# Statistically Rigorous Regression Modeling for the Microprocessor Design Space

Benjamin C. Lee, David M. Brooks
Harvard University
Division of Engineering and Applied Sciences
Cambridge, Massachusetts
{bclee,dbrooks}@eecs.harvard.edu

## Abstract

*Regression models enhance existing techniques in detailed microarchitectural simulation by reducing the number of required simulations and using simulation data more efficiently to identify trends and trade-offs. We present a rigorous derivation of such models for microprocessor performance and power prediction, emphasizing the need to apply domain-specific knowledge when performing statistical inference. In particular, we propose sampling observations uniformly at random from a large design space, discuss approaches for identifying statistically significant predictors, and detail strategies for effectively modeling predictor interaction and non-linearity. The resulting models enable computationally efficient statistical inference, requiring the simulation of only 1 in every 5 million points of a joint microarchitecture-application design space while achieving median prediction error rates as low as 4.1 percent for performance and 4.3 percent for power.*

## 1 Introduction

Although cycle-accurate simulators provide detailed insight into application performance on a wide range of microprocessor configurations, they are computationally expensive and cannot be used to effectively explore a large design space. The design space under consideration is often artificially constrained to accommodate the costs of simulation, leading to narrowly defined studies and conclusions that may not generalize. This is a fundamental challenge in simulator-based microarchitectural research that will become increasingly severe as research shifts toward multi-threaded, multi-core architectures. Introducing statistical inference into simulation frameworks modestly reduces detail for substantial gains in speed and tractability.

Techniques in statistical inference and machine learning have become increasingly popular for approximating solutions to intractable problems. 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 solving a system of linear equations and predicting the response requires evaluating a linear equation.

We survey applied statistical regression theory in Section 2. In Section 3, we draw from the statistical treatment of missing data to propose sampling uniformly at random from a large design space. We apply this statistical theory to demonstrate a rigorous approach to deriving regression models in Section 4 that includes (1) association testing, (2) variable clustering, (3) assessing strength of response-predictor relationships, and (4) significance testing with F-tests, to ensure statistically significant parameters are used as predictors. We also validate underlying regression assumptions, mitigating deviations with variance stabilizing transformations.

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

- **Performance Prediction:** Application-specific models predict performance with a median error as low as 4.1 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.

- **Power Prediction:** Regional models, leveraging only the most relevant samples, predict power with a median error as low as 4.3 percent. 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.

Collectively, these results suggest significant potential in accurate and efficient statistical inference for design space exploration via regression models. This paper details the model derivation in a recent technical report [1].

## 2 Regression Theory and Techniques

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). Mathematically, $\beta_j$ 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 $Var(e_i) = \sigma^2$.

$$y_i = \beta x_i + e_i = \beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} + e_i \tag{1}$$

The *method of least squares* is commonly used to identify the best-fitting model for a set of observations by determining $\beta$ to minimize $S(\beta)$, the sum of squared deviations between the predicted responses given by the model and the actual observed responses. $S(\beta)$ may be minimized by solving a system of $p + 1$ partial derivatives of $S$ with respect to $\beta_j$, $j \in [0, p]$. The solutions to this system, $\hat{\beta}_j$, are estimates of the coefficients in Equation (1).

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

### 2.1 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. This interaction may be modeled by constructing a third predictor $x_{i,3} = x_{i,1}x_{i,2}$ to obtain $y_i = \beta_0+\beta_1 x_{i,1}+\beta_2 x_{i,2}+\beta_3 x_{i,1}x_{i,2}+e_i$. Modeling predictor interaction in this manner makes it difficult to interpret $\beta_1$ and $\beta_2$ in isolation. After simple algebraic manipulations, 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 $\beta$ for more complex models lead us to prefer more indirect interpretations of the model via its predictions.

### 2.2 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.

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 [5]. 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. Cubic splines may have poor behavior in the tails before the first knot and after the last knot [12]. Restricted cubic splines that constrain the function to be linear in the tails are often better behaved.

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 [12]. 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 [12]. 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 [5].

### 2.3 Significance Testing

Given a model, we may wish to assess the significance of 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.

