Efficient System Design Space Exploration Using Machine Learning Techniques

Berkin Ozisikyilmaz, Gokhan Memik, Alok Choudhary
Department of Electrical Engineering and Computer Science
Northwestern University, Evanston, IL 60208
{boz283, memik, choudhar}@eecs.northwestern.edu

ABSTRACT

Computer manufacturers spend a huge amount of time, resources, and money in designing new systems and newer configurations, and their ability to reduce costs, charge competitive prices and gain market share depends on how good these systems perform. In this work, we develop predictive models for estimating the performance of systems by using performance numbers from only a small fraction of the overall design space. Specifically, we first develop three models, two based on artificial neural networks and another based on linear regression. Using these models, we analyze the published Standard Performance Evaluation Corporation (SPEC) benchmark results and show that by using the performance numbers of only 2% and 5% of the machines in the design space, we can estimate the performance of all the systems within 9.1% and 4.6% on average, respectively. Then, we show that the performance of future systems can be estimated with less than 2.2% error rate on average by using the data of systems from a previous year. We believe that these tools can accelerate the design space exploration significantly and aid in reducing the corresponding research/development cost and time-to-market.

Categories and Subject Descriptors

C.4 [**Computer Systems Organization**]: Performance of Systems, H.2.8 [**Database Applications**] Data Mining.

General Terms

Performance, Design.

Keywords

Design space, machine learning, performance prediction.

1. INTRODUCTION

Computer manufacturers spend tremendous effort in designing new desktop/server/laptop systems each year to gain advantage in a market that is worth hundreds of billions of dollars. At stake are revenue, profit and market-share. Which configurations are designed and manufactured determine the cost, price, and time-to-market. Thus, a lot is at stake to come up with cost-effective configurations amongst many possibilities. This paper examines the following question: "Is it possible to predict the performance of systems accurately from a small subset of configurations or is it possible to predict performance of future systems using existing ones?" If these can be achieved, a company could potentially reduce its cost while manufacturing good configurations of systems. In this paper we demonstrate that these can be achieved.

When a new system is developed, the designers are faced with an immense challenge: how can one estimate the performance of a particular system configuration? When one considers all the components that need to be configured (CPU type, CPU frequency, motherboard, memory speed, memory size, busses, hard disk, etc.), it is clear that the set of possible configurations is huge. The design space exploration is an important task for all system (e.g., desktop/server/laptop) manufacturers. Currently, most designers rely on their intuitions to make decisions during the system design.

Design space exploration is not limited to system performance extraction, but is used in various domains including processor design, computer-aided design, and VLSI among others. Generally, designers in such fields revert to simulation to estimate the performance of their selection. Nevertheless, since the design space can be extremely large, it is not conceivable to simulate all the possible configurations; hence the designers mostly use heuristics (such as simulated annealing [1]) to find the set of configurations they evaluate. There are also several methods and heuristics to guide the design space exploration process in such domains [2, 3, 4]. *Our work deviates from such studies significantly, because we do not attempt to control the simulations to be performed, instead we try to predict the real system performance.* Specifically, a major bottleneck in system design is that each possible configuration has to be manufactured before its performance can be found. To the best of our knowledge, there is no publicly available work in this arena. In this work, we fill this gap by developing predictive models that will aid the developers. Specifically, we

a) develop predictive models using neural networks and linear regression to estimate the performance of a system by just using the information about its components,

b) show that the performance of a system can be accurately predicted by using a small fraction of the overall design space, and

c) show that the performance of a system can be accurately predicted by using information from past systems.

The rest of this paper is organized as follows. In Section 2, we give an overview of design space exploration and how our models can be used by manufacturers. In Section 3, we discuss our predictive models. Section 4 presents the results. Sections 5 and 6 present the related work and conclusions, respectively.

