Real-Time Simulation of Ultrasound Fields

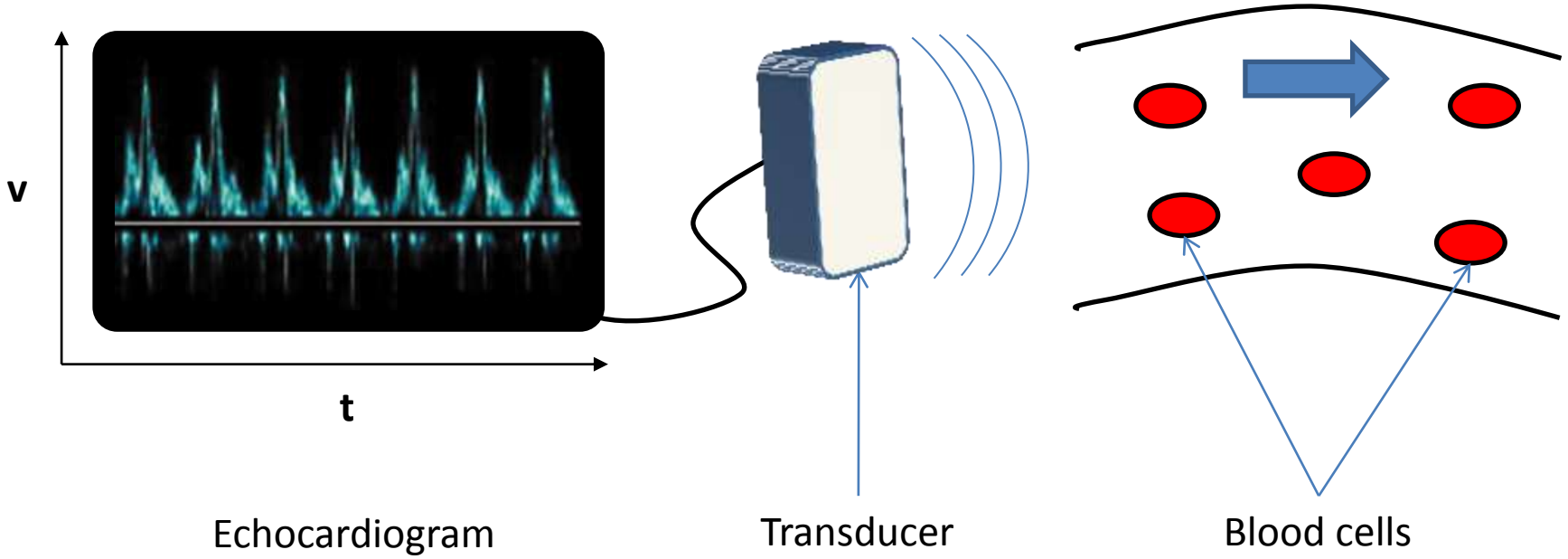
Richard Abrich

Valentin Berbenetz

Matthew Thorpe

What is Ultrasound?

- High frequency sound waves
  - Used to detect blood velocity (Doppler Effect)



Why Simulate Ultrasound?

	Real	Simulated
Pilot Training		
	High cost High risk	Low cost Low risk

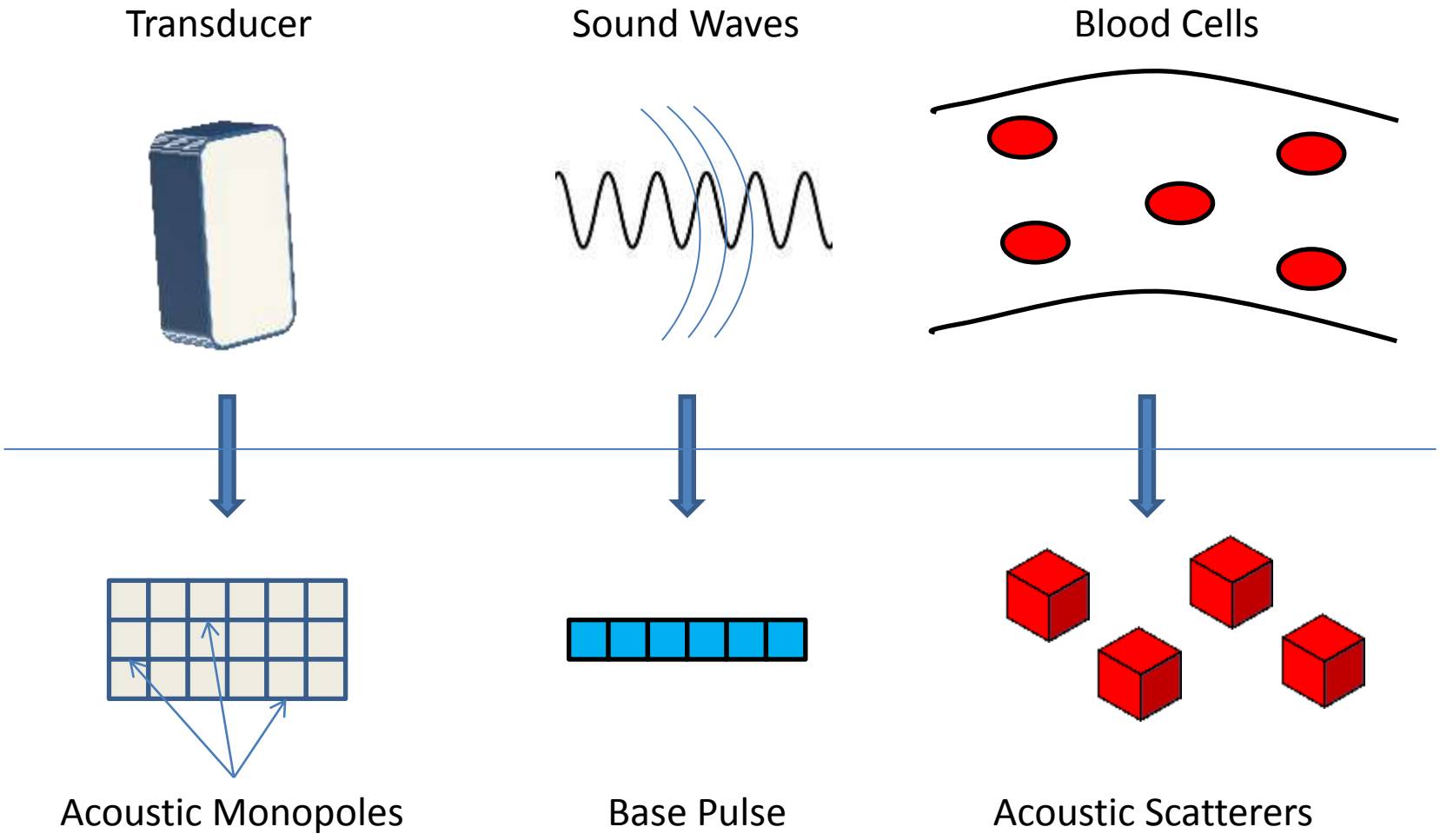
Why Simulate Ultrasound?

	Real	Simulated
Pilot Training		
	High cost High risk	Low cost Low risk
Medical Training		?

