Faster Finite Elements for Wave Propagation Codes

Max Rietmann
Institute for Computational Science
USI Lugano, Switzerland

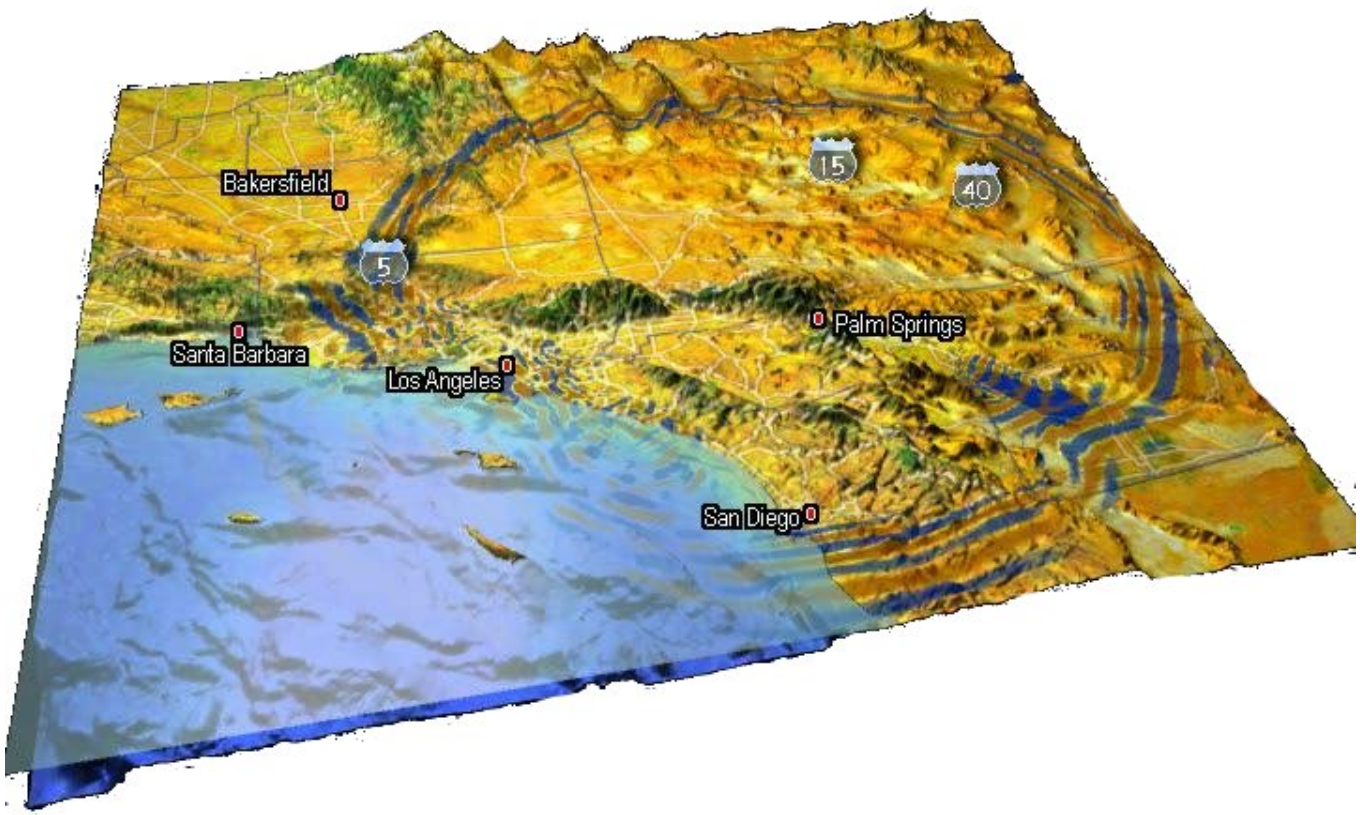
May 17, 2012

Computational Seismology

- Large Field, many focuses, many ways to use supercomputing resources.

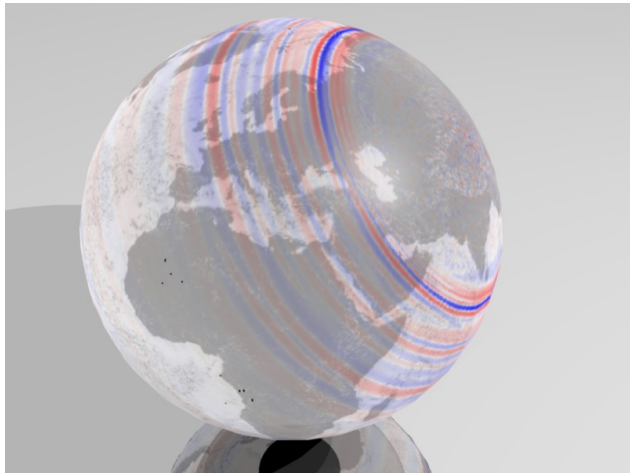
Our focus:

- Simulate earthquakes on global and regional scale.

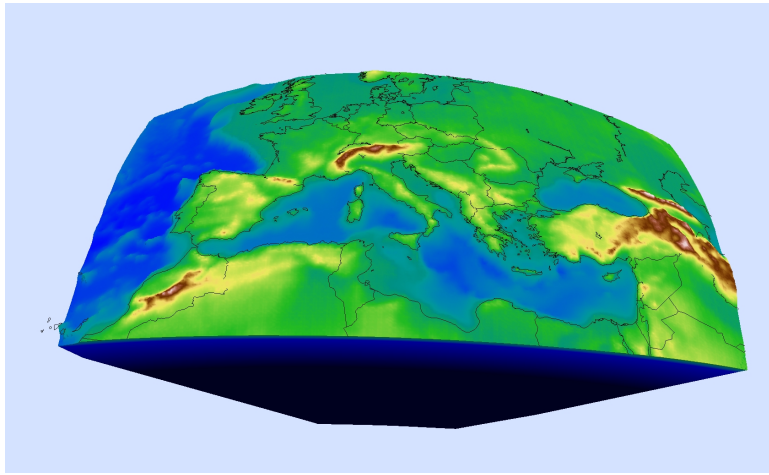


SPECFEM3D

SPECFEM3D split into two packages:



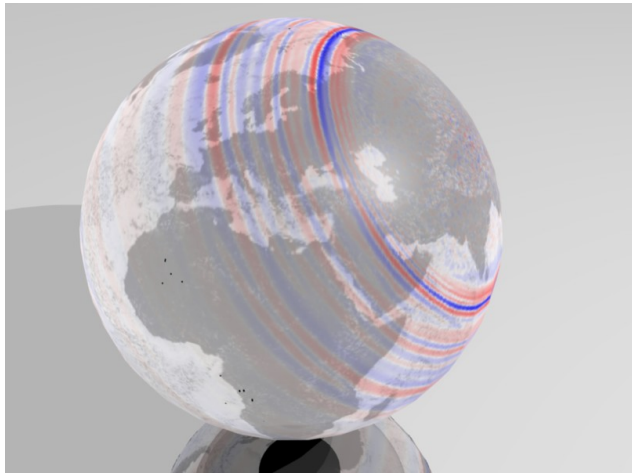
SPECFEM3D_GLOBE



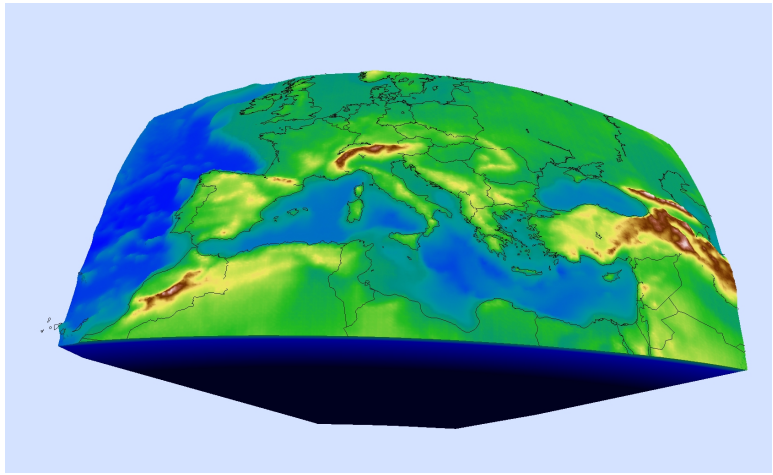
SPECFEM3D 2.0 “Sesame”

SPECFEM3D

SPECFEM3D split into two packages:



SPECFEM3D_GLOBE



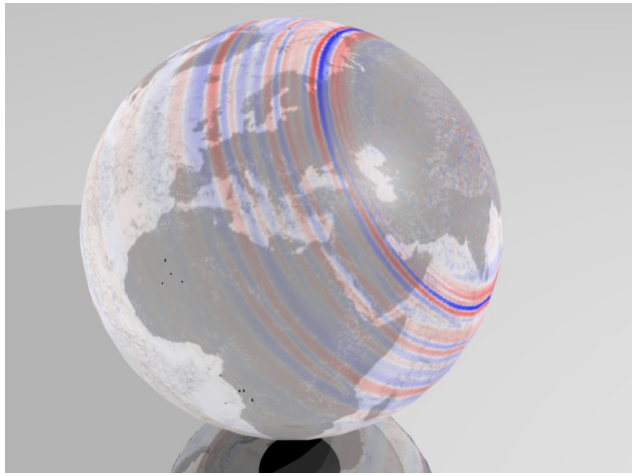
SPECFEM3D 2.0 “Sesame”

- GLOBE is optimized for global scale seismology. Accounts for gravity and the spin of the earth. Mesh is fixed and predetermined.

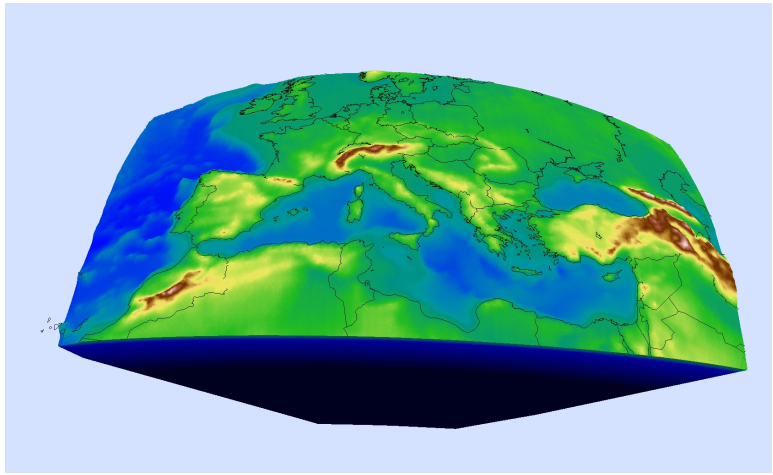
[1]: http://geodynamics.org/cig/software/specfem3d

SPECFEM3D

SPECFEM3D split into two packages:



SPECFEM3D_GLOBE



SPECFEM3D 2.0 “Sesame”

- GLOBE is optimized for global scale seismology. Accounts for gravity and the spin of the earth. Mesh is fixed and predetermined.
- SPECFEM3D Sesame can use arbitrary user-defined hexahedral meshes.

[1]: http://geodynamics.org/cig/software/specfem3d

Forward wave propagation

10 years ago, classic "forward" simulations were a breakthrough

- 2003 SPECFEM3D_GLOBE: 14.6×10^9 DOF earthquake simulations
- Japanese Earth Simulator at 5 TFLOP/s on 1944 processors on 243 nodes
- Gordon Bell Prize Winner
- J. Tromp, D. Komatitsch, C. Ji, S. Tsuboi

Seismic imaging via adjoint tomography

New goal: improve earth model for more accurate simulations

- Uses nonlinear-optimization techniques to update 3D-velocities
- Can harness existing forward solver with relatively small changes to code

Computational problem: Total number of simulations is very large

Application: European Tomography

Physical challenge: in general, Europe is not very seismically active.

