Accurate and Efficient Regression Modeling for Microarchitectural Performance and Power Prediction

Benjamin C. Lee, David M. Brooks

Division of Engineering and Applied Sciences
Harvard University
Cambridge, Massachusetts
{bclee,dbrooks}@eecs.harvard.edu

Abstract

We propose regression modeling as an efficient approach for accurately predicting performance and power for various applications executing on any microprocessor configuration in a large microarchitectural design space. This paper addresses fundamental challenges in microarchitectural simulation cost by reducing the number of required simulations and using simulated results more effectively via statistical modeling and inference.

Specifically, we derive and validate regression models for performance and power. Such models enable computationally efficient statistical inference, requiring the simulation of only 1 in 5 million points of a joint microarchitecture-application design space while achieving median error rates as low as 4.1 percent for performance and 4.3 percent for power. Although both models achieve similar accuracy, the sources of accuracy are strikingly different. We present optimizations for a baseline regression model to obtain (1) application-specific models to maximize accuracy in performance prediction and (2) regional power models leveraging only the most relevant samples from the microarchitectural design space to maximize accuracy in power prediction. Assessing sensitivity to the number of samples simulated for model formulation, we find fewer than 4,000 samples from a design space of approximately 22 billion points are sufficient. Collectively, our results suggest significant potential in accurate and efficient statistical inference for microarchitectural design space exploration via regression models.

Categories and Subject Descriptors B.8.2 [*Performance Analysis and Design Aids*]; I.6.5 [*Model Development*]: Modeling Methodologies

General Terms Design, Experimentation, Measurement, Performance

Keywords Microarchitecture, Simulation, Statistics, Inference, Regression

1. Introduction

Efficient design space exploration is constrained by the significant computational costs of cycle-accurate simulators. These simulators provide detailed insight into application performance for a wide range of microprocessor configurations, exposing performance and power trends in the microarchitectural design space. Long simulation times require designers to constrain design studies and consider only small subsets of the full design space. However, such constraints may lead to conclusions that may not generalize to the larger space. Addressing these fundamental challenges in microarchitectural simulation methodology becomes increasingly urgent as chip multiprocessors introduce additional design parameters and exponentially increase design space size.

We apply regression modeling to derive simulation-free statistical inference models, requiring a small number of sampled design points in a joint microarchitecture-application design space for initial simulation. Such an approach modestly reduces detail in return for significant gains in speed and tractability. Although we consider advantages in computational cost for microarchitectural design space exploration in this paper, these models may also provide increased profiling efficiency and fast performance prediction for system software and algorithms, such as thread-scheduling for heterogeneous multi-processors.

Techniques in statistical inference and machine learning are increasingly popular for approximating solutions to intractable problems. Even for applications in which obtaining extensive measurement data is feasible, efficient analysis of this data often lends itself to statistical modeling. These approaches typically require an initial set of data for model formulation or training. The model responds to predictive queries by leveraging trends and correlations in the original data set to perform statistical inference. Regression modeling follows this predictive paradigm in a relatively cost effective manner. Once domain-specific knowledge is used to specify predictors of a response, formulating the model from observed data requires numerically solving a system of linear equations and predicting the response simply requires evaluating a linear equation. Model formulation and evaluation are computationally efficient due to well optimized numerical linear algebra libraries.

We use a modest number of simulations to obtain sample observations from a large design space. In Section 3, we describe a sampling methodology to obtain 4,000 samples drawn uniformly at random from a design space with approximately 22 billion points. Each sample maps a set of architectural and application-specific predictors to observed simulator performance and power. These samples are used to formulate regression models that predict the performance and power of previously unsampled configurations based on the same predictors.

In Section 4, we summarize a statistically rigorous approach for deriving regression models that includes (1) variable clustering, (2) association testing, (3) assessing strength of response-predictor relationships, and (4) significance testing with F-tests. These tech-

ASPLOS'06 October 21–25, 2006, San Jose, California, USA.

niques ensure statistically significant architectural and application-specific parameters are used as predictors. Given a baseline model that accounts for predictor interaction and non-linearity, Section 5 presents model optimizations to improve prediction accuracy by (1) stabilizing residual variance, (2) deriving application-specific models, and (3) deriving regional models with samples most similar in architectural configuration to the predictive query.

The following summarizes experimental results of Section 5 from four different performance and power regression models formulated with 4,000 samples drawn from a joint microarchitecture-application design space with nearly 1 billion microarchitectural configurations and 22 benchmarks:

1. **Performance Prediction:** Application-specific models predict performance with median error as low as 4.1 percent (mean error as low as 4.9 percent). 50 to 90 percent of predictions achieve error rates of less than 10 percent depending on the application. Maximum outlier error is 20 to 33 percent.
2. **Power Prediction:** Regional models predict power with median error as low as 4.3 percent (mean error as low as 5.6 percent). Nearly 90 percent of predictions achieve error rates of less than 10 percent and 97 percent of predictions achieve error rates of less than 15 percent. Maximum outlier error is 24.5 percent.
3. **Model Optimizations:** Given a single set of predictors for both performance and power models, the model may be reformulated with different sampled observations and optimized. Application-specific models are optimal for performance prediction while regional models are optimal for power prediction.
4. **Sample Size Sensitivity:** Although 4,000 samples are drawn from the design space, accuracy maximizing applications-specific performance models do not require more than 2,000. Additional samples may improve regional power models by improving observation density and tightening regions around predictive queries, but diminishing marginal returns in accuracy advise against many more than 3,000 samples.

Collectively, these results suggest significant potential in accurate, efficient statistical inference for the microarchitectural design space via regression models. We provide an overview of the model derivation and a detailed validation of its predictive ability. Previous work details the model derivation [8, 9].

2. Regression Theory

We apply regression modeling to efficiently obtain estimates of microarchitectural design metrics, such as performance and power. We apply a general class of models in which a response is modeled as a weighted sum of predictor variables plus random noise. Since basic linear estimates may not adequately capture nuances in the response-predictor relationship, we also consider more advanced techniques to account for potential predictor interaction and non-linear relationships. Lastly, we present standard statistical techniques for assessing model effectiveness and predictive ability.

