Journal of Instruction-Level Parallelism 10 (2008) 1-22 Submitted 11/06; published 6/08

On the Nature of Cache Miss Behavior: Is It $\sqrt{2}$?

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

Abstract

It has long been empirically observed that the cache miss rate decreased as a power law of cache size, where the power was approximately -1/2. In this paper, we examine the dependence of the cache miss rate on cache size both theoretically and through simulation. By combining the observed time dependence of the cache reference pattern with a statistical treatment of cache entry replacement, we predict that the cache miss rate should vary with cache size as an inverse power law for a first level cache. The exponent in the power law is directly related to the time dependence of cache references, and lies between -0.3 to -0.7. Results are presented for both direct mapped and set associative caches, and for various levels of the cache hierarchy. Our results demonstrate that the dependence of cache miss rate on cache size arises from the temporal dependence of the cache access pattern.

1. Introduction

Most modern processor systems include a cache hierarchy [1-4], and will continue to do so in future. There are numerous papers [5 - 13] discussing the use of analytical cache models to predict cache behavior. However there are relatively few papers that deal with the origins of the dependence of the miss rate of a cache on the corresponding size of the cache. Chow [5, 6] formulates an analytical cost-performance model for caches in order to derive the optimum cache hierarchy that maximizes the performance. The model assumes that the miss rate is a power law function of the cache's capacity. Przybylski et. al. [8, 9], use the observed relationship between cache miss rate and capacity to size the various levels of the cache hierarchy.

All of these studies have been empirical in nature. It is clear that if a workload is small and fully fits within the size of a particular cache, the miss rate will be small. However, if the workload is large, the cache miss rate is observed to decrease as a power law of the cache size, where the power is approximately -1/2. This is the well known $\sqrt{2}$ rule, where if the cache size is doubled, the miss rate drops by the factor $\sqrt{2}$. We gave a preliminary account of this work at the Computing Frontiers Conference.[14]

In this paper we examine this dependency theoretically in order to understand the underlying cause of the behavior. We also use detailed simulations to test these theoretical ideas. Both first level caches and second level caches are studied. Results are presented for both direct mapped and set associative caches. The results can be readily extended to any level of a memory hierarchy. It is found that the dependence of miss rate on cache size arises from the time dependence of the cache reference pattern exhibited by a particular workload. In particular, the $\sqrt{2}$ rule emerges because most large complex workloads display a cache re-reference timing pattern, which obeys a power law. The power law, observed for the cache references, shows exponents ranging from $-1.7 \leq \beta \leq -1.3$. The corresponding power law for the miss rate, the square root rule, ranges from $-0.7 \leq \alpha \leq -0.3$. We show theoretically that these two power laws are interrelated; that is $\alpha = \beta - 1$.

A very different approach to the same problem was taken by Singh, Stone and Thiebaut [15]. In their paper the starting point is the number of unique cache references occurring in a finite time interval, and the growth of that number as the time interval increases. Next they extract the dependence of the miss rate on cache size from that function. Their analysis was for a fully associative first level cache, with an approximate extension to first level set associative caches. Our approach starts from the temporal dependence of cache references, and applies a statistical model of cache behavior to obtain the results presented here. It is applicable to both direct mapped and set associative caches without approximation, and is also applicable to all caches in a hierarchy and not limited to the first cache level.

2. Simulation Methodology

In order to explore the dependence of cache miss rate on the cache size, we have used a proprietary simulator [16]. As input the simulator uses design parameters that describe the organization of the processor and a trace tape. It produces a very flexible cycle accurate model of the processor. With this tool we are able to model two levels of cache hierarchy leading to main memory, numerous pipeline designs, various issue width superscalar designs, in-order execution processing and out-of-order execution processing. This tool has mainly been used for work on IBM zSeries processors.

Even though our simulator accurately models the entire processor pipeline, for this study the main portion of the model only serves to feed the cache model with an accurate time dependent reference stream. Many of the details of the simulated model are not important to this study. The cache hierarchy portion of the model allows for various cache sizes, set associativities, and line sizes for each level of the hierarchy. Both latencies and trailing edge effects are accurately modeled for all levels of the memory structure. Busses are accurately modeled, between different levels of the hierarchy and between the caches and the processor, including all timing and port contention. For this study the models employed separate instruction and data caches. For the set associative cache models an LRU replacement algorithm was used.

We also had the availability of 55 traces, encompassing traditional (legacy) workloads, “modern” workloads, SPEC95 and SPEC2000 workloads. The traditional workloads include both database and on-line transaction processing (OLTP) applications. The modern workloads were real, substantial workloads, written in either C++ or Java. These traces were carefully selected to accurately reflect the instruction mix, module mix and branch prediction characteristics of the entire application, from which they were derived.

Since our aim was to study cache behavior, most of the workloads available were not suitable. Any workload, which fits entirely within the second level cache, was discarded. Therefore, most of the SPEC benchmarks were not used for this study. Instead we focus on the behavior of large complex workloads, which adequately stress the memory structure. The workloads used were largely on-line transaction processing (OLTP) and a processor simulator. An exception to this protocol was made in order to understand workload dependences. Here we included all of the workloads, but only L1 cache studies could be performed.

3. Origin of the Square Root Rule

The dependence of the cache miss rate on cache size (Eq. 1), the square root rule, is well known [6, 11 - 13].

$$M = M_0 A^{-\alpha} \tag{1}$$

Here, A is the cache size and the exponent, α, typically takes on values between 0.3 and 0.7. However, the rule is only empirical and therefore is not necessarily accurate in all situations. In particular, if the memory footprint of a particular workload is smaller than the cache size, so that the data set fits within the cache, there will be no cache misses after the initial loading of the caches. The much more interesting case is for large workloads which are much larger than the cache size. For many of these large workloads, we show that the square root rule is a natural consequence of the time dependence of cache requests along with the probability of finding an item in the cache when requested.

A cache entry is only useful if it is re-referenced before it is evicted from the cache. An important parameter governing this process is the time interval between these references. This time interval is workload dependent and can in principle take on any functional form. We have modified our simulator to determine the time interval between each cache line reference and its next reference. It is found that most workloads have a similar dependence, shown in Fig. 1a.

Each data point in this graph shows the number of occasions in which the time interval between subsequent references to any cache line takes on specific values, e.g. the time interval of 200 cycles occurs 214 times for data references and 508 times for instruction references. Note these re-reference patterns are only dependent on the workload and the cache line size. The same

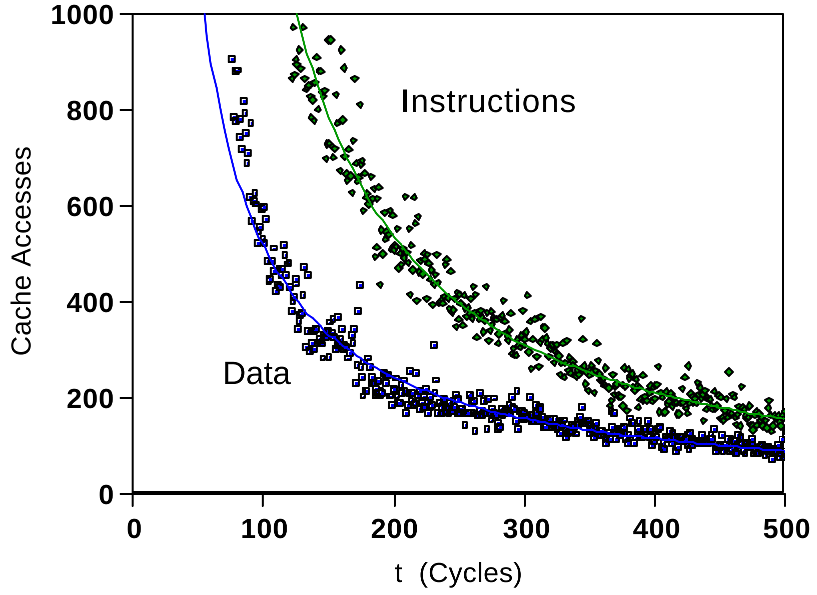

Figure 1a: Re-reference pattern for both data and instructions in the OLTP-3 program fit with a power law dependence.

