Real-Time Simulation of Ultrasound Fields

Richard Abrich
Valentin Berbenetz
Matthew Thorpe
What is Ultrasound?

• High frequency sound waves
  – Used to detect blood velocity (Doppler Effect)
Why Simulate Ultrasound?

Pilot Training

Real

High cost
High risk

Simulated

Low cost
Low risk
Why Simulate Ultrasound?

Pilot Training

Real
High cost
High risk

Simulated
Low cost
Low risk

Medical Training

?
Presentation Outline

• Modelling Ultrasound
• CUDA API
• CPU/GPU Algorithms
• Testing & Results
• Conclusion
Modelling Ultrasound

Transducer

Acoustic Monopoles

Sound Waves

Base Pulse

Blood Cells

Acoustic Scatterers
Modelling Ultrasound
Modelling Ultrasound

Base Pulse

M0

M1

d0

d1

S0

Amplitudes at S0

?
Modelling Ultrasound

Base Pulse

$M_0$

$d_0$

$M_1$

$d_1$

$S_0$

Amplitudes at $S_0$

$d_0 - d_1$   Pulse length

time
Modelling Ultrasound

Potential race condition

\[ d_0 - d_1 \]  Pulse length

\[ \text{time} \]
Modelling Ultrasound

Potential race condition

d0 – d1  Pulse length

+  time
Modelling Ultrasound

Many Potential race conditions!
Modelling Ultrasound

Many Potential race conditions!

Amplitude Accumulator

$M_0, M_1, M_2, \ldots, M_n, S_0, S_1, S_2, \ldots, S_m$

Amplitude Accumulator
System Overview

1. Calculate Distances Between each monopole/scatterer
2. Calculate Amplitudes As a function of distances
3. Generate ECG Reduction, FFT
System Overview

Calculate Distances Between each monopole/scatterer

Calculate Amplitudes As a function of distances

Generate ECG Reduction, FFT

Computationally Expensive!
How Expensive?

Acoustic Scatterers: $10^4$

Acoustic Monopoles: $10^3$

Base Pulse: $10^2$
How Expensive?

Acoustic Scatterers: $10^4$
Acoustic Monopoles: $10^3$
Base Pulse: $10^2$

$= 10^9$ calculations *per frame*

Higher values achieve greater accuracy
CUDA Architecture

- Threads arranged into 3D blocks
- Blocks arranged into 2D grid
CUDA Architecture

• Threads arranged into 3D blocks
• Blocks arranged into 2D grid

150x slower than shared
CUDA Architecture

5 Dimensions

1. Block height
2. Block width
3. Block depth
4. Grid height
5. Grid width

3 Variables

1. # scatterers
2. # monopoles
3. Pulse length

$5 \times 4 \times 3 = 60$ possible execution configurations

Limitations:
- Max # threads per block
- Max # blocks per grid
Implementations

• CPU
  – Single-threaded
  – Multi-threaded

• GPU
  – Loop
  – Reduction
CPU Implementation
(Single-Threaded)

1 for each monopole
2 for each scatterer
3 for each pulse entry
4 calculate contribution
5 add to accumulator
CPU Implementation (Multi-Threaded)

0 #pragma omp parallel for
1 for each monopole
2 for each scatterer
3 for each pulse entry
4 calculate contribution
5 add to accumulator
• 1 Block = 1 pulse
• 1 Thread = 1 entry in pulse
• Position in grid corresponds to monopole/scatterer pair
• Does not work due to a WAW dependence
  — Multiple accumulator elements may be written to simultaneously by different blocks
GPU Reduction

Scatterers

Monopoles

Accumulators
GPU Reduction

Monopoles

Scatterers

Accumulators
GPU Reduction

Monopoles

Scatterers

Accumulators
GPU Reduction

Reduction summation

Accumulators
GPU Reduction

• Each block has its own copy of data to read
  – Prevents thread read competition (serialization)
• Dynamically optimize grid size based on available memory
• Use shared memory where possible
• Use barriers to synchronize threads
GPU Loop

Monopoles

Scatterers

Accumulator
GPU Loop

Monopoles

Scatters

Accumulator
GPU Loop

• Loop through monopoles inside the kernel
  – Reduces kernel launching overhead
• Implement other code optimizations
  – Loop invariant code motion

• Use shared memory where possible
• Use barriers to synchronize threads
# Testbed

<table>
<thead>
<tr>
<th>Name</th>
<th>Low End</th>
<th>High End</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU</td>
<td>GPU</td>
</tr>
<tr>
<td>Name</td>
<td>Atom D525</td>
<td>NVIDIA Ion 2</td>
</tr>
<tr>
<td>Speed</td>
<td>1.8GHz (HT)</td>
<td>1.09GHz</td>
</tr>
<tr>
<td>Cores</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>Memory</td>
<td>2GB</td>
<td>0.43GB</td>
</tr>
</tbody>
</table>
Varying Pulse Length (Low End)

Frame Generation Time vs. Pulse Length (Low-End)

- CPU single-threaded
- CPU multi-threaded
- GPU Loop
- GPU Reduction

Frame Time [ms]

# Cycles
Varying Pulse Length (High End)

Frame Generation Time vs. Pulse Length (High-End)

Frame Time [ms]

# Cycles

- CPU single-threaded
- CPU multi-threaded
- GPU Loop
- GPU Reduction
Varying # Monopoles (Low End)

Frame Generation Time vs. # Monopoles (Low-End)
Varying # Monopoles (High End)

Frame Generation Time vs. # Monopoles (High-End)

Frame Time [ms] vs. # Monopoles

- CPU single-threaded
- CPU multi-threaded
- GPU Loop
- GPU Reduction
Varying # Scatterers
(Low End)

Frame Generation Time vs. # Scatterers (Low-End)
Varying # Scatterers
(High End)

Frame Generation Time vs. # Scatterers (High-End)

- CPU single-threaded
- CPU multi-threaded
- GPU Loop
- GPU Reduction
Future Work

• Multiple asynchronous kernels
• Multiple GPUs
• Low level CUDA optimizations
  – Warp synchronization
  – PTX machine code
• Impact of rendering on performance
Conclusion

- Sub-linear performance for all but single-threaded CPU

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Use Case</th>
<th>Real-Life Implication</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU Looping</td>
<td>Small workloads</td>
<td>Good for real-time applications</td>
</tr>
<tr>
<td>GPU Reduction</td>
<td>Large workloads</td>
<td>Good for high-accuracy applications</td>
</tr>
<tr>
<td>CPU multi-threaded</td>
<td>GPU unavailable</td>
<td>Simple to implement</td>
</tr>
</tbody>
</table>
Conclusion

- Sub-linear performance for all but single-threaded CPU

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Use Case</th>
<th>Real-Life Implication</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU Looping</td>
<td>Small workloads</td>
<td>Good for real-time applications</td>
</tr>
<tr>
<td>GPU Reduction</td>
<td>Large workloads</td>
<td>Good for high-accuracy applications</td>
</tr>
<tr>
<td>CPU multi-threaded</td>
<td>GPU unavailable</td>
<td>Simple to implement</td>
</tr>
<tr>
<td>CPU single-threaded</td>
<td>1985</td>
<td>Back to the Future</td>
</tr>
</tbody>
</table>
Thank You