Performance prediction on heterogeneous systems using statistical methods

Spyrou Michalis
University of Thessaly

Purpose of this thesis

- Predicting performance on heterogeneous systems
- Creating the prediction model
- Facilitating the mapping of applications on various computational resources

Our Approach

- Collection of hardware events
  - PAPI
  - Nvprof
- Use of Statistical methods
  - Linear Regression
  - Neural Networks
  - Random Forests

Background Issues

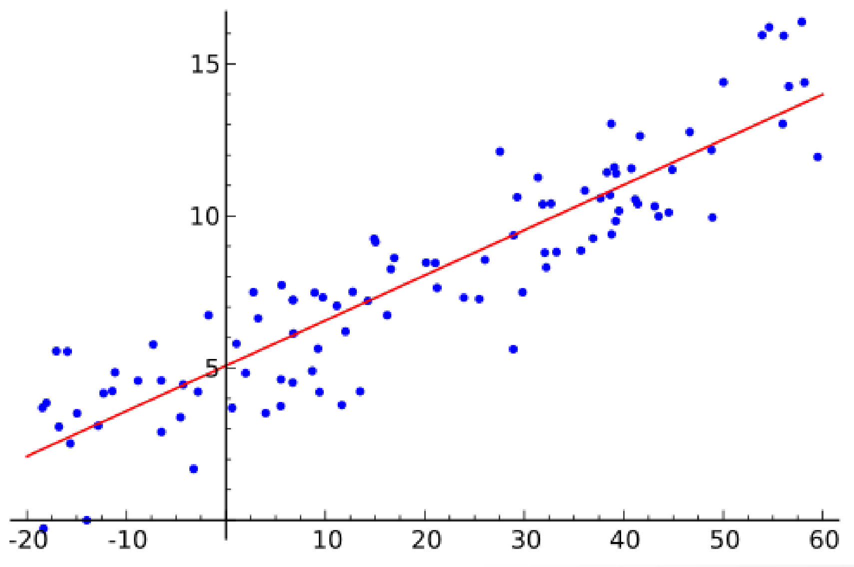
Architectural differences

- CPUs:
  - few processors supporting ~1 hardware thread
  - execute one stream of instructions as fast as possible
  - latency : large caches and branch prediction hardware
  - a few hundreds GFLOPS

- GPUs:
  - many processors supporting many hardware threads
  - execute many parallel streams of instructions as fast as possible
  - latency : supporting thousands of threads at once
  - thousands of GFLOPS

Linear Regression

- Linear relationship
- Equation form: **y = Xβ + ε**

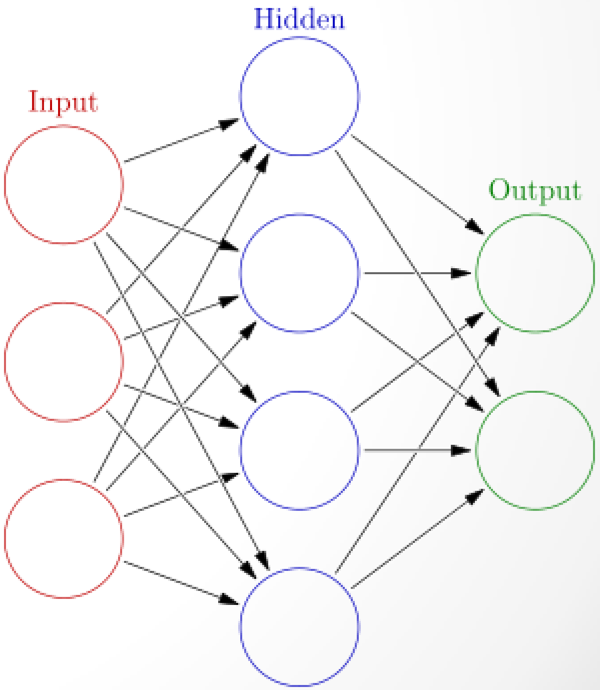


Neural Networks

- Interconnected **nodes**, forming **layers**
- Each node performs a mathematical function, such as

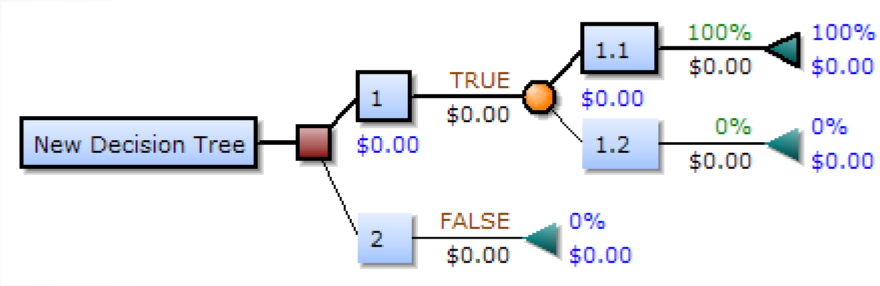
$$\boldsymbol{S(t) = \frac{1}{(1+e^{-t})}}$$

- **Black box**



Random Forests

- A combination of **decision trees**



- Node → **Test** on an attribute
- Branch → **Result** of a test
- Leaf → **Decision** taken

Data Collection

Benchmarks

- **Rodinia** benchmark suite 2.4
- The Scalable HeterOgeneous Computing (**SHOC**) Benchmark Suite

Intel Xeon CPUs

- Use of **OpenCL**
- Get number of compute units
- Use of low level **PAPI** Interface for events
- OpenCL built in functions for time

