ANALISIS KINERJA HADOOP PADA CLUSTER RASPBERRY PI

Disusun Oleh:

N a m a : La Ode Khairul Nugraha
NIM : 11523188

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
2017

HALAMAN PENGESAHAN DOSEN PEMBIMBING

ANALISIS KINERJA HADOOP PADA CLUSTER RASPBERRY PI

TUGAS AKHIR

Disusun Oleh:

N a m a : La Ode Khairul Nugraha
NIM : 11523188

Yogyakarta, 19 Mei 2017
Pembimbing,

(Mukhammad Andri Setiawan, S.T., M.Sc., PhD)

HALAMAN PENGESAHAN DOSEN PENGUJI

ANALISIS KINERJA HADOOP PADA CLUSTER RASPBERRY PI

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik Informatika
di Fakultas Teknologi Industri Universitas Islam Indonesia
Yogyakarta, 5 Juni 2017

Tim Penguji

Mukhammad Andri Setiawan, S.T., M.Sc., PhD. _____

Anggota 1

Andhik Budi Cahyono, ST., MT. _____

Anggota 2

Hamid, S.T., M.Eng. _____

Mengetahui,
Ketua Jurusan Teknik Informatika
Fakultas Teknologi Industri
Universitas Islam Indonesia

(Hendrik, S.T., M.Eng.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : La Ode Khairul Nugraha

NIM : 11523188

Tugas akhir dengan judul:

ANALISIS KINERJA HADOOP PADA CLUSTER RASPBERRY PI

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila dikemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung resiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 22 Mei 2017

(La Ode Khairul Nugraha)

HALAMAN PERSEMBAHAN

Karya ini saya persembahkan kepada:
Allah SWT yang telah melimpahkan segala Rahmat dan Hidayah-Nya,
Kedua orang tua yang saya cintai, La Ode Ridwan dan Paramita Lestarini
Teman-teman serta saudara-saudara saya.

HALAMAN MOTO

“Being pessimistic is not so bad. You won’t get disappointed when something goes wrong.”

“Life is like a game. Free to play, yet pay to win”

“Experts are once a beginner”

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Puji syukur saya panjatkan atas kehadirat Allah SWT, Tuhan yang telah memberikan kemudahan melalui rahmat dan hidayah-Nya sehingga atas ridho-Nya Tugas Akhir yang dengan judul "Analisis Kinerja Hadoop pada Cluster Raspberry Pi" dapat diselesaikan dengan baik.

Tugas Akhir ini disusun sebagai syarat terakhir yang harus di tempuh untuk menyelesaikan pendidikan pada jenjang Strata Satu (S1) pada Jurusan Teknik Informatika Universitas Islam Indonesia. Dalam penyelesaian laporan tugas akhir ini, penulis tidak luput dari bantuan, semangat, serta motivasi dari berbagai pihak, untuk itu penulis tidak lupa menyampaikan ucapan terimakasih yang kepada:

1. Bapak Nandang Sutrisno, SH., M.Hum., LLM., Ph.D. selaku Rektor Universitas Islam Indonesia.
2. Bapak Imam Djati Widodo, Dr., M.Eng.Sc. selaku Dekan Fakultas Teknologi Industri Universitas Islam Indonesia.
3. Bapak Hendrik, S.T., M.Eng., selaku Kepala Jurusan Teknik Informatika.
4. Bapak Mukhammad Andri Setiawan, S.T., M.Sc., PhD, selaku dosen pembimbing tugas akhir ini yang telah membimbing pelaksanaan tugas akhir secara keseluruhan.
5. Bapak Syarif Hidayat, S.Kom., M.I.T, yang telah memberi ide untuk penelitian tugas akhir ini.
6. Bapak dan ibu dosen Jurusan Teknik Informatika yang telah memberikan ilmunya kepada penulis selama menempuh pendidikan di Universitas Islam Indonesia, semoga ilmu yang telah diajarkan dapat bermanfaat kemudian hari.

7. Orang tua yang saya cintai, La Ode Ridwan dan Paramita Lestarini atas segala dukungan serta doa yang tidak pernah putus, nasihat dan petunjuk dari mereka yang tiada henti yang memotivasi hingga detik ini.
8. Teman – teman di Universitas Islam Indonesia atas bantuan dan dukungannya.
9. Rekan – rekan informatika angkatan 2011 "DEFINE" atas dukungan dan bantuannya dari awal perkuliahan hingga selesainya tugas akhir ini.
10. Kawan – kawan "GWS" yang selalu saling mendukung dan menghibur di kala suntuk.
11. Laboratorium Informatika yang telah meminjamkan alat untuk keperluan Tugas Akhir ini.
12. Serta pihak – pihak yang penulis tidak bisa sebutkan satu persatu sehingga penulis bisa menyelesaikan tugas akhir ini.

Penulis menyadari bahwa laporan yang dibuat ini masih memiliki banyak kesalahan dan kekurangan. Untuk itu penulis mengharapkan kritik serta saran yang membangun dari semua pihak agar dapat tercapai kesempurnaan laporan ini. Penulis juga berharap laporan ini dapat bermanfaat bagi para semua orang, temasuk penulis.

Wassalamu'alaikum Wr. Wb.

Yogyakarta, 22 Mei 2017

(La Ode Khairul Nugraha)

SARI

Apache Hadoop merupakan sebuah framework yang digunakan untuk pemrosessan *Big Data* yang dirancang untuk dijalankan di sebuah *cluster* yang terdiri dari beberapa komputer. Hadoop mengimplementasikan model pola komputasi yang disebut MapReduce, di mana aplikasi dibagi menjadi banyak fragmen yang dapat dijalankan dalam *node* pada sebuah *cluster*. Selain itu Hadoop menyediakan media penyimpanan data dengan sebuah file sistem terdistribusi yang disebut Hadoop Distributed File System (HDFS) yang menyimpan data di tiap *node* pada *cluster*. MapReduce dan HDFS dirancang agar kerusakan pada *node* ditangani secara otomatis oleh framework. Hadoop dapat digunakan di banyak sistem, termasuk *cluster* Raspberry Pi. Penelitian ini bertujuan untuk mengetahui bagaimana kemampuan *cluster* Raspberry Pi dalam menjalankan Hadoop dan membandingkannya dengan komputer dengan harga setara.

Analisis dilakukan dengan menjalankan program MapReduce di *cluster* Raspberry Pi yang dibangun kemudian mencatat waktu eksekusi program. Semakin cepat waktu eksekusi semakin baik performa Hadoop. Hal yang sama dilakukan pada laptop yang menjadi perbandingan.

Dari hasil analisis diketahui bahwa *cluster* Raspberry Pi dengan 5 *node* memiliki performa yang hampir sama dengan laptop yang menjadi perbandingan, bahkan mengungguli laptop saat memproses data berukuran besar. Dengan menggunkan fungsi power dapat diprediksi bahwa *cluster* Raspberry Pi dengan 6 *node* sudah cukup untuk dapat mengungguli performa komputer dengan harga setara .

Kata kunci: Hadoop, *Big Data*, MapReduce, HDFS, *cluster*, Raspberry Pi.

GLOSARIUM

Balancing	penyeimbangan jumlah blok pada cluster.
Bandwidth	jumlah data yang dapat ditransmisikan dalam waktu yang ditentukan.
Checkpoint	menyimpan status sistem pada waktu tertentu
Checksum	verifikasi keaslian atau keutuhan data
Daemon	program yang berjalan di balik antarmuka, tanpa interaksi langsung dengan pengguna.
Failover	memindahkan pekerjaan ke sistem cadangan saat terjadi kerusakan.
Framework	kerangka kerja.
Master node	node induk
Middleware	software yang berfungsi sebagai penghubung antar sistem operasi .
Network Switch	perangkat penghubung jaringan komputer.
Node	sebuah titik berupa perangkat dalam jaringan.
Open source	sumber terbuka.
Runtime	waktu kerja.
Script eksekusi	fungsi yang ditulis untuk mempermudah dan mengotomasi kerja.
Single-board	papan sirkuit tunggal.
Slave node	node pekerja
Snapshot	keadaan sistem pada suatu waktu tertentu
Spacer	pemisah
Throughput	rata-rata transmisi yang berhasil diakukan dalam jaringan.
Web crawler	program yang secara otomatis menjelajahi internet dan mengumpulkan data.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Apache Hadoop merupakan sebuah framework yang memungkinkan pemrosesan data terdistribusi antara rangkaian komputer menggunakan model programming sederhana. Hadoop dirancang untuk dijalankan di *cluster* yang terdiri dari beberapa komputer, di mana tiap komputer mengerjakan komputasi dan penyimpanan secara lokal ("Welcome to Apache Hadoop!," 2017). Hadoop mengimplementasikan model pola komputasi yang disebut Map/Reduce, di mana aplikasi dibagi menjadi banyak fragmen yang dapat dijalankan di tiap *node* pada sebuah *cluster*. Selain itu Hadoop menyediakan sebuah file sistem terdistribusi yang disebut Hadoop Distributed File System (HDFS) yang menyimpan data di tiap *node* pada *cluster*. Map/Reduce dan HDFS dirancang agar kegagalan pada *node* ditangani secara otomatis oleh framework (Agarwal, 2016).

Komputer *cluster* adalah sebuah sistem dengan pemrosesan terdistribusi, yang terdiri dari sekumpulan komputer yang terhubung dalam jaringan yang berkerja sebagai satu komputer terintegrasi. *Cluster* dibangun untuk untuk mendapatkan sistem dengan performa tinggi dan kemampuan skalabilitas, sehingga memungkinkan pengguna dapat terus meningkatkan kemampuan pemrosesannya (Buyya, 1999).

Raspberry Pi adalah sebuah komputer single-board seukuran kartu kredit yang dikembangkan oleh Raspberry Pi Foundation di UK ("Raspberry Pi FAQs - Frequently Asked Questions," n.d.). Raspberry Pi dapat digunakan layaknya komputer pada umumnya dengan sehingga dapat digunakan untuk membangun sebuah *cluster* kecil yang dapat dimanfaatkan sebagai alat eksperimentasi komputer (Richardson et al., 2012).

Hadoop dapat digunakan di banyak sistem, termasuk *cluster* Raspberry Pi. Kemampuan skalabilitas sebuah *cluster* memungkinkan peningkatan performa pada Hadoop. Hanya saja ukuran data yang dapat diolah dengan Hadoop pada

sebuah *cluster* Raspberry Pi sangat terbatas jika dibandingkan dengan *cluster* pada umumnya karena keterbatasan performa Raspberry Pi.

Maka dari itu diperlukan analisis kinerja Hadoop dalam memproses data pada *cluster* Raspberry Pi untuk mengetahui seberapa batas kemampuan maksimal Hadoop dan ukuran data yang dapat diolah pada *cluster* dengan sumber daya terbatas.

1.2 Rumusan Masalah

a. Bagaimana kinerja Hadoop dalam mengolah data pada *cluster* Raspberry Pi.
b. Berapa ukuran data yang dapat disimpan dan diolah dengan Hadoop pada *cluster* Raspberry Pi.
c. Bagaimana perbandingan kinerja *cluster* dengan sistem dengan harga setara.

1.3 Batasan Masalah

Agar penelitian yang dikerjakan dapat lebih terarah dan sesuai dengan rancangan, maka ditetapkan batasan-batasan dalam penelitian ini. Berikut adalah batasan masalah pada penelitian ini :

a. *Cluster* dibangun menggunakan 5 buah Raspberry Pi 3 Model B.
b. Antarmuka sistem menggunakan teks pada konsol.
c. Uji kinerja menggunakan beberapa dataset dengan ukuran minimal 400 MB.
d. Dataset diambil dari sumber publik (Google Books).
e. Kebutuhan *hardware* dipinjam dari laboratorium Informatika.

1.4 Tujuan Penelitian

Penelitian ini memiliki beberapa tujuan yang ingin dicapai, antara lain:

a. Membangun sebuah *cluster* untuk menguji kinerja Hadoop pada *cluster* Raspberry Pi dan ketahanannya dalam memproses data berukuran besar.
b. Membandingkan kinerja Hadoop pada *cluster* Raspberry Pi dengan komputer komersial dengan biaya setara.
c. Mengembangkan kemampuan penulis dalam merancang dan membangun sebuah proyek Raspberry Pi.

d. Implementasi ilmu yang didapatkan oleh penulis selama menempuh pendidikan di Teknik Informatika Universitas Islam Indonesia.

1.5 Manfaat Penelitian

Penelitian yang akan dilakukan dapat memberi manfaat bagi masyarakat umum, diantaranya:

a. Mengetahui bagaimana perbandingan kinerja Hadoop pada *cluster* Raspberry Pi dan komputer dengan harga setara
b. Megetahui kemampuan *cluster* Raspberry Pi dalam mengolah data berukuran besar.
c. Mengetahui seberapa besar ukuran data yang dapat diolah Hadoop pada *cluster* Raspberry Pi.
d. Memberi pengetahuan mengenai Hadoop dan komputer *cluster* .
e. Sebagai referensi untuk membangun proyek Raspberry Pi.

1.6 Review Penelitian Sejenis

Penelitian tentang *cluster* Raspberry Pi telah beberapa kali dilakukan. Salah satu *cluster* Raspberry Pi yang paling awal diantaranya adalah Iridis Pi (Cox et al. 2014) yang dirancang untuk mempelajari *cluster* menggunakan 64 buah Raspberry Pi model B, dan Glasgow Cloud Pi (Tso et al. 2013) yang dibangun dengan 54 buah Raspberry Pi model B untuk memahami virtualisasi dan teknologi *cloud computing*.

Penelitian tentang Hadoop pada *cluster* Raspberry Pi juga sudah pernah dilakukan. Penelitian sebelumnya (Rusyadi et.al, 2016) membahas tentang perbandingan kinerja Hadoop dalam menjalankan prosess Wordcount pada Raspberry Pi model B dan Raspberry Pi 2 model B yang dirakit menjadi *cluster* 3 *node*. 4 jenis percobaan dilakukan dimana setiap percobaan dikondisikan berbeda seperti jumlah *node* yang digunakan, jenis memori yang digunakan, peran Raspberry Pi sebagai *node*, dan kondisi meningkatkan clock CPU Raspberry Pi. Analisa yang dilakukan meliputi lama eksekusi proses Wordcount dan kondisi rata-rata CPU pada saat menjalankan Wordcount. Dari hasil beberapa kali

percobaan yang dilakukan dengan kondisi yang berbeda ditemukan bahwa Raspberry Pi dapat mampu menjalankan proses Wordcount Hadoop dengan baik dan dapat dijadikan sebagai media untuk mempelajari Hadoop *cluster*, penambahan memori dan *overclock* dapat meningkatkan performa Raspberry dalam proses Wordcount.

