Minimal Subset Evaluation: Rapid Warm-up for Simulated Hardware State

John W. Haskins, Jr. Kevin Skadron
Department of Computer Science
University of Virginia
Charlottesville, VA 22904
{predator,skadron}@cs.virginia.edu

Abstract

This paper introduces minimal subset evaluation *(MSE) as a way to reduce time spent on large-structure warm-up during the fast-forwarding portion of processor simulations. Warm up is commonly used prior to full-detail simulation to avoid cold-start bias in large structures like caches and branch predictors. Unfortunately, warm up can be very time consuming, often representing 50% or more of total simulation time. Previous techniques have used the entire fast-forward interval to obtain accurate warm up, which may be prohibitive for large parameter-space searches, or chosen a short but ad-hoc warm-up length that reduces simulation time but may sacrifice accuracy.*

MSE probabilistically determines a minimally sufficient fraction of the set of fast-forward transactions that must be executed for warm up to accurately produce state as it would have appeared had the entire fast-forward interval been used for warm up. The paper describes the mathematical underpinnings of MSE and demonstrates its effectiveness for both single-large-sample and multiple-sample simulation styles. In our experiments, MSE yields errors of less than 1% in IPC measurements with cycle-accurate simulation, while reducing simulation times by an average factor of two or more.

1 Introduction

This paper introduces a new technique for minimizing the amount of simulation required to place large processor structures like caches and branch predictors in an accurate state before full-detail simulation. Research in computer architecture almost always requires simulation, because simulated processor models are easier, faster, and cheaper to develop than hardware prototypes and vastly more flexible.

Detailed processor studies require *cycle-accurate* simulation that can model the step-by-step flow of instructions through today's complex processor pipelines. Unfortunately, these simulations are terribly slow, with slowdown factors of hundreds or thousands compared to native execution. With cycle-level simulations, running many of the SPEC95 benchmarks to completion with reference inputs takes days or weeks [8], and running some of the SPEC 2000 benchmarks takes over a year [5].

To combat these long simulation times, most simulation strategies either take data from multiple short samples throughout the program, "fast-forwarding" between samples [2], or else fast-forward to a single, large simulation window of 50–100 million instructions [8]. Both save time when advancing to a zone of full-detail simulation by performing minimal work between samples. A third approach to reduce simulation times is to use reduced inputs, but this raises the question of the reduced input's accuracy.

Unfortunately, the accuracy of simulation within these full-detail samples depends on avoiding *cold-start bias*, which requires each that each sample start with accurate cache and branch predictor state [2]. This typically means that fast-forwarding must not only update the program's architectural state, but must also perform cache and branch-prediction simulation for the entire fast-forward interval if large-structure state is to be completely accurate or "warmed up" at the beginning of a sample. Unfortunately, despite the time savings realized by not modeling pipeline state, cache and branch prediction simulation is still slow, and this makes state warm-up during fast-forwarding prohibitive, especially if it must be repeated for multiple simulations with varying parameters or to reach samples deep in a benchmark's execution. Checkpointing simulation state at the beginning of each sample would be one solution, but separate checkpoints would be required for each desired combination of cache and branch predictor configurations.

Current approaches [2, 4] use ad-hoc heuristics to reduce warm-up[1] length. MSE is a more formal mathematical approach that determines a minimal length of the warm-up period necessary to conform to a user-specified probability of accurate large-structure initialization. MSE is also flexible, being directly applicable to any desired hardware configuration or sampling regime. MSE therefore permits the

[1] By "warm up," we refer to the concluding portion of the fast-forward interval where transactions to large structures (*e.g.*, memory references' interactions with the cache) are simulated.

user to either tune the warm-up interval to specific cache and branch predictor configurations or to select a warm-up interval that provides at least the desired probability of accurate warm-up for every configuration of interest. MSE is very simple to implement and apply to any desired simulation system. We have developed software that facilitates its use in conjunction with the *SimpleScalar* [1] software suite.

This paper presents the MSE approach for both direct-mapped and set-associative structures, and presents results using direct-mapped caches for both a single-large-sample interval as in [8] and results using set-associative caches for multiple-sample intervals as in [2]. For maximum flexibility, the MSE approach also makes no assumptions about the starting state of the structures to be warmed up. This allows the approach to work with both the single-large-sample and the multiple-sample approaches.

The rest of this paper is organized as follows. We discuss related work in Section 2. Section 3 presents the formal MSE approach, and Section 4 presents our experimental methodology. Finally, we present our results in Section 5 and conclude in Section 6.

