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

–Carl Friedrich Gauss, 1777-1855

DIY Parallel Data Analysis

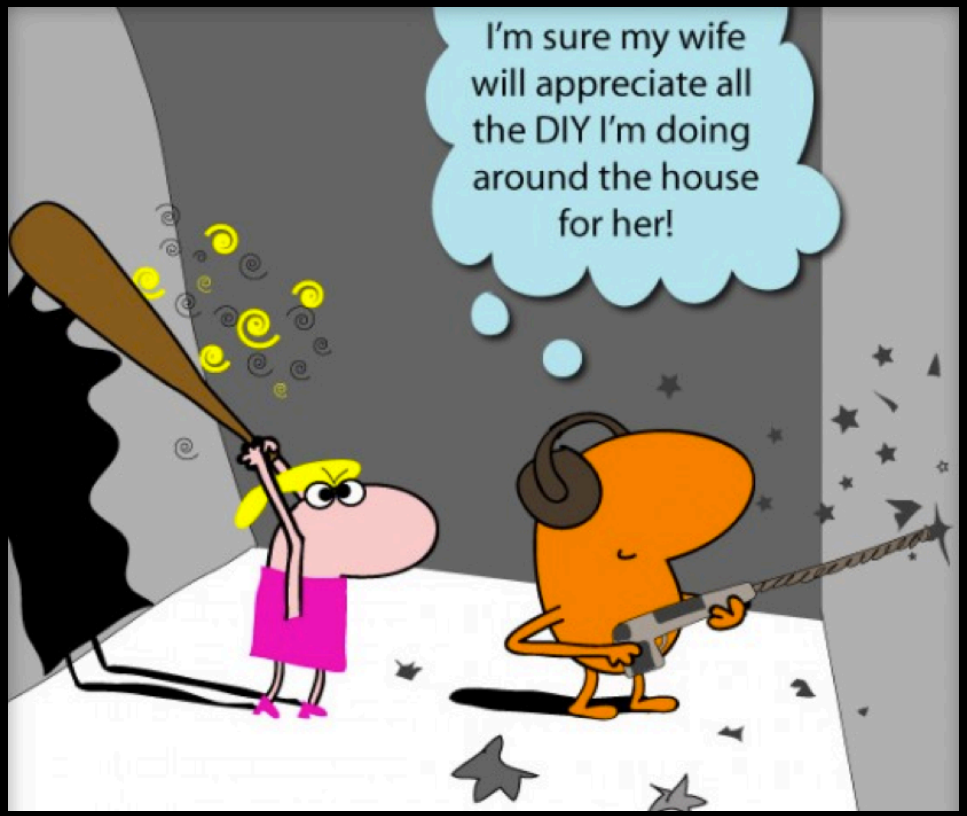


Image courtesy pigtimes.com

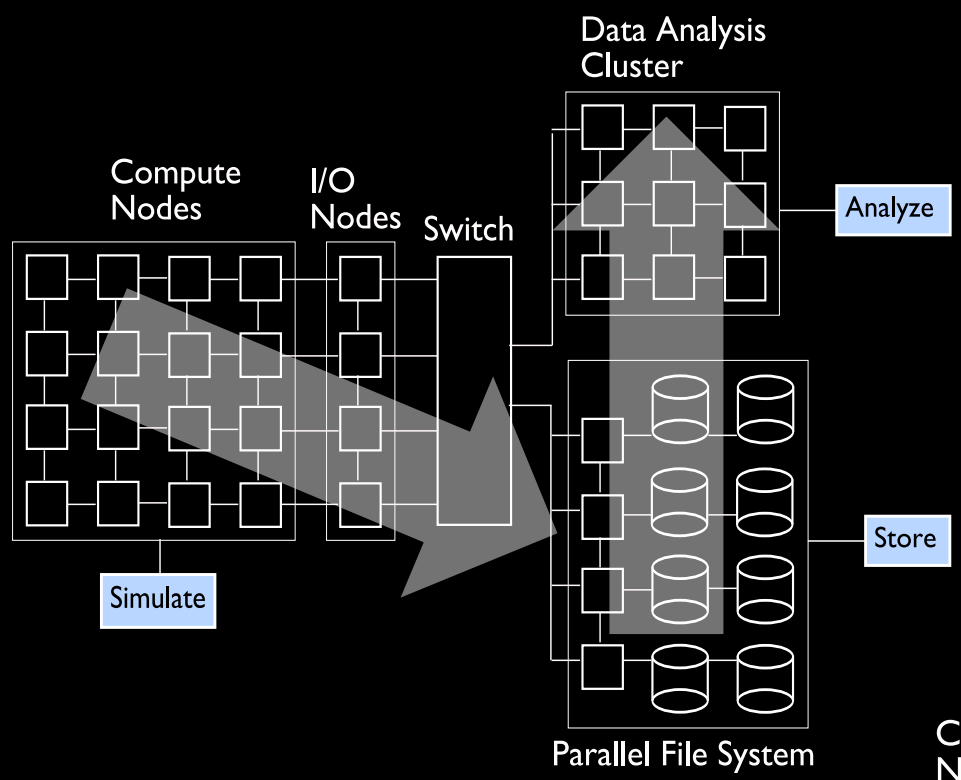
Tom Peterka

tpeterka@mcs.anl.gov

Mathematics and Computer Science Division

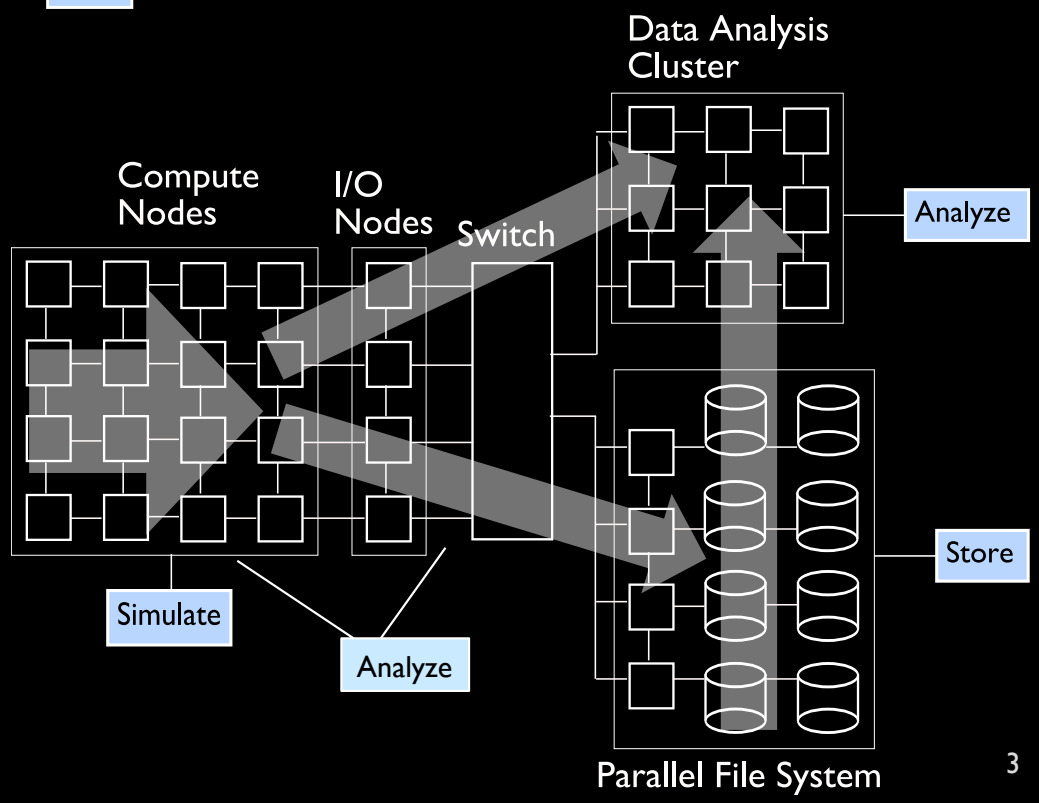
Preliminaries

Moving from Postprocessing to Run-Time Scientific Data Analysis in HPC



