Dealing with Large Datasets

or, “So I have 40TB of data..”

Jonathan Dursi, SciNet/CITA, University of Toronto

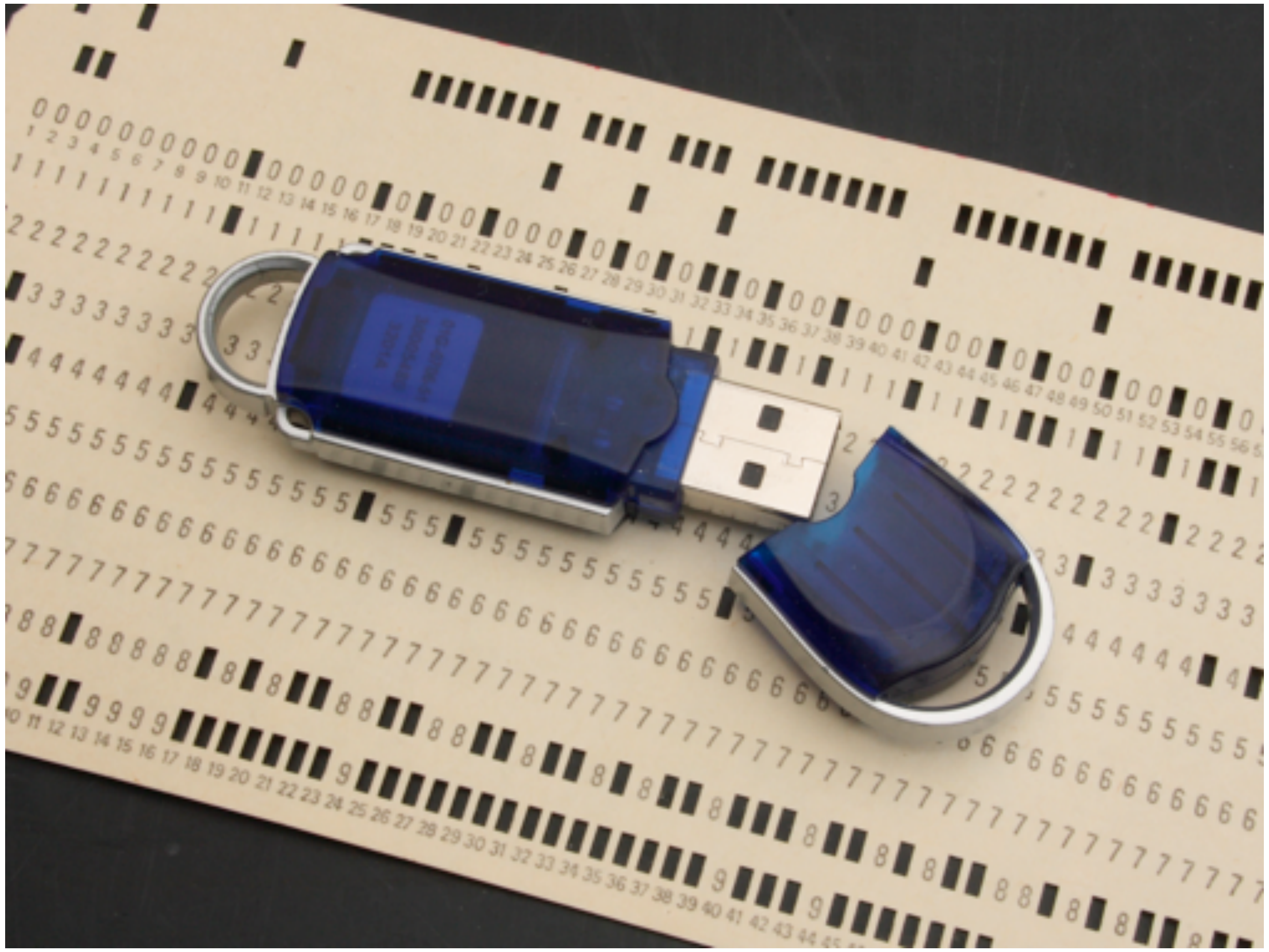
Data is getting bigger

- Increase in computing power makes simulations larger/more frequent
- Increase in sensor technology makes experiments/ observations larger
- Data sizes that used to be measured in MB/GB now measured in TB/PB.
- Easier to make big data than to do something useful with it!

Economist, 27 Feb 2010

What is “big data”?

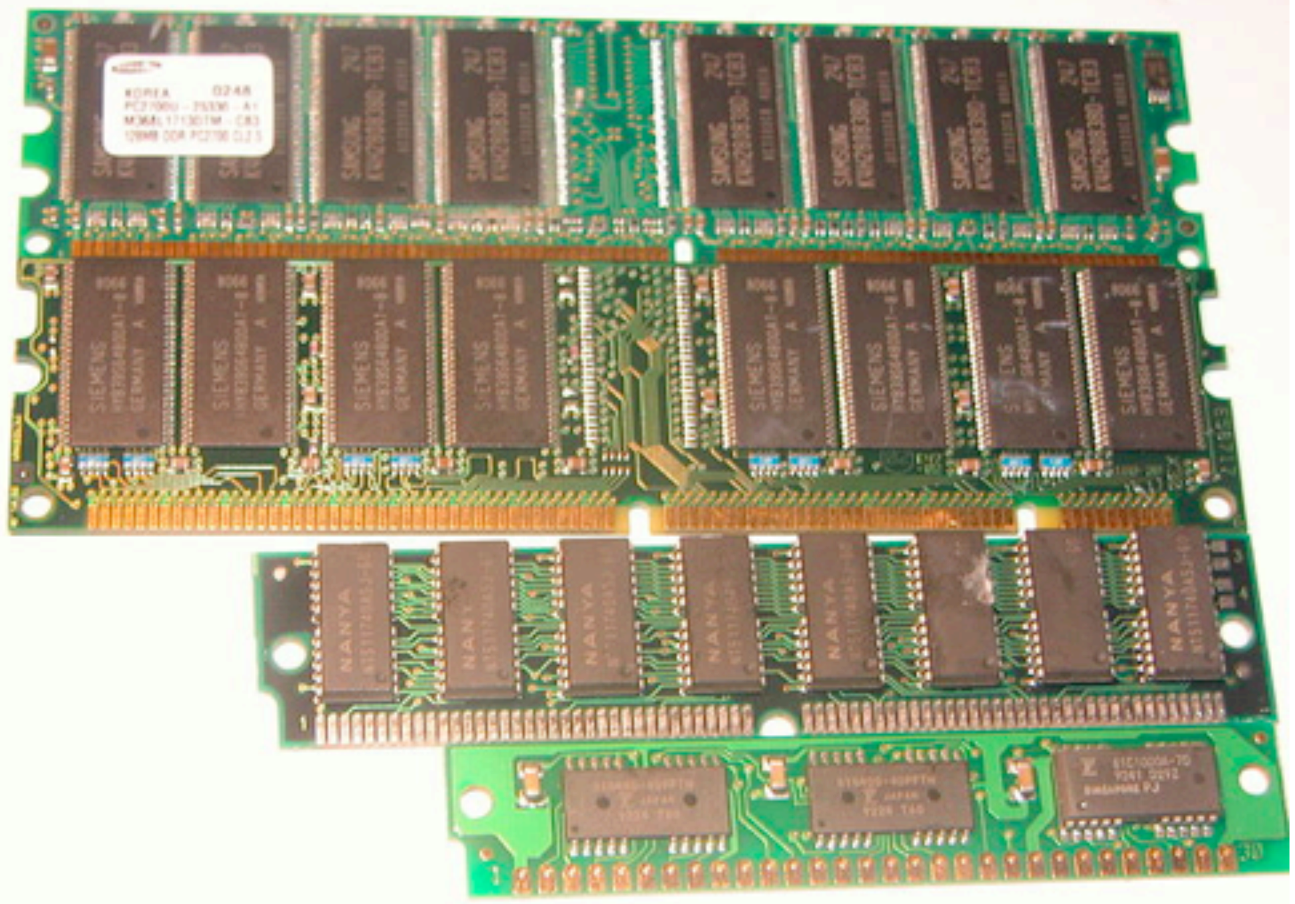
- Absolute numbers don’t matter
  - (and change rapidly anyway)
- Big is defined by its effects on us the scientists
  - What techniques must we use to analyze it?
- Two big milestones that define big..



Ian-S, http://www.fotopedia.com/items/flickr-2152798588

Big #1: Too Big to fit in Memory

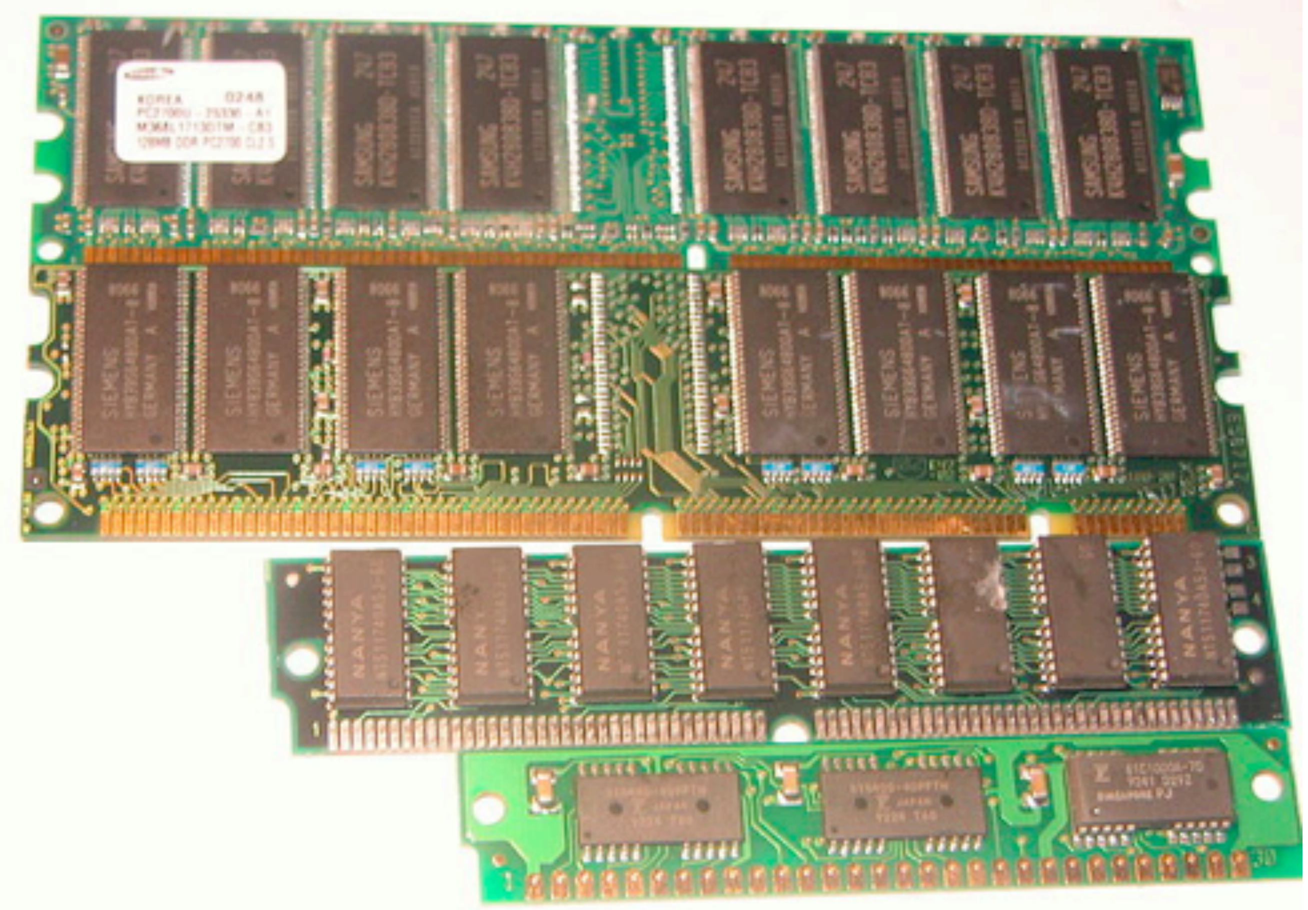
- If fits in memory,
  - easy global view of whole problem
  - simple workflow



http://commons.wikimedia.org/wiki/File:Kinds-of-RAM.JPG

Big #1: Too Big to fit in Memory

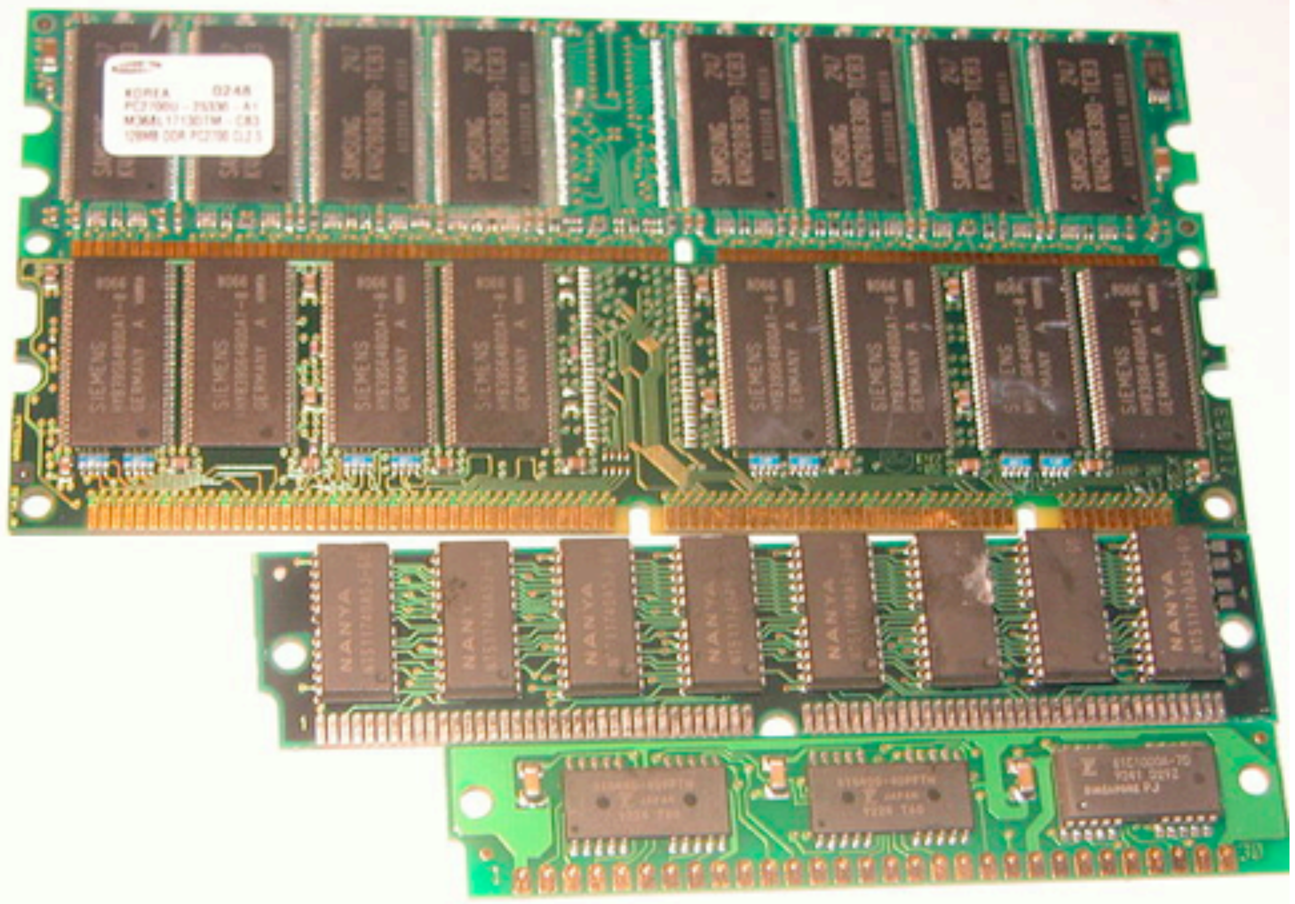
- Otherwise, must use more complicated techniques
  - Out of core
  - Multi-resolution
  - Parallel computation



http://commons.wikimedia.org/wiki/File:Kinds-of-RAM.JPG

Big #1: Too Big to fit in Memory

- Today:
  - ~2-16GB for workstation
  - ~128GB-256GB on specialized (shared) machine.



