Scientific Data Visualization

Dave Pugmire

Oak Ridge National Laboratory

Oak Ridge National Laboratory

OLCF Oak Ridge Leadership Computing Facility

“Who are you?”

“Who are you?”

Computer Scientist

Visualization Task in the Oak Ridge National Laboratory’s Leadership Computing Facility

OAK RIDGE National Laboratory

OLCF Oak Ridge Leadership Computing Facility

“Who are you?”

Computer Scientist

Visualization Task in the Oak Ridge National Laboratory’s Leadership Computing Facility

“Who are you?”

Computer Scientist

Visualization Task in the Oak Ridge National Laboratory’s Leadership Computing Facility

Areas of recent work:

- Vector field analysis
- Very large data
- Fusion
- Astrophysics
- Radiation Transport

What is Scientific Visualization?

Images are *not the point*!

Images are *not the point*!

The purpose of computing is insight, not numbers.

– Richard Hamming (1962)

Images are *not the point*!

The purpose of computing is insight, not numbers.

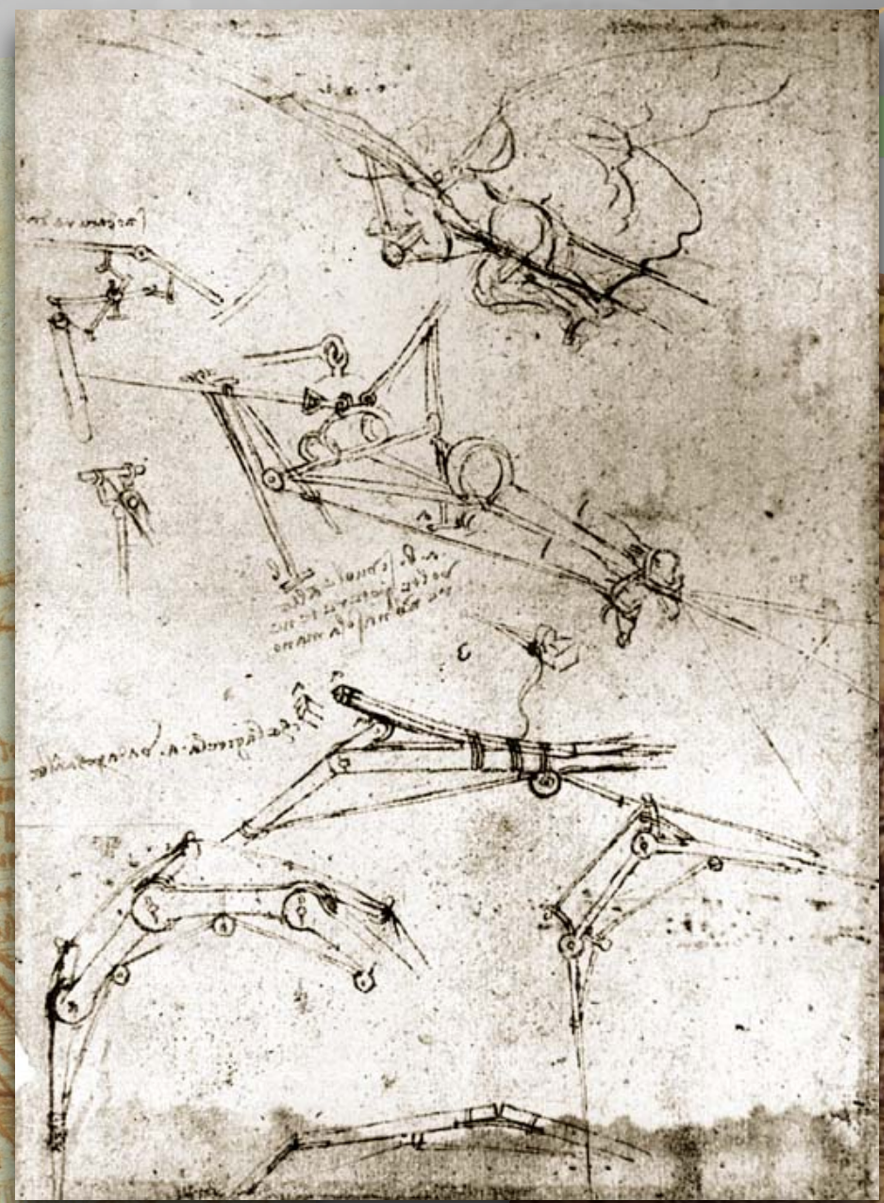
– Richard Hamming (1962)

The most exciting phrase to hear in science, the one that heralds new discoveries, is not 'Eureka!' (I found it!) but **'That's funny ...'**

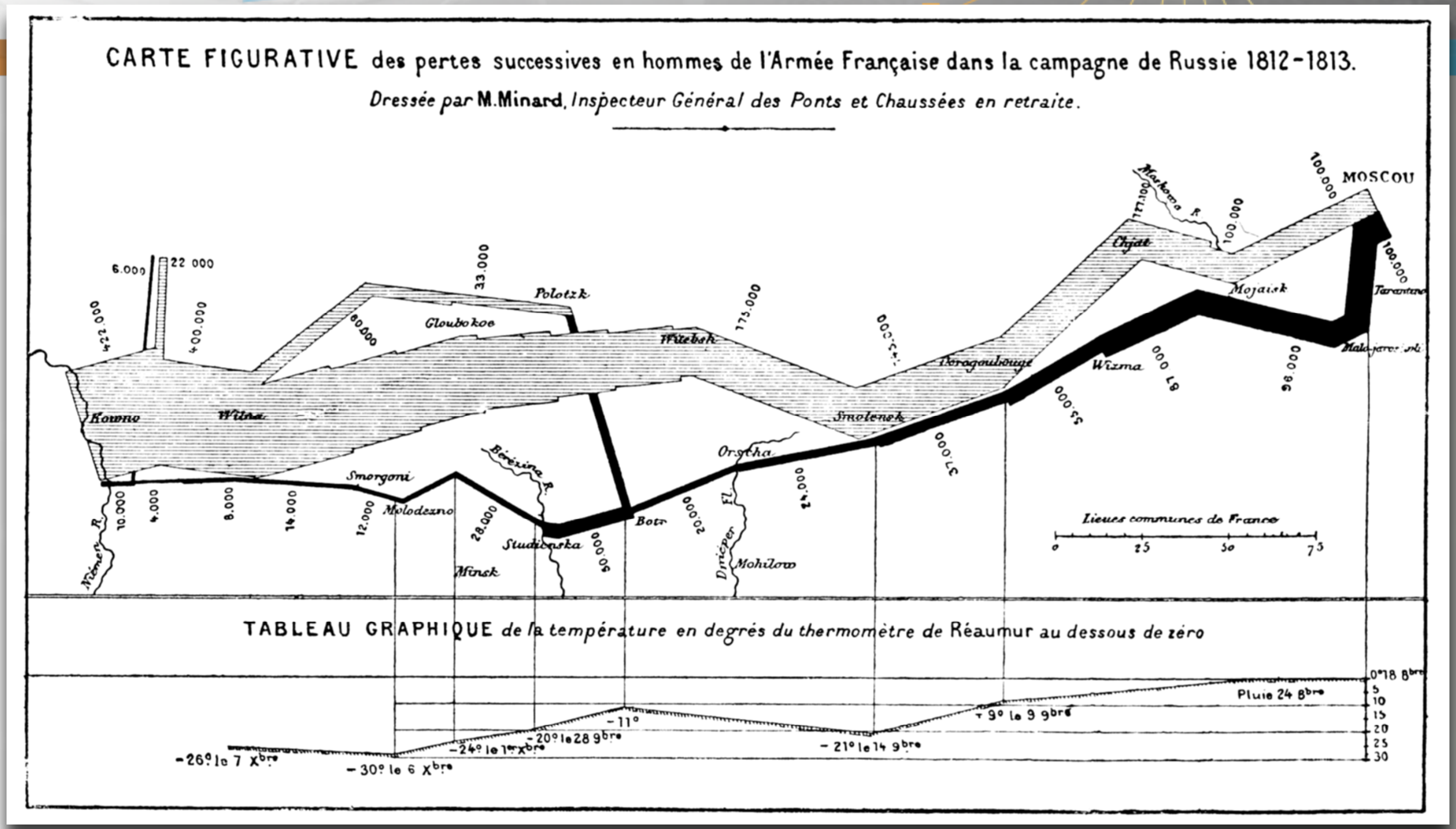
– Isaac Asimov

What is Visualization?

What is Visualization?



What is Visualization?

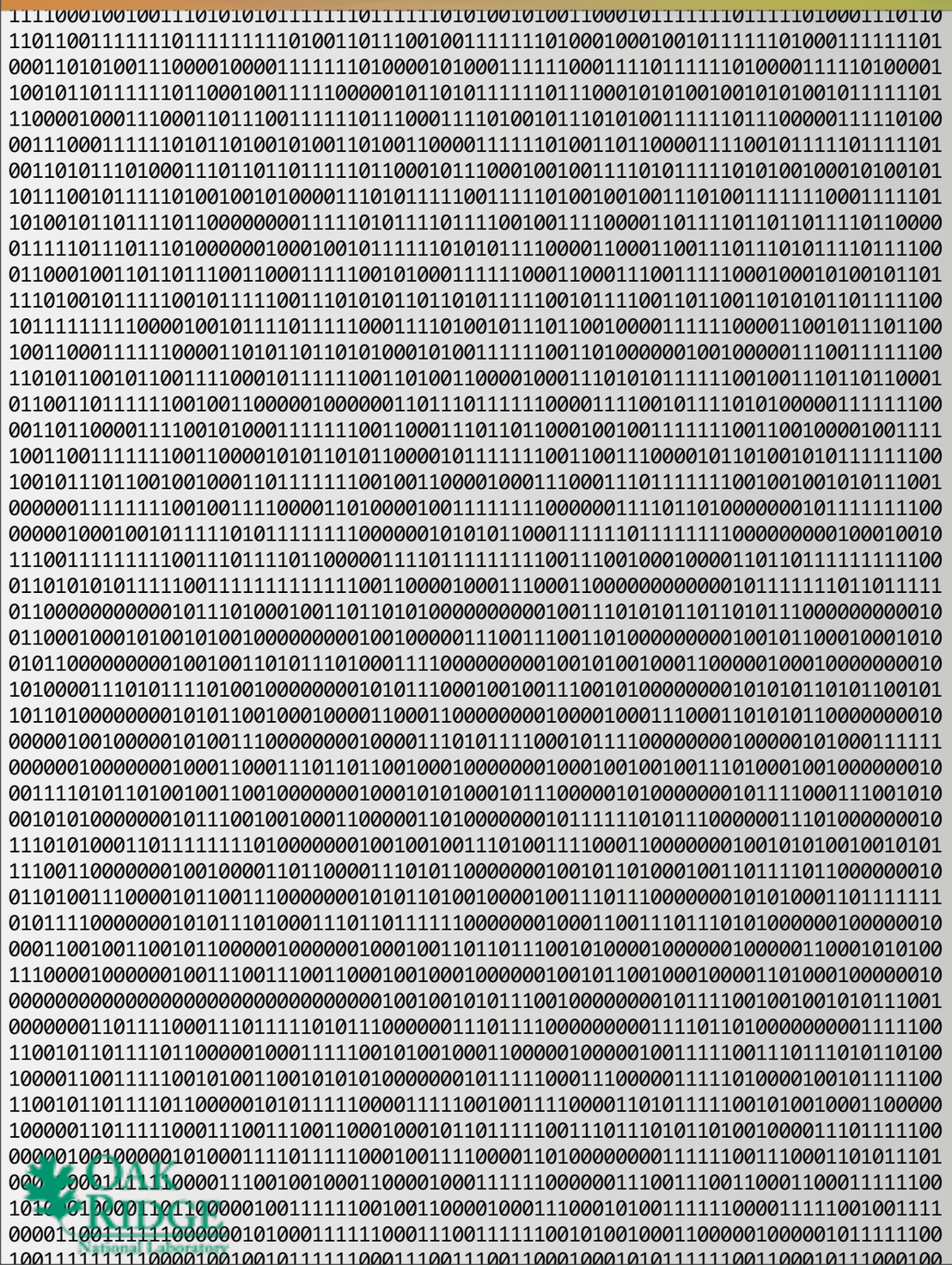


What is Scientific Visualization?

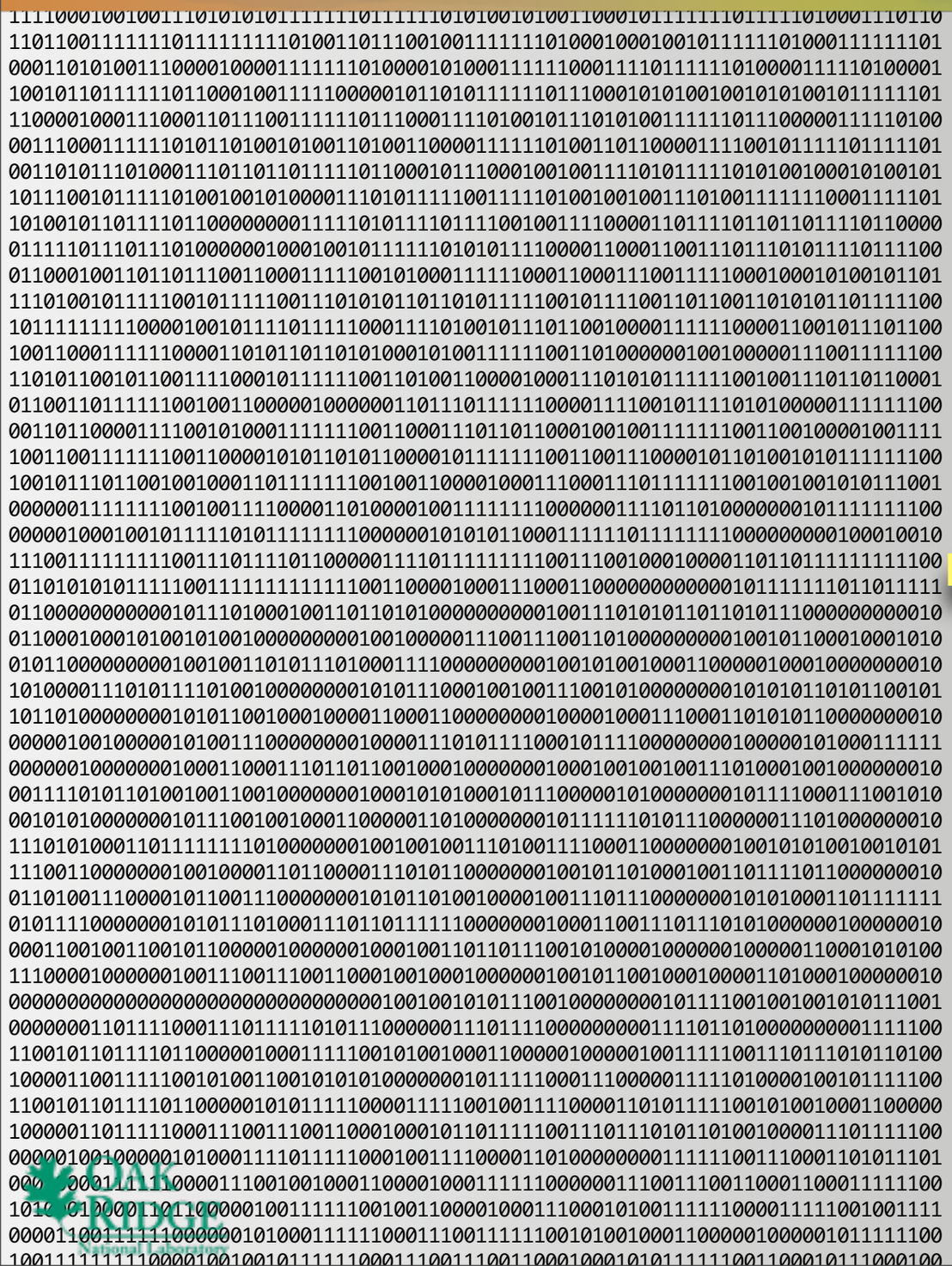
- Visually extracting meaning from a scientific data set

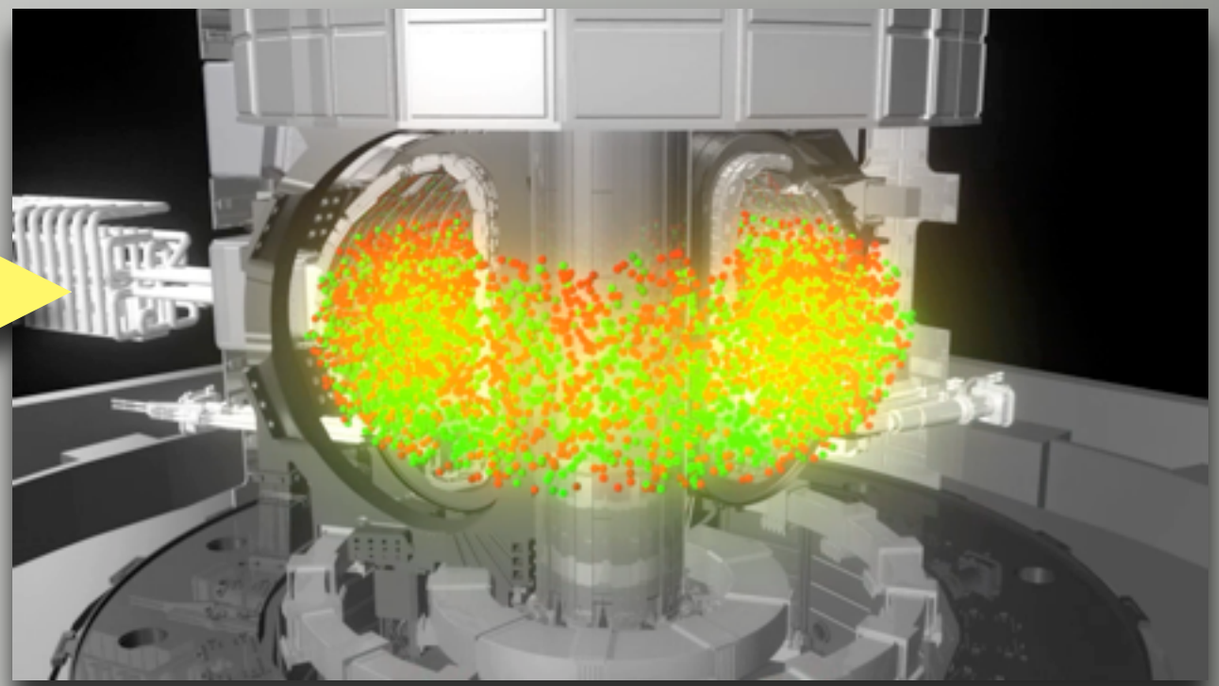
- Data set: Could come from a simulation, a physical experiment, a scientific instrument, anywhere.
- Techniques: art, statistics, image processing, “info” vis, “gaming” techniques

Process of converting information



Process of converting information





Visualization progresses up the scientific data model

Visualization progresses up the scientific data model

Visualization progresses up the scientific data model

bytes

bits

Simple

Complex

Visualization progresses up the scientific data model

bytes

bits floats

Simple Complex

Visualization progresses up the scientific data model

arrays

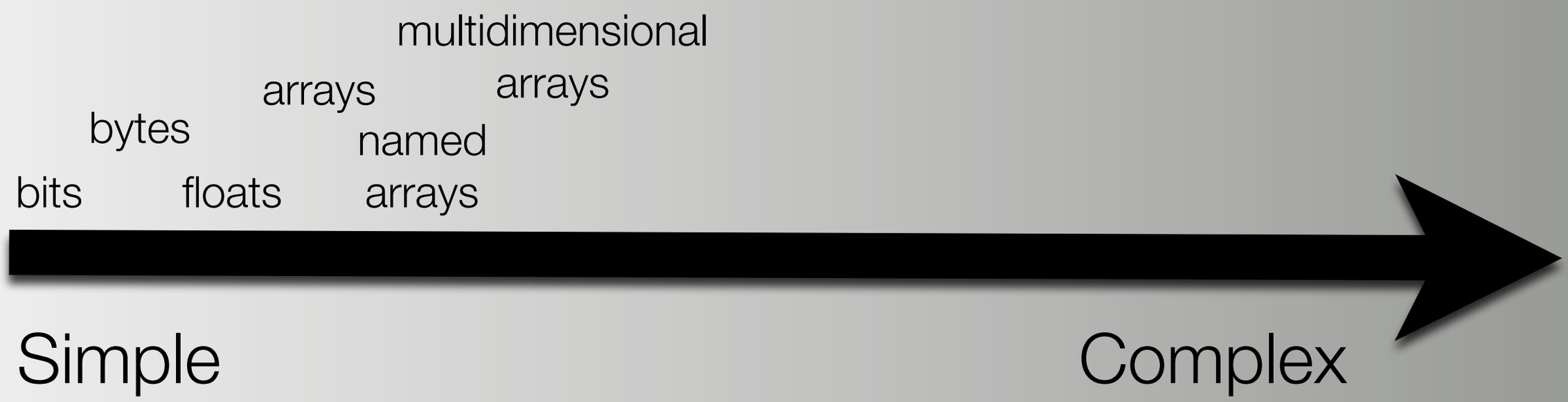
bytes

bits floats

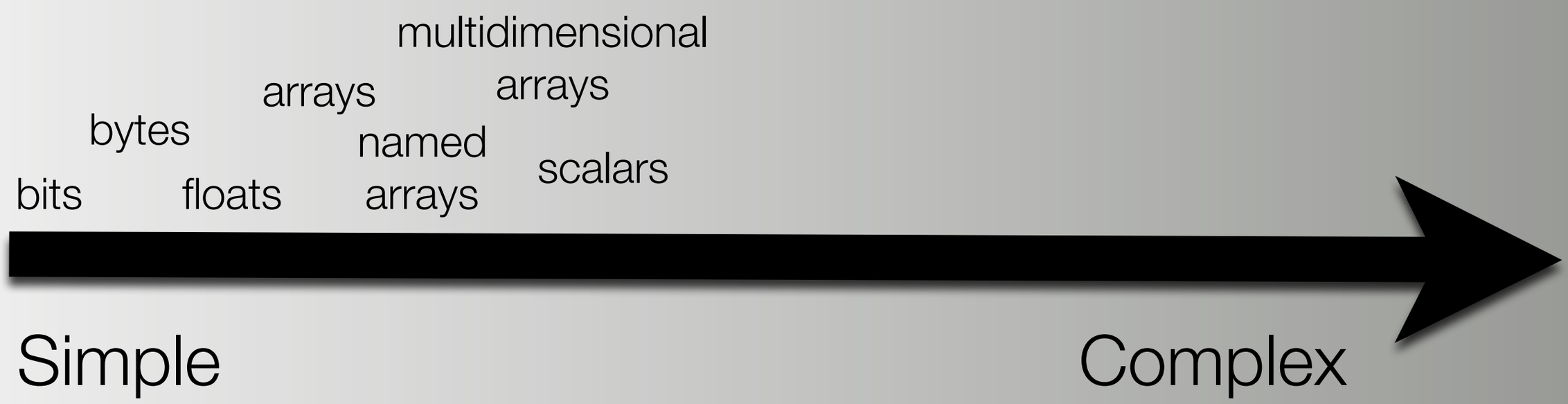
Simple Complex

Visualization progresses up the scientific data model

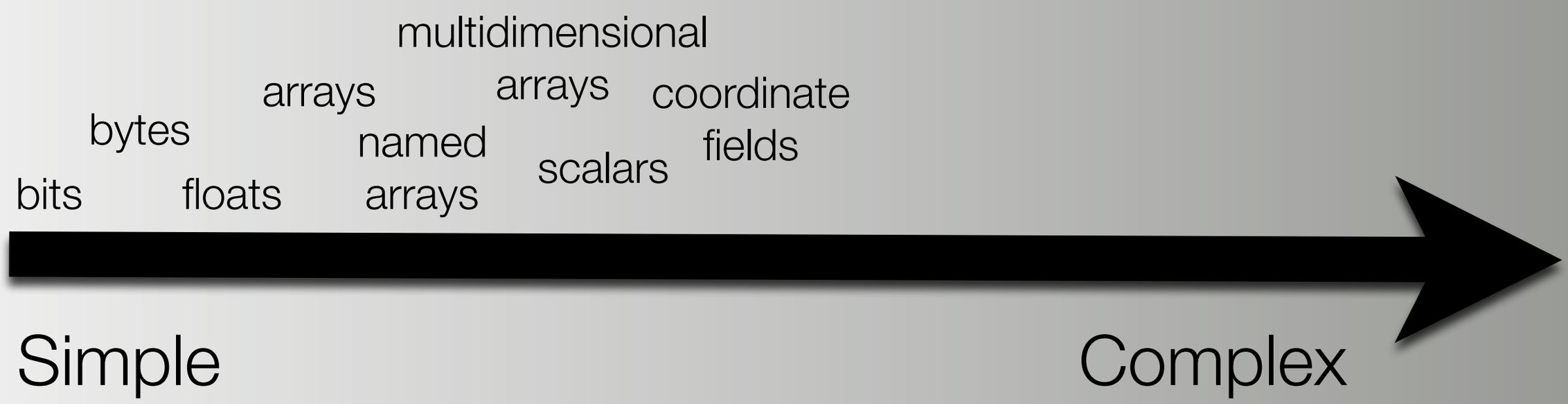
Visualization progresses up the scientific data model



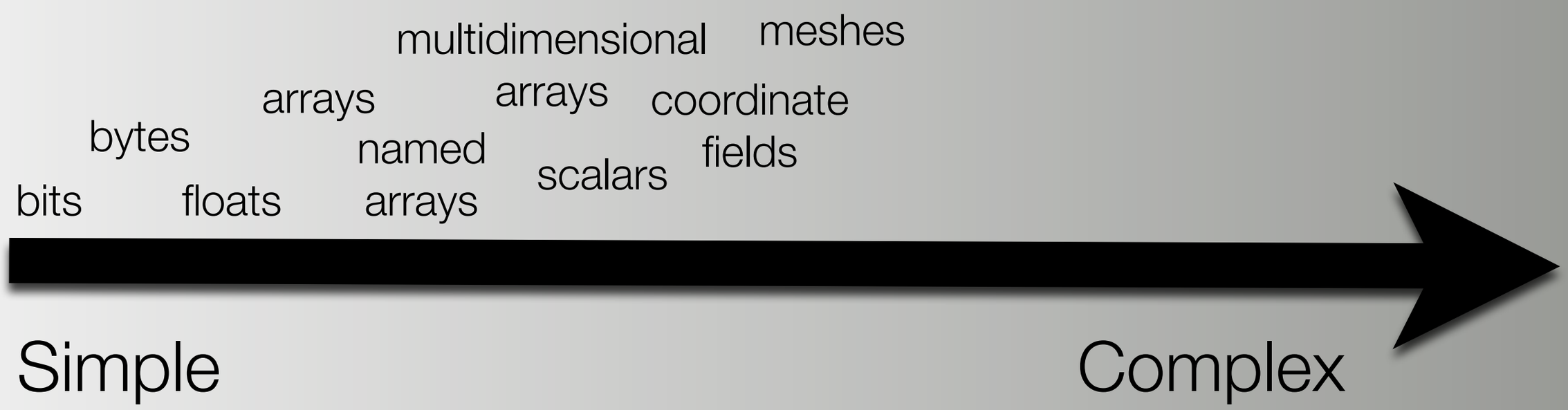
Visualization progresses up the scientific data model



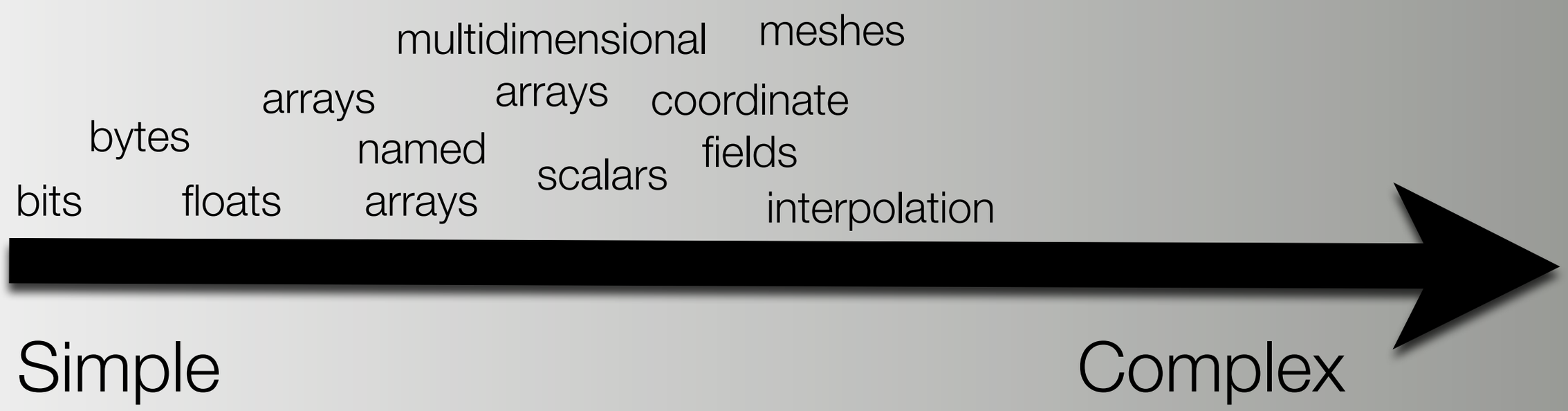
Visualization progresses up the scientific data model



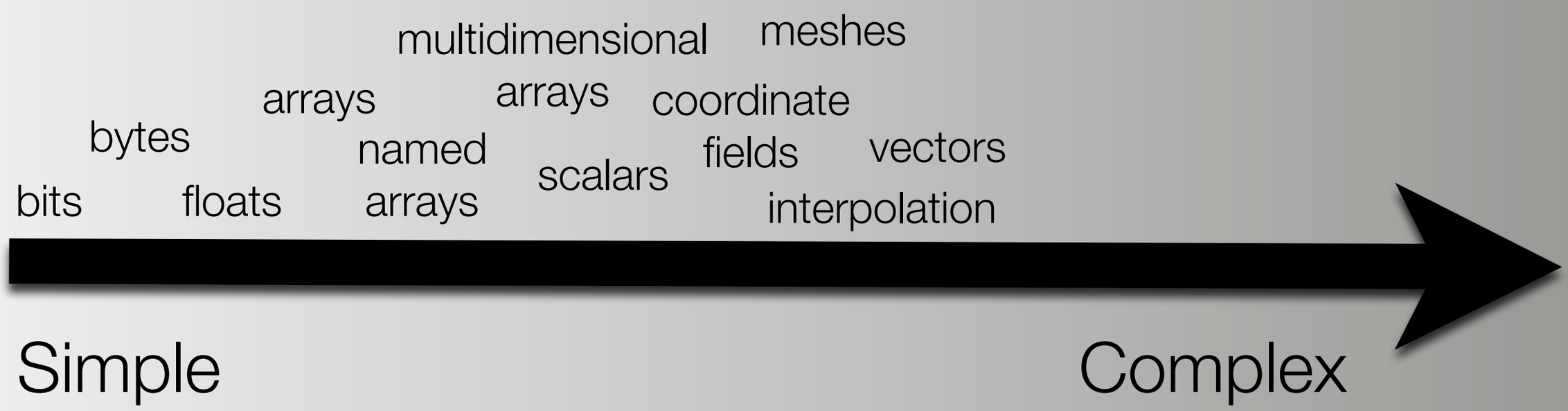
Visualization progresses up the scientific data model



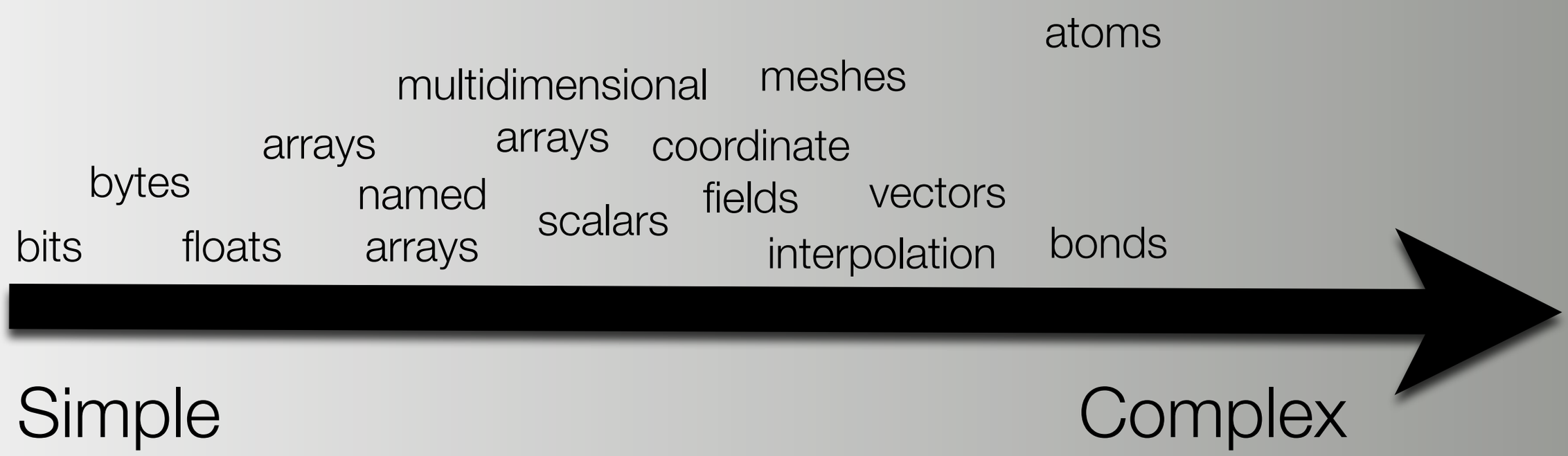
Visualization progresses up the scientific data model



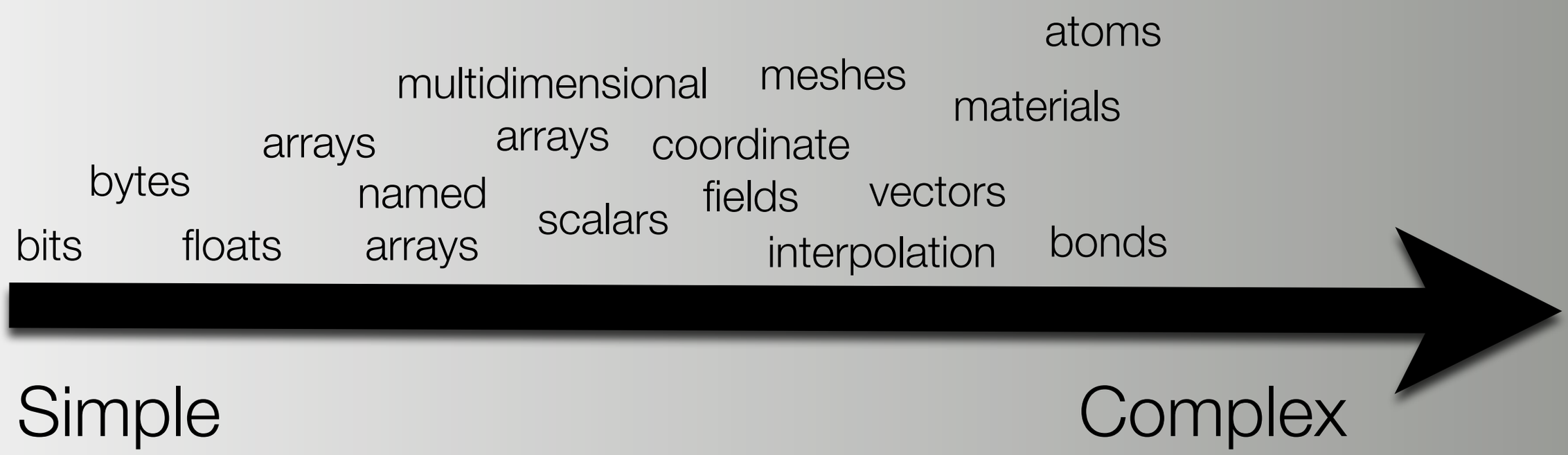
Visualization progresses up the scientific data model



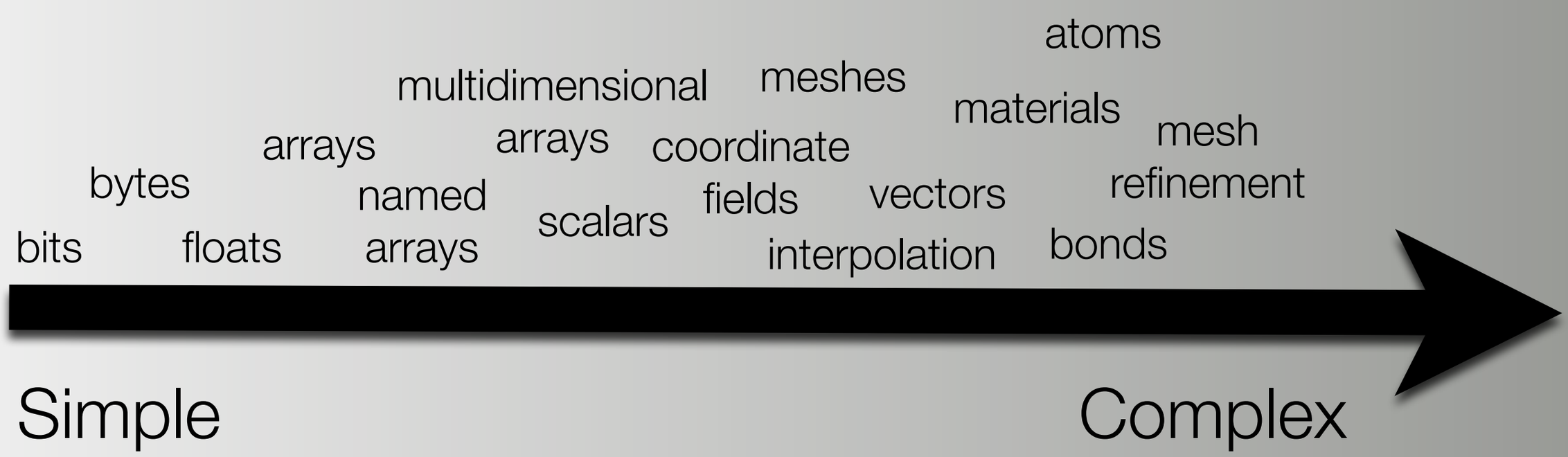
Visualization progresses up the scientific data model



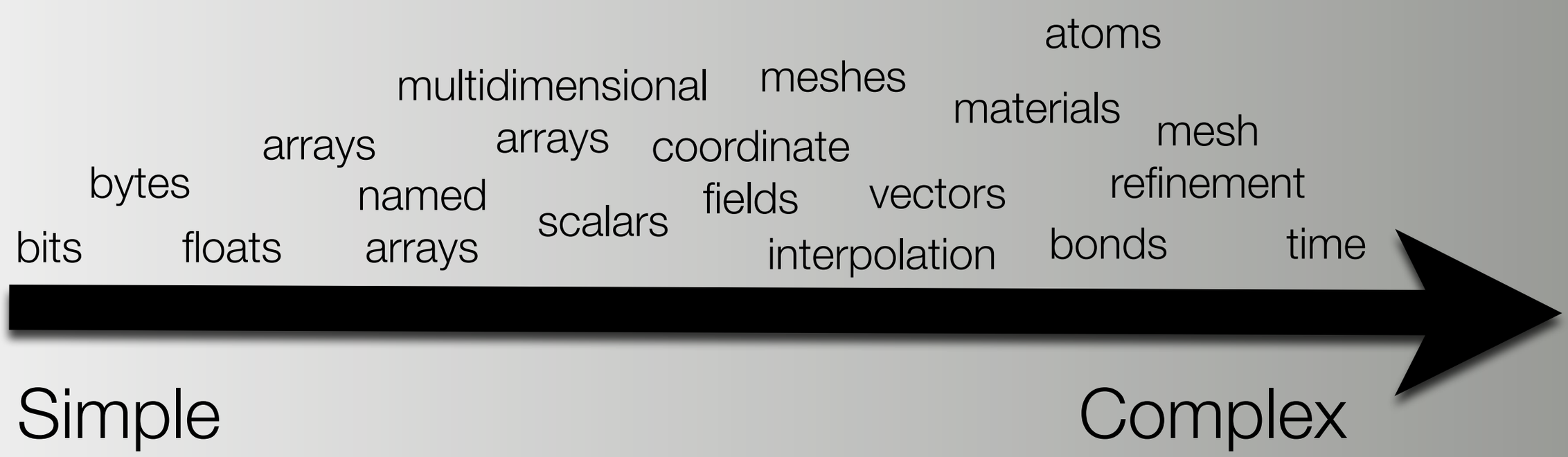
Visualization progresses up the scientific data model



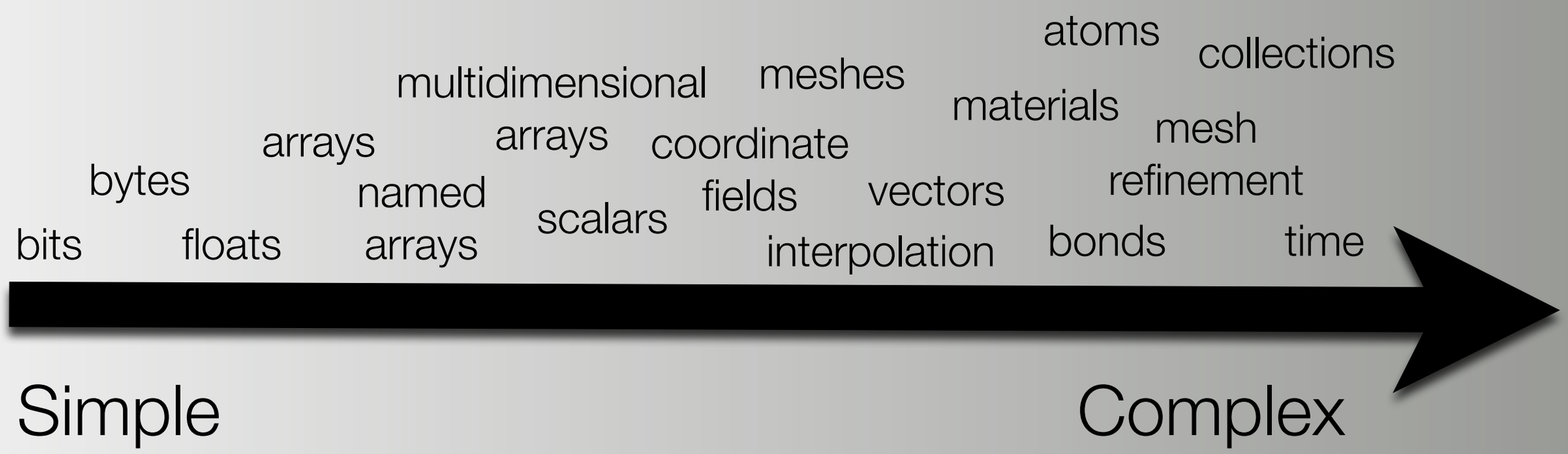
Visualization progresses up the scientific data model



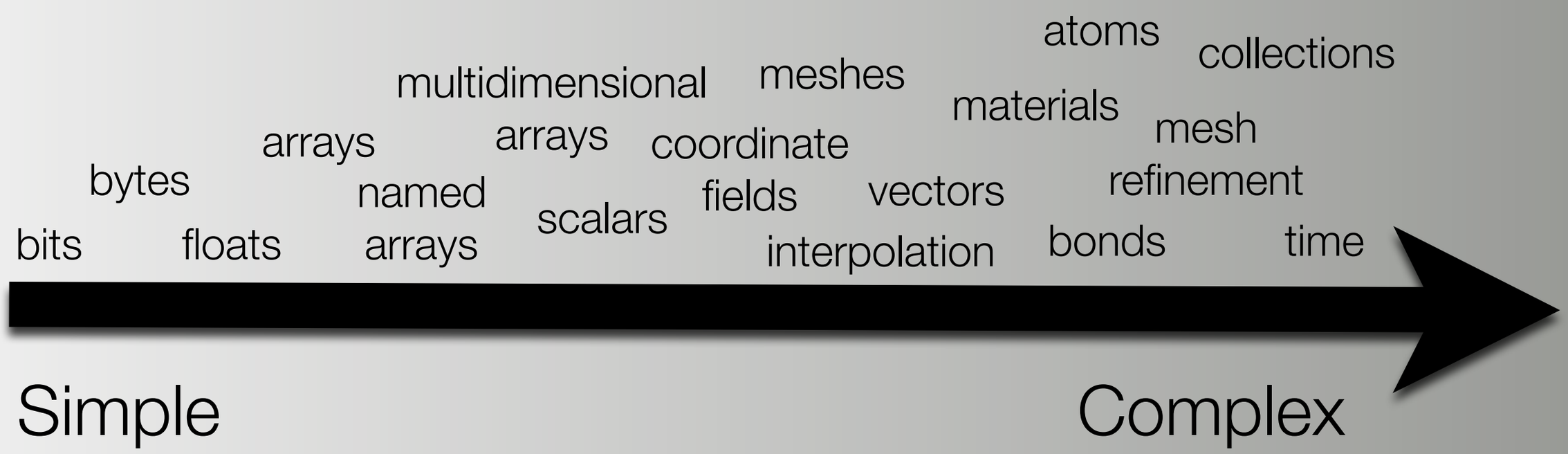
Visualization progresses up the scientific data model



Visualization progresses up the scientific data model

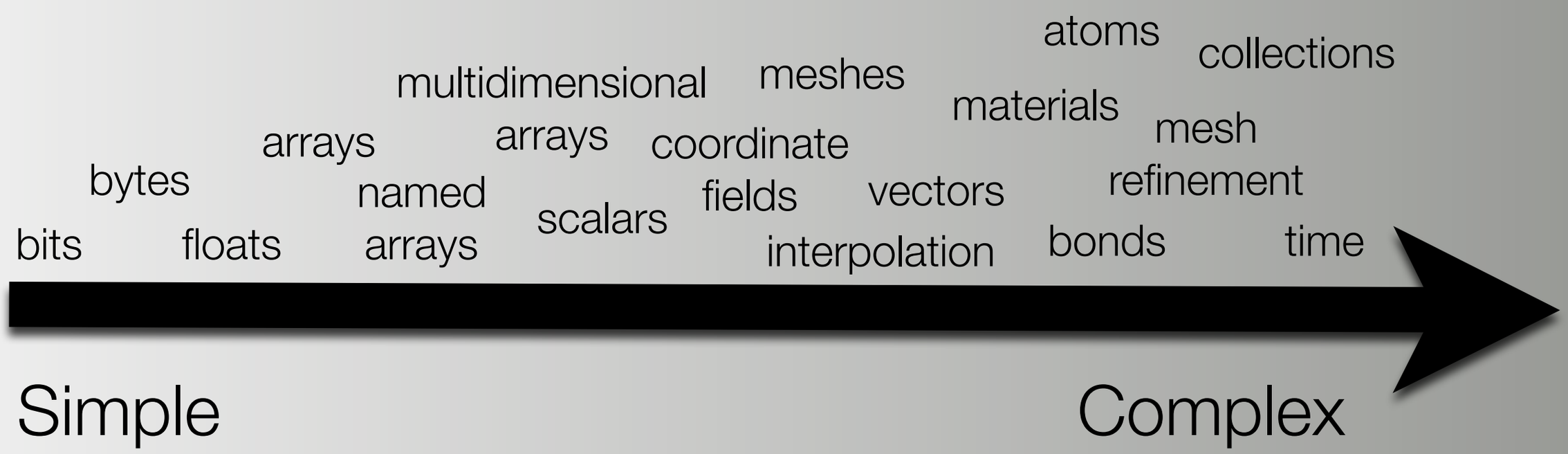


Visualization progresses up the scientific data model



Each step gets us closer to the “ideal”.

Visualization progresses up the scientific data model



Each step gets us closer to the “ideal”.

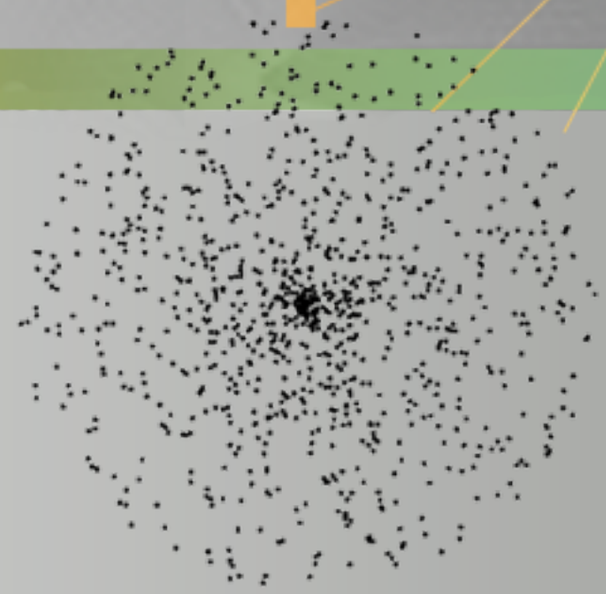
But each step is more domain-specific.

