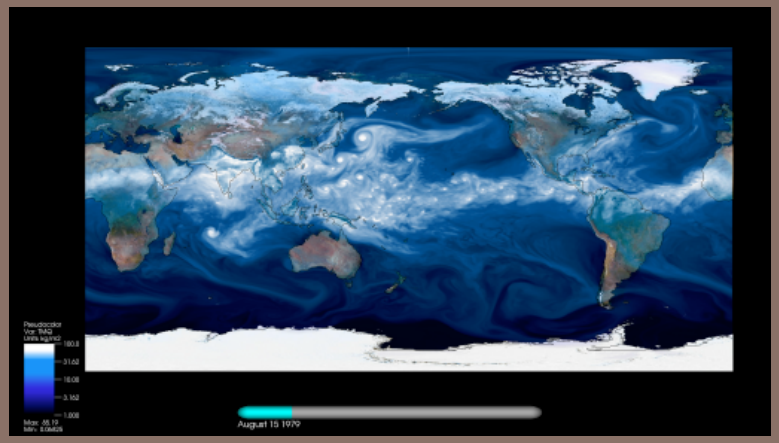
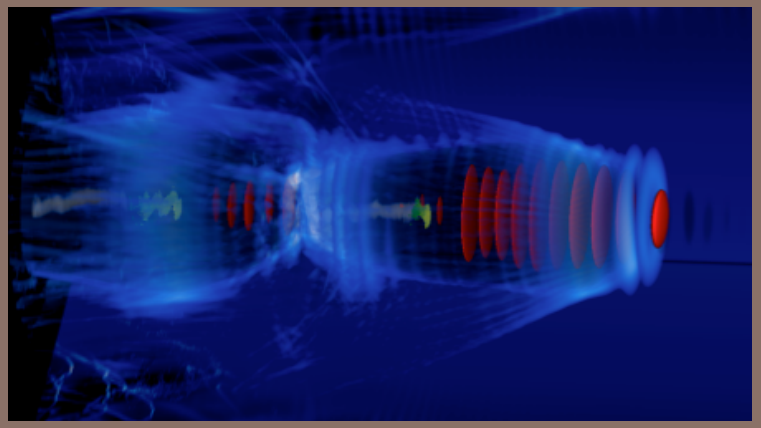


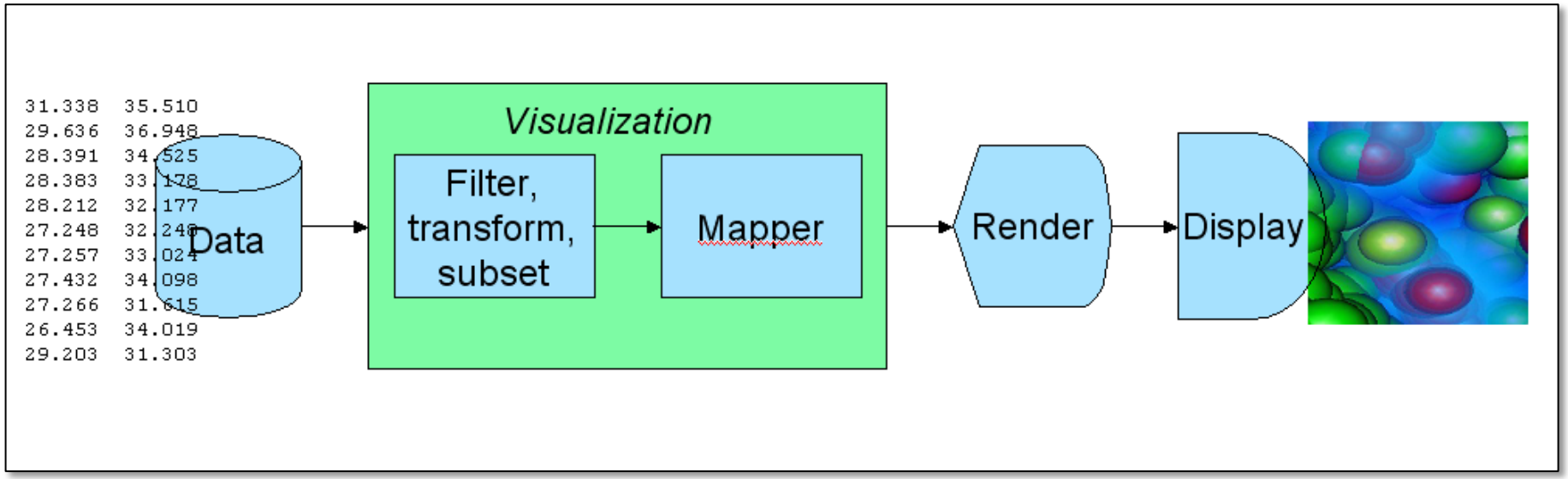


EXASCALE VISUALIZATION
GET READY FOR A WHOLE NEW WORLD
17 JUNE 2011

Wes Bethel, Lawrence Berkeley National Laboratory

Visualization 101

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



Supercomputing 101

- Why simulation?
  - Simulations are sometimes more cost effective than experiments.
  - New model for science has three legs: theory, experiment, and *simulation*.
- What is the "exascale"?
  - 1 FLOP = 1 Floating point operation per second
  - 1 GigaFLOP = 1 billion FLOPs, 1 TeraFLOP = 1000 GigaFLOPs
  - 1 PetaFLOP = 1,000,000 GigaFLOPs, 1 ExaFLOP = 1 billion GigaFLOPs
  - Petascale = PetaFLOPs (+ petabytes on disk + terabytes of memory)
  - Exascale = ExaFLOPs (+ exabyte disk + petabytes of memory)
- Why exascale?
  - More compute cycles, more memory, etc, lead for faster and/or more accurate simulations.

Fable: The Boy Who Cried Wolf

- Once there was a shepherd boy who had to look after a flock of sheep. One day, he felt bored and decided to play a trick on the villagers. He shouted, "Help! Wolf! Wolf!" The villagers heard his cries and rushed out of the village to help the shepherd boy. When they reached him, they asked, "Where is the wolf?" The shepherd boy laughed loudly, "Ha, Ha, Ha! I fooled all of you. I was only playing a trick on you."

Fable: The Boy Who Cried Wolf

- Once there was a viz expert who had to look after customers. One day, he felt bored and decided to play a trick on the villagers. He shouted, "Help! Wolf! Wolf!" The villagers heard his cries and rushed out of the village to help the shepherd boy. When they reached him, they asked, "Where is the wolf?" The shepherd boy laughed loudly, "Ha, Ha, Ha! I fooled all of you. I was only playing a trick on you."

Fable: The Boy Who Cried Wolf

- Once there was a viz expert who had to look after customers. One day, he needed funding and decided to play a trick on his funders. He shouted, "Help! Wolf! Wolf!" The villagers heard his cries and rushed out of the village to help the shepherd boy. When they reached him, they asked, "Where is the wolf?" The shepherd boy laughed loudly, "Ha, Ha, Ha! I fooled all of you. I was only playing a trick on you."

Fable: The Boy Who Cried Wolf

- Once there was a viz expert who had to look after customers. One day, he needed funding and decided to play a trick on his funders. He shouted, "Help! Big Big Data!" The villagers heard his cries and rushed out of the village to help the shepherd boy. When they reached him, they asked, "Where is the wolf?" The shepherd boy laughed loudly, "Ha, Ha, Ha! I fooled all of you. I was only playing a trick on you."

Fable: The Boy Who Cried Wolf

- Once there was a viz expert who had to look after customers. One day, he needed funding and decided to play a trick on his funders. He shouted, "Help! Big Big Data!" The funders heard his cries and sent lots of money to help the viz expert. When they reached him, they asked, "Where is the wolf?" The shepherd boy laughed loudly, "Ha, Ha, Ha! I fooled all of you. I was only playing a trick on you."

Fable: The Boy Who Cried Wolf

- Once there was a viz expert who had to look after customers. One day, he needed funding and decided to play a trick on his funders. He shouted, "Help! Big Big Data!" The funders heard his cries and sent lots of money to help the viz expert. When petascale arrived, they asked, "Where is the problem?" The shepherd boy laughed loudly, "Ha, Ha, Ha! I fooled all of you. I was only playing a trick on you."

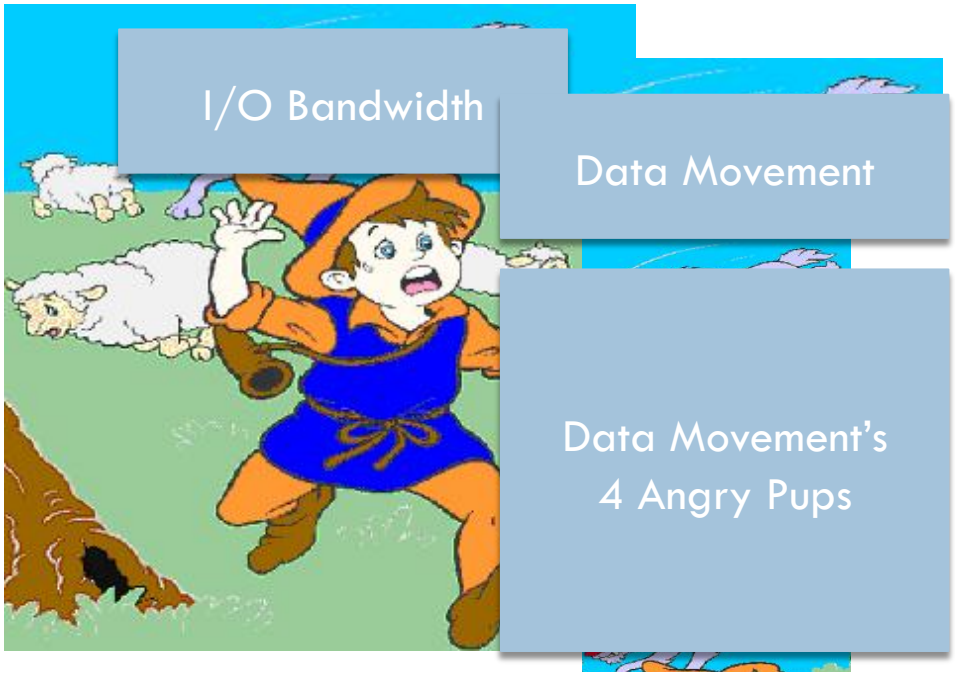
Fable: The Boy Who Cried Wolf

- Once there was a viz expert who had to look after customers. One day, he needed funding and decided to play a trick on his funders. He shouted, "Help! Big Big Data!" The funders heard his cries and sent lots of money to help the viz expert. When petascale arrived, they asked, "Where is the problem?" The viz expert shrugged and said, "The problem isn't quite here yet, but it will be soon."

This is NOT the story of this talk.

The message from this talk…

Petascale Visualization



Exascale Visualization

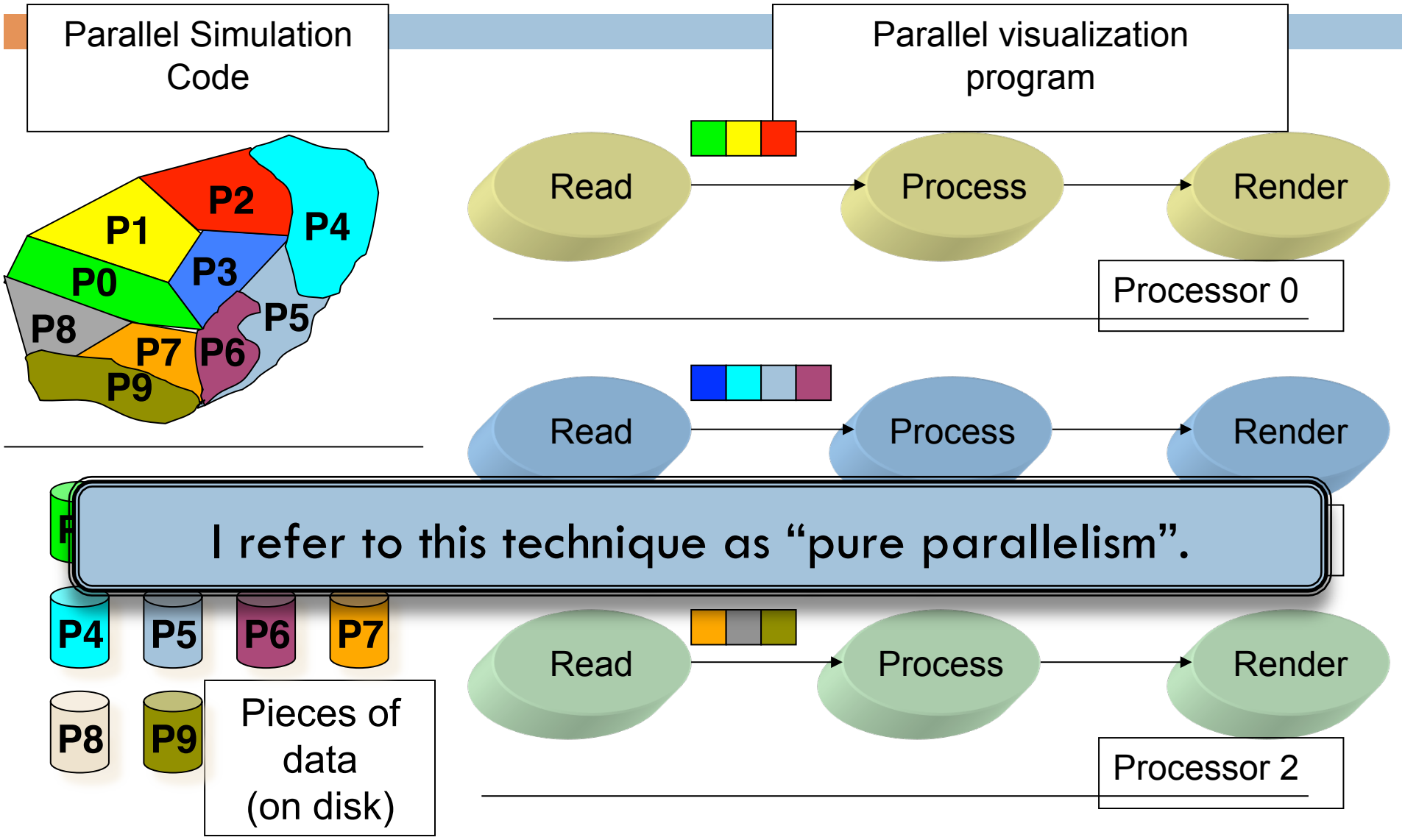
Outline

- The Terascale Strategy
- The I/O Wolf & Petascale Visualization
- An Overview of the Exascale Machine
- The Data Movement Wolf and Its 4 Angry Pups
- Under-represented topics
- Conclusions