http://commons.wikimedia.org/wiki/File:Kinds-of-RAM.JPG

Big #2: Too Big to fit on one disk

- Once data size becomes comparable to typical storage medium, I/O becomes significant limitation
- Hardware has to be considered (RAID, parallel file systems)
- Almost certainly need parallel computing.

http://commons.wikimedia.org/wiki/File:Hard-drive.jpg

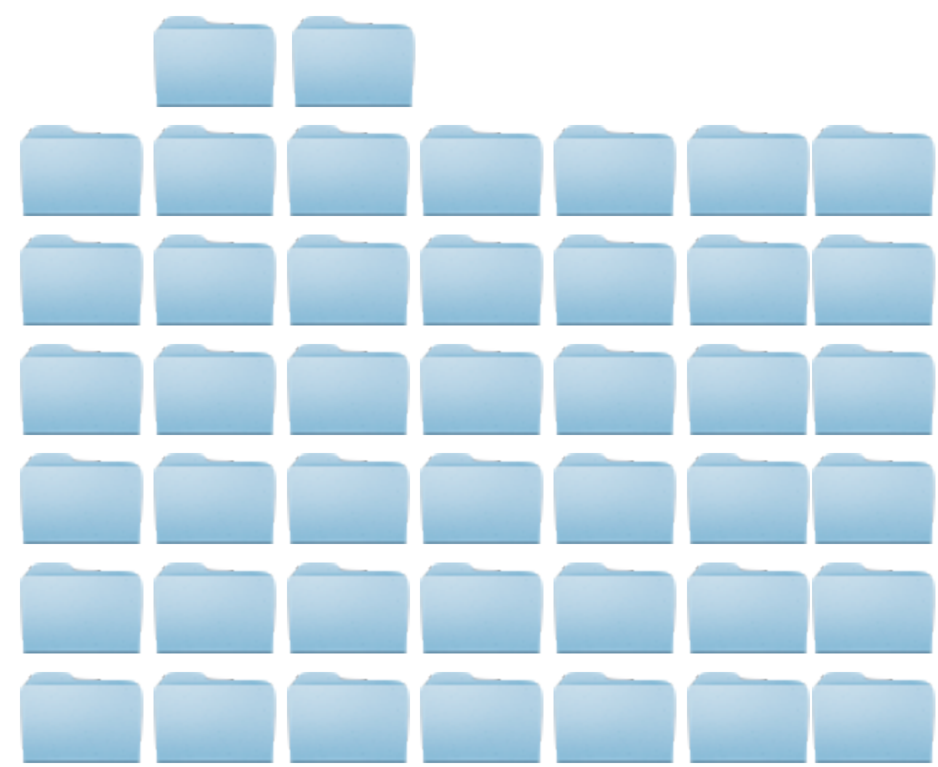
2 Terabytes $100 Hard drive	**20 Terabytes** medium-sized HPC simulation; 128 GPC nodes, 10 outputs.	**30 Terabytes** raw data for 30x short-read sequence of human genome
150 Terabytes Size of MERRA climate database of 1979-current obs. data	**330 Terabytes** Data output by LHC every week	**1 Petabyte:** LSST each ~month, must be searched in ~ real time for transient events

After Marian Bantjes, Wired magazine, Jun 2008

Big Data can be *big*, or *many*, data sets

- Few, very large, simulation outputs
- Very many, comparably small, experimental data
- Both can quickly add to TB
- Surprisingly, advice/techniques for these two cases overlap strongly.

vs



For big data, *scalability* is critical

- Many tools work great on desktop-sized datasets,
- But fail utterly at large scale:
  - on enormous files
  - when same task must be repeated on thousands of files.
- Need to use tools, techniques that handle scale well.

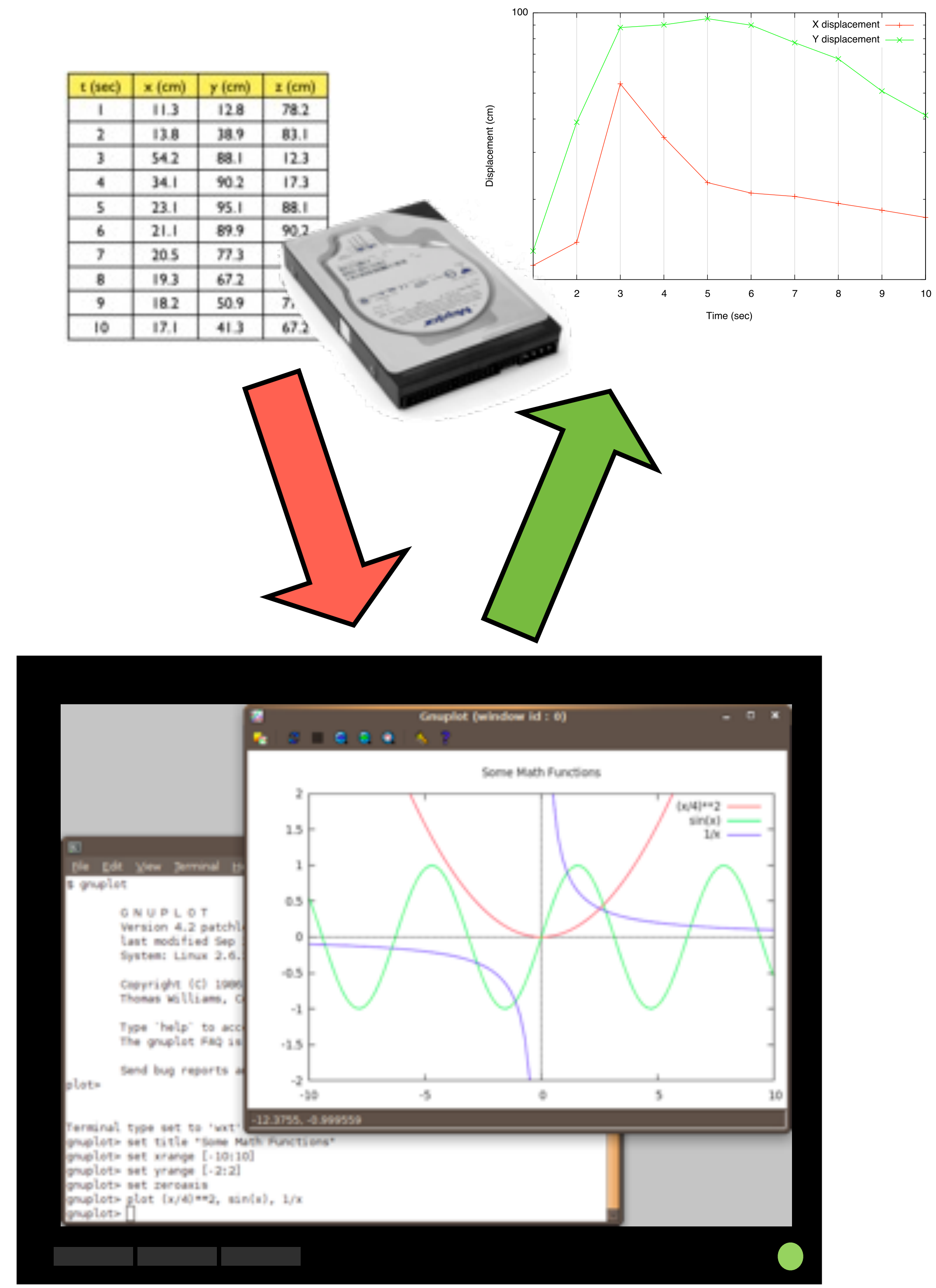
Not all tools will work under arbitrarily large loads.

Scalability requires many things:

- Scalable I/O strategy
- Scalable data management
- Automatability / scriptability
- Scalable analysis flow
- Scalable tools
- Least scalable link in the workflow will bottleneck **entire process.**

Jessie Menn
http://www.flickr.com/photos/jesse_menn/3164739580/

At scale, must plan how to do Input/Output



- Just reading and writing 40 TB from one drive at a time takes ~2weeks!
  - 50MB/s read, 100MB/s write
  - ..And that's **best** case
- Before planning analysis, planning *flow of data* important.

Where, how you do I/O matters.

- Binary - smaller files, *much* faster to read/write.
- You're not going to read 40TB of data yourself; don't bother trying.
- **Data:** machine-readable
- **Output:** graphs, summary tables - human-readable

	ASCII	binary
Large Parallel File System		
	173s	6s
Ramdisk		
	174s	1s
Typical work station disk		
	260s	20s

Timing data: writing 128M double-precision numbers

Where, how you do I/O matters.

- *All* disk systems do best when reading/writing large, contiguous chunks
- I/O operations (IOPS) are themselves expensive
  - moving around within a file
  - opening/closing
- Seeks - 3-15ms - enough time to read 0.75 MB!

Typical work station disk

binary - one large read
14s

binary - 8k at a time
20s

binary - 8k chunks, lots of seeks
150s

binary - seeky + open and closes
205s

Timing data: reading 128M double-precision numbers

Where, how you do I/O matters.

- RAM is much better for random accesses
- Use right storage medium for the job!
- Where possible, read in contiguous large chunks, do random access in memory
- Much better *if* you use most of data read in

Large Parallel File System

ASCII	**binary**
173s	6s

Ramdisk

ASCII	**binary**
174s	1s

Typical work station disk

ASCII	**binary**
260s	20s

Ramdisk

binary - one large read
1s

binary - 8k at a time
1s

binary - 8k chunks, lots of seeks
1s

binary - seeky + open and closes
1.5s

Parallel I/O and large file systems

- Large disk systems featuring many servers, disks
- Can serve files to many clients concurrently
- Parallel File Systems -
  - Lustre, Panasas, GlusterFS, Ceph, GPFS...

SciNet ~2k drives

Where, how you do I/O matters.

- Well built parallel file systems can greatly increase bandwidth
- But typically even worse penalties for seeky/IOPSy operations.
- Parallel FS can help with big data in two ways

