Using Machine Learning to Automatically Tune GPU Program Performance

Tianyi David Han and Tarek Abdelrahman

Department of Electrical and Computer Engineering
University of Toronto

November 19, 2013

Why GPU Auto-Tuning?

- Optimization plays a critical role in better utilizing GPU compute power
- Optimization effect heavily depends on GPU architecture and other optimizations
  - Difficult to select the best-performing optimization(s)
- Fast-paced architectural enhancements demand frequent re-tuning

Why Machine Learning?

- Traditional approaches
  - Analytical modeling (heuristic)
  - Empirical search

	Performance	Speed	Amount of Effort

Why Machine Learning?

- Traditional approaches
  - Analytical modeling (heuristic)
  - Empirical search

	Performance	Speed	Amount of Effort
Analytical Modeling			

Why Machine Learning?

- Traditional approaches
  - Analytical modeling (heuristic)
  - Empirical search

	Performance	Speed	Amount of Effort
Analytical Modeling			
Empirical Search			

Why Machine Learning?

- Traditional approaches
  - Analytical modeling (heuristic)
  - Empirical search

	Performance	Speed	Amount of Effort
Analytical Modeling	🟠	🟢	🔴
Empirical Search	🟢	🔴	🟠
Machine Learning	🟢	🟢	🟠

Outline

- Motivation
- Feasibility study
  - Should loops be interchanged in image-processing kernels?
- Conclusions and future work

Feasibility Study: Application Domain

- Auto-tuning for mobile GPUs
- Computational photography on smartphones
- Start with image processing applications

Feasibility Study

- A typical image processing application

```
read image;
:


for (row = 0; row < img_rows; ++row)

    for (col = 0; col < img_cols; ++col)
    {
        read image pixels at and around (row,col);
        process image pixels;
        write image pixel at (row,col);
    }
}
:
write image;
```

Feasibility Study

- A typical image processing application

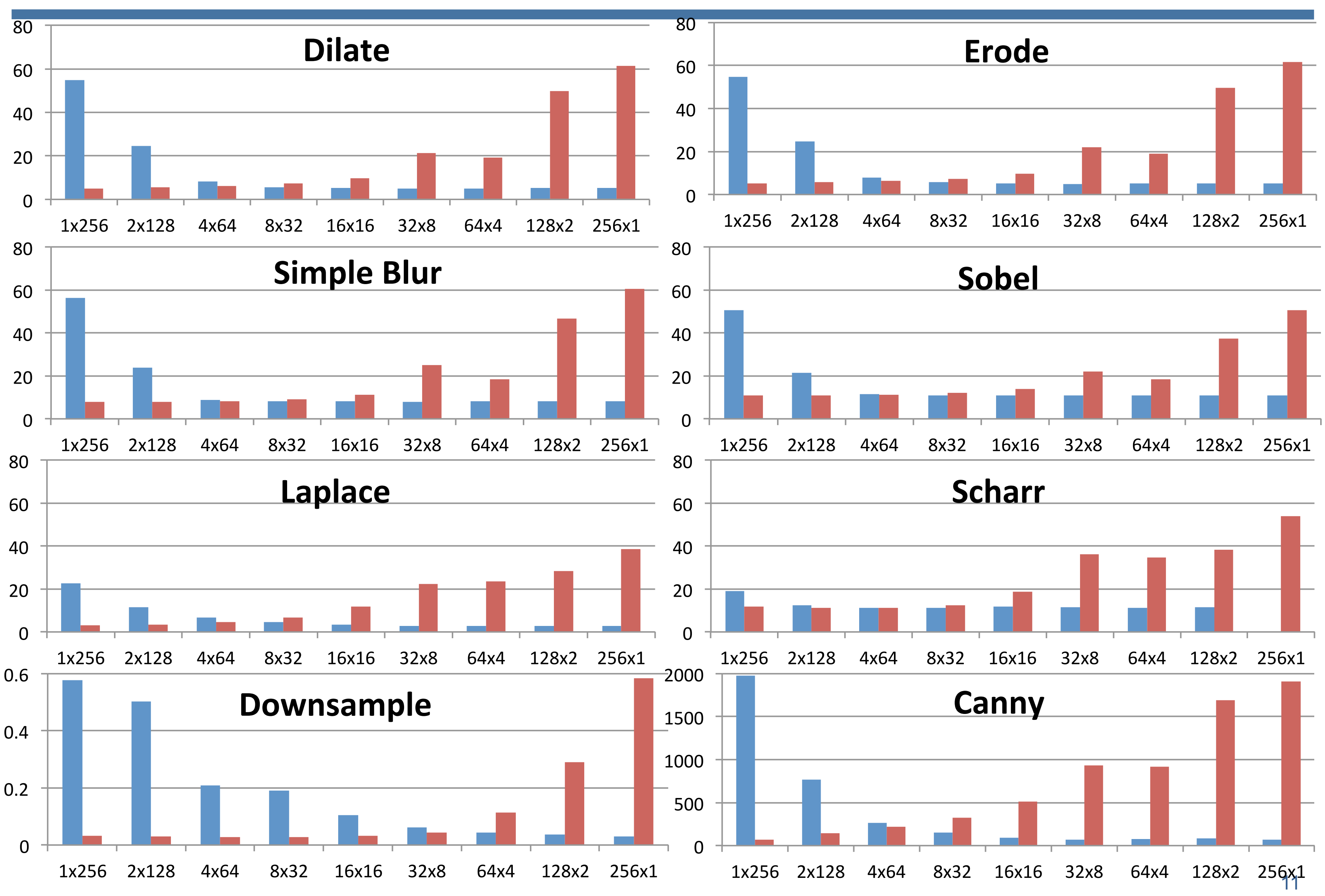
Launch Configuration

```
read image;
:
#pragma kernel main tblock(BY,BX) thread(TY,TX)
#pragma loop_partition over_tblock over_thread
for (row = 0; row < img_rows; ++row)
#pragma loop_partition over_tblock over_thread
    for (col = 0; col < img_cols; ++col)
    {
        read image pixels at and around (row,col);
        process image pixels;
        write image pixel at (row,col);
    }
}
:
write image;
```