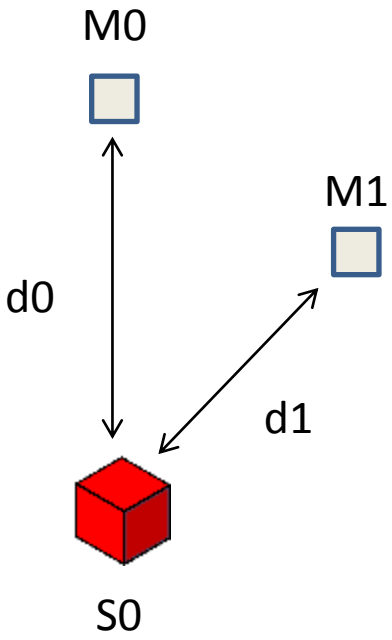
Presentation Outline

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

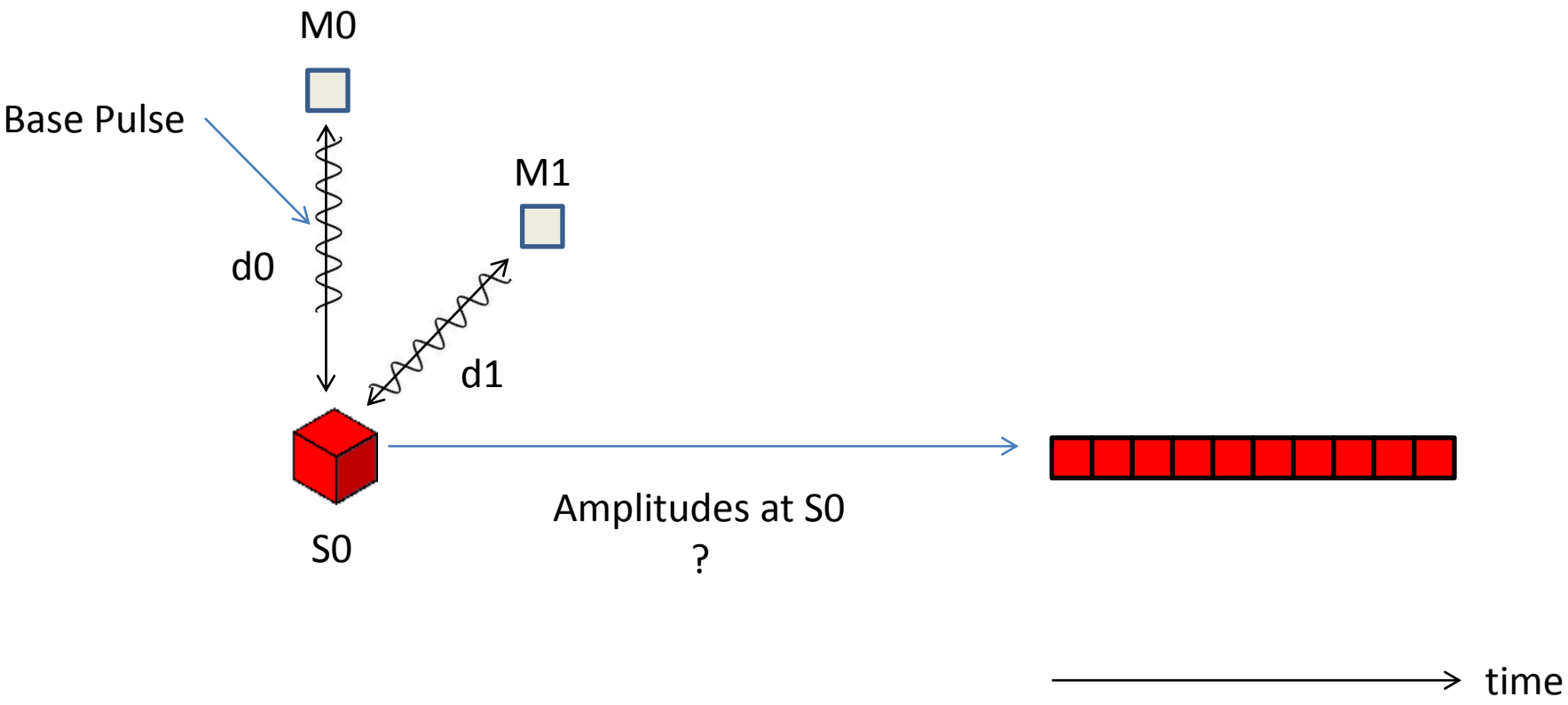
Modelling Ultrasound



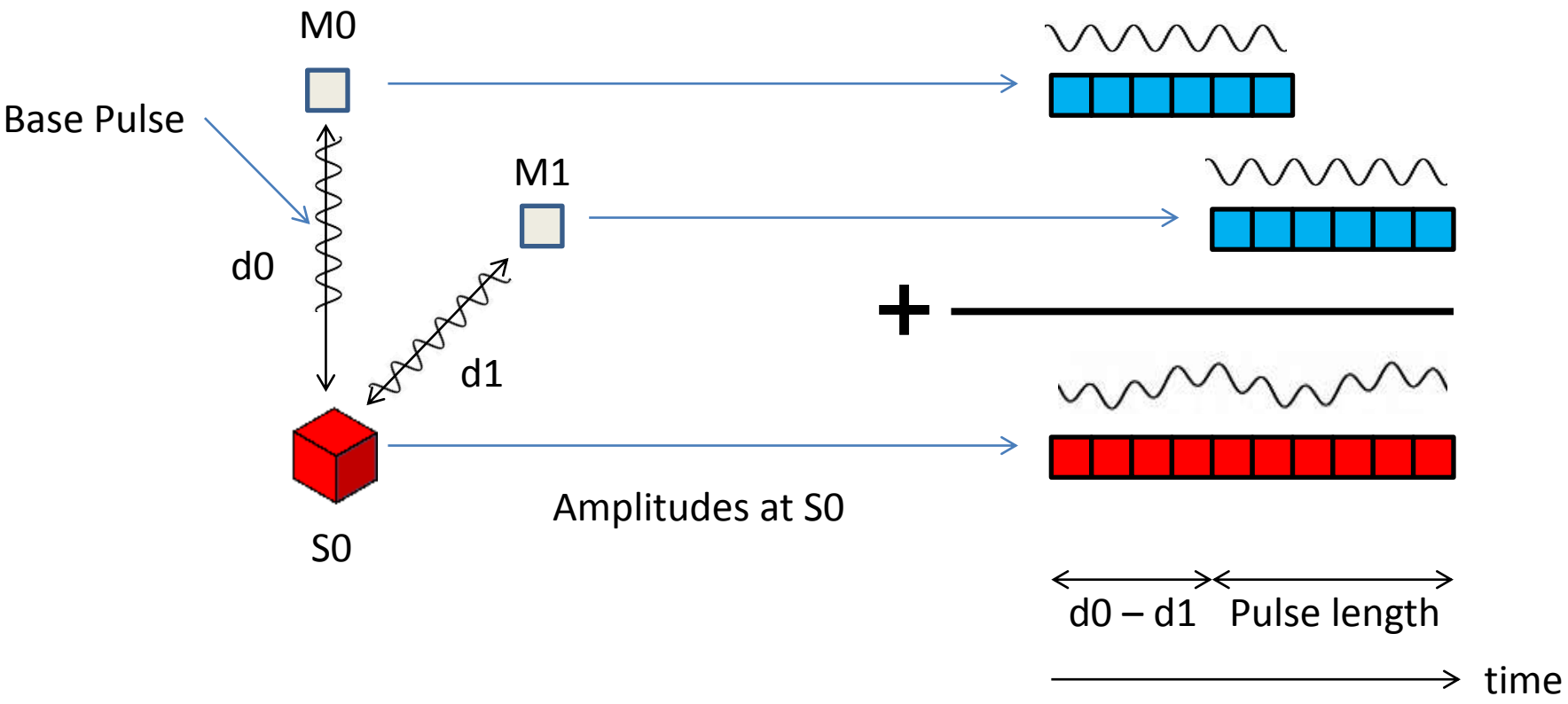
Modelling Ultrasound



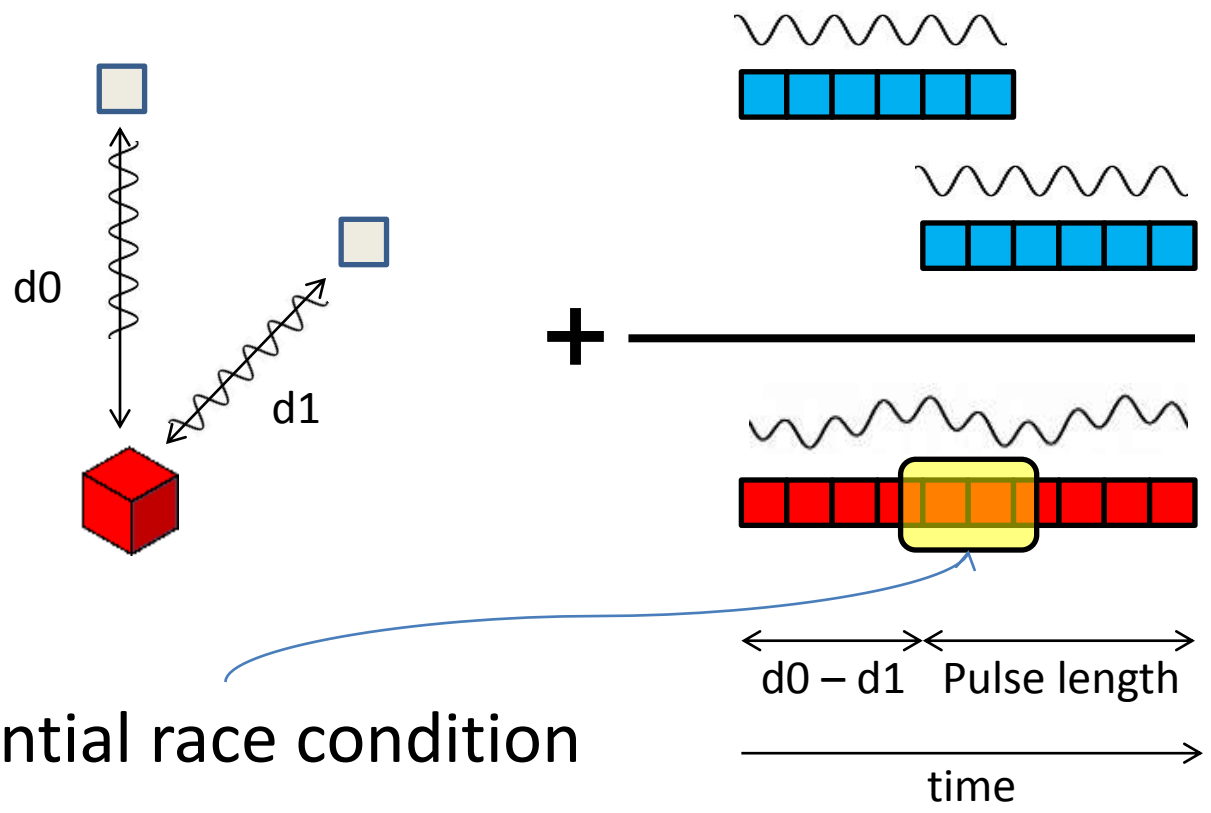
Modelling Ultrasound



Modelling Ultrasound



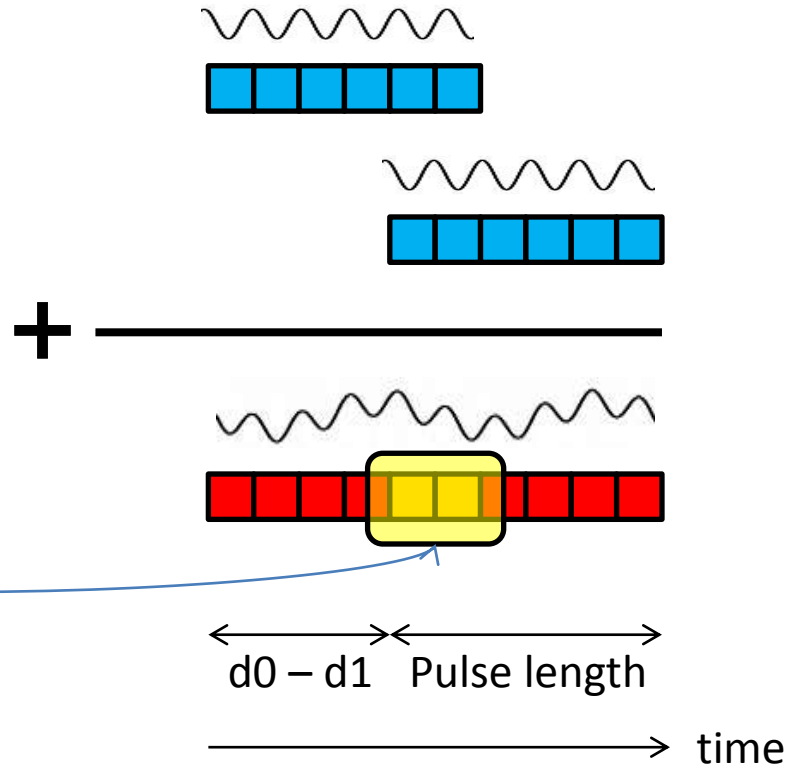
Modelling Ultrasound



Potential race condition

Modelling Ultrasound

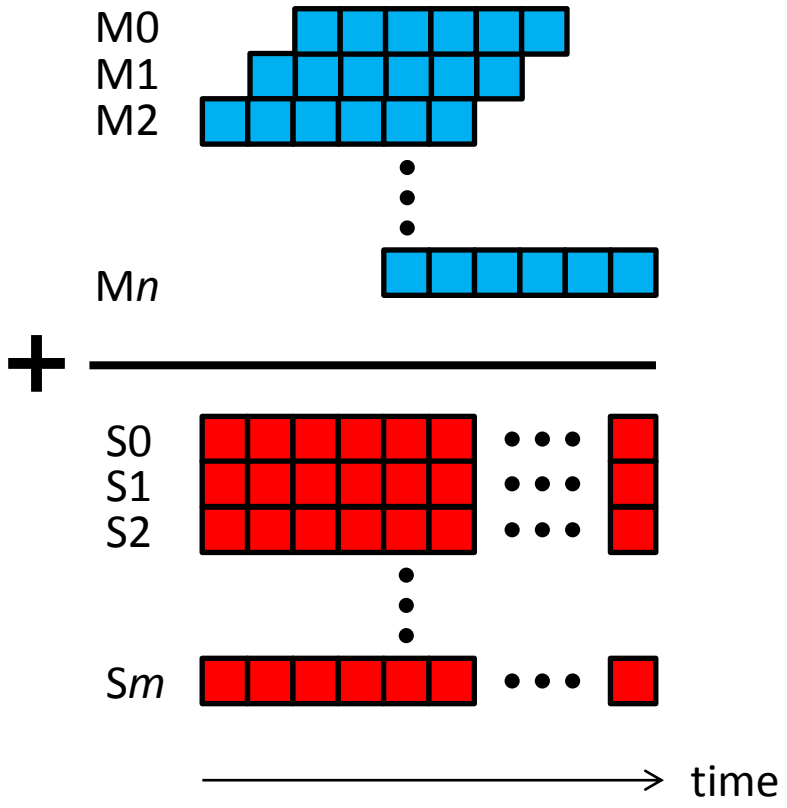
Potential race condition



Modelling Ultrasound

Many

Potential race conditions!

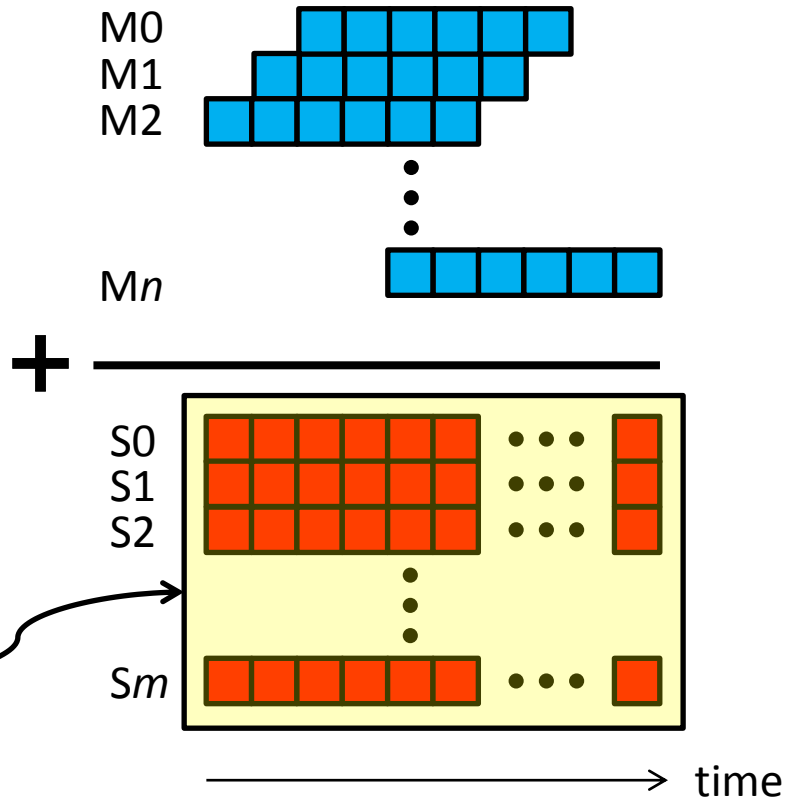


Modelling Ultrasound

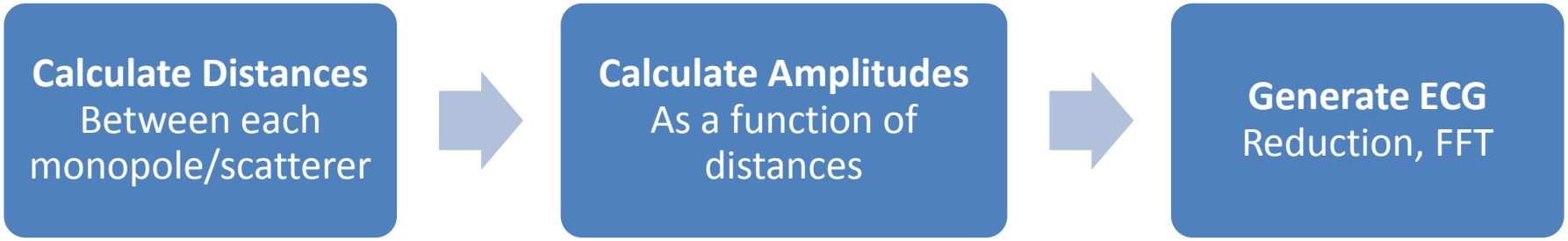
Many

Potential race conditions!

