Jurnal Informatika Terpadu

https://journal.nurulfikri.ac.id/index.php/JIT
ISSN ONLINE : 2460-8998

ANALISIS DAN EVALUASI ALGORITMA *MAPREDUCE WORDCOUNT* PADA CLUSTER *HADOOP* MENGGUNAKAN INDIKATOR KECEPATAN

Robiyatul Adawiyah[1], Sirojul Munir[2]

[1, 2]Teknik Informatika, Sekolah Tinggi Teknologi Terpadu Nurul Fikri
Jakarta Selatan, DKI Jakarta, Indonesia 12640
arobiyatul8@gmail.com, rojulman@gmail.com

Abstract

This research proposes to analyze the speed of the MapReduce algorithm on the Hadoop cluster and find out the time it takes to process GDELT data on Hadoop. This study uses qualitative analysis methods. The data analysis concludes that the Word Count algorithm applied to the GDELT data set can run on the Hadoop cluster. The speed of the WordCount algorithm on MapReduce, which applies to the GDELT data set on Hadoop, affects if the nodes using add, which in the study used as many as two physical machine nodes. Hadoop can process data that is large and a lot because Hadoop processes data in a distributed manner. Hadoop speed can adjust by adding nodes as well as other settings such as block size.

Keywords: *Hadoop, MapReduce, WordCount*

Abstrak

Penelitian diajukan untuk menganalisis kecepatan algoritma *MapReduce* pada *cluster Hadoop* dan mengetahui waktu yang dibutuhkan dalam mengolah data GDELT pada *Hadoop*. Penelitian ini menggunakan metode analisis kualitatif. Berdasarkan analisa data yang telah dilakukan, diperoleh kesimpulan bahwa algoritma *WordCount* yang diterapkan pada data set GDELT dapat berjalan pada *cluster Hadoop*. Kecepatan algoritma *WordCount* pada *MapReduce* yang diterapkan untuk data set GDELT pada *hadoop* berpengaruh apabila node yang digunakan ditambah, dimana dalam penelitian menggunakan sebanyak 2 *node physical machine*. *Hadoop* dapat mengolah data yang memiliki ukuran besardan banyak karena *Hadoop* mengolah data secara terdistribusi. Kecepatan *Hadoop* dapat diatur dengan menambahkan *node* dan juga pengaturan lainnya seperti halnya *block size*.

Kata kunci: *Hadoop, MapReduce, WordCount*

1. PENDAHULUAN

Saat ini, teknologi informasi berkembang pesat dengan ditandai melimpahnya data yang tersedia, mula dari data di media sosial, *E-Commerce*, kependudukan, industri, dan masih banyak lagi. Data yang semakin besar ukurannya tersebut dan sudah sangat sulit untuk dikoleksi, disimpan, dikelola maupun dianalisa dengan menggunakan sistem *database* biasa dikarenakan ukurannya yang terus bertambah disebut dengan *big data* [2]. *International Data Corporation* memperkirakan ukuran data semesta digital berada pada angka 0.18 zettabytes (1 zettabytes = 10^{12} bytes), serta meramalkan akan menjadi 10 kali lipatnya setiap 5 tahun [2].

Salah satu data dengan kapasitas dan ukuran besar yaitu data dari GDELT (*Global Dataset of Event, Language and Tone*) *Project* dimana data tersebut menampung data dari seluruh dunia. Proyek GDELT adalah yang terbesar, paling komprehensif, dan database terbuka dengan resolusi tinggi yang pernah dibuat. Pada data tahun 2015 saja mencatat hampir tiga perempat dari satu triliun snapshot emosional dan lebih dari 1,5 miliar referensi lokasi, sementara total arsipnya mencapai lebih dari 215 tahun, menjadikannya salah satu *dataset open-access spatio-temporal* terbesar yang ada dan mendorong batas-batas studi "*big data*" dari masyarakat manusia global. *Global Knowledge Graph*-nya menghubungkan orang-orang, organisasi, lokasi, tema, jumlah, gambar, dan emosi dunia ke dalam satu jaringan tunggal di seluruh dunia [5]. Adanya suatu sistem dalam pengolahan data memungkinkan pengguna untuk mempermudah dalam mengolah suatu data, ada beberapa sistem atau *framework* yang menggunakan teknik dan algoritma dalam mengolah suatu *big data* seperti *clustering* dan klasifikasi dari suatu *dataset*. *Clustering* adalah salah satu teknik yang saat ini banyak digunakan dalam mengolah

Diterima 05 Januari 2020, Direvisi 26 Februari 2020, Diterima untuk publikasi 06 Maret 2020

suatu *dataset*. Tujuan *clustering* itu sendiri adalah untuk mengumpulkan data dan mengelompokannya menjadi sekumpulan data yang sangat penting dan mempunyai nilai ke dalam sebuah kluster untuk selanjutnya dapat digunakan.