The *F-test* compares two nested models (*e.g.* a full model and a subset of the full model) using their multiple

correlation statistic $R^2$. Equation (3) computes this statistic by computing regression error ($SSE$) as a fraction of total error ($SST$) where $y_i$ and $\hat{y}_i$ are the true and modeled response, respectively. $R^2$ quantifies the percentage of variance in the response captured by the predictors.

$$R^2 = 1 - \frac{SSE}{SST} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \frac{1}{n}\sum_{i=1}^{n} y_i)^2} \quad (3)$$

Given $R^2$ for the full model and $R_*^2$ for the smaller model, define the *F-statistic* by Equation (4) where $p$ is the number of coefficients in the full model excluding the intercept $\beta_0$ and $k$ is the difference in degrees of freedom between the models.

$$F_{k,n-p-1} = \frac{R^2 - R_*^2}{k} \times \frac{n-p-1}{1-R^2} \quad (4)$$

The *p-value* is defined as $2P(X \geq |c|)$ for a random variable $X$ and a constant $c$. In our analyses, $X$ follows an F-distribution with parameters $k$, $n-p-1$ and $c$ is the F-statistic. The p-value is the probability an 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.

### 2.4 Assessing Fit

A model's 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 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 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.

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

Additionally, fit may be assessed by the $R^2$ statistic from Section 2.3. 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 [5].

### 2.5 Prediction

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.

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

## 3 Experimental Methodology

We use Turandot, a generic and parameterized, out-of-order, superscalar processor simulator [9, 10]. Turandot is enhanced with PowerTimer to obtain power estimates based on circuit-level power analyses and resource utilization statistics [2, 3]. The modeled baseline architecture is similar to the current POWER4/POWER5.

We use R, a free software environment for statistical computing, to script and automate the statistical analyses described in Section 2 [13]. Within this environment, we use the Hmisc and Design packages from Harrell [5].

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 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 [6].

| | 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 1. 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 $j$.**

### 3.1 Configuration Sampling

We report experimental results for sample sizes of up to $n = 4,000$ samples. Each sampled configuration is simulated with a benchmark also chosen uniformly at random, mapping predictors (configuration's parameter values) to observed responses (performance and power).

The approach to obtaining observations from a large microprocessor design is critical to efficient formulation of regression models. Table 1 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 $i$ is specified by a set of values, $S_i$. 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 each configuration 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 impractical despite continuing research to reduce simulation costs via trace sampling [4, 11]. Although these techniques reduce per simulation costs by a constant factor, they do not reduce the number of required simulations. Other studies have reduced these sets to an upper and lower bound ($|S_i| = 2$), but this approach masks performance and power trends between the bounding values and precludes any meaningful statistical inference or prediction [7, 14]. Thus, sampling must occur in the design space to control the exponentially increasing number of design points as the number of 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 high-resolution identification of trends and trade-offs. Furthermore, sampling UAR does not bias the observations toward any particular configuration. From the law of large numbers, [1] the expected number of samples with each configuration value is the same. Our approach is similar to Monte Carlo methods, which generate suitable random samples and observe their frequency distribution. Although we perform random sampling, we formulate non-parametric regression models instead of analyzing frequency.

### 3.2 Treatment of Missing Data

Suppose we treat the configurations for which responses are not observed as missing data from a hypothetical data set with all $|S|$ observations. 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 element. In the microarchitectural 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

[1] Theorems stating the difference between expected and actual values approaches zero as the number value-generating trials increases

| Performance | |
|---|---|
| instruction throughput (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. Application Predictors**

| 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.**

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.

# 4 Model Derivation

We consider 12 architectural predictors in 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. 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. Such potential interactions suggest both architectural and application-specific predictors are necessary.

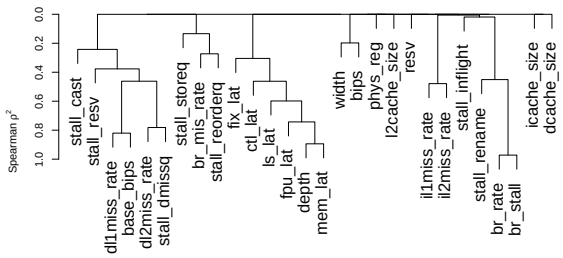


**Figure 1. Variable Clustering**

## 4.1 Variable Clustering

Variable clustering reveals key predictor interactions. Figure 1 presents the results of hierarchical variable clustering based on squared Spearman rank correlation coefficients where a larger $\rho^2$ indicates a greater correlation between variables. If over-fitting is a concern, redundant predictors may be eliminated within a cluster.

L1 and L2 misses due to instruction cache accesses are highly correlated. We found the absolute number of L2 cache misses from the L1 instruction cache accesses negligible and eliminate `il2miss_rate`. 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 [8]. 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.

