Big-Data Management Use-Case: A Cloud Service for Creating and Analyzing Galactic Merger Trees

Sarah Loebman[1], Jennifer Ortiz[2], Lee Lee Choo[2],
Laurel Orr[2], Lauren Anderson[3], Daniel Halperin[2],
Magdalena Balazinska[2], Thomas Quinn[3], and Fabio Governato[3]
Astronomy Department, [1]University of Michigan, [3]University of Washington
Department of Computer Science & Engineering, [2]University of Washington

ABSTRACT

We present the motivation, design, implementation, and preliminary evaluation for a service that enables astronomers to study the growth history of galaxies by following their 'merger trees' in large-scale astrophysical simulations. The service uses the Myria parallel data management system as back-end and the D3 data visualization library within its graphical front-end. We demonstrate the service at the workshop on a $\sim$5 TB dataset.

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems—Distributed databases, Query processing, Relational databases

Keywords

Myria; parallel data management; Cloud service; astronomy

1. INTRODUCTION

In the age of high-performance computing, scientists regularly use large-scale simulations to model the behavior of complex, natural systems [10]. As the volume of the data generated with the state-of-the-art code grows, it *is increasingly difficult to analyze that data via traditional scientific tools* (*e.g.*, IDL & Python) because of limitations in I/O, memory and CPU resources [9].

At the same time, we see the development of novel data management systems (*e.g.*, GraphLab and Shark), and Cloud *services* (*e.g.*, Amazon Elastic MapReduce and Google BigQuery) that can facilitate scientific data analysis at a large scale. However, it is often difficult to express scientific questions using the data models and query languages provided by such tools. Additionally, to achieve high-performance, users must physically tune their systems: the simplest ways to articulate queries and organize data on disk often lead to poor performance when processing terabytes of data.

One approach to enabling this science is to develop specialized services that facilitate specific types of analysis on specific types of data. This approach is practical when datasets can be hosted centrally and when large groups of users need to perform similar analyses. This is relevant to the field of astronomy, in which large-scale, shared scientific datasets are publicly available to and used by the entire scientific community (*e.g.*, [12]).

DanaC'14, June 22 2014, Snowbird, UT, USA.
Copyright 2014 ACM 978-1-4503-2997-2/14/06 ...$15.00.
http://dx.doi.org/10.1145/2627770.2627774.

In this paper, we present a use-case centered on building MyMergerTree, a vertical data-analysis service designed to meet the needs of astronomers who work with large-scale cosmological simulations. Cosmological simulations track the evolution of galaxies such as our own over a span of 14 billions years from the Big Bang to the present day. The standard model of galaxy formation involves the hierarchical assembly of galaxies through merging of smaller galaxies. Any given galaxy will have a unique merger history; scientists would like to know *how much do these merger histories vary depending on a galaxy's final mass, proximity to other galaxies, or other factors?* MyMergerTree targets this active area of research.

The cosmological simulation in MyMergerTree was produced by the University of Washington N-body shop. It uses ChaNGa [8], a fully hydrodynamic+N-body tree code, to track the mass distribution of the Universe using $\sim$2 billion particles. In this simulation, 46 snapshots were taken to record the position, velocity, and internal properties of all the particles at different moments in time. The amount of data recorded is immense: at 60 bytes of data per particle, each snapshot comprises more than 100 GB of data, resulting in a $\sim$5 TB dataset. Since reconstructing the merger histories of galaxies requires joining and aggregating the 5 TB of particle data across all simulation snapshots, computing these merger trees efficiently and quickly is a major challenge, even for Big Data systems.

In this paper, we present the design, implementation, and preliminary evaluation of the MyMergerTree service for the interactive analysis of galactic merger trees that form in cosmological simulations. We begin by describing the data and necessary analyses in Section 2. To facilitate these analyses at scale, MyMergerTree uses a new Big Data management and analysis service called Myria.[1] We adopt Myria for this problem because we are actively developing it; the merger tree use case is an excellent testbed for the efficiency of the Myria engine. Myria is an elastically-scalable, shared-nothing system with advanced features including native support for iterative computations. Our analysis is expressed using declarative queries that Myria executes in parallel on the cluster. To achieve high-performance, however, we need to carefully tune the Myria system as we describe in Section 3.