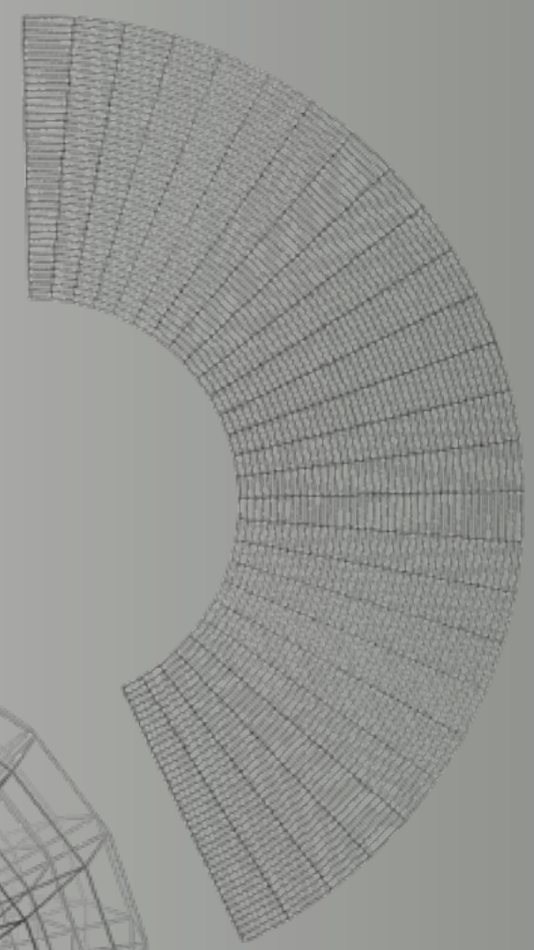
Meshes

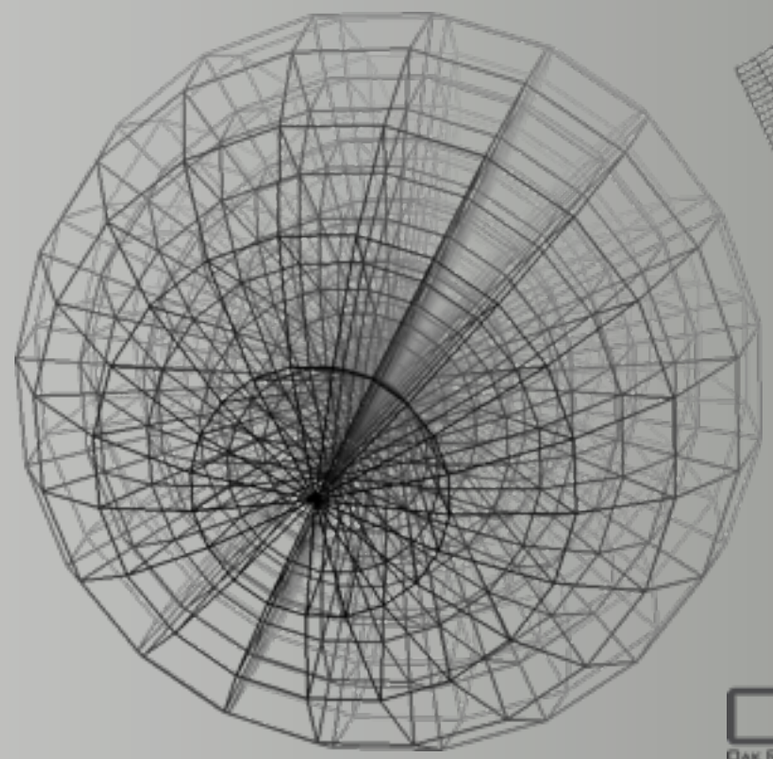
- All data lives on a mesh
- Discretizes space into points and cells
  - 1D, 2D, 3D
  - All of these over time (up to 4D)
  - Can have lower-dimensional meshes in a higher-dimensional space (e.g. 2D surface in 3D space)
- Provides a place for data to be located
- Defines how data is interpolated

Meshes

- All data lives on a mesh
- Discretizes space into points and cells
  - 1D, 2D, 3D
  - All of these over time (up to 4D)
  - Can have lower-dimensional meshes in a higher-dimensional space (e.g. 2D surface in 3D space)
- Provides a place for data to be located
- Defines how data is interpolated

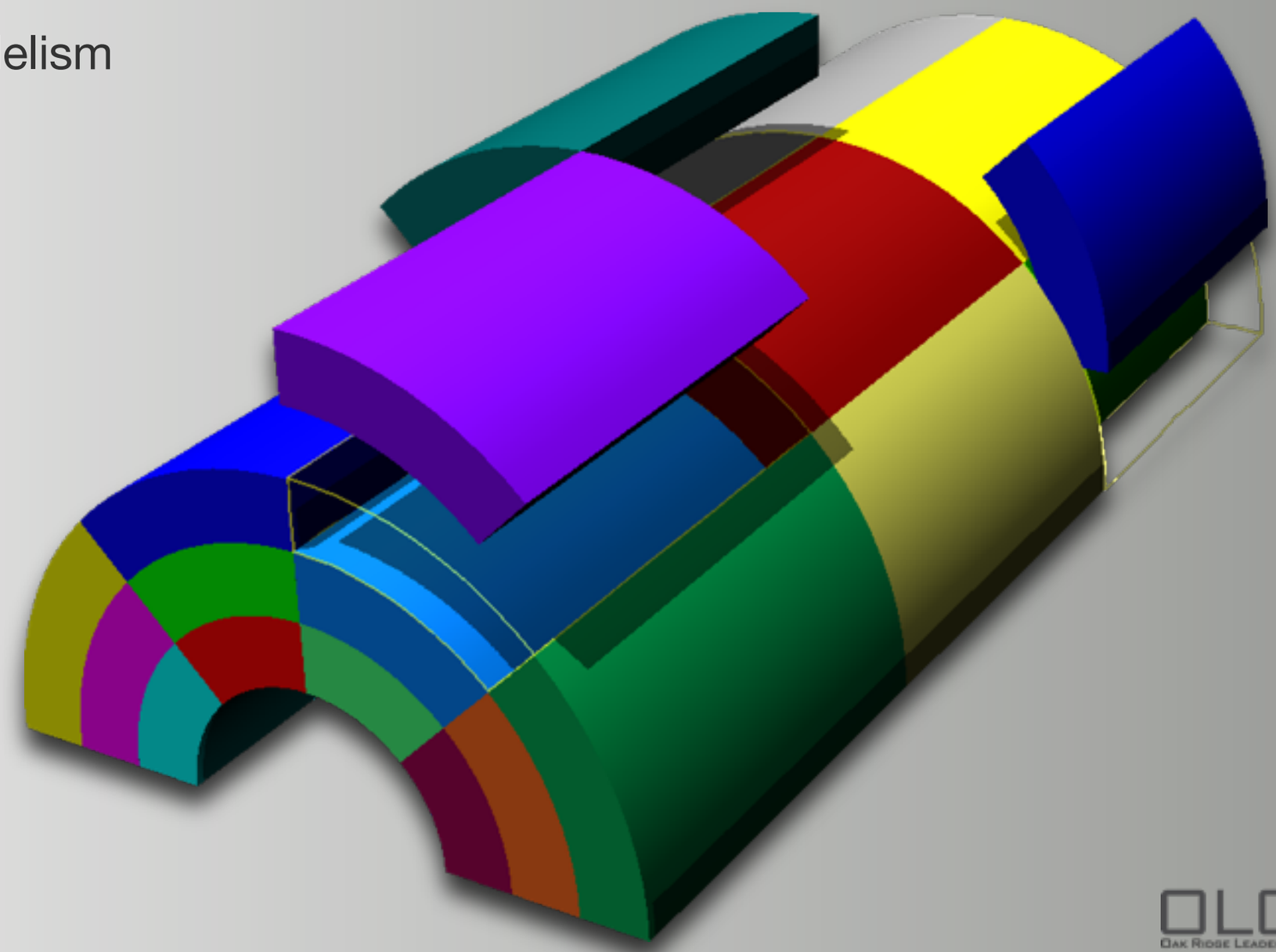






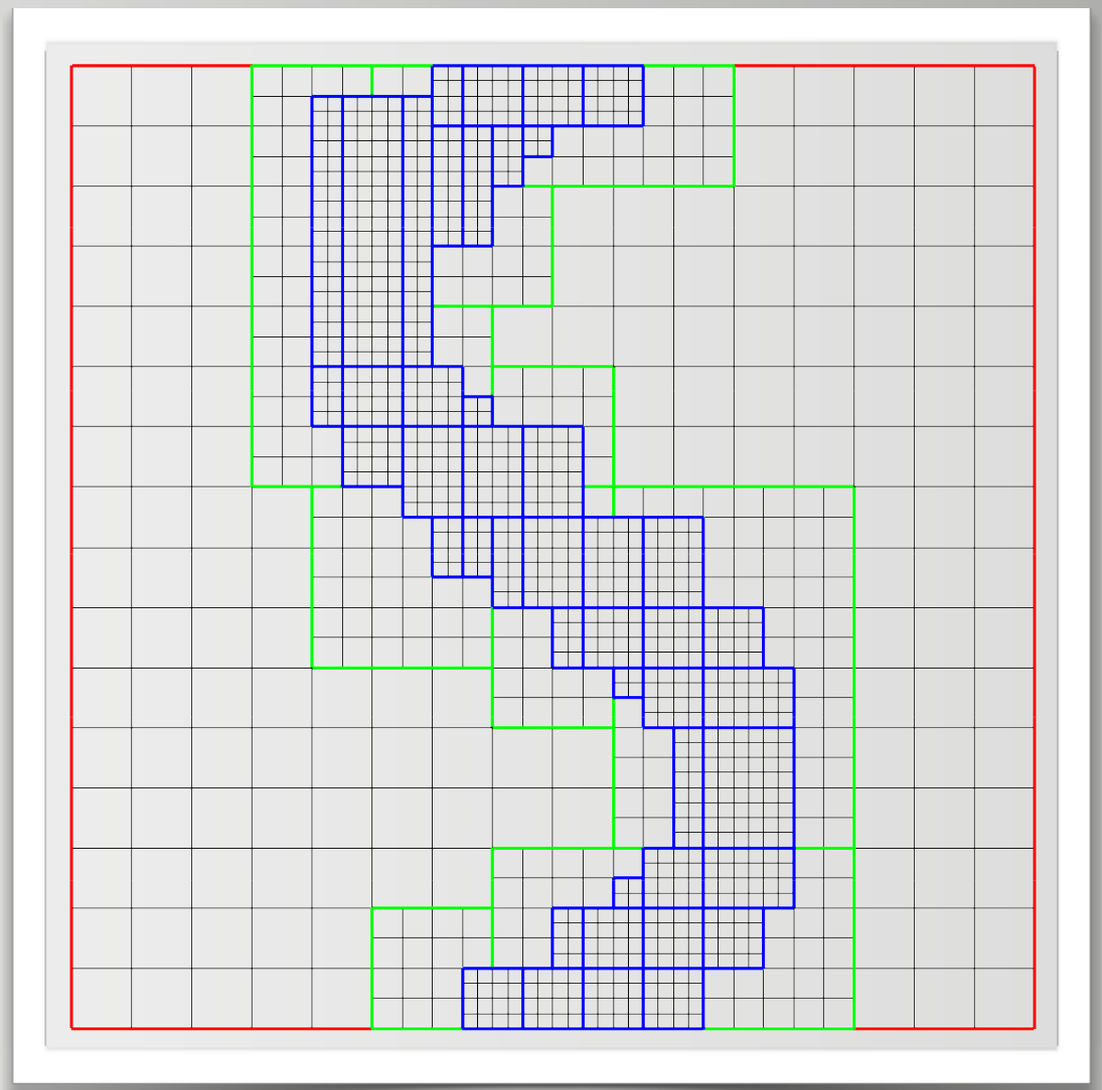
Parallel meshes

- Provides aggregation for meshes
- A mesh may be composed of hundreds of thousands of mesh “blocks”.
- Allows data parallelism



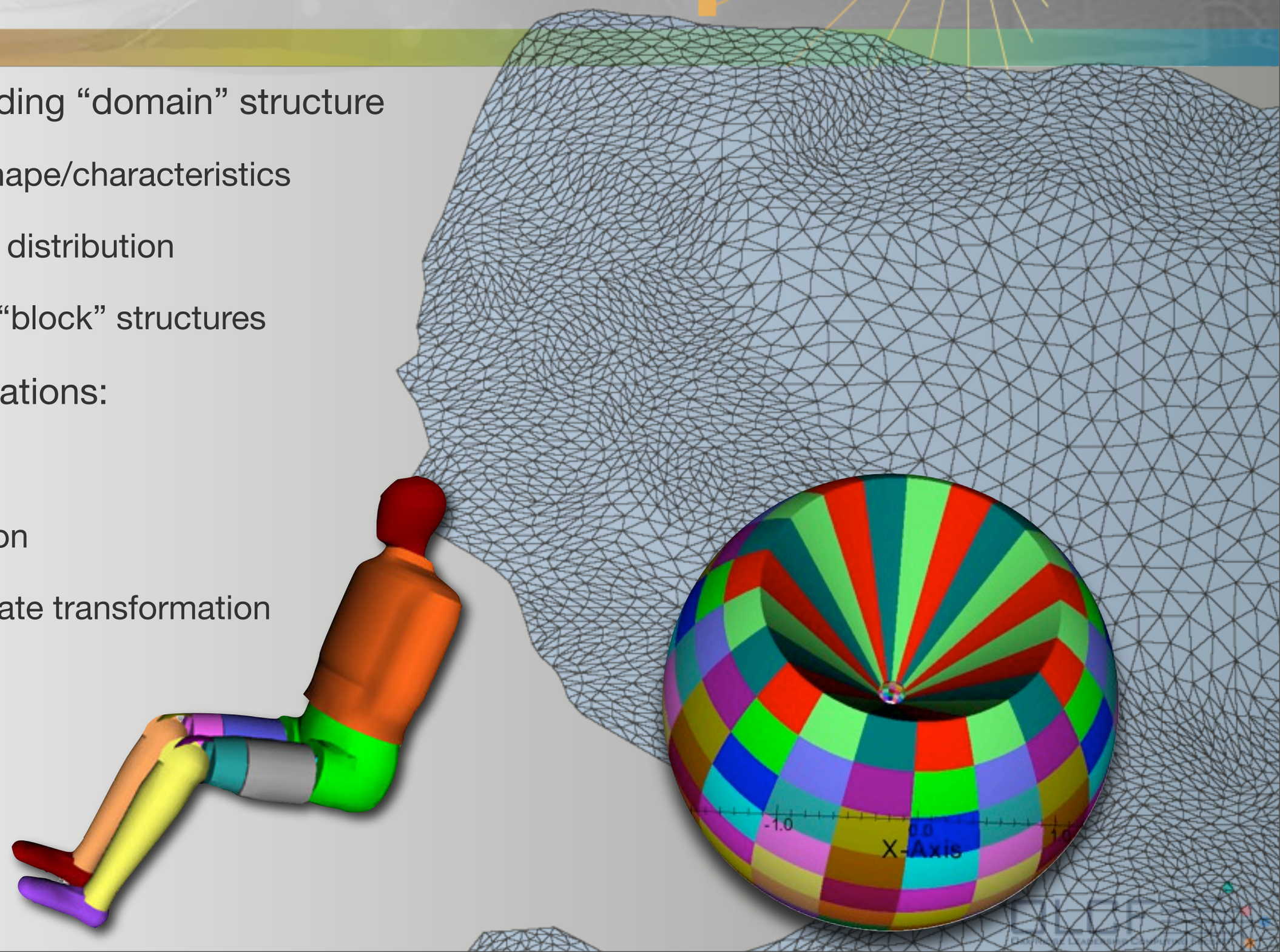
AMR meshes

- Mesh blocks can be associated with patches and levels.
- Allows for aggregation of meshes into AMR hierarchy levels.



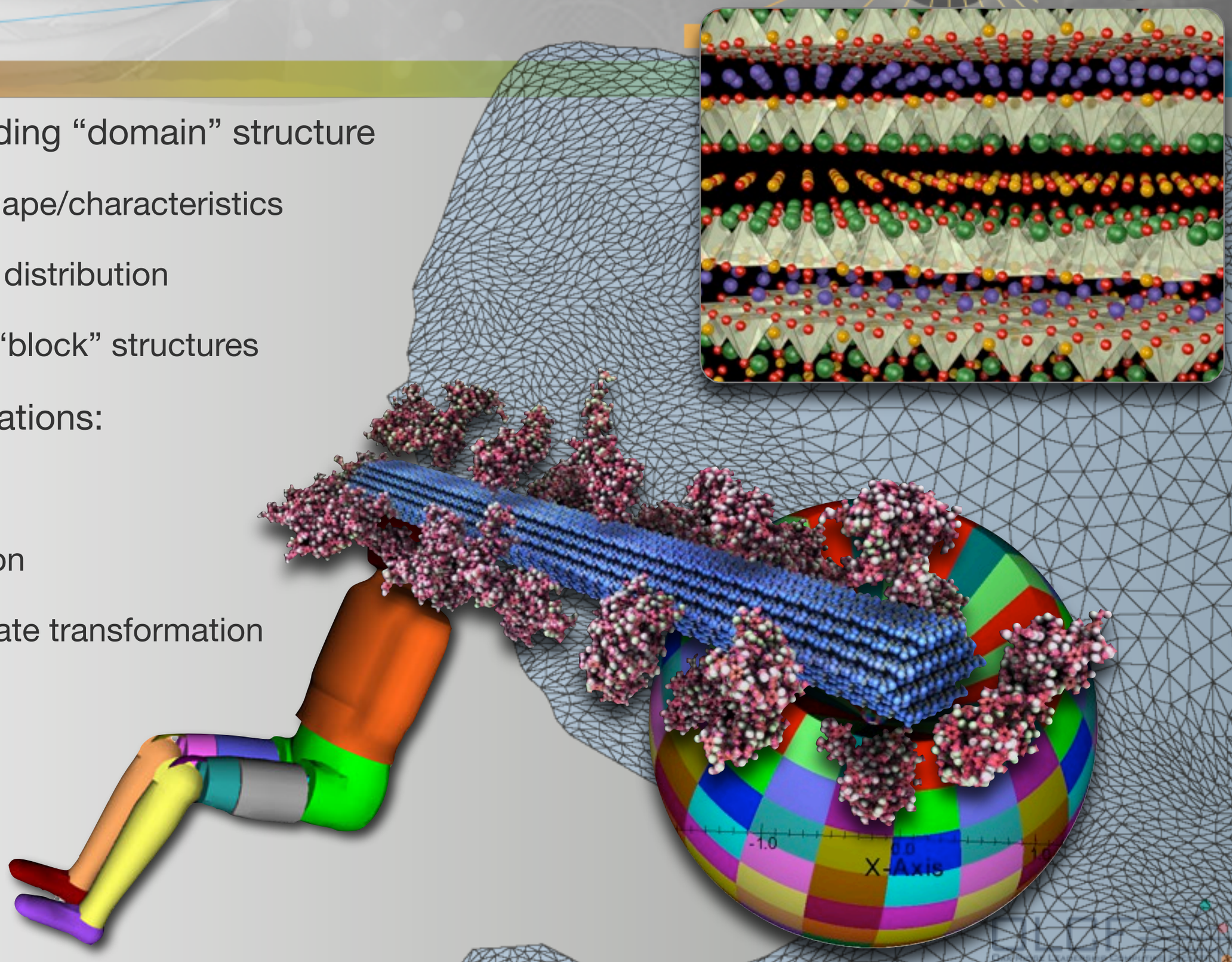
“Mesh” visualization

- Understanding “domain” structure
  - Mesh shape/characteristics
  - Material distribution
  - Parallel “block” structures
- Mesh operations:
  - Slicing
  - Projection
  - Coordinate transformation



“Mesh” visualization

- Understanding “domain” structure
  - Mesh shape/characteristics
  - Material distribution
  - Parallel “block” structures
- Mesh operations:
  - Slicing
  - Projection
  - Coordinate transformation

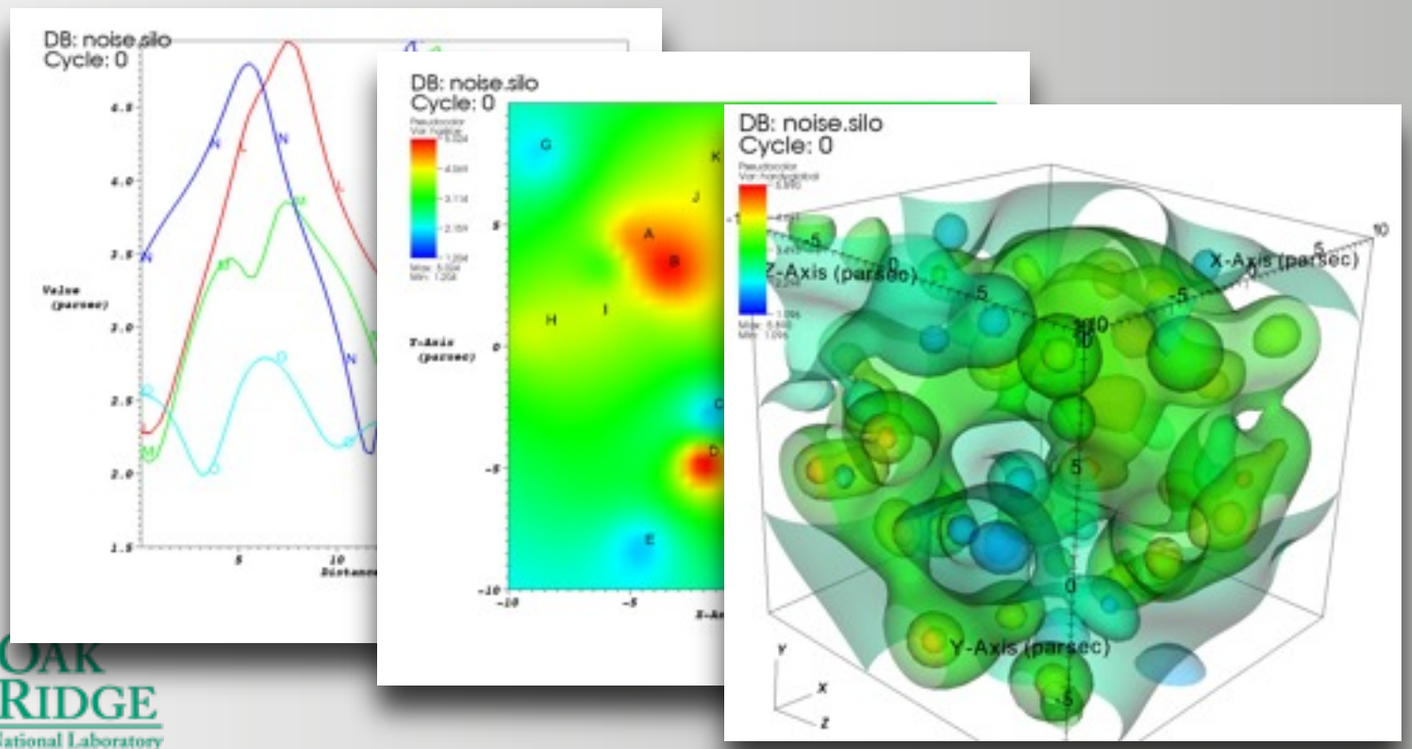


Variables

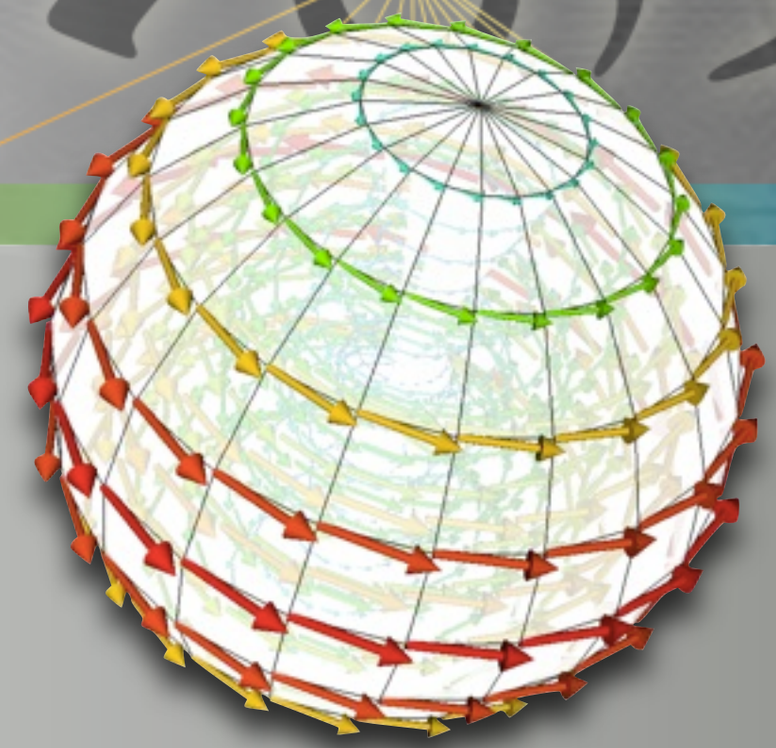
- Scalars, Vectors, Tensors
- Sits on points or cells of a mesh
  - Points: linear interpolation
  - Cells: piecewise constant
- Could have different dimensionality than the mesh (e.g. 3D vector data on a 2D mesh)

Variables

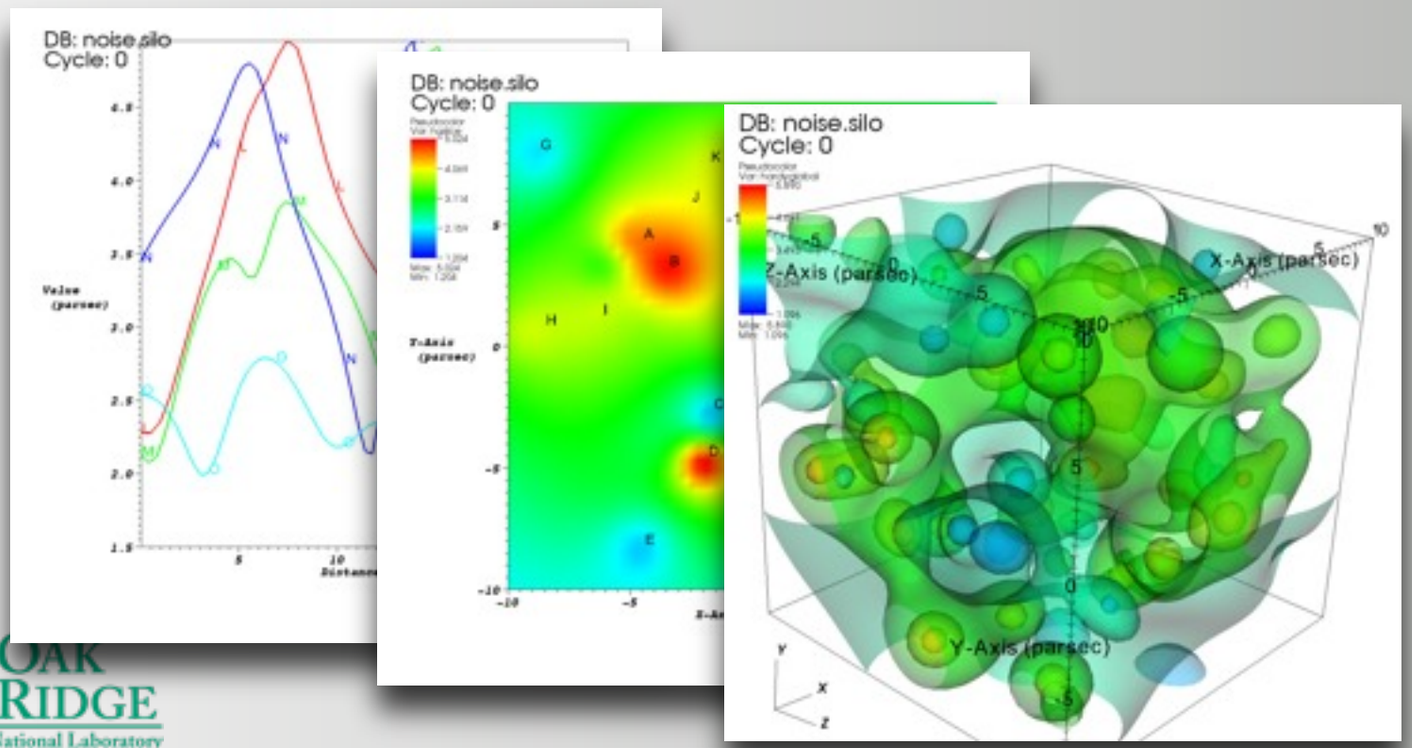
- Scalars, Vectors, Tensors
- Sits on points or cells of a mesh
  - Points: linear interpolation
  - Cells: piecewise constant
- Could have different dimensionality than the mesh (e.g. 3D vector data on a 2D mesh)



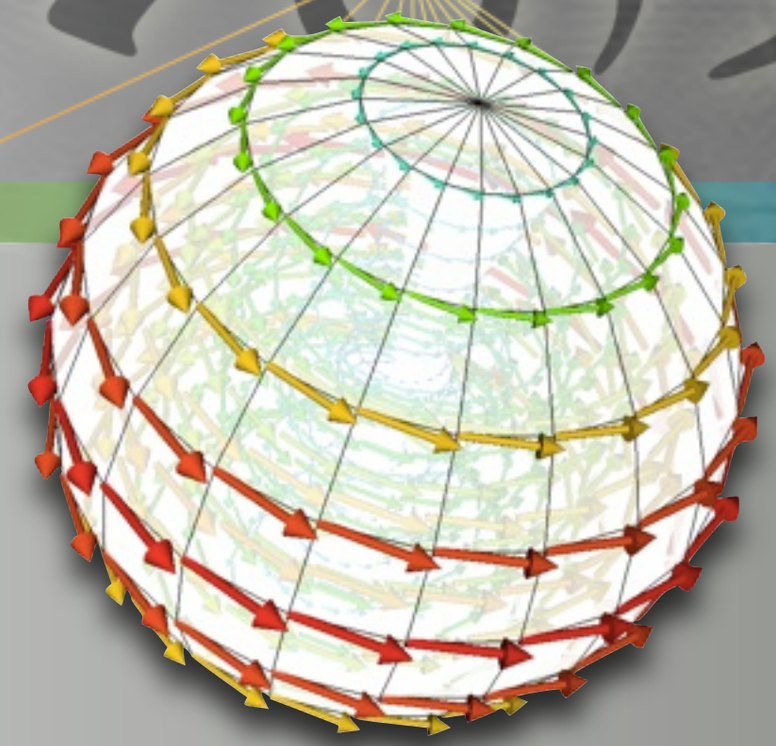
Variables



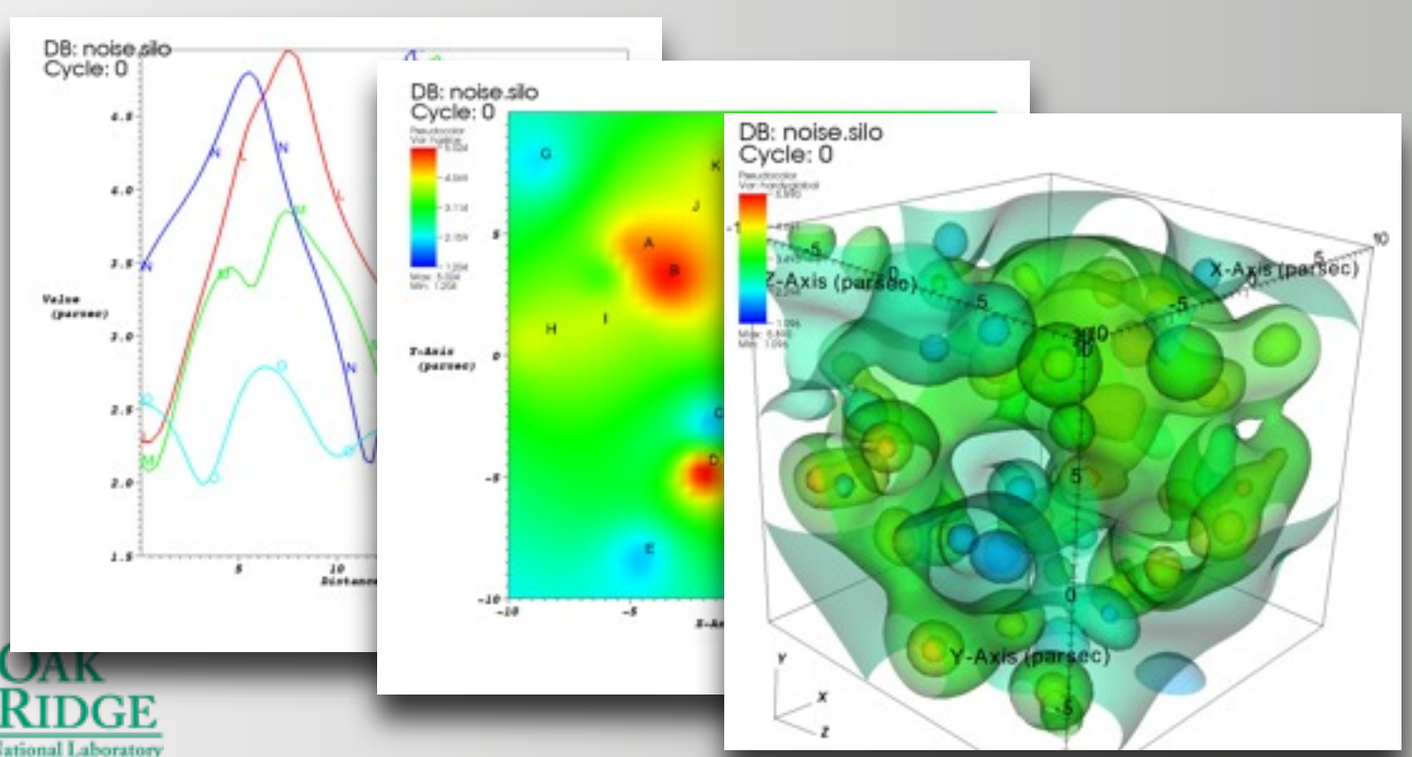
- Scalars, Vectors, Tensors
- Sits on points or cells of a mesh
  - Points: linear interpolation
  - Cells: piecewise constant
- Could have different dimensionality than the mesh (e.g. 3D vector data on a 2D mesh)

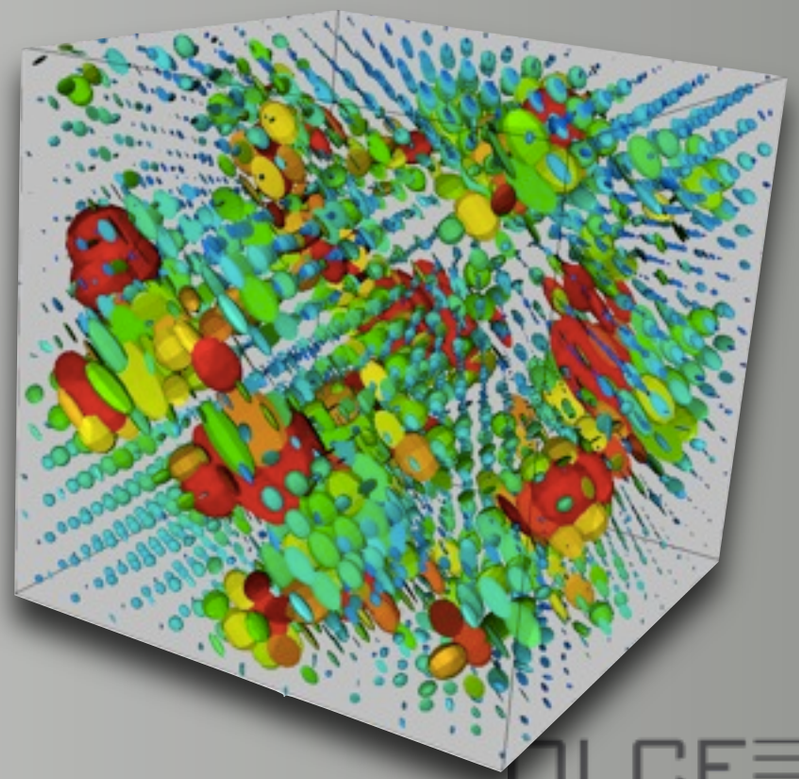


Variables



- Scalars, Vectors, Tensors
- Sits on points or cells of a mesh
  - Points: linear interpolation
  - Cells: piecewise constant
- Could have different dimensionality than the mesh (e.g. 3D vector data on a 2D mesh)



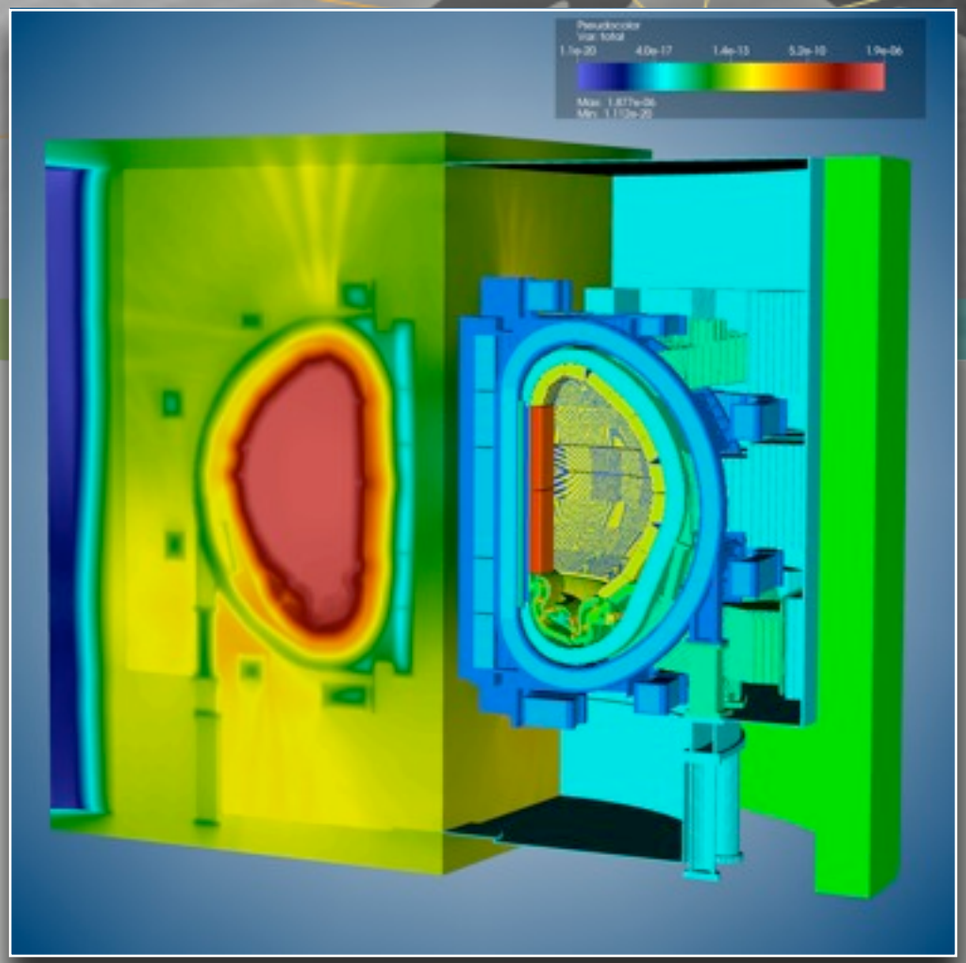


Scalar Field Visualization Techniques

- “Coloring”, pseudocolor plots
- Contours/Isosurfaces
- Direct Volume rendering
- Advanced: topology extraction

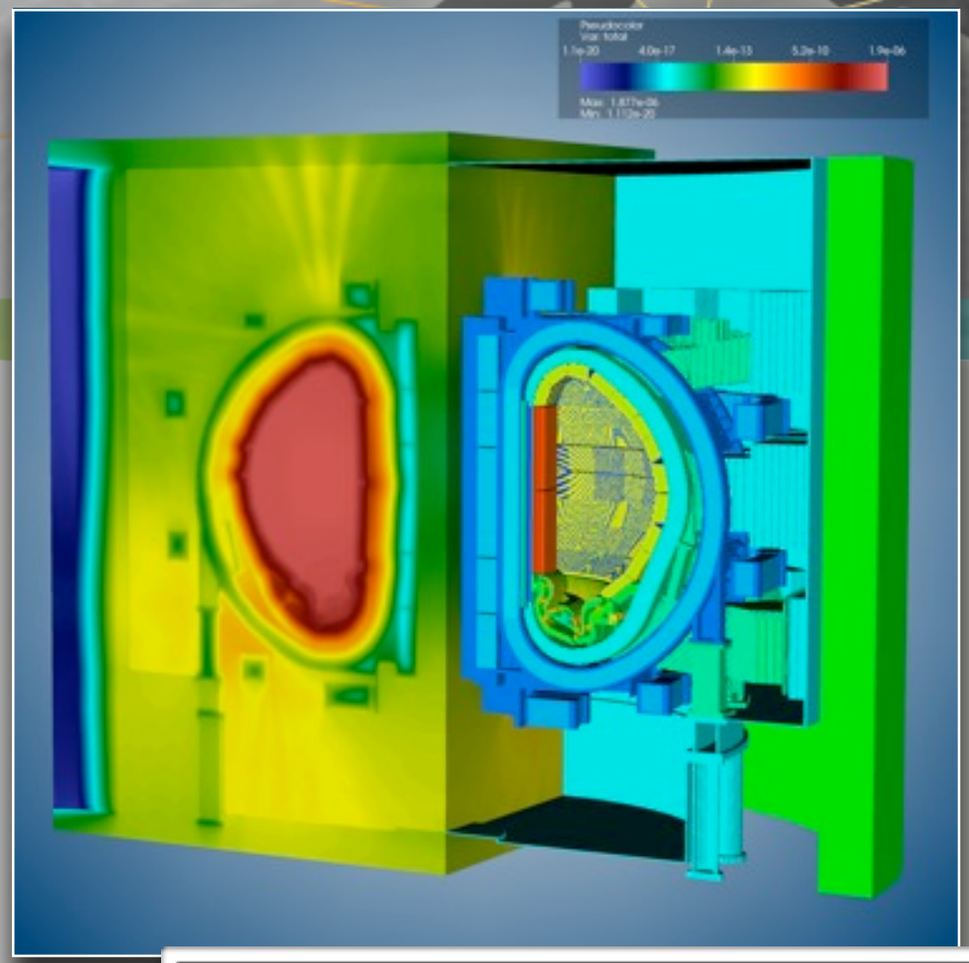
Scalar Field Visualization Techniques

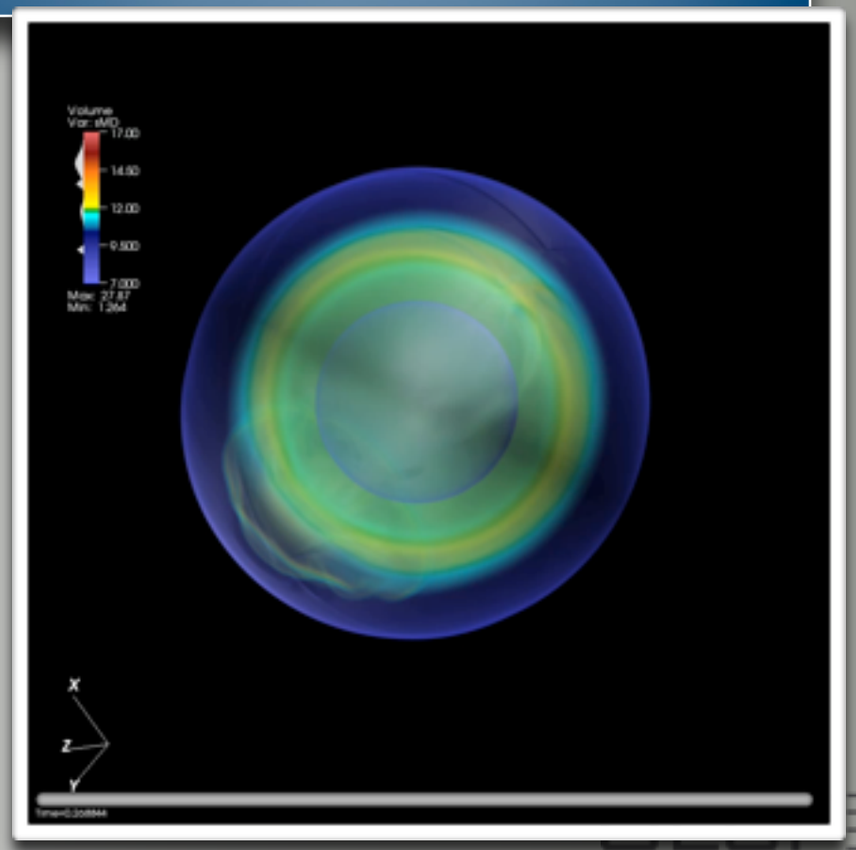
- “Coloring”, pseudocolor plots
- Contours/Isosurfaces
- Direct Volume rendering
- Advanced: topology extraction



Scalar Field Visualization Techniques

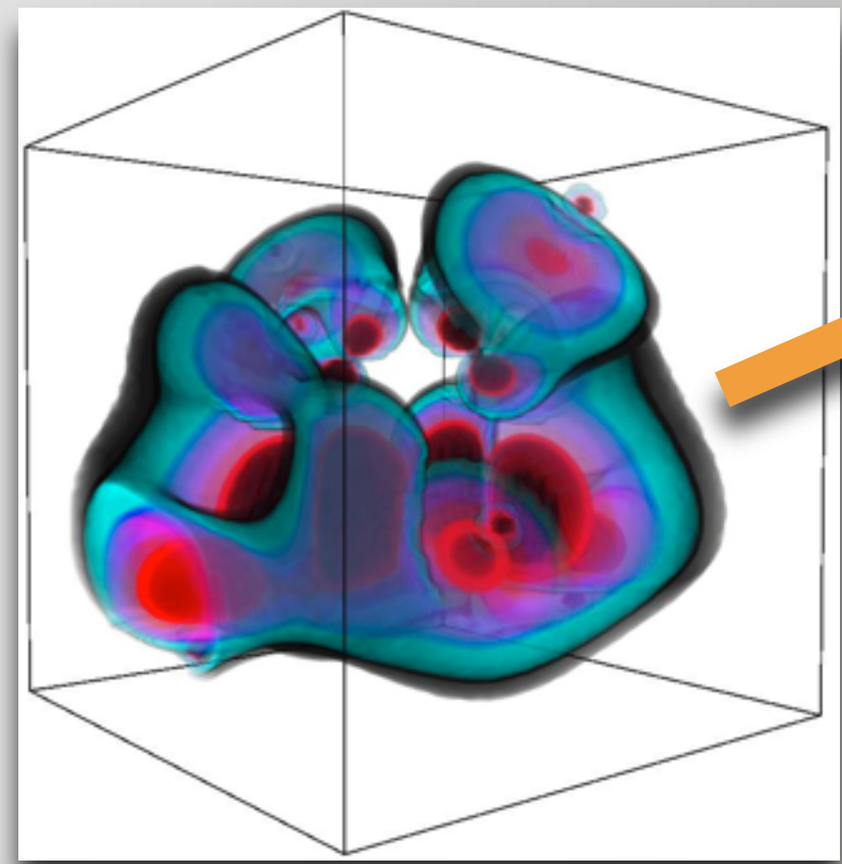
- “Coloring”, pseudocolor plots
- Contours/Isosurfaces
- Direct Volume rendering

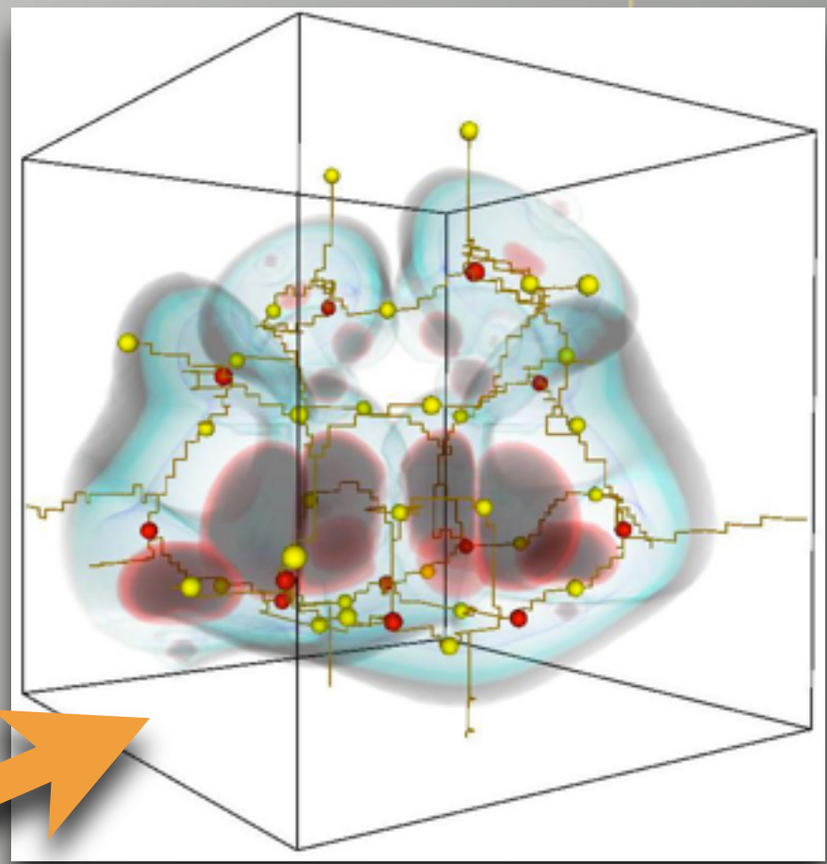




Scalar Field Visualization Techniques

- “Coloring”, pseudocolor plots
- Contours/Isosurfaces
- Direct Volume rendering
- Advanced: topology extraction



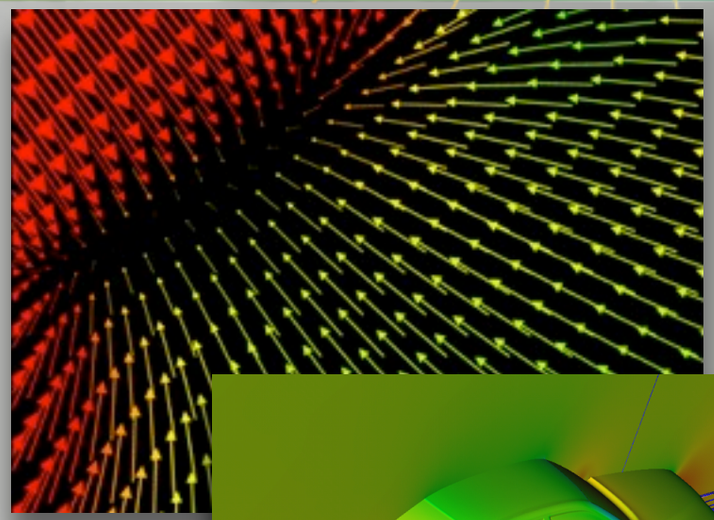


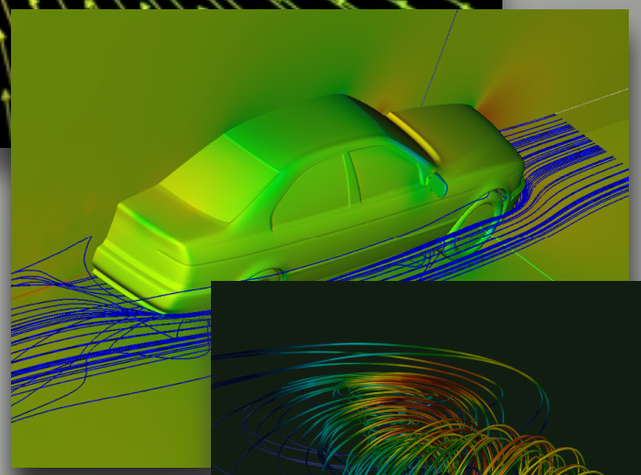
Vector Techniques (Flow)

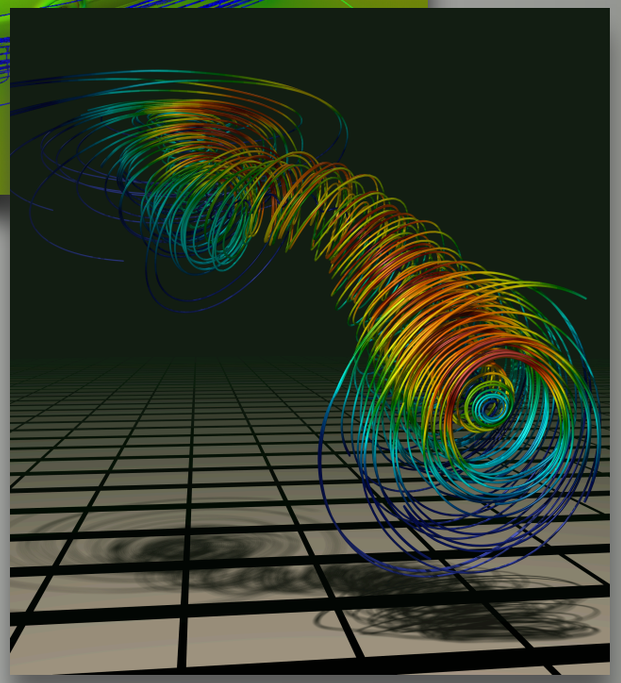
- Vector arrows
  (hedgehog plots)
- Streamlines, stream ribbons, stream tubes,
  stream surfaces
- Noise texture methods

Vector Techniques (Flow)

- Vector arrows
  (hedgehog plots)
- Streamlines, stream ribbons, stream tubes, stream surfaces
- Noise texture methods

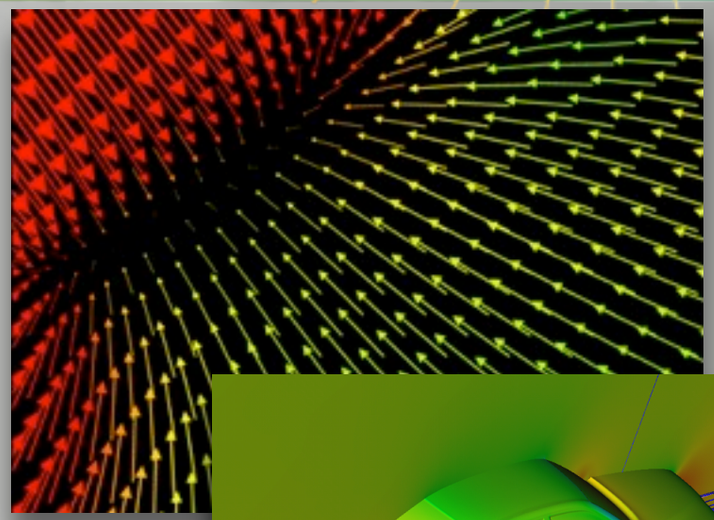


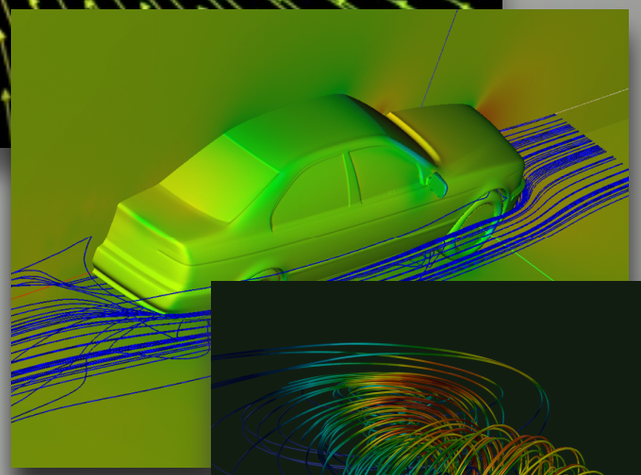


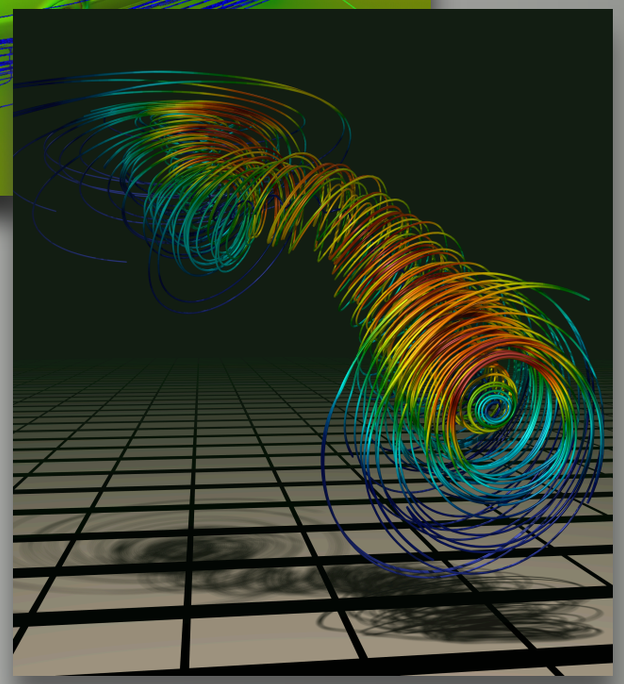


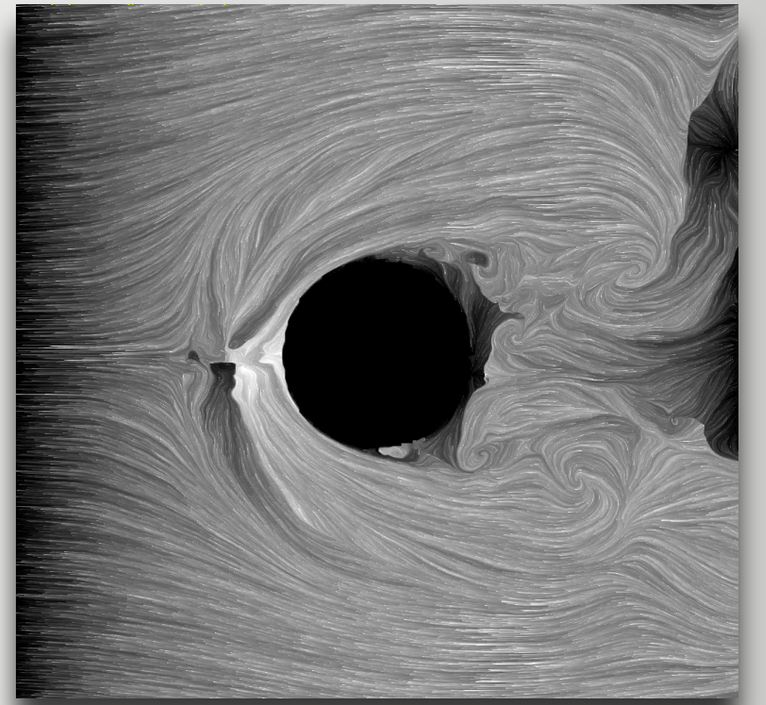
Vector Techniques (Flow)

- Vector arrows
  (hedgehog plots)
- Streamlines, stream ribbons, stream tubes, stream surfaces
- Noise texture methods