Outline

- The Terascale Strategy
- The I/O Wolf & Petascale Visualization
- An Overview of the Exascale Machine
- The Data Movement Wolf and Its 4 Angry Pups
- Under-represented topics
- Conclusions

The terascale visualization strategy



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

Pure parallelism and today's tools

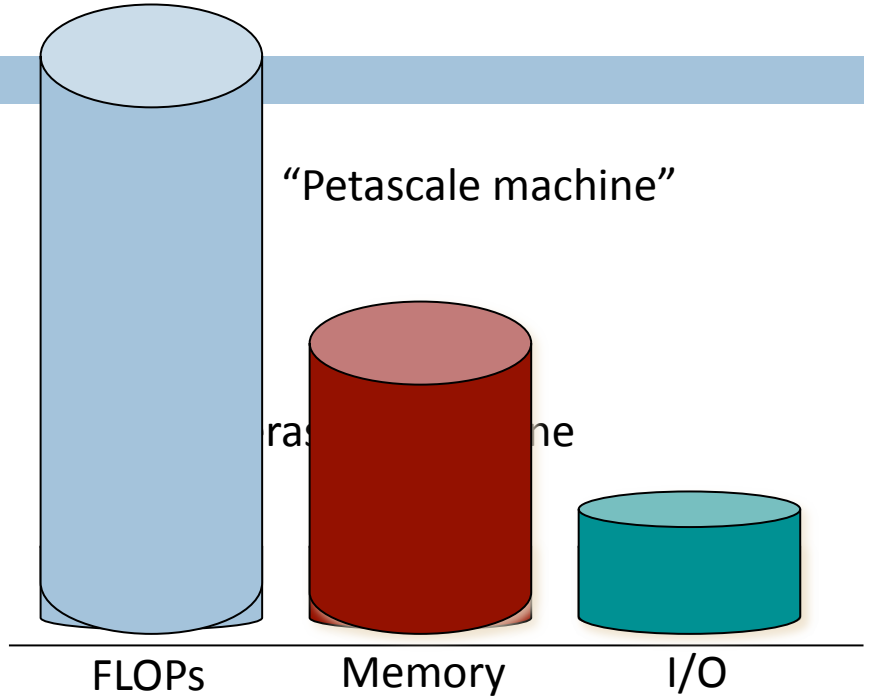
- VisIt, ParaView, & EnSight primarily employ a pure parallelism + client-server strategy.
  - All tools working on advanced techniques as well
- Of course, there's lots more technology out there besides those three tools…

Outline

- The Terascale Strategy
- The I/O Wolf & Petascale Visualization
- An Overview of the Exascale Machine
- The Data Movement Wolf and Its 4 Angry Pups
- Under-represented topics
- Conclusions

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

Trends in I/O

Machine	Year	Time to write memory
ASCI Red	1997	300 sec
ASCI Blue Pacific	1998	400 sec
ASCI White	2001	660 sec
ASCI Red Storm	2004	660 sec
ASCI Purple	2005	500 sec
Jaguar XT4	2007	1400 sec
Roadrunner	2008	1600 sec
Jaguar XT5	2008	1250 sec

c/o David Pugmire, ORNL

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.

How are we responding?

- Enable downstream tools to operate on smaller-sized representations: multi-resolution methods.
- Operate on data "in place:" in situ visualization and analysis.
- Load less of the data: data subsetting.

Multi-resolution techniques

- Idea: create multiple versions of a data set at different resolutions.
- Pros
  - Can drastically reduce I/O & memory requirements
  - (Sometimes better) confidence in pictures; multi-res hierarchy addresses "many cells to one pixel issue"
- Cons
  - Not always meaningful to process simplified version of the data.
  - How do we generate hierarchical representations during dump? What costs do they incur (data movement costs, storage costs)?

In situ

- In situ processing can mean multiple things
  - Idea: perform vis/analysis processing on data while it is still resident in memory.
  - Will discuss this more later in the talk
- Common perceptions of in situ
  - Pros:
    - No I/O & plenty of compute
  - Cons:
    - Operates in a memory constrained environment.
    - Some operations not possible
      - Once the simulation has advanced, you cannot go back and analyze it
    - User must know what to look for *a priori*.

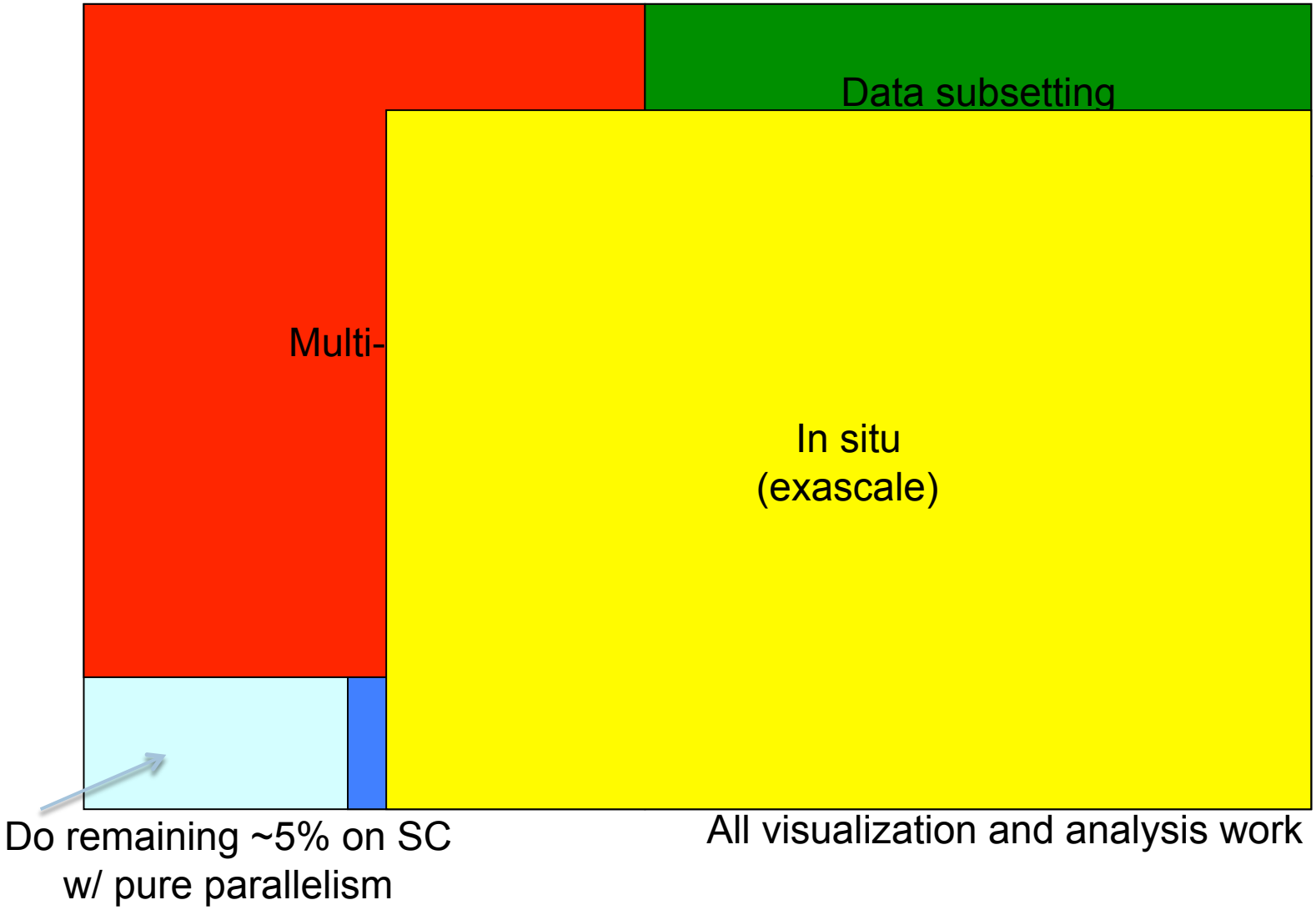
Data Subsetting

- Data Subsetting:
  - Idea: load and process only "interesting data."
  - Examples: reducing data processed via meta-data, query-driven visualization (forensic cybersecurity example, if we have time)
  - Pro: Less data to process (less I/O, less memory).
  - Con: Not applicable to all algorithms.

Out-of-core methods

- Out-of-core methods
  - Idea: rather than load the entire data set, load only smaller portions, or windows, and execute the vis/ analysis pipeline on that piece.
  - Pros:
    - Lower requirement for primary memory
    - Doesn't require big machines
  - Con:
    - Still paying large I/O costs (slow!)
    - Can result in I/O "multiplier" effect.

Assertion: we are going to need a lot of solutions.



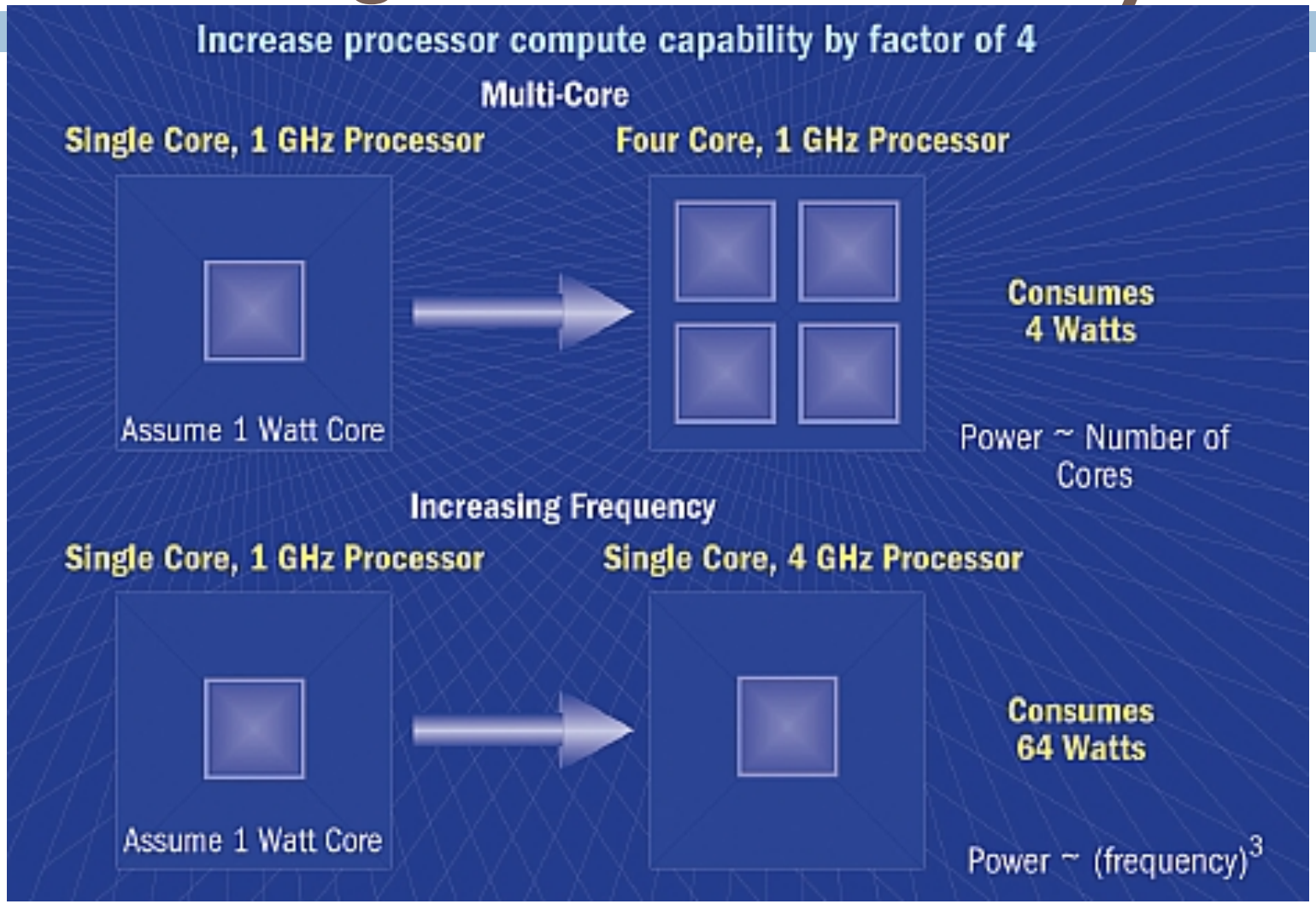
Outline

- The Terascale Strategy
- The I/O Wolf & Petascale Visualization
- An Overview of the Exascale Machine
- The Data Movement Wolf and Its 4 Angry Pups
- Under-represented topics
- Conclusions

Exascale assumptions

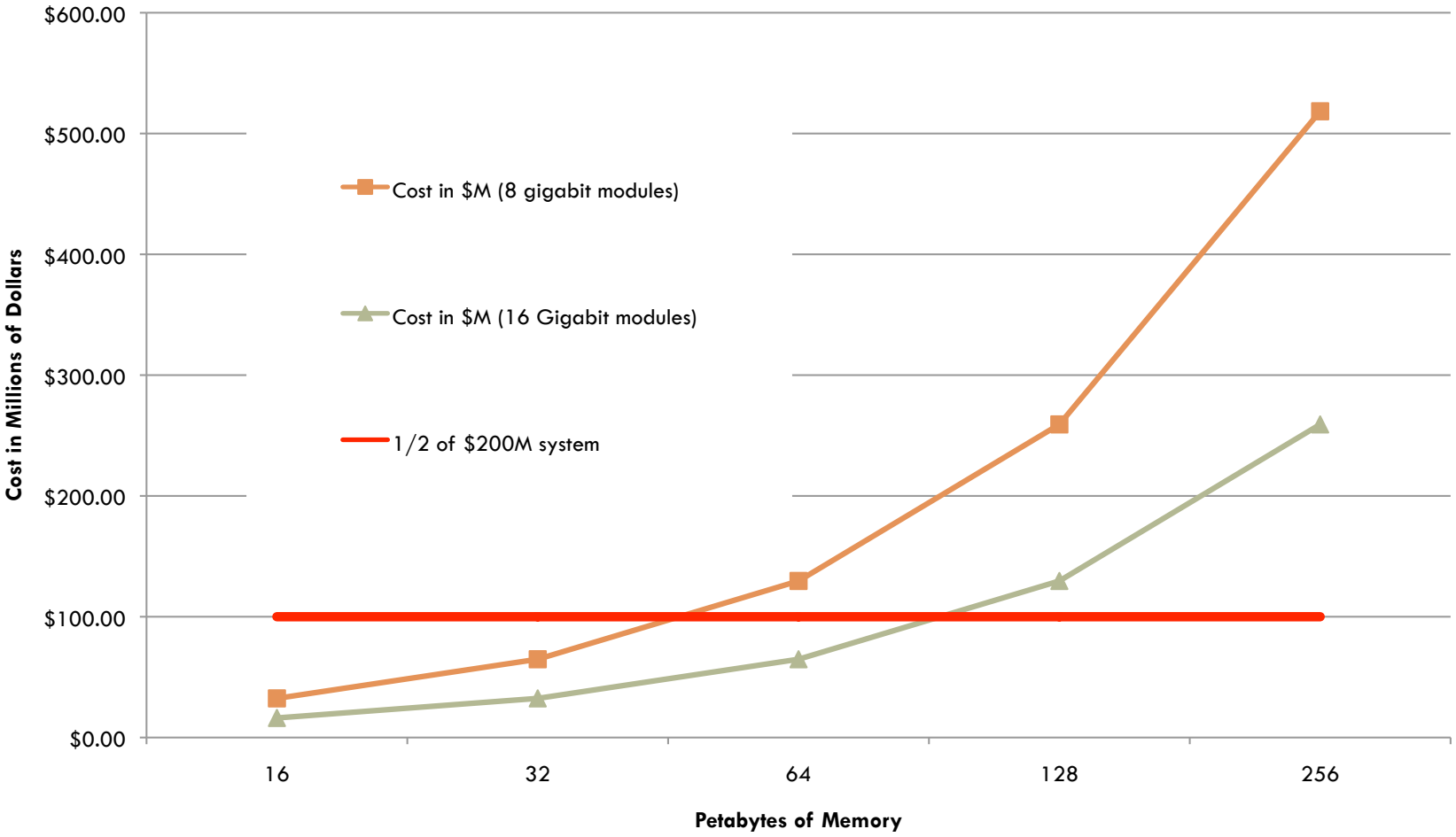
- The machine will be capable of one exaflop.
- The machine will cost < $200M.
- The machine will use < 20MW.
- The machine may arrive as early as 2018.