Penelitian lain yaitu Micro Data Center berbasis Raspberry Pi (N.J. Schot, 2015). Tujuannya adalah merancang alternatif untuk rak server standar untuk aplikasi *Big Data* yang murah, dengan konsumsi daya rendah serta mampu bekerja bersamaan dengan suhu stabil. *cluster* Raspberry pi 2 Model B dengan 8 *node* digunakan untuk merancang setup Hadoop serta melakukan *benchmark* performa dan konsumsi daya listrik. Rancangan rak *cluster* Raspberry Pi kemudian dibuat untuk memberi alternatif untuk rak server tradisional menggunakan rak ukuran 1U dengan perkiraan jumlah unit Raspberry Pi 2 sejumlah 72 hingga 80 buah dalam rak 1U dengan sebuah suplai daya. Hasil penelitian Schot menghasilkan rancangan rak server yang murah dengan konsumsi daya rendah tanpa perlu sistem pendingin tambahan, walaupun dengan kinerja di bawah server standar.

Penelitian lain mengenai Performa Raspberry Pi Cloud Berdaya Rendah untuk *Big Data* (Hajji and Tso, 2016) bertujuan mengevaluasi performa sistem di bawah beban kerja nyata, serta menguji sejumlah beban kerja httperf pada *node* tunggal serta Apache Spark. Selain itu dilakukan uji performa dengan dan tanpa virtualisasi. *Cluster* dibangun dengan 12 buah Raspberry Pi 2 Model B. Dalam tiap *node* dipasang *Docker* untuk membangun container virtualisasi. Sehingga dalam *cluster* terdapat *virtual machine* dengan spesifikasi yang sama dengan *cluster* asli. Uji performa dilakkukan dengan file berukuran 1GB, 4GB, hingga 6GB yang mewakili ukuran beban kerja kecil, sedang dan besar. Properti yang dinilai diantaranya waktu yang dibutuhkan untuk menjalankan beban kerja, kecepatan transmisi antar tiap *node*, penggunaan CPU, serta konsumsi daya listrik. Hasil penelitian menunjukkan bahwa dengan virtualisasi terdapat peningkatan konsumsi memori dan CPU, penurunan *throughput* jaringan, selain itu diketahi

bahwa peningkatan konsumsi daya Raspberry Pi dipengaruhi oleh adanya vistualisasi.

Dari pengamatan penelitian dan aplikasi sejenis, diketahui bahwa Raspberry Pi dapat mengolah data berukuran besar dengan baik. Perbedaan penelitian ini dengan penelitian sebelumnya yaitu penulis ingin mengetahui bagaimana performa Hadoop pada *cluster* Raspberry yang dibangun dengan sumber daya terbatas serta seberapa daya tahan *cluster* tersebut di bawah beban kerja yang besar dan waktu eksekusinya, dan membandingkannya dengan komputer dengan harga setara. Selain itu penulis ingin mengatahui seberapa besar ukuran data yang dapat disimpan dan diproses oleh *cluster* yang dibangun.. Disamping itu penelitian ini menggunakan *cluster* dengan Raspberry Pi 3 Model B yang belum dilakukan oleh penelitian sebelumnya, sehingga diharapkan adanya peningkatan performa yang cukup signifikan.

1.7 Metode Penelitian

Penelitian yang dilakukan penulis akan melalui suatu aturan perancangan yang berurutan serta memenuhi beberapa tahapan yaitu:

Metode Pengumpulan Data

Pada laporan Tugas Akhir ini, pengumpulan data yang digunakan adalah sebagai berikut:

a. Metode studi pustaka

   Pengumpulan data yang diperlukan dengan melakukan kajian literatur dan pustaka dari buku-buku dan jurnal. Data yang dibutuhkan berupa data yang berkaitan dengan penjelasan mengenai *Big Data*, Apache Hadoop, *cluster* dan Raspberry Pi.

b. Referensi internet

   Mengumpulkan informasi dan materi yang ada di internet dengan mengunjungi website yang berkaitan dengan penelitian, diantaranya mengenai Hadoop dan konfigurasinya, *cluster*, serta tutorial untuk proyek Raspberry Pi.

Pengembangan Sistem

Pada penelitian ini, untuk mencapai hasil yang baik dalam membangun sebuah *cluster*, maka harus melewati tahapan-tahapan sebagai berikut:

a. Analisis

   Tahapan analisis mengenai hal-hal apa saja yang dibutuhkan pada sistem yang akan dibangun.tahapan ini menghasilkan informasi kebutuhan sistem (*system requirements*) untuk mengetahui kebutuhan sistem.

b. Perencanaan dan Perancangan

   Tahap ini merupakan tahapan di mana penulis membangun rancangan sistem *cluster* yang akan digunakan serta mengumpulkan kebutuhan *hardware* dan *software* untuk *cluster*.

c. Instalasi dan Konfigurasi

   Tahap ini adalah proses membangun sistem *cluster* yang telah dirancang serta instalasi dan pembaruan Hadoop dan aplikasi yang dibutuhkan ke Raspberry Pi yang digunakan pada *cluster*.

d. Pengujian

   Tahap uji kinerja Hadoop pada *cluster* yang dibangun untuk memastikan Hadoop dapat bekerja dengan baik pada *cluster*.

1.8 Sistematika Penulisan

Untuk memberikan gambaran lengkap mengenai masalah yang akan dibahas dalam laporan ini, maka sistematika laporan akan dibagi menjadi lima bab, dengan rincian sebagai berikut:

BAB I PENDAHULUAN

Bagian ini membahas latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, metodologi penelitian dan sistematika penulisan yang dapat memberikan gambaran mengenai penelitian yang dilakukan.

BAB II LANDASAN TEORI

Merupakan penjelasan mengenai teori yang berhubungan dengan penelitian, meliputi Hadoop, MapReduce, Raspberry Pi dan komputer *cluster*.

BAB III METODOLOGI PENELITIAN

Bagian ini memuat uraian tentang analisis sistem, kebutuhan *hardware* dan *software*, rancangan sistem, fungsi-fungsi dan konfigurasi yang dibutuhkan, serta skenario percobaan yang akan dilakukan.

BAB IV HASIL DAN PEMBAHASAN

Menggambarkan bagaimana implementasi sistem *cluster* dan analisis sistem yang telah dibangun. Bagian ini memuat pembahasan mengenai analisis uji kinerja Hadoop pada *cluster* yang dibuat. Bab ini juga membahas kelebihan dan kekurangan sistem yang dibangun.

BAB V PENUTUP

Bagian ini memuat kesimpulan-kesimpulan yang merupakan rangkuman dari analisis kinerja ada bagian sebelumnya. Bagian ini juga berisi saran-saran yang perlu diperhatikan berdasar keterbatasan yang ditemukan selama penelitian dilakukan.

BAB II
LANDASAN TEORI

2.1 *Big Data*

Big Data adalah istilah untuk kumpulan data yang sangat besar dengan struktur yang lebih besar, bervariasi dan kompleks, dengan kesulitan dalam hal penyimpanan, analisa, dan visualisasi untuk diproses lebih lanjut (Sagiroglu & Sinanc, 2013). Menurut deRoos et al. (2014), suatu data dapat disebut *Big Data* jika memenuhi tiga karakteristik tertentu yang umumnya disebut 3V (*volume, variety, & velocity*):

a. *Volume* (Volume): *Big Data* memiliki volume yang sangat besar jika dibandingkan dengan data pada umumnya. Ukuran *Big Data* dapat mencapai ukuran terabyte hingga petabyte.
b. *Variety* (Keragaman): Data memiliki keberagaman dalam strukturnya. yang terdiri mulai dari data teks mentah yang tidak terstruktur, data semi terstruktur seperti file xml, hingga data terstruktur yang memiliki baris dan kolom atau bahkan kombinasi dari ketiganya.
c. *Velocity* (Kecepatan): Dalam konteks ini seberapa cepat data yang dihasilkan dapat diproses secara *real time*. Banyaknya jumlah data yang masuk ke suatu perusahaan membuat data memiliki sedikit waktu untuk diproses sebelum disimpan.

Pada umumnya jika suatu data memiliki salah satu dari tiga karakteristik tersebut maka data tersebut layak disebut *Big Data*.

Big Data memiliki kompleksitas tinggi dan ukuran yang sangat besar sehingga model *database* biasa tidak cukup untuk memprosesnya. Diperlukan metode tertentu untuk memproses *Big Data*. Hadoop dengan HDFS dan MapReduce memberikan solusi untuk mengelola *Big Data*.

2.2 Apache Hadoop

2.2.1 Pengertian Hadoop

Apache Hadoop adalah sebuah *framework* yang dirancang untuk menjalankan aplikasi pemrosesan *Big Data* pada *cluster* besar yang dibangun dengan *commodity hardware* (*hardware* komputer standar yang mudah didapat dengan harga terjangkau (deRoos et al., 2014)). Hadoop menyediakan sebuah file sistem terdistribusi yang disebut *Hadoop Distributed File System* (HDFS) yang menyimpan data di tiap *node* pada *cluster*. Selain itu Hadoop mengimplementasikan model pola komputasi yang disebut MapReduce, di mana aplikasi dibagi menjadi banyak fragmen di mana tiap fragmen dapat diproses di tiap *node* pada sebuah *cluster*. *Framework* Hadoop memberi ketersediaan dan pergerakan data dalam aplikasi serta *bandwidth* tinggi pada *cluster*. MapReduce dan HDFS dirancang agar kerusakan pada *node* ditangani secara otomatis oleh *framework* (Agarwal, 2016).

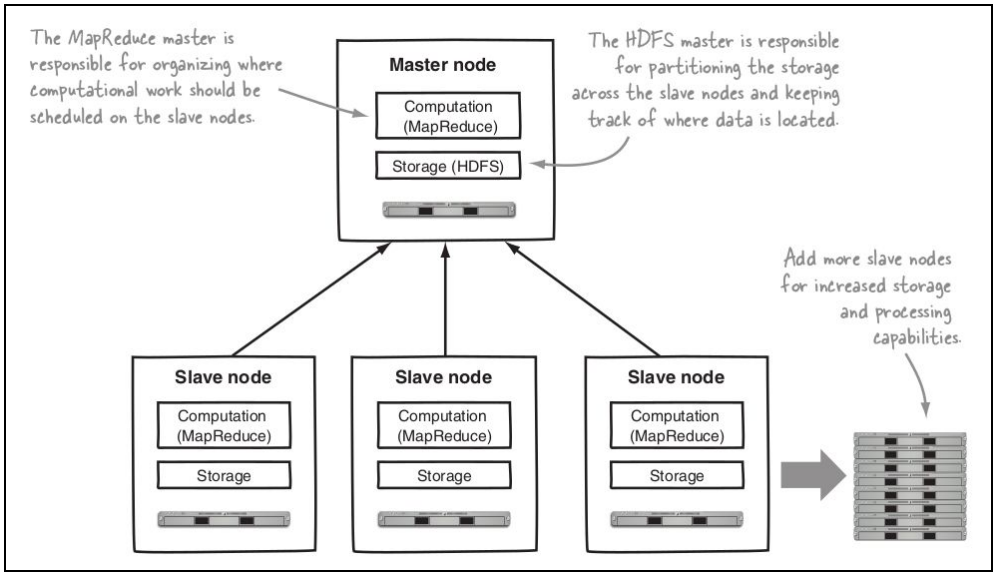
Sejarah Hadoop

Hadoop awalnya dibangun untuk infrastruktur proyek Nutch, *open source web crawler* yang dikembangakan Apache, untuk mengatasi masalah skalibilitas yang terdapat pada Nutch. Pada waktu itu Google menerbitkan paper tentang file system yang digunakannya, yang disebut *Google File System* (GFS) serta paper Map-Reduce, *framework* komputasi untuk pemrosesan paralel. Implementasi kedua paper tersebut pada Nutch membuat proyek Nutch terbagi menjadi dua proyek terpisah. Doug Cutting, yang saat itu bekerja di Yahoo! menamai proyek ini Hadoop, yang diambil dari nama mainan anaknya (Holmes, 2012).

Arsitektur Hadoop

Hadoop memiliki arsitektur *master/slave* terdistribusi yang terdiri dari Hadoop Distributed File System (HDFS) untuk penyimpanan dan MapReduce untuk kemampuan komputasi. Ciri khas dari Hadoop adalah pembagian data dan komputasi paralel pada *dataset* besar. Kemampuan penyimpanan dan komputasi

Hadoop dapat ditingkatkan dengan menambah *host* pada *cluster* Hadoop dan dapat mencapai ukuran data hingga petabyte dalam *cluster* dengan ribuan *host* (Holmes, 2012). Gambar 2.1 menujukkan arsitektur Hadoop.



Gambar 2.1. Arsitektur Hadoop

Sumber: Holmes (2012)

2.2.2 Hadoop Distributed File System

Hadoop Distributed File System (HDFS) adalah komponen penyimpanan dalam Hadoop. HDFS mengikuti model Google File System (GFS). HDFS dioptimalkan untuk *throughput* tinggi dan bekerja optimal dengan ukuran data yang besar dengan ukuran gigabyte dan yang lebih besar. Untuk mendukung throughput yang tinggi tersebut HDFS menggunakan ukuran blok file yang lebih besar dari *file system* pada umumnya serta optimasi data secara lokal untuk mengurangi beban *input/output* pada jaringan.

Skalabilitas dan ketersediaan juga merupakan ciri HDFS, yang dicapai dengan penggandaan data dan toleransi kerusakan. Dengan penggandaan data HDFS dapat mentolerir kerusakan hardware maupun software, serta menggandakan blok data pada *node* yang mati secara otomatis (Holmes, 2012).

HDFS dirancang untuk menyimpan ukuran data yang sangat besar di perangkat dalam *cluster* besar. HDFS membagi tiap file menjadi rangkaian blok-

blok data yang kemudian disimpan pada tiap *node* di *cluster*. Semua blok dalam file berukuran sama kecuali blok terakhir. Blok-blok file biasanya digandakan di tiap *node* untuk toleransi kerusakan. Ukuran blok *default* pada HDFS umumnya sebesar 64 hingga 128 MB. Ukuran blok dan faktor penggandaan dapat ditentukan oleh pengguna ("ProjectDescription - Hadoop Wiki," 2014).

Arsitektur HDFS

HDFS menggunakan arsitektur *master/slave*. Instalasi HDFS terdiri dari sebuah NameNode, server induk yang menangani namespace dalam *file system* dan mengatur akses ke file oleh klien. selain itu terdapat sejumlah DataNode yang berfungsi sebagai *slave* yang terdapat pada tiap *node* pada *cluster* dan menangani penyimpanan pada *node* tempatnya bekerja. NameNode menjalankan operasi seperti membuka, menutup, serta mengubah nama file dan direktori, serta menentukan pemetaan blok ke DataNode. DataNode berfungsi melayani permintaan *read* dan *write* dari klien *file system*. Selain itu DataNode menjalankan pembuatan, penghapusan, dan penggandaan blok dengan instruksi dari NameNode ("ProjectDescription - Hadoop Wiki," 2014).

