Rootbeer: Seamlessly using GPUs from Java

Phil Pratt-Szeliga. Dr. Jim Fawcett. Dr. Roy Welch. Syracuse University.

Rootbeer Overview and Motivation

- Rootbeer allows a developer to program a GPU in Java
- First GPU program was to speed up tomography in C
- The original program had many layers of looping and function calls

```
for(int degree = 0; degree < 360; ++degree){
    //function_calls
    for(int azimuth = 0; azimuth < 90; ++azimuth){
        //function_calls
        for(int x = 0; x < width; ++x){
            //function_calls
            for(int y = 0; y < width; ++y){
                //code_that_can_be_made_parallel_that_uses_globals
            }
        }
    }
}
```

- Developer needs to chose a cut level, collect jobs in a C++ vector, serialize state, transfer control to the GPU and deserialze after the GPU runs
- It was difficult to know in advance at what level to make the optimized cut to fit within GPU resource constraints while obtaining the maximum speedup
- Rootbeer makes trying cut levels simpler with automatic serialization and CUDA code generation for Java code

Hello World! Program

- Rootbeer allows a developer to program a GPU in Java

ArrayMultiplyKernel.java

```
01: public class ArrayMultiplyKernel implements Kernel {
02:     int[] m_array;
03:     public void gpuMethod(){
04:         m_array[RootbeerGpu.getThreadId()] *= 11;
05:     }
06: }
```

ArrayMultiply.java

```
01: public class ArrayMultiply {
02:     public void multiplyParallel(int[] array){
03:         List<Kernel> jobs = new ArrayList<Kernel>();
04:         for(int i = 0; i < array.length; ++i){
05:             ArrayMultiplyKernel kernel = new ArrayMultiplyKernel();
06:             kernel.m_array = array;
07:             jobs.add(kernel);
08:         }
09:         Rootbeer rootbeer = new Rootbeer();
10:         rootbeer.runAll(jobs);
11:     }
12: }
```

Advanced Matrix Multiply

- Uses shared memory and syncthreads with tiling

MatrixKernel.java (simplified)

```
01: public class MatrixKernel implements Kernel {
02:     public void gpuMethod(){
03:         int block_idxx = RootbeerGpu.getBlockIdxx();
04:         int thread_idxx = RootbeerGpu.getThreadIdxx();
05:         for(int block_iter = 0; block_iter < block_iters; ++block_iter){
06:             for(int sub_matrix = 0; sub_matrix < sub_matrix_size; ++sub_matrix){
07:                 float sum = 0;
08:                 for(int m = 0; m < m_size; ++m){
09:                     float a_value = a[a_src];
10:                     float b_value = b[b_src];
11:                     RootbeerGpu.setSharedFloat(thread_idxx, a_value);
12:                     RootbeerGpu.setSharedFloat((1024 + thread_idxx), b_value);
13:                     RootbeerGpu.synchthreads();
14:
15:                     for(int k = 0; k < 32; ++k){
16:                         a_value = RootbeerGpu.getSharedFloat(thread_row * 32 + k);
17:                         b_value = RootbeerGpu.getSharedFloat(1024 + k * 32 + thread_col);
18:                         sum += a_value * b_value;
19:                     }
20:                     RootbeerGpu.synchthreads();
21:                 }
22:                 c[dest_index] += sum;
23:             }
24:         }
25:     }
26: }
```

Advanced Matrix Multiply

MatrixApp.java (simplified) with Kernel Templates

```
public void gpuRun(){

    MatrixKernel matrix_kernel = new MatrixKernel(a, b, c, block_size, grid_size, block_iters);

    Rootbeer rootbeer = new Rootbeer();
    rootbeer.setThreadConfig(1024, grid_size);
    rootbeer.runAll(matrix_kernel);

}
```

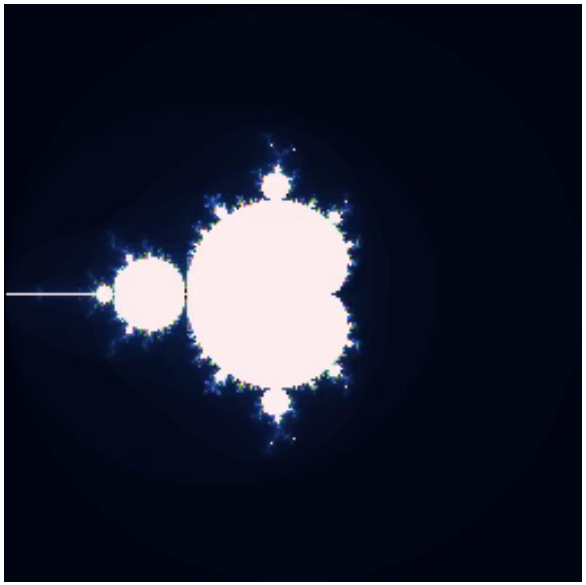
Performance (prototype, indexing needs work) (Java CPU time does not include transpose time)