Event measurement example

```
clFinish(command_queue);

start_cpu_counter ( counter, &EventSet,
                    proccess_threads, compute_units);

error = clEnqueueNDRangeKernel( command_queue, kernel, 1,
NULL,global_work_size, local_work_size, 0, NULL, NULL);

clFinish(command_queue);

stop_cpu_counter ( EventSet, &results, compute_units);
```

NVIDIA GPUs

- **NVprof**: not supporting OpenCL
- Used CUDA
- CL Interface

Model Formation

- Find **correlation** between target value and measured variables

Intel to NVIDIA		NVIDIA to Intel	
Counter	Correlation	Counter	Correlation
BR_TKN	0,8874	ldst_executed	0,88
TOT_CYC	0,868	inst_issued	0,88
BR_PRC	0,862	issue_slots	0,88
TOT_INS	0,861	inst_executed	0,86
LD_INS	0,8553	ldst_issued	0,86
BR_CN	0,8543	cf_issued	0,85
LST_INS	0,8503	cf_executed	0,85
BR_INS	0,8473	l2_read_trans	0,83

Predicting OpenCL CUDA execution time

- Linear regression

- Tested different number of hardware events. Calculated **RMSE**

Number of Events Used	RMSE
9	5,12
8	7,29
7	7,17
6	8,86
5	229

Predicting OpenCL CUDA execution time

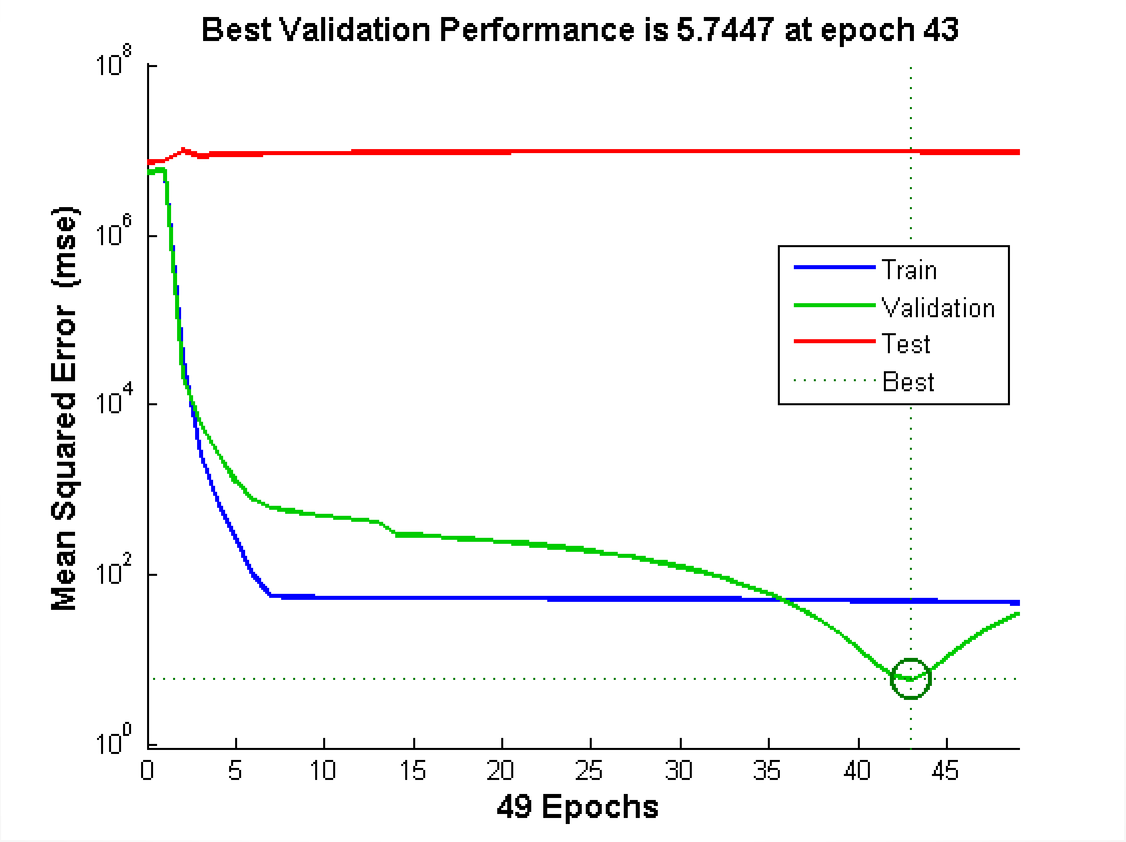
- Linear regression formula:

$$Execution\ time =$$

$$\begin{bmatrix} -2.9\times10^{-2} & -1.24\times10^{-3} & -1.12 & 7.865\times10^{-3} & -4.81\times10^{-3} & 1.08 \end{bmatrix} \times \begin{bmatrix} BR\_TKN \\ TOT\_CYC \\ BR\_PRC \\ TOT\_INS \\ LD\_INS \\ BR\_CN \end{bmatrix} \times 10^{-6} + 2.0829$$