- Seismologists cannot run experiments. Rely on earthquakes and other seismic sources to probe crust and mantle.

Application: European Tomography

Physical challenge: in general, Europe is not very seismically active.

- Seismologists cannot run experiments. Rely on earthquakes and other seismic sources to probe crust and mantle.

Solution: Utilize ubiquitous “noise” as a source for tomography. Algorithm requires:

- 3 simulations / seismic station

Application: European Tomography

Physical challenge: in general, Europe is not very seismically active.

- Seismologists cannot run experiments. Rely on earthquakes and other seismic sources to probe crust and mantle.

Solution: Utilize ubiquitous "noise" as a source for tomography. Algorithm requires:

- 3 simulations / seismic station
- $\times$150 stations

Application: European Tomography

Physical challenge: in general, Europe is not very seismically active.

- Seismologists cannot run experiments. Rely on earthquakes and other seismic sources to probe crust and mantle.

Solution: Utilize ubiquitous "noise" as a source for tomography. Algorithm requires:

- 3 simulations / seismic station
- $\times$150 stations
- $\times$ $\sim$20 optimizations steps $=$ 9,000 simulations

Application: European Tomography

Physical challenge: in general, Europe is not very seismically active.

- Seismologists cannot run experiments. Rely on earthquakes and other seismic sources to probe crust and mantle.

Solution: Utilize ubiquitous "noise" as a source for tomography. Algorithm requires:

- 3 simulations / seismic station
- $\times$150 stations
- $\times$ $\sim$20 optimizations steps $=$ 9,000 simulations

$\rightarrow 1.5 \times 10^6$ CPU hours: 168 days on 324 cores.

SPECFEM3D

SPECFEM3D 2.0 “Sesame” is a finite element (FEM), elastic/acoustic, seismic wave-propagation solver for arbitrary user-generated hexahedral meshes.

SPECFEM3D

SPECFEM3D 2.0 “Sesame” simulates seismic events

SPECFEM3D

SPECFEM3D 2.0 “Sesame” simulates seismic events

- Written using Fortran90 and MPI

SPECFEM3D

SPECFEM3D 2.0 “Sesame” simulates seismic events

- Written using Fortran90 and MPI
- Mature, flexible, and fast; great example why we still use Fortran in science.

SPECFEM3D

SPECFEM3D 2.0 “Sesame” simulates seismic events

- Written using Fortran90 and MPI
- Mature, flexible, and fast; great example why we still use Fortran in science.
- Large user community across large array of applications

SPECFEM3D

SPECFEM3D 2.0 “Sesame” simulates seismic events

- Written using Fortran90 and MPI
- Mature, flexible, and fast; great example why we still use Fortran in science.
- Large user community across large array of applications
- Calculates in single precsion.

SPECFEM3D

SPECFEM3D 2.0 “Sesame” simulates seismic events

- Written using Fortran90 and MPI
- Mature, flexible, and fast; great example why we still use Fortran in science.
- Large user community across large array of applications
- Calculates in single precsion.

GPU Version:

- Fortran + C/CUDA + MPI for GPU clusters

Goal:

SPECFEM3D

SPECFEM3D 2.0 “Sesame” simulates seismic events

- Written using Fortran90 and MPI
- Mature, flexible, and fast; great example why we still use Fortran in science.
- Large user community across large array of applications
- Calculates in single precsion.

GPU Version:

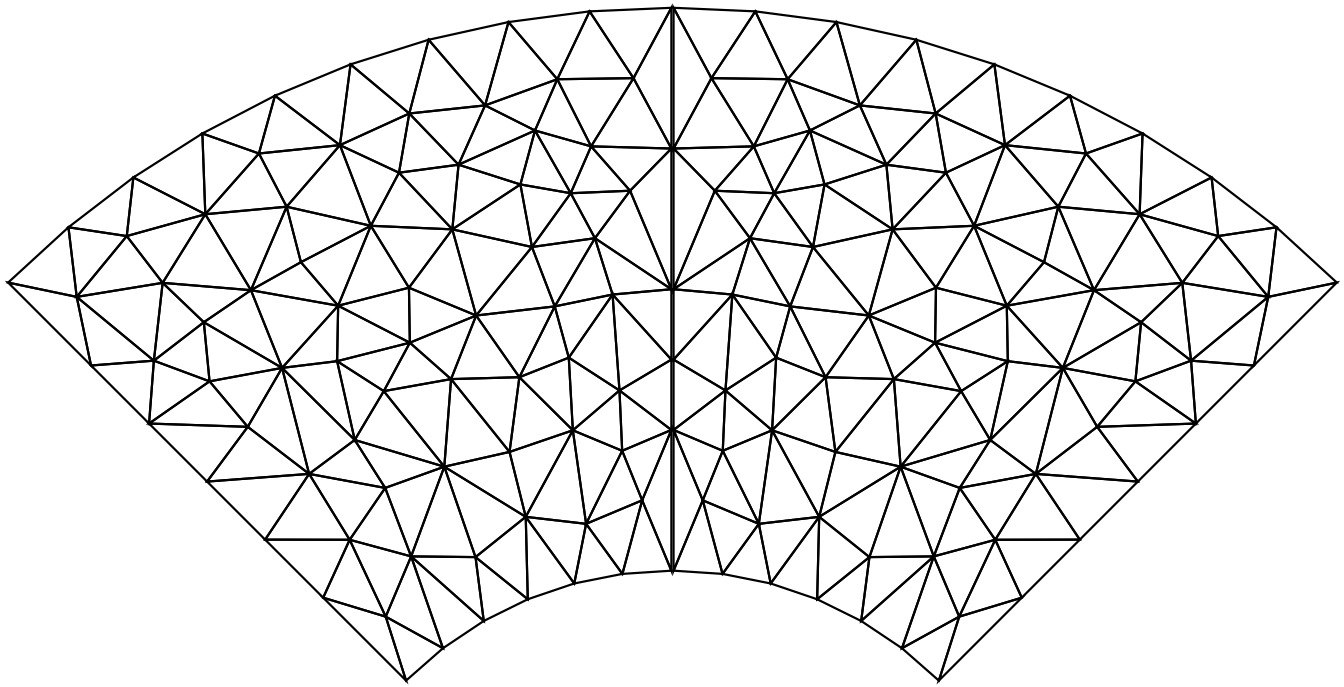
- Fortran + C/CUDA + MPI for GPU clusters

Goal:

Development Challenges

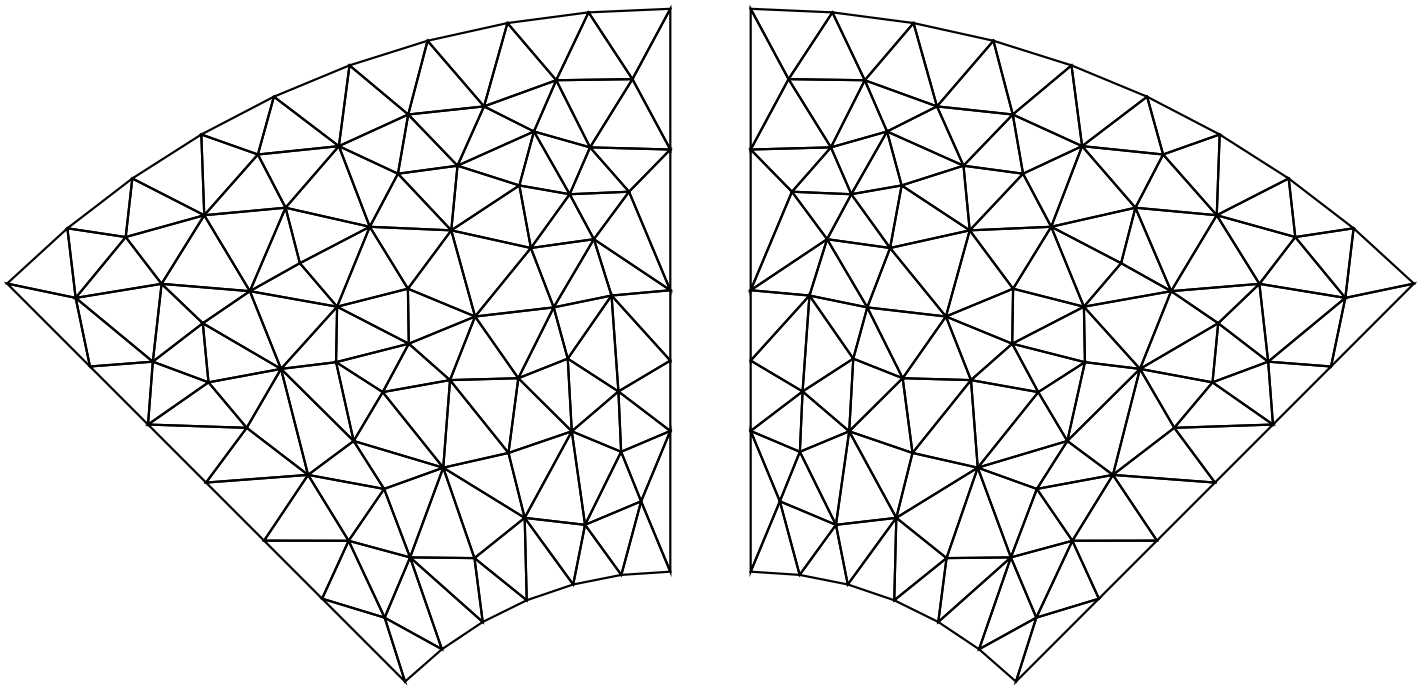
- **Asynchronous communications**
- Shared DOF race condition
- CUDA and memory optimizations

Note on FEM Parallelization



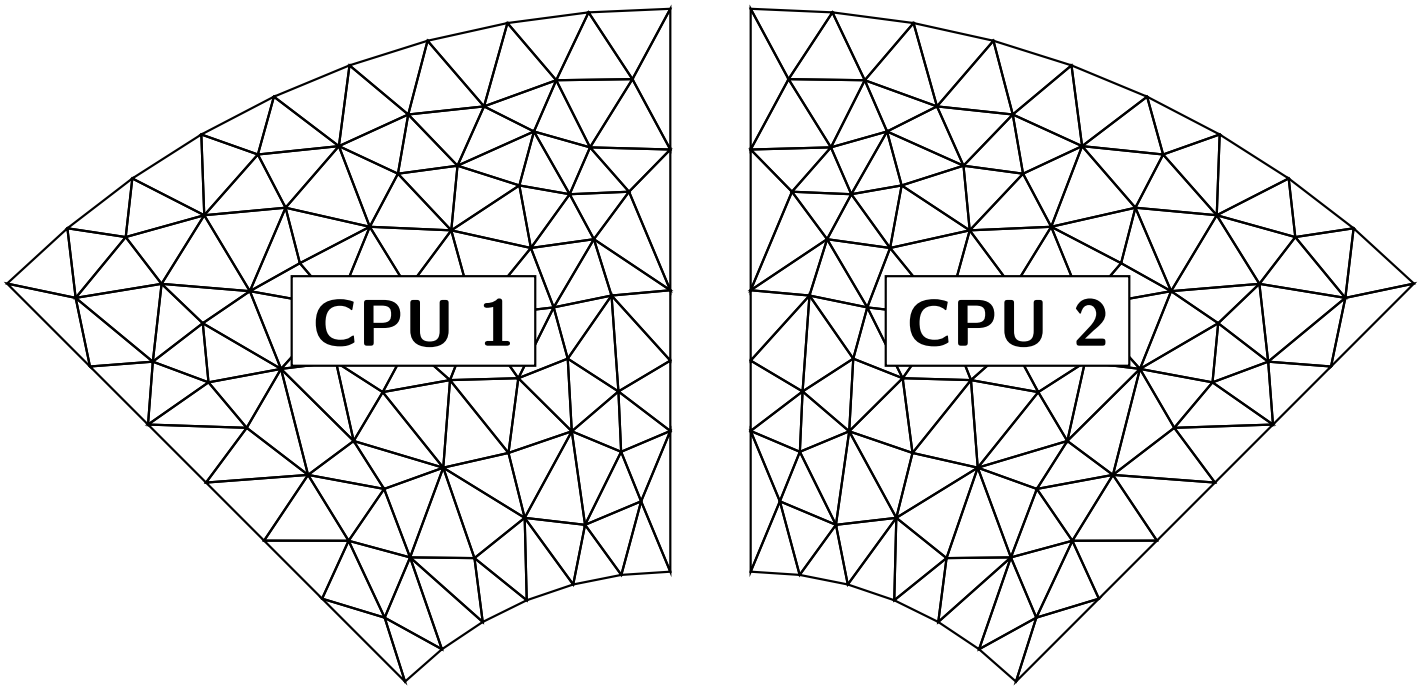
1. Start with finite element mesh.

