

结合访存失效队列状态的预取策略

郇丹丹^{1),2)} 李祖松^{1),2)} 胡伟武¹⁾ 刘志勇¹⁾

¹⁾(中国科学院计算技术研究所系统结构重点实验室 北京 100080)

²⁾(中国科学院研究生院 北京 100039)

摘 要 随着存储系统的访问速度与处理器的运算速度的差距越来越显著,访存性能已成为提高计算机系统性能的瓶颈.通过对指令 Cache 和数据 Cache 失效行为的分析,提出一种预取策略——结合访存失效队列状态的预取策略.该预取策略保持了指令和数据访问的次序,有利于预取流的提取.并将指令流和数据流的预取相分离,避免相互替换.在预取发起时机的选择上,不但考虑当前总线是否空闲,而且结合访存失效队列的状态,减小对处理器正常访存请求的影响.通过过滤机制提高预取准确性,降低预取对访存带宽的需求.结果表明,采用结合访存失效队列状态的预取策略,处理器的平均访存延时减少 30%,SPEC CPU2000 程序的 IPC 值平均提高 8.3%.

关键词 预取; Cache 失效; 龙芯 2 号

中图法分类号 TP302

Prefetching Policy Using Miss Queue Information

HUAN Dan-Dan^{1),2)} LI Zu-Song^{1),2)} HU Wei-Wu¹⁾ LIU Zhi-Yong¹⁾

¹⁾(Key Laboratory of Computer System and Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080)

²⁾(Graduate University of Chinese Academy of Sciences, Beijing 100039)

Abstract With the processor-memory performance gap continuing to grow, the performance of memory access becomes the major bottleneck of the performance improvement for modern micro-processors. Prefetching policy using miss queue information is proposed by investigating instruction cache misses and data cache misses. The prefetching policy increases the efficiency of stream allocation by maintaining the sequence of instruction cache misses and data cache misses. Prefetching is initiated when no future memory access requests are in miss queue and system bus is idle. Thus it reduces the negative effect on bandwidth. The prefetching policy has filter scheme which increases the accuracy of prefetching and decreases the waste of bandwidth. Experiment results indicate that on average 30% memory access latency is decreased. The performance is improved significantly. The average IPC speedup is 8.3%.

Keywords prefetch; Cache miss; Godson-2

1 引言

近年来,随着微处理器设计和生产工艺的快速

发展,存储系统的访问速度与处理器的运算速度的差距越来越显著,并且这个差距以每年 50% 的速度增长,使访存速度越来越成为提高处理器性能的瓶颈^[1]. 如何提高存储系统性能,弥补处理器与

收稿日期:2006-04-13;修改稿收到日期:2007-01-26. 本课题得到国家自然科学基金杰出青年基金(60325205)、国家自然科学基金(60673146)、国家“八六三”高技术研究发展计划项目基金(2005AA119020)和国家“九七三”重点基础研究发展规划项目基金(2005CB321600)资助. 郇丹丹,1979 年生,博士,主要研究方向为高性能计算机体系结构、操作系统和 VLSI 设计. E-mail: hdd@ict.ac.cn. 李祖松,1977 年生,博士,主要研究方向为高性能计算机体系结构、功能验证和 VLSI 设计. 胡伟武,1968 年生,博士,研究员,博士生导师,主要研究领域为高性能计算机体系结构、并行处理和 VLSI 设计等. 刘志勇,1946 年生,博士,研究员,博士生导师,主要研究领域为计算机算法、系统结构、并行处理和互联网络等.

存储系统性能的差距长期以来是体系结构领域的研究热点。

预取技术利用处理器访存行为的时间和空间局部性特征,在处理器发生 Cache 失效之前就预测需要的且不在 Cache 中的数据地址,提前发出内存访问请求,减少处理器 Cache 失效引起的流水线停顿时间,提高系统的整体性能。

预取技术的实施要尽量少地增加处理器设计复杂度。复杂的预取技术往往会使得处理器的设计变得相当复杂,而且这些预取对处理器性能的提高也不是非常明显,有时反而会恶化处理器的整体性能^[2]。因此,商用处理器所采用的预取技术通常是不增加处理器内核复杂度的类似于 Stream Buffer 的预取。Jouppi 首先提出 Stream Buffer 预取技术^[3],在访存失效模块一级对 Cache 失效地址的后续地址进行预取,预取回来的数据进入到 FIFO 结构的存储单元 Stream Buffer 中。Palacharla 等人对 Stream Buffer 技术进行了改进,提出带有过滤机制的 Stream Buffer^[4]。Iacobovici 等人提出基于 Stream Buffer 的多步长预取算法^[5],可以分析可变步长的访存模式。由于 Stream Buffer 技术硬件代价小,预取效果好,Cray T3E、IBM Power3^[6]、UltraSPARCI^[7]等处理器内部都实现了类似于 Stream Buffer 的预取技术。可见采用基于 Stream Buffer 的预取方法,在处理器中实现硬件预取是比较现实的。

预取技术的实现常常会加剧内存访问操作的数量^[8],造成系统有效带宽的降低,导致预取操作和处理器正常的内存访问操作之间竞争系统总线和内存的控制权。Stream Buffer 及其改进方法中预取的发起只考虑了处理器接口当前处于空闲状态,但处理器下一拍中可能有失效请求需要发出,由于处理器能够同时发出的访存请求数有限,预取会阻塞处理器的正常访存请求,增加处理器的流水线停顿时间。预取是一种猜测地从内存取数据的操作,预取应该尽可能少地影响处理器正常访存请求。否则,预取技术会恶化处理器的整体性能。为了减少预取造成的带宽浪费,降低预取对处理器正常访存请求的影响,需要对预取发起的时机进行研究,并进一步提高预取流提取的有效性和预取的准确性,从而提高处理器的性能。

由于目前的高性能处理器采用非阻塞 Cache,产生多个 Cache 失效请求,内存控制器同时处理多个内存访问,因此处理器核都设有访存失效队列(Miss Queue),也称为 MSHR(Miss Status Handling

Register)^[9]。访存失效队列是 Cache 与低层存储系统即内存之间的控制和通信枢纽,负责接收并向内存控制器转发 Cache 失效的访存请求,接收并向 Cache 转发内存的应答。访存失效队列能够提供未来内存访问请求的信息,因此可以结合访存失效队列进行预取技术的优化。

本文的贡献在于通过对 Cache 失效行为的分析,提出一种结合访存失效队列状态的预取策略。预取发起时机的选择,不但考虑当前总线是否空闲,而且结合访存失效队列的状态,减小预取对处理器正常访存请求的影响。预取策略保持了指令和数据访问的次序,有利于流的提取。将指令流和数据流的预取相分离,避免相互替换,并通过流过滤机制提高预取的准确性,降低预取操作对系统访存带宽的负面影响,有效地提高处理器系统的性能。实验结果表明,采用结合访存失效队列状态的预取策略,处理器的平均访存延时减少 30%,SPEC CPU2000 程序的 IPC 值平均提高 8.3%。

