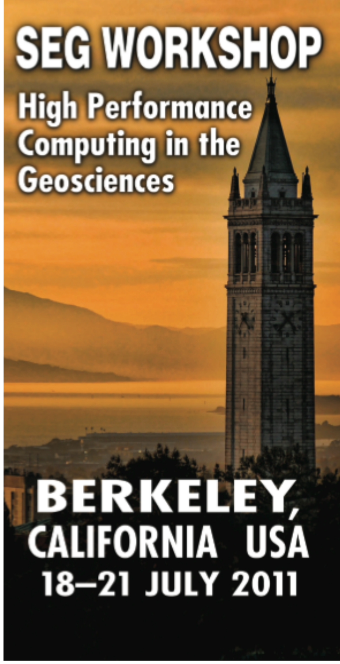
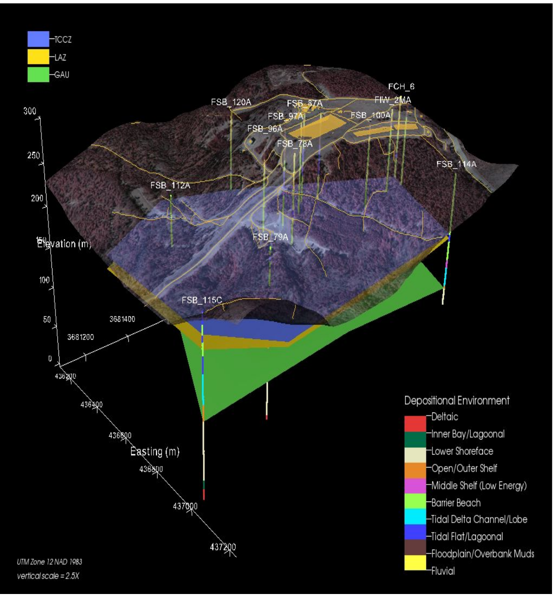
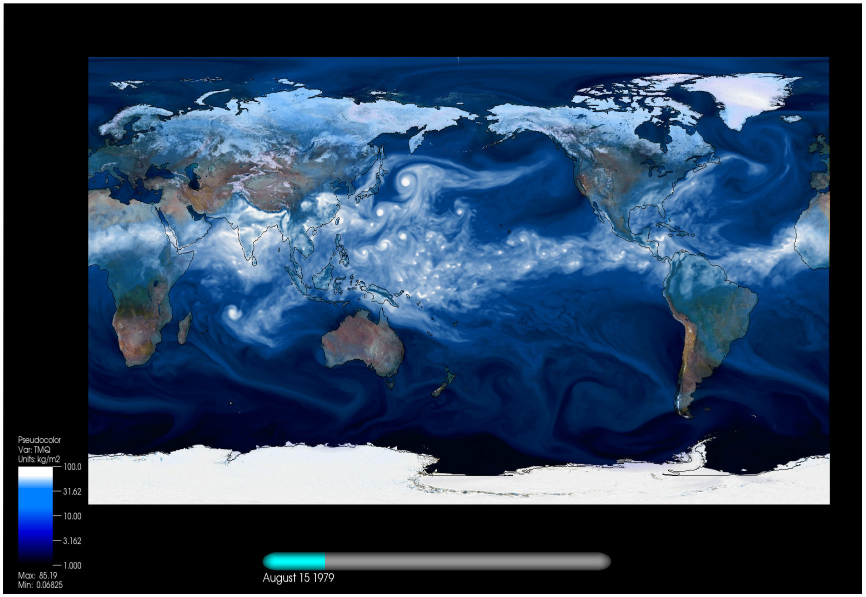
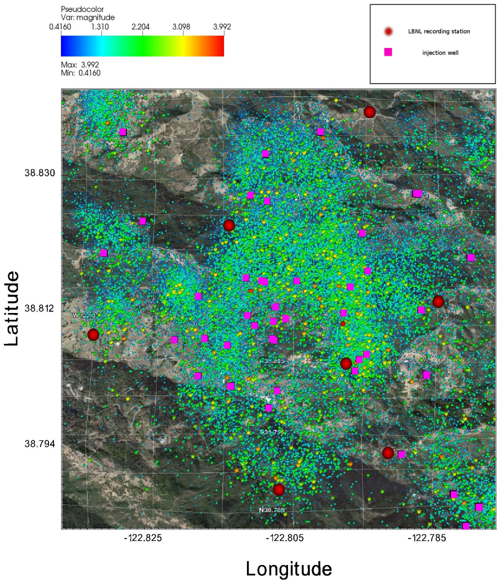
Challenges and Solutions for Visual Data Analysis on Current and Emerging HPC Platforms









July 20, 2011

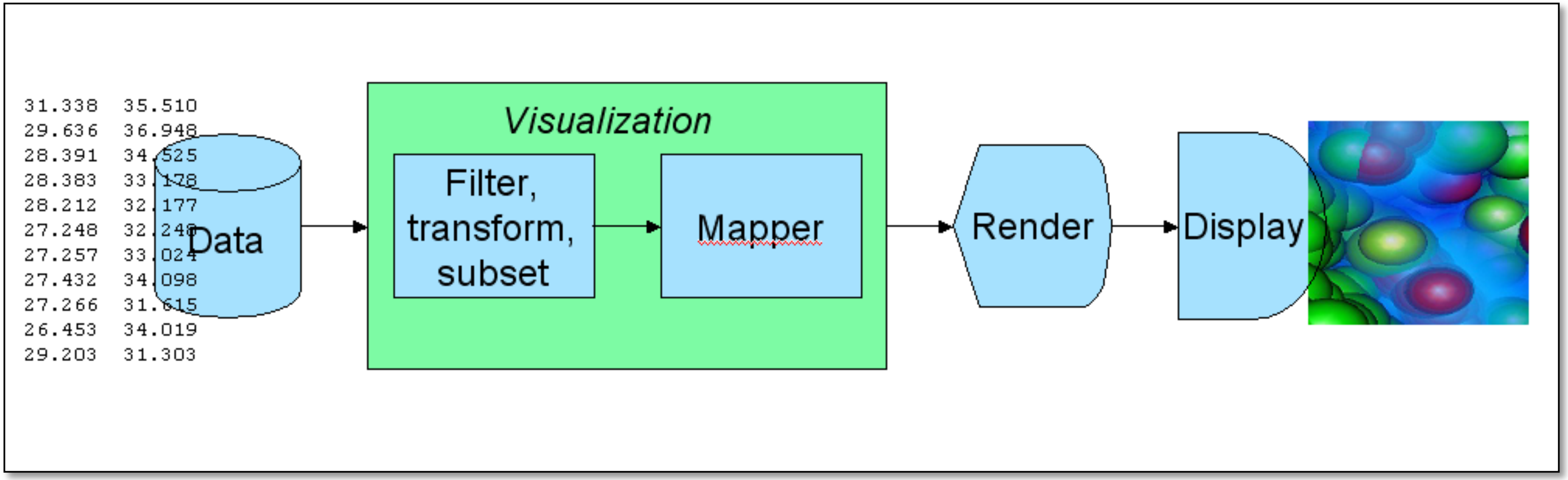
Wes Bethel & Hank Childs, Lawrence Berkeley Lab

Outline

- Petascale & Exascale issues (Childs)
- Tools: how the community comes together to collectively solve large data visual data analysis problems (Childs)
- Achieving extreme performance in visual data analysis (Bethel)
- Recent examples/case studies (Bethel)

Visualization 101

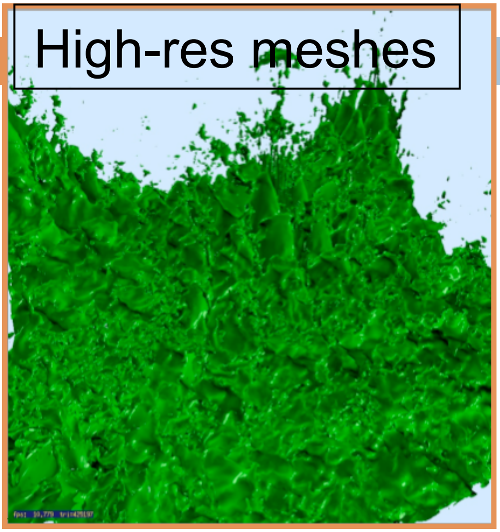
- Transformation of numbers (data) into readily comprehensible images.
- Plays an integral part in the scientific and analytic processes.
- Data intensive.



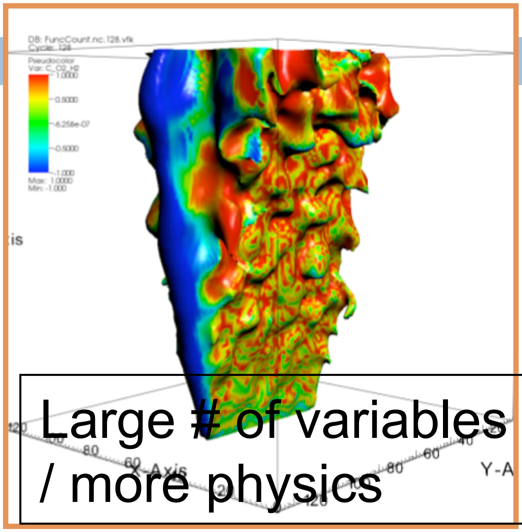
Why are supercomputing trends going to change the rules for visualization and analysis?

- Michael Strayer (U.S. DoE Office of Science) in 2006: *"petascale is not business as usual"*
  - Especially true for visualization and analysis!
- Large scale data creates two incredible challenges: scale and complexity
- Scale is not "business as usual"
  - Will discuss this assertion throughout this talk
  - Solution: we will need "smart" techniques in production environments
- More resolution leads to more and more complexity
  - Will the "business as usual" techniques still suffice?

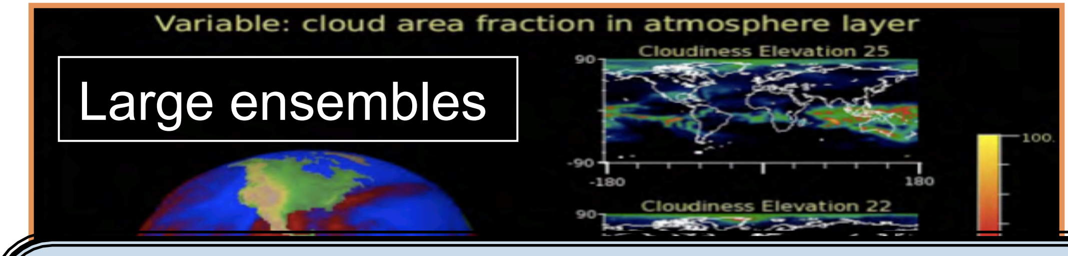
How does increased computing power affect the data to be visualized?



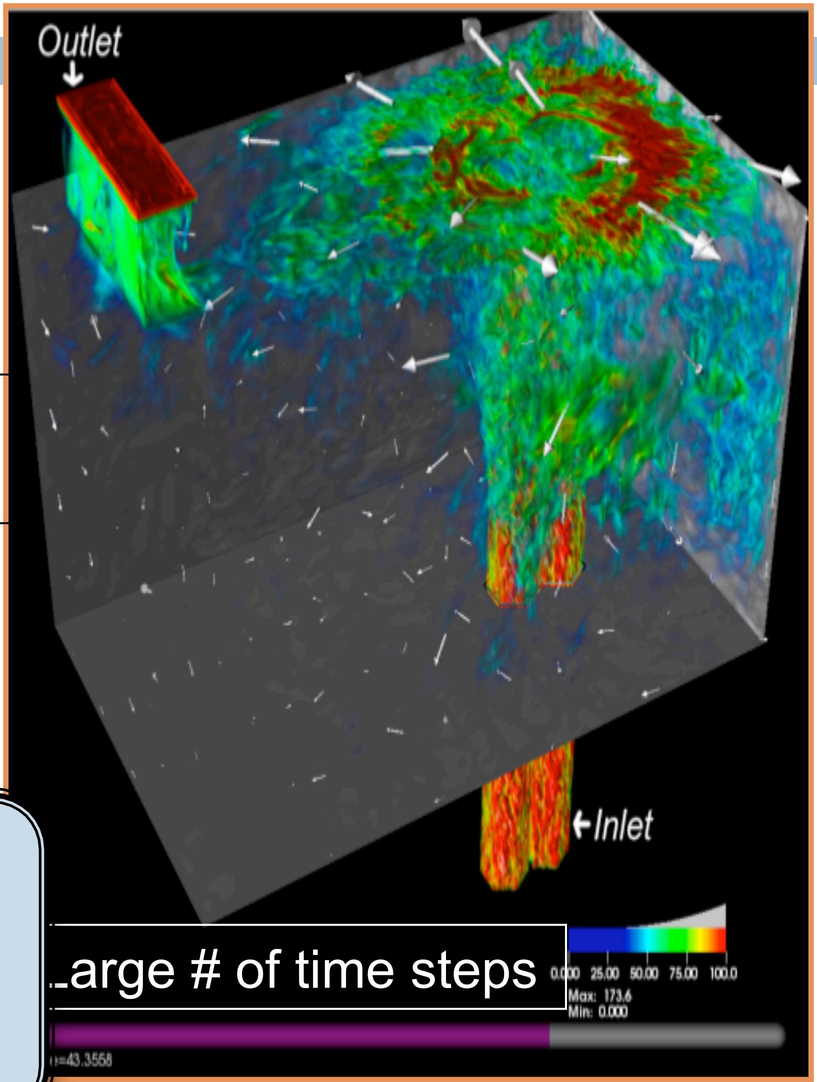
High-res meshes



Large # of variables / more physics



Large ensembles



Large # of time steps

Your mileage may vary; some simulations produce a lot of data and some don't.

Slide credit: Sean Ahern (ORNL) & Ken Joy (UCD)

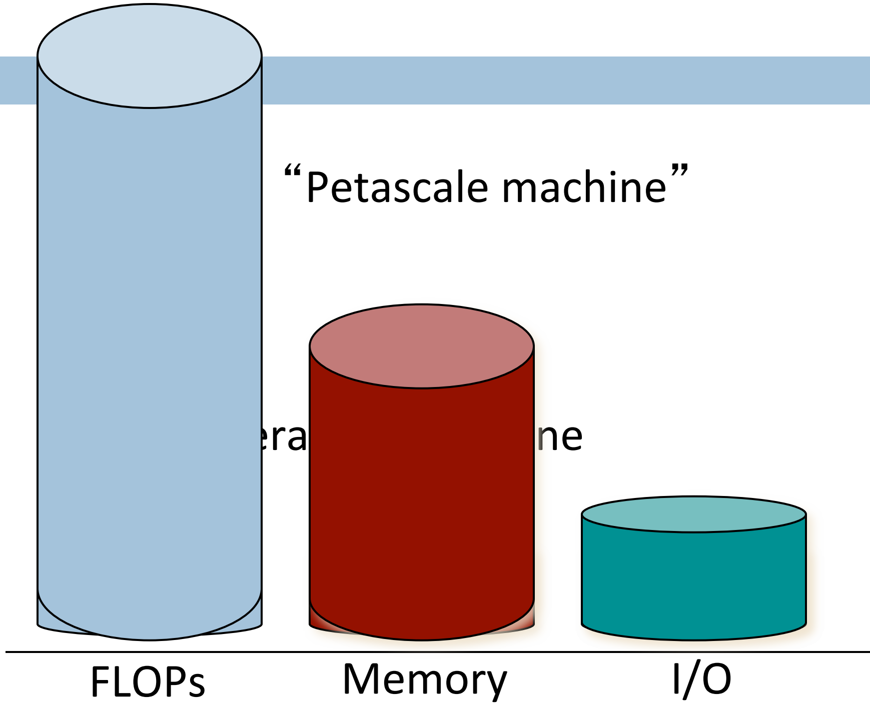
Today's production visualization tools use "pure parallelism" to process data.

