Modeling the Impact of Thread Configuration on Power and Performance of GPUs

Tiffany Connors
Texas State University
San Marcos, TX
Email: tac115@txstate.edu

Apan Qasem (Advisor)
Texas State University
San Marcos, TX
Email: apan@txstate.edu

Qing Yi (Advisor)
University of Colorado at Colorado Springs
Colorado Springs, CO
Email: qyi@uccs.edu

***Abstract*—Because graphics processing units (GPUs) are a low-cost option for achieving high computational power, they have become widely used in high-performance computing. However, GPUs can consume large amounts of power. Due to the associated energy costs, improving energy-efficiency has become a growing concern. By evaluating the impact of thread configuration on performance and power trade-off, energy-efficient solutions can be identified.**

The impact that a thread configuration will have on the performance and power trade-off of a GPU kernel can be accurately predicted using machine learning. Using dynamic features of a GPU kernel as input, a machine learning model can be used to assist in the selection of thread configurations which will improve performance and minimize power consumption.

I. Introduction

Due to the prevalence of GPUs in computational intensive work, there is a need for solutions which will decrease the associated energy costs of GPUs while continuing to provide performance speedup. Through the modeling of performance and power consumption, it is possible to identify a correlation between the two and determine ways in which to make systems more energy-efficient while continuing to provide high levels of performance speedup.

II. Methodology

This work focuses specifically on optimizing the performance and power trade-off of search algorithms. We investigated the impact of 20 feasible thread configurations, shown in Table I, on the power and performance of four CUDA programs. These programs solve the quadratic assignment problem using 4 different algorithms: tabu search, simulated annealing, and 2 variations of 2opt. Additionally, three input datasets were used from QAPLIB[1]: lipa20, lipa30, and tai25a. Each dataset varies in size and structure, thus affecting the program's behavior.

TABLE I
Set of thread configurations

64x32	128x16	256x8	512x4	1024x2
64x64	128x32	256x16	512x8	1024x4
64x96	128x48	256x24	512x12	1024x6
64x128	128x64	256x32	512x20	1024x8

The 20 thread configurations were applied individually to the four different codes, resulting in 20 different versions of each program. The GPU kernels were then executed using the three input datasets. For each run, the average power consumption was measured using the built-in power sensor of the Tesla K20c GPU and the execution time was recorded.

The change in execution time and power consumption was calculated by comparing the measured values to those of each version of the same program and the trade-off was computed.

$$trade-off = \frac{\Delta\ execution\ time}{\Delta\ power\ consumption}$$

The row of data was then assigned a classification of good or bad based on the trade-off value:

Good: trade-off ≥ 1
Bad: trade-off < 1

A. Feature Extraction

To create a machine learning model that works with more than one code, features which provide a good description of the kernel's characteristics must be determined. To do this, runtime behavior was analyzed using NVIDIA's GPU profiler, `nvprof`, and a set of 52 dynamic features was extracted. To normalize the values collected, each feature was divided by the number of instructions executed. The set of features was standardized by computing the z-score:

$$z_1 = \frac{x_1 - \mu}{\sigma}$$

B. Machine Learning

Nine different machine learning algorithms available from the R caret package were used in this work. The purpose of using varied methods was to determine if the same features were significant factors across all models, as well as identify which algorithms worked best with our data.

The data was sorted into separate files based on target thread configuration. Each file was treated independently and separate models were built, trained, and tested for each of these configurations. The model accuracy was determined using repeated k-fold cross-validation. The machine learning algorithm which performed the best across all models was selected to be used for building the final predictive model.

