Implementing a distributed spatial index for astronomical data

Motivation
Professor Paolo Coppi’s lab in the Yale Astronomy department is studying the motion of objects in the sky, via a time-series survey. In order to do this, they need to use data both generated from their own telescopes and from various other catalogs in which the observations were carried out at different times, so that they can determine the change in position over time relative to nearby stars. The current DBMS that they use is inadequate for this research, both because it cannot handle the scale of data and because the data they have cannot be queried adequately.

The entire astronomy community is struggling to keep their data management infrastructure up to date with the constant stream of data from new experiments. As CCDs, tools to photograph the sky using many different filters, have grown more technologically advanced, the amount of data available has grown exponentially. Currently 1 PB of data is available electronically, and that number is expected to grow to 60 PB or more by 2020 (Berriman and Groom, 2011). The 2010 Decadal Survey of Astronomy and Astrophysics, commissioned by the National Academy of Sciences, has in fact highlighted the storage of TB-sized data sets as an important issue facing the community for the next decade (Committee for a Decadal Survey, 2010).

One answer to this flood of data that has been a focus of much research recently is to build up a system of federated databases, so that each archive needs only to be stored in one place. The Virtual Observatory project is currently creating a federated databases, which allows the individual archives to keep their own infrastructure while at the same time allowing the entire system to behave like a single database, able to respond to user queries and perform joins between the different archives (Malik, 2002). However, even individual archives are overflowing with data. So, for my senior project, I will focus on optimizing a single distributed database for astronomical data, which can potentially later be linked in with a federation of astronomical data.

In the description of my project that follows, first, I will go into a more detailed description of the computational needs of the Yale astronomy research group. Second, I will describe the types of spatial indexes that exist currently. Finally, I will describe my proposed research topic, which is to design and implement a distributed spatial index for this astronomical data, including a list of deliverables at the end.

Needs of Yale Astronomy
In order to conduct their time-series research, the astronomers need the following capabilities in a DBMS:

- Spatial indexing of the right ascension and declination (latitude and longitude in the sky) of the astronomical objects, where right ascension and declination are measured with a certain degree of error.
- The ability to perform a “fuzzy join” between different catalogs, since the exact right ascension and declination of objects can change between catalogs based on random or systematic error.
- The ability to update the database nightly to add new observations and identify new objects.
Additionally, any database used would optimally also be able to scale to store petabytes of data. A traditional commercial database like DB2 or Microsoft SQL Server would not be a viable option, because none of them have scaled well into hundreds or thousands of nodes. Thus, a highly scalable and flexible DBMS is needed.

Types of spatial indexes

The spatial index requirement is essential for efficient queries, and a spatial index enables further features like finding neighbors or performing a fuzzy join. The nature of the sky as a sphere creates difficulties for this indexing, however, so often astronomers have to create their own spatial indexes. There are two main types of spatial indexes: 1) hash functions that map parts of the sky to an integer value, which can then be put into a standard 1-D index and 2) multi-dimensional indexes that can keep track of geometrical objects in addition to points.

Though there are many different hash functions that have been proposed (HEALPix, Q3C), the most commonly used one is Hierarchical Triangular Mesh (HTM). HTM divides the sphere into approximately equal triangles, and recursively adds more sub-triangles as the data grow more dense in a particular area. This method allows data with an uneven distribution to be subdivided approximately evenly. Each sub-triangle has a unique name based on how it has been divided, and that unique name can be stored in, for example, a B-tree index (Kunszt, 2001). When ordered alphabetically, the names that are near one another are near each other on the sphere. HTM is used by many projects, most notably the SkyServer, the DBMS for the Sloan Digital SkySurvey (SDSS), which is a Microsoft SQL Server with HTM spatial indexing capability added (Szalay, “SSDS”).

A multi-dimensional spatial index is sometimes used as well, both to store the 2-dimensional spatial data alone and many-dimensional data (such as when color, time, and other factors are included in the index along with the spatial data, or for geometric objects). The traditional example of a multi-dimensional spatial index is an R tree. R trees store bounding boxes around positions of objects, and the bounding boxes on different branches of the tree can be overlapping. As a result, the worst case search time is O(N), where N=number of objects in the database (Guttman, 1984). In addition to R trees, however, there are also k-d trees, R* trees, kdb trees, and many others that have various different properties and have different worst case search times.

As far as I have found in my survey of the literature, I have not seen any system that has used a distributed spatial index for astronomy. This distributed index, then, would be a useful contribution to the astronomy community.

My proposed project

The first step in my project will be to get a functional DBMS that can store the data that the astronomers have. The DBMS on which I will start building my distributed spatial index is HadoopDB, a scalable database project in Professor Daniel Abadi’s lab. In particular, I will be using Hadapt, the commercialized form of HadoopDB, because HadoopDB is missing some key features like the ability to create indexes in the underlying databases and is also not as stable as Hadapt. I will need to make decisions about how to store the data, and consult with the astronomy team to determine what fields they would want indexed.

Hadapt, though it supports creation of B-tree indexes in the underlying PostgreSQL databases, does not support a spatial index. Thus, the second step is to modify an existing PostgreSQL library to work in Hadapt, likely an HTM index since that is the standard spatial
indexing technique in astronomy.

This second step will have created a distributed spatial index, because of the way that Hadapt works, but the index may well not be as effective as it could be because of data locality. It would be best to have the images that are close together in the sky stored on the same node, but the partitioning scheme of Hadapt may well not do that. The next step, then, would be to determine how best to partition the data, ensuing locality.

An additional consideration, however, is that all of the right ascension and declination values have an error, so properly their values are a box of possible values. In order for all queries to take place on one node even with these error boxes, the edges of the sky on each node need to be stored multiple times.

These further directions of research on how to optimize the distributed spatial index may well change as the project progresses and I find different bottlenecks in the process.

**Deliverables**

- The working database for astronomers to use (with the ability to load data, perform some selected queries, etc.)
- Code modifications, in the form of a UDF to assist in making the index and/or modifications to Hadapt.
- Written report in a format that is suitable to be submitted to a journal (e.g. approximately 12 pages)
- Web pages describing my project

**Bibliography**