Pure parallelism

- Pure parallelism: “brute force” … processing full resolution data using data-level parallelism
- Pros:
  - Easy to implement
- Cons:
  - Requires large I/O capabilities
  - Requires large amount of primary memory

I/O and visualization

- Pure parallelism is almost always >50% I/O and sometimes 98% I/O
- Amount of data to visualize is typically O(total mem)



- Two big factors:
  ① how much data you have to read
  ② how fast you can read it
- → Relative I/O (ratio of total memory and I/O) is key

Why is relative I/O getting slower?

- I/O is quickly becoming a dominant cost in the overall supercomputer procurement.
  - And I/O doesn't pay the bills.
- Simulation codes aren't as exposed.

We need to de-emphasize I/O in our visualization and analysis techniques.

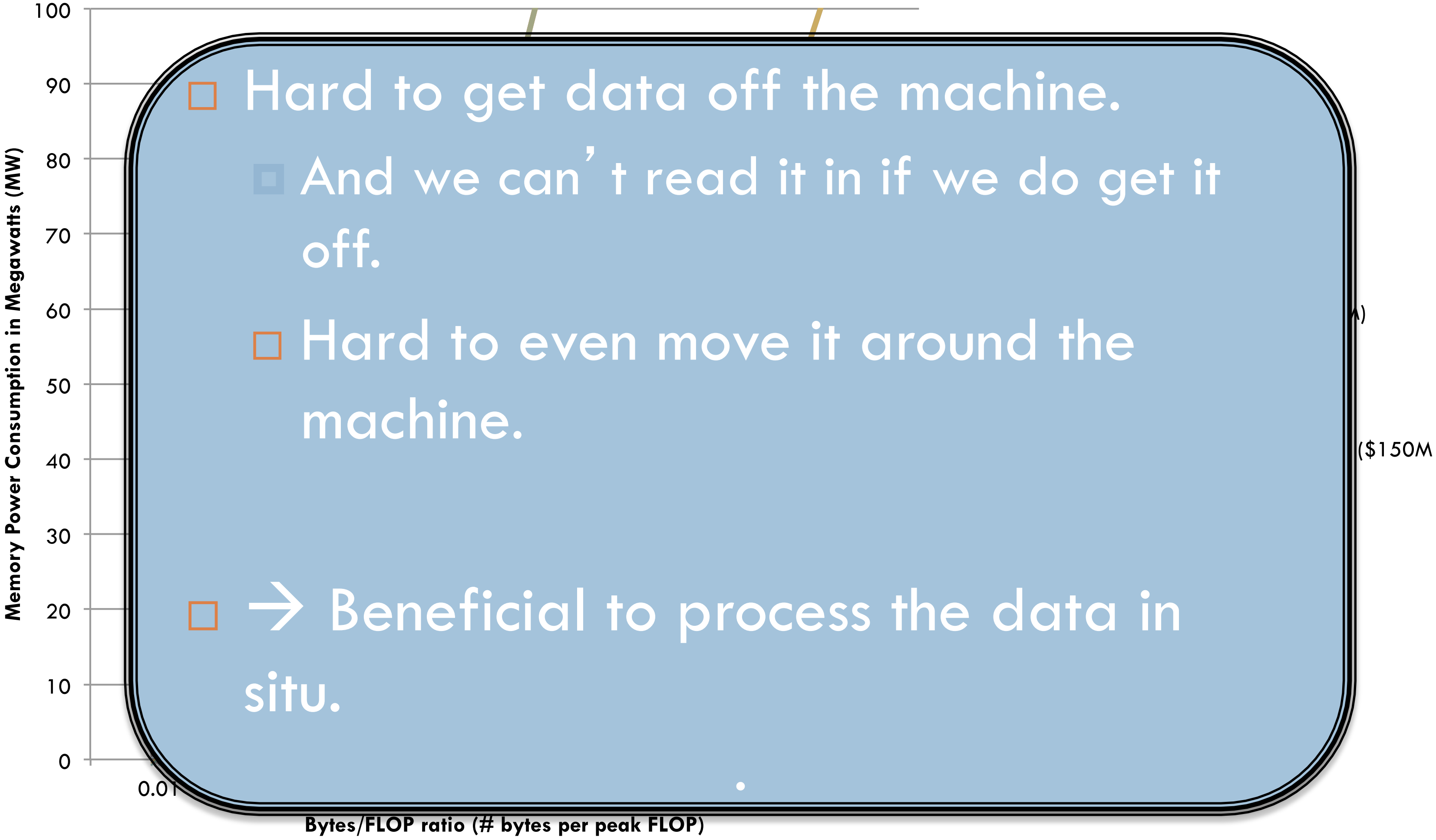
There are "smart techniques" that de-emphasize memory and I/O.

- Out of core
- Data subsetting
- Multi-resolution
- In situ

- … the community is currently getting these techniques deployed in production tools.

- This will be the primary challenge of the <100PFLOP era.

Exascale hurdle: memory bandwidth eats up the entire power budget

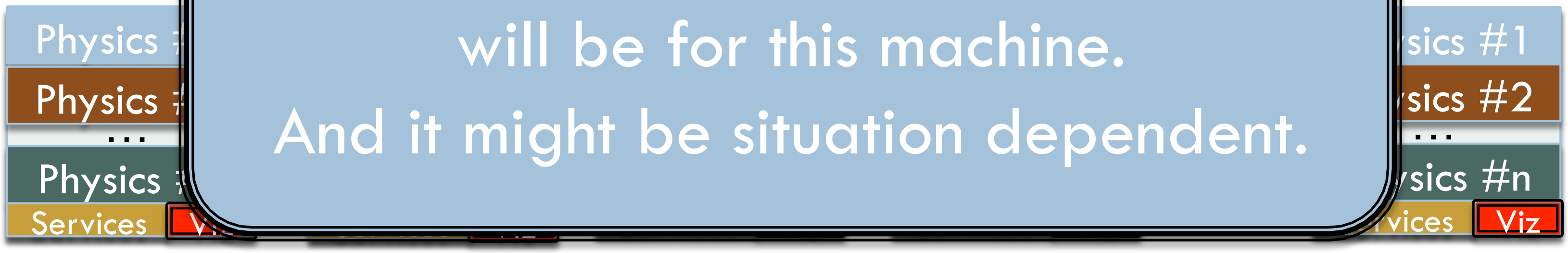


- Hard to get data off the machine.
  - And we can't read it in if we do get it off.
  - Hard to even move it around the machine.
- → Beneficial to process the data in situ.

c/o John Shalf, LBNL

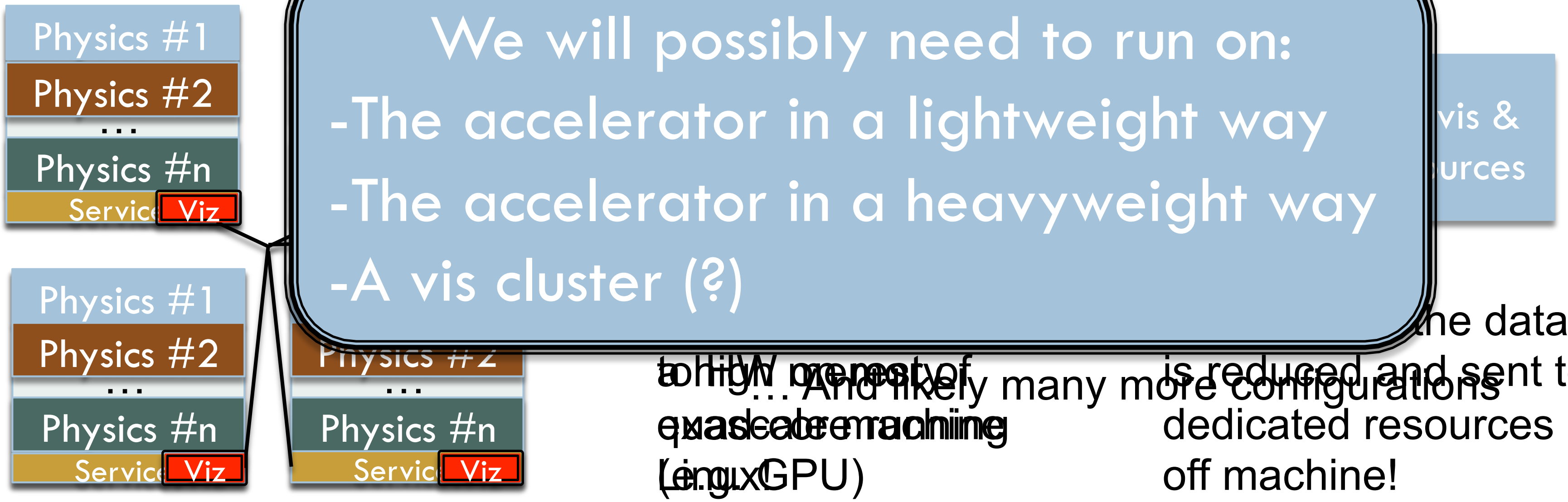
Possible in situ visualization scenarios

Visualization



We don’t know what the best technique will be for this machine.
And it might be situation dependent.

… or visualization could be done on a separate node located nearby dedicated to visualization/analysis/IO/etc. (loosely coupled)



We will possibly need to run on:
- -The accelerator in a lightweight way
- -The accelerator in a heavyweight way
- -A vis cluster (?)

a HW partition of exascale machine (e.g. GPU)

to high memory quad-core running Linux!

… And likely many more configurations

the data is reduced and sent to dedicated resources off machine!

Additional exascale challenges

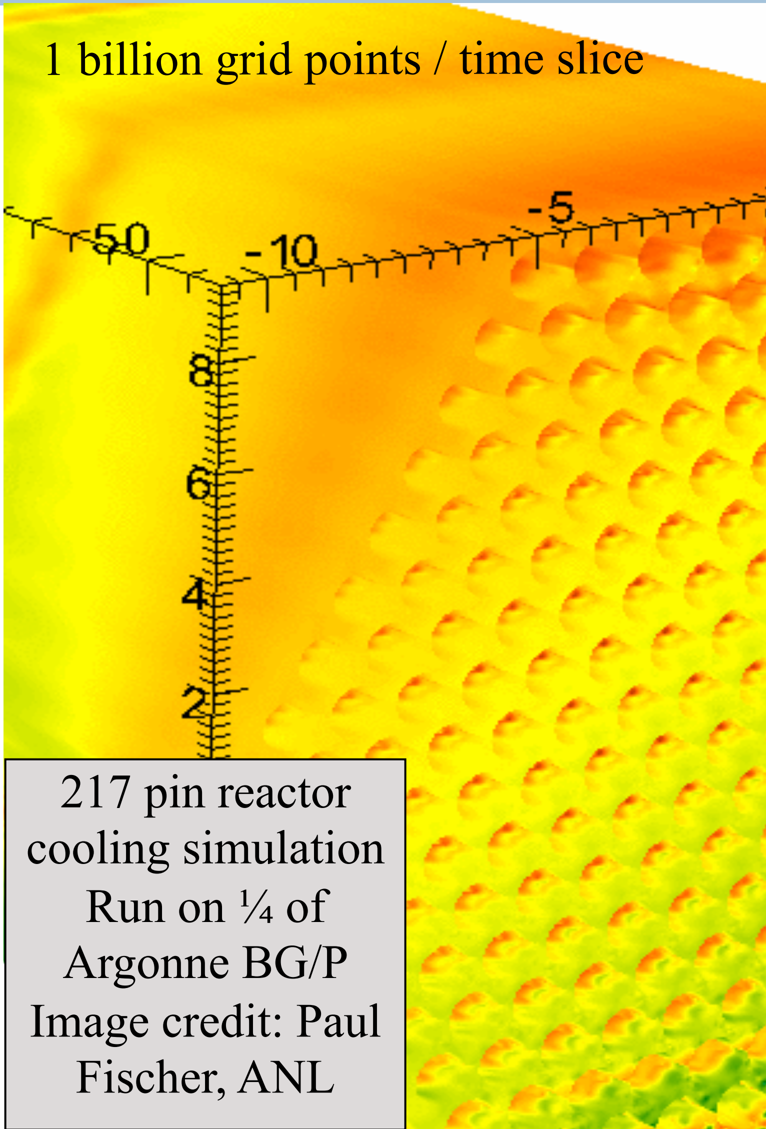
- Programming language:
  - OpenCL? Domain-specific language?
  - We have a substantial investment in CPU code; we can’t even get started on migrating until language is resolved.
- Memory efficiency
- How do we explore data?
  - In situ reductions that are post-processed afterwards?
- Resiliency
- New types of data – massive ensembles, multi-physics, etc – will require new techniques
- Reducing complexity

Tools: the VDA community achieves an economy of scale by collectively developing a shared infrastructure that is used by many application areas.

Food co-op, Ralston, Iowa

`VisIt` is an open source, richly featured, turn-key application for large data.

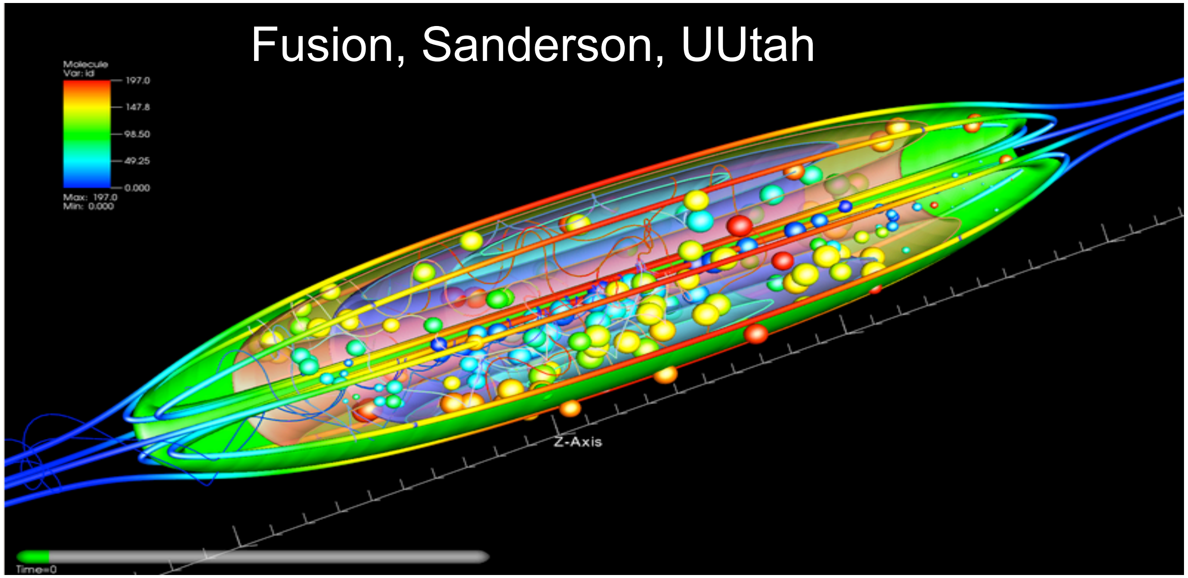
- Terribly named!!!:
  - Visual debugging
  - Quantitative & comparative analysis
  - Data exploration
  - Presentations
- Popular
  - R&D 100 award in 2005
  - Used on many of the Top500
  - >>>100K downloads