Large Parallel File System

ASCII	**binary**
173s	6s

Ramdisk

ASCII	**binary**
174s	1s

Typical work station disk

ASCII	**binary**
260s	20s

Large Parallel File System

binary - one large read
7.5s

binary - 8k at a time
62 s

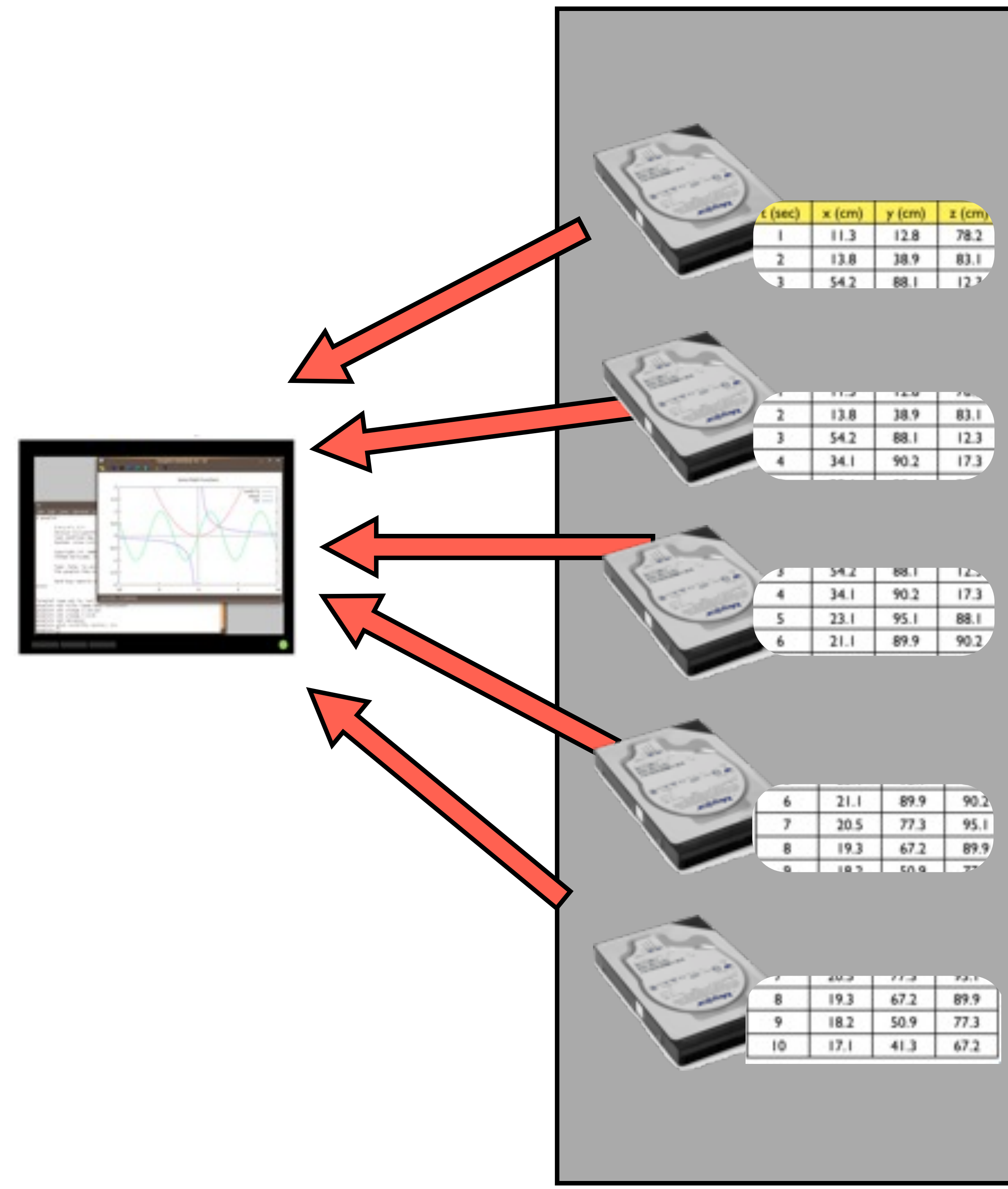
binary - 8k chunks, lots of seeks
428 s

binary - seeky + open and closes
2137 s

Striping data across disks

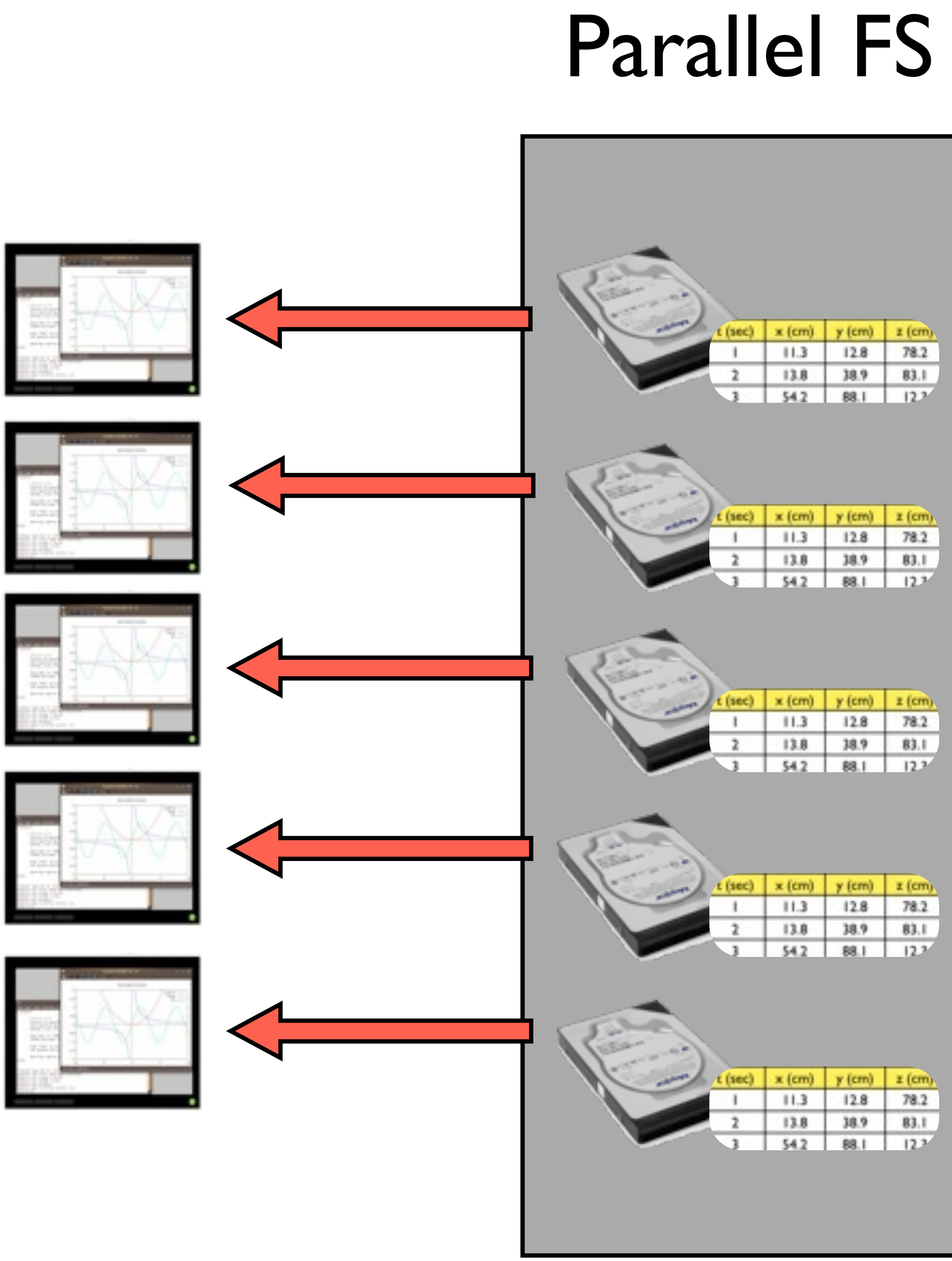
- Single client can make use of multiple disk systems simultaneously
- “Stripe” file across many drives
- One drive can be finding next block while another is sending current block

Parallel FS



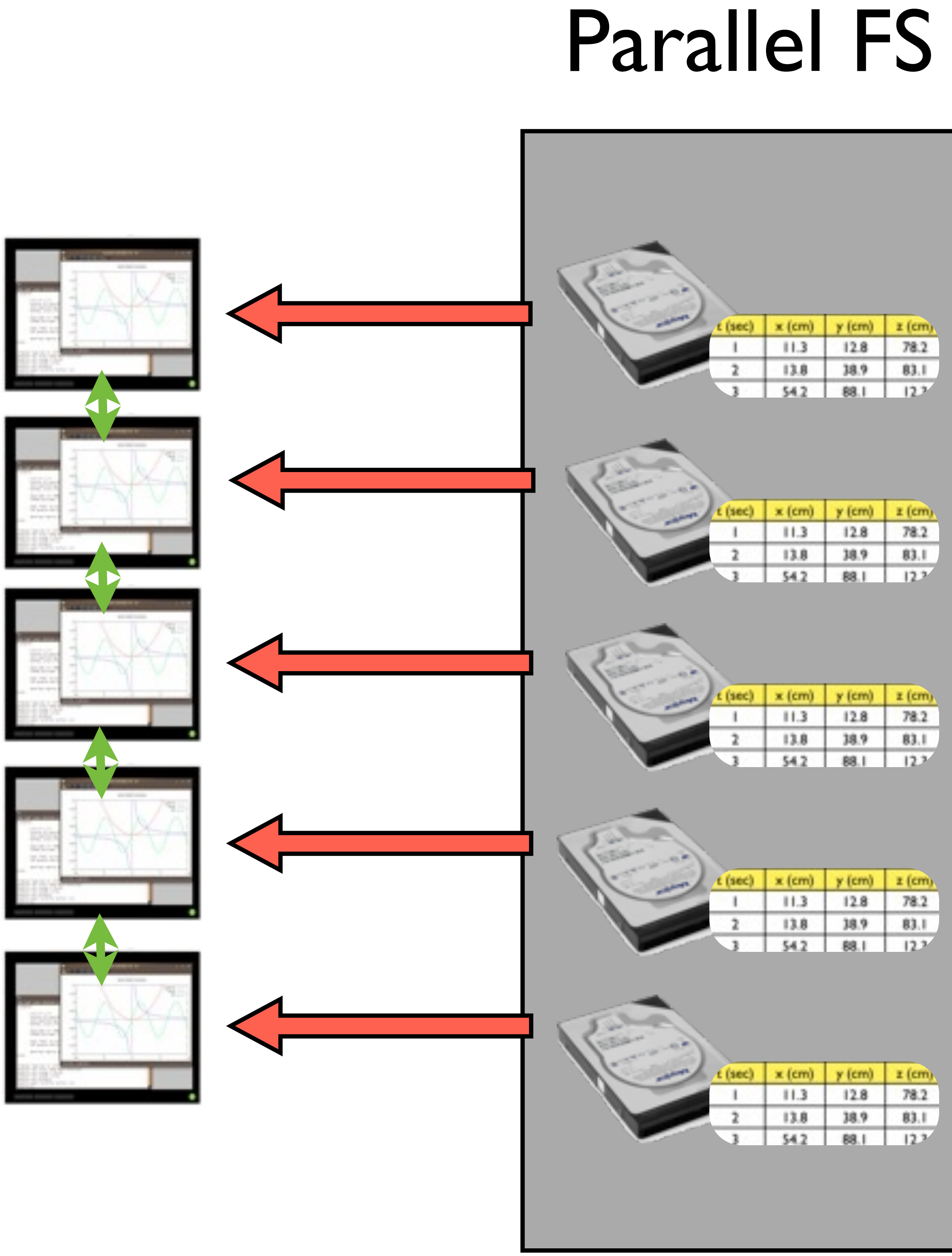
Parallel operations on separate data

- Or can do truly parallel operations
  - multiple clients doing independent work
- Easy parallelism (good for lots of small data) - process many small files separately



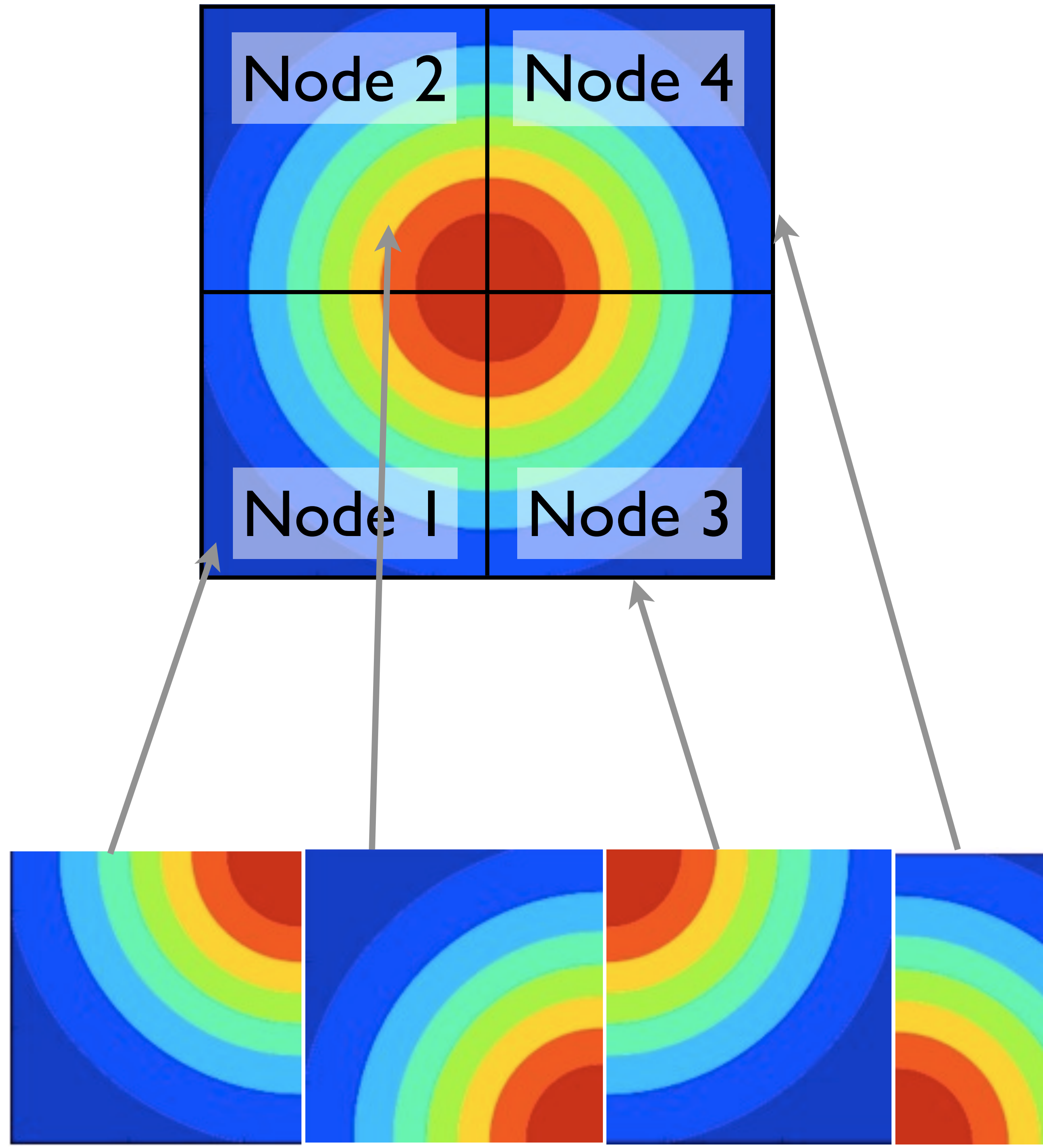
Parallel operations on separate data

- Or can do truly parallel operations
  - multiple clients doing independent work
- Easy parallelism (good for lots of small data) - process many small files separately
- Harder parallelism - each does part of a larger analysis job on a big file.