System	Time	Speedup
Java CPU 4 cores	109 seconds	-
Rootbeer Java	5 seconds	21.8x
CUDA (matrixMul)	822 milliseconds	132.6x (6x)

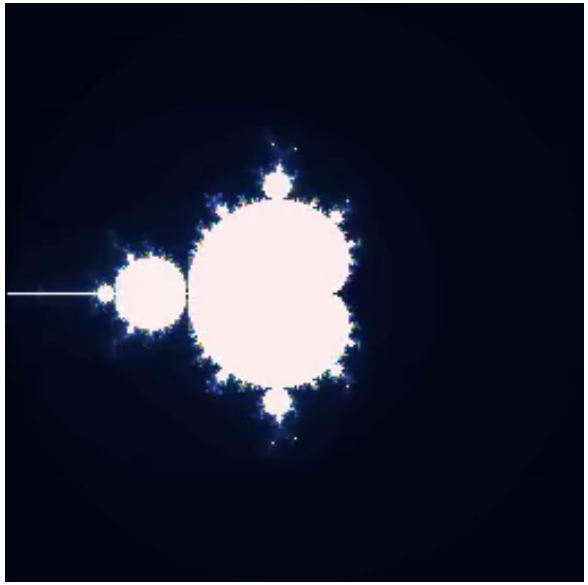
A[256][256]/B[256][256*256*14]
CPU: 4 core Xeon E5405 @ 2.00GHz with 16GB ram (DDR2 @ 667MHz (1.5ns))
GPU: 448 core Tesla C2050 @ 1.147GHz with 3GB (384-bit) ram over PCI-e x16 Gen2

MegaMandel

- Shows the power of using Java with a GPU



CPU Average Time: 80ms



GPU Average Time: 8ms

Example Provided by Thorsten Kiefer

Language Features

- Supported
  - Composite Reference Types (Classes)
  - Instance and Static Methods and Fields
  - N-Dimensional Arrays of Any Type
  - Exceptions
  - Strings
  - Synchronized Keyword
- Unsupported
  - Reflection
  - Dynamic Method Invocation
  - Native Code (JNI)
  - System.out.println
  - File/Network IO
- Coming Soon (two month's time expected)
  - Java Collections Framework support

Running Rootbeer

Download binary from http://rbcompiler.com/

- Running easy tests

```
$ java -jar Rootbeer.jar –runeasytests
```

- Running single test case (and with Native Emulation)

```
$ java -jar Rootbeer.jar –runtest SimpleTest -nemu
```

- Compiling and Running an Application

```
$ java -jar Rootbeer.jar app.jar app-gpu.jar
$ java -jar app-gpu.jar
```