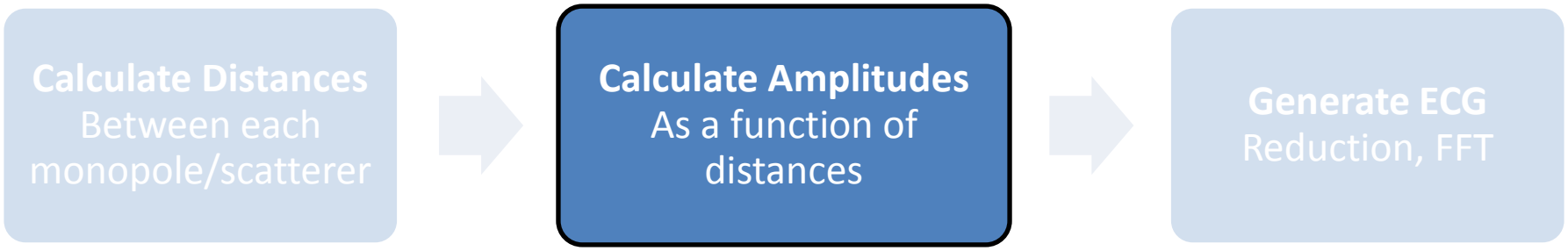
Amplitude Accumulator



System Overview



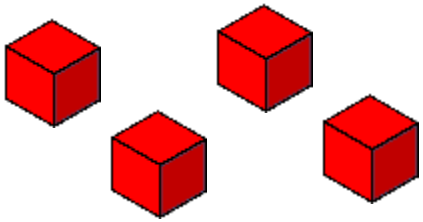
System Overview



Computationally
Expensive!

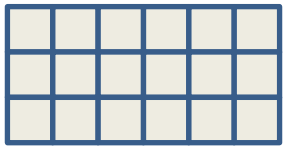
How Expensive?

Acoustic Scatterers



10^4

Acoustic Monopoles



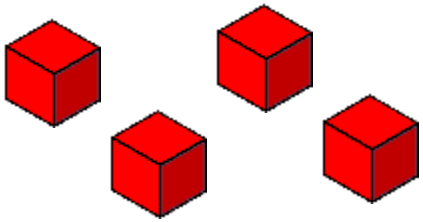
10^3

Base Pulse

10^2

How Expensive?

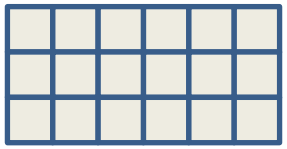
Acoustic Scatterers



10^4

×

Acoustic Monopoles



10^3

×

Base Pulse

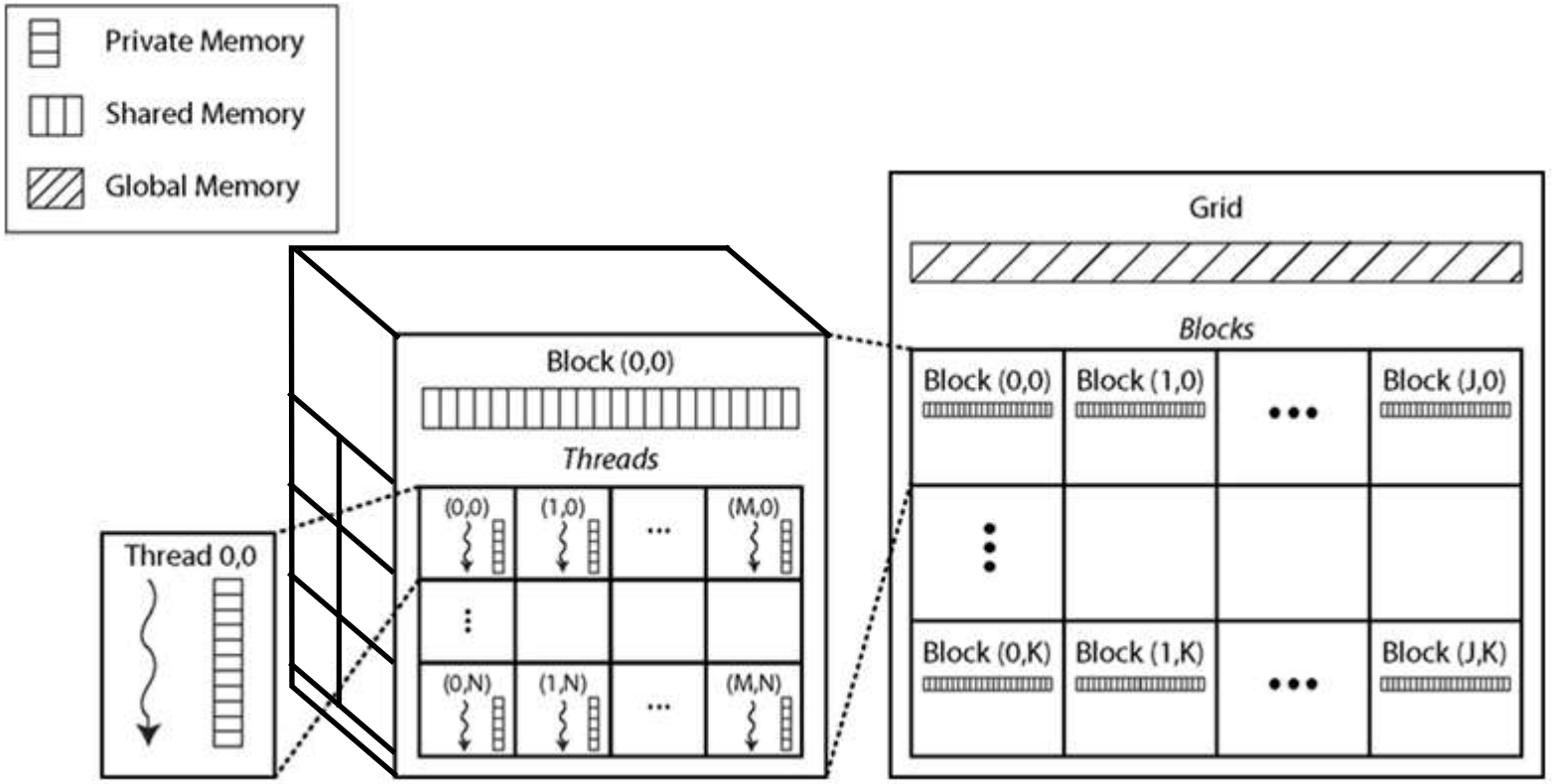
10^2

= **10^9 calculations *per frame***

Higher values achieve greater accuracy

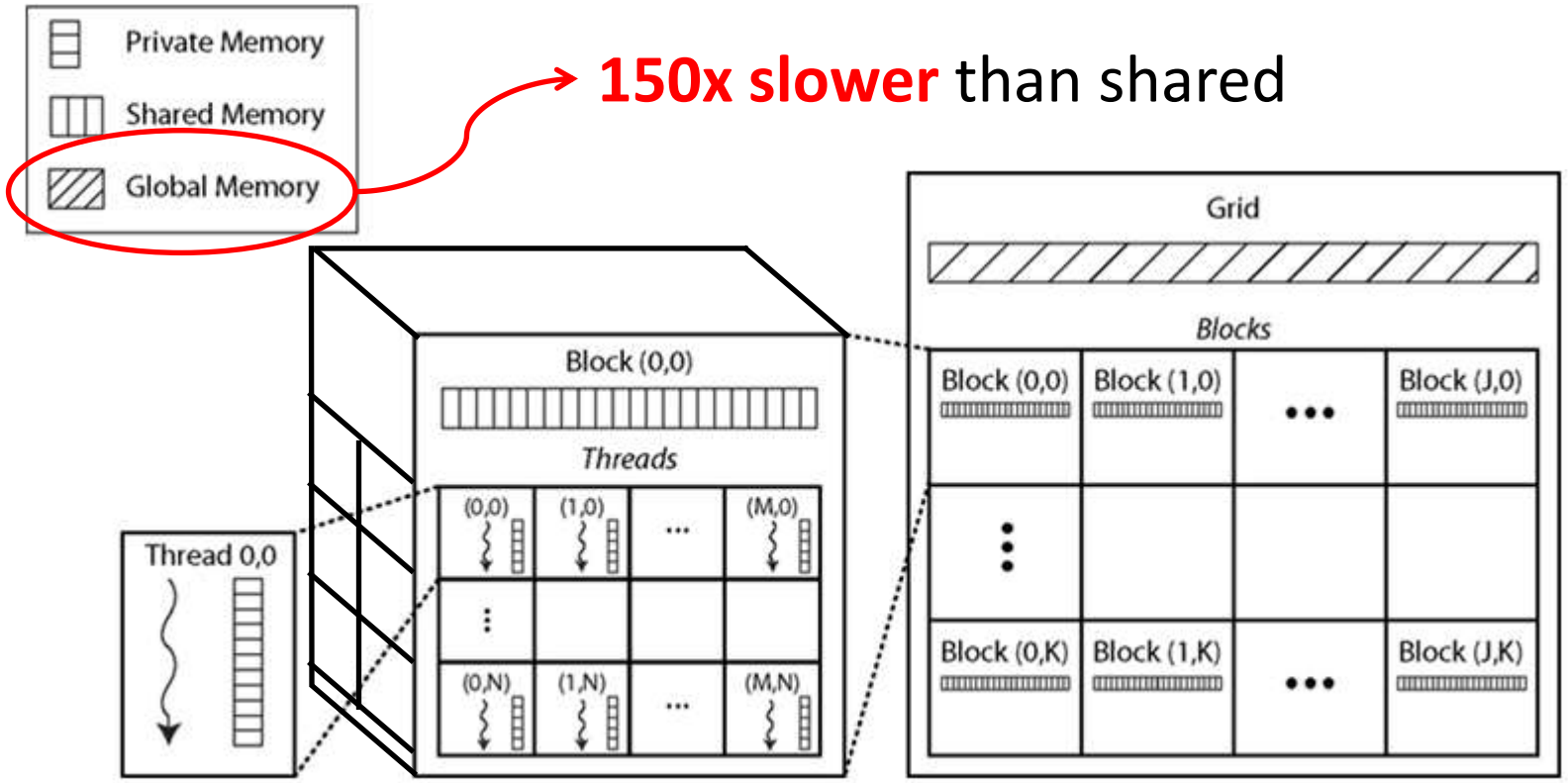
CUDA Architecture

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



CUDA Architecture

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



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 x 4 x 3 = 60 possible execution configurations

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

Implementations

- CPU
  - Single-threaded
  - Multi-threaded
- GPU
  - Loop
  - Reduction

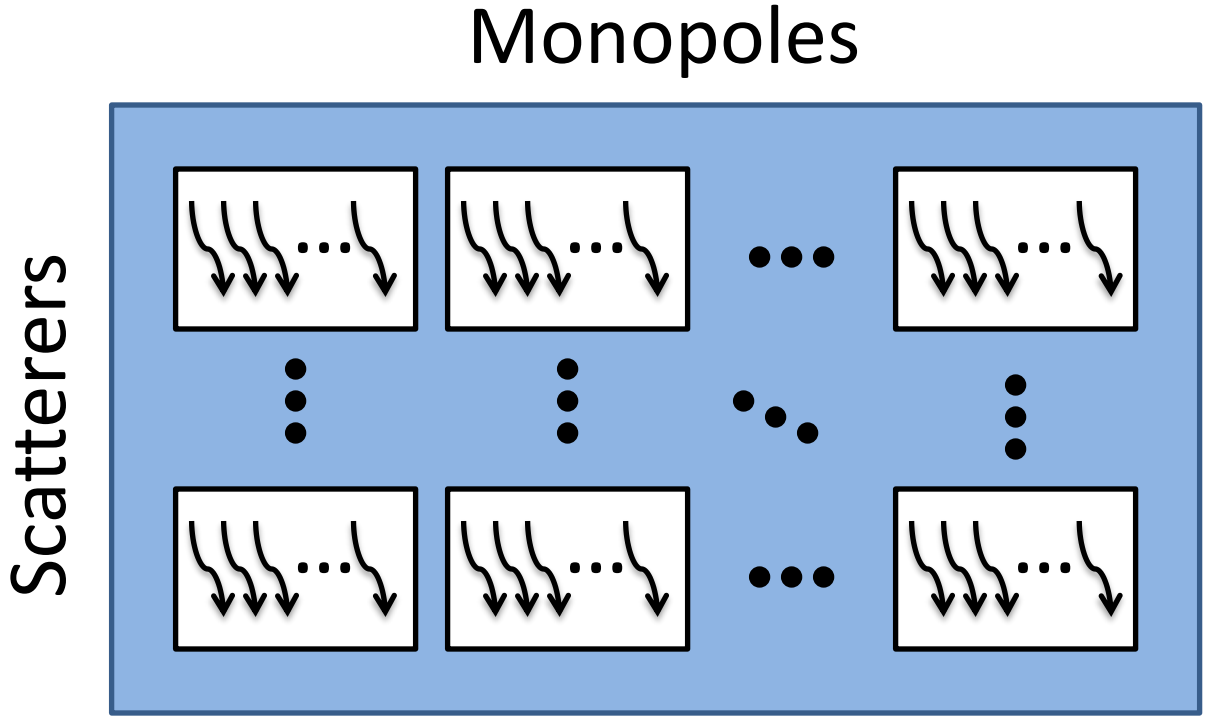
CPU Implementation (Single-Threaded)

```
1 for each monopole
2   for each scatterer
3     for each pulse entry
4       calculate contribution
5       add to accumulator
```