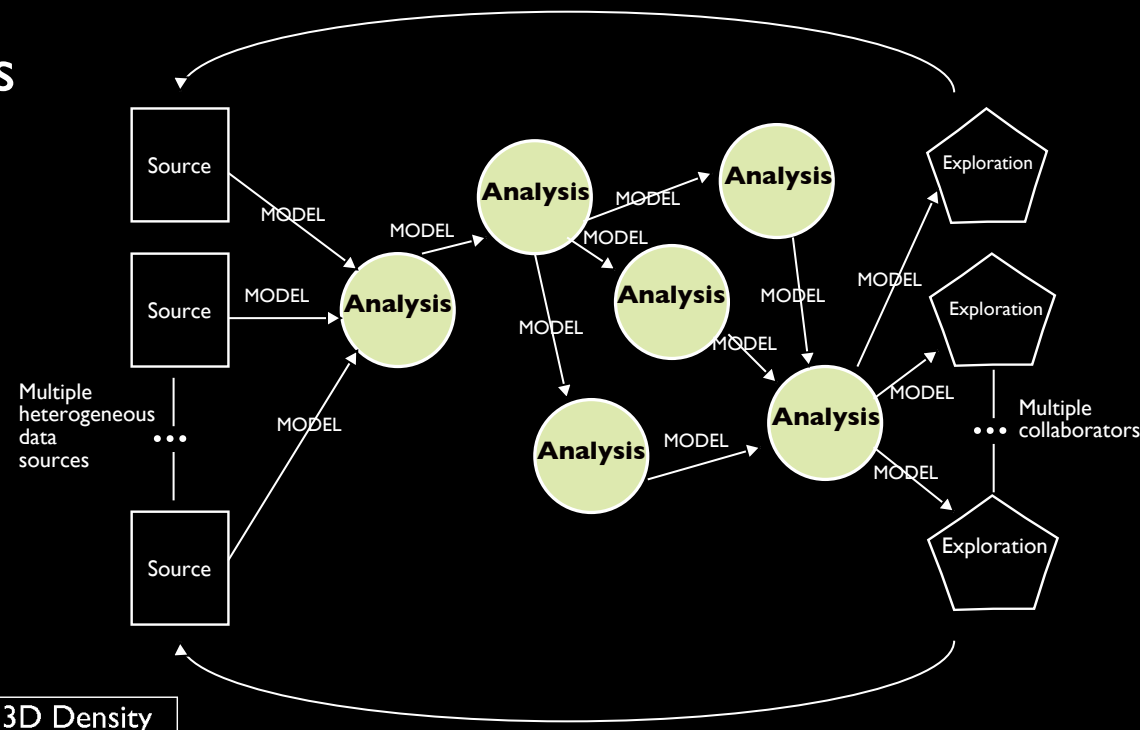
Postprocessing analysis and visualization

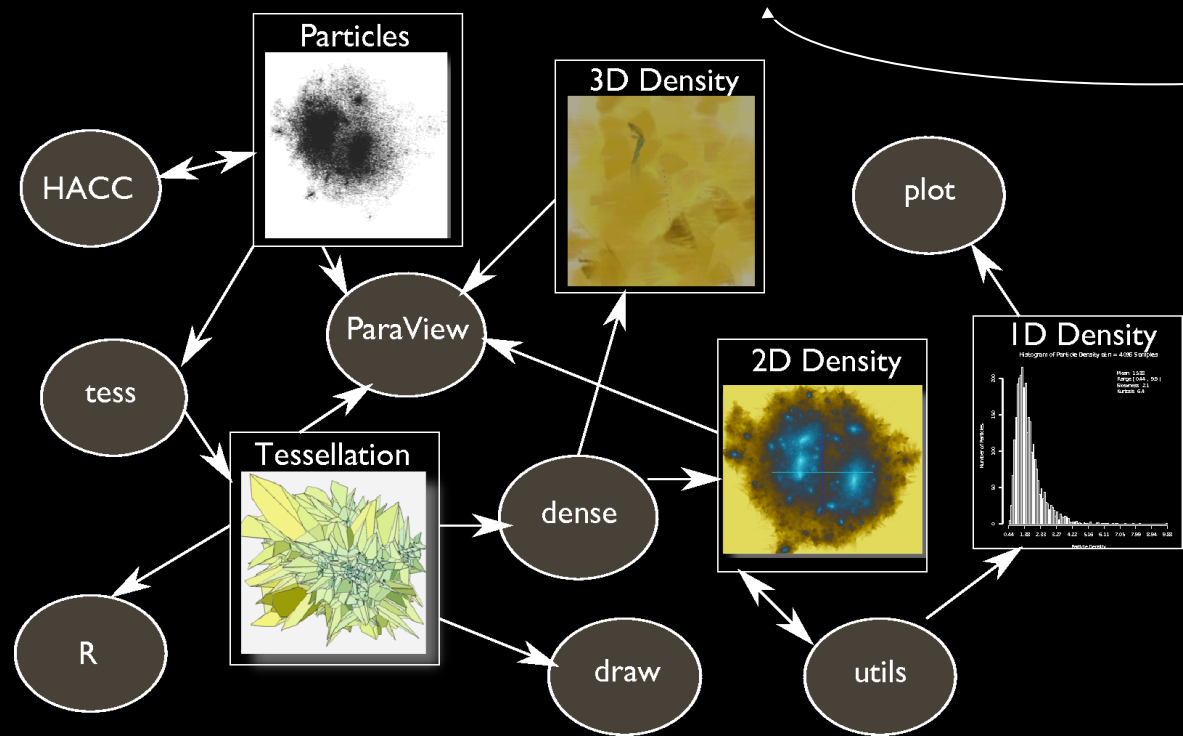
Run-time analysis and visualization



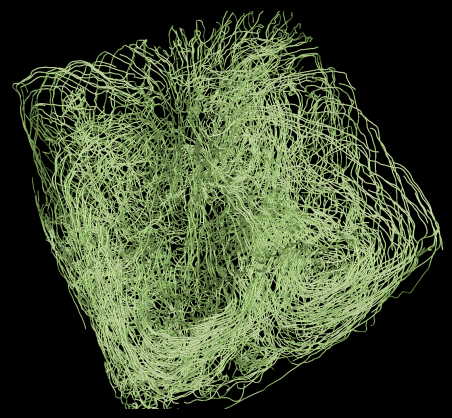
Definition of Data Analysis

- Any data transformation, or a network or transformations.
- Anything done to original data beyond its original generation.
- Can be visual, analytical, statistical, or data management.