2 Related Work

Because simulating benchmarks to completion is prohibitive, several studies have explored ways to simulate only portions of the program's overall execution. Skadron *et al.* [8] used a sequence of heuristics to find a single, short but representative simulation window of 50 million instructions. The most important component of their approach is to exclude unrepresentative start-up behavior from early in the benchmark's execution; [8] goes on to present a table of fast-forward instruction counts for the SPECInt95 benchmarks.

Conte *et al.* [2] instead simulate multiple different, short samples from a benchmark's overall execution and use statistical techniques to identify the samples and ensure that together they capture representative behavior. Key to this technique is ensuring the accuracy of the state in large structures like the caches and branch predictor. Their work focuses on the branch prediction structures (assuming a perfect cache) and shows that using stale predictor state from the previous sample plus a short warm-up interval [7] of at least 7,000 instructions prior to the next sample is sufficient to minimize cold-start bias and achieve very small errors of a few percent in the mean observed IPC. We call this warm-up approach "short warm-up."

Lafage and Seznec [6] refine this sampling approach by using statistical classification methods to characterize the entire benchmark and provide a more rigorous guarantee of the chosen samples' representativeness. A potential problem with this approach is that finding configuration-independent metrics for representativeness is difficult. This work does not treat cold-start bias between samples.

Other heuristics for reducing cold-start bias are studied by Kessler *et al.* [4]. They consider using half of a sample's references for warm-up purposes; tracking only entries that are known to contain good state; using stale state from the previous sample; and flushing state but estimating how much error this introduces.

In short, several formal techniques exist for *sampling* execution, and these projects demonstrate the effectiveness of sampling in reducing simulation times while preserving accuracy. Yet all these techniques are dependent on accurate *warm-up* of large structures prior to each sample. While some heuristics for warm-up have been described, we are not aware of any efforts to develop a more formal approach to minimizing warm-up lengths while preserving accuracy.

Our MSE approach uses a combination of probabilistic analysis and profiling to develop a formal technique for identifying how long the warm-up interval must be to obtain a desired level of probability of accuracy in the large structures. To prove its effectiveness we make no assumptions about the prior state of the large structures. This is more conservative than the techniques that use stale state, but as we show, still produces short warm-up times. This also provides more flexibility in how MSE can be applied.

3 Reducing Warm-up Times

The MSE formulas determine the probability that warming-up only a t-instruction-long contiguous subset of fast-forward instructions prior to the beginning of a sample will accurately reproduce state. The MSE approach consists of the following steps:

1. The user first selects a desired probability of accuracy $p \in (0, 1)$. This value is then used to determine for a given cache configuration, the contiguous subset of instructions will with probability p reproduce the simulated hardware state exactly as if the entire set of fast-forward instructions had been used for warm-up.

2. The user profiles the benchmark to characterize, for any point in the benchmark, how many total instructions t must be seen in order to observe m unique references. This is a one-time cost for each benchmark–input pair; these profiles are valid for any p, configuration, or sample set of interest.

3. The user applies the MSE formulas to the desired hardware configuration and probability (p) to obtain the number m of unique references required in any warm-up interval.

4. The user then uses the profile to determine, for each simulation sample, how many total instructions t must be executed in order to observe the desired m unique references and hence achieve the desired probability p of accurate warm-up.

5. The simulation can then be run in an aggressive fast-forward mode consisting of only functional simulation in which just architected state is simulated. At t instructions prior to the beginning of the full-detail simulation sample, the fast-forward mode changes into warm-up mode, in which updates to large structures are now modeled. Then once the sample is reached, these large structures will with probability p contain accurate state. Alternatively, a second profiling pass can be run in which the benchmark's architected state is checkpointed for each sample (using SimpleScalar version 3.0 EIO traces, for example [1]). These checkpoints must occur t instructions before full-detail simulation is to begin. Simulating each sample then entails loading the checkpoint, warming up large structures for t instructions, and then beginning full-detail simulation. Note that the checkpoints are independent of processor configurations, so for a defined set of samples, these checkpoints are valid for any configuration.

These steps are described in further detail below.

3.1 MSE

To discuss MSE in the context of cache simulation, several more variables are necessary. Let N be the number of sets in the cache and a be its associativity. (For direct-mapped caches, $a = 1$.) The MSE formulas are then used to determine t for any p, a and N; that is, $t = MSE(N, a, p)$.

