ANALISIS IMPLEMENTASI ALGORITMA *MAPREDUCE K-MEANS CLUSTERING* PADA HADOOP

ANALYSIS THE IMPLEMENTATION OF MAPREDUCE K-MEANS CLUSTERING ALGORITHM IN HADOOP

Aditya Alifinsyah[1], Fitriyani S.Si.,M.T.[2]
Prodi S1 Ilmu Komputasi, Fakultas Teknik, Universitas Telkom
aditya.alvinsyah@gmail.com, fitriyani@telkomuniversity.ac.id,

Abstrak

***Clustering* merupakan sebuah teknik yang banyak digunakan untuk pendistribusian dan pengolahan data. Tujuan dari klustering itu sendiri adalah untuk menemukan struktur dasar dari sebuah data dan mengelompokkannya menjadi sekumpulan data yang mempunyai nilai untuk dapat dipelajari dan dianalisis lebih lanjut. Sebuah teknik pengelompokan dan pendistribusian data yang banyak digunakan saat ini adalah *K-Means Clustering. K-Means Clustering* banyak digunakan karena kemudahan dalam pengaplikasiannya serta memberikan hasil klustering yang cukup baik.**

Ditengah era *Big Data* yang semakin berkembang seperti saat ini, penggunaan teknik dan analisis data yang masih bersifat tradisional ataupun serial mungkin tidak akan efisien lagi dalam pengolahan data yang jumlah dan ukurannya sangat besar. Maka dari itu penggunaan sebuah hardware ataupun system seperti Hadoop akan sangat membantu dalam proses klustering data yang sangat besar tersebut. Hadoop dapat digunakan secara efisien untuk pengolahan data dalam jumlah besar dikarenakan Hadoop memiliki sebuah algoritma pemrosesan data sendiri yang disebut *MapReduce*. *MapReduce* adalah sebuah algoritma yang dapat digunakan untuk mengatasi ukuran dan jumlah data yang besar dengan melakukan pendistribusian dan pengolahan data secara bersamaan.

Pada penelitian ini akan dianalisis bagaimana implementasi penggunaan *MapReduce* pada algoritma *K-Means Clustering* dengan menggunakan sebuah *Single Node* Hadoop yang akan dibandingkan dengan pemrosesan algoritma *K-Means Clustering* secara sekuensial dengan melihat waktu komputasinya.

Kata kunci: Hadoop , *MapReduce, K-Means, Mapreduce K-Means*

Abstract

Clustering is a technique much used for the distribution and data processing. The purpose of clustering itself is to find the fundamental structure of a data and categorized it to be a bunch of the data in that it has value for learning and analyzed further. A technique grouping and distribution of the data much used now is K-Means Clustering. K-meansCclustering mostly used because they ease in application and results from clustering is a good enough .

In the era of big data which keeps growing like today, the use of technic and analysis of data who is still in traditional or serial may not be efficient in data processing that is the sum of it measured the very large. Therefore the use of a hardware or system as Hadoop would be very helpful in the clustering process for the huge data. Hadoop can be used efficiently to processing data in large numbers because hadoop having an algorithm data processing own called MapReduce. Mapreduce is an algorithm that can be used to overcome size and the amount of data large by doing the distribution and processing data simultaneously .

In this research will be analyzed how the implementation of the use of algorithms MapReduce in K-Means Clustering by using a single node Hdoop who will be compared to processing algorithms K-Means Clustering in sequential by looking at computation or running process time.

Keyword: Hadoop , *MapReduce, K-Means, Mapreduce K-Means*

1. Pendahuluan

1.1. Latar Belakang Masalah

Data mining adalah teknologi gabungan untuk menganalisis suatu informasi menggunakan beberapa teknik tertentu seperti klasifikasi, klustering, dll, untuk mengumpulkan informasi berharga dari suatu data set. Klustering data adalah salah satu topik yang sangat banyak digunakan dan penting dalam data mining. Tujuan dari *clustering* itu sendiri adalah untuk menemukan data atau informasi yang berharga dari sekumpulan banyak data dan mengelompokannya menjadi sekumpulan data atau informasi yang mempunyai nilai ke dalam sebuah kluster untuk selanjutnya dapat digunakan sebagai bahan pembelajaran atau analisis.

Terdapat banyak teknik dan algoritma klustering yang dapat digunakan, seperti yang paling banyak digunakan adalah algoritma *K-Means Clustering*. Namun penggunaan algoritma *K-Means Clustering* untuk mengolah dan memproses data yang berukuran atau berjumlah sangat banyak dapat membutuhkan waktu yang sangat lama dalam proses komputasinya jika dengan hanya dengan menggunakan algoritma sekuensial *K-Means Clustering*.

Untuk dapat mengatasi permasalahan diatas maka pada penelitian ini akan dilakukan proses clustering data menggunakan Hadoop. Dimana Hadoop adalah sebuah sistem khusus untuk mendukung kinerja komputasi yang dirancang khusus untuk menyimpan dan menganalisa data dalam jumlah besar yang tidak terstruktur. Hadoop bekerja secara terdistribusi dengan 2 buah proses utama yaitu *MapReduce* dan *Hadoop Distributed File System* (HDFS). Hadoop memiliki kelebihan dapat secara cepat dan optimal dalam mengolah data yang sangat besar dengan kualitas *hardware* yang standar [5]. Maka dari itu penggunaan algoritma *MapReduce* yang ada pada Hadoop dapat menjadi sebuah solusi untuk memaksimalkan kinerja algoritma *K-Means Clustering*. Penggunaan algoritma *MapReduce K-Means Clustering* ini sendiri telah banyak dilakukan seperti pada penelitian yang dilakukan oleh Shweta Mishra dan Vivek Badhe dengan judul penelitian "Improved *MapReduce K-Means Clustering* Algorithm for Hadoop Architecture"[1].

1.2. Perumusan Masalah

Masalah yang akan dibahas pada tugas akhir ini meliputi

1. Bagaimana implementasi Algoritma *MapReduce K-Means Clustering* dengan menggunakan Hadoop ?
2. Bagaimana analisis performansi sekuensial *K-Means Clustering* dengan *MapReduce K-Means Clustering* ?

1.3. Tujuan

Beberapa tujuan yang ingin dicapai melalui tugas akhir ini sebagai berikut.

1. Menganalisis hasil dari implementasi algoritma *MapReduce K-Means Clustering* dengan menggunakan Hadoop.
2. Menganalisis performansi sekuensial *K-Means Clustering* dengan *MapReduce K-Means Clustering*.