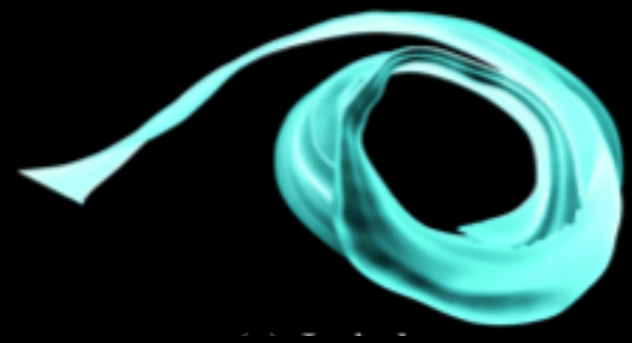




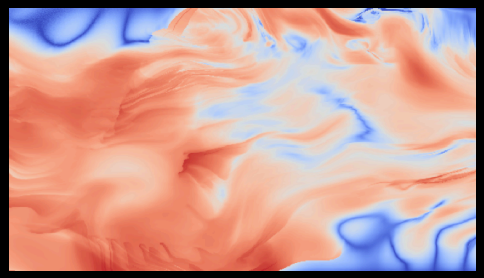
Examples



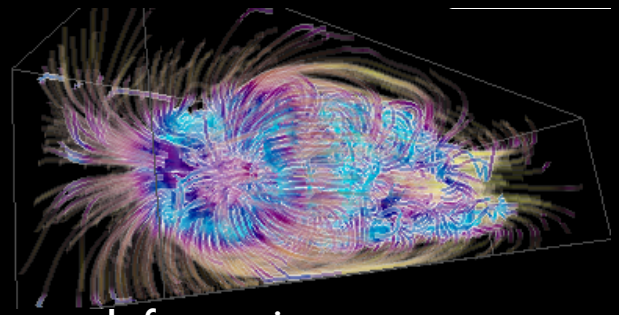
Streamlines and pathlines



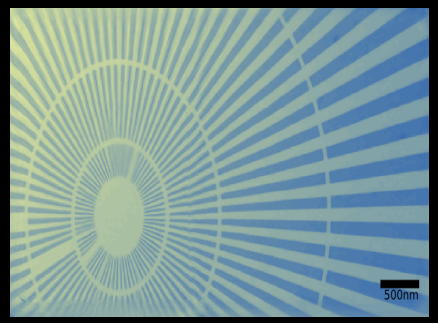
Stream surfaces



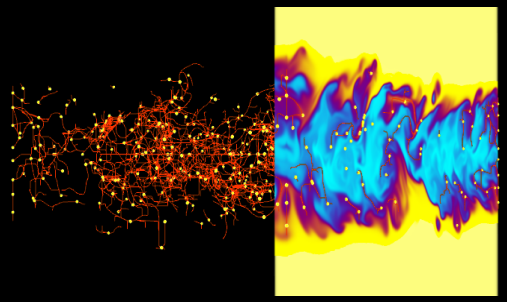
FTLE



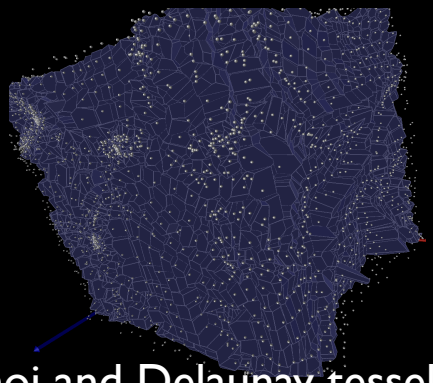
Information entropy



Ptychography



Morse-Smale complex



Voronoi and Delaunay tessellation

Common Denominators

- Big science => big data, big machines
- Most analysis algorithms are not up to speed
  - Either serial, or
  - Overheads kill scalability
- Solutions
  - Process data closer to the source
  - Write scalable analysis algorithms
  - Parallelize in various forms
  - Build software stacks of useful and reusable layers
- Usability and workflow
  - Develop libraries rather than tools
  - Users write small main programs and call into libraries

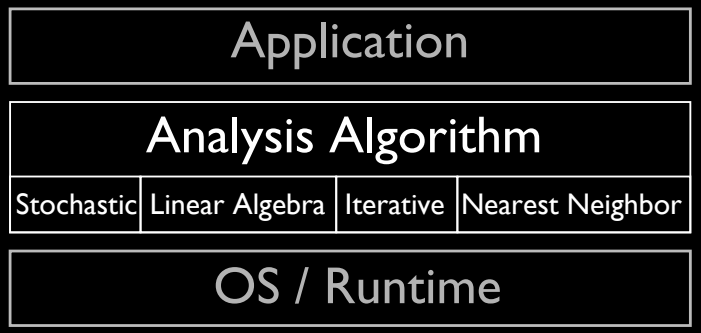
Abstractions Matter: Think Blocks, not Tasks

- Block = unit of decomposition
- Block size, shape can be configured
  - From coarse to fine
  - Regular, adaptive, KD-tree
- Block placement is flexible, dynamic
  - Blocks per task
  - Tasks per block
  - Memory / storage hierarchy
- Data is first-class citizen
  - Separate operations per block
  - Thread safety

Parallel data analysis consists of decomposing a problem into blocks, operating on them, and communicating between them.

You Have Two Choices to Parallelize Data Analysis

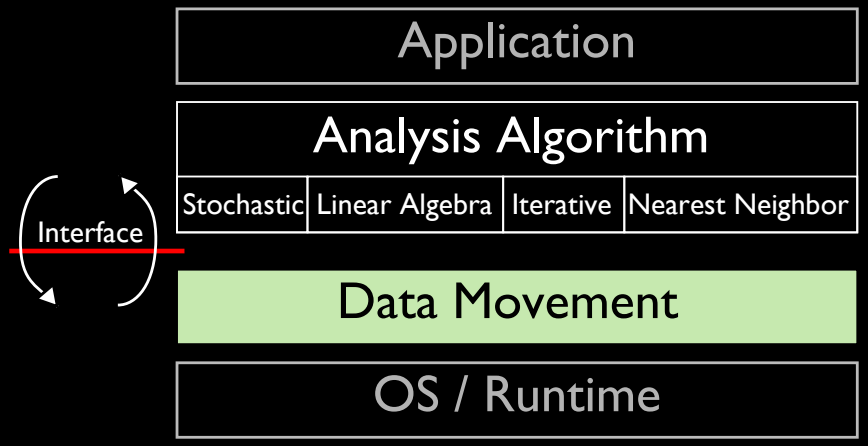
By hand




```
void ParallelAlgorithm() {
   …
   MPI_Send();
   …
   MPI_Recv();
   …
   MPI_Barrier();
   …
   MPI_File_write();
}
```

or

With tools




```
void ParallelAlgorithm() {
   …
   LocalAlgorithm();
   …
   DIY_Merge_blocks();
   …
   DIY_File_write()
}
```

DIY Concepts

DIY

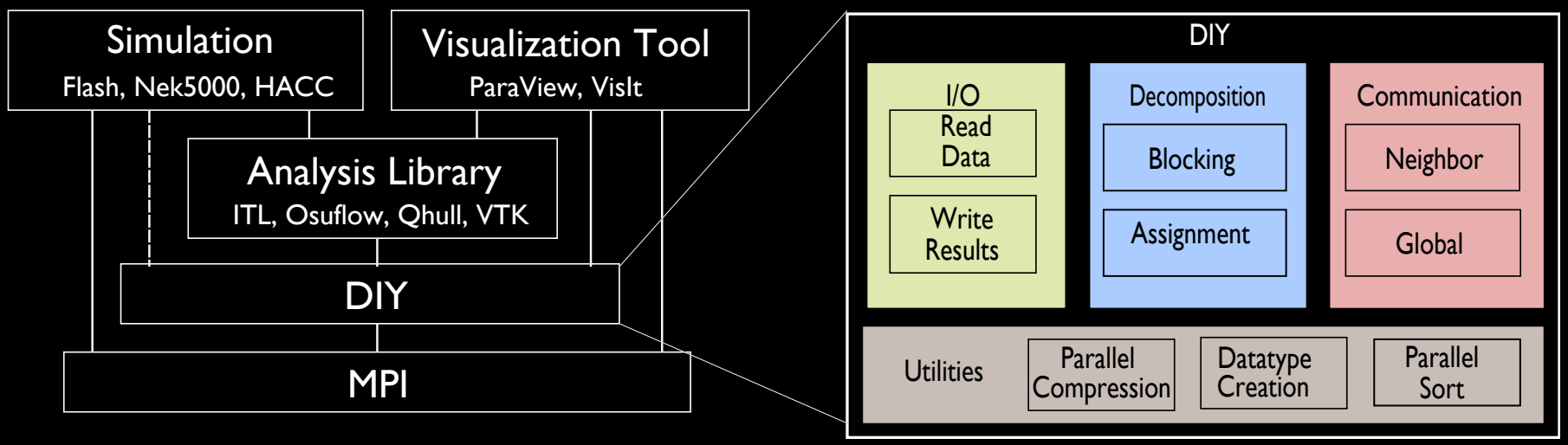
helps the user write data-parallel analysis algorithms by decomposing a problem into blocks and communicating items between blocks.

