2009

# Processor design space exploration and performance prediction

Balachandran Ramadass
*Louisiana State University and Agricultural and Mechanical College*

# PROCESSOR DESIGN SPACE EXPLORATION AND PERFORMANCE PREDICTION

A Thesis
Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering
in
The Department of Electrical and Computer Engineering

by
Balachandran Ramadass
B.Tech, Pondicherry University, Pondicherry, India, 2005
August, 2009

**Acknowledgements**

I am pleased to thank every one who made this thesis Possible. I will remember, and appreciate their contributions forever. First, I deeply thank my advisor Dr. Lu Peng for his valuable advice and support. His constant motivation and long hours of discussion taught me to do high-quality research in the field of Computer Architecture. .

I would like to thank my Co-adviser Dr. Bin Li for his valuable guidance and suggestions towards the completion of this thesis. Throughout the process or working on my thesis, he has been very patient and supportive. I am grateful for all the help he has provided.

I would also like to thank my committee members Dr. J. Ramanujam and Dr. Jerry Trahan for their valuable feedback.

I would like to thank my colleague Tribuan Kumar Prakash, Lide Duan and Ran tao for their generous company and brainstorming technical discussions. I would also like to thank my best friends Sakthivel, Prasath, Jayanth and Deepak for their constant motivation. I would also like to thank my friends Karthik, Ashwin, Mahesh, Rajesh, Kim, keyyoung-park, Girish, Jagadish, Bhaskar, Archana, Rajiv, vikram, lohit, suresh, and many others for making my life enjoyable in Baton Rouge.

Finally I am thankful to my parents and family for their endless love and blessings, especially my mother Saraswathy Ramadass for her enormous support and devotion. Without her constant motivation and sacrifice I would not have achieved all of this in my life and I dedicate this thesis to her.

## Table of Contents

## List of Tables

## List of Figures

**Abstract**

The use of simulation is well established in processor design research to evaluate architectural design trade-offs. More importantly, Cycle by Cycle accurate simulation is widely used to evaluate the new designs in processor research because of its accurate and detailed processor performance measurement. However, only configuration in a subspace can be simulated in practice due to its long simulation time and limited resources, leading to suboptimal conclusions that might not be applied to the larger design space. In this thesis, we propose a performance prediction approach which employs a state-of-the-art technique from experimental design, machine learning and data mining. Our model can be trained initially by using Cycle by Cycle accurate simulation results, and then it can be implemented to predict the processor performance of the entire design space. According to our experiments, our model predicts the performance of a single-core processor with median percentage error ranging from 0.32% to 3.01% for about 15 million design spaces by using only 5000 initial independently sampled design points as a training set. In CMP the median percentage error ranges from 0.50% to 1.47% for about 9.7 million design spaces by using only 5000 independently sampled CMP design points as a training set. Apart from this, the model also provides quantitative interpretation tools such as variable importance and partial dependence of the design parameters.

# 1. Introduction

## 1.1 Overview

Exponential increase in transistor density inside the chip allows us to build chips with enhanced capability and functionality. Processor design industries have seen major growth in the past few decades such as out-of-order processor, super scalar processor, SMT processor, CMP processor, and clock frequency enhancement technology. These technologies, not only enhance the performance of the processor, but also increase the complexity of processor design and evaluation. This rapid technological advancement in chip design creates two major challenges in evaluating the new design space.

## 1.2 Slow Simulation Environment

Computer architects have to use complex simulation models. Usually, they evaluate their new designs using cycle-accurate processor simulators which provide them with insight details on processor performance, power consumption and complexity. The code complexity of these architecture simulators increases drastically due to the advancement in processor technology. As a result of this technological advancement, the simulator requires longer simulation time and larger computational resources.

## 1.3. Huge Design Space

This design space is composed of the product of choices of many architectural design parameters, such as frequency, issue width, cache parameters, different branch predictor settings, ALU size, ROB size, LSQ size etc. To understand the impact of design parameters and their interactions with a new design, the computer architect will have to evaluate the huge spectrum of design space before coming to any conclusions.

These two challenges render the evaluation of a new design over the entire design space using a

cycle-accurate simulator nearly impractical, so architects resort to analyzing their new designs for a selected important configuration in a subspace. These kinds of analyses might lead to sub-optimal conclusions which might not be applicable to the whole design space. In addition, more parameters brought by chip-multiprocessors (CMPs) make this problem more urgent.

In this thesis, we propose to use a state-of-art tree based predictive modeling technique combined with advanced sampling techniques from statistics and machine learning to predict processor performance for the entire design space based on initial independent training design points. This technique fills the gap between simulation requirement, simulation time and resource costs.

The proposed method includes the following four components: (1) the maximin space-filling sampling method that selects initial design representatives from a large amount of design alternatives; (2) the state-of-the-art predictive modeling method - Multiple Additive Regression Trees (MART) [8] which builds an ensemble of trees with high prediction accuracy; (3) an active learning method which adaptively selects the most informative design points needed to improve the prediction accuracy sequentially; (4) interpretation tools for MART-fitted models which are able to show the importance and partial dependence of design parameters and shed light on the underlying working mechanism. These provide computer architects a quantitative and efficient approach to optimize processor performance by tuning key design factors.

For each workload, 500 initial design points were sampled based on the maximin distance methods (detailed in Chapter 3.1). Then another 500 points were sampled according to an adaptive sampling scheme (described in Chapter 3.3). We repeat the sampling process until 3000 design points were sampled for each benchmark. An independent test set which consists of another 5000 points is used to evaluate the prediction performance of fitted models. After

sampling and testing, the interpretation is performed based on the fitted MART model with all the 3000 sampled points.

According to our experiments on 12 SPEC CPU2000 and 4 SPLASH2 benchmarks, by sampling 3000 points drawn from a microarchitecture design space with nearly 15 million configurations for each SPEC program and a CMP design space with nearly 9.7 million points for each SPLASH2 workload, we can summarize the following result:

**1.4. Performance Prediction**

Application-specific models predict performance, based on 5000 independently sampled design points, with median percentage error ranges from 0.32% to 3.01% for our single-core processor performance prediction in a design space with about 15 million points and 0.50% to 1.47% for the CMP performance prediction in a design space with about 9.7 million points.

**1.5. Worst-Case Performance**

For the single-core processor performance prediction, the worst percentage errors are within 7% for 10 out of the 12 benchmarks; the largest worst-case percentage error is 22.56% for art. In the CMP performance study, the worst percentage errors are within 16.1% for all the 4 benchmarks.

**1.6. Relative Performance**

Compared to a classical linear regression model with a random sampling scheme, our method typically reduces 88% and 98% average percentage error for the single-core and CMP study respectively. Our method also has 8% and 37% less average percentage error in the single-core and CMP study separately than multiple additive regression trees with a random sampling scheme.

**1.7. Model Interpretation**

The model can be used to explain variable importance and partial dependence to each variable.

For the two selected benchmarks, we find that “Width/ALU” and “L2Size” are key factors to bzip2 while “LSQ” is important to MCF. Tuning these factors helps to improve processor performance to these programs. The partial dependence plots clearly illustrate processor design trends and bottlenecks for the processor.

The remainder of this thesis is organized as follows. Chapter 2 describes the related work. Chapter 3 introduces the MART-aided methodology. Chapter 4 describes the experimental setup, while experiment results are presented in Chapter 5. The model interpretation tools and model discussion are demonstrated in Chapter 6. Chapter 7 concludes this thesis.

## 2. Related Work

As we know, the transistor budget has been increasing exponentially after advancements in circuit design technology. Over the past decades, architects are concentrating more on building detailed simulation environments to evaluate their new designs. Such detailed simulation results are more accurate and reliable but also for certain far-reaching new ideas, building a simulation environment is more complex and time consuming. Design space exploration and performance prediction have been emerging as hot problems recently. Researchers explore various performance predictive modeling techniques to reduce the painstakingly slow simulation times and resource constraints. These models can predict the expected behavior of future systems before they are developed and such analysis could reduce the burden of building unnecessary wasteful systems. Martonosi and Skadron et al [19] pointed out that research in multiprocessor (MP) systems are affected by the difficulty in building complex simulation environments. The number of MP system research papers which appeared in ISCA (International Symposium of Computer Architecture) was 50% during 1985 and was reduced to 10% in 2001. Their NSF workshop conclusion includes developing scientific methods for abstracting evaluations to explore large design spaces accurately and efficiently. They recommended developing more abstract modeling techniques that includes analytical, statistical and neural network models that provide deeper insight into system behavior. Kumar et al [14] found out that the design decisions without the interaction of interconnections were more likely wrong when the interconnect interactions were considered. So sensitivity studies of larger design space are essential for making optimal conclusions.

### 2.1 Design Space Exploration

Li *et al.* [17] explored the multi-dimensional design space across a range of possible chip sizes

and thermal constraints for both CPU-bound and memory-bound workloads. They showed the inter-related nature of parameters such as core count, pipeline depth, super scalar width, L2 cache size, operating voltage and frequency. The methodology used by Li *et al.* [18] was, decoupling core and interconnect or cache simulation to reduce simulation time. They used single core L2 cache-access traces along with the zauber; a cache simulator they developed to model the interaction of multiple threads on one or more shared interconnects.

