CS 657 - Paper Review 1

SciDB: A Database Management System for Applications with Complex Analytics

Nick Alvarez

25 February 2021

1 Clarity

This article is presented in a concise, yet clear, manner. In a broad sense, there is an problem-solution organization. Listing the current needs of the scientific community in relation to database management systems (DBMS) and breaking down why the existing state-of-the-art solutions are inadequate sets the stage for a competitor that not only meets but exceeds those needs. This is exactly what the authors do. Each key request is implemented into the system and documented in the article with its functionalities and inspiration. Finally, the article lists some practical use cases where researchers have utilized SciDB and produced the desired results and/or showcased excellent performance gains.

Presenting one's work in this manner provides a clear problem and highlights why the new solution is the best fit for the issue.

2 Problems and Existing Solutions

Scientific data is very large, sometimes nearing 100 petabytes. Stonebraker et al. states that traditional relational database management systems (RDBMs) are not equipped to handle this volume of data, and operations like "regridding" would be extremely costly. They proceed to claim that using tables to store this data instead of leaving them as native arrays is "unnatural" for this type of data. Stonebraker et al. abstracts their issues with traditional RDBMs and states these systems are not able to handle the specificities of the data from the scientific community, namely, imprecise data and varying null entries. Additionally, these systems were not designed with no-overwrite stores and distributed computing.

Other state-of-the-art methods of storing data are mentioned: File systems (difficult to search metadata and standard DBMS capabilities unavailable), HDF (lesser performance and no transactional guarantees), and Hadoop (unable to scale to size of required projects due to inefficient communication model).

These problems are important within the scientific community as when situations arise that require the features of a DBMS but the flexibility to handle their unique data, existing solutions are not feasible for their work. While it has been done in the past using a variety of methods, it was done out of necessity, not preference. Providing the community with the ability to handle these massive amounts of (structurally) unique data would allow for more focus on research and less on how to store, query, and modify the data.

The problem of creating a database management system for applications with complex analytics is most certainly a hard one. Stonebraker et al. notes that teams faced with the task of storing and manipulating data of this magnitude and type, in the past, have created their own proprietary solution "from the bare metal on up." Nothing that meets the needs of the scientific community had been created or prioritized, and RDBMS vendors did not believe their software was inadequate. Creating an array-centric DBMS would be uncharted territory.

3 Techniques and New Solutions

SciDB works, and was designed, with native array data models in mind. There can be N number of dimensions, with each cell containing a vector of varying data types. Switching between dimensions and individual vector values ("attributes") is also supported, as the expected workload can vary. Users who are familiar with SQL will be prepared to use SciDB's Array Query Language. Operations are similar, but a key difference is filters produce sub-arrays with the same dimensions and attributes of the original array, but with more "not valid" cells. This is as opposed to creating a smaller table with omitted information. AQL can compile into Array Functional Language (AFL) so programmers are able to create their own desired results by cascading language primitives together. Performance is increased over RDBMS using "chunks" of just a few megabytes, each representing a single cell value (multiple chunks for multiple vector values) so disk seek times may be decreased, with vectors being treated similarly. Dealing with multidimensional queries is more straightforward than RDBMS, since those two dimensions are already turned into their own chunks, making searching as quick as any other query. No-overwrite storage is included, with each array containing a "version" dimension, tracking cell values as a backwards delta of values.

From the information presented, readers can clearly determine that SciDB is more suitable to the needs of the scientific community compared to existing state-of-the-art solutions. SciDB is shown to deal with data of N-dimensions, distributed clusters, no-overwrites, and the performance requirements of the large magnitudes of data, among other requirements. They took each main request from the scientific community, implemented it into SciDB, and then elaborated on how it works and the reason it was added.

Stonebraker et al. proves to the reader that the new DBMS works by detailing individual use cases, like satellite imagery, astronomy, and genomics. One case to note is from the Lyra astronomy project, where they coded their application on SciDB to perform “fuzzy joins” of data. The authors then provide a comparison between SciDB and Postgres, stating that their application ran twice as fast as the existing solution. A similar comparison was provided in a human genome application, where a user coded application in SciDB was compared to an application meant to provide the same results, but coded in R. The SciDB solution was scalable and completed in 20 minutes, in parallel, with test data. The R solution was limited to a single node and could not be run due to matrix indexing limits. With these use cases and evidence, SciDB clearly proves its merit as a viable DBMS for the scientific community’s needs.

With SciDB being both unprecedented and newly formed, there are some limitations to the software. Linear algebra operations within SciDB rely on ScaLAPACK, and all data must be converted to the latter’s format, processed, and reformatted again. These separate but mutually required systems compete for resources, and with large data sets, ScaLAPACK can reach running times of O(N3). Another issues lies within the chunking of data, where fixed logical-size chunks can vary in actual size, creating a skew in chunk size. This creates issues when processing chunks, as the time to set up a sparse chunk is spread out over much fewer values. However, with the limitations discussed, the authors present potential options for circumventing or eliminating the limitations.

4 Suggestions

This paper was presented well. The audience was intended to be members of the scientific community and those familiar with DBMS. Detailing the functionality of SciDB could have been accompanied with more figures and examples. Specifically, clarifying the joins discussed in *Query Execution* with a figure would have made understanding that section easier. The use cases section could have used figures describing how satellite or telescopic imagery is “cooked” together. Even tables with data comparing existing RDBMS and SciDB performance on large data sets would have provided more evidence to the reader that SciDB is the best choice for scientific applications with complex analysis. However, as SciDB is unique in its field, not necessarily for-profit (there is a free, unsupported version), and provided as a resource to the scientific community, the authors may have surmised that those interested in the application will understand its merits at face-value and can perform comparisons on their own data, if they wished.