Rethinking Visualization Frameworks for Extreme-Scale Computing

Ultrascale Visualization Workshop

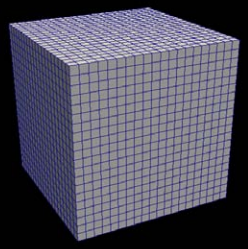
November 15, 2010

Kenneth Moreland
Sandia National Laboratories

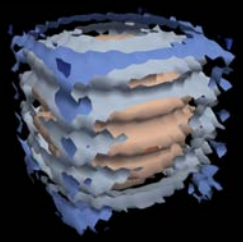
Stéphane Marchesin
University of California at Davis

SAND 2010-8034P

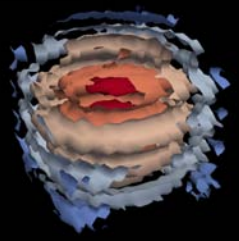
Serial Visualization Pipeline



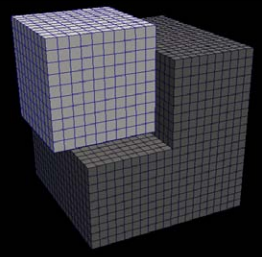
Contour



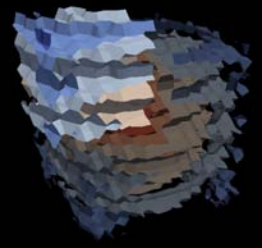
Clip



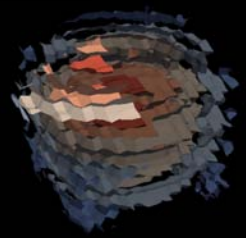
Parallel Visualization Pipeline



Contour | Contour | Contour



Clip | Clip | Clip



Exascale Projection

	Jaguar – XT5	Exascale*	Increase
Cores	224,256	100 million – 1 billion	~1,000×
Concurrency	224,256 way	10 billion way	~50,000×
Memory	300 Terabytes	128 Petabytes	~500×

*Source: International Exascale Software Project Roadmap, J. Dongarra, P. Beckman, et al.

Exascale Projection

	Jaguar – XT5	Exascale*	Increase
Cores	224,256	100 million – 1 billion	~1,000×
Concurrency	224,256 way	10 billion way	~50,000×
Memory	300 Terabytes	128 Petabytes	~500×

MPI Only?

Vis object code + state: 20MB

On Jaguar: 20MB × 200,000 processes = 4TB

On Exascale: 20MB × 10 billion processes = 200PB !

*Source: International Exascale Software Project Roadmap, J. Dongarra, P. Beckman, et al.

Exascale Projection

	Jaguar – XT5	Exascale*	Increase
Cores	224,256	100 million – 1 billion	~1,000×
Concurrency	224,256 way	10 billion way	~50,000×
Memory	300 Terabytes	128 Petabytes	~500×

Visualization pipeline too heavyweight?

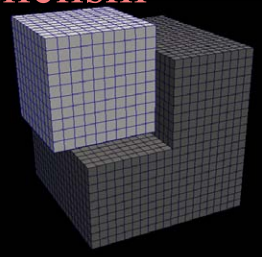
On Jaguar: 1 trillion cells → 5 million cells/core

On Exascale: 500 trillion cells → 50K cells/core

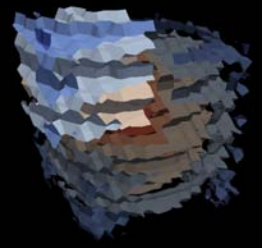
*Source: International Exascale Software Project Roadmap, J. Dongarra, P. Beckman, et al.

