STATuner : Efficient Tuning of CUDA Kernels Parameters

Ravi Gupta[1]; Ignacio Laguna[2]; Dong H. Ahn[2];
Todd Gamblin[2]; Saurabh Bagchi[1]; Felix Lin[1];
[1] Purdue University , [2] Lawrence Livermore National Lab

Lawrence Livermore National Lab

PURDUE UNIVERSITY

ABSTRACT

To predict parameters used for launching kernels on the accelerator in a Heterogeneous System. We study a well known tool "Occupancy Calculator" to see its effectiveness and identify its pitfalls.

We identify other static metrics that can be used to characterize an application and build a Support Vector Classifier as the feature set. Finally, we compare our model with "Occupancy Calculator" and see that we get good prediction with first trial itself. We are able to predict parameters with an upper bound of 15% overshoot as against Occupancy Calculator with bound of 32% overshoot.

INTRODUCTION

Heterogeneous computing systems consists of numerous GPUs and CPUs where the computation intensive and parallelizable region of the code runs on the GPU with very high throughput and relatively sequential part of the code runs on the CPU.

The code region designated to run on GPU is called a kernel. Each instance of the kernel code is called a thread. Number of threads that are launched depends on the data set partition. One important parameter in using CUDA and launching applications on GPU is the grouping of threads into a block(Block_Size) to effectively use the resources on a multiprocessor(SM) of GPU and have high throughput.

We propose a solution to predict the optimal Block_Size based on supervised machine learning and hence reduce effort related to tuning CUDA programs.

BACKGROUND

In the past, active blocks per SM has been linked with performance such that number of active warps should be high to hide memory latencies. "Occupancy Calculator" from NVIDIA is one such tool which tries to predict the number of active warps that will run on a GPU limited by three physical resources –

Registers Shared memory Blocks / SM

Figure 1. Inside a GPU Multiprocessor

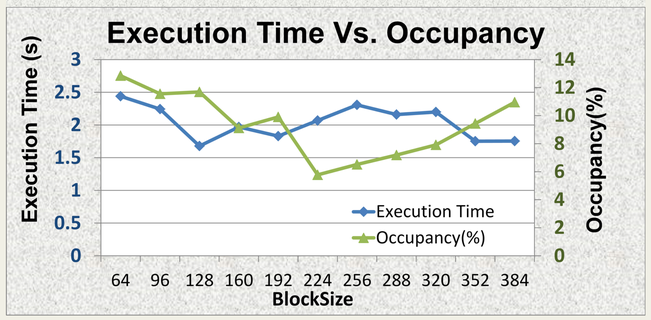

Figure 2. Execution Time vs. Occupancy Correlation

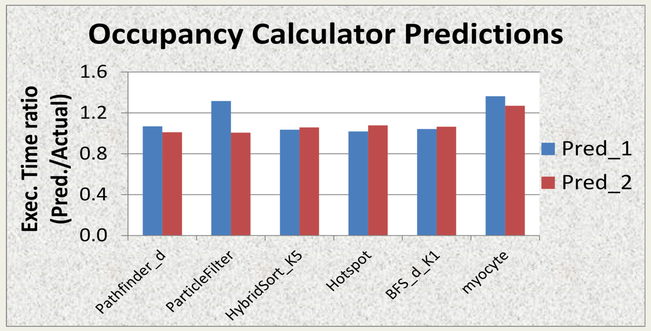

Figure 3. Occupancy Calculator Predicting Block_Size

METHODOLOGY

We extract the kernel code from the application code and use LLVM to identify static metrics to characterize the code. Only kernel code is relevant as that is the part of code that runs on the GPU. Some of the Static features we take into consideration –

-Total Insts. -Loops -Registers
-Branch Insts. -Shared Mem. -Memory Inst.

To use a supervised machine learning technique we first need to build a labeled training and test set. We use applications from Rodinia benchmark suite which is a well known benchmark suite for gauging heterogeneous computing.

