Performance Prediction Model for Applications on GPUs

Sriram Kanchiraju[1], Pinak Panigrahi[1], Ashok Srinivasan[2], Pallav Kumar Baruah[1]

[1]*Dept. of Mathematics and Computer Science, Sri Sathya Sai Institute of Higher Learning, Prasanthi Nilayam, India*

[2]*Dept. of Computer Science, Florida State University, Tallahassee, USA*

Motivation

- The modern GPUs coupled with the programming environments like NVIDIA's CUDA have seen an enormous amount of growth in terms of computational power and have become popular platforms for parallel computing.
- However, these developments pose some difficulties for users, including the following:
  - They need to know if the new architecture would be suitable for their application i.e., they need to predict the performance of their application on these new architectures.
  - They need to choose an optimal configuration from an extremely large configuration space.
- One typically wants to learn these with minimal effort so that the workload on the programmers is reduced.

Current Approaches

- Ryoo et.al [1] derive metrics from static code that capture various factors influencing performance. These metrics are then used to prune the configuration space down to those that lie on a pareto-optimal curve. However, this does not consider memory bound kernels.
- The authors of [2] propose a performance prediction model for the CUDA GPGPU platform that encompasses the various facets of GPU architecture like scheduling, memory hierarchy, etc. However, their model does not consider intra-block synchronization calls and atomic operations.
- The authors of [3] propose a automated GPU performance exploration framework that uses stepwise regression modeling. The result is an application specific model.

Our approach

- Let the time taken be $T(C)$, where C is the configuration space. In our case, this refers to various number of *threads per block* of a CUDA kernel.
- We assume that $T(C)$ is a linear combination of some basis functions for different values of C from a finite set $\dot{C} = \{ c_1, c_2, \ldots, c_n \}$.
- We will use a model reduction technique (like PCA) to figure out the basis functions (β_i). For each basis function we create a vector with its i^{th} component being the value of the function for c_i number of threads per block.
- We express the time taken in terms of a small number (k) of basis vectors. i.e. $T = \sum_{i=1}^{k} x_i \beta_i$
- T is a vector whose i^{th} component gives $T(c_i)$.

