Adaptive Simulation Sampling Using An Autoregressive Framework

Sharookh Daruwalla
Department of Computer Science
Portland State University
Portland OR, U.S.A.
sharookh@cecs.pdx.edu

Resit Sendag
Department of Electrical & Computer Engineering
University of Rhode Island
Kingston, RI, U.S.A.
sendag@ele.uri.edu

Joshua Yi
Networking & Computer Systems Group
Freescale Semiconductor Inc.
Austin, TX, U.S.A.
joshua.yi@freescale.com

Abstract – Software simulators remain several orders of magnitude slower than the modern microprocessor architectures they simulate. Although various reduced-time simulation tools are available to accurately help pick truncated benchmark simulation, they either come with a need for offline analysis of the benchmarks initially or require many iterative runs of the benchmark. In this paper, we present a novel sampling simulation method, which only requires a single run of the benchmark to achieve a desired confidence interval, with no offline analysis and gives comparable results in accuracy and sample sizes to current simulation methodologies. Our method is a novel configuration independent approach that incorporates an Autoregressive (AR) model using the squared coefficient of variance (SCV) of Cycles per Instruction (CPI). Using the sampled SCVs of past intervals of a benchmark, the model computes the required number of samples for the next interval through a derived relationship between number of samples and the SCVs of the CPI distribution. Our implementation of the AR model achieves an actual average error of only 0.76% on CPI with a 99.7% confidence interval of ±0.3% for all SPEC2K benchmarks while simulating, in detail, an average of 40 million instructions per benchmark.

I. Introduction

Since the speed of most cycle-accurate, execution-driven simulators is several orders of magnitude slower than silicon and since architects simulate programs that are intended to run on real machines, simulating all benchmarks in a suite to completion is virtually impossible. To minimize simulation time, architects typically simulate only a subset of benchmarks or use reduced-time simulation techniques. Reduced-time simulation techniques try to minimize simulation time while still achieving results that are representative of the complete program execution. Current reduced-time simulation techniques include finding and simulating representative sections of a program [2,3,4,5,8], random or periodic sampling of the program's execution streams [6,7], methods that use a combination of both [9], and reduced input set simulation [17,18]. SimPoint [3,4,5] and SMARTS [7] are the most notable and popular of these techniques. Although these sampling-based techniques are very accurate, they require several profiling runs before simulation to properly configure the sampling algorithm. In this paper, we propose a novel sampling simulation method that eliminates the need for offline analysis or multiple iterative runs with an Autoregressive (AR) model that adapts to the changing behavior of the benchmark. The proposed AR model predicts the size of the next interval and the required number of samples within that interval through a derived relationship between the number of samples and the Squared Coefficient of Variance (SCV) of the Cycles per Instruction (CPI) distribution in past intervals. Our adaptive sampling technique requires only a single simulation run to achieve a desired confidence and maintains the same level of accuracy compared to current sampling-based techniques.

The contributions of this paper are as follows:

1. It analyzes the SCVs of CPI distributions in varying and fixed size intervals of SPEC2K benchmark programs and evaluates the correlation between past and current interval SCVs.
2. It presents a dynamically adaptive sampling simulation technique based on an AR model, which does not require offline analysis or multiple runs of the benchmarks.

The remainder of the paper is organized as follows. Section II describes the problem tackled; Section III presents the AR framework and its implementation. Section IV outlines the evaluation methodology while Section V evaluates the effectiveness of the AR framework. Section VI describes some related work and we conclude in Section VII.

II. Sampling With An Error Bound

Although current sampling-based simulation tools are highly accurate, they come with an overhead of offline analysis before actual simulation or need to iteratively re-run simulations until a desired level of confidence in the estimation is obtained. It is therefore desirable to have an adaptive sampling framework that does not require offline analysis and can complete an entire benchmark's simulation within a single iteration. Borrowing ideas from statistics, in this paper, we develop such a framework. This section introduces the proposed AR framework and the theory behind it. Based on sampling theory, the derived equations in this section show that the minimum number of samples needed to achieve a desired confidence and limit the sampling error within a desired tolerance level, is linearly proportional to the squared coefficient of variance (SCV) of the CPI distribution. However, SCVs are not known until after sampling is done.

This work was supported in part by National Science Foundation Grant CCF-0541162.

Our AR framework uses an Autoregressive model to predict future SCVs based on previously sampled SCVs during a benchmark's simulation. We then compute the number of samples needed to achieve the desired confidence in an AR predicted interval of the benchmark. The remainder of this section discusses why an AR model can be used for dynamic sampling microarchitecture simulation, and describes the implementation of the AR framework.

A. Bounding Sampling Errors in CPI Estimation

CPI is defined as the average number of cycles needed to commit one instruction. Suppose a program's execution stream is divided into non-overlapping observation intervals (*B*) of fixed sampling units. A sampling unit is defined as a fixed length of consecutive instructions in a program's execution stream. Assume that there are *m* sampling units in an interval, where X_i (i = 1,2… m) is the CPI of the i^{th} sampling unit. Hence the average CPI of an interval can be given by,

$$\boldsymbol{CPI} = \frac{\mathbf{1}}{\boldsymbol{m}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{m}} \boldsymbol{X_i}$$

In order to estimate the CPI of an interval, suppose we periodically sample *n* sampling units out of a total *m* sampling units. Let X_j (j = 1,2… n) denote the CPI of the j^{th} sampling unit. Therefore, the estimated CPI, *CPI'*, for the same interval can be given by,

$$\boldsymbol{CPI'} = \frac{\mathbf{1}}{\boldsymbol{n}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{X_j} \qquad .. (1)$$

For a sufficiently large sample size ($n > 30$), it can be shown that *CPI'* is an unbiased estimator of *CPI* [13]. The estimator can also be shown to be consistent i.e. as $n \rightarrow m$, $CPI' \rightarrow CPI$.

