Journal of Instruction-Level Parallelism 10 (2008) 1-33 Submitted 6/07; published 6/08

# Analyzing the Cost of a Cache Miss Using Pipeline Spectroscopy

**Thomas R. Puzak** TRPUZAK@US.IBM.COM
**A. Hartstein** AMH@US.IBM.COM
**P. G. Emma** PEMMA@US.IBM.COM
**V. Srinivasan** VIJI@US.IBM.COM
*IBM – T. J. Watson Research Center*
*PO Box 218*
*Yorktown Heights, NY 10598 USA*

**Arthur Nadus** NADAS@ENV.MED.NYU.EDU
*NYU School of Medicine*
*Old Forge Road*
*Tuxedo, NY 10987 USA*

## Abstract

We describe a new technique called Pipeline Spectroscopy that allows us to precisely measure the cost of each cache miss. The cost of a miss is displayed (graphed) as a histogram, which represents a precise readout showing a detailed visualization of the cost of each cache miss throughout all levels of the memory hierarchy. We call the graphs 'spectrograms' because they reveal certain signature characteristics of the processor's memory hierarchy, the pipeline, and the miss pattern itself. We show that in a memory hierarchy with N cache levels (L1, L2, ..., $L_N$, and memory) and a miss cluster of size C, there are $\binom{C+N}{C}$ possible miss penalties. This represent all possible sums from all possible combinations of the miss latencies with and without overlap from each level of the memory hierarchy (L2, L3, ... Memory) for a given cluster size. Additionally, a theory is presented that describes the shape of a spectrogram, and we use this theory to predict the shape of spectrograms for larger miss clusters. Next we provide to examples using spectroscopy to optimize the processor's hardware or application's software. The first example uses a miss spectrogram to improve the software design of an application. The second example uses a miss spectrogram to analyze bus queuing. Our experiments show that performance gains of up to 8% are possible. Detailed analysis of a spectrograph leads to much greater insight in pipeline dynamics, including effects due to prefetching, and miss queuing delays.

## 1. Introduction

In order to improve the performance of a processor or an application, designers have increased the amount of parallelism between the levels of the memory hierarchy. This area of research, termed memory-level-parallelism (MLP) has been explicitly studied in [1, 2, 3] while early studies focused on modeling and evaluating performance with ILP processors [4, 5, 6]. Chou, et al. [7, 8] studied several techniques (out-of-order, runahead, value prediction, prefetching, and store handling optimization) for increasing MLP in applications that are dominated by memory delays. They show that substantial amounts of performance gains are possible by increasing the MLP in these applications. Qureshi et al. [9] demonstrates that not all misses have the same cost, and measures miss parallelism to improve cache performance by altering the replacement

algorithm. In this paper, we build on this work by describing a new technique that quantitatively measures the cost of each cache miss throughout all levels of the memory hierarchy. A preliminary report describing this work was presented at the Computing Frontiers Conference [14].

We call this new technique 'pipeline spectroscopy' and the graphs representing the miss cost a 'spectrogram'. The graphs are called spectrograms because they reveal certain signature features of the processor's memory hierarchy, the pipeline, and the miss pattern itself (e.g. amount of overlap between misses in the miss cluster). Using pipeline spectroscopy, we are able to quantify the cost of each cache miss and measure the rate that these misses are satisfied from the different levels of the memory hierarchy (L2, L3, ..., Memory). This quantification leads to a much greater understanding of the amount of parallelism or overlap (miss cost) that the micro architecture and application allow. Armed with this information designers can individually analyze each miss and improve the performance of the hardware or software.

Several mechanisms that measure pipeline stalls and miss costs are described in the patent and publication literature [10-19]. Each technique describes the difficulty in determining an accurate measure for the cost of the miss and relies on hardware monitors to count events (cycles) that indicate when the decoder or execution unit are delayed (stalled) while waiting for an operand (data) to estimate this cost. However, not all of these events contribute to the loss of performance in a program. Today's processors have superscalar capabilities and parallel execution paths, and a delay suffered in one component of a processor can be overlapped with other events to mask any performance loss. For example, consider two events occurring in parallel: a branch misprediction and a cache miss. Simply counting the number of cycles an instruction (in the decoder or execution unit) is stalled waiting on a miss is not an accurate measure of the cost of the miss since many of the stall cycles are already overlapped with the delays caused by the branch misprediction.

In our work, we apply pipeline spectroscopy to produce a cache miss spectrogram which represents a precise readout showing a detailed histogram (visualization) of the cost of each cache miss through out all levels of the memory hierarchy, with and without overlap. We find this visualization very helpful. It allows us to directly see the amount of parallelism (overlap) between two or more misses, and analyze cache and pipeline performance (prefetching algorithms, miss clustering, and bus queuing). Cache miss spectrograms are produced by comparing instruction sequences and execution times that occurred near a miss in a 'finite cache' simulation run to the same set of instructions and their execution times in an 'infinite cache' run.

Next, we present a theory that describes the underlying shapes visualized in a miss spectrogram. The theory is based on observations and probability distributions drawn from the miss clusters of size 1 and 2, and is able to predict the miss patterns that will occur in larger miss clusters. By understanding the forces that contribute to the size and shape of the spectrogram, we hope to gain insight into pipeline dynamics, program structures, and techniques that can be used to improve the hardware and software.

The rest of this paper is organized as follows: Section 2 contains definitions and terminology. Section 3 describes constructing a miss spectrogram. The simulation model is described in Section 4. In Section 5 we measure the cost of a data miss. In Section 6, we make several observations regarding the properties of a spectrogram, which leads to developing a theory that describes the miss patterns, described in Section 7. In Section 8, we show simulation results for the theory. Section 9 describes an instruction miss spectrogram, and Section 10 shows how bus

queuing changes the shape of a spectrogram. Summary and conclusions are discussed in Section 11.

## 2. Performance Terminology

The overall methodology used to calculate the cost of a miss and the visualization process are explained as a prelude to analyzing a miss spectrogram. First, the definitions and formulas used to calculate the cost of a miss are described, then a description is set forth relative to how misses cluster and affect the standard operation of a high performance processor, followed by a description of the visualization process.

The most commonly used metric for processor performance is, "Cycles Per Instruction" (CPI). The overall CPI for a processor has two components: an "infinite cache" ($CPI_{INF}$) component and a "finite cache adder" ($CPI_{FCA}$).

$$CPI_{OVERALL} = CPI_{INF} + CPI_{FCA} \quad (1)$$

$CPI_{INF}$ represents the performance of the processor in the absence of misses (cache, and TLB). It is the limiting case in which the processor has a first-level cache that is infinitely large and is a measure of the performance of the processor's organization with the memory hierarchy removed. $CPI_{FCA}$ accounts for the delay due to cache misses and is used to measure the effectiveness of the memory hierarchy.

Just as processor performance (for both in-order and out-of-order machines) can be expressed in terms of a CPI, the "memory adder" can be expressed as the product of an event rate (specifically, the miss rate), and the average delay per event (cycles lost per miss):

$$CPI_{FCA} = \left(\frac{Misses}{Instruction}\right)\left(\frac{Cycles}{Miss}\right) \quad (2)$$

Substituting for $CPI_{FCA}$ in (1), the overall performance for a processor can be expressed as:

$$CPI_{OVERALL} = CPI_{INF} + \left(\frac{Misses}{Instruction}\right)\left(\frac{Cycles}{Miss}\right) \quad (3)$$

Solving for the average cost of a cache miss, we have:

$$\left(\frac{Cycles}{Miss}\right) = (CPI_{OVERALL} - CPI_{INF})\left(\frac{Instructions}{Miss}\right) \quad (4)$$

We use this formula to calculate the amount of time (cycles) a processor loses due to each cache miss. The following example illustrates calculating cycles per miss using equation (4). Consider an application whose entire run length is one million instructions and a processor where each cache miss is satisfied from the L2 that is 20 cycles away. If an infinite cache simulation run takes one million cycles ($CPI_{INF}$ = 1), and a finite cache simulation run takes 1.3 million cycles, then cache misses account for 300,000 cycles and the total $CPI = 1.3$ and $CPI_{FCA}$=.3. If the finite cache simulation run generates 25,000 misses, then $\left(\frac{Misses}{Instruction}\right) = \frac{25{,}000}{1{,}000{,}000} = \frac{1}{40}$ and $\left(\frac{Cycles}{Miss}\right) = \frac{300{,}000}{25{,}000} = 12.$ By applying this equation over the entire length of an application, the average cost for all misses can be calculated.

In the example above, we applied Eq. 4 macroscopically to calculate the average cost of a miss over the total run time of an application. However, Eq. 4 can also be used microscopically

to calculate the cost of a single miss. We will take a microscopic approach in using Eq. 4 to calculate the cost of each miss and produce a miss spectrogram. As presented in Section 5, the information contained in a miss spectrogram represent the cost of all misses throughout all levels of the memory hierarchy, including the amount of overlap (parallelism) achieved between any two misses.

## 3. Spectrogram Construction

A description of how misses can cluster and affect the performance of a processor is now described. Figure 1 shows the same five instructions executed as an 'infinite cache' sequence of instructions and a 'finite cache' sequence of instructions. In the finite cache sequence, the instruction decode times are shown in bold (without parenthesis) and instruction completion or EndOp times are shown in parenthesis. In the infinite cache run only the instruction decode times (in bold) are shown.

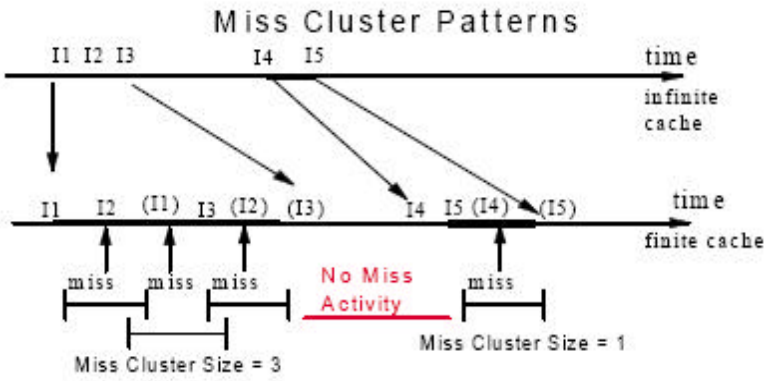


Figure 1: Miss cluster patterns for an application; miss clusters of sizes 1 and 3 are shown.

Associated with the finite cache run are two miss clusters, where a miss cluster is the maximal continuous interval of time characterized by at least one miss in progress during this time. The size of the miss cluster is the number of misses that started during this interval. In the finite cache run, the first miss cluster has three misses with overlap (size = 3) and the second miss cluster is size = 1 (a miss in isolation).

Next we describe a technique used to calculate the cost of the miss (cluster). The time to process the first miss cluster (in the finite cache run) is bounded by the decode time for instruction I1 and the EndOp time of I3, $(\text{I3EndOp - I1Decode})_{\text{Finite-Cache}}$ time. Instruction I1 (the decode time) represents the greatest lower bound of the miss cluster, while instruction I3 (EndOp time) is the least upper bound of the cluster. We call these points the infimum and supremum of the miss cluster. (By convention, the infimum of a miss cluster is the greatest instruction (time) that decoded just prior to the beginning of the first miss in the miss cluster and the supremum is the first instruction (time) that completed (EndOp) just after the last miss in the miss cluster finished.)

Similarly, the infimum instruction of the second miss cluster is I4 and the supremum is I5. The time to process the second miss cluster is then $(\text{I5EndOp - I4Decode})_{\text{Finite-Cache}}$. To calculate

the amount of delay associated with the first miss cluster we must subtract the amount of time to process the same set of instructions in an infinite cache run from the finite cache run. That is, $[(I3EndOp - I1Decode)_{Finite\text{-}Cache} - (I3EndOp - I1Decode)_{Infinite\ Cache}]$ equals the number of cycles the pipeline was stalled due the first miss cluster. Similarly, the amount of delay associated with the second miss cluster is $[(I5EndOp - I4Decode)_{Finite\text{-}Cache} - (I5EndOp - I4Decode)_{Infinite\ Cache}]$.

The reader will note that in the derivation above and in the equations presented in Section 2, no mention was made as to whether the processor is in-order or out-of-order. That is because out-of-order processing will not change the analysis. However, it may affect the manner in which the infinite cache running times for the sequence of instructions that surround a miss need to be determined. For example, if instruction processing is from an out-of-order processor, it may be necessary to save the sequence of instructions between the infimum and supremum of the miss (from the finite cache run) and use this same sequence of instructions (and their order) while determining the infinite cache run time.

By applying the above technique repeatedly, we can calculate the cost of a miss or miss cluster for either in-order or out-of-order processors, one cluster at a time. In the example above, I1, I2, and I3 can even be from three different threads running on a multithreaded processor (or three out-of-order instructions), but as long as the same three instructions (and their order) are used to determine the infinite cache run time, the cost of the miss cluster can be determined.