Note on FEM Parallelization



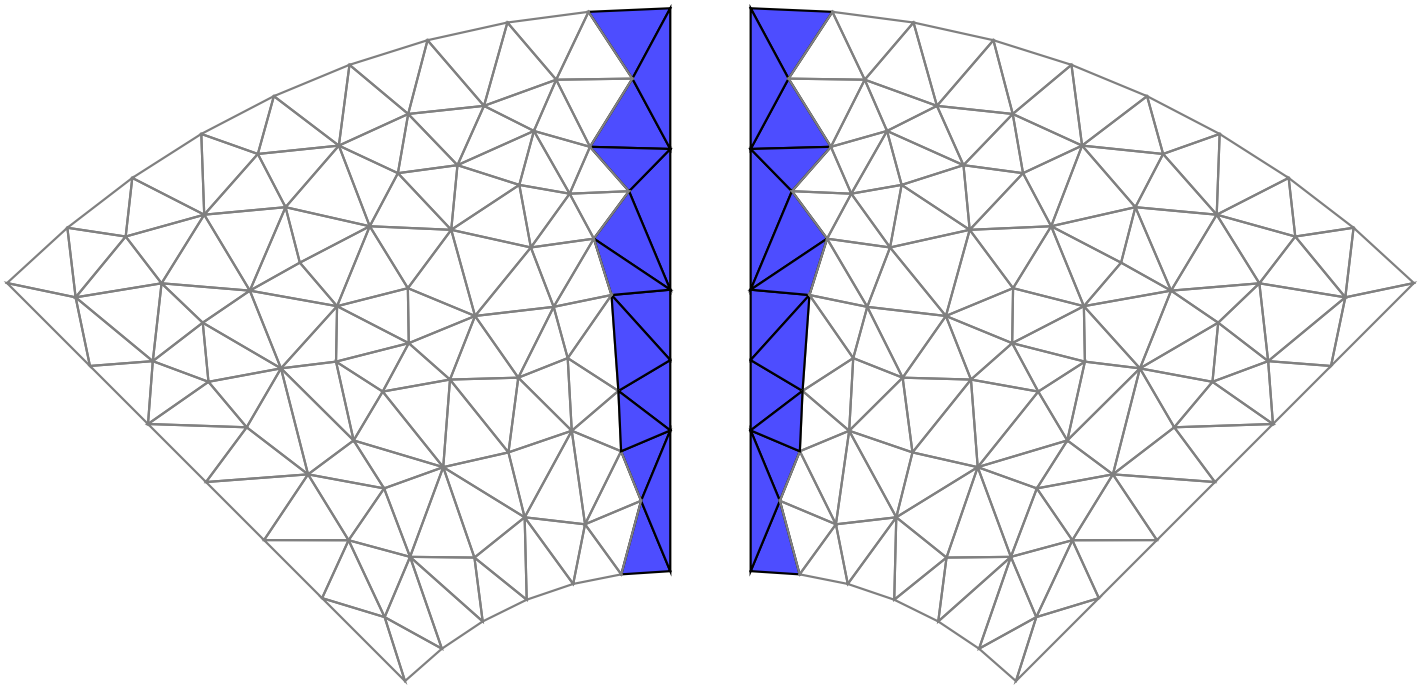
1. Start with finite element mesh.
2. Partition using SCOTCH or METIS.

Note on FEM Parallelization



1. Start with finite element mesh.
2. Partition using SCOTCH or METIS.
3. Each MPI rank gets a single partition.

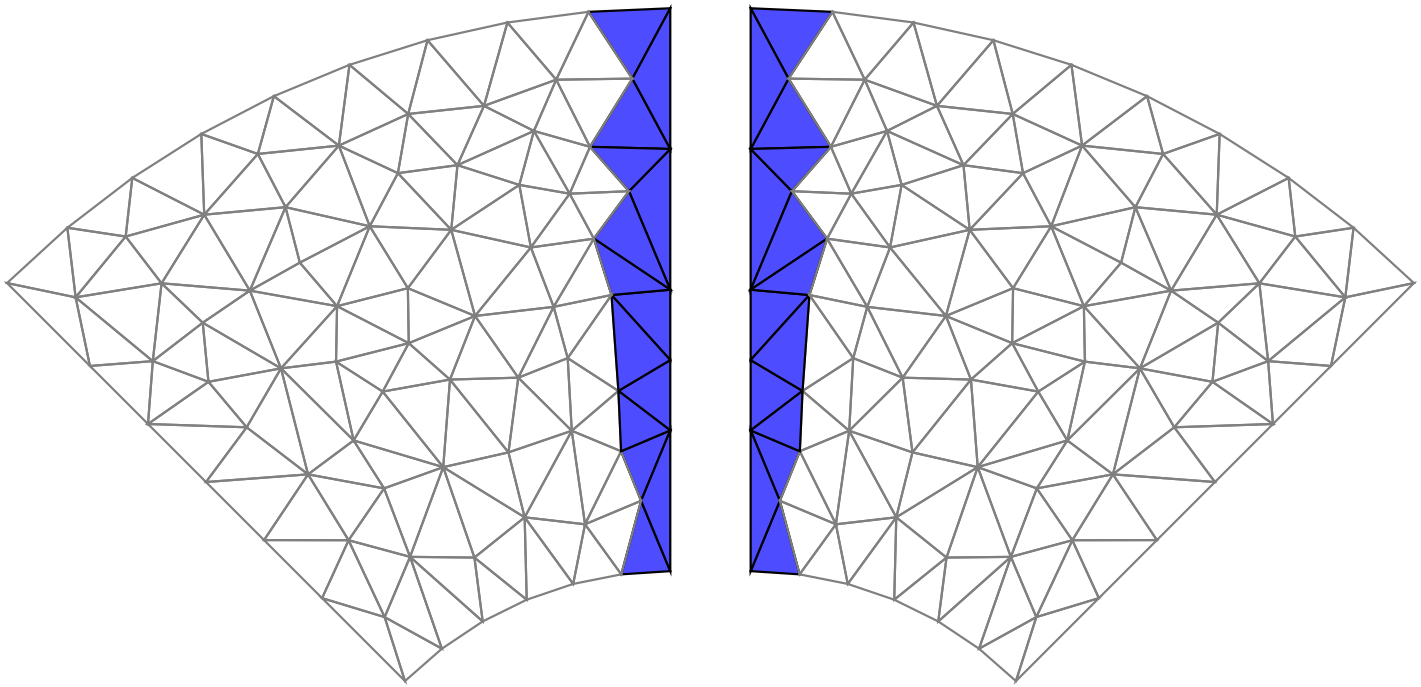
Development Challenge: Overlapping Communications



CPU Strategy:

1. Compute outer boundary nodes.

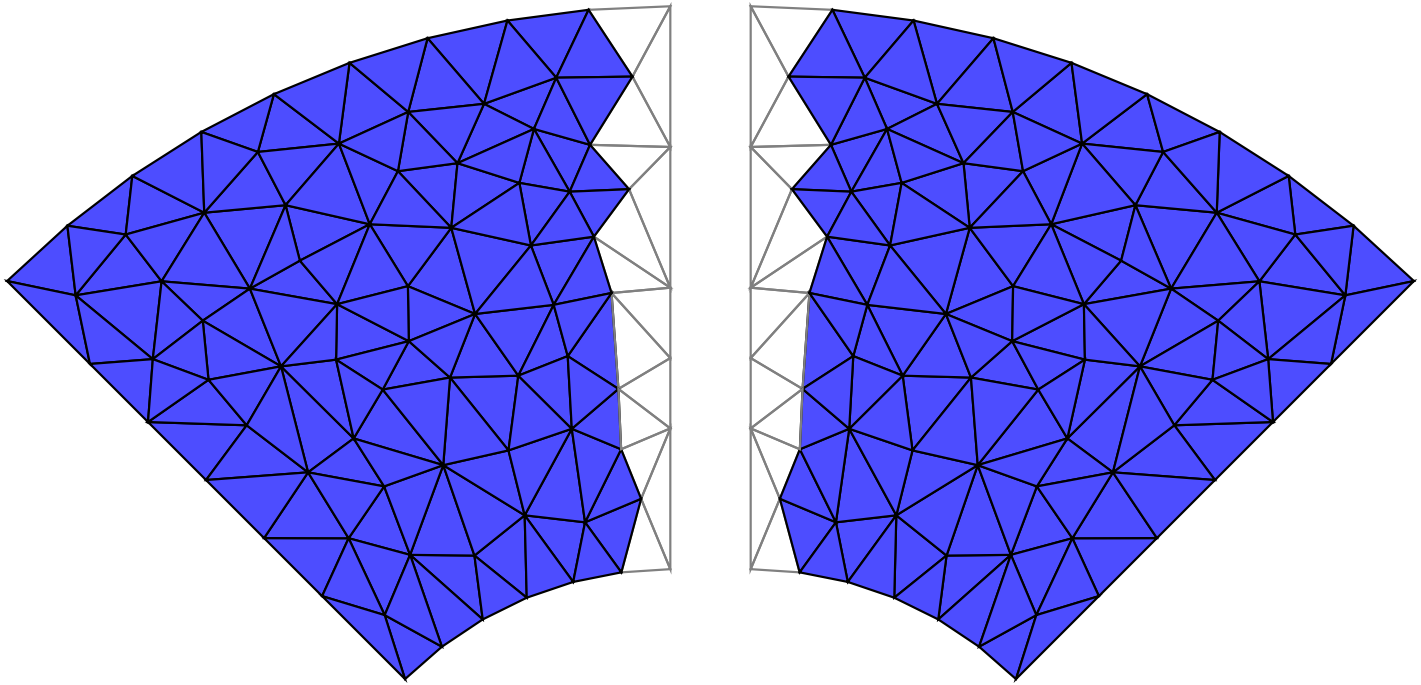
Development Challenge: Overlapping Communications



CPU Strategy:

1. Compute outer boundary nodes.
2. Start nonblocking MPI Communications.

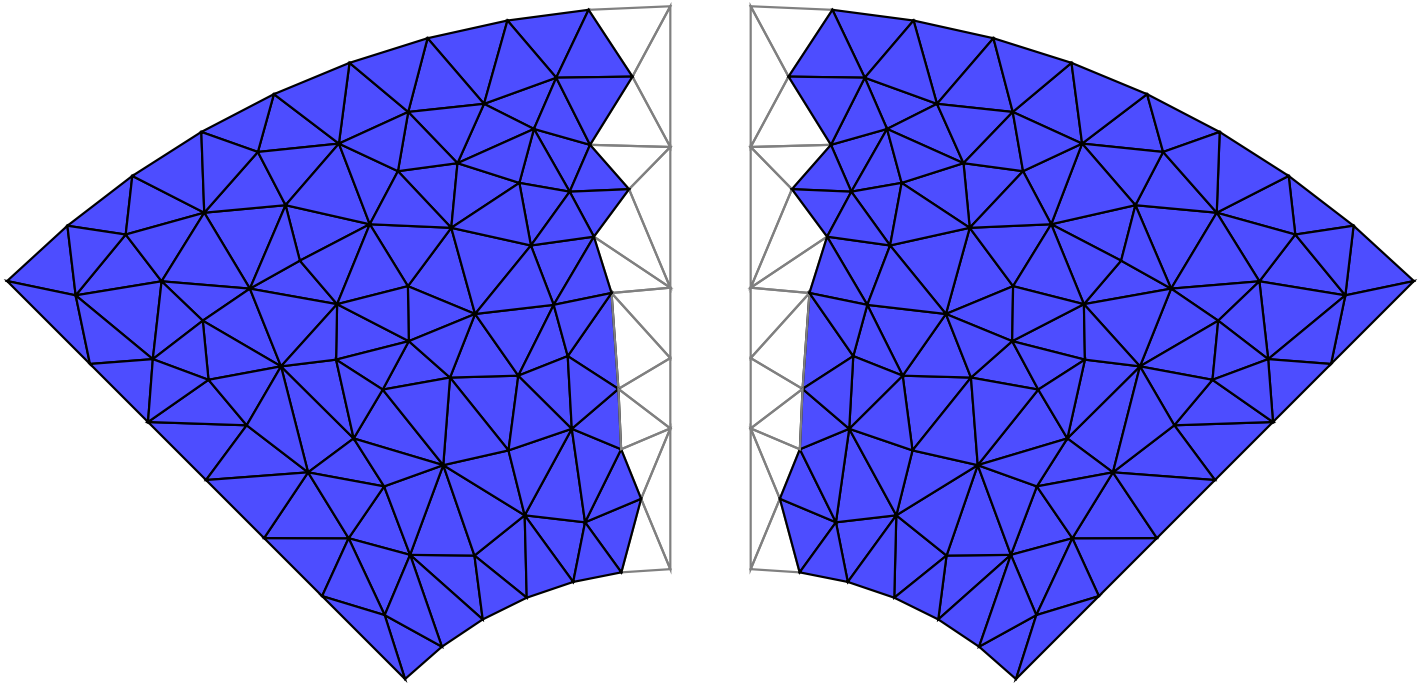
Development Challenge: Overlapping Communications



CPU Strategy:

1. Compute outer boundary nodes.
2. Start nonblocking MPI Communications.
3. Compute inner boundary nodes.

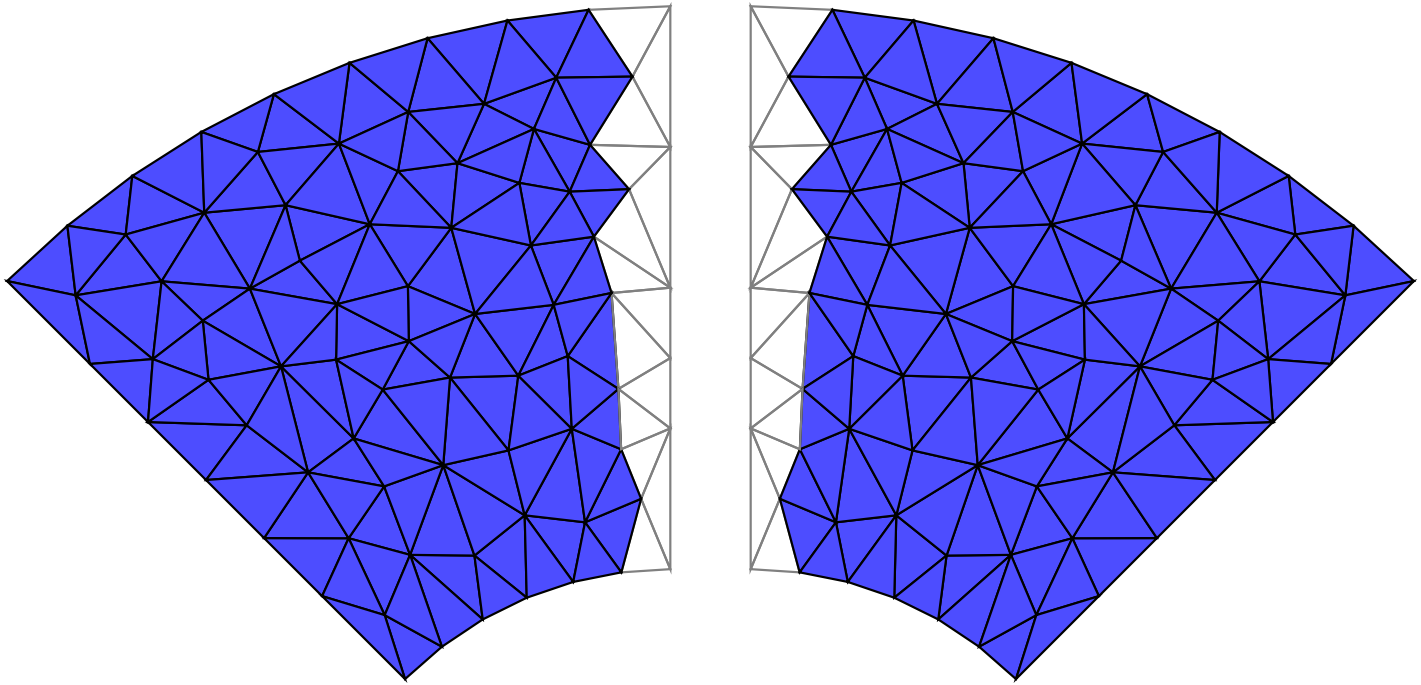
Development Challenge: Overlapping Communications



CPU Strategy:

1. Compute outer boundary nodes.
2. Start nonblocking MPI Communications.
3. Compute inner boundary nodes.
4. MPI Finishes.

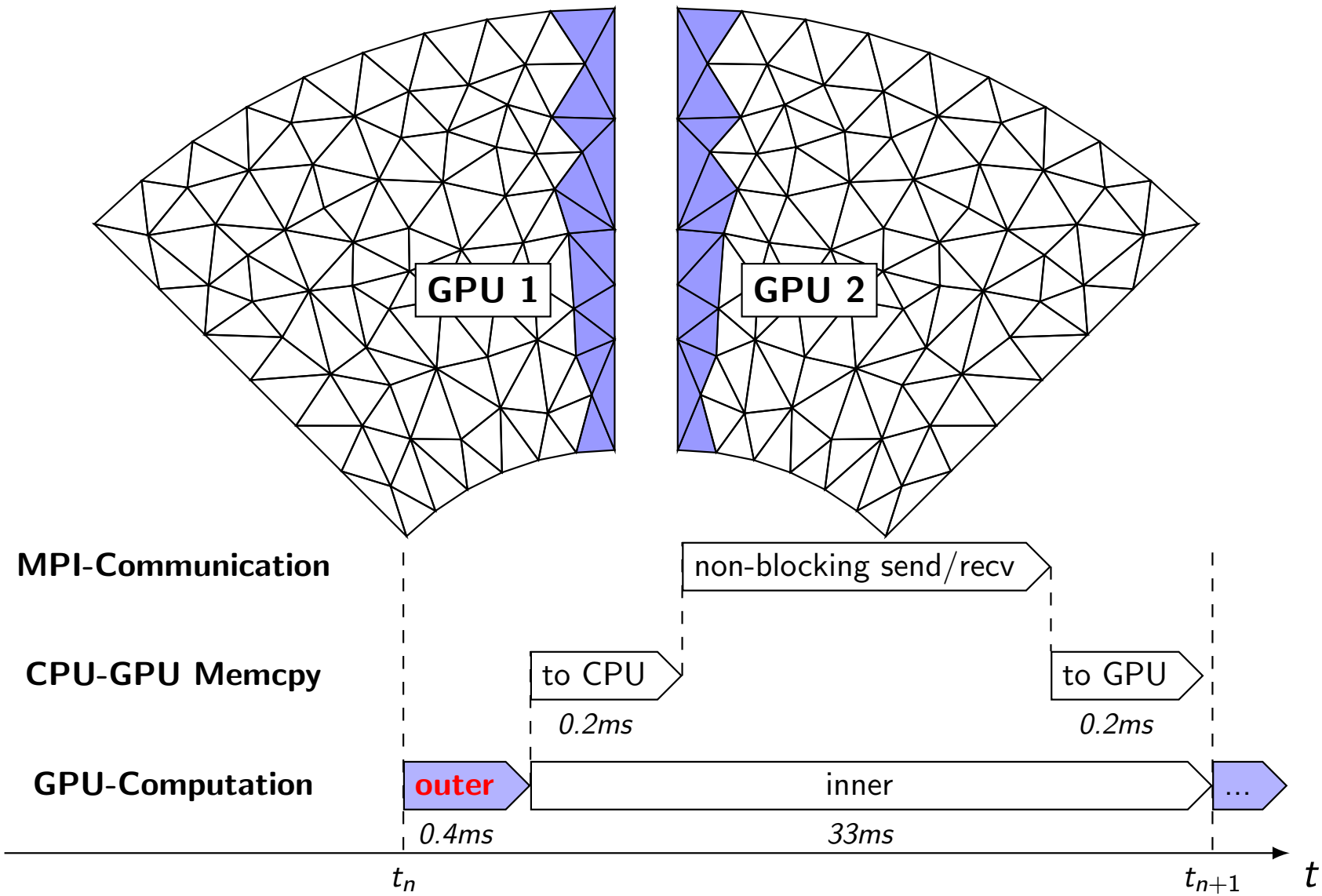
Development Challenge: Overlapping Communications



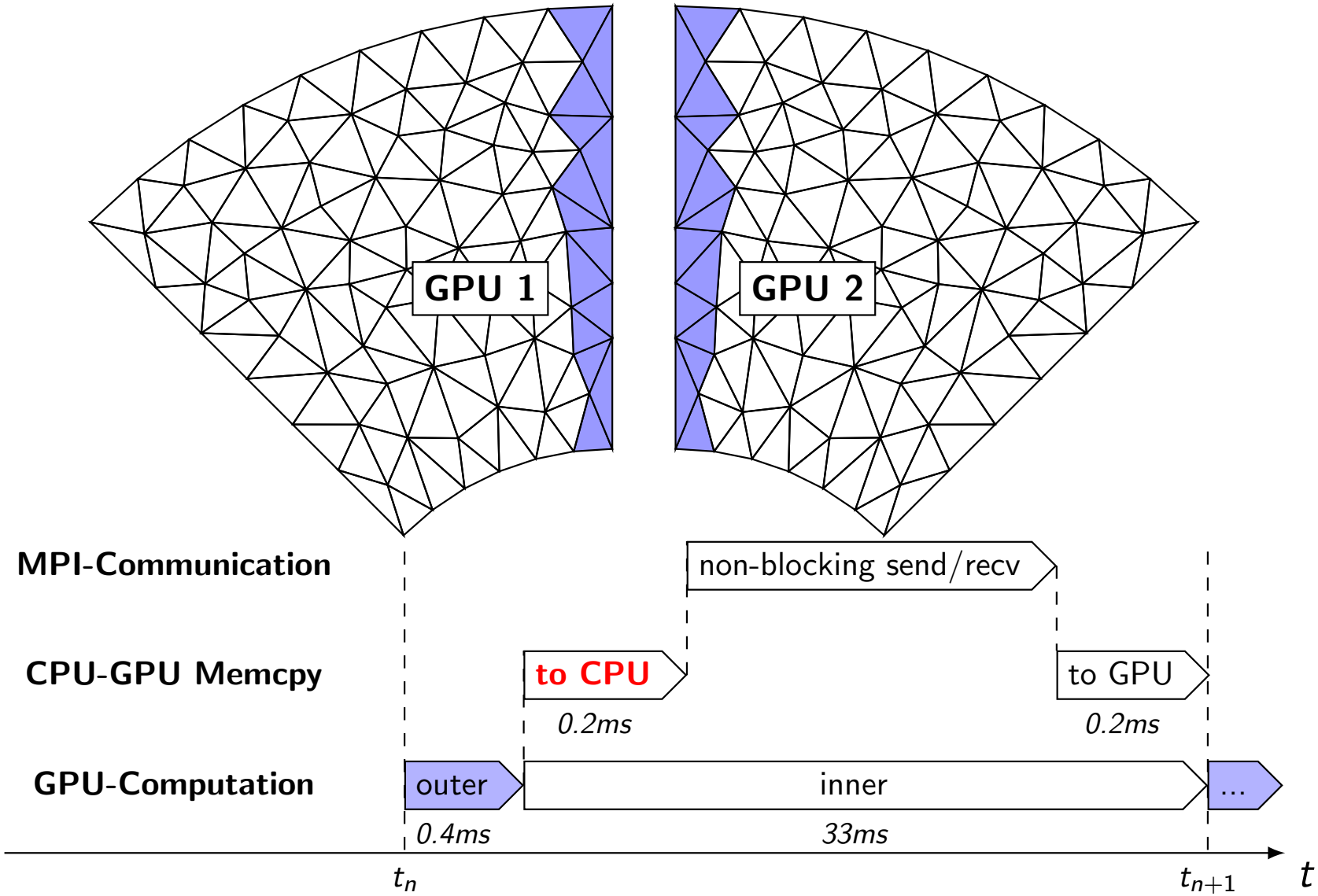
CPU Strategy:

1. Compute outer boundary nodes.
2. Start nonblocking MPI Communications.
3. Compute inner boundary nodes.
4. MPI Finishes.
5. Inner compute finishes.

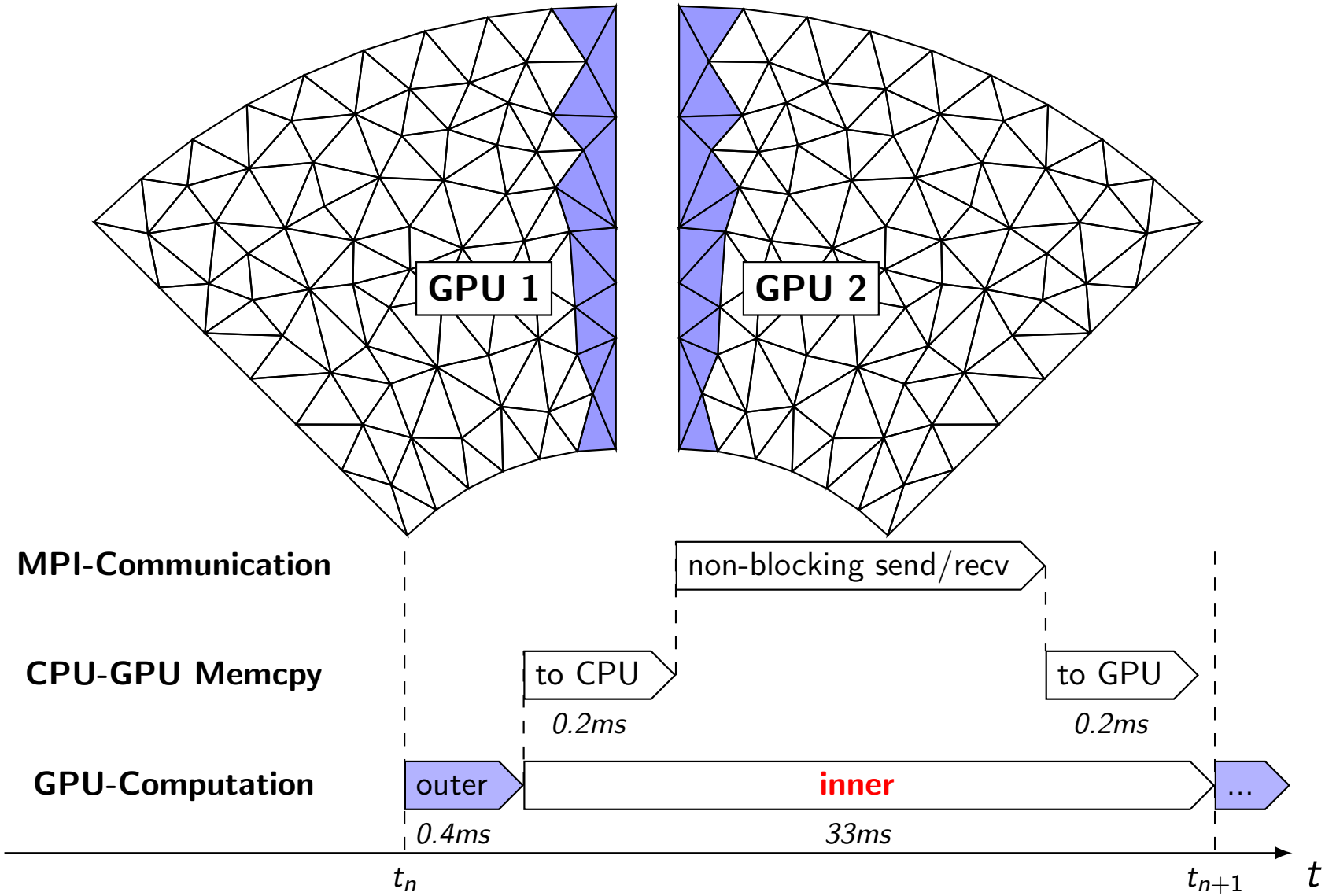
GPU Overlapping Timeline



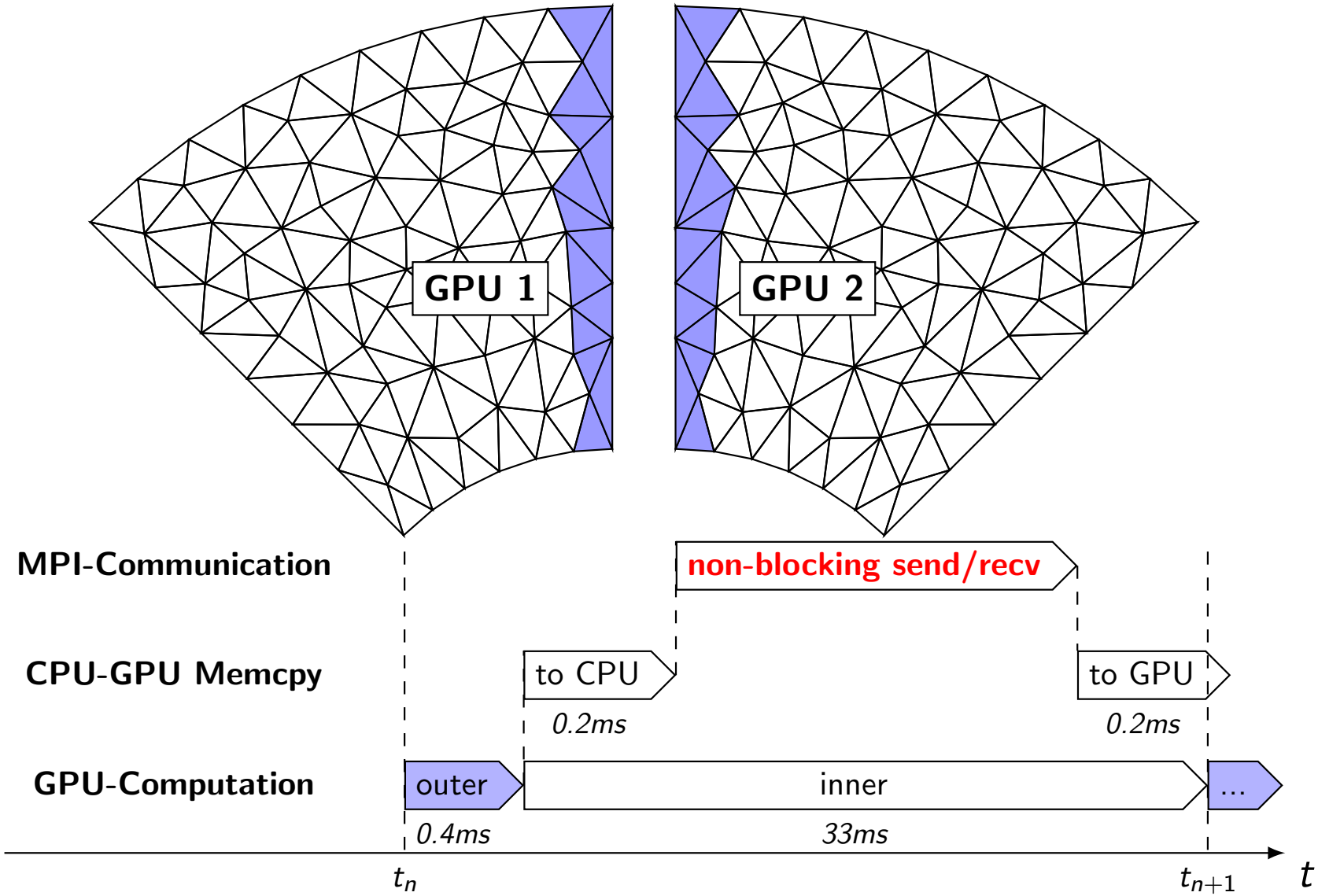
GPU Overlapping Timeline



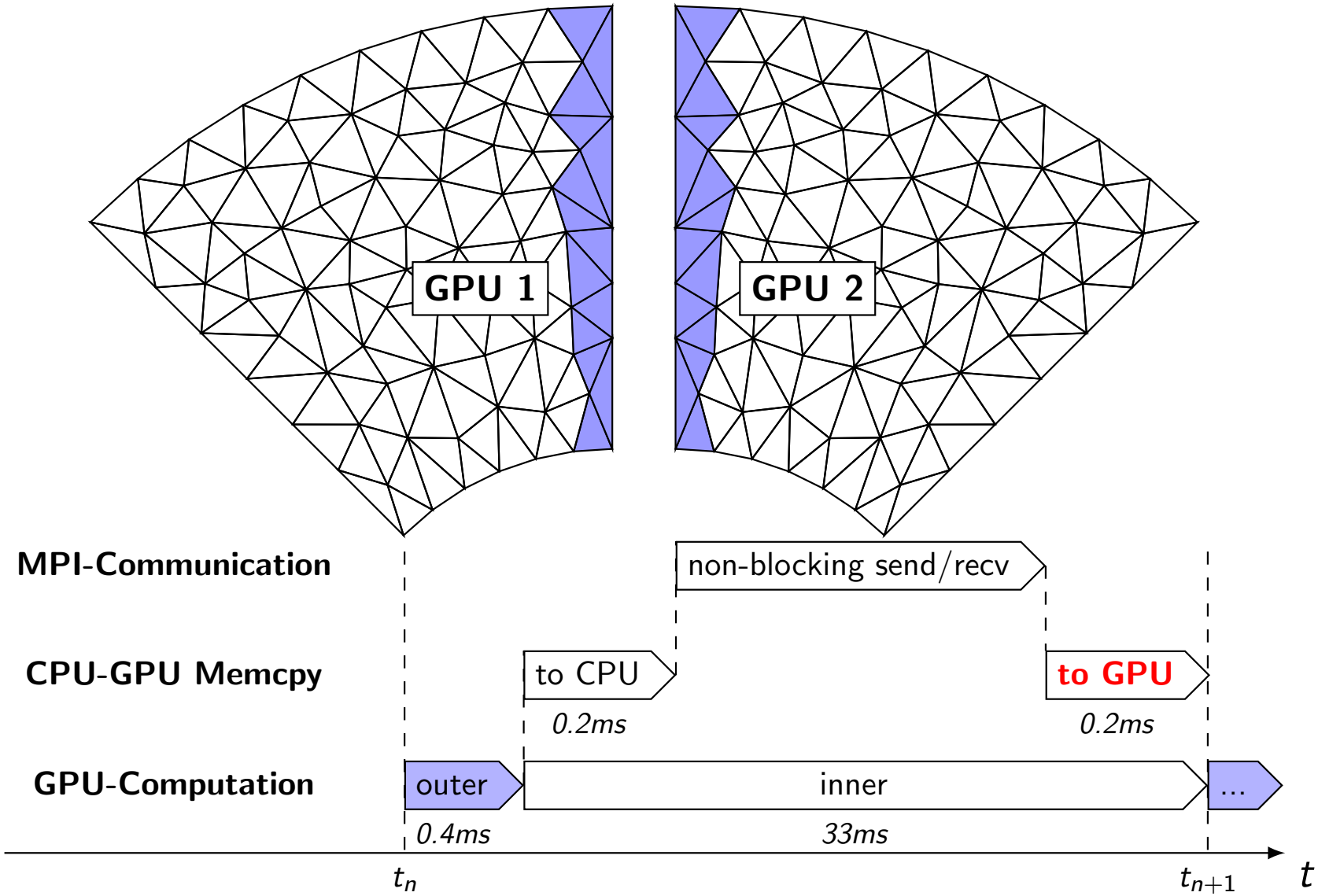
GPU Overlapping Timeline



GPU Overlapping Timeline



GPU Overlapping Timeline



Code Structure

for $t = 1$,NSTEPS **do**
    Time-step update ()
    for phase=outer,inner **do**
        Stiffness Assembly (phase)
        Absorbing Boundaries (phase)
        Source Forcing (phase)
        MPI-Communications (phase) // *sent asynchronously*
    end for
    Time-step finalize ()
    Calculate Seismograms ()
end for

Code Structure

for $t = 1$,NSTEPS **do**
    Time-step update ()
    for phase=outer,inner **do**
        Stiffness Assembly (phase) $>65\%$ of runtime.
        Absorbing Boundaries (phase)
        Source Forcing (phase)
        MPI-Communications (phase) // *sent asynchronously*
    end for
    Time-step finalize ()
    Calculate Seismograms ()
end for

Development Challenges

- Asynchronous communications
- **Shared DOF race condition**
- CUDA and memory optimizations

Assembly in CUDA

For 3D-elements:

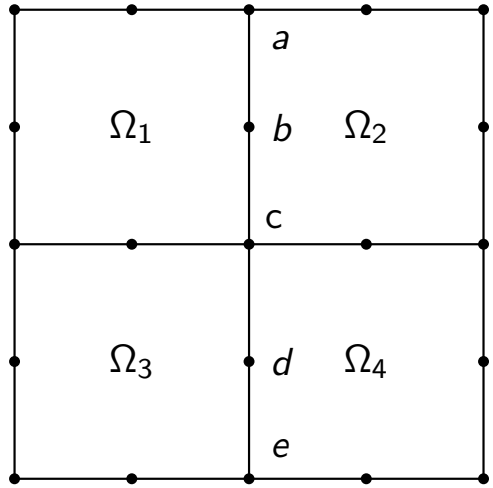
- 125 nodes per element

Each:

- CUDA Block assigned to Ω_k
- CUDA Thread assigned to node in Ω_k

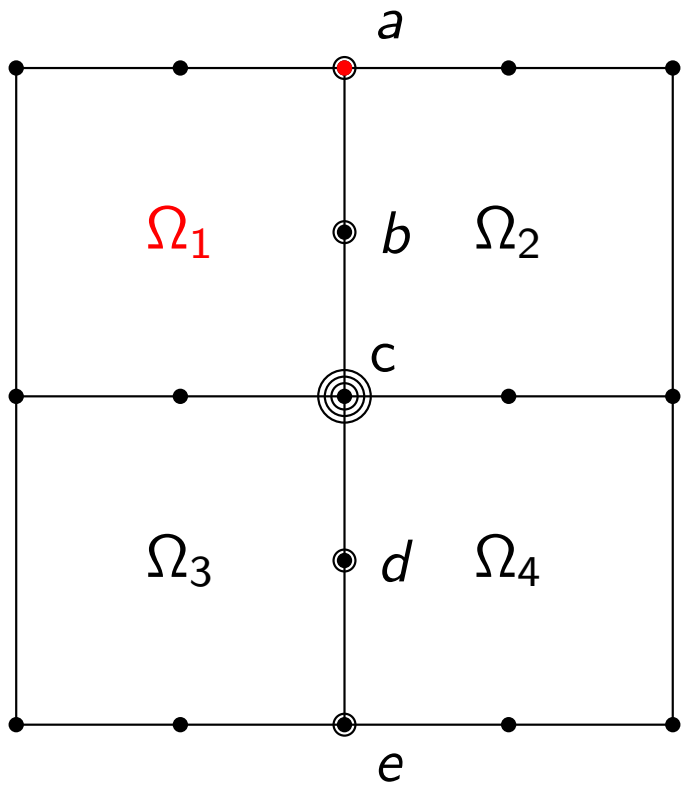
Threads cache matrix $\times$ matrix entries via shared memory.