The objective of this paper is to bound the relative error $\left|\frac{\boldsymbol{CPI'-CPI}}{\boldsymbol{CPI}}\right|$ within a desired error tolerance level, i.e.

$$\boldsymbol{Pr}\left\{\left|\frac{\boldsymbol{CPI'-CPI}}{\boldsymbol{CPI}}\right| > \boldsymbol{\varepsilon}\right\} \leq \boldsymbol{\eta} \qquad .. (2)$$

Eq.(2) states that the relative error in CPI estimation can be bounded by ε with a probability of (1-η). The main question to be asked is – What is the minimum number of sampling units required to be sampled to maintain the desired accuracy?

B. Optimal Sampling Probability

The central limit theorem states that as sample size $n \rightarrow \infty$, the average of the sampled data approaches the population mean, regardless of the population distribution. Based on this theory, Eq. (2) can be rewritten as [14]:

$$\boldsymbol{Pr}\left\{\left|\frac{\boldsymbol{CPI'-CPI}}{\boldsymbol{CPI}}\right| > \boldsymbol{\varepsilon}\right\} \approx \mathbf{2}\left(\mathbf{1} - \boldsymbol{\Phi}\left(\frac{\boldsymbol{\varepsilon.\mu.\sqrt{n}}}{\boldsymbol{\sigma}}\right)\right) \leq \boldsymbol{\eta} \qquad .. (3)$$

where μ is the population mean and σ is the standard deviation of the CPI distribution in an interval. $\Phi(.)$ is the cumulative distribution function of a standard normal distribution i.e. $N(0,1)$. Therefore, we can use Eq.(3) to calculate the required number of sampling units to maintain the desired error tolerance [14],

$$\boldsymbol{n} \geq \boldsymbol{n'} = \left(\frac{\boldsymbol{\Phi^{-1}(1-\eta/2)}}{\boldsymbol{\varepsilon}}\right)^{\mathbf{2}} * \left(\frac{\boldsymbol{\sigma}}{\boldsymbol{\mu}}\right)^{\mathbf{2}} = \boldsymbol{z} * \boldsymbol{S} \qquad .. (4)$$

where $\boldsymbol{z} = \left(\frac{\boldsymbol{\Phi^{-1}(1-\eta/2)}}{\boldsymbol{\varepsilon}}\right)^{\mathbf{2}}$ and $\boldsymbol{S} = \left(\frac{\boldsymbol{\sigma}}{\boldsymbol{\mu}}\right)^{\mathbf{2}}$ is the squared coefficient of variance (SCV) of the CPI distribution in an interval. Eq.(4) shows a relation between minimum number of sampling units needed, to the estimation accuracy and the variability in CPI. It shows that the minimum number of sampling units, *n'*, is linearly proportional to the SCV, *S*, of the CPI distribution in a given interval. Although Eq.(4) works well for estimating CPI, it also means that we need to know the actual SCV of each interval, which is not available to us at runtime. To overcome this problem, we use an Autoregressive (AR) model to predict this parameter of each interval based on previously sampled intervals.

III. Adaptive Sampling using Autoregression

When implementing a regression model, there is a choice between using a simple linear regression model that would use one or more independent variables to predict a dependant variable versus using an autoregressive model that regresses upon itself [15]. We empirically tested an extensive set of parameters, including but not limited to past observations of sampled standard deviation, variance, SCV, number of samples, error in estimating past intervals' SCVs, on simple linear regression and autoregression models. We also tested these models using one or more independent variables as well as taking into account single or higher order interactions. We observed that, empirically, an Autoregressive Model, used on the Squared Coefficients of Variance (SCV) and taking into account second order interactions, fits best into our requirements of small sample sizes and tight confidence intervals.

A. Correlation between intervals

Prediction only works when there is strong correlation between past and future values of a parameter. In Figure 1, we show the scatter plots for SPEC2K benchmark *177.mesa* to show the relation between current and past 1, 10, and 20 intervals respectively. The scatter plots of *177.mesa* are consistent with most of the SPEC2K benchmarks. We observed that the correlation coefficients between current and past 1, 10, and 20 intervals was 0.513, 0.708, and 0.708 respectively (values close to 1 show highest correlation). From Figure 1, we see that the SCV correlation between the current and immediately past interval (top graph) is not as highly correlated as those between the current and past 10 or 20 intervals. Therefore, implementing an Autoregressive (AR) model that uses just the past interval for a next interval prediction would not be as accurate as using past 10 or 20 intervals.

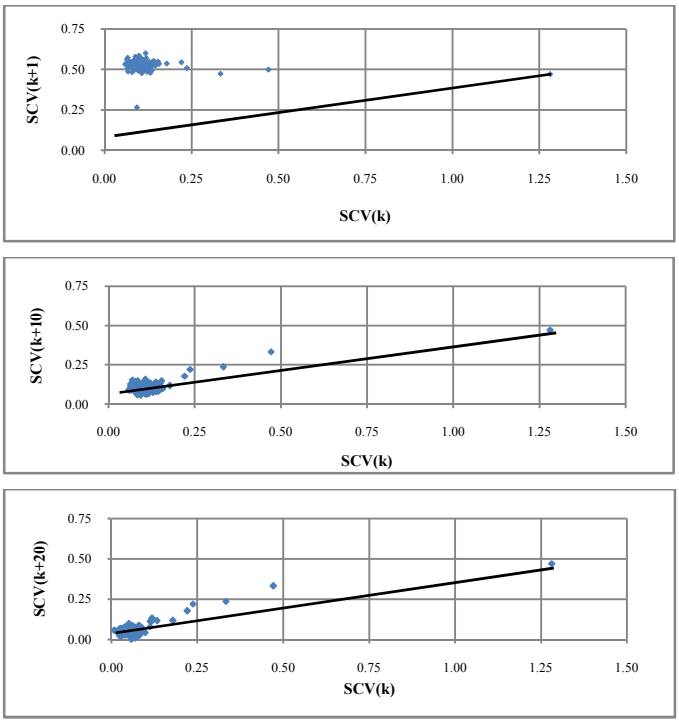

Figure1: Correlation between SCVs of past 1, 10, and 20 intervals on the CPI Distribution for *177.mesa* for $B = 100$.