1.4. Batasan Masalah

Adapun batasan masalah pada tugas akhir ini sebagai berikut :

1. Menggunakan *Single-Node* atau satu node sebagai *Name Node* dan *Data Node*
2. Instalasi dan konfigurasi Hadoop Cluster menggunakan *Hadoop Data Platform Sandbox*
3. Sistem Operasi berjalan diatas Windows 10

2. Landasan Teori

2.1. Analisis Clustering

Clustering adalah teknik data mining yang membagi data menjadi beberapa grup berdasarkan kemiripannya. Tujuan dari *clustering* adalah setiap objeknya yang mirip dapat menjadi satu grup dan yang berbeda menjadi grup yang lainnya. Analisis *cluster* adalah subjek yang luas dan karenanya banyak algoritma *clustering* tersedia untuk data kelompok set. Metode pengelompokan yang sangat umum melibatkan jarak komputasi, kepadatan dan interval atau distribusi statistik Bergantung pada persyaratan dan kumpulan data yang diterapkan pada algoritma *clustering* yang sesuai ekstrak data[2].

Partitional Clustering disebutkan sebagai pembagian obyek-obyek data ke dalam kelompok dimana tidak ada data pencilan sehingga setiap objek pasti masuk kedalam satu *cluster*. data dikelompokkan ke dalam sejumlah *cluster* tanpa adanya struktur hirarki antara satu dengan yang lainnya. Pada metode *partitional clustering* setiap *cluster* memiliki titik pusat *cluster* (centroid) dan secara umum metode ini memiliki fungsi tujuan yaitu meminimumkan jarak (*dissimilarity*) dari seluruh data ke pusat *cluster* masing-masing. Contoh metode *partitional clustering*: *K-Means*, Fuzzy *K-Means* dan *Mixture Modelling*. [6].

Terdapat beberapa keunggulan dari analisis *clustering* sehingga banyak digunakan, diantaranya [2]:

1. Meningkatkan efisiensi data mining dengan menggabungkan data dengan karakteristik yang relatif sama sehingga generalisasi dapat diturunkan untuk masing masing *cluster*.
2. *Clustering* memberikan gambaran yang berarti dan tercepat dari repositori data
3. Analisis *Clustering* memberikan pemahaman yang baik dari persamaan – persamaan yang tidak biasa pada sebuah data yang mungkin terjadi ketika proses *clustering* selesai.
4. *Clustering* dapat digunakan sebagai alat data mining sendiri untuk dapat digunakan sebagai pendistribusian data, atau sebagai langkah *pre-processing* untuk algoritma data mining yang lainnya.
5. *Clustering* dapat digunakan dalam banyak bidang dalam kehidupan seperti bidang biologi, *image pattern recognition, business intelligence,* kecerdasan buatan dan Web Search.

2.2. K-Means Clustering

Algoritma *K-Means* adalah algoritma *clustering* yang paling banyak digunakan[7]. Awalnya ditentukan K *cluster* yang akan dibentuk. Pilih secara acak objek yang akan ditetapkan sebagai centroid awal. Algoritma *K-Means* selanjutnya akan melakukan pengulangan langkah-langkah berikut sampai tidak ada centroid yang berubah[6]:

```
Input : Data, Jumlah Cluster (K)
Output : K Centroid
Pilih K point sebagai inisial centroid
Repeat
      Dari K yang telah terdefinisi masukan objek ke
                                        centroid terdekat.
      Hitung Ulang centroid dari setiap cluster
Until
      centroid tidak berubah
```

Gambar 2.1. Algoritma K-Means Clustering [6]

Algoritma *K-Means* secara *iterative* meningkatkan variasi nilai dalam dalam tiap tiap *cluster* dimana obyek selanjutnya ditempatkan dalam kelompok yang terdekat, dihitung dari titik tengah *cluster*. Titik tengah baru ditentukan bila semua data telah ditempatkan dalam *cluster* terdekat. Proses penentuan titik tengah dan penempatan data dalam *cluster* diulangi sampai nilai titik tengah dari semua *cluster* yang terbentuk tidak berubah lagi [9].

Untuk menentukan jarak tiap titik ke *centroid* terdekat dapat digunakan perhitungan jarak menggunakan *Euclidean distance*. Misalkan dalam sebuah ruang dua dimensi terdapat $X= (x_1, x_2, x_3, \ldots x_n)$ dan $Y= (y_1, y_2, y_3, \ldots y_n)$ maka untuk menghitung *Euclidean distance* dapat dilakukan sebagai berikut :

$$d(X,Y)=\sqrt{(x_1+y_1)^2+(x_2+y_2)^2+(x_3+y_3)^2+\cdots+(x_n+y_n)^2}$$

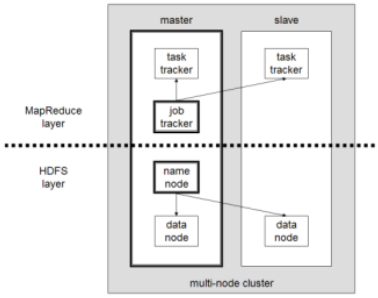
2.3. Hadoop

Hadoop adalah sebuah sistem *open-source* khusus untuk mendukung kinerja komputasi yang dirancang untuk dapat menyimpan dan menganalisa data dalam jumlah besar yang tidak terstruktur. Hadoop memiliki kelebihan dapat secara cepat dan optimal didalam mengolah data yang sangat berukuran sangat besar dengan kualitas *hardware* yang standar. *Framework* Apache Hadoop untuk Hadoop terdiri dari beberapa modul yaitu [3]:

1. Hadoop Common : berisi *library* umum yang mendukung modul Hadoop lainnya.
2. Hadoop Distributed Files (HDFS) : sebuah sistem distribusi file yang mendukung akses ke data aplikasi.
3. Hadoop YARN : *framework* untuk *resource-management* yang mengelola *resource* dalam *cluster* dan juga *scheduling*.
4. Hadoop *MapReduce* : sebuah model pemrograman untuk pengelolaan data yang besar. Hadoop bekerja dengan prinsip membagi-bagi skala data yang berukuran besar menjadi beberapa bagian kecil data dan kemudian memproses data – data skala kecil tersebut secara parallel.

