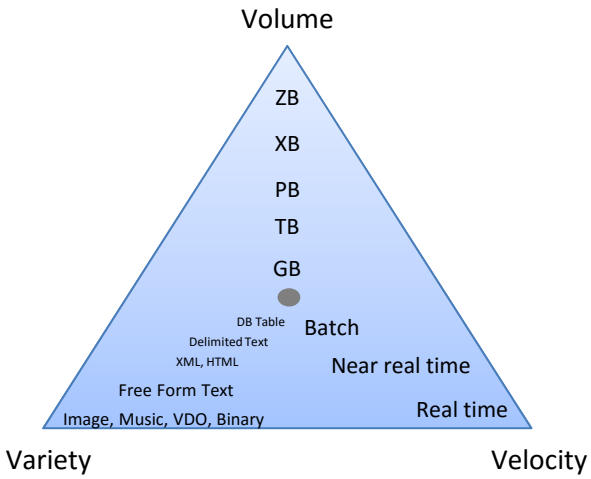
Learning Big Data Hadoop and Spark

Part 1 Big Data Technology and Example Cases
Part 2 Hadoop 2.7.2 Architecture
Part 3 Hands on: Hadoop HDFS and MRv2 on YARN
Part 4 Spark 2.0 Architecture
Part 5 Hands on: Spark on Hadoop HDFS
Part 6 Big Data Project Management

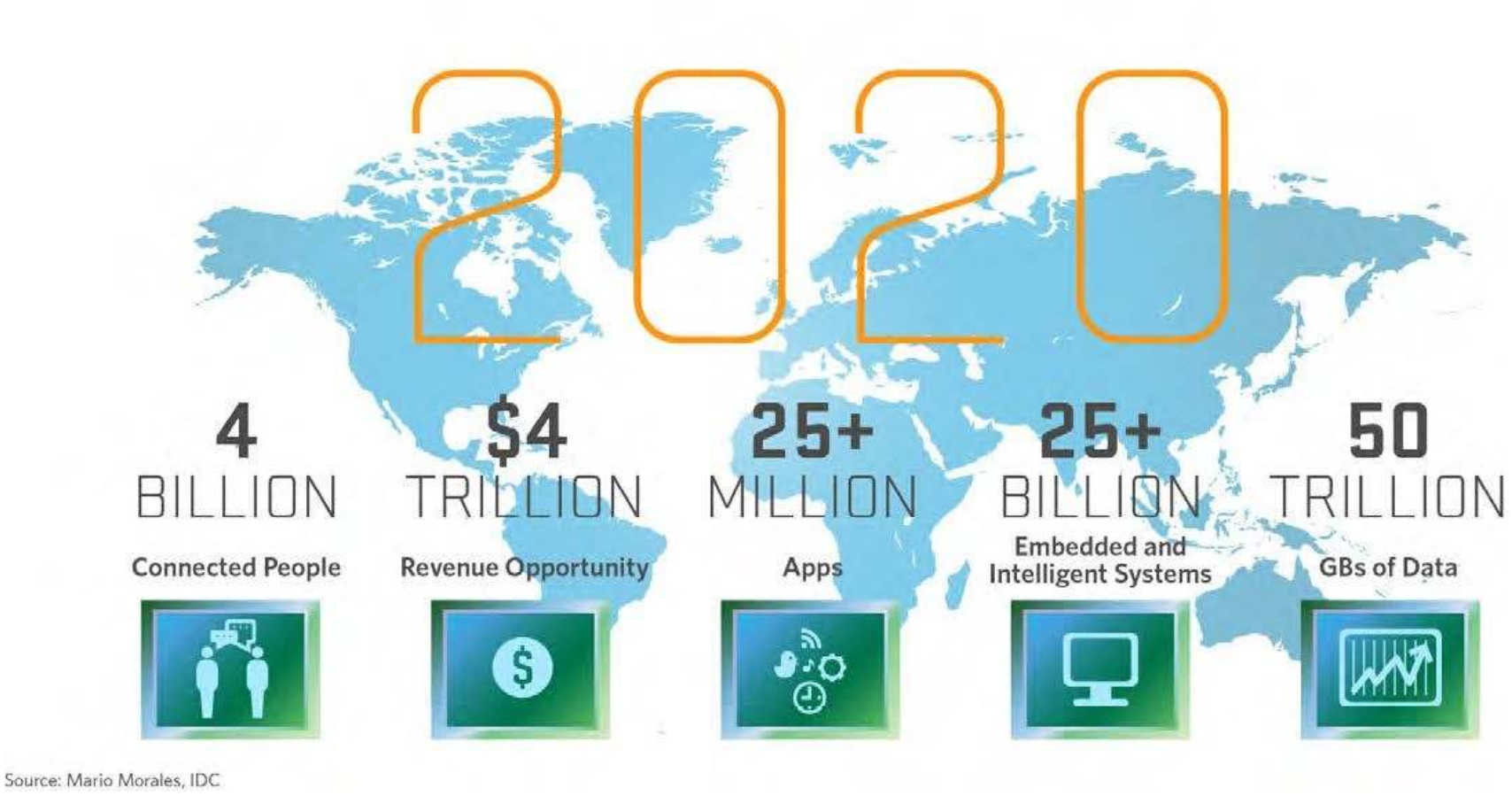
อ.ดนัยรัฐ ธนบดีธรรมจารี
+668-1559-1446 Line ID: danairat
FB: https://www.facebook.com/tdanairat



Big Data Introduction

อ.ดนัยรัฐ ธนบดีธรรมจารี
Line ID: Danairat
FB: Danairat Thanabodithammachari
+668-1559-1446

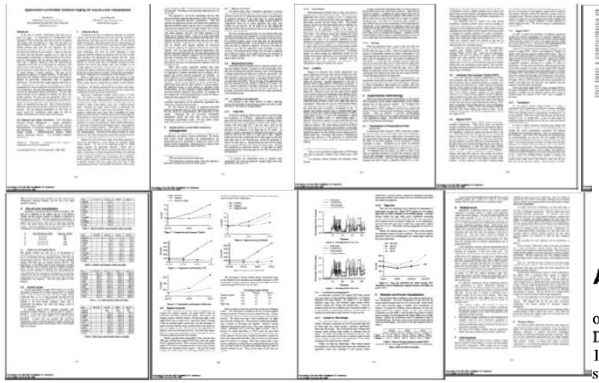
Why Big Data



Source: Mario Morales, IDC

A Short History Of Big Data

October 1997 Michael Cox and David Ellsworth published "**Application-controlled demand paging for out-of-core visualization**" in the Proceedings of the IEEE 8th conference on Visualization. When data sets do not fit in main memory (in core), or when they do not fit even on local disk, the most common solution is to acquire more resources. It is the first article in the ACM digital library to use the term **"Big Data"**.



https://www.evl.uic.edu/cavern/rg/20040525_renambot/Viz/parallel_volviz/paging_outofcore_viz97.pdf

Application-Controlled Demand Paging for Out-of-Core Visualization

Michael Cox
MRJ/NASA Ames Research Center
Microcomputer Research Labs, Intel Corporation
<mbc@nas.nasa.gov>

David Ellsworth
MRJ/NASA Ames Research Center
<ellswort@nas.nasa.gov>

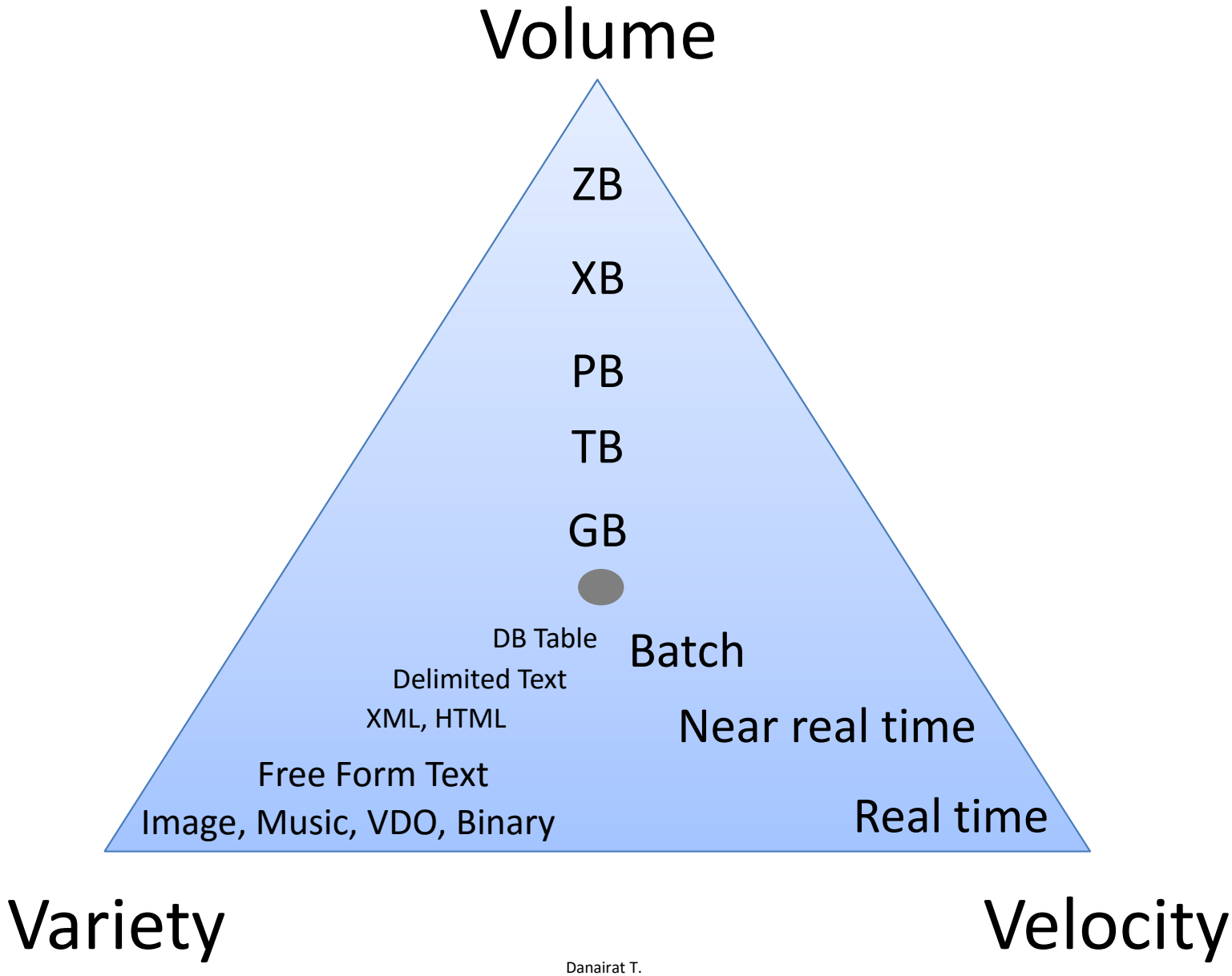
Abstract

In the area of scientific visualization, input data sets are often very large. In visualization of Computational Fluid Dynamics (CFD) in particular, input data sets today can surpass 100 Gbytes, and are expected to scale with the ability of supercomputers to generate them. Some visualization tools already partition large data sets into segments, and load appropriate segments as they are needed. However, this does not remove the problem for two reasons: 1) there are data sets for which even the individual segments are too large for the largest graphics workstations, 2) many practitioners do not have access to workstations with the memory capacity required to load even a segment, especially since the state-of-the-art visualization tools tend to be developed by researchers with

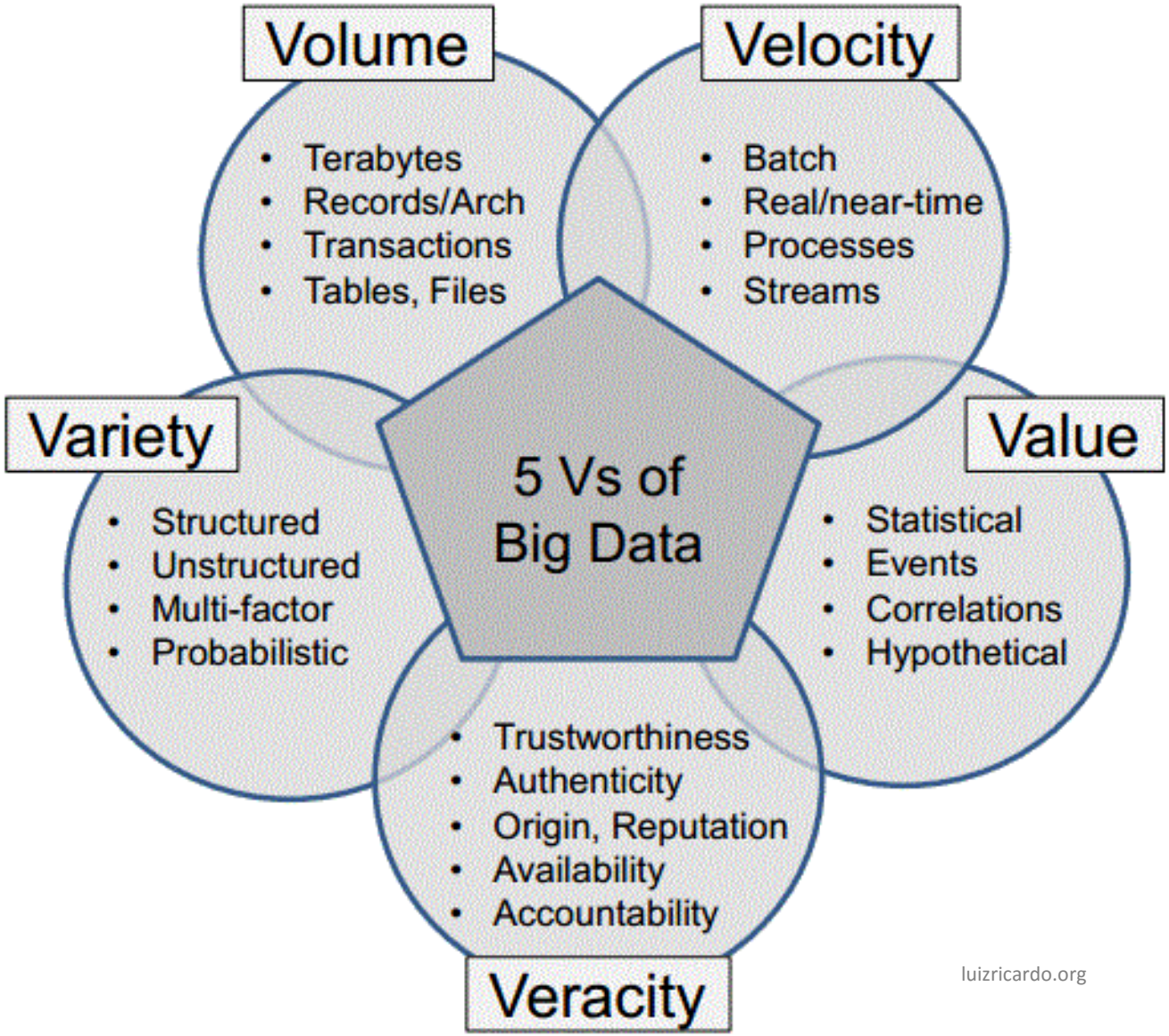
1 Introduction

Visualization provides an interesting challenge for computer systems: data sets are generally quite large, taxing the capacities of main memory, local disk, and even remote disk. We call this the problem of *big data*. When data sets do not fit in main memory (*in core*), or when they do not fit even on local disk, the most common solution is to acquire more resources. This *write-a-check* algorithm has two drawbacks. First, if visualization algorithms and tools are worth developing, then they are worth deploying to more production-oriented scientists and engineers who may have on their desks machines with significantly less memory and disk. Some researchers have noted that their software tools were not used in practice for several years after development because the tools required more power and memory than were available on the

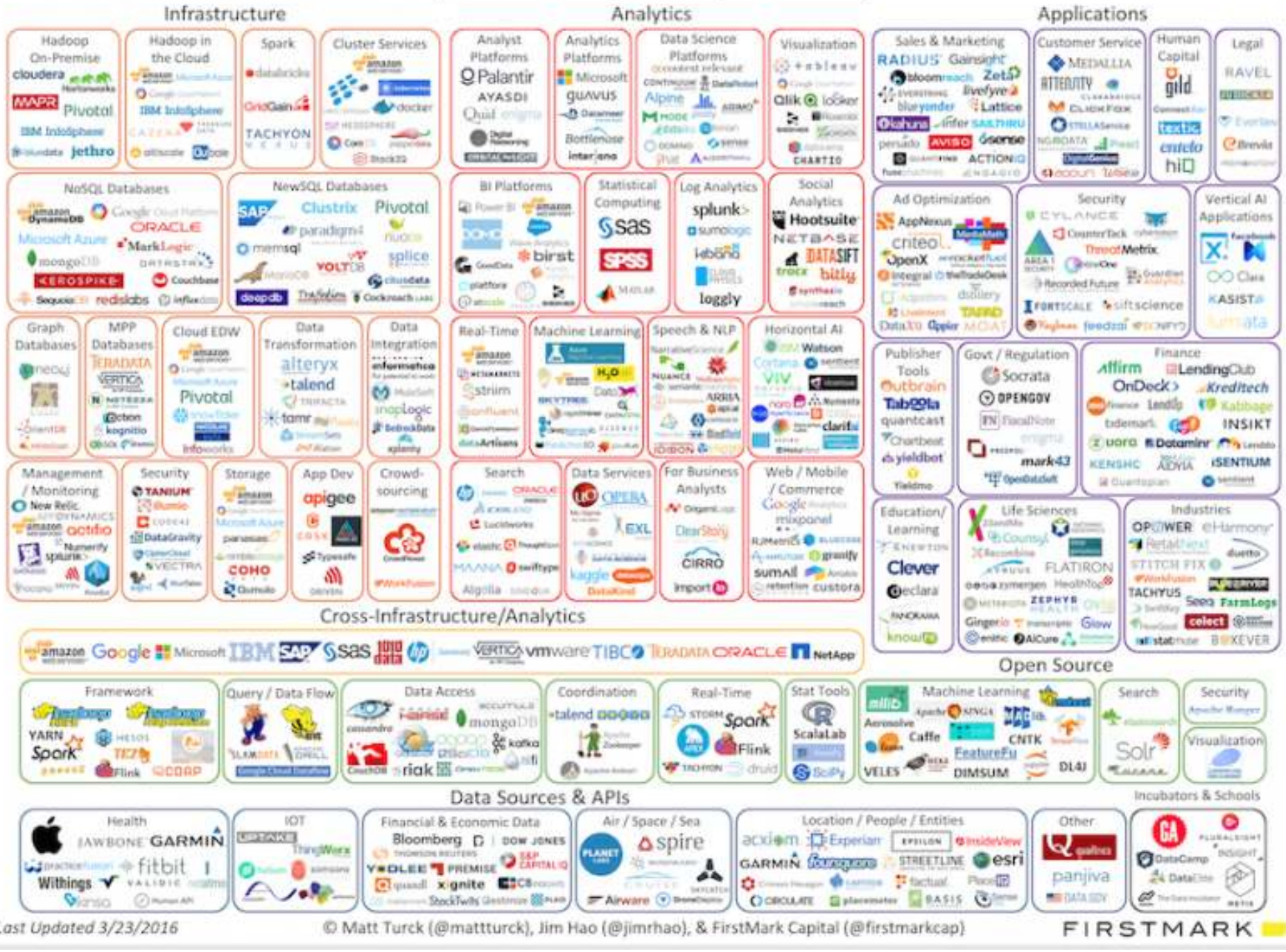
Introduction to Big Data 3Vs



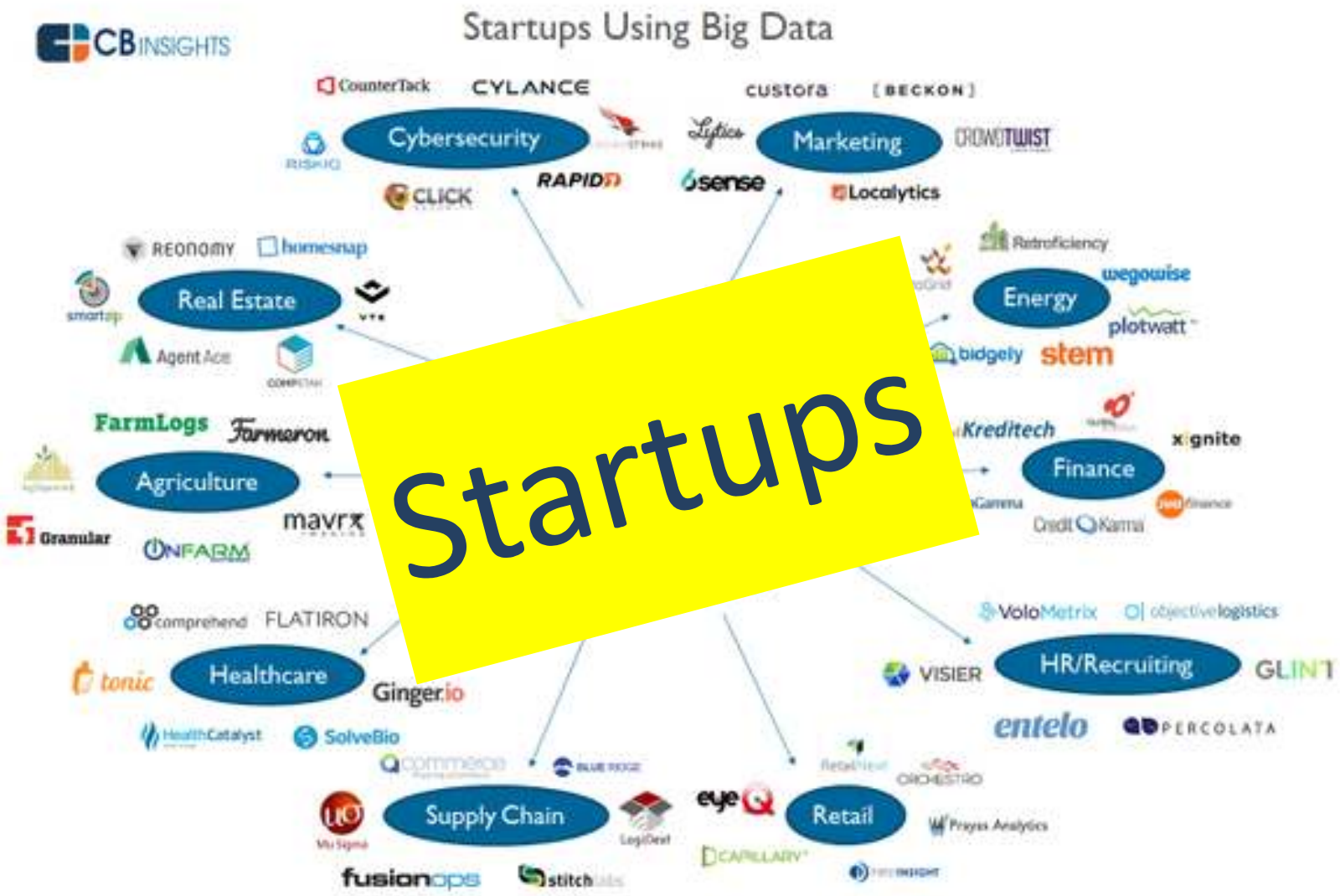
Introduction to Big Data 5Vs



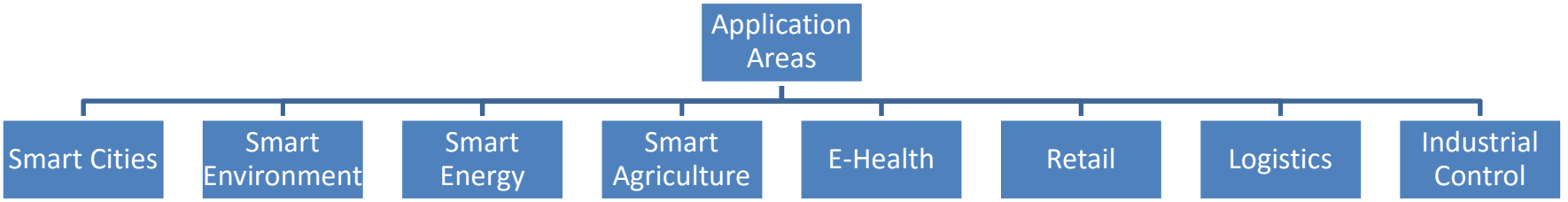
Big Data Landscape 2016 (Version 3.0)

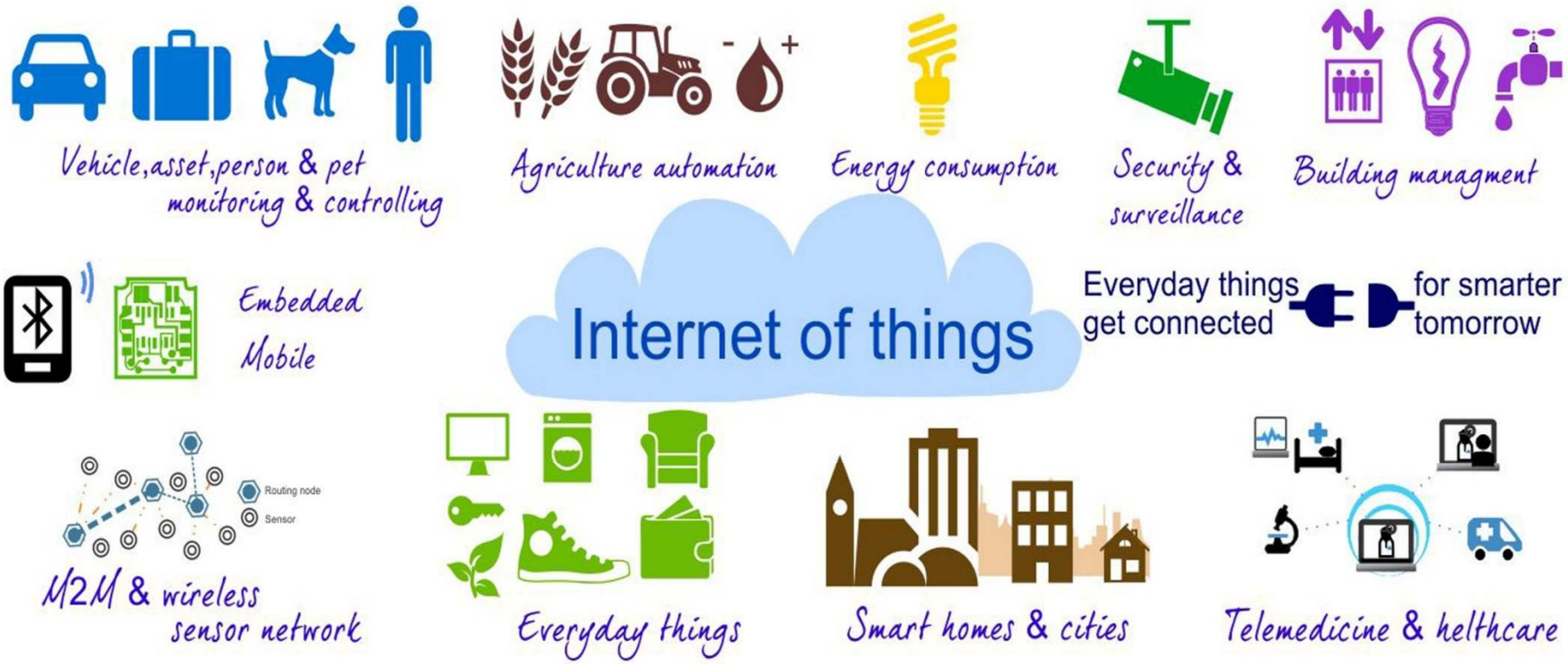


Startups using Big Data



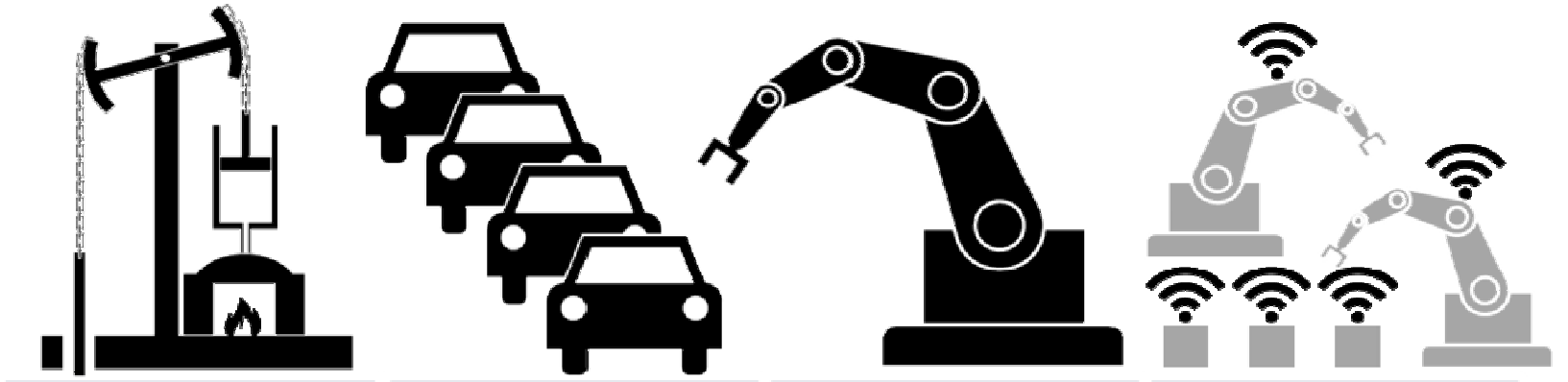
Internet of Things





Monika, 2015

Industry 4.0

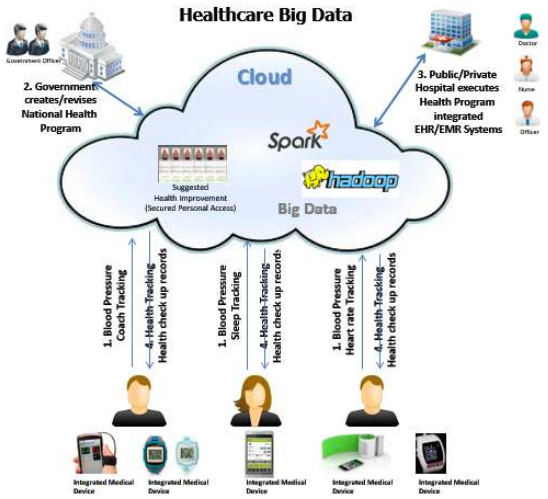


1st	2nd	3rd	4th
Mechanization, water power, steam power	Mass production, assembly line, electricity	Computer and automation	Cyber Physical Systems

https://en.wikipedia.org/wiki/Industry_4.0

The impacts of industry 4.0:-

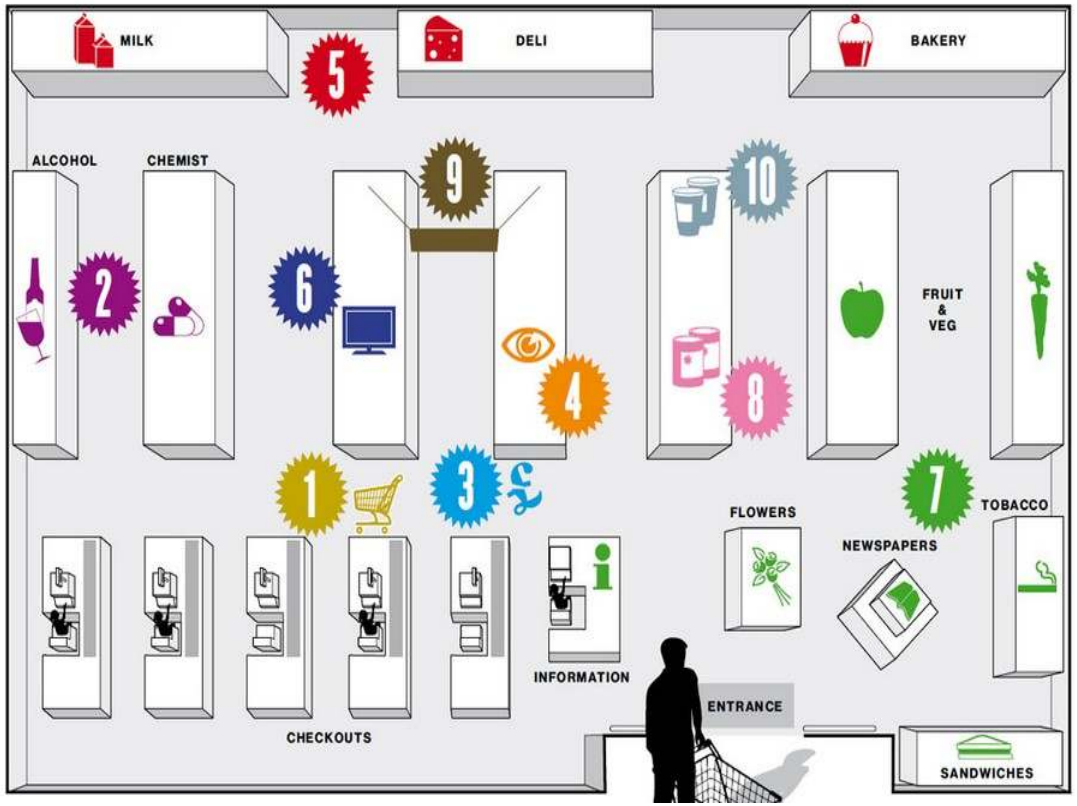
1. Services Innovation
2. Business Models
3. Manned and Unmanned
4. Digital Security



Big Data Example Cases

อ.ดนัยรัฐ ธนบดีธรรมจารี
Line ID: Danairat
FB: Danairat Thanabodithammachari
+668-1559-1446

Retails Store Use Cases

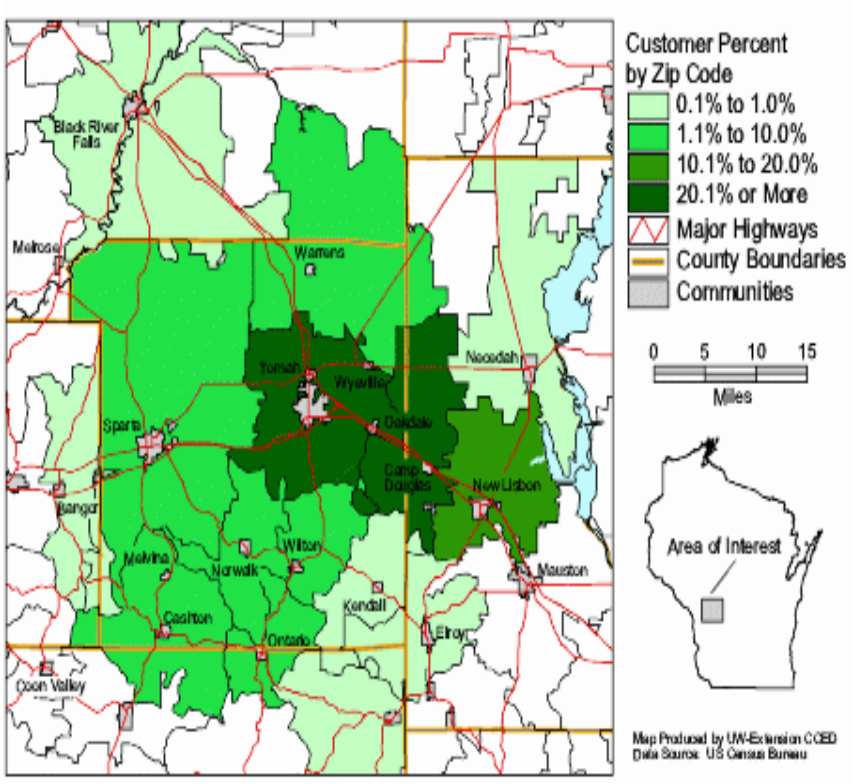


1. **Golden zones**
2. **More expensive items with bigger margins**
3. **Endcaps**
4. **Buy level**
5. **Traffic builder**
6. **Action alley**
7. **Front of shop**
8. **Signpost brands**
9. **Hanging signs and shelf signs**
10. **Range reduction**
    Less can be more. Average household uses 300 products in a year

