High-Order Finite-Element Earthquake Modeling on very Large Clusters of CPUs or GPUs

Gordon Erlebacher
Department of Scientific Computing
Sept. 28, 2012

with

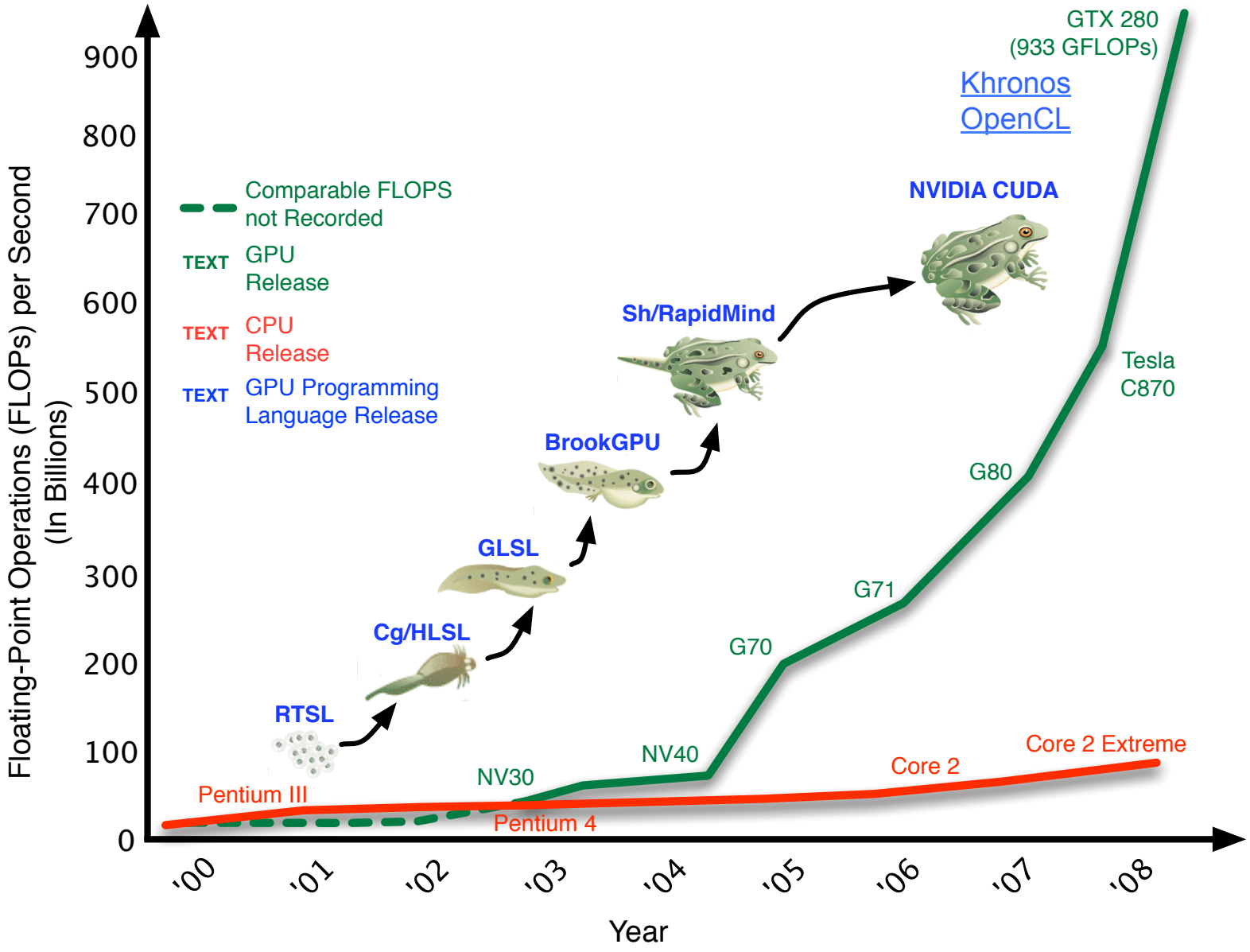
Dimitri Komatitsch (Pau,France)
David Michea (Pau, France)
Dominik Goddeke (Dortmund, Germany)

I met Yousuff in 1986, while at NASA Langley Research Center

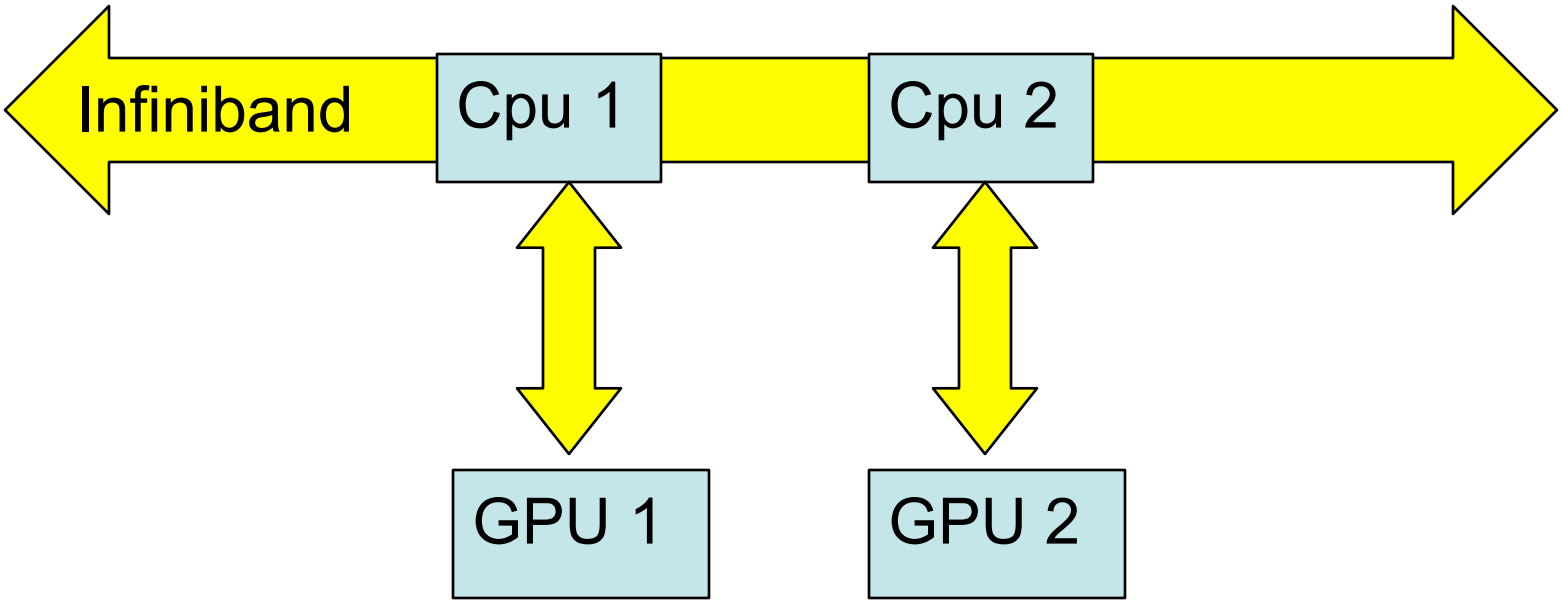
- Introduced me to spectral methods, Transition, Turbulence, parallelism, …
- Helped develop my talents in research, and writing through a grueling (for me) schedule
- Helped me achieve balance between my personal and professional life

Evolution GPU

Fermi, GTX480



Setup



10-30x speedup GPU vs CPU

Amdahl: Amount of serialization must decrease!!!

