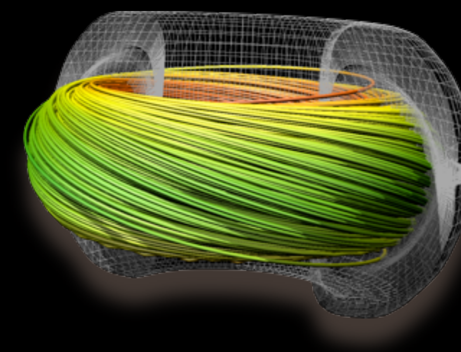
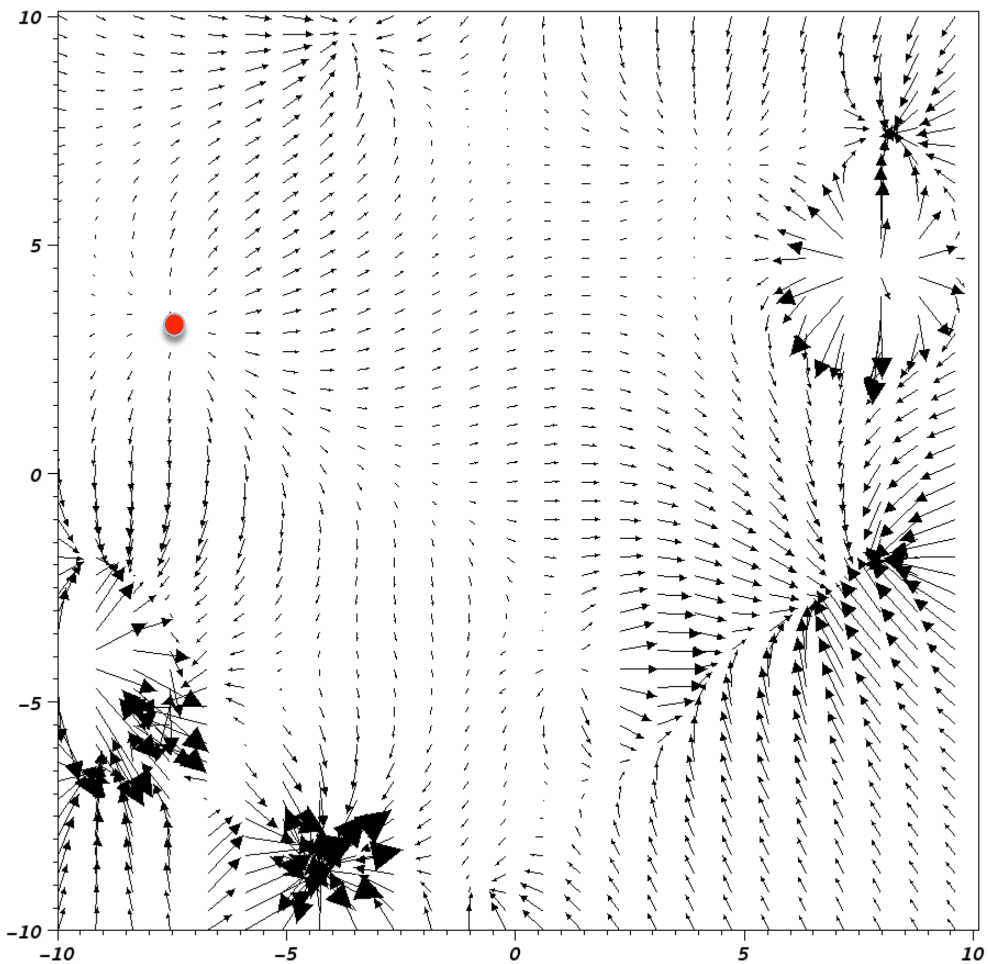
A Framework for Particle Advection for Very Large Data



Hank Childs, LBNL/UCDavis
David Pugmire, ORNL
Christoph Garth, Kaiserslautern
David Camp, LBNL/UCDavis
Sean Ahern, ORNL
Gunther Weber, LBNL
Allen Sanderson, Univ. of Utah

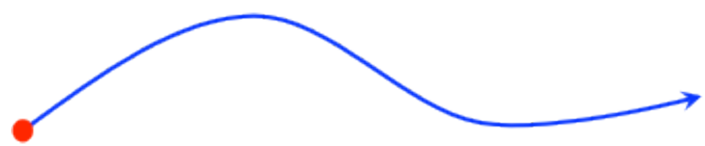
Advecting particles



Particle advection basics

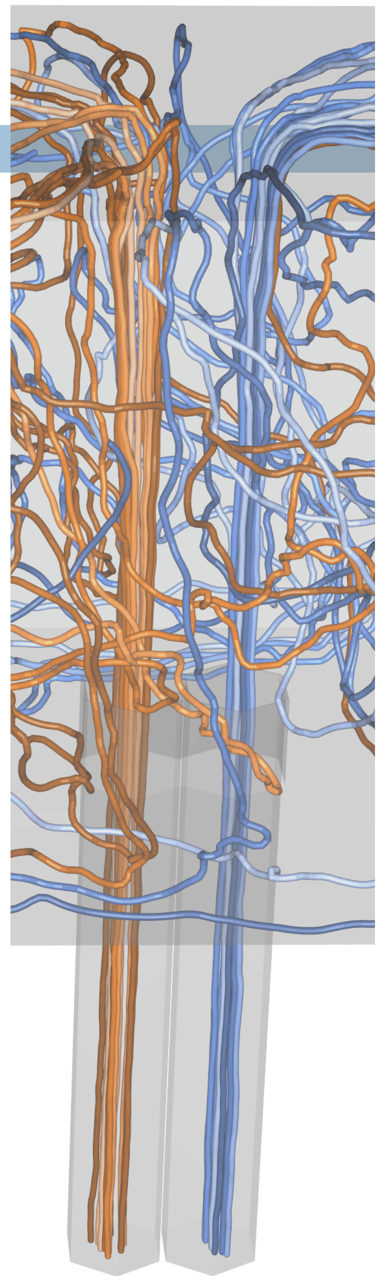
Advecting particles create integral curves

$$S'(t) = v(t, S(t)) \quad S(t_0) := x_0$$

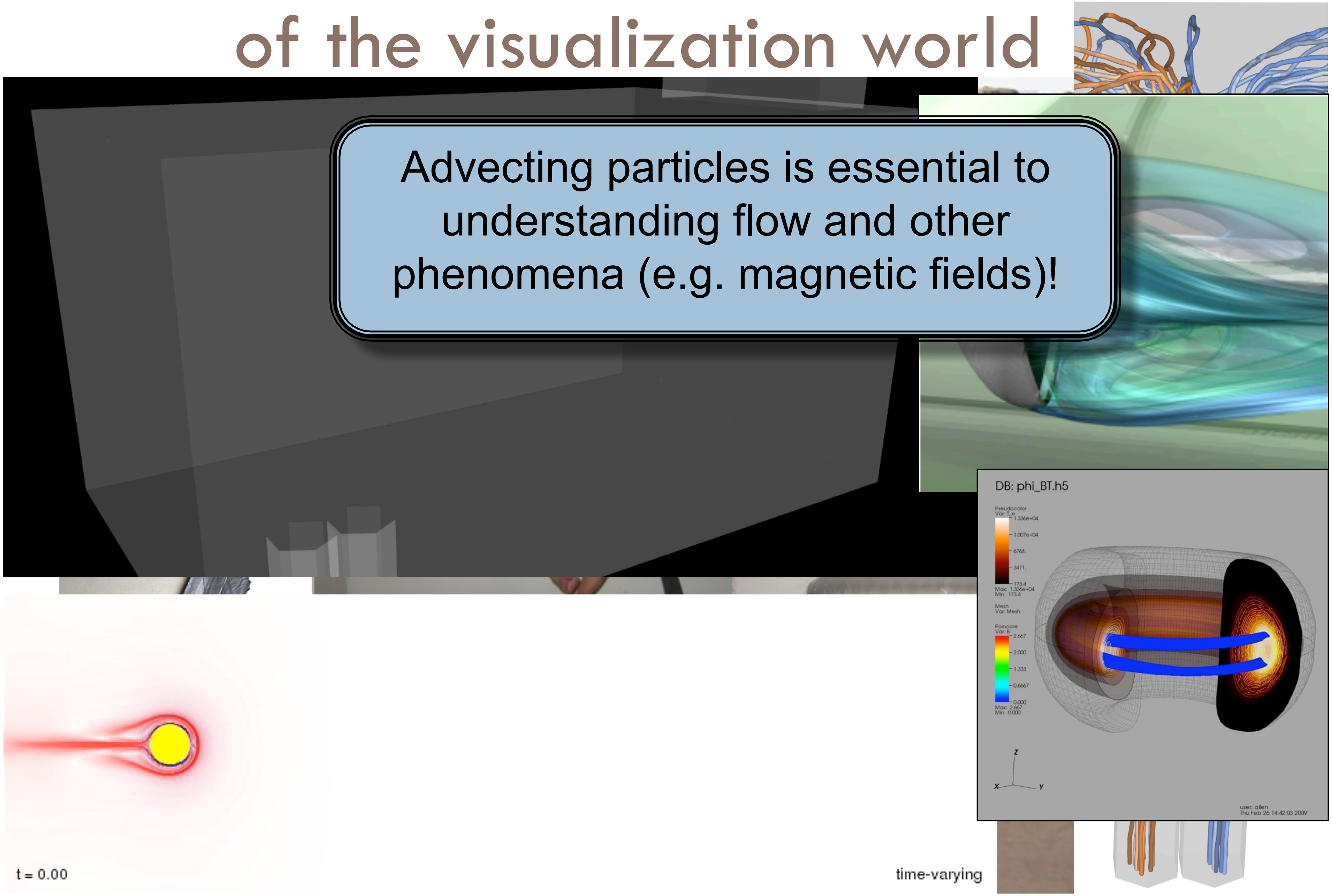


Streamlines: display particle path (instantaneous velocities)

Pathlines: display particle path (velocity field evolves as particle moves)



Particle advection is the duct tape of the visualization world



Advecting particles is essential to understanding flow and other phenomena (e.g. magnetic fields)!

Outline

- Efficient advection of particles
- A general system for particle-advection based analysis