Data files must take advantage of parallel I/O

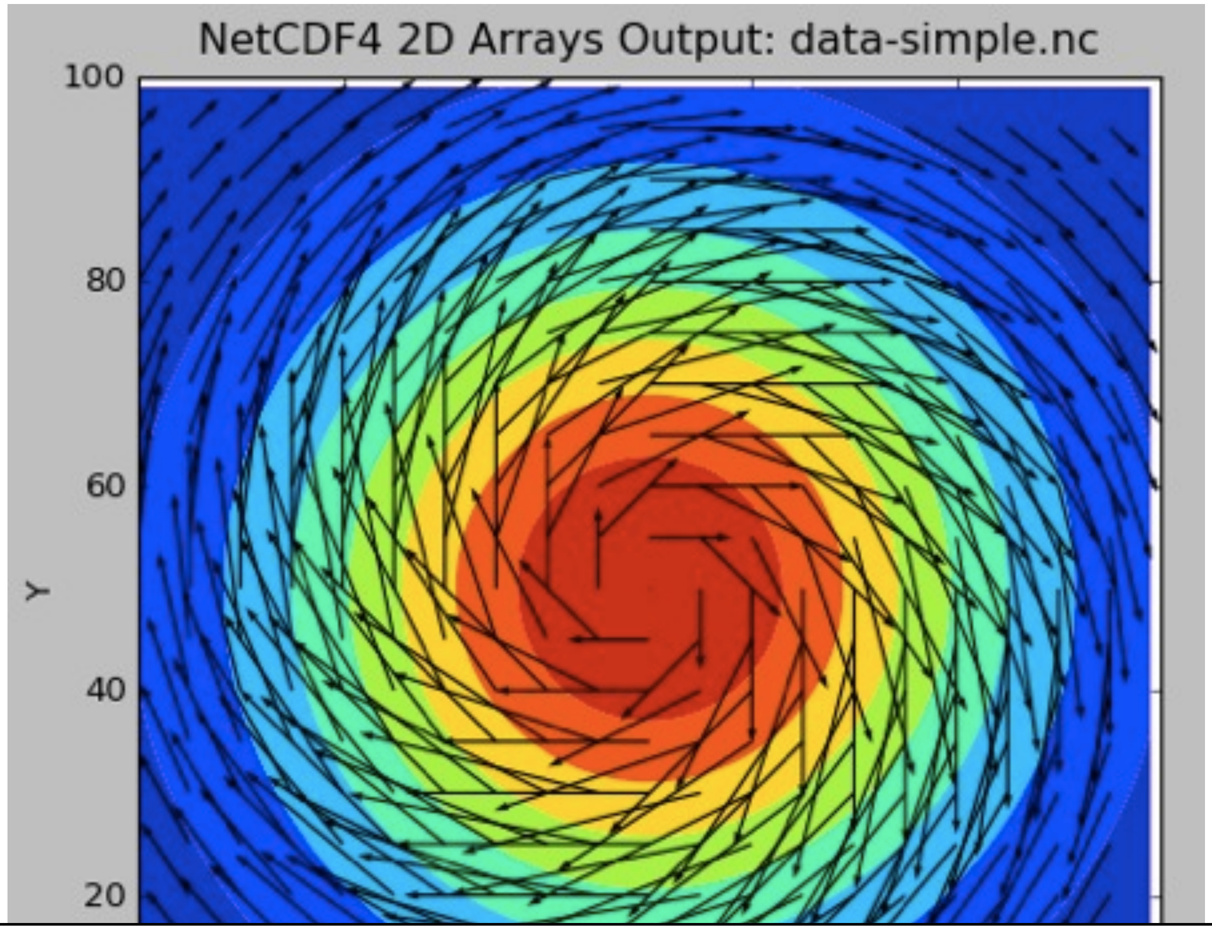
- For parallel operations on single big files, parallel filesystem isn't enough
- Data must be written in such a way that nodes can efficiently access relevant subregions
- HDF5, NetCDF formats typical examples for scientific data



Disk: (HDF5, NetCDF, pVTK..)

These formats are *self-describing*

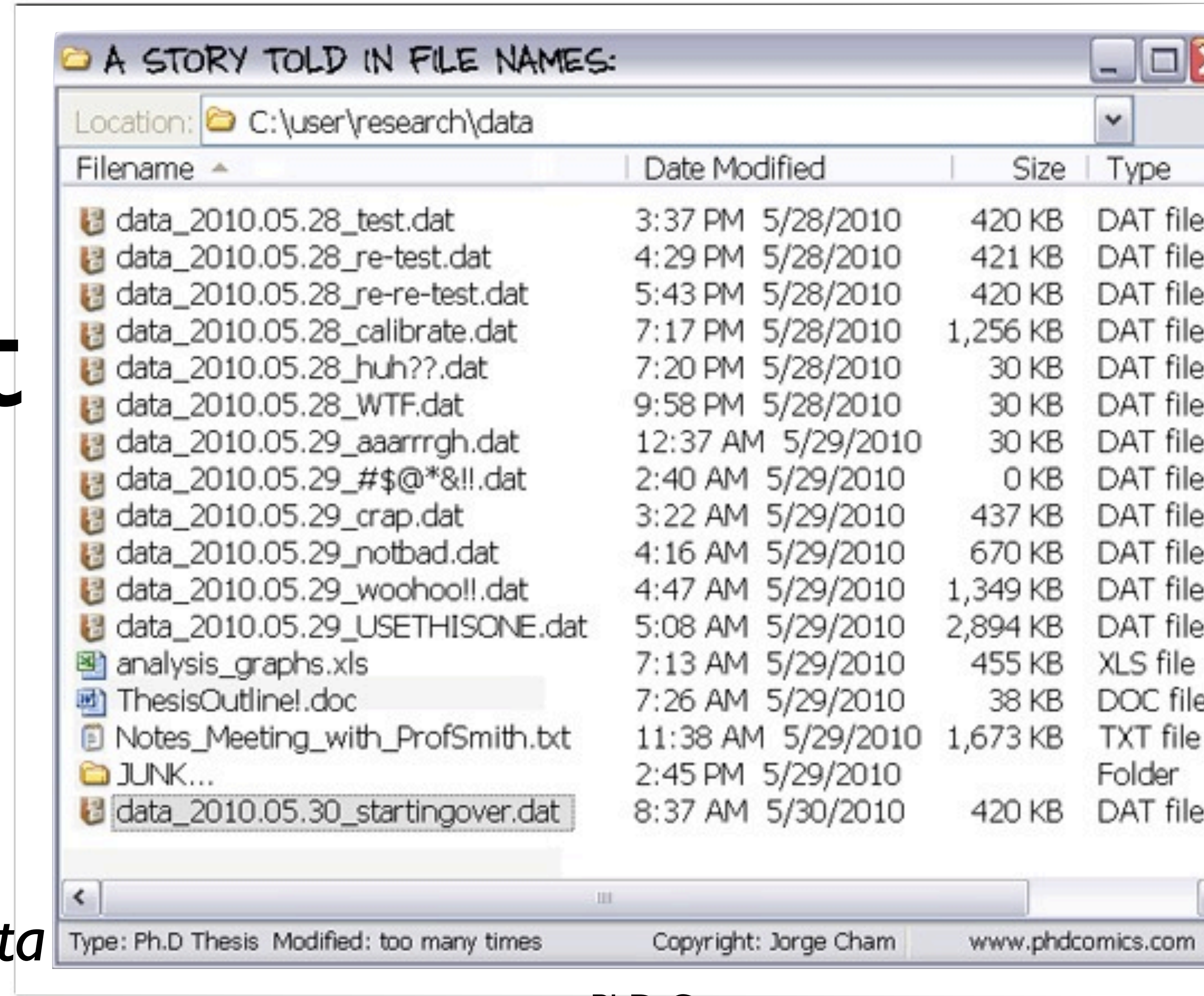
- HDF5, NetCDF have other advantages anyway
  - Binary
  - Self describing - contains not only data but names, descriptions of arrays, etc
  - Many tools can read these formats
- Big data - formats matter




```
$ ncdump -h data-simple-fort.nc
netcdf data-simple-fort {
dimensions:
   X = 100 ;
   Y = 100 ;
   velocity components = 2 ;
variables:
   double Density(Y, X) ;
   double Velocity(Y, X, velocity components) ;
}
```

Data Management

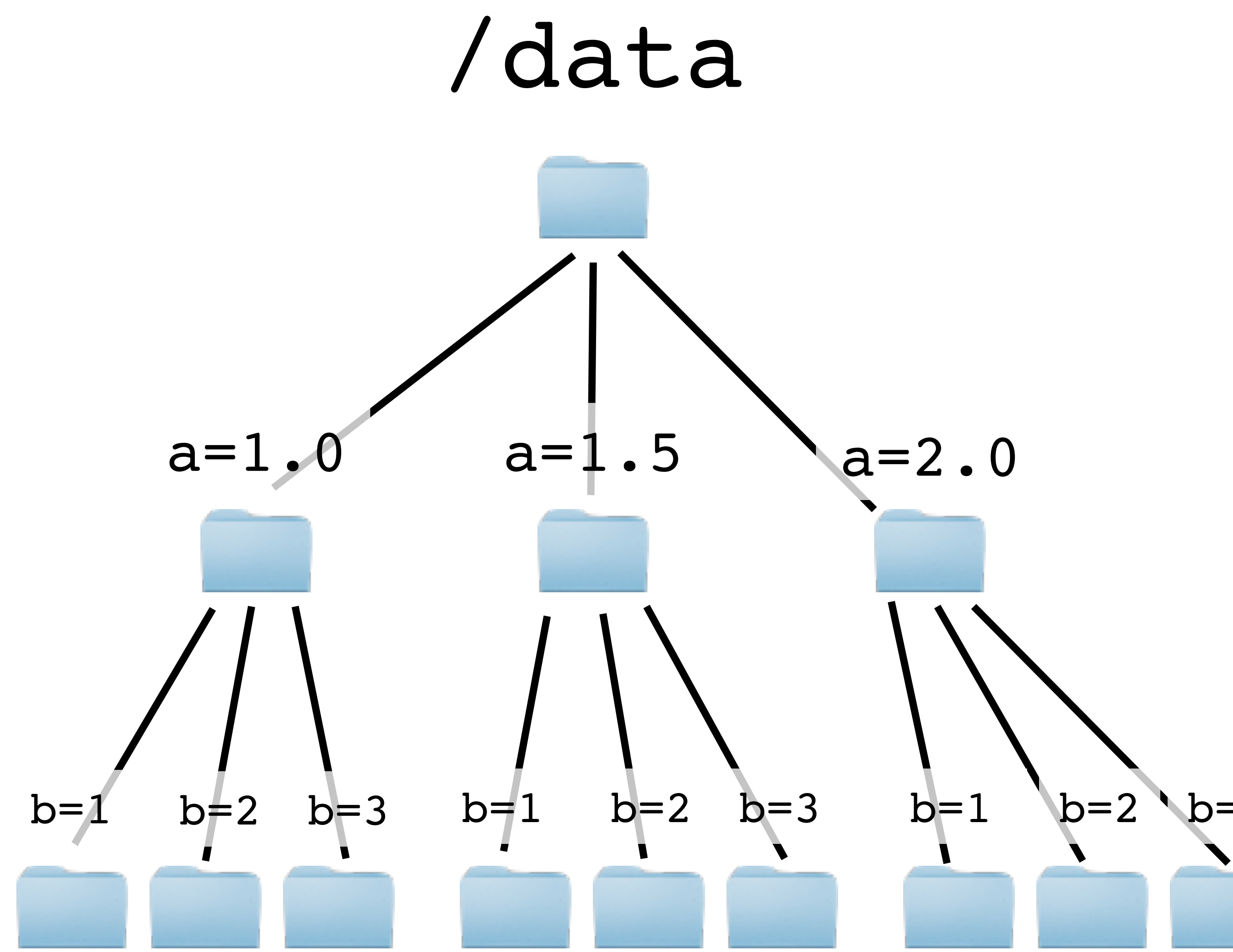
- Human-interpretable file-names lose their charm after few dozen files
- (or even after a few months pass)...
- Rigorously maintained *metadata* becomes essential.
- Also, need to avoid zillions of files in same directory



PhD Comics
http://www.phdcomics.com/comics/archive.php?comicid=1323

Data Management

- Hierarchy of directories work better,
- As long as layout will meet all analysis needs
- Some metadata is included in the directory structure itself
- Avoids tonnes of files in single directory



Databases for science

- Databases?
- Overhead (seeky), but not so bad if chunks in database very large
- Very handy if will be often regrouping the data
    - Don't yet know what relations you will highlight, or
    - Will highlight several overlapping relations

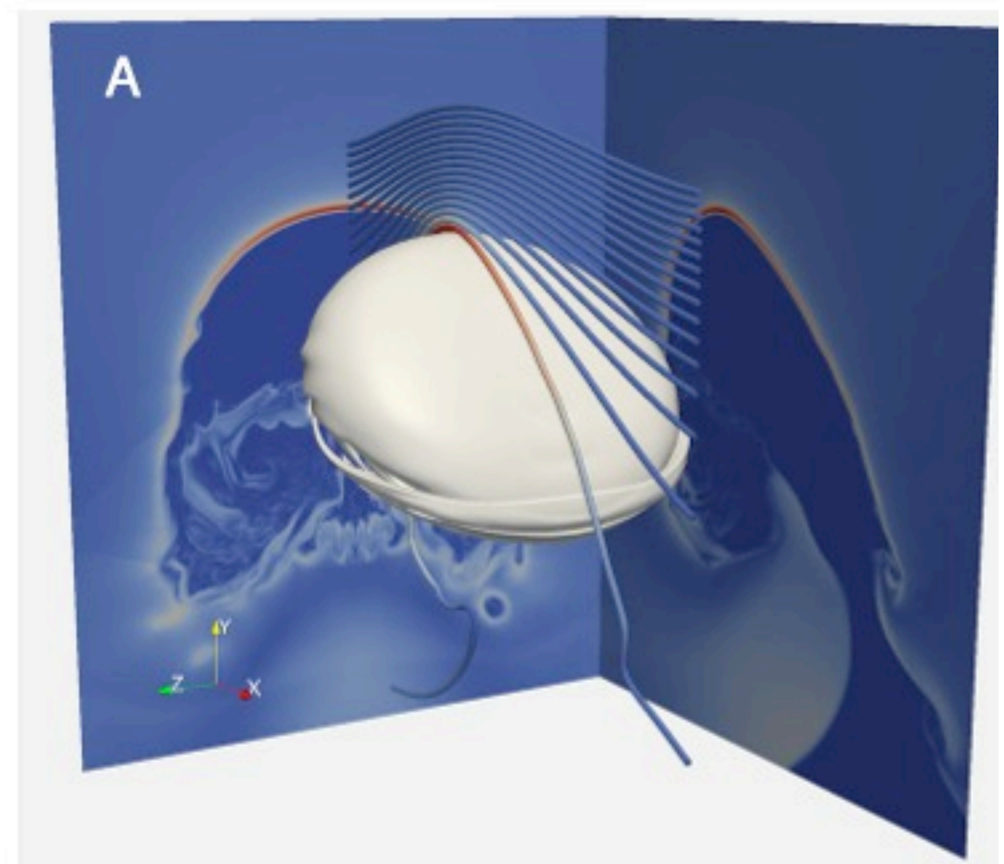
run#	success	size	transport	...
93	no	12k	eth	
1	yes	512	eth	
87	yes	64	ib	
13	no	32	eth	

```
insert into benchmarkruns
values (newrunnum, datestr,
timestr, juliannum)

...
SELECT
nprocs,test,size,transport,mpi
ype,runtime,mopsperproc,run
FROM mpirundata WHERE
(success=1)
```

At scale, *crucial* to track data provenance

- Two of my inputs and one analysis routine changed; do I need to redo this figure?
- How - What steps were involved in making this figure?
- 2 years later, someone questions the result - can I reproduce this key figure with new code, etc?
- Different at scale - more plots, longer to generate

