An Evaluation of Stratified Sampling of Microarchitecture Simulations

Roland E. Wunderlich Thomas F. Wenisch Babak Falsafi James C. Hoe
Computer Architecture Laboratory (CALCM)
Carnegie Mellon University, Pittsburgh, PA 15213-3890
{rolandw, twenisch, babak, jhoe}@ece.cmu.edu
http://www.ece.cmu.edu/~simflex

Abstract

Recent research advocates applying sampling to accelerate microarchitecture simulation. Simple random sampling offers accurate performance estimates (with a high quantifiable confidence) by taking a large number (e.g., 10,000) of short performance measurements over the full length of a benchmark. Simple random sampling does not exploit the often repetitive behaviors of benchmarks, collecting many redundant measurements. By identifying repetitive behaviors, we can apply stratified random sampling *to achieve the same confidence as simple random sampling with far fewer measurements. Our oracle limit study of optimal stratified sampling of SPEC CPU2000 benchmarks demonstrates an opportunity to reduce required measurement by 43x over simple random sampling.*

Using our oracle results as a basis for comparison, we evaluate two practical approaches for selecting strata, program phase detection and IPC profiling. Program phase detection is attractive because it is microarchitecture independent, while IPC profiling directly minimizes stratum variance, therefore minimizing sample size. Unfortunately, our results indicate that: (1) program phase stratification falls far short of optimal opportunity, (2) IPC profiling requires expensive microarchitecture-specific analysis, and (3) both methods require large sampling unit sizes to make strata selection feasible, offsetting their reductions of sample size. We conclude that, without better stratification approaches, stratified sampling does not provide a clear advantage over simple random sampling.

1. Introduction

One of the primary design tools in microarchitecture research is software simulation of benchmark applications. Timing-accurate simulation's flexibility and accuracy makes it indispensable to microarchitecture research. However, the applications we wish study continue to increase in length—hundreds of billions of instructions for SPEC CPU2000 (SPEC2K). At the same time, the speed gap between simulators and the simulated hardware is growing—with as much as five orders of magnitude slowdown currently. Thus, researchers have begun looking for ways to accelerate simulation without sacrificing the accuracy and reliability of results [1,5,6,7].

One of the most promising approaches to accelerate simulation is to evaluate only a tiny sample of each workload. Previous research has demonstrated highly accurate results while reducing simulation run time from weeks to hours [6,7]. These sampling proposals pursue two different approaches to sample selection: (1) statistical uniform sampling of a benchmark's instruction stream, and (2) targeted sampling of non-repetitive benchmark behaviors. Uniform sampling, such as the SMARTS framework [7], has the advantage that it requires no foreknowledge or analysis of benchmark applications, and it provides a statistical measure of the reliability of each experimental result. However, this approach ignores the vast amount of repetition within most benchmark's instruction streams, taking many redundant measurements. Targeted sampling instead categorizes program behaviors to select fewer measurements, reducing redundant measurements. The SimPoint approach [6] identifies repetitive behaviors by summarizing fixed-size regions of the dynamic instruction stream as *basic block vectors (BBV)*, building clusters of regions with similar vectors, and taking one measurement within each cluster.

The benefits of both sampling approaches can be achieved by placing the phase identification techniques of targeted sampling in a statistical framework that provides a confidence estimate with each experiment. *Stratified random sampling* is this statistical framework. Stratified sampling breaks a population into strata, analogous to targeted sampling, and then randomly samples within each stratum, as in uniform sampling. By separating the distinct behaviors of a benchmark into different strata, each behavior can be characterized by a small number of measurements. Each of these characterizations is then weighted by the size of the stratum to compute an overall estimate. The aggregate number of measurements can be lower than the number required by uniform sampling.

