The pitfall in comparing benchmarks using hardware performance counters

Kenneth Hoste*, Lieven Eeckhout*

*ELIS, Ghent University, Sint-Pietersnieuwstraat 41, 9000 Gent, Belgium

ABSTRACT

Understanding the behavior of emerging workloads is important for designing next generation microprocessors. For addressing this issue, computer architects and performance analysts build benchmark suites of new application domains and compare the behavioral characteristics of these benchmark suites against well-known benchmark suites. Current practice typically compares workloads based on microarchitecture-dependent characteristics measured using hardware performance counters while running these workloads. There is one pitfall though with comparing benchmarks using microarchitecture-dependent characteristics, namely that completely different inherent program behavior may yield similar microarchitecture-dependent behavior. We illustrate this by comparing the distance between benchmarks using both hardware performance counter metrics and microarchitecture-independent characteristics.

KEYWORDS: hardware performance counters, microarchitecture-independent characteristics, pitfall

1 Introduction

Being able to compare benchmarks is a very important aspect of computer architecture and performance analysis. In order to assess how different emerging workloads are from already existing benchmark suites, researchers typically characterize the emerging workloads by comparing the comparing the characteristics of these workloads versus the characteristics of well-known benchmark suites. Current practice in comparing benchmark suites is to characterize the suites in terms of a number of microarchitecture-dependent metrics. Most workload characterization papers run the benchmark suite that represents the emerging workload on a given microprocessor while measuring program characteristics using hardware performance counters; others use simulation for deriving similar results. The program characteristics typically being measured are instruction mix along with a number of microarchitecture-dependent characteristics such as IPC, cache miss rates, branch misprediction rates, TLB miss rates, *etc.* These studies then conclude by saying that two workloads are dissimilar if the hardware performance counter characteristics are dissimilar to each other; and reverse, two workloads are similar if the hardware performance counter characteristics are similar to each other.

[1] Email: {kehoste,leeckhou}@elis.UGent.be

There is one major pitfall though with this approach. Program characteristics measured using hardware performance counters may hide the underlying inherent program behavior, *i.e.*, although the hardware performance counter metrics may be similar to each other, the inherent program behavior can be different. And this can be misleading for driving microprocessor design, especially when today's processors are used for characterizing emerging workloads that will eventually run on future processors. As a solution to this problem, we propose to characterize benchmarks using microarchitecture-independent characteristics in order to capture the true inherent program behavior.

The main contribution of this paper is showing that measuring benchmark similarity based on program characteristics obtained from hardware performance counters can be misleading. In fact, we present a case study showing benchmarks that exhibit similar behavior in terms of the hardware performance counter metrics, however, the underlying inherent program behavior is quite different.

2 Experimental setup

There are 122 benchmarks in total from 6 benchmark suites: BioInfomark covering a bioinformatic workload, BioMetricsWorkload covering a biometric workload, CommBench covering a telecommunication workload, MediaBench covering a multimedia workload, MiBench covering embedded workloads and SPEC CPU2000 covering general-purpose workloads. For all benchmarks we use the largest available input.

3 Program characterization methodologies

We use two data sets in this paper, namely a microarchitecture-independent data set and a data set obtained from hardware performance counter profiling. These data sets are detailed in the following two subsections.

Microarchitecture-independent characterization The range of microarchitecture-independent characteristics is fairly broad in order to cover all major program behaviors such as instruction mix, inherent ILP, working set sizes, memory strides, branch predictability and register traffic. We use ATOM for collecting these characteristics. More details are available in [1].

Hardware performance counter characterization The hardware performance counter metrics that we use in this paper are typical for what is being observed in many workload characterization papers. We collect hardware performance counter values for IPC, branch misprediction rate, L1 D-cache miss rate, L1 I-cache miss rate, L2 cache miss rate and D-TLB miss rate. The machine on which we collect these hardware performance counter values is the Alpha 21164A processor. The Alpha 21164A (EV56) processor in an in-order dual-pipeline superscalar processor. We use the `dcpi`-tool for collecting these hardware performance counter values. Next to the above hardware counter values we also collect the IPC on the Alpha 21264A (EV67) which is an out-of-order four-wide superscalar processor.

