Analisa Perbandingan Original Hadoop Cluster Dan Modifikasi Hadoop Cluster

Iqbal Grady Favian
Jurusan Teknik Informatika, Fakultas Teknologi Informasi
Institut Teknologi Adhi Tama Surabaya
Email: iqbalgrady@gmail.com

Abstract. *Hadoop is an open-source framework. Various improvements may be applied to improve the performance of hadoop for processing data and stabilizing clusters. Stability is definitely required when distributions of blocks in clusters are not in order to keep the clusters stable. The experiments were in two system application versions, i.e. :original and modified. The sizes of files studied in the process were 1, 2, 3, and 4 GB resulting increase of write time speeds of the original version, as basis of comparison, compared to the ones of the modified version by 8.261 seconds, 25.7294 seconds, 9.49695, and 8.8813 seconds. On the other hands, the increases of read time speeds were by 0.1229 seconds, 2.0566 seconds, 24.3564 seconds, and 1.7612 seconds. The difference of average speed time between the original balancer and modified balancer was 3.7 minutes. Based on the results of the data analysis, it showed that block size affected read time speed and write time speed as well as balancer speed of hadoop cluster*

Keywords: Hadoop, Balancer, block size.

Abstrak. *Hadoop* merupakan sebuah framework yang bersifat open-source maka berbagai cara dapat diterapkan untuk meningkatkan performa hadoop untuk pengolahan data dan stabilitas cluster. Stabilitas diperlukan ketika distribusi block pada cluster tidak merata sehingga cluster tetap dalam keadaan stabil. Berdasarkan hasil uji coba yang telah dilakukan dalam 2 penerapan sistem yaitu secara original dan secara modifikasi. Pada prosesnya digunakan file dengan ukuran 1, 2, 3, dan 4 GB dengan hasil pertambahan kecepatan *write time* antara original sebagai dasar dan modifikasi sebesar 8.261 detik, 25.7294 detik, 9.49695, dan 8.8813 detik. Dan pada kecepatan *read time* sebesar 0.1229 detik, 2.0566 detik, 24.3564 detik, dan 1.7612 detik. Sedangkan perbedaan waktu rata-rata kecepatan *balancer* original dan *balancer* modifikasi sebesar 3.7 menit. Berdasar pada hasil data bahwa *block size* dapat mempengaruhi kecepatan *read time, write time* dan *balancer* pada hadoop cluster.

Kata Kunci : *Hadoop, Balancer, block size*.

1. Pendahuluan

Pada saat ini perkembangan IT berkembang dengan pesat. Dengan perkembangan teknologi saat ini, tidak dapat dipungkiri bahwa perkembangan ukuran data juga sangat berpengaruh dimulai dari pemrosesan data yang besar pada suatu kegiatan teknologi. Ukuran data yang semakin besar dan tidak dapat ditampung lagi sehingga sistem *database* manapun sudah tidak dapat menganalisa dan memproses data tersebut dengan cepat, maka dari itu data yang semakin besar mulai dikenal dengan nama *big data*. Dengan adanya *big data* maka proses pencarian, penyimpanan dan analisis pada sebuah data akan membutuhkan waktu yang lama.

Pengembangan teknologi informasi ini bertujuan untuk mempermudah kinerja dalam pencarian data dan penyimpanan data lainnya dan tidak membutuhkan waktu lama untuk dapat melakukan *job* tersebut. *Hadoop* menggunakan bahasa program yang disebut *MapReduce*

sebagai pengolah *big data*. Dengan adanya *MapReduce* maka *Hadoop* data memproses data secara terdistribusi dalam beberapa computer bahkan ribuan, sehingga kinerja *Hadoop* sangat terbantu dengan keberadaan *MapReduce*.

Bila semakin banyak data yang tersimpan dalam satu server, maka semakin penuh penyimpanan *storage server* tersebut. Jika *Hadoop framework* hanya diberikan satu *server* saja, tidak menutup kemungkinan *server* tersebut bisa *down* atau kondisi terburuk mengalamai masalah yang mengakibatkan isi dari *storage* tersebut tidak dapat di akses kembali. Dengan demikian diperlukan penambahan *node* baru sebagai tempat penyimpanan cadangan file. Tapi *node* baru tersebut tidak akan terisi data dengan sendirinya, karena data awal yang ada pada *node* sebelumnya tidak akan berpindah ke *node* baru tanpa adanya modifikasi sistem. Dengan demikian perlu di tambahkan modifikasi sistem untuk pengolahan data agar terbagi secara otomatis pada *server* baru guna mengurangi *storage* penuh pada *server* sebelumnya. Distribusi data ini yang disebut *Hadoop balancer* pada *Hadoop framework*.

Sebagaimana disebutkan diatas, dalam Tugas akhir ini penulis akan membahas tentang implementasi dan analisis optimasi *hadoop framework*.