Hurdle #1: power requires slower clocks and greater concurrency



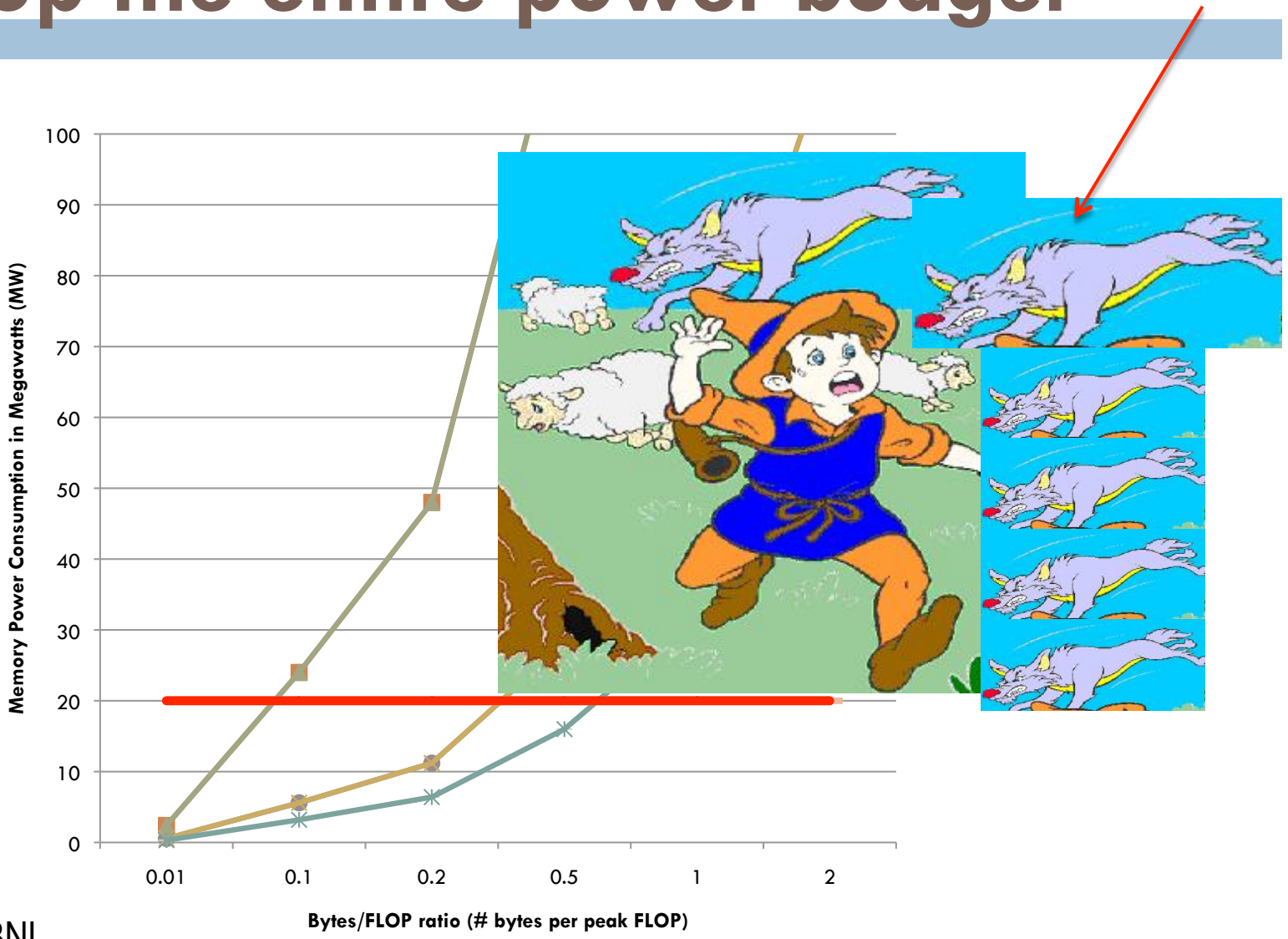
c/o *SciDAC Review* 16, February 2010

Hurdle #2: memory capacity eats up the entire fiscal budget



c/o John Shalf, LBNL

Hurdle #3: memory bandwidth eats up the entire power budget



c/o John Shalf, LBNL

Summarizing exascale visualization

- Power is king.
- Costly to move data off the machine.
  - And we can't read it in if we do get it off.
- Costly to even move it around the machine.

- → Beneficial to process the data in situ.

Outline

- The Terascale Strategy
- The I/O Wolf & Petascale Visualization
- An Overview of the Exascale Machine
- The Data Movement Wolf and Its 4 Angry Pups
  - Pup #1: In Situ Systems Research
- Under-represented topics
- Conclusions

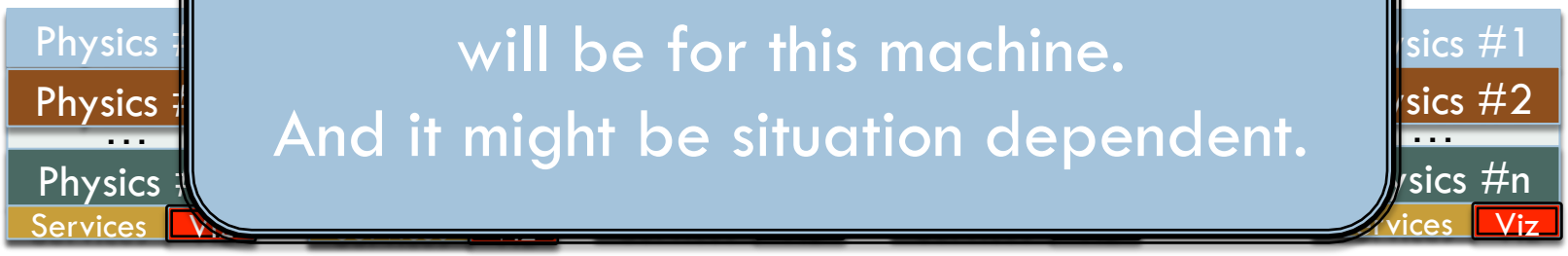
Summarizing flavors of in situ

In Situ Technique	Aliases	Description	Negative Aspects
Tightly coupled	Synchronous, co-processing	Visualization and analysis have direct access to memory of simulation code	1) Very memory constrained 2) Large potential impact (performance, crashes)
Loosely coupled	Asynchronous, concurrent	Visualization and analysis run on concurrent resources and access data over network	1) Data movement costs 2) Requires separate resources
Hybrid		Data is reduced in a tightly coupled setting and sent to a concurrent resource	1) Complex 2) Shares negative aspects (to a lesser extent) of others

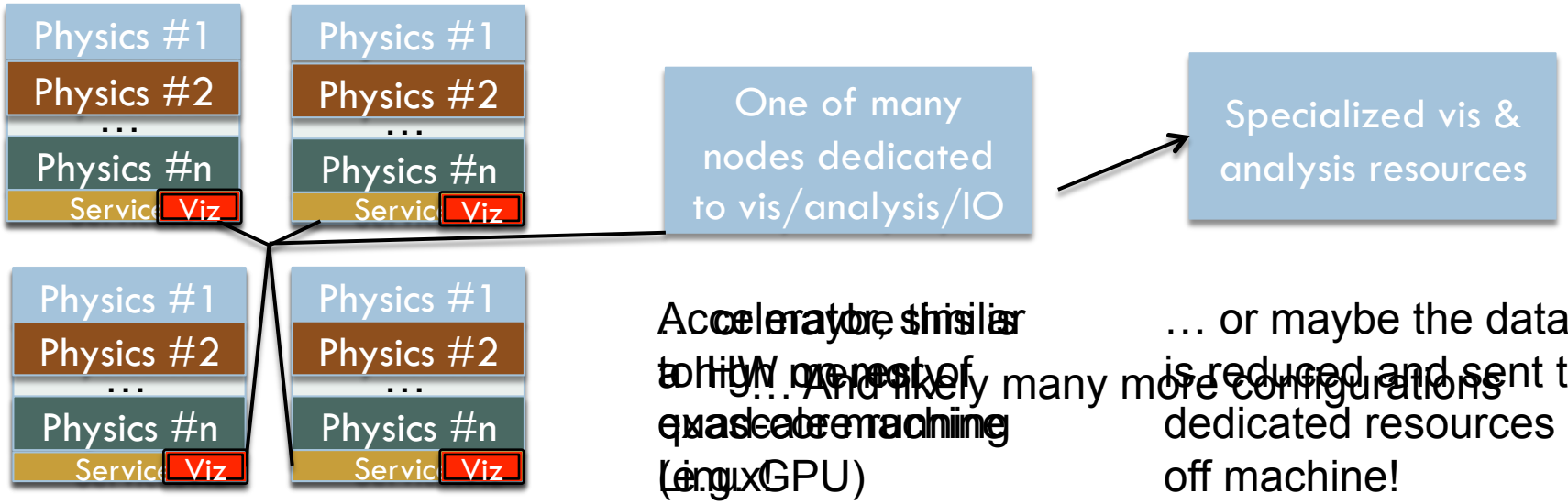
Possible in situ visualization scenarios

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)



… and likely many more configurations

Outline

- The Terascale Strategy
- The I/O Wolf & Petascale Visualization
- An Overview of the Exascale Machine
- The Data Movement Wolf and Its 4 Angry Pups
  - Pup #2: Programming Languages
- Under-represented topics
- Conclusions

Angry Pup #2: Programming Language

- VTK: enables the community to develop diverse algorithms for diverse execution models for diverse data models
  - Impor
  - Subst
- We nee
  - Will also be a substantial investment
- Must be:
  - Lightweight
  - Efficient
  - Able to run in a many core environment

OK, what language is this in?
OpenCL? DSL?
… not even clear how to start

Message-passing remains important at the exascale, but we lose its universality

MPI will be combined with other paradigms within a shared memory node (OpenMP, OpenCL, CUDA, etc.)

Codes will not be hardware-universal again, until a lengthy evolutionary period passes

c/o David Keyes, KAUST

Visualization and Hybrid Parallelism

State of Parallelism in Scientific Computing

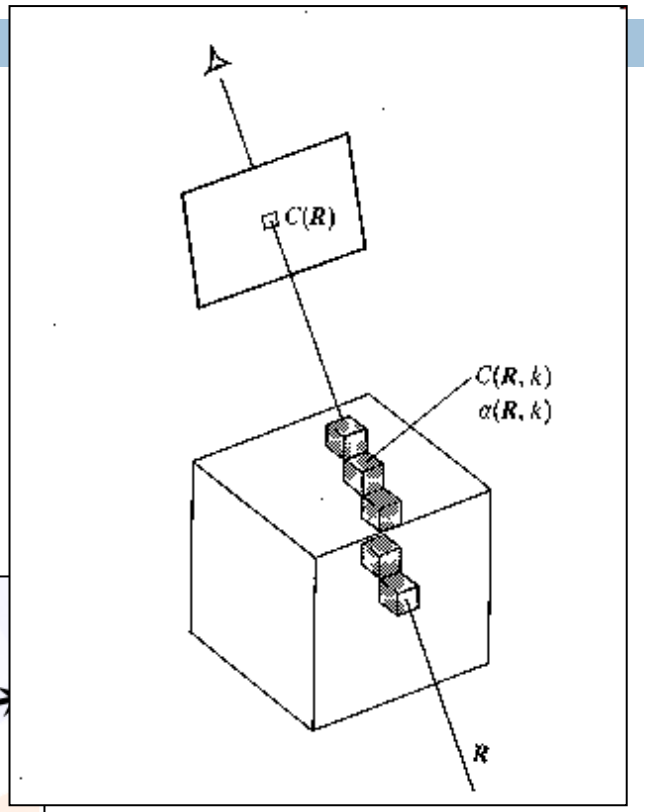
- Most production codes written using MPI, vendor MPI implementations optimized for their architecture.
- HPC community wondering how well MPI will scale to high concurrency, particularly on 100-core CPUs.
- What to do?
  - Some alternatives: data parallel languages (CUDA), PGAS languages (UPC), global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

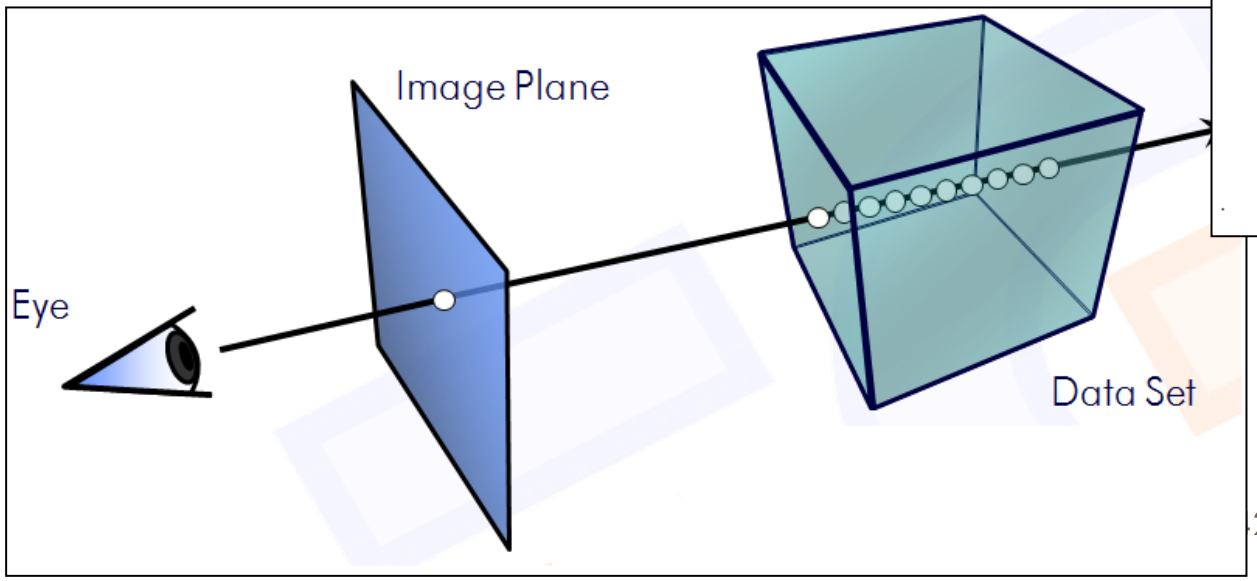
This Study

- 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.

