面向按序执行处理器的预执行指导的数据预取方法

党向磊,王箫音,佟 冬,陆俊林,程 旭,王克义
(北京大学微处理器研究开发中心,北京 100871;北京大学微处理器及系统教育部工程研究中心,北京 100871)

摘 要: 为提高按序执行处理器的访存性能,本文提出一种预执行指导的数据预取方法(PEDP).PEDP 利用跨距预取器对规则的访存模式进行预取,并在发生 L2 Cache 失效后通过预执行后续指令对不规则的访存模式进行精确的预取,从而结合两者的优势提高预取覆盖率.同时,PEDP 利用预执行过程中提前捕获的真实访存信息指导跨距预取器的预取过程.在预执行的指导下,跨距预取器可以对预执行能够产生的符合跨距访存模式的地址更早地发起预取请求,从而改善预取及时性.此外,为进一步优化上述指导过程,PEDP 使用更新过滤器有效去除指导过程中对跨距预取器的有害更新,从而提高预取准确率.实验结果表明,在平均情况下,PEDP 将基准处理器的性能提升 33.0%.与跨距预取和预执行各自单独使用相比,PEDP 将性能分别提高 16.2%和 7.3%.

关键词: 数据预取; 预执行; 访存延迟包容; 按序执行处理器
中图分类号: TP302.7 **文献标识码:** A **文章编号:** 0372-2112 (2012)11-2145-07
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2012.11.001

Pre-Execution Directed Prefetching for In-Order Processors

DANG Xiang-lei, WANG Xiao-yin, TONG Dong, LU Jun-lin, CHENG Xu, WANG Ke-yi
(*Microprocessor Research & Development Center, Peking University. Beijing* 100871, *China*;
Engineering Research Center of Microprocessor & System Ministry of Education, Peking University, Beijing 100871, *China*)

Abstract: This paper proposes a pre-execution directed prefetching(PEDP) method to improve the memory latency tolerance of in-order processors. PEDP utilizes stride prefetching to handle regular access patterns and pre-execution to generate accurate prefetches regardless of the regularity of access patterns when a L2 cache miss occurs, which combines the advantages of the two techniques to improve the prefetch coverage. Meanwhile, PEDP captures actual memory access patterns during pre-execution to guide the stride prefetcher's update process. Under the guide of pre-execution, the stride prefetcher can issue prefetches earlier than pre-execution for addresses that can be generated by both of the two techniques, thus improving the prefetch timeliness. In addition, PEDP achieves improvement in prefetch accuracy by an update filter which effectively eliminates the harmful updates to the stride prefetcher during the guide process. Experimental results demonstrate that PEDP increases the performance by 33.0% over the baseline processor. Compared with stride prefetching and pre-execution, PEDP improves the performance by 16.2% and 7.3%, respectively.

Key words: prefetching; pre-execution; memory latency tolerance; in-order processors

1 引言

按序执行处理器凭借其在能耗、面积和复杂度等方面的优势,获得了越来越广泛的应用[1~3].按序执行流水线在遇到缓存失效等事件时将被迫停顿,限制了处理器的单线程性能.随着处理器与存储器之间性能差距的日益扩大,访存延迟逐渐成为制约处理器单线程性能的重要瓶颈.

数据预取技术[4]在处理器访问数据之前预测其访存地址并提前发出访存请求,以隐藏访存延迟.传统的硬件数据预取技术[5~8]通过捕获程序运行过程中可重复的访存模式来产生预取地址.其中,预取规则的跨距(Stride)访存模式的跨距预取器[5,6]已被广泛应用于商用处理器中[9,10].预执行技术[3,11~15]在流水线因长延迟缓存失效而停顿时能够利用空闲的流水线资源预先执行失效访存指令的后续指令,以产生精确的数据预取.

与跨距预取相比,预执行的优势在于能够精确预取不规则的访存模式.但是,预执行只在处理器发生长延

收稿日期:2011-12-14;修回日期:2012-03-26
基金项目:"核高基"重大专项(No.2009ZX01029-001-002);中国博士后科学基金(No.20110490208)

迟缓存失效后才进行预取,且预取的提前时间不够长,可能无法完全隐藏访存延迟.与预执行相比,跨距预取能够在任意时刻对符合跨距访存模式的地址进行预取,且可以控制预取的提前时间以在处理器实际使用之前及时取回预取数据.可以看出,两种技术各有特点和优势,具有一定的互补性.

