一种基于值预测和指令复用的按序处理器预执行机制

党向磊,王箫音,佟 冬,陆俊林,易江芳,王克义
(北京大学微处理器研究开发中心,北京 100871)

摘 要: 为提高按序处理器的性能和能效性,本文提出一种基于值预测和指令复用的预执行机制(PVPIR).与传统预执行方法相比,PVPIR 在预执行过程中能够预测失效 Load 指令的读数据并使用预测值执行与该 Load 指令数据相关的后续指令,从而对其中的长延时缓存失效提前发起存储访问以提高处理器性能.在退出预执行后,PVPIR 通过复用有效的预执行结果来避免重复执行已正确完成的指令,以降低预执行的能耗开销.PVPIR 实现了一种结合跨距(Stride)预测和 AVD(Address-Value Delta)预测的值预测器,只记录发生过长延时缓存失效的 Load 指令信息,从而以较小的硬件开销取得较好的值预测效果.实验结果表明,与 Runahead-AVD 和 iEA 方法相比,PVPIR 将性能分别提升 7.5%和 9.2%,能耗分别降低 11.3%和 4.9%,从而使能效性分别提高 17.5%和 12.9%.

关键词: 预执行;值预测;指令复用;访存延时包容
中图分类号: TP302.7 **文献标识码:** A **文章编号:** 0372-2112 (2011) 12-2880-04

A Pre-Execution Mechanism Based on Value Prediction and Instruction Reuse for In-Order Processors

DANG Xiang-lei, WANG Xiao-yin, TONG Dong, LU Jun-lin, YI Jiang-fang, WANG Ke-yi
(*Microprocessor Research & Development Center of Peking University, Beijing* 100871, *China*)

Abstract: To improve the performance and energy-efficiency of in-order processors, this paper proposes a novel hardware mechanism, pre-execution based on value prediction and instruction reuse(PVPIR). If a load instruction incurs a long-latency cache miss, PVPIR predicts its data value and uses the predicted value to pre-execute the following dependent instructions, including loads that incur long-latency misses, thus improving the performance. To reduce the energy consumption, PVPIR reuses the valid pre-executed results and thus avoids the re-execution of completed instructions. PVPIR also implements a hybrid value predictor which is a combination of stride prediction and address-value delta(AVD) prediction. The predictor only records history value for loads that have incurred long-latency misses, thus gaining good prediction results with little overhead. Experimental results demonstrate that PVPIR improves the performance by 7.5% and 9.2% while decreases the energy consumption by 11.3% and 4.9%, thus improving the energy-efficiency by 17.5% and 12.9%, as compared to Runahead-AVD and iEA, respectively.

Key words: pre-execution; value prediction; instruction reuse; load latency tolerance

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.预执行技术[3~6]在流水线因长延时缓存失效而停顿时能够预先执行失效访存指令的后续指令,通过将多个缓存失效的延时相重叠来提高处理器的单线程性能.但是,因操作数未就绪,处理器无法预先执行与失效访存指令数据相关的后续指令,特别是其中的长延时访存指令,从而限制了对处理器性能的优化效果[7].

本文提出一种基于值预测和指令复用的预执行机制(Pre-execution based on Value Prediction and Instruction Reuse, PVPIR).在预执行指令时,PVPIR 预测发生长延时缓存失效(本文为 L2 Cache 失效)的 Load 指令的读数据并使用预测值执行与该 Load 指令数据相关的后续指令,以对其中的长延时缓存失效提前发起存储访问,从而提高对处理器性能的优化效果.在退出预执行后,PVPIR 能够复用有效的预执行结果,包括使用正确的预测值产生的结果,以避免重复执行已正确完成的指令,从而降低预执行的能耗开销.PVPIR 实现的值预测器能够同时对符合跨距(Stride)[8]和 AVD(Address-Value Delta)[7]模式的 Load 读数据进行捕获和预测,并且只记录发生过长延时缓存失效的 Load 指令的历史信息,从而以较小的硬件开销取得较好的值预测效果.

收稿日期:2011-04-21;修回日期:2011-05-05
基金项目:国家 863 高技术研究发展计划(No.2006AA010202);中国博士后科学基金资助项目(No.20110490208)

2 相关工作

已有的预执行技术可主要分为 Runahead 方法[3~5]和 Multipass 方法[6]两类.Dundas 等人在文献[4]中首次提出 Runahead 方法,用于提升按序执行处理器中 DCache 的性能.Mutlu 等人将 Runahead 方法应用于乱序执行处理器,用于构建大的指令窗口以改善对访存延时的包容[5].Wang 等人面向按序执行处理器提出了高能效的 iEA 方法[3],通过结果复用和收益预测等机制对 Runahead 进行方法改进和效率优化.Multipass 方法[6]通过插入 RESTART 指令实现在一段指令片段上进行多次预执行,但需要软硬件配合完成且会造成较大的能耗开销.

