基于缓存行为特征的线程数据预取距离控制策略

黄 艳*① 张启坤② 段赵磊② 古志民③
①(郑州轻工业学院软件学院 郑州 450002)
②(郑州轻工业学院计算机与通信工程学院 郑州 450002)
③(北京理工大学计算机学院 北京 100081)

摘 要：针对目前大多数面向指针应用程序的线程数据预取方法在预取距离控制方面的不足，该文提出一种基于缓存行为特征的数据预取距离控制策略。该策略利用指针应用程序执行时的数据缓存特征构建预取距离控制模型，以避免共享缓存污染，降低系统资源竞争，并通过忽略对部分非循环依赖数据预取平衡帮助线程与主线程间的执行任务，提高线程数据预取的时效性。实验结果表明，通过该策略控制线程数据预取距离能进一步提高线程预取性能。

关键词：片上多处理器；线程预取；帮助线程；预取率；预取距离
中图分类号： TP302 **文献标识码**： A **文章编号**：1009-5896(2015)07-1633-06
DOI: 10.11999/JEIT141429

Prefetch Distance Control Strategy Based on Cache Behavior in Threaded Prefetching

Huang Yan① Zhang Qi-kun② Duan Zhao-lei② Gu Zhi-min③
①(*Department of Software Engineering, Zhengzhou University of Light Industry, Zhengzhou* 450002, *China*)
②(*College of Computer and Communication Engineering, Zhengzhou University of Light Industry, Zhengzhou* 450002, *China*)
③(*Department of Computer Science and Technology, Beijing Institute of Technology, Beijing* 100081, *China*)

Abstract: Due to the deficiencies in prefetch distance controlling of most threaded data prefetching methods for pointer application, a prefetch distance control strategy based on the cache behavior characteristics is proposed. In this paper, the prefetch distance control model is constructed using the runtime data cache features of pointer applications to reduce cache pollution and system resources contention. By skipping loop-carried independencies data accesses, the task between main thread and helper thread is balanced and the timeliness of threaded prefetching is improved. The experimental results show that the proposed approach can optimize the performance of threaded prefetching mechanism.

Key words: Chip Multi-Processors (CMP); Threaded prefetching; Helper thread; Prefetch ratio; Prefetch distance

1 引言

随着片上多核处理器技术的快速发展，阻碍多核系统性能提升的“存储墙”问题进一步突显。目前，线程数据预取技术是隐藏存储器访问延迟、提高多核系统性能的有效手段。

预取距离指数据预取技术中发出预取请求到预取数据被使用之间的执行间隔。预取距离的大小直接决定了预取的有效性。一方面，为了让预取数据在被使用前到达以便完全隐藏该数据的缓存失效延迟，预取距离应该足够大[1]；另一方面，为了避免预取数据到达的太早造成缓存污染，预取距离又不能过大[2]。

近年来，不少研究者[3–16]利用帮助线程数据预取隐藏应用程序中长延迟取数指令的访存延迟，其中多数采用特殊的线程间同步机制来控制预取距离。文献[3]在程序执行过程中使用特殊的硬件预测表预测最优预取距离；文献[4]通过关联表评估连续两次缓存失效事件的时间间隔，在一定预取距离范围内尽早发出预取请求；文献[5]通过编译器分析设置特定的采样间隔，主线程在每个采样间隔触发一个帮助线程；文献[6]通过预解码(decode)检测到长延迟取数指令时触发帮助线程；文献[7]中帮助线程通过一个循环计数器与主线程进行同步；文献[8]使用信号量同步机制控制帮助线程的预取距离；文献[9]通过增量改变预取距离的值找到较优预取距离；文献[10]在检测到规律的只读缺失模式时进行数据

2014-11-13 收到，2015-03-17 改回，2015-06-01网络优先出版
国家自然科学基金(61370062)，郑州市科技攻关计划项目(20130725)和博士基金项目(2013BSJJ050)资助课题
*通信作者：黄艳 shelly_35@126.com