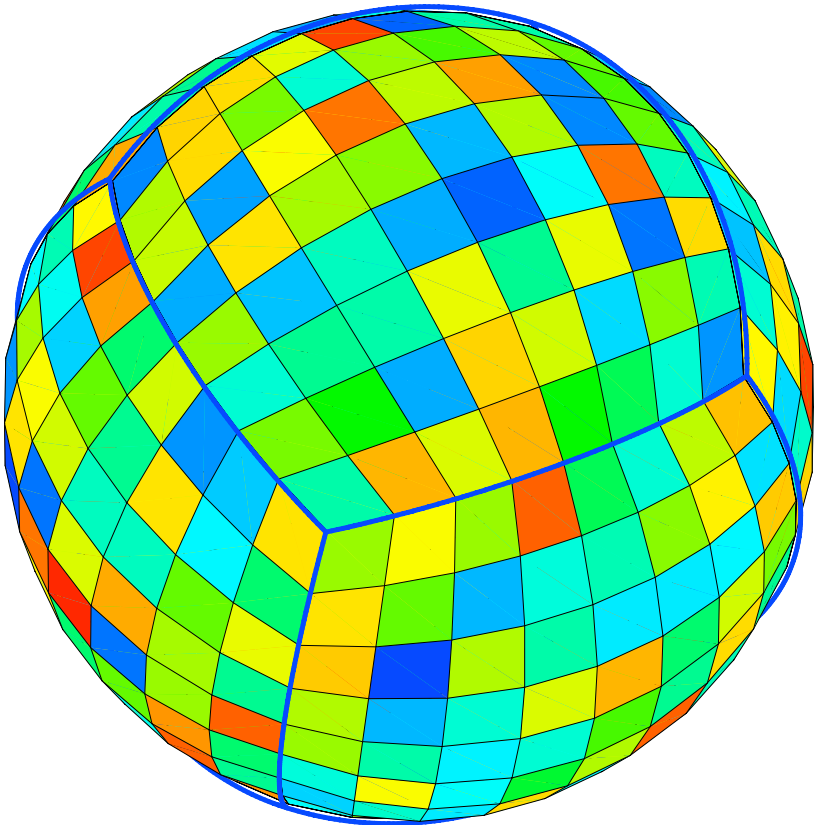
Topics

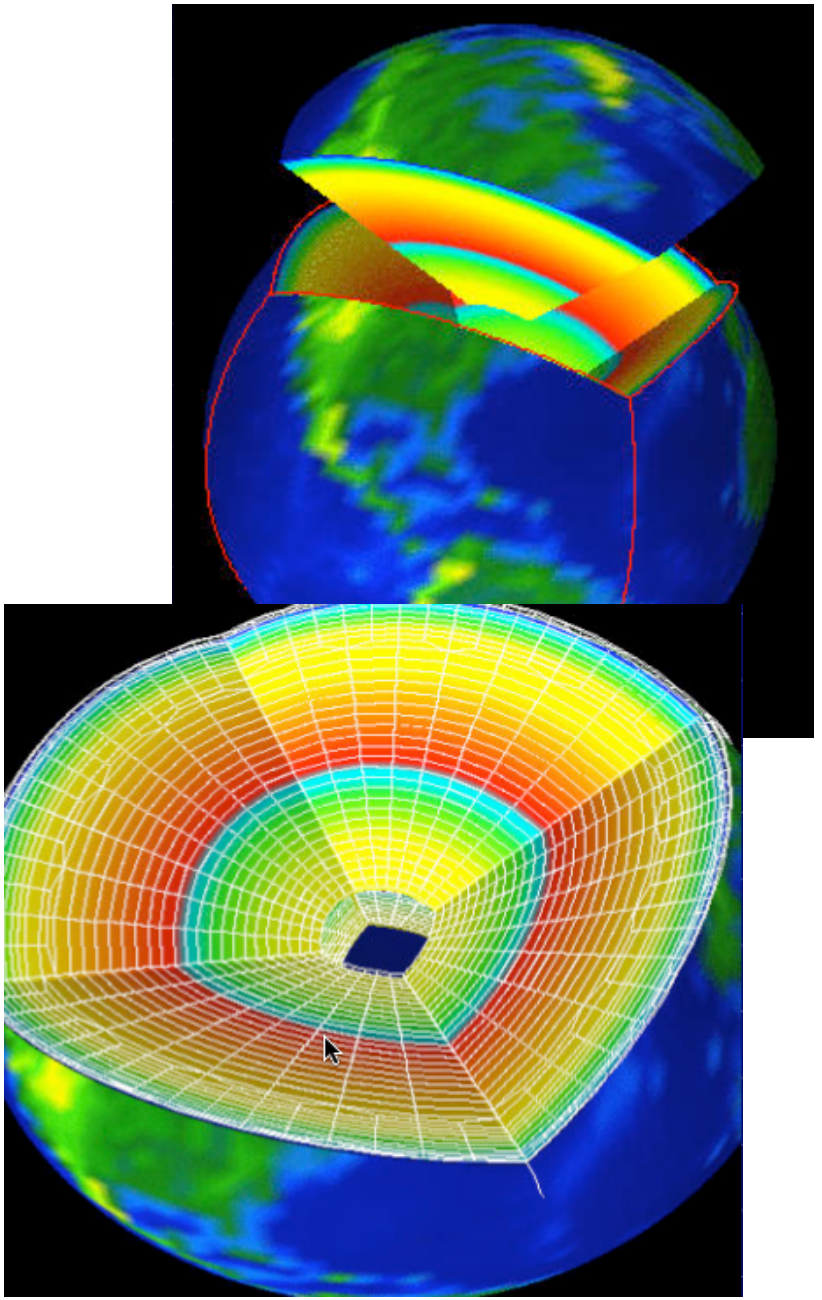
- SPECFEM3D
- GPU and parallelism
- Some results

Discuss the process of converting a Spectral Finite-Element Code to run on a Cluster of GPUs

SPECFEM3D

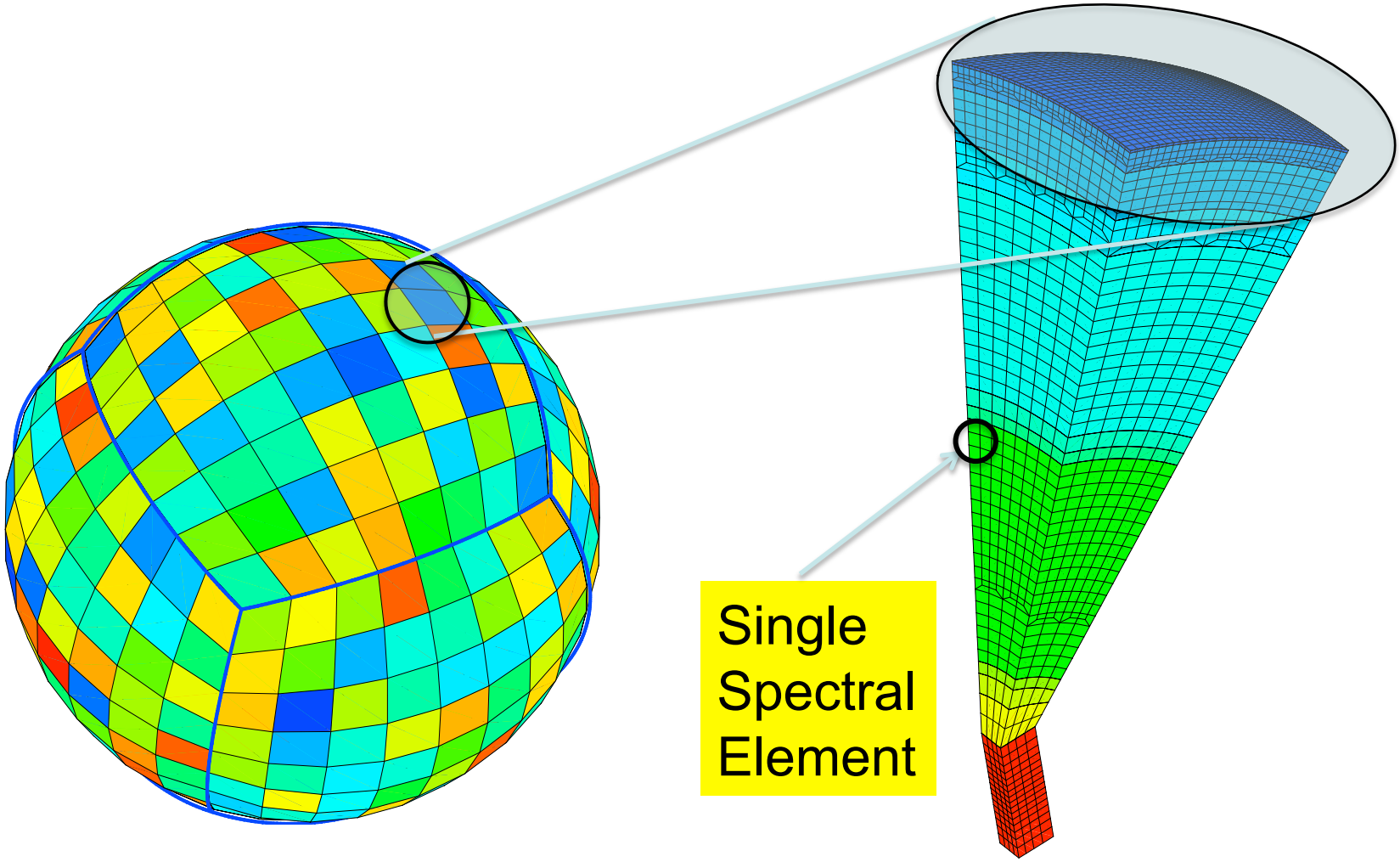
- Spectral Element Code
- Modeling of seismic wave propagation in the full earth or in densely populated regions following large earthquakes
- Developed by Dimitri Komatitsch, Jeroen Tromp and many collaborators
- Won Gordon Bell award in 2003
- Open Source.