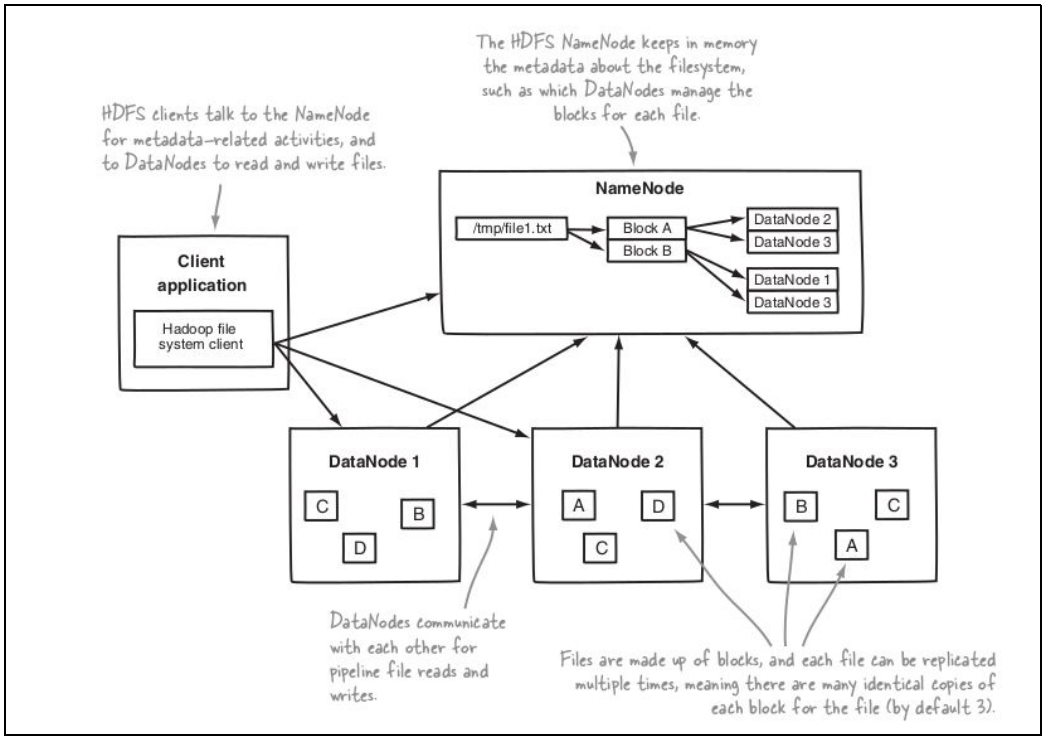
NameNode

NameNode adalah sebuah titik kerusakan tunggal dalam HDFS. Pada Hadoop, data dipecah menjadi blok-blok untuk pemrosesan. Informasi mengenai blok-blok tersebut serta cabang-cabang direktori data disimpan di NameNode. Selain NameNode juga terdapat NameNode sekunder. NameNode sekunder menyimpan *snapshot* dan *checkpoint* saat melakukan kontak dengan NameNode. *Snapshot* dan *checkpoint* ini digunakan untuk memulihkan *file system* jika terjadi kerusakan.

DataNode

DataNode digunakan untuk menyimpan blok-blok data. Jika diminta, DataNode juga dapat mengambilkan data. NameNode mengawasi seluruh blok

data yang disimpan di DataNode. DataNode memperbarui NameNode secara berkala agar tetap sinkron dengan operasi yang berjalan (Shenoy, 2014). Arsitektur HDFS serta skema NameNode dan DataNode dapat dilihat pada gambar 2.2.



Gambar 2.2. Arsitektur HDFS

Sumber: Holmes (2012)

2.2.3 MapReduce

MapReduce adalah model pemrograman yang menjalankan sebuah komputasi terdistribusi untuk pemrosesan *dataset* berukuran besar. MapReduce dirancang untuk dijalankan pada sebuah *cluster*. Komputasi MapReduce memliki dua tahap inti, yaitu *mapping* dan *reducing*, di mana pengguna menentukan fungsi dari proses *map* dan *reduce*. *Input* yang digunakan untuk komputasi tersebut adalah *dataset* berupa pasangan *key/value* (Dean & Ghemawat, 2004).

Dalam MapReduce, tidak terdapat kolom dan baris seperti pada *database*. Sebagai gantinya, bentuk data berupa pasangan *key/value*. *Key* (Kunci) adalah

informasi yang dicari oleh pengguna, sedangkan *value* (nilai) adalah data yang berhubungan dengan *key* tersebut (Shenoy, 2014).

Contoh pasangan *key/value* pada WordCount, program untuk menghitung jumlah kata yang muncul dalam sebuah file teks:

(kata, 1)

(kata, 3)

(kata, 2)

Key (kunci) Dalam kasus tersebut adalah kata. *Value* (nilai) mewakili data berupa jumlah kata yang muncul dalam teks ("WordCount - Hadoop Wiki," 2011). Proses MapReduce pada program WordCount dapat dilihat pada Gambar 2.3. Proses kerja MapReduce terdiri dari *Map*, *Sort and Shuffling,* serta *Reduce*.

Map

Dalam tahap *mapping*, *framework* membagi *dataset* masukan menjadi banyak fragmen kemudian mengirim tiap fragmen ke *map task*. Tiap *map task* membaca *input* berupa pasangan *key/value* (K,V) dari fragmen yang ditugaskan kemudian mengubah *input* tersebut dan mengeluarkan pasangan *key/value intermediate* (K', V'). Dalam deskripsi teknis Hadoop pasangan (K,V) juga dapat disebut *tuple*.

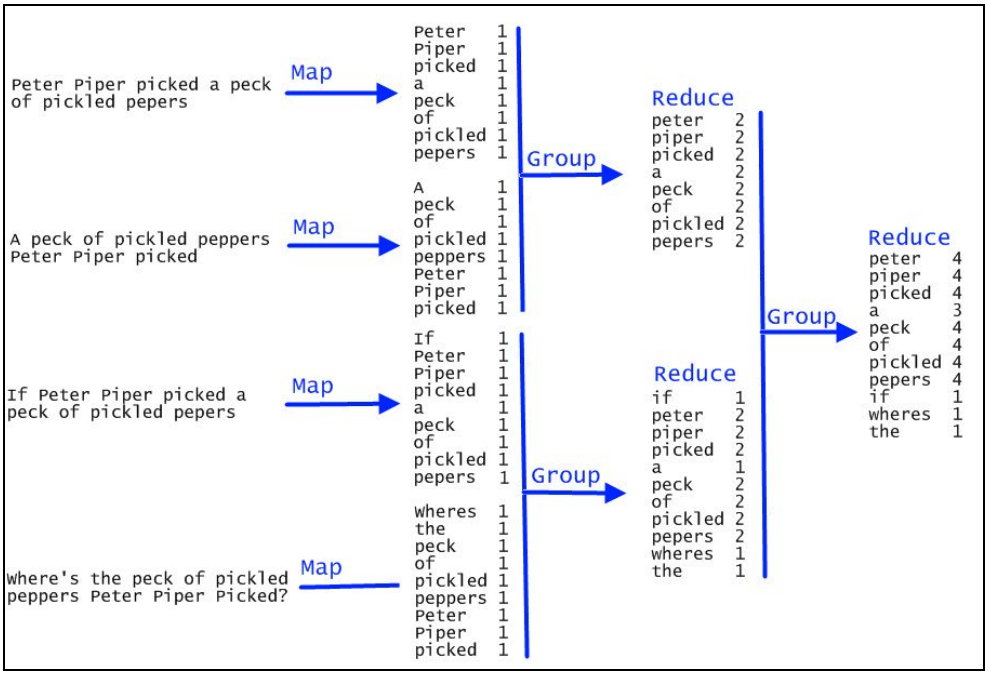
Sort and Shuffling

Setelah tahap *map*, *framework* kemudian mengelompokkan *dataset intermediate* berdasarkan kuncinya dan menghasilkan *tuple* berisi data (K', V'*) agar semua *value* dengan *key* yang sama muncul sebagai satu pasangan *key/value*. *Framework* juga membagi kumpulan *tuple* menjadi sejumlah fragmen sesuai dengan jumlah *reduce task* .

Reduce

Pada tahap *reduce*, tiap unit *reduce task* memakai fragmen dari *tuple* (K', V'*) dari hasil *Sort and Shuffling*. Pada tiap *tuple* unit tersebut memanggil fungsi

bentukan pengguna yang menggabungkan *tuple* dan mengubahnya menjadi *output* berupa pasangan *key/value* (K,V).



Gambar 2.3. Diagram proses MapReduce

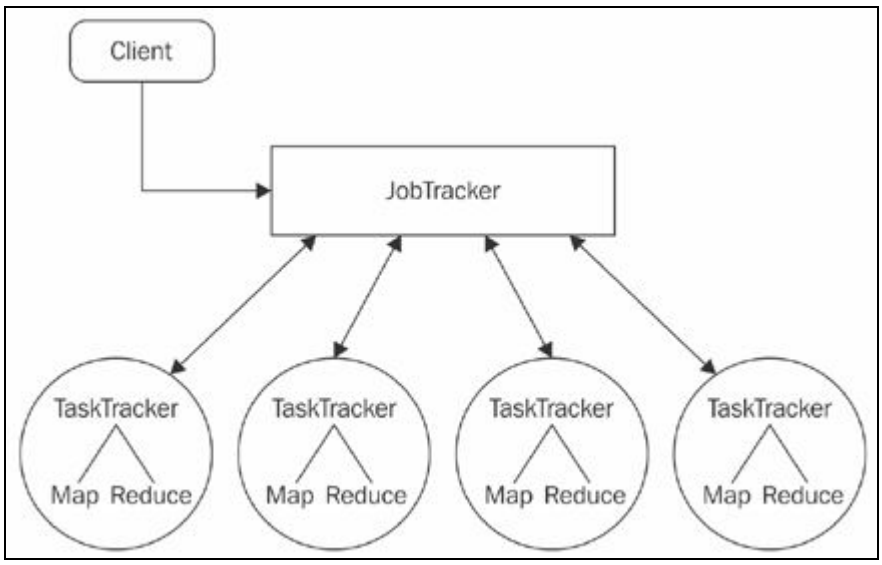
Sumber: Wjaya (2013)

Fungsi pada tiap tahap dijalankan dengan toleransi kerusakan, di mana jika salah satu *node* gagal dalam menjalankan komputasi, tugas yang dikerjakan dibagikan ke *node* lain yang masih aktif. Banyaknya *map task* dan *reduce task* dapat menyeimbangkan beban kerja dan memperkecil *runtime* tambahan ("ProjectDescription - Hadoop Wiki," 2014).

Arsitektur MapReduce

Hadoop MapReduce menggunakan arsitektur *master/slave*. Dengan sebuah server sebagai *master* yang disebut JobTracker dan sejumlah server sebagai *slave* yang disebut TaskTracker, yang terdapat pada tiap *node* pada *cluster*. JobTracker dan TaskTracker adalah komponen pada Hadoop yang bertugas mengelola *job*

pada MapReduce. Pengguna memasukan *job* MapReduce ke JobTracker, yang kemudian diletakkan pada antrian kerja dan dijalankan sesuai urutan masuknya. JobTracker menangani penugasan fungsi map dan reduce ke TaskTracker. TaskTracker menjalankan fungsi dengan instruksi dari JobTracker serta menangani perpindahan data antara tahap *map* dan *reduce* ("ProjectDescription - Hadoop Wiki," 2014). Gambar 2.4 menunjukkan skema JobTracker dan TaskTracker.



Gambar 2.4. Skema JobTracker dan TaskTracker pada MapReduce

Sumber: Shenoy (2014)

JobTracker dan TaskTracker

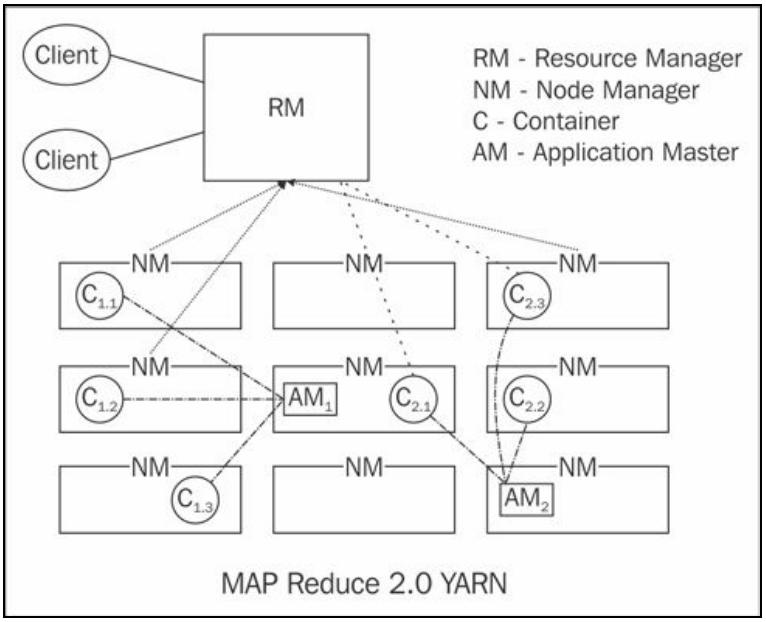
Sebuah JobTracker menjalankan fitur MapReduce di tiap *node* dalam *cluster*. JobTracker mengontak NameNode untuk mendapat informasi mengenai lokasi data. Setelah lokasi ditemukan, JobTracker mengontak TaskTracker yang dekat dengan DataNode tempat data tertentu disimpan. TaskTracker mengirim sinyal ke JobTracker untuk melaporkan keaktifannya. Jika TaskTracker mati pada suatu *node*, sinyalnya akan berhenti sehingga JobTracker akan menganggap *node* tersebut sudah tidak berfungsi, kemudian memindahkan tugasnya ke TaskTracker lain. JobTracker memperbarui informasi setelah *job* selesai, sehingga klien yang

mengontak JobTracker dapat mengetahui ketersediaan *node* yang akan digunakan (Shenoy 2014).

MapReduce diperbarui pada Hadoop versi 2.0. Fitur manajemen sumber daya dipisahkan dari *framework* MapReduce. Manajemen sumber daya ini disebut YARN.

