



Big Data

Pengantar Hadoop

Outline

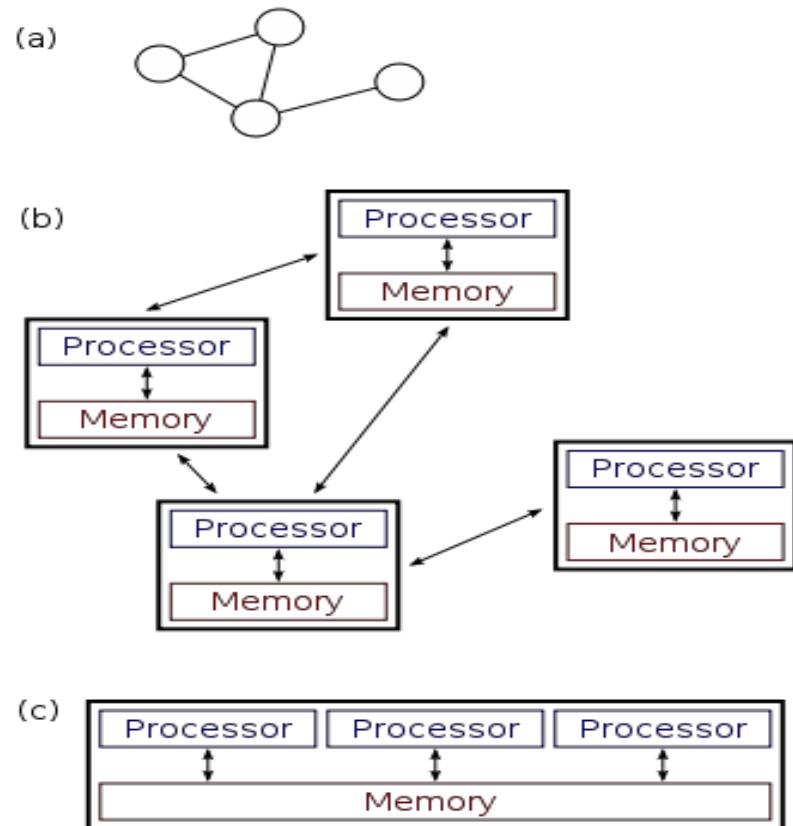
- Pengantar Hadoop
- Hadoop Distributed File System (HDFS)
 - Arsitektur Hadoop HDFS: Name Node, Data Node
 - Mengakses HDFS Melalui Command Line
- Hadoop MapReduce (MR)

Kita Butuh Sistem dengan Skala Besar

- Kita menghasilkan terlalu banyak data untuk diproses dengan alat bantu tradisional
- Dua masalah penting yang harus diatasi:
 - Bagaimana kita bisa menyimpan data yang sangat besar dengan cara yang handal dan biaya yang masuk akal?
 - Bagaimana kita bisa menganalisis data yang sudah disimpan?

Konsep Pengolahan Big Data

- Bayangkan ada sekumpulan data yang sangat besar (Big Data), bagaimana kita dapat melakukan Parallel atau Distributed Processing.
- File Sistem Terdistribusi (**Distributed File System**, atau disingkat dengan **DFS**) adalah **file system yang mendukung sharing files dan resources** dalam bentuk penyimpanan persistent (tetap) untuk tujuan tertentu di sebuah network

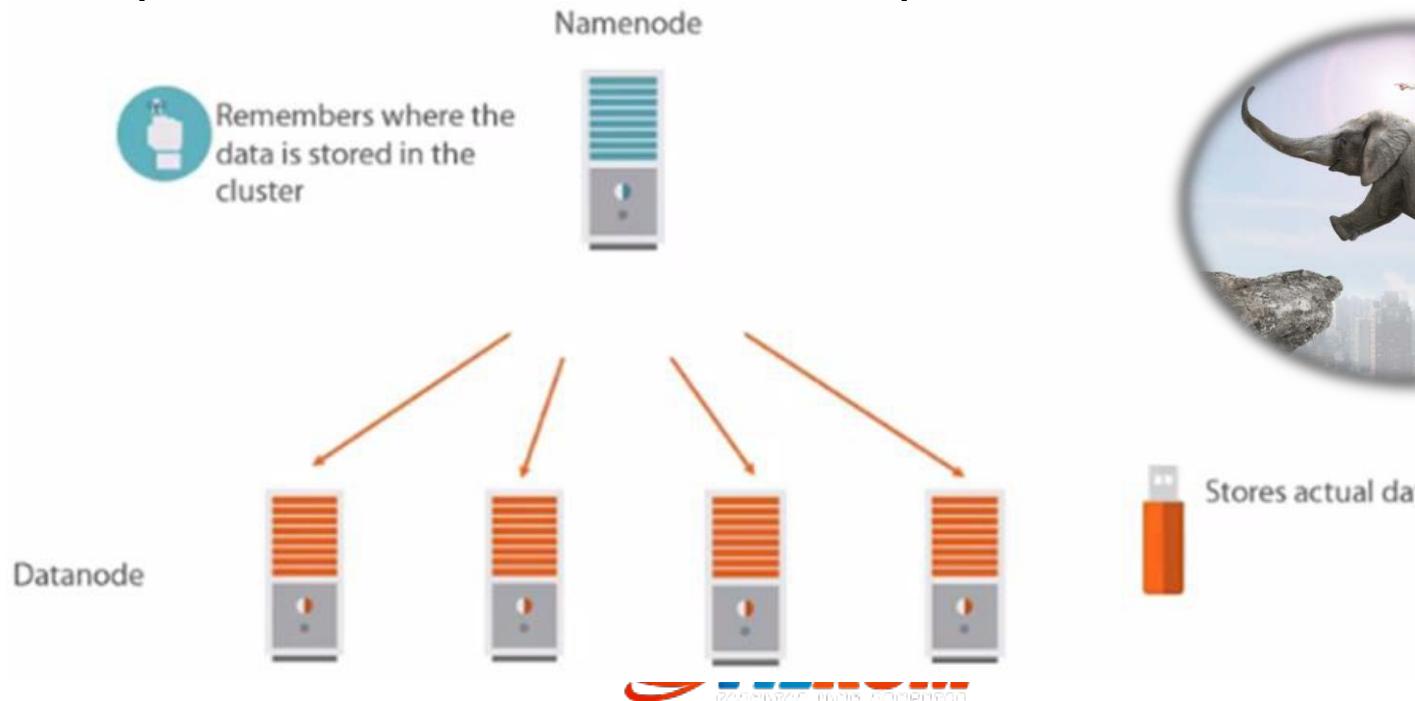


(a), (b): distributed system.
(c): parallel system.

Hadoop



- Apache Hadoop
 - Framework/sistem yang *scalable*, digunakan untuk penyimpanan data besar secara intensif dan pemrosesan/komputasi yang cepat dan ekonomis
 - Open source di bawah lisensi Apache



Sejarah Singkat



- Dimulai dari penelitian berawal di Google tahun 2003
 - Tujuan Google untuk mengindeks seluruh World Wide Web
 - Google sudah mencapai batas dari skalabilitas di teknologi RDBMS
 - Pendekatan baru untuk menyimpan dan menganalisis data sangat besar
 - Google File System (Ghemawat, et al, 2003)
 - MapReduce: Simplified Data Processing on Large Clusters (Dean and Ghemawat, 2008)
- Developer bernama Doug Cutting (Yahoo!) berusaha untuk mengatasi masalah yang sama pada implementasi search engine open source buatannya, Nutch
 - Memulai project open source berdasarkan penelitian Google dan membuat Hadoop 2005
 - Hadoop diberi nama dari nama mainan gajah anaknya

Hadoop

- **Hadoop Distributed File System** dikembangkan menggunakan desain sistem file yang terdistribusi.
 - Tidak seperti sistem terdistribusi, HDFS sangat fault tolerant dan dirancang menggunakan hardware low-cost.
- Dalam arti lain, **Hadoop** adalah *software platform* (platform perangkat lunak) sebagai *analytic engine* yang memungkinkan seseorang dengan mudah untuk melakukan pembuatan penulisan perintah (*write*) dan menjalankan (*run*) aplikasi yang memproses data dalam jumlah besar

Hadoop

- *Inti* dari Hadoop terdiri dari 2 komponen utama:
 - **Penyimpanan:** **Hadoop Distributed File System (HDFS)**
 - Framework untuk mendistribusikan data ke cluster supaya bisa *scalable* dan cepat
 - **Pemrosesan:** **MapReduce**
 - Framework untuk memproses data dengan cara yang handal (*reliable*) dan cepat
 - + infrastruktur untuk bekerja, termasuk:
 - Filesystem dan bantuan administrasi
 - Job scheduling dan monitoring

Apa itu Hadoop?

- Dalam komputasi, **platform** menggambarkan semacam (*hardware architecture*) arsitektur perangkat keras atau (*software framework*) kerangka kerja perangkat lunak (termasuk kerangka kerja aplikasi), yang memungkinkan perangkat lunak dapat berjalan.
- **Ciri khas dari platform** meliputi arsitekturnya komputer, sistem operasi, bahasa pemrograman dan runtime libraries atau GUI yang terkait.
- Apa yang ada pada Hadoop dari sudut pandang:
 - Platform: Komputer sebagai node, .. ?
 - Framework: HDFS Explorer, .. ?

Hadoop Distributed File System (HDFS)

- HDFS sebagai direktori di komputer di mana data Hadoop disimpan.
- Untuk pertama kalinya, direktori ini akan di “format” agar dapat bekerja sesuai spesifikasi dari Hadoop.

Hadoop Distributed File System (HDFS)

- Data di Hadoop disimpan dalam ***cluster***.
 - Cluster biasanya terdiri dari banyak node atau komputer/server.
 - Setiap node di dalam cluster ini harus terinstall Hadoop untuk bisa jalan.
- Hadoop versi 1.x ada beberapa jenis node di dalam ***cluster***:
 - **Name Node:**
 - node utama yang mengatur penempatan data di cluster, menerima job dan program untuk melakukan pengolahan dan analisis data misal melalui Map Reduce.
 - menyimpan metadata tempat data di cluster dan juga replikasi data tersebut.
 - **Data Node:**
 - node tempat data ditempatkan.
 - Satu block di HDFS/datanode adalah 64 MB.
 - Sebaiknya data yang disimpan di HDFS ukurannya minimal 64 MB untuk memaksimalkan kapasitas penyimpanan di HDFS.

Hadoop Distributed File System (HDFS)

- Hadoop versi 1.x ada beberapa jenis node di dalam *cluster*:
 - **Secondary Name Node:**
 - Bertugas untuk menyimpan informasi penyimpanan data dan pengolahan data yang ada di name node.
 - Fungsinya jika name node mati dan diganti dengan name node baru maka name node baru bisa langsung bekerja dengan mengambil data dari secondary name node.
 - **Checkpoint Node dan Backup Node:**
 - **Checkpoint node** melakukan pengecekan setiap interval waktu tertentu dan mengambil data dari name node.
 - Dengan check point node maka semua operasi perubahan pada data terekam. Namun, secondary name node hanya perlu menyimpan check point terakhir.
 - **Backup Node** juga berfungsi sama, hanya bedanya data perubahan yang disimpan dilakukan di memory bukan di file seperti checkpoint dan secondary node.
- Kelemahan HDFS di Hadoop versi 1.x adalah jika *name node* mati. Maka seluruh cluster tidak bisa digunakan sampai name node baru dipasang di cluster.