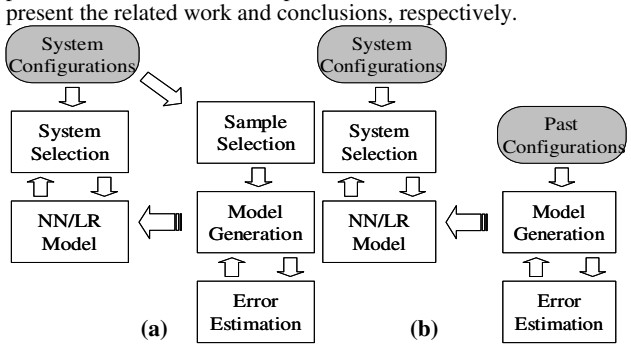

Figure 1. Overview of design space exploration using predictive modeling: (a) sampled design space exploration and (b) chronological predictive models.

2. OVERVIEW OF PREDICTIVE MODELING

In this work, we develop two types of predictive models that can be utilized by system designers. The overview of how these models are developed and used is depicted in Figure 1. The first

DAC 2008, June 8–13, 2008, Anaheim, California, USA

one (Figure 1(a)) is called sampled design space exploration. In this methodology, we first choose a random set of the configurations. Then, using this set as the training set, we build our models, which are later used to predict the performance of the rest of the design space. The second mode of operation is called chronological estimation (Figure 1(b)), where we generate models by using the results of the previous years' systems in the market to build the models, which are later used to predict the performance of the systems that are built in the following year. We must note that there may be other means of utilizing the predictive models during the design space exploration. However, we restrict ourselves to sampled design space exploration and chronological predictions, because they exhibit the most apparent use of predictive models in system design space exploration.

Another important aspect of our work is the use of real data. Specifically, we train and evaluate our models using previously announced SPEC results [5]. Hence, we can precisely report how accurately we can estimate the real system performance. We must highlight that SPEC rating is the most common means of comparing the performance of different systems and hence they are of tremendous importance to system manufacturers. SPEC ratings are commonly used for marketing and also used for setting the prices of the systems. Consequently, system manufacturers put great effort in optimizing their systems for these ratings.

Considering the tremendous cost advantages of using predictive models and such accurate predictions, the use of our models could provide a significant competitive advantage. *Another point that should be mentioned is that the parameters used in the models are not only processor parameters; other parameters related to memory and bus of the system are chosen by our models and contain high importance factors.*

3. PREDICTIVE MODELS

In this paper, we use predictive modeling techniques from machine learning to obtain estimates of performance of systems by using information about their components as the input. We use three different models. The linear regression model is described in the next section. Section 3.2 discusses the two neural network based models developed in this work.

3.1 Linear Regression (LR) Model

Regression analysis is a statistical technique for modeling relationship between variables [6]. We have n observations; $y=y_1,\ldots,y_n$ called the response variables, and $x_i=x_{i,1},\ldots,x_{i,p}$ for i=1..n that are predictor or regressor variables. The simplest linear regression is of the form $y=\beta_0+\beta_1 x+\varepsilon$. In this formula β represents the coefficients used in describing the response as a linear function of predictors plus a random error ε.

We used the linear regression model inside the SPSS Clementine [9] tool. In Clementine there are 4 available methods for creating the linear regression models: Enter (**LR-E**), Stepwise (**LR-S**), Forwards (**LR-F**), and Backwards (**LR-B**). In our experiments, for sampled design space we have seen that the Backwards (**LR-B**) method produced the best results. Therefore, we only present results for LR-B. The Backwards method builds the equation in steps. In this model, the initial model contains all of the input fields as predictors, and fields can only removed from the model. Input fields that contribute little to the model are removed form the model until no more fields can be removed without significantly degrading the model. Generally, we found that the linear regression models can be built quickly for our system. It took on the order of milliseconds to generate the models from our input data set.

3.2 Neural Network (NN) Models

Neural networks, or more accurately, Artificial Neural Networks (ANN), have been motivated by the recognition that the human brain processes information in a way that is fundamentally different from the typical digital computer [6]. A neural network, sometimes called multilayer perceptron, (feedforward ANN) is a multivariate statistical model used to relate p predictor variables $x_1,\ldots,x_p$ to q response variables $y_1,\ldots,y_q$. The model is very flexible containing many parameters and it is this feature that gives a neural network a nearly universal approximation property.