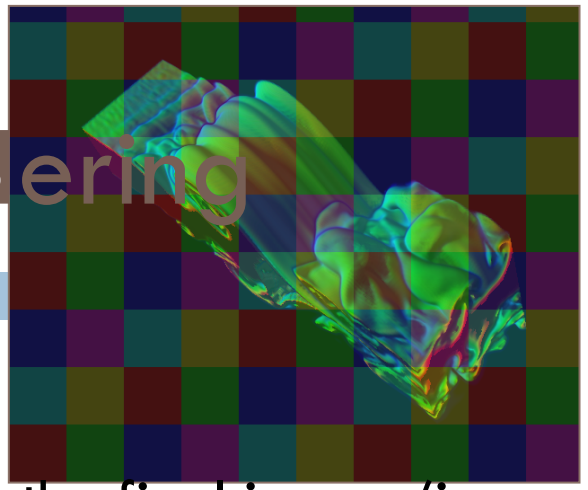
Algorithm Studied: Raycasting VR

- 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

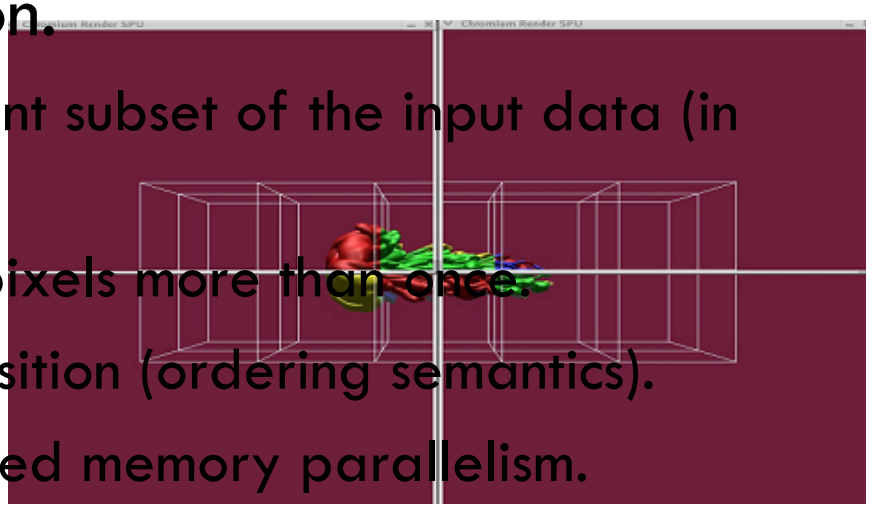




Parallelizing Volume Rendering



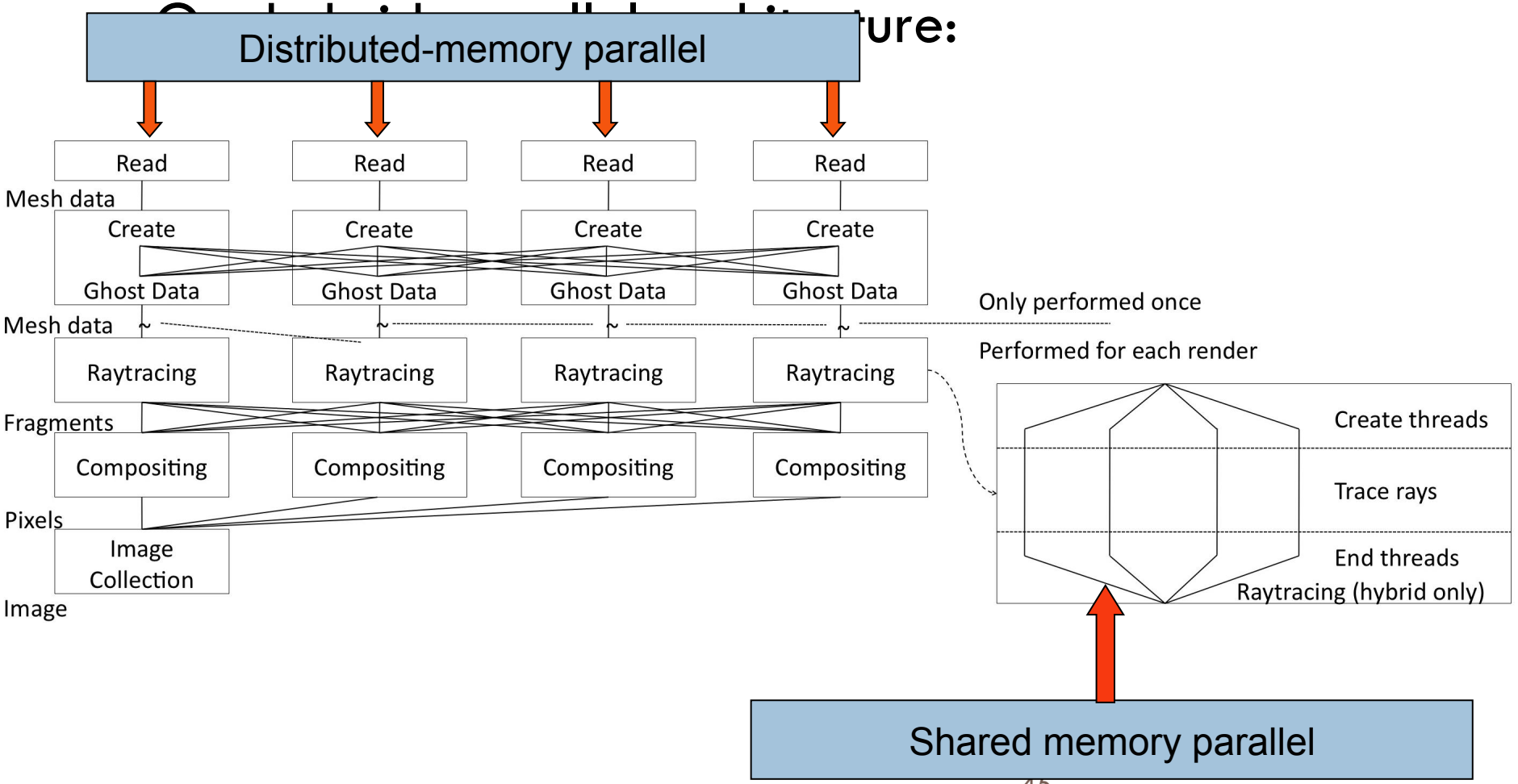
- Image-space decomposition.
  - Each process works on a disjoint subset of the final image (in parallel)
  - Processes may access source voxels more than once, will access a given output pixel only once.
  - Great for shared memory parallelism.
- Object-space decomposition.
  - Each process works on 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.



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

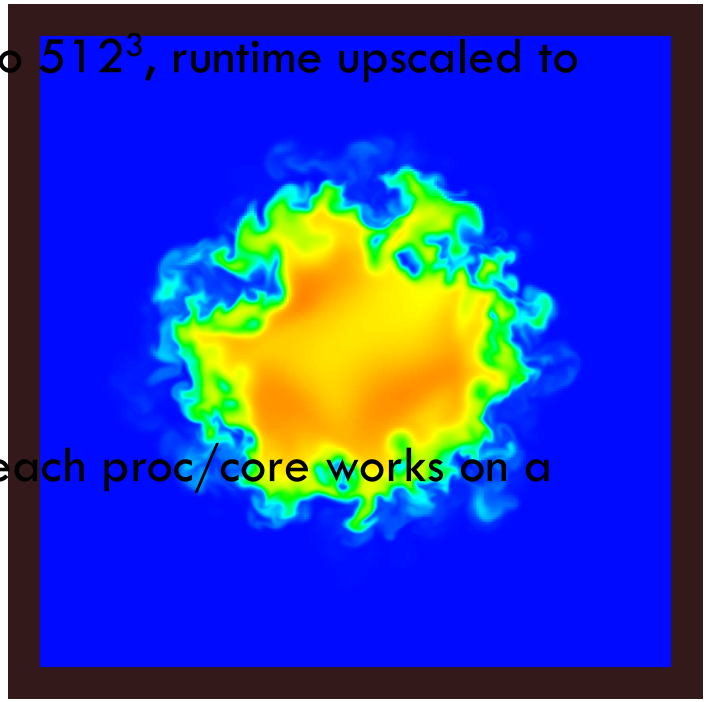


Our Experiment

- 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.



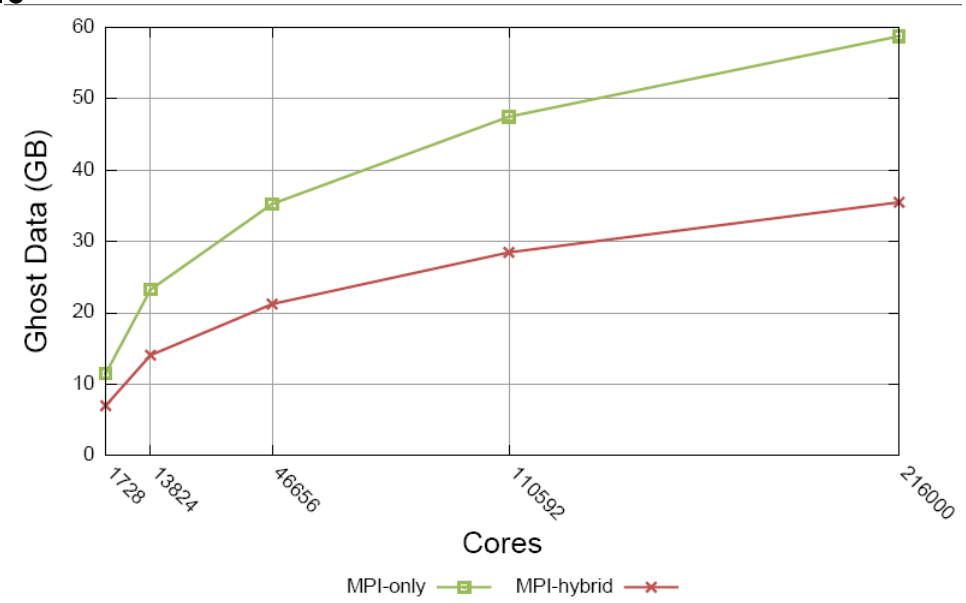
Memory Use – MPI_Init()

