Meshing the Universe: Integrating Analysis in Cosmological Simulations

Tom Peterka Juliana Kwan Adrian Pope
Hal Finkel Katrin Heitmann Salman Habib
Argonne National Laboratory
Argonne, IL USA
tpeterka@mcs.anl.gov

Jingyuan Wang
University of Tennessee Knoxville
Knoxville, TN USA

George Zagaris
Kitware Inc.
Clifton Park, NY USA

***Abstract*—Mesh tessellations are indispensable tools for analyzing point data because they transform sparse discrete samples into dense continuous functions. Meshing the output of petascale simulations, however, can be as data-intensive as the simulations themselves and often must be executed in parallel on the same supercomputers in order to fit in memory. To date, however, no general-purpose large-scale parallel tessellation tools exist. We present a prototype method for computing such a Voronoi tessellation in situ during cosmological simulations. In principle, similar methods can be applied to other computational geometry problems such as Delaunay tetrahedralizations and convex hulls in other science domains. We demonstrate the utility of our approach as part of an in situ cosmology tools framework that runs various analysis tools at selected time steps, saves results to parallel storage, and includes visualization and further analysis in a widely used visualization package. In the example highlighted in this paper, connected components of Voronoi cells are interrogated to detect and characterize cosmological voids.**

***Keywords*-In situ analysis, computational geometry, Voronoi tessellation.**

I. Introduction

The analysis of petascale simulations is a data-intensive process. Many researchers agree that one way to reduce this data intensity is to integrate more analysis with simulations during their execution. One such approach, in situ analysis, embeds the analysis directly into the execution of the simulation. This approach implies not only that the analysis algorithms need to be parallel but that they must scale to the same size and efficiency as the simulations. In this paper, we investigate one such algorithm for particle data: the computation of a Voronoi mesh tessellation.

Meshes are valuable representations for point data because they convert a sparse point cloud into a continuous field. Such a field can be used to interpolate across cells, compute cell statistics, and identify features. The computation of such a field is indeed data-intensive and is best performed on particles while still in the supercomputer's main memory.

Our contribution is the implementation of a parallel Voronoi tessellation in a prototype library called *tess* and a demonstration of its use in a computational cosmology N-body simulation running on a high-performance supercomputer. A snapshot of the result appears in Figure 1. We show good parallel performance and efficient strong and weak scaling when running in situ with the simulation. Tess is also part of a larger framework of cosmology tools for in situ, coprocessing, and postprocessing visualization and analysis. As part of this end-to-end workflow, we also developed a plugin for a production visualization tool so that our meshes can be viewed and further analyzed. When applied to finding cosmological voids, a minimum volume threshold can be used to partition Voronoi cells into connected components that correspond to voids of irregular, concave geometries. Further analysis of these geometries indicate that our cell distributions agree with underlying physical theory and that cell statistics can be used to characterize aspects of the simulation.

While motivated by cosmology, our tessellation algorithm is general-purpose and is not limited to a particular scientific domain. Other areas that would benefit from our approach include molecular dynamics, computational chemistry, groundwater transport, and materials science.

II. Background

Previous literature has shown the utility of tessellations to identify features in cosmology, although this is the first time that this technique is being used to analyze results in

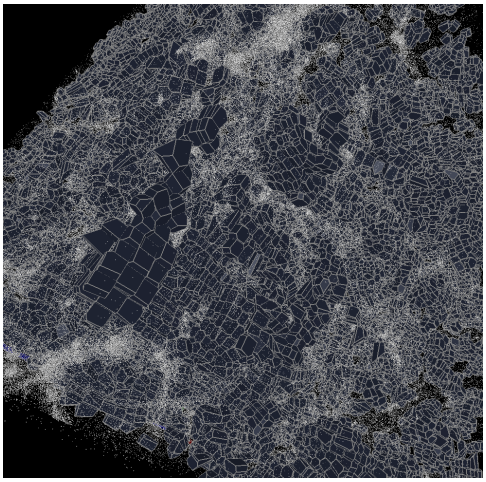

Figure 1. Voronoi tessellation of cosmological simulations reveals regions of irregular low-density voids amid clusters of high-density halos.

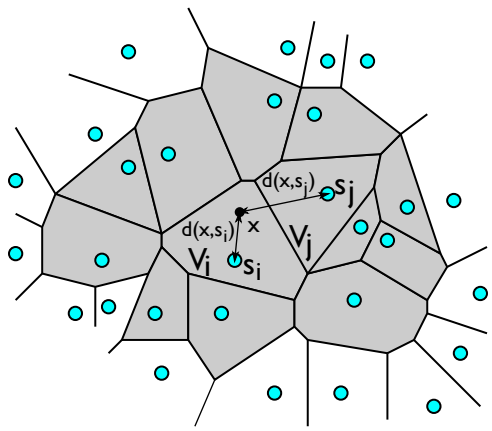

Figure 2. Example of a 2D Voronoi diagram.

situ with a full N-body simulation. One reason may be the lack of a readily available distributed-memory, large-scale parallel tessellation library.

A. Feature Identification in Cosmology

Dark matter is thought to account for over 80% of the matter content of the universe: its existence is inferred from a number of observations involving the cosmic microwave background [1], galactic dynamics [2], and gravitational lensing [3]. Indeed, indirect detections inform us that dark matter is the backbone of large-scale structure in the universe, over which the distribution of gas and galaxies is formed. We can simulate the nonlinear time evolution of the universe to high precision by using dark matter tracer particles and then investigate the large-scale structures that are formed by regions of varying particle density.

Being able to identify features such as halos, voids, filaments, and walls is an important part of understanding structure in the universe. Specifically, this additional shape information allows us to probe beyond the traditional two-point statistics such as power spectrum and correlation. The anisotropic distribution of tracer particles implies that a reconstruction of the density field should ideally be adaptive. Voronoi and Delaunay methods adapt by adjusting the resolution of the reconstruction in response to the local number of particles, since each particle is a Voronoi site or Delaunay vertex. Furthermore, while individual cells are convex, their tessellation into larger structures can produce arbitrary and concave regions.

