

Visualization and Data Analysis: Past, Present, and Future

“I have had my results for a long time, but I do not yet know how I am to arrive at them.”

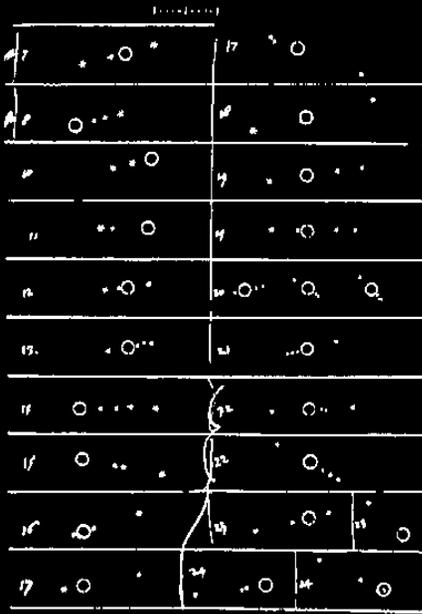
–Carl Friedrich Gauss, 1777-1855

Jet data courtesy Kwan-
Liu Ma, UC Davis.
Image courtesy Wes
Kendall, UTK

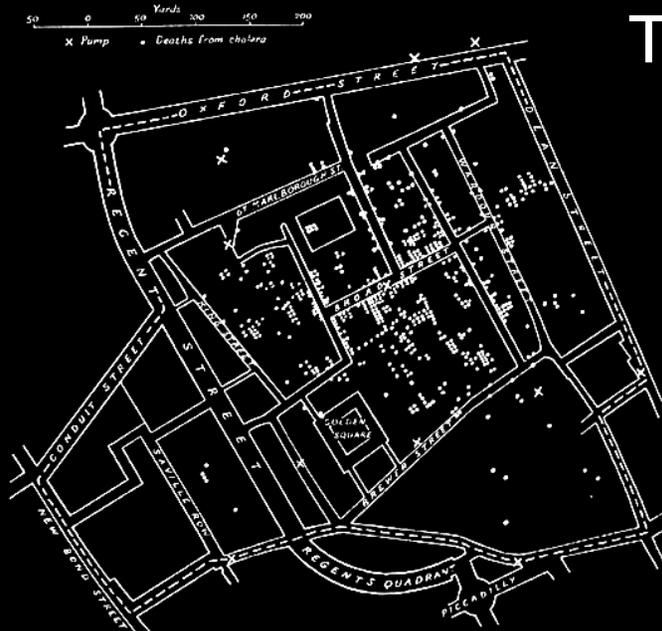
CScADS Summer
Workshop 7/22/10

Tom Peterka
tpeterka@mcs.anl.gov
Mathematics and Computer Science Division

The Past: 400 Years of Visualization

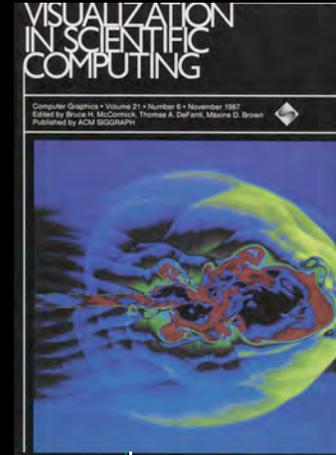


Galileo, 1610



William Playfair, 1786

John Snow, 1854

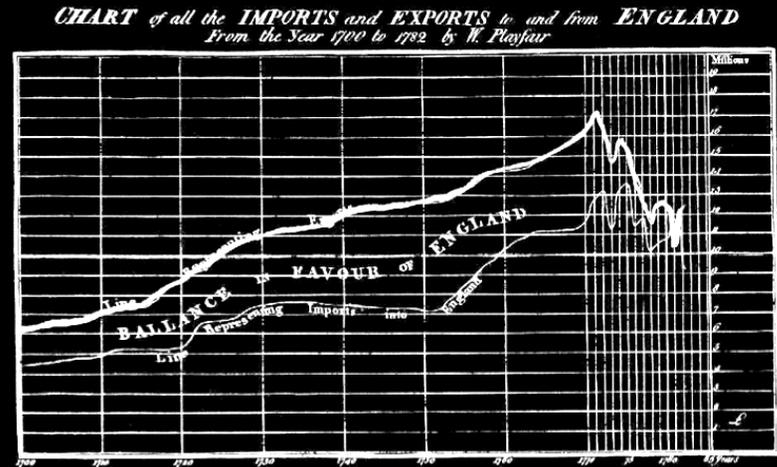


McCormick

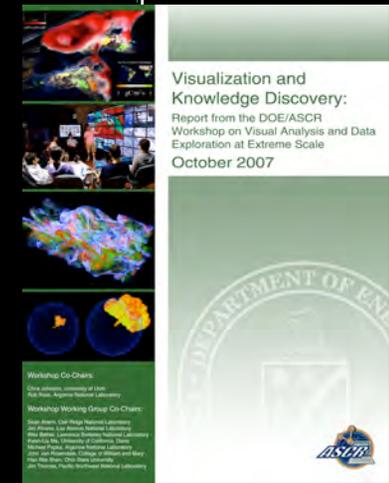
et al., 1987

Johnson

et al., 2007



The Divisions at the Bottom, express YEARS, & those on the Right hand, MILLIONS of POUNDS

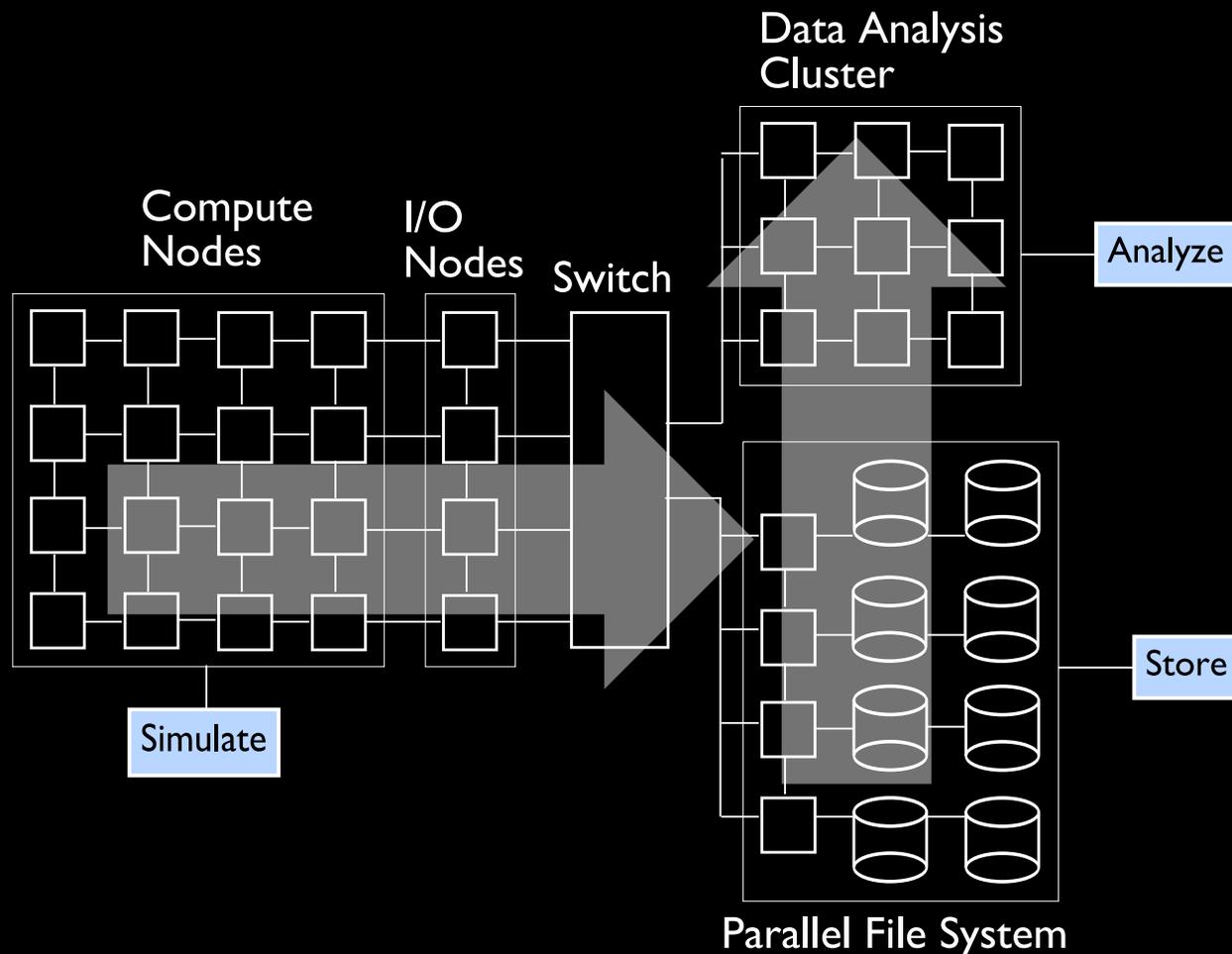


“Datasets being produced by experiments and simulations are rapidly outstripping our ability to explore and understand them” –Johnson et al., 2007.

