Enabling Access to Timeseries, Geospatial Data for On-demand Visualization

Sangmi Lee Pallickara, Matthew Malensek, and Shrideep Pallickara

(sangmi, malensek, shrideep)@cs.colostate.edu

Department of Computer Science
Colorado State University

ABSTRACT

Data volumes in the geosciences have been increasing substantially over the past decade. This data naturally includes both spatial and chronological information. Fast access to specific portions of large collections of such datasets is a precursor to efficiently visualizing them. Here, we provide an overview of how a geospatial storage system, GALILEO, provides storage of such timeseries data while providing access to specific portions of the dataset.

KEYWORDS:

Geoscience, time-series data, distributed geospatial storage.

1 INTRODUCTION

In the geosciences there is often a need to visualize portions of the data that represent measurements from data sources such as sensors, radars, satellites, and networked instruments. The earliest records in the geosciences relate to thermometer and barometer measurements that were made in the mid-to-late 1600s. The volume of data generated in the geosciences has increased dramatically over the past decade. Estimates [1] for the cumulative size of the global climate data from climate models, remotely sensed data, and *in situ* measurement data is expected to cross 300 petabytes. This represents a 30-fold increase in the estimated data volume for the year 2011.

Such rapid growth in data volumes poses challenges to scientists in both the geosciences and computer science. Data intensive computing [2] is an example of a new research paradigm to cope with such data volumes. Besides developing domain-specific algorithms, other critical requirements in such settings include efficient access, querying, sharing, and processing of large datasets. Besides being inherently geospatial, such data also has a time component: often, the same feature such as humidity, pressure, etc. will have observational measurements being reported for different times for the same geographic location.

We have developed a distributed scalable storage system, GALILEO, specifically for storing and querying large geospatial time-series datasets. GALILEO provides efficient access to specific portions of large datasets, and an ability to sift and process such observational data in real-time prior to storage via the Granules runtime [3]. There have been several approaches to provide a storage service to applications using geospatial datasets. Geographic Information Systems (GIS) provide advanced search and information overlay features [4, 5] for such data. GIS-based approaches are very useful for systems such as map-based information retrieval services, however, time-series datasets especially data streams that represent realtime measurements have not been their focus. GALILEO supports high-resolution time series datasets that are published as streams. Domain specific software and services for storing and sharing geospatial datasets are available [6],[7]. These systems incorporate support for distributing datasets within a specific community while incorporating support for advanced data access functionalities such as data subsetting or augmentation. However most of the data processing such as filtering must be performed on the client's computing resources. Data stored in GALILEO is co-located with the Granules distributed stream processing system. Granules allows computations to be developed in C, C++, Java, R, and Python. Granules manages the lifecycle of these computations, and it activates computations that have data available on their input data streams and keeps them dormant otherwise. The processing logic within these computations can be arbitrary and is not restricted to say, query evaluation. By interleaving multiple computations on the same machine, Granules maximizes resource utilizations. The execution of data processing in GALILEO may also be orchestrated using MapReduce. In the database community, there have been active investigations for managing geospatial datasets [8] and N-dimensional scientific array data [9] [10] however issues related to the size of these datasets and the size of database tables persist. Popular geoscience data formats include netCDF [11] and HDF [12].

2 MOTIVATIONS AND DESIGN ISSUES IN GALILEO

Periodically published time-series observational data is generally made available as a package, with care taken to encode the metadata and data such that it results in a compact representation with the smallest possible array of bytes. Often the metadata is minimally encoded in the name of the file that has been made available. (e.g. Data from the NESDIS Satellite between 2011-07-09 to 2011-07-16 would be encoded as /fileServer/satellite/3.9/WEST-CONUS_4km/20110716/WEST-CONUS_4km_3.9_20110716_0300.gini). However, this style of organizing the data for packaging and distribution can limit the amount of processing that can be done: for example, querying a collection of datasets for specific attributes that span multiple datasets is often a challenge in such encoding schemes.

Data accesses are closely related to the application and the algorithms that underpin the processing that will be performed. Data visualization represents an example of a unique data access style. For geospatial data visualization, there are several distinctive characteristics in terms of the data access:

- Data access is often closely associated with geospatial attributes.

- Time series data, for a specific set of features, associated with a specified area is often accessed.
- Interactive access to the selected parameters (e.g. humidity, temperature) is required.
- On-demand data processing is needed. For e.g. creating of an image, simple statistics such as computing the mean and standard deviation.

Static file based data storage systems cannot support visualization tools for near real time datasets.

In GALILEO, datasets are delivered in near real-time and stored in a cluster of commodity machines. Since many geoscientific computations are tightly related to the geospatial attributes, we stage datasets based on its geospatial information to maximize the data locality; this ensures that observations from locations that are *close* to each other are most likely on the same machine. Similarly, temporal information and features (e.g. wind speed, humidity) are major components that determine how and where we stage the dataset. Finally the data, especially for large datasets, is extremely hard to recover when there are data corruptions and node failures. In GALILEO we rely on replications and automated discovery of nodes holding replicas of data blocks to cope with such failures and faults.