- Testing GPU Installation

```
$ java -jar Rootbeer.jar -printdeviceinfo
```

Testing

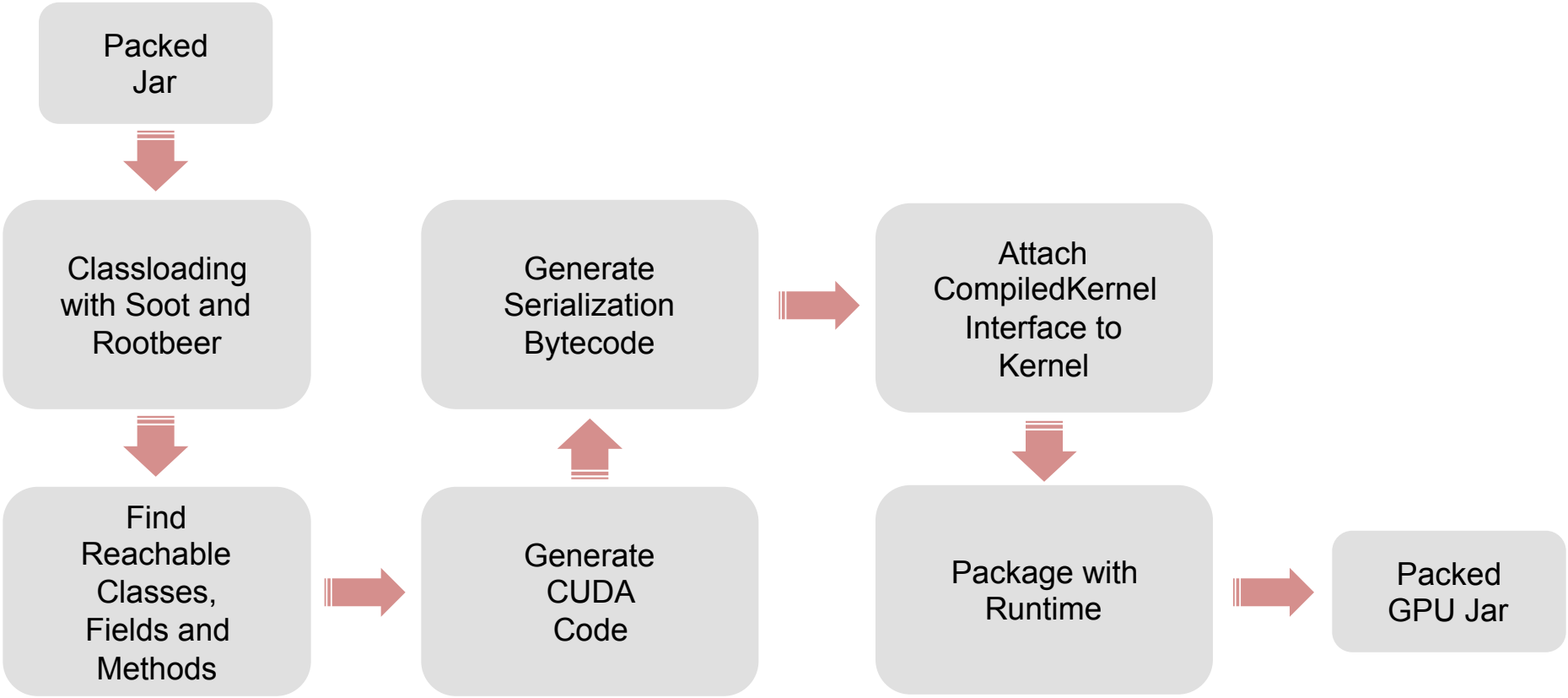
“If you haven’t tested it, it doesn’t work. Guaranteed.”

- Test Driven Development with Git/Github and Gitflow
  - 8.8k lines of test code
  - 20k lines of product code
  - Master
    - 42/43 tests pass on Windows, Linux and Mac
  - Develop
    - 49/60 tests pass on Windows, Linux and Mac

- Bugs (will be fixed in two month’s time approx.)
  - ExceptionBasicTest
  - AtomicLongTest
  - Java Collections
  - ClassLoading has issues
    - Separate Kernel reachable jar for now

Rootbeer Internals

Compilation



Rootbeer Internals

Memory spaces:

1. Root kernel pointer space
2. Object memory space (big)
3. Exception pointer space

CUDA Entry Point:

```
01: __global__ void entry(int * root_pointers, char * object_space, int * exceptions, int num_blocks){
02:     int loop_control = blockIdx.x * blockDim.x + threadIdx.x;
03:     if(loop_control >= num_blocks){
04:         return;
05:     } else {
06:         int handle = root_pointers[loop_control];
07:         int exception = 0;
08:         %%invoke_run%%(object_space, handle, &exception);
09:         exceptions[loop_control] = exception;
10:     }
11: }
```

cuModuleGetFunction(&cuFunction, cuModule, "_Z5entryPcS_PiPxS1_S0_S0_i");

