KONFIGURASI PENGGABUNGAN HADOOP DISTRIBUTED FILE SYSTEM (HDFS) DENGAN CASSANDRA FILE SYSTEM (CFS) UNTUK PENYIMPANAN DATA BESAR

Tri Adi Putra Ramdani, *triadiputraramdani@gmail.com*
Eko Fajar Yanuwarsa, S.kom ekofajar@unmuhjember.ac.id
Triawan Adi Cahyanto M.Kom informatika.unmuhjember@gmail.com

Program Studi Teknik Informatika, Universitas Muhammadiyah Jember

ABSTRAK

Saat ini pertumbuhan data begitu cepat, dalam beberapa tahun saja jumlah data yang harus dikelola oleh perusahan-perusahaan IT terkemuka didunia bisa mencapai ukuran *Peta Byte*. Salah satu teknologi yang ditawarkan untuk menangani laju pertumbuhan data dengan media penyimpanan adalah *HDFS (Hadoop Distributed File System)*. *HDFS* menggunakan konsep blok-blok data dari sebuah file yang disimpan dalam beberapa mesin yang saling terhubung dalam sebuah cluster. Sedangkan *Cassandra File System (CFS)* adalah sebuah proses yang memudahkan dalam menjalankan analisis data yang besar secara cepat. *Cassandra File System (CFS)* dirancang untuk menangani jumlah yang sangat besar data yang tersebar di banyak server komoditas sekaligus memberikan layanan sangat tersedia tanpa titik tunggal kegagalan. Dari hasil menggabungkan dua sistem yang berbeda (*Hadoop-CFS*) ini menunjukkan bahwa mampu menampung data besar dan menganalisis data secara cepat.

Kata kunci : *Apache Hadoop, Comparing HDFS&CFS*

1.1 Latar Belakang

Hadoop Distributed file System (HDFS) adalah sebuah media penyimpanan data utama dengan kapasitas data besar yang memiliki kehandalan dalam perhitungan data yang sangat cepat. *Hadoop* menggunakan arsitektur *scale-out* yang menggunakan server komoditas yang di konfigurasikan sebagai klaster dan setiap server memiliki disk internal. Dalam segi data, *Hadoop* akan memecah tiap data menjadi blok-blok dan tersebar di seluruh claster. *Hadoop* juga memberikan struktur data dalam penyimpanan yang memfasilitasi dalam memberikan efisiensi biaya untuk menganalisis dan memproses sejumlah data pada *warehouse*.

Apache Cassandra adalah database NoSQL yang besarannya secara scalable. NoSQL sering digunakan untuk mengelola infrastruktur data yang penting sehingga *Cassandra* dikenal sebagai solusi profesional secara teknis saat user membutuhkan database NoSQL secara tepat waktu dalam kinerja yang tinggi pada skala besar yang tidak pernah turun. *Cassandra* memiliki sistem kerja *peer to peer* yang arsitekturnya didistribusikan seperti "cincin", yang mudah dalam melakukan *setup* dan pemeliharaan.

1.1 Perumusan Masalah

1. Bagaimanakah sistem kerja dari *Hadoop Distributed file System (HDFS)* dengan *Cassandra File System (CFS)* menggunakan jaringan peer to peer?
2. Bagaimanakah konfigurasi penggabungan *Hadoop Distributed file System (HDFS)* dengan *Cassandra File System (CFS)* dalam penyimpanan data besar (*big data*)?

1.2 Batasan Masalah

Beberapa batasan masalah dalam penelitian ini adalah sebagai berikut :

1. Platform *Hadoop CFS* yang digunakan di OS Linux Ubuntu
2. *Hadoop HDFS* hanya sebagai penyimpanan Data dan *CFS* sebagai sistem Pencarian.
3. Sebatas pembangunan konfigurasi dari *Hadoop Distributed file System (HDFS)* dengan *Cassandra File System (CFS)*.
4. Tidak menganalisa peforma dan sistem kerja aplikasi (sebatas konfigurasi dan visualisasi).

