# Analyzing Commercial Processor Performance Numbers for Predicting Performance of Applications of Interest *

Kenneth Hoste
ELIS Department
Ghent University
Belgium
kehoste@elis.UGent.be

Lieven Eeckhout
ELIS Department
Ghent University
Belgium
leeckhou@elis.UGent.be

Hendrik Blockeel
CS Department
K.U. Leuven
Belgium
hendrik.blockeel@cs.kuleuven.be

## ABSTRACT

Current practice in benchmarking commercial computer systems is to run a number of industry-standard benchmarks and to report performance numbers. The huge amount of machines and the large number of benchmarks for which performance numbers are published make it hard to observe clear performance trends though. In addition, these performance numbers for specific benchmarks do not provide insight into how applications of interest that are not part of the benchmark suite would perform on those machines.

In this work we build a methodology for analyzing published commercial machine performance data sets. We apply statistical data analysis techniques, more in particular principal components analysis and cluster analysis, to reduce the amount of information to a manageable amount to facilitate its understanding. Visualizing SPEC CPU2000 performance numbers for 26 benchmarks and 1000+ machines in just a few graphs gives insight into how commercial machines compare against each other.

In addition, we provide a way of relating inherent program behavior to these performance numbers so that insights can be gained into how the observed performance trends relate to the behavioral characteristics of computer programs. This results in a methodology for the ubiquitous benchmarking problem of predicting performance of an application of interest based on its similarities with the benchmarks in a published industry-standard benchmark suite.

## Categories and Subject Descriptors

C.4 [**Performance of systems**]: Modeling techniques

## General Terms

Design, Performance, Experimentation

## Keywords

Performance analysis, benchmark similarity, performance prediction

## 1. INTRODUCTION

In order to characterize commercial machines, current practice is to run industry-standard benchmarks on those machines and to publicly report performance numbers. This practice is adopted by various benchmarking consortiums and corporations such as EEMBC for embedded systems, TPC for database systems and SPEC for high-performance computer systems. The information obtained from these benchmarking experiments provide valuable information for comparing existing commercial machines across a broad range of applications. This enables computer buyers to make appropriate decisions when purchasing a computer system.

The information provided by these performance evaluation studies consists of performance numbers for individual benchmarks on all of the reported machines. There are two consequences to this. First, given the large amount of data provided, it is difficult to get a good understanding on how performance on the various machines is affected by inherent program behavior. Second, the data presented by these sources show performance numbers for specific benchmarks. This may not provide a clue though about the performance of a given application of interest.

The goal of our work is to address these two concerns. First, we propose the use of statistical data analysis techniques such as principal components analysis (PCA) to identify performance trends across the various machines and benchmarks. The important benefit of using PCA is that a large data set is reduced to a tractable amount of information, just a few graphs visualizing how machines perform on the various benchmarks. This enables the easy understanding of how commercial machines compare against each other. As a second step, we use this information to build models for predicting performance of an application of interest. This is done by clustering machines that show similar behavior across benchmarks, and by building performance models per group of similarly behaving machines. These performance models use a genetic algorithm to learn how to relate inherent program behavior to performance. The performance of an application of interest on a machine of interest is then estimated by (i) profiling the inherent program behavior of the application of interest, (ii) using the performance model to find to which benchmarks in the benchmark suite the application of interest most closely resembles, and (iii) weighting the performance numbers for the most similar benchmarks to come up with a performance estimate for the application of interest on the machines of interest.

## 2. PERFORMANCE ANALYSIS

Our data set contains performance numbers for the industry-standard benchmarks on a number of commercial machines. The commercial machine performance numbers that we use in this paper are real hardware speedup numbers reported on the SPEC CPU website. In total, there are 26 performance numbers (one per benchmark) for 1000+ machines in our data set.

For gaining insight in this large data set, we use principal com-

*Lieven Eeckhout and Hendrik Blockeel are Postdoctoral Fellows with the Fund for Scientific Research – Flanders (Belgium) (FWO Vlaanderen). This research is also supported in part by Ghent University, IWT, and the HiPEAC Network of Excellence.


*SIGMETRICS'07,* June 12–16, 2007, San Diego, California, USA.
ACM 978-1-59593-639-4/07/0006.

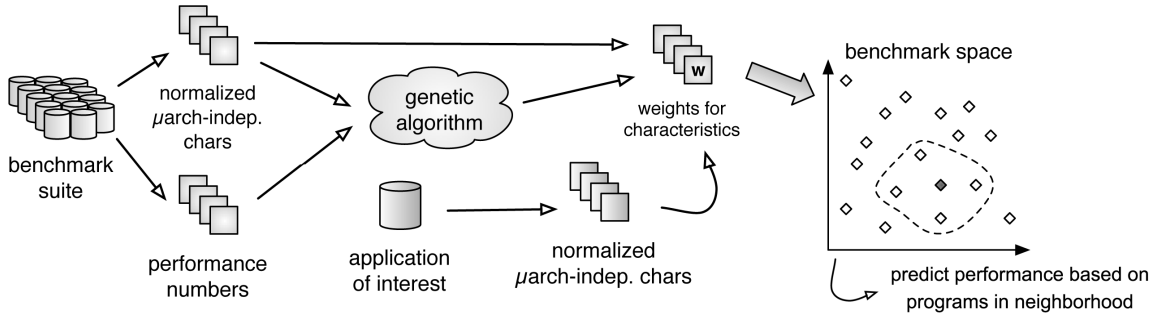


**Figure 1: Framework for predicting performance of an application of interest based on its microarchitecture-independent program behavior and its similarity with benchmarks from a reference benchmark suite.**