There are certain boundary conditions that must be considered when determining the infimum and supremum of a miss cluster. For example, the infimum of a miss cluster can only be established after the supremum of the previous miss cluster has been determined. This ensures that one miss cluster is terminated before another starts. If the upper and lower bounds of a miss cluster cannot be uniquely established, the two adjoining miss clusters are combined into a large miss cluster.

Also, when determining the infinite cache running time for an instruction sequence that occurred during a miss cluster, it may be necessary to prime the processor's pipeline with some of the instructions that occurred prior to the infimum instruction. This ensures that the correct execution and EndOp times of the infimum instruction are preserved as it passes through the processor's pipeline. By grouping misses according to their cluster size and calculating the delay associated with a miss cluster (number of stall cycles) using the method described above, the amount of time a processor loses due to cache misses is produced.

## 4. Simulation Methodology

To date, pipeline spectroscopy has been implemented in three proprietary processor simulators. Each simulator has produced results similar to those shown in Sections 5 and 9 below. Each simulator is cycle accurate and has been thoroughly validated against existing hardware. The processor model used in this paper is shown in Figure 2 and described in [20, 21], is a 4 issue superscalar, with address generation and cache access an independent out-of-order process. We use trace tapes produced for the IBM zSeries processor family. Instructions that typically produce addresses (L, LA, BXLE, SLL, SRL ...) are pre-executed after the decode stage of the pipeline to avoid future pipeline stalls due to address interlocks. Loads are executed as soon as the datum fetched returns from the cache and the results are forwarded to all dependent instructions. The instruction window was set at 32 entries. Separate L1 instruction and data

caches were modeled at 64 KB, the L2 size varies from 256K to 1 MB, and the L3 (when modeled) was varied from 1 MB to 4 MB. This processor model was chosen to illustrate the technique used to construct a spectrogram, and does not represent any existing or planned processor design. In our initial studies, Endop and those instructions not pre-executed after the decode stage are completed and executed in-order. Future work is planned to measure the benefits of prefetching, SMT, and SMT with out-of-order execution.

In order to stress different levels of the memory hierarchy (L1, L2 or L3), we use applications with large instruction and data footprints capable of stressing caches up to 4 Megabytes. Typically, commercial database applications have these characteristics [7]. In our study, we use eight workloads drawn from database workloads, SPEC 2000, and a C++ application. We use three proprietary commercial database applications running on zSeries servers, (oltp, and oltp2, and oltp3 described in [20, 21]); mcf, gcc, and perl from SPEC 2000, SPECjbb 2000, and perf1 a large-processor simulator written in C++. Typically, trace tape lengths are 5 to 100 million instructions. The simulation environment can handle the entire SPEC suite; the application subset used for this work was chosen for its ability to stress L2 and even L3 cache usage.

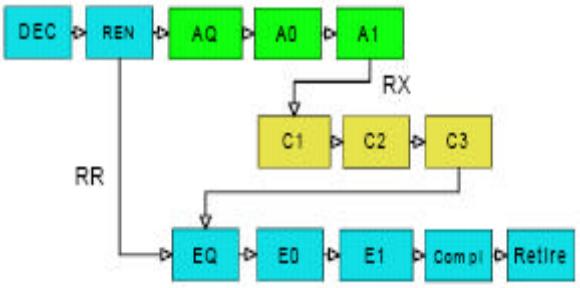


Figure 2: Pipeline modeled for study. Stages include: Decode, Rename, Agen Q, Agen, Cache Access, Execute Q, Execute, Completion, and Retire.

## 5. Spectrogram for Data Misses

In order to examine the miss spectrogram for data misses alone, we model an infinite or perfect instruction L1 cache, and set the data L1 cache to 64KB. The L2 is set to 256KB with 15 cycle latency, and set L3 latency to 100 cycles. All L2 misses are resolved in the L3. The line size and bus width are set at 128 bytes. No data prefetching was modeled. In fact, data prefetching is very difficult for many of these applications. However, prefetching will be explored when we examine the instruction miss spectrogram. Using the techniques described above, Figure 3 shows the miss spectrogram for the oltp workload. The overall hit ratio of the L2 was approximately 50%.

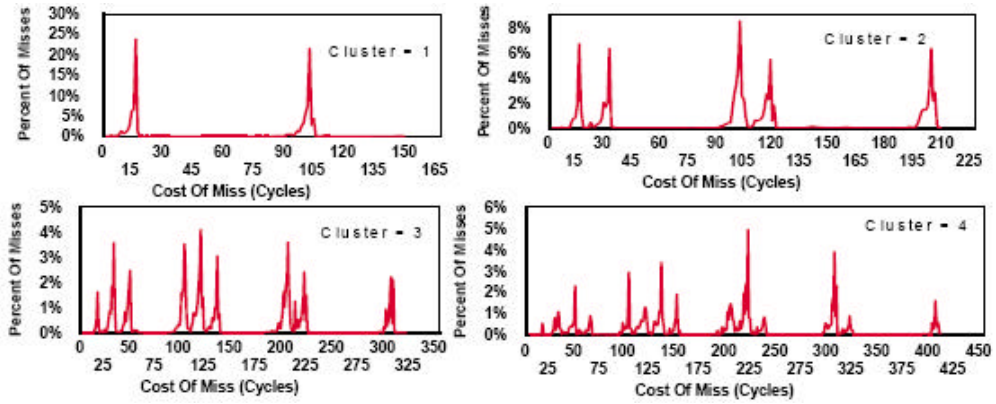


Figure 3: Miss Spectrogram for OLTP, L1 = 64K, L2 = 256K, Infinite L3; 15-cycle L2, 100-cycle L3.

The miss spectrograms for cluster sizes = 1, 2, 3, and 4 are shown. The X axis represents the cost of the miss for the specified cluster size. The Y axis shows the percent of misses that had that delay.

The cluster =1 plot (in Figure 3) shows two peaks. The first peak is centered near 15 cycles (the L2 hit latency), and the second peak is near 100 cycles (the L3 hit latency). The area under each peak is approximately the percentage of L1 misses resolved in the corresponding level of the memory hierarchy (i.e., the hit rates for the L2 and L3, or 50% in each).

The cluster size = 2 plot shows peaks at 15 and 30, 100 and 115, and 200 cycles. Notice that each peak represents the cost of a miss cluster (two L1 misses) and identifies one of all possible hit/miss combinations and all possible overlap/no-overlap patterns that the L1 misses had in the L2 or L3. Additionally, each peak identifies the degree of overlap/no-overlap (parallelism) between the two misses. Calculating the area under each peak, we see that the five miss costs (15, 30, 100, 115, and 200 cycles) have a probability of .138, .168, .288, .191, and .215, respectively. The peaks at 15 and 30 represent two L1 misses that both hit in the L2 but highlight two distinctively different outcomes. In the first case (peak at 15), both misses had a high degree of overlap (parallelism) and the overall cost was approximately the L2 hit latency while in the second case there was little overlap and the cost of the miss cluster was the sum of two L2 hits.

The peak at 100, again identifies two misses that were overlapped (had a high degree of MLP). Whether it was two misses that hit in the L3, one miss that hit in the L2 and one that hit the L3, the overall cost of the misses in the cluster was just the L3 hit latency.

The peak at 115 identifies two misses that had little or no overlap. Here, one miss hit in the L2 and one miss hit in the L3 but the cost of the miss cluster was the sum of the individual miss latencies. Finally, the peak at 200 identifies two misses that were resolved in the L3 and there was little overlap.

The peaks in the cluster = 3 graph again represent all possible hit/miss combinations (with and without overlap) of length 3 using the two miss latencies (15, 100) for the L2, and L3. For example, the peaks at 15, 30, and 45 present three L2 hits where two misses were overlapped, one miss was overlapped or no miss was overlapped with the other misses in the cluster, respectively. However, the peak at 300 represents three L3 hits with little overlap. Obviously, three dependent

misses that are resolved in the L3 can cause this miss penalty. Finally, the peaks in the cluster = 4 graph show all of the hit/miss, overlap/no-overlap, combinations of length 4 using the miss latencies 15 and 100.

Each peak represents the amount of time the group of cache misses (cluster) stalled the pipeline. By summing the 'stall cycles' calculated for each miss cluster, we can reconstruct the finite-cache-adder for the entire run, one cluster at a time. In many cases this involves summing the delay associated with 10s of thousands to over 100,000 miss clusters. Using this technique, we have always been able to calculate the total finite-cache-adder to within 5% (one cluster at a time), and in many cases the error is less than 2%. This shows how accurately we can identify miss clusters and evaluate their costs.

Prefetching and bus delays can change the shape of the peaks in a spectrogram. Prefetching can broaden the left shoulder of any peak and show the degree that a prefetch is being issued in advance of the nominal miss penalty. Queuing and bus delays can increase the right shoulder of a peak, adding miss latency. The effects of prefetching, bus queuing, and changes in the space of a miss spectrogram for both instruction and data misses are analyzed and discussed Sections 9 and 10.

Figures 4 and 5 plot the number of misses and cycles per miss versus cluster size, respectively. Even though the maximum miss cluster for the run was well over 1000 misses, typically the average miss cluster size is much smaller. For example, over 80% of the misses occur to miss clusters of size 6 or less and over 40% of the misses are a miss in isolation. This was observed for most applications used in this study.

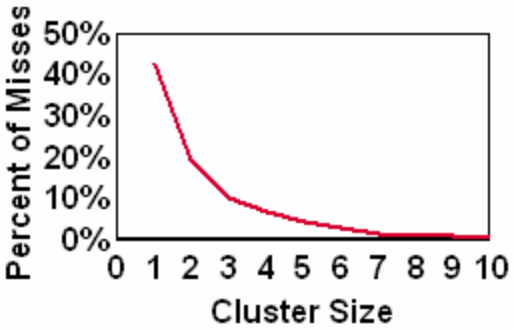


Figure 4: Misses by Cluster Size.

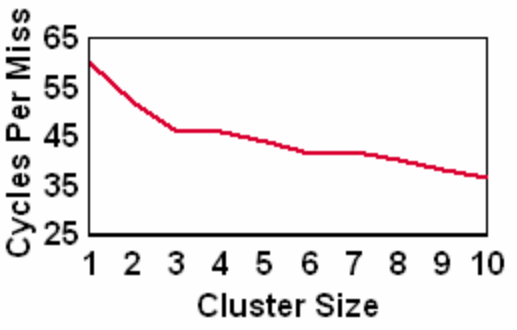


Figure 5: Cycles Per Miss Vs Cluster Size.

In Figure 5, notice how the average miss penalty decreases as the cluster size grows. The slope of the line indicates the degree that miss parallelism or miss overlap is occurring. Obviously, the greater the amount of miss overlap the greater the absolute value of the slope of the line. In this example the cost of a miss at a cluster size = 10 is approximately two thirds the cost of an isolated miss. This is far less than the potential for complete overlap.

Next, we repeat the above experiment but use the oltp2 workload and add an L3. We use the following memory hierarchy: data L1=64KB, L2=256KB 15 cycle latency, L3=1MB 75 cycle latency, and 300 cycle memory latency. Figure 6 shows the data miss spectrogram for cluster sizes = 1, 2, and 3.

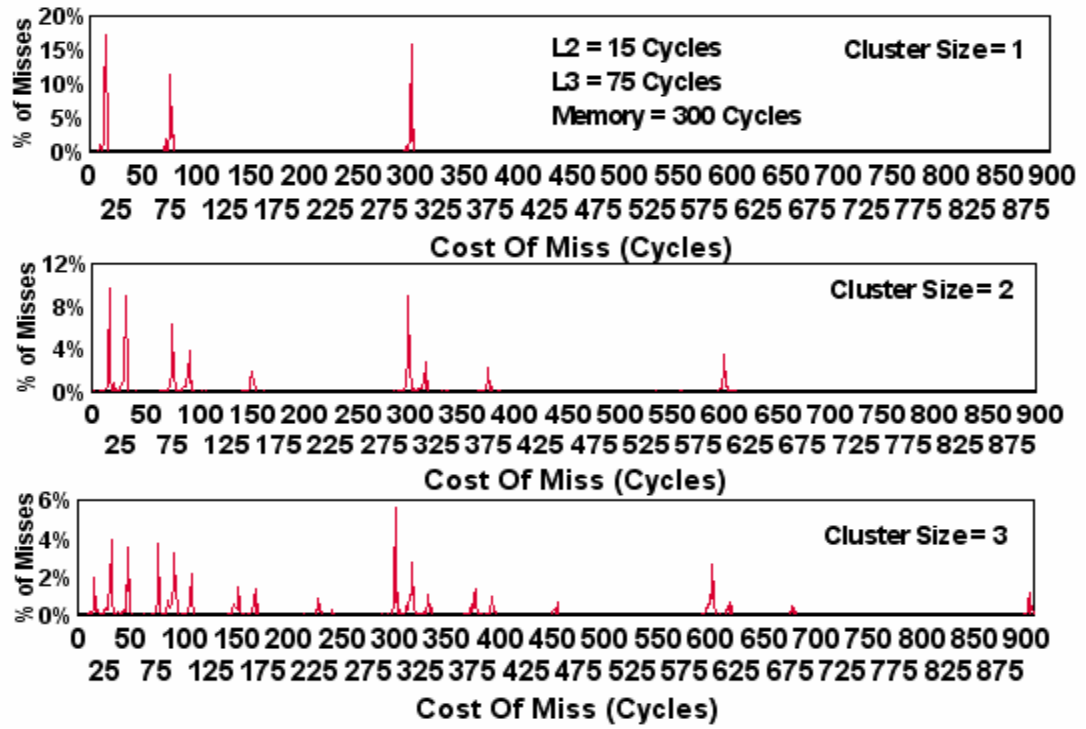


Figure 6: Data Miss Spectrogram for OLTP2, L1=64K, L2=256K, L3=1Meg.