Features

- Parallel I/O to/from storage
- Domain decomposition
- Network communication
- Utilities

Library

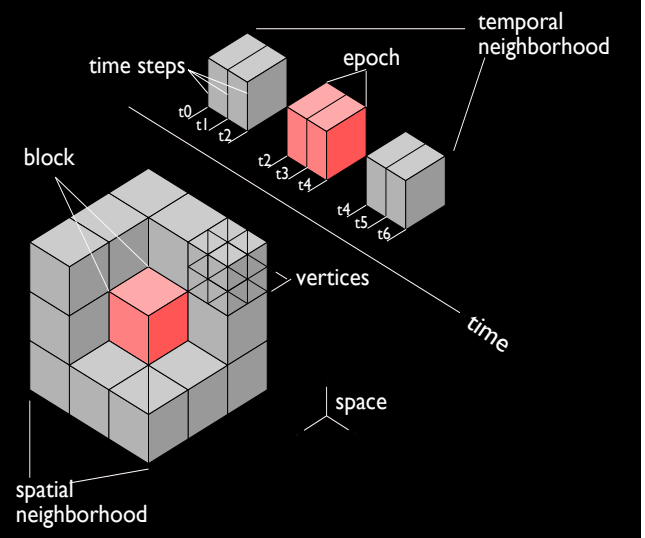
- Written in C++ with C bindings
- Autoconf build system (configure, make, make install)
- Lightweight: libdiy.a 800KB
- Maintainable: ~15K lines of code, including examples

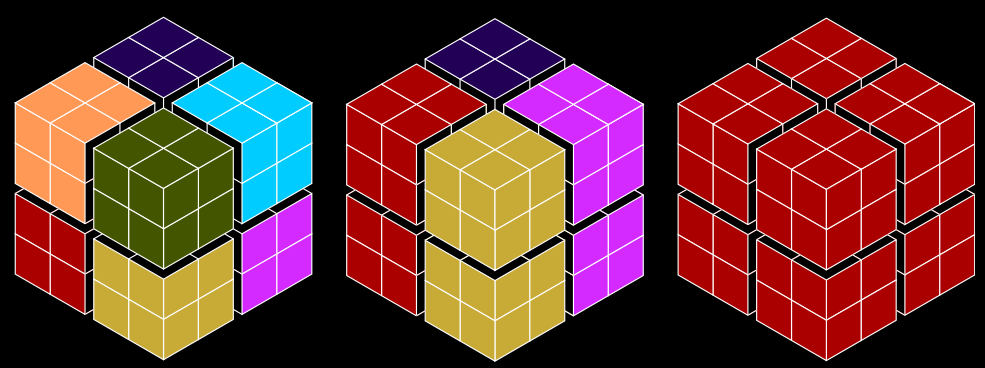


DIY usage and library organization

Nine Things That DIY Does

1. Separate analysis ops from data ops
2. Group data items into blocks
3. Assign blocks to processes
4. Group blocks into neighborhoods
5. Support multiple multiple instances of 2, 3, and 4
6. Handle time
7. Communicate between blocks in various ways
8. Read data and write results
9. Integrate with other libraries and tools