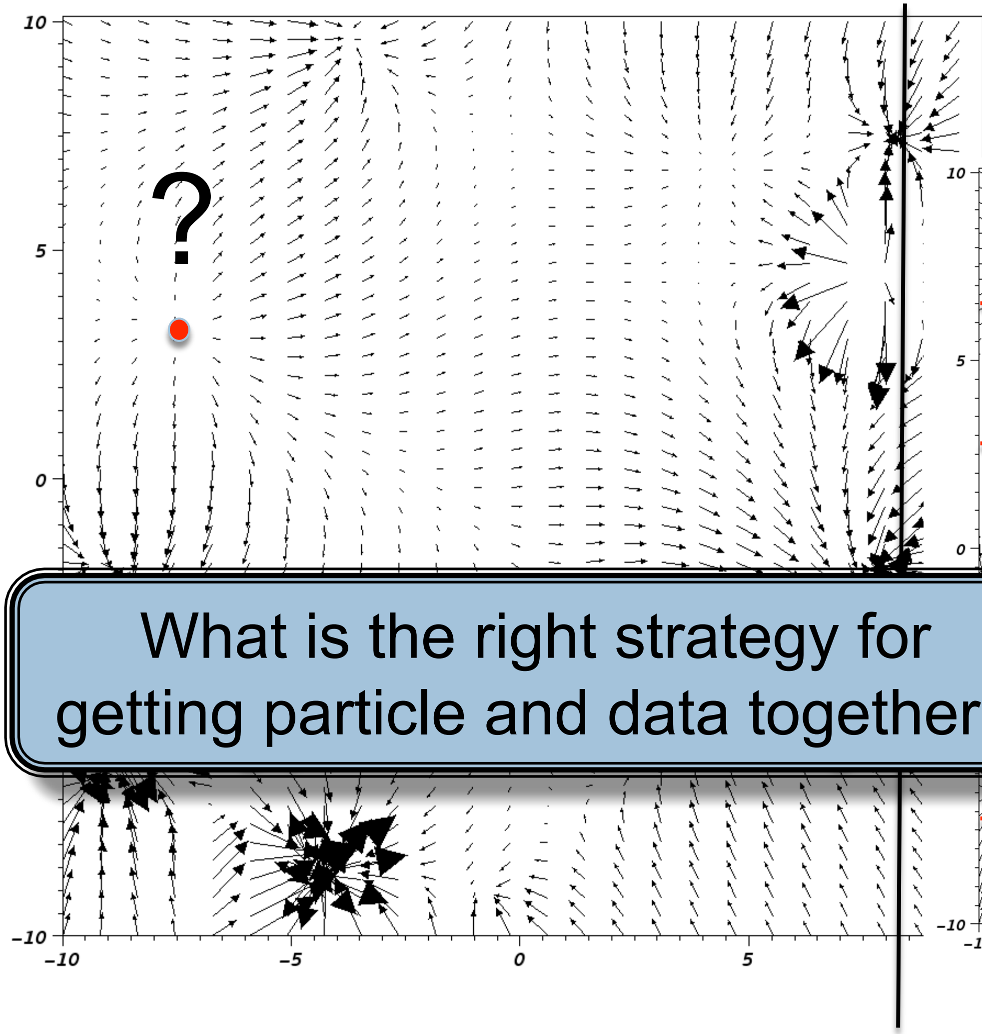
Particle Advection Load Balancing

- N particles (P1, P2, … Pn), M MPI tasks (T1, …, Tm)
- Each particle takes a variable number of steps, S1, S2, … Sn
- Total number of steps is Σ Si
  - We cannot do less work than this (Σ Si)
- Goal: Distribute the Σ Si steps over M MPI tasks such that problem finishes in minimal time

Particle Advection Performance

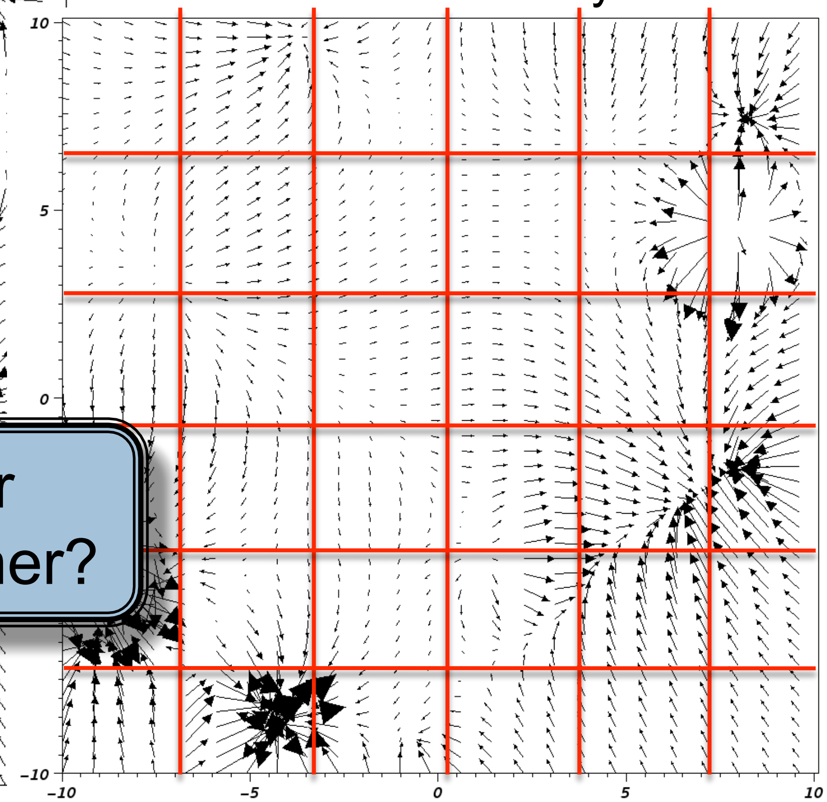
- Goal: Distribute the Σ Si steps over M MPI tasks such that problem finishes in minimal time
- Sounds sort of like a bin-packing problem, but…
  - particles can move from MPI task to MPI task
  - path of particle is data dependent and unknown a priori (we don't know Si beforehand)
  - big data significantly complicates this picture….
    - … data may not be readily available, introducing starvation

Advecting particles



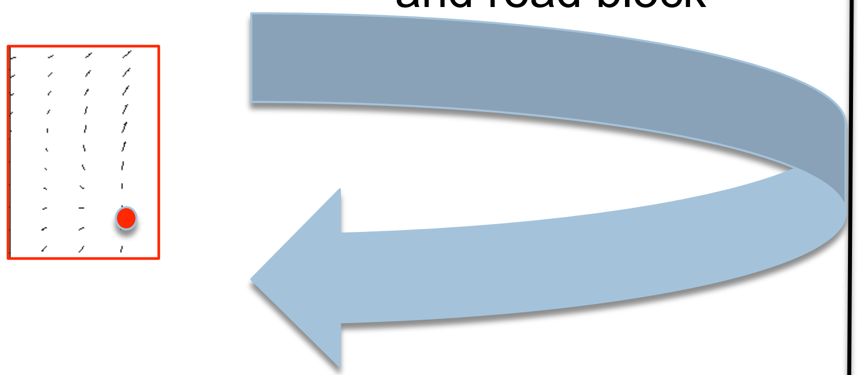
What is the right strategy for getting particle and data together?