预取。当面向非规则数据密集应用时，由于非规则数据访问间的相互依赖关系，这些方法难以实现对预取距离的有效控制，预取时效性难以得到保证。

本文提出一种基于缓存行为特征的有效预取距离控制策略。该方法的核心思想是：首先分析和预测热点循环数据访问的缓存行为特征；然后，根据应用程序热点循环的缓存行为特征选取最大有效预取距离；最后，在预测的有效预取距离取值范围内通过实验测试选择较优预取距离取值。通过该策略控制线程数据预取距离能进一步提高线程预取性能。

2 面向共享缓存的间隔预取策略(Skip-Prefetching, SP)

指针应用程序中的热点数据通常可以分为循环依赖数据和非循环依赖数据。在 SP 线程预取机制中，帮助线程通过忽略对部分非循环依赖数据的预取控制线程预取距离和平衡主线程与帮助线程。SP 的基本思想：如果在帮助线程与主线程的并行执行过程中，帮助线程的预取操作能与主线程的访存操作或计算操作完全并行地执行，互不干扰，且帮助线程预取的数据都恰好被主线程需要，则主线程将获得最大的性能收益。理论上，当帮助线程的执行延迟占热点循环执行总延迟的一半时，帮助线程与主线程达到最大程度的并行执行，主线程的性能得到最大限度地提高。图 1 抽象地说明了这种理想状况下的程序执行流程。依据图 1 (a)所示，源程序热点循环的单次循环延迟为($T_{\mathrm{FM}}+T_{\mathrm{SM}}+T_C$)。根据图 1(b)，应用线程预取机制后，通过合理地设置 A_SKI 与 A_PRE 的值(A_SKI 即是预取距离)，帮助线程的性能收益最大化时主程序热点循环的单次循环延迟减少至原来的一半，即($T_{\mathrm{FM}}+T_{\mathrm{SM}}+T_C$)/2。

我们已有的研究结果表明，可以根据 T_{FM}，T_{SM} 和 T_C 三者间的关系确定 A_SKI 与 A_PRE 的关系。本文的研究工作即是在 A_SKI 与 A_PRE 关系确定的基础上，通过分析应用程序热点数据的缓存行为特征，确定预取距离 A_SKI 的有效取值范围并通过实验选择较优的预取距离取值。

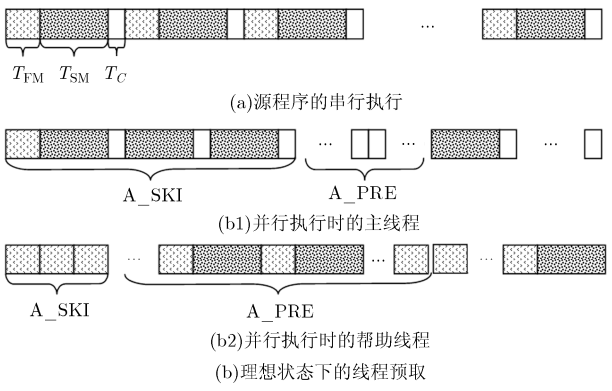

图 1 理想状态下的线程预取

图 1 中，T_{FM} 对应于单次热点循环中循环依赖数据的平均访存延迟；T_{SM} 对应于单次热点循环中非循环依赖数据的平均访存延迟；T_C 对应于单次热点循环中的平均计算延迟；A_SKI(预取距离)指帮助线程在执行热点循环时，不预取非循环依赖数据时连续执行的循环次数；A_PRE(预取长度)指帮助线程在执行热点循环时，预取非循环依赖数据时连续执行的热点循环次数。

3 基于缓存行为特征的线程预取距离控制策略

3.1 缓存行为特征分析

数据预取引发缓存污染的可能性与应用程序的访存特征及共享缓存的映射机制密切相关。对于片上多处理器中普遍使用的多路组相联共享缓存来说，执行帮助线程数据预取时，如果一个应用程序的热点访问数据都集中映射到共享缓存的同一个缓存组中，则很容易引起缓存污染；反之，如果一个应用程序的热点循环访问数据极少映射到共享缓存的相同缓存组中，则不容易引起缓存污染。因此，通过分析应用程序热点数据访问流，识别出当映射到同一个共享缓存组中的缓存行超过共享缓存的组相联度时的热点循环次数，可以预测在执行数据预取时热点循环何时会发生缓存污染。

