Methods to improve performance of instruction prefetching through balanced improvement of two primary performance factors

Gi-Ho Park \(^a,1\), Oh-Young Kwon \(^b,2\), Tack-Don Han \(^a,3\), Shin-Dug Kim \(^a,\ast\)
Sung-Bong Yang \(^a,4\)

\(^a\) Parallel Processing System Laboratory, Department of Computer Science, Yonsei University, 134, Shinchon-Dong, Seodaemun-Ku, Seoul 120-749, South Korea
\(^b\) System Engineering Research Institute (SERI), Taejon, South Korea

Received 23 September 1996; received in revised form 5 February 1997; accepted 12 March 1997

Abstract

The performance of conventional instruction prefetching mechanisms (IPMs) is analyzed in this paper based on two performance factors, i.e., the cache miss ratio and the average access time for successfully prefetched blocks. Although significant performance improvement (PI) can be obtained by improving these two factors, most conventional prefetching mechanisms improve only one factor out of these two factors. Fetching multiple blocks for a prefetch request and prefetching the sequentially next block together with the block that causes a cache miss in lookahead prefetching (LP) are proposed to improve both these factors. A new method to initiate a prefetch request earlier with no degradation of the prefetch accuracy is also presented for a memory system that is constructed as an interleaved memory. Performance evaluation is carried out through trace-driven simulation and the proposed prefetch scheme reduces 45–63% of the memory access delay time (MADT) for the cache system that does not perform any prefetching.

Keywords: Instruction prefetching; Cache performance model with prefetching; Instruction cache; Trace-driven simulation; Performance analysis

\(^\ast\) Corresponding author. E-mail: sdkim@kurene.yonsei.ac.kr

1 E-mail: ghpark@kurene.yonsei.ac.kr
2 E-mail: oykwon@kurene.yonsei.ac.kr
3 E-mail: hantack@kurene.yonsei.ac.kr
4 E-mail: yang@kurene.yonsei.ac.kr

1383-7621/0165-6074/98/$19.00 © 1998 Elsevier Science B.V. All rights reserved.

PII $1383-7621(97)00021-0$
1. Introduction

Modern microprocessors employ techniques such as superscalar and superpipelining to increase the degree of instruction level parallelism, and their operating speeds have eventually increased incredibly. Thus, as the performance gap between the Central Processing Unit (CPU) and the main memory increases, instruction cache misses can cause significant performance degradation in these high performance microprocessor-based systems.

Instruction prefetching mechanisms (IPMs) were proposed to reduce the number of instruction cache misses. Although most of these mechanisms are concentrated on the reduction of cache misses, the average access time for successfully prefetched blocks also affects the performance of instruction prefetching significantly. For this reason, Przybylski insisted that considerable performance improvement (PI) cannot be obtained by instruction prefetching [1]. This is due to the long access time for the successfully prefetched blocks. Some mechanisms to reduce the average access time for successfully prefetched blocks were proposed as well [2–4]. However, these methods heavily depend on the experimental approach rather than on the analysis of the effects of miss reduction (MR) and average access time for successfully prefetched blocks on the performance of instruction prefetching.

In order to analyze the influence of these two factors on the performance of instruction prefetching, a cache performance model is defined in this paper, for a cache memory system that performs any instruction prefetching. Based on the devised cache performance model, it is known that PI via instruction prefetching is equal to the value obtained by multiplication of the reduced cache miss ratio and the cache miss penalty reduced by the average access time for successfully prefetched blocks.

Performance analysis on instruction prefetching methods is carried out to evaluate the PI of these mechanisms from the point of view of two factors mentioned above. This analysis reveals that conventional IPMs reduce cache misses considerably, but they have relatively long average access times for successfully prefetched blocks in systems with limited memory bandwidth, such as high-performance microprocessor-based systems. As a result, these conventional IPMs cannot achieve considerable PI in general. Lookahead prefetching (LP) proposed to decrease the average access time for successfully prefetched blocks cannot achieve much PI either [2]. This is because LP cannot reduce cache misses effectively due to lower prefetch accuracy compared to the conventional prefetching mechanisms even though it decreases the average access time for successfully prefetched blocks.

In our previous work, a new prediction method using a prediction table and Non-Referenced Prefetch (NRP) cache was proposed to improve LP [3]. A few other methods are proposed in this paper based on the above observations. Two methods are devised to utilize the memory system characteristics for the improvement of IPM. First, a method to fetch multiple blocks for a prefetch request is presented when the memory system supports burst transfer. Second, a new scheme to determine the initiation time of a prefetch request is provided when the memory system can support multiple memory requests, such as the interleaved memory system. The time to initiate a prefetch request is moved from at the time when an instruction within a block is executed for the first time by the CPU to at the time when a block is determined to be referenced by the CPU. The idea is to initiate the prefetch request as early as possible without any degradation of prefetch accuracy by using the property of the table based prediction method. Prefetch on miss method is adapted to LP to alleviate its drawbacks. LP tends to experience burst cache misses when a cache miss occurs in many cases.
This drawback can be alleviated by prefetching the sequentially next block together with the block that causes a cache miss when a cache miss occurs.