## 4.2 Performance Associations

Plotting each of the predictors against the response may reveal particularly strong associations or identify nonlinearities. We stratify each predictor into four groups such that each group covers equally sized intervals of predictor values and plot the mean of each group's 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. The figures are effectively summarized scatterplots of predictors versus performance measured in billions of instructions per second (bips).

For architectural predictors, pipeline depth and width

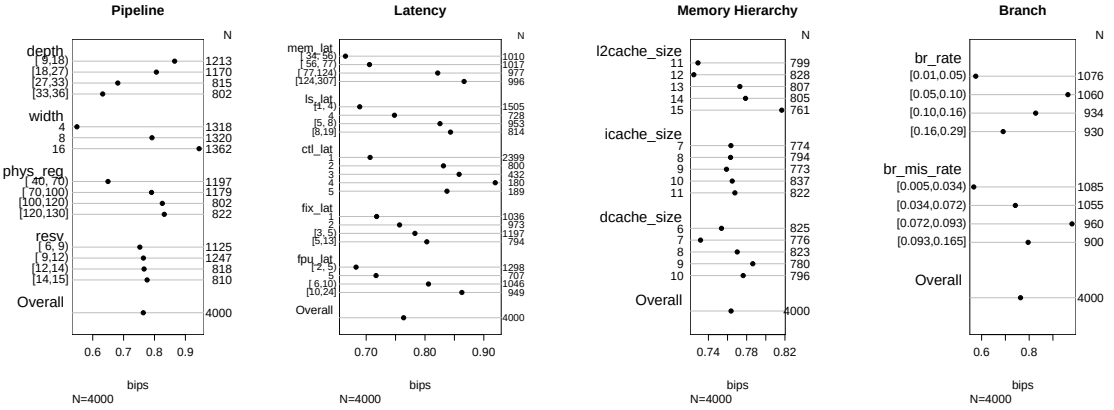


**Figure 2. Predictor Association. (1) Pipeline dimensions, (2) unscaled latencies, (3) memory hierarchy, (4) application baseline branching.**

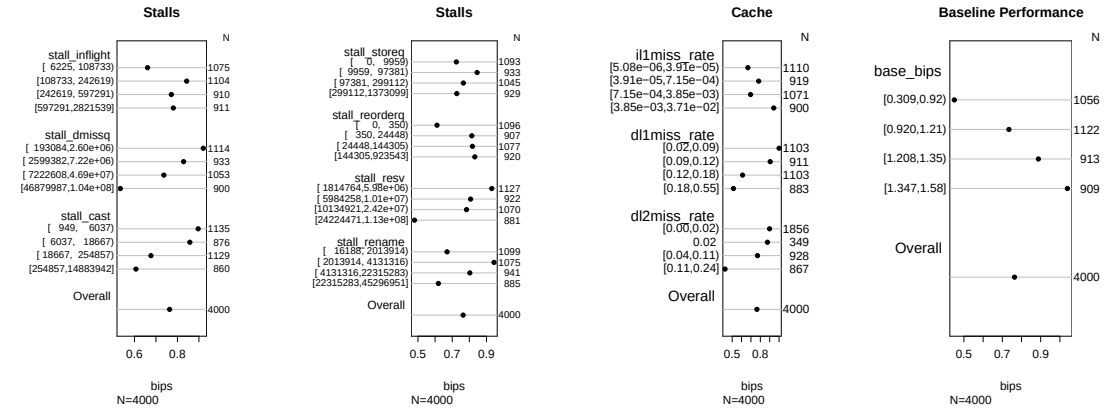


**Figure 3. Predictor Association. Application (1,2) stalls, (3) cache behavior, (4) baseline performance.**

are strong, monotonic factors (Figure 2.1). We also observe diminishing marginal returns on performance from increasing register file size. 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 (Figure 2.2). 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, we find a positive correlation between L2, but not L1, cache size and performance (Figure 2.3).

For application-specific predictors, branching behavior exhibits no obvious trends (Figure 2.4). Roughly half the stall characteristics (`dmissq`, `cast`, `reorderq`, `resv`) have monotonic relationships with performance (Figure 3.1,Figure 3.2). 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 (Figure 3.3). Lastly, a benchmark's observed sample and baseline performance are highly correlated (Figure 3.4).