We also describe the astronomer's interface to MyMergerTree, which enables interactive analysis of these data through a web browser. We develop a simple interface that enables users to configure the merger tree analysis to suit their specific needs, without needing to author or ever seeing SQL queries. This component builds on three important technologies to provide a useful, science-oriented lens on the data. First, in addition to providing massive computing power, Myria offers a RESTful cloud service that can

[1]Myria was developed by the database group at the University of Washington over the past year and a half and will be demonstrated at SIGMOD 2014 [7].

be easily used for 'mash-ups.' Second, we extended the University of Washington's web-based astronomy data visualization toolkit ASCOT [11] to generate queries based on user input, issue them to Myria, and fetch the results. Third, MyMergerTree displays the analysis results using the data visualization library D3 (screenshot in Figure 2) [3]. We describe this front-end interface in Section 4.

We finish our presentation of MyMergerTree with preliminary performance numbers and initial user feedback in Section 5. We discuss related work in Section 6. We conclude in Section 7 that such vertical services are valuable to users but challenging to implement.

2. GALACTIC MERGER TREES: DATA MODEL AND ANALYSIS

2.1 Data Model

The simulation data produced by ChaNGa takes the form of a series of 46 snapshots. Each snapshot captures the state of the Universe, as represented by a set of particles, at a specific moment in time during the simulation. Snapshots are saved in the simulation-specific NChilada File Format, which contains 30 attributes per particle, including position, mass, velocity, and temperature. The original data are provided in separate files for each snapshot. We ingest these snapshots into Myria and create a view to represent the entire simulation as a single relation by UNIONing all the data and appending a `time` column to identify the source snapshot.[2]

Creating a merger tree uses five attributes in the simulation data: the snapshot `time`, a particle's unique identifier `iOrd`, and the particle's `type`, `mass`, and `grpID`. For convenience we create a second view, `ParticleTable`, to represent these attributes, though the remaining features are maintained for future analyses:

```
ParticleTable(iOrd, type, mass, time, grpID)
```

A particle's `iOrd` tracks an individual particle across snapshots. The `type` refers to its material content (gas, stars or dark matter). Depending on the science involved, an astronomer may indicate which particle types should be used to construct a merger tree. Every particle has a `mass` attribute, which refers to how much matter the particle represents in the Universe.

For each snapshot, a stand-alone clustering application identifies groups of particles as galaxies based on spatial locality; each particle it tagged with a `grpID` corresponding to the galaxy it is associated with. Snapshots are clustered independently from one another, so there is no direct correspondence between `grpID` values across time; instead the combination (`time`, `grpID`) identifies a galaxy uniquely. We can determine the overall mass of a galaxy by summing the mass of all particles in that group at that time.

2.2 Merger Tree Computation

The merger-tree computation requires three steps: selecting galaxies of interest from the present day, selecting the particles associated with those galaxies, and tracing these particles to earlier times to find the galaxies that have merged together. These associated galaxies are used to generate the "edges" of the merger tree. Each step can be expressed as a query on the `ParticleTable` view.

Step 1: galaxiesOfInterest. Typical users want to see the merger trees for a subset of galaxies at present day, often predicated on galaxy mass. For example, many scientists are interested in dwarf galaxies, which at present day have a total mass below $10^{10.5}$ solar masses. We select these `galaxiesOfInterest` with a groupby-aggregation query over particles in `ParticleTable` at the most recent moment in time.

[2]Because Myria does not yet support views, we either materialize them (for small simulations) or manually rewrite queries to use the base tables (in this paper).

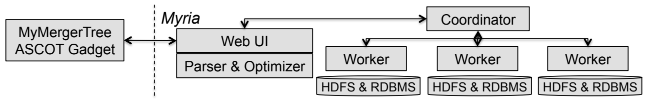

Figure 1: MyMergerTree service architecture.