.

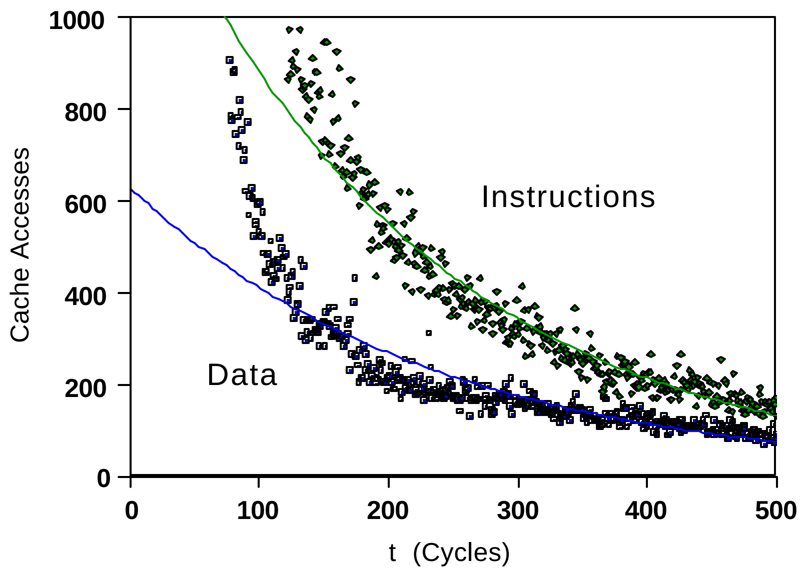

Figure 1b: Re-reference pattern for both data and instructions in the OLTP-3 program fit with an exponential dependence.

functional form is found for both instruction cache and data cache references. In general the re-reference interval follows a power law with additional structure, indicative of specific workload features, superimposed on the general behavior.

We initially thought that the re-reference pattern would follow a decaying exponential rather than the power law shown in Fig. 1a. In Fig. 1b the identical data (Fig. 1a) is plotted along with the best fits to an exponential decay. It is abundantly evident that an exponential decay does not fit the data nearly as well as the power law fit shown in Fig. 1a. The same results obtain for all of the workloads, studied. Therefore, our analysis for the remainder of this paper derives from the observed power law dependence

The rate of re-referencing a specific cache line is given by

$$R(t) = R_o t^{-\beta}. \tag{2}$$

The rate of re-references, as a function of the time interval, is a decaying function of time. The decay constant, β, is found to take on values ranging from 1.3 to 1.7. The data shown in the figure are average rates for all cache lines, rather than for any specific line. However, a program, which contains a large loop revisiting the same code time after time, will show very different behavior. We will return to this point in a later section.

In order to determine the cache miss rate we need to determine the probability that a re-referenced cache line is still resident in the cache. For simplicity we will first consider the problem of a direct mapped cache and later extend the analysis to set associative caches. We consider a cache with total capacity, A, and a line size, A_l. For the direct mapped cache the number of cache lines is simply given by $N = A/A_l$. Whenever a reference to the cache misses, it removes an entry from the cache. If the usage of the cache is uniform, the probability that a particular cache line is ejected is $P = 1/N$ after one miss. The probability that a particular cache line will still be in the cache after n misses is

$$P = (1 - 1/N)^n \ , \tag{3}$$

and the probability that a particular cache line will have been ejected after n misses is

$$P = 1 - (1 - 1/N)^n . \tag{4}$$

We can change this formulation of the probability in terms of discreet events, cache misses, into continuous values of the parameters. This approximation is valid as long as the number of events is large. It has the advantage that the mathematics for continuous variables is much easier. Eq. 4 then takes on the form

$$P(t) = 1 - e^{-\frac{M_{ave}}{N}t} , \tag{5}$$

where we have expressed the number of misses, n, as an average miss rate, M_{ave}, multiplied by a time, t, in cycles. We only utilize this average miss rate as a scaling factor between the number of misses and time. Note that Eqs. 4 and 5 are mathematically equivalent for large n.

The cache miss rate, essentially the probability that a cache reference at time, t, will miss in the cache, is given by the product of Eqs. 2 and 5.

$$M(t) = R_o t^{-\beta}(1 - e^{-\frac{M_{ave}}{N}t}) \tag{6}$$

Eq. 6 gives the miss rate as a function of the reference time. In order to obtain the total miss rate, Eq. 6 must be integrated over all time.

$$M = \int_0^\infty R_o t^{-\beta}(1 - e^{-\frac{M_{ave}}{N}t})dt \tag{7}$$

This equation can be evaluated by making a change of variables $T = \frac{M_{ave}}{N}t$, where $N = A/A_l$. Substituting this expression into Eq. 7 and pulling those parameters, independent of T, out of the integral gives:

$$M = R_o\left(\frac{A}{A_l M_{ave}}\right)^{1-\beta} \int_0^\infty T^{-\beta}(1 - e^{-T})dT \tag{8}$$

The important point shown in equation 8 is that the miss rate is dependent on the cache size, A, and that the integral is simply a constant. In fact the integral is well known as the gamma function, Γ(1-*β*) with tabulated values readily available. Therefore our final expression for the miss rate is

$$M = \frac{R_o}{(A_l M_{ave})^{1-\beta}} \Gamma(\beta - 1) A^{1-\beta} , \qquad (9)$$

where the miss rate is given by the cache size to the power 1-*β* multiplied by a particular set of constants. Eq. 9 predicts that the miss rate will vary with cache size by the "$\sqrt{2}$" rule, with the exponent, varying between -0.3 and -0.7 with workload. When *1-β =-0.5*, an exact $\sqrt{2}$ rule is observed.

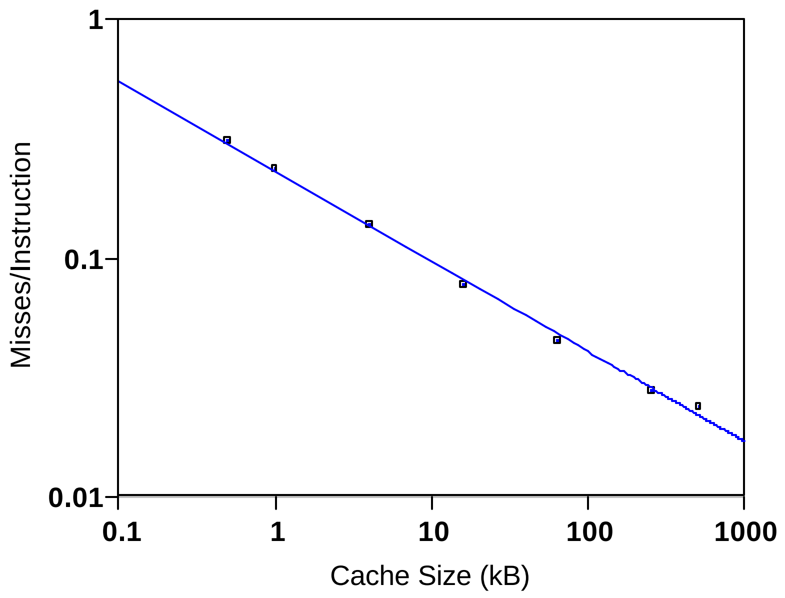

Figure 2: The cache miss rate as a function of cache size for the OLTP-1 workload.