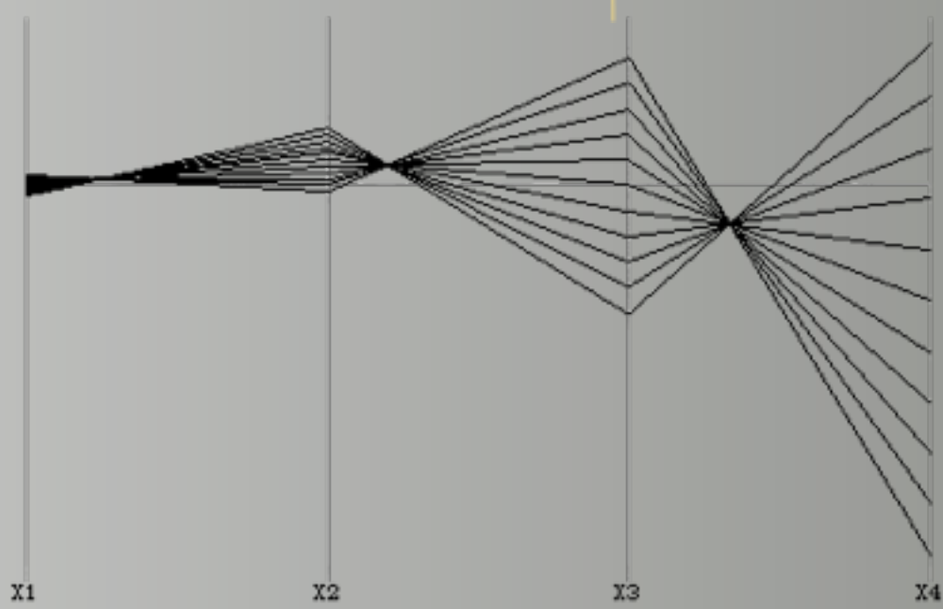


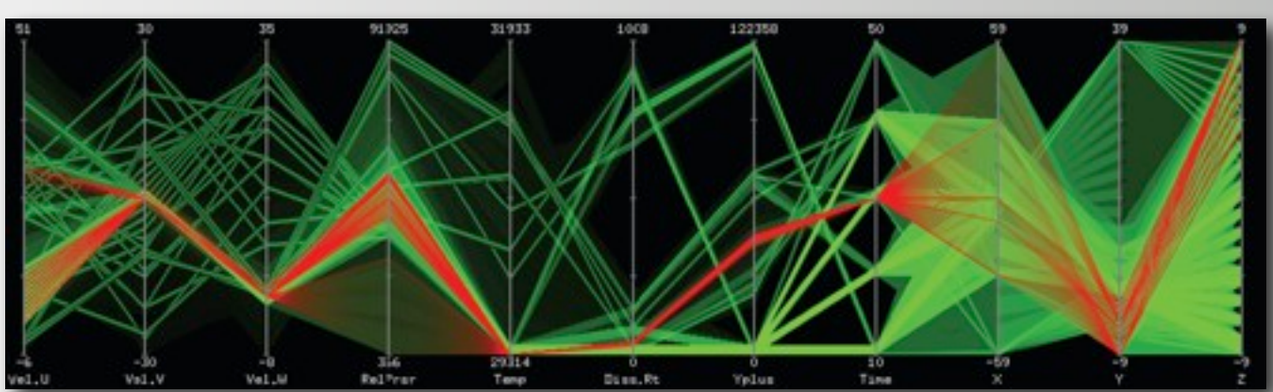


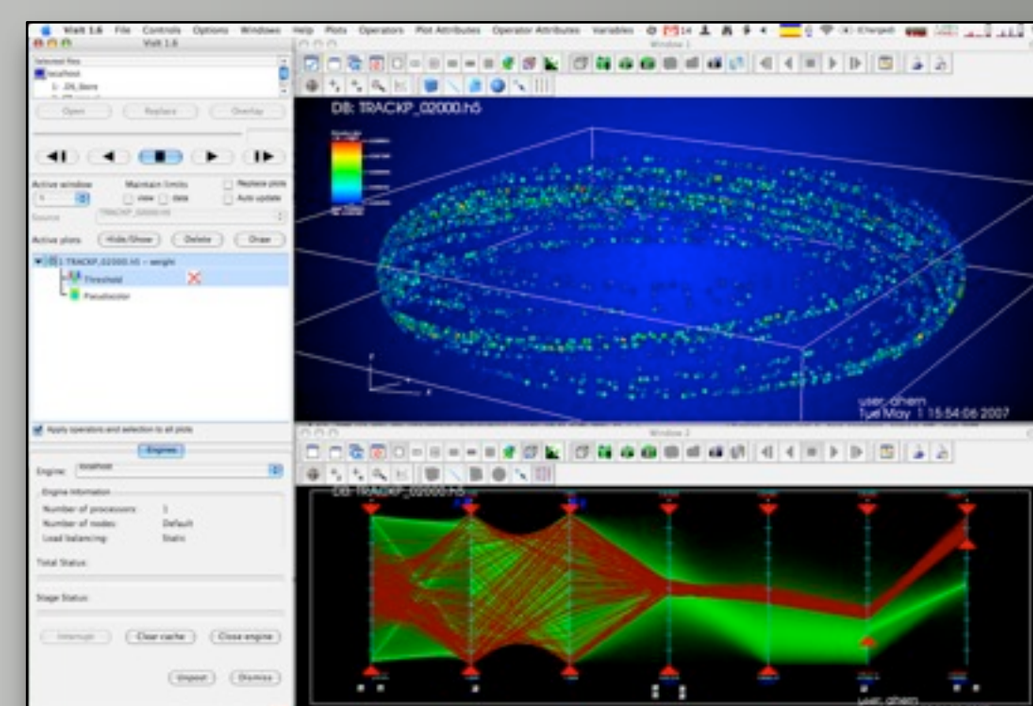


High-dimensional Visualization

- Use info-vis technique of *parallel coordinates* to summarize high-dimensional information







End-to-end schematic

HPC System

End-to-end schematic

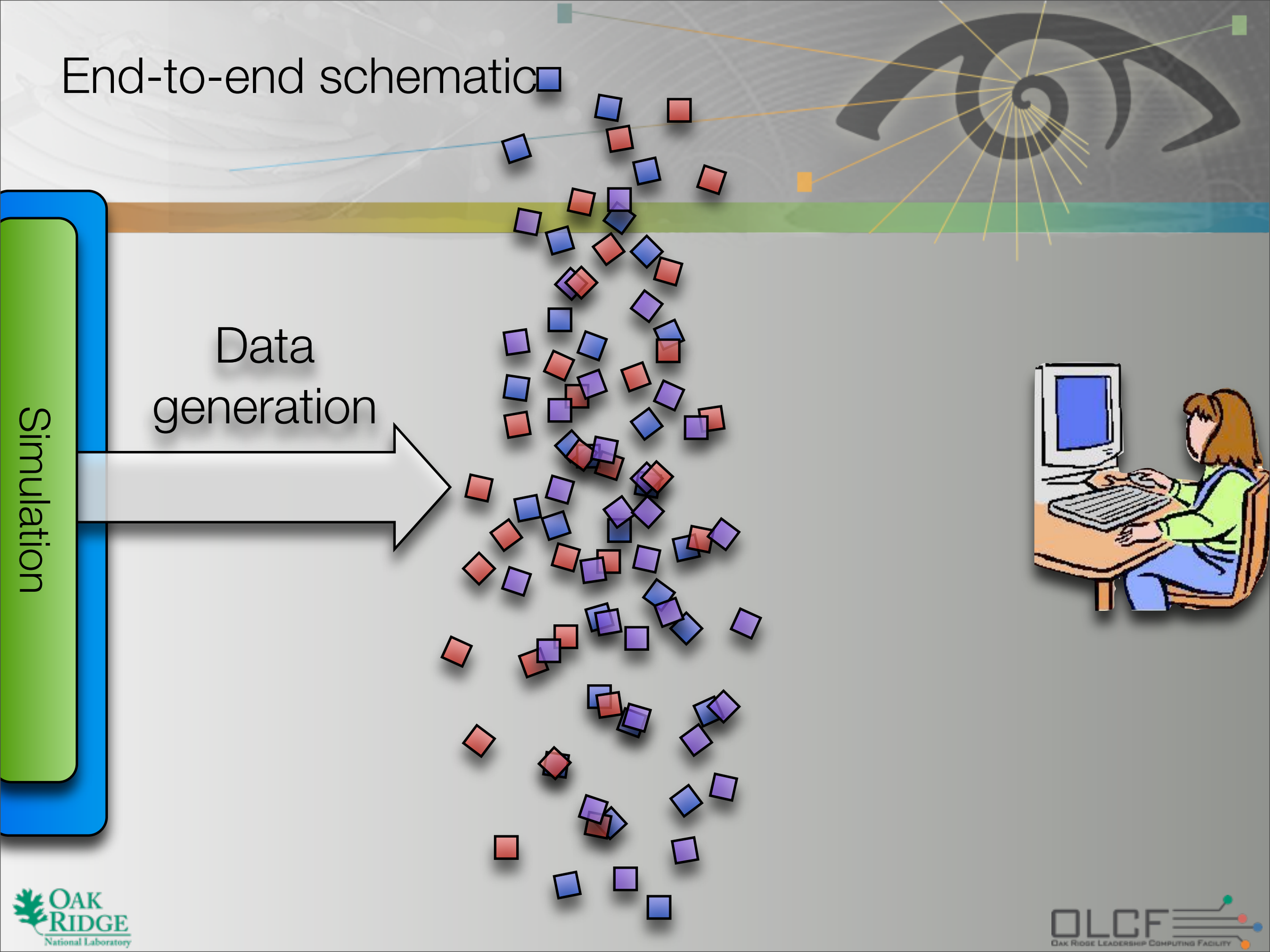
HPC System

Input

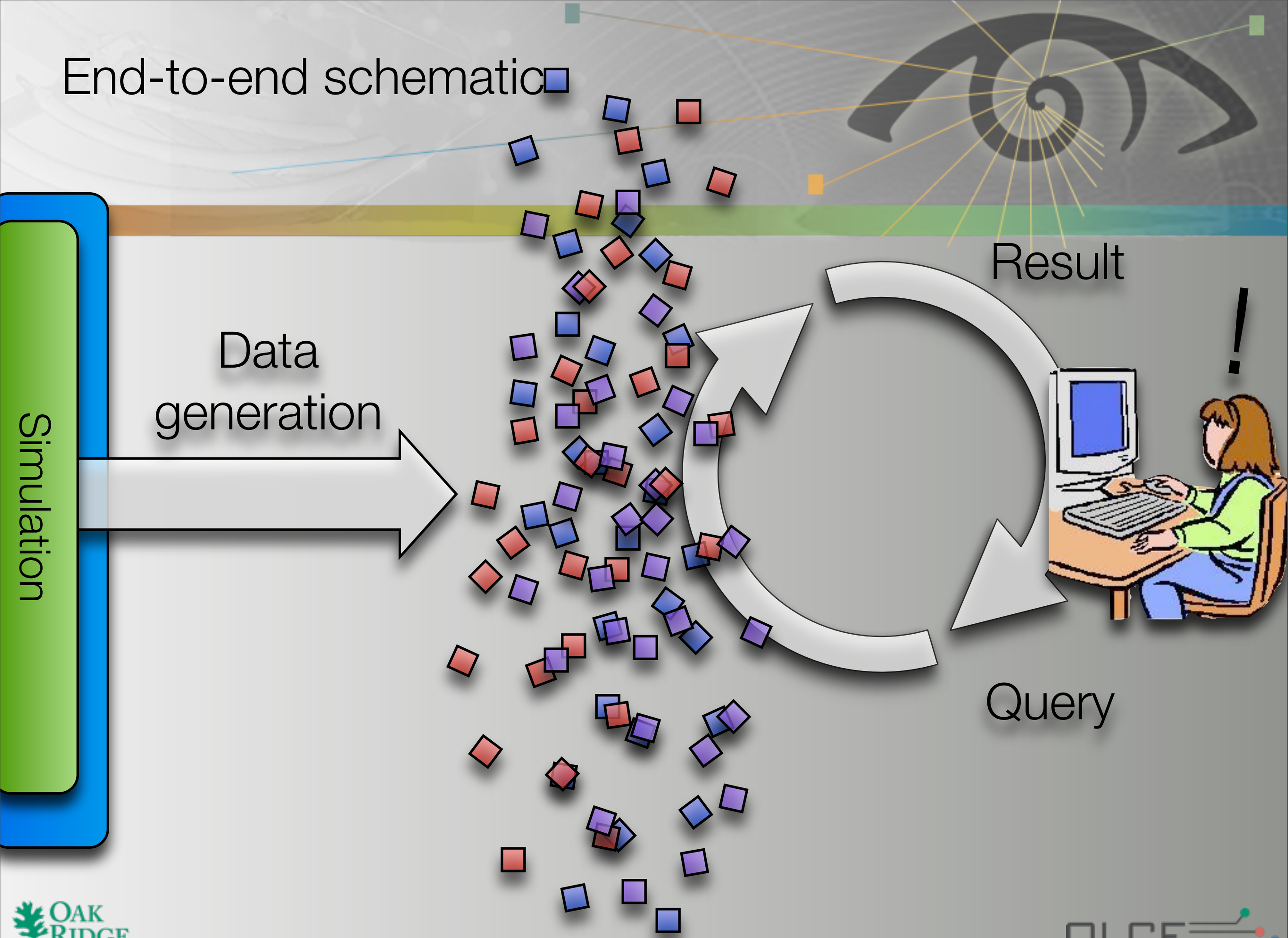
End-to-end schematic

Simulation

End-to-end schematic



End-to-end schematic



Applications are pushing data set sizes toward Exascale

Applications are pushing data set sizes toward Exascale

Spatial resolution

Astrophysics, Radiation transport, Combustion, Chemistry, Molecular dynamics, Fusion, …

Applications are pushing data set sizes toward Exascale

Spatial resolution	Temporal resolution
Astrophysics, Radiation transport, Combustion, Chemistry, Molecular dynamics, Fusion, …	Molecular dynamics, Climate, Fusion, …

Applications are pushing data set sizes toward Exascale

Spatial resolution	Temporal resolution
Astrophysics, Radiation transport, Combustion, Chemistry, Molecular dynamics, Fusion, …	Molecular dynamics, Climate, Fusion, …
Multivariate Climate, Radiation transport, V&V, …	

Applications are pushing data set sizes toward Exascale

Spatial resolution	Temporal resolution
Astrophysics, Radiation transport, Combustion, Chemistry, Molecular dynamics, Fusion, …	Molecular dynamics, Climate, Fusion, …
Multivariate	**Ensembles**
Climate, Radiation transport, V&V, …	Parameter Studies, Climate, V&V, …

Resolution does matter

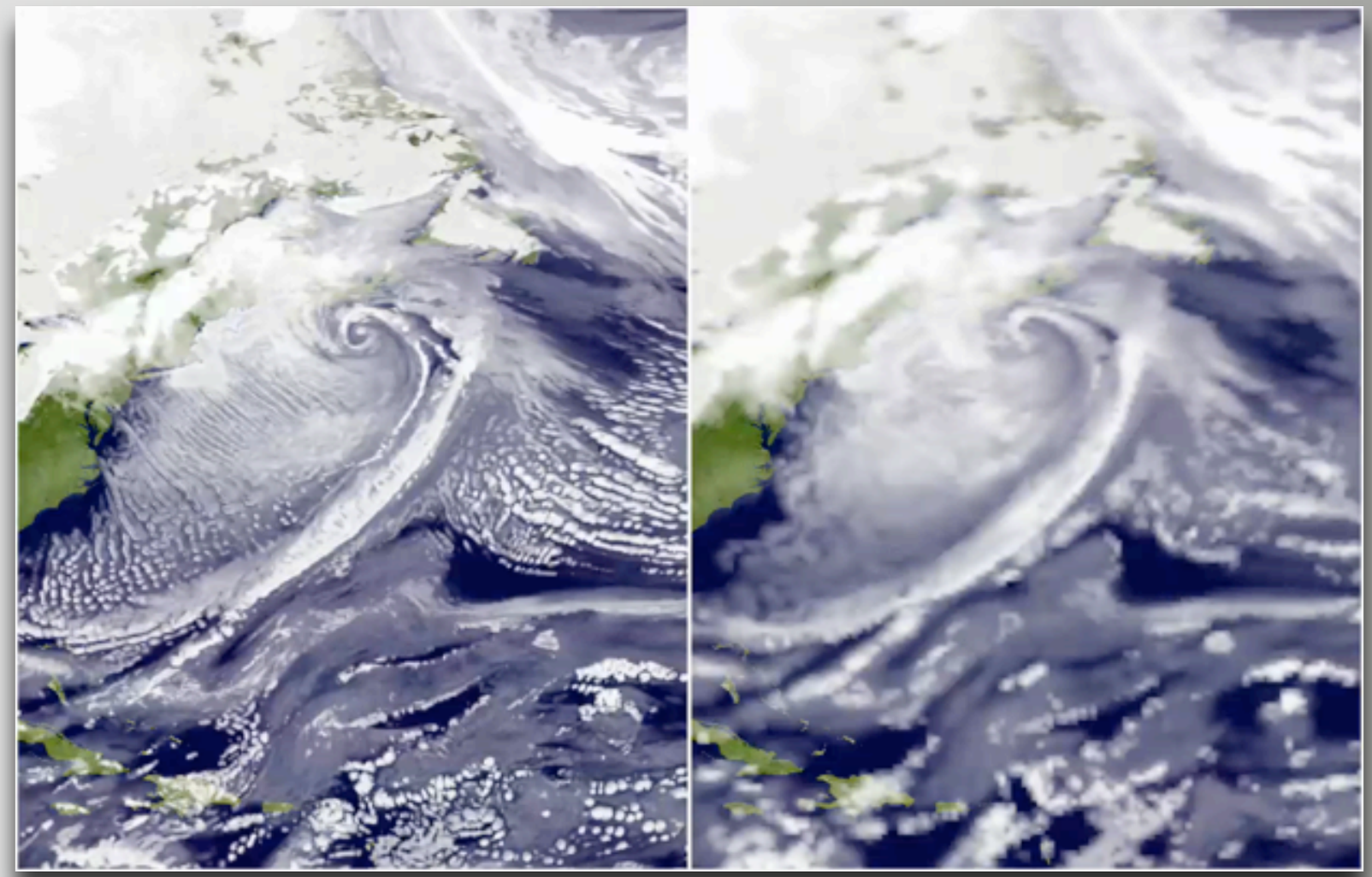
- NASA climate simulation run on a large portion of Jaguar

3 km

27 km

Resolution does matter

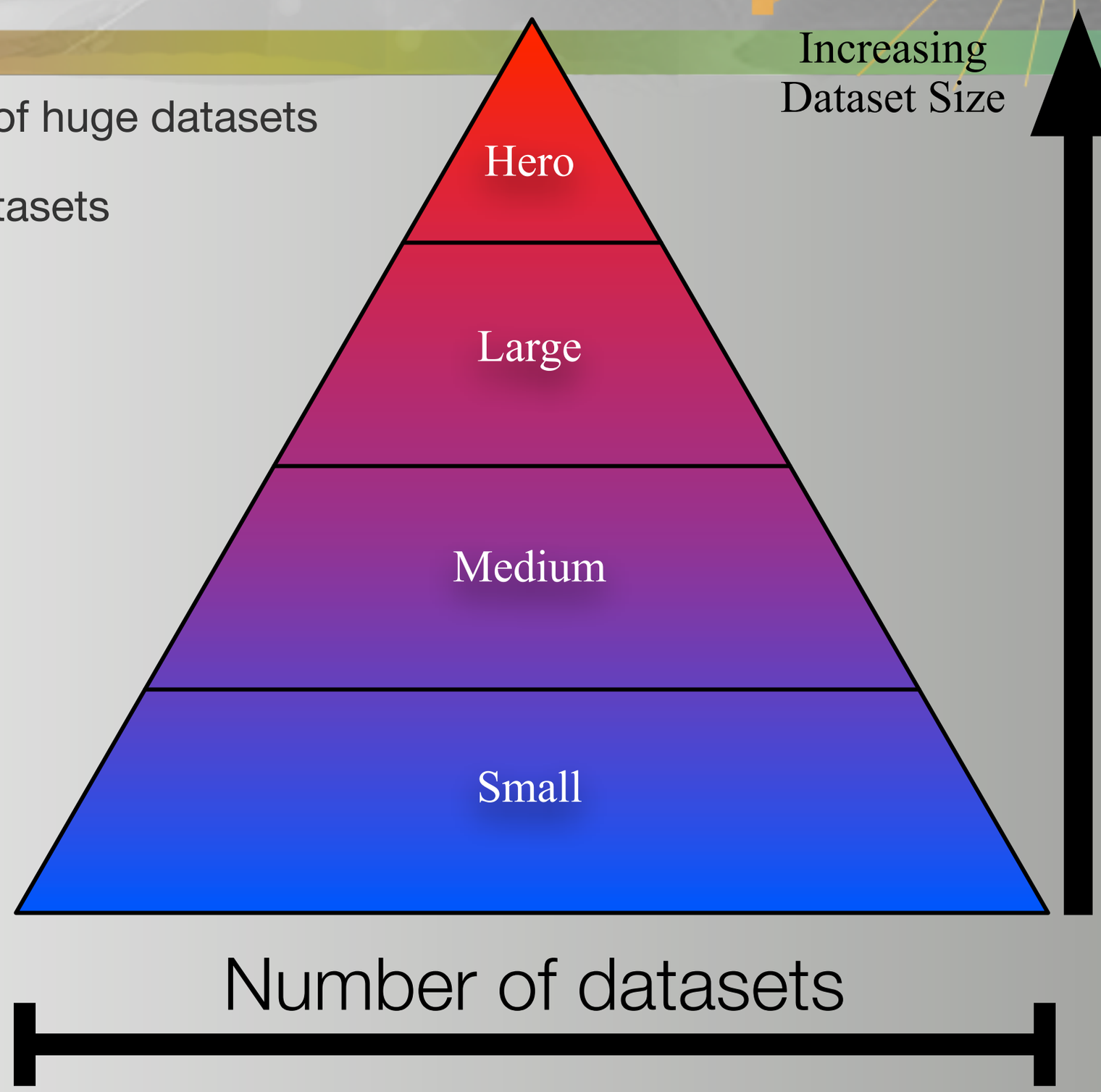
- NASA climate simulation run on a large portion of Jaguar



3 km 27 km

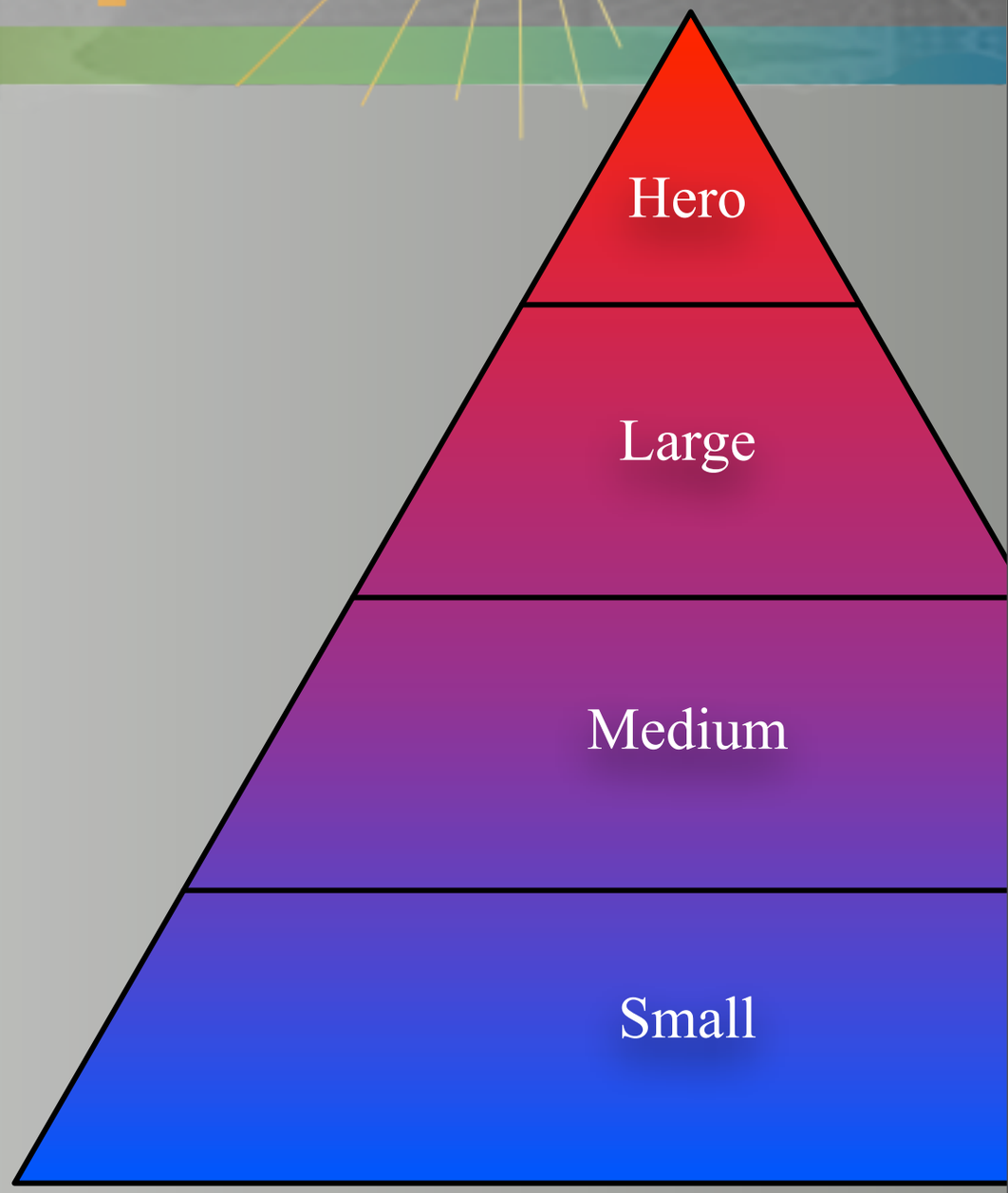
Dataset size growth

- Small number of huge datasets
- Many small datasets



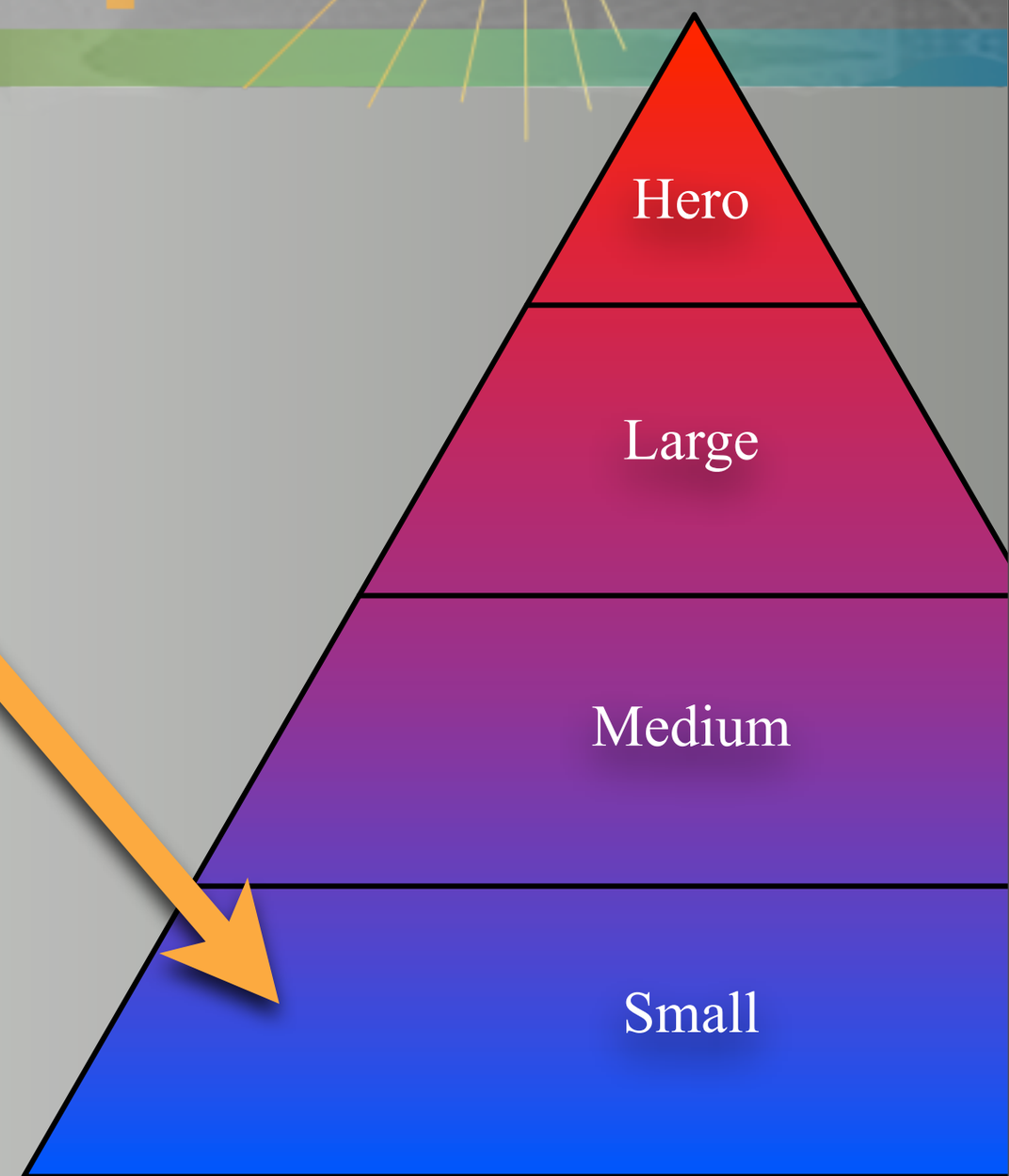
Data size categories

- Small:
  - Data is small enough to easily move anywhere
  - Analysis/vis is generally done on local workstations
- Medium:
  - Data won't fit on local workstations
  - Generally have to process on "fat" SMP systems
- Large:
  - Rather painful to move
  - Requires distributed parallelism
- Hero:
  - Functionally impossible to move
  - Only approachable on largest computational platforms



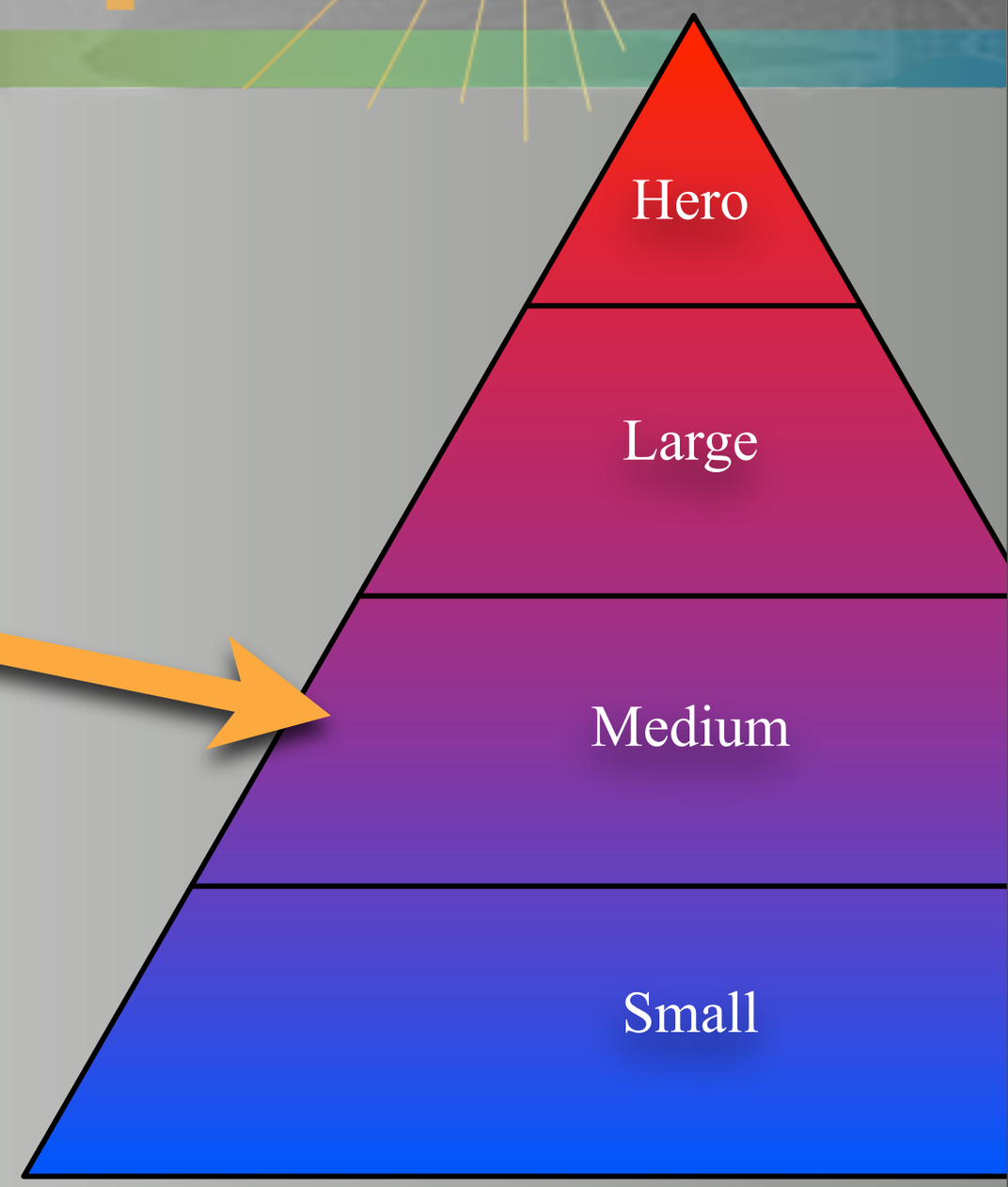
Data size categories

- Small:
  - Data is small enough to easily move anywhere
  - Analysis/vis is generally done on local workstations
- Medium:
  - Data won't fit on local workstations
  - Generally have to process on "fat" SMP systems
- Large:
  - Rather painful to move
  - Requires distributed parallelism
- Hero:
  - Functionally impossible to move
  - Only approachable on largest computational platforms



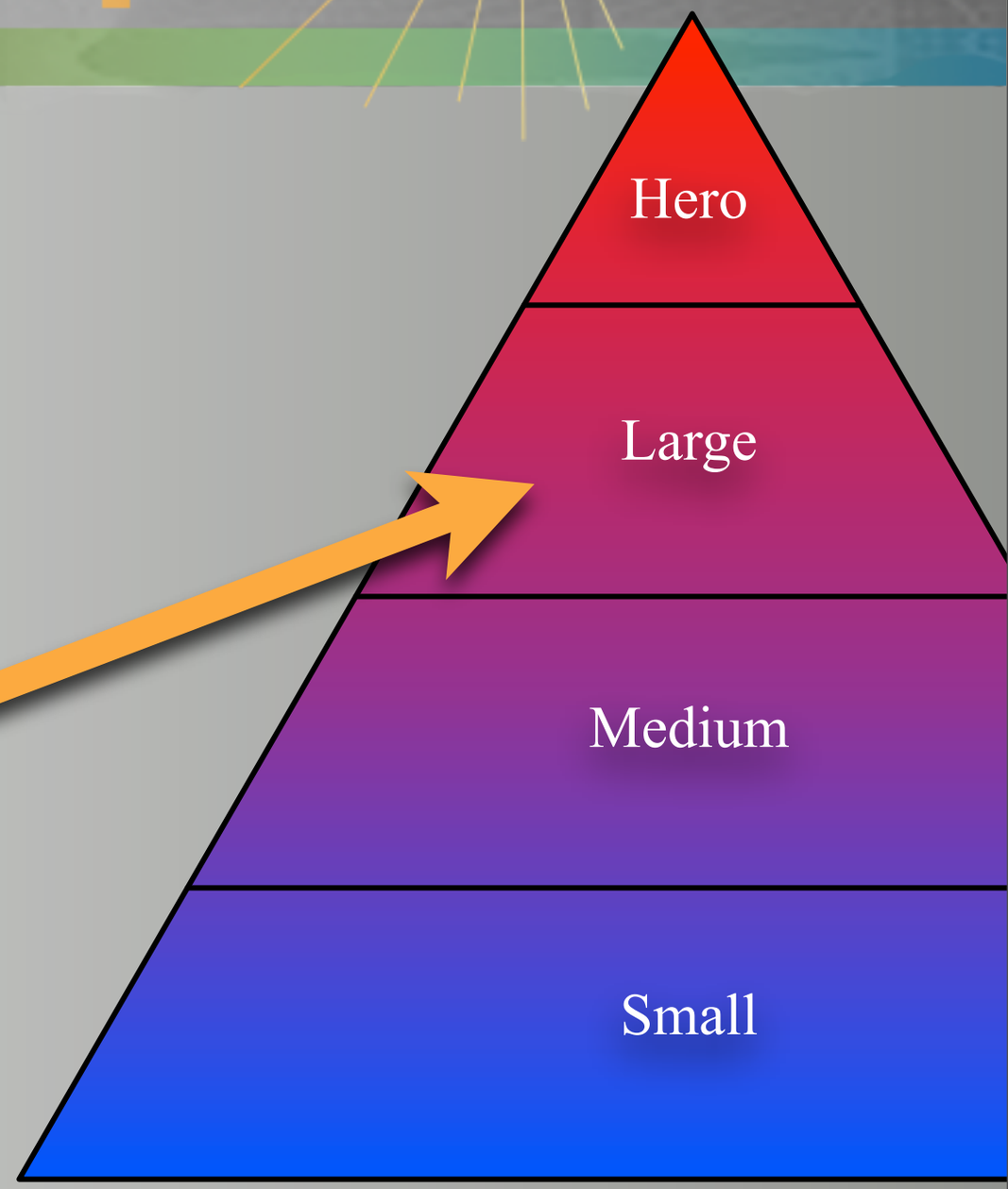
Data size categories

- Small:
  - Data is small enough to easily move anywhere
  - Analysis/vis is generally done on local workstations
- Medium:
  - Data won't fit on local workstations
  - Generally have to process on "fat" SMP systems
- Large:
  - Rather painful to move
  - Requires distributed parallelism
- Hero:
  - Functionally impossible to move
  - Only approachable on largest computational platforms



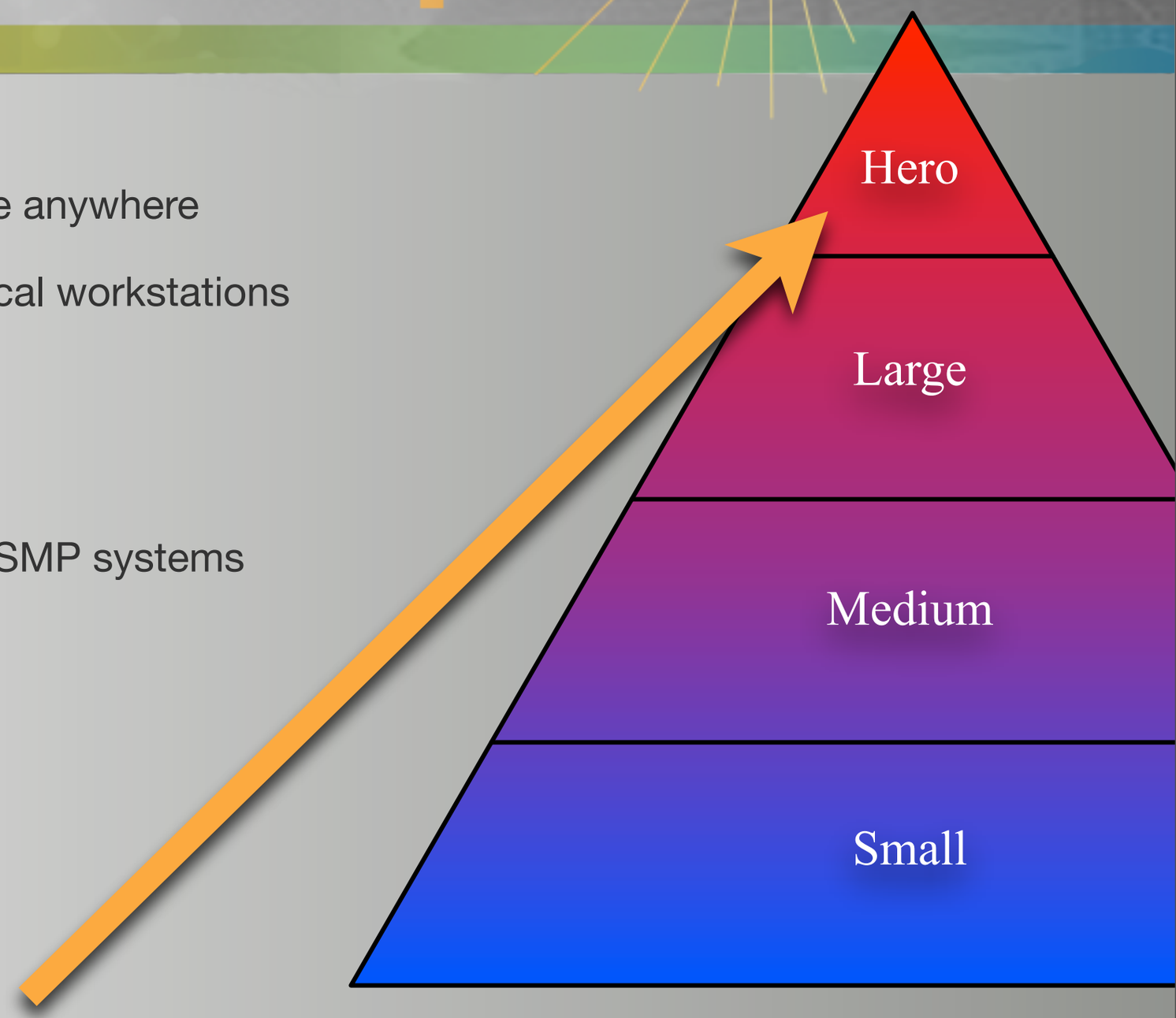
Data size categories

- Small:
  - Data is small enough to easily move anywhere
  - Analysis/vis is generally done on local workstations
- Medium:
  - Data won't fit on local workstations
  - Generally have to process on "fat" SMP systems
- Large:
  - Rather painful to move
  - Requires distributed parallelism
- Hero:
  - Functionally impossible to move
  - Only approachable on largest computational platforms

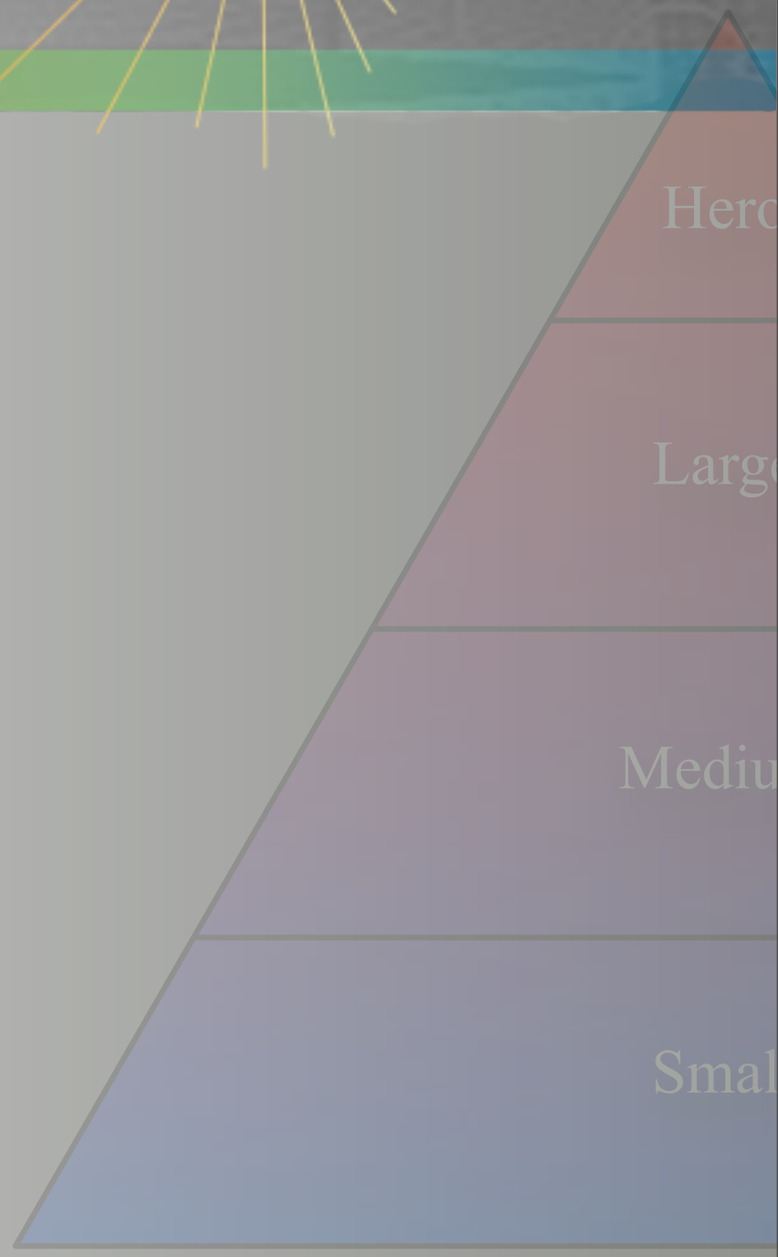


Data size categories

- Small:
  - Data is small enough to easily move anywhere
  - Analysis/vis is generally done on local workstations
- Medium:
  - Data won't fit on local workstations
  - Generally have to process on "fat" SMP systems
- Large:
  - Rather painful to move
  - Requires distributed parallelism
- Hero:
  - Functionally impossible to move
  - Only approachable on largest computational platforms

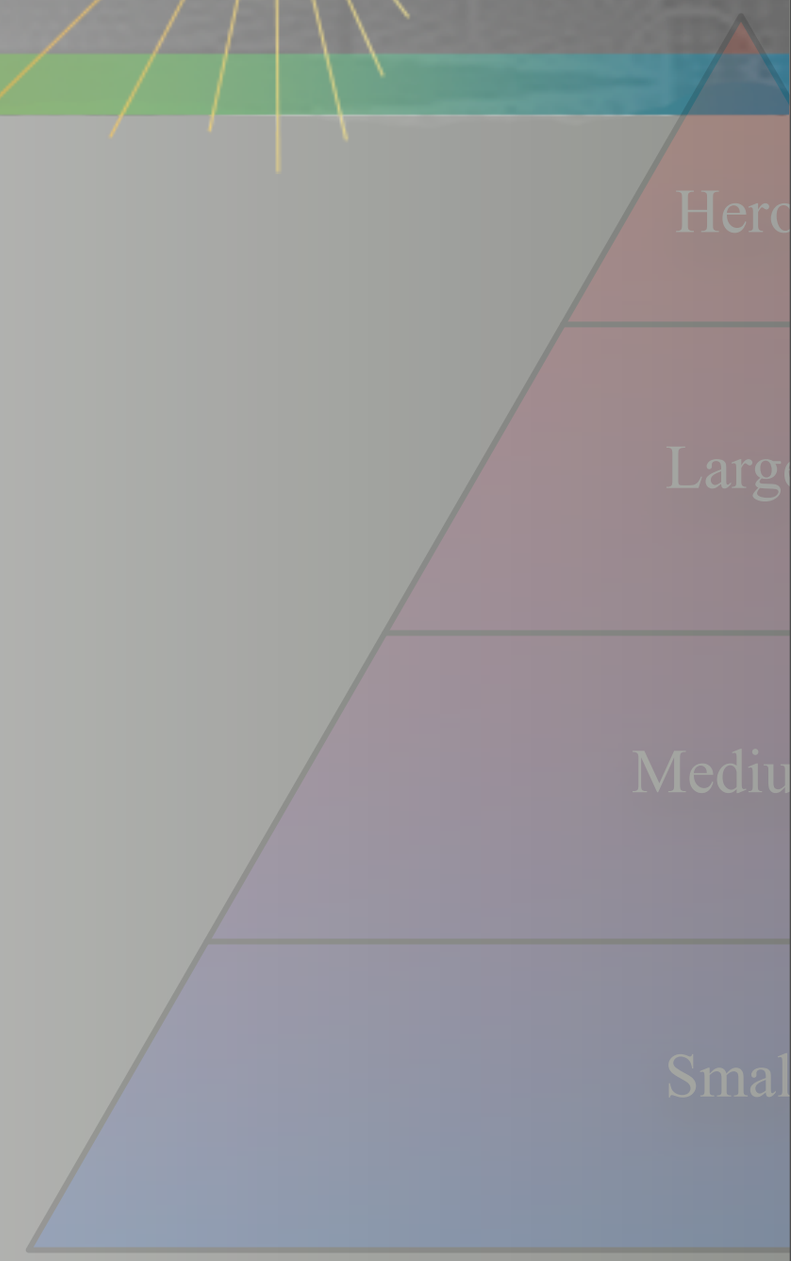


Shared memory is useful for other activities



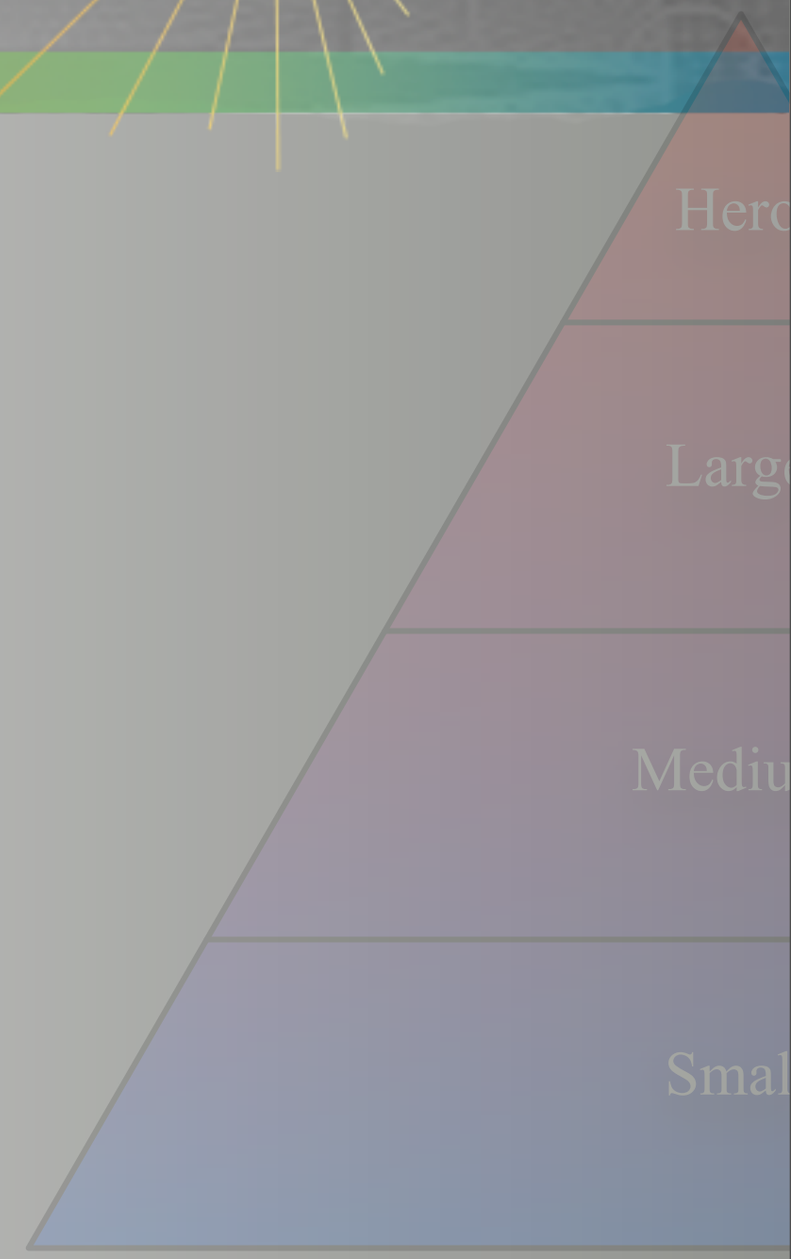
Shared memory is useful for other activities

- Many HPC users can use distributed memory analysis:
  - data parallel, space/time parallel



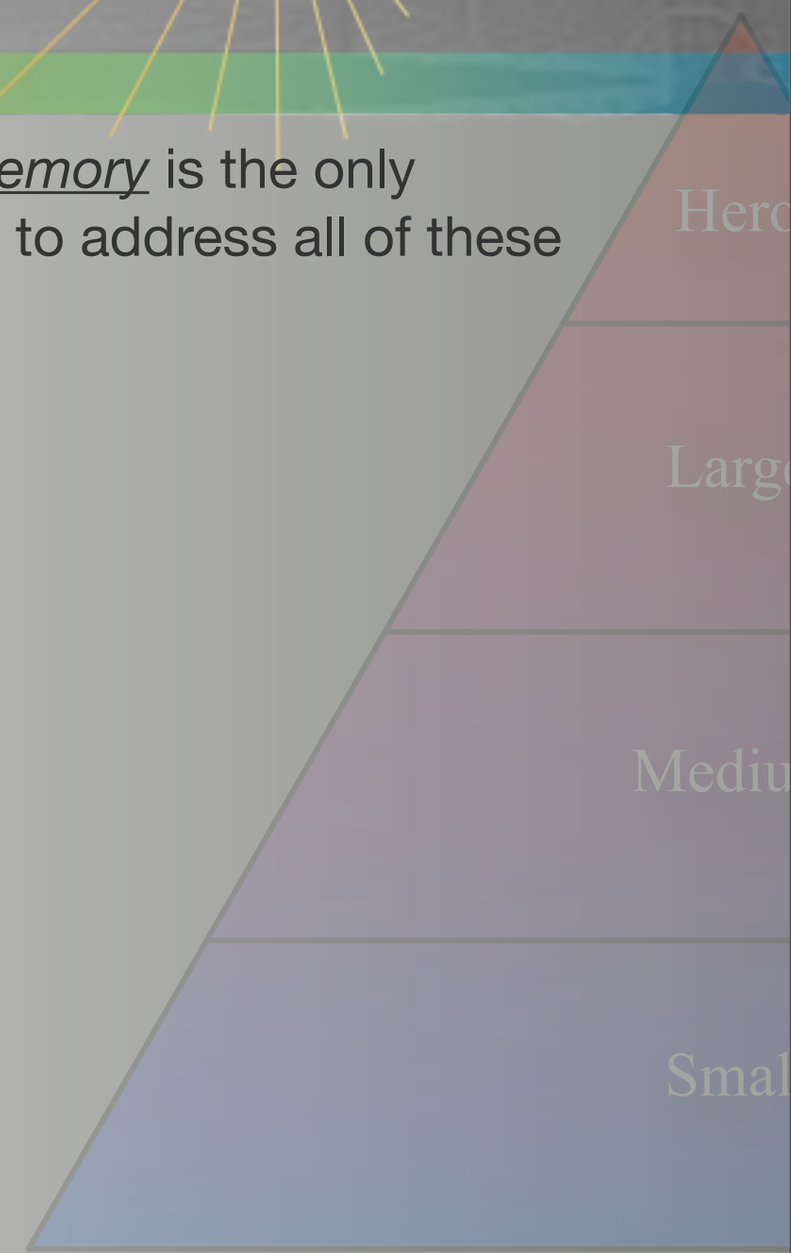
Shared memory is useful for other activities

- Many HPC users can use distributed memory analysis:
  - data parallel, space/time parallel
- However, many general and statistical analysis algorithms favor large shared memory
  - Document clustering/searching
  - Generalized graph structures
  - Bioinformatics, genomics
  - …



Shared memory is useful for other activities

- Many HPC users can use distributed memory analysis:
  - data parallel, space/time parallel
- However, many general and statistical analysis algorithms favor large shared memory
  - Document clustering/searching
  - Generalized graph structures
  - Bioinformatics, genomics
  - …
- *Large shared memory* is the only reasonable way to address all of these needs



