Regression Modeling Strategies for Microarchitectural Performance and Power Prediction

Benjamin C. Lee *David M. Brooks*

Report No. TR-08-06

March 2006

Division of Engineering and Applied Sciences
Harvard University
Cambridge, Massachusetts 02138

Abstract

We propose regression modeling as an effective approach for accurately predicting performance and power for various applications executing on any microprocessor configuration in a large microarchitectural design space. This report addresses fundamental challenges in microarchitectural simulation costs via statistical modeling.

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 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) per benchmark application-specific models designed 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 model sensitivity to sample and region sizes, we find 4,000 samples from a design space of approximately 22 billion points, are sufficient for both application-specific and regional modeling and prediction. Collectively, our results suggest significant potential in accurate and efficient statistical inference for microarchitectural design space exploration via regression models.

1 Introduction

Cycle-accurate architectural performance simulators provide detailed insights into application performance on a wide-range of microprocessor configurations. These simulators are often used to identify trends and trade-offs for metrics of interest in microprocessor design. However, the prohibitively long simulation times of these tools preclude the exploration of application characteristics across a large microarchitectural design space. This work leverages statistical regression to derive simulation-free statistical inference models using data points sampled from the design space. We find introducing statistical inference into simulation frameworks modestly reduces detail in return for tremendous gains in speed and tractability. Although we consider advantages in computational cost for microarchitectural design space exploration in this report, 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 have become increasingly popular for approximating solutions to intractable problems. Even when 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. Given a regression model, prediction of the response simply requires evaluating a linear equation. Model formulation and evaluation are computationally efficient due to extensive research in numerical linear algebra.

After surveying applied statistical regression theory in Section 2, we derive performance and power regression models using a modest number of sample observations obtained from a large design space via simulation. In particular, we show 4,000 samples drawn uniformly at random from a design space with nearly 22 billion points are sufficient to formulate these models in Section 3. Each sample maps a set of architectural and application-specific predictors to the observed performance and power. These samples are used to formulate regression models, which predict the performance and power of previously unsampled configurations based on the same predictors.

We detail a statistically rigorous approach for deriving regression models in Section 4 that includes: (1) variable clustering, (2) association testing, (3) assessing strength of response-predictor relationships, and (4) significance testing with F-tests. These techniques ensure statistically significant architectural and application parameters are used as predictors. We also state and validate underlying regression assumptions, checking to ensure (1) residuals exhibit no systematic correlation with the fitted values or predictors and (2) residuals follow a normal distribution. Deviations from these assumptions are mitigated with variance stabilizing transformations.

Given baseline performance and power models that account for predictor interaction and non-linearity, we present 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 in Section 5.

The following summarizes experimental results 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 a median error as low as 4.1 percent (mean error as low as 5.6 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 a 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 96 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 to achieve roughly the same accuracy. 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 beyond the initial 4,000.

Collectively, these results suggest significant potential in accurate and efficient statistical inference for microarchitectural design space exploration via regression models.

2 Regression Theory and Techniques

We apply regression modeling techniques to efficiently obtain empirical estimates of microarchitectural 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 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). [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 $Var(e_i) = \sigma^2$.

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

Transformations f and $g = g_1, \ldots, g_p$ may be applied to the response and predictors, respectively, to improve model fit by stabilizing a non-constant error variance or accounting for non-linear correlations between the response and predictors. Throughout this section, we will assume no transformations are necessary.

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 for a set of observations by minimizing the sum of squared deviations or prediction errors of the predicted responses (given by the model) from the actual observed responses. Thus, least squares finds the $p + 1$ coefficients in Equation (1) to minimize $S(\beta)$ in Equation (2).

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

[1] $x_{i,0} = 1$ is implied

Equation (2) 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, $\hat{\beta}_j$, are estimates of the coefficients in Equation (1). Furthermore, the 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) and the goodness of fit (Section 2.5).

2.2 Predictor Interaction

In some cases, the effect of two 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. The interaction between two predictors may be modeled by constructing a third predictor $x_{i,3} = x_{i,1}x_{i,2}$ to obtain $y_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \beta_3 x_{i,1} x_{i,2} + e_i$

Modeling predictor interactions in this manner makes it difficult to interpret β_1 and β_2 in isolation. After simple algebraic manipulation to account for interactions, 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 (Section 2.6).

2.3 Non-Linearity

Basic linear regression models often assume the response behaves linearly in all predictors. This assumption is often too restrictive 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. For example, a linear spline (*i.e.*, piecewise linear function) on x with three knots at a, b, and c is given by Equation (3) where $(u)_+ = u$ if $u > 0$ and $(u)_+ = 0$ otherwise.

$$y = \beta_0 + \beta_1 x + \beta_2 (x-a)_+ + \beta_3 (x-b)_+ + \beta_4 (x-c)_+ \tag{3}$$

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 [6]. 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. For example, a cubic spline on x with three knots is given by Equation (4).

$$\begin{aligned} y &= \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \\ &\quad + \beta_4 (x-a)_+^3 + \beta_5 (x-b)_+^3 + \beta_6 (x-c)_+^3 \end{aligned}$$

Cubic splines may have poor behavior in the tails before the first knot and after the last knot [16]. 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. A restricted cubic spline on x with k knots $t_1, \ldots, t_k$ is given by Equation (4) where $j = 1, \ldots, k-2$ [3].

$$\begin{aligned} y &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_{k-1} x_{k-1} && (4) \\ x_1 &= x \\ x_{j+1} &= (x-t_j)_+^3 - (x-t_{k-1})_+^3 (t_k - t_j)/(t_k - t_{k-1}) \\ &\quad + (x-t_k)_+^3 (t_{k-1} - t_j)/(t_k - t_{k-1}) \\ & && (5) \end{aligned}$$

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 [15]. 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. Recommended equally spaced quantiles are shown in Table 1.

In practice, five knots or fewer are generally sufficient for restricted cubic splines [15]. 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 [6]. For larger data sets with more than 100 samples, five knots may also be a good choice.

k	Quantiles						
3			0.1000	0.5000	0.9000		
4			0.0500	0.3500	0.6500	0.9500	
5		0.0500	0.2750	0.5000	0.7250	0.9500	
6	0.0500	0.2300	0.4100	0.5900	0.7700	0.9500	
7	0.0250	0.1833	0.3417	0.5000	0.6583	0.8167	0.9750

Table 1: **Recommended knot placement**. Knots are placed at fixed quantiles of the data.[6].

2.4 Significance Testing

Suppose we formulate a model with p predictors that allows for interaction and non-linearity. There are three contributions to a response estimate: (1) primary effects (*e.g.* x_i), (2) interaction effects (*e.g.* $x_i x_j$), and (3) non-linear effects (*e.g.* a restricted cubic spline of x_i with k knots, $rcs(x_i, k)$). The significance of the association between the response and each of these contributions may be quantified with standard statistical tests.