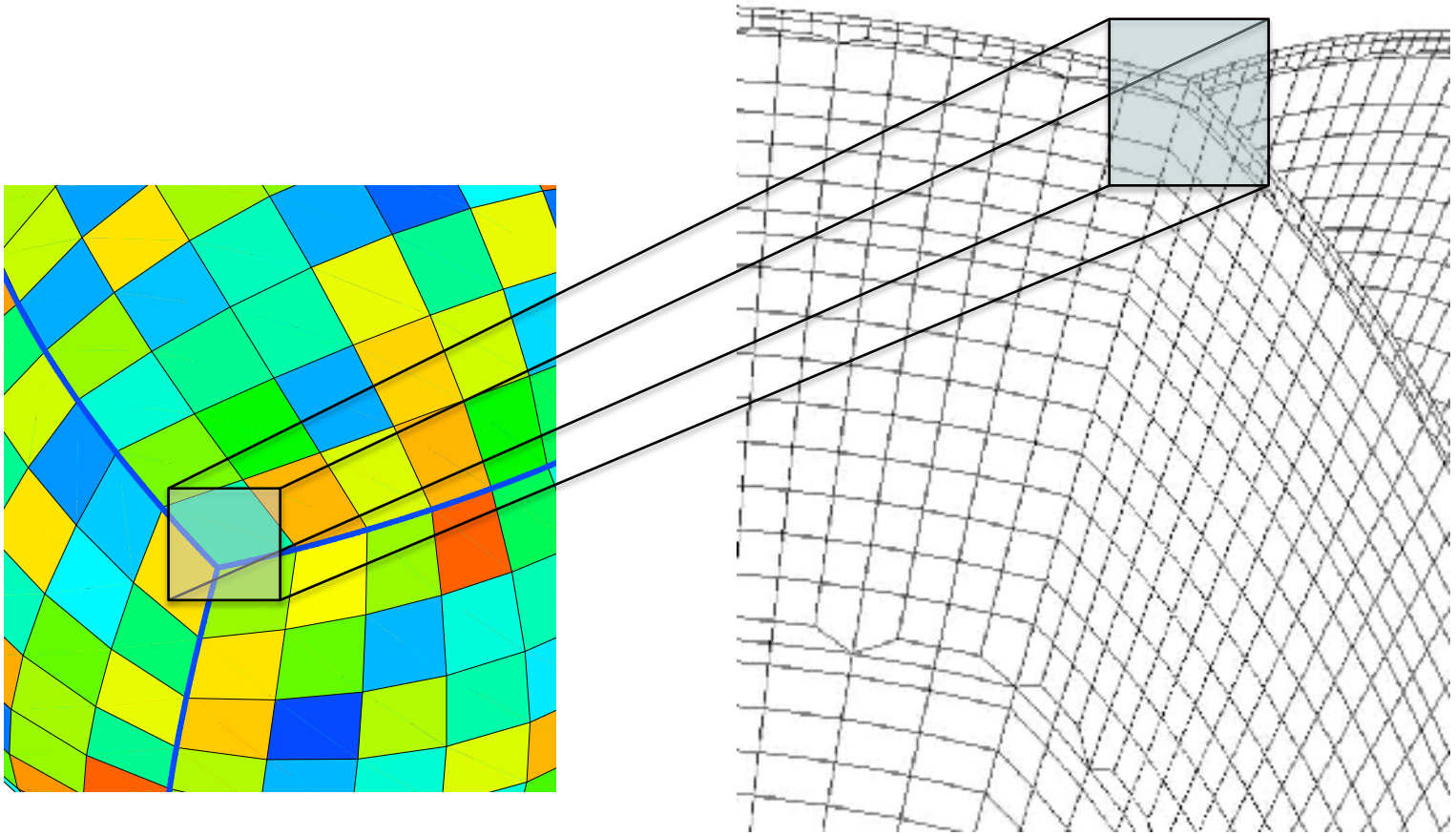


Global Grid

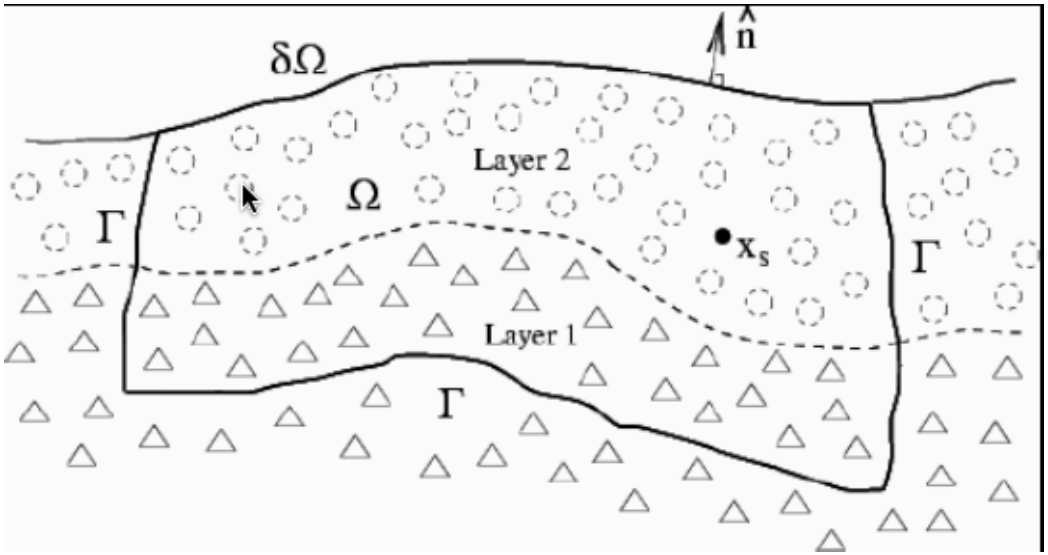
Single 3D Slice (= 1 GPU)



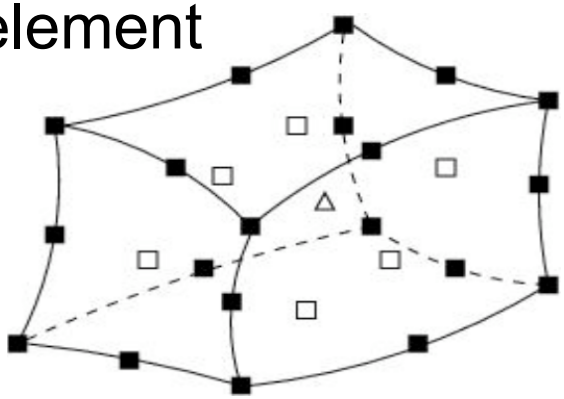
Non-Structured Mesh



Curved Physical Element



3D element



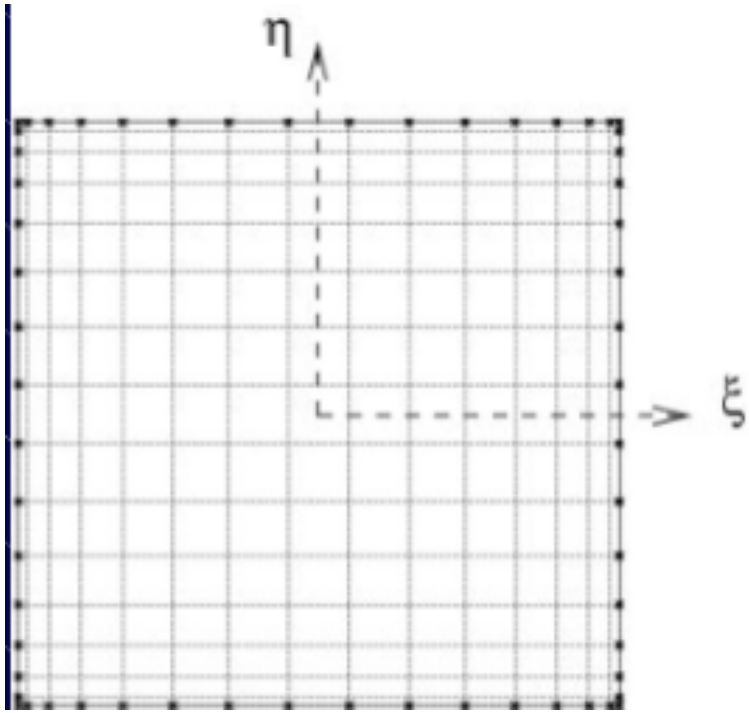
$$\mathbf{x}_n(\xi) = \sum_{n=1}^{27} N_n(\xi)\mathbf{x}_n$$

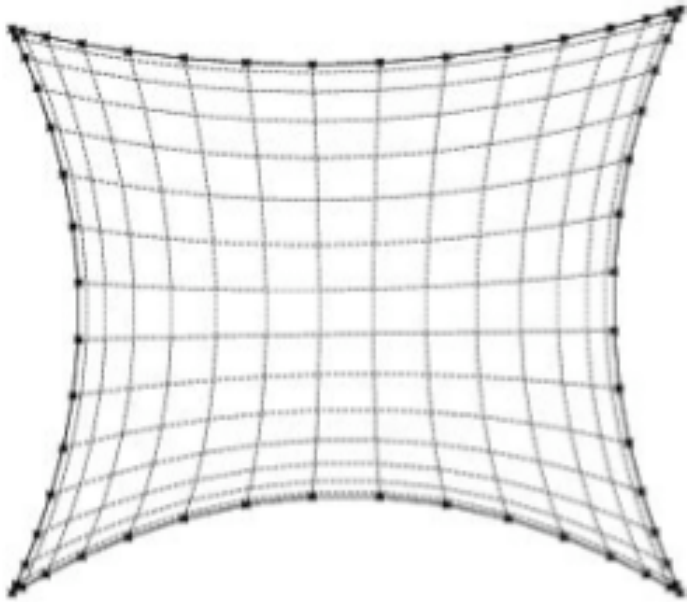
$$\xi \in [0,1]^3 \quad (\text{unit cube})$$

Gauss-Lobatto-Legendre

$$Q_N^{ijk}(\xi,\eta,\zeta) = l_N^i(\xi) l_N^j(\eta) l_N^k(\zeta)$$

$$f(x,y,z) = \sum_{i,j,k=1}^{5} Q_N^{ijk} f_{ijk} \quad \text{(4th degree Lagrange interpolants)}$$





Computational domain

Physical domain

Equations to solve

$$\rho \ddot{\mathbf{u}} = \nabla \cdot \sigma + \mathbf{f}$$

$$\sigma = \mathbf{C} : \varepsilon$$

$$\varepsilon = \frac{1}{2}\left[\nabla \mathbf{u} + (\nabla \mathbf{u})^T\right]$$

$\mathbf{u}$: displacement vector

σ : symmetric second order stress tensor

ε : symmetric second order strain tensor

$\mathbf{C}$: symmetric second order stiffness tensor

ρ : density

$\mathbf{f}$: known external force

Discrete Form per Element

$$M\ddot{u} + Ku = F$$

M : mass matrix (diagonal in this case)

K : stiffness matrix

F : external force

u : displacement vector

Full 5x5x5 matrix for each element

Many problems will reduce to this form, with different boundary conditions (I ignore boundary conditions in this talk)

Time Advancement

Time advancement is explicit

Discrete Equations

$n = 0$

while final time not reached:

for each element k :

$$Q_{ijk}^{k,n} = Q_{ijk}^{k,n} + \Delta t \; L\left(Q_{ijk}^{k,n}, t^n\right)$$ (actual code is 2/4th order)

endfor

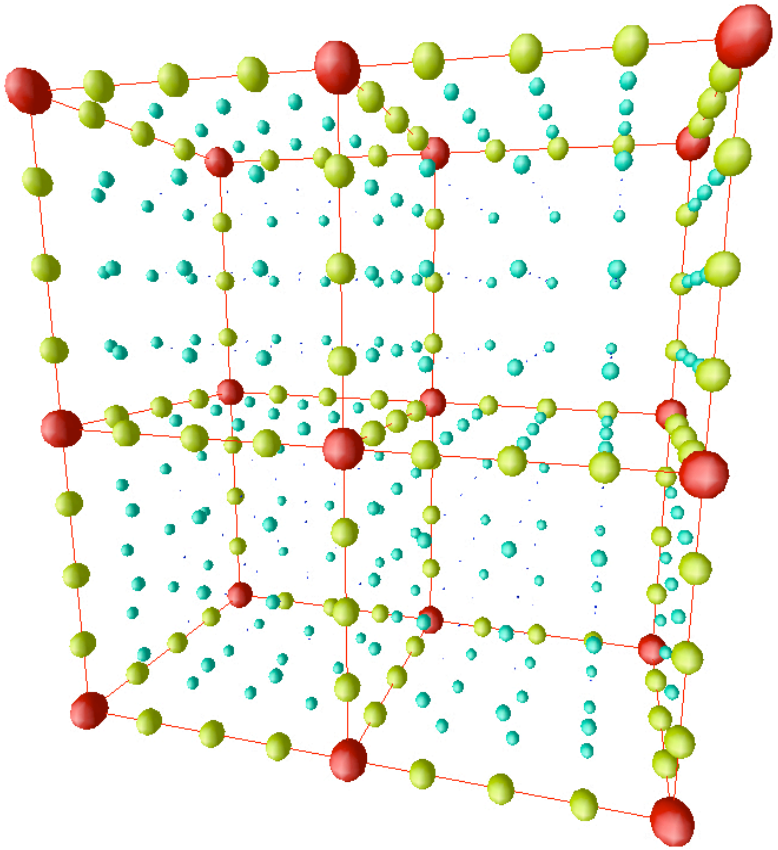
Update global points on edges and corners