Figure 2 shows the cache miss rate for a first level cache as a function of cache size on a log-log plot. The best fit to the form of Eq. 9 is also shown. In this and subsequent figures simulation data are shown as individual points, whereas theory is shown a continuous lines. For many workloads, those whose re-reference pattern is given by Eq. 2, the predicted dependence gives a good fit. Table 1 shows a comparison of *1-β* values obtained from the re-reference pattern by fitting those data to Eq. 2, with the *1-β* values obtained from fitting the miss rate versus cache size dependence to Eq. 9.

As is clearly evident, there is good agreement. This gives us a clear understanding of the origin of the miss rate dependence on cache size. It arises from the temporal dependence of the cache reference pattern.

4. Workload Differences

At this point we wish to consider the power law exponents ($\beta - 1$) obtained for different workloads. Table 1 enumerates some of these values, but the values of the exponent from data re-references for all of the workloads have been collected in the bar chart shown in Fig. 3.

	from Re-reference Data	from Cache Miss Data
OLTP-1	0.3	0.378
OLTP-2	0.39	0.361
OLTP-3	0.34	0.4
Crafty	0.71	0.713
GZIP	0.61	0.636

Table 1: Values of (β-1) determined by the two different methods.

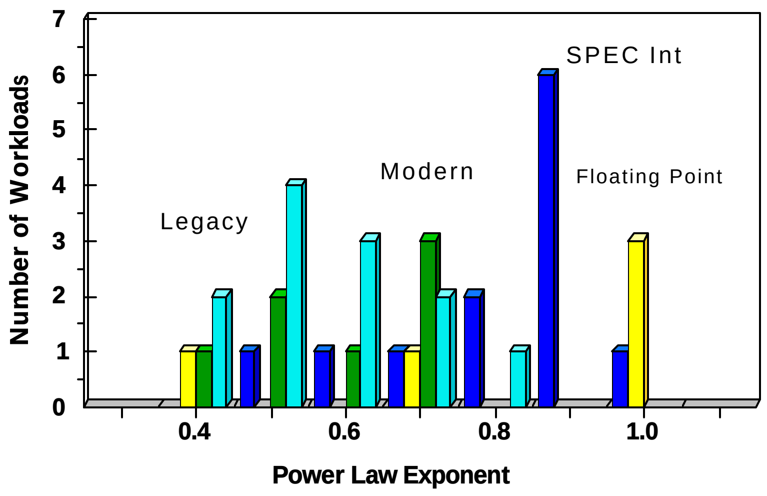

Figure 3: A compilation of the observed power law exponents for different classes of workloads.

Similar results obtain for instruction references. The workloads have been divided up into 4 categories: legacy workloads, modern workloads, SPEC Int workloads and floating point workloads. The legacy workloads are large server workloads, mainly written in assembly language. The modern workloads are also large server workloads, but written in either C++ or Java. The floating point workloads tend to be scientific calculations, some of which are from the SPEC suite, and some from other sources. All of the observed exponents have been divided up into bins of width 0.1. The bars show the number of workloads of each type that fall into particular bins. All exponents greater than 1.0 have been included in the 1.0 bin.

It is clear that the temporal re-reference patterns for different classes of workloads differ. In particular the re-reference interval for the SPEC Int workloads tends to be of shorter duration, the larger exponent means that the probability of re-referencing these cache lines fall off rapidly with time. This, coupled with the smaller working sets for most of these workloads, leads to the relatively small cache requirements for them.

Since the values of the power law exponent vary with workload and show variations dependent on the class of workload, one is led to search for an underlying reason for the observed variation. In this vein we speculate that workloads with smaller working sets will in general display larger exponents. The essence of the argument is that for small workloads it is statistically necessary to re-reference a cache line after a smaller time interval than for larger workloads. The argument is simply that for the smaller workloads there simply are not enough other cache lines to reference, so that one is forced to return to recently used ones rather quickly. For larger workloads this does not need to occur because there are more than enough other cache lines to reference. This naturally leads to larger values of $\beta - 1$ for smaller workloads.

In order to see if this argument has any validity we searched for a metric to account for the effective working set size of a workload. We settled on the total number of unique instruction or data references as a measure of the total working set size. Since it was hard for us to determine this directly, we determined the number of unique cache line references for instructions and data. For each workload we determined the number of cache misses for both data and instructions using very large (64 MB) L1 caches.

We note that even the total working set size of a workload is not a good measure of the effective working set size. Let's consider a workload that consists of numerous small modules that need to be executed. There are 2 very different ways in which this can be done. One workload may spend a large amount of time in one particular module, and then move on to the next module. This workload will behave as if it is a small workload since it spends most of its time in one particular module at a time. On the other hand a different workload may be the same size, but the program flow takes it from module to module in a fairly rapid manner. It would then come back to repeat a similar or different pattern of using the modules. This type of behavior will effectively look like a much larger workload since a much larger portion of the code is visited, on average, before returning to a particular module.

Nevertheless, in figure 4a we plot the observed values of the power law exponent as a function of the total working set size for instructions, and in figure 4b we do the same for data. Each point in one of the figures represents a different workload. By aggregating the data in this way it is possible for an individual workloads to behave as small for the instruction stream, while behaving as large for the data stream, and vice versa. Note that these are log-log plots and the best fits to the data are power laws.

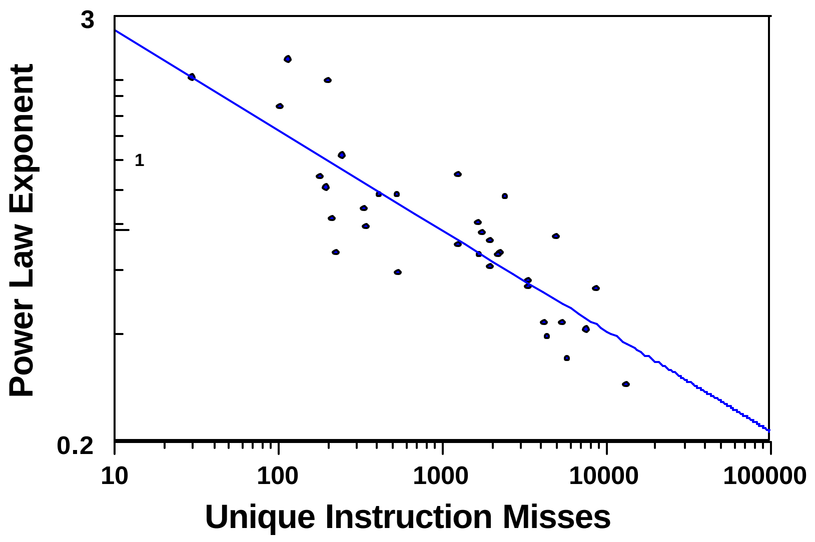

Figure 4a: The power law exponent (β-1) as a function of the number of unique instruction misses on a log - log plot. The line is a power law fit.

