Hadoop Based Enhanced Cloud Architecture

Hamoud Alshammari[1], *Hassan Bajwa*[2] *and Jeongkyu Lee*[1]
Department of Computer Science [1]
Department of Electrical Engineering[2]
221 University Ave, University of Bridgeport,
Bridgeport, CT, USA

***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 paper we report on the limitation of cloud based platform in the analysis of genomic data.**

I. 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 (Fig 1) 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 [1]. ZooKeeper is critical component of the infrastructure, it provide coordination and messaging across applications [2]. The ZooKeeper capabilities include naming, distributed synchronization, and group services. Hadoop framework leverages on large-scale data analysis by allocating data-blocks among distributed DataNodes.

Hadoop Distributed File System (Fig 1 and Fig2) allows the distribution of the data set into many machines across the network that can be logically combed for processing.

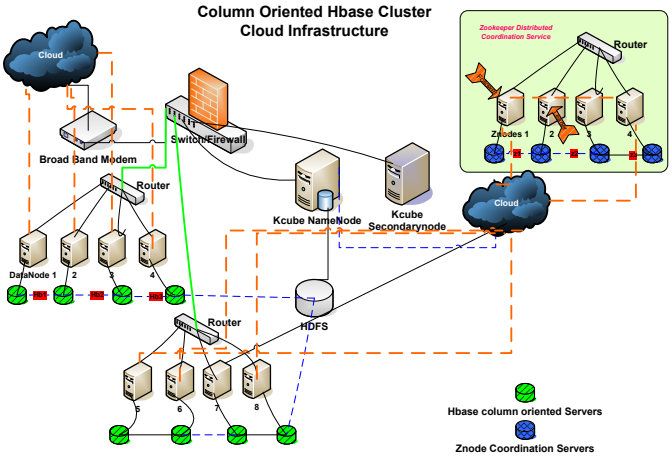

Figure 1: Hadoop Architecture

HDFS adopted Write-Once-Read-Many model to store data in distributed DataNodes. NameNode is responsible for maintaining namespace hierarchy, managing datablocks and DataNodes mapping. Once job information is received from the client, NameNode provides a list of available data nodes for the job. NameNode maintains the list of available data nodes and is responsible to update the index list when a DataNode is unavailable of failed due to hardware or network

issues. A heartbeat is maintained between the NameNode and the DataNodes to check the keep-alive status and health of the HDFS. Client writes data directly to the DataNode (fig 2). 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 [3, 4].

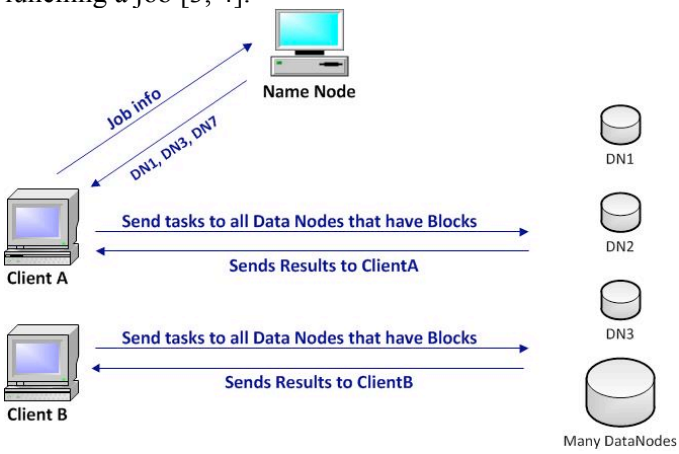

Figure 2: DataNodes and Task Assignment

A client operating on the HDFS has network file transparency and the distribution of blocks on different machines across the Cloud is transparent to the client. HDFS is oriented towards hardware transparency. Processing DataNodes can be commissioned or decommissioned dynamically without affecting the client.

II. HADOOP AND BIOIFNRMATICS 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 [5]. This project leverages on HDFS distributed computing model utilizing various commodity servers and Hadoop Apache Map-Reduce frameworks to explore genome data.

Example: fininding a Specific Pattern Sequence in DNA Genome

1) 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 [6]. Scientists are using innovative computational tools that allow them rapid and efficient data analysis [7, 8] . 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 [9]. 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.

2) The 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 suing 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.

III. LIMITTAION OF HADOOP

Many Big Data problems, especially genomic data, deal with similarities/sequences and sub-sequences searches. If a "sub-sequence" is found in a specific blocks in a DataNode, sequence containing that subsequence can only exist in the same DataNode. Since current Hadoop Framework does not support caching of data, it ignores location of DataNode with sub-sequence and read data from all DataNodes for every new job [10]. Shown in Figure 2 Client A and Client B are searching for similar sequence in BigData. Once Client A finds the sequence, Clint B will also go through the whole BigData again to find the same results. Since each job is independent clients do not share results. Any client looking for Super sequence with sub-sequence already searched will have to go through the BigData again. Thus the cost to perform the same job will stay the same each time.