- 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

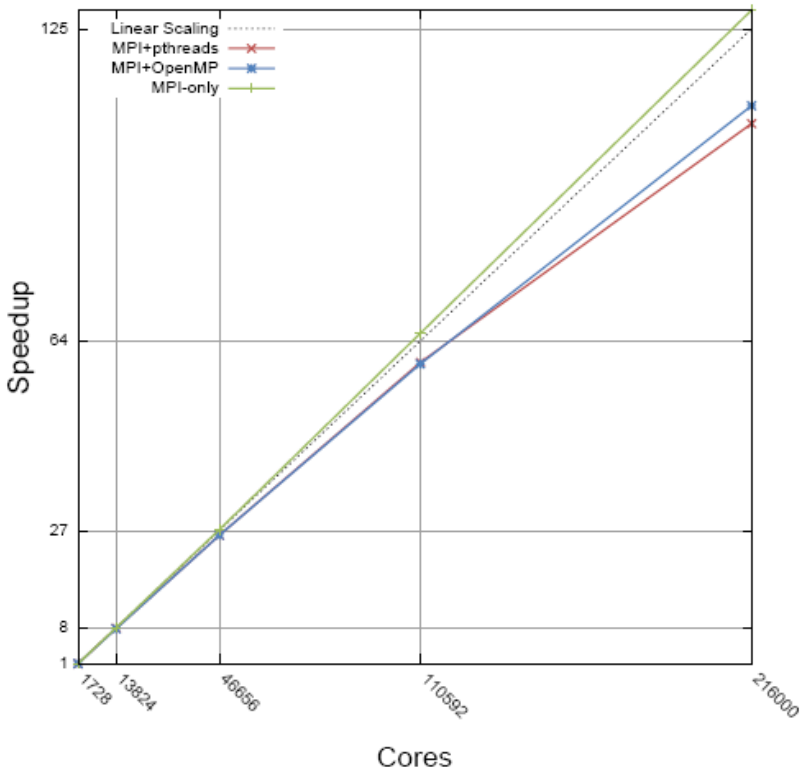
Memory Use – Ghost Zones

- 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



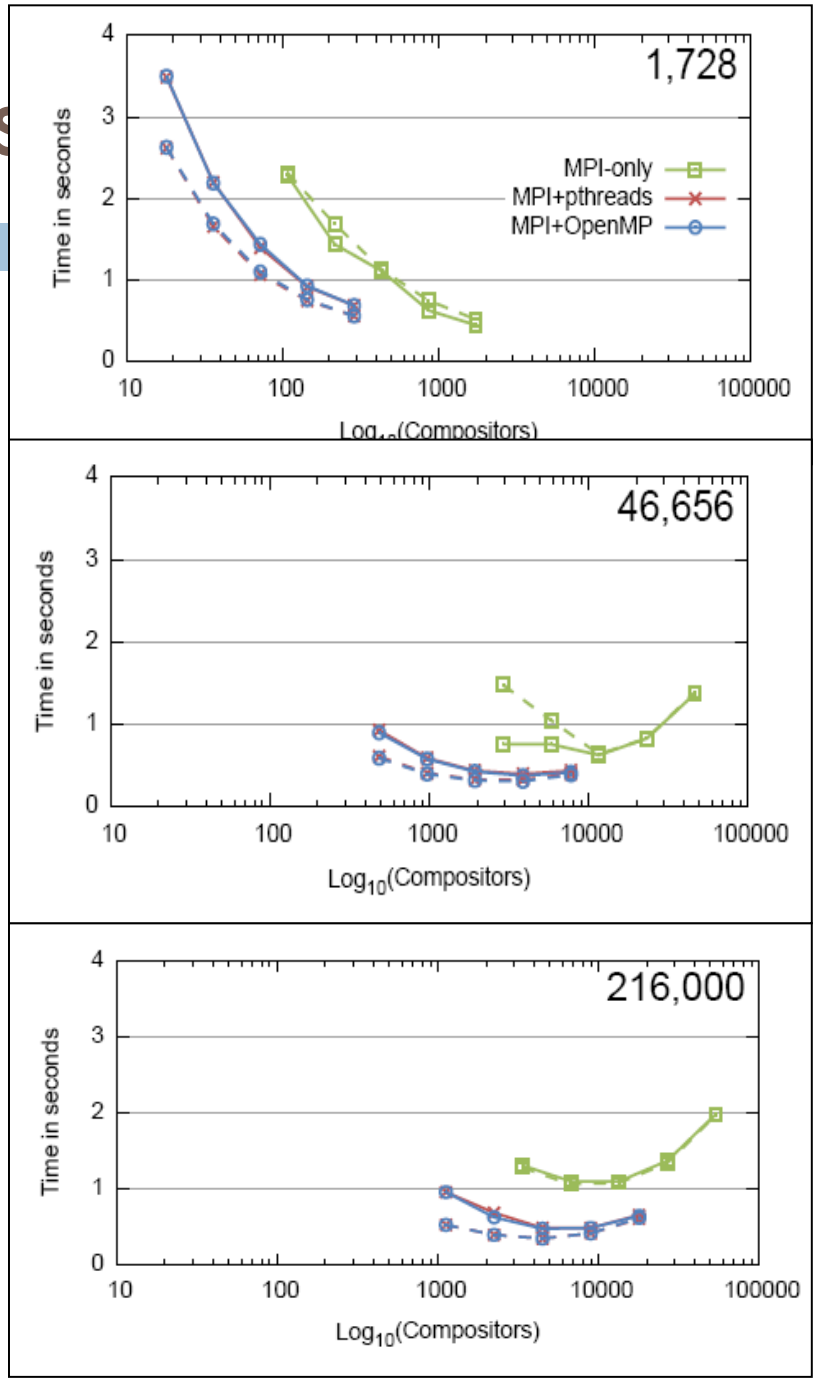
Scalability – Raycasting Phase

- 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.



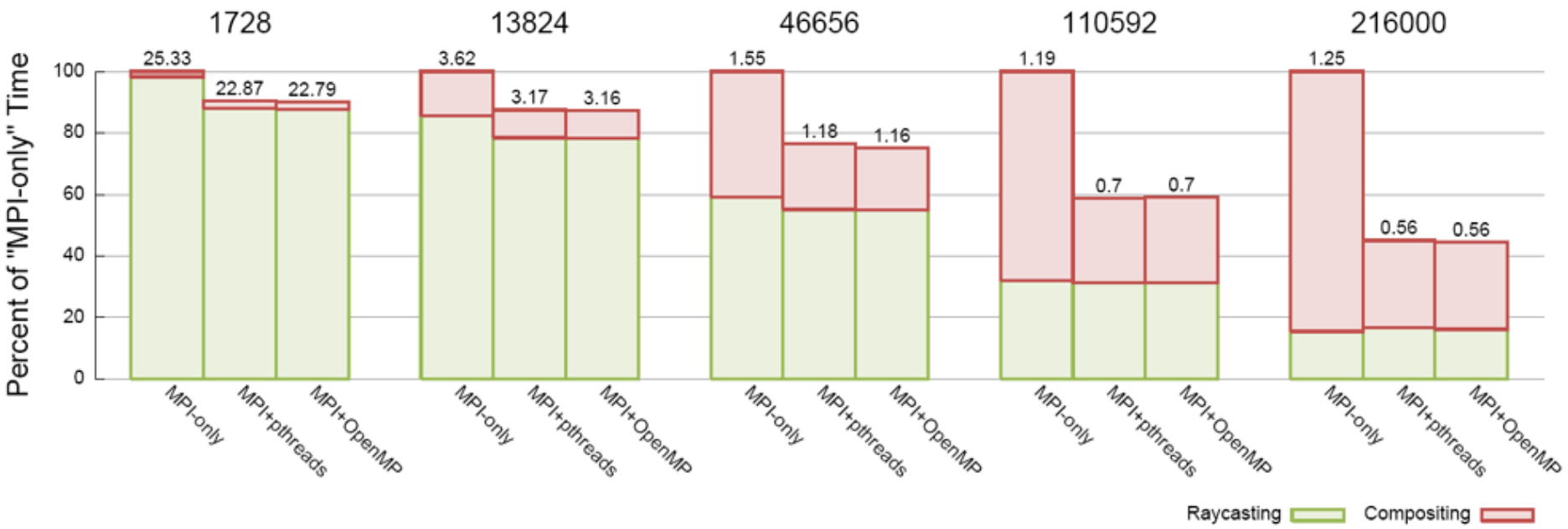
Scalability – Compos

- 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?



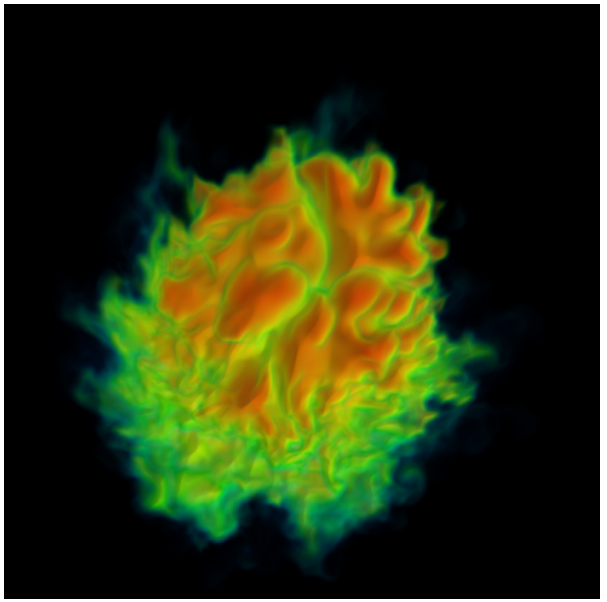
Absolute Runtime

- -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).




DETOUR

Outline

- The Terascale Strategy
- The I/O Wolf & Petascale Visualization
- An Overview of the Exascale Machine
- The Data Movement Wolf and Its 4 Angry Pups
  - Pup #3: Memory Footprint
- Under-represented topics
- Conclusions

Memory efficiency

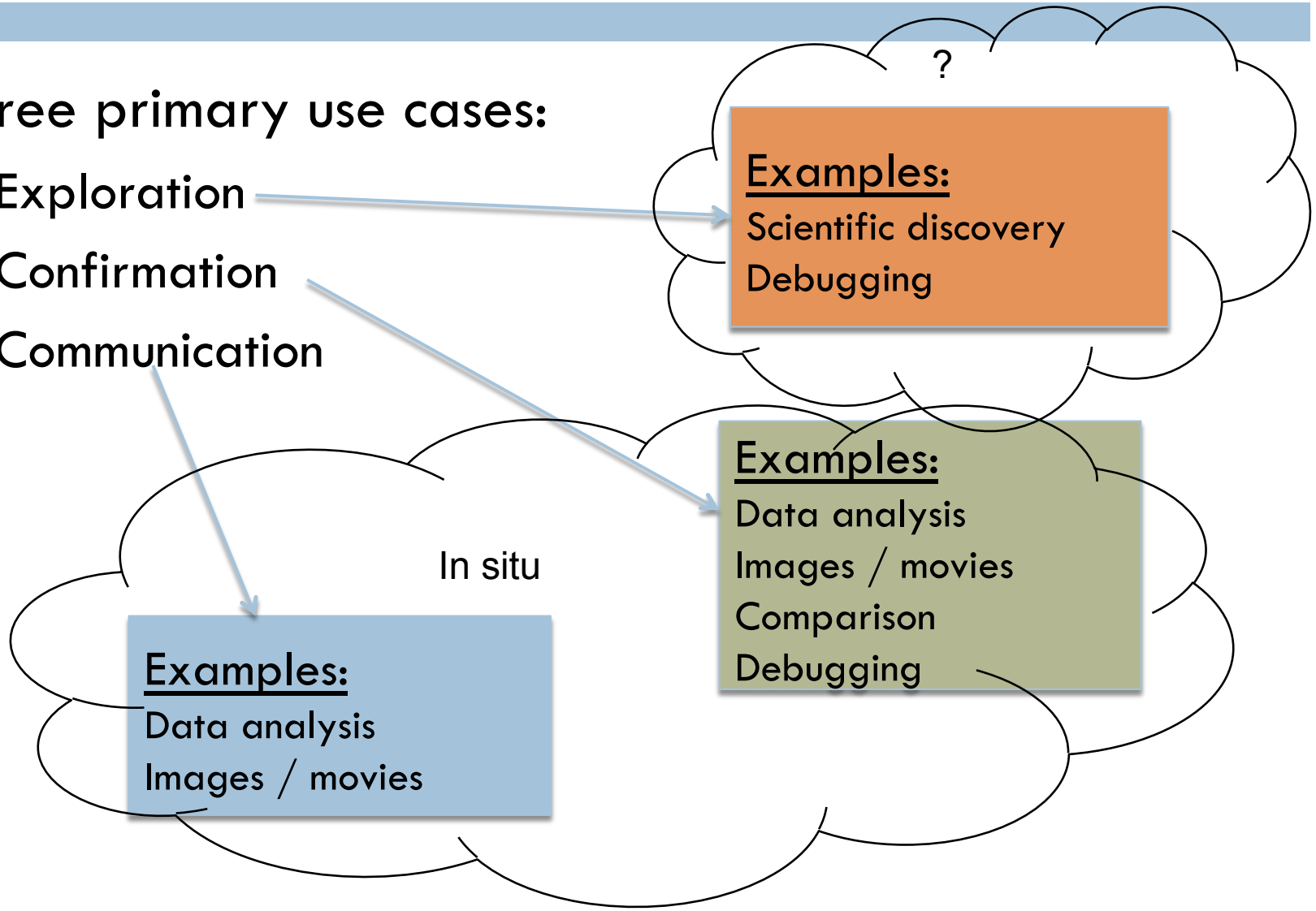
- 64 PB of memory for 1 billion cores means 64MB per core
  - (May be 10 billion cores and 6.4MB per core)
- Memory will be the 2nd most precious resource on the machine.
  - There won't be a lot left over for visualization and analysis.
- Zero copy in situ is an obvious start
  - Templates? Virtual functions?
- Ensure fixed limits for memory footprints (Streaming?)

Outline

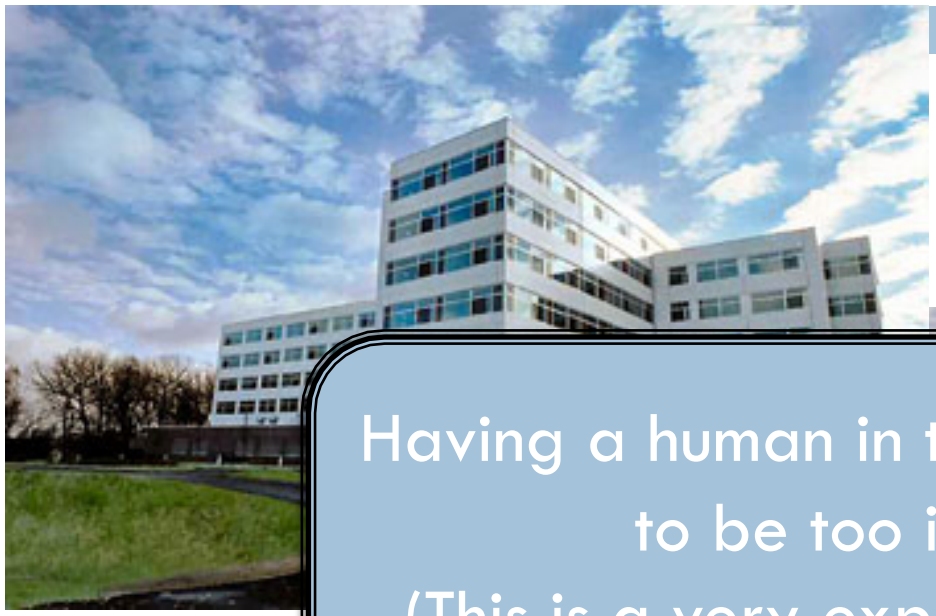
- The Terascale Strategy
- The I/O Wolf & Petascale Visualization
- An Overview of the Exascale Machine
- The Data Movement Wolf and Its 4 Angry Pups
  - Pup #4: In Situ-Fueled Exploration
- Under-represented topics
- Conclusions