Shared memory is useful for other activities

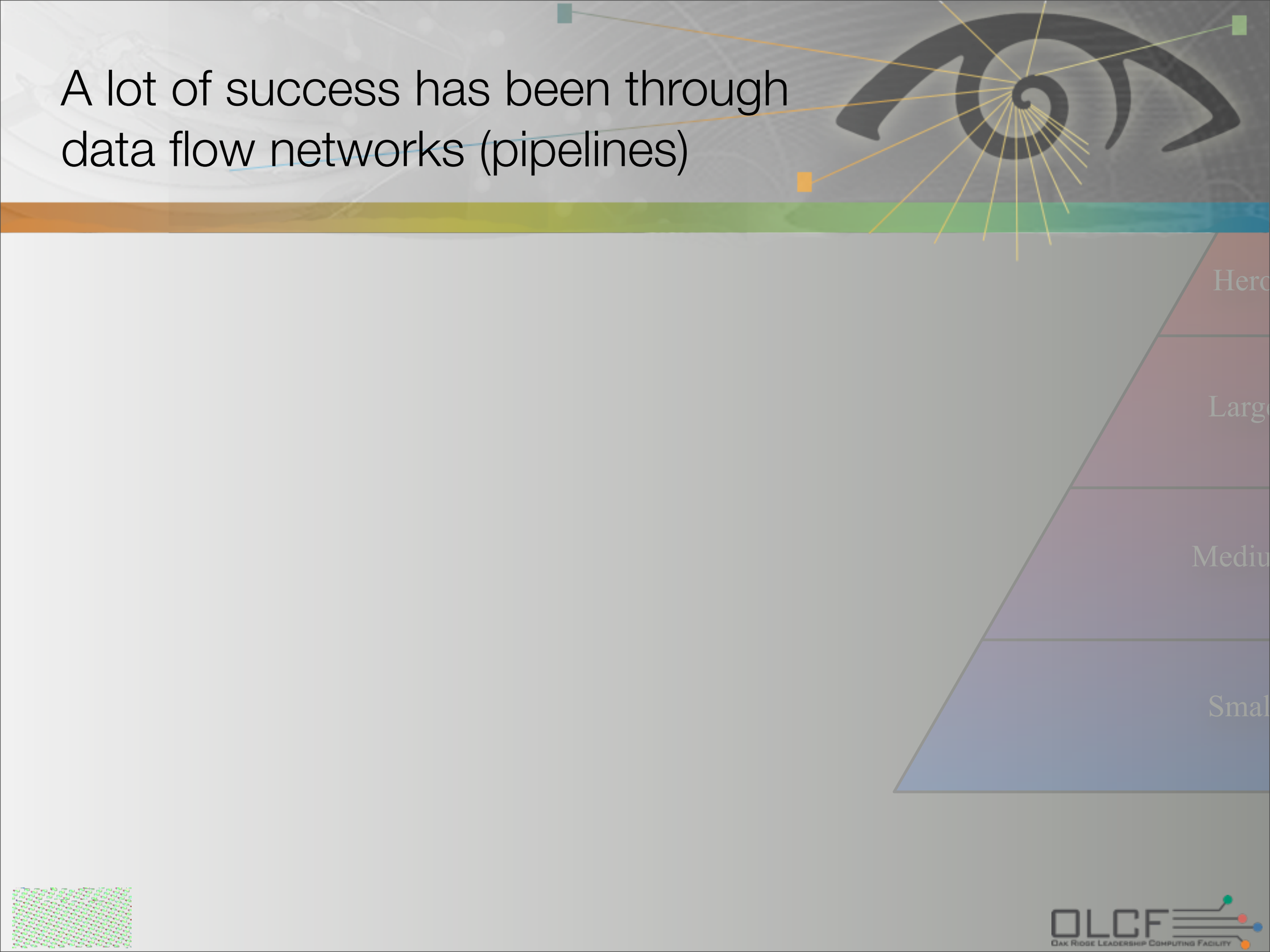
- Many HPC users can use distributed memory analysis:
  - data parallel, space/time parallel
- However, many general and statistical analysis algorithms favor large shared memory
  - Document clustering/searching
  - Generalized graph structures
  - Bioinformatics, genomics
  - …
- *Large shared memory* is the only reasonable way to address all of these needs
- University of Tennessee partnered with SGI to site an UltraViolet 1000 machine:
  - Large memory single-system image through NUMA
  - A "better" cluster architecture, accelerating distributed memory MPI

Shared memory is useful for other activities

- Many HPC users can use distributed memory analysis:
  - data parallel, space/time parallel
- However, many general and statistical analysis algorithms favor large shared memory
  - Document clustering/searching
  - Generalized graph structures
  - Bioinformatics, genomics
  - …

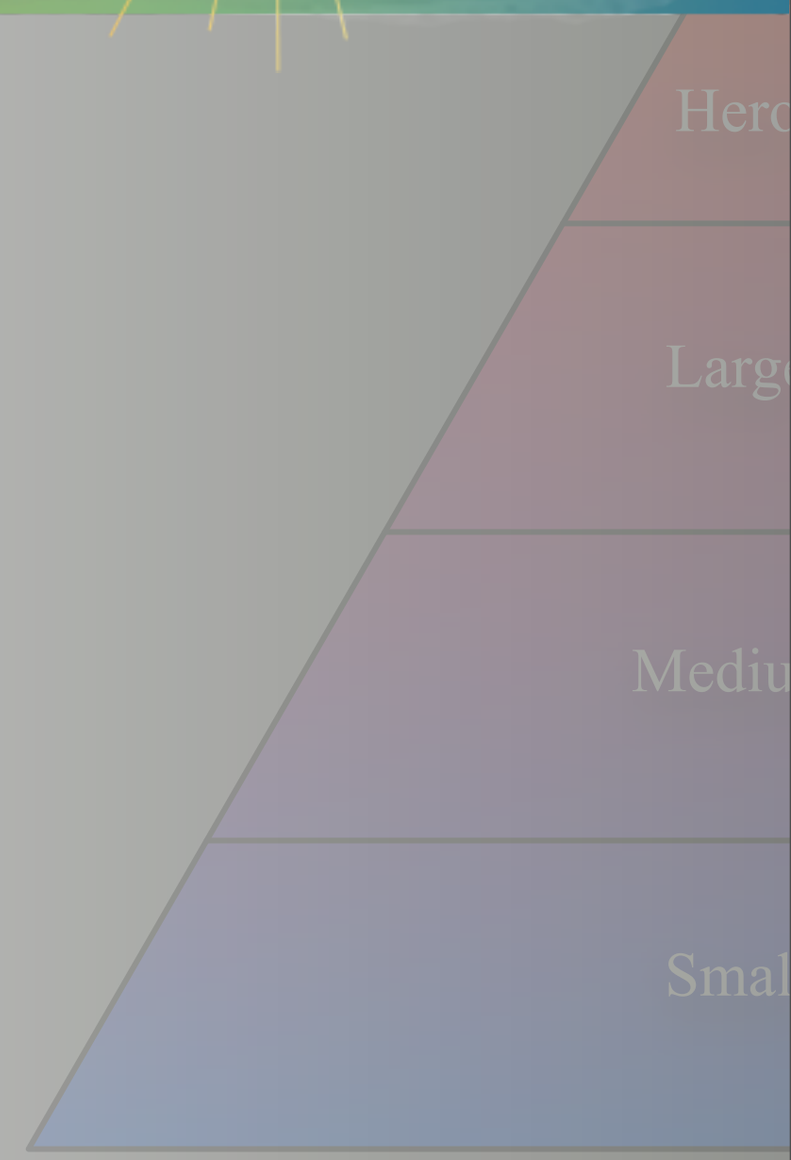
- *Large shared memory* is the only reasonable way to address all of these needs
- University of Tennessee partnered with SGI to site an UltraViolet 1000 machine:
  - Large memory single-system image through NUMA
  - A "better" cluster architecture, accelerating distributed memory MPI
- NSF TeraGrid resource dedicated to analysis and visualization:
  - 1,024 cores (Intel Nehalem EX)
  - 4 TB Global Shared Memory
  - 16 NVIDIA Fermi Tesla GPUs

A lot of success has been through data flow networks (pipelines)



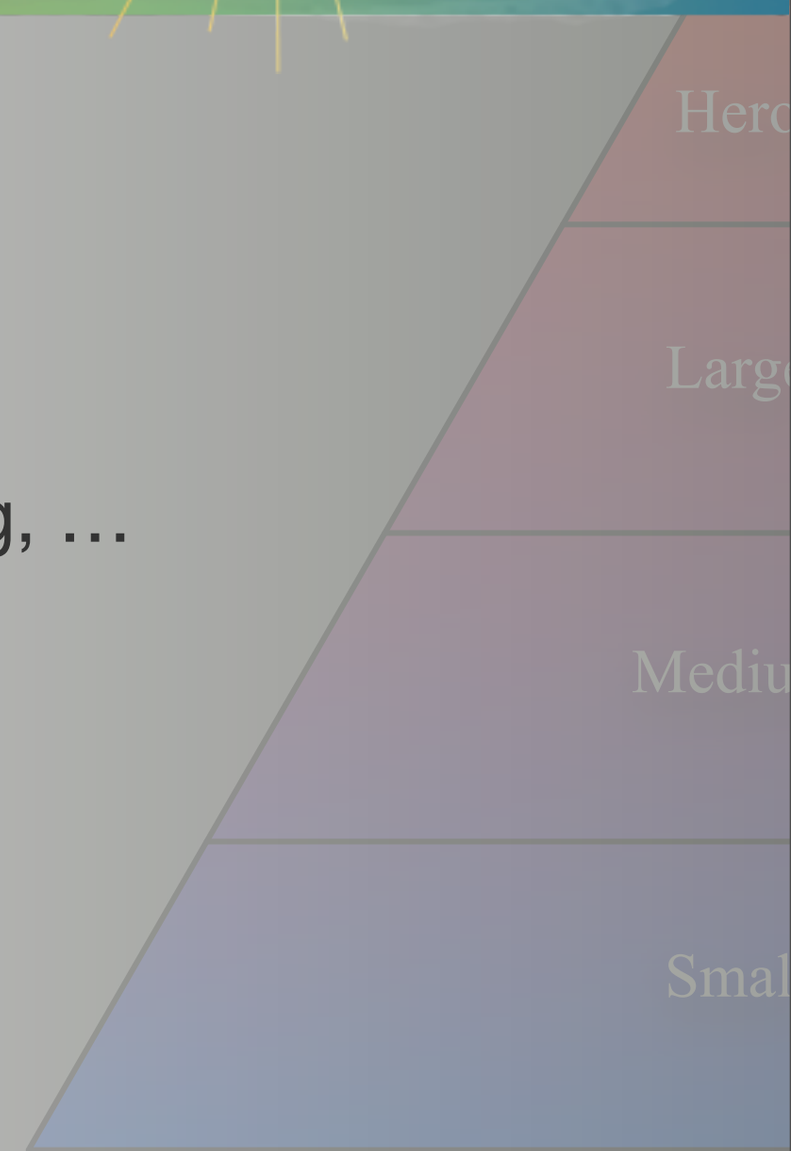
A lot of success has been through data flow networks (pipelines)

- Different input data formats:
  - NetCDF, HDF, text, CSV, PDB, …



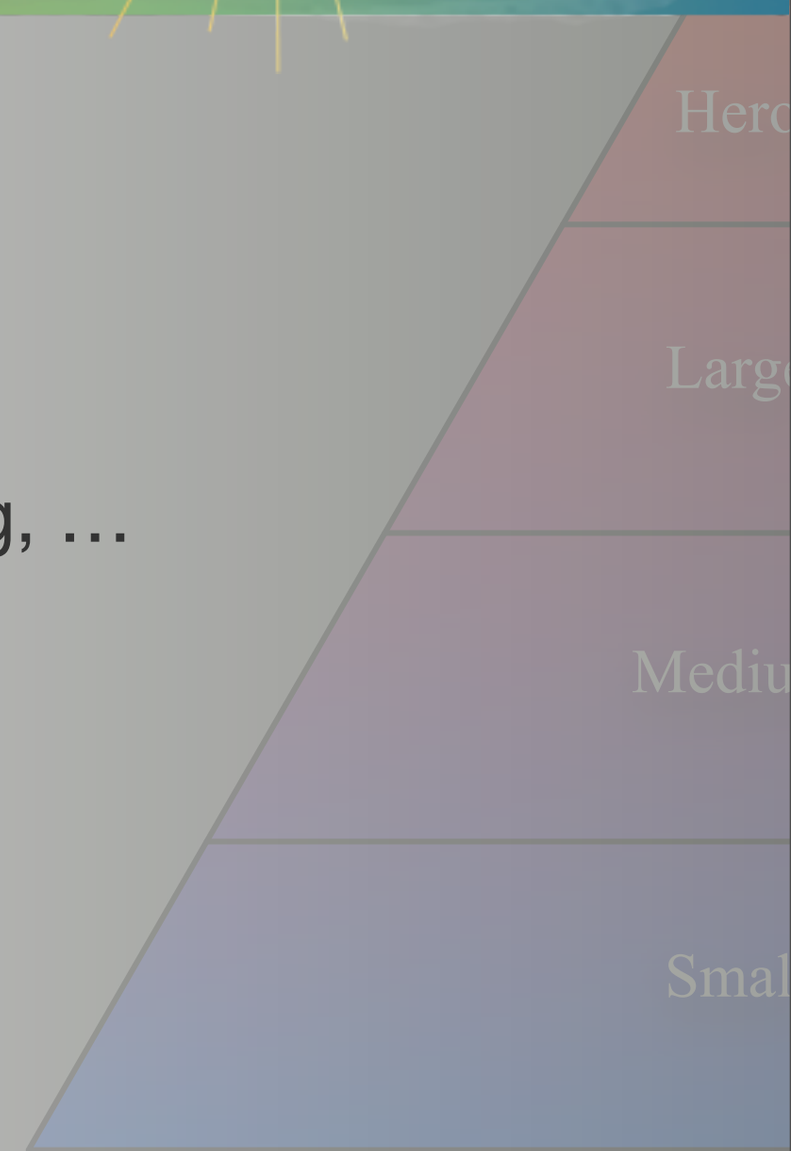
A lot of success has been through data flow networks (pipelines)

- Different input data formats:
  - NetCDF, HDF, text, CSV, PDB, …
- Different types of data operations:
  - Slicing, resampling, mesh transforms, filtering, …



A lot of success has been through data flow networks (pipelines)

- Different input data formats:
  - NetCDF, HDF, text, CSV, PDB, …
- Different types of data operations:
  - Slicing, resampling, mesh transforms, filtering, …
- Different ways of plotting:
  - Pseudocolor, isosurfaces, volume rendering



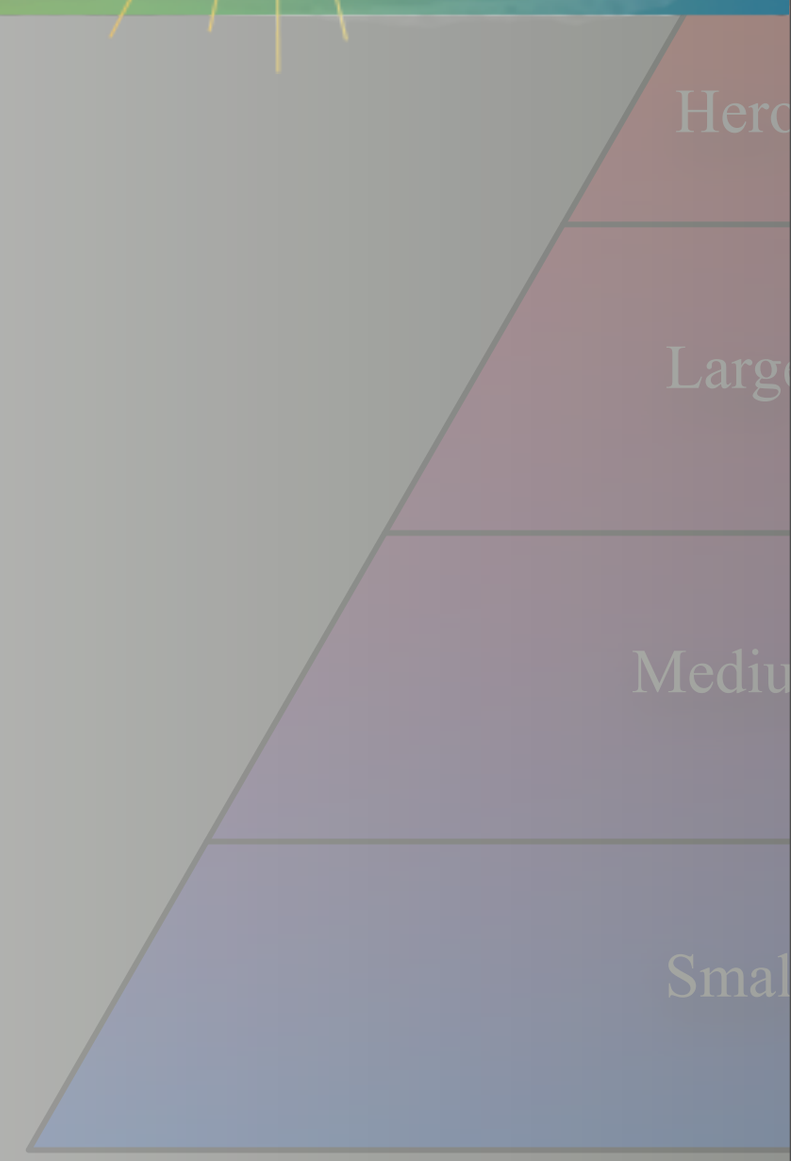
A lot of success has been through data flow networks (pipelines)

- Different input data formats:
  - NetCDF, HDF, text, CSV, PDB, …
- Different types of data operations:
  - Slicing, resampling, mesh transforms, filtering, …
- Different ways of plotting:
  - Pseudocolor, isosurfaces, volume rendering

These are independent of each other!

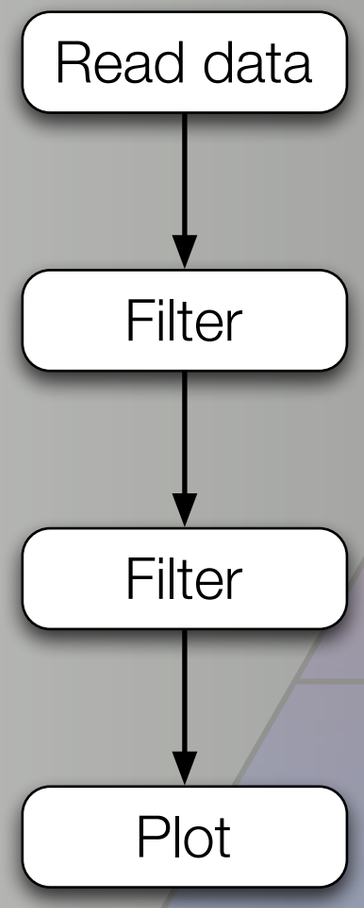
Make these independent modules

- Data reading
- Data operations
- Data plotting



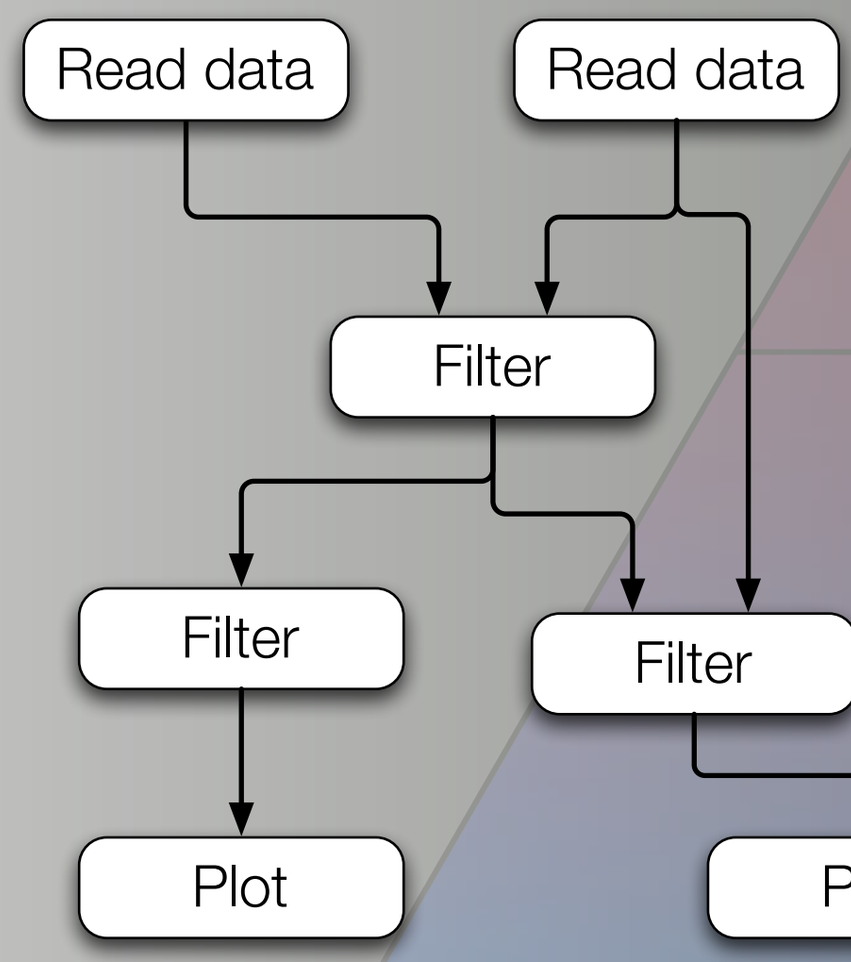
Make these independent modules

- Data reading
- Data operations
- Data plotting



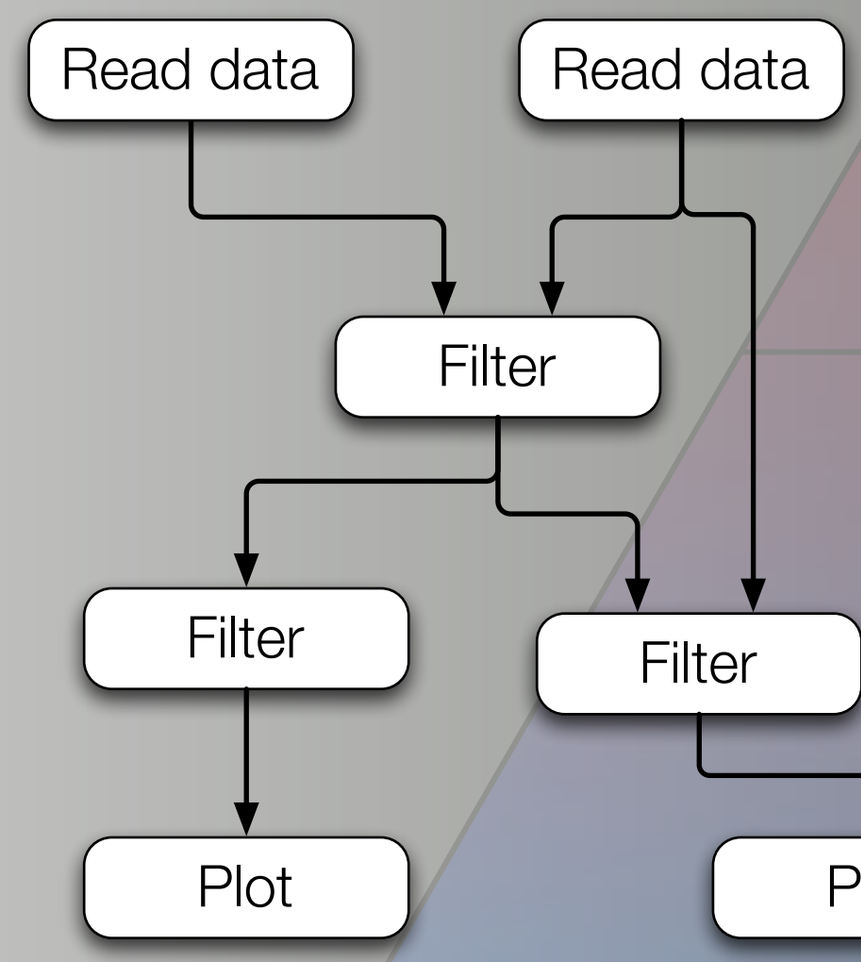
Make these independent modules

- Data reading
- Data operations
- Data plotting



Make these independent modules

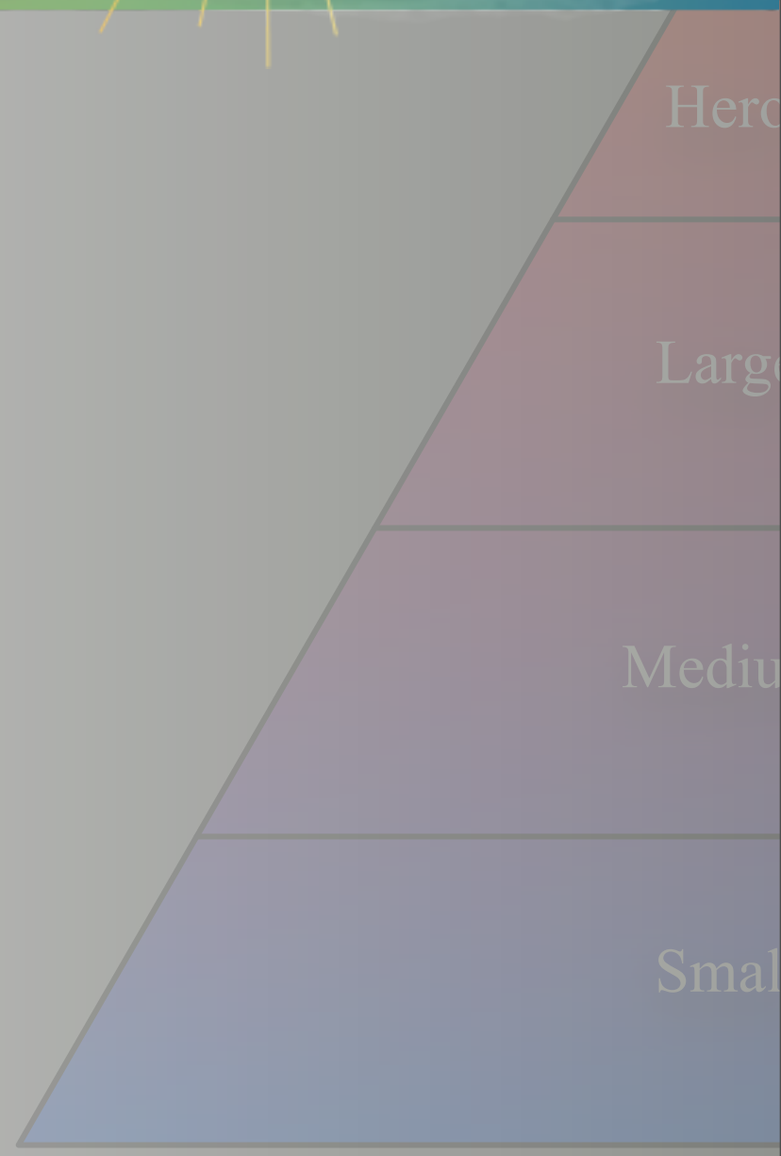
- Data reading
- Data operations
- Data plotting



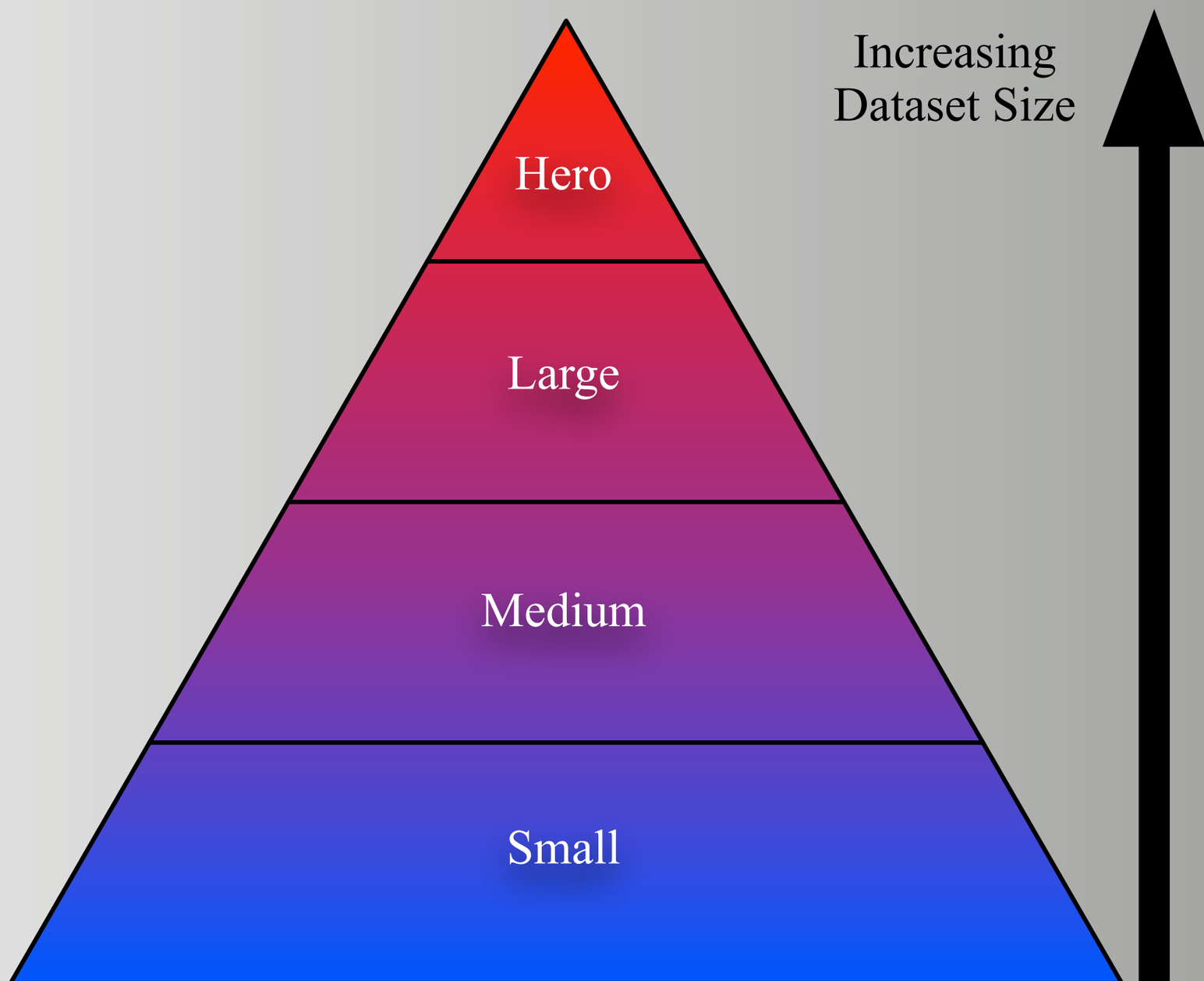
This is a data flow network

Many software systems use data flow networks

- VTK
- AVS/Express
- SCIRun
- …

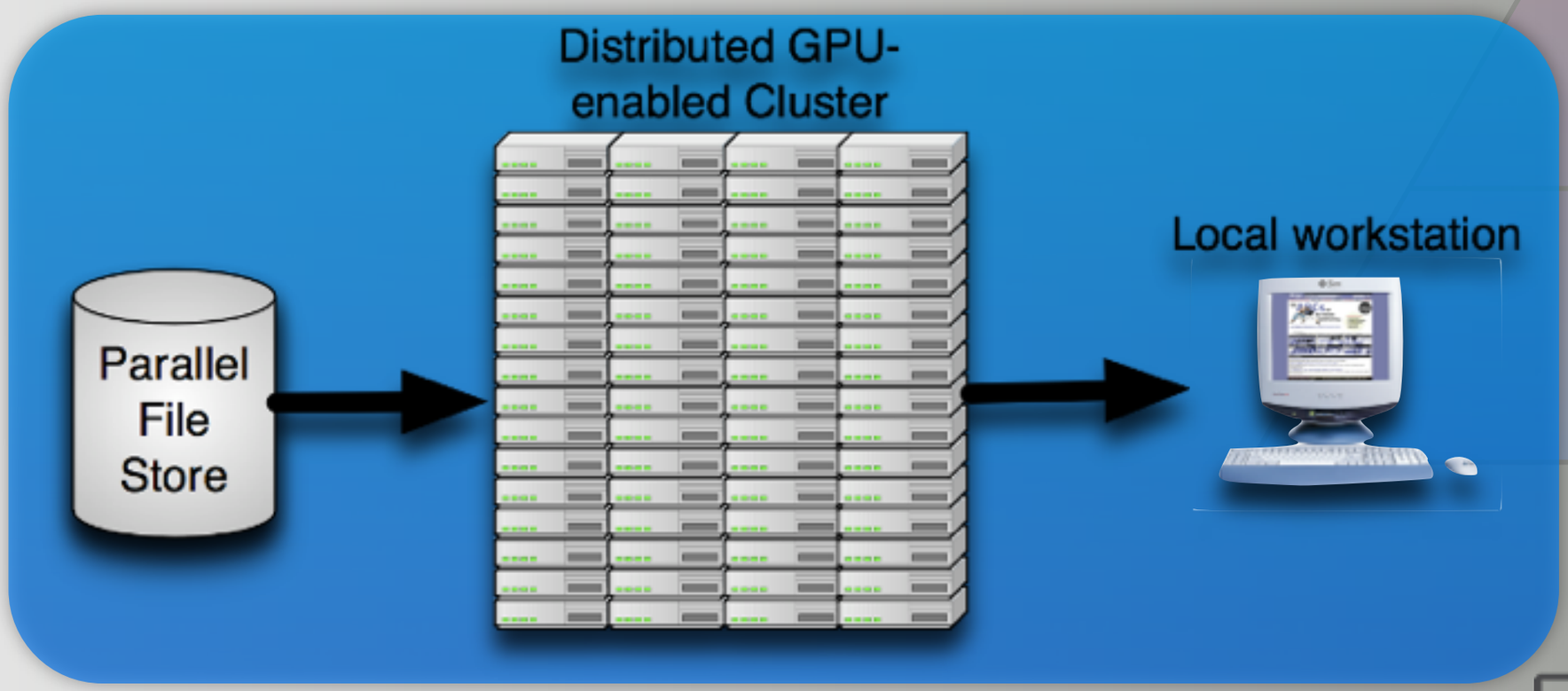


Fine for small to medium sized data



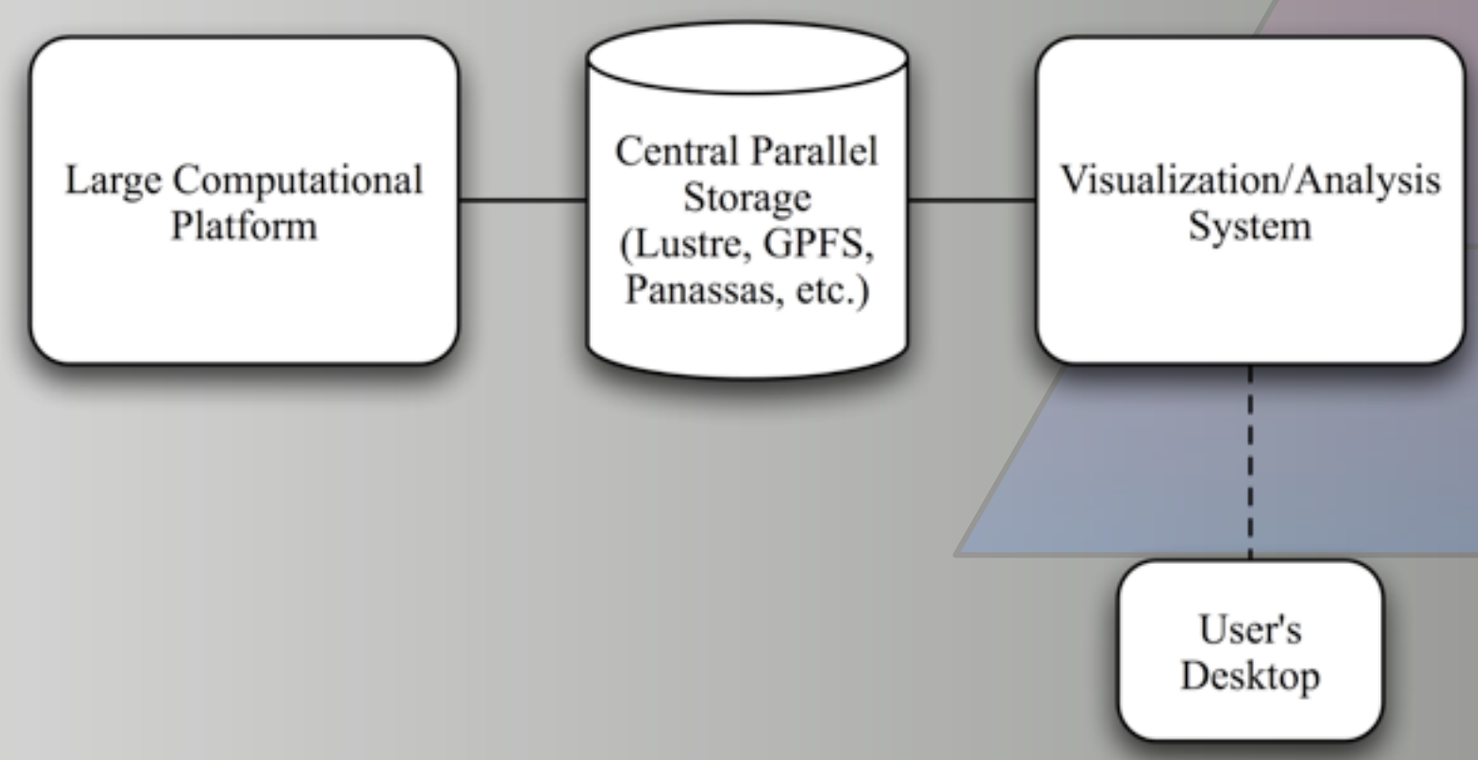
“Large” data – clusters and remote visualization

- Largest datasets require use of institutional resources
- Reduces data movement issues
- Allows exploitation of multiple GPUs
- Provides visualization to remote users
- Exploited by VisIt, ParaView, EnSight



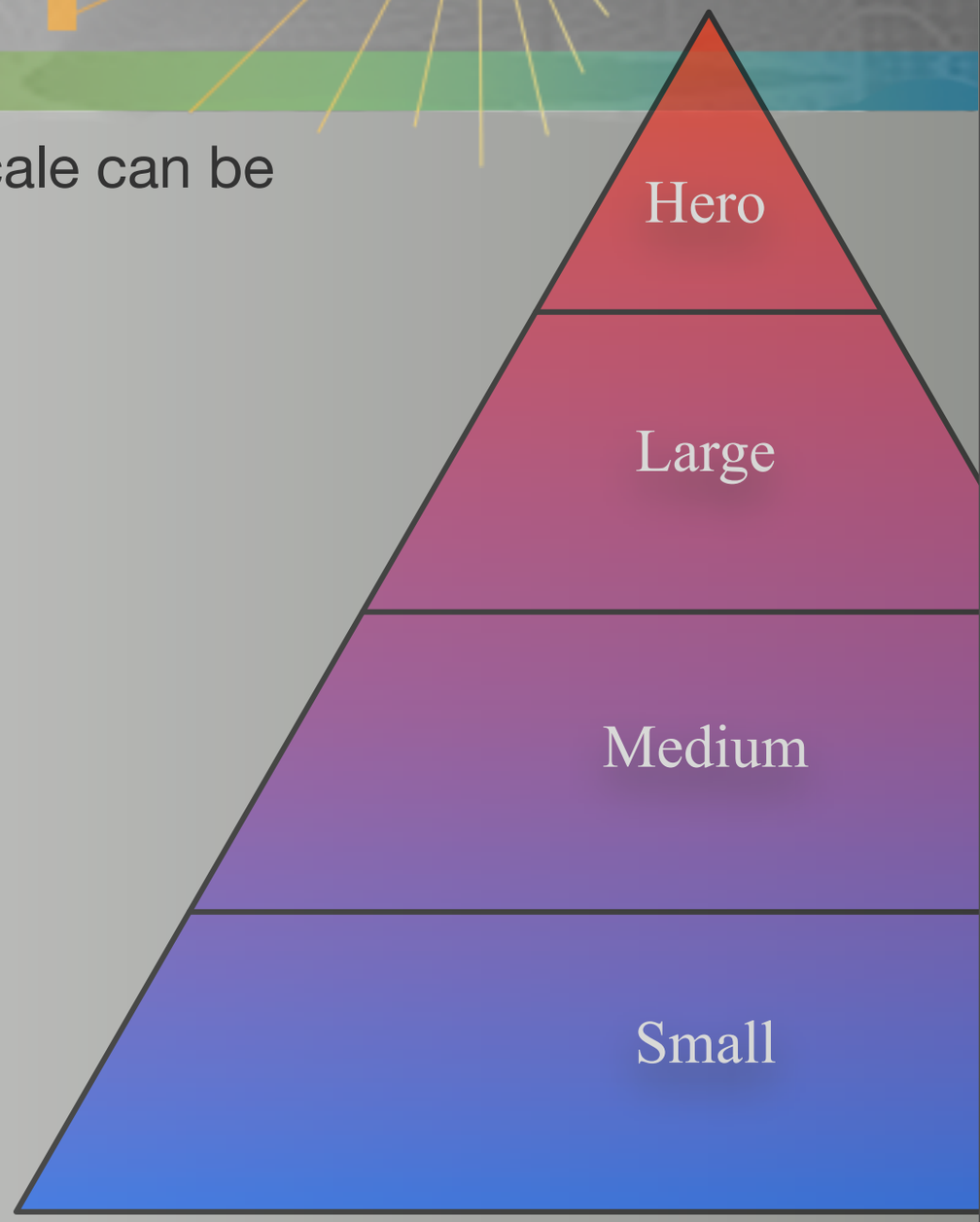
Central file store

- Scalable parallel throughput
  - Lustre, GPFS, Panassas, ...
- Shared between source and destination
- Provides “zero copy” access to simulation results
- Allows smart deployment of remote visualization capabilities



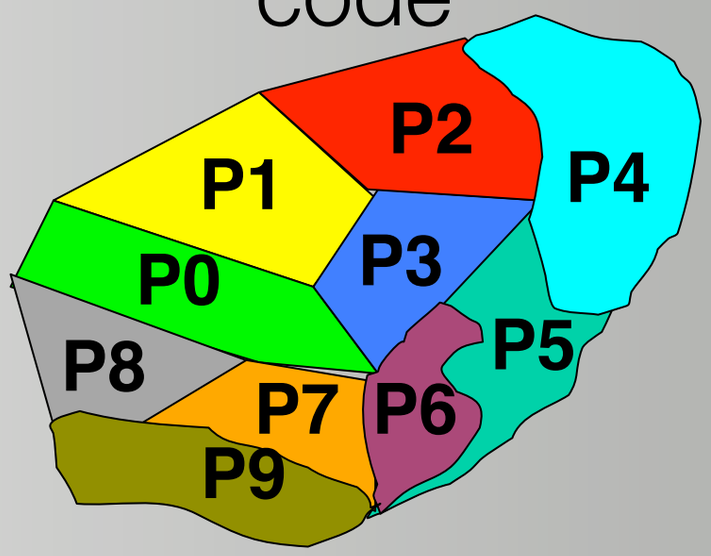
“Hero” data

- Purchasing separate analysis systems at the petascale can be prohibitively expensive: $5-20 million
- Working to move largest vis/analysis tools to HPC architecture
  - VisIt on Cray XT4/5
    - See Cray User’s Group ‘08
  - ParaView on Cray XT4/5
    - See Cray User’s Group ’09
  - Parallel R on Cray XT5
    - This year (hopefully)