Step 2: particlesFromGalaxies. To construct the merger tree for each galaxy in the `galaxiesOfInterest` relation, we need to extract the particles that belong to these galaxies at present day and track them backward in time. Conceptually, we use a join with `galaxiesOfInterest` in order to filter the `ParticleTable` and extract the `iOrd` for particles that belong to the selected galaxies. We then perform a semi-join between this result and `ParticleTable` to extract their information at earlier times. We call the resulting relation `particlesFromGalaxies`.

Step 3: edgesTable. Finally, we link galaxies across time if they have particles in common. This step requires a self-join of `particlesFromGalaxies` on the `iOrd` attribute, with the additional join constraint that matching particles must come from adjacent timesteps. An additional groupby-aggregate filters the very large number of possible pairs of galaxies to a scientifically significant set that, *e.g.*, contains a substantial amount of dark matter and many particles (see below). The resulting relation is called the `edgesTable`. As implied by the name, this query results in a leveled adjacency list, a common way to model a graph in SQL [4]. In this model, each edge of the graph is represented as a pair of nodes in which ordering matters. An additional column captures the timestep where the edge belongs. Myria supports multiple query languages [7]; We show the SQL query to generate `edgesTable`:

```
CREATE VIEW edgesTable AS
SELECT p1.time as time,
       p1.grpID as currentGalaxy,
       p2.grpID as nextGalaxy,
       SUM(p2.mass) as totalMass,
       SUM(CASE WHEN p2.type="dark"
           THEN 1 ELSE 0 END) as countDark,
       COUNT(p2.iOrd) as countParticles
FROM   particlesFromGalaxies p1,
       particlesFromGalaxies p2
WHERE  p1.iOrd = p2.iOrd and
       p1.time + 1 = p2.time
GROUP BY p1.time, p1.grpID, p2.grpID
HAVING SUM(CASE WHEN p2.type="dark"
           THEN 1 ELSE 0 END)  > 64
        and COUNT(p2.iOrd) > 100
```

3. SERVICE BACK-END: MYRIA

The architecture of MyMergerTree is shown in Figure 1. MyMergerTree uses Myria for data storage and parallel, distributed query evaluation on a shared-nothing cluster. Myria can read and process data directly from external sources, *e.g.*, HDFS or the Internet, and data loaded into Myria is stored in PostgreSQL, with an independent instance running at each cluster node (similar to HadoopDB [2]). By this design, Myria can leverage PostgreSQL's indexing capabilities and can also push some of the computation directly into that storage layer. Once in memory, Myria continues processing the data using its own in-memory relational and data shuffling operators.

Initially, the cosmological simulation data is loaded into HDFS, allowing us to replicate the snapshots across the cluster and ensure that one copy of the data always remains in its original format. Next, we execute a Myria query to read the data from HDFS, parse it with a custom NChiladaFileScan operator, distribute it in round-robin fashion across a set of Myria workers, and store it in the underlying PostgreSQL instances. Once the simulation is stored in PostgreSQL, we index the particle data (*i.e.*, one table per snapshot) on the `iOrd` and `grpID` attributes.

As outlined in Section 2, Step 1 of the merger tree analysis is an aggregation query that computes `galaxiesOfInterest`. This query can easily be executed in parallel across all workers. We discuss its performance in Section 5.

Step 2, the most computationally expensive part, involves tracing the particles from `galaxiesOfInterest` across time. This query joins the `galaxiesOfInterest` relation with two copies of the `ParticleTable` view of the entire 5 TB dataset! Similarly, Step 3 requires a parallel self-join of the entire `particlesFromGalaxies` relation. By default, Myria executes parallel hash-joins to evaluate these queries, but this would require re-shuffling the very large tables. Instead, we optimize the physical plan.