Van de Weygaert and Schaap [4] discuss the advantages of using tessellation-based methods in cosmology as opposed to a fixed grid. The ZOBOV void finder [5] begins with a Delaunay Tessellation Field Estimator (DTFE) [6]. The Watershed Void Finder [7] attempts to locate voids by using the DTFE algorithm to first reconstruct the density field and then connects local minima at some density threshold. The procedure is analogous to filling a landscape with water, with the valleys acting as voids and the ridges between valleys as filaments and walls. Shandarin et al. [8] combine tessellations with multistream techniques to identify Zel'dovich pancakes for the first time in N-body simulations. Besides being used for analysis, tessellations can be incorporated into the N-body calculations themselves: in hydrodynamics simulations, Springel [9] used a Voronoi tessellation to convert Lagrangian particle behavior to a moving mesh.

B. Computational Geometry Algorithms

Voronoi or Delaunay tessellations are constructed by partitioning the space into cells according to the positions of an input set of points. We focus on the Voronoi tessellation, but the Delaunay is simply its dual; Voronoi cells contain input points in their interiors, whereas Delaunay cells have input points at their vertices.

An example Voronoi diagram containing gray-colored cells is shown in Figure 2. In 2D, those cells are polygons, and in 3D they are polyhedra. Each Voronoi cell V_i is associated with one input point s_i, called the *site* of the cell; sites are the cyan-colored dots in the figure. The site appears somewhere in the interior of the cell, but not necessarily at the centroid.

Formally, each Voronoi cell is defined as

$$V_i = \left\{\mathbf{x} | d(\mathbf{x}, \mathbf{s_i}) < d(\mathbf{x}, \mathbf{s_j})\right\} \forall j \neq i, \tag{1}$$

where $d(\mathbf{x}, \mathbf{s_i})$ is the distance between points $\mathbf{x}$ and $\mathbf{s_i}$. In other words, given a set of 3D input sites, each Voronoi cell is formed by partitioning the 3D space into disjoint, convex regions that are nearer to a particular site than to all other sites.

Several high-quality serial implementations exist for computing such tessellations. Quickhull is a serial algorithm for computing convex hulls, from which Delaunay and Voronoi meshes are derived. Barber et al. [10] report that it is robust in the presence of imprecise floating-point inputs and improves over the performance of earlier algorithms such as Clarkson's [11].

CGAL (Computational Geometry Algorithms Library) [12] is an alternative implementation that can compute a Voronoi diagram given a set of input sites. Unlike Quickhull, a direct implementation of a 3D Voronoi tessellation does not exist; the Delaunay graph must be computed first, from which the dual is calculated to produce Voronoi vertices.

Parallel computational geometry algorithms of limited dimension and concurrency have been researched as well. Rycroft [13] published and implemented a parallel Voronoi algorithm for shared-memory threads in the Voro++ library. Miller and Stout examined parallel algorithms for a convex hull of 2D points, tuned for various network topologies [14]. Dehne et al. [15] presented a parallel 3D convex hull algorithm for distributed-memory architectures in $O(n \log n)$ local computation and one communication phase.

III. Method

Rather than developing a new parallel algorithm from the ground up, our approach is to take an existing serial implementation and parallelize it by combining local computation with communication. The local computation remains unchanged from the serial case, and our performance results will show that the communication is inexpensive. Thus, we can build on the many years of computational geometry research embodied in existing and mature tools. Furthermore, we wrapped our solution in a suite of in situ analysis tools that our team is developing and also coupled the output to a production visualization tool through a custom plugin. The entire software organization is shown in Figure 3.

A. Simulation Code

HACC, or Hardware/Hybrid Accelerated Cosmology Codes, is a simulation framework for computing cosmological models on various supercomputing architectures such as GPUs, Cell Broadband Engines, and multicore CPUs. It solves the Vlasov-Poisson equation, which evolves the phase-space distribution function for particles in an expanding universe. The solution method is based on N-body techniques melding three kinds of force solvers: direct particle summation, tree/multipole expansion, and spectral particle-mesh. HACC runs were the first scientific simulations to attain a sustained performance of greater than 10 PFlops on any architecture [16].

Large simulation sizes are required in order to compute even a fraction of the observable sky. Hundreds of billions to trillions of particles are required to track galaxy-scale mass distributions and to predict observables, for example, the matter density fluctuation power spectrum, correctly [17]. The spatial dynamic range set by the resolution requirements is $10^6 : 1$. Such simulations can easily generate data sizes over 100 terabytes to several petabytes in a single run.

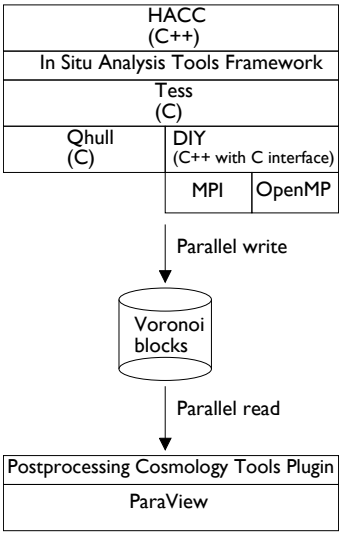

Figure 3. Overall software organization.

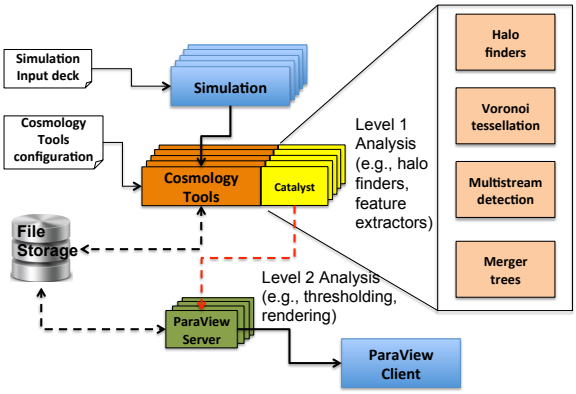

Figure 4. The cosmology in situ analysis framework provides the ability to apply various in situ analyses at selected time steps in the simulation. Analysis filters include halo finding, multistream feature classification, feature tracking, and Voronoi tessellation. The framework also connects to run-time or postprocessing visualization tools.

