

一种面向非规则数据的差别预取策略

朱会东, 段赵磊

(郑州轻工业学院 计算机与通信工程学院, 河南 郑州 450002)

摘 要: 提出一种面向非规则数据的差别预取策略, 研究数据访问的局部性特征和预取调度机制. 差别预取策略通过把访存指令相邻两次访存之间的距离加入到访存流模型中, 实现依据访存指令的访存行为动态调整预取操作. 实验结果表明, 与传统的基于访存流模型的数据预取技术相比较, 差别预取策略能够提高缓存命中率和更加有效地改善非规则数据应用程序性能.

关键词: 连续局部性; 访存流; 非规则数据; 数据预取

中图分类号: TP393

文献标识码: A

文章编号: 1000—7180(2013)06—0034—04

Differentiating Prefetching for Irregular Data References

ZHU Hui-dong, DUAN Zhao-lei

(College of Computer and Communication Engineering, Zhengzhou University of Light Industry,
Zhengzhou 450002, China)

Abstract: Anew differentiating prefetching technique for irregular data is presented based on memory access stream. Differentiating prefetching mechanism records the distance between two successive memory accesses of the same instruction when modeling the memory access stream of hot loops, and adjusts prefetch operations accordingly. Experimental results show that differentiating prefetching mechanism can improve cache hit rate and system performance comparing with conventional data prefetching techniques.

Key words: continuous locality; memory access stream; irregular data; data prefetching

1 引言

处理器和存储器之间的速度鸿沟导致的存储墙问题一直是制约处理器性能发挥的重要因素. 由于存储墙问题的存在, 导致 CPU 的运算能力因为等待从存储器存取数据而大量虚耗, 这使得现代计算机系统中强劲的处理性能难以得到充分发挥.

数据预取技术是解决存储墙问题的重要研究领域之一. 基于访存流模型的数据预取技术作为传统数据预取技术的扩展方案, 是隐藏非规则数据访存延迟的常用实现方案. 当前国内外基于访存流模型的数据预取技术的研究才刚刚起步, 还存在很多不足之处: 对存储空间划分区域后再构造时空局部性模型, 容易把访存地址规律而间距大的访存行为划分为很多不同的访存区域, 增加了模式及其识别的

复杂度; 对导致 Cache 访问失效的指令等同看待, 只是部分地重构了访存流; 对程序的所有 Cache 访问失效数据进行分析, 一方面浪费了空间资源, 另一方面有些随机的 Cache 访问失效会对主要关注数据造成不必要的干扰. 因此, 如何根据非规则计算的特性, 基于访存流构建面向复杂数据结构的通用的、高效的预取模型, 是本文研究的重点^[1].

2 数据访问的局部性特征

高速缓存 Cache 和数据预取技术之所以起到隐藏访存延迟的作用是因为程序运行时对存储器数据的访问具有局部性特征, 局部用到的数据存放在缓存中, 提高了缓存命中率^[2-3].

定义 1 时间局部性 (Temporal Locality). 指程序在运行时刚刚被访问过的数据很快又被访问

到,当程序需要访问的数据不在 Cache 中时,就把相应的内容从内存取到 Cache 行中,这样在下次访问它时就不会发生缓存失效了^[4-5]。

定义 2 空间局部性(Spatial Locality). 指在访问内存中一块数据后的极短时间段内,内存中与其邻接的一块内容也会被访问到. 如果在访问内存中的一个数据块时,顺带把与其邻接的数据块也取到 Cache 中^[6-7],这样在不久访问邻接数据块时就不会发生缓存失效了。

定义 3 连续局部性(Continuous Locality). 时间局部性和空间局部性都是指运行程序在极短时间段内的数据访问局部性特征,因此,又称它们为连续局部性。

定义 4 非连续局部性(Discontinuous Locality). 指程序在运行时,由于函数被重复调用,存在被重复访问的数据访问序列. 如果在该数据访问序列被重复访问之前,利用数据预取技术把该数据访问序列要访问的数据提前取到 Cache 中,这样在该数据访问序列被重复访问时就不会发生缓存失效了^[8]。