As mentioned, we take the conservative approach and make no use of any information about prior state. This means that our approach must find the warm-up interval t that touches all Na cache blocks. Since some benchmarks will not touch all entries in a large cache, even during a very long interval, MSE employs two more variables, α, $\beta \in (0, 1]$. These parameters tune N and a in the following way: $t = MSE(\alpha N, \beta a, p)$. Thus, if the user need only ensure that a fraction of sets and a fraction of blocks within each set are touched, $\alpha < 1$ and $\beta < 1$. For convenience, the experiments in this paper make the simplifying assumption that $\alpha = \beta = 1$.

Once we have selected our probability of accuracy p, we must select t, the number of instructions to execute during the warm-up interval. The t instructions of the warm-up should contain at least m unique memory references. By "unique" we mean that among these m memory references, no two access the same address. Conceivably t could be chosen precisely if a trace of the memory reference stream were available. Such traces, however, may rapidly become large and unwieldy. (Previous work [3] has addressed the cumbersome nature of traces and offer approaches that make their use more viable.)

Instead of dealing with full (or even compressed) traces, we obtain t using data gathered from the profiles. During the profiling run previously mentioned, a fully-associative cache is maintained for the stream of instruction memory references and for the stream of data memory references; in both cases, the cache block size is equal to the width of one memory word. Each time a memory reference occurs, the corresponding cache entry is logically "timestamped" with the executed instruction count. At the conclusion of the profiling run, the set of timestamps are sorted in descending order; the timestamp occurring m^{th} in the list is the number of instructions (t) prior to the full-detail sample that must be executed in order to encounter m unique memory references.

Armed with t which contains m unique memory reference addresses, we are ready to discuss the critically important implicit assumptions behind the MSE approach and apply the MSE formulas.

3.2 MSE Assumptions

The MSE formulas calculate the probability that at least αN sets of a cache will be touched at least βa times. Our formulas are based on the assumption that *unique memory references are typically distributed uniformly throughout the cache*. This assumption does not contradict the well-known, empirically demonstrated principle that "hot-spots"—regions of heightened activity due to locality—exist in caches. The critical difference is that the hot-spot principle considers the *entire* stream of memory references (L). Our assumption, by contrast, refers only to the subset of the memory reference stream that does *not* contain duplicates ($unique(L)$).

By filtering out duplicate memory reference addresses and focusing instead on only the subset of unique memory references, the hot-spot principle is irrelevant. Indeed, uniform distribution of unique memory reference addresses is exactly the ideal behavior for a cache because this would reduce the likelihood of conflicts. In an ideal direct-mapped cache, for instance, data in any set would have a $\frac{N-1}{N}$ chance of surviving a unique incoming reference address. Furthermore, the larger the N, the smaller the chance of conflict.

To verify the uniform distribution of $unique(L)$ throughout the cache analytically, we employed the χ^2 test. We developed software that takes, from the profile information, the number of references to each set of the cache (among the unique references only). From this, we first tally the total number of unique memory accesses, $|unique(L)|$. Then, we calculate a best estimate average number of hits per set

$$\bar{x} = \frac{|unique(L)|}{N}$$

Using $\bar{x}$ we group sets into bins such that the best estimate average number of hits per bin is at least 5 [11]. Finally, we use these data to compute $\tilde{\chi}^2_o$, the observed *reduced* χ^2.

In all cases, for all benchmarks used and for every N used, the raw profile data achieves $\tilde{\chi}_o^2 \leq 1.2$ and is almost always ≤ 0.5. The values of $\tilde{\chi}_o^2$ for each benchmark for all tested N from the SPECInt95 benchmarks are listed in Table 1 along with d, the number of degrees of freedom used in the reduced χ^2 computation. In most cases, a value of $\tilde{\chi}_o^2$ much greater than 1 indicates that the expected distribution (in this case, uniform) is unlikely. For our purposes—as evidenced by the accurate performance of the MSE techniques in our experiments (see Section 5)—the low reduced χ^2 validate the uniform distribution. We performed the same test on the profile data from the SPEC2000 benchmarks used in a separate set of experiments also presented in this paper. Experiments on these benchmarks use multiple-sample simulation; thus we were forced to perform the χ^2 test for each fast-forward period individually. The results were similar to those presented in Table 1: suitably close to uniform for the MSE techniques to be applicable. This conclusion is also supported by the successful application of the MSE techniques to the multiple-sample simulations (see Section 5).

