Towards as Strategy for Performance Prediction on Heterogeneous Architectures

Silvio Stanzani, Raphael Cóbe, Jefferson Fialho, Rogério Iope, Marco Gomes, Artur Baruchi and Júlio Amaral

[1] Núcleo de Computação Científica (NCC)
Universidade Estadual Paulista
[silvio, rmcobe,jfialho,rogerio,mgomes,abaruchi, julioamaral]@ncc.unesp.br

Abstract. Performance prediction on high performance computing has always been a great challenge, even for homogeneous architectures. However, today's trend is the design of cluster running in a heterogeneous architecture, which increases the complexity of new strategies to predict the behavior and time spent by an application to run. In this paper we present a strategy that compares the performance of an application on different architectures and returns which one the application can obtain the best performance. The proposed strategy was able to classify correctly three of four applications tested without overhead implications. Our next step is to extend the metrics used to perform the classification in order to increase the strategy accuracy.

Keywords: Performance Prediction, Heterogeneous Systems, Parallel Processing.

1 Introduction

One trend that has become popular in designing computer clusters is the use of heterogeneous architectures, for instance composed by multicore and manycore architectures [1], which leads to computer infrastructures that is more complex in scale, heterogeneous cores and memory system. Such a trend can be observed in the configuration of several supercomputers, such as, Stampede2[1], JURECA[2] and Santos Dumont[3].

Performance prediction are essential to guide performance optimization of an application or support runtime decisions [2,3]. Typically, the performance prediction of an application is a manual process, that can be based on simulation, profiling data or historical data, which is carried out by the user in order to optimized an application on a specific architecture.

The challenges related to performance prediction on heterogeneous infrastructures is the performance comparison of an application across different resources, and the

[1] https://www.tacc.utexas.edu/systems/stampede2
[2] http://www.fz-juelich.de/ias/jsc/EN/Home/home_node.html
[3] http://sdumont.lncc.br/

utilization of such prediction to support runtime decision. In order to overcome these challenges, the contribution of this work is a strategy based on profiling data to do performance prediction of application of different architecture, by classifying the architectures according to the performance gain that can be obtained with an application.

The remainder of this papers is organized as following: Section 2 presents the problem definition and concepts, Section 3 presents the related work, Section 4 presents the strategy proposed, Section 5 presents the evaluation of strategy and Section 6 presents the conclusions and future work.

2 Concepts and Problem Definition

In this section, we are going to present the problem definition and the main concepts related with the strategy proposed.

2.1 Concepts

One technique to characterize the performance of an application is using the concept of machine balance which defines the performance of an application using the following metric: the ratio of the amount of data transfer to the number of floating-point operations for a particular processor [4].

The information to characterize the performance of an application according to this model can be obtained from code profilers, which perform several measurements during the execution of a small part of application [8] and can be based on two approaches: measurement of source code or assembly code behavior, also called code profiling, and on the data provided by PMU (Performance Monitoring Units) in order to identify the hardware usage efficiency during the execution of an application.

A tool to perform hardware profiling is called Intel VTune [4] and a tool that perform several code profiling is the Intel Advisor[5] which provides several metrics and performance insights of an application. We describe the main analysis provided by Intel Advisor below:

- Survey Target: which measures how much time was spent for each application loop and also some static performance estimate gain obtained with optimizations performed by the compiler.
- Trip Count: which measures the performance achieved by an application and also for each loop in terms of FLOPS (Float Point Operations per Second) and data transfer from main memory to processor memory.

[4] https://software.intel.com/en-us/intel-vtune-amplifier-xe
[5] https://software.intel.com/en-us/advisor

2.2 Problem Definition

The problem we are tackling is the performance prediction of an application among different computational architectures, in order to support scheduler systems, classify the architectures according to the performance gain that architecture can provide to each application. In this sense, we impose the following restrictions to the performance prediction:

- The application is provided without the source code.
- The prediction has to be performed in the shortest possible time.
- The user can provide compiled versions, optimized to each heterogeneous resource, of the same application.

3 Related work

In [5], the authors present a method for profiling C code with OpenACC directives, from that, they create a computation model that will run on an HPC architecture simulator in order to measure some performance metrics. In our case, we use data collected from the real execution. This is very important since we are dealing with heterogeneous architectures that have different microarchitecture implementations.