这里把识别出的此特征定义为热点循环对应于共享缓存组的缓存相关度，并分析此特征与 SP 预取策略中预取距离 A_SKI 取值的关系。特别地，对本文中选择的实验平台 Intel Core 2 Quad 6600 处理器，其共享缓存的组相联度是 16。

定义 1 缓存相关度 SA(L, Sx)： 假设对于 CMP 结构中共享缓存(最低级)的组相联度是 C, Sx 是该共享缓存中的某一个缓存组，包含 C 个缓存块 $(B_1,B_2,\cdots,B_C)$。对于访存密集型程序中的热点循环 L，其循环次数是 I。共享缓存采用的是 LRU 替换算法。假设在 L 执行到第 $j(j<I)$次循环时，恰好 L 中有 C 个数据块映射到 Sx 中的 C 个缓存块中。这里定义 j 为 L 对应于 Sx 的缓存相关度，记为 SA(L, Sx)= j。图 2 进一步解释了缓存相关度的定义。

热点循环对应于共享缓存组 Sx 的缓存相关度表明热点循环访问数据间的缓存相关性。例如，对于某个热点循环，其对应于共享缓存中缓存组 Sx 的

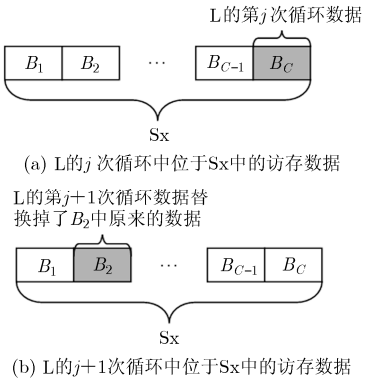

图 2 缓存相关度SA(L, Sx)=j 时的情况

缓存相关度为 40，表明在应用程序独立运行时，当热点循环执行到第 40 次循环时，共享缓存组 Sx 中的热点循环访问数据块已经达到饱和。当再有热点循环访问数据块到达此缓存组时，将要进行缓存替换。

定理 1　假设热点循环 L 对应于共享缓存组 Sx, SA(L, Sx)=j。应用 SP 预取机制后，当 A_SKI + A_PRE >j 时，帮助线程造成共享缓存污染的可能性增大。

证明　设 A_SKI+ A_PRE=R，图 3(a)说明了 A_SKI+A_PRE≤ j 时 Sx 的数据驻入情况。如图 3(a)所示，A_SKI+A_PRE≤ j 时，在主线程执行 A_SKI 次循环，同时帮助线程执行 A_PRE 次循环后，L 的访问数据只占据 Sx 缓存组中 C 个缓存块的一部分。当热点循环 L 的访问数据占用了 Sx 缓存组中的全部 C 个缓存块时，主线程早已执行过第 R 次循环，此时帮助线程预取的数据也已被主线程访问过，因此，除非数据被重复访问，否则，新驻入的数据替换掉的都是无用数据，不存在缓存污染。

图 3(b)分析了 A_SKI + A_PRE>j 时 Sx 的数据驻入情况。如图 3(b)所示，在主线程执行 N(A_SKI<N)次循环，同时帮助线程执行第 M(A_SKI<M<R)次循环时，热点循环 L 的访问数据已经占用了 Sx 缓存组中全部的 C 个缓存块。所以当 L 继续执行时，主线程将执行第 N+1 次循环，而帮助线程将执行第 M+1 次循环。此时主线程或帮助线程新驻入的访问数据将替换掉 Sx 缓存组中存储的数据，如果替换掉的数据是帮助线程最近驻入的访问数据，因这些数据还没有被主线程访问，从而造成缓存组 Sx 被污染。因此，当 A_SKI+A_PRE>j 时帮助线程造成共享缓存污染的可能性增大。　　证毕

定理 2　假设热点循环 L 对应于共享缓存组 Sx, SA(L,Sx)=j。应用 SP 预取机制时，当预取距离 A_SKI<j/2 时，帮助线程造成共享缓存污染的可能性比 A_SKI >j/2 小。

