# Accelerated Warmup for Sampled Microarchitecture Simulation

JOHN W. HASKINS, JR.
Center for Computing Sciences
and
KEVIN SKADRON
University of Virginia

To reduce the cost of cycle-accurate software simulation of microarchitectures, many researchers use statistical sampling: by simulating only a small, representative subset of the end-to-end dynamic instruction stream in cycle-accurate detail, simulation results complete in much less time than simulating the cycle-by-cycle progress of an entire benchmark. In order for sampled simulation results to accurately reflect the nature the full dynamic instruction stream, however, state in the simulated cache and branch predictor must match or closely approximate state as it would have appeared had cycle-accurate simulation been used for the entire simulation. Researchers typically address this issue by prefixing a period of warmup—in which cache and branch predictor state are modeled in addition to programmer-visible architected state—to each cluster of contiguous instructions in the sample.

One conservative, but slow approach is to always simulate cache and branch predictor state, whether among the cycle-accurate clusters, or among the instructions preceding each cluster. To save time, warmup heuristics have been proposed, but there is no one-size-fits-all heuristic for any benchmark. More rigorous, analytical warmup approaches are necessary in order to balance the requirements of high accuracy and rapidity from sampled simulations. This paper explores this issue and in particular demonstrates the merits of *memory reference reuse latency* (MRRL).

Relative to the IPC measured by modeling all precluster cache and branch predictor activity, MRRL generated an average error in IPC of less than 1% and simultaneously reduced simulation running times by an average of approximately 50% (or 95% of the maximum potential speedup).

Categories and Subject Descriptors: C.0 [**General**]: Modeling of Computer Architecture

General Terms: Measurement, Performance

Additional Key Words and Phrases: Reuse latency, sampled simulation, warmup

This work was supported in part by the National Science Foundation under grant no. CCR-0082671.
Authors' addresses: J. W. Haskins, Jr., Center for Computing Sciences, 17100 Science Drive, Bowie, MD 20715-4300; email: predator@super.org; K. Skadron, Department of Computer Science, School of Engineering and Applied Science, University of Virginia, 151 Engineer's Way, PO Box 400740, Charlottesville, VA 22904-4740.

## 1. INTRODUCTION

Highly detailed, cycle-by-cycle simulation of a microprocessor in software is prohibitively slow. On the fastest hardware, simulation commonly proceeds several orders of magnitude slower than native execution. KleinOsowski et al. [2000] show that modeling SPEC CPU2000 [Standard Performance Evaluation Corporation 1999] benchmarks in cycle-accurate detail on reference inputs can take many weeks. Still, software simulation is fundamental to all computer architecture research. To make cycle-accurate simulation a more tractable alternative, KleinOsowski and Lilja [2002] propose MinneSPEC: a novel workload of reduced inputs for the SPEC CPU2000 benchmarks that simulate to completion in significantly less time than original reference inputs, thereby allowing researchers to explore a larger space of simulator configurations in a reasonable amount of time. For several of the benchmarks, the MinneSPEC workload also mimics execution behaviors (e.g., function-level execution pattern, instruction mix) seen with the reference inputs.

Another approach to make simulation-driven research tractable used by many studies is to employ *sampling*: taking measurements from a small, representative subset of the end-to-end dynamic instruction stream. The remainder of this paper specifically concerns acceleration of simulation that uses sampling. Since it is precisely the software simulation of the cycle-by-cycle progression of individual instructions through the pipeline that produces the overwhelming slowdowns, in sampled simulation only the subset of instructions which constitutes the sample are modeled in cycle-accurate detail. Fortunately, measuring the instruction throughput (i.e., instructions per cycle, IPC) of only a subset of the instructions can—for a properly chosen subset—yield information about the instruction throughput of a benchmark's entire end-to-end execution. Conte et al. [1996], Sherwood et al. [2001, 2002], Skadron et al. [1999], Perelman et al. [2003], and Wunderlich et al. [2003] propose strategies for choosing representative samples that yield good approximations to the true end-to-end IPC; all of these will be discussed in Section 2.

To ensure the integrity of sampled measurements, the simulated processor state must be accurately established prior to the cycle-accurate simulation of each cluster. In other words, accuracy is predicated upon successfully defeating the so-called *cold-start* bias; because cache and branch predictor performance are critical to microprocessor performance, if the state of the cache (at all levels of the hierarchy) and branch predictor do not appear at least approximately as they would have had the entire simulation been performed in cycle-accurate detail as cycle-accurate simulation of a cluster begins, the simulation results may be inaccurate.

One straightforward technique to guarantee the accuracy of cache and branch predictor state is to model the interaction of each memory reference—instructions and data—with the cache hierarchy and every control-flow instruction with the branch predictor while the simulator is executing precluster instructions. (All cache and branch predictor interactions are already modeled within the cycle-accurate clusters.) Though its accuracy is sound in terms of cache and branch predictor state, this FULLWARMUP method is heavy handed.

While not as expensive (in terms of simulation running time) as cycle-accurate simulation, modeling all cache and branch predictor interactions is still costly.

To accelerate sampled simulation even further, one can avoid FULLWARMUP by only modeling those interactions that occur within a certain number of instructions prior to each sample cluster [Conte et al. 1996; Crowley and Baer 1999; Eeckhout et al. 2003; Haskins, Jr. and Skadron 2001, 2003; Kessler et al. 1991; Nguyen et al. 1997]. This approach exploits temporal locality [Hennessy and Patterson 1995]: the propensity of programs to demonstrate a strong correlation between recency of use and next use, for example, of cache blocks. In other words, *of those references that precede each sample cluster, references that occur nearest to the cluster are the most likely to be necessary for the accurate simulation of the cluster itself*. Hence, by modeling cache and branch predictor activity for only a subset of these precluster *warmup instructions*, the simulated cache and branch predictor state can still approximate the state that would have resulted had cycle-accurate simulation or FULLWARMUP been used.

Heuristic methods for determining the number of warmup instructions propose warming up a fixed number of instructions preceding each sample cluster [Conte et al. 1996], a fixed percentage of preceding instructions [Crowley and Baer 1999; Kessler et al. 1991], and/or simply recycling cache and branch predictor state as it existed at the conclusion of the previous cluster [Crowley and Baer 1999]. Haskins, Jr. and Skadron [2001] showed, however, that fixed-length warmup, such as the 7000-instruction duration proposed in Conte et al. [1996], for branch-predictor warmup[1] does not warmup cache state as effectively as the more rigorous, adaptive MSE. While 7000-instruction warmup produced an error of less than 4% relative to FULLWARMUP on average, such a technique cannot be reliably applied to caches because the results are not "trustworthy." This becomes apparent when one considers that the worst-case relative error produced by 7000-instruction warmup was almost 10%—more than five times worse than MSE's worst-case relative error. This result suggests that there is no one-size-fits-all fixed-length warmup duration: Some fixed-length approaches would fail to warmup enough references to ensure precise cycle-accurate simulation, while others will fail to optimally accelerate simulation by warming up too many instructions. An analogous argument can be made of warmup techniques that propose warming up some fixed percentage of instructions that precede each cycle-accurate cluster; this is corroborated by Crowley and Baer [1999], which claims that warming up 50% of references does not reliably eliminate bias for large caches. On the contrary, our research specifically compares warmup techniques that—according to some analytical model—ensure accurate warmup with high probability and produce as much speedup as possible within that analytical framework. Neither of these requirements is met by warmup techniques that specify an arbitrary amount of warmup. Hence, since the superiority of analytical methods has already been established we do not experiment with fixed-number or fixed-percentage warmup techniques in this paper.

[1]Conte et al. [1996] assume perfect cache warmup.

This paper compares minimal subset evaluation (MSE) and memory reference reuse latency (MRRL—MSE's progeny): two rigorous techniques proposed by the authors for accelerating sampled microarchitecture simulations by reducing the amount of cache and branch predictor warmup prior to each sample cluster. MSE determines the number of warmup instructions based on the number of unique memory references that would have to be modeled in order to touch a certain fraction of cache blocks with user-chosen probability $p$. MRRL determines when to engage cache and branch predictor warmup by exploiting *memory reference reuse latencies* (MRRL)—a count of the number of instructions that elapse between successive references to the same address.

In this paper, our objective is not to explore tactics for effective sampling; our experiments in this paper use random cluster sampling for ease of deployment, precedent [Conte et al. 1996; Haskins, Jr. 2003; Haskins, Jr. and Skadron 2003], and amenability to rigorous statistical analysis. Rather, having argued and cited the deficiencies of heuristic methods, our objective is *to demonstrate definitively the superiority of MRRL over its predecessors in term of flexibility, applicability, and amount of speedup achieved*. It should be noted, however, that while our experiments for this paper use random cluster sampling, one of the chief goals in the development of MSE and MRRL was to develop warmup methodologies that are, to the extent possible, independent of, and therefore useful with, other sampling techniques [Perelman et al. 2003; Sherwood et al. 2001, 2002; Skadron et al. 1999; Wunderlich et al. 2003].

Another significant contribution of this research has been the development of software that implements the techniques described in this paper; we have additionally implemented significant modifications to the *sim-outorder* component of the SimpleScalar [Austin and Burger 1998; Burger and Austin 1997] tool set which facilitate sampled simulation, and MRRL-based warmup. These tools are linked to from the Web site for the Laboratory for Computer Architecture and Virginia (LAVA) at http://lava.cs.virginia.edu/. Our chief contribution has been the development of two techniques that preserve simulation accuracy while significantly accelerating sampled simulation, thus rendering software simulation a more attractive technique for microarchitecture research.

The rest of this paper is organized as follows. We discuss related work in Section 2. Section 3 briefly reviews the MSE and MRRL acceleration techniques and discusses their application to sampled simulation. Section 4 contrasts MSE and MRRL and describes why MRRL is superior to its predecessor. Finally, we explain our experimental methodology in Section 5, present our results in Section 6, and conclude in Section 7.

## 2. RELATED WORK

Several studies examine ways to reduce overall simulation running times by executing only a small subset of the benchmark in cycle-accurate detail. Skadron et al. [1999] identify short, representative simulation windows of 50 million instructions for the SPECInt95 benchmarks. The key insight which guides their approach is to observe code segment execution frequencies to distinguish and bypass unrepresentative start-up (e.g., data structure setup and

initialization) behavior; cycle-accurate simulation begins after fast-forwarding beyond a benchmark's start-up phase.

Ochoa et al. [1996] and Conte et al. [1996] take a different approach and instead simulate multiple fixed-sized clusters of contiguous instructions from the complete dynamic instruction stream. The former opts for uniformly distributed sample clusters, whereas the latter opts for randomly distributed sample clusters. In spite of warming up all noncluster instructions, by modeling only a small fraction of the end-to-end instruction stream in cycle-accurate detail Ochoa et al. achieve a significant speedup while accurately estimating instruction latency within a simulated pipeline. Conte et al. realize further speedup by recycling stale predictor state from the previous cluster and prefixing a short warmup interval of at least 7000 instructions prior to each cluster, and achieve very small errors of a few percent in the observed mean IPC. In the experiments conducted for this research we used random cluster sampling and similar to Conte et al. [1996], break the precluster instructions into two segments with the latter segment composing the warmup interval determined by MSE/MRRL and recycling stale cache and branch predictor state.

Sherwood et al. [2001] propose *basic block distribution analysis* (BBDA). Their technique profiles the execution frequency of a benchmark's basic blocks in order to isolate a contiguous subset of the dynamic instruction stream whose execution characteristics closely mimic the complete, end-to-end execution of the benchmark. BBDA's key insight is that periodic basic block execution frequency behavior reflects the periodicity of various architectural metrics such as IPC, cache miss rate, and branch predictor accuracy in cycle-accurate simulation. In Sherwood et al. [2002], Sherwood et al. build upon the BBDA concept to create a technique that automatically isolates multiple contiguous subsets of the dynamic instruction stream since some benchmarks' behavior is too complex to be characterized by a single instruction stream slice. As a continuation of this research, Perelman et al. [2003] analyze algorithms that target the early dynamic instruction stream to find simulation points that closely approximate the end-to-end execution; because these simulation points occur close to the beginning of a benchmark's dynamic instruction stream, less simulation time is required to arrive at the sample clusters. In all three cases their aim is to reduce simulation times by only executing in cycle-accurate detail, a small representative subset of the dynamic instruction stream. MSE and MRRL are sampling regime agnostic; therefore, as is hypothesized in Perelman et al. [2003] (which states that their research uses "perfect warmup"), both should be well suited to the task of accelerating warmup prior to cycle-accurate modeling of their simulation points. Haskins, Jr. validates this hypothesis for MRRL in Haskins, Jr. [2003] by demonstrating that MRRL achieves well over 90% of the maximum potential speedup while diverging from the IPC measured by FULLWARMUP by much less than 1% on average for the automatically chosen simulation points published in Sherwood et al. [2002].