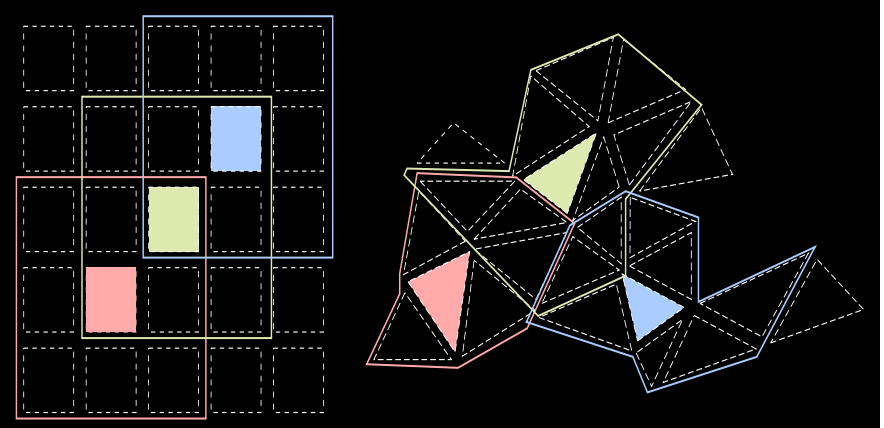




8 processes

4 processes

1 process



Two examples of 3 out of a total of 25 neighborhoods

Usage

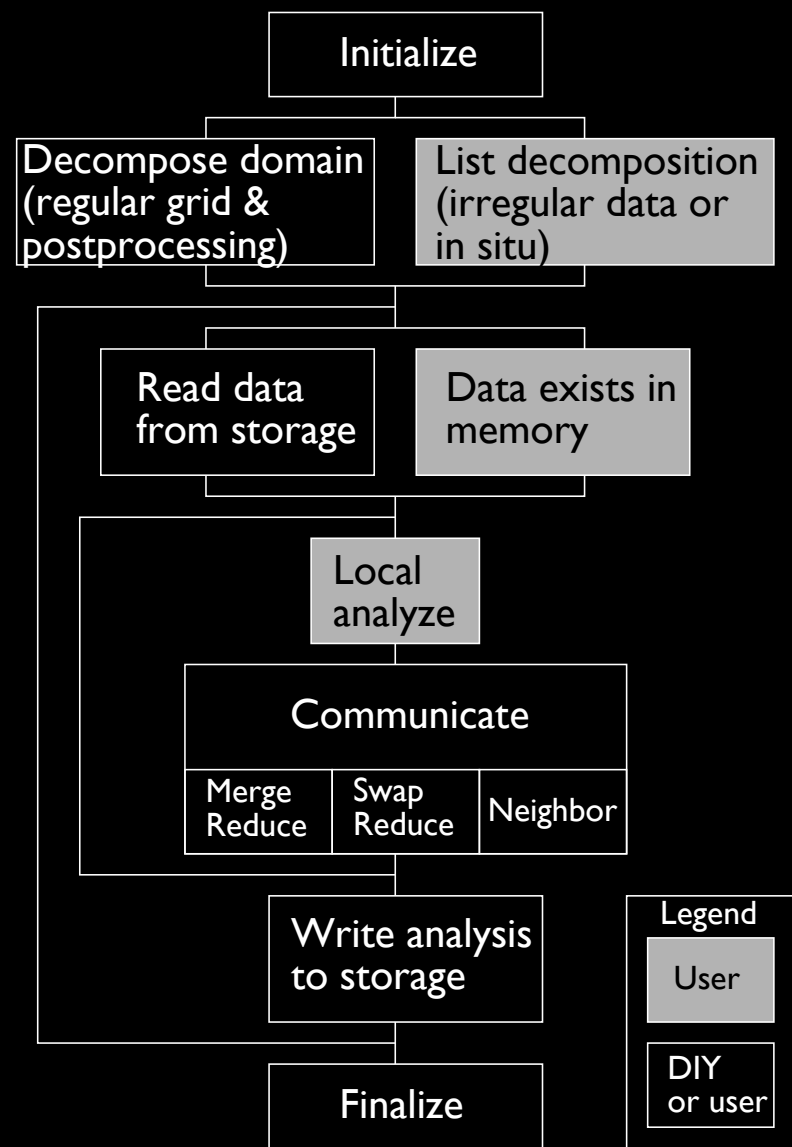
Writing a DIY Program

Documentation

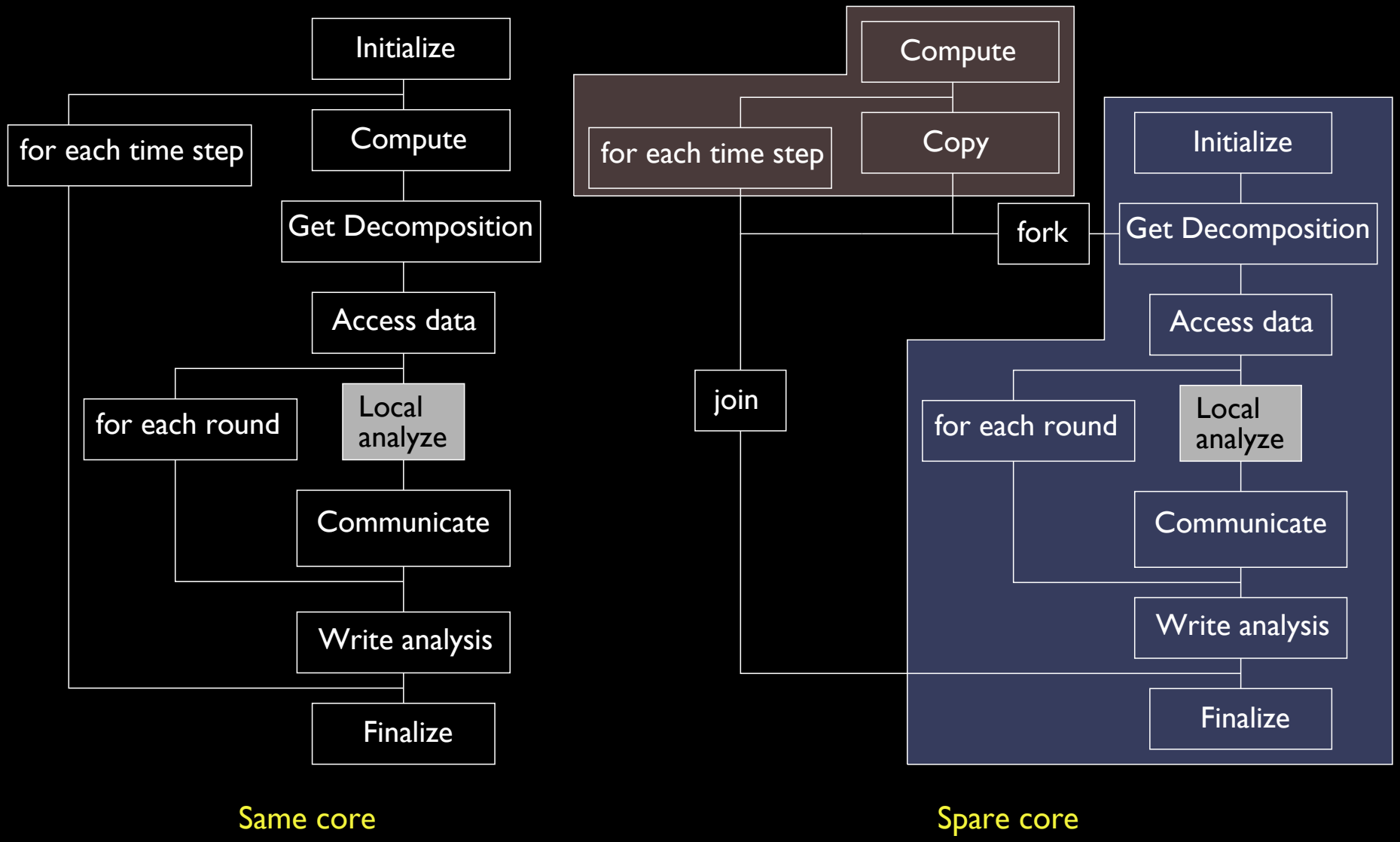
- README for installation
- User's manual with description, examples of custom datatypes, complete API reference

Tutorial Examples

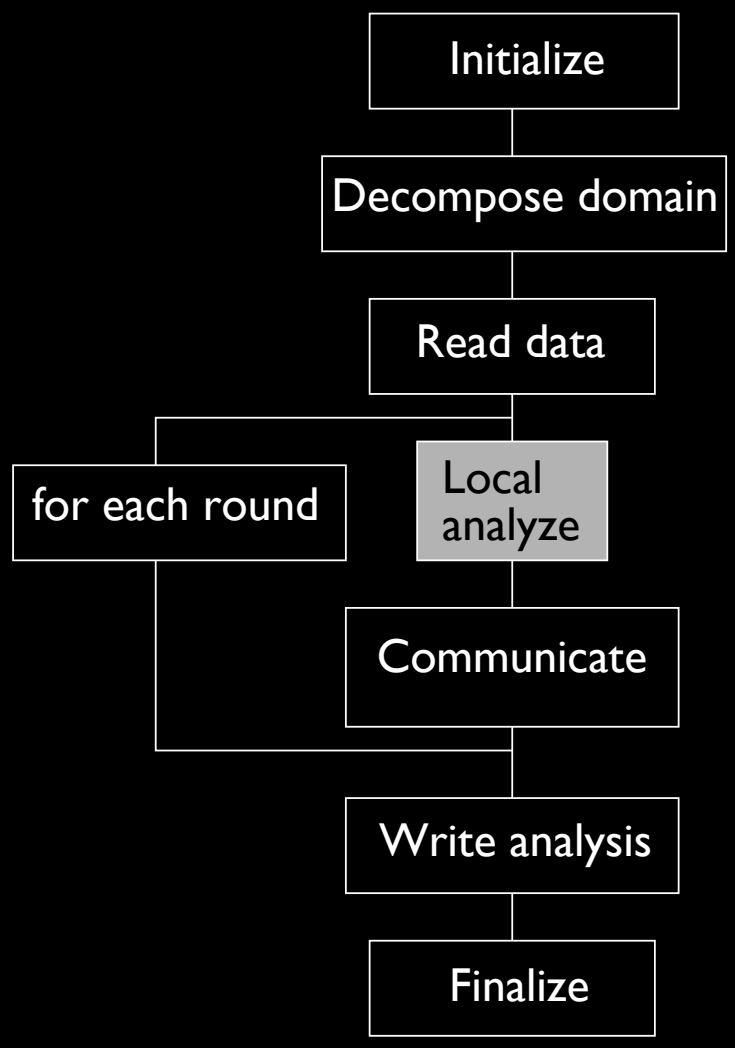
- Block I/O: Reading data, writing analysis results
- Static: Merge-based, Swap-based reduction, Neighborhood exchange
- Time-varying: Neighborhood exchange
- Spare thread: Simulation and analysis overlap
- MOAB: Unstructured mesh data model
- VTK: Integrating DIY communication with VTK filters
- R: Integrating DIY communication with R stats algorithms
- Multimodel: multiple domains and communicating between them