Decomposition of large data set into blocks on filesystem

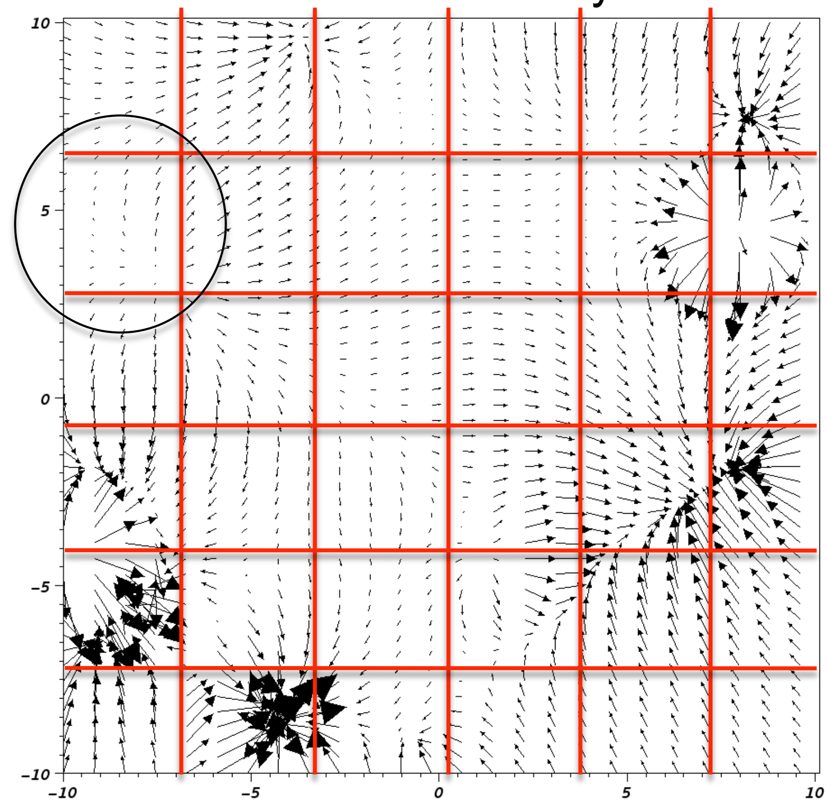


Strategy: load blocks necessary for advection

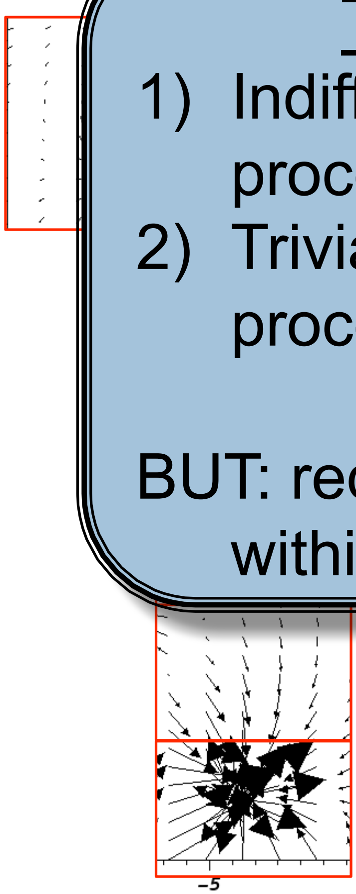
Go to filesystem
and read block

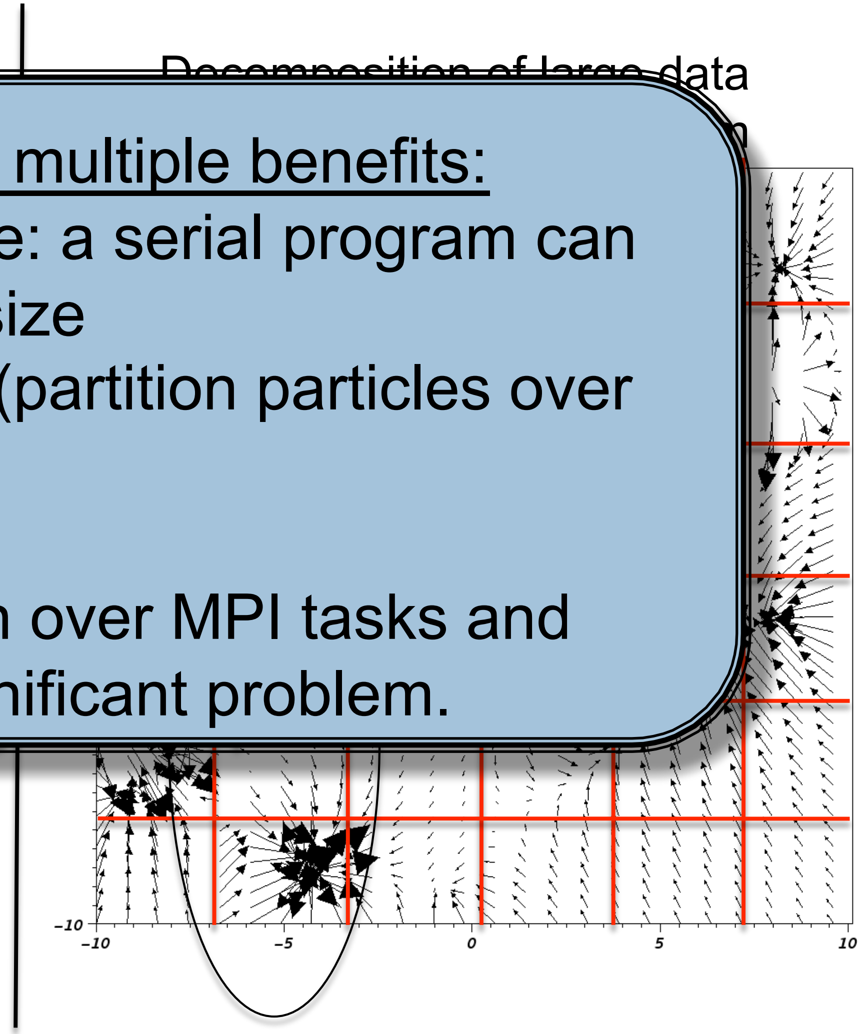


Decomposition of large data
set into blocks on filesystem



Strategy: load blocks necessary for advection





<u>This strategy has multiple benefits:</u>

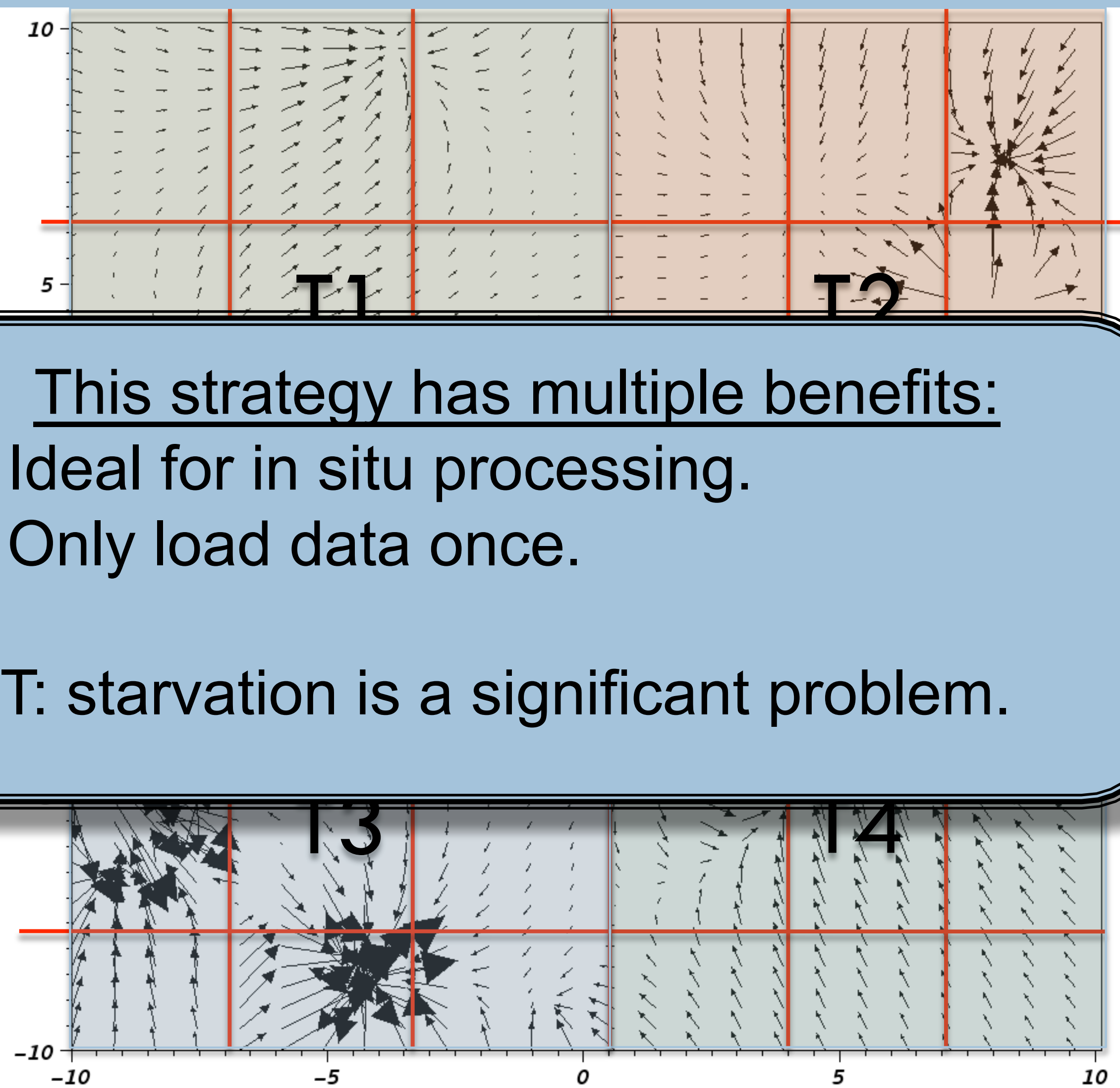
1) Indifferent to data size: a serial program can process data of any size
2) Trivial parallelization (partition particles over processors)

BUT: redundant I/O (both over MPI tasks and within a task) is a significant problem.

“Parallelize over Particles”

- “Parallelize over Particles”: particles are partitioned over processors, blocks of data are loaded as needed.
- Some additional complexities:
  - Work for a given particle (i.e. Si) is variable and not known a priori: how to share load between processors dynamically?
  - More blocks than can be stored in memory: what is the best caching/purging strategy?

"Parallelize over data" strategy: parallelize over blocks and pass particles



This strategy has multiple benefits:

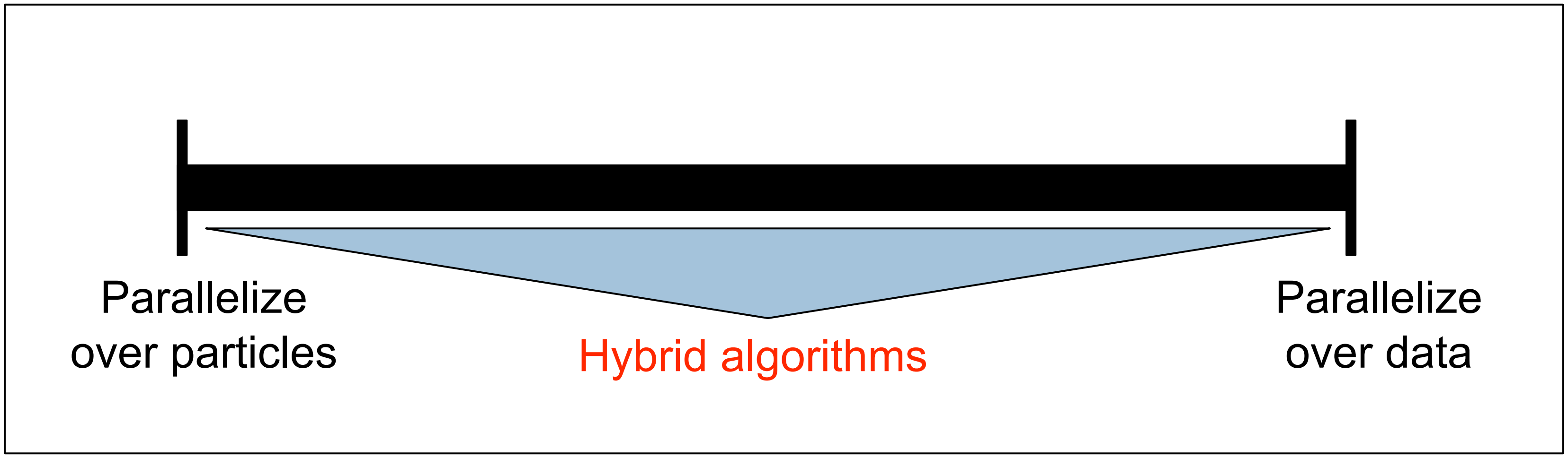
1) Ideal for in situ processing.
2) Only load data once.

BUT: starvation is a significant problem.

Both parallelization schemes have serious flaws.

- Two approaches:

Parallelizing Over	I/O	Efficiency
Data	Good	Bad
Particles	Bad	Good



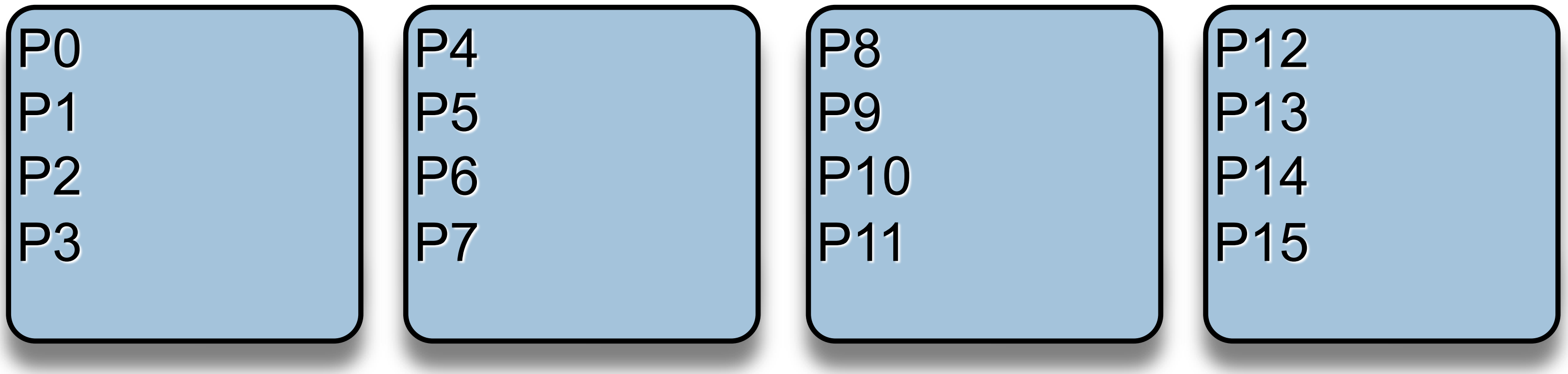
The master-slave algorithm is an example of a hybrid technique.

- Algorithm adapts during runtime to avoid pitfalls of parallelize-over-data and parallelize-over-particles.
  - Nice property for production visualization tools.
- Implemented inside VisIt visualization and analysis package.

D. Pugmire, H. Childs, C. Garth, S. Ahern, G. Weber, “Scalable Computation of Streamlines on Very Large Datasets.” SC09, Portland, OR, November, 2009

Master-Slave Hybrid Algorithm

- Divide processors into groups of **N**
- Uniformly distribute seed points to each group



Master:
- Monitor workload
- Make decisions to optimize resource utilization

Slaves:
- Respond to commands from Master
- Report status when work complete

Master Process Pseudocode

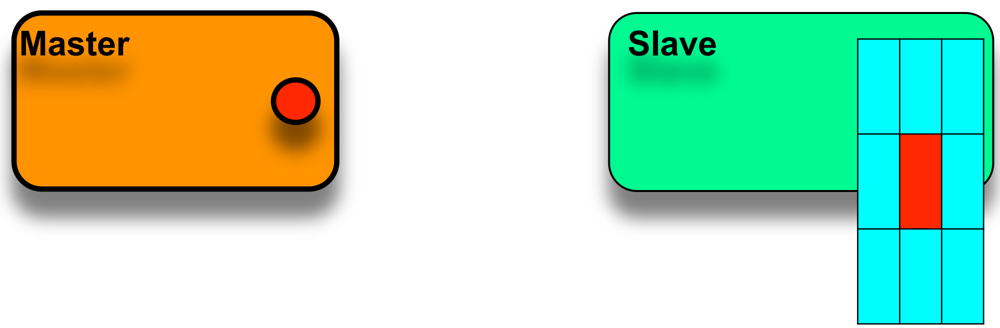
```
Master()
{
    while ( ! done )
    {
        if ( NewStatusFromAnySlave
        {
            commands = DetermineMostEfficientCommand()

            for cmd in commands
                SendCommandToSlaves( cmd )
        }
    }
}
```

What are the possible commands?