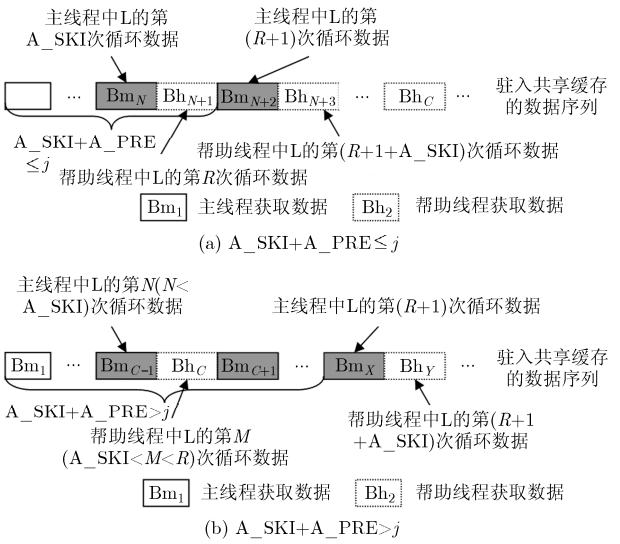

图 3 基于缓存相关度和预取距离的缓存污染分析

证明　(1)当 A_SKI≤ A_PRE 时，因 A_SKI ≤ A_PRE ，如果 A_SKI>j/2 ，则 A_SKI+ A_PRE>j。根据图 3 和定理 1 的证明过程，应用 SP 预取机制后，A_SKI+A_PRE>j 时，帮助线程造成共享缓存污染的可能性增大。因此，A_SKI ≤ A_PRE 时，预取距离 A_SKI<j/2 时，帮助线程造成共享缓存污染的可能性比 A_SKI >j/2 小。

(2)当 A_SKI>A_PRE 时，因 A_SKI>A_PRE，如果 A_SKI <j/2，则 A_SKI + A_PRE <j。根据图 3 和定理 1 的证明过程，应用 SP 预取机制后，A_SKI+A_PRE <j 时，帮助线程造成共享缓存污染的可能性比 A_SKI+A_PRE>j 时小。因此，A_SKI>A_PRE 时，预取距离 A_SKI<j/2 时，帮助线程造成共享缓存污染的可能性比 A_SKI>j/2 小。　　证毕

3.2 基于缓存行为特征的预取距离控制模型

预取距离的度量方式有多种，例如，预取指令与对应取数指令执行时间隔的 CPU 时钟周期数、分支语句数、指令数、缓存失效事件数等都可以作为预取距离的度量标准[17]。SP 预取机制中，帮助线程在每个循环中进行预取操作时都跳过 A_SKI 次内层循环，也就意味着帮助线程提前 A_SKI 次循环发出预取请求。因此，SP 中预取距离 A_SKI 就是指从发出预取请求到预取数据被使用时间隔的热点循环次数。

3.2.1 基于缓存相关度的最大有效预取距离取值　定理 2 表明，本文在应用 SP 预取机制时，应设置预取距离 A_SKI<SA(L, Sx)/2，即 A_SKI<j/2，以

便减少帮助线程引起的缓存污染。因此，为了减少整个共享缓存被污染的可能性，SP 的预取距离 A_SKI 与热点循环 L 对应于共享缓存组 Sx 的缓存相关度应该具有关系为

$$\text{A\_SKI} < \min(\text{SA(L, Sx)}/2) \quad (1)$$

3.2.2 最小有效预取距离取值 在 SP 预取策略中为了保证数据预取的及时性，可以使用式(2)求得 SP 预取的最小有效预取距离取值[18]：

最小有效预取距离=平均访存延迟/平均迭代延迟 (2)

这里平均访存延迟指的是源测试程序中单次热点循环的平均访存延迟；而平均迭代延迟指的是应用 SP 预取机制后，单次热点循环的平均执行延迟。在 SP 预取策略中，根据图 1，平均访存延迟=T_{FM}+T_{SM}，理想情况下的平均迭代延迟=(T_{FM}+T_{SM}+T_C)/2。则

$$\text{A\_SKI} > 2(T_{\text{FM}}+T_{\text{SM}})/(T_{\text{FM}}+T_{\text{SM}}+T_C) \quad (3)$$