The Present: Scientific Data Analysis in HPC Environments

Examples:

- 2D statistical graphics using R
- 3D scientific visualization using ParaView
- Scientific visualization using VisIt



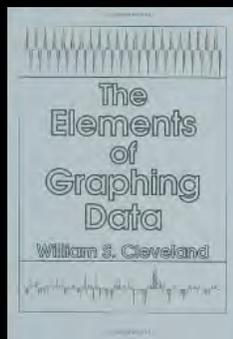
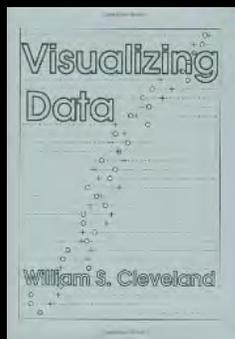
A linear, sequential pipeline where tasks mapped to architectures in fixed fashion is robust but not necessarily scalable.

Statistical Graphics:



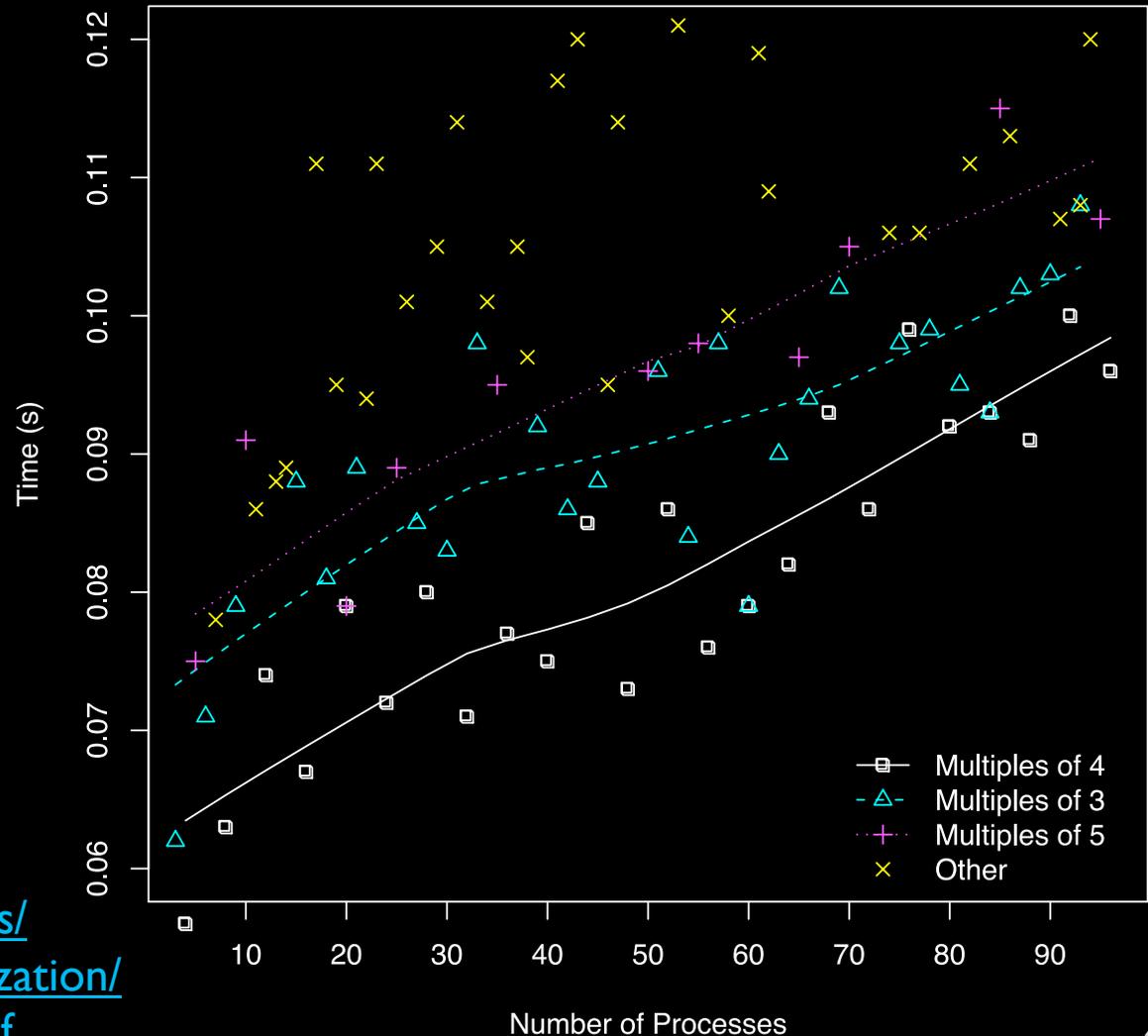
<http://r-project.org/>

- S (1976) John Chambers, Bell Labs
- R (1993) R. Gentleman and R. Ihaka, Auckland
- ~250K – IM users
- Steep learning curve (3000-page manual)
- Merges statistics with plotting
- Powerful plotting features
- Bill Cleveland approved
- Parallel R research



<http://cscads.rice.edu/workshops/summer09/slides/analysis-visualization/nagiza-samatova-cscads-2009.pdf>

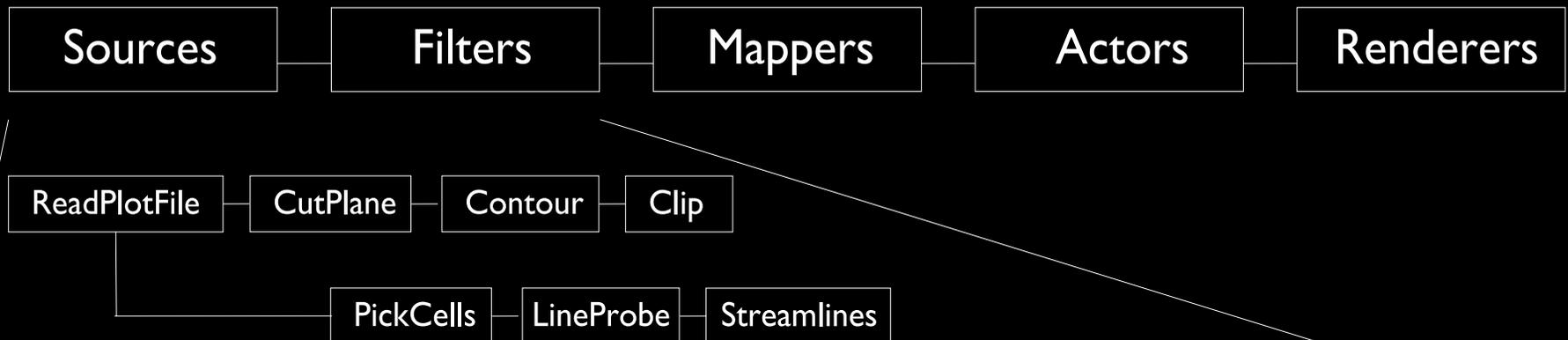
Eureka Data



Flexible Analysis & Visualization: The Pipeline Approach

Data analysis as a series of transformations

- Source, filters, and sink
- VTK (Schroeder, Martin, Lorensen 1993)
- Many tools on top of VTK: ParaView, VisIt, VisTrails
- Code reuse, portability, standardization