Boundary conditions

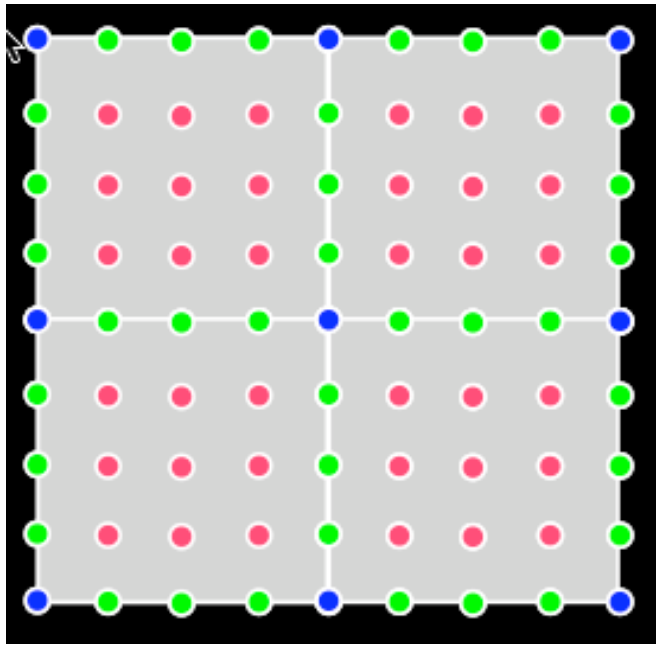
$n = n + 1$

endwhile

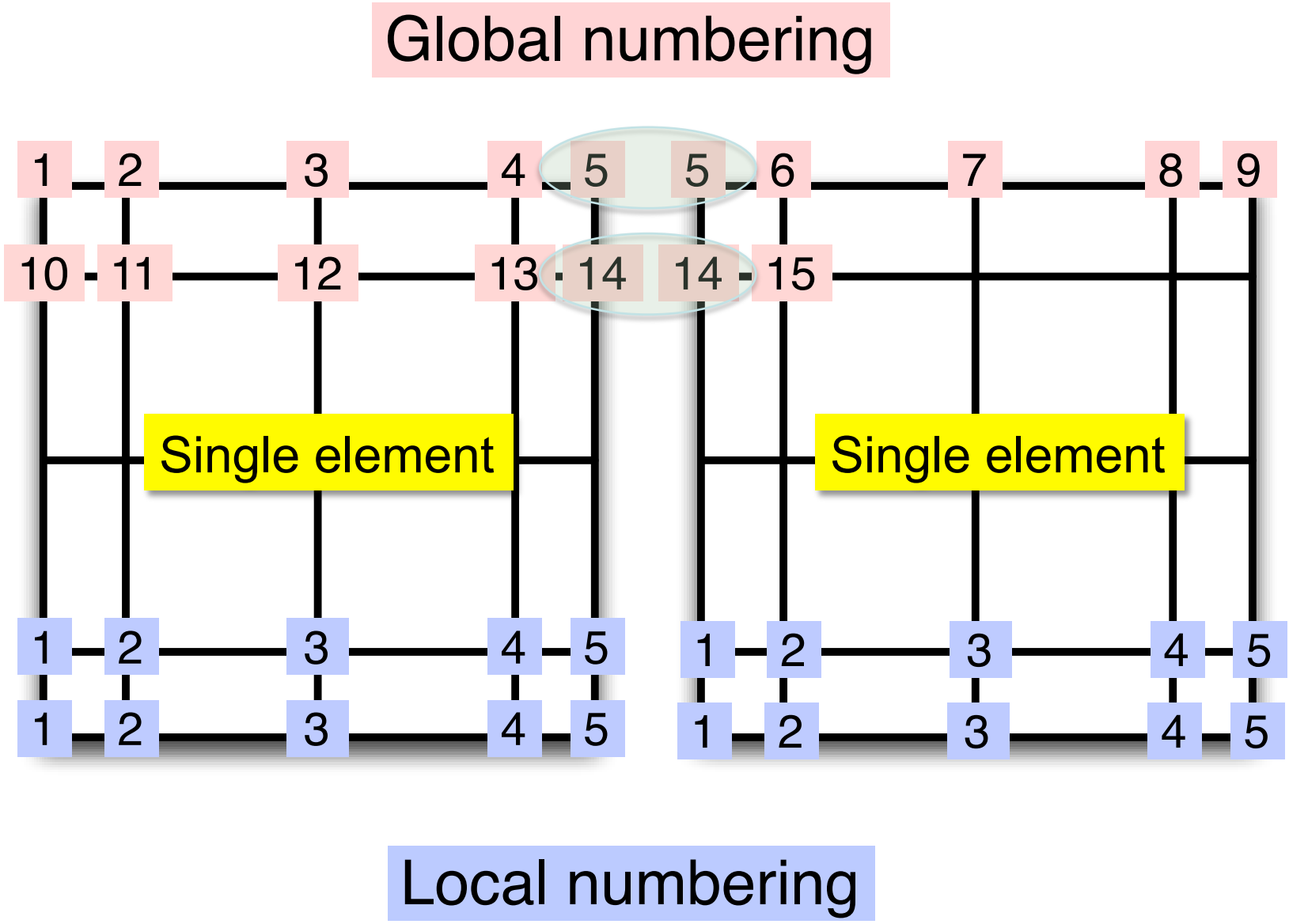
Four Spectral Elements



3D



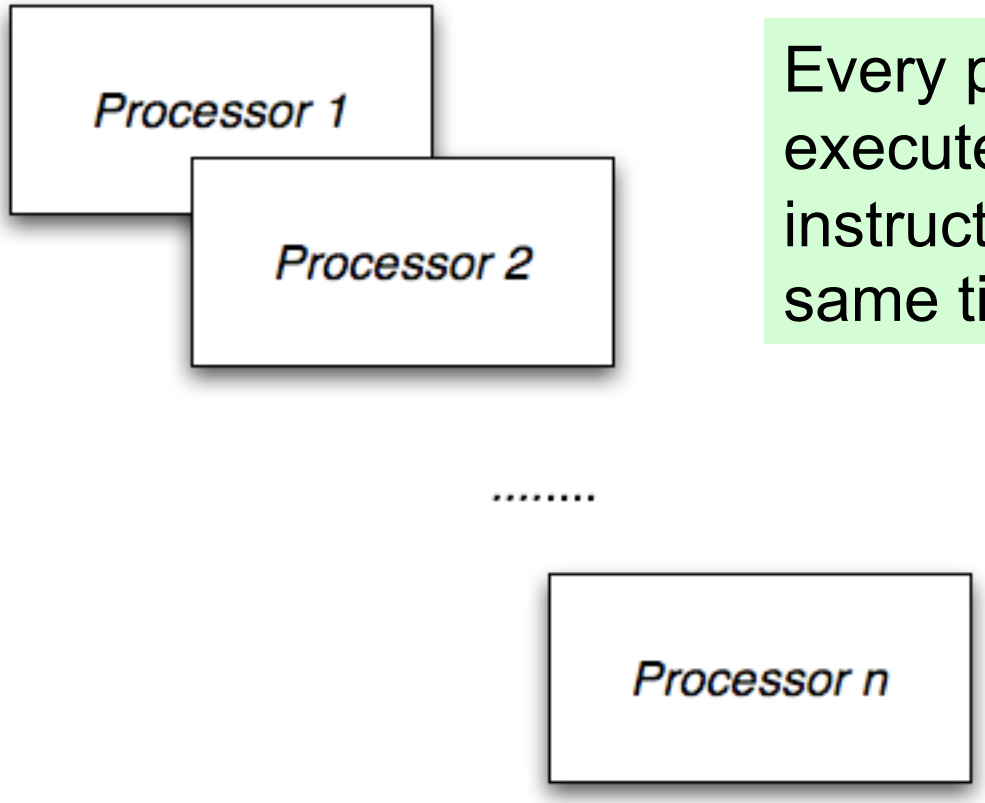
2D


Global numbering
1
2
3
4
5
5
6
7
8
9
10
11
12
13
14
14
15
Single element
Single element
1
2
3
4
5
1
2
3
4
5
1
2
3
4
5
1
2
3
4
5
Local numbering

Parallelism: Quick Tour

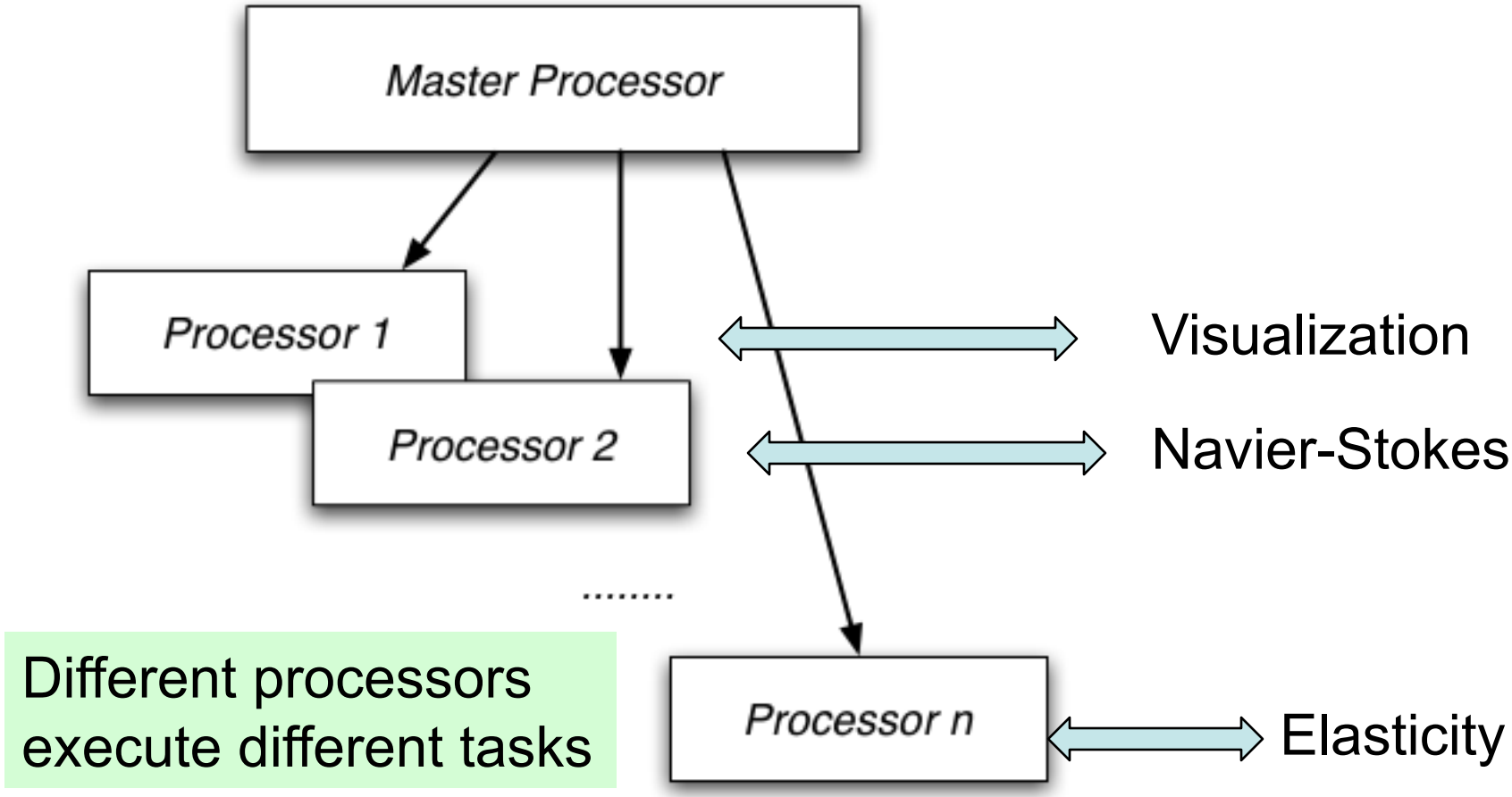
SIMD
MIMD
Stream
SIMT (T for **T**hread)