Xi *et al.* [29] explored 3D design spaces. Interconnects have become a major challenge in the achievement of good performance with given power budgets for the microprocessors. The authors considered 3D architectures as an attractive solution for this problem; they also focused on building CAD tools and architectural techniques to explore the design space of 3D.

**2.2 Analytical, Neural Networks and Statistical Models**

Karkhanis *et al.* [12] proposed an analytical performance model for superscalar processors. They used trace-derived data dependence information, data and instruction cache miss rate and branch miss prediction rates as inputs to the model and estimated the performance of a superscalar processor. The error rates are with 5.8 % of detailed simulation on average and 13 % in the worst case. Building analytical models required the collection of micro-architectural statistics. These techniques are useful exclusively to evaluate the performance of closely related designs points not for the entire design space.

Ipek *et al.* [9]attacked the design space exploration problem with a neural network based predictive modeling technique. They predicted the performance of memory subsystems, processors and chip multiprocessors. The authors used intelligent machine learning methods to sample a small number of design points from the larger design space to train the artificial neural networks (ANNs) model and continued training their model with sample points until the

prediction error estimates dropped sufficiently low. They predicted the entire design space with (98-99 % for uniprocessed studies and 95-96% for CMP study) by training their model with only 1-2 % of the entire design space.

Lee *et al*. [15] proposed regression models to efficiently predict the performance and power. They considered prediction based on both linear and nonlinear regression models as well as model inference such as significance testing for each design parameter and assessing goodness of fit. They also used regression models in Pareto frontier analysis, pipeline depth analysis and multiprocessor heterogeneity study [16]. They randomly sampled 4000 design points from the entire design space of 22 billion points and their application specific model predicted performance with median error as low as 4.1 percent. 50 to 90 % prediction achieved error rates of less than 10 percent depending on the applications. Their regional model predicted power with median error rates as low as 4.3 percent and 90 percent of prediction achieved error rate of less than 10 percent.

Joseph *et al*. [10] developed linear regression models that characterized the interaction between processor performance and microarchitectural parameters. They built the models by using Akaike's Information Criteria (AIC) directed iterative processes by which significance of the parameters to the CPI performance were ordered. They further approached this problem with nonlinear regression models using the functional approximation capabilities of radial basic function networks (RBF). They used Latin hypercube sampling to select design points for training their model. They obtained a prediction error rate of 2.8% [11].

Yi *et al.* [30] used Plackett and Baurman design to determine the most important parameter and used this design to reduce the simulation time.

Phansalkar *et al.* [21] approached this problem by exploring program similarity. They

used principal components analysis (multivariate statistical data analysis technique) for workload composition and benchmark sub-setting. They demonstrated that a subset of 8 programs can be effectively representing the entire SPEC CPU 2000 suite. Ould-Ahmed-Vall *et al*. [20] used a model tree to analyze the performance of a subset of SPEC2006 running on an Intel Core 2 Duo processor.

Li, Peng and Ramadass[17] approached the microarchitecture design space exploration and performance prediction problem using MART Model. These measurements are parallel to my work done upon which the thesis is based.

Compared with the analytical model, classical linear regression model, ANNs and RBF-networks, our proposed method has the following features: 1) this method is particularly well suited for the discrete (either ordinal or nominal variables) design space parameters; 2) MART achieves extremely accurate prediction which is supported by both lots of empirical evidence and theoretical proofs; 3) this method is highly robust to the tuning parameter values (needs minimal knowledge to tune the model); and 4) it also comes with model interpretation tools such as the measure of variable importance and the partial dependence plot, which provide computer architects a quantitative view for design alternatives and may shed light on the underlying mechanism.

## 3. Methodology and Background

We propose to use the nonparametric tree-based predictive modeling method combined with advanced sampling techniques from statistics and machine learning to explore the microarchitectural design space efficiently. The fitted model based on hundreds of regression trees can be summarized, interpreted and visualized similarly to conventional regression models.

### 3.1 Maximin Distance Design

In experiment design, the distance-based space-filling sampling methods are popular, especially, when we believe that interesting features of the true model are just as likely to be in one part of the experimental region as another. Among them, the maximin distance design is commonly used. The maximin distance criterion chooses a subset of design points (from the entire design space) in which the smallest pairwise distance is maximized.

In our study, some architectural design parameters are nominal with no intrinsic ordering structure and the others are discrete with a small number of values (see Table 1). Hence, we define the following distance measure used in our study. Let $wt_j$ be the weight for the jth design parameter. The distance between design points $x_1$ and $x_2$ are defined as

$$d(\mathbf{x}_1, \mathbf{x}_2) = \sum_{j=1}^{p} \left[ wt_j \times I(x_{1j} \neq x_{2j}) \right] \qquad (2.1)$$

where $wt_j = \log_2(\text{number of levels for } j^{th} \text{ design parameter})$ and $I(A)$ is an indicator function, equal to one when A holds, otherwise zero. Note that the weight for each design parameter is equal to its information entropy with uniform probability for each of its possible values.

### 3.2 Regression Model with MART

#### 3.2.1 Regression Tree

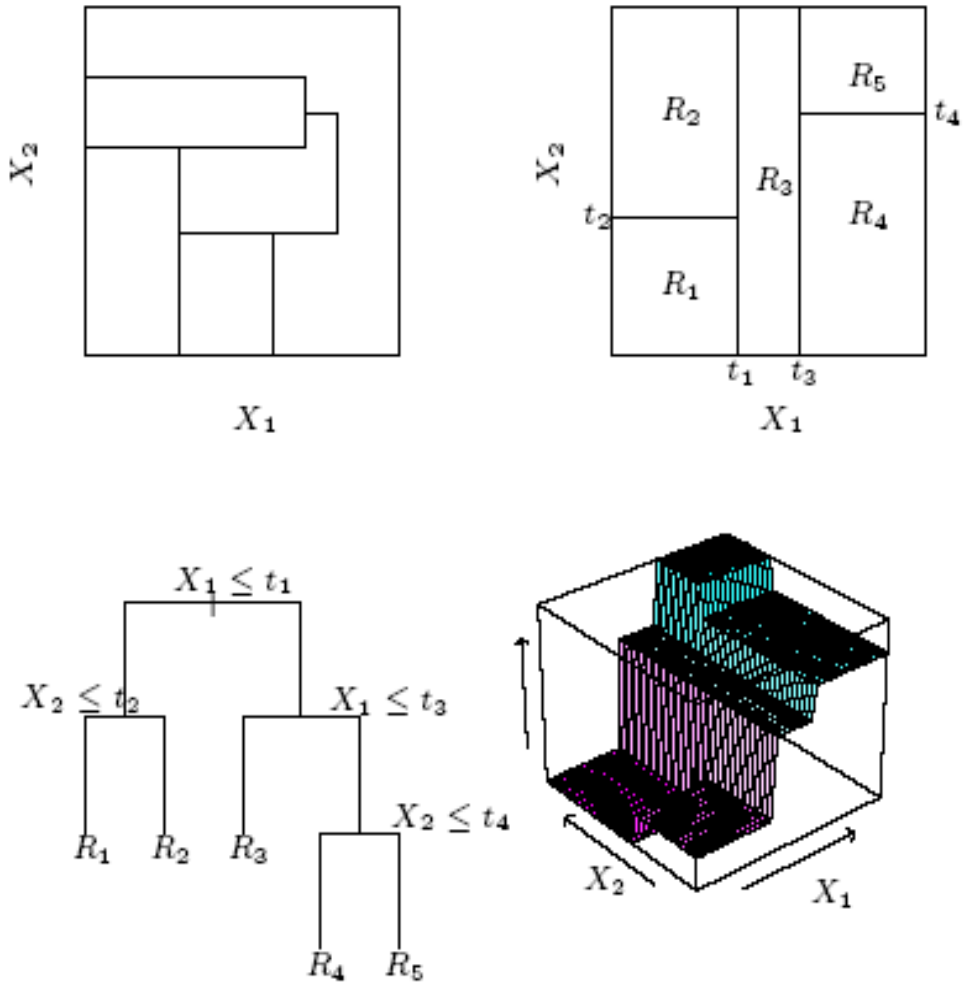


Fig.1. Regression Tree

The regression tree has been used as a tool for exploring multivariate data sets for some time. As in multiple linear regression, the technique is applied to a data set consisting of a continuous response variable y and a set of predictor variables {x1, x2,..., xk}. However, instead of modeling y as a linear function of the predictors, as shown in Fig.1, regression tree models y as a series of 'if-then-else rules based on values of the predictors.

### 3.2.2 MART (Multiple Additive Regression Tree)

MART is one of several techniques that aim to improve the performance of a single model by fitting many models and combining them for prediction. Fig.2, shows an illustration of MART in Regression. MART consists of two parts: classification and regression trees (CART [1]) and boosting technique.

CART analysis consists of two basic steps. The first step consists of tree building, during which a tree is built using recursive binary splitting. The term "binary" implies that we first split the space into two regions, and model the response by a constant for each region. Then we choose a variable and split-point to achieve the best fit again on one or both of these regions.