Again, each peak in the miss spectrogram identifies a signature feature of the memory hierarchy: either a hit/miss combination of the miss cluster throughout all levels of the memory hierarchy or an overlap/no-overlap condition among the misses (amount of parallelism between the misses in the cluster).

The cluster = 1 plot shows three peaks. The first peak is centered near 15 cycles (the L2 hit latency), the second peak is near 75 cycles (the L3 hit latency) and the third peak at 300 cycles (the memory latency). Calculating the area under each peak is approximately the hit ratio (regarding L1 misses) for that level of the memory hierarchy (i.e. the hit ratios for the L2, L3, and memory). Examining the plots for cluster sizes equal 2 and 3, we see that they show all of the hit/miss, overlap/no-overlap combinations for a miss cluster of size 2 and 3 using the 3 miss latencies: 15, 75 and 300. Obviously the peak at 15 for the cluster size = 3 indicates a great deal of overlap among the three misses. However, the peak at 390 (for cluster size =3) indicates very little overlap among the three misses. Here, the three miss cluster had one miss hit in the L2, one hit in the L3, and one went all the way to the memory but the total miss penalty for the whole cluster is the sum of the individual miss latencies (15, 75, and 300). Similar hit/miss patterns and overlap/no-overlap conclusions can be drawn for examining any peak in the miss spectrogram.

We will refer to spectrograms like the one shown in Figures 3 and 6 as the 'canonical' representation for the cost of a miss in a multilevel memory hierarchy. It is a canonical form because it represents the most general form (combinations) of the miss patterns in a memory hierarchy. Obviously, prefetching and bus queuing can alter the miss patterns, and costs. By including the possibility of a peak at zero, a miss spectrogram can have all possible sums from all

possible combinations of the miss latencies from each level of the memory hierarchy for a given cluster size. We show in Appendix A that for a memory hierarchy with N cache levels (L1, L2, L3,..., $L_N$, memory) and a miss cluster of size C, there are

$$\binom{C+N}{C} \tag{5}$$

possible penalties (peaks) that characterize the canonical form hit/miss and overlap patterns. A peak at zero has the physical meaning that a miss or cluster of misses has zero delay. Prefetches, if issued far enough in advance of their use, speculative misses that do not interfere with any other cache accesses, or unused prefetches have the possibility of causing zero delay. Using Eq. 5 and the memory hierarchy described in Figure 6, we see that plotting miss clusters of size 4, 5, and 6 could have 35, 56, and 84 peaks, respectively.

## 6. Observations

Before we develop a theory that describes the miss patterns of a spectrogram it is necessary to understand the ways hits and misses form clusters. We start by defining a notation, and then examine the individual probabilities for each hit/miss combination.

Let random variable $X_i$ denote the event that the $i_{th}$ L1 miss in a miss cluster of size N is a hit or miss in the L2. Then, the sequence HM in a miss cluster of size 2 (a hit followed by a miss) can be expressed as the pair $X_1X_2$ where $X_1$=H and $X_2$=M and has the conditional probability $Pr[X_2=M \mid X_1=H]$.

Let T be the total number of L1 misses and let T=M+H where M and H are total number of hits and misses in the L2. Let $p$ represent the probability of a miss in the L2, then Pr[M]=p=

M/(H+M)=M/T and the probability of a hit is Pr[H] = (1-p). Then, the probability of a miss in the ith cluster is $p_i = \frac{M_i}{M_i+H_i} = \frac{M_i}{T_i}$.

The L2 miss ratio p is related to the miss ratios of the individual clusters in the following manner,

$$p=(M/T)= \frac{\sum M_i}{T} = \frac{M_1}{T_1}\left(\frac{T_1}{T}\right) + \frac{M_2}{T_2}\left(\frac{T_2}{T}\right) + \ldots \frac{M_N}{T_N}\left(\frac{T_N}{T}\right)$$

$$= p_1\left(\frac{T_1}{T}\right) + p_2\left(\frac{T_2}{T}\right) + p_3\left(\frac{T_3}{T}\right) + \ldots p_N\left(\frac{T_N}{T}\right)$$

that is, the miss ratio is the weighted sum of the miss ratios for each miss cluster size, where the weights are the probabilities of a hit or miss to a given cluster size. We refer to P as the global miss ratio for the L2 and $P_i$ as the local miss ratio relative to all of the misses that make up cluster size i. Typically, each local miss ratio is within 3 percent of the global miss ratio.

We now examine the individual probabilities for each hit/miss combination that occurs within a miss cluster. We use the spectrogram shown in Figure 3. Recall, in this experiment we set the L1 and L2 cache sizes to 64KB and 256KB to produce a L2 hit/miss ratio as close to 50% as possible. The actual hit/miss ratios were .493 and .507, respectively.

The cluster = 1 spectrogram comes from a miss in isolation that either hit or missed in the L2. Each of these events has its own (unique) spectrogram. We show these in Figure 7. We see that the hits have values around 15 and the misses have values around 100 cycles. The individual hit/miss probabilities are $P_1$=Pr[$X_1$=M]=.519 and Pr[$X_1$=H]=.481.

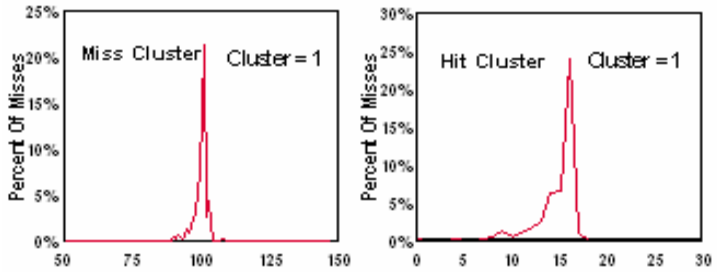


Figure 7: Individual Miss Spectrogram for Cluster Size = 1, L1=64KB, L2=256KB, 15 cycle latency for L2 Hit, L3 = 100 cycle latency for L2 Miss, Data for OLTP.

There are four different hit/miss combinations that make up the cluster = 2 spectrogram shown in Figure 3. The random variable pair $X_1X_2$ can either be a HH, MM, HM, or MH. Each of these hit/miss combinations have their own spectrograms. These are shown in Figure 8 along with their overlap/no-overlap probabilities. The overlap, no-overlap probabilities were obtained by calculating (integrating) the area under each of the two peaks in the graph. Each spectrogram describes a different probability distribution for the cost of the miss cluster. For example, the only spectrogram that has penalties near 15 and 30 cycles is the HH spectrogram. The peak at 15 identifies two L1 misses (L2 hits) that were almost entirely overlapped while the peak at 30 indicates two dependent misses with nearly no overlap. Calculating the area under each peak determined the probability of overlap and no overlap. The HH spectrogram had a miss overlap probability of .45 and .55 no overlap probability.

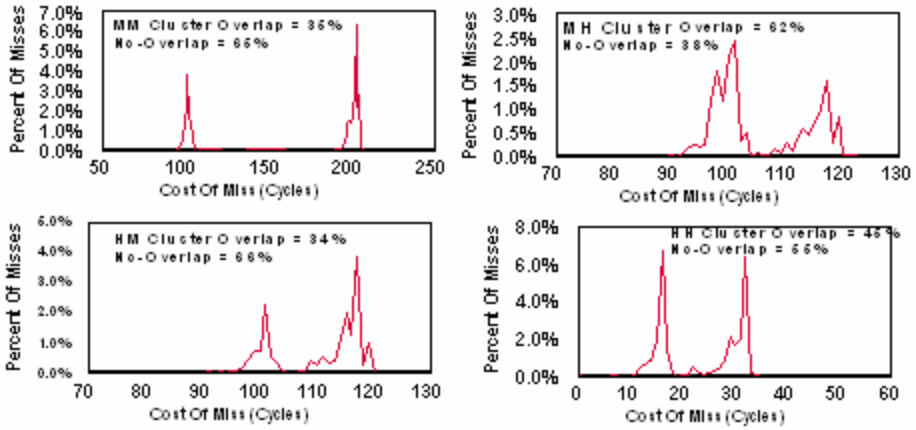


Figure 8: Cluster Size = 2 Individual Miss Spectrograms, L1=64KB, L2=256KB, 15 cycle latency, L3 = 100 cycle latency, OLTP.

The HM and MH spectrograms each have miss probabilities centered near 100 and 115. Again, the peak near 100 indicates two misses with overlap (one was a L2 hit and one was a L2

miss), while the peak at 115 indicates two misses with little overlap (the miss penalty is the sum of a L2 hit and L2 miss). The MM spectrogram has two peaks: one near 100, one near 200. The miss penalty near 100 indicates both misses were overlapped, while the peak near 200 (only in the MM spectrogram) indicates very little overlap. Combining each of four hit/miss spectrogram produces the cluster = 2 spectrogram shown in Figure 3.

The local miss probability was $P_2$ = .512 and the four hit/miss combinations had the following probabilities:

$$MM \sim \Pr[X_1, X_2 = MM] = .331, \quad HM \sim \Pr[X_1, X_2 = HM] = .188$$
$$MH \sim \Pr[X_1, X_2 = MH] = .175, \quad HH \sim \Pr[X_1, X_2 = HH] = .306$$

Examining the probabilities more closely, we see that the probability of a hit following a hit (HH) and a miss following a miss (MM) are much higher than independent events would predict. If the probability of a hit or miss were independent, each of the four hit/miss combinations would have a probability close to .25. For example, if they were independent we could use the binomial distribution to describe the probability of having k L2 misses out of N L1 misses as $\binom{N}{k} p^k (1-p)^{N-k}$ and then $\Pr[X_1 = M \cap X_2 = M] = \Pr[X_1 = M] \Pr[X_2 = M] = .5 \times .5 = .25$. However, this is not the case. We observe that the MM and HH events have a much higher probability than the MH and HM probabilities. Thus, in the random variable pair $X_1X_2$, $X_2$ is strongly correlated with $X_1$ .

This observation fits nicely with the spatial locality properties of a cache and Denning's [22] working set model for program behavior. If $X_1$ is an L2 hit, then $X_2$ is likely to be a hit (indicating the L2 contains the working set of the application). However, if $X_1$ is an L2 miss, then $X_2$ is likely to miss (indicating the application is changing working sets).

It is possible to construct the cluster = 2 spectrogram (from Figure 3) from the four hit/miss probabilities and their respective overlap, no-overlap probabilities. Figure 9 illustrates this process as a tree. The figure shows events $X_1X_2$ with its four hit/miss combinations and overlap/no-overlap probabilities. The values representing the area of each peak in the cluster size = 2 spectrogram are produced by multiplying the corresponding four hit/miss probabilities (.306, .188, .175, .331) by the appropriate overlap/no-overlap probability at the appropriate level shown in Figure 8. Note, the four hit/miss overlap/no-overlap probabilities were draw from the cluster size = 2 hit/miss combinations.

For example, the top of the tree shows $X_1$. The right fork of the tree represents a miss and the left fork a hit. The second level of the tree shows $X_2$ and its hit/miss possibilities. The four hit/miss combinations along with their probabilities are shown on the third level of the tree. The overlap, no-overlap probabilities are shown as branches off of each hit/miss possibility. The cost of each miss cluster is shown as a leaf node of the tree. Note, there are three events that have a possible 100 cycle miss penalty: the HM with overlap, the MH with overlap, and the MM with overlap. Summing all events with the same cost produces the probability distribution representing the cluster = 2 spectrogram shown in Figure 3.