Loop Order

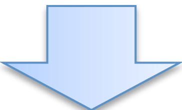
What should the loop order be given a launch configuration?

Performance Impact of Loop Order



Why Loop Order Matters?

Loop Order + Launch Configuration



Distribution of loop iterations among GPU threads

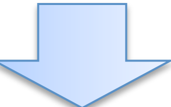
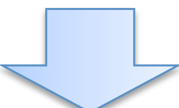
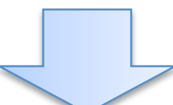


Image pixels accessed by concurrent GPU threads

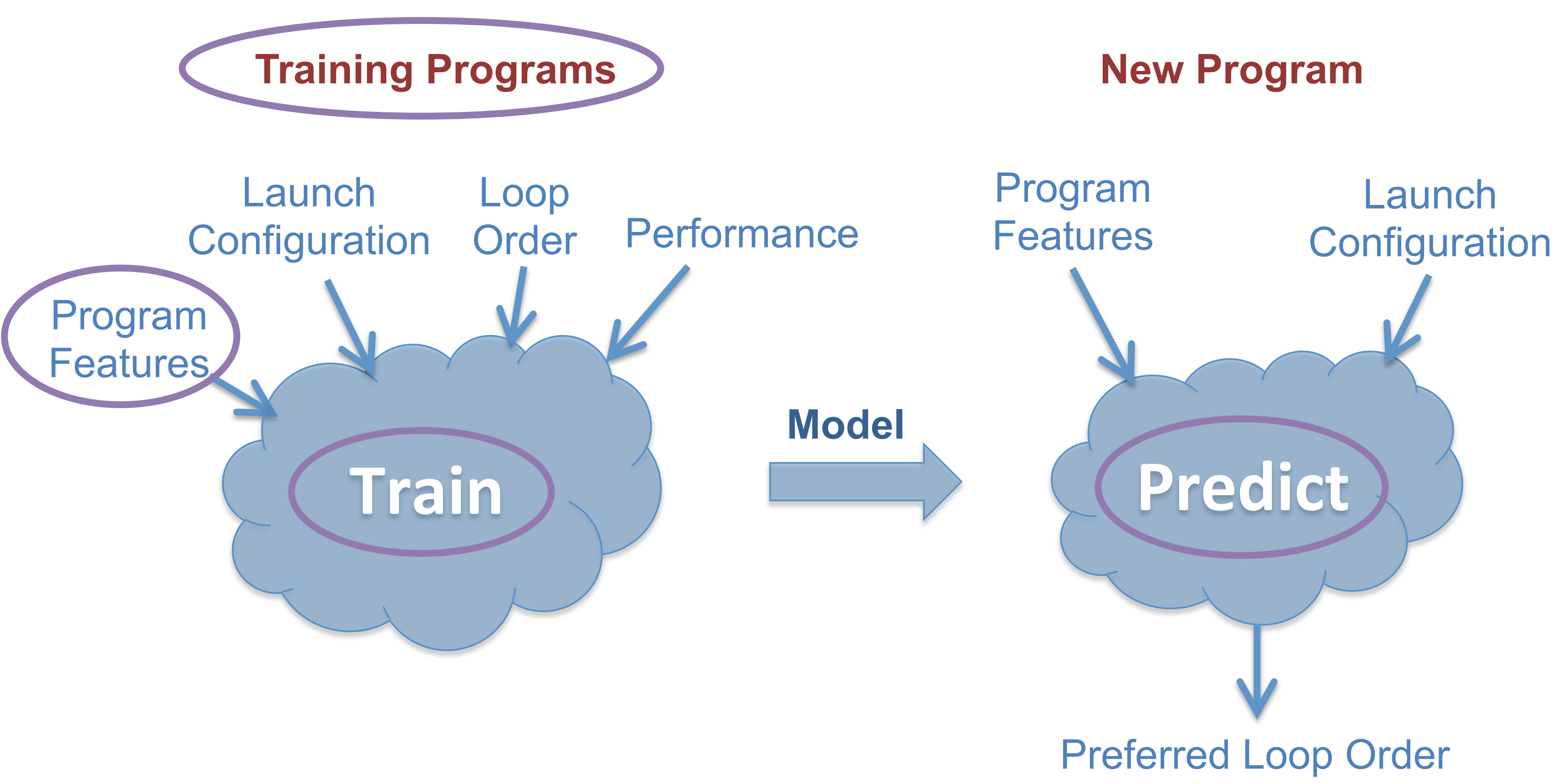


Degree of memory coalescing



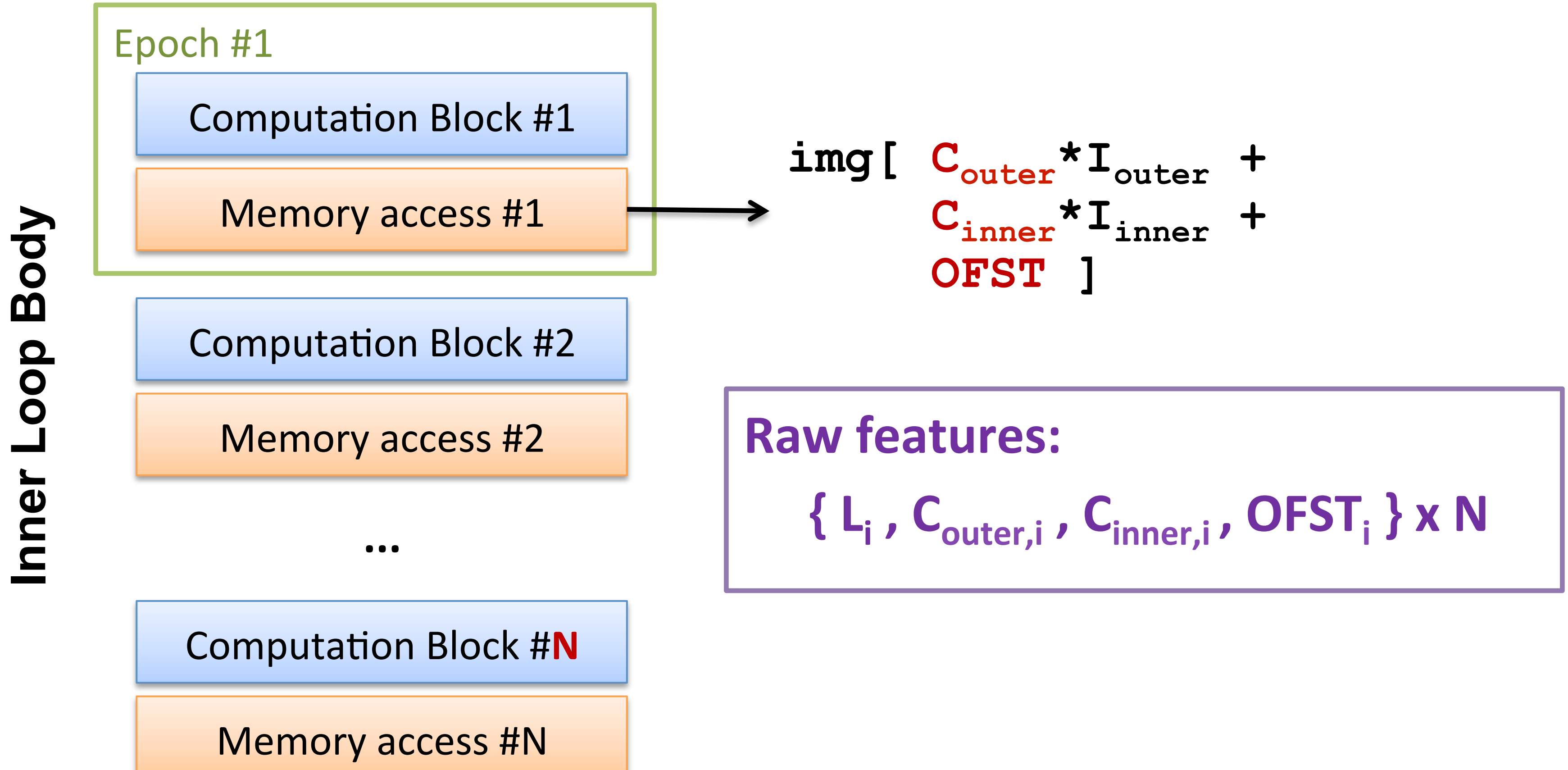