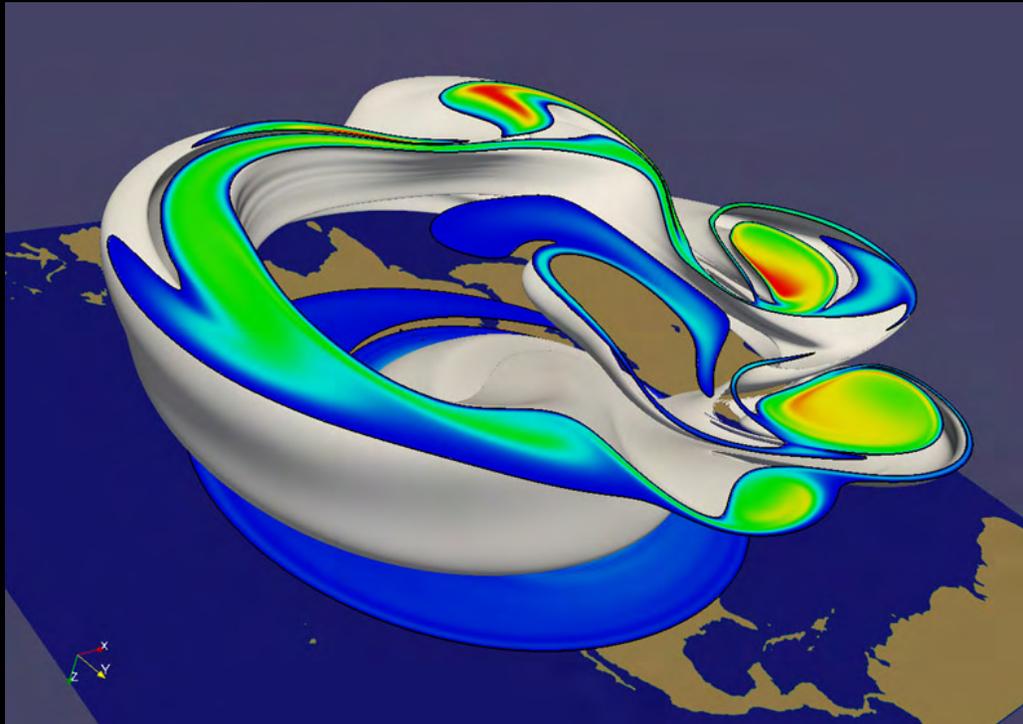
3D & 4D Scientific Visualization:



<http://www.paraview.org>

- Started in 2000 with Kitware and LANL, later included SNL and ARL
- VTK engine
- Qt interface
- Contacts: Ken Moreland (SNL), Berk Geveci (Kitware)
- Tutorials at SC, SciDAC, elsewhere

http://www.itk.org/Wiki/ParaView_2.X_documentation_and_tutorials



0.5 billion-cell
weather
visualization
courtesy Ken
Moreland

Advanced ParaView: Client-Server Mode

On Eureka:

-Add a few one-time items to `.softenvrc`, `.bashrc`

-Grab nodes in interactive mode for a time:

```
qsubi -n 4 -t 60
```

-Start the pvserver:

```
mpirun -np 4 -machinefile $COBALT_NODEFILE /soft/apps/  
paraview-3.4.0-mpich-mx/bin/pvserver
```

On local machine:

-Setup a tunnel:

```
ssh -NL 11111:vs37:11111 username@eureka.alcf.anl.gov
```

-Start ParaView, configure connection, connect

-Beware to have matched ParaView versions between client and server

<http://paraview.org/paraview/resources/software.html>

Eureka setup instructions at

https://wiki.alcf.anl.gov/index.php/Paraview_on_the_Data_Analytics_Cluster

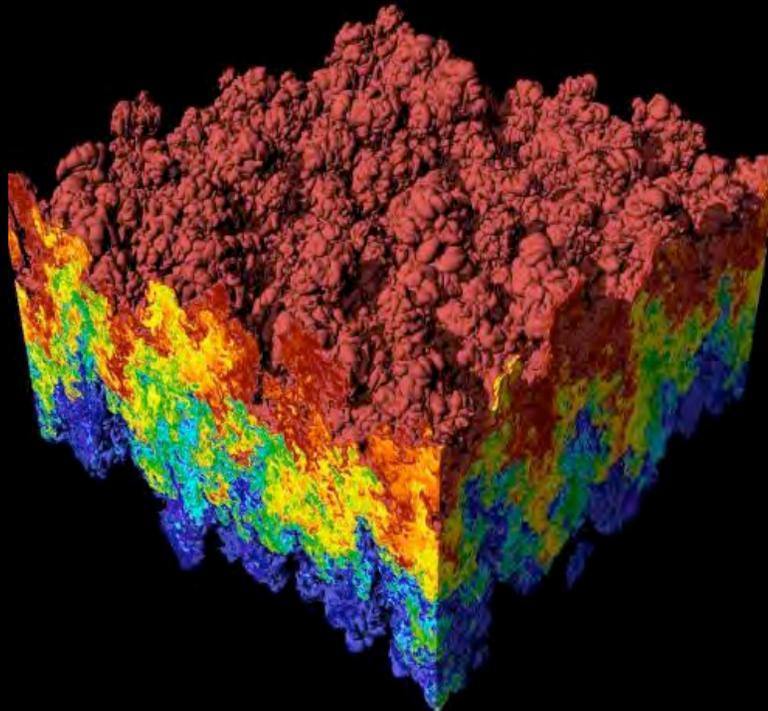
3D & 4D Scientific Visualization:



<https://wci.llnl.gov/codes/visit/home.html>

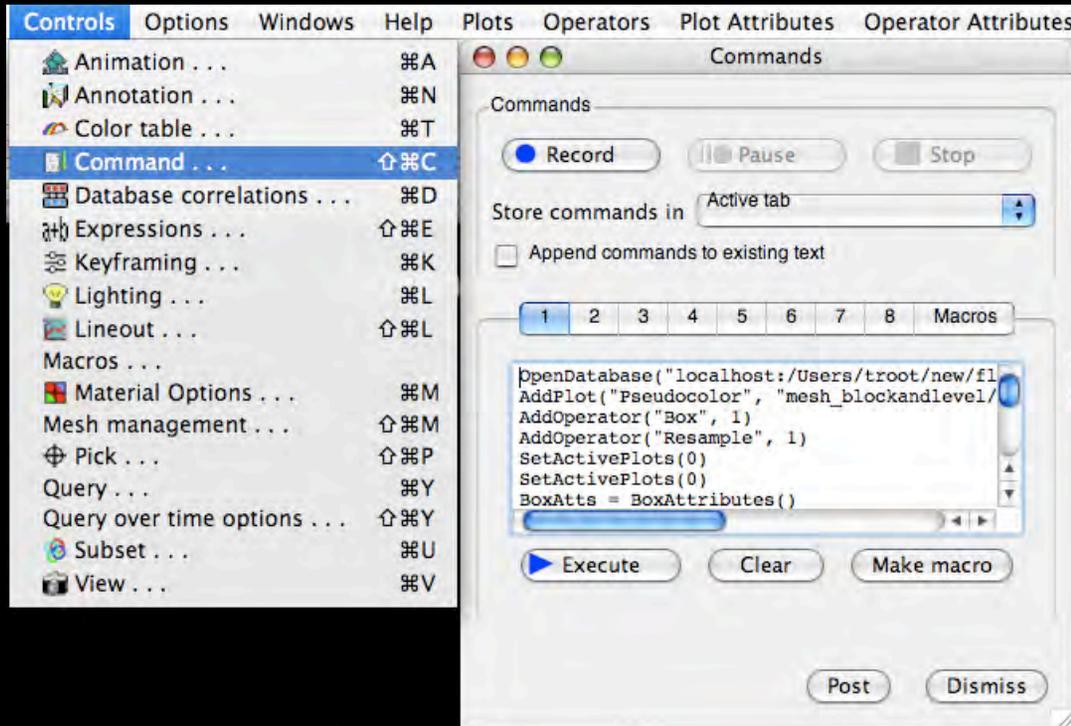
- Started in 2000 at LLNL as an ASCI-funded program
- VTK-like engine
- Qt interface
- Contacts: Hank Childs (BNL, UC-Davis), Jeremy Meredith (ORNL)
- Tutorials at SC, SciDAC, elsewhere

<https://wci.llnl.gov/codes/visit/1.4.1/VisualizationWithVisIt.pdf>



Rayleigh-Taylor
Instability
visualization
courtesy Hank
Childs

Advanced VisIt: Scripting Mode



Capture the script with Controls | Command and record

Save in script.py.