2.4.1 T-Tests

The statistical properties of the coefficient estimates are used to evaluate the association between the response and terms in the model. If the errors e_i in Equation (1) are independent normal random variables, then the estimated coefficients $\hat{\beta}_j$ are also normally distributed. This normality assumption leads to the relationship in Equation (6) that states the standardized coefficient estimate follows a t distribution with $n - p - 1$ degrees of freedom. The estimates $\hat{\beta}_j$ are obtained in closed form by solving a linear system and their standard deviations $s_{\hat{\beta}_j}$ may be obtained analytically from the closed form estimates.

$$\frac{\hat{\beta}_j - \beta_j}{s_{\hat{\beta}_j}} \sim t_{n-p-1} \tag{6}$$

A commonly tested null hypothesis states the j-th term in the model has no association with the response ($H_0 : \beta_j = 0$). This hypothesis is tested by evaluating Equation (6) under the null hypothesis to obtain $\hat{\beta}_j / s_{\hat{\beta}_j}$, a value often referred to as the *t-statistic*. The t-statistic is typically computed for every coefficient estimate $\hat{\beta}_j$ as a first step toward determining significance of the j-th term in the model. Note the t-statistic follows the t-distribution. Also note the j-th term may be a primary, interaction, or non-linear term.

The *p-value* is defined as $2P(X \geq |c|)$ for a random variable X and a constant c. In our analyses, $X \sim t_{n-p-1}$ and c is the t-statistic, $\hat{\beta}_j / s_{\hat{\beta}_j}$. The p-value may be interpreted as the probability a t-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 $\hat{\beta}_j$ casts doubt on the hypothesis $\beta_j = 0$ and suggests the effect from the j-th term in the model is statistically significant in predicting the response.

2.4.2 F-Tests

Although t-tests are often used for assessing 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, performing significance tests on a subset of the β's is preferable to individual t-tests for complex models with interaction and non-linear terms.

The *F-test* is a standard statistical test for comparing two nested models (*e.g.* a full model and a subset of the full model) using their multiple correlation statistic, R^2. Equation (9) computes this statistic by computing regression error (SSE) as a fraction of the total error (SST). From the equation, 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.

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{7}$$

$$SST = \sum_{i=1}^{n} \left(y_i - \frac{1}{n} \sum_{i=1}^{n} y_i \right)^2 \tag{8}$$

$$R^2 = 1 - \frac{SSE}{SST} \tag{9}$$

Given R^2 for the full model and R^2_* for a model constructed by dropping a number of terms from the full model, define the *F-statistic* by Equation

(10) 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. Since the F-statistic follows the F-distribution, a p-value may be calculated, as shown for t-tests, and the significance of the dropped terms may be assessed.

$$F_{k,n-p-1} = \frac{R^2 - R_*^2}{k} \times \frac{n-p-1}{1-R^2} \tag{10}$$

2.5 Assessing Fit

The model's fit to the observations used to formulate the model measures how well it 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 (11) 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 should be the same for all predictor and 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 $j \in [0,p]$ 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.

$$\hat{e}_i = y_i - \hat{\beta}_0 - \sum_{j=0}^{p} \hat{\beta}_j x_{ij} \tag{11}$$

Additionally, fit may be assessed by the R^2 statistic from Section 2.4. Larger values of 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 estimate relatively small data sets. Studies in which models are validated on independent data sets have shown a regression model is likely to be reliable when the number of predictors p is less than $n/20$, where n is the sample size [7].

2.6 Prediction

Once β is determined, evaluating Equation (1) for a given x_i will give the expectation of y_i and, equivalently, an estimate $\hat{y}_i$ for y_i (Equation (12)). 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.

$$\begin{aligned} \hat{y}_i &= E[y_i] \\ &= E\left[\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij}\right] + E[e_i] \\ &= \beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} \qquad (12) \end{aligned}$$

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.

3.1 Simulation Framework

We use Turandot, a generic and parameterized, out-of-order, superscalar processor simulator [12, 13]. Turandot is enhanced with PowerTimer to obtain power estimates based on circuit-level power analyses and resource utilization statistics [5, 2]. The modeled baseline architecture is similar to the current POWER4/POWER5.

3.2 Benchmarks

We consider a Java server benchmark, SPECjbb, 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 [8].

3.3 Statistical Analysis

The $n = 4,000$ observations from the full design space provide sufficient data for statistical regression analysis. We use R, a free software environment for statistical computing, to script and automate the statistical analyses described in Section 2 [17]. Within this environment, we use the Hmisc and Design packages implemented by Harrell [6].

3.4 Configuration Sampling

The approach to obtaining observations from a large microprocessor design is critical to efficient formulation of regression models. Table 2 identifies twelve groups of parameters varied simultaneously. Parameters within a group are varied together 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.

Traditional techniques of sweeping design parameter values to consider all points in a large design space is impossible despite continuing research in reducing simulation costs via trace compression [11, 1] and sampling [4, 14]. Although per simulation costs are reduced by a constant factor, these techniques do not reduce the number of required simulations. Other studies have reduced the cardinality of these sets to an upper and lower bound [9], but this approach masks performance and power trends between the bounding values and precludes any meaningful statistical inference or prediction. For these reasons, sampling must occur in the design space to control the exponentially increasing number of design points as the number parameter sets and their cardinality 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

	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 purpose (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 sache 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	Control Latency	branch latency	cycles	1,2	2
S_9	FX Latency	ALU latency	cycles	1::1::5	5
		FX-multiply latency	cycles	4::1::8	
		FX-divide latency	cycles	35::5::55	
S_{10}	FP Latency	FPU latency	cycles	5::1::9	5
		FP-divide latency	cycles	25::5::45	
S_{11}	L/S Latency	Load/Store latency	cycles	3::1::7	5
S_{12}	Memory Latency	Main memory latency	cycles	70::5::115	10

Table 2: **Range and grouping of microarchitectural parameters**. 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.

sets. Furthermore, sampling UAR does not bias the observations toward any particular configurations. From the law of large numbers, the expected number of samples with each configuration value is the same. Our approach is different from Monte Carlo methods. Monte Carlo methods 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 frequency analyses.

Suppose we treat the configurations for which responses are not observed as missing data from a full data set with all $|S|$ observations. Then sampling UAR ensures the observations are *missing completely at random* (MCAR). Under MCAR, data elements are missing for reasons unrelated to any characteristics or responses of the configuration. In this context, the fact a design point is unobserved is unrelated to the performance, power, or configuration of the design.

In contrast, *informative missing* describes the case when elements are more likely to be missing if their responses are systematically higher or lower. For example, simulator limitations may prevent data collection for very low performance architectures and the "missingness" of a configuration is correlated with its performance. In this case, the "missingness" is non-ignorable and we must formulate an additional model to predict whether a design point can be observed by the simulator. By sampling UAR from the design space, we ensure the observations are MCAR and avoid such modeling complications.

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 (configurations and applications) in the design space. We demonstrate the efficiency of this extremely sparse sampling for data collection, model formulation, and statistical inference in Section 5.

4 Model Derivation

We formulate regression models for performance based on the $n = 4,000$ sampled observations. To identify potential response-predictor relationships we first examine descriptive statistics for parameters in the data set. We then formulate a regression model, using our domain-specific understanding of microarchitectural design to specify predictors' primary effects, second- and third-order interactions, and non-linearities. Given an initial model, we perform significance testing to prune statistically insignificant predictors.