1.3 Tujuan Penelitian

1. Menganalisis proses konfigurasi dari penggabungan *HDFS* dan *CFS*.
2. Menyederhanakan *Overhead Operasional Hadoop* dengan menghilangkan titik tunggal kegagalan dalam *Hadoop Namenode*.

1.4 Manfaat Penelitian

1. Memberikan gambaran tentang bagaimana hasil dari proses

konfigurasi penggabungan *Hadoop Distributed file System (HDFS)* dengan *Cassandra File System (CFS)* dalam penyimpanan data besar (*Big Data*).

2. Memberikan gambaran tentang kemampuan menganalisis alur proses, manfaat dan perbandingan dari sistem *HDFS* dengan *CFS*.

2.1 Kajian Teori

2.1.1 Hadoop Apache

Big Data adalah sebuah data yang memilik skala penampungan data yang sangat besar dan banyak, tetapi sangat menguntungkan bagi perusahan. Kesulitan inilah yang berujung pada lamanya melakukan proses pengolahan dari big data yang membutuhkan resource yang cukup besar. Untuk itu munculnya Hadoop yang didampingi oleh Apache. Apache Hadoop adalah sebuah framework yang dibangun menggunakan bahasa java yang digunakan untuk komputasi dan pemrosesan dataset yang sangat besar secara terdistribusi. Hadoop terdiri dari 4 bagian yaitu :

1. Hadoop Common
2. Hadoop Distributed File System (HDFS)
3. Hadoop YARN
4. Hadoop Map Reduce

Selain itu, Apache juga memiliki hubungan dengan Hadoop yang lain seperti Hbase, Hive, Cassandra dan Mahout. Hadoop digunakan oleh perusahaan sebagai pengolah data secara distribusi seperti Yahoo, Facebook, dan Google. Hal ini yang membuat Hadoop bisa bekerja pada komputer dengan requarement yang cukup maksimal, sehingga bisa mengurangi biaya operasi di perusahaan.

2.1.2 Hadoop Distributed file System (HDFS)

Hadoop Distributed file System (HDFS) digunakan sebagai media penyimpanan file yang telah di bagi-bagi berdasarkan blok, dan blok-blok ini terdapat di lokasi yang berbeda-beda kemudian akan dilakukan replikasi data dengan mengurutkan blok yang mungkin tidak sama per node dan bisa bersifat single node atau multiple node.Dengan HDFS Claster yang terdiri dari NameNode utama sampai sebuah server master yang mengelola sistem file namespace dan juga mengatur akses ke data client. Pada HDFS menggunakan namespace file sistem yang memungkinkan data yang akan disimpan dalam file. Setiap file dibagi menjadi satu atau lebih blok yang kemudian akan membagi di satu set DataNode. NameNode bertanggung jawab untuk bertugas seperti membuka, mengubah nama, dan menutup file dari data direktori. Hal ini juga dapat menangani sistem kerja pada blok pemetaan untuk DataNode yang bertanggung jawab untuk mengelola permintaan I/O yang masuk dari client. DataNode menangani replikasi blok, penginputan dan penghapusan data ketika diperintahkan untuk oleh NameNode tersebut.

2.1.3 Apache Cassandra

Apache Cassandra adalah database NoSQL yang besarannya scalable. NoSQL digunakan pada saat ini oleh banyak bisnis modern untuk mengelola infrastruktur data yang penting. Apache Cassandra dikenal sebagai solusi profesional, teknisi mulai berpaling ketika mereka membutuhkan real time database NoSQL yang persediaan kinerja tinggi pada skala besar yang tidak mau turun.

