GIS PROCESSING ON COMPUTE CLUSTERS

by

Nathan Thomas Kerr

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

ARIZONA STATE UNIVERSITY

August 2009

GIS PROCESSING ON COMPUTE CLUSTERS

by

Nathan Thomas Kerr

has been approved

August 2009

Graduate Supervisory Committee:

Dr. Daniel Stanzione, Chair
Dr. Robert Pahle
Dr. Yi Chen

ACCEPTED BY THE GRADUATE COLLEGE

ABSTRACT

Abstract goes here

Dedication goes here

ACKNOWLEDGEMENTS

Acknowledgments go here.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Chapter 1

INTRODUCTION

GIS processing is important

Geospatial (GIS) simulation and analysis are important processes to understanding and improving our environment, both urban and natural. Geospatial data is made up of a georeferenced geometry such as a point or a polygon at a certain longitude, 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 amount of data increases, so does the processing power required to complete the geospatial analysis, processing, and simulation.

1.1 Current methods

Desktop GIS

Desktop GIS packages such as ArcGIS[1], QuantumGIS[2], and GRASS GIS[3] are commonly used for GIS processing and analysis. While these programs provide graphical interfaces to their GIS capabilities, their capabilities 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[4] faces these same constraints.

Database GIS

An alternative approach to using these desktop programs is to employ a geospatial database like PostGIS[5], ArcSDE[6] or Oracle Spatial[7]. Geospatial databases allow centralized access to, and processing of, geospatial data through query languages such as SQL. As data is stored 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 Frame-work[8] (USF) designed by the Digital Phoenix[9] project group at Arizona State University. Digital Phoenix tries to integrate 3D visualization technology with simulated and gathered GIS data to better understand the impacts of urban planning decisions.

1.2 Significance

Limitations of single processor implementations

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 service to the capabilities of private companies and individuals even to the point of open source style maps such as OpenStreetMap[10].

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 limitations 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.

1.3 Attempted solutions

Parallel DB

One method of using multiple computers to perform the required processing is the use of parallel databases[11]. Parallel databases should be able to spread both data storage and processing across multiple computers transparently from the view of the SQL query programmer. Several commercial databases such as TeraData[12] and Oracle[7] provide support for this method of operation. Open source databases such as MySQL[13] and PostgreSQL[14] currently do not support this methodology. Parallel databases generally do not require a shared data repository or filesystem.

Parallel database techniques have been used to build a scalable geospatial database system[15, 16]. Data is spread between computers using round-robin, hash, or spatial partitioning. 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 dependent on the speed of 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 commutation requirements is often referred to as grain size. Coarser grained processes have more computation per data record, while finer grained processes have little computation for each data record.

Databases excel at working with indexed data while allowing 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. The processing operations this research examines do not require these restrictions, and as such a more efficient system can be created.

Using databases can also limit the reusability 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 accommodate 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 significant 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[17]. By separating compute and storage resources at the cost of a high speed network, compute clusters are able to separate computation from data storage. Compare to SAN approaches

Parallel filesystems alleviate the problems of separating the data from the computer where the processing will be executed by spreading files across multiple network con-

nected fileservers allowing access that is sometimes faster than utilizing a computer's local disk for storage while also enabling processing to spread across the available compute resources based entirely on the process' grain size.

Parallel GRASS

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 resources is possible, it is non-trivial. GRASS GIS was reworked[18] to use its collaboration features 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.

Problems with desktop programs on clusters

While the method used to extend GRASS GIS will in fact speed up GIS processing, it has two flaws. First, GRASS is designed to be used in an interactive mode rather than a batch or script driven approach. Second, the entire set of data used must still fit on a single computer to move it in or out of the environment.

1.4 Needed solution, cluster programming model

batch

mpi

scalable architecture

spmd

A good cluster based GIS processing environment needs to be separate from a graphical user interface so that it does not incur additional 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 consistent with the required processing. Of course, the environment should be able to execute all the standard geospatial operations as defined by the Open Geospatial Consortium in their Simple Features[19] standard.

Thesis statement goes here

This thesis will evaluate a novel method of processing geospatial data, called PGIS, which treats datasets as an atomic unit (as opposed to databases where records are atomic). By dispensing with the requirements dictated by record level atomicity, software complexity is reduced significantly allowing for a simpler implementations with less overhead.