Hadoop Distributed File System (HDFS)

- Hadoop versi 2.x ada beberapa jenis node di dalam cluster:
 - **Lebih dari satu name nodes.**
 - Hal ini berfungsi sebagai implementasi dari High Availability.
 - Hanya ada satu name node yang berjalan di cluster (aktif) sedangkan yang lain dalam kondisi pasif. Jika name node yang aktif mati/rusak, maka name node yang pasif langsung menjadi aktif dan mengambil alih tugas sebagai name node.
 - **Secondary name node, checkpoint node** dan **backup node** tidak lagi diperlukan.
 - Meskipun ketiga jenis node tersebut menjadi optional, tetapi kebanyakan tidak lagi ada di cluster yang memakai Hadoop versi 2.x.
 - Hal ini karena selain fungsi yang redundan, juga lebih baik mengalokasikan node untuk membuat tambahan name node sehingga tingkat High Availability lebih tinggi.
 - **Data node** tidak ada perubahan yang signifikan di versi hadoop 2.x dari versi sebelumnya.

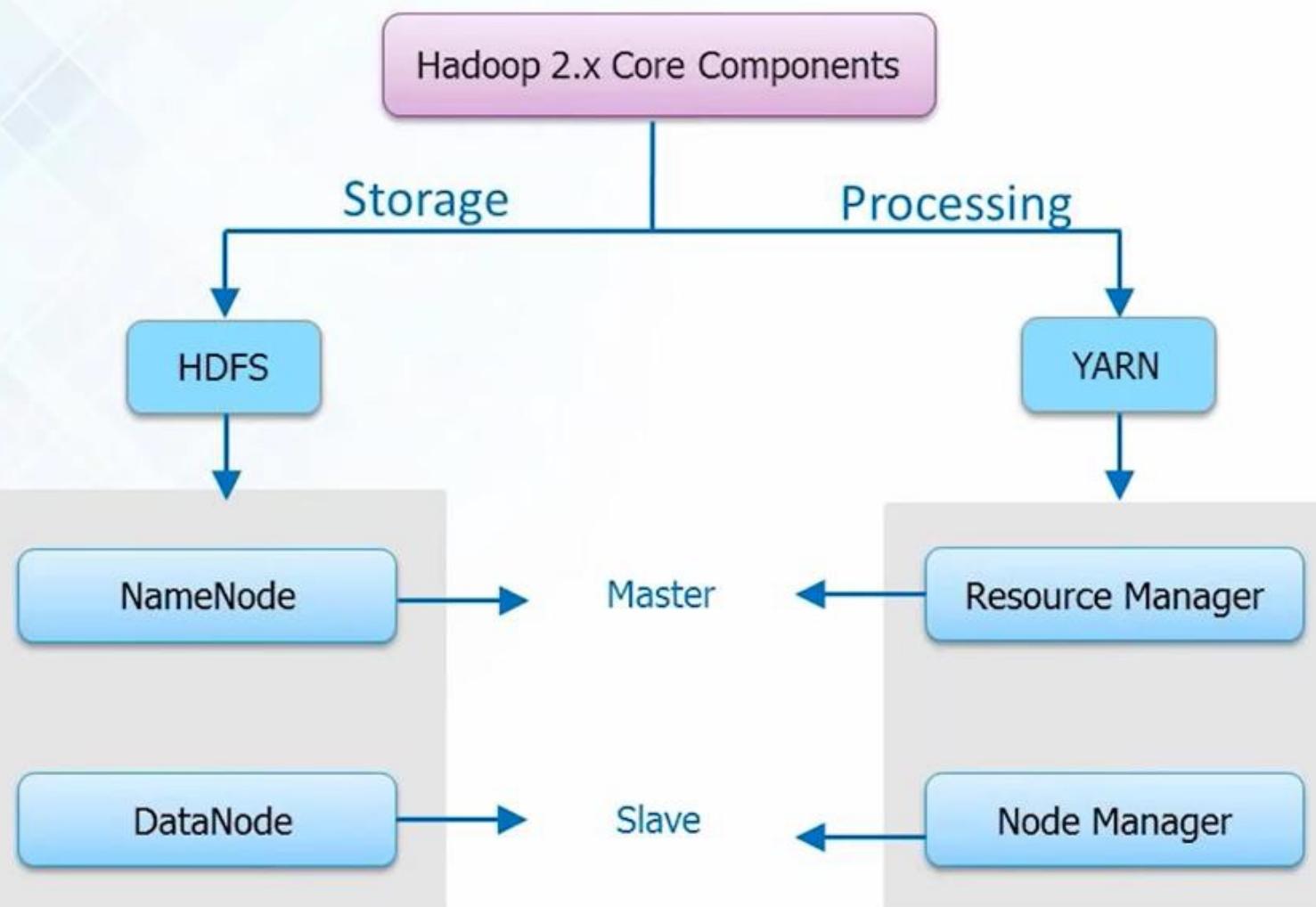
Hadoop Distributed File System (HDFS)

- Meskipun konteks yang dibicarakan adalah dalam cluster, Hadoop juga bisa dijalankan dalam single node
- Dalam single node maka semua peran di atas berada dalam satu komputer
- Biasanya single node ini digunakan hanya untuk training atau development. Bukan untuk produksi skala enterprise

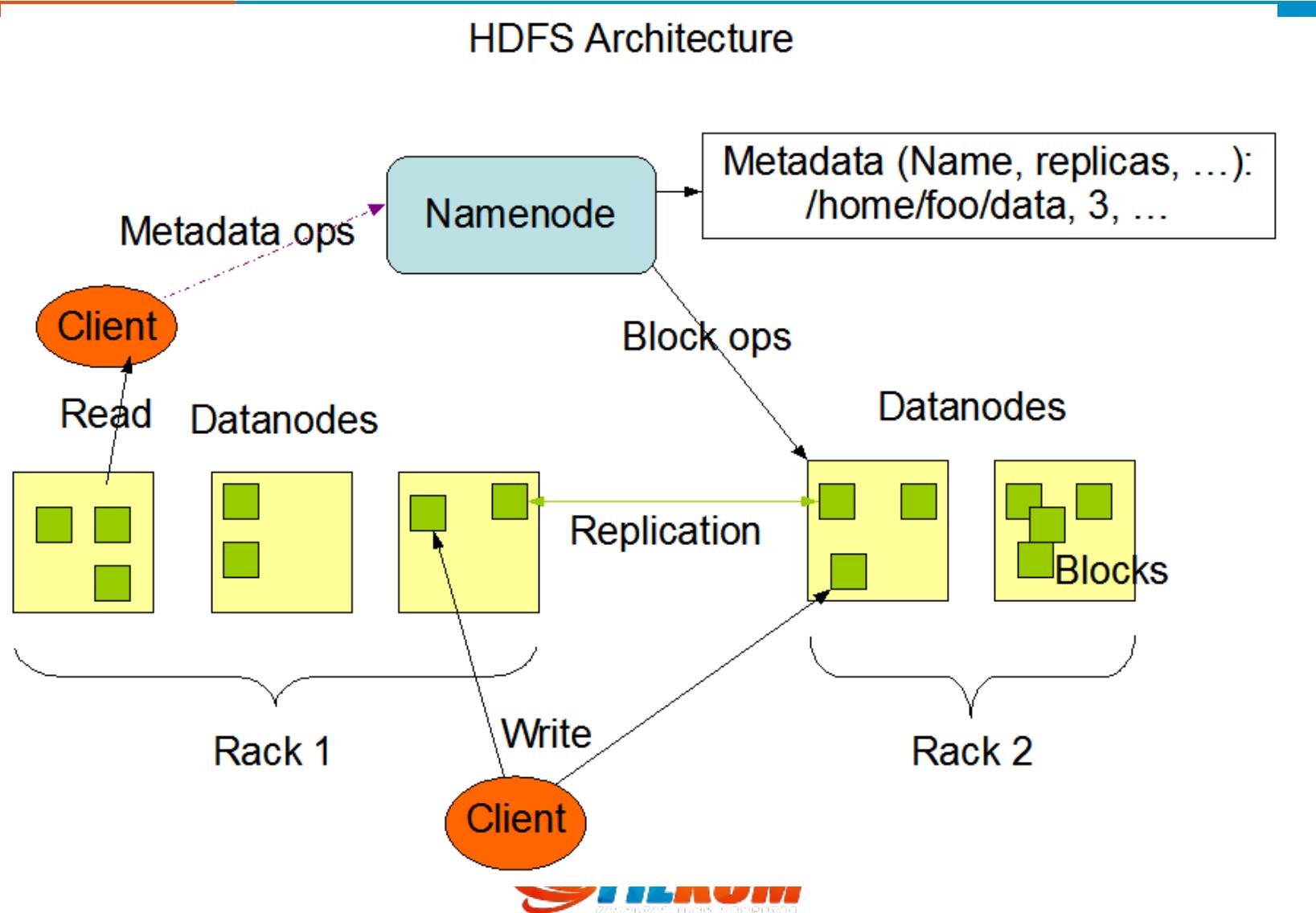
HDFS dan Unix File System

- Dalam beberapa hal, HDFS mirip dengan UNIX *filesystem*
- Hierarkis, dengan UNIX/style paths
 - misal. /sales/reports/asia.txt
- *File ownership* dan *permissions* mirip UNIX
- Ada beberapa perbedaan mayor dari UNIX
 - Tidak ada konsep current directory
 - Tidak bisa memodifikasi file setelah dituliskan
 - Bisa menghapus dan membuat ulang, tetapi tidak bisa memodifikasi
 - Harus menggunakan utilities dari Hadoop atau kode buatan sendiri untuk mengakses HDFS