2.2.4 YARN

YARN (Yet Another Resource Negotiator) merupakan manajemen sumber daya multi guna yang dapat menjadwalkan dan mengatur siklus CPU dari *cluster* Hadoop ke aplikasi yang akan dijalankan. YARN merupakan peningkatan dari MapReduce yang dikenalkan pada Hadoop versi 2.0 yang dikenal juga sebagai MapReduce 2.0. Dalam Hadoop YARN dideskrisipsikan sebagai alat yang memungkinkan *framework* pemrosesan data lain dijalankan di Hadoop, sehingga Hadoop tidak hanya dapat menjalankan MapReduce. (deRoos et al., 2014). Struktur YARN dapat dilihat pada gambar 2.5.



Gambar 2.5. Struktur YARN

Sumber: Shenoy (2014)

Resource Manager

Komponen inti dari YARN adalah *Resource Manager*, yang mengelola semua pemrosesan data dalam *cluster* Hadoop. *Resource Manager* mengelola, mengatur dan memberi sumber daya kepada aplikasi yang membutuhkan.

Node Manager

Tiap *slave node* terdapat *daemon Node Manager*, yang bekerja sebagai *slave* dari *Resource Manager*. Tiap *Node Manage*r melacak sumber daya pemrosesan data di *slave node* tersebut kemudian mengirim laporan reguler kepada *Resource Manager.* Sumber daya pemrosesan pada *cluster* Hadoop digunakan dalam bagian-bagian yang disebut *container*.

Container

Container adalah kumpulan dari sumber daya yang dibutuhkan untuk menjalankan sebuah aplikasi yang terdiri dari CPU *core*, memori, *bandwidth* jaringan dan ruang disk.

Sebuah *container* berjalan sebagai proses tersendiri pada sebuah *node* di *cluster* Hadoop.

Application Master

Tiap aplikasi yang berjalan di *cluster* Hadoop memiliki *Application Master* sendiri yang berjalan di sebuah *container* dalam salah satu *slave node*. Selama berjalan, komponen ini berkomunikasi dengan *Resource Manager* untuk memonitor siklus dari aplikasi serta mengatur permintaan dan pembagian sumber daya yang dibutuhkan *container*.

Job History Server

Job History Server merupakan komponen YARN yang bertugas menangani permintaan klien untuk riwayat *job* atau status *job* yang sedang berjalan dalam *cluster*. Komponen ini dijalankan di *daemon* dengan *container*-nya sendiri (deRoos et al. 2014).

Cara Kerja YARN

Resource Manager menerima *input* aplikasi dari klien, lalu mengatur sumber daya untuk *container* kemudian menjalankan *Application Master* untuk aplikasi tersebut. *Application Master* dan *Resource Manager* merundingkan *container* yang akan digunkan. *Application Master* kemudian memberi *container* ke *Node Manager* pada *host* yang akan menggunakan *container* tersebut. Eksekusi aplikasi diawasi oleh *Application Master* hingga selesai. Setelah selesai *Application Master* mengeluarkan diri dari *Resource Manager* (Shenoy, 2014).

2.2.5 Keunggulan Hadoop

Penggunaan Hadoop sangat bermanfaat dalam pemrosesan *Big Data*. Berikut ini adalah keuntungan yang bisa didapatkan oleh pengguna Hadoop:

a. Skalabilitas

   Hadoop dibangun dalam commodity hardware sehingga lebih mudah dan ekonomis dalam menambah perangkat jika terdapat tambahan beban atau pengguna.

b. Toleransi Kerusakan

   HDFS memberikan redundansi dan pemulihan. Jika salah satu *node* berhenti bekerja, *node* lain masih memiliki data yang dibutuhkan dengan adanya penggandaan data, yang merupakan fitur penting dari Hadoop. Proses komputasi yang mati akan dipindahkan ke *node* yang masih aktif sehingga tidak terjadi kehilangan data serta menjamin ketersediaan.

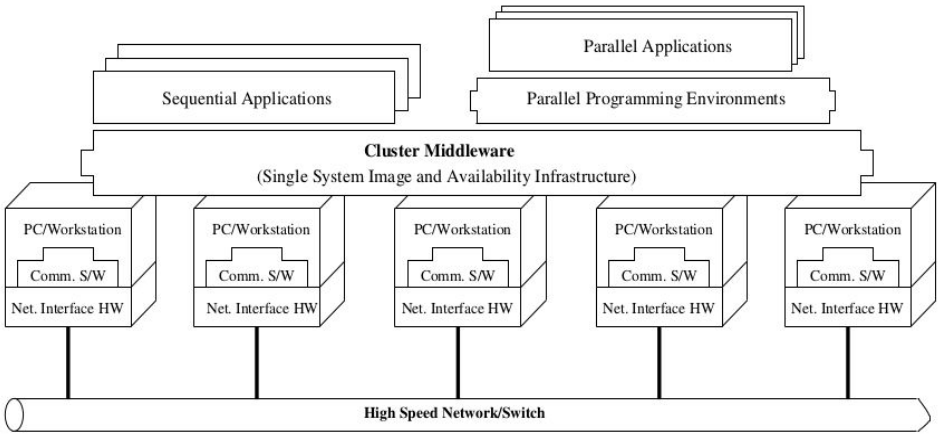
c. Sumber daya bersama

   Hadoop dirancang untuk dijalankan pada sebuah *cluster*, yang menggunakan konsep komputasi terdistribusi, sehingga sumber daya dan CPU di dalam *cluster* saling terhubung satu sama lain. Komputasi paralel dapat dicapai dengan mudah dengan Hadoop (Shenoy, 2014).

2.3 *Cluster Computing*

2.3.1 Pengertian *Cluster*

Cluster adalah jenis sistem pemrosesan paralel atau terdistribusi, yang terdiri dari sekumpulan komputer mandiri yang terhubung dan bekerja secara tunggal sebagai sebuah sumber daya komputasi terintegrasi. Sebuah *node* komputer dapat berupa sistem dengan prosesor tunggal atau ganda (komputer personal, workstation atau SMP) dengan memori, fasilitas input/output dan sistem operasi. *Cluster* biasanya merujuk kepada dua atau lebih komputer (*node*) yang dihubungkan. *node* bisa terdapat dalam satu tempat atau terpisah dan terhubung melalui LAN. komputer *cluster* yang terhubung dengan LAN dapat terlihat seperti sistem tunggal oleh pengguna dan aplikasi. Sistem tersebut dapat meringankan biaya untuk memperoleh fitur yang dulunya hanya dapat diperoleh dengan sistem berbagi memori yang mahal (Baker & Buyya, 1999). Arsitektur khas pada *cluster* dapat dilihat pada gambar 2.6.



Gambar 2.6. Arsitektur *cluster*

Sumber: Baker & Buyya (1999)

Beowulf *Cluster*

Salah satu contoh komputer *cluster* awal dan paling sering dibahas adalah Beowulf *Cluster*. Beowulf menyediakan performa tinggi, fleksibilitas dalam konfigurasi dan upgrade, serta skalabilitas untuk menyediakan sarana untuk

aplikasi komputasi. Beowulf *Cluster* dibuat dari beberapa buah komputer personal, prosesor, media penyimpanan berupa hard disk, serta modul memori murah sebagai memori utama. Beberapa merk prosesor telah digunakan dalam pembuatan Beowulf *Cluster*, seperti Intel, AMD, Compaq dan IBM. *Cluster* PC kelas Beowulf pertama dikembangkan oleh NASA di Goddard Space Flight Center pada tahun 1994 menggunakan sistem operasi Linux dan PVM yang berjalan pada 16 unit komputer personal berbasis prosesor Intel 100MHz 80486 yang dihubungkan menggunakan Ethernet LAN dengan kecepatan 10 Mbps. Sistem kelas Beowulf sejauh ini merupakan bentuk *cluster* komoditas yang paling populer (Sterling, 2002). *Cluster* Raspberry Pi yang dibangun dalam penelitian ini merupakan contoh *cluster* kelas Beowulf.

2.3.2 Komponen *Cluster*

Untuk membangun sebuah *cluster* diperlukan beberapa komponen, yang paling utama meliputi:

a. Beberapa komputer dengan performa tinggi (PC, *Workstation*, atau SMP).
b. Sistem operasi yang mutakhir (berlapis atau berbasis kernel mikro).
c. Jaringan beserta *switch* dengan performa tinggi (seperti Gigabit dan Myrinet).
d. *Network Interface Cards* / NIC (Kartu Antarmuka Jaringan).
e. Layanan dan protokol komunikasi cepat.
f. *Cluster middleware* (*Single System Image* (SSI) dan infrastruktur ketersediaan sistem).
g. *Environment* dan peralatan pemrograman paralel(seperti *compiler*, PVM (*Parallel Virtual Machine*), dan MPI (*Message Passing Interface*).
h. Aplikasi (Sekuensial dan paralel atau terdistribusi).

Node pada *cluster* dapat bekerja besamaan sebagai sumber daya komputasi terintegrasi, atau beroperasi sebagai komputer individual. *Middleware* dari *cluster* bertanggung jawab untuk menciptakan ilusi akan sistem tunggal, dan ketersediaan

melalui kumpulan komputer independen yang terhubung jadi satu. (Baker & Buyya, 1999).

2.3.3 Keunggulan *Cluster*

Penggunaan *cluster* berguna bagi perusahaan yang membutuhkan sistem dengan performa tinggi dengan kemampuan skalabilitas. *Cluster* memberikan fitur-fitur dengan biaya rendah, seperti:

a. Performa tinggi.
b. Skalabilitas dan ekspandibilitas.
c. *Throughput* tinggi.
d. Ketersediaan tinggi.

Teknologi *cluster* memungkinkan organisasi untuk meningkatkan kemampuan pemrosesannya menggunakan teknologi standar (*commodity hardware* dan komponen *software*) yang bisa didapatkan atau dibeli dengan biaya relatif rendah. Hal ini menyediakan ekspandibilitas dengan mempertahankan investasi yang ada tanpa mengeluarkan banyak biaya tambahan. Performa aplikasi juga meningkat dengan dukungan *software* yang dapat ditingkatkan. Keuntungan lain yaitu kemampuan *failover* yang memungkinkan komputer cadangan mengambil alih pekerjaan komputer yang mati dalam *cluster* (Baker & Buyya, 1999).

Sebuah *cluster* dibentuk dari sejumlah komputer personal (PC) yang terhubung dalam jaringan. Komputer pada *cluster* dapat terdiri dari beberapa bentuk. Komputer *single-board* seperti Raspberry Pi dapat digunakan sebagai pengganti PC untuk mempelajari *cluster*.

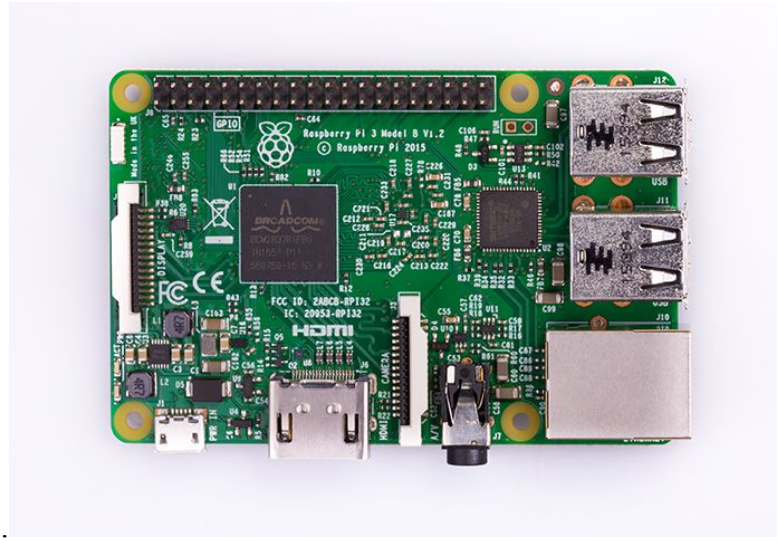
2.4 Raspberry Pi

2.4.1 Pengertian Raspberry Pi

Raspberry Pi adalah sebuah komputer *single-board s*eukuran kartu kredit yang dikembangkan oleh Raspberry Pi Foundation di UK. Raspberry Pi dibuat

sebagai media pembelajaran ilmu komputer dan pemrograman dengan harga terjangkau (“Raspberry Pi FAQs - Frequently Asked Questions,” n.d.)

Python dan Scratch merupakan bahasa pemrograman yang direkomendasikan oleh penciptanya. Selain dapat digunakan seperti komputer pada umumnya, membangun sistem fisik menggunakan sensor, motor dan pengontrol mikro juga dapat dilakukan. Raspberry Pi memiliki kemampuan multimedia dan grafis 3D yang mumpuni sehingga memiliki potensial sebagai *platform* video game (Upton and Halfacree, 2012). Gambar 2.7. memperlihatkan bentuk Raspberry Pi 3 model B yang digunakan dalam penelitian ini.



Gambar 2.7. Raspberry Pi 3 Model B

Sumber: “Raspberry Pi 3 Model B - Raspberry Pi,” n.d.

Sejarah Raspberry Pi

Rapsberry Pi awalnya dirancang dengan niat agar anak-anak mengerti cara kerja komputer dan aplikasi yang mereka gunakan. Eben Upton, pencipta Raspberry Pi ingin mempermudah pelajar dalam mempelajari pemrograman. Karena itu Upton ingin membuat komputer murah sebagai media untuk mempelajari pemrograman dengan ukuran kecil.

Awalnya proyek ini dinamai “ABC Micro” oleh Upton. Nama “Raspberry” digunakan karena mengikuti tradisi nama perusahaan komputer yang dulunya menggunakan nama buah (Apple, Tangerine, Apricot). Kata “Pi” diambil dari

"Python", yang awalnya akan digunakan sebagai bahasa pemrograman utama di Raspberry Pi. Namun pengguna dapat tetap menggunakan bahasa pemrograman lain.

Pada awal peluncurannya Rapsberry Pi mendapat pesanan dari 100.000 pembeli. Raspberry Pi Foundation merasa kesulitan untuk membuat dan memaketkan produk sebanyak itu sekaligus. Sehingga produksi dan distribusi diserahkan kepada perusahaan supplier elektronik , Element14 dan RS Component yang bebasis di UK (Upton & Halfacree, 2012).

Beberapa versi Raspberry Pi telah dirilis. Generasi pertama (Raspberry Pi Model B) dirilis pada Februari 2012, yang diikuti oleh Model A yang lebih murah dan sederhana. Tahun 2014 Raspberry Pi 1 Model B+ dirilis sebagai peningkatan dari Model B. Tahun 2015 Raspberry Pi Zero dirilis dengan ukuran lebih kecil dan I/O yang lebih sedikit disertai dengan GPIO (*General Purpose Input/Output*). Raspberry Pi 2 dengan peningkatan CPU dan RAM rilis pada Februari 2015. Setahun kemudian Raspberry Pi 3 dirilis dengan CPU dan RAM lebih besar serta penambahan fitur WiFi, Bluetooth dan USB *boot*. Pada Februari 2017 Raspberry Pi Zero W dirilis, dengan fitur yang sama dengan Raspberry Pi Zero serta penambahan fitur WiFi dan Bluetooth ("Raspberry Pi Products - Where to Buy Raspberry Pi," n.d.)