The SMARTS framework, by Wunderlich et al. [2003], proposes a rigorous statistical foundation for microarchitecture simulation built upon sampling theory to analytically determine a subset of the population of a benchmark's dynamic instruction stream to simulate in cycle-accurate detail. Their work

also explores the importance of warming up simulated hardware state, performing warmup in two stages: functional and detailed. "Functional warming" models cache and branch predictor activity in addition to functional simulation of programmer-visible hardware state; "detailed warming" models the microarchitecture in cycle-accurate detail, but does not gather performance statistics. Their research prepends a period of detailed warming to each simulation cluster; functional warming is performed prior to each period of detailed warming. MSE and MRRL specifically address minimizing the amount of what Wunderlich et al. refer to as functional warming, and is potentially useful for achieving further speedup of SMARTS simulations without compromising accuracy of the results. MSE and MRRL will only reduce the amount of functional warming, however, if doing so will not jeopardize simulation accuracy. Hence, for MSE or MRRL to reduce simulation times, there must be sufficient instructions between the conclusion of a sample cluster and the commencement of the subsequent period of detailed warming to justify exorcising some fraction of the inter-cluster warmup. In their paper, Wunderlich et al. determine the number of sample clusters to simulate, $n$, based (among other parameters) on a recommended sample cluster size of $U = 1000$ instructions apiece, starting with $n = 10,000$ and adjusting as necessary to achieve the desired level of confidence in the measured instruction latency (i.e., cycles per instruction or CPI). With so many small clusters, MRRL and MSE in particular, may not be able to reduce warmup time, degenerating instead to FULLWARMUP to preserve simulation accuracy. (The impact of intercluster distance upon MSE is demonstrated in Section 6.) This may be alleviated, however, by selecting fewer, larger samples; for example, let $U = 100{,}000$ and $n = 100$. This is a topic for future research.

Heuristics for reducing cold-start bias are studied by Kessler et al. [1991]. They consider using half of a sample's references for warmup purposes; tracking only entries that are known to contain good state, recycling stale state, and flushing state but estimating how much error this introduces. This method, however, of branch predictor warmup described in Conte et al. [1996] and the method of cache warmup described in Crowley and Baer [1999] may compromise accuracy if used to warm up cache sate. If, for instance, a fixed number (e.g., 7000) or fixed percentage (e.g., 50%) of warmup instructions is insufficient to accurately establish cache state, then simulation accuracy will be compromised. On the other hand, if a fixed number or fixed percentage of warmup instructions accurately warms up cache state when a smaller amount of warmup would have established accurate state in less time, then the heuristic has wasted time. It is impossible to state for all possible benchmark programs and input sets, some fixed quantity of warmup that will yield accurate, efficient warmup for cache and branch predictor state. For this reason, these static warmup methods are not included in our study. (Application of the fixed, 7000-instruction warmup to cache state was demonstrated to be untrustworthy in Haskins, Jr. and Skadron [2001].)

Nguyen et al. [1997], on the other hand, approach the problem of warmup analytically as a part of the trace-driven PARSIM parallel microprocessor simulation system. Their formula calculates a function of the cache block width,

associativity, the average population density of memory references within the instruction stream, and the average steady-state cache miss ratio. This solution is a substantial improvement over previous techniques; by approaching the problem analytically, their technique is able to achieve rapid warmup without compromising accuracy. MSE similarly uses probabilistic calculations to estimate a minimal warmup duration. MSE and MRRL are more flexible, however, in that neither imposes the cumbersome requirement of a priori knowledge of the steady-state cache miss ratio; the only straightforward way to determine this quantity would be to measure it directly (e.g., via FULLWARMUP) for each cache configuration under investigation. MSE and MRRL, on the other hand, incur a onetime profiling cost for any sample drawn from a benchmark's end-to-end execution that can be used to warmup any cache configuration (and, in the case of MRRL, any branch predictor configuration). For this reason, the warmup technique employed by the PARSIM research is not entirely comparable to MSE or MRRL, and is therefore not included among the warmup techniques measured in this paper.

Eeckhout et al. [2003] address the problem of accurately warming up cache state for sampled trace-driven cache research. Their approach defeats cold-start bias by initializing warmup according to those unique reference addresses that are touched during the sample clusters, capitalizing upon the insight that only as many precluster instructions as contain those references need to be warmed up to establish accurate state. This strategy can also be tuned by opting to warmup according to some user-chosen percentage of the unique references that are touched during the clusters. When measuring the sampled trace miss rate, they achieve results very close to the miss rate measured by simulating the whole trace by warming up a precluster interval containing at least 70% of the unique memory addresses touched in the cluster and recycling stale cache state from the previously simulated cluster. Application of their technique to the accurate establishment of branch predictor state, and for accurate sampled pipeline simulations is a subject for future research.

Phalke and Gopinath [1995] model *interreference gaps* (which are equivalent to memory reference reuse latencies) as $k$th-order Markov chains. By modeling peraddress temporal locality in this way, they were able to develop improved algorithms for page replacement, dynamic memory management, and trace compression. A possible topic for future research is to determine whether reuse latencies thus modeled (rather than measured directly as with MRRL) yield warmup windows that reduce simulation running time while warming up cache and branch predictor state accurately. Thiébaut [1989] draws an analogy between memory access patterns and fractal random walks on the one-dimensional lattice (where a large memory address space mimics the countably infinite lattice). From this framework, Thiébaut describes a method for accurately predicting the miss ratio of fully associative caches. Wood et al. [1991] establish the concept of *cache generations*. Each cache generation begins immediately after a new line is brought into the cache and ends when the line is evicted and replaced. Their notion of cache generations establishes a framework for analytically estimating the *unknown* or *cold-start*

reference miss ratio, $\mu$. They further establish that $\mu$ is substantially higher than the miss ratio of references chosen at random. Armed with reliable $\hat{\mu}$—estimated unknown reference miss ratio—they were able to accurately estimate cache miss ratios in sampled trace-driven simulations. While these works do not treat warmup in execution-driven simulation, they were instructive in their analytical assessment of temporal locality in memory reference streams.

In their cache decay research, Kaxiras et al. [2001] use the notion of cache generations to propose a technique of cutting power to (heuristically presumed) dead cache lines, thereby reducing leakage power. For the SPEC CPU2000 benchmarks, their measurements show that for a 32 KB L1 data-cache, a cache line's dead time can range from 45% to as much as 99% of the total time since being loaded. Their work shows that most cache lines' active lifetime is significantly longer than their useful lifetime, which confirms the MRRL hypothesis that references occurring many instructions before a cluster are unlikely to have any relevance within the cluster and can therefore be safely omitted from warmup.

Girbal et al. [2003] describe a novel technique for accelerating simulation by distributing the job across a number of independently executing CPUs. If, for instance, $S$ CPUs are available, then a benchmark's end-to-end dynamic instruction stream can be broken into $S$ mutually exclusive subsets and simultaneously simulated. Key to their research is synchronizing the simulated state as it appears at the beginning of subset $R$, with the simulated state as it appears at the conclusion of subset $R-1$. To ensure this synchrony, the CPU handling subset $R-1$ is allowed to continue simulating instructions beyond its allotted instruction count, concurrent with the simulation of subset $R$. When the difference between the measurement of some parameter (e.g., instruction throughput) falls below a user-specified threshold value for $R-1$ and $R$, the two are said to be synchronized; hence, subset $R$ may safely proceed, accurately warmed up.

## 3. ACCELERATING WARMUP

As in prior research, we achieve efficient execution by breaking the simulation into three separate phases as illustrated in Figure 1. The first, aggressive fast-forward phase can be considered the "cold" phase; this is followed by the "warm" phase, where cache and branch predictor interactions are modeled; and concluded by the "hot" phase where cycle-accurate simulation of the processor pipeline takes place. The hot phase contains sample cluster instructions and preceding cold and warm phases contain the precluster instructions. Hence, for each precluster–cluster pair, the aim of our research is to preserve simulation accuracy as we increase the duration of the cold phase while reducing the duration of the warm phase, always leaving the hot phase unchanged. Ad hoc warmup methods that guess a warmup amount (e.g., $X\%$ of all precluster instructions) may yield inaccurate results (if warming up only $X\%$ of precluster instructions is too few) or fall short of the potential speedup (if warming up fewer than $X\%$ of precluster instructions would have still yielded accurate

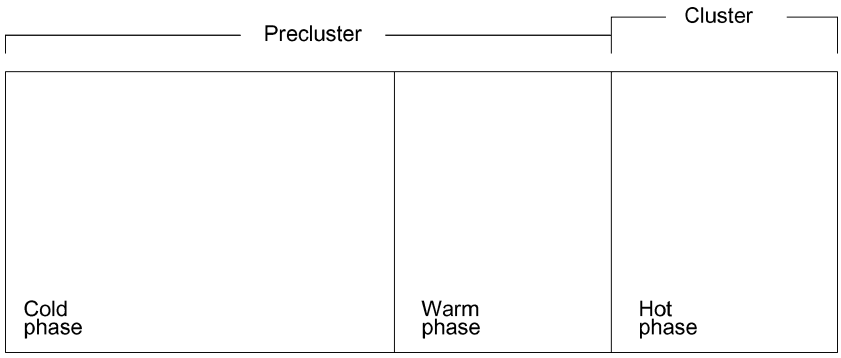


Fig. 1. Precluster–cluster pair subdivided into cold, warm, and hot phases. Cold phase models only architected state; warm phase models architected state, plus cache hierarchy, plus branch predictor; hot phase models pipeline in cycle-accurate detail.

results). Our techniques eschew these heuristics in favor of a more analytical approach and preserve accuracy by determining which references are likely to be germane to each cycle-accurate cluster.

### 3.1 Minimal Subset Evaluation

*Minimal subset evaluation* (MSE) [Haskins, Jr. 2003; Haskins, Jr. and Skadron 2001; Haskins, Jr. et al. 2002] computes the probability $p$ of touching some fraction of cache blocks after accesses to $m$ unique memory reference addresses. This probability is computed as a function of the number of cache sets and the degrees of associativity per set. Acceleration is achieved by warming up only the probabilistically minimal subset of precluster instructions as contain the $m$ unique references. Following this warmup period, the given fraction of cache blocks will have been touched at least once with probability $p$.

To thoroughly discuss MSE it is helpful to define several more variables. Let $N$ be the number of sets in the cache and $a$ be its associativity. (For a direct-mapped cache therefore, $a = 1$.) In its most basic form, the *MSE formula* is used to calculate $m$: the number of unique reference addresses that must be handled within the cache to touch all $Na$ cache blocks with probability $p$; that is, $m = \text{MSE}(N, a, p)$. To compute the $m$ necessary to touch only a fraction of the sets, and fraction of the blocks within each set, the basic MSE formula is adapted by adding two "tuning" variables, $\alpha, \beta \in (0, 1]$, thus: $m = \text{MSE}(\alpha N, \beta a, p)$. (The task of choosing good $\alpha$ and $\beta$ a priori is discussed in Section 4.)

It is interesting to note that the MSE formula does not compute $m$ directly, but rather can be used to iteratively determine $m$ based on the user-chosen $p$:

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

We have not derived a closed-form solution for $m$, but have written software that avoids the intractable computation of the pure MSE formula, and rapidly converges to $m$ for any given $p$ [Haskins, Jr. 2003], using the *direct-mapped*

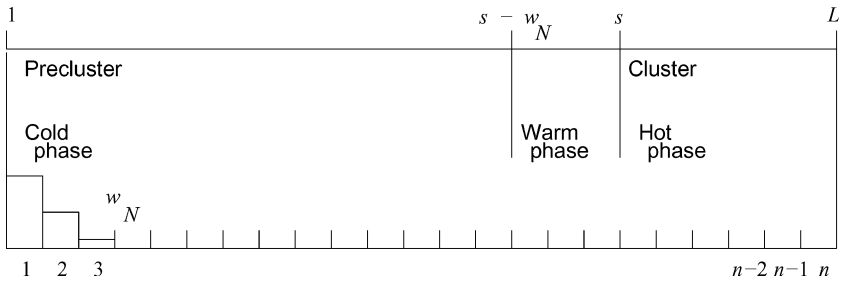


Fig. 2. Precluster–cluster pair as the discrete interval $[1, L]$ partitioned into $n$ mutually exclusive buckets to form the $\delta insn$ histogram; here $w_N \mapsto i$ bucket$_3$, therefore warmup begins $w_N$ instructions prior to instruction $s$ which borders the cluster.