Kernel Performance

Learning Experiment Overview



Program Features

- What influences the preference of loop order?
  - Degree of coalescing of each memory access
  - Interleaved computation that hides access latencies



First Experiment: Raw

- Model inputs:
  - { L_i , $C_{outer,i}$, $C_{inner,i}$, $OFST_i$ } x N
  - Launch configuration: BX, BY, TX, TY
  - When loops interchanged, swap C_{outer} and C_{inner}

- Model output: kernel execution time

- Given a new kernel + a launch configuration
  - Use model to predict execution time with both loop orders
  - Choose the loop order that gives lower execution time

Experiment Setup (Raw)

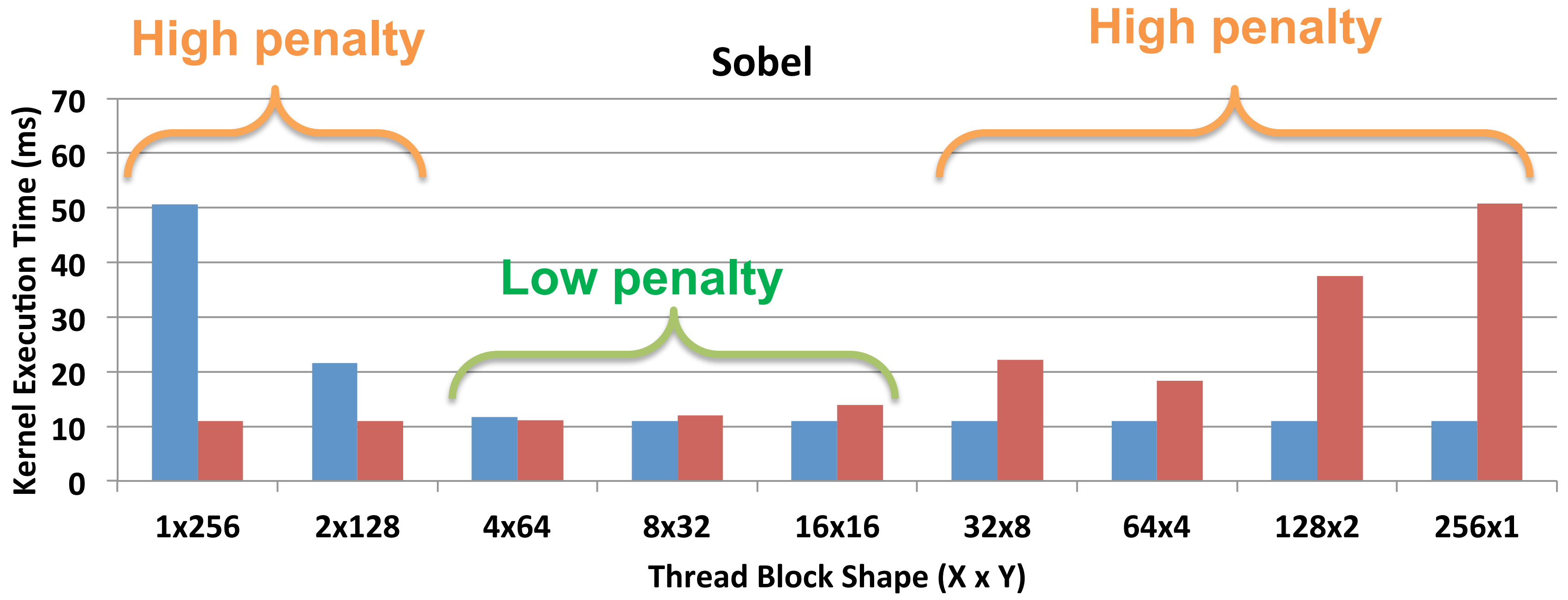
- Synthetically generated kernels
  - Each has two perfectly nested loops
  - … but differs in computation length and memory accesses in inner loop body
- Two kernel sets
  - K1: 4000 single-epoch kernels
  - K10: 4000 kernels with at most 10 epochs
- Collect execution time of each kernel with 3 launch configurations and both loop orders
  - On NVIDIA Tesla M2070
  - (TX, TY) = (32, 8), (8, 32), (2, 128); (BX, BY) = (360, 1)

