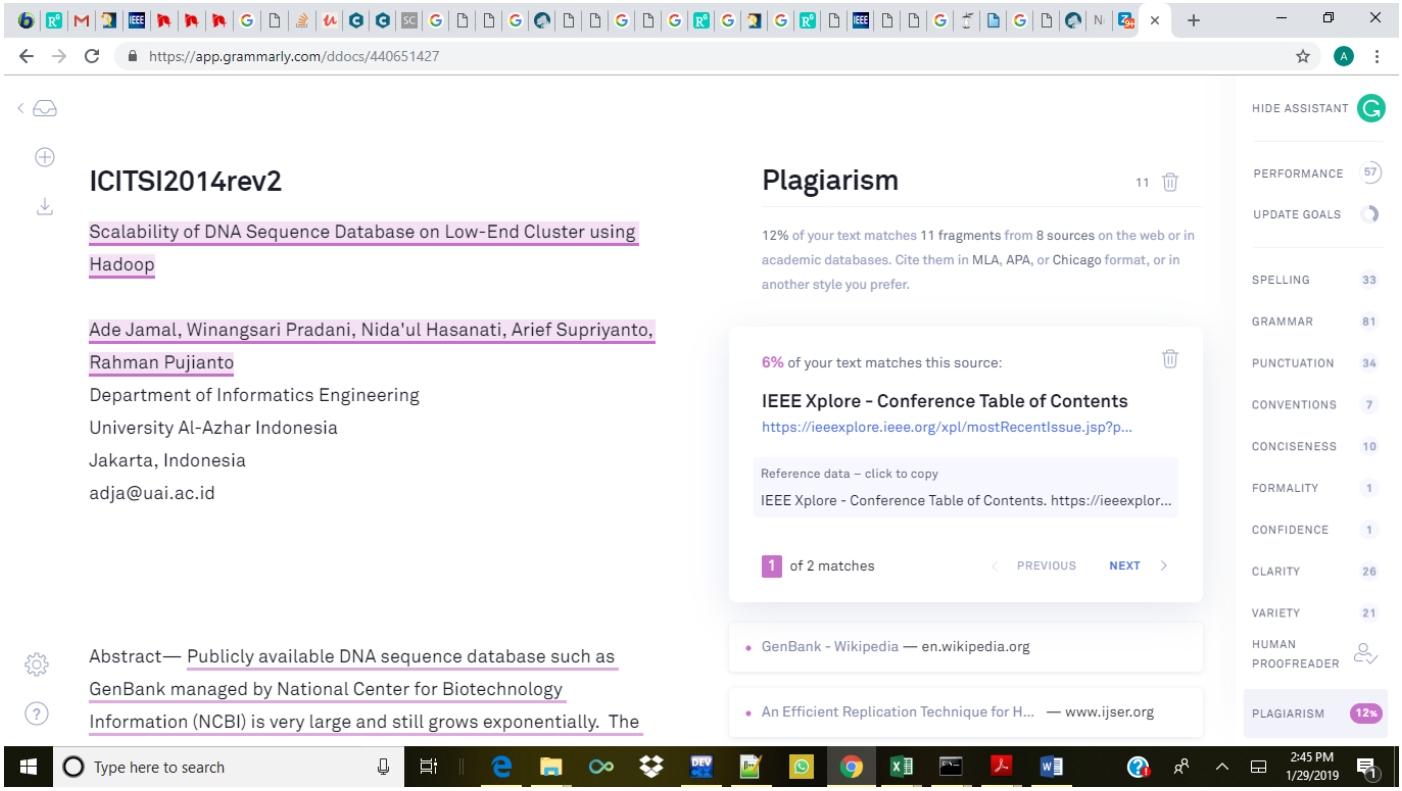


Scalability of DNA Sequence Database on Low-End Cluster using Hadoop

Ade Jamal, Winangsari Pradani, Nida'ul Hasanati, Arief Supriyanto, Rahman Pujianto
Department of Informatics Engineering
University Al-Azhar Indonesia
Jakarta, Indonesia
adja@uai.ac.id

