On Parallelizing Large Spatial Queries Using Map-Reduce

Umesh Bellur

GISE Lab, Department of Computer Science
Indian Institute of Technology Bombay, Powai, Mumbai 400076, India
umesh@cse.iitb.ac.in

Abstract. Vector Spatial data types such as lines, polygons or regions etc usually comprises of hundreds of thousands of latitude-longitude pairs to accurately represent the geometry of spatial features such as towns, rivers or villages. This leads to spatial data operations being computationally and memory intensive. A solution to deal with this is to distribute the operations amongst multiple computational nodes. Parallel spatial databases attempt to do this but at very small scales (of the order of 10s of nodes at most). Another approach would be to use distributed approaches such as Map-Reduce since spatial data operations map well to this paradigm. It affords us the advantage of being able to harness commodity hardware operating in a shared nothing mode while at the same time lending robustness to the computation since parts of the computation can be restarted on failure. In this paper, we present HadoopDB - a combination of Hadoop and Postgres spatial to efficiently handle computations on large spatial data sets. In HadoopDB, Hadoop serves as a means of coordinating amongst various computational nodes each of which performs the spatial query on a part of the data set. The Reduce stage helps collate the result data to yield the result of the original query. We present performance results to show that common spatial queries yields a speedup that nearly linear with the number of Hadoop processes deployed.

Keywords: MapReduce, Hadoop, postGIS, Spatial Data, HadoopDB.

1 Introduction

A Geographic information system (GIS) is one that captures, stores, analyzes, manages and presents spatial data along with relevant non spatial information. A GIS forms the core of many applications in areas as varied as agriculture to consumer applications such as location based services. Today, many computer applications, directly or indirectly, are based on carrying out spatial analysis at the back-end. Spatial analysis involve spatial operations to be performed on spatial data. We represent the spatial features such as roads, towns, cities etc as Vectored data. Vector data is collection of latitude-longitude pairs called Geospatial points, structured into a format so as to represent the geometry of spatial features. An example would be the use of vectored polygons to represent...
city or state boundaries. For example, to represent the road network of the state of Arizona in the USA, we require approximately ten million points, each of which is a coordinate involving a latitude and longitude. The number of geospatial coordinates required to represent the geometry of real world objects varies from few hundreds to tens of thousands. Spatial operations such as overlapping test (to check whether two areas overlap each other or not) etc are performed on a set of vector spatial data sets. These operations are generally the implementation of geometric algorithms. Because of the enormous number of points required to represent a single spatial object and complexity of geometric algorithms, carrying out spatial computation on real world data sets has been resource-intensive. A core-duo, 2G machine takes about 75-85% CPU consumption for spatial join queries. We therefore consider spatial operations to be a potential candidate for parallelism.

Parallel spatial DBMSs such as Oracle spatial are being widely used for carrying out parallel computation of spatial data across a cluster of machines. Parallel DBMSs design have been optimized to yield high performance but do not score well in terms of scalability. Asterdata (www.asterdata.com), a parallel database known to posses one of the best scalability in parallel database community is scalable to around 330-350 nodes. In parallel DBMSs, the intermediate results of query are pipelined to next query operator or another sub-query without being written to disk. Now if any sub-query fails, the intermediate results processed so far are lost and entire query have to be restarted again. Not writing intermediate data onto disks, results in high performance but at the same time avoid parallel DBMS from exhibiting good fault tolerance. With the increase in the size of a cluster of commodity machines, the probability of node or task failure also increase and this failure is likely to become a frequent event in case the parallel DBMS cluster size is increased to the order of few hundreds of nodes. This would result in a significant degradation in the performance of parallel DBMSs. Thus, poor fault tolerance capability puts an upper bound on the cluster size of parallel DBMSs (up to few tens of nodes), as a result of which parallel DBMSs have limited scalability.

MapReduce [1], on the other hand, provides a framework for processing large volumes of data, of the order of hundreds of terabytes. The scalability and fault tolerance features of MapReduce enable us to use a large number of commodity machines for carrying out data intensive computations cheaply. The Map-Reduce parallel programming model does not necessitate a programmer to understand and control the parallelism inherent in the operation of the paradigm.

In this paper we present the design of a shared nothing, data distributed, spatial query processing system that we term HadoopDB. We employ the Hadoop MapReduce libraries to process spatial data extracted from a spatial DB such as postGIS. We have written a query converter that takes a SQL like query at the front end and turns it automatically into a map reduce job that uses data from a set of postGIS instances in the back end in which the spatial data set to be operated on is distributed. We show that we can achieve near linear speed