Commands that can be issued by master

1. **Assign / Loaded Block**
2. Assign / Unloaded Block
3. Handle OOB / Load
4. Handle OOB / Send

   OOB = out of bounds

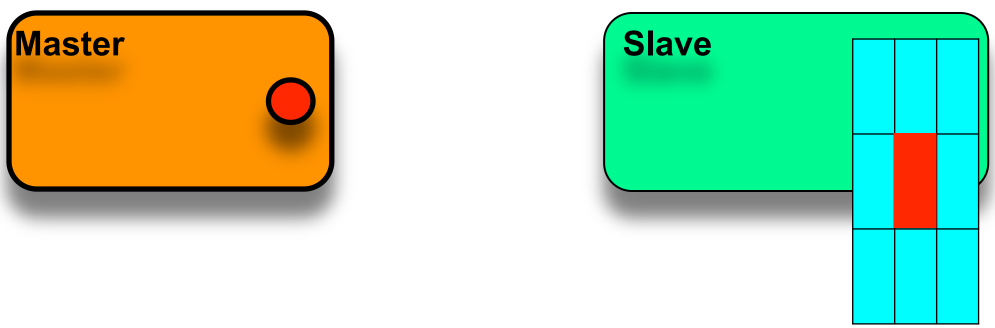


Slave is given a streamline that is contained in a block that is already loaded

Commands that can be issued by master

1. Assign / Loaded Block
2. **Assign / Unloaded Block**
3. Handle OOB / Load
4. Handle OOB / Send

   OOB = out of bounds

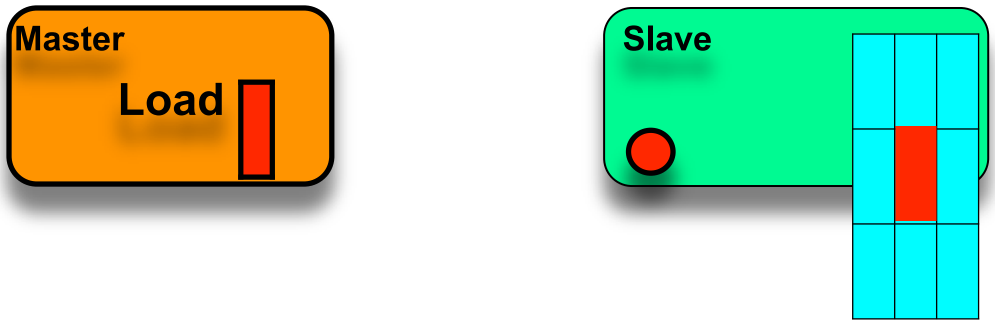


Slave is given a streamline
and loads the block

Commands that can be issued by master

1. Assign / Loaded Block
2. Assign / Unloaded Block
3. **Handle OOB / Load**
4. Handle OOB / Send

   OOB = out of bounds

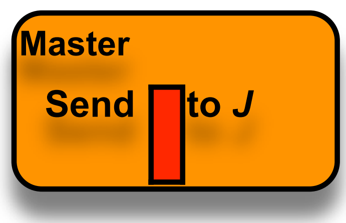


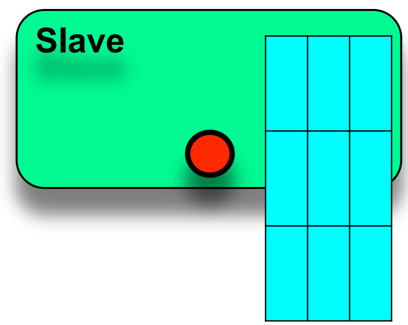
Slave is instructed to load a block. The streamline in that block can then be computed.

Commands that can be issued by master

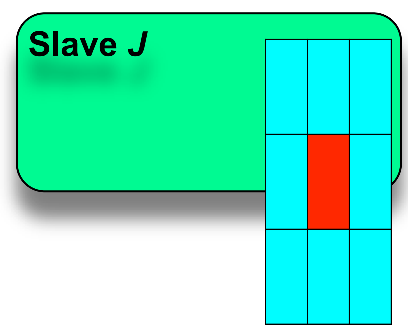
1. Assign / Loaded Block
2. Assign / Unloaded Block
3. Handle OOB / Load
4. **Handle OOB / Send**

   OOB = out of bounds





Slave is instructed to send a streamline to another slave that has loaded the block



Master Process Pseudocode

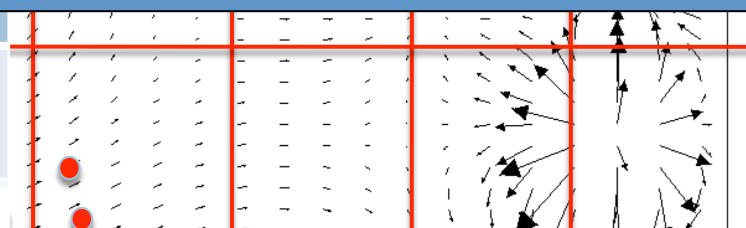
```
Master()
{
    while ( ! done )
    {
        if ( NewStatusFromAnySlave() )
        {
            commands = DetermineMostEfficientCommand()

            for cmd in commands
                SendCommandToSlaves( cmd )
        }
    }
}
```

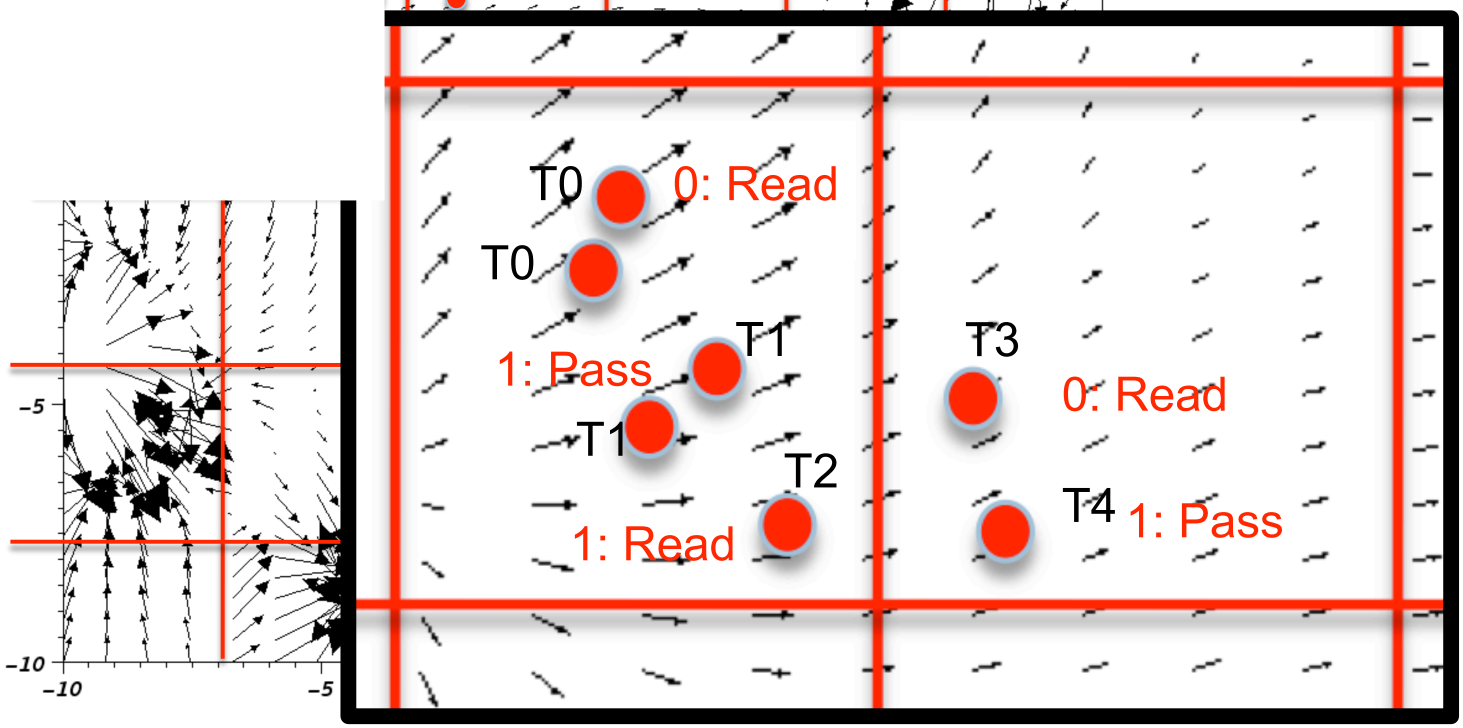
*** See SC 09 paper for details**

Master-s

Iteration	Action
0	T0 reads B0, T3 reads B1

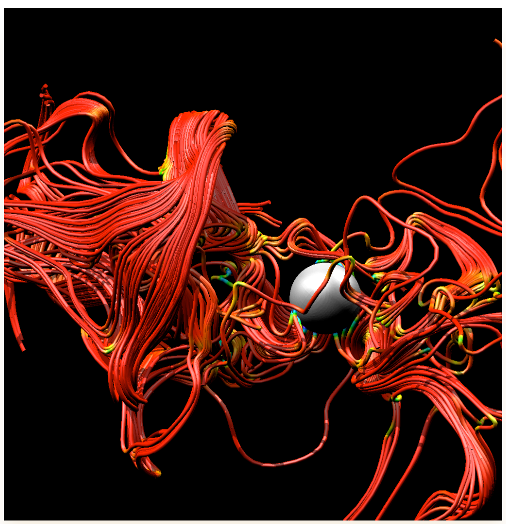
- When to pass and when to read?
- How to coordinate communication? Status? Efficiently?