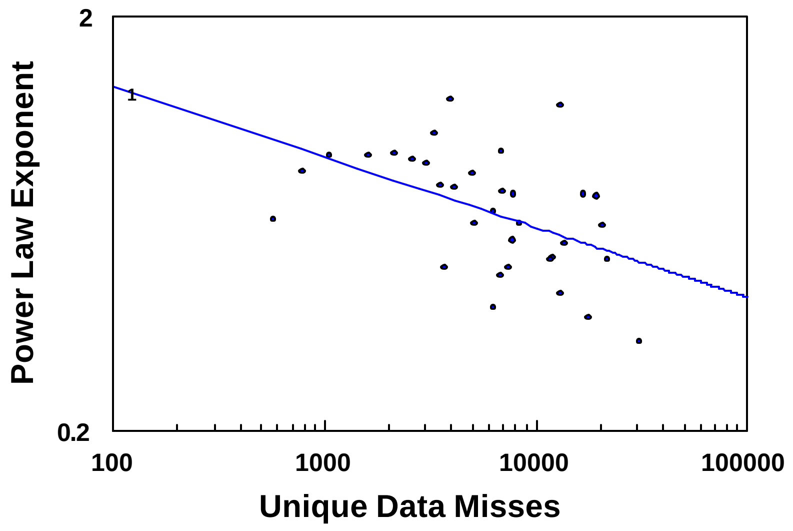

Figure 4b: The power law exponent (β-1) as a function of the number of unique data cache misses on a log - log plot. The line is a power law fit.

It is clear from the figures, that even though there is a large scatter in the behavior of the numerous workloads, there is indeed a correlation between the number of unique cache misses, and hence working set size, and the power law exponent. The correlation is stronger for the instruction stream and weaker for the data stream. The best fit lines, shown in the figures, represent power law dependences of the power law exponents on effective workload size. The correlation validates the speculations that were made above. We do not understand the exact functional form observed, but assume that it is grounded in the statistics of the problem.

5. Temporal Dependence of Cache Misses

Along with the prediction of the cache miss rate on cache size, Eq. 9, we have also obtained a prediction of the temporal dependence of cache misses, Eq. 6. This contains even more detail about the functioning of the first level cache. In our simulator we are able to obtain this cache miss timing distribution using much the same protocol that was used to obtain the original cache re-reference pattern. The data for one workload is shown in figure 5. Also shown in the figure are 2 theoretical curves. The upper one, which is not expected to fit the data, is Eq. 2, the original re-reference pattern for the cache. The lower curve, which fits the data quite well with no adjustable parameters, is a modified form of Eq. 6.

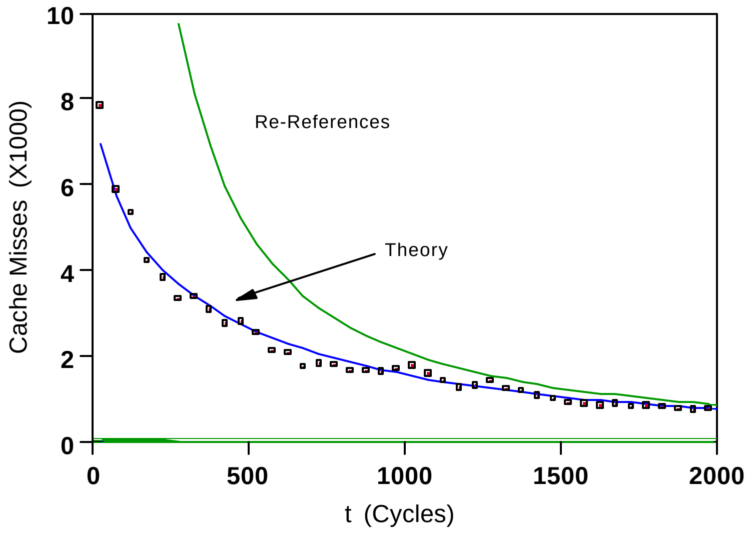

Figure 5: Cache misses as a function of the re-reference time for a direct mapped cache and the OLTP-1 workload.

In order to obtain Eq. 6 one assumption had been made, which turns out not to be accurate. In writing down Eq. 3 we assumed that cache lines were utilized uniformly, a quite reasonable, but incorrect, assumption. Figure 6 shows the actual distribution of cache misses to the various cache lines in a 4 kB direct mapped cache for the workload shown in figure 3. As is clear the usage is far from uniform. This modifies the statistics incorporated into Eq. 3. The statistics are changed in two ways. We had assumed that any cache miss was equally likely to occur for any line, hence the $1/N$ factor in Eq. 3. With non-uniform access the probability that any one line is ejected from the cache is M_i/M_{tot}, the fraction of misses attributed to line i. Additionally, one needs to form the weighted average of this probability over all lines in the cache. The more heavily used lines are more heavily weighted in this average. The $1/N$ factor is then replaced by

$$1/N \rightarrow \frac{\sum_i M_i^2}{M_{tot}^2}, \qquad (10)$$

where M_i are the number of misses for each cache line, and M_{tot} is the total number of

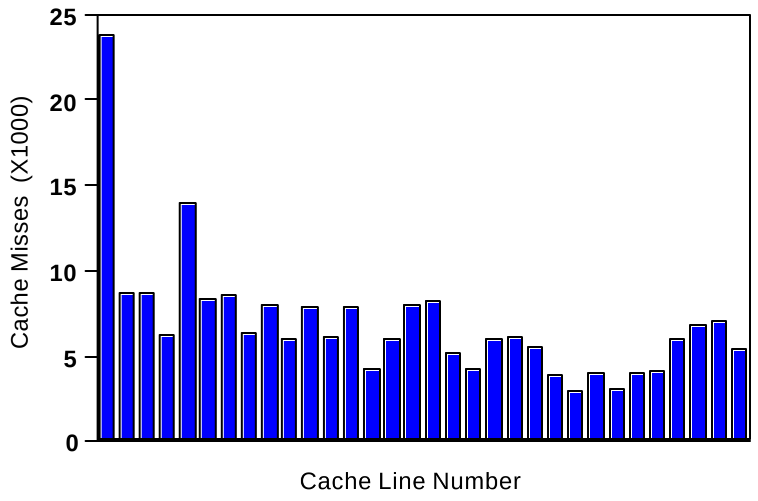

Figure 6: Distribution of Cache Misses for a direct mapped cache and OLTP-1.

cache misses to all lines. These non-uniform usage statistics result in a 25% change in the $1/N$ probability and have been incorporated into figure 3.

Several comments on figure 3 are in order. The upper curve in figure 3 is just the re-reference pattern of the workload, Eq. 2. The lower curve is the cache miss pattern of the cache, Eq. 6. The difference between these 2 curves represents cache hits. As we would expect, cache hits predominantly occur for short re-reference times. For long times the cache entry has almost always aged out and the cache miss pattern and the re-reference pattern converge. Anticipating our analysis for a second level cache, we note that the miss rate pattern from the first level cache becomes the reference pattern for the second level cache. The first level cache filters out a very specific portion of the cache references, which hit in that cache.

6. Set Associative Caches

The above analysis needs to be modified for set associative caches. The reference pattern is unchanged, but the probability of finding the referenced line in the cache is modified. It is modified because s separate references to the same cache congruence class need to occur prior to the particular re-reference that causes a cache miss, if s is the number of sets (ways) in the cache.

with each subsequent new cache reference the entry in question is pushed into a new set in the cache, until it is finally ejected from the cache.