IV. CURRENT ARCHITECTURE

In current Hadoop and MapReduce architecture, the client first sends a job to the cluster administrator, which is the NameNode or the master of the cluster. Before that, the data source file should be uploaded to the Hadoop Distributed File System by dividing the BigData into blocks that have the same size of data, usually 64 or 128 MB for each block. Then, these blocks are distributed among different Data Nodes within the cluster. Any job now has to have the name of the data file in HDFS, the source file of MapReduce code (e.g. Java file), and the name of the file that the result will be stored in also in HDFS.

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 (Figure 3). 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 [11]. In current Hadoop/MapReduce architecture (Figure 3) multiple job 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.

Hadoop MapReduce Workflow:

1. Client A sends a request to NameNode that includes the need to copy the data files to DataNodes with 3 copies.
2. NameNode replays with the IP address of five DataNodes (DN1, DN2, DN3, DN4, DN5).
3. Client A accesses the shared data files (DB).
4. Client A reformats the data files into HDFS format in many Blocks (B1, B2, B3, B4).
5. Client A sends the blocks to the DataNodes with three copies for each block.
6. Client A sends a MapReduce-job1 to the JobTracker daemon with the name of the data.
7. JobTracker sends the job to all TaskTrackers who hold the blocks of the data.
8. Each TaskTracker executes a specific task on each block and sends the results back to the JobTracker.
9. JobTracker sends the final result to Client A.
10. Client B sends a new MapReduce-job2 to the JobTracker with the same name of the data.
11. JobTracker sends job2 to all TaskTrackers.
12. TaskTrackers execute the tasks and send the results back to the JobTracker.
13. JobTracker sends the final result to Client B.

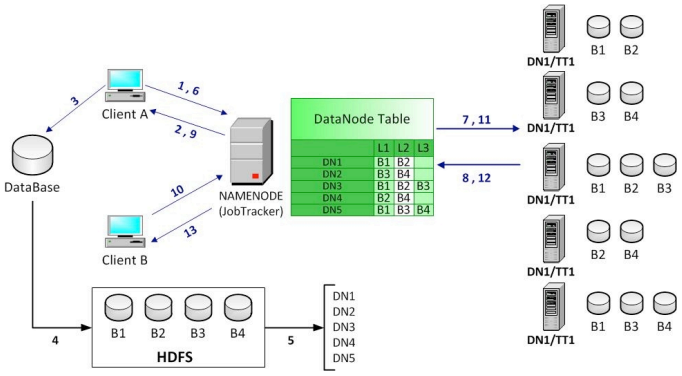

Figure 3: Current Hadoop/MapReduce Architecture

IV. 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.

REFERENCES

[1] A. Pavlo, E. Paulson, A. Rasin, D. J. Abadi, D. J. DeWitt, S. Madden, and M. Stonebraker, "A comparison of approaches to large-scale data analysis," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, 2009, pp. 165-178.

[2] Hadoop, "Hadoop: http://hadoop.apache.org/zookeeper/, accessed on 15 June 2010," 2010.

[3] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, J. Gerth, J. Talbot, K. Elmeleegy, and R. Sears, "Online aggregation and continuous query support in mapreduce," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pp. 1115-1118.

[4] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy, and R. Sears, "MapReduce Online," in *NSDI*, p. 20.

[5] S. Leo, F. Santoni, and G. Zanetti, "Biodoop: Bioinformatics on Hadoop, Parallel Processing Workshops, International Conference on, pp. 415-422, 2009 International Conference on Parallel Processing Workshops, 2009," 2009.

[6] Y. Liu, B. Schmidt, and D. L. Maskell, "DecGPU: distributed error correction on massively parallel graphics processing units using CUDA and MPI," *BMC bioinformatics,* vol. 12, p. 85.

[7] A. McKenna, M. Hanna, E. Banks, A. Sivachenko, K. Cibulskis, A. Kernytsky, K. Garimella, D. Altshuler, S. Gabriel, and M. Daly, "The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data," *Genome research,* vol. 20, pp. 1297-1303.

[8] P. Khatri and S. DrÄƒghici, "Ontological analysis of gene expression data: current tools, limitations, and open problems," *Bioinformatics,* vol. 21, pp. 3587-3595, 2005.

[9] H. Mathkour and M. Ahmad, "Genome Sequence Analysis: A Survey," *Journal of Computer Science,* vol. 5, 2009.

[10] A. Matsunaga, M. Tsugawa, and J. Fortes, "CloudBLAST: Combining MapReduce and Virtualization on Distributed Resources for Bioinformatics Applications," in *eScience, 2008. eScience '08. IEEE Fourth International Conference on*, 2008, pp. 222-229.

[11] B. Palanisamy, A. Singh, L. Liu, and B. Jain, "Purlieus: locality-aware resource allocation for MapReduce in a cloud," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, p. 58.