B. In Situ Analysis Framework

Our team is developing an entire suite of in situ analysis tools for cosmology applications, and tess is the first tool to be implemented in this framework. The overall structure of this framework is shown in Figure 4. Other tools that either exist already or are currently being developed include halo finders [18], feature classifiers, and feature trackers. We will be incorporating these tools under the same common analysis interface. Various tools will be turned on through the configuration file for the simulation, and the frequency of their execution will also be configurable. Upon each time step, the input particles will be sent to the appropriate analysis tools.

Output from the analysis will be available either at run time or for postprocessing. In the former case, a ParaView server can be launched and connected to the simulation through a tool called Catalyst. In the latter case, results of the in situ analysis are written to a parallel file system and later retrieved. This is the mode that we used in this paper.

C. Parallel Voronoi Tessellation

In general, our preferred approach to developing high-performance analysis solutions is to parallelize existing algorithms through a combination of serial operations and parallel communication. In this way, we can select a proven implementation of the serial algorithm and parallelize it over a data-parallel infrastructure that we wrote and maintain.

The result, tess, consists of the following main features.

- Standalone and in situ modes of operation
- Neighborhood particle ghost zone exchange
- Local Voronoi cell computation
- Identification of complete cells

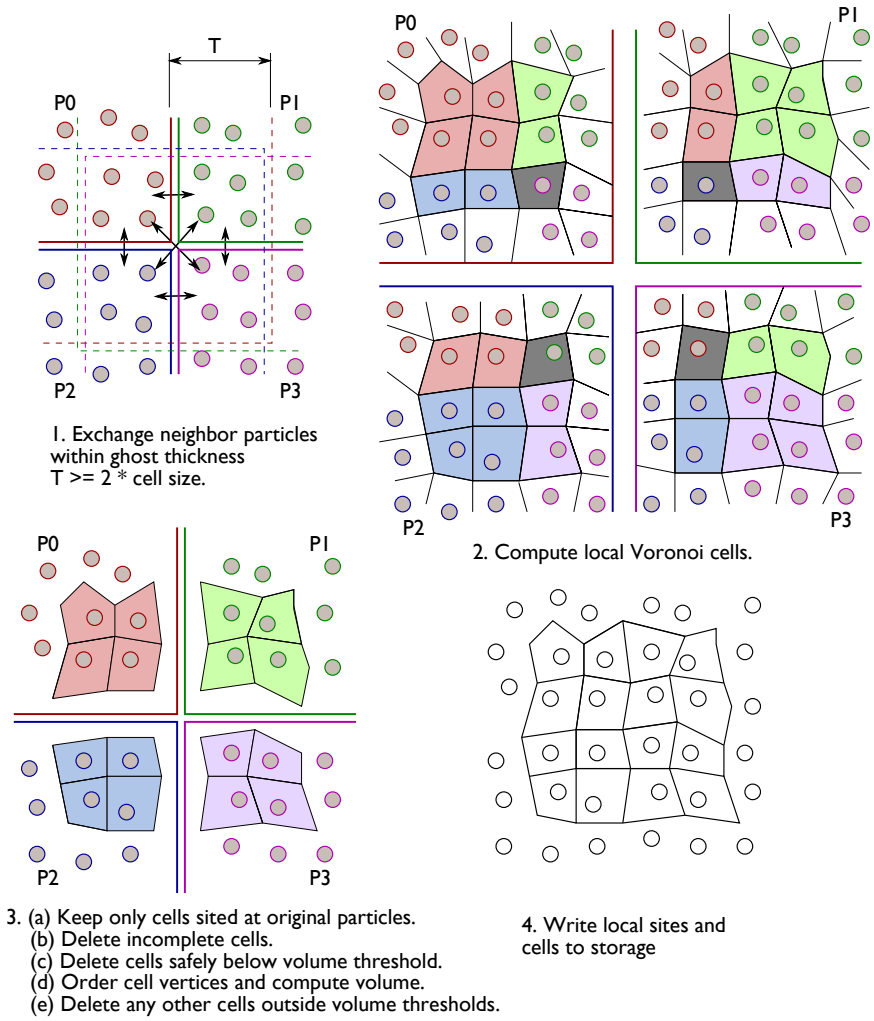

Figure 5. Overview of parallel algorithm by using an example with four processes. Particles and Voronoi cells are colored according to the process where they originated prior to exchanging ghost layer. Gray-colored cells are exchanged by multiple processes.

- Early volume threshold culling
- Local convex hull computation for faces, areas, volumes
- Parallel writing Voronoi cells to storage

We selected the Quickhull algorithm for the local computation because of its widespread use, documented performance, and numerical stability. This algorithm is implemented in an open-source package, Qhull,[1] a set of standalone command-line programs for computing convex hulls, Delaunay triangulations, and Voronoi tessellations. These programs call an underlying `libqhull` library. The standalone programs are well-documented, but the library API is not. By examining its printing routines, however, we were able to parse Qhull's data structure and redirect its output to our data model. No changes were made to Qhull itself.

For data parallelism, we took advantage of the features available in a library called DIY [19] that provides configurable data partitioning, scalable data exchange, and efficient parallel I/O. DIY is initialized with information from HACC about its block decomposition and neighborhood connectivity. Then, DIY performs data movement and communication on behalf of tess. Developing tess required two new features to be added to DIY: periodic boundary conditions and destination neighbor identification based on proximity to a target point (see Section III-C1).

The steps in our algorithm are illustrated in Figure 5. The first step is to exchange particles in a given size ghost region. Details of the neighborhood exchange are explained in Section III-C1. We rely on an estimate of the average cell size and a block size that is several times larger than that. In HACC, the average cell size is on the order of the initial particle spacing, and block size is approximately ten times that distance.