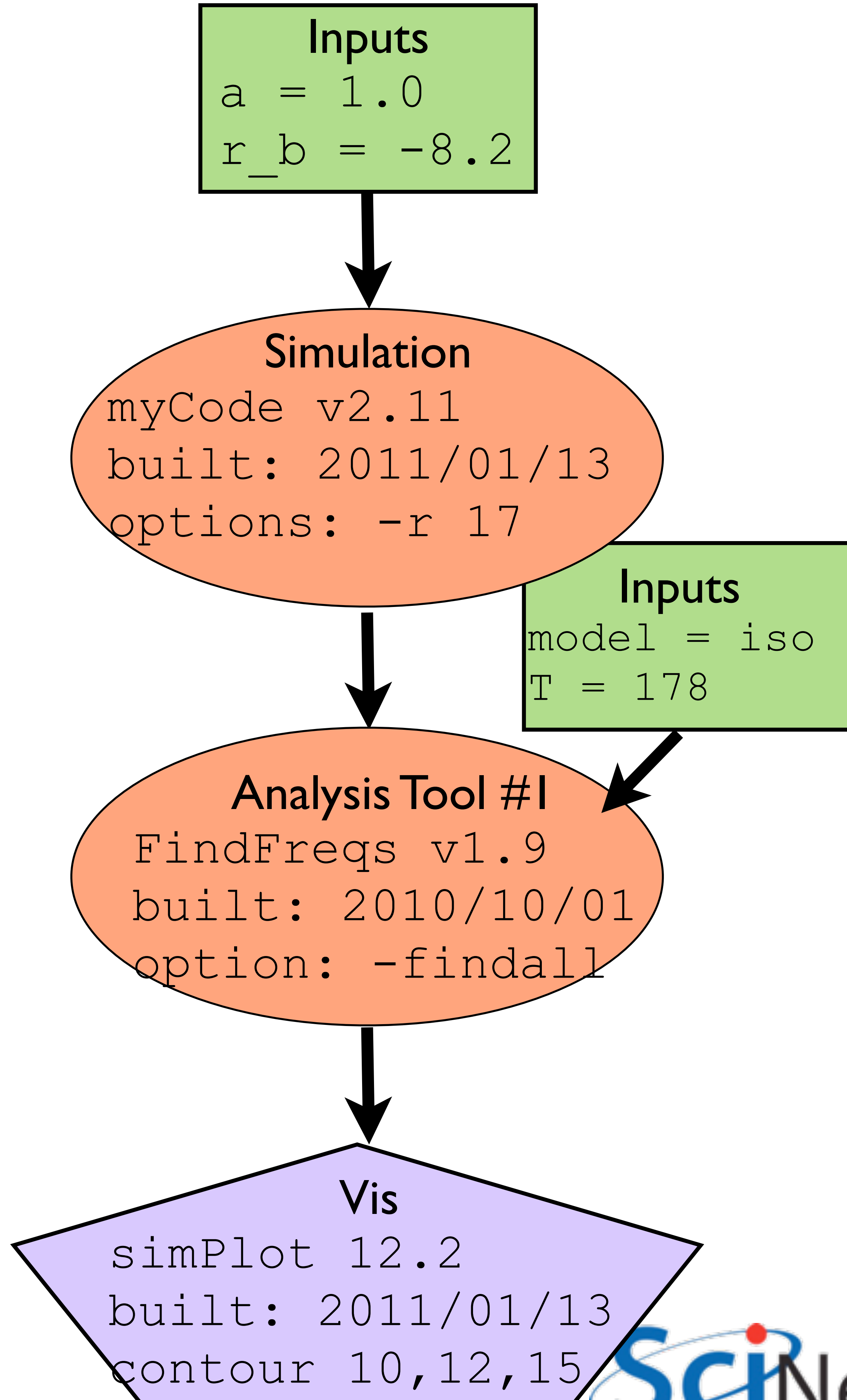


Si science illustrated

SciNet

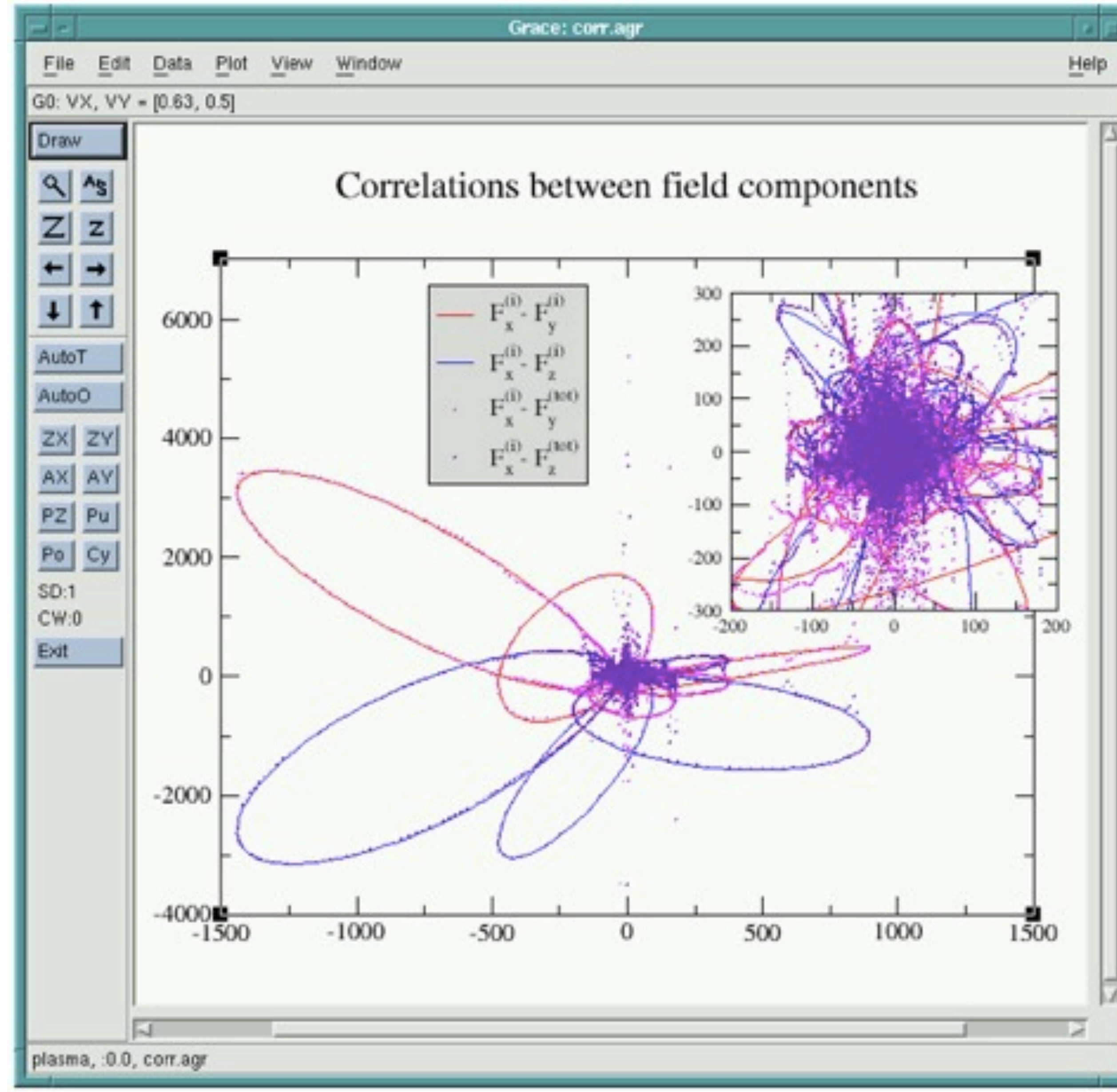
At scale, *crucial* to track data provenance

- Provenance - documentation of origin of a work
- Inputs, tools, steps in process
- Versions, dates, options
- Version control can greatly help you with this
- Then propagate the data along steps, keep it in (say) comment fields
- http://software-carpentry.org/4_0/essays/provenance



Scalability requires Automation

- Need for automation clear when dealing with thousands of small datasets..
- But large sets, too. (Sitting at a GUI for hours while waiting data to load not an option)

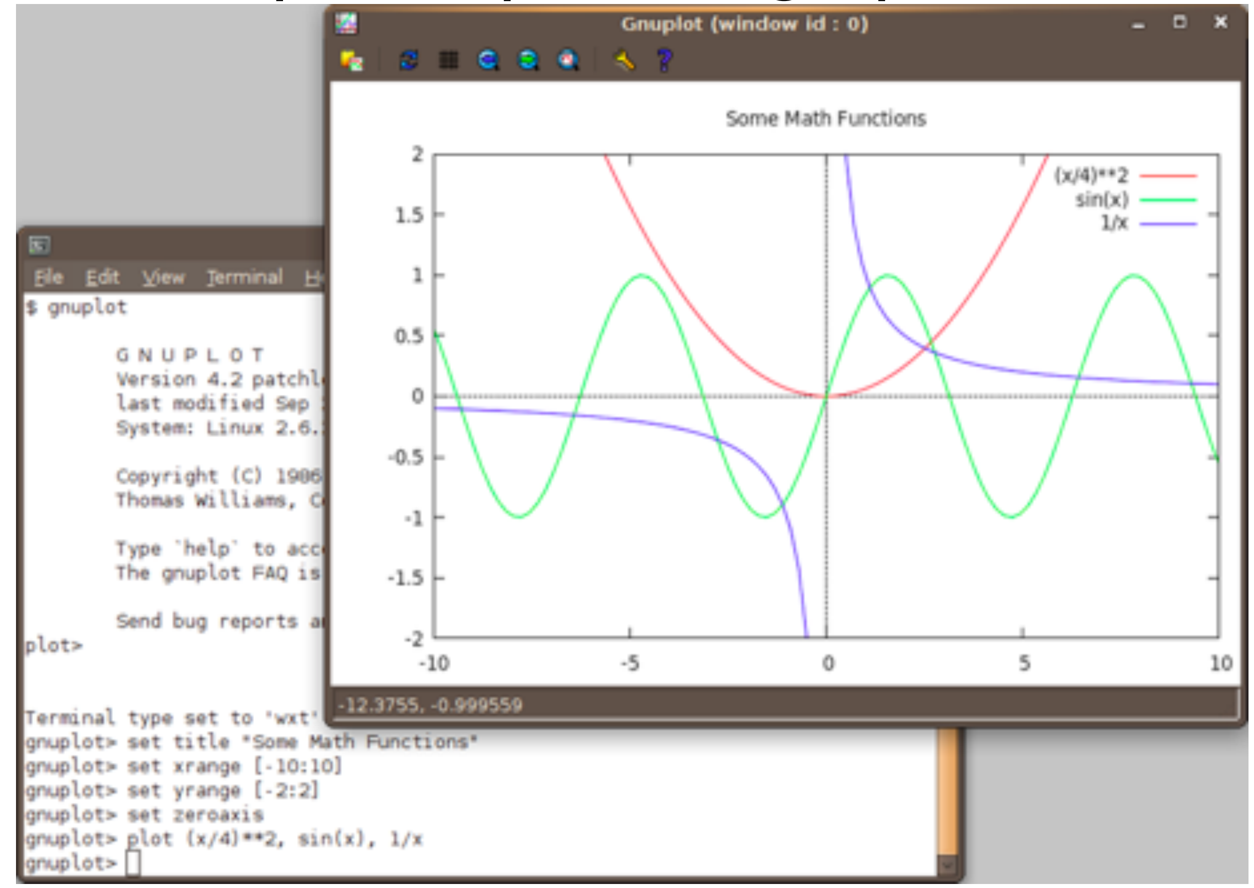


XMGrace,
http://plasma-gate.weizmann.ac.il/Grace/

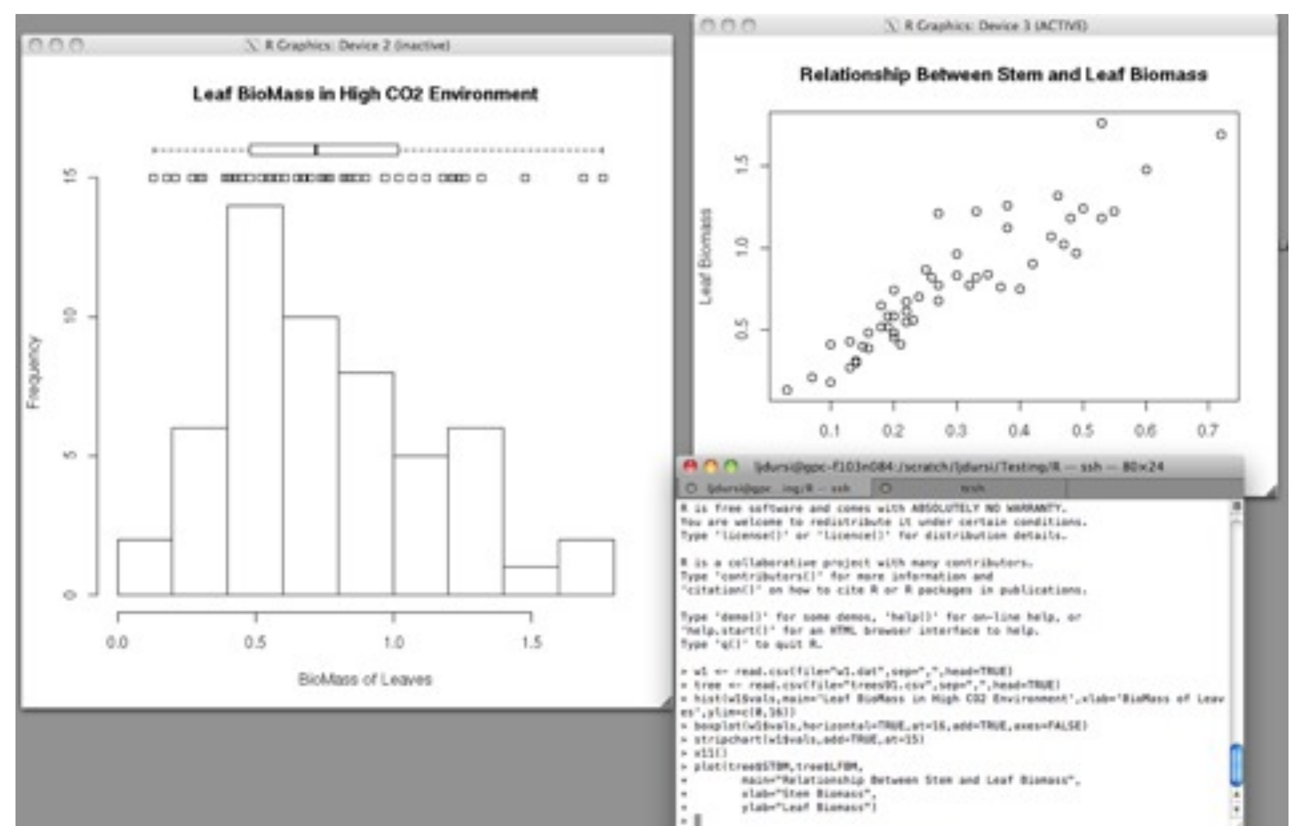
Scalability requires Automation

- Scripting based packages like gnuplot, matplotlib, R...
- Implicitly automatable
- Harder learning curve
- Learning basic Unix shell scripting *priceless* for automation
- http://software-carpentry.org/4_0/shell/

