Title: Portable Data-Parallel Algorithms for Scientific Applications

Author(s): Christopher Sewell

Intended for: DOE Computer Graphics Forum, Newport, RI, April 2015
Portable Data-Parallel Algorithms for Scientific Applications

Christopher Sewell
Li-ta Lo, Jim Ahrens, Marianne Francois, Pat Fasel

DOE Computer Graphics Forum, April 2015

LA-UR-15-23133
Overview of Portable Data-Parallelism at LANL

- Original PISTON project
  - Goal: Start developing visualization and analysis algorithms using data-parallel primitives so that they can take advantage of multi-core/many-core architectures, and be portable across them
  - Achievements: Implemented several algorithms (e.g., isosurface, cut surface, threshold, KD-tree, simple renderers), demonstrated their performance, extended Thrust (e.g., distributed, OpenCL), created VTK/ParaView plug-in

- Current projects
  - SDAV: Deliver portable data-parallel visualization and analysis algorithms to science codes (e.g., cosmology)
  - XVis: Integrate into emerging community-standard VTK-m, and build library of useful filters
  - ASC: Use portable data-parallelism for physics computations


SDAV
Scalable Data Management, Analysis, and Visualization
SciDAC Institute
Cosmology Halo Analysis: The Science Problem

- The Hardware/Hybrid Accelerated Cosmology Code (HACC) simulates the distribution of dark matter in the universe over time.
- Accurately determining halos (high density regions) and halo centers is essential for comparison of results with observations and for calculating properties such as concentration.
- To achieve high accuracy, simulations may use 100s of billions of particles.
- Existing serial-per-rank halo analysis code was far too slow to analyze halos in large simulations.
- Supporting the cross-product of multiple variants of halo and center finding definitions with all of the current and emerging multi-core and many-core architectures is a major burden.
- Cosmology collaborators: Katrin Heitmann and Salman Habib (Argonne).

Illustration of the role of analysis in cosmology simulations. Figure from “Large Scale Simulations of Sky Surveys” by Heitmann et. al.
Cosmology Halo Analysis: The Portable Data-Parallel Solution

- Halo finding algorithm
  - Based on standard parallel sparse connected components algorithm
  - Space partitioning scheme used since edge list or adjacent matrix would be too expensive to compute or store

- Most bound particle center finder
  - Compute potential for all particles, and find minimum
  - Simple but highly scalable

- Hybrid version: most speed-up from center finding, most memory usage from halo finder
PISTON on GPU ~9.0x faster than original with 1 rpn and ~4.9x faster than original with 16 rpn, with most speed-up due to center finder

PISTON’s portability allowed it to be run and scale with OpenMP on a MIC
On Titan, MBP centers to be found on the GPU ~50x faster than using the pre-existing algorithms on the CPU (with one rank per node)

This enabled us to run halo analysis for the full $8192^3$ particle data set

Cosmology Halo Analysis: The Science Result

- This work allowed halo analysis to be completed on all time steps of the Q Continuum simulation, with an $8192^3$ particle data set across 16,384 nodes on Titan for which analysis using the existing CPU algorithms was not feasible.

- This is the first time that the c-M relation has been measured from a single simulation volume over such an extended mass range.

---

Concentration-mass relation over the full mass range covered by the Q Continuum simulation at redshift $z=0$

Strong lensing arc generated from the cluster-scale halo


UNCLASSIFIED
Data-Parallel Cosmology Analysis on Titan

Objectives

- **Milestones**
  - Implement application-specific visualization and/or analysis operators needed for in-situ use by LCF science codes
  - Use VTK-m to take advantage of multi-core and many-core hardware technologies

- **Target Application**
  - The Hardware/Hybrid Accelerated Cosmology Code (HACC) simulates the distribution of dark matter in the universe over time
  - An important and time-consuming analysis function within this code is finding halos (high density regions) and centers of those halos

Impact

- **Portable, Data-Parallel Halo and Center Finding Algorithms**
  - Data-parallel algorithms for halo and center finding implemented using VTK-m allow the code to take advantage of parallelism on accelerators such as GPUs, and for the code to be portable across architectures
  - Can be used for post-processing or in-situ (including with CosmoTools library)

- **Streaming Workflow for Large Halos**
  - At late time steps, particles become more concentrated in some nodes, resulting in great load imbalance for center finding
  - During initial analysis task, particles in halos larger than specified size are output to disk using HACC’s GenericIO library, which can then be streamed into memory from file to find centers one at a time as single-node jobs

Results

- Concentration-mass relation over the full mass range covered by the Q Continuum simulation at redshift z = 0 (points with error bars) and the predictions from various groups. The yellow shaded region shows the intrinsic scatter. All predictions and the simulation results are well within that scatter.