The local Voronoi tessellation is then computed by using

[1] http://www.qhull.org

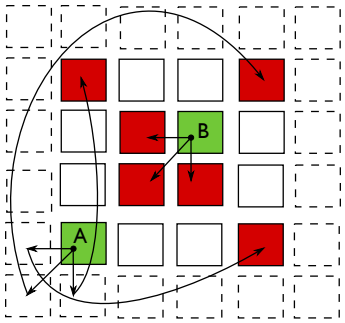

Figure 6. Neighborhood communication with periodic boundary conditions and neighbors that are near enough to a target point. Green blocks denote sources, and red blocks denote destinations. Particle A is at the domain boundary and is sent to the virtual neighbors marked with dashed lines actually on the other side of the domain. The coordinates of the particle are transformed accordingly. Particle B is sent to actual neighbors near enough to receive it.

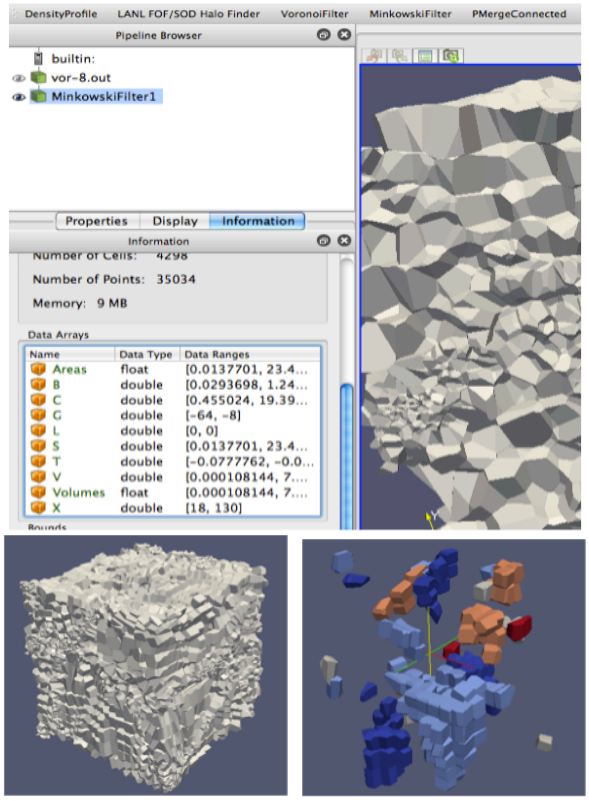

Figure 7. The cosmology tools plugin for ParaView provides interactive feature exploration of previously computed Voronoi tessellations. It includes a parallel reader, threshold filtering, connected component labeling, and computing Minkowski functionals.

the Quickhull algorithm. The Voronoi cells are each examined to see whether they should be removed. Incomplete cells are eliminated; these cells are not closed because they lack particles surrounding them on all sides.

The neighborhood exchange is bidirectional. This ensures that all local blocks have an adequate ghost region of particles needed to compute correct and complete Voronoi cells in their interior. A bidirectional exchange, however, results in duplicated cells across blocks. These are apparent in the upper right quadrant of Figure 5. In order to resolve such duplication, each process keeps the cells that have sites in its original particle set and deletes the cells owned by other processes.

As Section IV shows, a large fraction of cells have small volume and may be of little interest, especially for void finding. These will be eliminated once their volume is accurately computed, but we perform a conservative quick estimate of the cell volume and eliminate early as many cells as possible by keeping only those cells whose distance between any two vertices exceeds the diameter of a circumscribing sphere of the threshold volume.

We then compute the convex hull of the vertices in the Voronoi cell. Although Voronoi vertices are convex by definition, this step orders the vertices into faces and computes the volume and surface area of the cell. Afterwards, we recheck the volume and further cull cells below the volume threshold, writing the remaining data in parallel to storage according to the data model in Section III-C2.

1) Neighborhood Particle Exchange: Two new communication patterns were added to DIY as a result of this research: periodic boundary neighbors and targeted particle exchange. HACC's nearest-neighbor connectivity includes periodic boundaries, meaning that blocks at one edge of the overall domain have neighbors at the opposite edge of the domain. Targeted particle exchange refers to the fact that only those neighbors in the ghost zone distance from a particle will be destined to receive the particle during the exchange.

These ideas are diagrammed in Figure 6. Particles originate in the green blocks and follow the indicated paths to the red blocks. Particles in the ghost zone distance of a block boundary are exchanged with all neighbors, including periodic boundary neighbors, that share that boundary. If the neighbor is indeed a periodic boundary neighbor, the particle is translated in each of the periodic dimensions to the other side of the domain. Identifying which destinations are periodic and providing a callback to a user-specified transformation for those destinations were two details that were added to DIY as part of this work.

2) Analysis Data Model: In tess, each process maintains a data structure for the block of Voronoi cells local to it; these blocks are written in parallel to a single file by the DIY library. Each block contains a conventional unstructured mesh data model. Vertices are listed once, and integer indices connect vertices into faces and cells. Original particle locations are also saved, along with cell volumes, areas, and block extents.

On average, Voronoi cells in HACC simulations contain 15 faces per cell and 5 vertices per face. Each cell consists

Table I
PARALLEL ACCURACY

Ghost Size	Cells in Serial Version	Blocks	Matching Cells	% Accuracy
0	210181	2	201952	96.08
		4	196803	93.64
		8	192140	91.42
1		2	209367	99.61
		4	208564	99.23
		8	207024	98.50
2		2	210176	99.99
		4	210155	99.98
		8	210012	99.92
3		2	210181	100.00
		4	210180	99.99
		8	210180	99.99
4		2	210181	100.00
		4	210181	100.00
		8	210181	100.00

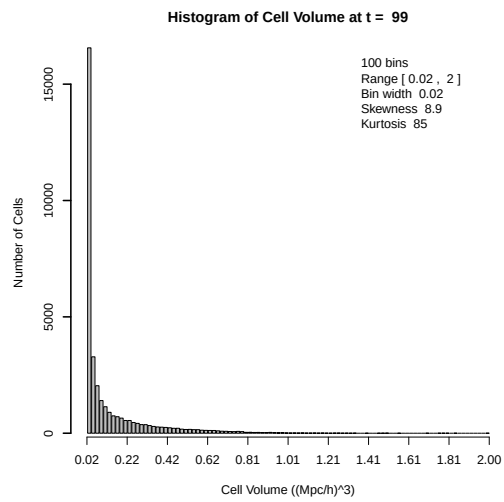

Figure 8. Histogram of cell volume distribution in small-scale test.

of approximately 35 total vertices, since vertices are shared between at least two faces in the same cell. Vertices are also shared among five cells on average; hence, approximately seven new Voronoi vertices are added for each new cell in a full tessellation.