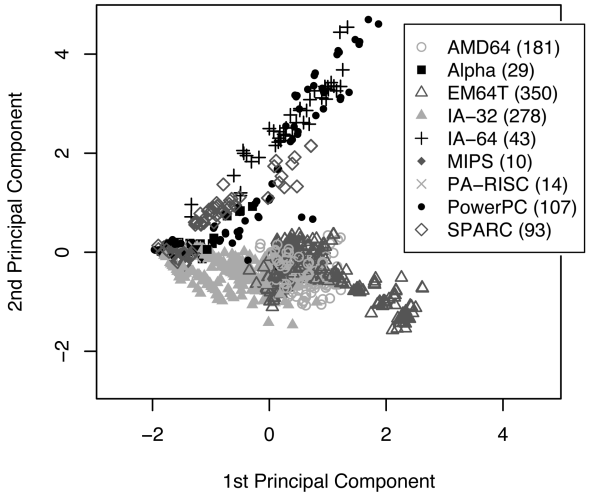


**Figure 2: Visualizing the PCA space obtained from the SPEC CPU2000 performance numbers. The various symbols reflect different ISAs.**

ponents analysis (PCA). PCA computes so called *principal components*, which are linear combinations of the original variables, such that all principal components are uncorrelated. PCA transforms the $p$ variables $X_1, X_2, \ldots, X_p$ into $p$ principal components $Z_1, Z_2, \ldots, Z_p$ with $Z_i = \sum_{j=1}^{p} a_{ij} X_j$. This transformation has the properties (i) $Var[Z_1] \geq Var[Z_2] \geq \ldots \geq Var[Z_p]$ — this means $Z_1$ contains the most information and $Z_p$ the least; and (ii) $Cov[Z_i, Z_j] = 0, \forall i \neq j$ — this means there is no information overlap between the principal components.

Figure 2 shows the 1000+ machines plotted in a two-dimensional space built up by the two most significant principal components. The first and second principal components explain 80.1% and 9.1% of the total variance, respectively. Interpreting the meaning of the principal components shows that a high value along the first principal component indicates high average performance across all benchmarks; and a high value along the second principal component indicates a high average performance for the floating-point benchmarks. This means that machines, such as the AMD64 and EM64T machines, having a high score along the first principal component and a low score along the second principal component achieve high average performance, more so for integer than for floating-point benchmarks. The PowerPC and IA-64 machines on the other hand have a high score along the second principal component indicating that they achieve relatively higher performance for the floating-point than for the integer benchmarks.

# 3. PERFORMANCE PREDICTION

We now take this performance analysis methodology one step further in order to predict performance for an application of interest based on its inherent program similarity with the benchmarks in the industry-standard benchmark suite. The performance prediction framework that we propose is to first apply cluster analysis to identify groups of similarly behaving machines based on the reported processor performance numbers — Figure 2 illustrates that groups of machines exist. As a second step, for each cluster of machines, we then build a performance model using a genetic algorithm that learns how to relate microarchitecture-independent characteristics to performance.

Figure 1 illustrates how we build the performance model — we refer to [1] for a more detailed description since we will only briefly summarize the method in what follows. Our approach assumes a collection of programs which we call the *benchmark suite*. For each of these benchmarks, we have a collection of microarchitecture-independent characteristics as well as performance numbers on a (number of) platform(s). These microarchitecture-independent characteristics along with the performance numbers are then used to compute data transformation weights to weight the original microarchitecture-independent characteristics. We use a genetic algorithm to compute the data transformation weights. The key observation that motivates us using a genetic algorithm is that when using the Euclidean distance in the unscaled microarchitecture-independent benchmark space, all microarchitecture-independent characteristics are implicitly assumed to have the same impact on overall performance. By multiplying program characteristics by the weights obtained through the genetic algorithm, a higher or lower impact can be given to particular characteristics. In other words, the goal of the genetic algorithm is to learn how to weight the various microarchitecture-independent characteristics so that the Euclidean distance in the rescaled benchmark space is a good measure for the performance (in)differences between benchmarks across the machines.

For an application of interest for which we want to predict performance, we then compute the set of microarchitecture-independent characteristics and subsequently weight these microarchitecture-independent characteristics. This locates the application of interest in the benchmark space. Performance is then predicted by appropriately weighting the performance numbers of the benchmarks, called *proxies*, in the neighborhood of the application of interest.

# 4. REFERENCES

[1] K. Hoste, A. Phansalkar, L. Eeckhout, A. Georges, L. K. John, and K. De Bosschere. Performance prediction based on inherent program similarity. In *Proceedings of the 2006 International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pages 114–122, Sept. 2006.