High Performance GPU Computing with Ruby

Prasun Anand

About me

- SciRuby Contributor
- Google Summer of Code 2016, 2017
- Genenetwork project
- Ruby Grant 2017

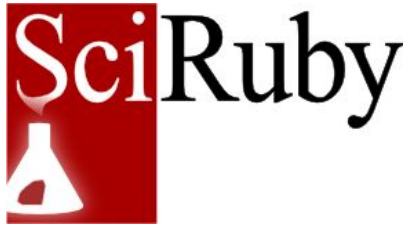
- Projects:
  - JRuby port of NMatrix
  - ArrayFire gem
  - RbCUDA

Google Summer of Code

SciRuby

GeneNetwork
University of Tennessee: www.genenetwork.org

SciRuby



SciRuby has been trying to push Ruby for scientific computing.

Popular Rubygems:

- NMatrix
- Daru
- Mixed_models
- Nyaplot
- Ipython Notebook

CUDA and OpenCL

CUDA is a parallel computing platform and programming model limited to NVIDIA hardware.

OpenCL (Open Computing Language) is supported across all GPU hardware.


ArrayFire

Af_Array

An array of numbers is stored as an Af_Array object.

This array is stored on GPU.

```
[1] pry(main)>    a = ArrayFire::Af_Array.new 2, [2,2],[1,2,3,4]
No Name  Array
[2 2 1 1]
  Offsets:    [0 0 0 0]
  Strides:    [1 2 4 4]
   1.0000     3.0000
   2.0000     4.0000

=>  #<ArrayFire::Af_Array:0x000000020aeab8>
```

```
[2] pry(main)>     b = a + a
No Name   Array
[2 2 1 1]
  Offsets:   [0 0 0 0]
  Strides:    [1 2 4 4]
   2.0000     6.0000
   4.0000     8.0000

=>  #<ArrayFire::Af_Array:0x000000020625c8>
```

```
[3] pry(main)>     b = a * a
No Name   Array
[2 2 1 1]
  Offsets:   [0 0 0 0]
  Strides:   [1 2 4 4]
   1.0000    9.0000
   4.0000   16.0000

=>   #<ArrayFire::Af_Array:0x00000001fe6f90>
```

```
VALUE arf_init(int argc, VALUE* argv, VALUE self)
{
  afstruct* afarray;
  Data_Get_Struct(self, afstruct, afarray);
  dim_t ndims = (dim_t)NUM2LONG(argv[0]);
  dim_t* dimensions = (dim_t*)malloc(ndims * sizeof(dim_t));
  dim_t count = 1;
  for (size_t index = 0; index < ndims; index++) {
    dimensions[index] = (dim_t)NUM2LONG(RARRAY_AREF(argv[1], index));
    count *= dimensions[index];
  }
  float* host_array = (float*)malloc(count * sizeof(float));
  for (size_t index = 0; index < count; index++) {
    host_array[index] = (float)NUM2DBL(RARRAY_AREF(argv[2], index));
  }

  af_create_array(&afarray->carray, host_array, ndims, dimensions, f64)

  return self;
}
```

BLAS and LAPACK

BLAS functionalities

- matmul
- Transpose

LAPACK functionalities

- det
- inverse
- norm
- qr
- cholesky
- svd
- lu

```
=> #<ArrayFire::Af_Array:0x00000001591db0>
[3] pry(main)>      result = ArrayFire::BLAS.matmul(left, right, :AF_MAT_NONE,
:AF_MAT_NONE)
No Name Array
[3 2 1 1]
  -39.0000   -74.0000
   68.0000   -17.0000
   86.0000   118.0000
```