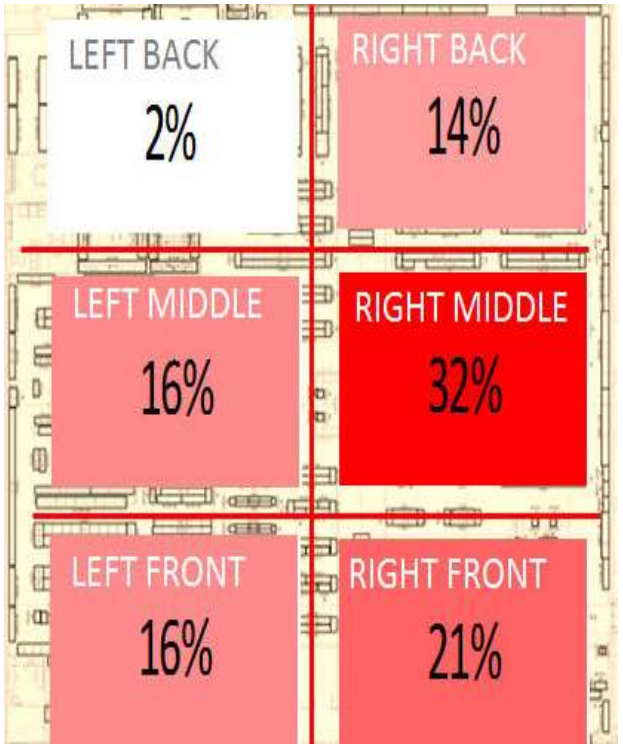
independent.co.uk

Retails Store Use Cases

Customer Density



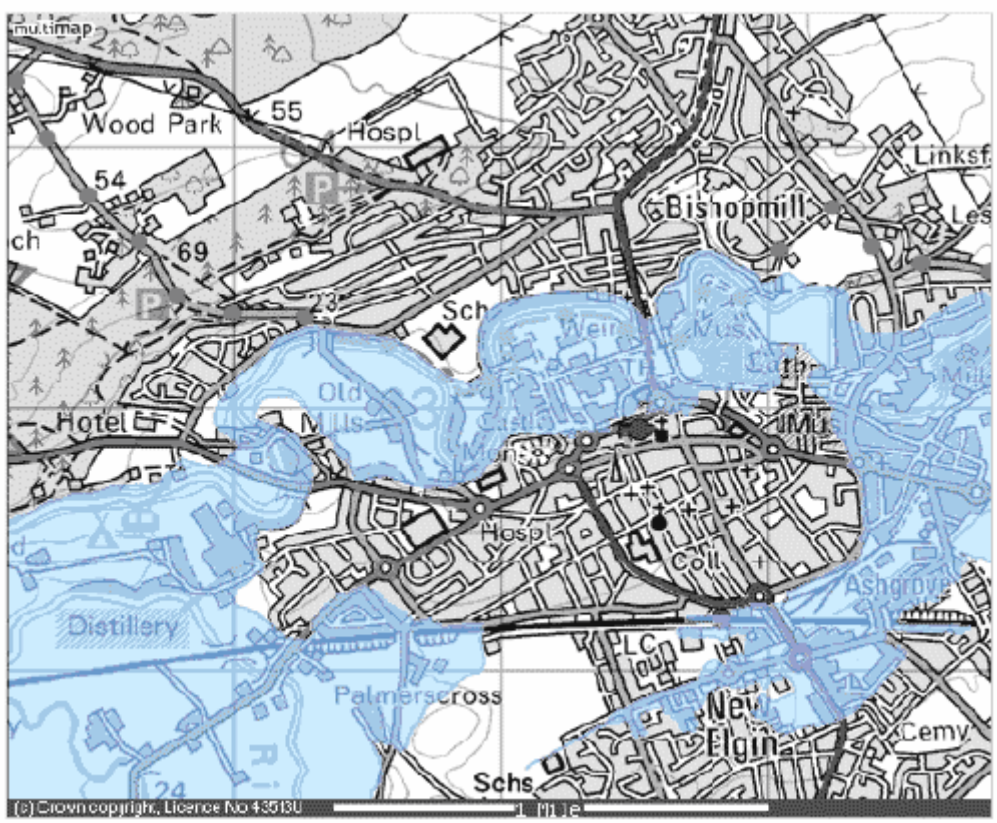
Store Layout Profits



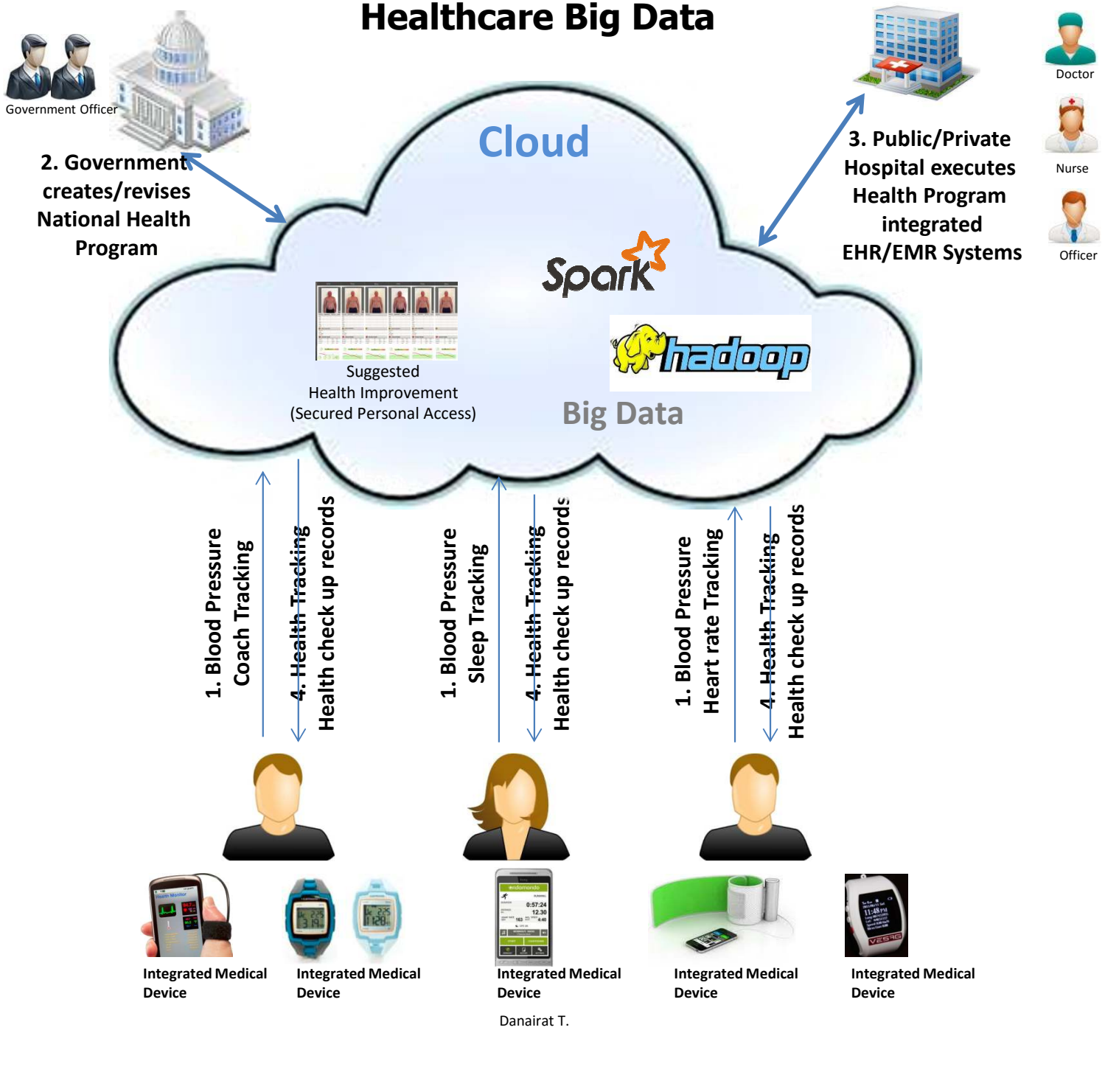
independent.co.uk

การเตือนภัยพิบัติ

Sensor ปริมาณน้ำฝน น้ำในเขื่อน ปริมาตรเขื่อน, แสดงแนวโน้มสภาวะนำท่วม
Nearly Real-Time



Healthcare Big Data



Danairat T.

Opportunity and Market Outlook

SECTOR	EXAMPLE APPLICATIONS	MAJOR DRIVER
Smart buildings	Automated monitoring of heating, ventilation and cooling	Reduced energy costs
Smart cities	Street lights that dim when roads are empty	Cost savings
Automotive	Emergency calling and accident alerts	
Leisure	Leisure vehicle and boat tracking	Safety and security
Consumer electronics	Connected satellite navigation devices to monitor traffic jams	Production innovation
Health	Remote monitoring of patients and personal health monitoring	Cheaper, home-based care
Utilities	Smart meters and energy demand response	Regulatory requirement
Transportation and logistics	Fleet optimization and supply-chain tracking and tracing	Cost savings
Retail	Wireless payments	Retail innovation
Manufacturing	Predictive maintenance through improved system monitoring	Reduced maintenance costs
Construction	Monitoring usage of equipment to improve efficiency and cut fuel usage	Cost savings
Agriculture and extraction	Remote monitoring of farm or mining operations and equipment	Proactive maintenance
Emergency services and national security	Disaster response and critical infrastructure protection	Faster response times

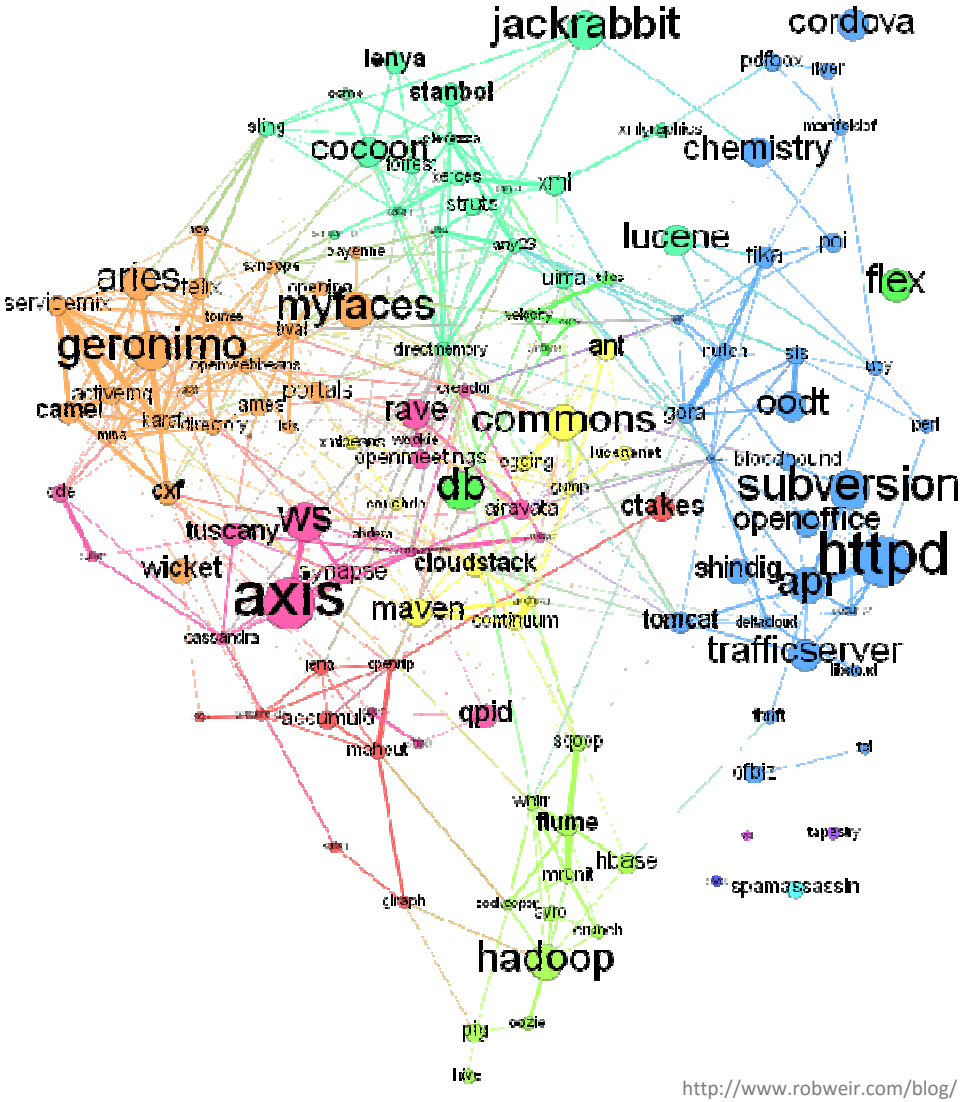
Source: Machina Research

Apache Software Foundation?

อ.ดนัยรัฐ ธนบดีธรรมจารี
Line ID: Danairat
FB: Danairat Thanabodithammachari
+668-1559-1446

Apache Software Foundation

Established in 1999, the ASF is a US 501(c)(3) charitable organization, funded by individual donations and corporate sponsors. Our all-volunteer board oversees more than 350 leading Open Source projects, including Apache HTTP Server -- the world's most popular Web server software.



http://www.robweir.com/blog/2013/05/mapping-apache.html

Apache Open Source Project

Over **300** Open Source Projects

Over **25** different programming languages

http://www.slideshare.net/WCGWorld/matt-franklin-apache-software-geekfest

Apache Open Source Project

Over **300** Open Source Projects

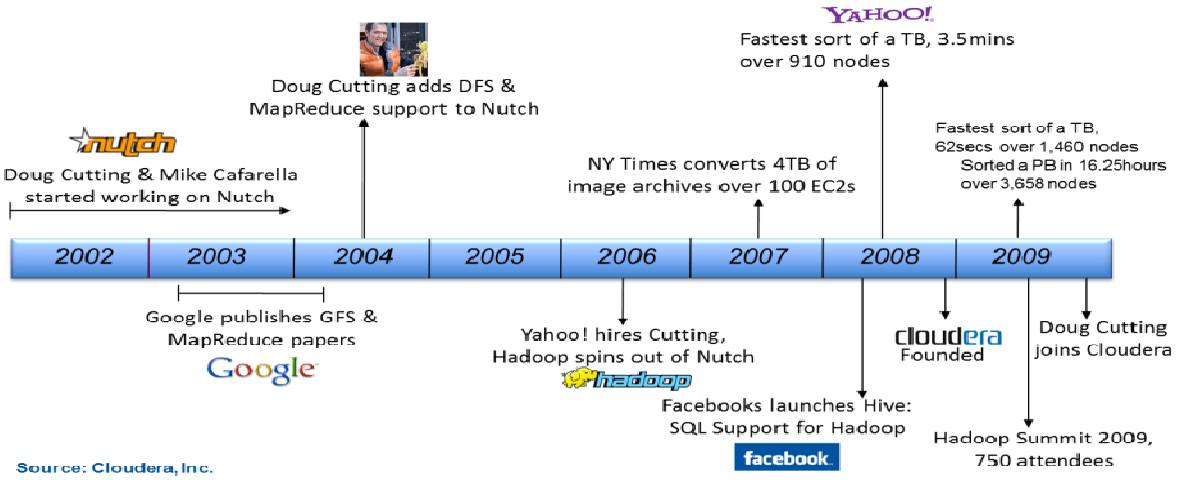
Over **25** different programming languages

http://www.slideshare.net/WCGWorld/matt-franklin-apache-software-geekfest

Apache Hadoop

อ.ดนัยรัฐ ธนบดีธรรมจารี
Line ID: Danairat
FB: Danairat Thanabodithammachari
+668-1559-1446

Hadoop History



- **2003:** Google publishes GFS and MapReduce papers
- **2004:** Cutting adds DFS & MapReduce support to Nutch
- **2006:** Yahoo! hires Cutting
- **2007:** NY Times converts 4TB of archives over 100 EC2s
- **2008:** Yahoo does fastest sort of a TB, 3.5mins over 910 nodes
- **2009:** First Hadoop Summit

apache.org/hadoop/

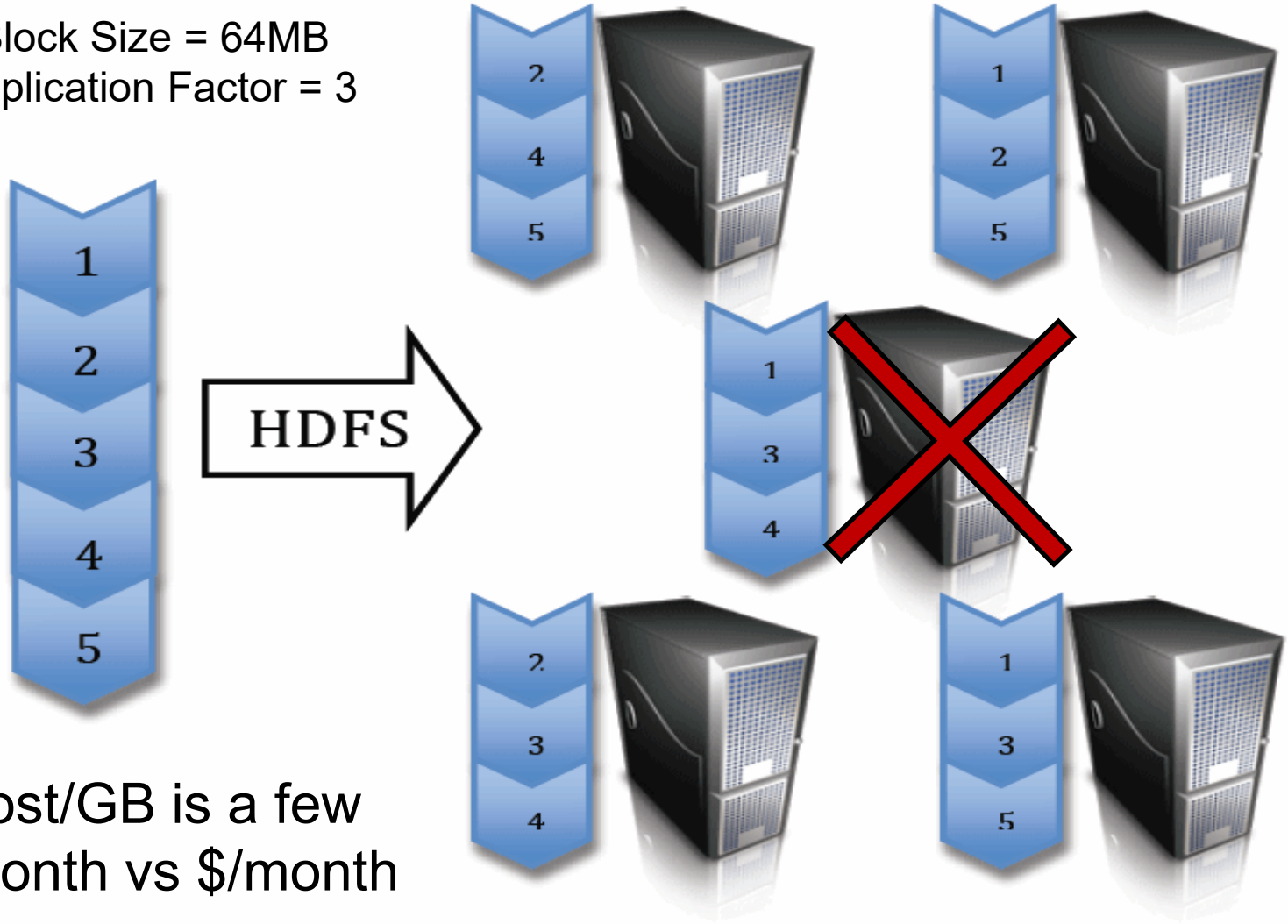
What is Hadoop?

- Am apache open source scalable fault-tolerant distributed system for data storage and processing:-
  - **HDFS**: Self-Healing High-Bandwidth Clustered Storage
  - **YARN**: Resources Management
  - **MapReduce**: Fault-Tolerant Distributed Processing
  - **Common**: Common utilities for Hadoop modules.
- Support both structured and complex data
- Large ecosystem eg. Spark, Cassandra, Ambari, …

apache.org/hadoop/

HDFS: Hadoop Distributed File System

Block Size = 64MB
Replication Factor = 3

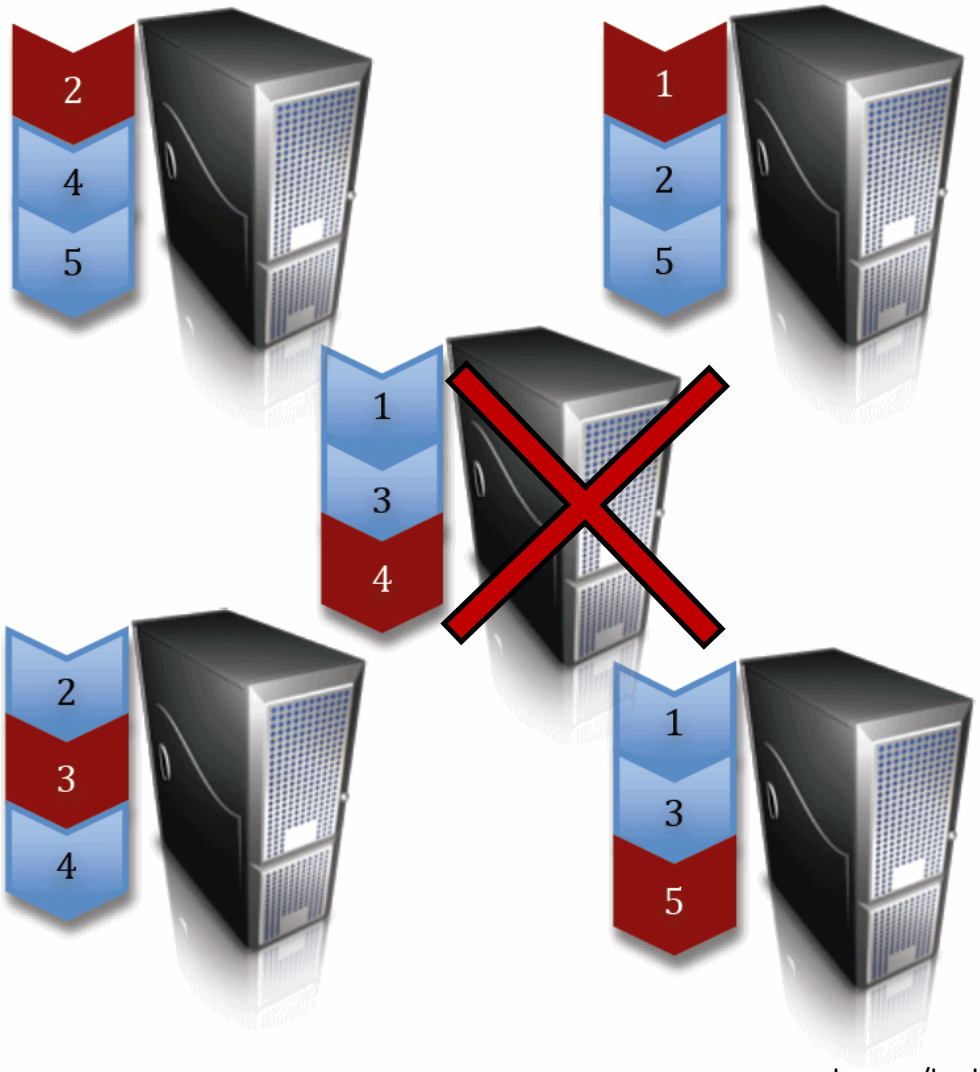


Cost/GB is a few ¢/month vs $/month

apache.org/hadoop/

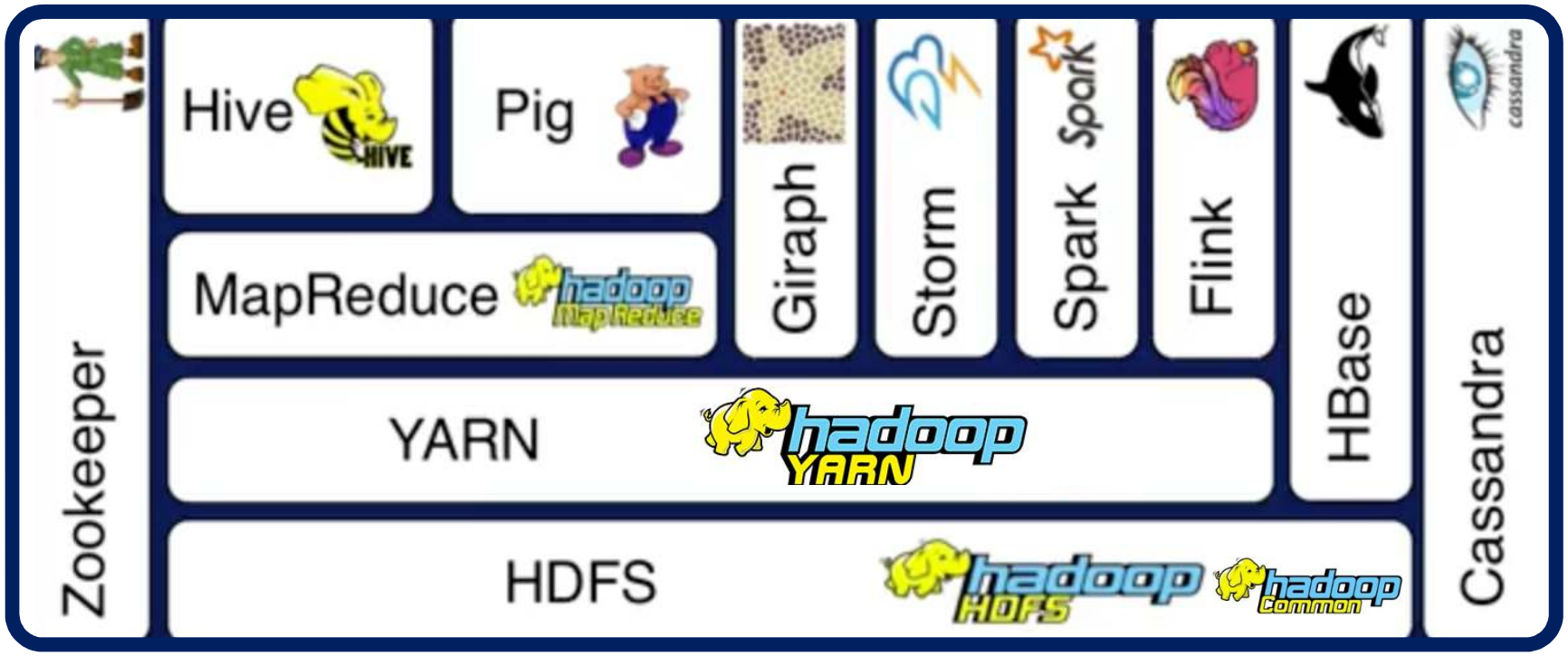
MapReduce: Distributed Processing

Hadoop takes advantage of HDFS' data distribution strategy to push work out to many nodes in a cluster. This allows analyses to run in parallel and eliminates the bottlenecks imposed by monolithic storage systems.



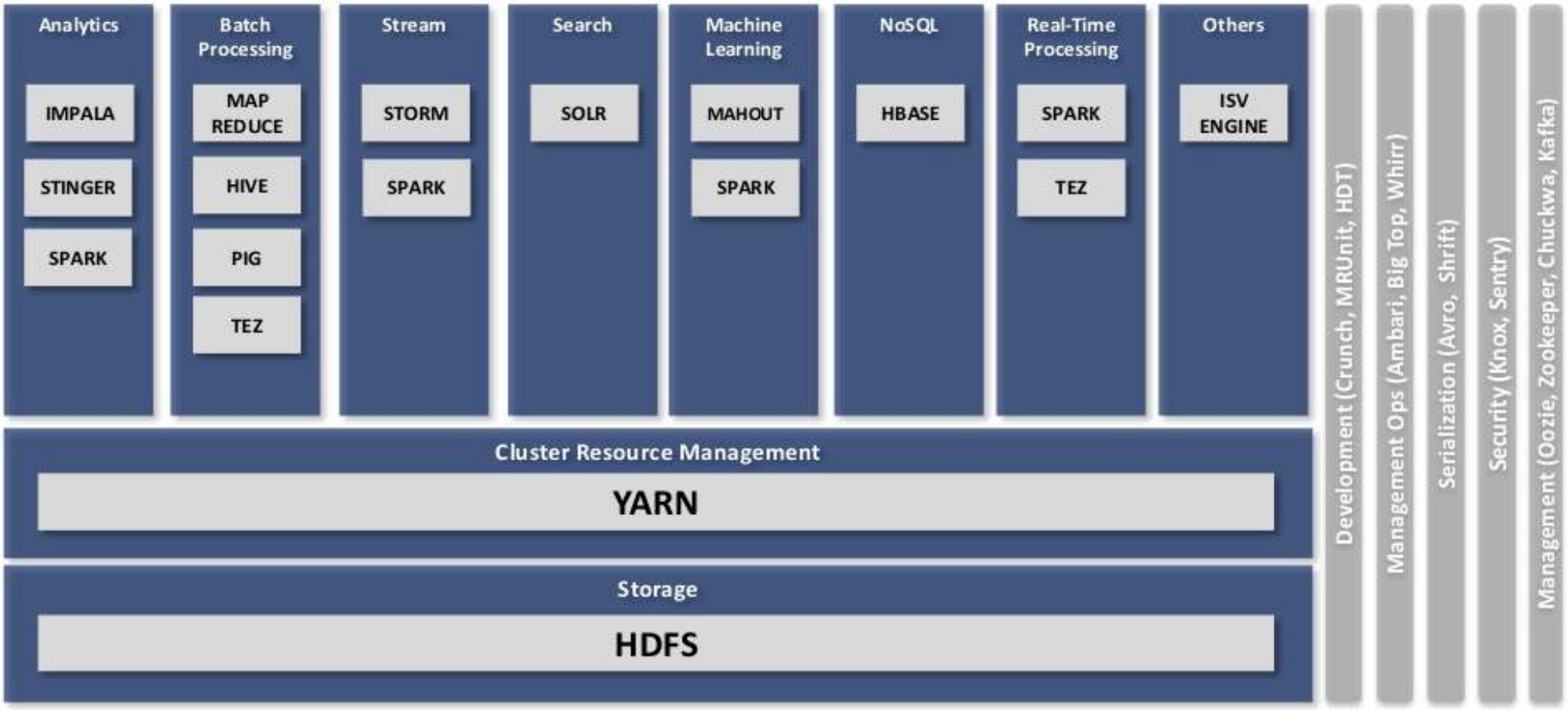
apache.org/hadoop/

Hadoop 2 Ecosystem



http://3v4n4r34l.blogspot.com/2016/04/coursera-ucsd-big-data-specialization.html

Hadoop 2 Ecosystem



http://incubator.apache.org/projects/

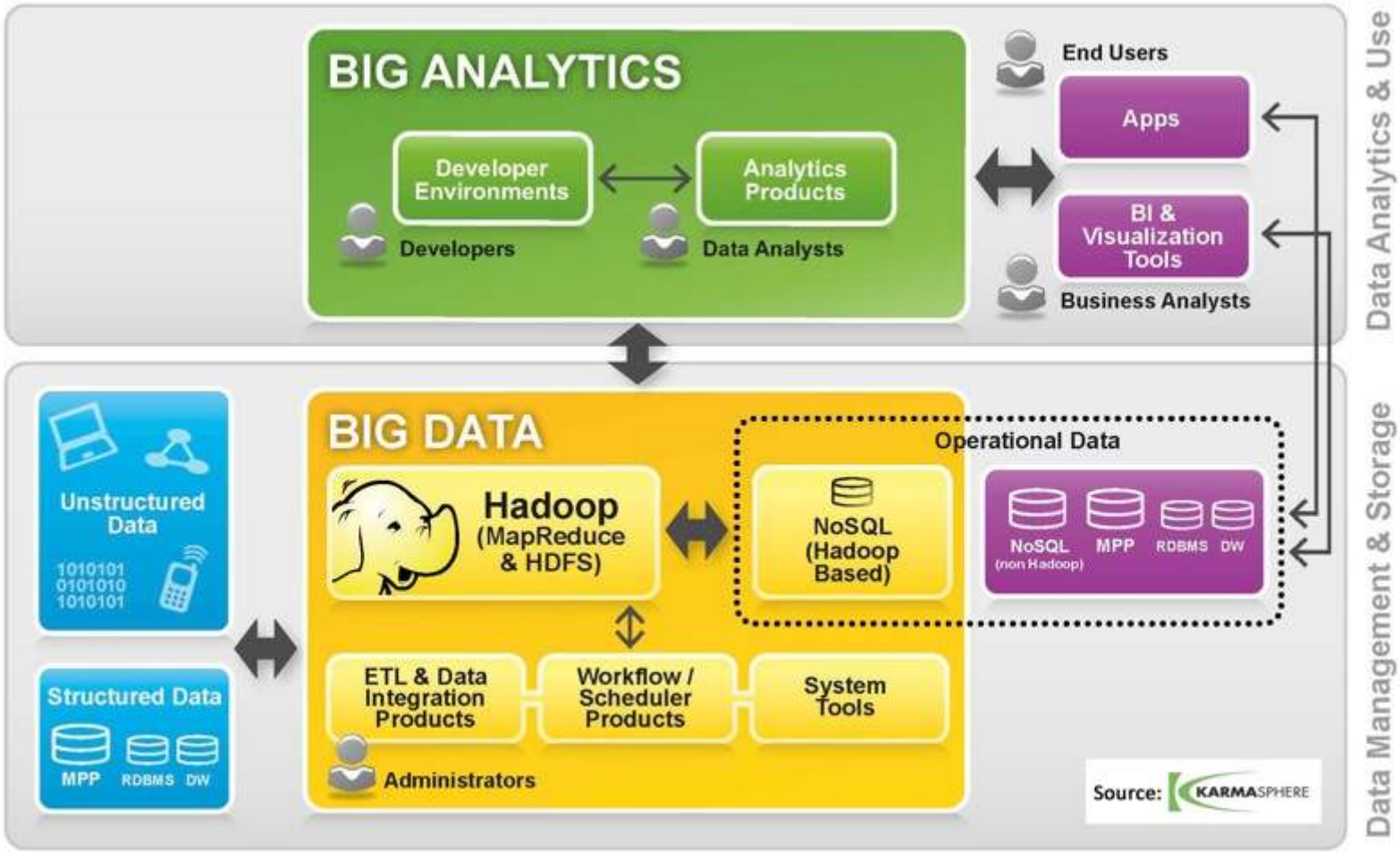
The Evolving Hadoop Landscape

Components	Description
mahout, R	Data Mining/machine learning tools used against Hadoop data to detect patterns and trends
Pig	Scripting language for analyzing large datasets. Compiles to MapReduce jobs
MapReduce (Hadoop 1.0), YARN (Hadoop 2.0)	Programming model for processing large data sets. YARN performs overall resource mgmt
OOZIE	A workflow scheduler tool to manage Hadoop MapReduce jobs
Sqoop, HIVE, APACHE DRILL	**Enable SQL for Hadoop data:** Sqoop - Data transfer between Hadoop and structured datastores. HIVE - data warehouse for Hadoop. Drill - open source, low latency SQL query engine for Hadoop and NoSQL.
ZooKeeper	Coordination of config, data, naming and synchronization of Hadoop projects