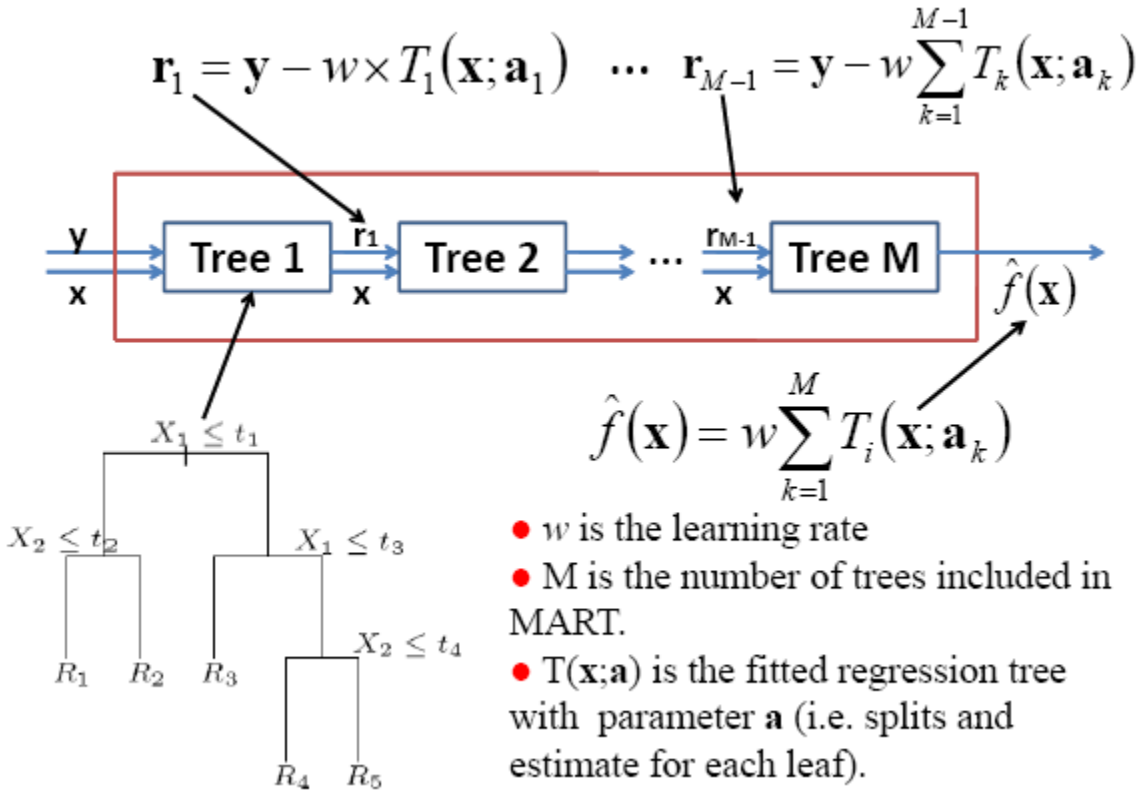


Fig.2. Illustration of MART in Regression

Thus, each node can be split into two child nodes, in which case the original node is called a

parent node. The term “recursive” refers to the fact that the binary partitioning process can be applied over and over again until some stopping criterion is reached. Each resulting node is assigned a value, which is based on the distribution of the observations in the training set that fall in that node. The second step consists of tree “pruning", which results in the creation of a sequence of simpler trees, through cutting off the weakest links.

Tree-based methods are popular because they represent information in a way that is intuitive and easy to visualize, and have several other advantageous properties. First, a tree is inherently nonparametric and can handle mixed-type of input variables naturally, i.e. no assumptions are made regarding the underlying distribution of the values for the input variables, e.g. numerical data that are highly skewed or multi-modal, as well as categorical predictors with either ordinal or non-ordinal structure. This eliminates researchers’ time which would otherwise be spent in determining whether variables are normally distributed, and making transformations if they are not. Second, a tree is adept at capturing non-additive behavior, i.e. complex interactions among predictors are routinely and automatically handled with relatively few inputs required from the analyst. This is in marked contrast to some other multivariate nonlinear modeling methods, in which extensive input from the analyst, analysis of interim results, and subsequent modification of the method are required. Third, a tree is insensitive to outliers, and unaffected by monotone transformations and differing scales of measurement among inputs. Despite these benefits, a tree is not usually as accurate as its competitors, and small changes in training data can result in very different series of splits [8].

Boosting is a general method for improving model accuracy of any given learning algorithm, based on the idea that it is easier to find an average of many rough rules of thumb than to find a single highly accurate prediction rule[25]. The idea of boosting has its roots in PAC (Probably

Approximately Correct) learning[28]. Kearns and Valiant [13] proved the fact that "weak" learners, each performing only slightly better than a random decision, can be combined to form a "powerful" learner. In boosting, models (e.g. trees) are fitted iteratively to the training data, using appropriate methods to gradually increase emphasis on observations modeled poorly by the collection of trees. Boosting algorithms vary in how they quantify lack of fit and select settings for the next iteration.

In MART, boosting is a form of "functional gradient descent". MART approximates the underlying function F(x) by an additive expansion

$$\hat{f}(x) = \sum_{m=1}^{M} \beta_m b(x;\gamma_m) \qquad (2.2)$$

where the expansion coefficients $\{\beta_m\}_1^M$ and the tree parameters $\{\gamma_m\}_1^M$ are jointly fitted into the training data. The model is fitted in a forward "stagewise" (not "stepwise") fashion, meaning that existing trees are left unchanged as the model is enlarged. In MART, one starts with an initial guess $\hat{f}_0(\mathbf{x})$ and then for each iteration $m = 1,\cdots,M$:

1) Compute the negative gradient $\{g_i\}_{i=1}^n$ as the working

$$g_i = \frac{\partial L(y_i,\mathbf{x_i})}{\partial f(\mathbf{x_i})}\Big|_{f(\mathbf{x_i})=\hat{f}(\mathbf{x_i})}, \qquad (2.3)$$

where $L(y, \hat{f})$ is some loss function we expect to minimize.

2) Fit a regression tree $b(\mathbf{x};\gamma)$ and a gradient descent step size β that minimizes

$$\sum_{i=1}^{n} L\left(y_i, \hat{f}_{m-1}(\mathbf{x}_i) + \beta b(\mathbf{x}_i;\gamma)\right). \qquad (2.4)$$

3) Update the estimate of F(x) as

$$\hat{f}_m(\mathbf{x}) = \hat{f}_{m-1}(\mathbf{x}) + \beta b(\mathbf{x};\gamma). \qquad (2.5)$$

From a user's perspective, MART, as applied in this paper, has the following features.

First, the model fitting process is stochastic – i.e. it includes a random or probabilistic component. The stochasticity improves predictive performance, reducing the variance of the final model, by only using a random subset of data to fit each new tree [7]. This means that, unless a random seed is set initially, final models will subtly differ each time they are run. Second, the gradient descent step determines the contribution of each tree to the growing model. Studies in [6] show that using small values of gradient descent step size always lead to better prediction performance. Hence, we fix $\beta$ at 0.01 in this study. Third, the tree complexity controls whether interactions are fitted: a tree with depth of one ("stump" with only two terminal nodes) fits an additive model without including any interaction. In this study, we set the maximum depth for trees at three, which fits a model with up to three-way interactions. Finally, the choice of M, i.e., when to stop the boosting algorithm, is based on monitoring the estimation performance on out-of-bag samples, the set of observations not selected (from the training data) to fit trees within each iteration.

In this paper, MART is run by using the **gbm**, an **R** implementation of the MART package produced by Greg Ridgeway [22].

### 3.3 Adaptive Sampling

Adaptive sampling, also known as active learning in machine learning literature, involves sequential sampling schemes that use information gleaned from previous observations to guide the sampling process. Several empirical and theoretical studies have shown that adaptively selecting samples in order to learn a target function can outperform conventional sampling schemes. For example, Freund *et al.* [5], Saar-Tsechansky and Provost [23] and Sung and P. Niyogi [27] all employed this method.

In this paper, we select the 500 subsequent samples in the following four steps:

1) Apply MART on the sampled points 20 times with different random seeds.

2) For each of the MART-fitted model, predict the unsampled points in the design space.

3) Sort these points (in a decreasing order) according to the coefficient of variance (CoV, the ratio of standard deviation to mean) for the model prediction.

4)Set an initial distance threshold θ. First, include the least confident point (one with the largest CoV value). Then, include the next least confident points if its distance to the first point is above θ. After that, we include the least confident points if its distances to the first two points are above θ. Repeat the procedure. If all 500 points have been selected, increase the value of distance threshold θ slightly and restart the selection procedure until we reach the maximum threshold that allows us to select all 500 points.

The intuition of selecting the subsequent samples as above is based on the bias-variance decomposition. Note that the decomposition is originally proposed for the squared loss, but can be generalized to other losses such as zero-one losses for classification [3]. In practice, since the bias is unknown before measuring, we can only measure the variance of predictions. However, there are two concerns. First, we often care more about the relative scale than the absolute loss in practice. This makes us to sort the design points based on their CoV rather than the variances. Second, if we select the points strictly according to their CoV, the selected ones are often clustered. In other words, they tend to be close to each other in the design space. In order to achieve global accuracy, we try to select sampling points as separated as possible, although the ones with large CoV should have higher preference to be selected.