本文第 2 节对龙芯 2 号处理器运行 SPEC CPU2000 测试程序时的 Cache 失效行为进行分析;第 3 节中基于 Cache 失效行为的分析结果,提出一种结合访存失效队列状态的预取策略;第 4 节介绍实验平台,并进行性能评测与分析;最后总结全文工作。

2 Cache 失效行为分析

预取技术实质是一种内存访问序列猜测技术,预取策略要结合应用程序的访存行为^[10]。预取的有效性基于程序访问的空间局部性,通过分析程序访问的历史信息来预测未来的访存序列。本节对龙芯 2 号处理器^[11]运行 SPEC CPU2000 程序 Cache 失效的空间局部性特征进行分析,通过分析结果指导预取策略设计。

龙芯 2 号处理器的指令 Cache 和数据 Cache 组织为 4 路组相联容量 64KB 形式,指令 Cache 存放程序的目标代码,数据 Cache 存放程序运行所需要的数据。通常应用程序目标代码的存储空间远小于程序运行所需数据的存储空间,因此大多数应用程序的数据 Cache 失效率远大于指令 Cache 失效率。图 1 给出了指令 Cache 失效和数据 Cache 失效在总 Cache 失效中所占的比例,其中前 15 个程序为浮点程序,后 10 个程序为定点程序。从图中可以看出,浮点程序的数据 Cache 失效率远大于指令 Cache 失效率。定点程序除 eon, perlbnk, vortex 和 twolf 外,

数据 Cache 失效率也都大于指令 Cache 失效率. 总 Cache 失效中平均 96.9% 是数据 Cache 失效. 此外, 指令访问空间局部性好, 取指除遇到分支指令外通常是顺序的, 而数据访问则与应用程序特征密切相关, 通常应用程序的数据访问是不太规则的. 指令 Cache 失效率高. 指令 Cache 命中过滤了相当大部分的取指操作, 指令访问的空间局部性好于数据访问的空间局部性, 这两个因素决定了处理器指令 Cache 失效的空间局部性和数据 Cache 失效的空间局部性有一定的差别.

图 2 为指令 Cache 失效的空间局部性分布, 横坐标表示 SPEC CPU2000 测试程序, 纵坐标表示在一定的时间窗口中 (设置为 100 个时钟周期), 统计

指令 Cache 失效地址的间隔 (图例中的数字表示 Cache 失效地址间隔是多少个 Cache 行), 计算具有各种间隔的指令 Cache 失效数占总指令 Cache 失效数的比例. 从图中可以看出, SPEC CPU2000 测试程序, 指令 Cache 失效中有 40% 以上的 Cache 行地址是相邻的, 说明处理器运行 SPEC CPU2000 测试程序时指令 Cache 失效具有明显的空间局部性.

图 3 为数据 Cache 失效的空间局部性分布, 横坐标表示 SPEC CPU2000 测试程序, 纵坐标为各种间隔的数据 Cache 失效数占总数据 Cache 失效数的比例. 从图中可以看出, SPEC CPU2000 不同测试程序数据 Cache 失效的空间局部性体现程度具有明显差异. 例如, wupwise, swim, mgrid, mesa 等 4 个

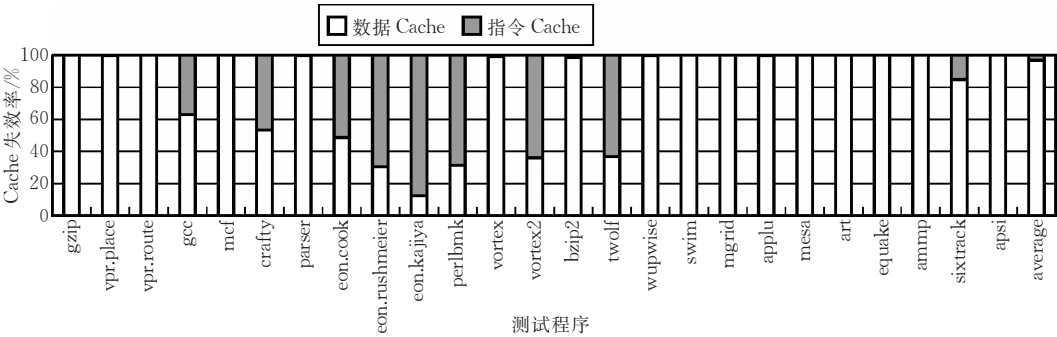


图 1 数据和指令失效的比例

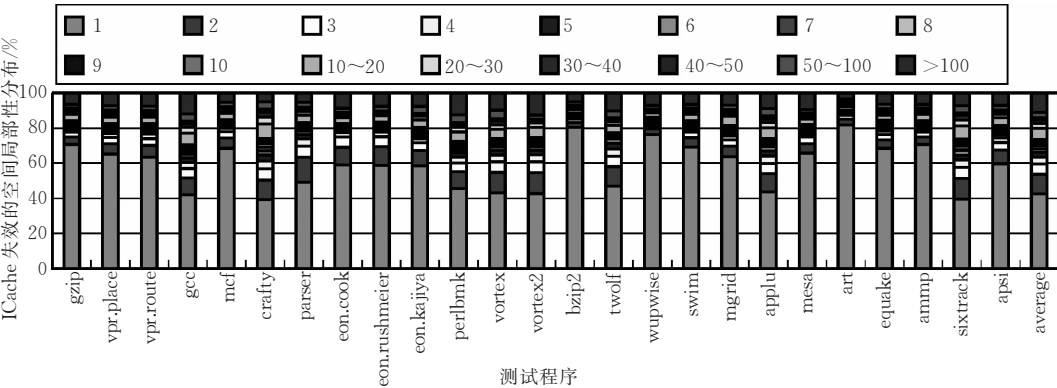


图 2 指令 Cache 失效的空间局部性

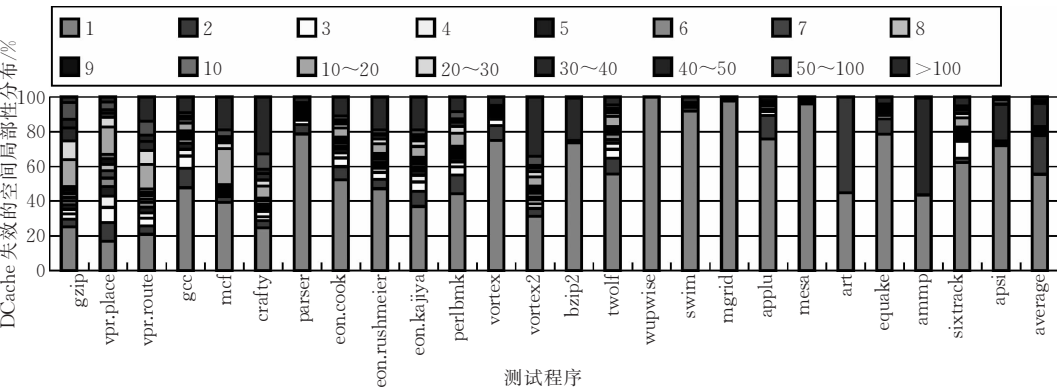


图 3 数据 Cache 失效的空间局部性

测试程序,有约 90%数据 Cache 失效的 Cache 行地址与后续数据 Cache 失效的 Cache 行地址是相邻的.而对于 gzip,vpr,crafty,vortex 等 4 个测试程序,连续数据 Cache 失效的 Cache 行地址在不同的地址间隔上均有一定分布.浮点程序 Cache 失效的空间局部性比定点程序 Cache 失效的空间局部性好.从整体上看,处理器运行 SPEC CPU2000 测试程序时

的数据 Cache 失效同样具有明显的空间局部性.