2. Landasan Teori

2.1. Hadoop

Hadoop merupakan software open-source yang dibuat menggunakan java untuk penyimpanan terdistribusi dan pemrosesan terdistribusi untuk data yang sangat besar yang berada dibawah apache software foundation. Menurut Colin White (2012) Hadoop dapat mengolah data dalam jumlah yang sangat besar hingga *petabyte* (1 *petabyte* = 1024^5 *bytes*) dan dijalankan diatas ratusan bahkan ribuan komputer.

Inti dari apache Hadoop terdiri dari bagian penyimpanan, dikenal sebagai Hadoop Distributed File System (HDFS) dan bagian pengolahan data yang disebut Mapreduce. Hadoop membagi file dalam block besar kepada seluruh node dalam sebuah cluster. Untuk mengolah data, Hadoop mengirim kode kepada node untuk memproses data secara parallel berdasarkan kebutuhan data. Proses ini memiliki keuntungan karena node dapat memanipulasi data yang berada pada node sehingga lebih cepat diakses daripada menggunkan jaringan konvensional yang mengandalkan jaringan kecepatan tinggi dengan arsitektur super computer

2.1.1. Arsitektur Hadoop cluster Multi node

Hadoop memiliki beberapa common paket yang berguna untuk mengakses fasilitas system yang ada pada Hadoop, yaitu Mapreduce dan HDFS. MapReduce merupakan jenis programming yang menganut system terdistribusi, sedangkan HDFS (Hadoop distributed file system) merupakan distribusi file system yang menganut system terdistribusi.

2.1.2. Kelebihan Hadoop

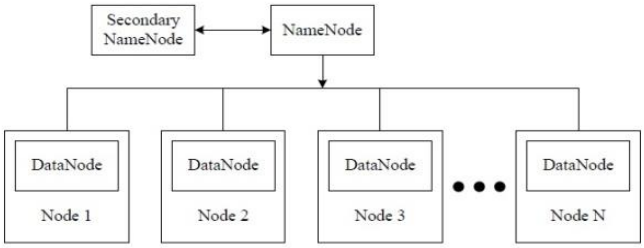
Hadoop memiliki beberapa kelebihan, yaitu (1) Mengatasi overload jaringan. (2) Multitasking dilakukan secara independen. (3) Simpel program karena hanya menggunakan satu model programming yaitu mapreduce. (4) HDFS sederhana dan dapat diandalkan. (5) HDFS dapat diakses oleh banyak client dengan menambahkan beberapa cluster server.

2.2. Hadoop Distributed File System (HDFS)

Hadoop distributed file system merupakan system yang terdistribusi, scalable, dan merupakan file system portable yang dibuat berdasarkan java programming untuk Hadoop framework. Yang dimaksud file system terdistribusi pada Hadoop yaitu data tidak disimpan dalam satu harddisk saja, tetapi disimpan dalam beberapa harddisk computer dalam cluster. File di distribusikan dalam bentuk blok (blok default 128 MB) pada masing-masing node dalam cluster. Pengaruh HDFS ini adalah untuk mendukung ukuran blok yang besar (untuk *file system*) dan optimasi data lokal untuk mengurangi *input/output* jaringan (Holmes, 2012).

2.2.1. Struktur HDFS

HDFS memiliki beberapa struktur pendukung utama dalam menjalankan tugasnya yaitu NameNode, Secondary NameNode, dan Data Node.



Gambar 1. Komponen HDFS, Sumber : (Dima May. 2012)

Pada gambar 1 pada komponen HDFS, disebutkan namenode sebagai pusat system dan menjadi penghubung pada semua datanode. Pada namenode terdapat secondary namenode yang bertugas sebagai penyimpan informasi terbaru dan konfigurasi terbaru pada cluster untuk membackup kinerja sementara namenode ketika tidak aktif.