In [7], the authors propose an approach that uses small input cases for performing the full execution of the program, in order to collect data of all the code regions. Their concern is the execution time of the program and to predict it, they use a regression model, based on small kernels – the same approach is adopted by [6] - to estimate the full program execution. In our case we are not only concerned with execution time, also, using such statistical models require training, to learn the correct parameters, that could be a hard task depending on the model used to perform the regression.

In [3], the authors propose a tool for making performance prediction named COMPASS. They also generate an execution model – a translation of the source code to the Aspen Language, that should be analyzed by a performance evaluation tool, named Aspen Performance Prediction Tools that should also be fed with an architecture descriptor file, containing the information regarding the architecture on which the performance prediction should be done. Also, the code can be instrumented to help the tool to better generate the program execution model. In our case, no manual instrumentation is needed. The metrics are obtained using standard tools and also no source code translation is required.

4 A Strategy for Performance Prediction

In this session, we will present a strategy to do performance prediction of an application on different computational architectures. The strategy will classify each architecture according to the performance gain that can be achieved.

The strategy receives the following inputs:

- Application: composed of one compiled version of the same source code optimized to each computational architecture.
- Set of resources to perform performance perdiction: The heterogeneous architectures we are considering in this work are the different Intel generations, such as, Intel Xeon (Sandy Bridge, Haswell and Skylake) and Intel Xeon Phi (Knights Landing).

We will describe the profiling collection performed by strategy on Section 4.1 and the data analysis and output on Section 4.2.

4.1 Profiling Collection

The strategy characterizes the performance of an application on each architecture according to the machine balance model [4], measuring two application characteristics: the amount of operations that the application is capable of execute on each architecture, and the amount of work that is wasted on overhead.

These measurements are obtained using the following metrics, that is collected by a script that manages the execution of Intel Advisor and Intel VTune:

- Ratio of giga floating-point operations per second (GFLOPS);
- Arithmetic Intensity (AI) which is the ratio of giga floating-point operations by data transfer;
- Clockticks per Instructions Retired (CPI). That is calculated by dividing the number of unhalted processor cycles (Clockticks) by the number of instructions retired, that is the instructions that effectively finishes its execution.

The GFLOPS measures the amount of operations that an application is capable of executing considering the useful work and any other overhead that can be present in the application, such as, cache miss, data transfer latency from main memory to processor, vectorization overhead, and so on.

The strategy identifies the overhead present in the application by the means of AI and CPI.

The CPI is a measure that helps estimate the overhead present in the application, because one the consequences of high overhead can be a decrease on retired instructions.

The AI measures the throughput of instructions execution. In this sense, higher level of AI indicates that the execution of application demands less data transfers from memory to CPU, which means that the latency is lower.

4.2 Data Analysis and Output

The strategy returns the performance prediction by the means of a numbered list that classify architectures from lowest to highest performance gain, based on a metric that we call estimated processing capacity (EPC) and is defined in the formula on equation (1).

$$EPC(Application, Resource) = \frac{GFLOPS * AI}{CPI} \quad (1)$$

The GFLOPS represents the amount of operations that an application is capable of perform on an architecture and AI represents the throughput of instructions execution, so to define the comparison metric we multiply these two variables in order to obtain a quality estimate about how the application is using the architecture. We divide by CPI, because it represents the applications overhead on an architecture, CPI level is proportional to overhead level. In this sense, lower CPI level will have lower impact on EPC.

The strategy calculates the EPC for an application for each resource. The list of value is normalized by one to the amount of resources defined in the resource set. This normalization is important because the value returned from EPC can not be used to characterize a proportional performance gain across different architectures.

5 Evaluation

In this section, we present an evaluation of the strategy Session 5.1 presents the workload and hardware infrastructure used to evaluate the strategy and Session 5.2 presents the results obtained.

5.1 Workload and Hardware Description

We created a workload composed by applications with different characteristics from a set of problem organized by Intel [9]:

- A matrix multiplication code developed using Intel Intrinsics;
- A numeric model in finance optimized using AVX-512 ER (Exponentials and Reciprocals);
- A N-Body simulation with major part of code vectorized;
- A Diffusion simulation developed using scalar instructions in major part.

The hardware we are going to use to compare the performance are presented in the Table 1.

Table 1. Hardware Infrastructure used for Evaluation.