Performance evaluation is carried out through the trace-driven simulation. The experimental results show that the LP mechanism which adopts the methods proposed in this work provides an improvement in the memory access time over other prefetching mechanisms. The proposed prefetching scheme reduces 45–63% of the memory access delay time (MADT) of the cache system that does not perform any prefetching.

In Section 2, a cache performance model is presented for a cache system that performs instruction prefetching. PI of instruction prefetching is also defined based on the model. Instruction prefetching schemes are introduced and the performance analysis of the devised model is presented in Section 3. Section 4 presents some new methods to reduce the average access time for successfully prefetched blocks. Experimental results obtained from performance evaluation are also presented. The conclusions are drawn in Section 5.

2. PI by instruction prefetching

Miss ratio and effective memory access time are commonly used as the performance metrics in most cache studies. However, it is desirable to use effective access time as the primary performance metric for a cache system that performs any instruction prefetching. This is mainly because effective access time includes any delay for the CPU to access the blocks prefetched, while miss ratio cannot reflect this delay. In Section 2.1, the cache performance model is defined for the cache memory system performing any prefetch mechanism. PI by instruction prefetching is determined based on the devised cache performance model in Section 2.2.

2.1. Cache performance model with prefetching

When a prefetching mechanism is not performed in the cache memory system, effective access time can be represented as follows.

\[ T_{\text{eff}} = T_c + mT_m, \]  

(1)

where \( T_{\text{eff}} \) is the effective access time, \( T_c \) the cache hit time, \( m \) the miss ratio, and \( T_m \) the miss penalty.

However, effective access time cannot be expressed simply as in the above equation when the cache memory system performs any instruction prefetching. In general, instruction prefetching means a mechanism to fetch a memory block from a lower level memory to an upper level memory that is expected to be referenced prior to the actual reference by the CPU. Instruction prefetching performed between the first level cache and the main memory is considered in this paper.

A Cache memory system that performs an instruction prefetching often consists of the cache memory, a prefetch buffer (PB), and a prefetch control unit. The PB may not be necessary if the blocks prefetched are directly stored in the cache memory, called the cache prefetching. Because cache prefetching might cause cache pollution problems, most prefetching mechanisms use PB to store the blocks prefetched. The general construction of a cache memory system combined with any prefetching mechanism is shown in Fig. 1.

Prefetching can be performed in two steps. First, the block to be prefetched is determined. The determination process may vary depending on the prefetching mechanism chosen. In the case of sequential prefetching, the block that is sequentially next to the currently referenced one by the CPU is determined as the block to be prefetched. In target prefetching (TP), a block to be prefetched is determined by using some information, such as previous execution flows and the target address of a branch instruction.
CPU, the block is moved from the PB into the instruction cache. In our model, the PB has eight entry spaces for prefetched blocks and uses the First-In-First-Out (FIFO) replacement policy.

A prefetch can be described as a *successful prefetch* when the block prefetched is actually referenced by the CPU. Successful prefetch, as opposed to cache hit, can cause delays for the CPU to access the block. Fetching a block into the PB or the cache memory requires the same time as the cache miss penalty, denoted as $T_m$. When a prefetch is initiated, the block to be prefetched can be loaded completely into the PB after a cache miss penalty, $T_m$. However, if a reference to the block occurs before a complete loading by the prefetching operation, the reference is delayed until complete loading.

*Prefetch duration time (PDT)* is defined as the time duration between a prefetch initiation time and the actual time of the first reference. This is shown in Fig. 2. Therefore, if PDT for a successful prefetch is less than the miss penalty, $T_m$, the access time for this successfully prefetched block, is $T_m$-PDT. Thus when the cache system performs any prefetching, memory references can be classified into three categories as follows.

1. **Cache hit**: Memory reference is served by the cache memory. Access time of this case is $T_c$.
2. **Successful prefetch**: CPU references the block prefetched. Access time of this case is

![Fig. 2. Prefetch duration time.](image-url)
\( T_m - \text{PDT} \) if \( T_m > \text{PDT} \). If \( \text{PDT} > T_m \), the access time is \( T_c \).

3. **Cache miss**: CPU references a block that does not exist in the cache memory or the PB, and no prefetching is in progress for that block. Because the block should be fetched from the main memory in this case, the access time is the same as the miss penalty, \( T_m \).

Now effective access time for the cache memory system with prefetching mechanism is calculated by

\[
T_{\text{eff}} = T_c + m_{\text{pr}} T_m + (m - m_{\text{pr}}) T_{\text{pr}},
\]

where \( T_c \) is the cache hit time, \( m \) the miss ratio without prefetching mechanism, \( m_{\text{pr}} \) the miss ratio with prefetching mechanism, \( T_m \) the miss penalty, and \( T_{\text{pr}} \) the average access time for successfully prefetched blocks.