为提高按序执行处理器的单线程性能,本文提出一种预执行指导的数据预取方法(Pre-Execution Directed Prefetching,PEDP).PEDP 利用跨距预取器预取规则的访存模式,并在发生 L2 Cache 失效后通过预执行后续指令精确预取不规则的访存模式,以结合两者在捕捉访存模式方面的优势提高预取覆盖率.同时,PEDP 利用预执行期间的真实访存信息指导跨距预取器的预取过程,使其对预执行能够产生的符合跨距访存模式的地址更早地发起预取请求,以改善预取及时性.为优化上述指导过程,本文使用更新过滤器将预执行期间的真实访存信息划分为有用信息和有害信息,有用信息用于指导跨距预取器使其及早发出预取请求,有害信息则被过滤掉以避免破坏跨距预取器的预取状态,从而提高预取准确率.实验结果表明,在平均情况下,PEDP 将基准处理器的性能提升 33.0%,将跨距预取和预执行各自单独使用的性能分别提高 16.2%和 7.3%.

2 相关工作

已有的硬件预取技术可主要分为基于关联性的(Correlation-based)预取和跨距预取两类.基于关联性的预取[7,8]发掘并记录失效地址之间的关联性规律,并在这些规律重复发生时发出预取请求.该类方法需要大容量的存储结构来记录关联性历史信息,难以应用到实际的处理器中[16].

跨距预取[5,6]针对相邻访存地址之差为常值的跨距访存模式进行预取,已被广泛应用于商用处理器中[9,10].对于跨距访存模式,根据跨距是否为 1,可以分为单步(Unit)和多步(Non-unit)跨距访存模式[5].跨距预取器将失效地址按照一定的规则划分为多个流(Stream)[17].当每个流中实际发生的缓存失效符合跨距访存模式时,使用当前失效地址和跨距计算后续的访存地址并发起预取请求.预取距离(Prefetch-ahead Distance)指待预取的 Cache 行与处理器当前访问的 Cache 行之间相隔的跨距数目,它决定每个流初次满足预取条件时预取的 Cache 行数目[17].通过选择合适的预取距离,跨距预取器可在合适的提前时间及时发出预取请求,以完全隐藏访存延迟[4].

在流水线因长延迟缓存失效而停顿时,预执行后续指令来提高访存性能的技术可分为两类.第一类为非阻塞执行[14,15],继续执行和伪提交与失效访存指令数据无关的指令,将数据相关的指令移出流水线单独保存,等访存完成后再重新放入流水线执行和提交.该类技术需实现指令的乱序提交,具有较高的设计复杂度和硬件开销.第二类技术[3,11~13]以 Runahead[11,12]为典型代表,继续执行与失效访存指令数据无关的指令以产生数据预取或有效的计算结果,并将数据相关的指令移出流水线.所有指令均不提交.当失效指令完成访存后,处理器退出预执行并重新执行和提交所有预执行指令.该类技术不需改变原处理器的执行与提交机制,在设计复杂度和硬件开销方面更有优势.本文集中关注预执行的预取效果,选择第二类技术进行分析和评测.

跨距预取器具有较低的复杂度和实现代价,但无法预取不规则的访存模式[4],这点可以由预执行来弥补.预执行只在发生长延迟缓存失效后才进行预取,且预取的提前时间不够长,影响预取的覆盖率和及时性,这点可以由跨距预取来弥补.已有研究[9,17]对跨距预取和预执行同时使用的性能进行了评测,但并没有分析和评测两者的互补性,也没有根据预执行的特点调整跨距预取的策略.本文深入分析了两种方法的局限性和互补性,并基于此提出 PEDP,利用两者各自的优势弥补对方的不足.同时,本文使用更新过滤器将预执行期间的真实访存信息划分为有用信息和有害信息,只使用有用信息指导跨距预取,以改善两者结合使用效果.

3 预执行指导的数据预取方法

本文提出的预执行指导的数据预取方法(PEDP)利用跨距预取器捕获和预取规则的访存模式,并在发生 L2 Cache 失效(简称 L2 失效)后通过预执行后续指令精确预取不规则的访存模式.同时,PEDP 捕获预执行过程中的真实访存信息并用于指导跨距预取器的预取过程.为优化上述指导过程,本文采用更新过滤器去除对跨距预取器的有害更新.

3.1 工作原理

下面通过图 1 中的访存场景示例介绍 PEDP 的工作原理及相对跨距预取和预执行的性能优势.基准处理器为典型的单发射按序执行处理器,流水线在遇到缓存失效等事件时发生停顿,假设指令片段在基准处理器中的执行情况如图 1(a)所示.当处理器执行访存指令 A、B、C、D 和 E 时,分别于 Cache 行 L、L+1、S、L+2 和 L+3 处发生 L2 失效,流水线将停顿并等待失效处理完成.可看出,大部分程序执行时间被用于等待访存完成.