Experiment Setup (Raw)

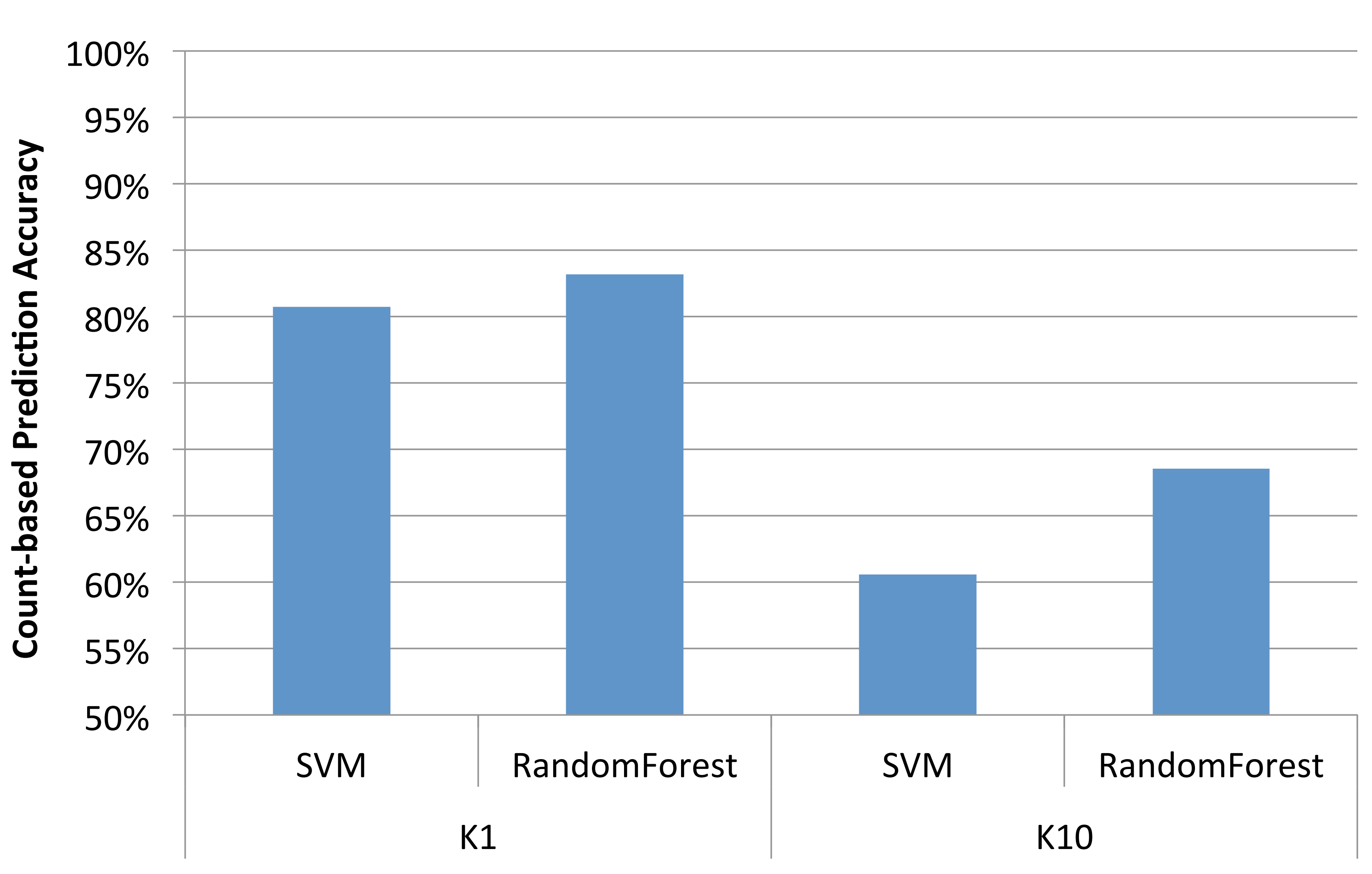
- Two ML algorithms (regression)
  - SVMLight (default over-fitting parameter, Gaussian kernels)
  - Waffles RandomForest (160 trees, 4 attributes per tree)
- For each kernel set, train on a random 1000 kernels and test on the remaining 3000 kernels
- How to evaluate prediction accuracy on the test set?

