Jurnal Informatika Terpadu

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

EVALUASI KINERJA HDFS SEBAGAI INFRASTRUKTUR PEMBANGUNAN BIG DATA

Yunita Surahman[1], Henry Saptono[2]

[1, 2]Teknik Informatika, Sekolah Tinggi Teknologi Terpadu Nurul Fikri
Jakarta Selatan, DKI Jakarta, Indonesia 12640
yunitasurahman1@gmail.com , henry@nurulfikri.co.id

Abstract

This research is proposing the implementation and analysis of HDFS as a big data development infrastructure. In this study, there are steps to implement HDFS and testing to determine the factors used to determine HDFS performance and determine HDFS performance as a big data development infrastructure. There are five tests, namely, testing the read and write execution time using the TestDFSIO benchmarks application, NNBench, changing blocksize, deleting files, and testing availability. The results obtained from testing on HDFS as a big data development infrastructure are the factors that determine the performance of the HDFS system design are the file size and block size. And also stated that the larger the size of the block size makes the execution time faster.

Keywords: *Performance, HDFS, TestDFSIO, NNBench, Blocksize*

Abstrak

Penelitian ini diajukan untuk implementasi dan analisis HDFS sebagai infrastruktur pembangunan *big* data.Pada penelitian ini terdapat langkah-langkah yang dilakukan dalam implementasi HDFS dan pengujian untuk mengetahui faktor-faktor yang digunakan untuk menentukan kinerja HDFS dan mengetahui kinerja HDFS sebagai infrastruktur pembangunan *big* data. Terdapat 5 pengujian yaitu, pengujian waktu eksekusi *read* dan *write* menggunakan aplikasi *benchmarks* TestDFSIO, NNBench, pengubahan *blocksize*, penghapusan file, dan pengujian *availability*. Hasil yang didapat dari pengujian pada HDFS sebagai infrastruktur pembangunan *big* data ini yaitu Faktor-faktor yang menentukan kinerja dari rancangan sistem HDFS adalah ukuran file dan ukuran *blocksize.* Dan juga menyatakan bahwa semakin besar ukuran dari *blocksize* membuat waktu eksekusinya menjadi lebih cepat.

Kata kunci: Kinerja, HDFS, TestDFSIO, NNBench, Blocksize

1. PENDAHULUAN

Seiring perkembangan teknologi informasi yang cepat memberikan konsekuensi pertumbuhan dan peningkatan jumlah data, di dalam sebuah riset dari *Vcould News* mencatat pada tahun 2015, pertumbuhan data statistik per hari mencapai 2,5 quintillion (10 pangkat 18) byte atau sekitar Giga Byte (GB) perdetik. Pada tahun 2018 pertumbuhan data ini diproyeksikan mencapai 50 ribu GB perdetik [1]. Dengan angka sebesar ini membuat suatu perusahaan atau organisasi mencari cara untuk menyimpan serta mengolah data yang banyak tersebut. Data yang banyak dapat diolah serta dianalisis sehingga melahirkan sebuah *value*. Nilai yang merupakan hasil analisis tersebut dapat dijadikan sebuah pertimbangan dalam pengambilan keputusan disuatu perusahaan atau organisasi.

Pertumbuhan data yang besar menyebabkan teknologi *big* data yang ada saat ini menjadi solusi untuk menyimpan serta mengolah data yang banyak tersebut, maka dibutuhkan juga suatu sistem yang mampu memanajemen data-data tersebut hingga menghasilkan suatu *value*. *Project open source* bernama Hadoop hadir sebagai salah satu sistem yang mampumemanjemen data secara terdistribusi

Pada tahun 2005, Doug Cutting dan Mike Cafarella menciptakan Hadoop saat bekerja pada perusahaan Yahoo! Hadoop adalah inspirasi yang didapatkan dari mainan gajah kecil berwarna kuning milik anak Doug Couting. Hadoop versi 0.1.0 akhirnya rilis pada bulan April 2006, sampai versi terakhir Hadoop yang rilis pada Maret 2017 adalah Apache Hadoop 2.8. Padaversi terbaru ini, layanan yang diberikan Hadoop juga termasuk untuk HDFS (*Hadoop*

Diterima 07 Juli 2018, Direvisi 12 Agustus 2018, Diterima untuk publikasi 15 September 2018

Distributed File System), Yarn (*Yet Another Resource Negotiator*) dan MapReduce [2].

Hadoop sendiri menawarkan teknologinya yaitu HDFS (Hadoop Distributed File System) dimana data-data yang ada akan di distribusikan dalam bentuk *block-block* data untuk disimpan dalam setiap media penyimpanan (*node*) dalam sebuah *cluster*. Penggunaan Hadoop di era *cloud computing* seperti saat ini sangatlah dibutuhkan karena hadoop sendiri menyebediakan *file system* yang dapat menyimpan data dalam ukuran besar secara *scalable*. MapReduce juga termasuk kedalam project Apache Hadoop yang dimana HDFS dan MapReduce yang berguna untuk menyelesaikan permasalahan data besar berbasiskan *Java* dan *open source*. Hadoop *file system* juga mengklaim dirinya memiliki fitur toleransi tinggi, ketersediaan tinggi, keandalan data, replikasi, *scalability*, dan *distributed storage*.