2.2.2. Kelebihan dan kekurangan HDFS

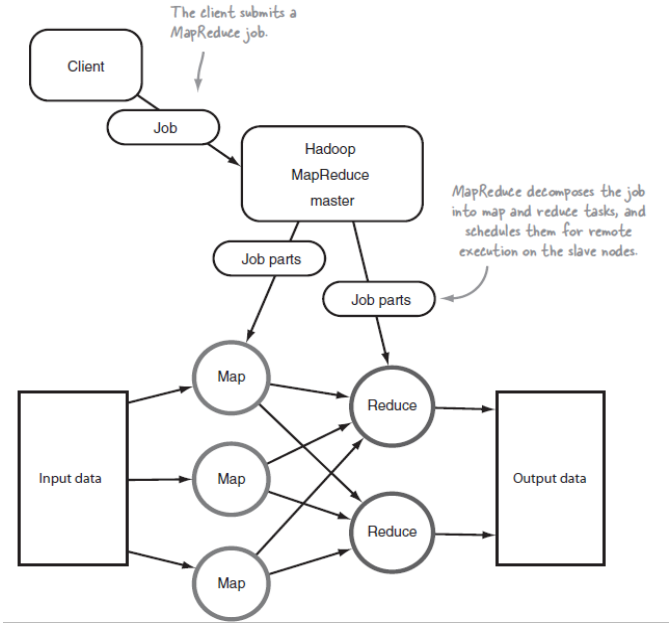
Kelebihan HDFS sebagai distributed file system adalah : (1) HDFS didesain untuk menyimpan data dengan ukuran yang sangat besar (mencapai *terabyte* bahkan *petabyte*)**.** (2) HDFS didesain untuk mengakses informasi secara cepat karena menangani banyak client yang terhubung. (3) HDFS menyimpan data secara reliable, artinya jika sebuah komputer gagal beroperasi maka data masih ada pada komputer yang lain**.** (4) HDFS terintegrasi dengan Hadoop Map Reduce, sehingga memudahkan membaca dan mengkomputasi data secara lokal**.** Kelemahan HDFS yaitu name node bersifat Single Point of Failures, jika server name node mati maka data akan hilang.

2.3. Hadoop Balancer

Balancer pada Hadoop framework merupakan daemon yang bertugas untuk menjaga kestabilan dari cluster Hadoop. Sehingga tidak ada ketimpangan dalam cluster tersebut. Peran balancer ditambahkan secara default ketika layanan HDFS diinstal.

2.4. Mapreduce

MapReduce adalah perangkat lunak yang digunakan untuk memproses data dalam jumlah besar dengan proses parallel pada cluster yang besar (ribuan komputer). Ketika client memberi job pada mapreduce, maka mapreduce membagi job tersebut menjadi beberapa bagian menjadi map dan reduce. Map berfungsi untuk mensorting dan menyaring data kemudian diteruskan pada reduce yang berfungsi untuk menghitung operasi hasil Map sebelumnya.

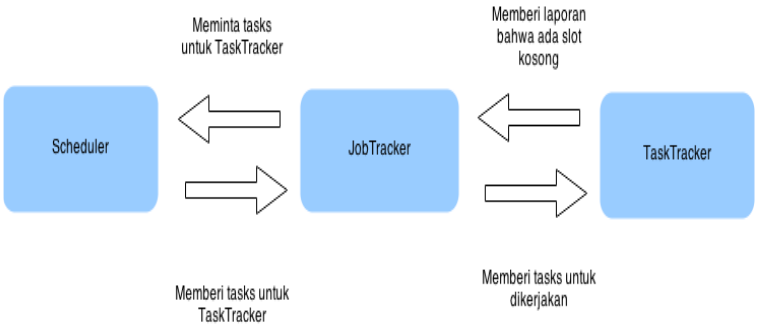


Gambar 2. Arsitektur MapReduce

2.4.1. Struktur Mapreduce

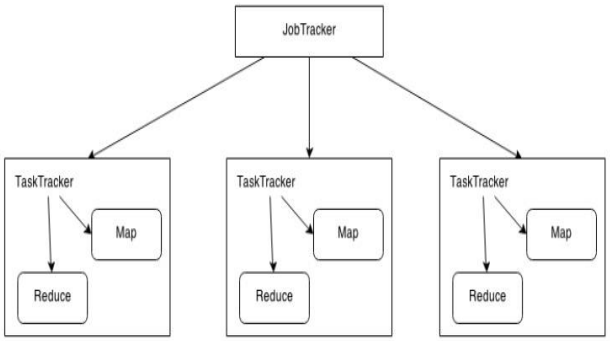
MapReduce memiliki 2 bagian struktur utama dalam prosesnya yaitu:

a. JobTracker



Gambar 3. Kerja *JobTracker* pada Hadoop

b. TaskTracker



Gambar 4. Kerja TaskTracker pada setiap *node, Sumber : (*Chuck Lam. 2011)

Gambar 4 menunjukkan kerja dari proses job tracker, task tracker, map dan reduce pada cluster. Terlihat bagaimana job tracker memberi job pada task tracker pada node cluster kemudian job tersebut di eksekusi oleh task yang kemudian di proses oleh map dan reduce. Kemudian hasilnya akan dikirim kembali ke job tracker.

2.4.2. Keuntungan Map Reduce

Keuntungan Mapreduce adalah : (1) Dapat menerapkan proses *map* dan *reduce* secara terdistribusi. (2) Proses *mapping* dan *reducing* dapat dijalankan secara paralel dalam waktu yang sama (selama *output* dari proses *mapping* mengirimkan *key value* yang sesuai dengan proses *reducing*nya).

2.5. Data Mining Pada Hadoop Framework

Hadoop framework secara default tidak melakukan analysis pada data dengan complex. Ada beberapa tools yang dapat membawa Hadoop Framework kedalam data mining salah satunya Apache Mahout. Apache mahout membawa beberapa algorithma untuk digunakan dalam memproses data mining, berikut merupakan salah satu algoritma yang sering digunakan yaitu Naïve Bayes.