*MSE approximation*:

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

To compensate for the fact that the approximation assumes a direct-mapped cache ($a = 1$), the MSE approximation becomes $m = a \cdot \text{MSEapprox}(N, p)$, which approximates the quotient of the number of ways to select fewer than $N$ cache sets, and the number of ways to select at most $N$ cache sets.

Prior to simulation, we use custom-made software to profile each benchmark's precluster–cluster pairs, to measure the number of unique memory reference addresses that are accessed prior to each cluster. From this, $t$—the number of instructions preceding a sample cluster that contains $m$ unique reference addresses—was determined. This profiling step is a onetime cost per set of precluster–cluster pairs. Only $m$ must be recomputed for a different cache configuration; from this, $t$ can be determined from previously gathered profile data.

### 3.2 Memory Reference Reuse Latency

*Memory reference reuse latency* (MRRL) [Haskins, Jr. 2003; Haskins, Jr. and Skadron 2003] refers to the number of completed instructions between a reference to some memory address M[$A$] and the next reference to M[$A$]. To facilitate a rigorous discussion of MRRL, we must establish a relationship between the $L$ instructions in a single precluster–cluster pair and the elements of the discrete interval $[1, L]$; let instruction$_i \mapsto i$, for $i \in \{1, 2, \ldots, L\}$. Imagine further, that $[1, L]$ is partitioned into $n \ll L$ mutually exclusive buckets whose union is exactly $[1, L]$, as pictured in Figure 2.

For our research, we developed software for profiling MRRLs for each precluster–cluster pair. The profiling software maintains several associative arrays of memory reference addresses—one for the instruction stream, one for the data stream, and one for the stream of branch instructions. Elements in the array are ordered pairs containing a memory address and a logical timestamp. As a precluster–cluster pair simulates, the array element corresponding to the currently accessed memory address is timestamped with the number of simulated instructions since the beginning of the current precluster–cluster pair; if

a previously encountered address is re-accessed, the difference of the previous timestamp and the current number of completed instructions is temporarily stored as $\delta insn$. These $\delta insn$ are used to concurrently build a reuse latency histogram by incrementing the count of the bucket, $[a, b]$, for which $\delta insn \in [a, b]$. When a precluster–cluster section concludes, the profiler outputs its $\delta insn$ histogram.

A precluster–cluster histogram gives the number of references whose reuse latencies fall within the $n$ disjoint length intervals of $[1, L]$. In other words, each histogram gives the count of references for which the number of elapsed instructions between successive accesses to the same address lies within the interval subset $\text{bucket}_j$, where $j \in \{1, 2, \ldots, n\}$ for all $n$ buckets. Not surprisingly, the histograms invariably tell the same story when plotted: an overwhelming majority of references are revisited a small number of instructions after their most recent access (i.e., the histogram bucket with the largest population was always $\text{bucket}_1$). Thus, the more instructions that complete after an access to M[$A$], the less likely M[$A$] is to be accessed again. This is exactly as we had expected, in light of concepts pioneered in Wood et al. [1991] and subsequent work in Kaxiras et al. [2001].

From the histograms, we calculated the reuse distance corresponding to any desired *percentile* $N$, that is, the $\text{bucket}_j$ for which at least $N\%$ of references are contained in $\sum_{k=1}^{j} \#_{\text{refs}}(\text{bucket}_k)$. Let $w_N \mapsto \text{bucket}_j$ mean that the $j$th bucket of the $[1, L]$ interval is upper-bounded at $L - w_N$ instructions into the precluster–cluster pair. In other words, of all the references in the current precluster–cluster pair, $N\%$ have reuse latencies of less than $w_N$ instructions.

By engaging warmup $w_N$ instructions prior to the current precluster–cluster boundary for large enough[2] $N$, we know that the overwhelming majority of addresses that will be accessed during the simulation cluster will have been initialized. We argue that if $N\%$ of references require only $w_N$ instructions between successive accesses, then it is pointless to attempt to initialize the minority ($(100 - N)\%$) of precluster cache and branch predictor interactions that occur more than $w_N$ instructions before the cluster, since these references would require disproportionately long to warm up, and probably *not* be relevant to the cluster's precision.

### 3.3 Sampled Simulation Acceleration

The steps of MSE/MRRL warmup acceleration are enumerated below:

(1) First, the user selects the locations of the cycle-accurate clusters within the benchmark; by corollary noncluster regions are selected simultaneously. Each cluster is paired with its own preceding noncluster (i.e., *precluster*) region.

(2) The user next profiles the benchmark to characterize, for each precluster–cluster pair, the occurrences of unique addresses (MSE) or reuse latencies (MRRL) among all references that occur. As this profile data is valid for

[2]A discussion of "large enough" $N$ appears in Section 6.

any cache and branch predictor configuration, this is a onetime cost for each benchmark sample.

(3) Simulations can then be run in an aggressive fast-forward mode, updating only architected state. At the designated number of instructions prior to each cluster (as determined by either MSE or MRRL), the simulator shifts into warmup mode where cache hierarchy and branch predictor activity are modeled. Once the cluster is reached, the cache(s) and branch predictor will contain accurate state, and cycle-accurate, simulation begins. This last step repeats for each precluster–cluster pair.

Contrast this approach to the more conservative technique of modeling all precluster cache and branch predictor interactions, that is, FULLWARMUP. Obviously, modeling all precluster cache and branch predictor interactions will maintain perfect state throughout all levels of the cache hierarchy and in the branch predictor, rendering the simulation data impervious to inaccuracies that arise from cold-start bias; only sampling error remains. Reciprocally, *STALEWARMUP*—as the name implies—does not model any precluster cache or branch predictor interactions, but merely recycles state as it appeared at the conclusion of the previous cluster. While fast—in fact, the fastest warmup strategy—failing to model any cache and branch predictor state prior to each cluster, makes STALEWARMUP susceptible to cold-start bias, as will be shown in Section 6.

Our strategy is to decide in advance of actual simulation, a reduced amount of warmup that will not sacrifice accuracy. This separate profiling step is a key component of our warmup acceleration strategy. Measuring critical information beforehand saves time by rescuing the simulator from having to make these measurements as it simulates; instead, these data can be passed to the simulator as command-line parameters or within a file read by the simulator. While we did not measure profiling time as precisely as simulation running times, it appears in general, that the time cost of profiling a benchmark is generally less than the time required to simulate the same benchmark using FULLWARMUP. The small, onetime profiling cost is then amortized over the set of simulator configurations a researcher experiments with, resulting in a substantial time saving since, as will be shown in Section 6, MRRL is capable of achieving nearly 95% on average, of the speedup achieved by FULLWARMUP.

## 4. MSE VERSUS MRRL

As stated in Section 1 and demonstrated in Haskins, Jr. [2003], MRRL supersedes MSE completely as a technique for warmup acceleration. The reasons for this include:

(1) *No probability calculations.* MSE determines a reduced subset of precluster instructions to use for warmup by facilitating a probabilistic assessment of cache capacity based on the number of unique references that the cache handles. While we have derived a tractable approximation to the MSE formula, the probability calculations to compute the minimum number of unique references are still time-consuming due to the lack of a closed-form solution

for $m$ which forces us to iterate to the correct value. MRRL makes no such calculations and eliminates this overhead altogether.

(2) *Directly applicable to all levels of the cache hierarchy.* The correctness of the MSE formula is based upon the assumed uniform distribution of unique reference addresses among the $N$ sets of a cache. This spacial uniformity assumption has been rigorously demonstrated [Haskins, Jr. 2003; Haskins, Jr. and Skadron 2001] by using $\chi^2$ analysis of the unique reference distribution pattern to disprove the null hypothesis that unique memory references are not uniformly distributed. However, while this uniformity was easily demonstrated for relatively small (e.g., $N \in \{512,1024\}$) first-level caches, this uniformity assumption may not necessarily hold for larger secondary and tertiary caches. Another, more subtle assumption underlying the correctness of MSE is that it is being applied to a stream containing one type of references: instruction fetches or data loads and stores. Deeper levels of the cache hierarchy tend to be unified, hosting both instructions and data. MRRL imposes none of these assumptions.

(3) *Applicable to warming up branch predictors.* Before discussing MSE's shortcomings, it is important to lay a framework for discussing branch prediction. Hennessy and Patterson [1995] describe the canonical *branch prediction buffer*[3] (BPB) as a tagless cache of $2^n$ saturating counters indexed by $n$ bits from each branch's instruction pointer. Predictions are based therefore, on perbranch local history. This canonical BPB is the basis for all discussions concerning MRRL's usefulness for accelerating branch predictor warmup. Given $p = 99.9\%$ a good direct-mapped estimate for $m$ is $16N$ [Haskins, Jr. 2003]. Hence, for even a modest, say, 1K-entry (e.g., Alpha 21264 [Kessler et al. 1996]) local-history BPB, if $N = 2^{10}$ a user would have to warm up enough precluster instructions to witness $m = 16(2^{10}) =$ 16,384 unique branches. However, since programs tend to spend a considerable amount of their execution in loops, revisiting the same regions of code repeatedly before moving to new regions [Thiébaut 1989], it is possible that any single precluster region may not include 16,384 unique *instructions*, much less unique branches. While this would not cause MSE to yield inaccurate warmup, the inability of the precluster region to fulfill the MSE-prescribed $m$ unique branch instructions will cause MSE to degenerate into FULLWARMUP, completely trading away speed for accuracy. MRRL's acceleration ability does not depend on a dense population of unique control flow instructions, and therefore does not normally degenerate to FULLWARMUP to achieve accurate branch predictor state.

[3]A popular variation on the BPB is a two-level branch prediction buffer (2LBPB) that utilizes the same $2^n$-entry tagless cache of saturating counters, that is instead indexed by an $n$-bit shift register that records the taken–not-taken history of the $n$ most recent branches, thereby rendering predictions based on global branch history. Even though the 2LBPB's global history-based prediction scheme largely divorces branch prediction from perbranch temporal locality, in our experiments we have not noticed any appreciable degradation in accuracy that can be traced to improper branch predictor warmup when a 2LBPB- or hybrid BPB–2LBPB-based predictor is simulated with MSE or MRRL warmup.

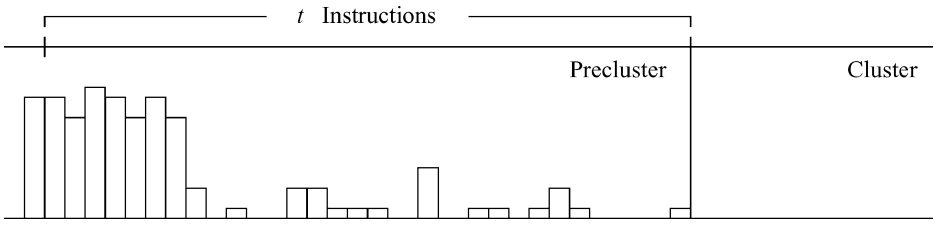


Fig. 3. Front-loaded precluster contains a burst of references to unique addresses very early during the precluster period, followed by a sparse population of uniques; $t$ therefore, encapsulates most of the precluster period.

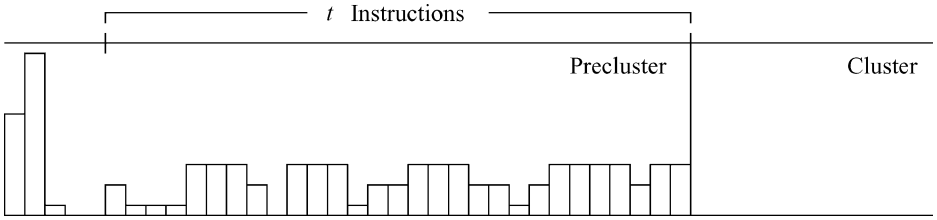


Fig. 4. Flat-loaded precluster contains sparse, but steady appearance of unique reference addresses throughout the precluster period; $t$ therefore, encapsulates most of the precluster period.

(4) *Independence from temporal unique reference distribution.* Equally as important as the quantity of unique references, and their spacial distribution among the cache sets, is their temporal distribution throughout the precluster instruction stream. Consider the case where the overwhelming majority of unique memory references are accessed during an intense burst of start-up activity very early during the precluster instructions. Immediately ensuing is a steady-state period where these references are reaccessed repeatedly. Assuming that there are $m$ unique references among the precluster instructions, this temporally front-loaded distribution of unique references diminishes MSE's ability to accelerate warmup because in order to successfully encapsulate these $m$ unique references, warmup must begin long before the cycle-accurate cluster commences, yielding a high value for $t$; this phenomenon is illustrated in Figure 3. A similar problem occurs when unique references are sparsely distributed during the precluster instruction stream; this scenario is depicted in Figure 4. Thus, while MSE yields accurate warmup regardless of the temporal distribution of unique references, maximum warmup acceleration occurs when unique references are located near the start of the cluster as illustrated in Figure 5. Unlike MSE, MRRL's ability to accelerate warmup does not depend on the temporal distribution of unique references; hence, MRRL naturally avoids complications due to front-loading and flat-loading altogether.