1.1 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan di atas, rumusan masalah dari penelitian ini adalah:

1. Langkah-langkah apa saja yang dilakukan dalam implementasi HDFS sebagai infrastruktur pembangunan Big Data?
2. Faktor- faktor apa saja yang menentukan kinerja dari rancangan sistem HDFS yang diteliti?
3. Bagaimanakah kinerja HDFS sebagai infrastruktur pembangunan Big Data?

1.2 Tujuan Penelitian

1. Faktor-faktor yang digunakan untuk menentukan kinerja HDFS.
2. Melakukan penelitian atau pengukuran unjuk kerja HDFS.

1.3 Manfaat Penelitian

1. Memahami langkah-langkah yang dilakukan.
2. Mengetahui faktor-faktor yang menentukan kinerja HDFS.
3. Menghasilkan suatu karya tulis yang bisa menjadi rujukan didalam mengimplementasikan HDFS sebagai infrastruktur pembangunan *big data.*

1.4 Batasan Masalah

Batasan masalah dalam penelitian ini adalah:

1. Lingkungan penelitian adalah lingkungan Virtualisasi, dimana sistem yang dipasang perangkat Hadoop adalah *Virtual Machine.*
2. Sistem HDFS ini diimplementasikan dalam konsep sistem terdistribusi dengan jumlah *node* adalah 3.
3. Penilaian kinerja peneliti menggunakan peralatanatau *tools* yang sudah ada atau tidak membuatnya sendiri.
4. *Platform* Sitem Operasi yang digunakan Homogen (Linux).
5. Faktor yang dianalisa untuk menentukan kinerja HDFS adalah waktu eksekusi faktor I/O atau *read & write* menggunakan aplikasi *benchmark* Hadoop NNBench & TestDFSIO, pengaruh *block size* pada HDFS, dan pengujian *availability* atau ketersediaan data.

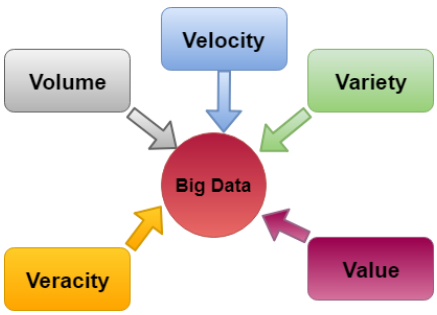
2. LANDASAN TEORI

2.1 Big Data

Big data adalah data yang sudah sangat sulit untuk dikoleksi, disimpan, dikelola maupun dianalisa dengan menggunakan sistem database biasa karena volumenya yang terus berlipat[3]. Dapat disimpulkan bahwa big data adalah data dengan ciri berukuran sangat besar, sangat variatif, sangat cepat pertumbuhannya danmungkin tidak terstruktur yang perlu diolah khusus dengan teknologi inovatif sehingga mendapatkan informasi yang mendalam dan dapat membantu pengambilan keputusan yang lebih baik serta tidak dapat di proses menggunakan alat tradisional biasa dan harus menggunakan cara dan alat baru untuk mengolahdata sehingga menjadi nilai.

Big data memiliki 5 karakteristik yaitu:

1. *Volume* yaitu sejumlah data besar yang dihasilkan perdetiknya.
2. *Velocity* yaitu kecepatan data baru yang dihasilkan dan pertumbuhan serta perubahannya.
3. *Variety* yaitu mengacu pada berbagai jenis data.
4. *Veracity* yaitu kebenaran serta keakuratan darisuatu data.
5. *Value* yaitu kemampuan kita mengubah datayang ada menjadi sebuah nilai.



Gambar 1. Karakteristik Big Data

2.2 Kinerja

Menurut kamus besar bahasa Indonesia, kinerja adalah kemampuan kerja. Kinerja suatu *software* bisa menurun karena suatu hal, dan kinerja suatu software juga bisa dioptimalkan. Untuk mengoptimalkan kinerja suatu *software* kita harus mengetahui faktor-faktor yang dapat mengoptimalkannya

2.3 Hadoop

Hadoop atau Apache Hadoop adalah *software open source* yang ditulis menggunakan bahasa Java untuk dijalankan