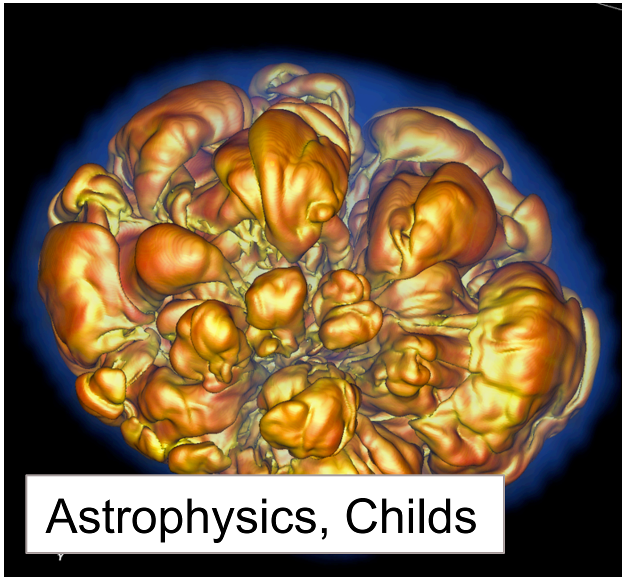


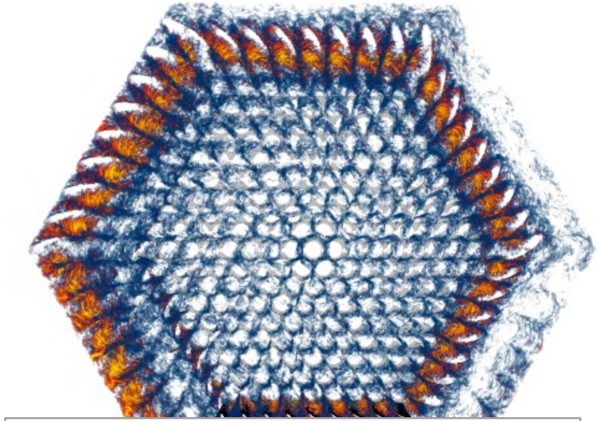
1 billion grid points / time slice

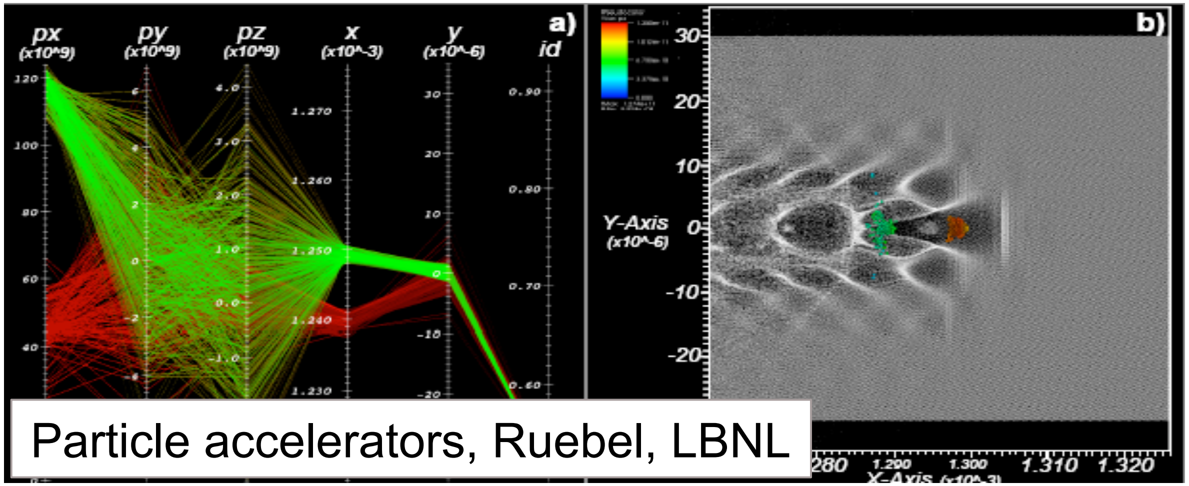
217 pin reactor cooling simulation
Run on ¼ of Argonne BG/P
Image credit: Paul Fischer, ANL

VisIt is used to look at lots of types of simulated and experimental data.


Fusion, Sanderson, UUtah


Astrophysics, Childs

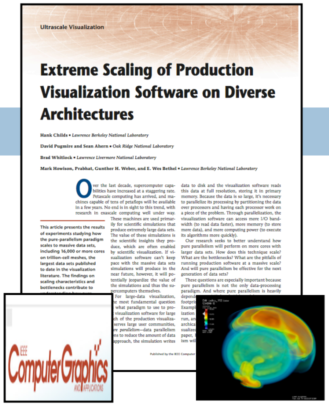

Nuclear Reactors, Childs


Particle accelerators, Ruebel, LBNL

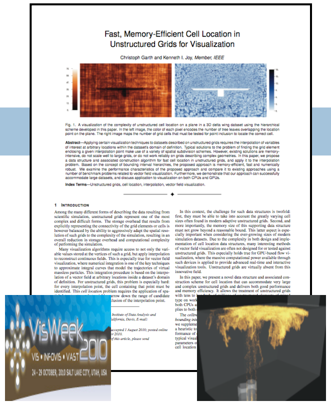
It has taken a lot of research to make VisIt work


A Contract Based System For Large Data Visualization

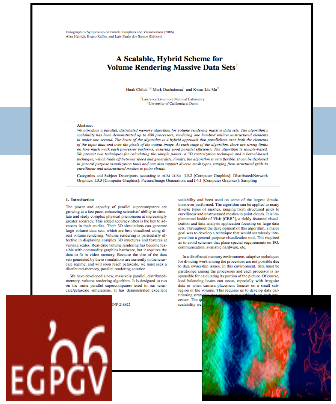
Systems research:
Adaptively applying algorithms in a production env.


Ultrascale Visualization
Extreme Scaling of Production Visualization Software on Diverse Architectures

Scaling research:
Scaling to 10Ks of cores and trillions of cells.


Fast, Memory-Efficient Cell Location in Unstructured Grids for Visualization

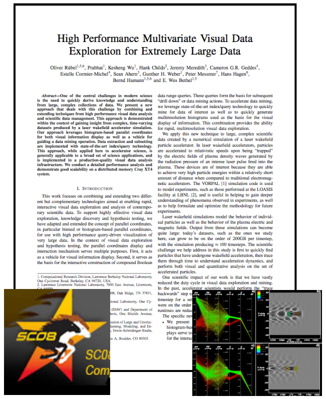
Algorithms research:
Accelerating field evaluation of huge unstructured grids


A Scalable, Hybrid Scheme for Volume Rendering Massive Data Sets

Algorithms research:
How to volume render efficiently in parallel.


Scalable Computation of Streamlines on Very Large Datasets

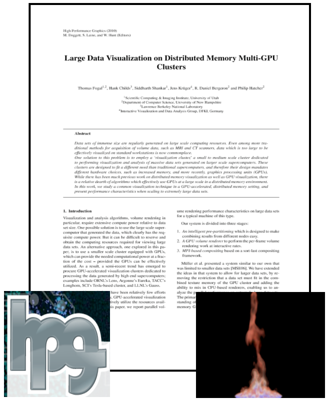
Algorithms research:
How to efficiently calculate particle paths in parallel.


High Performance Multivariate Visual Data Exploration for Extremely Large Data

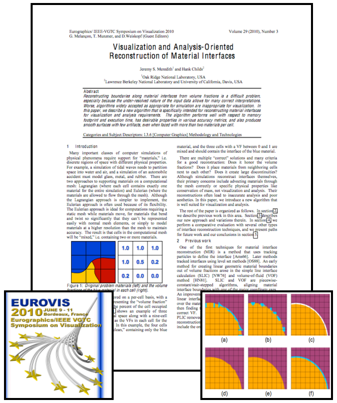
Systems research:
Using smart DB technology to accelerate processing


Streamline Integration using MPI-Hybrid Parallelism on a Large Multi-Core Architecture

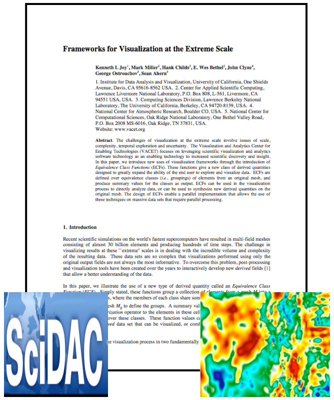
Architectural research:
Hybrid parallelism + particle advection


Large Data Visualization on Distributed Memory Multi-GPU Clusters

Architectural research:
Parallel GPU volume rendering


Visualization and Analysis-Oriented Reconstruction of Material Interfaces

Algorithms research:
Reconstructing material interfaces for visualization

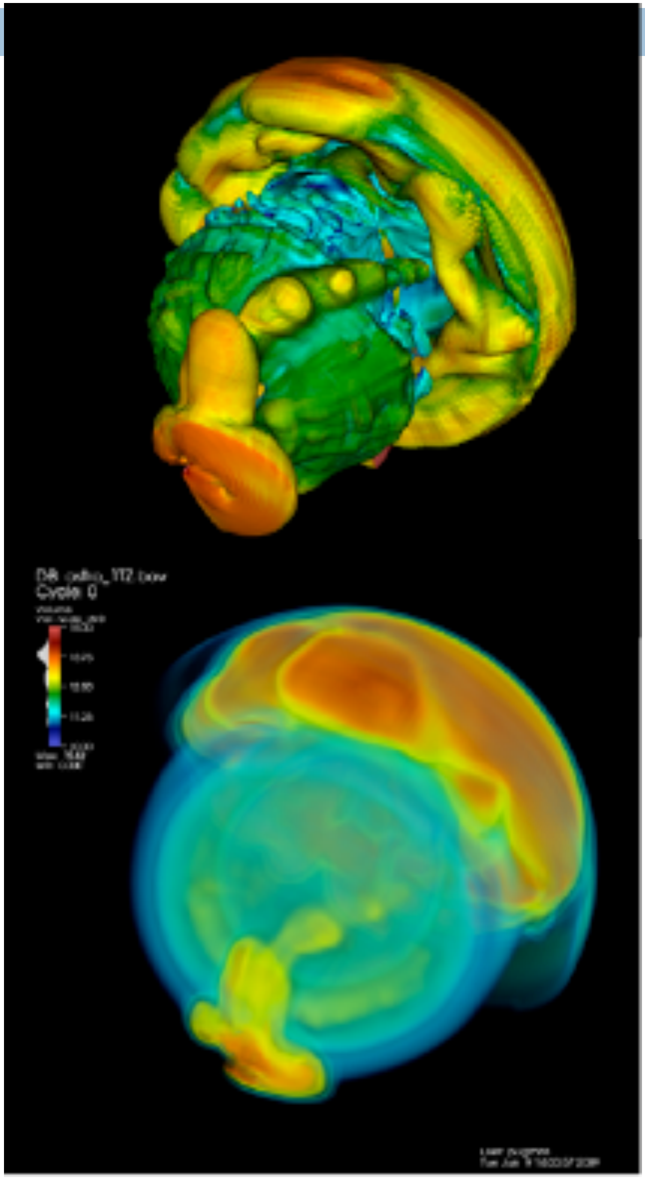

Frameworks for Visualization at the Exascale

Methods research:
How to incorporate statistics into visualization.

`VisIt` recently demonstrated good performance at unprecedented scale.

- Weak scaling study: ~62.5M cells/core

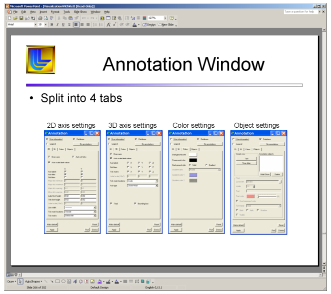
Machine	Model	Problem Size	#cores
Franklin	Cray XT4	1T, 2T	16K, 32K
Dawn	BG/P	4T	64K
JaguarPF	Cray XT5	2T	32K
Juno	X86_64	1T	16K
Purple	IBM P5	0.5T	8K
Ranger	Sun	1T	16K



Two trillion cell data set, rendered in `VisIt` by David Pugmire on ORNL Jaguar machine

The `VisIt` team focuses on making a robust, usable product for end users.

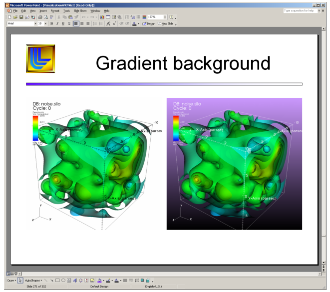
- Manuals
  - 300 page user manual
  - 200 page command line interface manual
  - "Getting your data into `VisIt`" manual
- Wiki for users (and developers)
- Revision control, nightly regression testing, etc
- Executables for all major platforms
- Day long class, complete with exercises



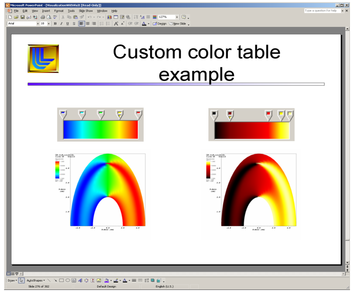
Annotation Window

- Split into 4 tabs

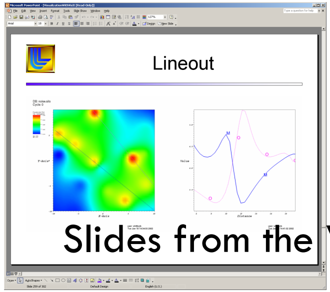
2D axis settings
3D axis settings
Color settings
Object settings



Gradient background



Custom color table example

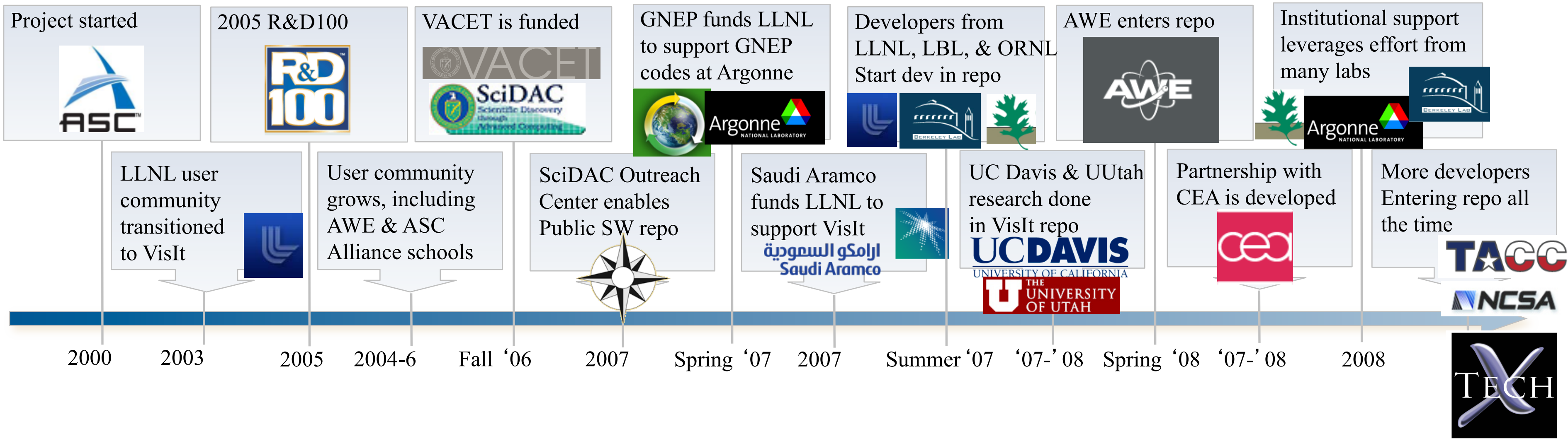


Lineout

Slides from the VisIt class

VisIt is a vibrant project with many participants.

- Over 75 person-years of effort
- Over 1.5 million lines of code
- Partnership between: Department of Energy's Office of Science, National Nuclear Security Agency, and Office of Nuclear Energy, the National Science Foundation XD centers (Longhorn XD and RDAV), and more….