本文使用跨距预取器与文献[5]类似,根据物理地址高位将失效地址划分为多个流.每个表项监测一个流的访存规律,在初次捕获跨距访存模式时预取两个

后续的 Cache 行,之后,处理器每访问一个预取的 Cache 行,继续预取一个后续的 Cache 行.跨距预取的优化效果如图 1(*b*)所示,假设行 L 和 S 属于不同的流,则对行 S 的访问不影响监测行 L 的表项的状态.当访存指令 A 和 B 导致连续的 Cache 行 L 和 L+1 均发生失效时,跨距预取器发出对行 L+2 和 L+3 的预取请求.因此,处理器在执行访存指令 D 和 E 时将发生命中,并继续预取行 L+4 和 L+5.但跨距预取器无法对行 S 进行预取,因此,处理器执行访存指令 C 时仍会发生 L2 失效.

本文使用文献[3]的预执行方法.处理器在发生 L2 失效后进入预执行模式,预先执行后续指令以进行精确的数据预取.同时,预执行期间的有效计算结果被保存并用于加速程序的正常执行.预执行的优化效果如图 1(*c*)所示,在访存指令 A 进行主存访问的过程中,处理器预先执行访存指令 B、C 和 D(在图中用 b′、c′和 d′表示)以对行 L+1、S 和 L+2 进行预取.当处理器返回正常执行模式后,访存指令 B 和 C 将发生命中.通过复用预执行产生的有效计算结果,访存指令 C 和 D 之间的执行间隔变短.因此,在访存指令 D 执行时,对行 L+2 的预取请求尚未返回,导致其仍发生 L2 失效,但失效延迟被部分隐藏.此外,预执行未能覆盖到访存指令 E,使得访存指令 E 的正常执行仍旧会发生 L2 失效.

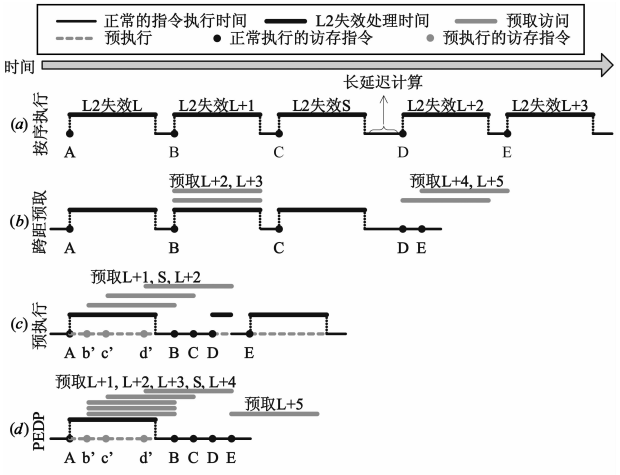


图1　基准处理器和三种预取方法的访存场景示例

本文提出的 PEDP 的优化效果如图 1(*d*)所示.与预执行类似,访存指令 B 和 C 的预执行(b′和 c′)触发对行 L+1 和 S 的预取.不同的是,访存指令 B 的预执行(b′)不仅触发对行 L+1 的预取,而且能够指导跨距预取器及早发起对行 L+2 和 L+3 的预取(连续的 Cache 行 L 和 L+1 均发生 L2 失效,满足跨距预取器的预取条件).这不仅能够提前取回访存指令 E 的数据(行 L+3),而且能够在预执行访存指令 D(d′)之前更早地发起对行 L+2 的预取(由延迟部分隐藏变为完全隐藏).因此,当处理器返回正常执行模式后,访存指令 B、C、D 和 E 均发生命中.可以看出,通过结合跨距预取和预执行两者的优势,PEDP 能够提高预取的覆盖率和及时性,进一步提高按序执行处理器的性能.

3.2　针对有害更新的过滤机制

在发生 Cache 失效时,若所访问的 Cache 行已由之前的 Cache 失效发起主存访问且尚未取回,则称该类 Cache 失效为 Secondary 失效,发起主存访问的 Cache 失效为 Primary 失效[18].在 PEDP 中,当处理器访问的 Cache 行已由跨距预取器或预执行的预取请求提前发起主存访问且尚未取回时,会产生 Secondary L2 失效.根据预取请求是由跨距预取器还是预执行发出的,可以分为跨距预取器引发的 Secondary L2 失效和预执行引发的 Secondary L2 失效.

上述两类 Secondary L2 失效在指导跨距预取器捕获访存模式方面具有不同的影响.若 Secondary L2 失效是由跨距预取器引发的,表明程序真实执行过程中的访存模式符合跨距预取器的预测,因此,使用该失效更新跨距预取器可以触发对后续 Cache 行的预取,增加有用预取的数目.若 Secondary L2 失效是由预执行引发的,表明之前已经有访问相同 Cache 行的访存指令发生 Primary L2 失效且更新过跨距预取器,因此,使用该失效对跨距预取器的更新是冗余的和有害的,可能对访存模式捕获过程造成负面影响,从而降低跨距预取器的预取准确率.