	benchmark	$\tilde{\chi}_o^2$ $(d = 254)$
$N = 512$	compress	0.001
	gcc	0.587
	go	1.166
	ijpeg	0.534
	m88ksim	0.014
	perl	0.162
	benchmark	$\tilde{\chi}_o^2$ $(d = 510)$
$N = 1024$	compress	0.000
	gcc	0.275
	go	0.000
	ijpeg	0.317
	m88ksim	0.022
	perl	0.294
	benchmark	$\tilde{\chi}_o^2$ $(d = 1022)$
$N = 2048$	compress	0.003
	gcc	0.925
	go	0.000
	ijpeg	0.242
	m88ksim	0.000
	perl	0.350
	benchmark	$\tilde{\chi}_o^2$ $(d = 2046)$
$N = 4096$	compress	0.000
	gcc	0.037
	go	0.000
	ijpeg	0.303
	m88ksim	0.000
	perl	0.350

Table 1. $\tilde{\chi}_0^2$ for SPECInt95 benchmarks and various N; d is the number of degrees-of-freedom

3.3 Direct-mapped Formula

To calculate the probability that αN sets of a direct-mapped cache will be touched at least once, the MSE formula calculates 1 minus the probability that $N - \alpha N + 1$ or more entries will go untouched:

$$p = 1 - \frac{\sum_{k=1}^{\lceil \alpha N \rceil - 1} \binom{N}{k} k^m}{\sum_{k=1}^{\lceil \alpha N \rceil} \binom{N}{k} k^m}$$

The numerator is the sum of the number of ways to touch at most $\alpha N - 1$ entries after m unique references. In other words, the numerator counts the number of ways to build a string of length m with fewer than αN symbols (*i.e.*, the number of ways to fail to touch αN sets). The denominator counts the number of ways to fail to touch αN sets plus the number of ways to succeed to do so. For $m \geq \alpha N$ their quotient is the probability of failing to touch αN sets at least once. One minus this quotient is the probability of succeeding to touch αN entries after m unique references.

3.4 Set-associative Formula

The formula for calculating the set-associative case is quite different:

$$p = \frac{\sum \left[\binom{m}{x_1, x_2, \ldots, x_{N-1}} \,\middle|\, s.t.\ at\ least\ \lceil \alpha N \rceil\ x_j \geq \lceil \beta a \rceil \right]}{\sum \binom{m}{x_1, x_2, \ldots, x_{N-1}}}$$

The sum of multinomial coefficients in the numerator is the total number of ways to succeed to touch αN sets at least βa times. (This is achieved by adding the restriction that all the lower terms [*i.e.*, x_j] in the multinomial coefficient be greater than or equal to βa.) Naturally, the denominator is the total number of ways to fail to touch αN sets at least βa times plus the number of ways to succeed to do so. For $m \geq \alpha N \beta a$, their quotient is the probability to succeed to touch αN sets at least βa times after m unique references to the set-associative cache.

Notice that neither sum has explicit bounds. Rather, the bounds are implicit—inherent by the definition of the multinomial coefficient: Essentially, the sum of all the lower terms must be less than or equal to the upper term. Thus, the exclusion of explicit bounds implies that the sum is over all valid multinomial coefficients having upper term m. Further restrictions are placed on the numerator to filter out those multinomials representing the case of failing to touch αN sets βa or more times.

3.5 Computing The MSE Formulas

Unfortunately, we have not yet found closed-form solutions for m given p for these formulas. Instead, we have written software that iteratively tests different values of m to find the appropriate m for a specified p, αN, and βa.

The software calculates the direct-mapped formula relatively quickly even on an older, 180MHz PentiumPro—the longest calculation we tried took approximately 30 minutes.

The multinomial coefficient in the set-associative formula, however, makes this calculation take many orders of magnitude longer to compute. The complexity of this computation is bounded by the complexity of calculating the sum in the denominator. The problem with performing this computation is the absence of explicit bounds on the sum which forces us to account for all valid combinations of lower terms in the multinomial coefficient. To combat this, we have tested brute-force multithreading techniques in conjunction with optimizations that exploit combinatorics to avoid "double-counting," *e.g.*, $\binom{8}{1,2,4} = \binom{8}{2,4,1}$. In addition, a single pass computes p for all associativities $\leq a$ for a given αN. A less expensive algorithm for calculating the set-associative formula is a point for future research. A simple approximation to the set-associative MSE formula is to multiply the result of the direct-mapped MSE formula by the associativity, *i.e.*, $m = a \cdot MSE(\alpha N, 1, p)$. This approximation essentially emulates the case where the first m uniques successfully touch the required αN sets; the second m uniques touch αN sets; ...; the a^{th} m uniques touch αN sets. This approximation's imprecision is due to the fact that it does not ensure that the αN sets touched after each bundle of m uniques is the same αN sets touched by the previous m uniques. However, in the case that $\alpha = 1$, this is a good and indeed conservative approximation.