2000 – Project started

2003 – LLNL user community transitioned to VisIt

2005 – 2005 R&D100

2004-6 – User community grows, including AWE & ASC Alliance schools

Fall '06 – VACET is funded

2007 – SciDAC Outreach Center enables Public SW repo

Spring '07 – GNEP funds LLNL to support GNEP codes at Argonne

2007 – Saudi Aramco funds LLNL to support VisIt

Summer '07 – Developers from LLNL, LBL, & ORNL Start dev in repo

'07-'08 – UC Davis & UUtah research done in VisIt repo

Spring '08 – AWE enters repo

'07-'08 – Partnership with CEA is developed

2008 – Institutional support leverages effort from many labs

More developers Entering repo all the time

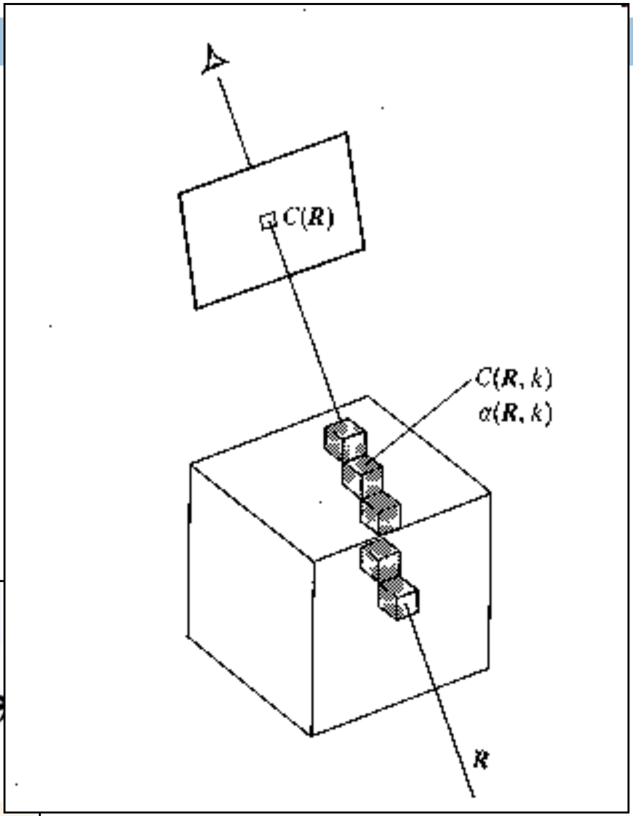
Achieving Extreme Performance

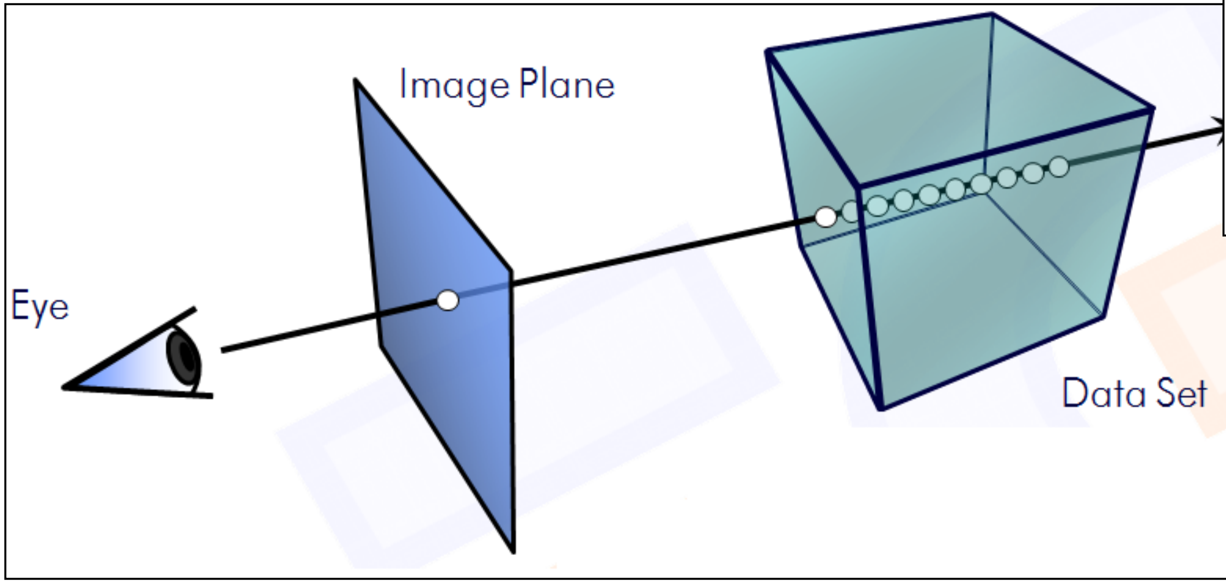
Achieving Extreme Performance

- The lessons of auto-tuning and multi-/many-core architectures:
  - Performance can vary by as much as 5x depending upon how tunable algorithmic parameters.
- Hybrid-parallelism:
  - MPI-only approaches not sustainable to extreme levels of concurrency.
  - Our results show hybrid parallelism runs faster, consumes less memory, requires less data movement.

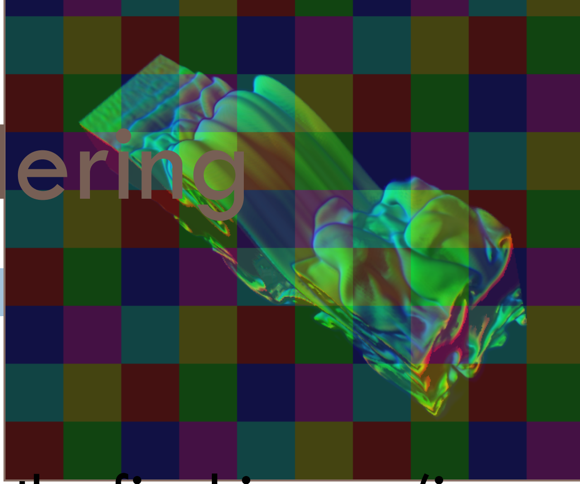
Algorithm Studied: Raycasting VR

- Overview of Levoy’s method
  - For each pixel in image plane:
    - Find intersection of ray and volume
    - Sample data (RGBa) along ray, integrate samples to compute final image pixel color

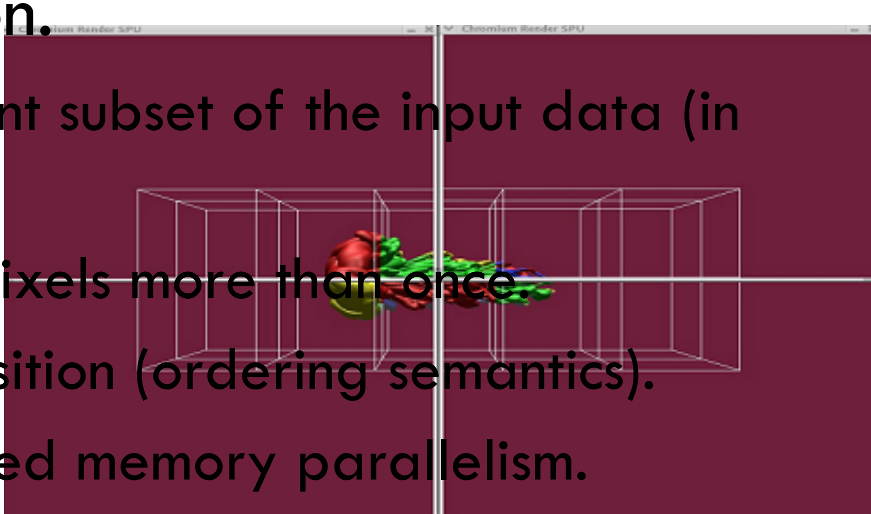




Parallelizing Volume Rendering



- Image-space decomposition.
  - Each process works on a disjoint subset of the final image (in parallel)
  - Processes may access source voxels more than once, will access a given output pixel only once.
  - Great for shared memory parallelism.
- Object-space decomposition.
  - Each process works on a disjoint subset of the input data (in parallel).
  - Processes may access output pixels more than once.
  - Output requires image composition (ordering semantics).
  - Typical approach for distributed memory parallelism.



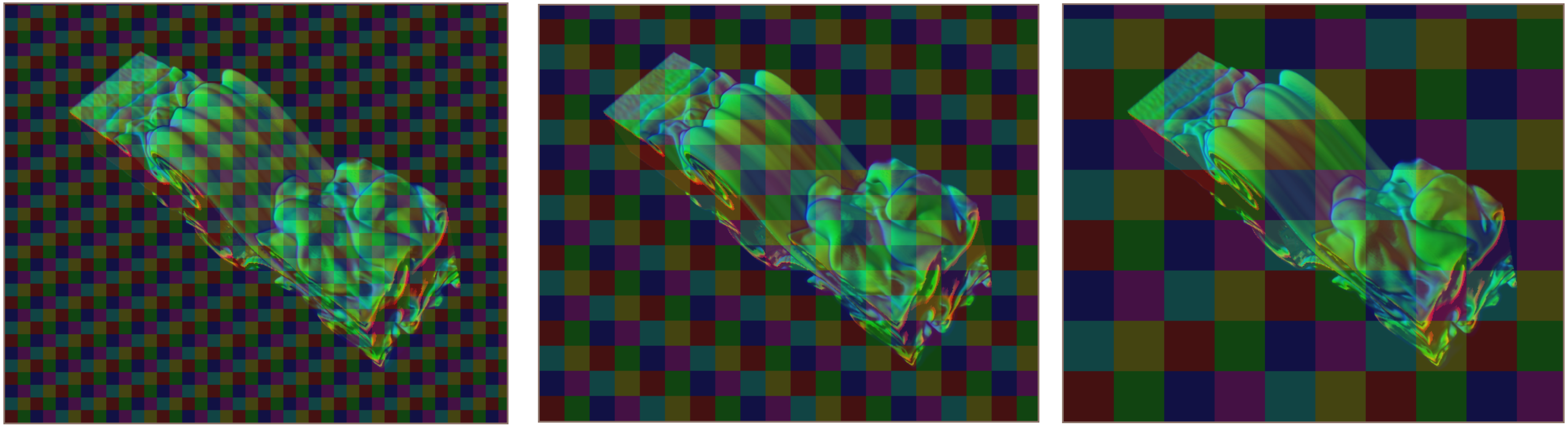
Autotuning and Performance Optimization

Motivation

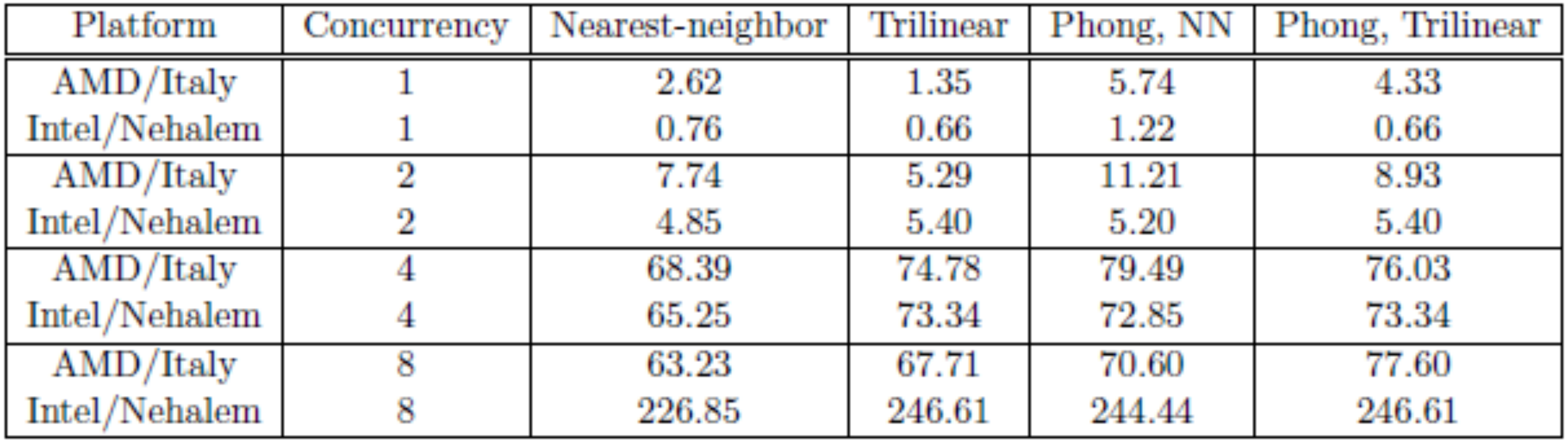
- Many algorithms have "tunable" parameters
  - CUDA: size/shape of thread block.
  - Image-parallel volume rendering: size of image tile.
- Choice of tunable parameters can have a huge impact on algorithm performance.
  - May vary by specific problem, architecture.
  - Examples that follow:
    - Shared-memory volume rendering: 2.5x difference between best and worst depending upon image tile size (unpublished work)
    - CUDA: 10x performance difference on stencil-based code (unpublished work, led to half of work for SC09 paper submission).
  - Other examples:
    - Multi-core CPU and GPU: 49x performance gain for a clinical medical imaging application. (Submitted to SC09)
    - Autotuning framework and multi-core CPUs and GPUs applied to stencil-based code. (CUG 2009, Best Paper Award)

Work Decomposition: Image Tile Size

- Final image divided into spatially disjoint regions, or work blocks.
- User specifiable block size/shape:
  - E.g., 8x8, 64x64, 512x1, 1x512
- Do some block sizes and shapes result in better performance than others?



Does Block Size/Shape Impact Performance?



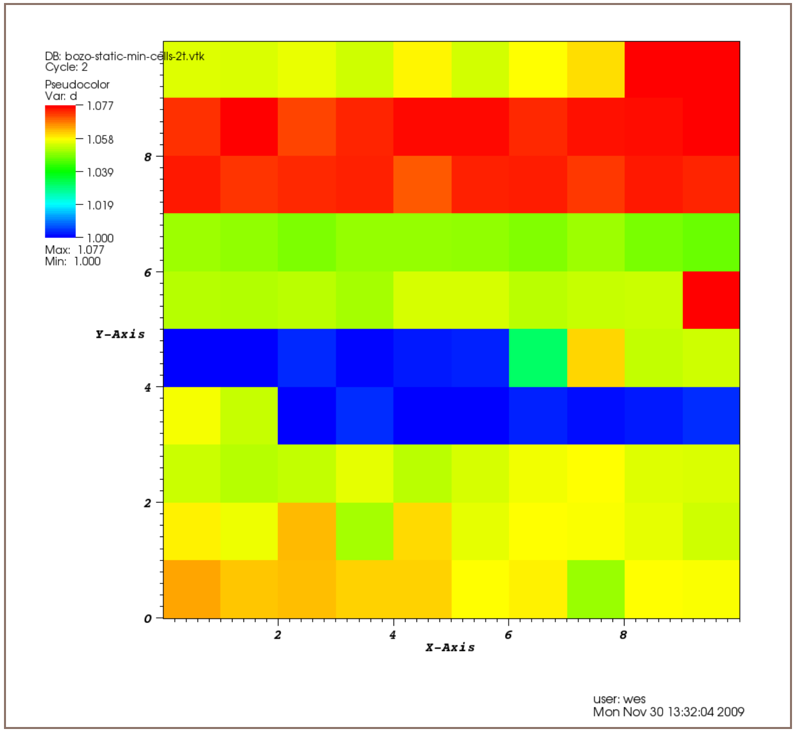
Platform	Concurrency	Nearest-neighbor	Trilinear	Phong, NN	Phong, Trilinear
AMD/Italy	1	2.62	1.35	5.74	4.33
Intel/Nehalem	1	0.76	0.66	1.22	0.66
AMD/Italy	2	7.74	5.29	11.21	8.93
Intel/Nehalem	2	4.85	5.40	5.20	5.40
AMD/Italy	4	68.39	74.78	79.49	76.03
Intel/Nehalem	4	65.25	73.34	72.85	73.34
AMD/Italy	8	63.23	67.71	70.60	77.60
Intel/Nehalem	8	226.85	246.61	244.44	246.61

Percent variation in runtime across entire test battery.

- Yes! Huge impact!
  - Greater variation at increasing concurrency.
  - Greater variation for more memory intensive algorithmic configurations.