2.4.2 *Hardware*

Semua model Raspberry Pi menggunkan SoC (*System on a Chip*) Broadcom, yang memuat CPU dengan arsitektur ARM dan GPU dalam chipnya. Kecepatan CPU mulai dari 700 MHz hingga 1.2 GHz pada Pi seri 3 serta memori RAM yang berkisar antara 256 MB hingga 1 GB . *Secure Digital* (SD) card digunakan untuk menyimpan sistem operasi dan memori program. Semua model memiliki satu hingga empat slot USB, slot HDMI dan 3.5 mm jack audio. Pin GPIO menyediakan output tingkat rendah dengan dukungan protokol seperti I^2C ("Raspberry Pi Hardware - Raspberry Pi Documentation," n.d.). Spesifikasi *hardware* Raspberry Pi dapat dilihat pada tabel 2.1.

Tabel 2.1 Spesifikasi *Hardware* Raspberry Pi

Model	**1 A**	**1A+**	**1 B**	**1 B+**	**2 B**	**2 B ver. 1.2**	**3 B**	**Zero**	**Zero W**
Architecture	ARMv6Z (32-bit)				ARMv7-A (32-bit)	ARMv8-A (64/32-bit)		ARMv6Z (32-bit)	
SoC	Broadcom BCM2835				Broadcom BCM2836	Broadcom BCM2837		Broadcom BCM2835	
CPU	700 MHz single-core ARM1176JZF-S				900 MHz 32-bit quad-core ARM Cortex-A7	900 MHz 64-bit quad-core ARM Cortex-A53	1.2 GHz 64-bit quad-core ARM Cortex-A53	1 GHz single-core ARM1176JZF-S	
GPU	Broadcom VideoCore IV @ 250 MHz (BCM2837: 3D part of GPU @ 300 MHz, video part of GPU @ 400 MHz)								
	OpenGL ES 2.0 (BCM2835, BCM2836: 24 GFLOPS / BCM2837: 28.8 GFLOPS)								
	MPEG-2 and VC-1 (with license), 1080p30 H.264/MPEG-4 AVC high-profile decoder and encoder (BCM2837: 1080p60)								
Memory	256 MB (shared with GPU)		512 MB (shared with GPU)		1 GB (shared with GPU)			512 MB (shared with GPU)	
Network	None		10/100 Mbit/s Ethernet (8P8C) USB adapter on the USB hub				10/100 Mbit/s Ethernet, 802.11n wireless, Bluetooth 4.1	None	802.11n wireless, Bluetooth 4.1
USB	1 USB 2.0		2 USB 2.0		4 USB 2.0			1 Micro USB	
Power Source	5 V via MicroUSB or GPIO header								

2.4.3 *Software*

Raspberry Pi Foundation menyediakan sistem operasi yang dirancang khusus untuk Raspberry Pi bernama Raspbian, distribusi Linux berbasis Debian. Selain itu terdapat sistem operasi lain yang dapat digunakan, seperti Arch-Linux, Ubuntu, Windows 10 IOT, OSMC dan lain-lain. Raspberry Pi Foundation juga menyediakan sebuah *installer* bernama NOOBS, yang dibuat untuk memudahkan pengguna dalam melakukan instalasi sistem operasi. Bahasa pemrograman utama yang digunakan adalah Python dan Scratch, namun pengguna dapat menggunakan bahasa pemrograman lain ("Raspberry Pi Downloads - Software for the Raspberry Pi," n.d.).

.

BAB III
METODOLOGI PENELITIAN

3.1 Studi Pustaka

Studi pustaka dilakukan untuk mengumpulkan materi dan referensi terkait sistem yang akan dibuat. Materi yang dikumpulkan berupa pemahaman mengenai Hadoop, Raspberry Pi dan *cluster*, serta tutorial proyek *cluster* Raspberry Pi. Referensi diperoleh dari buku, artikel dan forum online, serta jurnal akademik.

Pemahaman yang dperoleh dari pengumpulan referensi yaitu berupa tutorial Raspberry Pi, petunjuk membangun *cluster*, serta konfigurasi Hadoop dan *cluster*. Selain itu dari jurnal diperoleh contoh rancangan *cluster* Raspberry Pi sehingga terdapat gambaran *cluster* yang akan dibuat.

3.2 Analisis Kebutuhan Sistem

Dari hasil pengumpulan data yang telah dilakukan maka didapatkan hasil analisis kebutuhan sistem dalam proses penelitian analisis kinerja Hadoop pada *cluster* Raspberry Pi. Analisis kebutuhan sistem dalam penelitian ini dibagi meliputi analisis perangkat keras, analisis perangkat lunak dan analisis kebutuhan data.

3.2.1 Analisis Kebutuhan Perangkat Keras

Kebutuhan perangkat keras yang dibutuhkan untuk membangun *cluster* Hadoop yaitu :

a. Raspberry Pi
   - 5 buah Raspberry Pi 3 Model B dibutuhkan untuk membangun *cluster* yang akan digunakan.
   - Raspberry Pi sudah dilengkapi dengan CPU sebesar 1.2 GHz quad core dan 1 GB RAM namun tidak disertai media penyimpanan.

b. MicroSD Card
   - MicroSD digunakan untuk media penyimpanan pada tiap Raspberry Pi.

- MicroSD yang digunakan adalah class 6 dengan ukuran sebesar 8 GB, sesuai rekomendasi teknis.

c. USB *Power Supply*
   - Suplai daya listrik menggunakan sebuah USB charger merk Micropack dengan tegangan 5V dan kapasitas 12 A yang disertai 6 slot USB .

d. Kabel USB mikro
   - 6 buah kabel USB mikro digunakan untuk menghubungkan Raspberry Pi dengan suplai daya

e. Network Switch
   - *Network switch* TP-Link 10/100 mbit/s dengan 8 port ethernet digunakan untuk menghubungkan semua Raspberry Pi dan operator dengan jaringan LAN menggunakan kabel ethernet.

f. Kabel ethernet
   - Kabel ethernet digunakan untuk menghubungkan semua node dalam cluster melalui *switch.*

g. Laptop
   - Laptop digunakan sebagai operator untuk mengoperasikan dan mengontrol *cluster* karena *cluster* tidak menggunakan *display* dan *input*.
   - Sebuah Laptop dengan CPU 1.7 GHz dual core dan 6 GB RAM digunakan sebagai operator.
   - Laptop yang digunakan sebagai operator juga digunakan untuk konfigurasi dan perbandingan performa *cluster*.

Biaya *Hardware Cluster*

Tiap *hardware* yang digunakan untuk cluster memiliki biaya tersendiri. Tabel 3.1 menunjukkan harga tiap *hardware* untuk memberi gambaran biaya pembangunan *cluster* Raspberry Pi.

Tabel 3.1. Harga *Hardware*

Item	Harga
Raspberrry Pi	Rp. 550.000
MicroSD 8GB	Rp. 45.000
Power Supply Micropack	Rp. 375.000
Kabel USB Mikro	Rp. 10.000
Network Switch TP-Link	Rp. 85.000

3.2.2 Analisis Kebutuhan Perangkat Lunak

Kebutuhan perangkat lunak yang digunakan untuk penelitian ini adalah sebagai berikut :

a. Sistem Operasi
   - Raspbian

     Raspbian versi 4.4.50 digunakan sebagai sistem operasi pada tiap Raspberry Pi.
   - Kali Linux

     Operator menggunakan sistem operasi Kali Linux versi 4.6.0 yang telah dipasang sebelumnya.

b. Hadoop
   - Hadoop versi 2.7.3 dipasang di tiap Raspberry Pi dalam *cluster* serta pada operator.
   - Instalasi Hadoop sudah disertai contoh program MapReduce yang dapat digunakan untuk uji performa.

c. Perangkat Lunak Tambahan
   - Java Development Kit

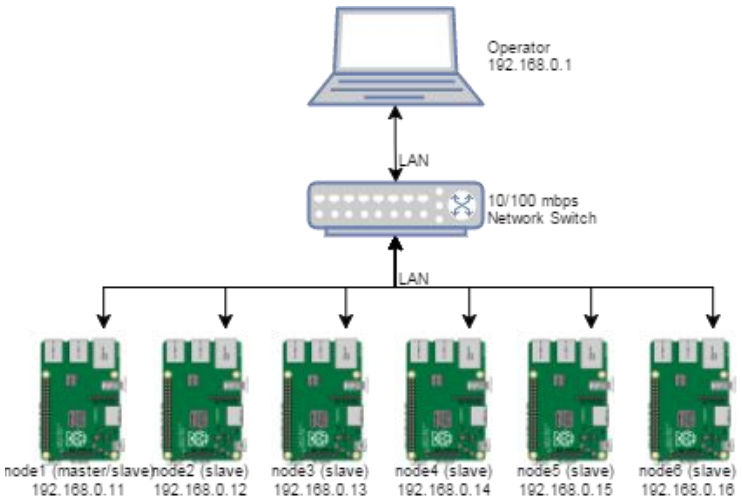
     Java dibutuhkan untuk menjalankan program MapReduce di Hadoop.
   - Java versi 1.8.0 dipasang pada *cluster* dan operator.

3.2.3 Kebutuhan Data

Pengujian performa Hadoop menggunakan data berukuran besar. Data yang akan digunakan diambil dari sumber publik yaitu Google Books. Data yang digunakan berupa file teks dengan ukuran 400 MB hingga 4 GB.

3.3 Rancang Bangun *cluster*

Perancangan *cluster* perlu dilakukan sebelum membangun *cluster* yang akan digunakan untuk uji performa. Gambar 3.1 menunjukkan diagram rancangan *cluster* Raspberry Pi yang akan dibangun.



Gambar 3.1. Diagram *cluster* Raspberry Pi

Setelah perancangan selesai berikutnya adalah membangun *cluster* dengan menghubungkan semua hardware untuk *cluster* yang telah disiapkan. Raspberry Pi disusun bertingkat dengan baut *spacer*. Raspberry kemudian dihubungkan dengan jaringan LAN menggunakan kabel ethernet melalui *switch*. Suplai daya dihubungkan dengan kabel USB mikro. Semua *hardware* disusun menjadi satu dengan papan akrilik. *cluster* Raspberry Pi yang telah dibangun dapat dilihat pada Gambar 3.2.

Gambar 3.2. *Cluster* Raspberry Pi

3.4 Instalasi dan Konfigurasi Perangkat Lunak

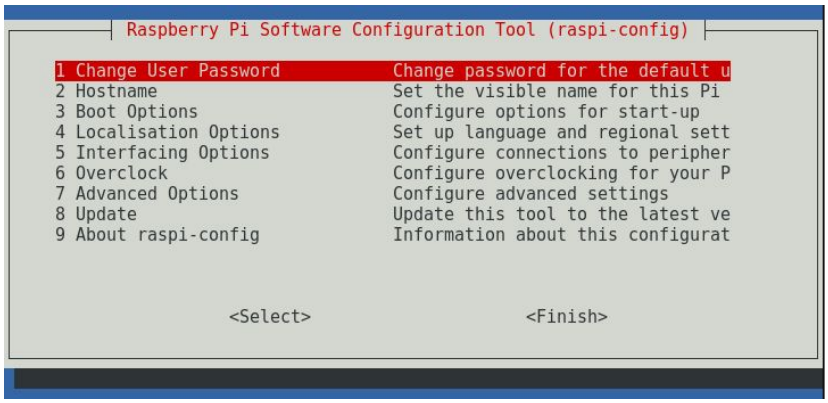
Setelah *cluster* selesai dibangun tahap berikutnya adalah instalasi sistem operasi dan software yang dibutuhkan serta konfigurasi sistem.

3.4.1 Instalasi Sistem Operasi

Instalasi sistem operasi pada Raspberry Pi dilakukan dengan memasukkan *image* sistem operasi Raspbian ke kartu MicroSD yang akan digunakan. Proses ini dilakukan di laptop dengan sistem operasi Kali Linux. *Image* sistem operasi ditulis ke MicroSD dengan memasukkan perintah berikut pada terminal Kali Linux : `dd bs=4M if=2016-05-27-raspbian-jessie-lite.img of=/dev/mmcblk0`. Perintah tersebut memasukkan image file bernama **2016-05-27-raspbian-jessie-lite.img** ke kartu MicroSD **/dev/mmcblk0**. Setelah selesai MicroSD dapat dipasang dan digunakan di Raspberry Pi.

3.4.2 Konfigurasi Sistem

Setelah instalasi, MicroSD dipasang dan Raspberry dinyalakan. Konfigurasi awal membutuhkan keyboard dan layar monitor. Sebelum konfigurasi dilakukan pembaruan dengan perintah `apt-get update`. Konfigurasi awal dilakukan dengan perintah `raspi-config` pada terminal, yang akan mengeluarkan tampilan pada Gambar 3.3.



Gambar 3.3. Tampilan Menu Perintah raspi-config

Konfigurasi yang dilakukan meliputi:

a. Mengganti *username* dan *password* bawaan Raspberry Pi.

b. Mengganti hostname menjadi **node1**.

c. Memperbesar file system agar seluruh ruang MicroSD digunakan.

Konfigurasi Jaringan

a. Mengubah alamat IP

Mengganti alamat IP dilakukan dengan mengubah isi file **/etc/network/interfaces** dengan teks editor pada terminal. Konfigurasi alamat IP dapat dilihat pada Gambar 3.4.

Gambar 3.4. Konfigurasi Alamat IP

b. Menambah host

Menambah host pada file **/etc/hosts** dilakukan untuk merujuk ke *hostname* pada *node* dalam *cluster* agar mempermudah dalam koneksi. Gambar 3.5 menunjukkan isi file ***/etc/hosts*** dengan *hostname* yang telah ditambahkan.