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 3: **Application characteristics**. Measured application behavior when executing on baseline architecture.

The fit of the refined model is assessed by examining its residuals.

4.1 Predictors

We consider the 12 architectural predictors drawn from the parameters in Table 2. We also consider 15 application-specific predictors drawn from an application's characteristics (Table 3) when executing on a baseline configuration (Table 4). These characteristics may be significant predictors of performance when interacting with architectural predictors. For example, the performance effect of increasing the data L1 cache 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. For example, an application with no bottlenecks and high baseline performance would benefit less from additional physical registers compared to an application experiencing heavy register pressure. These potential interactions suggest both architectural and application-specific predictors are necessary.

4.2 Summary Statistics

4.2.1 Variable Clustering

A variable clustering analysis reveals key variable interactions. Figure 1 presents the results of hierarchical variable clustering based on squared

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 4: **Baseline Architecture**. Configuration for which application characteristics are initially measured.

Spearman rank correlation coefficients as similarity measures where a larger ρ^2 indicates a greater correlation between variables. If the number of predictors is large relative to the number of observations, over-fitting the model may be a concern and redundant predictors may be eliminated by selecting only one predictor from each cluster.

L1 and L2 misses due to instruction cache accesses are highly correlated. We found the absolute number of L2 cache misses from the instruction probes to be negligible and eliminate `il2miss_rate` from consideration. Similarly, the branch rate is highly correlated with the number of branch induced stalls and we eliminate `br_stall`.

Pipeline depth is highly correlated with latency since we scale the original functional unit latencies with depth [10]. Since final latencies are a function of original latencies and pipeline depth, we choose to keep these original latency variables. Including both predictors enables us to differentiate the performance impact of individual functional unit latency changes from the global latency changes as depth varies. Similarly, we keep both d-L1 cache miss rate and the baseline performance predictors to differentiate cache performance from global performance.

Overall, the total number of observations is sufficiently large to avoid over-fitting. All other variables are correlated to a lesser degree, if at all,

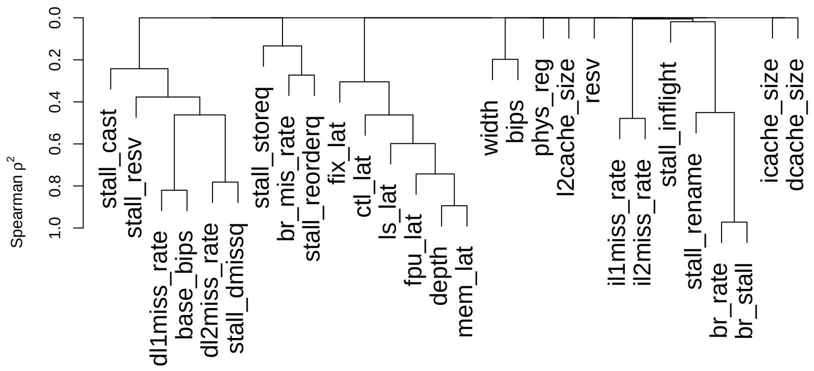


Figure 1: **Variable clustering**. The horizontal lines at which variables connect specify the correlation coefficient. A larger ρ^2 suggests greater correlation between variables.

and should not be eliminated from consideration.

4.2.2 Performance Associations

Plotting each of the predictors against the response may reveal particularly strong associations or identify non-linearities. We stratify each predictor into four groups such that each group covers equally sized intervals of predictor value and plot the mean of each group against performance (Figures 2–3). The numbers on the left axis specify each group's range and those on the right axis specify the number of observations within the range.

For architectural predictors, pipeline depth and width are strong, monotonic factors as shown in Figure 2L. The number of physical registers may be a significant, but non-linear, predictor. The number of reservation stations seems to have limited association with performance. The performance-latency relationships are counter-intuitive as performance increases with latency in Figure 2R. This trend is an artifact of latency scaling as pipeline depth varies; deeper pipelines increase instruction throughput but also increase latencies measured in cycles. Lastly, Figure 3L demonstrates a posi-

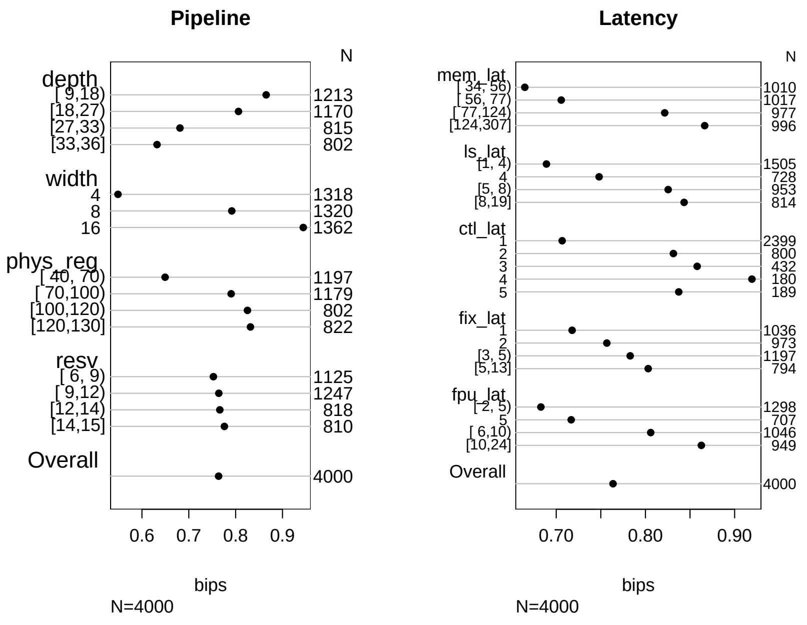


Figure 2: **Association Plots**. Measures of pipeline dimensions (L) and unscaled latencies (R).

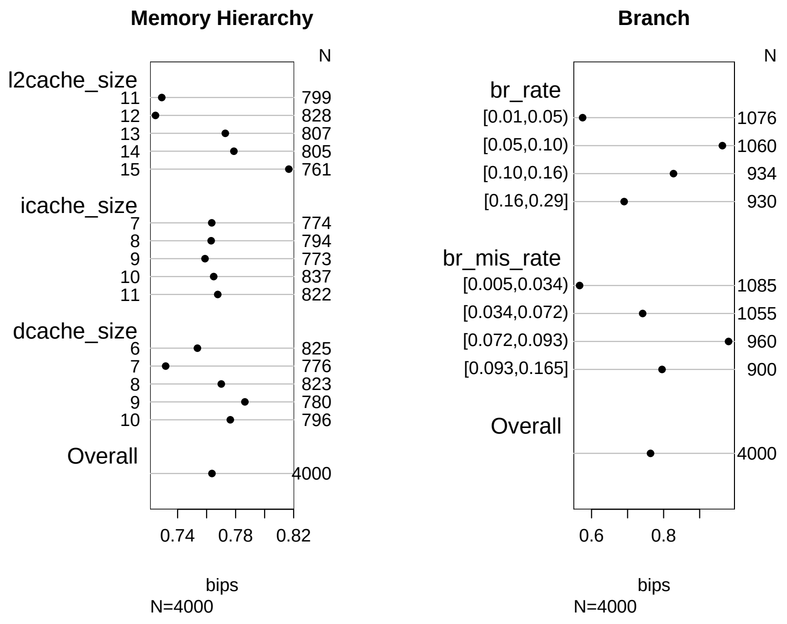
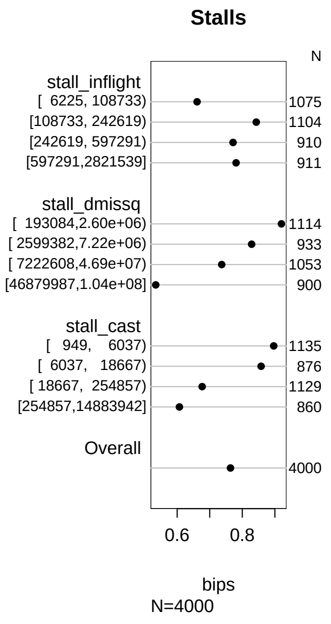


Figure 3: **Association Plots**. Memory hierarchy (L) and application branching behavior(R).