Components	Description
BigTop	Packaging services for Hadoop projects to ease testing and deployment
HBase	A non-relational, distributed database that runs on top of HDFS
Thrift, AVRO	Schema-based data serialization system using RPC calls
Solr, elasticsearch., nutch	Indexing and search tools for data stored in HDFS for Hadoop
Kafka, Storm	Collect, aggregate, and move streaming data from multiple sources into Hadoop
Spark	AppDev tool for Hadoop apps combining batch, streaming, and interactive analytics
Ambari, Hue	Monitoring & Management of Hadoop clusters and nodes

Big Data Platform & Big Data Analytics

Hadoop Technology



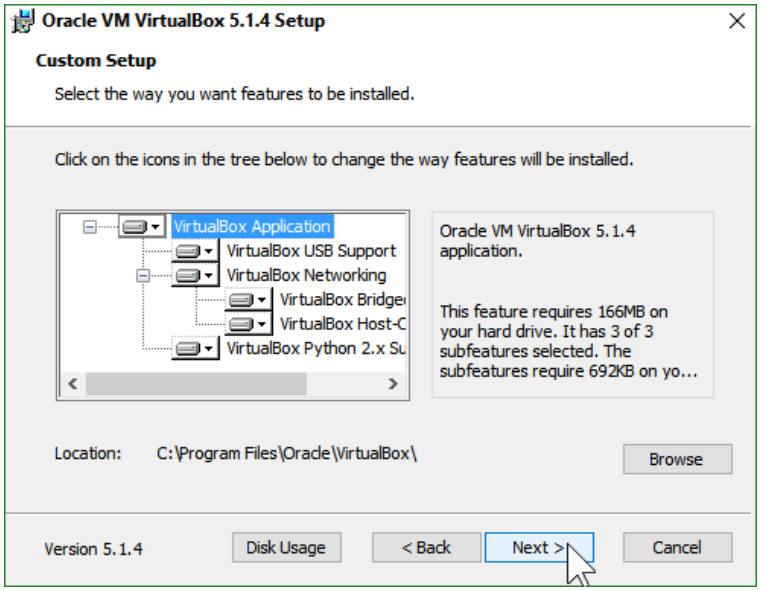
Hands On Labs

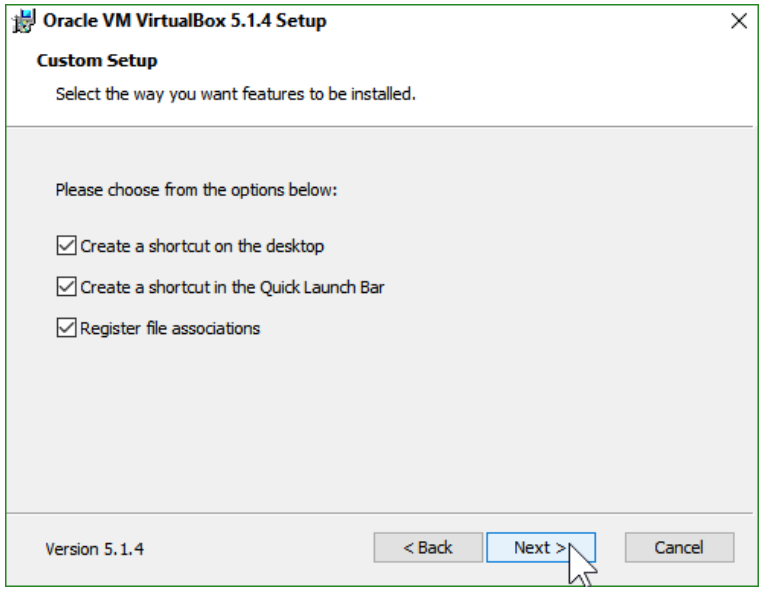
Installing Apache Hadoop 2.7.2

อ.ดนัยรัฐ ธนบดีธรรมจารี
Line ID: Danairat
FB: Danairat Thanabodithammachari
+668-1559-1446

