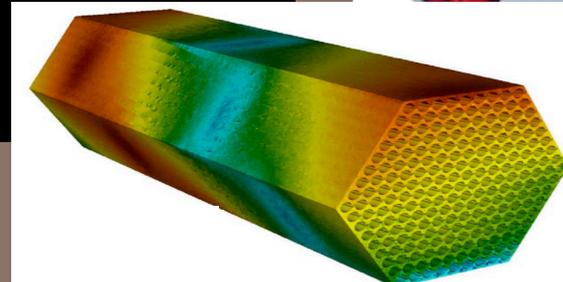
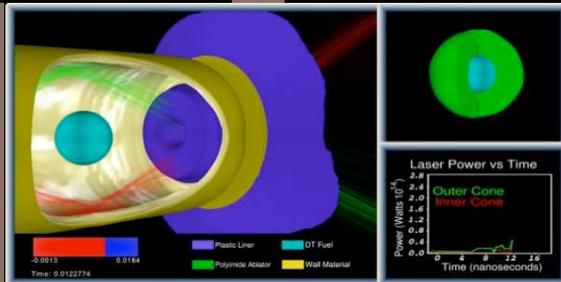
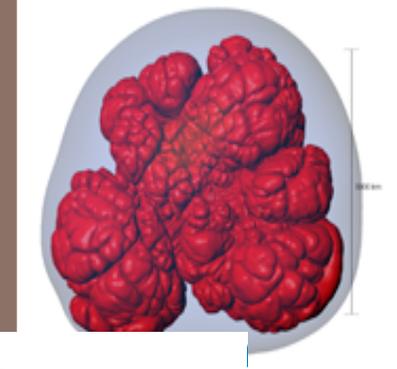
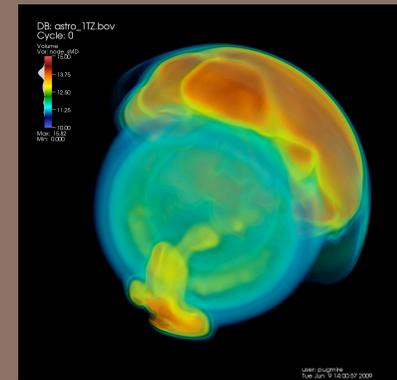
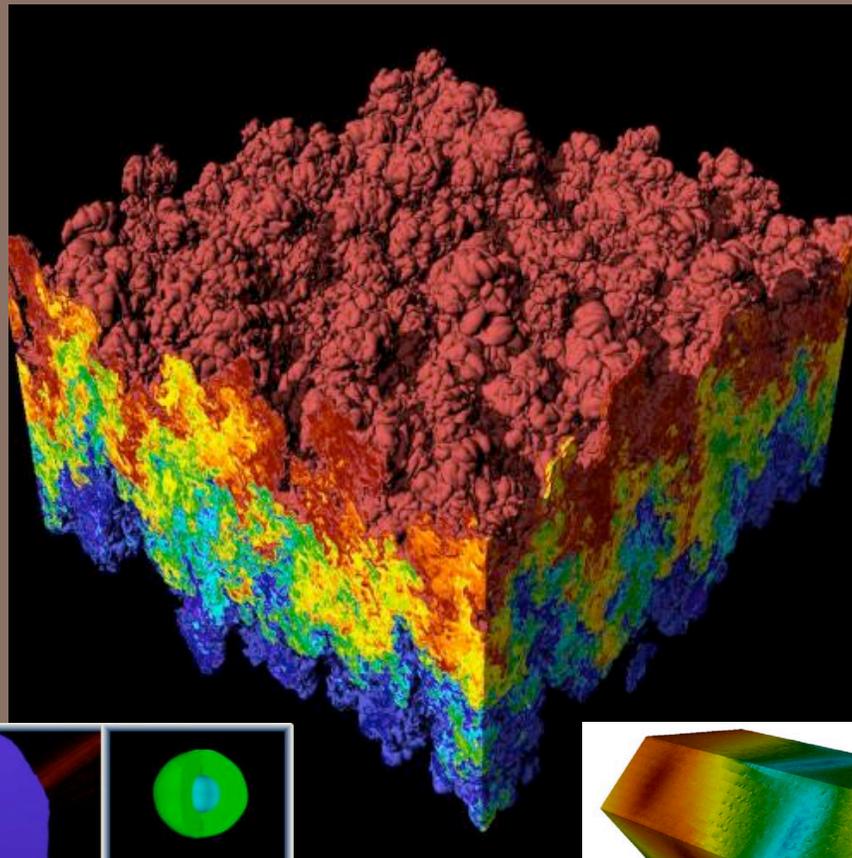
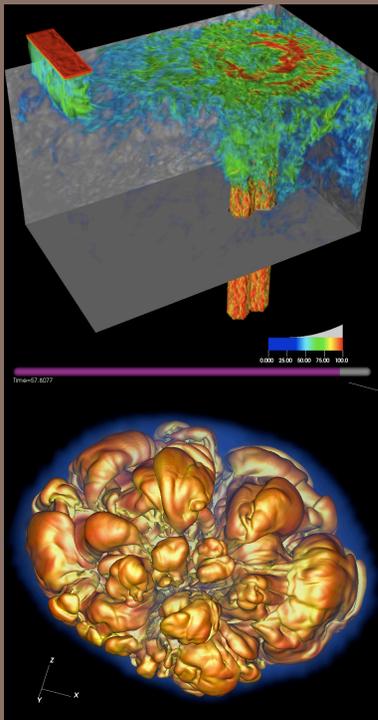


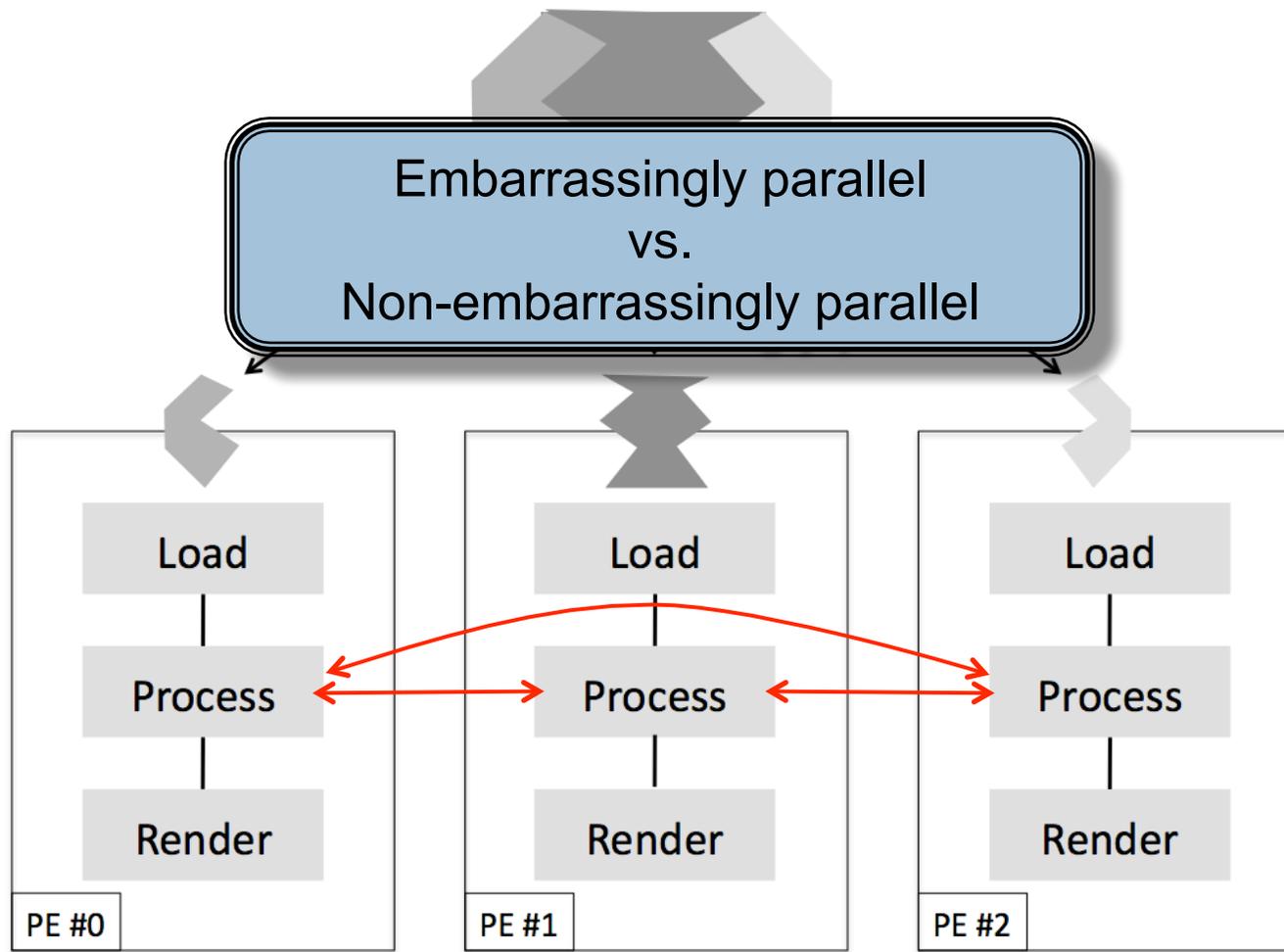
Hybrid Parallelism and Visualization



Sep 16th, 2013

Hank Childs, University of Oregon & Lawrence Berkeley

We achieve parallelism by parallelizing over pieces of data.



PE = Processing Element

Outline



- Overview:
 - What is hybrid parallelism?
- Motivation:
 - Why is hybrid parallelism so important?
- Results:
 - What has been demonstrated with hybrid parallelism so far?
- Challenges:
 - What work still needs to happen?

Outline

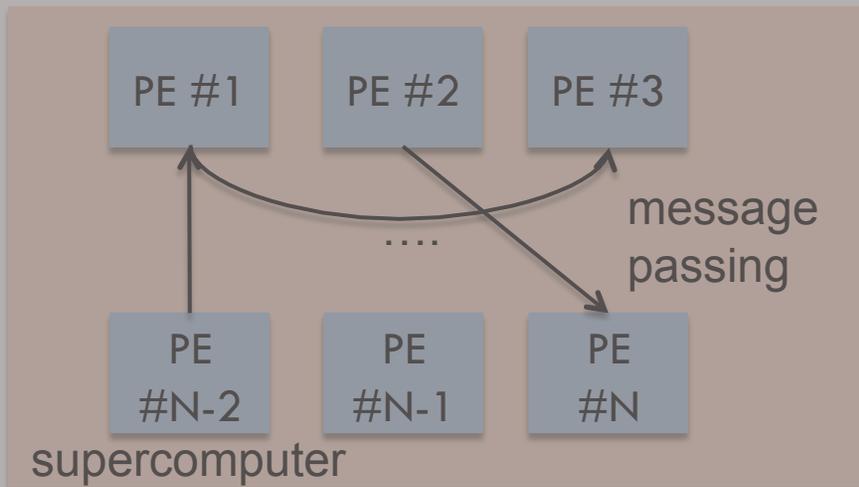


- **Overview:**
 - **What is hybrid parallelism?**
- **Motivation:**
 - Why is hybrid parallelism so important?
- **Results:**
 - What has been demonstrated with hybrid parallelism so far?
- **Challenges:**
 - What work still needs to happen?