The effectiveness of stratified sampling can be evaluated along two dimensions. First, it might reduce the total

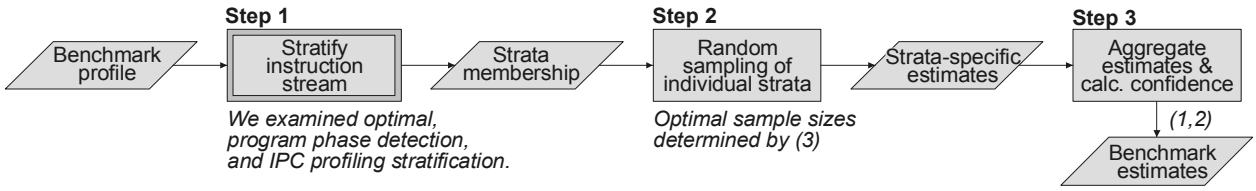

Figure 1. The stratified random sampling process. We focus on the relative effectiveness of two practical stratification approaches for Step 1 in this work. The referenced equations for Steps 2 & 3 are in Section 2.

quantity of measurements required. For simulators where a large number of measurements implies significant cost—for example, the storage of large architectural state checkpoints to launch each measurement—a reduction of measurements would imply cost savings.

More commonly, however, the total number of instructions measured has the larger impact on simulation cost. To improve total measurement, a stratification approach must reduce the quantity of required measurements while maintaining the small measurement sizes achievable with simple random sampling.

In this study, we evaluate the practical merit of combining sample targeting with statistical sampling in the form of stratified random sampling. We perform an oracle limit study to establish bounds on improvement from stratification and evaluate two practical stratification approaches: program phase detection and IPC profiling. We evaluate both approaches quantitatively in terms of *sample size* (measurement quantity) and *sampling unit size* (measurement size), and qualitatively in terms of the up-front cost of creating a stratification. We demonstrate:

- **Limited gains in sample size:** We show that stratifying via program phase detection achieves only a small reduction in sample size over uniform sampling, 2.2*x,* in comparison to the oracle opportunity of 43*x*. Phase detection assures that each stratum has a homogenous instruction footprint. Unfortunately, data effects and other sources of performance variation remain. The reduction in CPI variability achieved by stratifying on instruction footprint is not sufficient to approach the full opportunity of stratification.
- **Expensive analysis and limited applicability:** We show that IPC profiling requires an expensive analysis that is microarchitecture specific, and its gains do not justify this cost.
- **No improvement in total measurement:** We show that neither stratification approach improves over simple random sampling in terms of total instructions measured. Because of the computational complexity of clustering, neither stratification approach can be applied at the lowest sampling unit sizes achievable with random sampling. This increase in sampling unit size offsets reductions in sample size for stratified sampling.

The remainder of this paper is organized as follows. Section 2 presents stratified random sampling theory and details how to correctly achieve confidence in results from a stratified population. Section 3 discusses our optimal stratification study, while Section 4 covers our evaluations of two practical stratification techniques. In Sections 3 and 4, we explicitly cover the improvements to sample size and total measured instructions as compared to simple random sampling for each technique. We conclude in Section 5.

2. Stratified random sampling

The confidence in results of a simple random sample is directly proportional to the sample size and the variance of the property being measured. The sample size is the number of measurements taken to make up a sample, and variance is the square of standard deviation. Significant reductions in sample size can often be achieved when a population can be split into segments of lower variance than the whole.

Stratified random sampling of a population is performed by taking simple random samples of *strata*, mutually exclusive segments of the population, and aggregating the resulting estimates to produce estimates applicable to the entire population. Strata do not need to consist of contiguous segments of the population, rather every population member is independently assigned to a stratum by some selection criteria. If stratifying the population results in strata with relatively low variance, a small sample can measure each stratum to a desired confidence. By combining the measurements of individual strata, we can compute an overall estimate and confidence. With low variance strata, the aggregate size of a stratified sample can be much smaller than a simple random sample with equivalent confidence. A population whose distinct behaviors are assigned to separate strata will see the largest decreases in sample size when using stratified sampling.