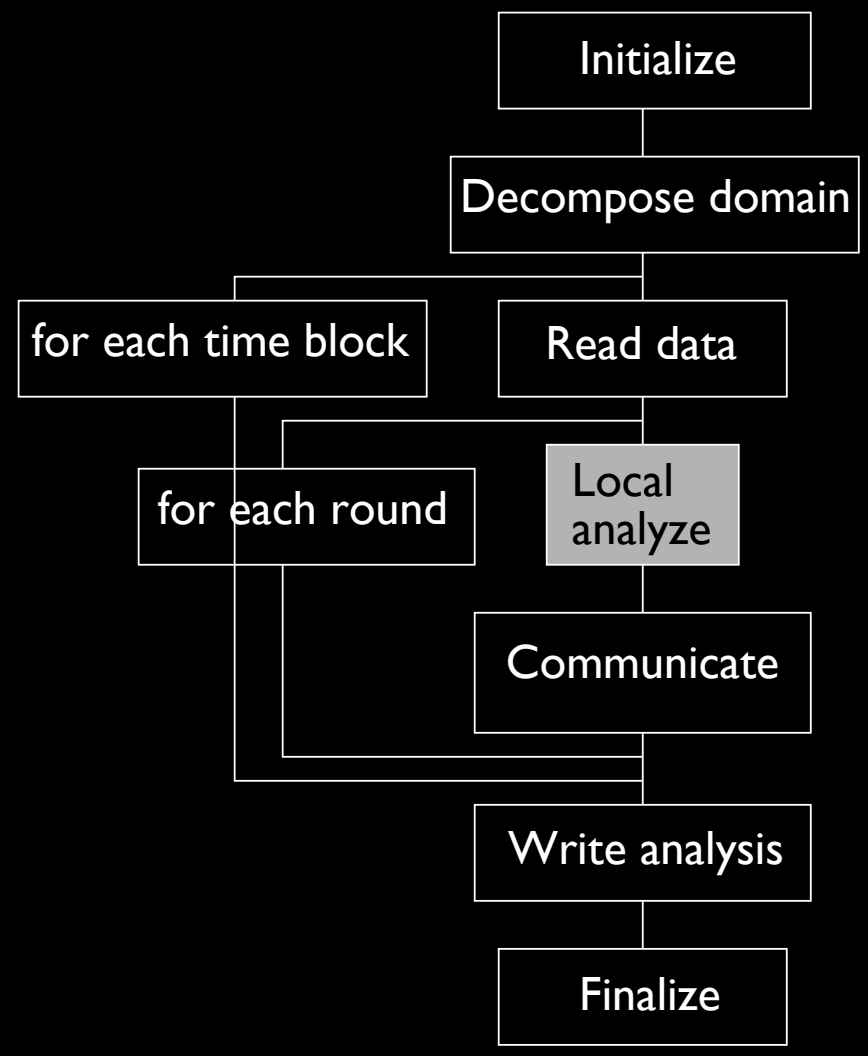
Execution: Same Core and Spare Core



Time: Static and Time-Varying

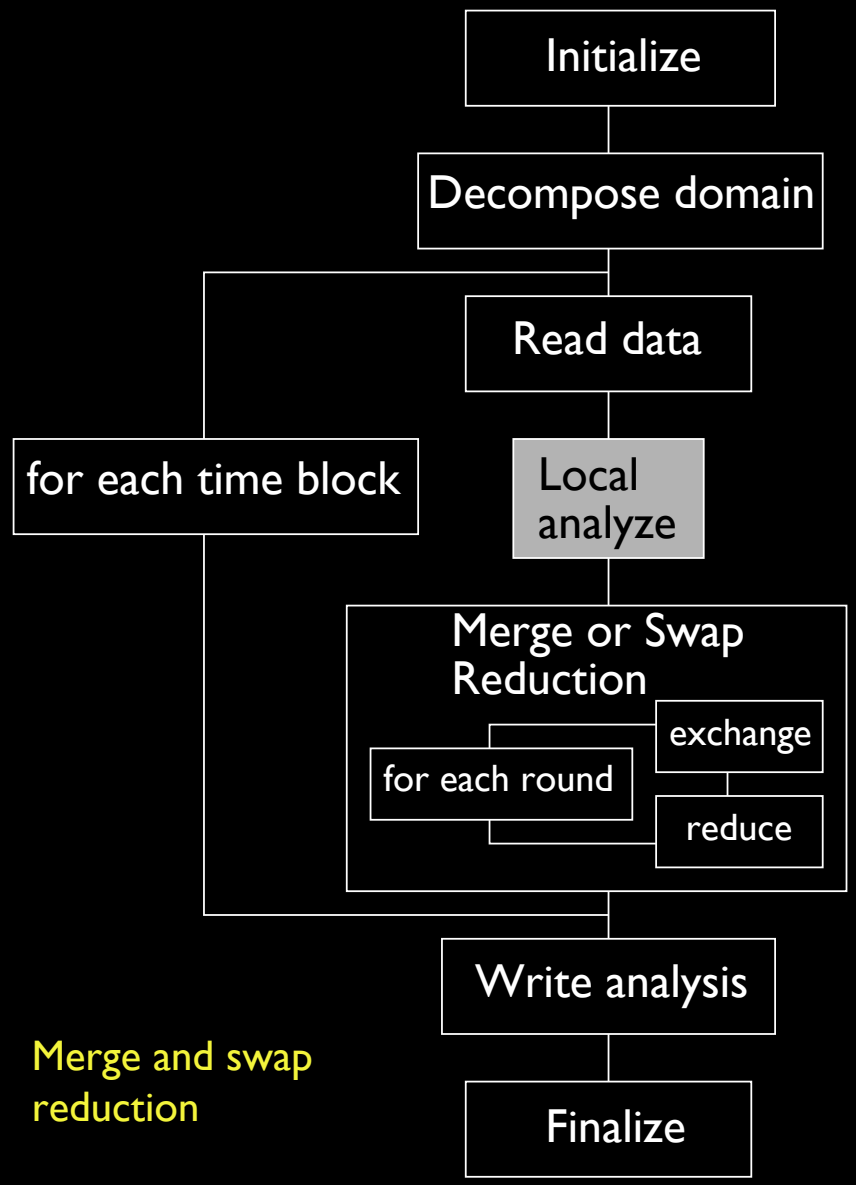


Static

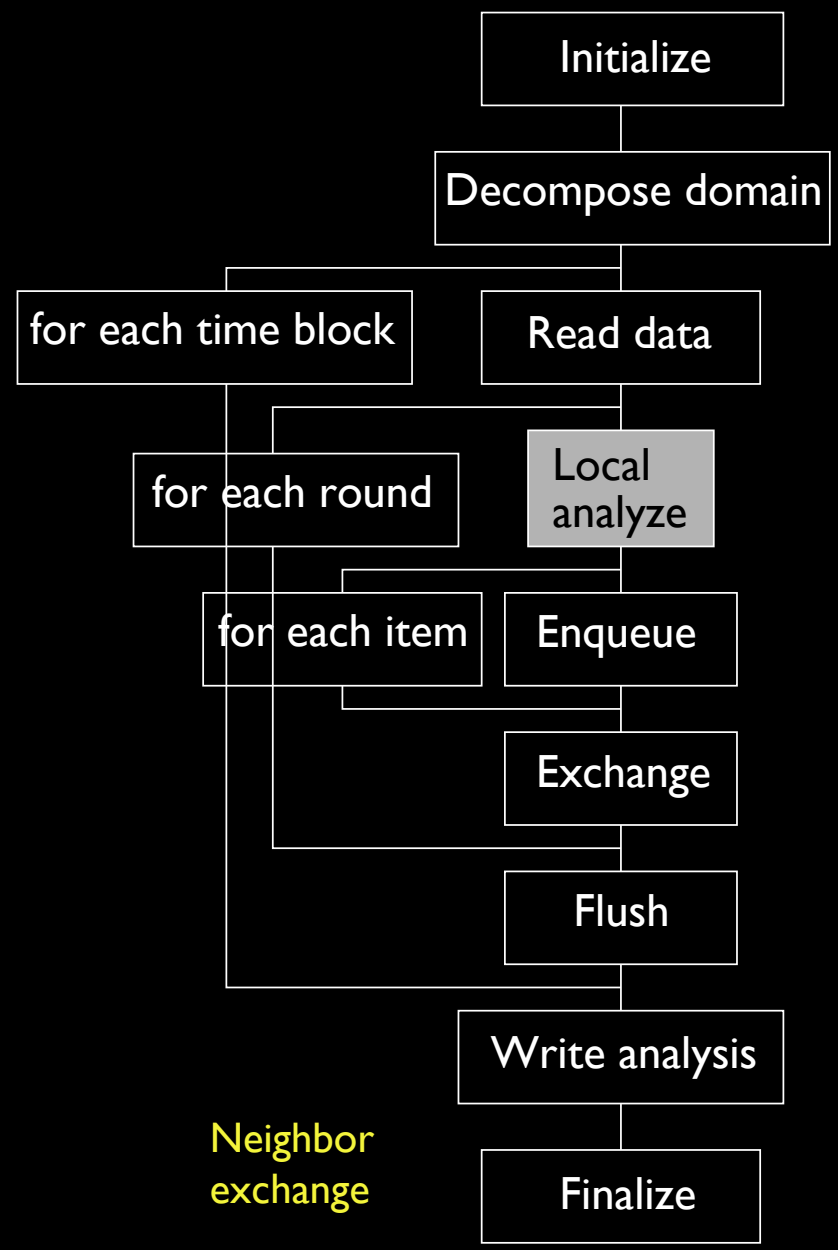


Time-varying

Communication: Merge and Swap Reduction, Neighbor Exchange



Merge and swap reduction



Neighbor exchange

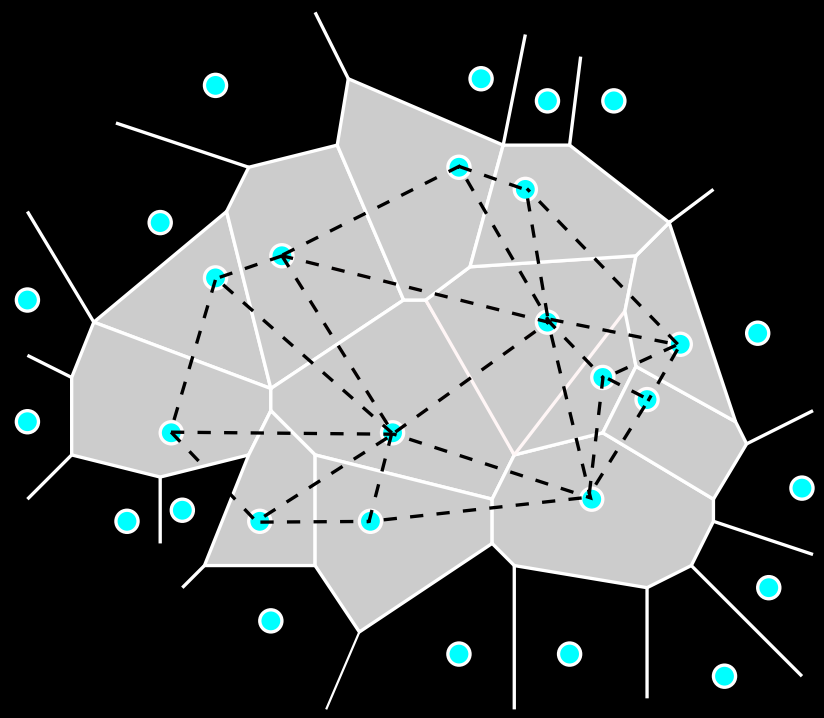
One Example in Greater Detail

Parallel Tessellation

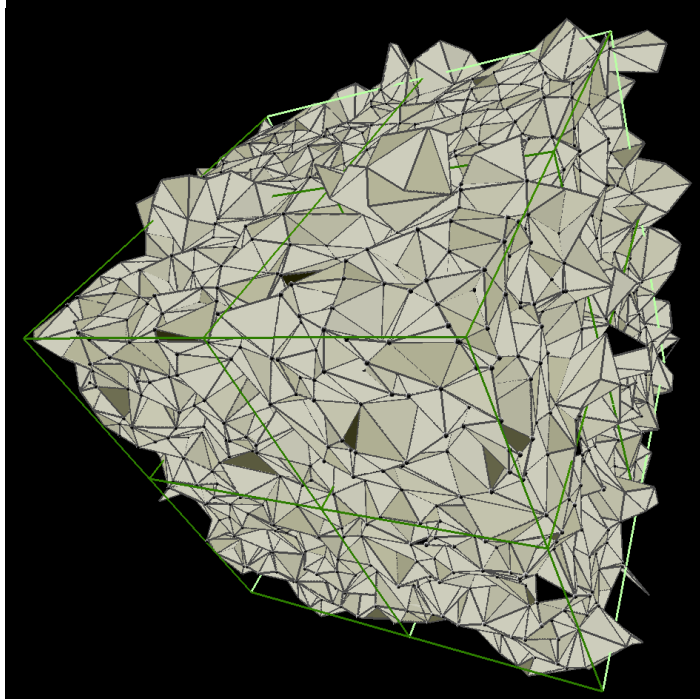