Arsitektur HDFS



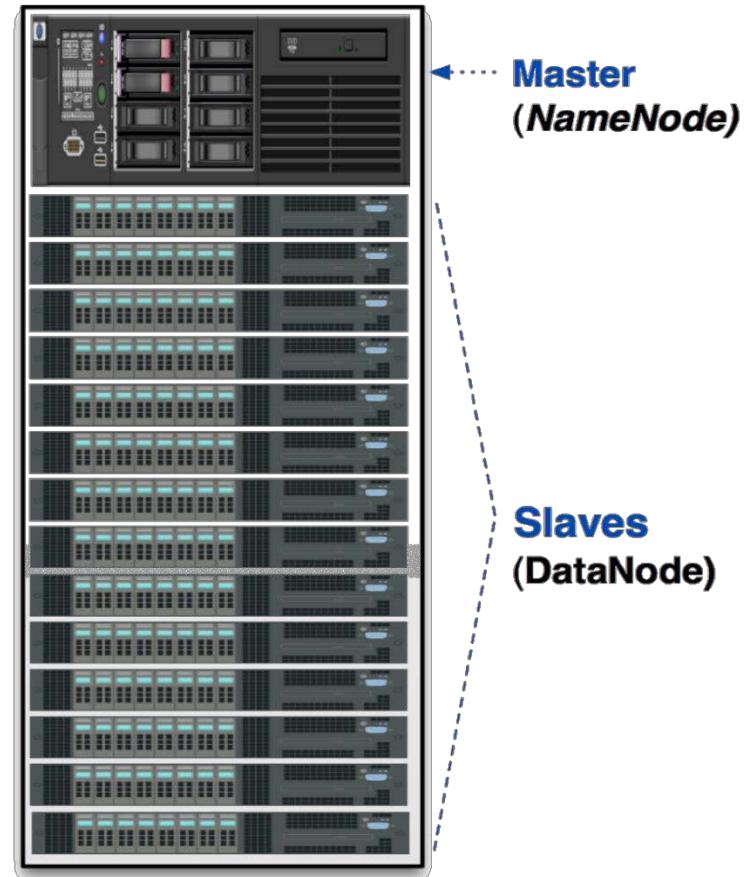
Arsitektur HDFS



Arsitektur HDFS

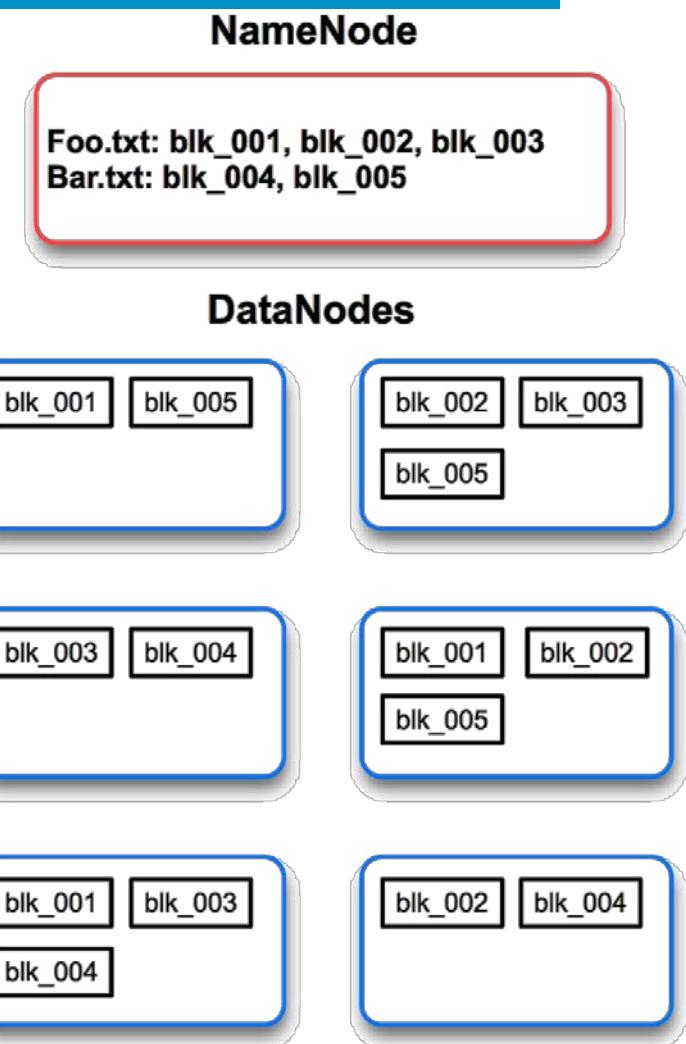
- Hadoop memiliki arsitektur **master/slave**
- HDFS master daemon: **Name Node**
 - Mengatur namespace (pemetaan file ke block) dan metada (pemetaan block ke mesin/komputer)
- HDFS slave daemon: **Data Node**
 - Membaca dan menulis data sebenarnya
 - Dapat berjalan di atas filesystem sebenarnya (ext3/4, NTFS, dll)

A Small Hadoop Cluster



Arsitektur HDFS

- Contoh:
- NameNode berisi metadata untuk dua file
 - Foo.txt (300MB) dan Bar.txt (200MB)
 - Dimisalkan HDFS dikonfigurasikan dengan block sebesar 128MB
- DataNodes berisi block data sebenarnya
 - Tiap block berukuran 128MB
 - Tiap block direplikasi 3x pada cluster
 - Laporan block dikirimkan ke NameNode secara periodik



Mengakses HDFS Melalui Command Line

- HDFS bukan general purpose file system
 - File HDFS tidak dapat diakses langsung oleh sistem operasi host
 - End user mengakses HDFS melalui perintah **hadoop fs**
- Contoh
 - Menampilkan isi dari file /user/fred/sales.txt

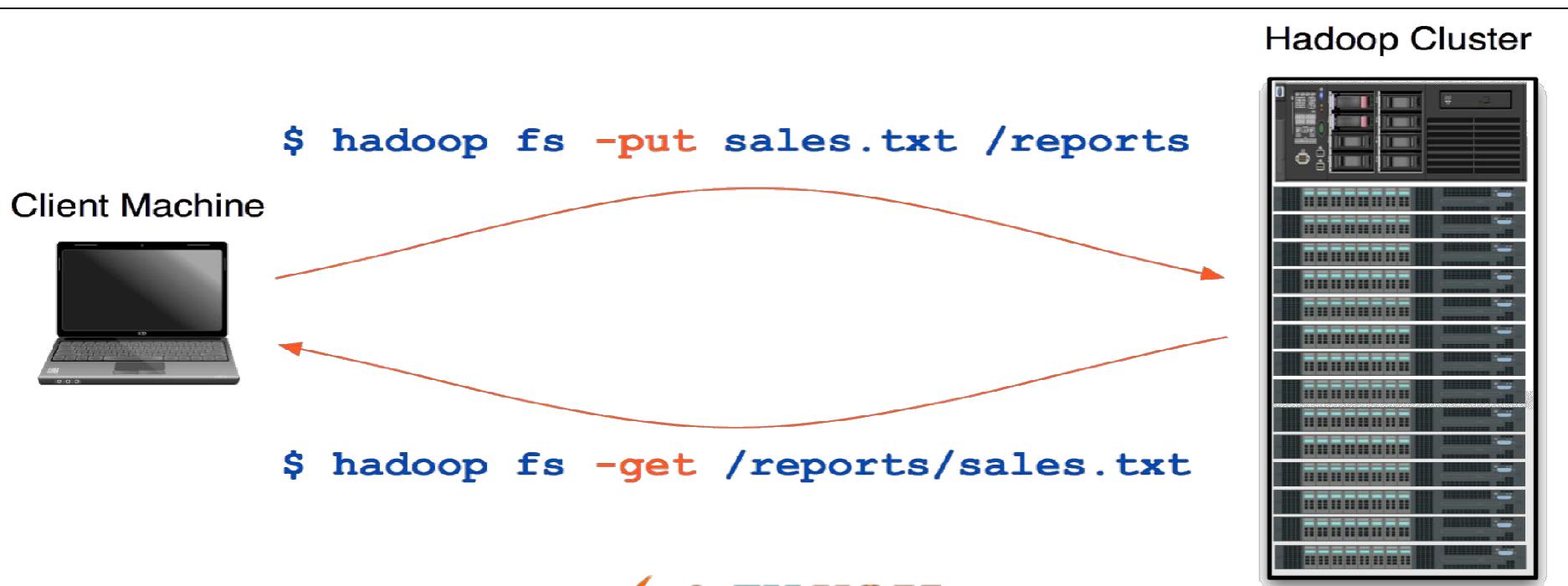
```
hadoop fs -cat /user/fred/sales.txt
```

- Membuat directory baru (di dalam root) dengan nama reports

```
hadoop fs -mkdir /reports
```

Mengakses HDFS Melalui Command Line

- Ingat, HDFS terpisah dari filesystem lokal milik OS
 - Mengopi file lokal ke HDFS: **hadoop fs -put**
 - Mengopi file HDFS ke lokal: **hadoop fs -get**



Mengakses HDFS Melalui Command Line

- Melihat daftar isi di dalam directory root

```
hadoop fs -ls /
```

- Menghapus file /reports/sales.txt

```
hadoop fs -rm /reports/sales.txt
```

- Perintah-perintah lainnya silakan dibaca sendiri

Pengantar, Paradigma Map Reduce (MR)

- Setelah mengetahui bagaimana Hadoop menyimpan data, maka berikutnya penting untuk memahami bagaimana Hadoop memproses data
- Pemrosesan data di Hadoop menggunakan **MapReduce**
- **MR** bukanlah suatu bahasa, tapi model pemrograman
 - Awalnya dikembangkan oleh Google
- MapReduce terdiri dari dua fungsi: **map** dan **reduce**
 - MR termasuk gaya pemrograman fungsional yang secara alami dapat diparalelkan di sekelompok besar workstation/PC
 - *Sortir/merge* berbasis komputasi terdistribusi
- Sistem yang mendasari untuk melakukan partisi dari input data membuat jadwal eksekusi program di beberapa mesin, handling kegagalan mesin, dan me-manage komunikasi yang diperlukan antar- mesin. (Ini adalah kunci untuk sukses Hadoop).

Apa itu Map dan Reduce?

- Fungsi **map** selalu berjalan lebih dulu
 - Biasanya digunakan untuk “memecah” data
 - Memfilter, metransformasi, *parse* data.
 - Misal parse simbol saham (*stock*), harga, dan waktu dari data feed
 - Output dari fungsi map akan menjadi input dari fungsi reduce
- Fungsi **reduce**
 - Biasanya digunakan untuk menggabungkan (*aggregate*) data dari fungsi map. Misal. Menghitung harga rata-rata per-jam dari saham
 - Tidak selalu diperlakukan jadi bisa bersifat optional
 - Kadang bisa juga melakukan job “map-only”
- Antara dua task ini ada fase tersembunyi dikenal dengan “Shuffle dan Sort”
 - Mengatur output map untuk dikirimkan ke reducer

Contoh MapReduce

- Kode MapReduce biasanya ditulis dalam bahasa Java
 - Tapi dimungkinkan menggunakan bahasa apapun melalui Hadoop Streaming
- Contoh job pada MapReduce:
 - **Input:** file teks berisi order id, employee name, dan sale amount
 - **Output:** jumlah seluruh sales per employee

Job Input

0	Alice	3625
1	Bob	5174
2	Alice	893
3	Alice	2139
4	Diana	3581
5	Carlos	1039
6	Bob	4823
7	Alice	5834
8	Carlos	392
9	Diana	1804

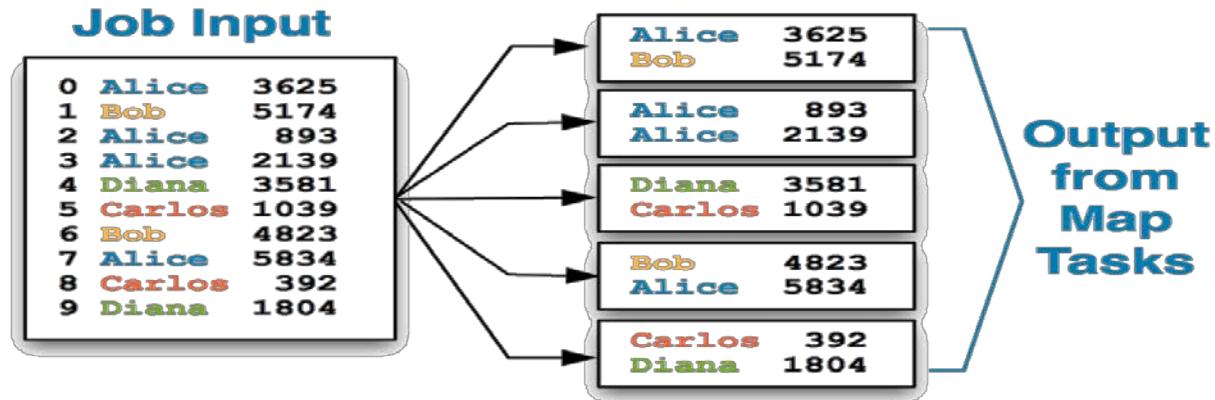
Job Output

Alice	12491
Bob	1431
Carlos	9997
Diana	5385

Penjelasan Fungsi Map

- Hadoop memecah job menjadi map tasks secara individual
 - Jumlah map tasks ditentukan oleh banyaknya data input
 - Tiap map task menerima sebagian dari keseluruhan job input untuk diproses secara paralel
 - **Mapper** memproses satu input record dalam satu waktu
 - Untuk tiap input record, dihasilkan nol atau lebih record sebagai output
- Dalam kasus ini, 5 map tasks masing-masing mem-parse sebagian dari input record
 - Misal output-nya adalah employee name dan sales

