自适应存储相关性预测器

班冬松 颜世云 李 礼 杨剑新 路冬冬
(上海高性能集成电路设计中心 上海 201204)

摘 要 访存指令的乱序执行会导致存储相关性冲突。存储相关性预测技术能够减少相关性冲突,提升处理器性能。已有学术研究工作普遍存在硬件开销大、实现复杂度高的不足;商业处理器中的存储相关性预测技术虽然实现简单,但又存在不具有自适应性或不利于利用指令并行性等问题。设计了一种简单、高效的存储相关性预测器SMDP,它具有自适应、易实现、充分利用指令并行性等优点。实验表明,SMDP能有效提高处理器性能,在与实际处理器接近的较小指令窗口配置下,与盲预测机制相比,平均性能提高0.7991%,最高可达4.9225%。

关键词 存储相关性预测,自适应,性能提升
中图法分类号 TP302 **文献标识码** A

Self-adaptive Memory Dependence Predictor

BAN Dong-song YAN Shi-yun LI Li YANG Jian-xin LU Dong-dong
(Shanghai High-Performance IC Design Center, Shanghai 201204, China)

Abstract The out of order execution of store and load instructions always results in memory dependence hazards. MDP (Memory Dependence Prediction) can reduce these hazards, and improve the processor performance. Most of academic research works are complex and spend high hardware cost. Although MDP is implemented simply in commercial processors, it still has some shortages, such as, no self-adaptive ability or blocking ILP (Instruction Level Parallelism). This paper proposed a simple and effective memory dependence predictor SMDP, which possesses self-adaptive ability, low hardware cost, and high ILP. The simulation shows that SMDP improves the processor performance effectively, by 0.7991% on average over blind prediction, and by 4.9225% at best.

Keywords Memory dependence prediction, Self-adaptive, Improving performance

1 引言

现代处理器普遍采用乱序执行技术,以尽可能利用指令级并行性来提升微处理器性能。但访存指令的乱序执行,可能会产生load/store指令的存储相关性冲突:存在写后读相关(访问同一地址)的一对store指令和load指令,如果load指令满足发射条件而早于store发射,先执行的load指令可能会读到错误的值。对于存储相关性冲突,处理器需要采用清空流水线等机制进行恢复,从而降低处理器性能。已有研究[1]表明,总指令数中访存指令(load/store)所占比例高达38%,其中load指令的数量约为store指令的2倍。因此针对数量众多的访存指令,研究存储相关性预测技术,以减少存储相关性冲突,对于有效提升处理器性能具有重要意义。

学术界针对存储相关性预测技术已经开展了大量的研究工作,主要包括基于"store-load对"的相关性预测器[2]、基于Store-Set的相关性预测器[3-5]、基于store-load冲突距离的相关性预测器[6]等。但这些研究成果普遍存在硬件开销大、实现复杂度高的不足,可实现性和性价比难以满足应用需求。商用处理器中使用简单的预测机制进行相关性预测,如alpha 21264中基于LWT(Load Wait Table)的预测技术[7]、Intel Core2中的存储消歧技术[1]。上述预测机制虽然实现简单,但仍有许多不足,如基于LWT的预测机制不具有自适应性,Core2中的存储消歧技术过于保守,不利于利用指令的并行性。因此,目前仍然缺乏能够充分利用指令并行性,具有可实现性高、自适应性等优点的存储相关性预测机制。

本文面向乱序发射微处理器结构,设计了一种低开销、高效的存储相关性预测器SMDP。SMDP具有自适应、易实现、充分利用指令并行性等优点。实验表明,SMDP能够有效提升处理器的性能,在贴近实际处理器参数配置情况下,与load指令直接发射的"盲预测"机制相比,在预测表为4096条目情况下,平均性能提升0.7991%,最高可达4.9225%,并能够以较小的硬件开销实现较高性能,且在1024条目情况下,平均性能提升可达到4096时的80%。