***Abstract*— Publicly available DNA sequence database such as GenBank managed by National Center for Biotechnology Information (NCBI) is very large and still grows exponentially. The sequence data are stored in flat file format grouped in various division based on the source taxonomy. Bacterial division alone consists of more than 100 files has size about 6 Gigabytes in total. Searching in 100 files using single server took time about 1500 seconds in a Quad Cores machine. An effort to speed up this process has been worked out by uploading the bacterial sequence data on Hadoop Distributed File System on low-end cluster. MapReduce computation model is invoked for searching algorithm in conjunction with Hadoop Distributed File System as both technologies are main component of Hadoop framework. Scalability evaluation has been performed to investigate whether increasing number of node in the cluster will be fruitful.**

Keywords—DNA; Hadoop; Distributed File System; MapReduce

I. INTRODUCTION

The advent of next-generation DNA (Deoxyribon Nucleic Acid) sequencing technology has created a very huge and still increasing of sequence data that has to be stored, organized and delivered to biomolecular scientist for further research [1]. The European Bioinformatics Institute (EBI), which organizes a central repository of sequence data called EMBL-bank, has increased storage capacity from 2.5 Petabytes to 5 Petabytes in 2009 [2]. Publicly available DNA sequence database called genbank is managed by National Center for Biotechnology Information (NCBI), at the National Institutes of Health (NIH, USA), which receives data through the international collaboration with DNA Databank of Japan (DDBJ) and European Molecular Biology Laboratory (EMBL) as well as from the scientific community [3]. These three partners in the collaboration called The International Nucleotide Sequence Database Collaboration (INSDC) exchange data daily to ensure that comprehensive data of sequence information is available

worldwide, for example NCBI make the Genbank data available for free through FTP (ftp://ftp.ncbi.nih.gov/genbank) [4].

The genbank database release 203 in August 2014 contains about 652 GByte sequence data in uncompressed traditional flat file as well as in ASN.1 format. The size of this database is expected to grow exponentially as from 1982 to the present; the number of bases in genbank has doubled approximately every 18 months [5] as shown in Fig. 1. The volume of data with this size presents a great challenge in storage and communication of data. Currently, sequence data in genbank is kept in large flat files which are compressed using standard gzip compression. Many researches have been done on compressing DNA sequence individually or the entire database as worked in [6] to circumvent the storage problem. While increasing the compression ratio of sequence individual data or sequence databases diminishes the storage problem, but the data communication problem is not directly taken care.

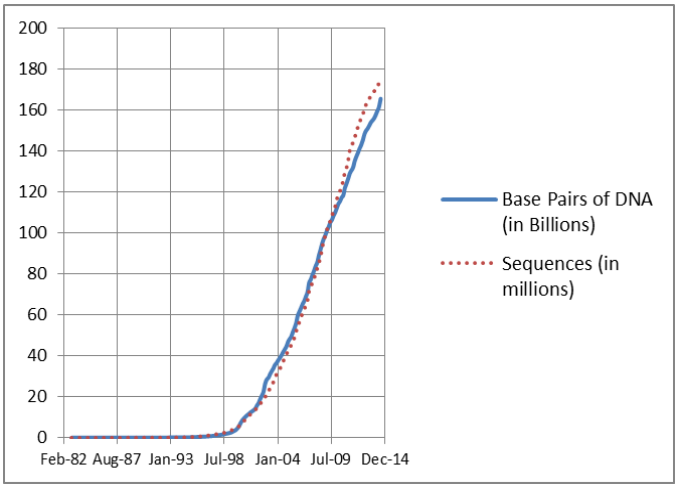
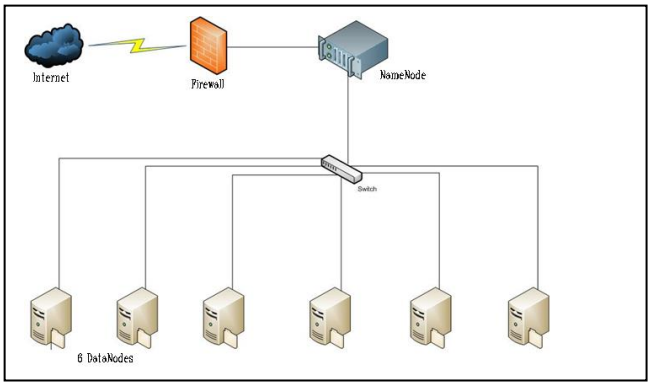