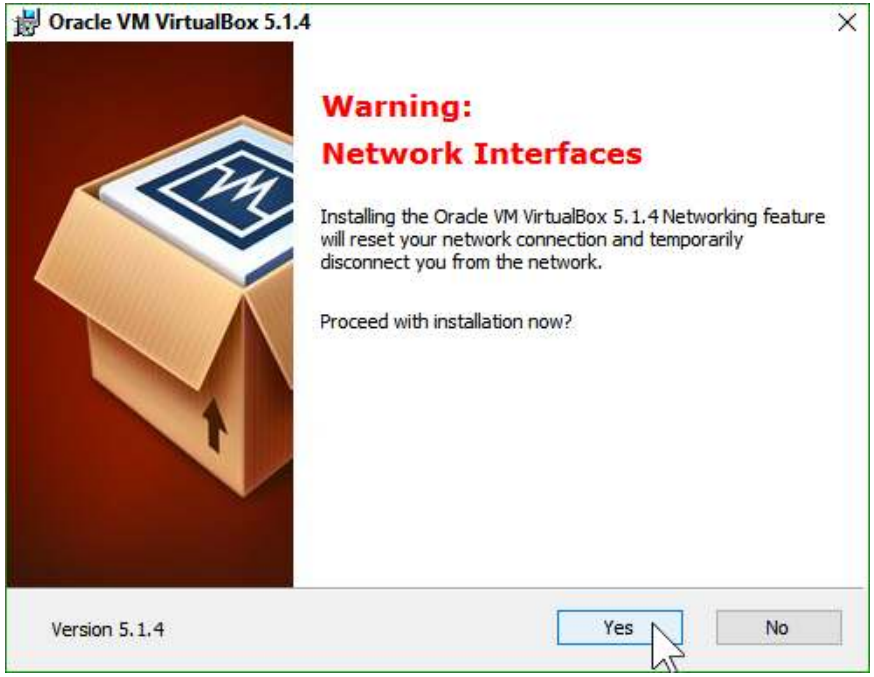
1. Install Virtual Box

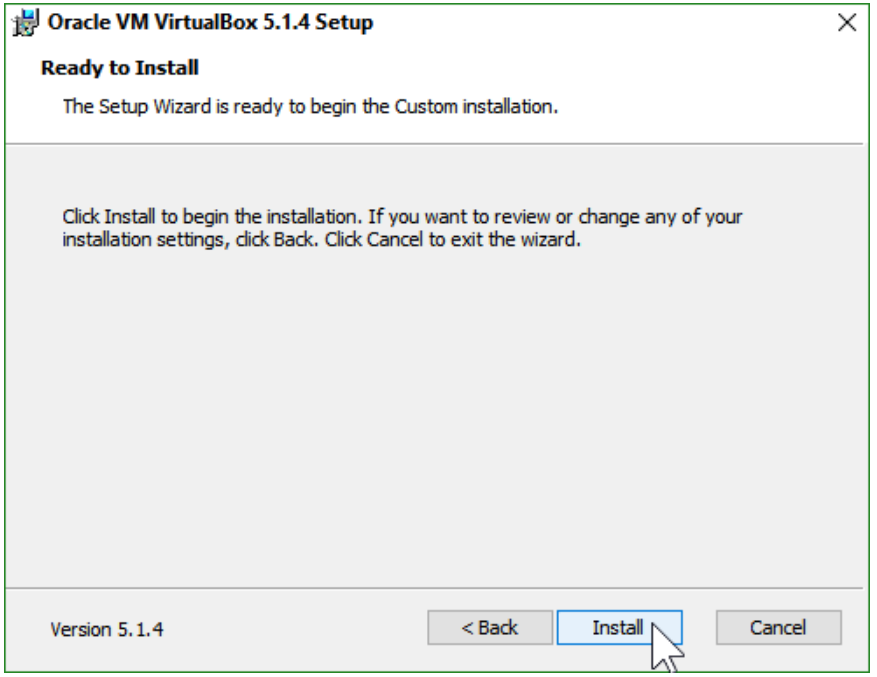
1. Install Virtual Box



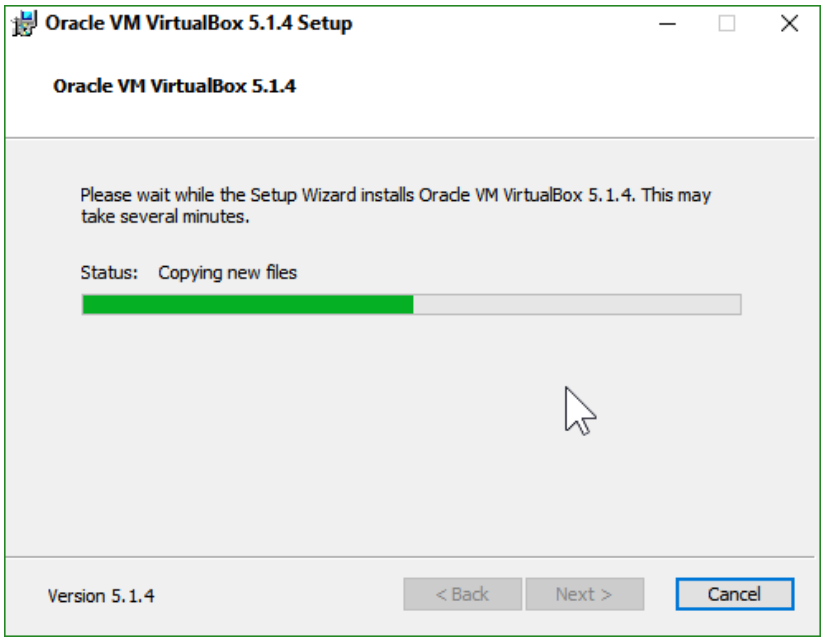


1. Install Virtual Box

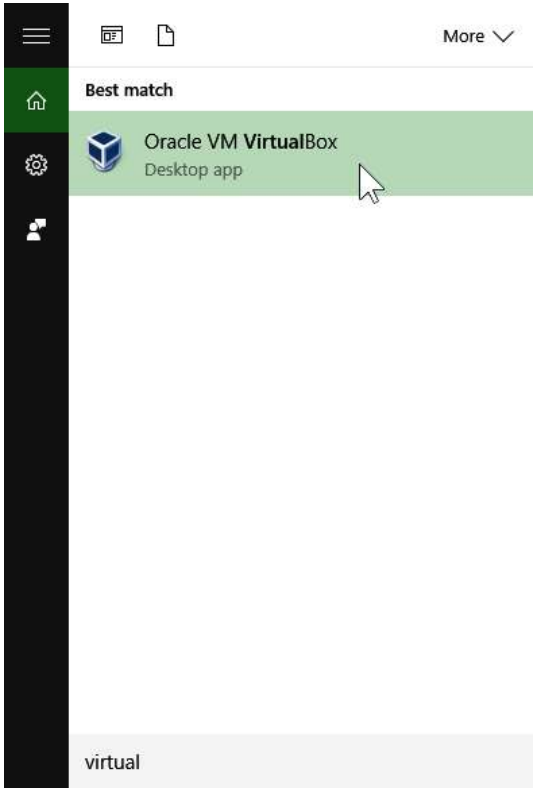


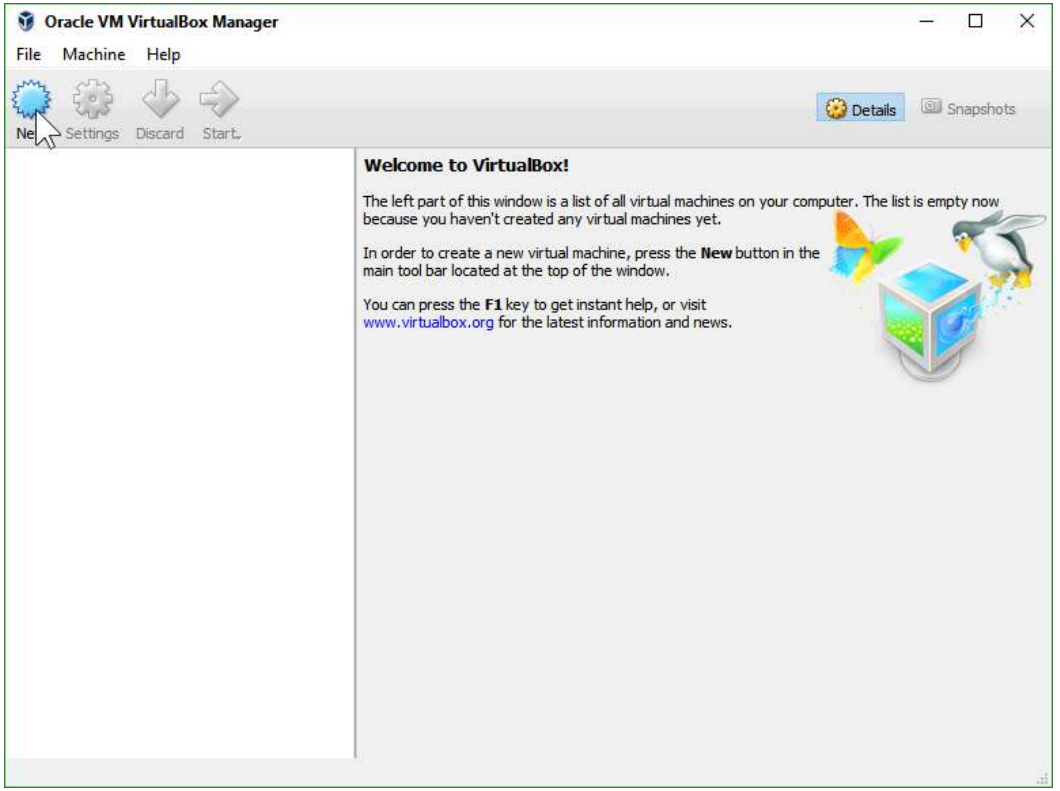


1. Install Virtual Box

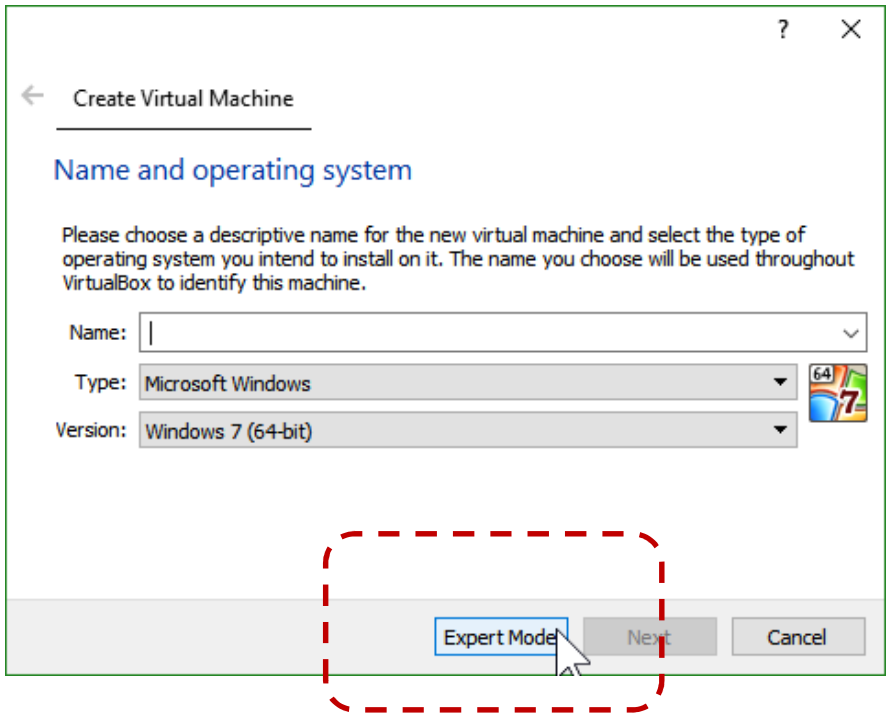


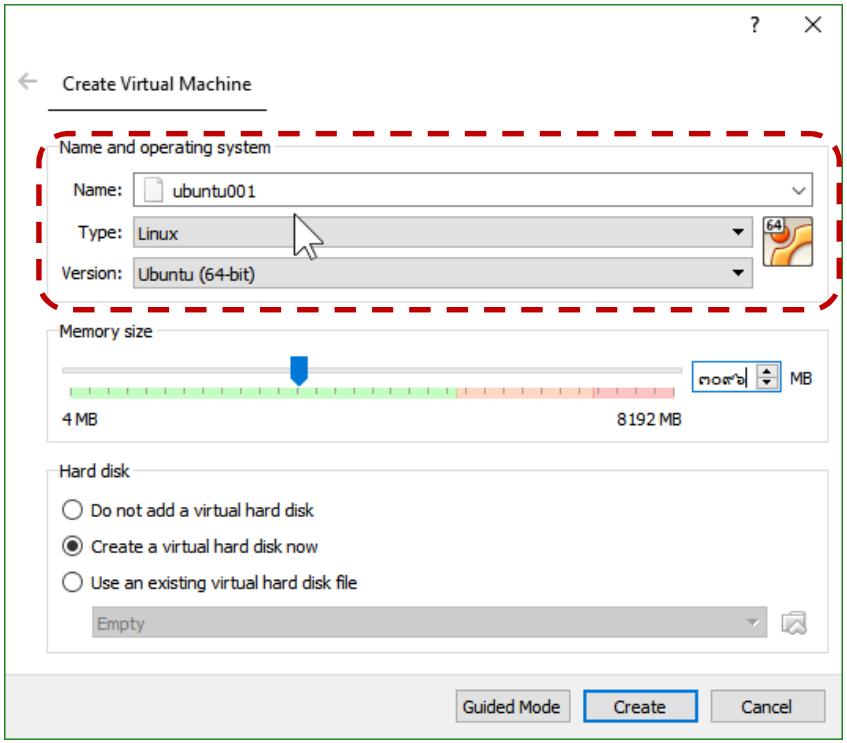
2. Start Ubuntu Linux Server



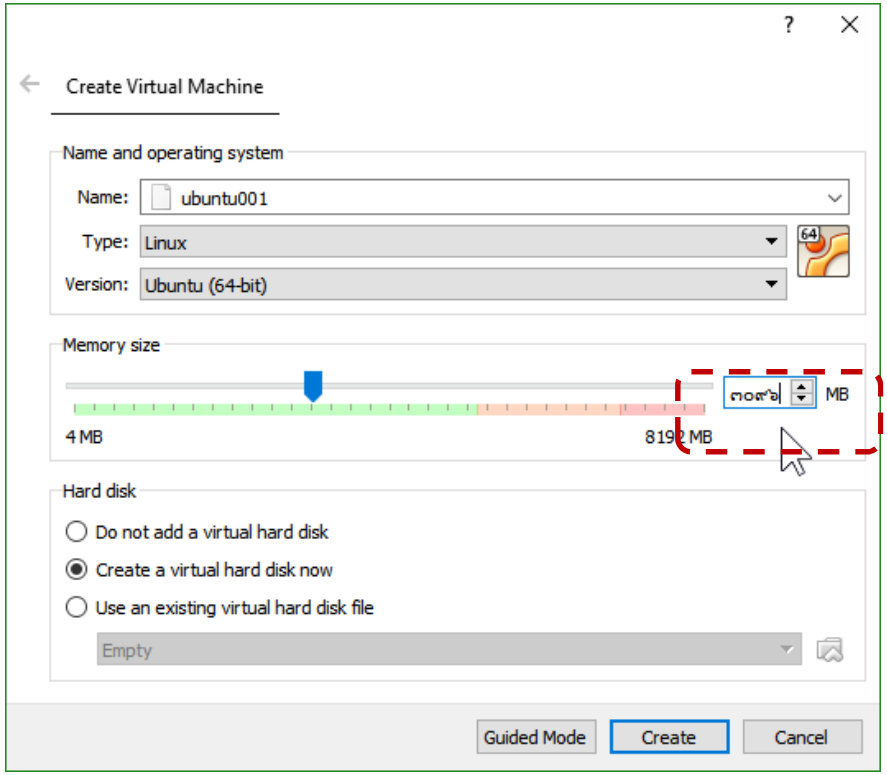


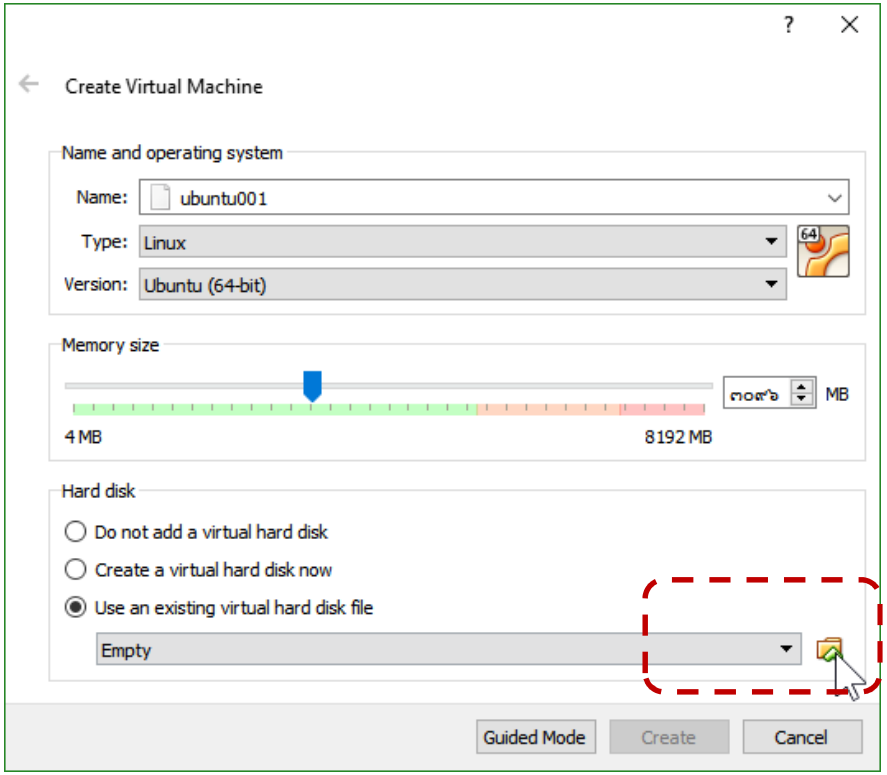
2. Start Ubuntu Linux Server



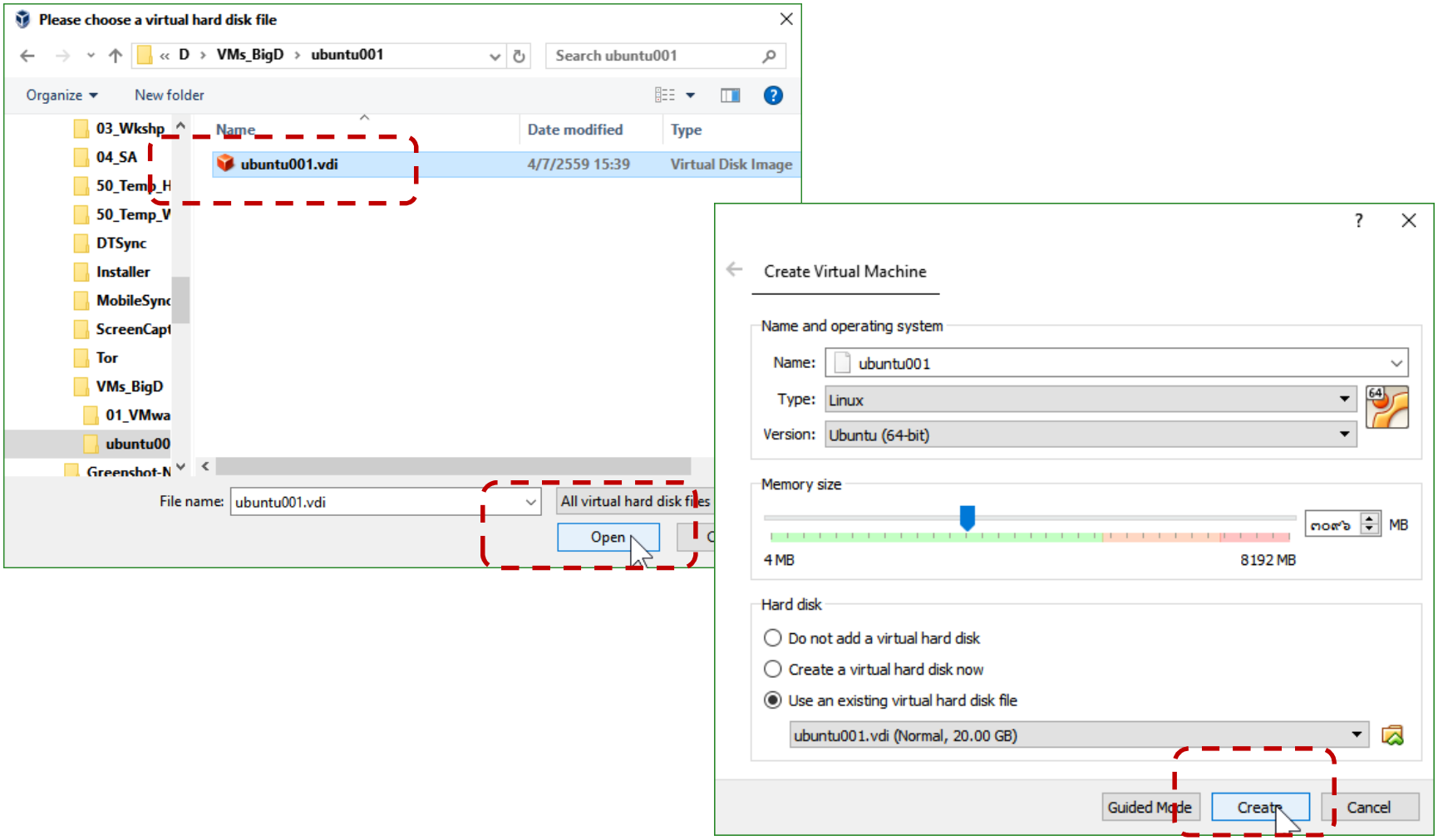


2. Start Ubuntu Linux Server

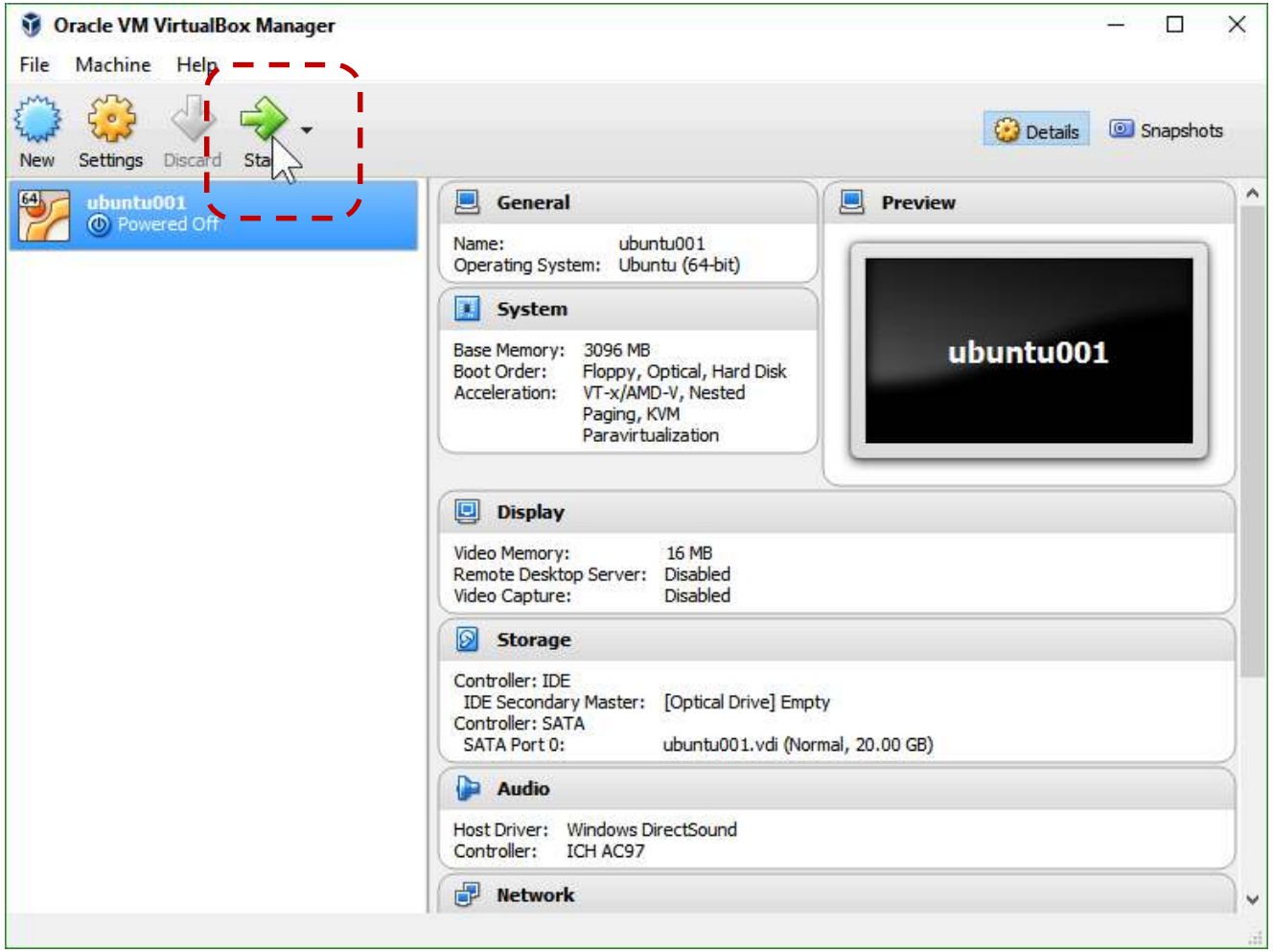




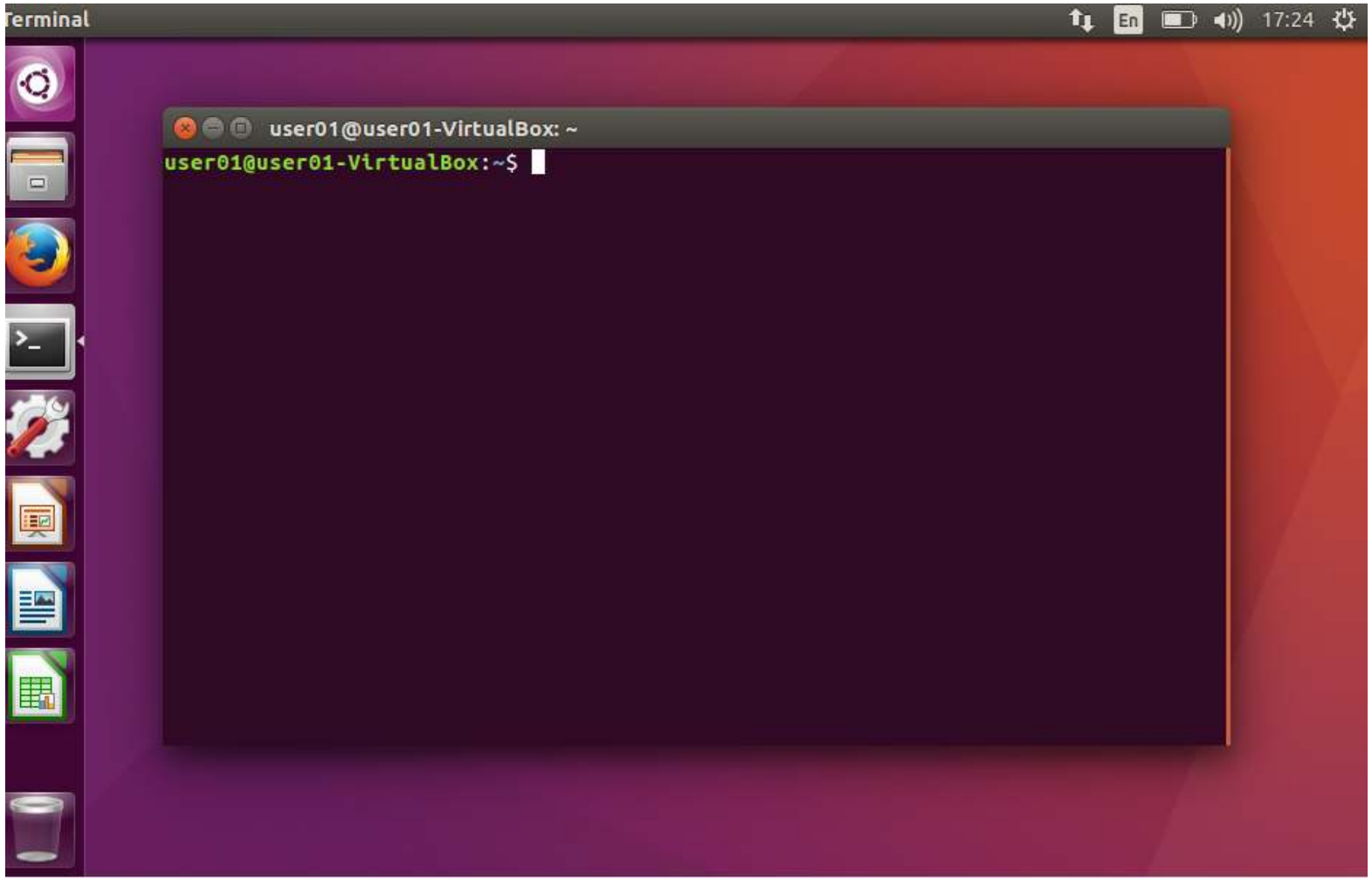
2. Start Ubuntu Linux Server



2. Start Ubuntu Linux Server

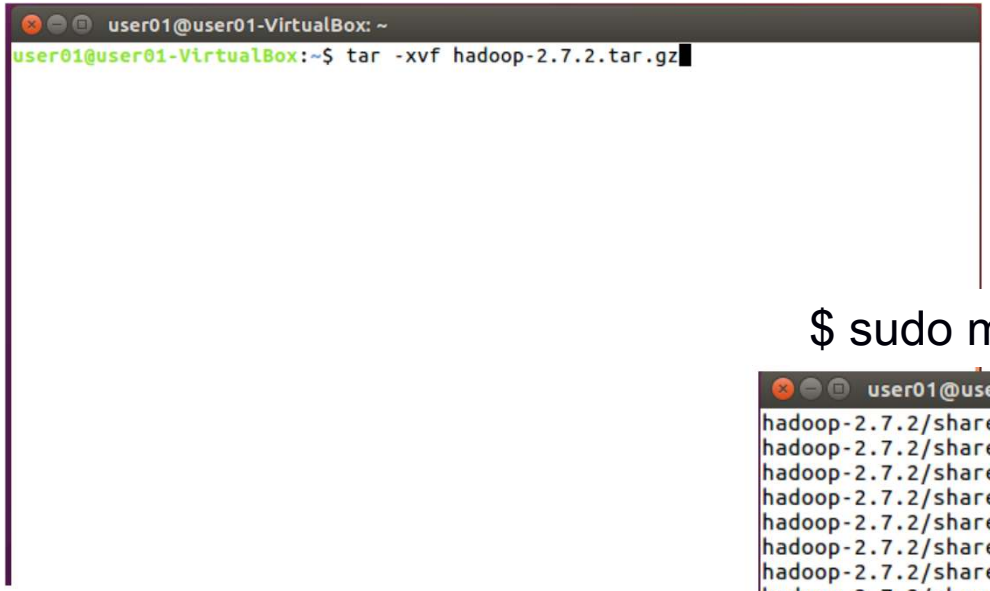


2. Start Ubuntu Linux Server



3. Setup Apache Hadoop 2.7.2

$ cd
$ tar -xvf hadoop-2.7.2.tar.gz



มีจุดที่นี่ มีเว้นวรรคที่นี่

$ sudo mv ./hadoop-2.7.2 /usr/local/hadoop

3. Setup Apache Hadoop 2.7.2

$ sudo nano ~/.bashrc

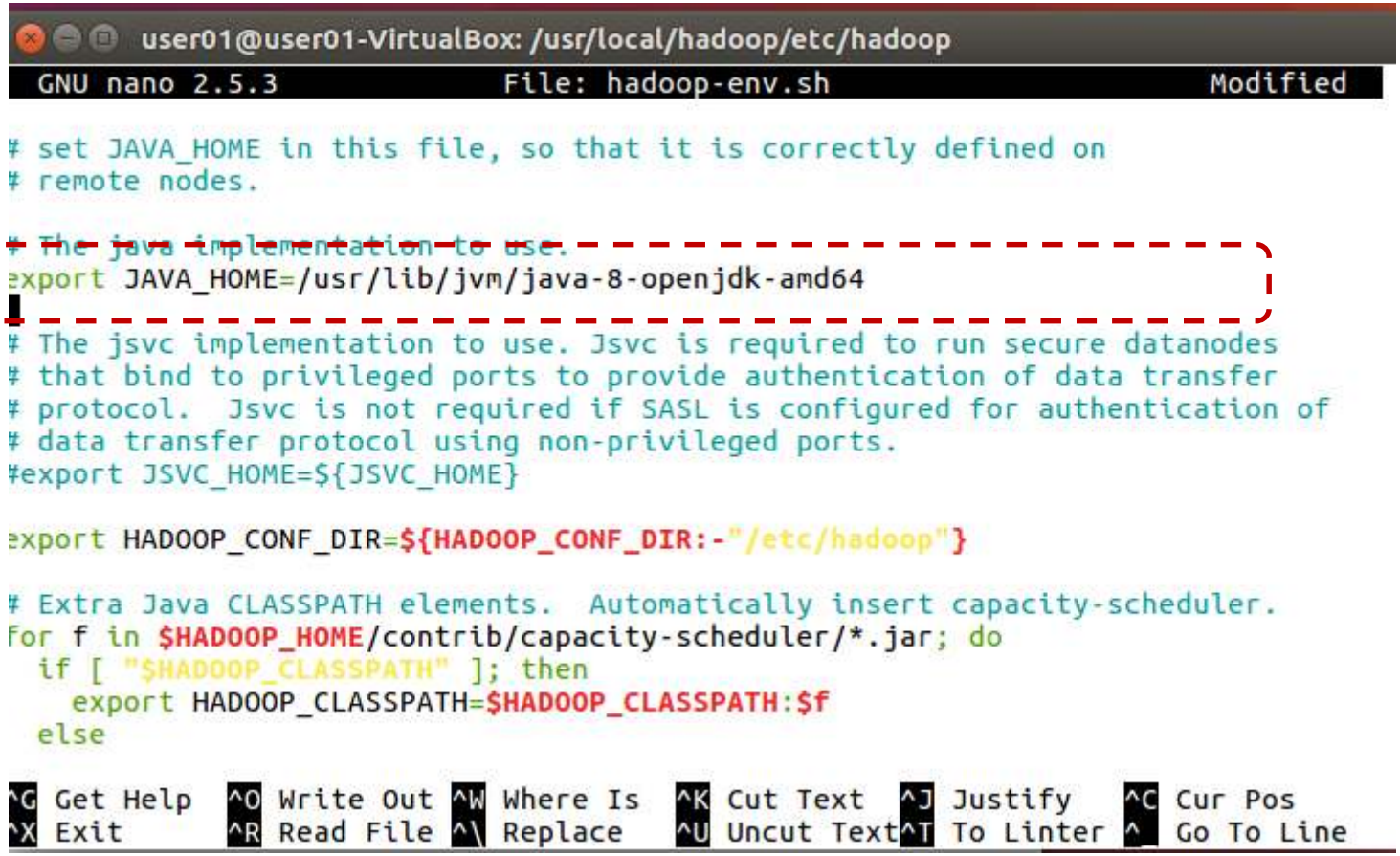
```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/
export PATH=$PATH:$JAVA_HOME/bin

export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
```

Ctrl + x and save file to save with exit the editor

1. Setup Apache Hadoop 2.7.2

```
$ source ~/.bashrc

$ sudo mkdir /var/log/hadoop

$ sudo chown -R user01:user01 /var/log/hadoop

$ cd /usr/local/hadoop/etc/hadoop

$ nano hadoop-env.sh
```

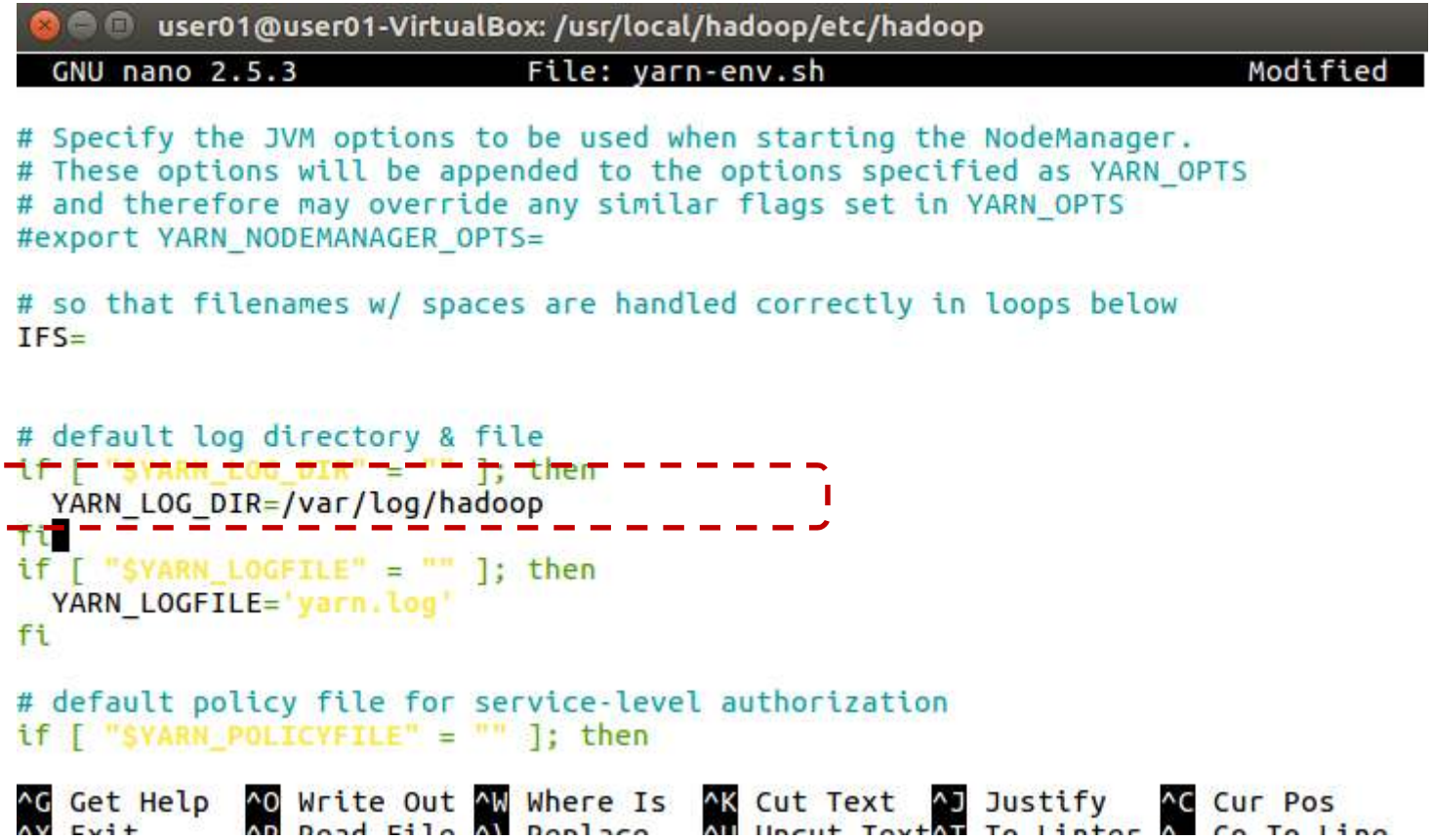
```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```


user01@user01-VirtualBox: /usr/local/hadoop/etc/hadoop
GNU nano 2.5.3 File: hadoop-env.sh Modified
set JAVA_HOME in this file, so that it is correctly defined on
remote nodes.
The java implementation to use.
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
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=${JSVC_HOME}
export HADOOP_CONF_DIR=${HADOOP_CONF_DIR:-"/etc/hadoop"}
# Extra Java CLASSPATH elements. Automatically insert capacity-scheduler.
for f in $HADOOP_HOME/contrib/capacity-scheduler/*.jar; do
if ["$HADOOP_CLASSPATH"]; then
export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:$f
else
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Linter ^_ Go To Line

3. Setup Apache Hadoop 2.7.2

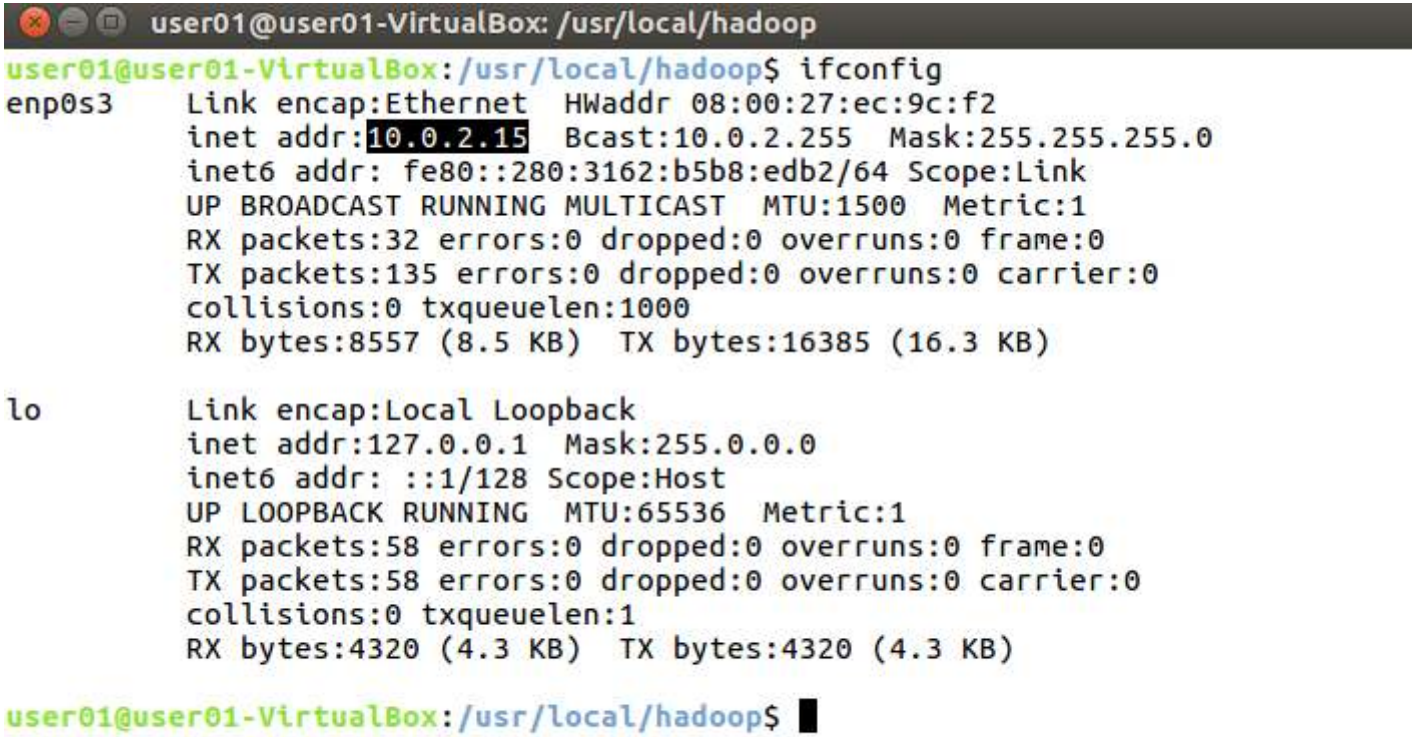
$ nano yarn-env.sh

YARN_LOG_DIR=/var/log/hadoop



3. Setup Apache Hadoop 2.7.2

$ ifconfig



วิธีดู IP Address ของเครื่องเรา

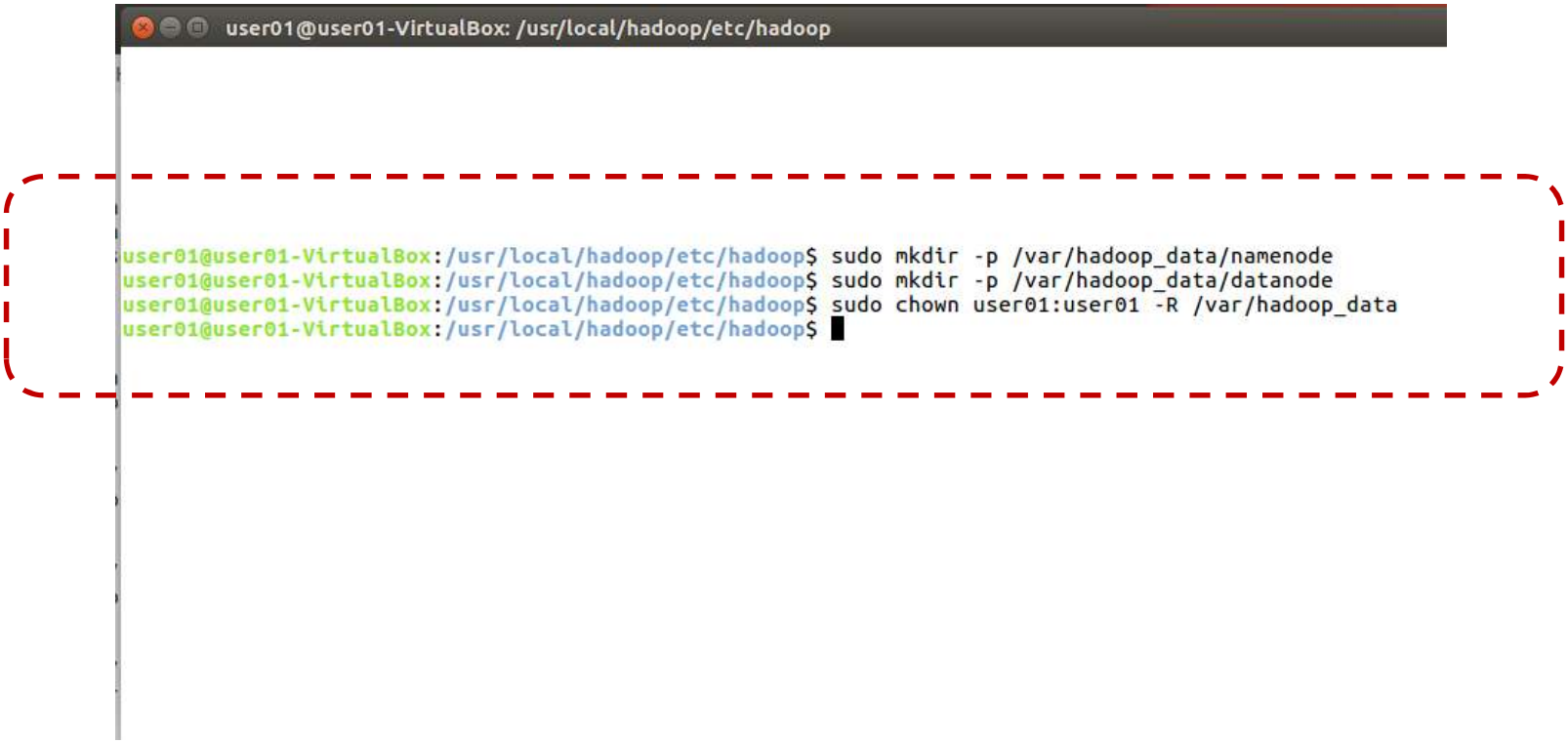
3. Setup Apache Hadoop 2.7.2

$ nano core-site.xml

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://10.0.2.15:9000</value>
  </property>
</configuration>
```

3. Setup Apache Hadoop 2.7.2

$ sudo mkdir -p /var/hadoop_data/namenode
$ sudo mkdir -p /var/hadoop_data/datanode
$ sudo chown user01:user01 -R /var/hadoop_data



3. Setup Apache Hadoop 2.7.2

$ nano hdfs-site.xml

```
<configuration>
        <property>
          <name>dfs.replication</name>
          <value>1</value>
        </property>
        <property>
          <name>dfs.namenode.name.dir</name>
          <value>file:/var/hadoop_data/namenode</value>
        </property>
        <property>
          <name>dfs.datanode.data.dir</name>
          <value>file:/var/hadoop_data/datanode</value>
        </property>
    </configuration>
```

3. Setup Apache Hadoop 2.7.2

$ nano yarn-site.xml

```
<configuration>
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>10.0.2.15</value>
  </property>
  <property>
    <name>yarn.resourcemanager.scheduler.address</name>
    <value>10.0.2.15:8030</value>
  </property>
  <property>
    <name>yarn.resourcemanager.resource-tracker.address</name>
    <value>10.0.2.15:8031</value>
  </property>
  <property>
    <name>yarn.resourcemanager.address</name>
    <value>10.0.2.15:8032</value>
  </property>
  <property>
    <name>yarn.resourcemanager.admin.address</name>
    <value>10.0.2.15:8033</value>
  </property>
  <property>
    <name>yarn.resourcemanager.webapp.address</name>
    <value>10.0.2.15:8088</value>
  </property>
  <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>
</configuration>
```

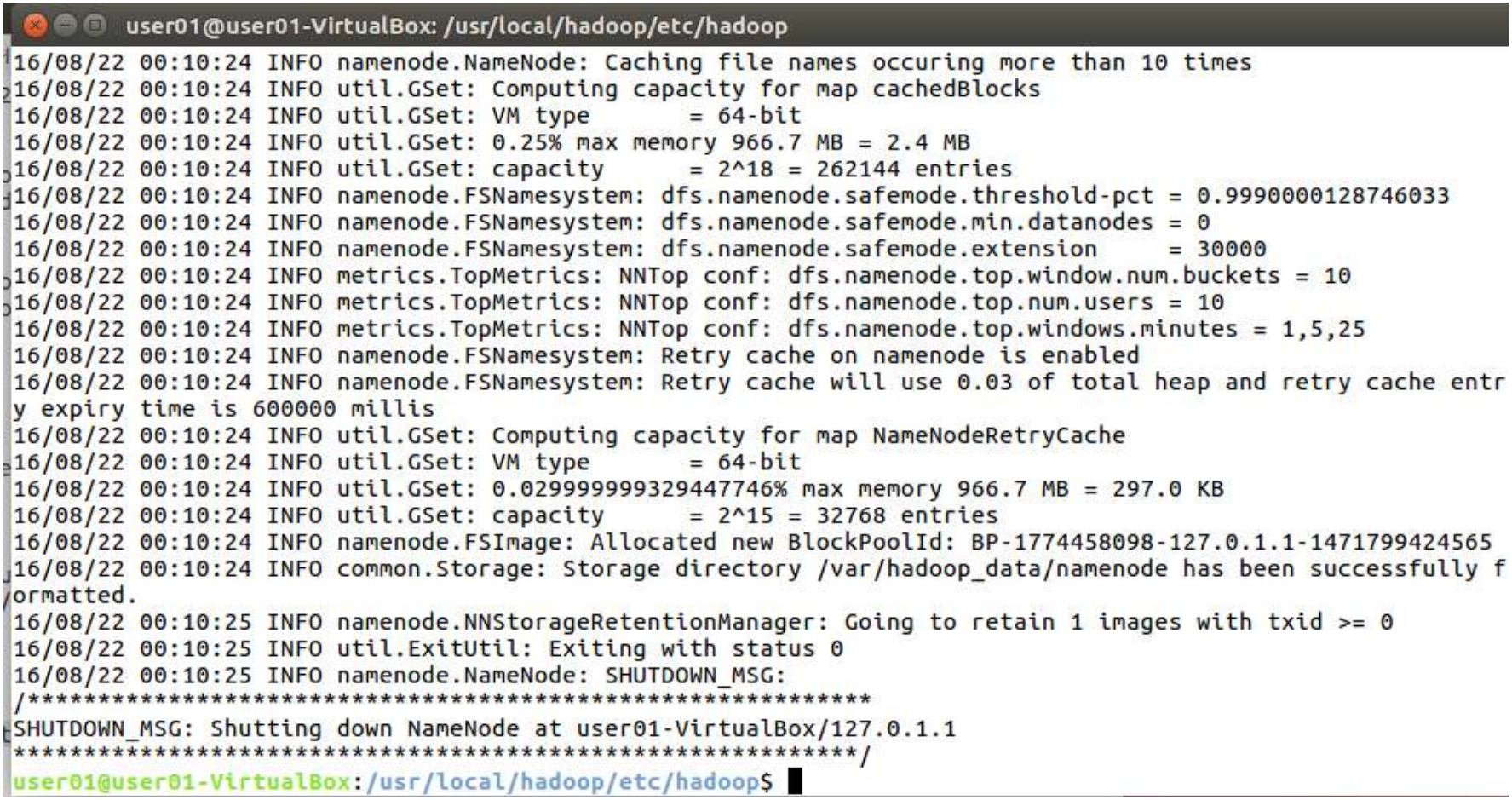
3. Setup Apache Hadoop 2.7.2

$ cp mapred-site.xml.template mapred-site.xml
$ nano mapred-site.xml

```
<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
</configuration>
```

