AeminiumGPU: An Intelligent Framework for GPU Programming

Alcides Fonseca, Bruno Cabral
{amaf,bcabral}@dei.uc.pt
Universidade de Coimbra, Portugal

ÆminiumGPU

- A component of the larger ecosystem of the Æminium Programming Language (http://aeminium.dei.uc.pt)

- Focus on taking advantage of the GPU to accelerate data-parallel operations. Task parallelization is done at the CPU level.

- Supports Map and Reduce, but more to come.

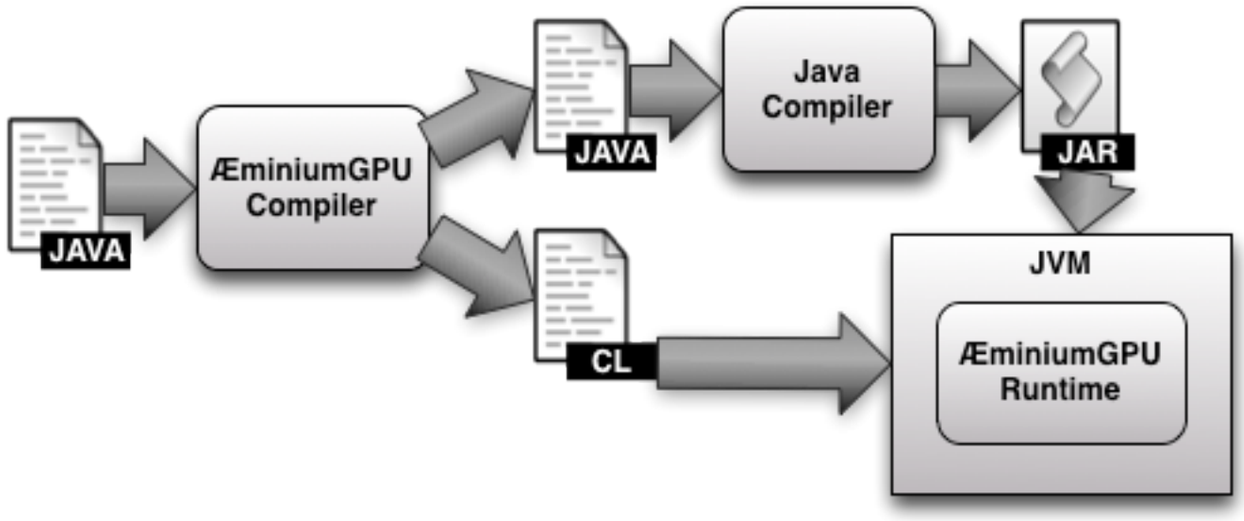
- Compiles from Java to Java with OpenCL.

- Kernels are only compiled at Runtime.

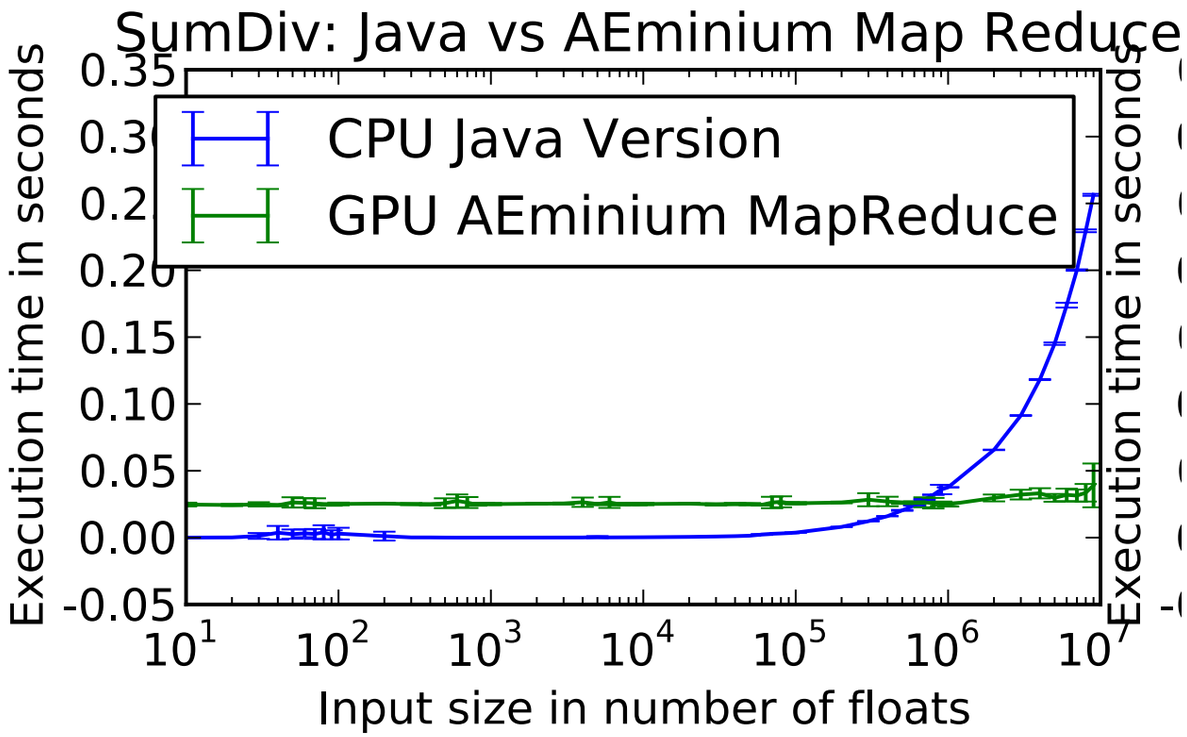
- Programs can execute on either CPU or GPU

ÆminiumGPU

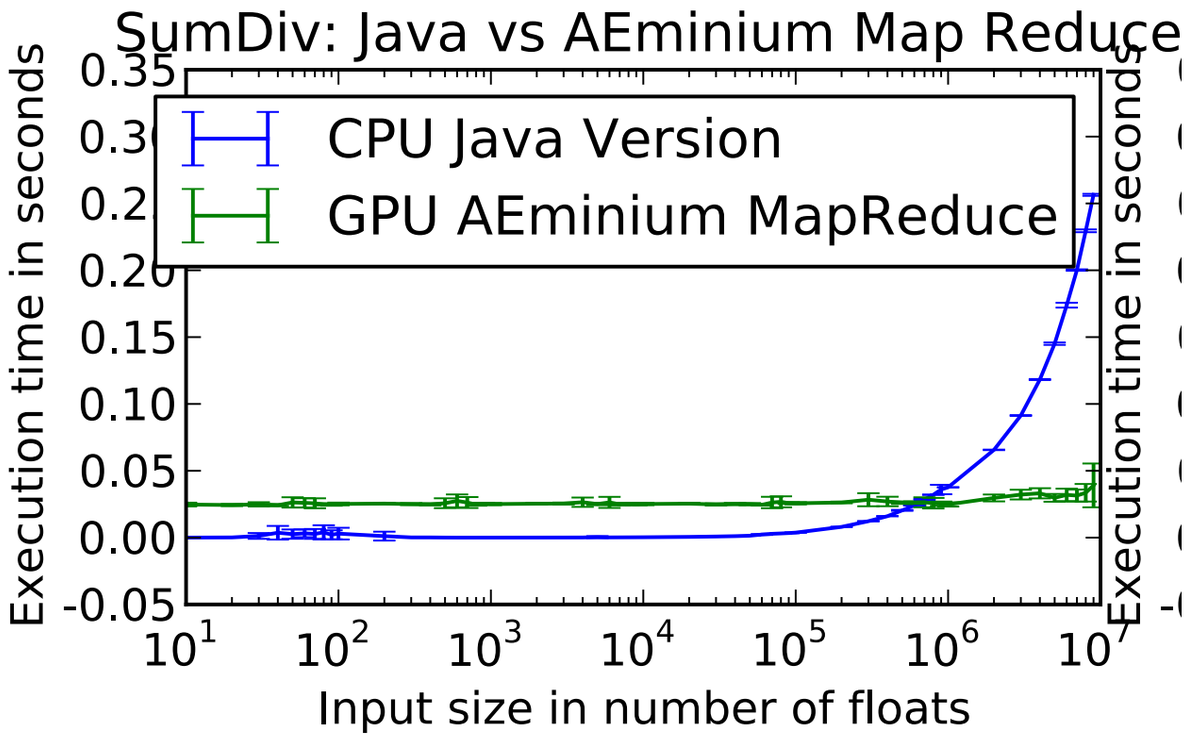
```
double pi = new RandomMatrix(N,2).map(x,y ->
(x*x + y*y < 1) ? 1 : 0).reduce(x,y -> x+y) * 4.0/N
```