**3.4 Stopping Criterion**

The procedure is stopped based on either the time/cost constraint or the performance measure. The former depends on investigators' time and cost budget, while for the latter, we can monitor

the procedure based on a cross-validation measure or prediction performance on an independent test set. Namely, if the prediction accuracy is above a pre-specified level and/or the improvement of prediction accuracy is below a pre-specified level, the procedure will be stopped. Since we consider the stopping issue is more on the user-end, in this study, we fix the total number of points we need to sample throughout the paper.

### 3.5 Interpreting the Model

Even producing a model with hundreds to thousands of trees, MART does not have to be treated like a black box. A MART model can be summarized, interpreted and visualized similarly to conventional regression models. This involves identifying those parameters that are most influential in contributing to its variation and visualizing the nature of dependence of the fitted model on these important parameters.

The relative variable importance measures are based on the number of times a variable is selected for splitting, weighted by the squared improvement to the model as a result of each split, and averaged over all trees. The relative influence is scaled so that the sum adds to 100, with higher numbers indicating stronger influence on the response.

Visualization of fitted functions in a MART model is easily achieved using a partial dependence function, which shows the effect of a subset of variables on the response after accounting for the average effects of all other variables in the model. Given any subset $\mathbf{x}_s$ of the input variables indexed by $\mathbf{s} \subset \{1, \cdots, p\}$. The partial dependence of $f(\mathbf{x})$ is

$$F_s(\mathbf{x}_s) = E_{\mathbf{x}_{\setminus s}}[f(\mathbf{x})] \qquad (2.6)$$

where $E_{\mathbf{x}_{\setminus s}}[.]$ means the expectation over the joint distribution of all the input variables with index not in $\mathbf{s}$.

In practice, partial dependence can be estimated from the training data by

$\hat{F}_{\mathbf{s}}(\mathbf{x}_{\mathbf{s}}) = (1/n)\sum_{i=1}^{n} \hat{f}(\mathbf{x}_{\mathbf{s}}, \mathbf{x}_{i\setminus\mathbf{s}})$, where $\{\mathbf{x}_{i\setminus\mathbf{s}}\}_1^n$ are the data values of $\mathbf{x}_{\setminus\mathbf{s}}$.

## 4. Experimental Setup

We have two sets of experiments: single core processor simulation and CMP simulation. For single core simulation, we modified the sim-outorder, the out-of-order pipelined simulator in Simplescalar [2], to be an eight-stage Alpha-21264 like pipeline. Only twelve (eight integer and four floating point) CPU and memory intensive programs from SPEC2000 were selected due to limited simulation time and resource. They are art, bzip2, crafty, equake, fma3d, gcc, mcf, parser, swim, twolf, vortex and vpr. We skipped a number of instructions for each SPEC program based on a previous work[24]. Then we collected the number of execution cycles for the next 100 million instructions. Since we use generic simulators and benchmarks, we believe that the validated model can be applied to other simulators and workloads. Table 1(a) lists 13 groups of design parameter choices for an OoO executed pipelined processor. The cross-product of these choices is about 15 million design points per benchmark, representing an intractably large space to fully simulate.

For CMP simulation, we employed SESC [26] which has detailed out-of-order processor simulation for each core and memory subsystem. We downloaded pre-compiled binaries of four SPLASH2 applications from [26]. They are barnes, fmm, radiosity and raytrace. During execution of these programs, the number of slave threads was set the same as the number of cores. We collected execution cycles for 100 million instructions. For results of multiple cores, we took the average number of execution cycles as CMP's execution cycles. Table 1(b) lists 14 groups of design parameter choices for a CMP. For each program, the design space size is about 9.7 millions. Fig. 3 shows a scheme plot of our sampling, modeling and interpretation procedure. For each workload, 500 initial design points were sampled based on the maximin distance criterion described in Chapter 3.1. Then another 500 points were sampled according to the

Table 1
(a) Single Core Processor Simulation Parameters

| Parameters | Values |
|---|---|
| Fetch/Issue/Commit Width | 2 / 4 / 8 instructions |
| Branch Predictor:<br>Bimod predictor<br>or<br>2-level predictor<br><L1>/<L2>/<History size> | Bimod predictor – 4096 entries or 8192 entries<br>2-level predictor:<br>1/4096/10, 1/8192/10,<br>2/4096/10, 2/8192/10,<br>4/4096/10, 4/8192/10 |
| Branch Target Buffer <Number of sets>/<Assoc> | 1024 / 4, 2048 / 2,<br>1024 / 8, 2048 / 4 |
| Integer/Floating ALUs | 1/1, 2/1-related to issue width 2<br>2/1, 2/2-related to issue width 4<br>2/2, 4/4-related to issue width 8 |
| Register Update Units | 64 / 128 / 256 entries |
| Load Store Queue | 16 / 32 / 64 entries |
| L1 Inst. Cache Size | 8K, 16K, 32K, 64KB |
| L1 Data Cache Size | 8K, 16K, 32K, 64KB |
| L2 Unified Cache Size | 256K, 512K, 1024K, 2048KB |
| Memory Latency | 100, 140, 180, 220, 260 cycles |
| L1 I/D and L2 Block Size | 32B, 64B, 128B |
| L1 I/D Cache Associativity | 1, 2, 4 |
| L2 Cache Associativity | 4, 8, 16 |

(B) CMP Simulation Parameters*

| Parameters | Values |
|---|---|
| Core Configuration | In order, Out of order |
| Issue Width | 1,2,4 |
| Number of Cores | 1,2,4,8 |
| Off Chip Bandwidth | 4,8,16,24,32 bytes/cycles |
| Memory Latency (cycles) | 200,250,300,350,400,450,500 |
| L2 Cache Size | 1, 2, 4, 8 MB |
| L2 Cache Block Size | 32, 64,128 B |
| L2 Cache Associativity | 1,2,4,8,16 way |
| L2 Cache Hit Latency | 7,9,11,13 cycles |
| L2 Replacement Policy | LRU, Random |
| L1 I/D Cache Size | 32 KB, 64 KB |
| L1 I/D Cache Block Size | 32, 64B |
| L1 I/D Cache Associativity | 1,2,4 |
| Branch Predictor | Hybrid , 2level |

*L2 cache is shared and L1 I/D cache are private to each core.

adaptive sampling scheme described in Chapter 3.3

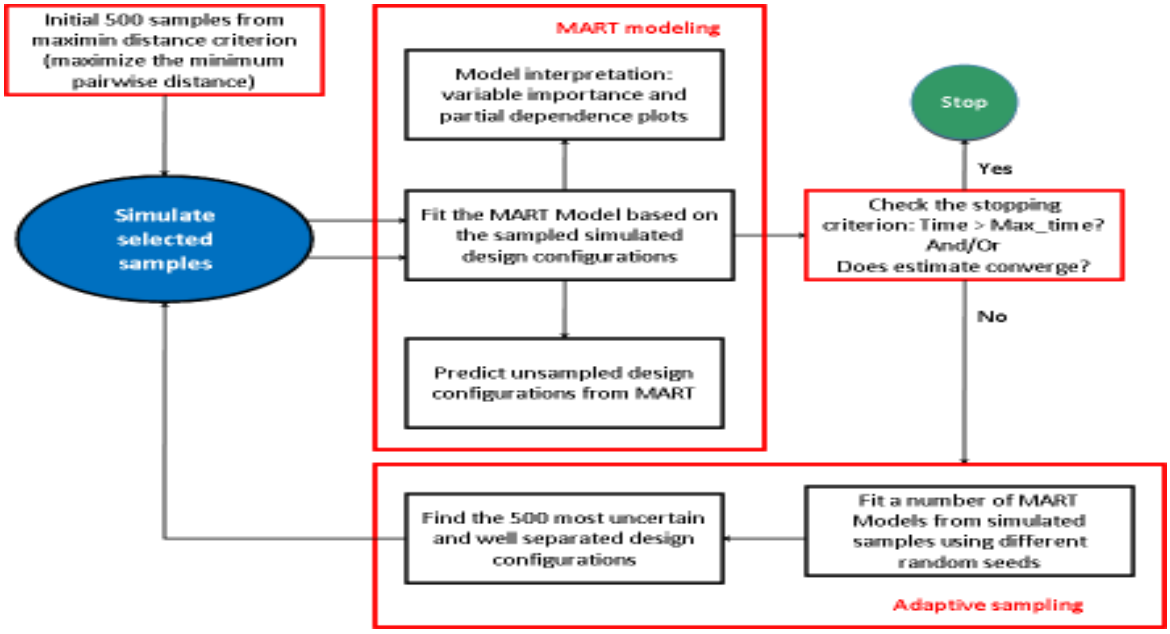


Fig. 3. Overview of the Proposed MART Model

We repeated the sampling process until 3000 design points were sampled for each benchmark. Notice that for the 3000 points, we only explored approximately 0.02% of the total 15 million points in the design space for the single-core study and about 0.03% of the 9.7 million points. An independent test set consisting of another 5000 points was used to evaluate the prediction performance of fitted models. The interpretation was based on the fitted MART model with all the 3000 sampled points.