(5) *Does not require $\alpha$ or $\beta$.* The large capacity of second-level and third-level caches may preclude their being completely filled, leaving many cache blocks untouched. To accommodate this, MSE utilizes the variables $\alpha$, $\beta \in (0, 1]$, which specify that only a certain fraction of the sets, and a certain fraction of blocks per set need to be touched since it would be inefficient to

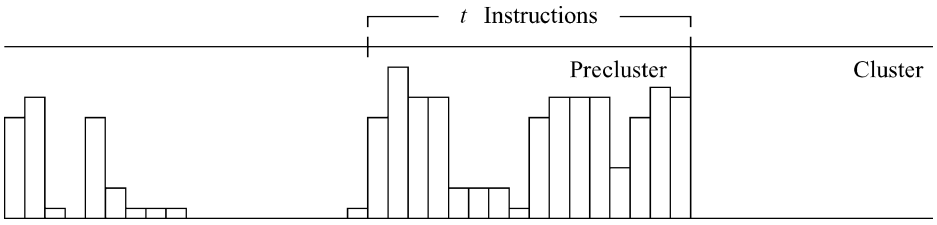


Fig. 5. Back-loaded precluster contains a burst of first references to unique addresses very late during the precluster period; $t$ therefore, requires only a small portion of the precluster period.

determine the MSE-prescribed $m$ unique references to touch all $Na$ cache blocks when only $\alpha N \beta a$ blocks will be touched during the precluster instructions. Unfortunately, we were unable to determine a technique for determining $\alpha$ and $\beta$ *without* running the simulation using FULLWARMUP to make this determination. This forces us to either guess (violating our quest to construct a rigorous, formal warmup technique) or to let $\alpha = \beta = 1$. The former may result in inaccurate warmup; the latter may be overkill which, due to a dearth of unique references, reverts to FULLWARMUP preserving accuracy while failing to accelerate warmup. Thus, while MSE is still a useful tool for reasoning about cache capacity based only on the occurrence of unique references and the dimensions of the cache, its usefulness for yielding actual warmup acceleration in realistic, multilevel cache simulations is limited.

## 5. METHODOLOGY

The data discussed in Section 6 were gathered using random cluster sampling as described by Conte et al. [1996]. Random cluster sampling is attractive because its results are amenable to rigorous statistical analysis; we exploit this to demonstrate quantitatively, the merits of MRRL and its predecessor, MSE.

In each experiment, clusters containing 1 million contiguous instructions apiece were chosen at random from the end-to-end dynamic instruction stream of each benchmark. Conte et al. use clusters of 100,000 instructions apiece, Wunderlich et al. [2003] use clusters of 1000 instruction apiece, Sherwood et al. [2001, 2002] use clusters with instruction counts that are integer multiples of 100 million, and Perelman et al. [2003] show results for clusters of 1-, 10-, and 100 million instructions apiece. Our hypothesis was that medium-sized clusters would cumulatively estimate the true, end-to-end IPC with good accuracy and avail plentiful opportunities for acceleration. For sufficiently large samples, the experimental data show that this hypothesis is correct. A possible avenue for future research would be to experiment with MRRL using random cluster sampling with varying cluster sizes.

To select the clusters, benchmarks were first executed by *sim-fast*—the rapid instruction-level simulator from the SimpleScalar [Austin; Burger and Austin 1997] toolset—to obtain the end-to-end dynamic instruction count, $L$. Next, a simple Perl script was used to select the 1-million-instruction clusters at random from the discrete interval $[1, L]$. The locations of the clusters (as the number of completed instructions relative to the start of execution) were saved

to a file, and subsequently used to drive the multiple cluster profiling and simulation steps enumerated in Section 3.3. For each benchmark, the same set of sample clusters was used to experiment with four warmup techniques: FULLWARMUP, MRRL, MSE, and STALEWARMUP.

Sampling (whether random, systematic, stratified, cluster, or multistage [Henry 1990]) always produces error because only a subset of a population is measured rather than the entire population. Hence, by sampling, one can only *estimate* the characteristics of an entire population. Random cluster sampling allows one to rigorously gauge the amount of error and the probability that the amount is significant, based upon the assumption that all members of the population had uniform probability of being included in the sample. Increasing the size of a sample increases the accuracy of the estimation by reducing the sampling error and bringing the estimation value asymptotically nearer to the true value.[4] A key consideration therefore was to determine the number of clusters to draw from each benchmark. For most benchmarks, 50 clusters were sufficient to estimate[5] the end-to-end IPC (i.e., $IPC_{true}$) when simulated with FULLWARMUP. For the benchmarks *applu* and *galgel*, however, a larger sample had to be drawn to obtain good accuracy; for these, we used samples of 500 clusters.

For the MRRL simulations, the warm phase was engaged $w_N$ instructions prior to each cluster for $N \in \{0.990, 0.999\}$. $N = 0.999$ has shown good performance in mimicking the accuracy of FULLWARMUP [Haskins, Jr. and Skadron 2003]. $N = 0.990$ was chosen to test whether the same performance could be demonstrated for a lower value of $N$. If a lower value for $N$ performs as well as $N = 0.999$ (i.e., does not deviate from FULLWARMUP by a statistically significant amount), then this lower value of $N$ establishes a tighter lower bound on the minimal necessary $N$ to achieve accurate simulation. If not, then the threshold minimal $N$ can be said to exist somewhere in the interval (0.990, 0.999]. We do not experiment with lower values for $N$ since, as will be shown in Section 6, $MRRL_{0.990}$ achieves well over 90% of the maximum potential speedup on average, and deviates from the FULLWARMUP IPC by only 3.88% in the worst case, and less than 1% on average.

Recall that MRRL's profiling step makes MRRL measurements separately for instruction-, data-, and branch addresses. This yields three $w_N$: $w_{N_i}$, $w_{N_d}$, and $w_{N_b}$, respectively. While instruction- and data reference reuse latencies are measured separately, the cache hierarchy we used for our experiments (Table I) specifies a *unified* second-level cache, hosting both instructions and data. Therefore, instruction cache modeling and data cache modeling must be engaged simultaneously, since engaging warmup of one reference stream before the other does not accurately mimic the potentially tumultuous relationship between instructions and data for space in the common L2. This would give the first reference stream an unfair, unrealistic opportunity to become established

[4]In this context "true value" refers to the value that would be obtained by measuring the complete, end-to-end average instruction throughput, $IPC_{true}$.

[5]Our threshold for a good estimate was to deviate from the end-to-end IPC by less than 5% when the sample is simulated with FULLWARMUP.

Table I. Configuration of Simulated Microarchitecture

| Pipeline | |
|---|---|
| Issue Width | 8 instructions/cycle |
| Decode Width | 8 instructions/cycle |
| Register Update Unit | 128 entries |
| Load–Store Queue | 32 entries |
| Commit Width | 8 instructions/cycle |
| **Cache Hierarchy** | |
| L1 Data | 16 KB; 4-way assoc., 32 B lines, 2-cycle hit |
| L1 Instruction | 8 KB; 2-way assoc., 32 B lines, 2-cycle hit |
| L2 Unified | 1 MB; 4-way assoc., 64 B lines, 20-cycle hit |
| Memory Access Latency | 151 cycles |
| **Combined Branch Predictor** | |
| Bimodal | 8192 entries |
| PAg | 8192 entries |
| Return Address Stack | 64 entries |
| Branch Target Buffer | 2048 entries; 4-way assoc. |
| Mispredict Latency | 14 cycles |

in the L2 and may lead to unrepresentative cache state that adversely impacts the respective miss rates of references to instructions or data during the cycle-accurate hot simulation phases. Accordingly, to capture the prescribed warmup duration, instruction cache simulation and data cache simulation begin at the larger of $w_{N_i}$ and $w_{N_d}$ instructions prior to the hot phase of simulation. Branch predictor simulation, on the other hand, is independent of cache state at all levels of the hierarchy and is engaged $w_{N_b}$ instructions prior to the sample cluster, independently of instruction- and data cache simulation.

Haskins, Jr. and Skadron [2001] show that using MSE, all blocks of a simulated direct-mapped cache were touched as predicted with $p = 99.9\%$; progressively more blocks were left untouched after reducing $p$ to 99.0% and 95.0%. Hence, for the MSE experiments, we tested $p \in \{95.0\%, 99.9\%\}$ as the probabilities of accurate warmup to measure the effect on simulation accuracy under random cluster sampling at these two extremes. The other MSE parameters were dictated by the first-level cache, shown in Table I. While the block width of both the L1 data cache and the L1 instruction cache are identical (32 bytes), the data cache has twice the capacity of the instruction cache. Since the data cache is larger, its dimensions guided the MSE calculation thus: $m = 4 \cdot \text{MSEapprox}(N, p) \approx \text{MSE}(128, 4, p)$. Hence, warmup for the MSE experiments was driven entirely by the data cache. MSE is not well suited to warming up very large structures or branch predictors, because the size of the warmup window necessary to achieve probability $p$ of accurate warmup grows rapidly, yielding virtually 0 acceleration over FULLWARMUP [Haskins, Jr. 2003]. Thus, branch prediction defaulted to FULLWARMUP for these experiments, assuring accurate warmup of the branch predictor.

Once each benchmark's sample was selected, the next step was to profile to gather MSE/MRRL data for each benchmark. A Perl script was then used to extract the MSE $t$ and MRRL $w_N$ for each benchmark's precluster–cluster pairs. When fed to the multiple cluster simulator, these data were used to demarcate the boundary between the precluster cold phase and warm phase; the previously

chosen hot phases (clusters) remained fixed just as they were during the profile.

The metrics used to measure MSE's and MRRL's merit are percent-error IPC deviation from FULLWARMUP, accuracy with respect to the true IPC, statistical significance of deviation by matched-pairs $t$-test, running time as a percentage of FULLWARMUP, and percentage achieved of the maximum potential speedup.

All benchmarks come from the SPEC CPU2000 suite [Standard Performance Evaluation Corporation 1999]; the binaries were compiled into the Alpha AXP instruction set and statically linked so that the simulations see all user-space program behavior, including library routines. The MRRL profiler and the multiple cluster simulator were adapted from *sim-safe* and *sim-outorder*, respectively, from SimpleScalar. To measure simulation time data as accurately as possible, *sim-outorder* was further modified to use the UNIX system call $getrusage()$ to monitor the CPU time of each simulation regardless of other activity on the host system. (All the scripts and software developed for this research are available for download from the MRRL Web site which is linked to from the Web site for the Laboratory for Computer Architecture at Virginia (LAVA) at http://lava.cs.virginia.edu/.)

## 6. EVALUATION

Before developing a more formal framework for MSE and MRRL accuracy analysis, it is important to define the components of error and their relationship to microprocessor simulation. Henry [1990] separates error into two components: sampling and nonsampling. Sampling error is an unavoidable consequence of the fact that a sample can only approximately capture characteristics of an entire population. Nonsampling error arises from a failure to ensure the representativeness of the environment in which the sample measurements are taken. In other words, if the environment of the sample does not at least approximate the environment of the population, the measurements taken by sampling will tend to be skewed. In a microprocessor pipeline, the state of the cache and the branch predictor heavily influence instruction throughput [Hennessy and Patterson 1995] and their state constitutes a major component of the instruction stream execution environment. Failure to accurately initialize state within the simulated cache and branch predictor may adversely affect measurements taken during cycle-accurate simulation of the sample clusters; this is the cold-start effect. By modeling all precluster cache and branch predictor interactions, however FULLWARMUP simulation is impervious to nonsampling error; hence, the chief objective for each of MSE and MRRL was to develop a warmup strategy that accelerates warmup without adding additional nonsampling error (i.e., to develop a warmup strategy whose estimated IPC does not deviate from the estimated IPC generated by FULLWARMUP by a statistically significant amount).

We begin by intuitively demonstrating (1) that MSE and MRRL reliably approximate FULLWARMUP, and (2) the importance of adequate warmup. Since, by definition, FULLWARMUP is invulnerable against nonsampling error, if the estimated IPCs generated by MSE and MRRL deviate only slightly from IPCs

generated by FULLWARMUP, then MSE and MRRL add negligible additional nonsampling error. On the other hand, STALEWARMUP, which merely recycles cache and branch predictor state between sample clusters, will be shown to be capable of increasing nonsampling error substantially.