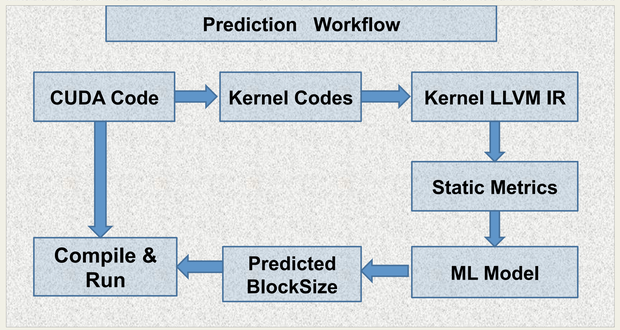

Figure 4. Prediction Workflow

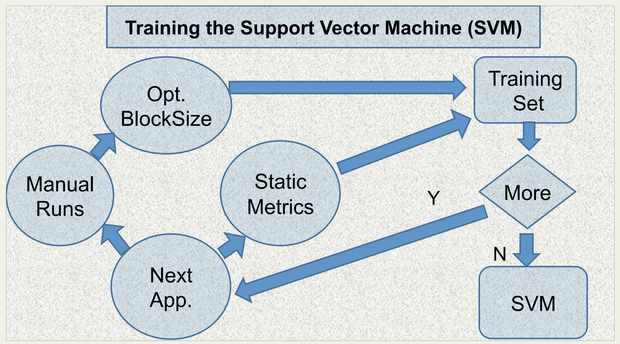

Figure 5. Training Support Vector Machine Classifier

SVM : Training Input Matrix								
	S1	S2	S3	S4	S5	S6	S7	Label(supervised)
App_1	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}	X_{16}	X_{17}	Y_1
App_2	X_{21}	X_{22}	X_{23}	X_{24}	X_{25}	X_{26}	X_{27}	Y_2
App_3	X_{31}	X_{32}	X_{33}	X_{34}	X_{35}	X_{36}	X_{37}	Y_3
.	.	.	.	.	.	.	.	Y_4
.	.	.	.	.	.	.	.	Y_5
App_n	X_{n1}	X_{n2}	X_{n3}	X_{n4}	X_{n5}	X_{n6}	X_{n7}	Y_6

Table 1. Training Input Matrix

RESULTS

- SVM Classifier with Kernel ='rbf'
- 'Classification' than 'Regression'
- Multi Labeling better than Single Labeling
- Train Set = 13 , Test Set = 5 , Features = 6

Occ_1 : Prediction 1 from Occupancy Calculator
Occ_2 : Prediction 2 from Occupancy Calculator
K=rbf,3 : SVM Prediction with 'rbf' Kernel, 3 labels
K=rbf,1 : SVM Prediction with 'rbf' Kernel, 1 label

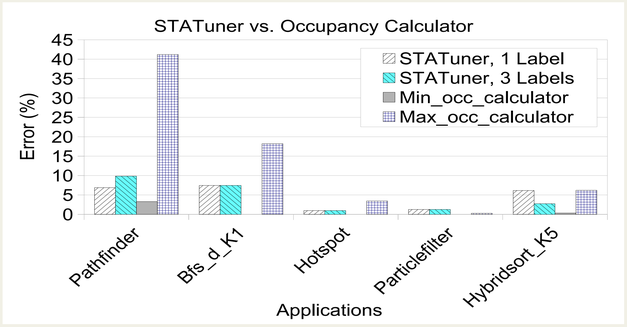

Figure 6. Percent Overshoot in Execution time

Average Percent Errors of the Models :
Occupancy_Calc: 6.6 % STATuner: 4.4 %

CONCLUSIONS

- Occupancy does not map very well to performance
- Static metrics characterize applications
- With enough metrics and training dataset, it is possible to define a good model
- Leverage Static metrics to reduce multiple trials
- Bound the percent overshoot to a limit with single trial

FURTHER WORK

We have here a very basic model with very basic features. We would like to find a more representative set of applications to base our training set on. We will improve our machine learning technique and also perform cross-validation to fine tune the model. We also plan to add more derived static features to characterize an application as well as dynamic metrics which can be collected from 'nvprof' NVIDIA profiler tool. We would like to extend our work to other known heterogeneous platforms like OpenCL and study its feasibility.

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344