Problem Definition



Problem Definition



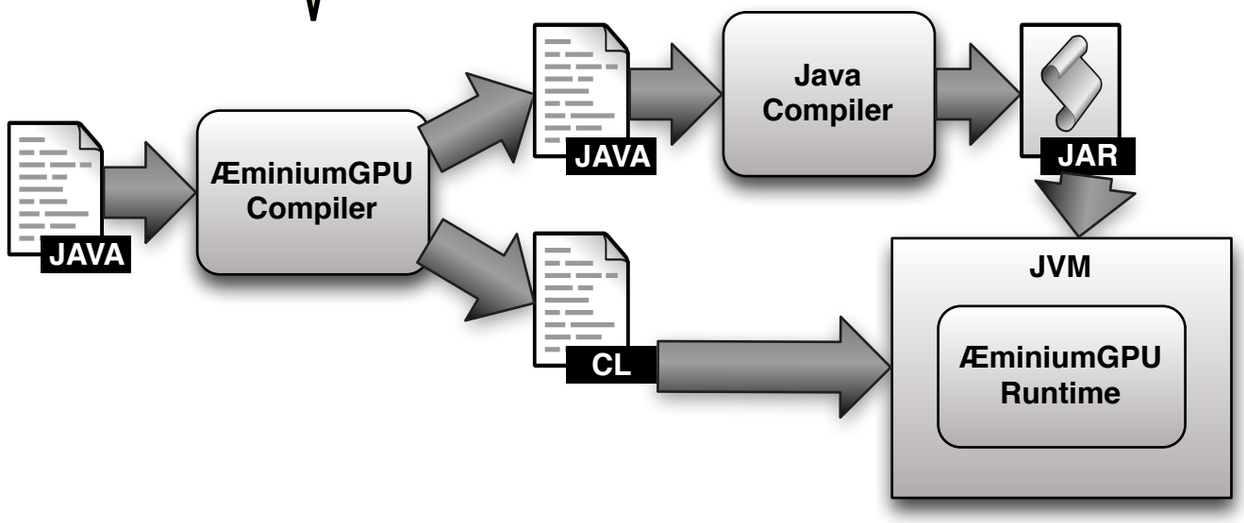
For a single program execution, should the GPU or the CPU be used?

Approach: Machine Learning

- From a benchmark suite, extract the most relevant features and the expected output.
- Build a classifier from the training data.
- Classify new program instances based on the previous data.

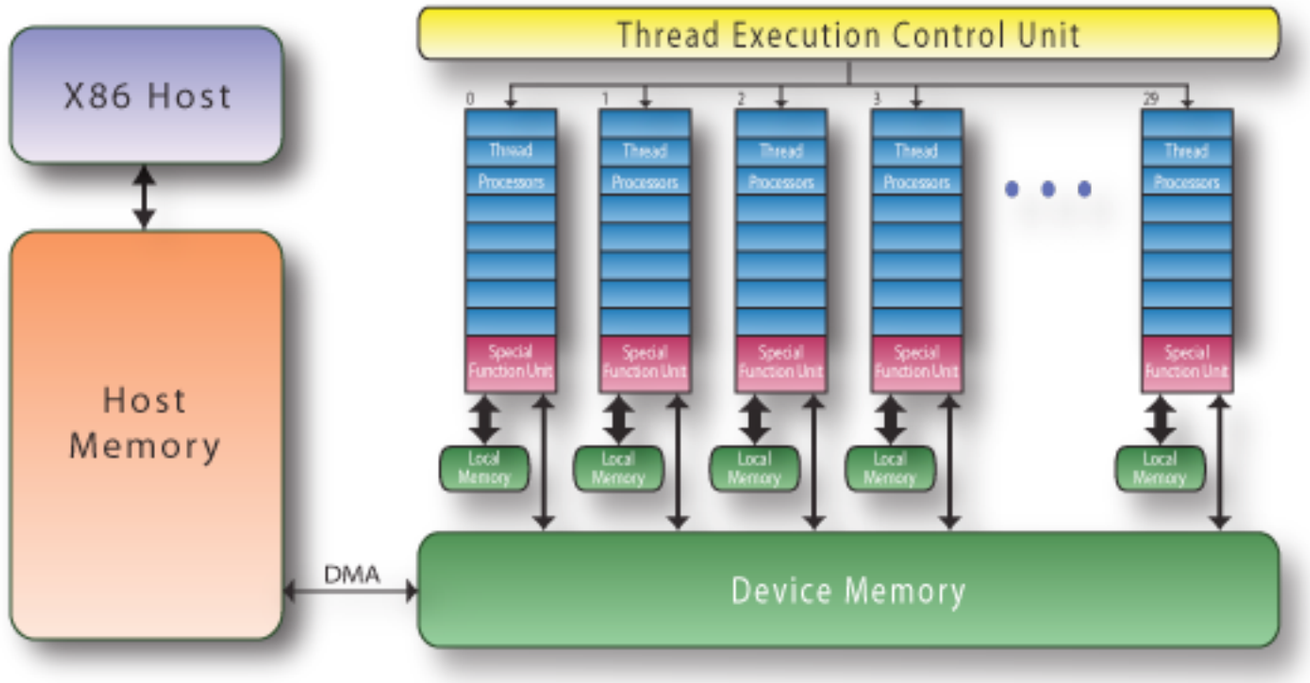
Approach: Machine Learning

Collect *C* features for all operations.



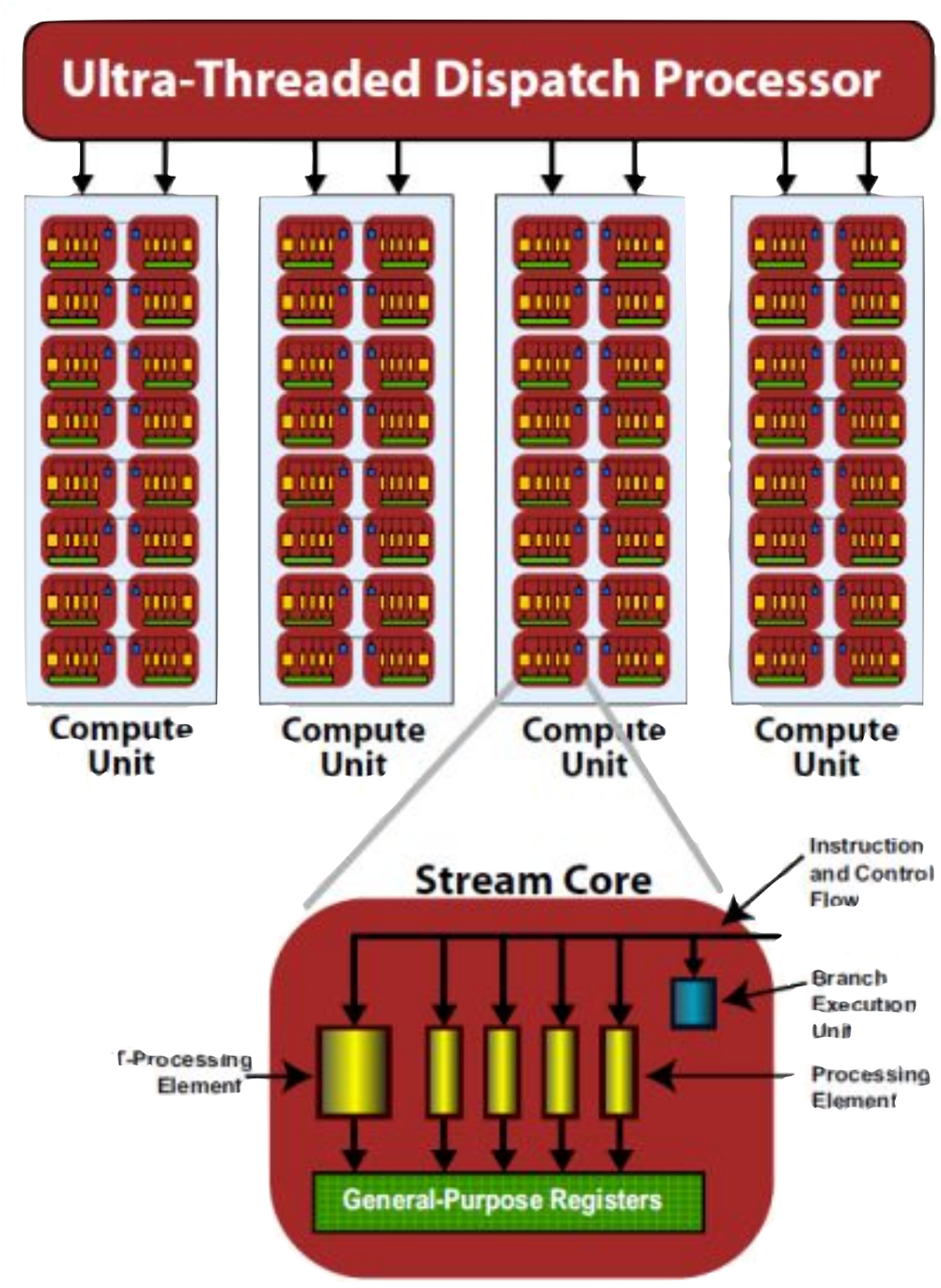
Before each operation,
a) Collect *R* features.
b) Merge *C* and *R*
c) Classify operation as either *Best-on-CPU* or *Best-on-GPU*

