“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

I'm sure my wife will appreciate all the DIY I'm doing around the house for her!

Image courtesy pigtimes.com

Tom Peterka
tpeterka@mcs.anl.gov
Mathematics and Computer Science Division
Postprocessing Scientific Data Analysis in HPC Environments

Examples:
- 2D statistical graphics using R
- 3D scientific visualization using ParaView
- Scientific visualization using VisIt

Compute Nodes  I/O Nodes  Switch  Data Analysis Cluster

Simulate  Store  Analyze

Parallel File System
Run-time Scientific Data Analysis in HPC Environments

Examples:
GLEAN, ADIOS, ParaView Coprocessing Library

Data Analysis Cluster

Parallel File System

I/O Nodes

Switch

Compute Nodes

Simulate

Analyze

Examples:
GLEAN, ADIOS, DIY

R, ParaView, VisIt

Analyze

Store
Scientific Data Analysis Today

• Big science = big data, and
  • Big data analysis =&gt; big science resources
• Data analysis is data intensive.
  • Data intensity = data movement.
• Parallel = data parallel (for us)
  • Big data =&gt; data decomposition
  • Task parallelism, thread parallelism, while important, are not part of this work
• Most analysis algorithms are not up to the challenge
  • Either serial, or
  • Communication and I/O are scalability killers
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
You Have Two Choices to Parallelize Data Analysis

<table>
<thead>
<tr>
<th>By hand</th>
<th>or</th>
<th>With tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Analysis Algorithm</td>
<td>Application</td>
</tr>
<tr>
<td>Stochastic</td>
<td>Linear Algebra</td>
<td>Linear Algebra</td>
</tr>
<tr>
<td>Iterative</td>
<td>Nearest Neighbor</td>
<td>Iterative</td>
</tr>
<tr>
<td>OS / Runtime</td>
<td>OS / Runtime</td>
<td>OS / Runtime</td>
</tr>
</tbody>
</table>

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

void ParallelAlgorithm() {
    ...
    LocalAlgorithm();
    ...
    DIY_Merge_blocks();
    ...
    DIY_File_write();
}
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
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
Particle tracing of ¼ million particles in a 2048\(^3\) thermal hydraulics dataset results in strong scaling to 32K processes and an overall improvement of 2X over earlier algorithms.
Computation of information entropy in 126x126x512 solar plume dataset shows 59% strong scaling efficiency.
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.
For $128^3$ particles, 41% strong scaling for total tessellation time, including I/O; comparable to simulation strong scaling.
Further Reading

**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**

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

http://www.mcs.anl.gov/~tpeterka/software.html
https://svn.mcs.anl.gov/repos/diy/trunk

Tom Peterka
tpeterka@mcs.anl.gov
Mathematics and Computer Science Division