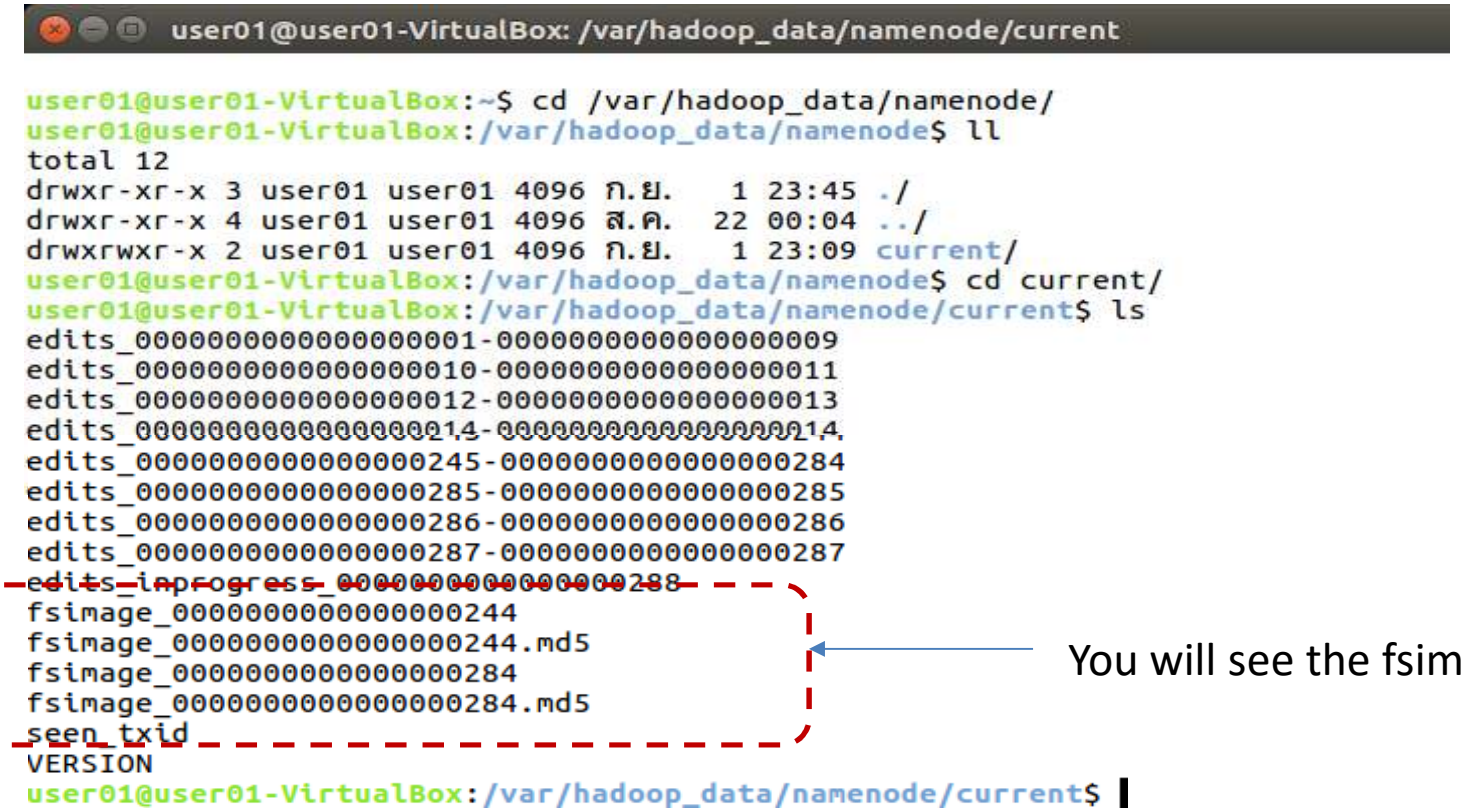
3. Setup Apache Hadoop 2.7.2

$ hdfs namenode -format



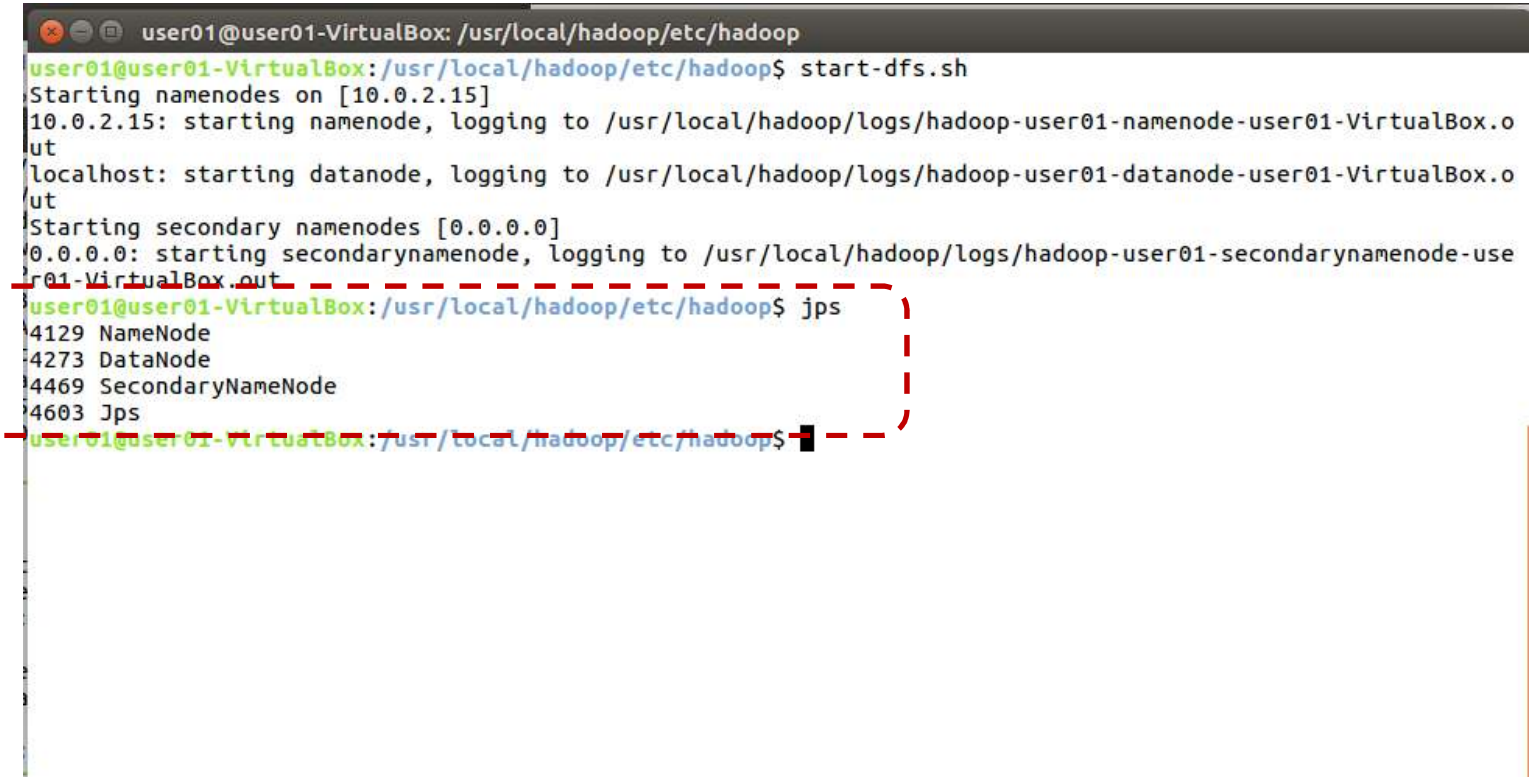
3. Setup Apache Hadoop 2.7.2

$ cd /var/hadoop_data/namenode/
$ ll
$ cd current/
$ ls



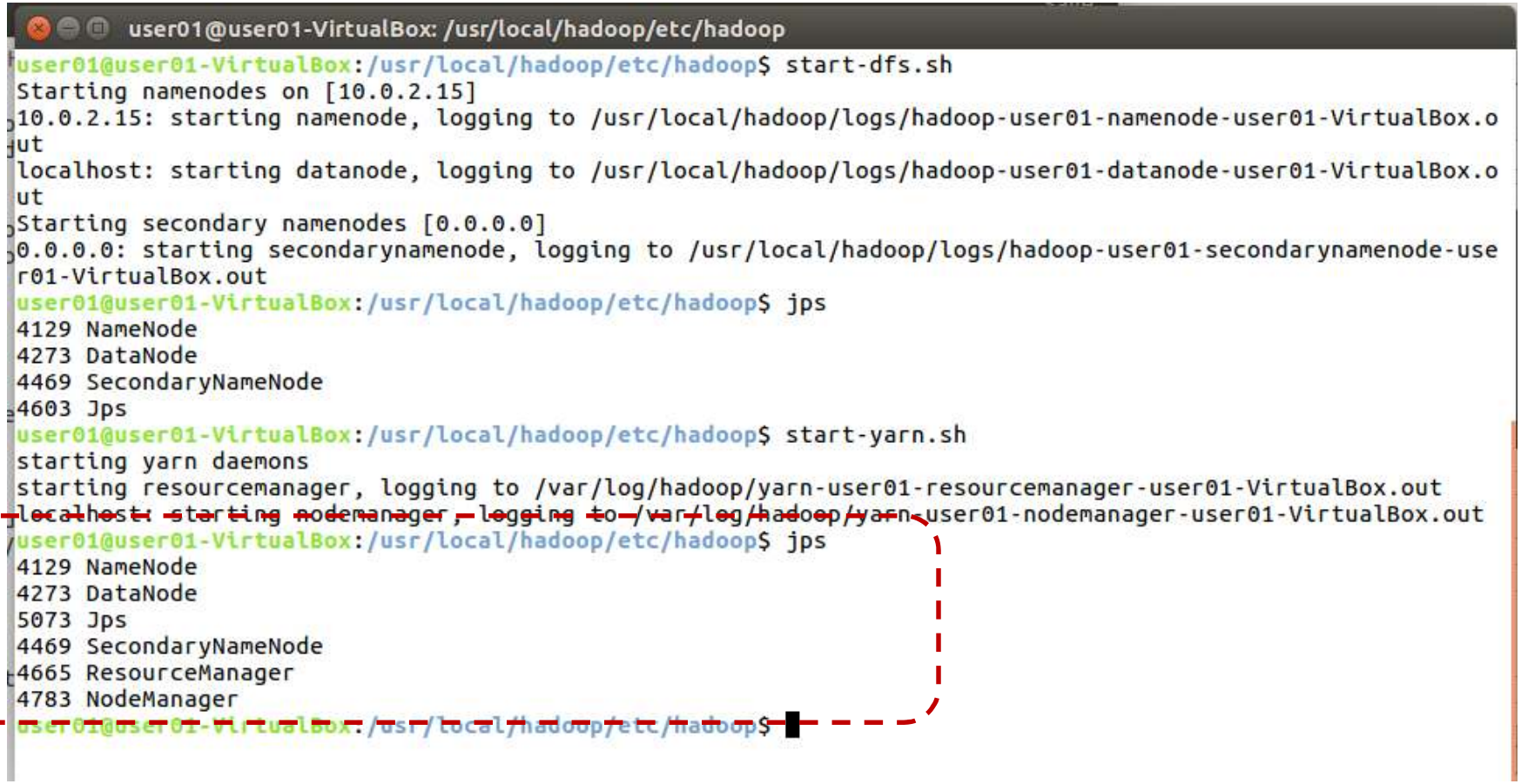
3. Setup Apache Hadoop 2.7.2

$ start-dfs.sh
ตอบ yes
$ jps



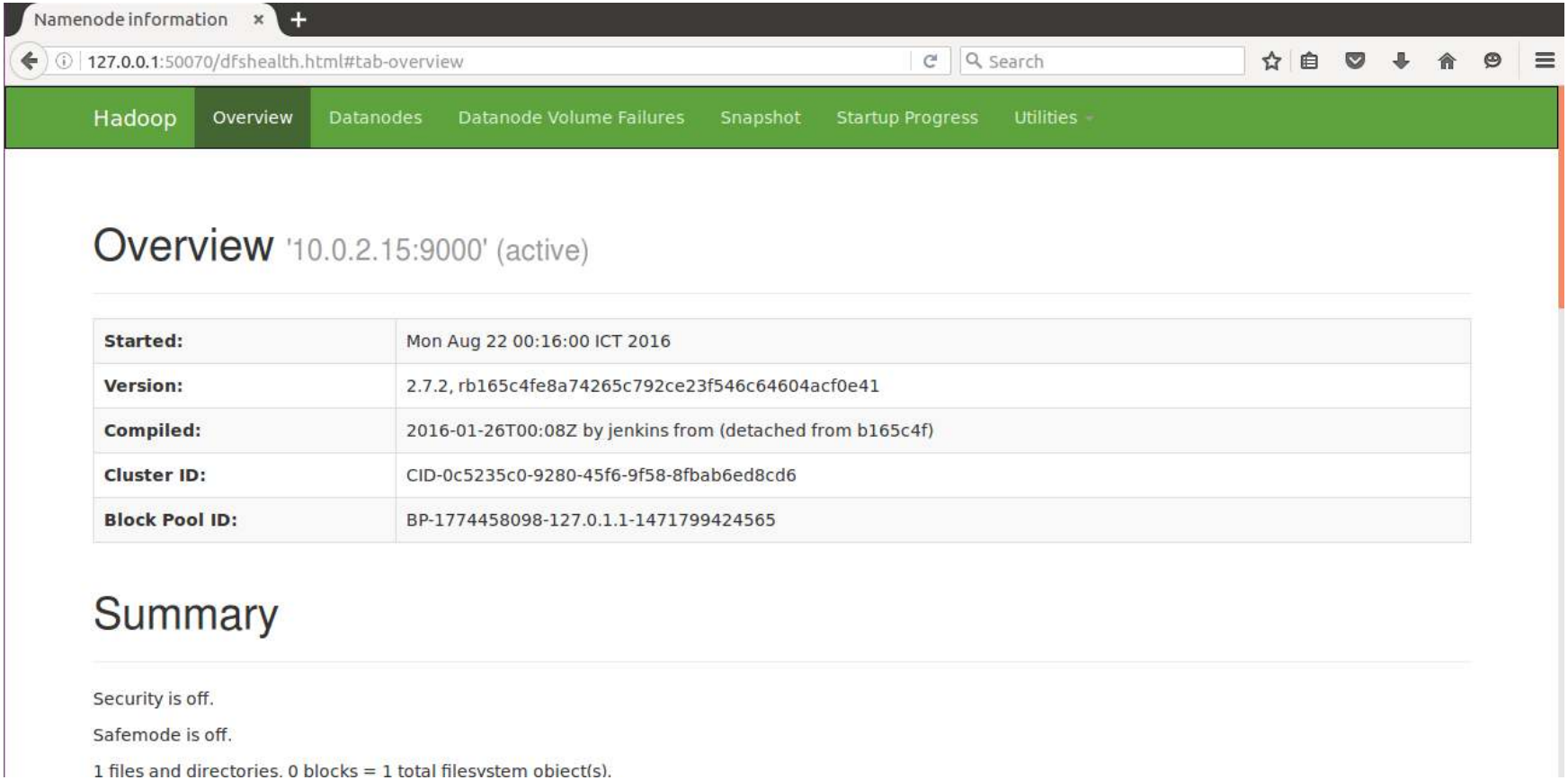
3. Setup Apache Hadoop 2.7.2

$ start-yarn.sh
$ jps



3. Setup Apache Hadoop 2.7.2

URL: http://127.0.0.1:50070

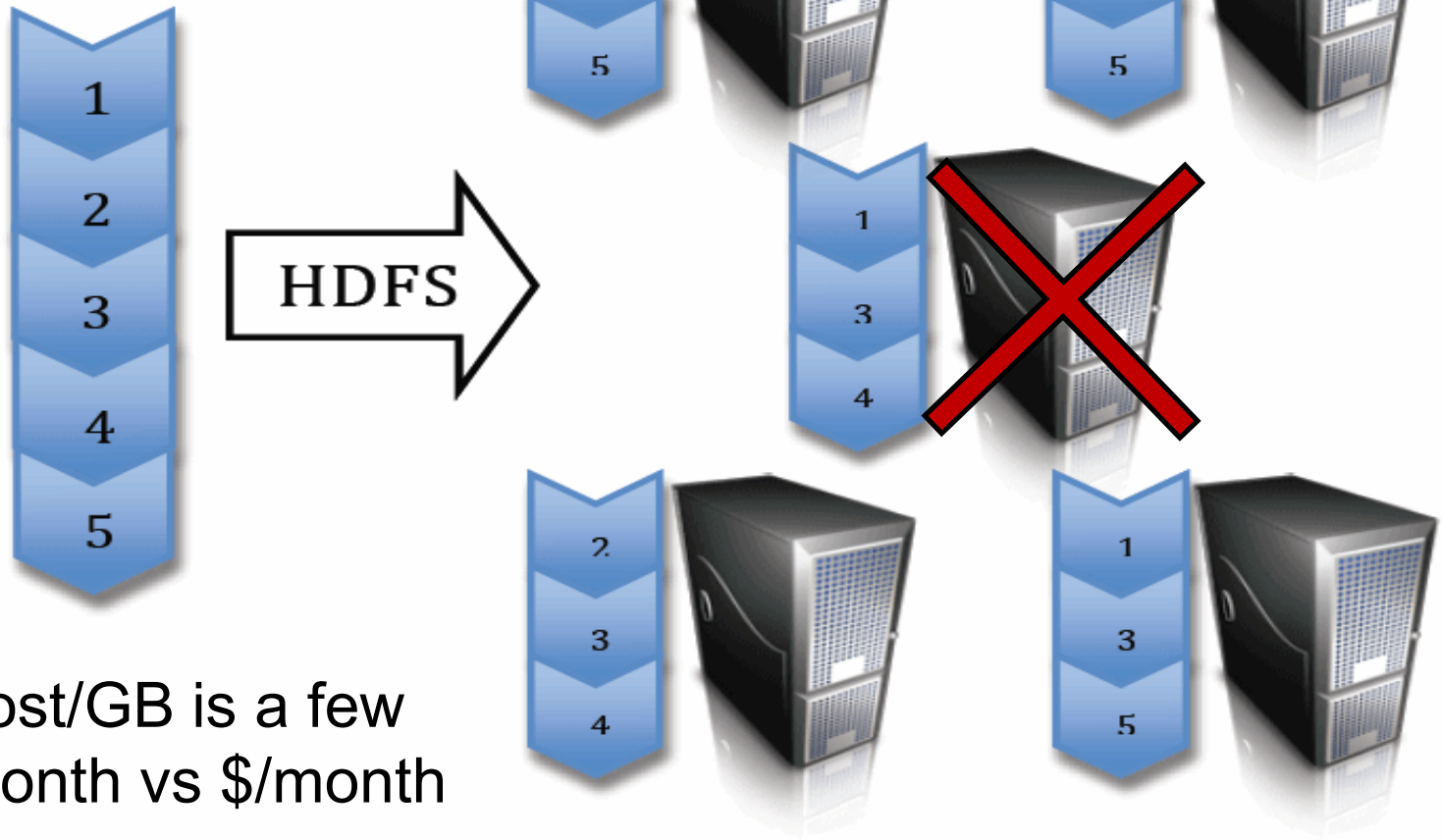


Apache Hadoop HDFS

อ.ดนัยรัฐ ธนบดีธรรมจารี
Line ID: Danairat
FB: Danairat Thanabodithammachari
+668-1559-1446

HDFS: Hadoop Distributed File System

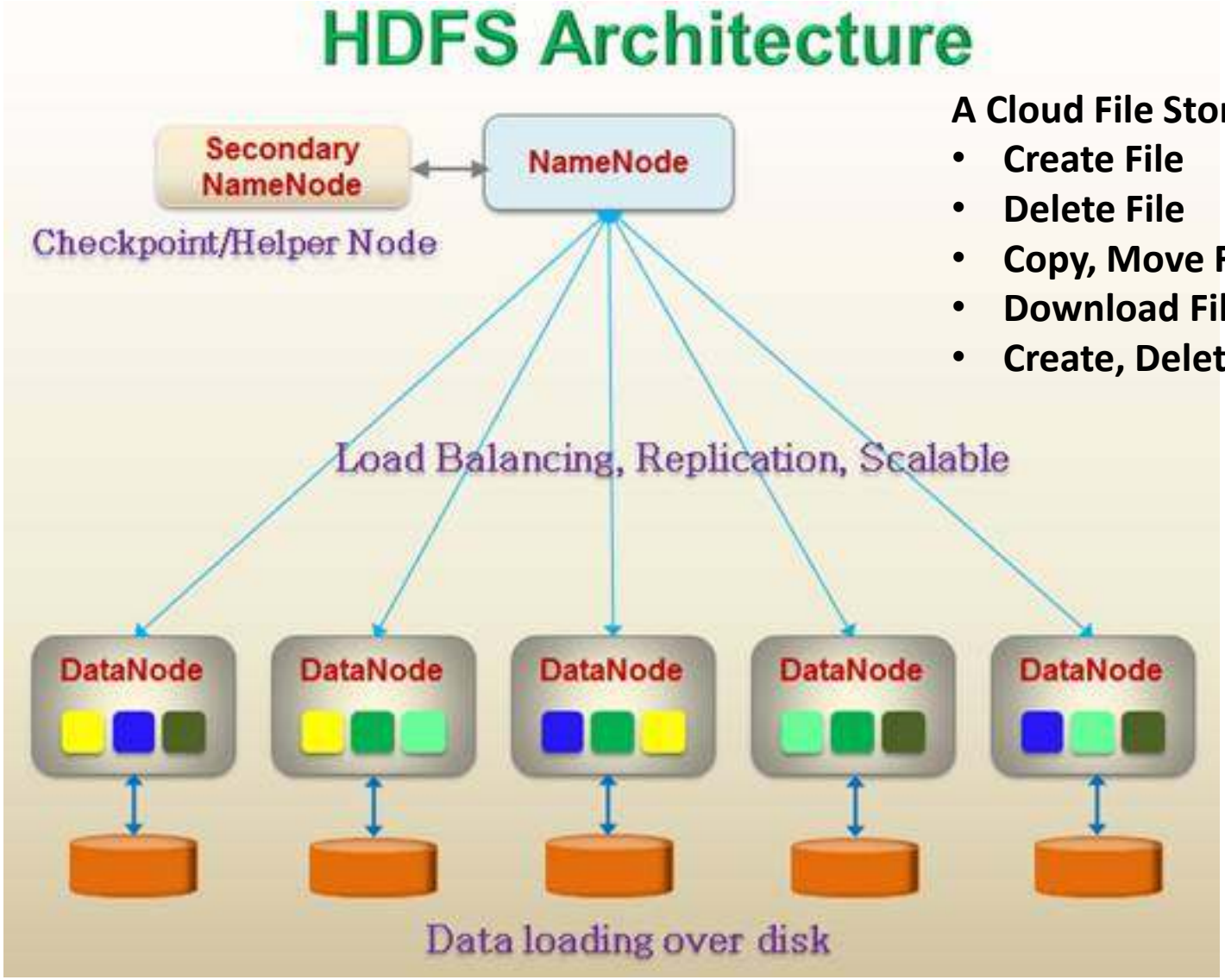
Block Size =
64MB/128MB/256MB
Replication Factor = 3



Cost/GB is a few
¢/month vs $/month

apache.org/hadoop/

HDFS

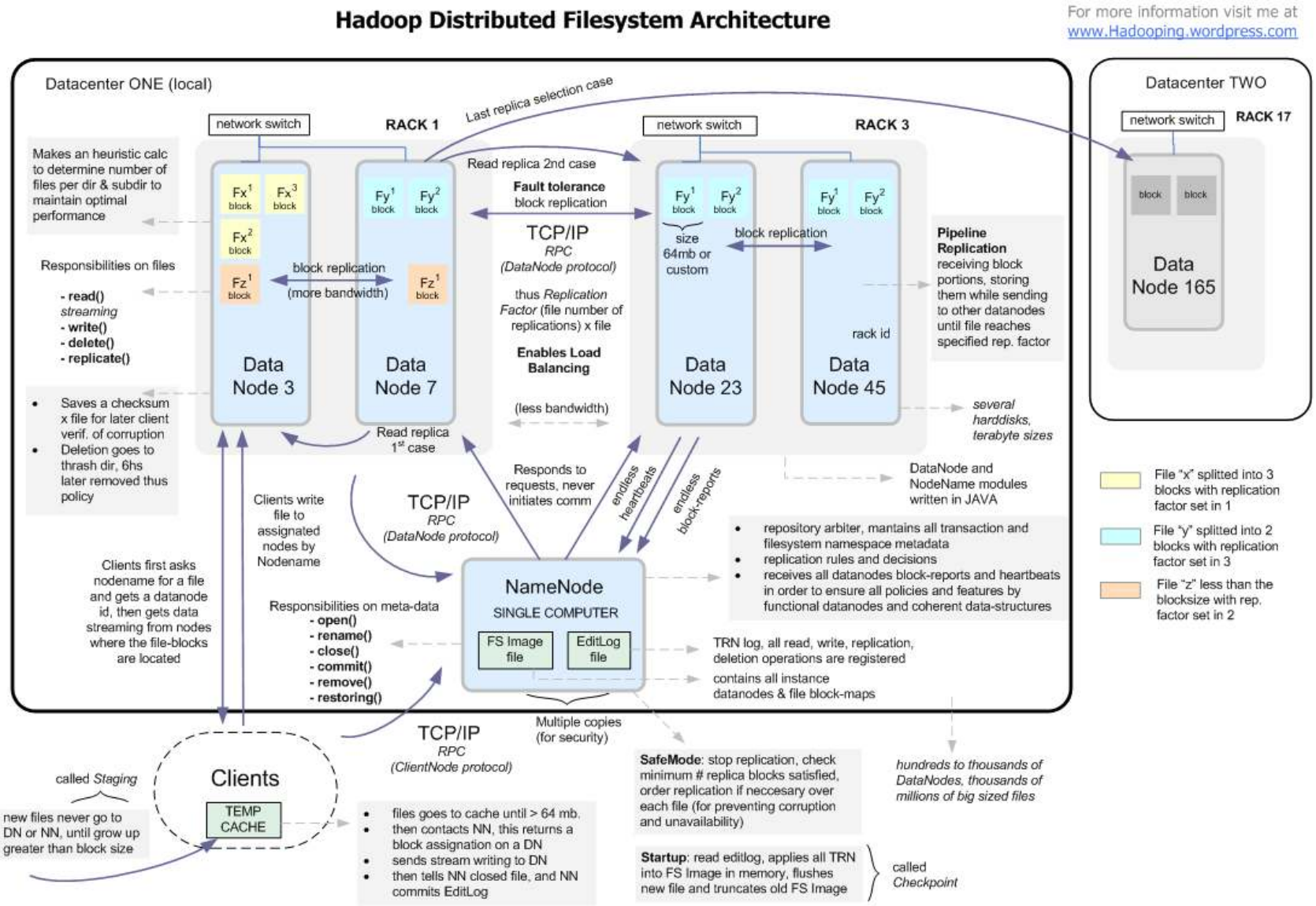


A Cloud File Storage:-

- **Create File**
- **Delete File**
- **Copy, Move File**
- **Download File**
- **Create, Delete Directory**

http://www.developer.com/db/getting-familiarized-with-the-hadoop-distribution-file-system.html

Hadoop HDFS Architecture



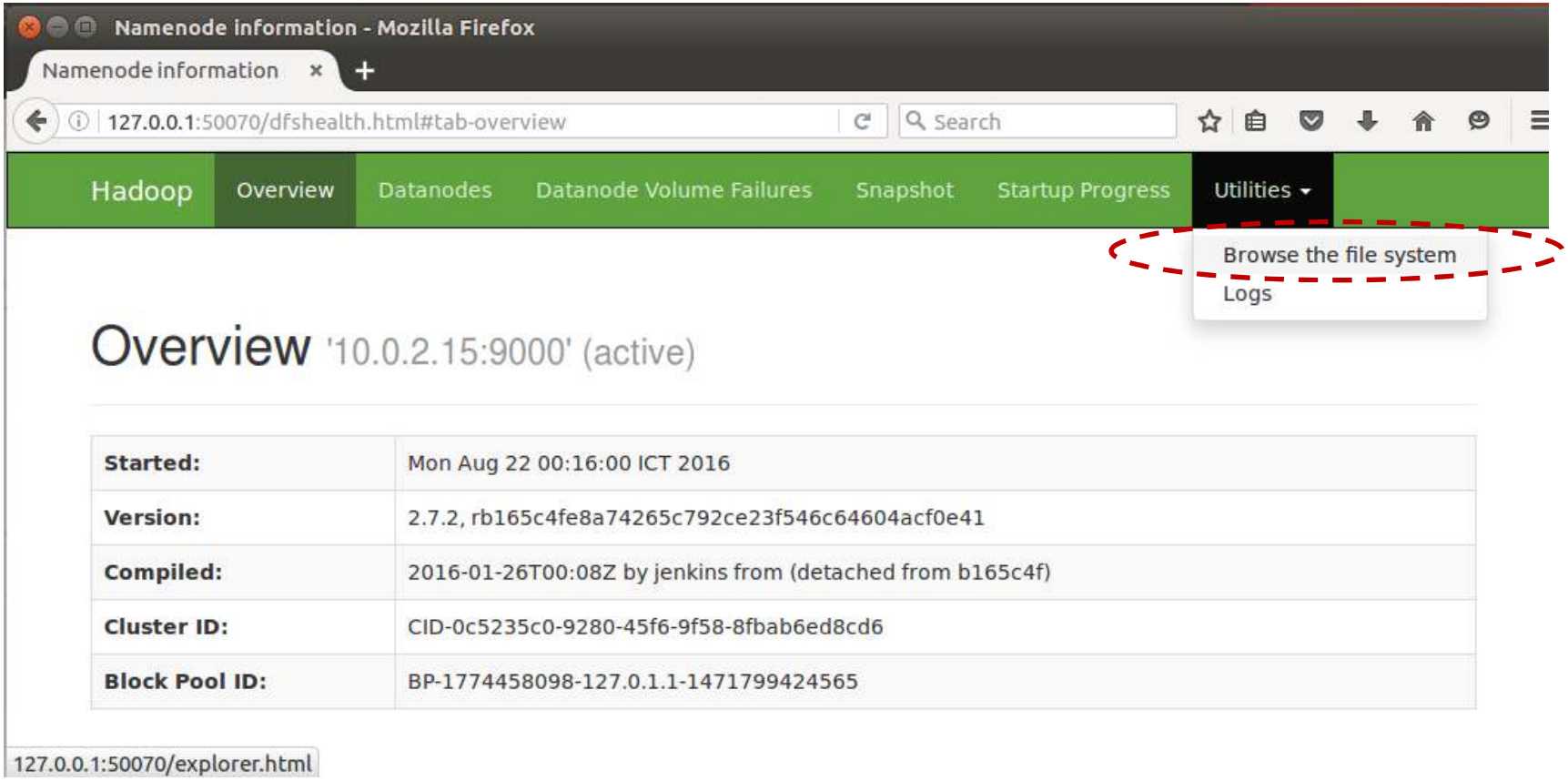
4. Hands on HDFS

```
$ hdfs dfs –mkdir /inputs
$ hdfs dfs –mkdir /outputs
```

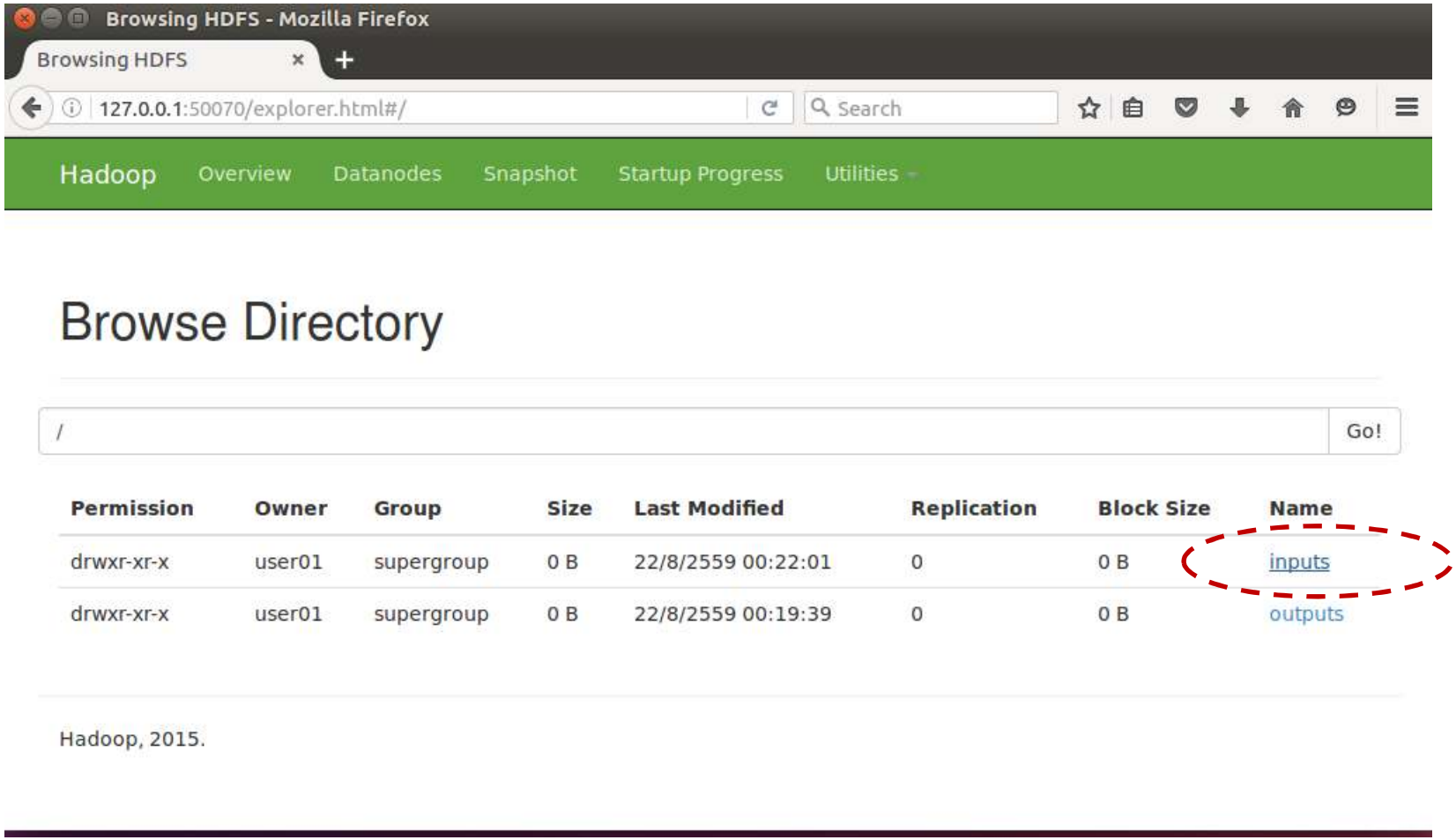
$ hdfs dfs -copyFromLocal ~/sales_data_jan2009.csv /inputs/

$ hdfs dfs -ls /inputs

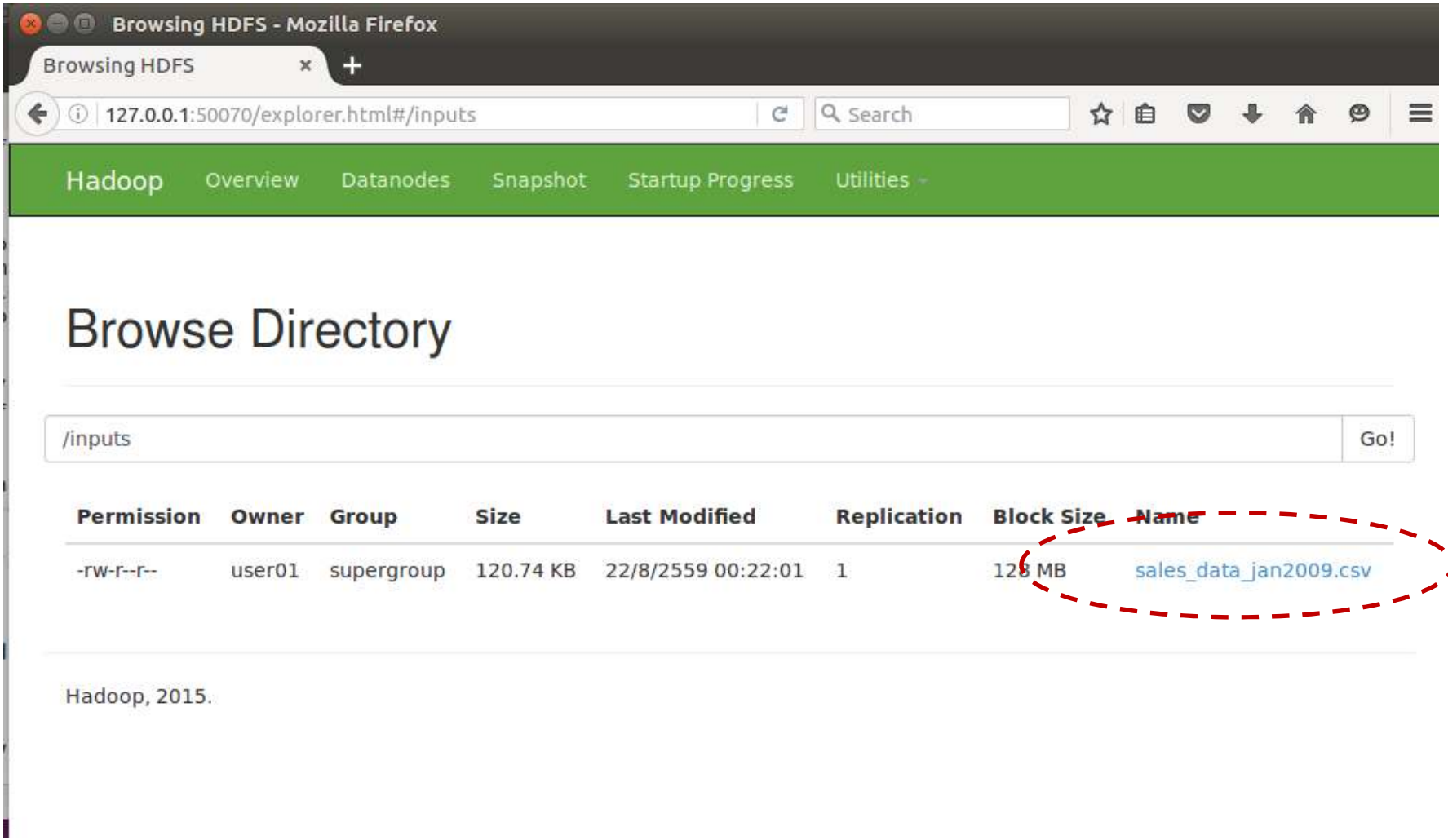
4. Hands on HDFS



4. Hands on HDFS



4. Hands on HDFS



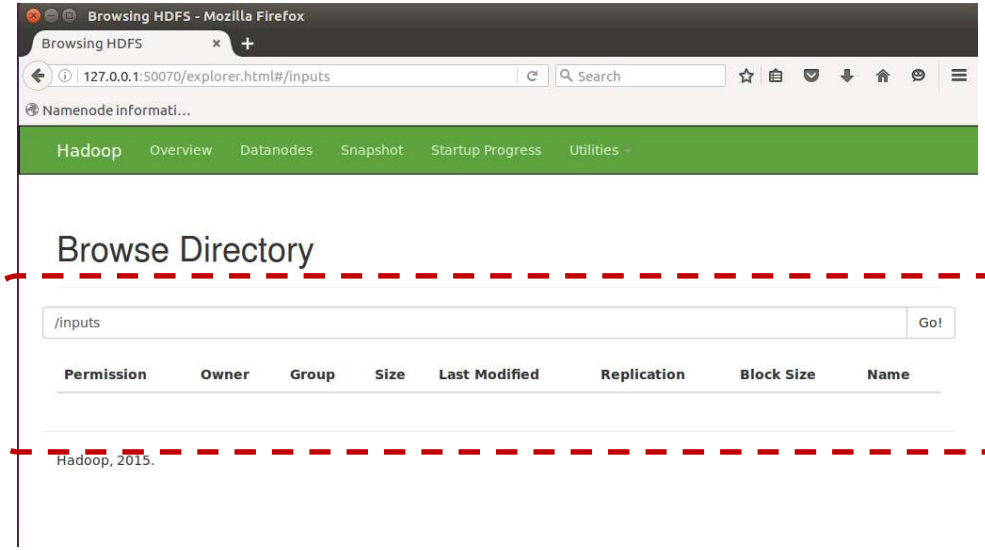
4. Hands on HDFS

$ hdfs dfs -cat /inputs/sales_data_jan2009.csv

4. Hands on HDFS

$ hdfs dfs -rm /inputs/sales_data_jan2009.csv

```
user01@user01-VirtualBox: ~
user01@user01-VirtualBox:~$ hdfs dfs -rm /inputs/sales_data_jan2009.csv
16/08/25 21:53:32 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interval
= 0 minutes, Emptier interval = 0 minutes.
Deleted /inputs/sales_data_jan2009.csv
user01@user01-VirtualBox:~$ hdfs dfs -ls /inputs/sales_data_jan2009.csv
ls: `/inputs/sales_data_jan2009.csv': No such file or directory
user01@user01-VirtualBox:~$
```



4. Hands on HDFS

$ hdfs dfs -copyFromLocal ./sales_data_jan2009.csv /inputs/sales_data_jan2009.csv

$ hdfs dfs -ls /inputs/

```
user01@user01-VirtualBox:/usr/local/hadoop/etc/hadoop$ hdfs dfs -ls /inputs/
Found 1 items
-rw-r--r--   1 user01 supergroup     123637 2016-08-22 00:22 /inputs/sales_data_jan2009.csv
user01@user01-VirtualBox:/usr/local/hadoop/etc/hadoop$
```

4. Hands on HDFS

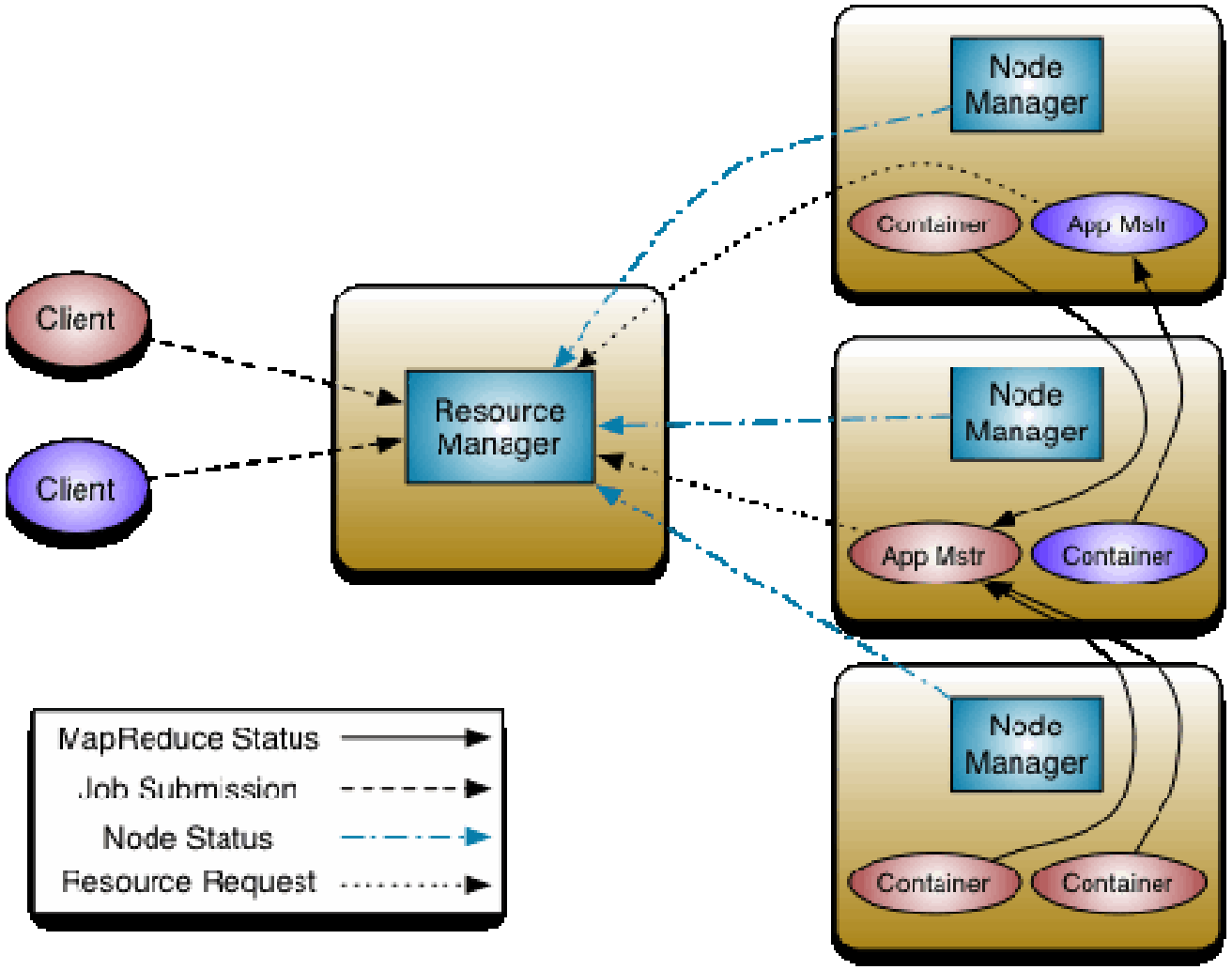
(Optional) Review linux file system

```
user01@user01-VirtualBox:~$ cd /var/hadoop_data/datanode/
user01@user01-VirtualBox:/var/hadoop_data/datanode$ ls
current
user01@user01-VirtualBox:/var/hadoop_data/datanode$ cd current/
user01@user01-VirtualBox:/var/hadoop_data/datanode/current$ ls
BP-1774458098-127.0.1.1-1471799424565  VERSION
user01@user01-VirtualBox:/var/hadoop_data/datanode/current$ cd BP-1774458098-127.0.1.1-1471799
424565/
user01@user01-VirtualBox:/var/hadoop_data/datanode/current/BP-1774458098-127.0.1.1-14717994245
65$ ls
current  scanner.cursor  tmp
user01@user01-VirtualBox:/var/hadoop_data/datanode/current/BP-1774458098-127.0.1.1-14717994245
65$ cd current/
user01@user01-VirtualBox:/var/hadoop_data/datanode/current/BP-1774458098-127.0.1.1-14717994245
65/current$ ls
dfsUsed  finalized  rbw  VERSION
user01@user01-VirtualBox:/var/hadoop_data/datanode/current/BP-1774458098-127.0.1.1-14717994245
65/current$
```

Apache Hadoop YARN and MRv2

อ.ดนัยรัฐ ธนบดีธรรมจารี
Line ID: Danairat
FB: Danairat Thanabodithammachari
+668-1559-1446

YARN: Yet Another Resource Negotiator

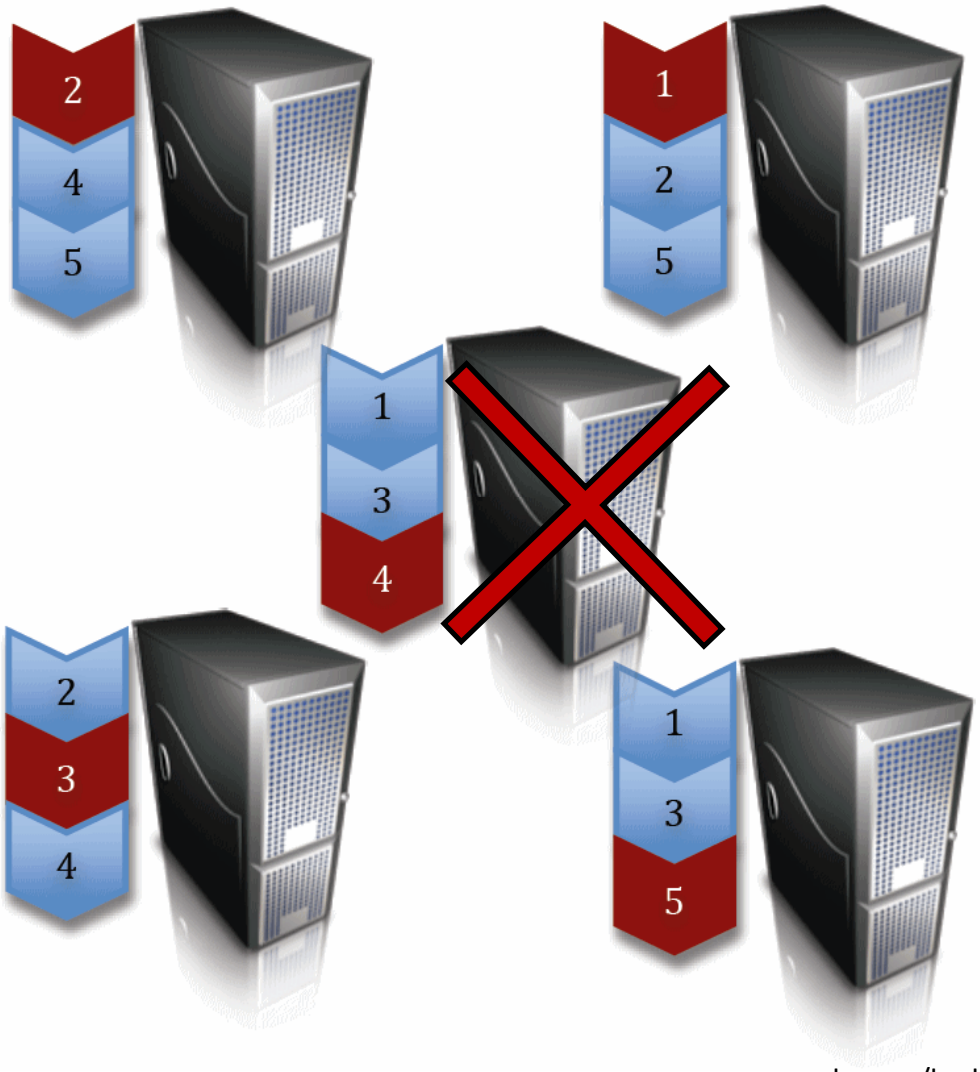


MRV2 maintains API compatibility with previous stable release (hadoop-1.x). This means that all Map-Reduce jobs should still run unchanged on top of MRv2 with just a recompile.