Dalam menggunakan sistem pengolahan data ada beberapa teknik dan algoritma *clustering* yang dapat digunakan, namun penggunaan algoritma untuk mengolah dan memproses data yang terlalu banyak membutuhkan waktu yang sangat lama dalam proses komputasinya jika hanya menggunakan algoritma yang sekuensial. *Hadoop* merupakan salah satu *framework* yang sering digunakan dalam hal pengolahan data dengan jumlah yang besar dimana *Hadoop* memiliki sistem khusus untuk mendukung kinerja komputasi yang dirancang khusus untuk menyimpan dan menganalisis data dalam jumlah yang sangat besar dan tidak terstruktur.

1.1 Rumusan Masalah

Rumusan masalah dalam penelitian ini adalah "Bagaimana algoritma *WordCount* diterapkan pada *dataset Global Dataset of Events, Language and Tone* (GDELT)" dengan pertanyaan penelitan ini adalah:

1. Apakah algoritma *WordCount* yang diterapkan pada *dataset* GDELT dapat berjalan pada cluster *Hadoop*?
2. Apakah kecepatan algoritma *WordCount* pada *dataset* GDELT dengan bertambahnya *node* akan lebih cepat?

1.2 Tujuan dan Manfaat Penelitian

1. Menganalisis kecepatan algoritma *MapReduce* pada *cluster Hadoop*.
2. Mengetahui waktu yang dibutuhkan dalam mengolah data GDELT pada *Hadoop*.

1.3 Batasan Masalah

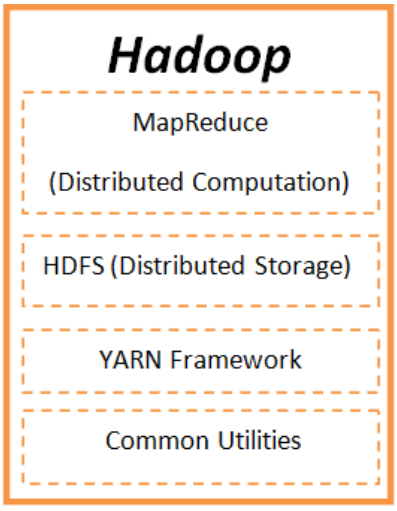
Batasan masalah dalam penelitian ini adalah:

1. Menggunakan sistem operasi Ubuntu.
2. Menggunakan *Hadoop*.
3. Menggunakan dua atau tiga *node.*

2. METODE PENELITIAN

2.1 *Hadoop*

Hadoop adalah *framework* Apache berjenis *open source* yang ditulis dalam Java yang memungkinkan pemrosesan terdistribusi dari kumpulan data besar kelompok komputer menggunakan model pemrograman sederhana. *Hadoop* bekerja di lingkungan yang menyediakan penyimpanan dan komputasi terdistribusi di seluruh kluster komputer. *Hadoop* dirancang untuk meningkatkan dari satu *server* hingga ribuan mesin, masing-masing menawarkan komputasi dan penyimpanan lokal [3].



Gambar 1. Arsitektur *Hadoop*

Hadoop Framework mencakup empat modul berikut :

1. *Hadoop Common*
   Hadoop Common adalah pustaka dan utilitas Java yang diperlukan oleh modul *Hadoop* lain. Pustaka-pustaka ini menyediakan file sistem dan abstraksi tingkat OS dan berisi file Java yang diperlukan dan skrip yang diperlukan untuk memulai Hadoop.

2. *Hadoop YARN*
   YARN adalah kerangka kerja untuk penjadwalan pekerjaan dan pengelolaan sumber daya klaster.