在上述预执行方法中,与失效 Load 指令数据相关的后续指令因操作数未就绪无法预先执行,因此无法对其中的长延时缓存失效提前发起存储访问,从而限制了对处理器性能的优化效果.文献[7]提出了结合值预测技术[8]的 Runahead-AVD(简称为 R-AVD)方法,通过 AVD 值预测提前获得失效 Load 指令的读数据,以预先执行与该 Load 指令数据相关的后续指令,从而改善对处理器性能的优化效果.

Runahead 方法[4,5]和 R-AVD 方法[7]在退出预执行后需重新执行所有预执行指令,包括已产生有效结果的指令,造成了性能和能耗的浪费.已有研究[3,6,9]通过结合指令复用技术[10]来解决该问题,在预执行过程中保存产生的有效结果,并在退出预执行后复用保存的有效结果以避免重复执行已正确完成的指令,从而降低预执行的能耗开销.

可以看出,值预测侧重于提高预执行的性能,指令复用侧重于降低预执行的能耗开销.为充分发挥两者的优势,本文提出了 PVPIR 方法,在预执行中同时结合两种技术以取得更好的能效优化效果.

此外,跨距预测[8]面向相邻两次执行的读数据之差为常值的 Load 指令进行值预测,而 AVD 预测[7]面向访存地址与读数据之差为常值的 Load 指令进行值预测,两者具有一定的互补性.为了取得较好的值预测效果,PVPIR 实现了一种混合的 Load 值预测器(Load Value Predictor,LVP),能够同时对符合跨距[8]和 AVD[7]模式的 Load 指令进行值预测.

3 基于值预测和指令复用的预执行机制

3.1 工作流程

PVPIR 的工作流程可分为正常执行、预执行和合并结果三个阶段.在初始时刻,处理器处于正常执行阶段,逐条执行并提交指令.当检测到数据访问发生长延时缓存失效,处理器使用检查点(Checkpoint)对寄存器进行备份,之后进入预执行阶段.

在预执行阶段,处理器不用等待失效访存指令完成主存访问即可继续执行后续指令,所有指令均不提交结果.其中,与失效访存指令数据无关的指令可正常执行并获得有效结果,标记为 VALID 状态.

若引发预执行的是 Load 指令或预先执行 Load 指令时发生长延时缓存失效,PVPIR 查找 LVP.若不能获得预测值,则将目标寄存器标记为 INV 状态,与其数据相关的后续指令直接被移出流水线并将结果标记为 INV 状态.若成功获得预测值,则将预测值写入目标寄存器并使用预测值执行与该 Load 指令数据相关的后续指令.对于引发预执行的 Load 指令,在退出预执行时其读数据已被取回,可以判断预测值是否正确,若正确,则可复用使用该预测值产生的结果,因此,将该指令及与其数据相关指令的结果标记为 VP_R 状态;对于预执行期间发生长延时缓存失效的 Load 指令,在退出预执行时其读数据可能尚未取回,无法判断预测值是否正确,因此,不可复用使用该预测值产生的结果,将该指令及与其数据相关指令的结果标记为 VP_P 状态.

在预执行过程中,指令结果的状态会随指令间的数据相关关系传递(由源操作数传递给目标操作数),因此,PVPIR 为每个寄存器设置 2 位的状态位(Status Flag).对于 Store 指令,为避免更新体系结构状态,PVPIR 使用 Store Cache[3]保存其存储数据并为每个字设置 2 位的状态位,以将存储数据及其状态传递给后续的 Load 指令.同时,处理器按顺序将指令结果及其状态保存在指令复用队列(Instruction Reuse Queue,IRQ)中,以在退出预执行后进行结果复用.IRQ 的每个表项代表一条指令,包含 4 个域,如图 1(a)所示.InstRes 为指令的预执行结果.Valid(V)和 VP_R 位分别与指令结果的 VALID 和 VP_R 状态相对应.Reuse(R)位标识指令结果是否可复用.

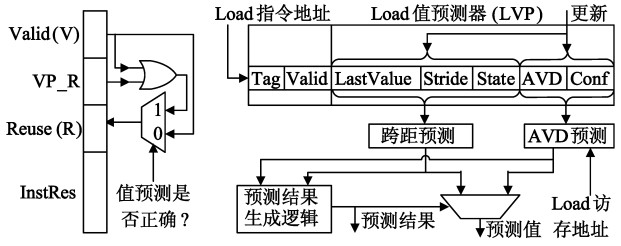
在引发预执行的指令完成主存访问后,若该指令为 Load 指令且成功获得预测值,则使用真实的读数据检查值预测是否正确并生成 IRQ 各表项的 R 位.若值预测正确,则 V 位或 VP_R 位为 1 的表项均可以复用,将相应的 R 位置 1;否则,只有 V 位为 1 的表项可以复用,将相应的 R 位置 1.之后,处理器清空流水线,恢复备份的寄存器,转换到合并结果阶段.