下面用图 2 中的示例说明有害更新对跨距预取的影响.访存指令①发生 L2 失效并导致处理器进入预执行模式.在预执行过程中,访存指令②至⑥均发生 L2 失效.当监测到访存指令①和②的失效 Cache 行 L 和 L+1 符合单步跨距访存模式时,跨距预取器发出对行 L+2 和 L+3 的预取.接下来,访存指令③(访问行 L)和访存指令⑤(访问行 L+1)均发生 Secondary L2 失效且两者均由预执行引发.访存指令④(访问行 L+2)和访存指令⑥(访问行 L+3)也均发生 Secondary L2 失效,但两者均由跨距预取器引发.

当访存指令③发生 Secondary L2 失效且更新跨距预取器时,将与已捕获的访存模式匹配失败,跨距预取

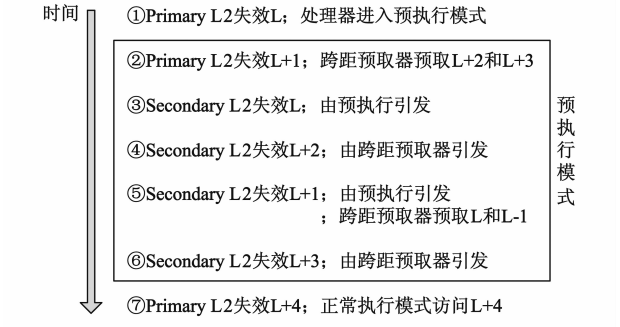


图2　有害更新的影响示例

器返回训练阶段重新捕捉跨距访存模式.访存指令④和⑤的执行使得连续的 Cache 行 L+2 和 L+1 均发生失效,此时,跨距预取器重新捕获单步跨距访存模式并发起对行 L 和 L-1 的预取(在此例中为无用预取).之后,访存指令⑥发生 Secondary L2 失效,与访存指令③类似,其对跨距预取器的更新也导致访存模式匹配失败.在处理器返回正常执行模式后,访存指令⑦将发生 L2 失效.

本文在 PEDP 中使用更新过滤器来识别并去除预执行引发的 Secondary L2 失效对跨距预取器的有害更新.对于上面的示例,更新过滤器能够避免使用 Secondary L2 失效 L 和 L+1(访存指令③和⑤)更新跨距预取器,则 Secondary L2 失效 L+2(访存指令④)对跨距预取器的更新将会访存模式匹配成功并触发对行 L+4 的预取,从而可以完全或部分隐藏访存指令⑦的访存延迟.同时,对行 L 和 L-1 的无用预取也能够被避免.可以看出,更新过滤器能够有效提高 PEDP 中跨距预取器的预取准确率.

4 PEDP 的设计与实现

4.1 总体设计

本文方法基于文献[5]的跨距预取器和文献[3]的预执行方法实现.实现 PEDP 的处理器可能运行于正常执行、预执行和合并结果三种模式[3].

在初始时刻,处理器处于正常执行模式,使用跨距预取器预取规则的访存模式.当检测到数据访问发生 L2 失效时,处理器使用检查点(Checkpoint)对寄存器进行备份,之后进入预执行模式.

在预执行模式,处理器继续执行与失效访存指令数据无关的后续指令以产生精确的数据预取和有效的计算结果.数据相关的指令被移出流水线并标记执行结果为无效.指令的预执行结果按程序顺序保存到指令复用队列(Instruction Reuse Queue, IRQ)[3]中,以在退出预执行后复用预执行得到的有效计算结果.为避免更改体系结构状态,Store 指令的写数据被保存在 Store Cache[3]中并前递给后续的 Load 指令使用.同时,预执行期间的 L2 失效信息被用来更新跨距预取器以指导其预取过程.如图 3 所示,PEDP 使用更新过滤器将预执行期间的 L2 失效信息划分成有用信息和有害信息.其中,有用信息被用于指导跨距预取器使其及早发出预取请求,有害信息则被过滤掉.当引发预执行的失效访存指令完成主存访问后,处理器清空流水线,恢复备份的寄存器,并转换到合并结果模式.

在合并结果模式,处理器从引发预执行的指令开始重新执行预执行指令或将预执行的有效计算结果合并到体系结构状态.若再次检测到数据访问发生 L2 失效,处理器对寄存器进行备份并重新进入预执行模式.当所有预执行结果均合并到体系结构状态后,处理器返回正常执行模式.