We used the SPSS Clementine tool to build the ANN models. The neural network node provides five different training methods: Quick (**NN-Q**), Dynamic (**NN-D**), Multiple (**NN-M**), Prune (**NN-P**), and Exhaustive Prune (**NN-E**). In our methods, we use two methods. The first one is the Single layer (**NN-S**) method (a modified version of NN-Q) and has a constant learning rate. This method uses only one hidden layer, which is smaller than the other methods. As a result, the models are faster to train. This model is similar to the one developed by Ipek et al. [2]. Note that Ipek et al. use this model for estimating the results of processor simulations. The other method that we use is Exhaustive prune (**NN-E**) method. In this model, network training parameters are chosen to ensure a very thorough search of the space of possible models to find the best one. This method is the slowest of all, but often yields the best results. During our analysis of the models, we have observed that the time it takes to build the neural network models vary significantly. While the NN-S model takes on the order of seconds to build, the NN-E models can take up to tens of minutes for the largest input data sets. However, relative to the time and cost of building a real system, these development times are still negligible.

3.3 Cross-validation

Clementine software does not provide the estimated predictive error for the model it creates. Therefore we have used 2 different sets of 5-fold cross-validation. In the first set of cross-validation, the training data is divided into 5 groups and 4 of them are used to create the predictive models using different methods. Then, the developed model is tested on the left out data group to calculate the estimated error. Afterwards, the group selection is rotated. The second set of cross-validation that we have used employs 3 sets to create the model, and 2 sets of data to calculate the estimated error. We similarly perform group rotation to extract the 5-fold cross-validation. We have observed that these 2 different cross-validation schemes produce similar results, and in general the second one produces estimations that are closer to the true error rates. However, since the former cross-validation uses more records during training, its true error rates can be lower. Therefore we always use the average of these 10 folds to calculate the true and estimated errors. Note that *true error rates* of the models are calculated by using the created models on the whole (100%) data.

3.4 Data Preparation and Input Parameters

Data preparation is an important part of the predictive modeling. In our experiments, Clementine software automatically scales the input data to the range 0-1 to prevent the effect of scales of different parameters. The linear regression methods expect the input parameters to be numerical. Therefore some of the inputs to Clementine need to be mapped to numeric values. For some other input parameters this kind of transformation is not possible, hence these are omitted by Clementine. However, neural network models can have any type of input (numeric, flag, categorical), and are automatically transformed and scaled for model generation usage.

In this work, we feed all the input available parameters to Clementine. Then the program automatically measures the importance of the parameters, and depending on the methodology adds or removes predictor variables to the model. In some of the chronological design space experiments Clementine omits some predictor variables because these input parameters does not have any variation (e.g. single L2 cache size configuration). Other than this kind of predictor elimination, we don't discard any input.

4. PREDICTION RESULTS

There are several possible methods of presenting SPEC numbers. The most commonly used one, *SPECint2000* rate is the geometric mean of the ratios (of execution to the base system) for

the 12 integer applications. In this work, we are mostly interested in being able to estimate this rate, because it is the most important metric used for determining the system performance, price, and marketability. Hence, we present the accuracy of our techniques for this rate (similar results were obtained for SPECfp2000 too).

SPEC contains results announced since 1999. Specifically, the SPEC results contain announcements from Intel, Alpha, SGI, AMD, IBM Power PC, Sun Ultra SPARC, etc. based systems. Among these, we have chosen to analyze the systems based on AMD Opteron (Opteron), AMD Opteron Dual (Opteron 2), Intel Pentium 4 (Pentium 4), and Intel Xeon (Xeon). The main reason for this selection is that these systems are the most commonly used systems. We analyze the systems based on the processor type because we have observed that when different processor types are used, the system configurations were significantly different from each other, preventing us from making a relative comparison.

An important property of the announcements is that even within a single processor family, the performance numbers showed significant variation: Opteron based systems has 210 records with a range of 2.21 times (i.e., the best system has 2.21 times better performance than the worst system) and variation of 0.15; Opteron 2 based systems have 197/2.47/0.15, Pentium 4 based systems have 241/7.70/0.37 and Xeon based systems have 216/1.34/0.09 records/range/variation values. The SPEC announcements contain information about the systems and as well as execution times of each application. Each announcement provides the configuration of 32 system parameters: Company, system name, processor model, number of processors and configuration, bus frequency, floating point unit, total cores and chips, SMT, Parallel, L1 instruction and data, L2 and L3 and L4 data cache configuration, memory size and frequency, hard drive size, speed and type, and extra components. Currently, there are 7032 announced results (3550 integer and 3482 floating-point).