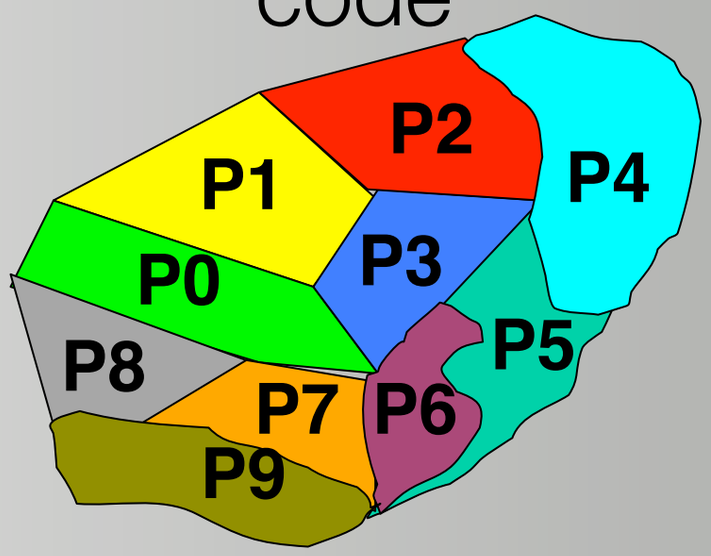
Parallel data flow networks

Parallel simulation code

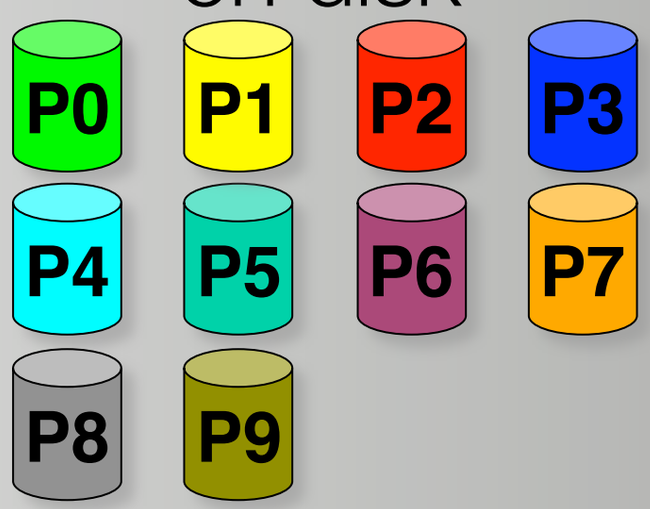


Parallel data flow networks

Parallel simulation code

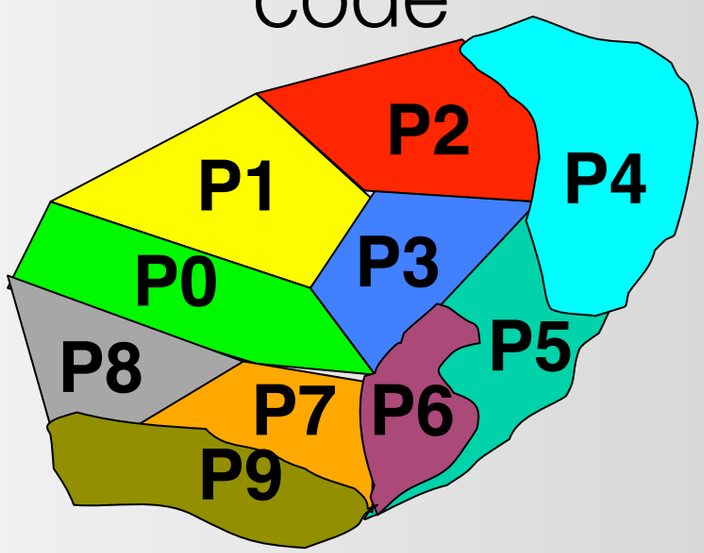


Data “chunks” on disk

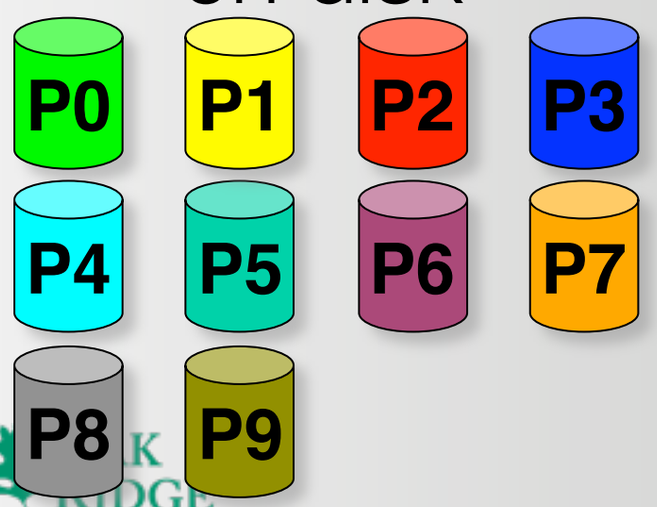


Parallel data flow networks

Parallel simulation code

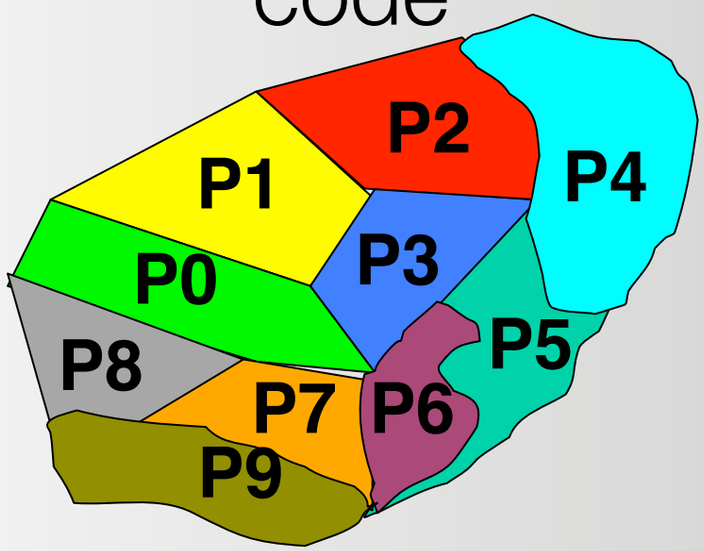


Data “chunks” on disk

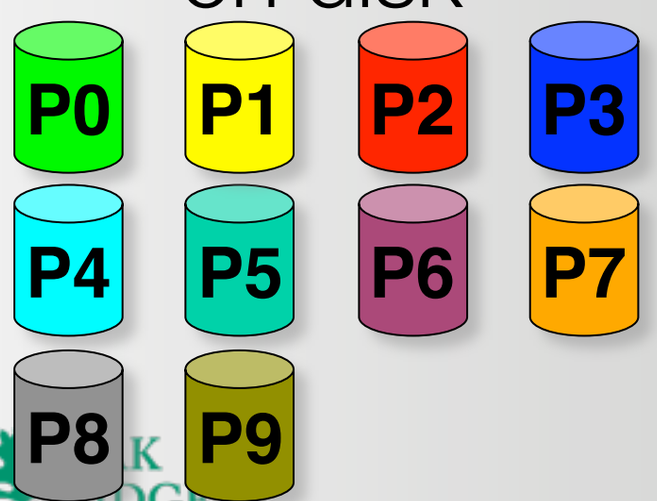


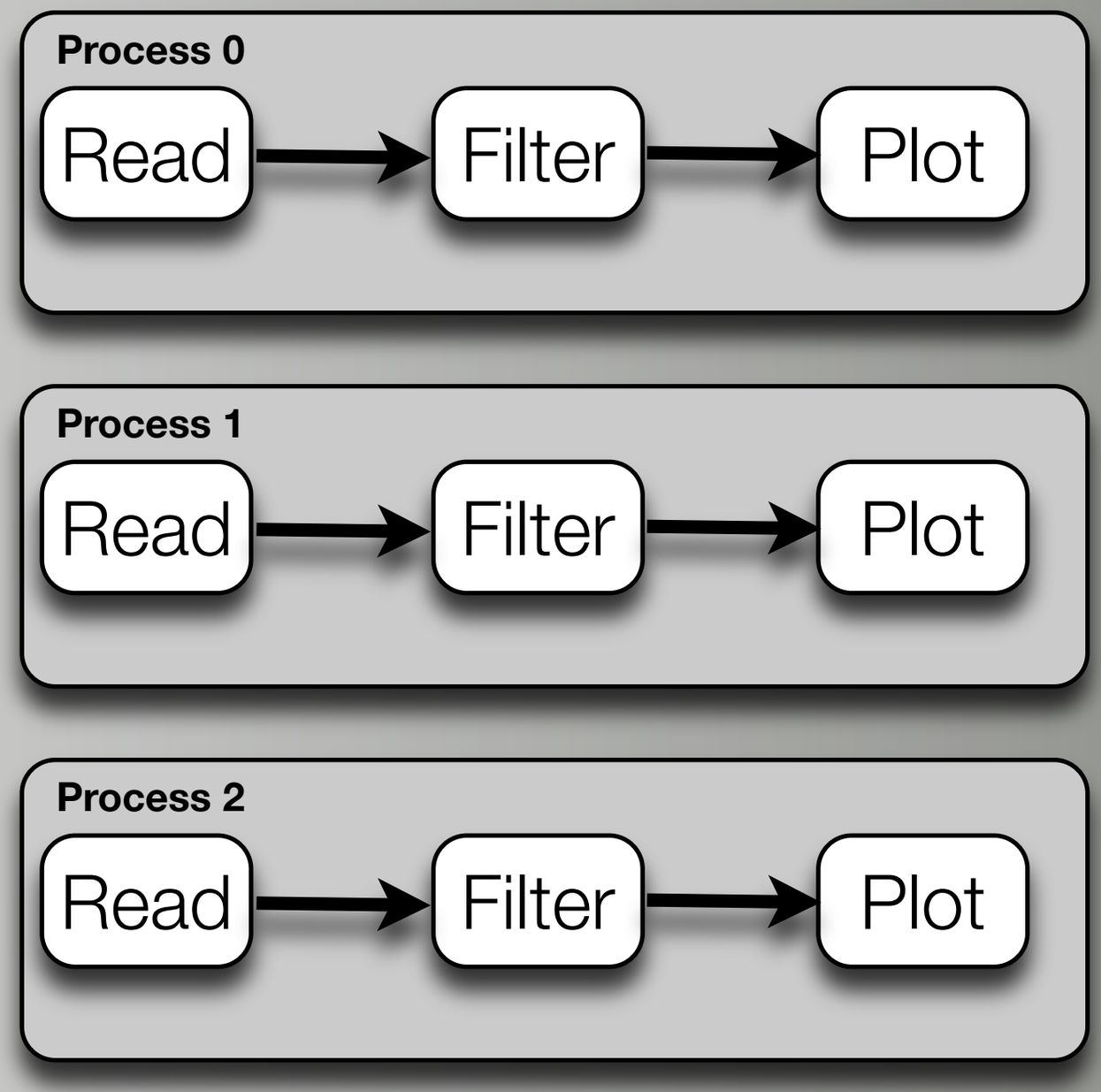
Parallel data flow networks

Parallel simulation code



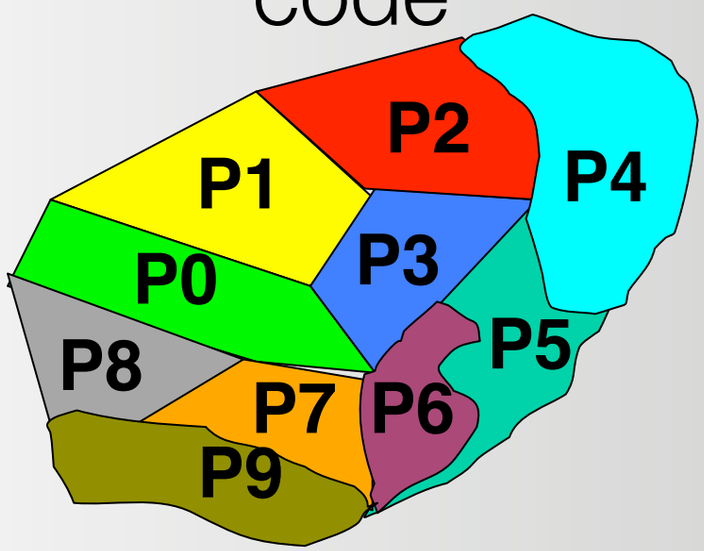
Data "chunks" on disk



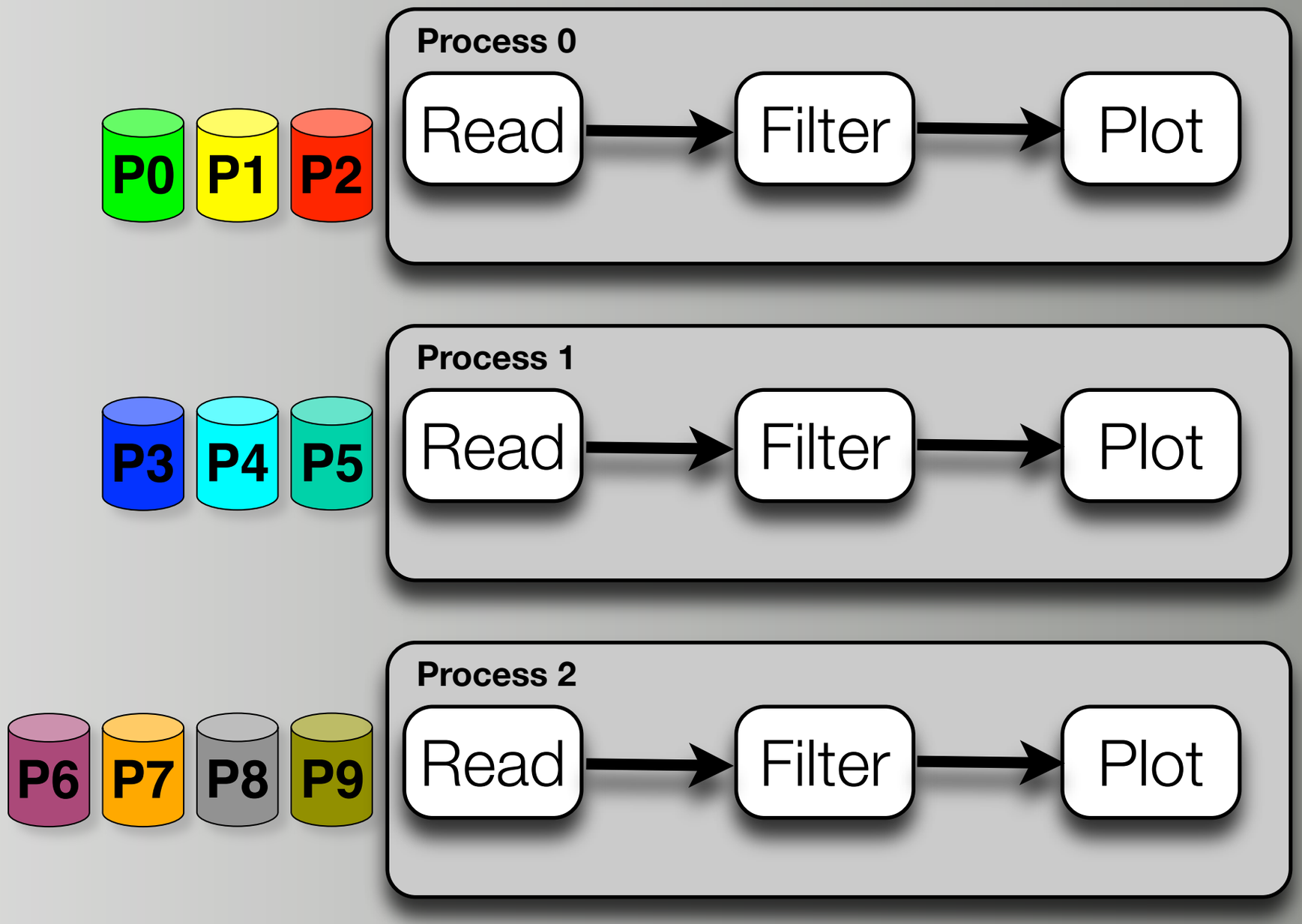


Parallel data flow networks

Parallel simulation code

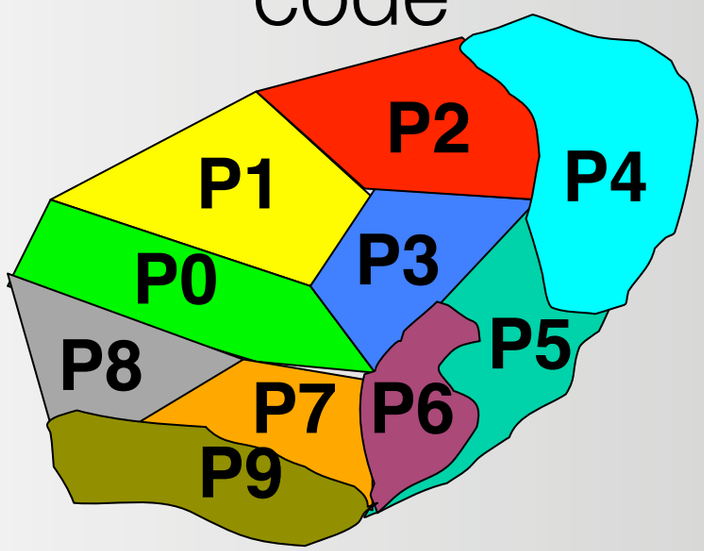


Data “chunks” on disk

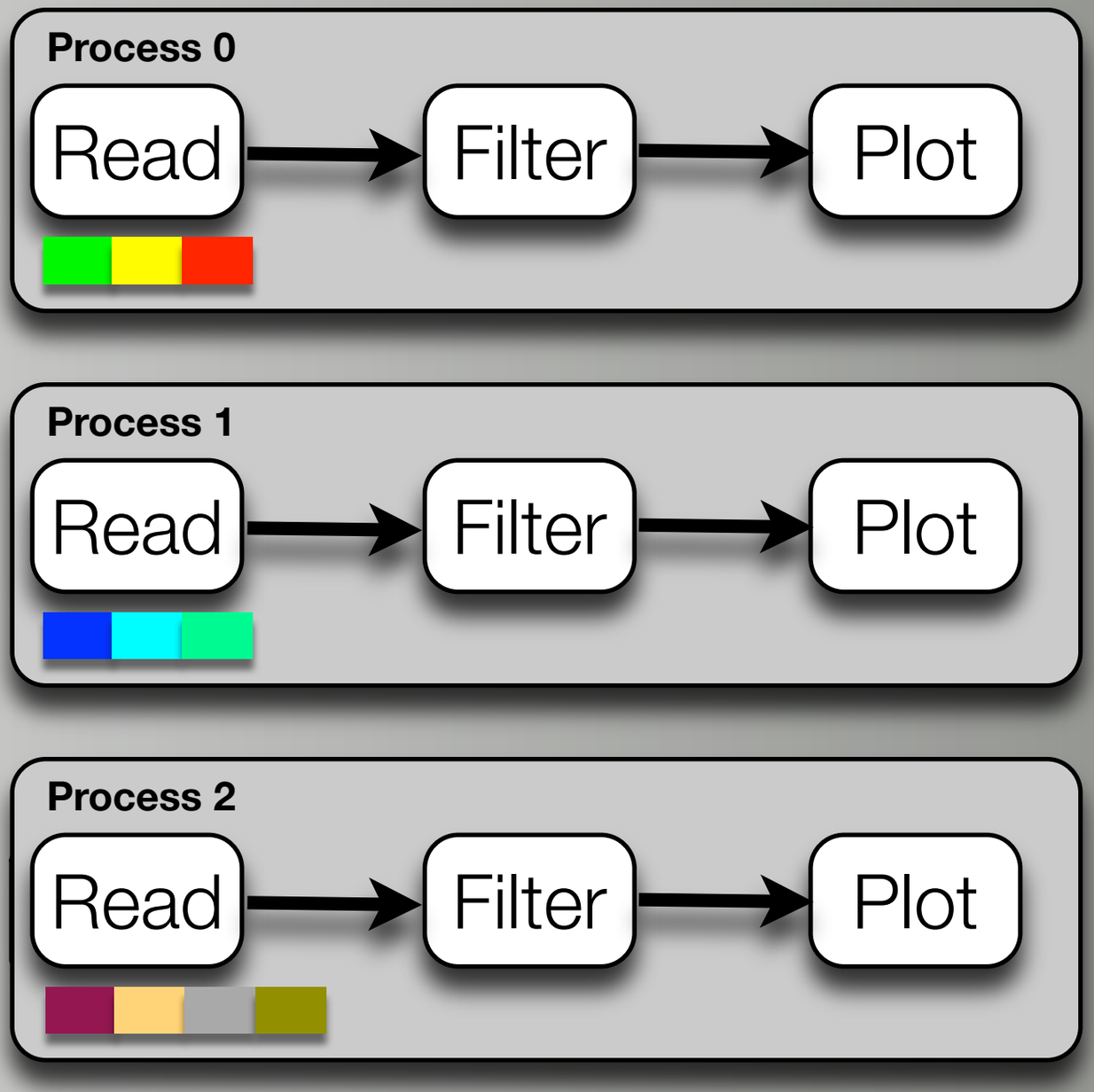


Parallel data flow networks

Parallel simulation code

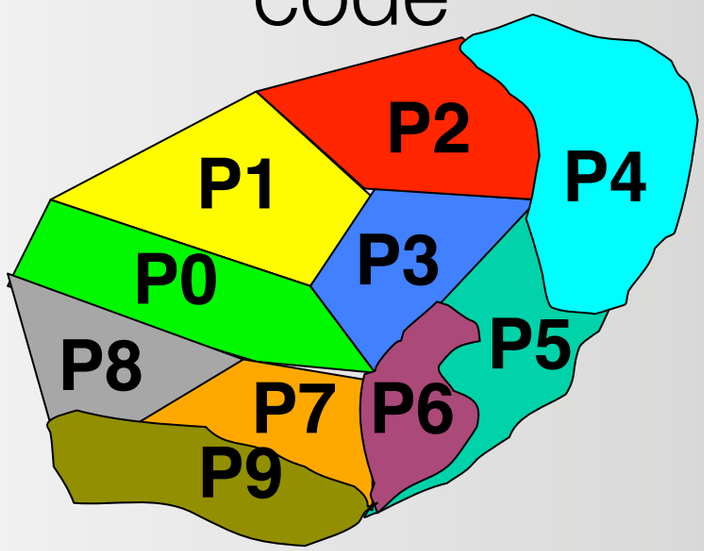


Data “chunks” on disk

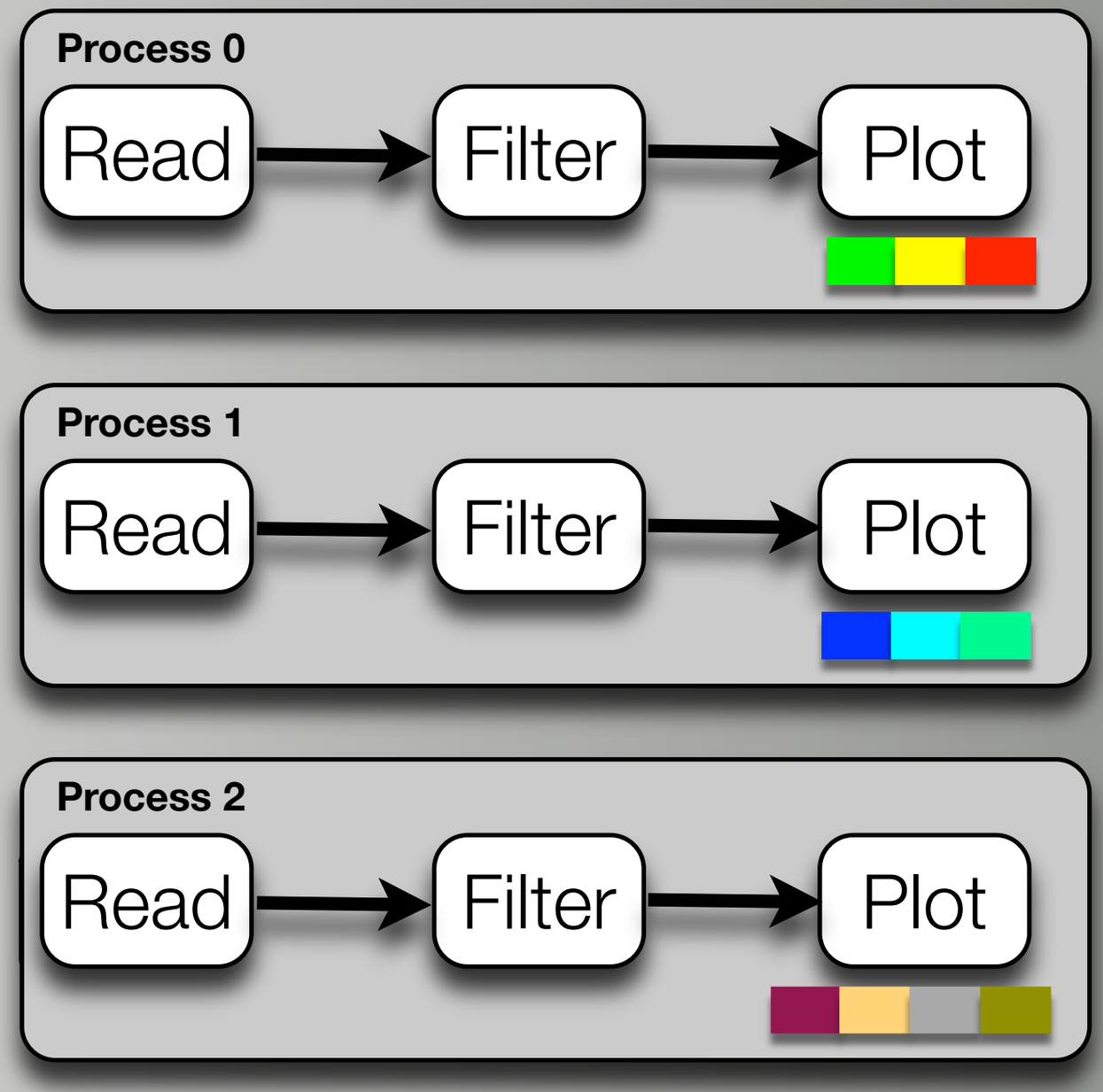


Parallel data flow networks

Parallel simulation code

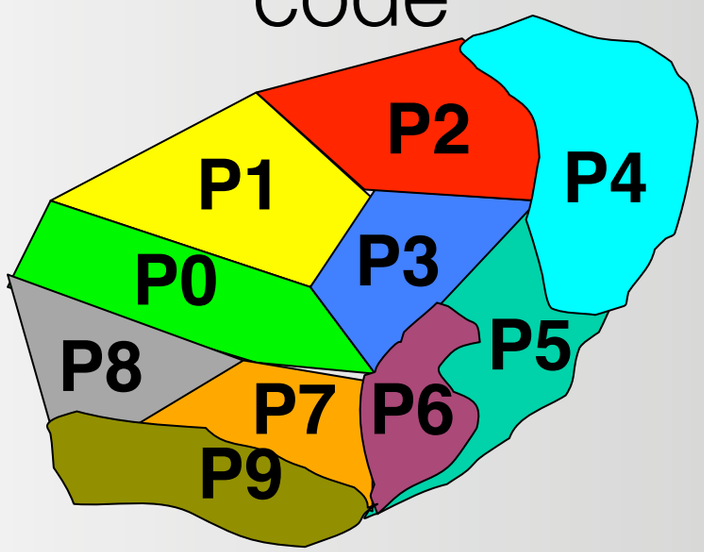


Data “chunks” on disk

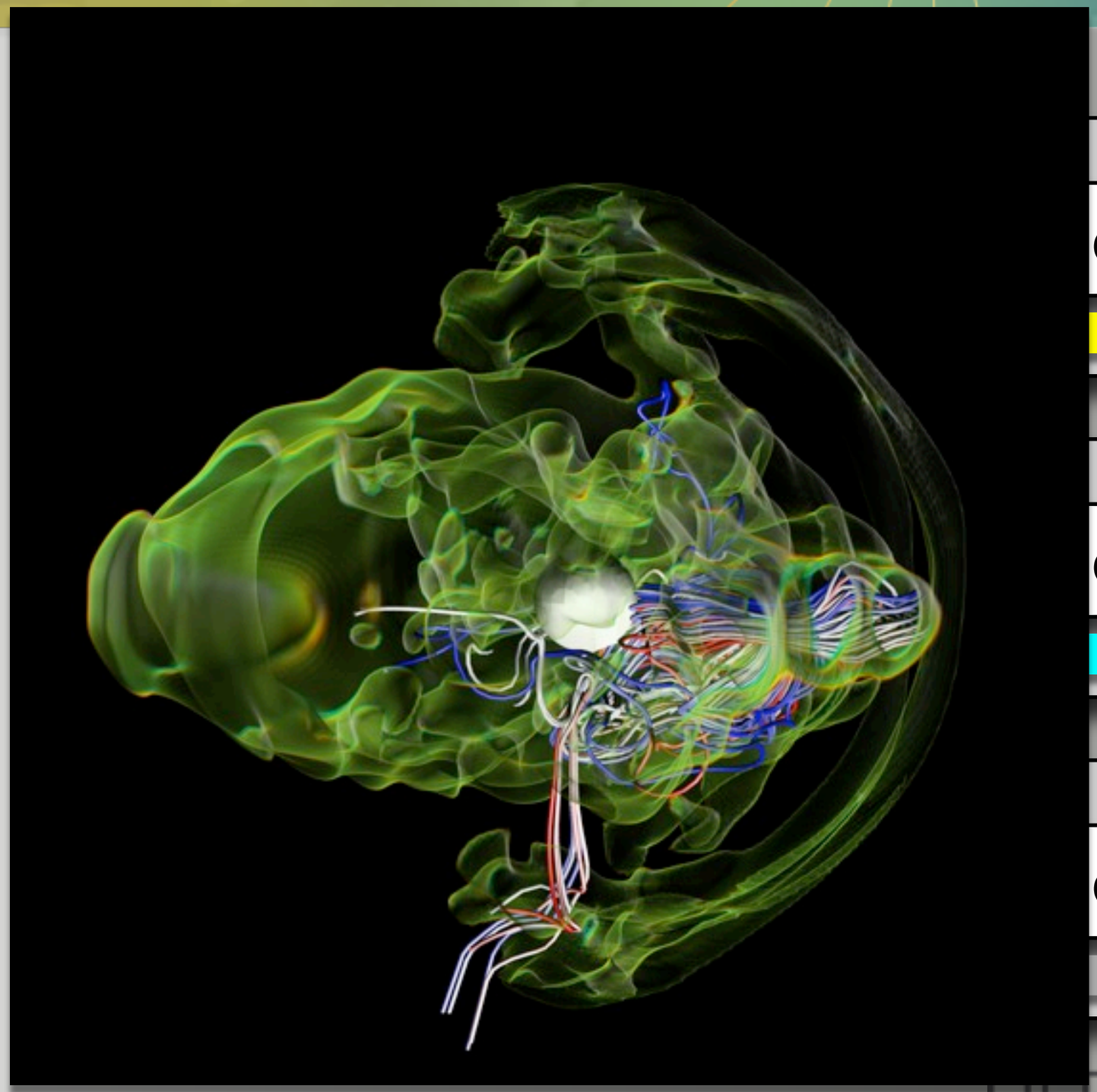


Parallel data flow networks

Parallel simulation code



Data “chunks” on disk



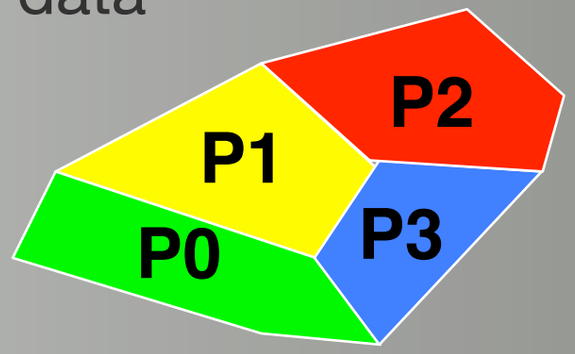
ot

ot

ot

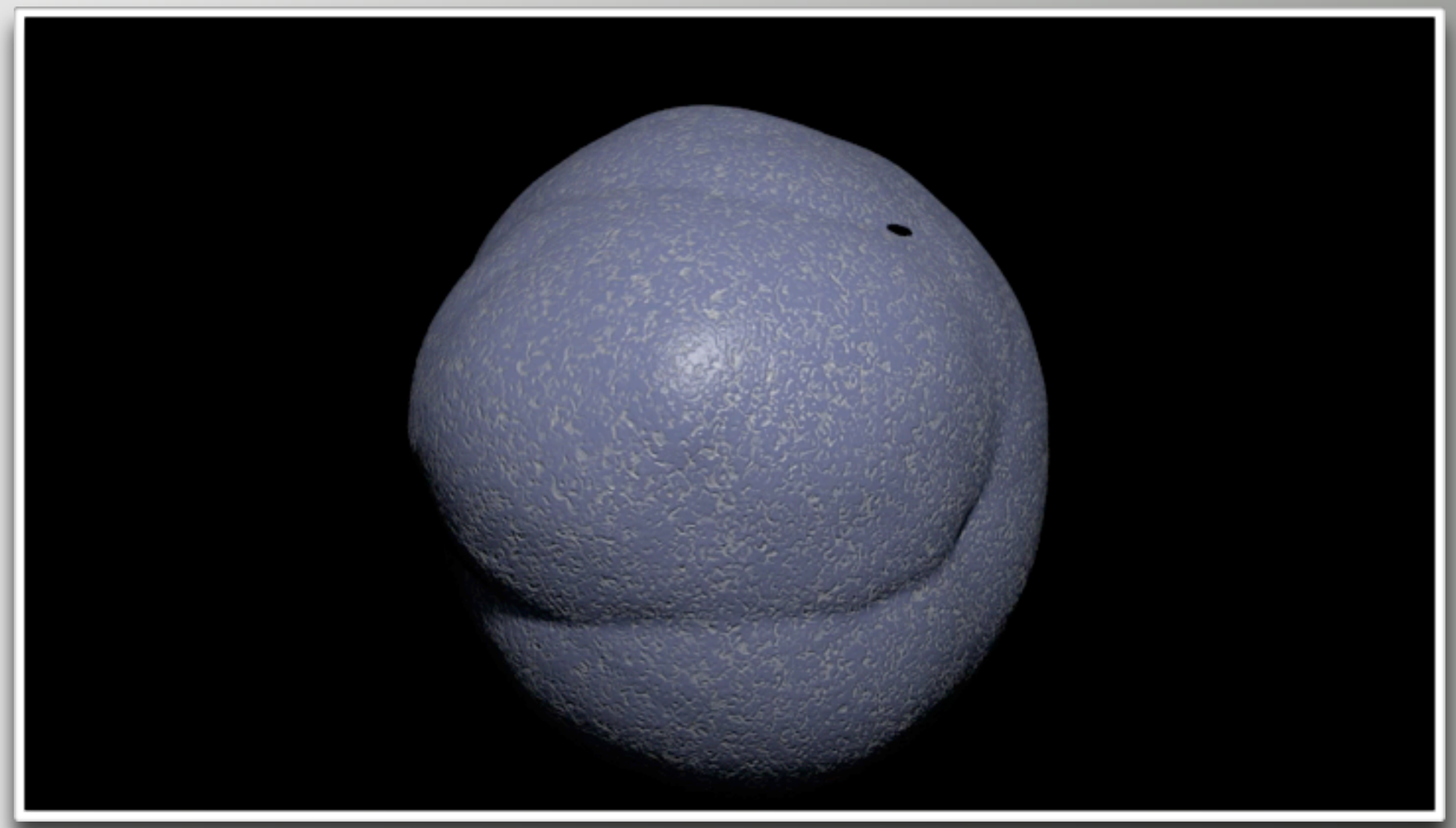
Decomposition of data is key issue

- Parallelization strategy: each processor works on piece of data
  - Key: Must have good decomposition of data
- Two I/O situations:
  - File format enforces decomposition
  - File format allows for arbitrary decomposition
- Three processing situations:
  - Pre-calculated decomposition
    - Round robin assignment of chunks to processors
  - Dynamic decomposition at I/O
    - Must understand how to read out arbitrary chunks of data
  - On-the-fly redistribution
    - Processing-dependent

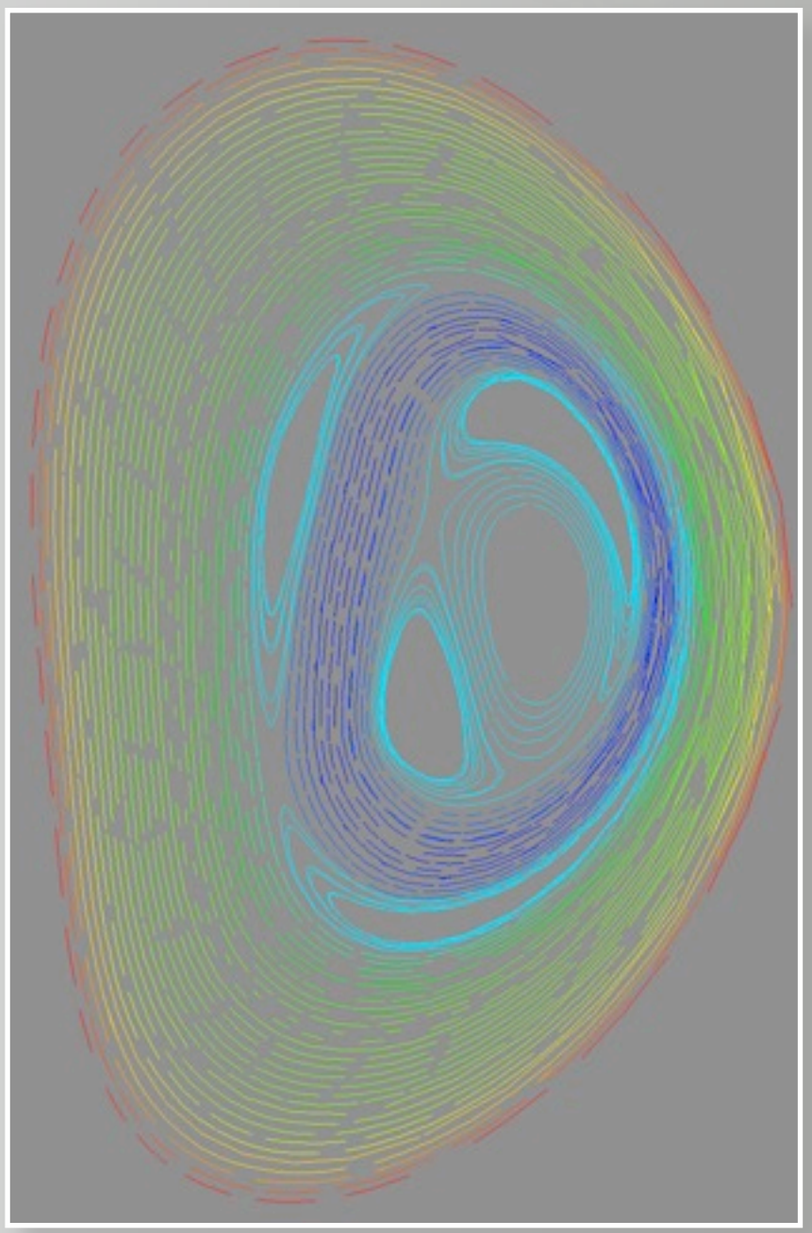


Understanding flow

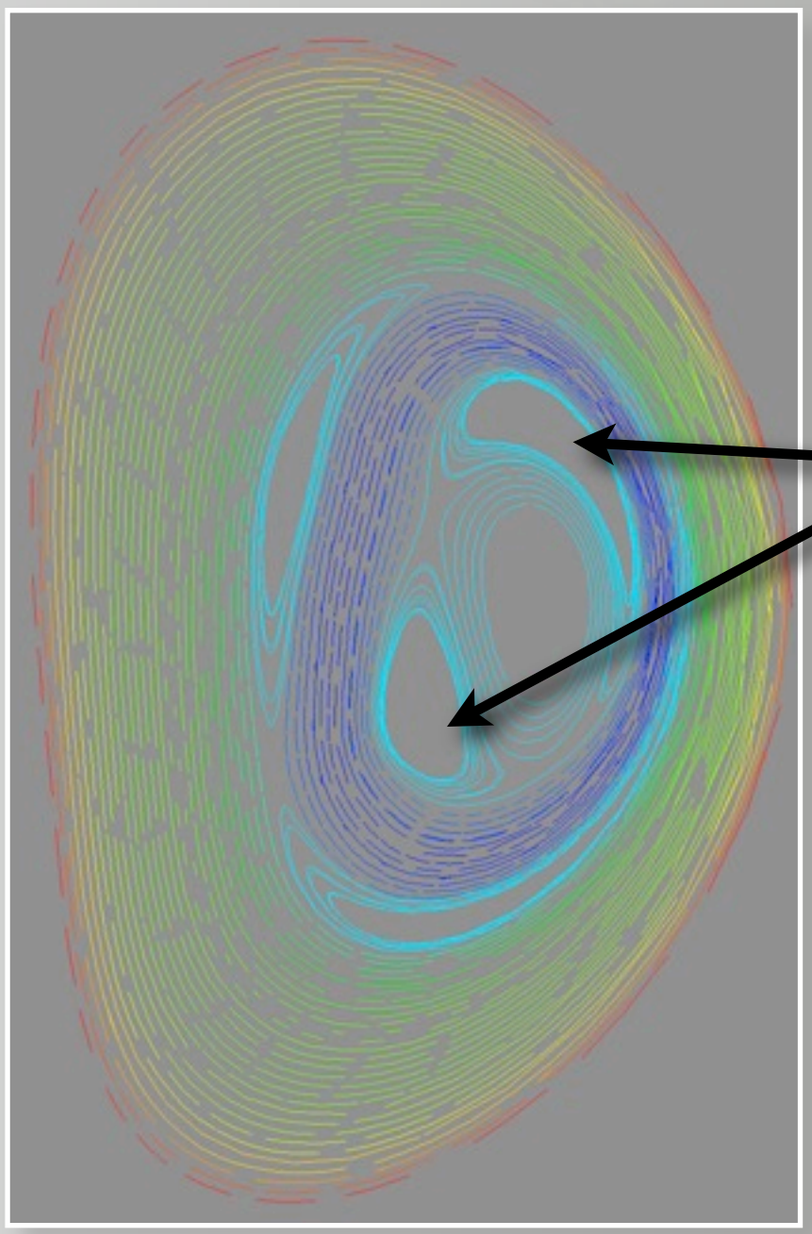
Understanding flow



Understanding flow

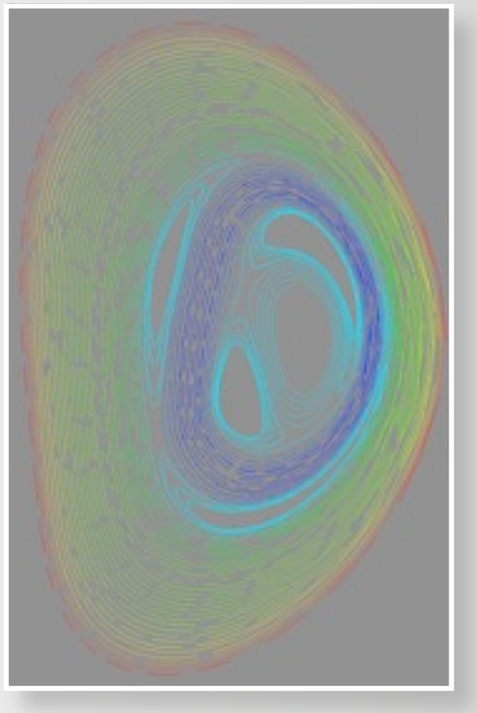


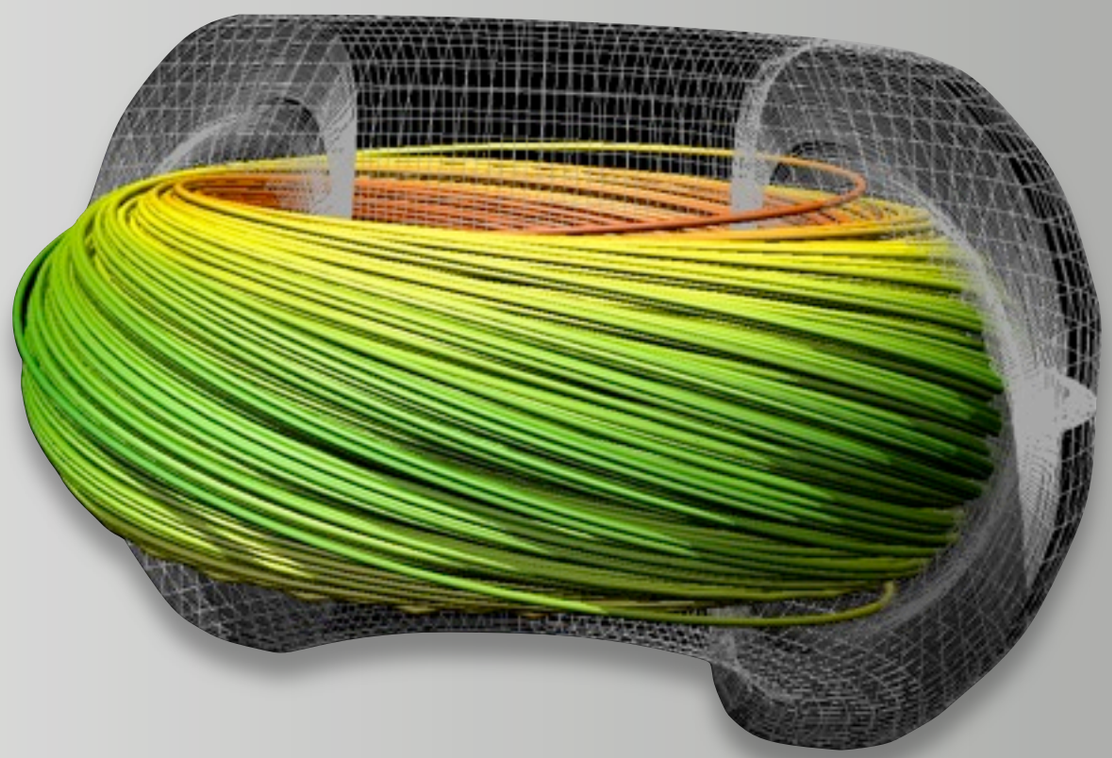
Understanding flow



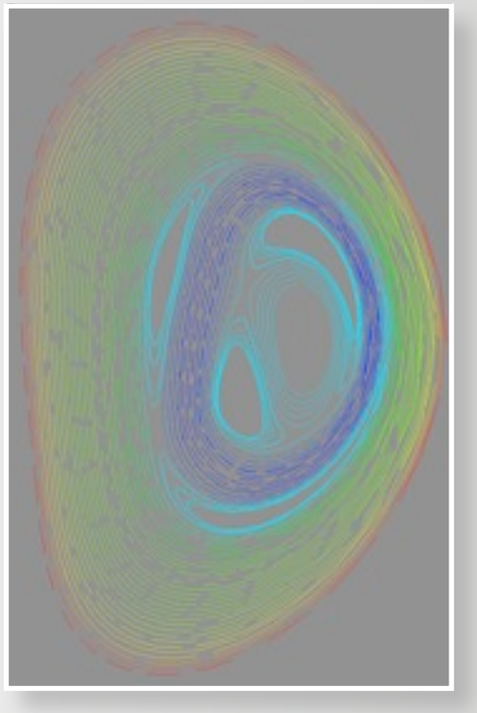
Magnetic "Islands"

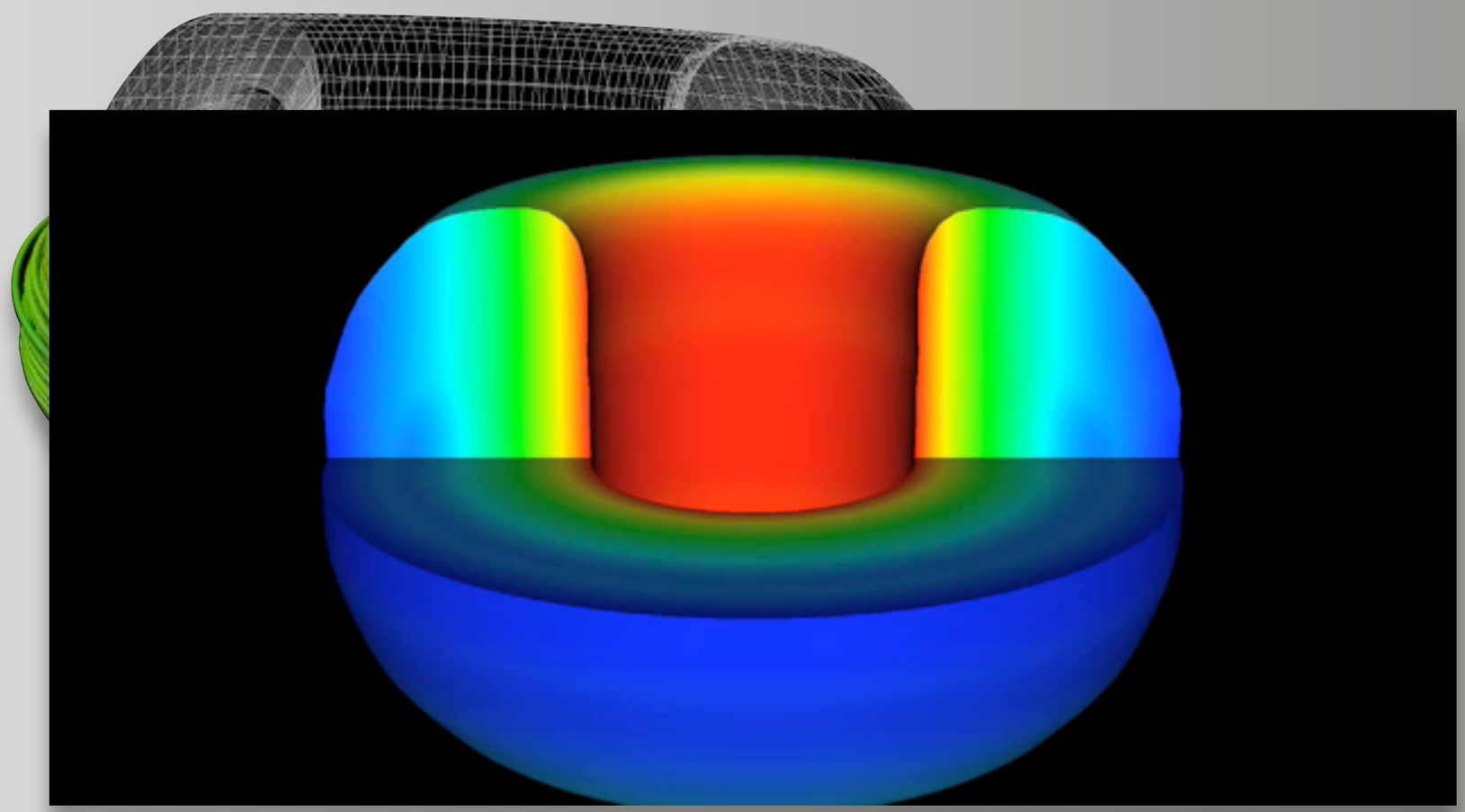
Understanding flow





Understanding flow



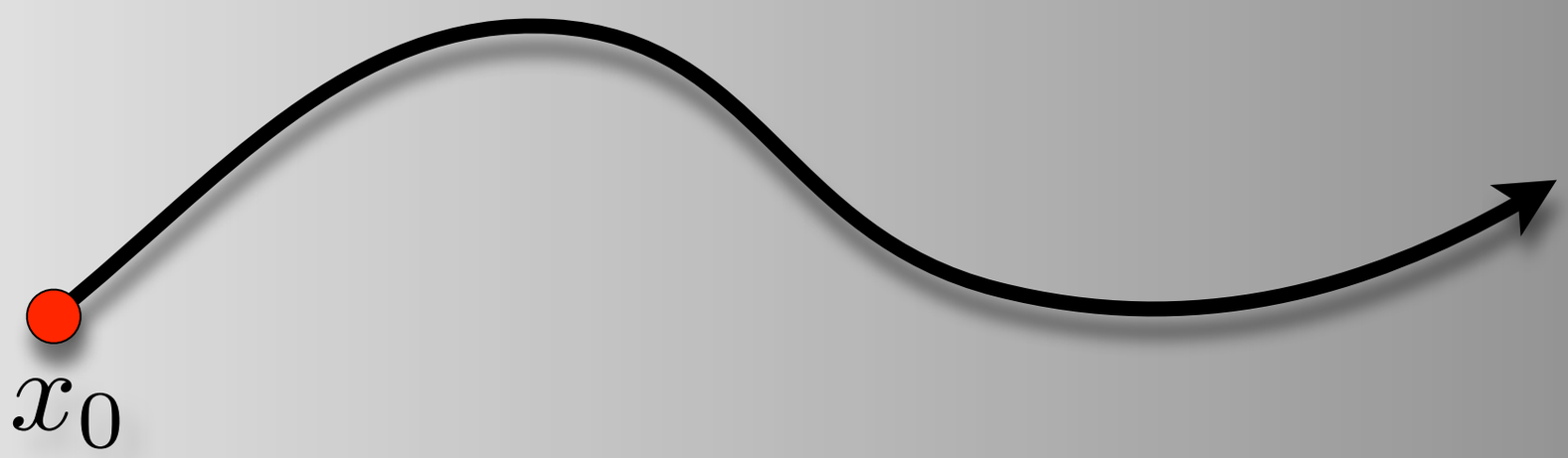


What is a streamline?

- A streamline $S(t)$ is the integral curve $\vec{v}(x)$ traced by a massless particle x_0 through a vector field

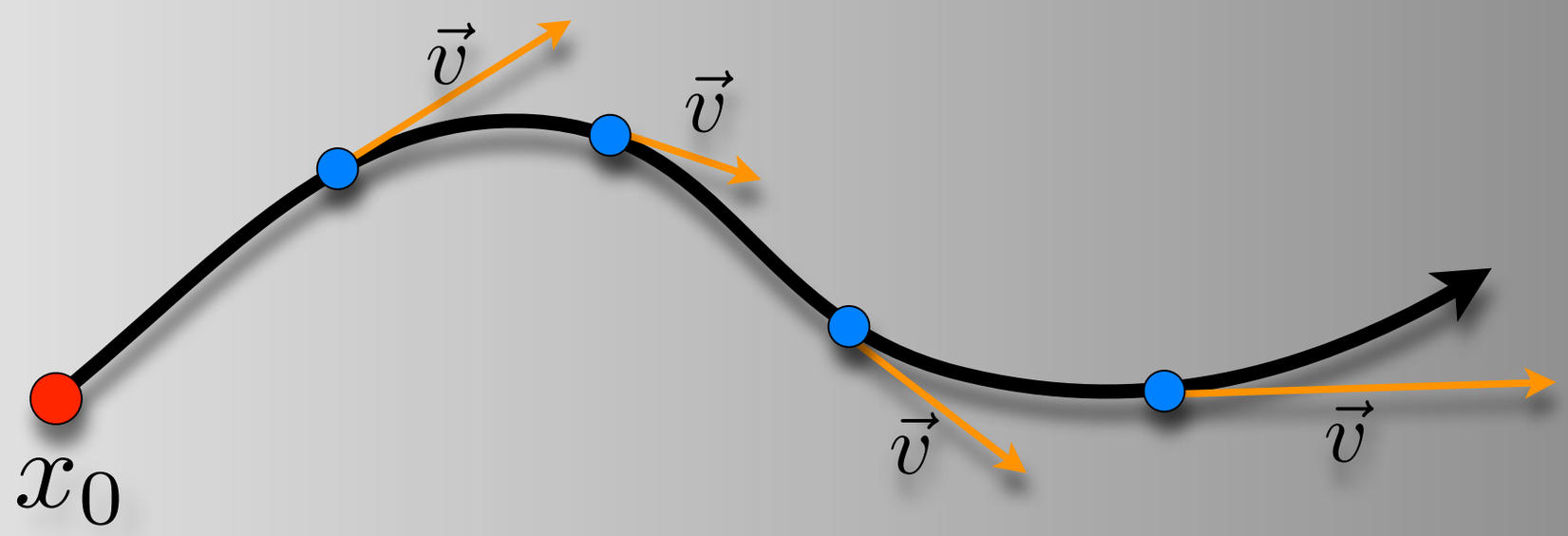
What is a streamline?

- A streamline $S(t)$ is the integral curve $\vec{v}(x)$ traced by a massless particle x_0 through a vector field



What is a streamline?

- A streamline $S(t)$ is the integral curve $\vec{v}(x)$ traced by a massless particle x_0 through a vector field



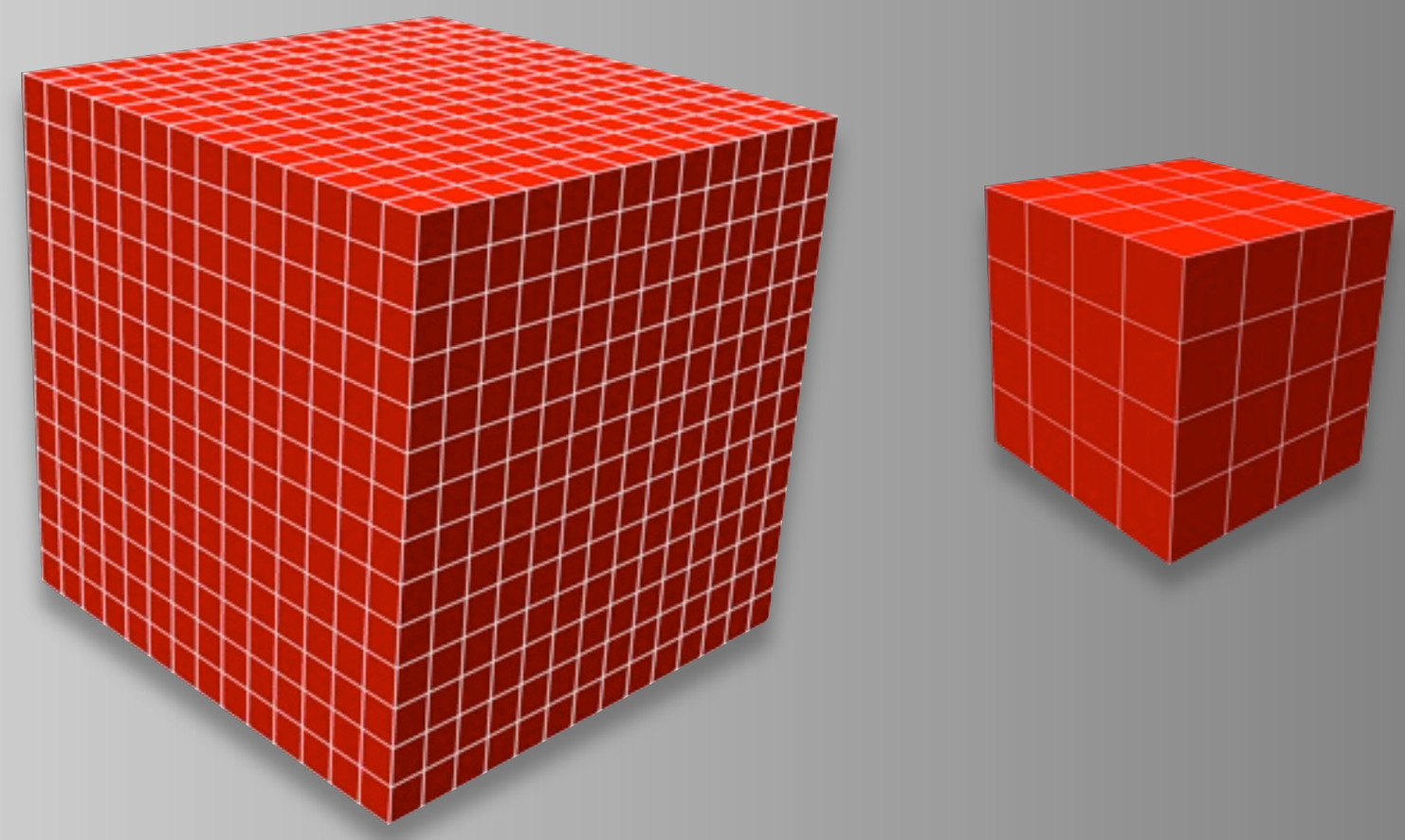
$$\dot{S}(t) = \vec{v}(S(t)) \qquad S(0) = x_0$$

Barriers to Scalable Streamline Computation

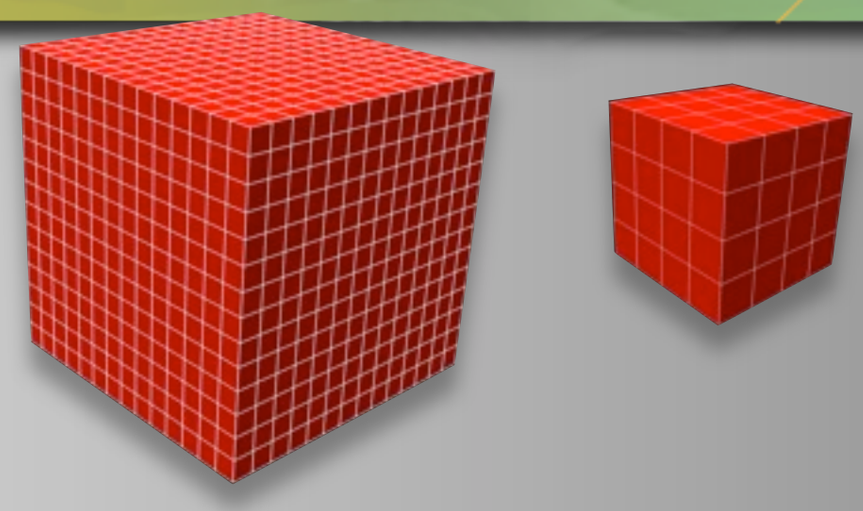
1. Dataset size

I/O

More cells =
- more integration steps
- smaller step sizes



Barriers to Scalable Streamline Computation



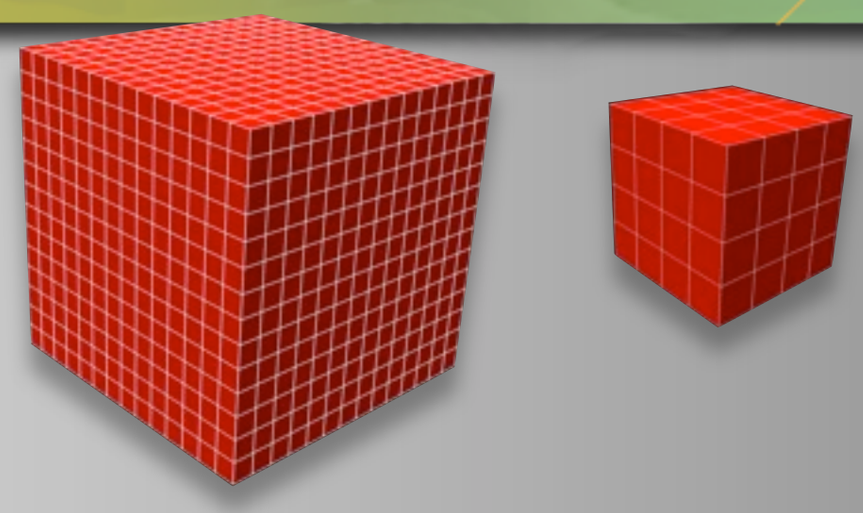
1. Dataset size

I/O

More cells =
more integration steps
smaller step sizes

Number of
blocks

Barriers to Scalable Streamline Computation

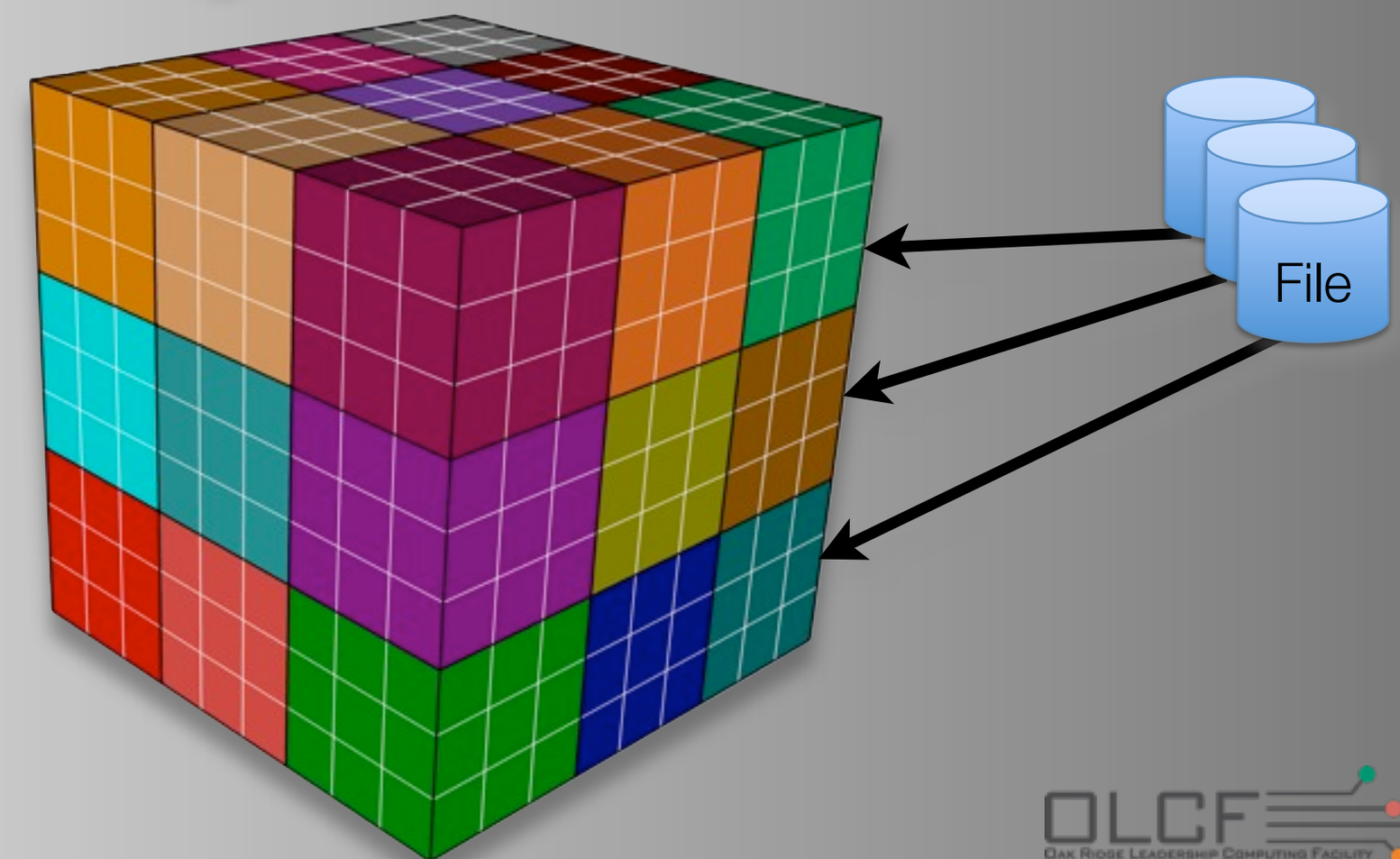


1. Dataset size

I/O

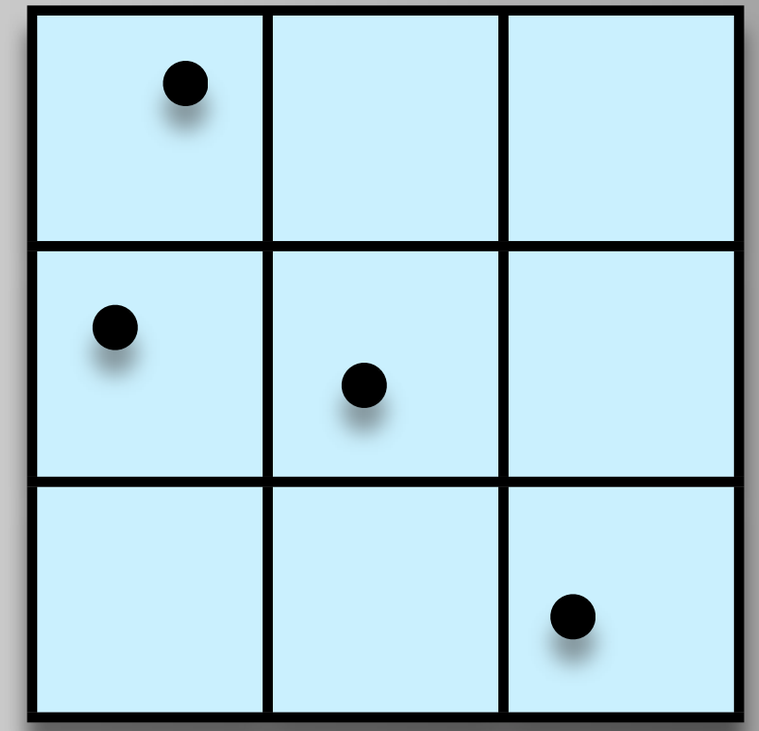
More cells =
more integration steps
smaller step sizes