The process of stratified random sampling is illustrated in Figure 1. The first of three steps is to stratify the population into K strata. We discuss various techniques for stratifying populations in the context of microarchitecture simulation in Sections 3 and 4. Second, we collect a simple random sample of each stratum. We represent the variable of interest as x, and strata-specific variables with the subscript h, where h ranges from 1 to K. Therefore, N_h

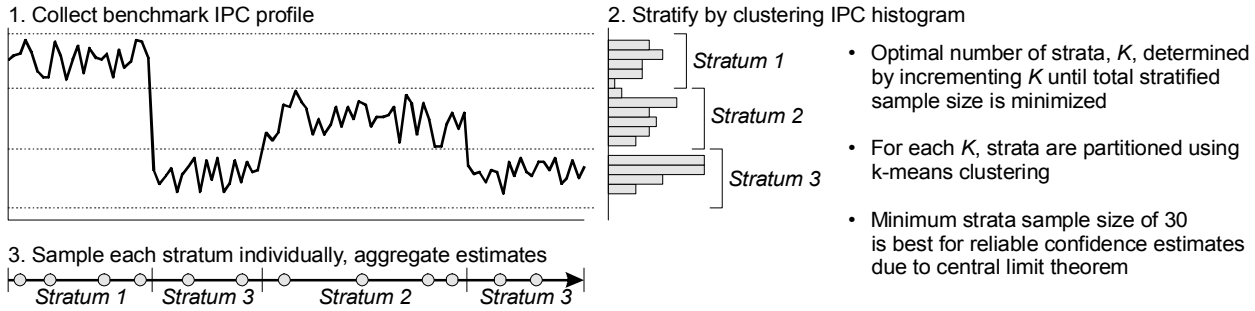

Figure 2. Determining the optimal stratification for a particular benchmark and microarchitecture. Collecting the IPC profile requires performance simulation of the full length of the target benchmark.

is the population size of stratum h, n_h is the sample size for stratum h, while σ_{hx} is that stratum's standard deviation of x. The final step is to aggregate the individual stratum estimates to produce estimates of the entire population. A simple weighted mean is used to produce a population mean estimate:

$$\bar{x} = \frac{\sum (N_h / N)\bar{x}_h}{K} \quad (1)$$

where the summation is over all strata of the population (h = 1 to K); thus, $\sum N_h = N$, and $\sum n_h = n$. Note, we assume $N_h \gg n_h \gg 1$ to simplify the stratified sampling expressions. The confidence interval of a mean estimate from a stratified random sample is determined by:

$$\pm(\varepsilon \cdot \bar{X}) \approx \pm z \sqrt{\sum \left(\frac{N_h}{N}\right)\left(\frac{\hat{\sigma}_{hx}^2}{n_h}\right)} \quad (2)$$

where z is the $100[1-(\alpha/2)]$ percentile of the standard normal distribution (z = 2.0 for 95% and z = 3.0 for 99.7% confidence). Note that a sampling estimate of a stratum's standard deviation is marked with a hat as $\hat{\sigma}_{hx}$.

The required sample size for each stratum, n_h, which produces a desired overall confidence interval with minimum total sample size n can be calculated if the standard deviation of each stratum σ_{hx} is known or can be estimated. The procedure for calculating the optimal stratified sample is known as *optimal sample allocation* [2]. To determine an optimally-allocated stratified sample for a desired confidence interval we first calculate the total stratified sample size:

$$n \geq \frac{\left(\frac{z^2}{N^2}\right)\left(\sum \frac{N_h^2 \sigma_{hx}^2}{\pi_h \bar{X}^2}\right)}{\varepsilon + \left(\frac{z^2}{N^2}\right)\left(\sum \frac{N_h \sigma_{hx}^2}{\bar{X}^2}\right)} \text{ where } \pi_h = \frac{N_h \sigma_{hx}}{\sum N_h \sigma_{hx}} \quad (3)$$

The sample size of each stratum is the fraction π_h of the total stratified sample size n; individual stratum sample sizes are $n_h = \pi_h \cdot n$.

3. Optimal stratification

In order to evaluate practical stratification approaches for the experimental procedure presented in Section 2, we first quantify the upper bound reduction in sample size achievable with an optimal stratification. As in previous studies of simulation sampling [7, 6], we focus on CPI as the target metric for our estimation, and use the same 8-way and 16-way out-of-order superscalar processor configurations, SPEC2K benchmarks, and simulator codebase as [7].