Fig. 1. The growth of genbank database from Dec 1982 to Aug 2014 (ftp://ftp.ncbi.nih.gov/genbank/release.notes/gb203.release.notes)

In traditional high performance computing (HPC) applications, it is a common practice to have "high-end processing nodes" with a large amount of shared memory and "storages nodes" attached together by a high-capacity interconnection device. This scaling "up" approach is not cost effective, since the cost of such machine does not scale linearly. Tackling large data problems asks an alternative approach than the traditional model of computing. Instead of scaling "up" approach, scaling "out" approach; i.e. utilizing a large number of commodity low-end servers, is preferred for data-intensive workload [2].

The separation of computing node and storage node creates bottleneck in the network. As an alternative to moving around the data, it is more efficient to move processing around to the data; hence the processors and the storage are co-located. In this type of computing model, one can take the benefit of data locality by running code on the computing processor directly attached to the block of they need. The architecture of computer cluster which complies to this model is called distributed file system. Over this type of cluster a so called MapReduce programming model is usually invoked as proposed by Dean and Ghemawat from Google in [7]. They claimed that Google's implementation of MapReduce and Google File System (GFS) [8] which run on a large cluster of commodity machines is highly scalable.



Doug Cutting who founded a Nutch project [9]; i.e. full-featured text indexing and searching library, faced with the scalability issues in his project. Inspired by Google's work in [7, 8], he implemented the new framework and ported Nutch to it. Using the new framework, named Hadoop, Nutch became more scalable than any web crawler engine at that time [10]. In 2006, Doug Cutting is hired by Yahoo! to work with a dedicated team on improving Hadoop as an open source Apache project [11, 12].

II. HADOOP FRAMEWORK

Now a day, Hadoop is a collection of many related subprojects of infrastructure for distributed computing. It consists of two main components, namely Hadoop Distributed File System (HDFS) and MapReduce distributed computation programming framework. Using these two main cores, one can decompose a very large data set and let computations run in parallel close to data across many commodity servers. Other subprojects under Hadoop, such as Pig (dataflow language and parallel execution framework), Hbase (column oriented table service), and Zookeper (distributed coordination service) and Hive (data warehouse infrastructure), provide complementary services build on the core to add higher level abstraction [11, 13].

A. Hadoop Distributed File System

HDFS is a file system that is designed for run a MapReduce application on a cluster of commodity computers. A big data set will be divided into smaller (say 64Mbyte) blocks/chunks that are spread among computer nodes in the cluster via HDFS. These chunks of data input will be read in parallel which provide a much higher throughput. For data-intensive processing, the number of chunks will be too large to be moved around between nodes in the cluster. Instead of moving the data, Hadoop lets program codes move around. This move-code-to-data concept is more efficient with respect to communication load because the program codes are orders of magnitude smaller than the data chunk.

HDFS stores file system metadata and application data separately in different servers. While file system metadata is stored on single dedicated server, called **NameNode;** application data are distributed on a number of other servers called **DataNodes**. As depicted in Fig. 2.

The NameNode server is the master of HDFS that manages the slave DataNodes daemons to do the low-level I/O tasks.

The NameNode acts as a bookkeeper that keeps track how the application data are split into data chunks and where those chunks are stored. The DataNodes reside in the slave machines; provide the block storage and data retrieval services for the client application.

Fig. 2. HDFS Cluster consists of a NameNode and 6 DataNodes servers.