4 Experimental Methodology

We performed two sets of experiments. In the first set, we sought merely to establish the validity and viability of the MSE approach by testing it against the single-large-sample technique and SPECInt95 [10] fast-forward intervals described in [8]. In the second set, we verified MSE's flexibility by applying the MSE techniques to the multiple-sample simulation technique discussed in [2]; the benchmarks for these experiments come from the more up-to-date SPEC2000 [9] suite.

We wrote software to perform the MSE calculations according to the formulas described in Section 3 that return a warm-up interval t, given N, a and p, by iterating over several values for m. For the first set of experiments, we chose our baseline probability of accuracy p to be 99.9%. To achieve probability $p = 99.9\%$, we found that a consistently good starting point for the MSE calculation of m is $16\alpha N\beta a$. (For direct-mapped cache configurations, this simplifies to $m \geq 16\alpha N$.) We also test the MSE calculations for $p \in \{99.0\%, 95.0\%\}$. (Table 2 shows for direct-mapped caches, the necessary m for lower probabilities of accuracy of 99.0% and 95.0% and gives the change in m relative to the m required for 99.9% probability of accuracy.)

	N	m	Δm
p=99.0%	512	5544	-32.32%
	1024	11803	-28.02%
	2048	25031	-23.61%
	4096	52906	-19.27%
	N	m	Δm
p=95.0%	512	4710	-42.50%
	1024	10135	-38.14%
	2048	21693	-33.80%
	4096	46230	-29.46%

Table 2. m summary for $p = 99.0$% and $p = 95.0$% compared to a baseline of $p = 99.9\%$.

For both sets of experiments, we first ran a modified version of *sim-cache* to perform the one-time profiling pass for each benchmark–input pair. Then, using the MSE software in conjunction with the profiling data, we found for each cache configuration, t that would satisfy p.

Next, for the first set we ran the unmodified *sim-cache*—once for each cache configuration—to examine the number of blocks touched after warming up only the t most recent instructions prior to the large-sample window; if the number of blocks touched is at least αN, the experiment was successful.

Finally, for the second set we used a cycle-accurate processor simulator that we developed within the SimpleScalar framework to measure instruction throughput (IPC). The simulator was configured as 6-stage, 4-way, in-order pipeline with hybrid branch prediction, its instruction- and data-cache were 2-way associative with 1024 sets and 32-byte blocks (for a total of 64 kilobytes each). Critical to these experiments was the cache warm-up interval prior to making IPC measurements during the full-detail simulation windows. Warm-up interval was determined in three ways: short, full and MSE. Short warms up cache state for 7,000 instructions ([2]) prior to full-detail simulation. Full warms up cache state during all non-full-detail simulation. MSE warms up cache state for t instructions prior to full-detail simulation. If the IPC obtained by warming up only the MSE-prescribed t instructions is closer to the IPC obtained from full warm-up than the IPC obtained from short warm-up and the simulation running time for MSE warm-up is less that the simulation running time for full warm-up, the experiment was successful.

5 Results

5.1 Single-large-sample

In the first set of experiments, we apply MSE to single-large-sample simulations to validate its effectiveness and accuracy. These experiments model direct-mapped caches with set sizes $N \in \{512, 1024, 2048, 4096\}$. Furthermore,