Determining an optimal stratification for CPI requires knowledge of the CPI distribution for the full length of an application—knowledge which obviates the need to estimate CPI via sampling. To perform this study, we have recorded complete traces of the per-unit IPC (not CPI, for reasons explained later) of every benchmark on both configurations. While not a practically applicable technique, this study establishes the bounds within which all practical stratification methods will fall. At worst, an arbitrary stratification approach will match simple random sampling, as random assignment of sampling units to strata is equivalent to simple random sampling. At best, any approach will match the bound established here.

Optimal stratified sampling. To minimize total sample size, we need to determine an optimal number of strata, and minimize their respective variances. Then, we calculate the correct sample size for a desired confidence using the optimal stratified sample allocation equation (3). This equation provides the best sample size for each stratum, given their variances and relative sizes. Larger and higher variance strata receive proportionally larger samples. We constrain sample size for each stratum to a minimum of 30 (or the entire stratum, if it contains fewer than 30 elements) to ensure that the central limit theorem holds, and that our confidence calculations are valid [2].

The optimal number of strata, K, cannot be determined in closed form. Intuitively, more strata allows finer classification of application behavior, reducing variance within each stratum, and therefore reducing sample size. However, at some critical K, the floor of 30 measurements

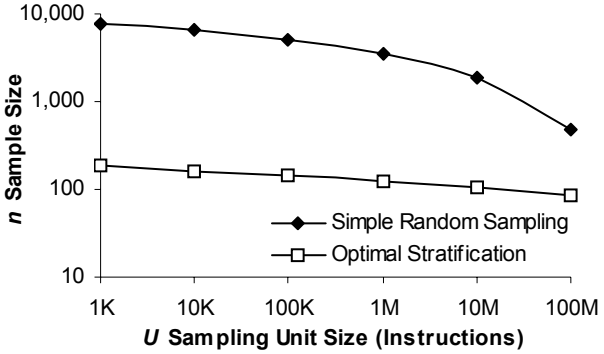

Figure 3. Optimal stratification's average sample size per benchmark vs. simple random sampling for SPEC2K with the 8-way processor configuration.

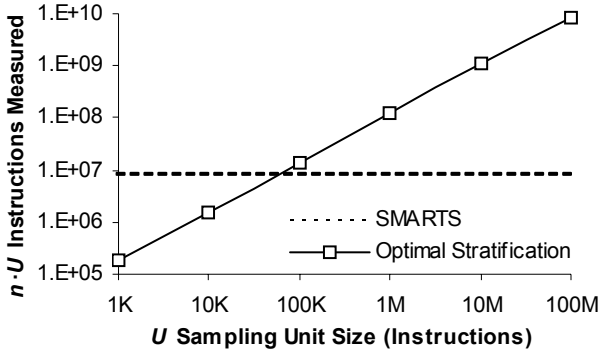

Figure 4. Total measured instructions per benchmark with optimal stratification.

per stratum dominates and increasing K increases sample size. For each combination of benchmark, microarchitecture, and sampling unit size, U, we determine the total stratified sample size for each value of K up to the optimal value, by starting with $K = 1$ and stopping when total sample size decreases to a minimum.

For each value of K, we determine the optimal assignment of sampling units to strata such that the CPI variance of each stratum is minimized. We employ the k-means clustering algorithm, using the implementation described in [3] that utilizes *kd*-trees and blacklisting optimizations. The k-means algorithm is one of the fastest clustering algorithms, and the implementation in [3] is optimized for clustering large data sets, up to approximately 1 million elements. (Beyond 1 million elements, the memory and computation requirements render the approach infeasible.) Each k-means clustering was performed with 50 random seeds to ensure an optimal clustering result. To stratify the large populations of SPEC2K benchmarks at small U (on average 174 million sampling units per benchmark at $U = 1000$ instructions), we must reduce the data set before clustering. Figure 2 illustrates how we reduce the data set without impacting clustering results. We assign sampling units to bins of size 0.001 IPC, and then cluster the bins using their center and membership count. We bin based on IPC rather than CPI as IPC varies over a finite range for a particular microarchitecture (i.e., 0 to 8 for our 8-way configuration, thus, 8000 bins). As long as the number of bins is much larger than K, and the variance within a bin is negligible relative to overall variance, binning does not adversely affect the results of the clustering algorithm.