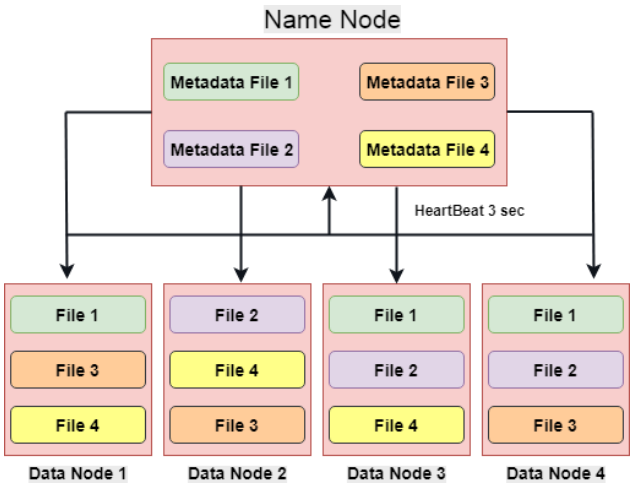
secara terdistribusi dan *scalable.* Hadoop dibangun berdasarkan algoritma MapReduce dari Google Inc. Hadoop menyediakan penyimpanan besar- besaran untuk jenis data, serta kemampuan untuk menangani tugas-tugas atau pekerjaan secara bersamaan.

Hadoop sendiri sejak tanggal 23 Januari 2008 telahmenjadi proyek tingkat atas yang dimiliki lingkungan Apache Software Foundation dan dikembangkan secara terbuka oleh komunitas kontributor secaraglobal. Owen O'Malley adalah orang yang diikut sertakan ke proyek Hadoop pada Maret 2006, dan hadoop 0.1.0 dirilis pada April 2006.

2.4 *Hadoop File System* (HDFS)

HDFS adalah sistem file yang dirancang untuk menyimpan file yang sangat besar dengan pola akses data streaming, berjalan di cluster pada perangkat keras komoditas [4]**.** Hadoop File System atau HDFS adalah sebuah file terdistribusi, skalabel, dan portable yang ditulis menggunakan bahasa pemrograman Java untuk kerangka Hadoop. HDFS menyimpan file besar (biasanya dalam kisaran gigabytes hingga terabytes) di dalam beberapa mesin. HDFS akan melakukan proses pemecahan file besar menjadi bagian-bagian lebih kecil yang kemudian akan didistribusikan ke *cluster-cluster* dari komputer. Data yang di potong menjadi lebih kecil disebut *block* dan ukuran *block* dapat diatur sesuai kebutuhan.

Setiap data atau file yang kita simpan di HDFS akan selalu memiliki lebih dari satu *copy* data atau file tersebut. Hal tersebut di namakan Replication Factor (RF) dimana satu file disimpan di 3 data *node* sehingga jika ada satu DataNode yang rusak, maka DataNode yang lainnya bisa memberikan filenya.



Gambar 2. HDFS

2.5 *Block Size*

Block size adalah potongan-potongan data yang tersimpan di dalam HDFS [5]. Ukuran block size pada HDFS umumnya adalah 128 MB. Dengan begitu, file HDFS dipotong menjadi potongan 128 MB, dan jika memungkinkan setiap potongan akan berada dalam DataNode yang berbeda.

2.6 *MapReduce*

Kerangka kerja perangkat lunak untuk menulis aplikasi dengan mudah yang memproses sejumlah besar data (kumpulan data multi-terabyte) secara paralel pada kelompok besar (ribuan node) perangkat keras komoditas dengan cara yang dapat diandalkan dan toleran kesalahan [6].

2.7 *Yet Another Resource Negotiator* (YARN)

YARN adalah platform sumber daya generik untuk mengelola sumber daya dalam sebuah typical cluster. YARN diperkenalkan pada Hadoop 2.0, yang merupakan kerangka kerja pemrosesan sumber terbuka yang terdistorsi dari Apache Software Foundation. YARN juga diciptakan oleh nama MapReduce 2.0. ini sejak apache hadoop mapreduce telah dirancang ulang apache hadoop YARN.YARN sebagai bahan komputasi generik untuk mendukung MapReduce dan paradigma aplikasi lainnya dalam cluster hadoop yang sama.

2.8 *Java Development Kit* (JDK)

Java Development Kit (JDK) adalah implementasi salah satu *Platform* Java, Edisi Standar, *Platform* Java, Edisi *Enterprise*, atau *platform Micro Edition* yang dirilis oleh Oracle Corporation dalam bentuk produk biner yang ditujukan untuk Pengembang Java di Solaris, Linux, MacOS atau Windows.

2.9 *Secure Shell* (SSH)

Secure Shell adalah sebuah protokol jaringan kriptografi untuk komunikasi data yang aman, login antarmuka baris perintah, perintah eksekusi jarak jauh, dan layanan jaringan lainnya antara dua jaringan komputer. Semua komunikasi antara klien dan server dienkripsi dengan aman dan dilindungi dari modifikasi [7].

2.10 TestDFSIO

TestDFSIO adalah aplikasi *benchmark* hadoop yang berfungsi menguji kinerja I/O pada HDFS dan juga ini sangat membantu untuk tugas-tugas seperti stress testing HDFS, untuk menemukan bottleneck kinerja di jaringan [8].