3. HDFS
   Sistem file terdistribusi yang menyediakan akses *throughput* tinggi ke data aplikasi.

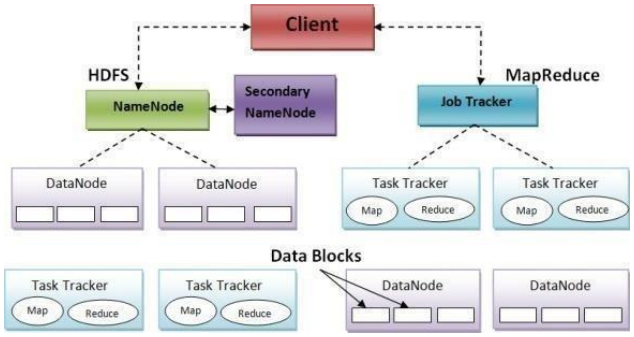
4. *MapReduce*
   MapReduce adalah sistem berbasis-*YARN* untuk pemrosesan paralel dari kumpulan data besar.

2.2 HDFS

Hadoop Distributed File System (HDFS) didasarkan pada *Google File System* (GFS) dan menyediakan sistem file terdistribusi yang dirancang untuk dijalankan pada kelompok besar mesin komputer kecil dengan cara yang dapat diandalkan dan toleran. HDFS menggunakan metode ini ketika mereplikasi data untuk redundansi data di beberapa rak. Pendekatan ini mengurangi dampak pemadaman listrik rak atau kegagalan sakelar; jika salah satu kegagalan perangkat keras ini terjadi, data akan tetap tersedia [3].

HDFS menggunakan arsitektur *master/slave* dimana *master* terdiri dari *NameNode* tunggal yang mengelola metadata sistem file dan satu atau lebih *slave DataNodes* yang menyimpan data aktual. Sebuah file dalam namespace HDFS dibagi menjadi beberapa blok dan blok tersebut disimpan dalam satu set *DataNodes*. *NameNode* menentukan pemetaan blok ke *DataNodes*. *DataNodes*

menangani operasi baca dan tulis dengan sistem file. Mereka juga mengurus pembuatan blok, penghapusan dan replikasi berdasarkan instruksi yang diberikan oleh *NameNode.*



Gambar 2. HDFS

Hadoop didukung oleh *platform* GNU/Linux. Kerangka *Hadoop* dapat dikonfigurasi dalam tiga mode berikut: *Mode Standalone, Pseudo-Distributed Mode*, dan Mode Terdistribusi Penuh. Secara *default*, *Hadoop* dikonfigurasi untuk berjalan dalam mode tidak terdistribusi, sebagai satu proses Java. Ini berguna untuk *debugging*. Ini juga dapat dijalankan pada *node* tunggal dalam mode *pseudo-distributed* dimana setiap *daemon* berjalan dalam proses Java yang terpisah. Mode *pseudo-distributed* juga disebut sebagai *nodenode* tunggal.

2.3 MapReduce

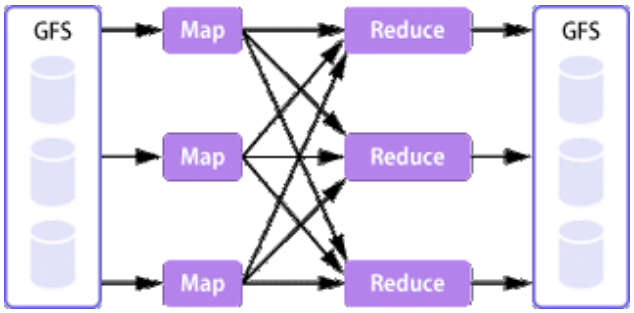
MapReduce sebuah *framework* untuk aplikasi dan *programming* yang dikenalkan oleh *Google* dan digunakan untuk melakukan suatu pekerjaan dari komputasi terdistribusi yang dijalankan pada sebuah *cluster* [3]. *MapReduce* ini terdiri dari konsep fungsi *map* dan *reduce* yang biasa digunakan pada *functional programming* [4].