After each clustering, we calculate the variance of the resulting strata and determine an optimal sample size as previously described. We iterate until the critical value for K is encountered. The optimal K lies between one and ten clusters for all benchmarks and configurations that we studied, and tends to decrease slightly with increasing U. Note that the optimal K is independent of the target confidence interval.

Impact on sample size. Figure 3 illustrates the impact of stratification on sample size, n, for the 8-way configuration. The top line in the figure represents the average sample size required for a simple random sample to achieve 99.7% confidence of ±3% error across all benchmarks. The bottom line depicts the average sample size with optimal stratification. Stratification can provide a 43*x* improvement in sample size for $U = 1000$ instructions, reducing average sample size from ~8000 to 185 measurements per benchmark. This result demonstrates that random sampling takes many redundant measurements, and that there is significant opportunity for improvement with an effective stratification technique.

Impact on total measured instructions. Figure 4 illustrates the impact of stratification on total measured instructions, $n \cdot U$. The dashed line illustrates the total instructions required for the SMARTS technique, which performs systematic sampling at $U = 1000$ instructions. The graph shows that any practical stratification approach must be applied at a unit size of 10,000 instructions or smaller in order to have a possibility of outperforming existing sampling methodology.

4. Practical stratification approaches

The optimal stratification study presented in Section 3 establishes upper and lower bounds by which we can measure the effectiveness of any stratification approach. However, creating the optimal stratification requires knowledge of the CPI distribution for the full length of an application, and is optimal only for that specific microarchitecture configuration. In order for stratification to be useful, we must balance the cost of producing a stratification with the time saved relative to simple random sampling over the set of experiments which can use the stratification. Thus, we desire stratifications that can be computed cheaply and can be applied across a wide range of microarchitecture configurations. In the following subsections, we analyze two promising stratification

approaches. However, we find that both approaches obtain insufficient execution time improvements over simple random sampling to justify their large costs.

4.1. Program phase detection

SimPoint [6] presents program phase detection as a promising approach for identifying and exploiting repetitive behavior in benchmarks to enable acceleration of microarchitecture simulation. SimPoint identifies program phases based upon a basic-block vector profile. SimPoint clusters measurement units based on the similarity of portions of the BBV profiles.

Statistically Valid SimPoint [4] presents a method for evaluating the statistical confidence of SimPoint simulations where only a single unit is measured from each cluster. However, the proposed use of parametric bootstrapping only provides confidence interval estimates for the specific microarchitecture where the bootstrap is performed, and does not account for individual experiment's variations in performance. In addition, this analysis requires CPI data for many points within each cluster.

Instead, by applying BBV phase detection in the context of stratified random sampling, we can obtain a confidence estimate with every experiment. By measuring at least 30 units from each stratum (BBV cluster), we satisfy the conditions of the central limit theorem and obtain a confidence estimate with each simulation experiment. The number of strata was optimally selected using the same technique as our optimal stratification study in Section 3. SimPoint seems a promising approach for stratification, as it achieves both of the goals outlined earlier. First, basic block vector analysis is relatively low cost, as it can be accomplished using a BBV trace obtained by functional simulation or direct execution of instrumented binaries (if experimenting with an implemented ISA). Second, basic block vectors are independent of microarchitecture, and thus, the resulting stratification can be applied across many experiments.