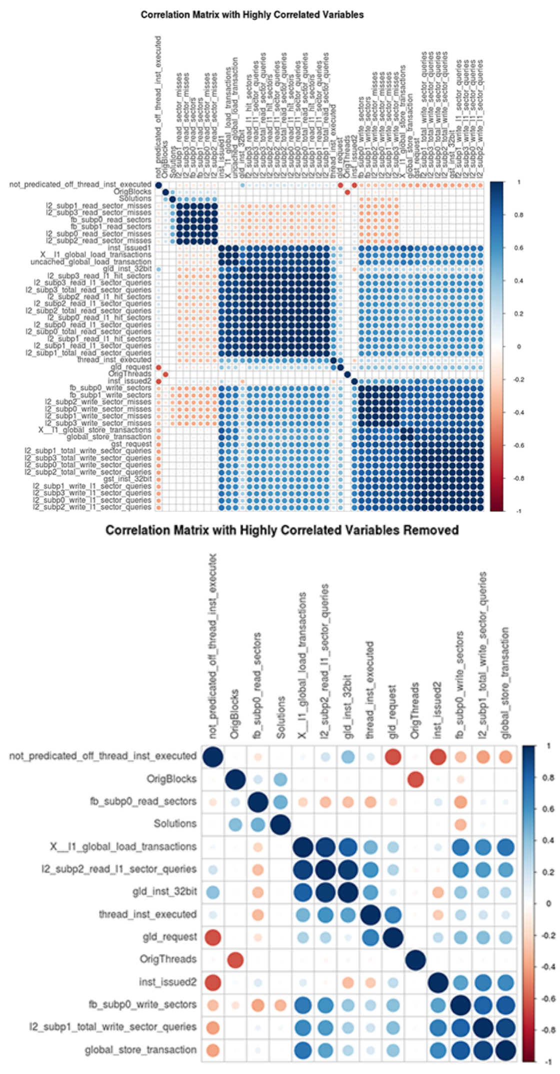

Fig. 1. Correlation matrices with and without highly-correlated variables.

C. Feature Selection

To reduce our set of features to include only those with the highest predictive power, feature selection was performed. First, any features with zero variance were removed. Features were then analyzed in R using correlation matrices, Figure 1.

A feature is considered good if it is not highly correlated to the other relevant features[2]. Therefore, any features with a correlation of 95% or higher to another feature were removed from the set. The remaining features were compared with each of the models' variable importance values to determine if the features identified as significant factors remained in the new set of features.

III. Results

Through feature selection, the initial set of features was reduced from 52 to 14. OrigThreads was most frequently identified as the most significant factor. It appears that if the thread count is large, reducing it to a lower number will be more beneficial.

The thread configuration 128x16 produced desirable trade-off 89.58% of the time. Changing the configuration to 1024x8 was always bad, therefore we were unable to build a model for 1024x8 since the response label had no variance.

The predictive model was built using the boosted C5.0 tree algorithm. Its average accuracy rate was 96%. The model predicted that modifying the configuration to 128x16 will negatively impact trade-off if OrigThreads is small, the number of instructions sent to 32-bit global memory is low, and a smaller number of read requests are sent to subp0. Otherwise, the change will likely result in improved trade-off.

IV. Conclusion

To improve the energy-efficiency of GPU systems, this poster proposes using machine learning to aid in the selection of thread configurations which increase performance while maintaining minimal increase in power consumption. By using dynamic feature extraction and selecting features most closely related to thread configuration, a machine learning model can accurately predict the impact of a thread configuration on the performance/power trade-off of a program.

Manually determining which thread configuration will provide optimal results is time consuming and involves modifying and running the program using each candidate thread configuration. This study addresses this issue by eliminating the need to modify and test the program for each candidate thread configuration, thus improving the efficiency of the process. Once set-up, the framework presented in this study requires only a single run in order to extract the dynamic features used by the predictive model.

V. Future Work

In addition to using machine learning to predict if a change in thread configuration will have a positive or negative impact on trade-off, we intend to create a model whose output is a recommended thread configuration. Future work will expand the training dataset to include features from stencil code and benchmark suites, allowing us to determine if the same features important for predicting the impact of thread configuration on QAP solvers are also significant for other codes.

Acknowledgment

This material is based upon work supported by the National Science Foundation REU program under Grant No. 1359275.

References

[1] "QAPLIB - A Quadratic Assignment Problem Library" http://anjos.mgi.polymtl.ca/qaplib/, 2014

[2] L. Yu and H. Liu, "Feature Selection for High-dimensional Data: A Fast Correlation-Based Filter Solution," in *Proc. of the 12th International Conference on Machine Learning,* ICML'03, Washington, DC, 2003, pp. 856-863.