B. The AR Model

A simple regression model is a compact mathematical representation of the relationship between a response variable x and input parameters y in a given design space [14]. When the model's dependent and independent variables are the same i.e. when the past values of a variable are used to predict the present value, the model becomes an Autoregressive (AR) model. In this section, we explain the AR model we used for CPI estimation. The same procedure is used for interval size estimation, which is briefly summarized at the end of the section.

Let us assume that $S(k)$ is the SCV of the CPI distribution for the k^{th} interval and $S'(k)$ is the SCV of the sampled CPI distribution for the same interval. Then, we can relate $S(k)$ and $S'(k)$ as follows:

$$\boldsymbol{S'(k) = S(k) \pm Z(k)} \quad .. (5)$$

where $Z(k)$ denotes the error in estimating the actual SCV of an interval. Using the AR(u) model, $S'(k)$ can be expressed as:

$$\boldsymbol{S'(k) = \sum_{i=1}^{u} a'(i) S'(k-i) \pm e'(k)} \quad .. (6)$$

where u is the number of past interval the AR model takes into account to predict the next interval's SCV, $a(i)$, $\{i = 1,2 \ldots u\}$ are the AR model parameters, and $e'(k)$ is the uncorrelated error (also known as the *prediction error*). The model parameters $a(i)$, $\{i = 1,2 \ldots u\}$ can be determined by solving a set of linear equations using the Yule-Walker approach [15,16] making it suitable for run-time CPI estimation and interval size prediction as it would not increase simulation time. Using this AR model, at the end of the $(k-1)^{th}$ interval, we predict the SCV of the k^{th} interval using the SCV values of the previously sampled u intervals as follows:

$$\boldsymbol{S''(k) = \sum_{i=1}^{u} a'(i) S'(k-i)} \quad .. (7)$$

If we combine Eq. (5), (6) and (7), we get

$$\boldsymbol{S''(k) = S(k) \pm Z(k) \pm e'(k)} \quad .. (8)$$

Eq.(8) shows that there are two types of errors occurring when we predict the SCV of the next interval using the sampled SCVs of the previous u intervals. The first is the estimation error $Z(k)$ introduced from sampling and the second is the prediction error $e'(k)$ introduced by the prediction model. Given the predicted SCV for the next interval, we can now calculate the minimum number of sampling units that need to be sampled using Eq. (4).

C. AR Model with higher order interactions

Eq. (6) shows the simple AR model with an order u, where u signifies the number of past intervals used to predict the next interval. It is also possible that the regressor variables interact with each other i.e. the variable from one interval k has a more pronounced interaction with its immediately adjacent interval (k-1) than other intervals. In such cases, the model given by the simple AR model can be inefficient in capturing this relation. Hence, it becomes necessary to introduce terms that explicitly model these two-factor interactions. This can be introduced in the simple AR model in the following manner,

$$\boldsymbol{S''(k) = \sum_{i=1}^{u} a'(i).S'(k-i) + \sum_{i=1}^{u} \sum_{j=i+1}^{u} a'(i,j).S'(k-i).S'(k-j)} \quad .. (9)$$

where the first term, in Eq.(9), accounts for simple autoregression while the second term takes into account the second order interactions between two adjacent intervals, when calculating the predicted SCV for the next interval. In our empirical tests, we found that second order interactions were sufficient to achieve highly accurate predictions with minimal complexity overhead. Eq. (9) can be written in matrix terms as follows,

$$\boldsymbol{S'' = S'.\beta + \epsilon} \quad .. (10)$$

where β denotes the regression coefficients vector, S' denotes the model matrix and $\boldsymbol{\epsilon}$ is known as the residual error (due to lack of fit). The model matrix has columns corresponding to the regressor variables $S'_1, S'_2, \ldots, S'_{k-1}$, columns for the order of interaction, and a column of 1's defines the intercept. The coefficients can then be estimated by

$$\boldsymbol{\beta = (S'^T.S')^{-1}.S'^T.S''} \quad .. (11)$$

where suffix T denotes the transpose of matrix S'.

D. The AR Algorithm

In this section, we discuss the details of the AR algorithm. Figure 2 shows a flowchart of the AR framework.

1. We start by setting the error tolerance level (ε), the probability (η) of maintaining that level, and the AR model order (u). The size of the interval (m) can either be fixed or we can use the AR model to predict it as well.
2. Based on η and ε, z is calculated using Eq. (4).
3. To initialize the AR(u) model, we uniformly sample the first u intervals with an initial sample size *n_init* and collect the SCV values for those intervals.
4. Once the AR model has been initialized, the model predicts the SCV value of future intervals based on the past u intervals' SCV values and the 2nd order interactions between them given by Eq. (9).
5. With the predicted SCV value of the next interval, the minimum number of samples required for that interval is calculated using Eq.(4) i.e. $n' = z*S$.
6. The next interval is then periodically sampled with the resultant n samples and the sampled SCV of that interval is calculated. This sampled SCV is then used by the AR model to calculate the next interval's SCV along with past (u-*1*) intervals.