Recall that our research objective is not to describe techniques for choosing good samples, but to compare warmup strategies that are capable of accelerating warmup while maintaining the accuracy of simulation results. We chose to employ random cluster sampling for ease of use, precedent [Conte et al. 1996; Haskins, Jr. 2003; Haskins, Jr. and Skadron 2003], and amenability to rigorous statistical analysis. Additionally, random cluster sampling is immune to systematic error [Taylor 1982] which may arise, for example, if uniform separation of sample clusters—the distinguishing characteristic of systematic sampling—overlays some periodic behavior embedded within the dynamic instruction stream. If the periodic behavior captured is exclusive of other behaviors, then the simulation results will be skewed accordingly, biasing the outcome. On the other hand, if the variance within the systematic sample is greater than the variance of the entire population (i.e., among *all* possible clusters within the end-to-end instruction stream), then the systematic sample will be more precise than a random sample of clusters [Cochran 1977]. It would be unduly cumbersome, however, to attempt to measure the population-wide variance of instruction throughput measurements to prove this superior accuracy if it exists for the chosen sample. While previous research indicates that systematic error is a relatively minor concern for accurate microarchitecture simulation [Wunderlich et al. 2003] and trace-driven evaluation of memory systems [Crowley and Baer 1999], random cluster sampling is a conservative alternative that freed us from concerns over systematic error as we performed our statistical analysis.

As a target accuracy we decided that the sampled results generated by FULLWARMUP (i.e., $IPC_{FULLWARMUP}$) should deviate from the true, end-to-end mean IPC (i.e., $IPC_{true}$) by less than 5%. Except for two benchmarks, *applu* and *galgel*, this was easily accomplished by simulating a randomly selected set of 50 1-million-instruction cycle-accurate sample clusters. A sample of 500 1-million-instruction cycle-accurate sample clusters was necessary for *facerec* and *galgel* to conform to this threshold.

As a first step, we begin by validating random cluster sampling under FULLWARMUP. This validation is critical because it justifies our comparisons of MSE, MRRL, and STALEWARMUP to FULLWARMUP. As Table II shows, for suitably chosen samples, FULLWARMUP does well at eliminating nonsampling error due to cold-start bias by approximating the true end-to-end IPC[6] with less

[6]Most of these IPCs come from the *SimPoint* [Sherwood et al. ] Web site; this was a very substantial benefit to our research, sparing us the irony of performing expensive end-to-end simulations to demonstrate the benefits of warmup techniques tasked with obviating precisely this. Each of the *SimPoint* IPCs were generated for a specific configuration of *sim-outorder* (linked to from the site). MSE, MRRL, FULLWARMUP, and STALEWARMUP experiments compared against these IPCs use the same *sim-outorder* configuration and the same benchmark binaries. The few other true IPCs were gathered by brute-force simulation on our own compute servers, also using the same *SimPoint* configuration file.

Table II. $IPC_{FULLWARMUP}$ Percent-Error Relative to $IPC_{true}$ ($100\% \cdot \frac{IPC_{FULLWARMUP} - IPC_{true}}{IPC_{true}}$); and $IPC_{STALEWARMUP}$ %-error relative to $IPC_{FULLWARMUP}$ ($100\% \cdot \frac{IPC_{STALEWARMUP} - IPC_{FULLWARMUP}}{IPC_{FULLWARMUP}}$). MEAN Calculations Based on the Absolute Value of Errors

| Benchmark | $IPC_{true}$ | $IPC_{FULLWARMUP}$ (%) | $IPC_{STALEWARMUP}$ (%) |
|---|---|---|---|
| applu | 0.831 | −0.36 | −1.09 |
| apsi | 1.008 | 3.12 | −2.23 |
| art_110 | 0.598 | −0.57 | 0.34 |
| crafty | 0.569 | −3.64 | −0.80 |
| equake | 0.310 | 0.42 | 2.22 |
| facerec | 1.446 | −4.87 | −10.46 |
| fma3d | 0.535 | −0.37 | 1.57 |
| galgel | 1.334 | −0.60 | −11.61 |
| gcc_integrate | 1.431 | −1.86 | −7.26 |
| gzip_graphic | 1.365 | −3.28 | −0.52 |
| lucas | 0.774 | 2.25 | 0.23 |
| mcf | 0.092 | 3.04 | 0.84 |
| mgrid | 0.987 | 4.72 | −1.87 |
| twolf | 0.636 | −1.08 | −1.76 |
| vortex_lendian2 | 1.057 | −3.18 | −0.63 |
| vpr_route | 1.023 | 0.18 | −1.16 |
| Mean | | 2.10 | 2.79 |

than 5% deviation in all cases. Table II furthermore compares STALEWARMUP IPC (i.e., $IPC_{STALEWARMUP}$) percent-error deviation relative to FULLWARMUP IPC. That is, Table II first compares the end-to-end mean $IPC_{true}$ to the sample mean $IPC_{FULLWARMUP}$, and compares $IPC_{FULLWARMUP}$ to the sample mean $IPC_{STALEWARMUP}$. This positions FULLWARMUP as the warmup gold standard: since, for enough sample clusters, FULLWARMUP is demonstrably able to achieve our target deviation of less than 5% relative to $IPC_{true}$, the ability of all the other warmup techniques (STALEWARMUP, MSE, MRRL) to defeat nonsampling error will be judged by their mean IPCs' relative error to the sample mean IPC generated by FULLWARMUP.

Notice the STALEWARMUP percent-error deviation from $IPC_{FULLWARMUP}$ for the benchmarks *facerec* and *galgel* of −10.46% and −11.61%, respectively. These deviations—substantially larger than those of the other benchmarks—are qualitative evidence that inadequate warmup can compromise simulation accuracy by failing to mitigate nonsampling error; although this only happens for two of the benchmarks, it evidences the devastating impact of rampant nonsampling error resulting from a failure to establish accurate cache and branch predictor state.

Having intuitively established the unreliability of STALEWARMUP, we sketch first-pass evidence of MSE's and MRRL's reliability in Section 6.1; more rigorous, quantitative evidence of the reliability of MSE and MRRL, and the unreliability of STALEWARMUP will be given in Sections 6.2 and 6.3. In Section 6.4, we will show that although MSE and MRRL are roughly identically successful at reducing nonsampling error (both deviating from FULLWARMUP by less than 1% on average), MRRL has a large speedup advantage over MSE.

Table III. IPC %-Error Relative to $IPC_{FULLWARMUP}$ ($100\% \cdot \frac{IPC_X - IPC_{FULLWARMUP}}{IPC_{FULLWARMUP}}$). MEAN Calculations Based on the Absolute Value of Errors

| Benchmark | $IPC_{FULLWARMUP}$ | $IPC_{MRRL_{0.999}}$ (%) | $IPC_{MRRL_{0.990}}$ (%) | $IPC_{MSE_{99.9\%}}$ (%) | $IPC_{MSE_{95.0\%}}$ (%) |
|---|---|---|---|---|---|
| applu | 0.828 | 0.01 | 0.29 | 0.00 | 0.00 |
| apsi | 1.039 | −0.01 | −0.04 | 0.03 | 0.03 |
| art_110 | 0.595 | 0.00 | 0.00 | 0.00 | 0.00 |
| crafty | 0.548 | −0.02 | −0.04 | 0.09 | 0.09 |
| equake | 0.311 | 0.00 | 0.00 | 0.00 | 0.00 |
| facerec | 1.376 | 0.18 | 0.36 | 0.63 | 0.98 |
| fma3d | 0.533 | 3.90 | 3.88 | 3.94 | 3.94 |
| galgel | 1.326 | −0.20 | −0.62 | 0.02 | 0.02 |
| gcc_integrate | 1.404 | 0.13 | −0.42 | −0.15 | 0.00 |
| gzip_graphic | 1.320 | −0.09 | −0.01 | 0.01 | 0.01 |
| lucas | 0.791 | −0.04 | −0.13 | −0.28 | −0.28 |
| mcf | 0.095 | 0.00 | 0.00 | 0.00 | 0.00 |
| mgrid | 1.034 | −0.01 | −0.01 | −0.01 | −0.09 |
| twolf | 0.629 | 0.13 | 0.14 | 0.19 | 0.19 |
| vortex_lendian2 | 1.023 | 0.06 | 0.07 | 0.31 | 0.13 |
| vpr_route | 1.025 | 0.00 | 0.00 | 0.00 | 0.00 |
| Mean | | 0.55 | 0.38 | 0.35 | 0.36 |

### 6.1 IPC Accuracy Compared to $IPC_{FULLWARMUP}$

Table II qualitatively demonstrates that inadequate warmup ($IPC_{STALEWARMUP}$) generated substantial additional nonsampling error for *facerec* and *galgel*. Table III, on the other hand, indicates that MSE and MRRL do not generate a large amount of nonsampling error. Table III lists the FULLWARMUP IPCs, and percent-error deviations therefrom for MRRL at $N = 0.999$ and $N = 0.990$ (i.e., $IPC_{MRRL_{0.999}}$ and $IPC_{MRRL_{0.990}}$), and for MSE at $p = 99.9\%$ and $p = 95.0\%$ (i.e., $IPC_{MSE_{99.9\%}}$ and $IPC_{MSE_{95.0\%}}$). For all benchmarks except *fma3d*, the magnitude of the percent difference deviation from FULLWARMUP is much less than 1%. *fma3d*'s seemingly drastic nonconformance, however, is due to the small numbers involved in the percent-error calculation. Take for example the largest deviation of 3.94%, due to MSE at $p = 95.0\%$ and $p = 99.9\%$; $IPC_{FULLWARMUP} = 0.533$, $IPC_{MSE_{95.0\%,99.9\%}} = 0.554$. The relative error, $100\% \cdot \left(\frac{0.554-0.533}{0.533}\right) = 3.94\%$ makes the deviation look much worse than it really is when one considers that the absolute error is so small: 0.554–0.533 = 0.021, or 21 thousands of an instruction per cycle. This assertion is further corroborated by Table IV, which shows that *fma3d* falls within the 95% confidence interval in all four cases.

### 6.2 IPC Accuracy Compared to $IPC_{true}$

While $MRRL_{0.999}$, $MRRL_{0.990}$, $MSE_{99.9\%}$, and $MSE_{95.0\%}$ are apparently sound warmup strategies, and STALEWARMUP apparently unsound, we will now rigorously demonstrate these hypotheses. For each benchmark, the mean instruction throughput was measured by counting the number of cycles consumed in executing the sample clusters. Dividing the total number of executed instructions by this amount yielded the overall sample IPC (i.e., $\overline{IPC}$). For a well-chosen sample, this sample IPC will be a good estimate of the end-to-end IPC. The

Table IV. IPC 95% Confidence Intervals Centered Around $\overline{\text{IPC}}$ (the Overall Sample IPC), for $\text{MRRL}_{0.999}$, $\text{MRRL}_{0.990}$, $\text{MSE}_{99.9\%}$, and $\text{MSE}_{95.0\%}$. Bold Entries Fail to Accurately Predict the Amount of Sampling Error

| Benchmark | $\text{IPC}_{\text{true}}$ | $\text{IPC}_{\text{MRRL}_{0.999}}$ | $\text{IPC}_{\text{MRRL}_{0.990}}$ | $\text{IPC}_{\text{MSE}_{99.9\%}}$ | $\text{IPC}_{\text{MSE}_{95.0\%}}$ |
|---|---|---|---|---|---|
| applu | 0.831 | $0.829 \pm 0.053$ | $0.831 \pm 0.053$ | $0.828 \pm 0.053$ | $0.828 \pm 0.053$ |
| apsi | 1.008 | $1.039 \pm 0.063$ | $1.039 \pm 0.064$ | $1.040 \pm 0.064$ | $1.040 \pm 0.064$ |
| art_110 | 0.597 | $0.595 \pm 0.029$ | $0.595 \pm 0.029$ | $0.595 \pm 0.029$ | $0.595 \pm 0.029$ |
| crafty | 0.569 | $\mathbf{0.548 \pm 0.014}$ | $\mathbf{0.548 \pm 0.014}$ | $\mathbf{0.549 \pm 0.014}$ | $\mathbf{0.549 \pm 0.014}$ |
| equake | 0.310 | $0.311 \pm 0.104$ | $0.311 \pm 0.104$ | $0.311 \pm 0.104$ | $0.311 \pm 0.104$ |
| facerec | 1.446 | $1.378 \pm 0.460$ | $1.381 \pm 0.460$ | $1.384 \pm 0.458$ | $1.389 \pm 0.458$ |
| fma3d | 0.535 | $0.554 \pm 0.058$ | $0.554 \pm 0.058$ | $0.554 \pm 0.061$ | $0.554 \pm 0.058$ |
| galgel | 1.334 | $1.323 \pm 0.112$ | $1.317 \pm 0.112$ | $1.326 \pm 0.112$ | $1.326 \pm 0.112$ |
| gcc_integrate | 1.431 | $1.406 \pm 0.159$ | $1.399 \pm 0.159$ | $1.402 \pm 0.159$ | $1.404 \pm 0.159$ |
| gzip_graphic | 1.365 | $1.319 \pm 0.094$ | $1.320 \pm 0.094$ | $1.320 \pm 0.094$ | $1.320 \pm 0.094$ |
| lucas | 0.774 | $0.791 \pm 0.157$ | $0.790 \pm 0.156$ | $0.789 \pm 0.157$ | $0.789 \pm 0.157$ |
| mcf | 0.092 | $0.095 \pm 0.052$ | $0.095 \pm 0.052$ | $0.095 \pm 0.052$ | $0.095 \pm 0.052$ |
| mgrid | 0.987 | $1.034 \pm 0.106$ | $1.034 \pm 0.106$ | $1.034 \pm 0.106$ | $1.033 \pm 0.106$ |
| twolf | 0.636 | $\mathbf{0.629 \pm 0.004}$ | $\mathbf{0.630 \pm 0.004}$ | $\mathbf{0.630 \pm 0.004}$ | $\mathbf{0.630 \pm 0.004}$ |
| vortex_lendian2 | 1.057 | $1.024 \pm 0.040$ | $1.024 \pm 0.040$ | $1.027 \pm 0.040$ | $1.025 \pm 0.040$ |
| vpr_route | 1.023 | $1.025 \pm 0.038$ | $1.025 \pm 0.038$ | $1.025 \pm 0.038$ | $1.025 \pm 0.038$ |

