Identifying the best performing hardware platform based on inherent program similarity

Kenneth Hoste, Lieven Eeckhout,
Andy Georges and Koen De Bosschere

* *ELIS, Ghent University, Belgium*

ABSTRACT

An important problem in benchmarking is to identify the platform that yields the best performance for an application of interest. This paper proposes a methodology for doing this, using both microarchitecture-independent characteristics and genetic algorithms. We first compare the application of interest with the programs from a profiled benchmark suite. We subsequently make a performance prediction based on the inherent program similarity of the application of interest with the benchmarks in the benchmark suite. The value of our methodology is shown by comparing it with the current approach for choosing the best platform for a given application, i.e. choosing the platform that yields the best average performance over all benchmarks.

1 Introduction

Identifying the best platform for an application of interest is an important aspect of benchmarking. Most of the time, the user has to rely on a standarized benchmark suite for estimating the performance of his application.

Choosing the best platform for an application of interest should be fast, but also reasonably accurate. Our methodology addresses this issue. We are able to characterize a program independent of the microarchitecture. We use known performance numbers of a standardized benchmark suite, spanning over a wide number of commercial machines. Our results show that the methodology succeeds in identifying the best machines for an application of interest.

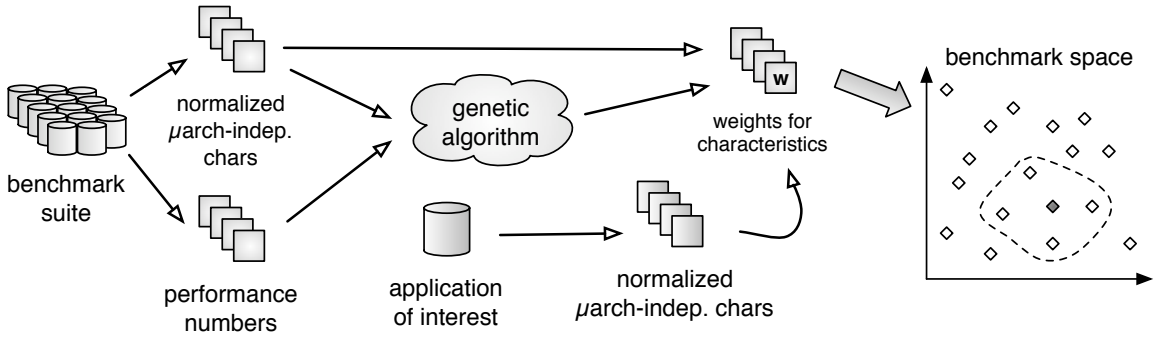

Figure 1: The framework proposed in this paper for predicting performance based on microarchitecture-independent program characteristics.

2 Performance Prediction Framework

Figure 1 illustrates the framework that we propose in this paper for identifying the best hardware platform for a given application. For each benchmark in the standardized benchmark suite, we have a collection of microarchitecture-independent program characteristics, as well as performance numbers on a (number of) platform(s).

For an application of interest, for which we want to identify the optimal hardware platform, we compute the same microarchitecture-independent characteristics as used with the benchmark suite. Together with the weights obtained with the genetic algorithm (see Section 2.2), this positions the application of interest in the benchmark space. Performance for each of the hardware platforms is then estimated by appropriately weighting the performance numbers of the benchmarks in the neighborhood of the application of interest.

We now (briefly) discuss a number of aspects of this framework. A more detailed discussion, along with an additional comparison to two other techniques (normalisation and Principal Components Analysis) can be found in [1].

2.1 Microarchitecture-independent characteristics

Ideally, the program characteristics are be platform-independent characteristics, i.e. they are compiler-, ISA- and microarchitecture-independent in order to capture the true inherent program behavior. Since this is difficult to do, we take a pragmatic approach and use microarchitecture-independent characteristics. These can be divided in 6 categories: instruction types, ILP, branch prediction (using a PPM predictor), register traffic, data stream strides and workingset size.

As will be shown in the evaluation section of this paper, these characteristics, inspite of being ISA-dependent and compiler-dependent, are accurate enough for tracking performance across different platforms with different ISAs and compilers.

2.2 Genetic Algorithm

Since we use the Euclidean distance as a distance measure in the benchmark space, we implicitly assume that the Euclidean distance in the benchmark space is proportional to the performance differences across a variety of platforms. Because some program characteris-

tics have a much larger impact on performance than others, an appropriate distance measure should give a higher weight to important program characteristics.

We propose a genetic algorithm (GA) for computing these weights. A genetic algorithm is an evolutionary optimization method that starts with a population of solutions. For each solution in the population, a fitness score is computed and the solutions with the highest fitness score are selected for constructing the next generation. This algorithm is repeated, *i.e.* new generations are constructed, until no more improvement is observed for the fitness score.