Though a successful prefetch can cause some delay compared to the cache hit, it can be considered as a partial hit since it usually causes less delay than the cache miss. Miss ratio with prefetching, \( m_{\text{pr}} \), represents the ratio of memory references which are neither cache hit nor successful prefetch. The average access time for successfully prefetched blocks can be expressed as \( T_{\text{pr}} \). This can be computed as

\[
T_{\text{pr}} = \frac{\sum_{i=1}^{\text{max(PDT)}} [N(PDT = i) \times (T_m - \min(i, T_m))] \times (T_m - \min(i, T_m))}{\sum_{i=1}^{\text{max(PDT)}} N(PDT = i)},
\]

where \( T_m \) is the miss penalty, PDT the prefetch duration time, \( \text{max(PDT)} \) the maximum PDT among all successful prefetches, and \( N(PDT = i) \) the number of prefetches with \( PDT = i \).

\[
\text{PI} = (T_c + m T_m) - (T_c + m_{\text{pr}} T_m + (m - m_{\text{pr}}) T_{\text{pr}}) \\
= (m - m_{\text{pr}}) T_m - (m - m_{\text{pr}}) T_{\text{pr}} \\
= (m - m_{\text{pr}})(T_m - T_{\text{pr}}).
\]  

Eq. (4) indicates that PI by instruction prefetching is equal to the value of \((m - m_{\text{pr}})(T_m - T_{\text{pr}})\), i.e., the reduced cache miss ratio times the cache miss penalty reduced by the average access time for successfully prefetched blocks. Therefore two factors must be improved to achieve significant PI through instruction prefetching. Much PI cannot be obtained by improving only one factor. This is especially true when one factor already has a higher efficiency than the other, e.g., a certain prefetch mechanism that reduces many cache misses with long average access times for successfully prefetched blocks. In this case, a more drastic PI can be obtained by reducing \( T_{\text{pr}} \) rather than by reducing \( m_{\text{pr}} \).

In Section 3, performance of conventional IPMs is analyzed based on the above observation. The following analysis would show which one of the two factors \((m_{\text{pr}} \text{ and } T_{\text{pr}})\) is more important in the PI of prefetching mechanisms.

### 3. Performance of instruction prefetching schemes

In this section, IPMs are introduced and each corresponding performance is then analyzed. For each IPM, the amount of cache MR and average access time for successfully prefetched blocks are investigated. Guidelines for achieving a higher PI can be obtained through these analyses.

#### 3.1. Instruction prefetching schemes

Many instruction prefetching schemes have been proposed to improve the cache memory performance by reducing cache misses. The conventional and simplest IPM is **sequential prefetching**
(SP) [1,5]. SP is designed to utilize the sequentiality of the memory access pattern. SP can reduce cache misses to some extent by prefetching the sequentially next block based on the sequential memory pattern. However, it cannot reduce cache misses caused by non-sequential paths, which are executed due to flow control instructions such as conditional branches, and subroutine calls.

TP was proposed to reduce cache misses caused by the branch instructions [2–4,6,7]. The history information of previous control flows is used to determine the block to be prefetched in TP based on the property that branch instructions, even if they are conditional branches, tend to follow the previous control flows again in most cases [8]. Because TP uses the operational characteristics of branch instructions that degrade the performance of SP, it has higher prefetch accuracy and lower cache miss ratio compared to SP.

A prefetch mechanism, called hybrid prefetching (HP), which combines both sequential and target prefetching, was proposed to reduce much more cache misses. This prefetch mechanism reduces cache misses that can be eliminated by SP and TP cumulatively.

These prefetch mechanisms, e.g., sequential, target, and hybrid prefetching, focus on the cache miss reduction to improve the performance of cache memory system, with little attention on the accessing delay for successfully prefetched blocks. This may be due to two reasons. First, most of the previous research uses cache miss ratio as the performance metric to evaluate the proposed mechanism. Because cache miss ratios cannot include the effects of accessing delays for successfully prefetched blocks, the delays are not considered seriously. Abundant memory bandwidth of the target machine can be the second reason [6]. In this case, most successfully prefetched blocks can be loaded before they are referenced by the CPU with high memory bandwidth, and no delay is required to access the blocks.

Accessing delay for successfully prefetched blocks affects the performance of instruction prefetching significantly when the system has the restricted memory bandwidth, such as high-performance microprocessor-based systems. LP was proposed to reduce the accessing delay for successfully prefetched blocks [2]. LP prefetches the d-th block that is expected to be referenced after the currently referenced block by the CPU rather than prefetch the block that has the highest possibility to be referenced right after the current block. The d is called the degree of LP. Although accessing delay for successfully prefetched blocks decreases as the degree of LP increases, fewer cache misses can be eliminated by LP because it is difficult to predict correctly the d-block to be referenced after the current block. Therefore, LP, when its degree of LP is two or three, can achieve considerable PI [2].

The cache miss ratio \( (m_p) \) and average access time for successfully prefetched blocks \( (T_p) \) are evaluated in the Section 3.2. PI obtained by each of the above IPMs is analyzed based on these two factors.

3.2. Performance analysis of instruction prefetching schemes

Performance of IPMs introduced in the previous section is analyzed in this section. Cache miss ratio, \( m_p \), effective access time, \( T_{eff} \), and average access time for successfully prefetched blocks, \( T_p \), are evaluated for each prefetch mechanism. The requirements for improving the performance of instruction prefetching are determined through this analysis.