Hadoop.apache.org

MapReduce: Distributed Processing

Hadoop takes advantage of HDFS' data distribution strategy to push work out to many nodes in a cluster. This allows analyses to run in parallel and eliminates the bottlenecks imposed by monolithic storage systems.

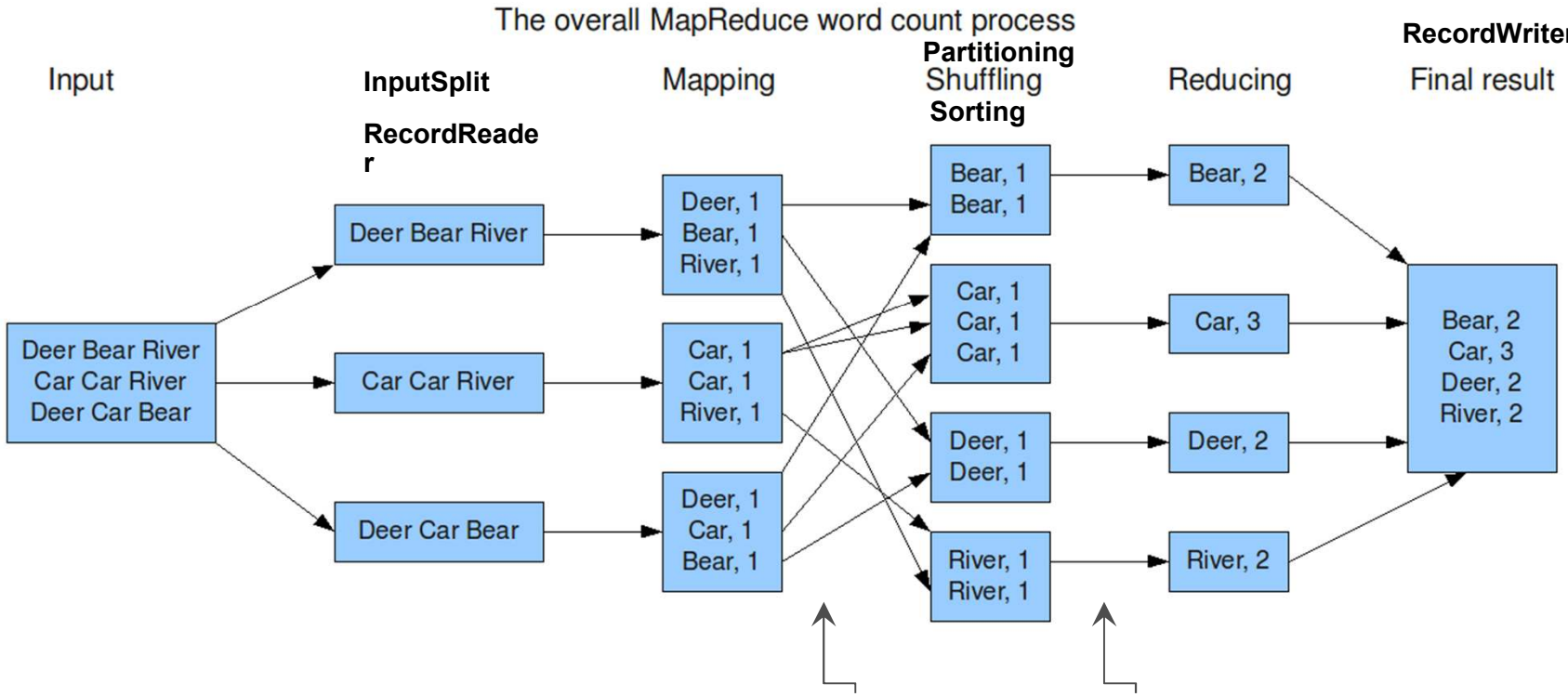


apache.org/hadoop/

MapReduce Framework

```
map: (K1, V1) -> list(K2, V2)
reduce: (K2, list(V2)) -> list(K3, V3)
```

How does the MapReduce work?



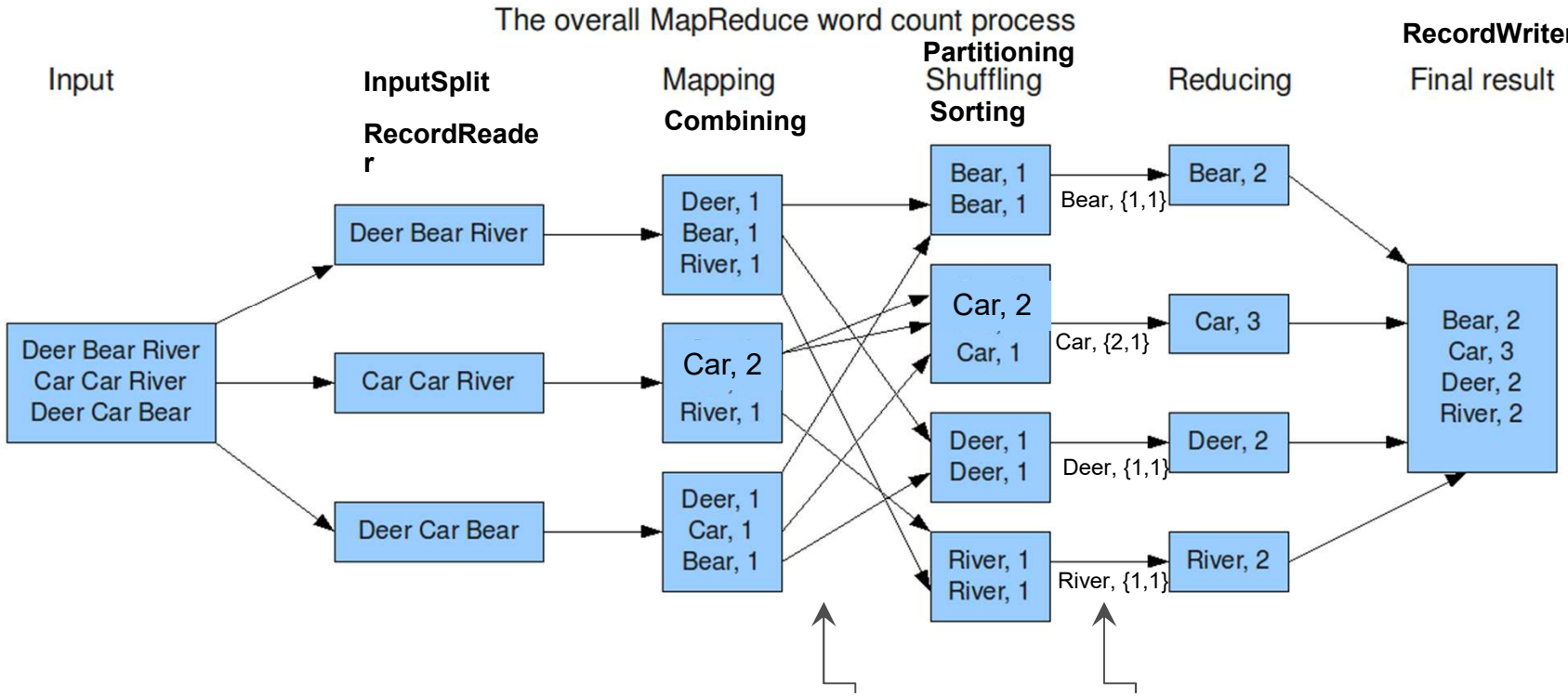
Output in a list of (Key, Value)

in the intermediate file

Output in a list of (Key, List of Values)

in the intermediate file

How does the MapReduce work?



Output in a list of (Key, Value) in the intermediate file

Output in a list of (Key, List of Values) in the intermediate file

MapReduce Processing – The Data flow

1. InputFormat, InputSplits, RecordReader
2. Mapper - your focus is here
3. Partition, Shuffle & Sort
4. Reducer - your focus is here
5. OutputFormat, RecordWriter

InputFormat

InputFormat:	Description:	Key:	Value:
TextInputFormat	Default format; reads lines of text files	The byte offset of the line	The line contents
KeyValueInputFormat	Parses lines into key, val pairs	Everything up to the first tab character	The remainder of the line
SequenceFileInputFormat	A Hadoop-specific high-performance binary format	user-defined	user-defined

InputSplit

An InputSplit describes a unit of work that comprises a single *map task.*

InputSplit presents a byte-oriented view of the input.

You can control this value by setting the mapred.min.split.size parameter in core-site.xml, or by overriding the parameter in the JobConf object used to submit a particular MapReduce job.

RecordReader

RecordReader reads <key, value> pairs from an InputSplit.

Typically the RecordReader converts the byte-oriented view of the input, provided by the InputSplit, and presents a record-oriented to the Mapper

Mapper

Mapper: The Mapper performs the user-defined logic to the input a key, value and emits (key, value) pair(s) which are forwarded to the Reducers.

Partition, Shuffle & Sort

After the first map tasks have completed, the nodes may still be performing several more map tasks each. But they also begin exchanging the intermediate outputs from the map tasks to where they are required by the reducers.

Partitioner controls the partitioning of map-outputs to assign to reduce task . he total number of partitions is the same as the number of reduce tasks for the job

The set of intermediate keys on a single node is automatically sorted by internal Hadoop before they are presented to the Reducer

This process of moving map outputs to the reducers is known as shuffling.

Reducer

This is an instance of user-provided code that performs read each key, iterator of values in the partition assigned. The *OutputCollector* object in Reducer phase has a method named collect() which will collect a (key, value) output.

OutputFormat, Record Writer

OutputFormat governs the writing format in OutputCollector and RecordWriter writes output into HDFS.

TextOutputFormat	Default; writes lines in "key \t value" form
SequenceFileOutputFormat	Writes binary files suitable for reading into subsequent MapReduce jobs
NullOutputFormat	generates no output files

5. Hands on MRv2

```
$ cd
$ nano WordCount.java
```

```
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);
     }
   }
 }

 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);
 }
}
```

5. Hands on MRv2

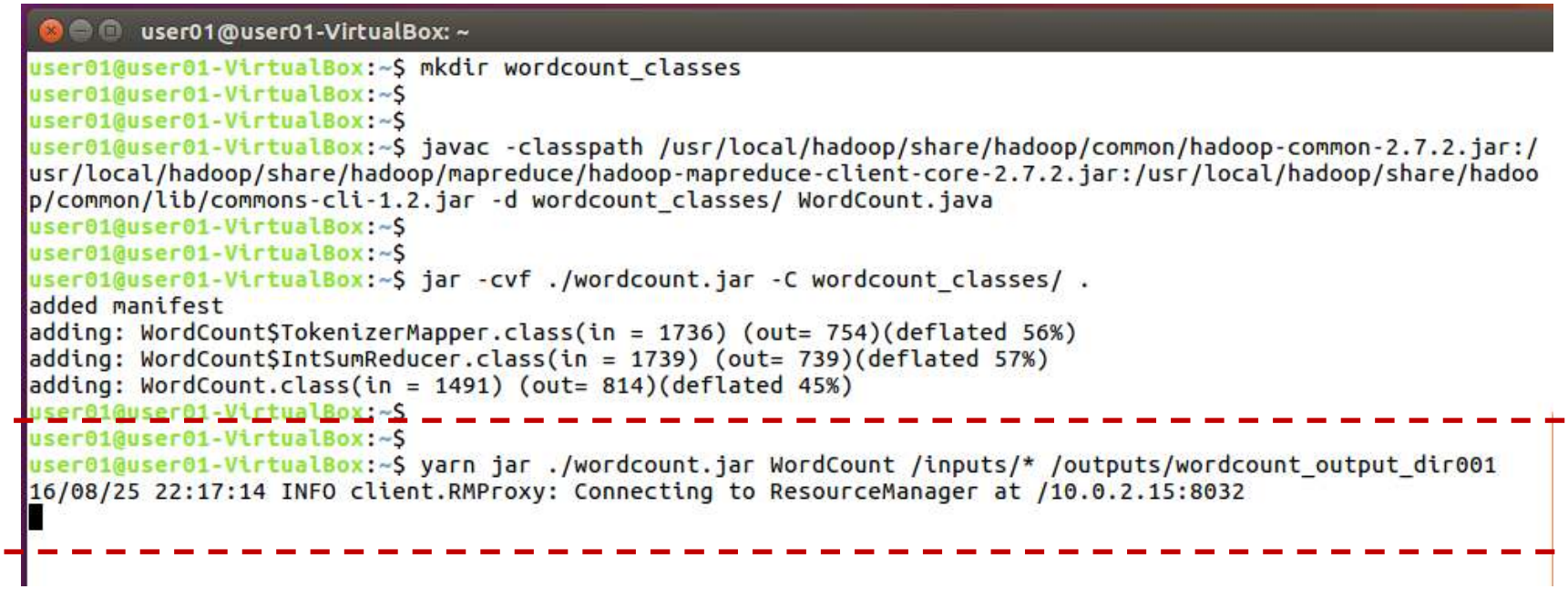
Compile .java to .class

```
$ mkdir wordcount_classes

$ javac -classpath /usr/local/hadoop/share/hadoop/common/hadoop-common-2.7.2.jar:/usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-client-core-2.7.2.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-cli-1.2.jar -d wordcount_classes/ WordCount.java

$ jar -cvf ./wordcount.jar -C wordcount_classes/ .
```

5. Hands on MRv2

Execute YARN

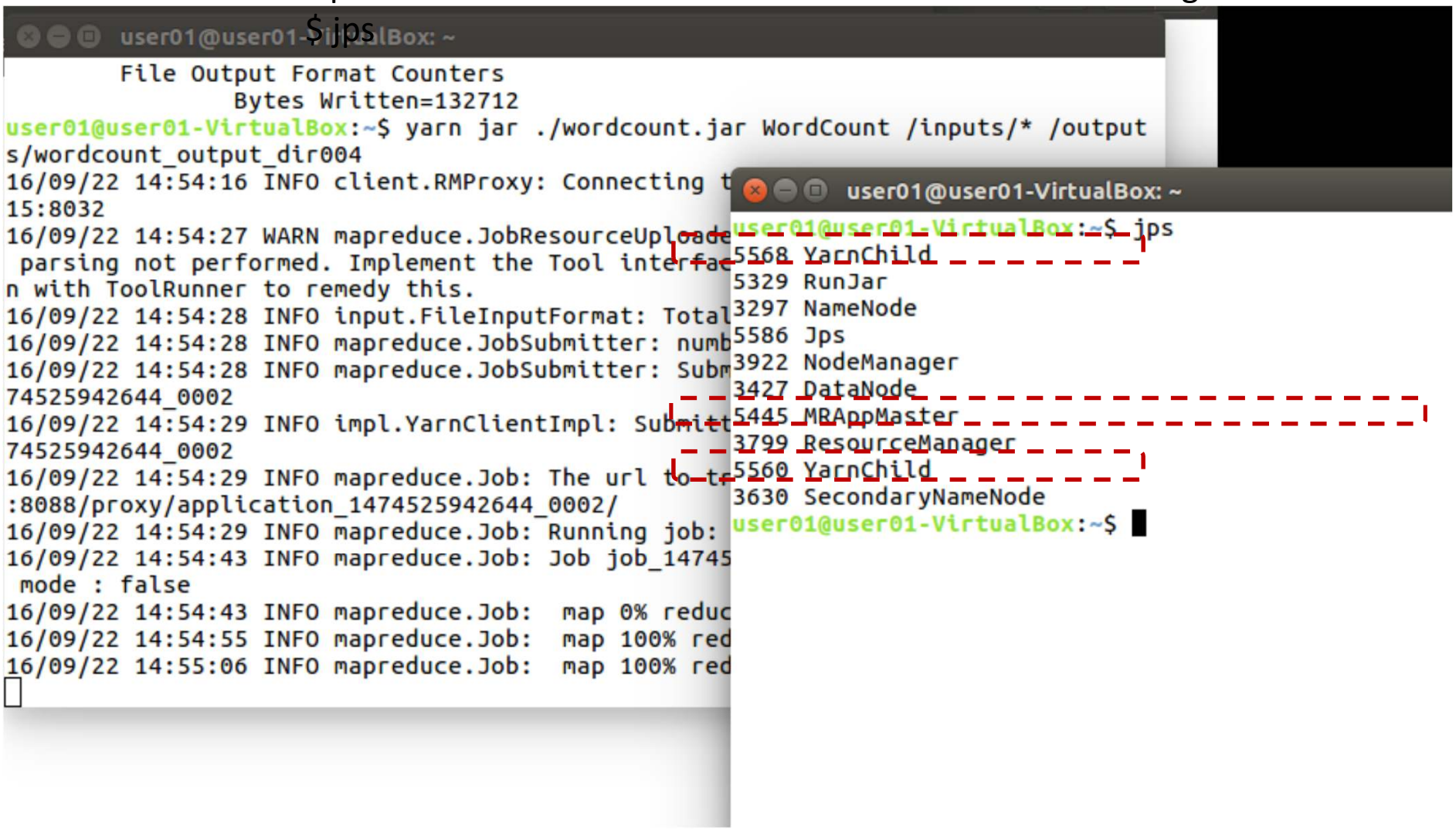
$ yarn jar ./wordcount.jar WordCount /inputs/* /outputs/wordcount_output_dir001



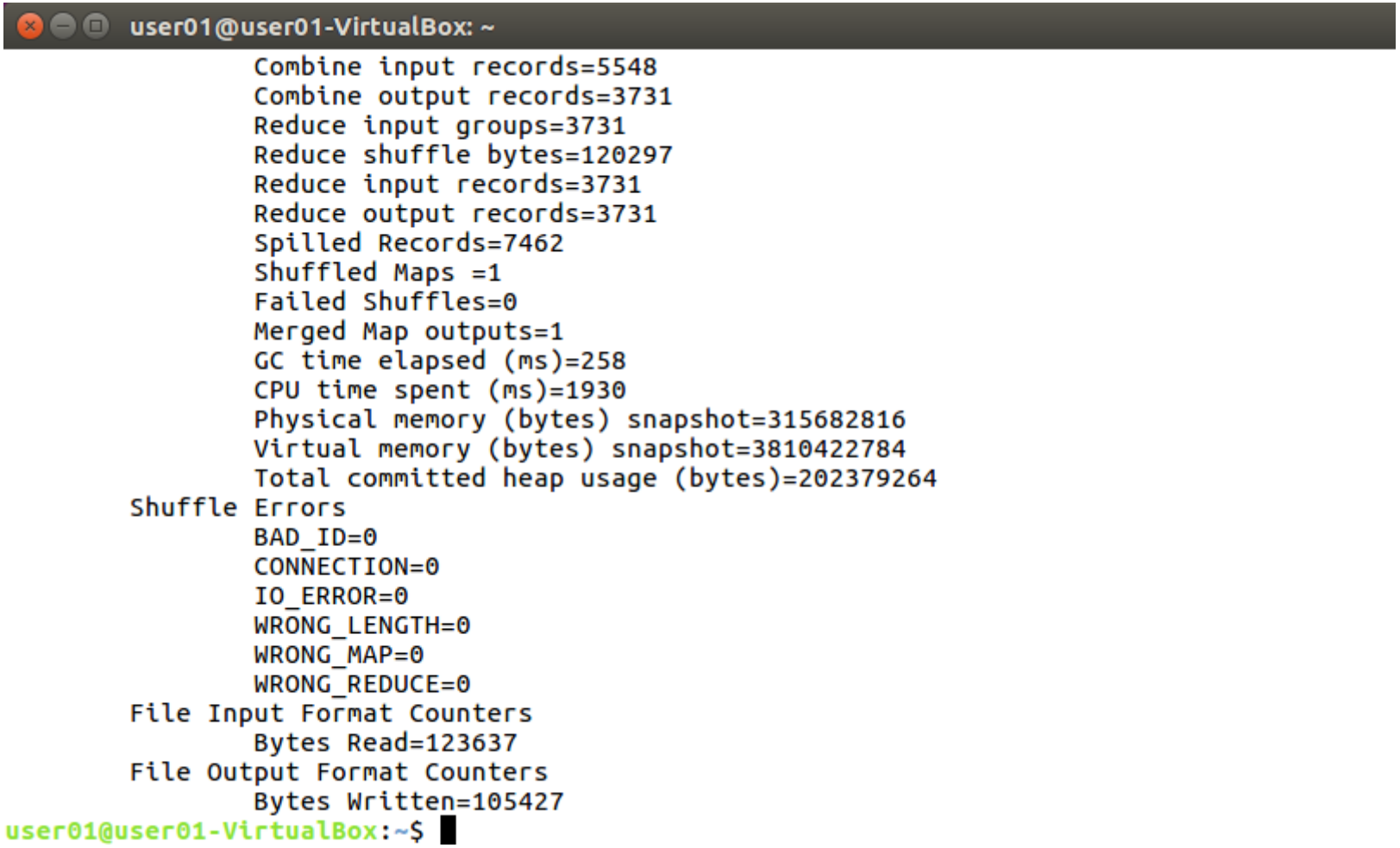
5. Hands on MRv2

Open another terminal to check Yarn Processes are running

$ jps



5. Hands on MRv2



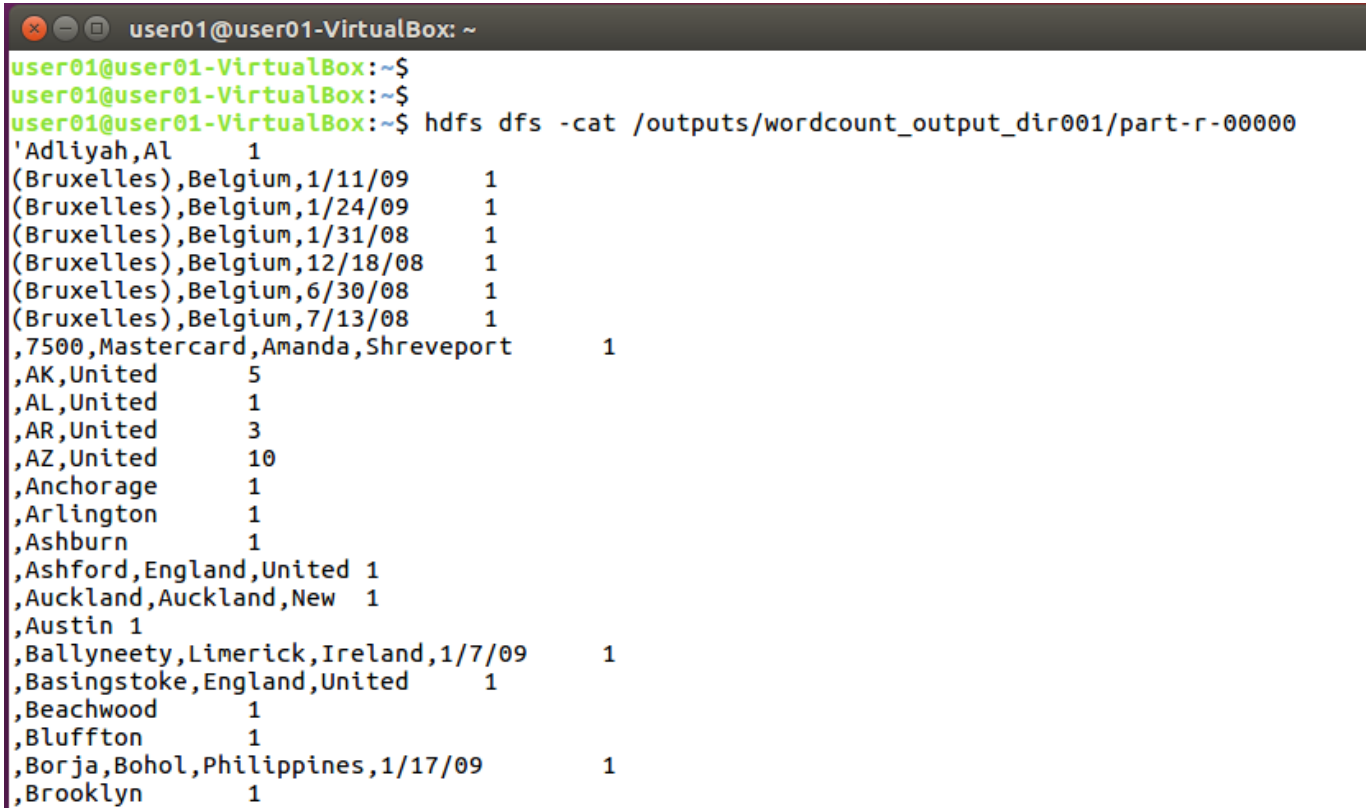
5. Hands on MRv2

Review the results from Map Reduce Command

$ hdfs dfs -ls /outputs

$ hdfs dfs -ls /outputs/wordcount_output_dir001

$ hdfs dfs -cat /outputs/wordcount_output_dir001/part-r-00000

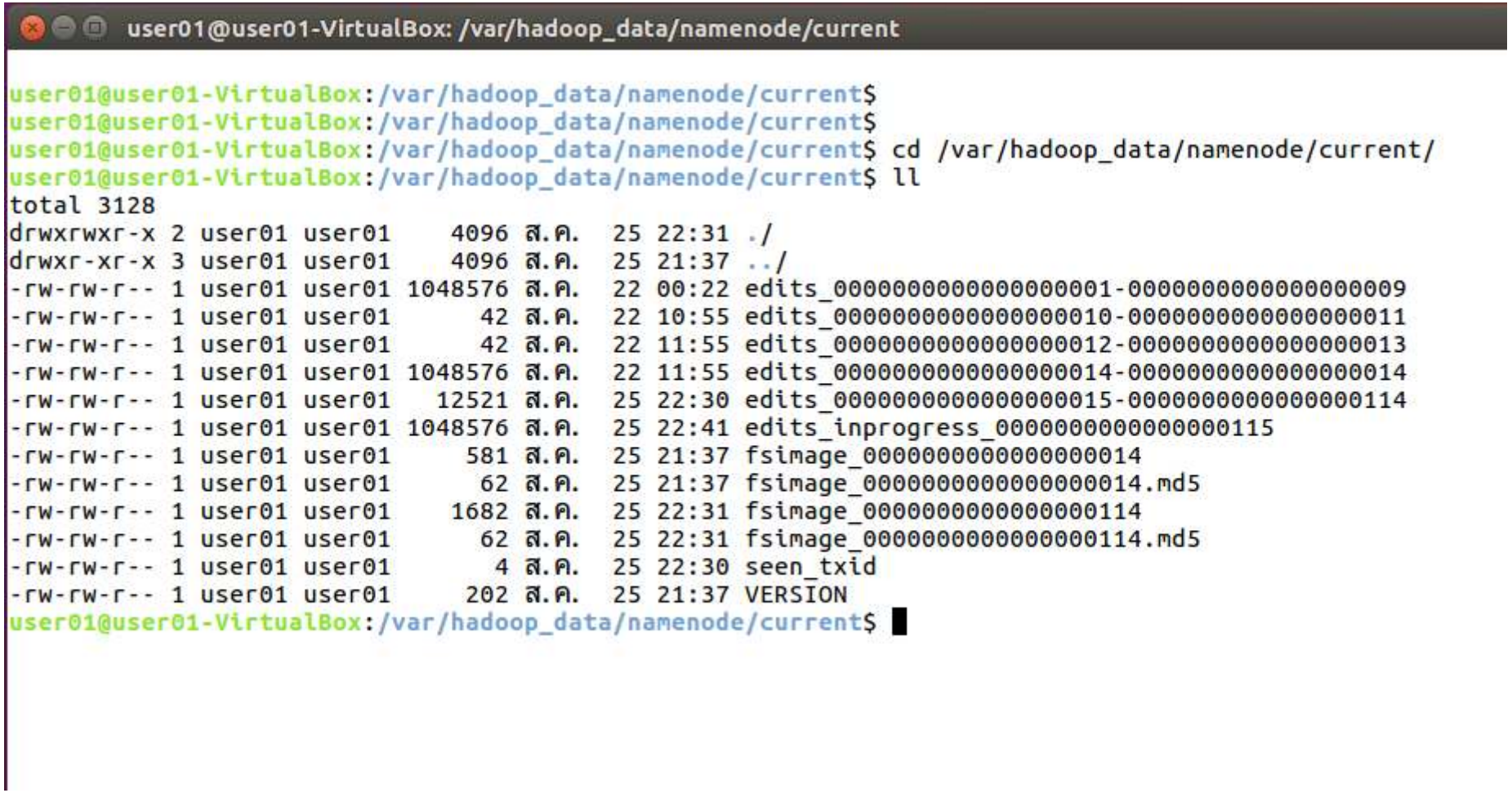


5. Hands on MRv2

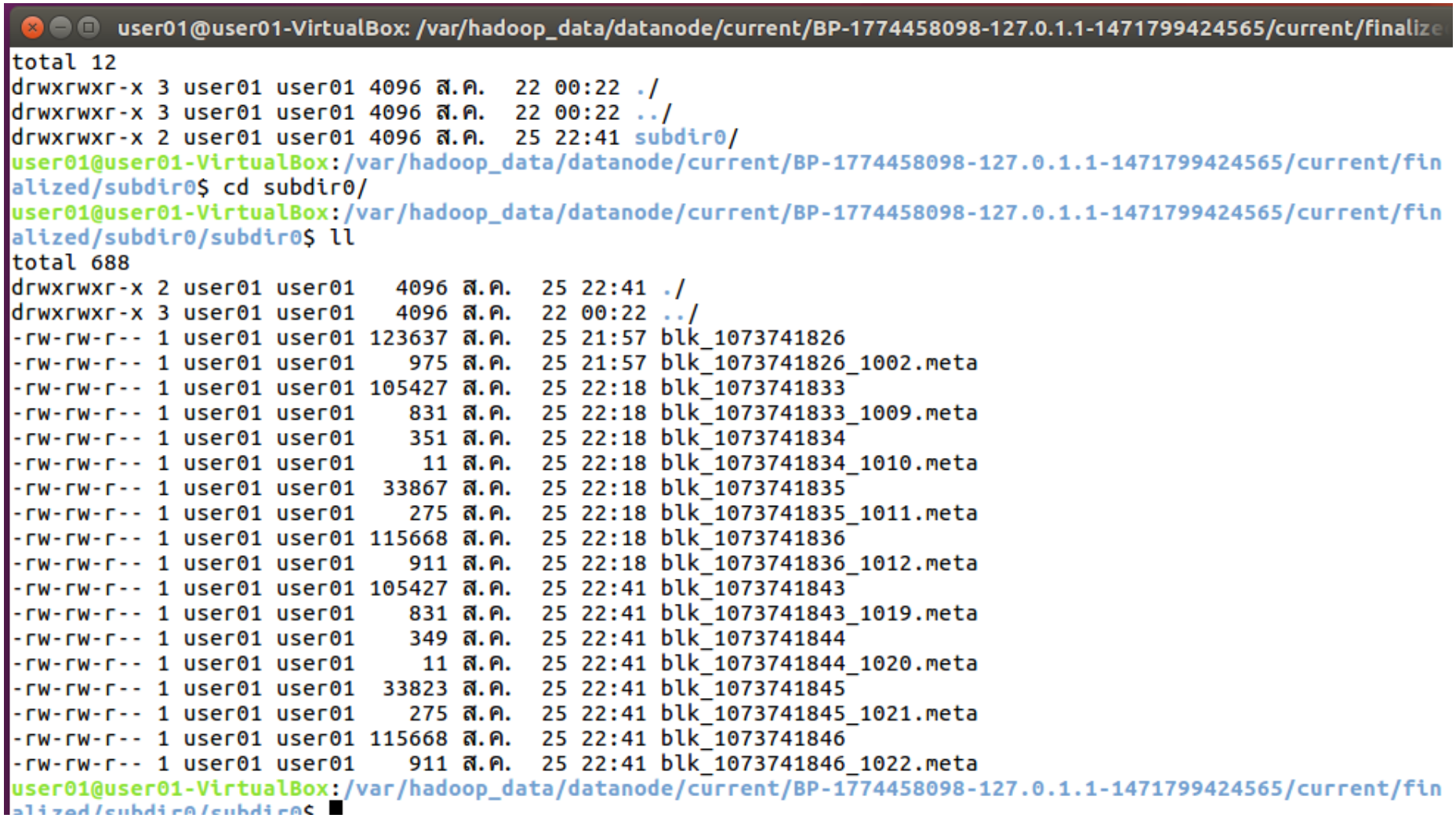
Review Linux File System

```
$ cd /var/hadoop_data/namenode/current/
$ ls -l

$ cd /var/hadoop_data/datanode/current/
$ ls -l
```

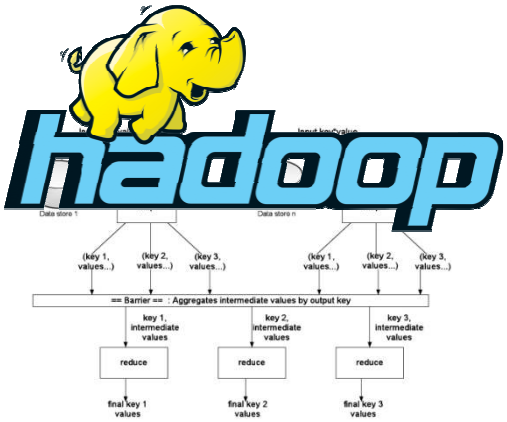
5. Hands on MRv2



5. Hands on MRv2



Very Large Server Log files



MRv2 in Data Lake

Analyzing Large Server Log Files with Hadoop MapReduce

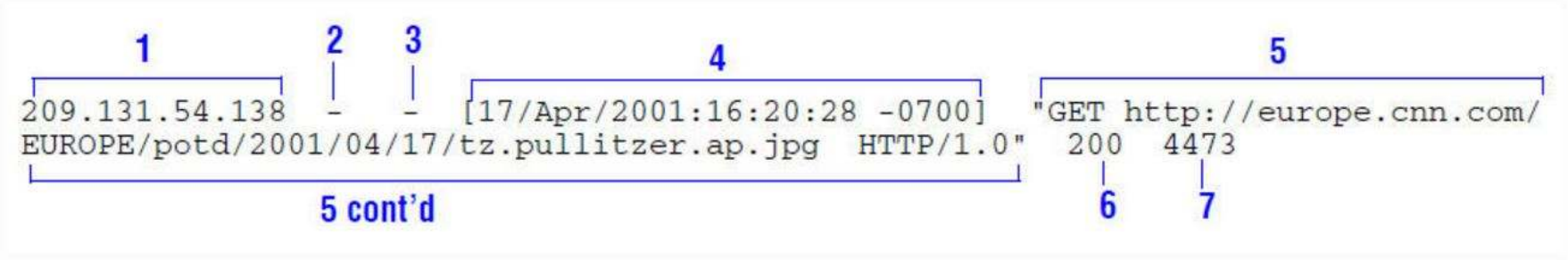
อ.ดนัยรัฐ ธนบดีธรรมจารี

Line ID: Danairat

FB: Danairat Thanabodithammachari

+668-1559-1446

Understand Server Log



No.	Field	Description
1	chi	The IP address of the client's host machine.
2	–	This hyphen (-) is always present in Netscape log entries.
3	caun	The authenticated client username. A hyphen (-) means no authentication was required.
4	cqtn	The date and time of the client request, enclosed in brackets.
5	cqtx	The request line, enclosed in quotes.
6	pssc	The proxy response status code (HTTP reply code).
7	pscl	The length of the Traffic Server response to the client in bytes.