What is hybrid parallelism?

- Hybrid parallelism combines distributed- and shared-memory techniques.

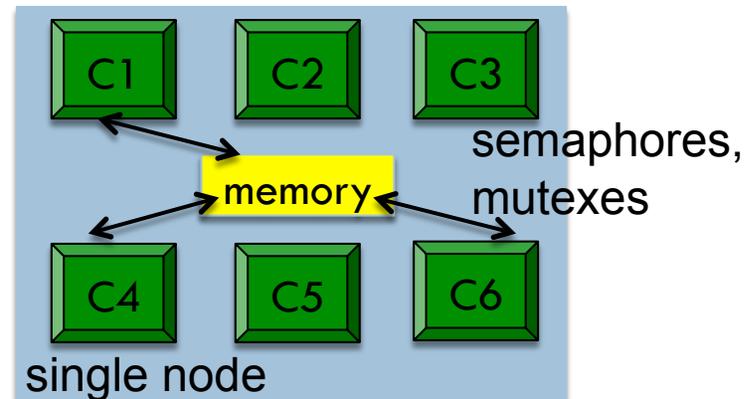
Distributed-memory parallelism



Examples: MPI (Message Passing Interface)

PE = Processing Element

Shared-memory parallelism



Examples: pthreads, OpenMP,

There are different types of shared-memory parallelism.

Definitions



- Core: a processing thread on a CPU
- Node: a group of cores that share memory
- Processing Element (PE): one instance (of many) of a distributed memory parallel program

Brief Historical Perspective

- Mid 1970s-mid 1990s:
 - ▣ Vector machines: Cray 1 ... NEC SX
 - ▣ Vectorizing Fortran compilers help optimize $a[i]=b[i]*x+c$.



Brief Historical Perspective

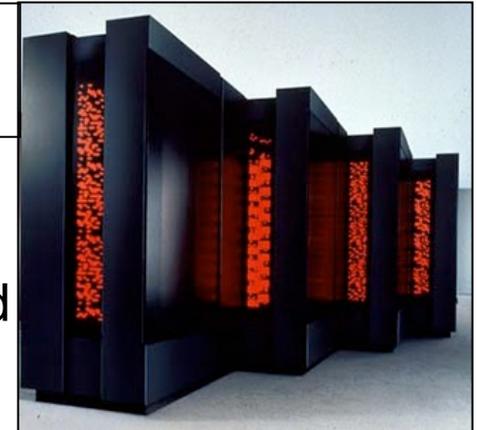


- Mid 1970s-mid 1990s:
 - ▣ Vector machines: Cray 1 ... NEC SX
 - ▣ Vectorizing Fortran compilers help optimize $a[i]=b[i]*x+c$.

- Early 1990s-present:

Massive programming investment!!

- ▣ The rise of MPP (massively parallel processing), based on the commodity microprocessor.
- ▣ Message Passing Interface (MPI) becomes the gold standard for building/running parallel codes on MPPs.



Brief Historical Perspective

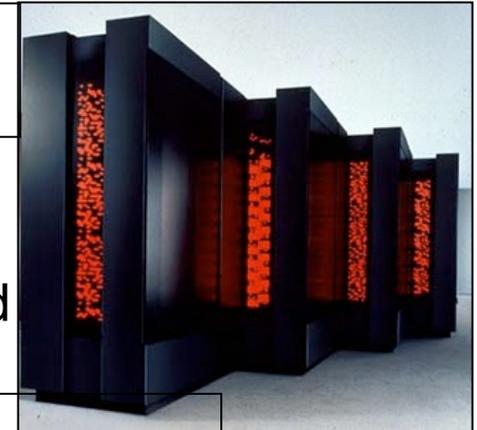
- Mid 1970s-mid 1990s:
 - ▣ Vector machines: Cray 1 ... NEC SX
 - ▣ Vectorizing Fortran compilers help optimize $a[i]=b[i]*x+c.$



Massive programming investment!!

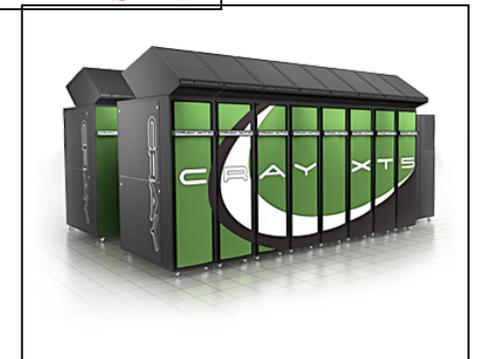
... massively parallel processing),
... multi-processor.

... Message Passing Interface (MPI) becomes the gold
... standard for running parallel codes on

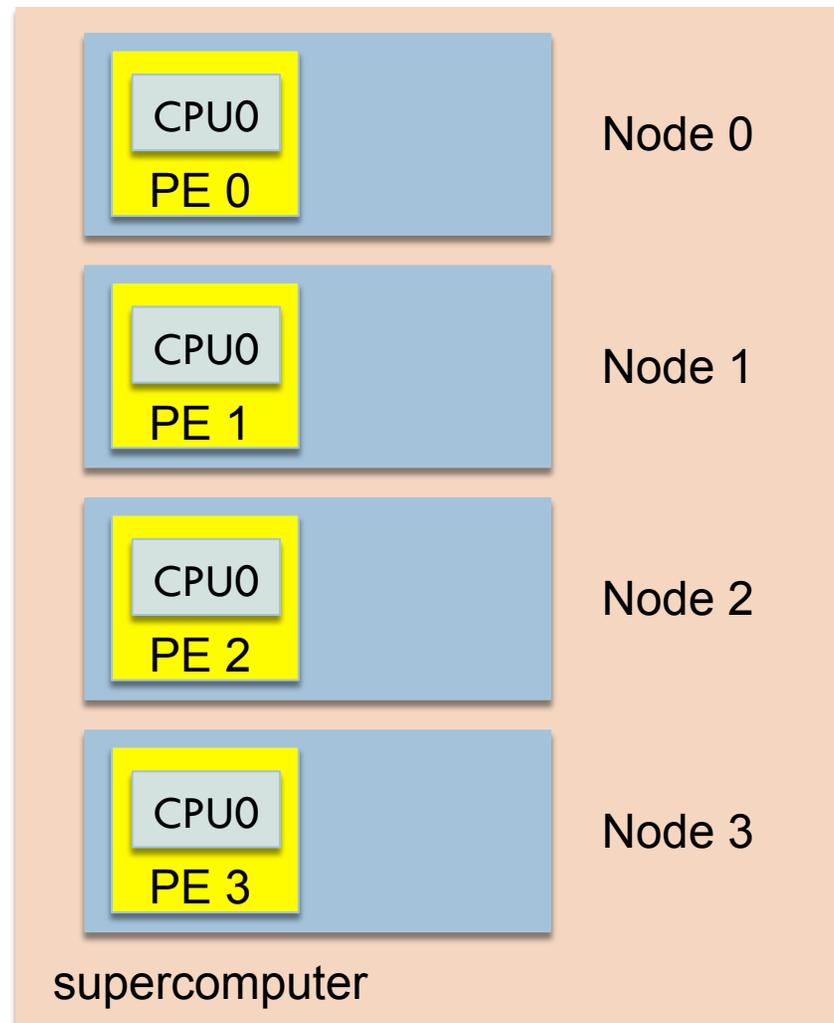


Massive programming investment? ... not so much (yet)

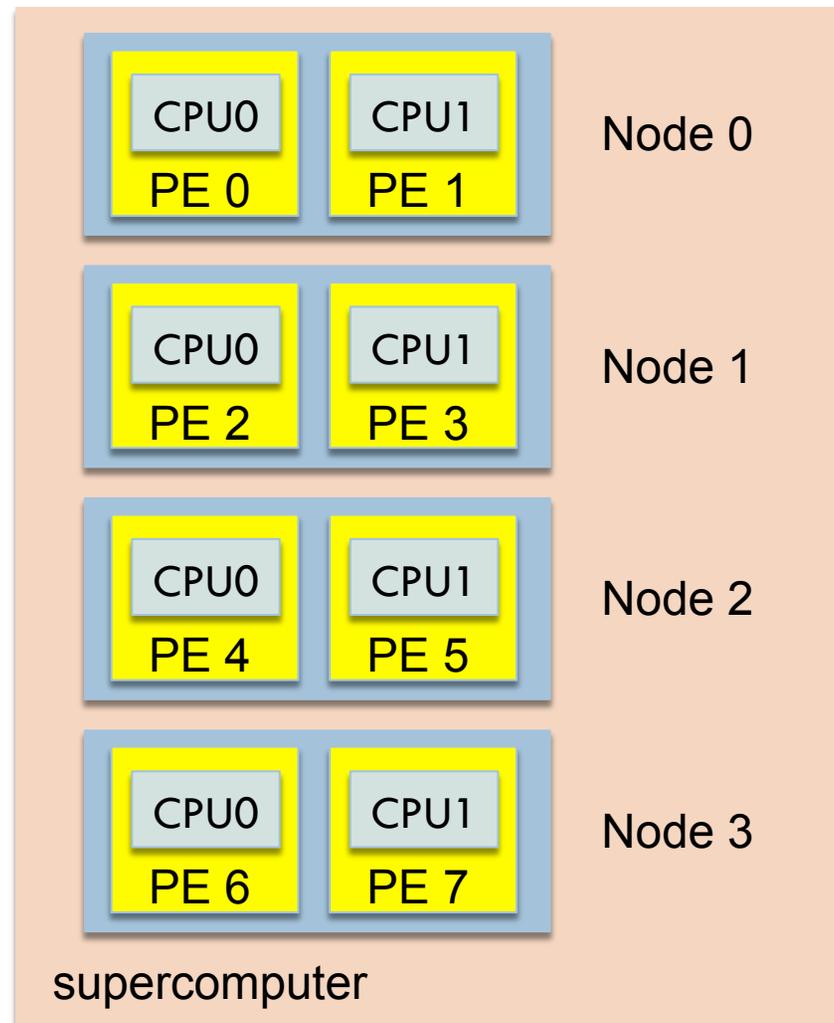
- Mid 2000s-present:
 - ▣ Rise of the multi-core CPU, GPU. AMD Opteron, Intel Nehalem, Sony Cell BE, NVIDIA G80, etc.
 - ▣ Large supercomputers comprised of lots of multi-core CPUs.