Run with:

visit -cli -nowin -s script.py

```
OpenDatabase("localhost:/filename", 0)
AddPlot("Pseudocolor", "velx", 1, 1)
AddOperator("Box", 1)
AddOperator("Resample", 1)
SetActivePlots(0)
```

```
SetActivePlots(0)
```

```
BoxAtts = BoxAttributes()
```

```
BoxAtts.amount = BoxAtts.Some
```

```
BoxAtts.minx = -0.4
```

```
BoxAtts.maxx = 0.4
```

```
BoxAtts.miny = -0.4
```

```
BoxAtts.maxy = 0.4
```

```
BoxAtts.minz = -0.4
```

```
BoxAtts.maxz = 0.4
```

```
SetOperatorOptions(BoxAtts, 1)
```

```
DrawPlots()
```

```
ExportDBAtts = ExportDBAttributes()
```

```
ExportDBAtts.db_type = "BOV"
```

```
ExportDBAtts.filename = "0.x"
```

```
ExportDBAtts.dirname = "."
```

```
ExportDBAtts.variables = "velx"
```

```
ExportDBAtts.opts.types = ()
```

```
ExportDatabase(ExportDBAtts)
```

```
quit()
```

The Data-Intensive Nature of Computing and Analysis

“Analysis and visualization will be limiting factors in gaining insight from exascale data.”

–Dongarra et al., International Exascale Software Project Draft Road Map, 2009.

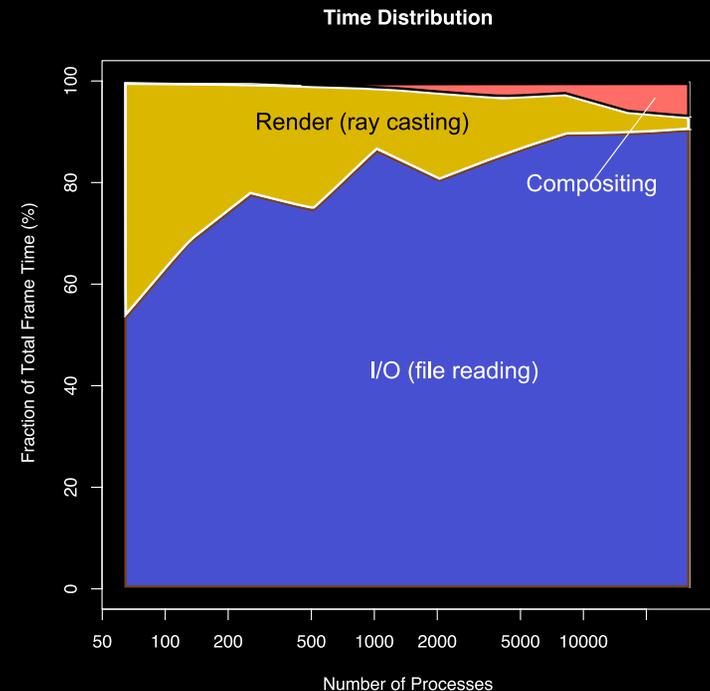
Normalized Storage / Compute Metrics

Machine	FLOPS (Pflop/s)	Storage B/W (GB/s)	Flops per byte stored	Bytes comp. per byte stored
LLNL BG/L	0.6	43	$O(10^4)$	$O(10^3)$
Jaguar XT4	0.3	42	$O(10^4)$	$O(10^3)$
Intrepid BG/P	0.6	50	$O(10^4)$	$O(10^3)$
Roadrunner	1.0	50	$O(10^5)$	$O(10^4)$
Jaguar XT5	1.4	42	$O(10^5)$	$O(10^4)$

-In 2001, Flops per bytes stored was approximately 500. Ref: John May, 2001.

-DOE science applications generate results at an average rate of 40 flops per byte of data. Ref: Murphy et al. ICS'05.

“Models ... produce data in amounts that make storage expensive, movement cumbersome, visualization difficult, and detailed analysis impossible.” -Mark Rast, Laboratory for Atmospheric and Space Physics, University of Colorado



The relative percentage of time in the stages of volume rendering as a function of system size. Large visualization is dominated by data movement: I/O and communication.

Scalable Analysis & Visualization: The Data Parallel Approach

Treat analysis as any other parallel computation

- Decompose the domain
- Assign to processors
- Combine local and global operations
- Use parallel I/O, MPI, other programming models
- Balance load, minimize communication
- Measure strong, weak scaling, efficiency, isoefficiency

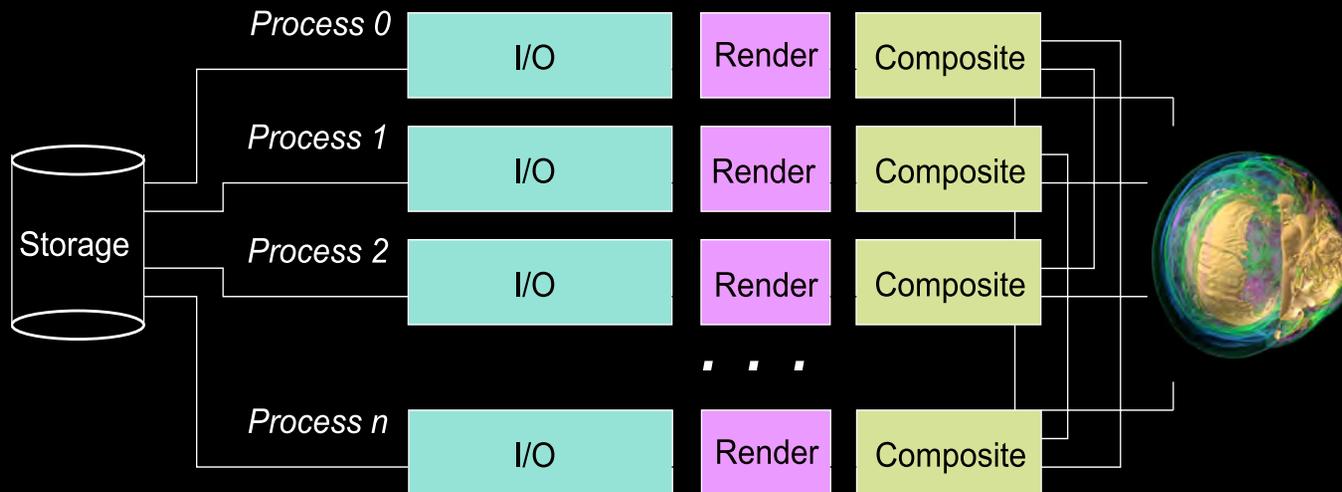


Integrate with simulation

“The combination of massive scale and complexity is such that high performance computers will be needed to analyze data, as well as to generate it through modeling and simulation.”

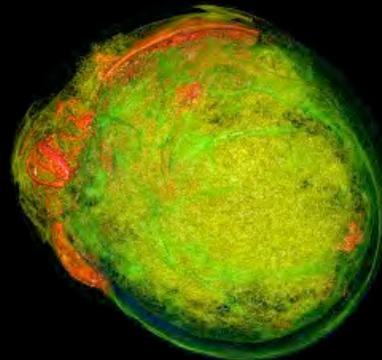
–Lucy Nowell, Scientific Data Management and Analysis at Extreme Scale, Office of Science Program Announcement LAB 10-256, 2010.

Large-Scale Parallel Volume Rendering



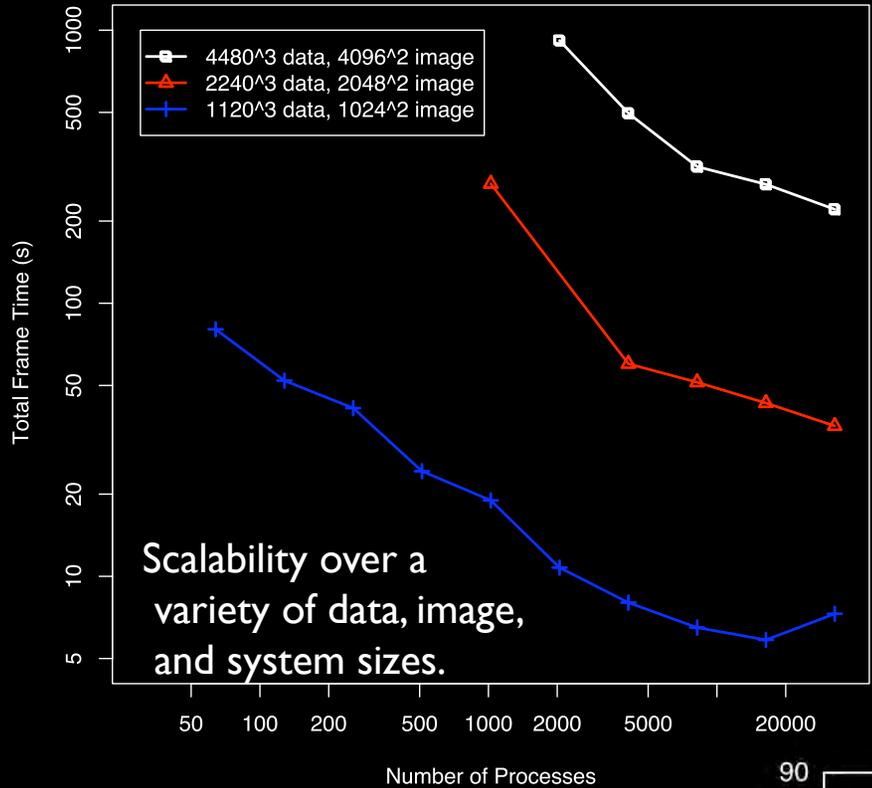
Parallel structure for volume rendering algorithm consists of 3 stages performed in parallel

Entropy over 100 time-steps



Parallel Volume Rendering on the IBM Blue Gene/P. EGPGV'08.