		$p = 99.9\%$		$p = 99.0\%$		$p = 95.0\%$	
$N = 512$	benchmark	% of N	% of fast-forward	% of N	% of fast-forward	% of N	% of fast-forward
	compress	100%	0.11%	100%	0.08%	100%	0.07%
	gcc	100%	4.16%	100%	3.28%	100%	3.05%
	go	100%	99.90%	100%	99.90%	100%	99.90%
	ijpeg	100%	0.28%	100%	0.12%	100%	0.12%
	m88ksim	100%	1.43%	100%	1.06%	100%	0.05%
	perl	100%	13.89%	99.61%	12.42%	99.02%	10.70%
$N = 1024$	benchmark	% of N	% of fast-forward	% of N	% of fast-forward	% of N	% of fast-forward
	compress	100%	0.25%	100%	0.17%	100%	0.15%
	gcc	100%	5.61%	100%	4.39%	100%	4.33%
	go	100%	99.90%	100%	99.90%	100%	99.90%
	ijpeg	100%	0.51%	100%	0.32%	100%	0.30%
	m88ksim	100%	2.51%	100%	1.80%	100%	0.12%
	perl	100%	14.09%	99.12%	13.95%	99.84%	13.91%
$N = 2048$	benchmark	% of N	% of fast-forward	% of N	% of fast-forward	% of N	% of fast-forward
	compress	100%	0.53%	100%	0.39%	100%	0.33%
	gcc	100%	9.88%	100%	8.16%	100%	6.44%
	go	100%	99.94%	100%	99.94%	100%	99.92%
	ijpeg	100%	0.94%	100%	0.74%	100%	0.69%
	m88ksim	100%	5.03%	100%	3.95%	100%	3.59%
	perl	100%	14.59%	100%	14.37%	100%	14.27%
$N = 4096$	benchmark	% of N	% of fast-forward	% of N	% of fast-forward	% of N	% of fast-forward
	compress	100%	0.96%	100%	0.77%	100%	0.69%
	gcc	100%	15.55%	99.98%	13.83%	99.98%	12.10%
	go	100%	99.99%	100%	99.99%	100%	99.97%
	ijpeg	100%	1.96%	100%	1.63%	100%	1.40%
	m88ksim	100%	10.05%	100%	7.90%	100%	7.16%
	perl	100%	15.57%	100%	15.21%	100%	15.01%

Table 3. SPECInt95 benchmark summary for $p \in \{99.9\%, 99.0\%, 95.0\%\}$.

we assume $\alpha = 1$ since [8] gives warm-up intervals that reach hundreds of millions of instructions within the benchmarks. For these experiments on all benchmarks, with probability of accurate warm-up chosen to be $p = 99.9\%$, all N sets are indeed touched after the prescribed t warm-up instructions as determined by our MSE techniques. When we adjusted the probability to $p = 99.0\%$ and $p = 95.0\%$, the MSE-prescribed t usually touched all N sets; when it did not, only a very small number of sets (fewer than 10) were excluded. Of particular interest in these experiments is the size of the warm-up subset as a percentage of all fast-forward instructions prior to the simulation window. Our data—the percentage of N sets touched and the percentage of the fast-forward interval that was warmed up—from the first set of experiments are given in Table 3.

For all benchmarks except *go*, fewer than 16% of all instructions suffices to touch all or minutely fewer than N sets. *go*'s anomalous behavior is due to the fact that *go* accesses very few unique memory addresses after an initial burst of chaotic startup activity. Hence, it was necessary to go farther back into the set of pre-full-detail simulation instructions to amass the MSE-prescribed t instructions that contained m unique memory references—in all cases, nearly all the instructions. A suitably-chosen $\alpha < 1$ for *go* would mitigate this effect. The small changes in percentage of warm-up instructions as the probability of accuracy, p, decreases are due to the fact that we sought t sufficient to touch all N sets in the cache—thus our simplifying assumption that $\alpha = 1$—even if some lines are dead (*i.e.*, are unused or overwritten before being read during the full-detail simulation). These first-pass results validate the MSE technique and justify the second segment of experiments.

5.2 Multiple-sample

In the second set of experiments, we used multiple-sample simulations. Each of the simulation's 50 full-detail samples was 1 million instructions long and each were separated by 499 million fast-forward instructions. To provide a thorough test of the MSE approach, both the instruction- and data-cache were flushed at the conclusion of each full-detail simulation window. Doing so actually makes MSE's task harder by rendering the simulations unable to take advantage of previously-loaded data that would have otherwise remained in the cache. This is in contrast to the multiple sample method proposed in [2] (that we used to test short warm-up) which opts instead to maintain "stale" state between samples. We tested two probabilities of successful warm-up: $p = 99.9\%$ and $p = 95.0\%$. Once again, we conservatively sought t prior to each full-detail sample sufficient to touch all N sets at least twice ($a = 2$)using the approximation for calculating the set-associative MSE formula described in Section 3.

We measured the goodness of the MSE techniques by two metrics. The first is the closeness of the IPCs obtained using MSE-prescribed warm-up intervals to those obtained by full warm-up. Table 4 shows the results obtained and contrasts them against the IPC obtained through short warm-up. The second metric is the fraction of simulation running time with MSE-prescribed warm-up relative to full warm-up. Table 5 gives these results and contrasts them against the fraction of full warm-up necessary for short warm-up.

