PERFORMANCE IMPLICATIONS OF TOLERATING CACHE FAULTS

by

Farid Pour and Mark D. Hill

Computer Sciences Technical Report #991

January 1991

Performance Implications of Tolerating Cache Faults[†]

Farid Pour
Mark D. Hill

Computer Sciences Department
University of Wisconsin — Madison
1210 West Dayton Street
Madison, Wisconsin 53706

ABSTRACT

Microprocessors are increasingly incorporating one or more on-chip caches. These caches are occupying a greater share of chip area, and thus may be the locus of manufacturing defects. Some of these defects will cause faults in cache tag or data memory. These faults can be tolerated by disabling the cache blocks that contain them. This approach saves chips with defects without requiring on-chip caches to have redundant row or columns or to use error correcting codes. Disabling blocks, however, typically increases a cache's miss ratio.

This paper investigates how much cache miss ratios increase when blocks are disabled. It shows how the mean miss ratio increase can be characterized as a function of the miss ratios of related caches, develops an efficient approach to calculating the exact distribution of increases from all fault patterns, and applies this approach to the ATUM traces [1]. Results for the ATUM traces indicate that the mean relative degradation in miss ratio from a few faults decreases with increasing cache size, and is negligible (< 2% per defect) unless a set is completely disabled by faults. The maximum relative degradation in miss ratio is also acceptable (< 5% per fault) if no set is entirely disabled.

Index Terms: computer architecture, cache performance, trace-driven simulation, fault tolerance, on-chip caches, microprocessors.

1. Introduction

Commercial and academic microprocessor architectures are increasingly incorporating caches on the processor chip itself to hide off-chip latencies [3, 6, 8, 12]. These *on-chip caches* are currently relatively small, but the trend is toward larger sizes to hide relatively slower off-chip memory speeds; thus, an increasing portion of chip area is being devoted to the memory (tags and blocks) of the cache. As cache chip area becomes large, so will the fraction of manufacturing defects that land in the cache.

A manufacturing defect is said to cause a *fault* in a cache if it impares the correct operation of the cache. We will study those faults that make a bit in the cache unable to retain the value written to it, but

[†] This work is supported in part by the National Science Foundation (MIPS-8957278 and CCR-8902536), A.T.& T. Bell Laboratories, Cray Research Foundation and Digital Equipment Corporation.

that do not otherwise perturb the operation of the cache. A fault causes an *error* if it causes the system to enter a logical state other than the one intended. We can prevent faults in an on-chip cache from causing errors by (1) discarding chips with such faults, (2) using redundant memory, or (3) disabling cache blocks that contain faults. The advantage of discarding chips, method (1), is that it works for any defect. Its disadvantage, however, is that it increases systems cost by reducing yield.

Redundant memory, method (2), can be employed to tolerate faults in, at least, three, ways: (a) add extra memory words (rows) that are selected instead of faulty ones, (b) add extra bits per word (columns) that are selected in instead of faulty ones, or (c) add extra bits per word that store an error correcting code. The advantage of each of these approaches is that they work for any memory. Two disadvantages, however, are (a) the cost or opportunity cost of the extra memory, and (b) a possible increase in memory access time required to implement them.

Disabling cache blocks that contain faults, method (3), is applicable to cache only. Caches are buffers used to hold data from recently-used parts of main memory [13]. Data is usually transferred from main memory in aligned *blocks* (also called *lines*). The number of bytes in a block is the *block size*. A block is stored in a cache with memory that holds its contents, its tag (part of its main memory address) and some state bits, including a *valid bit* that indicates whether a block is present. The blocks in a cache are usually divided into *sets*. Every block maps to one set, so only blocks in that set must be searched on a reference. A cache with n blocks in a set has *associativity* n and is called direct-mapped if $n = 1$, fully-associative if n equals the number of blocks in the cache, or n-way set-associative otherwise. *Cache size* is the number of blocks a cache can hold multiplied by the block size. Finally, the *number of sets* is cache size divided by the product of block size and associativity. Caches are usually characterized by their block size, associativity and cache size.

Method (3) exploits the fact that all memory in a cache is redundant memory, since a cache merely keeps a copy of data from main memory. Thus, instead of building redundancy on top of the redundant memory in a cache, this method just causes the cache to not use blocks that contain faulty bits. This can be implemented with a mechanism to test blocks and a second valid bit that is used to mark faulty blocks permanently invalid, as was proposed by Patterson, et al.[1] [11]. Consider, for example, a 64K-byte cache with associativity four and block size 32 bytes. This cache has 512 sets. If there is a fault in one block, the set containing it will behave as if it has associativity three, while the other 511 sets will behave normally. If all four blocks in one set were disabled, references to that set would be handled as if there was no cache (or the chip could be discarded).

Disabling cache blocks offers two advantages and two disadvantages over using redundant memory. The first advantage is that, unlike redundant memory, implementing a second valid bit does not increase cache access time on a hit. The second advantage is that it allows all non-faulty blocks to be used, whereas redundant memory is only used to replace faulty memory. The disadvantages of this method are that it can increase both the mean and variability of cache miss ratio, whereas using redundant memory leaves the miss ratio unchanged.

Nevertheless, disabling cache blocks can lead to a better average access time than the use of redundant memory. The average access with redundant memory, method (2), can be modeled as:

$$t_{(2)} = (t_{cache} + t_{\Delta}) + m\, t_{memory},$$

[1] This mechanism to be invoked when a chip is tested or when the system is reset.

where t_{cache} is the access time to a cache without redundant memory, t_Δ is the additional time penalty to implement it, m is the miss ratio of the cache, and t_{memory} is the access time for main memory. For disabling cache blocks, method (3):

$$t_{(3)} = t_{cache} + m\,(1+\delta)\,t_{memory},$$

where t_{cache}, m, and t_{memory} are the same as above and δ is the relative increase in the cache miss ratio due to faults. Rearranging the terms, we find that disabling blocks is superior to redundant memory when $t_\Delta > \delta\, m\, t_{memory}$. If we assume that $t_{cache} > m\, t_{memory}$, a pessimistic assumption for all but the smallest caches, then disabling blocks leads to better average access times whenever:

$$t_\Delta > \delta\, t_{cache}.$$

As we will see, values for δ from one or two faults are typically two to four percent, thus making disabling cache blocks is an attractive option. Of course, in cases where a cache is fault-free, disabling cache blocks is clearly faster, since $\delta = 0$ in that case. Furthermore, it is likely that performance of disabling cache blocks can be significantly improved through the use of *victim caches* [7].

To the best of our knowledge, the only paper to do a detailed investigation of the effect on miss ratios of disabling cache blocks is by Sohi [14]. Sohi investigated the degradation in cache performance by randomly injecting faults into the cache and then running a trace-driven simulation. Results for each cache configuration are the average of several simulations with different fault patterns. He presents results for the percentage of faulty blocks ranging from 0% to 50% of the blocks in the cache for cache sizes of 256-bytes, 1K-byte, and 8K-bytes. Three cache organizations (direct mapped, two-way set associative, and fully associative) and three block sizes (8-byte, 16-byte, and 32-byte blocks) were simulated.

This paper extends Sohi's work in several ways. This paper shows that the distribution of miss ratio increases can be calculated from LRU distance probabilities for each set, while Sohi's paper did not consider this issue. One implication of our equations is to confirm the intuition that the mean of the miss ratio for a cache with s sets and a single fault is equivalent to $s - 1$ sets seeing an unperturbed cache and a single set seeing an associativity decreased by one.[2] For example, let m_0 be the miss ratio for a 64K-byte cache with associativity four, and block size 32 bytes and no faults and m_1 be the miss ratio for a 48K-byte cache with associativity three, and block size 32 bytes and no faults. Then the expected miss ratio for 64K-byte cache with associativity four, and block size 32 bytes and one fault is $(511/512)\times m_0 + (1/512)\times m_1$.

This paper also shows how *all-associativity simulations* [5] can be extend to collect information for determining the effect of all possible patterns of faults to caches with many associativities and sizes (but one block size) *in one pass through an address trace*. In Sohi's paper, on the other hand, a simulation was preformed per fault pattern per cache configuration. For this reason, Sohi estimates the mean miss ratio increases from a small fraction of the possible fault patterns[3] and limits the cache configurations examined. Our approach, on the other hands, allows us to calculate the exact mean, maximum and standard deviation of miss ratio increases for a small number of faults and wider range of cache configurations. We concentrate on a small number of faults, because we believe that chips with large number of faults in the cache will frequently have faults in other critical resources, and thus will be discarded regardless.

---

[2] Assume that the miss ratio of a cache with associativity zero is one.

[3] For $m \ll s$, the number of ways to place m faults in s sets in $O(s^m)$

Results for the ATUM traces [1] indicate that the *mean* relative degradation in miss ratio from a few faults decreases with increasing cache size, and is usually small (< 5% per fault). Furthermore, if no set is completely disabled, mean degradation for large caches is negligible. Consequently, it is likely that the effective access time of a cache with some blocks marked faulty will be less than that of a cache using redundant memory.

The *maximum* relative change in miss ratio for a single cache fault – or for two cache faults in distinct sets – is acceptable (< 5%) if the associativity of the cache is 4 or greater and the block size is 8 or 16 bytes. Larger block sizes suffer greater penalties with permanent block invalidations. With a direct-mapped cache, however, there is a probability (albeit small with a large number of sets) that the executing program heavily references the faulty block(s), severely degrading the cache's performance. We expect that the overall impact of this worst-case behavior will not be significant for machines used to run many different programs.

The rest of the paper is divided up as follows. Section 2 develops equations for the impact of cache faults on the miss ratios. Section 3 discusses how data for many fault patterns in many cache configurations can be gathered with a single pass through an address trace. Section 4 presents the results of the investigation, and Section 5 concludes our discussion.

2. The Impact of Faulty Blocks

We now turn to the impact on the cache miss ratio of marking faulty cache blocks as unusable. We show how the impact can be expressed from per-set simulation data for any pattern of faults and then derive equations for some simple cases. These equations show what data needs to be gathered in trace-driven simulation so that miss ratios for any fault pattern can be calculated.