The total data size of a full tessellation is approximately 450 bytes per particle. As we show in Section IV, a large fraction of cells have insignificant volume with respect to voids. When we choose to eliminate these, as we often do, the data size is reduced to approximately 100 bytes per particle. By comparison, a HACC checkpoint that saves only particle data uses 40 bytes per particle.

Of tess's total output size, approximately 7% is used to store floating-point geometry of vertices, particles, volumes, and areas. The remaining 93% is used for connectivity of the mesh. A more efficient data structure for general polyhedral grids has been published by Muigg et al. [20], and we are investigating its use.

D. Postprocessing Tools

The visualization and subsequent analysis of our tessellation are performed in postprocessing with the help of a plugin that we developed for ParaView. Figure 7 shows a snapshot of the tool, which provides four main functions: parallel reading the tess output file, threshold filtering, parallel connected component labeling, and computing of Minkowski functionals on the connected components.

The Minkowski functionals are a set of statistics recognized by cosmologists for classifying structures [21]. The four basic functionals are volume, surface area, extrinsic curvature, and genus. From these basic quantities, three more derived metrics are computed: thickness, breadth, and length. Figure 7 shows the Minkowski filter in use, and the values of these metrics are shown for the connected components in the lower right portion of the figure. Such metrics are used to compare different simulations or different initial conditions, to study percolation theory, and to characterize the shapes of large-scale structures such as cosmological voids.

IV. RESULTS

Our small-scale tests were run on a Linux desktop workstation with a quad-core Intel i7 processor capable of running eight hardware threads with 12 GB of RAM. Our full-scale tests were run on *Intrepid*, a 557-teraflop IBM Blue Gene/P (BG/P) supercomputer operated by the Argonne Leadership Computing Facility (ALCF) at Argonne National Laboratory. Tests were run in symmetric multiprocessor mode (one MPI process per node) to maximize available memory per process; HACC and tess are memory-size bound. The libraries and simulation were all compiled with the IBM `xlcxx_r` compiler and by using `-O3 -qarch=450d -qtune=450` optimizations.

In our HACC configuration, we varied the number of particles from 32^3 to 1024^3. The particles are initialized on a grid of the same number of grid points (*ng*) per dimension as number of particles (*np*). The physical size of the simulation box is also the same as *ng* and *np*; hence, particles begin spaced 1 Mpc/h apart in each dimension, where h is the dimensionless Hubble parameter.

A. Parallel Accuracy Study

Our first test compared the accuracy of our parallel algorithm with a serial version. Table I shows the accuracy of the parallel version with particles distributed among multiple blocks compared with a serial version with all particles in one block. In this test, 64^3 particles were computed for 100 time steps of the HACC simulation. In our parallel tessellation algorithm, we varied the ghost zone size that was being exchanged, where this distance is measured in the same units of distance as the simulation. The table shows

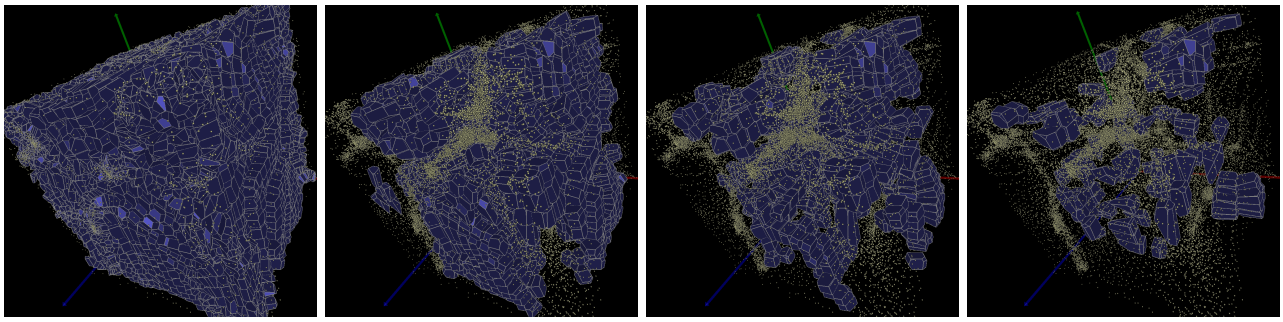

Figure 9. Culling cells below a minimum volume threshold reveals connected components of cells, which constitute voids. Left to right: original cells ranging from 0.0001 to 2.005 $(Mpc/h)^3$, and progressing through minimum volume thresholds of 0.0, 0.5, 0.75, and 1.0 $(Mpc/h)^3$, respectively.

that until the ghost zone is large enough, accuracy decreases with increasing block count. The reason is that the incorrect cells occur at block boundaries, and more blocks produce more errors. Without any ghost zone exchange (ghost size 0), eight blocks alone reduce accuracy to 91%.

As the ghost zone size increases, however, the overall accuracy improves, until it reaches 100% when the ghost zone is sufficient. This is the case at a ghost size of 4 units in this example. The ghost size parameter is provided by the user. This approach is acceptable because cosmologists have a good understanding of the largest spacing between particles through experience and smaller scale runs. Nonetheless, we are studying methods to determine the ghost size automatically in order to make the algorithm more robust and easier to use. Moreover, we are investigating the tradeoff between ghost zone size, neighborhood exchange time, and accuracy. For example, it may be desirable to exchange fewer particles with a smaller ghost zone if the reduction in accuracy is insignificant.

B. Characteristic Volume Distribution and Threshold

Our second test consisted of a 32^3 particle simulation run on a Linux workstation up to 32 processes. Particles were evolved for 100 time steps, and the Voronoi tessellation was computed at the end of the last time step. Several characteristics of the Voronoi tessellation applicable to larger scale were uncovered at this small scale.