Once the sample size has been predicted for the next interval, we use a similar AR model to predict the size of the next interval $m(k)$. This prediction is made based on the past interval sizes seen by the AR(u) model. So, let $m(k)$ denote the size of the k^{th} interval, then using the AR(u) model, we have

$$\boldsymbol{m''(k) = \sum_{i=1}^{u} b'(i) m' + \sum_{i=1}^{u} \sum_{j=i+1}^{u} b'(i,j).m'(k-i).m'(k-j)} \quad ..(12)$$

where, $b'(i)$and $b'(i,j)$ are AR model parameters for first and second order interactions respectively and $m''(k)$ denotes the predicted interval size of the k^{th} interval. Using the above AR prediction model, we can then predict $m''(k)$.

Now that we have both the sample size (calculated from the predicted SCV) and interval size for the next interval, we can go ahead and periodically sample the next interval.

E. Constructing Confidence Interval from multiple intervals

In statistical sampling, confidence intervals give an estimated range of values which is likely to include an unknown population parameter . The standard formula to calculate a 100(1-α)% confidence interval for the sample mean of a normal population is given by,

$$\boldsymbol{X' \pm \left(\frac{z_{\alpha/2}.V.X'}{\sqrt{n}}\right)} \quad .. (13)$$

where X' is the sample mean, $z_{\alpha/2}$ is the critical value of the standard normal distribution, V is the sample variance and n is the sample size. The above formula works well when we randomly or systematically sample a normal population. However, in our case, we are dividing the population into intervals which become our new sub-populations. In order to calculate the variance of the samples taken from these sub-populations and then apply it towards our aggregate population, we need to calculate variance V in a manner similar to stratified sampling. Calculating variance in stratified sampling for all strata or sub-populations is given by,

$$\boldsymbol{V = \sum_{h=1}^{k} \frac{N_h^2}{N^2}.(1 - f_h).\frac{S_h^2}{n_h}} \quad .. (14)$$

where n_h is the sample size of the h^{th} strata or interval, N_h is the h^{th} interval size, N is the population size, $f_h = {}^{n_h}/_{N_h}$, and S_h^2 is the variance for that interval. The confidence interval for our aggregate population is then computed by substituting Eq.(14) into Eq.(13).

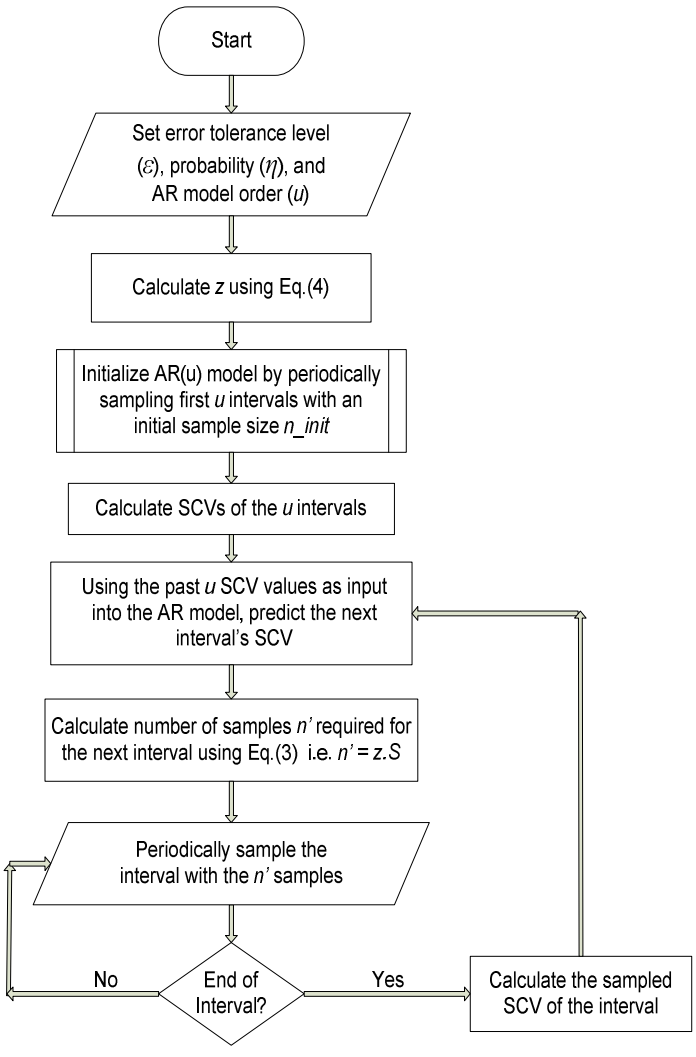

Figure 2: The AR framework Flowchart

IV. EXPERIMENT METHODOLOGY

In order to demonstrate the effectiveness of the AR framework, we tested the entire SPEC2K benchmark suite in our evaluation. We simulated CPI values obtained over small intervals of each benchmark using the *reference* input sets. Further, we implemented the AR framework in two modes: Mode-1 used the AR model (Eq.(9)) to predict the SCVs of the next interval alone while dividing the benchmarks into equal interval sizes. Mode-2 used the AR model to predict the SCVs of the next interval (Eq.(9)) and the size of that interval (Eq.(12)). Table I gives the various configurations analyzed in both Mode-1 and Mode-2 of the AR framework. Although we experimented with various model orders (u) and error bounds (ε), in this paper, we only present the results for $u=20$ and $\varepsilon=3\%$.

In our analysis, we also compare the AR framework with the widely used SMARTS framework [7]. The details of the microarchitecture configurations used for our evaluation are shown in Table II. The CPI data used, for both AR and SMARTS frameworks, has perfect warming of all architectural and microarchitectural states. Different warming methodologies can be used for both AR and SMARTS frameworks, which is outside the scope of this paper. Finally, we used a sample unit size of 1000 instructions for both the AR and SMARTS frameworks.