在合并结果阶段,处理器将从引发预执行的指令开始重新执行并提交预执行指令.对于每一条预执行指令,处理器检查 IRQ 中的相应表项,若 R 位为 1,则直接复用预执行结果,不再重复执行;若 R 位为 0,则重新执行该指令.若再次检测到数据访问发生长延时缓存失效,处理器对寄存器进行备份并重新进入预执行阶段.当 IRQ 为空时,表明正常执行已经追上预执行,处

理器返回正常执行阶段.

3.2 Load 值预测器的设计

LVP 的内部结构如图 1(*b*)所示,采用指令地址索引的组相联结构,每个表项保存一条 Load 指令的历史信息,包含 7 个域.Tag 域为指令地址的高位,用于标识不同的 Load 指令.Valid 域用于标识表项是否有效.Last-Value、Stride 和 State 域用于实现跨距预测,分别记录 Load 指令最近一次执行的读数据、相邻两次执行的读数据之差和跨距预测当前所处的状态(训练状态或预测状态).AVD 和 Conf 域用于实现 AVD 预测,分别记录 Load 指令最近一次执行的访存地址与读数据之差和 AVD 预测的可信度.



(*a*) IRQ的表项结构 (*b*) LVP的内部结构

图1 IRQ的表项结构和LVP的内部结构

进行值预测时,PVPIR 使用 Load 指令地址查找 LVP.若发生失效,则不可预测.若发现命中表项,则根据表项状态处理.若 State 标识该表项处于预测状态,则使用 LastValue 与 Stride 之和作为预测值;否则,若 Conf 的值大于阈值(本文为 2),则使用 Load 指令的访存地址与 AVD 之和作为预测值;否则,不可预测.在每条 Load 指令获得真实的读数据时,PVPIR 查找并更新 LVP.若发现命中表项,则使用访存地址和读数据对跨距预测和 AVD 预测的相应域进行更新;若发生失效且该 Load 指令发生长延时缓存失效,则为该 Load 指令创建新的表项,并初始化各域;否则,不分配新的表项.

4 实验评估

本文分别采用 SimpleScalar[11] 与 DRAMsim[12] 对处理器及主存系统进行建模,并集成 Wattch[13] 功耗模型,选取 SPEC2000 和 Olden[14] 中的程序进行评测.SPEC2000 中的程序使用 SimPoint[15] 运行 100M 代表程序片段.Olden 中的程序均运行完毕.基准处理器的配置基于 UniCore-2处理器[16] 的典型结构.基准处理器和本文方法的配置参数如表 1 所示.

本文选取 Runahead[5]、R-AVD[7] 和 iEA[3] 方法作为比较对象.Runahead Cache[5] 和 Store Cache[3] 均配置为 1KB 2 路组相联.对于 iEA 方法,Instruction and Result Buffer[3] 配置为 256 表项;对于 R-AVD 方法,AVD 值预测器[7] 配置为 256 表项 2 路组相联.

表 1 基准处理器和本文方法的配置参数

	参数	配置值
流水线		8 级,单发射按序执行,1GHz
L1 I/DCache		16KB,4 路组相联,每行 32 字节
L2 Cache		512KB,8 路组相联,每行 64 字节
主存		150 周期平均访问延时
本文方法	Store Cache	1KB,2 路组相联
	IRQ	256 表项
	LVP	256 表项,2 路组相联

4.1 性能与能耗

四种方法的性能优化效果如图 2 所示,在平均情况下,分别将基准处理器的性能提升 16.0%、18.1%、16.2%和 26.9%.可以看出,与 Runahead 和 iEA 方法相比,结合值预测的 R-AVD 和本文方法获得了更好的性能优化效果.与 R-AVD 方法相比,本文方法通过结合跨距预测和 AVD 预测的值预测器改善了值预测效果,从而可以预执行更多访存指令以隐藏访存延时,进一步提升基准处理器的性能.