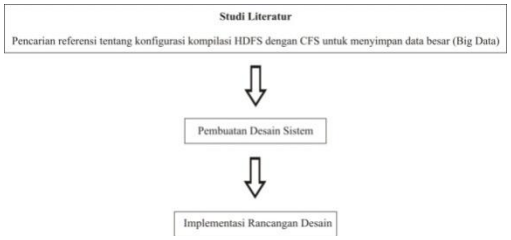
Alih-alih menggunakan master-slave atau kinerjanya secara manual dan sulit untuk dipertahankan dalam segi desain sharded, Cassandra memiliki sistem kerja peer to peer yang arsitekturnya didistribusikan seperti “cincin” yang jauh lebih elegan, mudah dalam melakukan setup dan pemeliharaan. Di Cassandra, semua node adalah sama, tidak ada konsep dari node master dengan semua node berkomunikasi dengan satu sama lain menggunakan protokol gosip.

3.1 Metode Penelitian

3.1.1 Tahapan Penelitian

Pelaksanaan tugas akhir ini dilakukan dengan metode dan sistematika sebagai berikut :

Metode penelitian yang digunakan bertujuan untuk mensinkronisasikan antara Hadoop Distributed file System (HDFS) dengan Cassandra File System (CFS) menggunakan jaringan peer to peer dengan monitoring dengan langkah – langkah yang dapat digambarkan seperti pada gambar 3.1

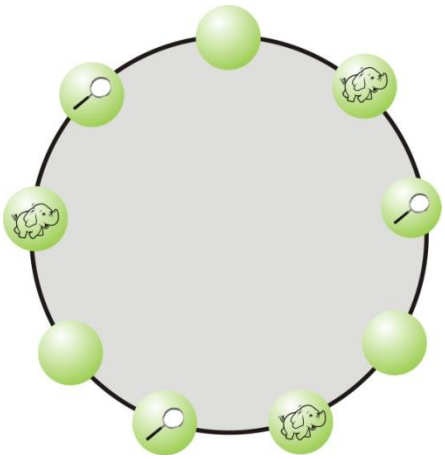


Gambar 3.1 Alur Metode Penelitian

3.1.2 Studi Literartur

Pada tahap studi literatur ini dilakukam proses pemilihan suatu masalah yang terbaru yang akan dilakukan sebagai tugas akhir. Selanjutnya diteruskan dengan pencarian referensi tentang konfigurasi penggabungan antara Hadoop Distributed file System (HDFS) dengan Cassandra File System (CFS) untuk penyimpanan data besar (Big Data) sebagai landasan dan penunjang dalam pengerjaan tugas akhir sekaligus sebagai solusi pemecah masalah yang di dihadapi. Tahapan terakhir dari studi pustaka ini adalah perumusan dan batasan yang di hadapi menjadi semakin jelas agar hasil yang diperoleh lebih maksimal.