Gnuplot http://www.gnuplot.info



R http://www.r-project.org/

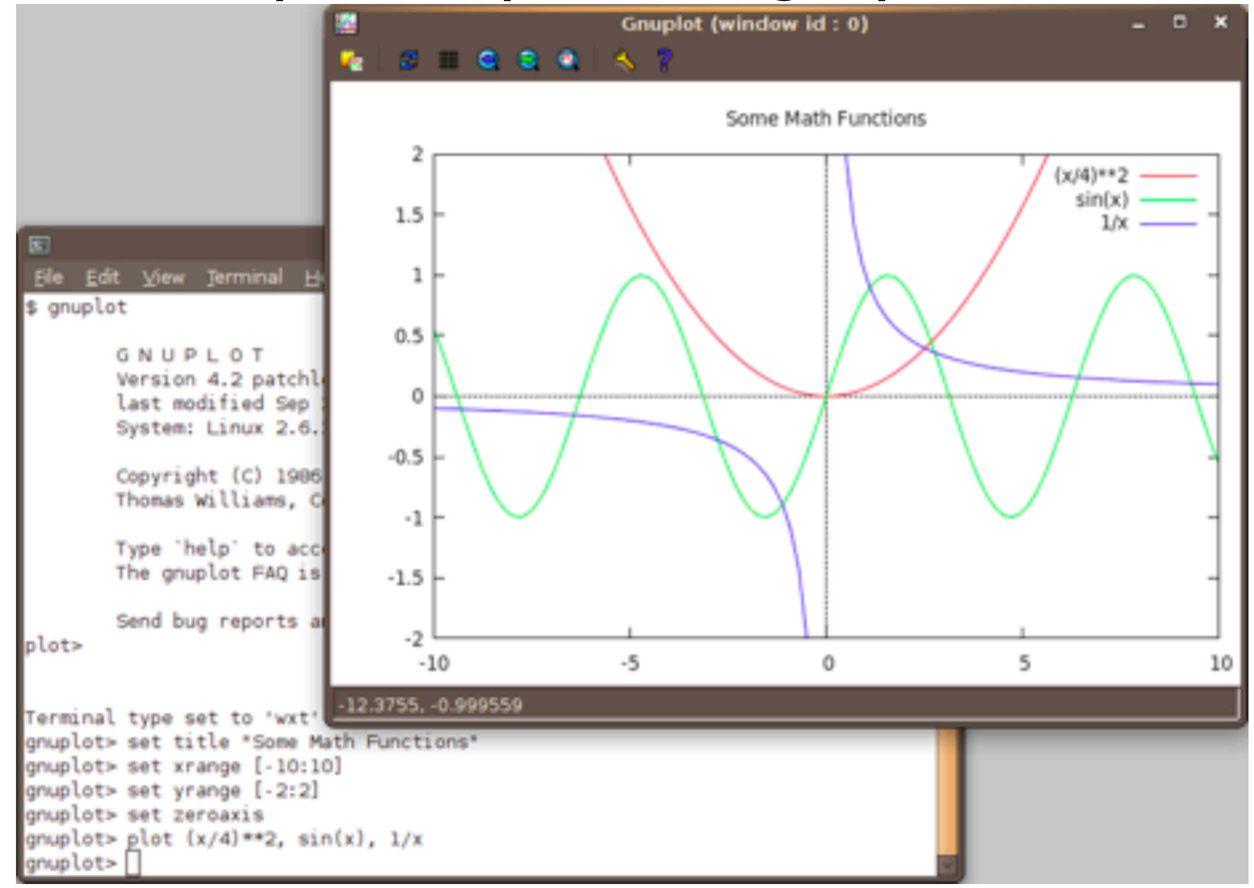


science illustrated

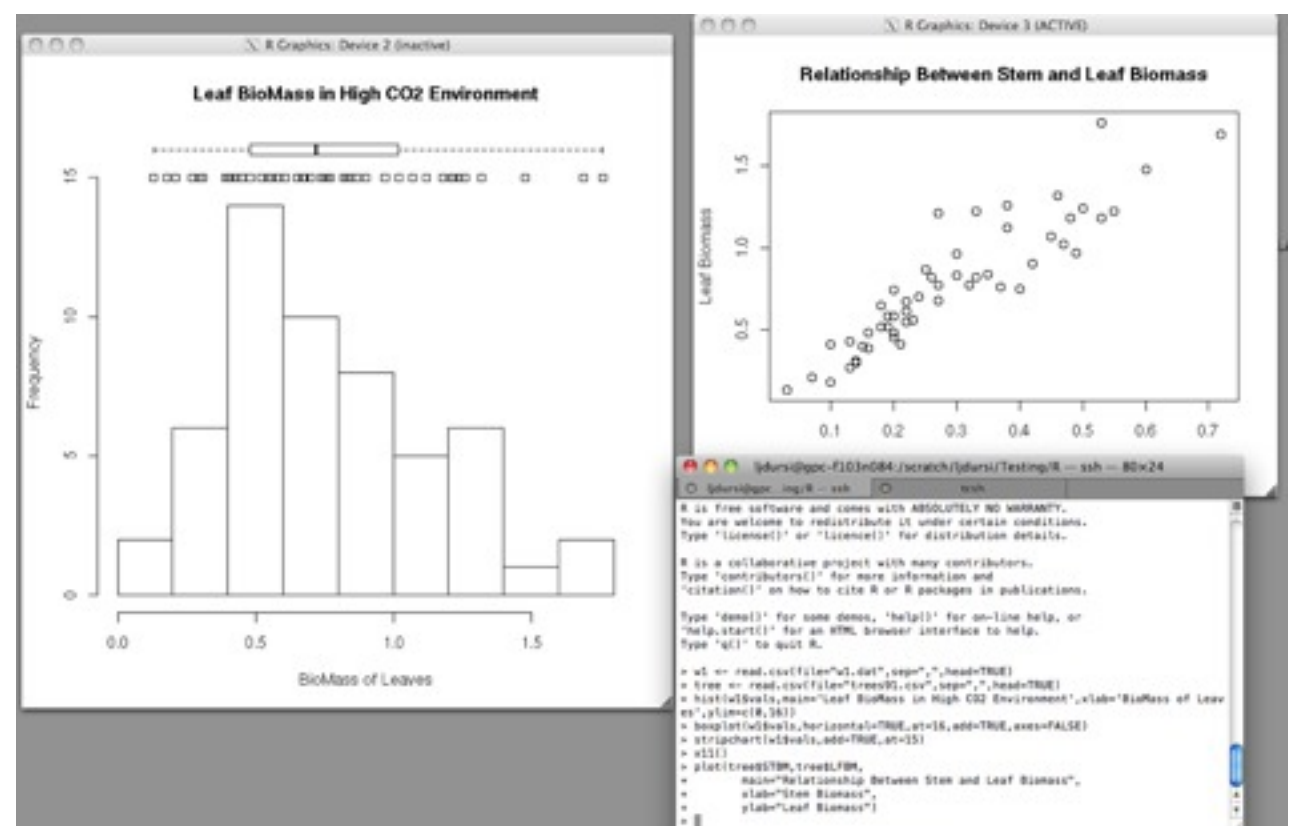
Scalability requires Automation

- Scripting makes processing thousands of files feasible, redoing huge visualizations less tedious
- Provides reproducibility, some form of documentation of process
- Scripts can be kept in version control

Gnuplot http://www.gnuplot.info



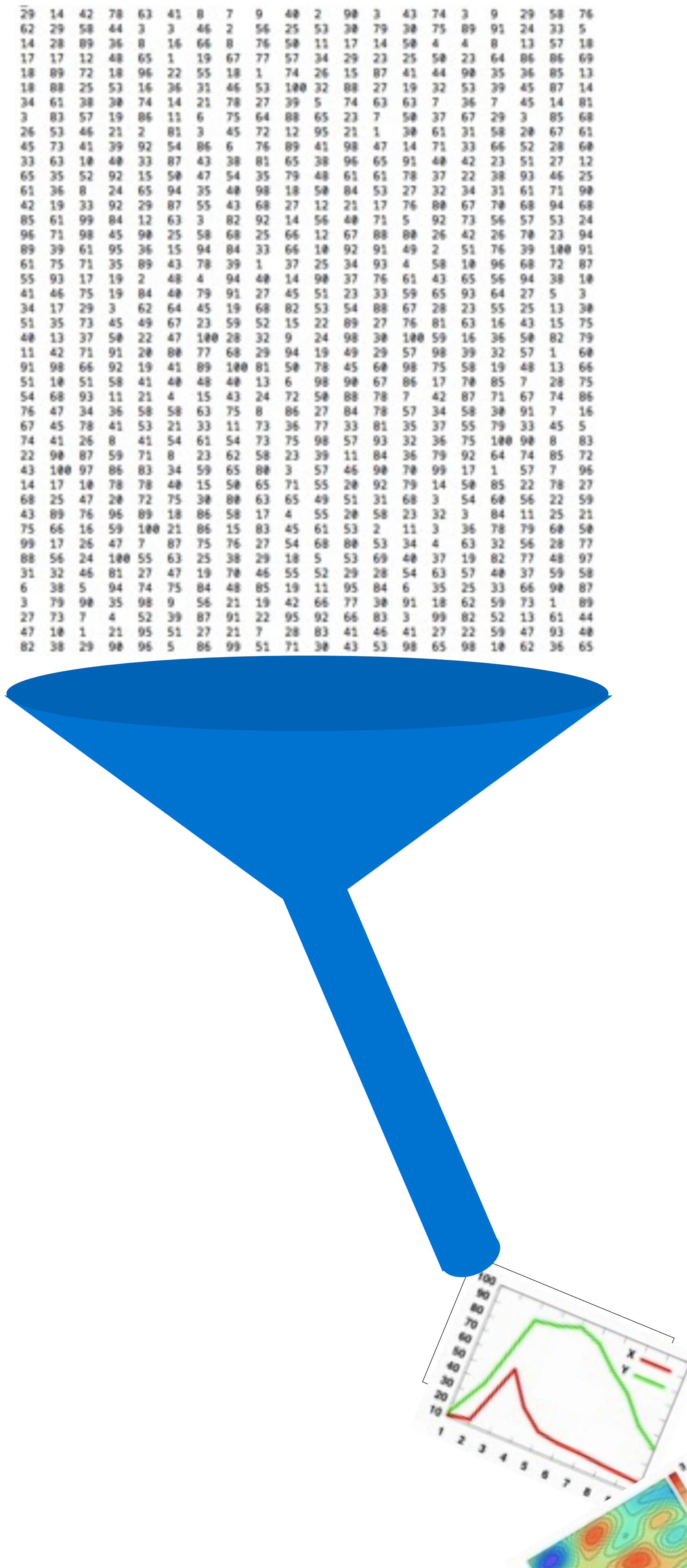
R http://www.r-project.org/



science illustrated

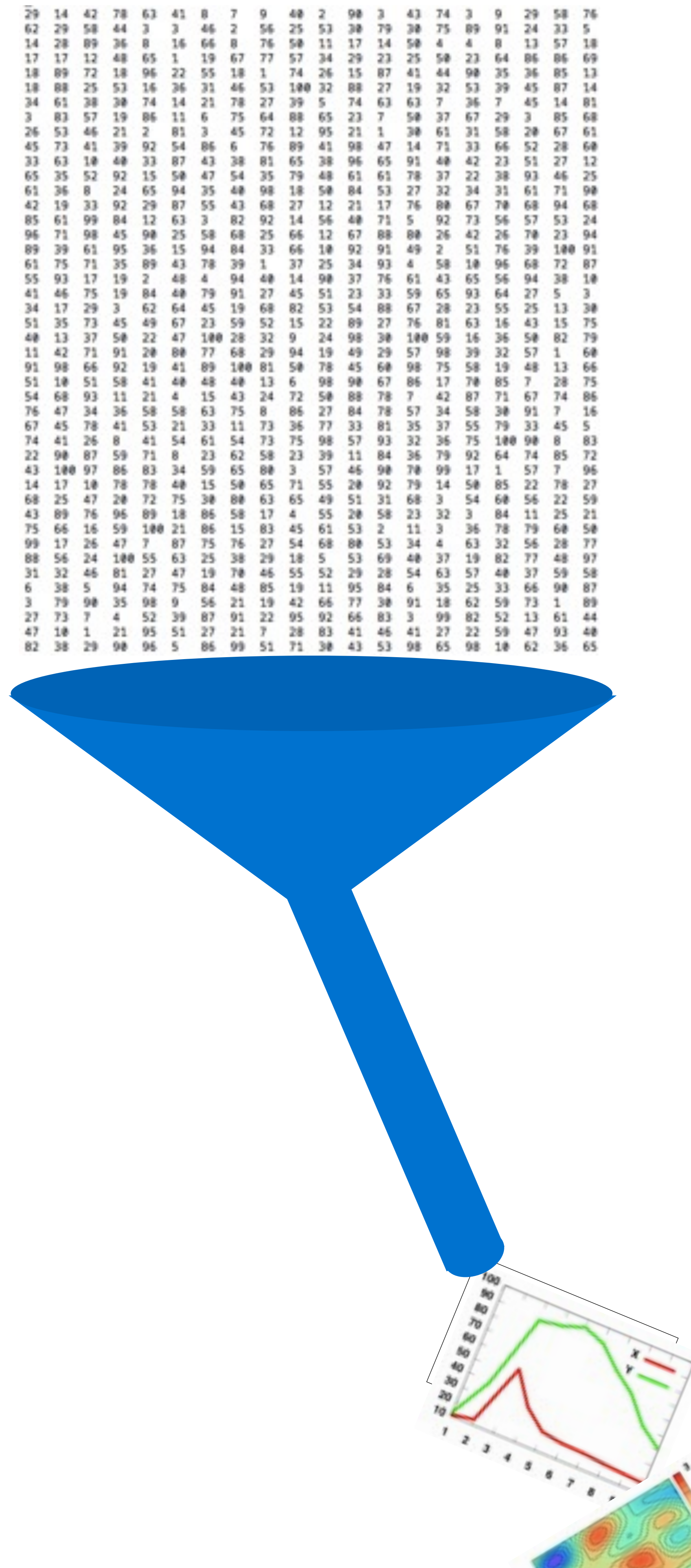
Planning your analysis pipeline

- Start w/ 40 TB, will presumably end with much less
- Do as much of that reduction as early as possible in the process
    - Average, bin, integrate, combine, contour
- Automate everything
- Easier if you know exactly what you'll be doing