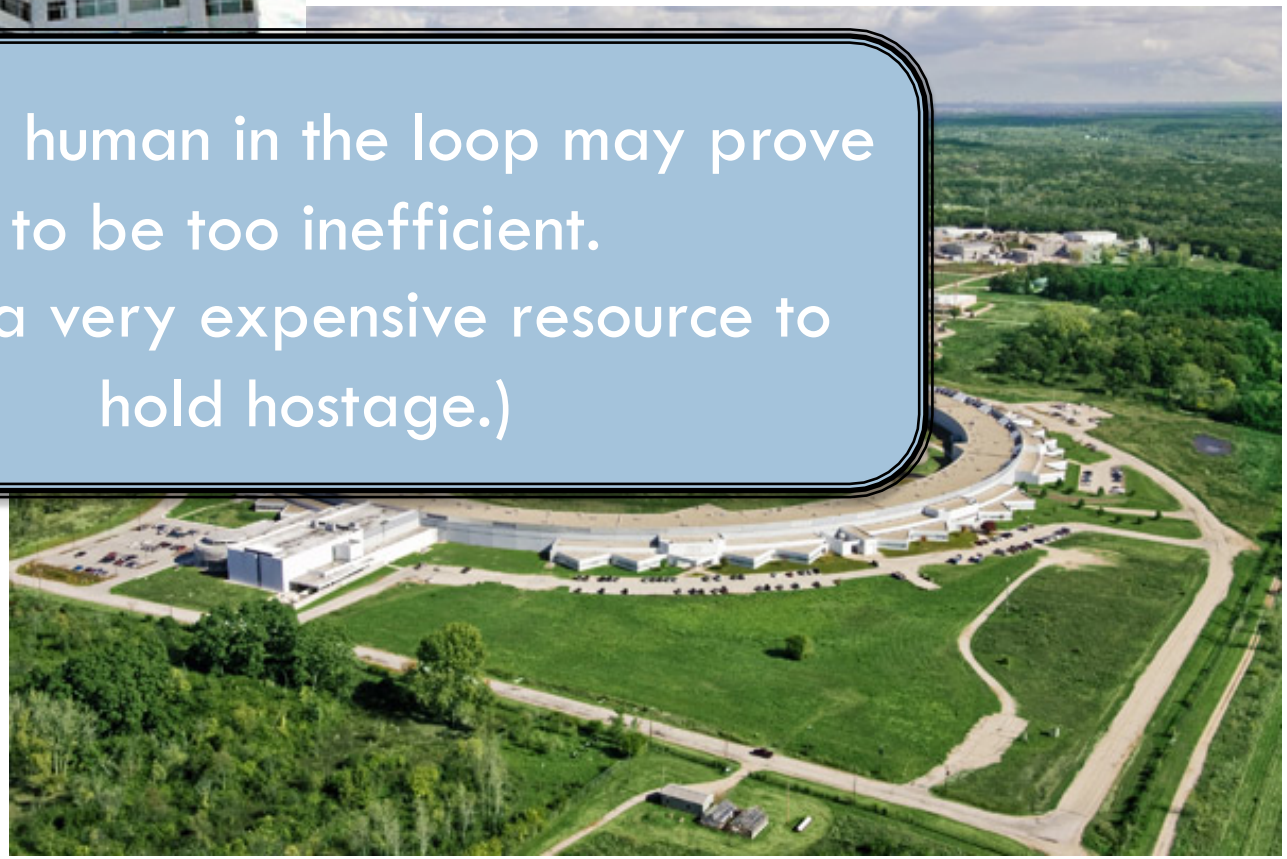
Do we have our use cases covered?

- Three primary use cases:
  - Exploration
  - Confirmation
  - Communication



Can we do exploration in situ?





Having a human in the loop may prove to be too inefficient.
(This is a very expensive resource to hold hostage.)

Enabling exploration via in situ processing

- Requirement: must transform the data in a way that both reduces and enables meaningful exploration.
- Subsetting
  - B… …lization
- Mu…
  - Old model: user looks at coarse data, but can dive down to original data.
  - New model: branches of the multi-res tree are pruned if they are very similar. (compression!)

It is not clear what the best way is to use in situ processing to enable exploration with post-processing … it is only clear that we need to do it.

Outline

- The Terascale Strategy
- The I/O Wolf & Petascale Visualization
- An Overview of the Exascale Machine
- The Data Movement Wolf and Its 4 Angry Pups
- Under-represented topics
- Conclusions

Under-represented topics in this talk.

- We will have quintillions of data points … how do we meaningfully represent that with millions of pixels?
- Data is going to be different at the exascale: ensembles, multi-physics, etc.
  - The outputs of visualization software will be different.
- Accelerators on exascale machine are likely not to have cache coherency
  - How well do our algorithms work in a GPU-type setting?
  - We have a huge investment in CPU-SW. What now?
- What do we have to do to support resiliency issue?

Outline

- The Terascale Strategy
- The I/O Wolf & Petascale Visualization
- An Overview of the Exascale Machine
- The Data Movement Wolf and Its 4 Angry Pups
- Under-represented topics
- Conclusions

It is funny how this happens…

- All processing techniques still are very relevant.
  - In situ: data movement wolf
  - Out-of-core: Pup #3: memory efficiency
  - Multi-res: Pup #4: exploration
  - Data subsetting: Pup #4: exploration
  - Pure parallelism: experiences at massive concurrency will be critical

Summary

- We are unusual: we are data consumers, not data producers, and the exascale machine is being designed for data producers
- So the exascale machine will almost certainly lead to a paradigm shift in the way visualization programs process data.
  - Where to process data and what data to move will be a central issue.

Summary

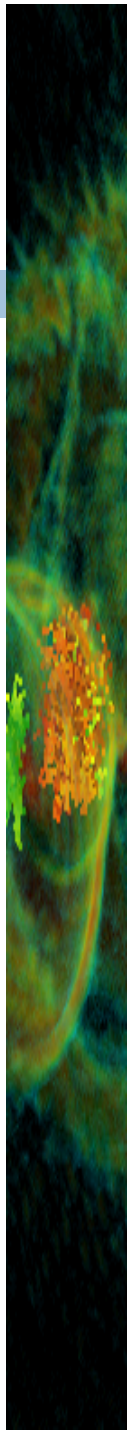
- In addition to the I/O "wolf", we will now have to deal with a data movement "wolf", plus its 4 pups:
  1) In Situ System
  2) Programming Language
  3) Memory Efficiency
  4) In Situ-Fueled Exploration

Acknowledgments

- Many of these slides courtesy Hank Childs (LBNL) who gave an Exascale Visualization keynote at EGPGV 2011 in May 2011.
- This work was supported by the Director, Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

QDV – Detecting Distributed Scans

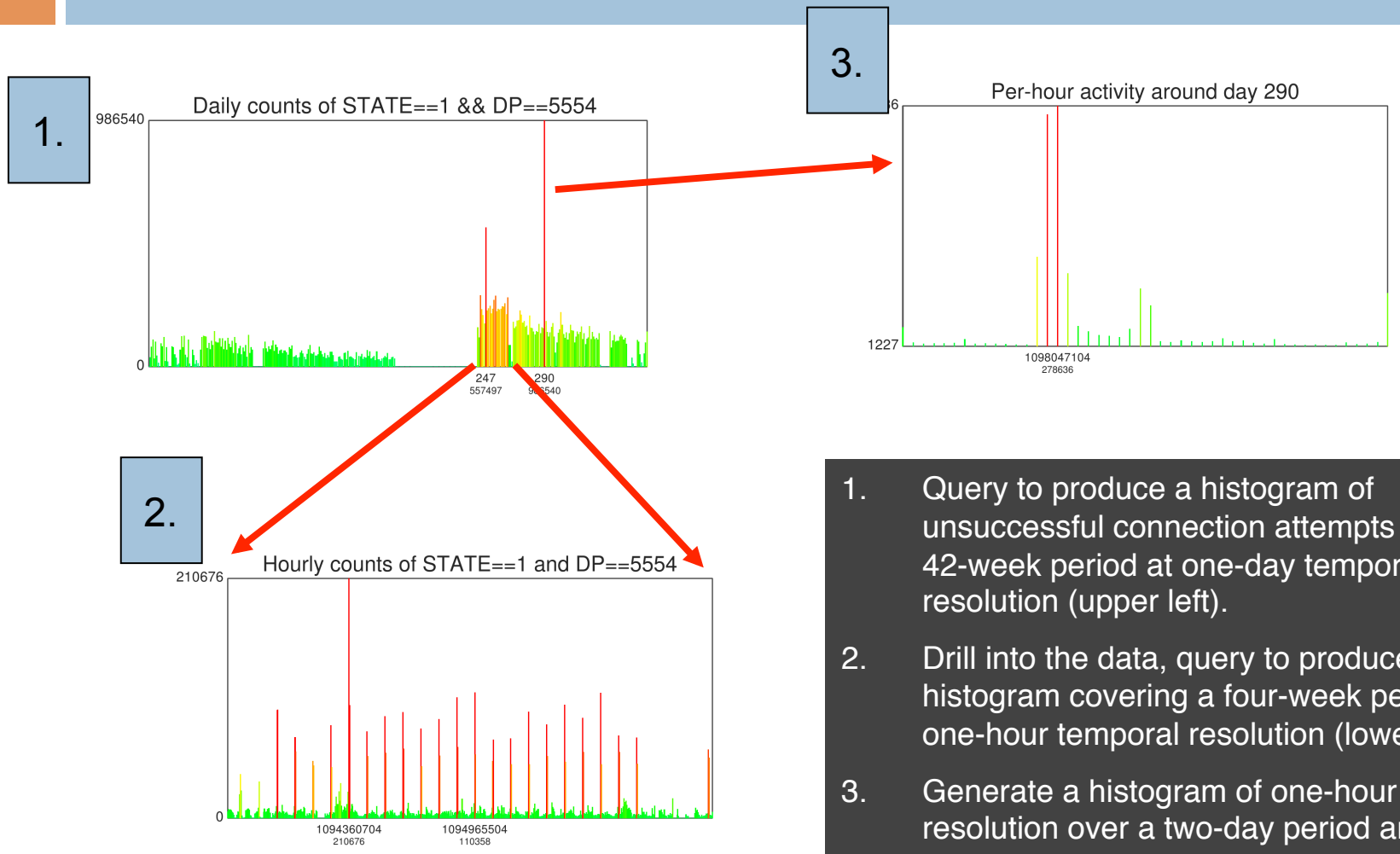
- The data:
  - 42 weeks' of connection records from Bro (NERSC).
  - 281GB for raw data, 78GB for compressed bitmap indices.
- "Hero-sized problem"
  - No previous network analysis work has ever attempted to perform interactive visual analytics on data of this scale (ca. 2006).
  - Result: what once took days (if at all possible) now takes seconds.



QDV – Detecting Distributed Scans

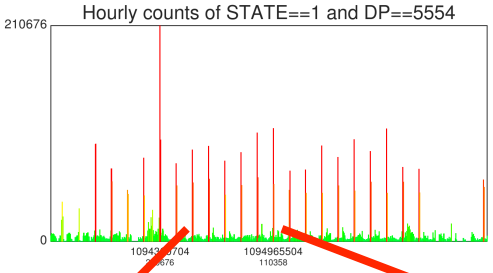
- The starting point:
  - You are a network security analyst
  - Your beeper goes off [at lunch, in the shower…]
  - You receive an alert that "something odd is happening with the network…IDS showing unusual levels of activity on port 5554"
  - Your job – answer questions:
    - What's going on now?
    - How long has this been happening?
    - Implications?

QDV – Detecting Distributed Scans

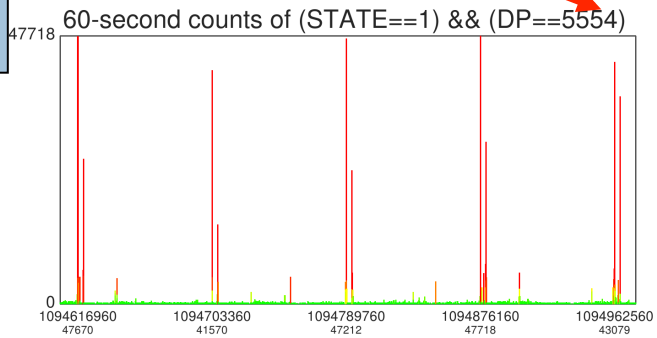


1. Query to produce a histogram of unsuccessful connection attempts over a 42-week period at one-day temporal resolution (upper left).
2. Drill into the data, query to produce a new histogram covering a four-week period at one-hour temporal resolution (lower left).
3. Generate a histogram of one-hour resolution over a two-day period around day 290 (upper right).

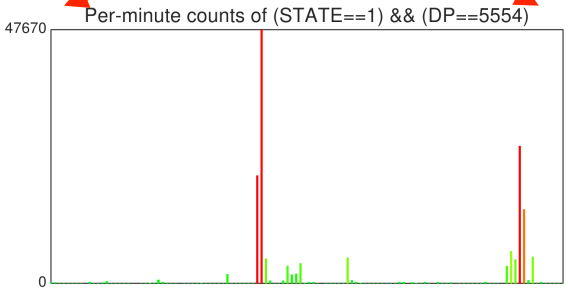
QDV – Detecting Distributed Scans



5.



6.

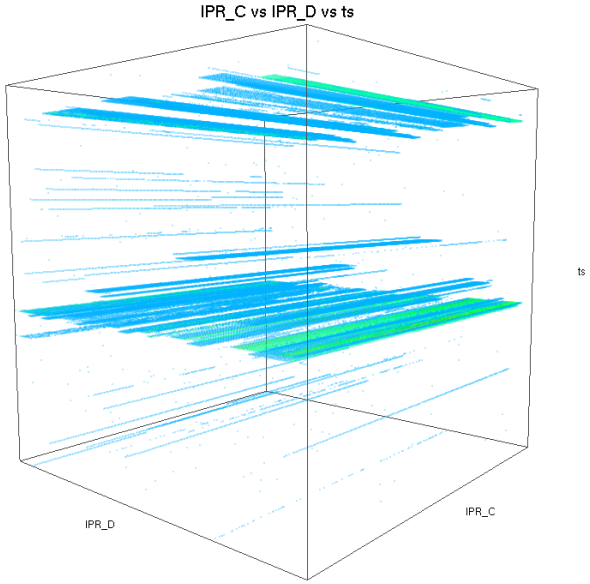


5. Query to generate a histogram of unsuccessful connection attempts over a five-day period sampled at one-minute temporal resolution (middle, left). Regular attacks occur at 21:15L, followed by a second wave 50 minutes later.

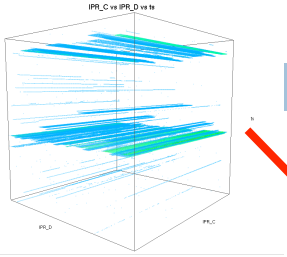
6. Query to generate histogram over a two-hour period at one-minute temporal resolution (lower left).

7. Query to generate a 3D histogram showing the coverage of attacks in destination address space (lower right).

7.