3. Perancangan Sistem

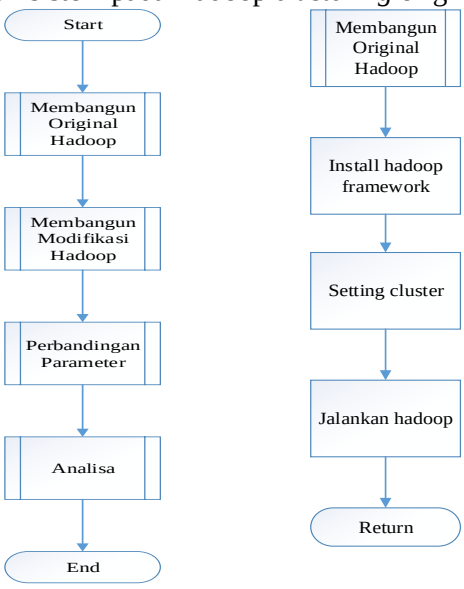
Analisa dan perancangan system ini dilakukan untuk pengoptimasian Hadoop *cluster*. Mulai dari pembahasan topologi yang dibuat dan setiap perangkat keras yang digunakan untuk implementasi sistem.

3.1. Tahapan penelitian

Analisa dan perbandigan original dan modifikasi HDFS system melalui beberapa tahapan yaitu (1) identifikasi dan perumusan masalah. (2) studi literature. (3) perancangan system. (4) konfigurasi system. (5) pengujian system dan (6) pembuatan laporan.

3.2. Desain Analisis System Hadoop Framework

Alur proses pengujian sistem pada Hadoop *clustering* original dan modifikasi:

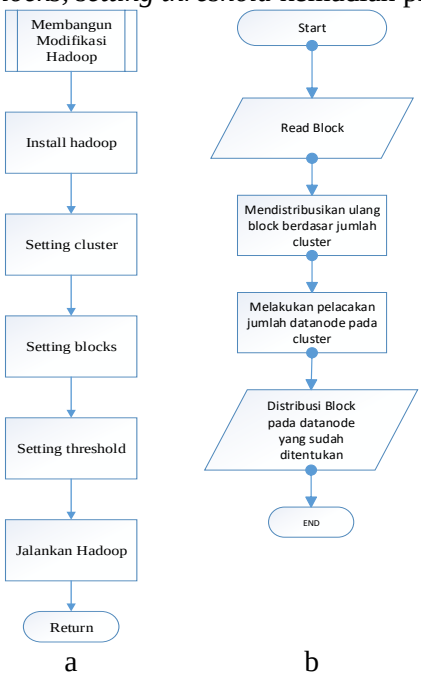


Gambar 5. *Flowchart* Pengujian (a) Sistem Hadoop *framework* dan (b) *Framework* Original

Gambar 5 menjelaskan tentang alur proses yang dikerjakan penulis untuk melakukan pengujian mengenai Hadoop *framework* original dan Hadoop *framework* modifikasi.

3.4. Desain System Hadoop Framework Modifikasi

Proses installasi Hadoop tidak berbeda dengan Hadoop original tetapi mulai berbeda pada saat *administrator* melakukan *setting cluster*, *setting block* yang digunakan untuk proses distribusi *block* pada tiap file, kemudian *setting threshold* yang digunakan untuk proses distribusi *block* pada *node* lama ke *node* baru sehingga terjadi keseimbangan antar *node* dalam *cluster*. Proses sistem Hadoop *framework* yang telah dimodifikasi dimulai dari proses instal Hadoop, *setting cluster*, *setting blocks*, *setting threshold* kemudian proses operasi Hadoop.



Gambar 6. *Flowchart (a)* Hadoop *Balancer* dan (b) Modifikasi

3.5. Hadoop Balancer

Hadoop balancer merupakan daemon yang bertugas untuk memproses distribusi block pada HDFS. Jadi ketika ada datanode yang tidak mendapat jatah block size sesuai dengan datanode lain dalam cluster, maka balancer akan mendistribusikan block yang ada pada datanode lain ke datanode tersebut yang masih belum terdistribusi block. Alur flowchart ditunjukan pada Gambar 6.

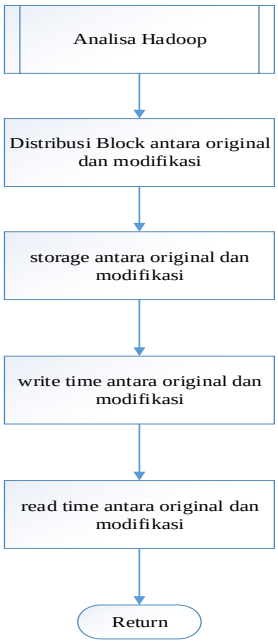
Pada Gambar 6 dijelaskan proses pembagian block ketika menggunakan daemon Hadoop balancer. Hadoop balancer akan membaca berapa jumlah block yang ada pada datanode kemudian mendeteksi jumlah datanode pada cluster. Jika ditemukan ketidakseimbangan jumlah block antar datanode dalam cluster, maka Hadoop balancer langsung membagi block tersebut sama rata antar datanode dalam cluster.