PEDP 中跨距预取器的内部结构如图 3 所示,可以同时预取地址递增和递减的访存模式[5].本文根据物理地址高位将失效地址划分为多个流[5].失效地址的 Tag 域用于区分不同的流,Index 域用于标识 Cache 行在流中的位置.跨距预取器可以同时监测多个流的访存规律,并将它们的信息保存在全相联或组相联结构的 Stream Table 中.Stream Table 的每个表项保存一个流的信息,包括标识位(Tag 域)、有效位(Valid 域)、跨距(Stride 域)、预取方向(Direction 域)、上一次发生失效的位置(Last Index 域)以及当前所处的状态(State 域).

在发生 L2 失效(或处理器初次访问跨距预取器取回的 Cache 行)时,PEDP 使用访存地址更新跨距预取器.Tag 域和 Valid 域用于判断当前的失效地址是否属于已处于监测范围的流,Stride 域和 Direction 域用于计算预取地址.对于每个流,当监测到连续的两个 L2 失效符合单步跨距访存模式(跨距为 1)或连续的三个 L2 失效符合多步跨距访存模式(跨距不为 1)时,进入预取状态.本文将预取距离设置为 2,即在初次预取时,预取两个后续的 Cache 行,之后,处理器每访问一个预取的 Cache 行,继续预取一个后续的 Cache 行.预取数据存放在 L2 Cache 中.

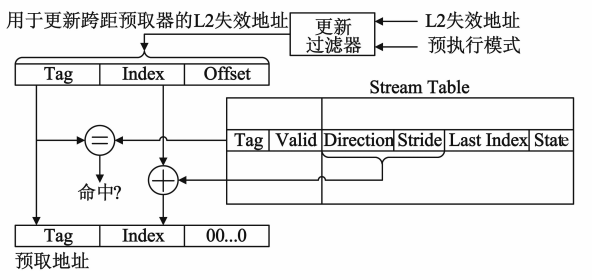


图3 带有更新过滤器的跨距预取器结构

4.2 更新过滤器的实现

PEDP 使用的更新过滤器可通过在 L2 Cache 的每个 MSHR(Miss Status Handling Register)[18]中增加 1 位的过滤位实现,工作流程如图 4 所示.当发生 L2 失效时,更新过滤器使用失效地址查找各个 MSHR.若无匹配的 MSHR,则该失效为 Primary L2 失效,处理器分配一个空闲的 MSHR 并初始化过滤位:若该失效是由跨距预取器的预取请求引发的,则初始化为 0;否则,初始化为 1.若存在匹配的 MSHR,则该失效为 Secondary L2 失效,更新过滤器将 MSHR 中的过滤位读出并进行检查:若过滤位为 1,则为预执行引发的 Secondary L2 失效,将其过滤掉,不允许更新跨距预取器;若过滤位为 0,则为跨距预取器引发的 Secondary L2 失效,允许更新跨距预取器以

使其及早发出预取请求,同时,将过滤位设置为 1,以避免后续在该行发生的 Secondary L2 失效再次更新跨距预取器.当 L2 失效完成主存访问后,其过滤位随相应的 MSHR 一起被释放掉.

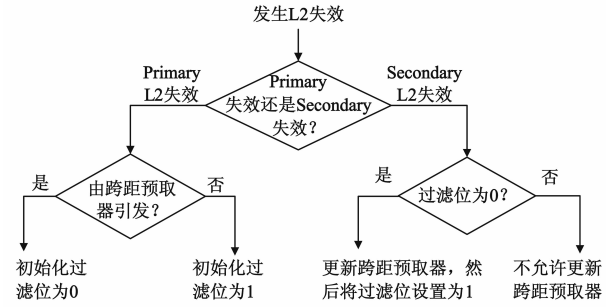


图4　更新过滤器的工作流程

5　实验评估

本文分别采用 SimpleScalar[19] 与 DRAMsim[20] 对处理器及主存系统进行建模,并集成 Wattch[21] 功耗模型,从 SPEC CPU2000 中选取 14 个访存密集型程序进行评测.每个程序运行 SimPoint[22] 选出的 100M 代表程序片段.本文选取 3.1 节介绍的跨距预取[5] 和预执行[3] 方法作为比较对象.基准处理器的配置基于 UniCore-2 处理器[23] 的典型结构,配置参数如表 1 所示.本文方法的配置参数如表 2 所示.