4 Pitfall in using hardware performance counters

As mentioned in the introduction, comparing benchmarks based on microarchitecture-dependent characteristcs can be misleading. The fundamental reason is that different inherent (microarchitecture-

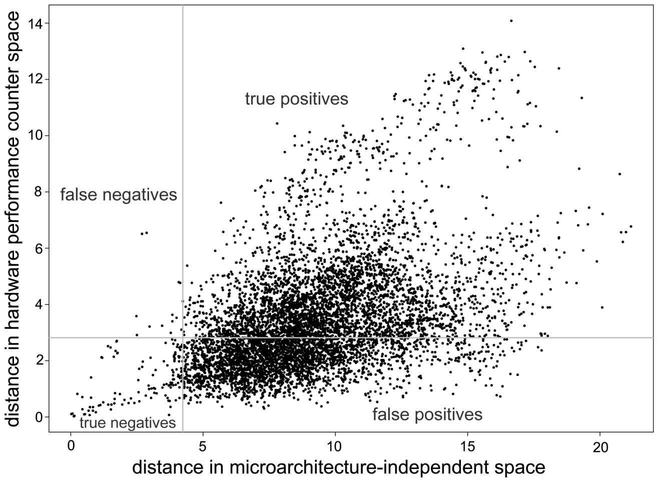

Figure 1: Distance in the hardware performance counter space versus the distance in the microarchitecture-independent space.

independent) program behavior can yield similar microarchitecture behavior. The pitfall in microarchitecture-dependent characterization is that the conclusions taken based on this microarchitecture-dependent characterization may not be generalized to other microarchitectures.

In order to quantify this, we classify all benchmark tuples into four categories according to the distances in the hardware performance counter space versus the distance in the microarchitecture-independent workload space, see Figure 1. Each dot represents a benchmark tuple. The threshold distance in the hardware performance counter space and microarchitecture-independent space are 20% of the maximum distance observed in both spaces. A distance in the either space is called to be large in case the distance is larger than 20% of the maximum distance observed.

A first aspect we quantify is the number of benchmark tuples categorized as a false negative, which is very smal (less than 1%). This shows that the microarchitecture-independent workload analysis is capable of identifying program similarity. Second, we observe that there is a large fraction of benchmark tuples categorized as a false positive (41% of all benchmark tuples). This corresponds to similarly behaving benchmarks in the hardware performance counter space while exhibiting dissimilar behavior in the microarchitecture-independent space. This illustrates that program characterization based on hardware performance counter values can be misleading in a fair amount of cases.

We now further illustrate this pitfall in hardware performance counter program characterization by looking into an example, namely SPEC CPU's bzip2 versus BioInfoMark's blast. Figure 2 shows normalized metrics in the hardware performance counter space and the microarchitecture-independent space, respectively; normalization is done per individual characteristic by dividing the measured value by the maximum value observed across the various benchmark tuples. Note that we use the instruction mix here as part of the hardware performance counter characterization as is done in many workload characterization papers. We observe that the hardware performance counter metrics are similar between bzip2 and blast. The other microarchitecture-independent program characteristics howoever are fairly different. The most strikingly different characteristics are the working set sizes for both the instruction stream and the data stream. Also the branch predictability based on global history seems to be fairly dissimilar; another example of dissimilar inherent program behavior is the amount of global store strides.