Single **I**nstruction **M**ultiple **D**ata



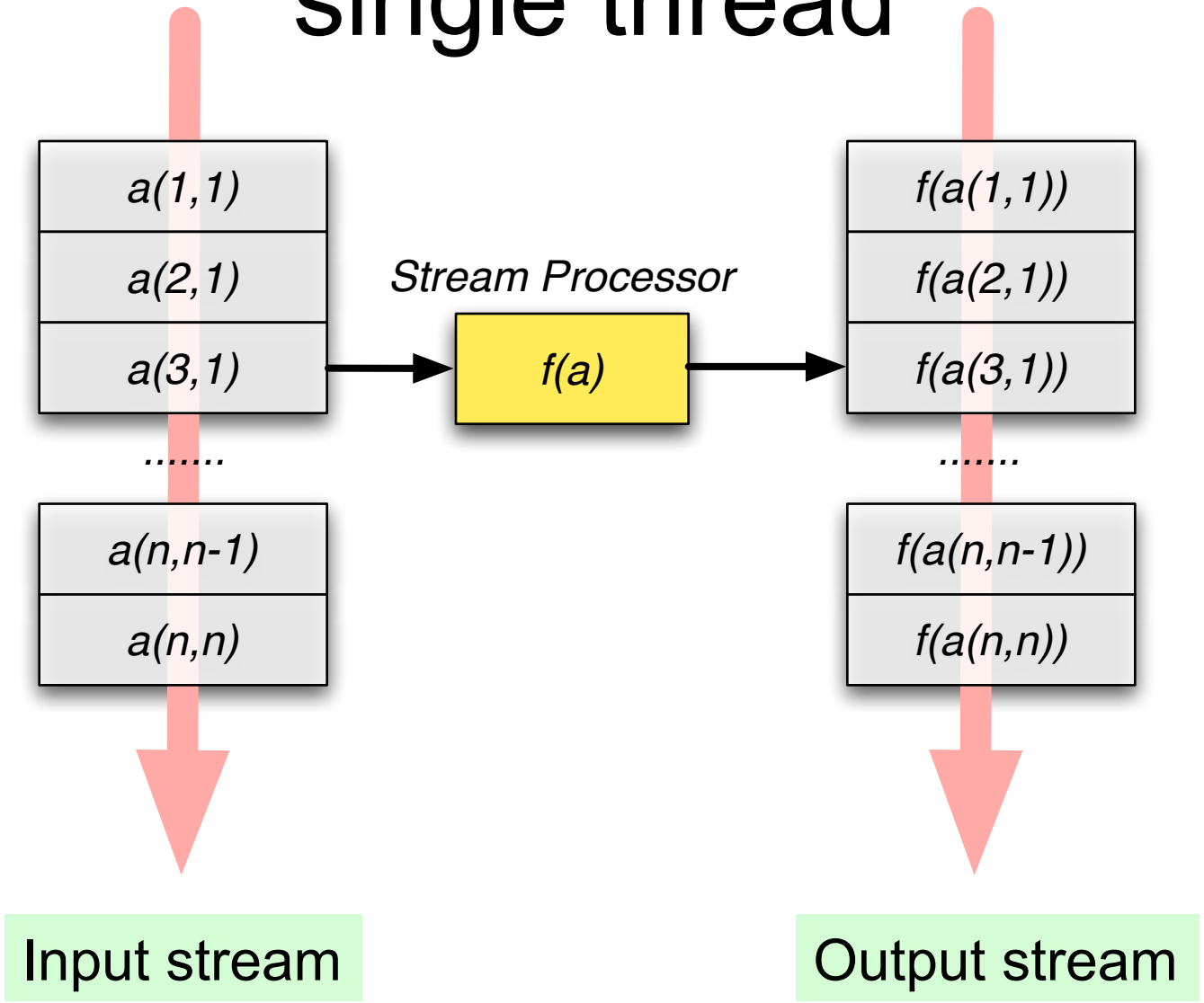
Every processor executes the identical instruction at the same time