图 4 给出了每执行一千条指令发生的 Cache 失效次数.从图中可以看出,处理器每执行一千条指令平均发生 Cache 失效 15.3 次.每执行千条指令发生的 Cache 失效次数多少反映了降低访存延时在各个程序的性能优化中可发挥的作用,只有 Cache 失效次数多的程序降低访存延时的技术才能够更好地发挥作用.

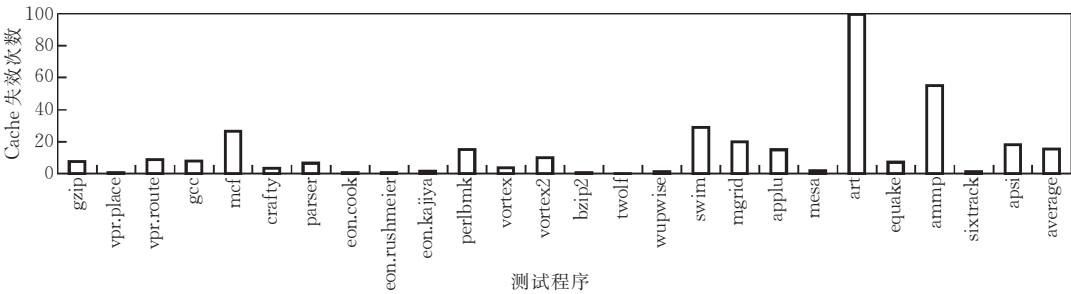


图 4 每执行千条指令发生 Cache 失效次数

图 1、图 3 和图 4 表明每执行一千条指令发生的 Cache 失效次数多的程序数据 Cache 的失效所占的比例很大,而且数据 Cache 失效空间局部性又都非常好,因此通过预取技术降低访存延时在处理器的性能优化中能够取得很好的效果.

3 结合访存失效队列状态的预取策略

3.1 结合访存失效队列状态的预取策略设计

结合访存失效队列状态的预取策略,预取流的提取和预取发起时机的控制都结合访存失效队列的状态.指令 Cache 或数据 Cache 失效,进入访存失效队列时访问预取缓冲区;如果预取缓冲区命中,则将数据取回访存失效队列,否则访问流过滤控制装置;流过滤控制装置定位步长为+1 和-1 的预取流,并在预取缓冲区中分配预取流;指令和数据的预取流

缓冲区和预取过滤缓冲区分离,采用 LRU 替换算法;访存失效队列中没有处于进入访存失效队列和等待发出访存请求状态的项时,预取发起控制装置发出预取请求.

图 5 为结合访存失效队列状态的预取策略的预取电路逻辑结构图,预取电路主要由流缓冲控制电路(Stream Buffer)、流过滤控制电路(包括流猜测 Stream Prediction Control、流过滤 Stream Filer 和流分配 Stream Allocation Control)和流预取发起控制电路(Stream Prefetch Control)组成.流过滤控制电路负责完成预取流的定位;流缓冲控制电路负责存放预取回来的数据;流预取发起控制电路负责向内存发起预取请求.流过滤电路决定进行预取的流,流预取发起电路决定何时发起预取,流缓冲控制电路决定保存哪些预取回来的流,因此这三部分共同构成了预取策略的控制核心.

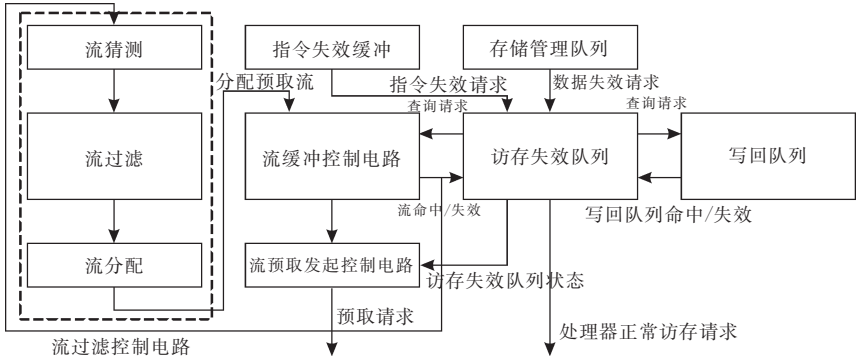


图 5 结合访存失效队列状态的预取策略的预取电路逻辑结构

图 6 给出了采用结合访存失效队列状态的预取策略,Cache 失效的处理流程图.指令失效缓冲区或

存储管理队列向访存失效队列发出 Cache 失效访问请求,进入访存失效队列.判断访存失效队列中

进入访存失效队列状态的项对应 Cache 块是否在写回队列命中,同时访问 Stream Buffer 判断是否在 Stream Buffer 命中.如果在写回队列或 Stream Buffer 命中,将数据取回访存失效队列.否则,访问流过滤控制装置,并访问内存,等待低层存储系统数据返回,填充 Cache.

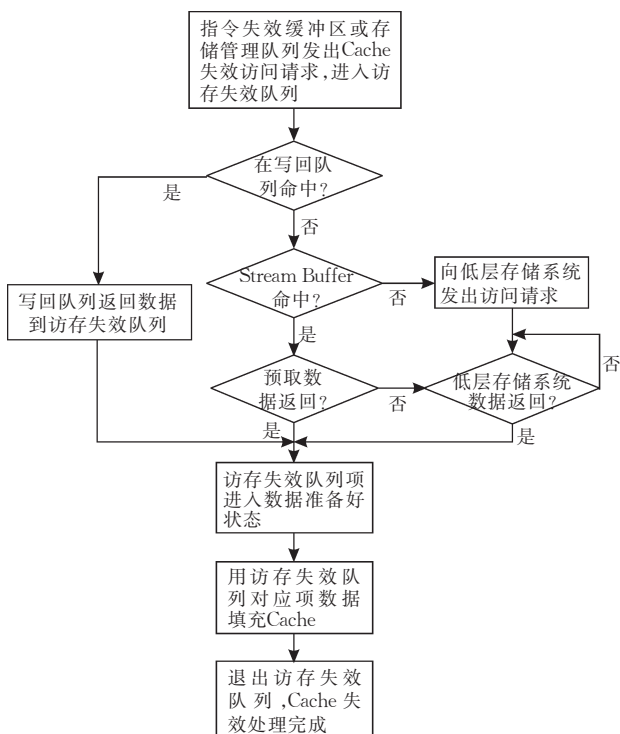


图 6 结合访存失效队列状态的预取策略
Cache 失效的处理流程

Stream Buffer 保存指令流或数据流的信息和预取回来的数据.其中包括 8 个指令流缓冲区和 8 个数据流缓冲区,每个流缓冲区深度为 2,即由两项组成.流缓冲区包括如下域:Valid(有效位域,标识该流缓冲区有效)、Stride(步长域,表示该预取流的步长)、Age(访问时间域,表示距离最后一次访问的时间)、Addr(地址域,表示该项所对应的预取 Cache 行地址)、Fetching(预取发出域,表示该项已经发出预取)、Data(数据域,表示预取回来的数据)、Mement(返回双字数域,表示预取数据返回了几个双字).其中 Valid 域、Stride 域和 Age 域每个流缓冲区的两项共用,其他域流缓冲区的每一项都有一项。