To avoid the default behavior, we invoke several known optimizations that prevent re-shuffling of large relations. The underlying premise is to identify the small tables that are constructed as part of the analysis and to broadcast them among all the workers, so that the large tables need not be shuffled. To do this, we first define a Myria query to broadcast the `galaxiesOfInterest` relation to all workers and then join `galaxiesOfInterest` locally at each worker with the table holding the most recent snapshot. This happens in parallel without the shuffling of any data. We call the result the `todayParticles` relation. Second, because the `todayParticles` table is typically small compared to any individual snapshot table, we broadcast it as well. We can now compute the `particlesFromGalaxies` relation with a parallel join between `todayParticles` and the `ParticleTable` view (*i.e.*, each underlying snapshot table) without data shuffling. We hash-partition the result on `iOrd` in preparation for Step 3. Finally, we execute the query shown in Step 3. Again, the join in that query is done in parallel without shuffling any data. Only partial aggregates are shuffled to produce the final result.

In contrast to this approach, we observe that if we had hash-partitioned all the snapshot data on the `iOrd` particle identifier when initially ingesting the data, we could have avoided broadcasting the `todayParticles` relation and re-hashing `particlesFromGalaxies`. When the input data is hash-partitioned on `iOrd`, all joins on that attribute can be done in parallel without any data shuffling. We compare runtimes of building a merger tree through the initial non-hashed setup to the hashed-on-ingest method in Section 5.

4. SERVICE FRONT-END

MyMergerTree's front-end uses the Astronomical Collaborative Toolkit (ASCOT) [11], which is a web based data processing platform that can access data from a wide array of sources and integrate it into a unified dashboard. We chose to build the MyMergerTree front-end as one "gadget" (stand-alone tool) in ASCOT to enable future integration with other existing gadgets. MyMergerTree's front-end enables scientists to analyze the simulation data directly from their browser without the need to write queries.

When a user first starts the ASCOT merger tree gadget, the user must provide selection criteria for the present day galaxies of interest based on calculable galaxy properties such as mass, luminosity, or gas content (the current version only supports predicates on mass). The user can also alter the particle conditions used in calculating the `edgesTable`. Given these galaxy specifications, MyMergerTree computes the requested merger trees by issuing the corresponding queries as described in Sections 2 and 3. These results are cached in Myria to eliminate re-computation for subsequent users.

Once merger trees of interest have been computed, MyMerg-

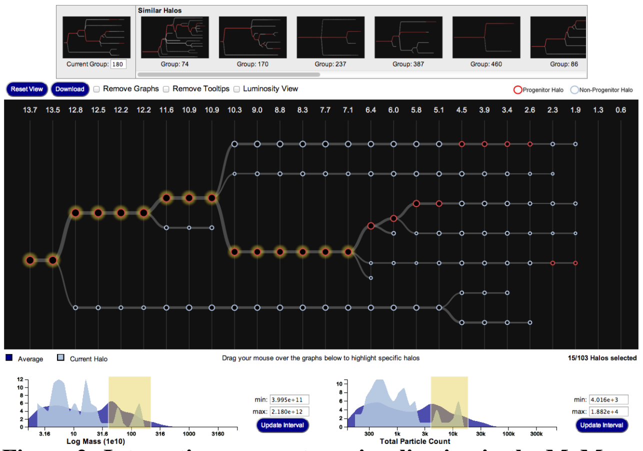

Figure 2: Interactive merger tree visualization in the MyMergerTree service.

erTree displays them using the D3 visualization library, which allows the user to interactively explore them as shown in Figure 2. MyMergerTree initially shows a random galaxy, but the user can enter a particular galaxy's `grpID` to view its merger tree. Galaxies are represented as nodes in the resulting tree.

When exploring the merger tree visualization, users may hover over galaxies and MyMergerTree will display the relevant galaxy information. The user can click on a galaxy to collapse/expand earlier galaxy leaves. By dragging and scrolling on the tree diagram, the user can zoom in/out to focus on interesting merger activity. The scroll bar at the top of the visualization window makes navigation between trees relatively simple.

Two function graphs located at the bottom of the visualization window allow users to interactively select relevant galaxies on the tree based on user defined criteria. For example, the left graph in Figure 2, selects specific galaxies based on their mass information. Upon selecting a mass threshold (which is done by brushing the graph), the galaxies meeting the specified criteria are highlighted in the graph. The user can highlight all earlier leaves associated with a particular galaxy by holding the "Shift" key and clicking on a galaxy node of the tree. Finally, users can export data in the highlighted portions of the merger tree by clicking the "Download" button; this returns a CSV list of galaxy properties including `grpID`, total mass, particle count, dark particle count and luminosity.