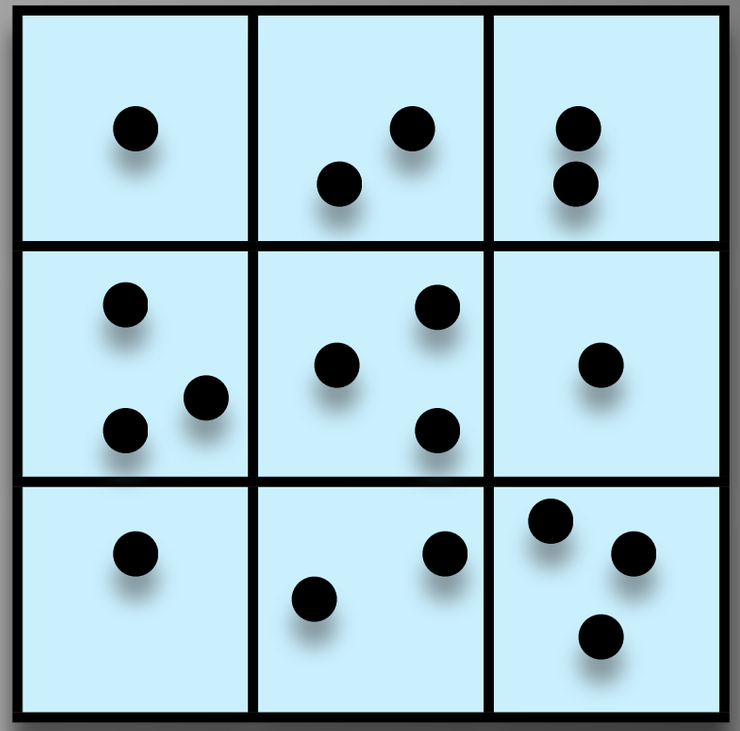
Number of
blocks



Barriers to Scalable Streamline Computation

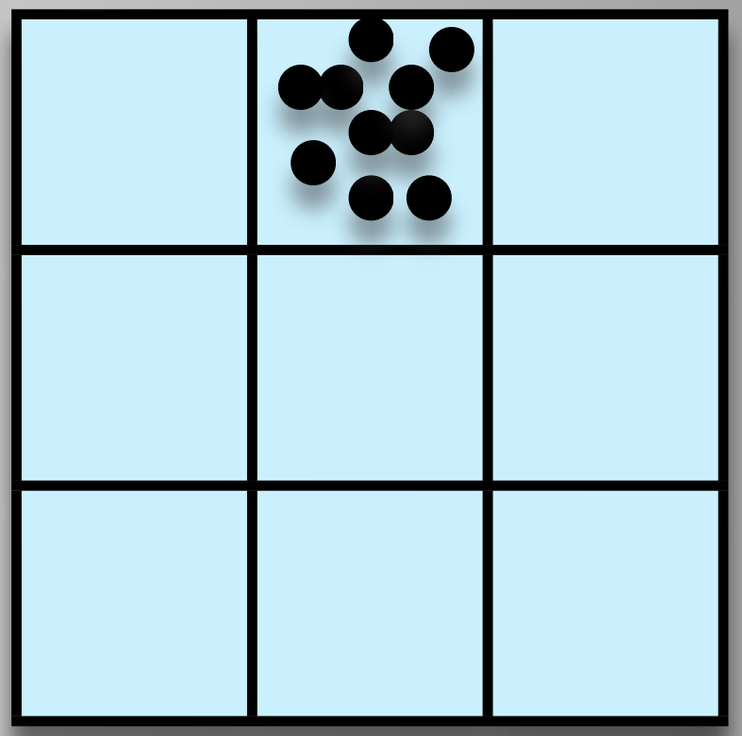
1. Dataset size
2. **Seed count**



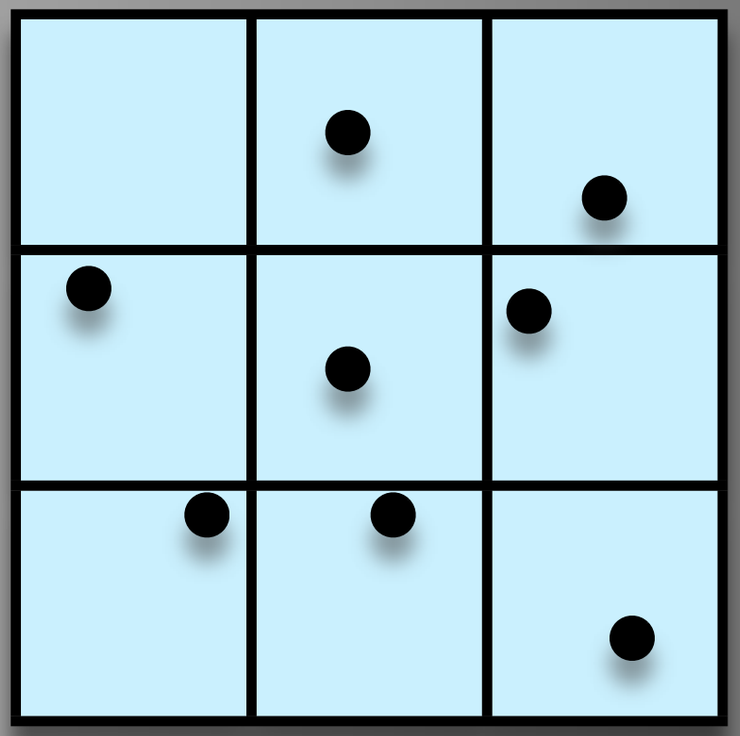


Barriers to Scalable Streamline Computation

1. Dataset size
2. Seed count
3. **Seed distribution**



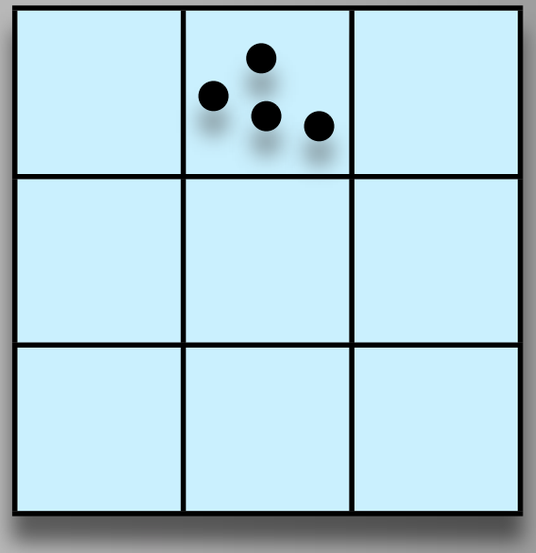
Dense



Sparse

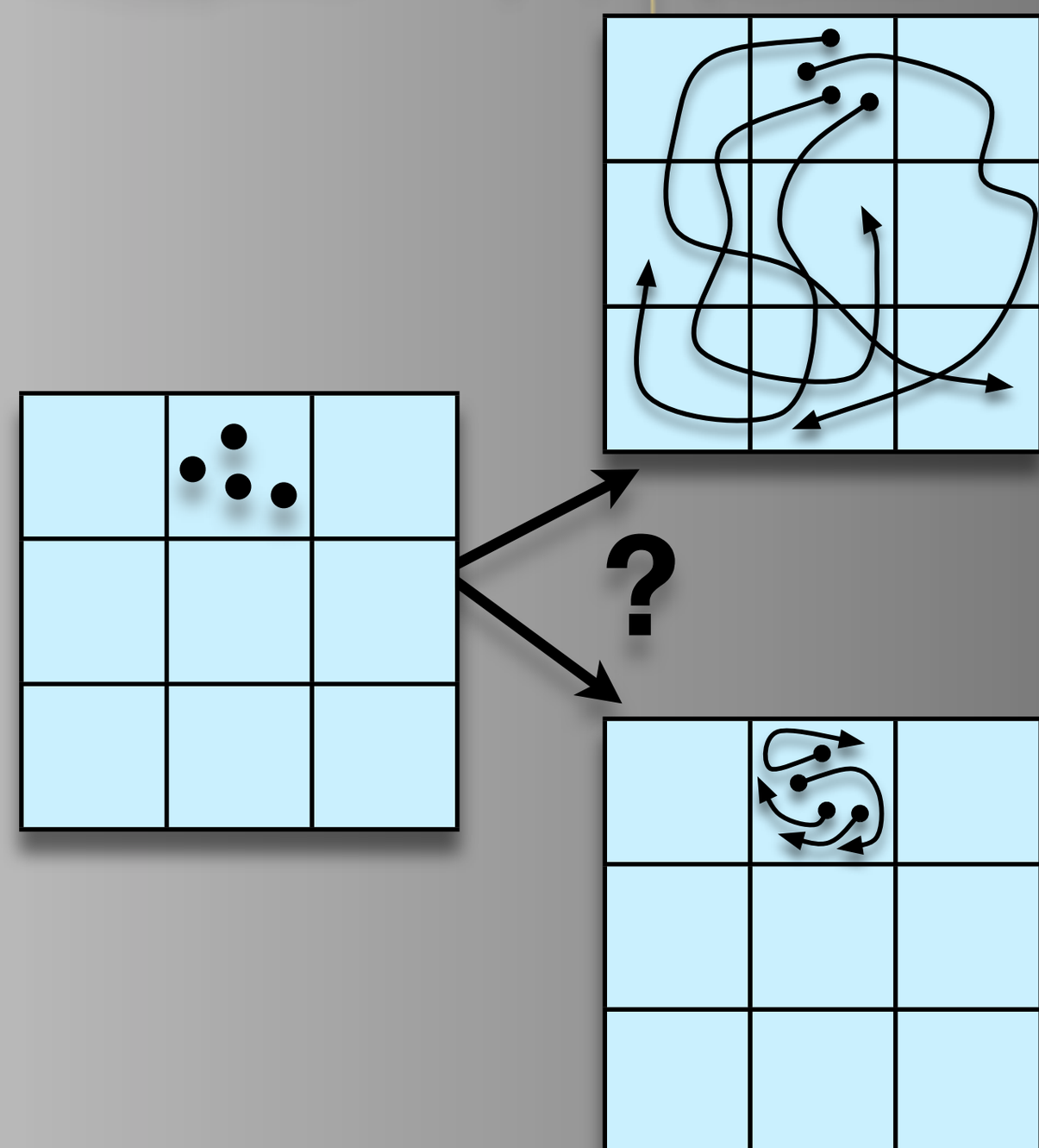
Barriers to Scalable Streamline Computation

1. Dataset size
2. Seed count
3. Seed distribution
4. **Vector field complexity**



Barriers to Scalable Streamline Computation

1. Dataset size
2. Seed count
3. Seed distribution
4. **Vector field complexity**



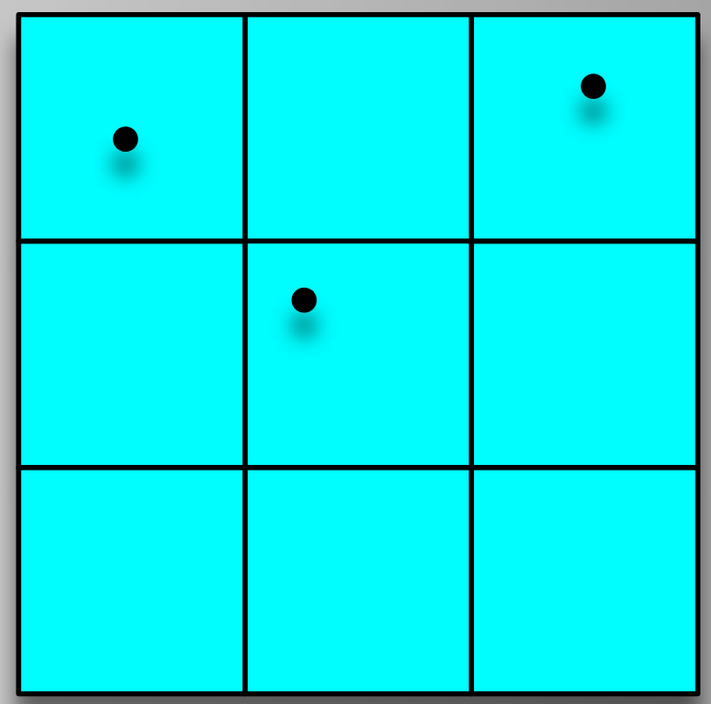
Traditional Parallelization Techniques

Traditional Parallelization Techniques

Block decomposed mesh

Seed points

Processors



P0

P1

P2

Traditional Parallelization Techniques

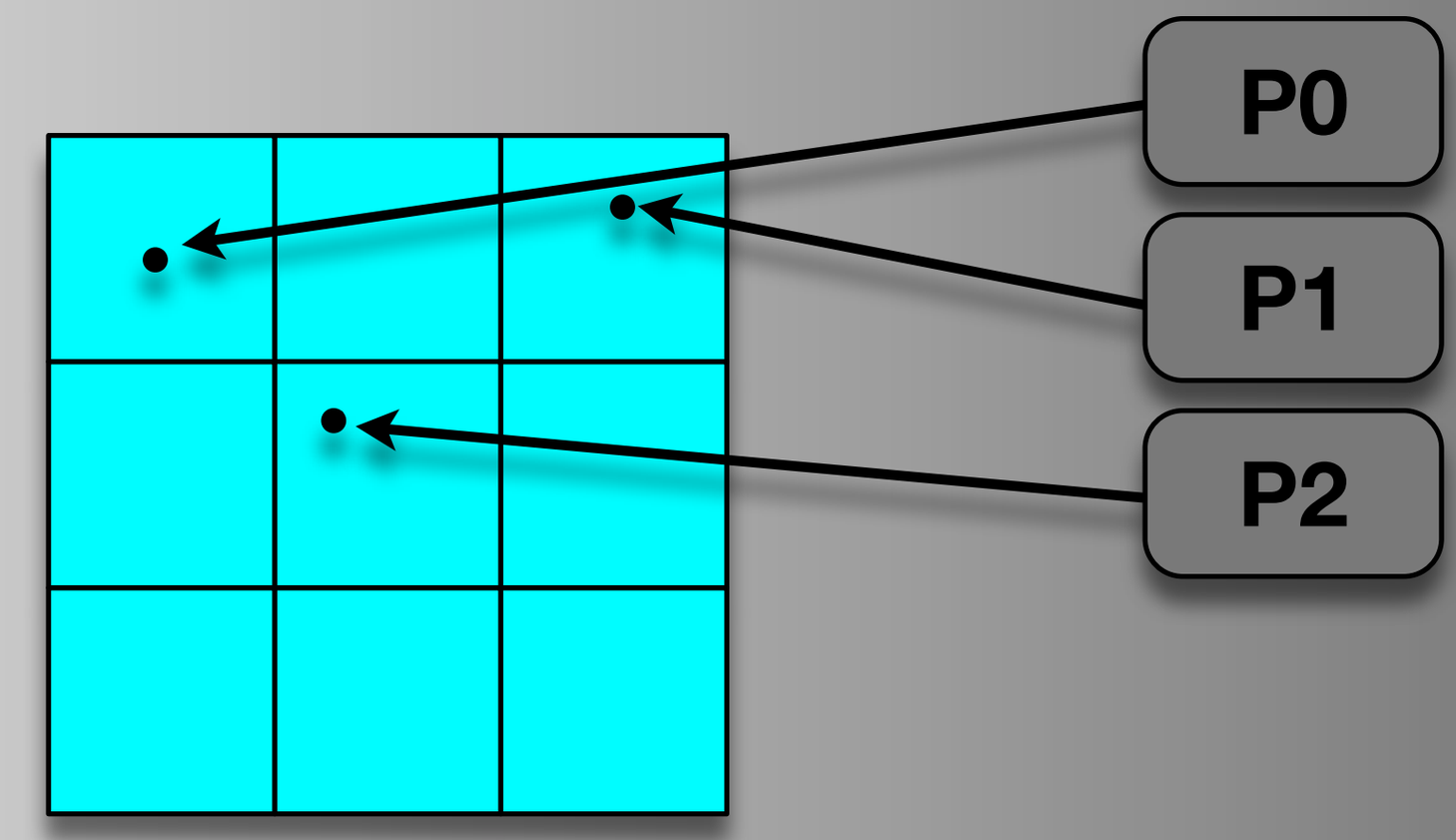
Block decomposed mesh

Seed points

Processors

Parallelize over:

Seeds



Traditional Parallelization Techniques

Block decomposed mesh

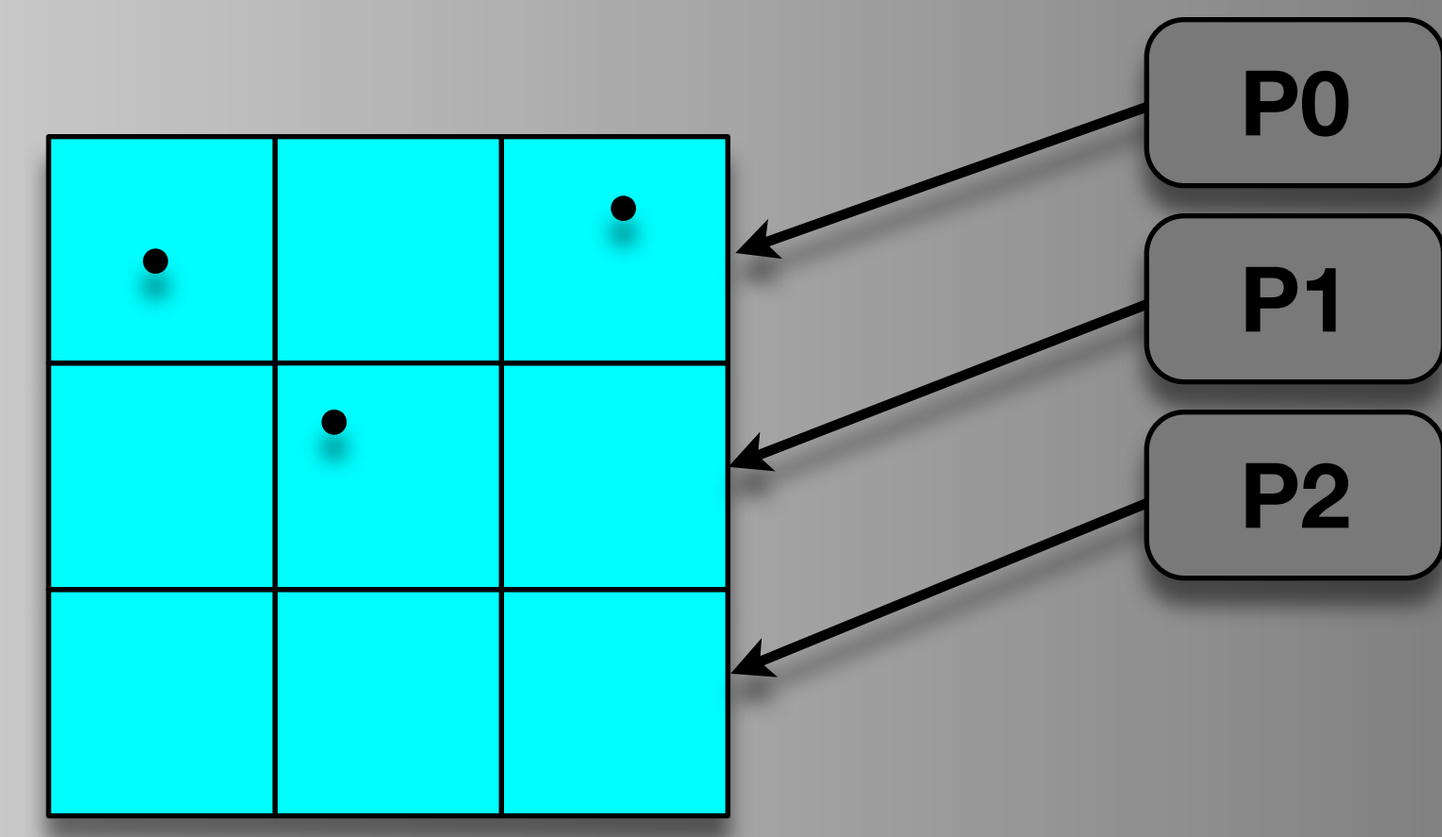
Seed points

Processors

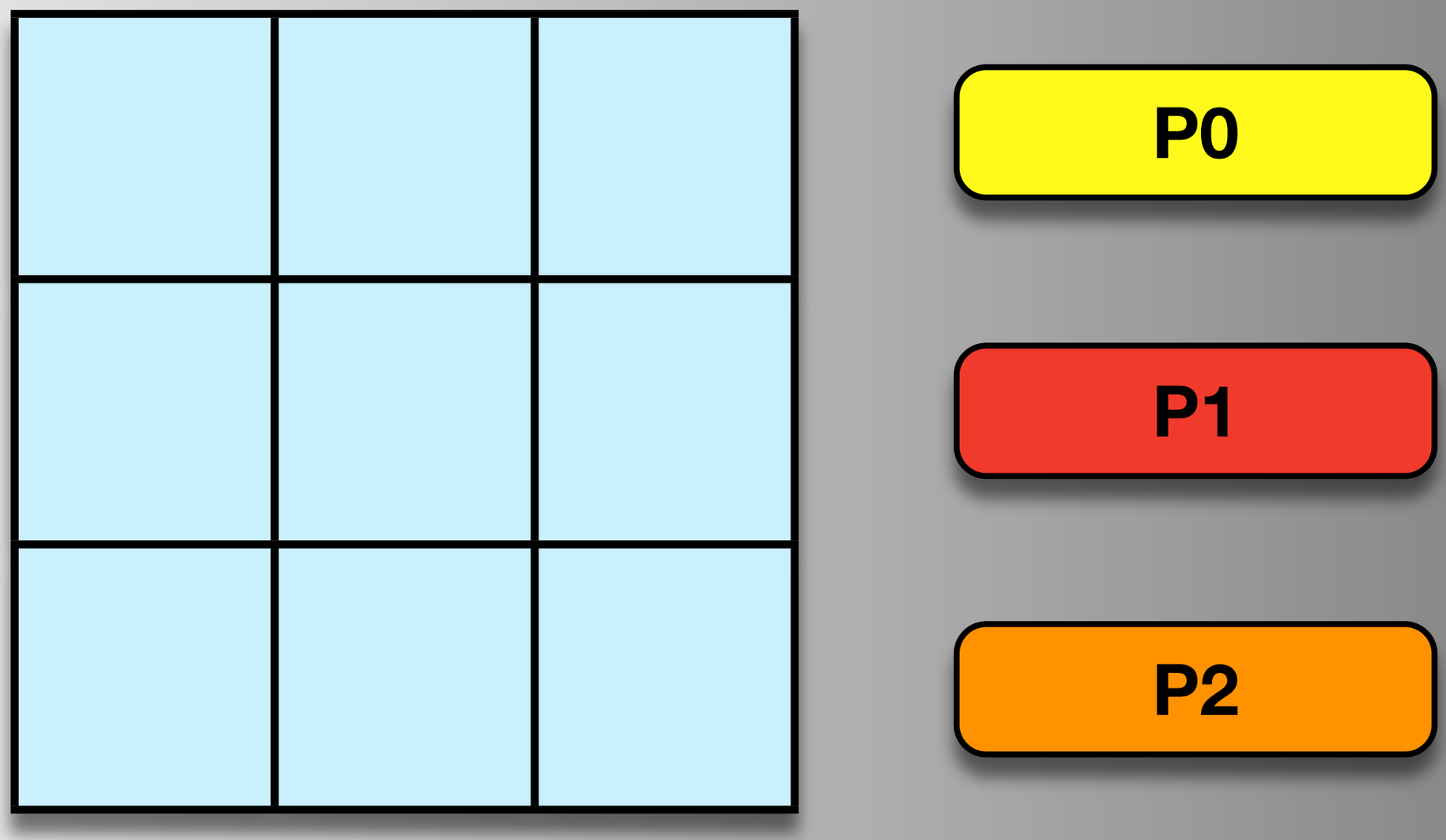
Parallelize over:

Seeds

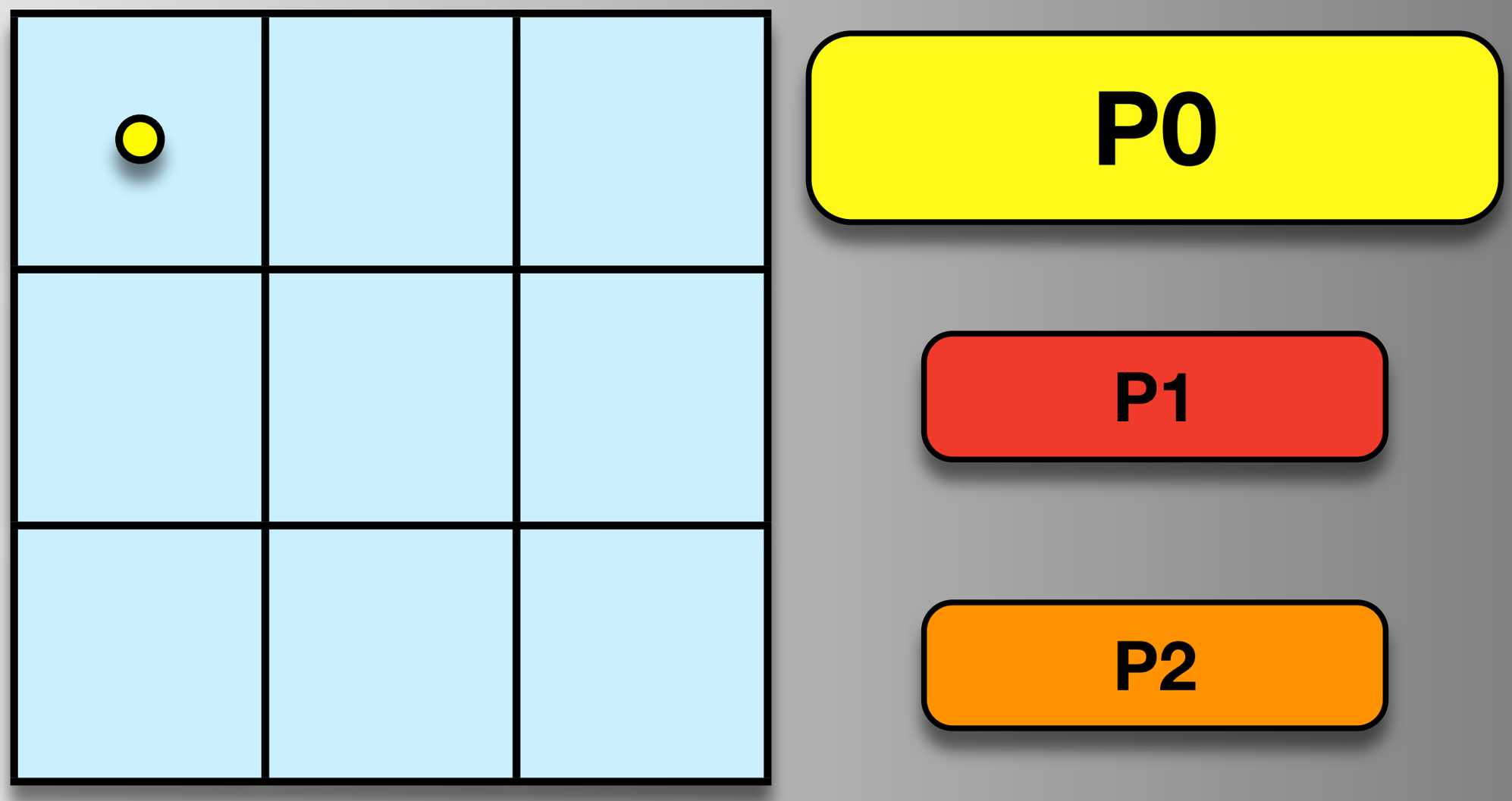
Blocks



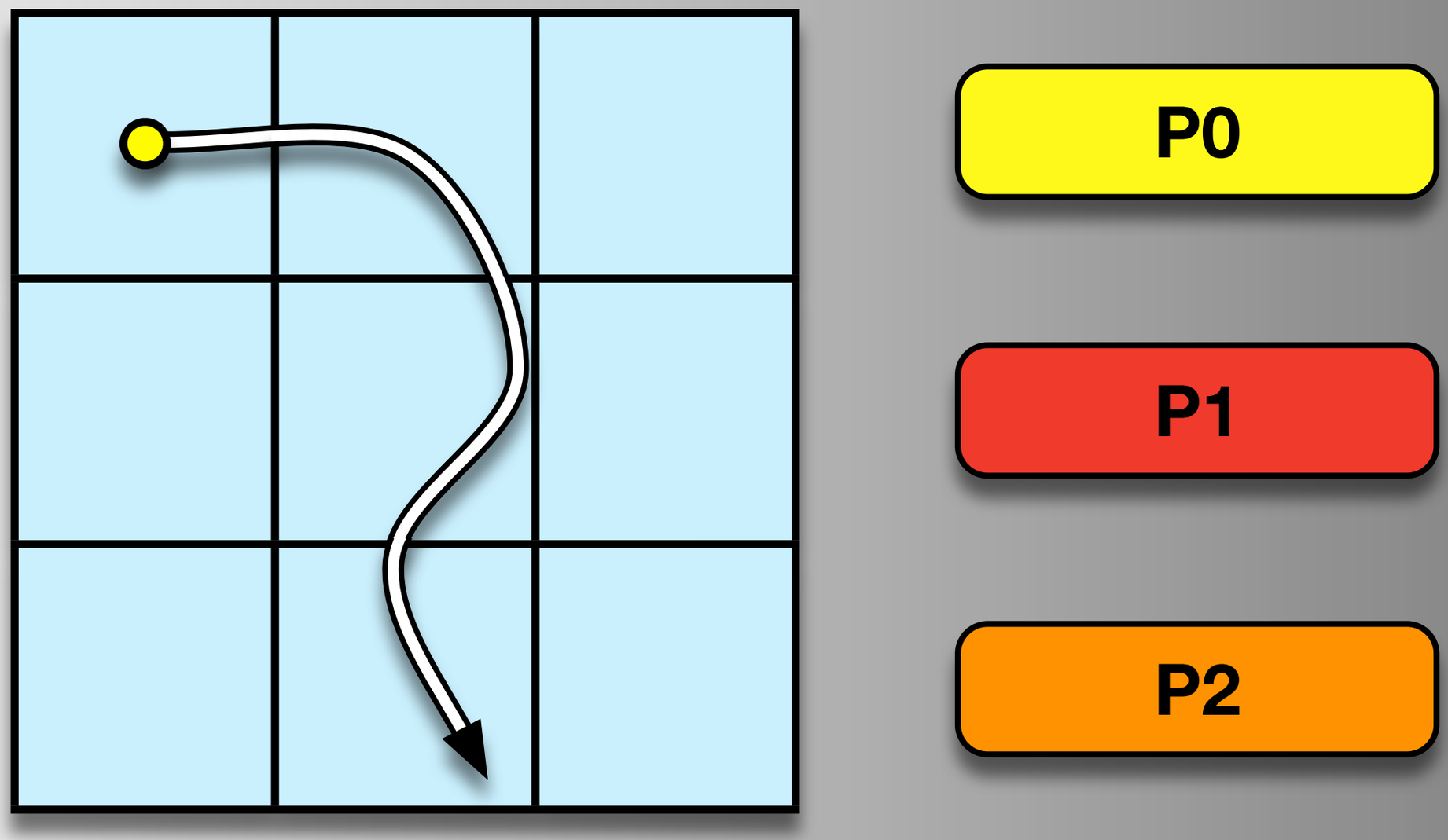
Parallelize over seed points



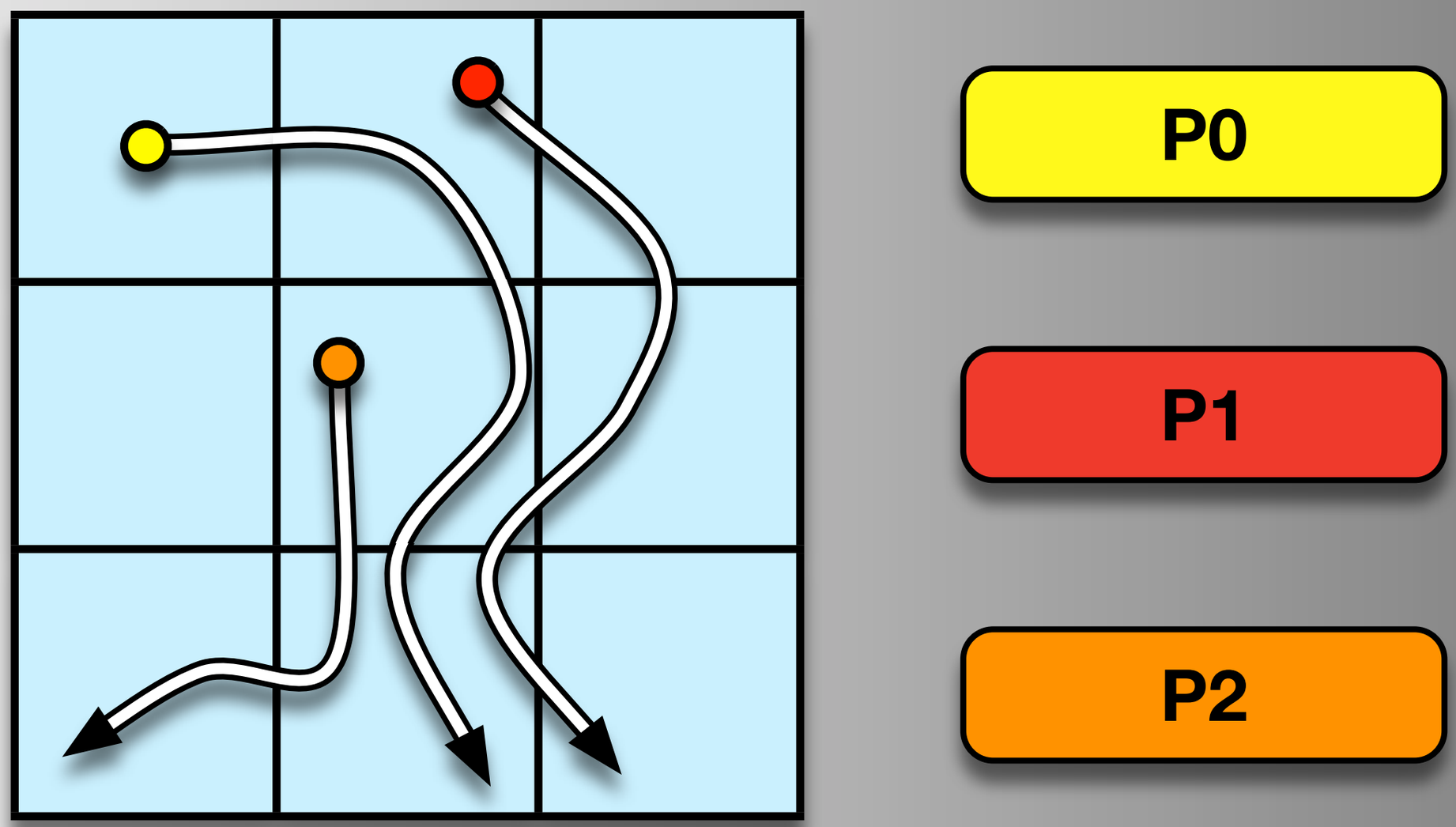
Parallelize over seed points



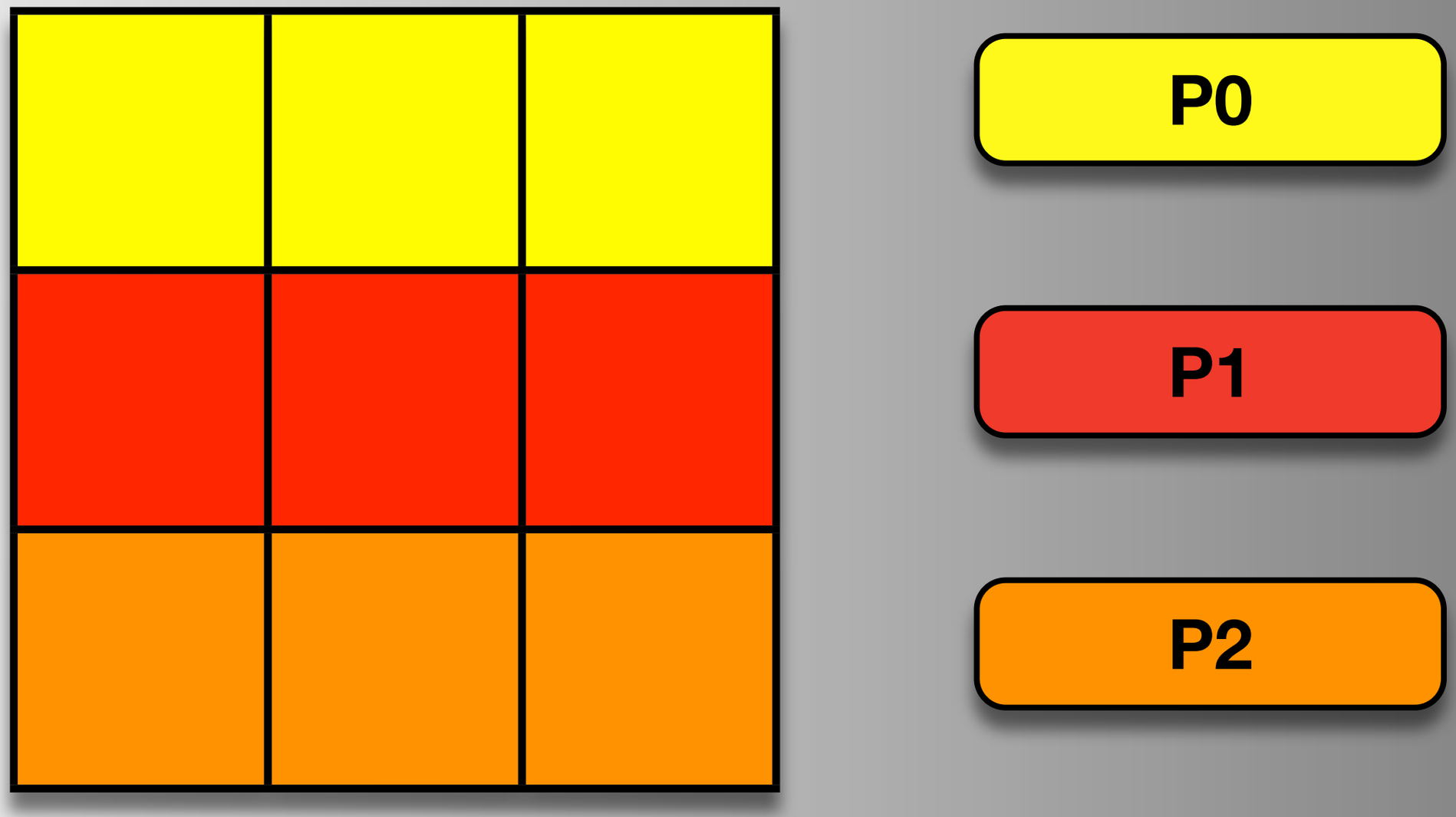
Parallelize over seed points



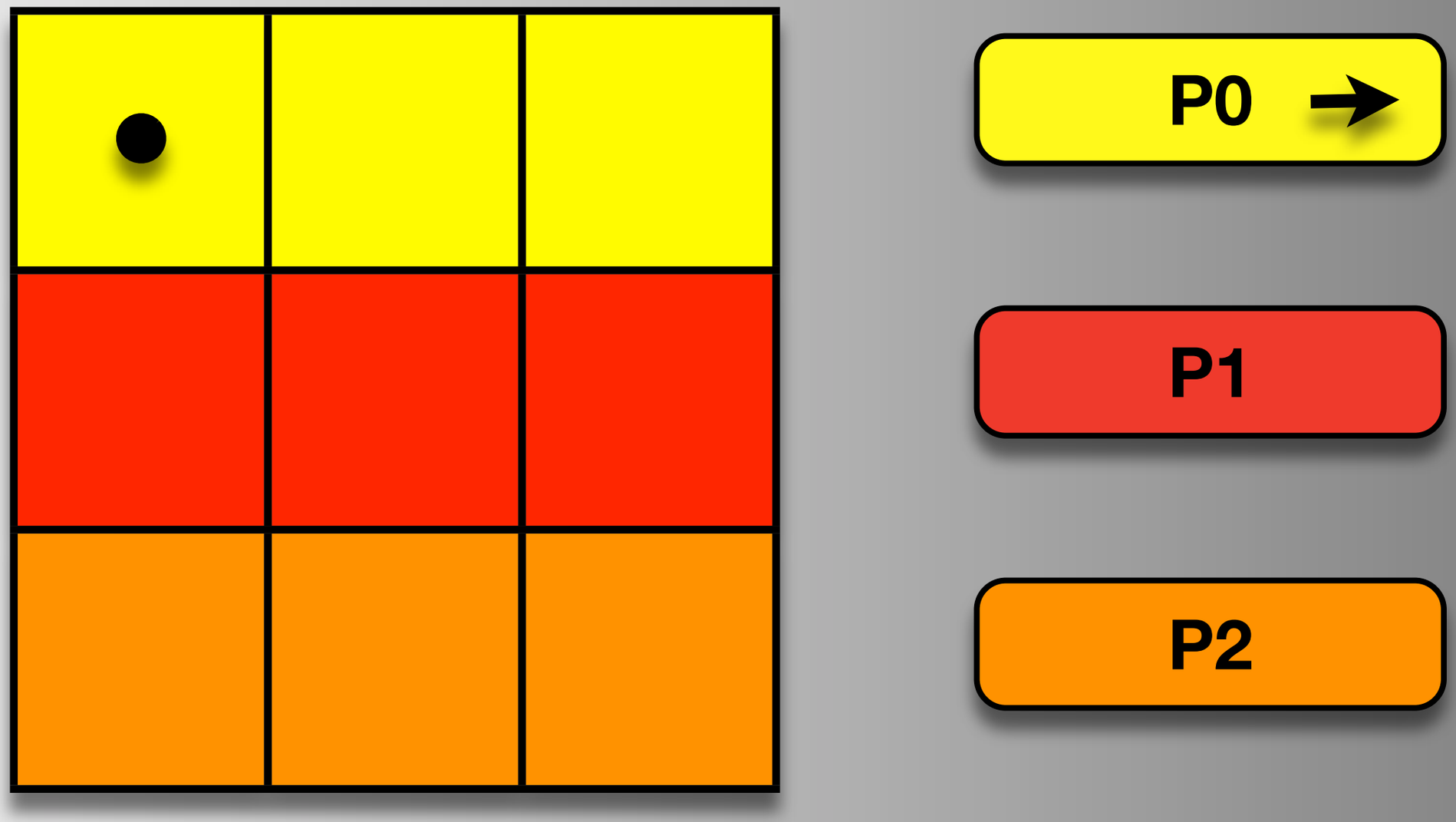
Parallelize over seed points



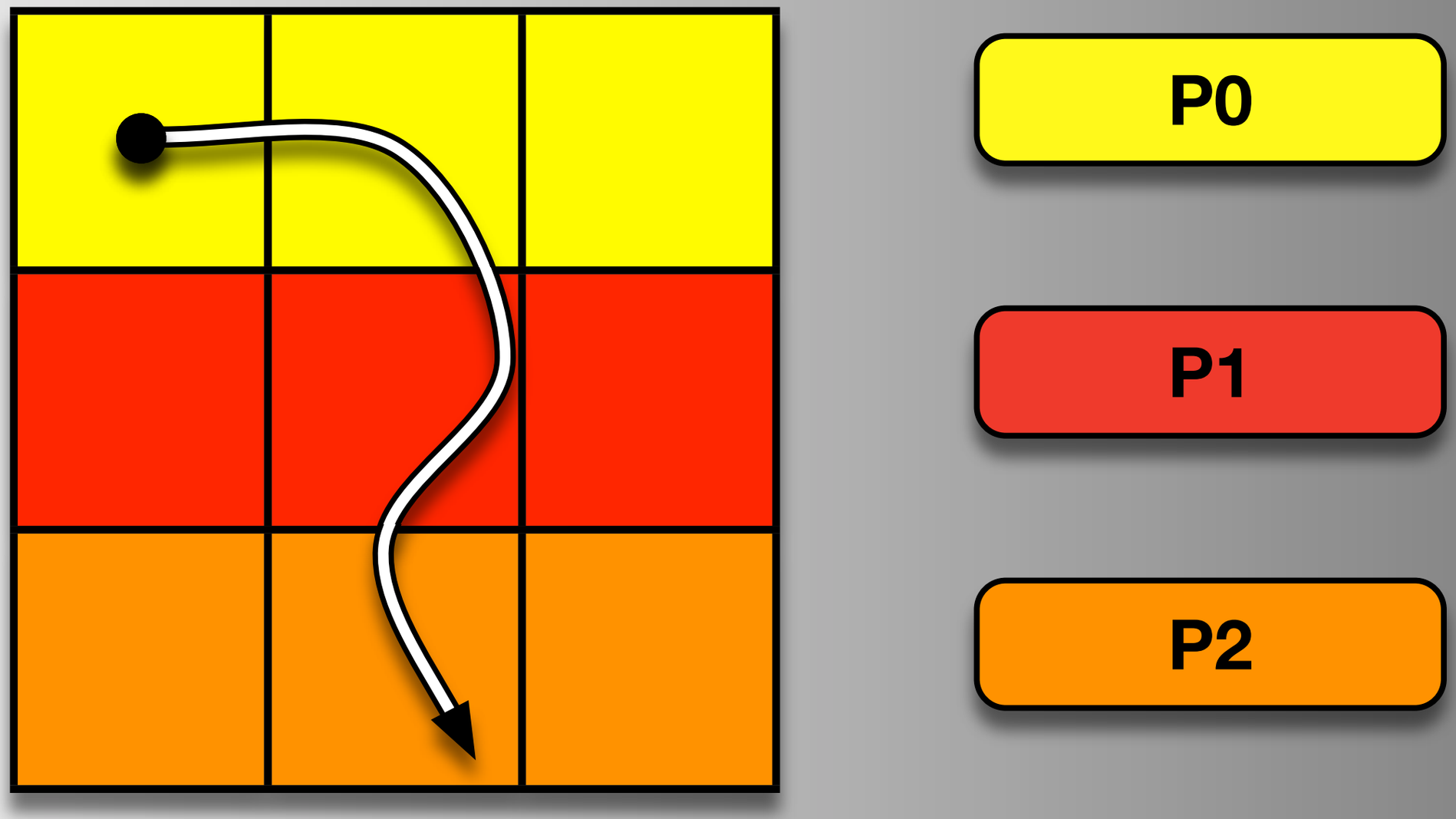
Parallelize over blocks



Parallelize over blocks



Parallelize over blocks



Parallelize over blocks

P0

P1

P2

Advantages / Disadvantages of traditional algorithms

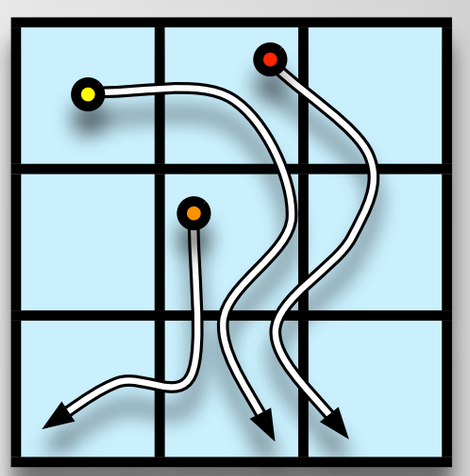
Seed point

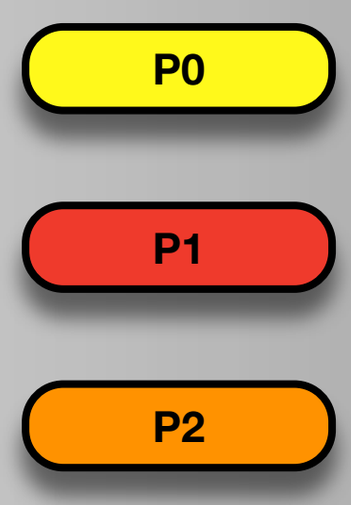
+ Advantages

- Simple
- Insensitive to seeding

- Disadvantages

- Redundant I/O
- Prone to load imbalance





Advantages / Disadvantages of traditional algorithms

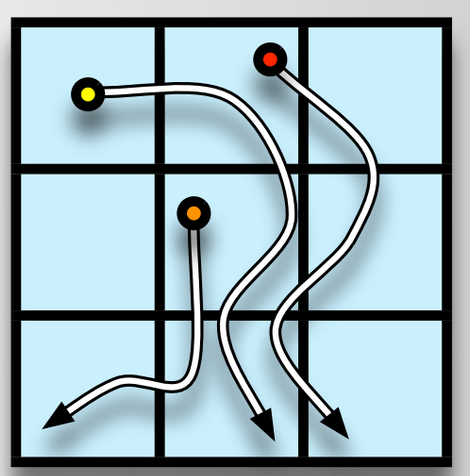
Seed point

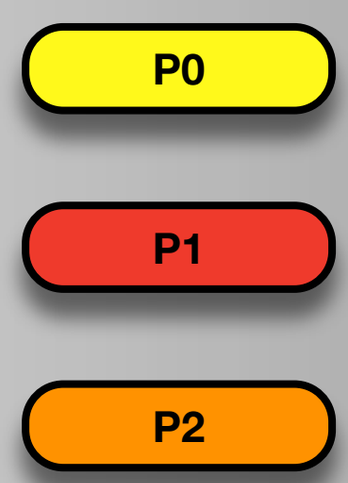
+ Advantages

- Simple
- Insensitive to seeding

- Disadvantages

- Redundant I/O
- Prone to load imbalance





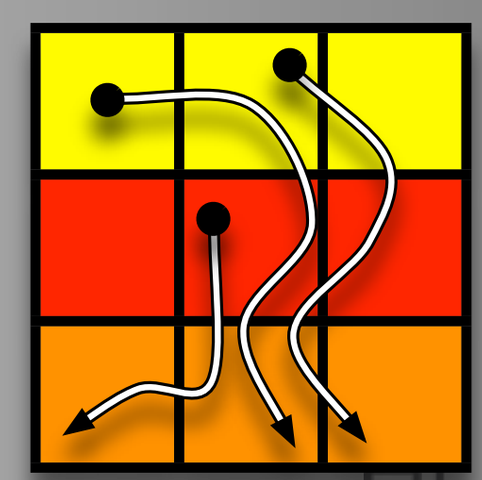
Block

+ Advantages

- Fairly simple
- Minimal I/O

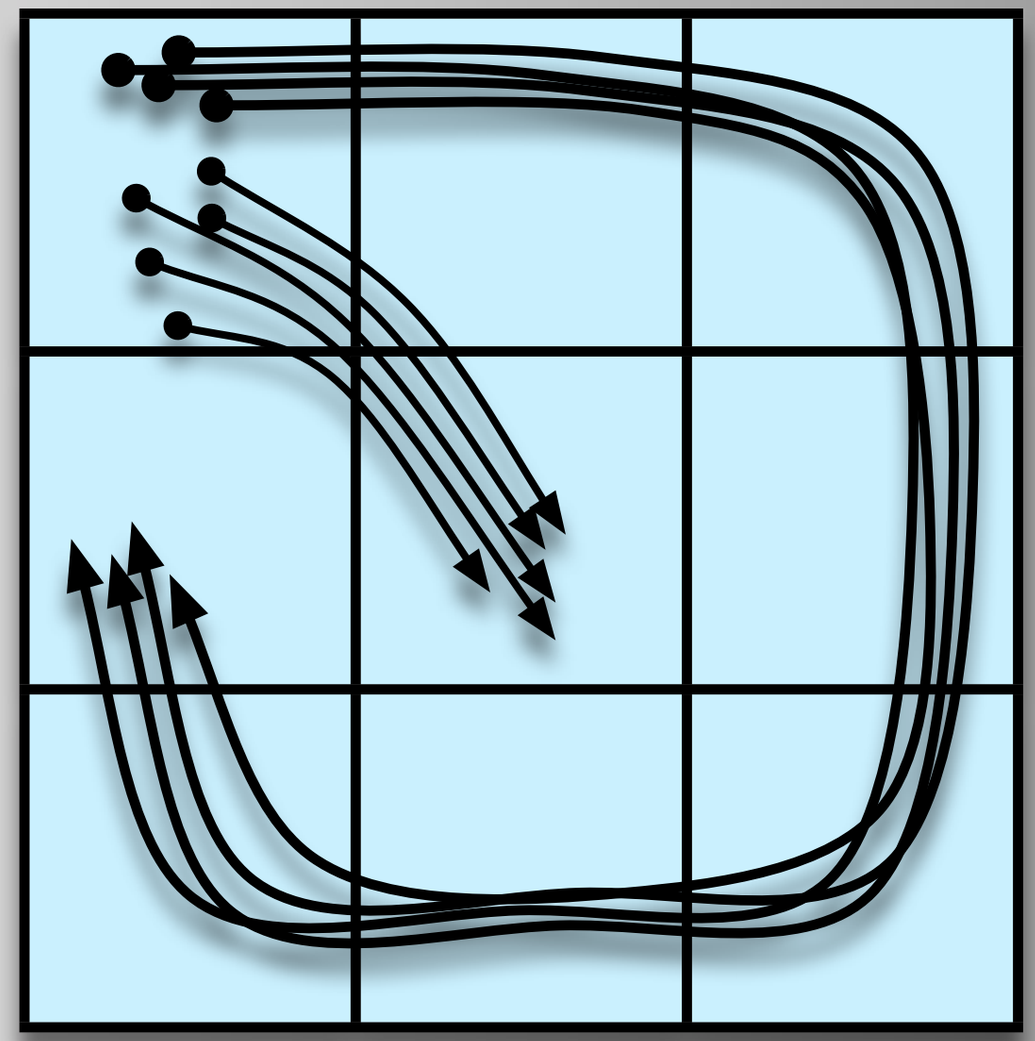
- Disadvantages

- Sensitive to seeding
- Prone to load imbalance



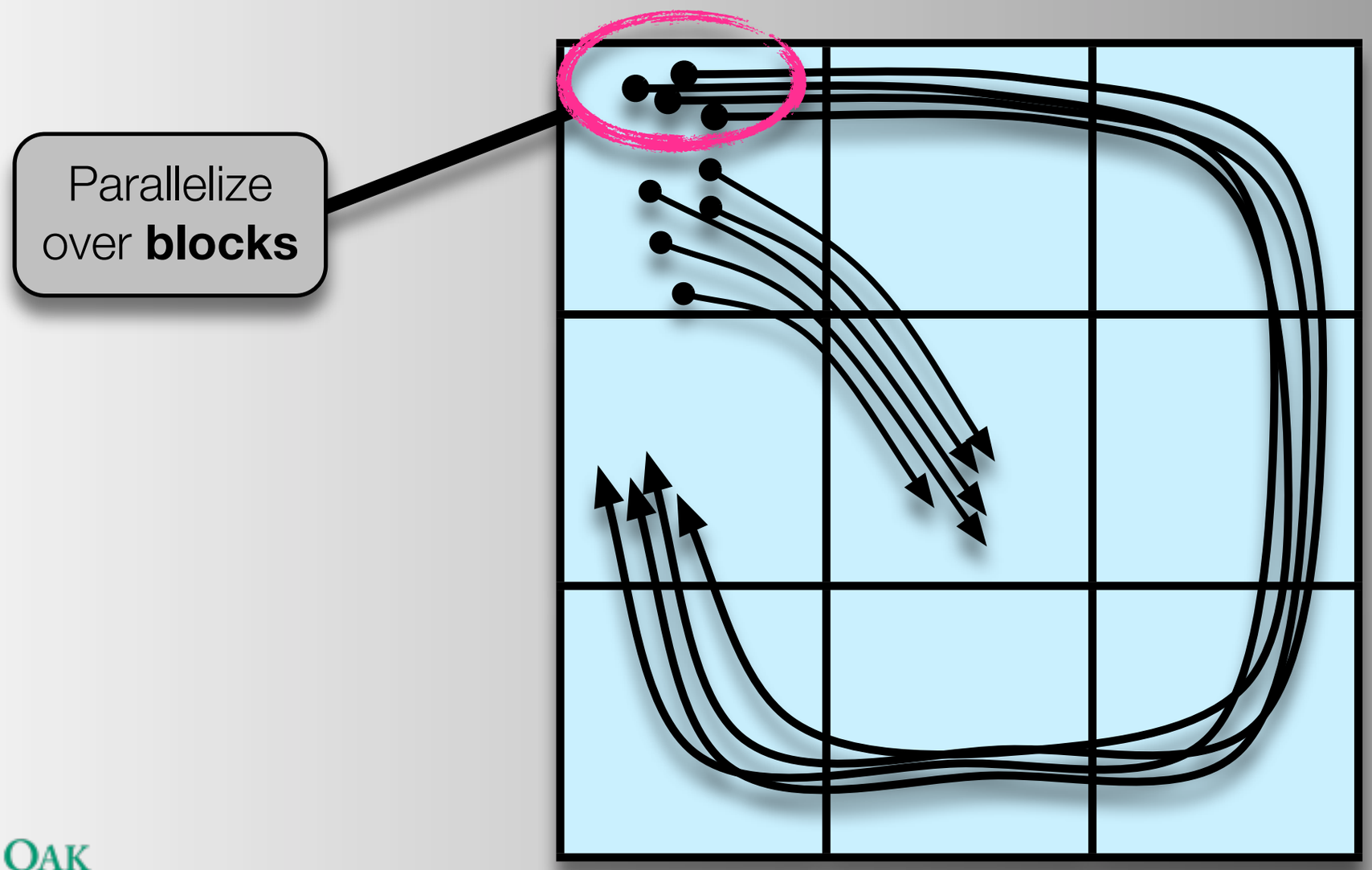
Traditional methods insufficient for general vector field analysis