图 1 中显示的数据访问局部性特征示例说明了时间局部性、空间局部性和非连续局部性的区别. 图 1 中, A_n 表示的是访存地址,左边的地址序列 $Seq1 = \{A1, A2, A7, A5, A3, A1, A2, A4, A6, A8\}$ 是程序运行过程中某一时段的顺序访存流,右边的地址序列 $\{A1, A2, A3, A4\}$, $\{A5, A6\}$ 和 $\{A7, A8\}$ 是该时间段内访问数据的几个连续存储空间. 根据定义 1, $A1, A2$ 是很短时间内被重复访问的数据地址,如果第一次访问 $A1, A2$ 时发生了 Cache 失效,需要从内存中读取 $A1, A2$ 的数据,则当下一次访问 $A1, A2$ 的数据时,就可以直接从 Cache 而不需访问内存了. 因此,程序对数据地址 $A1, A2$ 的访问具有时间局部性特征. 同样地,根据定义 2, $\{A1, A2, A3, A4\}$, $\{A5, A6\}$ 和 $\{A7, A8\}$ 在内存中属于邻接的数据块,如果在访问其中一个数据块时(如 $A7$)发生了 Cache 失效,需要从内存中读取数据(如 $A7$),则当不久访问邻接数据块(如 $A8$)时就不会发生缓存失效了. 因此,程序对地址序列 $\{A1, A2, A3, A4\}$, $\{A5, A6\}$ 和 $\{A7, A8\}$ 的访问具有空间局部性特征. 最后,假如数据访问地址序列 $Seq1$ 在程序运行过程中多次重复出现,当第一次访问 $Seq1$ 时暂时把这个序列保存起来,则第二次访问 $Seq1$ 时,利用数据预取技术提前把 $Seq1$ 中的数据取回到 Cache 中,这样就避免了第二次访问 $Seq1$ 的 Cache 失效。

因此,程序对地址序列 $Seq1$ 的访问具有非连续局部性特征。

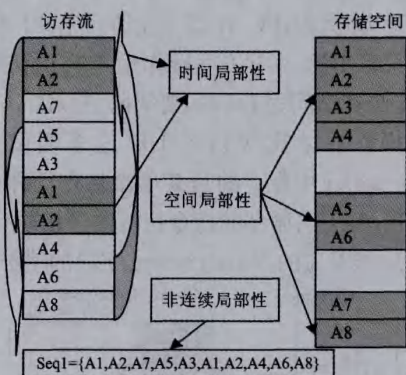


图 1 数据访问的局部性特征

3 面向非规则数据的差别预取策略

基于访存流模型的数据预取技术是隐藏非规则数据访存延迟的有效措施. 然而,已有的基于访存流模型的数据预取技术不是忽略了访存流中的时间顺序^[9],就是忽略了各访存指令访存行为间的差别^[8],造成由过早预取行为引起的缓存污染或由过晚预取行为引起的无效预取. 因此,建立能反映时间顺序和行为差别的访存流模型对数据预取的有效性来说至关重要。

为了保证差别预取策略的有效性,节约系统资源,本文选择位于程序关键执行路径上的,且其频繁缓存失效行为严重影响程序性能的热点循环作为差别预取对象. 为了选择合适的差别预取对象,本文在应用差别预取策略前先使用动态剖析工具分析测试程序,并在频繁发生缓存失效行为的热点循环入口处做标记,当程序运行到标记处时启动差别预取策略。

3.1 差别预取的访存流模型

本文采用与文献^[8]中类似的索引表 Index Table 和全局历史表 Global History Buffer 存储程序运行过程中热点循环的数据访问信息,如图 2 所示. 其中, A_n, B_n, C_n 和 D_n 分别表示与四条访存指令 PC_A, PC_B, PC_C 和 PC_D 对应的访存信息. 由于程序热点循环中访存指令的执行顺序相似,如 PC_A, PC_B, PC_C 和 PC_A, PC_B, PC_D , 因此,与文献^[8]中的 Index Table 不同,本文的 Index Table 表中直接按热点循环中访存指令的执行顺序保存各指令的访存流信息,而无须额外的字段保存访存指令的执行顺序. 另外,为了根据其访存行为特性区别对待各访存指令的预取操作,本文在文献^[8]中的

Global History Buffer 基础上又增加了一个 NextInterval 字段,用来保存各访存指令相邻两次访存之间的距离。例如,PC_A 指令在访问完 A1 后,要间隔 B1,C1,D1 这 3 个访存操作后才再次访问 A2;PC_D 指令在访问完 D3 后,要间隔 A1,B1,C1,这 3 个访存操作后才再次访问 D1(这里假设 Global History Buffer 中保存的是某单次热点循环的数据访问信息,且下次循环的指令执行顺序相同),因此,A1 和 D3 相对应的 NextInterval 字段的值都是 3。