A useful approximation for this entire statistical process can be obtained by noting that the probability of an entry being ejected from set 1 and put into set 2 is given by Eqs. 4 or 5, the same as ejection from a direct mapped cache. The same probability obtains for transfer from set 2 to 3, etc. In this way one obtains a new expression for the probability of a cache miss. Misses only occur if the entry misses in all of the sets of the cache. Therefore, we need to calculate the joint probability of missing all of the sets,

$$P(t) = (1 - e^{-\frac{M_{ave}}{N}t})^s = (1 - (1 - 1/N)^n)^s, \qquad (11)$$

where we have shown both the continuous and discreet forms of the probability. The number of cache congruence classes, N, now depends on the set associativity as well as on the cache size and the line size, $N = A/sA_l$. We note that Eq. 11 is an approximation in that it does not adequately take into account the sequential nature of the cache entry replacement problem. It also does not address the rearrangement of the entries into different sets following cache hits, an LRU replacement algorithm.

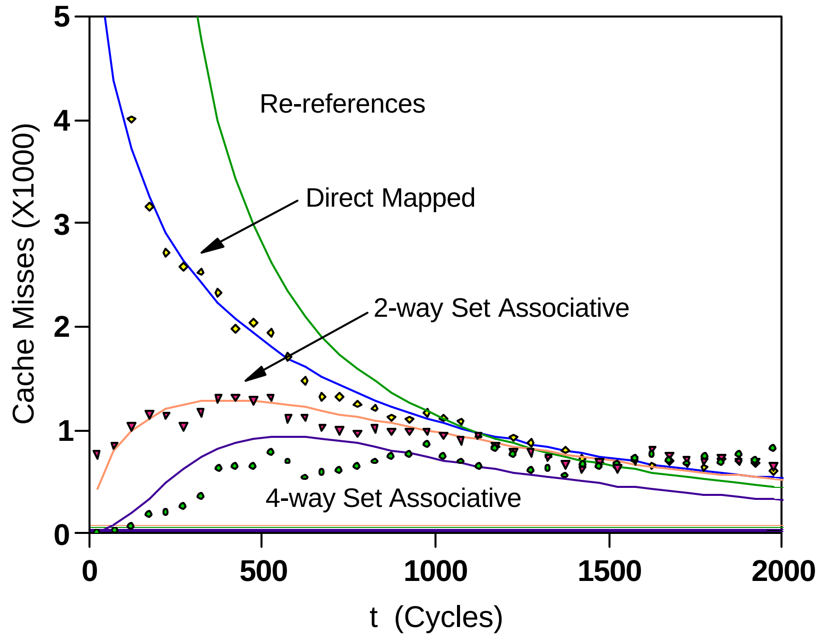

Figure 7: The time dependence of cache misses for set associative caches with OLTP-1.

Nevertheless, as can be seen in figure 7, this analysis captures the essential features of the time dependent miss curve for a set associative cache. In the figure data from one workload is shown for 4 kB caches, which are direct mapped, 2-way set associative and 4-way set associative. In addition we show the theoretical curves for the re-reference pattern and the miss patterns for each of the caches obtained by modifying Eq. 6 to incorporate the set associativity as expressed in Eq. 11.

$$M(t) = R_o t^{-\beta}(1 - e^{-\frac{M_{ave}}{N}t})^s \tag{12}$$

It is clear that our analysis accurately accounts for the data. By increasing the set associativity one does a better job of retaining useful information in the cache and the area between the curves, associated with cache hits, expands.

There is another way of qualitatively looking at the shape of these curves for the set associative case. Before a cache miss may occur, one must wait for some delay time, t_o, before the cache entry is in the last set (LRU set) of the cache. At this point the problem looks like the problem for a direct mapped cache. Qualitatively, one would expect a curve shaped like the direct mapped case, but offset by this average delay time. This essentially leads to zero probability of a miss for times shorter than t_o. Now in a real system this delay time will vary for each cache congruence class and for each occurrence of a cache miss. Therefore, the average behavior will smooth off the abrupt onset of the curves. This causes the type of peaked structure observed.

In order to obtain the dependence of the miss rate on cache size for the set associative case we need to integrate Eq. 12 over all time. This gives

$$M = \frac{R_o}{(sA_l M_{ave})^{1-\beta}} A^{1-\beta} \int_0^\infty T^{-\beta}(1 - e^{-T})^s dT, \tag{13}$$

after using the same substitution and procedure that was used to obtain Eq. 8. Again, the integral does not depend on the cache size. Therefore, the miss rate dependence on cache size has the same form as for the direct mapped cache, only the prefactor is changed. The value of the integral can be calculated as $f(s,\beta)\Gamma(1-\beta)$, where $f(s,\beta)$ is a complex function of the set associativity and β. This only contributes a numerical factor.

In figure 8 we plot the dependence of the miss rate on the cache size of a first level cache for three different values of the set associativity. The uppermost curve is for the direct mapped cache. We also show the fit to the theory. As is clearly seen, the exponent, $1-\beta$, remains the same for all set associativities, as predicted. Only the scale factor changes. This further explains the rather ubiquitous observation of the approximate "$\sqrt{2}$" rule.

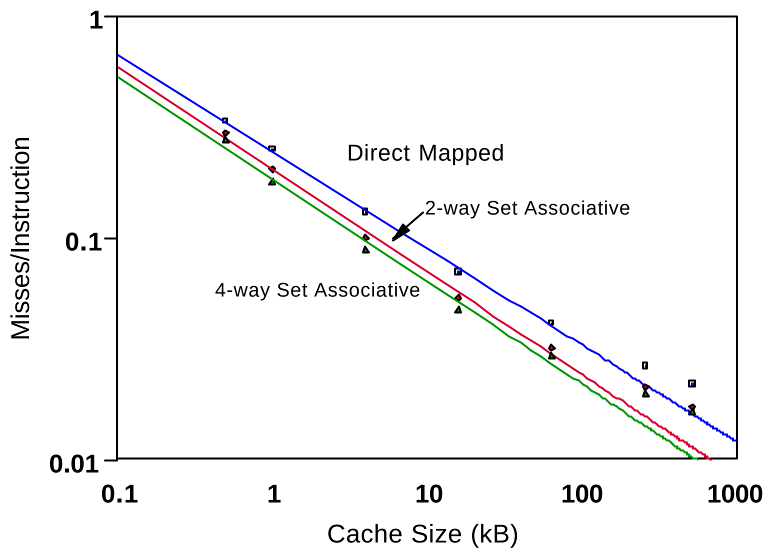

Figure 8: The dependence of the cache miss rate on cache size for difference set associativities using OLTP-3.

7. Second Level Caches

The behavior of second level caches is complicated by several factors. First, our assumption, that the working set size of the workload is much larger than the cache size, is harder to realize for the larger second level caches. Second is the fact that the input reference pattern is first filtered through the first level cache structure, prior to being passed on to the second level cache. The time dependent miss rate of the first level cache, Eq. 6 for a direct mapped cache or Eq. 11 for a set associative cache, becomes the reference pattern for the second level cache. Now the probability of finding an entry in the second level cache after being re-referenced can be calculated in a similar manner to our analysis for the first level cache. For a direct mapped cache the time dependent miss rate is

$$M_2(t) = R_o t^{-\beta}(1-e^{-\frac{Mave_1}{N_1}t})(1-e^{-\frac{Mave_2}{N_2}t}), \tag{14}$$

where the number of cache lines, N_m, and the average miss rate, M_{ave_m}, are explicitly shown as different for each cache level, m. Again, the number of cache lines is given in terms of the cache size and line size, $N_m = A_m / A_{l_m}$, for each level of the cache hierarchy.