*standard error* is a useful tool to analyze the goodness of a sample estimate [Freund 1971; Sternstein 1996]. The standard error is computed as the quotient of the percluster sample standard deviation in IPC and the square root of the number of clusters:

$$s_{\overline{\text{IPC}}} = \frac{\sigma}{\sqrt{\#_{\text{cluster}}}}$$

We assume that error is normally distributed[7] [Conte et al. 1996; Kendall and Stuart 1968]; hence, the 95% confidence interval is $\overline{\text{IPC}} \pm 1.96 s_{\overline{\text{IPC}}}$. In other words, for a well-chosen sample, one can assume $\text{IPC}_{\text{true}} \in [\overline{\text{IPC}} - s_{\overline{\text{IPC}}}, \overline{\text{IPC}} + s_{\overline{\text{IPC}}}]$ with 95% certainty.

Furthermore, let $e = |\text{IPC}_{\text{true}} - \overline{\text{IPC}}|$; if $[\text{IPC}_{\text{true}} - e, \text{IPC}_{\text{true}} + e] \subset [\overline{\text{IPC}} - 1.96 s_{\overline{\text{IPC}}}, \overline{\text{IPC}} + 1.96 s_{\overline{\text{IPC}}}]$, then the relative error between $\text{IPC}_{\text{true}}$ and $\overline{\text{IPC}}$ was accurately predicted by the 95% confidence interval. Table IV shows that the relative error between $\text{IPC}_{\text{MRRL}_{0.999}}$ and $\text{IPC}_{\text{true}}$, $\text{IPC}_{\text{MRRL}_{0.990}}$ and $\text{IPC}_{\text{true}}$, $\text{IPC}_{\text{MSE}_{99.9\%}}$ and $\text{IPC}_{\text{true}}$, and $\text{IPC}_{\text{MSE}_{95.0\%}}$ and $\text{IPC}_{\text{true}}$ was predicted by every benchmark's respective 95% confidence intervals except for *crafty*, and *twolf* (in bold typeface). Table V shows, however, that the 95% confidence interval failed to predict the relative error between $\text{IPC}_{\text{FULLWARMUP}}$ and $\text{IPC}_{\text{true}}$ for these same two benchmarks! Since FULLWARMUP models all precluster cache and branch predictor interactions, it impervious to nonsampling error; hence, its failure to predict the relative error for these benchmarks is attributable to sampling error. Perfectly mimicking FULLWARMUP in this way is further evidence that MRRL at $N = 0.999$ and $N = 0.990$, and MSE at $p = 99.9\%$ and $p = 95.0\%$ do well at approximating FULLWARMUP. In other words, $\text{MRRL}_{0.999}$, $\text{MRRL}_{0.990}$, $\text{MSE}_{99.9\%}$, and $\text{MSE}_{95.0\%}$ do well at eliminating nonsampling error. Notice also,

[7]The assumption of normality is safe since the samples contain 50 clusters apiece. Samples of 30 or fewer elements would use the Student's $t$-distribution [Sternstein 1996] with $\#_{\text{cluster}} - 1$ degrees of freedom.

Table V. 95% IPC Confidence Intervals Centered Around $\overline{\text{IPC}}$ (the Overall Sample IPC), for FULLWARMUP, and STALEWARMUP. Successful Simulations Contain $\text{IPC}_{\text{true}}$ within Their Confidence Interval. Bold Entries Fail to Predict the Amount of Sampling Error

| Benchmark | $\text{IPC}_{\text{true}}$ | $\text{IPC}_{\text{FULLWARMUP}}$ | $\text{IPC}_{\text{STALEWARMUP}}$ |
|---|---|---|---|
| applu | 0.831 | $0.828 \pm 0.053$ | $0.819 \pm 0.053$ |
| apsi | 1.008 | $1.039 \pm 0.063$ | $1.039 \pm 0.064$ |
| art_110 | 0.597 | $0.595 \pm 0.029$ | $0.597 \pm 0.029$ |
| crafty | 0.569 | $\mathbf{0.548 \pm 0.014}$ | $\mathbf{0.544 \pm 0.014}$ |
| equake | 0.310 | $0.311 \pm 0.104$ | $0.318 \pm 0.110$ |
| facerec | 1.446 | $1.376 \pm 0.460$ | $\mathbf{1.232 \pm 0.135}$ |
| fma3d | 0.535 | $0.533 \pm 0.061$ | $0.542 \pm 0.055$ |
| galgel | 1.334 | $1.326 \pm 0.112$ | $\mathbf{1.172 \pm 0.104}$ |
| gcc_integrate | 1.431 | $1.404 \pm 0.159$ | $1.302 \pm 0.142$ |
| gzip_graphic | 1.365 | $1.320 \pm 0.094$ | $1.313 \pm 0.094$ |
| lucas | 0.774 | $0.791 \pm 0.157$ | $0.793 \pm 0.144$ |
| mcf | 0.092 | $0.095 \pm 0.052$ | $0.096 \pm 0.050$ |
| mgrid | 0.987 | $1.034 \pm 0.106$ | $1.014 \pm 0.080$ |
| twolf | 0.636 | $\mathbf{0.629 \pm 0.004}$ | $\mathbf{0.618 \pm 0.009}$ |
| vortex_lendian2 | 1.057 | $1.023 \pm 0.040$ | $1.017 \pm 0.040$ |
| vpr_route | 1.023 | $1.025 \pm 0.038$ | $1.013 \pm 0.036$ |

as alluded to in Section 6.1, that the 95% confidence interval for *fma3d* does bound its true IPC, further supporting our claim that $\text{MRRL}_{0.999}$, $\text{MRRL}_{0.990}$, $\text{MSE}_{99.9\%}$, and $\text{MSE}_{95.0\%}$ impose negligible error in spite of their relatively large, nearly 4% deviation from $\text{IPC}_{\text{FULLWARMUP}}$ for *fma3d*.

In contrast, consider the $\text{IPC}_{\text{STALEWARMUP}}$ sample means of *facerec* and *galgel*. As Table V shows, the STALEWARMUP result does not successfully predict their relative error deviation from $\text{IPC}_{\text{true}}$. This evidence rigorously and quantitatively confirms the hypothesis that their respective $-10.46\%$ and $-11.61\%$ percent-error deviations from the $\text{IPC}_{\text{FULLWARMUP}}$ sample means are statistically significant.

### 6.3 IPC Accuracy According to Matched-Pairs $t$-Test

Statistical hypothesis testing can also be used to demonstrate the significance of the difference between IPC generated by FULLWARMUP and IPC generated by another warmup technique. In this way, hypothesis testing builds a case against the *null hypothesis*, which for our research is that simulations using FULLWARMUP yield a different mean IPC from simulations using $\text{MRRL}_{0.999}$, $\text{MRRL}_{0.990}$, $\text{MSE}_{99.9\%}$, $\text{MSE}_{95.0\%}$, and STALEWARMUP. (In our experiments this hypothesis was usually rejected for MRRL and MSE, but rarely for STALEWARMUP, providing further rigorous evidence in support of the necessity of adequate warmup prior to cycle-accurate simulation.) In particular, we used the matched-pairs $t$-test to compare each benchmark's FULLWARMUP percluster IPCs against the percluster IPCs generated by the other warmup techniques. In this test, the IPC of the $i$th FULLWARMUP cluster is paired with its counterpart $i$th, $\text{MRRL}_N$ or $\text{MSE}_p$ cluster IPC. From this set of pairs, the set of cluster IPC differences is calculated and used to compute a $t$-score based on the difference of the means, the standard error of the means, and their Pearson product-moment correlation coefficient [Underwood et al. 1954].

If, for example, one wishes to compute a $t$-score for the matched-pairs difference between FULLWARMUP and $\text{MRRL}_N$, $t$ is computed, thus:

$$t = \frac{\mu_X - \mu_Y}{\sqrt{\sigma_X^2 + \sigma_Y^2 - 2r_{XY}\sigma_X\sigma_Y}}$$

where $\mu_X - \mu_Y$ is the difference of the FULLWARMUP and $\text{MRRL}_N$ cluster means $\sigma_X$ and $\sigma_Y$ are the standard errors among the FULLWARMUP and $\text{MRRL}_N$ cluster IPCs,[8] and $r_{XY}$ is the Pearson product-moment correlation coefficient between the FULLWARMUP and $\text{MRRL}_N$ cluster IPCs. When computing $t$-scores for *matched samples*, it is necessary to compensate for the likely positive correlation that exists between the paired elements, since the difference between the means of a pair of positively correlated matched samples tends to be lower than the difference between the means of randomly paired (i.e., uncorrelated) samples. (In fact, when $r_{XY} = 0$, the denominator of the matched-pairs $t$-score formula degenerates to the denominator of the formula for computing the $t$-score of randomly paired samples.) The higher[9] the $r_{XY}$, the higher the $t$-score. In essence, the Pearson product-moment correlation coefficient protects against understating the case that a statistically significant difference exists between paired samples that are positively correlated.

Our experiments measure the effects of each warmup strategy as different "treatments" of the same sample population [Underwood et al. 1954], which obviously demonstrates a very high correlation between the paired sample elements from one warmup technique to the next. This process was used to compare FULLWARMUP to $\text{MRRL}_{0.999}$, $\text{MRRL}_{0.990}$, $\text{MSE}_{99.9\%}$, $\text{MSE}_{95.0\%}$, and STALEWARMUP.

At the 5% level of significance, the critical $t$-score[10] for 50-cluster samples is 2.0096, and the critical $t$-score for the 500-cluster samples (*applu* and *galgel*) is 1.9647. We converted the $t$-score generated by each experiment into its equivalent $p$-value which is a probabilistic measurement of the likelihood of obtaining the given $t$-score. Table VI lists the $p$-values of the benchmarks whose $t$-scores were calculated by pairing the cluster IPCs of $\text{MRRL}_{0.999}$, $\text{MRRL}_{0.990}$, $\text{MSE}_{99.9\%}$, $\text{MSE}_{95.0\%}$, and STALEWARMUP with the FULLWARMUP cluster IPCs. A $p$-value greater than 5% indicates an outcome that rejects the null hypothesis. Notice that STALEWARMUP rejects the null hypothesis for only five benchmarks (*equake*, *fma3d*, *lucas*, *mcf*, and *mgrid*). Among a preponderance of these experiments, however, there is a statistically significant difference at the 5% level which is to say that there is insufficient evidence to reject the null hypothesis that IPC measured via STALEWARMUP differs from IPC measured via FULLWARMUP. We conclude the same to be true of STALEWARMUP in general. This, combined with the large (and statistically proven significant

[8]For the matched pairs $t$-test, $\mu_X$ and $\mu_Y$ are computed differently from the sample IPC ($\overline{\text{IPC}}$) mentioned in Section 6.2; rather, they are computed as the mean of individual cluster IPCs. $\sigma_X$ and $\sigma_Y$ are computed using $\mu_X$ and $\mu_Y$, respectively, and are therefore also different from the $\sigma$ used in Section 6.2.