Put HTTP log to HDFS

hdfs dfs -put server.log /user/inputs

Input server log

Count by Host Name Results

dial22.lloyd.com	61716
ix-orl2-01.ix.netcom.com	3985
net-1-141.eden.com	34029
ppp-mia-30.shadow.net	14089
unicomp6.unicomp.net	46285
waters-gw.starway.net.au	6723

Write MR program

```
$ vi LogWritable.java
$ vi LogProcessorMap.java
$ vi LogProcessorReduce.java
$ vi LogProcessor.java
```

Write MR program - LogWritable.java

```
import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.WritableComparable;

public class LogWritable implements WritableComparable<LogWritable> {

        private Text userIP, timestamp, request;
        private IntWritable responseSize, status;

        public LogWritable() {
                this.userIP = new Text();
                this.timestamp =  new Text();
                this.request = new Text();
                this.responseSize = new IntWritable();
                this.status = new IntWritable();
        }

        public void set (String userIP, String timestamp, String request, int bytes, int status)
        {
                this.userIP.set(userIP);
                this.timestamp.set(timestamp);
                this.request.set(request);
                this.responseSize.set(bytes);
                this.status.set(status);
        }

        @Override
        public void readFields(DataInput in) throws IOException {
                userIP.readFields(in);
                timestamp.readFields(in);
                request.readFields(in);
                responseSize.readFields(in);
                status.readFields(in);
        }
```

Write MR program - LogWritable.java (Cont.)

```
@Override
        public void write(DataOutput out) throws IOException
{
                userIP.write(out);
                timestamp.write(out);
                request.write(out);
                responseSize.write(out);
                status.write(out);
        }

        @Override
        public int compareTo(LogWritable o) {
                if (userIP.compareTo(o.userIP) == 0) {
                        return
timestamp.compareTo(o.timestamp);
                } else
                        return
userIP.compareTo(o.userIP);
        }

        public int hashCode()
        {
                return userIP.hashCode();
        }
```

```
        public Text getUserIP() {
                return userIP;
        }

        public Text getTimestamp() {
                return timestamp;
        }

        public Text getRequest() {
                return request;
        }

        public IntWritable getResponseSize() {
                return responseSize;
        }

        public IntWritable getStatus() {
                return status;
        }
}
```

Write MR program - LogProcessorMap.java

```
package logprocess;
import java.io.IOException;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;


public class LogProcessorMap extends Mapper<Object, Text, Text, LogWritable > {
		LogWritable outValue = new LogWritable();
		Text outKey = new Text();

		public void map(Object key, Text value, Context context)
				throws IOException, InterruptedException {
			String logEntryPattern = "^(\\S+) (\\S+) (\\S+) \\[([\\w:/]+\\s[+\\-]\\d{4})\\] \"(.+?)\" (\\d{3}) (\\d+)";

			Pattern p = Pattern.compile(logEntryPattern);
			Matcher matcher = p.matcher(value.toString());
			if (!matcher.matches()) {
				System.err.println("Bad Record : "+value);
				return;
			}

			String userIP = matcher.group(1);
			String timestamp = matcher.group(4);
			String request = matcher.group(5);
			int status = Integer.parseInt(matcher.group(6));
			int bytes = Integer.parseInt(matcher.group(7));

			outKey.set(userIP);
			outValue.set(userIP, timestamp, request,
					bytes,status);
			context.write(outKey,outValue);
		}

}
```

Write MR program - LogProcessorReduce.java (cont.)

```
package logprocess;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class LogProcessorReduce extends
                              Reducer<Text,LogWritable,Text,IntWritable> {
  private IntWritable result = new IntWritable();

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

Write MR program - LogProcessor.java

```
package logprocess;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
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.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;


public class LogProcessor extends Configured implements Tool {

        public static void main(String[] args) throws Exception {
                int res = ToolRunner.run(new Configuration(), new LogProcessor(), args);
                System.exit(res);
        }

        @Override
        public int run(String[] args) throws Exception {
                if (args.length < 3) {
                        System.err.println("Usage:  <input_path> <output_path> <num_reduce_tasks>");
                        System.exit(-1);
                }
```

Write MR program - LogProcessor.java (cont.)

```
        /* input parameters */
        String inputPath = args[0];
        String outputPath = args[1];
        int numReduce = Integer.parseInt(args[2]);

        Job job = Job.getInstance(getConf(), "log-analysis");

        job.setJarByClass(LogProcessor.class);
        job.setMapperClass(LogProcessorMap.class);
        job.setReducerClass(LogProcessorReduce.class);
        job.setNumReduceTasks(numReduce);

        System.out.println("set output format");
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(LogWritable.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        System.out.println("set input path");

        FileInputFormat.setInputPaths(job, new Path(inputPath));
        FileOutputFormat.setOutputPath(job, new Path(outputPath));

        int exitStatus = job.waitForCompletion(true) ? 0 : 1;

        return exitStatus;
    }
}
```

Compile, Package, Run MapReduce

mkdir mr_log_classes

javac -cp /usr/lib/hadoop/*:/usr/lib/hadoop-mapreduce/* -d mr_log_classes ./*.java -Xlint

jar -cvf ./logprocessor.jar -C mr_log_classes/ .

yarn jar logprocessor.jar logprocess.LogProcessor /user/inputs/* /user/cloudera/logprocessor001 1

hdfs dfs -cat /user/cloudera/logprocessor001/part-r-00000

```
-bash-4.1$ hdfs dfs -cat /user/cloudera/logprocessor011/part-r-00000
199.120.110.21  9977
199.72.81.55    6245
burger.letters.com      0
dial22.lloyd.com        61716
ix-orl2-01.ix.netcom.com        3985
net-1-141.eden.com      34029
ppp-mia-30.shadow.net   14089
unicomp6.unicomp.net    46285
waters-gw.starway.net.au        6723
-bash-4.1$
```

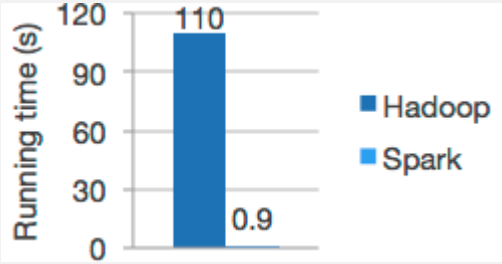
Apache Spark

อ.ดนัยรัฐ ธนบดีธรรมจารี
Line ID: Danairat
FB: Danairat Thanabodithammachari
+668-1559-1446

Apache Spark History

Apache® Spark™ is a powerful open source processing engine built around speed, ease of use, and sophisticated analytics. It was originally developed at UC Berkeley in 2009. Since its release,

Apache Spark™ is a fast and general engine for large-scale data processing.



Internet powerhouses such as Yahoo, Baidu, and Tencent, have eagerly deployed Spark at massive scale of data on clusters of over 8,000 nodes

It has quickly become the largest open source community in big data, with over 1000 contributors from 250+ organizations.

http://spark.apache.org/docs/

Who is Databricks?

Formed by the creators of Apache Spark and Shark, Databricks is working to greatly expand these open source projects and transform big data analysis in the process. We're deeply committed to keeping all work on these systems open source.

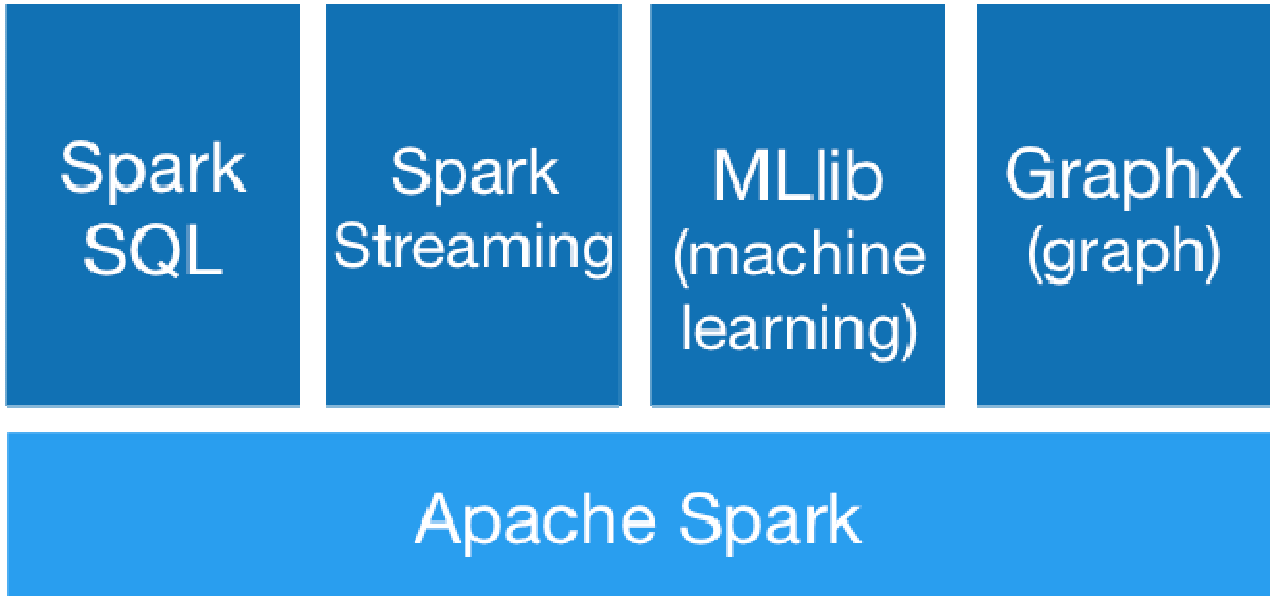
Databricks provided a hosted service to run Spark, Databricks Cloud, and partner to support Apache Spark with other Hadoop and big data companies.

Data Science made easy, from ingest to production. Powered by Apache® Spark.™

https://databricks.com/

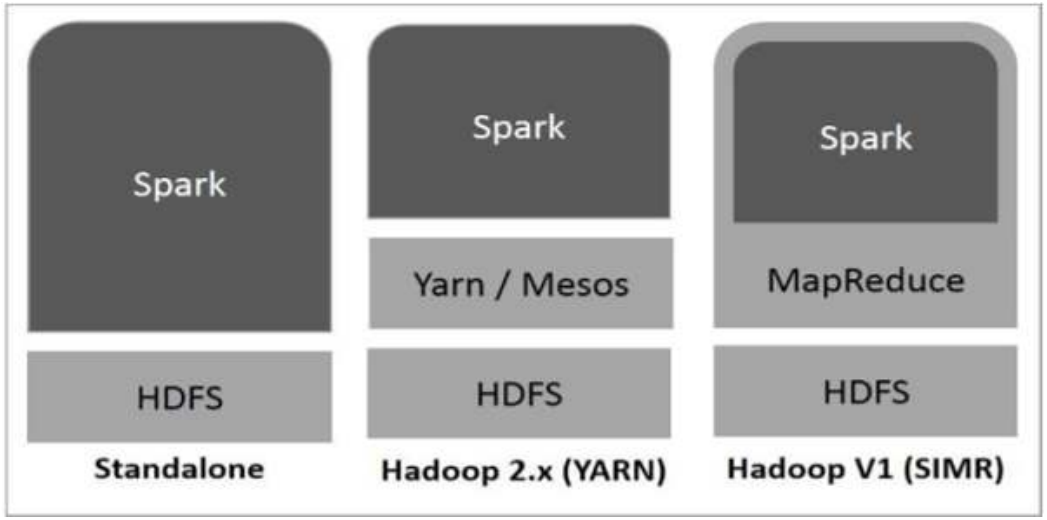
What is Spark?

Apache Spark is a fast and general-purpose cluster computing system. It provides high-level APIs in Java, Scala, Python and R, and an optimized engine that supports general execution graphs. It also supports a rich set of higher-level tools including Spark SQL for SQL and structured data processing, MLlib for machine learning, GraphX for graph processing, and Spark Streaming.



http://spark.apache.org/docs/

Spark and Hadoop Deployment



Standalone − Spark Standalone deployment means Spark occupies the place on top of HDFS(Hadoop Distributed File System) and space is allocated for HDFS, explicitly. Here, Spark and MapReduce will run side by side to cover all spark jobs on cluster.
Hadoop Yarn − Hadoop Yarn deployment means, simply, spark runs on Yarn without any pre-installation or root access required. It helps to integrate Spark into Hadoop ecosystem or Hadoop stack. It allows other components to run on top of stack.
Spark in MapReduce (SIMR) − Spark in MapReduce is used to launch spark job in addition to standalone deployment. With SIMR, user can start Spark and uses its shell without any administrative access.

https://www.tutorialspoint.com/apache_spark/apache_spark_introduction.htm

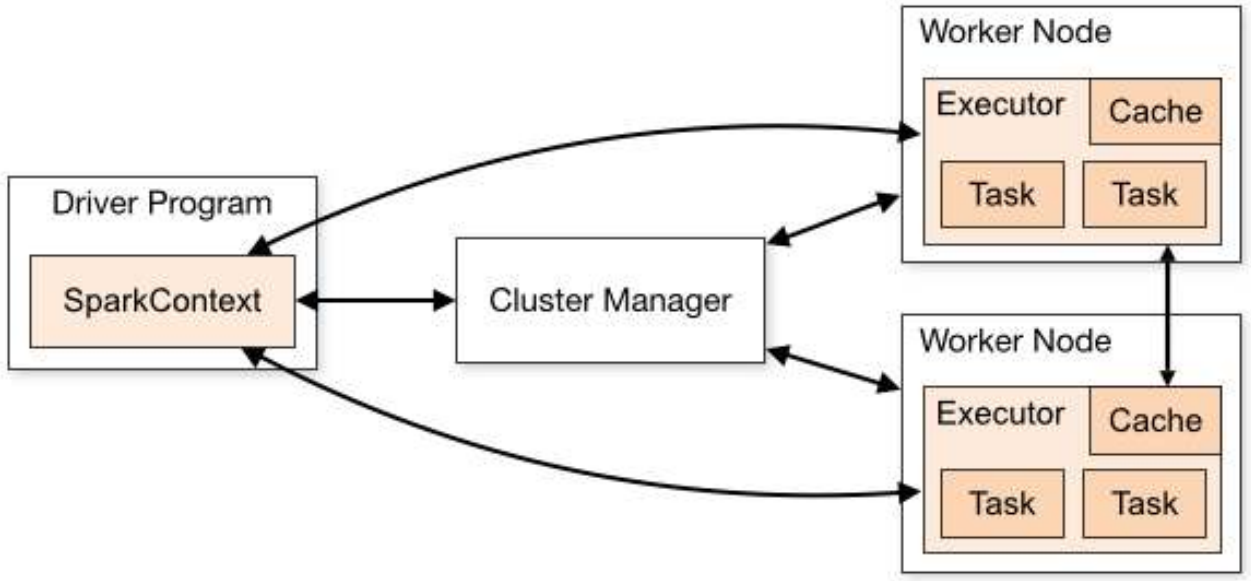
Launching Spark on a Cluster

The Spark cluster mode overview explains the key concepts in running on a cluster. Spark can run both by itself, or over several existing cluster managers. It currently provides several options for deployment:

- Standalone Deploy Mode: simplest way to deploy Spark on a private cluster
- Apache Mesos
- Hadoop YARN

http://spark.apache.org/docs/

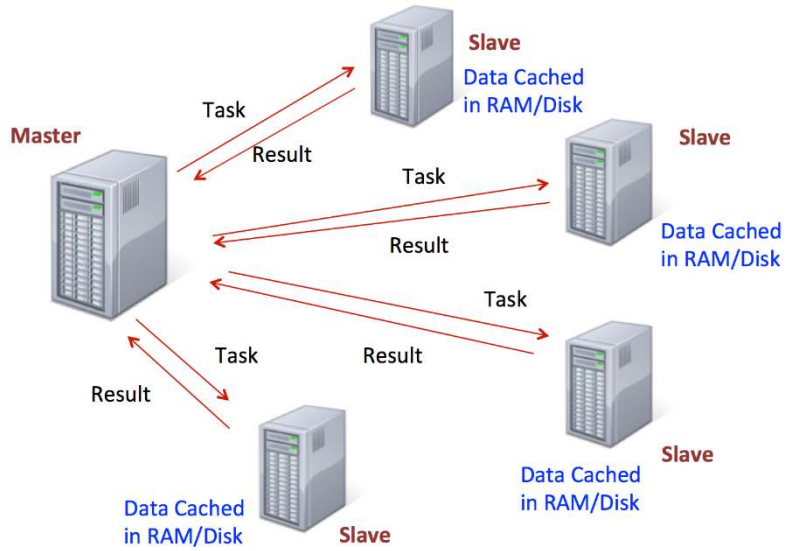
Spark Cluster Mode Overview



Specifically, to run on a cluster, the SparkContext can connect to several types of *cluster managers* (either Spark's own standalone cluster manager, Mesos or YARN), which allocate resources across applications. Once connected, Spark acquires *executors* on nodes in the cluster, which are processes that run computations and store data for your application. Next, it sends your application code (defined by JAR or Python files passed to SparkContext) to the executors. Finally, SparkContext sends *tasks* to the executors to run.

Spark 2.0 Programming

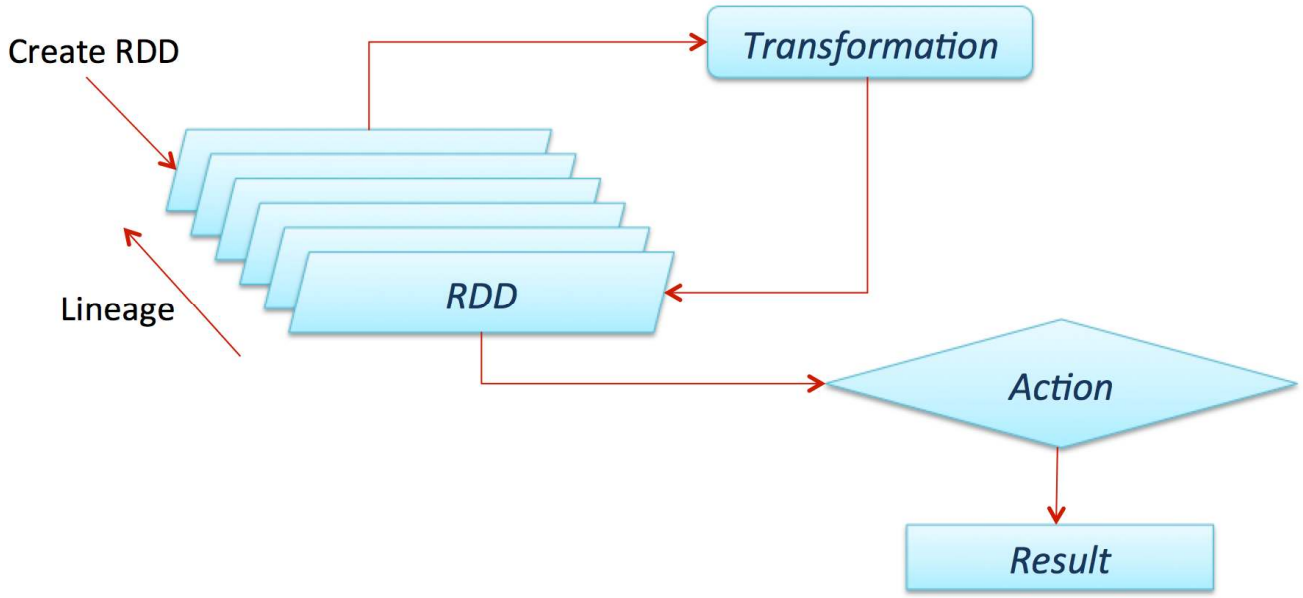
How does Spark execute a job



A second abstraction in Spark is shared variables that can be used in parallel operations. By default, when Spark runs a function in parallel as a set of tasks on different nodes, it ships a copy of each variable used in the function to each task. Sometimes, a variable needs to be shared across tasks, or between tasks and the driver program. Spark supports two types of shared variables: broadcast variables, which can be used to cache a value in memory on all nodes, and accumulators, which are variables that are only "added" to, such as counters and sums.

Spark 2.0 Programming

At a high level, every Spark application consists of a driver program that runs the user's main function and executes various parallel operations on a cluster. The main abstraction Spark provides is a resilient distributed dataset (RDD), which is a collection of elements partitioned across the nodes of the cluster that can be operated on in parallel. RDDs are created by starting with a file in the Hadoop file system (or others).. Finally, RDDs automatically recover from node failures.



Transformations

The following table lists some of the common transformations supported by Spark. Refer to the RDD API doc (Scala, Java, Python, R) and pair RDD functions doc (Scala, Java) for details.

Transformation	Meaning
map(func)	Return a new distributed dataset formed by passing each element of the source through a function func.
filter(func)	Return a new dataset formed by selecting those elements of the source on which funcreturns true.
flatMap(func)	Similar to map, but each input item can be mapped to 0 or more output items (so funcshould return a Seq rather than a single item).
mapPartitions(func)	Similar to map, but runs separately on each partition (block) of the RDD, so func must be of type Iterator<T> => Iterator<U> when running on an RDD of type T.
mapPartitionsWithIndex(func)	Similar to mapPartitions, but also provides func with an integer value representing the index of the partition, so func must be of type (Int, Iterator<T>) => Iterator<U> when running on an RDD of type T.
sample(withReplacement, fraction, seed)	Sample a fraction fraction of the data, with or without replacement, using a given random number generator seed.

Transformations (cont.)

Transformation	Meaning
union(otherDataset)	Return a new dataset that contains the union of the elements in the source dataset and the argument.
intersection(otherDataset)	Return a new RDD that contains the intersection of elements in the source dataset and the argument.
distinct([numTasks]))	Return a new dataset that contains the distinct elements of the source dataset.
groupByKey([numTasks])	When called on a dataset of (K, V) pairs, returns a dataset of (K, Iterable<V>) pairs. Note: If you are grouping in order to perform an aggregation (such as a sum or average) over each key, using reduceByKey or aggregateByKey will yield much better performance. Note: By default, the level of parallelism in the output depends on the number of partitions of the parent RDD. You can pass an optional numTasks argument to set a different number of tasks.
reduceByKey(func, [numTasks])	When called on a dataset of (K, V) pairs, returns a dataset of (K, V) pairs where the values for each key are aggregated using the given reduce function func, which must be of type (V,V) => V. Like in groupByKey, the number of reduce tasks is configurable through an optional second argument.
aggregateByKey(zeroValue)(seqOp, combOp, [numTasks])	When called on a dataset of (K, V) pairs, returns a dataset of (K, U) pairs where the values for each key are aggregated using the given combine functions and a neutral "zero" value. Allows an aggregated value type that is different than the input value type, while avoiding unnecessary allocations. Like in groupByKey, the number of reduce tasks is configurable through an optional second argument.
sortByKey([ascending], [numTasks])	When called on a dataset of (K, V) pairs where K implements Ordered, returns a dataset of (K, V) pairs sorted by keys in ascending or descending order, as specified in the booleanascending argument.

Transformations (cont.)

Transformation	Meaning
join(otherDataset, [numTasks])	When called on datasets of type (K, V) and (K, W), returns a dataset of (K, (V, W)) pairs with all pairs of elements for each key. Outer joins are supported through leftOuterJoin,rightOuterJoin, and fullOuterJoin.
cogroup(otherDataset, [numTasks])	When called on datasets of type (K, V) and (K, W), returns a dataset of (K, (Iterable<V>, Iterable<W>)) tuples. This operation is also called groupWith.
cartesian(otherDataset)	When called on datasets of types T and U, returns a dataset of (T, U) pairs (all pairs of elements).
pipe(command, [envVars])	Pipe each partition of the RDD through a shell command, e.g. a Perl or bash script. RDD elements are written to the process's stdin and lines output to its stdout are returned as an RDD of strings.
coalesce(numPartitions)	Decrease the number of partitions in the RDD to numPartitions. Useful for running operations more efficiently after filtering down a large dataset.
repartition(numPartitions)	Reshuffle the data in the RDD randomly to create either more or fewer partitions and balance it across them. This always shuffles all data over the network.
repartitionAndSortWithinPartitions(partitioner)	Repartition the RDD according to the given partitioner and, within each resulting partition, sort records by their keys. This is more efficient than calling repartition and then sorting within each partition because it can push the sorting down into the shuffle machinery.

Actions

The following table lists some of the common actions supported by Spark. Refer to the RDD API doc (Scala, Java, Python, R)

and pair RDD functions doc (Scala, Java) for details.

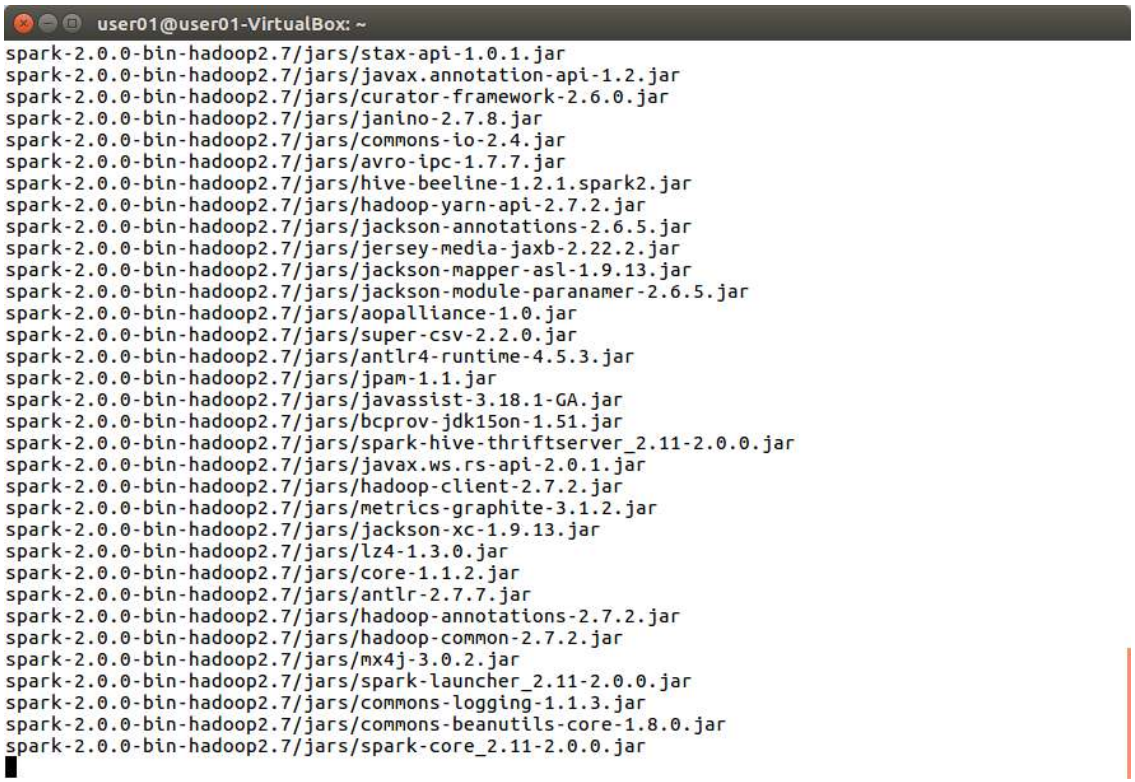
Action	Meaning
reduce(func)	Aggregate the elements of the dataset using a function func (which takes two arguments and returns one). The function should be commutative and associative so that it can be computed correctly in parallel.
collect()	Return all the elements of the dataset as an array at the driver program. This is usually useful after a filter or other operation that returns a sufficiently small subset of the data.
count()	Return the number of elements in the dataset.
first()	Return the first element of the dataset (similar to take(1)).
take(n)	Return an array with the first n elements of the dataset.
takeSample(withReplacement,num, [seed])	Return an array with a random sample of num elements of the dataset, with or without replacement, optionally pre-specifying a random number generator seed.
takeOrdered(n, [ordering])	Return the first n elements of the RDD using either their natural order or a custom comparator.

Actions (Cont.)

Action	Meaning
saveAsTextFile(path)	Write the elements of the dataset as a text file (or set of text files) in a given directory in the local filesystem, HDFS or any other Hadoop-supported file system. Spark will call toString on each element to convert it to a line of text in the file.
saveAsSequenceFile(path) **(Java and Scala)**	Write the elements of the dataset as a Hadoop SequenceFile in a given path in the local filesystem, HDFS or any other Hadoop-supported file system. This is available on RDDs of key-value pairs that implement Hadoop's Writable interface. In Scala, it is also available on types that are implicitly convertible to Writable (Spark includes conversions for basic types like Int, Double, String, etc).
saveAsObjectFile(path) **(Java and Scala)**	Write the elements of the dataset in a simple format using Java serialization, which can then be loaded usingSparkContext.objectFile().
countByKey()	Only available on RDDs of type (K, V). Returns a hashmap of (K, Int) pairs with the count of each key.
foreach(func)	Run a function func on each element of the dataset. This is usually done for side effects such as updating anAccumulator or interacting with external storage systems. Note: modifying variables other than Accumulators outside of the foreach() may result in undefined behavior. See Understanding closures for more details.

6. Installing Spark

$ tar –xvf spark-2.0.0-bin-hadoop2.7.tgz



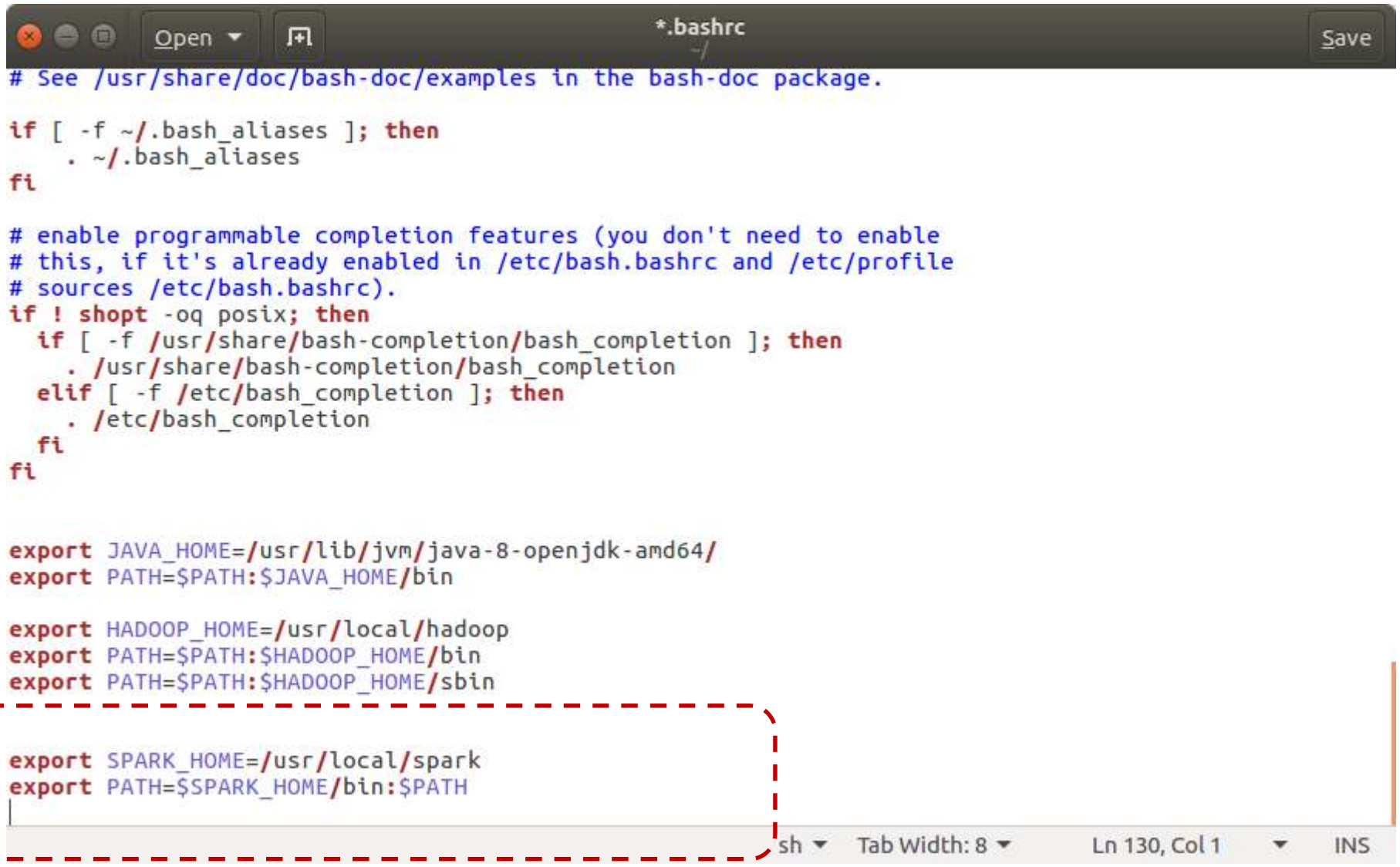
6. Installing Spark

$ sudo mv spark-2.0.0-bin-hadoop2.7 /usr/local/spark

$ cd
$ nano ~/.bashrc