We follow a similar procedure to obtain the total miss rate for the second level cache, as we did for the first level cache. One takes Eq. 14, makes the substitution $T = \frac{M_{ave_2}}{N_2}t$, where

$N_2 = A_2/A_{l_2}$, and integrates over all time from 0 to ∞. This time, after substitution, the integral is still not independent of the cache sizes. However, it is still solvable. The result is

$$M = \frac{R_o}{(A_{l_2} M_{ave_2})^{1-\beta}} \beta\Gamma(-\beta)\left[1-(1+\frac{A_2}{A_1}\frac{A_{l_1}}{A_{l_2}}\frac{M_{ave_1}}{M_{ave_2}})^{\beta-1}+(\frac{A_2}{A_1}\frac{A_{l_1}}{A_{l_2}}\frac{M_{ave_1}}{M_{ave_2}})^{\beta-1}\right]A_2^{1-\beta}. \qquad (15)$$

In looking at this equation it is important to note that the dominant dependence on cache size is the last term, $A_2^{1-\beta}$. This is the same dependence as for the first level cache. The large expression in brackets is a correction term that depends on the ratio of the cache sizes for the first and second levels. All of the initial terms are simply constants, which set the scale.

For a typical design case the second level cache is considerably larger than the first level cache, $A_2 \gg A_1$. In addition that means that $M_{ave_1} \gg M_{ave_2}$. The large expression in brackets in Eq. 15, the correction factor for a second level direct mapped cache, becomes approximately unity. Therefore, the dependence of cache miss rate as a function of cache size for the second level cache, while not strictly a power law any longer, comes very close to the same power law as for a first level cache.

A set associative second level cache can be handled in much the same way as the first level set associative cache was handled. If we designate the set associativities of a cache level as s_m, Eq. 14 generalizes to

$$M_2(t) = R_o t^{-\beta}(1-e^{-\frac{M_{ave_1}}{N_1}t})^{s_1}(1-e^{-\frac{M_{ave_2}}{N_2}t})^{s_2}. \qquad (16)$$

The only differences are the set associativity exponents. Using the same procedures discussed for the first level set associative cache and the second level direct mapped cache, Eq. 16 can be integrated over time to give the miss rate as a function of cache size. The result is

$$M = \frac{R_o}{(s_2 A_{l_2} M_{ave_2})^{1-\beta}} \beta\Gamma(-\beta) f(\frac{A_2}{A_1}, \frac{A_{l_1}}{A_{l_2}}, \frac{M_{ave_1}}{M_{ave_2}}, \beta, s_1, s_2) A_2^{1-\beta}, \qquad (17)$$

where f is a very complex function of the parameters listed. It is only practical to evaluate f numerically. Note that the dominant dependence of the miss rate on cache size is still the last term, $A_2^{1-\beta}$, for the second level set associative cache. Again, it turns out that as long as the second level cache is much larger than the first level cache, the correction factor, f, simply becomes a constant, largely independent of the cache sizes.

Figure 9 shows the miss rate for a second level 4-way set associative cache as a function of the cache size along with similar data to that shown in Fig. 8 for the L1 cache. For the L2 data the first level cache was 4 kB direct mapped. Also shown is a plot of Eq. 17. The curves are still straight lines because the correction term, f, in Eq. 17 is effectively a constant. The fit between

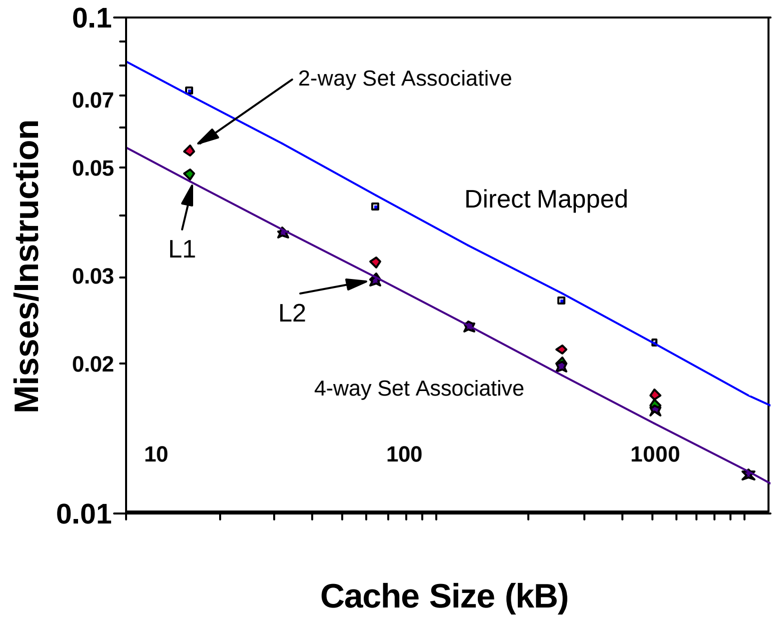

Figure 9: The cache miss rate as a function of cache size for direct mapped and set associative L1 caches and a set associative L2 cache for OLTP-1.

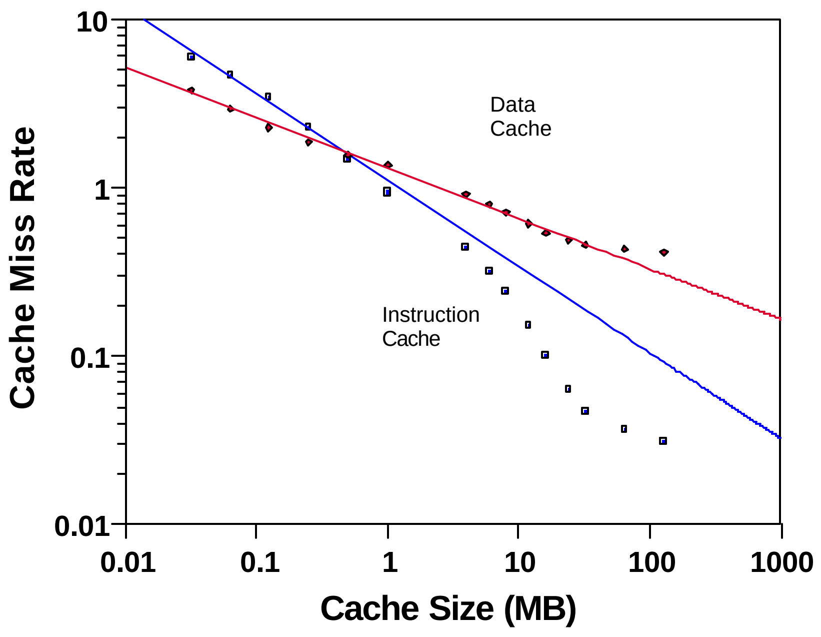

Figure 10: The cache miss rate as a function of cache size for a set associative L2 cache, separating out the data and instruction misses for a database workload.

our theory and the measured curves is quite good. This also highlights the prediction that the curves for all levels of cache, as well as all associativities have the same slope, $1-\beta$.

In order to show just how ubiquitous this power law dependence can be, we show data obtained from hardware in Fig. 10. These are data for an L2 cache for a database workload. It is clear that for data references, the power law dependence is observed for more than 3 orders of magnitude in the cache size. The saturating effects for very large caches are artificial and due to the finite size of the data sample. On the instruction side there is a deviation from the simple power law dependence for cache sizes above 10 MB, due to the smaller working set size of instructions.