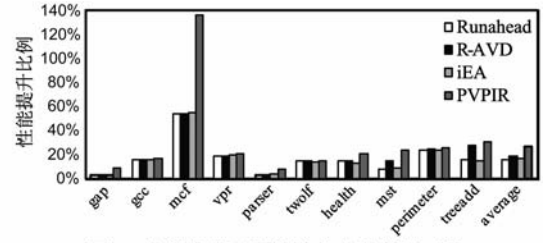


图2 对基准处理器的性能优化效果对比

四种方法的能耗开销如图 3 所示,在平均情况下,分别使基准处理器的能耗增加 18.1%、18.4%、10.3%和 5.0%.实验结果表明,通过与值预测和指令复用相结合,本文方法能够有效提高处理器性能,并减少指令执行开销,从而在能耗方面获得优势.

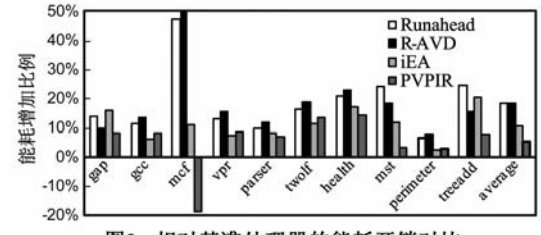


图3 相对基准处理器的能耗开销对比

4.2 能效性与硬件开销

本文采用能耗延迟积(Energy-Delay Product, EDP)[17] 作为能效性的度量指标,用于衡量处理器设计在性能与能耗两个方面的综合结果(数值越小,能效性越高)[3].以基准处理器的能耗延迟积为基准值,将四种方法的能耗延迟积作规格化,结果如图 4 所示.在平均情况下,Runahead 和 R-AVD 方法分别将基准处理器的能效性降低 1.9%和 0.2%,而 iEA 和本文方法分别将基准处理器的能效性提高 5.1%和 17.3%.与 Runa-

head、R-AVD 和 iEA 方法相比,本文方法取得了更好的性能优化效果且能耗开销更小,从而使能效性分别提高 18.8%、17.5%和 12.9%.

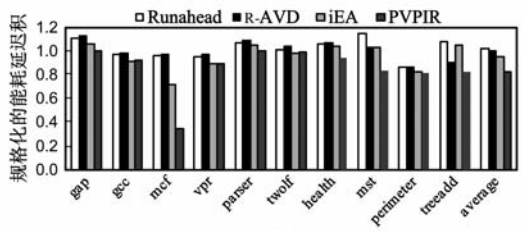


图4　能效性对比

在基准处理器中实现 PVPIR 所增加部件包括 Checkpoint、Status Flag、Store Cache、IRQ 和 LVP,共占用存储容量约为 5KB,仅为 L1 Cache 总容量的六分之一,因此,本文方法硬件开销比较小.

5　结论

本文提出了一种基于值预测和指令复用的预执行机制(PVPIR),使用值预测技术来预先执行与失效访存指令数据相关的长延时访存指令,从而隐藏该类指令的访存延时;同时,通过复用有效的预执行结果来避免重复执行已正确完成的指令,以降低预执行的能耗开销.为了取得较好的值预测效果,PVPIR 实现了一种结合跨距预测和 AVD 预测的值预测器,并通过只记录发生过长延时缓存失效的 Load 指令的历史信息来减少硬件开销.

与已有的预执行方法相比,本文方法以较小的硬件开销,能够有效改善对处理器性能的优化效果,并且具有更小的能耗开销,从而可以在保持按序执行处理器能耗优势的基础上,有效提高处理器的单线程性能,获得更好的能效性.

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.
Wang X Y, 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] 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.

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

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

[7] O Mutlu, et al. Address-value delta(AVD) prediction: A hardware technique for efficiently parallelizing dependent cache misses[J]. IEEE Transactions on Computers, 2006, 55(12): 1491 – 1508.

[8] Y Sazeides, J E Smith. The predictability of data values[A]. Int'l Symposium on Microarchitecture[C]. Los Alamitos, California, USA: IEEE Computer Society, 1997. 248 – 258.

[9] O Mutlu, et al. On reusing the results of pre-executed instructions in a runahead execution processor[J]. IEEE Computer Architecture Letters, 2005, 4(1): 2 – 5.

[10] A Sodani, G S Sohi. Dynamic instruction reuse[A]. Int'l Symposium on Computer Architecture[C]. Denver, Colorado, USA: IEEE Computer Society, 1997. 194 – 205.

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

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

[13] 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.

[14] A Rogers, et al. Supporting dynamic data structures on distributed-memory machines[J]. ACM Trans on Programming Languages and Systems, 1995, 17(2): 233 – 263.

[15] 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.

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

[17] R Gonzalez, M Horowitz. Energy dissipation in general purpose microprocessors[J]. IEEE Journal of Solid-State Circuits, 1996, 31(9): 1277 – 1284.

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、存储系统性能优化和系统芯片设计.
E-mail: dangxianglei@mprc.pku.edu.cn

王箫音(通信作者)　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