### 4.3 Strength of Marginal Relationships

Figure 4 plots the non-monotonic generalization of the Spearman rank correlation coefficient for each of the prespecified predictor variables and performance. This information will guide our choice in the number of spline knots and predictor interactions. A lack of fit for predictors with higher $\rho^2$ 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. Application-specific baseline performance is the most significant predictor. Collectively, application-specific predictors are most highly correlated with performance indicating the application's interactions with the microarchitecture are primary determinants of performance. Microarchitectural predictors considered in isolation are less significant.

### 4.4 Model Specification

We model non-linearities for architectural predictors with restricted cubic splines. The strength of predictors' marginal relationships with performance in Figure 4 guides 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.* cache sizes, reservation stations). Despite their importance, width and latencies do not take a sufficient number of unique values to apply splines and we consider their linear effects

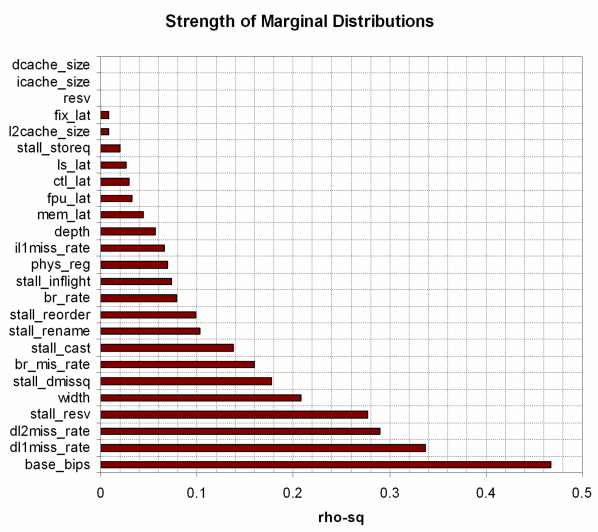


**Figure 4. Predictor Strength**

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 in the memory hierarchy (*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.

### 4.5 Model Refinement

Figure 5 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.[2] 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 5. 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 6.

In Table 4, 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`, `stall_inflight`, and `stall_storeq` appear insignif-

[2] Residuals are defined in Equation (5)

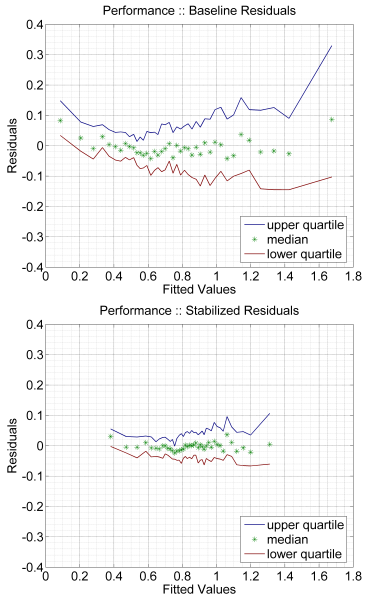


**Figure 5. Residual Correlations**

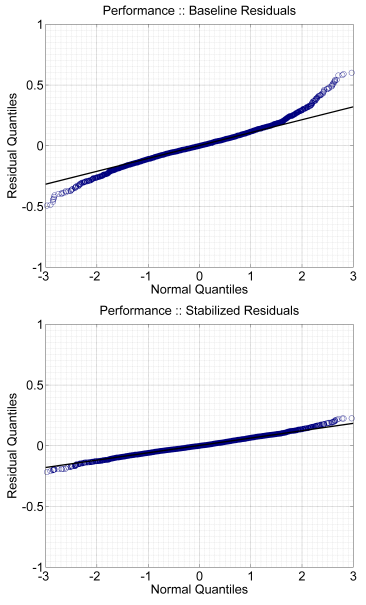


**Figure 6. Residual Distribution**

| 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 4. Predictor Significance Testing. * denotes inclusion in model.**

icant. 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.

## 5 Model Evaluation

We use subsets of the $n = 4,000$ random observations to formulate regression models, referring to the cardinality of these sets as the *sample size*, denoted by $n_*$. Each model may require different sample sizes to maximize accuracy. We also obtain 100 additional random validation samples for predictive queries. We compare and contrast the predictive ability of four regression models, differing in specification and data fit:

- **Baseline (B):** Model specified in Section 4 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 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 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 relative eu-

