TerraPop Raster Ops - Performance Problem Statement

Terra Populus is a project that queries and analyzes raster datasets containing environmental data (e.g. land cover, crop yield) covering the entire world at various levels of granularity (e.g. 1km, 100m). There are two challenging IT problems that arise from performing raster operations in this project.

1. **Dealing with Projections**. Worldwide raster datasets are represented as a grid layered over a world map. For many rasters, the world map is represented using lat/long coordinates (known as geographic coordinate systems, e.g., WGS84). In other cases, the world map is a projection, such as the Mercator projection, common for world maps. In geographic coordinate systems and many projections, the on-the-ground area underneath each raster cell (i.e. pixel) is not uniform, since the 2D projection distorts the 3D Earth. This is the "Greenland looks huge on this map" effect. We therefore have to adjust for represented area when we perform calculations.
2. **Size of Data**. Analyzing rasters becomes computationally intensive depending on the resolution of the raster and the geographic extent of the region in question. Doing an operation to a raster which represents the entire world at 100m resolution is computationally quite demanding. We have at least one dataset of interest which is at 30m resolution and requires 600GB to represent the entire world.

The Terra Pop project is unique in that we allow researchers to extract, combine, calculate, and transform any of the data within our system into three different data types:

- **Microdata**: Data about individual persons. Examples include age, sex, or level of education.
- **Areal**: Data representing a given geographic area. Examples include median income and % of population with college degree. Example geographic areas include state, county and census tract.
- **Raster**: Data divided into cells (i.e. bitmap). Often derived from satellite imagery. Examples would include corn yield in bushels, % tree cover or categorical data such as terrain type.

With this core requirement to allow connections and transformations between these heterogeneous data types, the project team looked for a single IT platform that could store and operate on all of these data. We currently use PostgreSQL with the PostGIS extension for our extraction system. This is ideal because we are able to store and interact with all three data types within one system, but it is not without challenges. In particular, the PostGIS raster extension is fairly new and developing efficient queries has been difficult.

Returning to the two challenges of the first paragraph, we've tried a number of approaches to the first challenge. Initially we hoped the "area represented by this cell" calculation could be done on the fly in within the database as we traversed a raster, but it seems that the expense is too great for PostGIS because the operation that extracts a cell's geometry (essentially vectorizing a raster) is computationally intensive. We then tried to only vectorize one column of the raster (since the area value will be the same for all of the cells at the same latitude in geographic coordinate systems), but applying those values across the entire raster still proved infeasible. Our current solution to the first challenge is to "pre-cook" multibanded (i.e. multilayered) rasters, with the first band containing the represented area and the second band containing the actual raster data from the dataset. Since we do this ahead of time before loading the rasters into the database, we avoid PostGIS limitations, but of course this is a less flexible approach that requires lots of up-front compute time. Incidentally, we have found this layering approach to be useful more generally for multiple raster analysis. For example, if user does a temporal analysis across 4 rasters, by combining all of the rasters into one multibanded raster we can significantly reduce query time.

The second challenge of poor performance for large geographic extents still remains without much progress. Since the PostGIS raster data type contains an array-like property and the engine wants to load the entire raster into memory, we hit memory limits with certain operations if the geographic area is large enough. Typically this means that instead of being able to use a built-in PostGIS summarize function we have to go cell by cell and extract values using custom code that we write. With that approach, a region the size of Canada, Russia, or Alaska will take hours or not finish. The PostGIS raster functions are somewhat limited, and writing spatial SQL queries is something of an art form. We have had numerous instances where the initial SQL query took hours to complete, but by fine tuning the query we have achieved completion times of minutes.

Terra Pop is fundamentally a batch-request-and-extract system, so we're not striving for real-time performance. Ideally we would like any spatial query to finish within a half hour. That goal is currently well out of reach, since we expect our users to routinely apply operations across the entire world or a large subset of it, and we also hope to add finer resolution data to the system. The current performance limitations prevent us from doing so.

Currently, we are exploring a variety of options. Raster operations should be highly parallelizable - most operations are simply repeating a simple operation across many cells independently of one another. Therefore, it should be straightforward to subdivide the work of a raster operation. Within a single system, we would consider multithreading, but PostGIS inherits the base restriction of PostgreSQL in that each query is farmed off to a single process and is not threadsafe.

There are many efforts to parallelize/scale out PostgreSQL, but most are not compatible with the PostGIS extension. Postgres-XL is one that claims to be PostGIS-compatible, so we're beginning to explore that option. We are also looking at alternative database systems like Spatial Hadoop (UMN) and Rasdaman (array database). Spatial Hadoop remains an academic work-in-progress, and the Rasdaman database is still immature as well.

Given the effort expended to build the project on the current platform over the past three years, and the promise offered by a mainstream platform that can represent and operate on each of our three data types, we feel the most promising path forward is finding a PostGIS-compatible PostgreSQL parallelization approach. We are seeking guidance and consultation on best approaches given our somewhat unique business requirements. Within our domain (social sciences), we are not aware of other organizations doing these sorts of large-scale, on-the-fly raster operations across such large geographic extents. Most projects seem to focus on much smaller geographic areas, and do not have the additional requirement to be able to transform/attach to and from other forms of data. We believe there must be parallels in other fields which deal with large-scale raster data (astronomy? medical imaging? meterology?), but have not had much success identifying specific projects which we might use as a model or a consulting source.

David Haynes
Research Associate, TerraPop project, MPC

Fran Fabrizio
IT Director, MPC