Rootbeer Internals

CUDA Code for ArrayMult.gpuMethod

```
01: __device__ void rootbeer_examples_arraymult_ArrayMult_gpuMethod0_(char * gc_info, int thisref, int * exception){
02:     int r0 =-1 ;
03:     int $r1 =-1 ;
04:     int $i0 ;
05:     int $i1 ;
06:     int $i2 ;
07:     $r1 = instance_getter_rootbeer_examples_arraymult_ArrayMult_m_source(gc_info, r0, exception);
08:     if(*exception != 0){
09:         return;
10:     }
11:     $i0 = edu_syr_pcpratts_rootbeer_runtime_RootbeerGpu_getThreadIdxx(gc_info, exception);
12:     if(*exception != 0){
13:         return;
14:     }
15:     $i1 = int__array_get(gc_info, $r1, $i0, exception);
16:     if(*exception != 0){
17:         return;
18:     }
19:     $i2 = $i1 * 11;
20:     int__array_set(gc_info, $r1, $i0, $i2, exception);
21:     if(*exception != 0){
22:         return;
23:     }
24:     return;
25: }
```

Rootbeer Internals

CUDA Code for ArrayMult.m_source field ref

```
01: __device__ int
02: instance_getter_rootbeer_examples_arraymult_ArrayMult_m_source(char * gc_info, int thisref, int * exception){
03:     char * thisref_deref ;
04:     if(thisref == -1){
05:         *exception = 11;  //corresponds to CompiledKernel.getNullPointerNumber()
06:         return 0;
07:     }
08:     thisref_deref = edu_syr_pcpratts_gc_deref(gc_info, thisref);
09:     return *((int *) &thisref_deref[32]);
10: }
```

Rootbeer Internals

CUDA Code for getting m_source[index]

```
__device__ int int__array_get(char * gc_info, int thisref, int parameter0, int * exception) {
    int offset;
    int length;
    char * thisref_deref;
    offset = 16 + (parameter0 * 4);  //object header: 16 bytes. 4 is element size that is generated for each type
    if(thisref == -1){
        *exception = 11;
        return 0;
    }
    thisref_deref = edu_syr_pcpratts_gc_deref(gc_info, thisref);
    length = edu_syr_pcpratts_getint (thisref_deref, 8);
    if(parameter0 >= length){
        *exception = edu_syr_pcpratts_rootbeer_runtimegpu_GpuException_arrayOutOfBounds(gc_info,
            parameter0, thisref, length, exception);
        return 0;
    }
    return *((int *) &thisref_deref[offset]);
}
```

- ArrayOutOfBounds throws can be disabled
  - Compile with -noarraychecks

Rootbeer Internals

Optimizations/Notes

- **Pointers are 32 bit integers**
  - Smallest object is 16 bytes
  - Pointers are shifted before using
  - Allows for addressing of about 56GB inside an int
- **Fields are packed according to sort of size**
  - 64 bit fields must be read/written on 64 bit boundary, etc
- **Concurrent malloc is supported on GPU**
  - Heap is serialized packed
  - A heap end pointer is kept
  - AtomicAdd is used
    - Previous size is the returned malloc address
    - Add size is the size of the object
- **CUDA code is generated blindly (and some is hand coded)**
  - A simple pure Java C parser does dead code elimination before sending to nvcc
- **Temporaries are kept in ~/.rootbeer/**

Best Performance

- **Kernel Templates**
  - Reduces serialization time

- **Single-Dimensional Arrays of Primitive Types**
  - Optimizations quickly copy directly from JVM

- **Read a field once per GPU thread if it never changes**
  - Java Bytecode is compiled to CUDA C.
  - In un-optimized bytecode fields are always read
  - This translates to unnecessary reads to global ram

Future Work

- Support 2-3 dimensional block/thread ids
- Compile to ptx rather than CUDA C
- Allow leaving Java objects in GPU memory
- Support Java Collections Framework
  - Possibly swap in Optimized GPU Hash Table from Dr. John Owen's Lab (UC Davis).
- Support System.out.println
- Support File and Network IO
  - NVIDIA: Can you support sockets between CPU and GPU inside CUDA?
- GPU Garbage Collection
- Allow to disable null pointer checks
- Other small issues

Credits

- **Rootbeer is Open Source**
  - MIT license

- **Rootbeer Open Source Developers**
  - Thorsten Kiefer: http://toki.burn3r.de/
  - Aaloan Miftah: aaloanmiftah@yahoo.com
  - ValeriyB

- **Funding**
  - Rootbeer is supported by Syracuse University and NSF grant number MCB-0746066 to Roy D. Welch

Questions?

What are your favorite brands of Rootbeer?

1. Virgil's
2. Boylan
3. Saranac Rootbeer
4. IBC
5. Barq's
6. Mug
7. A&W

What is the difference between a computer scientist and a mathematician?

1. A computer scientist knows no limits.