Table I: AR parameters for Mode-1 and Mode-2

Parameter	Mode – 1	Mode – 2
Model Order (u)	10, 15, 20	10, 15, 20
No. of Intervals (B)	Fixed (100, 1000, 10,000)	Predicted
Error Tolerance (ε)	20%, 10%, 3%	20%, 10%, 3%
Probability (η)	99.7%	99.7%

Table II: Processor Configuration used in simulation

Parameter	Config #1	Config #2
Decode, Issue, Commit Width	4-way	8-way
Branch Predictor, BHT Entries	Combined, 4K	Combined, 16K
ROB/LSQ Entries	32/16	128/64
Int/FP ALUs (Mult/Div Units)	2/2(1/1)	6/6(4/4)
L1-D Size (KB), Assoc, Lat(cycles)	32, 2-way, 1	128, 2-way, 1
L2 Size (KB), Assoc, Lat(cycles)	256, 4-way, 10	1024, 4-way, 15
Memory Lat(cycles)	150, 10	300, 20

V. AR FRAMEWORK EVALUATION AND RESULTS

Figure 3 shows the accuracy of the AR model in predicting the next interval SCV compared to the actual SCV of that interval. We can see that the predicted and actual SCVs of an interval overlap, which shows that the AR model is very successful in predicting future SCV values. In the remainder of this section, we discuss the effectiveness of the AR framework for CPI estimation for the SPEC2k benchmarks and the number of samples the AR framework needed. We also study the effects of varying the AR parameters used for both Mode-1 and Mode-2. At the end, we compare the AR and the SMARTS frameworks.

A. Mode–1: Predicting SCVs (S) with fixed intervals

In Mode-1 of the AR framework, we use the AR model to predict the SCV (S) of the next interval (k) while keeping the interval size (m) fixed. We periodically sampled 0.1% of each of the initial 'u' intervals as it empirically gave the best trade-off between sample sizes (n) taken versus error in CPI estimation (e). Figure 4 shows the number of samples taken for each SPEC2K benchmark to achieve an error bound of ±3% with 99.7% confidence. On average, the AR model sampled 40K sampling units or 40 million instructions for all SPEC2K benchmarks. Figure 5 shows the percentage error in the CPI estimation from using the AR framework. The error bars show the confidence intervals for each benchmark. We can see that all SPEC2K benchmarks stay well within 3% error bound with an average actual error of only 0.58%. Since actual errors are not a good estimate of how the framework performs, confidence intervals are also calculated. Our results show that the worse-case confidence interval for 99.7% confidence level is ±0.5%, while the average confidence interval for all SPEC2k benchmarks is ±0.2%.

B. Mode–2: Predicting Interval size and SCV (m, S)

The AR framework's Mode-2 uses two separate AR models; one to predict the interval size (m) and the other to predict the SCV (S) of that interval (k). Since we predict the interval size along with SCV, we also need to set initial interval sizes to warm the AR model. In our empirical tests, we found that an initial interval size of 0.5% of the total benchmark length, while periodically sampling 0.1% of the interval, gave the best balance between number of samples (n) and error (e). In Figure 4, we see that the most SPEC2k benchmarks required small sample sizes of 30k sampling units or 30 million instructions. The only exception is *176.gcc-1* on Config #2, which needed 172k samples to achieve a 99.7% confidence of ±3%. In Figure 5, we see that all SPEC2K benchmarks stay within 3% error with an average error of 0.7%. Confidence intervals for 99.7% confidence were calculated for all simulations and we noted that the confidence intervals achieved were ±0.5% in the worst case while the average converged around ±0.3%.

While not shown in the graphs, we observed that the most reasonable balance (between the number of samples taken (n) versus a corresponding low error (e) with tight confidence interval) was achieved with AR model order (u) set to 10 or 20 while setting the acceptable error bound (ε) at 3% for both AR modes. We also observed that the number of samples taken by either AR modes decreases linearly for larger error bounds (ε).

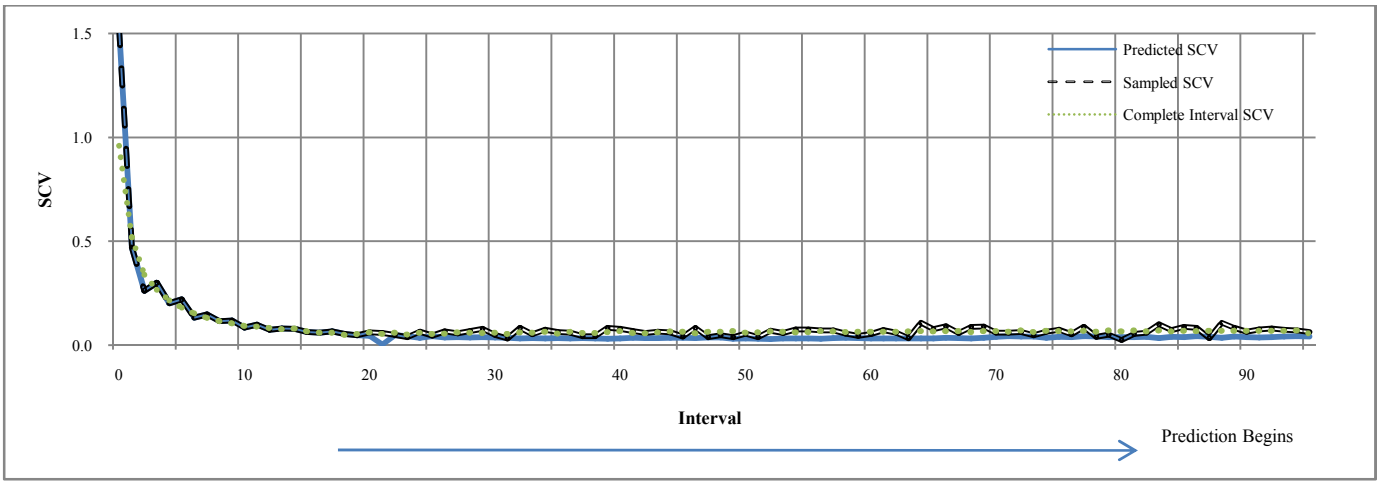

Figure 3: Predicted SCV vs. Sampled SCV vs. Actual SCV for SPEC2K benchmark *177.mesa*. AR Mode-1 ($u = 20$, $e = 3\%$, $B = 100$).