2.1 Model Formulations

For a large universe of interest, suppose we have a subset of n observations for which values of the response and predictor variables are known. Let $y = y_1, \ldots, y_n$ denote the vector of observed responses. For a particular point i in this universe, let y_i denote its response variable and $x_i = x_{i,1}, \ldots, x_{i,p}$ denote its p predictors. These variables are constant for a given point in the universe. Let $\beta = \beta_0, \ldots, \beta_p$ denote the corresponding set of regression coefficients used in describing the response as a linear function of predictors plus a random error e_i as shown in Equation (1). Mathematically, β_j may be interpreted as the expected change in y_i per unit change in the predictor variable $x_{i,j}$. The e_i are independent random variables with zero mean and constant variance; $E(e_i) = 0$ and $Var(e_i) = \sigma^2$. We consider a joint microarchitecture-application universe with design metrics of interest as response variables predicted by microarchitectural configurations and application characteristics.

$$\begin{aligned} f(y_i) &= g(x_i)\beta + e_i \\ &= \beta_0 + \sum_{j=1}^{p} \beta_j g_j(x_{ij}) + e_i \end{aligned} \quad (1)$$

Fitting a regression model to observations, by determining the $p+1$ coefficients in β, enables response prediction. The *method of least squares* is commonly used to identify the best-fitting model by minimizing $S(\beta)$, the sum of squared deviations of predicted responses given by the model from actual observed responses.

$$S(\beta_0, \ldots, \beta_p) = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \quad (2)$$

$S(\beta)$ may be minimized by solving a system of $p+1$ partial derivatives of S with respect to β_j, $j \in [0, p]$. The solutions to this system are estimates of the coefficients in Equation (1). Furthermore, solutions to this system of linear equations may often be expressed in closed form. Closed form expressions enable using the statistical properties of these estimates to identify the significance of particular response-predictor correlations (Section 2.4).

2.2 Predictor Interaction

In some cases, the effect of predictors $x_{i,1}$ and $x_{i,2}$ on the response cannot be separated; the effect of $x_{i,1}$ on y_i depends on the value of $x_{i,2}$ and vice versa. This interaction may be modeled by constructing a third predictor $x_{i,3} = x_{i,1}x_{i,2}$ to obtain $y_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \beta_3 x_{i,1} x_{i,2} + e_i$. For example, pipeline depth and L2 cache size do not impact performance independently; a smaller L2 cache leads to additional memory hazards in the pipeline. These stalls affect, in turn, instruction throughput gains from pipelining. Thus, their joint impact on performance must also be modeled.

Modeling predictor interactions in this manner makes it difficult to interpret β_1 and β_2 in isolation. After simple algebraic manipulation to account, we find $\beta_1 + \beta_3 x_{i,2}$ is the expected change in y_i per unit change in $x_{i,1}$ for a fixed $x_{i,2}$. The difficulties of these explicit interpretations of β for more complex models lead us to prefer more indirect interpretations of the model via its predictions.

2.3 Non-Linearity

Basic linear regression models assume the response behaves linearly in all predictors. This assumption is often too restrictive (*e.g.*, power increases quadratically with pipeline depth for more aggressive designs) and several techniques for capturing non-linearity may be applied. The most simple of these techniques is a polynomial transformation on predictors suspected of having a non-linear correlation with the response. However, polynomials have undesirable peaks and valleys. Furthermore, a good fit in one region of the predictor's values may unduly impact the fit in another region of values. For these reasons, we consider splines a more effective technique for modeling non-linearity.

Spline functions are piecewise polynomials used in curve fitting. The function is divided into intervals defining multiple different continuous polynomials with endpoints called *knots*. The number of knots can vary depending on the amount of available data for

fitting the function, but more knots generally leads to better fits. Relatively simple linear splines may be inadequate for complex, highly curved relationships. Splines of higher order polynomials may offer better fits and cubic splines have been found particularly effective [4]. Unlike linear splines, cubic splines may be made smooth at the knots by forcing the first and second derivatives of the function to agree at the knots. However, cubic splines may have poor behavior in the tails before the first knot and after the last knot [13]. Restricted cubic splines that constrain the function to be linear in the tails are often better behaved and have the added advantage of fewer terms relative to cubic splines.

The choice and position of knots are variable parameters when specifying non-linearity with splines. Stone has found the location of knots in a restricted cubic spline to be much less significant than the number of knots [13]. Placing knots at fixed quantiles of a predictor's distribution is a good approach in most datasets, ensuring a sufficient number of points in each interval.

In practice, five knots or fewer are generally sufficient for restricted cubic splines [13]. Fewer knots may be required for small data sets. As the number of knots increases, flexibility improves at the risk of over-fitting the data. In many cases, four knots offer an adequate fit of the model and is a good compromise between flexibility and loss of precision from over-fitting [4]. For larger data sets with more than 100 samples, five knots may also be a good choice.

2.4 Significance Testing

Although T-tests are often used to assess the significance of individual terms, it is often more useful to assess a group of terms simultaneously. Consider a model $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 + e$. Testing the significance of x_1 requires testing the null hypothesis $H_0 : \beta_1 = \beta_3 = 0$ with two degrees of freedom. More generally, given two nested models, the null hypothesis states the additional predictors in the larger model have no association with the response. For example, suppose x_1 and x_2 are pipeline depth and width, respectively. We must then compare a model without depth(x_1) and its width interaction($x_1 x_2$) against the full model to assess depth's significance.

The *F-test* is a standard statistical test for comparing two nested models using their multiple correlation statistic, R^2. Equation (3) computes this statistic as regression error (SSE) relative to total error (SST). R^2 will be zero when the error from the regression model is just as large as the error from simply using the mean to predict the response. Thus, R^2 is the percentage of variance in the response variable captured by the predictor variables.

$$\begin{aligned} R^2 &= 1 - \frac{SSE}{SST} \\ &= 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \frac{1}{n}\sum_{i=1}^{n} y_i)^2} \end{aligned} \tag{3}$$

Given R^2 for the full model and R^2_* for a model constructed by dropping terms from the full model, define the *F-statistic* by Equation (4) where p is the number of coefficients in the full model excluding the intercept β_0 and k is the difference in degrees of freedom between the models.

$$F_{k,n-p-1} = \frac{R^2 - R^2_*}{k} \times \frac{n-p-1}{1-R^2} \tag{4}$$

The *p-value* is defined as $2P(X \geq |c|)$ for a random variable X and a constant c. In our analyses, X follows the F distribution with parameters k, $n-p-1$ and c is the F-statistic. The p-value may be interpreted as the probability a F-statistic value greater than or equal to the value actually observed would occur by chance if the null hypothesis were true. If this probability were extremely small, either the null hypothesis holds and an extremely rare event has occurred or the null hypothesis is false. Thus, a small p-value for for a F-test of two models casts doubt on the null hypothesis and suggests the additional predictors in the larger model are statistically significant in predicting the response.