The first step in PGIS is to distribute the data between the computers participating in the operation. PGIS is a data parallel system, meaning that a dataset can be split up onto multiple computers and the operations on the data require little if any data on another computer. I will refer to this dataset as the primary dataset. For some operations, a secondary dataset is required in its entirety. For instance if the operation is to find the nearest record to a given georeferenced point. If both the primary and secondary datasets were spatially distributed between the computers as described in [15] additional preprocessing would be needed. The limitations of having the entire secondary dataset on each computer are increased memory footprint and number of records that have to be looked at. While PGIS does not implement a spatial distribution method, one could be added in future work.

After the data is distributed, a user defined operation is executed. This operation should iterate over the subset of records from the primary dataset, creating a new dataset

or modifying the existing one. This is also where the secondary dataset can be referenced to perform operations like nearest neighbor. After the user defined operation is completed, either the modified primary dataset or a newly created dataset is saved.

How this document goes about supporting the thesis statement.

PGIS will be evaluated for speedup and scaleup, using equivalent operations executed in PostGIS as a reference. Speedup is obtained when more execution speed is gained by adding additional processors, while keeping the same dataset. Scaleup measures how well a program can handle different sizes of datasets while adding more processors at the same time the dataset size grows.

A series of operations meant to be representative of real-world uses will be used. The first set of operations look at single record create, read, update, and destroy. This set of operations is meant to show the strength of the database approach and the weakness of this new method. The second set of operations will extract a subset of a datset using a geospatial operation to determine if a record is part of the subset. This operation should show the strength of the new method as compared to the database approach. The third, and final, operation will find the spatially nearest feature of one dataset for each record of a dataset. This operation is the hardest for the database approach and shows the strength of the new approach. Full details for datasets and operations used will be given in the thesis.

Chapter 2

RELATED WORK

2.1 Desktop GIS

ArcInfo, QuantumGIS, GRASS GIS

Desktop GIS packages such as ArcGIS[1], QuantumGIS[2], and GRASS GIS[3] are commonly used for GIS processing and analysis. While these programs provide graphical interfaces to their GIS capabilities, their capabilities 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[4] faces these same constraints.

2.2 Database GIS

PostGIS, Oracle Spatial, Client-Server Paradise

An alternative approach to using these desktop programs is to employ a geospatial database like PostGIS[5], ArcSDE[6] or Oracle Spatial[7]. Geospatial databases allow centralized access to, and processing of, geospatial data through query languages such as SQL. As data is stored 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 Frame-work[8] (USF) designed by the Digital Phoenix[9] project group at Arizona State University. Digital Phoenix tries to integrate 3D visualization technology with simulated and gathered GIS data to better understand the impacts of urban planning decisions.

2.3 GIS libraries

JTS, GEOS

2.4 Parallel DB

One method of using multiple computers to perform the required processing is the use of parallel databases[11]. Parallel databases should be able to spread both data storage and processing across multiple computers transparently from the view of the SQL query programmer. Several commercial databases such as TeraData[12] and Oracle[7] provide support for this method of operation. Open source databases such as MySQL[13] and PostgreSQL[14] currently do not support this methodology. Parallel databases generally do not require a shared data repository or filesystem.

Parallel database techniques have been used to build a scalable geospatial database system[15, 16]. Data is spread between computers using round-robin, hash, or spatial partitioning. 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 dependent on the speed of 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 commutation requirements is often referred to as grain size. Coarser grained processes have more computation per data record, while finer grained processes have little computation for each data record.

Databases excel at working with indexed data while allowing 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. The processing operations this research examines do not require these restrictions, and as such a more efficient system can be created.

Using databases can also limit the reusability 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 accommodate 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 significant 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[17]. By separating compute and storage resources at the cost of a high speed network, compute clusters are able to separate computation from data storage. Compare to SAN approaches

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 available compute resources based entirely on the process' grain size.

Paradise

2.5 Parallel GRASS

Other converted programs?

2.6 MRGIS MapReduce GIS

Chapter 3

REQUIREMENTS

batch mode processing

data distribution options

standard geospatial operations

scalable, high performance

dataset centric

A good cluster based GIS processing environment needs to be separate from a graphical user interface so that it does not incur additional 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 consistent with the required processing. Of course, the environment should be able to execute all the standard geospatial operations as defined by the Open Geospatial Consortium in their Simple Features[19] standard.