Practical costs. The primary costs of program phase stratification are the collection of a benchmark's raw BBV data and the clustering analysis time. Collection of BBV data can be done with direct execution for existing instruction set architectures, otherwise functional simulation is required. For the unit sizes advocated in [4] and [6] of 1 million to 100 million instructions, analysis time for clustering is a few hours at most. However, clustering quickly becomes intractable as we reduce U further. It is infeasible to compute a k-means clustering for $U < 100{,}000$, since, for most SPEC2K benchmarks, this results in more than 1 million sampling units. The high dimensionality (15 dimensions after random linear projection) of BBV data prevents the binning optimization done for the optimal stratification study in Section 3 due to the sparseness of the vector space.

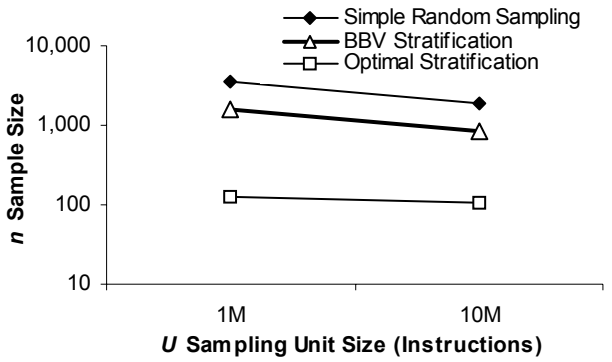

Figure 5. BBV program phase stratification average sample size with the 8-way microarchitecture. BBV stratification reduces average sample size by 2.2x over simple random sampling, but it requires $U > 100{,}000$.

Impact on sample size. Program phase detection does provide a modest improvement in sample size over simple random sampling. However, phase detection falls short of optimal stratification since it seeks to ensure the homogeneity of the instruction footprint of each stratum. This does not necessarily lead to minimal CPI variance within each stratum. On average, program phase clustering improves sample size by only 2.2*x* over simple random sampling as shown in Figure 5. The average sample size at $U = 1$ million instructions was 3590 for simple random sampling and 1615 for BBV stratified random sampling, as compared to 125 for optimal stratified sampling.

Impact on total measured instructions. Because the BBV clustering analysis cannot be performed for U below 100,000, stratification based on program phase cannot match the total measured instructions achievable with simple random sampling. With $U = 1$ million, BBV stratification results in an average of 1.6 billion instructions measured per benchmark, while a simple random sample with $U = 1000$ requires only 8 million instructions per benchmark to be measured.

4.2. IPC profiling

The optimal stratification study in Section 3 achieves large gains with stratification by stratifying directly on the target metric, in this case CPI. Optimal stratification can not be done for each experiment in practice because it requires the very same detailed simulation that we are trying to accelerate. However, if it were possible to perform this expensive stratification *once* per benchmark on a test microarchitecture, and then apply this stratification to many other microarchitecture configurations over many experiments, the long term savings might justify the one time cost. The key question is whether strata with minimal variance on one microarchitecture also have low variance on another microarchitecture. We evaluate the promise of this approach by computing a stratification using an IPC profile of our 8-way processor configuration

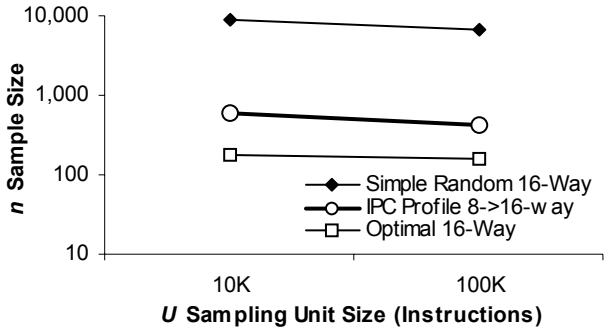

Figure 6. Average sample size per benchmark for IPC profile stratification with an 8-way profile applied to the 16-way microarchitecture configuration.