Hybrid Parallel Pipeline

Distributed Memory Parallelism

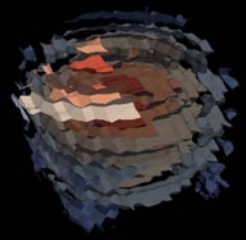


Contour | Contour | Contour



Shared Memory
Parallel Processing

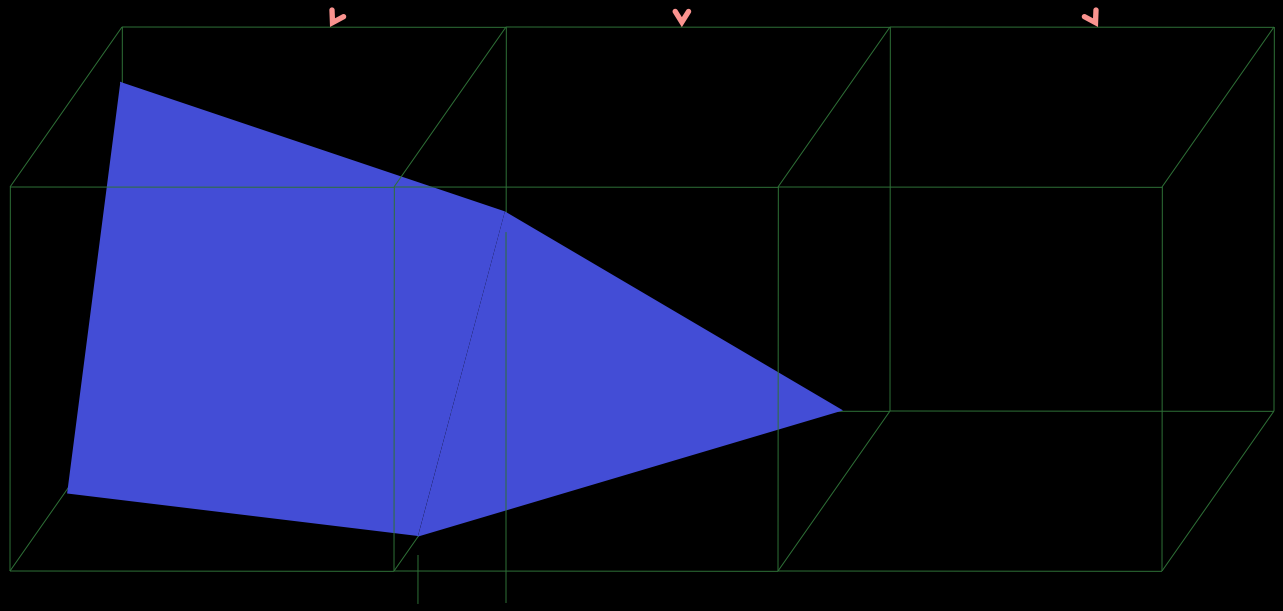
Clip | Clip | Clip



Threaded Programming is Hard
Example: Marching Cubes

Sandia National Laboratories

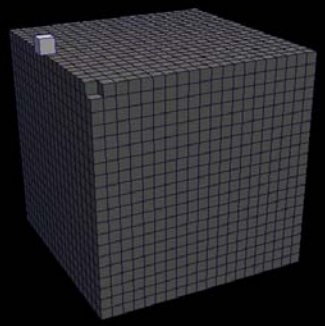
Easy because cubes can be processed in parallel, right?



How do you resolve coincident points?
How do you capture topological connections?

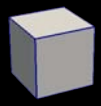
How do you pack the results?

Revisiting the Pipeline



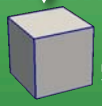
- Lightweight Object
- Serial Execution
- No explicit partitioning
- No access to larger structures
- No state

Sandia National Laboratories

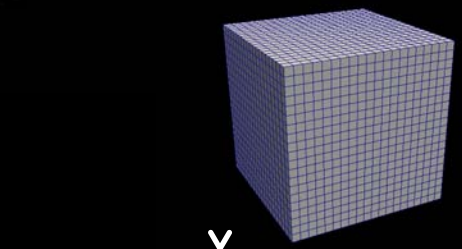
function (in  , out )