Salah satu program yang menggunakan konsep *MapReduce* yang telah disediakan oleh *Hadoop* adalah *WordCount*. *WordCount* merupakan program yang bertujuan untuk menghitung kata pada *file plaintext*. Proses *MapReduce* pada *WordCount* ini dibagi menjadi 2 tahap yaitu proses *mapping* dan *reducing.*

2.4 Tahapan Proses *MapReduce*

Hadoop menyediakan dua jenis *slot* untuk melakukan *MapReduce* yaitu *slot map* dan *slot reduce*. Secara *default Hadoop* telah menentukan jumlah *slot map* dan *slot reduce* untuk setiap *node*yaitu dua *slot map* dan satu *slot reduce*. Pada saat memproses data, *Hadoop* terlebih dahulu melakukan proses *mapping* pada *task* yang terdapat pada *slot map* sampai selesai kemudian dilanjutkan dengan proses *reduce* pada *slot reduce.*

MapReduce terdiri atas tiga tahap, yaitu tahap *map*, tahap *shuffle*, dan terakhir tahap *reduce*. Untuk tahapan *shuffle* dan *reduce* digabungkan ke dalam satu tahap besaran-nya yaitu tahap *reduce*.



Gambar 3. *MapReduce*

1. Tahap *map*, memproses data inputan yang umumnya berupa file yang tersimpan dalam HDFS, inputan tersebut kemudian diubah menjadi *tuple* yaitu pasangan antara *key* dan *value*-nya.
2. Tahap *reduce*, memproses data inputan dari hasil proses *map*, yang kemudian dilakukan tahap *shuffle* dan *reduce* yang hasil *dataset* barunya disimpan di HDFS kembali.

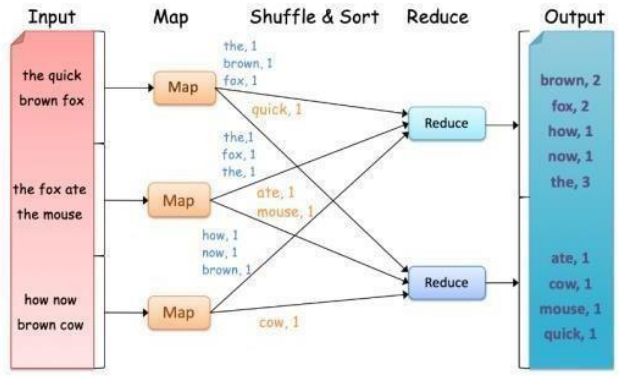
3. PERANCANGAN SISTEM

3.1 Algoritma *Wordcount*

Wordcount adalah algoritma yang sederhana dan mudah dipahami yang dapat dengan mudah diimplementasikan sebagai aplikasi *mapreduce*. Diberikan seperangkat dokumen teks, program menghitung jumlah kemunculan setiap kata. Algoritma ini terdiri dari tiga bagian utama [7].

1. Main Program
2. *Mapper*
3. *Reducer*

Berikut contoh dari *Wordcount*:



Gambar 4. Contoh *Wordcount*

Fungsi *map* mengeluarkan setiap kata ditambah dengan jumlah perhitungan yang terkait. Dokumen yang dimasukkan *tokenized*, dimana *key* adalah nama dokumen dan *value* adalah isi dokumen.

Reducer: Fungsi *reduce* menjumlahkan semua jumlah yang dikeluarkan untuk kata tertentu.

Mapper Classs: Wordcount Mapper Class dibuat dengan meng-*extend Mapper Class* dan fungsi map diimplementasikan dengan meng-*override method* pada *Mapper Class*. Fungsi *mapper* mengambil *key-value* sebagai *input* dan *output key-value* sebagai *output*.

Reducer Class: Wordcount Reducer Class dibuat dengan meng-*extend org.apache.hadoop.mapreduce*. *Reducer class* dan *reduce method* diimplementasikan dengan meng-*override reduce method* dari *Reducer Class*. Fungsi *reduce* mengumpulkan semua *key-value* (word,1) digenerasi dengan beberapa fungsi *map*.

Kode Akhir:

```
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "word count");
        job.setJarByClass(WordCount.class);
        job.setMapperClass(WordCountMapper.class);
        job.setCombinerClass(WordCountReducer.class);
        job.setReducerClass(WordCountReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
```

```
public static class WordCountReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values, Context context)
            throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        result.set(sum);
        context.write(key, result);
    }
}
```

```
    public static class WordCountMapper extends Mapper<Object, Text, Text, IntWritable> {

        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

        public void map(Object key, Text value, Context context) throws IOException,
                InterruptedException {
            StringTokenizer itr = new StringTokenizer(value.toString());
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
            }
        }
    }

    public static class WordCountReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
        private IntWritable result = new IntWritable();

        public void reduce(Text key, Iterable<IntWritable> values, Context context)
                throws IOException, InterruptedException {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            result.set(sum);
            context.write(key, result);
        }
    }
}
```