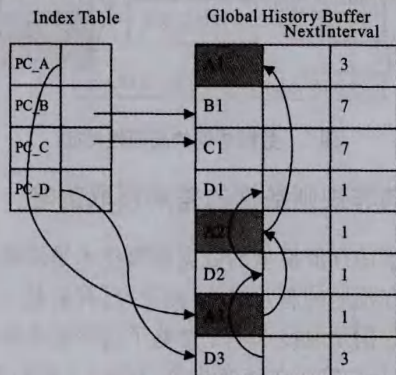


图2 差别预取的访存流模型

Index Table 表和 Global History Buffer 表的更新机制如下:①当程序执行到热点循环入口处时,初始化 Index Table 表和 Global History Buffer 表,即按执行顺序把第一次热点循环的访存指令保存到 Index Table 表中,按数据访问顺序把相应的数据地址保存到 Global History Buffer 表中,并依次更新 NextInterval 字段的值。②当单次热点循环的访存指令顺序没有发生改变时,只需更新 Global History Buffer 表中的地址信息。③当单次热点循环的访存指令顺序改变但没有增加新的访存指令时,更新 Global History Buffer 表中的地址信息和 NextInterval 字段值,并修改 Index Table 表中相应指针的值。④当单次热点循环的访存指令顺序改变且增加新的访存指令时,重新初始化 Index Table 表和 Global History Buffer 表。

3.2 差别预取操作

差别预取操作的执行过程如下:

(1)根据当前访存指令索引 Index Table 表,找到下一条访存指令和该指令的上一个访存地址;

(2)根据 Global History Buffer 表追溯下一条访存指令的访存地址信息,确定该访存指令的预取度,并按 NextInterval 字段值确定预取间隔;

(3)把下一条访存指令作为当前指令,重复步骤(1)和(2),直至到达 Index Table 表末尾。

例如在图2中,当前访存指令是 PC_D,当前访存地址是 D3,则首先索引 Index Table 表找到下一条访存指令是 PC_A 和它的上一个访存地址 A3;接着,根据 Global History Buffer 表追溯 PC_A 的访存地址信息 A1—A2—A3,确定 PC_A 的预取度为 3,并按 NextInterval 字段值确定 A1—A2—A3 之间的预取间隔分别为 3 和 1;然后,按同样的方式处理 PC_B 和 PC_C,确定 PC_B 和 PC_C 的预取度为 1,不需考虑预取间隔,但因为上一条指令 PC_A 中 A1—A2 的预取间隔为 3,所以 PC_A,PC_B 和 PC_C 处理后预取操作序列为 {A1,B1,C2,A2,A3},且 C2—A2 与 A2—A3 之间分别还可插入一个预取操作;最后,同样道理处理 PC_D,得到最后的预取操作序列为 {A1,B1,C2,D1,A2,D2,A3}。这里 A1,A2,B1,B2,D1,D2 的值泛指各访存指令在单次热点循环中不同的访存信息,具体值可根据情况确定,如使用步长、关联信息等计算得到的访存地址。

从差别预取操作的执行过程可以看出,差别预取策略不仅可以根非规则计算的特性,把非连续局部性转换为连续局部性来提高缓存命中率,而且还根据各访存指令相邻两次访存之间的距离调整预取度,达到及时预取和消除无用预取的目的。

4 性能评价

为了本文提出的差别预取策略的有效性,我们对测试程序的热点循环模拟实现了差别预取策略和文献[8]提出的 Stream Chaining 预取策略。实验中差别预取策略 Global History Buffer 表中的地址信息为关联地址信息。Stream Chaining 预取策略中 Ctr 字段的阈值设置为 3(即相邻访存指令重复出现超过 3 次时,认为这两个指令是一对强连接,并把此强连接加入到 Miss Graph 中),预取度设为 2。

本文实验采用的是 SimpleScalar 3.0 模拟器,模拟器的配置如表1所示。Index Table 表的大小是 2KB,在差别预取策略中可以存储 256 条记录,在 Stream Chaining 预取策略中可以存储 128 条记录;Global History Buffer 表的大小是 8 kB,在差别预取策略中可以存储 256 条记录,在 Stream Chaining 预取策略中可以存储 512 条记录。

本文根据在线剖析工具对 Olden 基准测试程序的分析结果,从中选择热点循环所消耗的 CPU 时钟周期数占总时钟周期数 20% 以上的 5 个非规则数据密集应用程序作为测试程序。这 5 个程序分别是:em3d, mst, tsp, gcc 和 health。本文中所有的应

用程序都使用 GCC 4.1 编译器进行编译,编译选项为“-O2”.实验结果如图 3 和图 4 所示.