特别地，当 T_C 相比于(T_{FM}+T_{SM})可以忽略不计时，有效预取距离 A_SKI 最小为 2 次热点循环；当 T_C 大于(T_{FM}+T_{SM})，有效预取距离 A_SKI 最小为 0 次热点循环。

式(1)与式(3)为设置 SP 预取距离 A_SKI 提供了直接依据。例如，热点循环对应于各共享缓存组的缓存相关度最小为 40，且 T_C 大于(T_{FM}+T_{SM})，为减少共享缓存污染和提高预取时效性，最大预取距离取值应该设置为 20，最小预取距离为 0 次热点循环，SP 预取距离 A_SKI 的较优取值应在 0 与 20 之间搜索。

4 实验结果与分析

为了得到 A_SKI 预取距离的较优取值，先分析应用程序热点循环对应于共享缓存组的缓存相关度分布，并在此基础上预测 A_SKI 预取的有效预取距离取值范围；然后，通过实验测试结果确定 SP 对特定测试程序的较优预取距离 A_SKI 取值；最后，为几个非规则数据密集应用实现了 SP 预取机制和传统帮助线程数据预取机制，并通过对实验结果的比较分析，评估 SP 预取策略的性能和有效性。

4.1 实验背景

本文采用 Intel Core 2 Quad 6600 处理器作为实验平台评估 SP 预取策略对非规则数据密集应用的性能影响。该处理器上有两个 L2 硬件预取装置，ACL 与 DPL[19]，每个处理器上的两个核动态共享这两个硬件预取装置。共享缓存的大小为 4 MB，组相联度是 16。

从基准测试程序包 SPEC2006 和 Olden 中选择 EM3D, MCF 和 MST 作为测试对象。所有的测试程序都使用 GCC 4.3 编译器进行编译，编译选项为“-O2”。表 1 中显示了这些测试程序的基本信息，包括输入参数、热点函数等信息。为了分析热点循环访问数据的缓存相关度，表 1 中最后一列显示了各测试程序的热点循环的循环次数。如果此项数据是个具体数值，说明对热点函数的每次调用，外层循环次数始终是该数值。如果此项数据是一个数值区间，说明对热点函数的不同次调用，外层循环次数在此区间内变化。

表1 热点循环的循环次数

基准测试程序名称	输入	热点函数名称	热点循环的循环次数估计
EM3D	4×10^5 结点，128 元数	fill_from_fields	4×10^5
MCF	参考输入	refresh_potential	[1.4×10^4, 5×10^4]
MST	1×10^4 结点	HashLookup	[1,1×10^4]

4.2 基于缓存相关度的 SP 有效预取距离取值结果

表 2 是各测试程序基于缓存相关度的 SP 有效预取距离取值结果。值得一提的是，本文中各程序的有效预取距离取值范围只是一个比较粗略的界限，旨在为我们选择较优预取距离取值提供指导作用。

4.3 SP 的有效预取距离取值分析

为了检测硬件预取对有效预取距离取值范围的影响[20]，在两种硬件预取配置下测试 EM3D, MCF 和 MST 3 个测试程序在预取距离取值范围内的性能变化。这两种硬件预取配置情况描述如下：

PF_ON：指 DPL 和 ACL 都是打开状态。PF_OFF：指 DPL 和 ACL 都是关闭状态。

图 4 中显示的是在 PF_ON 和 PF_OFF 两种配置下，相对于源程序，SP 预取程序的执行时间、主线程 L2 缓存失效次数、帮助线程 L2 缓存失效次数、主线程访存次数和帮助线程访存次数的规范化数值。

比较图 4(a)中 PF_ON 和 PF_OFF 两种配置

表2 SP预取距离参数的取值范围

基准测试程序名称	A_SKI 最小值	min(SA(L, Sx))	A_SKI 最大值
EM3D	2	40	20
MCF	2	3000	1500
MST	2	6300	3150