本文第2节介绍存储相关性预测领域的相关工作;第3节详细介绍SMDP的预测过程、预测信息位更新策略,并对SMDP的实现难度与开销进行分析;第4节是实验评估;最后

到稿日期:2012-07-22 返修日期:2012-09-30 本文受国家"核高基"重大专项课题(2009ZX01028-002-001)资助。
班冬松(1982-),男,博士,工程师,研究领域为微处理器结构设计;颜世云(1981-),男,工程师,主要研究领域为微处理器结构设计;李 礼(1981-),男,博士,工程师,主要研究领域为微处理器结构设计与无线网络;杨剑新(1976-),男,高级工程师,主要研究领域为高性能微处理器结构设计;路冬冬(1988-),男,硕士生,主要研究领域为微处理器结构设计。

总结全文。

2 相关研究

Moshovos 等人[2]提出了一种基于“store-load 对”的相关性预测器。该预测器使用两个表：MDPT 和 MDST。MDPT 记录发生了冲突的 store-load 对，MDST 中记录发生冲突的 store 指令是否执行的信息。Load 指令发射前查找 MDPT，如果存在匹配的条目，则根据 MDST 中对应的 store 指令的执行情况来确定是否发射。

George Z. 等人[3]提出了一种基于 Store-Set 的相关性预测器。一条 load 指令的 Store-Set 是与其存在相关性的 Store 指令的集合。初始时，所有 load 指令的 Store-Set 为空，load 指令乱序执行。若 store 指令 S1 与 load 指令 L1 产生相关性冲突，则将 S1 加入 L1 的 Store-Set，记为 A；如果有另外一条 store 指令 S2 与 L1 也产生相关性冲突，同样将 S2 加入 Store-Set A。L1 发射前检查 store-set A，等待 Store-Set A 中的所有 store 指令发射后才能发射。许多研究人员对 Store-Set 方法进行了改进。Subramaniam 等人[4]提出一种类似 Store-Set 的预测机制，但用直接映射表代替了 Store-Set 方法中的全相联表，减少了存储空间和预测处理时间。Onder 等人[5]提出了一种 Enhanced Store-Set 预测器，弥补了原有 Store-Set 方法中同一集合中的 store 指令需要顺序发射的不足，提高了 IPC。

Yoaz 等人[6]提出了一种基于 store-load 冲突距离的相关性预测器。该预测器利用冲突历史表 CHT (Collision History Table)实现，每个 CHT 条目包含 load 指令地址、冲突距离、预测标志。如果 load 指令被预测为冲突，则需要等待冲突距离之内的 store 指令全部发射后，load 指令才可以发射。

上述学术研究成果虽然预测效果较好，但往往存在方案复杂、开销大等不足，不易在实际处理器中实现。商用处理器中往往使用简单、实现复杂度低的预测机制。Alpha21264 使用 Load Wait Table(LWT)进行预测[7]。LWT 的每个条目包含 1bit 的初值为 0 的预测信息位。Load 指令发射前查找 LWT，如果预测信息位为 0，则可以乱序发射。若 load 指令发生相关性冲突，则对应的预测信息位被置 1，后续 load 指令只能顺序发射。LWT 会“定期”被刷新，以防止出现不必要的等待。Intel 在 Core 2[1] 等后续产品中应用了存储消歧(Memory Disambiguation)技术：预测表每个条目包含一个初值为 0 的饱和计数器。如果 load 指令提交时没有冲突，则计数器在 load 指令提交时加 1。一旦 load 指令发生相关性冲突，计数器将清零。当计数器饱和时，load 指令才被预测为乱序发射。

Alpha21264 的基于 LWT 的存储相关性预测器实现简单，但没有自适应性，不能根据指令执行的动态情况进行自适应更新。Intel 的存储相关性预测器必须使连续的具有相同地址的 load 指令都不产生冲突，才能使后续的 load 指令被预测为乱序发射。虽然预测正确率较高，但由于预测策略过于保守，可能会导致过多的 load 指令延迟发射，不利于利用指令级并行性。