From Table 4 it is clear that the MSE-prescribed warm-up in general yields superior IPC precision relative to short warm-up. Of fifteen benchmarks, fourteen yielded IPCs that are closer (*i.e.*, have a smaller absolute value percent difference) than the short warm-up for both $p = 99.9\%$ and $p = 95.0\%$. The singular benchmark *applu* was probably adversely affected by the fresh-state approach our experiments took, flushing the instruction- and data-cache at the conclusion of each full-detail simulation window.

Short warm-up is clearly superior in terms of wall-clock running time for simulations, never taking longer than 22% of the time required for full warm-up simulation. The running times obtained by MSE are also smaller than full warm-up, taking only as much as 86% or as little as 26% for $p = 99.9\%$ and as much as 69% or as little as 22% for $p = 95.0\%$ of the time required by full warm-up. These running times reflect the number of fast-forward instructions that were used for warm-up between the full-detail intervals. In a simulation such as *vpr* (the highest ranking benchmark for $p = 99.9\%$), the explanation for its nearly 86% measurement is the fact that each 499-million-instruction fast-forward was sparsely populated by *unique* memory references. Therefore, the MSE-prescribed t memory references during the fast-forward periods had to be large to capture the m necessary uniques in order to achieve probability p of accurate warm-up. In fact, for some of the benchmarks, several of the fast-forward intervals were so sparse with unique memory references that they did not contain m uniques; for these fast-forward intervals, the MSE-prescribed t is the entire fast-forward window. A more efficient solution to this situation is already under investigation.

In conclusion, short warm-up reduced simulation times by roughly a factor of 2.75 compared to $\text{MSE}_{99.9\%}$ warm-up simulation times and by roughly a factor of 2.05 compared to $\text{MSE}_{95.0\%}$; $\text{MSE}_{99.9\%}$, on the other hand, yields results that are roughly 12.9 times more accurate than short warm-up and $\text{MSE}_{95.0\%}$ is roughly 10.6 times more accu-

		% error		
benchmark	Full	Short	$\text{MSE}_{99.9\%}$	$\text{MSE}_{95.0\%}$
applu	0.7857	-1.425%	-1.769%	-1.769%
crafty	1.3946	-2.373%	-0.029%	-0.029%
equake	0.6146	0.358%	-0.016%	-0.016%
facerec	1.2042	-4.293%	-0.008%	-0.008%
fma3d	0.8492	1.896%	-0.742%	-0.565%
gcc	1.0665	-7.979%	-0.169%	-0.291%
gzip	1.5224	-1.399%	-0.085%	-0.099%
lucas	0.7439	0.255%	0.121%	0.121%
mesa	1.3797	-1.160%	0.210%	0.275%
parser	1.0851	-9.833%	-0.065%	-0.175%
perlbmk	1.0542	-1.916%	0.844%	1.157%
twolf	1.2008	-2.682%	-0.167%	-0.208%
vortex	1.1118	-1.727%	0.072%	-0.063%
vpr	1.0675	-16.42%	-0.019%	-0.206%
wupwise	0.9783	-2.361%	-0.020%	-0.307%
MEAN		3.738%	0.289%	0.353%

Table 4. Result summary for 50-sample simulation IPCs. This table compares the IPC from full warm-up to the percent difference $\frac{(IPC - IPC_{full})}{IPC_{full}}$ in IPC for both short and MSE warm-up. The mean of percent differences was calculated using their absolute values.

		% of original running time		
benchmark	Full	Short	$\text{MSE}_{99.9\%}$	$\text{MSE}_{95.0\%}$
applu	26572	17.53%	54.59%	38.98%
crafty	27175	19.80%	34.01%	27.00%
equake	27220	19.35%	63.14%	47.92%
facerec	26190	20.28%	55.11%	37.93%
fma3d	27591	17.65%	55.96%	42.17%
gcc	28377	19.07%	42.10%	32.30%
gzip	27037	21.24%	62.73%	46.94%
lucas	25739	17.70%	84.07%	68.73%
mesa	26602	20.30%	42.14%	32.48%
parser	27735	17.98%	35.48%	26.59%
perlbmk	27905	18.98%	26.27%	22.59%
twolf	27967	19.64%	47.90%	35.67%
vortex	28301	19.40%	25.49%	22.27%
vpr	28235	19.96%	85.53%	54.67%
wupwise	26173	17.66%	76.32%	54.93%
MEAN		19.10%	52.72%	39.41%

Table 5. Result summary for 50-sample simulation running times (in seconds). This table compares the running times for full warm-up to the percentage of this time for both short and MSE warm-up.

rate (see the MEAN entries of Table 4 and Table 5). This is an exciting result: A decreased probability of accurate warm-up p reduces simulation running time, yet in general still achieves a more accurate IPC measurement than short warm-up. As hypothesized, the MSE technique's rigorous mathematical approach to determining suitable fast-forward warm-up intervals is more reliable than previous, more ad-hoc methods.