The distribution of Voronoi cell volume after 100 time steps appears in Figure 8. The distribution is skewed toward zero, with the majority of cells at the left side and a long thin tail at the right. In fact, 75% of the cells are in the smallest 10% of the volume range.

This characteristic distribution holds in all our larger-scale runs and indicates that a simple threshold operator can dramatically reduce the number of cells. Shandarin et al. also describe voids in terms of thresholds [22]. Voids by definition are regions of low particle density found in the long tail at the right of the histogram. When studying voids, we found that a 10% volume threshold is a reasonable value that eliminates many small, uninteresting cells while safely retaining all the cells that contribute to voids.

Thresholding serves a second purpose besides reducing output size. We can further filter cells during either computation or rendering to reveal voids more clearly. Figure 9 shows a sequence of progressive thresholding during rendering. From left to right, culling cells below an increasing volume threshold reveals connected components of cells that correspond to voids.

In the left image, the difference in cell size is visible, but the connection of cells into larger structures is not. The other images reveal a small number (approximately 7-10) distinct connected components, or voids. We are able to select the volume threshold range either in situ or during postprocessing as explained in III-D. In the latter case, we can generate logical connectivity and connected component labeling. In the former case, the connectivity is purely visual; but in the future, we plan to label connected components automatically in situ as well.

C. Performance and Scalability

Full timing results for a range of problem and system sizes appear in Table II, which shows the performance running one Voronoi tessellation in situ after running a number of simulation time steps on BG/P. The fourth column of the table is the total time, which is the sum of the simulation time in column five and the tessellation time in column six. The tessellation time is further itemized in columns seven through nine into particle exchange time, Voronoi computation time, and output time. This test was run with culling the smallest 10% volume range of the Voronoi cells, resulting in the file sizes shown in the last column.

The tessellation time is 1-10% of the total run time, depending on the number of time steps executed before calling tess. The cost of tessellation compared with simulation is reasonable, especially considering that HACC takes longer to compute later time steps. The particle exchange time is negligible; this is a testament to the efficiency of the neighborhood exchange algorithm in DIY. The output time is also minimal, although the I/O performance begins to

Table II
PERFORMANCE DATA

No. of Particles	Time Steps	Processes	Total Time (s)	Simulation Time (s)	Tessellation Total Time (s)	Particle Exchange Time (s)	Voronoi Computation Time (s)	Output Time (s)	Output Size (GB)
128^3	100	128	1862	1809	53	1	50	2	0.3
		256	1354	1322	32	1	29	2	
		512	1116	1096	20	1	17	2	
		1024	745	729	16	1	12	3	
256^3	100	512	3090	3016	74	2	69	3	1.7
		1024	2391	2346	45	2	39	4	
		2048	1861	1830	32	2	26	4	
		4096	1334	1305	29	2	15	12	
512^3	50	2048	3852	3684	167	4	157	6	14
		4096	2008	1918	89	3	77	9	
		8192	1784	1722	62	3	48	11	
		16384	1406	1344	61	2	32	27	
1024^3	25	8192	2331	2119	212	6	186	20	101
		16384	1446	1289	157	4	113	40	

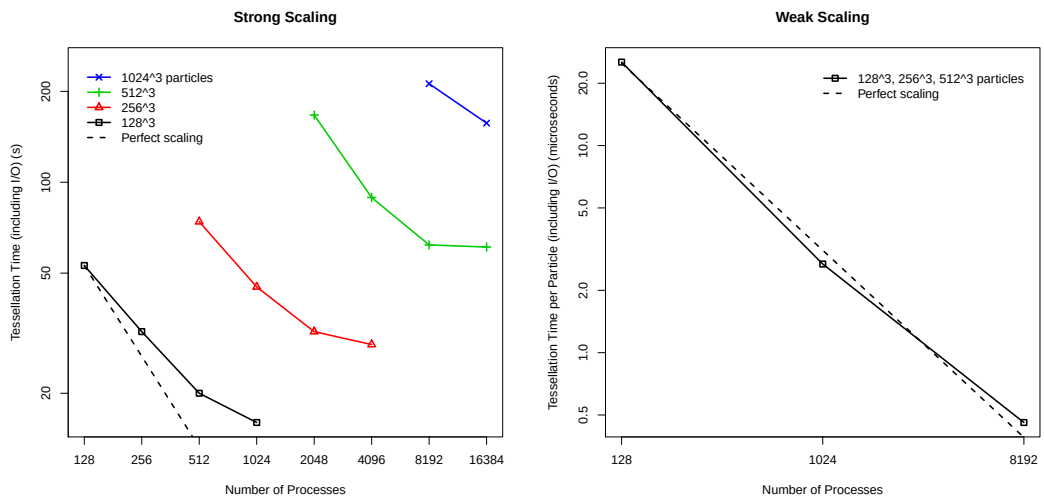

Figure 10. Strong scaling (left) and weak scaling (right) are plotted on a log-log scale. Weak scaling time is normalized by the number of particles, resulting in a downward path whose slope can be compared with the ideal. All graphs represent the total tessellation time, including the time to write the result to storage. The strong scaling efficiency is 41%, and the weak scaling efficiency is 86%.

wane at larger problem sizes and higher process counts. We continue to work to improve our I/O algorithm. The most expensive component of the tessellation is the serial Voronoi computation, although this component scales well with increasing process count. Such scalability is typical of the serial component of parallel analysis algorithms.

Strong and weak scaling curves are plotted on a log-log scale in Figure 10. These graphs represent the total tessellation time, including writing the results to storage. Strong scaling is shown for each of the four problem sizes. Weak scaling is normalized by the number of particles, so that the curve slopes linearly downward. Strong scaling efficiency for tess is 30-40%, and weak scaling efficiency is 86%. As column five of Table II shows, these efficiencies are consistent with the scalability of the original simulation.