Example 2D mesh:



Shared DOF

Example 2D FEM grid with shared nodes a, b, c, d and e:

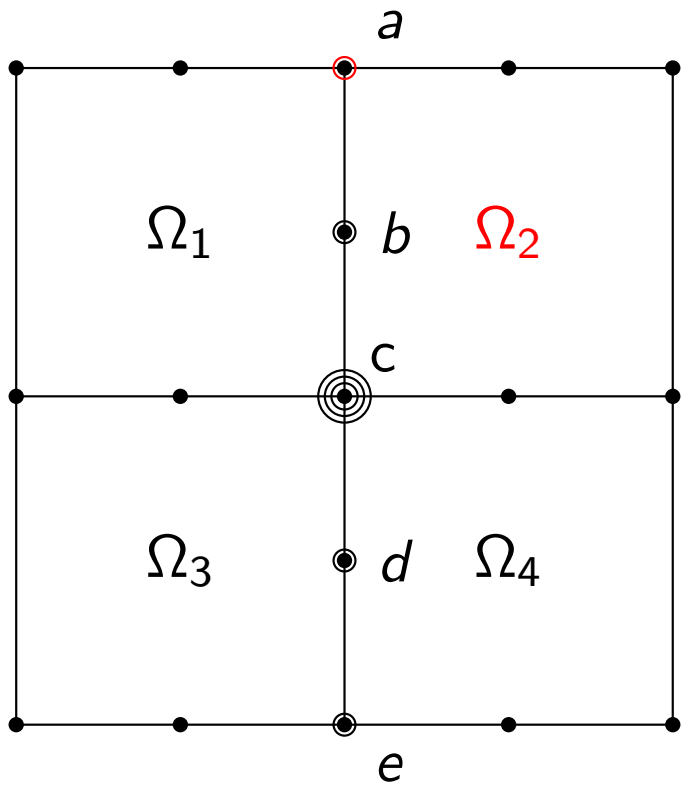


Assembly:

$a = f(\Omega_1) + f(\Omega_2)$

Shared DOF

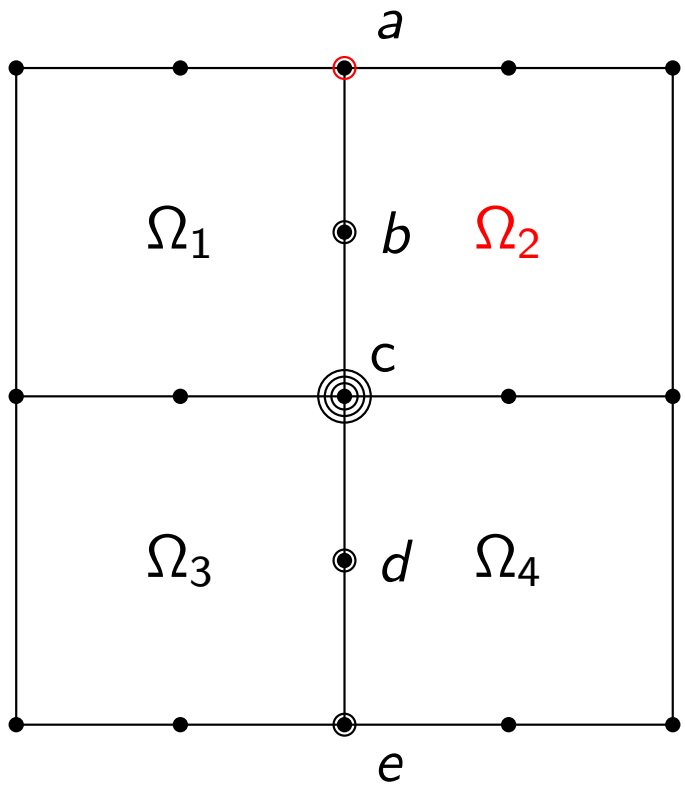
Example 2D FEM grid with shared nodes a, b, c, d and e:



Assembly:
$a = f(\Omega_1) + f(\Omega_2)$

Shared DOF

Example 2D FEM grid with shared nodes a, b, c, d and e:



Assembly:

$a = f(\Omega_1) + f(\Omega_2)$

$c = f(\Omega_1) + f(\Omega_2) + f(\Omega_3) + f(\Omega_4)$

Shared DOF Race Condition

In code:

```
d_accel[iglob*3]   += sum_terms1; // x
d_accel[iglob*3+1] += sum_terms2; // y
d_accel[iglob*3+2] += sum_terms3; // z
```

Concurrent thread view:

Shared DOF Race Condition

In code:

```
d_accel[iglob*3]   += sum_terms1; // x
d_accel[iglob*3+1] += sum_terms2; // y
d_accel[iglob*3+2] += sum_terms3; // z
```

Concurrent thread view:

thread $\Omega_1(a)$	thread $\Omega_2(a)$
read \$1=a, (a==0)	read \$1=a, (a==0)
$\$1 = 0 + f(\Omega_1)$	$\$1 = 0 + f(\Omega_2)$
store a	store a

$$a = f(\Omega_1) \text{ or } f(\Omega_2)$$

Shared DOF Solution

Easiest solution:

- `atomicAdd()` — Correct but slow!

In code:

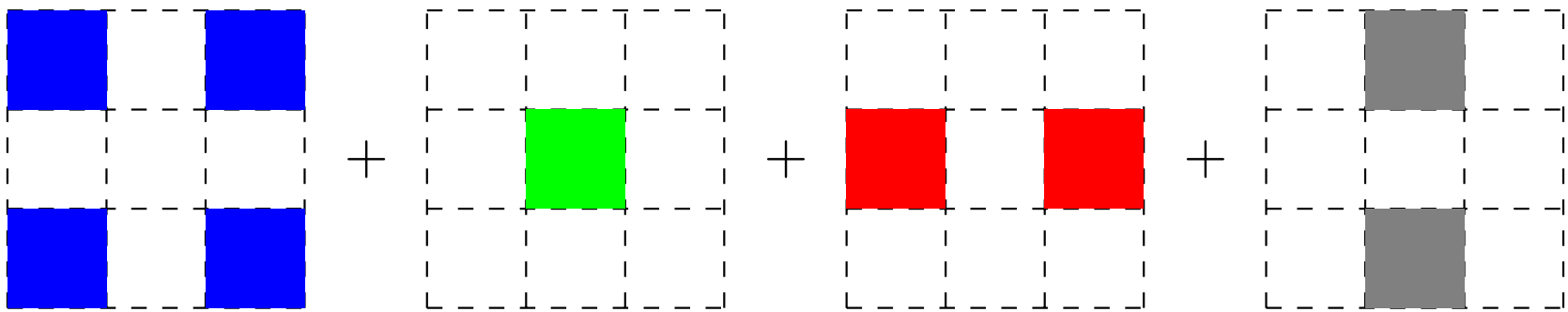
```
atomicAdd(&d_accel[iglob*3],   sum_terms1); // x
atomicAdd(&d_accel[iglob*3+1], sum_terms2); // y
atomicAdd(&d_accel[iglob*3+2], sum_terms3); // z
```

Mesh Coloring

Best solution: Mesh Coloring

$$\bigcup_{k}^{K} \Omega_k = \Omega$$

2D Grid needs 4 colors:



Each Ω_k is independent.