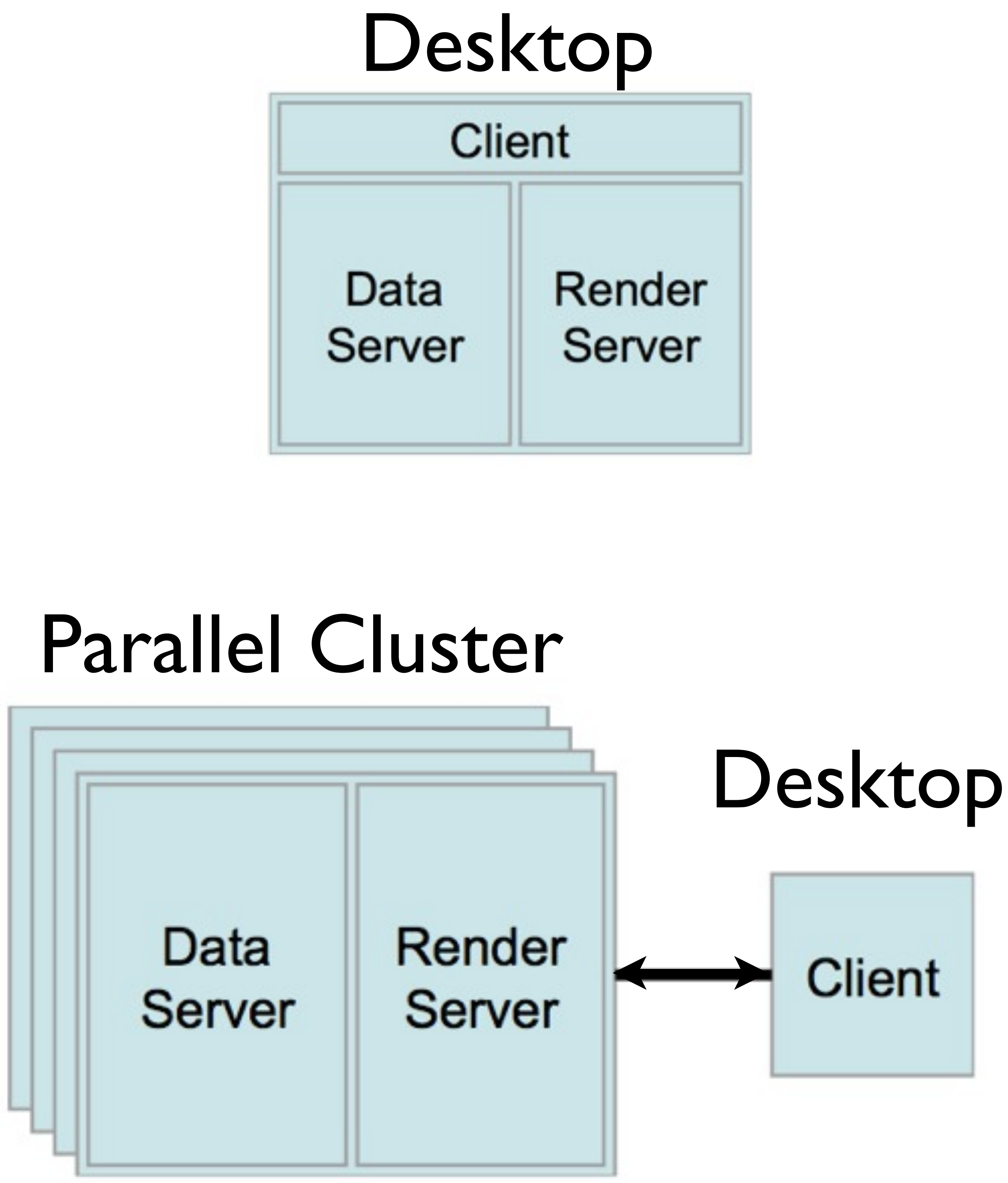
Planning your analysis pipeline

- Don't generate dozens of TB of data without knowing what you're going to do with it!
- Start small
- Explore on smaller data, or subsets



Scalability may require parallel

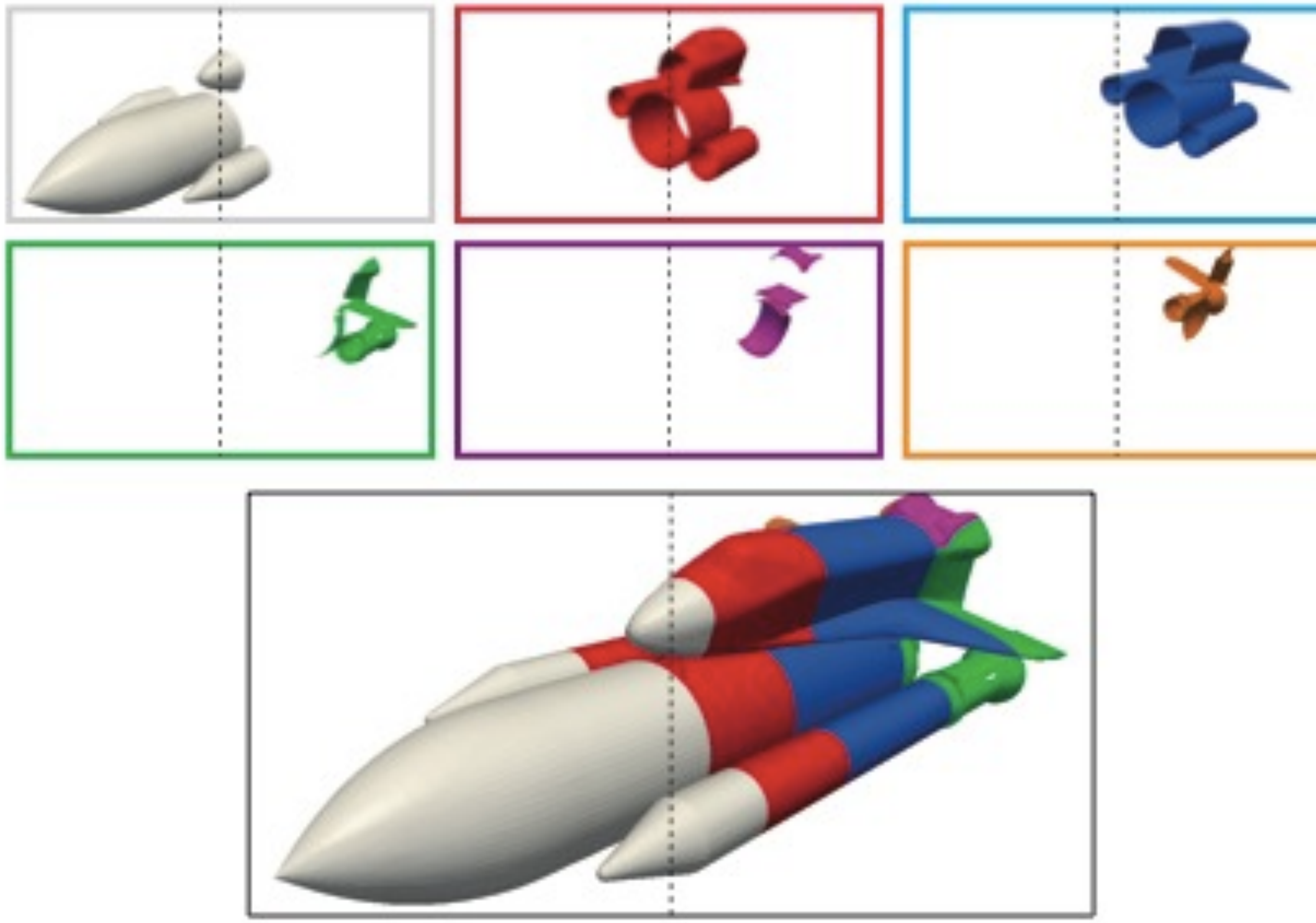
- Scalable visualization packages
  - eg, ParaView, VisIT
- Client/Server model
- On desktop, client/server coexist
- Or, servers can be **many** nodes on cluster!



Paraview Tutorial, http://paraview.org/

Parallel Visualization

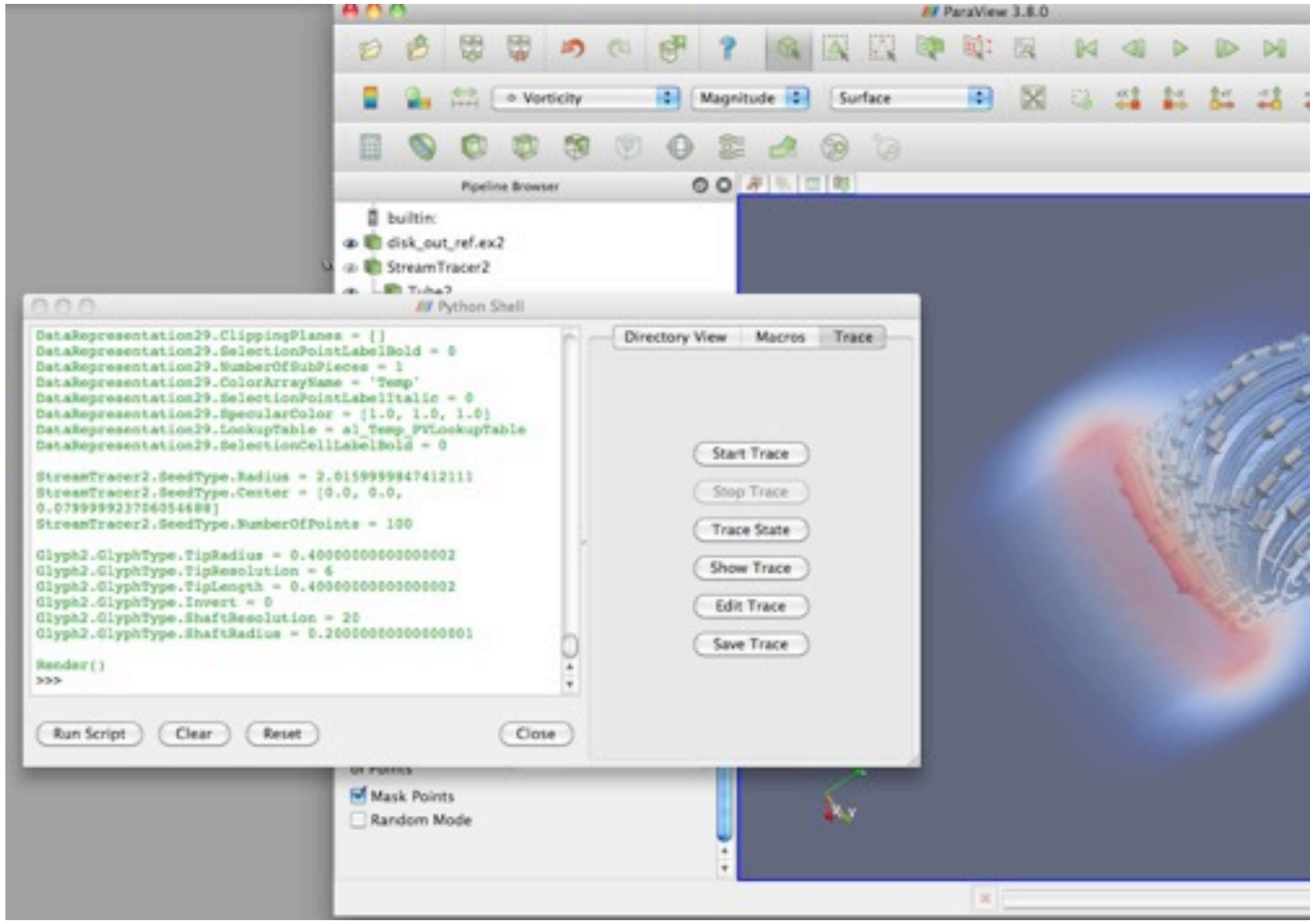
- Decompose data onto many processors (need parallel file systems, format!)
- Each processor generates its portion of the geometry, or image
- Composited en route to client
- Can control visualization interactively from desktop



IceT users guide,
http://www.cs.unm.edu/~kmorel/IceT/

Remains scriptable

- Even in client/server mode, these tools are scriptable and automatable
- Client end can be run as a script
- Client coordinates all communication with server nodes
- Can also be run without GUI, pure batch mode on cluster.