Hadoop dapat bekerja pada sebuah komputer atau lebih (*cluster*). Hadoop memiliki 2 proses utama:

1. *Hadoop Distributed Files (HDFS)*
2. *MapReduce*

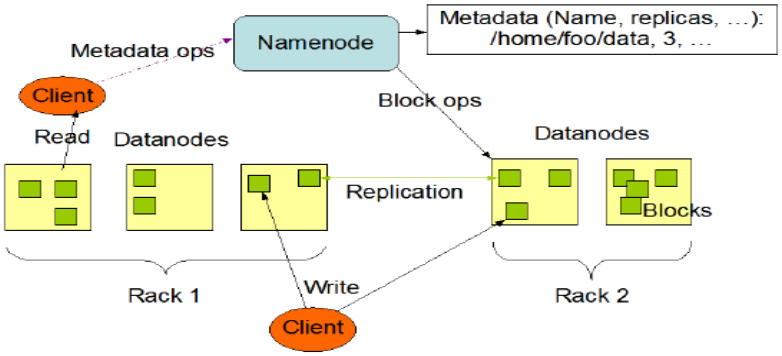


Gambar 2.2. Arsitektur Hadoop [11]

2.4. Hadoop *Distributed File System (HDFS)*

Hadoop *Distributed File System* (HDFS) adalah sebuah sistem pendistribusian file yang didesain untuk bisa dijalankan di *hardware* pada umumnya. Pada umunya HDFS memiliki banyak persamaan dengan sistem distribusi file yang telah ada. Namun, perbedaan HDFS dengan sistem distribusi file yang lain cukup signifikan. HDFS sangat memperhatikan error yang terjadi pada saat pendistribusian file sedang berlangsung dan HDFS didesain sehingga dapat dikembangkan pada *hardware* yang murah.[11]

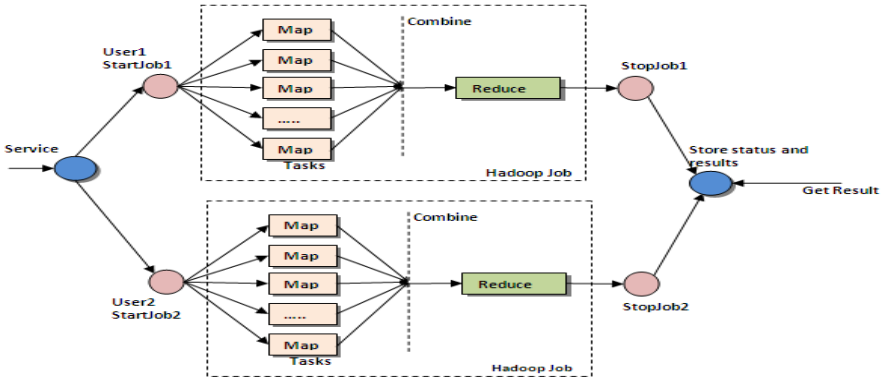
HDFS memiliki 2 buah komponen utama, yaitu *NameNode* dan *DataNode*. *NameNode* adalah sebuah komputer yang berperan sebagai *master* pada *cluster*, sedangkan *DataNode* merupakan komputer-komputer yang berperan sebagai *slave* pada *cluster*. Keberhasilan proses distribusi file sistem pada HDFS ini ditentukan oleh kinerja dari 2 buah komponen utama diatas.



Gambar2.3. Arsitektur Hadoop Distribution File System[11]

2.5. MapReduce

Model algoritma *MapReduce* dikemukakan pertama kali oleh Google yang menyediakan framework yang sangat menarik untuk memproses data yang besar didalam sebuah Cloud[12]. Dalam tingkatan yang lebih tinggi, algoritma *MapReduce* pada dasarnya bekerja dalam beberapa bagian yang terhubung secara parallel pada setiap node dalam sebuah cluster. Dalam implementasinya *MapReduce* memiliki dua tahapan utama dalam menjalankan prosesnya yaitu *Map Procedure* dan *Reduce Procedure*.



Gambar2. 4. Arsitektur MapReduce [11]

2.5.1. Map Procedure

Map Procedure didefinisikan dengan sebuah pasangan masukan kunci/nilai dan keluaran fungsi tersebut adalah himpunan dari pasangan-pasangan kunci/nilai (*intermediate key*). Algoritma *MapReduce* akan mengelompokkan semua pasangan-pasangan kunci/nilai yang diasosiasikan dengan kunci *intermediate i* yang sama dan memberikannya pada fungsi *Reduce*.

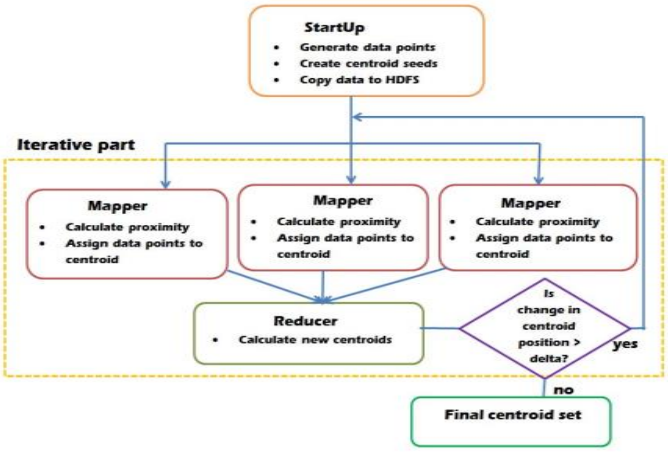
2.5.2. Reduce Procedure

Reduce Procedure didefinisikan dengan mengambil sebuah kunci *intermediate i* dengan himpunan nilai untuk kunci yang dihasilkan oleh fungsi *Map*. Nilai tersebut digabung seluruhnya untuk membentuk nilai yang mungkin sebuah himpunan nilai yang lebih kecil. Biasanya hanya 0 atau 1 himpunan nilai yang keluar untuk setiap fungsi *Reduce* dipanggil. Nilai *intermediate* yang diberikan masuk ke dalam fungsi *reduce* melalui iterator. Iterator tersebut digunakan untuk dapat mengelola nilai yang terlalu besar untuk disimpan dalam memori dalam bentuk list.

2.6. MapReduce K-Means Clustering

Seperti yang dijelaskan pada subbab 2.2. bahwa *K-Means* adalah algoritma *clustering* yang digunakan untuk mengelompokkan satu set objek data menjadi sejumlah cluster berdasarkan jarak antar titik. Maka pemrosesan algortima *MapReduce K-Means Clustering* dilakukan dengan menghasilkan dataset acak yang akan mewakili objek data asli dan membuat sekumpulan data centroid.