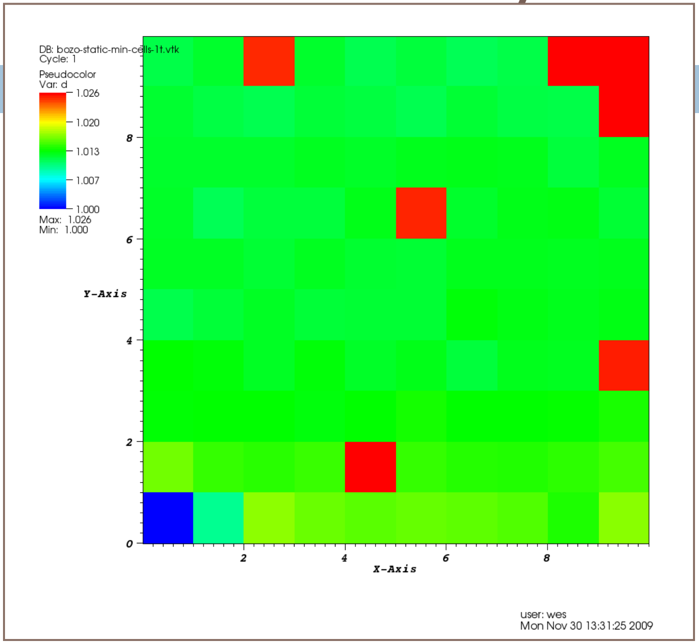
Optimal Block Size/Shape?

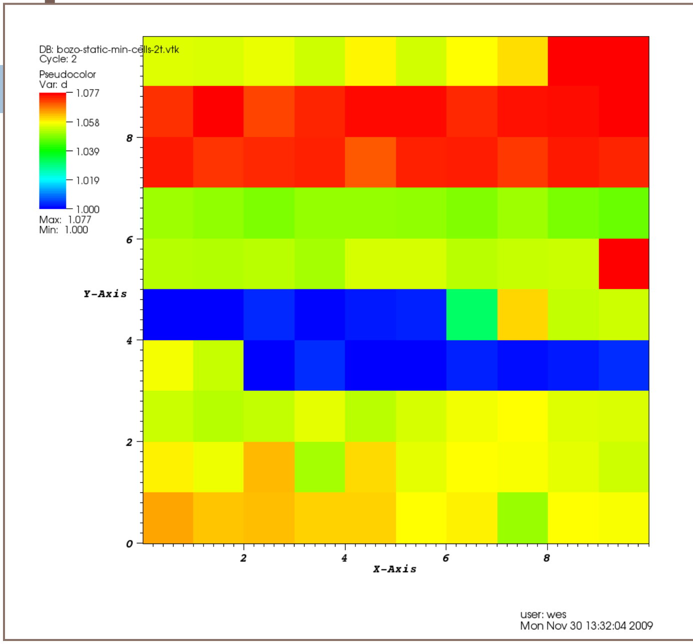
- Raw render time normalized by minimum render time value (shows sweet spots)
- This example:
  - 2 threads, AMD/Italy
  - X-axis/Y-axis: block sizes
  - Blue: sweet spot
  - Red: sour spot

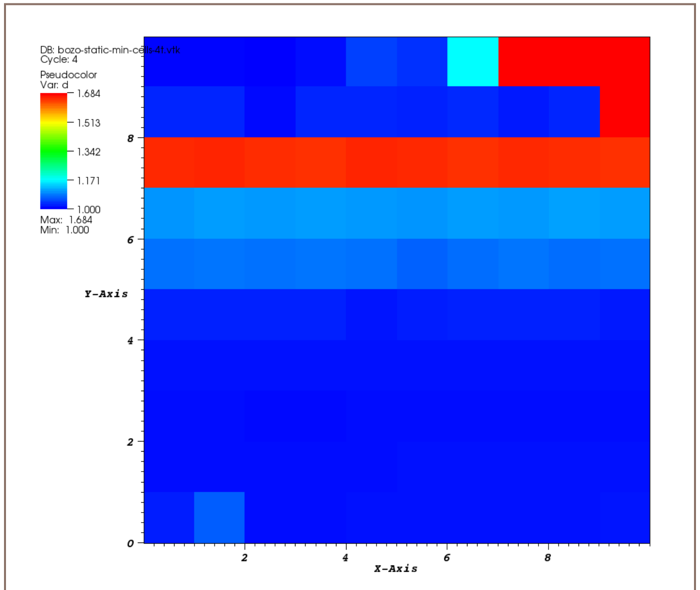


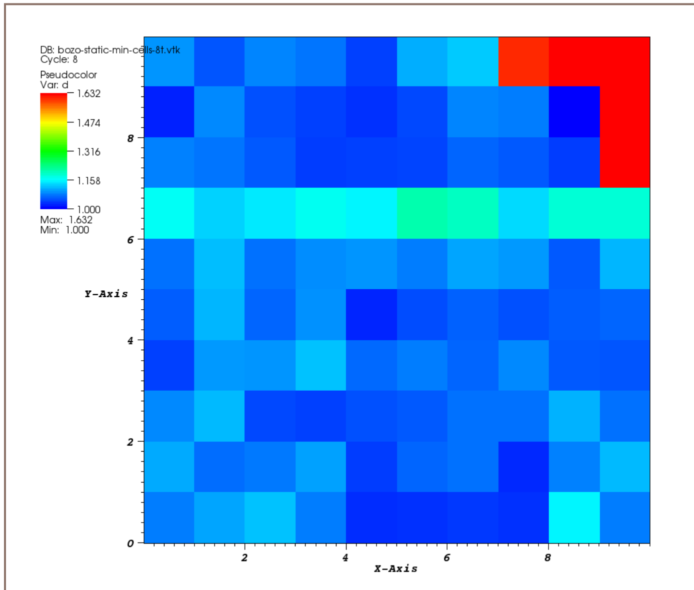
Optimal Block Size/Shape?

- Avg render time.
- 1, 2, 4, 8 threads.
- NN, NL.
- Bw=128 is really bad at t=2, 4, and bad at t-8.
- Dual socket, dual core machine.

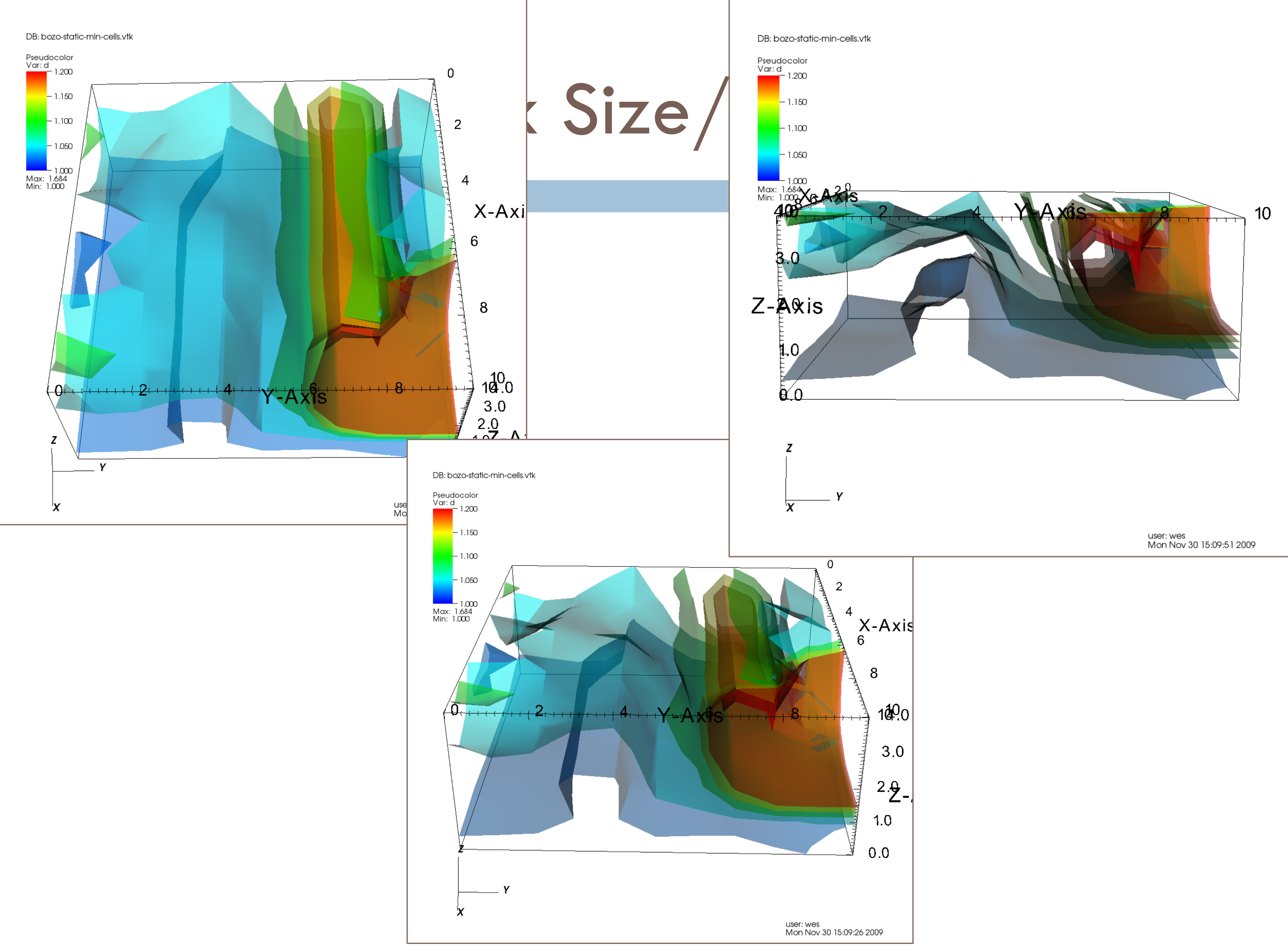






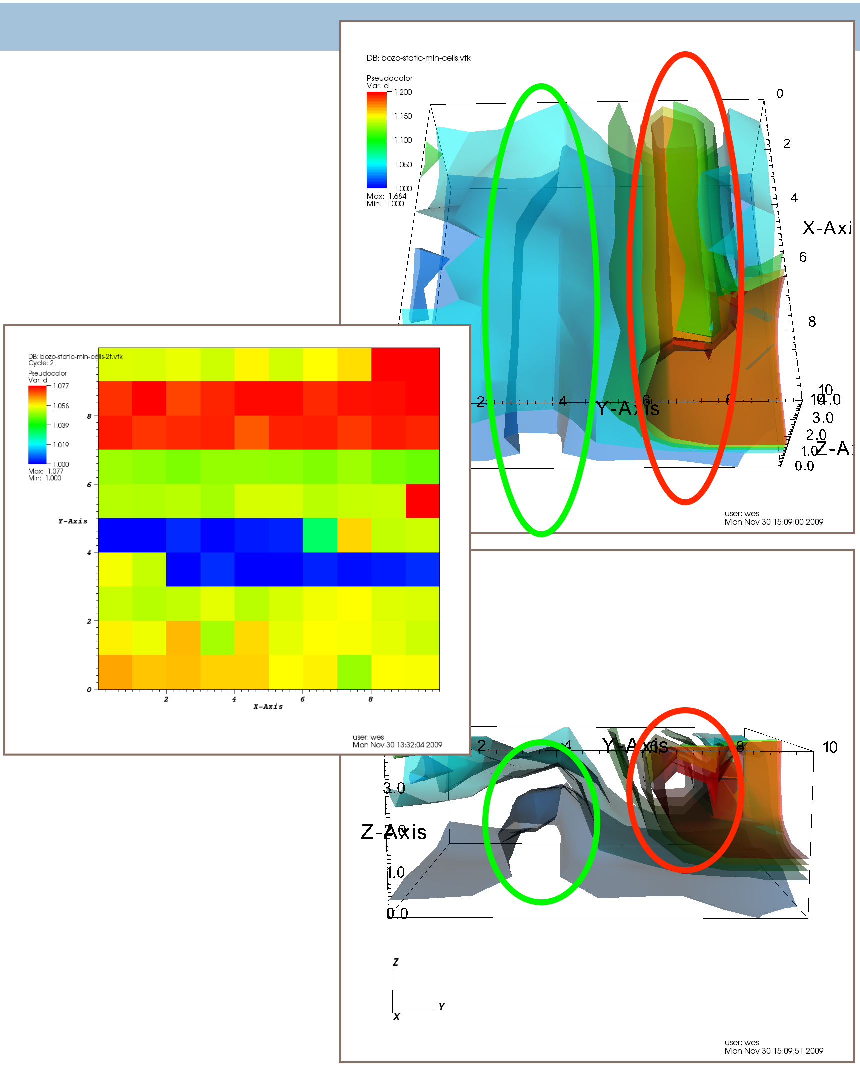


Size/


DB: bozo-static-min-cells.vtk
Pseudocolor
Var: d
1.200
1.150
1.100
1.050
1.000
Max: 1.684
Min: 1.000
X-Axis
Y-Axis
Z-Axis
Z
Y
X
user: wes
Mon Nov 30 15:09:51 2009
user: wes
Mon Nov 30 15:09:26 2009

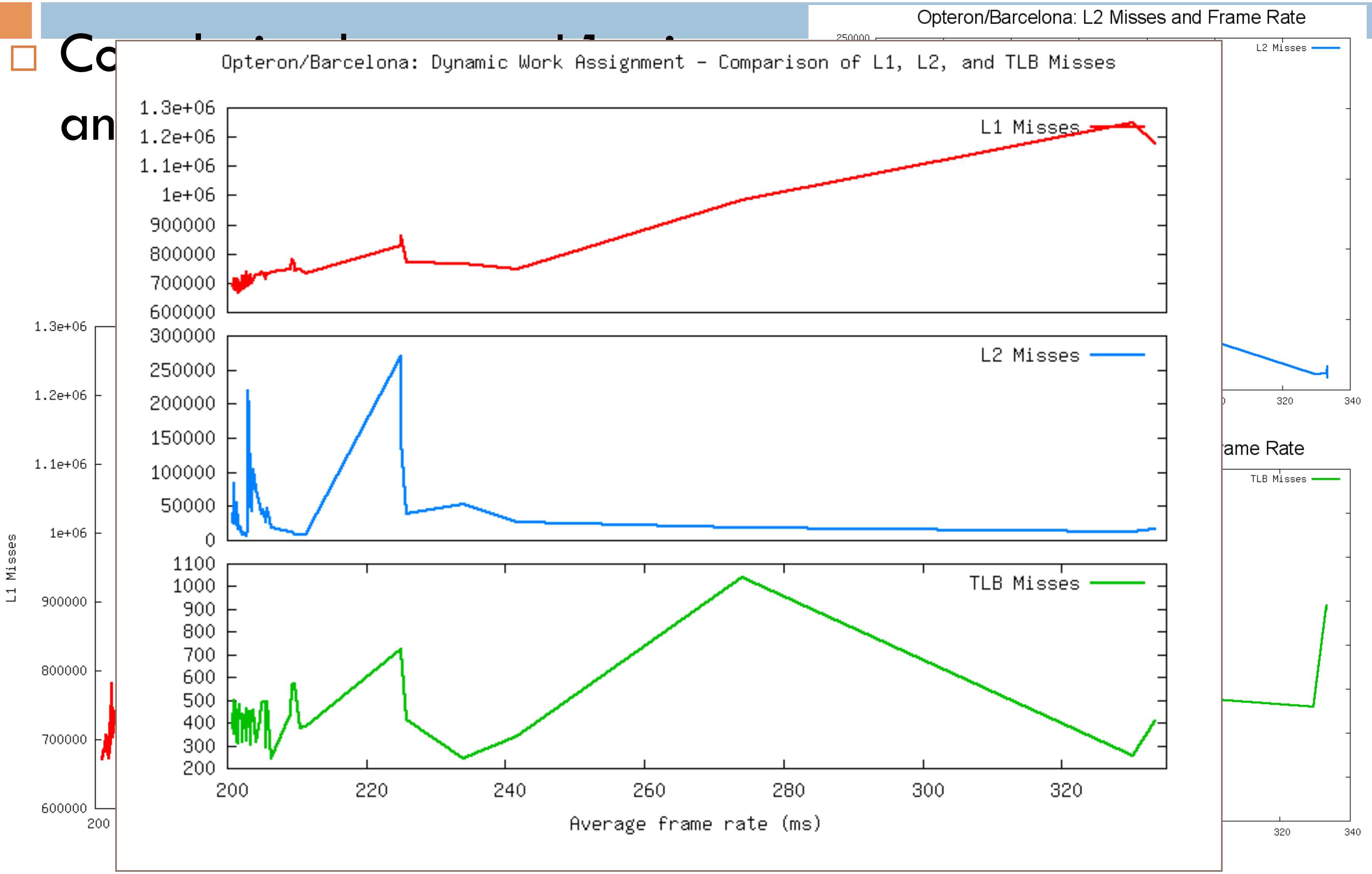
Optimal Block Size/Shape?