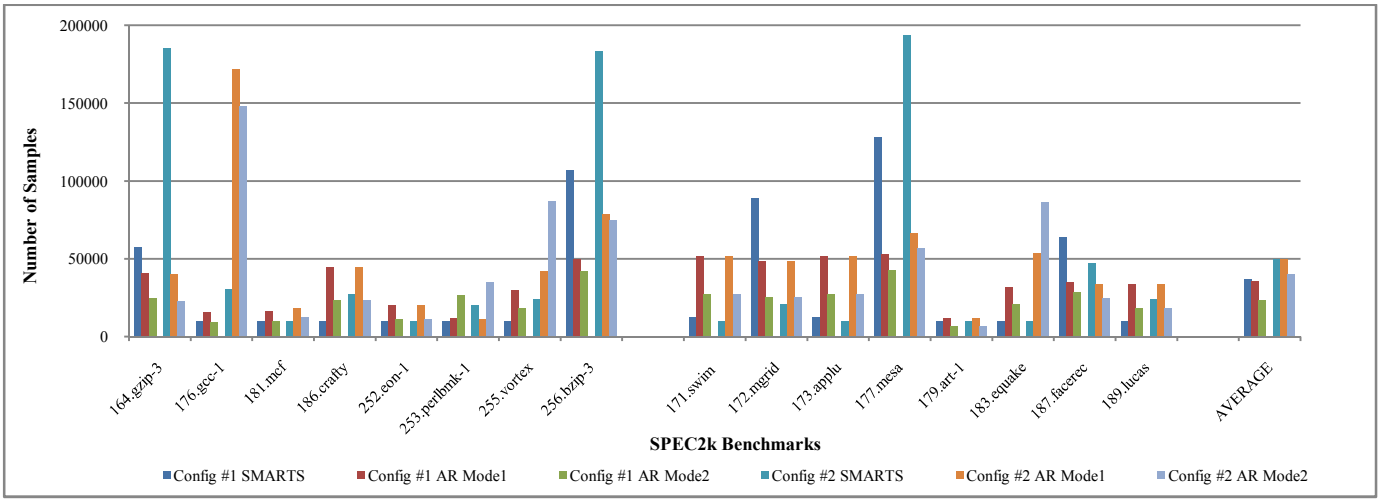

Figure 4: Number of samples taken by SMARTS vs. AR. SPEC2K Benchmarks for both Config #1 and #2. $u=20$, $\varepsilon=3\%$, $\eta = 99.7\%$.

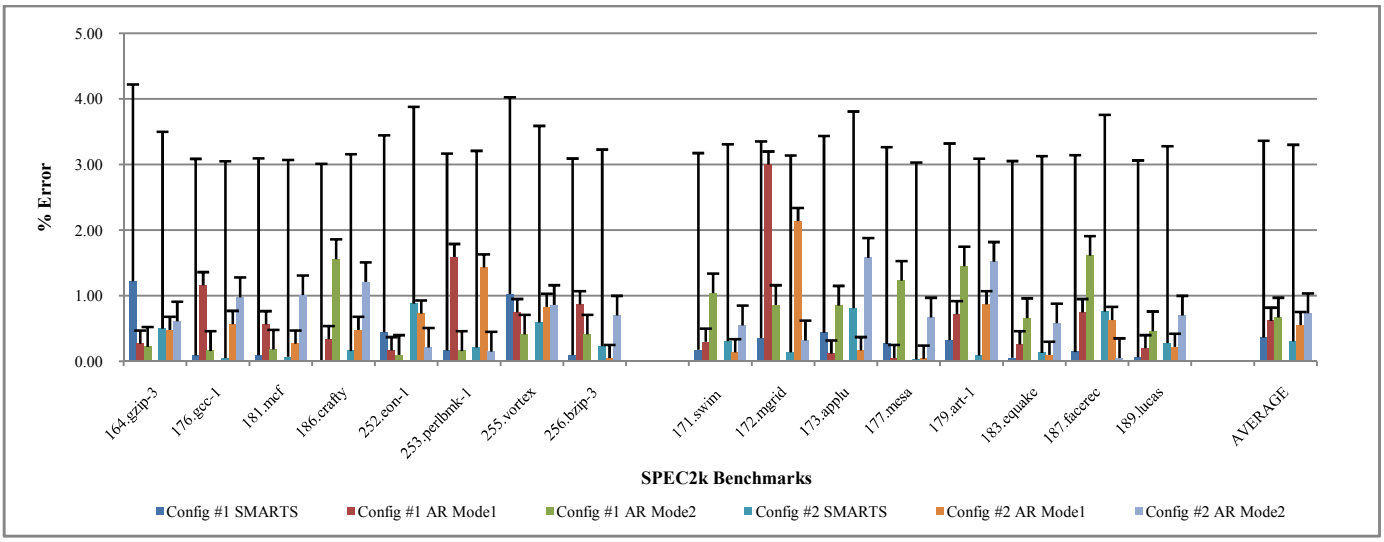

Figure 5: Error in CPI estimation for SMARTS vs. AR. SPEC2K Benchmarks for both Config #1 and #2. $u=20$, $\varepsilon=3\%$, $\eta= 99.7\%$. The error bar for each benchmark - configuration combination shows the confidence interval for that combination.