CPU Implementation (Multi-Threaded)

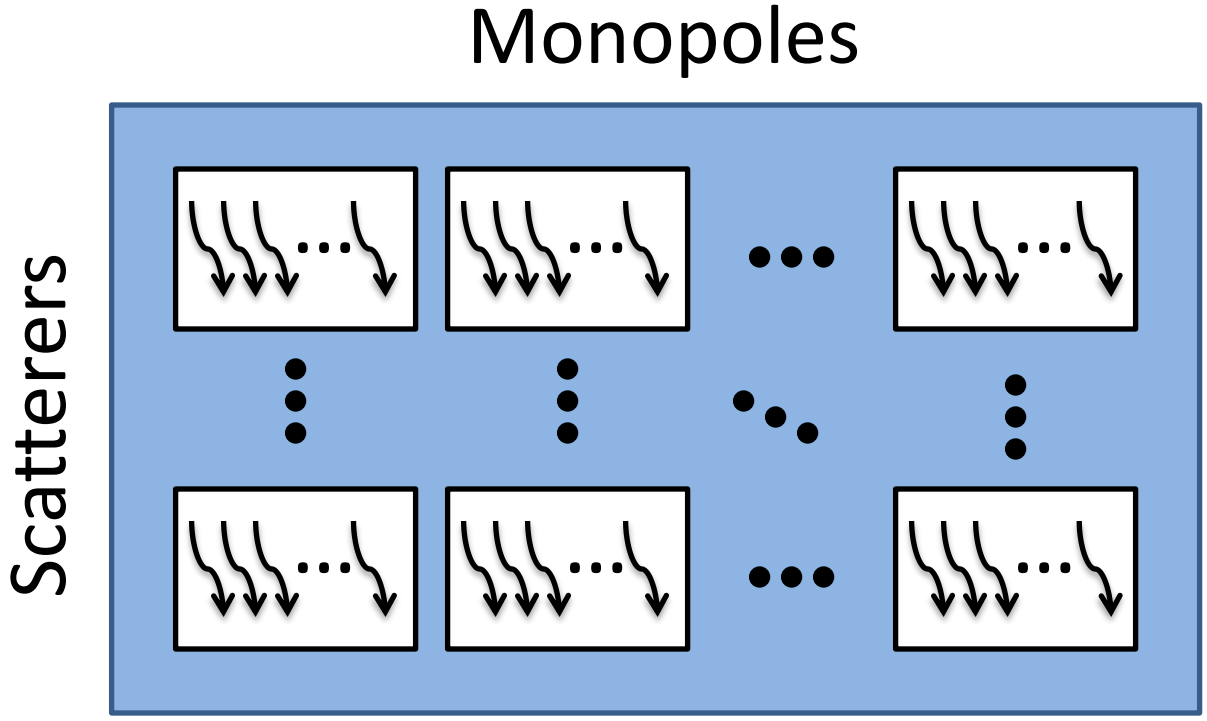
```
0 #pragma omp parallel for
1 for each monopole
2   for each scatterer
3     for each pulse entry
4       calculate contribution
5       add to accumulator
```

GPU Base Design



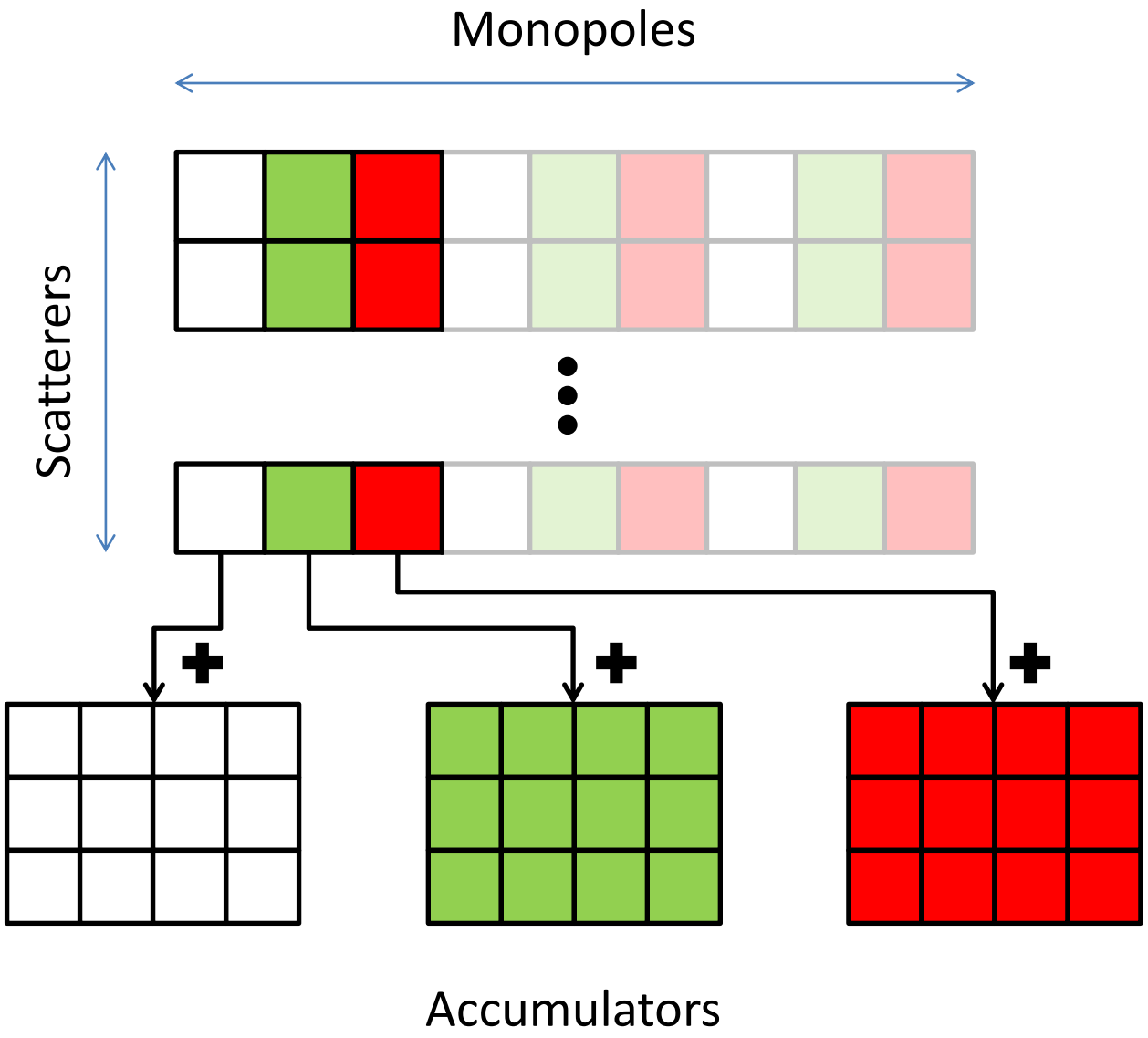
- 1 Block = 1 pulse
- 1 Thread = 1 entry in pulse
- Position in grid corresponds to monopole/scatterer pair

GPU Base Design

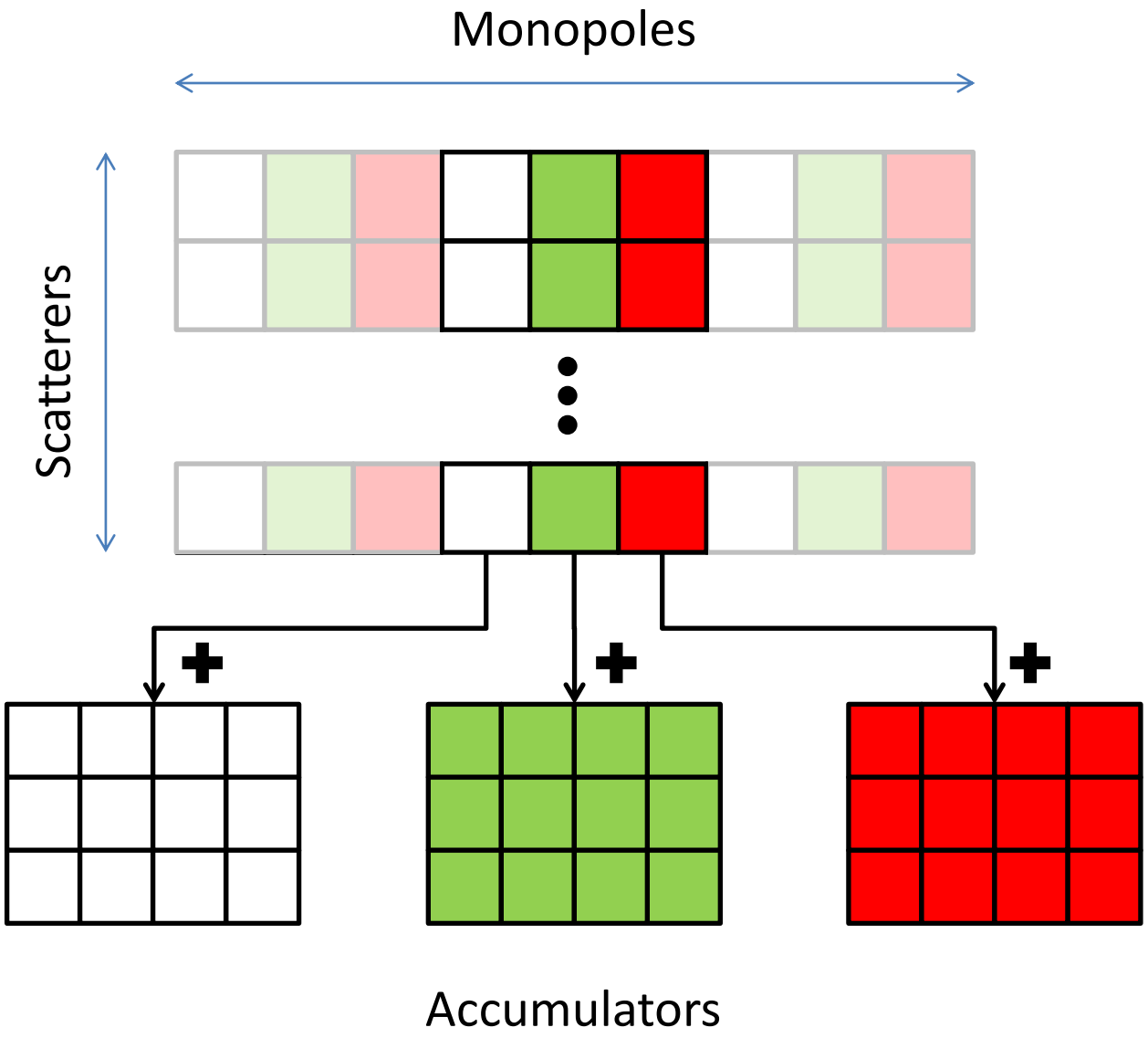


- Does not work due to a WAW dependence
  - Multiple accumulator elements may be written to simultaneously by different blocks