Statistics

- Mean
- Median
- Variance

Random Engine

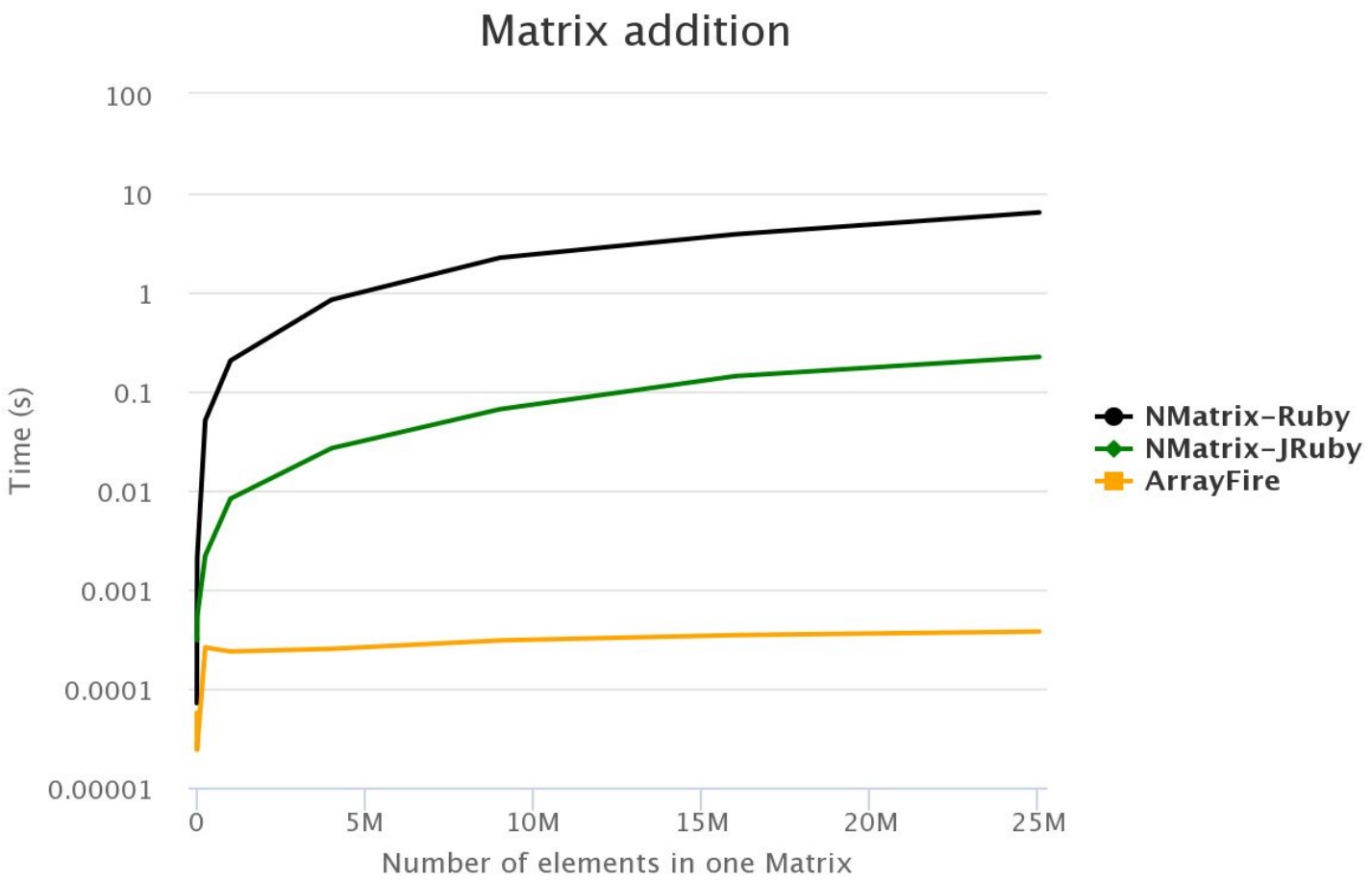
Random number generators

Types of Engines:

- :AF_RANDOM_ENGINE_PHILOX_4X32_10
- :AF_RANDOM_ENGINE_THREEFRY_2X32_16
- :AF_RANDOM_ENGINE_MERSENNE_GP11213

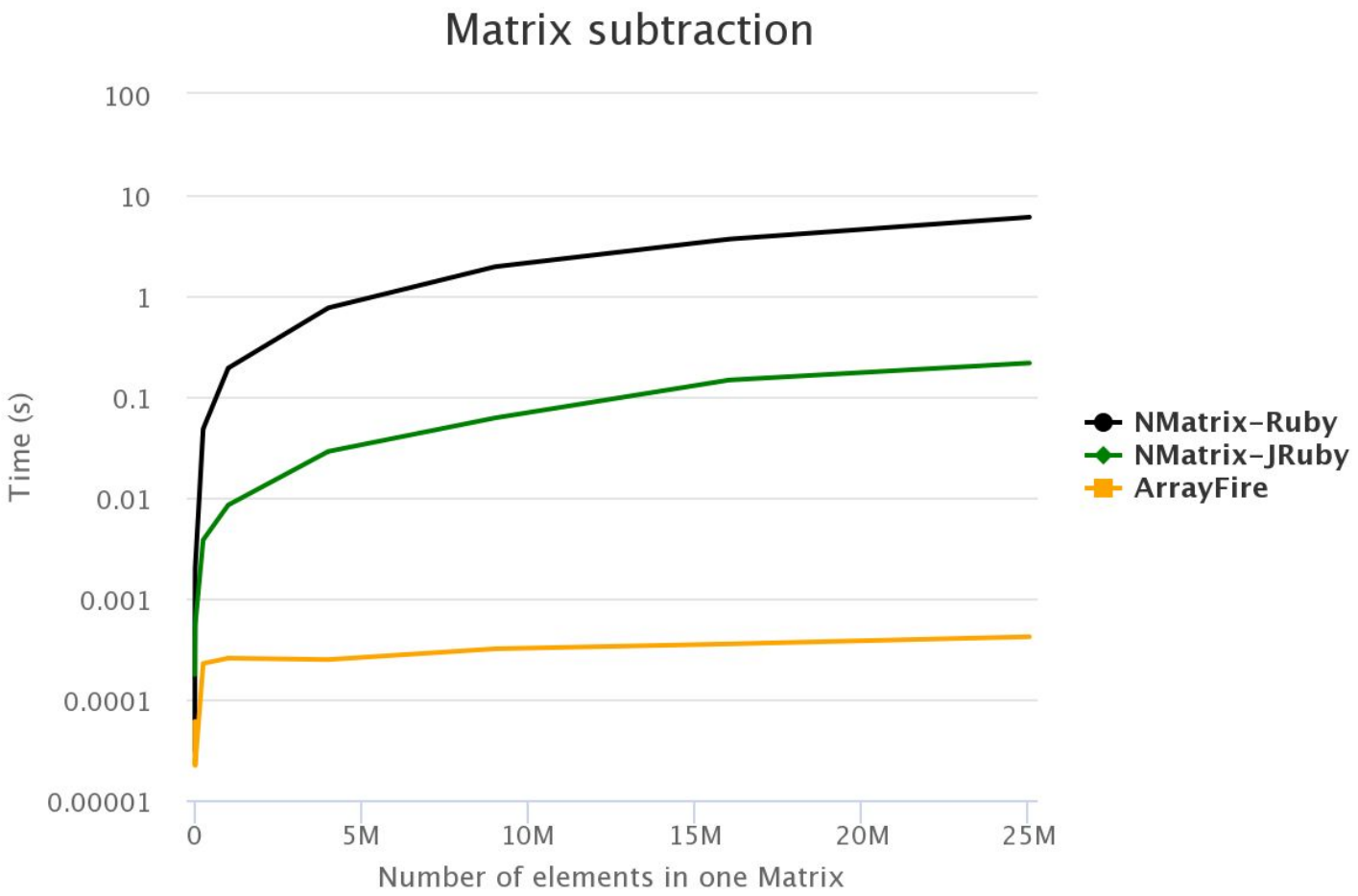