- AMD/Opteron-Italy
- Sweet spot/region: ○
  - Block width: 2^3-2^4
  - Block height: 2^2-2^6
- Sour spot/region: ○
  - Block width: 2^7-2^8
  - Block height: all



Cache Utilization and Block Size

- Co
  an



Many-core Platform: GPU

Performance Optimization and Auto-tuning: Summary

- Different settings for tunable algorithmic parameters can have a huge impact on performance on m-core platforms.
- Our results show roughly 2.5x variation on 6-core CPUs, up to 4.5x variation on GPUs.
- Code: unstructured memory access, largely memory bound rather than compute bound.
- Optimal settings from this study feed into the next study…

Hybrid Parallelism

State of Parallelism in Scientific Computing

- Most production codes written using MPI, vendor MPI implementations optimized for their architecture.
- HPC community wondering how well MPI will scale to high concurrency, particularly on 100-core CPUs.
- What to do?
  - Some alternatives: data parallel languages (CUDA), PGAS languages (UPC), global shared memory (CAF).
  - Various research projects explore different aspects of this space: Chombo in Titanium, autotuning, hybrid parallelism.

This Study

- First-ever study of hybrid parallelism on visualization: raycasting volume rendering.
  - Parallels similar work done for scientific computing.
- Hybrid-parallel implementation/architecture.
- Performance study.
  - Runs at 216K-way parallel: 6x larger than any published results (circa May 2010).
  - Look at:
    - Costs of initialization, Memory use comparison, Scalability, Absolute runtime.

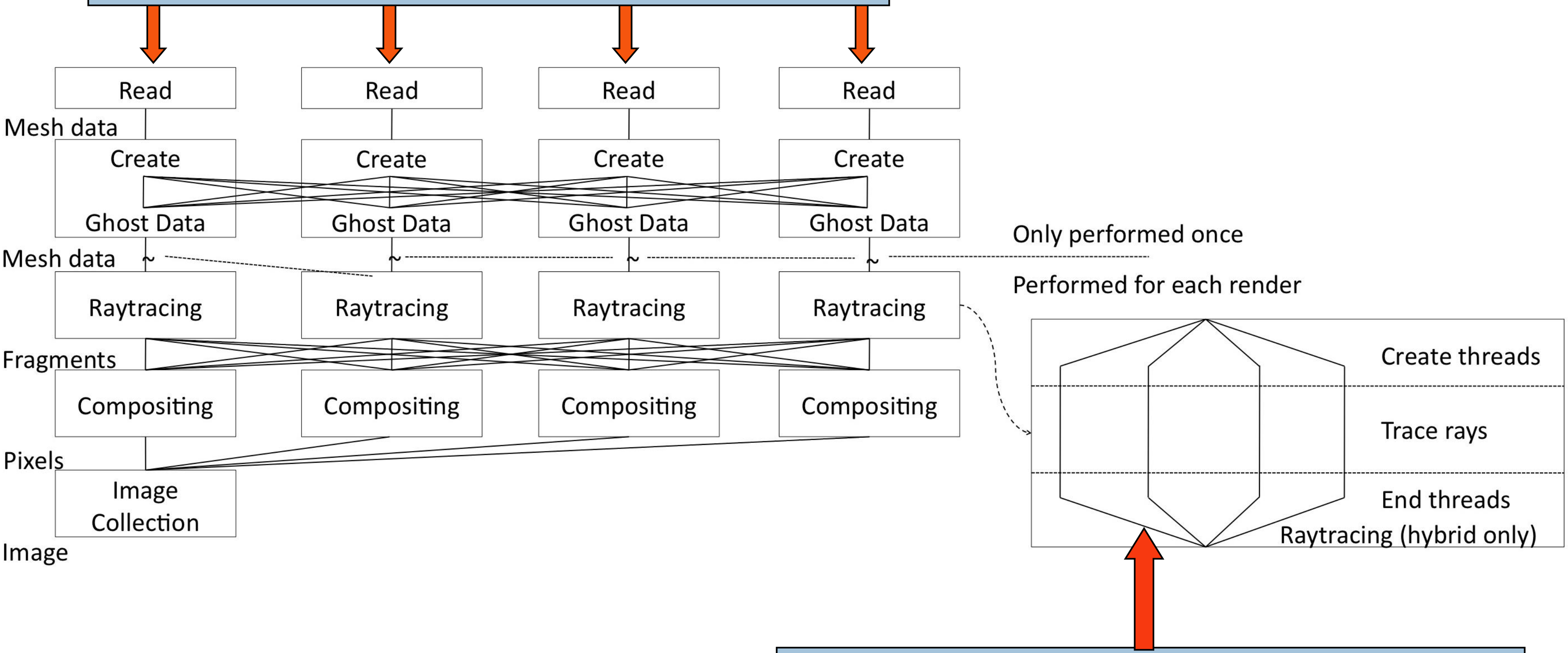
Hybrid Parallel Volume Rendering

- Hybrid-parallelism a blend of shared- and distributed-memory parallelism.
- Distributed-memory parallelism:
  - Each socket assigned a spatially disjoint subset of source data, produces an image of its chunk.
  - All subimages composited together into final image.
  - MPI implementation.
- Shared-memory parallelism:
  - Inside a socket, threads use image-space partitioning, each thread responsible for a subset of the final image.
    - What is the best image tile size? (Autotuning presentation)
  - Implementations (2): pthreads, OpenMP.

Hybrid Parallel Volume Rendering

ture:

Distributed-memory parallel



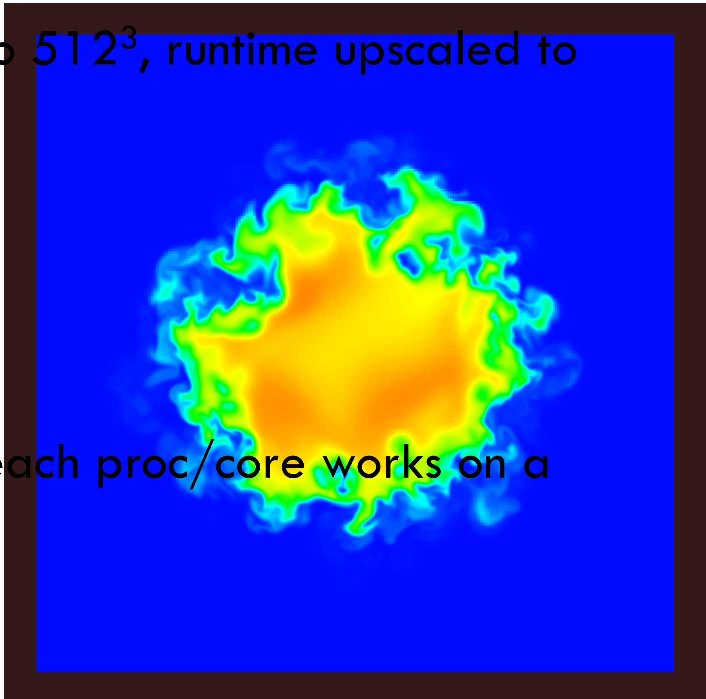
Shared memory parallel

Our Experiment

- Thesis: hybrid-parallel will exhibit favorable performance, resource utilization characteristics compared to traditional approach.
- How/what to measure?
  - Memory footprint, communication traffic load, scalability characteristics, absolute runtime.
  - Across a wide range of concurrencies.
  - Algorithm performance somewhat dependent upon viewpoint, data:
    - Vary viewpoints over a set that cut through data in different directions: will induce different memory access patterns.
- Strong scaling study: hold problem size constant, vary amount of resources.

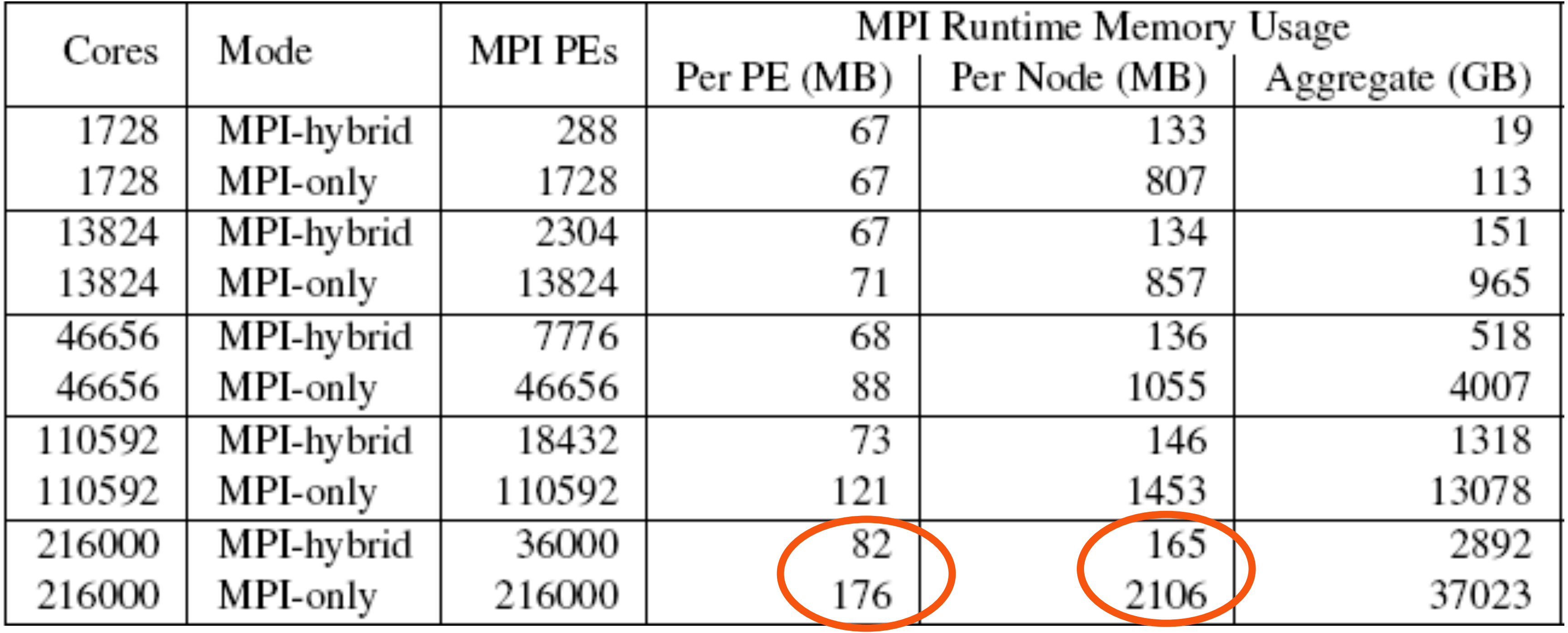
Experiment: Platform and Source Data

- Platform: JaguarPF, a Cray XT5 system at ORNL
  - 18,688 nodes, dual-socket, six-core AMD Opteron (224K cores)
- Source data:
  - Combustion simulation results, hydrogen flame (data courtesy J. Bell, CCSE, LBNL)
  - Effective AMR resolution: 1024^3, flattened to 512^3, runtime upscaled to 4608^3 (to avoid I/O costs).
- Target image size: 4608^2 image.
  - Want approx 1:1 voxels to pixels.
- Strong scaling study:
  - As we increase the number of procs/cores, each proc/core works on a smaller-sized problem.
  - Time-to-solution should drop.



Memory Use – MPI_Init()

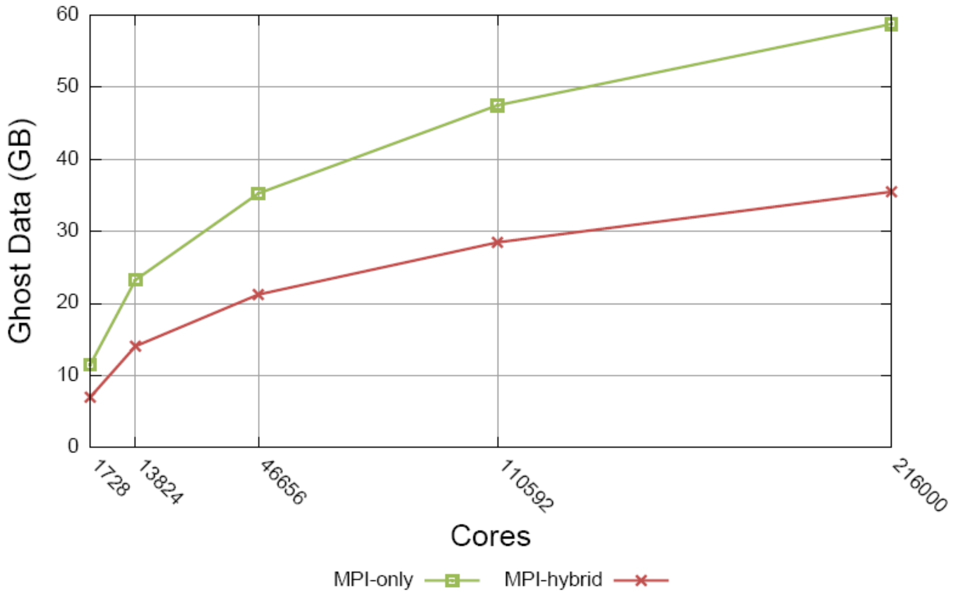
- Per PE memory:
  - About the same at 1728, over 2x at 216000.
- Aggregate memory use:
  - About 6x at 1728, about 12x at 216000.