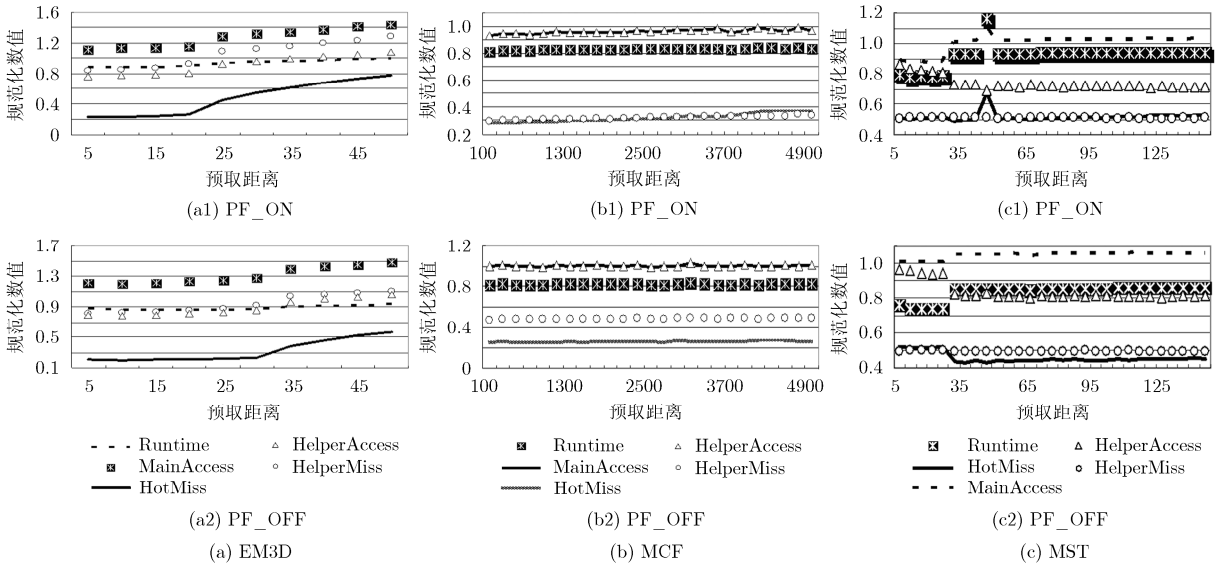

图 4　预取距离对 SP 预取的性能影响

下的结果表明，L2 硬件预取对 EM3D 的 SP 预取程序性能产生了明显影响，但总的来说，随着预取距离的不断增大，程序性能逐渐降低。这一方面说明 L2 硬件预取增加了缓存污染的严重性，另一方面说明随着预取距离的不断增大，缓存污染的严重性也不断增加。

图 4(b)中的结果显示，当预取距离参数取值逐渐变大时，MCF 的 SP 预取程序中，除 L2 缓存失效次数外，其它各性能参数的变化趋势均相同。而在 PF_OFF 配置下，各性能参数的变化均不明显。这表明当关闭 L2 硬件预取时，MCF 的 SP 预取程序性能对预取距离的变化不敏感。

图 4(c)显示的是，当预取距离参数取值逐渐变大时，MST 的 SP 预取程序性能变化情况。从图 4(c)显示的结果可以看出，在 PF_ON 和 PF_OFF 两种配置下，都是在预取距离为 25 时，MST 的 SP 预取程序性能发生了大的变化；当预取距离小于 25 时，各性能参数有缓慢减少的趋势；当预取距离大于 30 时，程序的执行时间，主线程的 L2 缓存失效次数和访存次数均有逐渐增长的趋势，而帮助线程的 L2 缓存失效次数和访存次数均有逐渐减小的趋势。

总之，根据图 4 中的显示结果，可以得到如下结论。首先，程序执行时间总是与其访存次数有相同的变化趋势，说明访存次数是衡量程序性能的一个重要指标。其次，程序执行时间与其 L2 缓存失效次数的变化趋势不总是相同，表明 L2 缓存失效次数不能作为衡量程序性能的唯一指标。最后，当预取距离高于预测范围内的某一数值时，程序的性能开始逐渐下降。这说明当预取时效性得到保证后，更大的预取距离导致更严重的缓存污染。