Benchmarks

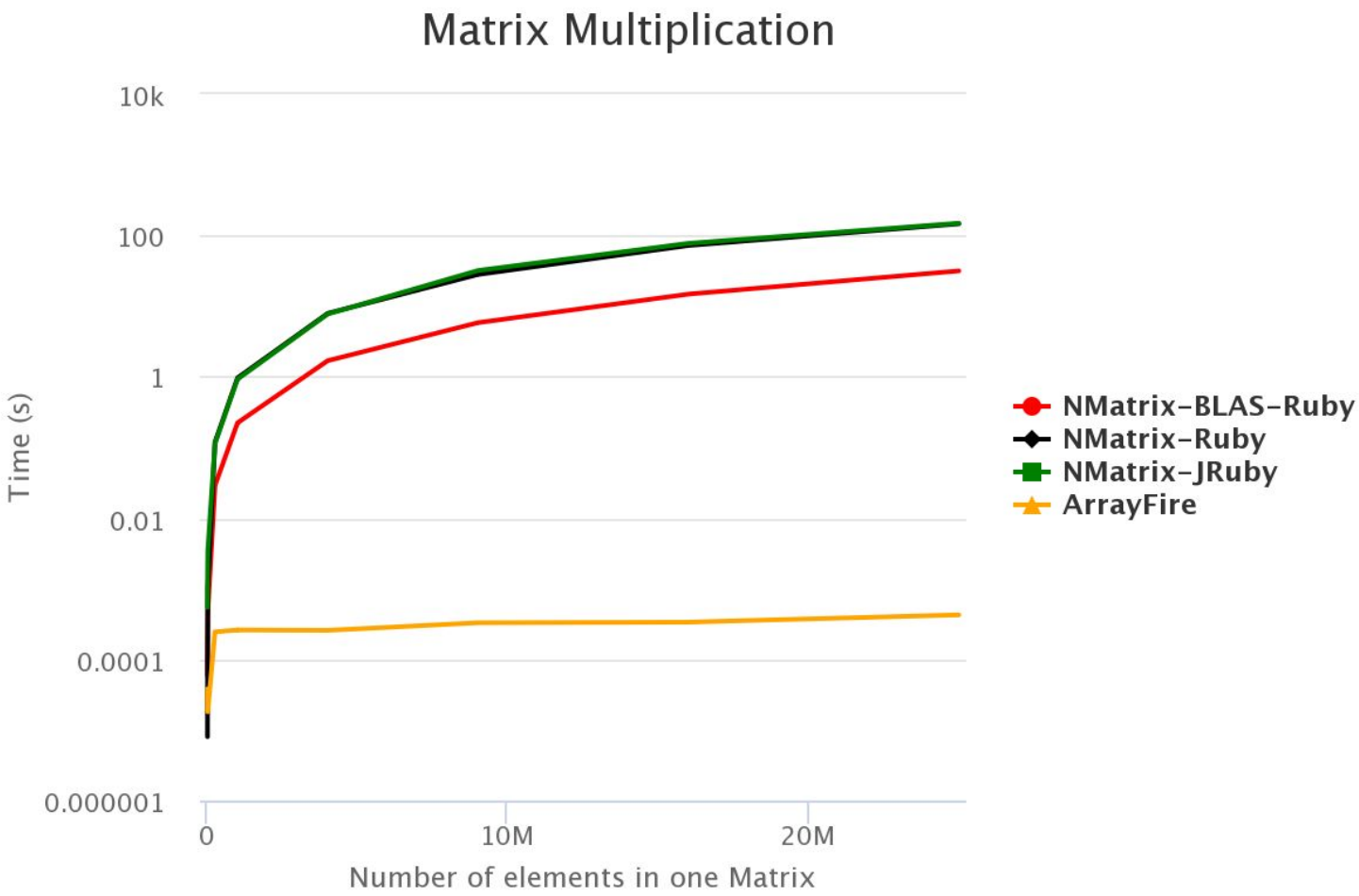
- AMD FX 8350 octacore processor
- Nvidia GTX 750Ti GPU
- CUDA backend
- Double dtype