## 5. Prediction Performance Evaluation

Table 2 shows the average and maximum percentage error (PE)

$$PE = \left| \left( \text{Predicted} - \text{Actual} \right) / \text{Actual} \right| \times 100\% \qquad (4.1)$$

of our method on an independent test set with 5000 design points. We see that the average PE

Table 2
Summary of relative prediction performance with specified error and sample size

| Benchmark | Single-Core Study | | | | | |
|---|---|---|---|---|---|---|
| | 0.0067% Sample (1000 pts.) | | 0.013% Sample (2000 pts.) | | 0.02% Sample (3000 pts.) | |
| | **Mean PE** | **Max PE** | **Mean PE** | **Max PE** | **Mean PE** | **Max PE** |
| art | 6.30 | 42.79 | 4.63 | 24.95 | 4.18 | 22.56 |
| bzip2 | 0.73 | 4.50 | 0.46 | 3.16 | 0.41 | 3.33 |
| crafty | 1.62 | 13.11 | 1.02 | 7.17 | 0.87 | 5.53 |
| equake | 2.65 | 18.69 | 2.26 | 15.77 | 2.13 | 15.04 |
| fma3d | 0.91 | 5.43 | 0.70 | 3.36 | 0.62 | 2.96 |
| gcc | 0.74 | 4.04 | 0.49 | 3.02 | 0.43 | 2.26 |
| mcf | 0.67 | 4.99 | 0.50 | 4.22 | 0.46 | 4.24 |
| parser | 0.83 | 4.90 | 0.52 | 3.65 | 0.42 | 2.30 |
| swim | 1.44 | 9.59 | 0.90 | 5.94 | 0.66 | 4.63 |
| twolf | 1.83 | 10.36 | 1.38 | 7.53 | 1.23 | 6.31 |
| vortex | 1.36 | 13.07 | 0.93 | 7.11 | 0.80 | 6.88 |
| vpr | 0.98 | 6.93 | 0.62 | 4.53 | 0.53 | 4.32 |
| **Benchmark** | **CMP Study** | | | | | |
| | 0.01% Sample (1000 pts.) | | 0.02% Sample (2000 pts.) | | 0.03% Sample (3000 pts.) | |
| | **Mean PE** | **Max PE** | **Mean PE** | **Max PE** | **Mean PE** | **Max PE** |
| barnes | 2.53 | 17.26 | 2.05 | 13.43 | 1.90 | 13.91 |
| fmm | 2.17 | 15.71 | 1.72 | 11.78 | 1.45 | 10.15 |
| raytrace | 1.12 | 21.90 | 0.83 | 17.36 | 0.75 | 16.02 |

decreases as the training set sizes increase (via adaptive sampling). In the single-core study, by sampling only 0.02% of sample points from the design space, the mean average PE's are within 1% for 9 out of 12 benchmarks. For *art*, which achieves least accuracy, the mean average PE is 4.18%. Similarly, in the CMP study, by sampling only 0.03% of sample points from the design

space, the mean average PE's are all within 2% for the four multithreaded benchmarks.

Lee and Brooks [15] proposed a regression modeling method for performance prediction. In their paper, they used the similar design space as ours. Their mean percentage error was 4.9%. İpek *et al*. [9] proposed a neural network based active learning method to explore the design space.

Although their design spaces are quite different from ours, they reported 4-5% error on average on the CMP study based on training the model on a 1.03% sample drawn from the design space (about 2500 sampled design points). Hence, we conclude that our method has a highly compatible mean PE compared to those two methods.

One advantage for the tree-based methods is that they are highly stable and robust to the outliers (or extreme values). A common way to evaluate the robustness of a method is to check the worst-case performance. Table 2 also shows the maximum percentage errors among the 5000 testing points. By exploring 0.02% of the design space in the single-core study, the maximum PE is less than 10% in 10 out of 12 benchmarks. Even in the least accurate benchmark, *art*, the maximum PE is 22.55%. In the CMP study, the maximum PE is within 20% for the 4 benchmarks. As comparison, the maximum PE, in Lee and Brooks[15], is 20.298%. In the CMP study from İpek *et al*. [9], their maximum PE ranges from about 34% to 53%. Therefore, our method has a comparable maximum PE to Lee and Brooks' work and a highly compatible worst-case performance to İpek *et al*.'s approach. This shows strong robustness of our models.

Fig. 4 shows the average PE's (black line) with different number of sampled points in the single-core study. The *x*-axis represents the portion of full parameter space simulated to form the training sets, and the *y*-axis stands for the average percentage error rate across the independent test set. The gray lines represent the mean plus/minus one standard deviation of PE's. We see

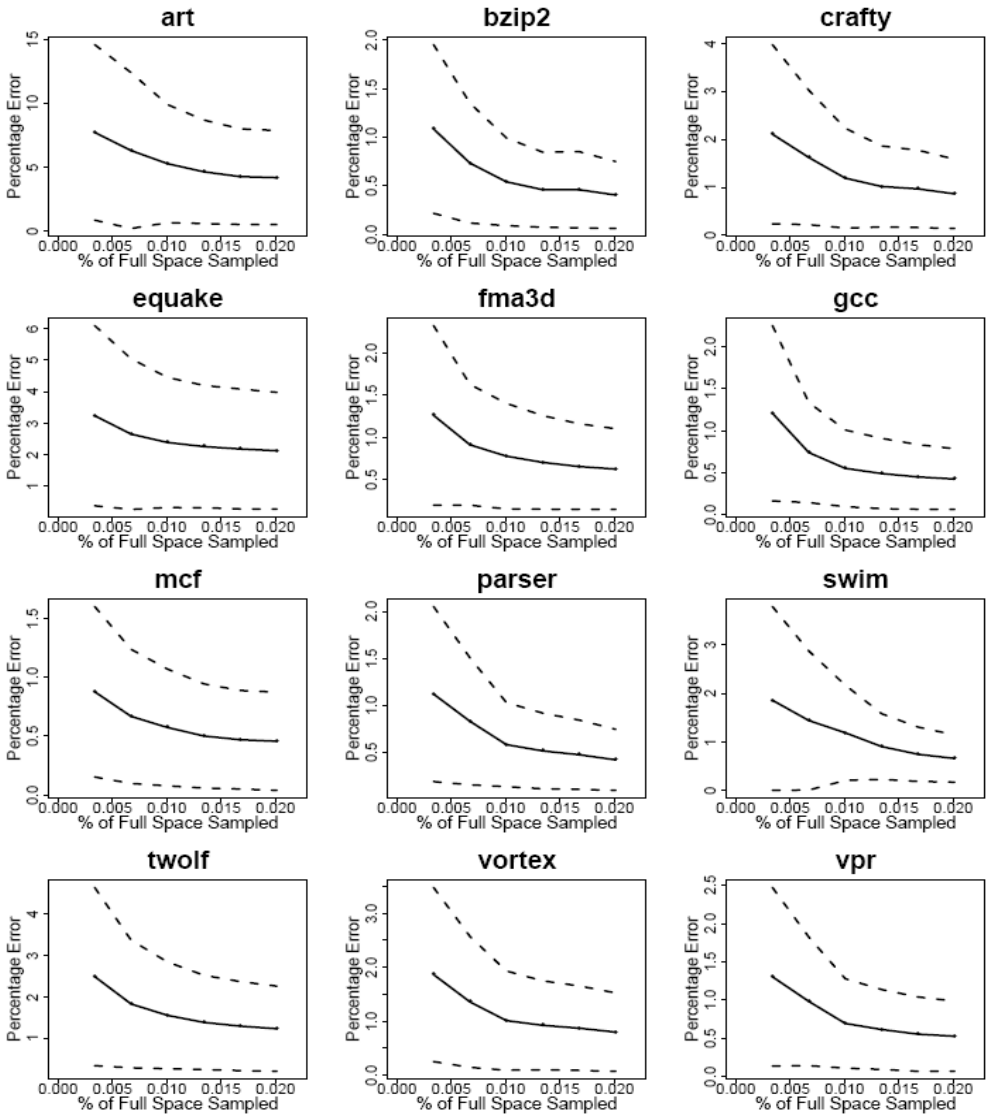

Fig. 4. Prediction accuracy of the models on the design space

the average percentage error rate decreases monotonically as the training set size increases (via adaptive sampling).

Fig. 5 depicts the empirical CDF plots of percentage errors (on test set) with about 0.02% sampled points in the single-core study. The *x*-axis shows the PE, and the *y*-axis shows the percentage of data points that achieve error less than each *x* value. We see that for 9 out of 12 benchmarks, more than 90% of the points are predicted with less than 2% error.