4.4 SP 的性能优势

为了有效分析传统线程预取的局限性，作者实现了两种传统线程预取技术。其中，提前式线程预取(Ahead Prefetching, AP)是在文献[6]中所描述的线程预取技术的基础上实现的。能动式线程数据预取方法(Active threaded Prefetching, PV)是在文献[8]中所描述的线程预取技术的基础上实现的。

图 5 和图 6 显示了 AP, PV 和 SP 对测试程序的性能影响。总体来看，SP 为这几个测试程序平均取得了约 13 %的加速比，AP 平均取得了约 4.4 %的加速比，PV 平均取得了约 5.3 %的加速比。这说明本文提出的 SP 预取策略比传统帮助线程数据预取方法的平均性能好。尤其值得注意的是，对于测试程序 MST, SP 取得了 27.1 %的性能提高，而 AP 和 PV 下却都遭受到近 2%的性能降低。图 6 中显示的规范化 CPI 数值(相对于源程序)进一步说明了 SP 预取策略相对于传统帮助线程数据预取方法的优势。因此，实验结果表明，通过该策略控制线程数据预取距离能进一步提高线程预取性能。

5　结束语

在 SP 线程预取机制基础上，通过分析应用程序热点循环执行的延迟特征和缓存行为特征评估有效的 SP 预取距离取值范围，并在该范围内通过实验测试为特定的非规则数据密集应用程序选择合适的 SP 预取距离。通过实验结果我们可以看出，合适的 SP 预取距离取值不仅提高了 SP 预取机制的预取效率，减小了帮助线程引发缓存污染的可能性，还减少了系统总线压力和共享资源竞争，进而优化了 SP 预取性能。

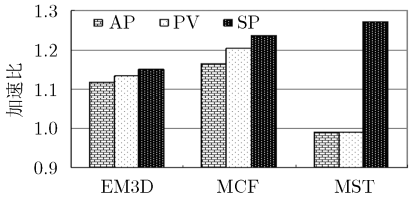

图 5 加速比

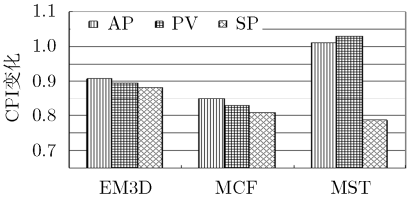

图 6 CPI变化

参 考 文 献

[1] Chen T F and Baer J L. A performance study of software and hardware data prefetching schemes[C]. Proceedings of 21st International Symposium on Computer Architecture, Chicago, USA, 1994: 223–232.

[2] Saavedra R H and Daeyeon P. Improving the effectiveness of software prefetching with adaptive execution[C]. Proceedings of Conference on Parallel Architectures and Compilation Techniques, Boston, USA, 1996: 68–78.

[3] Hur I and Lin C. Feedback mechanisms for improving probabilistic memory prefetching[C]. Proceedings of 15th International Symposium on High Performance Computer Architecture, North Carolina, USA, 2009: 443–454.

[4] Dongkeun K, Liao S S W, Wang P H, *et al*.. Physical experimentation with prefetching helper threads on Intel's hyper-threaded processors[C]. Proceedings of International Symposium on Code Generation and Optimization, California, USA, 2004: 27–38.

[5] Lu J. Design and implementation of a lightweight runtime optimization system on modern computer architectures[D]. [Ph.D. dissertation], University of Minnesota, 2006.

[6] Ro W W and Gaudiot J L. Speculative pre-execution assisted by compiler (SPEAR)[J]. *Journal of Parallel and Distributed Computing*, 2006, 66(8): 1076–1089.

[7] Somogyi S, Wenisch T F, Ailamaki A, *et al*.. Spatial-temporal memory streaming[C]. Proceedings of the 36th International Symposium on Computer Architecture, Austin, USA, 2009: 69–80.