当预取电路接收到来自访存失效队列的查询请求以后,进行 Stream Buffer 的地址相联查找.如果在 Stream Buffer 中命中且预取数据已经返回,则将相应的数据返回给访存失效队列.如果在 Stream Buffer 中命中但数据没有返回,则等待数据的返回,数据返回后直接传递给访存失效队列.如果在

Stream Buffer 中没有命中,则通过流过滤机制来进行预取流的定位.当一个流被定位后,如果有空的流缓冲区则直接放入新的流缓冲区,否则流缓冲控制电路基于 LRU 替换算法选择一个 Stream Buffer 进行替换.

从图 2、图 3 中的统计数据可见,几乎所有的 SPEC CPU2000 测试程序都具有明显的 Cache 失效地址空间局部性特征.这种空间局部性主要体现在:在一定时间范围内,访存操作 Cache 行地址间隔为 1 的访存操作在该程序整体访存操作中占有很大的比重,而访存操作 Cache 行地址间隔大于 1 以后,除 art 程序外局部性特征不明显.此外,在访存地址间隔为 1 个 Cache 行地址的访存 Stream 中,有些 Stream 的特征是后续访存 Cache 行地址是当前访存 Cache 行地址+1,而有些 Stream 的特征是后续访存 Cache 行地址是当前访存 Cache 行地址-1,因此在 Stream 过滤机制中应以当前访存地址递增和递减两个方向按照步长为+1 和-1 进行定位.

Jouppi 所提出的 Stream Buffer 预取^[3]的思想是:总是取当前访存地址的下一个 Cache 行地址的内容.然而,在实际的访存序列中,访存地址空间局部性只是在一定程度上体现,并且大多数应用程序的空间局部性也是比较有限的.因此,Stream Buffer 预取策略往往会进行不必要的预取,尤其是对于访存带宽要求比较高而空间局部性不好的应用程序来说,这种预取策略往往会导致系统有效访存带宽的降低,有时反而会恶化系统的访存性能.为了克服 Stream Buffer 预取策略中因为总是预取下一行而降低系统有效访存带宽的缺点,Palacharla 等人基于 Stream Buffer 的预取思想,提出了一种基于访存历史轨迹来定位 Stream 的过滤机制^[4],这种过滤机制不仅提高了预取的准确性,而且减少了预取对系统有效访存带宽的负面影响.当访问 Stream Buffer 不命中时,流过滤控制电路中的流猜测逻辑就对下一个失效地址进行猜测,猜测方法是对当前失效地址递增或递减,并将猜测所得的地址放入到 Stream Filter 中.如果后面的 Cache 失效地址与 Stream 过滤缓存中的某一项相同时,一个 Stream 就被定位,并分配到 Stream Buffer 中.为进一步提高预取的准确性,本文的流过滤控制电路在流提取时只有两次预测相同步长,才确定为一个预取流,进入 Stream Buffer.

基于以上分析,在 Stream 过滤控制电路中维护一个 8 项的指令 Stream 过滤表和一个 8 项的数据

Stream 过滤表,用来存放指令流和数据流定位需要的信息,Stream 过滤表(Stream Filter)结构如表 1 所示.其中每一项包括如下域:有效位域(Valid)、步长域(Stride)、定位该步长的次数域(Times)、该项存在的时间即年龄域(Age)、当前访存操作对应 Cache 行地址域(Addr),Cache 行地址加 1 域(Addr_add_one),Cache 行地址减 1 域(Addr_sub_one).在 Stream 过滤表中,Valid 位为 1 的每一项都对应着一个可能即将被定位的步长为+1 或-1 的 Stream 序列.

表 1 Stream 过滤表

有效位	步长	次数	年龄	Cache 行地址	Cache 行地址加 1	Cache 行地址减 1
1	0	0	0	A	A+1	A-1
0	0	0	0	B	B+1	B-1
1	1	1	1	C	C+1	C-1
1	-1	1	10	D	D+1	D-1
.....						

图 7 给出了采用结合访存失效队列状态的预取策略,流过滤控制电路的工作过程,具体包括如下步骤:

1. 访存失效队列的失效请求查询 Stream Buffer 没有命中,查询 Stream Filter. 如果 Stream Filter 没有命中,执行步 2;如果 Stream Filter 命中,执行步 3.
2. 如果 Stream Filter 有空项,随机选择一个空项进入. 否则,Stream 过滤控制电路基于 LRU 替换算法在 Stream 过滤表中为其分配一个新的项,并且在这一项中记录与当前访存操作地址间隔为+1、-1 的 Cache 行地址.
3. 地址与命中项中的任意一个地址值相同时,如果步长与流过滤表中该项记录的步长相同,执行步 5,否则执行步 4.
4. 将步长置为新的步长,流命中次数置为 0,执行步 6.
5. 如果流命中次数大于 1,一个 Stream 被定位,此时 Stream 过滤电路向流缓冲控制电路发出 Stream 已经被定位

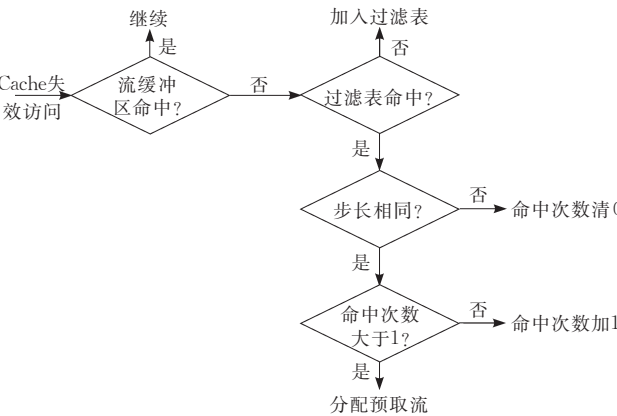


图 7 结合访存失效队列状态的预取策略的流过滤控制电路工作过程

信号,新定位的 Stream 进入一个独立的 Stream Buffer 来进行相应的预取.如果流命中次数小于等于 1,则流命中次数加 1.

6. 流过滤控制电路处理完成.

3.2 结合访存失效队列状态的预取策略的优点

与现有的预取技术相比,结合访存失效队列状态的预取策略的优点在于:

(1) 指令 Cache 或数据 Cache 失效进入访存失效队列时访问预取装置,与在访存前访问预取装置相比,保持了指令和数据访问的次序,有利于流的提取.