Note that the following derivation makes no assumptions about the distribution of the reference stream. Assume a cache has s sets labeled 0 through $s-1$, and is referenced by a dynamic reference stream of R references. Assuming all blocks within a set are ordered according to some *stack replacement algorithm* such as LRU [10], define $D_i(j)$ to be the number of references to the j-th block in the i-th set and $D(j)$ be the number of references to the j-th block in any set. Then,

$$D(j) = \sum_{i=0}^{s-1} D_i(j), \quad j > 0.$$

Let $M(n)$ be the number of misses in an n-way associative cache with s sets accessed by R references. Since a miss occurs when a reference is not to one of the first n blocks of a set,

$$M(n) = R - \sum_{j=1}^{n} D(j), \quad n > 0$$

$$M(0) = R.$$

Further define $\mathbf{f}$ to be an s-element fault vector $(f_0, f_1, \cdots, f_{s-1})$, where f_i is the number of faulty blocks in set i. Then the additional misses induced by $\mathbf{f}$, $\Delta(n, \mathbf{f})$, and the number of misses in an n-way associative cache with faults according to $\mathbf{f}$, $M(n, \mathbf{f})$ are defined as:

$$\Delta(n, \mathbf{f}) = \sum_{i=0}^{s-1} \sum_{j=0}^{f_i - 1} D_i(n-j), \quad n > 0, \quad \text{and}$$

$$M(n, \mathbf{f}) = \Delta(n, \mathbf{f}) + M(n), \quad n > 0.$$

Finally, the miss ratio of an n-way associative cache with fault vector $\mathbf{f}$, $m(n, \mathbf{f})$, can be be calculated by dividing the number of misses by the number of references R, and the *relative change* in miss ratio in an n-way associative cache due to $\mathbf{f}$, $\delta(n, \mathbf{f})$, can be calculated by dividing the additional misses induced by $\mathbf{f}$ by the number of misses with no faults:

$$m(n, \mathbf{f}) = \frac{M(n, \mathbf{f})}{R}, \quad n > 0, \quad \text{and}$$

$$\delta(n, \mathbf{f}) = \frac{M(n, \mathbf{f}) - M(n)}{M(n)} = \frac{\Delta(n, \mathbf{f})}{M(n)}, \quad n > 0.$$

Thus, the miss ratio of a cache with s sets and associativity n with *any* pattern of faults can be determined from $D_i(j)$ for $i = 0, s-1$ and $j = 1, n$. Thus, trace-driven simulation can be done before fault patterns are selected.

2.1. Single Faults

Here we develop equations for the mean, maximum and standard deviation of absolute miss ratio increase from a single fault. Single fault vectors $\mathbf{f}_1$ are a special case of a fault vector $\mathbf{f}$ where

$$f_i = 1, \quad 0 \le i \le s-1, \quad \text{and}$$

$$f_k = 0, \quad k \ne i.$$

The effect of a fault in set i of an n-way associative cache is to cause a cache miss on references to block n in set i, which would not have missed without the fault. The other $n-1$ blocks in the set with the fault are unaffected, as are the remaining $s-1$ sets in the cache. The miss ratio with one fault is thus:

$$M(n, \mathbf{f}_1) = M(n) + D_i(n).$$

This implies that the additional misses induced by the fault vector $\mathbf{f}_1$ are:

$$\Delta(n, \mathbf{f}_1) = D_i(n).$$

Since the number of sets s in a cache can be very large, it is worthwhile to distill the distribution of $\Delta(n, \mathbf{f}_1)$ across all sets i. Assume that the fault is equally likely to be present in any set i. Then the standard deviation (STD), mean (E), and maximum (MAX) $\Delta(n, \mathbf{f}_1)$ are:

$$STD\left[\Delta(n, \mathbf{f}_1)\right] = \left[\sum_{i=0}^{s-1}\left[\frac{D_i(n)}{s}\right]^2 - \left[\sum_{i=0}^{s-1}\frac{D_i(n)}{s}\right]^2\right]^{\frac{1}{2}},$$

$$E\left[\Delta(n, \mathbf{f}_1)\right] = \frac{1}{s}\sum_{i=0}^{s-1} D_i(n) = \frac{D(n)}{s}, \quad \text{and}$$

$$MAX\left[\Delta(n, \mathbf{f}_1)\right] = \max_i\left[D_i(n)\right]. \tag{1}$$

Several substitutions may be made to aid in understanding the arithmetic means of Δ (additional misses induced by $\mathbf{f}$), m (miss ratio with fault vector $\mathbf{f}$), and δ (relative change in miss ratio due to $\mathbf{f}$):

$$E\left[\Delta(n, \mathbf{f}_1)\right] = \frac{D(n)}{s} = \frac{M(n-1) - M(n)}{s},$$

$$E\left[m(n, \mathbf{f}_1)\right] = m(n) + \frac{m(n-1) - m(n)}{s} = \frac{s-1}{s} m(n) + \frac{1}{s} m(n-1), \quad \text{and} \tag{2}$$

$$E\left[\delta(n, \mathbf{f}_1)\right] = \frac{1}{s} \frac{D(n)}{M(n)} = \frac{1}{s} \frac{m(n-1) - m(n)}{m(n)} = \frac{1}{s}\left[\frac{m(n-1)}{m(n)} - 1\right]. \tag{3}$$

Equation (2) confirms the intuition that the expected mean of the miss ratio with a single fault is equivalent to $s - 1$ sets seeing an unperturbed cache and a single set seeing an associativity decreased by 1. Equation (3) above indicates that the expected relative change in miss ratio will be small to the extent that:

i) s is large, or

ii) $(m(n-1) - m(n))$ is small, as we expect with $n \geq 2$.

The standard deviation will not be pursued further. Suffice it to note that for a large number of faults, exhaustive and thus quite expensive simulations must be done to determine the standard deviation.

2.2. Double Faults

The case of two faults may be subdivided into the case of the faults occurring anywhere (denoted *no restrictions*) and the case where the faults are restricted so that there is at most one fault in a given set (denoted *different sets*).

2.2.1. Double Faults, Different Sets

Double fault vectors $\mathbf{f}_{2,1}$ are a special case of a fault vector $\mathbf{f}$ with two faults where:

$$f_i = f_j = 1, \quad 0 \leq i, j \leq s-1, \quad i \neq j, \quad \text{and}$$

$$f_k = 0, \quad k \neq i, \quad k \neq j.$$

In words, the vector $\mathbf{f}_{2,1}$ contains two faults, each in a distinct set. This effectively doubles the additional misses induced by the fault vector:

$$\Delta(n, \mathbf{f}_{2,1}) = D_i(n) + D_j(n).$$

Assume that the first fault is equally likely to land in any set i and the second fault is equally likely to land in any other set. Then,

$$\begin{aligned} E\left[\Delta(n, \mathbf{f}_{2,1})\right] &= \frac{1}{s(s-1)} \sum_{i=0}^{s-1} \sum_{j=0, j\neq i}^{s-1} \left[D_i(n) + D_j(n)\right] \\ &= \frac{D(n)}{s} + \frac{1}{s(s-1)} \sum_{j=0}^{s-1} (s-1) D_j(n) \\ &= \frac{2D(n)}{s} = \frac{2\left[M(n-1) - M(n)\right]}{s} \\ &= 2E\left[\Delta(n, \mathbf{f}_1)\right]. \end{aligned}$$

Similarly,

$$E\left[m(n, \mathbf{f}_{2,1})\right] = \frac{s-2}{s} m(n) + \frac{2}{s} m(n-1), \quad \text{and}$$

$$E\left[\delta(n, \mathbf{f}_{2,1})\right] = 2E\left[\delta(n, \mathbf{f}_1)\right]. \tag{4}$$

In general, the expected miss ratio and relative change in miss ratio for a fault vector with g faults, no two of which map to the same set ($0 \le g \le s$), are:

$$E\left[m(n, \mathbf{f}_{g,1})\right] = m(n) + \frac{\left[m(n-1) - m(n)\right] g}{s} = \frac{s-g}{s} m(n) + \frac{g}{s} m(n-1) \tag{5}$$

$$E\left[\delta(n, \mathbf{f}_{g,1})\right] = g\, E\left[\delta(n, \mathbf{f}_1)\right].$$

2.2.2. Double Faults, No Restrictions

Double fault vectors $\mathbf{f}_{2,2}$ are the general case of a fault vector $\mathbf{f}$ with two faults where:

$$f_i + f_j = 2, \quad 0 \le i, j \le s-1, \quad i \ne j, \quad \text{and}$$

$$f_k = 0, \quad k \ne i, \quad k \ne j.$$

In other words, both faults may or may not occur in the same set. Since necessarily $f_i, f_j \le n$, this section assumes $n > 1$. Assume that the first fault is equally likely to land in any set i, while the second fault is independent of the first and equally likely to land in any set i. Then,

The additional misses induced by the fault vector $\mathbf{f}_{2,2}$ can then be described by:

$$\Delta(n, \mathbf{f}_{2,2}) = \begin{cases} D_i(n) + D_j(n), & i \ne j \\ D_i(n) + D_i(n-1), & i = j \end{cases}$$

The expected means of Δ, δ, and m are then[4]:

$$E\left[\Delta(n, \mathbf{f}_{2,2})\right] = \frac{1}{s^2}\left[\sum_{i=0}^{s-1}\sum_{j=0, j\ne i}^{s-1}\left(D_i(n) + D_j(n)\right) + \sum_{i=0}^{s-1}\left(D_i(n) + D_i(n-1)\right)\right]$$

$$= \frac{1}{s^2}\left[\sum_{i=0}^{s-1}\sum_{j=0, j\ne i}^{s-1}\left(D_i(n) + D_j(n)\right)\right] + \frac{1}{s^2}\left[\sum_{i=0}^{s-1}\left(D_i(n) + D_i(n-1)\right)\right].$$

For $s \gg 1$,

$$E\left[\Delta(n, \mathbf{f}_{2,2})\right] \approx E\left[\Delta(n, \mathbf{f}_{2,1})\right] + \frac{1}{s^2}\left(D(n) + D(n-1)\right)$$

$$E\left[m(n, \mathbf{f}_{2,2})\right] = m(n) + \frac{E\left[\Delta(n, \mathbf{f}_{2,2})\right]}{R}$$

$$\approx m(n) + \frac{2D(n)}{s} + \frac{D(n) + D(n-1)}{s^2} = m(n) + \frac{\left(2s+1\right) D(n) + D(n-1)}{s^2}$$

$$E\left[\delta(n, \mathbf{f}_{2,2})\right] \approx \frac{2E\left[\Delta(n, \mathbf{f}_1)\right] + \frac{1}{s^2}\left(D(n) + D(n-1)\right)}{M(n)}$$

$$= 2E\left[\delta(n, \mathbf{f}_1)\right] + \frac{1}{s^2}\left(\frac{m(n-2)}{m(n)} - 1\right). \tag{6}$$

---

[4] Define $\sum_{j=0, j\ne i}^{n} x_j \equiv \left[\sum_{j=0}^{n} x_j\right] - x_i$.