4.1 Sampled Design Space Modeling

In this section we present results investigating the accuracy of our models. In these models, we randomly sampled 2% to 10% of the data to build our models, and then used the entire data set to predict the accuracy of our models. As described in Section 2, this approach can be used to reduce the design space size and hence accelerate the design space exploration. The percentage error is calculated by the formula: $100*|\hat{y}_i-y_i|/\ y_i$, where $\hat{y}_i$ is the predicted and y_i is the true (reported) number for the i^{th} record in the data used. By using only the training set during cross-validation (c.f. Section 3), we also extract estimated error rates for each model.

Figure 2 presents the results for the estimated and true error rates for the linear regression backwards LR-B (leftmost), neural network with exhaustive prune NN-E (middle), and neural network with single layer NN-S (rightmost) methods on Opteron based systems for varying the sampling rate. For the linear regression, we observe that the difference between the estimated and true error rates is generally small. In the neural network methods, we see that the exhaustive method has a slightly higher error rate than the single layer method. The backwards method of linear regression produces the best results, and has a prediction accuracy of 95.68% accuracy at 4% sampling.

The results for the Opteron 2 system are similar to the Opteron based systems, 8.5% and 4.2% error rate for LR model at 2% and 5% sampling, respectively. We can conclude that the estimation error is generally small for small sampling rates, and becomes very accurate with increasing data set size. An important property of all the presented results is that as the training set size is increased, the accuracy of the models generally increases. This is expected because with a larger training set, the variations in the design space can be modeled better.

For the Pentium 4 based systems, the error rates for the prediction accuracy are higher than the previously discussed systems unless a high sampling rate is used. One reason for this inaccuracy is that the range of Pentium 4 machines is very wide (the fastest machine is more than 7.7 times faster than the slowest machine). Hence, the systems are very different from each other and the selected training points are not always representative of the whole design space. However, when a larger data set is used for model creation, the accuracy increases rapidly. The estimated error rates are also close to the true error rates. When compared to other systems, we see that neural networks are optimistic when estimating the error rates. A similar observation can be made for the Xeon systems (presented in Figure 3). The prediction accuracy is generally better when there is little variation among the different systems. As a result, the training set used during the development of the model tends to contain similar records and particularly the neural network models are able to fit to these elements tightly. This results in very low estimated error rates.

The models developed in this section include 5 to 7 predictor variables. Within these, there are usually two or three factors that are significantly more important than others. The important factors and their order of importance change from a processor family to another. For example, *for the Pentium 4 systems, the most important parameters for neural networks (with their relative importance presented in parenthesis) are processor speed (0.503) L2 cache size (0.501), memory size (0.279), bus frequency (0.145), and L1 instruction cache size (0.115).* Note that the importance factor denotes the relative importance of the input factor (0 denoting that the field has no effect on the prediction and 1.0