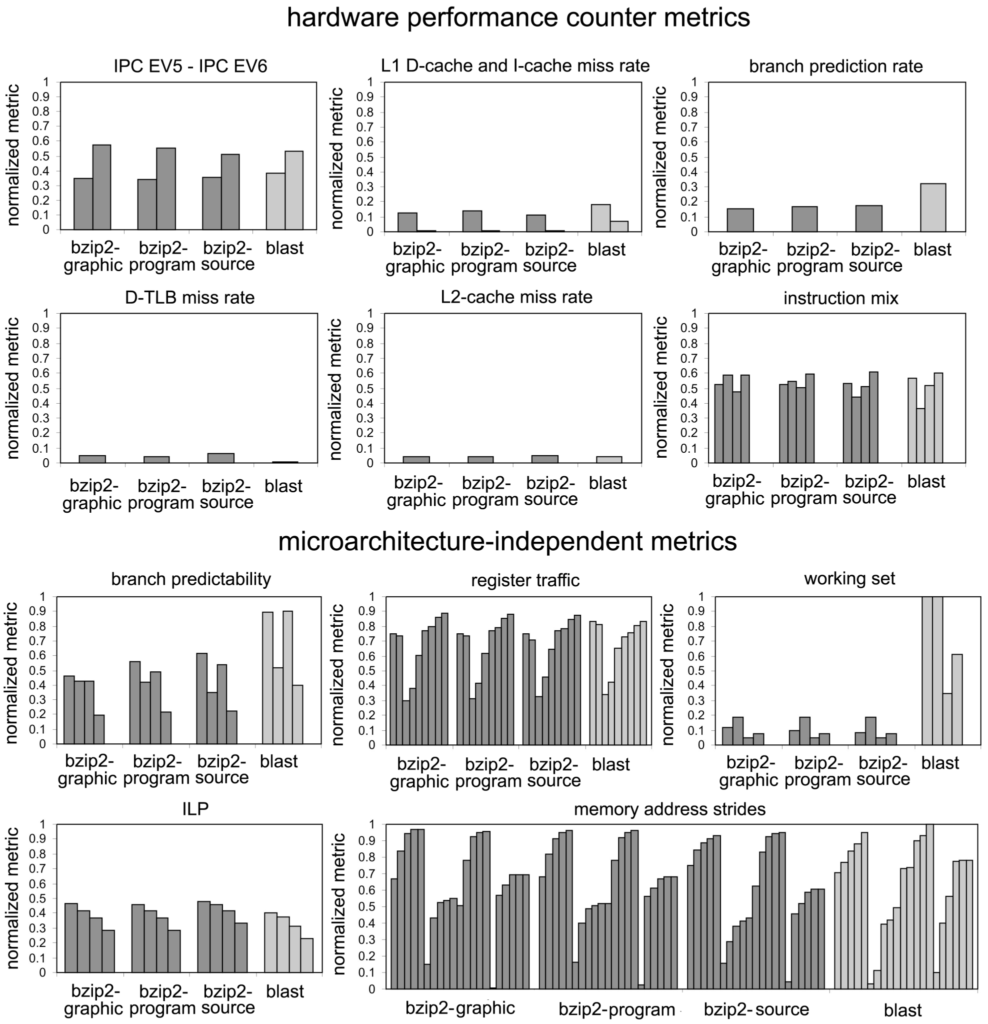

Figure 2: Evaluating the program characteristics for bzip2 versus blast.

5 Conclusion

We showed that a workload characterization based on microarchitecture-dependent characteristics can be misleading because it does not reveal the true inherent behavior of an application. In fact, two benchmarks may look similar in terms of their microarchitecture-dependent behavior, however, the inherent program behavior may be dissimilar. This motivates us for proposing a workload characterization methodology based on microarchitecture-independent program behavior. More details on this analysis are available in [1]. In this paper, we also evaluate approaches for limiting the number of the microarchitecture-independent characteristics that need to be measured, and make a comparison between the workloads available in the six benchmark suites we used.

References

[1] Kenneth Hoste and Lieven Eeckhout. Comparing benchmarks using key microarchitecture-independent characteristics. Accepted for *IEEE International Symposium on Workload Characterization (IISWC-2006)*, October 2006, San Jose, California, US.