2.11 *NameNode Benchmarks* (NNBENCH)

NNBench adalah aplikasi *benchmark* hadoop yang memiliki tujuan untuk memeriksa konfigurasi NameNode pada multinode cluster hadoop. Pengujian ini dilakukan dengan menggunakan pekerjaan MapReduce sebagai cara yang nyaman untuk membaca dan menulis file secara parallel. Tolok ukur nnbench berguna untuk menguji-beban namenode. Benchmark ini mensimulasikan volume tinggi permintaan manipulasi file terhadap HDFS untuk "stress-test" kemampuan namenode untuk mengelola HDFS [9].

2.12 Throughput

Throughput adalah kemampuan sebenarnya suatu jaringan dalam melakukan pengiriman data. Biasanya throughput selalu dikaitkan dengan bandwidth, karena throughput bisa juga disebut dengan bandwidth. Akan tetapi bandwidth bersifat *fix*, sedangkan throughput sifatnya adalah dinamis tergantung *traffic* yang sedang terjadi [10].

3. ANALISIS DAN PERANCANGAN

3.1 Analisis Kebutuhan *Hardware*

Tabel 1. Analisis Kebuthan *Hardware*

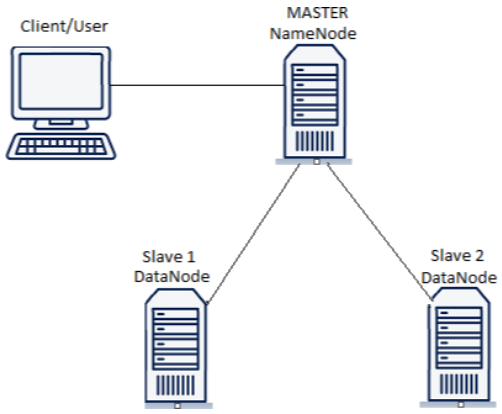
Spesifikasi	NameNode	DataNode1	DataNode2
Processor	One Core	One Core	One Core
RAM	1.5 GB	1.5 GB	1.5 GB
System Operation	Linux Ubuntu 16.04	Linux Ubuntu 16.04	Linux Ubuntu 16.04

3.2 Analisis Kebutuhan *Software*

Dalam penelitian analisis kinerja HDFS sebagai infrastruktur pembangunan *big* data, peneliti menggunakan perangkat lunak (*software)* yang akan digunakan:

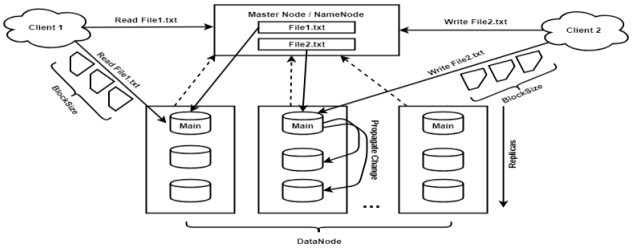
- VirtualBox
- Java Development Kit (JDK) 1.8
- Secure Shell (SSH)
- OS Linux Ubuntu 16.04
- Hadoop 2.7.2
- MapReduce/YARN

3.3 Perancangan Sistem



Gambar 3. Perancangan Sistem

3.4 Rancangan Pengujian



Gambar 4. Rancangan Pengujian

4. IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi

1. Instalasi Java pada NameNode dan DataNode dengan mendownload pada websitenya. Lalu ekstrak dengan perintah:`sudo tar xvzf jdk-8u191-linux-x64.tar.gz`

2. Membuat User Hadoop yang akan dijadikan user untuk menjalankan hadoop di NameNode dan DataNode, dengan perintah:

   ```
   $sudo su
   $sudo addgroup hadoop
   $sudo adduser hduser
   $sudo adduser hduser hadoop
   ```

3. Menambah alamat IP pada konfigurasi NameNode dan Datanode dengan perintah:

   ```
   $sudo /nano/hosts
   ```

4. Melakukan instalasi ssh dengan perintah:

   ```
   $sudo apt-get install openssh-server
   ```

5. Disable IPV6 dengan perintah:`$sudo nano /etc/sysctl.conf`

6. Sebelum melakukan instalasi, download hadoop terlebih dahulu kemudian dapat melakukan instalasi hadoop-2.7.2 dengan perintah:

   ```
   $cd /usr/local
   $sudo tar –xvf hadoop-2.7.2.tar.gz
   $sudo ln –s hadoop-2.7.2 hadoop
   ```

7. Karena hadoop berplatform aplikasi java maka penulis menambahkan konfigurasi pada aplikasi hadoop:

   ```
   $sudo nano /usr/local/hadoop/etc/hadoop/hadoop-env.sh
   #export JAVA_HOME=/usr/local/java/jdk1.8.0_191
   ```