指令 Cache 或数据 Cache 访问是有序的,因此指令 Cache 或数据 Cache 失效请求发送到访存失效队列时是有序的.目前基于 Stream Buffer 的预取技术都是从访存失效队列中随机选择一个等待访问内存的访存失效队列项,访问 Stream Buffer.这样访问 Stream Buffer 的地址就已经是无序的,不利于预取流的步长确定和预取流的提取.因此,在指令 Cache 或数据 Cache 失效处于进入访存失效队列状态时访问预取装置,与在访问内存前访问预取装置相比,保持了指令和数据访问的次序,更有利于流的提取.

(2) 预取发起时机的选择,不但考虑当前总线是否空闲,而且结合访存失效队列的状态,在访存失效队列中没有处于当前需要发出和可能发出访存请求的项时,Stream Buffer 发出预取,减小预取对处理器正常访问请求的影响.

预取发起时机的选择是预取技术的一个重要组成部分,预取应尽量减少对处理器正常访存请求的影响.Stream Buffer 及其改进方法中预取的发起只考虑了处理器接口当前处于空闲状态,没有考虑处理器在最近是否会有访存请求发出,由于处理器能够同时发出的访存请求数有限,如果处理器在下一拍需要访问内存则预取会阻塞处理器正常的访问请求.因此不但需要考虑当前总线是否空闲,而且应该结合访存失效队列的状态,在处理器当前和下一拍可能有失效请求发出的情况下也不发起预取操作.当访存失效队列中有处于等待发出访存请求状态的项时,为不影响处理器的正常访存请求,不应该发起预取请求.当访存失效队列中有处于进入访存失效队列状态的项时,表示该项下一拍要查询写回队列和 Stream Buffer,如果不命中则需要发起访存操作,为不影响处理器的正常访存请求,也不应该发起预取请求.所以,在访存失效队列中没有处于进入访

存失效队列和等待发出访存请求状态的项时,发出预取请求,可以减小对处理器正常访存请求的影响.

(3) Stream Filter 两次预测相同步长,且步长为+1、-1时,定位为一个预取流进入 Stream Buffer,提高预取的准确性,降低预取对带宽的需求.

预取技术发挥作用需要带宽的保证.一些 Cache 失效频繁的访存密集型程序很适合做预取,但其本身对处理器的带宽需求就很大,这就要求预取的准确性,尽量避免不必要的预取对带宽的浪费.通过第2节程序 Cache 失效的局部性特征分析,大部分 Cache 失效的步长绝对值都为1,步长大于1以后,除 art 程序外局部性特征不明显.因此,流过滤机制在流提取时只有两次预测相同步长(Stride),并且步长为+1, -1,才确定为一个预取流,进入 Stream Buffer,提高预取准确性,降低对带宽的需求.

(4) 指令和数据 Stream Buffer 及 Stream Filter 相分离,避免相互替换.

根据第2节 Cache 失效的空间局部性分析,说明处理器运行 SPEC CPU2000 测试程序时指令 Cache 失效和数据 Cache 失效分别具有很好的空间局部性.但目前基于 Stream Buffer 的预取技术都是根据处理器访问内存的地址序列进行流的分析,并没有区分指令 Cache 和数据 Cache 的失效.定位预取流时,将指令和数据的流过滤控制装置相分离,可以避免指令和数据的相互替换,有利于流的提取.指令失效的空间局部性非常好,因此指令流的预取准确度都很高.程序发生函数调用并完成后会重新回到调用函数继续执行,所以指令的预取可能会经过一定的时间才会被使用.由于数据 Cache 的失效率一般远大于指令 Cache 的失效率,预取缓冲区在发生溢出时按照普遍采用的先进先出策略(FIFO)或最近最少使用策略(LRU)等进行流缓冲区的替换,都会导致指令 Cache 失效所占用的流缓冲区被数据 Cache 失效所提取的预取流替换出去,浪费指令流的预取.预取缓冲区分配时,也将指令和数据相分离,各占一半的项数,避免数据流缓冲区与指令流缓冲区的相互替换.

4 性能评测与分析

4.1 定义

衡量各种预取技术的预取效果主要有以下4个指标:流提取有效性(Efficiency)、预取及时性(Timeliness)、预取准确性(Accuracy)和预取覆盖

率(Coverage).在下面的定义中,*stream_hit* 表示预取提取流的数量;*stream_hit_ready* 表示预取提供的数据在需要时已经取回来的数量;*stream_pre-fetch* 表示总的预取 Cache 块的数量;*total_cache_miss* 表示总的 Cache 访问失效数.

流提取有效性(Efficiency)定义为预取提取的流覆盖必要访存数据的比例,计算公式:

$$Efficiency = stream_hit / total_cache_miss \quad (1)$$

预取及时性(Timeliness)定义为预取提供的数据在需要时已经取回的比例,计算公式:

$$Timeliness = stream_hit_ready / stream_hit \quad (2)$$

预取覆盖率(Coverage)定义为预取的数据覆盖必要访存数据的比例,计算公式:

$$Coverage = stream_hit_ready / total_cache_miss \quad (3)$$

预取准确性(Accuracy)定义为预取数据中有效数据的比例,计算公式:

$$Accuracy = stream_hit_ready / stream_prefetch \quad (4)$$

理想的预取是具有大的预取覆盖率,较高的预取准确性以及能够及时为处理器提供必要的数据.此外,处理器的 IPC 值仍然是衡量预取技术对于处理器整体性能优化作用的重要标准.

4.2 实验平台

龙芯2号^[11]是实现64位指令集的RISC处理器,采用四发射超标量流水结构.一级指令和数据Cache为非阻塞Cache、容量64KB、四路组相联结构,访存失效队列8项.写回队列4项,用来缓解对低层存储系统的读写冲突,也作为Victim Cache解决Cache的冲突失效.访存失效队列与低层存储系统的接口采用SysAD总线协议,并支持Split读.访存延时为第一个子块返回50拍,子块间隔3拍.本文采用龙芯2号处理器的C模拟器作为实验平台.测试程序采用国际通用处理器生产厂家公认的性能测试标准程序SPEC CPU2000^[12].

