Distributed Data Analysis at Scale

“Data movement, rather than computational processing, will be the constrained resource at exascale.” – Dongarra et al. 2011.
Examples

Ridge detection in meteorology

Computational geometry in molecular dynamics

Density estimation in cosmology
DIY is a programming model and runtime for HPC block-parallel data analytics.

- Block parallelism
- Flexible domain decomposition and assignment to resources
- Efficient reusable communication patterns
- Automatic dual in- and out-of-core execution
- Automatic block threading
Basic Concepts
Partition Data Into Blocks

The block is the basic unit of data decomposition. Original dataset is decomposed into generic subsets called blocks, and associated analysis items live in the same blocks. Blocks don’t have to be “blocky.” Any subdivision of data (eg., a set of graph nodes, a group of particles, etc.) is a block.
1. Decomposition can be a regular grid of blocks or a k-d tree.

2. For a regular grid, constraints on numbers of blocks can be imposed to get pencil or slab shapes.

3. Multiple decompositions can co-exist.
Neighborhood Links

- Limited-range communication
- Allow arbitrary groupings
- Distributed, local data structure and knowledge of other blocks (not master-slave global knowledge)

Examples of 3 neighborhoods in a regular grid, unstructured mesh, and graph.
Communicate over the Link

DIY provides point to point and different varieties of collectives within a neighborhood via its enqueue/exchange/dequeue mechanism.

How to enqueue items for neighbor exchange

• DIY offers several options
• Send to a particular neighbor or neighbors, send to all nearby neighbors, send to all neighbors
• Support for periodic boundary conditions
Global Communication Patterns

Merge-reduce

Round 0
k = 4

Round 1
k = 2

Results

Swap-reduce

Round 0
k = 4

Round 1
k = 2

Results

Swap-reduce

Round 0
k = 4

Round 1
k = 2

Results

Swap-reduce

Round 0
k = 4

Round 1
k = 2

Results
// initialization
Master master(world, num_threads, mem_blocks, ...);
ContiguousAssigner assigner(world.size(), tot_blocks);
decompose(dim, world.rank(), domain, assigner, master);

// compute, neighbor exchange
master.foreach(&foo);
master.exchange();

// reduction
RegularSwapPartners(dim, tot_blocks, k);
reduce(master, assigner, partners, &foo);

// callback function for each block
void foo(void* b, const Proxy& cp, void* aux)
{
    for (size_t i = 0; i < in.size(); i++)
        cp.dequeue(cp.link()->target(i), incoming_data);
    // do work on incoming data
    for (size_t i = 0; i < out.size(); i++)
        cp.enqueue(cp.link()->target(i), outgoing_data[i]);
}
One Example in Detail
Self-Adaptive Density Estimation

Sampling a regular density field from a distribution of particle positions using a Voronoi tessellation as an intermediate data model.

Key Ideas

• Convert discrete particle data into continuous function that can be interpolated, differentiated, interpolated, represented as a regular grid (field)
• Automatically adaptive window size and shape
• Comparison with CIC using synthetic and actual data
• Voronoi tessellation and density estimation computed in parallel on distributed-memory HPC machines
In cloud-in-cell (CIC) methods, particles are distributed to a fixed number of grid points.

In tessellation (TESS) methods, particles are distributed to a variable number of grid points according to the Voronoi or Delaunay tessellation that has variable size and shape cells.
Overall Algorithm

1. Form Voronoi cells in 3D
2. Find covered grid points in 3D
3. Distribute particle mass over covered 3D grid points
4. Optionally project grid point mass to 2D
5. Convert mass to 3D or 2D density

```plaintext
for (all Voronoi cells) {
    compute grid points in Voronoi cell interior
    for (all interior grid points) {
        if (grid point is inside local block)
            add mass contribution to grid point
        else
            send mass contribution to neighbor block containing grid point and add it there
        if (2D projection) {
            accumulate mass at 2D pixel
            divide by pixel area for 2D density
        }
    }
    else
        divide by voxel volume for 3D density
} // interior grid points
} // Voronoi cells
```
Accuracy
Navarro-Frenk-White (NFW)

Synthetic dataset derived from an analytical density function commonly used in cosmology.

$k$ is a constant, 1 for us

$\rho(r)$ is Monte Carlo sampled to get test set of particles

Ground truth is 2D plot of $\rho(r)$

We limit $r$ to [-1.5, 1.5] and $\text{NFW}(r)$ to $10^6$

$$\rho(r) = \frac{k}{(r(r + 1)^2)}$$
NFW 2D Density Fields

Top row:
1024^3 3D density projected to 1024^2 2D density field and rendered in ParaView

Bottom row:
Ratio of analytical divided by estimated density
Comparison between analytical 2D density and estimated density at $y = 0$ cross section.

Ratio between analytical 2D density divided by estimated density at $y = 0$ cross section.

$Y = 0$ Slice of Ratio of Analytical / Estimated Density

$Y = 0$ Slice of Analytical and Estimated Density
Comparison between analytical 2D density and estimated density at $y = 0$ cross section.

Ratio between analytical 2D density divided by estimated density at $y = 0$ cross section.
Complex NFW (CNFW)

Our second synthetic dataset is a combination of several NFWs of varying cutoff densities and asymmetric scaling factors.

Analytical cutoff density contours

2e5 sampled particles

Voronoi tessellation
CNFW 2D Density Fields

Top row: $1024^3$ 3D density projected to $1024^2$ 2D density field and rendered in ParaView

Bottom row: Ratio of analytical divided by estimated density
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.
Performance of Density Estimation

Left: Strong scaling of estimating the density of $512^3$ synthetic particles onto grids of various sizes.

Below: Density estimation of one halo of dark matter particles in a cosmology simulation.

Recap
How to DIY Data Analysis

**DIY data movement library for parallelizing data analysis**
- Decompose data into blocks
- Assign blocks to processing elements
- Have several decompositions at once
- Overload blocks, migrate blocks between processing elements
- Communicate between blocks
- Migrate blocks in and out of core
- Thread blocks with finer-grained processing elements

**Tessellation-based density estimation example**
- Parameter-free
- Shape-free
- Automatically adaptive
- Higher quality estimation in high-contrast data
- Scalable parallel performance
References

DIY Papers

• Peterka, Ross, Kendall, Gyulassy, Pascucci, Shen, Lee, Chaudhuri: Scalable Parallel Building Blocks for Custom Data Analysis. LDAV 2011.
• Morozov, Peterka: Block-Parallel Data Analysis with DIY2. Submitted to LDAV 2016.

Selected DIY Application Papers

• Lu, Shen, Peterka: Scalable Computation of Stream Surfaces on Large Scale Vector Fields. SC14.
• Gyulassy, Peterka, Pascucci, Ross: The Parallel Computation of Morse-Smale Complexes. IPDPS 2012.
Facilities
Argonne Leadership Computing Facility (ALCF)
Oak Ridge National Center for Computational Sciences (NCCS)
National Energy Research Scientific Computing Center (NERSC)

Funding
DOE SDMAV Exascale Initiative
DOE SciDAC SDAV Institute

People
Dmitriy Morozov (LBNL)

github.com/diatomic/diy2
github.com/diatomic/tess2

Analysis, Storage, and Privacy for Big Data Seminar
JSM 2016
August 4, 2016