系结构.
E-mail: bingcaisui@nudt.edu.cn

沈 立 男,1976年出生于四川省成都市. 现为国防科技大学计算机学院教授. 主要研究方向为多核/众核体系结构、运行时和编译优化、高性能计算.
E-mail: lishen@nudt.edu.cn

徐炜遐 男,1963年出生于湖南省常德市. 现为国防科技大学计算机学院研究员. 主要研究方向为高性能计算机系统结构.
E-mail: xuweixia@nudt.edu.cn