2.5 Assessing Fit

The model's fit to the observations used to formulate the model measures how well the model captures observed trends. Fit is usually assessed by examining residuals and the degree to which regression error contributes to the total error. Residuals, defined in Equation (5), are examined to validate three assumptions: (1) the residuals are not correlated with any predictor variable or the response predicted by the model, (2) the randomness of the residuals is the same for all predictor and predicted response values, and (3) the residuals have a normal distribution. The first two assumptions are typically validated by plotting residuals against each of the predictors and predicted responses (*i.e.*, $(\hat{e}_i, x_{ij})$ for each $j \in [1,p]$ and $(\hat{e}_i, \hat{y}_i)$ for each $i \in [1,n]$) since such plots may reveal systematic deviations from randomness. The third assumption is usually validated by a quantile-quantile plot in which the quantiles of one distribution are plotted against another. Practically, this means ranking the residuals $\hat{e}^{(1)}, \ldots, \hat{e}^{(n)}$, obtaining n ranked samples from the normal distribution $s^{(1)}, \ldots, s^{(n)}$, and producing a scatter plot of $(\hat{e}^{(i)}, s^{(i)})$ that should appear linear if the residuals follow a normal distribution.

$$\hat{e}_i = y_i - \hat{\beta}_0 - \sum_{j=0}^{p} \hat{\beta}_j x_{ij} \tag{5}$$

Fit may also be assessed by the R^2 statistic where a larger R^2 suggests better fits for the observed data. However, a value too close to $R^2 = 1$ may indicate over-fitting, a situation in which the worth of the model is exaggerated and future observations will not agree with the model's predicted values. Over-fitting typically occurs when too many predictors are used to model relatively small data sets. A regression model is likely reliable when the number of predictors p is less than $n/20$, where n is the sample size [4].

2.6 Prediction

Once β is determined, evaluating Equation (6) for a given x_i will give the expectation of y_i and, equivalently, an estimate $\hat{y}_i$ for y_i. This result follows from observing the additive property of expectations, the expectation of a constant is the constant, and the random errors have mean zero.

$$\begin{aligned} \hat{y}_i &= E\big[y_i\big] \\ &= E\big[\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij}\big] + E\big[e_i\big] \\ &= \beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} \end{aligned} \tag{6}$$

3. Experimental Methodology

Before developing regression models to predict performance and power for any configuration within the microarchitectural design space, we must first obtain a number of observations within this space via simulation. These observations are inputs to a statistical computing package used to perform the regression analysis and formulate the models.

	Set	Parameters	Measure	Range	$\|S_i\|$
S_1	Depth	Depth	FO4	9::3::36	10
S_2	Width	Width	insn b/w	4,8,16	3
		L/S Reorder Queue	entries	15::15::45	
		Store Queue	entries	14::14::42	
		Functional Units	count	1,2,4	
S_3	Physical Registers	General Purpose (GP)	count	40::10::130	10
		Floating-Point (FP)	count	40::8::112	
		Special Purpoes (SP)	count	42::6::96	
S_4	Reservation Stations	Branch	entries	6::1::15	10
		Fixed-Point/Memory	entries	10::2::28	
		Floating-Point	entries	5::1::14	
S_5	I-L1 Cache	I-L1 Cache Size	log_2(entries)	7::1::11	5
S_6	D-L1 Cache	D-L1 Cache Size	log_2(entries)	6::1::10	5
S_7	L2 Cache	L2 Cache Size	log_2(entries)	11::1::15	5
		L2 Cache Latency	cycles	6::2::14	
S_8	Main Memory	Main Memory Latency	cycles	70::5::115	10
S_9	Control Latency	Branch Latency	cycles	1,2	2
S_{10}	Fixed-Point Latency	ALU Latency	cycles	1::1::5	5
		FX-Multiply Latency	cycles	4::1::8	
		FX-Divide Latency	cycles	35::5::55	
S_{11}	Floating-Point Latency	FPU Latency	cycles	5::1::9	5
		FP-Divide Latency	cycles	25::5::45	
S_{12}	Memory Latency	Load/Store Latency	cycles	3::1::7	5

Table 1. Parameters within a group are varied together. A range of $i::j::k$ denotes a set of possible values from i to k in steps of j.

3.1 Simulation Framework

We use Turandot, a generic and parameterized, out-of-order, superscalar processor simulator [10]. Turandot is enhanced with PowerTimer to obtain power estimates based on circuit-level power analyses and resource utilization statistics [1]. The modeled baseline architecture is similar to the POWER4/POWER5. The simulator has been validated against both a POWER4 RTL model and a hardware implementation. We do not use any particular feature of the simulator in our models and believe our approach may be generally applied to other simulatorx frameworks. We evaluate performance in billions of instructions per second (bips) and power in watts.

3.2 Benchmarks

We consider SPECjbb, a Java server benchmark, and 21 compute intensive benchmarks from SPEC2k (ammp, applu, apsi, art, bzip2, crafty, equake, facerec, gap, gcc, gzip, lucas, mcf, mesa, mgrid, perl, sixtrack, swim, twolf, vpr, wupwise). We report experimental results based on PowerPC traces of these benchmarks. The SPEC2k traces used in this study were sampled from the full reference input set to obtain 100 million instructions per benchmark program. Systematic validation was performed to compare the sampled traces against the full traces to ensure accurate representation [5].

3.3 Statistical Analysis

We use R, a free software environment for statistical computing, to script and automate the statistical analyses described in Section 2. Within this environment, we use the Hmisc and Design packages implemented by Harrell [4].

3.4 Configuration Sampling

Table 1 identifies twelve groups of parameters varied simultaneously. Parameters within a group are varied together in a conservative effort to avoid fundamental design imbalances. The range of values considered for each parameter group is specified by a set of values, $S_1, \ldots, S_{12}$. The Cartesian product of these sets, $S = \prod_{i=1}^{12} S_i$, defines the entire design space. The cardinality of this product is $|S| = \prod_{i=1}^{12} |S_i| = 9.38E + 08$, or approximately one billion, design points. Fully assessing the performance for each of the 22 benchmarks on these configurations further scales the number of simulations to well over 20 billion.

The approach to obtaining observations from a large microprocessor design space is critical to efficient formulation of regression models. Traditional techniques of sweeping design parameter values to consider all points in the large design space, S, is impossible despite continuing research in reducing simulation costs via trace sampling [12, 14]. Although these techniques reduce per simulation costs by a constant factor, they do not reduce the number of required simulations. Other studies have reduced the cardinality of these sets to an upper and lower bound [6, 15], but this approach masks performance and power trends between the bounds and precludes any meaningful statistical inference. For these reasons, sampling must occur in the design space to control the exponentially increasing number of design points as the number of parameter sets and their cardinalities increase.