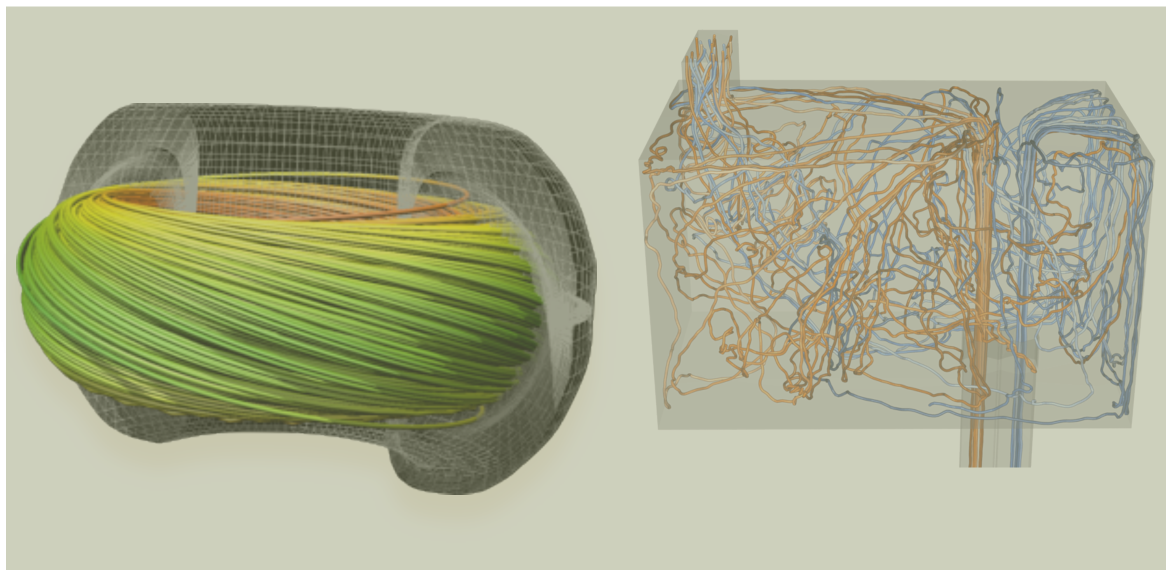




Algorithm Test Cases

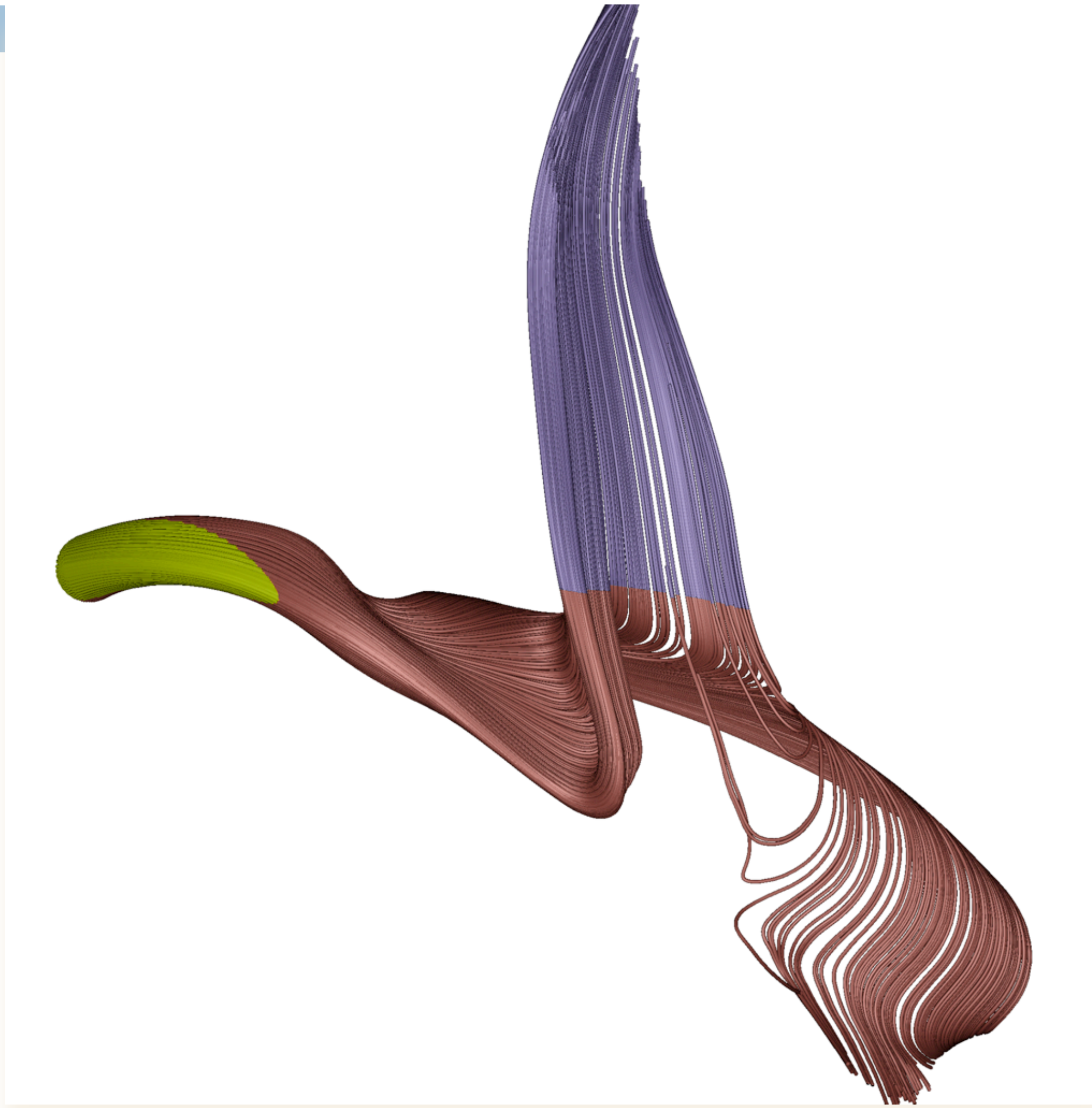
- Core collapse supernova simulation
- Magnetic confinement fusion simulation
- Hydraulic flow simulation





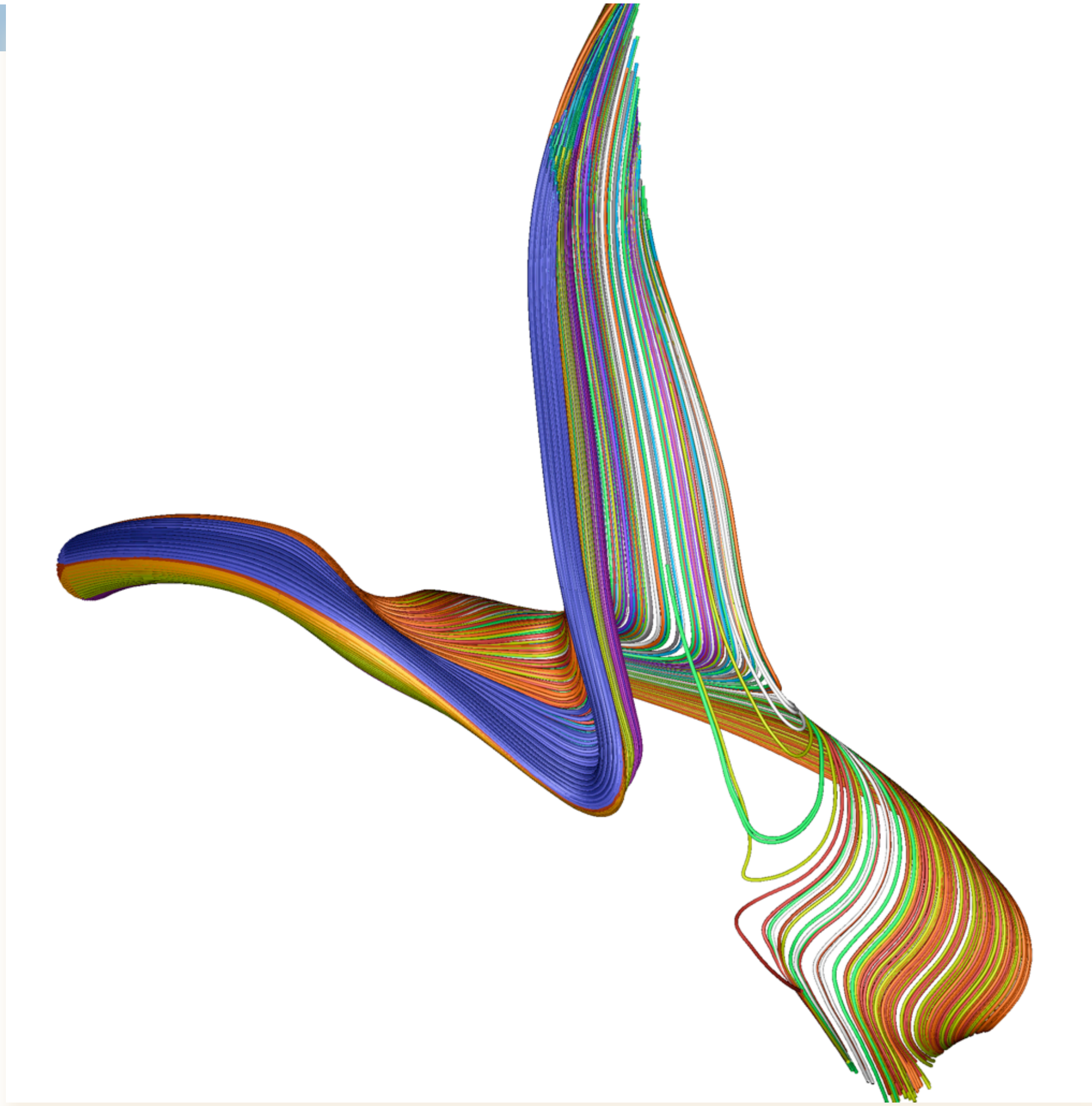
Workload distribution in parallelize-over-data

Starvation



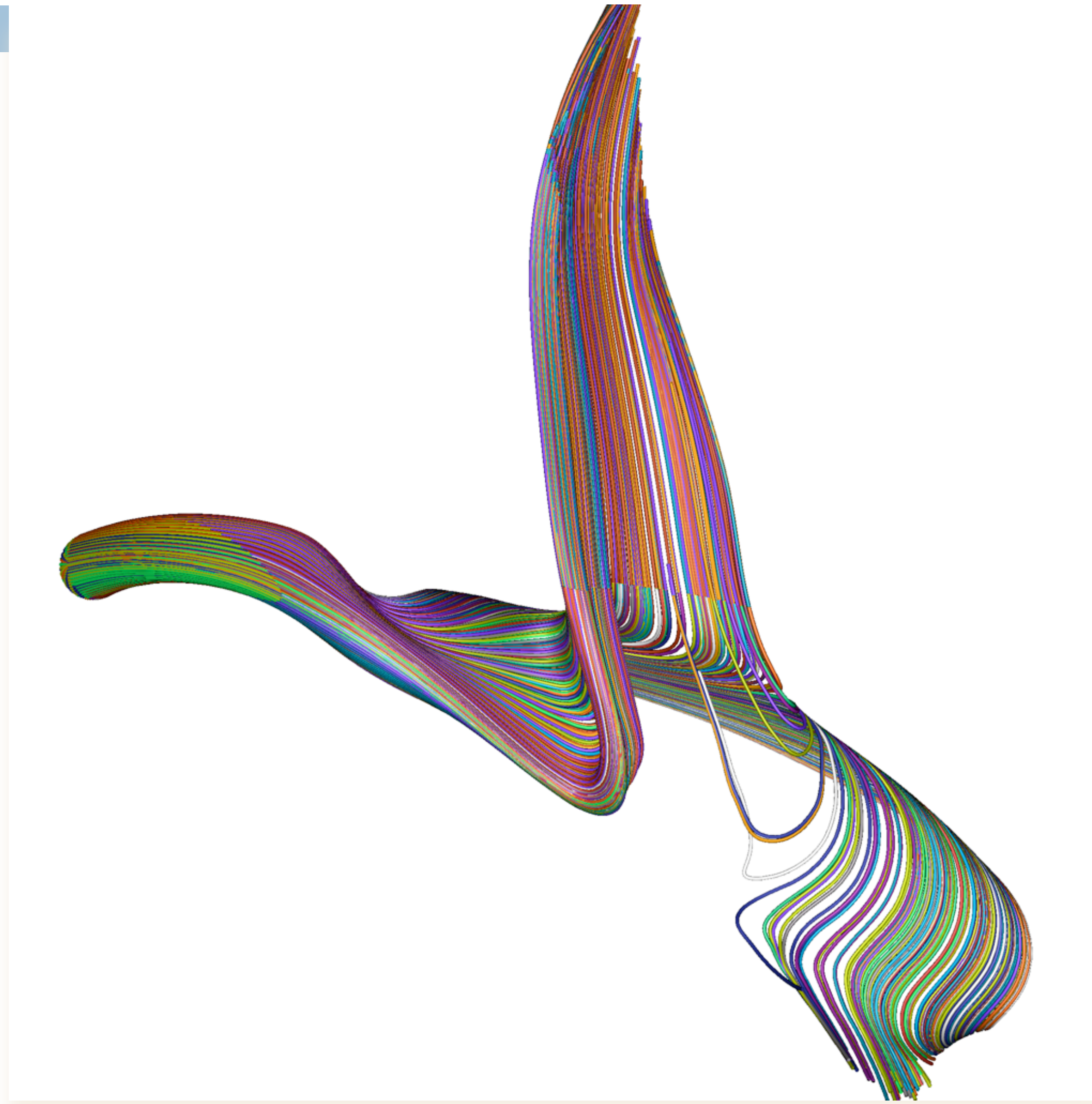
Workload distribution in parallelize-over-particles

Too much I/O



Workload distribution in master-slave algorithm

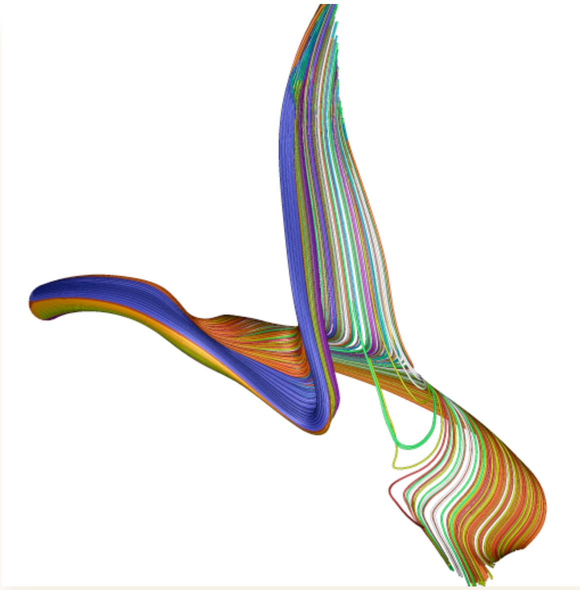
Just right



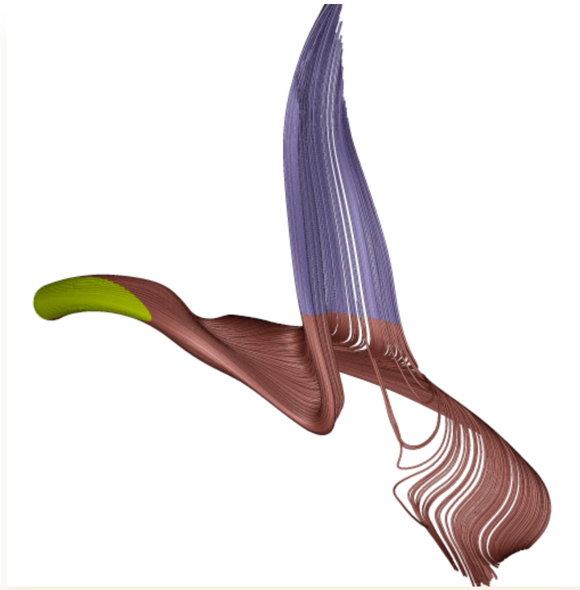
Workload distribution in supernova simulation

