Machine Learning Performance Prediction Model for Heterogeneous Systems

Daniel Nemirovsky*, Tugberk Arkose*, Osman Unsal*, Adrian Cristal* and Mateo Valero*
*Barcelona Supercomputing Center
{daniel.nemirovsky, tugberk.arkose, osman.unsal, adrian.cristal, mateo.valero}@bsc.es

Recent advances in machine learning techniques and specialized hardware has enabled a resurgence in the interest and applicability of powerful artificial neural network based prediction systems. However, as of yet no significant leaps have been taken towards applying machine learning in heterogeneous scheduling in order to maximize system throughput. As heterogeneous systems become more ubiquitous, computer architects will need to develop new CPU scheduling approaches applying novel techniques capable of exploiting the diversity of computational resources. However, non-heterogeneous aware schedulers like the current Linux Completely Fair Scheduler (CFS) [3] cannot take advantage of diverse system resources. Heterogeneous scheduling approaches have been previously proposed by V. Craeynest et al. [6] and Markovic et al. [2] which intend to provide full and fair utilization of the different hardware resources for all threads. These approaches yielded significant performance benefits compared to the CFS on heterogeneous architectures. In this extended abstract, we describe a novel performance prediction model which is the first of its kind to utilize machine learning performance predictors at the granularity of scheduling quanta. We then highlight how a heterogeneous system scheduler may be improved by the addition of this model.

The prediction model is composed of the following:

- Statistical information about each thread's state and which core type it has been executing on.
- A next quantum behavior predictor (NQP) that predicts what will be a thread's behavior during the next scheduling quantum.
- Machine learning based performance predictors which use a thread's behavior statistics to estimate its performance for a given core type.

Recognizing and exploiting the variations in program behavior is instrumental for effective schedulers in achieving optimal mapping schemes to maximize system performance. While not all programs exhibit the same behavior, studies [1], [4] have shown that the behavioral periodicity in different applications is typically consistent. In fact, the behavioral periodicity has been shown to be roughly on the order of several millions of instructions and is present in various different and even non correlated metrics stemming from looping structures inside of applications.

In addition, in order to provide contextual awareness to a CPU scheduler, certain thread and hardware statistics should be periodically collected. These may include values indicating a thread's state (e.g. running, ready, or stalled), execution time, number of instruction executed, types of instructions executed, number of memory operations, cache accesses as well as hit or misses, and available cores and their types. The amount of statistics needed to be collected depends upon the complexity and optimization scheme of the scheduler.

Several novel approaches such as [5] have been proposed which predict program behavior based upon various statically or dynamically collected program statistics. However, for our model, we propose using a next quantum thread behavior predictor (NQP) that will always predict the next behavior to be equal to the previous quantum behavior. It takes as input a selection of thread statistics and by utilizing the "previous-value" prediction, it then passes these same thread statistics as the input parameters to the machine learning based performance predictors.

The machine learning architectures that the predictors are based upon can include a diverse selection. Artificial neural networks, decision trees, and clustering schemes such as k-means or nearest neighbors are examples of machine learning (ML) models that can be utilized alone or in conjunction as an ensemble to improve the prediction accuracy and training generalization. A separate ML based predictor (whether individual or ensemble based) should be implemented and trained for each different core type present in the target system. Therefore, the ML predictors take as input the thread statistics (which describe its expected behavior for the next quantum) and predict an estimated IPC value. This IPC value is the performance expected of executing that thread on the core type for which the predictor has been trained. The central features to research in detail are: the hyper-parameters of the ML predictors, the target applications of the system (or what benchmarks provide enough generalization for most programs), and which thread statistics to collect (parameter engineering). The hyper-parameters include items such as the number of hidden units and hidden layers of a neural network, the training algorithms, and which regularization techniques to use. These may be fine tuned by hand or through a machine learning approach as well and should be further evaluated by the amount of overheads that they would add. For instance, a leaner but less precise neural network may lead to a more optimal all around predictor when considering overheads compared to a very deep or complex ensemble of predictors.

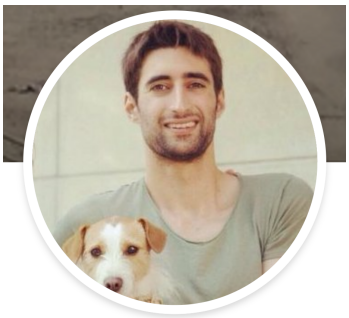
Identifying target applications is also critical in the sense that

the data collected for training, validation, and testing should be composed of execution results from benchmarks which reflect these target applications. These data should be gathered by running the predetrmined benchmarks on the different core types and sampling the IPC result and other thread statistics periodically after each scheduling quantum. The model could also be trained online to improve its ability to predict for newly executed applications dynamically.

When a scheduling quantum is completed, the statistics of each thread passes through the NQP and then the predictors of all the different core types (with the exception of the core type the thread last executed on since the NQP prediction will suffice in this case). In this manner, the system throughput of all the different thread mappings on the heterogeneous architecture can be compared. Assuming that the prediction error of the model is not onerously high, then these comparisons will offer a heterogeneous system scheduler ample knowledge in order to constantly maximize the system throughput. Issues to be improve upon and included in the future consist of specific ML model implementations, parameter tuning, including thread interference effects, and possibly targeting other scheduling goals such as reducing energy consumption.

I. Bio

Daniel Nemirovsky is a PhD candidate at the Universitat Politecnica de Catalunya (UPC) and a researcher in the Computer Architecture for Parallel Paradigms Group at the Barcelona Supercomputing Center. His research includes improving heterogeneous systems using machine/deep learning techniques, CPU schedulers alternative cache configurations, and advanced heterogeneity software/hardware abstractions. Daniel holds an MSc in computer architecture, networks, and systems from the UPC and dual bachelors in electrical engineering and political science from the University of Michigan.



References

[1] E. Duesterwald, C. Cascaval, and S. Dwarkadas, "Characterizing and predicting program behavior and its variability," in *Parallel Architectures and Compilation Techniques, 2003. PACT 2003. Proceedings. 12th International Conference on*. EEE, 2003, pp. 220–231.

[2] N. Markovic, D. Nemirovsky, V. Milutinovic, O. Unsal, M. Valero, and A. Cristal, "Hardware round-robin scheduler for single-isa asymmetric multi-core," in *European Conference on Parallel Processing*. Springer, 2015, pp. 122–134.

[3] C. S. Pabla, "Completely fair scheduler," *Linux Journal*, vol. 2009, no. 184, p. 4, 2009.

[4] T. Sherwood, E. Perelman, G. Hamerly, S. Sair, and B. Calder, "Discovering and exploiting program phases," *IEEE micro*, vol. 23, no. 6, pp. 84–93, 2003.

[5] O. S. Unsal, I. Koren, C. Khrishna, and C. A. Moritz, "Cool-fetch: A compiler-enabled ipc estimation based framework for energy reduction," in *Interaction between Compilers and Computer Architectures, 2004. INTERACT-8 2004. Eighth Workshop on*. IEEE, 2004, pp. 43–52.

[6] K. Van Craeynest, S. Akram, W. Heirman, A. Jaleel, and L. Eeckhout, "Fairness-aware scheduling on single-isa heterogeneous multi-cores," in *Proceedings of the 22nd international conference on Parallel architectures and compilation techniques*. IEEE Press, 2013, pp. 177–188.