Hadoop Based Enhanced Cloud Architecture For Bioinformatics Algorithms

[1] Hamoud Alshammari, [2] Hassan Bajwa, [1] Jeongkyu Lee
[1] Department of Computer Science, University of Bridgeport, CT.
[2] Department of Electrical Engineering, University of Bridgeport, CT.

Abstract

Explosion of biological data due to large-scale genomic research and advances in high throughput data generation tools result in massive distributed datasets. Analysis of such large non-relational, heterogeneous, and distributed datasets is emerging challenge in data driven biomedical industries. Highly complex biological data require unconventional computational approaches and knowledge-based solutions. Distributed datasets need to be reduced to smaller datasets that can be efficiently queried. Since genomic and biological data is generated in large volume and is stored in geographically diverse locations, distributed computing on multiple clusters, our objective here is to assess the feasibility of using Cloud based platform to analyze genomic big data. In this project we report on the limitation of cloud based platform in the analysis of genomic data.

Introduction

Bioinformatics applications usually require large complex amounts of data processing and computational capabilities. A large distributed file based processing is adopted in this project to process large data files, which can scale up to few terabytes. Hadoop based cloud architecture is composed of Hadoop Distributed File System (HDFS), MapReduce programming model and Apache Zookeeper as coordination service. HDFS cluster is composed of a centralized indexing system called NameNode and its data processing units called DataNodes; together they form a unique distributed file system. NameNode plays an important part in supporting the Hadoop Distributed File System by maintaining a File-Based block index map, this map is responsible to locate all the blocks related to the HDFS. HDFS is the primary storage system, HDFS creates multiple replicas of data blocks and is further responsible to distributes data blocks throughout a cluster to enable reliable, extremely rapid computations.

Hadoop Distributed File System (Figure 1) allows the distribution of the data set into many machines across the network that can be logically combed for processing. Hadoop framework leverages on large-scale data analysis by allocating data-blocks among distributed DataNodes.

HDFS is architected to have the block fault and replication tolerance. NameNode is responsible to maintain a healthy balance between disk processing on various DataNodes and has an ability to restore the failed operations on the remote blocks. Data Locality is achieved through cloud file distribution, the file processing is done local to each machine, and any failed reads from the blocks are recovered through block replication. The process of selecting the mappers and the reducers is done by the JobTracker immediately after lunching a job.