We propose sampling configurations *uniformly at random* (UAR) from S. This approach provides observations from the full range of parameter values and enables identification of trends and trade-offs between the parameter sets. We can include an arbitrarily large number of values into a given S_i since we decouple the number of simulations from the set cardinality via random sampling. Furthermore, sampling UAR does not bias the observations toward particular configurations. Prior design space studies have considered points around a baseline configuration and may be biased toward the baseline. Our approach is different from Monte Carlo methods, which generate suitable random samples and observe the frequency of samples following some property or properties. Although we perform random sampling, we formulate regression models instead of analyzing frequency.

We report experimental results for sample sizes of up to $n = 4,000$ samples. Each sampled configuration is simulated with a benchmark also chosen UAR, providing one set of observed responses (performance and power) for every 5 million sets of predictors (configuration-application pairs) in the design space.

Performance	
bips	
L1 Cache	
I-L1 misses	D-L1 misses
I-L2 misses	D-L2 misses
Branches	
branch rate	branch stalls
branch mispredictions	
Stalls	
inflight	cast
dmissq	reorderq
storeq	rename
resv	

Table 2. Measured application characteristics on a baseline architecture.

Processor Core	
Decode Rate	4 non-branch insns/cy
Dispatch Rate	9 insns/cy
Reservation Stations	FXU(40),FPU(10),LSU(36),BR(12)
Functional Units	2 FXU, 2 FPU, 2 LSU, 2 BR
Physical Registers	80 GPR, 72 FPR
Branch Predictor	16k 1-bit entry BHT
Memory Hierarchy	
L1 DCache Size	32KB, 2-way, 128B blocks, 1-cy lat
L1 ICache Size	32KB, 1-way, 128B blocks, 1-cy lat
L2 Cache Size	2MB, 4-way, 128B blocks, 9-cy lat
Memory	77-cy lat
Pipeline Dimensions	
Pipeline Depth	19 FO4 delays per stage
Pipeline Width	4-decode

Table 3. Baseline architecture.

4. Model Derivation

To identify potential response-predictor relationships we first examine a number of descriptive statistics for parameters in the data set. We then formulate a regression model, accounting for predictors' primary effects, second- and third-order interactions, and non-linearities. Given an initial model, we perform significance testing to prune statistically insignificant predictors from the model. The fit of the refined model is assessed by examining its residuals.

4.1 Predictors

In addition to the 12 architectural predictors (Table 1), we also consider 15 application-specific predictors drawn from an application's characteristics (Table 2) when executing on a baseline configuration (Table 3). These characteristics may be significant predictors of performance when interacting with architectural predictors and include baseline performance (base_bips), cache access patterns, branch patterns, and sources of pipeline stalls (*e.g.* stalls from limits on the number of inflight instructions). For example, the performance effect of increasing the data L1 cache size will have a larger impact on applications with a high data L1 miss rate. The baseline application performance may also impact the performance effects of further increasing architectural resources. An application with no bottlenecks and high baseline performance would benefit less from additional registers compared to an application experiencing heavy register pressure. These potential interactions suggest both architectural and application-specific predictors are necessary.

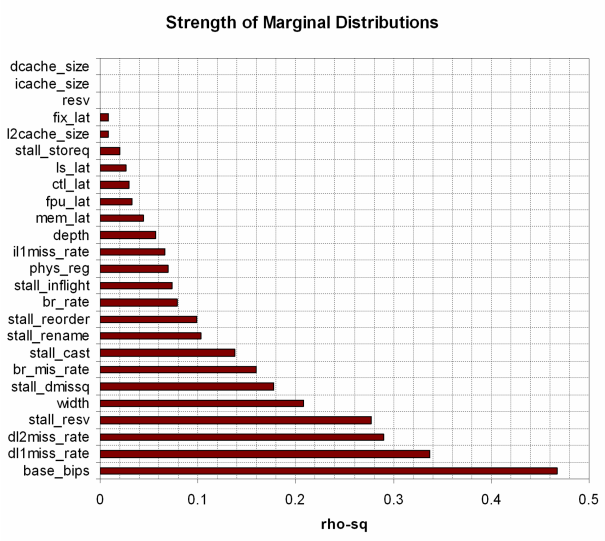

Figure 1. Predictor Strength

4.2 Summary Statistics

We provide a brief overview of analyses used to identify relevant predictors for an initial performance regression model. The associated figures are omitted due to space constraints. Details are available in a technical report [8].

4.2.1 Variable Clustering

A variable clustering analysis reveals key parameter interaction by computing squared correlation coefficients as similarity measures. A larger ρ^2 suggests a greater correlation between variables. If overfitting is a concern, redundant predictors may be eliminated by selecting only one predictor from each cluster.

Our clustering analysis indicates L1 and L2 misses due to instruction cache accesses are highly correlated. We find the absolute number of L2 cache misses from the instruction cache to be negligible and eliminate `il2miss_rate`. Similarly, the branch rate is highly correlated with the number of branch induced stalls and we eliminate `br_stall`. All other variables are correlated to a lesser degree and need not be eliminated from consideration. We find the number of observations sufficiently large to avoid over-fitting.

4.2.2 Performance Associations

Plotting each of the predictors against the response may reveal particularly strong associations or identify non-linearities. For architectural predictors, we find pipeline depth and width strong, monotonic factors. The number of physical registers may be a significant, but non-linear, predictor. Correlations between L2, but not L1, cache size and performance also appear significant. For application-specific predictors, data cache access patterns may be good predictors of performance. We also find roughly half the stall characteristics have monotonic relationships with performance (*i.e.*, `dmissq`, `cast`, `reorderq`, `resv`). A benchmark's observed sample and baseline performance are highly correlated.