Gambar 5. Kode Akhir

3.2 Kebutuhan *Hardware*

Tabel 1. Spesifikasi Komputer

Hardware	PC 1	PC 2	PC 3
CPU	Intel *Core* i7-7700	Intel *Core* i7-7700	Intel *Core* i7-7700
RAM	8.00 GB	8.00 GB	8.00 GB
SistemOperasi	Ubuntu18.04	Ubuntu 18.04	Ubuntu18.04

4. IMPLEMENTASI DAN PENGUJIAN

4.1 Instalasi dan Konfigurasi

Instalasi dan konfigurasi konfigurasi *Hadoop* terdapat dalam halaman lampiran.

4.2 Menjalankan *WordCounter*

1. Membuat direktori didalam file system dengan perintah:
   $ hadoop fs -mkdir -p /tes1/hduser/input

2. Masukkan data yang akan diolah dengan perintah:
   $ hadoop fs -put data3.csv /tes1/hduser/input

3. Mengecek data dengan perintah:
   $ hadoop fs -ls /tes1/hduser/input/

   Apabila data telah tersedia di dalam *file system* akan muncul keterangan seperti berikut:

   Found 1 items -rw-r--r-- 1 hduser supergroup **1001914352 2019-08-19 13:44** /tes1/hduser/input/data3.csv

4. Menjalankan program *WordCount*
   $ cd /usr/local/hadoop

   $ hadoop jar wordcount.jar wordcount /tes1/hduser/input /tes1/hduser/output

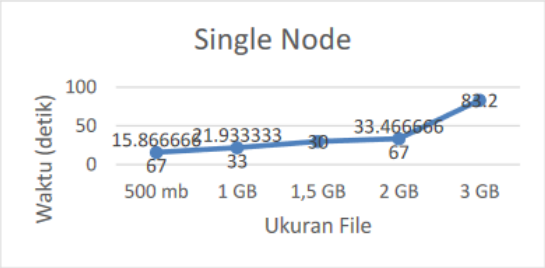
4.3 Pengujian Skenario Pertama

Skenario pertama bertujuan untuk mengetahui pengaruh *physical machine* sebagai *node* terhadap kecepatan *MapReduce* pada *Hadoop*. Pada skenario pertama, ukuran file yang digunakan pada saat *singlenode* yaitu 500 MB, 1 GB, 1.5 GB, 2 GB. Percobaan pada skenario pertama dilakukan sebanyak 15 kali percobaan.

1. Hasil Pengujian
   Hasil kecepatan rata-rata *MapReduce* pada skenario pertama yang dijalankan secara *single node* menggunakan *physical machine*.

Tabel 2. Pengujian Pertama

Ukuran File	Rata-rata Waktu (detik)
500 mb	15,8667
1 GB	21,9333
1,5 GB	30
2 GB	33,4667
3 GB	83,2



Gambar 6. Grafik Pengujian Pertama

2. Analisis
Berdasarkan hasil pengujian yang dilakukan pada skenario pertama dapat disimpulkan bahwa semakin besar ukuran file yang diuji maka kecepatan *MapReduce* akan semakin menurun.

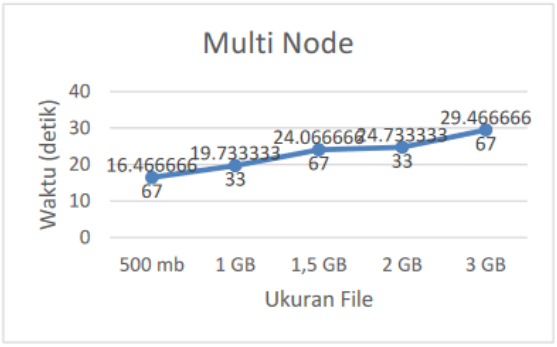
4.4 Pengujian Skenario Kedua

Skenario Kedua bertujuan untuk mengetahui pengaruh *physical machine* sebagai *node* terhadap kecepatan *MapReduce* pada *Hadoop*. Pada skenario kedua, ukuran *file* yang digunakan padasaat *multinode*yaitu 500 MB, 1 GB, 1.5 GB, 2 GB dan 3 GB. Percobaan pada skenario pertama dilakukan sebanyak 15 kali percobaan.