Lastly, define the fault vector $\mathbf{f_2}$ to be a fault vector $\mathbf{f}$ with faults according to:

$$f_i = 2, \quad 0 \le i \le s-1, \quad \text{and}$$

$$f_k = 0, \quad k \ne i.$$

I.e., both faults occur in the same set. Then, the expected relative change in miss ratio is:

$$E\left[\delta(n, \mathbf{f_2})\right] = \frac{1}{s}\frac{m(n-2)-m(n)}{m(n)} = \frac{1}{s}\left[\frac{m(n-2)}{m(n)} - 1\right].$$

2.2.3. Multiple Faults, No Restrictions

Multiple fault vectors are the general case of a fault vector $\mathbf{f}$ where the number of faults m are in the range $2 \le m \le sn$; thus, all the blocks in the set may be faulty. These cases do not allow for concise descriptions and require a probabilistic approach, since exhaustive simulation has time complexity of $O(s^m)$.[5]

To do these calculations, it is necessary to determine the probabilities of a certain number of faults occurring in a particular set given the total number of faults. Then, the expected miss ratio can be described by:

$$E\left[m(n, \mathbf{f})\right] = \frac{P[Y=0]m(n) + P[Y=1]m(n-1) + P[Y=2]m(n-2) + \cdots + P[Y=g]m(n-g)}{P[Y=0] + P[Y=1] + P[Y=2] + \cdots + P[Y=g]} \quad (7)$$

where $P[Y=i]$ is the probability that a particular set has i faults when all m faults are uniformly randomly distributed amongst s sets with associativity n and at most g faults are allowed in a particular set ($0 \le g \le n$). Probabilities were calculated by iteratively determining all possible ways to distribute f faults over s sets (a partitioning problem), and then using multinomials to calculate how many ways a particular distribution may occur.

3. Methodology

In the last section, we showed how the $D_i(j)$'s, the number of references to the j-th block in the i-th of s sets, can be used to calculate the miss ratio for a single cache with any fault pattern. In this section, we shows how to determine the $D_i(j)$'s for many cache configurations with a single pass through an address trace. We also describe the address traces that we will use.

All-associativity simulation [5] is an algorithm that calculates the miss ratios for caches of many sizes and associativities with a single pass through an address trace, provided that all caches have the same block size, use least-recently-used and do no prefetching. It does this by calculating, $D(j)$'s, the number of references to the j-th block in any of s sets, for all associativities and number of sets of interest.

We extended all-associativity simulations to record $D_i(j)$'s instead of $D(j)$'s by expanding the metrics to keep track of which set is referenced. Thus, instead of recording that there is a reference to the j-th block in one of s sets, we record that there is a reference to the j-th block in i-th of s sets. Doing this simultaneously for all cache configurations being simulated expands storage for the metrics by a factor less than the largest number of sets considered, but the storage for the metrics is still smaller than the storage

---

[5] This approximation holds only for the range $0 < m < \frac{sn}{2}$.

for the contents of the caches being simulated. Furthermore, this change does not significantly affect simulation run-time.

Once each $D_i(j)$ is known, exact calculations of new (mean and maximum) miss ratios can be performed without additional trace-driven simulation. The faults are distributed in every possible way and the relevant statistics extracted as by the equations presented in section 2. Since a direct mapped 32K cache with a block size of 8 bytes contains 2^{12} sets, exhaustively calculating all miss ratios for three faults in the cache involves $(2^{12})^3 = 2^{36}$ calculations. Therefore, a probabilistic model was used for more than two faults. Nevertheless, our results are based on more cases than Sohi's, since we can calculate a new miss ratio by summing appropriate $D_i(j)$'s rather than performing a complete trace-driven simulation. Our results were validated by comparing, for a large number of cases, with the results of the exhaustive simulations.

Traces were obtained using the ATUM [1] microcode-generated trace method. Table 1 shows the number of instruction fetches, data reads, and data writes for each of the traces used, as well as a brief description of their origins. Due to the large number of traces, results can not be provided for each trace individually. Instead, for most purposes the traces will be simulated together, with cache flushes separating the individual traces. The combined trace will be denoted by *all*. Two traces representative of the others will be individually analyzed. Since the traces are each only about 400,000 references long, cache sizes ≥ 64K are not simulated since they are doubtlessly too large to provide reliable results.

4. Results

Figure 1 shows the mean and maximum miss ratios of the "all" simulation with one fault ($\mathbf{f}_1$) for various associativities and cache sizes. Unless otherwise indicated, the block size will be 16 bytes. The miss ratios ostensibly follow the behavior pattern for faultless cache miss ratios as reported in the literature [2,4,5]: for all associativities, the miss ratios decrease with increasing cache size and for a given cache size, the miss ratios decrease with increasing associativity. This comes as no surprise, since equation (2) predicts the mean miss ratio to be equivalent to a cache with $s - 1$ unperturbed sets and one set with an actual associativity of $n - 1$. Thus in the worst case ($n = 1$) the mean miss ratio will degrade by $\frac{1}{s}$.

Figure 1 by itself in not very informative; it should be plotted against the miss ratio with no faults. Instead, this information will be condensed into a single metric: the relative change in miss ratio in the next section.

Results for the standard deviation will not be provided. The general trend for the standard deviation in relative change of miss ratio with increasing cache size is for a steady, small decrease for associativities greater than one and for a constant or slightly increasing standard deviation for a direct mapped cache. The standard deviation in relative change of miss ratio decreases with greater associativity.

4.1. Relative Change in Miss Ratio

Figure 2 shows the relative change in miss ratios for the "all" simulation. Figure 2a reveals that as the associativity or number of sets increases, the mean relative change in miss ratio decreases. With an associativity of 1, an entire set is invalidated with a fault in the cache; thus, on average, $\frac{D(n)}{s}$ additional references will miss. As the associativity increases, the effect of the fault is to simply reduce the associativity of a particular set by 1. Locality of references reduces the impact of this with larger associativities; at the extreme of an

Name	Data Reads	Data Writes	Instruction Fetches	Description
dec0.000	106459	72500	183023	DECSIM behavioral simulation of some cache hardware
dec0.003	103906	73001	176533	Same as previous
fora.000	108979	79156	199799	FORTRAN compiler compiling airco.for
forf.000	108048	85845	207284	Two FORTRAN compilations: 4x1x5.for and linpack.for
forf.001	110027	73093	203595	Same as previous
forf.002	105131	91233	217509	Same as previous
forf.003	107969	69328	190915	Same as previous
fsxzz.000	78265	37840	123229	ULTRIX file system exerciser, 20 tasks
ivex.000	97335	41123	203510	Interconnect verify
macr.000	96904	57222	188702	Macro assembler assembling linpack2.mar
memxx.000	126660	99139	219050	ULTRIX memory exerciser, 10 tasks
mul2.001	112102	71845	201866	Multiprocessing 2 jobs: ALLC (Microcode address allocator, bit string inner loop) and SPIC (Spice simulating output buffer)
mul2.002	106329	74357	201941	Same as previous
mul2.003	96662	64884	205295	Same as previous
mul8.001	126139	74749	207538	Multiprocessing 8 jobs: Unknown jobs
mul8.002	105813	74881	208900	Multiprocessing 8 jobs: Unknown jobs
mul8.003	141126	88851	199455	Multiprocessing 8 jobs: Unknown jobs
null.000†	73217	15660	139615	ULTRIX Null job (daemons running)
savec.000	130288	85373	215867	ULTRIX C compiler
ue02.000	98385	59452	199973	UETP (User Environment Test Program, a VMS diagnostic), 2 tasks
ue10.000	98494	61476	212150	UETP, 10 tasks
ue20.000	100670	62188	201582	UETP, 20 tasks

Table 1 Traces used in the simulations. A dagger (†) indicates the trace was not included in the ''all'' simulation.

all-associative cache, the cache size is merely reduced by the block size, resulting in a nominal impact on the miss ratio.

As the cache size increases with a given associativity, the mean relative change in miss ratio decreases for all associativities simulated. This is caused by the fact that as the cache size increases while holding the

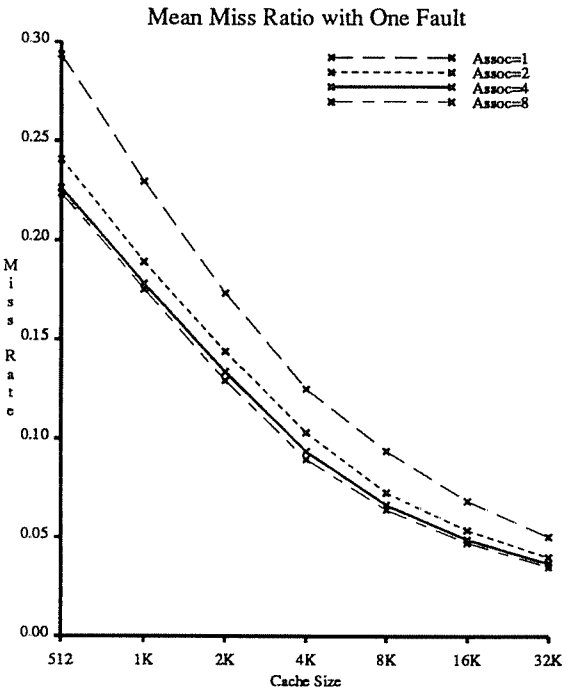

Figure 1a Mean miss ratio of the "all" simulation with one fault for various associativities