Accomplishments

- This work allowed halo analysis to be completed on all time steps of a very large 8192^3 particle data set across 16,384 nodes on Titan for which analysis using the existing CPU algorithms was not feasible

- This is the first time that the c-M relation has been measured from a single simulation volume over such an extended mass range (see graph at left)

In-Situ and Co-Scheduling Workflows on Titan

- Analysis algorithms have been implemented in-situ with HACC using the CosmoTools library.
- At late time steps, particles become more concentrated in some nodes, resulting in great load imbalance for analysis tasks such as center finding and subhalo finding.
- Analysis for large halos can be co-scheduled with main simulation run.

In-Situ and Co-Scheduling Workflows: Q Continuum Run

- At final time step, center finding on slowest node (5.9 hours) is about four orders of magnitude slower than on fastest node (2.4 seconds).
- Finding centers for all halos under 300,000 particles in-situ took less than one minute.
- Halos over 300,000 particles were written to disk (>5x less data than writing all halos to disk) and analyzed by a set of single-node jobs.
- Total of 0.52M core-hours compared to 3.4M to find all in-situ.

<table>
<thead>
<tr>
<th>SLICE</th>
<th>Redshift z</th>
<th>Max Find</th>
<th>Min Find</th>
<th>Max Center</th>
<th>Min Center</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>1.680</td>
<td>433</td>
<td>352</td>
<td>449</td>
<td>19</td>
</tr>
<tr>
<td>64</td>
<td>1.433</td>
<td>483</td>
<td>385</td>
<td>668</td>
<td>19</td>
</tr>
<tr>
<td>73</td>
<td>0.959</td>
<td>663</td>
<td>532</td>
<td>1819</td>
<td>19</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>2143</td>
<td>1859</td>
<td>21250</td>
<td>2.4</td>
</tr>
</tbody>
</table>
In-Situ and Co-Scheduling Workflows: Comparing Approaches

- Ran downscaled simulation ($1024^3$ particles on 32 nodes) and analysis with four workflows
  - In-situ only: no extra I/O needed, but idle nodes due to load imbalance (15x slowest/fastest for center-finding and 5.6x for subhalo finding)
  - Offline-only: extra I/O and redistribution for all halos and queuing of another full-size job, but all data available for exploration
  - In-situ/off-line: I/O only for large halos (~2x reduction), and queuing of smaller job; 30% reduction in total core hours
  - In-situ/co-scheduling: same core hours as in-situ/off-line, but no extra queuing and simpler workflow

- This approach should also enable efficient use of a burst buffer

<table>
<thead>
<tr>
<th>Method</th>
<th>I/O</th>
<th>Redist.</th>
<th>Queueing</th>
<th>Core hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>in situ</td>
<td>none</td>
<td>none</td>
<td>none</td>
<td>193</td>
</tr>
<tr>
<td>off-line</td>
<td>Level I</td>
<td>Level I</td>
<td>full</td>
<td>356</td>
</tr>
<tr>
<td>in situ/off-line</td>
<td>Level II</td>
<td>Level II</td>
<td>partial</td>
<td>135</td>
</tr>
<tr>
<td>in situ/co-sched.</td>
<td>Level II</td>
<td>Level II</td>
<td>none</td>
<td>135</td>
</tr>
<tr>
<td>in situ/in transit</td>
<td>none</td>
<td>Level II</td>
<td>none</td>
<td>n/a</td>
</tr>
</tbody>
</table>
Cosmology Halo Analysis: Center Estimation Using Poisson

- Fast heuristic for finding the MBP center
- For each halo…
  - Superimpose a grid over the (extended) extents of the halo
  - Estimate particle density on the grid using binning
  - Solve Poisson equation for the potential on the grid using FFT
  - Find the grid point with minimum potential
  - Search the neighborhood of the minimum potential grid point for the particle with minimum potential
- There is an overhead for setting up grid and solving Poisson equation, but it is much faster than computing all potentials for large halos
Cosmology Halo Analysis: Exploring Linking Lengths with Dendrograms

- Collaboration with Wathsala W., Timo Bremer, Valerio Pascucci at U. of Utah
- Supports queries with different linking lengths

Illustration of the dendrogram data structure, showing merger of halos as linking length increases

Halos with different linking lengths, found using data-parallel dendrogram implementation

Cosmology Halo Analysis: Data-Parallel Dendrogram Construction

- Three-step algorithm
  - Local step: With cell diagonal length equal to min linking length, all particles in cell are in same halo
  - Compute edges linking cells to neighbors
  - Iteratively merge pairs of cubes in parallel (log total steps) based on edges
- Hybrid implementation: local step on GPU (CUDA), global step on CPU (OpenMP)
- Combined construction plus query time beats ParaView fixed-length halo finder after just a few queries

