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

```java
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 deserialize 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**

```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**

```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)

```java
public class MatrixKernel implements Kernel {
    public void gpuMethod() {
        int block_idxx = RootbeerGpu.getBlockIdxx();
        int thread_idxx = RootbeerGpu.getThreadIdxx();
        for (int block_iter = 0; block_iter < block_iters; ++block_iter) {
            for (int sub_matrix = 0; sub_matrix < sub_matrix_size; ++sub_matrix) {
                float sum = 0;
                for (int m = 0; m < m_size; ++m) {
                    float a_value = a[a_src];
                    float b_value = b[b_src];
                    RootbeerGpu.setSharedFloat(thread_idxx, a_value);
                    RootbeerGpu.setSharedFloat((1024 + thread_idxx), b_value);
                    RootbeerGpu.synchthreads();
                    for (int k = 0; k < 32; ++k) {
                        a_value = RootbeerGpu.getSharedFloat(thread_row * 32 + k);
                        b_value = RootbeerGpu.getSharedFloat(1024 + k * 32 + thread_col);
                        sum += a_value * b_value;
                    }
                    RootbeerGpu.synchthreads();
                }
                c[dest_index] += sum;
            }
        }
    }
}```
Advanced Matrix Multiply

MatrixApp.java (simplified) with Kernel Templates

```java
01:    public void gpuRun(){
02:    03:        MatrixKernel matrix_kernel = new MatrixKernel(a, b, c, block_size, grid_size, block_iters);
04:    05:        Rootbeer rootbeer = new Rootbeer();
06:        rootbeer.setThreadConfig(1024, grid_size);
07:        rootbeer.runAll(matrix_kernel);
08:    09:    }
```

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

<table>
<thead>
<tr>
<th>System</th>
<th>Time</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java CPU 4 cores</td>
<td>109 seconds</td>
<td>-</td>
</tr>
<tr>
<td>Rootbeer Java</td>
<td>5 seconds</td>
<td>21.8x</td>
</tr>
<tr>
<td>CUDA (matrixMul)</td>
<td>822 milliseconds</td>
<td>132.6x (6x)</td>
</tr>
</tbody>
</table>

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
“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

1. Packed Jar
2. Classloading with Soot and Rootbeer
3. Find Reachable Classes, Fields and Methods
4. Generate Classloading Bytecode
5. Generate CUDA Code
6. Attach CompiledKernel Interface to Kernel
7. Package with Runtime
8. Packed GPU Jar
Memory spaces:
1. Root kernel pointer space
2. Object memory space (big)
3. Exception pointer space

CUDA Entry Point:

```c
__global__ void entry(int * root_pointers, char * object_space, int * exceptions, int num_blocks){
    int loop_control = blockIdx.x * blockDim.x + threadIdx.x;
    if(loop_control >= num_blocks){
        return;
    } else {
        int handle = root_pointers[loop_control];
        int exception = 0;
        invoke_run(object_space, handle, &exception);
        exceptions[loop_control] = exception;
    }
}
```

cuModuleGetFunction(&cuFunction, cuModule, "_Z5entryPcS_PiPxs1_S0_S0_i");
CUDA Code for ArrayMult.gpuMethod

```
__device__ void rootbeer_examples_arraymult_ArrayMult_gpuMethod0_(char * gc_info, int thisref, int * exception){
    int r0 =-1 ;
    int $r1 =-1 ;
    int $i0 ;
    int $i1 ;
    int $i2 ;
    $r1 = instance_getter_rootbeer_examples_arraymult_ArrayMult_m_source(gc_info, r0, exception);
    if(*exception != 0){
        return;
    }
    $i0 = edu_syr_pcpratts_rootbeer_runtime_RootbeerGpu_getThreadIdxx(gc_info, exception);
    if(*exception != 0){
        return;
    }
    $i1 = int__array_get(gc_info, $r1, $i0, exception);
    if(*exception != 0){
        return;
    }
    $i2 = $i1 * 11;
    int__array_set(gc_info, $r1, $i0, $i2, exception);
    if(*exception != 0){
        return;
    }
    return;
}
```
CUDA Code for ArrayMult.m_source field ref

```c
__device__ int instance_getter_rootbeer_examples_arraymult_ArrayMult_m_source(char * gc_info, int thisref, int * exception) {
char * thisref_deref;
if (thisref == -1) {
    *exception = 11;  //corresponds to CompiledKernel.getNullPointerNumber()
    return 0;
}
thisref_deref = edu_syr_pcpratts_gc_deref(gc_info, thisref);
return *((int *) &thisref_deref[32]);
```
CUDA Code for getting m_source[index]

```c
__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-optimizied 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
  • ValeriB

• **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.