Multiple **I**nstruction **M**ultiple **D**ata

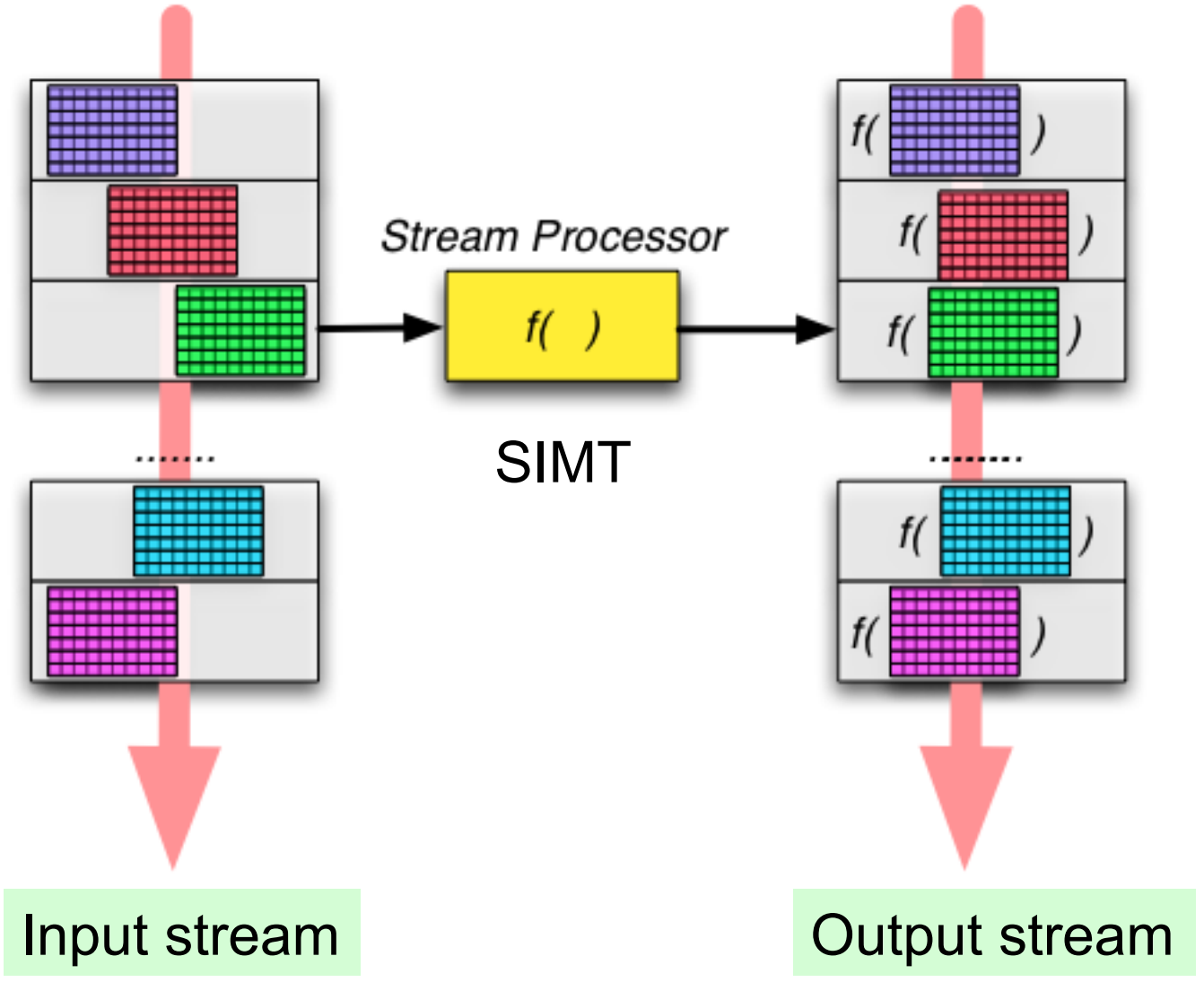


Different processors execute different tasks

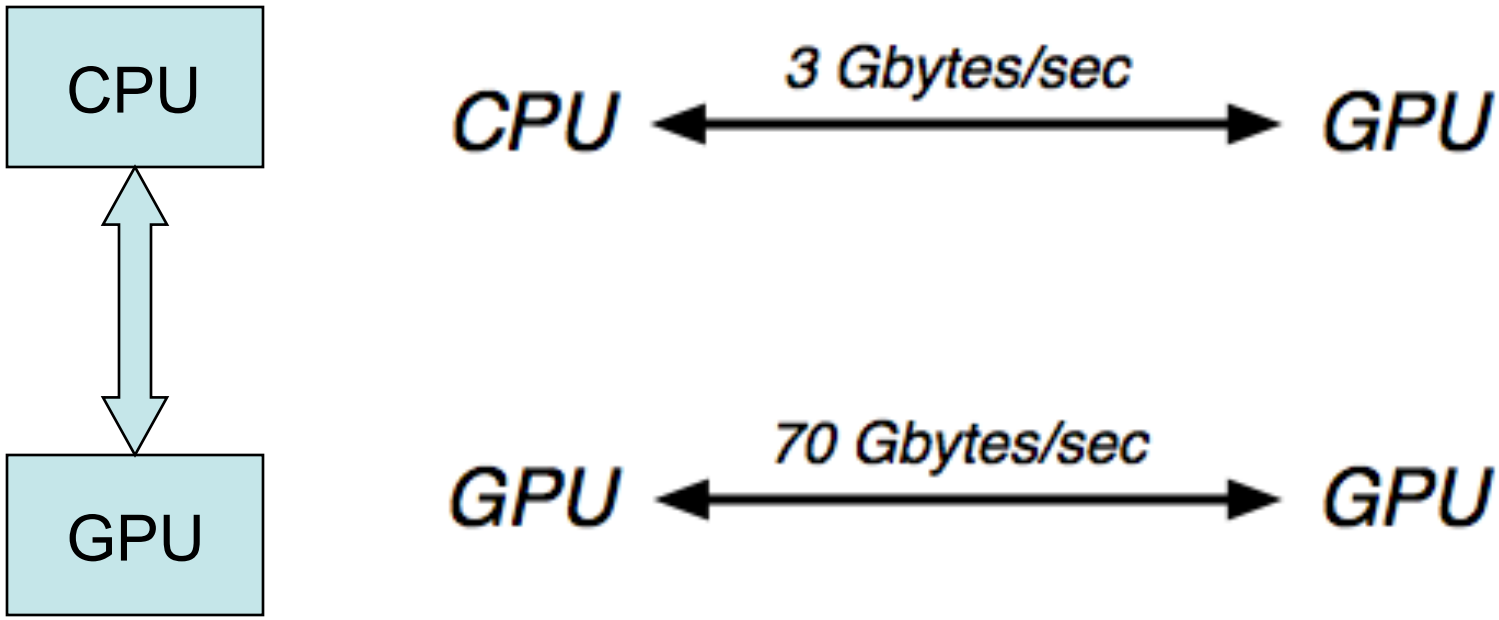
Stream Processing single thread



Stream Processing block of threads

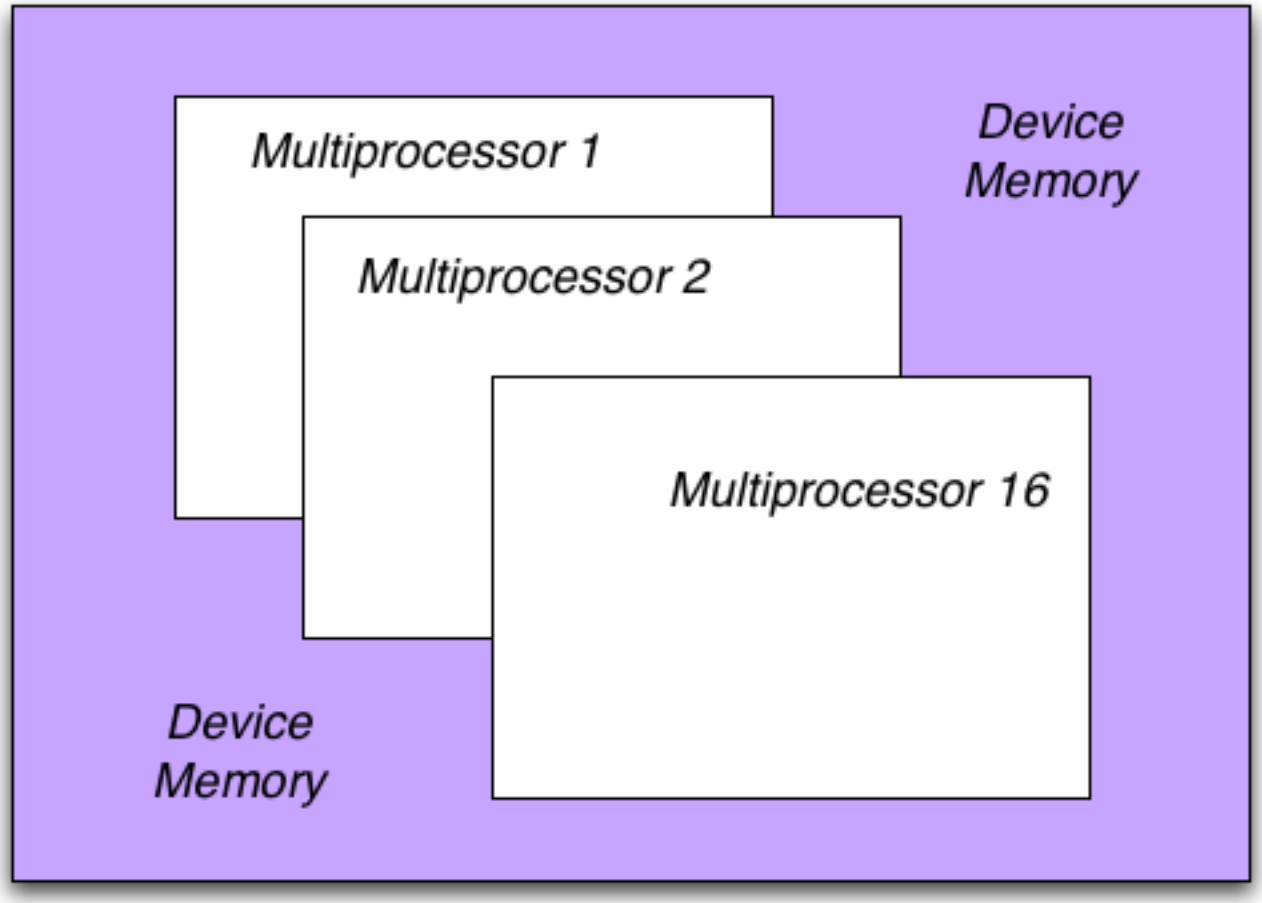


Memory Bandwidth

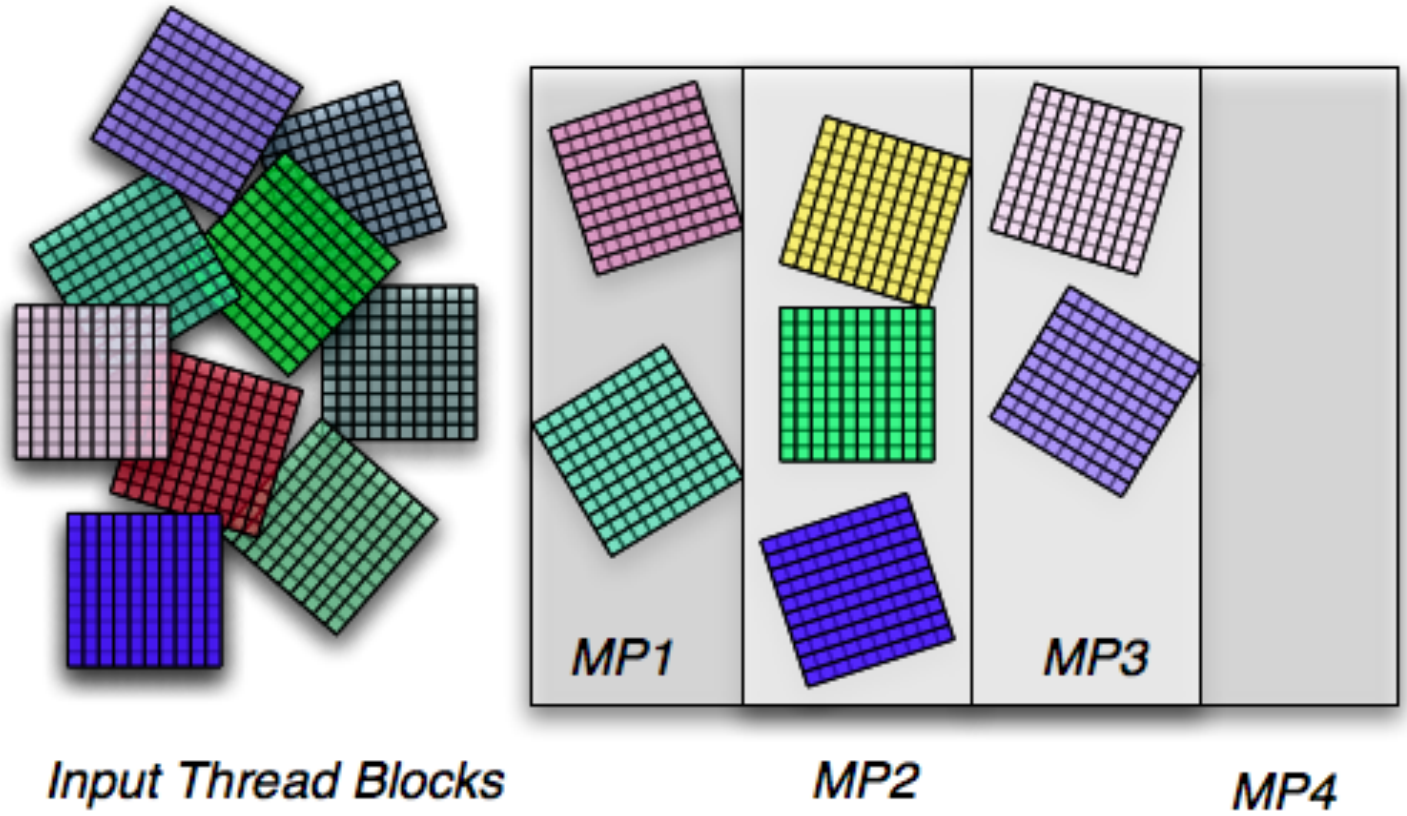


Minimize CPU to GPU transfers

Several Multiprocessors per GPU (280GTX has 24 MP)



Treatment of input blocks



CUDA

Let us return to the Spectral Finite-Element code

Paris Cluster

- CCRT/CEA/GENCI, Paris, France
- 48 Teslas S1070
- Each Tesla: 4 GT200 GPUs and two PCI Express 2 buses (*two GPUs share a PCI Express 2 bus*)
- GT200 cards: 240 cores and 4 GB of device memory
- The Teslas are connected to BULL Novascale R422 E1 nodes with two quad-core Intel Xeon Nehalem processors operating at 2.93 GHz
- Each node has 24 GB of RAM and runs Linux kernel 2.6.18.
- Infiniband network
- CUDA 2.2