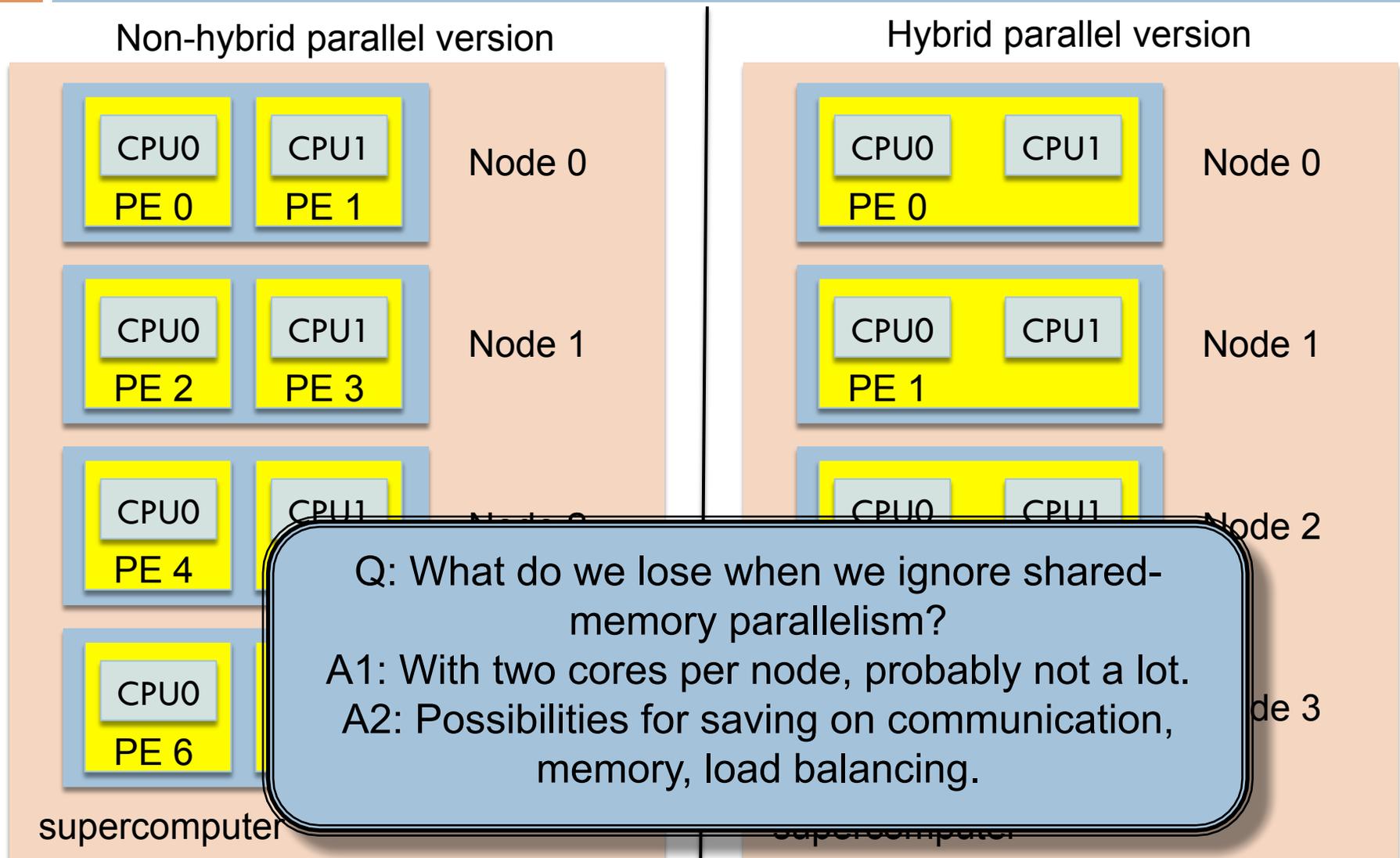
Distributed Memory Parallelism (early 2000s)



De Facto Standard for Distributed-Memory Parallelism



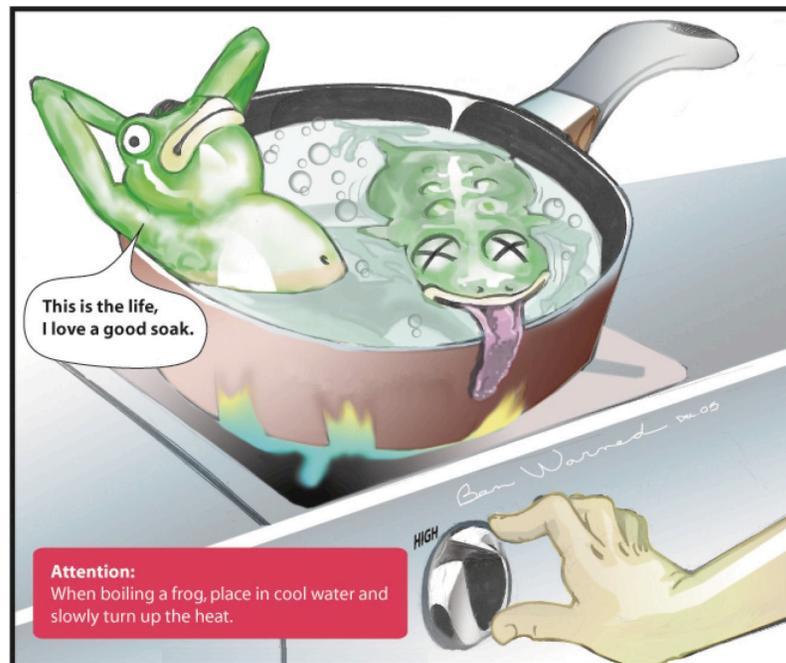
Contrasting Hybrid Parallel and Non-Hybrid Parallel Versions



On the (non-) transition to hybrid parallelism...

- In our defense,
 - Biggest gains were in scalability with many PE's
 - Putting two MPI tasks on a node wasn't so bad
 - Lack of will/enthusiasm for big code re-writes

□ But...



Research Overview For Hybrid Parallelism



- Fundamental questions:
 - ▣ How to map algorithm onto a complex memory, communication hierarchy?
 - ▣ What is the right balance of distributed- vs. shared-memory parallelism? How does balance impact performance?

Research Overview For Hybrid Parallelism



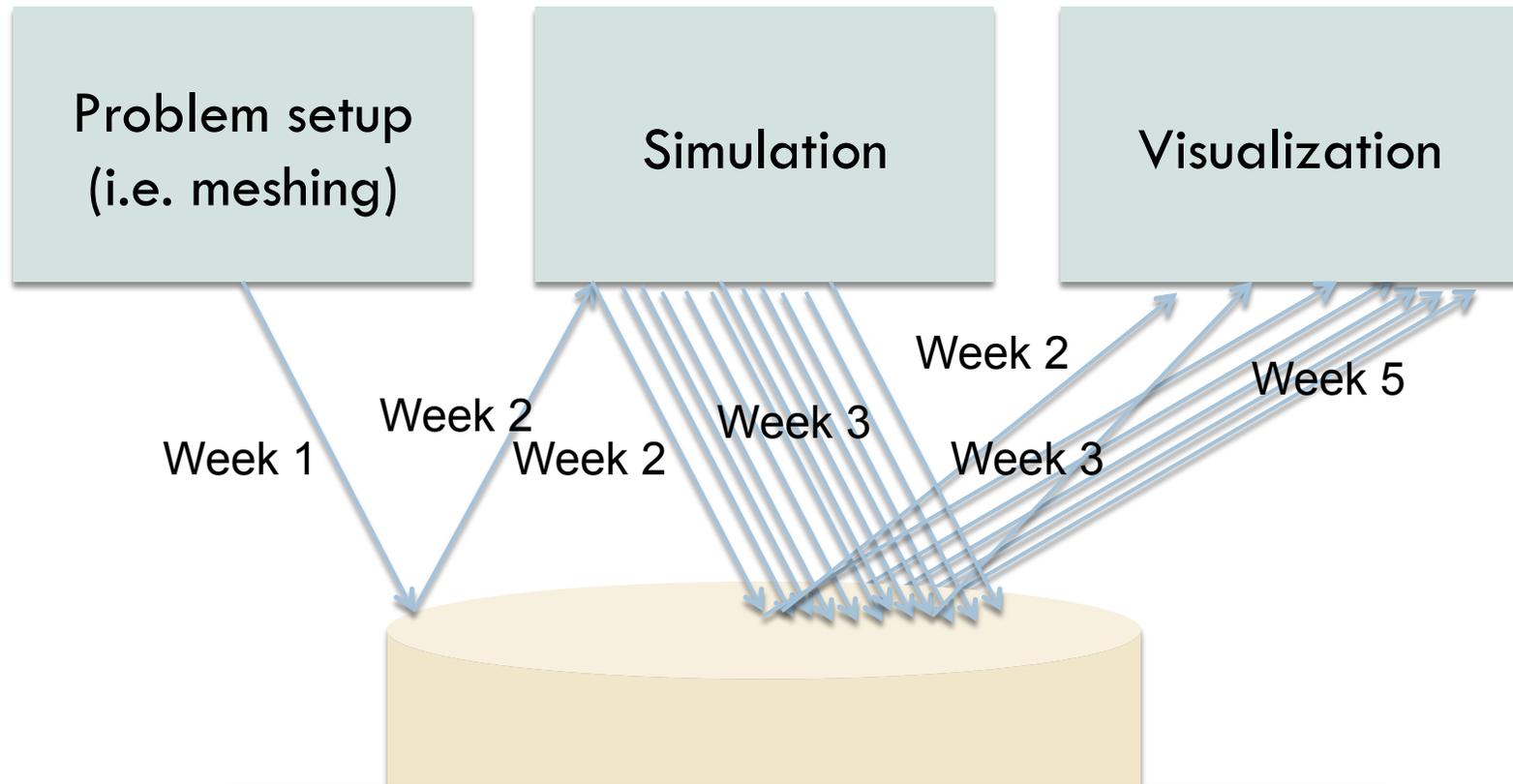
- Many factors influence performance/scalability:
 - Synchronization overhead.
 - Load balance (intra- and inter-chip).
 - Communication overhead and patterns.
 - Memory access patterns.
 - Fixed costs of initialization.
 - Number of runtime threads.

Outline



- Overview:
 - What is hybrid parallelism?
- Motivation:
 - Why is hybrid parallelism so important?
- Results:
 - What has been demonstrated with hybrid parallelism so far?
- Challenges:
 - What work still needs to happen?

The (Soon to Be) Good Old Days: Visualization as a Post-Processor



The shift away from this model has been happening for a while.

In Situ Processing

- Defined: couple visualization and analysis routines with the simulation code (no I/O)
- Pros:
 - No I/O!
 - Can access all the data
 - Computational power readily available
- Cons:
 - Must know what you want to look for a priori
 - Increasing complexity
 - Constraints (memory, network, execution time)

Some history behind this presentation...

- “Why Petascale Visualization Will Change the Rules” (2007-2010)

SciDAC 2007



NSF Workshop on Petascale I/O



- “Why Exascale Visualization Will Change the Rules” (2011-2013)



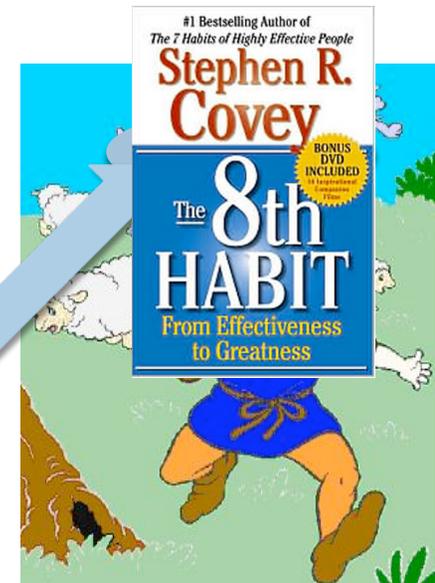
The context for this talk...