Queries in GALILEO can specify geographical ranges, temporal ranges, and specific features. Each query is evaluated concurrently on the nodes that comprise the distributed storage and the results of these queries are streamed back to the requestor. The responses generated by the individual nodes (each of which is responsible for storing a portion of the data collection) to such queries are then used to construct a *virtual dataset*.

This virtual dataset is constructed on the fly, organized in memory using a tree-based data structure, and evolves as data blocks are found (and added to the virtual dataset) to satisfy the query. The graph can be traversed and the block metadata queried for relevant information. Should the need arise, blocks can retrieved from the nodes that host them since this information is encoded in the block metadata. The virtual dataset allows creation of multiple views so that only those elements that need to be visualized would be retrieved. These data blocks can be retrieved in the presence of failures.

3 PRELIMINARY PERFORMANCE EVALUATION

We have performed some preliminary experiments with GALILEO. These experiments were performed on a cluster of 48 quad-core Xeon-based servers with a gigabit interconnect. Each machine is equipped with 12GB of RAM and a 300GB, 15,000 RPM hard disk. The experiments ran on version 1.6.0_20 of the OpenJDK Runtime Environment.

To create a test scenario, 100 million data blocks were generated randomly and streamed into the system from four different sources. Each block contained 1000 simulated sensor readings. Along with the data itself, metadata was also generated randomly. Blocks were assigned temporal ranges within the years 2002-2011 and were evenly dispersed across the continental United States' spatial range. Each block represented one of three features: pressure, humidity, and temperature readings.

Next, we executed queries to retrieve parts of the randomly generated collection of measurements representing a data collection of 100 million blocks each of size 4 KB. We queried for blocks that contained pressure readings for the month of July in 2011 within a geospatial range that roughly bounded the state of Colorado and parts of Utah and Wyoming. This query was evaluated concurrently on all the nodes that comprised the storage network. This query returned 105,556 data blocks, with the first result payload being received in 542.96ms and the final result being received in 5769.21ms. The results that were streamed back in this experiment included the data blocks themselves. We also have the ability to stream back only the block metadata that summarizes the observations and the locations of the storage nodes that held the corresponding block rather than the individual blocks; in such a setting, a client can interactively choose specific blocks that it would like to retrieve and visualize. Such selective retrievals underpin on-demand visualizations.

4 CONCLUSIONS

Large datasets that arrive in near realtime have been a challenge in the geosciences. This is especially true for an application that requires high accuracy and efficient access to portions of large data volumes. GALILEO supports efficient distribution of data over a collection of nodes in a decentralized fashion. The system allows access to specific portions of the dataset with the results generated in response to a queries being streamed back to the requestor. Streaming allows visualization to make progress as data continues to become available, instead of awaiting the completion of data transfers; such interleaving of transfers and visualization result in faster visualizations.

REFERENCES

[1] J. T. Overpeck, *et al*., "Climate Data Challenges in the 21st Centry," *Science* vol. 221, pp. 700-702, 2011.

[2] T. Hey, *et al*., *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Redmond, Washington: Microsoft Corporation, 2009.

[3] S. Pallickara, *et al*., "Granules: A Lightweight, Streaming Runtime for Cloud Computing With Support for Map-Reduce.," in *the IEEE International Conference on Cluster Computing*, 2009.

[4] P. Ramsey, "PostGIS Manual," ed: Refractions Research.

[5] A. Guttman, "R-trees: a dynamic index structure for spatial searching," in *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, ed. Boston, Massachusetts: ACM, 1984, pp. 47-57.

[6] P. Cornillon, *et al*., "OPeNDAP: Accessing data in a distributed, heterogeneous environment," *Data Science Journal,* vol. 2, pp. 164-174, 2003.

[7] B. Domenico, *et al*., "Thematic Real-time Environmental Distributed Data Services (THREDDS): Incorporating Interactive Analysis Tools into NSDL," *Journal of Interactivity in Digital Libraries,* vol. 2, 2002.

[8] K. Chodorow and M. Dirolf, *MongoDB: The Definitive Guide*: O'Reilly Media, 2010.

[9] P. Cudre-Mauroux, *et al*., "A Demonstration of SciDB: A Science-Oriented DBMS," in *the 2009 VLDB Endowment* 2009.

[10] P. G. Brown, "Overview of sciDB: large scale array storage, processing and analysis," in *Proceedings of the 2010 international conference on Management of data*, ed. Indianapolis, Indiana, USA: ACM, 2010, pp. 963-968.

[11] R. Rew and G. Davis, "NetCDF: an interface for scientific data access," *IEEE Computer Graphics and Applications,* vol. 10, pp. 76-82, 1990.

[12] (2010, *The HDF Group. Hierarchical data format version 5*. http://www.hdfgroup.org/HDF5.