Volume Rendering End-to-End Performance



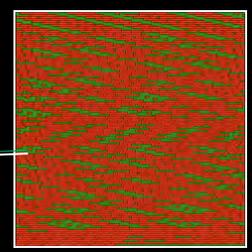
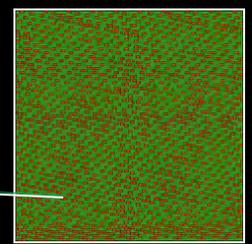
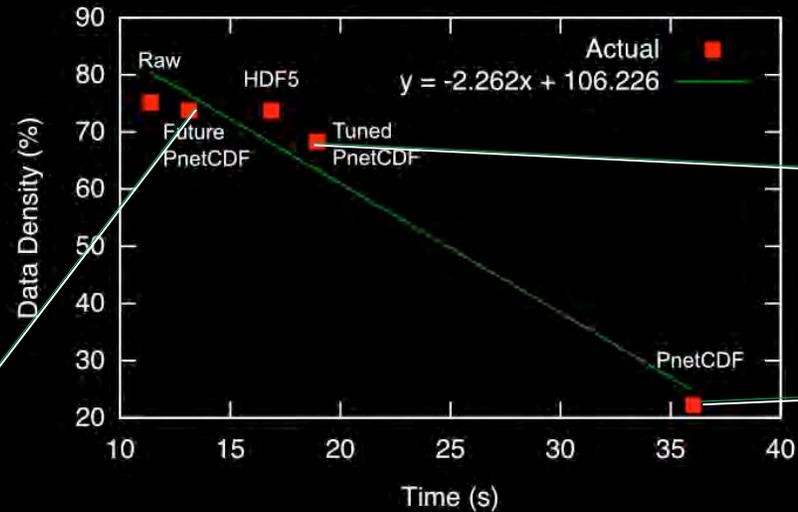
Scalability over a variety of data, image, and system sizes.

Benchmarking Performance

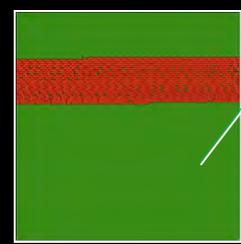
Grid Size	Time-step size (GB)	Image size (px)	# Procs	Tot. time (s)	% I/O	Read B/W (GB/s)
2240 ³	42	2048 ³	8K	51	96	0.9
			16K	43	97	1.0
			32K	35	96	1.3
4480 ³	335	4096 ³	8K	316	96	1.1
			16K	272	97	1.3
			32K	220	96	1.6

Volume rendering performance at large size is dominated by I/O.

I/O Mode Comparison



Changing data file layout can improve I/O performance, shown by access pattern signatures and performance data.



Large Scale Parallel Image Compositing

The final stage in sort-last parallel visualization algorithms:

1. Partition data among processes
2. Visualize local data
3. Composite resulting images into one

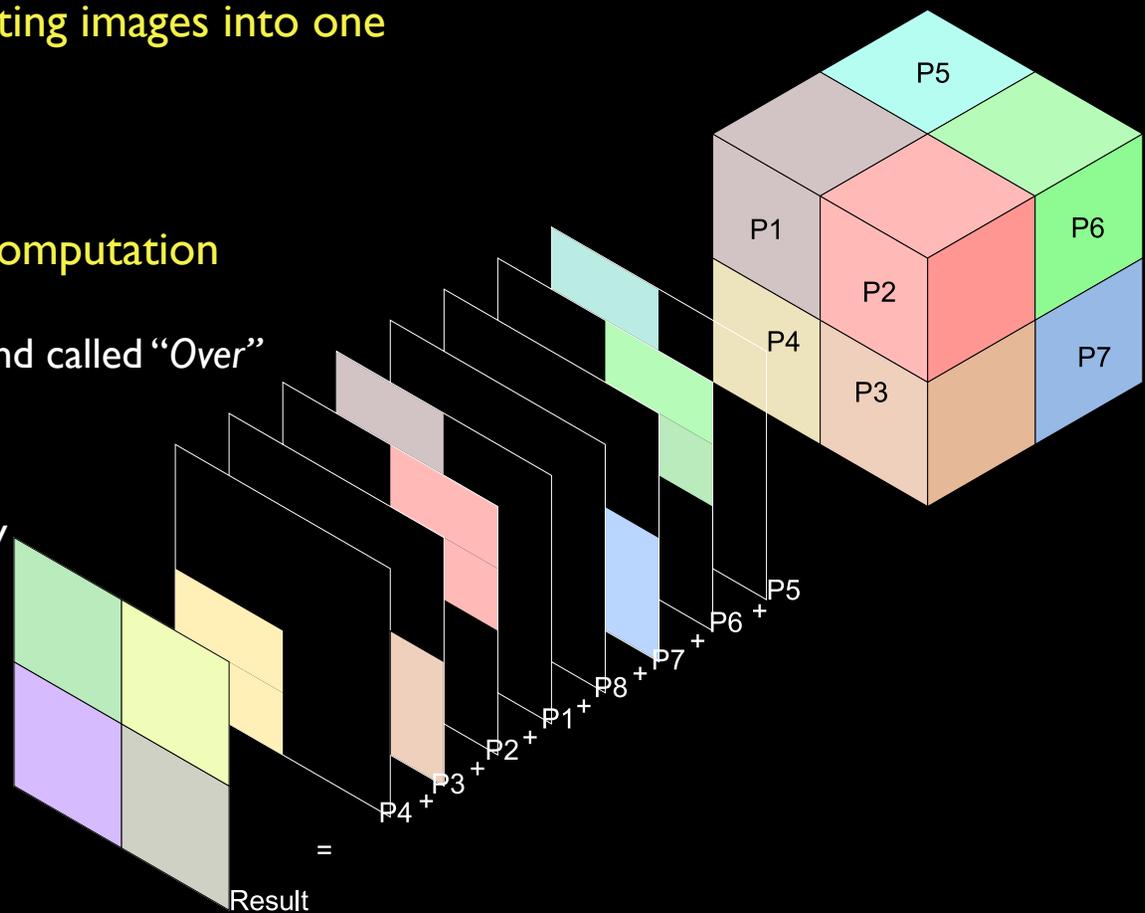
Composition = communication + computation

The computation is usually an alpha-blend called “Over”

$$i = (1.0 - \alpha_{old}) * i_{new} + i_{old}$$

$$\alpha = (1.0 - \alpha_{old}) * \alpha_{new} + \alpha_{old}$$

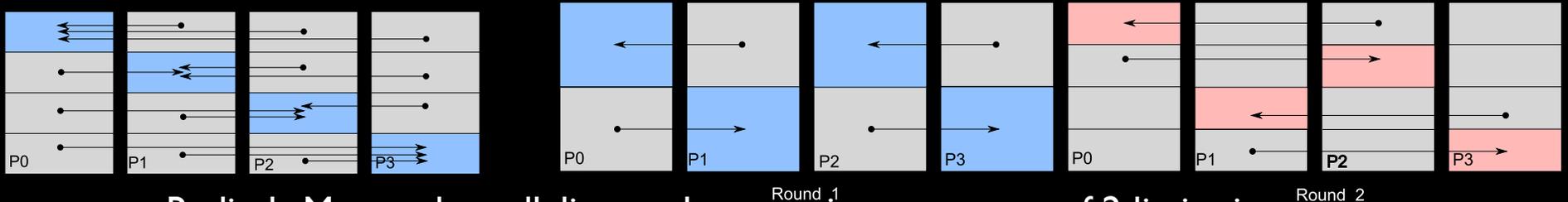
where i = intensity (R,G,B), α = opacity