Petascale
Visualization



Exascale
Visualization

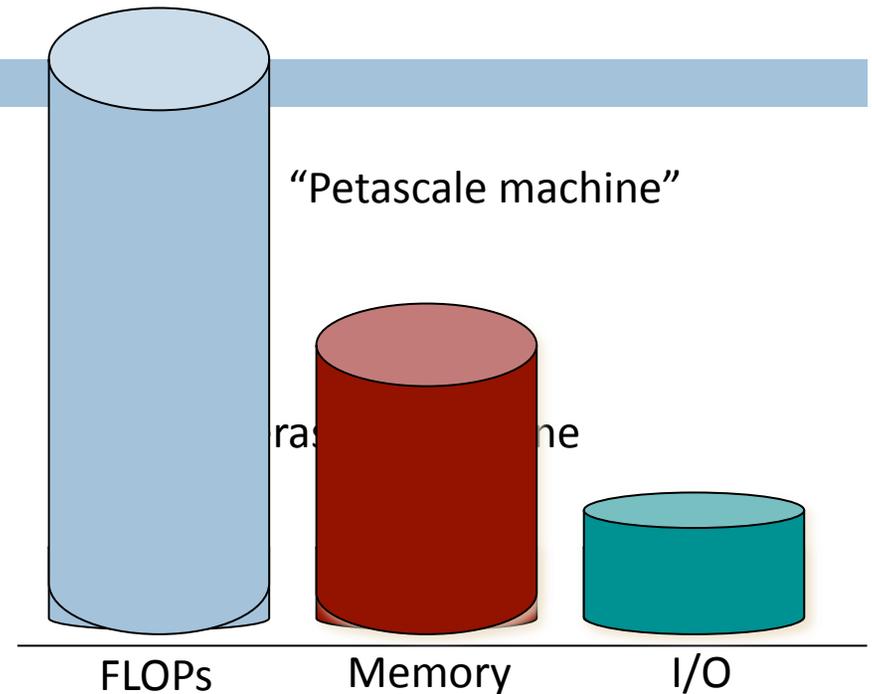


Hybrid
Parallelism &
Visualization

Will describe petascale and exascale wolves and pups in upcoming slides.

The I/O Wolf

- Large data visualization is almost always $>50\%$ I/O and sometimes 98% I/O
- Amount of data to visualize is typically $O(\text{total mem})$
- Two big factors:
 - ① how much data you have to read
 - ② how fast you can read it
- \rightarrow Relative I/O (ratio of total memory and I/O) is key



Exascale: a heterogeneous, distributed memory GigaHz KiloCore MegaNode system

Systems	2009	2018	Difference Today & 2018
System peak	2 Pflop/s	1 Eflop/s	O(1000)
Power	6 MW	~20 MW	~3
System memory	0.3 PB	32 - 64 PB [.03 Bytes/Flop]	O(100)
			O(10) - O(100)
			O(100)
			O(100) - O(1000)
			O(100)
			O(10) - O(100)
			O(10,000)
		latency hiding]	
Storage	15 PB	500-1000 PB (>10x system memory is min)	O(10) - O(100)
IO	0.2 TB	60 TB/s (how long to drain the machine)	O(100)
MTTI	days	O(1 day)	- O(10)



Data movement costs power and we have to get ~100X more power efficient.

The Four Angry Pups

I/O Bandwidth

Data
Movement



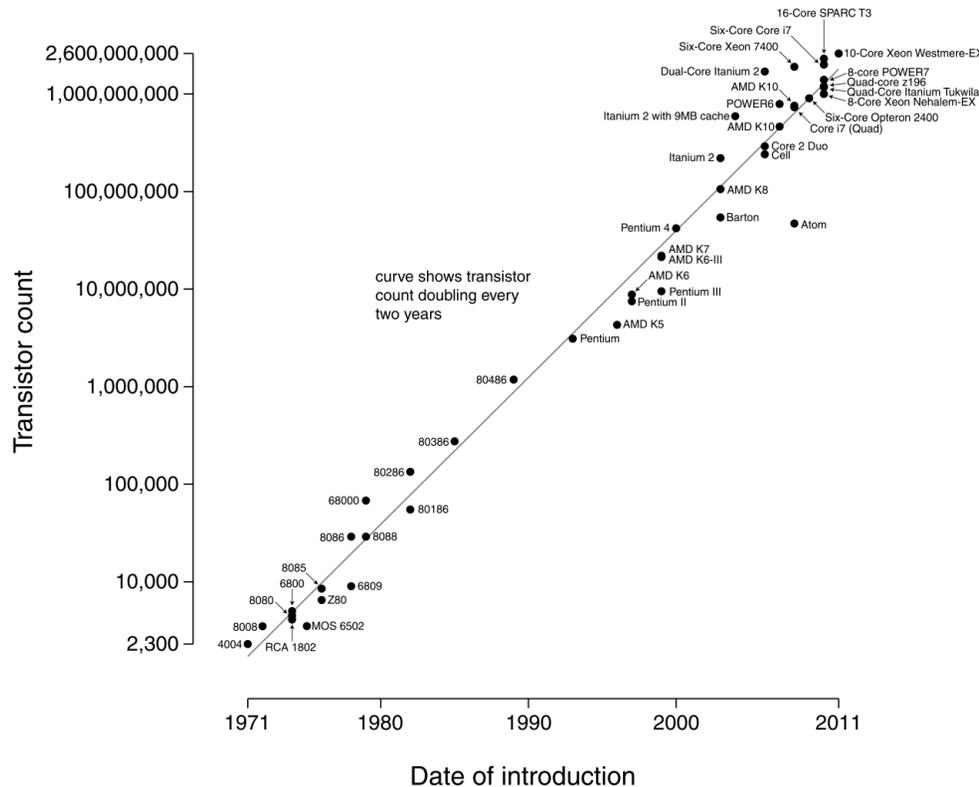
Exascale
Visualization

Data Movement's
4 Angry Pups

- In situ system design
- Memory footprint
- Programming languages
- Exploration with in situ
- (5th pup: hybrid parallelism)

Exascale Hardware: Moore's Law Is Alive and Well

Microprocessor Transistor Counts 1971-2011 & Moore's Law



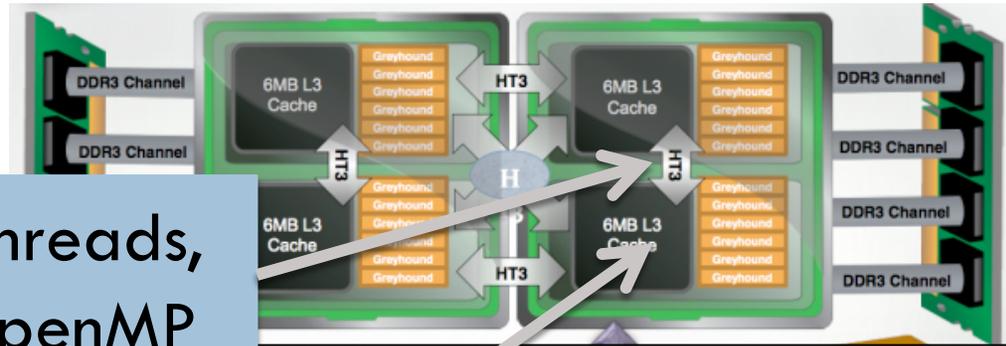
- Moore's Law was approximated to apply to clock speed...
 - ... that approximation no longer applies
- But transistor counts continue to climb.
- And this is equating to more and more cores per node...
 - ... how far will this go?

There are qualitative differences between programming multi-core and many-core nodes.

- Multi-core node: few dozens of cores per node



Intel Pentium Dual-Core, 2006



NERSC Hopper machine, 24 cores per node

Pthreads,
OpenMP

- Many-core node: 100s of cores per node

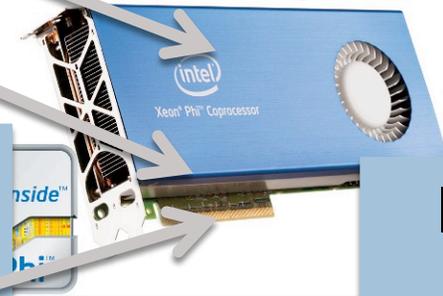
CUDA



NVIDIA K20 GPU Accelerator
2496 thread processors

OpenCL

Typically used as
accelerator (no
access to network)



Intel Xeon Phi, 57-61 cores per node,
16-way vector compute

Intel
TBB

Definitions

- Core: a processing thread on a CPU
- Node: a group of cores that share memory
- Processing Element (PE): one instance (of many) of a distributed memory parallel program
- Multi-core: dozens of cores per node
- Many-core: 100s to 1000s of cores per node

Petascale and Exascale Solutions



- Petascale: we must reduce I/O
 - Multi-resolution
 - In situ
 - Subsetting
 - Out-of-core
- Exascale: we must minimize power & data movement
 - In situ
 - Power efficiency within a core
 - Power efficiency across nodes

“The world as we know it” / “The world as we assume it”

- We will need to run in situ
 - We will need to minimize impact on the simulation code: memory, execution time, communication bandwidth, power consumption
- The HW already is multi-core, and increasingly many-core
 - We are going to have to adapt to the HW

Hybrid parallelism will be necessary to achieve these goals.

Outline

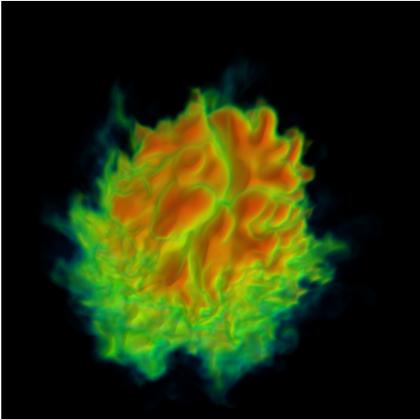
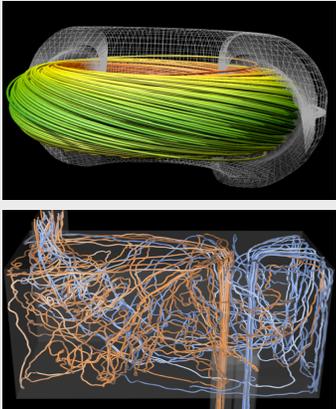
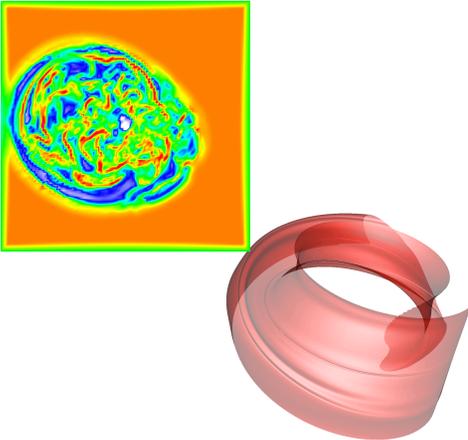


- Overview:
 - ▣ What is hybrid parallelism?
- Motivation:
 - ▣ Why is hybrid parallelism so important?
- Results:
 - ▣ What has been demonstrated with hybrid parallelism so far?
- Challenges:
 - ▣ What work still needs to happen?