Trace-driven simulation is performed to evaluate the performance of IPMs. A system that has a high performance RISC processor is assumed as the base system [7]. Basic architectural parameters used are presented in Table 1. Some of the cache parameters are fixed due to tremendous
Table 1
Base architectural parameters used for performance analysis

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>I-cache size</td>
<td>8/16/32 Kbytes</td>
</tr>
<tr>
<td>I-cache block size</td>
<td>16/32 bytes</td>
</tr>
<tr>
<td>I-cache associativity</td>
<td>direct-mapped</td>
</tr>
<tr>
<td>( T_c )</td>
<td>1 cycle</td>
</tr>
<tr>
<td>CPI on the I-cache hit</td>
<td>1</td>
</tr>
<tr>
<td>Transfer rate</td>
<td>1 word (4 bytes)/cycle</td>
</tr>
<tr>
<td>( T_m )</td>
<td>12/16/20/24 cycles</td>
</tr>
<tr>
<td>Prefetch buffer size</td>
<td>8 entry</td>
</tr>
<tr>
<td>Number of prediction table</td>
<td>Same number of cache entry blocks</td>
</tr>
<tr>
<td>Prediction table associativity</td>
<td>Direct-mapped</td>
</tr>
</tbody>
</table>

Simulation time. In particular, cache organization is considered as the direct-mapped cache in the simulation. The reason for selectively simulating the direct-mapped cache is not only for the simulation time but also for its architectural merits. Specifically, the direct-mapped cache has fast access time for a cache hit, superior effective (average) access time, and simple control logic. In general, direct-mapped caches are adopted in most RISC microprocessors-based systems in which the CPU cycle time is often determined by the cache hit (access) time [9]. A baseline architecture is assumed to execute one instruction per CPU cycle, i.e., Cycles Per Instruction (CPI) = 1, when no cache miss occurs.

One cycle delay is assumed for prediction table access and cache lookup in the prefetch mechanisms that use the prediction table, e.g., TP, HP, and LP. Access on a block in the PB has to experience a one cycle delay because the CPU cycle time can be increased in the case of accessing the block in the PB without one cycle penalty [5]. This is mainly because performance degradation induced by the increased CPU cycle is more critical than PI gained by the prefetch mechanism [1].

A cache miss has higher priority than any prefetch request. This is because the normal memory access should not be affected by the prefetch request. A new prefetch request can have a higher priority than the current prefetch in progress. Because initiating a new prefetch request means that the ongoing prefetch will probably be useless, PI can be expected by assigning a higher priority for a new prefetch request. However, an ongoing prefetch has higher priority in the LP because a new prefetch request can be initiated before the prefetch request, which might be correct, completes.

Traces are obtained by Spa tool on SUN Sparstation [10]. Ten million instruction references from each program are used in the simulation. Application programs used for generating traces are shown in Table 2. Performance metrics, i.e., cache miss ratio, \( m \), without prefetching no prefetching (NP), cache miss ratio with prefetching mechanism, \( m_{pr} \), effective access time, \( T_{eff} \), and average access time for successfully prefetched blocks, \( T_{pr} \), of each prefetch mechanism are shown in Sections 3.2.1–3.2.3.

3.2.1. Effective access time (\( T_{eff} \))

Effective access time (\( T_{eff} \)) represents the performance of the IPM. To compare the PI of IPMs more clearly, MADT is defined. MADT of a cache system is used to denote the average delayed cycle time incurred to access the memory for instructions other than cache hit during the program execution. Thus, MADT can be calculated as in the following equation.

Table 2
Programs used for generating traces

<table>
<thead>
<tr>
<th>Program</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gcc</td>
<td>GNU C compiler</td>
</tr>
<tr>
<td>Espresso</td>
<td>Boolean function minimization</td>
</tr>
<tr>
<td>Spice</td>
<td>Analog circuit simulator</td>
</tr>
<tr>
<td>F77</td>
<td>FORTRAN compiler</td>
</tr>
<tr>
<td>Latex</td>
<td>Type setter</td>
</tr>
</tbody>
</table>
MADT = \[ \text{total amount of delay in accessing instructions} \]
\[ \text{number of instructions executed} \] (5)

The amount of memory accessing delay reduced by each IPM can be compared clearly by using the MADT. Because it is assumed that one instruction can be executed in one CPU cycle, effective access time \( T_{\text{eff}} \) is equal to the value of \( (1 + \text{MADT}) \).

Fig. 3 shows the MADT of each IPM when the system is constructed to have a 8 Kbyte instruction cache (I-cache), an 8 word (32 byte) cache block, and a 16 CPU cycle miss penalty. These cache parameters are assumed to be used as the base cache configuration throughout the simulation and analysis.