3.1.3 Desain Sistem

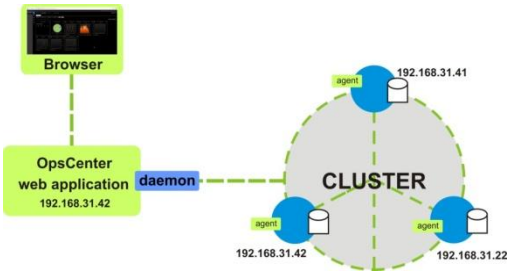


Gambar 3.2 Cluster Cassandra

Gambar di atas menjelaskan tentang **pemantauan Cassandra Cluster menggunakan OpsCenter** yang prosesnya dari konfigurasi penggabungan antara Hadoop Distributed file System (HDFS) dengan Cassandra File System (CFS). Gambar lingkaran besar menjelaskan bahwa dalam konfigurasi penggabungan ini menggunakan sistem jaringan peer to peer yang melingkar seperti cincin yang biasa disebut sebagai cincin cluster. Sedangkan 3 lingkaran kecil yaitu Hadoop Distributed file System (HDFS) dengan simbol gajah yang berfungsi sebagai menampung semua data besar (Big Data), Cassandra File System (CFS) dengan simbol Lup yang berfungsi sebagai sistem pencarian di data besar yang berada pada Hadoop Distributed file System (HDFS) dan lingkaran tanpa simbol berfungsi sebagai keyspace (ruang kunci), bertindak sebagai wadah untuk data, mirip dengan skema RDBMS. Hal ini menentukan sebagai parameter replikasi seperti faktor replikasi dan strategi penempatan replikasi. Dalam keyscape dapat memiliki satu atau lebih keluarga kolom. Hal ini mirip dengan tabel di RDBMS yang berisi beberapa kolom yang direferensikan oleh tombol baris. Maksudnya dari kolom adalah kenaikan data terkecil yang memiliki beberapa bagian seperti nama, nilai dan mendapatkan waktu. Untuk memulai OpsCenter dalam browser web, hanya pergi ke http://hostname:8888 dan kemudian masukkan ke alamat IP atau hostname dari node Cassandra.

3.1.4 Alur Proses Rancangan

Pada bagian ini akan dilakukan implementasi dari rancangan alur yang telah dibuat, mulai dari menyiapkan perangkat keras berupa laptop yang terinstal operating sistem Linux, software pendukung seperti Oracle Java 7, Root atau akses sudo, Python 2.6+ (diperlukan jika memasang OpsCenter) dan memori 256MB (hanya untuk mencoba beban kerja ringan). Jika menggunakan mesin virtual, pastikan untuk menggunakan alokasi memori yang direkomendasikan atau lebih untuk sistem operasi. Contoh kinerja dari konfigurasi penggabungan antara Hadoop Distributed file System (HDFS) dengan Cassandra File System (CFS)



Gambar 3.3 Alur Proses Rancangan

Gambar di atas adalah implementasi dari konfigurasi penggabungan antara Hadoop Distributed file System (HDFS) dengan Cassandra File System (CFS). Permasalahan yang saya angkat dari judul diatas yaitu memperoleh informasi tentang konfigurasi penggabungan antara Hadoop Distributed file System (HDFS) dengan Cassandra File System (CFS) untuk penyimpanan data besar (Big Data) secara otomatis dan real time.

4.1 Implementasi Pengujian Program

Berikut ini adalah hasil dari pengujian yang telah dilakukan oleh penulis yang tersusun sesuai dengan tahapan pengujian yang telah direncanakan.

4.1.1 PengujianMenjalankan Cassandra 2.0.11, Sysstat dan Ops-Center

5.1 di Ubuntu 12.04 LTS, berikut daftar node yang digunakan :

a. **Node 1 (192.168.31.42),**

b. **Node 2 (192.168.31.41),**

c. **Node 3 (192.168.31.22)**

Ada 2 proses tahapan dalam membangun program,sebagai berikut :

a. Menjalankan proses penggunaan Cassandra di node 1, node 2 dan node 3 sebagai berikut :

Di jendela terminal:

1. Memeriksa versi java yang terpasang dengan menjalankan perintah berikut:

   ```
   $java -version
   ```

   Disarankan untuk menggunakan versi terbaru dari Oracle Java 8 pada semua node.

2. Tambahkan repositori dari DataStax Komunitas

   ```
   ke /etc/apt/sources.l
   ist

   $deb
   http://debian.datastax.c
   om/community stable main
   ```

3. Tambahkan juga kunci repository DataStax untuk mendapatkan kunci akses ke cassandra

   ```
   $meringkuk -L
   http://debian.datastax.co
   m/debian/repo_key | sudo
   apt-key add -
   ```

4. Setelah selesai menambahkan repository sekarang install Cassandra:

   ```
   $sudo apt-get update

   $sudo apt-get install
   dsc20=2.0.11-1
   cassandra=2.0.11
   ```

a. DataStaxkomunitas distribusi cassandra siap untuk dikonfigurasikan. Source code di atas merupakan fungsi untuk instalasi Cassandra, tahap selanjutnya akan di lakukan instalasi ops-center beserta konfigurasi ke ops-center.