4.3 实验结果与分析

图8为采用结合访存失效队列状态的预取策略,运行SPEC CPU2000测试程序进行数据统计,并根据定义的公式(1)~(4)计算的流提取有效性、预取及时性、预取覆盖率和预取准确性.结果表明,结合访存失效队列状态的预取策略平均流提取有效性为40%,平均预取及时性为45%,平均预取准确

性为 58%, 平均预取覆盖率为 20%。程序的预取流提取有效性和预取及时性越好, 预取的覆盖率就越大。预取的准确性越高, 带宽浪费就越少, 对处理器的正常访存请求的影响就越小。

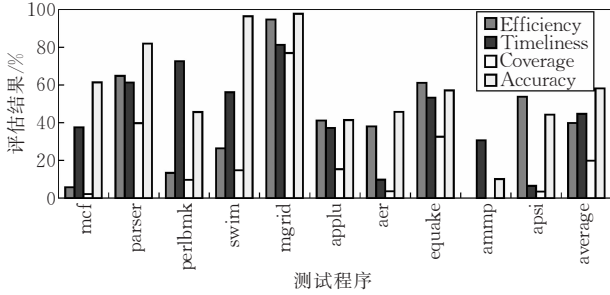


图 8 流提取有效性、预取及时性、预取覆盖率、预取准确性

图 9 给出了采用结合访存失效队列状态的预取策略 IPC 值提高比例。从图中可以看出, 采用结合访存失效队列状态的预取策略使得处理器的 IPC 值得到明显地提高, SPEC CPU2000 程序 IPC 值平均提高 8.3%。特别是对于访存空间局部性好的测试程序, 预取技术对处理器 IPC 值的提高程度非常显著, 其中 mgrid 程序的 IPC 值提高达到 43.1%。在图 8 中, mgrid 程序的流提取有效性、预取及时性、预取覆盖率和预取准确性也都是最高的。因此, 用流提取有效性、预取及时性、预取覆盖率和预取准确性共同构成衡量预取效果的量化指标是适合的。与只使用 IPC 值来评价预取技术相比较, 增加流提取有效性、预取及时性、预取覆盖率和预取准确性作为衡量标准更有针对性, 有利于分析预取技术存在的不足, 以更好地加以改进。

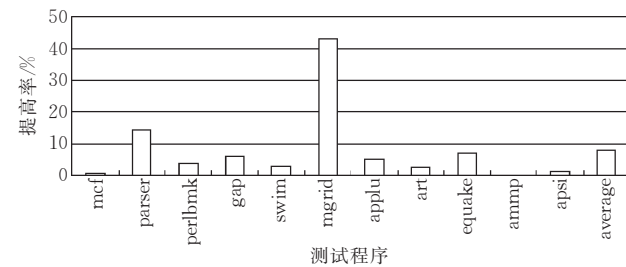


图 9 采用结合访存失效队列状态的预取策略 IPC 值提高比例

图 10 为采用结合访存失效队列状态的预取策略前后的处理器内存占用率(处理器发送正常访存请求时, 内存忙的比例)的变化图, base 表示没有采用结合访存失效队列状态的预取策略处理器的内存占用率, prefetch 表示采用结合访存失效队列状态的预取策略处理器的内存占用率。从图中可以看出, 采用预取技术后, 内存占用率并没有明显的增加, 预取准确性的提高减少了带宽的浪费, 预取发起时机

的优化减小了预取对处理器正常访问请求的影响。值得注意的是 swim 测试程序的空间局部性很好, 预取准确性很高, 但采用预取技术前处理器的内存占用率已经很大, 因此没有足够的带宽进行预取, 所以 swim 程序的 IPC 值提高并不显著。结果表明, 处理器带宽是预取技术发挥作用的保证, swim 测试程序的预取效果就是带宽敏感的一个例子。

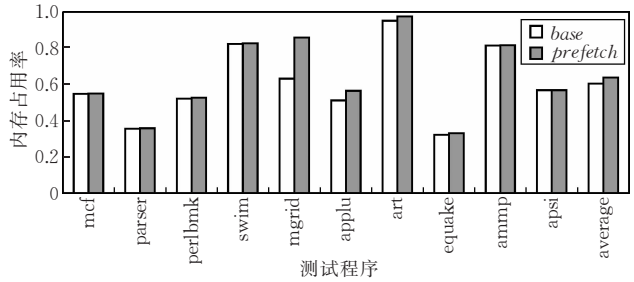


图 10 内存占用率

4.4 预取技术对访存延时的影响

采用结合访存失效队列状态的预取策略, 当 Cache 失效请求在位于处理器内部的 Stream Buffer 命中时, 对应数据立即返回 Cache, 因此 Stream Buffer 命中率直接影响到处理器的平均访存操作延时。采用预取策略后, 处理器的访存延时定义为

$$Latency = P_{hit_rdy} \times L_{hit_rdy} + P_{hit_no_rdy} \times L_{hit_no_rdy} + P_{miss} \times L_{miss} \quad (5)$$

$$P_{hit_rdy} + P_{hit_no_rdy} + P_{miss} = 1 \quad (6)$$

在式(5)中, $Latency$ 表示处理器 Cache 失效的平均访存延时。

P_{hit_rdy} 表示 Cache 失效请求在 Stream Buffer 命中已经返回数据项的次数与处理器所有 Cache 失效数的比值, 与预取覆盖率($Coverage$)相同, 即 $P_{hit_rdy} = Coverage = 0.2$ 。

$P_{hit_no_rdy}$ 表示 Cache 失效请求在 Stream Buffer 命中但数据还没有返回的次数与处理器所有 Cache 失效数的比值, 等于流提取有效性($Efficiency$)减去预取覆盖率($Coverage$), 即 $P_{hit_no_rdy} = Efficiency - Coverage = 0.4 - 0.2 = 0.2$ 。

P_{miss} 表示 Cache 失效请求在 Stream Buffer 没有命中的次数与处理器 Cache 失效数的比值, 根据式(6), $P_{miss} = 1 - P_{hit_rdy} - P_{hit_no_rdy} = 1 - 0.2 - 0.2 = 0.6$ 。

L_{hit_rdy} 表示 Cache 失效请求在 Stream Buffer 命中且预取数据在 Stream Buffer 中, 所需的数据返回延时。由于预取电路在处理器核实现, 因此可以直接将数据取回访存失效队列并填充 Cache。访问

Stream Buffer 的延时远小于访问内存的延时,所以 L_{hit_rdy} 近似等于 0.

L_{miss} 表示 Cache 失效请求在 Stream Buffer 没有命中,处理器访问内存的延时. 根据处理器的配置, L_{miss} 的值为 50 个时钟周期.

$L_{hit_no_rdy}$ 表示 Cache 失效请求在 Stream Buffer 命中但需要等待预取数据返回后才能填充 Cache, 所需的数据返回延时. 由于 Stream Buffer 命中, 虽然数据没有返回但预取请求先于处理器正常访存请求发出, 节省了访问内存的时间. $L_{hit_no_rdy}$ 为处理器正常访存请求的延时和 Stream Buffer 命中且预取数据已经返回的延时的平均值, 即 $L_{hit_no_rdy} = (L_{hit_rdy} + L_{miss})/2 = 50/2 = 25$ 个时钟周期.

由 $P_{hit_rdy} = 0.2$, $P_{hit_no_rdy} = 0.2$, $P_{miss} = 0.6$, $L_{hit_rdy} = 0$, $L_{hit_no_rdy} = 25$ 个时钟周期, $L_{miss} = 50$ 个时钟周期, 根据式(5), 采用结合访存失效队列状态的预取策略后, 处理器的平均访存延时 $Latency = P_{hit_rdy} \times L_{hit_rdy} + P_{hit_no_rdy} \times L_{hit_no_rdy} + P_{miss} \times L_{miss} = 0.2 \times 0 + 0.2 \times 25 + 0.6 \times 50 = 35$ 个时钟周期. 因此, 采用结合访存失效队列状态的预取策略处理器的平均访存延时减少了 $(L_{miss} - Latency)/L_{miss} = (50 - 35)/50 = 30\%$.