B. *MapReduce: Distributed Computation Model*

MapReduce is a distributed programming model which is inspired from functional programming model. MapReduce proceed large datasets normally in two stages, i.e. **map** and **reduce** stage. In the first stage, the map function is applied over all input records in the large datasets and can be performed in parallel since each functional application happens independently. The intermediate result of the first stage if required could be read in aggregation way by the reduce (folding) function. Application programmer just needs to define these two main functions and the MapReduce framework will execute the actual processing which decomposes the job into a set of map tasks, shuffle-sort and a set of reduce tasks.

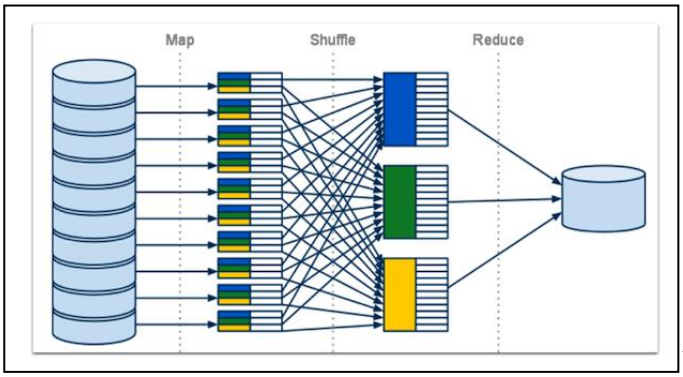


Fig. 3. Simplified MapReduce process

In contrary to the more common relational tables, MapReduce uses its basic data structure in form of key-value pairs (*k, v*). This form of data structure provides the flexibility to tackle semi structured or even unstructured data sets.

The mapper is applied to every input key-value pair (k_1, *v1*) spread over a number of files (or blocks) to produce a list of intermediate key-value pairs [(k_2, *v2*)].

$$\text{map: } (k_1,v_1) \rightarrow [(k_2,v_2)] \quad (1)$$

The reducer is applied to all values corresponding to the same intermediate key (k_2, [v_2]) to generate a list of output key-values [(k_3,v_3)].

$$\text{reduce: } (k_2, [v_2]) \rightarrow [(k_3,v_3)] \quad (2)$$

As in HDFS, the master-slave architecture is invoked to control the job execution processes in the MapReduce. As slaves, **TaskTrackers** run the actual computation jobs of MapReduce and send progress report to a master, called **JobTracker.** JobTracker coordinates all the jobs run on the system by scheduling the task to run on the TaskTrackers and monitoring the entire task as they are running. If a task fails, the JobTracker can relaunch the task, possibly on a different slave node.

Normal configuration of Hadoop sets TaskTrackers and DataNodes in the same machine of slave nodes, and JobTracker master normally share the same server with the NameNode as depicted in Fig.4.

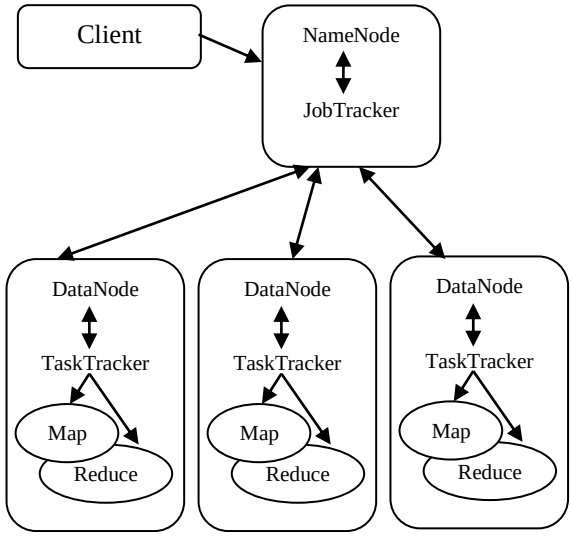


Fig. 4. Topology of typical Hadoop cluster

IV. DNA SEQUENCE GENBANK DATABASE

A. *GenBank File System*

NCBI provides GenBank database available for public via ftp in two formats, namely the GenBank Flat File format available at NCBI's anonymous FTP server `ftp://ftp.ncbi.nih.gov/genbank` and ASN.1 format available at `ftp://ftp.ncbi.nih.gov/ncbi-asn1`.