Matrix addition
100
10
1
0.1
0.01
0.001
0.0001
0.00001
Time (s)
0
5M
10M
15M
20M
25M
Number of elements in one Matrix
NMatrix-Ruby
NMatrix-JRuby
ArrayFire
Highcharts.com

10,000 X

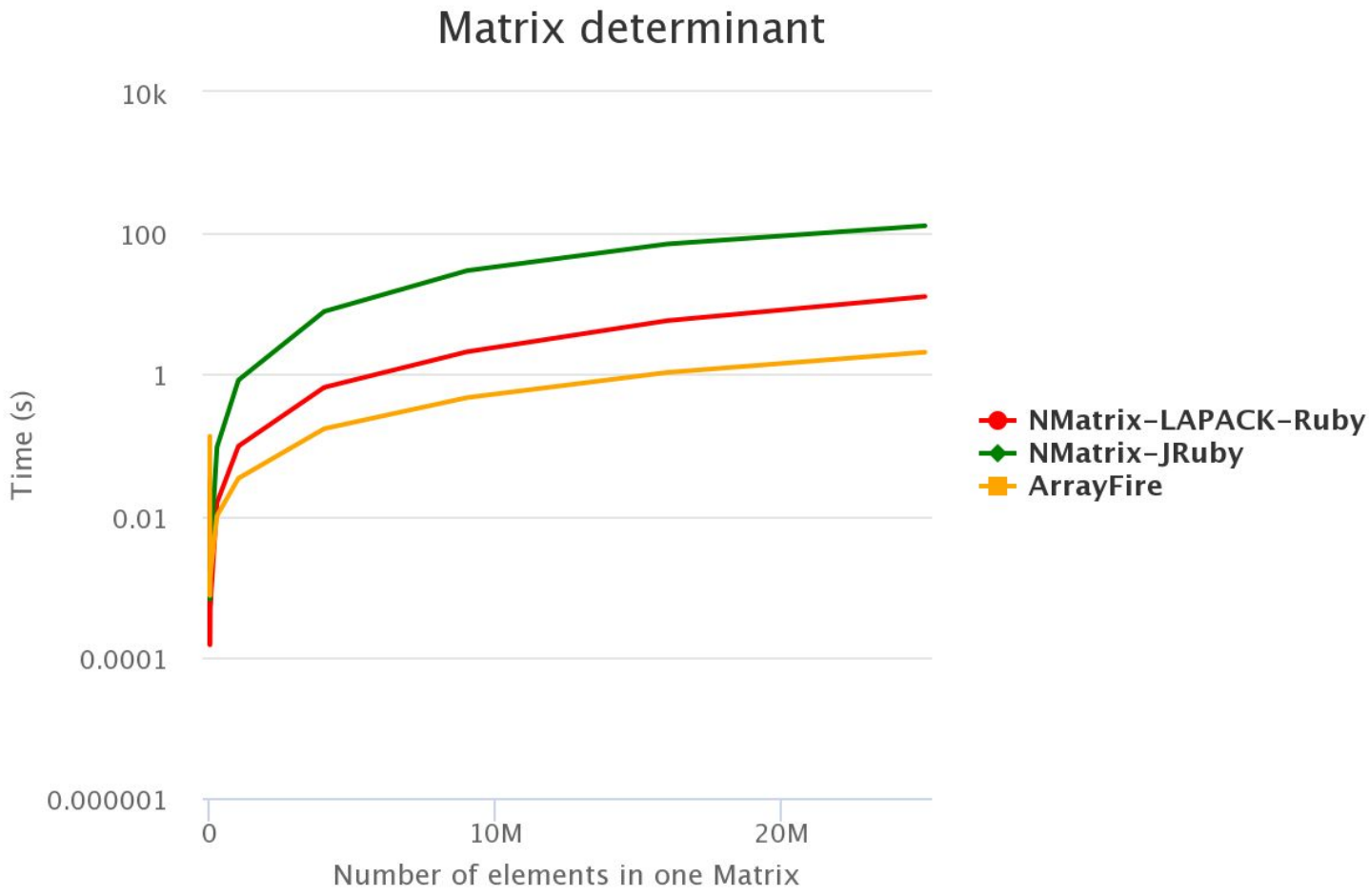
Faster than NMatrix-Ruby


Matrix subtraction
100
10
1
0.1
0.01
0.001
0.0001
0.00001
Time (s)
0
5M
10M
15M
20M
25M
Number of elements in one Matrix
NMatrix-Ruby
NMatrix-JRuby
ArrayFire
Highcharts.com


Matrix Multiplication
10k
100
1
0.01
0.0001
0.000001
Time (s)
0
10M
20M
Number of elements in one Matrix
NMatrix-BLAS-Ruby
NMatrix-Ruby
NMatrix-JRuby
ArrayFire
Highcharts.com