8. Special Cache Reference Patterns (Cyclical Programs)

All of the results shown so far have been applicable to large complex programs running on the processor. These programs show a power law dependence for the cache re-reference pattern. As we have already discussed, small programs, which fit in a particular cache size have a very different dependence. In that case as long as the cache is larger than the footprint of the workload, the miss rate will be very small. In effect curves of miss rate versus cache size show a saturating behavior. Figure 11 shows several workload for which this behavior is evident. The cache miss rate decreases with cache size, as before, until the workload fits within the cache, and the miss rate saturates.

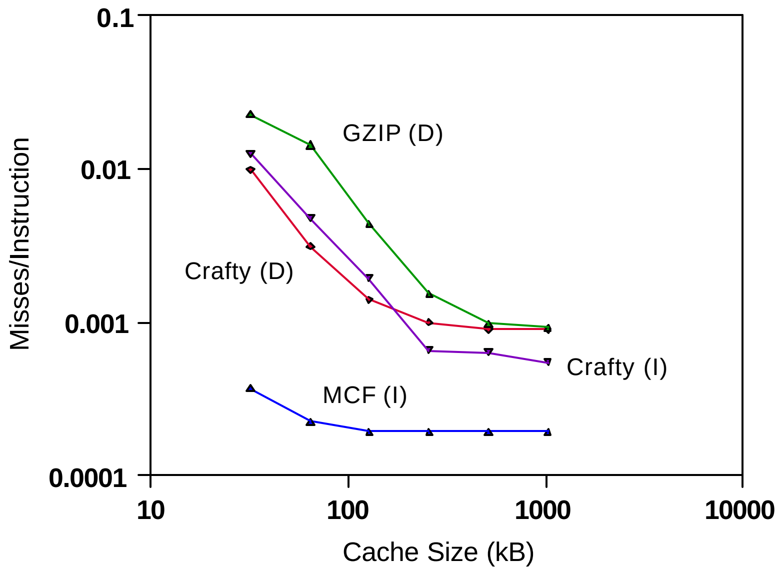

Figure 11: The variation of cache miss rate with cache size for some SPEC workloads. Saturation for large caches is clearly observed.

A far more interesting case involves cyclical programs, that is programs which are basically one big execution loop. Figure 12 shows the cache re-reference patterns for both the instructions and data for such a workload. The workload is actually a cycle accurate simulator and the loop is the repetition of each execution cycle as instructions are passed through the modeled processor. The large peaks in the re-reference pattern near t = 50000 cycles are due to the cyclical nature of the program. The peak is much larger for the instruction cache than for the data cache as might be expected. Instructions are completely repetitious, whereas only some fraction of the data accesses will be repeated exactly.

One can approximate the re-reference pattern as a sum of contributions from the peaks and from the background. The cache behavior arising from the background references will mirror the large complex system behavior, that has been our focus until now. We can analyze the consequences of the re-reference peaks by approximating the peaks as delta functions, $R_p\delta(t_p)$.

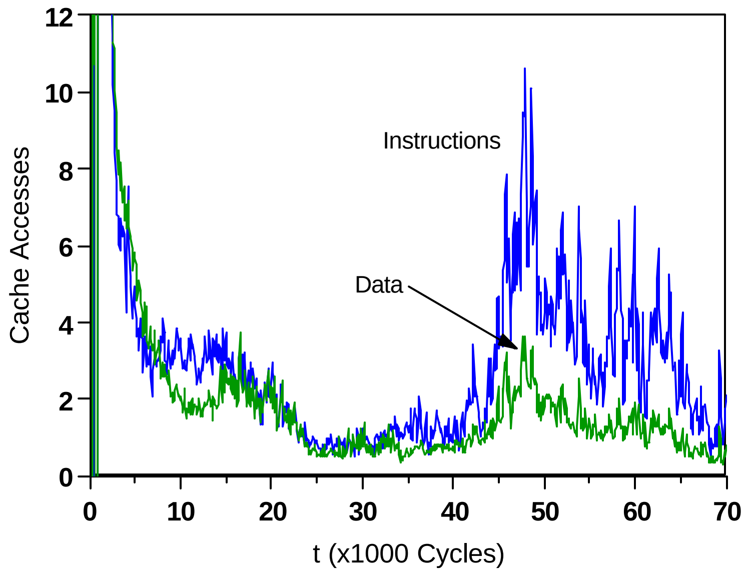

Figure 12: The cache re-reference pattern for the simulator workload. The large peak for both data and instructions is clearly evident.

We have assumed that the integrated size of the peak is R_p, and that the repeat time is t_p. We pick this delta function approximation for mathematical simplicity; it is easily handled in an integral. Another approximation, which might be used, is a Gaussian, but the mathematics are much more difficult.

Using this approximation in Eq. 7 gives the following result

$$M = \int_0^\infty [R_o t^{-\beta} + R_p\delta(t_p)](1 - e^{-\frac{M_{ave}}{N}t})dt. \tag{18}$$

The equation now contains 2 terms in the re-reference time, which can be separated, that is we can rewrite Eq. 18 as

$$M = \int_0^\infty R_o t^{-\beta}(1 - e^{-\frac{M_{ave}}{N}t})dt + \int_0^\infty R_p \delta(t_p)(1 - e^{-\frac{M_{ave}}{N}t})dt. \tag{19}$$

We already know how to handle the first integral, it was done in Section 3. The second integral is easily handled because of the properties of the delta function; the integral of any function times the delta function is simply $\int f(t)\delta(x)dt = f(x)$. The overall miss rate is then given by

$$M = \frac{R_o}{(A_l M_{ave})^{1-\beta}}\Gamma(\beta - 1)A^{1-\beta} + R_p(1 - e^{-\frac{M_{ave}A_l}{A}t_p}), \tag{20}$$

where we have substituted $N = A/A_l$. The second term gives a sudden dropoff in the miss rate for a cache size, $A = M_{ave}A_l t_p$. That is when the entire loop fits within the cache, the miss rate drops precipitously. This dependence is evident in the miss rates for both the instruction and data caches as shown in figure 13, where a drop off of more than a factor of 10 is observed. On the

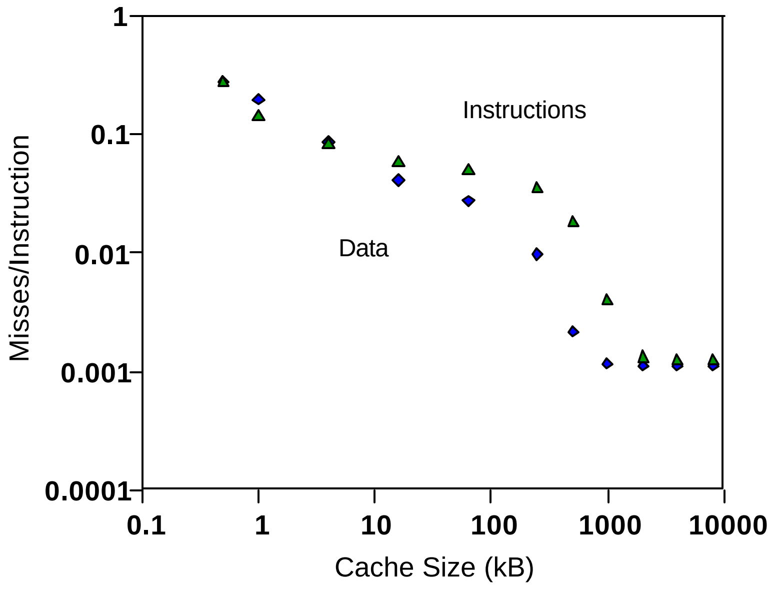

Figure 13: The variation of cache miss rate with cache size for the simulator workload.

instruction side the first term in Eq. 20 is small and the second term dominates the dependence. On the data side both terms are important, so the straight line dependence, which arises from the first term, is more clearly evident on the small cache size limit of the curve. The dropoff for the middle range is dominated by the second term. The saturating behavior for large cache sizes may be artificial, due to the finite size of the trace analyzed.