Single GPU Results

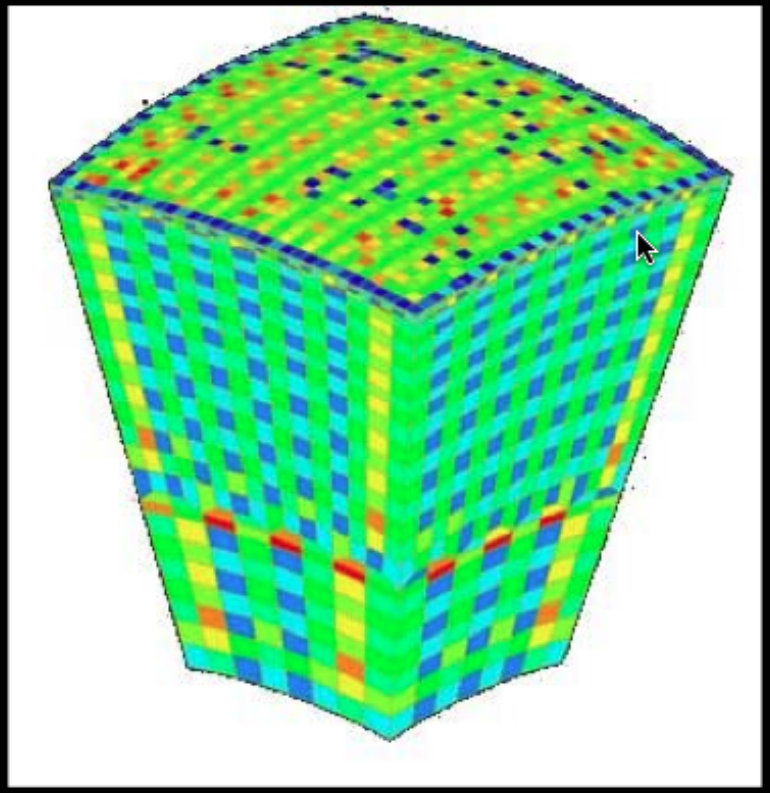
	GTX 280	8800 GTX		
	Version 1	Version 1	Version 2	
Mesh Size	speedup	speedup	speedup	% transfer
65 MB	21	13	4.5	68%
405 MB	25	15	5.3	68%
633 MB	25	15	5.3	68%

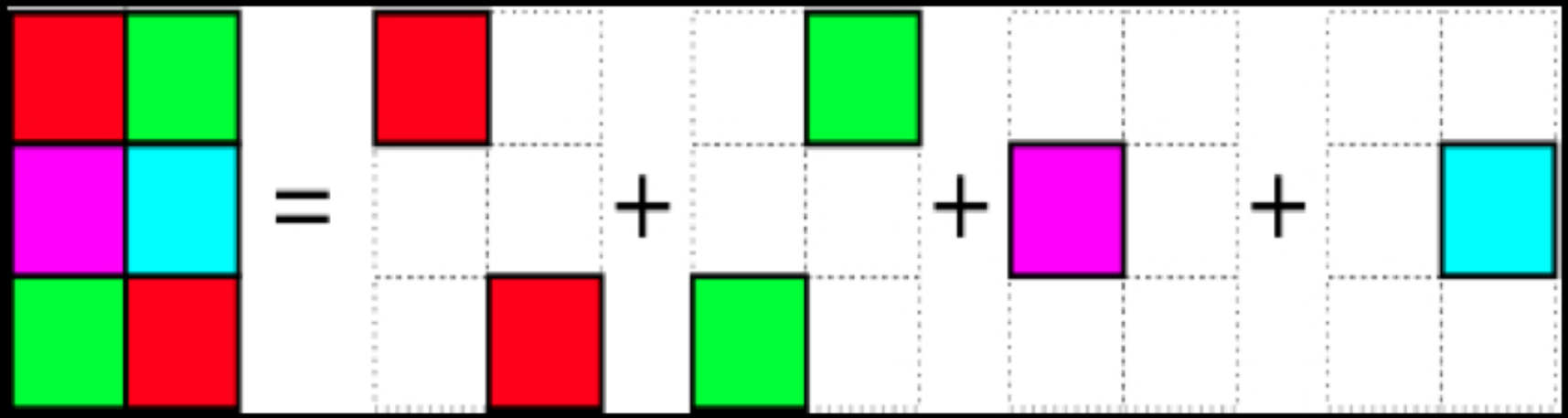
Version 1: fully on the GPU
Version 2: Problem size larger than GPU memory
Update all elements in multiple passes

%transfer: time spent in CPU to GPU data transfers

Porting SPECFEM3D on CUDA: mesh coloring

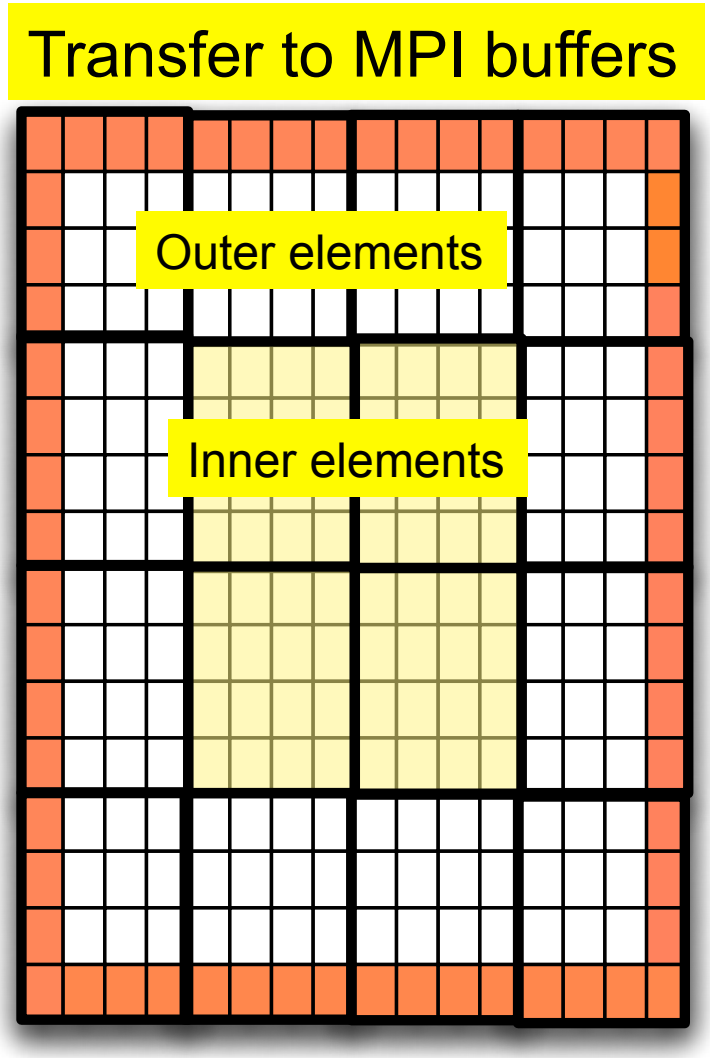
- Key challenge: ensure that contributions from two local nodes never update the same global value from different warps
- Use of mesh coloring: suppress dependencies between mesh points inside a given kernel





Overlap Communication and Computation

- Compute (on GPU) outer elements first
- Fill MPI buffers
- Issue non-blocking MPI instruction
- During MPI transfers, update inner elements on the GPU