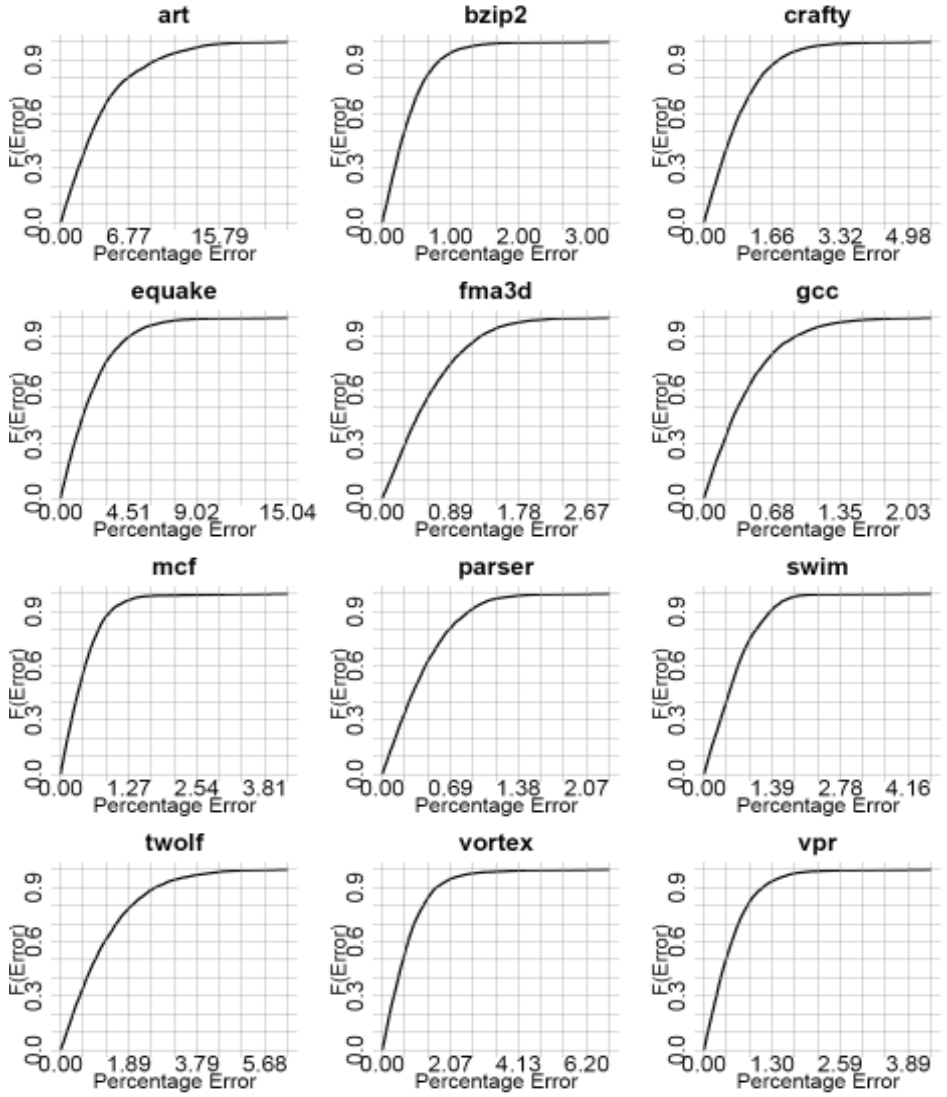


Fig. 5. Empirical CDF of prediction errors (single-core study)

The three with largest percentage errors are *art*, *equake* and *twolf*. For these three workloads, more than 90% of the points are predicted with less than 10%, 5% and 3% error respectively. Figure 6 and 7 illustrate the average PE's (with standard deviations) across different number of sampled points and CDF plots in the CMP study.

Note that for all the four benchmarks, 90% of the points are predicted with less than 4.2% error.

The reasons that our method achieves high prediction accuracy are twofolds: (1) the superior prediction performance inherited from MART;

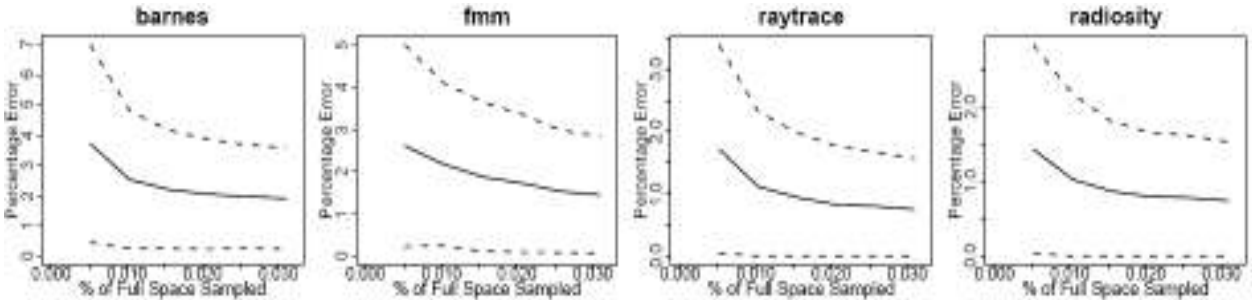


Fig. 6. Prediction accuracy of the models on the design space (adaptive sampling in the CMP study)

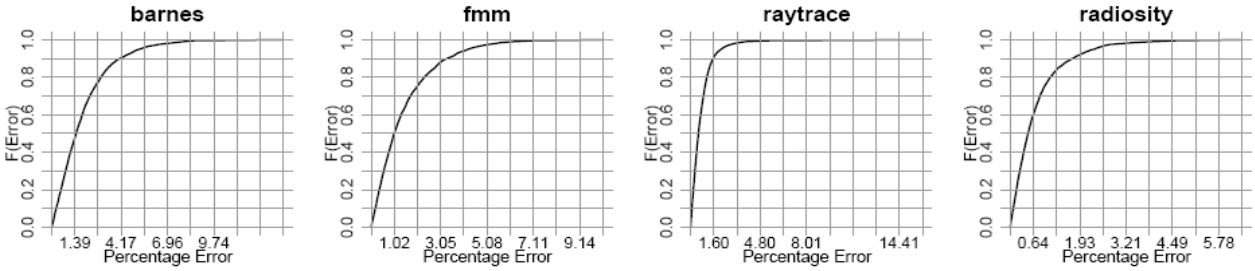


Fig. 7. Empirical CDF of prediction errors for the CMP performance study

(2) and the proposed adaptive sampling scheme. To illustrate the robustness of our model, we compare the predictive ability of our method with another two regression models, differing in specification and data used to perform the fit. The two models are listed as follows.

1) R+M: Fitting MART on the randomly selected sample points.

2) R+L: Fitting traditional linear regression model with all possible two-way interactions on randomly selected sample points.

Table 3 lists the relative percentage errors in R+L and R+M against our proposed method. The numbers in Table 3 are the ratios of the average and maximum PE (based on the test set) from the specified model and the proposed method.

Note that in Table 3, the larger the numbers, the more improvements from using our proposed method. If the ratio is equal to one, it means that the averages PEs are the same. From Table 3, we have the following two conclusions: (1) The ratios for R+L are larger than those for R+M in both studies, which means using MART substantially improves the prediction performance under the random sampling scheme. This is due to the fact that MART is adept at capturing nonlinear

Table 3
Summary of relative predictive accuracy (against A+M) with specified sample size

| Benchmark | Single-Core Study | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 0.013% Sample (2000 pts.) | | | | 0.02% Sample (3000 pts.) | | | |
| | R+L | | R+M | | R+L | | R+M | |
| | Mean PE | Max PE | Mean PE | Max PE | Mean PE | Max PE | Mean PE | Max PE |
| art | 4.05 | 3.76 | 1.12 | 1.09 | 4.44 | 3.89 | 1.00 | 1.04 |
| bzip2 | 7.40 | 4.13 | 1.21 | 1.31 | 8.27 | 4.02 | 1.11 | 1.01 |
| crafty | 7.62 | 6.02 | 1.11 | 1.41 | 8.64 | 7.41 | 1.10 | 1.50 |
| equake | 2.74 | 2.50 | 1.04 | 1.00 | 2.81 | 2.54 | 1.04 | 0.94 |
| fma3d | 3.25 | 3.79 | 1.07 | 1.22 | 3.56 | 4.32 | 1.04 | 1.14 |
| gcc | 9.63 | 7.25 | 1.07 | 1.52 | 10.89 | 9.47 | 1.10 | 1.59 |
| mcf | 10.90 | 4.99 | 1.06 | 1.15 | 11.84 | 4.58 | 1.10 | 1.05 |
| parser | 7.83 | 4.06 | 1.06 | 0.90 | 9.45 | 6.33 | 1.08 | 1.19 |
| swim | 14.76 | 8.99 | 1.06 | 0.87 | 20.11 | 10.67 | 1.04 | 0.78 |
| twolf | 3.55 | 4.26 | 1.05 | 1.20 | 3.90 | 4.93 | 1.04 | 1.44 |
| vortex | 3.87 | 2.87 | 1.11 | 1.67 | 4.38 | 2.77 | 1.11 | 1.40 |
| vpr | 13.25 | 6.89 | 1.14 | 1.03 | 15.26 | 7.45 | 1.18 | 1.02 |
| **Benchmark** | **CMP Study** | | | | | | | |
| | 0.02% Sample (2000 pts.) | | | | 0.03% Sample (3000 pts.) | | | |
| | R+L | | R+M | | R+L | | R+M | |
| | Mean PE | Max PE | Mean PE | Max PE | Mean PE | Max PE | Mean PE | Max PE |
| barnes | 25.68 | 15.20 | 1.38 | 1.89 | 27.55 | 12.67 | 1.37 | 1.95 |
| fmm | 48.50 | 27.79 | 1.66 | 2.19 | 57.86 | 31.26 | 1.83 | 2.24 |
| raytrace | 65.58 | 10.18 | 1.64 | 1.15 | 72.29 | 10.12 | 1.69 | 1.09 |
| radiosity | 65.57 | 24.39 | 1.42 | 1.98 | 70.36 | 23.62 | 1.50 | 2.01 |