表 1　基准处理器配置参数

参数	配置值
流水线	8 级,单发射按序执行,1GHz
分离的 L1 ICache 和 L1 DCache	16KB,4 路组相联,每行 32 字节 2 周期访问延迟
统一的 L2 Cache	512KB,8 路组相联,每行 64 字节 12 周期访问延迟
主存	150 周期平均访问延迟

表 2　PEDP 配置参数

参数	配置值
预执行:Store Cache	1KB,2 路组相联
预执行:指令复用队列(IRQ)	256 表项
跨距预取:Stream Table	8 表项,全相联
更新过滤器	8 表项

5.1　预取效果和预取代价

对于基准处理器的 L2 失效,若其访存请求可由预取器或预执行提前发起,则其访存延迟可被完全或部分隐藏.这类 L2 失效称为预取覆盖到的 L2 失效,包括完全消除的 L2 失效和延迟部分隐藏的 L2 失效.本文将基准处理器的所有 L2 失效中被预取覆盖到的比例称为预取覆盖率.图 5 中的柱状数据从左到右依次给出了跨距预取、预执行和 PEDP 的预取覆盖率.在平均情况下,跨距预取、预执行和 PEDP 的预取覆盖率分别为 38.5%、44.3% 和 64.1%.实验结果表明,PEDP 通过结合跨距预取和预执行两者的优势,能够显著提高预取覆盖率.

图 5 还给出了延迟部分隐藏的 L2 失效的比例.可以看出,该比例在跨距预取中很小,而在预执行和 PEDP 中分别达到 30.8% 和 21.3%.通过设置合适的预取距离,跨距预取可以在合理的提前时间发出预取请求,从而能够在处理器实际访问之前将大部分预取数据取回.PEDP 利用跨距预取的上述特性来改善预执行的预取及时性,利用预执行期间的真实访存信息指导跨距预取器,使其对预执行能够产生的符合跨距访存模式的地址更早地发出预取请求,从而增加完全消除的 L2 失效的比例.

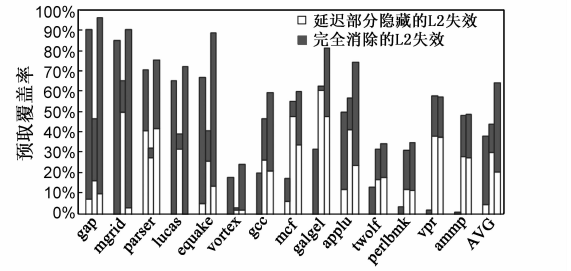


图5　三种方法的预取覆盖率

对于预取请求,根据其取回的数据是否被处理器实际使用,可以分为有用预取和无用预取.预取准确率是指有用预取在所有预取请求中所占的比例.无用预取会造成带宽和能耗浪费,是预取的主要代价之一.对于跨距预取,在预取地址预测错误时会产生无用预取.对于预执行,在预执行模式,当无法及时判定的转移指令发生误预测时,处理器将沿错误路径执行指令,也会产生无用预取.

如图 6 所示,在平均情况下,更新过滤器能够将 PEDP 中跨距预取器的预取准确率从 62.1% 提高到 74.0%,从而增加有用预取并减少无用预取,在改善性能优化效果的同时降低预取代价.

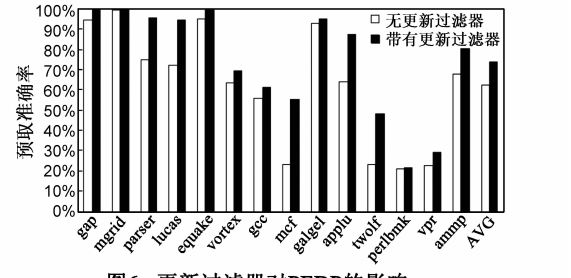


图6　更新过滤器对PEDP的影响

本文将实现预取方法的处理器相比于基准处理器,因无用预取而增加的访存请求的比例称为预取代价.如图 7 所示,在平均情况下,跨距预取、预执行和 PEDP 的预取代价分别为 7.6%、1.2% 和 6.4%.可以看

出,PEDP 在改善预取覆盖率和预取及时性的同时,并没有明显增加预取代价.其中,更新过滤器将 PEDP 的预取代价从 14.1%减少到 6.4%.

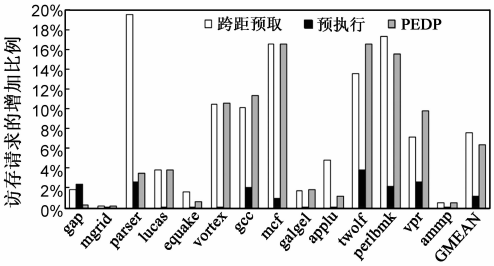


图7 三种方法的预取代价

5.2 性能和能耗

通过对预取覆盖率、及时性和准确率的优化,PEDP 能够减少 L2 失效次数或隐藏 L2 失效的延迟.同时,通过预执行,PEDP 还能够将程序所需数据预取到 L1 Cache 中,从而减少 L1 Cache 失效(简称 L1 失效)次数.为评估上述效果,本文统计了 PEDP 对基准处理器 L1 失效和 L2 失效的优化情况,如图 8 所示.在平均情况下,PEDP 能够将 L1 Cache 和 L2 Cache 的失效率分别降低 9.9%和 42.8%.此外,还有 21.3%的 L2 失效的延迟能够被部分隐藏.