Parallelization by:

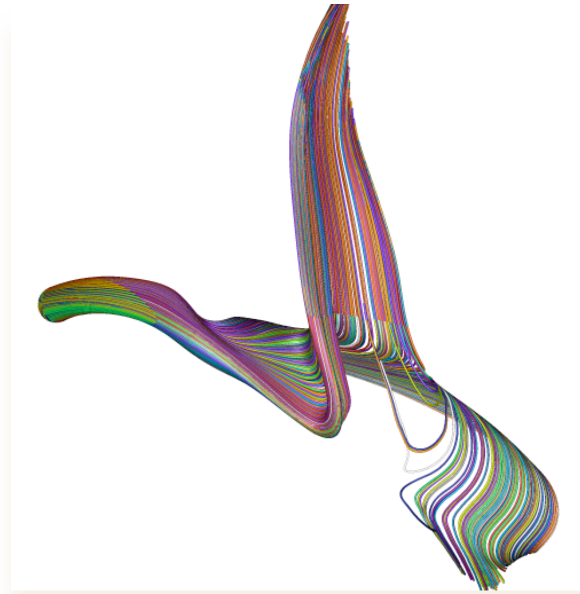
Particles



Data

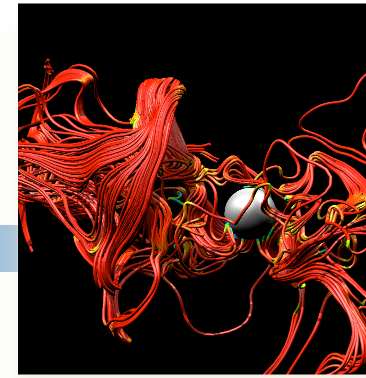


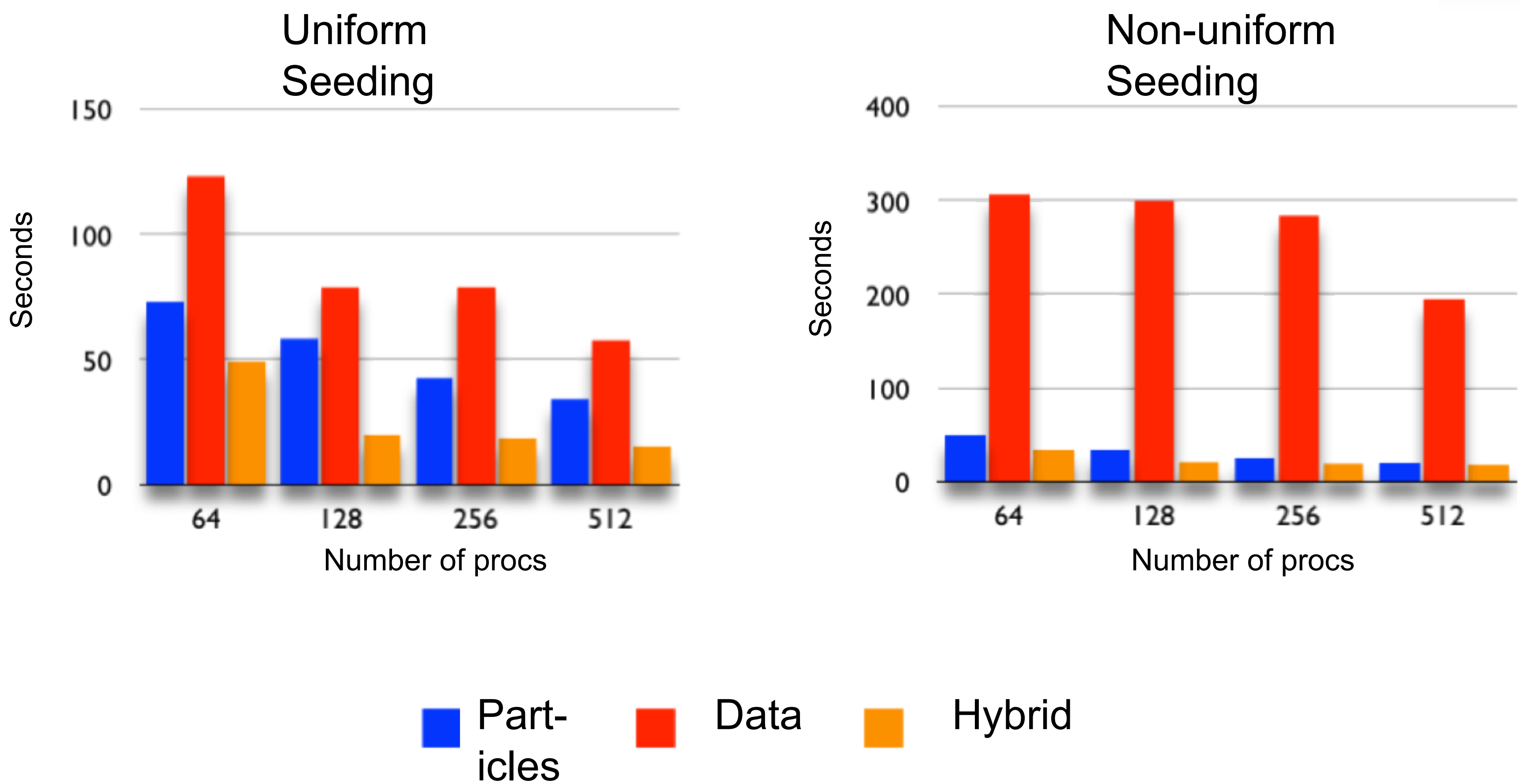
Hybrid



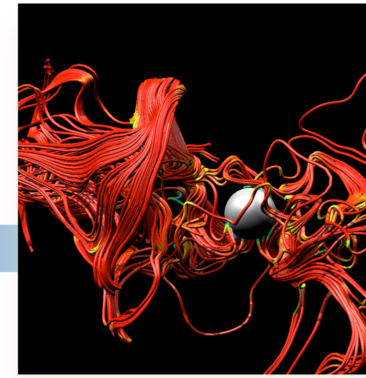
Colored by processor doing integration

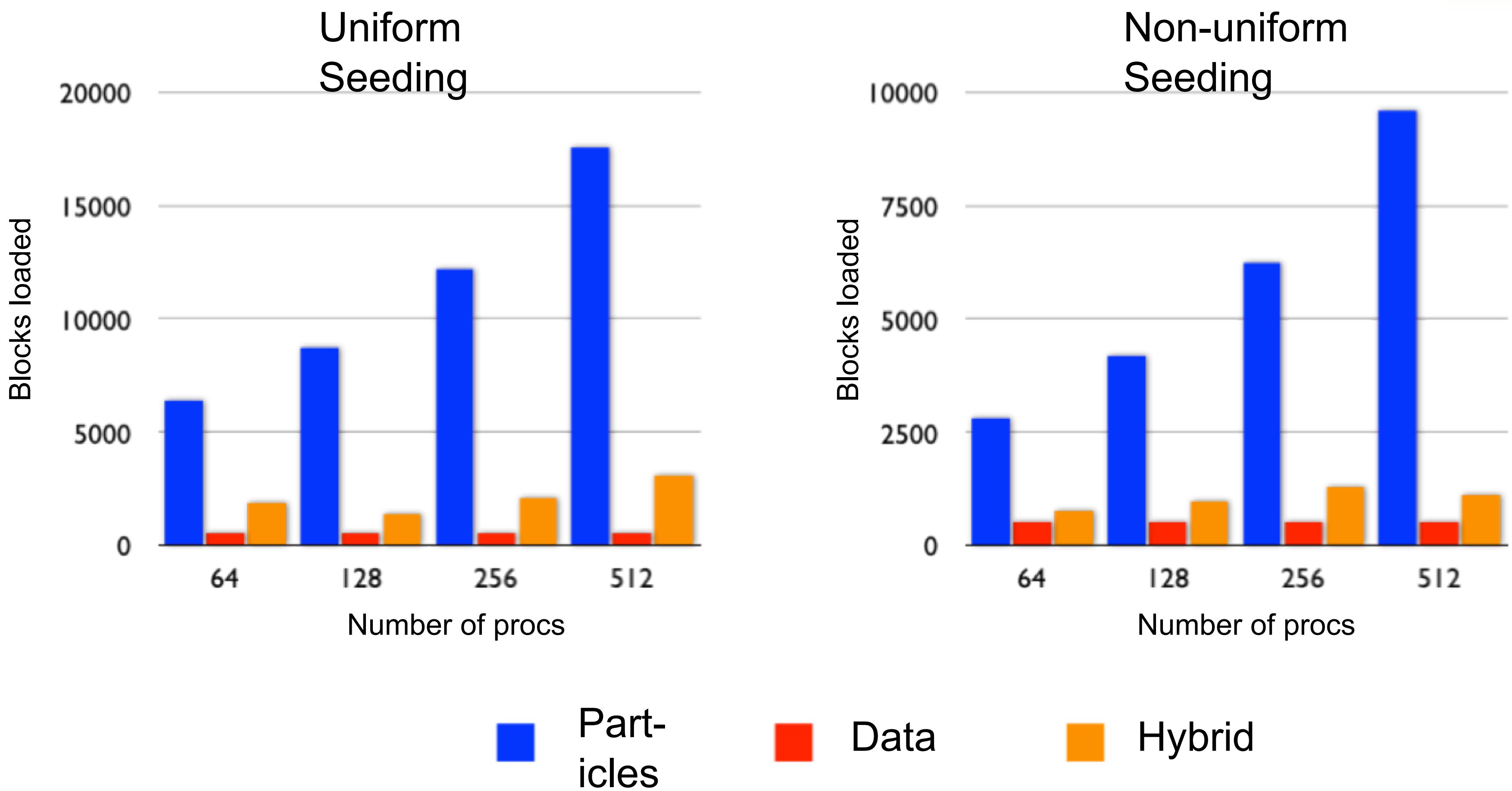
Astrophysics Test Case:
Total time to compute 20,000 Streamlines





Astrophysics Test Case:
Number of blocks loaded





Summary: Master-Slave Algorithm

- First ever attempt at a hybrid parallelization algorithm for particle advection
- Algorithm adapts during runtime to avoid pitfalls of parallelize-over-data and parallelize-over-particles.
  - Nice property for production visualization tools.
- Implemented inside VisIt visualization and analysis package.