Secara garis besar struktur dari implentasi *MapReduce* pada algoritma *K-Means Clustering* sebagai berikut :



Gambar 2.5. Struktur MapReduce pada K-Means Clustering [1]

Langkah awal dalam perancangan *MapReduce* ke dalam algoritma *K-Means Clustering* adalah untuk mennentukan dan menganalisa input dan output dari implementasi. Input didefinisikan sebagai pasangan <key, value> , dimana "key" adalah *cluster* centroid dan "value" adalah implementasi serial dari vektor yang ada pada dataset. Persyaratan untuk menerapkan proses Map dan Reduce adalah memiliki dua file yang harus berisi dari centroid yang telah ditentukan jumlahnya, dan yang lainnya harus memiliki kluster yang udah berupa vector. Centroid - centroid dan vector - vector yang ditentukan tadi dikelompokan secara terorganisir ke dalam dua file yang terpisah [2]. Hal ini dapat digambarkan dengan desain algoritma *MapReduce* untuk *K-Means Clustering* seperti dibawah.

```
Procedure KmeansMapper
      Load Cluster file
      Fp = MapClusterfile
      Create two list
      listnew = listold
      Call read(Mapclusterfile)
      newfp = MapCluster()
      dv= 0
      Assign correct centroid
      read (dv)
      calculate centroid
      dv = minCenter()
      Call KmeansReduce()
end procedure = 0
```

Gambar 2.6. Pseudo-code Procedure Mapper K-Means [2]

Kumpulan centroid awal dimasukkan kedalam sebuah directori input pada HDFS untuk selanjutnya dapat dipanggil oleh *Map Procedure* melalui "key" yang ada pada pasangan <key,value>. Pada *Map Procedure* ini sendiri nantinya akan dilakukan penghitungan jarak antara dataset yang telah ditentukan dengan *cluster* centroid yang juga telah ditentukan sebelumnya. Prosedur ini menghitung jarak melalui nilai vector yang dihasilkan dengan setiap klister centroid yang ada pada dataset cluster dan akan seterusnya mencari jarak terdekat antara vektor tersebut dengan centroid. Setelah *Map Procedure* telah mendapatkan nilai dari jarak cluster centroid terdekat tersebut akan dilakukan penghitungan ulang oleh centroid dari cluster tersebut.

```
procedure KmeansReduceDesign
      NEW ListofClusters
      COMBINE resultant clusters from MAP CLASS
         if cluster size too high or too low then
           RESIZE cluster
      CMax = findMaxSize(ListofClusters)
      CMin = findMinSize(ListofClusters)
      if CMax > 1/20 totalSize then Resize(cluster)
      WRITE cluster FILE to output DIRECTORY
end procedure = 0
```

Gambar 2.7. Pseudo-code Procedure Reducer K-Means [2]

Penghitungan ulang dari setiap centroid diselesaikan dengan *Reduce Procedure*. Prosedur ini juga akan mengelompokan ulang cluster untuk mencegah terbentuknya cluster dengan ukuran yang sangat besar sperti cluster memiliki data vector yang terlalu sedikit atau cluster memiliki terlalu banyak data vector. Setelah centroid telah diperbarui, kumpulan dataset vector dan cluster akan ditulis ulang pada directori dan dapat digunakan untuk iterasi berikutnya.

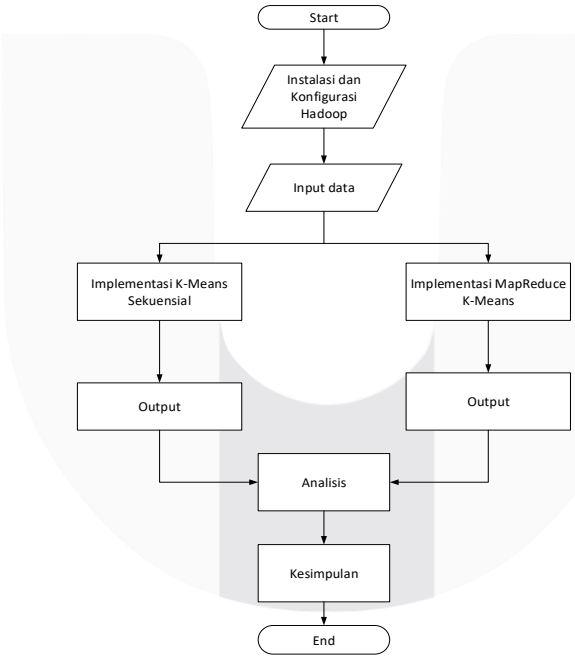
```
procedure Kmeans Function
      if Initial Iteration then
         LOAD cluster file from DIRECTORY
      else READ cluster fie from previous iteration
      Create new JOB
      SET MAPPER to map class defined
      SET REDUCER to reduce class define
      path for output DIRECTORY
      SUBMIT JOB
end procedure = 0
```

Gambar 2.8. Fungsi untuk menjalankan MapReduce K-Means Clustering [2]

3. Analisis Perancangan

3.1. Rancangan Umum

Pada penelitian ini perancangan sistem yang akan dibangun dengan memulai pada proses instalasi dan konfigurasi Hadoop. Kemudian setelah proses konfigurasi dan instalasi selesai dan *environtment* Hadoop telah dapat dijalankan maka akan dilakukan implementasi dari Algoritma *MapReduce K-Means* yang akan dibandingkan dengan algoritma sekuensial *K-Means* dari sisi process running time sebagaimana diilustrasikan sebagai berikut.



Gambar 3.1. Rancangan umum system

3.2. Konfigurasi Hadoop

3.2.1. Spesifikasi *Hardware* dan *Software*

Spesifikasi *hardware* pada Hadoop yang dibangun. Adalah sebagai berikut :

Tabel 1. Spesifikasi Software Hadoop

Software	
Sistem Operasi	Red Hat (64-bit) / CentOS release 6.9 (Final)
Java Version	1.8.0_141
Hadoop Version	Hadoop 2.7.3.2.6.1.0-129

Tabel 2. Spesifikasi Hardware Hadoop

Hardware	
Processor	4 cores Intel(R) Core(TM) i5-4460 CPU @ 3.20GHz (4 CPUs), ~3.2GHz
Vmdk Storrage	48,83 GB
Memory	8 GB

3.2.1. Proses Instalasi

Pada tahap instalasi Hadoop menggunakan Hortonworks Sandbox VM yang akan dijalankan diatas *software* Oracle Virtual Box. Penggunaan Hortonworks Sandbox VM didasarkan terdapatnya Hortonworks sebagai Hadoop distribusi yang berupa *open source* dan telah banyak digunakan oleh professional maupun *developer*. Versi Hortonworks Data Platform (HDP) yang digunakan dalam perancangan system ini adalah versi 2.6.1. Dimana proses Instalasi dan konfigurasi Hadoop tersebut terdapat dengan jelas pada laman website https://hortonworks.com/tutorial/learning-the-ropes-of-the-hortonworks-sandbox/

3.3. Input Data

Adapaun data yang dipakai adalah data set *20 Newsgroup* yang didapat dari repository UCI. Data set *20 Newsgroup* adalah kumpulan sekitar 20.000 dokumen, yang dipartisi (hampir) secara merata dalam 20 kelompok yang berbeda. Data set ini 20 newsgroup telah menjadi kumpulan data populer untuk eksperimen dalam aplikasi teks teknik pembelajaran mesin, seperti klasifikasi teks dan pengelompokkan teks.