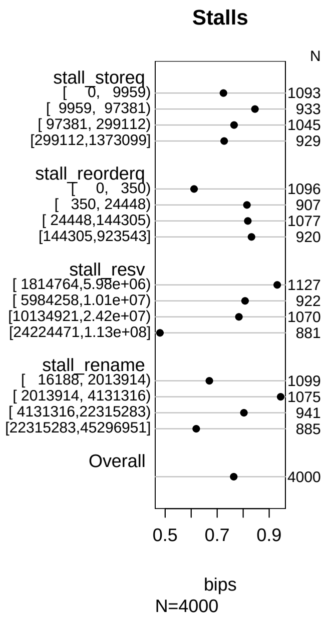


Figure 4: **Association Plots**. Application baseline stall behavior.

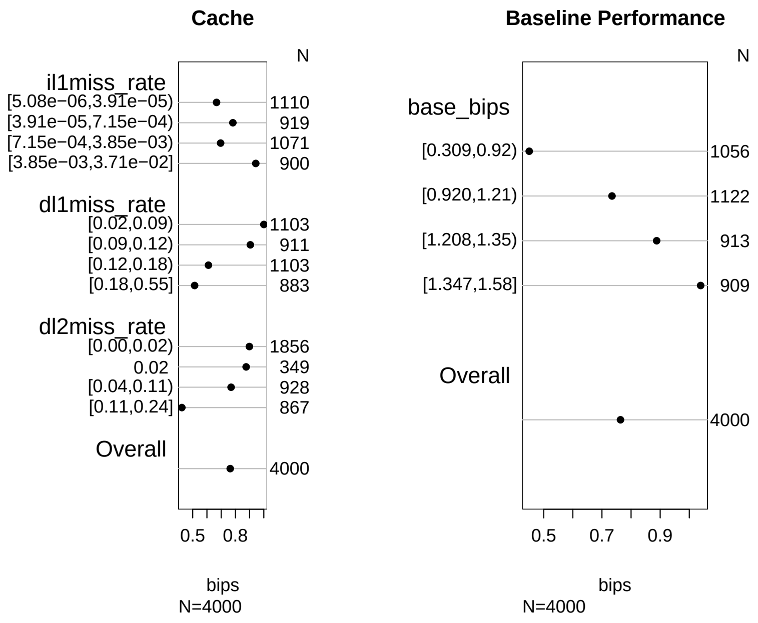


Figure 5: **Association Plots**. Application cache behavior (L) and baseline performance (R).

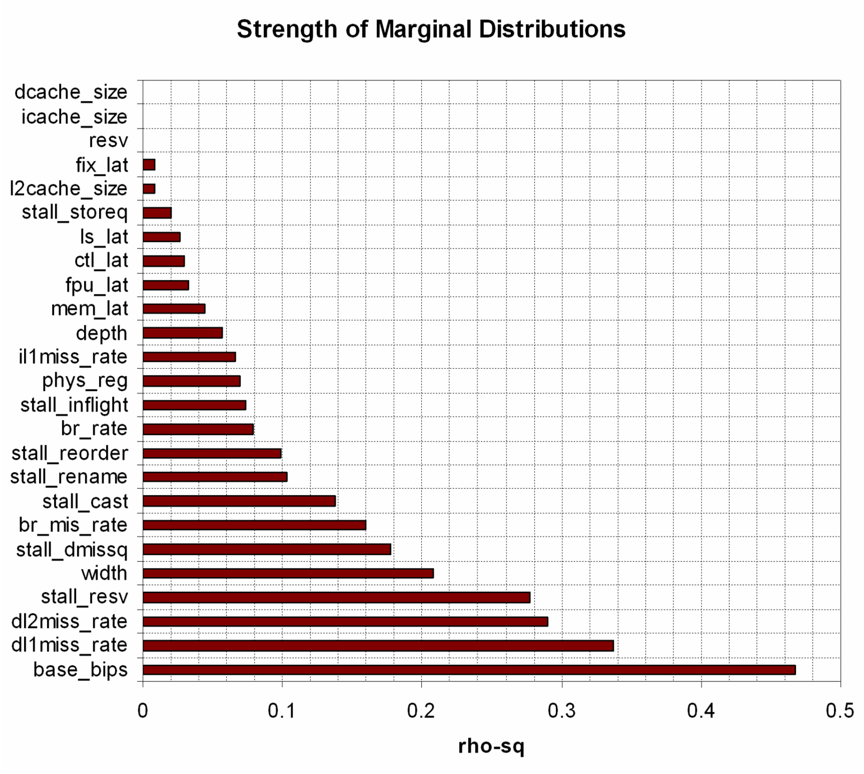


Figure 6: **Strength of Marginal Relationships**. A lack of fit for predictors with high ρ^2 will more consequentially impact performance prediction.

tive correlation between L2, but not L1, cache size and performance.

For application-specific predictors, branching behavior exhibits no obvious trends as shown in Figure 3R. Figure 4 indicates roughly half the stall characteristics have monotonic relationships with performance (*i.e.*, `dmissq`, `cast`, `reorderq`, `resv`). Although instruction L1 miss rate seems to have limited predictive ability, probably due to the small number of such misses in absolute terms, data cache access patterns are good predictors of performance as shown in Figure 5L. Lastly, Figure 5R shows a benchmark's observed sample and baseline performance are highly correlated.

4.2.3 Strength of Marginal Relationships

Figure 6 plots the non-monotonic generalization of the Spearman rank correlation coefficient for each of the pre-specified predictor variables. This information will guide our choice in the number of spline knots and predictor interactions. 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 descending order of importance) for width, depth, physical registers, functional unit latencies, cache sizes, and reservation stations. However, application-specific baseline performance is the most significant predictor.

4.3 Model Specification

We choose to model non-linearities for architectural predictors using restricted cubic splines. The strength of predictors' marginal relationships with performance will guide our choice in the number of knots. Predictors with stronger relationships 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 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 second- and third-order interactions. Generally, we expect pipeline dimensions to interact with structures that control instruction bandwidth and caches that impact hazard counts. We also expect the memory hierarchy to interact with application-specific access rates. Lastly, we include interactions with baseline performance to account for diminishing/increasing marginal returns from low/high baseline performance.