本文提出的 SMDP 预测器，与传统的学术研究成果相比，具有设计简单、易于实现的特点；与 LWT 预测器相比，能够根据指令执行的动态情况自适应更新预测信息；与 Intel 的预测机制相比，是相对激进、高效的，能充分利用指令级并行性。

3 SMDP 预测器

图 1 给出了一种基本的处理器流水线划分方式：取指站台、译码站台、重命名站台、发射站台、执行站台、退出站台。本文提出的存储相关性预测机器 SMDP 在重命名站台实现。

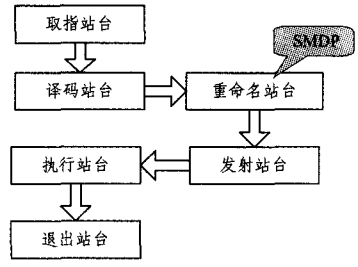

图 1 SMDP 在流水线中的实现阶段

本节首先介绍利用 SMDP 对 load 指令进行预测和处理的过程；然后重点介绍预测状态机，说明相关性预测表 DPT 中的预测信息位如何根据具体冲突情况进行更新；最后对 SMDP 的硬件开销和实现难度进行分析。

3.1 预测过程

我们为寄存器重命名部件设置一个直接映射的相关性预测表 DPT(Dependence Prediction Table)，每个条目包括 2bit 的预测信息位、1bit 的有效位、tag 信息。对 2^n 条目的 DPT，使用 PC[$n-1:0$]进行索引；为节省硬件开销，tag 信息为 PC[$15:n$]。

具体预测处理流程如下：

1. 当指令进入重命名站台时，首先对输入的指令类型进行判断：如果为 load 指令，则用该 load 指令的 pc 值低位索引 DPT，根据对应的预测信息位，输出预测结果。例如，信息位为 00 或 01 时，预测 load 指令可乱序发射；信息位为 10 或 11 时，预测 load 指令只能顺序发射。如果 DPT 中不存在该 load 指令对应的条目，则建立一条新的条目，预测该 load 指令可乱序发射。

2. 将预测结果记录到指令发射队列(Issue Queue)中该 load 指令对应的条目上。发射部件根据预测结果选择是否乱序发射。如果预测结果为顺序发射，则只有当该 load 指令之前的所有 store 指令发射后，load 指令才发射；否则，只要 load 指令满足发射条件，即可发射。

3. 如果 load 指令发生存储相关性冲突，则清空流水线，load 指令重新发射，并同时更新 DPT 中 load 指令对应条目的预测信息位(更新策略见 3.2 节)。

4. 如果 load 指令正常提交，说明没有发生冲突，更新 DPT 中该 load 指令对应条目的预测信息位(更新策略见 3.2 节)。

Load 指令相关性预测与处理流程如图 2 所示。

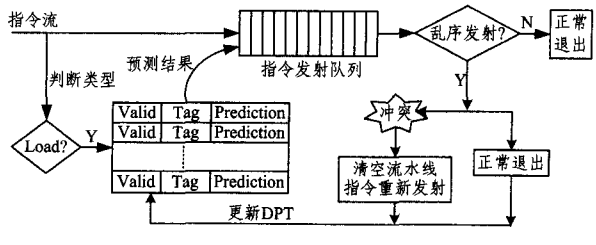

图 2 Load 指令预测及处理过程

3.2 预测信息位更新策略

SMDP 能够根据程序运行的动态情况，对预测表 DPT 进行自适应更新。DPT 中每个条目内存储 2bit 的预测信息位：00 和 01 表示预测乱序发射，10 和 11 表示预测顺序发射。乱序发射的 load 指令在正常退出或产生冲突时，对 DPT 中对应条目的预测信息位进行更新。预测信息位更新策略如图 3 所示。如果 load 指令正常退出，则说明“无冲突”；如果 store 指令在地址计算完成后，发现在 load 队列中存在比其年轻且访问同一地址的 load 指令，则说明产生 load/store“冲突”。