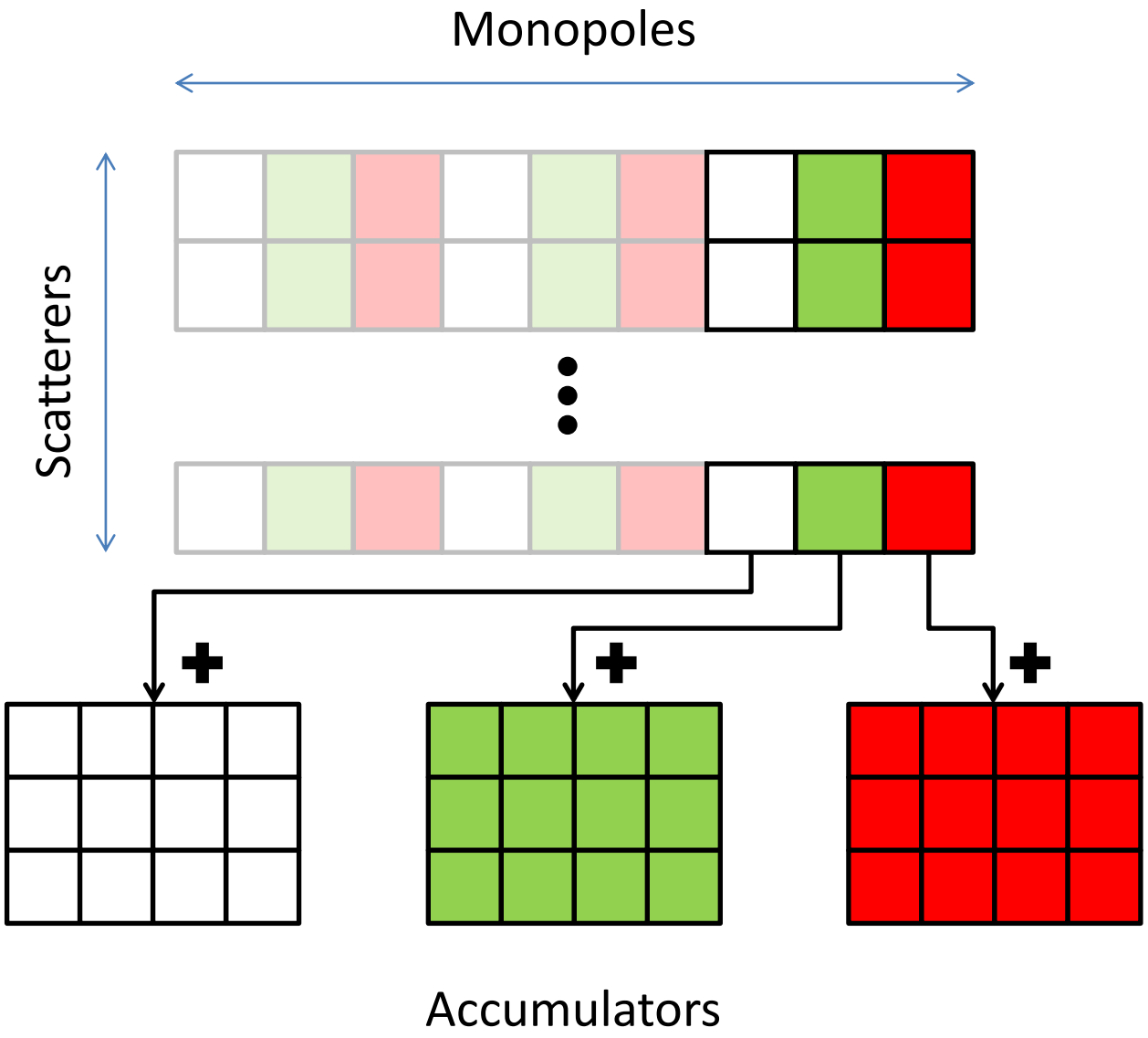
GPU Reduction



GPU Reduction

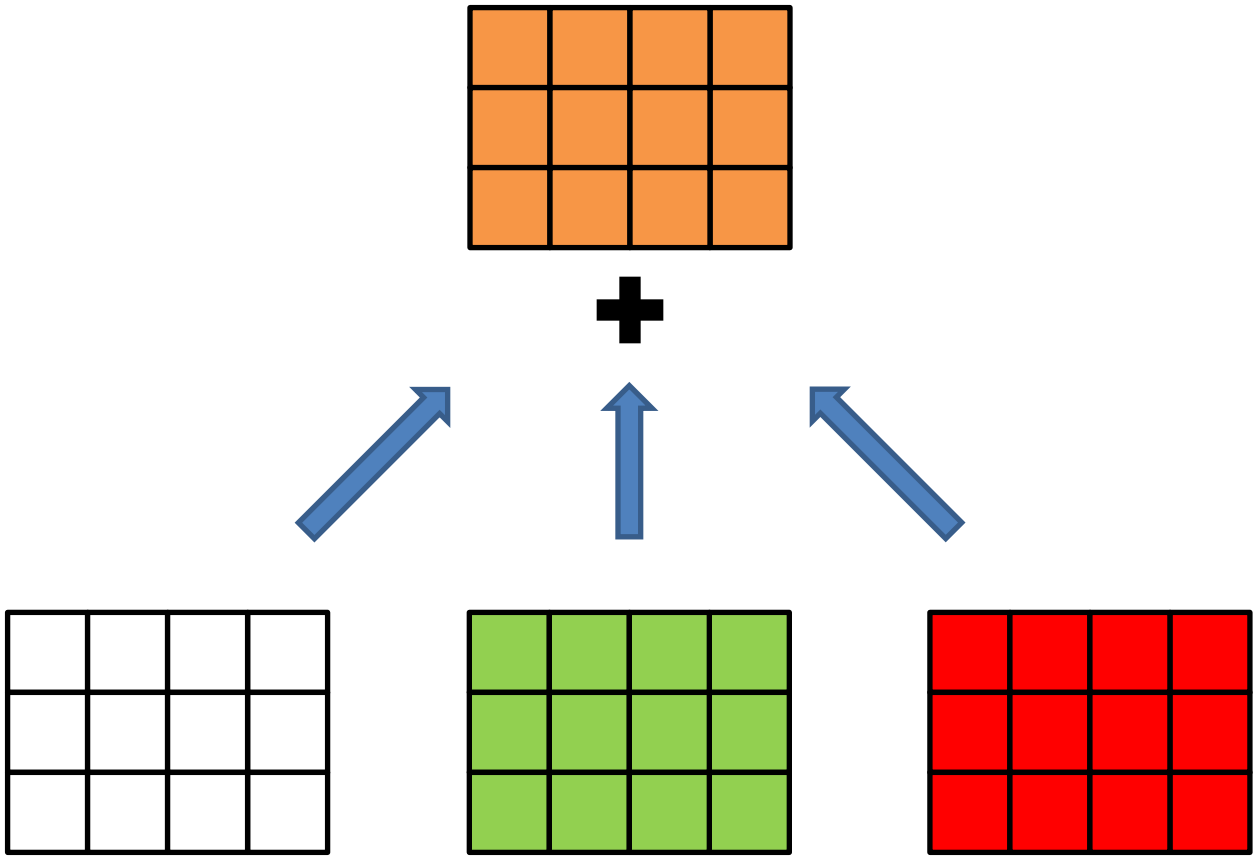


GPU Reduction



GPU Reduction

Reduction summation



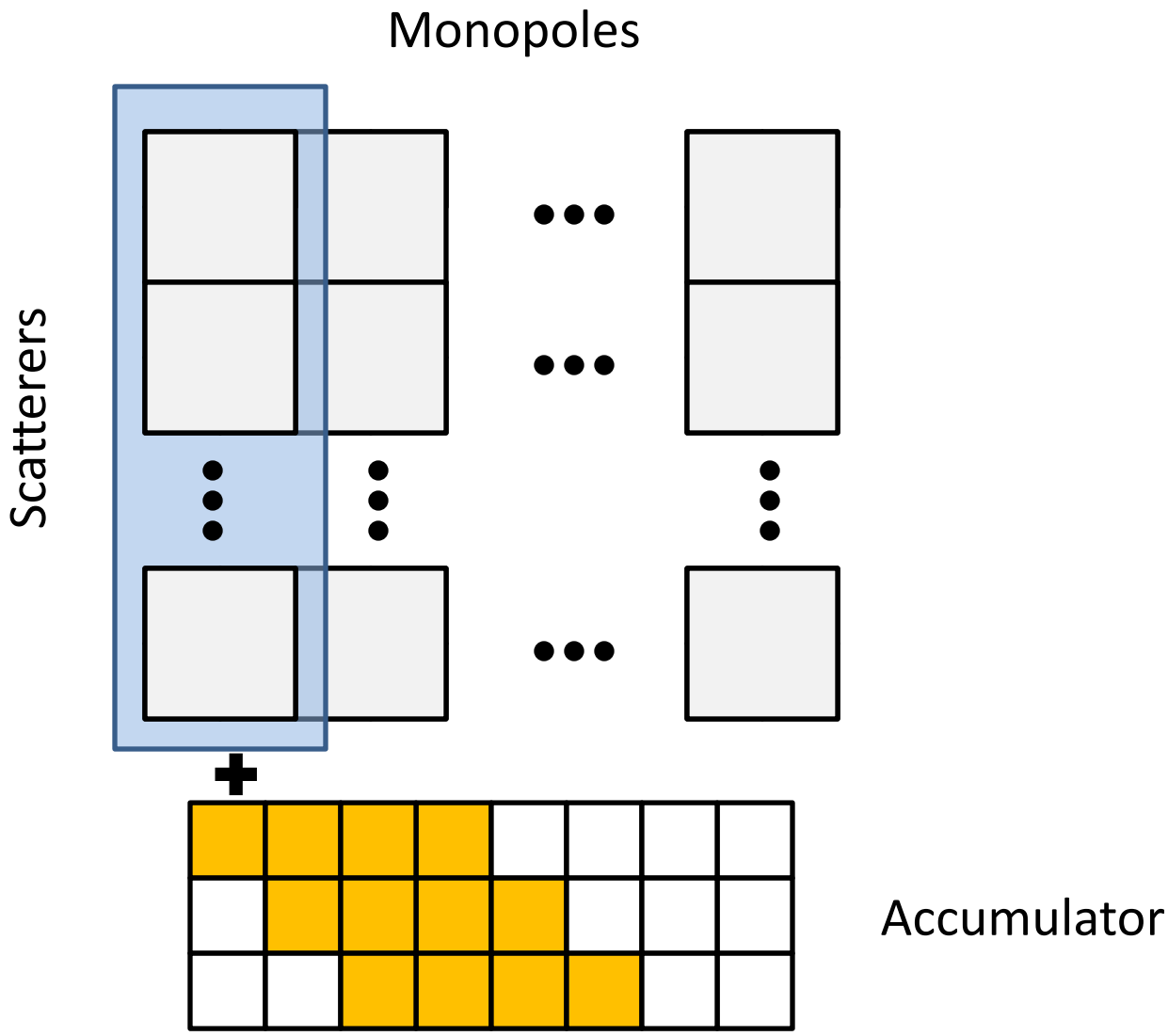
Accumulators

GPU Reduction

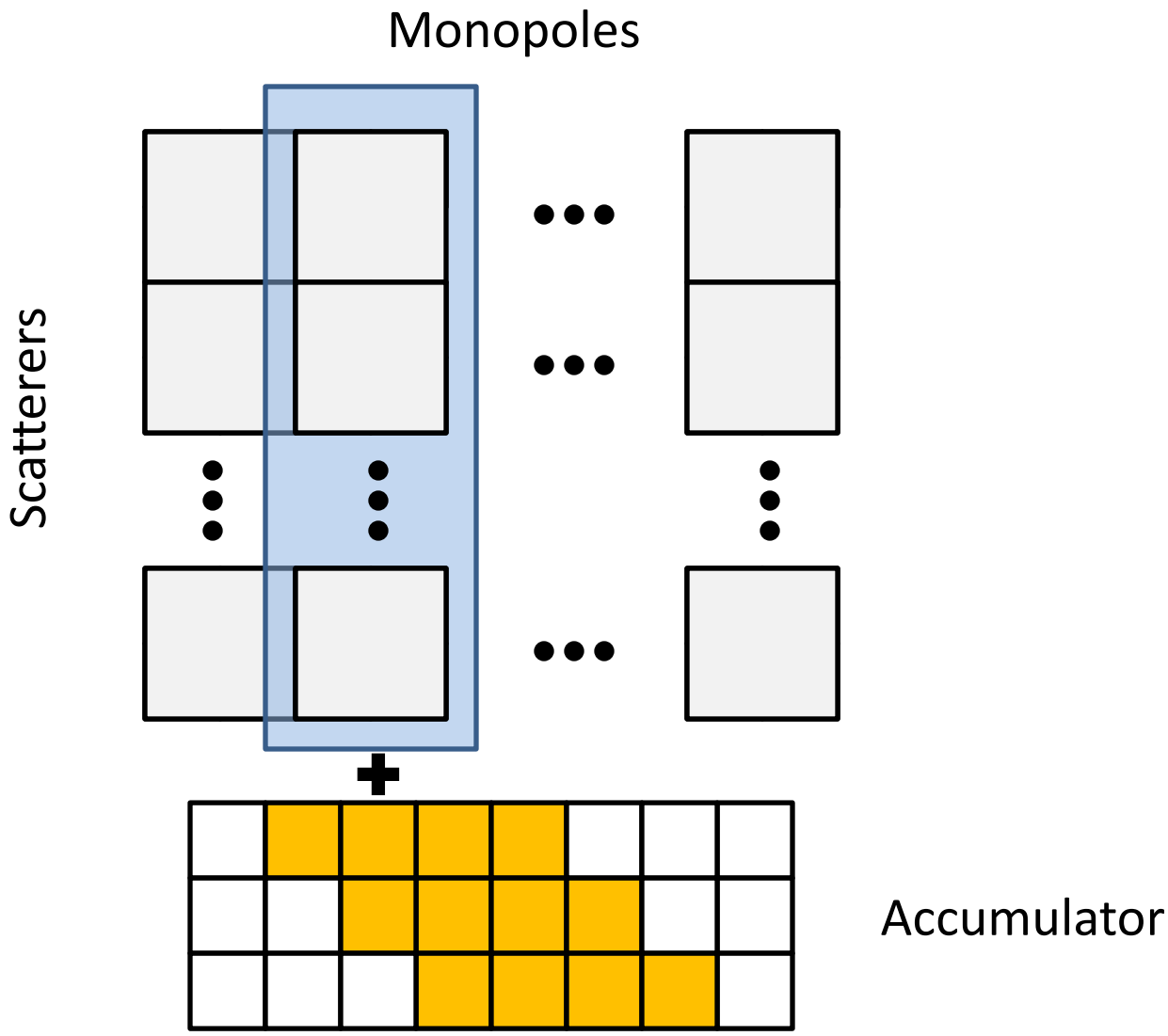
- Each block has its own copy of data to read
  - Prevents thread read competition (serialization)
- Dynamically optimize grid size based on available memory