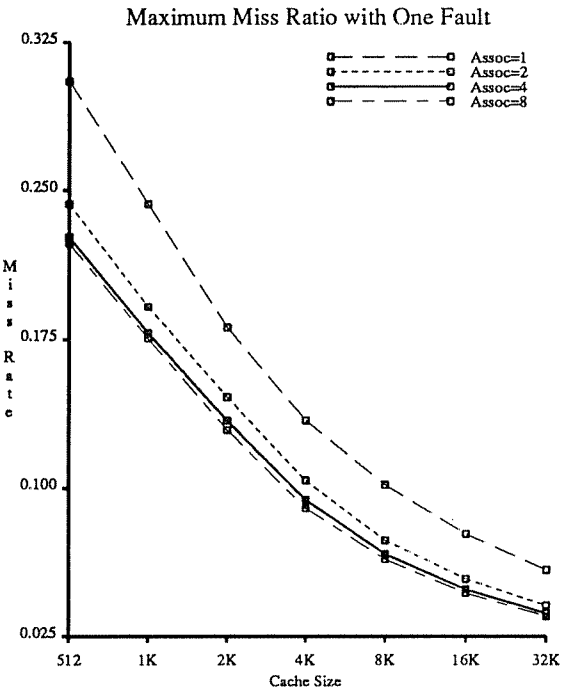

Figure 1b Maximum miss ratio of the "all" simulation with one fault for various associativities

These figures show the maximum and mean miss ratios for simulating all the ATUM traces. Caches were flushed between each of the different traces. The replacement policy is LRU; the block size is 16 bytes.

block size and associativity constant, the number of sets in the cache increases; from equation (3) we glean that the expected relative change in miss ratio is proportional to $\frac{1}{s}$. The fraction $\frac{m(n-1)}{m(n)}$ also tends to decrease with increasing cache size, since the "derivative" of the cache miss ratio tends to monotonically decrease with cache size (i.e., increasing cache size provides an increasingly smaller improvement in miss ratio). Thus we can be confident that this mean relative change behavior will be observed over a wide variety of programs.

Figure 2b illustrates that, with an associativity of 1, the maximum relative change in miss ratio increases with cache size after a dip with a 2K-byte cache, while with other associativities the maximum relative change slowly decreases over the whole range of cache sizes simulated. The odd behavior with $n = 1$ is explained by the observation that when a particular cache block is referenced exceedingly often, increasing the cache size does not affect how often it is referenced. Increased cache size only reduces contention for a particular set, not the number of accesses to a particular block. Thus $\underset{i}{MAX}\left[D_i(1)\right]$ may not decrease very much as n increases.

Table 2 presents data to aid in understanding this better. In it, the number of hits[6] to the set accessed least often $\left[\underset{i}{MIN}\left[D_i(1)\right]\right]$, the set accessed most often $\left[\underset{i}{MAX}\left[D_i(1)\right]\right]$, the mean number of hits to a set $\left[\frac{D(1)}{s}\right]$, and the number of misses in a cache of the specified size with no faults ($M(1)$) are shown. Recall that the

[6] A hit is defined as an access to the cache that is not a miss.

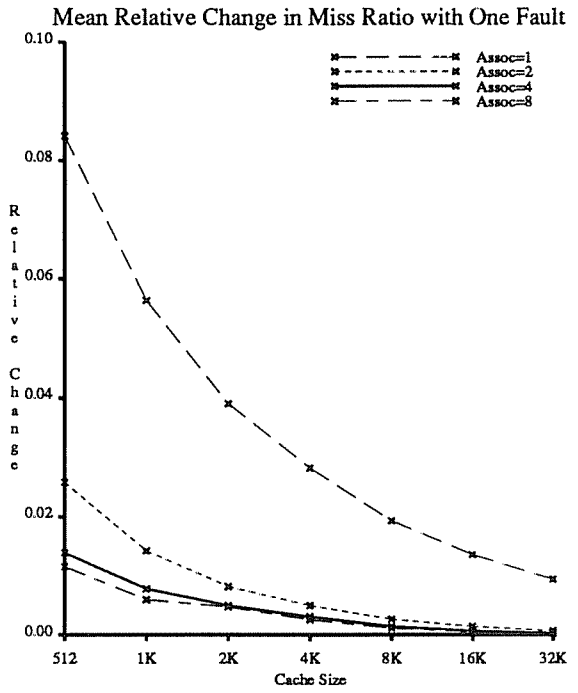

Figure 2a Mean relative change in miss ratio of the "all" simulation with one fault for various associativities

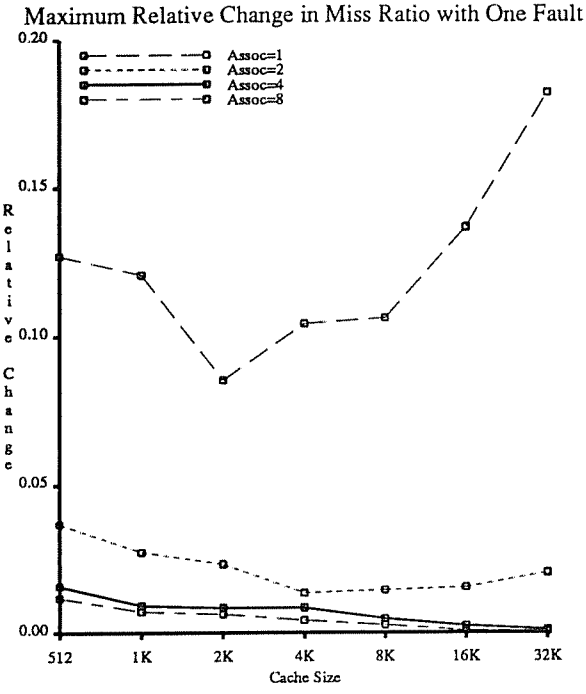

Figure 2b Maximum relative change in miss ratio of the "all" simulation with one fault for various associativities

These figures show the mean and maximum relative change in the miss ratio of simulating all the ATUM traces with a block size of 16 bytes and one fault *vs* no fault. As the cache size increases, the mean relative change in the miss ratio decreases. However, the maximum relative change in miss ratio vacillates with smaller associativities, with an apparent minimum, and steadily decreases with larger associativities.

mean relative change in miss ratio, $\delta(n, f_1)$, is $\frac{1}{s}\frac{D(1)}{M(1)}$, and the maximum relative change is $\frac{\underset{i}{MAX}\left[D_i(1)\right]}{M(1)}$. It is apparent from the table that in the transition from 32 to 64 and from 64 to 128 sets the maximum relative change decreases because $\underset{i}{MAX}\left[D_i(1)\right]$ decreases proportionately faster than $M(1)$. However, with larger cache sizes, the reduction in $\underset{i}{MAX}\left[D_i(1)\right]$ bottoms out; at the extreme in an infinitely-sized cache, the maximum number of references to a set will be equal to the maximum number of references to a particular memory block. In smaller caches, the maximum number of references to a particular set consists of references to numerous memory blocks mapping to the same set; as the cache size increases, the maximum becomes increasingly dominated by the references to a particular memory block.

This is further demonstrated by considering the final column in Table 2, labelled "Next Block", which indicates the number of hits at depth 2 to the block that receives the most hits at depth 1[7]. The infrequent hits to this depth with larger caches indicates that references are already being made almost exclusively to a single memory block that maps to the set in question; several memory blocks competing for block $D_k(1)$ would lead to

[7] I.e., let $D_k(1) \equiv \underset{i}{MAX}\left[D_i(1)\right]$; then "Next Block" refers to the number of hits at $D_k(2)$.

Sets	M(1)	Max	Mean	Min	Next Block
4	3804106	1089301	1031731	955855	274075
8	3164487	686390	595818	528229	125116
16	2627118	414123	331494	264980	68197
32	2148000	272690	180719	139381	40233
64	1720476	207438	97039	50462	8752
128	1322866	112895	51626	23559	3036
256	971398	100479	27186	8003	3445
512	732006	77351	14060	3335	213
1024	535274	73039	7222	844	97
2048	394121	71676	3680	226	12

Table 2 Block access statistics for the "all" simulation, $n = 1$

Table 2 presents the number of block references for the "all" simulation with an associativity of 1 and various cache sizes (number of sets). Included are the number of hits to the block referenced most (**max**) and least (**min**) often, the **total** number of hits at depth 1 ($D(1)$), and the **mean** number of hits per set $\left\lfloor \frac{D(1)}{s} \right\rfloor$. In addition, the table shows the number of hits to the **next block** for the set referenced most often. I.e., let k be the set such that $D_k(1)$ has the maximum number of accesses in the cache as shown in the table. Then "next block" refers to $D_k(2)$.

a large $D_k(2)$, as can be seen in the table for smaller caches. The upshot of this discussion is that the *maximum* relative change in miss ratio will inevitably deteriorate with larger cache sizes in a direct-mapped cache.

However, with increasing cache size, the maximum and mean miss ratios do continue to improve (see Figure 1). Therefore, the maximum relative change (Equation (1)) increases with increasing cache size for a direct mapped cache.

4.2. Changing Block Size

Figure 3 depicts the effect of changing the block size on the relative change in miss ratio for the "all" simulation. All the graphs in Figure 3 include block sizes of 8 bytes, 16 bytes, and 32 bytes. Figure 3a indicates the mean relative change in miss ratio for associativities of 1 and 2 and Figure 3b does so for associativities of 4 and 8. Qualitatively, the results do not digress from the observations made in the previous section; the lines for a block size of 16 are unchanged from Figure 2. The most notable aspect of these graphs is expected: the larger the block size, the greater the relative change in miss ratio. Invalidating a 32-byte block is for all practical purposes equivalent to invalidating two "adjacent"[8] 16-byte blocks. Thus on the average it seems reasonable to expect an approximate doubling in the additional misses caused by invalidating a B-byte block *vis-á-vis* a $\frac{B}{2}$-byte block; some discrepancy is caused by the fact that the miss ratios for equal sized B-byte

[8] Adjacent in the sense that the mapping function from the real address to the appropriate set in the 16-byte block cache sees a difference only in bit 5 of the real address.