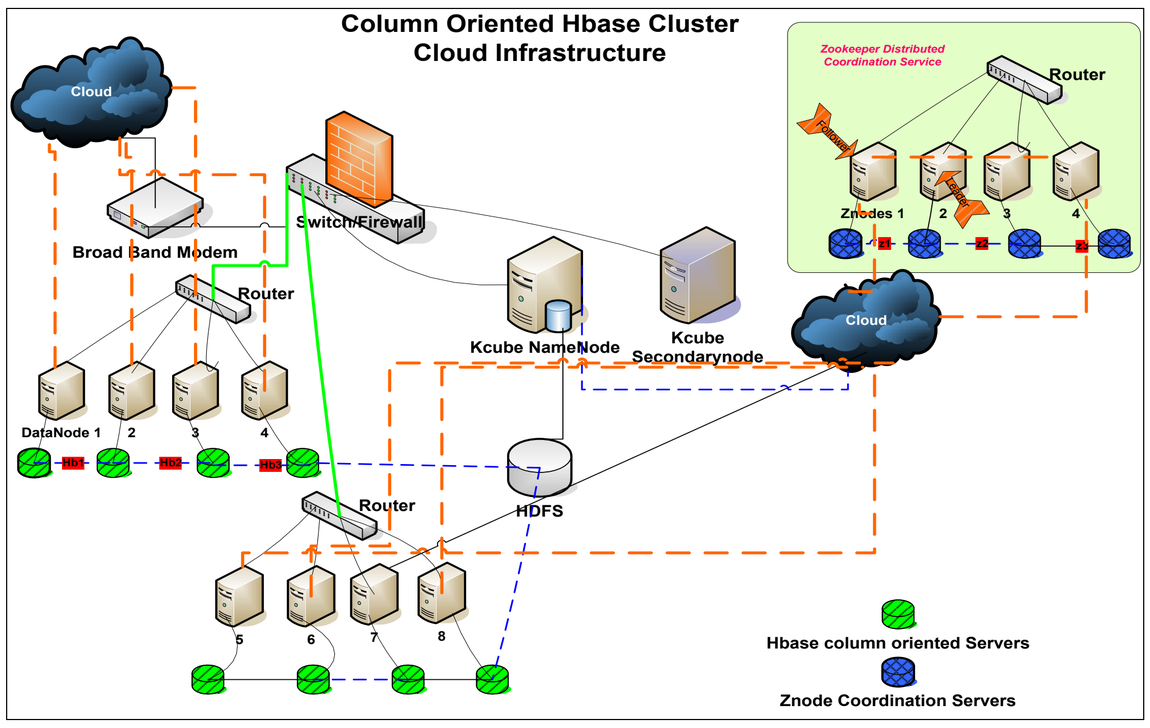

Figure 1: Hadoop Architecture.

Hadoop and Bioinformatics Data

It is also important to realize that any bioinformatics tools such as (BLAST, FASTA etc.) can be processed in parallel, but most of the users are not trained to modify the existing applications to incorporate parallelism effectively. This project leverages on HDFS distributed computing model utilizing various commodity servers and Hadoop Apache Map-Reduce frameworks to explore genome data.

Example: Finding a Specific Pattern Sequence in DNA Genome

Background:

Massive genomic data, driven by unprecedented technological advances in genomic technologies, have made genomics a computational research area. Next generation sequencing (NGS) technologies produce high throughput short read (HTSR) data at a lower cost. Scientists are using innovative computational tools that allow them rapid and efficient data analysis. DNA genome sequence consists of 24 chromosomes. The compositions of nucleotides in genomic data determine various traits such as personality, habits, and inheritance characteristics of species. Finding sequences, similarities in sequences, subsequences or mutation of sequences are important research area in genomic and bioinformatics.

Scientists need to find subsequences within chromosomes to determine either some diseases or proteins frequently. Each chromosome has many known genes and many unknown sequences. For example, chromosome one consists of about 249 million of nucleotide base pairs, which represent about %8 of the total DNA in human cells. The total number of genes in chromosome 1 is about 4,316 genes each one has different length of base pairs.

Problem definition and solution using Hadoop

Searching for sequences or mutation of sequences in a large unstructured dataset can be both time consuming and expensive. Sequence alignment algorithms are often used to align multiple sequences. Due to memory limitation, aligning more than three – four sequences is often not allowed by traditional alignment tools.

In this project we proposed using Hadoop MapReduce to align genomic data. We tested MapReduce for sequence alignment by building a complete MapReduce program that takes the pattern sequence as a key and find this sequence in a chromosome and also in the whole DNA sequence. We executed the job within a cluster that has three DataNodes. As expected Hadoop cluster with three nodes was able to search human genome much faster than single node, it is expected that search time will reduce as number of DataNodes are increased in the cluster.