[8] Lee J, Jung C, Lim D, *et al*.. Prefetching with helper threads for loosely coupled multiprocessor systems[J]. *IEEE Transactions on Parallel and Distributed Systems*, 2009, 20(9): 1309–1324.

[9] 单书畅, 胡瑜, 李晓维. 基于数据预取的多核处理器末级缓存优化方法[J]. 计算机辅助设计与图形学学报, 2012, 24(9): 1241–1248.
Shan Shu-chang, Hu Yu, and Li Xiao-wei. Date prefetching based last-level cache optimization for chip multiprocessors [J]. *Journal of Computer-Aided Design & Computer Graphics*, 2012, 24(9): 1241–1248.

[10] 张建勋, 古志民, 胡潇涵, 等. 面向非规则大数据分析应用的多核帮助线程预取方法[J]. 通信学报, 2014, 35(8): 137–146.
Zhang Jian-xun, Gu Zhi-min, Hu Xiao-han, *et al*.. Multi-core helper thread prefetching forirregular data intensive applications[J]. *Journal on Communications*, 2014, 35(8): 137–146.

[11] Marin G, McCurdy C, and Vetter J S. Diagnosis and optimization of application prefetching performance[C]. Proceedings of the 27th International ACM Conference on International Conference on Supercomputing, Oregon, USA, 2013: 303–312.

[12] Garside J and Audsley N C. Prefetching across a shared memory tree within a network-on-chip architecture[C]. Proceedings of 15th International Symposium on System-on-Chip, Melbourne, Australia, 2013: 1–4.

[13] Jain A and Lin C. Linearizing irregular memory accesses for improved correlated prefetching[C]. Proceedings of the 46th IEEE/ACM International Symposium on Microarchitecture (MICRO), Davis, USA, 2013: 247–259.

[14] Zhao Y, Yoshigoe K J, and Xie M J. Pre-execution data prefetching with I/O scheduling[J]. *The Journal of Supercomputing*, 2014, 68(2): 733–752.

[15] 巫旭敏, 殷保群, 黄静, 等. 流媒体服务系统中一种基于数据预取的缓存策略[J]. 电子与信息学报, 2010, 32(10): 2440–2445.
Wu Xu-min, Yin Bao-qun, Huang Jing, *et al*.. A prefetching-based caching policy in streaming service systems[J]. *Journal of Electronics & Information Technology*, 2010, 32(10): 2440–2445.

[16] 刘斌, 赵银亮, 韩博, 等. 基于性能预测的推测多线程循环选择方法[J]. 电子与信息学报, 2014, 36(11): 2768–2774.
Liu Bin, Zhao Yin-liang, Han Bo, *et al*.. A loop selection approach based on performance prediction for speculative multithreading[J]. *Journal of Electronics & Information Technology*, 2014, 36(11): 2768–2774.

[17] Emma P G, Hartstein A, Puzak T R, *et al*.. Exploring the limits of prefetching[J]. *IBM Journal of Research and Development*, 2005, 49(1): 127–144.

[18] Srinath S, Mutlu O, Hyesoon K, *et al*.. Feedback directed prefetching: improving the performance and bandwidth-efficiency of hardware prefetchers[C]. Proceedings of the IEEE 13th International Symposium on High Performance Computer Architecture, Arizona, USA, 2007: 63–74.

[19] Doweck J. White paper: inside intel core microarchitecture and smart memory access[R]. Intel Corporation, 2006.

[20] Hui K and Jennifer L W. To hardware prefetch or not to prefetch?: a virtualized environment study and core binding approach[C]. Proceedings the 8th International Conference on Architectural Support For Programming Languages And Operating Systems, Houston, USA, 2013: 357–368.

黄 艳： 女，1976 年生，博士，副教授，主要研究领域为高性能计算、并行计算.

张启坤： 男，1980 年生，博士，讲师，主要研究领域为分布式计算，网络与信息技术.

段赵磊： 男，1978 年生，博士，副教授，主要研究领域为分布式计算、网络与信息技术.

古志民： 男，1964 年生，博士后，教授，主要研究领域为并行计算、网络与信息技术.