Limitations of traditional methods are encountered **frequently** and **unknown a priori**



Traditional methods insufficient for general vector field analysis

Limitations of traditional methods are encountered **frequently** and **unknown a priori**

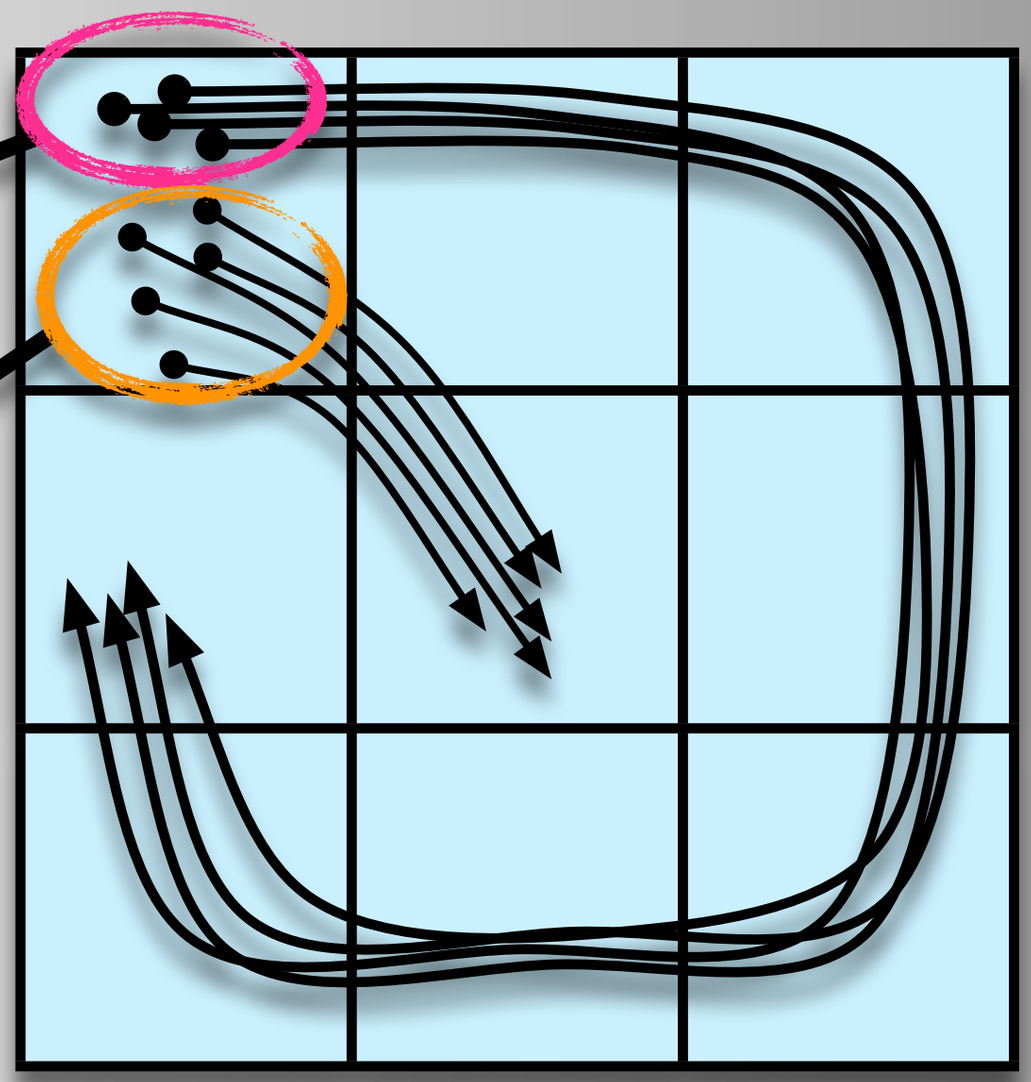


Traditional methods insufficient for general vector field analysis

Limitations of traditional methods are encountered **frequently** and **unknown a priori**

Parallelize over **blocks**

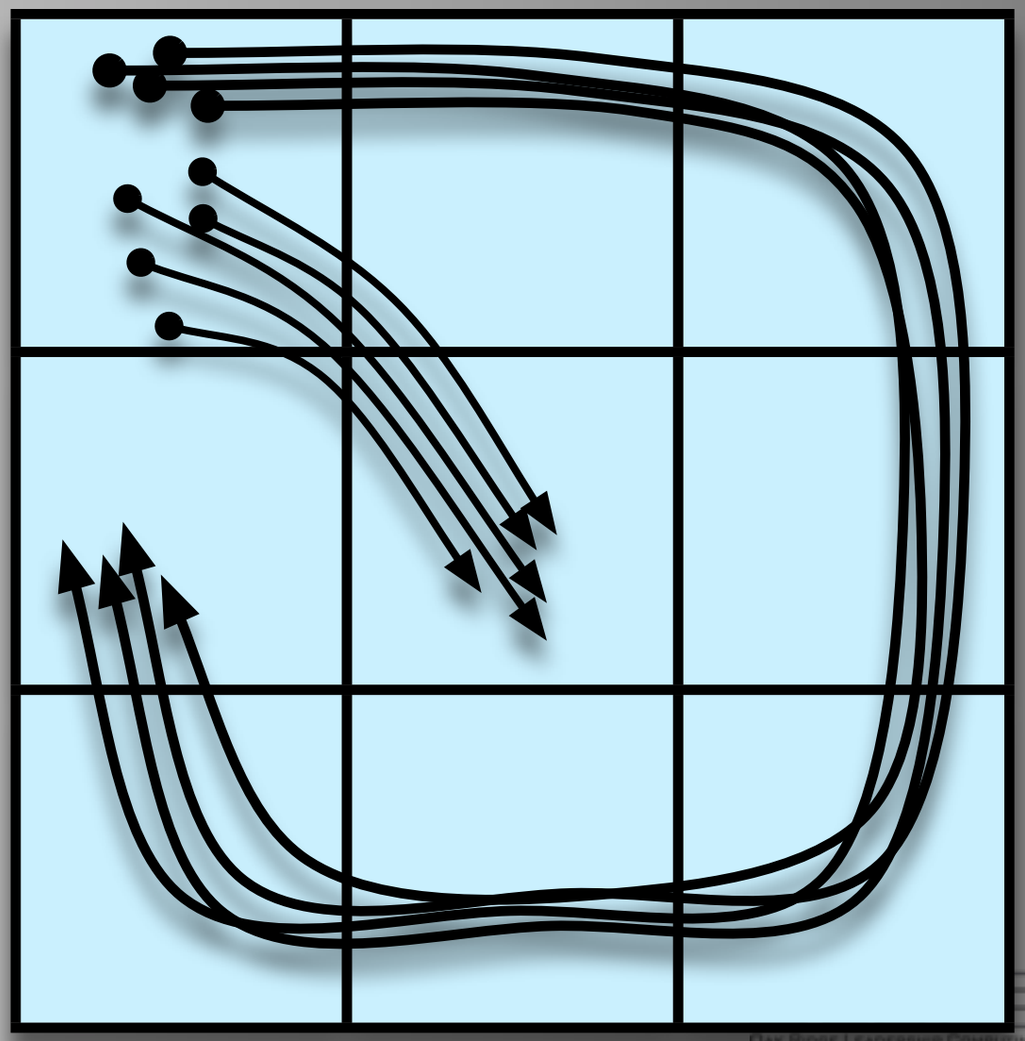
Parallelize over **seeds**



We need a new scalable algorithm

- Works well in all use cases
- Use a combination of seed and block parallelization
  - Dynamically steer computation
  - Allocate resources where needed
- Parallelize over seeds **and** blocks
- Goals:
  1. Maximize processor utilization
  2. Minimize IO
  3. Minimize Communication
- See Super Computing 2009 paper

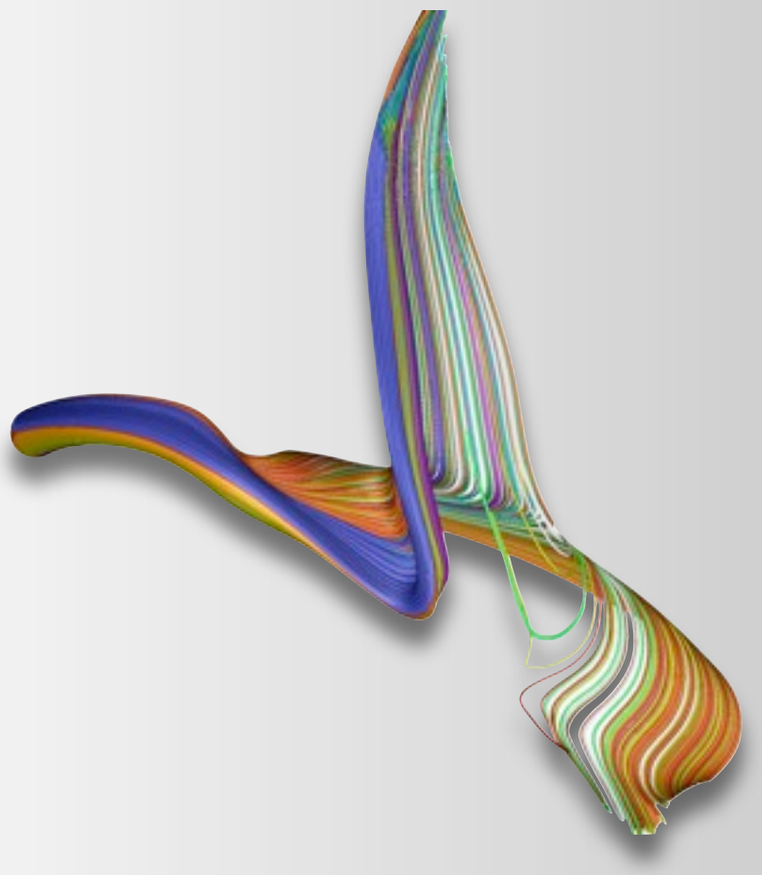
Scalable Computation of Streamlines on Very Large Datasets,
Pugmire, et al



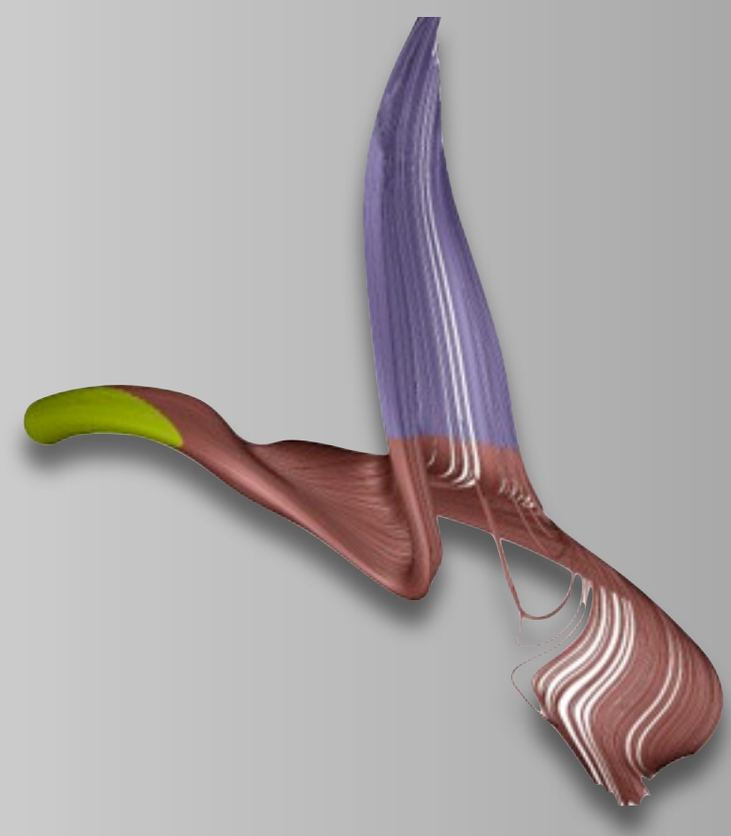
Workload distribution in supernova simulation

Parallelization by:

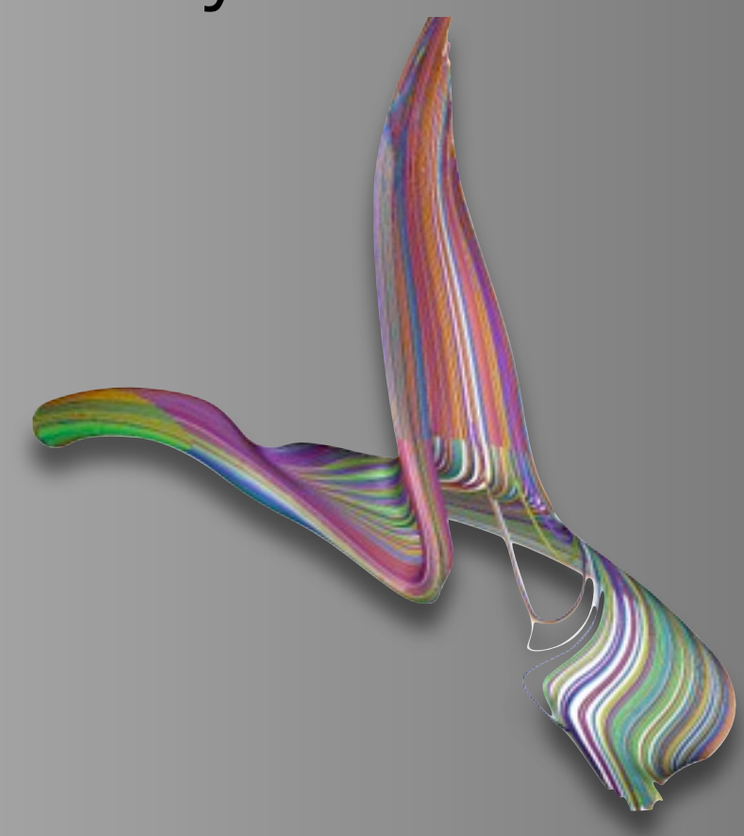
Seeds



Blocks



Hybrid

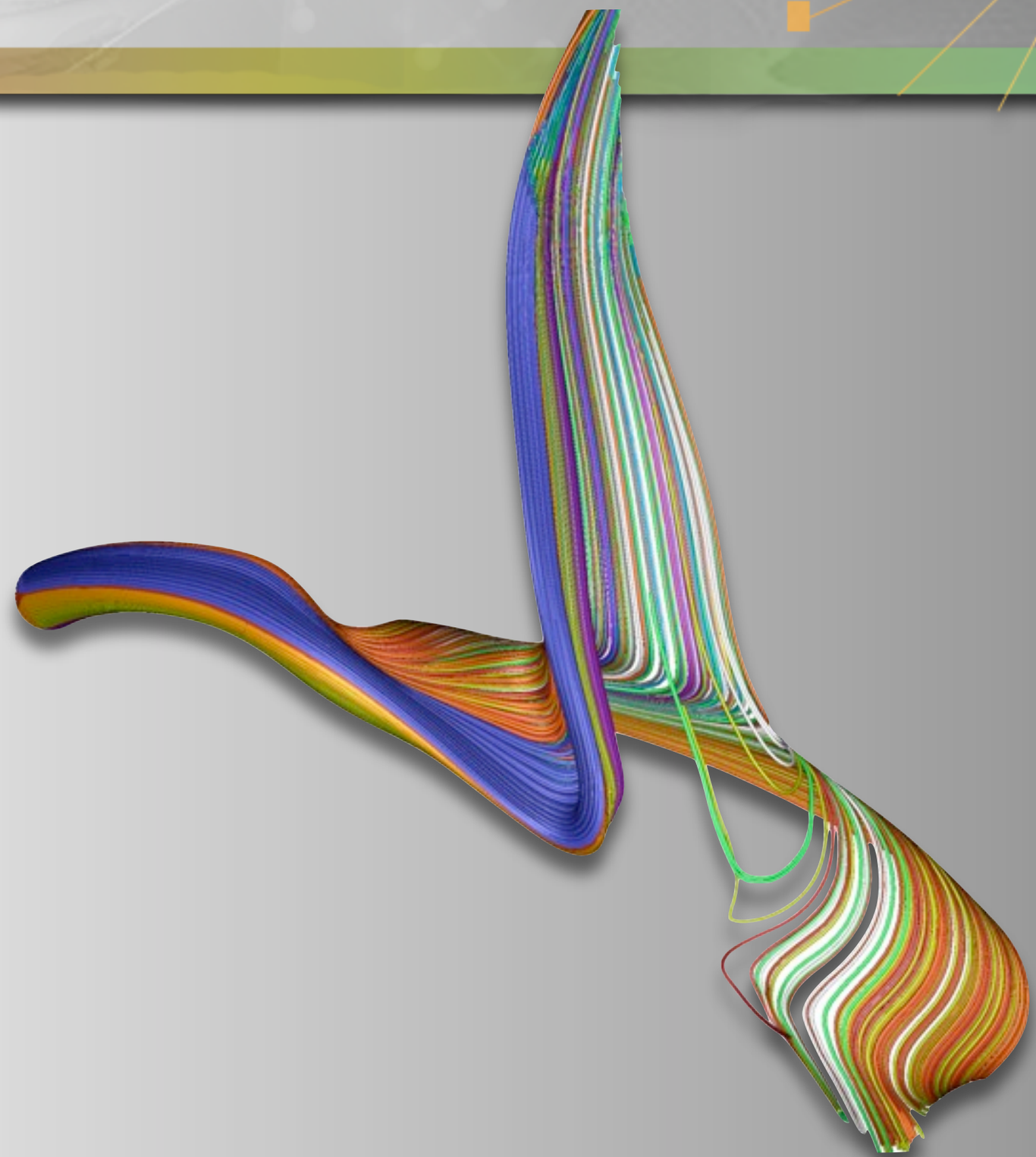


Colored by processor doing integration

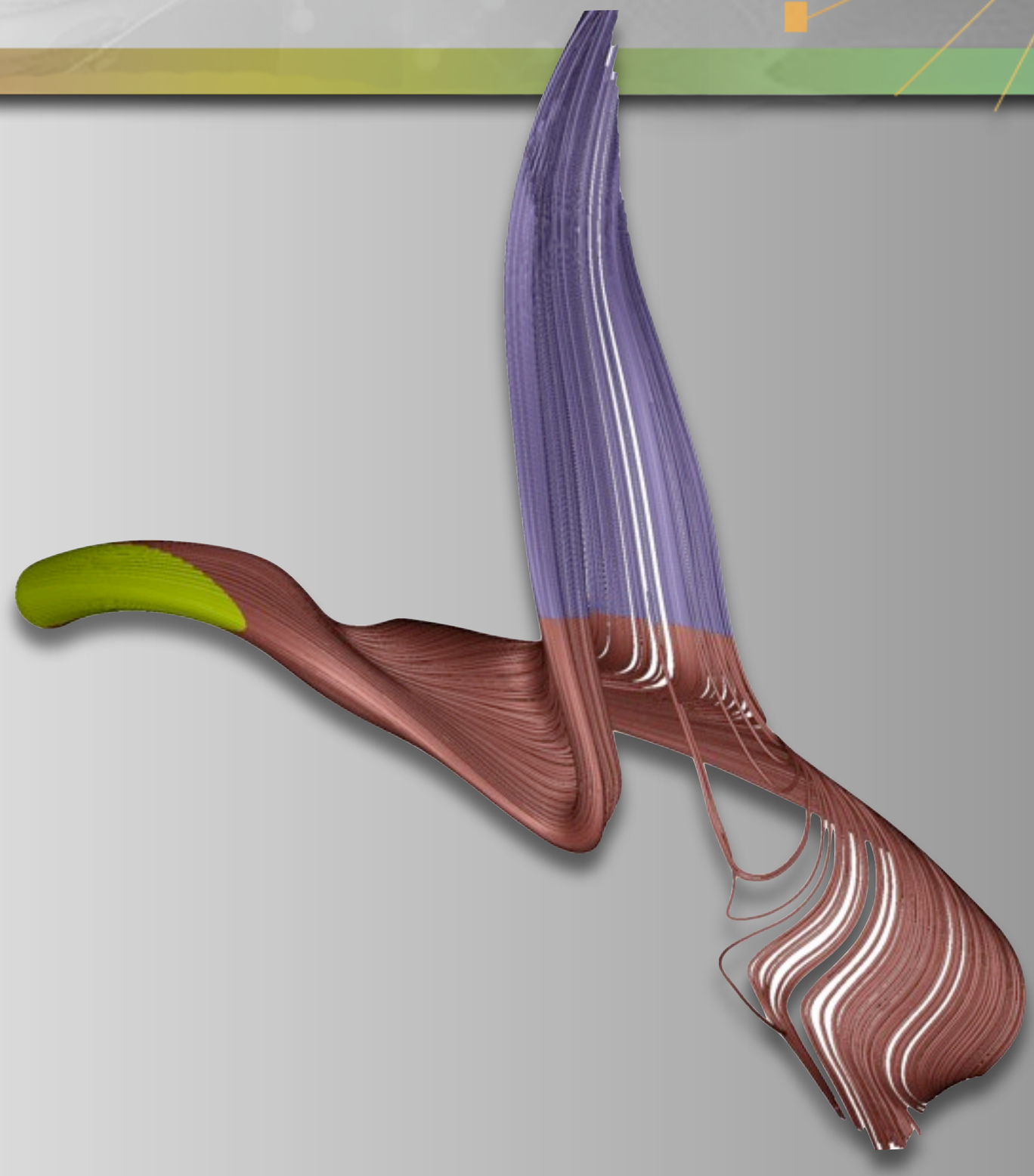
GenASiS data courtesy E. Endeve, C. Cardall, R. Budiardja, and A. Mezzacappa

Workload distribution in seed algorithm

Too much I/O

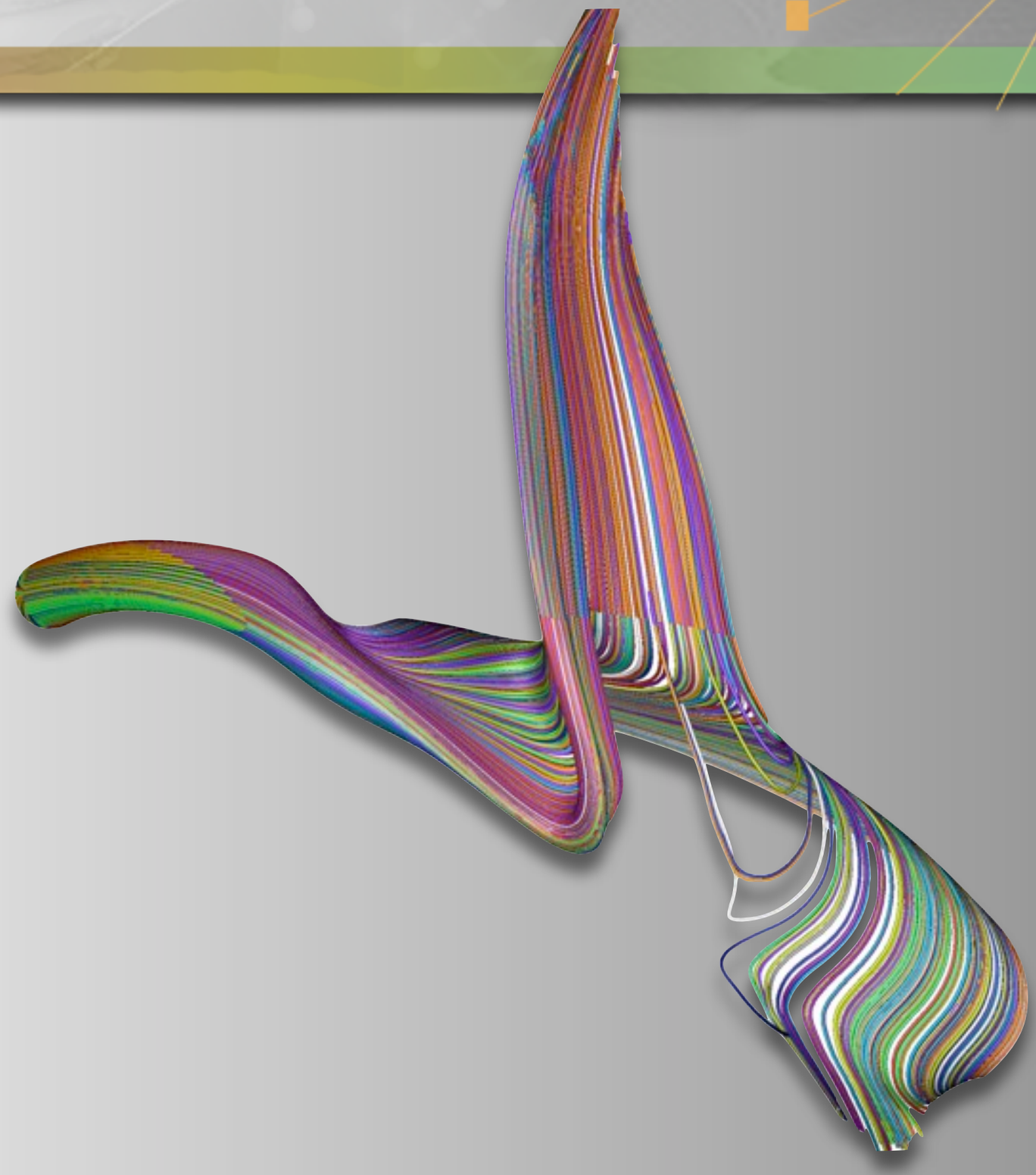


Workload distribution in block algorithm



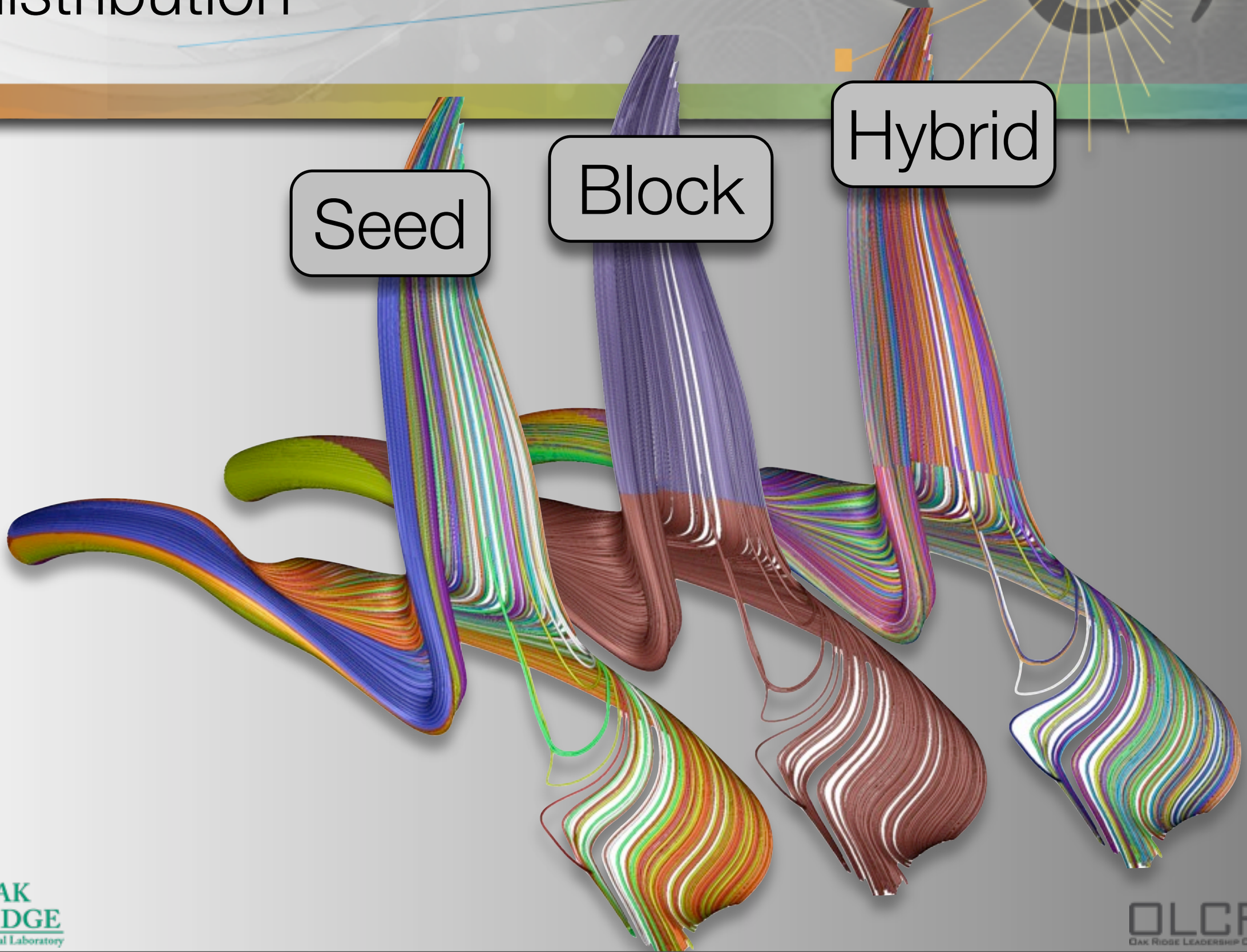
Starvation

Workload distribution in hybrid algorithm

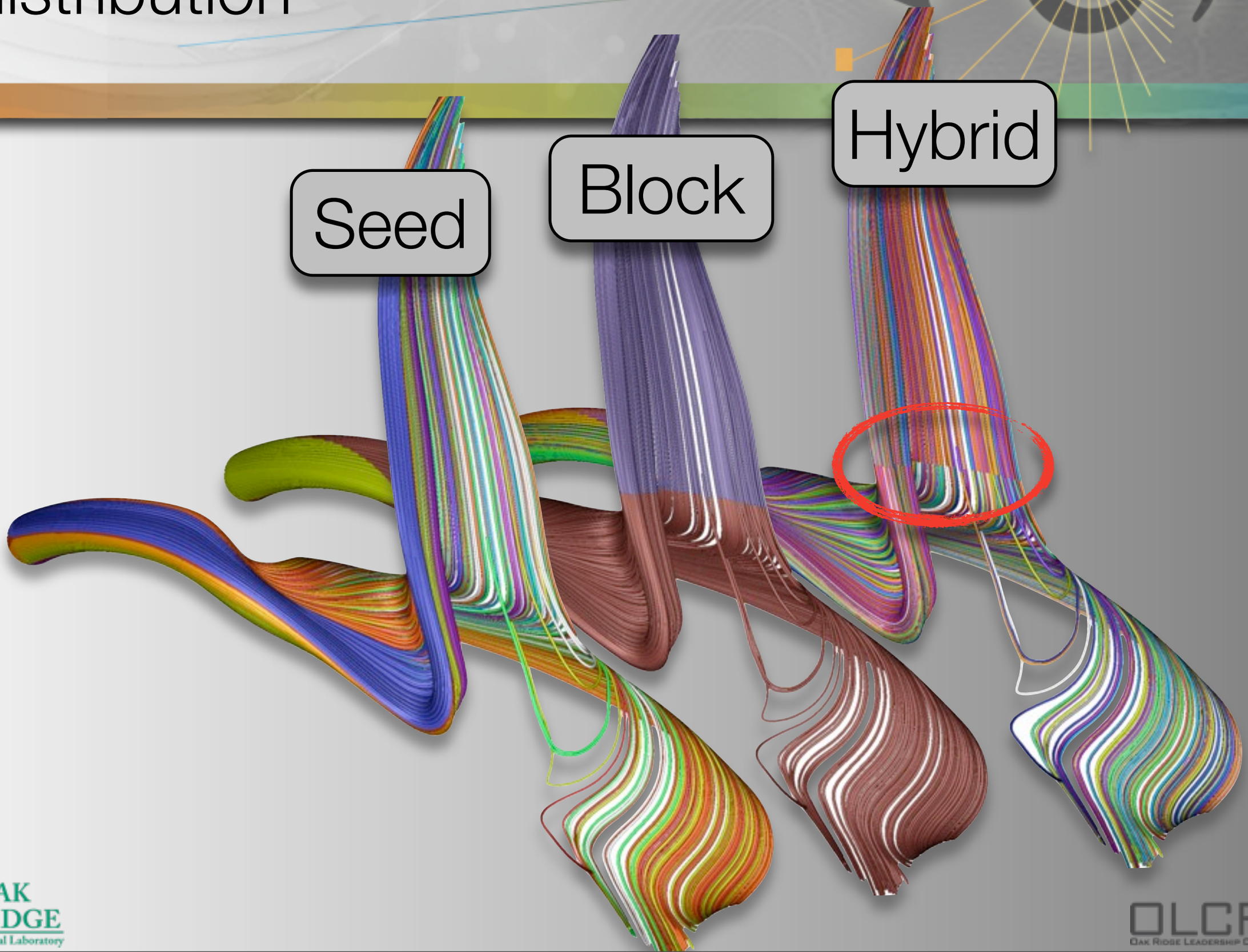


Just right

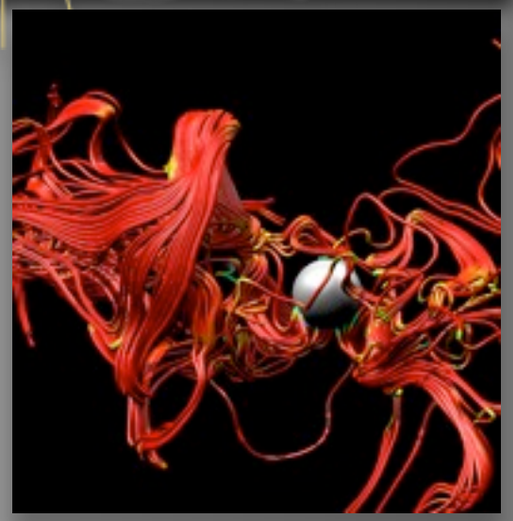
Comparison of workload distribution

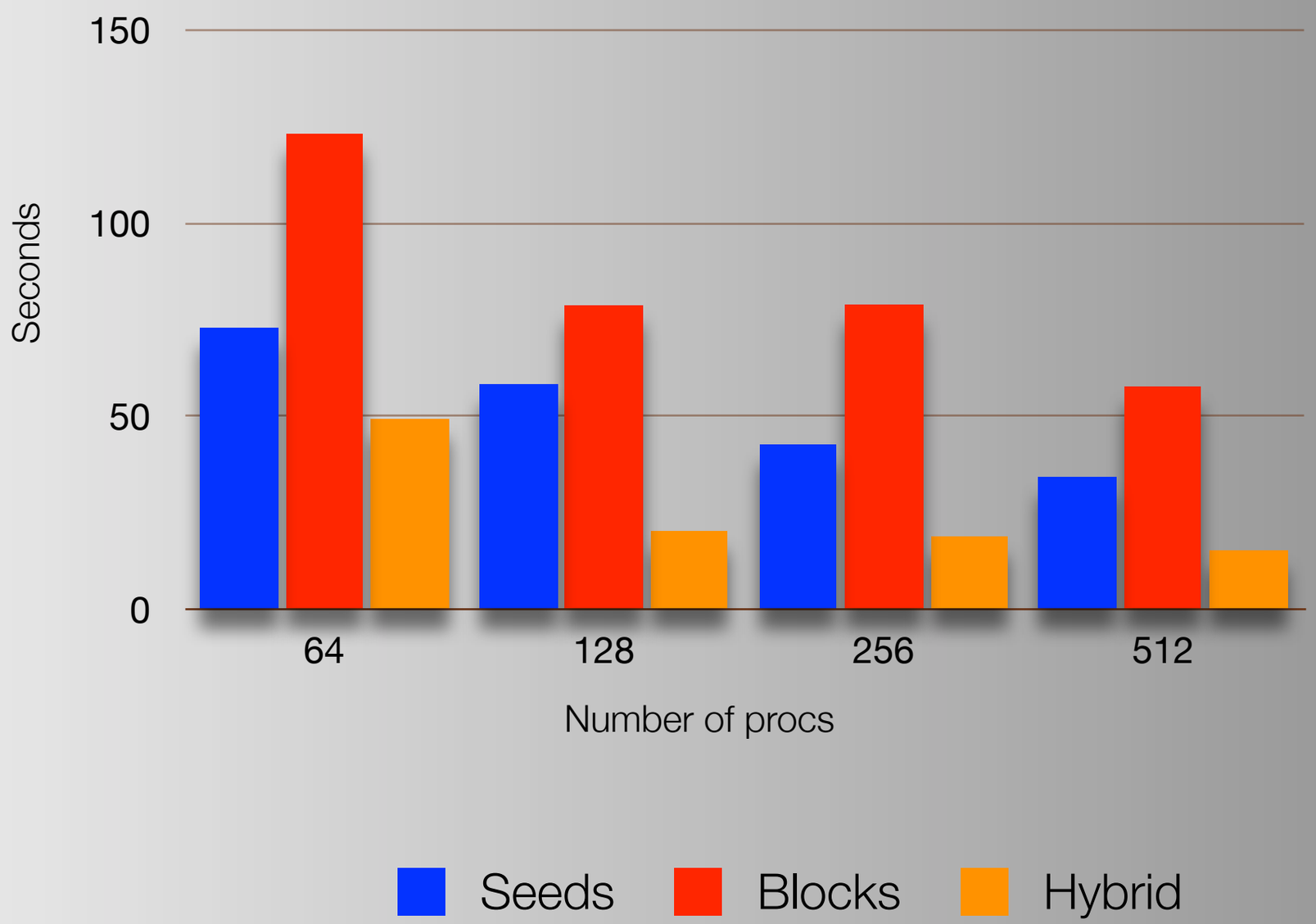


Comparison of workload distribution

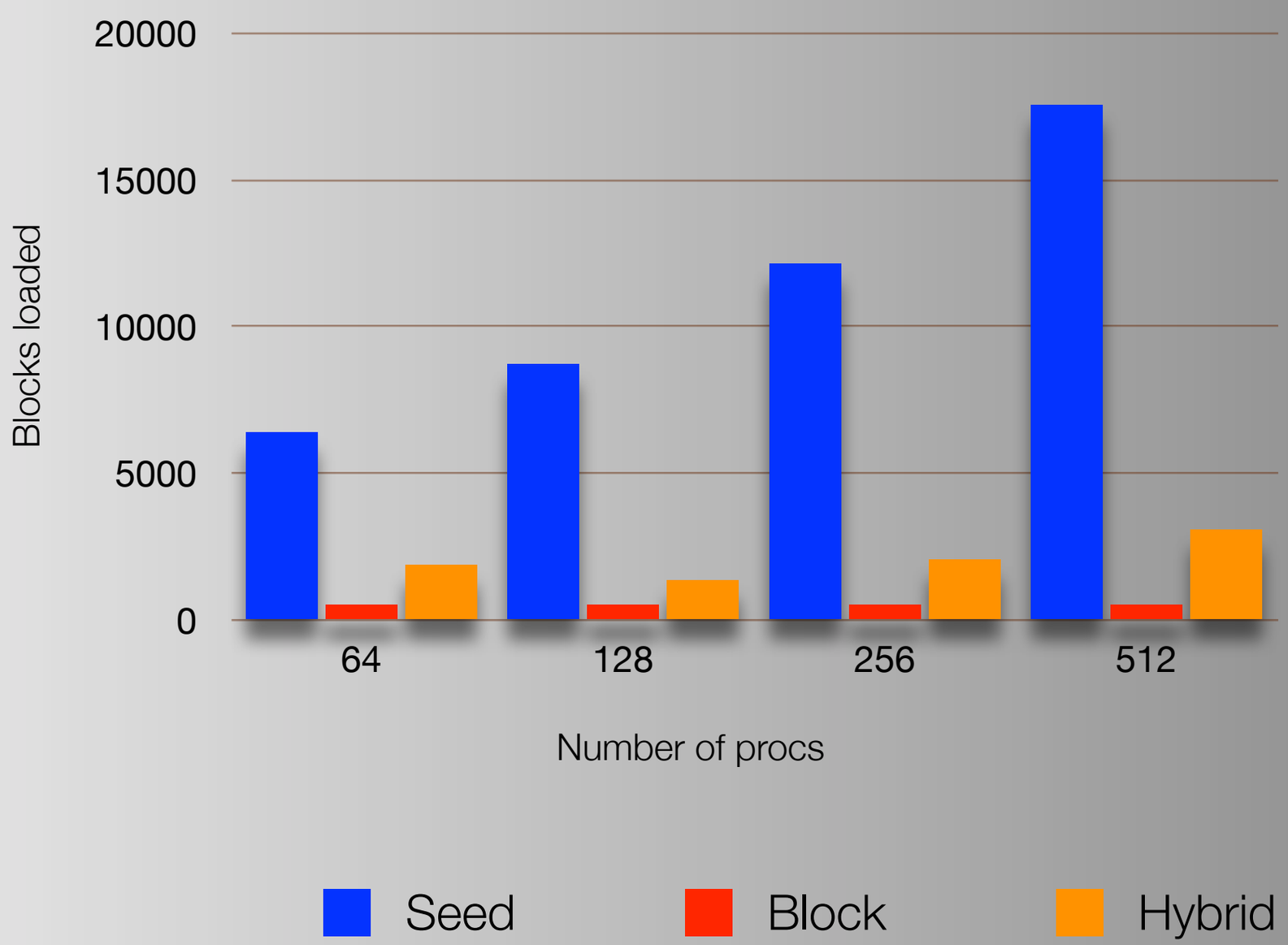


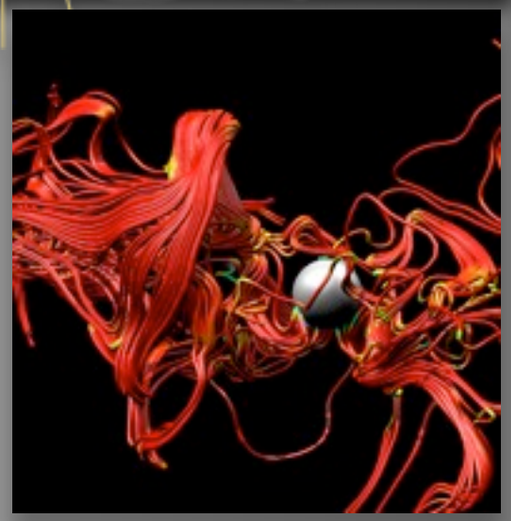
Astrophysics Test Case:
Total time to compute 20,000 Streamlines



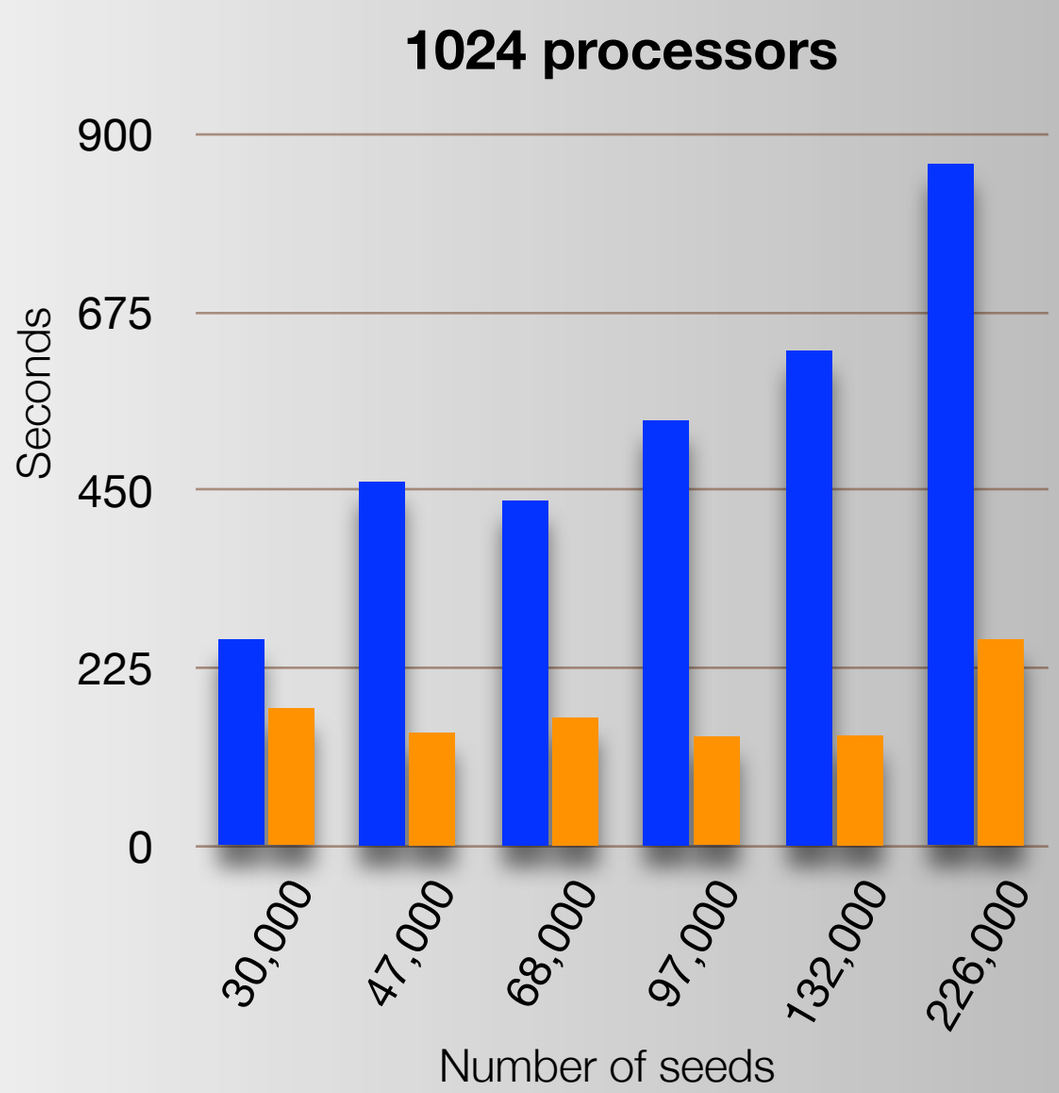


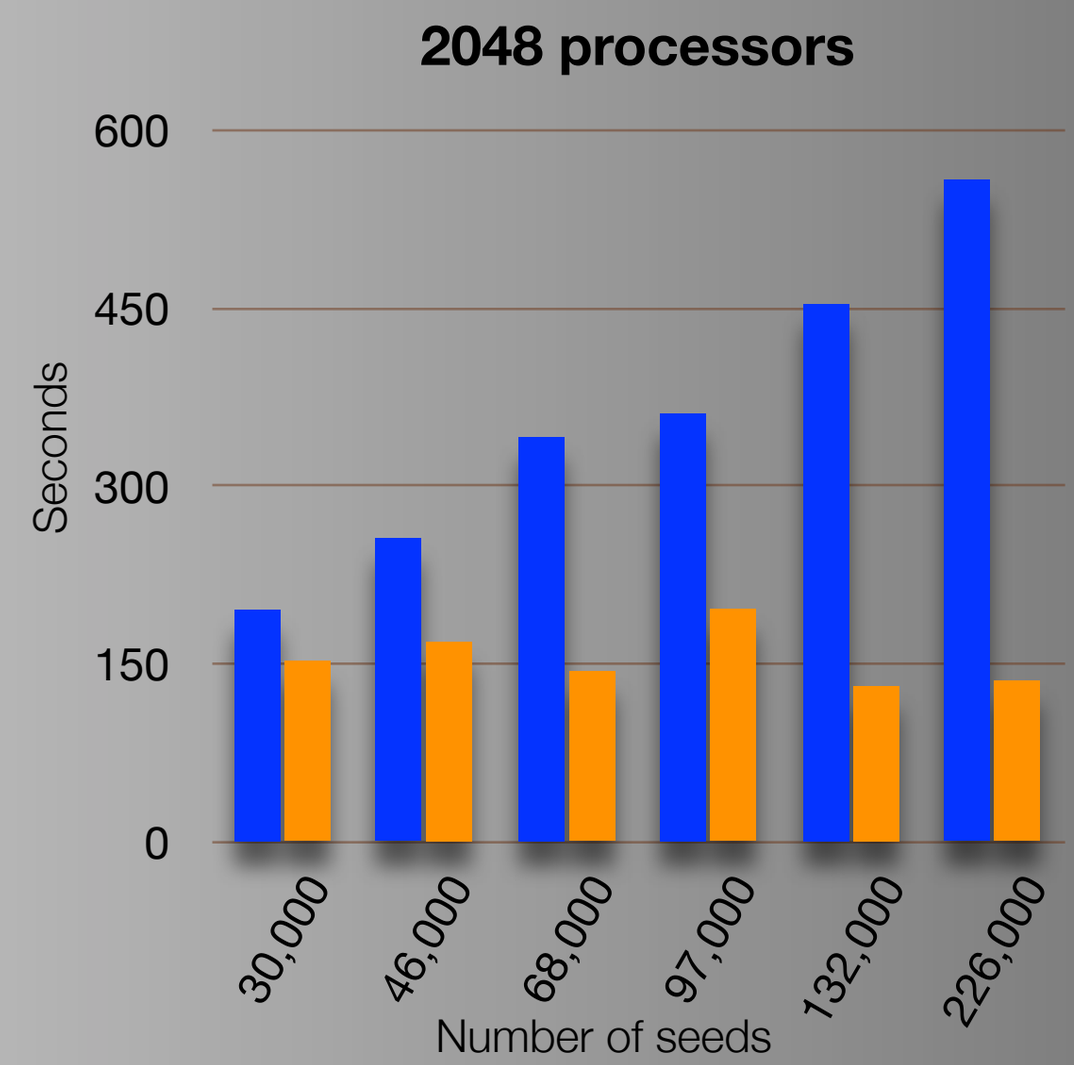
Astrophysics Test Case: Number of blocks loaded





Strong Scaling
Total time to compute streamlines



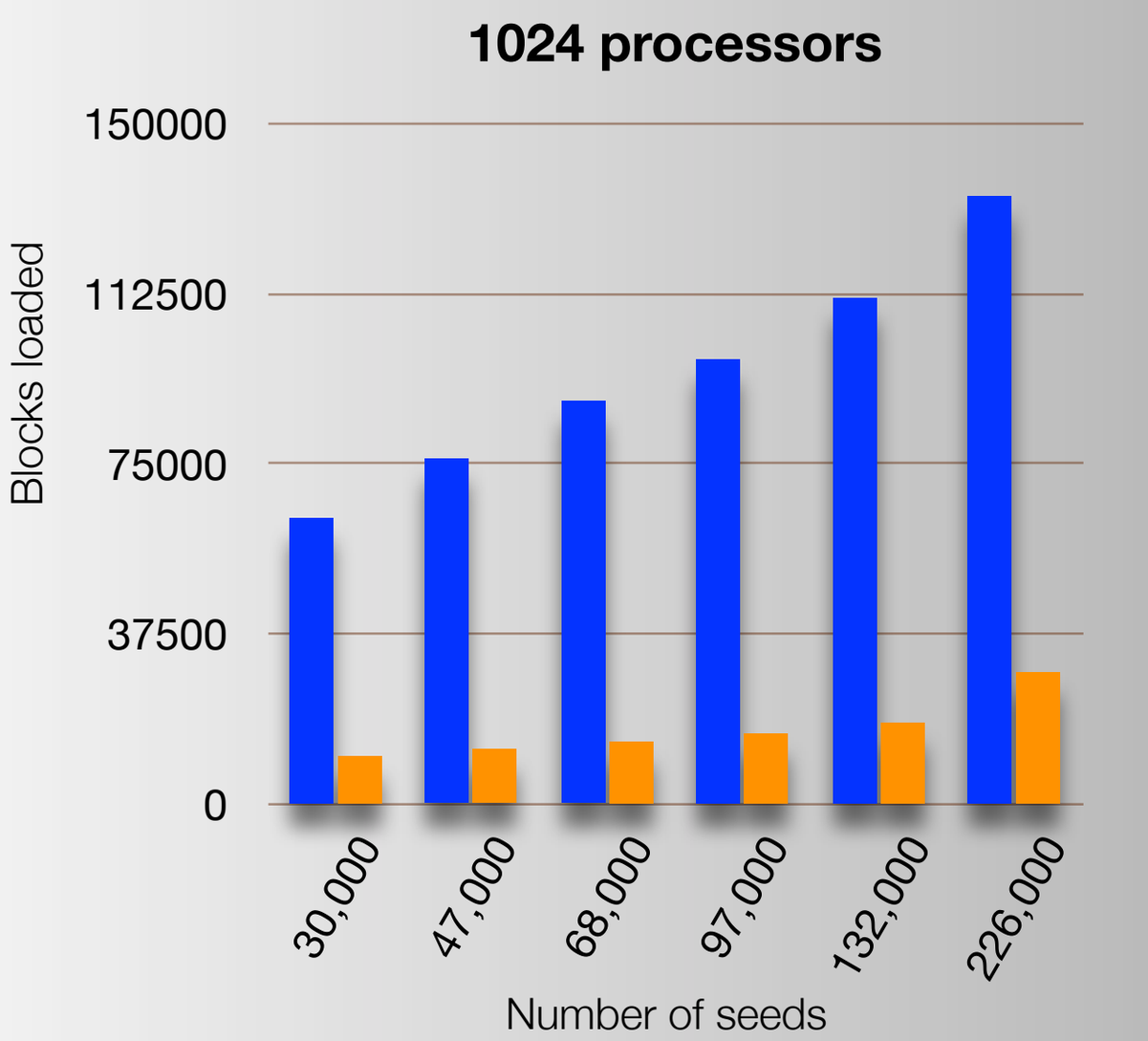


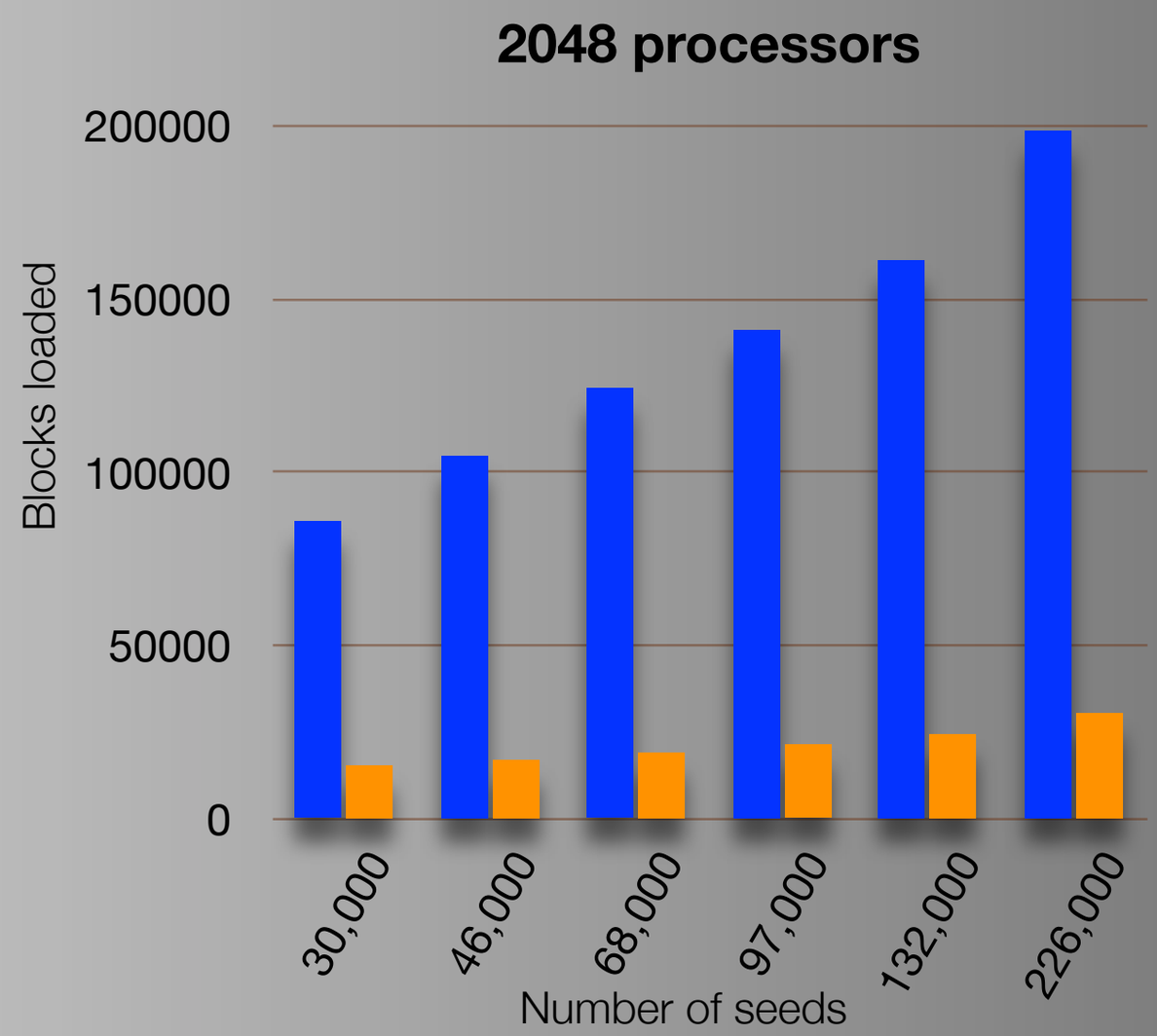
Seed

Hybrid

Block parallelization won't run!