4.4 Model Refinement

We check predictor significance to refine and check the efficiency of the model. 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. Variables `ctl_lat` (p-value=0.1247), `stall_inflight` (p-value=0.7285), and `stall_storeq` (p-value=0.4137) appear insignificant. The high p-values for the F-test indicate these predictors do not significantly contribute to a better fit when included to form a larger model and may be removed.

Predictor	$R^2 - R^2_*$	$DF - DF_*$	F-test	P-value
depth*	-6.2688	-60	24.72	$< 2.2e-16$
width*	-6.4709	-19	80.58	$< 2.2e-16$
phys reg*	-11.713	-11	251.95	$< 2.2e-16$
resv	-0.0177	-2	2.10	0.1239
i-L1$ size*	-0.0906	-11	1.9491	0.02941
d-L1$ size*	-0.4206	-29	3.4322	1.629e-09
L2$ size*	-5.4024	-45	28.412	$< 2.2e-16$
ctl lat	-0.010	-1	2.3585	0.1247
fx lat*	-2.4166	-1	571.92	$< 2.2e-16$
fp lat*	-0.4383	-2	51.861	$< 2.2e-16$
mem lat*	-0.5321	-1	125.89	$< 2.2e-16$
i-L1miss rate*	-1.9947	-16	29.505	$< 2.2e-16$
d-L1miss rate*	-1.5303	-19	19.061	$< 2.2e-16$
d-L2miss rate*	-0.917	-14	15.502	$< 2.2e-16$
branch rate*	-0.5629	-2	66.607	$< 2.2e-16$
branch mispred*	-0.1231	-5	5.8246	2.293e-05
stall inflight	-0.0005	-1	0.1205	0.7285
stall dmissq*	-0.2938	-1	69.53	$< 2.2e-16$
stall cast*	-0.1825	-1	43.198	5.61e-11
stall storeq	-0.0028	-1	0.6684	0.4137
stall reorderq*	-0.1033	-1	24.440	7.994e-07
stall resv*	-0.1033	-1	24.440	7.944e-07
stall rename*	-0.1033	-1	24.440	7.944e-07
base bips*	-5.1509	-29	42.035	$< 2.2e-16$

Table 5: **Significance Testing**. F-test results for each predictor where lower p-values indicate greater significance. * denotes inclusion in general model (Figure 15).

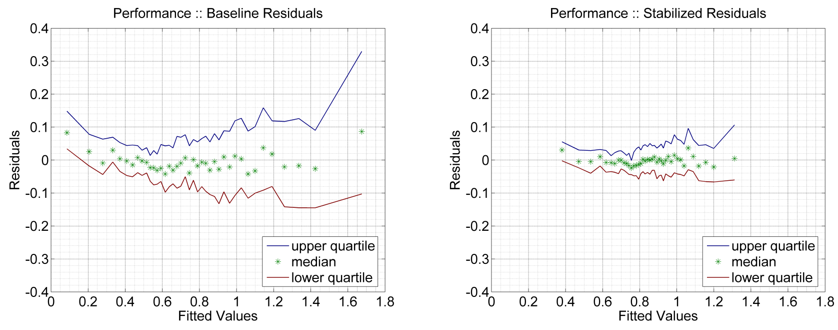


Figure 7: **Residual Plots**. Quartiles of residuals, stratifying 4000 fitted performance into groups of 100, before (L) and after(R) variance stabilization. The x-coordinate is the mean predicted performance of the 100 group members.

We examine the residuals to assess model fit. Figure 7L plots the median, lower, and upper quartiles of the residuals for 40 groups, each with 100 observations, against modeled performance. For each group, the x-coordinate is the mean performance of the observations in the group. The plot reveals significant correlations between residuals and fitted values; residuals are larger for the smallest and largest fitted values. We apply a standard variance stabilizing technique to mitigate the magnitude of these correlations, performing a square root transformation on performance. Figure 7R shows stabilized residuals for performance. Variance stabilization also cause the residuals to follow a normal distribution more closely as shown in Figure 8. The observed linear trend indicates the residuals follow the normal distribution when normal and residual quantiles are plotted on the x-axis and y-axis, respectively.

Figure 15 presents the final model specification using standard syntax for the R statistical package.

5 Model Evaluation

5.1 Model Variants and Optimizations

We use data sets of varying size drawn from the $n = 4,000$ random observations to formulate regression models. We refer to the cardinality of these sets as the *sample size*, denoted by by n_*. Each model may require different

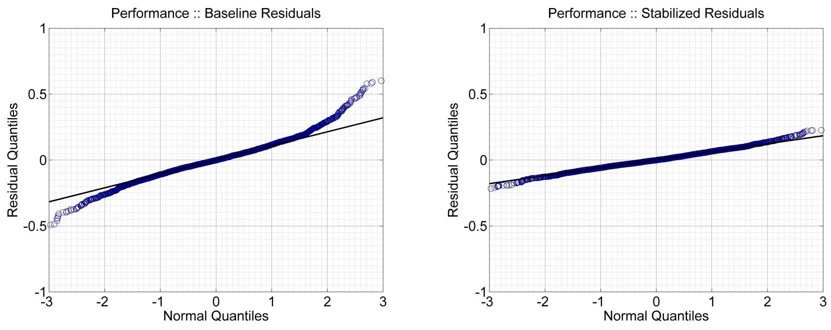


Figure 8: **Quantile-Quantile Plots**. Quantiles of residuals (y-axis) plotted against normal quantiles (x-axis). A line is drawn between the second and third quartile of the residuals to check linearity.

sample sizes to maximize accuracy. We also obtain 100 additional random samples for predictive queries and validation. We compare and contrast the predictive ability of four regression models, differing in specification and data used to perform the fit:

- **Baseline (B):** Model specified 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 variance stabilizing square-root transformation on the response and formulated with a naive subset of $n_S < n$ observations.

- **Regional (S+R):** Model is formulated using the $r_{S+R} < n_{S+R} < n$ observations with microarchitectural configurations most similar to each predictive query's configuration. We refer to r_{S+R} as the *region size* drawn from the larger sample size. Similarity is quantified by the relative 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 (Figure 16) and reformulating the model with a naive subset of $n_{S+A} < n_A$. We

Model	Min	1st-Q	Median	Mean	3rd-Q	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,applu)	0.270	4.066	8.497	8.804	12.230	24.423
S+A (1k,equake)	0.181	4.132	7.385	8.064	11.202	20.825
S+A (1k,gcc)	0.151	4.170	8.813	9.364	13.073	27.327
S+A (1k,gzip)	0.313	3.546	6.982	7.486	10.549	17.984
S+A (1k,mesa)	0.170	5.736	10.775	10.810	15.025	33.129

Table 6: **Summary of Performance Prediction Error**. The error minimizing sample and region sizes is specified.

consider S+A models for six benchmarks: ammp, applu, equake, gcc, gzip, and mesa.