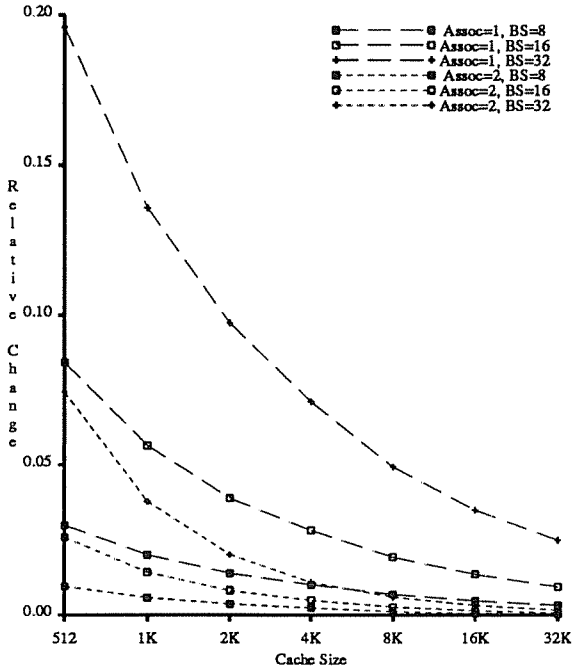

Figure 3a Mean relative change in miss ratio of the "all" simulation with one fault for associativities of 1 and 2 and various block sizes

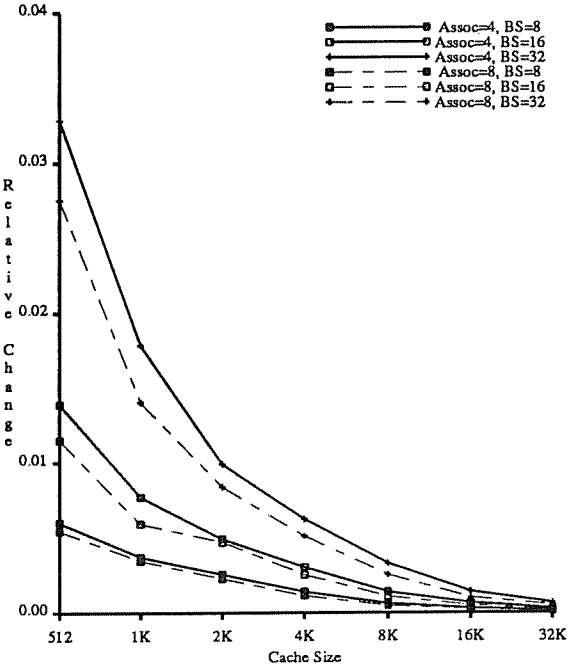

Figure 3b Mean relative change in miss ratio of the "all" simulation with one fault for associativities of 4 and 8 and various block sizes

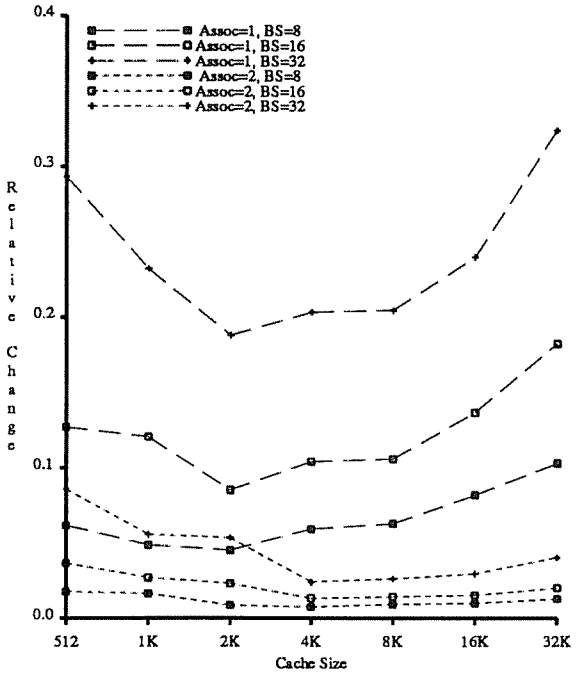

Figure 3c Maximum relative change in miss ratio of the "all" simulation with one fault for associativities of 1 and 2 and various block sizes

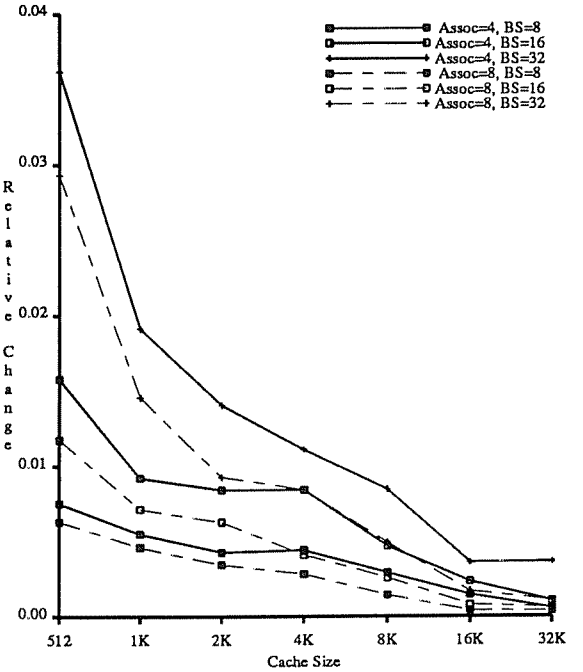

Figure 3d Maximum relative change in miss ratio of the "all" simulation with one fault for associativities of 4 and 8 and various block sizes

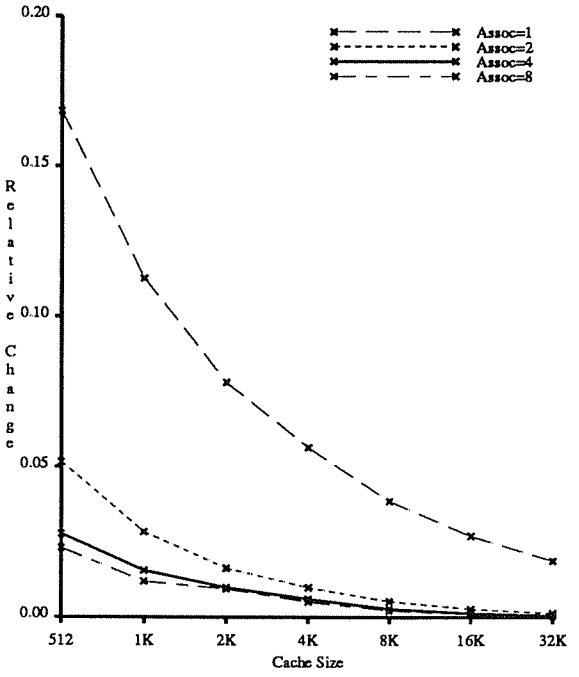

Figure 4a Mean relative change in miss ratio of the "all" simulation with two faults for various associativities; a set may have at most one fault

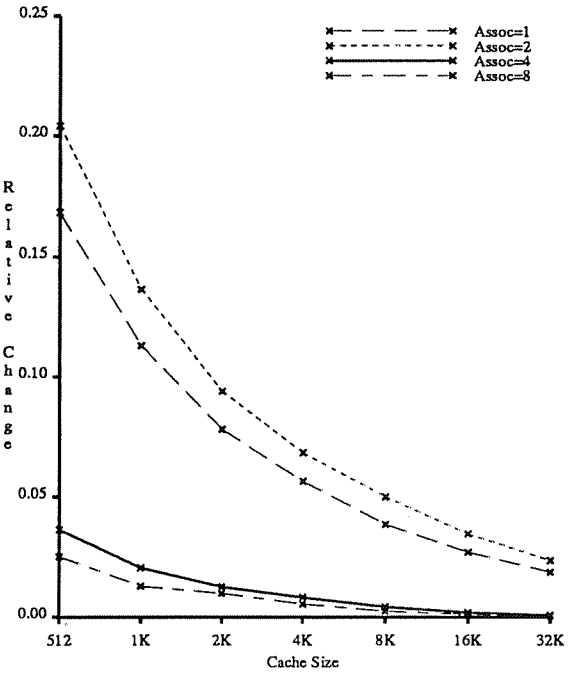

Figure 4b Mean relative change in miss ratio of the "all" simulation with two faults for various associativities; a set has two faults

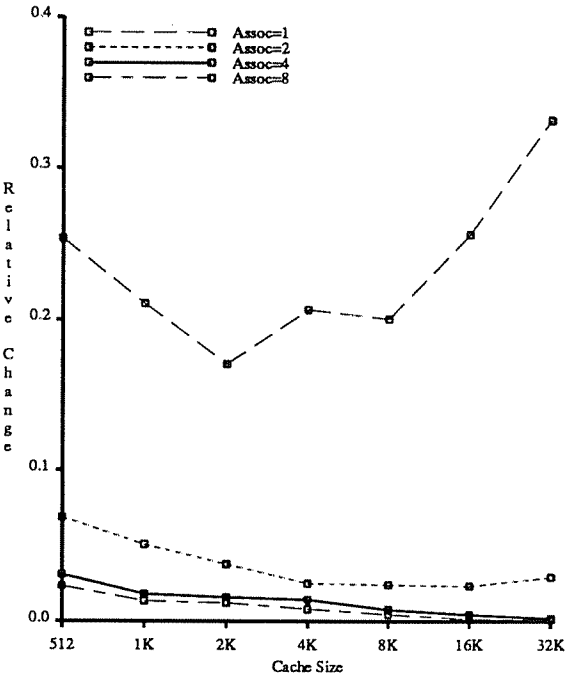

Figure 4c Maximum relative change in miss ratio of the "all" simulation with two faults for various associativities; a set may have at most one fault

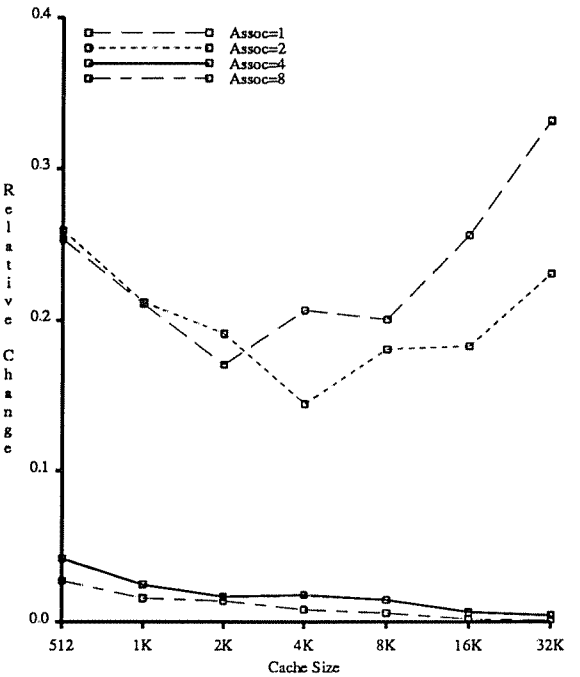

Figure 4d Maximum relative change in miss ratio of the "all" simulation with two faults for various associativities; a set has two faults

block and $\frac{B}{2}$-byte block caches will differ.