GPU Programming



[2]

GPU Programming II



[3]

```
if (condition) {
  doSomething()
} else {
  doSomethingElse()
}
```

Features

```
a(); // Level 1
for (int i=0; i<10; i++) {
    b(); // Level 2
    while ( j < 20) {
        c(); // Level 3
    }
}
```

Features

Name	Size	C/R	Description
OuterAccess	3	C	Global GPU memory read.
InnerAccess	3	C	Local (thread-group) memory read. This area of the memory is faster than the global one.
ConstantAccess	3	C	Constant (read-only) memory read. This memory is faster on some GPU models.
OuterWrite	3	C	Write in global memory.
InnerWrite	3	C	Write in local memory, which is also faster than in global.
BasicOps	3	C	Simplest and fastest instructions. Include arithmetic, logical and binary operators.
TrigFuns	3	C	Trigonometric functions, including *sin*, *cos*, *tan*, *asin*, *acos* and *atan*.
PowFuns	3	C	*pow*, *log* and *sqrt* functions
CmpFuns	3	C	*max* and *min* functions
Branches	3	C	Number of possible branching instructions such as *for*, *if* and *whiles*
DataTo	1	R	Size of input data transferred to the GPU in bytes.
DataFrom	1	R	Size of output data transferred from the GPU in bytes.
ProgType	1	R	One of the following values: Map, Reduce, PartialReduce or MapReduce, which are the different types of operations supported by ÆminiumGPU.

Classifiers

- Random
- AlwaysCPU
- AlwaysGPU
- NaiveBayes
- SVM (using SMO)
- MLP
- DecisionTable
- CSDT
- RegressionBased*

Evaluation Environment

- Intel Core2 Duo E8200
  - 2 cores at 2.66GHz
  - 4GB of RAM
- NVIDIA GeForce GTX 285
  - 240 CUDA cores
  - 1GB of Memory

Dataset

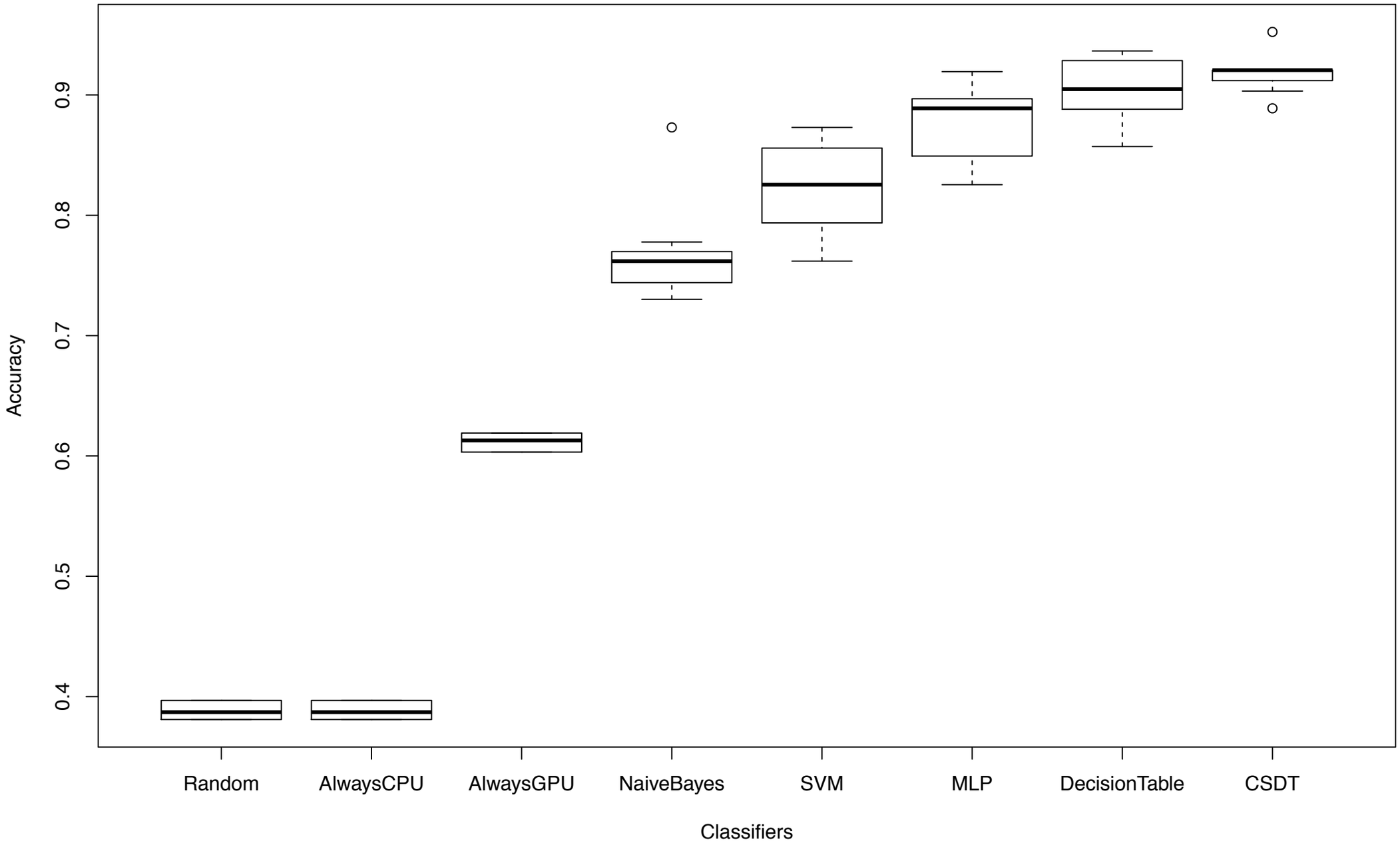
- 8 different programs
  - 4 simple Map-Reduce operations
  - 4 complex, mathematically intensive programs
- 55 different data sizes
  - {10, 20, 30...100} x 10^{1..6}
- 440 instances on the dataset.
- 7-fold cross validation on non-randomized dataset.