5.2 Performance Prediction

Table 6 summarizes the predictive accuracy of each model by presenting the percentage error, $100 * |\hat{y}_i - y_i|/y_i$. Each model is presented with error minimizing sample and region sizes. Variance stabilization reduces median error from 13.1 to 10.9 percent and applying regional or application-specific models may further reduce median error. Application-specific models predict performance with the greatest accuracy with median error ranging from 4.1 to 10.8 percent. The spread between the mean and median suggest a number of outliers.

Figure 9L plots the empirical cumulative distribution (CDF) of prediction errors, quantifying the number of predictions with less than a particular error. We believe the error distribution is a more effective measure of accuracy when compared to the usually reported median and mean. 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 errors, respectively. The flattening CDF slopes also indicate the number of outliers decrease with error.

In addition to relative accuracy, Figure 9R 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 over-estimate

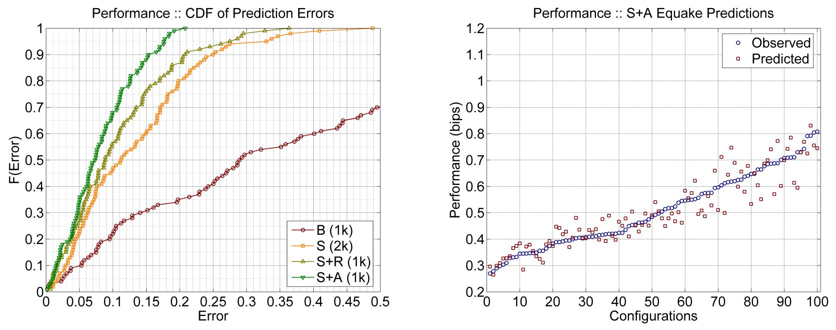


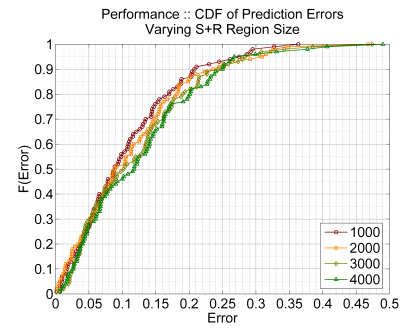
Figure 9: Empirical CDF of prediction errors with error minimizing sample and region sizes. Equake achieves median accuracy and is plotted for S+A (L). Prediction results for S+A (R).

performance in the range of [0.3,0.5] . 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, but did not eliminate the normality deviations (Figure 7). A different variance stabilizing transformation on the response or additional transformations on the predictors may further reduce these biases.

5.3 Performance Sensitivity Analyses

We assess the sensitivity of regional and application-specific models to region sizes, sample sizes, and choice of benchmarks. Although 4,000 observations are available for model formulation, the optimal models use at most 2,000 observations to determine regression coefficients. Regional model accuracy is modestly sensitive to region size, r_{S+R}, with smaller, tighter regions around the predictive query yielding better accuracy (Figure 10L). As the region size decreases, the CDF shifts to the upper left quadrant, resulting in lower median/mean errors and smaller outlier errors. Increasing the sample size, n_{S+R}, from which the region is drawn improves observation density in the design space, enables tighter regions, and improves accuracy (Figure 10R). However, there appears to be no additional benefit from increasing the sample size beyond 3,000.

The application-specific models are insensitive to sample size. Increasing n_{S+A} from 1,000 to 4,000 for the equake-specific model has no impact



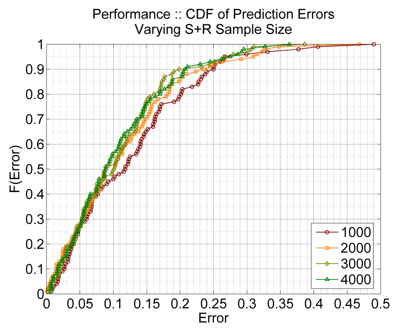


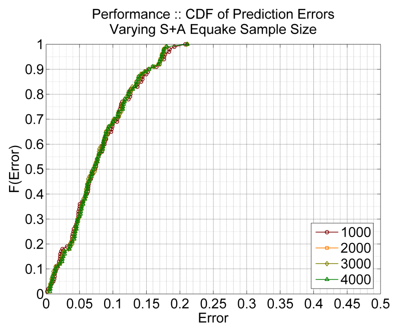
Figure 10: S+R Performance Model Sensitivity: Varying r_{S+R} with fixed $n_{S+R} = 4,000$ (L) and n_{S+R} with fixed $r_{S+R} = 1,000$ (R).

on accuracy (Figure 11L). This observation for the equake-specific model is representative of other benchmark models. Application-specific model accuracy varies with the benchmark (Figure 11R). 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. Application-specific models also appear to reduce outlier error more effectively for different benchmarks. The differences in accuracy across benchmarks likely 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 model also recognizes an application's power impact as dynamic power dissipation is often a function of application-specific microarchitectural resource utilization.

Table 7 summarizes the predictive accuracy of each power model. Variance stabilization has a significant impact on accuracy, reducing median error from 22.0 to 9.3 percent. Application-specific power modeling, with a median error between 9.9 and 11.5 percent, and does not contribute signifi-



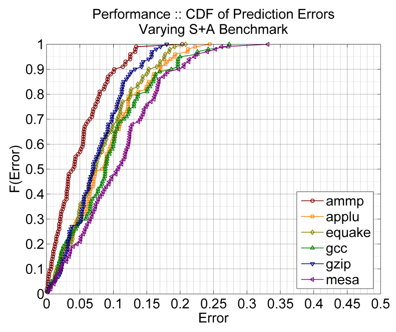


Figure 11: S+A Performance Model Sensitivity: Varying n_{S+A} (L) and benchmark (R).

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)	0.185	4.687	11.519	13.538	20.965	50.090
S+A (2k,applu)	0.081	4.846	11.006	13.067	18.191	44.662
S+A (2k,equake)	0.112	4.806	11.332	13.190	19.141	44.429
S+A (2k,gcc)	0.190	5.123	9.950	12.752	16.647	43.757
S+A (2k,gzip)	0.504	5.120	9.925	12.405	17.946	39.552
S+A (2k,mesa)	0.021	5.150	10.316	13.491	20.233	45.158

Table 7: **Summary of Power Prediction Error**. The error minimizing sample and region sizes is specified.

cantly to accuracy. Regional power modeling achieves the greatest accuracy with only 4.3 percent median error. The spread between the mean and median suggest a number of outliers.

Figure 12L 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.

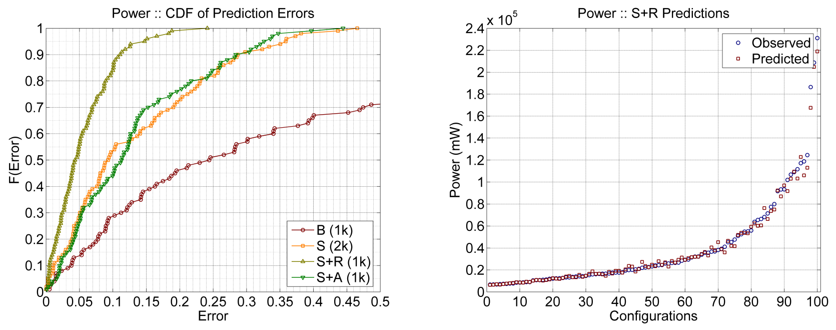


Figure 12: Empirical CDF of prediction errors with error minimizing sample and region sizes (L). Prediction results for S+R (R).