3.6. Desain Proses Perbandingan Parameter Hadoop Original dan Modifikasi

Pada proses ini menjelaskan tentang proses perbandingan parameter yang digunakan untuk mengukur kinerja dari Hadoop original dan Hadoop modifikasi. Proses yang dibandingkan antara lain proses distribusi *block* antar *node* dalam *cluster*, perbandingan *storage* antar *node* dalam *cluster*, proses *write time* dalam *cluster*, dan proses *read time* dalam *cluster*. Proses *write time* yang dimaksud yaitu melihat seberapa cepat Hadoop memproses *input*-an dari *user* ke dalam sistem *HDFS*. Dan proses *read time* yaitu untuk melihat seberapa cepat *Hadoop* dalam membaca file pada sistem *HDFS*.

3.7. Desain Proses Analisa Hadoop Framework Original dan Modifikasi

Pada proses ini, dijelaskan mengenai proses analisa pada Hadoop original dan Hadoop modifikasi. Alur proses analisa :



Gambar 7. *Flowchart* Analisa Hadoop *Framework*

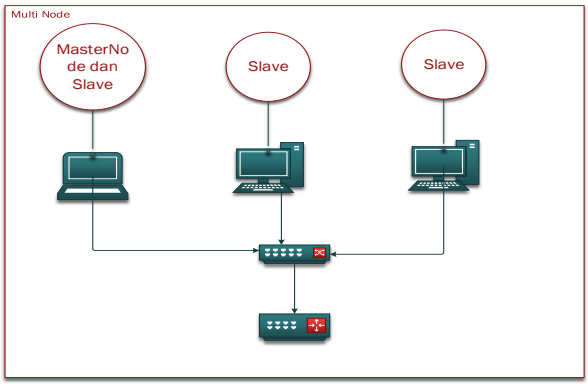
Analisa dimulai dari proses distribusi *block* antar *node* pada sistem original dan sistem modifikasi, kemudian menganalisis kapasitas *storage* antar *node* antara sistem original dan sistem modifikasi, proses *write time* antara sistem original dan sistem modifikasi, dan proses *read time* antara sistem original dan modifikasi.

3.8. Original Hadoop Cluster

Pada tahap ini akan di jelaskan beberapa topologi yang akan mendukung original Hadoop *cluster*.

3.8.1. Topologi Hadoop Cluster Original

Pada topologi ini, *Hadoop* akan dijalankan dengan *physical machine* secara *multi node*.



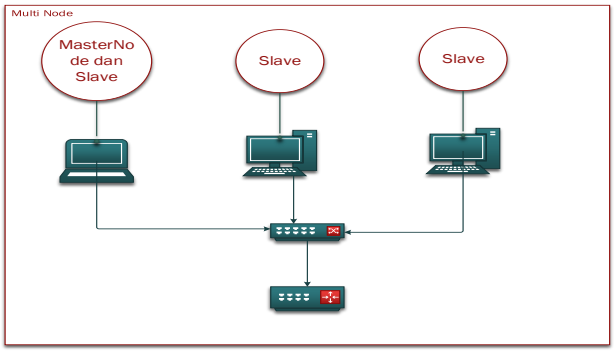
Gambar 8. Rancangan Topologi *Multi Node*

Pada Gambar 8 dijelaskan mengenai susunan topologi *cluster* kedua yang menggunakan 3 *physical machine* untuk menjalankan Hadoop *framework*. Pada *cluster* tersebut digunakan *switch* agar masing-masing *node* dapat terhubung satu sama lain.

3.9. Modifikasi Hadoop Cluster

3.9.1. Topologi Hadoop Cluster Modifikasi

Pada topologi ini, Hadoop akan dijalankan dengan *physical machine* secara *multi node* ditunjukkan pada Gambar 9.



Gambar 9. Rancangan Topologi *Multi Node*

4. Implementasi Sistem

4.1. Analisa Perbandingan

Pada analisa perbandingan ini akan di lakukan analisis terhadap proses *write time*, *read time*, dan hadoop *balancer* pada hadoop original dan hadoop modifikasi.