Data disusun dalam 20 kelompok yang berbeda, masing-masing sesuai dengan topik yang berbeda. Beberapa newsgroup sangat terkait erat satu sama lain (contoh, comp.sys.ibm.pc.hardware / comp.sys.mac.hardware), sementara yang lain sangat tidak terkait (contoh, Misc.forsale / soc.religion.christian). Berikut adalah daftar dari data set *20 newsgroup*, yang dipartisi menurut materi pelajaran:

Tabel* 3. Dataset *20_Newsgroup

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

4. Implementasi dan Analisis

4.1. Implementasi Sistem

Implementasi sistem yang dilakukan dalam penelitian tugas akhir ini dilakukan dengan menjalankan program *MapReduce K-Means Clustering* yang telah dibuat dan membandingkan proses *running time*-nya dengan algoritma sekuensial *K-Means Clustering*.

4.1.1. Skenario Percobaan

Skenario percobaan merupakan gambaran bagaimana cara percobaan system akan dilakukan. Dalam tugas akhir ini percobaan dilakukan berdasarkan pembagian besar data yang berbeda – beda yang semakin meningkat dimulai dari 100 subset dokumen hingga 100.000 subset dokumen. Kemudian masing – masing bukuran data tersebut juga nantinya akan dilakukan percobaan dengan mengganti jumlah centroid pada masing – masing data.

Dalam setiap proses percobaan akan dihitung dan dibandingkan hasil dari proses *running time* baik dari algoritma sekuensial *K-Means Clustering* dan *MapReduce K-Means Clustering*.

4.1.2. Pengujian Sistem

Terdapat empat file.jar yang digunakan untuk menjalankan algortima sekuensial *K-Means Clustering* dan *MapReduce K-Means Clustering* dan yaitu "KMeans.jar" "ProcessCorpus.jar", "GetCentroids.jar", dan "MapRedKMeans.jar".

Langkah – langkah pengujian system dilakukan dengan tahapan sebagai berikut :

1. Pertama, jalankan file "ProcessCorpus.jar" yang akan mengambil semua data set dan mengubahnya menjadi sekumpulan data *"*. Jalankan dengan perintah :
   `java -jar ProcessCorpus.jar`
2. Akan muncul tampilan perintah dan isi setiap perintah seperti berikut :

```
root@sandbox:~/KMeans
[root@sandbox KMeans]# java -jar ProcessCorpus.jar
Enter the directory where the corpus is located: data
Enter the name of the file to write the result to: data1
Enter the max number of docs to use in each subdirectory: 100
data
Counting the number of occurs of each word in the corpus...Found 3
9907 unique words in the corpus.
How many of those words do you want to use to process the docs? 10
000
Done creating the dictionary.
Converting the corpus to a list of vectors...Done vectorizing all of the docs!
[root@sandbox KMeans]#
```

Gambar 4.1 Perintah ProcessCorpus.jar

Langkah ini akan membuat sebuah file bernama "data1" yang memiliki 20 * 100 baris, yang masing-masing merupakan representasi vektor dari sebuah dokumen. Ukuran masing-masing vektor yang digunakan adalah 10000 kata.

3. Sekarang, jalankan sebuah program yang akan memilih seperangkat centroid awal dari data:
   `java -jar GetCentroids.jar`
4. Akan muncul tampilan perintah dan isi setiap perintah seperti berikut :

```
root@sandbox:~/KMeans
[root@sandbox KMeans]# java -jar GetCentroids.jar
Enter the data file to select the clusters from: data1
Enter the name of the file to write the result to: cent1
Enter the number of clusters to select: 10
..Done selecting centroids.
[root@sandbox KMeans]#
```

Gambar 4.2 Perintah GetCentroids.jar

Proses ini akan membuat sebuah file bernama "cent1" yang memiliki 10 baris, yang masing-masing menggambarkan *cluster* centroid Kumpulan *cluster* centroid awal ini secara acak diambil dari file "data1".

5. Copy file "data1" dan "cent1" tadi ke HDFS untuk memulai menjalankan Mapreduce

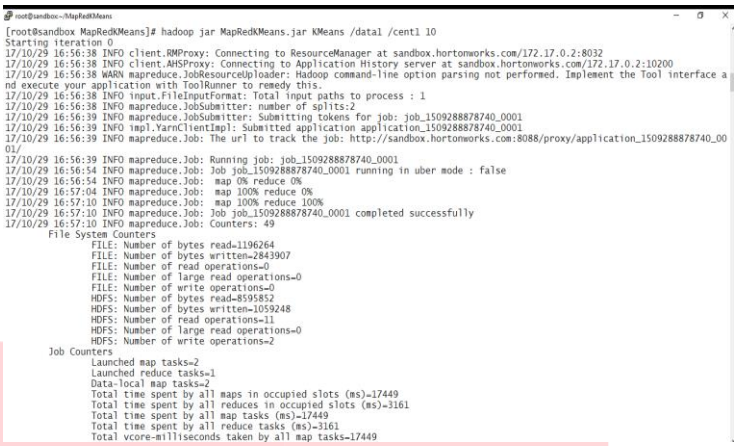
```
root@sandbox:~/KMeans
[root@sandbox KMeans]# hdfs dfs -mkdir /data1
[root@sandbox KMeans]# hdfs dfs -mkdir /cent1
[root@sandbox KMeans]# hdfs dfs -copyFromLocal data1 /data1
[root@sandbox KMeans]# hdfs dfs -copyFromLocal cent1 /cent1
[root@sandbox KMeans]#
```

Gambar 4.3 Perintah mengopy file kedalam HDFS

6. Kemudian jalankan program "Mapreduce *K-Means*" dengan perintah
   `hadoop jar MapRedKMeans.jar KMeans /data1 /cent1 10`

   Proses ini akan menjalankan 10 iterasi algoritma *KMeans* di atas semua 20.000 dokumen dalam kumpulan data 20_newsgroups. "/ data" adalah direktori di HDFS dimana data berada, "/ *cluster*" adalah direktori tempat *cluster* awal berada, dan "10" adalah jumlah iterasi yang akan dijalankan; Ini berarti bahwa tiga pekerjaan *MapReduce* yang terpisah akan dijalankan secara berurutan