4.5 预取参数对性能的影响

(1) 流过滤表项数(stream filter number)

由于程序中存在循环、分支及递归等复杂的结构, 程序运行时 Cache 失效的空间局部性特征不一定是在连续的序列中体现, 而是在一定时间窗口内几个具有空间局部性的 Cache 失效序列交替存在. 如果流过滤表(Stream Filter)项数少, 会造成有效预取流在被提取之前就从流过滤表替换出去. 如果流过滤表项数过多, 则会定位到很长时间才会被用到一次的不活跃的预取流, 从而替换出 Stream Buffer 中的有效预取流, 由于它本身用到的时间间隔又很长, 在被访问之前可能又被活跃的预取流替换出去, 这样既浪费了处理器的带宽又没有达到预取的效果. 此外, 指令 Cache 和数据 Cache 失效的空间局部性的存在是有一定范围的, Stream Filter 中的一项如果在比较长的时间间隔内没有被定位为一个有效预取流, 这一项在流过滤表中存在的价值就不大了, 即使 Stream Buffer 的个数无限大, 流过滤控制电路对 Stream Filter 项数的需求也是有限的.

为了分析流过滤表的项数对处理器性能的影响, 本文在各种 Stream Filter 项数的配置下运行 SPEC CPU2000 测试程序并进行性能评测, 比较各

种配置下程序的流提取有效性和程序的 IPC 值. 图 11 为 Stream Buffer 个数为 16, 即指令和数据 Stream Buffer 个数分别为 8 时, 流提取的有效性随 Stream Filter 项数增加的变化情况. 从图中可以看出, Stream Buffer 项数为 16 项时, Stream Filter 从 2 项增加到 16 项, 即指令和数据 Stream Filter 分别从 1 项增加到 8 项时, 流提取的有效性随着 Stream Filter 项数的增加而增加, 当 Stream Filter 项数从 16 项增加到 32 项, 由于 Stream Buffer 中预取流的相互替换导致流提取有效性并没有增加反而降低. 图 12 为 Stream Buffer 个数是 16 时程序的 IPC 值随 Stream Filter 项数的变化情况. 从图中可以看出, 当 Stream Buffer 为 16 个时, 流过滤表项数是 16 项时程序的 IPC 值达到最大值.

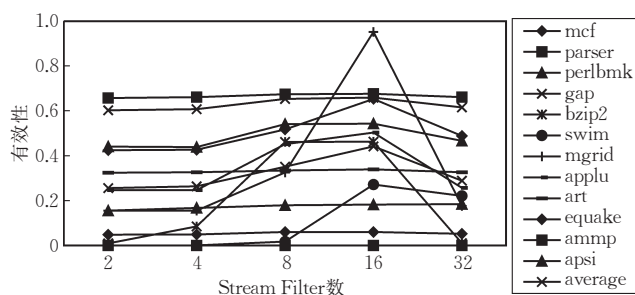


图 11 流提取有效性随 Stream Filter 项数的变化情况 (Stream Buffer 数: 16)

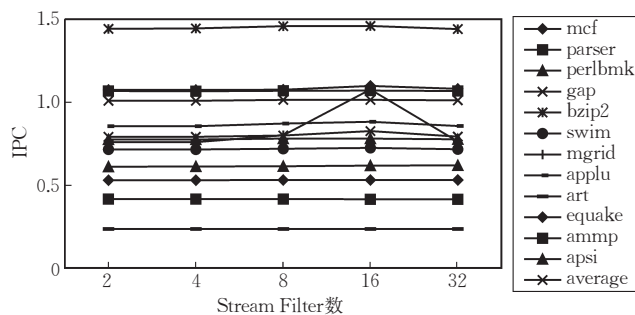


图 12 程序的 IPC 值随 Stream Filter 项数的变化情况 (Stream Buffer 数: 16)

图 13 和图 14 为将 Stream Buffer 个数增加到 32, 流提取的有效性及程序的 IPC 值分别随 Stream Filter 项数增加的变化情况. 增加 Stream Buffer 个数进行分析可以降低 Stream Buffer 个数不足引起流的替换对 Stream Filter 项数分析的影响, 从程序的空间局部性来分析 Stream Filter 的项数需求. 由图 13 可见, Stream Buffer 个数为 32 时, Stream Filter 从 2 项增加到 16 项, 流提取的有效性随着 Stream Filter 项数的增加而增加, 当 Stream Filter 项数从 16 项增加到 32 项, 除 swim 程序外流提取

的有效性趋于稳定. 由图 14 可见,Stream Buffer 个数为 32 时,Stream Filter 从 2 项增加到 16 项,程序的 IPC 值随着 Stream Filter 项数的增加而增加,当 Stream Filter 项数从 16 项增加到 32 项,程序的 IPC 值基本保持不变. Swim 程序虽然流提取有效性提高显著,但由于处理器的正常访存已经非常频繁,带宽因素制约了预取的发出,因此 swim 程序的 IPC 值没有明显提高. 结果表明,流过滤表项数为 16 项,即指令和数据的流过滤表分别为 8 项,有利于流的提取. 量化的分析方法证明本文采用 16 项 Stream Filter 是比较合适的流过滤表项数.

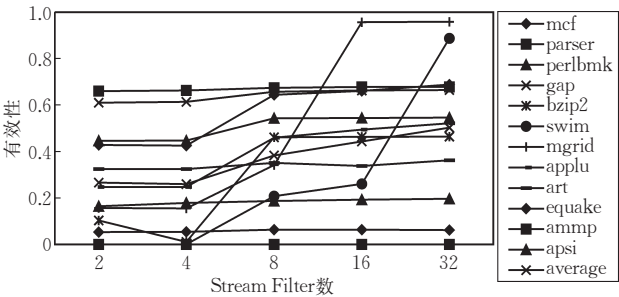


图 13 流提取有效性随 Stream Filter 项数的变化情况 (Stream Buffer 数: 32)

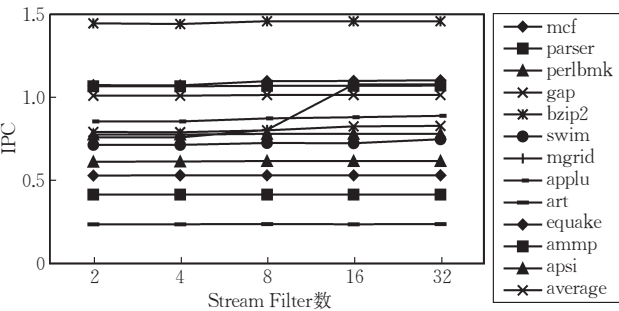


图 14 程序的 IPC 值随 Stream Filter 项数的变化情况 (Stream Buffer 数: 32)

(2) 流缓冲区个数(stream buffer number)

Stream Buffer 个数决定其所能存放的预取流个数,因此 Stream Buffer 的个数直接关系到流提取的有效性,进而影响处理器的性能. 通过上述 Stream Filter 项数的分析,结果表明比较合适的 Stream Filter 项数是 16,因此选择 Stream Filter 为 16 项来分析 Stream Buffer 个数对流提取的有效性和程序 IPC 值的影响.