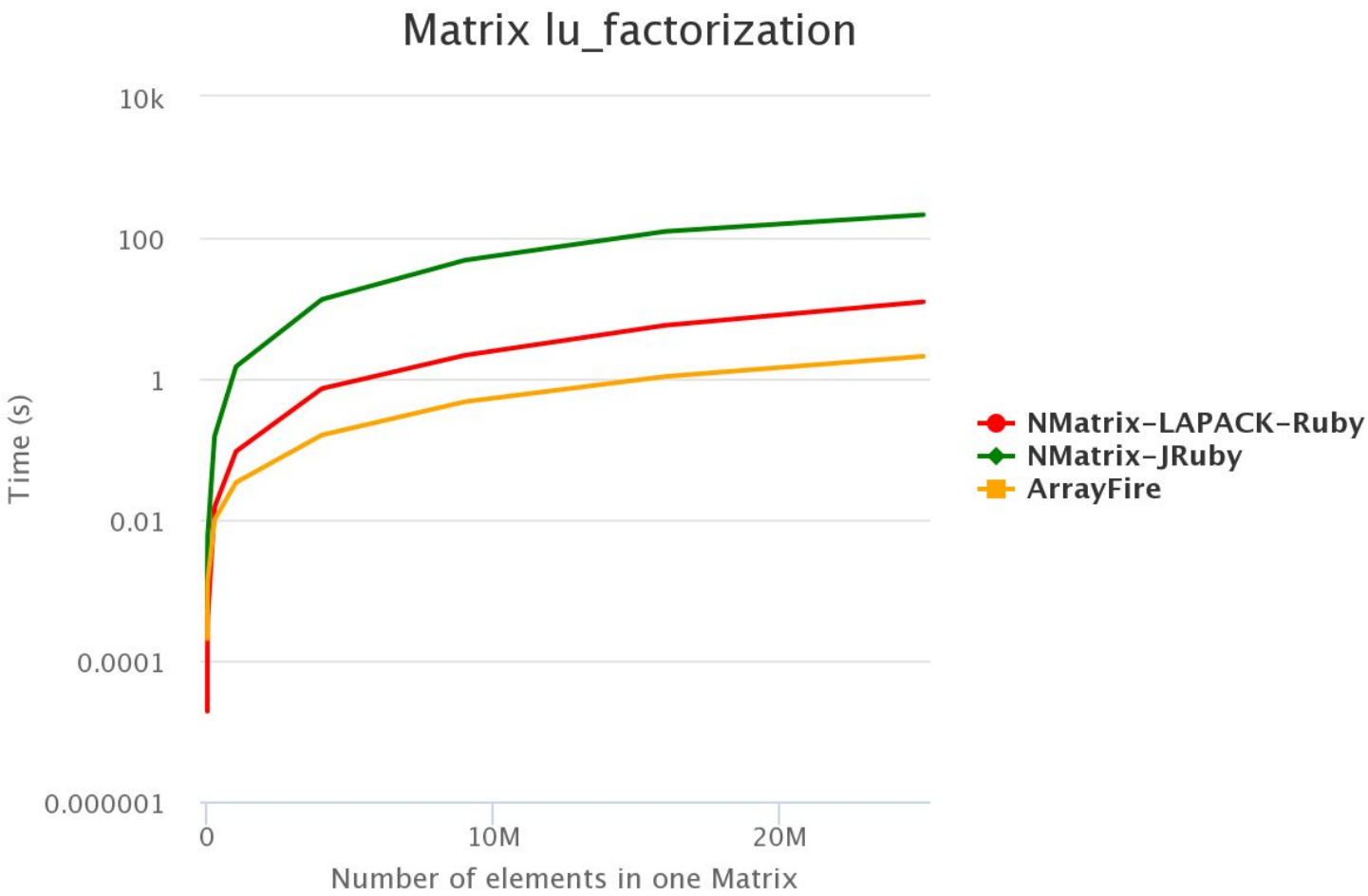
100,000 X

Faster than NMatrix-Ruby-BLAS


Matrix determinant
10k
100
1
0.01
0.0001
0.000001
Time (s)
0
10M
20M
Number of elements in one Matrix
NMatrix-LAPACK-Ruby
NMatrix-JRuby
ArrayFire
Highcharts.com

10 X

Faster than NMatrix-Ruby-Lapack


Matrix lu_factorization
10k
100
1
0.01
0.0001
0.000001
Time (s)
0
10M
20M
Number of elements in one Matrix
NMatrix-LAPACK-Ruby
NMatrix-JRuby
ArrayFire
Highcharts.com

RbCUDA

Custom Kernels

Scientific software require custom kernel code that suites to its needs.

Run kernel from a Ruby file, on the fly.

The kernel code is dynamically compiled

Run on the GPU hardware to manipulate the array pointers.

```
vadd_kernel_src = <<-EOS
extern "C" {
  __global__ void matSum(int *a, int *b, int *c)
  {
    int tid = blockIdx.x;
    if (tid < 100)
        c[tid] = a[tid] + b[tid];
  }
}
EOS

f = compile(vadd_kernel_src)
puts f.path
```

GPU Array

Generic pointer used to handle an array of elements on the GPU.

Memory copying from CPU to GPU and vice-versa.