We developed a prototype library for computing in situ Voronoi and Delaunay tessellations from particle data and applied it to cosmology, molecular dynamics, and plasma fusion.

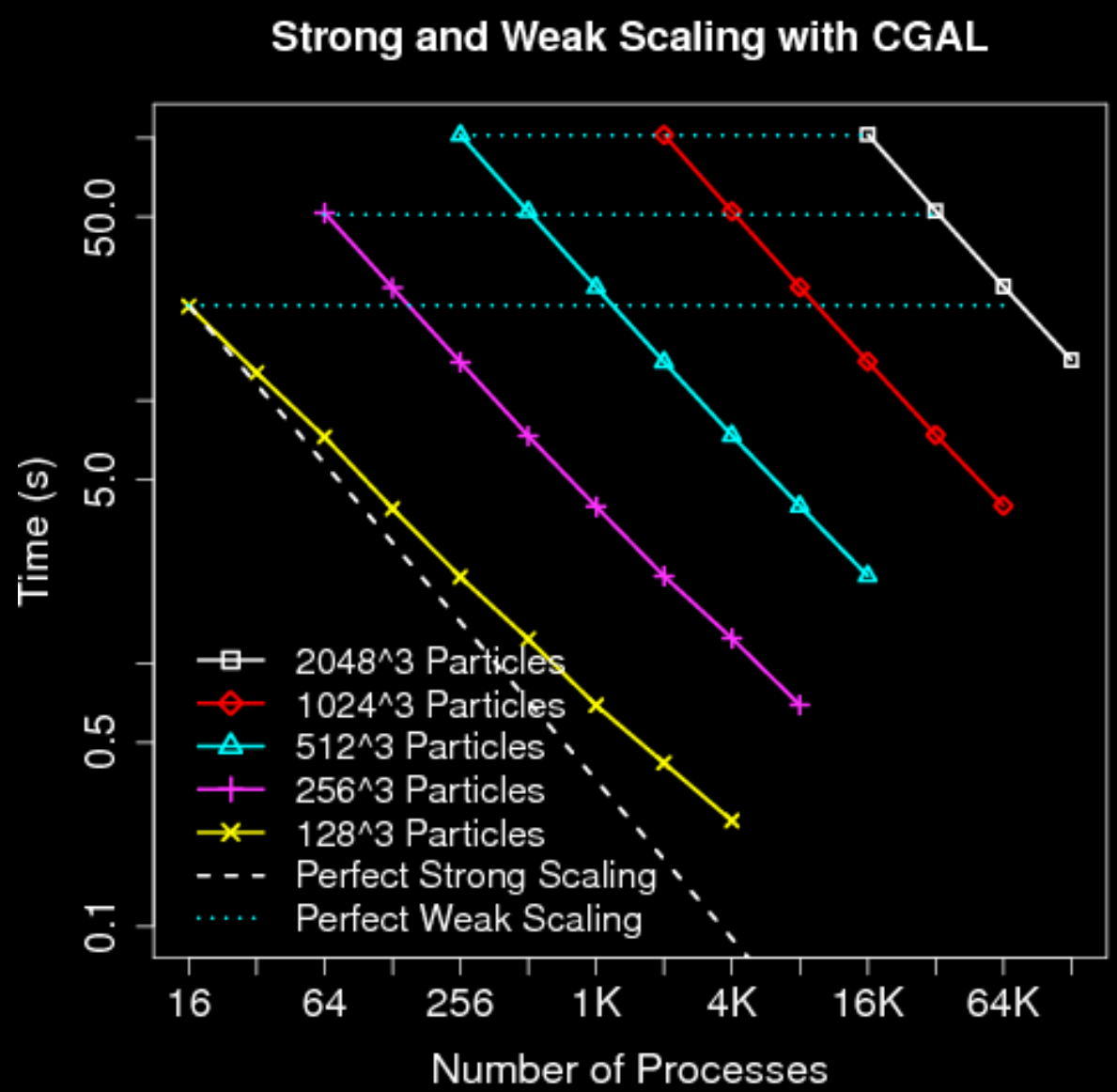
Key Ideas

- Mesh tessellations convert sparse point data into continuous dense field data.
- Meshing output of simulations is data-intensive and requires supercomputing resources
- No large-scale data-parallel tessellation tools exist.
- We developed such a library, tess.
- We achieved good parallel performance and scalability.
- Widespread GIS applicability in addition to the datasets we tested.