Gambar 3.5. File hosts

Menambah Akun Pengguna Hadoop

Akun pengguna untuk Hadoop dibuat untuk memsihakan dengan akun pengguna lain. Gambar 3.6. Menunjukkan perintah untuk membuat akun **hduser** dalam grup **hadoop**

```
sudo addgroup hadoop
sudo adduser --ingroup hadoop hduser
sudo adduser hduser sudo
```

Gambar 3.6 Membuat Akun hduser

Konfigurasi SSH

Konfigurasi SSH dilakukan untuk membuat pasangan kunci RSA untuk SSH dengan *password* kosong agar *node* pada *cluster* dapat berkomunikasi tanpa meminta *password*. Perintah konfigurasi dapat dilihat pada pada Gambar 3.7.

```
su hduser
mkdir ~/.ssh
ssh-keygen -t rsa -P ""
cat ~/.ssh/id_rsa.pub > ~/.ssh/authorized_keys
```

Gambar 3.7. Konfigurasi SSH

3.4.3 Instalasi dan Konfigurasi Java

Java dibutuhkan untuk menjalankan Hadoop. Untuk instalasi Java dilakukan dengan perintah `apt-get install oracle-java8-jdk`. Perintah tersebut akan melakukan instalasi java pada sistem. Untuk mengecek apakah instalasi Java berhasil dimasukkan perintah `java -version`.

Setelah konfigurasi sistem selesai Raspberry Pi di-*restart*. Setelah itu konfigurasi dapat dilakukan dengan laptop melalui koneksi SSH.

3.4.4 Instalasi dan Konfigurasi Hadoop

Hadoop diunduh kemudian diekstrak pada direktori **/opt** dengan perintah seperti pada Gambar 3.6. Direktori hasil ekstrak diubah namanya menjadi **hadoop** dan diubah kepemilikannya menjadi milik akun **hduser**.

```
sudo mkdir /opt
sudo tar -xvzf hadoop-2.7.3.tar.gz -C /opt/
cd /opt
sudo mv hadoop-.2.7.3 hadoop
sudo chown -R hduser:hadoop hadoop
```

Gambar 3.8. Instalasi dan Pergantian Pemilik Hadoop.

Konfigurasi *Environment Variable*

a. *Environment Variable* Raspbian

Baris seperti Gambar 3.9 ditambahkan pada akhir baris file **/etc/bash.bashrc** untuk mengarahkan sistem pada direktori Hadoop dan Java.

```
  GNU nano 2.2.6               File: /etc/bash.bashrc

export HADOOP_INSTALL=/opt/hadoop
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:/opt/hadoop/sbin
export PATH=$PATH:/home/hduser
```

Gambar 3.9. *Environment Variable* Raspbian

b. *Environment Variable* Hadoop

Baris yang diblok pada Gambar 3.10 ditambahkan pada file **/opt/hadoop/etc/hadoop/hadoop-env.sh** untuk mengarahkan Hadoop pada direktori Java.

```
  GNU nano 2.2.6      File: /opt/hadoop/etc/hadoop/hadoop-env.sh

# Set Hadoop-specific environment variables here.

# The only required environment variable is JAVA_HOME.  All others are
# optional.  When running a distributed configuration it is best to
# set JAVA_HOME in this file, so that it is correctly defined on
# remote nodes.

# The java implementation to use.
#export JAVA_HOME=$(readlink -f /usr/bin/java | sed "s:bin/java::")
export JAVA_HOME=/usr/lib/jvm/jdk-8-oracle-arm32-vfp-hflt
# The jsvc implementation to use. Jsvc is required to run secure datanodes
# that bind to privileged ports to provide authentication of data transfer
# protocol.  Jsvc is not required if SASL is configured for authentication of
# data transfer protocol using non-privileged ports.
#export JSVC_HOME=${JSVC_HOME}
```

Gambar3.10. Direktori Java pada *Environment Variable* Hadoop

Konfigurasi Hadoop

Konfigurasi Hadoop dilakukan dengan mengubah isi file konfigurasi pada direktori **/opt/hadoop/etc/hadoop/**. Hadoop akan menggunakan aturan default pada bagian konfigurasi yang tidak diubah. File konfigurasi yang diubah adalah sebagai berikut:

a. core-site.xml

File **core-site.xml** memberi informasi ke Hadoop mengenai lokasi NameNode berjalan pada *cluster*. File ini berisi konfigurasi dasar Hadoop seperti direktori dan lokasi HDFS Konfigurasi seperti pada Gambar 3.11 ditulis pada file **core-site.xml**.

```
<configuration>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/hdfs/tmp</value>
  </property>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://node1:54310</value>
  </property>
</configuration>
```

Gambar 3.11. core-site.xml

b. hdfs-site.xml

File **hdfs-site.xml** berisi konfigurasi untuk daemon HDFS. Pada file ini pengguna dapat mengatur faktor penggandaan dan ukuran blok data. Faktor penggandaan menentukan berapa banyak salinan data yang dimasukkan di HDFS. Konfigurasi seperti pada Gambar 3.12 ditulis pada file **hdfs-site.xml**.

```
<configuration>
 <property>
    <name>dfs.replication</name>
    <value>1</value>
 </property>
 <property>
    <name>dfs.block.size</name>
    <value>15728640</value>
 </property>
</configuration>
```

Gambar 3.12. hdfs-site.xml

c. mapred-site.xml

File **mapred-site.xml** berisi aturan konfigurasi untuk *daemon* MapReduce. Pada file ini pengguna menentukan lokasi JobTracker, *framework* MapReduce, serta alokasi sumber daya untuk aplikasi MapReduce. Konfigurasi seperti pada Gambar 3.13 ditulis pada file **mapred-site.xml**.

```
<configuration>
  <property>
    <name>mapred.job.tracker</name>
    <value>node1:54311</value>
  </property>
  <property>
     <name>mapreduce.framework.name</name>
```

```
     <value>yarn</value>
  </property>
<!--Task Allocation-->
  <property>
<name>mapred.tasktracker.map.tasks.maximum</name>
<value>4</value>
  </property>
  <property>
<name>mapred.tasktracker.reduce.tasks.maximum</name>
<value>4</value>
  </property>
<!--Memory Allocation-->
  <property>
      <name>mapreduce.map.memory.mb</name>
      <value>256</value>
  </property>
  <property>
     <name>mapreduce.map.java.opts</name>
     <value>-Xmx205m</value>
  </property>
  <property>
     <name>mapreduce.map.cpu.vcores</name>
     <value>1</value>
  </property>
  <property>
     <name>mapreduce.reduce.memory.mb</name>
     <value>128</value>
  </property>
  <property>
     <name>mapreduce.reduce.java.opts</name>
     <value>-Xmx102m</value>
  </property>
  <property>
     <name>mapreduce.reduce.cpu.vcores</name>
     <value>1</value>
  </property>
<!--Application Master-->
  <property>
     <name>yarn.app.mapreduce.am.resource.mb</name>
     <value>256</value>
  </property>
  <property>
     <name>yarn.app.mapreduce.am.command-opts</name>
     <value>-Xmx205m</value>
  </property>
  <property>
     <name>yarn.app.mapreduce.am.resource.cpu-vcores</name>
     <value>1</value>
  </property>
</configuration>
```

Gambar 3.13 mapred-site.xml

d. yarn-site.xml

File ini berisi konfigurasi untuk YARN. Pada file ini pengguna dapat menentukan lokasi *Resource Manager* serta alokasi sumber daya untuk

container. Konfigurasi seperti pada Gambar 3.14 ditulis pada file **yarn-site.xml**.

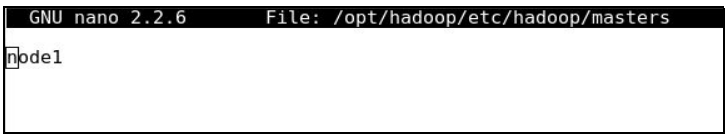
```
<configuration>
 <property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
 </property>
 <property>
  <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
  <value>org.apache.hadoop.mapred.ShuffleHandler</value>
 </property>
<!--Addresses-->
 <property>
  <name>yarn.resourcemanager.scheduler.address</name>
  <value>node1:8030</value>
 </property>
 <property>
  <name>yarn.resourcemanager.address</name>
  <value>node1:8032</value>
 </property>
 <property>
  <name>yarn.resourcemanager.webapp.address</name>
  <value>node1:8088</value>
 </property>
 <property>
  <name>yarn.resourcemanager.resource-tracker.address</name>
  <value>node1:8031</value>
 </property>
 <property>
  <name>yarn.resourcemanager.admin.address</name>
  <value>node1:8033</value>
 </property>
<!--CPU-->
 <property>
  <name>yarn.nodemanager.resource.cpu-vcores</name>
  <value>4</value>
 </property>
 <property>
  <name>yarn.scheduler.minimum-allocation-vcores</name>
  <value>1</value>
 </property>
 <property>
  <name>yarn.scheduler.maximum-allocation-vcores</name>
  <value>4</value>
 </property>
<!--Memory-->
 <property>
  <name>yarn.nodemanager.resource.memory-mb</name>
  <value>768</value>
 </property>
 <property>
  <name>yarn.scheduler.minimum-allocation-mb</name>
  <value>256</value>
 </property>
 <property>
  <name>yarn.scheduler.maximum-allocation-mb</name>
  <value>768</value>
 </property>
```

```
</configuration>
```

Gambar 3.14 yarn-site.xml

e. masters

File **masters** berisi nama host yang menjalankan NameNode cadangan. Biasanya host tersebut menetukan *master node* pada *cluster*. File **masters** disimpan di *master node* . Isi file ditulis seperti pada Gambar 3.15 untuk menjadikan **node1** sebagai *master node*.



Gambar 3.15. masters

f. slaves

File **slaves** berisi *slave node* dari *cluster*. File ini dapat diubah untuk memasukkan atau mengeluarkan *node* pada *cluster*. File **slaves** disimpan di *master node.* Isi file diubah menjadi seperti pada Gambar 3.16 untuk menyertakan kelima *node* pada *cluster*.

Gambar 3.16. slaves

Membuat Hadoop File System

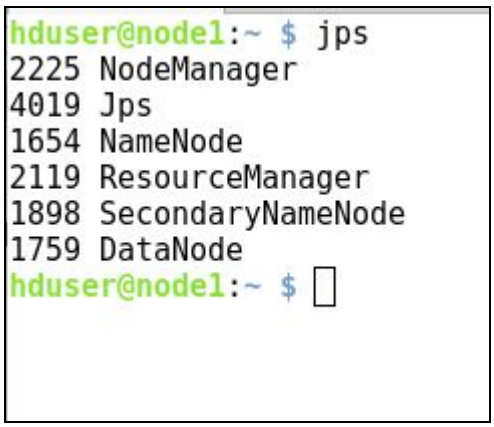
File system pada Hadoop menggunakan HDFS. *File system* dibuat dengan memasukkan perintah seperti Gambar 3.17.

```
sudo mkdir -p /hdfs/tmp
sudo chown hduser:hadoop /hdfs/tmp
sudo chmod 750 /hdfs/tmp
hadoop namenode -format
```

Gambar 3.17. Membuat HDFS

Menjalankan Hadoop

Perintah `start-all.sh` digunakan untuk menjalankan *framework* Hadoop pada semua *node*. Perintah `jps` dapat dimasukkan untuk mengkonfirmasi bahwa semua layanan Hadoop telah berjalan. Jika tampilan perintah `jps` muncul seperti Gambar 3.18 artinya semua layanan Hadoop telah berjalan.



Gambar 3.18. *Ouput* Perintah `jps`

3.5 Kloning MicroSD

Setelah semua konfigurasi selesai Raspberry Pi dimatikan dan MicroSD dikeluarkan untuk di-klon. Kloning mempermudah konfigurasi sehingga tidak perlu dilakukan dari awal di setiap *node*.

Kloning diakukan dengan membuat image dari MicroSD. Proses ini dilakukan dengan perintah `dd bs=4M if=/dev/mmcblk0 of=raspbian-hadoop.img`. Perintah ini membuat salinan dari MicroSD ke file bernama **raspbian-hadoop.img**.

Image file kemudian dimasukkan ke kartu MicroSD yang akan digunakan pada *node cluster*. *Hostname* dan alamat IP diganti sesuai dengan file **/etc/hosts** pada Gambar 3.5.

3.6 Uji Performa *Cluster* Hadoop

Setelah *cluster* dibangun dan software terpasang, uji performa Hadoop dapat dilakukan. Uji performa dilakukan dengan beberapa skenario, yaitu uji skalabilitas, uji ketahanan, uji toleransi kerusakan, serta penggunaan sumber daya.

3.6.1 Uji skalabilitas

Uji skalabilitas dilakukan untuk mengetahui bagamana peningkatan performa *cluster* dengan penambahan *node*. Pengujian dilakukan dengan menjalankan aplikasi WordCount dengan file teks sebesar 400 MB. Aplikasi dijalankan hingga selesai di satu *node*, kemudian *node* lain ditambahkan dan aplikasi dijalankan lagi. Pengujian dilakukan hingga semua *node* digunakan. Waktu eksekusi kemudian diperiksa untuk menilai perubahan dengan penambahan *node*. Waktu eksekusi dibandingkan dengan laptop yang menjadi operator.

Konfigurasi

File teks yang akan digunakan disalin terlebih dahulu ke HDFS dengan perintah `hdfs dfs -copyFromLocal /nama-file /file-input-hdfs`. Hadoop akan menyalin file dari direktori lokal sebuah *node* dan membaginya menjadi blok-blok data yang kemudian didistribusikan dalam HDFS pada *cluster*

Konfigurasi jumlah *node* dilakukan dengan mengubah isi file **slaves** pada direktori konfigurasi Hadoop. Pengurangan *node* dilakukan dengan menghapus *hostname node* yang tidak disertakan, sedangkan penambahan *node* dilakukan dengan menuliskan *hostname node* yang akan ditambahkan.

Setelah file disalin ke HDFS aplikasi WordCount dijalankan dengan perintah `hadoop jar /opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.3.jar wordcount /file-input-hdfs /file-output`.

3.6.2 Uji ketahanan

Uji ketahanan dilakukan untuk mengetahui bgaimana performa *cluster* dalam memproses file berukuran besar. Pengujian dilakukan menggunakan

aplikasi WordCount dengan file teks berukuran besar, mulai dari 1 GB hingga 4 GB. Waktu eksekusi diperiksa untuk menilai perbedaan waktu berdasarkan ukuran file. Waktu eksekusi dibandingkan dengan laptop yang menjadi operator.

Konfigurasi

Konfigurasi pada skenario ini hampir sama dengan uji skalabilitas, bedanya dalam skenario ini semua *node* digunakan dalam tiap pengujian. Setelah aplikasi selesai dijalankan, HDFS dikosongkan dan diisi file baru. Selain itu dalam uji kinerja dengan file berkuran besar dilakukan *balancing* untuk mendistribusikan blok file secara merata dalam *cluster*. *Balancing* dilakukan setelah file dimasukkan ke HDFS, dengan perintah `hadoop balancer`. Opsi `-threshold` dapat ditambahkan untuk menentukan batas persentase penggunaan disk pada *node*. Setelah *balancing* selesai aplikasi WordCount dapat dijalankan.