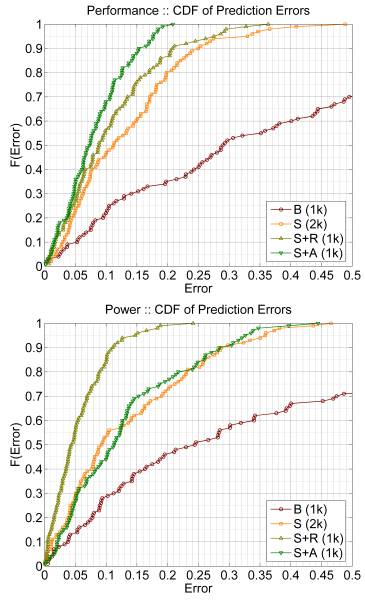


**Figure 7. Empirical CDF of Performance (T) and Power (B) Prediction Errors.**

clidean distance between two vectors of parameter values, $d = \sqrt{\sum_{i=1}^{p} |1 - b_i/a_i|^2}$.

- **Application-Specific (S+A):** We use a new set of $n_A = 4,000$ observations for varying configurations, but a fixed benchmark, eliminating application-specific predictors from the general model and reformulating the model with a naive subset of $n_{S+A} < n_A$. We consider six representative benchmarks (ammp, applu, equake, gcc, gzip, mesa).

### 5.1 Performance Prediction

Figure 7T plots the empirical cumulative distribution function (CDF) of prediction errors, specifying error minimizing sample sizes for each model in the legend. The error distribution is a more effective measure of accuracy when compared to the usually reported median and mean. Variance stabilization reduces median error from 13.1 to 10.9 percent; regional or application-specific models further reduce error. Application-specific models predict performance with the greatest accuracy. The representative equake-specific model achieves 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 indicate the number of outliers decrease with error. We found no significant correlations between outliers and their configurations.

### 5.2 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. Furthermore, an application's impact on dynamic power dissipation is a function of application-specific microarchitectural resource utilization.

Figure 7B plots the empirical CDF's of power prediction errors, emphasizing the differences in regional and application-specific modeling. Variance stabilization provides the first significant accuracy improvement, reducing median error from 22.0 to 9.3 percent. Regional modeling provides the second accuracy gain, reducing median error from 9.3 to 4.3 percent. 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.

### 5.3 Performance and Power Comparison

Although the best performance and power models achieve comparable median error rates, 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. Holding the application constant 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 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 deeper pipelines and linearly for shallower pipelines. Application-specific models add little accuracy since architectural configurations and resource sizings are primary determinants in unconstrained power. The effects of scaling unconstrained power to obtain dynamic power dissipation as a function of application-specific resource utilization may be relatively small.

## 6 Related Work

Yi, *et al.*, identify significant microprocessor parameters using Plackett-Burman designs. They suggest fixing non-critical parameters to reasonable constants and performing

extensive simulation by sweeping a range of values for critical parameters. We ensure model predictors are statistically significant, but instead of performing further simulation, we rely on regression models to explore the design space.

Joseph, *et al.*, derive performance models using stepwise regression, an automatic iterative approach to adding and dropping predictors from a model depending on measures of significance [7]. Although commonly used, stepwise regression has several problems cited by Harrell [5]. Furthermore, they consider only two values for each predictor and do not predict performance, using the models only for significance testing. In contrast, we leverage prior knowledge of microarchitectural design to derive predictive models.

Eeckhout, *et al.*, study statistical simulation for simplifying workloads in architectural simulation [4]. Nussbaum, *et al.*, examine similar statistical superscalar and symmetric multiprocessor simulation [11]. Both profile benchmarks to obtain relevant program characteristics, such as instruction mix and data dependencies between instructions, constructing a smaller, synthetic benchmark with similar characteristics. The statistical approach we propose and those 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 outputs (*i.e.*, performance and power results).

## 7 Conclusions and Future Directions

Regression 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 median error rates as low as 4.1 percent for performance and 4.3 percent for power. Despite a basic variance stabilizing transformation, our models produce non-random residuals. Different transformations on the response and predictors may mitigate the resulting bias. We use the same model specification for both performance and power and for each application-specific model regardless of application. A rigorous derivation to specify different models for each metric would require additional designer effort, but may improve accuracy.

Given their accuracy for random design points and low computational costs of obtaining predictions, we may pursue more aggressive design studies previously not possible via simulation. We also intend to compare regression accuracy and costs to those of other statistical techniques, such as machine learning and neural networks. Although the specific techniques differ, statistical techniques for 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.

## References


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

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

[3] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[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 *PACT2001: International Conference on Parallel Architectures and Compilation Techniques*, Barcelona, Sept 2001.

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

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

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