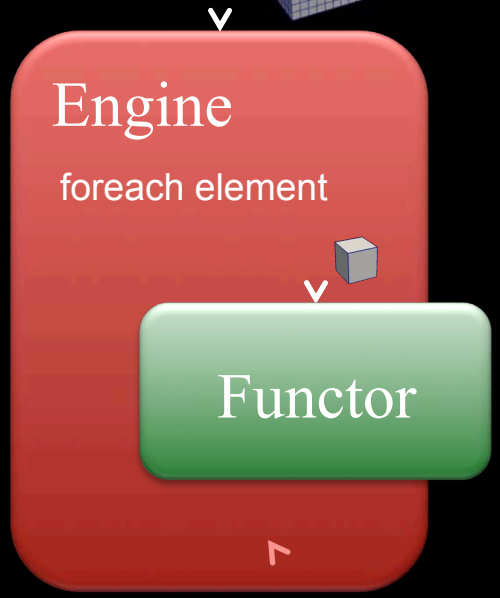
Sandia National Laboratories

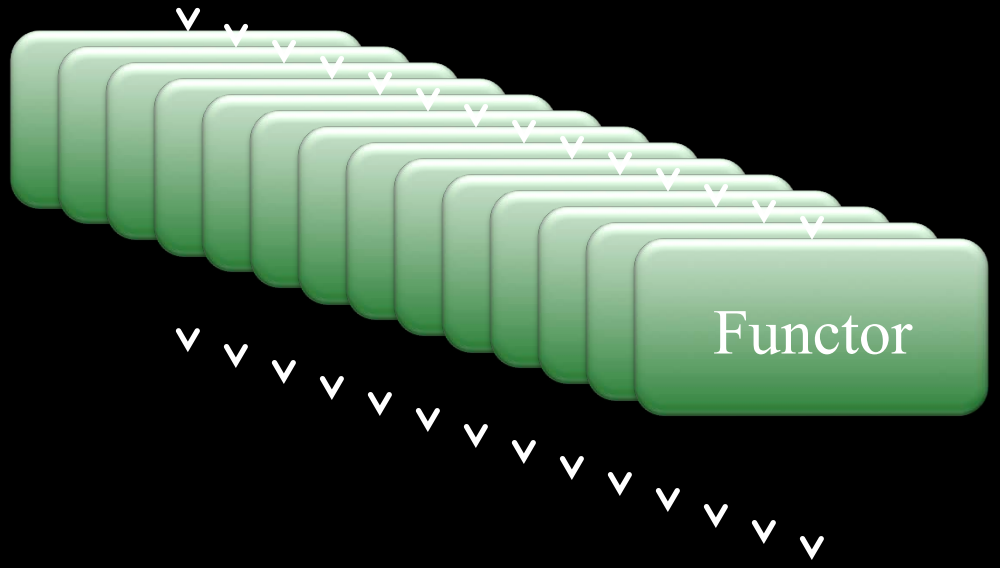
Functor

function (in  , out )

Iteration Mechanism



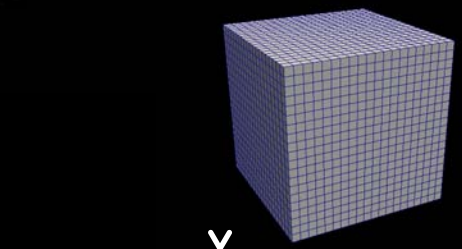




Conceptual Iteration

Reality: Iterations can be scheduled in parallel.