4.1.1. Analisa Write Time

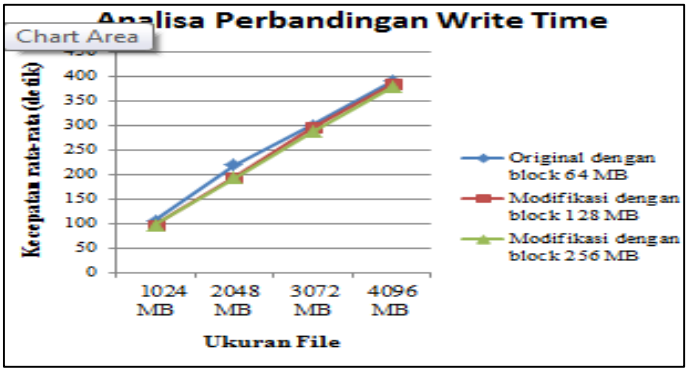
Pada proses analisa ini, akan di bandingkan performa antara *write time* original dan *write time* modifikasi. Hasil uji coba akan tertera pada table berikut:

Tabel 1. Analisa Write Time

Uji Coba Cluster	Kecepatan rata-rata dalam detik			
	File 1024 MB	File 2048 MB	File 3072 MB	File 4096 MB
Original	105.862	219.416	300.219	390.261
Modifikasi I	97.89	193.901	295.412	384.529
Modifikasi II	97.3124	193.473	286.958	378.230

Pada tabel 1 menampilkan data analisa kecepatan *write time* Original, Modifikasi I dan Modifikasi II. Data pada tabel 4.1 merupakan total dari rata-rata uji coba yang sebelumnya telah dilakukan pada masing-masing cluster. Setiap cluster melakukan uji coba sebanyak 10 kali tiap file sehingga didapat nilai rata-rata seperti diatas. Dari data yang diperoleh perubahan antara

original dan modifikasi pada file dengan ukuran kecil tidak berbeda jauh dikarenakan adanya pembagian jumlah *block* berdasarkan konfigurasi dari masing-masing cluster tersebut. Maka ketepatan pengaturan block sangat berpengaruh dalam proses *write time*. Pada original digunakan block 64 MB, dengan pembagian ukuran file dan *block*, maka ukuran *block* pada *cluster* original lebih kecil dan jumlah *block*-nya lebih banyak dari pada modifikasi. Sehingga kinerja hadoop lebih banyak tetapi lebih cepat ketimbang modifikasi. Sedangkan pada modifikasi I dan Modikasi II pada file kecil meskipun kecepatan lebih cepat daripada original, tetapi antara *block* 128 dan 256 tidak terdapat perbedaan yang mencolok karena jumlah *block* yang dihasilkan sama-sama sedikit dan ukuran *block* yang dihasilkan lebih besar dari pada original. *Cluster* modifikasi lebih optimal dalam file dengan ukuran besar karena ketika menggunakan original maka jumlah *block* yang dihasilkan banyak dan mengakibatkan *task tracker* menerima banyak *job* dari *job tracker*, sedangkan modifikasi menghasilkan jumlah *block* yang lebih sedikit dari pada original sehingga *task tracker* pada data *node* lebih cepat untuk proses eksekusinya. Waktu diperoleh tergantung kondisi jaringan yang ada ketika pada saat uji coba



Gambar 10. Grafik Analisa Perbandingan Write Time

Pada Gambar 10 merupakan penerapan dari data pada tabel 1 yang terlihat pada gambar bahwa ukuran *block* mempengaruhi proses eksekusi *write time* pada *benchmark* TestDFSIO. Ukuran block tidak memiliki pengaruh signifikan pada file 1024 MB melihat dari grafik yang tampil, tetapi memiliki perubahan yang terlihat pada file ukuran 3072 dan 4096 MB pada grafik. Sehingga semakin menjelaskan bahwa ukuran block yang tepat dapat meningkatkan performa kecepatan write time pada hadoop sesuai pada penjelasan pada Tabel 1.

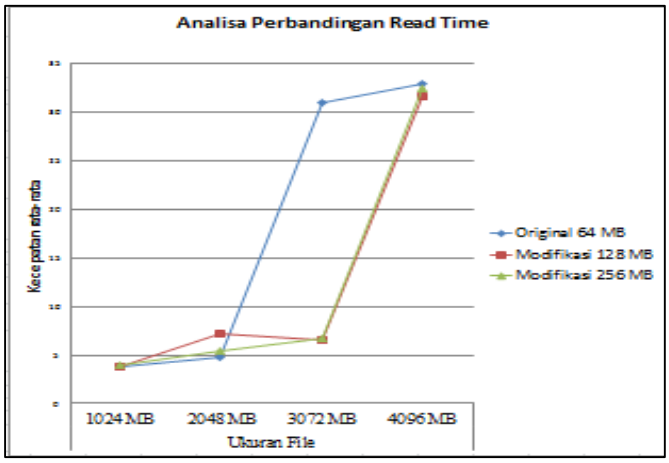
4.1.2. Analisa Read Time

Pada proses analisa ini, akan di bandingkan performa antara *read time* original dan *read time* modifikasi. Hasil uji coba akan tertera pada Table 2.