表 1 模拟器配置

Branch Predictor	Tournament Predictor 16K entry tables
Issue Width	4 instructions
RUU Size	64 entries
Load/Store Queue	32 entries
Level 1 D-Cache	8 kB,2-way, 1 cycle latency, 32 B lines
Level 1 I-Cache	16 kB,direct mapped, 1 cycle latency, 32 B lines
Leve2 Cache	512 kB,4-way, 14 cycle latency, 64 B lines
Index Table	2 kB
Global History Buffer	8 kB

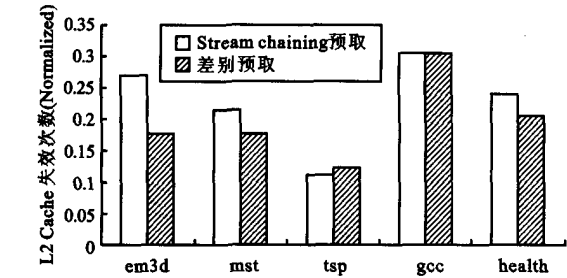


图 3 L2 Cache 失效次数变化

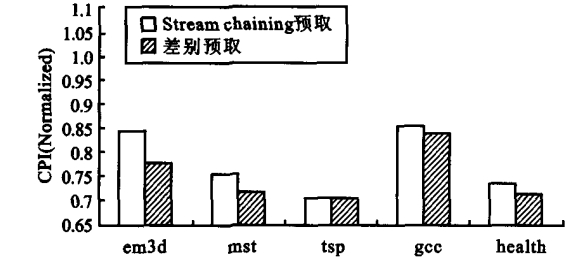


图 4 CPI 变化

图 3 中显示了相对于源程序,应用差别预取和 Stream Chaining 预取后测试程序 L2 Cache 失效次数的规范化数值.从图 3 中可以看出,差别预取消除了各测试程序热点循环中 80% 以上的 L2 Cache 失效行为,而 Stream Chaining 预取消除了各测试程序热点循环中 70% 以上的 L2 Cache 失效行为.除 TSP 和 GCC 外,其它程序在应用差别预取后的 L2 Cache 失效次数都比在应用 Stream Chaining 预取后时少.例如,MST 在应用差别预取后,其 L2 Cache 失效次数减少了约 82.2%,而在应用 Stream Chaining 预取后,只减少了约 79.7%.这主要是由两个原因造成的.首先,差别预取策略根据访存指令的行为差别采取不同预取度,减少了无用预取,提高

了预取及时性和缓存命中率.其次,差别预取根据访存指令相邻两次访存之间的距离确定预取间隔,避免了过早预取引起的无用预取和缓存污染,进一步提高了缓存命中率.

图 4 中显示了相对于源程序,应用差别预取和 Stream Chaining 预取后测试程序的 CPI 规范化数值.从图 4 中可以看出,在应用差别预取和 Stream Chaining 预取后,测试程序的 CPI 都有所降低,差别预取比 Stream Chaining 预取取得了更加明显的效果.尤其对于源热点循环所消耗的 CPU 时钟周期数占总时钟周期数 70% 以上的 MST, TSP 和 HEALTH 这 3 个测试程序来说,应用 Stream Chaining 预取后,其 CPI 降低了 25% 以上;而应用差别预取后,其 CPI 降低了 28% 以上.这说明差别预取相对于 Stream Chaining 预取,能更加有效地改善程序性能.

5 结束语

本文实现了一种面向非规则数据的差别预取策略,以应用程序热点循环的访存流模型为基础,研究了访存指令的访存行为差别和预取操作的动态调整机制.差别预取根据访存指令的访存行为特征确定预取度和预取操作间隔,提高了预取及时性和缓存命中率.实验结果表明,与传统的基于访存流模型的数据预取技术相比,根据访存指令的访存行为差别动态调整数据预取操作能够更加有效地隐藏非规则数据的访存延迟,改善应用程序性能.

参考文献:

[1] 邱保志,陈本华,张真.一种新的快速混合聚类算法[J].微电子学与计算机,2008,25(7):78-80.