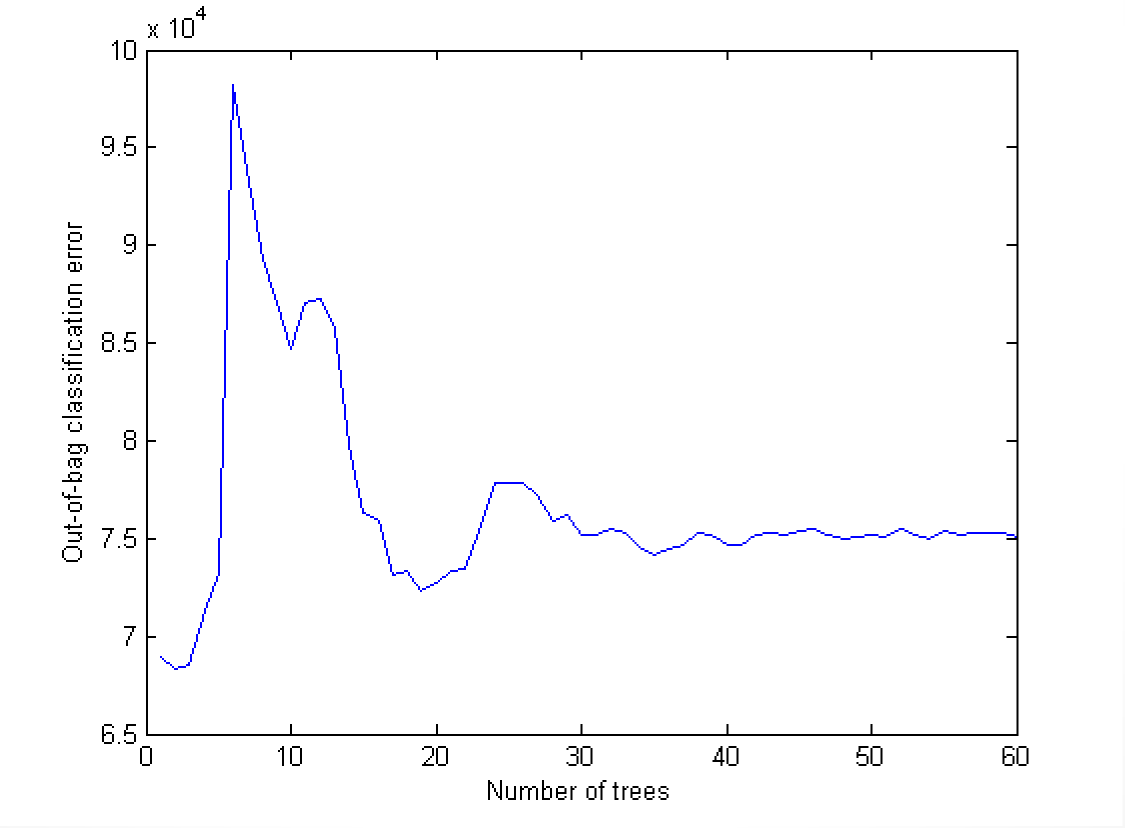
Predicting OpenCL CUDA execution time

- Neural Networks



Predicting OpenCL CUDA execution time

- Random forests



Predicting Intel OpenCL execution time

- Linear regression

Number of Events Used	RMSE
10	92,6
9	99,1
8	110
7	110
6	109
5	115

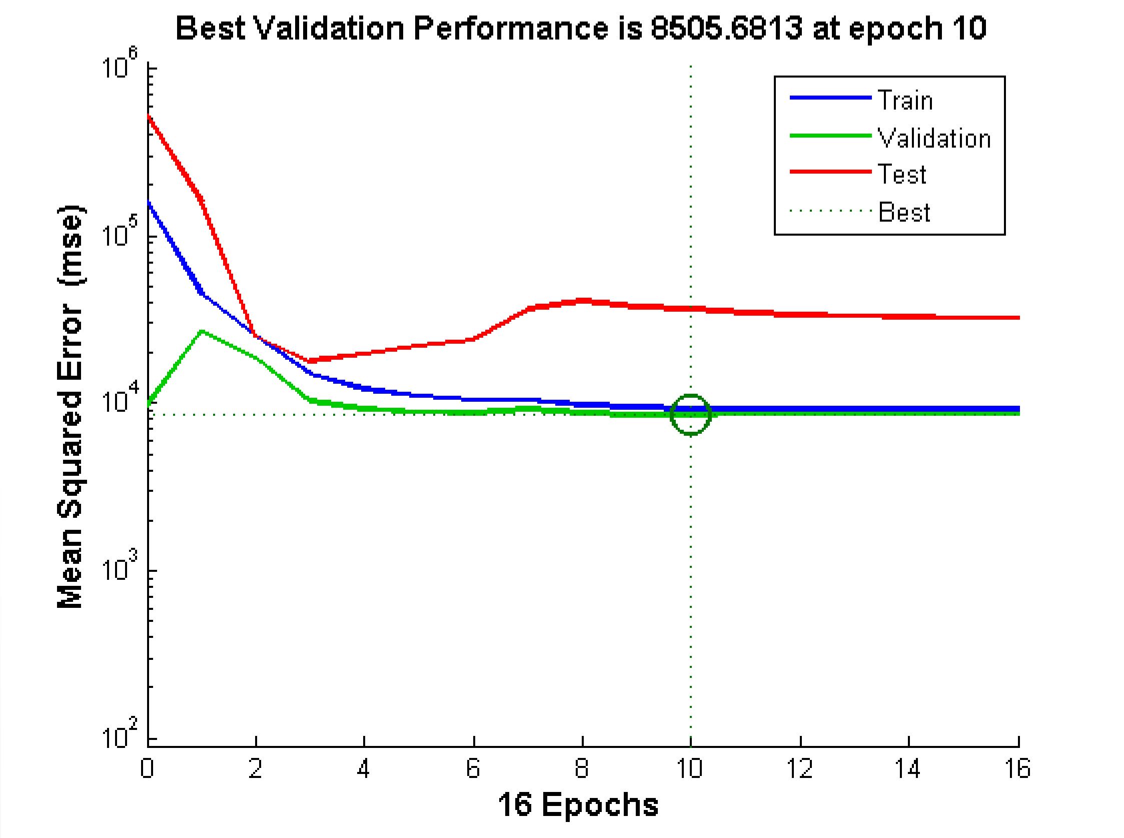
Predicting Intel OpenCL execution time

- Linear regression formula:

$$Execution\ time = \begin{bmatrix} -2.12 & 65.06 & -65.9 & 1.07 & 0.4 & 7.96 & -7.89 & 0.37 & -35.8 & 43.4 \end{bmatrix} \times \begin{bmatrix} ldst\_executed \\ inst\_issued \\ issue\_slots \\ inst\_executed \\ ldst\_issued \\ cf\_issued \\ cf\_executed \\ l2\_read\_trans \\ l2\_write\_trans \\ dram\_write\_trans \end{bmatrix} \times 10^{-5} + 14.791$$

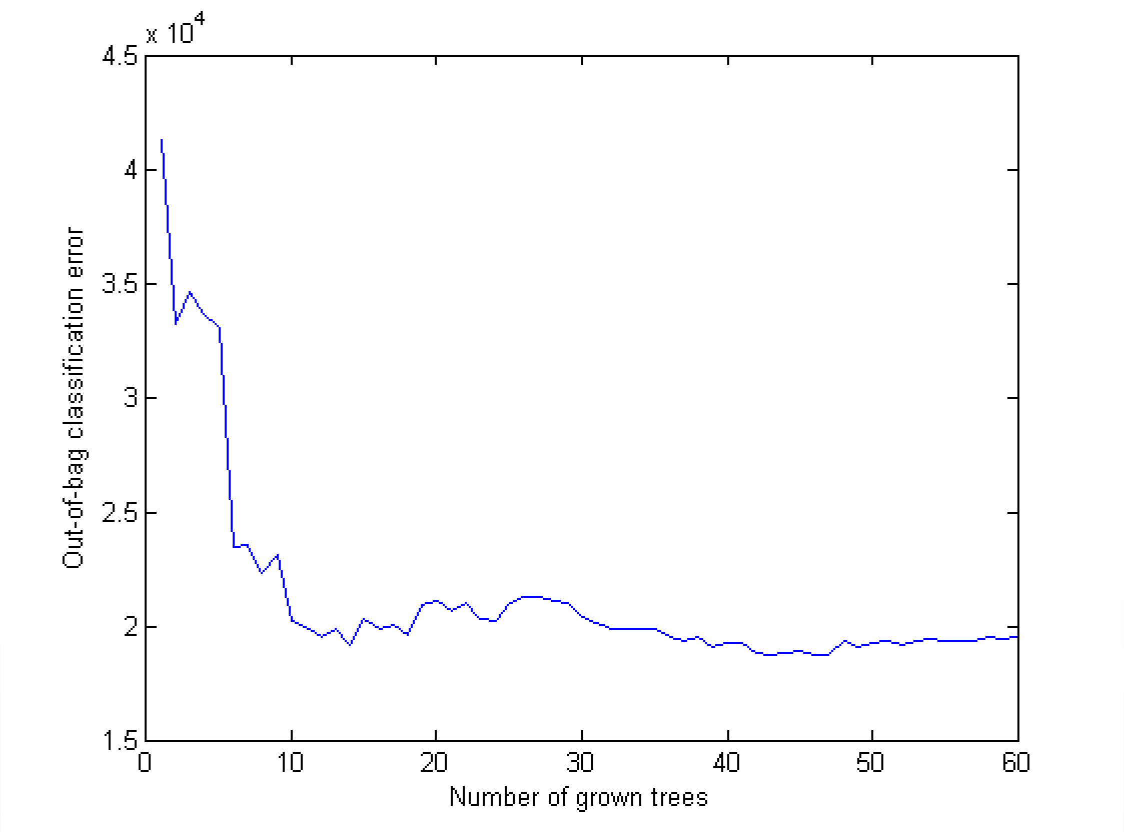
Predicting Intel OpenCL execution time

- Neural Networks