3.1 easy to use to form new operations

Sample program:

initClusterGIS

loadData (distributed/replicated)

process (user code here)

saveData (distributed/replicated)

finalizeClusterGIS

Chapter 4

DESIGN

How to fulfill requirements

- **Architecture of solution**
- **Split dataset across tasks (MPI I/O)**
- **Each task works on their own part**
- **MPI to communicate if needed**

Chapter 5

EXPERIMENTAL SETUP

How to show how well requirements were filled

Some of the needed research has already been completed. After determining the query set and creating the PostGIS implementation of it, the processing environment was prototyped in Hadoop[20], an opensource implementation of Google's MapReduce[21] framework. MapReduce works by utilizing two user defined function, map and reduce. For each record of the primary dataset the map function is called. This process is made parallel by splitting the data up among several computers and running the individual map functions on those computers. After the map function is complete, the resulting set of data is passed to several reduce functions. Each instance of the reduce function receives all the data from the resulting dataset associated with the same key. The lessons learned from this prototyping phase will be applied to the MPI implementation.

5.1 Operation set

These operations are representative of problem space.

Data access based: parallelization is based off data decomp, so operations are

Section	1 Geometry	2 Geometries
6.1.2 (Geometry)	16	15
6.1.4 (Point)	4	0
6.1.6 (Curve)	5	0
6.1.7 (LineString, Line, LinearRing)	2	0
6.1.8 (MultiCurve)	2	0
6.1.10 (Surface)	3	0
6.1.11 (Polygon, Triangle)	3	0
6.1.12 (PolyhedralSurface)	4	0
6.1.13 (MultiSurface)	3	0
6.1.15 (Relational Operators)	0	9
Total	42	24

Table 5.1: OGC Methods by Number of Required Geometries

too.

OGC standard requirements - survey of methods requiring 1, 2, or more data points.

http://www.opengeospatial.org/standards/sfa (Simple Feature Access - Part 1: Common Architecture 1.2.0)

All operations must be performed as if they were part of a series of operations.

1. Create a new record

PostGIS: insert... Hadoop: map: emit all record giving them the same key (forcing one reduce); reduce emits all received records with the addition of one ClusterGIS: emits all records and the new one

2. Read an existing record (by id)

PostGIS: select ... where id = Hadoop: map: emit only the record with the correct id; reduce: identity ClusterGIS: emit only the record with the correct id;

3. Update

PostGIS: update.... Hadoop: map: emit all records, updating the one with the correct id; reduce: identity ClusterGIS: emit all records updating the correct one

4. Destroy

PostGIS: delete ... Hadoop: map: emit all records except for the one with the correct id; reduce: identity ClusterGIS: emit all record except the the correct one

5. Filter

PostGIS: select...where Hadoop: map: emit only records matching the requested criteria; reduce: identity ClusterGIS: emit only records matching the requested criteria

6. Find Nearest

PostGIS: Crazy set of queries (nested would have been nice, but too slow) Hadoop: map: loop through secondary dataset emitting id of primary and secondary; reduce: identity ClusterGIS: loop through primary and secondary dataset, emitting the id of primary and secondary which match

7. Chaining (Filter the datasets, then find the nearest in the filtered data)

PostGIS: Modify crazy set of queries to use the filter Hadoop: filter secondary dataset on load; map: only search for nearest for filtered data, skip others; reduce: identity ClusterGIS: remove nodes in the datasets not matching the filter, find nearest as before

5.2 Dataset Description

Full datasets; 34k employers, 1.2m parcels

sub datasets, how to generate from full

listing of datsets used

5.3 Hardware setup

Saguaro

5.4 implementations

5.4.1 PostGIS

5.4.2 Hadoop Prototype

5.4.3 ClusterGIS

Chapter 6

RESULTS

6.1 Performance Analysis

vary procs, maintain data

vary data, maintain procs

vary data, vary procs

6.2 Comparisons

PostGIS as baseline

PostGIS to Hadoop

Hadoop to ClusterGIS

PostGIS to ClusterGIS

Chapter 7

RECOMMENDATIONS

Synthesis

7.1 Applicable Problem Spaces

Analysis of problem types that are good/bad for this approach

Chapter 8

CONCLUSION

By simplifying the requirements to handle process GIS operations in parallel, the cost to implementing parallel solutions will decrease, enabling more parallel processing methods to be created. This parallel methods will be able to take advantage of the increased processing powers made available through compute clusters.