- Use shared memory where possible
- Use barriers to synchronize threads

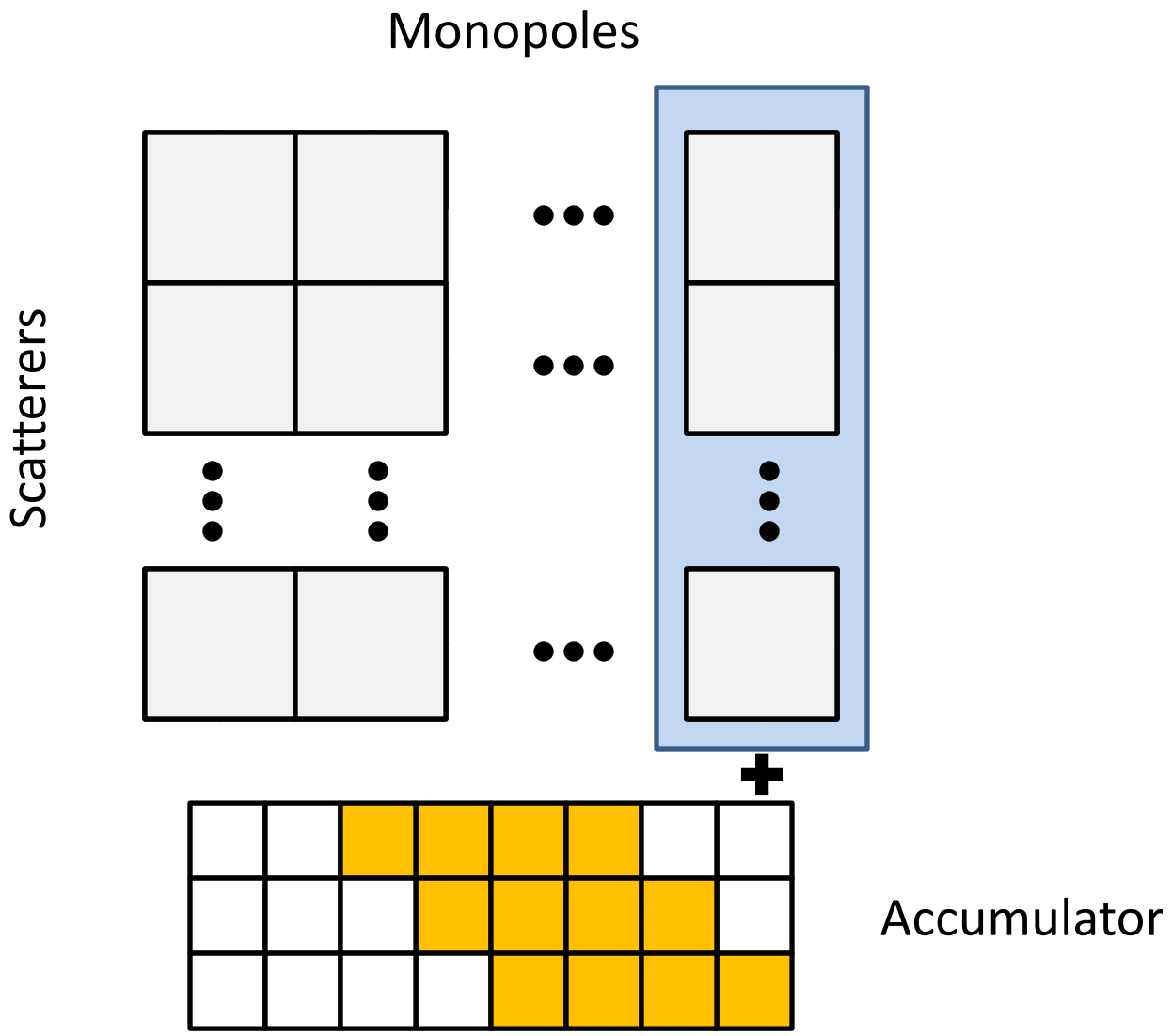
GPU Loop



GPU Loop



GPU Loop



GPU Loop

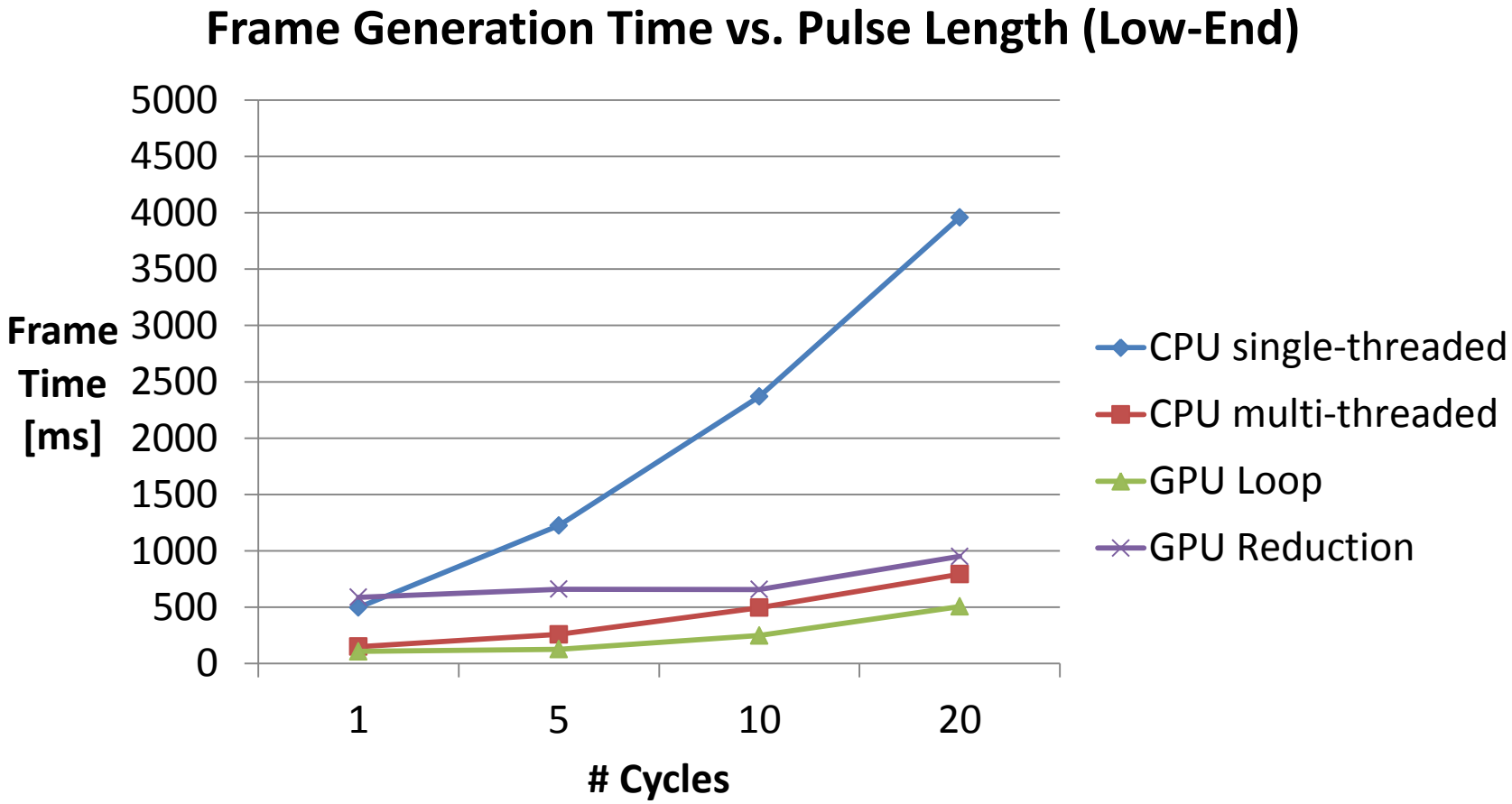
- Loop through monopoles inside the kernel
  - Reduces kernel launching overhead
- Implement other code optimizations
  - Loop invariant code motion

- Use shared memory where possible
- Use barriers to synchronize threads

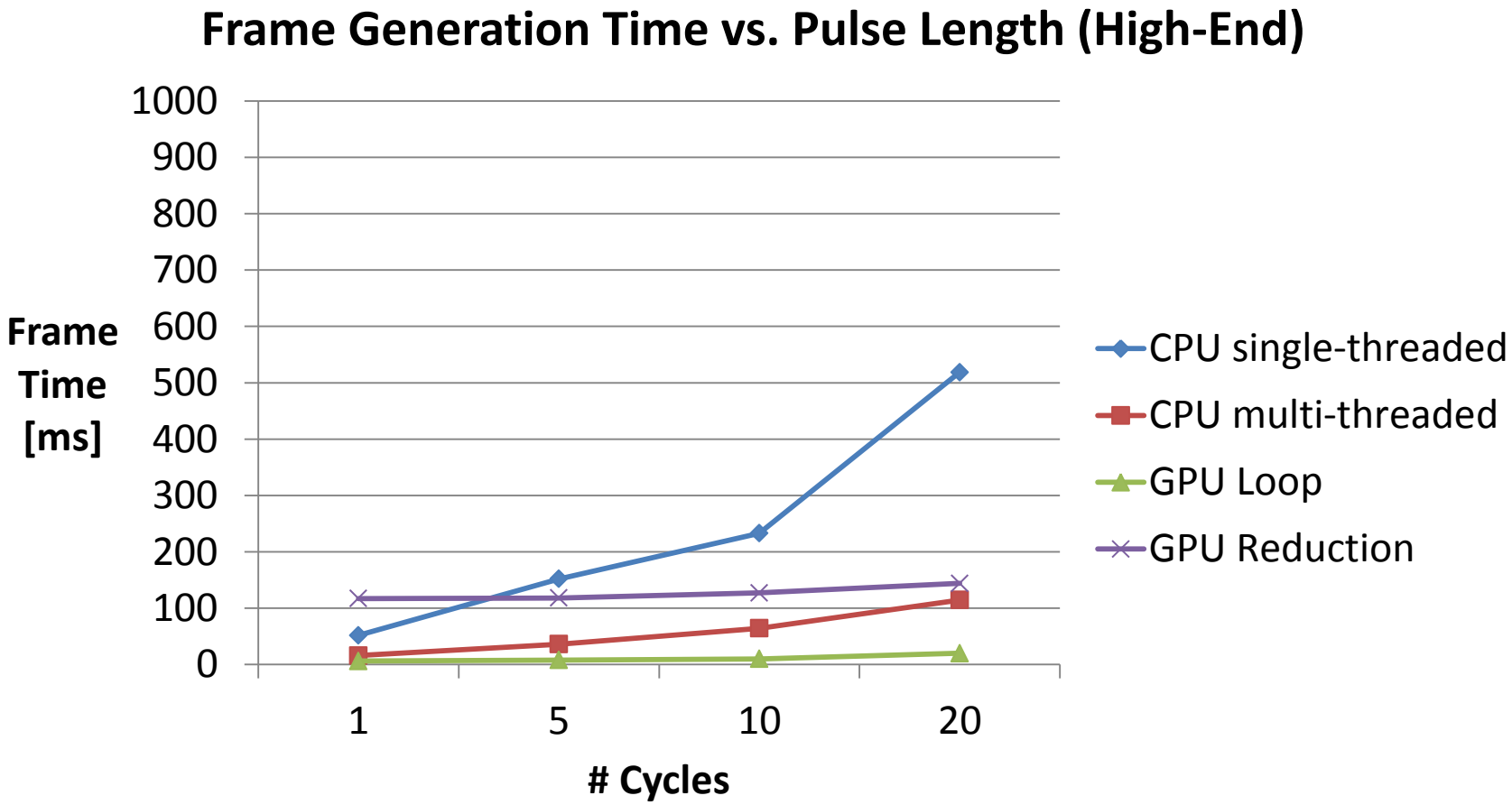
Testbed

	Low End		High End	
	CPU	**GPU**	**CPU**	**GPU**
Name	Atom D525	NVIDIA Ion 2	Core i7-2630QM	NVIDIA GeForce GTX470
Speed	1.8GHz (HT)	1.09GHz	2GHz (HT)	1.215GHz
Cores	2	16	4	448
Memory	2GB	0.43GB	8GB	1.25GB