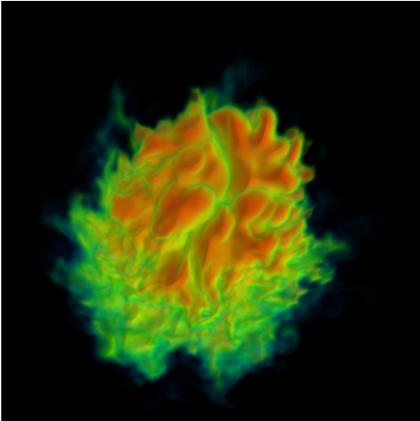
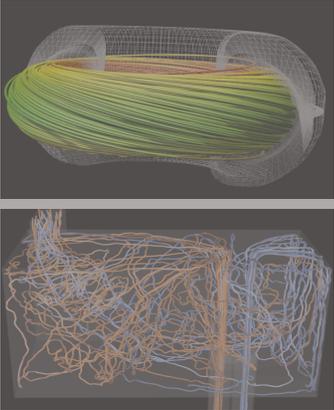
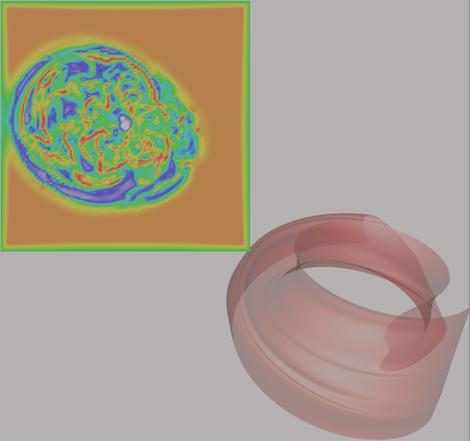
Previous work on hybrid parallelism

- Simulations: GPGPU, others
- Visualization: lots of single GPU work
- Hybrid parallelism + visualization: only a handful of studies.
 - ▣ This oversight is significant: parallel visualization and analysis algorithms have markedly different characteristics – computational load, memory access pattern, communication, idle time, etc. – than the other two categories.

Studies on hybrid parallelism and visualization to date

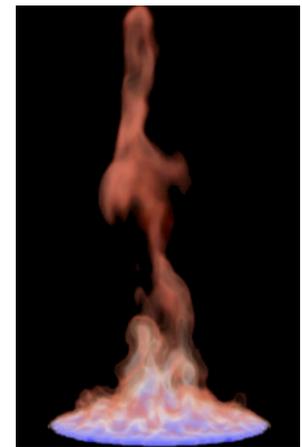
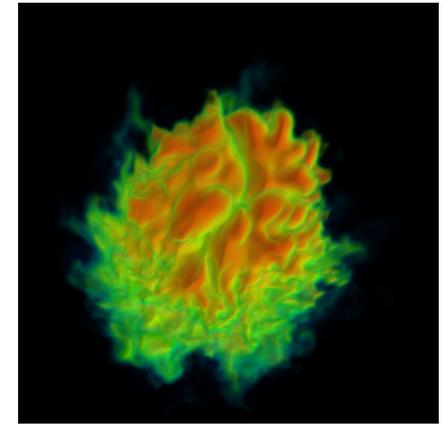
Architecture	Volume Rendering	Particle Advection
Multi-core		
Many-core		

Studies on hybrid parallelism and visualization to date

Architecture	Volume Rendering	Particle Advection
Multi-core		
Many-core		

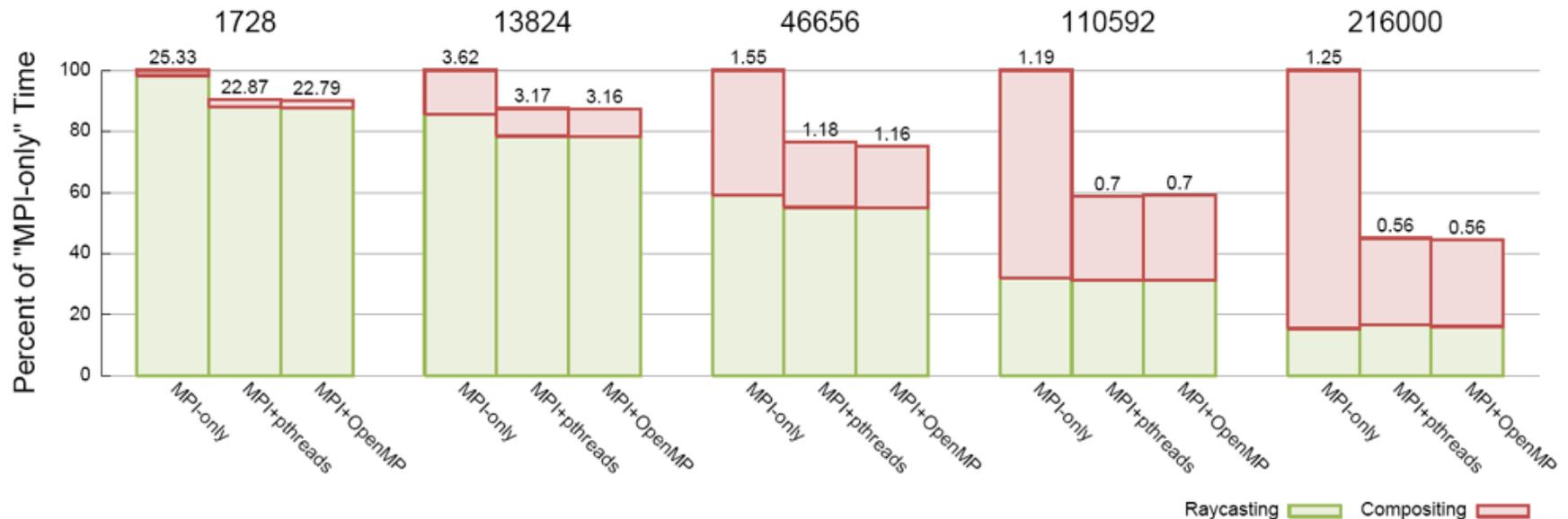
Volume Rendering Studies

- Volume rendering: use a combination of color and transparency to see an entire three-dimensional volume at one time.
- Consists of 2 phases: sample & composite
 - ▣ Sampling: embarrassingly parallel
 - ▣ Compositing: parallel communication
- Goal: sampling continues to work well, compositing gets faster



Multi-core Volume Rendering

- Study: use same HW, with hybrid parallelism and without

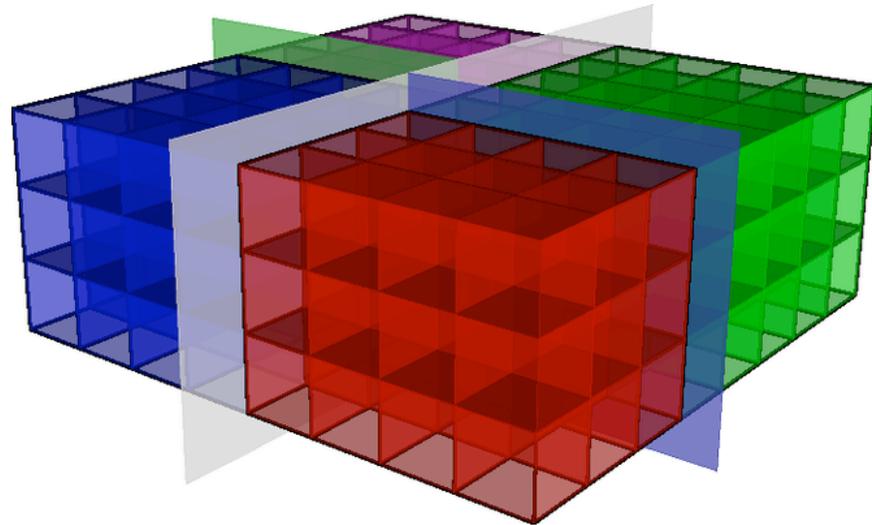
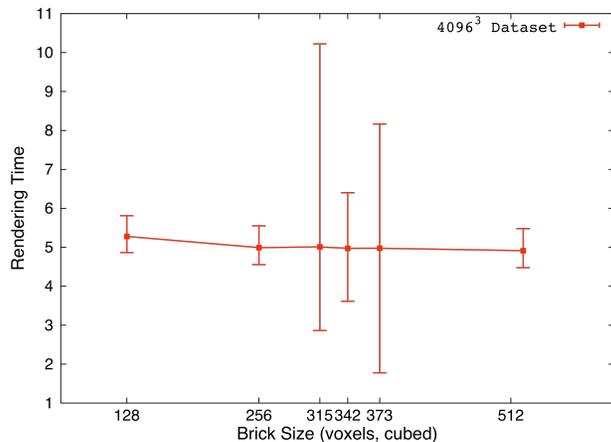


- References:

- Howison et al. "MPI-hybrid Parallelism for Volume Rendering on Large, Multi-core Systems" In EGPGV'10
- Howison et al. "Hybrid Parallelism for Volume Rendering on Large-, Multi-, and Many-Core Systems" in TVCG, Jan. 2012

Many-core Volume Rendering

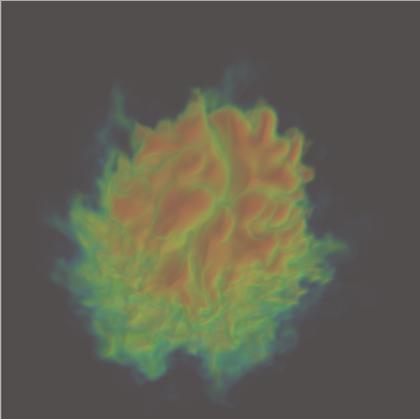
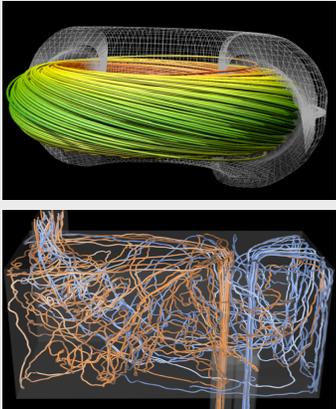
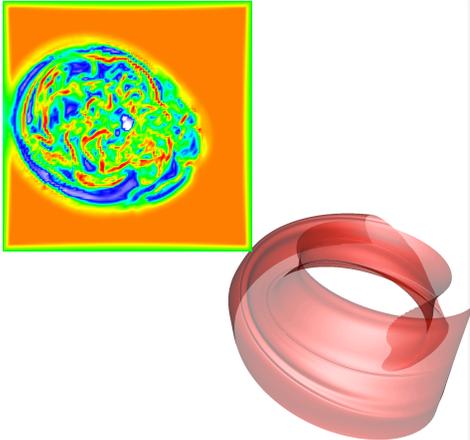
- Study concerned with mapping the algorithm and optimizing performance, not with comparisons
 - Result: can use GPU cluster to do volume rendering of very large data sets.



References:

- T. Fogal et al. Large Data Visualization on Distributed Memory Multi-GPU Clusters. In HPG 2010.

Studies on hybrid parallelism and visualization to date

Architecture	Volume Rendering	Particle Advection
Multi-core		
Many-core		

What is advection?