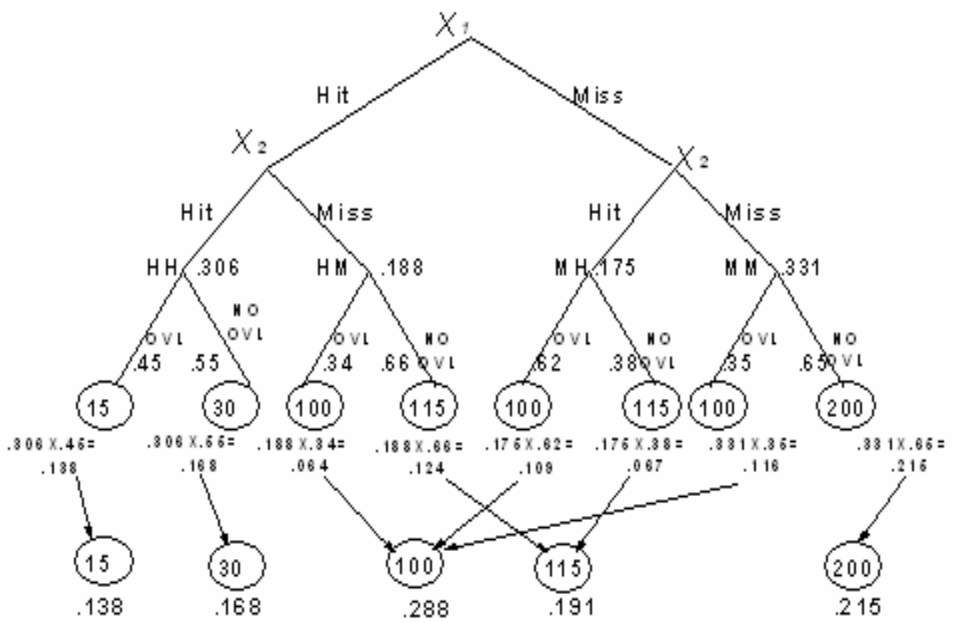


Figure 9: Miss Spectrogram Construction. HH, MM, HM, MH, and overlap probabilities for Cluster = 2 Spectrogram. Data is for the OLTP workload.

Returning to Figure 3, the cluster = 3 has eight possible hit/miss patterns (MMM, MMH, MHM, MHH, HMM, HMH, HHM, HHH). Thus a miss cluster of size N has $2^N$ hit/miss combinations. The local miss probability was $P_3 = .502$ and the eight hit/miss combinations had probabilities:

$$MMM \sim \Pr[X_1, X_2 X_3 = MMM] = .219,\ MMH \sim \Pr[X_1, X_2 X_3 = MMH] = .083$$
$$MHM \sim \Pr[X_1, X_2 X_3 = MHM] = .094,\ MHH \sim \Pr[X_1, X_2 X_3 = MHH] = .103$$
$$HMM \sim \Pr[X_1, X_2 X_3 = HMM] = .098,\ HMH \sim \Pr[X_1, X_2 X_3 = HMH] = .090$$
$$HHM \sim \Pr[X_1, X_2 X_3 = HHM] = .105,\ HHH \sim \Pr[X_1, X_2 X_3 = HHH] = .208$$

Again we see that the events $X_1X_2X_3$ (in a miss cluster of size 3) are not independent. The probabilities for the HHH and MMM events have over twice the probability of the other hit/miss combinations. Each of the eight hit/miss combinations has their own individual spectrogram. Because of space limitations, we only present the spectrograms for the MMM, HHH, MHM, and HMH combinations along with their overlap/no-overlap probabilities in Figure 10. Each graph has three or four peaks, indicating the overlap/no-overlap probabilities of the hit/miss pattern.

The HHH graph has peaks near 15, 30 and 45 indicating the degree of overlap between the 3 L2 hits (12, 48, and 40 percent). The sum of all eight individual spectrograms produces the nine peak (cluster = 3) spectrogram shown in Figure 3.

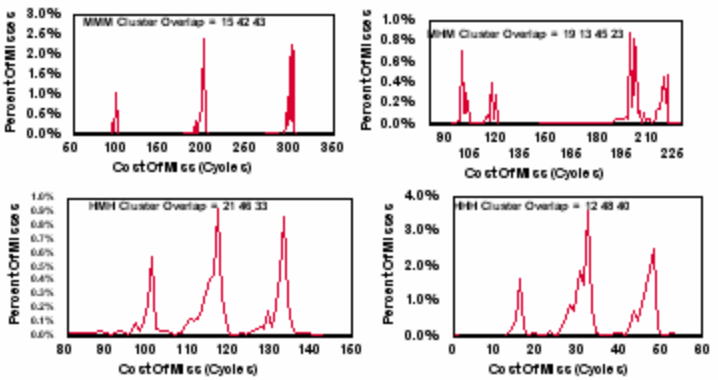


Figure 10: Cluster Size = 3 Individual Miss Spectrogram, L2 = 15 Cycle, L3 = 100 Cycles Latency. Data for OLTP.

Summarizing the above experiments, we observe that in a cluster of N misses, there are $2^N$ possible hit/miss combinations. Also, hit/miss patterns are not independent. In the miss sequence $X_1X_2X_{3,\ldots,}$ $X_N$, the probability that $X_{i+1}$ has the same outcome as $X_i$ is much higher than independent miss probabilities. Finally, a miss spectrogram can be reconstructed by knowing the individual probabilities of each of the $2^N$ hit/miss combinations and their associated overlap/no-overlap probabilities. In the next section we create a model for predicting the miss costs and associated probabilities found in a spectrogram.

## 7. Theory

In this section we develop a theory that allows us to predict the size and shape of a spectrogram. The theory draws on six parameters: the overall L2 miss ratio, a correlation parameter describing the probability of a hit following a hit or miss following a miss in the L2, and the four overlap/no-overlap probabilities calculated from the four hit/miss combinations found in a cluster size =2 miss pattern. Knowing these parameters (mostly drawn from cluster size = 2 miss pattern) we can predict the patterns found in spectrograms for miss clusters of size 3, 4, and 5.

In order to predict the miss costs and associated probabilities found in a spectrogram, we describe a two step process. The first step involves predicting the probabilities for the individual $2^N$ hit/miss combinations that make up a cluster of size N. Recall, it was shown that these probabilities are dependent. The second step involves determining the overlap/no-overlap probabilities for adjacent misses in a cluster size = 2 miss pattern. Recall, there are four of them. A miss spectrogram is produced from the product of these two probabilities, then combining alike terms (cost).

We start by developing a model that increases (boosts) the probability of a miss following a miss (or hit following a hit). Let Pr[M] = p and Pr[H] = (1-p), we seek a function that increases the probability of a MM being greater than the product of p?p. Consider the random pair $X_iX_{i+1}$,

we need to define transition probabilities that describe the four hit/miss combinations in a cluster of size 2. To increase $Pr[X_2=M|X_1=M]$, we define $\alpha$ (alpha), $0 \le \alpha \le 1$ and form a convex combination between the values of p and 1 such that $Pr[X_2=M|X_1=M] = \alpha p + (1-\alpha) = (1-\alpha+\alpha p)$.

Note that the function $(1-\alpha+\alpha p)$ has the properties we desire. For $0 \le \alpha \le 1$ and $p \le 1$, $(1-\alpha+\alpha p) \ge p$. In Appendix B, we show $\alpha = 1-\rho$ where $\rho$ (rho) is the correlation coefficient between $X_i$ and $X_{i+1}$. As $\rho \to 1$ (the correlation between $X_i$ and $X_{i+1}$ becomes stronger) $\alpha \to 0$, and the $Pr[X_2=M|X_1=M] \to 1$ signifying that the probability of a miss following a miss is 1 (perfect correlation). Similarly, as there becomes less correlation between $X_iX_{i+1}$ then $\rho \to 0$ and $\alpha \to 1$ then $Pr[X_2=M|X_1=M] \to p$ signifying that the probability of a miss following a miss is p (as if they were independent). We will determine the value for $\alpha$ by fitting the individual hit/miss probabilities for the cluster = 2 data (spectrogram) and use this value of $\alpha$ to predict the hit/miss patterns for larger miss clusters.

The same technique is used to increase the probability of a hit following a hit. Again we use $\alpha$ (the same $\alpha$ used in boosting the probability of a miss) and define a convex combination between (1-p) and 1 such that $Pr[X_2=H|X_1=H] = (1-p)\alpha + (1-\alpha) = (1-\alpha p)$. This also has the properties we desire, for $\alpha \ge 0$ and $p \le 1$, $(1-p) \le (1-\alpha p) \le 1$. Additionally, as $\rho \to 1$, $Pr[X_2=H|X_1=H] \to 1$ (perfect correlation) and as $\rho \to 0$, $Pr[X_2=H|X_1=H] \to p$ (as if they were independent). The remaining two hit/miss probabilities are defined using the HH and MM probabilities. That is, a MH has $Pr[X_2=H|X_1=M]=1- Pr[X_2=M|X_1=M] = \alpha-\alpha p$ and HM has $Pr[X_2=M|X_1=H] = 1- Pr[X_2=H|X_1=H]=\alpha p$.

Figure 11 shows each of the four hit/miss combinations as a state transition matrix. Each entry shows the transition probability for one of the four hit/miss combinations in the sequence $X_iX_{i+1}$. Using Figure 11, we see that a MM sequence has probability $p(1-\alpha+\alpha p)$, and a MH has probability $p(\alpha-\alpha p)$. Similarly a HM has probability $(1-p)(\alpha p)$, while a HH has probability $(1-p)(1-\alpha p)$. From the simple assumption used to define a single parameter $(\alpha)$ used to increase the HH or MM probabilities, we are able to generate the probabilities of larger hit/miss sequences (using Figure 11).

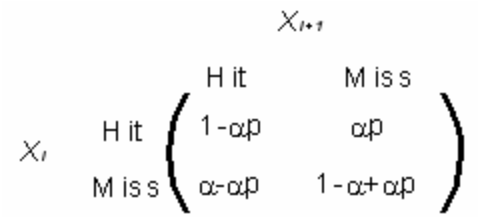


Figure 11: Probability State Transition Matrix for HH, MM, MH, and HM events.

For example, patterns HMM, MHHM, and HMMMH have probabilities $(1-p)(\alpha p)(1-\alpha+\alpha p)$, $p\ (\alpha-\alpha p)\ (1-\alpha p)\ (\alpha p)$, and $(1-p)(\alpha p)(1-\alpha+\alpha p)^2(\alpha-\alpha p)$, respectively. Also hit/miss probabilities of any length can be calculated using the generating function $(1-p)(1-\alpha+\alpha p)^w(\alpha-\alpha p)^x(1-\alpha p)^y(\alpha p)^z$, if the pattern begins with a hit, or if the sequence begins with a miss $p(1-\alpha+\alpha p)^w(\alpha-\alpha p)^x(1-\alpha p)^y(\alpha p)^z$. Note w, x, y and z then have the property w+x+y+z = C-1 where C is the length of the miss cluster.

## 8. Simulation Results

To test the robustness of the theory, each workload was simulated twice: first with a 256KB L2, and then with a 1MB L2. (Additionally, mcf, oltp, oltp2, and perf1 were simulated a third time using a 2MB L2 for 20 simulation runs in all.) The L1's were 64KB in all cases. The spectrograms, along with the probabilities for each hit/miss combination were produced for clusters 1 through 5. The value for $\alpha$ was determined by least square fit using the four hit/miss equations from the cluster size = 2 data. The global miss probability $p$ was used in all cases along with the hit/miss state transition defined in Figure 11. Using the oltp workload (shown in Figure 3), Table 1 shows the results of the fit. The best fit value for $\alpha$ = .73. As can be seen, the least

| Cluster = 2 | HH | HM | MH | MM |
|---|---|---|---|---|
| Real | .306 | .175 | .188 | .331 |
| Predicted | .308 | .182 | .182 | .328 |

Table 1: Hit/Miss Probabilities for Cluster = 2. Data is for the OLTP workload. L1=64KB, L2=256KB.

squares fit is very good. Next, $\alpha$ and the global miss ratio $p$ are used to predict the hit/miss probabilities for larger cluster sizes. Table 2 shows the eight hit/miss probabilities for cluster size = 3. As can be seen the agreement between theory and experiment is quite good. Next we simulated all eight workloads varying the L2 cache size. For each workload, we determined $\alpha$ by

| CL 3 | HHH | HHM | HMH | HMM | MHH | MHM | MMH | MMM |
|---|---|---|---|---|---|---|---|---|
| Real | .208 | .105 | .090 | .098 | .103 | .094 | .083 | .219 |
| Pred | .193 | .115 | .065 | .117 | .115 | .068 | .117 | .210 |

Table 2: Hit/Miss Probabilities for Cluster = 3. Data is for OLTP, L1=64KB, L2=256KB.

a least squares fit using the cluster = 2 data and then predicted the $2^N$ hit/miss probabilities for clusters 3, 4 and 5. We measured the error between our prediction and the true hit/miss probabilities using the Kolmogorov-Smirnov (K-S) [23] test, where the K-S test measures the maximum difference between two cumulative distributions $T_N(x)$ (the true distribution) and $P_N(x)$ (the predicted distribution), that is Max $|T_N(x) - P_N(x)|$ *over all x*. To represent each of the $2^N$ hit/miss probabilities as a distribution we ordered the probabilities from 0 to $2^N - 1$ based on a hit = 0 and a miss = 1. Thus, in a cluster size = 3, HHH=0 and MMM=7.

