Predicting Cross-Platform Relative Performance with Deep Generative Models

Daniel Nichols¹, Jae-Seung Yeom², Abhinav Bhatelé¹
¹University of Maryland, College Park, ²Lawrence Livermore National Laboratory

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 paper, we present a methodology for predicting the relative performance of an application across multiple systems using profiled performance counters.

Motivation
Predicting relative performance enables easier use of performance modelling results for downstream tasks. A motivating example of this is releasing performance counters.

Problem Overview
Goal: Learn a latent space for a fixed set of resources that can map tasks to their relative performance across those resources.

Training Results
• Models are trained with 80/20 cross-validation split.
  • DNN regressor has R² value of ≈0.81. Baseline RandomForest regressor has R² value of ≈0.68.
  • >91% of predicted relative performance vectors are in the correct order.
  • Predictions are often better using CPU profiles.

Data Set
Recorded counters for E4S¹, EC², and cora² applications on 4 LINL systems:

<table>
<thead>
<tr>
<th>Counter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Control flow & parallelism
• Measure instruction level parallelism
• Divergent behavior is bad for GPU performance

Conclusion and Future Work
We are able to train a deep learning model to generate relative performance vectors for a set of resources based on profiled counter data.

Future work will:
• Explore transfer learning to retrain the model for new resource sets with few samples.
• Include resource property counters in input so model can handle unseen architectures.
• Utilize performance modelling in multi-system job scheduler.

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

Image Annotations
- VAE: Variational Autoencoder
- KL-Divergence: Kullback-Leibler divergence
- 3-Layer DNN: Three-layer Deep Neural Network

Image 1: Workflow Diagram
- Task 1: Input Domain
- Task 2: Performance Map
- Task 3: Latent Space

Image 2: Data Set Table
- Examples of different counters recorded for various applications.

Image 3: Model Diagram
- VAE: Variational Autoencoder
- KL-Divergence: Kullback-Leibler divergence
- 3-Layer DNN: Three-layer Deep Neural Network

Acknowledgements
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-POST-838517).

Image 4: Performance Counter Graph
- Comparison of relative performance across different platforms.