In GenBank database, sequence records are grouped into various divisions based either on the source taxonomy or the sequencing strategy on which the data is obtained. Some of taxonomic divisions are presented in Table 1. The number of sequence data files shown increases for each new release. Total all GenBank file from this release is 2093 files [5].

TABLE I. GENBANK TAXONOMIC DIVISION AND NUMBER OF FILE PER RELEASE 203 (AUG 2014)[5]

Division code	Description	Number of files
bct	Bacteria	142
inv	Invertebrate	40
mam	Other mammals	9
pln	Plant (inc. Fungi and algae)	86

pri	Primate	48
phg	Phage	2
rod	Rodent	31
vrl	Viruses	32
vrt	Other vertebrate	33

B. GenBank Flat File Format

All GenBank flat file has the same format and consists of two main parts; i.e. header information ant sequence entries. Header information includes the name file, release number, released datum, division description, and number of sequence entries as shown in Fig 5.

```
GBBCT1.SEQ          Genetic Sequence Data Bank
                          August 15 2014

                NCBI-GenBank Flat File Release 203.0

                     Bacterial Sequences (Part 1)

 51396 loci,    92682287 bases, from    51396 reported sequences
```

Fig. 5. Sample of header information of flat file gbbct1.seq from bacterial division

The second portion contains sequence entries where separator token "//" put between two successive entries. Within sequence entry, each line consists of two part, i.e. the first ten (10) columns in line may contain:

- Keyword; if it begins in column 1. Example: REFERENCE, LOCUS etc.
- Sub-keyword; if the first two columns in line are blank. Example AUTHORS as sub-keyword of REFERENCE.
- Blank character indicating that this line a continuation of information under keyword or sub-keyword above it.
- Number ending in column 9 of the line designates the numbering of the actual nucleotide sequence position.
- Two slashes (//) in column 1 and 2 indicating the end of entry.

The second part in position 13 to 80 contains the information associated to its keyword or sub-keyword.

```
GBSMP.SEQ          Genetic Sequence Data Bank
                         October 15 1992

                GenBank Flat File Release 74.0

                    Structural RNA Sequences

      2 loci,       236 bases, from     2 reported sequences

LOCUS       AAURRA        118 bp ss-rRNA            RNA       16-
JUN-1986
DEFINITION  A.auricula-judae (mushroom) 5S ribosomal RNA.
ACCESSION   K03160
VERSION     K03160.1  GI:173593
KEYWORDS    5S ribosomal RNA; ribosomal RNA.
SOURCE      A.auricula-judae (mushroom) ribosomal RNA.
  ORGANISM  Auricularia auricula-judae
            Eukaryota; Fungi; Eumycota; Basidiomycotina;
            Heterobasidiomycetidae; Auriculariales;
Auriculariaceae.
REFERENCE   1  (bases 1 to 118)
  AUTHORS   Huysmans,E., Dams,E., Vandenberghe,A. and De Wachter,R.
... deleted ....
BASE COUNT       27 a     34 c     34 g     23 t
ORIGIN      5' end of mature rRNA.
        1 atccacggcc ataggactct gaaagcactg ca...deleted...
       61 gtaccgccca gttagtacca cggtgggga cc...deleted...
//
LOCUS       ABCRRAA       118 bp ss-rRNA            RNA
15-SEP-1990
DEFINITION  Acetobacter sp. (strain MB 58) 5S ribosomal
ACCESSION   M34766
VERSION     M34766.1  GI:173603
KEYWORDS    5S ribosomal RNA.
SOURCE      Acetobacter sp. (strain MB 58) rRNA.
  ORGANISM  Acetobacter sp.
            Prokaryotae; Graci
... deleted ....
```

Fig. 6. Sample of flat file shows header and the first two entries where some lines deleted for clarity.