Figure 12 shows the distribution of the maximum error between 20 simulation runs. The cluster = 2 results are included to indicate the 'goodness' of the fit in determining $\alpha$. As can be

seen, the predictions are very good. Cluster = 3 results show nearly 75% of the simulation runs had less than 7% error. As the cluster size increases, the amount of error grows. However, even in the cluster = 5 predictions, over two thirds have less than a 10% maximum error.

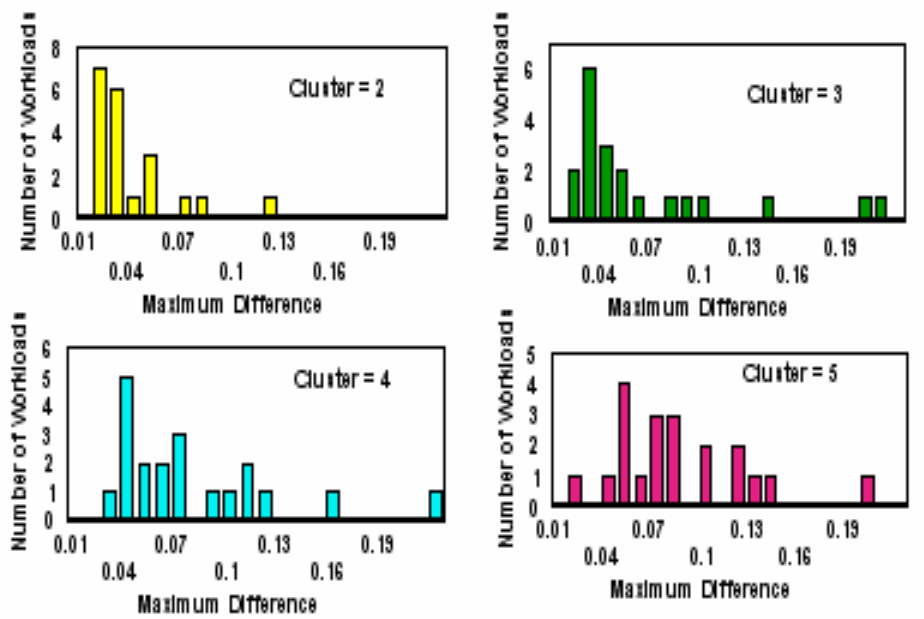


Figure 12: Kolmogorov-Smirnov test for predicted Hit/Miss probabilities for Clusters = 2, 3, 4, and 5.

Once the $2^N$ hit/miss probabilities are determined (predicted), next we need to determine the four (MM, HM, MH, and HH) overlap/no-overlap probabilities for the cluster = 2 case (as shown in Figure 8), then a spectrogram for a larger cluster can be constructed. Note this was done for each of the workloads and the 20 simulation runs. Intuitively, we believe that the same forces (program dependencies, micro architecture features) that determine the overlap probabilities for the cluster = 2 case will also be present and can also be used to determine the overlap probabilities in larger miss clusters.

Again, we simulated all workloads and determined the overlap/no-overlap probabilities from the cluster = 2 case. Next, we produced an overlap/no-overlap tree (similar to the one shown in Figure 9) for each of the $2^N$ hit/miss combinations for that miss cluster.

For example, Figure 13 shows the cluster size=3 trees for events $X_1X_2X_3$=HHH or a MMH sequence using values from the simulation run described in Figures 3, and Tables 1 and 2. The figure shows two trees. The top tree shows the transitions for the HHH miss pattern. At the root node, the probability for the HHH outcome is shown .193. The overlap/no-overlap probabilities are shown for events $X_1X_2$. A left branch indicates that the two misses are overlapped; a right branch indicates no overlap. These are represented by the .45 and .55 probabilities. The second level of the tree shows the overlap/no-overlap probabilities for $X_2X_3$ and again is represented by

.45 and .55 probabilities. Thus a miss cost of 15 cycles requires that miss events $X_1X_2$ are overlapped and miss events $X_2X_3$ are overlapped and $X_1$ is overlapped with $X_3$. The leaf nodes show the calculations for determining the probabilities (integral size) for the spectrogram peaks shown at miss costs 15, 30, and 45 cycles. Note, each node uses the predicted probability for the HHH pattern (.193), shown in Table 2.

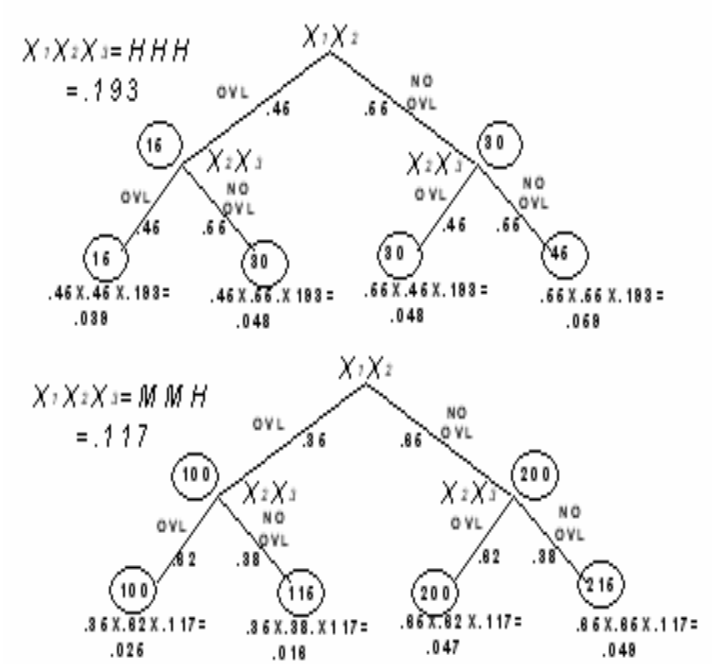


Figure 13: Overlap/No-Overlap Probability Tree for HHH and MMH. Data are for OLTP.

The bottom tree shows transition probabilities for the MMH pattern. Recall from Table 2, the MMH pattern has a probability of .117. The first two events in the miss pattern $X_1X_2$ use the MM overlap/no-overlap probabilities (.35 and .65), while next sequence $X_2X_3$ use the MH overlap/no-overlap probabilities (.62 and .38). Thus, the overlap/no-overlap probabilities for miss sequence $X_iX_{i+1}$ are drawn from their unique hit/miss pattern (HH, HM, MH, or MM) determined from the cluster size = 2 miss patterns. The leaf nodes show the calculations to produce miss penalties of 100, 115, 200 and 215 cycle miss penalties.

Using this technique on all of the $2^N$ hit/miss combinations with their predicted probabilities along with the four hit/miss overlap/non-overlap probabilities from a cluster size = 2, a spectrogram for a larger cluster is produced. Figure 14 shows our prediction for the cluster size = 3 and 4 spectrograms for the oltp workload shown in Figure 3. The red series with symbol marker is the measured spectrogram, while the blue series is our predicted spectrogram. Since our theory only predicts the area of a peak, we plot our predictions as a pulse under the true spectrogram for that cluster. The height of the peak is scaled to match the height of the true spectrogram according to the amount of error between our prediction and true measurement. For example, if there is a 10% error between our prediction and the true area under a peak, then there is a 10% difference between the heights of the peaks. Also, we have not attempted to calculate

the position of the peak, but instead use the miss latencies to generate their position, (i.e.. 15, 30 45, 100 ...). In future work we plan a theoretical treatment of the position for each peak.

As can be seen, the agreement between theory and experiment is very good. The difference between our prediction and the true probability of a miss pattern (the error) agrees closely with the errors produced in estimating the $2^N$ hit/miss probabilities shown in Figure 12.

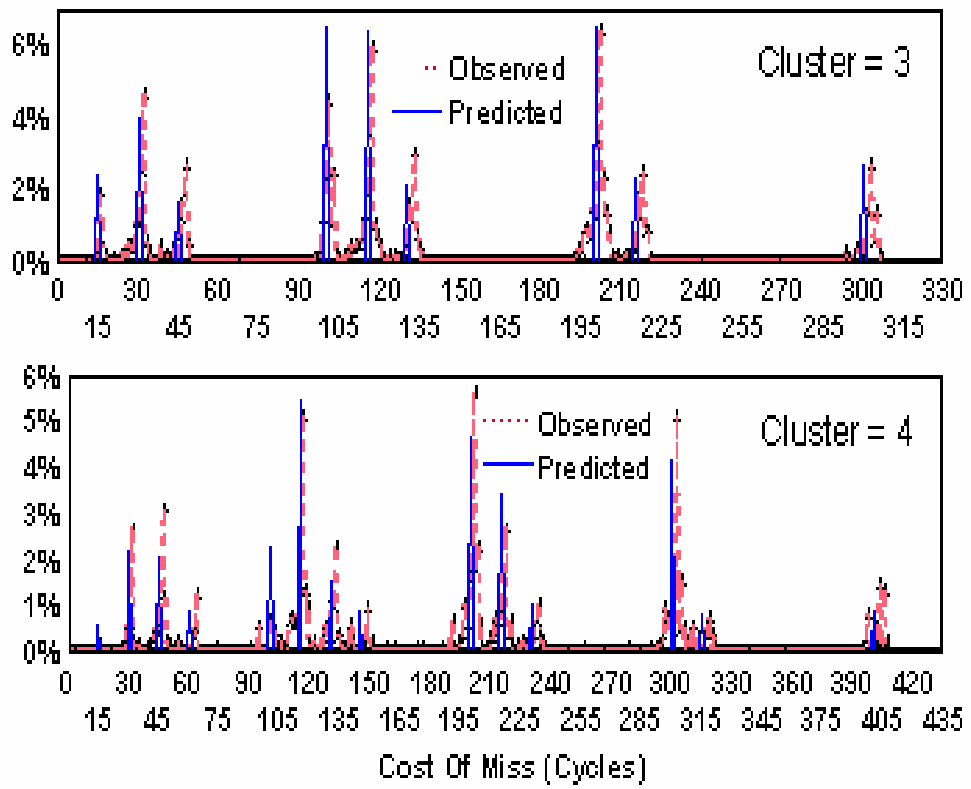


Figure 14: Observed and predicted Spectrogram for Clusters = 3 and 4. L1=64KB, L2=256KB, 15 cycle latency, L3 = 100 cycle latency.

## 9. Spectrogram Analysis

In this section we provide two examples that use the information displayed in a cache miss spectrogram to optimize the processor's hardware or application's software. The first experiment demonstrates how a miss spectrogram can aid software designers in analyzing the performance of an application. The second example uses a miss spectrogram to analyze hardware performance.

Each experiment highlights the effects that bus queuing has on performance. Bus queuing can occur whenever the linesize of the cache is greater than the bus width and multiple cycles are needed to transfer a line between levels of caches during a miss. We begin by defining the terms that will be used throughout this section. Bus queuing time is lengthened whenever the bandwidth between cache levels is decreased or the bus frequency ratio is increased.

The baseline processor organization and memory hierarchy are described in Table 4. The L2 is set at 512 KB with a 15 cycle latency with an infinite size L3 set at 75 cycles (i.e. all L2 misses are resolved at the L3, 75 cycles away). A 256 byte line size is modeled throughout all three levels of the memory hierarchy. We consider an on-chip L2 and configure the bus connecting the L1 and L2 to be able to move 32 bytes each processor cycle. Thus the LTI for an L2 hit is 8 cycles.

| Term | Definition |
|---|---|
| Bus Width | The number of bytes moved in or out of the cache per cycle during a miss. |
| Packet | Subsection of cache line. Equals size of bus width (in bytes). Multiple packets makeup a cache line |
| Packets Per Line | The ratio of cache linesize to bus width (packet size). This is also the number of bus cycles needed to move a line between levels of the memory hierarchy |
| Miss Latency | Number of cycles before first data (packet) returned after a cache mss is detected |
| Leading Edge penalty | Once a cache miss is detected, amount of time the processor stalls waiting for the first data (packet of line) to returned. |
| Trailing Edge Penalty | The observed cost of caching the remainder of the cache line after first packet is returned. The trailing-edge penalty would be zero if the entire line could be returned and installed in the cache in a single cycle. However, few designs can afford the implied luxuries of a line-wide data-return bus and cache-write capability. |
| Bus Frequency Ratio | The ratio of the processor frequency to the bus frequency. It is determined by logic speed, packaging and power limits. |
| Line Transfer Interval (LTI) | The number of processor cycles needed to move a cache line (Packets) on the bus and equals bus frequency ratio times the packets per line. |

Table 3 Terms and Definitions

Table 3: Terms and Definitions.

| | |
|---|---|
| L1 Cache Configuration | Split I and D each 64KB, 4 Way Set Associative 256 Byte Line 2 Cycle access |
| L2 Cache Configuration | Unified 1/2 Meg, 8-Way Set Associatvie, 256 byte Line, 15 cycle access |
| L3 Cache Configuration | 4 Meg, 8-Way Set accociative, 256 byte line, 75 cycle access |
| Processor Pipeline | 4 issue superscalar, out-of-order address generation and cache access, in-order execution, 32 entry instruction window |

Table 4: Baseline Processor Configuration.