Feature analysis

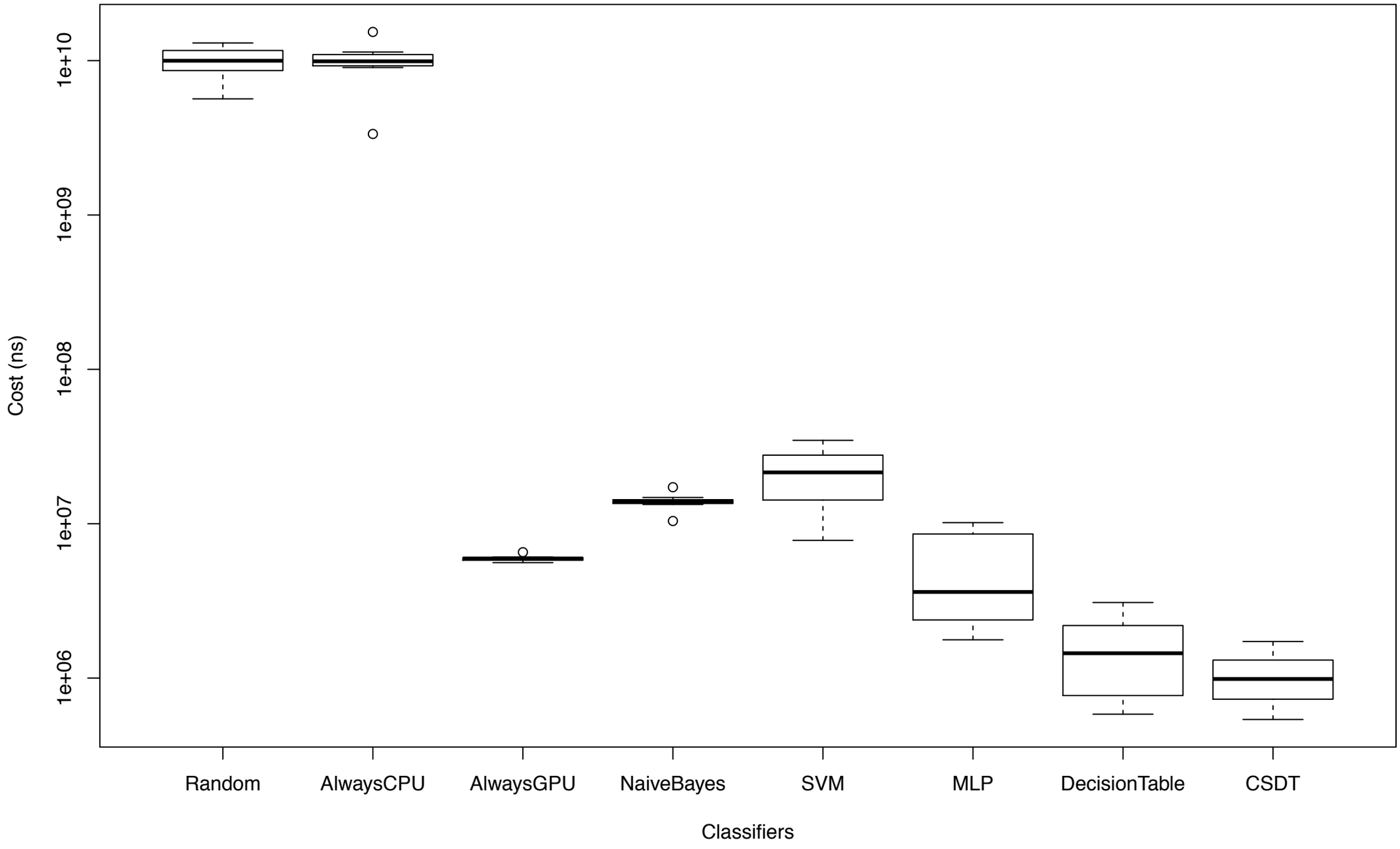
Using Information Gain (and Gain Value)

Rank	Feature
0.2606	DataTo
0.2517	DataFrom
0.1988	BasicOps2
0.1978	BasicOps1
0.1978	ProgType
0.1978	OutterWrite1
0.172	OutterAccess1
0.0637	Branches1
0.0516	InnerAccess1
0.0425	TrigFuns1
0.0397	InnerWrite2
0.0397	InnerAccess2

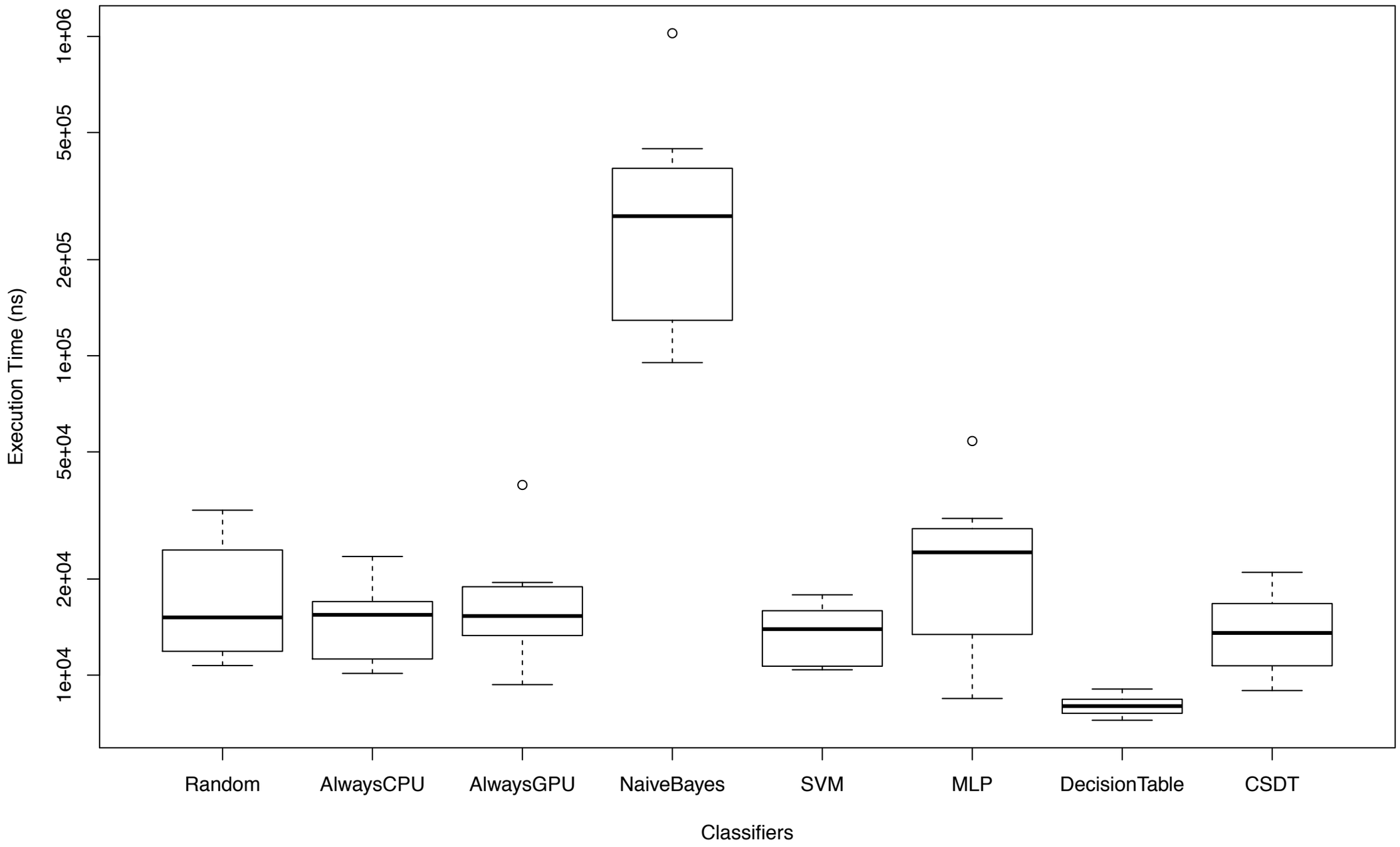
Classifier Comparison



Cost of Classification



Classification time



Future Work

- Larger Dataset
- Hybrid GPU/CPU computations
- Cost-per-miss-sensitive classifier

Conclusions

- 33 features for GPU vs CPU classification
- 92% average accuracy
- Low penalty (1ms)
- Classification time very low (10µs)

Thank you!