b. Menjalankan proses penggunaan Ops-Center di node 1 sebagai berikut :

Dijendela terminal :

1. Menginstal python-CGL dan dsc:

   ```
   $apt-get install
   python-CQL = 1.0.10-1

   $apt-get install dsc
   ```

2. Hentikan layanan Casssandra

   ```
   $service cassandra stop
   ```

3. Sekarang install DataStax OpsCenter dan prasyarat yang:

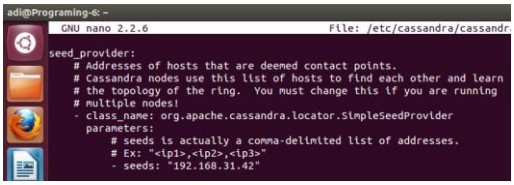
   ```
   $apt-get install
   libssl0.9.8

   $apt-get install
   opscenter free
   ```

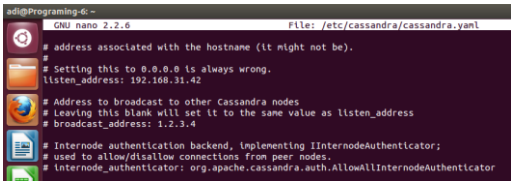
4. Pada tahap ini adalah konfigurasi Cassandra dengan Ops-Center di Node 1 (192.168.31.42), lakukan perintah berikut :

   ```
   $nano/etc/cassandra/cassa
   ndra.yaml
   ```

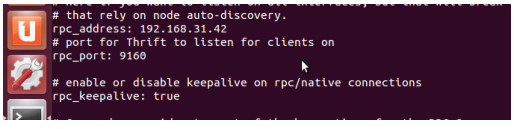
setting ipv4 di bagian seeds, listen_address dan rcp_address kemudian simpan.



4.1 setting ipv4 di seeds



4.2 setting ipv4 di listen _address



4.3 Setting ipv4 di rcp_address

5. Selanjutnya mengedit opscenterd.conf untuk mengubah tampilan pada alamat

IP, lakukan perintah berikut :
`$nano/etc/opscenter/opscenterd.conf`

Setting pada interface dengan ipv4 pada node 1 (192.168.31.42)

```
# opscenterd.conf

[webserver]
port = 8888
interface = 172.0.2.1

[logging]
# level may be TRACE, DEBUG, INFO, WARN, or ERROR
level = INFO
[authentication]
# if this file does not exist, there will be no password protection. Use the
# set_passwd.py tool (included with OpsCenter) to set passwords.
#passwd_file = /etc/opscenter/.passwd
```

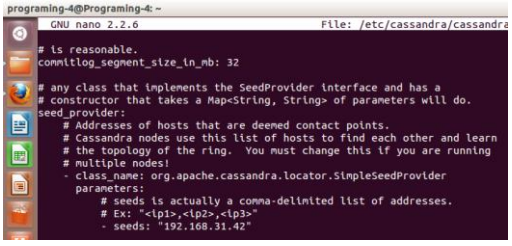
6. Sekarang jalankan Cassandra, dengan perintah :

`$service cassandra start`

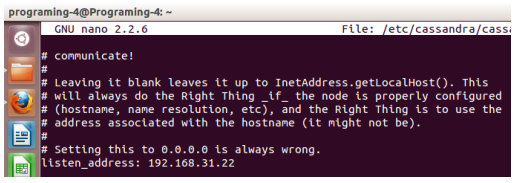
7. Node 1 telah selesai disetting, selanjutnya setting node ke 2 dan node 3 dengan perintah :