We consider an off-chip L3 with a 32 byte bus and consider three different line transfer rates for an L3 hit: 32 bytes are transferred (from the L3 to the L2 and L1) every other cycle, every third cycle, or every fourth cycle. Thus it takes 16, 24, or 32 cycles to transfer a line from the L3 to the L2 and L1 (on a L3 hit).

Figure 15 shows a miss spectrogram for OLTP3 for cluster size= 1 using the three different LTIs for L3 hits. Immediately visible, in all three graphs, is the similarity in the peaks centered at 15 cycles (an L2 hit). Recall the L2 is on chip and all three processor configurations deliver 32 bytes every cycle for an L2 hit. Each graph has a small left shoulder (at 15 cycles) and shows some misses costing as low as 8 cycle (leading edge penalty). However, the right shoulder is much larger (in area) than the left shoulder with many misses costing more than the 15 cycle miss latency. Also notice, the peak at 75 cycles is different in the three graphs. The right shoulder

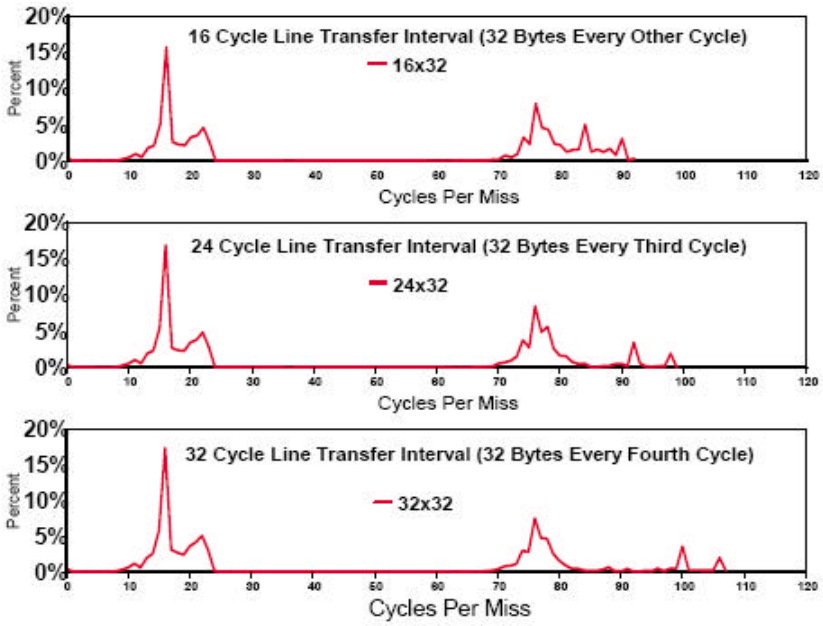


Figure 15: Cluster = 1 Miss Spectrogram for OLTP3 for 16, 24, and 32 cycle trailing edge.

grows as the line transfer interval grows from 16 cycles, 24 cycles, to 32 cycles. There are two very distinguishable sub peaks in the right shoulder of the three graphs. The spectrogram for a 16 cycle LTI has sub peaks at 84 and 90 cycles. Similarly the spectrograms for the 24 and 32 cycle LTIs have sub peaks at 92 and 98 cycles, and 100 and 108 cycles, respectively. Notice, as the line transfer intervals grows the sub peaks correspondingly shift eight cycles to the right.

In order to investigate the cause of these sub-peaks, we use two reports that are produced during a spectroscopy run: the Miss-Cost report and Cost-Analysis report[1]. Figure 16 gives an example of a small portion of the Miss-Cost report for the 24 cycle LTI spectrogram. It lists every miss by cluster size, cost, infimum and supremum instruction on the trace, miss address and instruction address that make up the miss spectrogram.

| Cluster Num | Size | Cost | Infimum Inst | Supremum Inst | Miss Address | Inst Address | Inst Number |
|---|---|---|---|---|---|---|---|
| 27973 | 1 | 98 | 348389 | 348417 | 20B1FA20 | 000668EC | 348389 |
| 27984 | 3 | 75 | 348421 | 348449 | 20C1FD20 | 00066EEC | 348421 |
| 27984 | 3 | 75 | 348421 | 348449 | 20C1FE40 | 00066EEC | 348433 |
| 27984 | 3 | 75 | 348421 | 348449 | 20C1FF00 | 00066EEC | 348441 |

Figure 16: Miss Cost Report for OLTP3.

Figure 17 shows the Cost-Analysis report. It summarizes all misses according to the instruction that generated the miss and sorts them by frequency (number of times the instruction generated a miss) and by cost (total cost of misses). The top five instructions generating the most

Highest count items:

| ASID | Inst Addr | Count | % of Total |
|---|---|---|---|
| 012E | 93DAA | 4410 | 0.49% |
| 0134 | 9CDAA | 3854 | 0.43% |
| 0146 | 3B46E | 3023 | 0.34% |
| 0146 | 9724EA6 | 2596 | 0.29% |
| 012E | 3CF3DD62 | 2444 | 0.27% |

Highest cost items:

| ASID | Inst Addr | Total Cost | % of Total |
|---|---|---|---|
| 0146 | 9724EA6 | 144045 | 0.33% |
| 0146 | 3BE10A3E | 142423 | 0.32% |
| 0146 | 3BE10A02 | 139804 | 0.32% |
| 0146 | 96F2A68 | 136345 | 0.31% |
| 012E | 93DAA | 125253 | 0.28% |

Figure 17: Cost Analysis Report for OLTP3.

[1]In the analysis that follows we investigate the cause of the sub peak at 98 cycles shown in the 24 cycle LTI spectrogram. The sub peak at 92 cycles is caused by a similar addressing pattern and discussions regarding its removal are omitted for brevity (but accomplished in the same manner).

misses (by frequency) along with their address-space-identifier (ASID) are shown on the left, while the top five instructions generating the most miss cost are shown on the right. Notice that the instruction positions change when sorted by frequency and then by cost. That is, only two of the instructions that produced the most misses (when sorted by frequency) are in the top five instructions that produced the most cost (when sorted by miss cost). Obviously, software tuning would focus on miss-penalty cost rather than miss frequency because most of the cost of a miss can be overlapped with another miss if it occurs in parallel with other misses. This was clearly highlighted in Figures 3 and 6 when examining cluster size 3 misses and observing miss penalty peaks at 15, 30, and 45 cycles.

Returning back to Figure 15, the file was sorted on cost and cluster size and misses that make up the sub peak at 98 cycles were identified. Once the infimum and supremum instructions on the trace are known, the simulation run is repeated and a detailed cycle-by-cycle instruction trace (report) is produced highlighting the events that occurred during the miss.

Examination of the cycle-by-cycle traces shows that the sub peaks were caused by two byte-compare instructions, two instructions apart. Figure 18 illustrates the cause for this cost. The miss address (generated by the first byte compare) was to the 6th 32 byte packet of a 256 byte line. This is the first packet that is returned on the bus when processing the miss. During the miss, packets are returned in increasing order (starting from the miss address) until the end of the line is reached, then processing continues to the beginning of the line until the entire line is transferred. Thus, the 8 packets that make up the line are returned in the following order: 6, 7, 8, 1, 2, 3, 4 and 5.

The next reference to the line (the second compare instruction, two instructions later) is 10 bytes prior to the miss address and depending on byte alignment is to the preceding 32 byte segment (the 5th packet) in the line. The targets of both compares are referenced using a common base register (data structure) and these two compare instructions always referenced two different bytes, 10 bytes apart. Since the packets in the line are returned in ascending order until the end-of-line is reached then wrapping around to the beginning of the line, the packet containing the second byte will not be available on the bus until the entire line is transferred to the cache (when they are not in the same packet). This accounts for the 8 cycle sub peak shift as the TLI is increased from 16, to 24, and 32 cycles. This instruction sequence occurred repeatedly on the trace (hundreds of times) and depending on byte alignment within a cache line would generate a miss cost of approximately 75 cycles (if both bytes were contained in the same 32 byte packet) or 98 cycles if the two bytes were in neighboring packets.

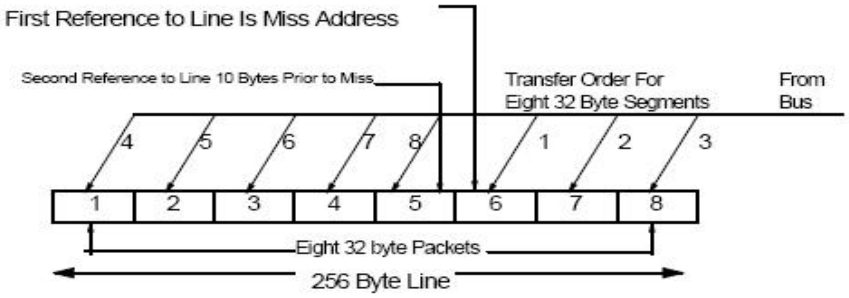


Figure 18: Miss Pattern for 96 cycle Cost per Miss for OLTP3.

Once the cause of the miss cost was identified a second simulation run was conducted to test whether the sub peaks can be removed. The instruction trace was modified and the two bytes compared by each instructions were switched. This can be accomplished (in reality) by switching the locations of the two bytes in the data structure. Now the instruction that compares the first byte in the line (the first reference to the line) is 10 bytes prior to the second byte compared (referenced) in the line. With these bytes switched, the first reference will cause a cache miss (as before) however the second reference to the line will be to a byte that is 10 bytes after the miss location and be available on the same cycle, if contained in the first packet returned during a miss, or one packet (transfer) later.

The simulator was rerun using the modified trace and data structure (MtDStr) and the spectrogram for the original run and modified data structure are shown in Figure 19. Most of the two spectrograms overlap along the X axis between miss values of 0 to 70 cycles. This should be expected because the L2 bussing strategy was not changed between the two simulation runs.

However, notice the change in the shape of the spectrogram between 75 cycles to 110 cycles is all three graphs. The sub peaks from the original run are gone and more misses now have a cost near 75 cycles, the miss latency for a L3 hit. The results show that swapping the locations of the bytes compared reduces the cost of the targeted misses and the sub peaks are removed. Recall, this instruction sequence was identified through the use of the miss spectrogram.

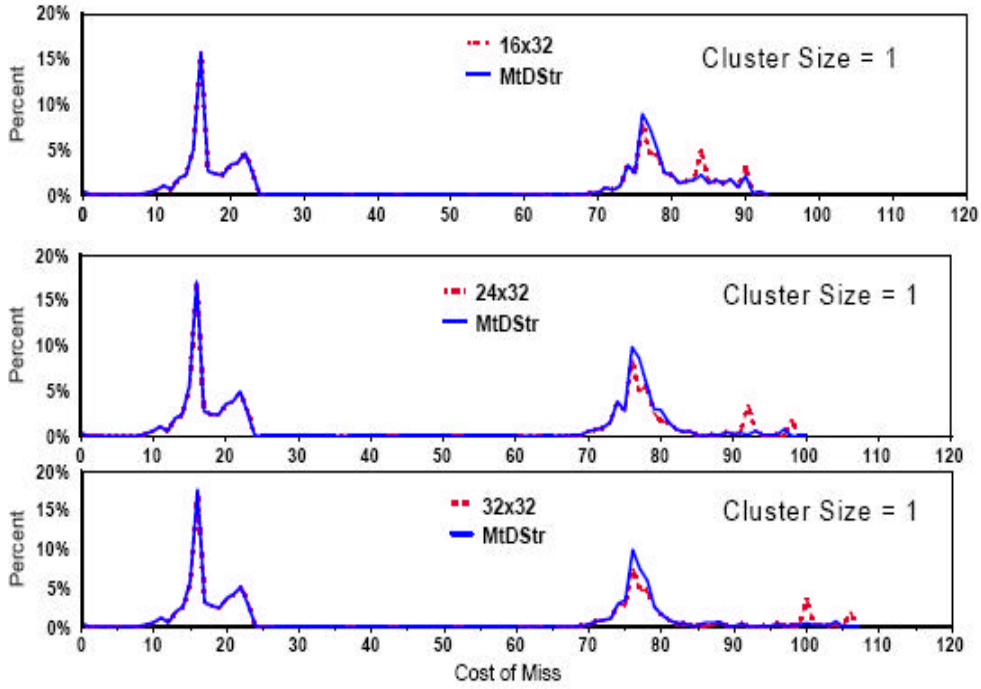


Figure 19: Spectrograms for OLTP3 with compare bytes switched.