4.2.3 Strength of Marginal Relationships

We consider the squared correlation coefficient between each predictor variable and observed performance in Figure 1 to choose the number of spline knots. A lack of fit for predictors with higher ρ^2 will have a greater negative impact on performance prediction. For architectural predictors, a lack of fit will be more consequential (in

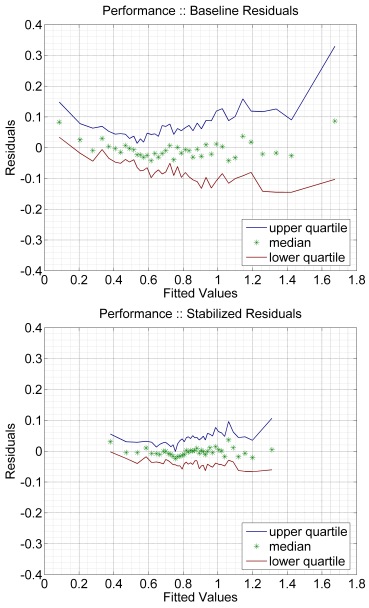

Figure 2. Residual Correlations

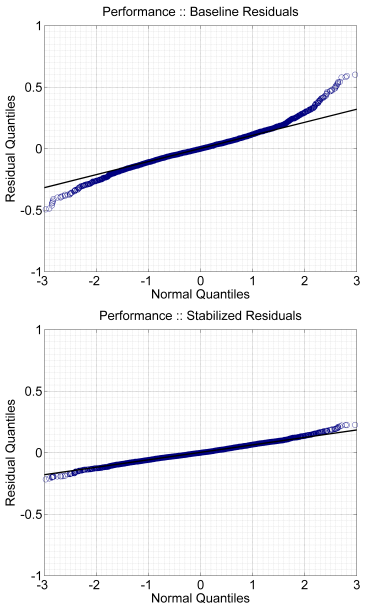

Figure 3. Residual Distribution

descending order of importance) for width, depth, physical registers, functional unit latencies, cache sizes, and reservation stations. An application's baseline performance is the most significant predictor for its performance on other configurations.

Overall, application-specific predictors are most highly correlated with observed performance. An application's interaction with the microarchitecture, not the microarchitecture itself, is the primary determinant of performance. For example, the degree to which an application is memory bound, captured by baseline cache miss rates, are much more highly correlated with performance than the cache sizes.

4.3 Model Specification and Refinement

After a first pass at removing insignificant predictors, we formulate an initial performance regression model. We model non-linearities for architectural predictors using restricted cubic splines. As shown in Figure 1, the strength of predictors' marginal relationships with performance will guide our choice in the number of knots. Predictors with stronger relationships and greater correlations with performance will use 4 knots (*e.g.* depth, registers) and those with weaker relationships will use 3 knots (*e.g.* latencies, cache sizes, reservation stations). Despite their importance, width and certain latencies do not take a sufficient number of unique values to apply splines and we consider their linear effects only. With the exception of baseline performance, for which we assign 5 knots, we do not model non-linearities for application-specific predictors to control model complexity.

We draw on domain-specific knowledge to specify interactions. We expect pipeline width to interact with register file and queue sizes. Pipeline depth likely interacts with cache sizes that impact hazard rates. We also expect the memory hierarchy to interact with adjacent levels (*e.g.* L1 and L2 cache size interaction) and application-specific access rates. Interactions with baseline performance account for changing marginal returns in performance from changing resource sizes. Although we capture most relevant interactions, we do not attempt to capture all significant interactions via an exhaustive search of predictor combinations. The results of Section 5 suggest such a high-level representation is sufficient.

Figure 2 plots residual quartiles for 40 groups, each with 100 observations, against median modeled performance of each group, revealing significant correlations between residuals and fitted values.[1] Residuals are larger for the smallest and largest fitted values. We apply a standard variance stabilizing square root transformation on performance to reduce the magnitude of the correlations as shown in Figure 2. The resulting model predicts the square-root of performance (*i.e.* $\sqrt{y}$ instead of y). Variance stabilization also cause residuals to follow a normal distribution more closely as indicated by the linear trend in Figure 3.

To further improve model efficiency, we consider each predictor's contribution to its predictive ability. We assess the significance of predictor p by comparing, with F-tests, the initial model to a smaller model with all terms involving p removed [8]. Although we found most variables significant with p-values less than $2.2e-16$, `br_lat` (p-value=0.1247), `stall_inflight` (p-value=0.7285), and `stall_storeq` (p-value=0.4137) appear insignificant. High p-values indicate these predictors do not significantly contribute to a better fit when included in a larger model.

5. Model Evaluation

5.1 Model Variants and Optimizations

We use data sets of varying size drawn from $n_* < n = 4,000$ random observations to formulate regression models. We refer to n_* as the *sample size* for a particular model; each model may require different sample sizes to maximize accuracy. We arbitrarily chose 4,000 points for the initial data set, reflecting our lack of prior intuition regarding the number of samples required for accurate modeling. We subsequently assess model sensitivity to sample size, demonstrating $n_* < 2,000$ samples are sufficient.

Separately, we obtain 100 additional random samples for predictive queries and validation against the simulator. We compare the predictive ability of four regression models, differing in specification and data used to perform the fit:

[1] Residuals are defined in Equation (5)

Model	Min	1st Quartile	Median	Mean	3rd Quartile	Max
B (1k)	0.571	7.369	13.059	16.359	20.870	56.881
S (2k)	0.101	5.072	10.909	13.015	17.671	51.198
S+R (1k)	0.360	4.081	8.940	10.586	15.183	35.000
S+A (1k,ammp)	0.029	1.815	4.055	4.912	7.318	20.298
S+A (1k,equake)	0.181	4.132	7.385	8.064	11.202	20.825
S+A (1k,mesa)	0.170	5.736	10.775	10.810	15.025	33.129

Table 4. Summary of performance prediction error with specified error minimizing sample and region sizes.

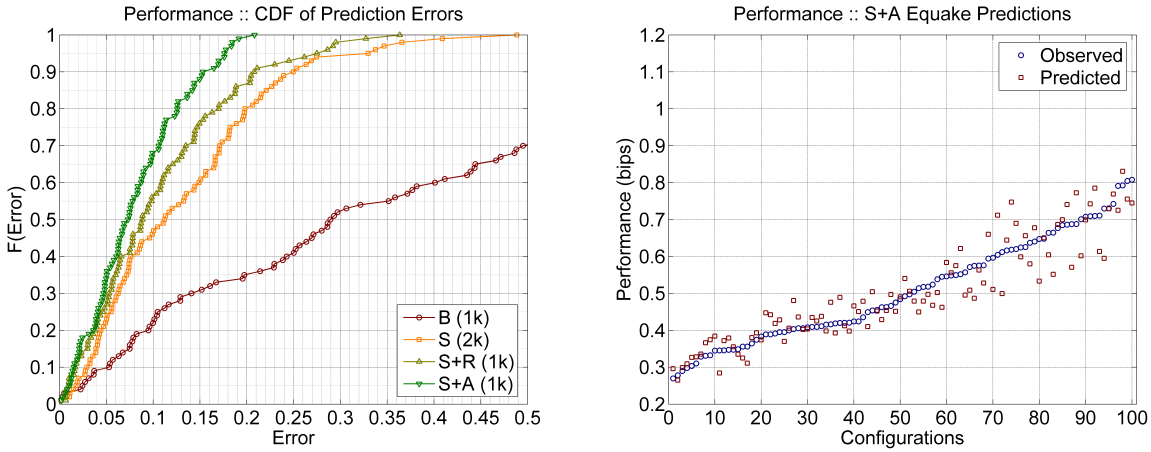

Figure 4. Empirical CDF of prediction error with error minimizing sample and region sizes. Equake achieves median accuracy and is plotted for S+A (L). Prediction results for S+A (R).

