Parallel GIS Processing on Compute Clusters Thesis Proposal

Nathan Kerr

June 1, 2009

1 Introduction

Geospatial (GIS) simulation, analysis, and simulation are important processes to understanding and improving our environment, both urban and natural. Geospatial data is made up of a georefrenced geometry such as a point or a polygon at a certain logitude, latitude, and altitude with related descriptive information such as a land use type. GIS data can be used to represent a large range of real-world objects such as road or power networks, building locations, or natural features such as lakes and rivers. Utilizing GIS data is one method toward processing, analyzing, and simulating real-world systems. One consumer application of GIS are the GPS based car navigation systems common today.

Larger scale GIS applications also exist in areas such as city planning. City planners use GIS to study road networks, zoning issues, and to simulate population growth. As cities and metropolises grow, the amount of data

required to represent these areas also increases. As the data increases, so does the processing power required to complete the geospatial analysis, processing, and simulation.

Desktop GIS packages such as ArcGIS[2], QuantumGIS[3], and GRASS GIS[4] are commonly used for GIS processing and analysis. While these programs provide graphical interfaces to their GIS capabilities, their capabilites are limited by the computers they run on. Datasets can be too large for their memories and computations can take too long to be practical. The popular simulation package, UrbanSim[5] faces these same constraints.

An alternative approach to using these desktop programs is to employ a geospatial databases like PostGIS[6]. Geospatial databases allow centralized access to, and processing of, geospatial data through query languages such as SQL. As the data is stored by and managed by the database software, advanced database features such as indexes can be utilized to speedup data access and processing.

PostGIS is utilized as the core component of the Urban Systems Framework[17] (USF) designed by the Digital Phoenix[12] project group at Arizona State University. Digital Phoenix tries to integrate 3D visualization technology with simulated and gathered GIS data to better understands impacts of planning decisions.

GIS data has become easier to gather in recent years with the proliferation on low cost GPS devices. The availability of these devices significantly reduces the cost of data collection, moving it from a government funded ser-

vice to the capabilities of private companies and individuals even to the point of open source style maps such as OpenStreetMap[18]. With the reduced cost of data collection, the amount of data has grown significantly. As datasets grow and the associated processing becomes more sophisticated, the abilities of programs that only work on a single computer are felt through long processing times and inability to work with the required data. Thus a method of utilizing multiple computers to complete the required processing is needed to increase the size of the data that can be processed and reduce the time required to complete the processing.

One method of using multiple computers to perform the required processing is the use of parallel databases[16]. Parallel databases should be able to spread both data storage and processing across multple computers transparently from the view of the SQL query programmer. Several commercial databases such as TeraData[7] and Oracle[8] provide support for this method of operation. Opensource databases such as MySQL[**?**] and PostgreSQL[**?**] currently do not support this methodology. Parallel databases generally do not require a shared datarepository or filesystem.

Parallel database techniques have been used to build a scalable geo-spatial database system[**?**, **?**]. Data is spread between computers using round-robin, hash, or spatial partioning. Because the data is able to be distributed between multiple computers, the processing is able to scale to larger datasets.

When a query is processed across the database, a thread is created for each fragment of the data. Thus as the data grows larger, the processing

capabilities of the system also increase. If a particular processing operation requires relatively less computation for each data record this is fine. However, after the amount of computation per data record increases beyond a certain point, which is dependant on the machine used, the processing operation can be sped up by utilizing more processors. Major factors in determining how much data should be processed on each machine, and therefore how the data should be spread between machines, are memory, computation, and communication overhead to move the data and computation to another machine. The ratio of computation to memory and commucation requirements is often referred to as grain size. Finer grained processes have more computation per data record, while coarser grained processes have little computation for each data record.

Databases excel at working with indexed data while providing multiple users to interact with the data in a concurrently safe manner through the use of atomic transactions. The requirements placed upon database systems to handle these situations slow down computations that don't utilize indexes or work on an entire dataset at once by added overhead. The processing operations this research looks at do not require these restrictions, and as such a more efficient system can be created.

Using databases can also limit the reusablity of computer resources to complete other processes. For example, a process could benefit from utilizing more processors, but the disks on those database nodes could already be fully utilized, thus making them unable to accomodate the computation as it is

directly tied to data storage. This is only really an issue when trying to fully utilize hardware resources for different purposes.

Many universities and research institutions already have investment in compute clusters. These clusters are groups of computer linked together with high speed network interconnects and high performance parallel filesystems such as Lustre[**?**]. By separting compute and storage resourses at the cost of a high speed network, compute clusters are able to separate computation from data storage.

Parallel filesystems alleviate the problems of separating the data from the computer where the processing will be executed by spreading files across multiple network connected fileservers allowing access that is sometimes faster than utilizing a computer's local disk for storage while also enabling processing to spread across the availble compute resourses based entirely on the process' grain size.

Current desktop approaches to GIS processing such as ArcGIS are unable to make use of the multi-machine processing environment that compute clusters provide. While reworking these programs to utilize these extended resourses is possible, it is non-trivial. The collaboration capabilities of GRASS GIS[13] have been used to distribute sub-queries among computers. The method described in this paper utilizes multiple instances of GRASS in a master-slave configuration where all participants access a shared data repository or filesystem. The geometries are portioned between the various nodes. Operations are done on the subsets, and the results are merged to produce

the final result.

While the method used to extend GRASS GIS will in fact speed up GIS processing, it has two flaws. First, the overhead of a graphic user environment is required on each node to support the graphical nature of the program, reducing the amount of memory and processing capability available. Second, the entire set of data used must still fit on a single computer to move it in or out of the environment.