We should offer a few comments regarding this experiment. First, we only changed the addresses generated (compared) by two instructions and the results are clearly visible in the corresponding spectrogram. The total number of misses for each simulation run was virtually the same, however, the sub peaks from the original runs are gone and the peak at 75 cycles (for the

new simulation run) grew indicating that the cost for these misses is shifted to the nominal miss penalty (latency) for an L3 hit. Second, individual misses can be identified and investigated using the miss-cost report or the cost analysis report. For example, a software designer can list the 'top 100 instructions' that caused the most misses (by cost). Once an instruction or miss is identified, a cycle-by-cycle trace will reveal the cause of the miss and miss cost. Substantial performance gains are possible even if only 10 to 20 percent of the misses (from the top 100 instructions) can be optimized. For example, the performance gain in just this one example was approximately 0.2 %. Third, pipeline spectroscopy has both the accuracy and precision we need to measure the cost of even a single cache miss. Accuracy is demonstrated by having the ability to reconstruct the true total finite cache adder by adding the cost of each individual miss cluster. Precision is demonstrated by the repeatability of the readings. For example, in the experiment above the LTI for an L3 hit was modeled at 16, 24, and 32 cycles and a shift of eight cycles was observed for each LTI change in the sub peaks shown in Figure 19.

In our next experiment we attempt to reduce the trailing edge penalty of a cache miss even further by modifying the order that the packets within a line are returned on the bus. That is, instead of returning packets in a fixed order (from the miss address, to the end-of-line address, then wrapping around to the beginning of the line) hardware is modeled that can transfer the line (packets) in the order they are referenced by the processor. Other mechanisms aimed at reducing the trailing edge penalty can be found in [25-28]. These describe techniques that multiplex data on the bus from concurrent misses, or can remember the referencing pattern generated by the processor during a prior miss and can rearrange the returning packets to match the previous referencing pattern when the miss or instruction causing the miss repeats.

In our scheme, we also have the ability to transfer the packets in a line in an out-of-order sequence. This applies to both L2 and L3 hits. Obviously, this involves adding (modeling) hardware that recognizes the order that the bytes are referenced within a line and supplying the packet (within the line) on the next available bus cycle along with the packet address. If there is not a new reference to the line during the time the line is being transferred between caches, the line is transferred in the original order as described above (the next in order packet).

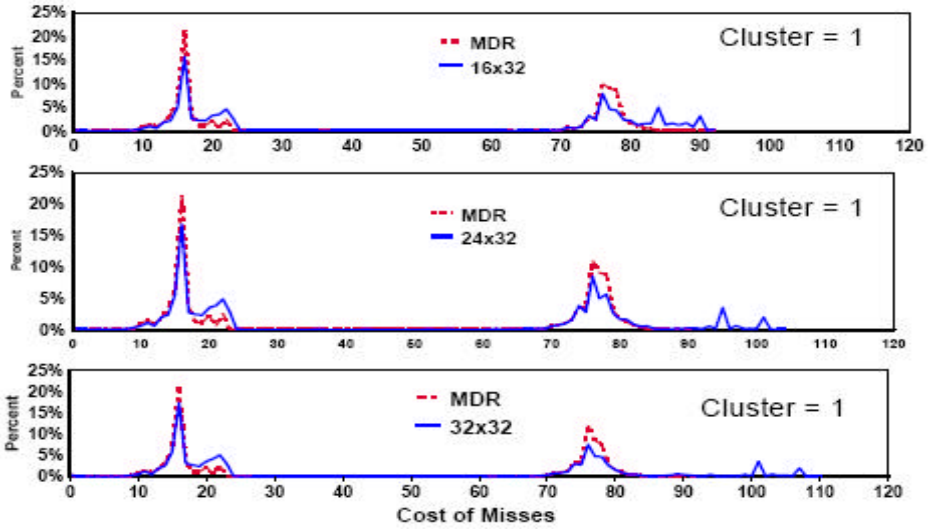


Figure 20: Modified data return for OLTP3.

Again we model the same L2, L3 and bus structure and model the three LTIs as before (16, 24, and 32 cycles). Figure 20 shows the before and after miss spectrogram for the OLTP3 workload for a miss cluster of size = 1. The modified data return is labeled MDR. The extent that this improves performance should be immediately visible by examining the shape of the miss spectrogram peaks centered at 15 (an L2 hit) and 75 (an L3 hit). Notice the change in the shape of the right shoulder for each peak, indicated a change in the trailing edge penalty. In each graph, the right shoulders for the peaks at 15 and 75 are reduced and more misses are shifted to have a nominal 15 or 75 cycle miss penalty.

Each workload described in Section 4 was rerun though the simulator using the modified data return mechanism described above. Figure 21 shows the overall performance impact for the six workloads studied. The performance improvements of changing the order that the line is returned increases as the number of cycles to transfer a line increases from 16, to 24, and 32 cycles. Depending on the LTI, performance improvements range from .5% to over 8%.

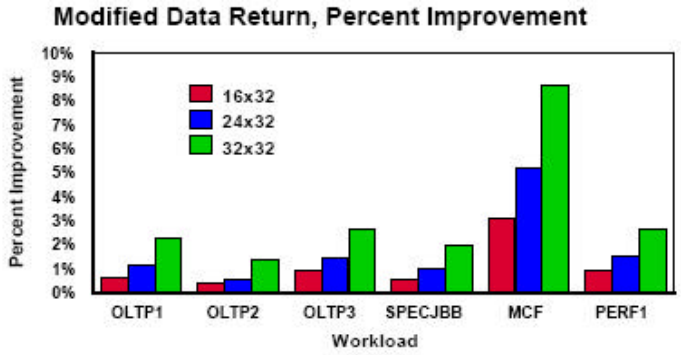


Figure 21: Percent Improvement.

## 10. Instruction Spectrograms

In the next set of experiments, we set the instruction cache to 64KB, L2=256KB with a 15 cycle latency, L3=1MB 75 cycles away, and a 300 cycle memory latency (the reverse of the experiment shown in Figure 6). Instruction fetching is guided by a 32K-entry branch target buffer (BTB) that runs well ahead of instruction fetching and the decoder, predicting branches and aiding prefetching. The benefits of using a BTB to guide instruction fetching and prefetch instructions have been well documented [24-28].

Figure 22 shows the instruction spectrogram for cluster sizes = 1, 2, 3, and 4 for oltp3. These spectrograms are different from their data counterparts. The spectrograms for cluster sizes 1, 2, and 3 have dominant peaks at 15, 75, and 300, with few misses (clusters) costing more than 300 cycles. The amount of overlap, in the cluster = 2 and 3 spectrograms, is substantial.

The cluster size = 2 spectrogram is still dominated by peaks at 15, 75 and 300, even though there are two misses. Obviously, one of the misses is overlapped with the first. In the cluster size = 3 spectrogram, there are four dominate peaks: 15, 30, 75, and 300. Here we have three misses but only the peak at 30 indicates that two out of the three misses were not overlapped. In the other cases (peaks at 15, 75, and 300), two out of the three misses were overlapped with the first.

Recall that the data spectrograms for these cluster sizes had miss penalties extending to 600 and 900 cycles, respectively. Even the cluster size = 4 spectrogram has peaks at 15, 75, and 300 cycles, indicating three out of the four misses were overlapped. Analyzing the peak at 150

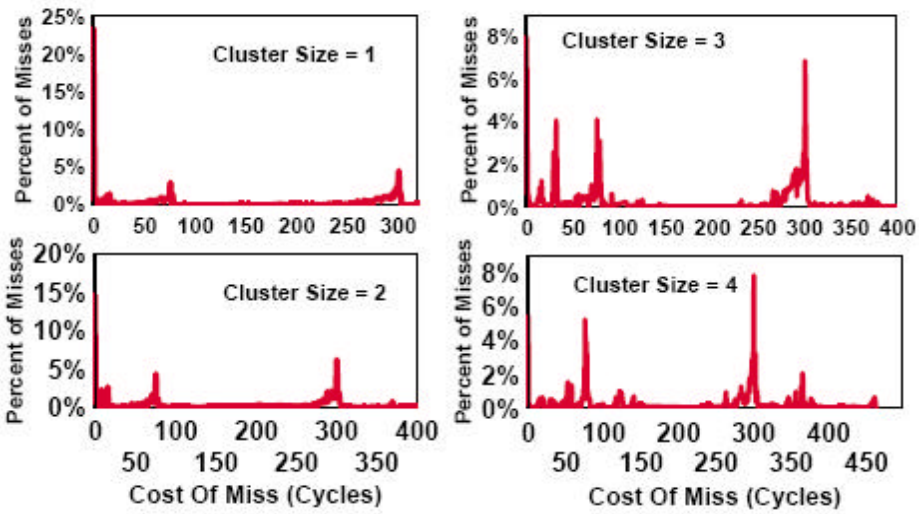


Figure 22: Instruction Miss Spectrogram. L1=64KB, L2=256KB with 15 cycle latency, L3=1MB with 75 cycle latency, and memory with 300 cycle latency.

indicates two of the four misses were resolved in the L3, while the other two misses were overlapped. Similarly, the peak at 375 indicates one miss went to memory and the other was resolved in the L3 (the other two misses were overlapped).

Notice the large left shoulders on each of the peaks in the cluster = 1 spectrograms. There is also a large peak at 0 indicating that nearly 25% of the instruction misses had zero delay. Even the spectrograms for cluster sizes = 2, 3 and 4 show peaks at zero. For example, the cluster = 4 spectrograms shows approximately 6% of the miss clusters had zero cycles penalties. Here we have four misses that occur in parallel but yet the BTB was able to prefetch them far enough in advance of their use to cause zero cycles difference between the finite cache simulation run and an infinite cache simulation run.

In Figure 23, we change the range of the X and Y axis of the cluster=1 spectrogram to emphasize the left shoulder of the 15 and 75 peaks (zoom in). We see a large left shoulder of the peak at 75 cycles extending to nearly 20 cycles (indicating that some prefetches are issued nearly 55 cycles ahead of their use). Typically, misses with penalties of 75 cycles or more are misses that are still not prefetched. Here, hardware or software designers would analyze these misses in detail (through a new simulation run) to determine why they were not prefetched and devise new mechanisms (prefetching algorithms) to prefetching these misses.

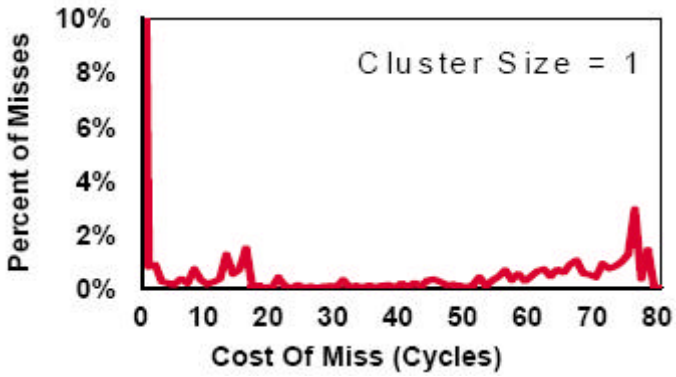


Figure 23: Instruction Spectrogram Data for the OLTP workload.

## 11. Spectrogram Shape With Bus Queuing

In our last experiment we highlight the effects bus queuing has on the shape of a spectrogram. We repeat the experiment shown in Figure 3 but use the oltp3 workload and change the memory latency to 50 cycles and model a 16 byte bus that transfers a packet of information every other cycle, as opposed to the optimistic 128B bus transfer used previously. Now, it takes 16 cycles to transfer a line between the caches. Figure 24 shows the data spectrograms for clusters = 1, 2 and 3. Notice the effects that bus queuing have on the shape of the right shoulder.

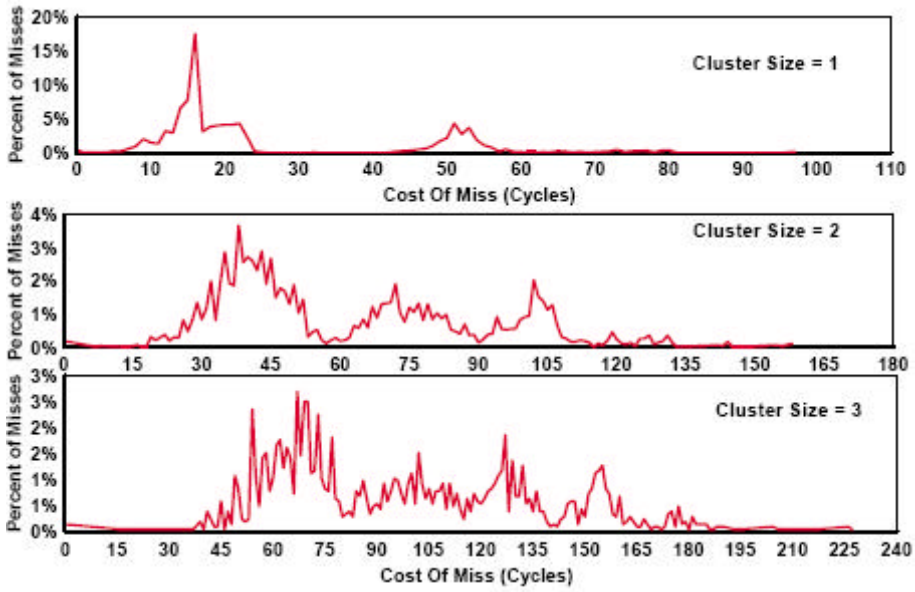


Figure 24: Data Miss Spectrogram. L1=64KB, L2=256KB with 15 cycle latency, L3=50 cycles latency. Line Transfer = 16 Cycles.