Because the conventional HP mechanism that prefetches both sequential and non-sequential paths can be performed only in systems that have much higher memory bandwidth such as supercomputers, it cannot be performed in a base system that is assumed to have memory bandwidth restriction. Adaptive hybrid prefetching (AHP) is proposed to achieve PI by prefetching both paths under the limited memory bandwidth [3]. MADT of a modified version of the conventional HP, called AHP, is shown in Fig. 3. AHP determines the block to be prefetched first between the sequential block and the non-sequential block by using the information of previous control flows. If the first candidate block is already in the on-chip memory, i.e., the I-cache and the PB, prefetching of the other block is tried. Because the hit ratio of a several kilobyte cache is usually over 90%, there is a good possibility that one of the two prefetch candidate blocks is already in the on-chip memory. Therefore, given a specific block, the AHP can provide the same effect as that of the two candidate blocks which are prefetched by prefetching only one of the two candidate blocks in many cases [3].

TP, AHP, and LP mechanisms shown in Fig. 3, use the prediction table to store the history information. The number of prediction table entries is equal to the number of cache blocks existing in the instruction cache. A direct-mapped scheme is used for the prediction table. Each entry of the prediction table consists of three fields, i.e., current block address, target block address, and history information field. The history information field is the bit-field that indicates whether the previous control flow is sequential or non-sequential [3,6].

MADT of LP presented in Fig. 3 is that of two-block LP. LP that the degree of lookahead is two tries to prefetch the block which is expected to be referenced after the next block of the current block.

TP shows the best performance for all applications. AHP, SP, and LP show better performance in that order for most applications. In the case of MADT, however, a cache system with any prefetching mechanism can provide negligible PI than in the case without any prefetching.

There are interesting observations that can be made from Fig. 3. One is that MADT of the AHP mechanism is worse than that of the TP mechanism. Average access time for successfully prefetched blocks \( T_{\text{eff}} \) may be the reason for the

![Graph](image_url)
poor performance of AHP as it can reduce more cache misses than TP by prefetching both sequential and non-sequential paths. The little PI of the LP mechanism may result from a higher cache miss ratio due to lower prefetch accuracy since the LP mechanism must have smaller average access time for successfully prefetched blocks than any other prefetch mechanisms in nature [2]. The cache miss ratio, \( m_{pr} \), and average access time for successfully prefetched blocks, \( T_{pr} \) of IPMs are evaluated for analyzing the PI of each prefetch mechanism in Sections 3.2.2 and 3.2.3.

### 3.2.2. Cache miss ratio \( (m, m_{pr}) \)

Cache miss ratio, \( m \), for the cache system without prefetching, and cache miss ratio, \( m_{pr} \), for the cache system with prefetching are shown in Fig. 4. Cache miss ratio, \( m_{pr} \), for each IPM shows its corresponding effect as each IPM is basically designed to achieve. NP shows the highest miss ratio, and LP, SP, TP, and AHP have lower miss ratios than NP in the same order.

One interesting observation made from Fig. 4 is that SP can reduce the cache miss ratio significantly over the cache system without any prefetch mechanism while the TP and AHP mechanisms can reduce only a few more cache misses than SP.

### 3.2.3. Average access time for successfully prefetched blocks \( (T_{pr}) \)

For each prefetching mechanism, the average access time for successfully prefetched blocks is presented in Fig. 5. As the miss penalty of the simulated system is assumed to be 16 CPU cycles, the performance gain by one successful prefetch is equal to the value of 16 – \( T_{pr} \). Two-block LP shows the smallest \( T_{pr} \) because it initiates prefetch requests earlier than any other prefetching mechanism.

Considering that SP tries to prefetch sequential blocks which usually have longer PDT, it is expected that SP provides smaller average access time for successfully prefetched blocks, \( T_{pr} \), than that of TP. This is mainly because the sequential blocks are accessed after more instructions are executed than in the case of target blocks in general. However, the average access time for successfully prefetched blocks in the case of SP is actually larger than that in the TP as shown in Fig. 5. The main reason for this is thought that many of the blocks prefetched by wrong prediction tend to be accessed at some later times, as indicated in the wrong path instruction prefetching method [3,4].

---

**Fig. 4.** Cache miss ratios of various IPMs.

**Fig. 5.** Average access time for successfully prefetched blocks, \( T_{pr} \), for various IPMs.
Note that eight block spaces are provided in the PB as explained.

3.2.4. Two alternatives to improve the performance of instruction prefetching

PI attainable by instruction prefetching is the value of \( (\frac{m - m_{pr}}{m} \times 100)(\%)) \) as shown in Eq. (4). MR and prefetch efficiency (PE) are defined to represent these two factors as follows.

\[
MR = \frac{m - m_{pr}}{m} \times 100(\%)
\]

\[
PE = \frac{T_m - T_{pr}}{T_m}
\]

Fig. 6 shows the PI of the IPM using the two metrics, MR and PE. In Fig. 6, the x-axis displays the values of MR, and the y-axis gives the values of PE. If any specific IPM reduces 60% of cache misses, i.e., MR = 60%, and has a PE of 0.4, i.e., PE = 0.4, it is represented by the point A in Fig. 6. PI of the given IPM is represented as the area of B in Fig. 6(a). It means that the IPM reduces 24% (60% \times 0.4) of MADT in the cache memory system that does not perform any instruction prefetching. Two alternatives can be considered to increase the PI of the given IPM. The first one would be to increase MR, which moves the point A to the right along the x-axis, and the second would be to improve PE, which moves the point A upward along the y-axis. PI's that can be obtained by these two methods are represented as areas C and area D, respectively. In the case of the IPM denoted as A in Fig. 6(a), 16% (represented by area C) of MADT can be reduced by eliminating whole cache misses, i.e., MR = 100%, while 36% of MADT can be reduced by achieving perfect PE, i.e., PE = 1. The diagonal line in Fig. 6 represents the prefetching mechanisms that have the equal values of MR and PE. Thus, if any