Table 4: Summary of Bootstrap interval for mean and median PE

| Benchmark | Mean PE | 95% BI on Mean PE | Median PE | 95% BI on Median PE |
|---|---|---|---|---|
| art | 4.18 | (4.05,4.31) | 3.01 | (2.90, 3.11) |
| equake | 2.13 | (2.08, 2.19) | 1.62 | (1.57, 1.67) |
| barnes | 1.90 | (1.85, 1.94) | 1.47 | (1.42, 1.51) |

and non-additive behavior, e.g. nonlinear dependence and interactions among independent variables are routinely and automatically handled. (2) The ratios of relative prediction performance for R+M are greater than 1 in most of cases of the single-core study and all the cases in the CMP study. The benefit of using the adaptive sampling scheme is obvious, especially in the CMP study. For example, with 0.03% of the sampled points, using the proposed adaptive

sampling scheme reduces more than 37% of the average PE for all of the four benchmarks and more than 95% worst-case PE for three out of the four benchmarks. Multithreaded programs running on CMPs generates high variability that can be easier caught by our adaptive sampling method compared to random sampling approach. The comparisons illustrate the robustness of our proposed method.

## 6. Model Discussion, Interpretation and Visualization

### 6.1 Model Discussion

#### 6.1.1 Statistical Justification of Test Set Size

In this study, we evaluated the prediction performance based on 5000 independently and randomly selected test samples. Compared with the overall design space, it is still a small set. A natural question is: whether the test sample size is large enough to have an accurate evaluation of the prediction performance. Hence, we applied a bootstrapping technique to justify our choice of test sample size. Bootstrapping developed by Efron [4] is a general tool of estimating statistical properties of an estimator (e.g. the 95 percent confidence interval for the median and mean percentage errors). By using bootstrapping, we can estimate the confidence interval for the mean and median PE based on 5000 test samples. Note that the confidence interval based on bootstrapping does not assume any distribution assumption on the population (e.g. normal distribution assumption on PE's). However, bootstrapping procedure needs resampling the samples (in this case, they are the 5000 test samples) with replacement a large number of times (say 1000 times). We can check whether the test sample size is large enough based on the width of the bootstrap interval. Namely, if the width of the bootstrap interval (BI) is small, it indicates the test sample size is large enough and the mean and median PE based on this test samples have small sampling variation. Table 4 shows the 95 percent bootstrap interval based on 1000 bootstrapped samples for the mean and median PE in three benchmarks, which have the highest mean PE in single-core and CMP studies. We see that both upper and lower confidence limits for the 95% bootstrapped intervals are very close to their corresponding point estimates. For example, the mean PE for *art* is 4.18 which is very close to its upper and lower limits for the 95% bootstrap interval on mean PE. This indicates that the test sample size is large enough to have an accurate estimate of prediction performance in the study.

### 6.1.2 Sensitivity Study of Sampling Set Size

Like other regression methods, MART typically predicts better when trained on more data. On the other hand, data collection in architecture design space exploration is expensive, and a tradeoff exists between number of simulations and model accuracy. As we mentioned in Chapter 3.4, determination of the training sample size is an end-user issue. Namely, the stopping criterion is based on either the investigator's time and cost budget or convergence of prediction performance. Fig. 8 shows two typical curves of the percent of improvement in the single-core and CMP studies, which is defined as the proportion of improvement for each additional batch over the total improvement (in terms of the mean PE in the test set) in six batches. For example, suppose the mean PE based on the first 500 points (first batch) is 0.11. The mean PE based on the first 1000 points (two batches) is 0.06. The mean PE based on the total 3000 points (six batches) is 0.01. Then the total improvement is 0.11-0.01=0.1. The improvement based on the second batch is 0.06-0.01=0.05. Hence, the proportion of improvement for the second batch is then (0.05/0.1)×100%=50%. Based on Fig. 8, we see that the percent of improvement after the

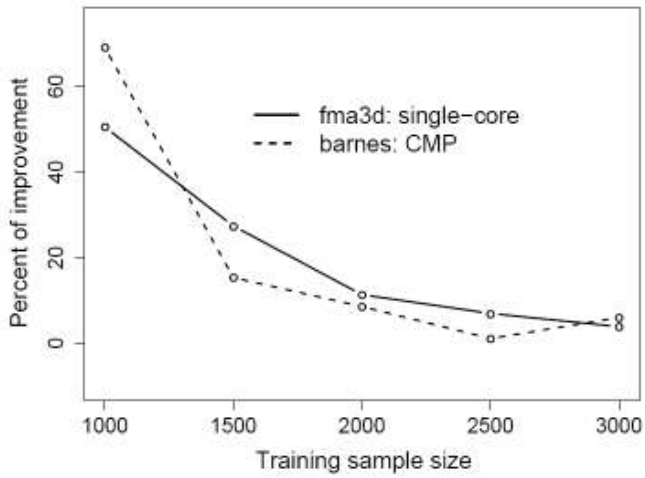

Fig. 8 Sensitivity of Training Set Size

fourth batch (2000 design points) is relatively small comparing to the first three batches in both single-core and CMPs studies. Hence, the results suggest the reasonable number of training

samples to be 2000 for a large simulation study.

### 6.2 Variable Importance

The proposed method can easily be used to analyze the importance of individual design parameters. This illustrates an inside view and provide computer architects with efficient directions of improving processor performance. We select two workloads *bzip2* and *mcf* in the single-core study as an example. Fig. 9 shows the relative variable influence, scaled to have a sum added to 100. For *bzip2* shown in the left figure, the most important variables are

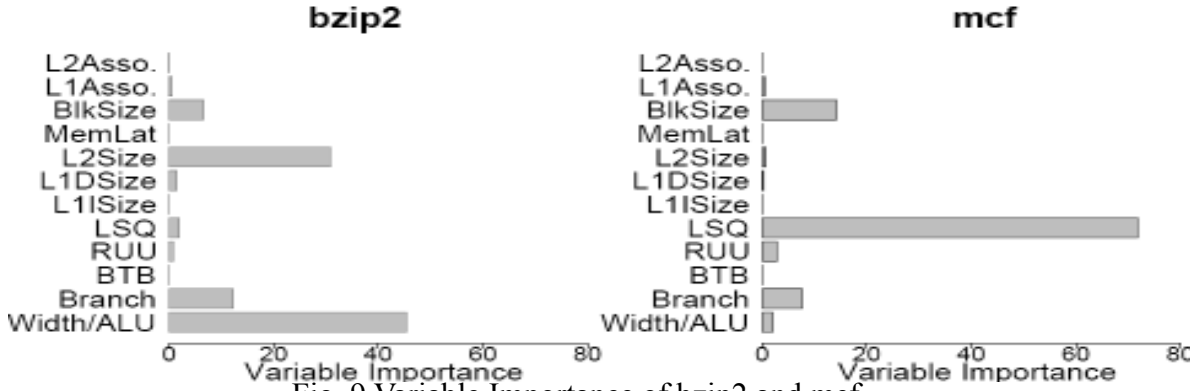


Fig .9 Variable Importance of bzip2 and mcf

"Width/ALU" and "L2Size" while "LSQ" is the most important factors for *mcf*. From these figures, we can see that CPU intensive programs such as *bzip2* are very sensitive to the instruction issue width and integer/floating point units as well as L2 cache size. Increasing these parameters provides an efficient way to improve processor performance. *Mcf* presents another sensitivity preference on LSQ size. This is reasonable because *mcf* has a considerable percentage of L2 cache misses due to its intensive pointer chasing. These outstanding load instructions tend to exhaust LSQ entries. The right part of Fig. 9 indicates that tuning LSQ entries will obtain greater performance benefits than other design parameters. Similarly, the variable importance method can also apply to multi-core processors. We found that the number of cores is the most

important parameters with an importance factor over 90 in average. This indicates that thread-level parallelism is the key factor for CMPs performance.

### 6.3 Partial Dependence Plot

Another advantage of the proposed model is that visualization and interpretation of fitted functions in a MART model can be achieved through partial dependence plots even though functions fitted by the MART models can be highly variable in shape and are frequently non-linear. The partial dependence plots can provide computer architects with visible interactions between different design parameters and performance trends and bottlenecks. From the plots, they can select configurations with optimized performance given a cost budget. As an example, we illustrate the two-dimensional partial dependence plot of the execution cycle of *mcf* to the two most important variables in Fig. 10. From this figure, we can see that a processor with a large LSQ size and cache block size tends to have high performance (the white region marked by "H"). On the other hand, the bottom left region marked by "L" indicates low performance configurations with a large execution cycle suffering from the small sizes of the LSQ and cache blocks. Moreover, we can see the tradeoffs between the design alternatives from this figure. For example, to reach a performance design goal which demands about 6.5e+08 for the execution cycle of *mcf*, one can design a processor with the following alternatives for the LSQ and the cache block size:

(1) the LSQ size equals to 26 and the cache block size is larger than 88 (part A of the line marked with "6.5e+08");

(2) the LSQ size ranges from 26 to 46 and the cache block size equals to 88 (part B of the line);

(3) the LSQ size equals to 46 but the cache block size ranges from 32 to 88 (part C of the line).