In the cluster = 1 spectrogram, we clearly have peaks centered at 15 and 50 cycles, but the peaks are not as sharp as when the line was transferred in one cycle. However, in the cluster = 2 spectrogram, the original five peaks shown in Figure 3, broaden and merge. Now, there are three large peaks, (not five as seen earlier) with one peak between 30 and 60, one between 60 and 90, and one between 90 and 120 with a large right shoulder. Obviously, there are four hit/miss combinations for this cluster but the cost of each miss cannot be identified as clearly as when the bus transfer interval was 1 cycle. Finally, in the cluster = 3 spectrogram, all three peaks start to merge between a range of 30 to 180. Obviously queuing is increasing the cost of each miss or miss cluster well beyond the nominal miss latency.

Additionally, in this configuration, the bus is limiting performance and inhibiting the parallelism we saw in Figure 3. The spectrograms shown for cluster sizes = 2 and 3 immediately give the designer visual feedback to measure the impact of the queuing. We find this visual feedback very helpful in designing hardware and prefetching algorithms.

## 12. Summary

A new technique has been presented for calculating the cost of a miss and displaying images that represent their cost. We call this technique pipeline spectroscopy. The underlying principles of this technique are very simple: the cost of a miss can be determined by knowing the finite cache and infinite cache execution times for the same sequence of instructions. The difference between these two times is the cost of the miss (cluster).

We use this principle to produce a miss spectrogram, which represents a precise readout of the cost of every miss. A miss spectrogram has enormous value in analyzing the performance of an application or microarchitecture. Detailed analysis of a spectrogram leads to insights in pipeline dynamics, including effects due to prefetching, bus queuing, and underlying architectural features that allow or inhibit memory level parallelism.

We also presented a theory that describes the observed properties that make up a spectrogram. The theory uses 6 parameters to predict the shape of a spectrogram: the global miss probability for the L2 ($p$), a correlation parameter $\alpha$, and four hit/miss overlap/no-overlap probabilities drawn from the four hit/miss combinations found in the cluster size = 2 miss patterns (from an application). We applied this theory and were able to predict the miss patterns for larger miss clusters.

In this study, we demonstrated that pipeline spectroscopy can be used to explore the amount of memory level parallelism an application can achieve. Future work is planned to study in-order versus out-out-order effects on MLP, as well as SMT, and shared cache contention (queuing) on multi-core processor organizations. There also appears to be much more information in the shape of the spectrogram than just the cost of a miss. Additional work is needed to understand all of the peaks and sub-peaks that are visible in a spectrogram (left and right shoulders and their sizes), as well as understand the number of cycles a peak shifts from its predicted position in the spectrogram to the observed position of the peak.

## References


[1] A. Glew, “MLP yes! ILP no!,” in ASPLOS Wild and Crazy Idea Session, October 1998.

[2] V. Pai and S. Adve, “Code Transformations to Improve Memory Parallelism,” in 32nd International Symposium on Microarchitecture, November 1999.

[3] H. Zhou and T. Conte, “Enhancing Memory Level Parallelism via Recovery-Free Value Prediction,” in International Conference on Supercomputing, June 2003.

[4] D. Sorin et al, “Analytic Evaluation of Shared-Memory Systems with ILP Processors,” in 25th International Symposium on Computer Architecture, 1998.

[5] V. Pai, P. Ranganathan and S. Adve, “The Impact of Instruction- Level Parallelism on Multiprocessor Performance and Simulation Methodology,” in HPCA February 1997.

[6] P. Ranganathan, K. Gharachorloo, S. Adve and L. Barroso, “Performance of Database Workloads on Shared-Memory Systems with Out-of-Order Processors,” in ASPLOS-VIII, 1998.

[7] Y Chou, B. Fahs, and S Abraham, “Microarchitecture Optimizations for Exploiting Memory-Level Parallelism Exploiting Memory-Level Parallelism” in 31st International Symposium on Computer Architecture, 2004.

[8] Yuan Chou, Lawrence Spracklen, Santosh G. Abraham. "Store Memory-Level Parallelism Optimizations for Commercial Applications," pp. 183-196, 38th MICRO 2005.

[9] M. Qureshi, D. Lynch, O. Mutlu, Yale Patt, “A Case for MLP-Aware Cache Replacement” in 33rd ISCA June 2006.

[10] A. Zahir, V. Hummel, M. Kling, T Yeh, *US. Patent 6,353,802*, “Apparatus and Method for Cycle Accounting in Microprocessors”.

[11] B. Gaither, R. Smith, *US Patent 6,892,173 B1*, “Analyzing Effectiveness of a Computer Cache By Estimating a Hit Rate Based on Applying a Subset of Real-time Addresses to a Model of the Cache”.

[12] H. Ravichandran, *US Patent 6,341,357 B1*, “Apparatus and Method for Processor Performance Monitoring”,

[13] R. Trauben, *US Patent 5,594,864*, “Method and apparatus for unobtrusively monitoring Processor States and Characterizing Bottlenecks in a Pipeline Processor Executing Grouped Instructions”

[14] G. Brooks, *US Patent 5,845,310* “System and Methods For Performing Cache Latency Diagnostics in Scalable Parallel Processing Architectures Including Calculating CPU Idle Time and Counting Number of Cache Misses.

[15] W. Flynn, *US Patent 6,256,775 B1*, “Facilities For Detailed Software Performance Analysis in a Multithreaded Processor”

[16] F. Levine, B. McCredie, W. Starke, E. Welbon, *US Patent 5,862,371*, “Method and System for Instruction Trace Reconstruction Utilizing Performance monitor outputs and bus Monitoring”

[17] J. Dean, J. E. Hicks, C. A. Waldspurger, W. E. Weihl, and G. Z. Chrysos. *ProfileMe : Hardware support for instruction-level profiling on out-of-order processors*. In MICRO'97: pages 292--302, 1997.

[18] Brian A. Fields, Rastislav Bodik, Mark D. Hill, Chris J. Newburn., Interaction cost and shotgun profiling. ACM Transactions on Architecture and Code Optimization, Vol 1, No. 3. Sept 2004.

[19] Tejas Karkhanis, James E. Smith, A First-Order Superscalar Processor Model. Proceedings of the 31st ISCA. pages 338–349, June 2004.

[20] A. Hartstein and T. Puzak. The optimum pipeline depth for a microprocessor, 29th ISCA, pages 7-13 May 2002.

[21] A. Hartstein and T. Puzak. Optimum power/performance pipeline depth. 36th Annual IEEE/ACM International Symposium on Microarchitecture In *MICRO*, Dec. 2003.

[22] P.J. Denning, "The Working Set Model for Program Behavior", CACM 19(5) pp.285-294 (1976).

[23] R. Bartoszynski, M Niewiadomska-Bugaj,Probability and Statistical Inference, (Wiley series in probability and statistics) 1996

[24] J.S. Liptay, "Design of the IBM Enterprise System/9000 High-End Processor," IBM Journal of Research and Development, Vol. 36, No. 4, pp. 713-731, July, 1992.

[25] D.B. Fite, J.E. Murray, D.P. Manley, M.M. McKeon, E.H. Fite, R.M. Salett, and T. Fossum, "Branch Prediction," *U.S. Patent #5142634*, assigned to Digital Equipment Corporation, Filed Feb. 3, 1989, Issued Aug. 25, 1992.

[26] N. Suzuki, "Microprocessor Having Branch Prediction Function," *U.S. Patent #5327536*, assigned to NEC Corporation, Filed May 22, 1991, Issued July 5, 1994.

[27] C.H. Perleberg, and A.J. Smith, "Branch Target Buffer Design and Optimization," *IEEE Transactions on Computers*, Vol. 42, Issue 4, pp. 396-412, Apr., 1993.

[28] B. Calder, and D. Grunwald, "Fast and Accurate Instruction Fetch and Branch Prediction," *21st ISCA* pp. 2-11, April 18-21, 1994. [2] D. Boggs et al. "The microarchitecture of the Intel Pentium 4 processor on 90 nm technology", *Intel Technology Journal*, vol. 8, Issue 1, 1997.

## Appendix A

In a memory hierarchy with N cache levels (L1, L2, ..., $L_N$, and memory) and a miss cluster of size C, there are $\binom{C+N}{C}$ possible miss penalties. Let each level of the memory hierarchy be represented by a distinct (non-multiple) miss penalty (number), there are N of them. We represent this problem as sampling with replacement to determine the number of unique combinations (sums) using these numbers. It is sampling with replacement because there is an inexhaustible supply of miss latencies regardless of the cluster size. We use the notation $\left[\begin{matrix} N \\ C \end{matrix}\right]$ to

denote N items choose C (the cluster size) for sampling with replacement. The problem can then be expressed as determining the number of unique sums from N items as we vary the number of picks from 0 to C. Note that *i* and *C* can be greater than N in (1a) because sampling is done with replacement.

$$\sum \left[ \begin{matrix} N \\ i \end{matrix} \right] 0 \le i \le C = \left[ \begin{matrix} N \\ 0 \end{matrix} \right] + \left[ \begin{matrix} N \\ 1 \end{matrix} \right] + \left[ \begin{matrix} N \\ 2 \end{matrix} \right] + \left[ \begin{matrix} N \\ 3 \end{matrix} \right] + ,\ldots, + \left[ \begin{matrix} N \\ C \end{matrix} \right] \qquad \text{(1a).}$$

from [23]

$$\left[ \begin{matrix} N \\ k \end{matrix} \right] = \left( \begin{matrix} N+k-1 \\ k \end{matrix} \right)$$

So we can rewrite (1a) as

$$\left( \begin{matrix} N-1 \\ 0 \end{matrix} \right) + \left( \begin{matrix} N \\ 1 \end{matrix} \right) + \left( \begin{matrix} N+1 \\ 2 \end{matrix} \right) + \left( \begin{matrix} N+2 \\ 3 \end{matrix} \right) \ldots \left( \begin{matrix} N+C-1 \\ C \end{matrix} \right) \qquad \text{(2a)}$$

Also, from [23]

$$\left( \begin{matrix} N \\ k \end{matrix} \right) = \left( \begin{matrix} N-1 \\ k-1 \end{matrix} \right) + \left( \begin{matrix} N-1 \\ k \end{matrix} \right) \qquad \text{(3a)}$$

So we can combine the first two terms of (2a) and obtain

$$\left( \begin{matrix} N+1 \\ 1 \end{matrix} \right) + \left( \begin{matrix} N+1 \\ 2 \end{matrix} \right) + \left( \begin{matrix} N+2 \\ 3 \end{matrix} \right) \ldots \left( \begin{matrix} N+C-1 \\ C \end{matrix} \right) \qquad \text{(4a)}$$

Applying (3a) repeatedly the series collapses and the desired sum is produced $\left( \begin{matrix} C+N \\ C \end{matrix} \right)$.

## Appendix B

We show that the correlation coefficient $\rho = (1-\alpha)$. We represent hits as 0 and misses as 1 such that $X_i = M$ or $X_i = H$ is equivalent to $X_i = 1$ or $X_i = 0$, respectively. Let $\Pr[X_1 = 1] = p$. By definition the correlation coefficient

$$\rho = \frac{COV(X_1, X_2)}{\sqrt{Var(X_1) Var(X_2)}}.$$

In order to compute $COV(X_1, X_2)$ we need to calculate the $E[X_1, X_2]$, $E[X_1]$, and $E[X_2]$.

$$E[X_1X_2] = \Pr[X_1X_2 = 1] =$$
$$\Pr[X_1 = 1, X_2 = 1] = p(1 - a + ap)$$

$$E[X_1] = E[X_2] = p.$$

Combining results from above, we obtain,

$$COV(X_1, X_2) = p(1 - a + ap) - p^2 =$$
$$(1 - a)[p(1 - p)]$$

Finally,

$$Var(X_1) = Var(X_2) = p(1 - p)$$

So,

$$\rho = \frac{(1-a)[p(1-p)]}{p(1-p)} = 1 - a$$