Strong Scaling
Total number of blocks loaded





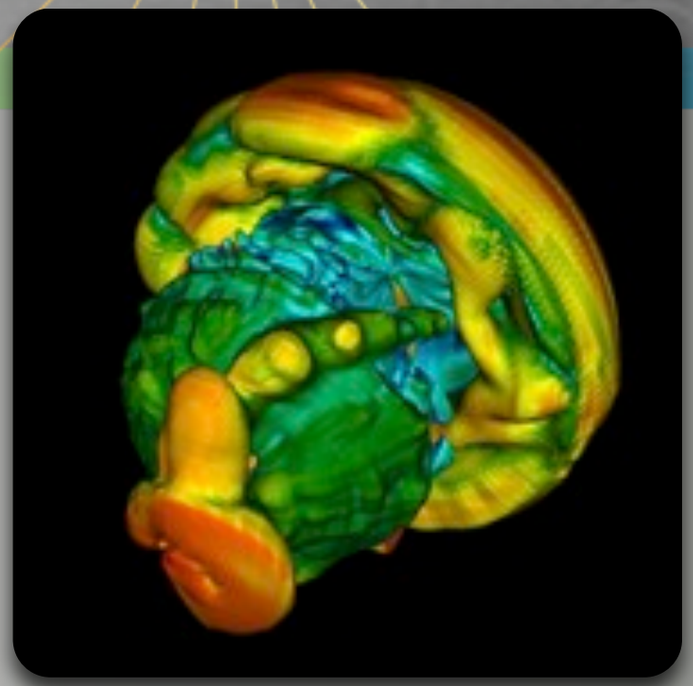
Seed

Hybrid

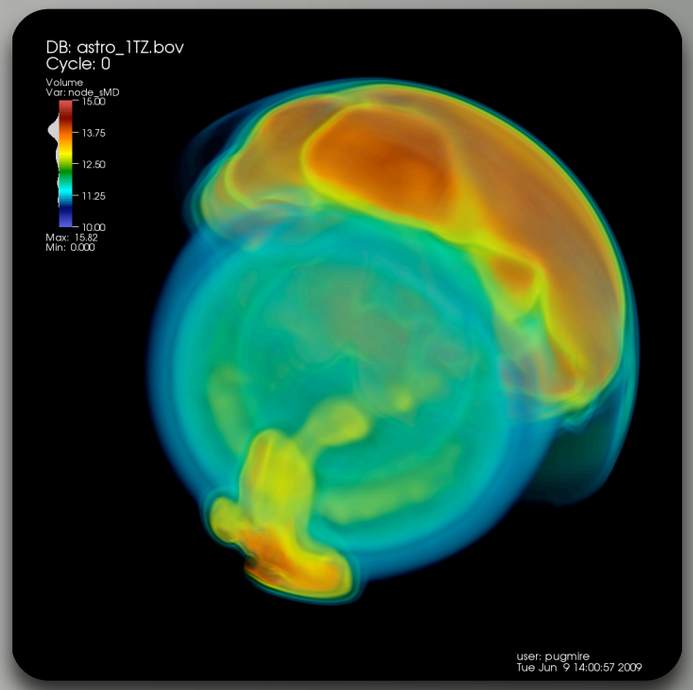
Parallel pipeline methods can take us a long way

- Weak scaling study 2009 (isocontouring, volume rendering): ~63 million cells/core

Machine	Type	Problem Size	# cores
Jaguar	Cray XT5	2T	32k
Franklin	Cray XT4	1T, 2T	16k, 32k
Dawn	BG/P	4T	64k
Juno	Linux	1T	16k, 32k
Ranger	Sun Linux	1T	16k
Purple	AIX	0.5T	8k



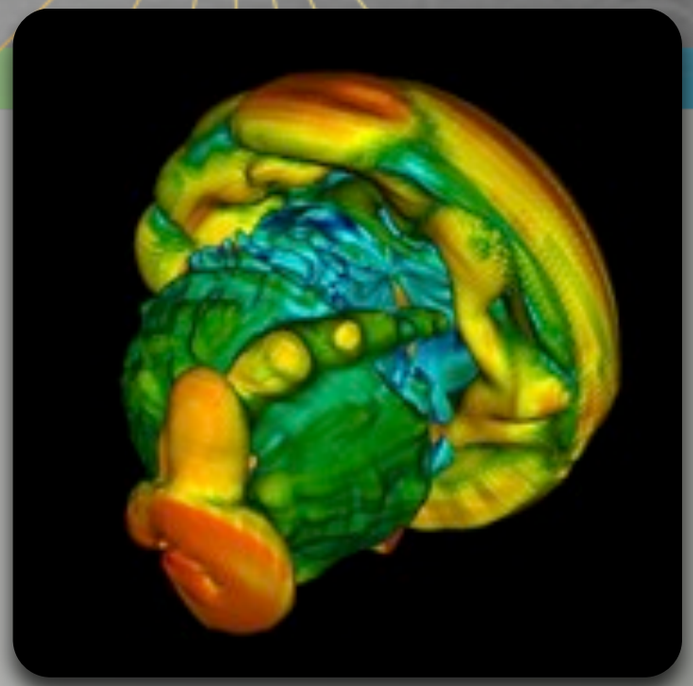
2T cells, 32K procs



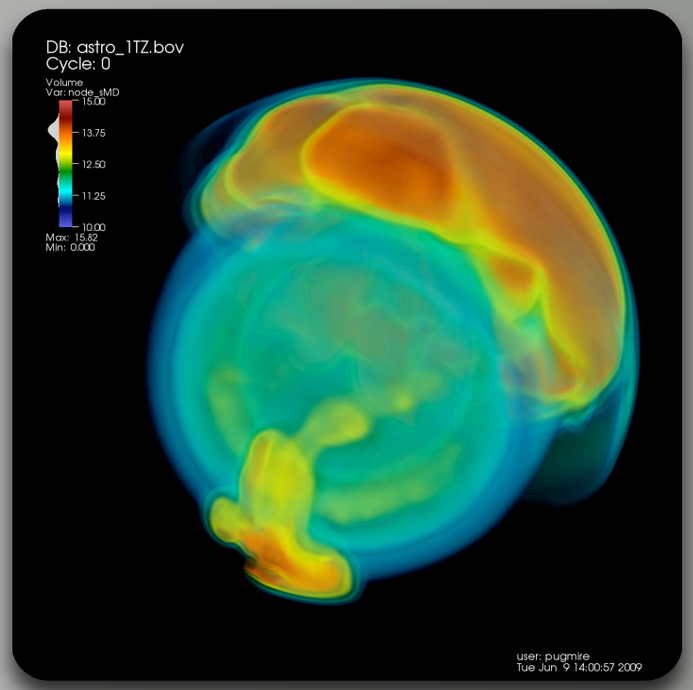
Parallel pipeline methods can take us a long way

- Weak scaling study 2009 (isocontouring, volume rendering): ~63 million cells/core

Machine	Type	Problem Size	# cores
Jaguar	Cray XT5	2T	32k
Franklin	Cray XT4	1T, 2T	16k, 32k
Dawn	BG/P	4T	64k
Juno	Linux	1T	16k, 32k
Ranger	Sun Linux	1T	16k
Purple	AIX	0.5T	8k



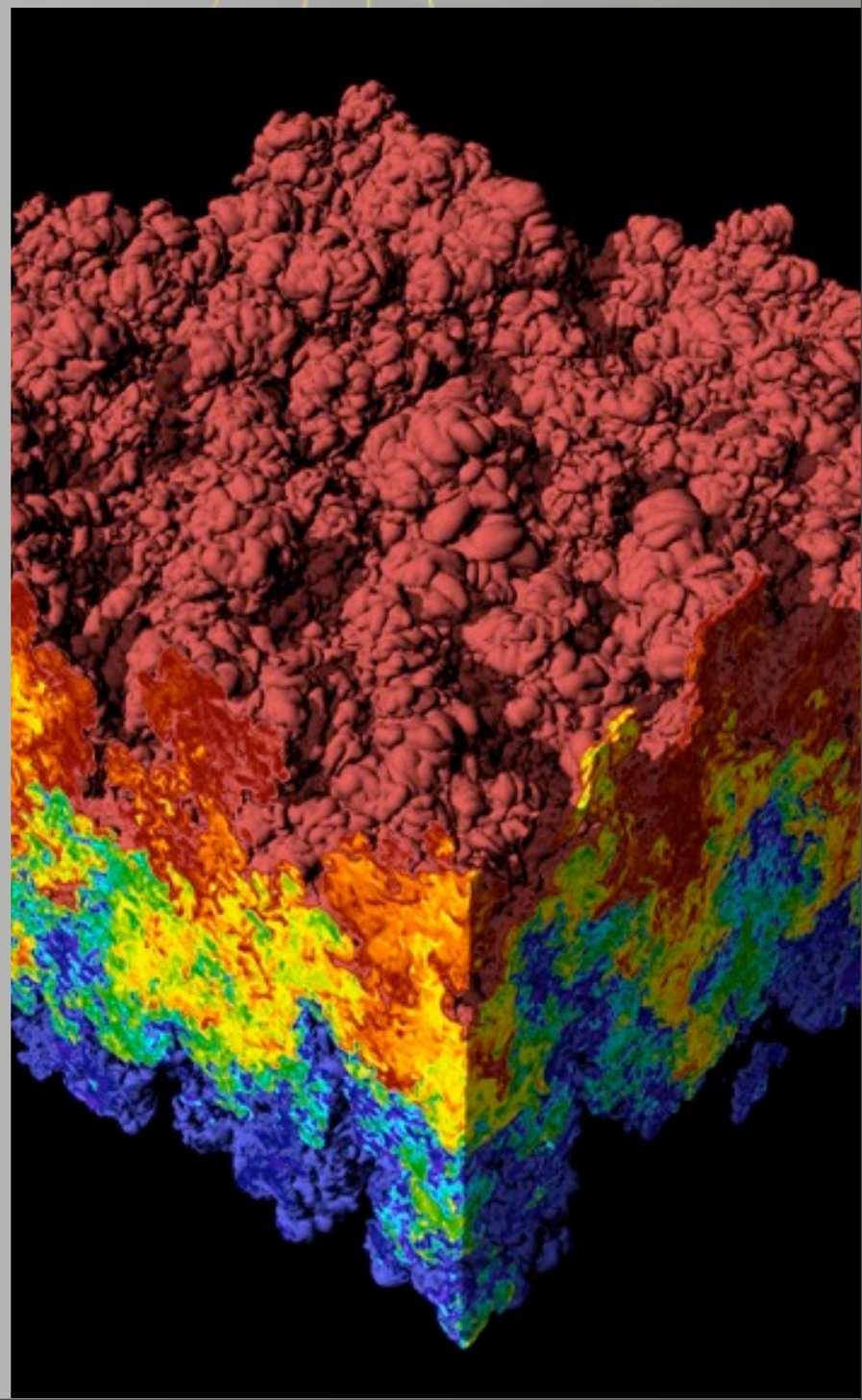
2T cells, 32K procs



Since this work, people have reached 8 trillion cells.

VisIt: an end user visualization and analysis tool for extremely large data

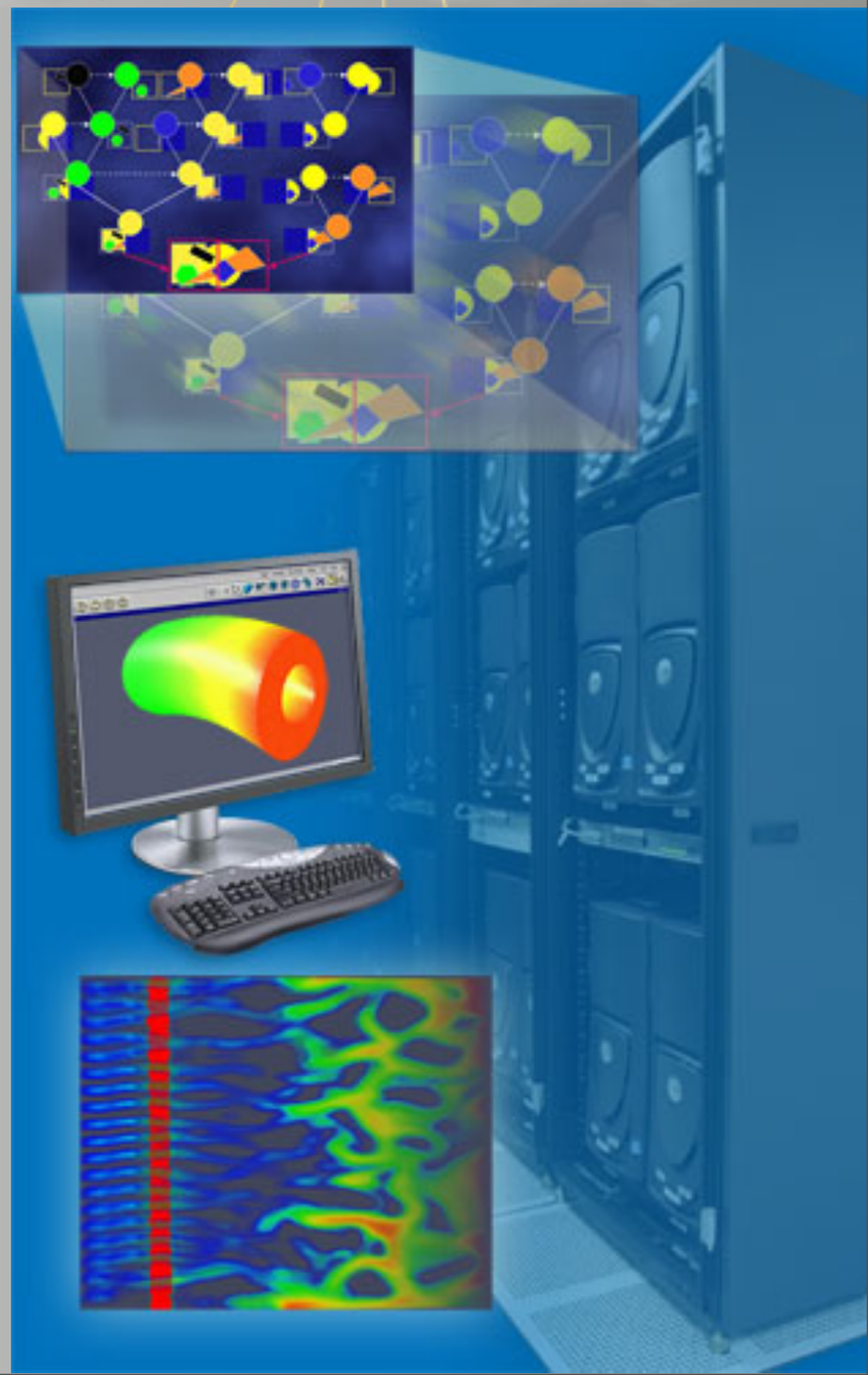
- 5 major use cases: data exploration, data analysis, visual debugging, comparison, and presentation
- Demonstrated scaling to >100k cores
- Strong emphasis on building a usable, robust tool for scientists.
  - Plugin model allows for wide range of capabilities
- Open source project, with direct support from DOE (SciDAC, NEAMS, ASC, BER) and NSF (TeraGrid XD)
- visitusers.org wiki with 400+ pages of documentation and over 80,000 page views
- Over 100,000 downloads
- R&D100 Award in 2005



OAK RIDGE National Laboratory

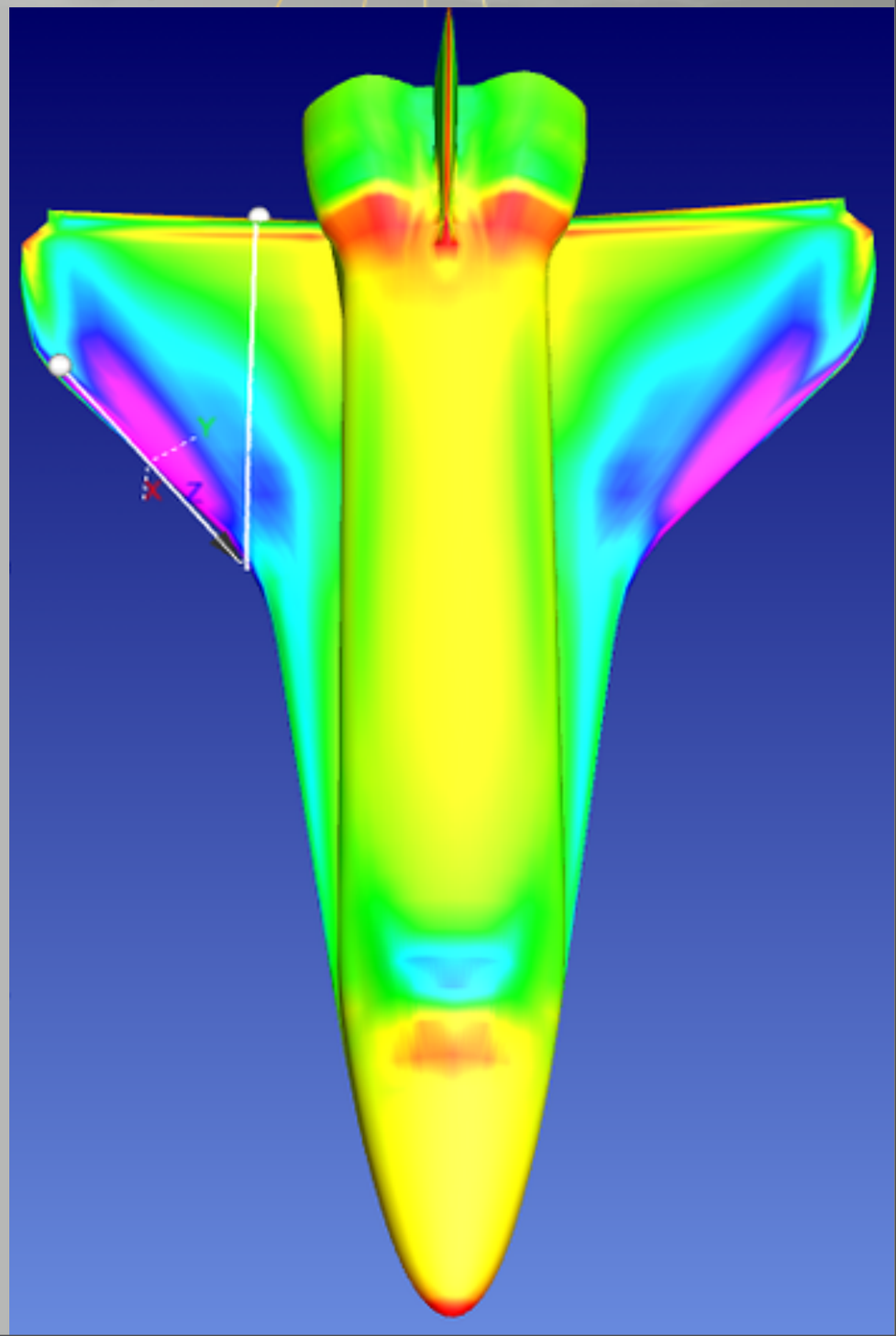
ParaView: open-source, multi-platform data analysis and visualization application

- Based on VTK: Deep pipeline modification
- Quantitative and qualitative use cases
- Developed to process large datasets using distributed memory computing resources: laptops to supercomputers
- Superb, best-in-class rendering infrastructure
- Customizable GUI
- Access grid integration for collaboration
- Commercial support from Kitware



EnSight: leading commercial parallel visualization tool

- Specialized for engineering analysis
  - CFD, FEA, etc.
- Commercial support from CEI
- Distributed rendering infrastruture
  - Direct Powerwall support
  - Virtual reality
- Distributed parallel data processing and rendering



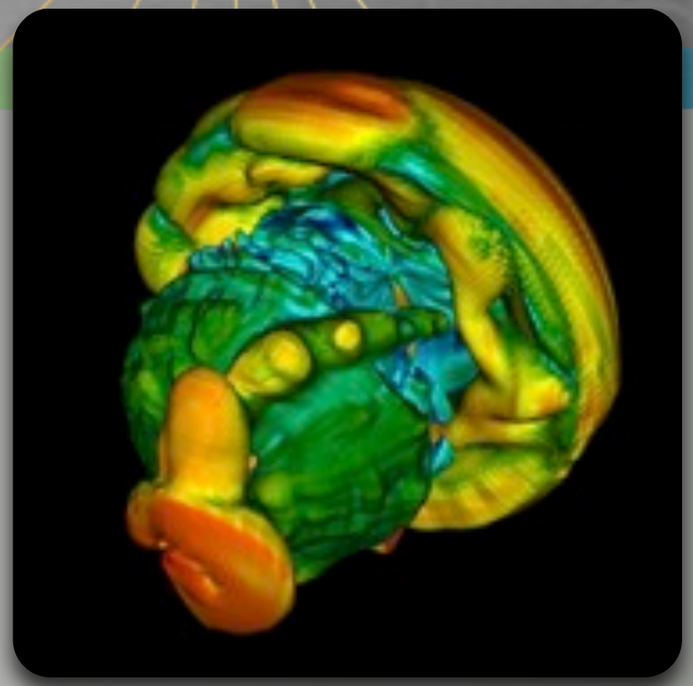
Performance is highly dependent upon I/O rates

- Reading data into memory takes the most time
  - Much more than data processing
  - Much more than rendering and compositing
- Parallel filesystems are essential for large data
- Data formats designed for parallel filesystems are essential for large data

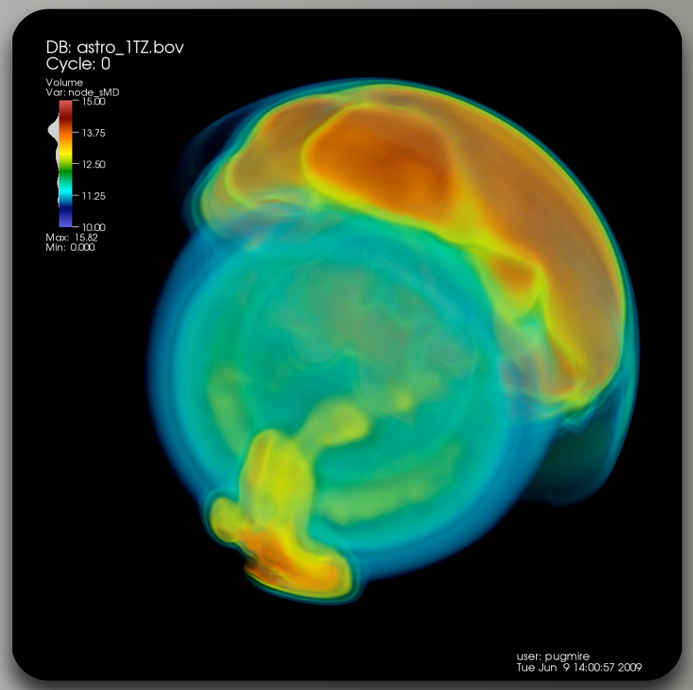
Trillion cell study confirms that I/O dominates

- Weak scaling study 2009 (isocontouring, volume rendering): ~63 million cells/core

Machine	Type	Problem Size	# cores
Jaguar	Cray XT5	2T	32k
Franklin	Cray XT4	1T, 2T	16k, 32k
Dawn	BG/P	4T	64k
Juno	Linux	1T	16k, 32k
Ranger	Sun Linux	1T	16k
Purple	AIX	0.5T	8k



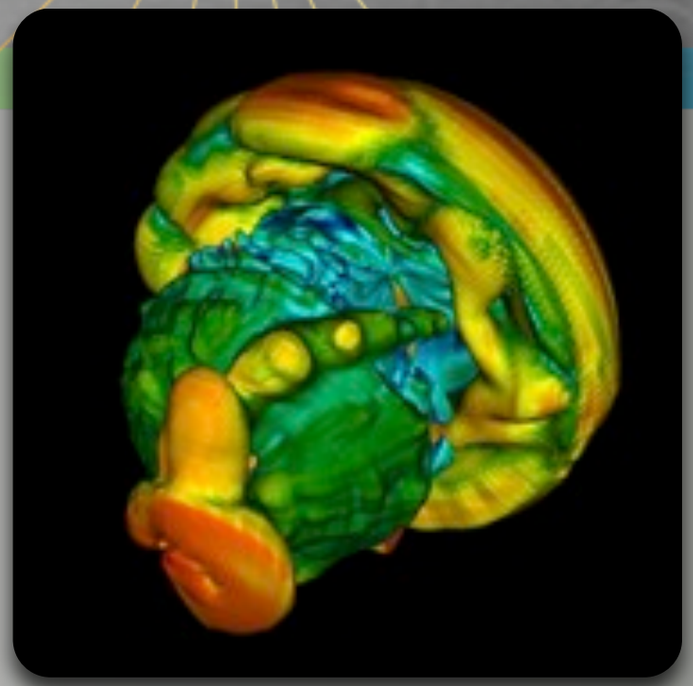
2T cells, 32K procs



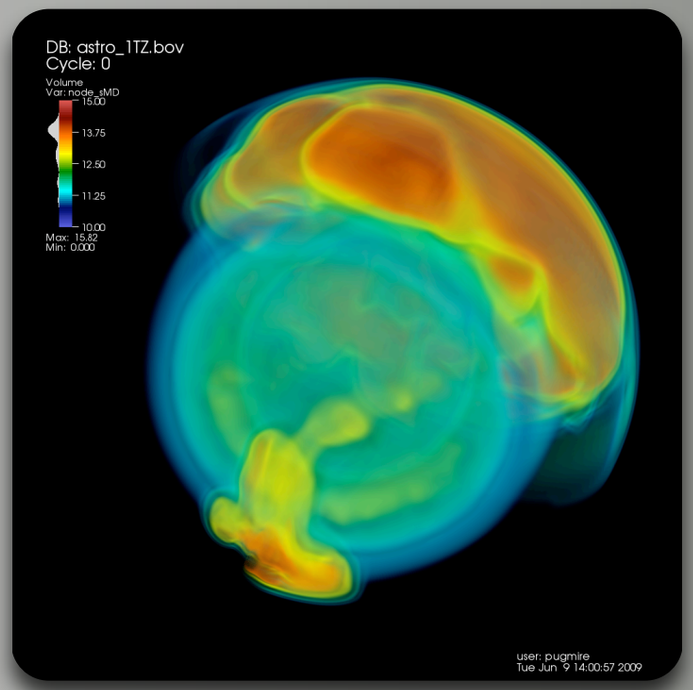
Trillion cell study confirms that I/O dominates

- Weak scaling study 2009 (isocontouring, volume rendering): ~63 million cells/core

Machine	Type	Problem Size	# cores
Jaguar	Cray XT5	2T	32k
Franklin	Cray XT4	1T, 2T	16k, 32k
Dawn	BG/P	4T	64k
Juno	Linux	1T	16k, 32k
Ranger	Sun Linux	1T	16k
Purple	AIX	0.5T	8k



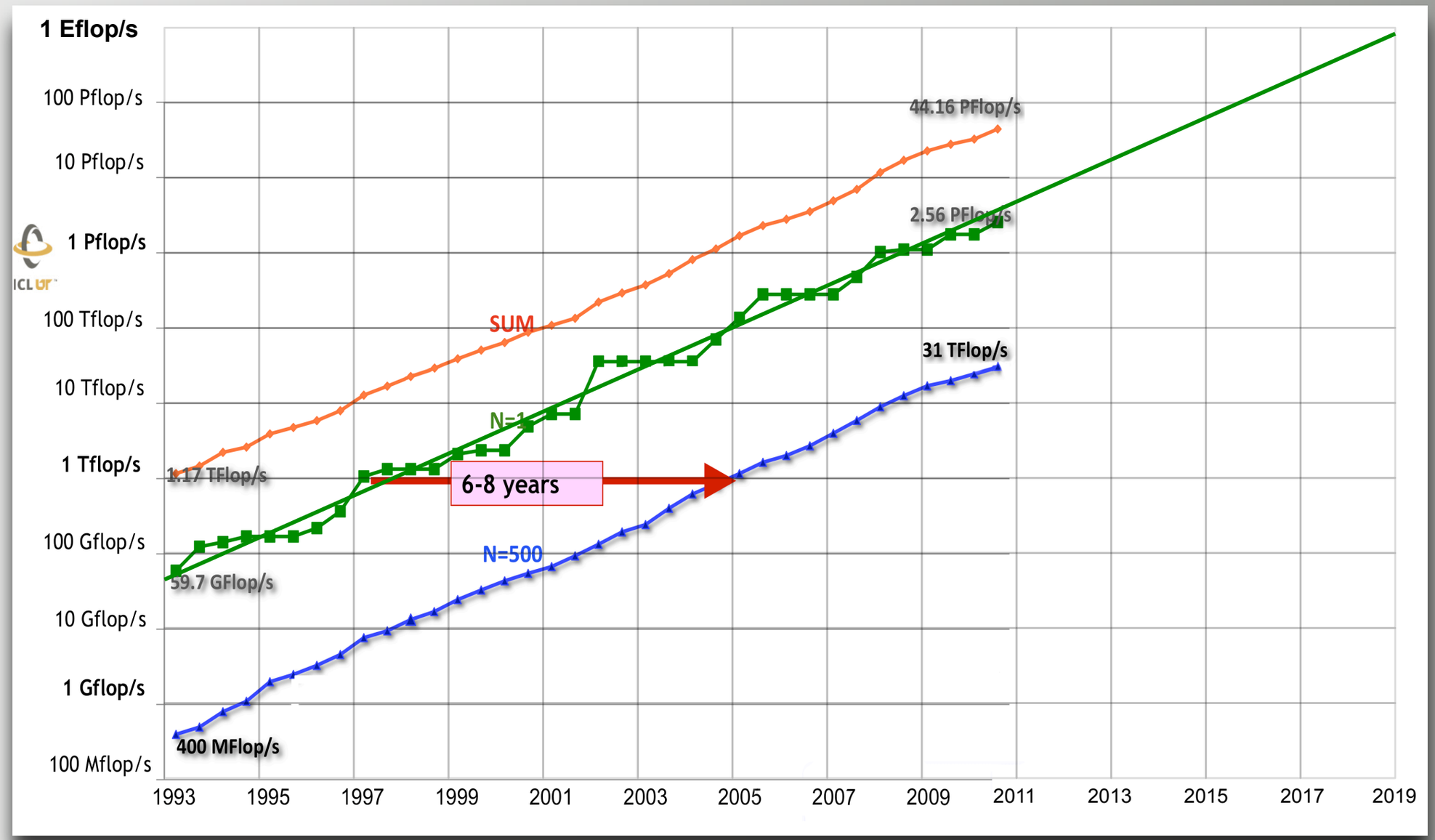
2T cells, 32K procs



-Approx I/O time: 2-5 minutes
-Approx processing time: 10

Challenges for the Exascale

Top 500 extrapolation



Challenges for the Exascale

- Billion-way concurrency
- Constrained memory environment
- MTBF extremely high (minutes)
- Heterogenous computing
- Very constrained I/O (in relation to compute capability)

I/O wall = memory wall

- The rate of improvement in microprocessor speed exceeds the rate of improvement in memory speed.
  - Latency increasing
  - Relative bandwidth decreasing
- The rate of improvement in microprocessor speed exceeds the rate of improvement in network speed.
- Even if we had an infinitely fast network, we're still limited by memory bandwidth.
- Multicore only makes it significantly worse (bandwidth/core is decreasing)

I/O wall = memory wall

- The rate of improvement in microprocessor speed exceeds the rate of improvement in memory speed.
  - Latency increasing
  - Relative bandwidth decreasing
- The rate of improvement in microprocessor speed exceeds the rate of improvement in network speed.
- Even if we had an infinitely fast network, we're still limited by memory bandwidth.
- Multicore only makes it significantly worse (bandwidth/core is decreasing)

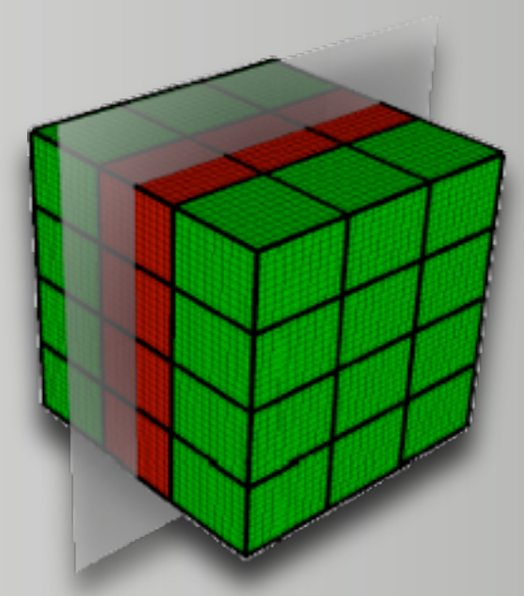
A few approaches

- Block skipping
- Data indexing
- Multiresolution
- Out of core
- In situ and co-processing
- Summary techniques

Block skipping

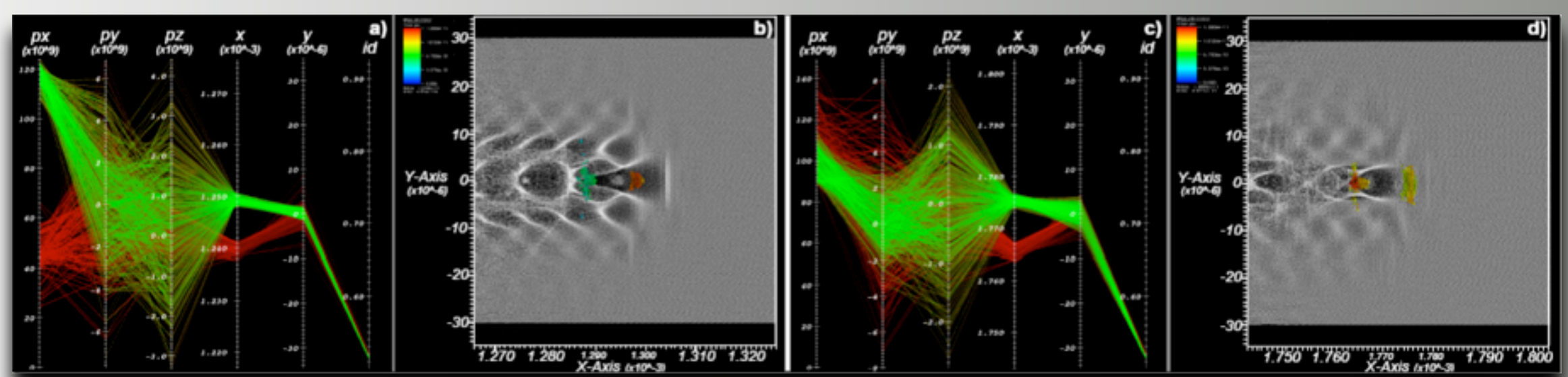
- Decomposed data
  - block structured, octtree, BSP, AMR, etc.
  - Allows efficient parallelism and load balancing
- Metadata stored with each portion of decomposition
  - Spatial extents
  - Data extents
- Allows for I/O optimization of users' queries

- Pros:
  - Significant I/O reductions
  - Successful at high parallelism
  - Contracts enable filter-filter optimizations
- Challenges:
  - Unsuitable for complex analysis



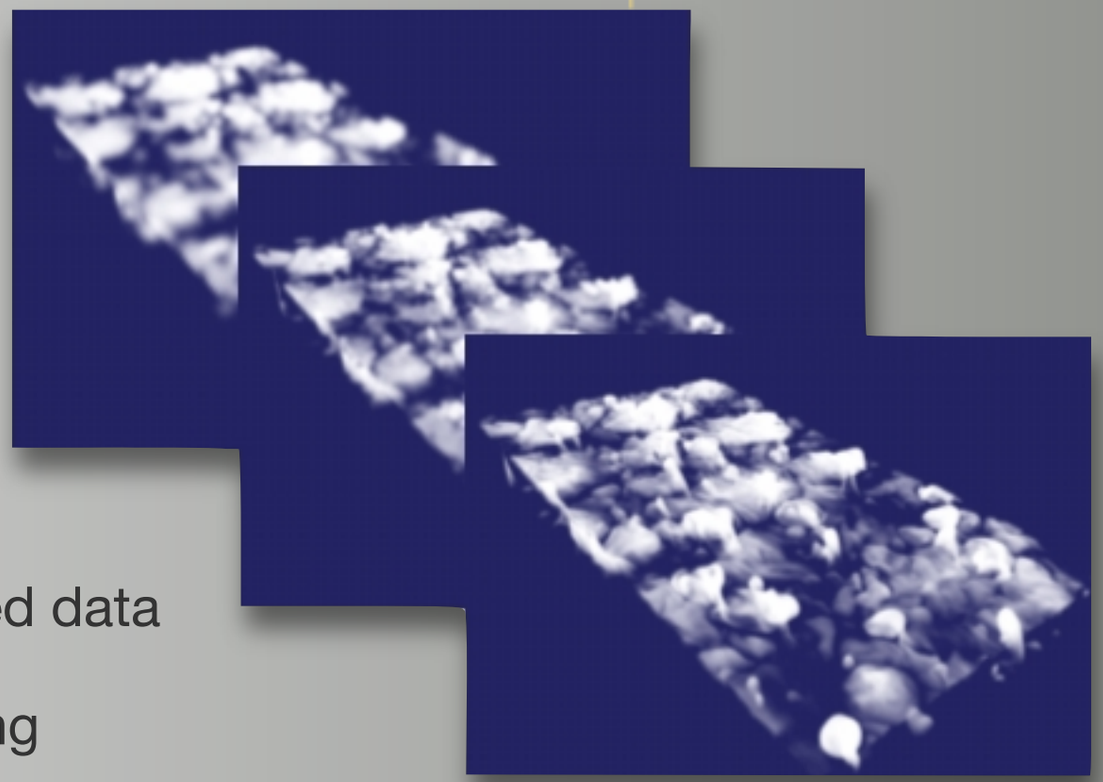
Bitmap indexing

- FastBit project from the SciDAC SDM Center.
- Accelerates interaction with laser wakefield simulations.
- Allows for interactive seeking of particles, with execution time linear in the amount of returned particles.
- See SC08 paper -> demonstrated 10^3 I/O performance improvement



Multiresolution

- Pros:
  - I/O bandwidth becomes a “knob”
  - Progressive updates are possible
- Challenges:
  - Error bounds can be difficult to compute
  - Users often unsure if they can believe coarsened data
  - Constrained coarsening techniques still maturing
  - Unstructured data can be a complicated statistical problem – spacefilling curves are a start



Out-of-core

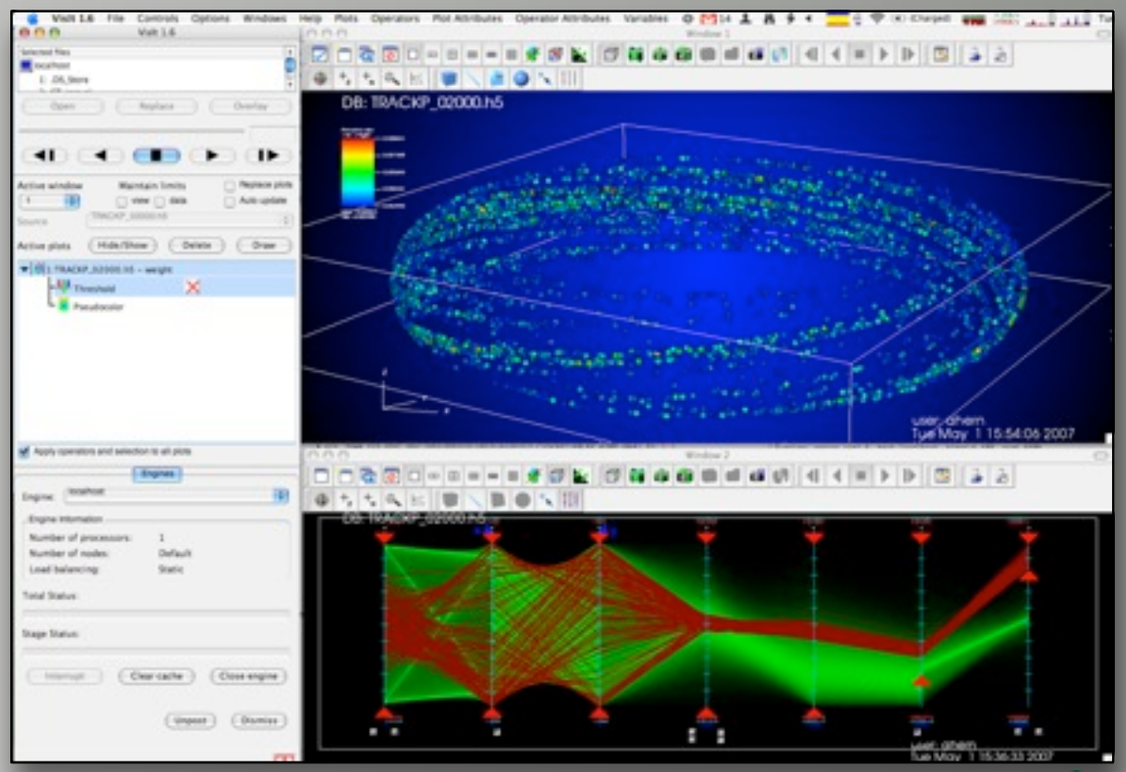
- Pros:
  - Significantly hides latency
  - Demonstrated for production vis
  - Interruptible
- Challenges:
  - Total I/O remains the same, so issues remain
  - Out-of-core priority is not always clear
  - Not applicable to all algorithms

In situ and co-processing

- Pros:
  - No I/O at all!
  - Lots of compute power available
  - Co-processing exploits high bandwidth, low latency HPC interconnect
- Challenges:
  - Memory footprint constrained
  - In situ memory sharing between simulation and analysis is tricky
  - Once the simulation has advanced, you cannot go back and analyze it
  - Temporal analysis is difficult

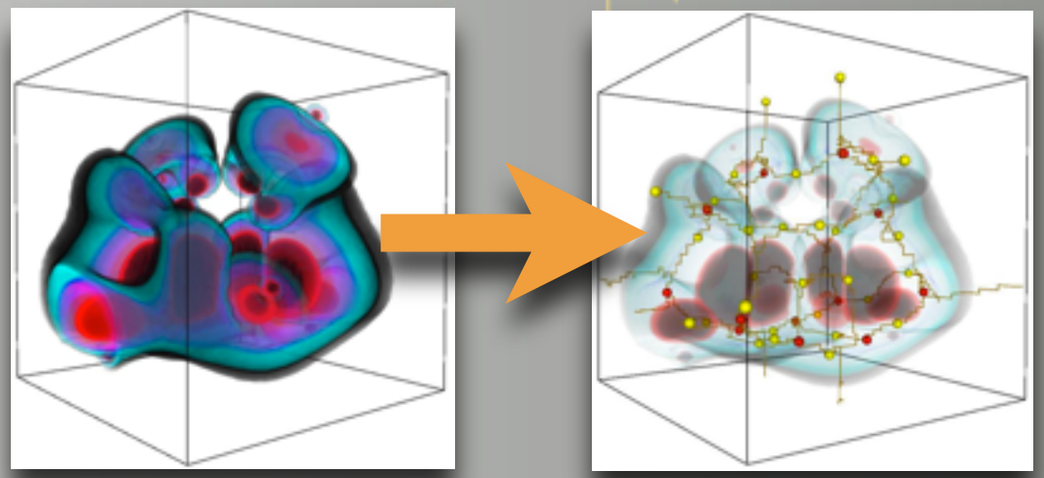
Summary techniques

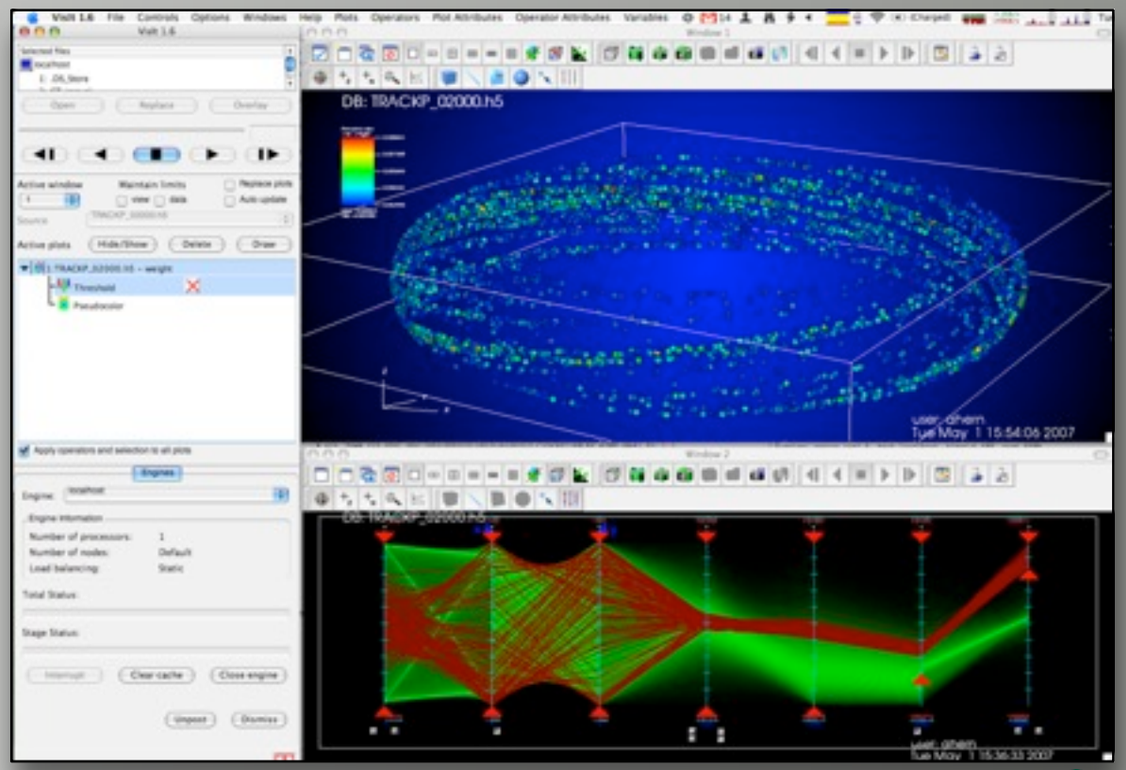
- Examples:
  - Reeb graphs and Contour trees
  - Parallel Coordinates
- Pros:
  - Compliments “SciVis” with “InfoVis” for guided visualization
  - Lots of success with topological techniques
  - Can dramatically reduce interactive I/O
- Challenges:
  - Can be very domain-specific



Summary techniques

- Examples:
  - Reeb graphs and Contour trees
  - Parallel Coordinates
- Pros:
  - Compliments “SciVis” with “InfoVis” for guided visualization
  - Lots of success with topological techniques
  - Can dramatically reduce interactive I/O
- Challenges:
  - Can be very domain-specific





Summary

- We've had a lot of success with both general and domain-specific visualization techniques.
- General data flow networks (both serial and parallel) have taken us a long way toward widely applicable purpose visualization tools.
- Data storage, data models, and I/O patterns are *critical* to high performance and scientific understanding.
- The exascale will bring disruptive changes the likes of which we haven't seen since the advent of MPI.