QDV – Detecting Distributed Scans

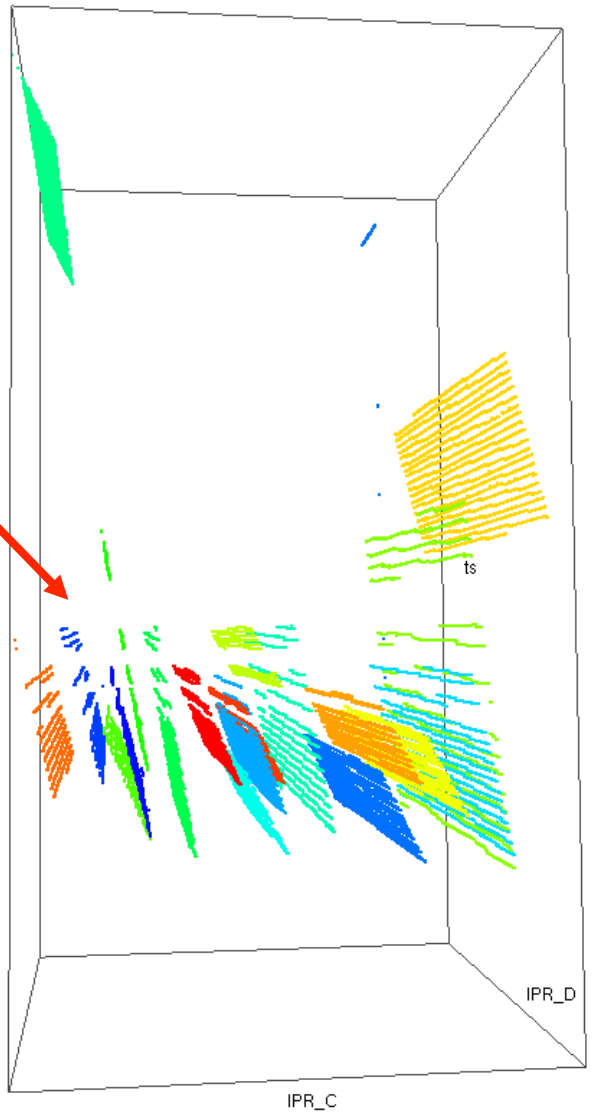


8.

After establishing that (1) a temporally regular activity is occurring, and (2) that it is in fact a systematic probe (scan) of entire blocks of network addresses, the next task is to determine the set of remote hosts participating in the attack.

Working backwards, we isolate the A, B, C and D address octets of the hosts participating in the attack.

8. This image shows a 3D histogram of the destination address space being attacked by each of 20 different hosts. The vertical axis is time – a seven-minute window at one-second temporal resolution.

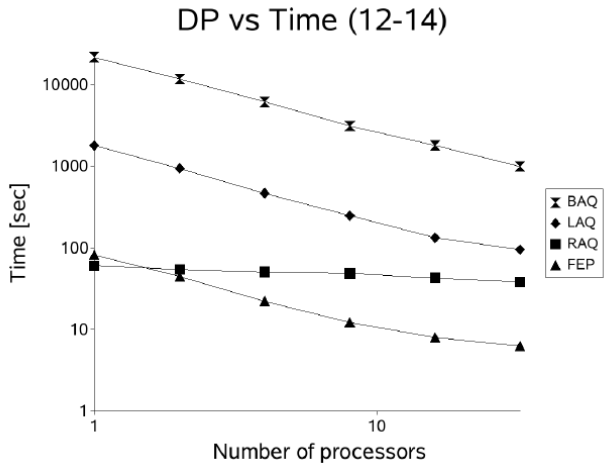


Performance Experiment

- How fast?
  - 3 to 6 orders of magnitude faster than shell-based tools.
  - 2-3 orders of magnitude faster than ROOT, the "gold standard" for search/ analysis in the HEP community.
  - Shows favorable scaling characteristics up to 32P on an SMP.

1D Histogram, per-bin queries

PEs	Shell-based	ROOT/Projection Index	ROOT/FastBit
1	156381.14	1357.07	5.36
2	71835.32	600.05	3.72
6	21952.12	214.14	2.66
13	9389.96	113.88	2.58
21	2237.53	98.95	2.05



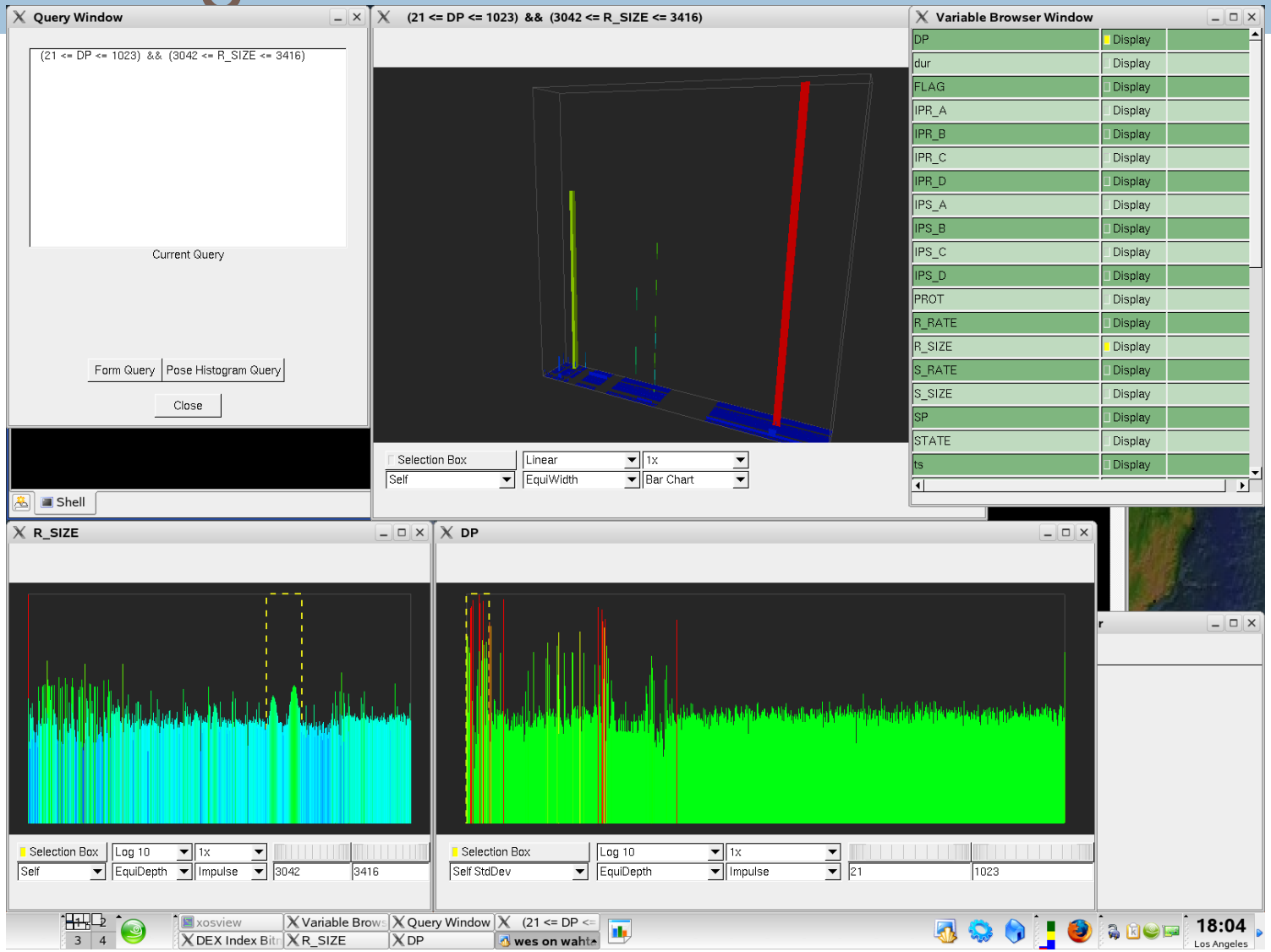
Conditional 2D histogram processing time

Query: (1000 < DP < 11000) AND (50 <= tsyday <= 350) AND (state==1) AND (12 <= tshour <=14), 10K total bins.

QDV – Detecting Distributed Scans

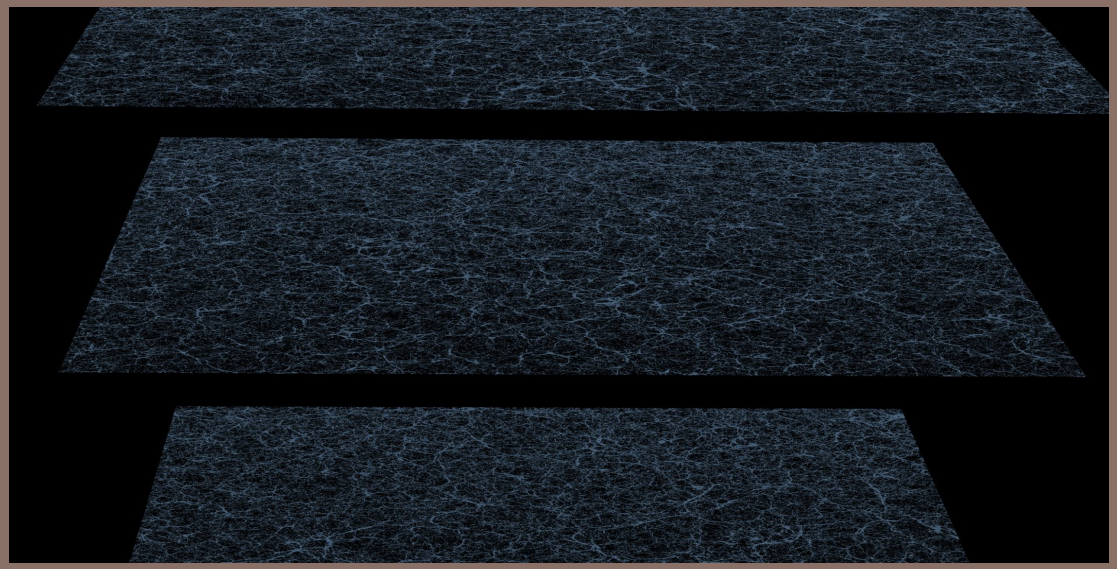
- Our analysis was performed in statistical space only.
  - We never accessed the raw data.
  - Our processing and visualization used only the index data.
  - Performance study focuses on parallel algorithms for multidimensional, conditional histogram computation from compressed bitmap indices.

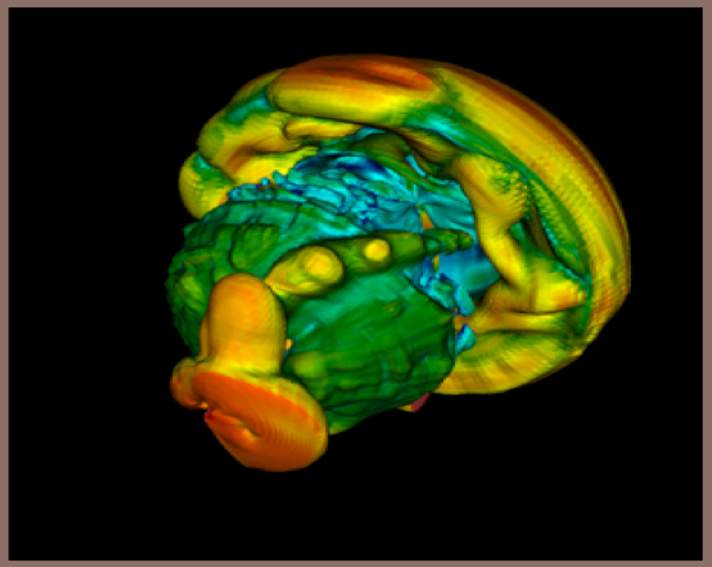
Visual Interface of the Application – Histograms!



EGPGV '11
Eurographics Symposium on Parallel Graphics and Visualization '11
Eurographics Symposium on
Parallel Graphics and Visualization

EXASCALE VISUALIZATION: GET READY FOR A WHOLE NEW WORLD



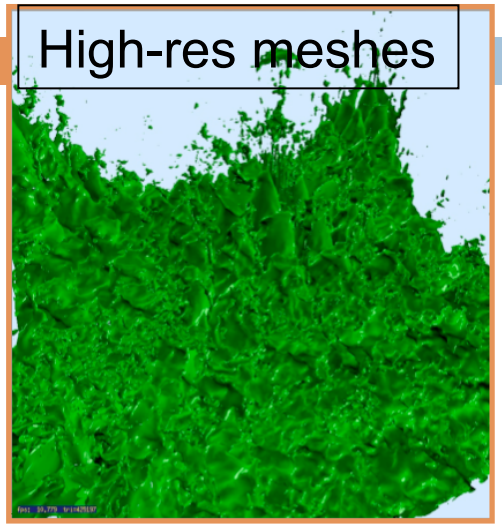


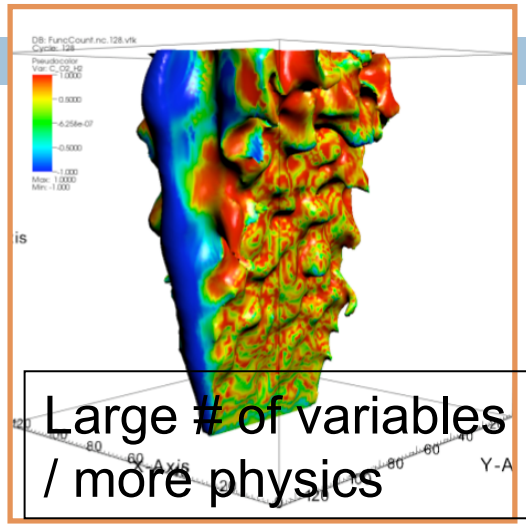
April 10, 2011

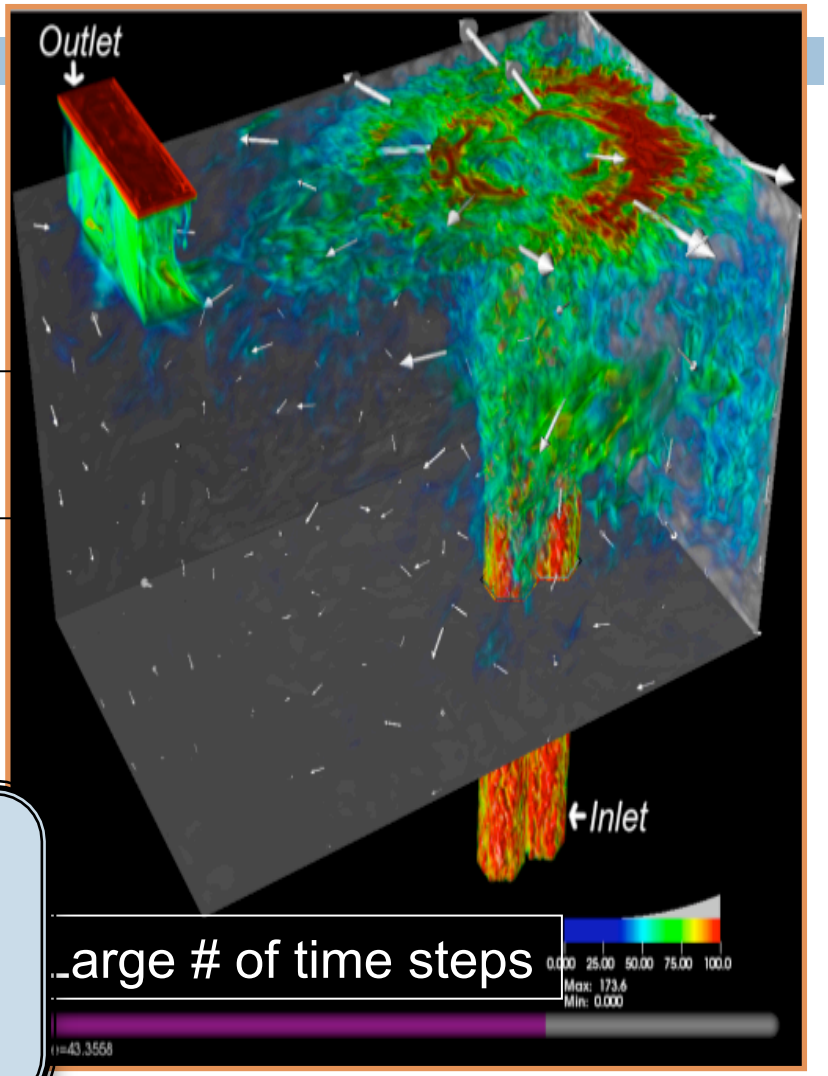
Hank Childs, Lawrence Berkeley Lab & UC Davis