- **Baseline (B):** Model specified in Section 4 without variance stabilization and formulated with a naive subset of $n_B < n$ observations (*e.g.* first 1,000 obtained).
- **Variance Stabilized (S):** Model specified with a square-root transformation on the response and formulated with a naive subset of $n_S < n$ observations.
- **Regional (S+R):** Model is reformulated for each query by specifying a naive subset of $n_{S+R} < n$ observations. These observations are further reduced to include the $r_{S+R} < n_{S+R}$ designs with microarchitectural configurations most similar to the predictive query. We refer to r_{S+R} as the *region size*. Similarity is quantified by the normalized euclidean distance between two vectors of architectural parameter values, $d = \sqrt{\sum_{i=1}^{p} |1 - b_i/a_i|^2}$.
- **Application-Specific (S+A):** We use a *new* set of $n_A = 4,000$ of observations for varying microarchitectural configurations, but a fixed benchmark. An application-specific model is obtained by eliminating application-specific predictors from the general model and reformulating the model with a naive subset of $n_{S+A} < n_A$. We consider S+A models for six benchmarks: ammp, applu, equake, gcc, gzip, and mesa.

5.2 Performance Prediction

Table 4 summarizes model predictive accuracy in terms of percentage error, $100 * |\hat{y}_i - y_i| / y_i$. Each model is presented with error minimizing sample and region sizes. We present the ammp, equake, and mesa application-specific models that achieve the greatest, median, and least accuracy, respectively. Variance stabilization reduces median error from 13.1 to 10.9 percent and regional or application-specific models may further reduce median error from 10.9 to 8.9 or 4.1 percent, respectively. Application-specific models predict performance most accurately with median error ranging from 4.1 to 10.8 percent. The spread between the mean and median suggest a number of outliers.

Figure 4L plots the empirical cumulative distribution (CDF) of prediction errors, quantifying the number of predictions with less than a particular error. The legend specifies the accuracy maximizing sample size for each model. The best performance models are application-specific. The equake-specific model is a representative S+A model, achieving the median accuracy over the six benchmarks we consider. 70 and 90 percent of equake predictions have less than 10 and 15 percent error, respectively. The flattening CDF slopes also indicate the number of outliers decrease with error.

Figure 4R considers absolute accuracy by plotting observed and predicted performance for each point in the validation set. Although predictions trend very well with actual observations, the figure suggests slight systematic biases. The model tends to overestimate performance in the range of [0.3,0.5] BIPS. This bias is likely due to imperfect normality of the residuals. The normal distribution of residuals is an underlying assumption to regression modeling. We initially found a significant deviation from normality and attempted a correction with a square-root transformation of the response variable. This transformation significantly reduced the magnitude of, but did not eliminate, the normality deviations [8]. Other transformations on the response or predictors may further mitigate these biases.

5.3 Performance Sensitivity Analyses

Although 4,000 observations are available for model formulation, the accuracy maximizing models use at most 2,000 observations to determine regression coefficients. We performed a sensitivity analysis for both regional and application-specific models. The

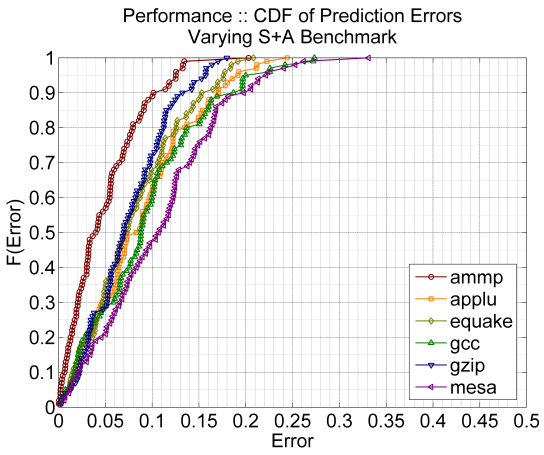

Figure 5. S+A Performance Model Sensitivity: Varying benchmarks.

regional model, with the region and sample size, has two free parameters. We find regional model accuracy generally insensitive to region size r_{S+R}, only observing a few smaller outlier errors with smaller region sizes. Similarly, larger sample sizes ($n_{S+R} > 2,500$) reduce the magnitude of outlier errors. Larger sample sizes increase observation density in the design space and enable tighter regions around a predictive query, thereby improving accuracy. There is little additional benefit from increasing the sample size beyond 3,000 [8].

For application-specific models, increasing n_{S+A} from 1,000 to 2,500 in increments of 500 provides no significant accuracy improvements. We also consider the sensitivity of application-specific models to benchmark differences in Figure 5. While the ammp-specific model performs particularly well with 91 percent of its predictions having less than 10 percent error, the mesa-specific model is least accurate with 47 percent of its predictions having less than 10 percent error and a single outlier having more than 30 percent error. The differences in accuracy across benchmarks may result from using the same predictors deemed significant for an average of all benchmarks and simply refitting their coefficients to obtain models for each benchmark. Predictors originally dropped/retained may become significant/insignificant when a particular benchmark, and not all benchmarks, is considered.

5.4 Power Prediction

The power model uses the performance model specification, replacing only the response variable. Such a model recognizes statistically significant architectural parameters for performance prediction are likely also significant for power prediction. The power model also recognizes an application's impact on dynamic power is a function of its microarchitectural resource utilization.

Table 5 summarizes the predictive accuracy of each power model. Variance stabilization has a significant impact on accuracy, reducing median error from 22.1 to 9.3 percent. Application-specific power modeling, with a median error between 10.3 and 11.3 percent, does not contribute significantly to accuracy. Regional power modeling achieves the greatest accuracy with only 4.3 percent median error.