3.6.3 Uji Toleransi Kerusakan

Uji toleransi kerusakan (*fault tolerance*) dilakukan untuk memastikan *cluster* Hadoop tetap berjalan jika salah satu *node* mati saat aplikasi berjalan. Pengujian dilakukan dengan aplikasi WordCount dengan ukuran data 400MB. Salah satu *node* diputuskan dari jaringan secara paksa saat aplikasi berjalan untuk mensimulasikan kerusakan pada *node*. Pengujian ini juga untuk memastikan ketersediaan (*availability*) data dengan membandingkan kedua *output* hasil pengujian.

3.6.4 Penggunaan Sumber Daya

Penggunaan sumber daya pada tiap *node* dicatat saat menjalankan aplikasi. Pencatatan dilakukan saat uji ketahanan dengan ukuran data 1 GB. Penggunaan sumber daya diperoleh dengan perintah `top` untuk CPU, `free` untuk memori, serta `iostat` untuk *input/output*. Sebuah *script* dibuat untuk mempermudah pengumpulan data serta mempersingkat penggunaan perintah pada terminal. Script yang dibuat dapat dilihat pada Gambar 3.19.

```
#!/bin/bash
#Script for collecting resource usage value
echo -e  "\nCollecting resource value. press Ctrl+C when done\n "
while true;
do
#CPU read
    top -b -n 2 -d.5 | grep Cpu |tail -n1 |awk '{printf 100-$8"\t"}' >>
stat
#get value of CPU usage then put into temporary stat file
#Memory read
    free -m | grep Mem | awk '{printf $3/$2 *100"\t "}' >> stat
#get value of memory usage then put int temporary stat file
#I/O read
    iostat -yhkxd 1 1 | grep "                  " | awk ' {print $13"\t",
$5"\t", $6}' >>   stat
#get value of Input/Output % & read/write speed then put into temporary
stat file
#Write into txt file
    cat stat | nl > stat.txt
#include line number and write the content of temporary stat file into
stat.txt file
#Delay
    sleep 0.5
#0.5 second interval between each loop
done
```

Gambar 3.19. Script Pencatat Penggunaan Sumber Daya.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Analisis dan Hasil Evaluasi Uji Performa

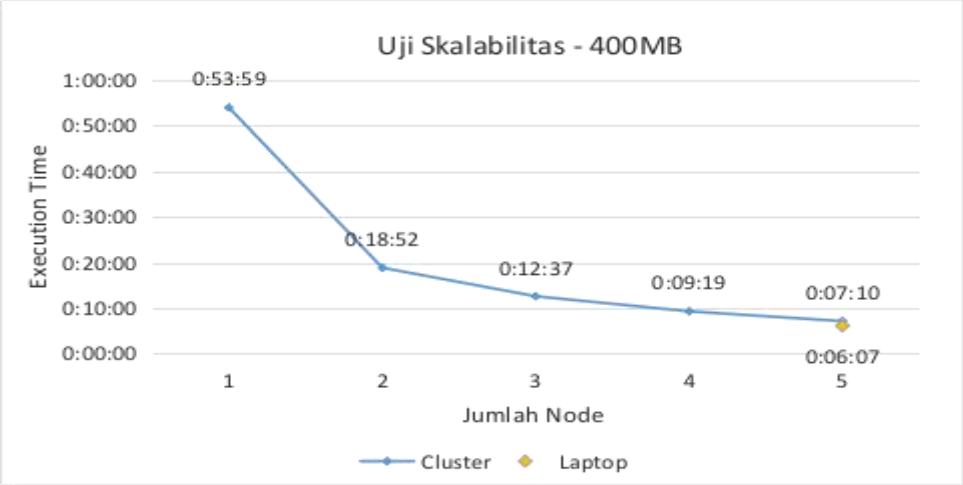
Pada bagian ini akan dibahas mengenai analisis hasil uji performa Hadoop yang telah dilakukan. Hasil yang dianalisis berupa grafik dan tabel yang menunjukkan hasil pengujian Hadoop di *cluster* Raspberry Pi. Variabel yang dianalisis berupa ukuran data yang diproses, jumlah *node* serta waktu eksekusi.

4.2 Uji skalabilitas

Uji skalabilitas menggunakan aplikasi WordCount dengan ukuran data sebesar 400 MB. Hasil uji skalabilitas dapat dilihat pada Tabel 4.1. Dari tabel dapat dilihat waktu eksekusi lebih cepat saat *node* lain ditambahkan. Hal ini membuktikan skalabilitas *cluster* Hadoop. Grafik perubahan waktu eksekusi dapat dilihat pada Gambar 4.1.

Tabel 4.1. Uji Skalabilitas

Jumlah Node Cluster	**Waktu Eksekusi**		**Persentase Peningkatan**
	Cluster	**Laptop (1 node)**	
1	53m 59s	6m 7s	0%
2	18m 52s		65%
3	12m 37s		33%
4	9m 19s		26%
5	7m 10s		23%



Gambar 4.1. Perbandingan Waktu Eksekusi

Pada gambar 4.1 diketahui bahwa penambahan *node* dapat mempercepat waktu eksekusi aplikasi. Peningkatan performa paling signifikan terlihat pada penambahan **node2**. Titik kuning pada grafik menunjukkan waktu eksekusi laptop yang menjadi perbandingan. Dari hasil pengujian diketahui dengan 5 buah *node* belum dapat mengimbangi performa laptop denga harga setara.

Prediksi Penambahan *Node*

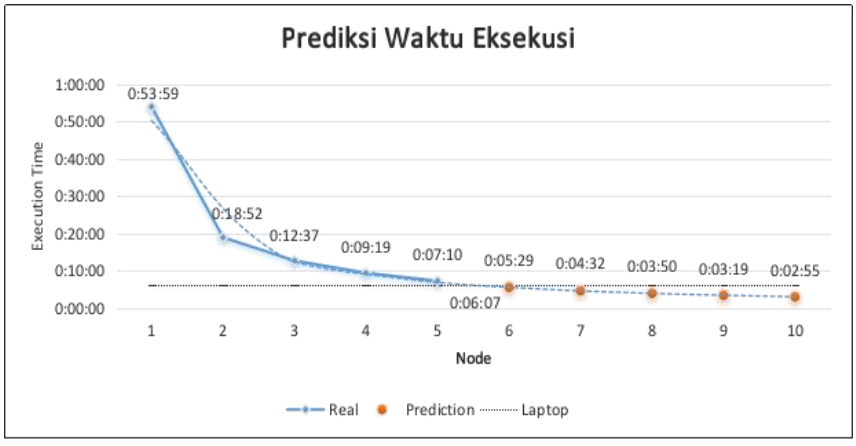
Meski belum dapat menyamai performa laptop, dari data grafik dapat dilakukan prediksi waktu eksekusi pada tiap penambahan jumlah *node*. Fungsi **power** digunakan pada grafik untuk mencari rumus prediksi untuk waktu eksekusi pada penambahan *node* selanjutnya. Waktu eksekusi jumlah *node* diprediksi dan dibandingkan dengan nilai aslinya untuk dilakukan perbandingan akurasi prediksi. Tabel 4.2 memperlihatkan rumus prediksi di mana y adalah waktu eksekusi dan x adalah jumlah *node*, serta perbandingan hasil prediksi dan hasil pengujian.

Tabel 4.2 Prediksi Waktu Eksekusi

Jumlah Data Prediksi	Rumus	Jumlah Node	Waktu Asli	Waktu Prediksi	Selisih
3	$y = 0.0363x^{-1.344}$	4	9m 19s	8m 7s	1m 12s
		5	7m 10s	6m 1s	1m 9s
4	$y = 0.03539x^{-1.2684}$	5	7m 10s	6m 37s	0m 33s

Dari tabel diketahui dengan prediksi menggunakan 3 *node* terdapat selisih sekitar **1 menit** antara waktu asli dan waktu hasil prediksi. Sedangkan dengan 4 *node* selisih waktu berkurang menjadi **33 detik.** Terdapat peningkatan akurasi antara jumlah data prediksi, sehingga dengan prediksi menggunakan 5 *node* dapat diasumsikan bahwa akurasi akan meningkat dan selisih antara waktu prediksi dan waktu asli akan lebih kecil. Gambar 4.2 menunjukkan grafik hasil prediksi dengan 5 *node* sebagai data dengan persamaan (4.1).

$$y = 0.03486x^{-1.2361} \tag{4.1}$$



Gambar 4.2. Prediksi Waktu Eksekusi

Dari grafik diketahui bahwa *cluster* diprediksi dapat mengungguli laptop pada 6 *node*. Namun penambahan *node* selanjutnya tidak meningkatkan performa

secara signifikan. Hal ini dipengaruhi oleh hukum **Amdahl**, yang menyatakan terdapat batas peningkatan kecepatan pada pemrosesan paralel dimana kecepatan pemosesan hanya meningkat secara signifikan saat mencapai jumlah tertentu.

4.3 Uji ketahanan

Uji ketahanan dilakukan dengan memproses data berukuran besar, mulai dari 1 GB hingga 4GB. Ukuran data yang lebih besar dari 4 GB mengakibatkan kepenuhan ruang disk *master node* pada *cluster* sehingga proses tidak bisa dilanjutkan. Tabel 4.2 menunjukkan ukuran data yang diproses serta waktu eksekusinya dengan waktu ekesekusi laptop sebagai perbandingan

Tabel 4.3. Perbandingan Uji Ketahanan

Ukuran Data	**Waktu Eksekusi**	
	Cluster	**Laptop**
1 GB	16m 41s	15m 33s
2 GB	28m 1s	27m 15s
4 GB	50m 48s	55m 2s

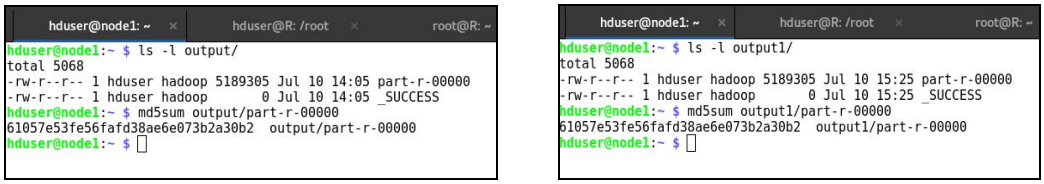
Dari tabel dapat diketahui bahwa *cluster* dapat memproses data hingga 4 GB. Pada tabel juga dapat dilihat bahwa pada ukuran data 4 GB performa *cluster* mengungguli laptop dalam hal kecepatan waktu eksekusi. Dari hasil tersebut diketahui bahwa *cluster* dapat bekerja dengan performa lebih baik dengan ukuran data di atas 2 GB. Namun pengujian dengan ukuran data yang lebih besar tidak memungkinkan pada *cluster* karena keterbatasan *storage.*

4.4 Uji Toleransi Kerusakan

Pengujian ini dilakukan dengan memutus jaringan LAN secara paksa pada **node3** saat aplikasi berjalan sehingga hanya 4 *node* yang tersisa dalam *cluster*. Pengujian menggunakan aplikasi WordCount dengan ukuran data 400 MB. Dalam pengujian ini Hadoop tetap dapat menyelesaikan aplikasi yang berjalan. Namun

waktu eksekusi yang dibutuhkan untuk menyelesaikan aplikasi meningkat menjadi **12 menit 30 detik**, setara dengan 3 buah *node* yang berjalan secara normal.

Pengujian ini juga untuk memastikan ketersediaan (*availability*) data. Data o*utput* hasil pengujian dibandingkan dengan pengujian normal dengan kelima *node*, untuk mengetahui keutuhan data. Perbandingan kedua *output* data pengujian dapt dilihat pada Gambar 4.3. **part-r-00000** merupakan nama file teks hasil *output* dari pengujian.



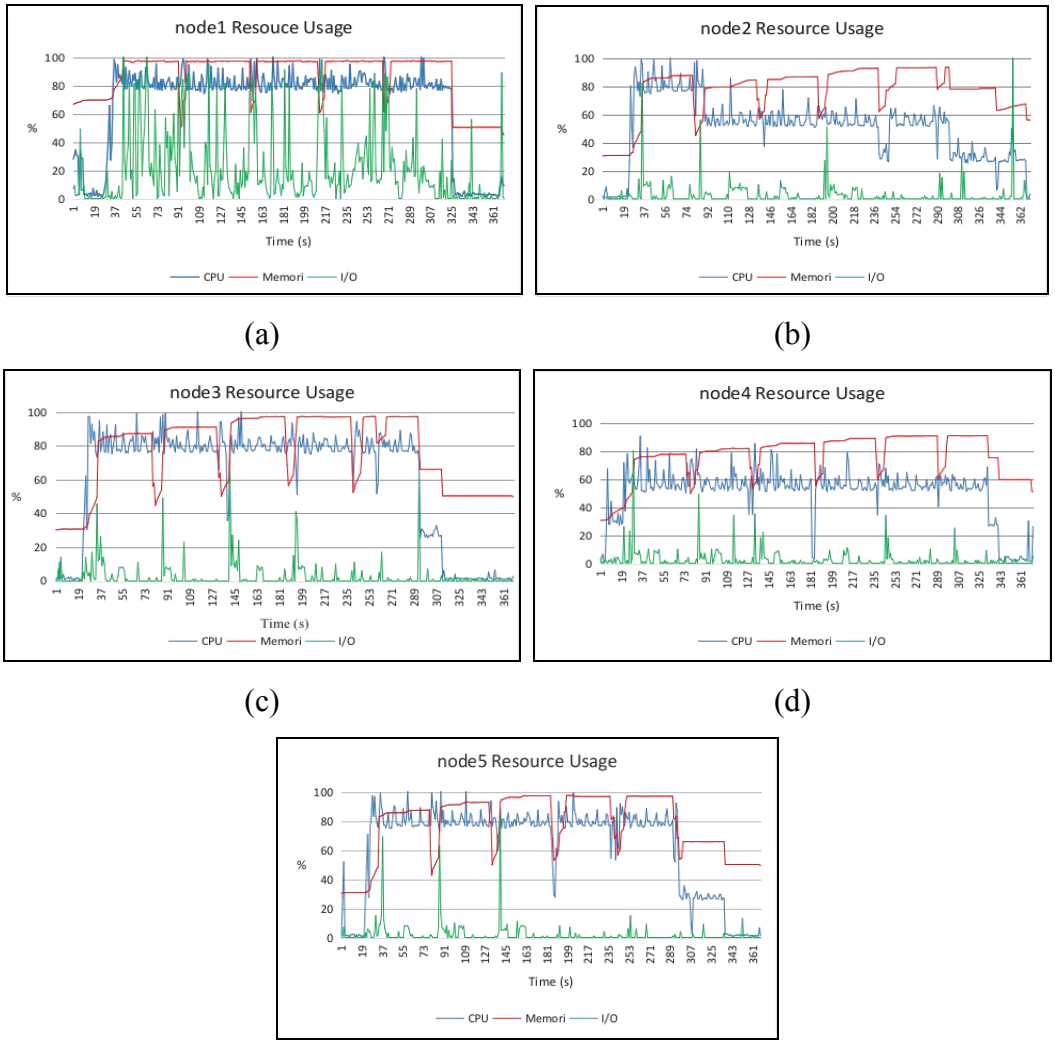
(a) (b)

Gambar 4.3. Perbandingan Data Output. (a) Uji Toleransi Kerusakan; (b) Pengujian Normal

Dari gambar 4.3 dapat dilihat bahwa file teks dari kedua pengujian berukuran sama. Dengan melakukan *checksum* dapat dilihat bahwa kedua file memiliki nilai MD5 yang sama. Hal ini memastikan tidak ada data yang hilang jika salah satu *node* mati saat menjalankan aplikasi, sehingga memastikan ketersediaan (*availability*) data.