Mapper melakukan tugas untuk mengekstraksi a2 field dari 3



Shuffle dan Sort

- Hadoop secara otomatis melakukan sort dan merge output dari seluruh map tasks
 - Mengurutkan record berdasarkan key (misal nama)
 - Proses antara (transparent) ini dikenal sebagai “Shuffle and Sort”
 - Hasilnya digunakan untuk reduce task

Map Task #1 Output

Alice	3625
Bob	5174

Map Task #2 Output

Alice	893
Alice	2139

Map Task #3 Output

Diana	3581
Carlos	1039

Map Task #4 Output

Bob	4823
Alice	5834

Map Task #5 Output

Carlos	392
Diana	1804

Input to Reduce Task #1

Alice	3625
Alice	893
Alice	2139
Alice	5834
Carlos	1039
Carlos	392

Input to Reduce Task #2

Bob	5174
Bob	4823
Diana	3581
Diana	1804

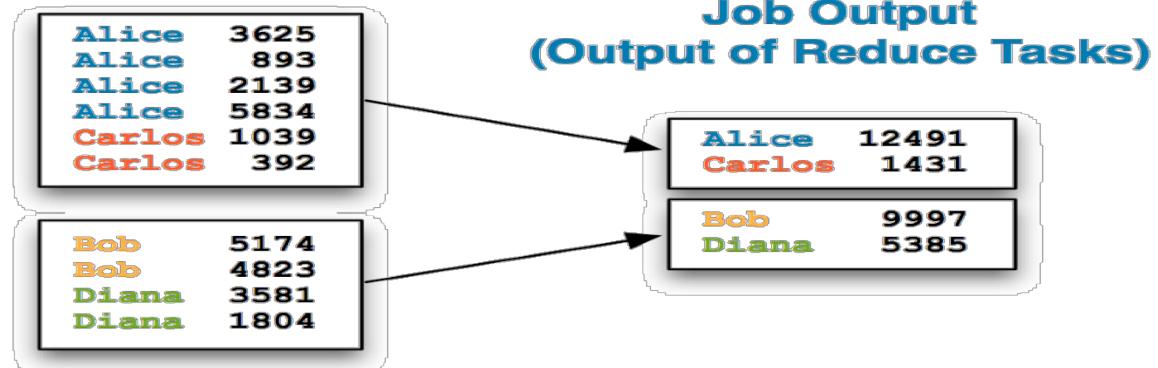
Penjelasan Fungsi Reduce

- Input pada reducer berasal dari shuffle and sort
 - Seperti map, fungsi reduce menerima satu record satu waktu
 - Suatu reducer menerima semua record dari suatu key
 - Untuk tiap input record, reduce dapat menghasilkan nol atau lebih output record
- Dalam kasus ini, fungsi reduce menjumlahkan total sales per employee
 - Menghasilkan employee name (key) dan total sales sebagai output dalam bentuk sebuah file
 - Tiap file output reducer dituliskan ke directory job tertentu di HDFS

Input to Reduce Task #1

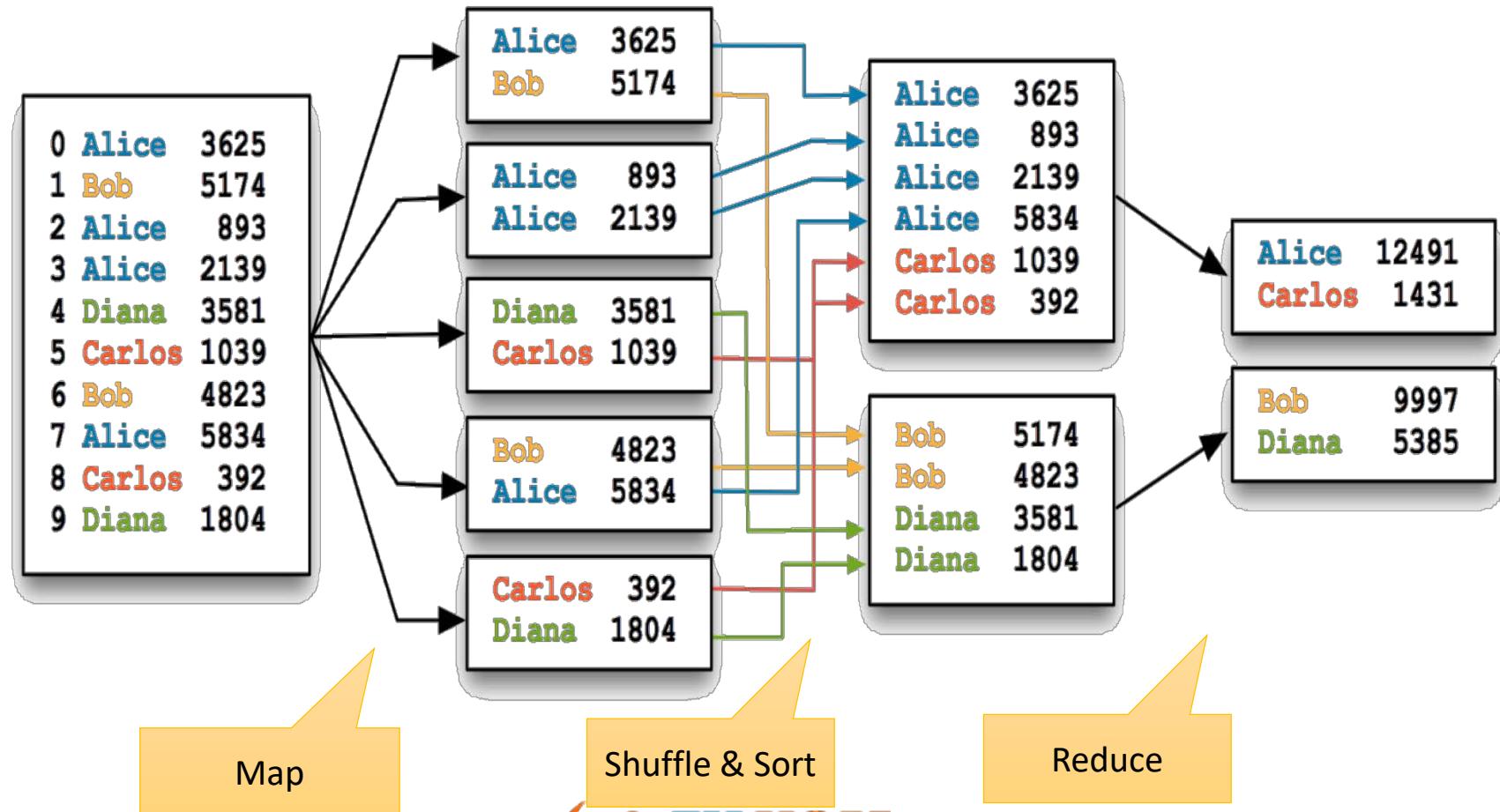
Reducer menjumlahkan sales

Input to Reduce Task #2



Alur dari Semua Proses

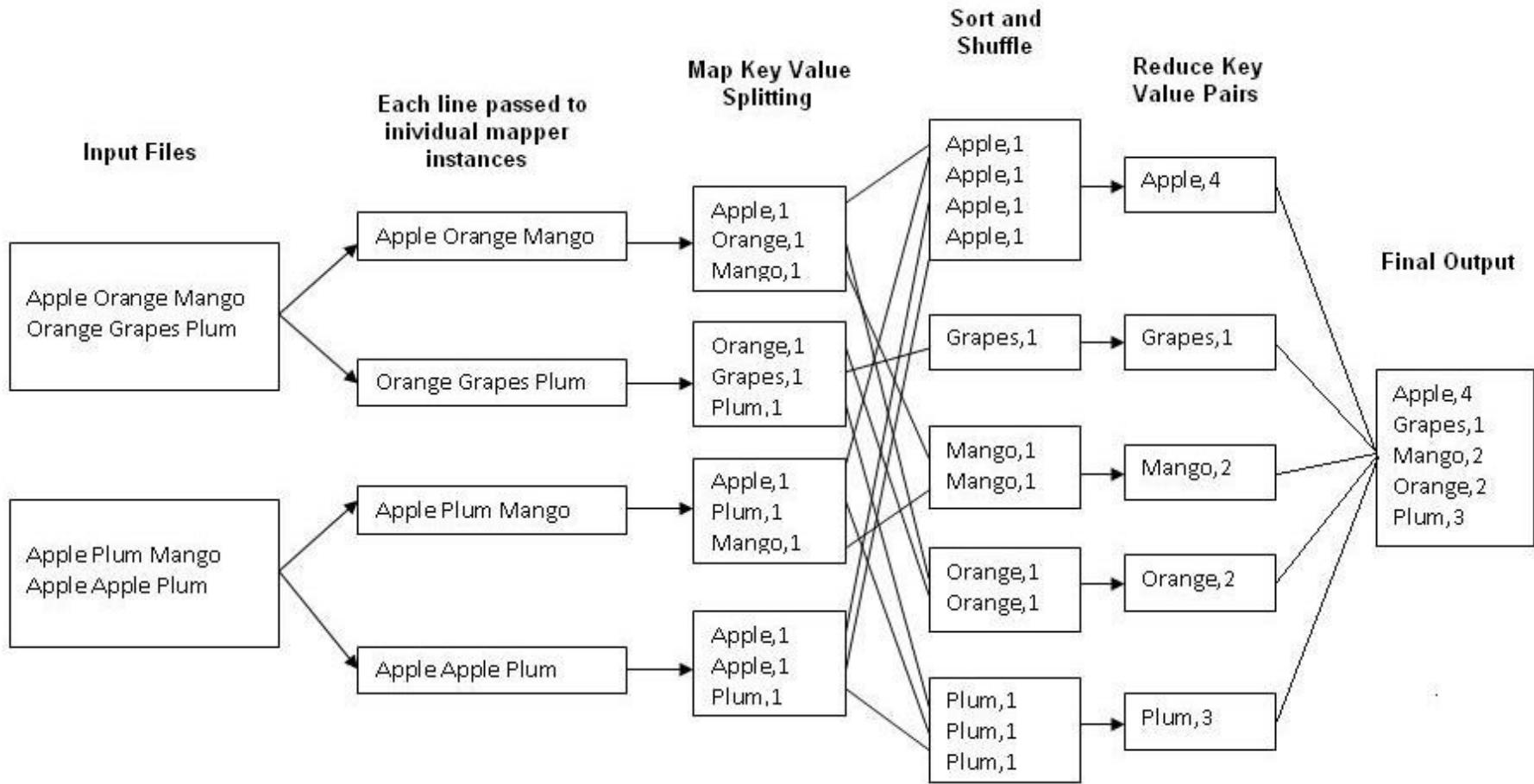
- Alur data dari seluruh job MapReduce



Keuntungan MapReduce

- Sederhana (beserta fault tolerance)
 - Terutama dibandingkan dengan model pemrograman terdistribusi lainnya
- Fleksibel
 - Menyediakan kemampuan analitik dan bekerja dengan banyak tipe data dibandingkan SQL
- Skalabilitas
 - Karena bekerja dengan
 - Data berkuantitas kecil pada satu waktu
 - Berjalan secara paralel antar cluster
 - Tidak berbagi pakai apapun di node yang tersedia

Contoh lain



Hadoop Resource Management: YARN

- Apa itu YARN?
 - YARN = Yet Another Resource Negotiator
 - YARN adalah lapisan (*layer*) pemrosesan dari Hadoop, berisi
 - Resource Manager → pengelola *resource*
 - Job-scheduler → menjadwalkan *job*
 - YARN memungkinkan beberapa mesin pemroses data untuk berjalan dalam single cluster Hadoop
 - Batch program (mis. Spark, MapReduce)
 - Interactive SQL (mis. Impala)
 - Analitik lanjut (mis. Spark, Impala)
 - Streaming (mis. Spark Streaming)