Architecture	Processor	Cores	Threads	Dram
SandyBridge	2x 2.6GHz	8	32	64GB
Haswell	2x 2.3GHz	36	72	128 GB
Skylake	2x 2.1GHz	48	96	192 GB
Knights Landing (cache mode)	1x 1.4GHz	68	272	192 GB
Knights Landing (flat mode)	1x 1.4GHz	68	272	192 GB

5.2 Results and Discussion

We executed the strategy with the workload and infrastructure presented in Section 5.1 and show the results in Table 2.

Table 2. Comparing Performance Classification from Strategy against Execution Time.

Numeric Model in Finance					
Architecture	Skylake	Haswell	SandyBridge	KNL (Flat Mode)	KNL (Cache Mode)
Execution Time (Seconds)	458	1036	3443	235	224
Performance Classification	3	2	1	4	5
Diffusion					
Architecture	Skylake	Haswell	SandyBridge	KNL (Flat Mode)	KNL (Cache Mode)
Execution Time (Seconds)	511	309	220	425	1557
Performance Classification	3	4	5	2	1
N-Body					
Architecture	Skylake	Haswell	SandyBridge	KNL (Flat Mode)	KNL (Cache Mode)
Execution Time (Seconds)	306	467	1253	343	347
Performance Classification	5	4	3	1	2
Matrix Multiplication (Intrisincs)					
Architecture	Skylake	Haswell	SandyBridge	KNL (Flat Mode)	KNL (Cache Mode)
Execution Time (Seconds)	172	159	344	132	227
Performance Classification	4	5	1	3	2

The results show that three applications were correctly classified according to performance gains.

The Numeric Model for Finance is optimized for AVX-512 ER, so the execution on KNL returned a much higher value for AI than in any other architecture, which demonstrates the relation of AI on increase the precision of performance prediction.

The strategy predicted that KNL present the worse performance to execute N-Body, but the execution time shows that KNL presents the best performance. This happened because for this work we are using metrics that corresponds to an average of entire application, in the case of N-Body some loops which is in the critical path presents higher performance on KNL that was not expressed in the average result.

6 Conclusions and Future Work

This work presented a strategy to identify the best architecture for a given application. It can be helpful when buying new compute nodes for a running cluster, decide in which nodes of a cluster an application is preferable to run (schedule decisions for example) and so on. Three of four applications used in our experiments were correctly predicted using our strategy. The low overhead of the strategy is another feature that differs from existing predictions strategies.

As future work, we intend to extend the metrics used to perform the classification to increase accuracy, such as loops and others events (i.e. cache hit rate).

6.1 Acknowledgements

The authors would like to thank the Center for Scientific Computing at the São Paulo State University (NCC/UNESP) for the use of the manycore computing resources, partially funded by Intel in the context of the following projects: "Intel Parallel Computing Center", “Intel Modern Code Partner”, and "Intel/Unesp Center of Excellence in Machine Learning”.

References

1. Yang, X. J., et al.: The TianHe-1A supercomputer: its hardware and software. Journal of computer science and technology, 344-351 (2011).
2. Rosales, C., et al.: Performance Prediction of HPC Applications on Intel Processors. In: Parallel and Distributed Processing Symposium Workshops (IPDPSW), IEEE (2017).
3. Lee, S., Meredith, J. S., Vetter, J.S.: Compass: A framework for automated performance modeling and prediction. In: Proceedings of the 29th ACM on International Conference on Supercomputing, ACM (2015).
4. McCalpin, J. D.: Memory bandwidth and machine balance in current high performance computers. IEEE computer society technical committee on computer architecture (TCCA) newsletter 2, 19–25 (1995).
5. Obaida, M. A., et al.: Parallel Application Performance Prediction Using Analysis Based Models and HPC Simulations. In: Proceedings of the 2018 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation, ACM (2018).
6. Benoit, N., Louise, S.: A First Step to Performance Prediction for Heterogeneous Processing on Manycores. Procedia Computer Science 51, 2952-2956 (2015).
7. Escobar, R., Boppana, R.V.: Performance prediction of parallel applications based on small-scale executions. In: High Performance Computing (HiPC), 2016 IEEE 23rd International Conference, IEEE (2016).
8. Browne S., Dongarra J., Garner N., London K., and Mucci P.. 2000. A scalable cross-platform infrastructure for application performance tuning using hardware counters. In Proceedings of the 2000 ACM/IEEE conference on Supercomputing (SC '00). IEEE Computer Society, Washington, DC, USA, , Article 42.
9. Reinders J. and Jeffers J. High Performance Parallelism Pearls: Multicore and Many-core Programming Approaches, 2015.