8. Kemudian konfigurasi hadoop untuk merubah parameter dalam memproses dokumen.

   ```
   $ sudo nano
   ```

```
/usr/local/hadoop/etc/hadoop/core-
site.xml
$ sudo nano
/usr/local/hadoop/etc/hadoop/hdfs-
site.xml
$ sudo nano
/usr/local/hadoop/etc/hadoop/yarn-
site.xml
$ sudo nano
/usr/local/hadoop/etc/hadoop/mapred-
site.xml
```

9. Format NameNode dengan perintah:

```
$hdfs namenode –format
```

10. Menjalankan hadoop HDFS:

```
$ start-dfs.sh
```

11. Menjalankan YARN dengan perintah:

```
$ start-yarn.sh
```

4.2 Pengujian

A. Menjalankan Aplikasi TestDFSIO

```
$ hadoop jar
/usr/local/hadoop/share/hadoop/mapr
educe/hadoop-mapreduce-client-
jobclient-2.7.2-tests.jar TestDFSIO
–write –nrFiles 1 –fileSize 500
$ hadoop jar
/usr/local/hadoop/share/hadoop/mapr
educe/hadoop-mapreduce-client-
jobclient-2.7.2-tests.jar TestDFSIO
–read –nrFiles 1 –fileSize 500
```

B. Menjalankan Aplikasi NNBeach

```
$hadoop jar
/usr/local/hadoop/share/hadoop/mapr
educe/hadoop-mapreduce-client-
jobclient-2.7.2-tests.jar nnbench -
operation create_write –
numberOfFiles500
$hadoop jar
/usr/local/hadoop/share/hadoop/mapr
educe/hadoop-mapreduce-client-
jobclient-2.7.2-tests.jar nnbench -
operation open_read –
numberOfFiles500
```

C. Pengubahan *Blocksize*

```
$ start=$(date +'%s%N'/1000000) &&
hadoop fs -D dfs.block.size=
201326592 –put Desktop/perc1
/pengujian1 && echo "it took $(($(
date +'%s%N'/1000000) - $start))
miliseconds"
```

D. Penghapusan File

```
$ start=$(date +'%s%N'/1000000) &&
hadoop fs –rmr –R
/pengujian1/perc1 && echo "it
took $(($(date +'%s%N'/1000000) –
$start)) miliseconds"
```

E. Pengujian *Availability*

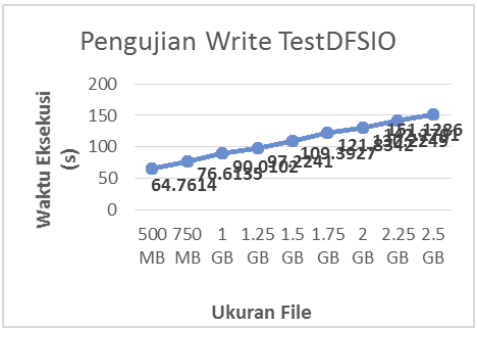
Pengujian *availability* atau ketersedian data menggunakan skenario dimana salah satu datanode di non-aktifkan, dan namenode melakukan penulisan (*write)* ke HDFS. Pada pengujian ini, peneliti menjadikan datanode2 sebagai datanode yang di-non aktifkan saat menyalin atau menuliskan file ke dalam HDFS. Setelah namenode selesai melakukan penulisan pada HDFS, datanode2 yang di non-aktifkan akan diaktifkan kembali, lalu dicek apakah datanode2 tersebut bisa mengakses atau terdapat data yang namenode sudah tuliskan atau salin dari *local* komputerke HDFS.

4.3 Hasil Pengujian

A. Pengujian menggunakan TestDFSIO

Tabel 2. Rata-Rata Hasil Pengujian Waktu Eksekusi *Write* TestDFSIO

Ukuran Data	Rata-Rata Waktu
500 MB	64.7614
750 MB	76.6135
1 GB	90.0102
1.25 GB	97.2241
1.5 GB	109.3927
1.75 GB	121.8342
2 GB	130.2249
2.25 GB	142.2701
2.5 GB	151.1286



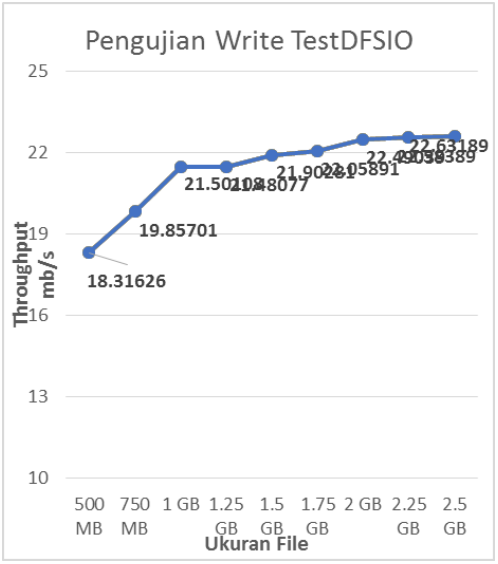
Gambar 5. Grafik Pengujian Waktu Eksekusi *Write* TestDFSIO

Dari hasil pengujian waktu eksekusi *write* TestDFSIO, menghasilkan waktu eksekusi (ms) yang kemudian peneliti ambil rata-rata waktu eksekusi tersebut. Dari tabel 2 dan gambar grafik 5 dapat dilihat bahwa terdapat peningkatan waktu dalam eksekusi *write* file pada tiap size file yang berbeda dan dapat disimpulkan dari hasil percobaan, bahwa

semakin besar data yang diinputkan atau dituliskan kedalam HDFS maka dibutuhkan waktueksekusi yang lebih banyak untuk menyelesaikannya atau mendistribute file tersebut kedalam multinode cluster.