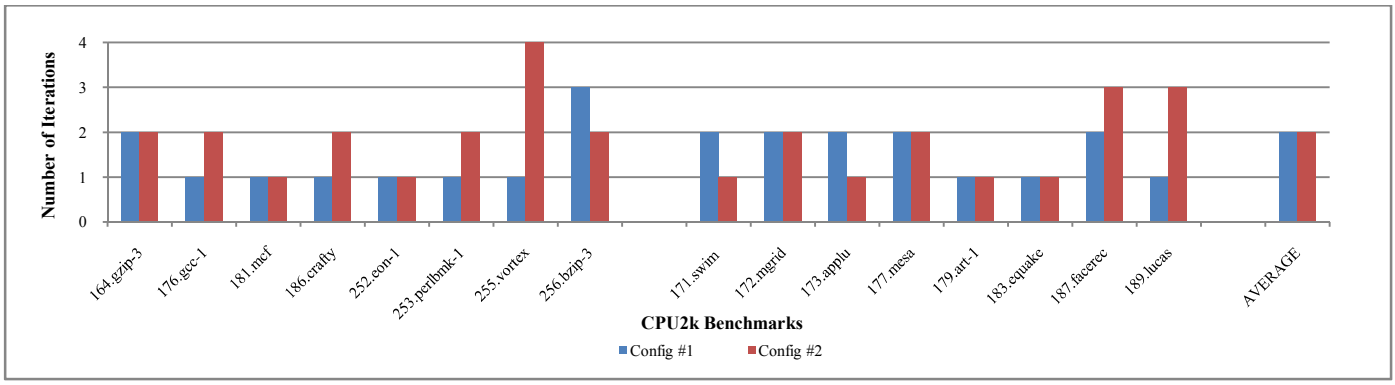

Figure 6: Number of Iterations taken by SMARTS for SPEC2K Benchmarks to achieve a ±3% confidence interval for both configurations (*n_init* = 10k).

C. *Consistency in CPI Estimation for Varying Processor Configurations*

In Figures 4 and 5, we have shown AR results for two different processor configurations. We see that the AR framework is, for most benchmarks, very consistent between the two configurations for both the number of samples (*n*) taken as well as the error in CPI estimation (*e*). The average difference in the number of samples taken is around 15k samples or 15 million instructions while the average difference in CPI error is observed to be only 0.07%.

D. *AR Modes versus SMARTS*

When we compare the two AR Modes, we see that, on average, both AR modes take similar number of samples for all SPEC2K benchmarks, although Mode-1 tends to take more samples than Mode-2. The reason behind this increased number of samples is that Mode-1 cannot intuitively vary the size of the interval (*m*) which, in turn, causes the increase in number of samples taken. Further, errors (*e*) for both modes stay within ±3%. Overall, Mode-2 gives the best trade-off between sample size (*n*) and error (*e*). Confidence intervals for Mode-1 stays within ±0.2% as against the ±0.3% range of Mode-2 which can be explained as an effect of Mode-1 taking more samples. In summary, both AR framework modes do extremely well in terms of number of samples taken and relative error in CPI estimation, while achieving a very tight confidence interval with a single iteration run.

Figures 4 and 5 also show the results of comparing the AR framework with SMARTS in terms of number of samples taken and error in CPI estimation. As we know, SMARTS iteratively samples a benchmark till the desired confidence is reached. A major disadvantage of SMARTS is that, most times, it ends up collecting redundant measurements over multiple iterations. This is evident from Figure 6 that shows the number of iterations taken by SMARTS to achieve a ±3% error with 99.7% confidence. Simulations were started with initial sample sizes '*n_init*' of 10000 sampling unit (SMARTS recommended) for both Config #1 and Config #2. From Figure 6 we see that 7 and 10 of the 16 benchmarks shown required two or more iterations to achieve the ±3% confidence interval for Config #1 and Config #2 respectively. This highlights the second disadvantage of using the SMARTS framework – researchers need to iteratively increase the number of samples per iteration, which can be a painstaking and cumbersome task. The AR framework, on the other hand, requires only a single run of the benchmarks to achieve a desired confidence. When comparing the number of samples taken by both AR and SMARTS, we see that both AR modes take comparable number of samples to SMARTS (Figure 4). Finally, while the error in CPI estimation for both AR modes and SMARTS stay well within ±3%, as shown in Figure 5, the confidence intervals achieved by AR framework, are ±0.5% in the worst case (while the average is within ±0.2%) in comparison to the ±3% achieved by SMARTS. Hence, the overall accuracy for the AR framework is better than that of SMARTS.

VI. Related Work

Reduced simulation time techniques such as population sampling and reduced input sets try to minimize the simulation time, but still achieve results that are representative of the complete execution of the program. One approach for finding representative samples is SimPoint [3,4,5], which uses basic block vectors combined with K-means [3,4] clustering to accurately identify similar intervals within a large application and strategically chooses sample intervals that are highly representative of the entire benchmark. Only simulating these sample intervals greatly reduces the simulation time, but still yields accurate results. The main disadvantage of SimPoint is that it requires an offline analysis before it can start detailed simulation. For benchmarks that take hours to run on a real machine or benchmarks that may be infrequently used, i.e. the cost of the profiling runs are amortized across just a few simulations, the cost of profiling may completely offset the reduction in simulation time. SMARTS [7] is another simulation-time reducing approach, which employs statistical sampling theory in order to select an appropriate number of program subsets representative of the entire program. This approach uses inferential statistics, which provides methods to estimate parameters of a large population from a representative subset. SMARTS employs systematic sampling while the most basic sample design is simple random sampling [6]. The main disadvantage of SMARTS is that it does not take advantage of the often repetitive behavior seen in benchmarks and hence can

end up collecting many redundant measurements. Another disadvantage of SMARTS is that, most times, the desired confidence is not achieved in the first simulation run which ultimately increase simulation time. In this paper, our main objective was to remove the need offline analysis or multiple runs to achieve a confidence level.