`$nano/etc/cassandra/cassandra.yaml`

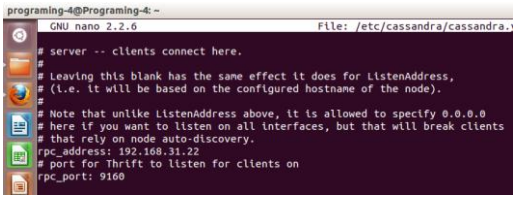
setting ipv4 di bagian seeds, listen_address dan rcp_address dengan ipv4 sesuai node yang digunakan, dalam tahap ini setting node ke 2 (192.168.31.41) dan node 3 (192.168.31.22) kemudian di bagian seeds diganti dengan ipv4 pada node 1 (192.168.31.42) kemudian simpan.



Gambar 4.4 setting ipv4 node 1 di seeds



4.5 setting ipv4 node 3 di listen _address



4.6 setting ipv4 node 3di rcp _address

8. Sekarang jalankan Cassandra, dengan perintah :

   $ `service cassandra start`

9. Selanjutnya jalankan konfigurasinya dan pastikan cincin di node 1, node 2 dan node 3 bekerja seperti yang diharapkan dengan perintah :

`$ nodetool cincin localhost h`

10. Sebelum menghubungkan ke browser kita menginstall sysstat tujuannya menangani I / O dalam kecepatan transfer, aktivitas paging, proses-kegiatan terkait, interupsi, aktivitas jaringan, memori dan pemanfaatan ruang swap, penggunaan CPU, kernel kegiatan dan statistik TTY di semua node.

    a. Tahap 1. Install sysstat :
       `$sudo apt-get install sysstat`

    b. Tahap 2. Aktifkan Stat Koleksi
       `$sudo nano/etc/default/sysstat`
       ubah ENABLED=”false” menjadi ENABLED=”true”
       dan simpan file

    c. Tahap 3.Ubah interval koleksi dari setiap 10 menit atau setiap 2 menit.
       $`sudo nano /etc/cron.d/sysstat`
       Ubah :
       5-55/10 * * * * root command -v debian-sa1 > /dev/null

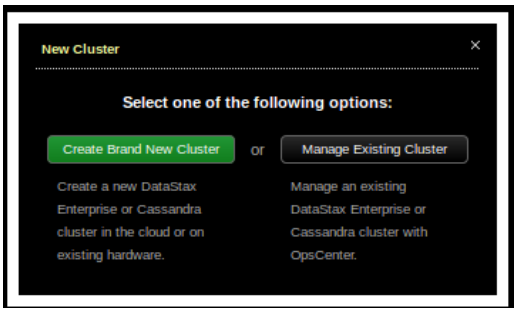
&& debian-sa1 1 1
To
*/2 * * * * root command -
v debian-sa1 > /dev/null &&
debian-sa1 1 1
dan simpan file

d. Tahap 4. Restart sysstat
```
$sudo service sysstat restart
```

e. Step 5. Jika ingin mengetahui semua statistik:
```
$sar -A
```

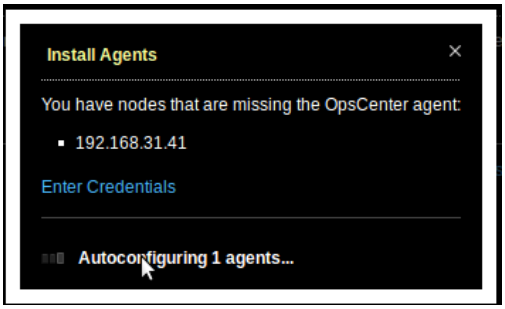
f. Step 6. Jika ingin menyimpan statistik untuk analisis lebih lanjut untuk penggunaan berkas :
```
$sudo sar -A > $(date +`hostname`-%d-%m-%y-%H%M.log)
```

2. Kemudian ke browser dan masukkan perintah http://192.168.31.42:8888, browser yang digunakan yaitu google chrome karena support di tampilan Ops-Center.