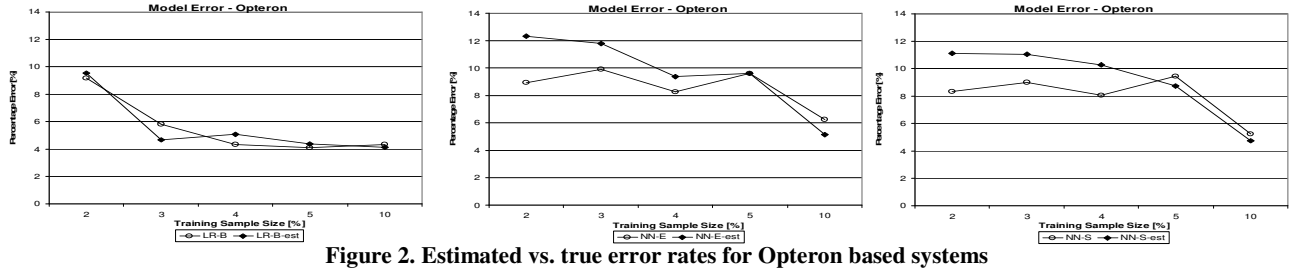

Figure 2. Estimated vs. true error rates for Opteron based systems

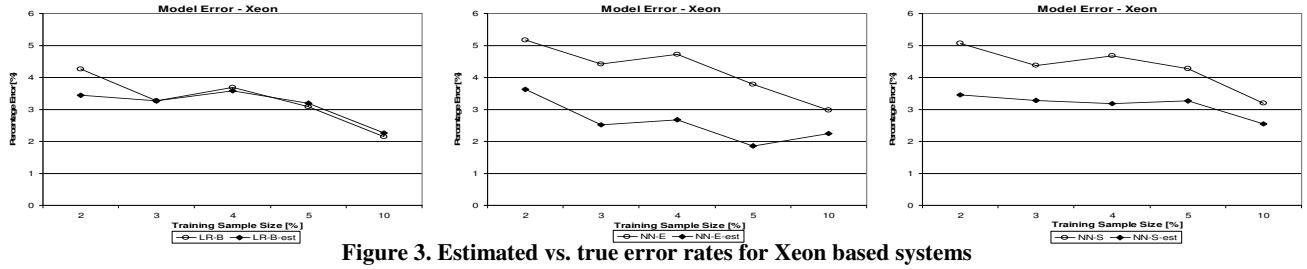

Figure 3. Estimated vs. true error rates for Xeon based systems

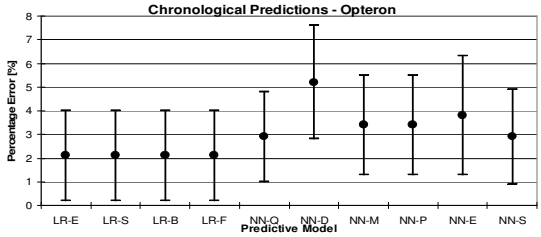

Figure 4. Chronological predictions for Opteron systems

completely determines the prediction). For the same predictions LR model has processor speed, memory size and L2 cache size with standardized beta coefficients of 0.510, 0.406 and 0.123, respectively. Standardized beta coefficients show the relative importance of the predictor variable.

4.2 Chronological Predictive Modeling

This section presents the results for chronological predictive models, which use historical performance announcements to predict the performance of future systems. In these models we have used announcements from 2005 as the training data and then the 2006 announcements have been used as the testing data.

In this section we have created models and calculated the prediction error for different LR and NN methods for Opteron, Opteron Dual, Pentium 4 and Xeon systems. The percentage error is calculated as described in the previous section. In general we see that LR models perform better than NN. One of the main reasons for this is the model built using 2005 data is very accurate for predicting 2005, however when we try to predict 2006, the over-fitting in NN causes larger errors in estimations. However, LR does not have this problem and is successful predicting 2006 results. Another point is that, for some systems we see that there are not significant changes to the configurations, e.g., L2 cache configuration. Hence the records are similar to each other allowing the LR model to perform well.

In Opteron systems (Figure 4), we see the best accuracy is achieved using with LR-B method of linear regression with an error rate of 2.1%. Going to a more complex system, Opteron-2, we have a higher minimum error rate of 3.1% with the stepwise method. Pentium 4 and Xeon systems have 1.5% and 2.4% error rates with LR-E method, respectively. We must note that various parameters are used during the predictions. For example, for the Opteron systems, the most important parameters for neural networks are processor speed (0.659), memory frequency (0.154), L2 being on or off chip (0.147), and L1 data cache size (0.139). For the same predictions, the linear regression model has processor speed and memory size with standardized beta coefficients of 0.915 and 0.119.