Tabel 3. Rata-Rata Hasil Pengujian Throughput *Write* TestDFSIO

Ukuran Data	Rata-Rata Throughput
500 MB	18.31626
750 MB	19.85701
1 GB	21.50108
1.25 GB	21.48077
1.5 GB	21.90281
1.75 GB	22.05891
2 GB	22.49039
2.25 GB	22.58389
2.5 GB	22.63189

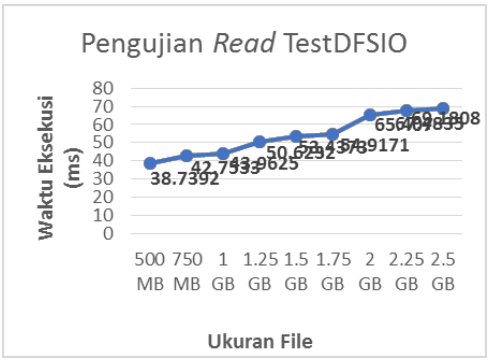


Gambar 6. Grafik Pengujian Throughput *Write* TestDFSIO

Dari hasil pengujian throughput *write* TestDFSIO menghasilkan throughput yang kemudian peneliti ambil rata-rata waktu throughput tersebut. Throughput merupakan rate atau kecepatan ideal proses transferdata. Dari tabel 3 dan gambar grafik 6 dapat dilihat bahwaterdapat penurunan dan peningkatan throughput dalam eksekusi write file pada tiap size file yang berbeda.

Tabel 4. Rata-Rata Hasil Pengujian Waktu Eksekusi *Read* TestDFSIO

Ukuran Data	Rata-Rata Waktu
500 MB	38.7392
750 MB	42.7533
1 GB	43.9625
1.25 GB	50.6232
1.5 GB	53.4373
1.75 GB	54.9171
2 GB	65.4070
2.25 GB	67.4835
2.5 GB	69.1808

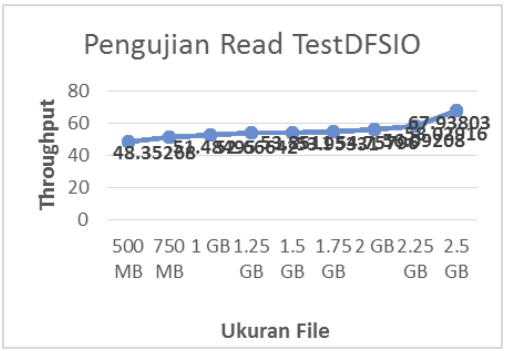


Gambar 7. Grafik Pengujian Ekseskusi *Read* TestDFSIO

Dari pengujian waktu eksekusi *read* TestDFSIO, menghasilkan waktu eksekusi (ms) yang kemudian peneliti ambil rata-rata waktu eksekusi tersebut. Dari tabel 4 dan gambar grafik 7 dapat dilihat bahwa terdapat peningkatan waktu dalam eksekusi *read* file pada tiap size file yang berbeda dimana semakin banyak jumlah file yang di *create* maka membutuhkan waktu yang lebih banyak.

Tabel 5. Rata-Rata Hasil Pengujian Waktu *Throughput Read* TestDFSIO

Ukuran Data	Rata-Rata Throughput
500 MB	48.35268
750 MB	51.48495
1 GB	52.66642
1.25 GB	53.85110
1.5 GB	53.95331
1.75 GB	54.75796
2 GB	56.09208
2.25 GB	58.02916
2.5 GB	67.93803



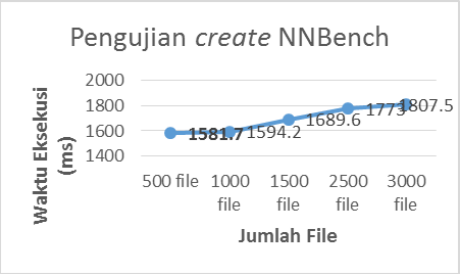
Gambar 8. Grafik Hasil Pengujian Throughput *Read* TestDFSIO

Dari hasil pengujian throughput *read* TestDFSIO, menghasilkan throughput yang kemudian peneliti ambil rata-rata waktu throughput tersebut. Throughput merupakan rate atau kecepatan ideal proses transferdata. Dari tabel 5 dan gambar grafik 8 dapat dilihat bahwa terdapat penurunan dan peningkatan throughput dalameksekusi *read* file pada tiap size file yang berbeda.