A good cluster based GIS processing enviroment needs to be separate from a graphical user interface so that it does not incur addtional overhead on the compute nodes and so that it can interact well with the generally batch-scheduled environment of a cluster. In addition, it must be able to distribute data between all the nodes used such that a single node does not become a limiting factor in the environment's scalability. A variety of data distribution models should also be available so that the data is distributed in a manner consistant with the required processing. Of course, the environment should be able to execute all the standard geospatial operations.

2 Thesis Work

Create an architecturally sound parallel GIS processing enviroment utilizing MPI based on lessons learned from Hadoop based implimentation.

* Load and distribute data (modular for differing domain decompositions) (How to note what decomp is used? struct? pgisData datatype with data and

pgis_file_load_as_distributed_array(file, &pgisdata) load(file, ref to pgisData, decompType) loadChunked(file, chunk#, &pgisData, decompType)

pgisdata-¿data[12][123] to access local data (all are char arrays) * Impliment basic processing queries * select a row * perform geospatial filter (search) - representative of any geospatial function that does not require access to other parts of the dataset * nearest 'x' - multi-dataset or introspective look w/o complicated decomp. * Utilize GEOS library (SimpleFeatures standard) * Inspired by Hadoop and work with hadoop. Created pgis. * Importance of fault recovery? RERUN

Future: * Raster data

Use Case: * User tells enviroment how to load the data (location of data, and decomposition method) If preprocessing is needed for a dataset, the loader will do it and then cache the results to disk. cache used if newer than dataset. * User interacts with the data as needed * Export of distributed data

Pgis has methods to: * load data (partioned or complete copy per task) * Full geospatial functionality (JTS, GEOS)

Develop a fairly generic GIS processing model for cluster compute environments.

I intend to demonstrate two different approaches to executing GIS queries on high performance computing clusters. These approaches will be compared with the database driven SQL query approach in terms of ease of programming and execution time.

The first approach will utilize the standard means of programming for high performance computing clusters, MPI [15]. The second will use the open source version of Google's MapReduce [14], Hadoop [9].

A set of sample queries will be taken from work encountered with the Digital Phoenix project and with other GIS research as available.

The Open Geospatial Consortium has created several standards for capabilities of GIS systems including interoperability. Among these standards are the set of Simple Features standards which define data types and operations for GIS data. This research makes use of several implementations of libraries supporting these standards. This research does not intend to create this functionality, but instead consume it. The libraries used are the Java Topology Suite (JTS) [10] and the GEOS library [11]. GEOS is a port of the JTS and was developed as part of the PostGIS extensions for PostgreSQL.

3 Value and Benefits of Research

While any sort of programming or query model moves away from the seeming simplicity of graphical, clickable, user interfaces, any form of script is easy to save for later replication and thus traceability of the research done. In addition, complicated tasks are more easily repeatable.

The benefits of a programming or query model come with the definite tradeoff of a seemingly increased learning curve for the researchers. With the right model, these approaches will be no harder than writing simple SQL

statements, while proving reduced time to solution on inexpensive and more available resources than the commercial parallel databases offer.

4 Research Plan

The research will progress on two interrelated paths. The first path is determining a sufficient set of GIS operations to validate a processing environment on real world operations. The second path will be developing the MPI and Hadoop environments to execute the set of operations determined. All operations will also be executed using PostGIS to compare for query development and execution times.

An approximate timeline for these activities follows:

1. Fall 2008: Definition of GIS queries
2. Fall 2008 - Early Spring 2009: Development and execution of queries in PostGIS
3. February - April 2009: Development of queries in MPI
4. April 2009 - May 2009: Development of queries in Hadoop
5. June - July 2009: Analysis and Evaluation

5 Completion Criteria

1. A set of GIS operations sufficient to validate an environment against real world operation
2. Working implementations for each operation in each environment.
3. Comparisons of the ease of programability and execution times for each environment

References

[1] http://postgresql.org/

[2] http://www.esri.com/software/arcgis/

[3] http://www.qgis.org/

[4] http://grass.itc.it/

[5] http://urbansim.org/

[6] http://postgis.refractions.net/

[7] http://teradata.com/

[8] http://www.oracle.com/

[9] http://hadoop.apache.org/

[10] http://www.vividsolutions.com/jts/jtshome.htm

[11] http://trac.osgeo.org/geos/

[12] http://digitalphoenix-asu.net/

[13] Fang Huang; Dingsheng Liu; Peng Liu; Shaogang Wang; Yi Zeng; Guoqing Li; Wenyang Yu; Jian Wang; Lingjun Zhao; Lv Pang, "Research On Cluster-Based Parallel GIS with the Example of Parallelization on GRASS GIS," Grid and Cooperative Computing, 2007. GCC 2007. Sixth International Conference on , vol., no., pp.642-649, 16-18 Aug. 2007

[14] Jeffrey Dean; Sanjay Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," OSDI 2004.

[15] http://www.mpi-forum.org/

[16] DeWitt, D. and Gray, J. 1992. Parallel database systems: the future of high performance database systems. Commun. ACM 35, 6 (Jun. 1992), 85-98. DOI= http://doi.acm.org.ezproxy1.lib.asu.edu/10.1145/129888.129894

[17] Robert Pahle and Nathan Kerr. 2009. A data centric framework for research in planning. UPE8.

[18] http://www.openstreetmap.org