Illustration of the local (left) and global (right) steps for dendrogram construction
Visualizations for Plasma Surface Interactions

- Collaboration with Wathsala Widanagamaachchi (Utah student), Karl Hammond (Missouri), Shawn Walden (Kitware), Francesca Samsel (UT)
- Creating visualizations of helium bubbles and tungsten cavities from LAMMPS simulations using VTK
- Next step: parallelization to run at scale in-situ
  - Need to parallelize VTK splatter filters
  - May use DIY2 for inter-node parallelism
  - May use VTK-m for intra-node parallelism

XVis

Visualization for the Extreme-Scale Scientific-Computation Ecosystem
VTK-m

- Developing a community standard for enabling visualization and analysis operators to take advantage of multi-core and many-core accelerators, such as GPUs and Xeon Phis

- Unification of three efforts
  - Dax (SNL)
  - EAVL (ORNL)
  - PISTON (LANL)

- Our primary role: developing filters
VTK-m Isosurface

- Isosurface algorithm (for structured 3D grids, using Marching Cubes) implemented using VTK-m worklets and device adapter algorithms
- Variants to compare performance and memory usage (by Robert Maynard at Kitware)
  - fusing cells
  - slicing the data set
  - parallelize over output triangles rather than input voxels
- Will co-evolve with data models and infrastructure
Statistical Filters Implemented

- Basic stats
  - Min, max
  - Mean, median
  - Variance, standard deviation

- Moment calculations
  - First, second, third, fourth
  - Skew, Kurtosis

- Histogram
  - Sparse
  - Dense
Contour Trees (work in progress)

- Collaboration with Hamish Carr (University of Leeds)
- Contour trees encode the topological changes that occur to the contour as the isovalue ranges between its minimum and maximum values
- Contour trees can be used to identify the most “important” isovalues in a data set according to various metrics (e.g., persistence)
- Hamish Carr (and others) have published efficient serial sweep algorithms
  - Compute join and split trees
  - Merge into contour tree
- Currently developing a portable data-parallel implementation
  - Quantized approach
  - Possible integration with Cinema

ASC

Pinion and In-Situ PISTON
Pinion: Portable Data-Parallel Physics

- Developed 1D, 2D, and 3D structured mesh data structures on top of Thrust 1D host/device vectors.
- Mesh data structures provide information about adjacencies and neighbors.
- Implemented 1D, 2D, and 3D volume fraction initialization, Green-Gauss gradient computation, and two-material interface reconstruction for hydro simulation.
- Optimized backend implementations of data parallel primitives (e.g., used SIMD intrinsics and parallel algorithms to improve performance of Thrust scan OpenMP backend on Xeon Phi).
Interface Reconstruction Results

2D circle

3D sphere

2D Interface Reconstruction Performance

- **Grid size:** $1024^2$

- **Graphs:**
  1. **Interface Reconstruction**
     - Time in seconds (log scale)
     - Grid Size (log scale)
     - Devices: CPU, GPU,OMP,MIC

  2. **MIC Strong Scaling**
     - Time in seconds (log scale)
     - Number of Threads (log scale)
     - Algorithms: gradient, initialization, reconstruction
Portable Data-Parallelism in Physics Codes

- VPIC (Vector Particle in Cell) Kinetic Plasma Simulation Code
  - Implemented an in-situ adapter based on Paraview CoProcessing Library (Catalyst)
  - PISTON contour pipeline using VTK-PISTON integration

- CoGL
  - Stand-alone meso-scale simulation code developed as part of the Exascale Co-Design Center for Materials in Extreme Environments
  - Studies pattern formation in ferroelastic materials using the Ginzburg-Landau approach
  - Models cubic-to-tetragonal transitions under dynamic strain loading
  - Simulation code and in-situ viz implemented with PISTON

- ATDM / Next-Generation Code Project
  - Investigation use of Pinion
  - Legion + { Pinion/Thrust, Scout, Kokkos}
Conclusion
Portable Data-Parallelism at LANL
Summary

- **SDAV**: Delivering portable data-parallel visualization and analysis algorithms to science codes
  - Halo and center finding algorithms enabled Q Continuum run
  - Developed and evaluated in-situ and co-scheduling workflows for simulations with large data
  - Implemented variable-linking-length dendrogram halo finder (collab. with Utah)

- **XVis**: Integrating into emerging community-standard VTK-m, and building a library of useful filters
  - Ported isosurface algorithm from PISTON to VTK-m and evaluating variants
  - Implemented statistical filters
  - Developing data-parallel contour tree algorithms

- **ASC**: Using portable data-parallelism for physics computations in codes
  - Implemented 1D, 2D, and 3D interface reconstruction with Pinion meshes
  - Created proxy apps and in-situ data-parallel visualizations
  - Having an impact on ATDM efforts