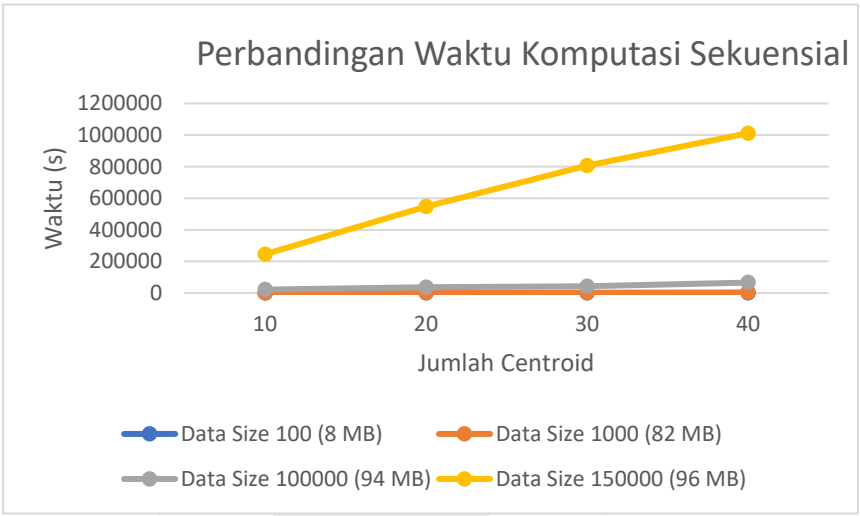
.



Gambar 4.4 Perintah eksekusi MapReduce K-Means

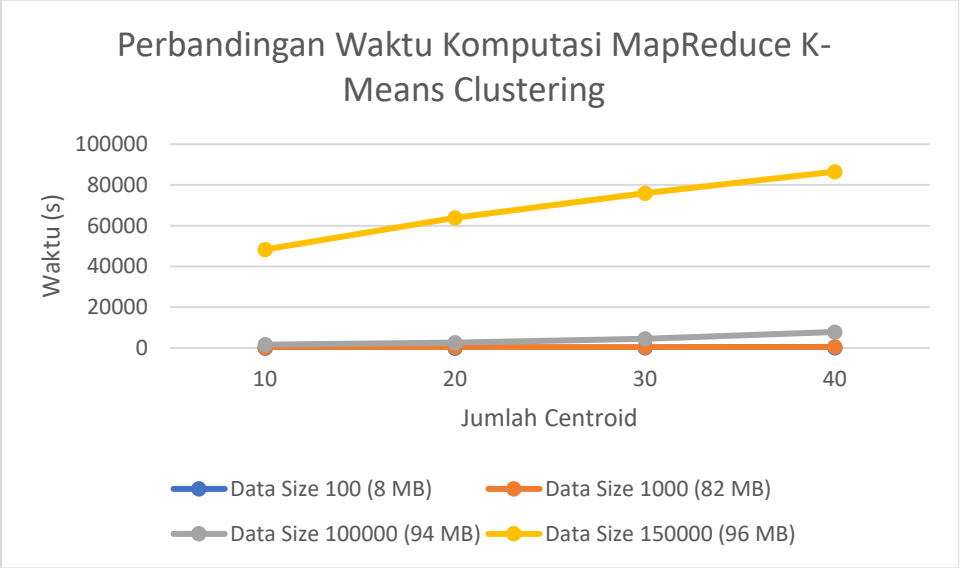
4.2. Hasil dan Analisis Pengujian Sistem

Hasil pengujian system merupakan data atau keterangan dari hasil proses *running time* yang telah dilakukan baik itu pada algoritma sekuensial *K-Means Clustering* dan juga pada *MapReduce K-Means Clustering*. Waktu merupakan salah satu indikator yang dianalisis pada penelitian ini. Berikut adalah hasil waktu komputasi yang didapat.



Gambar 4.5 Perbandingan waktu komputasi sekuensial pada masing – masing data berdasarkan jumlah centroid

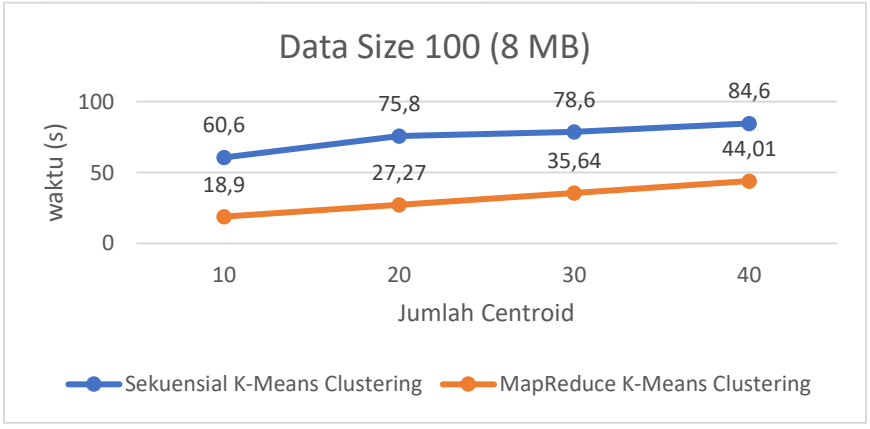
Dapat dilihat pada gambar 4.4 bahwa terdapat peningkatan waktu komputasi yang sangat besar pada *Data Size* yang juga semakin besar. Dengan *Data Size* yang diperbesar perbedaan waktu komputasi antara *Data Size* 100 (8 MB) dengan *Data Size* 1000 (82 MB) menjadi sangat signifikan terlihat. Semakin besar ukuran data, maka semakin banyak juga jumlah nilai dan informasi yang harus dilakukan oleh centroid untuk menemukan *cluster* yang terdekat.