Another benefit of this research is a classification of geospatial operations based on data access requirements and a sample set of queries to evaluate the efficiency of a parallel GIS processing implementation.

8.1 Future Work

additional decomposition methods, combined with alternative MPI communicators

addition of preprocessing methods (indexing, etc)

chunking of replicated dataset

Bibliography

[1] T. Ormsby and R. Burke, *Getting to Know ArcGIS Desktop: Basics of ArcView, ArcEditor, and ArcInfo*. ESRI, Inc., 2nd ed., 2004.

[2] J. Gray, "Quantum gis: the open-source geographic information system," *Linux J.*, vol. 2008, no. 172, p. 8, 2008.

[3] M. Neteler and H. Mitasova, *Open source GIS: a grass GIS approach*. Springer, 3rd ed., 2002.

[4] P. Waddell, A. Borning, M. Noth, N. Freier, M. Becke, and G. Ulfarsson, "Microsimulation of urban development and location choices: Design and implementation of urbansim," *Networks and Spatial Economics*, vol. 3, pp. 43–67, 2003.

[5] D. Blasby, "Building a spatial database in postgresql," in *Open Source Database Summit*, 2001.

[6] R. West, *Understanding ArcSDE.* ESRI, Inc., 2001.

[7] S. Ravada and J. Sharma, "Oracle8i spatial: Experiences with extensible databases," in *SSD '99: Proceedings of the 6th International Symposium on Advances in Spatial Databases*, (London, UK), pp. 355–359, Springer-Verlag, 1999.

[8] R. Pahle and N. Kerr, "A datacentric framework for research in planning," in *UPE8, The 8th International Symposium (UPE 8) of the International Urban Planning and Environment Association*, 2009.

[9] S. Guhathakurta, Y. Kobayashi, M. Patel, J. Holston, T. Lant, J. Crittenden, K. Li, G. Konjevod, and K. Date, "Digital phoenix project: A multidimensional journey through time." Not published, 2006.

[10] M. Haklay and P. Weber, "Openstreetmap: User-generated street maps," *Pervasive Computing, IEEE*, vol. 7, pp. 12–18, Oct.-Dec. 2008.

[11] D. DeWitt and J. Gray, "Parallel database systems: the future of high performance database systems," *Commun. ACM*, vol. 35, no. 6, pp. 85–98, 1992.

[12] "Dbc/1012 data base computer concepts & facilities," tech. rep., Teradata Corp Document No C02-0001-00, 1983.

[13] "Mysql 6.0 reference manual," manual, MySQL AB, 2008.

[14] M. Stonebraker and G. Kemnitz, "The postgres next generation database management system," *Commun. ACM*, vol. 34, no. 10, pp. 78–92, 1991.

[15] J. Patel, J. Yu, N. Kabra, K. Tufte, B. Nag, J. Burger, N. Hall, K. Ramasamy, R. Lueder, C. Ellmann, J. Kupsch, S. Guo, J. Larson, D. De Witt, and J. Naughton, "Building a scaleable geo-spatial dbms: technology, implementation, and evaluation," in *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 336–347, ACM, 1997.

[16] D. J. DeWitt, N. Kabra, J. Luo, J. M. Patel, and J.-B. Yu, "Client-server paradise," in *VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases*, (San Francisco, CA, USA), pp. 558–569, Morgan Kaufmann Publishers Inc., 1994.

[17] P. J. Braam, "Lustre file system," white paper, Cluster File Systems, Inc., 2007.

[18] F. Huang, D. Liu, P. Liu, S. Wang, Y. Zeng, G. Li, W. Yu, J. Wang, L. Zhao, and L. Pang, "Research on cluster-based parallel gis with the example of parallelization on grass gis," in *Grid and Cooperative Computing, 2007. GCC 2007. Sixth International Conference on*, pp. 642–649, Aug. 2007.

[19] "Opengis implementation specification for geographic information - simple feature access - part 1: Common architecture," standard, Open Geospatial Consortium, Inc., 2006.

[20] G. Mackey, S. Sehrish, J. Bent, J. Lopez, S. Habib, and J. Wang, "Introducing map-reduce to high end computing," in *Petascale Data Storage Workshop, 2008. PDSW '08. 3rd*, pp. 1–6, Nov. 2008.

[21] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.