MEMORY LATENCY REDUCTION USING AN ADDRESS PREDICTION BUFFER

Arthur Billingsley and Douglas Fouts

Department of Electrical and Computer Engineering,
United States Naval Postgraduate School
Monterey, California 93943
e-mail: billings@ece.nps.navy.mil

ABSTRACT

Developing memory systems to support high-speed processors is a major challenge of computer architects. Caches are implemented to improve system performance but the latency of main memory remains a major penalty for a cache-miss. A novel approach to improve system performance is the implementation a memory prediction buffer. The memory prediction buffer (MPB) is inserted between the cache and main memory. It predicts the next cache-miss address and pre-fetches the data. This action decreases main-memory latency and increases system performance.

1. INTRODUCTION

The technological advances in high-speed general purpose processors have outpaced the support provided by main memory systems. In addition, software applications continue to grow in processor and memory requirements. The major factors in the design of memory systems are bandwidth required, main-memory latency, and memory subsystem cost. Large memory subsystems use dynamic random-access-memories because of their low cost per bit. Caching schemes, which employ high-cost, high-speed memories, are implemented to overcome main-memory latency and increase bandwidth. Further gains in memory system performance are possible and design constraints must be examined. One such memory performance enhancement is the prediction of a cache-miss read address request to main memory. If the read address is predicted and the data made available, then the overall system performance is improved.

Designing high-performance computer systems requires viewing processor and the memory structure together. Graphically, a computer system is viewed as in figure 1. It is at this point in design that memory specifications must be considered. If the requirement is for a large address space, slower cheaper main memory is the natural response. This requirement gives rise to the need for an efficient (for the processor perspective) memory hierarchy. Each level of the memory hierarchy, as it approaches the processor, is smaller, faster and more expensive. Each level of the hierarchy is influenced by different factors [PRZYBY90].

- The architectural implementation of the processor impacts cycle time and the number of CPU cycles per instructions
- The instruction set architecture influences the instruction count, the number of CPU cycles per instruction, and the number of references per instruction
- Compiler construction influences instruction count, the number of CPU cycles per instruction, and the number of references per instruction
- Memory subsystems (memory hierarchy) directly impact the number of cycles per reference and the access cycle time

1. MPB - Memory Prediction Buffer
2. PERFORMANCE METRICS

From the system perspective, work completed in time defines system performance. Hence system performance can be described analytically as equation 1.

\[
\text{System Performance} = \frac{\text{Instructions Completed}}{\text{Elapsed Time}}
\]  

This definition of system performance does derive the ubiquitous MIPS units. This unit of measurement should not be used in comparison of different systems performing the same task [PATHEN90]. However, for characterization of a specific system performing the same task, this unit of measure is useful. This measure of performance can be focused in terms of processor cycles.

\[
E = I \cdot CPI \cdot f
\]

Expanding this model gives: The number of cycles per instruction executed is the metric that is directly influenced by the memory subsystem. Statistically, a more stable metric is the effective CPI. The effective CPI is the statistical average of several measurements. The effective CPI is

\[
CPI_{EFF} = \frac{\sum CPI_i}{i}
\]

The number of cycles per instructions is largely determined by processor architecture and register/cache structure (effectiveness). With a focus toward the memory structure, the effective access time of the memory subsystem is the best metric to indicate memory subsystem performance. This parameter depends on the cache access time and the main memory access time. By decreasing the number of cycles per instruction, the system performance is improved. The speedup in system performance is modelled by equation 4.

\[
S = 1 - \frac{CPI_{EFF(MPB)}}{CPI_{EFF}}
\]

Nominal figures for the number of cycles per instruction in high performance processors is 1.2-2.0 CPI. If we assume that the processor can execute instructions at the bandwidth of the memory subsystem, the speedup becomes a function of the effective access time of the memory subsystem.

\[
S = 1 - \frac{E_{AT(MPB)}}{E_{AT}}
\]

The effective access time is a function of the cache performance and main memory performance.

\[
E_{AT} = C_S + C_{HR} \cdot C_R + (1 - C_{HR}) \cdot (C_S + M_R + C_P)
\]

This relationship can be simplified by noting the time for a cache tag search \(C_S\) is very small. In addition, the cache tag search and cache fetch are much smaller than the time to read/fetch from main memory, \(M_R\).

\[
E_{AT} = C_{HR} \cdot C_R + (1 - C_{HR}) \cdot (M_R)
\]

3. MEMORY PREDICTION BUFFER

Since current RISC processors far exceed the capability of main memory systems, the focus for the computer systems architect is how to improve the performance of the memory hierarchy. Caching techniques have proved effective in hiding the latency of main memory. Cache memories are fast and are therefore expensive. Fully associative caches are cost prohibitive and direct-mapped caches offer an excellent alternative [HILL88]. Direct-mapped caches have a higher miss rate than fully-associative or set-associative caches. A disadvantage of cache memories, in general, is the miss penalty [PATHEN90], [PRZYBYZ90]. The reduction of the miss rate and subsequent miss penalty is the motivation for the memory prediction buffer (MPB).

The memory prediction buffer was conceived to predict the next cache-miss address and prefetch the data before the request is made by the processor. In the early research of this idea, efforts turned instinctively toward statistical methods for prediction. The area of digital signal processing was explored for possible solutions to the prediction requirement. Kalman filters, Wiener filters and other adaptive techniques for prediction were proposed and investigated. However, further characterization of the problem gave more constraints for possible solutions.