Tabel 2. Analisa Read Time

Uji Coba Cluster	Kecepatan rata-rata			
	File 1024 MB	File 2048 MB	File 3072 MB	File 4096 MB
Original	3.7975	4.8134	30.9902	32.9688
Modifikasi I	3.7842	7.1567	6.6198	31.6929
Modifikasi II	3.9138	5.3868	6.6478	32.4835

Pada tabel 2 mengenai analisa *read time* menjelaskan mengenai berapa kecepatan yang diperoleh antara hadoop original dan modifikasi. Pada waktu terlihat *relative* sama tanpa ada perbedaan yang mencolok dikarenakan master *node* telah menyimpan metadata dari tiap-tiap file yang telah didistribusikan pada cluster sehingga proses *read time* antar file *cluster* original maupun modifikasi *relative* sama. Waktu diperoleh tergantung kondisi jaringan ketika pada saat uji coba.



Gambar 11. Grafik Analisa Perbandingan Read Time

Pada Gambar 11 merupakan penerapan data dari Tabel 2 yang menunjukkan proses kecepatan *write time* yang *relative* sama. Tergantung dari keadaan jaringan pada saat uji coba.

4.1.3. Block Pada Original dan Modifikasi

Berdasarkan data pada Tabel 1 dan Tabel 2 maka dibuat tabel block untuk digunakan sesuai kebutuhan:

Tabel 3. Ukuran *block*

Ukuran block	Ukuran file			
	1024 MB	2048 MB	3072 MB	4096 MB
64 MB	Cocok	Cocok	-	-
128 MB	-	-	Cocok	Cocok
256 MB	-	-	Cocok	Cocok

Berdasarkan analisis pada tabel 1 dan 2, maka pada tabel 3 dibuat untuk sebagai acuan penerapan ukuran *block* pada masing-masing file yang ingin digunakan. *Block* 64 MB cocok digunakan pada file 1024 dan 2048 MB karena membuat kinerja *task tracker* lebih cepat dalam eksekusi *block*-nya karena pecahan *block* yang mempunyai ukuran kecil. Tapi *block* 64 MB tidak efisien untuk ukuran file 3072 dan 4096 MB karena jumlah *block* lebih banyak dan membuat *task tracker* lama dalam eksekusi *block*-nya. Sedangkan pada *block* 128 dan 256 MB tidak efisien karena jumlah *block* pada file 1024 dan 2048 MB sedikit tetapi ukuran *block*-nya besar, sedangkan pada file ukuran 3072 dan 4096 jumlah *block* sedikit dan ukuran *block* seimbang dengan jumlah *block* maka kinerja *task tracker* lebih cepat ketimbang harus mengeksekusi jumlah *block* yang banyak pada *block* 64 MB.

4.1.4. Analisa Balancer

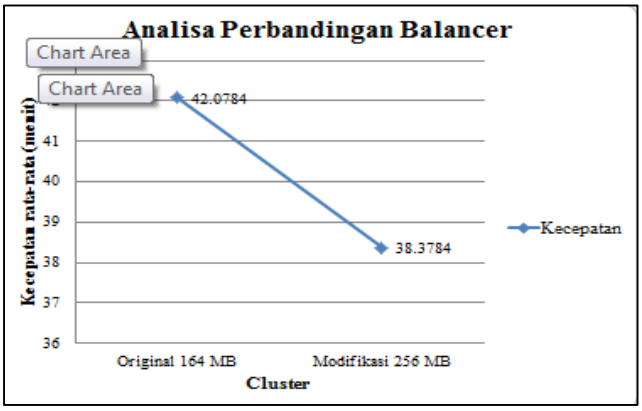
Pada proses analisa ini, akan di bandingkan performa antara *balancer time* original dan *balancer time* modifikasi. Hasil uji coba akan tertera pada tabel berikut:

Tabel 4. Analisa Balancer

Cluster	Kecepatan rata-rata (menit)
Original	42.0784
Modifikasi	38.3784

Pada tabel 4 mengenai analisa *balancer*, dapat dilihat bahwa performa *balancer* modifikasi lebih unggul jauh daripada original. Hal ini terjadi karena ukuran *block* pada modifikasi adalah 256 MB dan original 64 MB. Untuk *balancing* data yang besar, dibutuhkan ukuran *block* yang besar karena sangat berpengaruh, sedangkan pada block 64 MB akan memecah ukuran file seperti pada table sesuai dengan jumlah bagi antara file dengan *block* 64 MB (*File/64*). Butuh proses yang lebih lama karena pengerjaannya banyak. Berbeda dengan *block* 256 MB yang mempunyai jumlah *block* lebih sedikit dan waktu pengerjaan oleh *task* lebih cepat. Pada balancer original maupun modifikasi, menggunakan threshold dengan nilai 10, karena nilai 10 merupakan standar penggunaan dalam hadoop untuk memindahkan block secara

tepat dan tidak mengakibatkan hilangnya beberapa block. Berbeda dengan ketika menggunakan threshold diatas 10, resiko kehilangan block file dan penyebaran block yang tidak merata dapat lebih besar melihat dari kinerja balancer yang diharuskan cepat dalam memindahkan block pada cluster.