[9]Reciprocally, for lower, negative values of $r_{XY}$, it does not even make sense to use matched samples since there is little similarity between the elements in each sample pair.

[10]According to the Student's $t$-distribution for 49 degrees of freedom and 499 degrees of freedom.

Table VI. Matched-Pairs $t$-test $p$-Values Measuring the Statistical Significance of Cluster Differences between FULLWARMUP and, $MRRL_{0.999}$, $MRRL_{0.990}$, $MSE_{99.9\%}$, $MSE_{95.0\%}$ and STALEWARMUP

| | $p$-Value | | | | |
|---|---|---|---|---|---|
| Benchmark | $MRRL_{0.999}$ (%) | $MRRL_{0.990}$ (%) | $MSE_{99.9\%}$ (%) | $MSE_{95.0\%}$ (%) | STALEWARMUP(%) |
| applu | 36.22 | 15.73 | 11.31 | 38.57 | **0.00** |
| apsi | 36.96 | 49.40 | 45.73 | 45.54 | **0.64** |
| art_110 | 30.01 | 100.0 | **0.68** | **0.68** | **0.01** |
| crafty | 17.41 | **3.01** | **0.00** | **0.00** | **0.00** |
| equake | 11.70 | 53.25 | 26.10 | 56.77 | 17.89 |
| facerec | 17.18 | 8.93 | 6.80 | 8.84 | **0.02** |
| fma3d | 22.03 | 22.67 | 20.69 | 20.72 | 45.85 |
| galgel | 55.80 | 26.09 | 5.67 | 6.51 | **0.00** |
| gcc_integrate | 57.20 | **4.73** | 38.53 | 37.63 | **0.04** |
| gzip_graphic | 5.57 | 19.13 | 12.40 | 12.40 | **4.36** |
| lucas | 26.84 | 32.36 | 85.88 | 86.02 | 52.16 |
| mcf | 55.39 | 26.10 | 7.79 | 17.25 | 15.01 |
| mgrid | 18.58 | 20.05 | 27.00 | 67.20 | 9.08 |
| twolf | **0.01** | **0.00** | **0.00** | **0.00** | **3.36** |
| vortex_lendian2 | **0.00** | **0.00** | **0.00** | **0.00** | **0.16** |
| vpr_route | 35.13 | 35.13 | 35.13 | 35.13 | **0.00** |

At the 5% level of significance, 11 of 16 benchmarks' $t$-scores showed insufficient evidence to reject the null hypothesis which states that IPC as measured by FULLWARMUP differs from IPC as measured by STALEWARMUP. For a strong majority of the benchmarks that were warmed up by the other warmup strategies, on the other hand, sufficient evidence does exist to reject this hypothesis.
Entries in bold typeface fail to reject the null hypothesis at the 5% level of significance.

[Section 6.2, Table V]) deviations from FULLWARMUP for the benchmarks *facerec* and *galgel* are firm evidence that adequate warmup is essential if sampled simulation results are to be trustworthy.

Recall *fma3d*'s relatively large percent-error deviation from FULLWARMUP for $MRRL_{0.999}$, $MRRL_{0.990}$, $MSE_{99.9\%}$ and $MSE_{95.0\%}$ (Section 6.1, Table III). We qualitatively drew the conclusion that although the percent-error deviation was 3.94%, because the absolute error was so small—0.021 instructions per cycle—the deviation was insignificant. Table VI quantitatively confirms this conclusion since the *fma3d* $t$-scores are less than the critical $t$-score for $MRRL_{0.999}$, $MRRL_{0.990}$, $MSE_{99.9\%}$, and $MSE_{95.0\%}$. In other words, the differences between the IPCs generated by these techniques and FULLWARMUP are statistically insignificant at the 5% level. $MRRL_{0.999}$, $MRRL_{0.990}$, $MSE_{99.9\%}$, and $MSE_{95.0\%}$ reject the null hypothesis for *twolf*, and *vortex_lendian2*; $MRRL_{0.990}$, $MSE_{99.9\%}$, and $MSE_{95.0\%}$ additionally reject the null hypothesis for *crafty*; and $MSE_{99.9\%}$ and $MSE_{95.0\%}$ reject the null hypothesis for *art_110*. While initially alarming, further inspection reveals that although their $t$-scores imply statistically significant deviation at the 5% level, the absolute differences $(IPC_{FULLWARMUP} - IPC_{MRRL_{\{0.990,\,0.999\}},MSE_{\{95.0\%,\,99.9\%\}}})$ for *twolf* and *vortex_lendian2* are all 0.001. That is, the $MRRL_{0.999}$, $MRRL_{0.990}$, $MSE_{99.9\%}$, and $MSE_{95.0\%}$ estimates of the FULLWARMUP IPC for the benchmarks *twolf* and *vortex_lendian2* are off by 1 one-thousandth of an instruction per cycle—an amount that we safely, albeit qualitatively assume to be insignificant. The absolute error between $MRRL_{0.990}$, $MSE_{99.9\%}$ and $MSE_{95.0\%}$ and FULLWARMUP for *crafty*, and

Table VII. Random Cluster Sampling Acceleration Relative to FULLWARMUP as a Percentage: $100\% \cdot \frac{t}{t_{\text{FULLWARMUP}}}$

| Benchmark | $t_{\text{FULLWARMUP}}$ (s) | $\%t_{\text{MRRL}_{0.999}}$ (%) | $\%t_{\text{MRRL}_{0.990}}$ (%) | $\%t_{\text{MSE}_{99.9\%}}$ (%) | $\%t_{\text{MSE}_{95.0\%}}$ (%) |
|---|---|---|---|---|---|
| applu | 80,887 | 64.30 | 63.79 | 100.0 | 100.0 |
| apsi | 120,925 | 59.56 | 59.46 | 97.19 | 96.88 |
| art_110 | 19,613 | 36.94 | 37.22 | 98.82 | 98.91 |
| crafty | 78,906 | 48.20 | 48.71 | 87.01 | 86.24 |
| equake | 54,675 | 57.11 | 57.80 | 91.25 | 91.87 |
| facerec | 70,587 | 51.98 | 51.16 | 95.40 | 95.64 |
| fma3d | 96,462 | 61.38 | 61.25 | 96.66 | 96.47 |
| galgel | 162,606 | 55.45 | 55.70 | 95.69 | 96.27 |
| gcc_integrate | 5569 | 75.83 | 68.56 | 100.0 | 100.0 |
| gzip_graphic | 26,643 | 34.58 | 34.89 | 97.81 | 97.39 |
| lucas | 46,730 | 50.67 | 50.79 | 100.0 | 99.21 |
| mcf | 36,014 | 46.45 | 46.79 | 89.09 | 89.17 |
| mgrid | 142,334 | 60.50 | 60.41 | 100.0 | 100.0 |
| twolf | 133,069 | 44.04 | 44.35 | 98.91 | 98.82 |
| vortex_lendian2 | 64,839 | 38.06 | 37.83 | 96.72 | 98.78 |
| vpr_route | 28,358 | 39.75 | 39.79 | 98.63 | 98.27 |
| Mean | | 51.55 | 51.16 | 98.03 | 96.70 |

between $\text{MSE}_{99.9\%}$ and $\text{MSE}_{95.0\%}$ and FULLWARMUP is (to three decimal places) 0.000—again, a qualitatively insignificant amount.

While the raw statistical evidence provided by the $t$-tests implies that $\text{MRRL}_{0.990}$, $\text{MSE}_{95.0\%}$, and $\text{MSE}_{99.9\%}$ are less reliable than $\text{MRRL}_{0.999}$ (by exceeding the critical $p$-value for two extra benchmarks), putting these results into perspective, vis-á-vis their absolute difference deviation from the FULLWARMUP IPC shows that all perform about as well as $\text{MRRL}_{0.999}$ at eliminating nonsampling error.

### 6.4 Simulation Acceleration

Table VII shows the speedup results relative to FULLWARMUP for the random cluster sampling experiments. (All timing data for MSE and MRRL experiments represent the sum of the time spent in all three phases of simulation: cold, warm, and hot. FULLWARMUP timing data, on the other hand, by definition represent the sum of the time spent in warm and hot simulation exclusively; STALEWARMUP timing data used to compute the quantities in Table VIII, by definition represent the sum of the time spent in cold and hot simulation exclusively.) Notice the high percentage of FULLWARMUP running times required by benchmarks for $\text{MSE}_{99.9\%}$ and $\text{MSE}_{95.0\%}$, relative to the more drastic reductions for $\text{MRRL}_{0.999}$ and $\text{MRRL}_{0.990}$. In all cases, the MSE experiments required greater than 85% of the running times required by FULLWARMUP, while MRRL generally required *less* than 70%. This is one of MRRL's more significant strengths over its predecessor: the ability to maintain a high level of accuracy in much less time than MSE.

Table VIII gives the running time for STALEWARMUP relative to FULLWARMUP, which we claim represents the maximum potential speedup for each benchmark for its set of sample clusters. Consider the three-stage

Table VIII. Maximum Potential ($\%_{\text{STALEWARMUP}}$) Acceleration ($100\% \cdot \frac{t_{\text{STALEWARMUP}}}{t_{\text{FULLWARMUP}}}$) and Achieved Percentage of Potential ($\%_{\text{MRRL}_N}$) Running Time Speedup ($100\% \cdot (1 - \frac{t_{\text{MRRL}_N} - t_{\text{STALEWARMUP}}}{t_{\text{STALEWARMUP}}})$)

| Benchmark | $t_{\text{FULLWARMUP}}$(s) | $\%_{\text{STALEWARMUP}}(\%)$ | $\%_{\text{MRRL}_{0.999}}(\%)$ | $\%_{\text{MRRL}_{0.990}}(\%)$ |
|---|---|---|---|---|
| applu | 80,887 | 60.54 | 93.79 | 94.64 |
| apsi | 120,925 | 58.11 | 97.51 | 97.68 |
| art_110 | 19,613 | 35.52 | 96.00 | 95.23 |
| crafty | 78,906 | 46.52 | 96.41 | 95.31 |
| equake | 54,675 | 55.49 | 97.07 | 95.83 |
| facerec | 70,587 | 47.88 | 91.44 | 93.16 |
| fma3d | 96,462 | 59.84 | 97.43 | 97.65 |
| galgel | 162,606 | 53.15 | 95.67 | 95.20 |
| gcc_integrate | 5569 | 65.49 | 84.12 | 95.31 |
| gzip_graphic | 26,643 | 32.31 | 92.98 | 92.03 |
| lucas | 46,730 | 48.55 | 95.65 | 95.39 |
| mcf | 36,014 | 44.55 | 95.74 | 94.99 |
| mgrid | 142,334 | 58.57 | 96.71 | 96.86 |
| twolf | 133,069 | 42.36 | 96.04 | 95.32 |
| vortex_lendian2 | 64,839 | 36.71 | 96.95 | 96.32 |
| vpr_route | 28,358 | 35.66 | 88.55 | 88.42 |
| Mean | | | 94.47 | 94.99 |

cold–warm–hot simulation loop discussed in Section 3. Since the complexity of the simulation performed by the cold phase is less than the complexity of the simulation performed by the warm phase, whose complexity is less than the hot phase, perinstruction simulation time within the cold phase is less than that for the warm phase, which is less than that for the hot phase (i.e., $O(\text{cold}) < O(\text{warm}) < O(\text{hot})$). MSE, MRRL, and STALEWARMUP reduce simulation running times by modifying the simulation of the precluster instructions to increase the number of cold phase instructions while reducing the number of warm phase instructions. (Recall that for a precluster–cluster pair, the cold phase and warm phase instructions compose the precluster set of instructions, while the hot phase is the set of cluster instructions.) Hence, the maximum running time reduction for a precluster–cluster pair eliminates all warm phase simulation, rendering the precluster–cluster pair a cold phase followed immediately by a cycle-accurate hot phase; this is precisely what STALEWARMUP does.

Table VIII gives the maximum potential speedup for each benchmark as a percentage of FULLWARMUP running time, and the percentage of the maximum potential actually achieved for $\text{MRRL}_{0.999}$ and $\text{MRRL}_{0.990}$. The percentage of the maximum potential speedup is calculated as a ratio of the difference of $\text{MRRL}_N$ and STALEWARMUP running times and the STALEWARMUP running time, thus:

$$100\% \cdot \left(1 - \frac{t_{\text{MRRL}_N} - t_{\text{STALEWARMUP}}}{t_{\text{STALEWARMUP}}}\right).$$

$\text{MSE}_{99.9\%}$ and $\text{MSE}_{95.0\%}$ are omitted because, as Table VII demonstrates, MSE cannot accelerate running times as well as MRRL. (The scarcity of unique references made it difficult for MSE to model the prescribed $m$ uniques

during the precluster regions. Thus, for precluster warmup, MSE degenerated, or nearly so, to FULLWARMUP and completely traded away simulation speed for accuracy.) In our experiments, $\text{MRRL}_{0.999}$ and $\text{MRRL}_{0.990}$ achieve nearly identical[11] amounts of the maximum potential speedup: roughly 95% on average.