The fitness score that we use here is the prediction accuracy of our framework to predict performance speedups across a wide range of machines. As such, the genetic algorithm learns how the distance measure in the benchmark space correlates with performance across a variety of platforms.

2.3 Performance Prediction

Once the weights for each program characteristics are computed, we scale each of the dimensions of the benchmark space, i.e. the normalized program characteristics. This way, we obtain a benchmark space in which the Euclidean distance is a good measure for the performance distance between two benchmarks.

Predicting performance for an application of interest then requires that microarchitecture-independent characteristics are measured and that these characteristics are normalized and weighted using the weights obtained with the genetic algorithm. This positions the application of interest in the benchmark space.

The performance prediction is calculated by taking a weighted average over the performance numbers of the benchmarks in the neighborhood of the application of interest, which we call *proxies*. In fact, the weights w_i are inversely proportional to the distance d_i between the application of interest and proxy i. The weight w_i is computed as

$$w_i = \frac{1}{d_i \cdot \sum_{i=1}^{n} \frac{1}{d_i}}, \tag{1}$$

with n being the number of proxies of the application of interest we take into account.

In this paper, we focus on predicting performance speedups, rather than predicting raw performance. Predicting relative performance differences is often more important in practice. The performance speedup of the application of interest is computed as the weighted harmonic average over the speedups of the proxies:

$$S = \frac{1}{\sum_{i=1}^{n} \frac{w_i}{S_i}}. \tag{2}$$

3 Evaluation

The evaluation of the framework is performed using the full SPEC CPU2000 benchmark suite and speedup rates for 36 machines, taken from the SPEC website. This section only briefly covers the results of our research. We limit ourselves to comparing our approach with the current practice, i.e. assuming the best machine for any application is the one with the best average performance over all benchmarks.

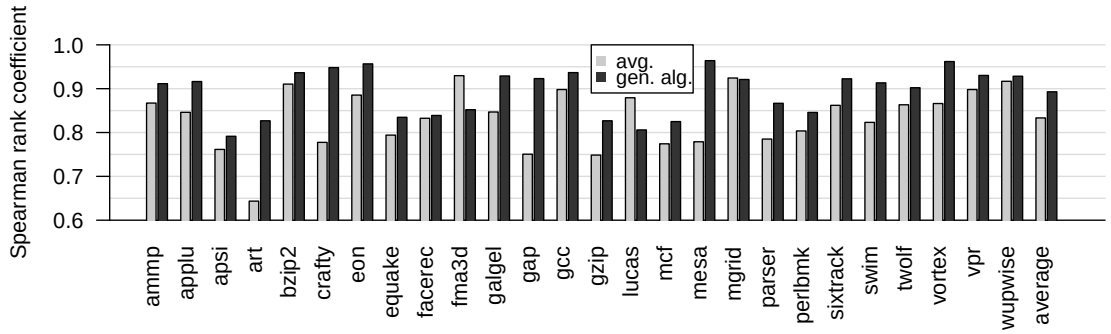

Figure 2: The Spearman correlation coefficients for estimating the ranks for the average benchmark suite speedup results and the proposed framework.

To quantify the accuracy of predicting the ranks of different systems we compute rank correlation coefficients. For a given application of interest we rank all the machines based on the predicted speedups. A similar rank can be computed based on the measured speedups. We then compute the Spearman rank correlation coefficient which is a measure for how well the estimated rank corresponds to the measured rank. The Spearman rank correlation coefficient for a rank based on average speedup numbers across all benchmarks is 0.83 on average, and shows a minimum of 0.64. The genetic algorithm achieves a higher correlation coefficient on average, namely 0.89, and improves the worst case to 0.79.

The correlation coefficients for all benchmarks are shown in Figure 2. Our approach, using the genetic algorithm, outperforms the current approach for 23 out of the 26 benchmarks. For some benchmarks, for example *art* and *mesa*, a relatively big improvement is seen in correlation coefficient, which means the hardware platform chosen by our framework will be far better than the one obtained using current practice.

4 Summary

This paper proposed an approach for addressing the ubiquitous problem in benchmarking which is the identification of the platform that yields the best performance for the given application of interest. The key idea is to compare inherent program characteristics of the application of interest against the same characteristics for all programs in the standardized benchmark suite. Weighting the program characteristics is done through genetic algorithms. Based on the inherent similarity of the application of interest with the benchmarks in the benchmark suite, a number of proxies are identified and a performance prediction can be made using the performance numbers of the proxies.

For more details on the framework, we refer to [1].

References

[1] K. Hoste, A. Phansalkar, L. Eeckhout, A. Georges, L.K. John, and K. De Bosschere. Performance Prediction based on Inherent Program Similarity. Accepted for the 15th International Conference on Parallel Architectures and Compilation Techniques (PACT-2006), Sept. 2006