References

1.http://csgillespie.wordpress.com/2011/01/25/cpu-and-gpu-trends-over-time/

2.http://www.pgroup.com/lit/articles/insider/v1n1a1.htm

3.http://www.geeks3d.com/public/jegx/201003/gpu_compute_device.jpg

State of the Art

- Russell et al[5] used ML to decide what would be a basic block during compilation.
    - Features: AST internal distance, #child nodes, #successors, latency, #load/stores, #integers

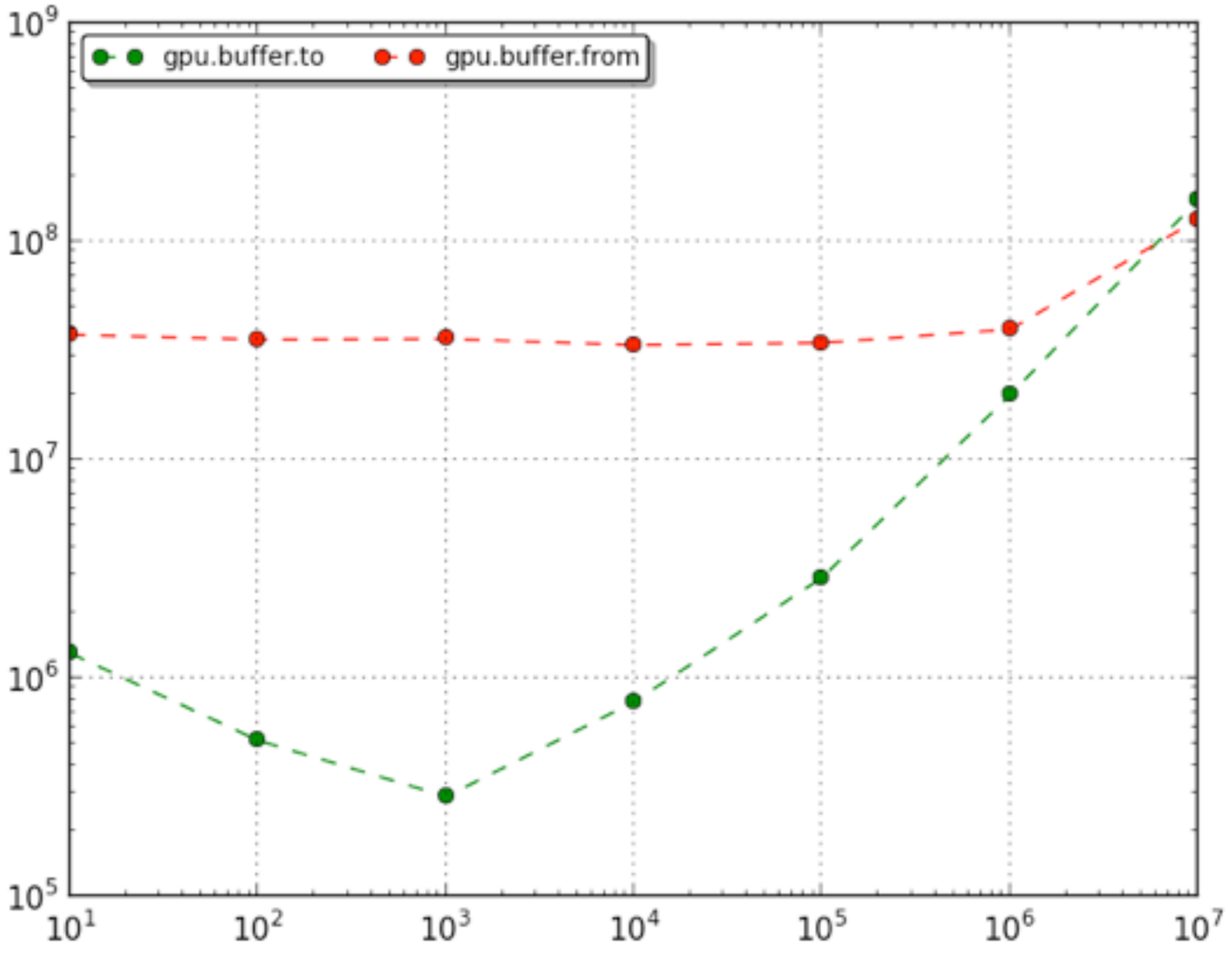
State of the Art II

- Cavazos et al[6] did the same for Java.
    - Features: #instructions per block, ratio of {load, call, branches}, #GC points, yield points.