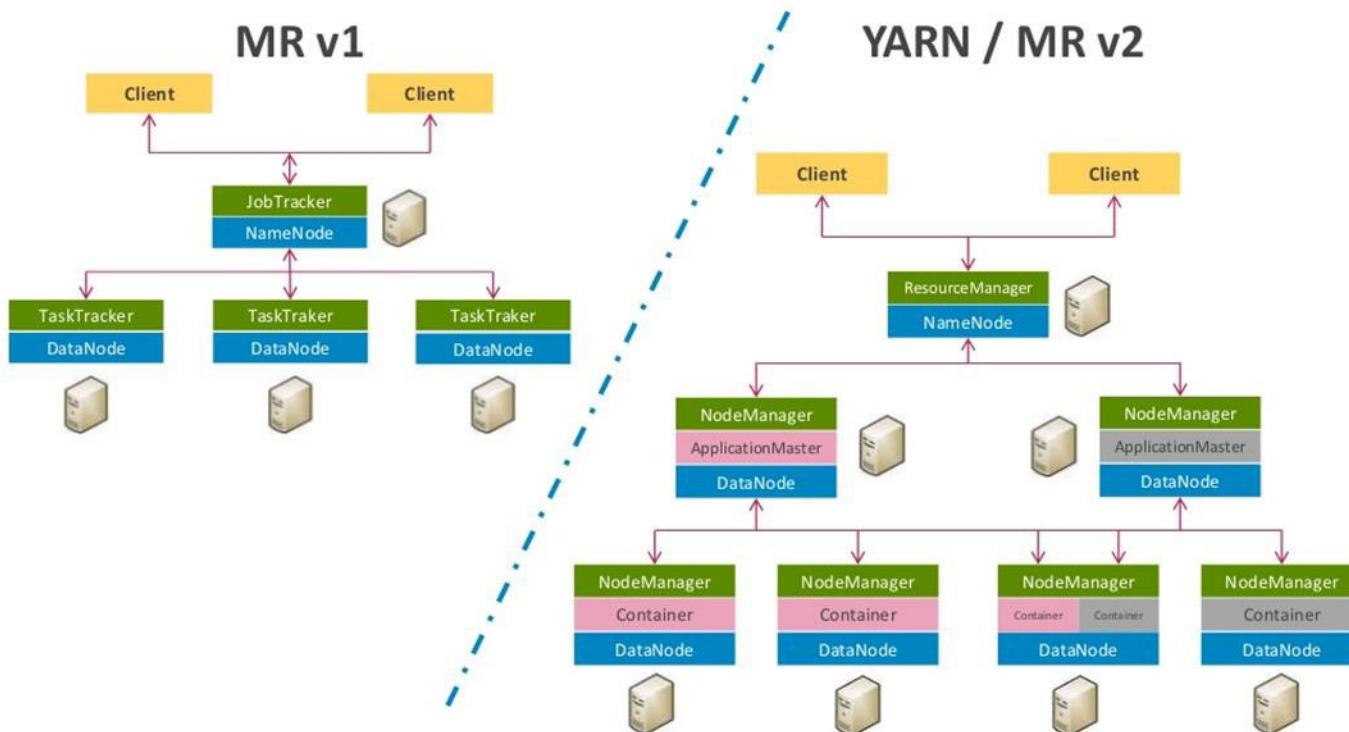
YARN Daemon

- Resource Manager (RM)
 - Berjalan di master node
 - Resource scheduler global
 - Mendukung berbagai algoritme scheduler (capacity, fair scheduler, dll)
- Node Manager (NM)
 - Berjalan di slave node
 - Berkommunikasi dengan RM

Konfigurasi Hadoop

- Arsitektur Map/Reduce (MR) vs Yet Another Resource Negotiator (YARN)

MR VS. YARN ARCHITECTURE



- **YARN** : Yet Another Resource Negotiator
- **MR** : MapReduce

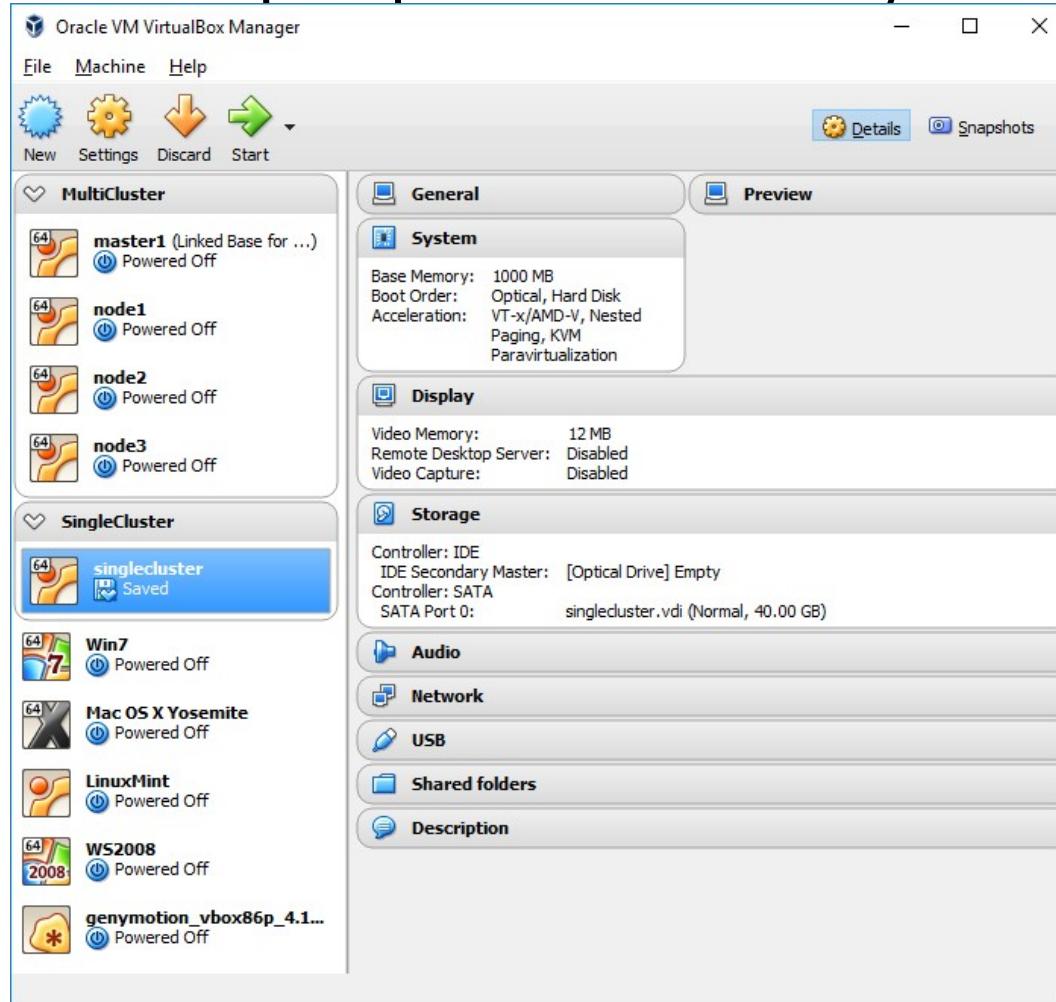
Konfigurasi Hadoop

- Single Node Cluster (pseudo-distributed) Pada Linux & Windows (download file tutorial lengkapnya dari link:
<https://goo.gl/7bJhdi>):
 - Buka terminal dan ketikkan "sudo nano /etc/hosts"
 - sudo apt-get update
 - sudo apt-get install default-jdk (cek dengan java -version)
 - sudo addgroup hadoop
 - sudo adduser -ingroup hadoop hduser
 - sudo adduser hduser sudo
 - sudo apt-get install ssh
 - su hduser
 - ssh-keygen -t rsa -P ""
 - ...
 - etc

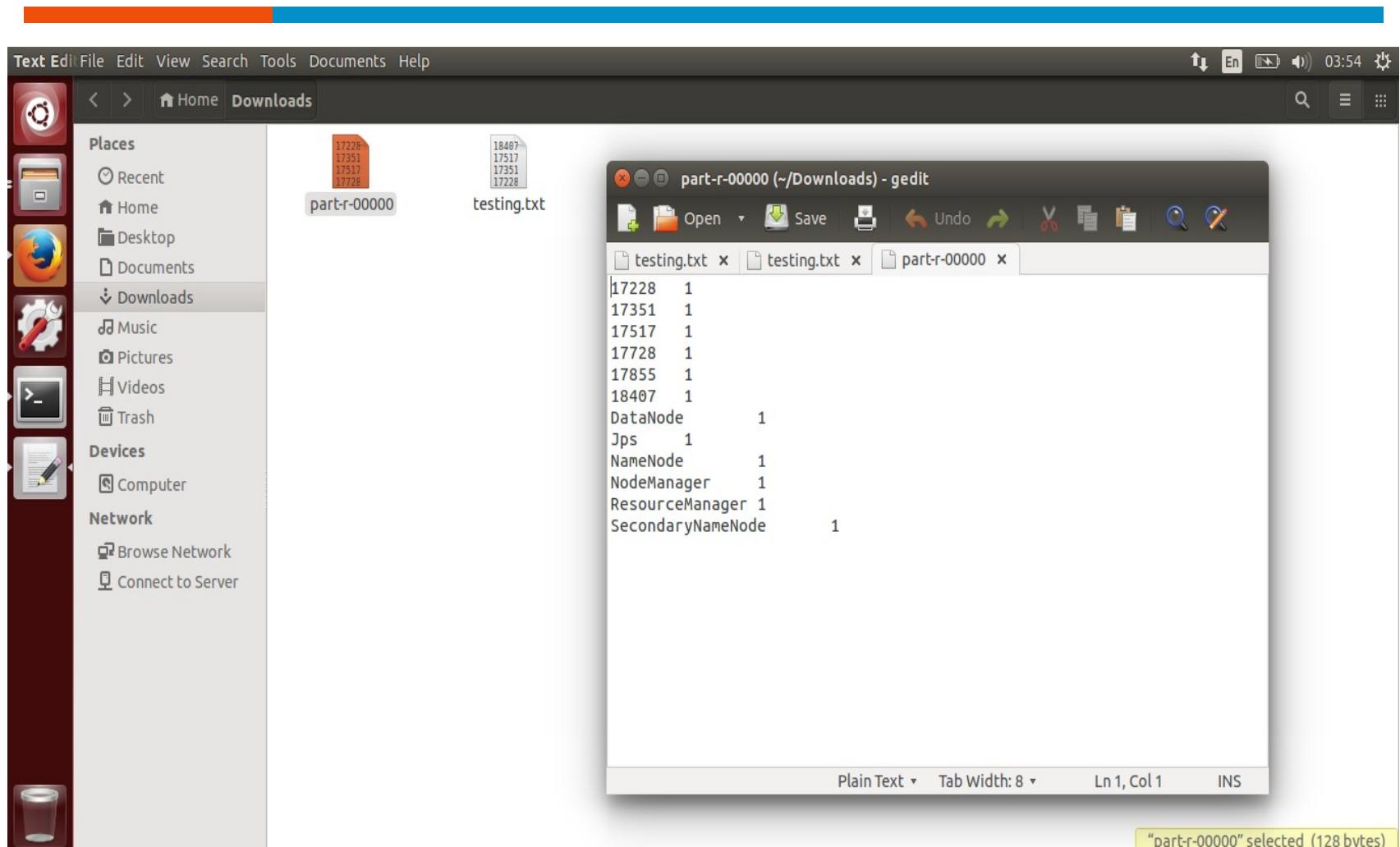
<https://hadoop.apache.org/docs/r2.8.2/hadoop-project-dist/hadoop-common/SingleCluster.html>

Konfigurasi Hadoop

- Persiapan pada Virtual Box/VMware



Studi Kasus WordCount



Kelemahan Hadoop

- Inti dari Hadoop adalah HDFS dan MapReduce
 - HDFS untuk menyimpan semua data
 - Map Reduce untuk mengolah data dan mendapatkan informasi yang berguna dari HDFS
- Kelemahan dari Hadoop:
 - MapReduce hanya bisa berjalan secara serial untuk mengolah data
 - Artinya tidak bisa dilakukan pemrosesan data secara paralel. Hal ini sangat terasa dengan Hadoop versi 1.x.
 - Untuk Hadoop versi 2.x sudah ada teknologi baru yang ditambahkan yaitu YARN, sehingga kelemahan ini bisa diabaikan
 - Map Reduce hanya bisa berjalan dalam batch atau secara periodik dan tidak bisa terus menerus secara *realtime*.
 - Hal ini membuat Map Reduce tidak bisa mengolah data dalam bentuk streaming tanpa henti seperti misalnya tweet dari Twitter.