Figure 3c shows the maximum relative change in miss ratio for $n = 1$ and $n = 2$, while Figure 3d shows the same for $n = 4$ and $n = 8$. Note that the scale of the two figures differs by a factor of 10. Again, the lines for BS = 16 are the same as in Figure 2. No qualitative differences are found here, either. As with the mean, the maximum relative change in miss ratio is approximately proportional to the block size.

4.3. Two Faults

Figure 4 shows the maximum and mean relative change in miss ratios for the "all" simulation and fault vectors $\mathbf{f}_{2,1}$ and $\mathbf{f}_2$. Block size is 16 bytes to allow for comparison with Figure 2. All graphs show results for associativities of 1, 2, 4, and 8.

Figure 4a displays the mean relative change in miss ratio with the two faults restricted to be in different sets. Equation (6) indicates that the mean relative change of $\mathbf{f}_{2,2}$ should not be much different from $\mathbf{f}_{2,1}$, since the additional misses generated by having two faults in the same set are amortized by the preponderance of distributions with one fault in each of two different sets. Therefore the fault vector $\mathbf{f}_2$, in which both faults occur in the same set, is shown in Figure 4b, highlighting the effect.

The behavior of the $\mathbf{f}_{2,1}$ cache is qualitatively as with a single fault: the mean relative change is monotonically decreasing for all associativities, and the maximum relative change is increasing after a slight dip for the direct mapped cache and slowly decreasing for associative caches. The greatest difference between Figures 4a and 4c and Figures 2a and 2b is the scale. Equation (4) states the mean for the fault vector $\mathbf{f}_{2,1}$ will be twice that for $\mathbf{f}_1$; that assertion has been verified.

Figure 4c demonstrates the maximum relative change in miss ratio with the two faults restricted to be in different sets. The maximum relative change in miss ratio in Figure 4c is always under twice the corresponding values in Figure 2b. Basically, the maximum in Figure 4c is the sum of the number of references to the two largest $D_i(n)$'s. Thus the maximum relative change for two faults in separate sets is less than or equal to twice the maximum relative change for a single fault.

The maximum relative change in Figure 4d for a 2-way associative cache is almost the same as for the direct mapped cache. In both cases, an equal number of bytes are invalidated. The maximum for the 2-way associative cache (usually) occurs when the two faults are in the same set. This has the net effect of eliminating an entire set from the cache. Although there is a similarity between this and the single fault for a direct mapped cache case, there are many differences. For one, in the double-fault, 2-way associative cache, twice as many bytes are invalidated. Yet the greatest discrepancy arises from the fact that, given the same cache size, the direct mapped cache has a significantly higher miss ratio, particularly for small cache sizes. Thus, an equal absolute increase in the number of misses – which occurs when both caches have an entire set invalidated – results in a larger relative change for the 2-way associative cache. In general, as the associativity of a cache increases while maintaining a constant cache size, so will the relative change in miss ratio caused by eliminating an entire set.

4.4. Large Numbers of Faults

Figures 5 demonstrate the impact of a large number of faults on the maximum and mean miss ratios for the "all" simulation for a cache with 16-byte blocks and 64 sets. Associativity is varied from one to eight. In Figure 5a, a given set is constrained to have at most one fault in it; results are presented for direct mapped and

2-way and 4-way associative caches. In Figure 5b, each set is allowed to have at most two faults in it; results are shown for 2-way, 4-way, and 8-way associative caches. Figure 5c shows results for at most 4 errors per set; the associativities shown are 4 and 8. Lastly, Figure 5d shows the resulting miss ratios for an 8-way associative cache with both a maximum of 6 and a maximum of 8 faults in a given set. For all the graphs, all possible distributions of faults were considered; in some simulations, as many as 10^{419} distributions are possible. Of course, with as many faults as were simulated in these cases, the probability is nearly 0 that all of the faults would occur in non-critical resources, such as tags and data bits.

The maximum and mean miss ratios are equal at the endpoints since the endpoints correspond to the cases where either none or all possible blocks – based on the maximum number of faulty blocks in a given set – are faulty. Also note that the mean and maximum miss ratios in Figure 5a for a 2-way associative cache when 64 of the blocks are faulty is equal to the miss ratios for a direct mapped cache with no faults and the same number of sets; essentially, the 2-way associative cache has been rendered direct mapped. A similar phenomenon occurs in Figure 5b between the 2-way associative and 4-way associative caches.

In Figure 5a, the mean miss ratio is a straight line with slope $\frac{1}{s}$, as described by Equation (6). The slopes for the other graphs are more complicated but are described by polynomials whose degree is bounded by $MIN(n, maxFPS)$, where *maxFPS* is the Maximum number of Faults per Set allowed.

As can be seen from the graphs, even with a large number of faults, the miss ratio is not exceedingly large as long as the maximum number of faults per set remains below the associativity, i.e., as long as an entire set is not allowed to be faulty. For example, for the 4-way associative cache in Figure 5a, even 64 faults only cause a 52% mean and maximum *relative* change in miss ratio – from 0.1427 with no faults to 0.2169 with 64 faults – as long as only one fault occurs in each set. The maximum usually occurs when all the faults are conglomerated in a minimum number of sets since $D_k(j)$ tends to decrease as j increases and k is constant. The mean is dominated by a more even distribution, which implies fewer faults in a particular set when the number of faulty blocks is small compared to the total number of blocks. The actual distribution depends on the probability distributions alluded to by Equation (7). The fewer the number of faults, the higher $P[Y=0]$ and the lower $P[Y=h]$, where $h = MIN(n, maxFPS)$.

4.5. Well-behaved Programs

The relative effect of faults on miss ratio are displayed in Figure 6 for a trace (forf.000) representative of well-behaved traces. By well-behaved traces we mean that the number of accesses to each set in the cache is relatively evenly distributed. Figure 6a is virtually identical to Figures 5a and 5b.

The maximum relative change in miss ratio shown in Figure 6b is greater than those in shown in Figures 3c and 3d, particularly for large caches. This difference can be attributed to the lower miss ratio of the forf.000 trace and its larger referencing bias for large cache sizes. For example, with a block size of 16 bytes the miss ratio for the direct mapped, 512-byte cache of the forf.000 trace is 0.2618 and for a 32K-byte cache is 0.0375. With the same cache parameters, the "all" simulation has miss ratios of 0.2708 and 0.0497 for a 512-byte and 32K-byte cache, respectively. The number of references to the most heavily referenced set $\left\lceil \underset{i}{MAX} \left\lceil D_i(1) \right\rceil \right\rceil$ in a direct mapped cache with a block size of 16 bytes, the total number of references to the cache that hit ($D(1)$), and the percentage of all references to the cache that these comprise $\left\lceil \frac{\underset{i}{MAX} \left\lceil D_i(1) \right\rceil}{D(1)} \right\rceil$ are shown in Table 3 for

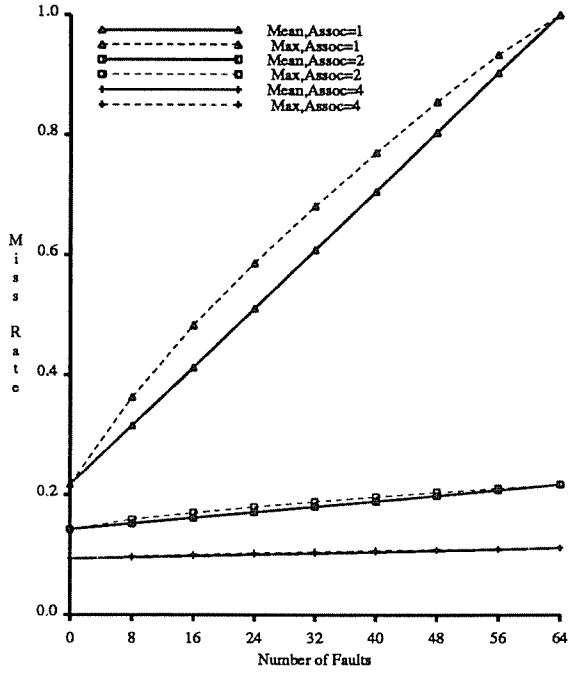

Figure 5a Mean and maximum miss ratios with numerous faults for the "all" trace with at most one fault per set

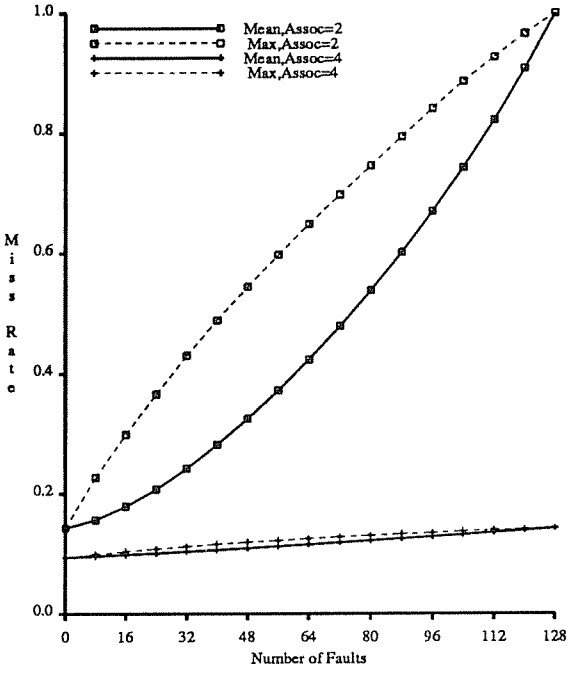

Figure 5b Mean and maximum miss ratios with numerous faults for the "all" trace with at most two faults per set