Cores	Mode	MPI PEs	MPI Runtime Memory Usage		
			Per PE (MB)	Per Node (MB)	Aggregate (GB)
1728	MPI-hybrid	288	67	133	19
1728	MPI-only	1728	67	807	113
13824	MPI-hybrid	2304	67	134	151
13824	MPI-only	13824	71	857	965
46656	MPI-hybrid	7776	68	136	518
46656	MPI-only	46656	88	1055	4007
110592	MPI-hybrid	18432	73	146	1318
110592	MPI-only	110592	121	1453	13078
216000	MPI-hybrid	36000	82	165	2892
216000	MPI-only	216000	176	2106	37023

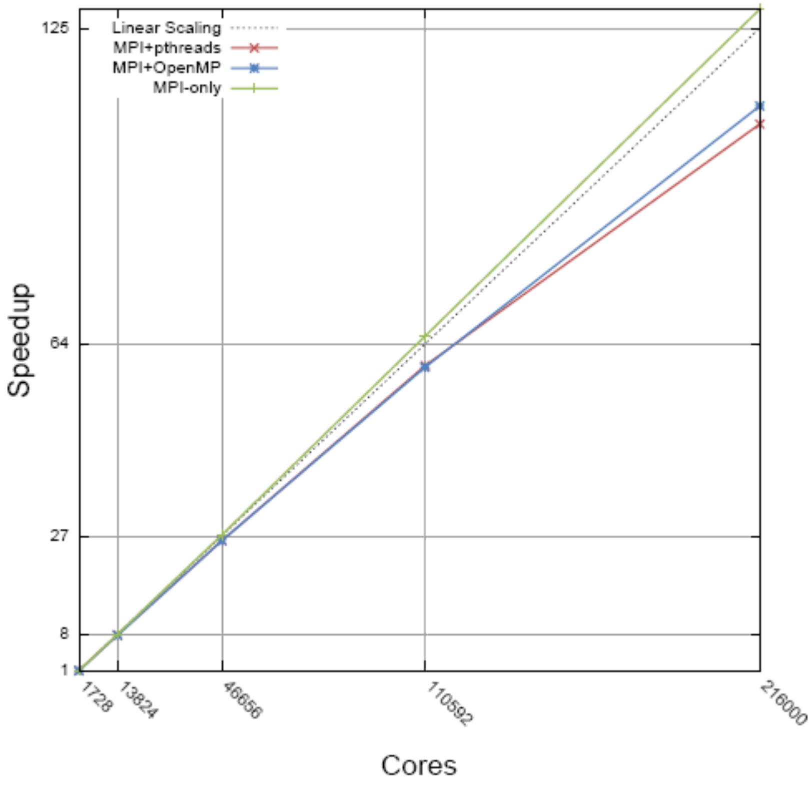
Memory Use – Ghost Zones

- Two layers of ghost cells required for this problem:
  - One for trilinear interpolation during ray integration loop.
  - Another for computing a gradient field (central differences) for shading.
- Hybrid approach uses fewer, but larger data blocks.
  - ~40% less memory required for ghost zones (smaller surface area)
  - Reduced communication costs



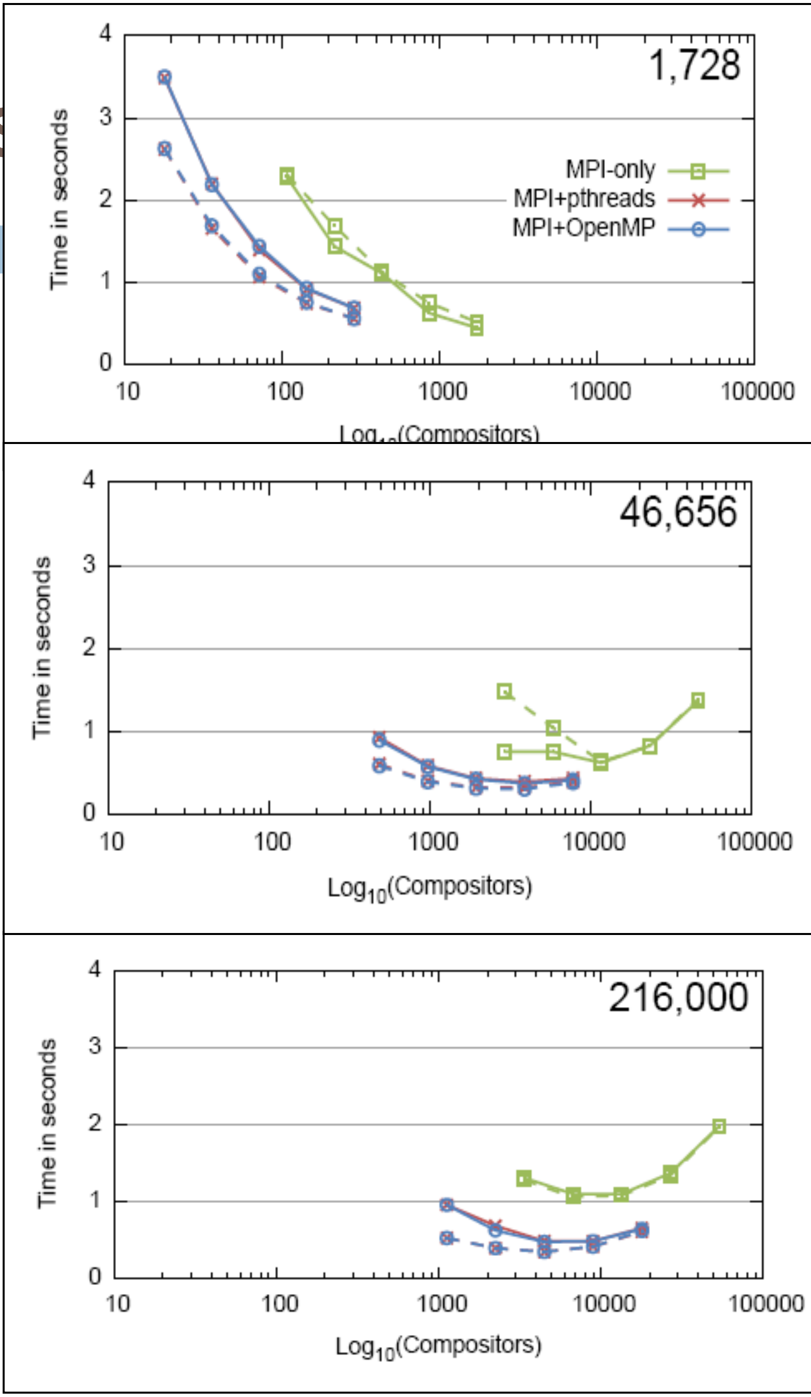
Scalability – Raycasting Phase

- Near linear scaling since no interprocess communication.
- -hybrid shows sublinear scaling due to oblong block shape.
- -only shows slightly better than linear due to reduced work caused by perspective foreshortening.



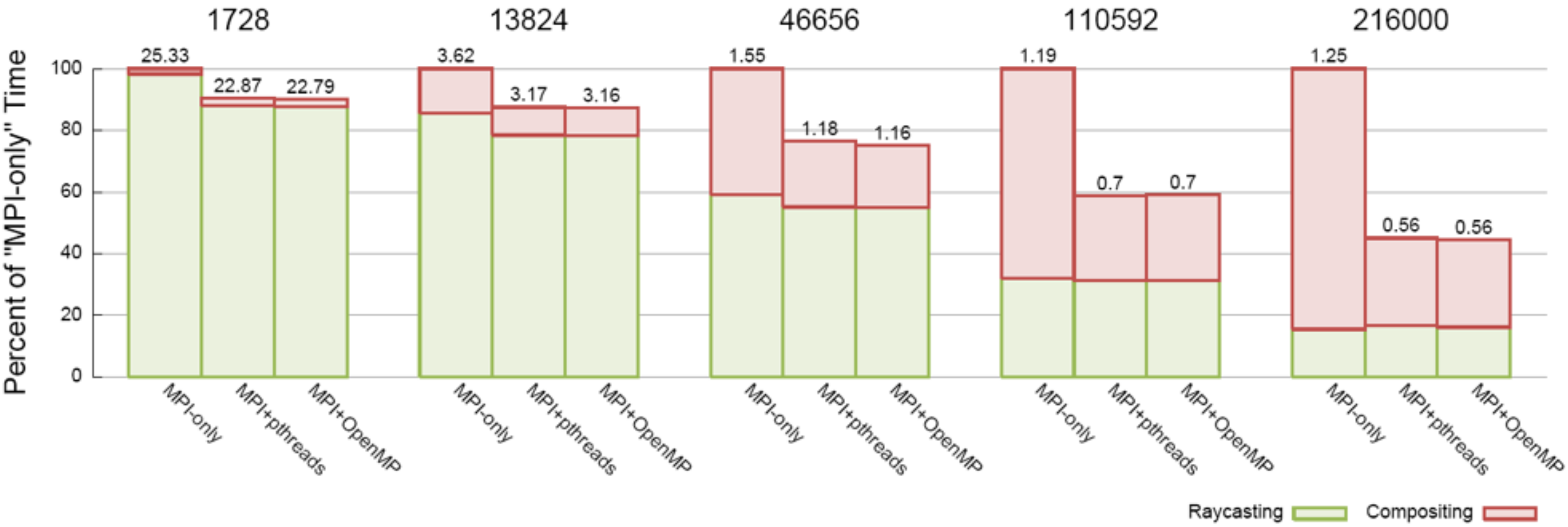
Scalability – Compos

- How many compositors to use?
  - Previous work: 1K to 2K for 32K renderers (Peterka, 2009).
  - Our work: above ~46K renderers, 4K to 8K works better.
  - -hybrid cases always performs better: fewer messages.
  - Open question: why the critical point?



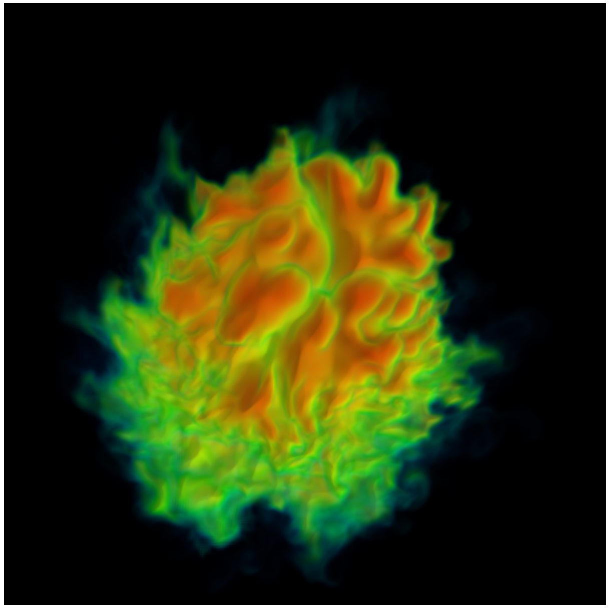
Absolute Runtime

- -hybrid outperforms –only at every concurrency level.
  - At 216K-way parallel, -hybrid is more than twice as fast as –only.



Summary of Results

- Absolute runtime: -hybrid twice as fast as –only at 216K-way parallel.
- Memory footprint: -only requires 12x more memory for MPI initialization then –hybrid
  - Factor of 6x due to 6x more MPI PEs.
  - Additional factor of 2x at high concurrency, likely a vendor MPI implementation (an N^2 effect).
- Communication traffic:
  - -hybrid performs 40% less communication than -only for ghost zone setup.
  - -only requires 6x the number of messages for compositing.
- Image: 4608^2 image of a ~4500^3 dataset generated using 216,000 cores on JaguarPF in ~0.5s (not counting I/O time).

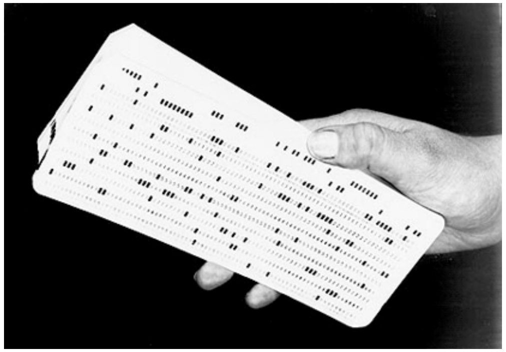


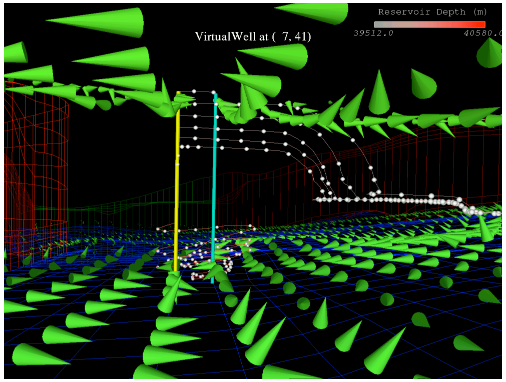
Examples/Case Studies

- 1995: Concurrent (in situ) visualization: UTCHEM simulation code and AVS.
- 2011: Advanced Simulation Capability for Environmental Management (ASCEM).

Reservoir modeling and VDA

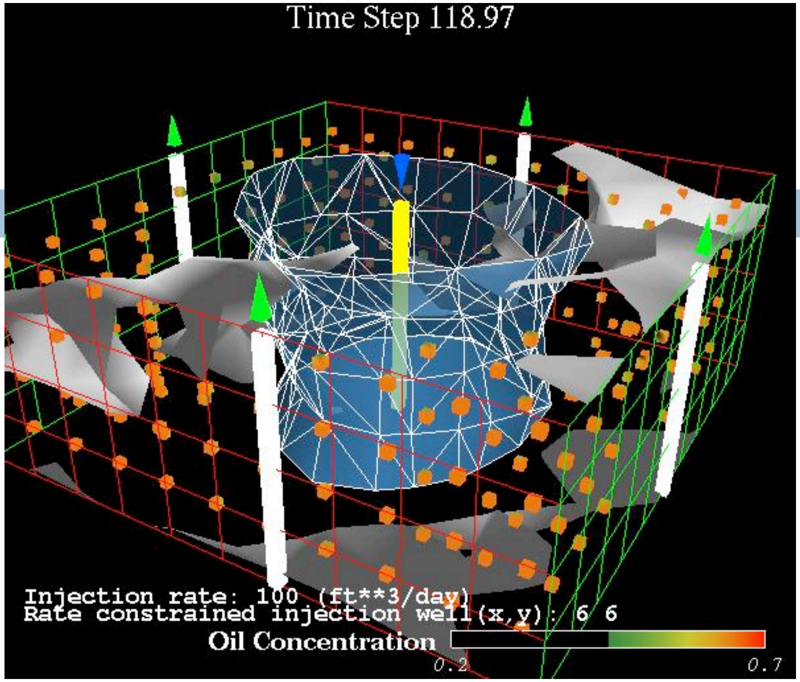
- Problem(s):
  - Setting up inputs (well locations) to optimize production (secondary, tertiary recovery, or EM applications) is tricky; input "card deck."
  - Simulations generate tabular output, difficult to quickly gain insight.
- Approach:
  - Couple model setup with intuitive input devices.
  - Couple simulation directly with VDA software.
  - Closed-loop system easy to use, quickly converge on optimal setup.



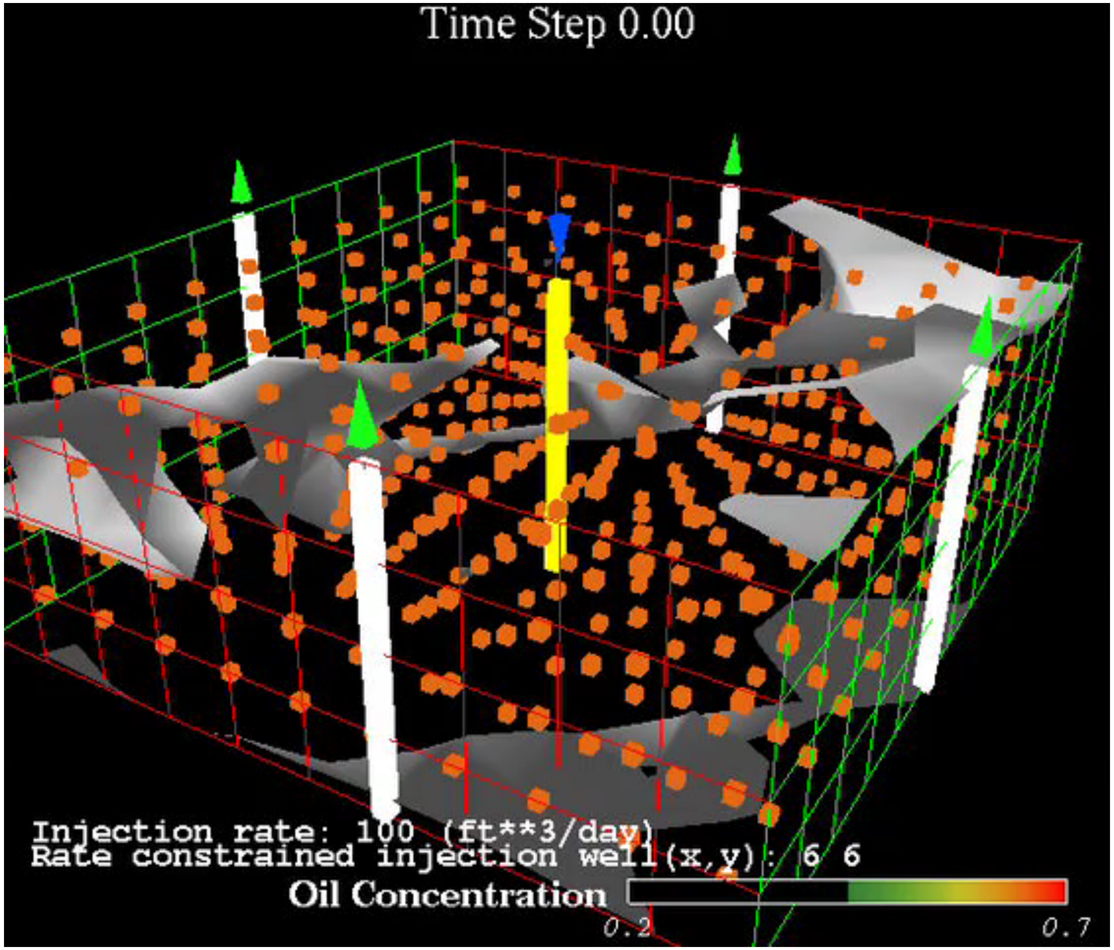


Problem Setup

- Placement of production and injection wells to optimize recovery/ mobilization.
- Supplant manual editing of card deck with 3D input device for easy well placement.
- (movie, next slide)



