Infrastructure for Topological (and Statistical) Analysis of Extreme-Scale Data
Data Analysis Comes in Many Flavors

Visual
Particle tracing of thermal hydraulics flow

Statistical
Information entropy analysis of astrophysics

Topological
Morse-Smale complex of combustion

Geometric
Voronoi tessellation of cosmology
Fortunately, Their Data Movement Patterns Do Not

Many different analysis operations share a small set of communication patterns. These communication kernels together with supporting utilities for decomposition and I/O can be encapsulated, optimized, and reused.
How to Parallelize Data Analysis: 2 Ways

By hand

Application

Analysis Algorithm

Stochastic | Linear Algebra | Iterative | Nearest Neighbor

OS / Runtime

or

With tools

void ParallelAlgorithm() {
    ...
    MPI_Send();
    ...
    MPI_Recv();
    ...
    MPI_Barrier();
    ...
    MPI_File_write();
}

void ParallelAlgorithm() {
    ...
    LocalAlgorithm();
    ...
    DIY_Merge_blocks();
    ...
    DIY_File_write();
}
DIY is a Library for Option #2.

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
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
Parallel Information-Theoretic Analysis
Collaboration with the Ohio State University and New York University Polytechnic Institute

Objective
- Decide what data are the most essential for analysis
- Minimize the information losses and maximize the quality of analysis
- Steer the analysis of data based on information saliency

Information-theoretic approach
- Quantify Information content based on Shannon’s entropy
- Use this model to design new analysis data structures and algorithms

Shannon’s Entropy
The average amount of information expressed by the random variable is

$$H(x) = -\sum_{i=1}^{n} p_i \log p_i$$
The computation of information entropy in a 126x126x512 solar plume dataset shows 59% strong scaling efficiency. 

Chaudhuri et al., Scalable Computation of Distributions from Large Scale Data Sets, LDAV '12
Parallel Topological Analysis
Collaboration with SCI Institute, University of Utah

- Transform discrete scalar field into Morse-Smale complex
- Nodes are minima, maxima, saddle points of scalar values
- Arcs represent constant-sign gradient flow
- Used to quickly see topological structure

Two levels of simplification of the Morse-Smale complex for jet mixture fraction.

Gyulassy et al., The Parallel Computation of Morse-Smale Complexes, IPDPS ‘12
Top: overview of our algorithm, the arrows and circled component indicate the sequence of operations performed by a single process: (a) parallel read, (b) local gradient computation, (c) local MS complex computation, (d) simplification, (e) preparing data structures for communication, (f) merging complexes, (g) parallel write.

Left: Typical flow chart of a DIY parallel analysis program. Lines to upper diagram map these to the specific parallel Morse-Smale Complex construction algorithm.
An artificially generated $256^3$ dataset is volume rendered (top row), and the corresponding Morse-Smale Complex is illustrated (bottom row) for varying feature counts.
Compute time, merge time, and output size as a function of number of processes, data size, and data complexity.
Above: Performance of parallel Morse-Smale analysis in log-log scale. Total time includes reading the dataset from storage, computing the analysis, and writing results to storage. Data size 1 timestep @ 1408x1080x1100.

Right: Percentage of time spent in each component of the Morse-Smale analysis.
Jet Mixture Fraction

Above: Overall time and four components: read data, compute, merge, and write results, plotted in log-log scale. At small numbers of processes, time is dominated by computing, and at higher numbers of processes by merging. Right: Full complex (top) and simplified complex (bottom). Data size one timestep @ 768x896x512.
Computation of Morse-Smale complex in $1152^3$ Rayleigh-Taylor instability data set results in 35% end-to-end strong scaling efficiency, including I/O. Data size 1 timestep @ $1152 \times 1152 \times 1152$. 

Strong Scaling
Recap and Looking Ahead

**Done: Benefits**
- Productivity
  - Express complex algorithms flexibly
  - Multiple blocks per process
  - Complete / partial reductions
  - Neighbor inclusion and communication
- Simplify existing tasks
  - Custom data type creation
  - Compression
- Performance
  - Published scalability
  - Configurable algorithms

**To Do: Research Directions**
- Advanced decomposition
  - Block groups
- Improved communication algorithms
  - Less synchronous, more overlap with computation
- High-level communication operations
  - Ghost cell exchange, kernel convolution (stencil)
- Load balancing
  - Block overloading, dynamic reassignment
- Programming models
  - MPI + X
- Usability
  - Improved API
References

DIY
• 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
3 Communication Patterns

Nearest neighbor

Swap-based reduction

Merge-based reduction