![Graph showing MR and PE for various prefetch mechanisms.](image)

Fig. 6. MR and PE for various prefetch mechanisms.
IPM is positioned in the above region of the diagonal, PE > MR, and the condition PE < MR is satisfied for the IPMs which are represented by the points below the diagonal. Greater PI can be achieved by increasing the PE if the prefetching mechanism satisfies the condition PE < MR, i.e., positioned in the region below the diagonal, while increasing MR would be more effective if PE > MR.

Various IPMs introduced in Section 3.1 can be classified by MR and PE obtained from the simulation. Performance characteristics for these IPMs are presented in Fig. 6(b). SP has a large MR ranging 64–74% while its PE is small, i.e., 0.25–0.33. TP and AHP have relatively large MR, 66–76% and 75–86% with respect to the small PE of 0.31–0.54 and 0.26–0.49, respectively. TP and AHP have larger MR and PE than SP, which results in better performance of these mechanisms. Though AHP achieves a larger MR than TP, it shows a poorer performance than TP because of much lower PE. Because SP, TP and AHP are positioned below the diagonal, increasing PE is more important for PI in these mechanisms. The LP mechanism that is proposed to increase PE appears above the diagonal. Although LP increases PE to some extent as expected, it, however, shows a poorer PI than TP and AHP due to small MR, as shown in Fig. 3. Therefore, methods to increase PE with small or no degradation of MR are essential for significant PI of these IPMs.

4. Methods to increase PE

Increasing PE is essentially required to improve the performance of most IPMs introduced in Section 3. PDT can be increased by initiating the prefetch request as early as possible, which improves PE. It was proposed that LP increases PE by initiating the prefetch request earlier. However, there is a trade-off between PE and MR in LP. PE is improved with increment in the degree of LP. But MR goes down as the degree of LP increases due to lower prefetch accuracy [2,3]. To overcome this drawback of LP, Park et al. proposed NRP cache, with a method to determine the block to be prefetched based on the prediction table [3]. In this paper, methods that utilize the characteristics of the memory system structure are proposed to improve PE without significant degradation of MR.

Prefetching multiple sequential blocks for a prefetch request is one method of improving PE. When multiple sequential blocks are prefetched, the blocks except the first block can be fetched more efficiently. In this case, both memory latency and transfer time are required to fetch the first block. However, the ensuing sequential blocks can be fetched by the transfer time when the memory system supports the burst block transfer. The average access time for those sequential blocks can thus be reduced by this property resulting in improved PE.

Besides fetching multiple blocks for a prefetch request, a technique to fetch the block that causes a cache miss and prefetch its sequentially next block when a cache miss occurs in LP, is proposed to utilize this property and to overcome the shortcoming of LP. The shortcoming of LP is that LP tends to experience burst cache misses when a cache miss occurs in many cases.

Also a new idea about the prefetch initiation time is devised when the memory system can support multiple memory requests simultaneously, such as the interleaved memory. This mechanism improves the PE of the IPMs that use the prediction table to determine the block to be prefetched without any degradation of MR.

4.1. LP with prefetch on cache miss

There are many parameters which influence the determination of how many blocks are prefetched by a single prefetch request. Sequentiality of
memory reference and additional memory traffics caused by instruction prefetching particularly should be seriously considered. A prefetching mechanism that prefetches two consecutive blocks when a prefetch request is initiated is considered in this work.

There is a trade-off between PE and MR in LP as mentioned before. Though a two-block LP mechanism achieves much improvement on PE, small MR due to lower prefetch accuracy causes poor performance of LP, as shown in Figs. 6(b) and 3. Prefetch accuracy of LP degrades about 10% for most application programs when the degree of LP increases from one to two, and the prediction mechanism becomes more complicated as the degree of LP increases [2,3]. Considering these facts, PI cannot be obtained by merely increasing the degree of LP, and this is the reason that two-block LP is chosen in this work.

Large cache miss ratio of LP is caused by the property of LP. Because the block expected to be referenced after the next block of the current block is prefetched in two-block LP, PI by prefetching the second block can hardly be achieved in the case of cache miss.