Figure 12R 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 range predominantly have deep pipelines for which power dissipation scales quadratically. This power range contains relatively few configurations, but all configurations in this range are leveraged for prediction in the regional model. Region identified 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

The sensitivity analyses for power are the same as those for performance. We assess the sensitivity of regional and application-specific models to region sizes, sample sizes, and choice of benchmarks. Regional model accuracy is strongly impacted by the region size, r_{S+R} (Figure 13L). As the region size decreases from 4,000 to 1,000, the 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, the sample size, n_{S+R}, heavily influences accuracy (Figure 13R). As the sample size increases from 1,000 to 4,000, outlier error is progressively reduced from a maximum of 42 percent to a maximum of 24 percent. As with region size, we see shifts in the error distribution toward the upper left quadrant.

The application-specific power models are insensitive to both sample size

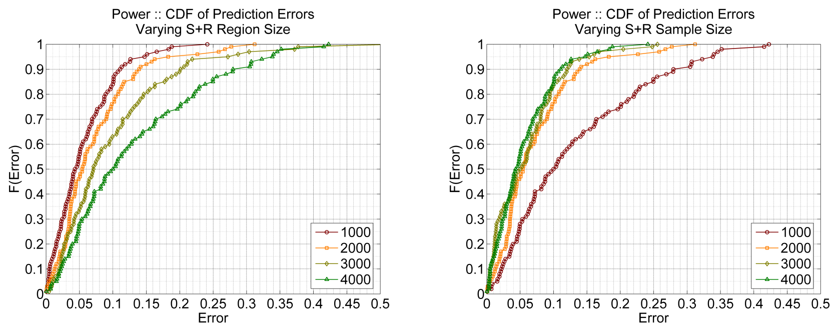


Figure 13: S+R Power Model Sensitivity: Varying r_{S+R} with fixed $n_{S+R} = 4,000$(L) and n_{S+R} with fixed $r_{S+R} = 1,000$ (R).

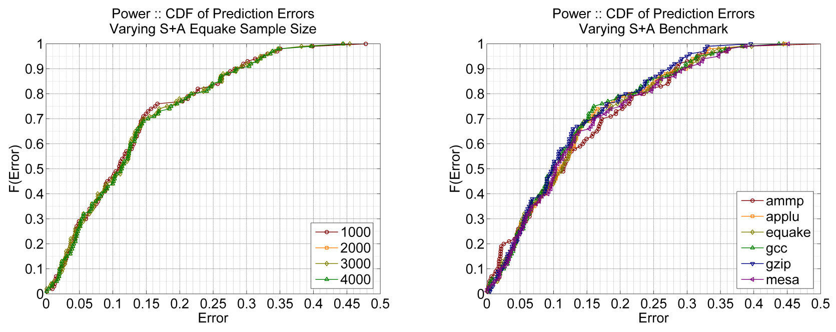


Figure 14: S+A Power Model Sensitivity: Varying n_{S+A}(L) and benchmark(R).

and benchmark. Increasing n_{S+A} from 1,000 to 4,000 for the representative equake-specific model has no impact on accuracy (Figure 14L), suggesting additional observations are unlikely to improve S+A power model accuracy. Application-specific power models also exhibit similar accuracy across several different benchmarks (Figure 14R). This insensitivity to benchmarks suggests the accuracy of power estimates are more heavily influenced by the query's microarchitectural configuration and significantly less impacted by the particular executing application.

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. Fixing the application in the model eliminates this variance. Regional modeling does not contribute much more to accuracy 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 tighter regions around each predictive query (S to S+R), thereby reducing microarchitectural variability. This is especially true for high power ranges since power tends to scale quadratically with aggressive deeper pipelines and linearly for more conservative configurations. Application-specific models add little accuracy since architectural configurations and resource sizings are primary determinants in unconstrained power. The effects of scaling unconstrained power for application-specific resource utilization are relatively small. Thus, formulating a power model from observations with minimal differences in resource sizing reduces this variance and associated errors.

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. In both performance and power models, larger sample sizes for regional power models increase observation density, enable tighter regions around a configuration of interest, and improve accuracy.

6 Related Work

Our work is unique in its joint consideration of architectural and application-specific parameters for performance and power prediction. In contrast, there has been significant work in optimizing the performance of microprocessor simulators. Typical approaches include (1) reducing the number of configurations to be simulated by identifying statistically significant parameters or (2) statistically simplifying and reducing the size of instruction traces to approximate metrics of interest.

6.1 Statistical Significance Ranking

Yi, *et al.*, have studied improving computer architecture simulation by adding statistical rigor [18]. This rigor is achieved by identifying critical, statistically significant processor parameters using Plackett-Burman design matrices to design optimal multi-factorial experiments. Given these critical parameters, they suggest fixing all non-critical parameters to reasonable constants and performing more extensive simulation by sweeping a range values for the critical parameters.

In deriving regression models, we perform a number of tests to ensure model predictors are statistically significant: (1) variable clustering, (2) response-predictor association testing, (3) assessing the strength of marginal relationships by calculating Spearman rank correlation coefficients, and (4)computing F-tests and p-values for each predictor. Once a modest number of observations have been sampled via simulation, significant parameters are used as predictors in a regression model to predict metrics of interest. Instead of performing further simulation, we rely on the regression model to explore the design space.

Joseph, *et al.* derive performance regression models using stepwise regression, an automatic iterative approach to adding and dropping predictors from a model depending on measures of significance [9]. Although commonly used, stepwise regression has several problems cited by Harrell [6]: (1) R^2 values are biased high, (2) standard errors of regression coefficients are biased low leading to falsely narrow confidence intervals, (3) p-values are too small, and (4) coefficients are biased high. In contrast, we prefer to 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.

6.2 Synthetic Traces

Eeckhout, *et al.*, have studied statistical simulation in the context of workloads and benchmarks for architectural simulators [4]. Nussbaum, *et al.* has examined similar statistical approaches for simulating superscalar and symmetric multiprocessor systems [14]. 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.

A synthetic benchmark that is statistically equivalent to a longer, real benchmark is an approach to reduce simulation time. However, we perform simulations only to obtain the necessary samples from the full design space to develop regression models. We have not yet determined whether collecting these samples with real or synthetic traces will affect the final accuracy of these models. The longer simulation time from using real benchmarks amortized over repeated use of the resulting models may be preferable if model accuracy benefits.

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 Directions

We detail the derivation and validation of performance and power regression models. Such models enable computationally efficient statistical inference, requiring the simulation of only 1 in 5 million points in a joint microarchitecture-application design space while achieving 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 differ markedly. Whereas application-specific models are most accurate for performance prediction, regional models are most accurate for power prediction.

Given their accuracy when validating against random design points, our regression models may be applied to specific parameter studies. We intend to reproduce prior design space studies with our purely analytical model, validating its ability to capture performance and power trade-offs. The low computational costs of obtaining predictions also suggest more aggressive studies previously not possible via simulation.