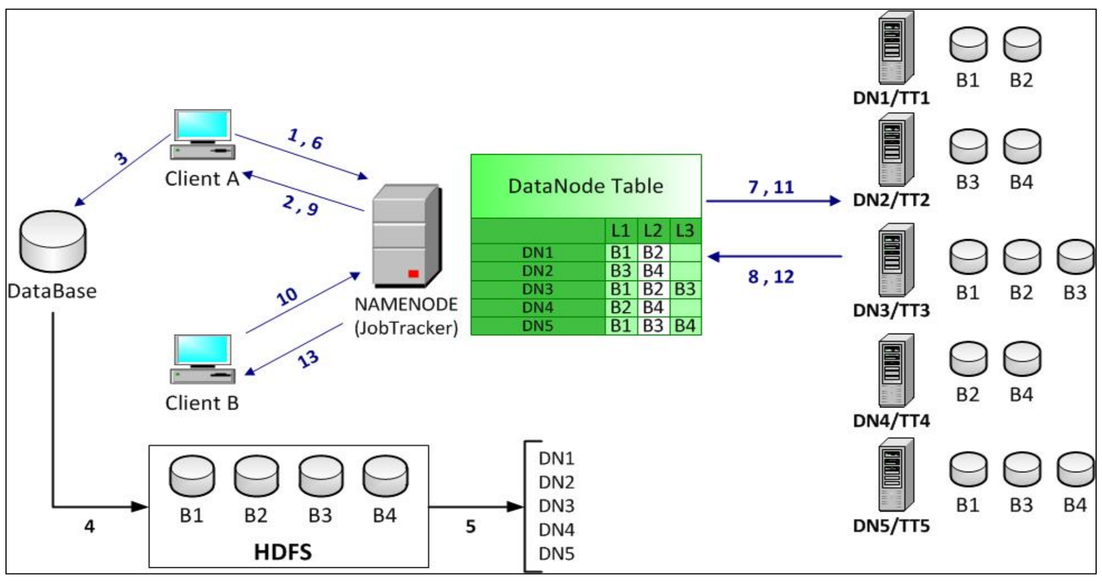

Figure 2: Current Hadoop/MapReduce Workflow.

In current architecture, data follows the concept of write-one read-many, so there is no ability to do any changes in the source file in HDFS. However, each job has to get the data from all blocks that store the source data file in HDFS. Some research groups have already presented solution to address issue of latency while reading data from DataNodes. In current Hadoop/MapReduce workflow (Figure 2) multiple jobs that need the same data set work independent of each other. We also noticed that searching for the same sequence require same amount of time each time we execute the job, also searching for the subsequence of a sequence that has already been searched require the same amount of time.

Figure 2 shows the steps that each job's workflow. Numbers 1-5 explains the steps where the process of uploading the data file/files to the HDFS in the cluster. When the data file/files sent to be uploaded, different blocks will be created to represent data in the cluster [B1, B2, B3, B4]. These blocks will be sent to the DataNodes [DN1, DN2, DN3, DN4, DN5] and replicated three times (default replication value in Hadoop).

After that, any job in the cluster can use this file as a resource of data to get a solution. So, steps 6-9 show the workflow of the first job that has sent from Client A to the NameNode, which then send the job as tasks to all DataNodes that carries the blocks. Then the result gets back to the Client A. Steps 10-12 show the same as workflow as steps 6-9 but for Client B.

Problem Definition

As we see in Figure 2, each job has to go throw same steps (6-9 for Client A) to get the results. However, there might be some common features between some jobs that makes the jobs can get benefit of each other. In other words, jobs that share some common features like same data source, same job parameters, or same target can build their jobs based on the results of each other, which give the related jobs to not go throw the whole steps or at least read the data from specific sources.

Proposed Solution

In current Hadoop architecture, NameNode knows the location of data blocks in HDFS. NameNode is also responsible for assigning jobs to the clients. Knowing which DataNode contains these blocks, with the required data. NameNode should be able to direct the jobs to read the specific DataNodes without going through all DataNodes. This solution solves the problem of wasting some computation costs and time. However, there are some problem with this proposed solution that should be considered as naming of the jobs, passing common parameters and dealing either with the blocks or DataNodes themselves.

Conclusion

In this project we present an enhanced Hadoop architecture to reduce computation by utilizing "Common Features" before performing redundant computation. The enhanced Hadoop architecture allows the jobs to share these common features among them. The common features describe the contents of data in blocks and can be used to determine DataNodes that store the required data. However, we still need to implement the idea of having common features to reduce the computation time and cost comparing with the current Hadoop architecture.