With the help of this tool, computer architects can make judicious choices with other constraints

from power, cost and complexity.

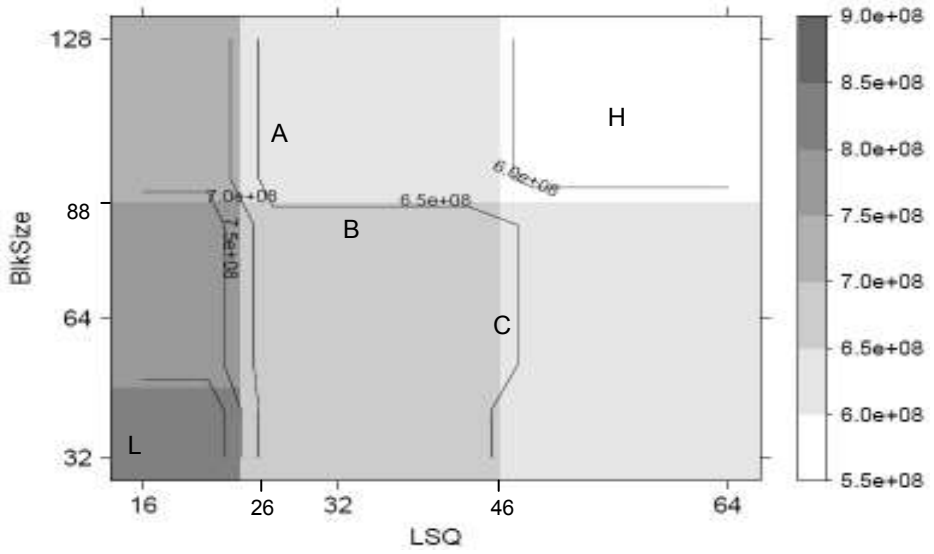


Fig. 10. The two-dimensional partial dependence plot of the execution cycle of *mcf* to the most important variables: *"LSQ Size"* and *"Cache Block Size"*.

## 7. Conclusion

In this thesis, we proposed a MART model that exploits micro-architectural design space and predicts performance of a single-core processor and a CMP. This model samples up to 0.02% of the full design space for a single-core processor with about 15 million points but achieves a very high accuracy. The median percentage error rate, based on an independent 5000 test points, ranges from 0.32% to 3.12% in 12 SPEC CPU2000 benchmarks. For a CMP design space with about 9.7 million points, the median percentage error is limited to a range from 0.50% to 1.89%. These results show that our model has highly compatible prediction performance to recently proposed regression and neural network models. The comparison of worst case prediction also shows that our model has stronger robustness. In addition, our model reflects performance trends and bottlenecks by showing the importance and partial dependence of processor design parameters.

## References


[1] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*, Wadsworth, 1984.

[2] D. Burger and T. Austin, "The SimpleScalar Tool Set, Version 2.0," Technical Report #1342, CS Department, University of Wisconsin-Madison, June 1997.

[3] P. Domingos, A Unified Bias-Variance Decomposition and its Applications. In *Proc. 17th International Conf. on Machine Learning*, 231-238, 2000.

[4] B. Efron, "The Jackknife, The Bootstrap, and Other Resampling Plans," *Philadelphia, Penn.: Society for Industrial and Applied Mathematics*, 1982.

[5] Y. Freund, H. S. Seung, E. Shamir, and N. Tishby. Information, prediction, and query by committee. In Proc. *Advances in Neural Information Processing Systems*, pages 483-490, 1993.

[6] J. Friedman. Greedy function approximation: a gradient boosting machine. *The Annals of Statistics*, 29: 1189-1232, 2001.

[7] J. Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38: 367-378, 2002.

[8] T. Hastie, R. Tibshirani and J. Friedman. *The Elements of Statistical Learning*, Springer, NY, 2001.

[9] E. İpek, S.A. McKee, B.R. Supinski, M. Schulz and R. Caruana. Efficiently exploring architectural design spaces via predictive modeling. *Twelfth Intl. Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS XII),* San Jose, CA, Oct. 2006.

[10] P. Joseph, K. Vaswani, and M. Thazhuthaveetil, "Use of linear regression models for processor performance analysis," In *Proc. 12th IEEE Symposium on High Performance Computer Architecture (HPCA-12)*, pages 99–108, Feb. 2006.

[11] P. Joseph, K. Vaswani, and M. Thazhuthaveetil, "A Predictive Performance Model for Superscalar Processors," In *Proc. Intl. Symp. on Microarchitecture*, Dec. 2006.

[12] T.Karkhanis and J.Simith. A First order superscalar processor model. *In Proc.31st IEEE/ACM International symposium on Computer Architecture*. 338-349, 2004

[13] M. Kearns and L.G. Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. *Journal of the Association for Computing Machinery*, 41: 67–95, 1994.

[14] R.Kumar, V.Zyuban and D.Tullsen. Interconnections in multi-core architectures: understanding mechanism, overheads and scaling. *In Proc.32nd IEEE/ACM International symposium on computer Architecture*, 408-419, 2005.

[15] B. Lee, D. Brooks, "Accurate and efficient regression modeling for microarchitectural performance and power prediction," *Twelfth Intl. Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS XII),* San Jose, CA, Oct. 2006.

[16] B. Lee, D. Brooks, "Illustrative Design Space Studies with Microarchitectural Regression Models," *In Proc. 13th Intl. Symposium on High Performance Computer Architecture (HPCA-13)*, Feb. 2007.

[17] B. Li, L. Peng and B. Ramadass. Efficient MART-Aided Modeling for Microarchitecture Design Space Exploration and Performance Prediction. *In Proceedings of 2008 ACM International conference on Measurement and Modeling of Computer Systems(SIGMETRICS)*, Jun 2008.

[18] Y.Li, B.LEE, D.Brooks, Z.Hu and K.Skadron. CMP design space exploration subject to physical constraints. *In Proc. 12th IEEE symposium on High Performance Computer Architecture*. 15-26, 2006.

[19] M.Martonosi and K. Skadron.NSF computer performance evaluation workshop: summary and action items, http://www.princeton.edu/~mrm/nsf_sim_final.pdf, 2001.

[20] E. Ould-Ahmed-Vall, J. Woodlee, C. Yount, K. A. Doshi and S. Abraham, "Using Model Trees for Computer Architecture Performance Analysis of Software Applications," *In Proc. 2007 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS),* Apr. 2007.

[21] A.Phansalkar, A.Joshi, L.Eeckhout and L.Jhon. Measuring Program similarity: Exoerunebts with SPEC CPU benchmark suites. *In Proc.IEEE International Symposium on Performance Analysis of Systems and Software*. 10-20, 2005.

[22] G. Ridgeway, "Generalized Boosted Models: A guide to the gbm package" http://cran.r-project.org/web/packages /gbm/vignettes/gbm.pdf, 2007.

[23] M. Saar-Tsechansky and F. Provost. Active learning for class probability estimation and ranking. In *Proc. 17th International Joint Conference on Artificial Intelligence*, pages 911-920, Aug. 2001.

[24] S. Sair, M. Charney, "Memory Behavior of the SPEC2000 Benchmark Suite," Tech. Report, IBM Corp. Oct. 2000.

[25] R. Schapire, "The Bosting Approach to Machine Learning - An Overview," *In MSRI Workshop on Nonlinear Estimation and Classification*, (eds D.D. Denison, M.H. Hansen, C. Holmes, B. Mallick & B. Yu), Springer, NY, 2002.

[26] SESC, http://sesc.sourceforge.net/.

[27] K. Sung and P. Niyogi. Active learning for function approximation. In Proc. *Advances in Neural Information Processing Systems*, vol. 7, pages 593-600, 1995.

[28] L.G. Valiant. A theory of the learnable. *Communications of the ACM*, 27: 1134-1142, 1984.

[29] Y.Xie, G.Loh, B.Black and K.Bernstein. Design Space exploration for 3D Architectures. *ACM journal on Emerging Technologies in Computing Systems*, 2, 2, 65-103, 2006.

[30] J.Yi, D.Lilja and D.Hawkins. A statistically-rigrous approach for improving simulation methodology. *In Proc. 9th IEEE Symposium on High Performance Computer Architecture*. 281-291, 2003.

## Vita

Balachandran Ramadass was born in Pondicherry, India. He graduated from High School in April 2000 with first class. He received his Bachelor of Technology in Electronics and Communication Engineering degree from Pondicherry University, India, in April 2005 with distinction. He also worked as a Junior Engineer in Network administration department, SIFY Limited, Tidal Park, Chennai, India, from April 2006-July 2006.

He is currently doing his Master of Science in Electrical and Computer Engineering Degree in Louisiana State University. He was also worked as a Graduate Technical Intern in Enterprise Power Path finding Group/DEG, INTEL Corporation, Dupont, Washington, for summer 2008 and fall 2008. He was also a Research Assistant in the Department of Electrical and Computer Engineering from May 2007-May 2008 and January 2009- June 2009.