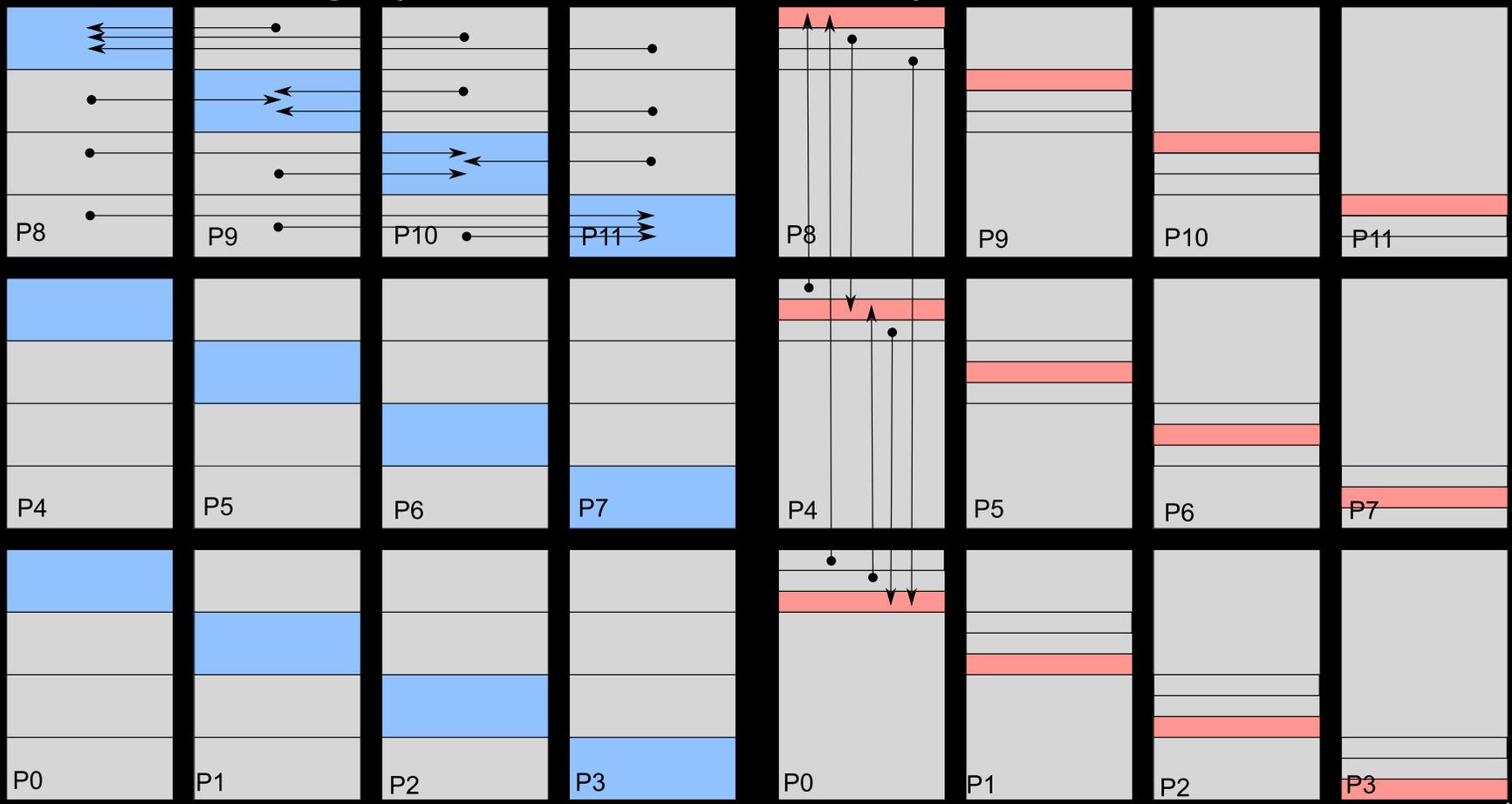
Direct-Send, Binary Swap, and Radix-k

Direct-send: Parallel, contentious

Binary swap: Low parallelism, limited to powers of 2



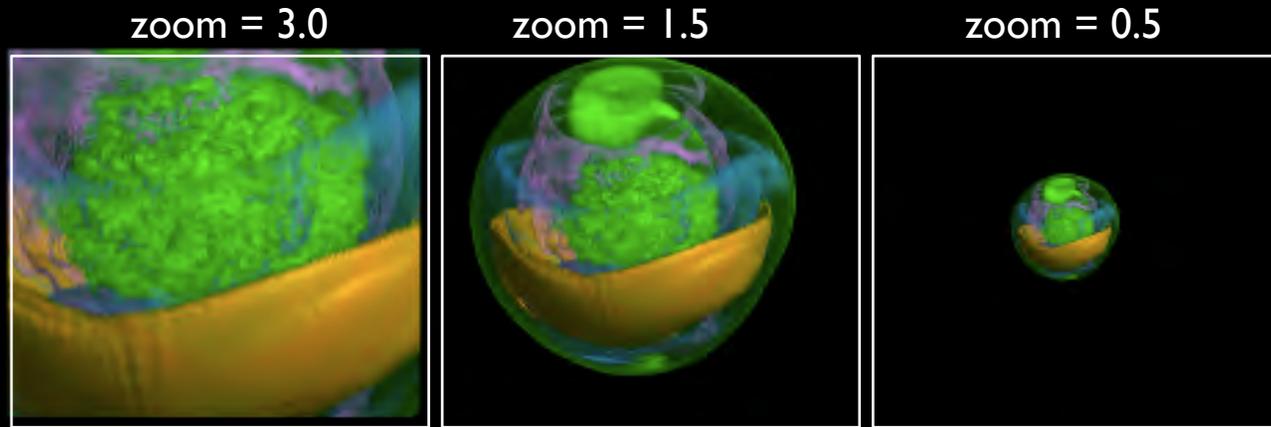
Radix-k: Managed parallelism and contention, no power of 2 limitations