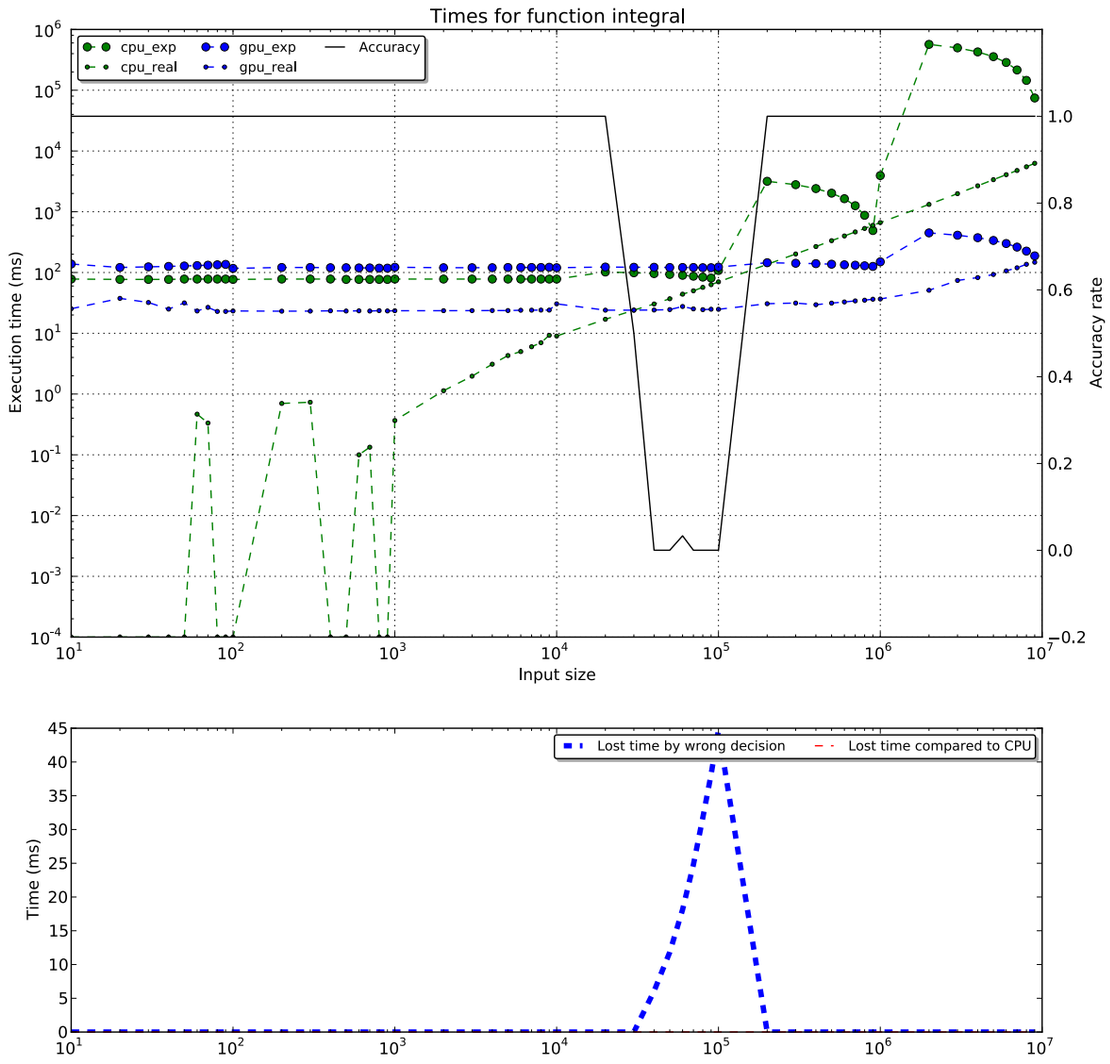
State of the Art III

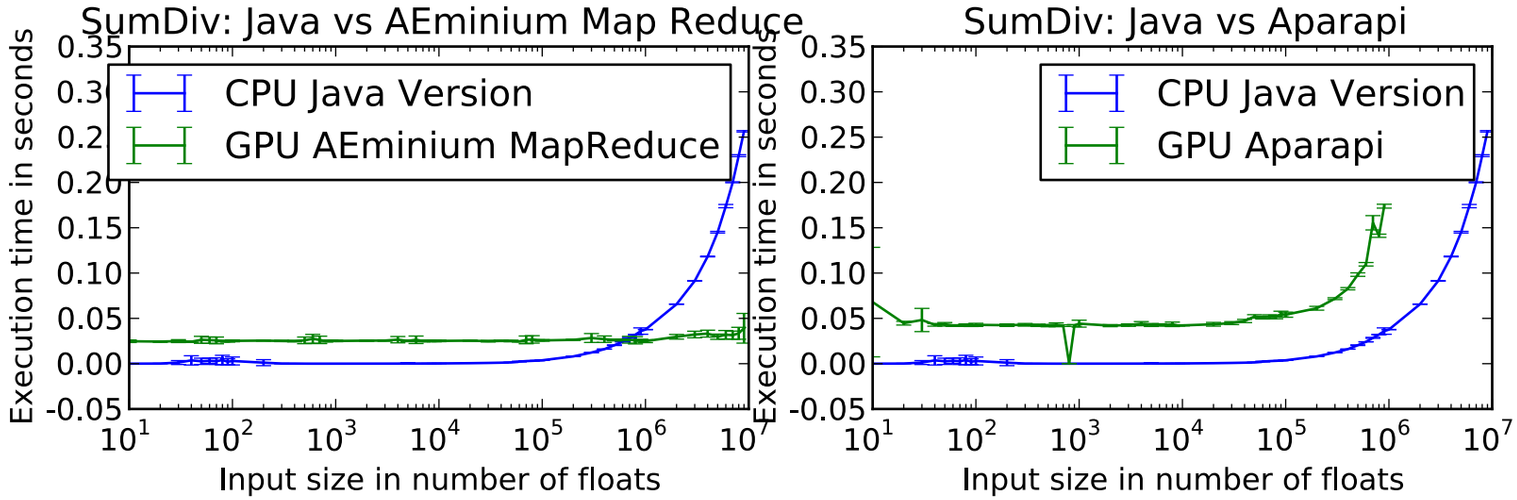
- Wang et al[7] used ML to map Tasks to Threads in a parallel environment.
    - Features: same as [6], data size, cache hits

GPU To vs From

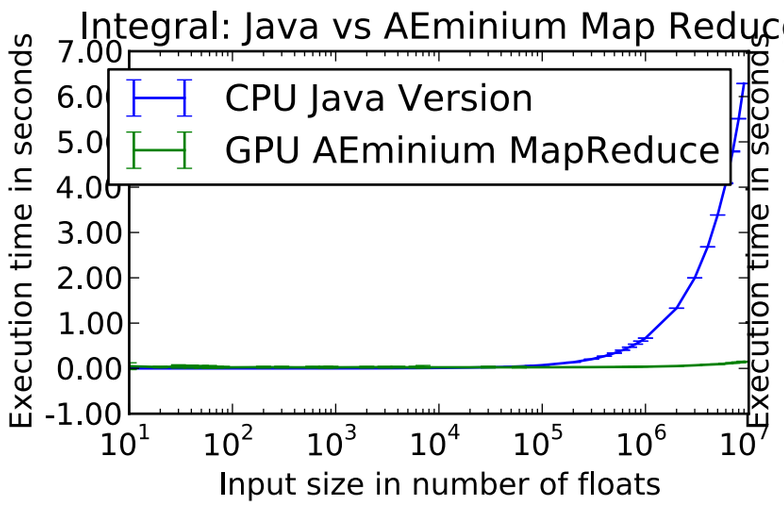


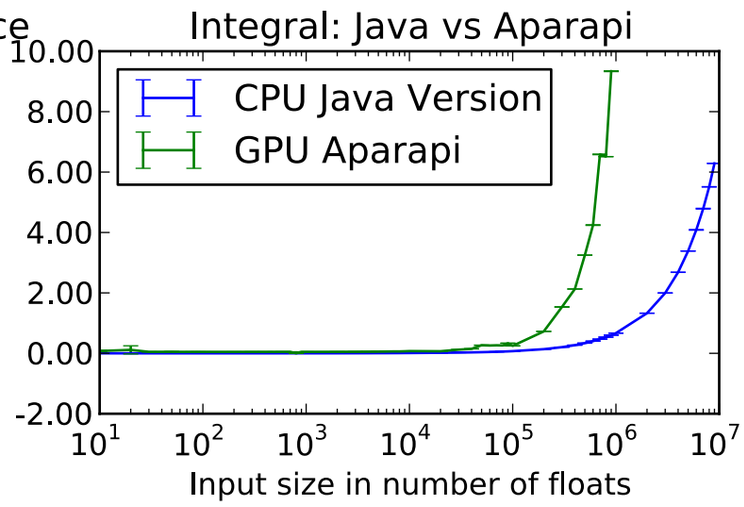
Old Method (exp)

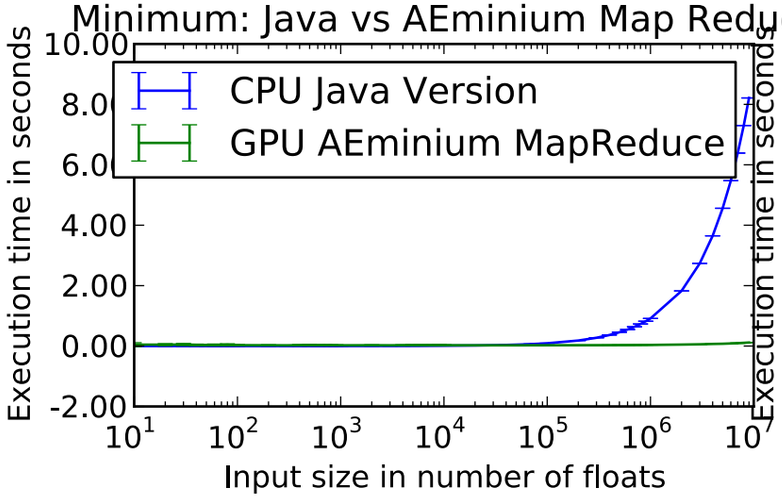


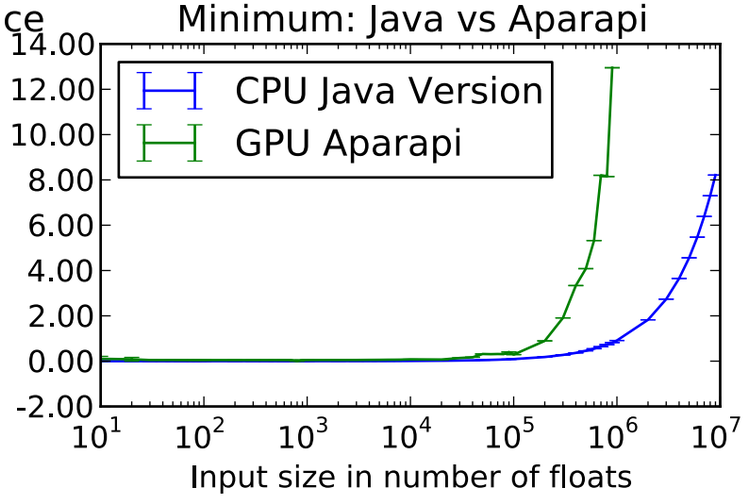

SumDiv: Java vs AEminium Map Reduce
CPU Java Version
GPU AEminium MapReduce
Execution time in seconds
Input size in number of floats