UTCHEM+AVS

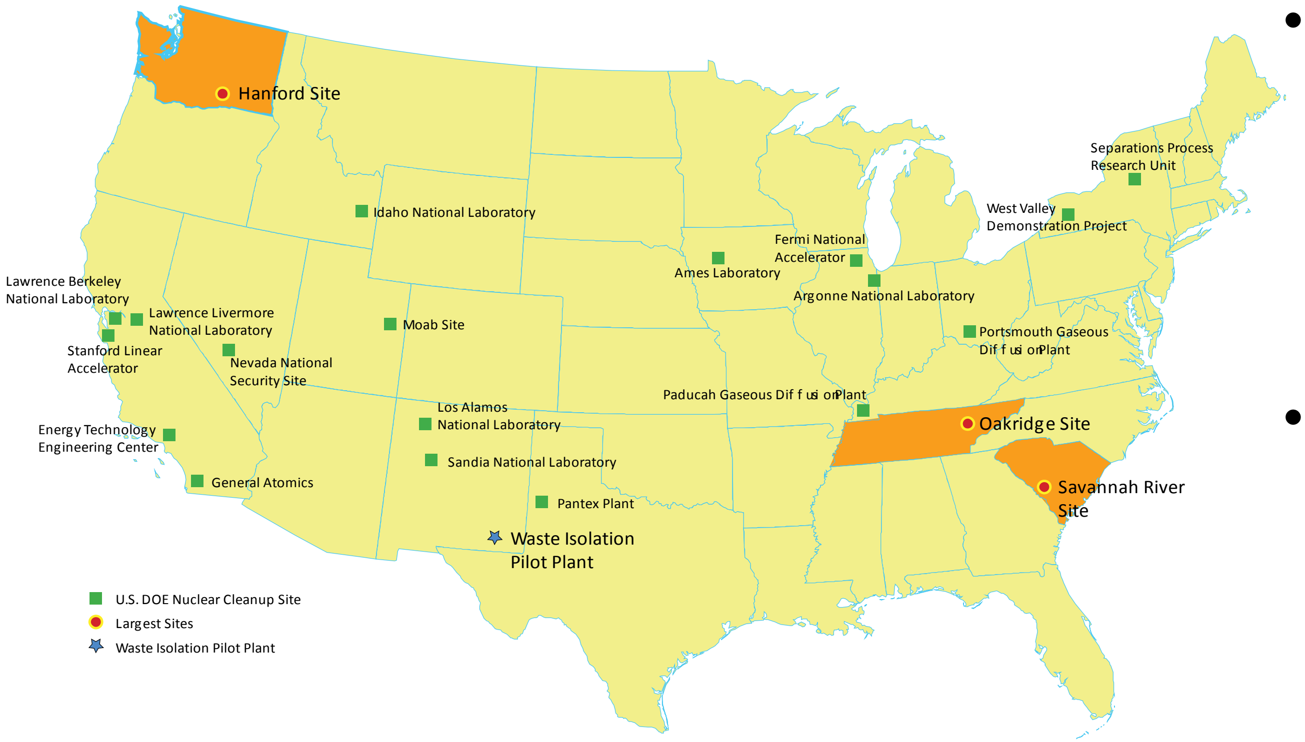


Movie file located here: http://vis.lbl.gov/Research/utchem-1993/images/utchem.mov

Environmental Management

Environmental Management - Mission

"Complete the safe cleanup of the environmental legacy brought about from five decades of nuclear weapons development, production, and Government-sponsored nuclear energy research."



- 6.4 trillion liters & 40 million cubic meters of contaminated groundwater and soil respectively
- Distributed across 30 states and 10,000 individual sites

Solving highly complex technical problems with transformational technologies can lead to billions of dollars of savings and improved clean up

The Challenge

- Current practices work for some sites and lead to closure
- Many of the subsurface contamination problems at DOE sites have no practical remedy

- Some successes: SRS F-Area groundwater contaminated with metals and radionuclides
  - Pump & treat system cost $1M/month to operate and generated waste products
  - Barrier system installed in 2006 costs <$10K/month and generates no waste products
- Need more options to address costly systems and meet regulatory requirements for closure of sites
  - Hanford pump & treat systems for the 200 Area cost ~ $10M/year
  - Oak Ridge mercury contamination in debris, soil, groundwater, and stream systems is estimated to cost $1B to meet regulatory requirements

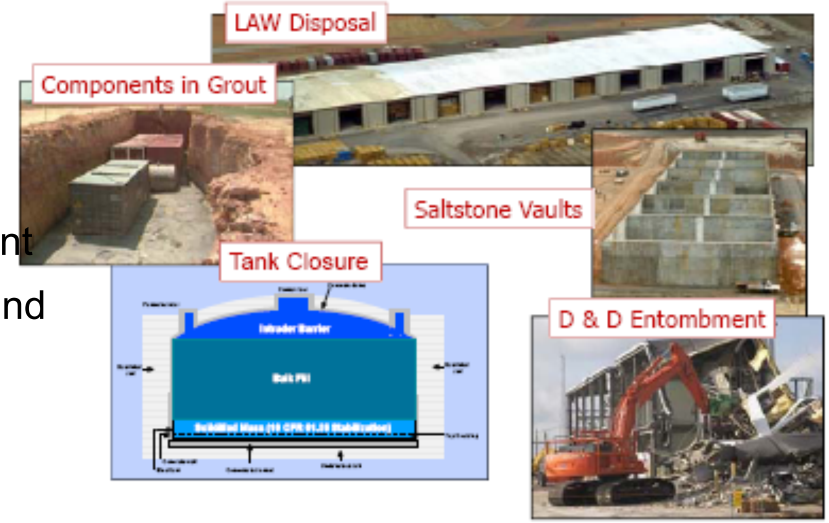
ASCEM Challenge and Impact

➢ Challenge

- Reduce time required and financial cost of remedial actions at sites within EM complex by providing scientifically defensible modeling and simulation tools that accurately address complex environmental management situations
- Develop an integrated, high-performance computer modeling capability to simulate multiphase, multi-component, multi-scale flow and contaminant transport, waste degradation and contaminant release, including
- <u>Provide (software) tools for decision making: parameter estimation, visualization, uncertainty quantification, data management, risk analysis, and decision support</u>
- Leverage investments made by SC, NE, RW, and FE as well as other Federal agencies to capitalize on significant investments and reduce the lifecycle development time and costs

➢ Impact

- Near-term: *technically underpin*” existing site RA’s and PA’s
- Inform strategic data collection for model improvement
- Scientifically defensible and standardized EM RA’s and PA’s



ASCEM Leverages SciDAC and ASC

- Significant leveraging of investments by Advanced Simulation and Computing (ASC /DOE NNSA) and Advanced Scientific Computing Research (ASCR/DOE SC)
- Examples include:
  - VisIt – visualization and graphic analysis tool developed by ASC and ASCR **SciDAC** Program
  - PSUADE – uncertainty analysis tool developed by **ASC**
  - Trilinos Framework – services for parallel programming and integrated software packages developed by **ASC** and ASCR **SciDAC** program
  - PETsc – Portable, Extensible Toolkit for Scientific Computation developed by ASCR **SciDAC** Program
  - BoxLib – parallel AMR framework developed by ASCR **Base Math** and **SciDAC**
  - MFD – Mimetic Finite Difference discretization methods developed by ASCR **Base Math** Program
  - Geochemistry Toolset – developed by computational scientists funded through DOE SC **BER**

ASCEM Delivered via a National Laboratory Consortium

BERKELEY LAB

Los Alamos
NATIONAL LABORATORY
EST. 1943

Pacific Northwest
NATIONAL LABORATORY

OAK RIDGE
National Laboratory

SRNL™
SAVANNAH RIVER NATIONAL LABORATORY

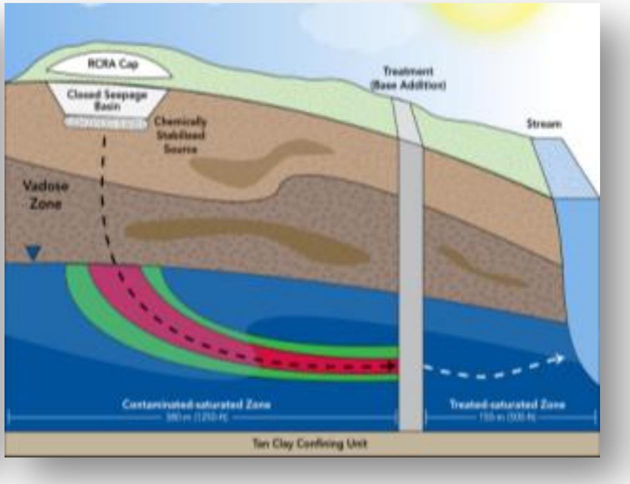
Lawrence Livermore National Laboratory
Science in the National Interest

Argonne
NATIONAL LABORATORY

INL
Idaho National Laboratory

Savannah River Site F-Area Background

- Disposal of low-level radioactive, acid waste solutions (1955–1989) created groundwater plume (pH 3–3.5, NO_3, U, ^{90}Sr, ^{129}I, ^{99}Tc, tritium)
- Ongoing remediation includes capping (1989), active pump and treat (1997-2003), and pH manipulation since 2004
- ***Natural attenuation is desired as a long-term remediation strategy***

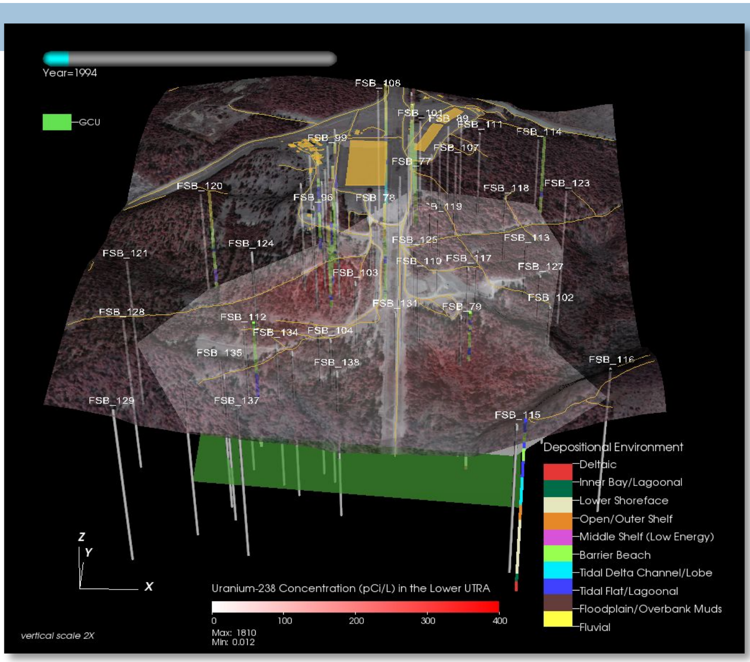


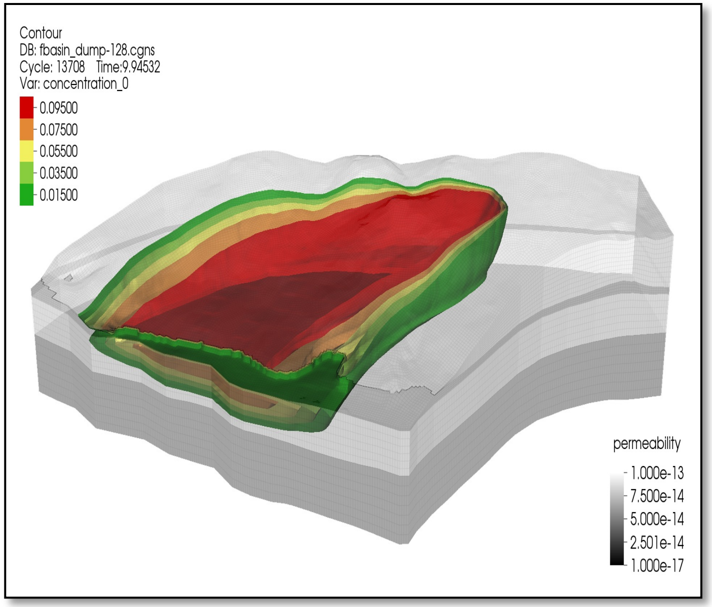
Goals: Phase/Year 1 Demonstration

Platform Data Management	Platform Visualization	Platform Uncertainty Quantification	HPC
• Depositional data entered into databases • Contaminant concentrations entered into databases • Browsing and query interface with tabled output	• 3D navigation of concentration data • Visualization of time lapse contaminant concentrations through subsurface domain • Selectively enable several types of data	• Initial Monte Carlo sampling capability • Automatic creation of forward simulation runs • Visualization of relationships between key parameters and model outputs	• 3D simulation • Richards' equation • Parallel (100 cores) • Reactive transport of uranium • Advection of non-reactive species • Aqueous speciation • Sorption • Mineral precipitation, and dissolution

ASCEM Visualization Year 1 Objectives

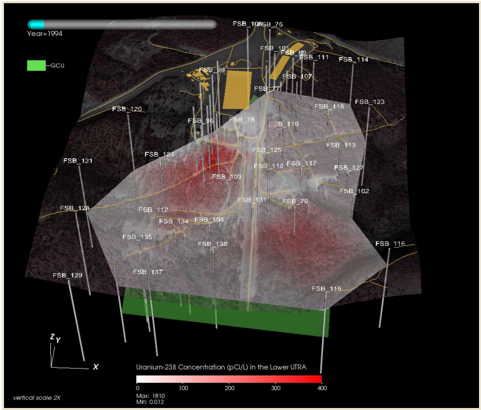
- Visual data exploration of
  - Part 1: Historical field data.
  - Part 2: Simulation data.
  - Part 3: Ensemble data.

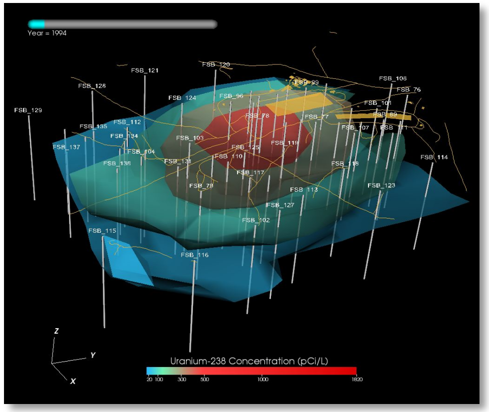




Visualization – Goals and Approach (Part 1)

- Visual data exploration of different types of geospatially registered data for the F-area seepage basin:
  - Observation wells, surface topography, depositional environment, observed concentration from many years, GIS data.
- Phase I Demonstration Objectives:
  - 3D navigation through the F-area historical concentration data.
  - Temporal browsing to show time evolving contaminant plume.
- Approach
  - Add new capabilities to well-established, production-quality, open source visual data analysis and exploration software infrastructure to meet ASCEM needs.
  - Demonstrate viability via application to ASCEM-specific problems.





ASCEM Animation

Movie file located here: http://vis.lbl.gov/Vignettes/ASCEM/ascem.mp4

Summary

- DOE and LBNL Visualization team studying different aspects of scalable visualization and analysis, deploying working technologies to the science research community.

- Terascale: Easy. Petascale: lots of work to do. Exascale: Hard.
  - Exascale requires us to rethink everything, and requires us to change course in terms of programming models/ languages, and in how we think about performance.


THE
END