Suppose that four consecutive cache blocks, e.g., A, B, C, and D, are not in the cache memory and control is transferred to the cache block A. Also assume that the prediction table corresponding to the blocks A and B have the contents shown in Fig. 7(a), and that control flow will be transferred from the cache block A to the cache blocks B, C, and D, in this order. The history information field of the prediction table contains the value for indicating that the blocks C and D are the blocks to be prefetched. Because the actual value of the history information field is determined irrespective of whether blocks C and D are sequential or not, the value is denoted as X. Then, a prefetch for the block C is tried when the block A is referenced by the CPU in two-block LP as shown in Fig. 7(a). Thus, a cache miss on block B occurs after the cache miss on block A. If the prefetch request for block C is aborted to service the cache miss on block B, another cache miss on block C may occur after block B is referenced by the CPU while prefetching on block D is in progress. This situation can occur in many cases when the cache miss has a higher priority than an ongoing prefetch and the memory system cannot support multiple memory requests.

This drawback of LP can be alleviated by adopting the prefetch on miss technique to the LP. In the proposed LP mechanism that adopts Prefetch on Miss technique (LPPM), two blocks, i.e., block A that causes a cache miss and the sequentially next

![Diagram](image-url)
block $A + 1$ are fetched. The prefetch operation in the LPPM mechanism is shown in Fig. 7(b) when a cache miss occurs due to block $A$. After complete fetching of the two blocks, prefetch request on block $C$ determined by the prediction mechanism is initiated. Block $C + 1$ is also prefetched because two consecutive blocks are prefetched by one prefetch request in the mechanism. The bold rectangle in Fig. 7(b) represents the blocks to be prefetched by a memory request.

This mechanism takes advantage of two aspects, namely multiple block fetching and alleviating the drawback of LP. First, because of the sequential memory access pattern, there is a good possibility that block $B$ is actually block $A + 1$. In that case, blocks $B$ and $C$ are prefetched in the correct order in the proposed LPPM mechanism, which results in improvement of MR. Second, block $A + 1$ can be prefetched more efficiently by fetching two blocks, blocks $A$ and $A + 1$, together as explained before. This second aspect can improve PE.

Performance evaluation of the LPPM mechanism is performed via simulation. The MADT of each prefetch mechanism is shown in Fig. 8. The LPPM mechanism shows smaller MADT than any other prefetch mechanism and reduces about 32-56% of the original MADT. To analyze the effect of the proposed method from the point of PE and MR, these two factors are presented in Fig. 9. As shown in Fig. 9, both MR and PE of LPPM are improved over LP for all application programs. The points, namely the performance of each application program, are consistently moved in an upper-right direction, which indicates an improvement in both these factors.

4.2. Prefetch request initiation time in the interleaved memory system

It is very important that the performance factor determines the time to initiate a prefetch request in the prefetching mechanism. In general, if a prefetch request is initiated earlier, PE is increased while MR tends to be decreased. Considering this fact, determination of the time when a prefetch request is initiated is very crucial for acceptable performance of instruction prefetching. It is desirable to initiate a prefetch request as early as possible if it does not increase cache misses.

The sequentially next block to the current block is prefetched unconditionally in SP, and a block predicted to be referenced after the currently referenced block by the CPU is prefetched in TP. A prediction mechanism is used to determine the block to be prefetched in TP. The block to be prefetched is not changed even though the prefetch request is initiated at any time during the execution of the currently accessed block. This is because in general the information used to determine the block to be prefetched is managed by the unit of a cache block in TP. Thus, PE can be increased by initiating the prefetch request when a block is newly referenced by the CPU without any increase of cache misses.

Fig. 8. MADT of the proposed LPPM mechanism.

There is another method that increases PE but does not decrease MR, when the memory system
is constructed as an interleaved memory. PE can be increased based on the property that multiple memory requests can be serviced in parallel in the interleaved memory system without decreasing MR. To use this property effectively, a prefetch request is initiated not at the time when an instruction of a newly referenced block is executed by the CPU, but at the time when a block is determined to be referenced by the CPU.

Prefetch initiation time is changed from the case of the conventional method as follows. If a newly referenced block exists in the cache already, a prefetch request is initiated when the block is referenced by the CPU in the same way as the conventional case. But if any reference to a new block, say block A, causes a cache miss or prefetching on block A is in progress, a prefetch request for a block, say B, determined by the prediction mechanism, is initiated immediately without waiting for the loading of block A to be completed. So, the fetching of block A and prefetching of block B are performed in parallel, which is possible only when the memory system can support multiple memory requests, as in the interleaved memory system. The delay time to fetch block A is used to prefetch the block B by overlapping these two memory requests. Thus, it can be expected that PE is increased significantly by this method.

However, initiating a prefetch request at the same time when a block is determined to be referenced can be performed only in prefetching
mechanisms that use the prediction table. In threaded prefetching, the history information which determines the prefetch candidate is not available before the corresponding block is loaded completely [2]. Because the block to be prefetched is determined during the execution of the corresponding block in the wrong-path instruction prefetching method, the above idea cannot be adopted to this method to improve PE either [4].

The MADT of an LPPM mechanism when the memory system is constructed as a four-way cache block-interleaved memory is shown in Fig. 10. The proposed LPPM mechanism is denoted as LPPM(MI). Performance of the LPPM mechanism in the interleaved memory system is more superior to all other prefetching mechanisms. The LPPM(MI) mechanism reduces 45–63% of the MADT in the cache system that does not perform any prefetching. Fig. 11 shows that PE and MR of the LPPM(MI) mechanism are
improved in most application programs. Significant improvement of MR is achieved in LPPM(M1) while PE of LPPM(M1) is improved a little compared to those of LPPM. Because initiating prefetch requests is not delayed in the interleaved memory, more cache misses can be eliminated in LPPM(M1) than in LPPM. However, PE of LPPM(M1) cannot be improved much more than that of LPPM because only one memory request is in transfer at any time even in the interleaved memory.