Coloring Algorithms

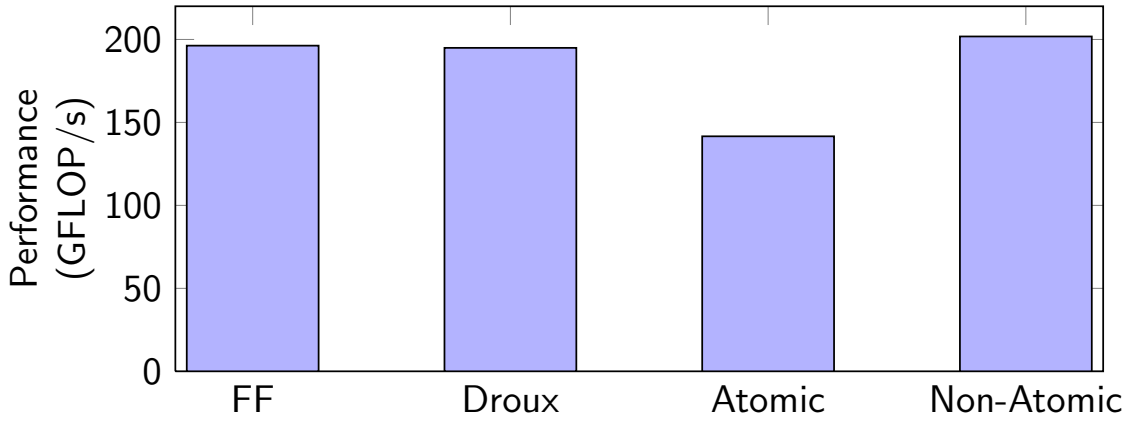
Two types we implemented

- First Fit (FF): Greedy Algorithm that always chooses first available color.
- Droux: Chooses the least used color, balancing elements / color.

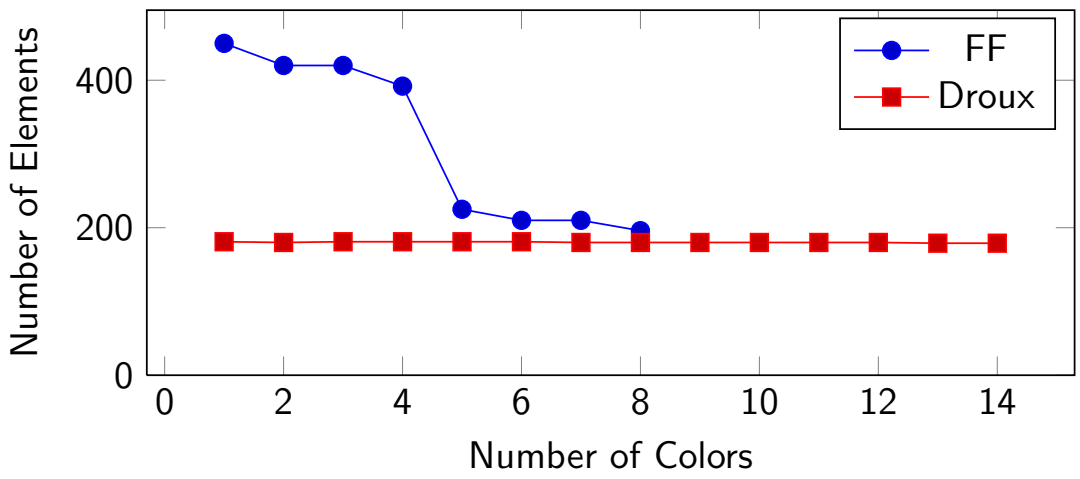
Mesh coloring requires a serial loop over colors, with one kernel launch per color.

Coloring Algorithms in Practice

Coloring Performance:



Coloring balance:



Development Challenges

- Asynchronous communications
- Shared DOF race condition
- **CUDA and memory optimizations**

CUDA Optimizations

The stiffness assembly kernel is memory bound (arithmetic intensity of $\sim$2.0). Important to optimize GPU global-memory transfers:

- Mesh constants 128 padded
- Global memory instead of `__constant__` memory
- Updated fields bound to texture memory

Performance Comparison

Performance experiments conducted on:

- Cray XK6: 16-core AMD Opteron + X2090 Tesla GPU

Performance Comparison

Performance experiments conducted on:

- Cray XK6: 16-core AMD Opteron + X2090 Tesla GPU
- Cray XE6: $2 \times$ 16-core AMD Opteron

Performance Comparison

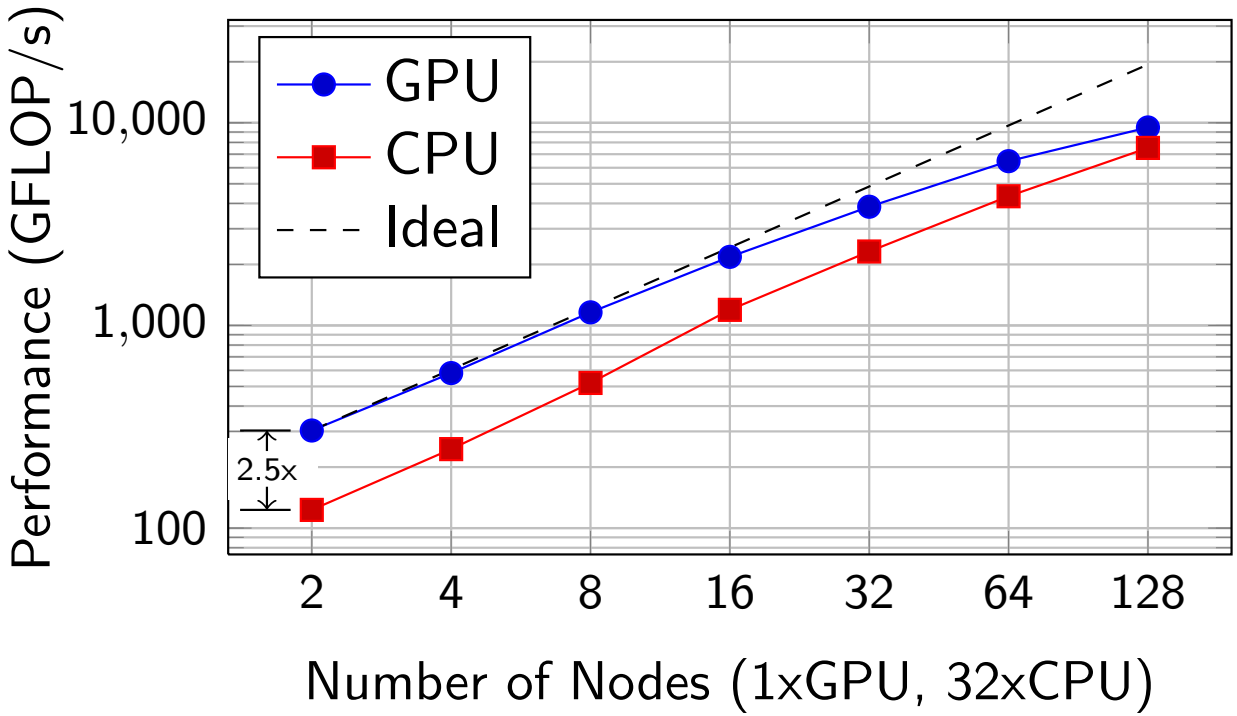
Performance experiments conducted on:

- Cray XK6: 16-core AMD Opteron + X2090 Tesla GPU
- Cray XE6: $2 \times$ 16-core AMD Opteron

We compare performance **node-to-node**.

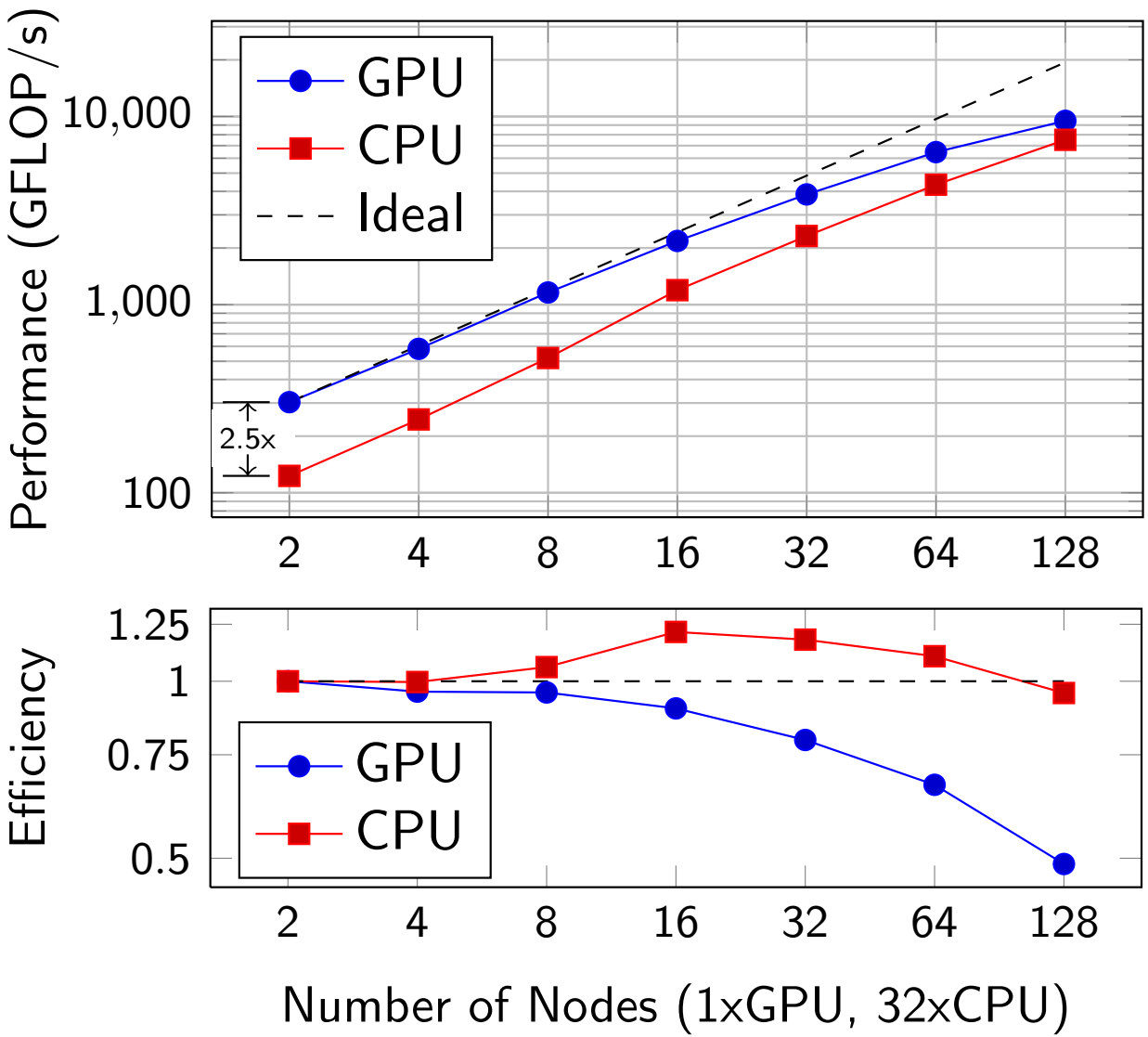
Strong Scaling

300×10^3 element mesh:

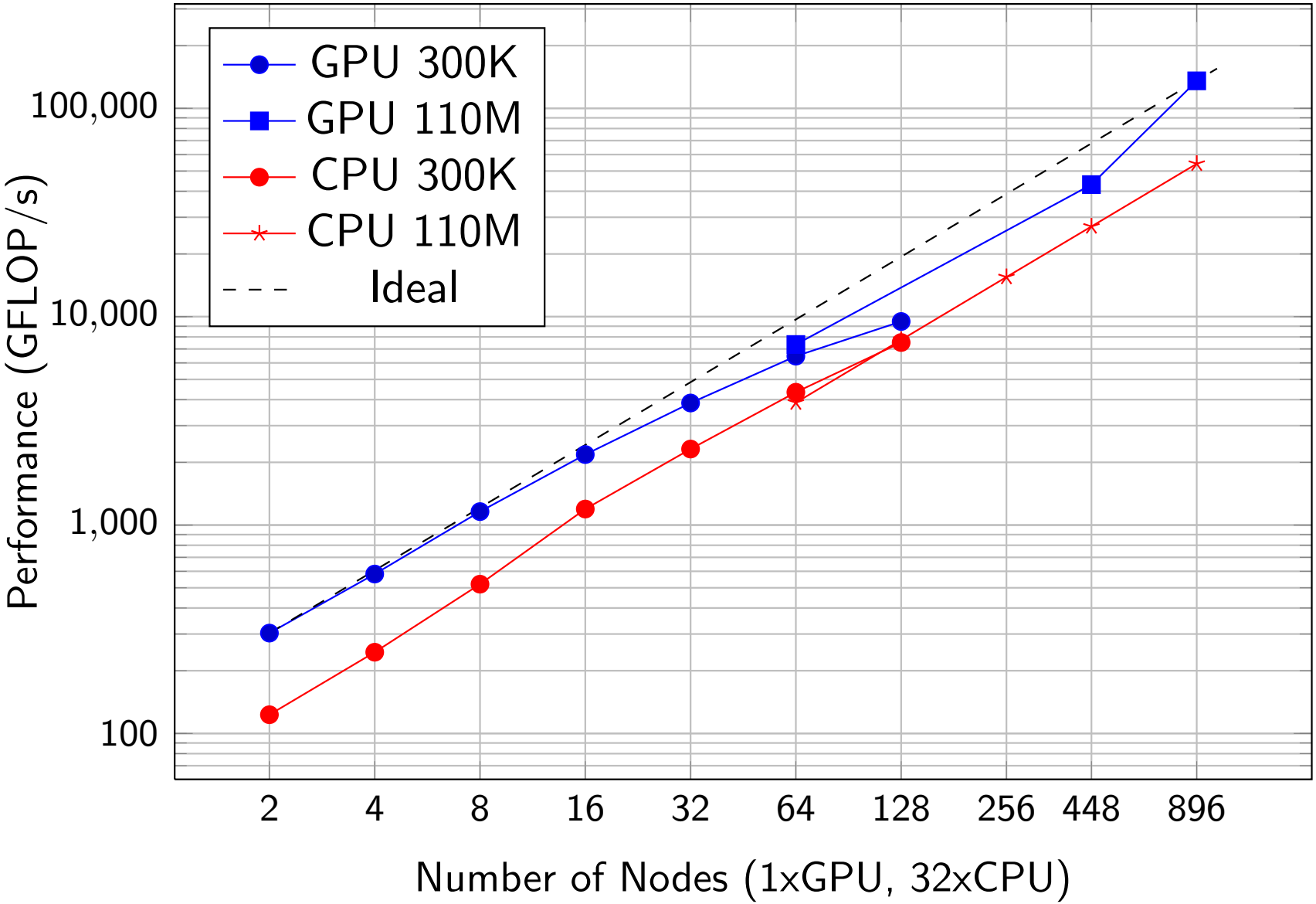


Strong Scaling

300×10^3 element mesh:



Strong Scaling: Varied Meshes

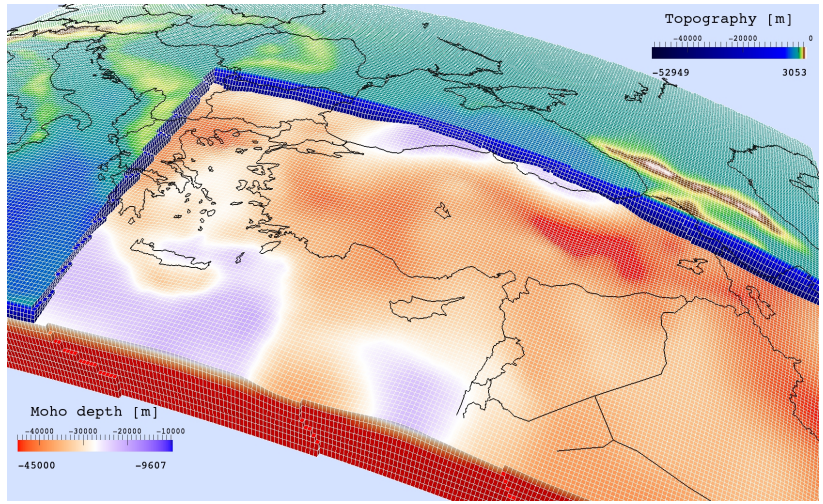


Case Study 1: Turkey Earthquakes

We created 19M mesh covering Europe, Middle East, and Northern Africa.

Simulated 3 earthquakes in Turkey:

1. 1999 Izmit Mw 7.8
2. 2011 Mw 7.3
3. Scenario Mw 7.3 20 Km from Istanbul.



Case Study 1: Performance

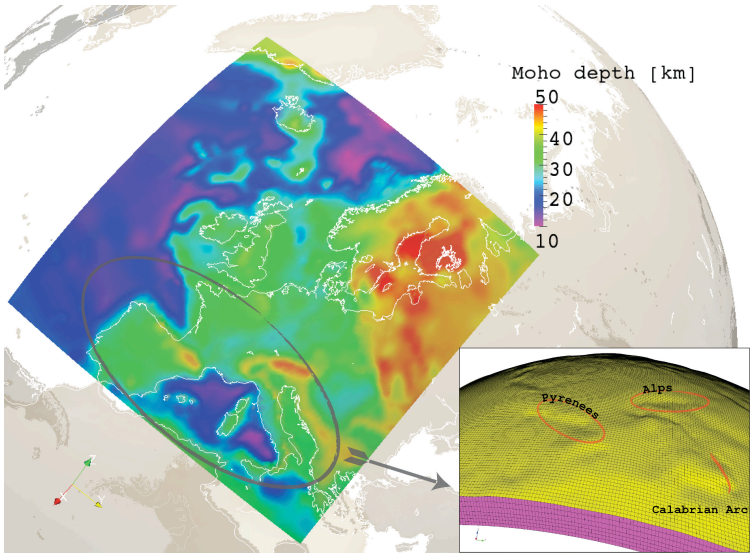
Simulated on 896 Cray XK6 nodes on Titandev at ORNL:

- 75,000 time steps, taking 28 minutes = 35 TFLOP/s.
- Over 739 recording stations, which are very *unbalanced*.
- Removing stations led to 78 TFLOP/s.

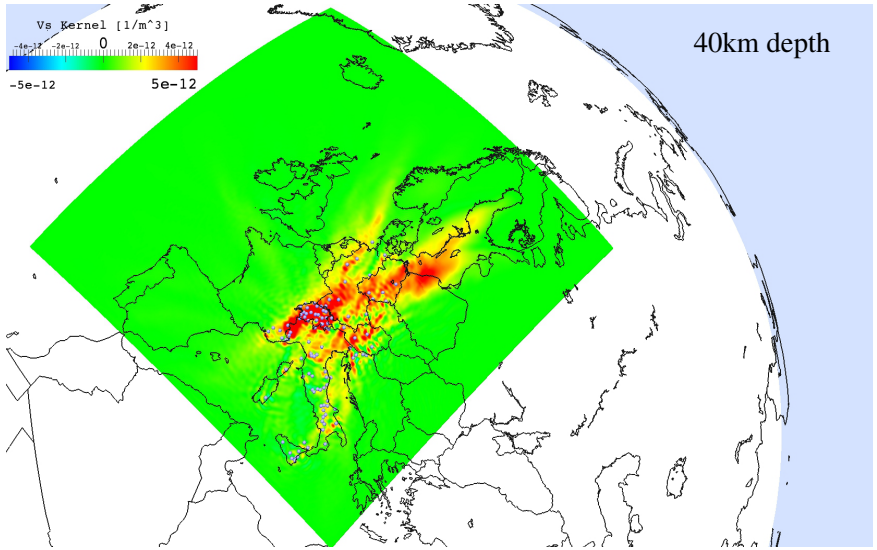
Highlights need to optimize for application, not just synthetic benchmarks.

Case Study 2: Noise tomography

A project to image Europe using ambient noise sources:



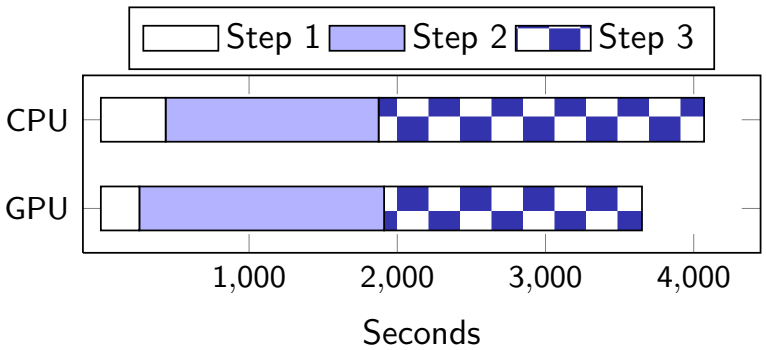
200,000 element mesh of Europe

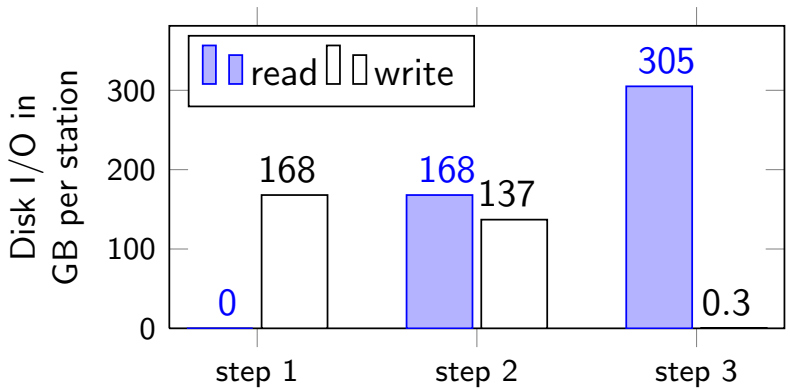


Gradient sum from 150 station contributions

Case Study 2: Performance

The tomography algorithm requires 3-steps, with significant I/O in each step.





Simulations run on *Tödi* at the Swiss Supercomputing Center CSCS.

Conclusion

- We extended community code SPECFEM3D to work with GPU-clusters.

Conclusion

- We extended community code SPECFEM3D to work with GPU-clusters.
- A combination of asynchronous communications, mesh coloring, and cuda-specific optimizations yield a speedup of 1.7x to 2.5x.

Conclusion

- We extended community code SPECFEM3D to work with GPU-clusters.
- A combination of asynchronous communications, mesh coloring, and cuda-specific optimizations yield a speedup of 1.7x to 2.5x.
- Showed two real examples: a large earthquake simulation on 896 GPUs, and a tomography example using adjoint methods to image Europe.
- Important to profile a real scientific application — synthetic benchmark examples may not show the whole story.