Kelemahan Hadoop

- Namun kelemahan tsb dapat diatasi dengan teknologi lain
 - Misal **Apache Storm** dan **Apache Spark** yang berada di atas Hadoop, sehingga penggabungan ini menjadi satu kesatuan yang masih merupakan solusi Big Data paling populer.
- Kelemahan lain dari Hadoop yaitu:
 - Latency data processing.
 - Latency adalah keterlambatan data untuk diambil dari HDFS, dengan menggunakan Map Reduce, ke level aplikasi misalnya web app. Bagi yang sudah pernah menjalankan Map Reduce di Hadoop akan merasakan adanya kelambatan dalam mengolah data.
 - Kelambatan ini selain karena sifat Map Reduce yang berdasarkan batch, juga karena ukuran data yang relatif sangat besar.
 - Untuk mengatasi masalah ini, software lain (Non Relational DB (NoSQL) seperti Mongo DB, Apache HBase, Apache Cassandra, dll) bisa ditambahkan.

Kelemahan Hadoop

- Kelemahan lain dari Hadoop yaitu:
 - Streaming data processing.
 - Pada streaming data secara realtime. Kelemahan ini banyak dikeluhkan oleh pengguna Hadoop karena data yang masuk secara terus-menerus tidak bisa dianalisis langsung secara realtime karena map reduce berjalan secara batch atau periodik. Contoh kasus ini misalnya dalam menghitung hashtag paling populer / trending dari seluruh tweets yang dibuat di twtitter secara realtime.
 - Ada tiga software yang saya temukan bisa menutupi kelemahan ini. Ketiga software itu adalah Spring-XD, Apache Storm dan Apache Spark Streaming.

Terima kasih



Solusi Hadoop: Permission

- Localhost:50070 tidak ada koneksi:
- Apabila ada *permission denied* berarti karena masalah permission (ownership). Lakukan hal berikut:

path tergantung dari config

```
hduser@Master:~$ sudo rm -rf /usr/local/hadoop_tmp
hduser@Master:~$ sudo chown hduser:hadoop -R /usr/local/hadoop_tmp
chown: cannot access '/usr/local/hadoop_tmp': No such file or directory
hduser@Master:~$ sudo mkdir -p /usr/local/hadoop_tmp
hduser@Master:~$ sudo mkdir -p /usr/local/hadoop_tmp/hdfs/datanode
hduser@Master:~$ sudo mkdir -p /usr/local/hadoop_tmp/hdfs/namenode
hduser@Master:~$ sudo chown
hduser:hadoop -R /usr/local/hadoop_tmp
hduser@Master:~$ sudo chown hduser:hadoop -R /usr/local/hadoop/
hduser@Master:~$ hadoop namenode -format
```

Solusi Hadoop: Membuat folder

- Membuat Directories di HDFS harus satu demi satu:
- Lakukan hal berikut:

```
hduser@Master:/usr/local/hadoop$ bin/hdfs dfs -mkdir /user  
hduser@Master:/usr/local/hadoop$ bin/hdfs dfs -mkdir /user/hduser  
hduser@Master:/usr/local/hadoop$ bin/hdfs dfs -mkdir  
/user/hduser/wordcount  
hduser@Master:/usr/local/hadoop$ bin/hdfs dfs -ls /user/hduser  
Found 1 items  
drwxr-xr-x    - hduser supergroup    0 2016-11-20 22:03  
/user/hduser/wordcount  
hduser@Master:/usr/local/hadoop$ bin/hdfs dfs -mkdir  
/user/hduser/wordcount/input
```

Solusi Hadoop: Contoh MapReduce dengan Java

- Siapkan file *.java (misal WordCount.java Part 1 of 2) untuk dikompilasi ke *.jar:

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

    public static class TokenizerMapper
        extends Mapper<Object, Text, Text, IntWritable> {

        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

        public void map(Object key, Text value, Context context) throws IOException, InterruptedException {
            StringTokenizer itr = new StringTokenizer(value.toString());
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
            }
        }
    }
}
```

Solusi Hadoop: Contoh MapReduce dengan Java

- Siapkan file *.java (misal WordCount.java Part 2 of 2) untuk dikompilasi ke *.jar:

```
public static class IntSumReducer
    extends Reducer<Text, IntWritable, Text, IntWritable> {

    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values, Context context
    ) throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        result.set(sum);
        context.write(key, result);
    }
}

public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```

Solusi Hadoop: Contoh MapReduce dengan Java

- Error: Could not find or load main class com.sun.tools.javac.Main:
- Lakukan hal berikut:

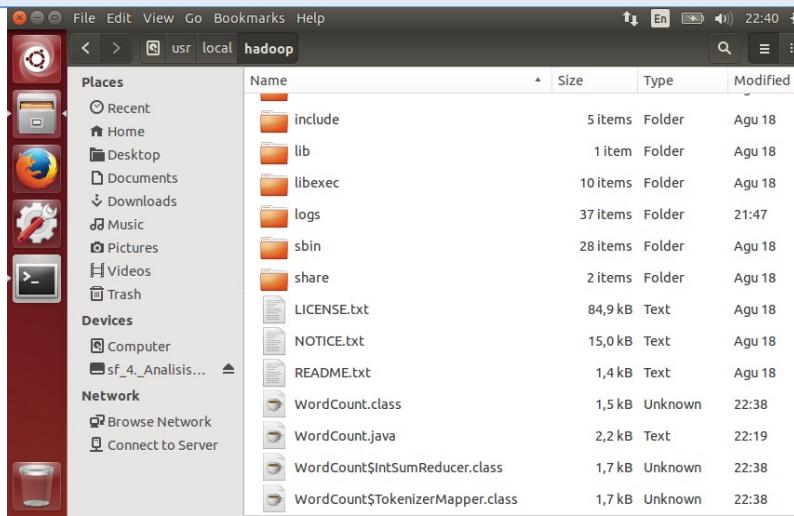
```
hduser@Master:/usr/local/hadoop$ sudo gedit ~/.bashrc
```

Lalu tambahkan di line paling akhir:

```
export HADOOP_CLASSPATH=/usr/lib/jvm/java-7-openjdk-amd64/lib/tools.jar
```

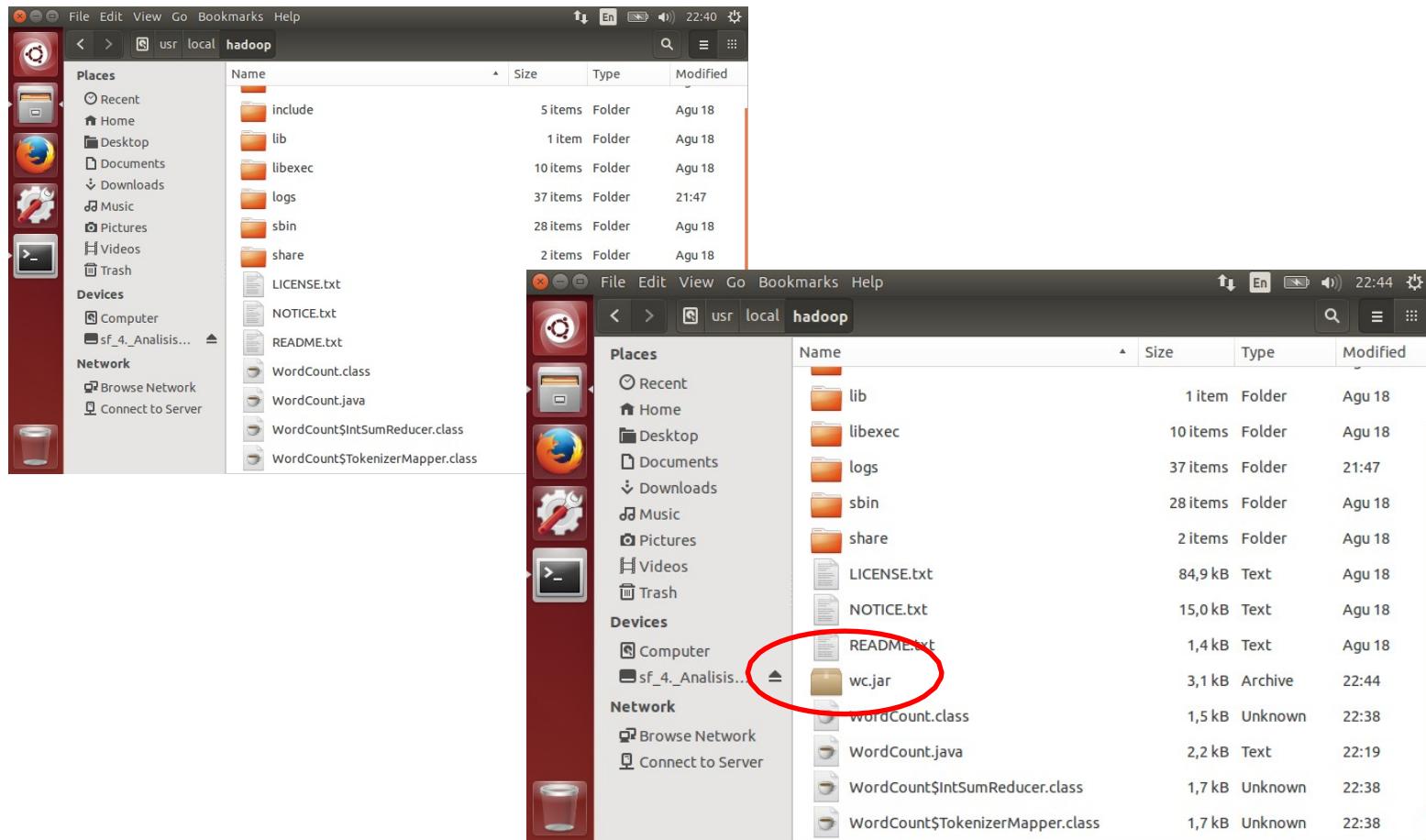
Lalu ketikkan “hduser@Master:/usr/local/hadoop\$source ~/.bashrc” atau dengan me-restart PC anda, lalu coba lalu coba lagi:

```
hduser@Master:/usr/local/hadoop$ bin/hdfs com.sun.tools.javac.Main WordCount.java
```