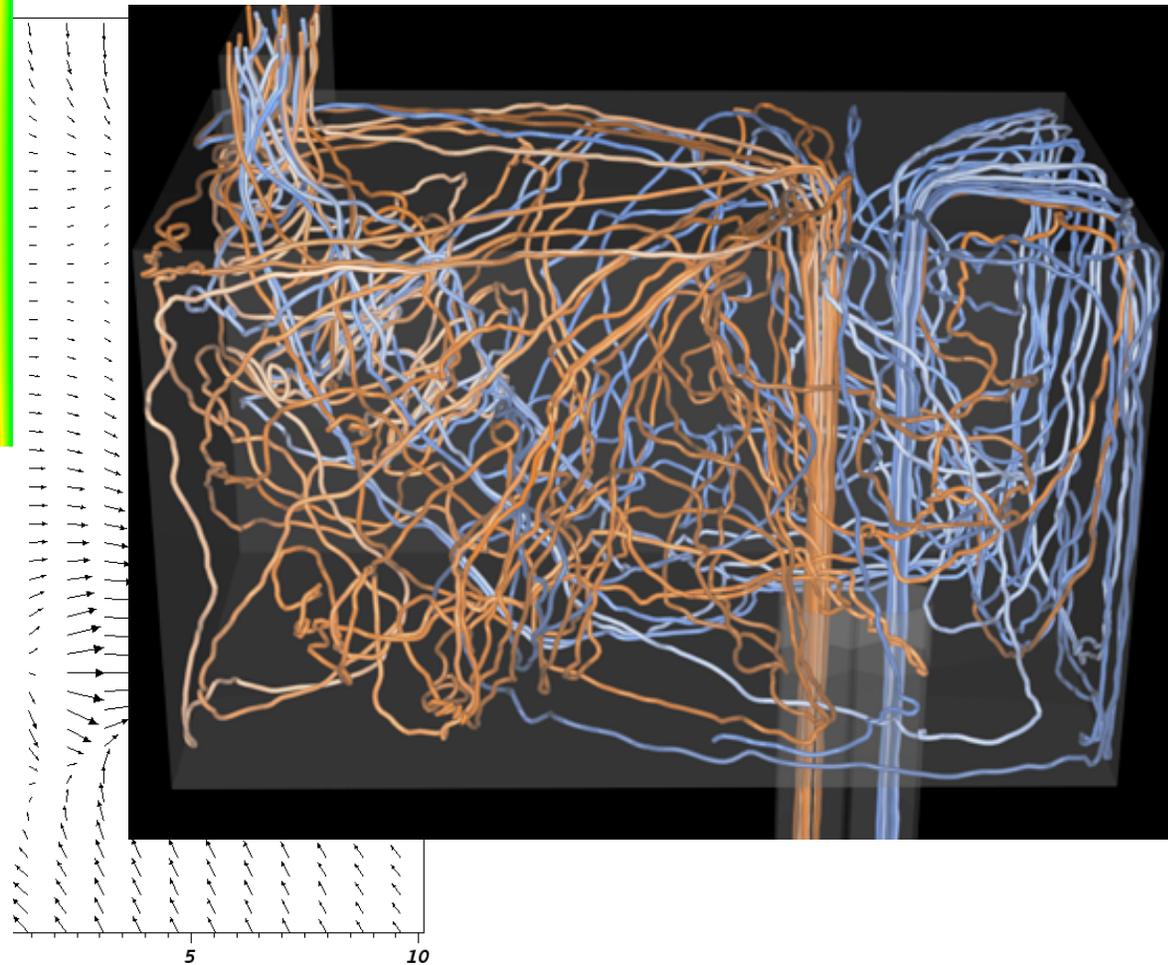
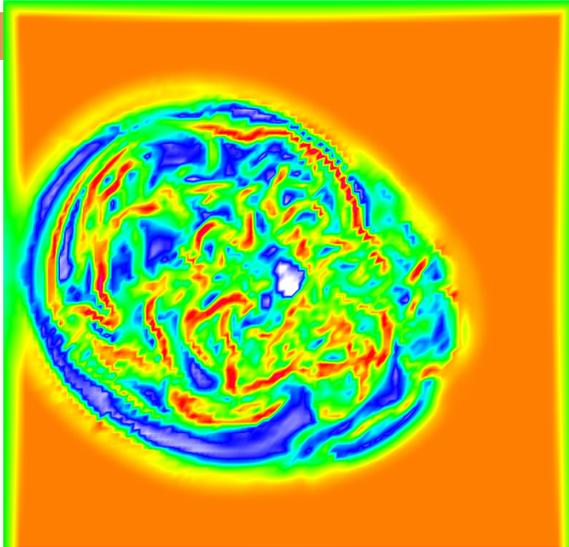
ad·vec·tion ¹ (ăd-vĕk'shən)

n.

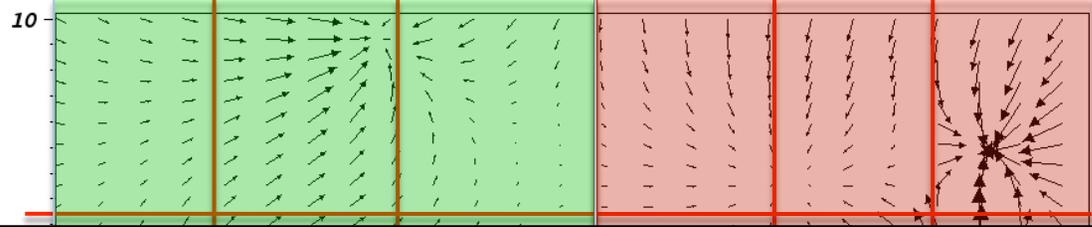
1. The transfer of a property of the atmosphere, such as heat, cold, or humidity, by the horizontal movement of an air mass: *Today's temperatures were higher due to the advection of warm air into the region.*
2. The rate of change of an atmospheric property caused by the horizontal movement of air.
3. The horizontal movement of water, as in an ocean current.

[Latin *advectiō*, *advectiōn-*, *act of conveying*, from *advectus*, past participle of *advehere*, *to carry to* : *ad-*, *ad-* + *vehere*, *to carry*; see *wegh-* in Indo-European roots.]

Particle advection is a foundational visualization algorithm

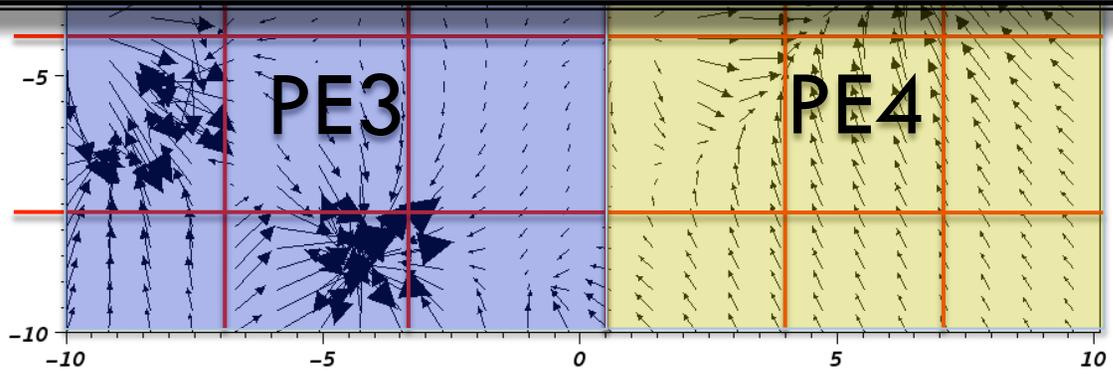


“Parallelize over data” strategy: parallelize over pieces and pass particles



Other parallelization schemes, but this one is common.
Very hard to get good efficiency, especially due to data dependency.

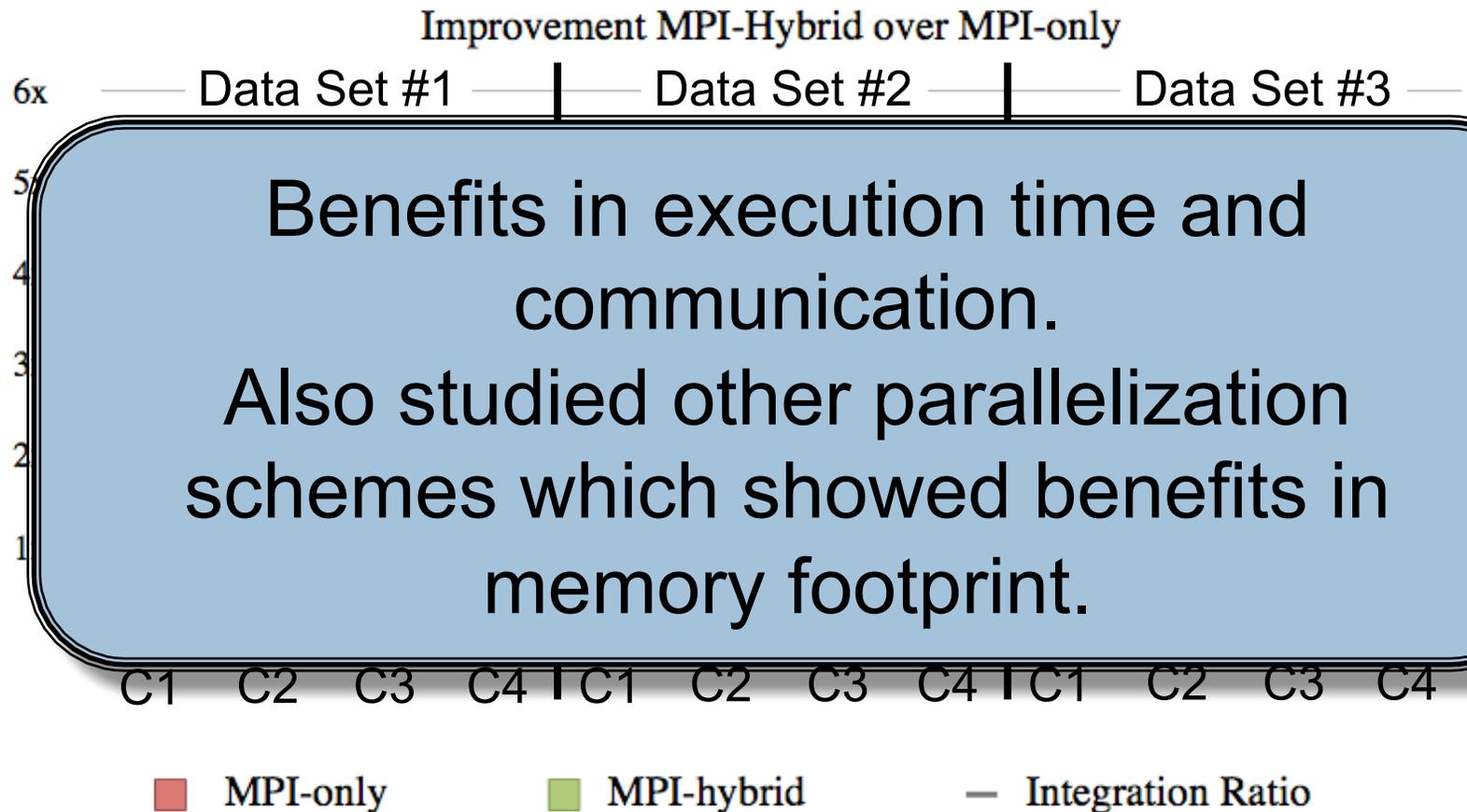
PE = Processing Element, i.e. an instance of the program.