D. Time-Varying Void Evolution

Since tess can generate tessellations at any number of time steps in the simulation, we can study the evolution of voids over time. We tested the time-varying capability by producing outputs at every ten time steps of a simulation with 100 time steps. This test was at small scale, on 32^3 particles on our workstation; the results are shown in Figure 11. The top row shows images of the tessellation at time steps 11, 21, and 31, while the bottom row shows the cell density contrast δ. All particles have unit mass; hence, cell density d is simply the reciprocal of cell volume. Density contrast δ is the difference between d and the mean density μ_d normalized by μ_d, and much of the cosmology theory is developed in terms of δ.

$$\delta = (d - \mu_d)/\mu_d \tag{2}$$

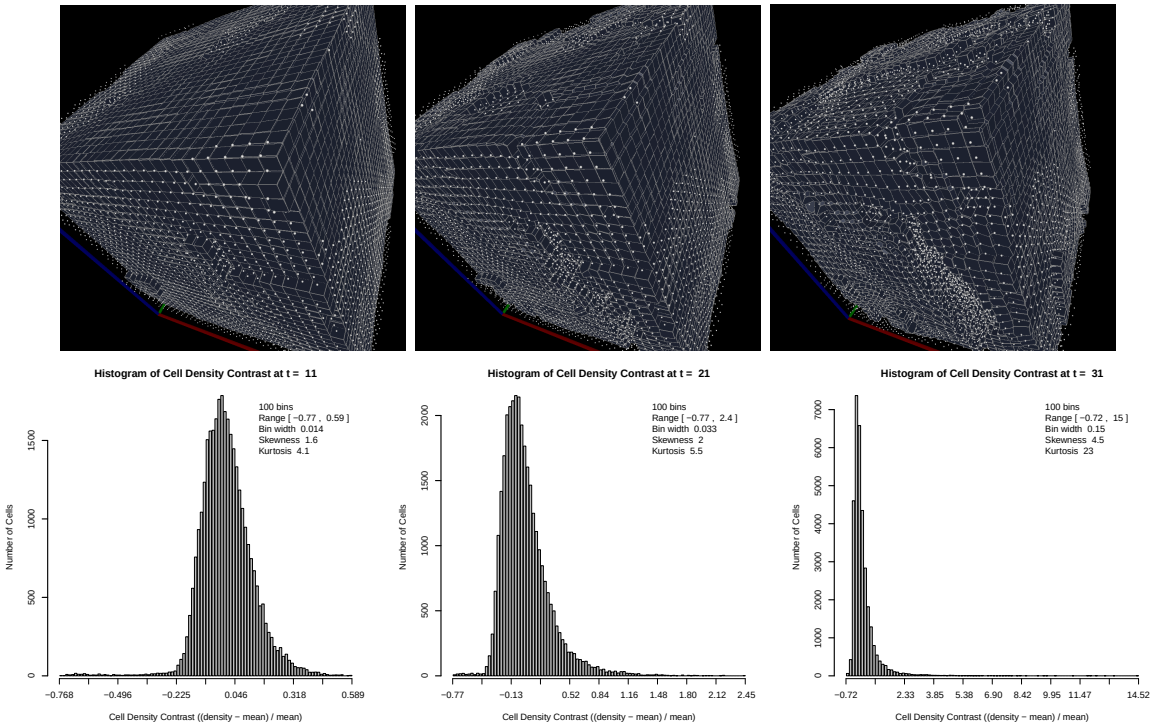

Figure 11. Evolving Voronoi cells at three time steps during the simulation (top) and corresponding density contrast distribution (bottom).

The early time steps begin with a normally distributed cell size and shape because particles begin their evolution randomly displaced from regular grid points. As time progresses, the range of δ expands. The kurtosis increases as the distributions become more pointed, and skewness increases as well. The images in the top row confirm this behavior. These statistics correspond to the breakdown of perturbation theory governing the physics; in the future, we will investigate whether such summary statistics can be used as simple indicators of such changes in physical behavior. For example, as particles coalesce into high-density halos, the quantity of small cells increases; at the same time, large cells that are diminishing in number are increasing in size.

V. Summary

We presented a solution for computing scalable parallel tessellations and demonstrated its in situ application to cosmological computations. This produced statistical summaries of volume and density distributions and identification of large-scale structures such as voids. Performance was benchmarked with good scalability, and our time-varying results were consistent with published findings.

We showed how our approach is part of a growing set of in situ cosmology analysis tools and how the results are postprocessed to investigate void statistics. This has been proposed as a means of distinguishing between competing cosmological models but has never been verified with a full nonlinear N-body solution.

It would also be interesting to perform these reconstructions with halos as Voronoi sites instead of directly by using the tracer particles, since halos can be matched to direct observables such as galaxies. This work would involve smaller, prefiltered data and a combination of in situ analysis techniques from our common tools framework.

Of course, improvements could be made to the algorithm itself, such determining the ghost size automatically. We are also considering moving more postprocessing tasks in situ, such as connected component labeling, Minkowski functionaks, and histogram summary statistics. We will also look to tracking temporal evolution of connected components by using the feature tree method of Chen et al. [23]. Another use of the Voronoi tessellation that we are considering is to augment the output of particle positions with the cell volume or density at each site as an indication of the density of the region surrounding each particle. Such information could be used to guide structure detection, sampling, and other density-based operations in the future.

Acknowledgment

We gratefully acknowledge the use of the resources of the Argonne Leadership Computing Facility at Argonne National Laboratory. This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357. Work is also supported by the DOE Office

of Science, Advanced Scientific Computing Research award No. DE-FC02-06ER25777, program manager Lucy Nowell.

REFERENCES

[1] N. Jarosik, C. L. Bennett, J. Dunkley, B. Gold, M. R. Greason, M. Halpern, R. S. Hill, G. Hinshaw, A. Kogut, E. Komatsu, D. Larson, M. Limon, S. S. Meyer, M. R. Nolta, N. Odegard, L. Page, K. M. Smith, D. N. Spergel, G. S. Tucker, J. L. Weiland, E. Wollack, and E. L. Wright, "Seven-year Wilkinson Microwave Anisotropy Probe (WMAP) Observations: Sky Maps, Systematic Errors, and Basic Results," *Astrophysical Journal Supplement*, vol. 192, p. 14, Feb. 2011.