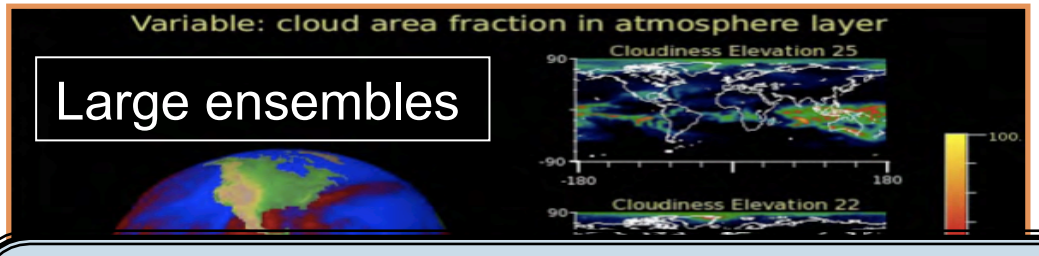
Backup slides

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









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

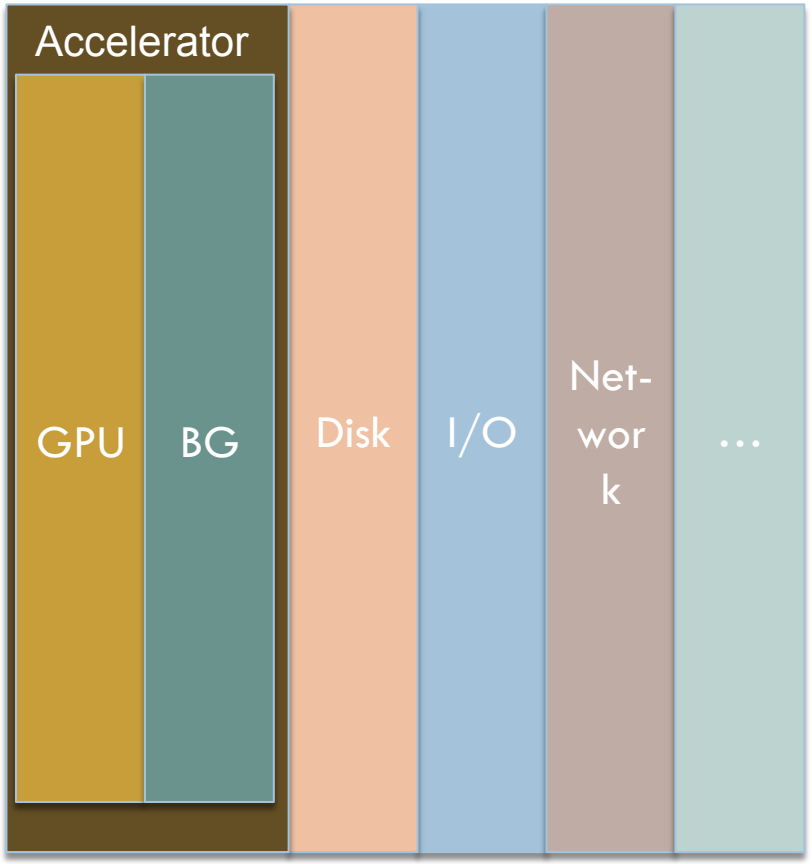
Thanks!: Sean Ahern & Ken Joy

Vis clusters

- Today's vis clusters are designed to be proportional to the "big iron".
  - Typical rule of thumb is that the little iron should have 10% of the memory of the big iron.
  - (Most vis clusters have fallen short of the 10% rule for the last few years.)
- Exascale machine = $200M, ~1/3rd of which is for memory. Memory for proposed vis cluster = $6.7M.
- I personally view it as unlikely that we will have 10%-sized vis clusters.
  - Additional issue with getting data off machine.

Accelerator technologies

- Currently simultaneously thinking about two different accelerator technologies:
  - IBM BlueGene's successor – some architectural merger of BlueGene, Power, and Cell
  - GPU / GPU evolution
- Referred to as "swim lanes": a visual element used in process flow diagrams, or flowcharts, that visually distinguishes responsibilities for sub-processes of a business process.

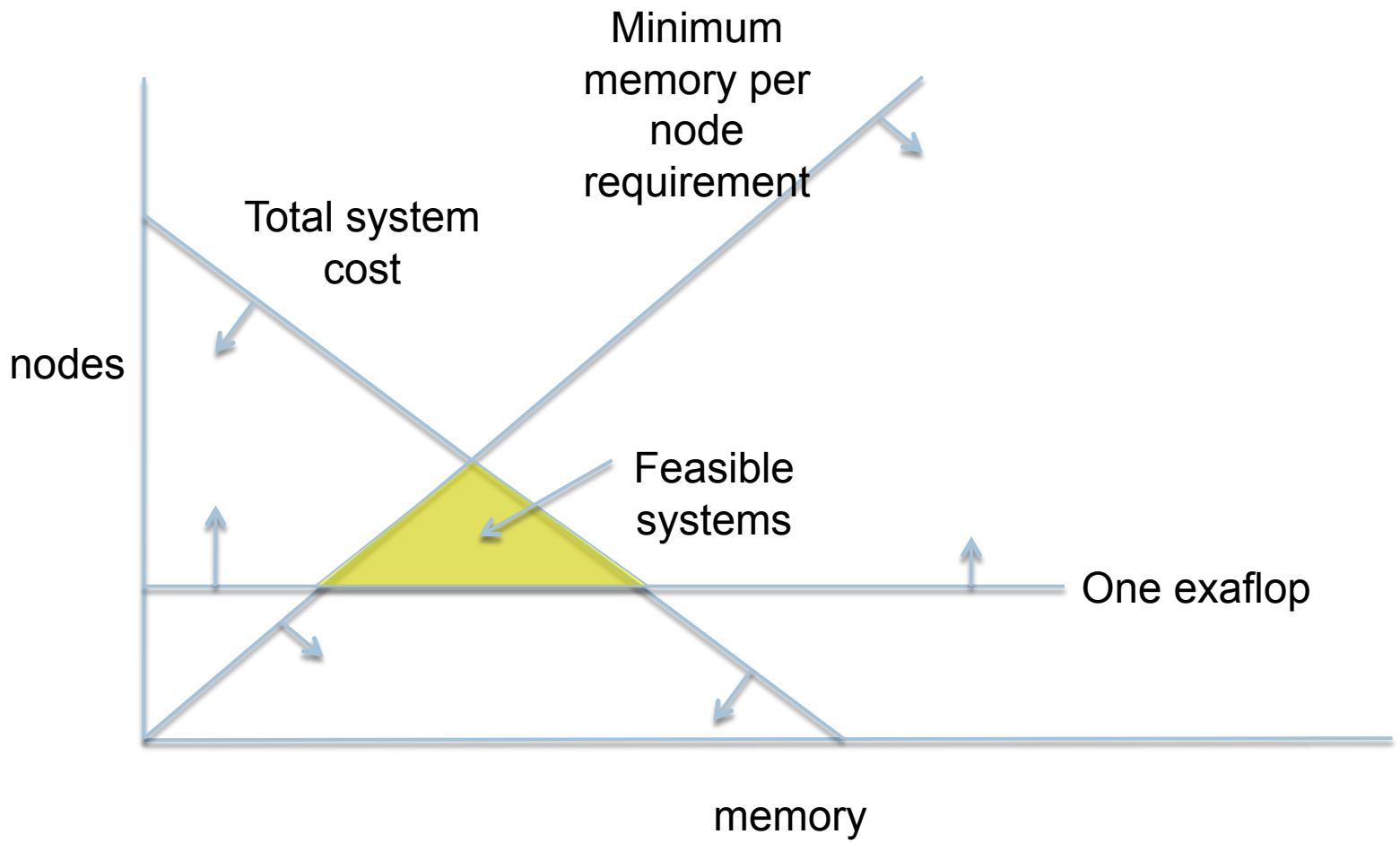


The change in memory bandwidth to compute ratio will lead to new approaches.

- Example: linear solvers
  - They start with a rough approximation and converge through an iterative process.
    - 1.125 → 1.1251 → 1.125087 → 1.12508365
  - Each iteration requires sending some numbers to neighboring processors to account for neighborhoods split over multiple nodes.
  - Proposed exascale technique: devote some threads of the accelerator to calculating the difference from the previous iteration and just sending the difference.
    - Takes advantage of "free" compute and minimizes expensive memory movement.

Inspired by David Keyes, KAUST

The trade space for exascale is very complex.



c/o *A. White, LANL*

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)
Node performance	125 GF	1,2 or 15TF	O(10) – O(100)
Node memory BW	25 GB/s	2 - 4TB/s [**.002 Bytes/Flop**]	O(100)
Node concurrency	12	O(1k) or 10k	O(100) – O(1000)
Total Node Interconnect BW	3.5 GB/s	200-400GB/s (1:4 or 1:8 from memory BW)	O(100)
System size (nodes)	18,700	O(100,000) or O(1M)	O(10) – O(100)
Total concurrency	225,000	O(billion) [O(10) to O(100) for latency hiding]	O(10,000)
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)

c/o P. Beckman, Argonne

Architectural changes will make writing fast and reading slow.

- Great idea: put SSDs on the node
  - Great idea for the simulations …
  - … scary world for visualization and analysis
    - We have lost our biggest ally in lobbying the HPC procurement folks
    - We are unique as data consumers.
- $200M is not enough…
  - The quote: "1/3 memory, 1/3 I/O, 1/3 networking … and the flops are free"
  - Budget stretched to its limit and won't spend more on I/O.

International Exascale Software Project

www.exascale.org

INTERNATIONAL EXASCALE SOFTWARE PROJECT ROADMAP 1.0

Jack Dongarra, Pete Beckman, Terry Moore, Patrick Aerts, Giovanni Aloisio, Jean-Claude Andre, David Barkai, Jean-Yves Berthou, Taisuke Boku, Bertrand Braunschweig, Franck Cappello, Barbara Chapman, Xuebin Chi

Alok Choudhary, Sudip Dosanjh, Thom Dunning, Sandro Fiore, Al Geist, Bill Gropp, Robert Harrison, Mark Hereld, Michael Heroux, Adolfy Hoisie, Koh Hotta, Yutaka Ishikawa, Fred Johnson

Sanjay Kale, Richard Kenway, David Keyes, Bill Kramer, Jesus Labarta, Alain Lichnewsky, Thomas Lippert, Bob Lucas, Barney Maccabe, Satoshi Matsuoka, Paul Messina, Peter Michielse, Bernd Mohr

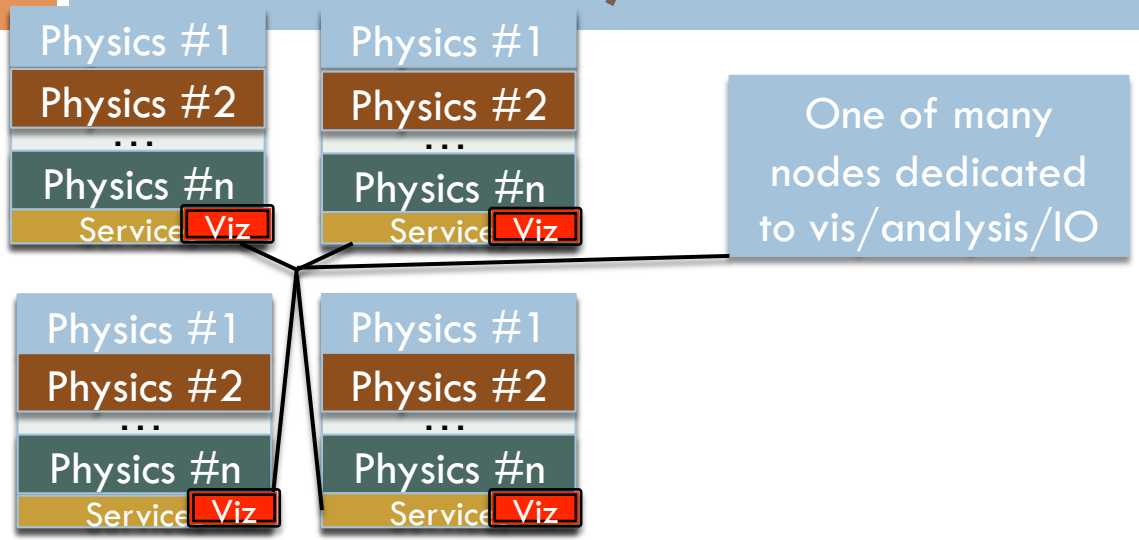
Matthias Mueller, Wolfgang Nagel, Hiroshi Nakashima, Michael E. Papka, Dan Reed, Mitsuhisa Sato, Ed Seidel, John Shalf, David Skinner, Marc Snir, Thomas Sterling, Rick Stevens, Fred Streitz

Bob Sugar, Shinji Sumimoto, William Tang, John Taylor, Rajeev Thakur, Anne Trefethen, Mateo Valero, Aad van der Steen, Jeffrey Vetter, Peg Williams, Robert Wisniewski, Kathy Yelick

SPONSORS

The International Exascale Software Roadmap, J. Dongarra, P. Beckman, et al., *International Journal of High Performance Computer Applications* **25**(1), 2011, ISSN 1094-3420. (Publ. 6 Jan 2011)

Reducing data to results (e.g. pixels or numbers) can be hard.



- Must to reduce data every step of the way.
  - Example: contour + normals + render
    - Important that you have less data in pixels than you had in cells. (*)
    - Could contouring and sending triangles be a better alternative?
  - Easier example: synthetic diagnostics