V. GENBANK DATABASE ON LOW-END HADOOP CLUSTER

DNA sequence GenBank data used in this work is obtained via NCBI FTP server. In the previous work we used a single server to manage DNA sequence for each sequence division. The bacterial division (bct) was used since it consists the largest file numbers and grows quite fast. When we do sequence alignment searching, the process of searching in single database needs process time too long. Using a Quad Core server at 2.13GHz, it took about 1500 seconds for searching in 100 bacterial division flat files from GenBank release 195 in April 2013 (current release 203 in August 2014 they are 149 files). In the present work, the same data are uploaded on HDFS cluster consists of one master powered by Intel Quad Core X3210 at 2.13 GHz and 12 node slaves powered by Intel Dual Core E2160 @1.8 GHz. This cluster is built up of low-end commodity computers typically used for teaching laboratory. For this research purpose the memory is upgraded from minimum 1 GByte to 4GByte for all node slaves.

A. Mapper Process

The present work will focus in the searching phase in the flat files. Hence, only the mapper function of MapReduce model is needed where filtering is applied against the search-keyword.

Initially, the flat files are read by InputReader to form RecordReader which is input pairs for Mapper(*k,v*) where key k is entry number and value v is a sequence entry. In the InputReader a complex RecordReader is built up containing another key/value pairs where k is either keyword or sub keyword within sequence entry and value is the real information appropriate to the key.

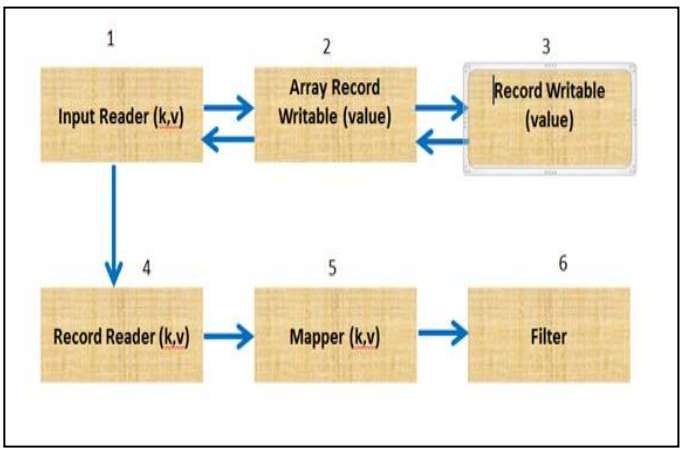


Fig. 7. Mapper process model

In the Mapper, filtering process is performed by comparing the search keyword with the value from information in each sequence entry and resulting a list of all sequence entries containing the search keyword.

B. *Scalability Evaluation*

Two superior characteristics of Hadoop framework are scalability and availability performance. The present work will study more on the scalability of Hadoop using a low-end cluster. For data-intensive processing such as DNA sequence searching scalable systems are highly desirable.

The definition of scalability can be considered from two points of views. First, in terms of data, i.e. given twice the amount of data, the same application should take at most twice as long to run. Second in terms of resources: given a cluster with twice the size, the same application should take no more than half as long to run.

The study of scalability in term of data was done first using a cluster with 6 slave nodes where the input data of GenBank flat file varied from 1 GByte to 6 Gbyte. The 6 Gbyte flat files are the total size all 100 files of bacterial division from release 195 downloaded from GenBank NCBI in 2013 in the gzip compressed format. Fig. 8 shows the curve almost linear, i.e. the growth of required times in same order as the growth of data size. In other word the low-end cluster indicates very good data scalability up to the normal data size. Noted that the input data used are in compressed format as they were downloaded from NCBI.