5. Related Work

There have been numerous works in the area of design space exploration. Eyerman et al. [10] uses different heuristics to model the shape of the design space of superscalar out-of-order processors. Ipek et al. [2] use artificial neural networks (with cross-validation to calculate their prediction accuracy) to predict the performance of memory, processor and CMP design spaces. Meanwhile, Lee et al. [4] use regression models to predict performance and power usage of the applications found in the SPECjbb and SPEC2000 benchmarks. In both of these references the data points are created using simulations. Kahn et al. [14] uses predictive modeling to tackle the problem of accurately predicting the behavior of unseen configurations in CMP environment. Ghosh et al. [11] have presented an analytical approach to the design space exploration of caches that avoids exhaustive simulation. The problem that they are trying to solve is small compared to our work. Dubach et al. [13] has used a combination of linear regressor models in conjunction with neural networks to a model that can predict the performance of programs on any microarchitectural configuration with 32 further simulations. In this work, we target system performance rather than processor performance. To the best of our knowledge there has not been any work done in this area. The closest work is by Ipek et al [12], where they use neural networks to predict the performance of SMG2000 applications run on multi-processor systems, where they change the application inputs and the number of processors used. Unlike our work, they do not explore system parameters.

6. Conclusion

In this work, we have developed two different statistical modeling techniques, linear regression and neural network, to predict the performance of computer systems. First we have used a small fraction of the whole design space to create prediction models that are then used to predict performance numbers on the whole design space. To show the accuracy of this performance prediction models, we have used the published SPECint2000 benchmark results. We create our models using 2% to 10% of the whole data. While the models are built, we also calculate an estimate for their accuracy using cross-validation. With the use of the sampling of the input data set and the predictive models, our results reveal 95.3% accuracy rate on average using only 5% of the possible machines in the design space. We generally observe that the estimated error rates are close to these true error rates. Secondly, we have shown that the performance of future systems can be estimated with less than 2.2% error rate on average by using the data from previous systems. These results indicate that the designers can estimate the performance of new systems using the limited data available for the already built systems and use them to estimate similar as well as future systems. The system manufactures can use current system configuration and the important factors provided by the models to start the search towards a system that will provide the highest performance.

7. Acknowledgements

This work was supported in part by NSF grants CNS-0406341, CNS-0551639, IIS-0536994, CCR-0325207, CNS-0720691, IIS-0613568, CCF-0541337, and DOE's SCIDAC and FASTOS programs.

8. References

[1] Moya F., Moya J. M. and Lopez J. C., Evaluation of Design Space Exploration Strategies. In Proc. of the EUROMICRO Conference, Sep 1999, York, England

[2] Ipek E., McKee S. A., deSupinski B. R., Schultz M. and Caruana R. Efficiently Exploring Architectural Design Spaces via Predictive Modeling. In Proc. of the ASPLOS, Oct. 2006, San Jose, CA.

[3] Cavazos J., Dubach C., Fursin G. and Temam O. Automatic Performance Model Construction for the Fast Software Exploration of New Hardware Designs. In Proc. of the Int. Conference on Compilers, Architecture and synthesis for embedded systems. 2006, Seoul Korea.

[4] Lee B. C. and Brooks D. M. Accurate and Efficient Regression Modeling for Microarchitectural Performance and Power Prediction. In Proc. of the ASPLOS. Oct 2006, San Jose, CA.

[5] The Standard Performance Evaluation Corporation, http://spec.org

[6] Tan P., Steinbach M. and Vipin K. Introduction to Data mining. Addison-Wesley, 2005.

[9] SPSS Clementine version 11, http://www.spss.com/clementine

[10] Eyerman S., Eeckhout L. and Bosschere K. D. The Shape of the Processor Design Space and its Implications for Early Stage Explorations. In Proc. of Int. Conf. on ACMOS. Mar 2005, Prague, Czech Republic.

[11] Ghosh A. and Givargis T. Analytical Design Space Exploration of Caches for Embedded Systems. In the Proc. of the DATE Conference. Mar 2003, Munich Germany.

[12] Ipek E., de Supinski B. R., Schulz M. and McKee S. A. An Approach to Performance Prediction for Parallel Applications. In Proc. of the Euro-Par. May 2005, Monte de Caparica, Portugal.

[13] Dubach C., Jones T. M. and O'Boyle M. F. P. Microarchitectural Design Space Exploration Using An Architecture-Centric Approach. In Proc. of the 40th MICRO, Dec 2007, Chicago, IL.

[14] Khan S., Xekalakis P., Cavazos J. and Cintra M. Using Predictive Modeling for Cross-Program Design Space Exploration in Multicore Systems. In Proc. of Int. Conf. on PACT, Sep 2007, Brasov, Romania.