Figure 6L plots the empirical CDF's of power prediction errors, emphasizing the differences in regional and application-specific modeling. Variance stabilization provides the first significant reduction in error and regional modeling provides the second. Application-specific modeling appears to have no impact on overall accuracy. The most accurate regional model achieves less than 10 percent error for nearly 90 percent of its predictions. The maximum error of 24.5 percent appears to be an outlier as 96 percent of predictions have less than 15 percent error. Again, the flattening CDF slopes also indicate the number of outliers decrease with error.

Figure 6R demonstrates very good absolute accuracy, especially for low power configurations less than 30 watts. The magnitude of prediction errors tend to increase with power and is most notable for high-power configurations greater than 100 watts. Configurations in the 100 watt region are dominated by deep pipelines for which power scales quadratically. This region in the design space contains relatively few configurations, all of which are leveraged for prediction in the regional model. Regions formed at the boundaries of the design space are often constrained by the bias toward configurations away from the boundary and, hence, produce a bias toward more conservative estimates.

5.5 Power Sensitivity Analyses

Like the performance models, we find application-specific power models show no sensitivity to the sample size n_{S+A}. The application-specific power models are also insensitive to the benchmark chosen [8]. In contrast, Figure 7L suggests region size r_{S+R} affects predictive ability. As the region size decreases from 4,000 to 1,000 in increments of 1,000, the model becomes increasingly localized around the predictive query. This locality induces shifts in the error distribution toward the upper left quadrant of the plot such that a larger percentage of predictions has smaller errors. Similarly, Figure 7R indicates the sample size n_{S+R} from which the region is drawn influences accuracy. As the sample size increases from 1,000 to 4,000, outlier error is progressively reduced from a maximum of 38 percent to a maximum of 25 percent. As with region size, we see shifts in the error distribution toward the upper left quadrant. Larger sample sizes increases observation density and enables tighter regions around a query.

5.6 Performance and Power Comparison

Compared against performance models, power models realize larger gains from variance stabilization. Although the best performance and power models achieve comparable median error rates of 4 to 5 percent, the source of accuracy gains are strikingly different. The greatest accuracy gains in performance modeling arise from eliminating application variability (S to S+A). Given a particular architecture, performance varies significantly across applications depending on its source of bottlenecks. Keeping the application constant in the model eliminates this variance. Regional modeling is relatively ineffective since application performance is dominated by its interaction with the microarchitecture and not the microarchitecture itself.

In contrast, power models are best optimized by specifying regions around each predictive query (S to S+R), thereby reducing microarchitectural variability. This is especially true for high power ranges in which power tends to scale quadratically with aggressive deeper pipelines, but linearly for more conservative configurations. Compared to regional performance models, regional power models are much more sensitive to region and sample sizes, probably because the regional optimization is much more effective for power prediction. Application-specific models add little accuracy since architectural configurations are primary determinants in unconstrained power. Power scaling effects for application-specific resource utilization are relatively small. Agressive clock gating, power gating, or DVFS techniques will likely increase the significance of application-specific predictors in power prediction. Combining the approaches of application-specific and regional modeling may capture these effects.

Model	Min	1st-Q	Median	Mean	3rd-Q	Max
B (1k)	0.252	8.381	22.066	39.507	55.227	244.631
S (2k)	0.168	3.796	9.316	13.163	21.849	45.004
S+R (1k)	0.076	2.068	4.303	5.6038	8.050	24.572
S+A (2k,ammp)	0.117	5.109	10.753	12.672	17.508	39.651
S+A (2k,equake)	0.112	4.806	11.332	13.190	19.141	44.429
S+A (2k,mesa)	0.021	5.150	10.316	13.491	20.233	45.158

Table 5. Summary of power prediction error with specified error minimizing sample and region sizes.

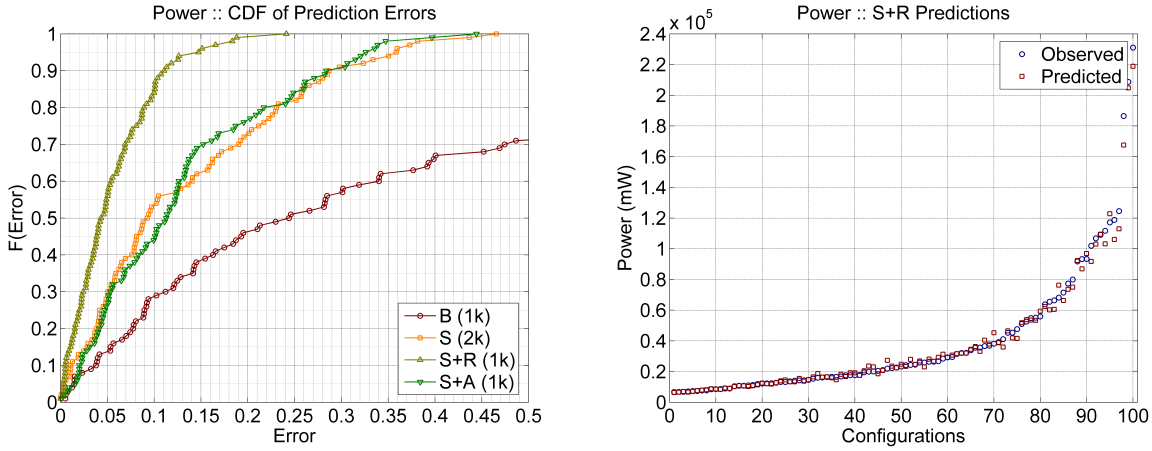

Figure 6. Empirical CDF of prediction errors with error minimizing sample and region sizes (L). Prediction results for S+R (R).

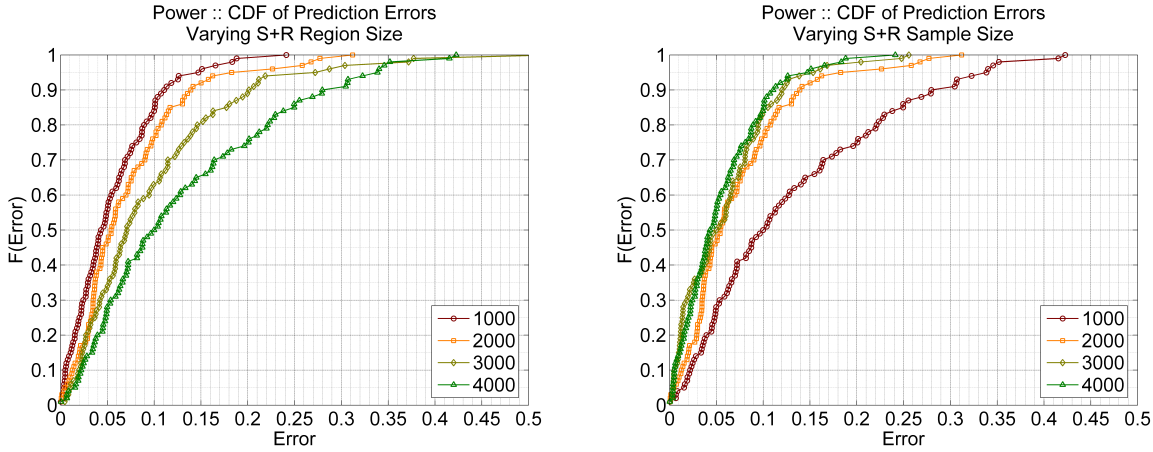

Figure 7. S+R Power Model Sensitivity: Varying r_{S+R} with fixed sample size $n_{S+R} = 4,000$ (L). Varying n_{S+R} with fixed region size $r_{S+R} = 1,000$ (R).