## 7. CONCLUSIONS AND FUTURE WORK

Minimal subset evaluation and memory reference reuse latency are two techniques that can *reduce the running times of sampled simulations by reducing the amount of time spent warming up simulated cache and branch predictor state prior to each sample cluster*. MSE computes a minimal number of unique references that must be handled within a cache in order to touch a certain fraction of cache blocks at least once; warmup commences at a point prior to cycle-accurate simulation such that the MSE-prescribed number of unique references will be encountered. MRRL works by profiling the reuse latency (in number of completed instructions) between consecutive accesses to each memory address or control-flow instruction; warmup commences at a point coincident with a user-chosen percentile of reuse latency measurements.

Both warmup acceleration methodologies speed up simulation by performing simulation in three phases: cold, warm, and hot. Hot phase simulation models microarchitectural activity in cycle-accurate detail; this is the level of modeling done during the sample clusters. The warm phase and the cold phase jointly compose the precluster instruction stream. Whereas the cold phase models only the programmer-visible architected state, the warm phase additionally models cache and branch predictor activity. Hence, hot phase simulation is more time-consuming per simulated instruction than warm phase simulation, which is more time-consuming per simulated instruction than cold phase simulation. Both acceleration methodologies reduce simulation running times by simultaneously maximizing the duration of the cold phase and minimizing the duration of the warm phase. This is in contrast to other simulation research which only implement, (coarsely speaking) a warm phase and a hot phase of simulation [Perelman et al. 2003; Sherwood et al. 2001, 2002, Wunderlich et al. 2003]. While the experiments presented in this paper utilize random cluster sampling, a key research goal of MSE and MRRL was to develop a *warmup acceleration methodology* that is as independent as possible from the sampling regime. Both can offer even further acceleration to previously discussed sampling techniques, but both prioritize simulation accuracy above acceleration, and (as was discussed in conjunction

[11]As described in Section 5, we used the UNIX system call $getrusage()$ to accurately measure actual CPU running time, regardless of additional load placed upon the host CPU by other processes. To gauge walk-clock running time for *sim-outorder*, we summed the user-space CPU time and the kernel-space CPU running time. Since kernel-space timing on the host system (Linux/$\times$86) was nondeterministic (e.g., page faults, disk seek/access, higher priority threads, variable scheduling time), there is mild variation between the MRRL running times, and in some instances, $\text{MRRL}_{0.999}$ completes in less time than $\text{MRRL}_{0.990}$ for the same benchmark. Indeed, the mean percentage of realized maximum potential speedup is greater for $\text{MRRL}_{0.999}$. The amount of variation is so slight, however, that this is almost certainly the result of nondeterminism in the host system kernel.

with Wunderlich et al. [2003] in Section 2, and demonstrated for MSE in Section 6) will completely trade away speedup to preserve accuracy if there are insufficient intercluster instructions to justify shortening the warmup period.

Critically important to both acceleration methodologies is that neither MSE nor MRRL contributes significantly to nonsampling error due to cold-start bias. Whereas cache and branch predictor activity are always modeled during cycle-accurate cluster simulation, MSE and MRRL only model a subset of the cache and branch predictor activity immediately preceding each sample cluster. For both, the percent-error deviation from FULLWARMUP was less than 1% on average. This was shown to be statistically insignificant in that all but two benchmarks' 95% confidence intervals predicted the observed deviation from the true, end-to-end IPC; furthermore, FULLWARMUP simulations of the same two benchmarks also failed to predict the observed error. FULLWARMUP's resistance to cold-start nonsampling error implies that the failure of MSE, MRRL, and FULLWARMUP to predict the observed error is attributable to sampling error: the inevitable consequence of measuring only a subset of any population. Hence, both MSE, and MRRL accomplish the objective of maintaining small nonsampling error, and mimic FULLWARMUP well. Merely recycling cache and branch predictor state between successive clusters (as with STALEWARMUP) on the other hand, introduced large, statistically significant increases in the nonsampling bias.

While both MSE and MRRL perform very well at abating nonsampling bias, the clear victor at realizing the maximum potential speedup (i.e., the speedup achieved by STALEWARMUP) is MRRL cutting simulation running times by approximately 50% on average, which is roughly 95% of the maximum potential speedup. Thus, we have established MRRL, the successor and replacement to MSE, for accelerating warmup in sampled simulation.

With the growing interest in rigorous and accelerated simulation techniques, a major area for future work is to find the optimal combination of sampling and reduced-warmup techniques and parameters. Other interesting avenues of future research include determining whether statistical measurements of reference reuse latencies such as explored in Phalke and Gopinath [1995] (who model reuse latencies as $k$th order Markov chains) can be successfully applied to quantify temporal locality with sufficient accuracy to enable accurate, accelerated MRRL simulation, and whether single-pass methods can be used to facilitate accurate accelerated branch predictor warmup.

The MRRL profiler software and modified version of *sim-outorder* are linked to from the Laboratory for Computer Architecture at Virginia Web site at http://lava.cs.virginia.edu/.

ACKNOWLEDGMENTS

The authors would like to thank Prof. Tom Conte, Prof. Mircea Stan, Prof. Bradley Calder, and the anonymous reviewers for their valuable feedback and insights.

## REFERENCES


AUSTIN, T. M. SimpleScalar home page. http://www.simplescalar.com/.

AUSTIN, T. M. AND BURGER, D. C. 1998. Simplescalar 3.0 prerelease.

BURGER, D. C. AND AUSTIN, T. M. 1997. The simplescalar tool set, version 2.0. *Computer Architecture News 25*, 3 (June), 13–25.

COCHRAN, W. G. 1977. *Sampling Techniques*, 3rd ed. Wiley, New York.

CONTE, T. M., HIRSCH, M. A., AND MENEZES, K. N. 1996. Reducing state loss for effective trace sampling of superscalar processors. In *Proceedings of the International Conference on Computer Deisgn*. 468–477.

CROWLEY, P. AND BAER, J. L. 1999. On the use of trace sampling for architectural studies of desktop applications. In *Proceedings ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*. 208–209.

EECKHOUT, L., EYERMAN, S., CALLENS, B., AND DE BOSSCHERE, K. 2003. Accurately warmed-up trace samples for the evaluation of cache memories. In *Proceedings of the High Performance Computing Symposium—HPC2003*. 267–274.

FREUND, J. E. 1971. *Mathematical Statistics*. Prentice-Hall, Englewood Califf, NJ.

GIRBAL, S., MOUCHARD, G., COHEN, A., AND TEMAM, O. 2003. DiST: A simple, reliable and scalable method to significantly reduce processor architecture simulation time. In *Proceedings of ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*. 1–12.

HASKINS, JR., J. W. 2003. *Accelerating Sampled Microarchitecture Simulations: Rapid Warm Up for Simulated Hardware State*. Ph.D. Thesis, University of Virginia.

HASKINS, JR., J. W. AND SKADRON, K. 2001. Minimal subset evaluation: Rapid warm-up for simulated hardware state. In *Proceedings of the International Conference on Computer Design*. 32–39.

HASKINS, JR., J. W. AND SKADRON, K. 2003. Memory reference reuse latency: Accelerated warmup for sampled microarchitecture simulation. In *Proceedings of the International Symposium on Performance Analysis of Systems and Software*. 195–203.

HASKINS, JR., J. W., SKADRON, K., KLEINOSOWSKI, A. J., AND LILJA, D. L. 2002. *Techniques for Accurate, Accelerated Processor Simulation: Analysis of Reduced Inputs and Sampling*. Technical Report CS-2002-01, Univ. of Virginia, Dept. of Computer Science. Jan.

HENNESSY, J. L. AND PATTERSON, D. A. 1995. *Computer Architecture: A Quantitative Approach*, 2nd ed. Morgan Kaufman, San Mateo, CA.

HENRY, G. T. 1990. *Practical Sampling*. Sage, Thousand Oaks, CA.

KAXIRAS, S., HU, Z., AND MARTONOSI, M. 2001. Cache decay: Exploiting generational behavior to reduce cache leakage power. In *Proceedings of the 28th International Symposium on Computer Architecture*. 240–251.

KENDALL, M. G. AND STUART, A. 1968. *The Advanced Theory of Statistics*, 2nd ed. Hafner Publishing New York.

KESSLER, R. E., HILL, M. D., AND WOOD, D. A. 1991. *A Comparison of Trace-Sampling Techniques for Multi-Megabyte Caches*. Tech. Rep. 1048, Univ. of Wisconsin-Madison Computer Sciences Dept. Sept.

KESSLER, R. E., MCLELLAN, E. J., AND WEBB, D. A. 1996. *The Alpha 21264 Microprocessor Architecture.* Tech. Report, Compaq Computer Corporation. http://h18002.www1.hp.com/alphaserver/download/ev6chip.pdf.

KLEINOSOWSKI, A., FLYNN, J., MEARES, N., AND LILJA, D. J. 2000. *Adapting the SPEC 2000 Benchmark Suite for Simulation-Based Computer Architecture Research*. In *Proceedings of the 3rd IEEE Annual Workshop on Workload Characterization*. 73–82.

KLEINOSOWSKI, A. AND LILJA, D. J. 2002. MinneSPEC: *A New SPEC Benchmark Workload for Simulation-Based Computer Architecture Research*. Tech. Report 02–08, University of Minnesota ARCTiC Labs. Oct.

NGUYEN, A., WELLMAN, J., AND BOSE, P. 1997. PARSIM: A parallel trace-driven simulation facility for fast and accurate performance analysis studies. In *Proceedings of the International Performance Computing, and Communications Conference*. 291–297.

OCHOA, L. M. J., IBÁÑEZ, P. E., AND VIÑALS, V. 1996. Warm time-sampling: Fast and accurate cycle-level simulation of cache memory. In *Proceedings of the 22nd Euromicro International Conference Short Contributions*. 39–44.

PERELMAN, E., HAMERLY, G., AND CALDER, B. 2003. Picking statistically valid and early simulation points. In *Proceedings of the International Conference on Parallel Architecture and Compilation Techniques*. 244.

PHALKE, V. AND GOPINATH, B. 1995. An inter-reference gap model for temporal locality in program behavior. In *Proceedings of SIGMETRICS 1995*. 291–300.

SHERWOOD, T., CALDER, B., PERELMAN, E., AND HAMERLY, G. SimPoint home page. http://www-cse.ucsd.edu/ calder/simpoint/.

SHERWOOD, T., PERELMAN, E., AND CALDER, B. 2001. Basic block distribution analysis to find periodic behavior and simulation points in applications. In *Proceedings of the International Conference on Parallel Architecture and Compilation Techniques*. 3–14.

SHERWOOD, T., PERELMAN, E., AND CALDER, B. 2002. Automatically characterizing large scale program behavior. In *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems*. 45–57.

SKADRON, K., AHUJA, P. S., MARTONOSI, M., AND CLARK, D. W. 1999. Branch prediction, instruction-window size, and cache size: Performance tradeoffs and simulation techniques. *IEEE Trans. Comput. 48*, 11 (Nov.), 1260–1281.

STANDARD PERFORMANCE EVALUATION CORPORATION. 1999. SPEC CPU2000 Benchmarks. WWW site: http://www.specbench.org/osg/cpu2000.

STERNSTEIN, M. 1996. *Statistics*. Barron's Educational Series, Inc., New York.

TAYLOR, J. R. 1982. *An Introduction to Error Analysis: The Study of Uncertainty in Physical Measurements*. University Science Books, Mill Valley, CA.

THIÉBAUT, D. 1989. On the fractal dimension of computer programs and its application to the prediction of the cache miss ratio. *IEEE Trans. Comput. 8*, 7 (Jul.), 1012–1026.

UNDERWOOD, B., DUNCAN, C., TAYLOR, J., AND COTTON, J. 1954. *Elementary Statistics*. Appleton-Century-Crofts, New York.

WOOD, D. A., HILL, M. D., AND KESSLER, R. E. 1991. A model for estimating trace-sample miss ratios. In *Proceedings ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*. 79–89.

WUNDERLICH, R. E., WENISCH, T. F., FALSAFI, B., AND HOE, J. C. 2003. SMARTS: Accelerating microarchitecture simulation via rigorous statistical sampling. In *Proceedings of the 30th International Symposium on Computer Architecture*. 84–95.

Received April 2004; revised February 2005; accepted February 2005