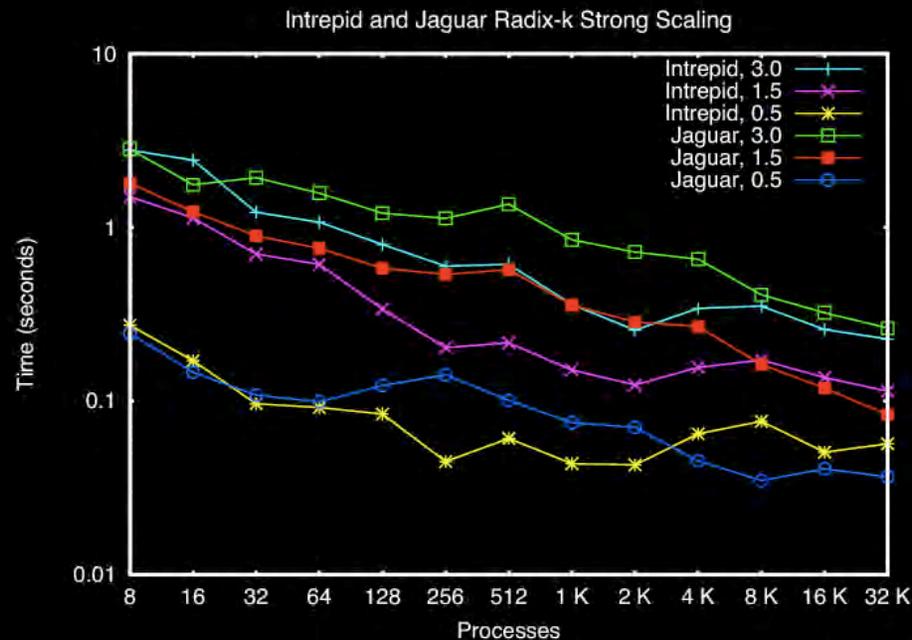
Round 1

Round 2

Radix-k at Scale

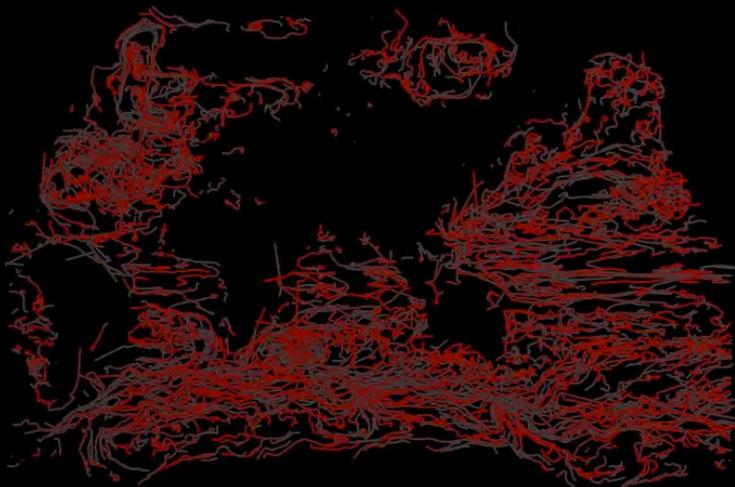


Examples of volume rendering at the 3 zoom levels shown below



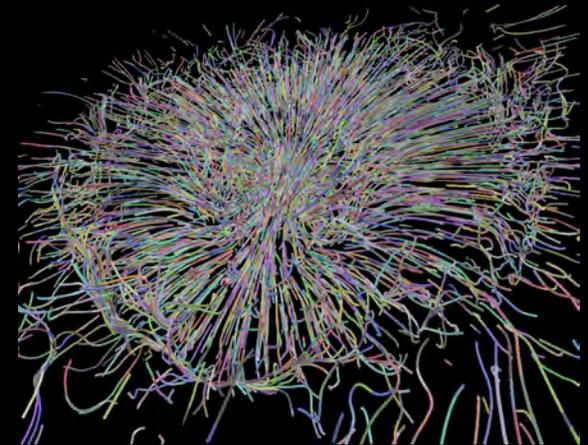
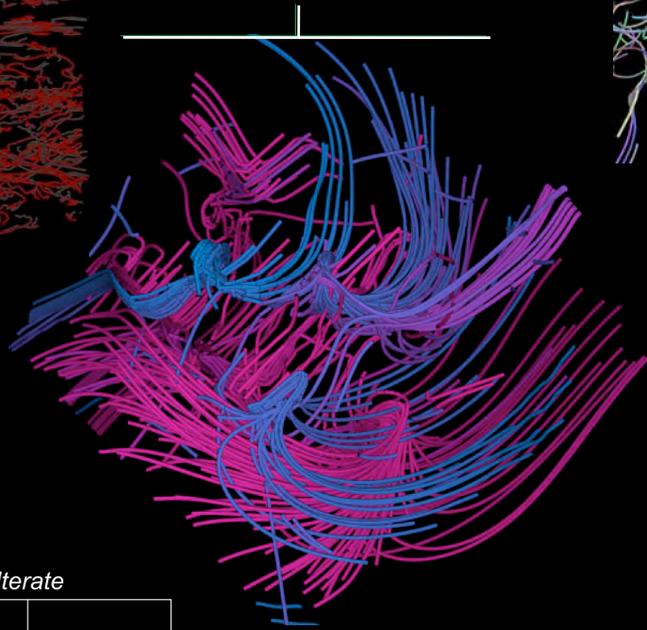
3X – 6X
improvement over
optimized binary
swap (with
bounding boxes
and RLE) in many
cases. 64Mpix at
32K processes can
be composited at .
08 s, or 12.5 fps.

Large-Scale Parallel Particle Tracing

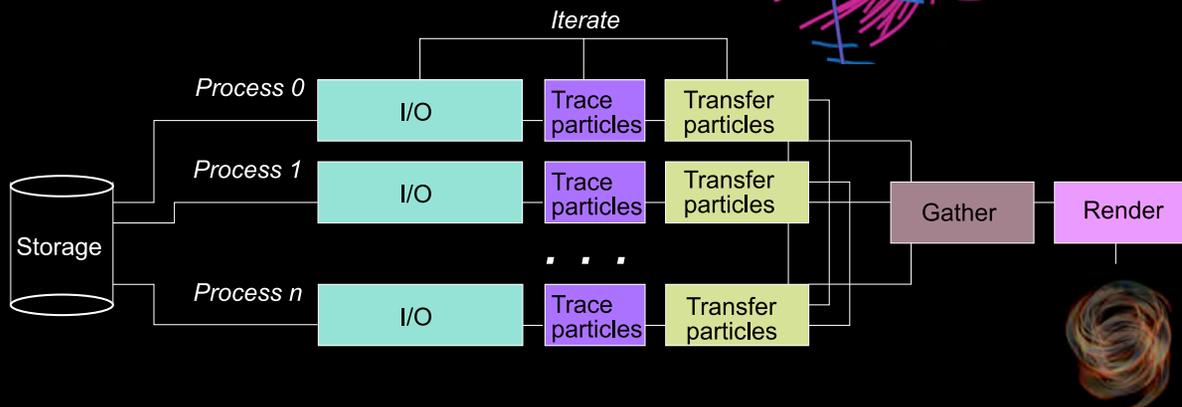


Ocean current data courtesy Rob Jacob, ANL

Jet data courtesy Kwan-Liu Ma, UC Davis

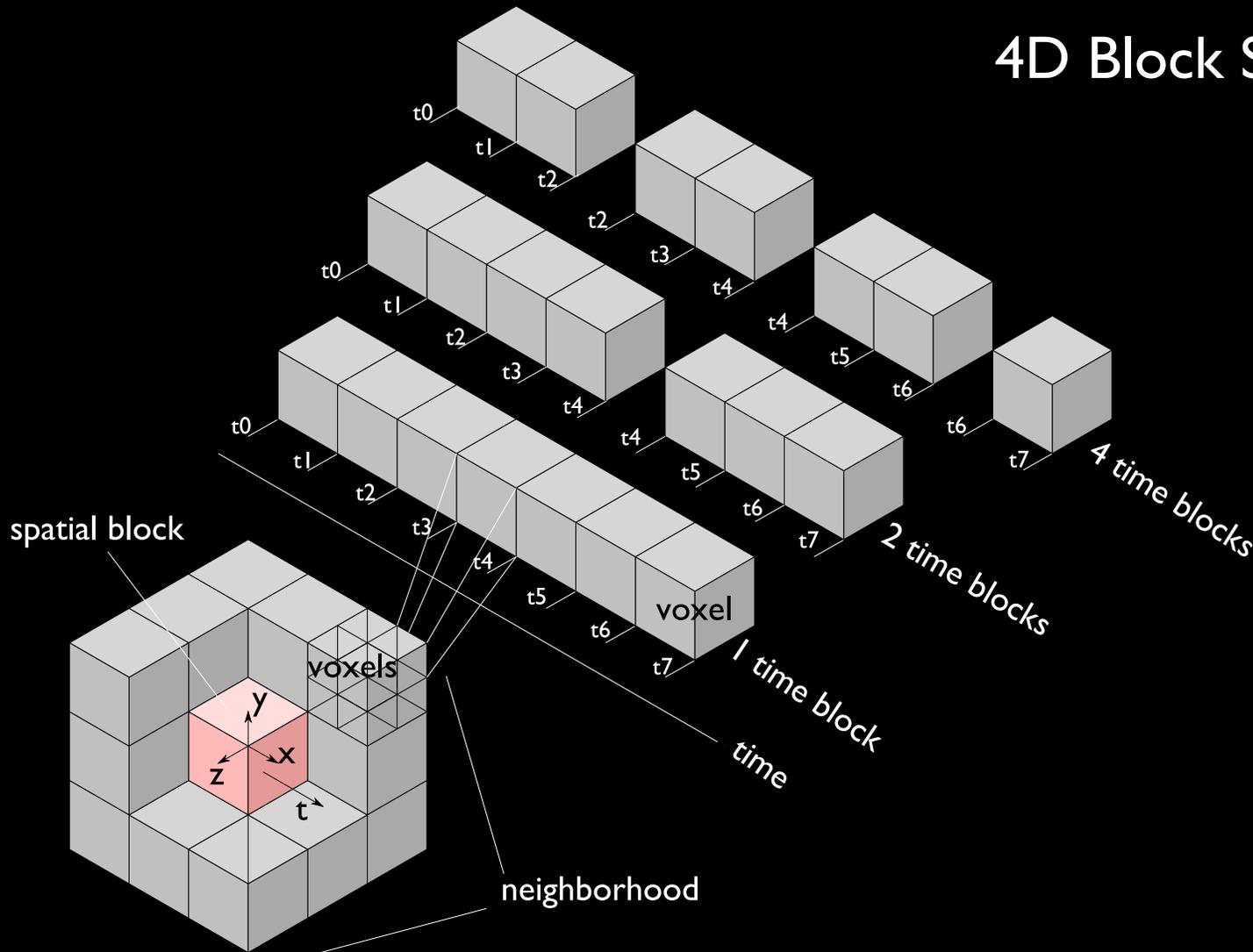


Type IA supernova data courtesy George Jordan, UofC FLASH Center



Parallel structure for flow visualization algorithm consists of iterations of particle tracing and transfer, followed by a rendering stage.