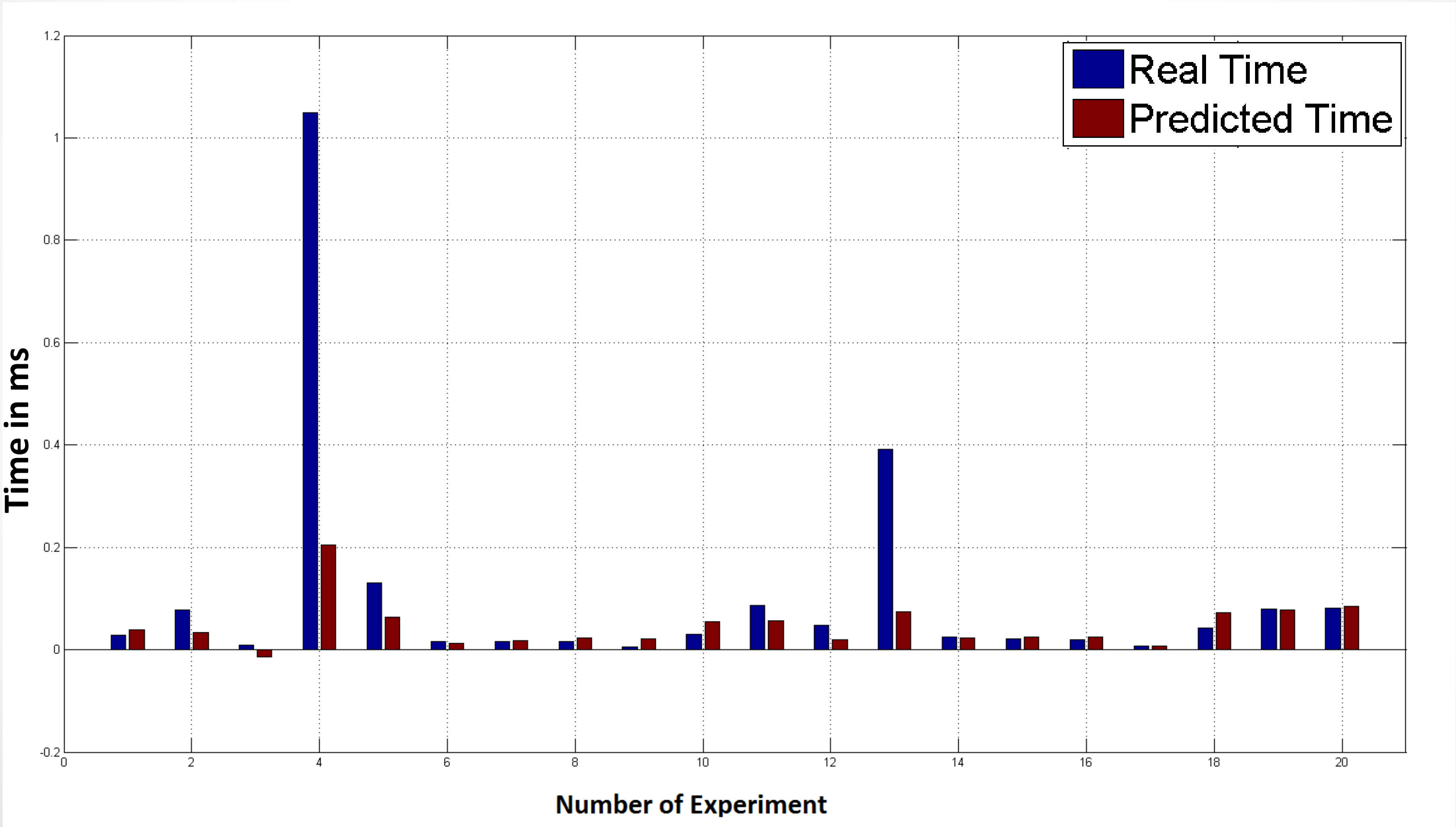
Predicting Intel OpenCL execution time

- Random Forests

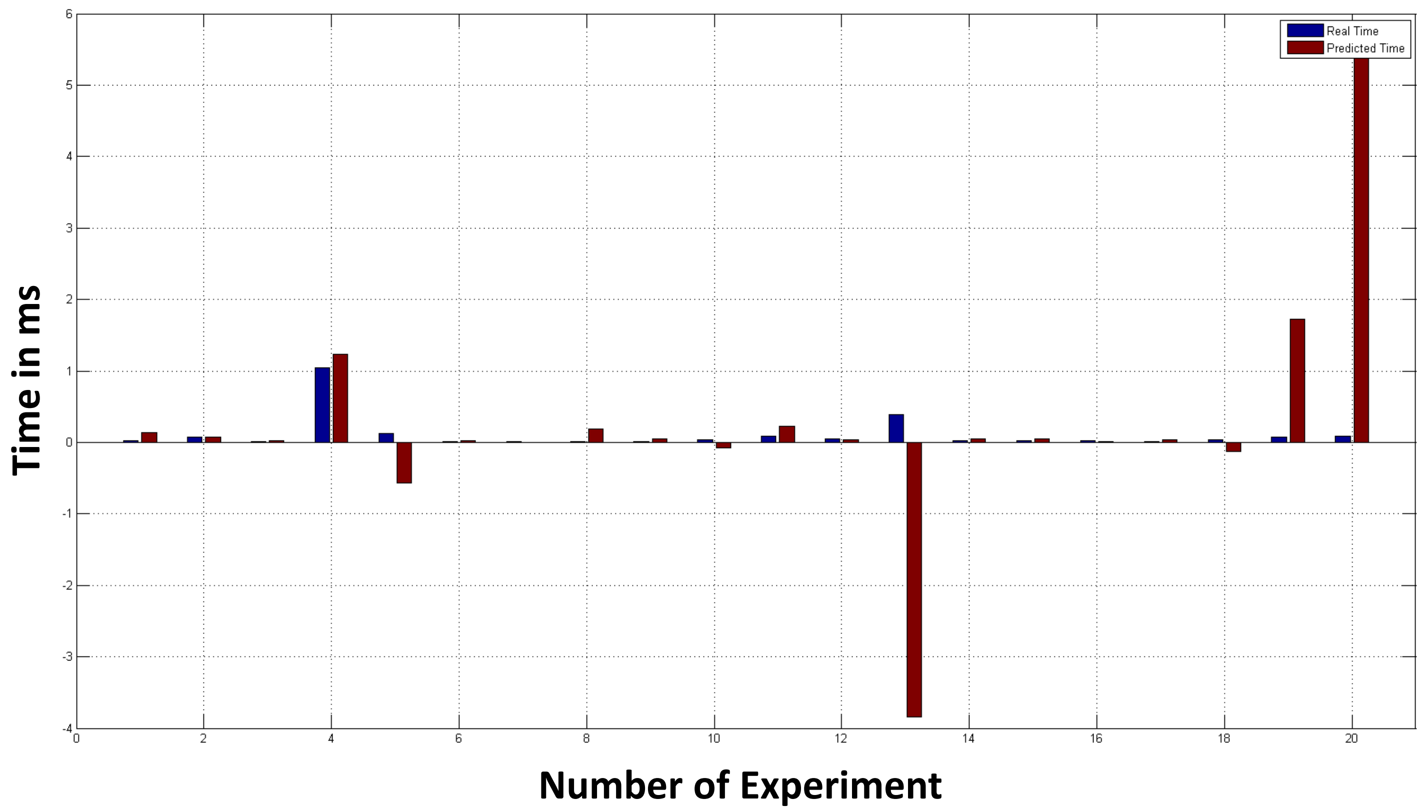


Evaluation

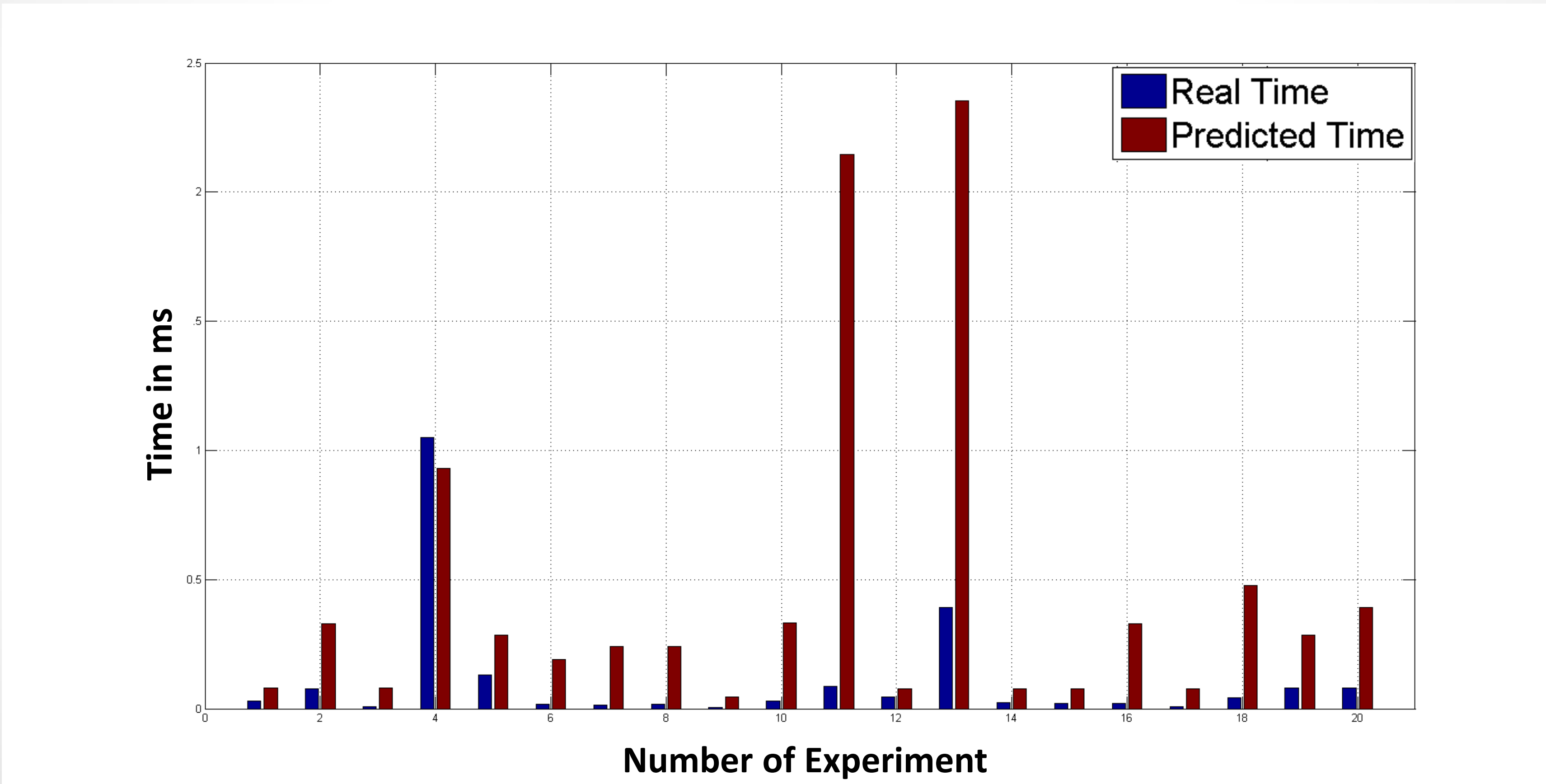
Intel Xeon OpenCL to GTX480 CUDA - Linear