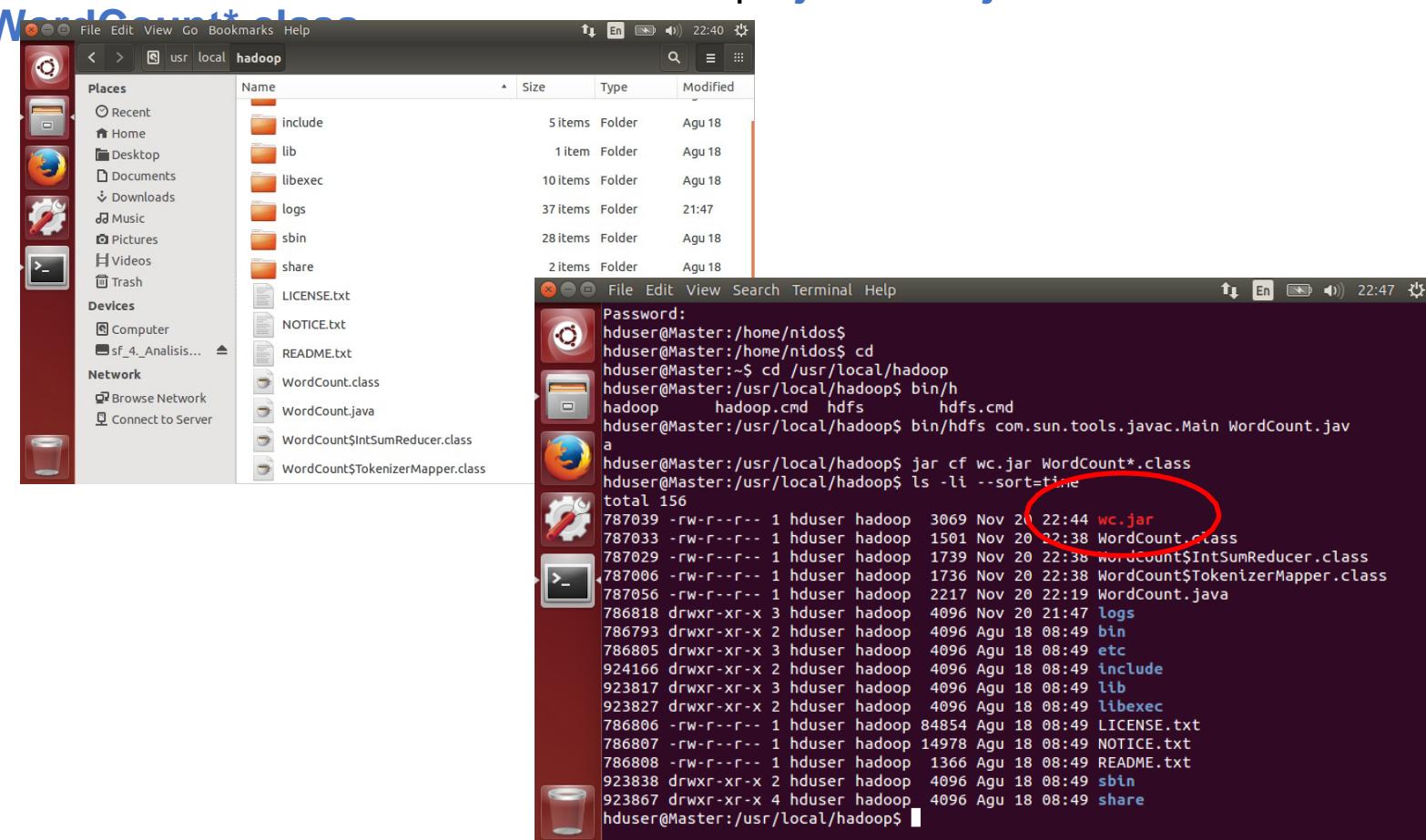
Solusi Hadoop: Kompilasi ke .jar

- Hasil: hduser@Master:/usr/local/hadoop\$ **jar cf wc.jar WordCount*.class**



Solusi Hadoop: Kompilasi ke .jar

- Hasil: hduser@Master:/usr/local/hadoop\$ **jar cf wc.jar**



The screenshot shows a Linux desktop environment with a file manager window and a terminal window.

File Manager: The left window shows the directory structure of /usr/local/hadoop. It contains subfolders like include, lib, libexec, logs, sbin, and share, along with several text files: LICENSE.txt, NOTICE.txt, README.txt, WordCount.class, WordCount.java, WordCount\$IntSumReducer.class, and WordCount\$TokenizerMapper.class.

Terminal: The right window is a terminal window titled "WordCount class". It shows the user's command-line session:

```
hduser@Master:/home/nidos$  
hduser@Master:/home/nidos$ cd  
hduser@Master:~$ cd /usr/local/hadoop  
hduser@Master:/usr/local/hadoop$ bin/hadoop hadoop.cmd hdfs hdfs.cmd  
hduser@Master:/usr/local/hadoop$ bin/hdfs com.sun.tools.javac.Main WordCount.java  
a  
hduser@Master:/usr/local/hadoop$ jar cf wc.jar WordCount*.class  
hduser@Master:/usr/local/hadoop$ ls -li --sort=time  
total 156  
787039 -rw-r--r-- 1 hduser hadoop 3069 Nov 20 22:44 wc.jar  
787033 -rw-r--r-- 1 hduser hadoop 1501 Nov 20 22:38 WordCount.class  
787029 -rw-r--r-- 1 hduser hadoop 1739 Nov 20 22:38 WordCount$IntSumReducer.class  
787006 -rw-r--r-- 1 hduser hadoop 1736 Nov 20 22:38 WordCount$TokenizerMapper.class  
787056 -rw-r--r-- 1 hduser hadoop 2217 Nov 20 22:19 WordCount.java  
786818 drwxr-xr-x 3 hduser hadoop 4096 Nov 20 21:47 logs  
786793 drwxr-xr-x 2 hduser hadoop 4096 Agu 18 08:49 bin  
786805 drwxr-xr-x 3 hduser hadoop 4096 Agu 18 08:49 etc  
924166 drwxr-xr-x 2 hduser hadoop 4096 Agu 18 08:49 include  
923817 drwxr-xr-x 3 hduser hadoop 4096 Agu 18 08:49 lib  
923827 drwxr-xr-x 2 hduser hadoop 4096 Agu 18 08:49 libexec  
786806 -rw-r--r-- 1 hduser hadoop 84854 Agu 18 08:49 LICENSE.txt  
786807 -rw-r--r-- 1 hduser hadoop 14978 Agu 18 08:49 NOTICE.txt  
786808 -rw-r--r-- 1 hduser hadoop 1366 Agu 18 08:49 README.txt  
923838 drwxr-xr-x 2 hduser hadoop 4096 Agu 18 08:49 sbin  
923867 drwxr-xr-x 4 hduser hadoop 4096 Agu 18 08:49 share  
hduser@Master:/usr/local/hadoop$
```

The line "787039 -rw-r--r-- 1 hduser hadoop 3069 Nov 20 22:44 wc.jar" is circled in red.

Solusi Hadoop

- Error: Could not find or load main class fs:

Jika menggunakan hdfs, maka gunakan dfs

Jika menggunakan hadoop, maka gunakan fs

Contoh yang salah:

```
hduser@Master:/usr/local/hadoop$ bin/hdfs fs -copyFromLocal /home/nidos/Desktop/data/a.txt  
/user/hduser/wordcount/input
```

Contoh yang benar:

```
hduser@Master:/usr/local/hadoop$ bin/hdfs dfs -copyFromLocal /home/nidos/Desktop/data/a.txt  
/user/hduser/wordcount/input
```

Jika error:

```
hduser@Master:/usr/local/hadoop$ bin/hdfs dfs -copyFromLocal /home/nidos/Desktop/data/a.txt  
/user/hduser/wordcount/input
```

```
16/11/20 22:56:34 WARN util.NativeCodeLoader: Unable to load native-hadoop library for  
your platform... using builtin-java classes where applicable
```

```
copyFromLocal: Call From Master/127.0.1.1 to localhost:9000 failed on connection  
exception: java.net.ConnectException: Connection refused; For more details see:
```

<http://wiki.apache.org/hadoop/ConnectionRefused>

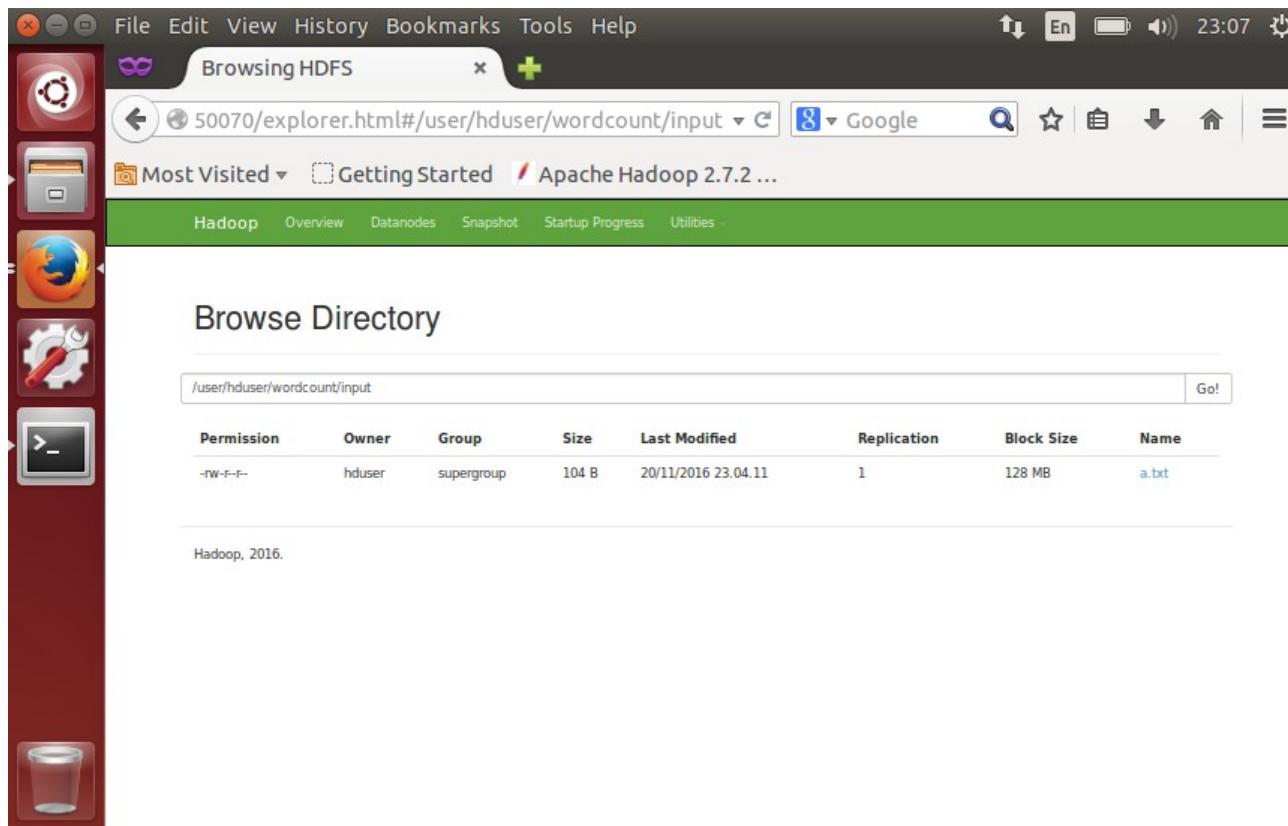
Maka, lakukan

```
hduser@Master:/usr/local/hadoop$ start-all.sh
```

Solusi Hadoop: Unggah File

Bisa menggunakan
-put

- Hasil:
- `hduser@Master:/usr/local/hadoop$ bin/hdfs dfs -copyFromLocal /home/nidos/Desktop/data/a.txt /user/hduser/wordcount/input`