B. Pengujian Menggunakan Aplikasi *Benchmarks NNBench*

Tabel 6. Rata-Rata Hasil Pengujian *Create* NNBench

Ukuran data/file	Rata-Rata Waktu
500 File	1.5817
1000 File	1.5942
1500 File	1.6896
2500 File	1.7730
3000 File	1.8075

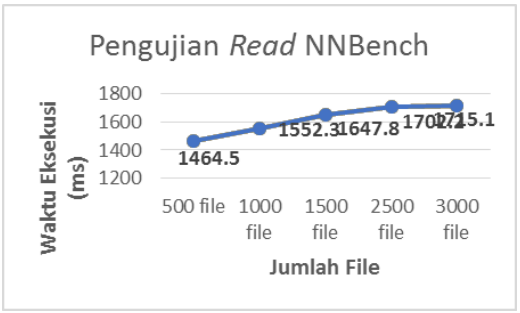


Gambar 9. Grafik Hasil Pengujia *Create* NNBench

Dari hasil pengujian *create* NNBench, kemudian peneliti ambil rata-rata waktu eksekusi tersebut. Dari tabel 6 dan gambar grafik 9 dapat dilihat bahwa terdapat peningkatan waktu dalam eksekusi create file pada tiapjumlah file yang berbeda, dimana semakin banyak file yang di*create* atau di*write* maka akan menyebabkan bertambahnya atau semakin lamanya waktu eksekusinya.

Tabel 7. Rata-Rata Hasil Pengujian *Read* NNBench

Ukuran data/file	Rata-Rata Waktu
500 File	1.4645
1000 File	1.5523
1500 File	1.6478
2500 File	1.7022
3000 File	1.7151



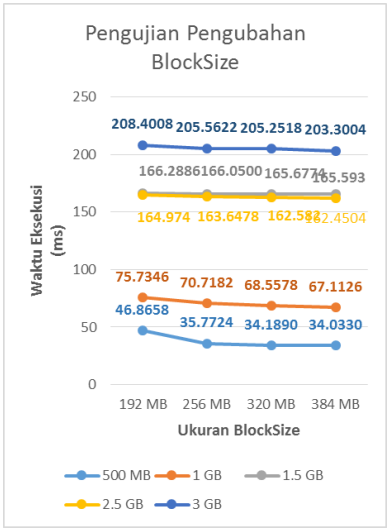
Gambar 10. Grafik Hasil Pengujia *Read* NNBench

Dari hasil pengujian *read* NNBench, kemudian peneliti ambil rata-rata waktu eksekusi tersebut. Dari tabel 7 dan gambar grafik 10 tersebut dapat dilihat bahwa terdapat penurunan serta peningkatan waktu dalam eksekusi *read* file pada tiap jumlah file yang berbeda. Peneliti menyimpulkan dari hasil percobaan, bahwa semakin banyak file yang dieksekusi *read* maka dibutuhkan waktu eksekusi yang lebih banyak untuk menyelesaikannya.

C. Pengujian Pengubahan *BlockSize*

Tabel 8. Rata-Rata Hasil Pengujian *Bloxksize*

Ukuran Data	Blocksize 192 MB	Blocksize 256 MB	Blocksize 320 MB	Blocksize 384 MB
500 MB	46.8658	35.7724	34.1890	34.0330
1 GB	75.7346	70.7182	68.5578	67.1126
1.5 GB	166.2886	166.0500	165.6774	165.693
2.5 GB	164.974	163.6478	162.582	162.4504
3 GB	208.4008	205.5622	205.2518	203.3004



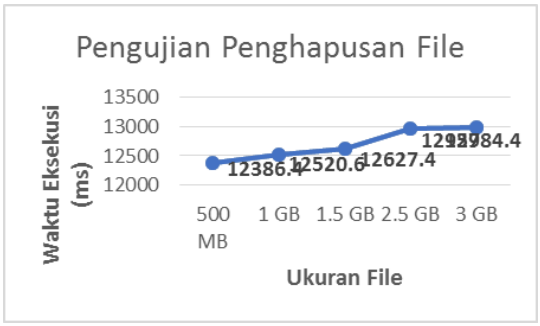
Gambar 11. Grafik Hasil Pengujian *Blocksize*

Dari tabel 8 dan gambar grafik 11 pengujian pengubahan blocksize peniliti menyimpulkan dengan pengubahan *blocksize* dapat mempercepat waktu eksekusi penulisanfile atau data ke HDFS. File yang berukuran 500 MB akan dibagi atau di*split* menjadi beberapa ukuran *block*, jika ukuran masing-masing *block* 192 MB maka akan menghasilkan 3 *block* dimana *block* 1 berukuran 192 MB, *block* 2 berukuran 192 MB, dan *block* 3 berukuran 116 MB. Jika ukuran *block*nya menjadi 384 MB, maka kan menghasilkan 2 *block* dimana *block* 1 berukuran 384 MB dan *block* 2 berukuran 116 MB. Jika jumlah *block* semakin sedikit maka akan mengurangi kerja dari namenode.