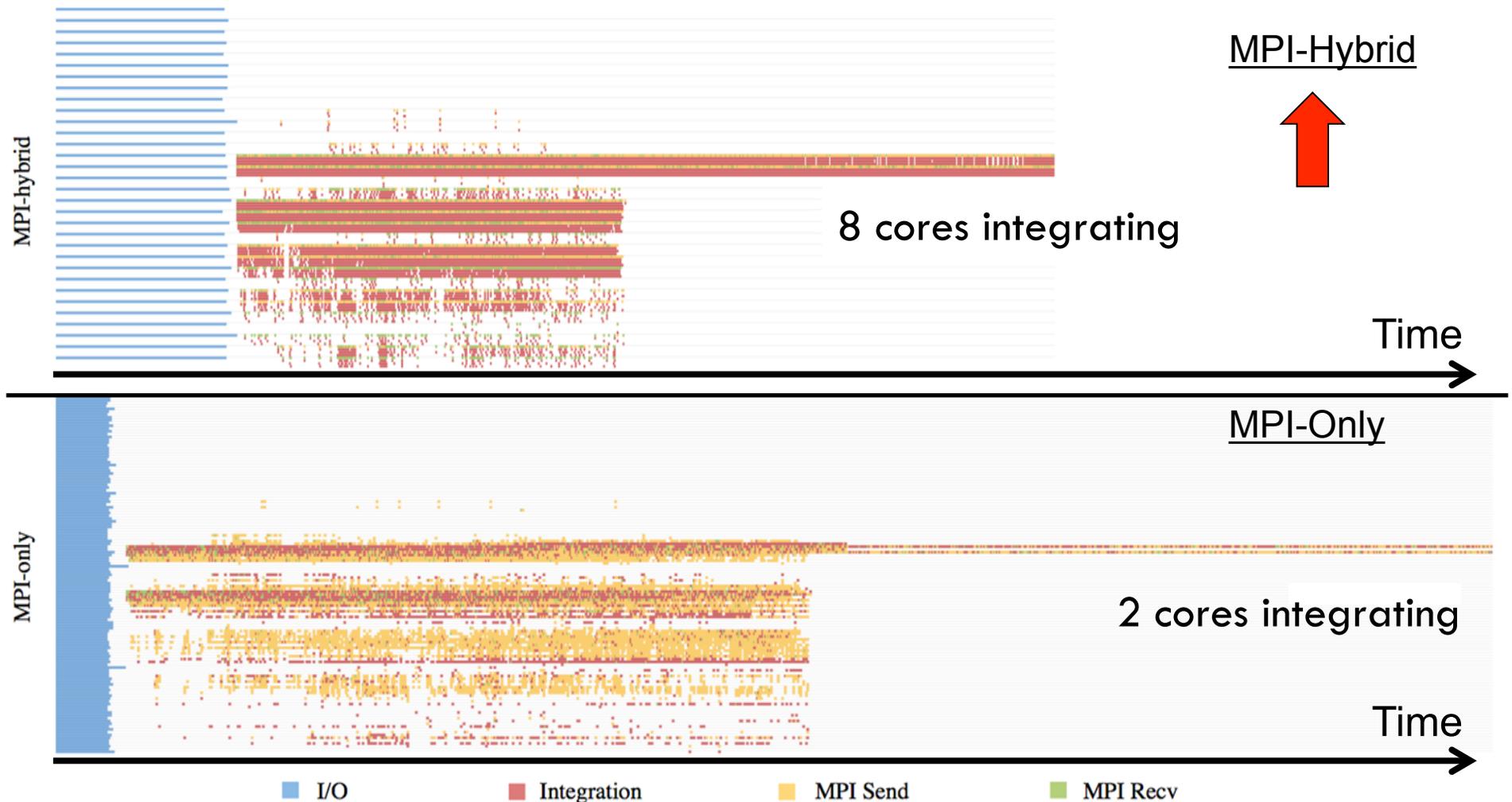
Studies for Multi-core and Many-core Particle Advection

- “No two particle advection problems are alike.”
 - Studies vary over:
 - Number of seed points.
 - Duration of advection.
 - Vector field complexity (via multiple data sets)
- Both studies involve comparisons:
 - Multi-core: how does hybrid parallelism improve performance?
 - Many-core: if we have a GPU cluster, should we even use the GPUs? (latencies, slower processor cores)

Multi-core: comparing speedups for hybrid parallelism

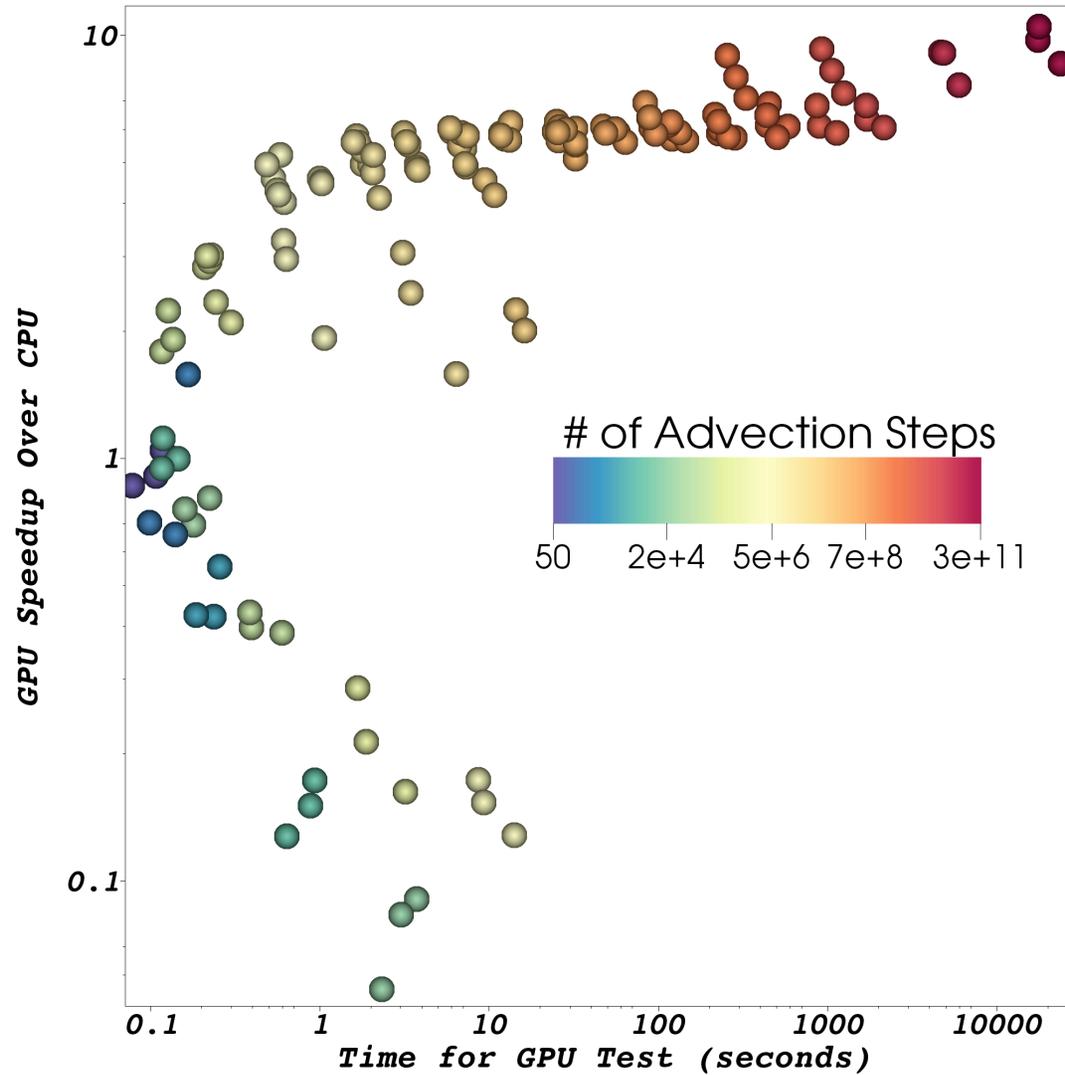


Multi-core comparisons: Gantt Chart



- References: D. Camp et al. "Streamline Integration Using MPI-Hybrid Parallelism on a Large Multicore Architecture." TVCG Nov. 2011

Many-core: studying 150 pairs of tests...

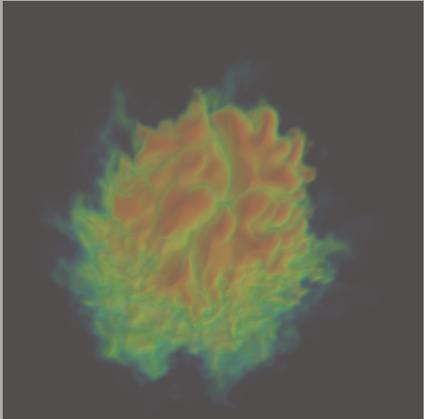
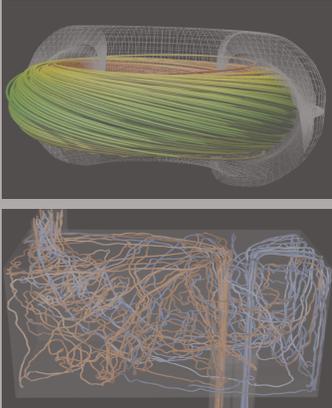
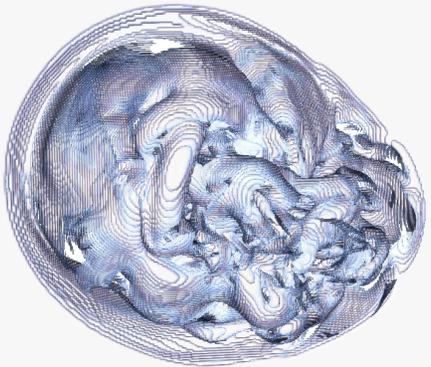
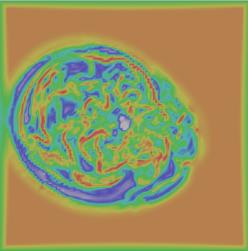
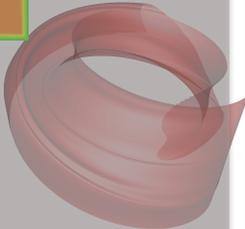


Many-core: identifying the most dominant factors for performance

	Event	Slope	Correlation Coefficient
GPU Advantage	High activity	11.6	0.81
	Low activity	-5.3	-0.63
	CPU overhead	-0.05	0.01
	Latency	-0.25	-0.44
	Idle	-6.0	-0.50
CPU Advantage	High activity	0.0	0.0
	Low activity	3.6	0.5
	CPU overhead	-0.03	-0.333
	Latency	-0.13	-0.69
	Idle	-3.4	-0.50

- References: D. Camp et al. "GPU Acceleration of Particle Advection Workloads in a Parallel, Distributed Memory Setting." EGPGV 2013

Studies on hybrid parallelism and visualization to date

Architecture	Volume Rendering	Particle Advection	Production Usage
Multi-core			
Many-core		 	 

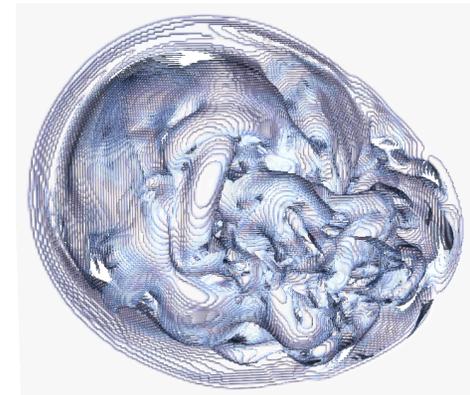
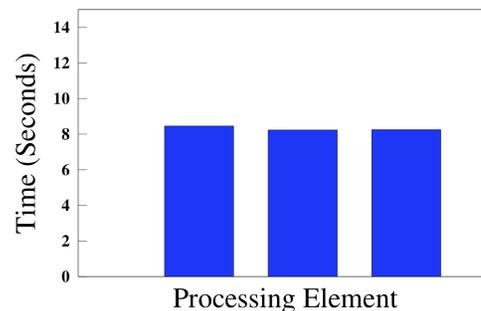
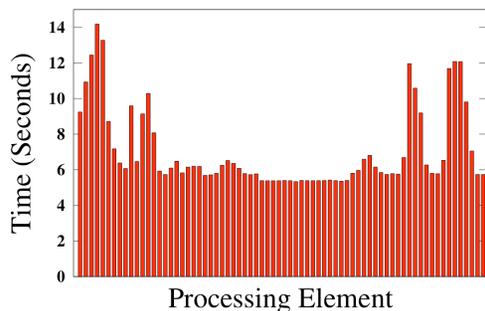
Multi-core production usage



- Many visualization algorithms are embarrassingly parallel.
- Further, nature of implementation for visualization software lends itself to hybrid parallel model.
 - ▣ One PE per node.
 - ▣ PE divides work among cores.
 - ▣ Cores can execute on their piece of data without needing to coordinate with other cores/PEs.