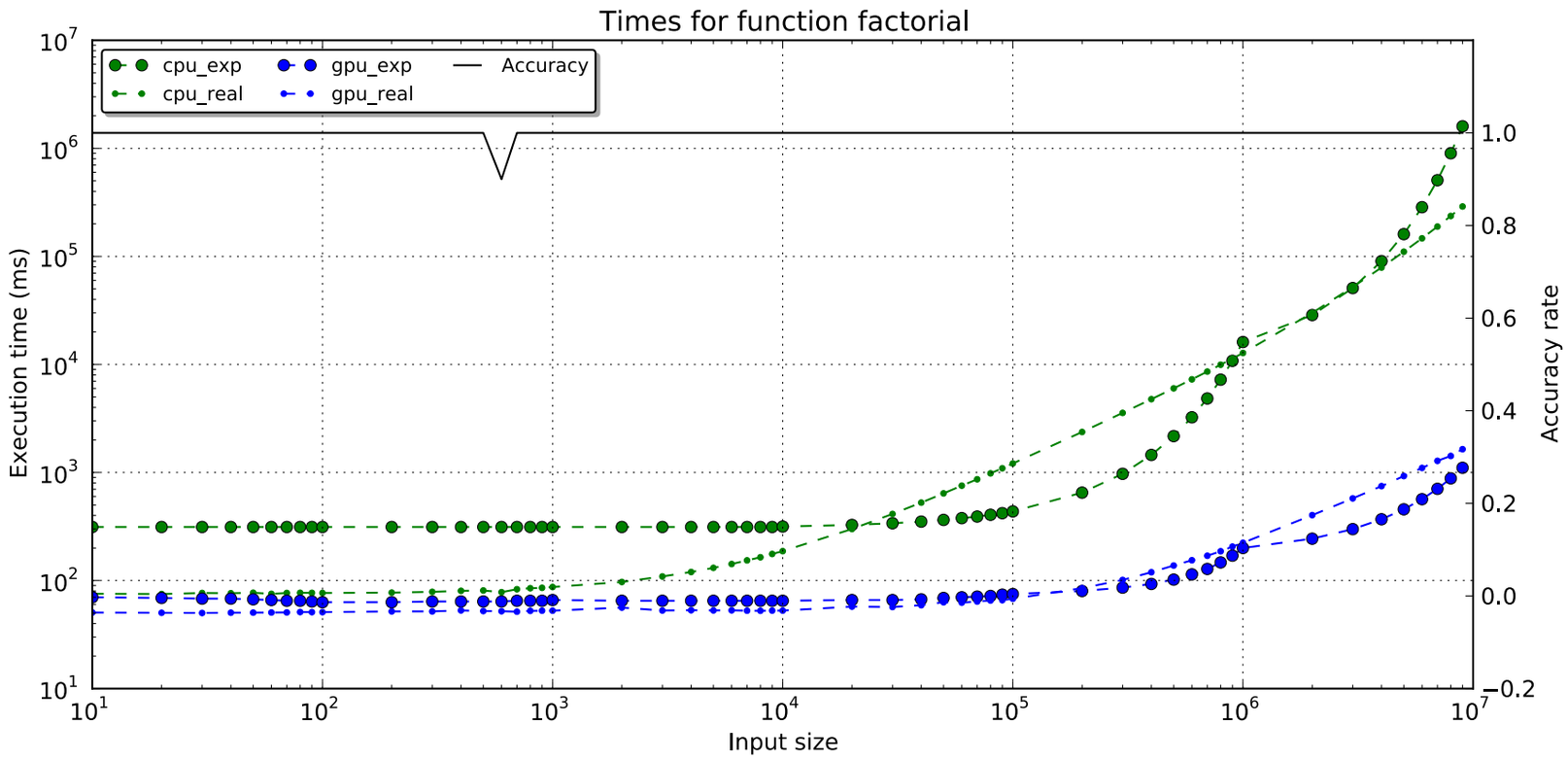
SumDiv: Java vs Aparapi
CPU Java Version
GPU Aparapi
Execution time in seconds
Input size in number of floats

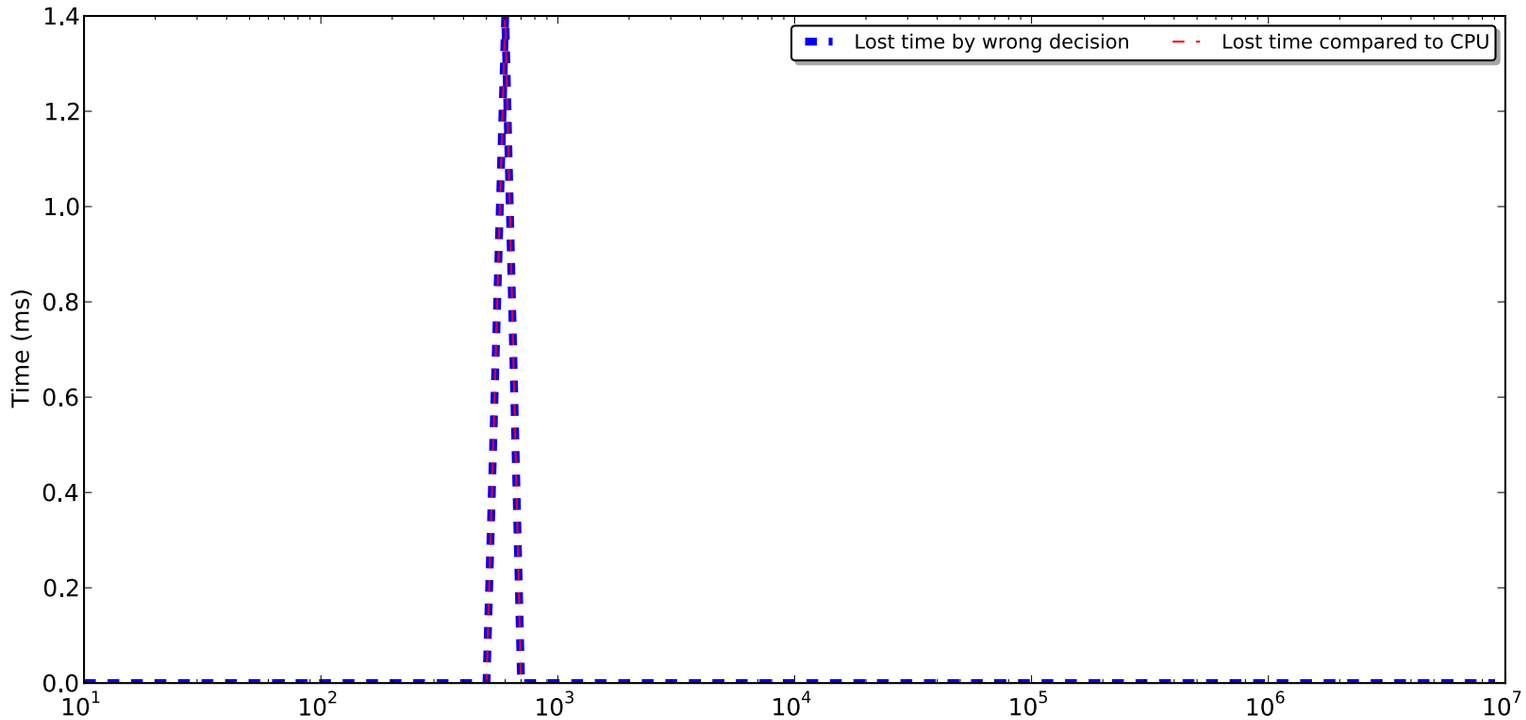

Integral: Java vs AEminium Map Reduce
CPU Java Version
GPU AEminium MapReduce
Execution time in seconds
Input size in number of floats