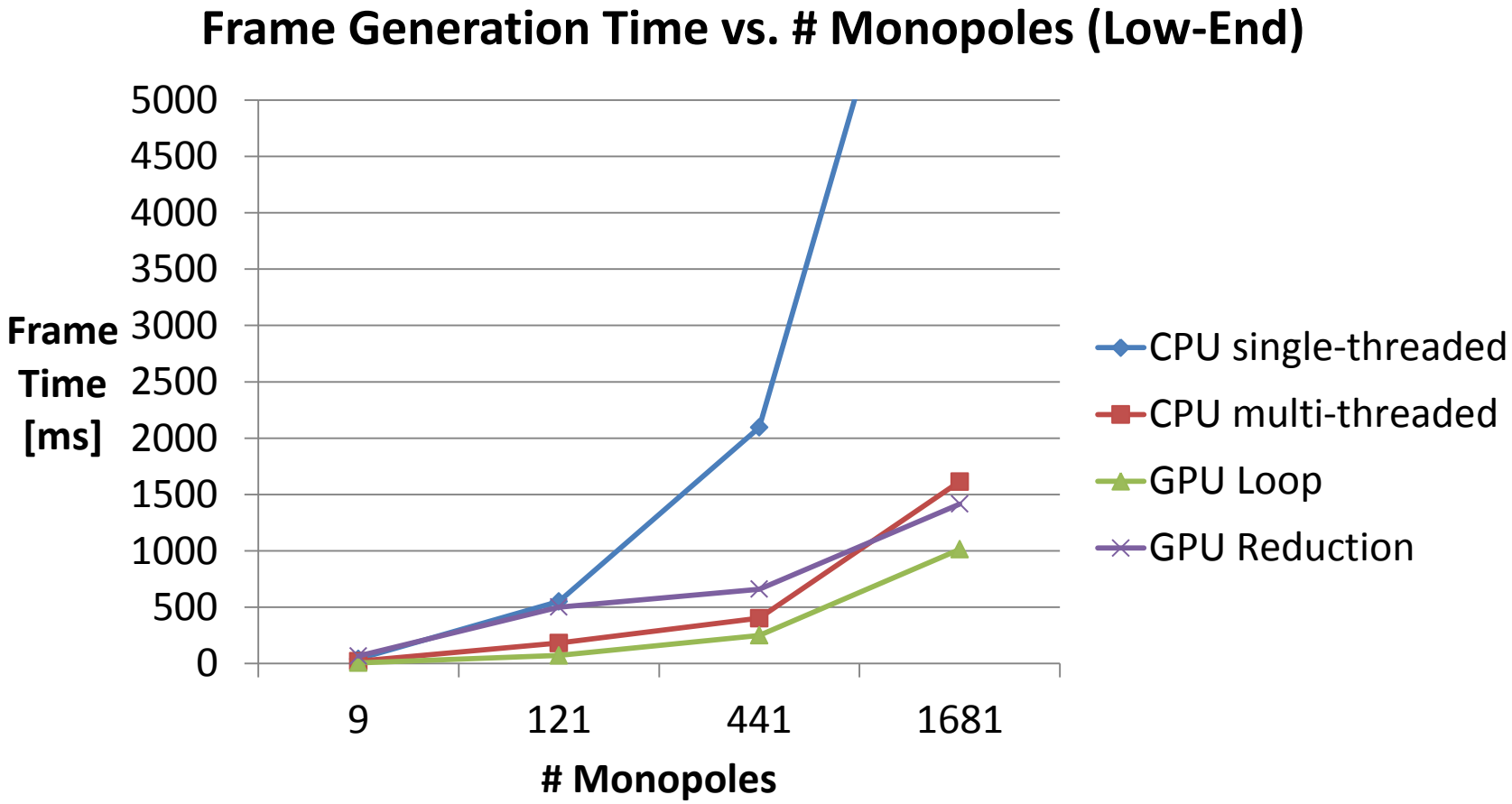
Varying Pulse Length (Low End)



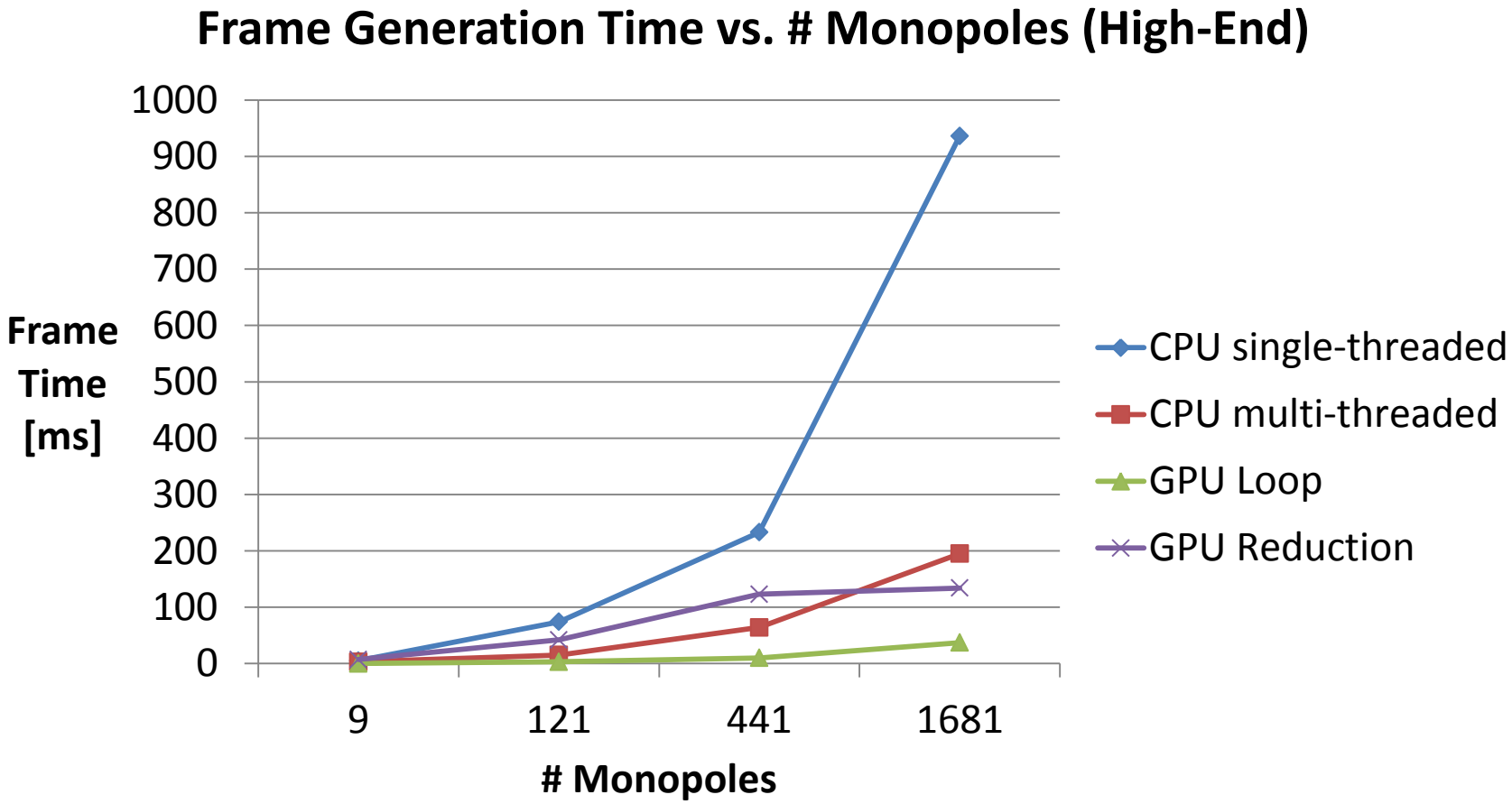
Varying Pulse Length (High End)



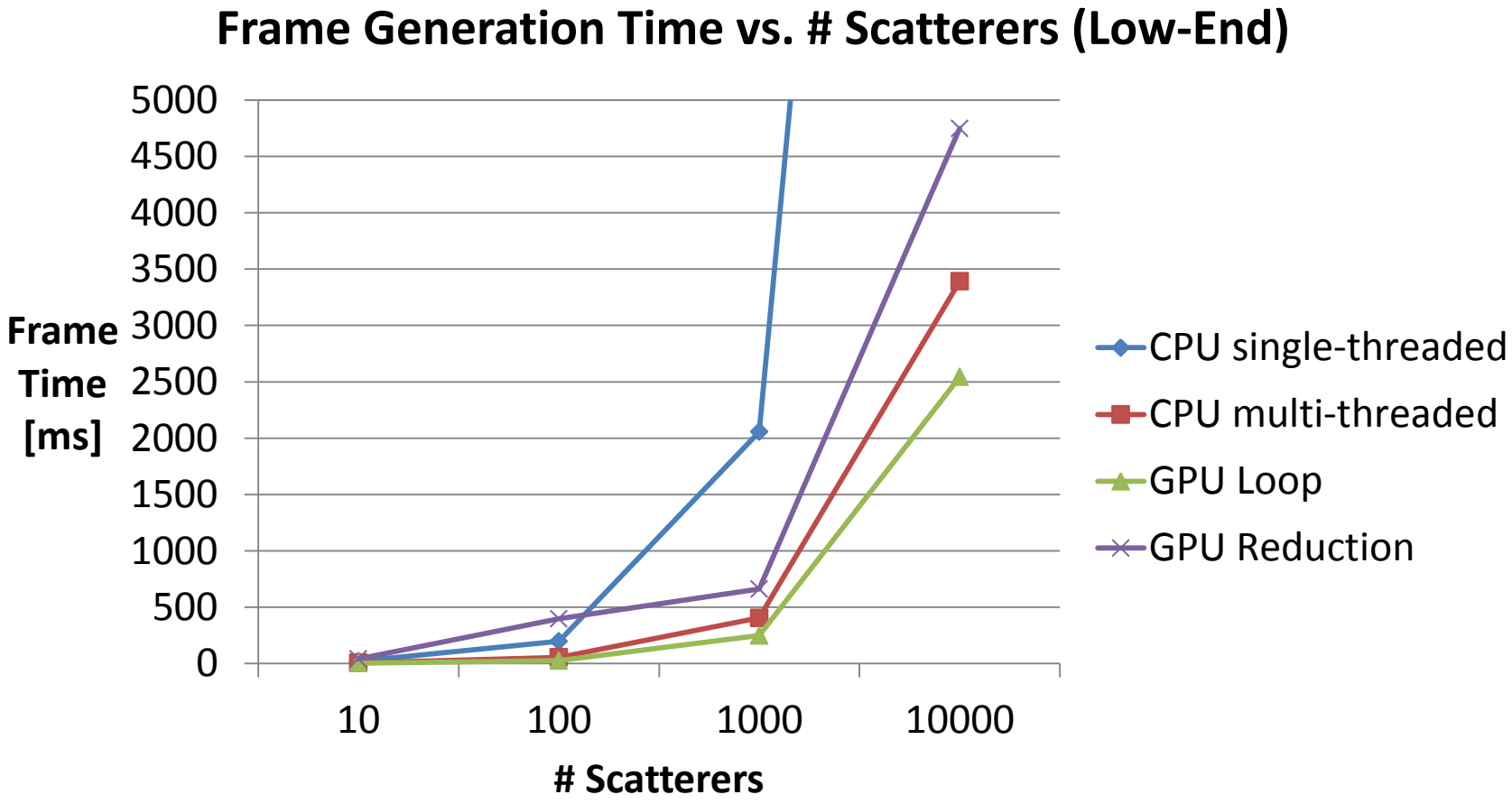
Varying # Monopoles (Low End)