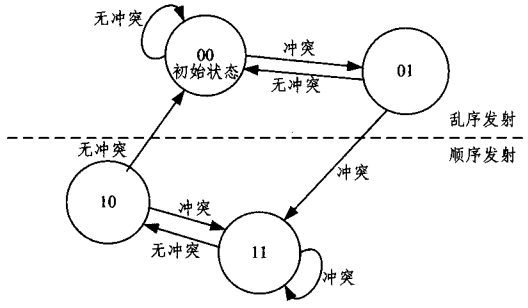

图 3 预测信息位更新策略

SMDP 能够让被预测为顺序发射的 load 指令在保证正确率的前提下，尽早乱序发射，以充分利用指令级并行性。相关性冲突通常具有时间局部性[2]。即一对 load/store 指令如果发生一次冲突，会以较大概率接连发生几次冲突，然后又变成连续的不冲突状态。因此若 load 指令被预测为冲突的 load 指令，并连续几次在实际执行中没有产生冲突，则说明该指令后续产生冲突的概率较低，可以将该 load 指令预测为乱序发射。图 3 中，如果预测为顺序发射的 load 指令，连续 2 次正常退出无冲突，则该 load 指令后续将被预测为乱序发射（即从 11 回到 00 状态）。

10 和 11 状态时，load 指令将被预测为顺序发射，不会产生冲突，但图 3 中仍然对 10 和 11 状态设置了产生冲突时的跳转路径。这是由于 DPT 条目数量有限，会出现条目被淘汰替换的情况。比如，一条被预测为乱序发射的 load 指令 L1 在冲突发生后更新 DPT 中对应条目的预测信息位时，L1 的条目可能被另一条 load 指令 L2 替换，并且 L2 的预测信息位为 10（或 11）。对这种情况，为简化处理逻辑，SMDP 不对 L1 和 L2 的条目进行区分，统一按照图 3 的更新策略修改预测信息位。

3.3 实现分析

SMDP 易于实现。乱序发射处理器的最简单的存储相关性预测机制是盲预测（blind prediction）机制。盲预测始终预测乱序发射 Load 指令不会引起存储相关性冲突，即只要 load 指令满足发射条件就立即发射；一旦预测失败，再通过相应的恢复机制来处理冲突。对基于盲预测机制的微处理器，在实现 SMDP 时只需进行 2 处主要逻辑更改：1）在寄存器重命名站台增加预测表 DPT；2）在指令发射队列中，每个条目需要增加 1bit 的预测位 PB（Prediction Bit）。同时修改指令发射逻辑：如果 PB=1，则发射部件将顺序发射 load 指令，否则乱序发射。上述的逻辑更改与原有设计粘连性较弱，工作量较小，易于在实际处理器中实现。

SMDP 具有较小的硬件开销。从第 4 节的实验结果可以看出，DPT 为 1024 个条目时即能够实现较大的性能提升，平均性能提升可达到 4096 时的 80%，最大性能提升可达4.77%。DPT 每个条目中存储的预测信息位为 2bit，tag 信息为 6bit，有效位为 1bit，1024 个条目时的 DPT 硬件存储开销仅为 9kb。指令发射队列通常有 10～20 个条目，在实现时每个条目只需增加 1bit 预测位；与 DPT 的开销相比，指令队列上增加的硬件开销非常小，可以忽略。

4 实验评估

本实验通过运行 SPEC2000 课题，评估 SMDP 对处理器性能的提升情况。本实验评估与文献[2-4]实验相比，指令窗口相对较小，更贴近实际处理器参数配置，更能够反映 SMDP 应用到实际处理器中的性能提升效果。具体参数配置如表 1 所列。

表 1 实验配置参数

Parameter	Value
Decode width	2instructions/cycle
Issue width	3 instructions/cycle
Commit width	4 instructions/cycle
Issue Queue Size	Int Queue 12, Float Queue 10
Load Queue Size	16
Store Queue Size	16
L1 ICache	32kB, 2-way, LRU replacement policy
L1 DCache	32kB, 4-way, LRU replacement policy
L2 Cache	512kB, 8-way, LRU replacement policy