Solusi Hadoop

- Solusi jika sering muncul warning: WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

```
hduser@Master:/usr/local/hadoop$ sudo gedit ~/.bashrc
```

Lalu replace di line:

```
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"
```

Dengan:

```
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib/native"
```

Lalu ketikkan “hduser@Master:/usr/local/hadoop\$ **source** ~/.bashrc” atau dengan *me-restart* PC, lalu coba cek apakah *warning* tersebut muncul lagi atau tidak

Solusi Hadoop: Melihat isi file

- Cara untuk melihat isi dari dokumen yang akan diproses:

```
hduser@Master:/usr/local/hadoop$ bin/hdfs dfs -cat  
/user/hduser/wordcount/input/a.txt  
18407 Jps  
17517 SecondaryNameNode  
17351 DataNode  
17228 NameNode  
17728 ResourceManager  
17855 NodeManager
```

```
hduser@Master:/usr/local/hadoop$ bin/hdfs dfs -cat  
/user/hduser/wordcount/input/b.txt  
1 8 4 0 7 J p s  
1 7 5 1 7 Secondary Name Node  
1 7 3 5 1 Data Node  
17228 Name Node  
17728 Resource Manager  
17855 Node Manager  
hduser@Master:/usr/local/hadoop$
```

Solusi Hadoop: Menjalankan MR Java 1 Input File

- Menjalankan JAR untuk wordcount (file a.txt saja):

```
hduser@Master:/usr/local/hadoop$ bin/hdfs dfs -ls /user/hduser/wordcount
Found 2 items
drwxr-xr-x      - hduser supergroup          0 2016-11-20 23:08
/usr/hduser/wordcount/input
drwxr-xr-x      - hduser supergroup          0 2016-11-20 23:27
/usr/hduser/wordcount/output
```

Jika folder output sudah ada, maka sebaiknya membuat output lainnya, misal “output2”

```
hduser@Master:/usr/local/hadoop$ bin/hadoop jar wc.jar WordCount
/usr/hduser/wordcount/input/a.txt /user/hduser/wordcount/output2
```

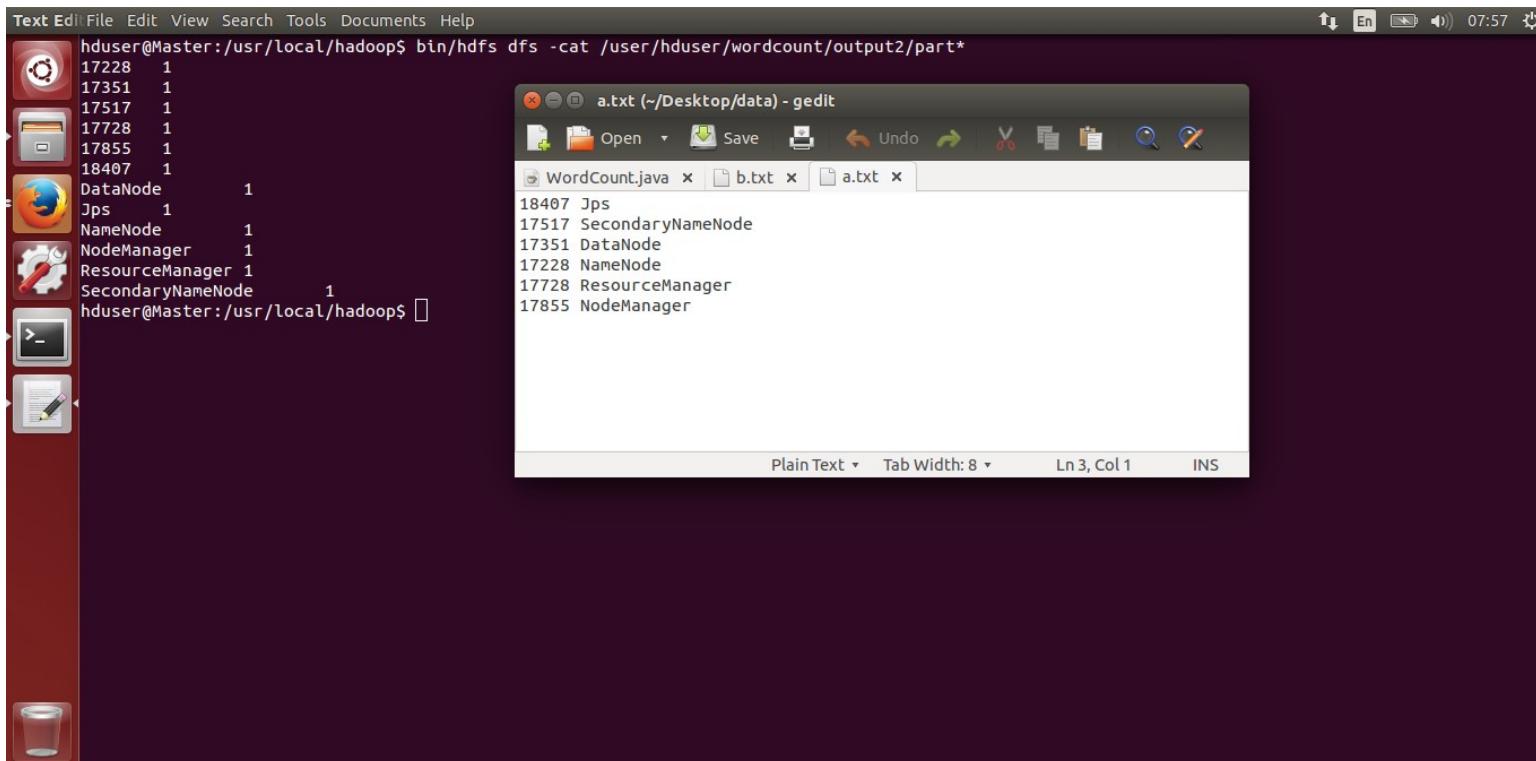
```
hduser@Master:/usr/local/hadoop$ bin/hdfs dfs -ls
/usr/hduser/wordcount/output2
Found 2 items
-rw-r--r--  1 hduser supergroup    0 2016-11-21 07:24
/usr/hduser/wordcount/output2/_SUCCESS
-rw-r--r--  1 hduser supergroup  128 2016-11-21 07:24
/usr/hduser/wordcount/output2/part-r-
00000
```

Untuk menampilkan isi dari output2/*

```
hduser@Master:/usr/local/hadoop$ bin/hdfs dfs -cat /user/hduser/wordcount/output2/part*
```

Solusi Hadoop: Menjalankan MR Java 1 Input File

- Menjalankan JAR untuk wordcount (file a.txt saja):



Solusi Hadoop: Menjalankan MR Java 1 Input File

- Menjalankan JAR untuk wordcount (file b.txt saja):

```
hduser@Master:/usr/local/hadoop$ bin/hadoop jar wc.jar WordCount /user/hduser/wordcount/input/b.txt  
output
```

```
hduser@Master:/usr/local/hadoop$ bin/hdfs dfs -ls /user/hduser/wordcount/  
Found 3 items
```

```
drwxr-xr-x - hduser supergroup      0 2016-11-20 23:08 /user/hduser/wordcount/input  
drwxr-xr-x - hduser supergroup      0 2016-11-20 23:27 /user/hduser/wordcount/output  
drwxr-xr-x - hduser supergroup      0 2016-11-21 07:24 /user/hduser/wordcount/output2  
hduser@Master:/usr/local/hadoop$ bin/hdfs dfs -ls
```

```
Found 2 items  
drwxr-xr-x - hduser supergroup      0 2016-11-21 07:32 output  
drwxr-xr-x - hdusersupergroup      0 2016-11-21 07:24 wordcount
```

```
hduser@Master:/usr/local/hadoop$ bin/hdfs dfs -ls /output
```

```
ls: `/output': No such file or directory
```

```
hduser@Master:/usr/local/hadoop$ bin/hdfs dfs -ls output
```

```
Found 2 items  
-rw-r--r--    1 hduser supergroup          0 2016-11-21 07:32 output/_SUCCESS  
-rw-r--r--    1 hduser supergroup        118 2016-11-21 07:32 output/part-r-00000  
hduser@Master:/usr/local/hadoop$ bin/hdfs dfs -cat output/part*
```

Solusi Hadoop: Menjalankan MR Java 1 Input File

- Hasil menjalankan JAR untuk wordcount (file b.txt saja):

The screenshot shows a Linux desktop environment with a terminal window and a text editor window.

Terminal Output:

```
Text Editor
Found 3 items
drwxr-xr-x  1 hduser supergroup      0 2016-11-20 23:08 /user/hduser/wordcount/input
drwxr-xr-x  1 hduser supergroup      0 2016-11-20 23:27 /user/hduser/wordcount/output
drwxr-xr-x  1 hduser supergroup      0 2016-11-21 07:24 /user/hduser/wordcount/output2
hduser@Master:/usr/local/hadoop$ bin/hdfs dfs -ls
Found 2 items
drwxr-xr-x  - hduser supergroup      0 2016-11-21 07:32 output
drwxr-xr-x  - hduser supergroup      0 2016-11-21 07:24 wordcount
hduser@Master:/usr/local/hadoop$ bin/hdfs dfs -ls /output
ls: `/output': No such file or directory
hduser@Master:/usr/local/hadoop$ bin/hdfs dfs -ls output
Found 2 items
-rw-r--r--  1 hduser supergroup      0 2016-11-21 07:32 output/_SUCCESS
-rw-r--r--  1 hduser supergroup    118 2016-11-21 07:32 output/part-r-00000
hduser@Master:/usr/local/hadoop$ bin/hdfs dfs -cat output/part*
0      1
1      5
17228  1
17855  1
3      1
4      1
5      2
7      4
8      1
Data   1
J      1
Manager 2
Name   2
Node   4
Resource 1
Secondary 1
p      1
s      1
hduser@Master:/usr/local/hadoop$
```

Text Editor Content:

```
b.txt (~/Desktop/data) - gedit
WordCount.java x b.txt x
Plain Text ▾ Tab Width: 8 ▾ Ln 6, Col 19 INS
1 8 4 0 7 J p s
1 7 5 1 7 Secondary Name Node
1 7 3 5 1 Data Node
17228 Name Node
17728 Resource Manager
17855 Node Manager
```

Hasil dari

```
hduser@Master:/usr/local/hadoop$ bin/Hadoop jar wc.jar WordCount
/user/hduser/wordcount/input/b.txt output
```

(Note: Sebaiknya "output" dibuat menjadi spesifik misal, "/user/hduser/wordcount/output3")

Solusi Hadoop: Menjalankan MR Java 2 Input File

- Menjalankan JAR untuk wordcount untuk semua file dalam satu folder (file a.txt dan b.txts):

```
hduser@Master:/usr/local/hadoop$ bin/hadoop jar wc.jar WordCount  
/user/hduser/wordcount/input/ /user/hduser/wordcount/output4
```

```
hduser@Master:/usr/local/hadoop$ bin/hdfs dfs -ls  
/user/hduser/wordcount/output4
```

Found 2 items

```
-rw-r--r--    1 hduser supergroup    0 2016-11-21 07:46  
/user/hduser/wordcount/output4/_SUCCESS  
-rw-r--r--    1 hduser supergroup   222 2016-11-21 07:46  
/user/hduser/wordcount/output4/part-r-00000
```

```
hduser@Master:/usr/local/hadoop$ bin/hdfs dfs -cat  
/user/hduser/wordcount/output4/part*
```



Credit

- Presentasi Putra pandu Adikara, Universitas Brawijaya
- Presentasi Imam Cholisoddin, Universitas Brawijaya