Reduced input sets [17,18] have two advantages over other simulation techniques. First, the benchmark runs to completion, which means that the simulator executes all pieces of the program. Second, using reduced input sets typically does not require any changes to the simulator. KleinOsowski and Lilja [17] created the MinneSPEC reduced input set for the SPEC2k benchmarks. The input sets were reduced by modifying command-line parameters, truncating the input set, or creating a completely new input set. Alameldeen *et al.* [18] created reduced input sets for TPC-C based on the metric of transaction throughput. The key conclusion from these two studies is that creating a reduced input sets is a very time-consuming and user-intensive task, especially so since reduced input sets need to be re-created for each new suite.

Other suggested approaches include using cluster sampling through predictive mechanisms for predicting future phases in a program's dynamic instruction stream [10], stratified sampling [11] with offline analysis [8] and a combination of both *i.e.* using phase-based small sample simulations [9]. Yi *et al.* [12] characterized and compared most of the aforementioned simulation techniques. Their results showed that, the truncated execution techniques and reduced input sets had very poor accuracy, as compared to the reference input set. Overall, both SimPoint and SMARTS are very accurate techniques.

VII. CONCLUSION

This paper presents a novel configuration-independent simulation-time reducing approach that adaptively samples intervals in a benchmark using an Autoregressive model (AR) on the Squared Coefficient of Variance (SCV) of the Cycles per Instruction (CPI) distribution. Using the SCV's of past intervals, the AR model calculates the required number of samples for the current interval. Unlike current reduced-time simulation techniques, the AR framework requires only a single run of the benchmark to achieve a desired confidence in Cycles per Instruction (CPI) estimation and requires no offline analysis. Our results show that the AR framework performs as well as SMARTS while providing more accurate results.

REFERENCES

[1] D. C. Burger and T. M. Austin. The SimpleScalar tool set, version 2.0. *Technical Report CS-TR-97-1342, University of Wisconsin, Madison, June 1997*.

[2] T. Sherwood and B. Calder. Time varying behavior of programs. *Technical Report UCSD-CS99-630, UC San Diego, August 1999*.

[3] T. Sherwood, E. Perelman, and B. Calder. Basic block distribution analysis to find periodic behavior and simulation points in applications. In *International Conference on Parallel Architectures and Compilation Techniques*, September 2001.

[4] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. Automatically characterizing large scale program behavior. In *10th International Conference on Architectural Support for Programming*, October 2002.

[5] E. Perelman, G. Hamerly, M. V. Biesbrouck, T. Sherwood, and B. Calder. Using SimPoint for Accurate and Efficient Simulation. *ACM SIGMETRICS the International Conference on Measurement and Modeling of Computer Systems*, June 2003.

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

[7] T. Wenisch, R. Wunderlich, B. Falsafi and J. Hoe, SMARTS: Accelerating Microarchitecture Simulation via Rigorous Statistical Sampling, In *Proceedings of the 31th Annual International symposium on Computer Architecture, IEEE Computer Society, 2003*.

[8] S. V. Kodakara, J. Kim, W. Hsu, D. J. Lilja, P. Yew. PASS: Program Structure Aware Stratified Sampling for Statistically selecting instruction traces and simulation points. *Technical report TR 05-044, University of Minnesota, December 2005*.

[9] J. L. Kihm, S. D. Strom, D. A. Connors. Phase-Guided Small-Sample Simulation. In *Proceedings of the International Symposium on Performance and Analysis of System Software*, April 2007

[10] E. Duesterwald, C. Cascaval, S. Dwarkadas, "Characterizing and Predicting Program Behavior and its Variability," *pact*, p. 220, *12th International Conference on Parallel Architectures and Compilation Techniques (PACT'03), 2003*.

[11] R.E. Wunderlich et al., "An Evaluation of Stratified Sampling of Microarchitecture Simulations," *3rd Ann. Workshop Duplicating, Deconstructing, and Debunking (WDDD 04)*, 2004, pp. 13–18; http:/www.ece.wisc.edu/~wddd/2004/WDDD2004_proceedings.pdf.

[12] J. J. Yi, S. V. Kodakara, R. Sendag, D. J. Lilja, and D. M. Hawkins, Characterizing and Comparing Prevailing Simulation Techniques, *IEEE International Symposium on High-Performance Computer Architecture (HPCA-11)*, pp. 266-277, Feb. 2005.

[13] B. Choi, J. Park, Z. Zhang, ``Adaptive Random Sampling for Traffic Load Measurement," *IEEE International Conference on Communications (ICC'03), Anchorage, Alaska, May, 2003*.

[14] D. A. Berry and B. W. Lindgren, "Statistics theory and Methods", *2nd ed., Duxbury Press, ITP, 1996*.

[15] J. M. Gottman, "Time-series analysis", *Cambridge UniversityPress, 1981*.

[16] G. Eshel, "The Yule-Walker Equations for AR coefficients", *http://minerva.simons-rock.edu/~geshel/geos31415/YW.pdf*

[17] A. KleinOsowski and D. Lilja, "MinneSPEC: A New SPEC Benchmark Workload for Simulation-Based Computer Architecture Research," Computer Architecture Letters, Vol. 1, June 2002.

[18] A. Alameldeen, M. Martin, C. Mauer, K. Moore, M. Xu, D. Sorin, M. Hill and D. Wood, "Simulating a $2M Commercial Server on a $2K PC," IEEE Computer, Vol. 36, No. 2, February 2003, Pages 50-57.