4.5 Penggunaan Sumber Daya

Penggunaan sumber daya diukur menggunakan perintah `top`, `free` dan `iostat` untuk membaca nilai pengunaan CPU, memori dan *input/output*. Nilai pembacaan disimpan di file teks kemudian dibuat menjadi grafik. Gambar 4.4 menunjukkan persentase penggunaan sumber daya pada kelima *node* pada *cluster*, sedangkan Gambar 4.5 menunjukkan kecepatan *read* dan *write* selama aplikasi berjalan.



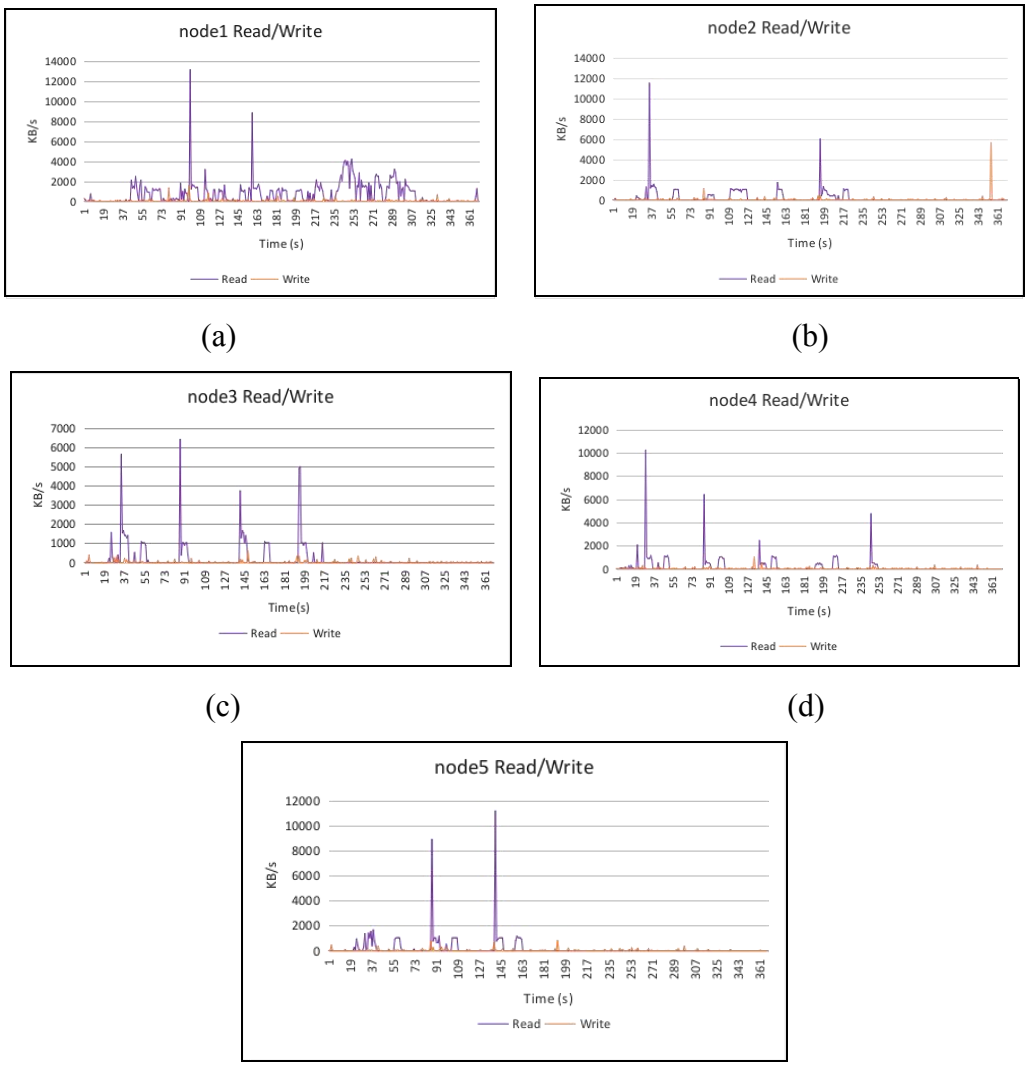
(a) (b)

(c) (d)

(e)

Gambar 4.4. Penggunaan Sumber Daya. (a) node1; (b) node2; (c) node3; (d) node4; (e) node5

Pada Gambar 4.4 dapat dilihat bahwa penggunaan CPU berkisar antara 80% hingga 100% saat aplikasi berjalan. Penggunaan CPU pada **node2** dan **node4** terlihat lebih rendah daripada *node* lain. Penggunaan memori meningkat seiring dengan berjalannya aplikasi dengan beberapa lonjakan penurunan. Penggunaan *input/output* paling aktif pada **node1**. Penggunaan *input/output* di *node* lain cukup berbeda dengan CPU dan memori. Terdapat lonjakan pada beberapa waktu, yang umumnya terjadi saat lonjakan penurunan penggunaan memori.



Gambar 4.5. Kecepatan *Read* dan *Write*. (a) node1; (b) node2; (c) node3; (d) node4; (e) node5

Dari Gambar 4.5 dapat dilihat bahwa kecepatan *read* terlihat rendah dengan beberapa lonjakan peningkatan pada beberapa waktu. Kecepatan *write* terlihat stabil di posisi rendah tanpa ada peningkatan yang signifikan.

BAB V
PENUTUP

5.1 Kesimpulan

Dari hasil penelitian mengenai Analisis Kinerja Hadoop pada *cluster* Raspberry Pi dapat diperoleh kesimpulan sebagai berikut:

a. Hadoop dapat dijalankan dengan lancar di *cluster* Raspberry Pi
b. Kemampuan skalabilitas *cluster* Hadoop dapat dibuktikan dengan peningkatan performa pada penambahan *node*.
c. Penambahan *node* pada *cluster* hanya meningkatkan performa secara signifikan pada 3 *node* pertama. Penambahan *node* lebih lanjut tetap meningkatkan performa namun tanpa perubahan yang signifikan, karena hukum Amdahl menyatakan performa pemrosesan paralel hanya meningkat secara signifikan hingga jumlah tertentu.
d. *Cluster* Hadoop menggunakan Raspberry Pi dapat menjalankan aplikasi MapReduce untuk pemrosesan data namun dengan 5 *node* belum dapat mengungguli laptop pada ukuran data 400 MB hingga 2 GB.
e. Pada ukuran data 4 GB *cluster* Hadoop dengan 5 buah Raspberry Pi dapat mengungguli laptop dalam hal kecepatan waktu eksekusi. Hal tersebut dapat memastikan bahwa *cluster* Hadoop lebih unggul dalam memproses *Big Data*.
f. Waktu eksekusi untuk *node* yang akan ditambahkan berikutnya dapat diprediksi dengan menggunakan fungsi **power** pada grafik skalabilitas.
g. Melalui prediksi dengan fungsi **power** diketahui bahwa dengan 6 *node cluster* dapat mengungguli laptop dalam hal kecepatan waktu eksekusi.
h. Hadoop dapat mentoleransi kerusakan pada *node* dengan baik dan menyelesaikan aplikasi yang dijalankan tanpa ada kehilangan data, namun dengan waktu yang lebih lama.
i. Persentase penggunaan sumber daya berupa CPU dan memori sangat tinggi saat aplikasi berjalan, sementara persentase *input/output* terbilang rendah.

5.1.1 Perbandingan *Cluster* dan Laptop

Dari hasil penelitian dapat dibuat perbandingan antara *cluster* dan laptop dengan harga setara. 6 *node cluster* digunakan untuk perbandingan karena dengan *node* sejumlah 6 buah *cluster* sudah dapat mengungguli laptop, sehingga terdapat gambaran keuntungan dan kerugian dengan menggunakan *cluster*. Tabel 5.1 menunjukkkan perbandingan *cluster* dan laptop yang digunakan.

Tabel 5.1. Perbandingan *Cluster* dan Laptop

Merk	***Cluster* @6node**	**Sony Vaio SVE1115EGB**
Sistem Operasi	Raspbian 4.4.50	Kali Linux 4.6.0
CPU	ARM Cortex-A53 1.2 GHz	AMD E2-1800 1.7 GHz
Jumlah Core	24	2
Memori	6 GB combined	6 GB
Penyimpanan Data	48 GB on HDFS	500 GB
Daya Listrik	77 Watt	39 Watt With Charger
Dimensi	16x10x13 cm	29x20x2,6 cm
Biaya	Rp. 4.100.000	Rp. 4.500.000

Dari hasil kesimpulan di atas dapat dirangkum bahwa *cluster* Hadoop dengan Raspberry Pi yang dibangun pada penelitian ini dapat menjalankan fungsi-fungsi dan fitur Hadoop dengan baik. Namun secara realistis, jika melihat perbandingan performa dan spesifikasi antara *cluster* dan laptop dengan biaya setara, *cluster* hanya unggul dalam hal kecepatan pemrosesan. Hal ini menjadikan *cluster* Raspberry Pi tidak efisien untuk menggantikan sistem dengan harga setara. Namun demikian *cluster* Raspberry Pi tetap dapat digunakan untuk media pembelajaran, terutama pada bidang komputasi dan pemrograman paralel.

5.2 Saran

Dalam penelitian ini terdapat keterbatasan terutama dalam hal sumber daya dan waktu, sehingga masih terdapat skenario lain yang belum dijalankan pada

penelitian *cluster* Hadoop ini. Maka dari itu penulis menyarankan beberapa hal untuk pengembangan lebih lanjut, diantaranya:

a. Penggunaan Raspberry Pi dengan jumlah lebih banyak.
b. *Overclocking* pada Raspberry Pi untuk meningkatkan performa.
c. Penggunaan sistem dengan performa lebih tinggi dari Raspberry Pi 3
d. Penggunaan media penyimpanan yang lebih besar untuk tiap *node*.
e. Perbandingan *cluster* dengan sistem selain laptop.

DAFTAR PUSTAKA

Agarwal, A. (2016, January 20). FrontPage - Hadoop Wiki. Retrieved April 14, 2017, from https://wiki.apache.org/hadoop

Baker, M., & Buyya, R. (1999). *Cluster* computing at a glance. *High Performance Cluster Computing*, *1*, 3–47.

Buyya, R. (1999). *High Performance Cluster Computing: Programming and applications*. Prentice Hall PTR.

Cox, S. J., Cox, J. T., Boardman, R. P., Johnston, S. J., Scott, M., & O'Brien, N. S. (2014). Iridis-pi: a low-cost, compact demonstration *cluster*. *Cluster Computing*, *17*(2), 349–358.

Dean, E., & Ghemawat, S. (2004). MapRednce: Simplified Data Processing on Large *Cluster*s.

deRoos, D., Zikopoulos, P. C., Melynk, R. B., Brown, B., & coss, R. (2014). *Hadoop For Dummies*. John Wiley & Sons.

Hajji, W., & Tso, F. (2016). Understanding the Performance of Low Power Raspberry Pi Cloud for Big Data. *Electronics*, *5*(2), 29. https://doi.org/10.3390/electronics5020029

Holmes, A. (2012). *Hadoop in practice*. Shelter Island, NY: Manning.

N.J. Schot. (2015). Feasibility of Raspberry Pi 2 based Micro Data Centers in Big Data Applications. *23rd Twente Student Conference on IT*, *23*.

ProjectDescription - Hadoop Wiki. (2014, January 13). Retrieved April 14, 2017, from https://wiki.apache.org/hadoop/ProjectDescription

Raspberry Pi 3 Model B - Raspberry Pi. (n.d.). Retrieved April 30, 2017, from https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

Raspberry Pi Downloads - Software for the Raspberry Pi. (n.d.). Retrieved April 30, 2017, from https://www.raspberrypi.org/downloads/

Raspberry Pi FAQs - Frequently Asked Questions. (n.d.). Retrieved April 30, 2017, from https://www.raspberrypi.org/help/faqs/

Raspberry Pi Hardware - Raspberry Pi Documentation. (n.d.). Retrieved April 30, 2017, from https://www.raspberrypi.org/documentation/hardware/raspberrypi/README.md

Raspberry Pi Products - Where to Buy Raspberry Pi. (n.d.). Retrieved April 30, 2017, from https://www.raspberrypi.org/products/

Richardson, M., & Wallace, S. (2012). *Getting Started with Raspberry Pi*. Maker Media, Inc.

Rusyadi, M., Marrowsi, R., & Wijaya, H. H. (2016). Analisa Hadoop *Cluster* Dengan Raspberry pi Model B+ dan Raspberry pi 2 Model B Studi Kasus Wordcount.

Shenoy, A. (2014). *Hadoop Explained*. Packt Publishing Ltd.

Sagiroglu, S., & Sinanc, D. (2013). Big data: A review. In Collaboration Technologies and Systems (CTS), 2013 International Conference on (pp. 42–47). IEEE.

Sterling, T. L. (Ed.). (2002). *Beowulf Cluster computing with Linux*. Cambridge, Mass: MIT Press.

Tso, F. P., White, D. R., Jouet, S., Singer, J., & Pezaros, D. P. (2013). The Glasgow Raspberry Pi Cloud: A Scale Model for Cloud Computing Infrastructures (pp. 108–112). IEEE.

Upton, E., & Halfacree, G. (2012). *Raspberry Pi user guide*. Chichester, West Sussex, UK: Wiley.

Welcome to Apache™ Hadoop®! (2017, March 28). Retrieved May 10, 2017, from http://hadoop.apache.org/

Wjaya, V. (2013, February 3). MapReduce: Besar dan Powerful, tapi Tidak Ribet. Retrieved April 22, 2017, from http://www.teknologi-bigdata.com/2013/02/mapreduce-besar-dan-powerful-tapi-tidak.html

WordCount - Hadoop Wiki. (2011, June 10). Retrieved April 23, 2017, from https://wiki.apache.org/hadoop/WordCount