Paraview, http://paraview.org/

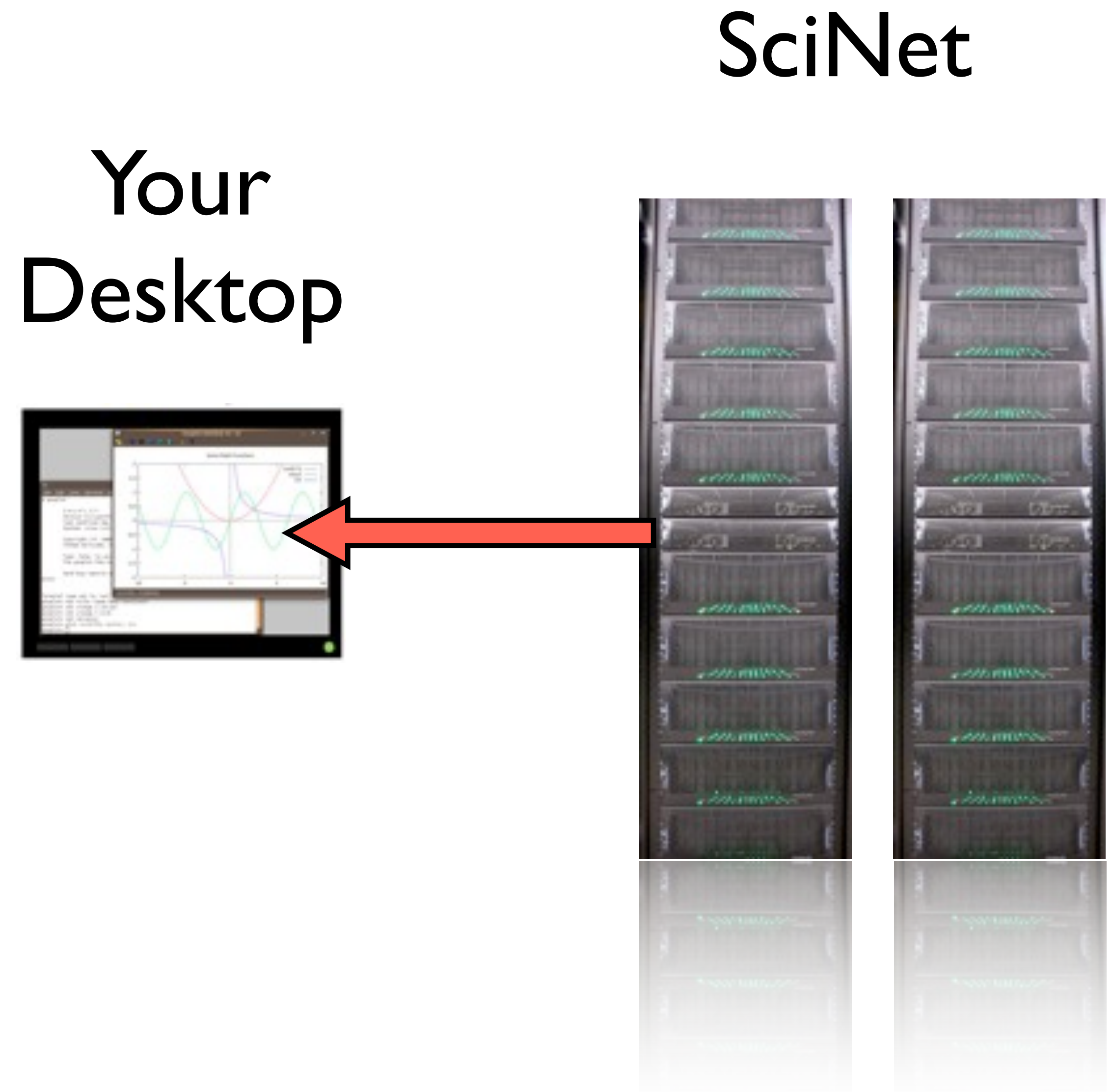
Highly Scalable Parallel Visualization

- Can work to extremely large scales
- VisIt - 4 trillion zone simulations
- ParaView - billions of polygons/sec

Red RoSE visualization cluster (credit: Sandia Nat'l Lab)

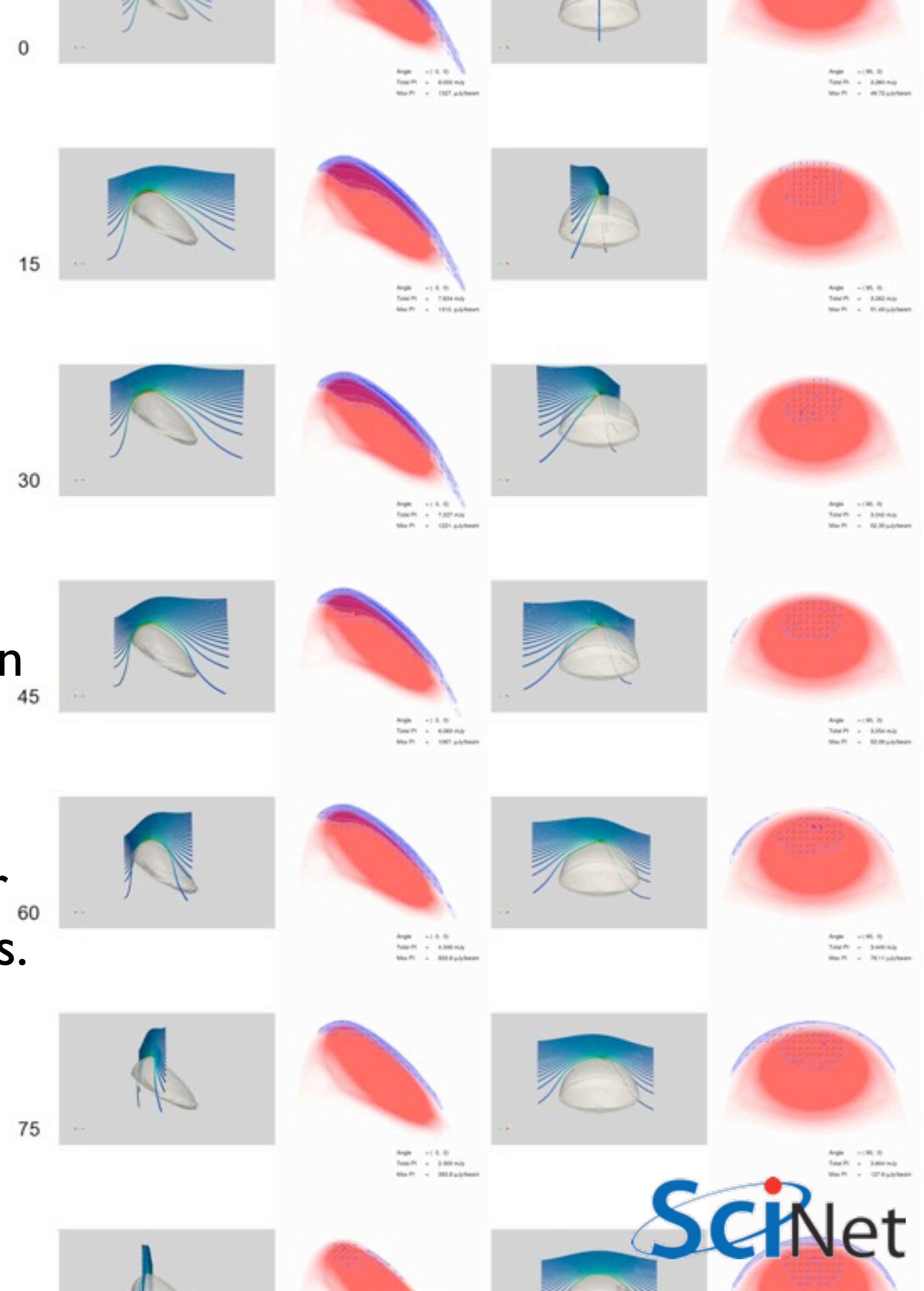
Paraview on SciNet

- Launch job on cluster
- Module load paraview
- Mpirun paraview server
- Interactive? Connect your desktop client
- Run visualization



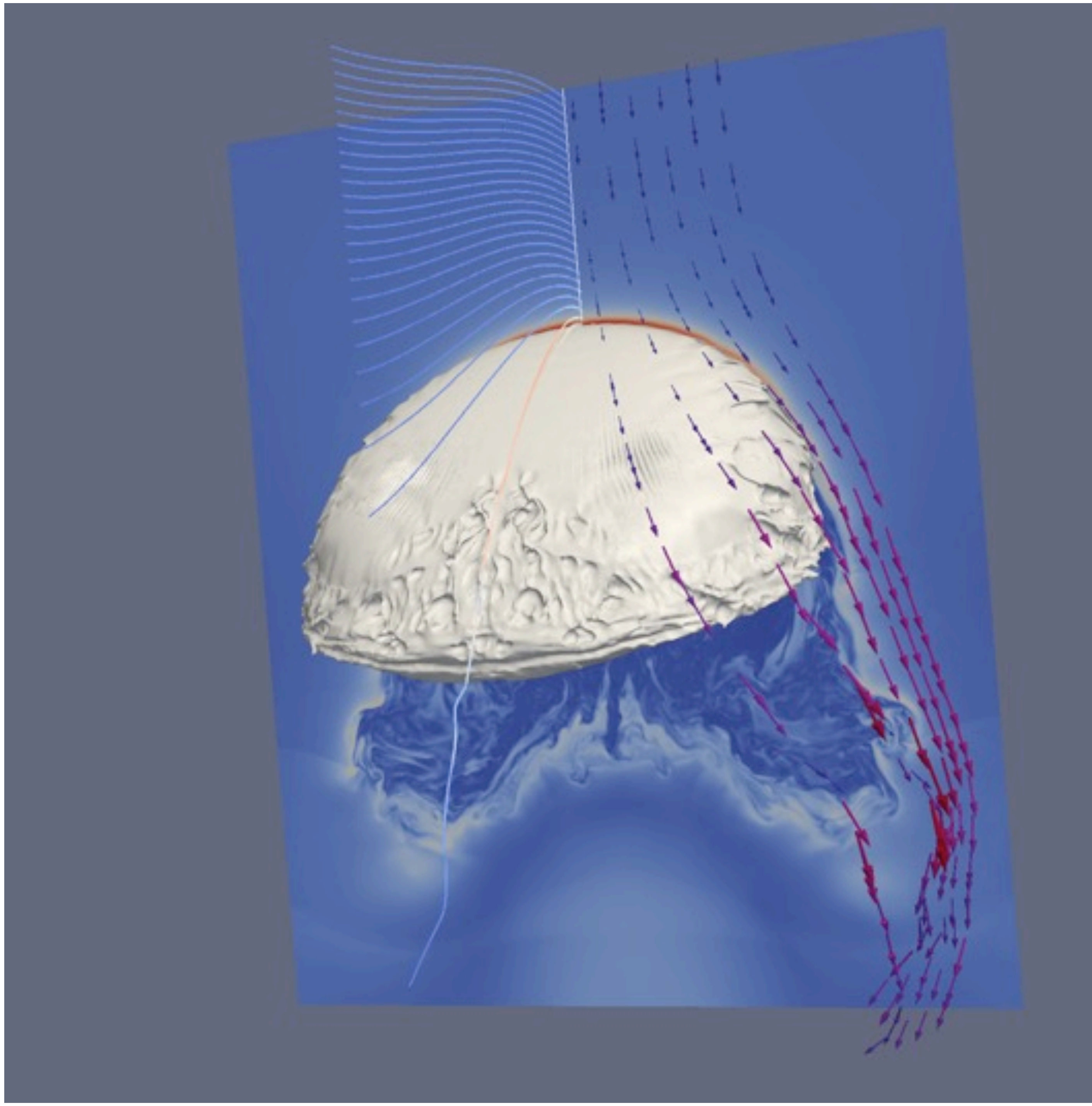
Paraview on SciNet

- One of first simulations run on SciNet's GPC
- Full data set was about 40 TB
- Included collections of smaller simulations and big simulations.



Paraview on SciNet

- High resolution simulation
- 2.5 billion zones
- 330,000 cpu hours
- 256 processors just to load, visualize data



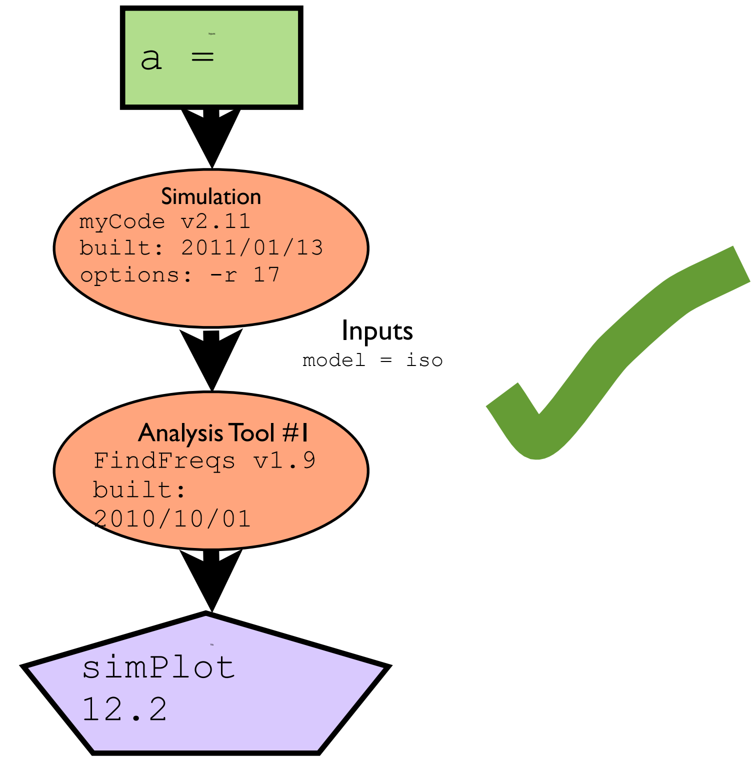
Plan I/O carefully

- Binary files
- As much as possible, large files
- File formats that can be read in parallel, subregions extracted
- Avoid zillions of files
- Especially avoid zillions of files in single directory

010101001111

Data management must scale

- Include metadata for provenance
  - Reduce need to re-do
- Sensible data management
  - Hierarchy of data directories only if that will always work
  - Data bases, formats that allow metadata



?

Use scalable, automatable tools.

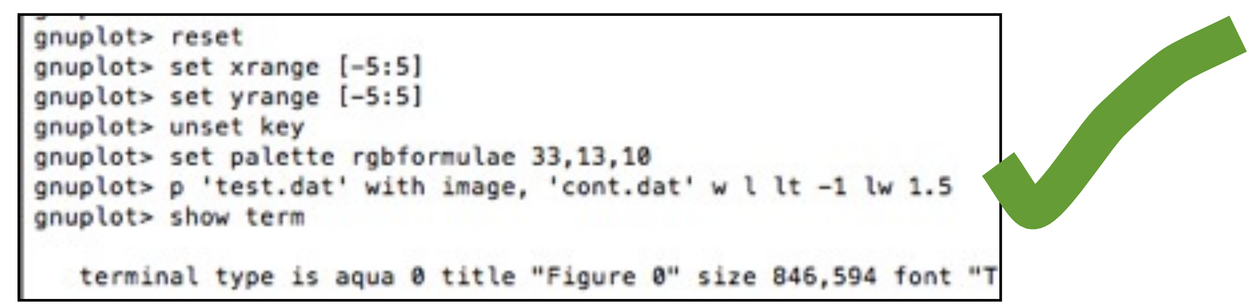
- For large data sets, parallel tools - ParaView, VisIt, etc
- Scriptable tools - gnuplot, python, R, ParaView/Visit...
- Scripts provide reproducability!

```
gnuplot> reset
gnuplot> set xrange [-5:5]
gnuplot> set yrange [-5:5]
gnuplot> unset key
gnuplot> set palette rgbformulae 33,13,10
gnuplot> p 'test.dat' with image, 'cont.dat' w l lt -1 lw 1.5
gnuplot> show term

   terminal type is aqua 0 title "Figure 0" size 846,594 font "T
```

Carefully plan your workflow

- Reduce data as early as possible in the process
  - Average, integrate, combine, contour
- Automate everything



```
gnuplot> reset
gnuplot> set xrange [-5:5]
gnuplot> set yrange [-5:5]
gnuplot> unset key
gnuplot> set palette rgbformulae 33,13,10
gnuplot> p 'test.dat' with image, 'cont.dat' w l lt -1 lw 1.5
gnuplot> show term

   terminal type is aqua 0 title "Figure 0" size 846,594 font "T
```

Thanks to

- Mubdi Rahman, for putting this all together
- Ramses van Zon, Scott Northrup, Danny Gruner - importance of I/O