Evaluation Metric

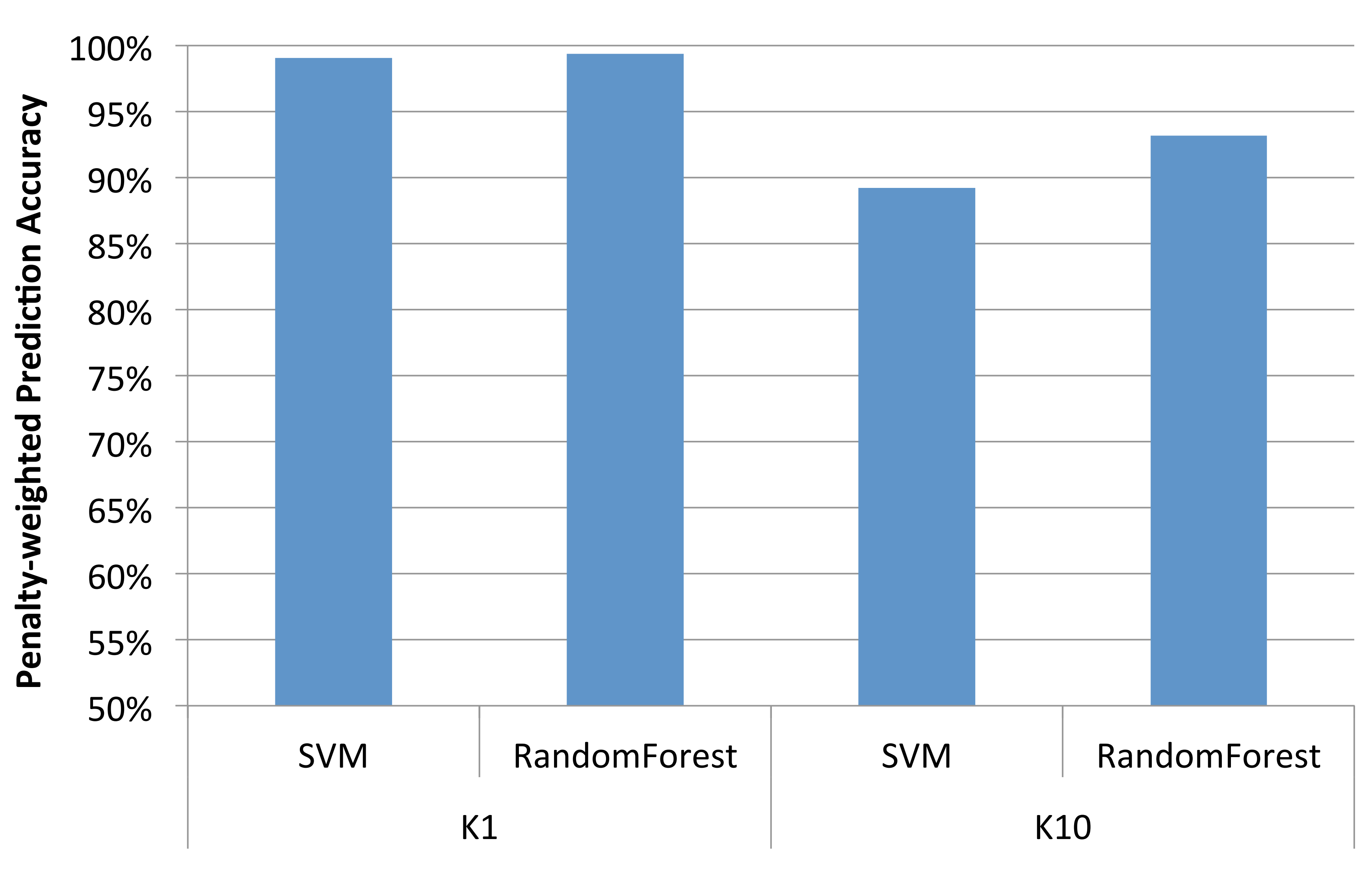
- Count-based Prediction Accuracy
  - % of test samples where the predicted loop order does give better kernel performance
- Penalty-weighted Prediction Accuracy
  - % of best performance achieved by predicted loop order



Experiment Result (Raw)



Experiment Result (Raw)



Dealing with Large Program Space

- Program space is inherently large
- The number of input features in RAW grows with program length (# of epochs)
- Train one or more **smaller** models, each focusing on a short program segment
- Use **program structure** to link these models
  - e.g., execution time of a series of code segments is roughly the sum of per-segment execution times