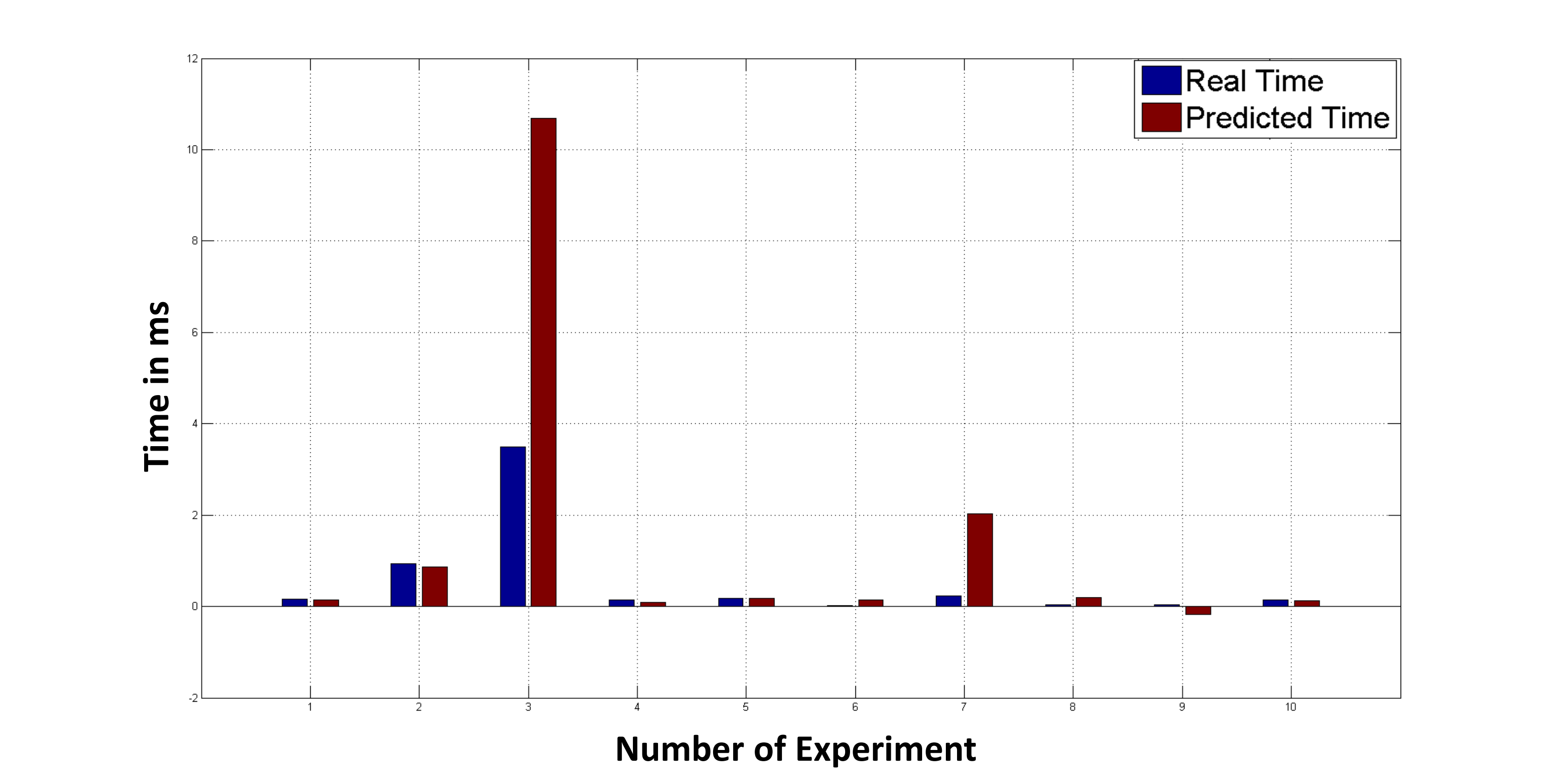
Intel Xeon OpenCL to GTX480 CUDA - NN



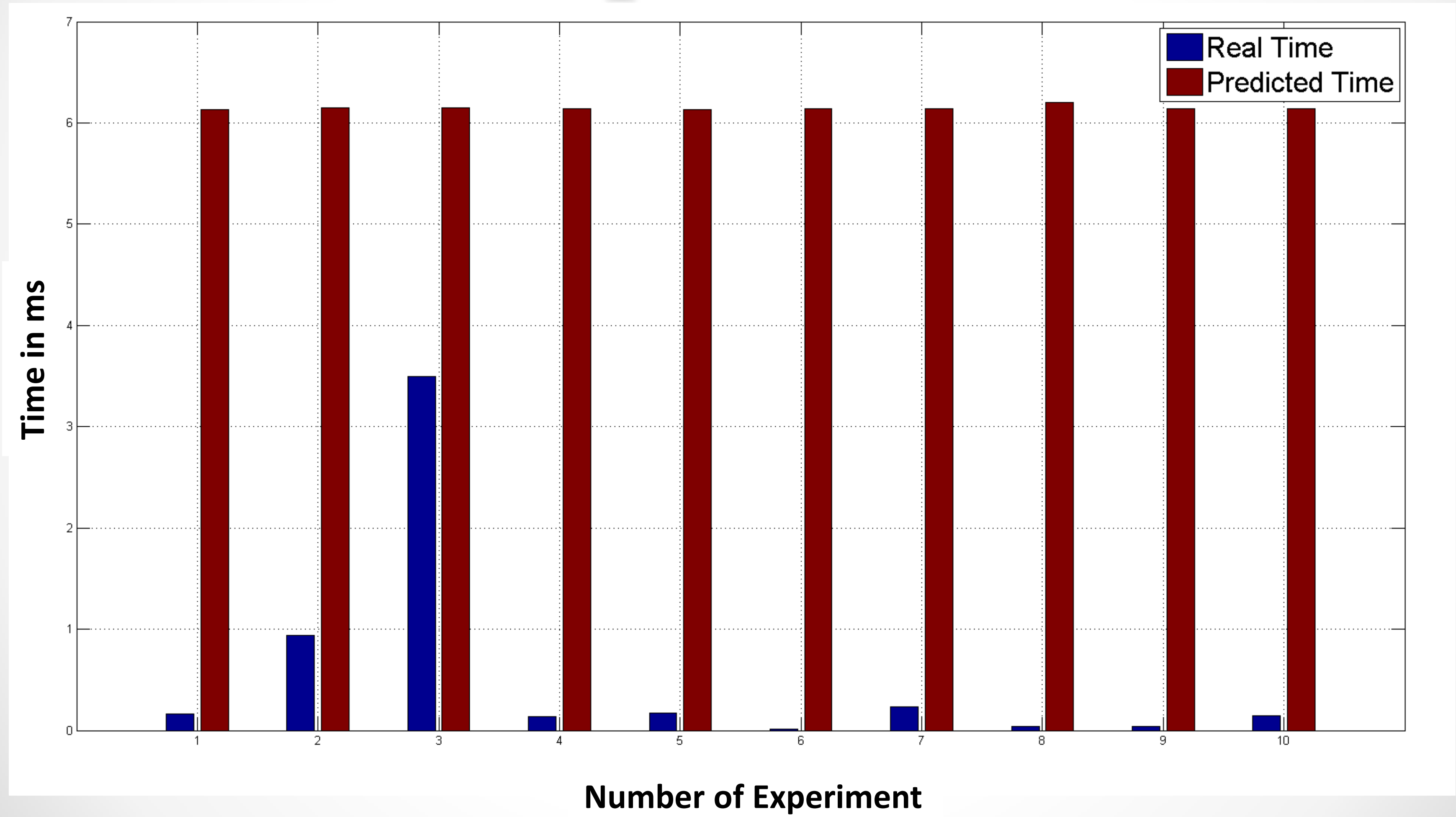
Intel Xeon OpenCL to GTX480 CUDA - RF



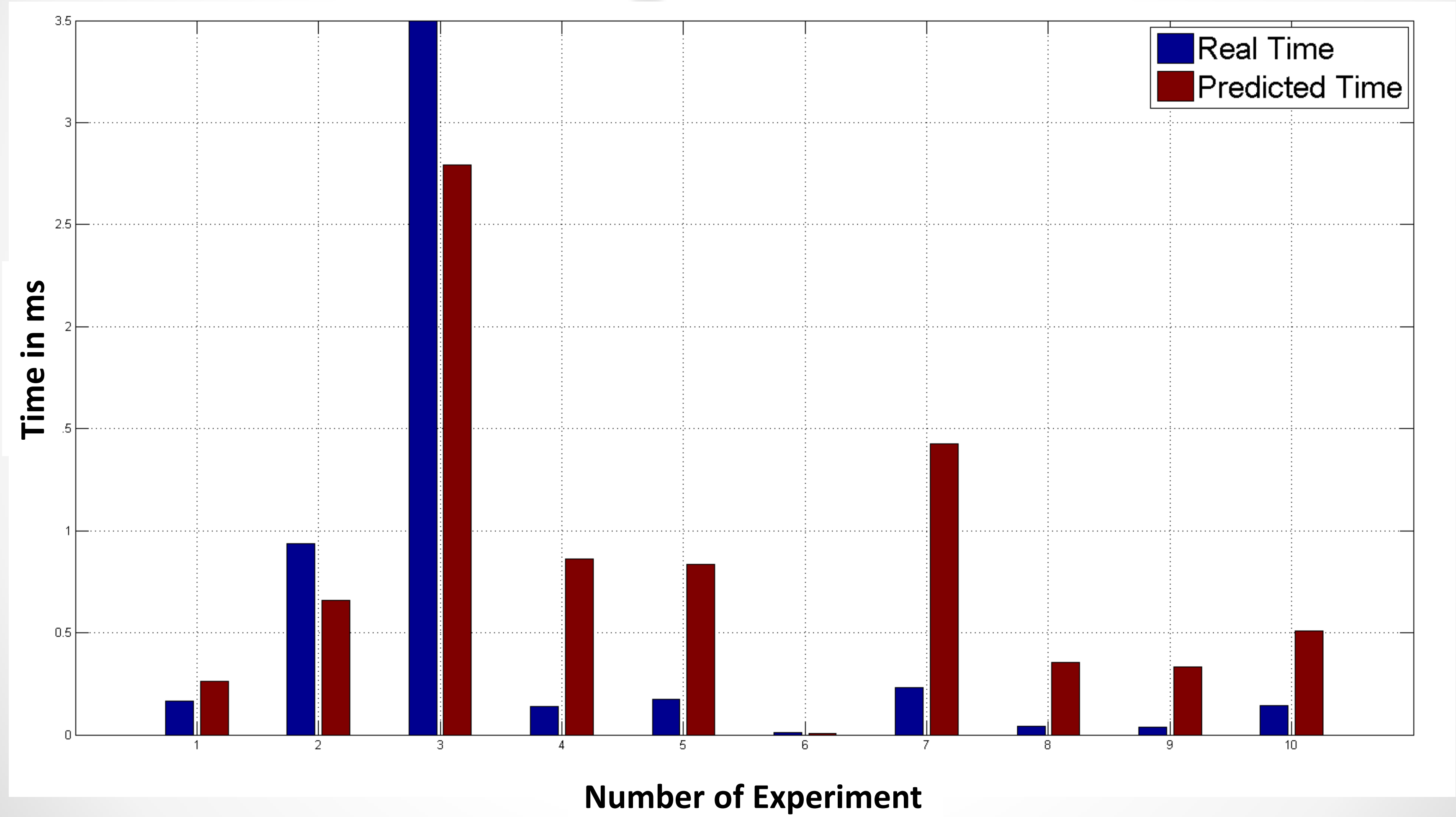
GTX480 CUDA to Intel Xeon OpenCL - Linear



GTX480 CUDA to Intel Xeon OpenCL - NN



GTX480 CUDA to Intel Xeon OpenCL - RF



Future work

- Predict execution time on ATI GPUs
- Increase size of training set
- Predict power
- Use our models with a run-time system

Questions