D. Pengujian Penghapusan Data

Tabel 9. Rata-Rata Hasil Pengujian Penghapusan Data

Ukuran data/file	Rata-Rata Waktu
500 MB	12.3864
1 GB	12.5206
1.5 GB	12.6274
2.5 GB	12.9570
3 GB	12.9844



Gambar 12. Grafik Hasil Pengujian Penghapusan File

Dari tabel 9 dan gambar grafik 12 menunjukan peningkatan waktu eksekusi penghapusan file pada ukuran yang berbeda. Peneliti dapat menyimpulkan bahwa semakin besar ukuran file, maka akan membutuhkan waktu yang lebih lama untuk melakukan penghapusan filepada HDFS.

E. Pengujian *Availability*

Pada pengujian *availability* data ini peneliti menarik kesimpulan bahwa HDFS tingkat ketersediaan datanya baik, karena telah dilakukan percobaan menonaktifkan salah satu datanode yaitu datanode2. Kerika namenode menginputkan atau menuliskan data ke HDFS, setelah selesai namenode menuliskan data ke HDFS, maka datanode2 diaktifkan kembali. Lalu dicek atau dipastikan apakah datanode2 mendapatkan data yang namenode sudah tuliskan. Dan hasilnya *space* harddisk datanode2 berkurang, yang membuktikan bahwa saat keadaan dinonaktifkan pun datanode2 tetap mendapatkan data percobaan yang tadi dituliskan.

5. KESIMPULAN

5.1 Kesimpulan

a. Langkah yang dilakukan dalam implementasi HDFS sebagai infrastruktur pembangunan *big* data adalah menyiapkan lingkungan kerja HDFS dan *big* data dengan melakukan instalasi Oracle Java JDK, pembuatan *user* hadoop, mendaftarkan IP Address dan Hostname, instalasi dan konfigurasi SSH,melakukan disable IPV6, dan installasi dankonfigurasi Hadoop pada 3 node menggunakan teknik virtualisasi.
b. Faktor- faktor yang menentukan kinerja darirancangan sistem HDFS adalah ukuran file dan ukuran *blocksize*.
c. Kinerja HDFS sebagai infrastruktur pembangunan *big* data menyatakan bahwa semakin besar ukuran dari *blocksize* membuat waktu eksekusinya menjadi lebih cepat.

5.2 Saran

a. Penelitian selanjutnya dapat diimplementasikan pada sistem *real* atau tidakdalam lingkungan virtualisasi
b. Penelitian selanjutnya dapat dikembangkan dengan menambah jumlah datanode lebih dari 2.
c. Penelitian selanjutnya dapat dikembangkan menggunakan aplikasi *benchmarks* hadoop sejenis dengan TestDFSIO.
d. *Size* atau ukuran data yang jauh lebih besar dan lebih banyak dari penelitian ini untuk dijadikan bahan didalam pengujian.

DAFTAR PUSTAKA

[1] D. J. Barrent, et al., "*SSH The Secure Shell: The Devinitive Guide",* United Of America: O'Reilly Media Inch, 2005.

[2] "Data Statistik Tumbuh 50.000 GB per detik," Databooks 3 Maret 2017. (Online) Available: https://databoks.katadata.co.id/datapublish/2017/03/2018-data-statistik-tumbuh-50000-gb-per-detik [diakses 20 Januari 2018]

[3] D. deRoos, "*Hadoop For Dummies,"* John Wiley & Sons, 2014.

[4] gtBlogger, " Mengulas Lengkap Tentang Hadoop: *Software* Pengelolaan Big Data," blog.gamatechno.com, 15 Juni 2017. (Online) Availbale: https://blog.gamatechno.com/software-hadoop- big-data/ [diakses 20 Januari 2018]

[5] *"HDFS Architecture Guide*," TheApache Software Foundation, 08 April 2013. (Online) Available: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html #Data+Blocks [diakses 9 Mei 2018]

[6] Herman Yuliandoko S.T., M., "Jaringan Komputer *Wire* dan *Wireless* BesertaPenerapannya*,"* Sleman: Deepublish, 2018.

[7] *"MapReduce Tutorial*," The Apache Software Foundation, 08 April 2013. (Online) Available: https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html [diakses 8 Mei 2018]

[8] E. Paulson, "*HDFS Benchmark*," 28 Agustus 2013. (Online) Available: http://blog.unit1127.com/blog/2013/08/28/benchmarks/#nnbench [diakses 9 Mei 2018]

[9] W. M. Wijaya, B. M., "Teknologi BIGDATA*,"* Samudera Hindia, 2015.

[10] M.H. Yuliandoko, "Jaringan Komputer *Wire* dan *Wireless* Beserta Penerapannya," Sleman: Deepublish, 2018.