Second Experiment: Raw-S

Train on single-epoch kernels

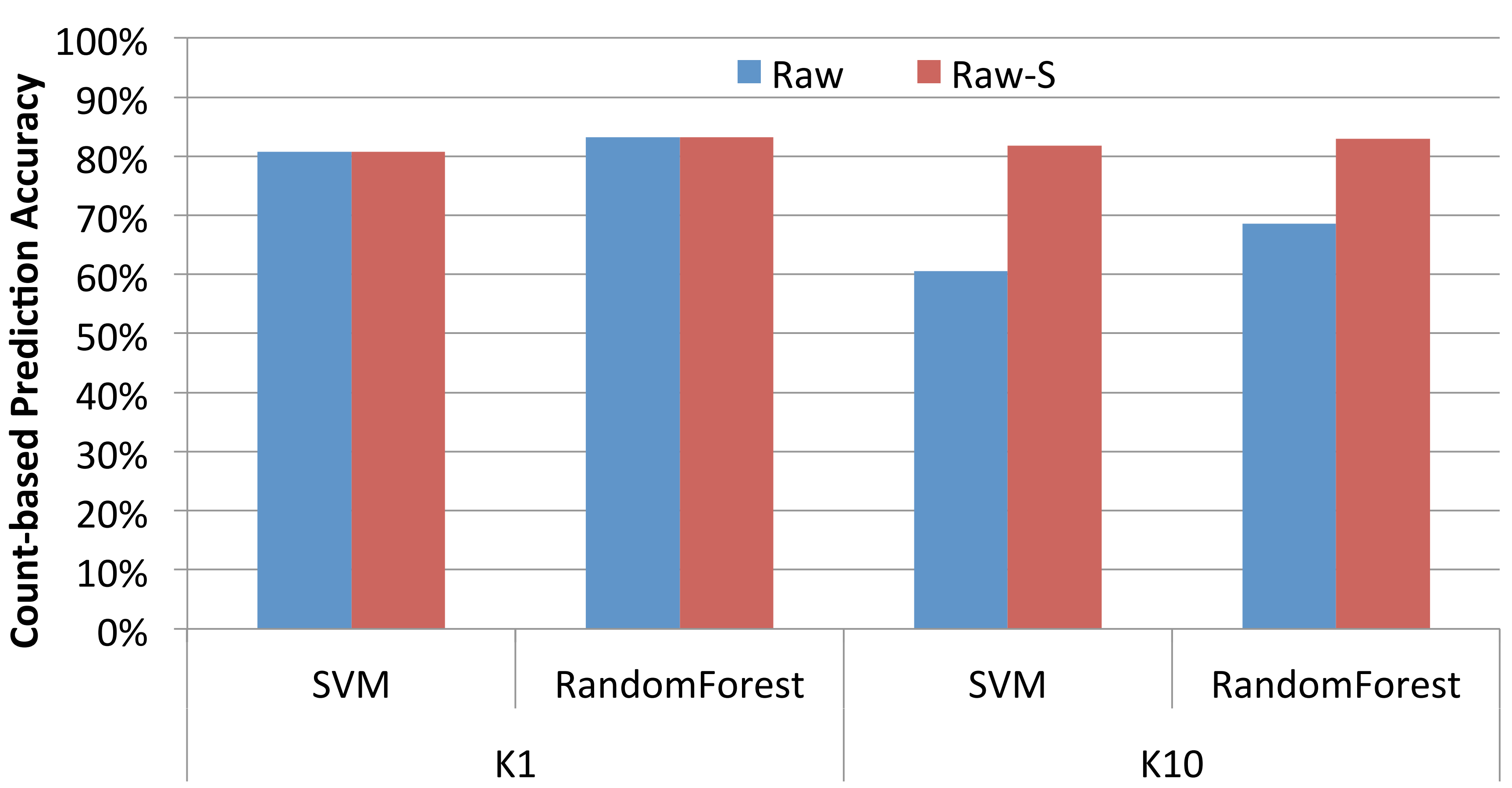
- Model inputs:
  - **{ L_i , $C_{outer,i}$, $C_{inner,i}$, $OFST_i$ }**
  - Launch configuration: BX, BY, TX, TY
  - When loops interchanged, swap C_{outer} and C_{inner}
- Model output: kernel execution time

Given a new ***K***-epoch kernel + a launch configuration

- Predict execution time with each loop order, by
  - Using the model (***K*** times) to predict execution time of each epoch
  - Summing the per-epoch predicted time
- Choose the loop order that gives lower execution time

Experiment Setup / Result (Raw-S)