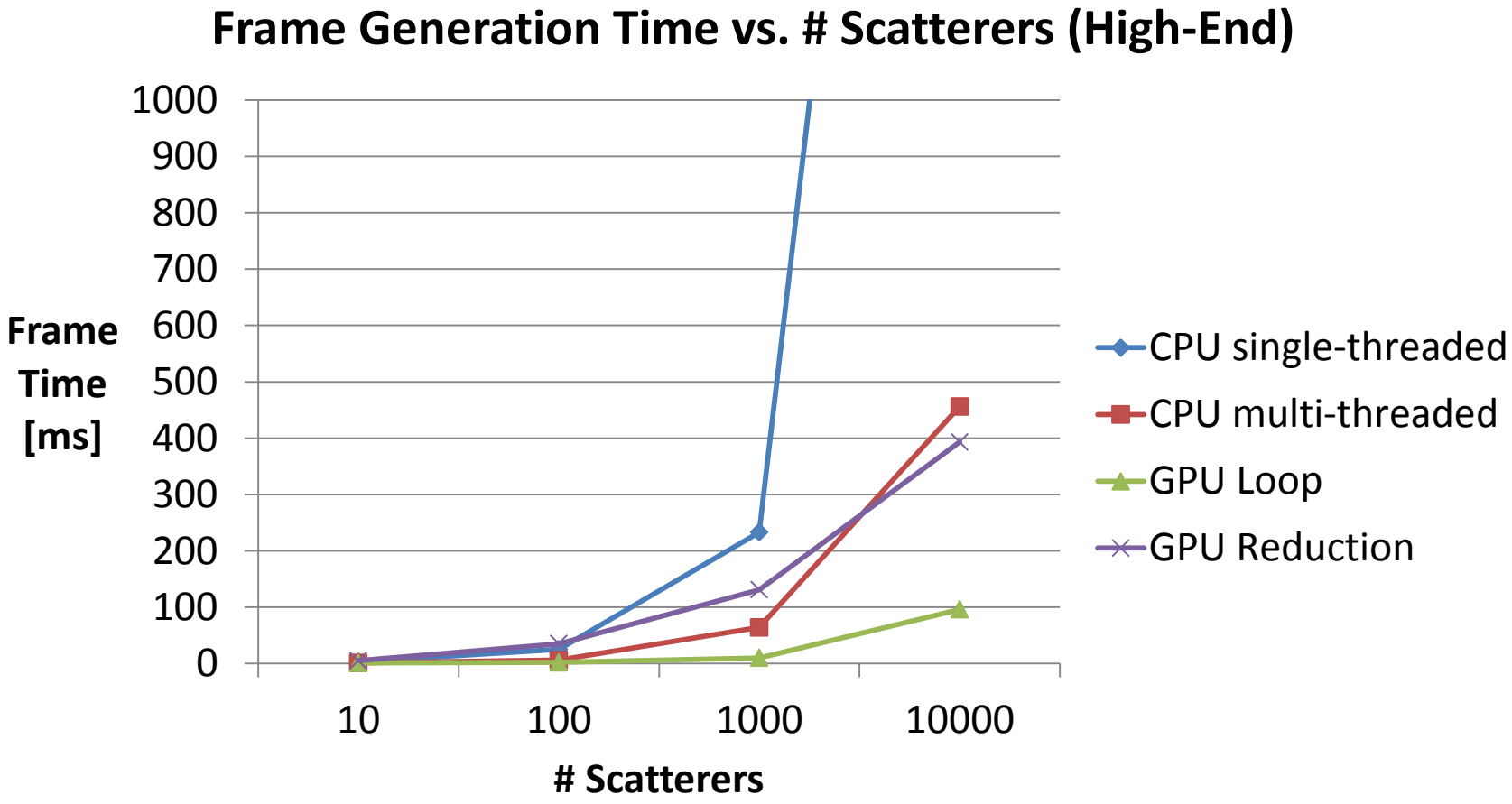
Varying # Monopoles (High End)

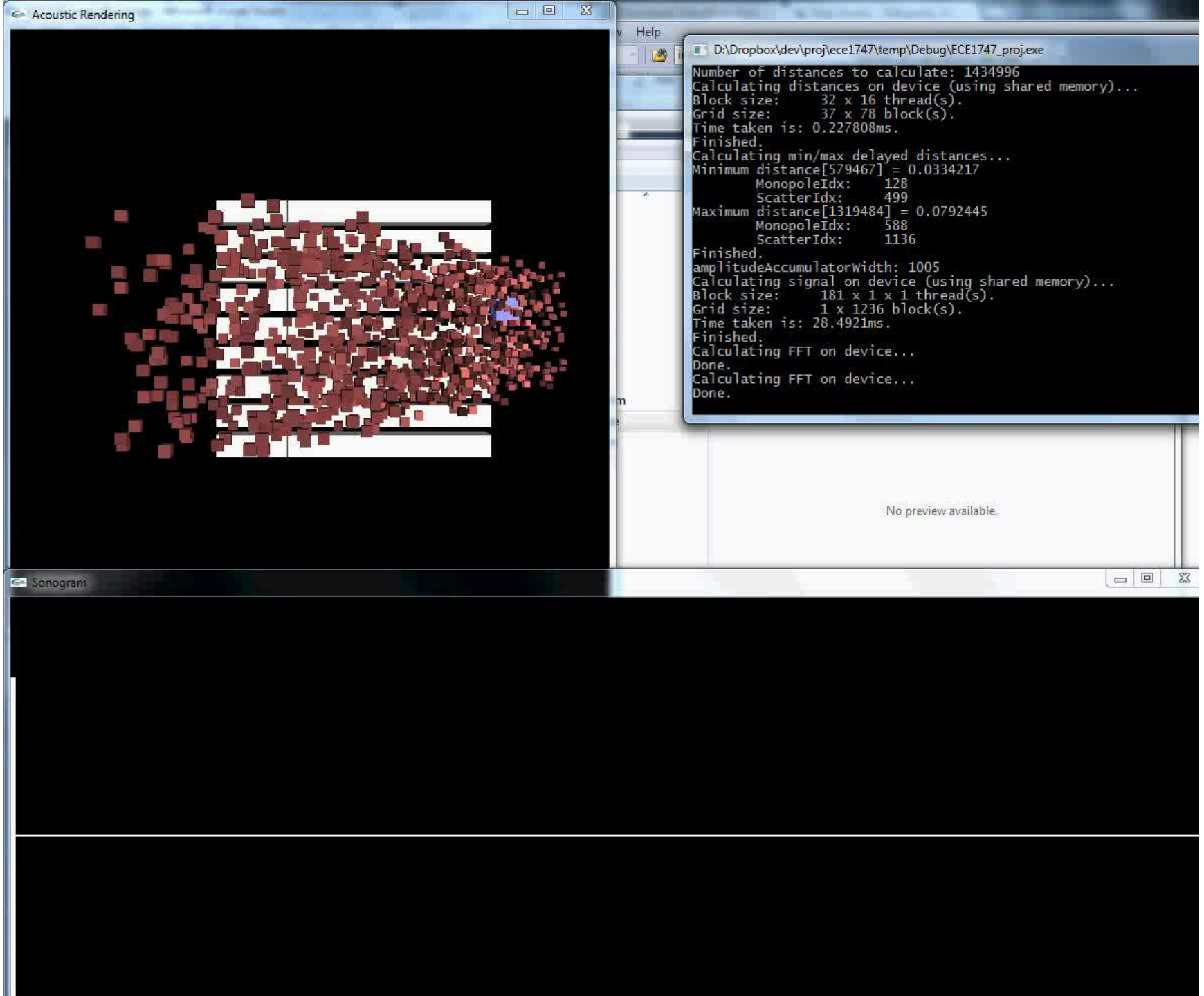


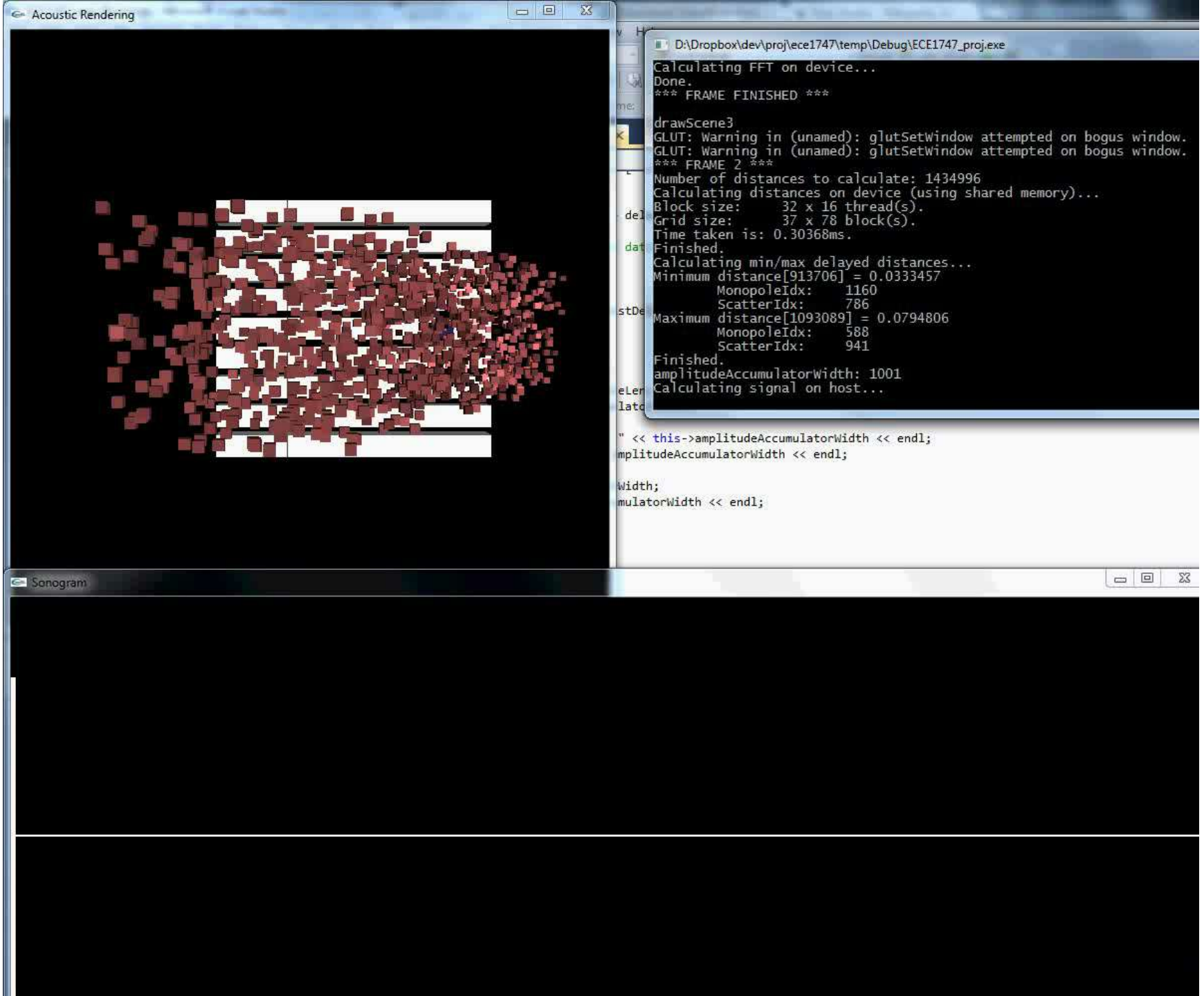
Varying # Scatterers (Low End)



Varying # Scatterers (High End)




Acoustic Rendering
Help
D:\Dropbox\dev\proj\ece1747\temp\Debug\ECE1747_proj.exe
Number of distances to calculate: 1434996
Calculating distances on device (using shared memory)...
Block size: 32 x 16 thread(s).
Grid size: 37 x 78 block(s).
Time taken is: 0.227808ms.
Finished.
Calculating min/max delayed distances...
Minimum distance[579467] = 0.0334217
MonopoleIdx: 128
ScatterIdx: 499
Maximum distance[1319484] = 0.0792445
MonopoleIdx: 588
ScatterIdx: 1136
Finished.
amplitudeAccumulatorWidth: 1005
Calculating signal on device (using shared memory)...
Block size: 181 x 1 x 1 thread(s).
Grid size: 1 x 1236 block(s).
Time taken is: 28.4921ms.
Finished.
Calculating FFT on device...
Done.
Calculating FFT on device...
Done.
No preview available.
Sonogram


Acoustic Rendering
D:\Dropbox\dev\proj\ece1747\temp\Debug\ECE1747_proj.exe
Calculating FFT on device...
Done.
*** FRAME FINISHED ***
drawScene3
GLUT: Warning in (unamed): glutSetWindow attempted on bogus window.
GLUT: Warning in (unamed): glutSetWindow attempted on bogus window.
*** FRAME 2 ***
Number of distances to calculate: 1434996
Calculating distances on device (using shared memory)...
Block size: 32 x 16 thread(s).
Grid size: 37 x 78 block(s).
Time taken is: 0.30368ms.
Finished.
Calculating min/max delayed distances...
Minimum distance[913706] = 0.0333457
MonopoleIdx: 1160
ScatterIdx: 786
Maximum distance[1093089] = 0.0794806
MonopoleIdx: 588
ScatterIdx: 941
Finished.
amplitudeAccumulatorWidth: 1001
Calculating signal on host...
" << this->amplitudeAccumulatorWidth << endl;
mplitudeAccumulatorWidth << endl;
Width;
mulatorWidth << endl;
Sonogram

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

Algorithm	Use Case	Real-Life Implication
GPU Looping	Small workloads	Good for real-time applications
GPU Reduction	Large workloads	Good for high-accuracy applications
CPU multi-threaded	GPU unavailable	Simple to implement

Conclusion

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

Algorithm	Use Case	Real-Life Implication
GPU Looping	Small workloads	Good for real-time applications
GPU Reduction	Large workloads	Good for high-accuracy applications
CPU multi-threaded	GPU unavailable	Simple to implement
CPU single-threaded	**1985**	**Back to the Future**

Thank You