Interfaced with NMatrix

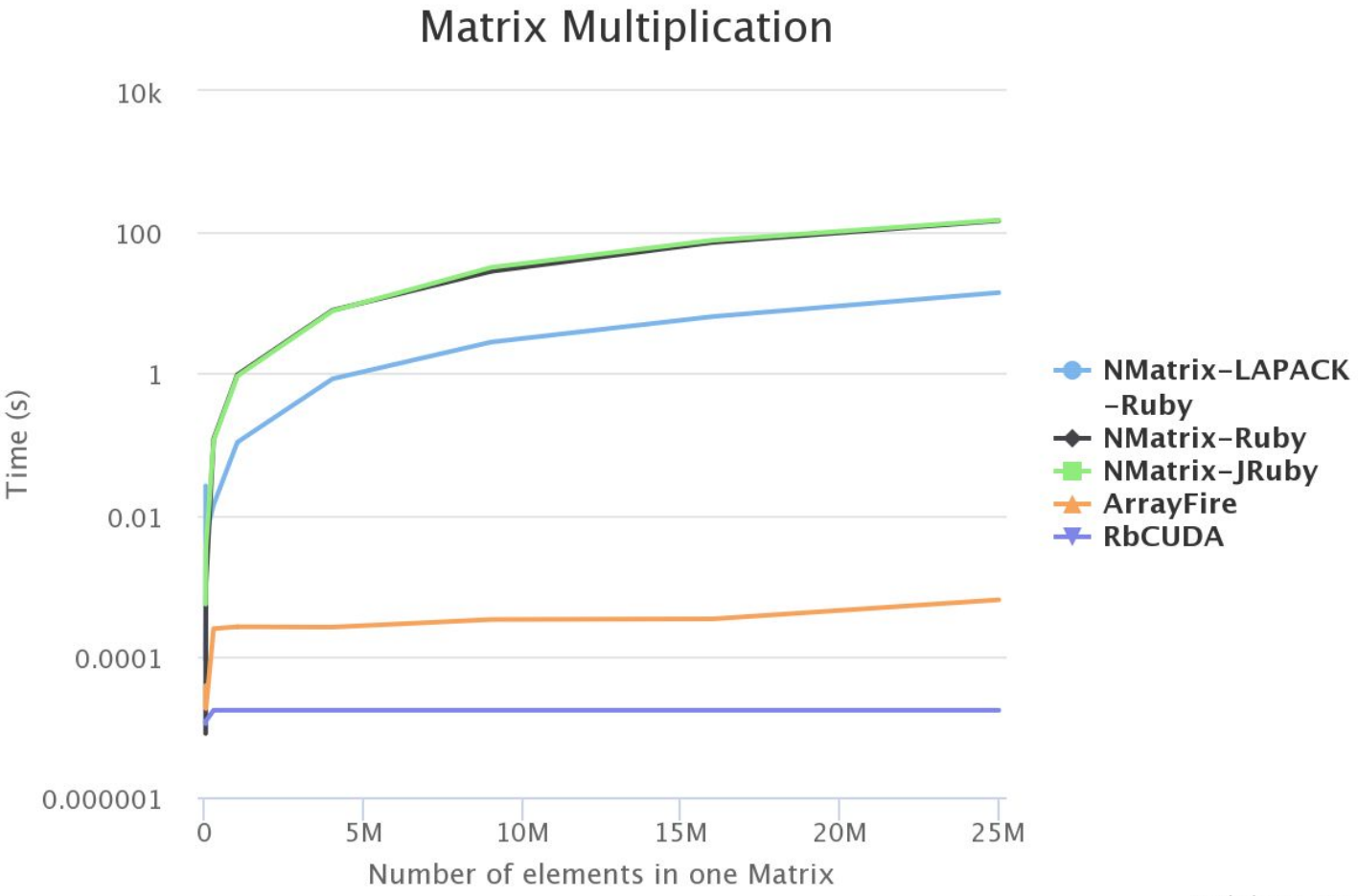
Interface with NArray

CuBLAS, CuSolver and CuRand

- BLAS routines
- Matrix Decomposition Routines
- Random Number Generators

Benchmarks

- AMD FX 8350 octacore processor
- Nvidia GTX 750Ti GPU
- Double dtype


Matrix Multiplication
10k
100
1
0.01
0.0001
0.000001
Time (s)
0
5M
10M
15M
20M
25M
Number of elements in one Matrix
NMatrix-LAPACK-Ruby
NMatrix-Ruby
NMatrix-JRuby
ArrayFire
RbCUDA
Highcharts.com

1,000,000 X

Faster than NMatrix-Ruby-BLAS

Future Work

- Image Processing APIs and Indexers
- Multiple dtypes

- RbCUDA is under active development.

Contributions are Welcome!

- https://github.com/arrayfire/arrayfire-rb
- https://github.com/prasunanand/rbcuda

Acknowledgements

1. Pjotr Prins
2. Pradeep Garigipati

Google Summer of Code

Ruby Association

Thank You

Github: prasunanand

Twitter: @prasun_anand

Blog: prasunanand.com