9. Discussion

Our most important result was obtained in section 3. In it we show that by combining the typical re-reference time pattern for cache accesses along with a simple probabilistic analysis of cache entry replacement, we obtain the inverse square root law for cache miss rate as a function of cache size. The important point is that the inverse square root law is a consequence of the temporal reference pattern of large complex workloads. The value of the exponent for a first level cache only depends on the nature of the workload. It is independent of the microarchitecture of the processor. The prefactor depends on the microarchitecture.

Since the input to a second level cache depends on both the temporal reference pattern of a workload and the filtering properties of the first level cache, the inverse square root law becomes only an approximation. That is, the power law dependence is approximate and no single exponent is obtained. The specifics of the curves then depend on both the workload and the microarchitecture. These have been detailed in terms of cache sizes, associativities and cache line sizes. For the cases we have discussed, the deviations from the simple power law are small and the results can be treated as an inverse square root law with the appropriate exponent.

It is important to note that the predicted, and observed, exponents for the power law for cache misses as a function of cache size are only dependent on the temporal reference pattern of the workload. The exponent, and hence the slopes of the curves on the log-log plots, are equal for each level of the cache hierarchy, as well as for both direct mapped and set associative caches. Those architectural features only affect the magnitude of the cache misses.

One should further note that the magnitude of the cache miss rate depends only on the temporal reference pattern of the workload and a statistical treatment of cache entry replacement, which in turn is dependent on the cache architecture. Therefore, for a given workload, after the temporal dependence of the references has been determined, the miss rates for all of the caches in the hierarchy are completely determined. We accurately predict both the magnitude of the cache miss rates, as well as the functional dependence on cache size. In this paper we have focused on the dependence on cache size, but we also predict the dependence on the degree of set associativity and the relative miss rates for each cache level in the hierarchy.

Now let us consider a more fundamental question. We have observed that the re-reference pattern for cache references obeys a power law. In thinking about the problem, one would assume that the probability of referencing an entry is highest for short time intervals and falls off for large times. One may well have expected that the dependence would have been a decaying exponential. In fact, that was our initial assumption. Since our results depend directly on the form of this drop-off, the fact that a power law governs the time dependence is crucial to our results; and the whole observation of an approximate inverse square root law. It is clear from data as shown in figure 1, that the power law dependence indeed governs this behavior. It is a much better fit than an exponential dependence.

This leads us to speculate as to why the power law is observed. Exponential dependencies typically result from random processes. Random processes are events like the probability of coin tosses and radioactive decay. Any of these types of processes tend to give exponential behavior. In fact that is why the probability of a cache entry being removed from the cache results in an exponential. However, the behavior of a large complex program is not random but rather is deterministic but very complex. The complex nature of a program means that, among other things, small changes during the execution can lead to a completely different program flow. This has all of the earmarks of chaotic behavior in large complex nonlinear systems. We note that chaos theory results in quantities that tend to obey power law dependencies. It might just be that this is the ultimate explanation of both the re-reference pattern observed and, therefore, the approximate inverse square root law.

One such model has been proposed to produce synthetic traces [17]. The model employs a "hyperbolic random walk" through memory address space. This model generates addresses, which are generally local in nature, but have a finite probability of large deviations. It uses a probability function, which is a power law, and hence nonlinear. This a particular nonlinear model, which produces a synthetic memory trace that mirrors the statistics of real traces, and the same approximate inverse square root law for the miss rate as a function of cache size as is observed for actual traces.

10. Summary

We have examined the typical behavior of caches in a processor. By combining the temporal dependence of the cache re-reference pattern with a statistical treatment of the cache replacement algorithm, we predict the inverse square root power law dependence of the cache miss rate on the cache size. Therefore, the observed dependence of miss rate on cache size arises from the temporal dependence of cache references from a workload. This is not an intuitive result.

Acknowledgements

We would like to thank Hanno Ulrich for many stimulating and helpful discussions**.**

References

[1] R. Kalla, B.Sinharoy, and J. Tendler. "IBM Power5 Chip: A dual-core multi-threaded processor", *IEEE Micro*, vol. 24(2), pp. 40-47, 2004.

[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.

[3] http://www.amd.com, Technical Documentation. "AMD Opteron Product Data Sheet", *Publication number 23932*, 2004.

[4] http://www.sun.com/processors/manuals/USIV_v1.0.pdf. "UltraSPARC IV Processor", *User's Manual Supplement*, Version 1.0, 2004.

[5] C. K. Chow. "On Optimization of Storage Hierarchies", *IBM Journal of R & D*, vol. 18, pp. 194 - 203, 1974.

[6] C. K. Chow. "Determination of Cache's Capacity and its Matching Storage Hierarchy", *IEEE Transactions on Computers*, vol. c-25, pp. 157 - 164, 1976.

[7] J. S. Harper, D. J. Kerbyson and G. R. Nudd. "Efficient Analytical Modelling of Multi-Level Set-Associative Caches", *Proceedings of the International Conference HPCN Europe '99'*, vol. 1593, pp. 473 - 482, 1999.

[8] S. Przybylski, M. Horowitz and J. Hennessy. “Performance Tradeoffs in Cache Design”, *Proceedings of the 15th Annual International Symposium on Computer Architecture*, pp. 290 - 298, 1988.

[9] S. Przybylski, M. Horowitz and J. Hennessy. “Characteristics of Performance-Optimal Multi-Level Cache Hie rarchies”, *Proceedings of the 16th Annual International Symposium on Computer Architecture*, pp. 114 - 121, 1989.

[10] G. S. Rao. “Performance Analysis of Cache Memories”, *JACM*, vol. 25, pp. 378 - 395, 1978.

[11] J. H. Saltzer. “A Simple Linear Model of Demand Paging Performance”, *CACM*, vol. 17, pp. 181 - 186, 1974.

[12] A. J. Smith. “Cache Memories”, *Computing Surveys*, vol. 14, 473 - 528, 1982.

[13] M. H. Macdougall. “Instruction-level Program and Processor Modeling”, *Computer*, vol. 7, pp. 14 - 24, 1984.

[14] A. Hartstein, T. R. Puzak, V. Srinivasan and P. G. Emma. “Cache Miss Behavior: Is It $\sqrt{2}$?”, *Proc. of the ACM International Conference on Computing Frontiers*, pp. 313 - 320, 2006.

[15] J. P. Singh, H. S. Stone and D. F. Thiebaut. “A Model of Workloads and Its Use in Miss-Rate Prediction for Fully Associative Caches”, *IEEE Transactions on Computers*, vol. 41, pp. 811 - 825, 1992.

[16] A. Hartstein and T. R. Puzak. “Optimum Power/Performance Pipeline Depth”, *Proc. of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 117 - 125, 2003.

[17] D. Thiebaut, J. L. Wolf and H. S. Stone. “Synthetic Traces for Trace-Driven Simulation of Cache Memories”, *IEEE Transactions on Computers*, vol. 41, pp. 388 - 410, 1992.