Predicting Cross-Platform Relative Performance with Deep Generative Models

Daniel Nichols†, Jae-Seung Yeom*, Abhinav Bhatele†

†Department of Computer Science, University of Maryland, College Park, MD 20742 USA
*Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA 94551 USA

***Abstract*—Applications can experience significant performance differences when run on different architectures. For example, GPUs are often utilized to accelerate an application over its CPU implementation. Understanding how performance changes across platforms is vital to the design of hardware, systems software, and performance critical applications. However, modelling the relationship between systems and performance is difficult as run time data needs to be collected on each platform. In this abstract, we present a methodology for predicting the relative performance of an application across multiple systems using profiled performance counters. This model will be useful for integrating high performance computing software into heterogeneous environments such as cloud systems.**

***Index Terms*—performance, performance modelling, deep learning**

Multi-staged, pipelined workflows that combine high performance computing (HPC) simulations, machine learning, and data analytics are becoming increasingly popular within scientific computing research. The tasks in these workflows often span a variety of heterogeneous resources and compute requirements. Thus, running these workflows in cloud environments is desirable as their hardware and software is designed to facilitate large numbers of tasks across heterogeneous resources. However, effectively utilizing cloud is difficult as HPC software is not well supported. One means to bridge this gap is multi-resource task scheduling.

Scheduling jobs *efficiently* across resources is far from trivial. It requires a model for how applications will behave across the spectrum of available resources. When a scheduler has access to multiple resource types it can more efficiently schedule jobs across them. However, to effectively make use of these available resources it needs needs to know the relative performance of a job across them. Typically runtime estimates are provided to a scheduler by the user as a max wall-time. As the number of resources available to a scheduler grows, this becomes impractical to require from the user. Thus, some performance model has to provide these estimates of relative performance to the scheduler. With these it is then able to decide which resources and how much of them to allocate.

We propose a machine learning based methodology that can learn to predict relative performance across a set of platforms based on profiled system counters. We first learn a latent space that captures information about the relationship of performance counters across a set of platforms. Then we use counters mapped into this latent space to predict relative performance. By learning a latent space first it is simple to continue learning different downstream tasks with this data.

TABLE I
TABLE OF RECORDED PERFORMANCE COUNTERS. DATA LOCALITY AND MEMORY RELATED COUNTERS ARE IN BLUE. CONTROL FLOW AND PARALLELISM RELATED COUNTERS ARE IN RED. I/O RELATED COUNTERS ARE IN GREEN.

Counter
Instructions
Ratio Int Arith. to Total Instr.
Ratio Float Arith. to Total Instr.
Ratio Mem. to Total Instr.
Ratio Caches Misses to Hits
Page Levels
Page Faults
GPU2CPU & CPU2GPU Bandwidth
Ratio Control to Total Instr.
Ratio Indep. Instr. to Total Instr.
Ratio Branch Misses to Branch Instr.
Threads / Streams
Warp Blocks
IO Bytes Read/Written
IO File Descriptors Opened

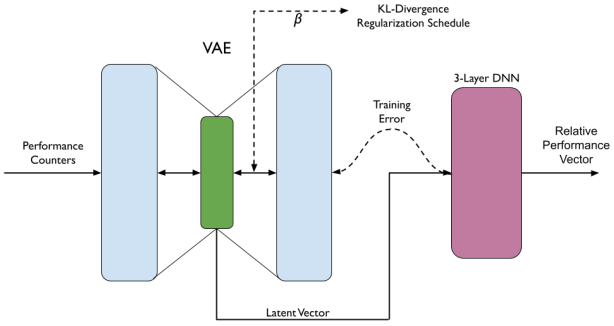

Fig. 1. An overview of the VAE training. The VAE is trained on performance counter data and over a range of epochs it receives feedback from the downstream DNN regressor. Additionally, the decoder update weight is changed gradually with a KL-divergence-based regularization scheme.

Before we train the model we first collect data from a variety of applications. Profiling counters for the gpu-enabled applications in the e4s [1] and ecp-proxy-applications [2] suites are recorded on four different systems at Lawrence Livermore National Laboratory (LLNL): Quartz, Ruby, Corona,

[1]https://e4s-project.github.io/index.html
[2]https://proxyapps.exascaleproject.org/

and Lassen. Quartz and Ruby are CPU based machines with Intel Xeon E5-2695 v4 and CLX-8276L CPUs, respectively. Corona and Lassen are GPU enabled machines with AMD MI50 and NVIDIA V100 GPUs, respectively. Performance counters are collected with HPCToolkit [1] and PAPI [2] and are listed in Table I. Figure 2 presents the distribution of relative performance across the four machines in the data set and Figure 3 shows the relative performances for a particular application Laghos [3].

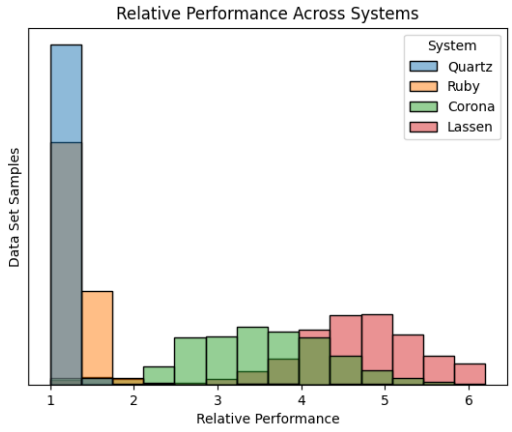

Fig. 2. The distribution of relative performance numbers in our dataset across each machine.