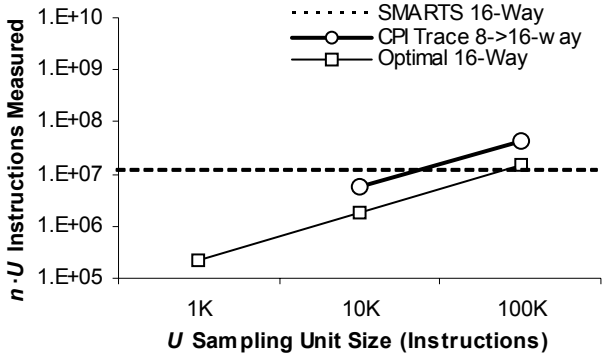

Figure 7. Total measured instructions per benchmark with IPC profile stratification.

and evaluating this stratification when applied to the 16-way configuration. The two microarchitectures differ in their fetch, issue and commit widths, functional units, memory ports, branch predictor and cache configurations, and cache latency (details in [7]).

Practical costs. This approach needs a trace of the IPC of every block of U instructions, requiring a detailed simulation of the entirety of every benchmark. The longest SPEC2K benchmarks require up to a month to simulate in detail. We have successfully clustered sampling units for $U = 10{,}000$, but storage requirements and processing time prevent clustering at $U = 1000$ instructions. Unlike the optimal stratification experiment of Section 3, practical use of IPC profile stratification requires storing the strata assignment of every sampling unit to disk (to allow strata selection for a second experiment), and the storage needs becomes prohibitive at $U = 1000$.

Impact on sample size. Two measurements units which have identical performance on one microarchitecture, and are thus members of the same stratum, may be affected differently by microarchitectural changes, increasing variance in the stratum. Thus, a larger sample is required to accurately assess the stratum. Figure 6 compares the sample size obtained with an 8-way IPC profile stratification to the optimum stratification and simple random sampling for the 16-way configuration. The 8-way stratification improves over purely random sampling by a factor of 15x, as compared to an opportunity of 48x for the 16-way microarchitecture. An IPC profile stratification will provide large returns only for microarchitectures very similar to the test microarchitecture that generated the profile.

Impact on total measured instructions. As Figure 7 shows, IPC profile stratification at $U = 10{,}000$ roughly breaks even with SMARTS in terms of total measured instructions. This performance does not justify the significant one time cost of creating the stratification. Even if a method were developed which could stratify at $U = 1000$, the limited microarchitecture portability of the stratification renders it unlikely that the high cost of generating an IPC profile will be worthwhile.

5. Conclusion

While our opportunity study of stratified sampling shows promise for reducing sample size, our analysis of practical stratification techniques indicates little advantage over simple random sampling. Program phase detection stratification achieves only a small fraction of the available opportunity, since the discovered homogenous instruction footprints do not translate to homogenous performance. IPC profiling requires expensive and potentially non-portable stratification that is not justified by improvements in sample size. Neither approach improves in total measurement over simple random sampling because stratification cannot be performed at small sampling unit sizes. Thus, we conclude that stratified sampling provides no benefit for the majority of sampling simulators where the primary interest is in reducing total instructions measured.

6. References

[1] V. Krishnan and J. Torrellas. A direct-execution framework for fast and accurate simulation of superscalar processors. In *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques*, Oct. 1998.

[2] P. S. Levy and S. Lemeshow. *Sampling of Populations: Methods and Applications*. John Wiley & Sons, Inc., 1999.

[3] D. Pelleg and A. Moore. Accelerating exact k-means algorithms with geometric reasoning. In S. Chaudhuri and D. Madigan, editors, *Proceedings of the Fifth International Conference on Knowledge Discovery in Databases*, pages 277–281. AAAI Press, Aug. 1999.

[4] E. Perelman, G. Hamerly, and B. Calder. Picking statistically valid and early simulation points. In *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques*, Sep 2003.

[5] E. Schnarr and J. Larus. Fast out-of-order processor simulation using memoization. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*, Oct. 1998.

[6] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. Automatically characterizing large scale program behavior. In *Proceedings of the Tenth International Conference on Architectural Support for Programming Languages and Operating Systems*, Oct. 2002.

[7] R. E. Wunderlich, T. F. Wenisch, B. Falsafi, and J. C. Hoe. SMARTS: Accelerating microarchitecture simulation via rigorous statistical sampling. In *Proceedings of the International Symposium on Computer Architecture*, June 2003.