6. Related Work

Ipek, *et al.*, consider predicting performance of memory, core and CMP design spaces with artificial neural networks (ANN) [3]. ANN's are automated and do not require statistical analyses. Training implements gradient descent to optimize network edge weights and prediction requires evaluation of nested weighted sums of non-linear sigmoids. Leveraging statistical significance testing during model formulation, regression may be more computationally efficient. Training and prediction require numerically solving and evaluating linear systems. Furthermore, we perform regression for a substantially larger design space.

6.1 Statistical Significance Ranking

Joseph, *et al.*, derive performance models using stepwise regression, an automatic iterative approach for adding and dropping predictors from a model depending on measures of significance [6]. Although commonly used, stepwise regression has several significant biases cited by Harrell [4]. In contrast, we use domain-specific knowledge of microarchitectural design to specify non-linear effects and interaction between predictors. Furthermore, the authors consider only two values for each predictor and do not predict performance, using the models only for significance testing.

Yi, *et al.*, identify statistically significant processor parameters using Plackett-Burman design matrices [15]. Given these critical parameters, they suggest fixing all non-critical parameters to reasonable constants and performing more extensive simulation by sweeping a range of values for the critical parameters. We use various statistical techniques to identify significant parameters (Section 4), but instead of performing further simulation, we rely on regression models based on these parameters to explore the design space.

6.2 Synthetic Traces

Eeckhout, *et al.*, have studied statistical simulation in the context of workloads and benchmarks for architectural simulators [2]. Nussbaum, Karkhanis, and Smith have examined similar statistical approaches for simulating superscalar and symmetric multiprocessor systems [11, 7]. Both researchers claim detailed microarchitecture simulations for specific benchmarks are not feasible early in the design process. Instead, benchmarks should be profiled to obtain relevant program characteristics, such as instruction mix and data dependencies between instructions. A smaller synthetic benchmark is then constructed with similar characteristics.

The statistical approach we propose and the approaches proposed by Eeckhout and Nussbaum are fundamentally different. Introducing statistics into simulation frameworks reduces accuracy in return for gains in speed and tractability. While Eeckhout and Nussbaum suggest this trade-off for simulator inputs (*i.e.* workloads), we propose this trade-off for simulator outputs (*i.e.* performance and power results).

7. Conclusions and Future Work

We derive and validate performance and power regression models. Such models enable computationally efficient statistical inference, requiring the simulation of only 1 in 5 million points of a large design space while achieving median error rates as low as 4.1 percent for performance and 4.3 percent for power. Whereas application-specific models are most accurate for performance prediction, regional models are most accurate for power prediction.

Given their accuracy, our regression models may be applied to specific parameter studies. The computational efficiency of obtaining predictions also suggest more aggressive studies previously not possible via simulation. We use the same model specification for both performance and power. Similarly, applictation-specific models contain the same predictors regardless of benchmark. Fine-grained model customization would require additional designer effort, but may improve accuracy. Techniques in statistical inference are necessary to efficiently handle data from large scale simulation and are particularly valuable when archives of observed performance or power data are available.

Acknowledgments

We thank Patrick Wolfe at Harvard University for his valuable feedback regarding our regression strategies. This work is supported by NSF grant CCF-0048313 (CAREER), Intel, and IBM. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, Intel or IBM.

References

[1] D. Brooks, P. Bose, V. Srinivasan, M. Gschwind, P. G. Emma, and M. G. Rosenfield. New methodology for early-stage, microarchitecture-level power-performance analysis of microprocessors. *IBM Journal of Research and Development*, 47(5/6), Oct/Nov 2003.

[2] L. Eeckhout, S. Nussbaum, J. Smith, and K. DeBosschere. Statistical simulation: Adding efficiency to the computer designer's toolbox. *IEEE Micro*, Sept/Oct 2003.

[3] E.Ipek, S.A.McKee, B. de Supinski, M. Schulz, and R. Caruana. Efficiently exploring architectural design spaces via predictive modeling. In *ASPLOS-XII: Architectural support for programming languages and operating systems*, October 2006.

[4] F. Harrell. *Regression modeling strategies*. Springer, New York, NY, 2001.

[5] V. Iyengar, L. Trevillyan, and P. Bose. Representative traces for processor models with infinite cache. In *Symposium on High Performance Computer Architecture*, February 1996.

[6] P. Joseph, K. Vaswani, and M. J. Thazhuthaveetil. Construction and use of linear regression models for processor performance analysis. In *Symposium on High Performance Computer Architecture*, Austin, Texas, February 2006.

[7] T. Karkhanis and J. Smith. A first-order superscalar processor model. In *International Symposium on Computer Architecture*, June 2004.

[8] B. Lee and D. Brooks. Regression modeling strategies for microarchitectural performance and power prediction. Technical Report TR-08-06, Harvard University, March 2006.

[9] B. Lee and D. Brooks. Statistically rigorous regression modeling for the microprocessor design space. In *ISCA-33: Workshop on Modeling, Benchmarking, and Simulation*, June 2006.

[10] M. Moudgill, J. Wellman, and J. Moreno. Environment for powerpc microarchitecture exploration. *IEEE Micro*, 19(3):9–14, May/June 1999.

[11] S. Nussbaum and J. Smith. Modeling superscalar processors via statistical simulation. In *International Conference on Parallel Architectures and Compilation Techniques*, Barcelona, Sept 2001.

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

[13] C. Stone. Comment: Generalized additive models. *Statistical Science*, 1:312–314, 1986.

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

[15] J. Yi, D. Lilja, and D. Hawkins. Improving computer architecture simulation methodology by adding statistical rigor. *IEEE Computer*, Nov 2005.