Gambar 15. Grafik Analisa Perbandingan Balancer

Pada gambar 15 merupakan penerapan dari data pada tabel 4 yang memperlihatkan kecepatan *balancer* pada hadoop *custer* original dan modifikasi.

5. Penutup.

5.1. Kesimpulan

Dari hasil analisis yang telah dilakukan pada bab sebelumnya, dapat di ambil kesimpulan :

1. Berdasarkan pada hasil analisa dan uji coba hadoop framework original dan modifikasi, dihasilkan proses Write time untuk file ukuran 1024 MB adalah 105.8622 detik untuk original, 97.89 detik untuk modifikasi block 128 MB dan 97.3124 untuk modifikasi block 256 MB. Write time untuk file ukuran 2048 MB adalah 219.4167 detik untuk original, 193.901 detik untuk modifikasi block 128 MB dan 193.4736 untuk modifikasi block 256 MB. Write time untuk file ukuran 3072 MB adalah 300.2199 detik untuk original, 295.412 detik untuk modifikasi block 128 MB dan 286.9582 untuk modifikasi block 256 MB. Write time untuk file ukuran 4096 MB adalah 390.2612 detik untuk original, 384.529 detik untuk modifikasi block 128 MB dan 378.2308 untuk modifikasi block 256 MB.
2. Berdasarkan pada hasil analisa dan uji coba hadoop framework original dan modifikasi, dihasilkan proses Read time untuk file ukuran 1024 MB adalah 3.7975 detik untuk original, 3.7842 detik untuk modifikasi block 128 MB dan 3.9138 untuk modifikasi block 256 MB.
   Read time untuk file ukuran 2048 MB adalah 4.8134 detik untuk original, 7.567 detik untuk modifikasi block 128 MB dan 5.3868 untuk modifikasi block 256 MB. Read time untuk file ukuran 3072 MB adalah 30.9902 detik untuk original, 6.6198 detik untuk modifikasi block 128 MB dan 6.6478 untuk modifikasi block 256 MB.
   Read time untuk file ukuran 4096 MB adalah 32.9688 detik untuk original, 31.6929 detik untuk modifikasi block 128 MB dan 32.4835 untuk modifikasi block 256 MB.
3. Pada performa balancer hadoop di peroleh data untuk hadoop original 42.0784 menit untuk balancing block data dan untuk hadoop modifikasi 38.3784 meni untuk balancing block data.
4. Berdasarkan hasil uji coba maka untuk hadoop cluster dalam proses write data pada ukuran file 1, 2, 3, dan 4 GB menggunakan modifikasi block 256 MB dengan rata-rata waktu tiap file 1 GB adalah 97.3124 detik, file 2 GB 193.4736 detik, file 3 GB 286.9582 dan file 4 GB 378.2308.
5. Berdasarkan hasil uji coba hadoop balancer antara original dan modifikasi maka hadoop balancer modifikasi lebih optimal dengan waktu rata-rata 38.3784 menit.

5.2. Saran dan Pengembangan

Dari hasil evaluasi uji coba sistem hadoop framework yang telah di kerjakan, ada baiknya di pertimbangkan hal-hal berikut :

1. Digunakan node yang dikhususkan untuk big data.
2. Menambah jumlah node dalam cluster agar lebih banyak perbandingan performa antar cluster.
3. Menggunakan aplikasi bencmark lain seperti teragen dan sebagainya.
4. Menggunakan aplikasi streaming seperti apache storm untuk streaming data secara online.
5. Membangun hadoop untuk message brodcast pada suatu cluster menggunakan apache kafka.
6. Membangun aplikasi untuk melakukan proses distribusi database pada hadoop.

Referensi

Chuck Lam. (2011). *Hadoop In Action*. Stamford: Mainning Publications Co.

Colin White. (2012). *MapReduce and the Data Scientist*. BI Research

Dima May. (2012). *Hadoop Distributed File System (HDFS) Overview*. coreservlets.com.

Holmes, A. (2012). *Hadoop in Practice*. New York: Manning Publications Co

White, T. (2012). *Hadoop: The Definitive Guide (3rd ed.)*. O'Reilly Media, Inc

Priagung, Khusumanegara. (2014). Analisis Performa Kecepatan Mapreduce Pada Hadoop Menggunakan Tcp *Packet Flow Analysis*. *Skripsi*. Depok : Universitas Indonesia

Nchimbi Edward Pius, Liu Qin, Fion Yang, Zhu Hong Ming.(2012). *Optimizing Hadoop Block Placement Policy & Cluster Blocks Distribution*. International Journal of Computer, Electrical, Automation, Control and Information Enginering.