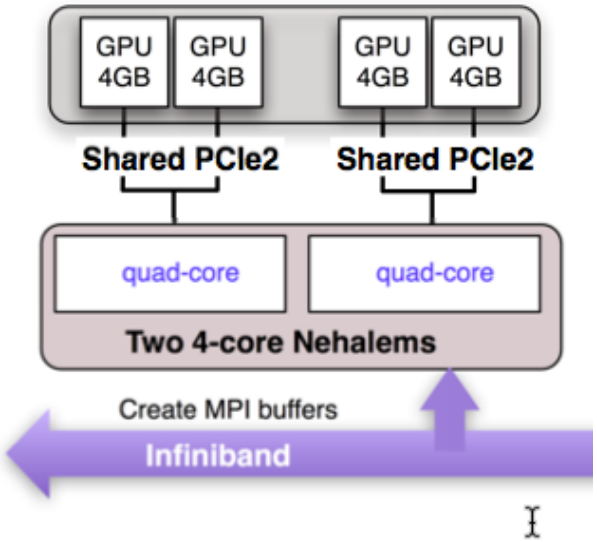


Some Weak Scaling Tests

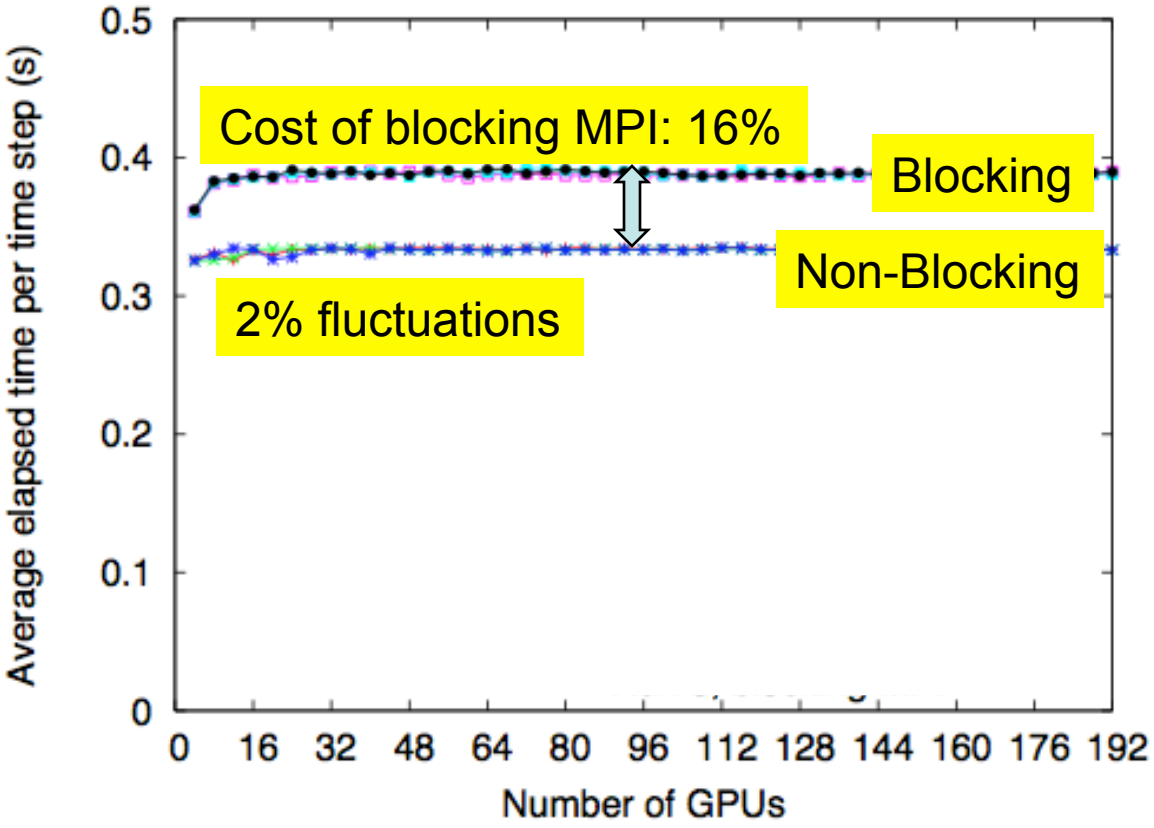
Parameters

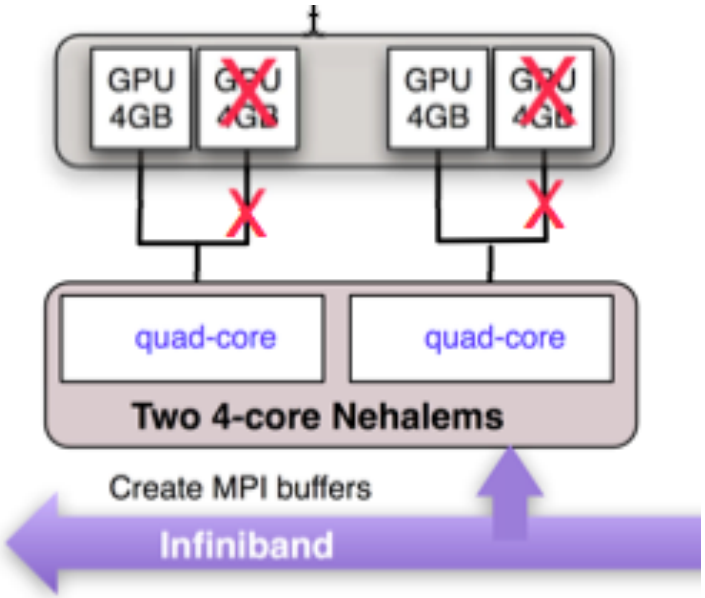
- **192 slices**
- Each slice: 446,080 spectral elements
- Total number elements: 85.6 million
- Each element: 125 points
- Unique points per slice: 29.6 million
- To number unique points: 5.6 billion
- 27% outer elements, 73% inner elements
- **Number of data points to transfer to adjacent GPUs: 12% of the slice total**

Test 1

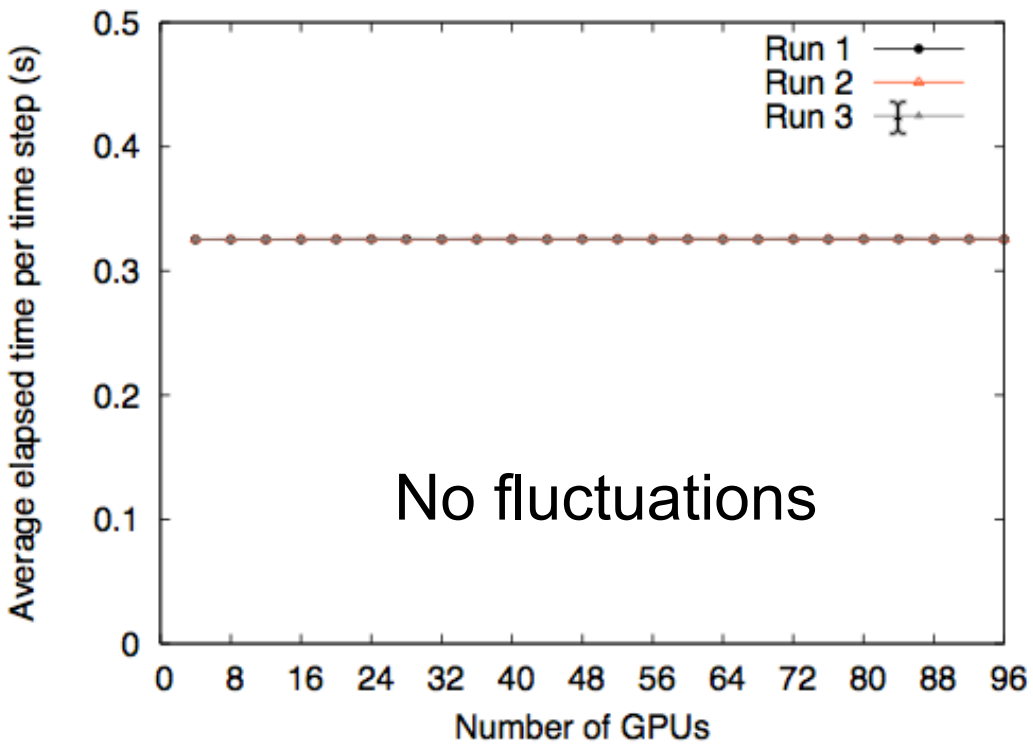


- Range: 4-192 GPUs
- Each PCI-2 bus shared by 2 GPUs
- 3 runs of test





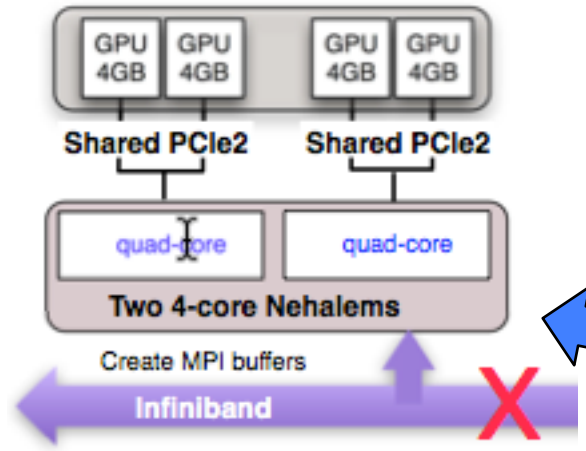
Test 2

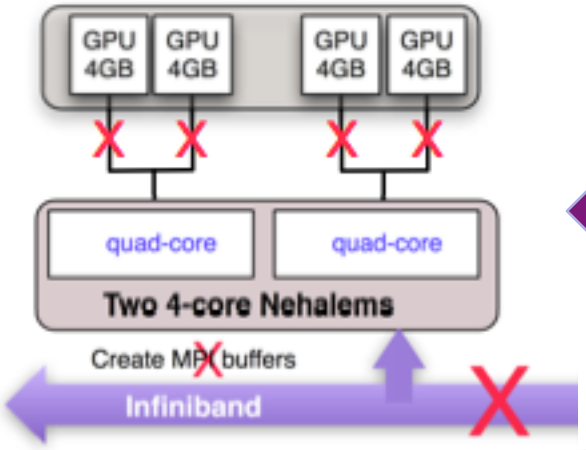


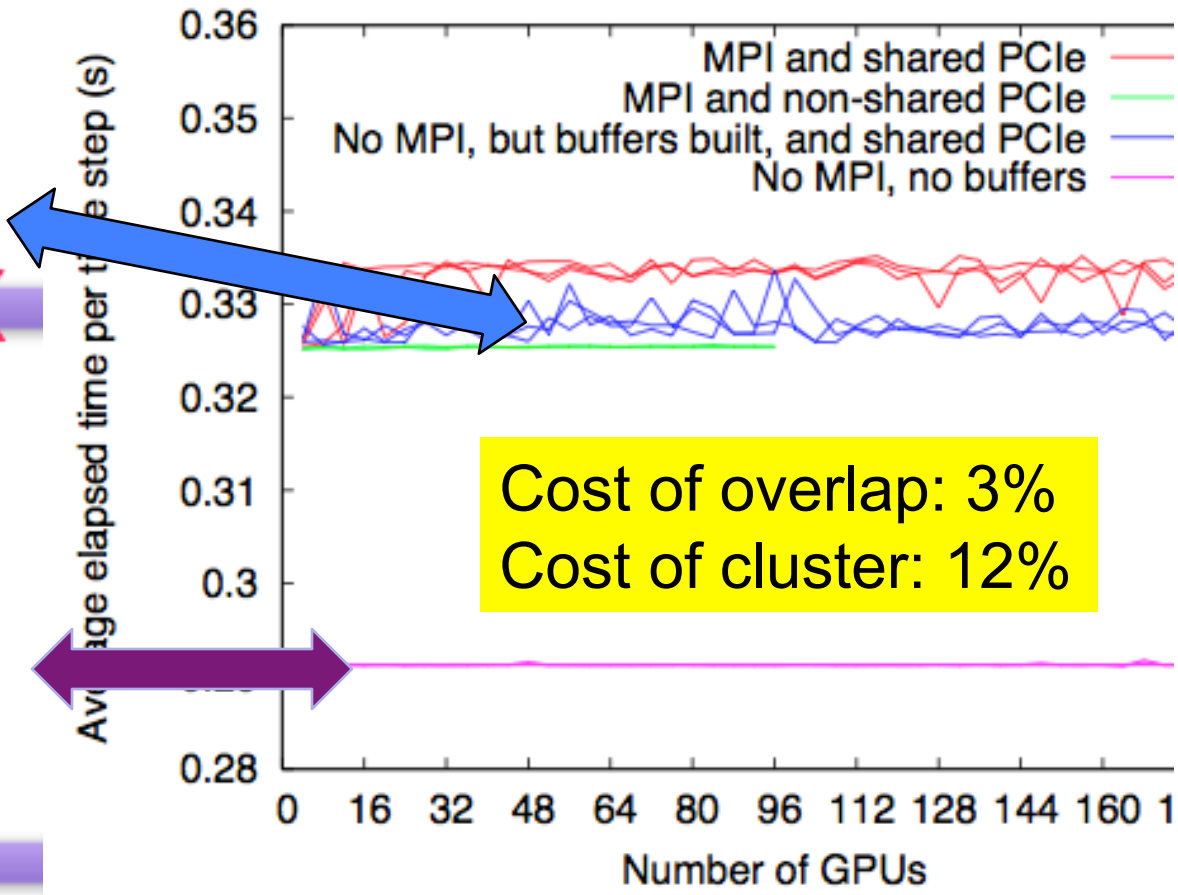
- Range: 4-96GPUs
- Each PCI-2 bus shared by 2 GPUs
- 3 runs of test

Previous fluctuations due
To sharing of PCIe bus

Tests 3-4







Cost of overlap: 3%
Cost of cluster: 12%

Conclusions

- CUDA on a single GPU leads to a speedup of 25x for our application
- 25x speedup is maintained when we use a cluster of GPUs with non-blocking MPI
- Crucial to use larger domains to compensate for the very high speedup offered by GPUs
- OpenCL (less efficient, more portable)
- Above results are valid in 2010
- Benchmarking not yet performed on most recent GPUs.