流提取的有效性随 Stream Buffer 个数变化情况如图 15 所示,程序的 IPC 值随 Stream Buffer 个数的变化情况如图 16 所示. 由图 15 和图 16 可见,当 Stream Buffer 的个数增加时,流提取的有效性也随之增加,并且当 Stream Buffer 个数为 16 时,几乎

所有测试程序的流提取有效性基本趋于稳定,程序 IPC 值趋于最大值. Stream Buffer 的个数增加为 32 时,硬件成本随之增加,但处理器的性能并没有得到明显提升. 实验结果表明,本文采用 16 个 Stream Buffer,即用来进行指令和数据预取的 Stream Buffer 个数分别为 8 个,是比较合适的 Stream Buffer 个数.

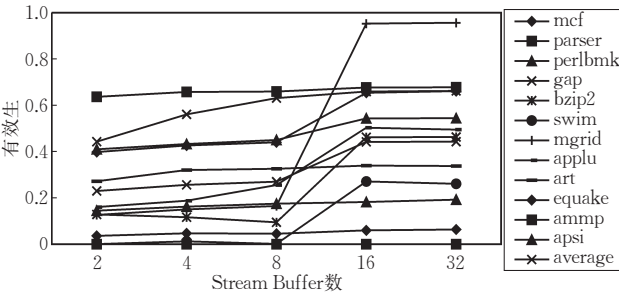


图 15 流提取有效性随 Stream Buffer 个数的变化情况

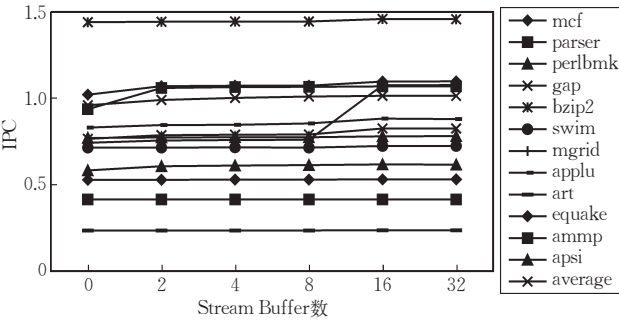


图 16 程序的 IPC 值随 Stream Buffer 个数的变化情况

5 总结与未来工作

本文基于对 SPEC CPU2000 程序的指令 Cache 和数据 Cache 失效行为的深入分析,提出结合访存失效队列状态的预取策略,并给出了性能评测结果与分析. 结合访存失效队列状态的预取策略,提出了利用访存失效队列信息来指导预取的新思路,降低了预取造成的带宽浪费,提高了处理器系统的性能. 结果表明,采用结合访存失效队列状态的预取策略后,处理器的平均访存延时减少了 30%, SPEC CPU2000 程序的 IPC 值平均提高了 8.3%.

随着半导体技术的发展,芯片集成度增加,利用片上晶体管资源充分挖掘线程级并行成为未来高性能处理器发展的一个趋势. 在片上多核多线程环境中,多核多线程竞争单芯片有限的 Cache、带宽等存储资源,造成访存冲突加剧,传统的访存问题变得更加突出. 本文未来的研究工作主要关注于适用于多核多线程环境的预取策略的研究,并进行更加深入的性能分析.

参 考 文 献

- [1] Wulf W, McKee S. Hitting the memory wall: Implications of the obvious. *ACM Computer Architecture News*, 1995, 23(1): 20-24
- [2] Acquaviva J T. Data prefetching efficiency on two commercial systems//*Proceedings of the 5th European SGI/Cray MPP Workshop*. Bologna, Italy, 1999
- [3] Jouppi N P. Improving direct-mapped Cache performance by the addition of a small fully-associative Cache and prefetch buffers//*Proceedings of the 17th Annual International Symposium on Computer Architecture (ISCA' 90)*. Seattle, Washington, USA, 1990: 364-373
- [4] Palacharla S, Kessler R. Evaluating stream buffers as a secondary Cache replacement//*Proceedings of the 21st International Symposium on Computer Architecture (ISCA' 94)*. Chicago, Illinois, 1994: 24-33
- [5] Iacobovici Sorin, Spracklen Lawrence, Kadambi Sudarshan, Chou Yuan, Abraham S G. Effective stream-based and execution-based data prefetching//*Proceedings of the 18th Annual International Conference on Supercomputing*. Malo, France, 2004: 1-11
- [6] O'Connell F P, White S W. POWER3: The next generation of PowerPC processors. *IBM Journal of Research and Development*, 2000, 44(6): 873-884
- [7] Horel Tim, Lauterbach Gary. UltraSparc-III: Designing third-generation 64-bit performance. *IEEE Micro*, 1999, 19(3): 73-85
- [8] Burger D, Goodman J R, Kagi Alain. Memory bandwidth limitations of future microprocessors//*Proceedings of the 23rd International Symposium on Computer Architecture (ISCA-23)*. Pennsylvania, USA, 1996: 78-89
- [9] Kroft D. Lockup-free instruction fetch/prefetch Cache organization//*Proceedings of the 8th Annual Symposium on Computer Architecture (ISCA' 81)*. Minneapolis, Minnesota, USA, 1981: 81-87
- [10] Charney M J, Puzak T R. Prefetching and memory system behavior of the SPEC95 benchmark suite. *IBM Journal of Research and Development*, 1997, 41(3): 265-286
- [11] Hu Wei-Wu, Zhang Fu-Xin, Li Zu-Song. Microarchitecture of the Godson-2 processor. *Journal of Computer Science and Technology*, 2005, 20(2): 243-249
- [12] Standard Performance Evaluation Corp. CPU2000 documentation. <http://www.spec.org/osg/cpu2000/docs>, 2000



HUAN Dan-Dan, born in 1979, Ph. D.. Her research interests include high performance computer architecture, operating system and VLSI design.

LI Zu-Song, born in 1977, Ph. D.. His research inter-

ests include high performance computer architecture, verification and VLSI design.

HU Wei-Wu, born in 1968, Ph. D., professor, Ph. D. supervisor. His research interests include high performance computer architecture, parallel processing and VLSI design.

LIU Zhi-Yong, born in 1946, Ph. D., professor, Ph. D. supervisor. His research interests include computer architecture, algorithm, parallel processing and computing network.

Background

As processor performance increases, there is a corresponding increase in the demand on the memory system. This paper introduces the prefetching policy using miss queue information. The Godson series processors are the first attempt to design high performance general-purpose microprocessor in China. Prefetching policy using miss queue information im-

proves the performance of Godson-2 processor efficiently. This work is supported by the National Science Foundation of China (60325205), the National High Technology Research and Development Program (863 Program) of China (2005AA119020) and the National Basic Research Program (973 Program) of China (2005CB321600).