5. EVALUATION

We present initial performance results and some of the new science that the N-body group seeks to carry out using MyMergerTree.

Quantitative: We run all experiments in a cluster of 32 Myria processes (workers) spread across 16 physical machines (Ubuntu 13.04 with 4 disks, 64 GB of RAM, and 4 Intel(R) Xeon(R) 2.00 GHz processors). We first ingest the data from HDFS and distribute it in a round-robin fashion in the PostreSQL instances on the machines, which takes an average $\sim$5 hours per snapshot.

To evaluate the merger tree computation time, we vary two parameters. First, we vary the number of snapshots (representing the depth of the tree) up to 7 snapshots, representing $\sim$700 GB of data. Second, we vary the number of galaxies in the `galaxiesOfInterest` table. Figure 3 shows the total time to construct the `edgesTable` in each configuration. As the number of snapshots increases, more local joins are required per worker. Additionally, as the number of galaxies in the `galaxiesOfInterest` table increases, the size of the relation in the local join increases, leading to an approximately linear in-

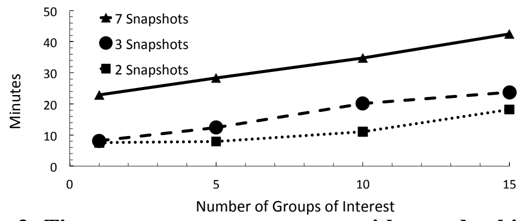

Figure 3: Time to compute merger trees with round-robin data partitioning.

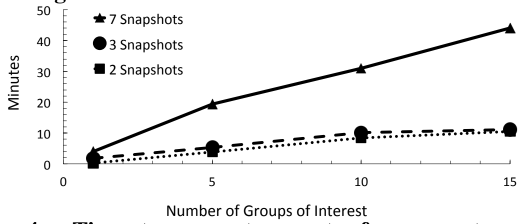

Figure 4: Time to compute a set of merger trees with application-aware data partitioning, hashing data on `iOrd`.

crease in query runtimes. On average, MyMergerTree can build a single merger tree in approximately 22 minutes for seven snapshots and within 42 minutes for building 15 trees simultaneously.

To reduce runtimes, we re-hash the input data on the `iOrd` attribute as described in Section 3. Re-hashing takes $\sim$1.6 hours per snapshot. Although this process is time consuming, this cost is incurred only once and the benefit is worthwhile. As shown in Figure 4, runtimes for the tree generation decrease by as much as 80% for some cases. Through this method, MyMergerTree can take at most 3 minutes per tree when building 15 trees at a time for 7 snapshots. We measure worst-case runtimes for the service by running a large, unrelated query before each timed query.

In the figures, approximately 11% percent of the time shown goes to compute `todayParticles`, 86% percent goes into `particlesFromGalaxies`, and a remaining 3% to build the edges. The time to build the `galaxiesOfInterest` table is not included. It takes approximately 3.75 minutes time to run this query.

Although these runtimes are promising, we seek to improve them. MyMergerTree already caches merger trees to avoid recomputing them when possible. We can also use more workers: we have successfully executed queries with Myria on 100-instance Amazon EC2 deployments, but with a different dataset. We are also working on further optimizing the queries themselves and the split of operations between Myria and the underlying database system.

Qualitative: The initial reactions of the N-body group to the new service have been positive. The group wants to use the MyMergerTree service to visually and quantitatively correlate the merger histories of individual galaxies with the evolution of particular properties, such as star formation rate and gas fraction. The group would also like to use MyMergerTree in a more statistical manner, where MyMergerTree will use a metric to quantify the similarity of merger trees. While MyMergerTree can already compute and display similar merger trees, we are further developing this feature to allow astronomers to correlate similar merger histories with global trends.