微处理器性能的一种直观度量方式是评测程序执行所需时间，即程序的执行时间越短，性能越好。本文实验将使用盲预测机制的处理器运行时间作为参考基准值 R，与应用 SMDP 预测器的运行时间 M 进行对比，$(R-M)/R$ 即为 SMDP 对处理器的性能提升比例。

图 4 显示了 DPT 条目数量为 4096 时，spec2000 各课题的性能提升比例。从图 4 可以看出，SMDP 对部分课题的性能提升比较明显，如 vortex 提升 4.9225%，fma3d 提升3.0986%，sixtrack 提升 2.2021%；SMDP 对部分 load/store 冲突不密集的课题的性能提升作用相对有限，但未降低课题的运行性能。

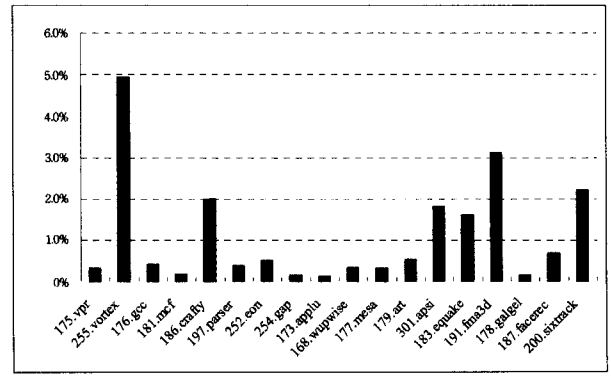

图 4 spec2000 各课题性能提升比例（4096）

如图 5 所示，随着 DPT 条目数量的增加，性能提升的比例越来越多。当条目数量从 128 增加到 4096 时，最大性能提升比例从 2.1506%到 4.9225%，平均性能提升从 0.3564%到 0.7991%。从图 5 可以看出，条目增加到一定数量后，性能提升趋于平缓；后续条目数量的增加，成倍地增大了硬件开销，但对性能的提升越来越有限。因此下面将通过实验评估性能提升与硬件开销之间的关系，帮助设计人员在实际中选择合

（下转第 54 页）

表 3　矩阵乘算法程序 IPC 分析结果

次数	单线程(ns)	多线程(ns)	Δ	百分比↑(%)
1	26710.0	16216.0	10494.0	39.29
2	728.0	668.0	60.0	8.24
3	728.0	668.0	60.0	8.24

从表 3 可以得出，当程序派发执行稳定后，在多线程模式下的派发执行效率比单线程模式下提高了 8.24%，多线程模式下 IPC 的性能提升了 8.24%。

结束语　本文详细介绍了多线程模式下循环指令缓冲的读写机制以及单线程与多线程的切换机制。使用 NC Verilog 对 RTL 级代码进行了系统级仿真验证，用 65nm 标准单元库对 RTL 级代码进行了 DC 综合，在保证了功能正确性的同时，综合后的运行频率可达 714.286MHz，通过程序验证了在多线程模式下，处理器在性能上得到了提升。

参 考 文 献

[1] Merten M C, et al. Modulo Scheduling Buffer[C]// Proceeding of the 34th annual ACM international symposium on Microarchitecture. Texas, Dec. 2001

[2] Hennessy J L, Patterson D A. Computer Architecture: A Quantitative approach(4th Edition)[M]. Elsevier, 2007: 274-363

[3] Ramakrishna B, Schlansker M S, Tirumalai P. Code Generation Schema for Modulo Scheduled Loops[C]// Proceeding of MICRO-25. 1992: 297-326

[4] Grunewald W, Ungerer T. Towards Extremely Fast Context Switching in a Block Multithreaded Processor[C]// Proceedings of the 22nd EUROMICRO Conference. September 1996

[5] Gwennap L. MAJC Gives VLIW a New Twist[J]. Microprocessor Report, 1999, 13(12)