4D Block Structure

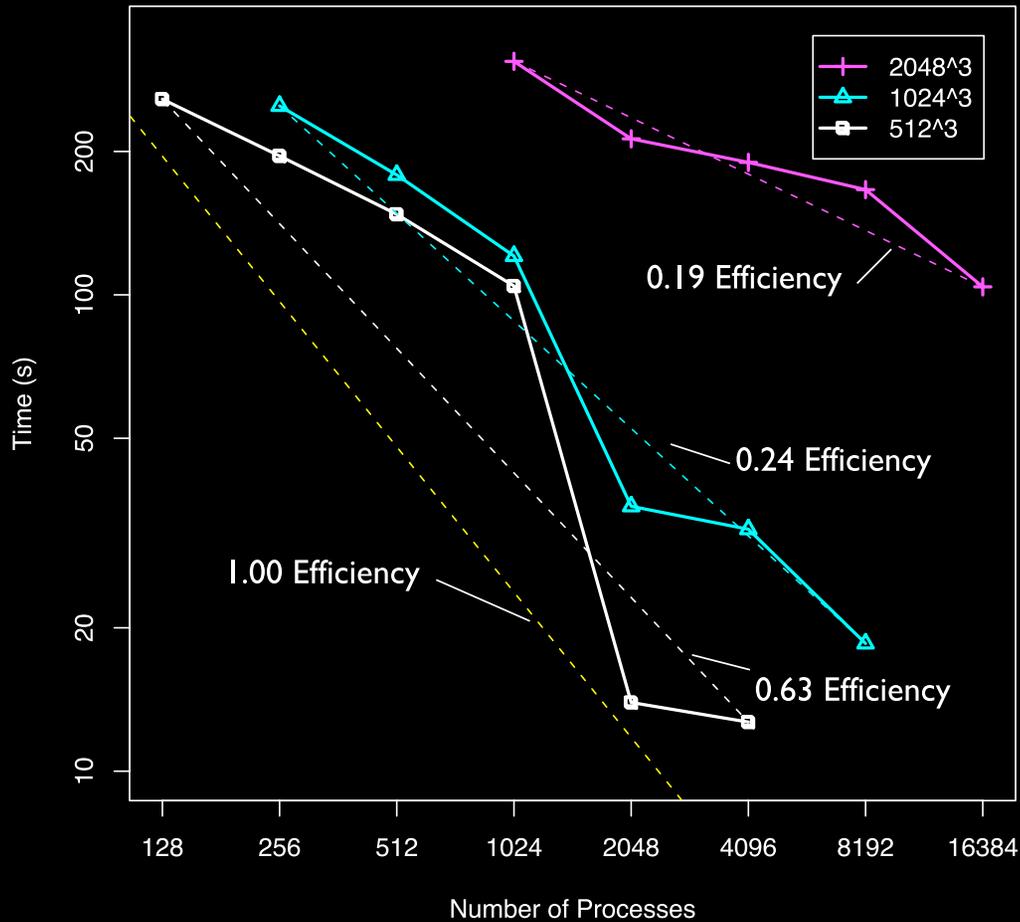


- Time blocks are control in-core / out-of-core behavior
- One time block resident in memory at any one time
- Memory distributed in spatial (x,y,z) dimensions, serialized in time dimension

- True 4D blocks
- Blocks consist of 4D voxels (eg 16x16x16x4 time steps)
- Messages are sent when any of the 4 extents are exceeded
- $3^4 = 81$ neighbors for regular grid, counting self

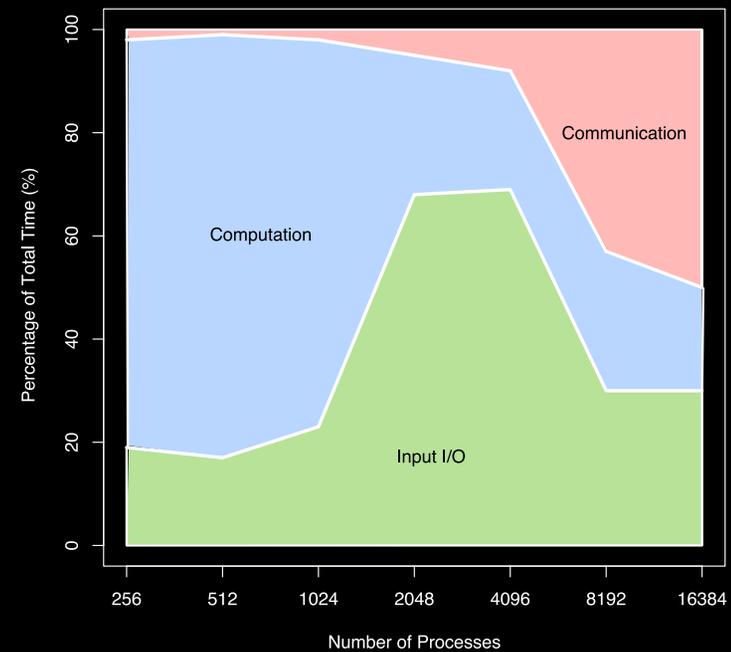
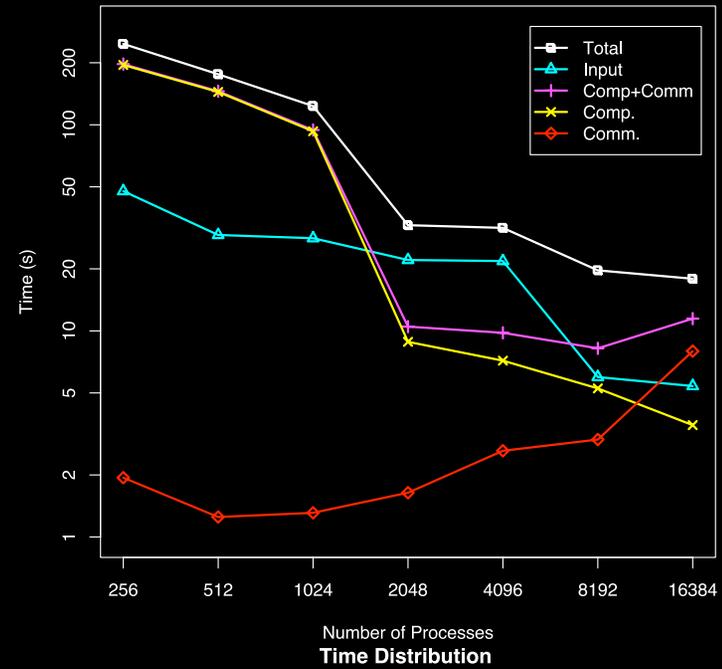
Strong Scaling Baseline Performance

Time for 10 Rounds, Various Data Sizes, 128K Particles



Thermal hydraulics flow. 134M cells, 8K particles.
1,2,4,8,16 round robin blocks per process.

Time for 10 Rounds, 1024³ Data Size



Conclusions

- There are different levels of analysis needs
- Tools have steep learning curves, may require expert assistance
- Peta- and exascale requires new thinking about analysis
- HPC resources can be harnessed for scalable run-time analysis
- Scalable analysis is data-intensive
- Detailed study of analysis data patterns is needed
- Continued collaboration with scientists is necessary

“The purpose of computing is insight, not numbers.”

–Richard Hamming, 1962

Acknowledgments:

Facilities

Argonne Leadership Computing Facility (ALCF)
Oak Ridge National Center for Computational
Sciences (NCCS)

Funding

US DOE SciDAC UltraVis Institute

People

Rob Ross, Han-Wei Shen, Jian Huang, Wes
Kendall, Rajeev Thakur, Dave Goodell, Kwan-Liu
Ma, Hongfeng Yu, Rob Latham

Tom Peterka

tpeterka@mcs.anl.gov