Gambar 4.6 Perbandingan waktu komputasi MapReduce K-Means Clustering pada masing – masing data berdasarkan jumlah centroid

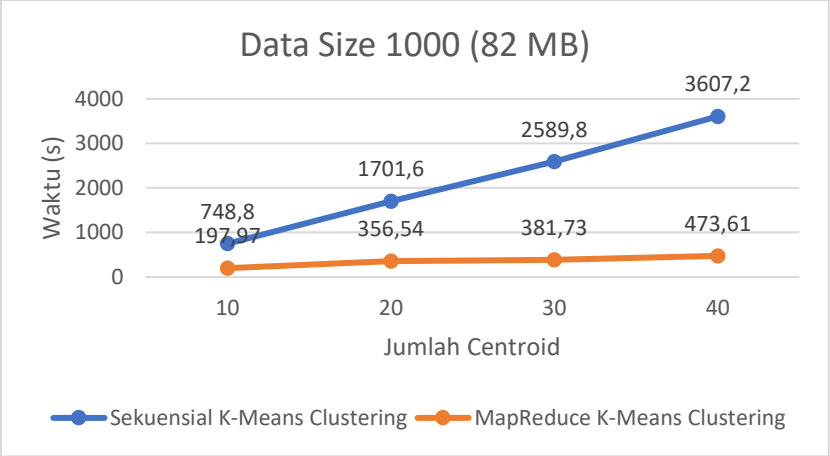
Sama halnya dengan algoritma sekuensial terdapat peningkatan waktu komputasi yang sangat besar pada *Data Size* yang juga semakin besar. Dengan *Data Size* yang diperbesar perbedaan waktu komputasi antara *Data Size* 100 (8 MB) dengan *Data Size* 1000 (82 MB) menjadi sangat signifikan terlihat. Semakin besar ukuran data, maka semakin banyak juga jumlah nilai dan informasi yang harus dilakukan oleh centroid untuk menemukan *cluster* yang terdekat.

Dengan adanya hasil komputasi dari dua algoritma tersebut, dapat dilihat perbandingan waktu komputasi pada algoritma sekuensial *K-Means Clustering* dengan *MapReduce K-Means Clustering* dengan masing – masing data.



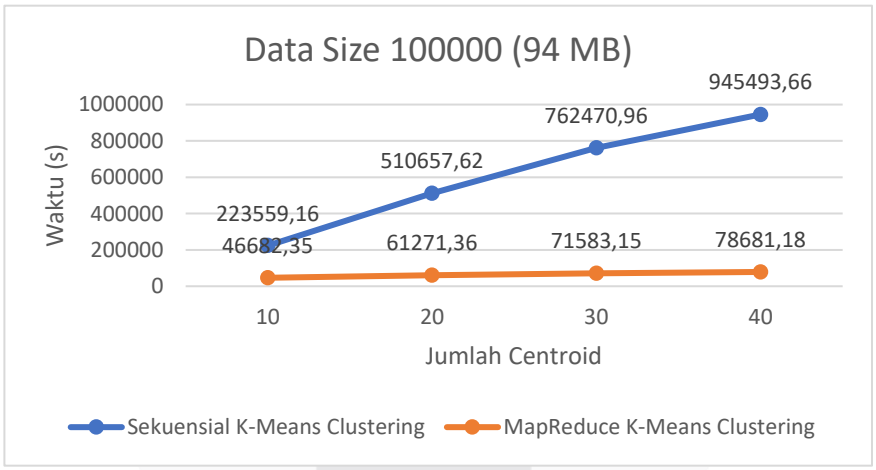
Gambar 4.7 Perbandingan waktu komputasi sekuensial K-Means dan MapReduce K-Means Clustering

Terlihat pada *Data Size* 100 (8 MB) perbedaan waktu komputasi antara kedua algoritma terlihat cukup signifikan. Terdapat perbedaan kecepatan waktu komputasi pada masing – masing jumlah centroid yang ditentukan. Jika pada algoritma *MapReduce K-Means Clustering* peningkatan lama waktu komputasi terlihat cenderung konstan, berbeda pada algoritma sekuensial dimana pada jumlah centroid 20 dan 30 tidak mengalami peningkatan lama waktu komputasi yang signifikan bahkan hampir sama dengan rata – rata lama waktu sebanyak 4,53 second.



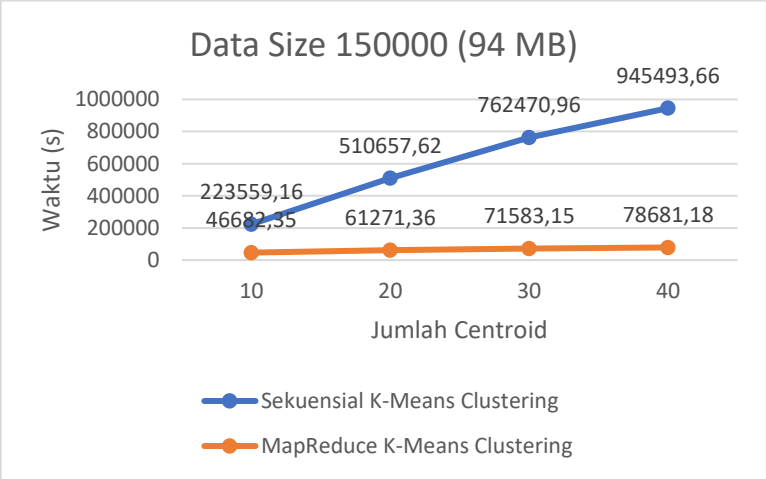
Gambar 4.8 Perbandingan waktu komputasi sekuensial K-Means dan MapReduce K-Means Clustering

Jika pada *Data Size* 100 (8 MB) terlihat bahwa algoritma *MapReduce K-Means Clustering* yang memiliki peningkatan lama waktu komputasi yang konstan, hal ini terjadi sebaliknya pada *Data Size* 1000 (82 MB) dimana terlihat pada Gambar 4.7 bahwa peningkatan lama waktu algoritma sekuensial terlihat cenderung konstan. Namun peningkatan tersebut membutuhkan selisih waktu yang cukup lama antar masing - masing centroid yang ditentukan. Berbeda halnya dengan algortima *MapReduce K-Means Clustering* dimana peningkatan lama waktu komputasi tidak terlalu signifikan.



Gambar 4.9 Perbandingan waktu komputasi sekuensial K-Means dan MapReduce K-Means Clustering

Pemaksimalan *Data Size* yang sebanyak 75 persen membuat peningkatan lama waktu komputasi pada algoritma sekuensial menjadi sangat tinggi pada jumlah centroid 10 dan 20. Peningkatan lama waktu komputasi tersebut sebanyak hampir 10 kali lipat, namun pada jumlah centroid selanjutnya peningkatan tidak terlalu signifikan. Sama halnya pada algoritma MapReduce *K-Means*, peningkatan lama waktu komputasi yang signifikan hanya terlihat pada jumlah centroid 10 dan 20.