Outline

- Efficient advection of particles
- A general system for particle-advection based analysis

Goal

- Efficient code for a variety of particle advection based techniques
- Cognizant of use cases with >>10K particles.
  - Need handling of every particle, every evaluation to be efficient.
- Want to support diverse flow techniques: flexibility/extensibility is key.
- Fit within data flow network design (i.e. a filter)

Motivating examples of system

- FTLE
- Stream surfaces
- Streamline
- Dynamical Systems (e.g. Poincaré Maps)
- Statistics based analysis
- + more

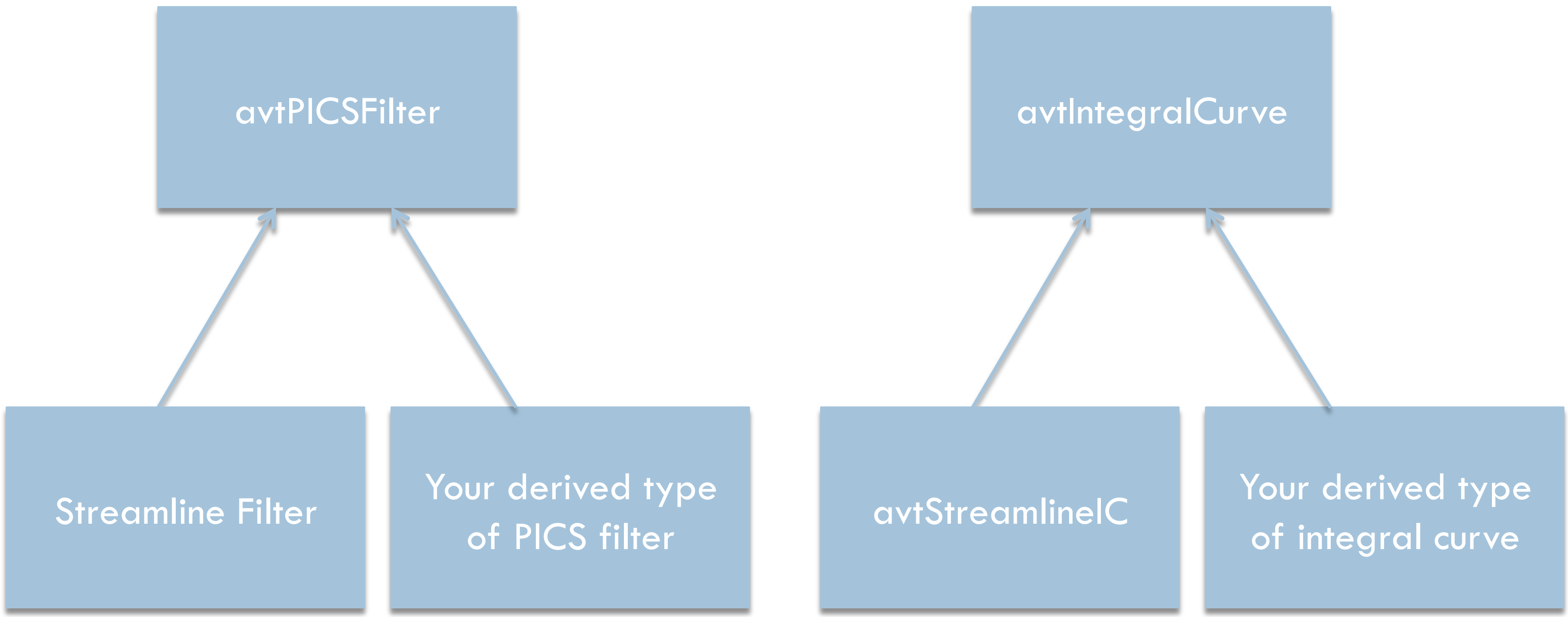
Design

- PICS filter: parallel integral curve system
- Execution:
  - Instantiate particles at seed locations
  - Step particles to form integral curves
    - Analysis performed at each step
    - Termination criteria evaluated for each step
  - When all integral curves have completed, create final output

Design

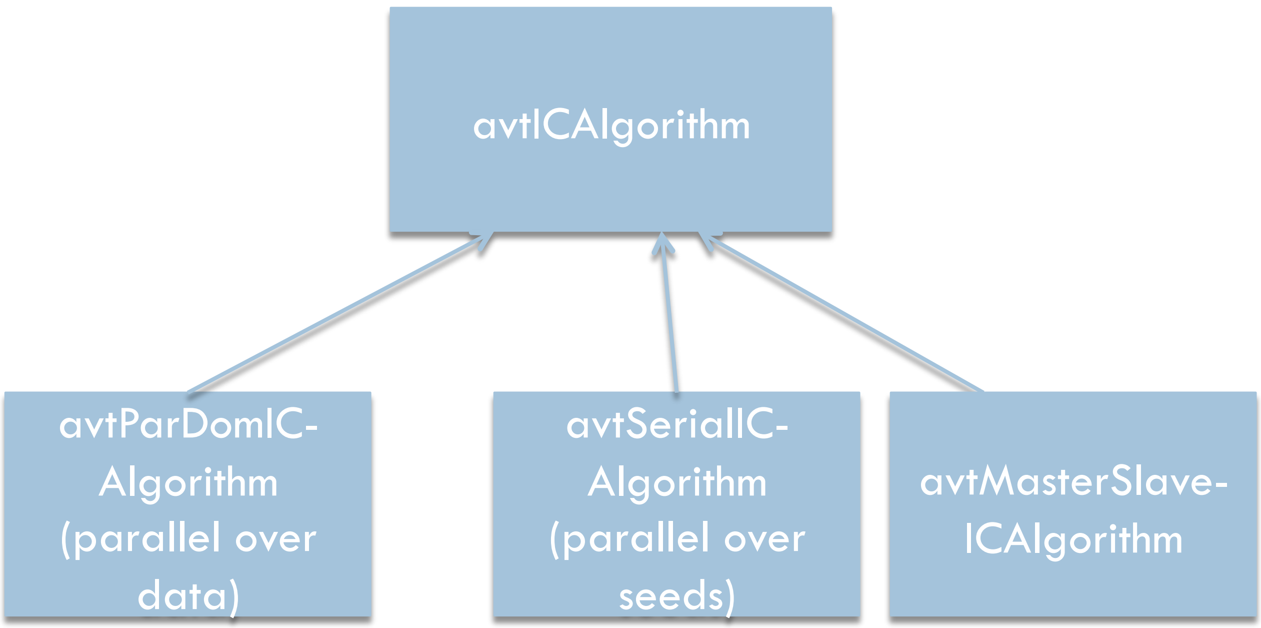
- Five major types of extensibility:
  - How to parallelize?
  - How do you evaluate velocity field?
  - How do you advect particles?
  - Initial particle locations?
  - How do you analyze the particle paths?

Inheritance hierarchy

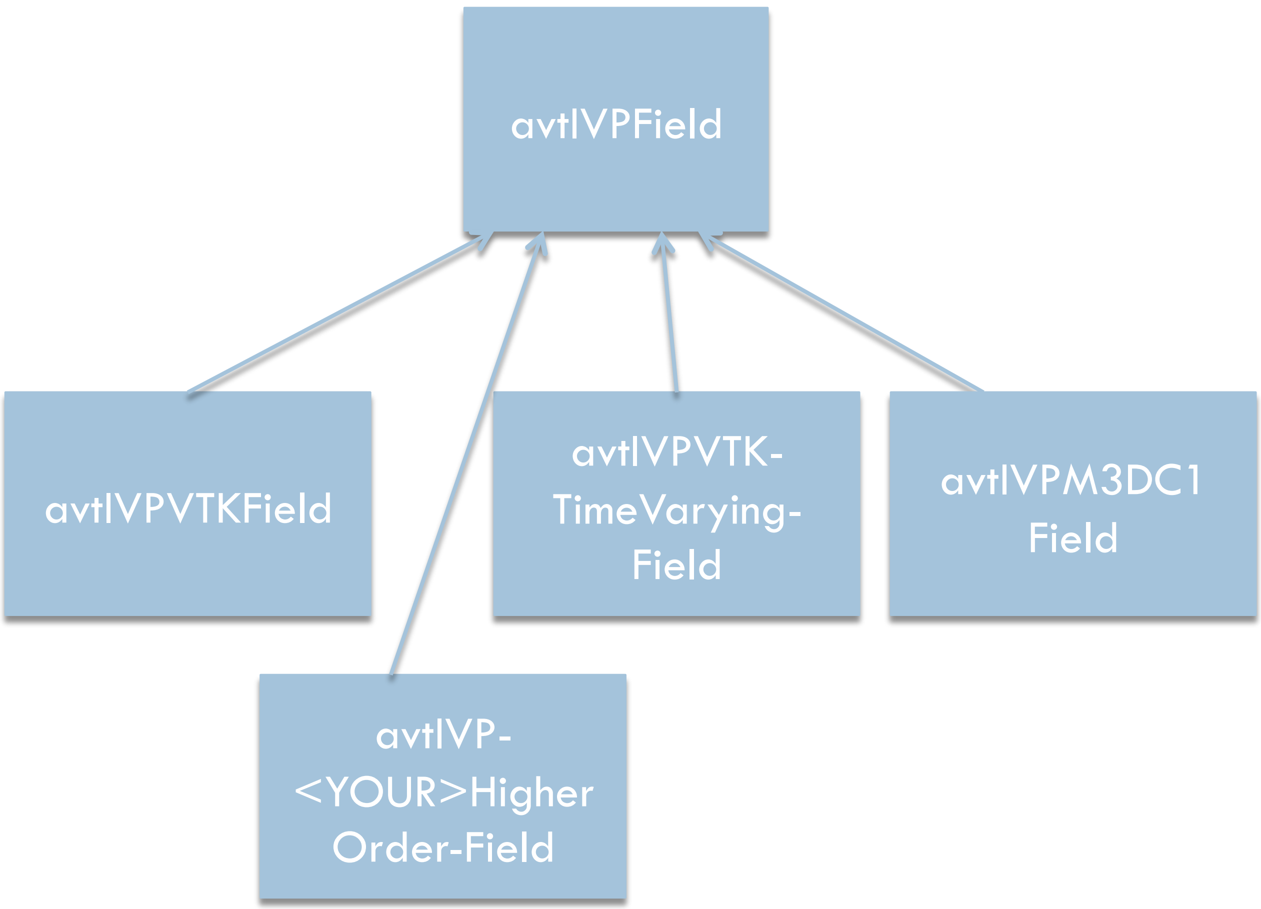


- We disliked the "matching inheritance" scheme, but this achieved all of our design goals cleanly.

#1: How to parallelize?

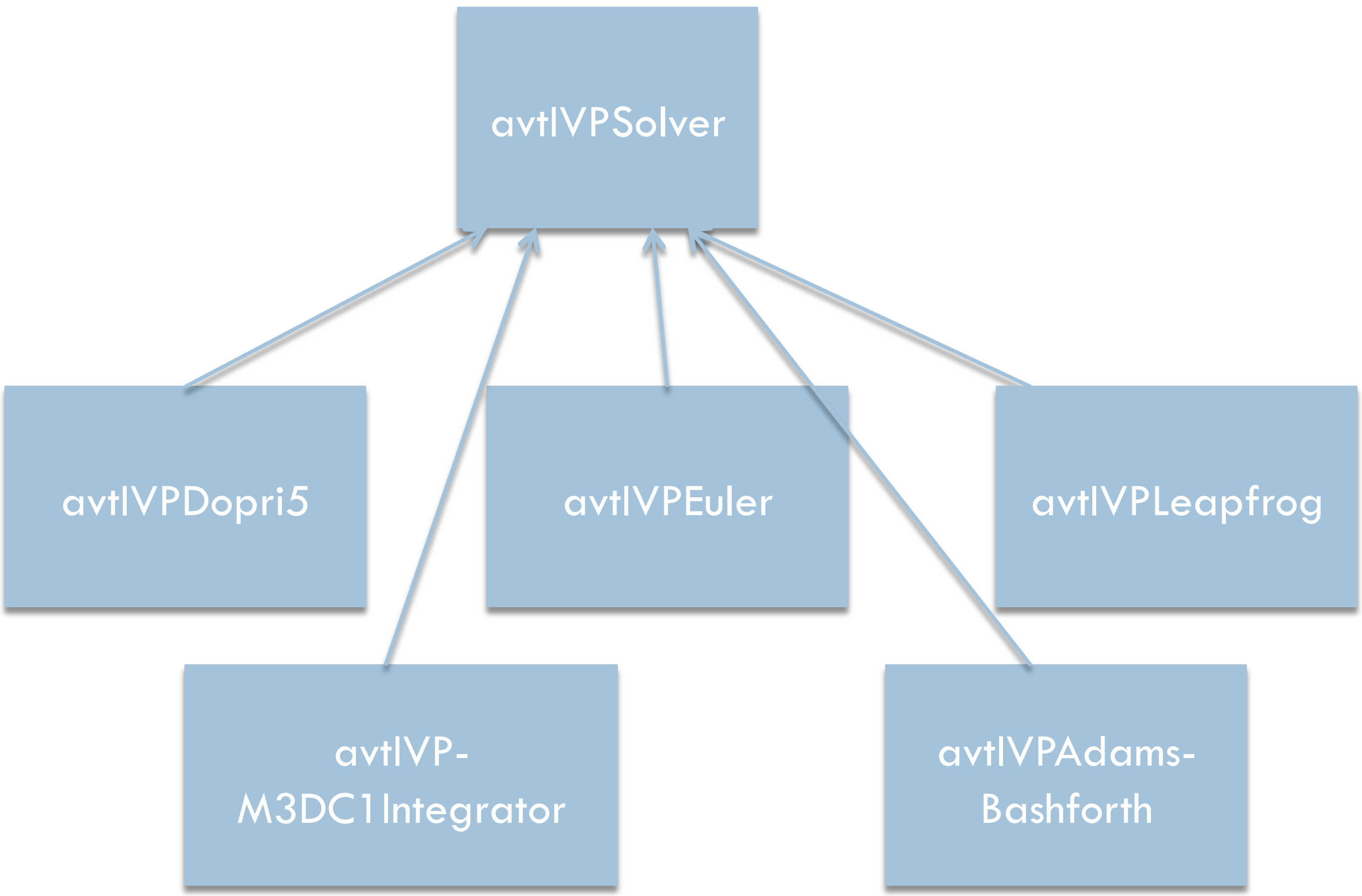


#2: Evaluating velocity field



IVP = initial value problem

#3: How do you advect particles?