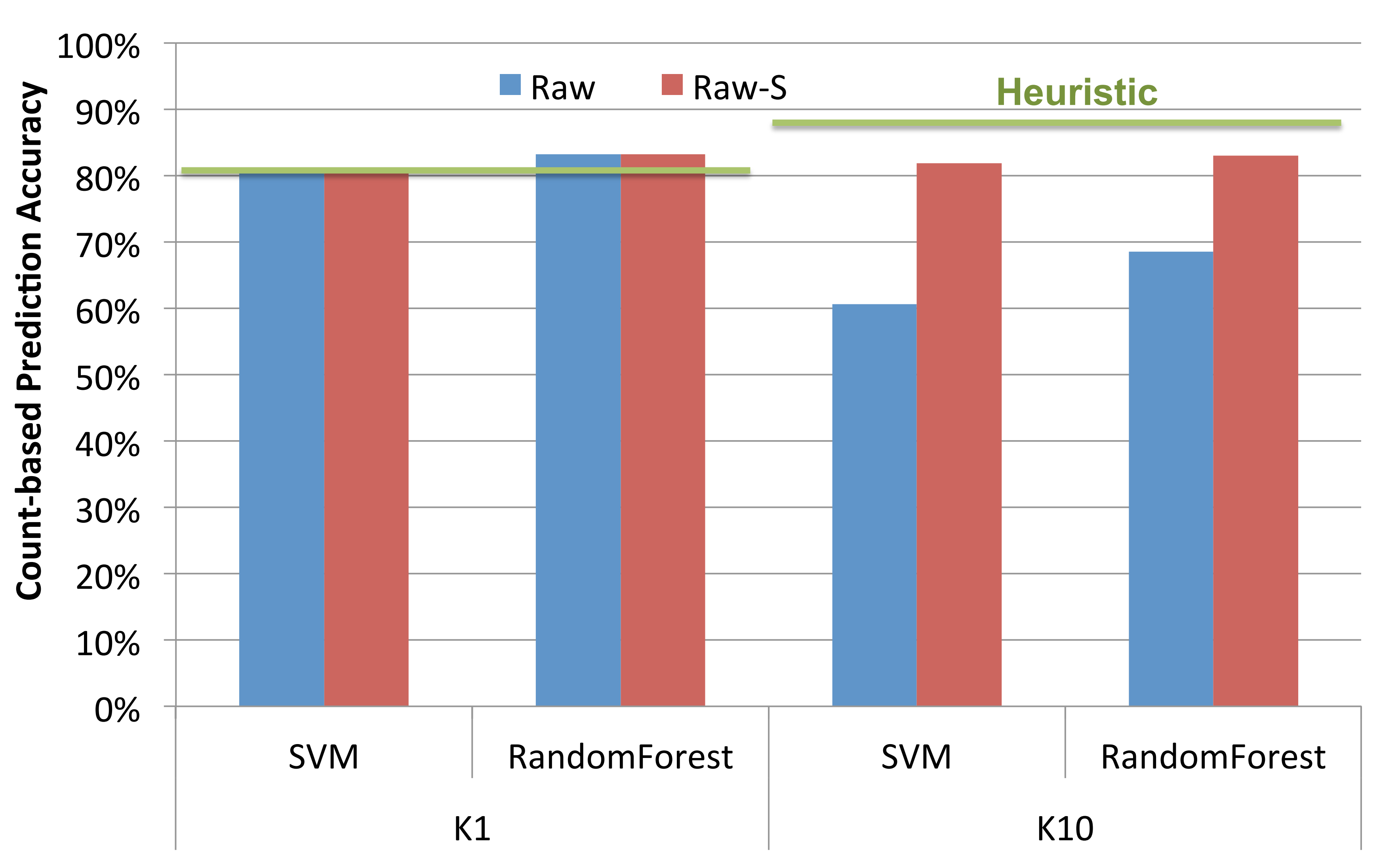
- Same two kernel sets K1 and K10
- Train on a random 1000 kernels in K1, and test on all kernels in K10



Applying GPU Expertise

- Loop order affects the degree of memory coalescing, thus kernel performance
- We can estimate # of DRAM transactions for each memory access in the inner loop body
  - Given the launch configuration
- A heuristic for kernel performance: total memory transactions from the inner loop body

Experiment Result (All)



What We Learnt

Machine learning can be a fast, accurate solution to auto-tuning,

If it is intelligently applied and integrated with our generic knowledge about programs

Beyond the "Toy" Problem

- Expand the optimization space to consider
- Currently working on a 7-D optimization space, with about ~50K valid configurations

- Two challenges:
  - Large program space
  - Large optimization space

- Collect kernel performance data for all configurations?
- Train a performance predictor?
- Compare ML performance against Oracle?

Thank You!

Questions?