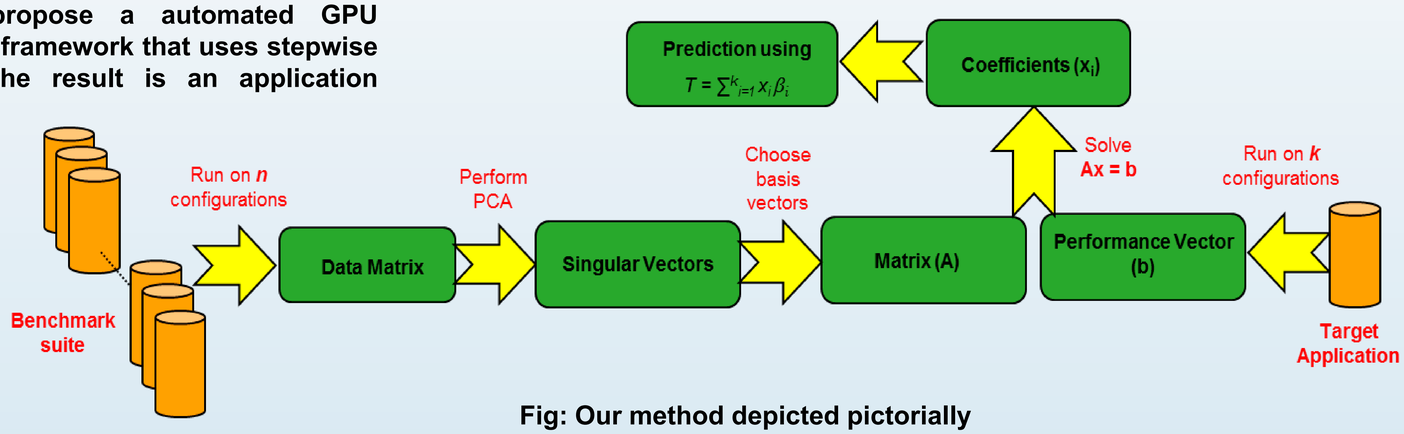

Fig: Our method depicted pictorially

Results

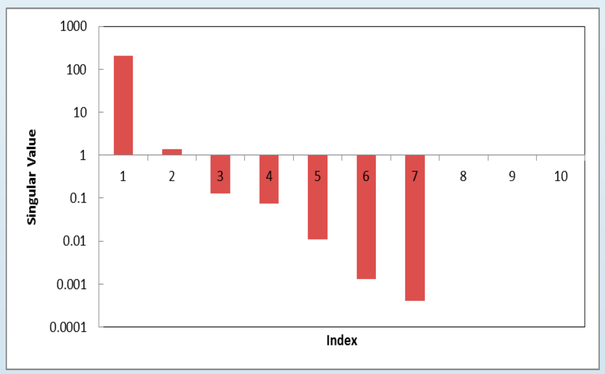

Fig 1: Singular values for the benchmark data plotted in a logarithmic scale

To illustrate our idea we timed the following benchmarks on Kepler K20: matmul, transpose from CUDA-SDK, and bfs, lud, hotspot, nw, pathfinder, and sradv from RODINIA.

We have transformed the timings to speedups in order to normalize the data from different runs. Our prediction process actually predicts the speedup, from which the timing can be easily obtained. Fig. 1 shows that much of the data is accounted in the first 7 singular vectors. Hence we predict using these seven singular vectors. Note that in fig. 2a, the speedup increases upto 324 and a slight drop in speedup at 400 and after 625, which our model predicts accurately. Moreover, interpolation would not be able to give such a prediction.

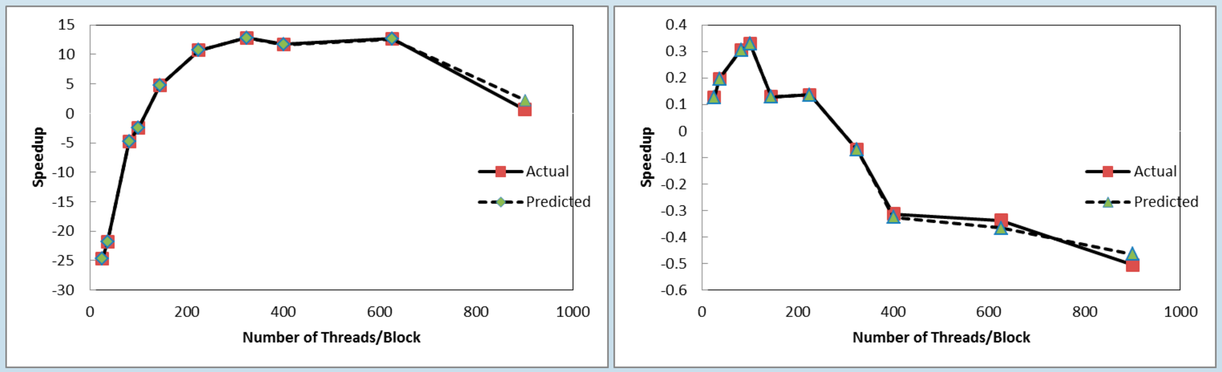

Fig 2: Prediction of speedup on 400, 625, and 900 threads/block with 7 basis vectors. (a) Hotspot (b) NW

Conclusions

- We have developed a performance model on GPUs that uses a dimensionality reduction technique and parameter estimation on the data collected from the standardized benchmark suite.
- The key idea is to use the performance numbers obtained from running the application of interest for a few configurations and predict for a larger number of configurations.
- In the future, we intend to show that our model predicts accurately for many more applications.

References

[1] RYOO, S., RODRIGUES, C. I., STONE, S., BAGHSORKHI, S. S., UENG, S.-Z., STRATTON, J. A., AND HWU, W. W. Program Optimization Space Pruning for a Multithreaded GPU. In *Proc. The Intl. Symp. Code Gen. and Opt. (2008), pp. 195-204.*

[2] K. Kothapalli, R. Mukherjee, M.S. Rehman, S. Patidar, P.J. Narayanan, and K. Srinathan. A performance prediction model for the CUDA GPGPU platform. In *High Performance Computing (HiPC),* 2009 International Conference on, pages 463{472. IEEE, 2010.

[3] W. Jia, K. Shaw, and M. Martonosi, *"Stargazer: Automated Regression-Based GPU Design Space Exploration,"* IEEE ISPASS, 2012.

ACKNOWLEDGEMENT: We are grateful to our Founder Chancellor, Bhagawan Sri Sathya Sai Baba for being a source of inspiration to all of us and for HIS unseen hand behind all the work.