4.3. Hardware cost for LPPM

Hardware cost and PI of LPPM are compared to evaluate the cost/performance efficiency. Hardware cost for LPPM includes the storage space for prediction table, PB, and the logic to control the prefetch operations. Hardware cost for prediction table depends on the address space, cache size, and cache block size. In a base system that has 8 Kbyte cache and 8 word cache block with 32 bit address space, the hardware cost for LPPM can be calculated as follows:

Number of bits for prediction table

\[= (\text{current block address field size} + \text{target block address field size} + \text{history information field size}) \times \text{No. of prediction table entries} \]

\[= (19 + 27 + 1) \times 256 = 12032 \text{ bits}.\]

Number of bits for PB

\[= \text{No. of bits for a cache block entry} \times 8 \]

\[= (8 \times 32) \times 8 = 2048 \text{ bits}.\]

Storage space for the prediction table and PB is smaller than 2 Kbytes, i.e. one-fourth of the cache size. Though the hardware cost of the logic for prefetch operation is not included in the above calculation, the cost of the additional hardware is not increased significantly because the space for cache tag and cache control logic for the cache system are not included.

The MADTs of LPPM and the cache system that does not perform any instruction prefetching with double sized cache memory are shown in Fig. 12. MADTs of LPPM(M1) with 8 Kbyte cache are smaller than those of NP with 16 Kbyte cache except one application program, i.e., latex. The reason is thought to be the working set of the latex program is small to fit in the 16 Kbyte cache. Considering these facts, hardware cost for LPPM can be affordable.

5. Conclusion

A cache performance model for a cache system performing an instruction prefetching and the performance analysis of IPMs based on the model are presented in this paper. Two important performance factors are determined through performance analysis. These are the average access time for successfully prefetched blocks \(T_{pr}\) and cache miss ratio \(m_{pr}\) with prefetching. It is known through the performance analysis that reducing
$T_{pr}$ is more important than reducing more cache misses in the IPMs, especially when the systems have relatively lower memory bandwidth and small cache block size. This is mainly because only a small portion of the miss penalty can be hidden by a successfully prefetched block in these systems.

Fetching multiple blocks for a prefetch request and prefetching the sequentially next block with the block that causes a cache miss in LP are investigated to reduce the average access time for successfully prefetched blocks. A new prefetch scheme that adopts these methods, called LPPM, is presented as well. In addition, a new method to determine the initiation time of a prefetch request is presented for a memory system that is constructed as an interleaved memory system. An LPPM(MI) mechanism is introduced by adopting the method that initiates prefetch request at the time when a block is determined to be referenced by the CPU rather than at the time when an instruction is executed by the CPU into the LPPM mechanism. Performance evaluation is performed to clarify PI of the proposed prefetch mechanism. The proposed prefetch scheme (LPPM(MI)) reduces around 45–63% of the MADT of the cache system that does not perform any prefetching.

Acknowledgements

This paper was supported by a Special fund for the University research institute, Korea Research Foundation, 1995.

References

Shin-Dug Kim received a B.S. degree in electronic engineering from the Yonsei University, Seoul, Korea, in 1982, and an M.S. degree in electrical engineering from the University of Oklahoma in 1987. In 1991, he received his Ph.D. degree in electrical engineering from the Purdue University. He was an associate professor of computer engineering at the KwangWoon University, Seoul, Korea from 1992 to 1994. He is currently an assistant professor of computer science at the Yonsei University, Seoul, Korea. His research interests include parallel system architectures, parallel algorithm design, heterogeneous computing, and computer architecture. He is a member of the IEEE Computer Society.

Sung-Bong Yang received his B.S. degree from the Yonsei University, Seoul, Korea, in 1981 and his M.S. and the Ph.D. degrees from the University of Oklahoma, in 1986 and 1992, respectively. He was an adjunct assistant professor of the School of Computer Science at the University of Oklahoma for the Fall semester of 1992. He is currently an assistant professor of computer science at the Yonsei University, Seoul, Korea. His research interests include parallel and distributed algorithms, heterogeneous computing, and genetic algorithms.

Tack-Don Han received a B.S. degree in electronic engineering from the Yonsei University, Seoul, Korea, in 1978, and an M.S. degree in computer engineering from the Wayne State University in 1983. In 1987, he received his Ph.D. degree in computer engineering from the University of Massachusetts at Amherst. He was an assistant professor of electrical engineering at the Cleveland State University from 1987 to 1989. He is currently an associate professor of computer science at the Yonsei University, Seoul, Korea. His research interests include VLSI design, digital logic design, advanced computer architecture, and parallel algorithms/architecture. Dr. Han is a member of the IEEE Computer Society and the Association for Computer Machinery.