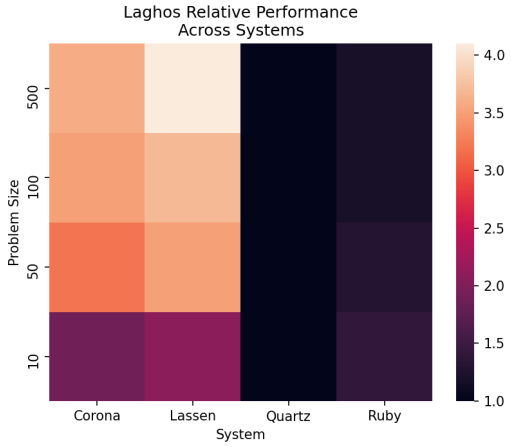

Fig. 3. An example relative performance vector for the Laghos [3] proxy application.

Using this data set, a variational auto-encoder (VAE), as highlighted in Figure 1, is trained to learn a latent representation of the counter data. The VAE decoder is *calibrated* [4] on just the CPU data set before training on the entire data set. Additionally, the KL-divergence loss is weighted with a changing regularization parameter. For several iterations the loss from the downstream regression task is also included in the VAE loss. The latent representation learned by the VAE is then used to train a 3-layer deep neural network (DNN) that predicts the relative performance value across the four systems ($\in \mathbb{R}^4$). Each layer of the DNN has 1024 hidden units.

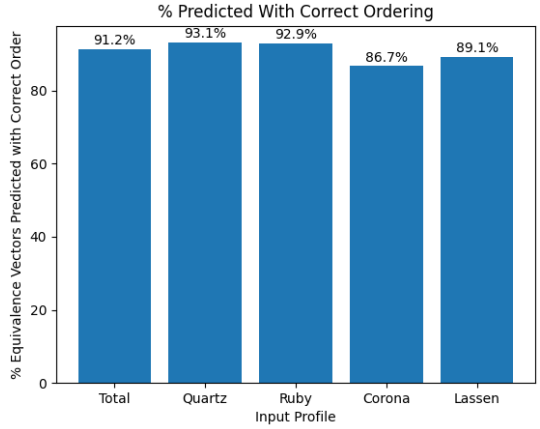

Fig. 4. Percentage of samples that the DNN generates a relative performance vector in the correct order for.

With this training setup and an 80-20 cross-validation split the DNN trains to a final R^2 score of 0.81. A baseline RandomForest regressor achieves $R^2 = 0.68$. Most notably the DNN predicts the vector with the correct ordering for $\approx 91\%$ of samples (see Figure 4). Since $\approx 80\%$ of the data set is in the order (Quartz, Ruby, Corona, Lassen) we mark this as the *positive* outcome and look the specificity/sensitivity. Figure 5 shows that the model performs well above random guessing the order based on priors.

In conclusion we have presented a methodology for predicting relative performance across platforms based on profiled counters. In future work we will focus on improving the final R^2 score, develop transfer learning based methods to model new resource sets with few samples, and apply the model to end applications such as multi-resource job scheduling.

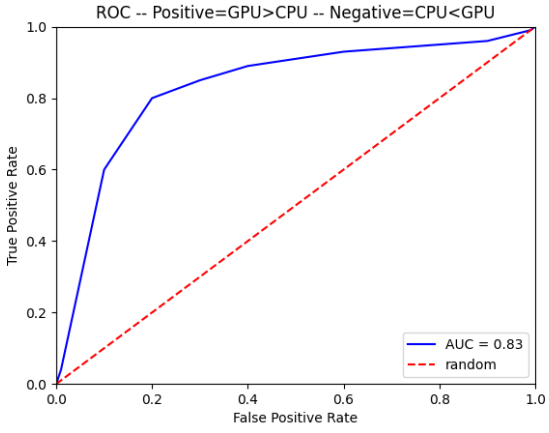

Fig. 5. ROC curve for DNN. Here a run with order (Quartz, Ruby, Corona, Lassen) is considered a positive sample. The red line represents the baseline performance of a random classifier.

ACKNOWLEDGMENT

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 (LLNL-ABS-838516).

REFERENCES

[1] L. Adhianto, S. Banerjee, M. Fagan, M. Krentel, G. Marin, J. Mellor-Crummey, and N. R. Tallent, "Hpctoolkit: Tools for performance analysis of optimized parallel programs," *Concurrency and Computation: Practice and Experience*, vol. 22, no. 6, pp. 685–701, 2010.

[2] D. Terpstra, H. Jagode, H. You, and J. Dongarra, "Collecting performance data with papi-c," in *Tools for High Performance Computing 2009*, M. S. Müller, M. M. Resch, A. Schulz, and W. E. Nagel, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 157–173.

[3] V. A. Dobrev, T. V. Kolev, and R. N. Rieben, "High-order curvilinear finite element methods for lagrangian hydrodynamics," *SIAM Journal on Scientific Computing*, vol. 34, no. 5, pp. B606–B641, 2012. [Online]. Available: https://doi.org/10.1137/120864672

[4] O. Rybkin, K. Daniilidis, and S. Levine, "Simple and effective vae training with calibrated decoders," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 9179–9189. [Online]. Available: https://proceedings.mlr.press/v139/rybkin21a.html