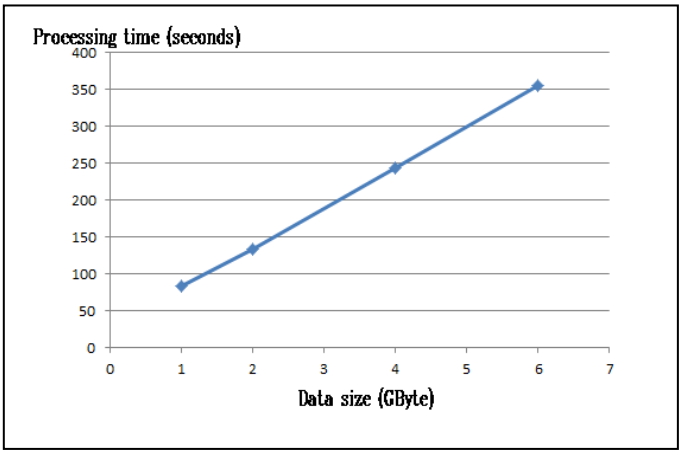


Fig. 8. Scalabilty in terms of data

The scalability in terms of resources is tested by varying number of slave nodes in the cluster and internal memory for every node. The input data used are all 100 flat files of bacterial sequence division from release 195 GenBank in compressed format

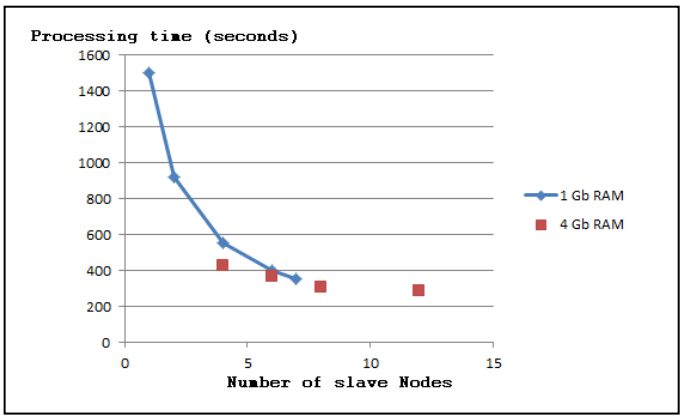


Fig. 9. Test of Scalability in terms of resources. Processing time decreases not in the same order of increasing resources (number of nodes)

TABLE II. PROCESSING TIME WITH HADOOP CLUSTER OF DIFFERENT NUMBER OF NODE (ALL 4GB RAM)

No. Nodes	Process. Time (seconds)	No. Nodes	Process. Time (seconds)
4	425.8	6	367.2
8 (2x)	305.2 (0.72x)	12 (2x)	284.4 (0.77x)

Fig. 9 shows that the scalability in terms of resources is far from desired order except in the range of small cluster size. Table II gives more detail information for scalability testing in cluster with 4 GB RAM. A cluster with twice the size, the same application requires more than half time (0.72x and 0.77x) to run. Increasing internal memories from 1 Gbyte to 4 Gbyte does reduces the processing time but again it is still not satisfactory.

We run again the same input flat files data but in uncompressed format hence the total size of data is about 22 Gbyte or 3.5 times bigger. In uncompressed format, input files are divided in blocks by HDSF according limit size of each block. Different block size is studied and the results are given in Fig. 9.

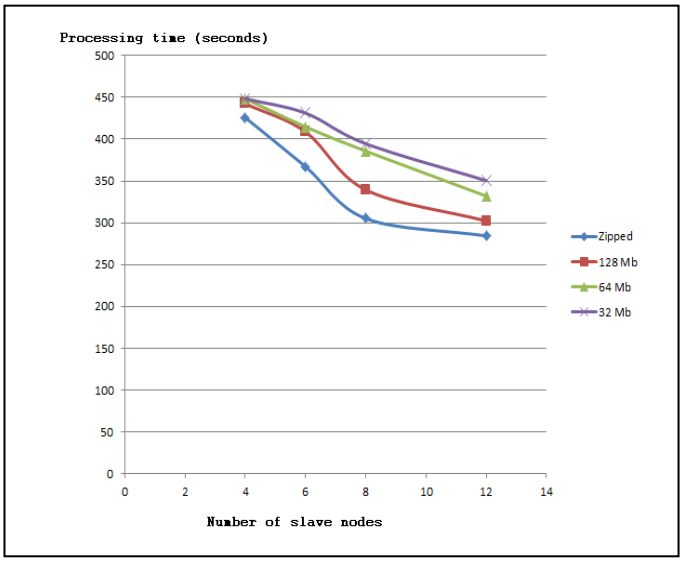


Fig. 10. Processing time for compressed and uncompressed flat file decreases not in the same order of increasing resources (number of nodes)