[2] Chi-Keung Luk, Todd C Mowry. Compiler-based prefetching for recursive data structures[C]//Proceedings of the Seventh International Conference on Architectural Support for Programming Languages and Operating Systems. Association for Computing Machinery. Cambridge,Massachusetts, 1996: 222-233.

[3] Chi-Keung Luk, Mowry T C. Automatic compiler-inserted prefetching for pointer applications [J]. IEEE Transactions on Computers, 1999, 48(2): 134-141.

[4] Jamison D Collins, D M T, Hong Wang, et al. Dynamic speculative precomputation[C]// 34th International Symposium on Microarchitecture. IEEE Comput. Soc. USA: Washington. 2001: 306-317.

由表 4 和表 5 可以看出:在面向大规模页面信息时,传统方法 SVM 和 SH 基本上无法求解(除 S₆ 外),而 NACM 能快速得到分类结果.由此可见,NACM 在解决大规模页面分类问题上的有效性是传统方法所不具备的.而且 NACM 在一定程度上对用户信息进行了隐私保护.

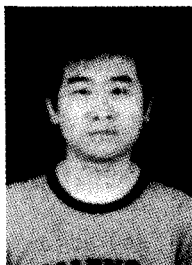
5 结束语

如何从用户访问的页面信息对用户进行归类是进行信息主动推送的关键一步.本文在分析当前主流分类方法 SVM 的基础上,提出面向大规模数据的用户分类方法 UCM.该方法不仅解决了大规模用户分类问题,还保证分类过程中用户信息安全.上述优势使得 UCM 非常适合在互联网环境下对海量用户信息分类,也适用于其他海量数据分类问题.

参考文献:

- [1] 徐谦.网络信息检索的智能化趋势[J].图书馆理论与实践,2006(2):63—65.
- [2] 曹红兵.搜索引擎的个性化检索研究[J].图书情报工作,2007,51(3):139—136.
- [3] Pal M, Foody G M. Feature selection for classification of hyper spectral data by SVM [J]. IEEE Trans. on Geoscience and Remote Sensing, 2010, 48 (5): 2297—2307.
- [4] 刘红泉,张亮峰.布尔逻辑检索模型的分析探讨[J].现代情报,2004(9):4—6.
- [5] 唐明伟,卞艺杰,陶飞飞.基于语义向量空间模型的文档检索系统研究[J].情报杂志,2010(5):167—170.
- [6] 刘华.文本分类相似度模型和概率模型的实现与比较[J].现代图书情报技术,2006(4):53—55.
- [7] 许林杰.中文文本分词研究[D].山东:山东师范大学,2003.
- [8] 顾立帆,王永成.联想树分析法及其在无词库中文自动标引中的应用[J].情报学报,1992,11(5):354—360.
- [9] Marcin O. New separating hyperplane method with application to the optimization of direct marketing campaigns[J]. Pattern Recognition Letters, 2011(32): 540—545.

作者简介:

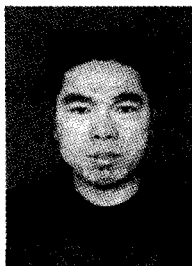


刘忠宝 男,博士.研究方向为机器学习与信息检索.

(上接第 37 页)

- [5] Steve S Liao, P H W, Hong Wang, et al. Post-pass binary adaptation for software-based speculative pre-computation[C]//Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2002). Germany: Berlin, Association for Computing Machinery, 2002: 117—128.
- [6] Grunwald D, D J a. Prefetching using markov predictors [J]. IEEE Transactions on Computers, 1999, 48 (2): 121—133.
- [7] Nesbit KJ, Smith J E. Data cache prefetching using a global history buffer [J]. IEEE Micro, 2005, 25(1): 90—97.
- [8] Diaz P, Cintra M. Stream chaining: exploiting multiple levels of correlation in data prefetching[C]//Proceedings of the 36th International Symposium on Computer Architecture. USA: Austion, Texas, IEEE Comput. Soc, 2009: 81—92.
- [9] Stephen Somogyi, T F W, Anastassia Ailamaki, et al. Spatial memory streaming [C] // Proceedings of the 33rd International Symposium on Computer Architecture. USA: Boston, IEEE Comput. Soc, 2006: 252—263.

作者简介:



朱会东 (1975—)硕士,讲师.研究方向为高性能计算、缓存性能优化.

段赵磊 男,博士,讲师.