High Performance Java

Jordan J. Ruloff, James A. Ross, David A. Richie, Song J. Park, Dale R. Shires, Brian J. Henz
September 11, 2012
Introduction
Context
Java
Aparapi
Conclusion
Questions
Introduction

Software Goals:
Software Goals:

- Compatibility
Software Goals:

- Compatibility
- Reliability
Introduction

Software Goals:

- Compatibility
- Reliability
- Fault-Tolerance
Software Goals:

- Compatibility
- Reliability
- Fault-Tolerance
- Security
Software Goals:

- Compatibility
- Reliability
- Fault-Tolerance
- Security
- Reusability
Software Goals:
- Compatibility
- Reliability
- Fault-Tolerance
- Security
- Reusability
- Usability
Introduction

"For over a decade prophets have voiced the contention that the organization of a single computer has reached its limits and that truly significant advances can be made only by interconnection of a multiplicity of computers." - Gene Amdahl

“We stand at the threshold of a many core world. The hardware community is ready to cross this threshold. The parallel software community is not.” - Tim Mattson
"For over a decade prophets have voiced the contention that the organization of a single computer has reached its limits and that truly significant advances can be made only by interconnection of a multiplicity of computers." - Gene Amdahl
"For over a decade prophets have voiced the contention that the organization of a single computer has reached its limits and that truly significant advances can be made only by interconnection of a multiplicity of computers." - Gene Amdahl

"We stand at the threshold of a many core world. The hardware community is ready to cross this threshold. The parallel software community is not." - Tim Mattson
Context

Threat

Watch Point

Best Position
Java Threading

- Thread
  - Java 5
  - 1 Runnable Task
- ThreadPoolExecutor
  - Java 5
  - 1 Concurrent Queue
- Thread
  - Java 5
  - 1 Runnable Task

- ThreadPoolExecutor
  - Java 5
  - 1 Concurrent Queue

- ForkJoinPool
  - Java 7
  - N Concurrent Queues
  - Work-Stealing
Load A Set Of GeoTIFFs Into A Usable Data Format

NASA WorldWind Java Data Formats:
- ElevationModel
- SurfaceImage

Performance Comparison Between ThreadPoolExecutor And ForkJoinPool
Execution Time For Loading A GeoTIFF Multiple Times

![Graph showing execution time for loading GeoTIFFs](image)

- 2 x Intel® Xeon® X5675
- AMD Radeon™ HD 6970
- ThreadPoolExecutor
- ForkJoinPool

**Number Of GeoTIFFs**

**Time(s)**
Execution Time For Loading Multiple GeoTIFFs Multiple Times

![Graph showing execution time vs. number of GeoTIFFs]

- **2 x Intel® Xeon® X5675**
- **AMD Radeon™ HD 6970**
- **ThreadPoolExecutor**
- **ForkJoinPool**
Hardware

- Central Processing Units (CPUs)
- Graphics Processing Units (GPUs)
- Field-Programmable Gate Arrays (FPGAs)
- Digital Signal Processors (DSPs)
- Microcontrollers
Java Libraries

- JCuda
- jocl.org JOCL
- JogAmp.org JOCL
- JavaCL
- LWJGL
- Aparapi
- Rootbeer
Java Libraries

- JCuda
- jocl.org JOCL
- JogAmp.org JOCL
- JavaCL
- LWJGL
- Aparapi
- Rootbeer
Java Libraries

- JCuda
- jocl.org JOCL
- JogAmp.org JOCL
- JavaCL
- LWJGL
- Aparapi
- Rootbeer
Java Libraries

▶ JCuda
▶ jocl.org JOCL
▶ JogAmp.org JOCL
▶ JavaCL
▶ LWJGL
▶ Aparapi
▶ Rootbeer
- Bounding Box Construction
  - $O(1)$ Parallel Time
- Bitonic Sort
  - $O(\log^2(n))$ Parallel Time

http://upload.wikimedia.org/wikipedia/en/b/bd/BitonicSort1.svg
Preparation Kernels
Execution Time

<table>
<thead>
<tr>
<th>Number Of Triangles</th>
<th>2\times Intel® Xeon® X5675</th>
<th>AMD Radeon™ HD 6970</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Java Threads</td>
<td>OpenCL™ CPU</td>
</tr>
<tr>
<td></td>
<td>OpenCL™ GPU</td>
<td></td>
</tr>
</tbody>
</table>

Graph showing the execution time in seconds (Time(s)) for different numbers of triangles (Number Of Triangles) on various platforms. The graph includes a logarithmic scale on the y-axis and a linear scale on the x-axis.
**BVH Tree Construction**

- $O(\log(n))$ Parallel Time
- $O(2^{\text{Iteration}})$ Parallel Execution Units
1: procedure BuildBVHTree
2:   SetUpInitialNode
3:   while NotFullySplit do
4:     for all NodesWhichExist do
5:       for all Dimension ∈ Dimensions do
6:         ChooseBestSplit
7:       end for
8:     end for
9:     SplitNode
10:    SetupNodeForDataParallelPortion
11:   for all Dimension ∈ Dimensions do
12:     if DimensionWasNotSplitDimension then
13:       for all Indexes ∈ SortedIndexes do
14:         CalculateNewPosition
15:       end for
16:     end if
17:         end for
18:     end for
19: end while
20: for all \texttt{Node} \in \texttt{Nodes} do
21:     \texttt{CalculateAxisAlignedBoundingBoxes}
22: end for
23: for \texttt{Values} \in \texttt{Sorted} do
24:     \texttt{SetToSortedIndexInFirstDimension}
25: end for
26: end procedure
BVH Tree Construction
Kernel Execution Time

Number Of Triangles

Time(s)

2 x Intel® Xeon® X5675
AMD Radeon™ HD 6970
Java Threads
OpenCL™ CPU
OpenCL™ GPU

Brown Deer Technology
Intersection Kernel

- $O(\log(n))$ Parallel Time
- $O(2^{\text{Iterations} - \text{Iteration}})$ Parallel Execution Units
Intersection Kernel
Algorithm

1: procedure IntersectBVHTree
2: for all Ray ∈ Rays do in Parallel
3: for all Node ∈ LeafNodes do in Parallel
4: if RayHitsBoundingBox then
5:       FindClosestStructureHit
6: end if
7: end for
8: FindClosestNodeHit
9: end for
10: end procedure
Intersection Kernel Execution Time

2 x Intel® Xeon® X5675
AMD Radeon™ HD 6970
Java Threads
OpenCL™ CPU
OpenCL™ GPU

Brown Deer Technology

DRC HIGH PERFORMANCE TECHNOLOGIES GROUP
Conclusion

Java 7 ForkJoinPool Should Be Utilized Where It Is More Efficient

Aparapi - Pros:

- Higher Performance
- Utilize Additional Resources

Cons:

- Limited Usability
Java 7 ForkJoinPool Should Be Utilized Where It Is More Efficient
Conclusion

- Java 7 ForkJoinPool Should Be Utilized Where It Is More Efficient
- Aparapi
  - Pros:
    - Higher Performance
    - Utilize Additional Resources
Java 7 ForkJoinPool Should Be Utilized Where It Is More Efficient

Aparapi
- Pros:
  - Higher Performance
  - Utilize Additional Resources
- Cons:
  - Limited Usability
Questions?