Multi-core production usage

- Savings in memory: 22.3 GB \rightarrow 20.7GB (=1.6GB)
 - ▣ Strictly because of fewer instances of the binary
- Improvements in performance: 14s \rightarrow 8.4s (1.67X)
 - ▣ Due to improved load balancing



References:

- ▣ Camp et al. “Transitioning Data Flow-Based Visualization Software to Multi-Core Hybrid Parallelism.” Workshop on Data-Flow Execution Models for Extreme Scale Computing (DFM 2013)

Many-core production usage

- See future work... ☹️

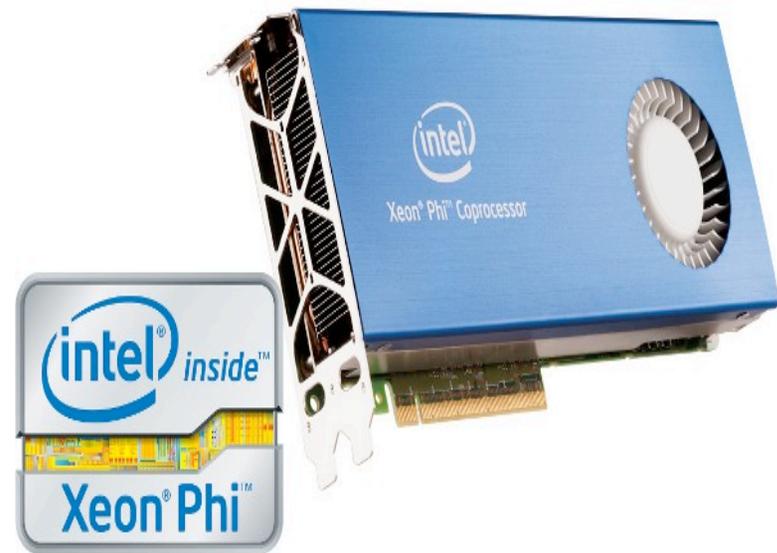
Outline



- Overview:
 - ▣ What is hybrid parallelism?
- Motivation:
 - ▣ Why is hybrid parallelism so important?
- Results:
 - ▣ What has been demonstrated with hybrid parallelism so far?
- Challenges:
 - ▣ What work still needs to happen?

Many-core is scary.

- Two reasons legacy code may not map well to many-core space:
 - Language mismatch
 - Need to re-think algorithms for 100s to 1000s of cores per node.



Many-core is scary.

- New libraries in development for many-core visualization
 - DAX
 - EAVL
 - PISTON
 - (panel at SC12 & upcoming panel @ Vis13)
- These libraries could be integrated with tools that have legacy approaches
- Still need to explore hybrid parallelism!!

More future work



- New dimension of evaluation: power
 - What is the most power-efficient way to do hybrid parallel particle advection? ... we don't know

Summary, Part 1 (of 4)

- Hybrid parallelism research questions:
 - How to map the algorithm to this complex architecture?
 - How to best take advantage of the architecture?
- Hybrid parallelism has been demonstrated to improve:
 - Memory usage
 - Execution time
 - Communication bandwidth

Summary, Part 2 (of 4)

- The volume rendering and particle advection studies help understand the impacts of hybrid parallelism, but more work is needed for these algorithms.
 - ▣ And lots of work for other algorithms.

Adding it all up

- Today:
 - ▣ Simulation scientist: “can you help me visualize X?”
- Soon:
 - ▣ Simulation scientist: “can you help me visualize X? ... and I have the following constraints --- C1, C2, C3”
- Example constraints:
 - ▣ Execution time
 - ▣ Memory usage
 - ▣ Power consumption
 - ▣ Communication bandwidth

The primary message of this talk



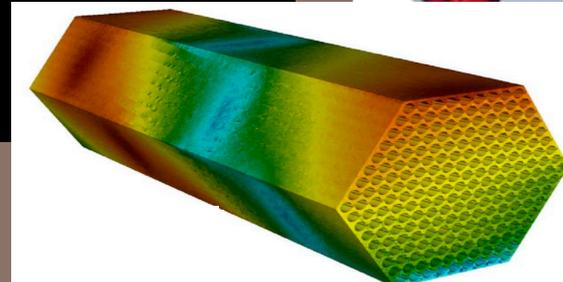
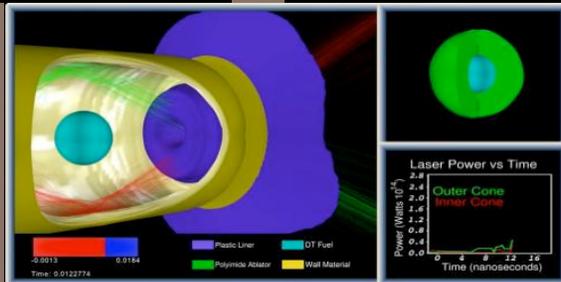
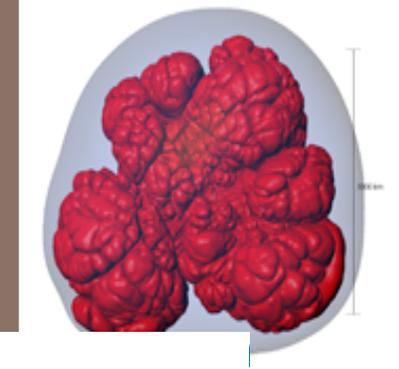
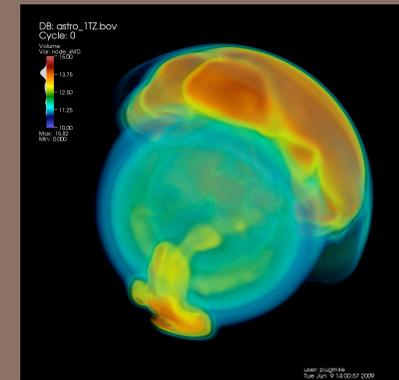
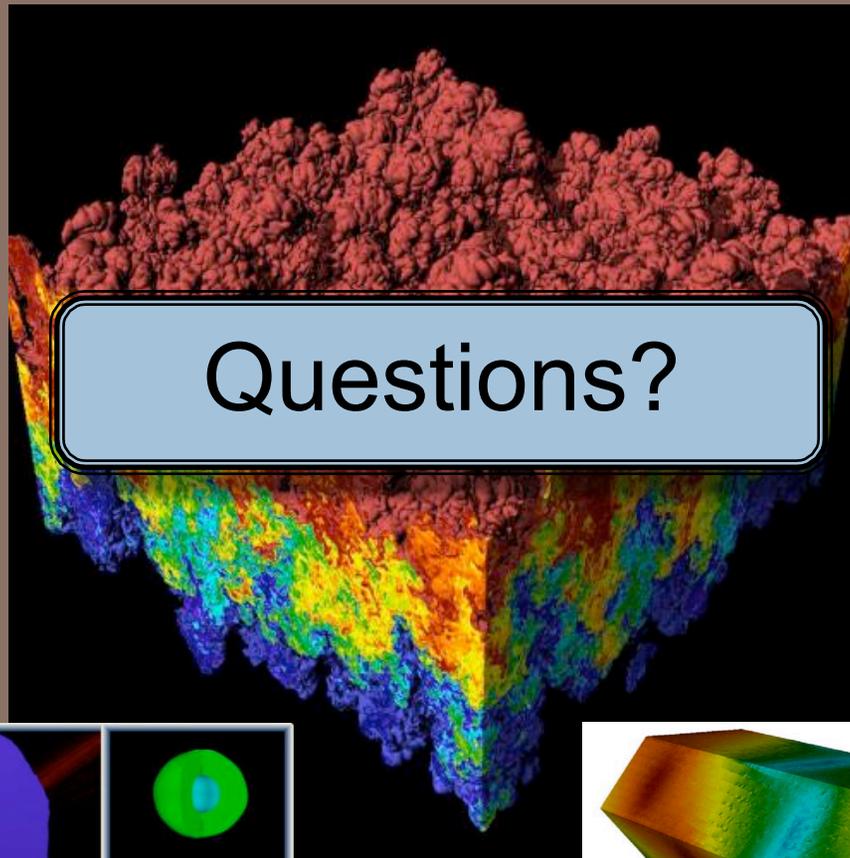
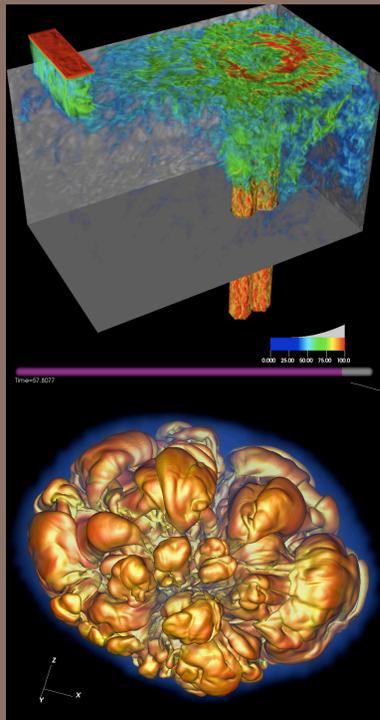
- We will soon live in a world where simulation scientists will ask us to solve problems with constraints.
- Hybrid parallelism will help us meet those constraints, by lowering our requirements.
- Our community has a lot of work to do ... let's get to it!!!

Acknowledgments



- Thank you to:
 - Funding agencies:
 - U.S. Dept. of Energy CAREER award
 - SciDAC Institute on Scientific Data Management, Analysis and Visualization (SDAV)
 - Dr. Garth & TU Kaiserslautern for arranging this presentation.
 - You, the audience!

Hybrid Parallelism and Visualization



Sep 16th, 2013

Hank Childs, University of Oregon & Lawrence Berkeley