[2] V. C. Rubin, D. Burstein, W. K. Ford, Jr., and N. Thonnard, "Rotation Velocities of 16 SA Galaxies and a Comparison of Sa, Sb, and Sc Rotation Properties," *Astrophysical Journal*, vol. 289, pp. 81–98, Feb. 1985.

[3] M. Markevitch, A. H. Gonzalez, L. David, A. Vikhlinin, S. Murray, W. Forman, C. Jones, and W. Tucker, "A Textbook Example of a Bow Shock in the Merging Galaxy Cluster 1E 0657-56," *Astrophysics Journal Letters*, vol. 567, pp. L27–L31, March 2002.

[4] R. van de Weygaert and W. Schaap, "The Cosmic Web: Geometric Analysis," *ArXiv e-prints*, Aug. 2007.

[5] M. C. Neyrinck, "ZOBOV: a Parameter-Free Void-Finding Algorithm," *Monthly Notices of the Royal Astronomical Society*, vol. 386, pp. 2101–2109, June 2008.

[6] W. E. Schaap, *DTFE: The Delaunay Tesselation Field Estimator*, University of Groningen, The Netherlands, 2007, Ph.D. Dissertation.

[7] E. Platen, R. van de Weygaert, and B. J. T. Jones, "A Cosmic Watershed: The WVF Void Detection Technique," *Monthy Notices of the Royal Astronomical Society*, vol. 380, pp. 551–570, Sept. 2007.

[8] S. Shandarin, S. Habib, and K. Heitmann, "Cosmic Web, Multistream Flows, and Tessellations," *Physical Review D*, vol. 85, p. 083005, Apr 2012. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevD.85.083005

[9] V. Springel, "Hydrodynamic Simulations on a Moving Voronoi Mesh," *ArXiv e-prints*, Sept. 2011.

[10] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The Quickhull Algorithm for Convex Hulls," *ACM Trans. Math. Softw.*, vol. 22, pp. 469–483, Dec. 1996. [Online]. Available: http://doi.acm.org/10.1145/235815.235821

[11] K. L. Clarkson, "Applications of Random Sampling in Computational Geometry, II," in *Proceedings of the fourth annual symposium on Computational geometry*, ser. SCG '88. New York, NY, USA: ACM, 1988, pp. 1–11. [Online]. Available: http://doi.acm.org/10.1145/73393.73394

[12] A. Fabri and S. Pion, "CGAL: the Computational Geometry Algorithms Library," in *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ser. GIS '09. New York, NY, USA: ACM, 2009, pp. 538–539. [Online]. Available: http://doi.acm.org/10.1145/1653771.1653865

[13] C. Rycroft, "Voro++: A Three-dimensional Voronoi Cell Library in C++," Tech. Rep., 2009, http://www.osti.gov/energycitations/servlets/purl/946741-A8FxbI/946741.pdf.

[14] R. Miller and Q. F. Stout, "Efficient Parallel Convex Hull Algorithms," *IEEE Trans. Comput.*, vol. 37, no. 12, pp. 1605–1618, Dec. 1988.

[15] F. Dehne, X. Deng, P. Dymond, A. Fabri, and A. A. Khokhar, "A Randomized Parallel 3D Convex Hull Algorithm for Coarse Grained Multicomputers," in *Proceedings of the Seventh Annual ACM Symposium on Parallel Algorithms and Architectures*, ser. SPAA '95. New York, NY, USA: ACM, 1995, pp. 27–33. [Online]. Available: http://doi.acm.org/10.1145/215399.215410

[16] S. Habib, V. Morozov, H. Finkel, A. Pope, K. Heitmann, K. Kumaran, T. Peterka, J. Insley, D. Daniel, P. Fasel, N. Frontiere, and Z. Lukic, "The Universe at Extreme Scale: Multi-Petaflop Sky Simulation on the BG/Q," *ArXiv e-prints*, Nov. 2012.

[17] K. Heitmann, M. White, C. Wagner, S. Habib, and D. Higdon, "The Coyote Universe. I. Precision Determination of the Nonlinear Matter Power Spectrum," *Astrophysical Journal*, vol. 715, pp. 104–121, May 2010.

[18] J. Woodring, K. Heitmann, J. Ahrens, P. Fasel, C.-H. Hsu, S. Habib, and A. Pope, "Analyzing and Visualizing Cosmological Simulations with ParaView," no. arXiv:1010.6128. LA-UR-10-06301, Nov. 2010.

[19] T. Peterka, R. Ross, W. Kendall, A. Gyulassy, V. Pascucci, H.-W. Shen, T.-Y. Lee, and A. Chaudhuri, "Scalable Parallel Building Blocks for Custom Data Analysis," in *Proceedings of the 2011 IEEE Large Data Analysis and Visualization Symposium LDAV'11*, Providence, RI, 2011.

[20] P. Muigg, M. Hadwiger, H. Doleisch, and E. Groller, "Interactive Volume Visualization of General Polyhedral Grids," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, pp. 2115–2124, 2011.

[21] J. V. Sheth, V. Sahni, S. F. Shandarin, and B. S. Sathyaprakash, "Measuring the Geometry and Topology of Large Scale Structure Using SURFGEN: Methodology and Preliminary Results," *Monthly Notices of the Royal Astronomical Society*, vol. 343, no. astro-ph/0210136, p. 22. 26 p, Oct. 2002.

[22] S. Shandarin, H. A. Feldman, K. Heitmann, and S. Habib, "Shapes and Sizes of Voids in the Lambda Cold Dark Matter Universe: Excursion Set Approach," *Monthly Notices of the Royal Astronomical Society*, vol. 367, no. 4, pp. 1629–1640, 2006. [Online]. Available: http://dx.doi.org/10.1111/j.1365-2966.2006.10062.x

[23] J. Chen, D. Silver, and L. Jiang, "The Feature Tree: Visualizing Feature Tracking in Distributed AMR Datasets," in *Proceedings of the 2003 IEEE Symposium on Parallel and Large-Data Visualization and Graphics*, ser. PVG '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 14–. [Online]. Available: http://dx.doi.org/10.1109/PVGS.2003.1249048