6. RELATED WORK

In order to obtain robust merger rates and merger trees, one needs rich galaxy statistics from a large and well resolved cosmological simulation and a careful treatment of the systematic effects due to the galaxy finding and merger algorithms employed [6]. One of the largest, publicly available N–body simulations that has been used to create merger trees is the Millennium simulation [13]; merger trees from this dark matter only realization of the Universe have been thoughtfully treated using database technology [1]. However, the merger trees from this analysis that are freely distributed lack a high degree of time resolution and are generally used only for semi-analytic purposes. The merger trees we create with the MyMergerTree service can consider any desired degree of time resolution and allow for detailed individual studies of simulated galaxies; moreover, MyMergerTree can be leveraged by any simulation group, regardless to their personal access to database technology.

Crankshaw *et al.* [5] develop an inverted index for particle tracking in cosmological simulations. Our work is complementary: we focus on the building blocks involved in developing a complete service for computing and analyzing merger trees.

7. CONCLUSION

We presented the MyMergerTree service developed to facilitate the study of galactic merger trees found in large-scale cosmological simulations. There are many lessons learned from building MyMergerTree: while data ingest times could use optimizations, they are not the key challenge as ingestion happens only once. Tools to automatically validate ingested data against the original data are more critically needed. The most interesting challenge, in our experience, remains query tuning: one needs to transform queries expressed in a manner close to the spirit of the original analysis into queries that can efficiently be evaluated, which may require changing query result schemas. Overall, building vertical services, such as MyMergerTree, remains a non-trivial project for a highly-qualified inter-disciplinary team. Methods and tools to accelerate this process are important challenges for the data management community.

Acknowledgments: This work was supported in part by the National Science Foundation through NSF grant IIS-1247469, the Intel Science and Technology Center for Big Data, a grant from Gordon and Betty Moore Foundation and the Alfred P. Sloan Foundation, and the University of Washington eScience Institute. JO is also supported by an NSF Graduate Fellowship. SL acknowledges support from the Michigan Society of Fellows. The simulation was run on NCSA Blue Waters under NSF PRAC Award 1144357. LA was supported by NSF AST-1311956 and NASA HST AR-13264. FG acknowledges support from NSF grant AST-0607819 and NASA ATP NNX08AG84G.

8. REFERENCES

[1] Virgo–millennium database merger trees. `http://gavo.mpa-garching.mpg.de/Millennium/Help?page=mergertrees`.

[2] A. Abouzeid et al. HadoopDB: An architectural hybrid of MapReduce and DBMS technologies for analytical workloads. *Proc. VLDB Endow.*, 2(1):922–933, Aug. 2009.

[3] M. Bostock, V. Ogievetsky, and J. Heer. D3: Data-driven documents. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2011.

[4] J. Celko. *Joe Celko's Trees and Hierarchies in SQL for Smarties*. Morgan Kaufmann, second edition, 2012.

[5] D. Crankshaw et al. Inverted indices for particle tracking in petascale cosmological simulations. In *Proc. of SSDBM*, pages 25:1–25:10, 2013.

[6] O. Fakhouri and C.-P. Ma. The nearly universal merger rate of dark matter haloes in ΛCDM cosmology. *MNRAS*, 386:577–592, May 2008.

[7] D. Halperin et al. Demonstration of the Myria big data management service. *SIGMOD 2014*.

[8] P. Jetley et al. Massively parallel cosmological simulations with ChaNGa. In *Proc. IPDPS*, 2008.

[9] S. Loebman et al. Analyzing massive astrophysical datasets: Can Pig/Hadoop or a relational DBMS help? IASDS, 2009.

[10] S. R. Loebman. *The Milky Way in SDSS and in N-body Models*. PhD thesis, University of Washington, 2013.

[11] D. Marcos et al. ASCOT: A Collaborative Platform for the Virtual Observatory. In *Astronomical Data Analysis Software and Systems XXI*, volume 461, page 901, 2012.

[12] SDSS SkyServer DR7.

[13] V. Springel et al. Simulations of the formation, evolution and clustering of galaxies and quasars. *Nature*, 435:629–636, June 2005.