1. Hasil Pengujian
Hasil kecepatan rata-rata *MapReduce* pada skenario kedua yang dijalankan secara *multimode* menggunakan *physical machine.*

Tabel 3. Pengujian Kedua

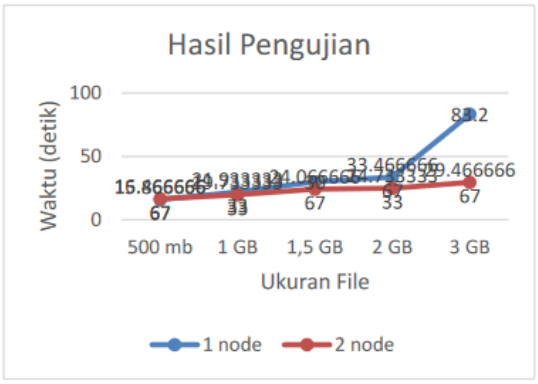
Ukuran File	Rata-rata Waktu (detik)
500 mb	16,4667
1 GB	19,7333
1,5 GB	24,0667
2 GB	24,7333
3 GB	29,4667



Gambar 7. Grafik Pengujian Kedua

2. Analisis
Berdasarkan hasil pengujian yang dilakukan pada skenario kedua dapat disimpulkan bahwa jumlah *physical machine* yang digunakan sebagai *node* dapat mempengaruhi kecepatan proses *MapReduce* pada *Hadoop*. Penambahan jumlah *physical machine* dengan spesifikasi yang sama sesuai perancangan dapat mempercepat kecepatan rata-rata *MapReduce* pada setiap file yang diuji. Hal ini terjadi karena setiap *node* akan mengambil *resource* dari komputer yang berbeda dimana setiap *node* akan mengambil *resource* dari masing-masing komputer, sehingga dengan menambah jumlah *node* akan memperingan proses kerja dari *cluster* yang digunakan.

4.5 Grafik Hasil Pengujian



Gambar 8. Grafik Hasil Pengujian

5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

a. *Algoritma Word Count* yang diterapkan pada data set GDELT dapat berjalan pada *cluster Hadoop*.
b. Kecepatan *Algoritma WordCount* pada *MapReduce* yang diterapkan untuk *dataset* GDELT pada *Hadoop* berpengaruh apabila *node* yang digunakan ditambah, dimana dalam penelitian menggunakan sebanyak 2 *node physical machine.*
c. *Hadoop* dapat mengolah data yang memiliki ukuran besar dan banyak karena *Hadoop* mengolah data

secara terdistribusi. Kecepatan *Hadoop* dapat diatur dengan menambahkan *node* dan juga pengaturan lainnya seperti halnya *blocksize*.

5.2 Saran

a. Disarankan untuk menganalisa kecepatan *MapReduce* dengan menggunakan algoritma yang berbeda dan lebih kompleks seperti algoritma *Agglomerative* atau algoritma *K-Means Clustering*.
b. Disarankan bagi penelitian selanjutnya untuk meneliti pengaruh *blocksize* pada kecepatan algoritma *MapReduce WordCount*.

DAFTAR PUSTAKA

[1] I. N. Aziz, Fitriyani, K. R. S. Wiharja, "Analisis Pengolahan *Text File*pada *Hadoop Cluster* dengan memperhatikan Kapasitas *Random*," Bandung: Universitas Telkom, 2015.

[2] C. Lam, "*Hadoop in Action*," Stamford: Manning Publications Co., 2011.

[3] M. Industri, "Definisi *Cloud Computing*," Meruvian.org Cloud Computing, 2013.

[4] "Apache *Hadoop*," Hadoop Apache, [Online]. Available: http://hadoop.apache.org/ [diakses 08 Juni 2018]

[5] K. Letaru, "*The GDELT Project*," GDELTProject, [Online]. Available: www.gdeltproject.org [diakses 08 Juni 2018]

[6] J. D. Ghenawat, "*MapReduce: Simplified Data Processing on Large Clusters*," 2004.

[7] M. G. Noll, "*Running Hadoop On Ubuntu Linux (Multi-Node Cluster)*," 2004-2019, [Online]. Available: https://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/ [diakses Juli 2019]