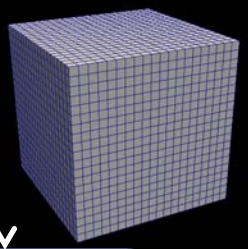
Comparison



Engine

foreach element

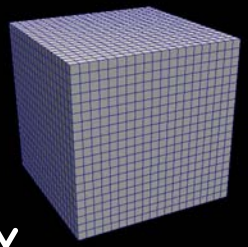
Functor



Filter

foreach element

Comparison

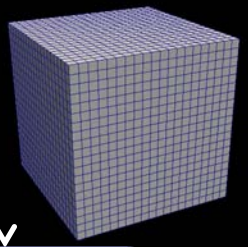


Engine

foreach element

Functor 1

Functor 2



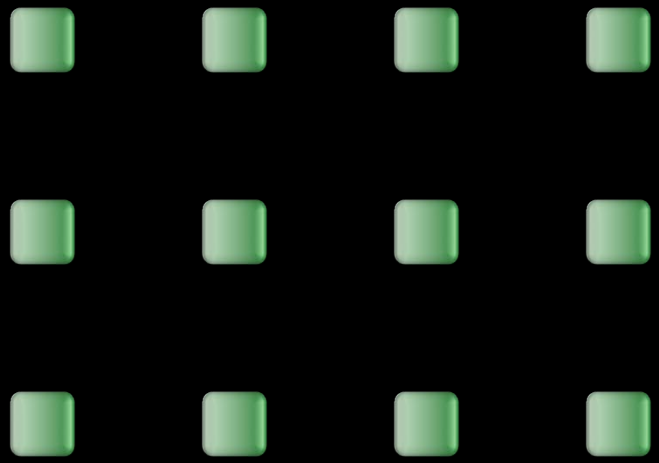
Filter 1

foreach element

Filter 2

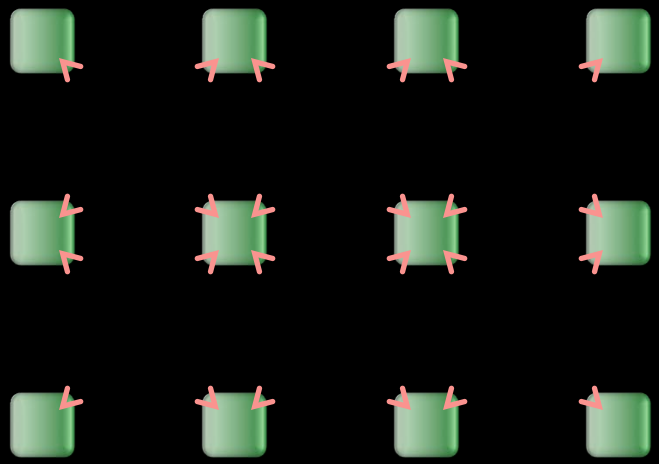
foreach element

Engine Types: Map



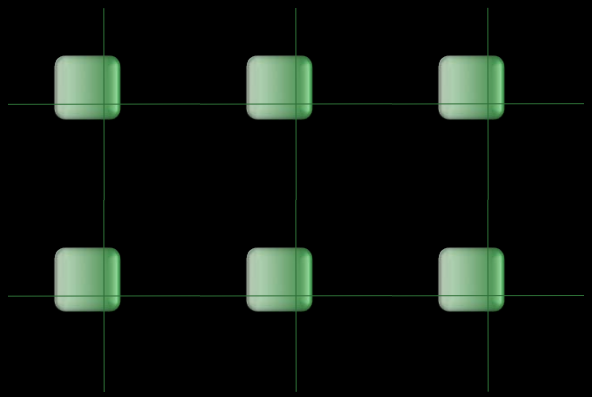
Example Usage: Vector Magnitude

Engine Type: Topological Reduce



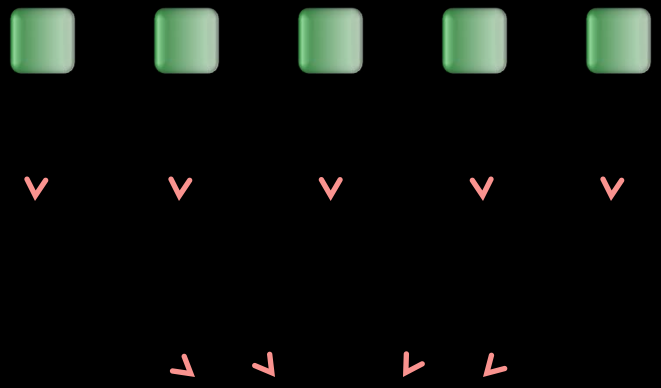
Example Usages: Cell to Point, Normal Generation

Engine Types: Generate Geometry



Example Usages: Subdivide, Marching Cubes

Engine Types: Compress



Example Usage: Marching Cubes

Conclusion

- Why now? Why not before?
  - Rules of efficiency have changed.
- Concurrency: Coarse → Fine
- Execution cycles become free
- Minimizing DRAM I/O critical
- The current approach is unworkable
  - The incremental approach is unmanageable
- Designing for exascale requires lateral thinking