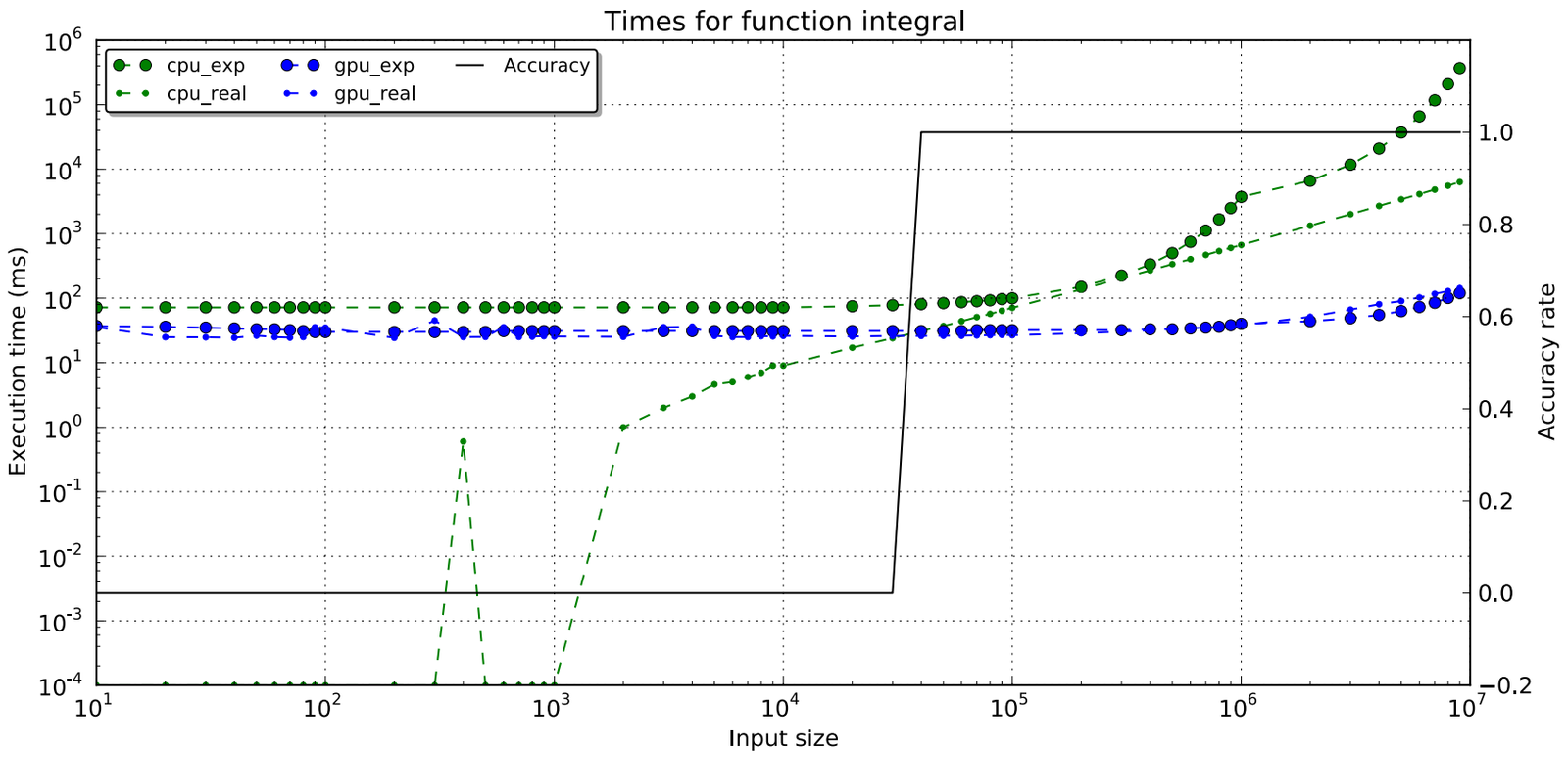

Integral: Java vs Aparapi
CPU Java Version
GPU Aparapi
Execution time in seconds
Input size in number of floats

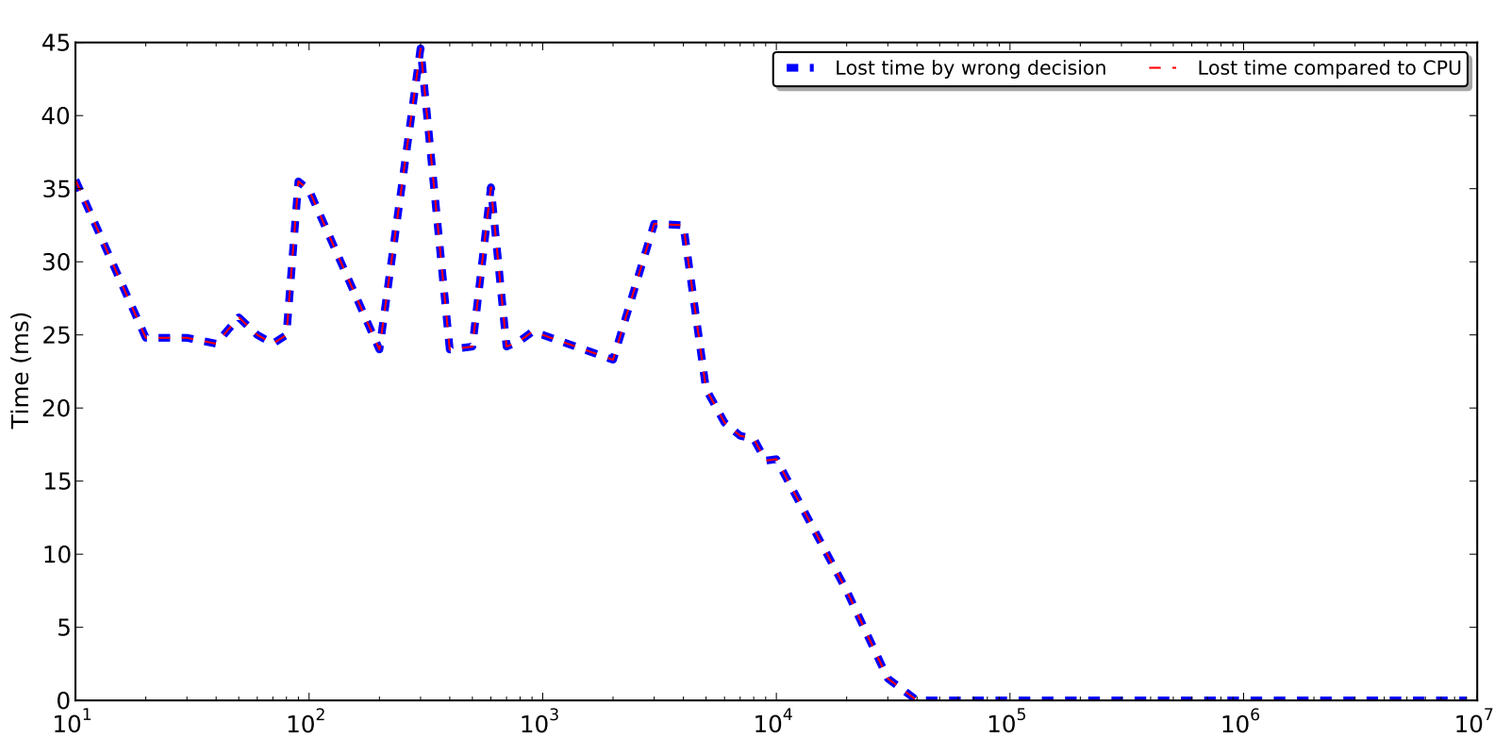

Minimum: Java vs AEminium Map Reduce
CPU Java Version
GPU AEminium MapReduce
Execution time in seconds
Input size in number of floats


Minimum: Java vs Aparapi
CPU Java Version
GPU Aparapi
Execution time in seconds
Input size in number of floats


Times for function factorial
cpu_exp
cpu_real
gpu_exp
gpu_real
Accuracy
Execution time (ms)
Accuracy rate
Input size


Lost time by wrong decision
Lost time compared to CPU
Time (ms)


Times for function integral
cpu_exp
cpu_real
gpu_exp
gpu_real
Accuracy
Execution time (ms)
Accuracy rate
Input size


Lost time by wrong decision
Lost time compared to CPU
Time (ms)