6 Conclusions and Future Work

Minimal subset evaluation is a formal mathematical technique for determining a reduced number of instructions necessary to warm-up simulated hardware state. Our research extends prior work by using probability theory to more rigorously make this determination. The MSE formulas presented here are predicated upon a uniform distribution of *unique* memory references throughout the cache; we were able to verify this behavior using the χ^2 test. The test-bed for our research was caches.

A crude heuristic for obtaining m—the necessary number of unique memory references—to obtain a probability of accurate warm-up of 99.9% is $m \geq 16\alpha N\beta a$ where N is the number of sets and a is the associativity of each set; α, $\beta \in (0, 1]$ tune the number of sets and blocks per set to be touched, respectively.

Our results show that MSE yields highly accurate simulations while substantially reducing simulation times. For $p = 99.9\%$, the average error in IPC was 0.3% and the average reduction in simulation running time was 47%; for $p = 95.0\%$, the average error in IPC was 0.4% and the average reduction in simulation running time was 60%. MSE is flexible while maintaining accuracy.

MSE is a new approach; its refinement is the subject of future research. We will next apply MSE techniques to other hardware structures such as the prediction history tables of dynamic branch predictors and to deeper levels of the cache hierarchy. Efficiently dealing with benchmarks that have a sparse population of unique memory references is also a topic for future research. Finally, the development of a mathematically tractable solution for computing the set-associative MSE formula is a point of further research. The current algorithm's extremely long running time makes it suitable only for computations with a very small number of sets (*e.g.*, $N = 16$). For the case where $\alpha = 1$, a conservative approximation to the set-associative formula is a times the direct-mapped MSE solution, *i.e.*, $m = a \cdot MSE(N, 1, p)$.

Acknowledgments

This material is based upon work supported in part by the National Science Foundation under grant no. CCR-0082671. The authors would also like to thank Dr. Dee A. B. Weikle, Prof Margaret Martonosi and the anonymous reviewers for their valuable feedback and insights.

References

[1] T. M. Austin. SimpleScalar home page. http://www.simplescalar.org.

[2] T. M. Conte, M. A. Hirsch, and K. N. Menezes. Reducing state loss for effective trace sampling of superscalar processors. In *Proceedings of the 1996 International Conference on Computer Design*, Oct. 1996.

[3] E. N. Elnozahy. Address trace compression through loop detection and reduction. In *Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 214–15, May 1999.

[4] R. E. Kessler, Mark D. Hill, and David A. Wood. A Comparison of Trace-Sampling Techniques for Multi-Megabyte Caches. Technical Report 1048, University of Wisconsin-Madison Computer Sciences Department, Sep. 1991.

[5] AJ KleinOsowski, J. Flynn, N. Meares, and D. J. Lilja. Adapting the SPEC 2000 benchmark suite for simulation-based computer architecture research. In *Proceedings of the Third IEEE Annual Workshop on Workload Characterization*, pages 73–82, Sep. 2000.

[6] T. Lafage and A. Seznec. Choosing representative slices of program execution for microarchitecture simulations: A preliminary application to the data stream. In *Proceedings of the Third IEEE Annual Workshop on Workload Characterization*, pages 102–110, Sep. 2000.

[7] Subhasis Laha, Janak H. Patel, and Ravishankar K. Iyer. Accurate Low-Cost Methods for Performance Evaluation of Cache Memory Systems. *IEEE Transactions on Computers*, 37(11):1325–1336, Nov. 1988.

[8] K. Skadron, P. S. Ahuja, M. Martonosi, and D. W. Clark. Branch prediction, instruction-window size, and cache size: Performance tradeoffs and simulation techniques. *IEEE Transactions on Computers*, 48(11):1260–81, Nov. 1999.

[9] Standard Performance Evaluation Corporation. SPEC CPU2000 Benchmarks. http://www.specbench.org/osg/cpu2000.

[10] Standard Performance Evaluation Corporation. SPEC CPU95 Benchmarks. http://www.specbench.org/osg/cpu95.

[11] J. R. Taylor. *An Introduction to Error Analysis: The Study of Uncertainty in Physical Measurements*. University Science Books, Mill Valley, California, 1982.