Additional model refinements may further improve accuracy. Despite a basic variance stabilizing transformation, our models still produce slightly non-normal residual distributions. Different transformations on the response and predictors may eliminate the resulting bias. Although we present per benchmark application-specific models, a coarse-grained partitioning of the application domain would result in fewer models, lowering sampling costs, at the cost of accuracy. Quantifying this trade-off is future work. Lastly, we use the same model specification for both performance and power. We also use the same specification for each application-specific model regardless of application. A rigorous derivation to specify different models would require additional designer effort, but may also yield accuracy improvements.

We also intend to compare and contrast the accuracy and costs of regression modeling with other statistical techniques, such as machine learning and neural networks. Although the specific techniques may differ, we believe statistical techniques for inference are necessary to efficiently handle data from large scale simulation. Statistical approaches are particularly valuable when archives of observed performance or power data are available.

References

[1] K. Barr and K. Asanovic. Branch trace compression for snapshot-based simulation. In *International Symposium on Performance Analysis of Systems and Software*, Austin, Texas, March 2006.

[2] D. Brooks, J.-D. Wellman, P. Bose, and M. Martonosi. Power-performance modeling and tradeoff analysis for a high-end microprocessor. In *Power Aware Computing Systems Workshop at ASPLOS-IX*, November 2000.

[3] T. Devlin and B. Weeks. Spline functions for logistic regrssion modeling. In *Proceedings of the Eleventh Annual SAS Users Group International Conference*, Cary, NC, 1986.

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

[5] D. B. et. al. Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors. *IEEE Micro*, 20(6):26–44, Nov/Dec 2000.

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

[7] F. Harrell, K. Lee, D. Matchar, and T. Reichert. Regression models for prognostic prediction: Advantages, problems and suggested solutions. *Cancer Treatment Reports*, 69:1071–1077, 1985.

[8] V. Iyengar, L. Trevillyan, and P. Bose. Representative traces for processor models with infinite cache. In *Proceedings of the 2nd Symposium on High Performance Computer Architecture*, February 1996.

[9] P. Joseph, K. Vaswani, and M. J. Thazhuthaveetil. Construction and use of linear regression models for processor performance analysis. In *Proceedings of the 12th Symposium on High Performance Computer Architecture*, Austin, Texas, February 2006.

[10] B. Lee and D.Brooks. Effects of pipeline complexity on smt/cmp power-performance efficiency. In *ISCA-32: Proceedings of the Workshop on Complexity Effective Design*, June 2005.

[11] R. Liu and K. Asanovic. Accelerating architectural exploration using canonical instruction segments. In *International Symposium on Performance Analysis of Systems and Software*, Austin, Texas, March 2006.

[12] M. Moudgill, P. Bose, and J. Moreno. Validation of turandot, a fast processor model for microarchitecture exploration. In *Proceedings of the IEEE International Performance, Computing, and Communications Conference (IPCCC)*, February 1999.

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

[14] S. Nussbaum and J. Smith. Modeling superscalar processors via statistical simulation. In *PACT2001: International Conference on Parallel Architectures and Compilation Techniques*, Barcelona, Sept 2001.

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

[16] C. Stone and C. Koo. Additive splines in statistics. In *Proceedings of the Statistical Computing Section ASA*, Washington, DC, 1985.

[17] R. D. Team. *R Language Definition.*

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

```
sqrt(bips)~(     # first-order effects
            rcs(depth,4) + width + rcs(gpr_phys,4)
            + rcs(dmem_lat,3) + load_lat + fix_lat + rcs(fpu_lat,3)
            + rcs(l2cache_size,3) + rcs(icache_size,3) + rcs(dcache_size,3)
            + il1miss_rate + dl1miss_rate + dl2miss_rate
            + br_rate + br_mis_rate
            + stall_dmissq + stall_cast + stall_reorderq
            + stall_resv + stall_rename
            + rcs(base_bips,5)
                 # second-order effects
                 # interactions of pipe dimensions and in-flight queues
            + rcs(depth,4) %ia% rcs(gpr_phys,4)
            + width %ia% rcs(gpr_phys,4)
                 # interactions of depth and hazards
            + rcs(depth,4) %ia% rcs(l2cache_size,3)
            + rcs(depth,4) %ia% dl2miss_rate
            + rcs(depth,4) %ia% br_mis_rate
                 # interactions of width and i-cache
            + width %ia% il1miss_rate
            + width %ia% br_rate
            + width %ia% br_mis_rate
                 # interactions of L1 cache size and access rates
            + rcs(icache_size,3) %ia% il1miss_rate
            + rcs(dcache_size,3) %ia% dl1miss_rate
            + rcs(dcache_size,3) %ia% dl2miss_rate
                 # interactions of L2 cache size and access rates
            + rcs(l2cache_size,3) %ia% il1miss_rate
            + rcs(l2cache_size,3) %ia% dl1miss_rate
            + rcs(l2cache_size,3) %ia% dl2miss_rate
                 # interactions of stalls with bottleneck locations
            + width %ia% stall_resv
            + width %ia% stall_rename
                 # diminishing marginal returns with higher base perf
            + rcs(depth,4) %ia% rcs(base_bips,5)
            + width %ia% rcs(base_bips,5)
            + rcs(l2cache_size,3) %ia% rcs(base_bips,5)
            + rcs(dcache_size,3) %ia% rcs(base_bips,5)
            + rcs(icache_size,3) %ia% rcs(base_bips,5)
                 # third-order effects
                 # depth interactions with memory hierarchy and stalls
            + rcs(depth,4):rcs(dcache_size,3):rcs(l2cache_size,3)
            + rcs(depth,4):rcs(dcache_size,3):dl1miss_rate
            + rcs(depth,4):rcs(l2cache_size,3):il1miss_rate
            + rcs(depth,4):rcs(l2cache_size,3):dl1miss_rate
            + rcs(depth,4):rcs(dmem_lat,3):dl2miss_rate
                # width interactions with memory hierarchy and stalls
            + width:rcs(icache_size,3):il1miss_rate
            + width:rcs(l2cache_size,3):dl1miss_rate
            + width:rcs(l2cache_size,3):il1miss_rate
            );
```

Figure 15: **General Model**. Model specification in standard R syntax. Restricted cubic splines for parameter p with k knots is denoted as `rcs(p,k)`. Interaction of non-linear and all terms is denoted by `%ia%` and `:`, respectively.

```
sqrt(bips)~(     # first-order effects
            rcs(depth,4) + width + rcs(gpr_phys,4)
            + rcs(dmem_lat,3) + load_lat + fix_lat + rcs(fpu_lat,3)
            + rcs(l2cache_size,3) + rcs(icache_size,3) + rcs(dcache_size,3)
                 # second-order effects
                 # interactions of pipe dimensions and in-flight queues
            + rcs(depth,4) %ia% rcs(gpr_phys,4)
            + width %ia% rcs(gpr_phys,4)
                 # interactions of depth and hazards
            + rcs(depth,4) %ia% rcs(l2cache_size,3)
                 # third-order effects
            + rcs(depth,4):rcs(dcache_size,3):rcs(l2cache_size,3)
            );
```

Figure 16: **Application-Specific Model**. Model specification in standard R syntax as in Figure 15.