Cache simulation was achieved using Mark Hill's DINEROIII cache simulator. The model cache is a direct-mapped, 8K data, 8K instruction with a 32 byte line size. Using an ATUM SPICE trace, cache miss addresses were investigated [AGARWL86]. Subsequent traces show spatial locality and temporal locality for the SPICE process. Since no curves are obvious, prediction should employ linear methods. The simulation was configured to give the number of cache hits before a miss is encountered for the data cache only. The average of these miss events give the constraint of time available to predict a miss address. Since the average of cache-hits before a cache-miss is 4-6, it is possible that some 6-10 cycles are available for prediction and prefetch. In addition, the system bus bandwidth must be considered for prefetch solution. These constraints were responsible for the development of a simpler prediction algorithm. The prediction algorithm yields a bias for the ensuing prefetch. If the current address is larger than the past address, then the bias is positive (negative otherwise). The algorithm for the MPB is given in figure 3. The structure of the memory prediction buffer is similar to a conventional fully-associative
cache. The MPB is composed of 16-256 lines of 32 byte blocks. The block size is dependent on the first level cache block size. The optimal size of the MPB is 64-256 line. This is due to the fan-out requirements for the construction of a fully associative cache and the replacement policy used (random vice LRU, FIFO, etc).

4. SIMULATION

The memory prediction buffer determines the future cache miss address using previous cache miss addresses. For this analysis, only the data cache is given a MPB. Given a cache hit ratio of 95%, if the MPB is conservatively correct on 33% of its predictions, a 1.65% cache hit rate increase is realized. For example, an application required 1,000,000 memory accesses. The processor system has a cache implemented with a hit ratio of 95%. Of those 1,000,000 address requests, 50,000 are cache misses. If the MPB is used, 1/3, or 16,666 address are viewed, from the processor, as an additional cache hit. The effective cache hit ratio is improved to 96.6% from 95%. The graph of figure 3 gives the effective cache hit rate as a function of MPB effectiveness. There are four cache hit ratios that are compared, 80, 85, 90, and 95. A sample reading is shown for a base cache hit ratio of 80% with an MPB effectiveness rating of 20%. The resulting effective cache hit ratio for this sample is 84%. This is an increase of 4% in the effective cache hit ratio. The resulting system performance achieves a speedup of 9%. This theoretical example was realized when the MPB was implemented with DINER0 cache simulator.

Table 1: MPB PERFORMANCE

<table>
<thead>
<tr>
<th>LINES</th>
<th>SIZE (BYTES)</th>
<th>LOCAL HITS</th>
<th>TOTAL MISS</th>
<th>SPEED-UP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>32</td>
<td>1024</td>
<td>17.7%</td>
<td>5.7%</td>
</tr>
<tr>
<td>2</td>
<td>64</td>
<td>2048</td>
<td>24.4%</td>
<td>5.3%</td>
</tr>
<tr>
<td>3</td>
<td>128</td>
<td>4096</td>
<td>33.5%</td>
<td>4.7%</td>
</tr>
</tbody>
</table>

Table 1 gives a summary of MPB performance for three various implementations. For these examples, each line of the MPB consists of 32-byte blocks. These are boundary aligned in the same way as the cache. The speedup numbers are modest but, the cost of this implementation is minimal when compared to a 256K next level cache. The MPB is a favorable architectural concept for consideration in systems where the highest possible performance is desired and systems costs are constrained.

5. CONCLUSIONS

The memory prediction buffer is proposed as a component for high performance computer systems. The widening gap between processors speed and memory dictates alternative architectures for reducing main memory latency while restraining costs. The MPB outperforms prefetch always strategies by allowing addressing in the up and down direction. In addition, the MPB does not contribute to pollution of the cache. Effective memory latency reduction must be addressed at the time of system design. As the requirements for a larger address space grows, memory hierarchy design and implementation will continue to increase in complexity. The implementation of a MPB is less expensive than a next-level cache and delivers a comparable performance enhancement. In addition, the algorithm used can be tailored to the proposed system environment. The MPB is shown to improve overall system performance and provide reasonable gains in speedup.

ACKNOWLEDGEMENTS

This research is part of a larger investigation of memory latency reduction strategies for uniprocessor architectures. Besides the authors, others have contributed to this work. In particular, Anita Borg of DEC (for help with address traces), Mark Hill of the University of Wisconsin (for his cache simulator and other assistance) and Richard Ham-
ming of the Naval Postgraduate School (for his research
guidance and statistical insight).

SYMBOLS

\[ E \] 
\text{efficiency}

\[ I \] 
\text{instructions}

\[ CPI_{EFF} \] 
\text{effective cycles per instruction}

\[ S \] 
\text{speedup}

\[ CPI_{EFF(MPB)} \] 
\text{cycles per instruction with memory prediction buffer}

\[ EAT \] 
\text{effective access time}

\[ C_S \] 
\text{cache search time}

\[ C_{HR} \] 
\text{cache hit ratio}

\[ C_F \] 
\text{cache fetch time}

\[ M_R \] 
\text{memory read time}

\[ M_W \] 
\text{memory write time}

\[ f \] 
\text{cycles per second}

REFERENCES


[GRIMSR92] Grimsrud, K. et al., \textit{Estimation of Simulation Error Due to Trace Inaccuracies}, Brigham Young University, Nov 92, unpublished