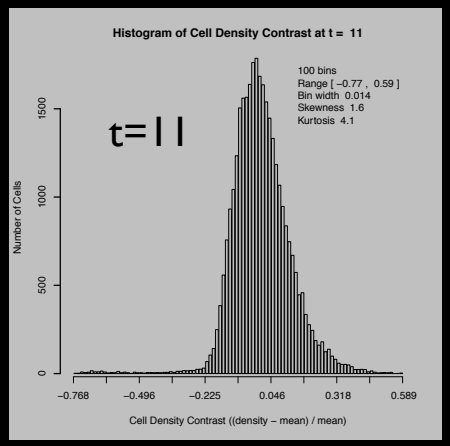
Scalability

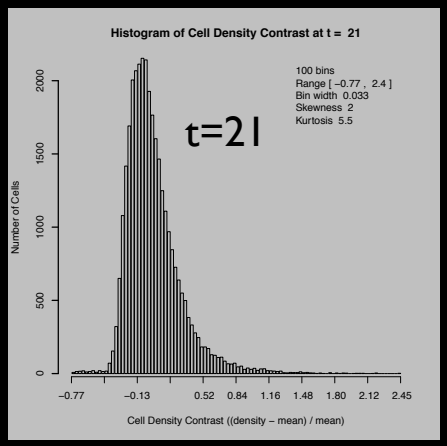


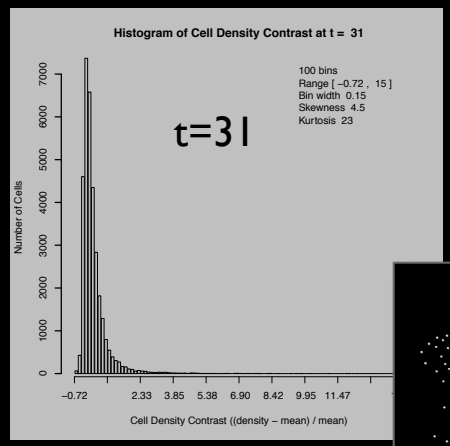


Strong and weak scaling for up to 2048^3 synthetic particles and up to 128K processes (excluding I/O) shows up to 90% strong scaling and up to 98% weak scaling.

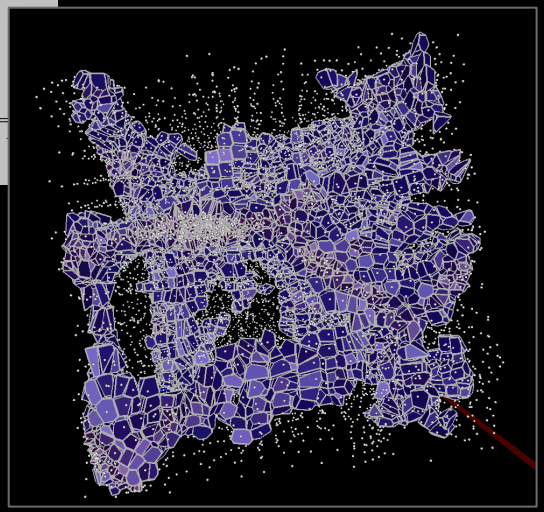
Applications in Cosmology



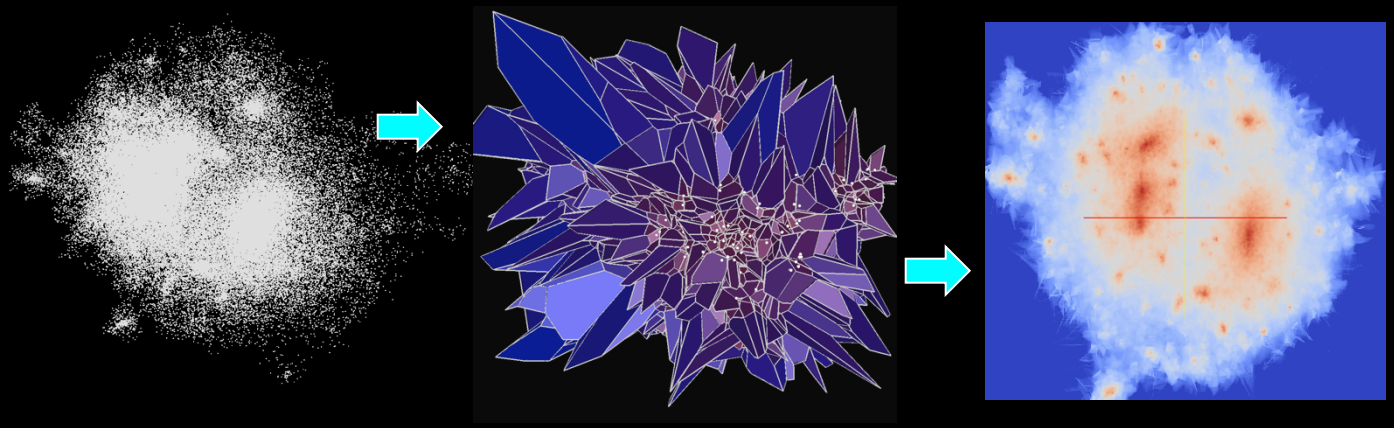




Feature statistics: Total volume, surface area, curvature, topology of connected components of Voronoi cells classify and quantify features.



Temporal structure dynamics: As time progresses, the range of cell volume and density expands, kurtosis and skewness increases. These statistics are consistent with the governing physics of the formation of high- and low-density structures over time and can perhaps be used to summarize evolution at given time steps.



Density estimation: Tessellations as intermediate representations enable accurate regular grid density estimators.

Recap

Block abstraction for parallelizing data analysis

Encapsulate data movement in a separate library

Define design patterns for data movement in HPC data analysis in terms of:

- Execution
- Temporal behavior
- Communication pattern

The benefits are:

- Abstraction, implementation independence
- Reuse, programmer productivity
- Standardization
- Benchmarking

Further Reading

DIY

- Peterka, T., Ross, R., Kendall, W., Gyulassy, A., Pascucci, V., Shen, H.-W., Lee, T.-Y., Chaudhuri, A.: Scalable Parallel Building Blocks for Custom Data Analysis. Proceedings of Large Data Analysis and Visualization Symposium (LDAV'11), IEEE Visualization Conference, Providence RI, 2011.
- Peterka, T., Ross, R.: Versatile Communication Algorithms for Data Analysis. 2012 EuroMPI Special Session on Improving MPI User and Developer Interaction IMUDI'12, Vienna, AT.

DIY applications

- Peterka, T., Ross, R., Nouanesengsey, B., Lee, T.-Y., Shen, H.-W., Kendall, W., Huang, J.: A Study of Parallel Particle Tracing for Steady-State and Time-Varying Flow Fields. Proceedings IPDPS'11, Anchorage AK, May 2011.
- Gyulassy, A., Peterka, T., Pascucci, V., Ross, R.: The Parallel Computation of Morse-Smale Complexes. Proceedings of IPDPS'12, Shanghai, China, 2012.
- Nouanesengsy, B., Lee, T.-Y., Lu, K., Shen, H.-W., Peterka, T.: Parallel Particle Advection and FTLE Computation for Time-Varying Flow Fields. Proeedings of SC12, Salt Lake, UT.
- Peterka, T., Kwan, J., Pope, A., Finkel, H., Heitmann, K., Habib, S., Wang, J., Zagaris, G.: Meshing the Universe: Integrating Analysis in Cosmological Simulations. Proceedings of the SC12 Ultrascale Visualization Workshop, Salt Lake City, UT.
- Chaudhuri, A., Lee-T.-Y., Zhou, B., Wang, C., Xu, T., Shen, H.-W., Peterka, T., Chiang, Y.-J.: Scalable Computation of Distributions from Large Scale Data Sets. Proceedings of 2012 Symposium on Large Data Analysis and Visualization, LDAV'12, Seattle, WA.

U.S. DEPARTMENT OF ENERGY

Argonne NATIONAL LABORATORY

"The purpose of computing is insight, not numbers."

–Richard Hamming, 1962

Acknowledgments:

Facilities

Argonne Leadership Computing Facility (ALCF)

Oak Ridge National Center for Computational Sciences (NCCS)

Funding

DOE SDMAV Exascale Initiative

DOE Exascale Codesign Center

DOE SciDAC SDAV Institute

https://bitbucket.org/diatomic/diy

Tom Peterka

tpeterka@mcs.anl.gov

Mathematics and Computer Science Division