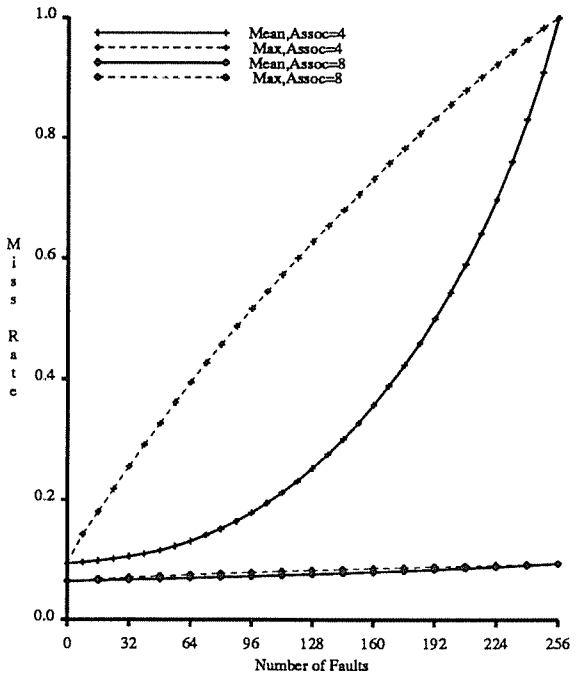

Figure 5c Mean and maximum miss ratios with numerous faults for the "all" trace with at most four faults per set

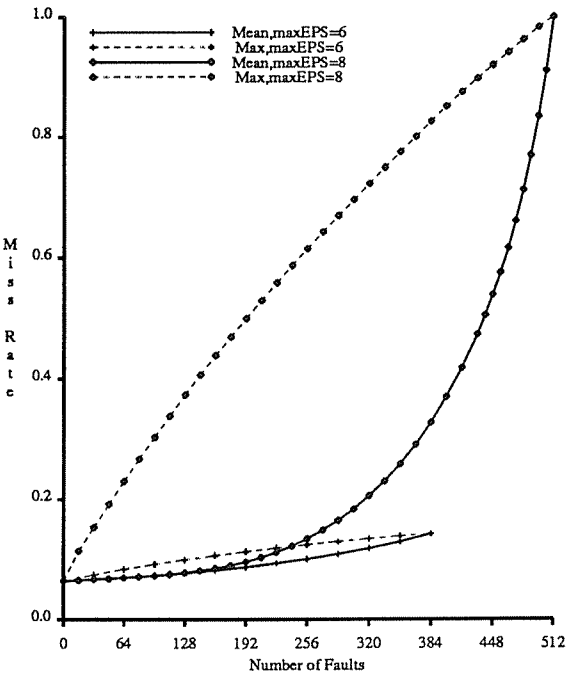

Figure 5d Mean and maximum miss ratios with numerous faults for the "all" trace with at most six and eight faults per set; associativity is 8

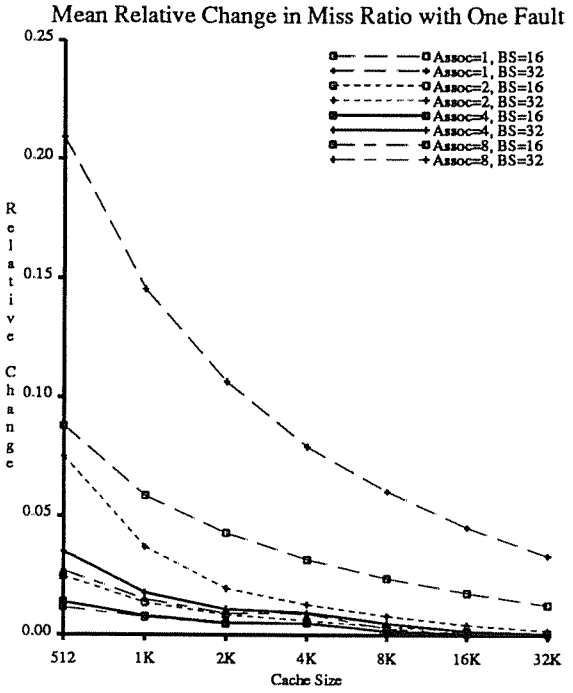

Figure 6a Mean relative change in the miss ratio with one fault for the "forf.000" trace for various associativities and block sizes

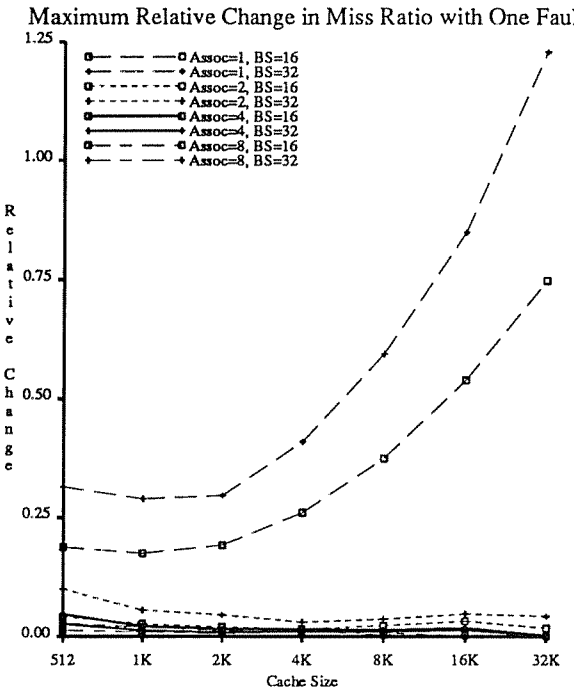

Figure 6b Maximum relative change in the miss ratio with one fault for the "forf.000" trace for various associativities and block sizes

Figures 6a and 6b show the impact of a fault in the miss ratios of the "forf.000" trace, which is representative of the majority of traces that were simulated. Doubling the block size approximately doubles the mean relative change in miss ratio; increasing the cache size decreases the mean relative change in miss ratio. However, the maximum relative change in miss ratio increases with low associativities and increasing cache size. Block size is 16 bytes.

the "all", forf.000, and null.000 simulations for various set sizes. The table demonstrates why the maximum relative change for forf.000 is greater than for "all". For a particular trace, the maximum number of references to a block scales worse than when traces are combined: different traces tend to favor different sets in the cache. To assess the impact of increasing the associativity to 2, the next block information is also included (see section 4.1) in the table.

4.6. Poorly-behaved Programs

The relative effect of faults on the miss ratio of poorly-behaved programs is exemplified by Figure 7, which shows the results of simulating null.000. A poorly-behaved program is one in which the reference pattern is highly skewed toward particular blocks. The fact that null.000 fits this description can be verified with Table 3. Even with 2048 sets, over 41% of the program's references are to a single 16-byte block! This percentage only decreases to 31% for a block size of 8. When instruction fetches are simulated independent of data accesses, 51% of all references that hit in a direct mapped, 8-byte block, 32K-byte cache are to the same block. When the block size is increased to 16 bytes, an adjacent block which is also referenced very often joins the most-referenced block. Then over 67% of all instruction references that hit in a 32K-byte cache are to the same block.

Sets	all				forf.000				null.000			
	Max	Total	Max %	Next	Max	Total	Max %	Next	Max	Total	Max %	Next
4	1089301	4126927	26.39	274075	63257	206608	30.62	12708	102418	189611	54.01	2511
8	686390	4766546	14.40	125116	38635	241282	16.01	4585	97070	198163	48.98	572
16	414123	5303915	7.80	68197	24253	269771	8.99	2968	95485	202701	47.11	151
32	272690	5783033	4.71	40233	19634	296133	6.63	914	93951	208156	45.13	170
64	207438	6210557	3.34	8752	14728	316634	4.65	614	93210	211141	44.15	14
128	112895	6608167	1.70	3036	11836	339157	3.49	207	93190	219795	42.40	16
256	100479	6959635	1.44	3445	11472	357041	3.21	158	93143	221302	42.09	4
512	77351	7199027	1.07	213	11429	370704	3.08	137	93112	222508	41.85	7
1024	73039	7395759	0.98	97	11279	380234	2.97	25	93107	224055	41.56	6
2048	71676	7536912	0.95	12	11248	386143	2.91	19	93068	224744	41.41	7

Table 3 Comparison of the various traces simulated

Table 3 compares the values for $\underset{i}{MAX}\left[D_i(1)\right]$, $D(1)$, and $\dfrac{\underset{i}{MAX}\left[D_i(1)\right]}{D(1)}$ for the three traces simulated in this study. The caches for the particular simulations presented had a block size of 16 bytes. The large number of references to a single block by the null.000 trace makes it particularly sensitive to a fault in a direct mapped cache. To provide some insight into the consequences of changing the associativity to 2, the next block information is also included (as defined in section 4.1).

Analyses of the program trace reveals that the large number of accesses to a single block are due to a tight loop which is used to test two memory locations. The entire loop is contained within a 16-byte block. One of the two longwords referenced in each iteration of the loop is to a word which coincidentally maps to the same block as the loop itself[9]. Thus, for an 8-byte block, 32K-byte direct mapped cache, 27% of the data references are to the same block. There is an equal number of references to the immediately adjacent block; however, it would require a 64-byte block size to have the data items map to the same block.

There are several interesting facets about Figure 7. The first concerns the magnitude of the maximum relative change in miss ratio for the direct mapped caches, which approaches 3500% for a 32K-byte cache. This is due, as explained above, to the large number of references to the same block. With an associativity of 4, the maximum relative change occurs with a block size of 32 bytes and a cache size of 2K-bytes, and is less than 7%.

The most striking feature of Figure 7a is the bizarre behavior of the mean relative change in miss ratio for a block size of 32 bytes in a direct mapped cache. When the number of sets increases from 32 to 64 (cache size changes from 1K to 2K), the miss ratio drops from 0.3111 to 0.0484, because two different blocks – one for data, one for instructions – that contend for the same set with 32 sets resolve the contention with 64 sets. Since the mean number of additional misses due to the fault $\left\lceil \dfrac{D(1)}{s} \right\rceil$ is only approximately cut in half -- changing from 5077 with 32 sets to 3451 with 64 sets -- the relative change increases substantially.