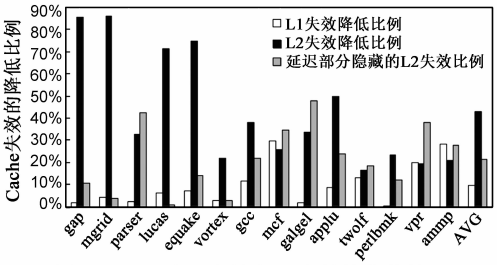


图8 PEDP对L1失效和L2失效的优化情况

PEDP 与跨距预取和预执行各自单独使用对基准处理器的性能优化效果如图 9 所示.在平均情况下,跨距预取、预执行和 PEDP 分别将基准处理器的性能提升 14.5%、24.0% 和 33.0%.其中,更新过滤器能够将 PEDP 的性能提升比例从 30.9%提高到 33.0%.实验结果表明,PEDP 通过降低 L1 Cache 和 L2 Cache 的失效率,能够有效提高基准处理器的性能.与跨距预取和预执行各自单独使用相比,PEDP 通过结合两者的优势能够提高预取的覆盖率和及时性,从而将性能分别提高 16.2%和 7.3%.

以基准处理器的能耗为基准值,将 PEDP 的能耗进行规格化,结果如图 10 所示.在平均情况下,PEDP 使基准处理器的能耗降低 1.6%.实验结果表明,PEDP 在提高处理器性能的同时并没有明显增加处理器的能耗,而且会降低部分程序的能耗.

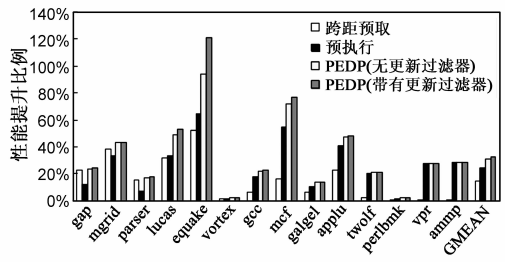


图9 三种方法的性能优化效果对比

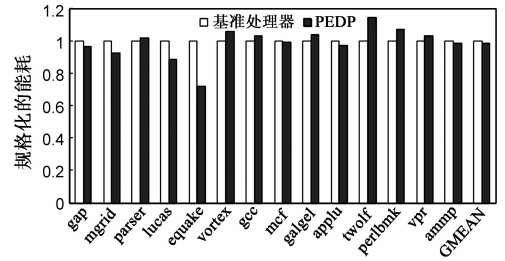


图10 基准处理器和PEDP的能耗对比

5.3 硬件开销

在基准处理器中实现 PEDP 所增加部件的硬件开销如表 3 所示.其中,Invalid Flag 用于在预执行模式标识寄存器的值是否有效[3].其它部件的功能参见 4.1 节的描述.从表 3 可以看出,PEDP 所增加的部件共占用存储容量约 2.5KB,不到 L1 Cache 总容量的十分之一,因此,本文方法的硬件开销较低.

表 3 PEDP 所增加部件的硬件开销

部件名称	存储类型	组织结构	存储容量
Checkpoint	寄存器	32-bit × 32	128B
Invalid Flag	寄存器	1-bit × 32	4B
Store Cache	SRAM	287-bit × 32	1148B
IRQ	FIFO	33-bit × 256	1056B
Stream Table	寄存器	40-bit × 8	40B
更新过滤器	寄存器	1-bit × 8	1B

6 结论

本文面向按序执行处理器提出一种预执行指导的数据预取方法(PEDP).PEDP 使用跨距预取器捕获和预取规则的访存模式,并在发生 L2 Cache 失效后通过预执行后续指令对不规则的访存模式进行预取,从而结合两种方法在捕捉访存模式方面的优势提高预取覆盖率.同时,PEDP 捕获预执行过程中的真实访存信息并用于指导跨距预取器的预取过程,使其对预执行能够产生的符合跨距访存模式的地址更早地发起预取请求,从而改善预取的及时性.此外,PEDP 使用更新过滤器对上述指导过程进行优化,有效去除预执行过程中对跨距预取器的有害更新,从而提高预取准确率.

实验结果表明,PEDP 能够以较低的预取代价和硬件开销有效提高预取的覆盖率、及时性和准确率,从而提高按序执行处理器的单线程性能.与跨距预取和预

执行各自单独使用相比,PEDP 能够获得更好的性能优化效果.