IVP = initial value problem

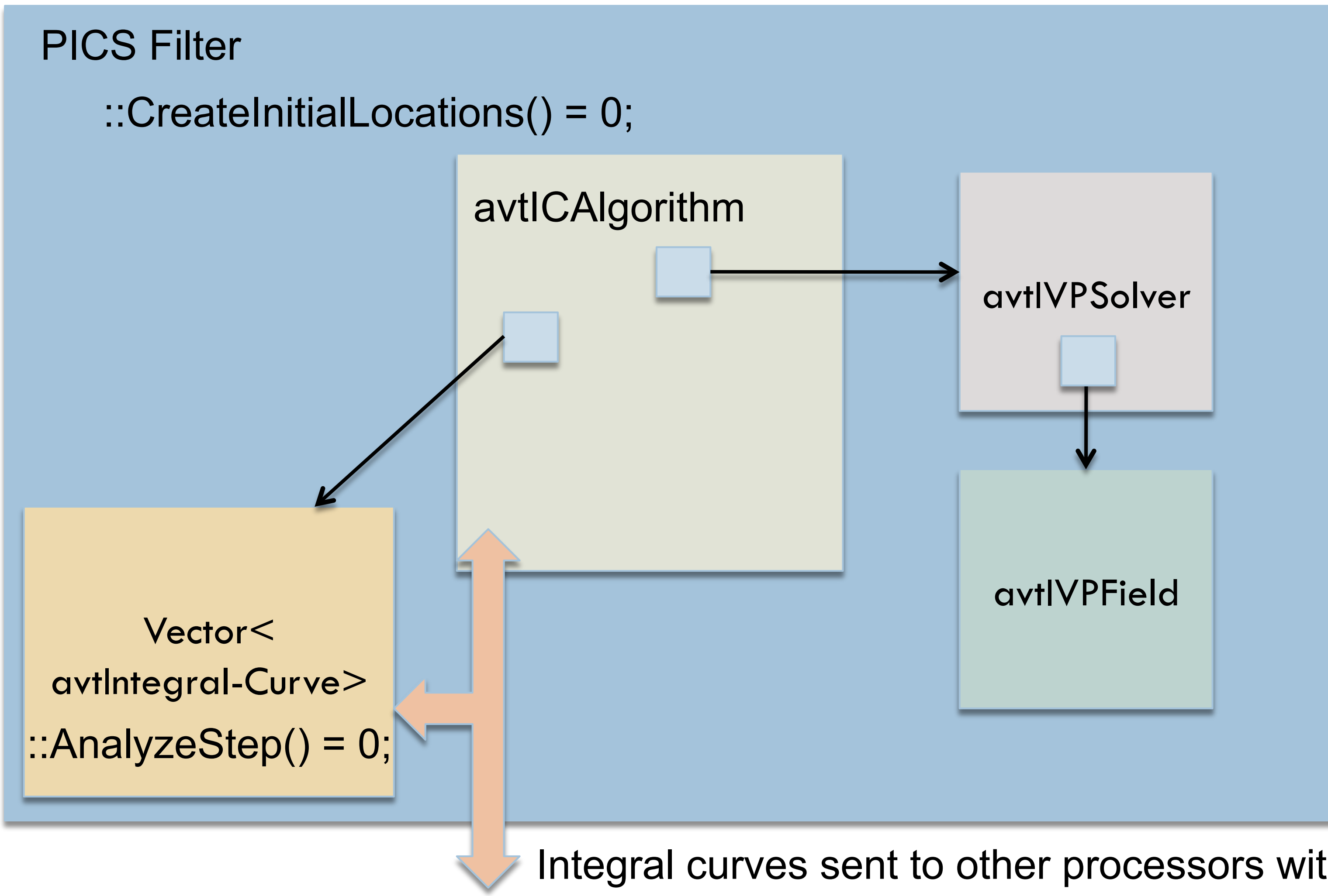
#4: Initial particle locations

- avtPICSFilter::GetInitialLocations() = 0;

#5: How do you analyze particle path?

- avtIntegralCurve::AnalyzeStep() = 0;
  - All AnalyzeStep will evaluate termination criteria
- avtPICSFilter::CreateIntegralCurveOutput(
  std::vector<avtIntegralCurve*> &) = 0;
- Examples:
  - Streamline: store location and scalars for current step in data members
  - Poincare: store location for current step in data members
  - FTLE: only store location of final step, no-op for preceding steps
- NOTE: these derived types create very different types of outputs.

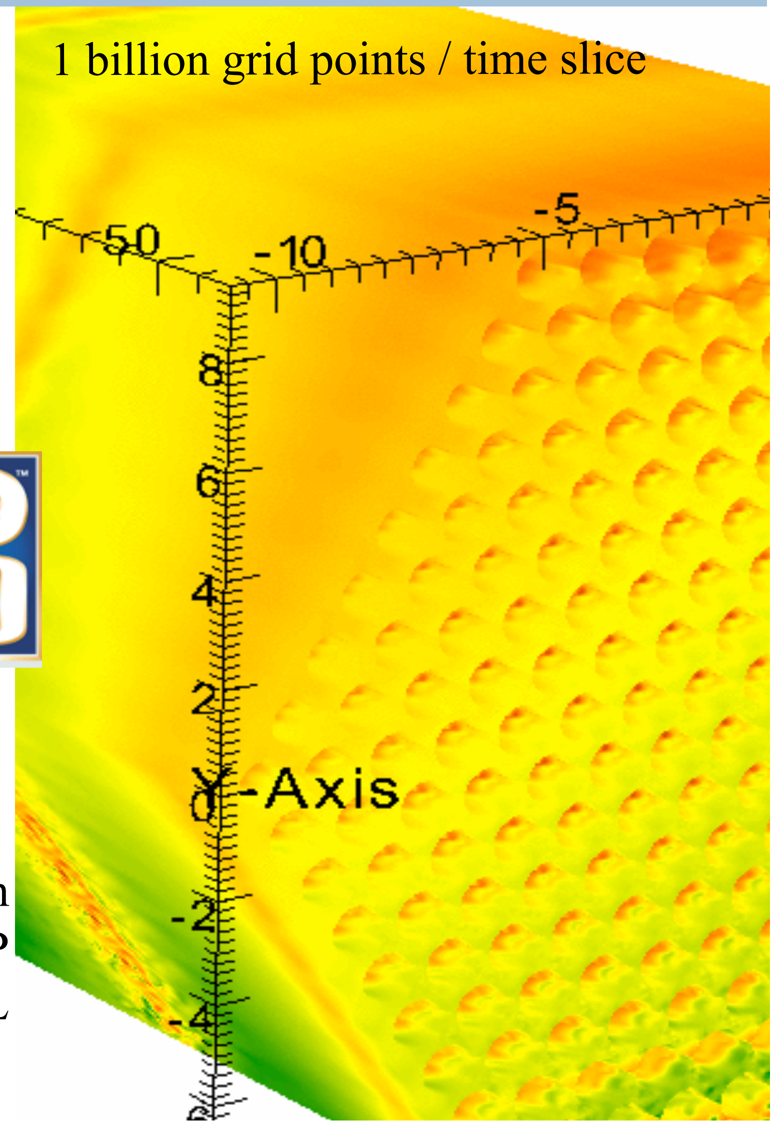
Putting it all together



Integral curves sent to other processors with some derived types of avtICAlgorithm.

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

- Used by:
  - Visualization experts
  - Simulation code developers
  - Simulation code consumers
- Popular
  - R&D 100 award in 2005
  - Used on many of the Top500
  - >>>100K downloads



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

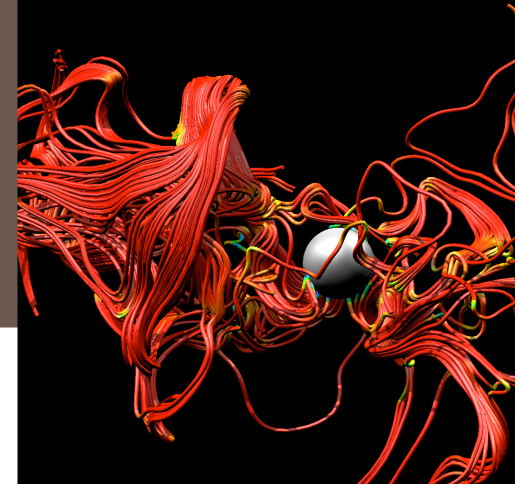
Final thoughts…

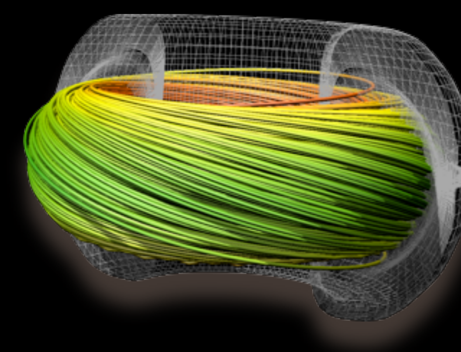
- Summary:
  - Particle advection is important for understanding flow and efficiently parallelizing this computation is difficult.
  - We have developed a freely available system for doing this analysis for large data.
- Documentation:
  - (PICS) http://www.visitusers.org/index.php?title=Pics_dev
  - (VisIt) http://www.llnl.gov/visit
- Future work:
  - UI extensions, including Python
  - Additional analysis techniques (FTLE & more)

Acknowledgements

- Funding: This work was supported by the Director, Office of Science, Office and Advanced Scientific Computing Research, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 through the Scientific Discovery through Advanced Computing (SciDAC) program's Visualization and Analytics Center for Enabling Technologies (VACET).
- Program Manager: Lucy Nowell
- Master-Slave Algorithm: Dave Pugmire (ORNL), Hank Childs (LBNL/UCD), Christoph Garth (Kaiserslautern), Sean Ahern (ORNL), and Gunther Weber (LBNL)
- PICS framework: Hank Childs (LBNL/UCD), Dave Pugmire (ORNL), Christoph Garth (Kaiserslautern), David Camp (LBNL/UCD), Allen Sanderson (Univ of Utah)

A Framework for Particle Advection for Very Large Data





Hank Childs, LBNL/UCDavis
David Pugmire, ORNL
Christoph Garth, Kaiserslautern
David Camp, LBNL/UCDavis
Sean Ahern, ORNL
Gunther Weber, LBNL
Allen Sanderson, Univ. of Utah