[6] Aa T V, Jayapala M, Barat F, et al. Instruction Buffer Exploration for Low Energy VLIWs with Instruction Clusters[C]// Proceedings of the 2004 conference on Asia South Pacific design automation. Yokohama, Japan, Jan. 2004: 163-247

[7] Tullsen T D, Eggers S, Emer J, et al. Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor[C]// Proceedings of the 23rd Annual International Symposium on Computer Architecture. Philadelphia, May 1996

(上接第 40 页)

适的条目数量。

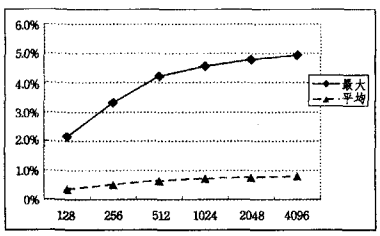

图 5　不同条目数量下性能变化趋势

以 4096 条目数量的性能提升作为参考值，将其他数量的提升情况与 4096 情况进行对比。图 6 详细给出了部分 spec2000 课题在不同 DPT 条目数量下的性能提升对比情况。当条目数量为 1024 时，性能提升约为 4096 时的 80%；当条目数量为 512 时，降为 4096 的 60%左右。4096 情况比 1024 硬件开销增加 4 倍，但性能提升只增加 20%。综合考虑，在实际使用中，1024 条目数量是一种合适的选择。

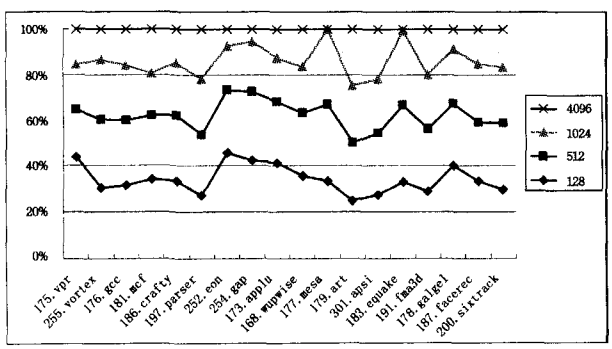

图 6　spec2000 各课题在不同条目数量下的性能提升对比

结束语　本文设计了一种简单、高效的存储相关性预测器 SMDP。SMDP 易于实现，硬件开销小；能够根据程序运行动态情况自适应更新预测信息表；能够充分利用指令级并行性，使得顺序发射的 load 指令在保证正确率的前提下，能尽早乱序发射。本文详细介绍了 SMDP 的预测及处理过程、预测信息位更新策略，并分析了 SMDP 的实现难度与开销。实验表明，SMDP 能够有效提升处理器性能，在贴近实际处理器参数配置情况下，平均性能提升达到 0.7991%，最大可达 4.9225%。

参 考 文 献

[1] Doweck J. Inside Intel® Core™ Microarchitecture and Smart Memory Access[M]. Intel White Paper, 2006

[2] Moshovos A, Breach S E, Vijaykumar T N, et al. Dynamic Speculation and Synchronization of Data Dependences[C]// the Proceedings of the 24th Annual International Symposium on Computer Architecture (ISCA). 1997

[3] Chrysos G Z, Emer J S. Memory dependence predictor using store-sets[C]// Proceedings of the 25th Annual International Symposium on Computer Architecture (ISCA). 1998

[4] Loh S S G H. Store Vectors for Scalable Memory Dependence Prediction and Scheduling[C]// the Proceedings of the 12th International Symposium on High-Performance Computer Architecture(HPCA). 2006

[5] Onder S, Gupta R. Dynamic memory disambiguation in the presence of out-of-order store issuing[C]// Proceedings of the 32nd Annual ACM/IEEE International Symposium on Microarchitecture (MICRO). 1999

[6] Yoaz A, Erez M, Ronen R, et al. Speculation Techniques for Improving Load Related Instruction Scheduling[C]// Proceedings of the 26th Annual International Symposium on Computer Architecture (ISCA). 1998

[7] Kessler R E. The Alpha 21264 Microprocessor Architecture[C]// IEEE MICRO. 1999