参考文献

[1] K Asanovic, et al. The landscape of parallel computing research: A view from Berkeley[R]. California, USA: Dept of EECS, University of California at Berkeley, 2006.

[2] P Kongetira, et al. Niagara: A 32-way multithreaded Sparc processor[J]. IEEE Micro, 2005, 25(2): 21 – 29.

[3] 王箫音,等.一种高能效的面向单发射按序处理器的预执行机制[J].电子学报,2011,39(2):458 – 463.
X Y Wang, et al. An energy-efficient executing ahead mechanism for improving the performance of single-issue in-order microprocessors[J]. Acta Electronica Sinic, 2011, 39(2): 458 – 463. (in Chinese)

[4] S P Vanderwiel, D J Lilja. Data prefetch mechanisms[J]. ACM Computing Surveys, 2000, 32(2): 174 – 199.

[5] S Palacharla, R E Kessler. Evaluating stream buffers as a secondary cache replacement[A]. Int'l Symposium on Computer Architecture[C]. Chicago, Illinois, USA: IEEE Computer Society, 1994. 24 – 33.

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

[7] D Joseph, D Grunwald. Prefetching using Markov predictors [A]. Int'l Symposium on Computer Architecture [C]. Denver, Colorado, USA: IEEE Computer Society, 1997. 252 – 263.

[8] S Somogyi, et al. Spatio-temporal memory streaming[A]. Int'l Symposium on Computer Architecture[C]. Austin, Texas, USA: IEEE Computer Society, 2009. 69 – 80.

[9] H W Cain, P Nagpurkar. Runahead execution vs. conventional data prefetching in the IBM POWER6 microprocessor[A]. Int'l Symposium on Performance Analysis of Systems and Software [C]. White Plains, New York, USA: IEEE Computer Society, 2010. 203 – 212.

[10] G Hinton, et al. The microarchitecture of the Pentium 4 processor[J]. Intel Technology Journal, 2001, 5(1): 1 – 13.

[11] J Dundas, T Mudge. Improving data cache performance by pre-executing instructions under a cache miss[A]. Int'l Conference on Supercomputing[C]. Vienna, Austria: IEEE Computer Society, 1997. 68 – 75.

[12] O Mutlu, et al. Runahead execution: An effective alternative to large instruction windows[J]. IEEE Micro, 2003, 23(6): 20 – 25.

[13] R D Barnes, et al. Tolerating cache-miss latency with multipass pipelines[J]. IEEE Micro, 2006, 26(1): 40 – 47.

[14] S Nekkalapu, et al. A simple latency tolerant processor[A]. Int'l Conference on Computer Design[C]. Lake Tahoe, California, USA: IEEE Computer Society, 2008. 384 – 389.

[15] A Hilton, et al. iCFP: Tolerating all-level cache misses in in-order processors[A]. Int'l Symposium on High Performance Computer Architecture[C]. Raleigh, North Carolina, USA: IEEE Computer Society, 2009. 431 – 442.

[16] T F Wenisch, et al. Making address-correlated prefetching practical[J]. IEEE Micro, 2010, 30(1): 50 – 59.

[17] S Iacobovici, et al. Effective stream-based and execution-based data prefetching[A]. Int'l Conference on Supercomputing [C]. Saint-Malo, France: IEEE Computer Society, 2004. 1 – 11.

[18] K I Farkas, N P Jouppi. Complexity/performance trade-offs with non-blocking loads[A]. Int'l Symposium on Computer Architecture[C]. Chicago, Illinois, USA: IEEE Computer Society, 1994. 211 – 222.

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

[20] D Wang, et al. DRAMsim: A memory system simulator[J]. ACM Computer Architecture News, 2005, 33(4): 100 – 107.

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

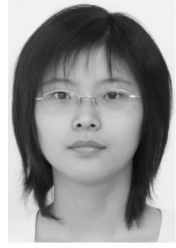
[22] E Perelman, et al. Picking statistically valid and early simulation points[A]. Int'l Conf on Parallel Architectures and Compilation Techniques[C]. New Orleans, Louisiana, USA: IEEE Computer Society, 2003. 244 – 255.

[23] X Cheng, et al. Research progress of UniCore CPUs and PKUnity SoCs[J]. Journal of Computer Science and Technology, 2010, 25(2): 200 – 213.

作者简介

党向磊　男,1988 年生于山东滕州,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结构设计、访存性能优化和系统芯片设计.

E-mail: dangxianglei@mprc.pku.edu.cn



王箫音(通讯作者)　女,1983 年生于河北保定,北京大学信息科学技术学院博士后.主要研究方向为微处理器结构设计、低功耗设计和存储系统性能优化.

E-mail: wangxiaoyin@mprc.pku.edu.cn