Processing time depends on the block size. Smaller block size means number of chunks bigger and more time needed for Hadoop framework to finish the jobs. Increasing number of chunks will get an advantage if number of node available large enough to proceed more chunks in parallel. Looking at the curves in Fig. 10 it is possibly that run time of smaller block size could be faster when number of nodes is more than 20.

VI. CONCLUDNG REMARKS

A scalability study has been performed for sequence searching of DNA database on the low-end Hadoop cluster. The data scalability is good in the meaning that the growth of required times in same order as the growth of data size. Two different internal memory sizes has been compared that quadrupling the RAM from 1 Gbyte to 4 Gbyte gains speed up but not significant enough.

Increasing cluster size by doubling the number of node gained speed up about 1.3; this is much lower than the ideal speed up of 2. Nevertheless, that the processing time decreases steadily with the growth of the cluster size give a good promising gain in a larger cluster.

ACKNOWLEDGMENT

The authors would like to thank Direktorat Jendral Pendidikan Tinggi, Kemendikbud, for funding the presented work through research grant under "Hibah Bersaing" program.

REFERENCES

[1] E.R. mardis, "The impact of next-generation sequenceing technology on genetics", in *Trends in Genetics* Vol. 24 No.3, Elsevier Ltd, 2008, pp. 133-141.

[2] J. Lin, C. Dyer, "Data-Intensive Text processing with MapReduce", in Synthesis Lectures on Human Language Technlogy, vol. 3, no. 1, 2010, pp.1-177.

[3] E.W.Sayers, T. Barrett, ,"Database Resources of The National Center for Biotechnology Information", Nucleic Acids Research., January: Vol.37, 2009, Database issue, pp. D5-D15.

[4] D.A. Benson, I. Karsch_Mizrachi, K. Clarck, D.J. Lipman, J. Ostel, E.W. Sayers, ,"GenBank", Nucleic Acids Research, Vol. 40, Database issue, 2012, pp. D48-D53.

[5] NCBI: NCBI-GenBank Flat File Release 203 Release Notes. ftp://ftp.ncbi.nih.gov/genbank/release.notes/gb203.release.notes

[6] W.T.J. White, M.D. Hendy, "Compressing DNA Sequence databases with coil", BMC Bioinformatics, 2008 vol. 9:242.

[7] J. Dean, S. Ghemawat, "MapReduce:Simplified data Processingon Large Cluster", in Proceeding of the 6th Symposium on Operating System Design and Implementation (OSDI 2004), pp.137-150, California, 2004.

[8] S. Ghemawat, H. Gobioff, S.T. Leung, "The Google File System", in Proceeding of the 19th ACM Symposium on Operating Systems Principles (SOSP 2003), New York, pp. 29-43, 2003.

[9] M. Cafarella, D. Cutting, "Building Nutch: Open Source Search", ACM Queue, April 2004, vol.2 (2), pp. 54-61, 2004.

[10] R. Khare, D. Cutting, K. Sitaker, A. Rifkin, "Nutch: A Flexible and Scalable Open-Source Web Search Engine", Technical Report, CN-TR-04-04, CommerceNet Labs, 2004.

[11] K. Shvachko, H. Kuang, S. Radia, R. Chansler, "The Hadoop Distributed File System", in Proceeding of the 26th IEEE Symposium on Massive Storage Systems and Technologies (MSST 2010), May, 2010.

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

[13] T. White, Hadoop: The Defintive Guide, 3rd Edition, USA, O'Reilly, 2012.