[9] These references are not to set 0, as may seem intuitive.

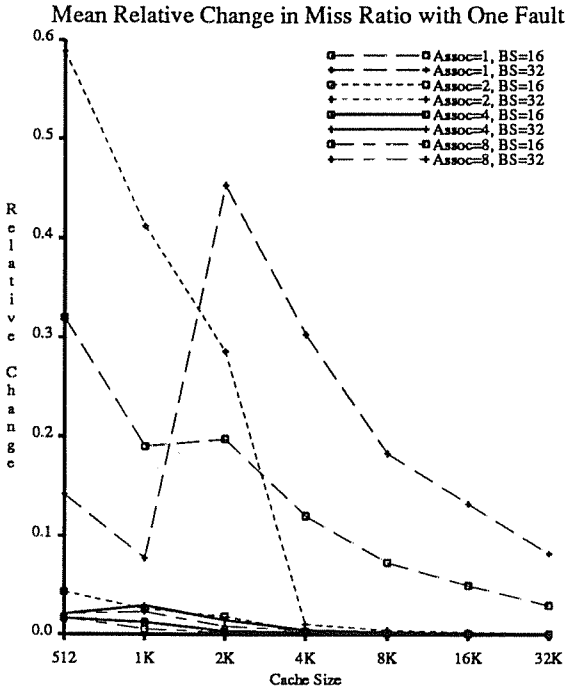

Figure 7a Mean relative change in the miss ratio with one fault for the "null.000" trace for various associativities and block sizes

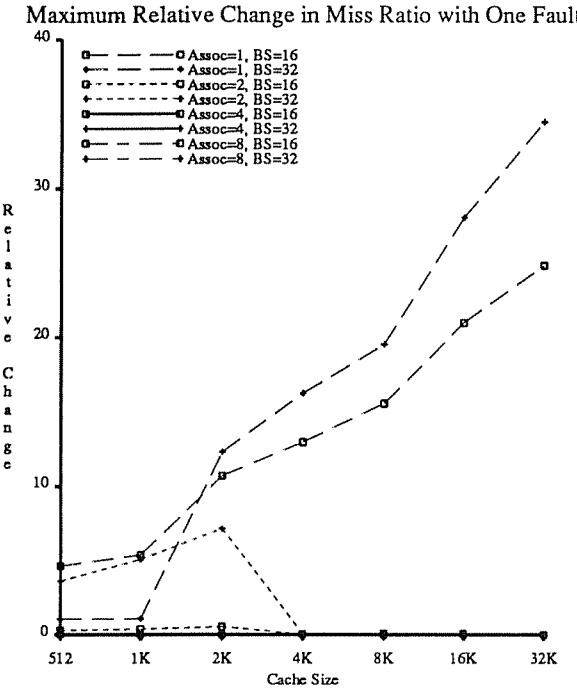

Figure 7b Maximum relative change in the miss ratio with one fault for the "null.000" trace for various associativities and block sizes

Figures 7a and 7b show the impact of a fault in the miss ratios of the "null.000" trace, which is representative of the minority of traces that were simulated. The erratic behavior is caused by a particular block that is referenced frequently. Worst-case behavior causes a forty-fold increase in the miss ratio. Block size is 16 bytes.

The other striking feature is the rapid decrease in the mean relative change for a 2-way associative cache with 32-byte blocks. In this case, the miss ratio only varies by a factor of 4.5 between cache sizes of 512 bytes and 4K bytes. Equation (3) indicates that the mean relative change is $\frac{D(2)}{sM(2)}$. Part of the decrease may be attributed to the doubling of s, particularly for the 512-byte – 2K-byte improvements. The large downward jump at the 4K-byte cache is caused by the dive in $D(2)$ from 59508 with 32 sets to 3058 with 64 sets[10]. This jump is again caused by the resolution of the contention between an often-referenced data and often-referenced instruction block.

Since the poor performance of this trace can be attributed to its spinning for the arrival of data, it appears that the longer delays associated with cache misses in the spinning loop are acceptable. However, if no second-level cache exists to supply the data, a tremendous amount of memory bandwidth will be unproductively consumed.

[10] Recall that for a 2-way associative cache, $M(2) = R - D(1) - D(2)$. With one fault, $\Delta D(2) = -\Delta M(2)$, where $\Delta D(2) < 0$ is the increase in the number of hits at depth 2 due to a fault and $\Delta M(2)$ is the increase in the number of misses in a cache of associativity 2 due to a fault. Therefore, the mean relative change with one fault $\frac{D(2)}{sM(2)} \propto \frac{D(2)+\Delta D(2)}{K-\Delta D(2)}$, where the constant $K = M(2) - D(1) - D(2)$.

5. Conclusions

We have attempted to provide insight into the affect on cache miss ratio of tolerating non-critical faults in an on-chip microprocessor cache. Since a cache is a non-critical resource – it is primarily used to increase the performance of, not ensure the correct operation of, a processor – it becomes a reasonable alternative to use a microprocessor chip that contains a processing fault in one of the data blocks or tag bits. The performance of this microprocessor is reduced by some factor; for medium to large caches, this factor may still result in a faster memory system than results from a cache that uses redundant memory.

We first how the miss ratio a cache with any fault pattern can be calculated from the number of references to the j-th block in the i-th of s sets. We then show how to extend all-associativity simulation to calculate these metrics for many cache configurations with a single pass through an address trace. Lastly, we applied this technique to the ATUM traces.

Results indicate that the *mean* relative degradation in miss ratio from a few faults decreases with increasing cache size, and is usually small (< 5% per fault). Furthermore, if no set is completely disabled, mean degradation for large caches is negligible. Consequently, it is likely that the effective access time of a cache with some blocks marked faulty will be less than that of a cache performing error detection and correction on all cache references.

The *maximum* relative change in miss ratio for a single cache fault – or for two cache faults in distinct sets – is acceptable (< 5%) if the associativity of the cache is 4 or greater and the block size is 8 or 16 bytes. Larger block sizes suffer greater penalties with permanent block invalidations. With a direct-mapped cache, however, there is a probability (albeit small with a large number of sets) that the executing program heavily references the faulty block(s), severely degrading the cache's performance. We expect that the overall impact of this worst-case behavior will not be significant for machines used to run many different programs.

6. Acknowledgments

We are indebted to the Condor project at the University of Wisconsin [9] for providing us with the computational resources required for the large number of lengthy simulations performed and G. Sohi for reading and improving drafts of this paper.

References

1. Agarwal, A., Sites, R., and M. Horowitz, "ATUM: A New Technique for Capturing Address Traces Using Microcode," *Proc. the 13th Annual Symposium on Computer Architecture*, pp. 119 - 129, Tokyo, Japan, June 1986.
2. Agarwal, A., Horowitz, M., and J. Hennessy, "An Analytical Cache Model," *ACM Trans. on Computer Systems*, vol. 7, no. 2, pp. 184 - 215, May 1989.
3. Berenbaum, A. D., Colbry, B. W., D. R. Ditzel, R. D. Freeman, H. R. McLellan, K. J. O'Connor, and M. Shoji, "CRISP: A Pipelined 32-bit Microprocessor with 13-kbit of Cache Memory," *IEEE Journal of Solid-State Circuits*, vol. SC-22, no. 5, pp. 776 - 782, October 1987.
4. Goodman, J. R., "Using Cache Memory to Reduce Processor-Memory Traffic," *Proc. Tenth International Symposium on Computer Architecture*, pp. 124 - 131, Stockholm, Sweden, June 1983.
5. Hill, M. D. and Smith, A. J., "Evaluating Associativity in CPU Caches," *IEEE Trans. on Computers*, vol. C-38, no. 12, pp. 1612 - 1630, December 1989.

6. Horowitz, M., Chow, P., D. Stark, R. T. Simoni, A. Salz, S. Przybylski, J. Hennessy, G. Gulak, A. Agarwal, and J. M. Acken, "MIPS-X: A 20-MIPS Peak, 32-bit Microprocessor with On-Chip Cache," *IEEE Journal of Solid-State Circuits*, vol. SC-22, no. 5, pp. 790 - 799, October 1987.

7. Jouppi, Norman P., "Improving Direct-Mapped Cache Performance by the Addition of a Small Fully-Associative Cache and Prefetch Buffers," *Proc. 17th Annual Symposium on Computer Architecture, Computer Architecture N ews*, vol. 18, no. 2, pp. 364-373, ACM, June 1990.

8. Kadota, H., Miyake, J., I. Okabayashi, T. Maeda, T. Okamoto, M. Nakajima, and K. Kagawa, "A 32-bit CMOS Microprocessor with On-Chip Cache and TLB," *IEEE Journal of Solid-State Circuits*, vol. SC-22, no. 5, pp. 800 - 807, October 1987.

9. Litzkow, M., Livny, M., and M. W. Mutka, "Condor — A Hunter of Idle Workstations," *Proceedings of the 8th International Conference os Distributed Computing Systems*, San Jose, California, June 1988.

10. Mattson, R. L., Gecsei, J., D. R. Schultz, and I. L. Traiger, "Evaluation Techniques for Storage Heirarchies," *IBM Systems Journal*, vol. 9, no. 2, pp. 78 - 117, 1970.

11. Patterson, D. A., Garrison, P., M. Hill, D. Lioupis, C. Nyberg, T. Sippel, and K. Van Dyke, "Architecture for a VLSI Instruction Cache for a RISC," *The Tenth Annual Symposium on Computer Architecture*, vol. 11, no. 3, pp. 108 - 116, Stockholm, Sweden, June 13-17, 1983.

12. Phillips, D., "The Z80000 Microprocessor," *IEEE Micro*, pp. 23 - 36, December 1985.

13. Smith, A. J., "Cache Memories," *Computing Surveys*, vol. 14, no. 3, pp. 473 - 530, September 1982.

14. Sohi, G., "Cache Memory Organization to Enhance the Yield of High-Performance VLSI Processors," *IEEE Transactions on Computers*, vol. 38, no. 4, pp. 484 - 492, April 1989.