Gambar 4.10 Perbandingan waktu komputasi sekuensial K-Means dan MapReduce K-Means Clustering

Dengan *Data Size* yang hampir mencapai maksimum *Data Size* yang tersedia, proses komputasi kedua algoritma tentunya mengalami peningkatan lama waktu komputasi yang sangat besar. Masih sama seperti pada percobaan – percobaan sebelumnya bahwa algoritma sekuensial *K-Means Clustering* mengalami peningkatan lama waktu komputasi yang sangat besar. Hal ini selain dikarenakan oleh ukuran data yang semakin besar, juga disebabkan oleh jumlah centroid yang semakin banyak dan proses dilakukan dengan literasi satu persatu pada setiap pencarian *Euclidian Distance* oleh centroid ke *cluster* yang terdekat.

Algoritma *MapReduce K-Means Clustering* dapat mempercepat proses waktu komputasi dibandingkan dengan menggunakan algoritma sekuensial *K-Means Clustering* dikarenakan pada algoritma *MapReduce K-Means Clustering* ditambahkan fungsi atau prosedur *MapReduce* untuk dapat digunakan pada Hadoop. *MapReduce* dapat mempercepat proses waktu komputasi dikarenakan *MapReduce* memiliki sebuah metode pendistribusian data menggunakan *Hadoop Distributed File System (HDFS)* dengan dua prosedur atau fungsi utama yang ada pada *MapReduce* itu sendiri yaitu prosedur *Map* yang berfungsi untuk membagi atau memetakan semua data yang ada dan mengelompokannya menjadi sekumpulan data yang memiliki ukuran yang lebih kecil daripada data aslinya tanpa mengubah informasi atau nilai yang ada pada data tersebut untuk kemudian data tersebut dapat diolah atau diproses; dan juga prosedur *Reducer* yang berfungsi untuk mennghitung atau memproses nilai atau informasi yang ada pada data yang telah dilakukan proses *Map* sebelumnya dan kemudian mengembalikan data yang tadinya telah diubah dan diproses menjadi output ke sebuah nilai dan data yang asli.

5. Kesimpulan dan Saran

5.1. Kesimpulan

Berdasarkan penilitian yang dilakukan, kesimpulan yang didapat antara lain:

1. Intalasi Hadoop *Single Node* saat ini dapat dengan mudah dilakukan untuk dapat mempelajari lebih lanjut tentang Hadoop dan kegunaannya.
2. Pengimplementasian *MapReduce K-Means Clustering* pada hadoop dapat digunakan untuk mempercepat proses komputasi dalam menjalankan algoritma *K-Means Clustering*. Hal ini dibuktikan dengan hasil percobaan yang telah dilakukan bahwa terlihat jelas algoritma *MapReduce K-Means Clustering* lebih cepat dalam waktu komputasi dibandingkan dengan algoritma sekuensial *K-Means Clustering*.
3. Perbedaan waktu komputasi antara algoritma sekuensial *K-Means Clustering* dengan *MapReduce K-Means Clustering* terlihat sangat siginifikan berbeda, dimana dengan menggunakan algoritma *MapReduce K-Means Clustering* waktu komputasi menjadi hampir 8 kali lebih cepat daripada hanya menggunakan algoritma sekuensial *K-Means Clustering*.
4. Adanya fungsi atau prosedur *MapReduce* pada hadoop sangat membantu dalam pemrosesan data dengan ukuran yang besar untuk mendapatkan waktu komputasi yang lebih cepat daripada menggunakan algoritma sekuensial dengan error yang dapat dikurangi.
5. Fungsi atau prosedur *MapReduce* menjadikan waktu komputasi lebih cepat dikarenakan adanya dua fungsi utama yaitu prosedur *Map* dan prosedur *Reducer* yang digunakan untuk memetakan semua data menjadi ukuran data yang lebih kecil sehingga waktu pendistribusian data untuk diolah menjadi lebih cepat.

5.2. Saran

Beberapa saran untuk pengembangan lebih lanjut yang dapat diberikan penulis adalah:

1. Penggunaan hadoop selanjutnya dapat ditingkatkan dengan tidak lagi menjalankannya pada *Single Node*, namun pada *Multi Node* sehingga terbentuk arsitektur Hadoop Cluster.
2. Penggunaan MapReduce K-Means Clustering bisa dilakukan dengan menggunakan data set yang lain untk memecahkan masalah dalam aplikasi dunia nyata.
3. Proses uji coba atau pengimplementasian algoritma *MapReduce* dapat dilakukan dengan algoritma lain yang memiliki kompleksitas yang lebih besar seperti *Naïve Bayes*.

Daftar Pustaka

[1] Mishra Shweta, Badhe Vivek. (2016), *Improved Map Reduce K Means Clustering Algorithm for Hadoop Architectur* , International Journal Of Engineering and Computer Science, 2016, IJECS.

[2] S.Patil Yaminee, Vaidya M.B. (2015), *K-Means Clustering with MapReduce Technique*, International Journal Of Advenced Research in Computer and Communication Engineering, 2015, IJARCCE.

[3] Apache Hadoop. http://hadoop.apache.org/

[4] Sumarsono. A . N. R, 2015, *Implementasi dan Analisis Perbandingan K-Means dan K-Medoids Paaralel dengan Message Passing Interface(MPI),*Universitas Telkom, Bandung.

[5] Aziz. I. N.,2015, *Analisis Pengolahan Text File pada* Hadoop Cluster *dengan Memperhatikan Kapasitas Random Access Memory (RAM),* Universitas Telkom, Bandung.

[6] Tan, Pang-Ning; Steinbach,Michael; Kumar ,Vipin. *Introduction to Data Mining*.

[7] Jing Zhang, Gongqing Wu, Xuegang Hu, Shiying Li dan Shuilong Hao. (2011), *A Parallel K-Means Clustering Algorithm with MPI*, International Symposium on Parallel Architectures, Algorithms and Programming, 2011 IEEE.

[8] Jain. A.K ,2009. *Data Clustering: 50 Years Beyond K-Means*. Pattern Recognition Letters, 2009

[9] Han, J., Kamber, M., Pei, J.,2012, *Data Mining Concept and Techniques, 3rd ed. Morgan Kaufmann-Elsevier*, Amsterdam

[10] K-Means : http://stanford.edu/~cpiech/cs221/handouts/kmeans.html

[11] S. S Hosale Harshawardhan, Prof. PGadekar Devendrea. (2014), *A Review Paper on Big Data and Hadoop* , International Journal Of Scientific and Research Publications, 2014, IJRSP.

[12] J. Dean and S. Ghemawat, 2004 ,*MapReduce: Simplified Data Processing on LargeCluster*, Google.Inc..

[13] Zaharia. M., Konwinski. A., Joseph. A. D., Katz. R, &Stoica. I.,2008, *Improving MapReduce Performance in Heterogeneous Environments*, OSDI'08