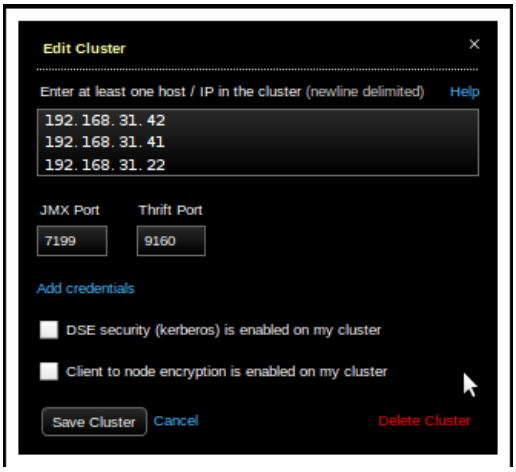
Gambar 4.7 Membuat cluster baru

3. Pada tampilan OpsCenter, menunjukkan ada tiga node aktif tetapi membutuhkan penggunaan *agent* pada setiap node. Jadi, pilih *fix* untuk menggunakan *agent*



Gambar 4.8 Menginstal agent

4. Masukkan kredensial untuk menginstal *agent* dan menerima sidik jari. Kemudian, pada dashboard OpsCenter akan melihat bahwa semua *agent* terhubung:



Gambar 4.9 Node terhubung

5. Bila semua sudah terkonfigurasi, berikut tampilan dashboard Ops-Center 5.1.2

Gambar 4.10 Tampilan Ops-Center

5.1 Kesimpulan

Dari keseluruhan permasalahan dan pembahasan dalam skripsi ini, penyusun dapat mengambil kesimpulan sebagai berikut :

1. Dengan adanya program ini diharapkan mampu menangani masalah tentang data dengan dengan cepat.
2. Dengan adanya program ini diarapkan mampu memberikan inormasi tentang keunggulan HDFS dan CFS.

5.2 Saran

Adapun saran-sarang yang berkaitan dengan pengembangan dari program yang telah penyusun buat, yaitu :

1. Perlu adanya pembahasan secara detail tentang performa program dan penginputan data besar secara real
2. Perlu pengembangan lebih lanjut dari sisi pemakaian cluster selain menggunakan linux (windows, CentOS dll).

DAFTAR PUSTAKA

[i] Devopa Mittra, 2014, Hadoop Apache, Amerika Serikat, http://hadoop.apache.org/ (diakses 13 Juli 2014)

[ii] Devopam Mittra, 2014, Arsitektur Hadoop HDFS, Amerika Serikat, http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html. (diakses 13 Juli 2014)

[iii] Devopam Mittra, 2014, Hadoop Apache, Amerika Serikat, http://cassandra.apache.org/ (diakses 13 Juli 2014)

[iv] DevopamMittra, 2014, Hadoop MapReduce, Amerika Serikat, http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html#Overview (diakses 13 Juli 2014)

[v] Horton, 2011, Apache Hive, Amerika Serikat, http://hortonworks.com/hadoop/hive/ (diakses 13 Juli 2014)

[vi] Nareswara, 2011, Apa itu NoSQL Database ?, http://nareswara.com/2011/07/06/apa-itu-nosql-database/ (diakses 13 Juli 2014)

[vii] Prado, 2012, How to run Apache Cassandra database on a two nodes Ubuntu 12.04 LTS cluster, http://www.prado.it/2012/05/04/how-to-run-apache-cassandra-database-on-a-two-nodes-ubuntu-12-04-lts-cluster/ (diakses 15 Mei 2015)

[viii] Per Christensson, 2005, Definisi Peer to Peer (P2P), http://www.techterms.com/definition/p2p (diakses 09 September 2014)

[ix]Pestra cristin, 12 Oktober 2011, Cassandra Query Language (CQL) v2.0

, https://cassandra.apache.org/doc/cql/CQL.html (diakses 15 Mei 2015)