```
…
export SPARK_HOME=/usr/local/spark
export PATH=$SPARK_HOME/bin:$PATH
```

6. Installing Spark



6. Installing Spark

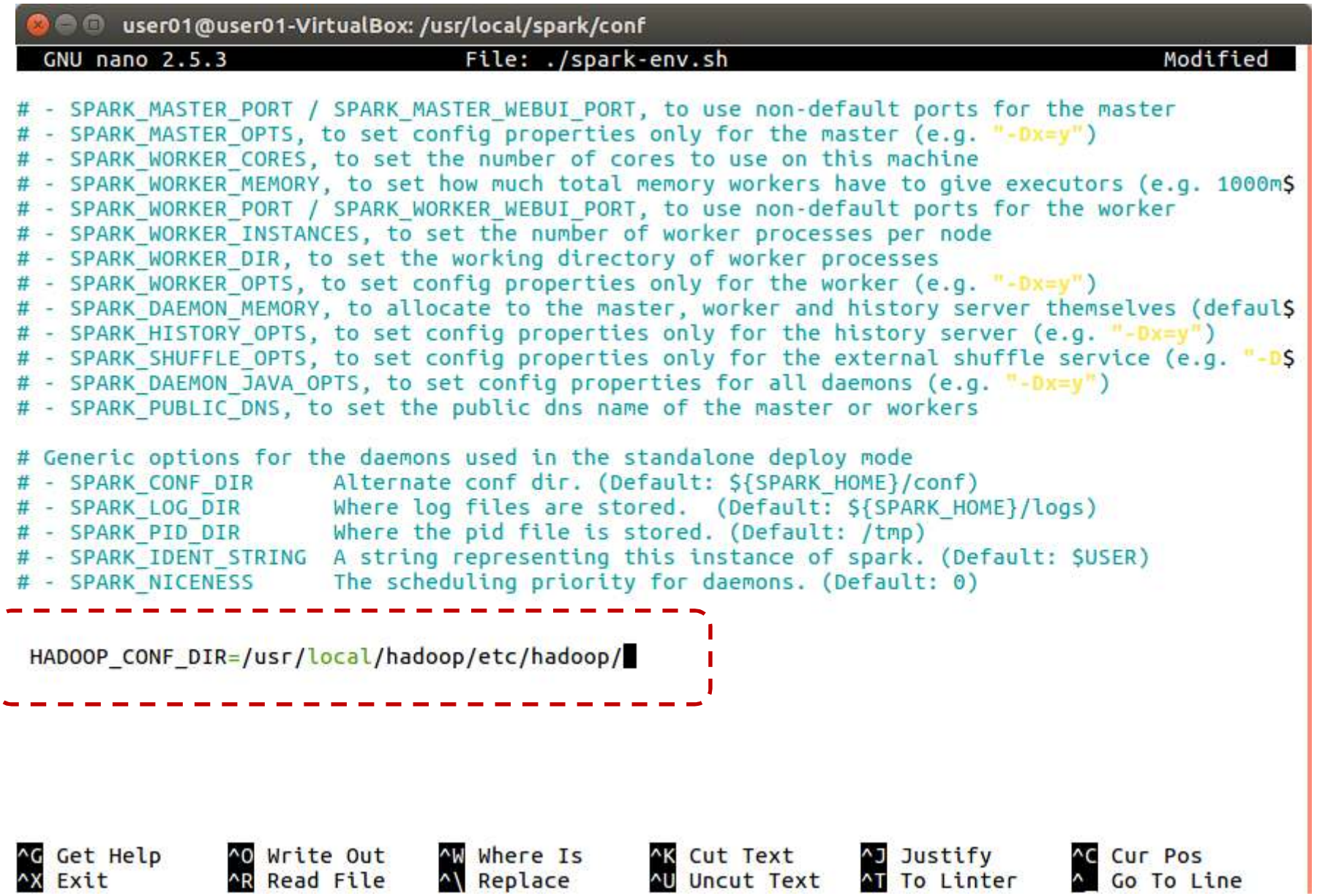
$source ~/.bashrc

$ cd /usr/local/spark/conf
$ cp spark-env.sh.template spark-env.sh
$ nano /usr/local/spark/conf/spark-env.sh

```
…
HADOOP_CONF_DIR=/usr/local/hadoop/etc/hadoop/
```

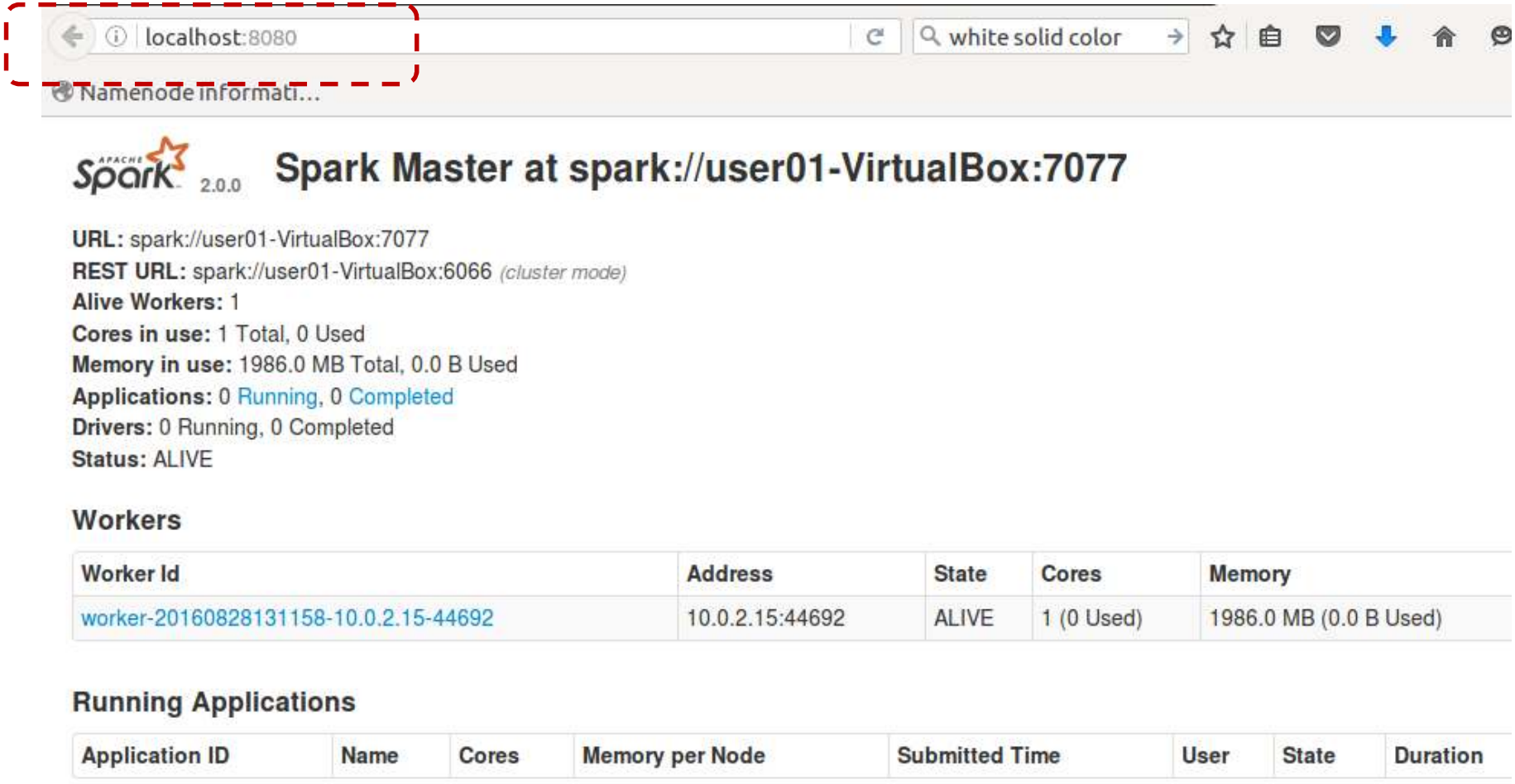
Notes: To be done in Master only, 5.In slaves file in /usr/local/spark/conf, add the slaves hostname [If slaves file is not found, copy slaves.template to slaves file]

6. Installing Spark



6. Installing Spark

$ /usr/local/spark/sbin/start-all.sh
http://localhost:8080



7. Test Submit Application to Spark

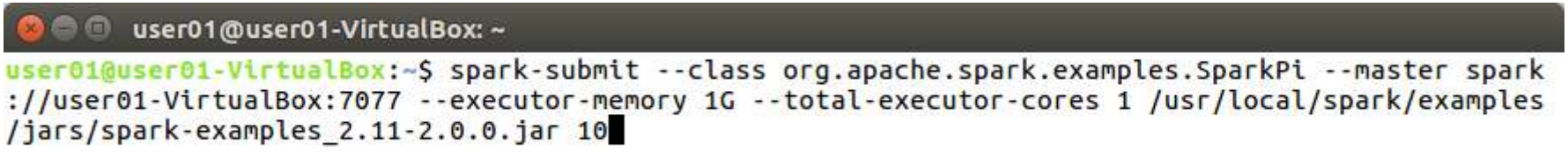
$ spark-submit --class org.apache.spark.examples.SparkPi --master spark://user01-VirtualBox:7077 --executor-memory 1G --total-executor-cores 1 /usr/local/spark/examples/jars/spark-examples_2.11-2.0.0.jar 10

–class: The entry point for your application.

–master: The master URL for the cluster.

–executor-memory: Specify memory to be allocated for the application

–total-executor-cores: Specify no of cpu cores to be allocated for the application

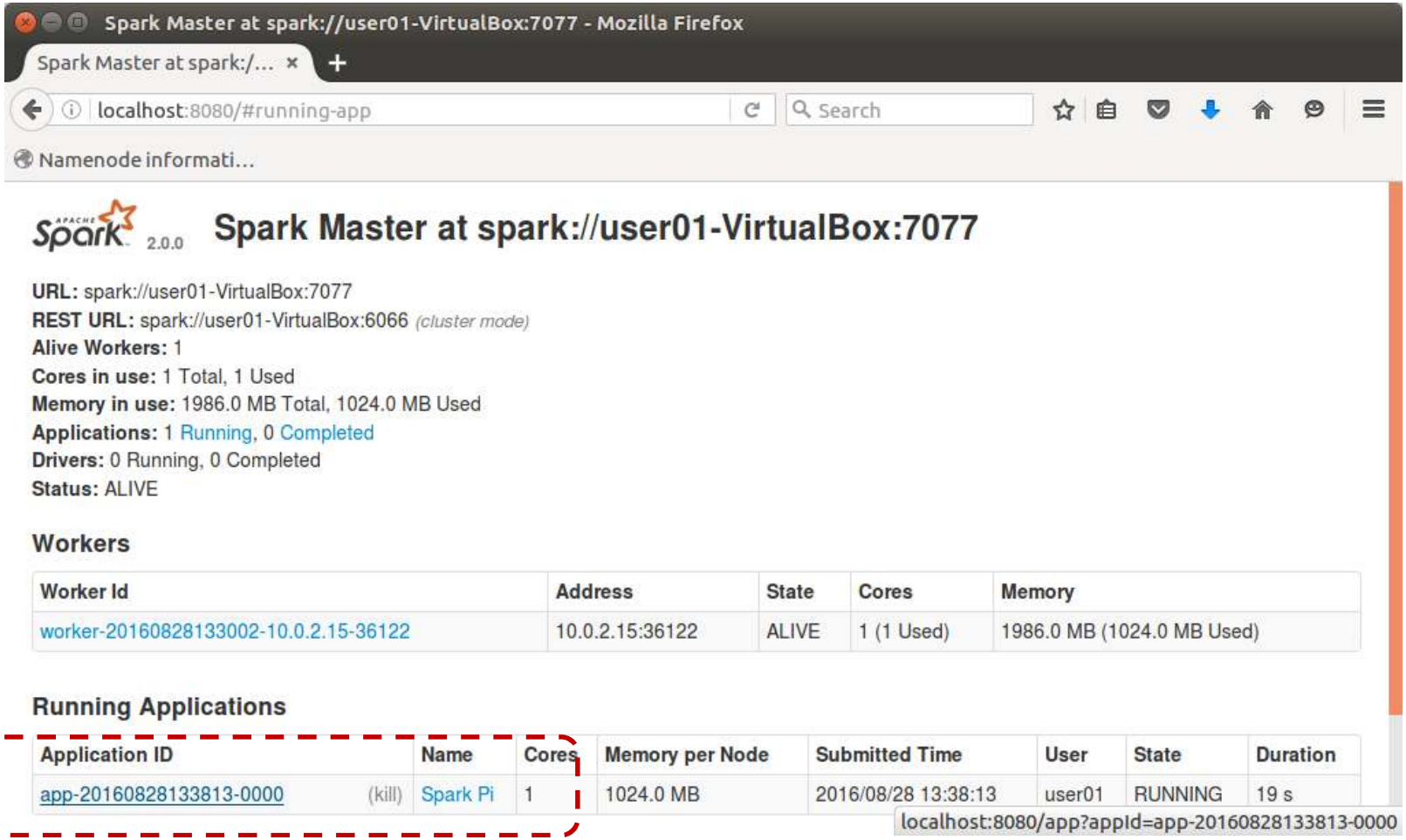


7. Test Submit Application to Spark

```
user01@user01-VirtualBox: ~
orage/rdd,null,AVAILABLE}
16/08/28 13:38:11 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@64beebb7{/st
orage/rdd/json,null,AVAILABLE}
16/08/28 13:38:11 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@7813cb11{/en
vironment,null,AVAILABLE}
16/08/28 13:38:11 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@bcec031{/env
ironment/json,null,AVAILABLE}
16/08/28 13:38:11 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@21005f6c{/ex
ecutors,null,AVAILABLE}
16/08/28 13:38:11 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@32f0fba8{/ex
ecutors/json,null,AVAILABLE}
16/08/28 13:38:11 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@545de5a4{/ex
ecutors/threadDump,null,AVAILABLE}
16/08/28 13:38:11 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@29ef6856{/ex
ecutors/threadDump/json,null,AVAILABLE}
16/08/28 13:38:11 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@ab7a938{/sta
tic,null,AVAILABLE}
16/08/28 13:38:11 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@3faf2e7d{/,n
ull,AVAILABLE}
16/08/28 13:38:11 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@4648ce9{/api
,null,AVAILABLE}
16/08/28 13:38:11 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@569bf9eb{/st
ages/stage/kill,null,AVAILABLE}
16/08/28 13:38:12 INFO server.ServerConnector: Started ServerConnector@4d9c69e2{HTTP/1.1}{0.0.0.0
:4040}
16/08/28 13:38:12 INFO server.Server: Started @5635ms
16/08/28 13:38:12 INFO util.Utils: Successfully started service 'SparkUI' on port 4040.
16/08/28 13:38:12 INFO ui.SparkUI: Bound SparkUI to 0.0.0.0, and started at http://10.0.2.15:4040
16/08/28 13:38:12 INFO spark.SparkContext: Added JAR file:/usr/local/spark/examples/jars/spark-ex
amples_2.11-2.0.0.jar at spark://10.0.2.15:43959/jars/spark-examples_2.11-2.0.0.jar with timestam
p 1472366292200
16/08/28 13:38:12 INFO client.StandaloneAppClient$ClientEndpoint: Connecting to master spark://us
er01-VirtualBox:7077...
```

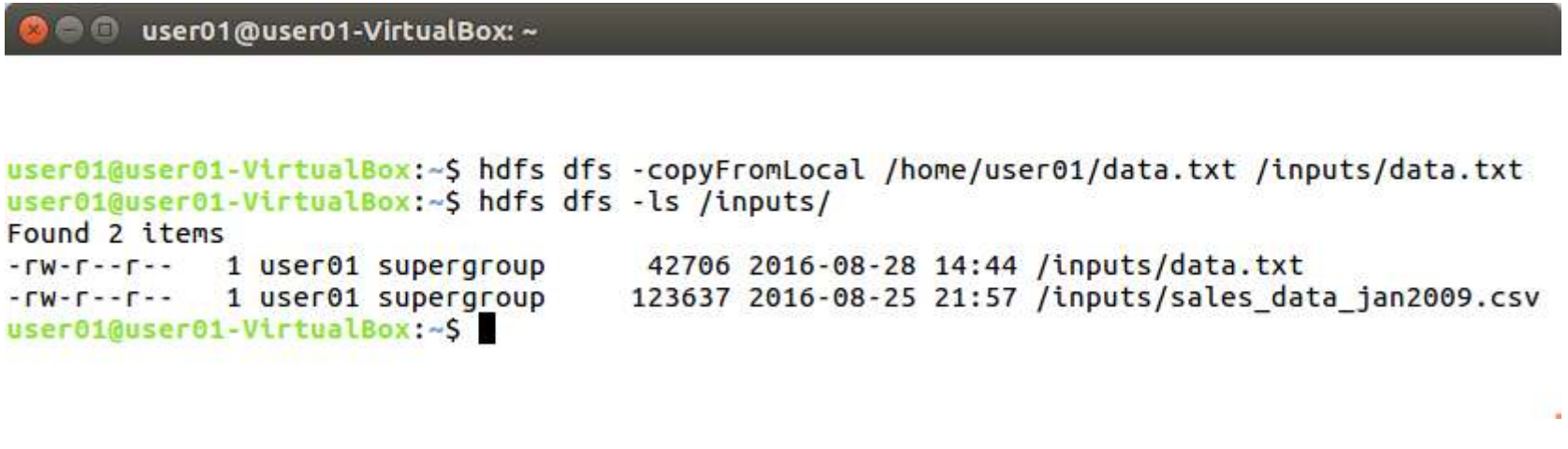
7. Test Submit Application to Spark

http://10.0.2.15:8080/



8. Test Core Spark RDD on HDFS Data

$ hdfs dfs -copyFromLocal /home/user01/data.txt /inputs/data.txt
$ hdfs dfs -ls /inputs/



8. Test Core Spark RDD on HDFS Data

$ spark-shell spark://10.0.2.15:7077/

8. Test Core Spark RDD on HDFS Data

WordCount Processing

scala> var hFile = sc.textFile("hdfs://10.0.2.15:9000/inputs/data.txt")
scala> val wc = hFile.flatMap(line => line.split(" ")).map(word => (word, 1)).reduceByKey(_ + _)

```
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.0.0
      /_/

Using Scala version 2.11.8 (OpenJDK 64-Bit Server VM, Java 1.8.0_91)
Type in expressions to have them evaluated.
Type :help for more information.

scala> var hFile = sc.textFile("hdfs://10.0.2.15:9000/inputs/data.txt")
hFile: org.apache.spark.rdd.RDD[String] = hdfs://10.0.2.15:9000/inputs/data.txt MapPartitionsRDD[
1] at textFile at <console>:24

scala> val wc = hFile.flatMap(line => line.split(" ")).map(word => (word, 1)).reduceByKey(_ + _)
wc: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[4] at reduceByKey at <console>:26

scala>
```

Note: sc is the object of SparkContext

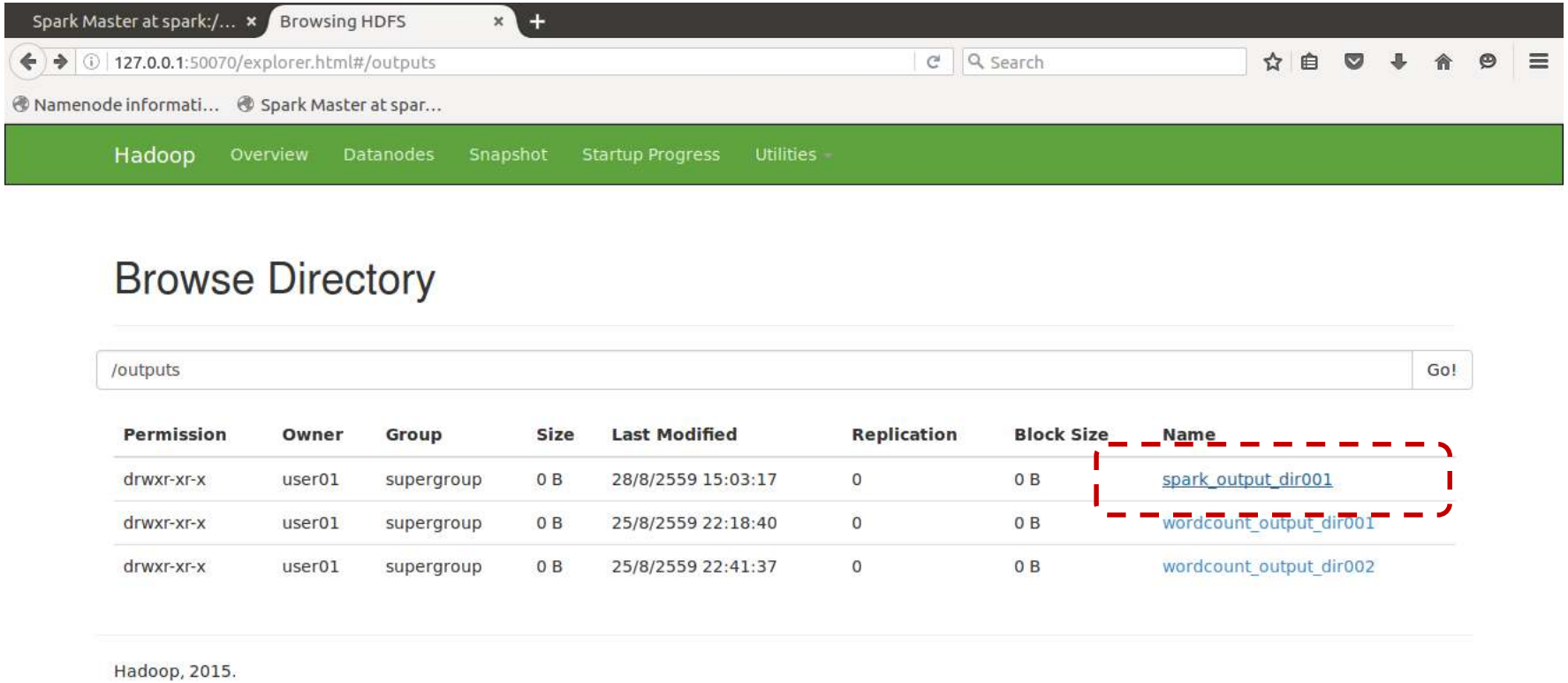
8. Test Core Spark RDD on HDFS Data

display first 10 results
scala> wc.take(10)

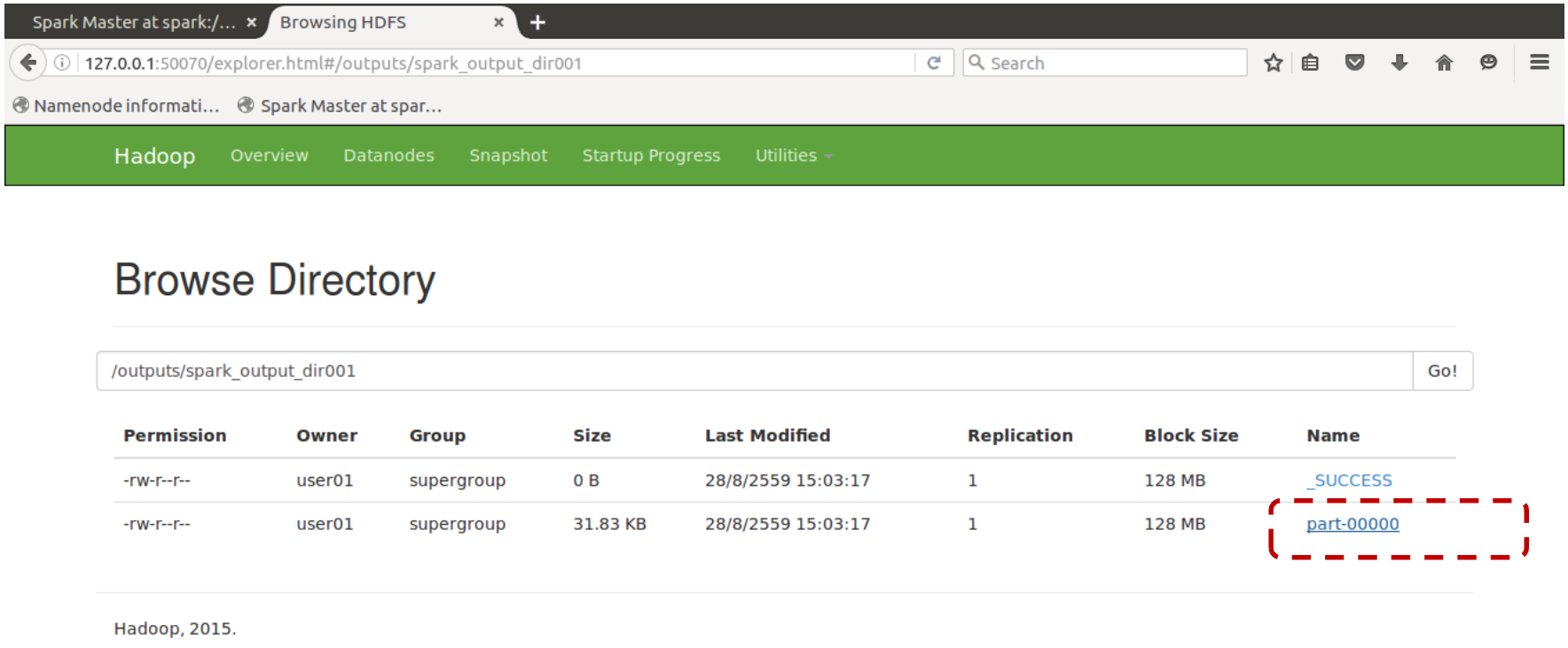
scala>
wc.saveAsTextFile("hdfs://10.0.2.15:9000/outputs/spark_output_dir001")
scala> :q

```
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.0.0
      /_/

Using Scala version 2.11.8 (OpenJDK 64-Bit Server VM, Java 1.8.0_91)
Type in expressions to have them evaluated.
Type :help for more information.

scala> var hFile = sc.textFile("hdfs://10.0.2.15:9000/inputs/data.txt")
hFile: org.apache.spark.rdd.RDD[String] = hdfs://10.0.2.15:9000/inputs/data.txt MapPartitionsRDD[
1] at textFile at <console>:24

scala> val wc = hFile.flatMap(line => line.split(" ")).map(word => (word, 1)).reduceByKey(_ + _)
wc: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[4] at reduceByKey at <console>:26

scala> wc.take(10)
res0: Array[(String, Int)] = Array((hippopotamus-sized,1), (favour,1), (1881),,1), (someone,1), (
young,2), (surgery,1), (House,2), (Glyptodon,,1), (ship,,1), (urging,1))

scala> wc.saveAsTextFile("hdfs://10.0.2.15:9000/outputs/spark_output_dir001")

scala>
```

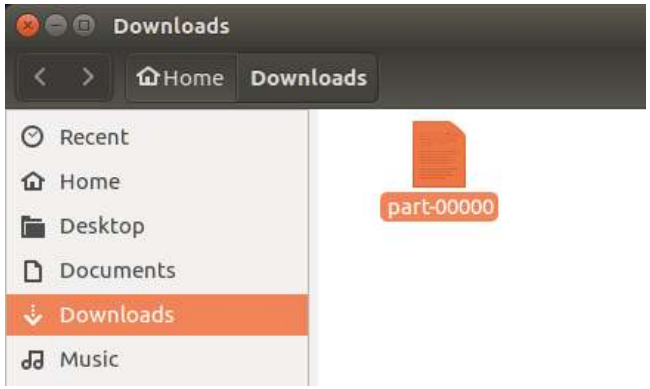
8. Test Core Spark RDD on HDFS Data

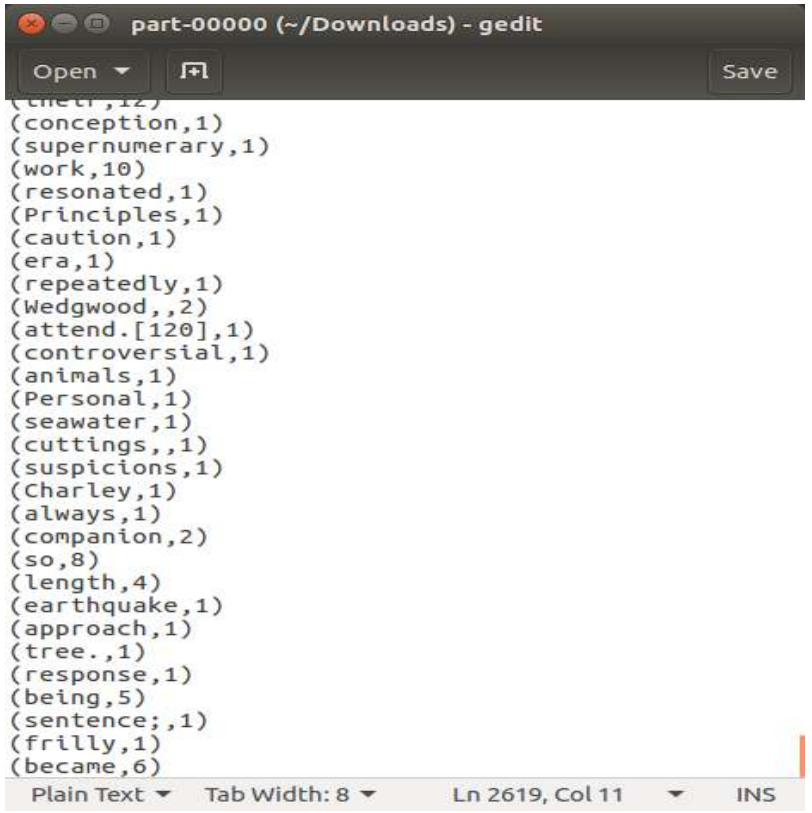


8. Test Core Spark RDD on HDFS Data



8. Test Core Spark RDD on HDFS Data





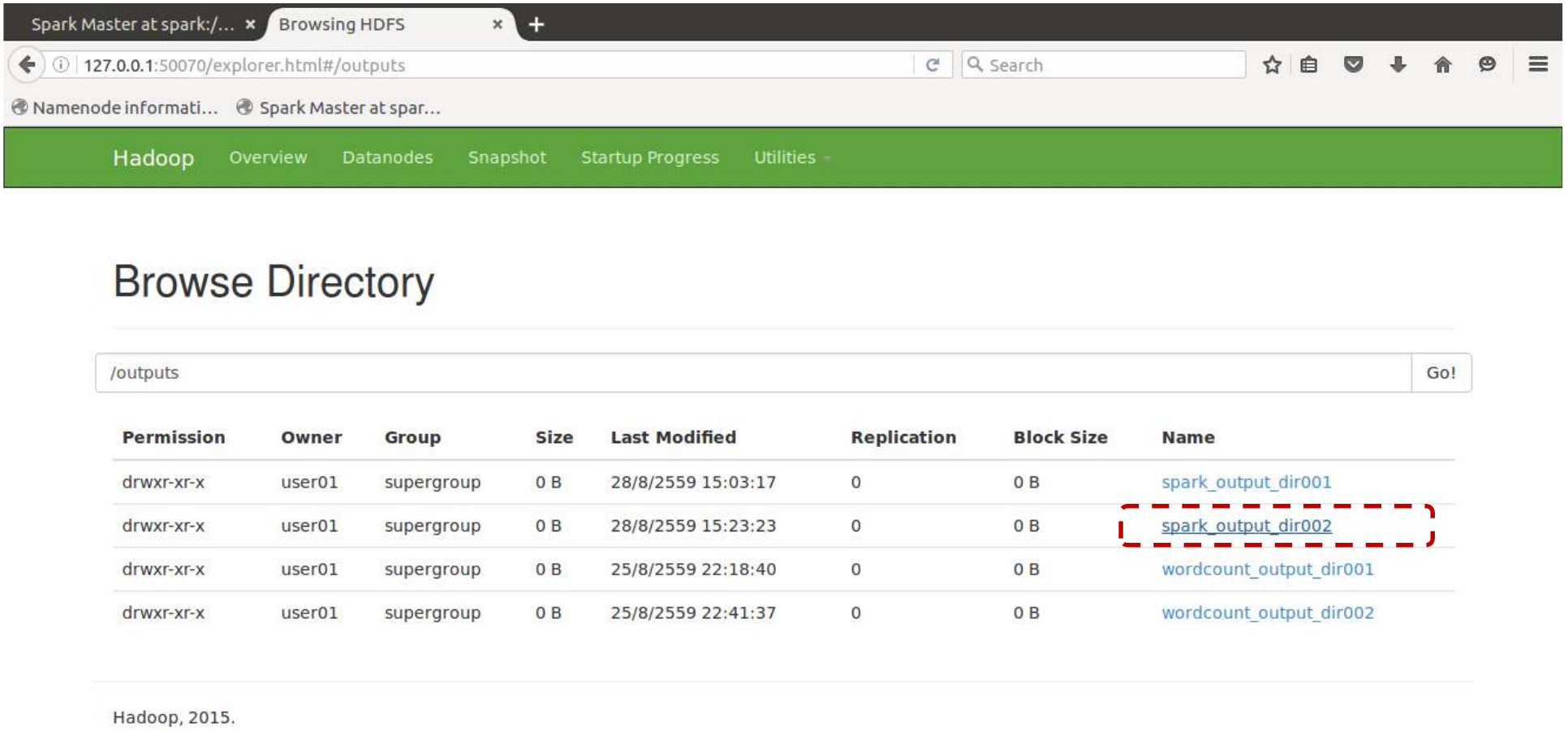
$ hdfs dfs -ls /outputs/spark_output_dir001/

$ hdfs dfs -ls /outputs/spark_output_dir001/

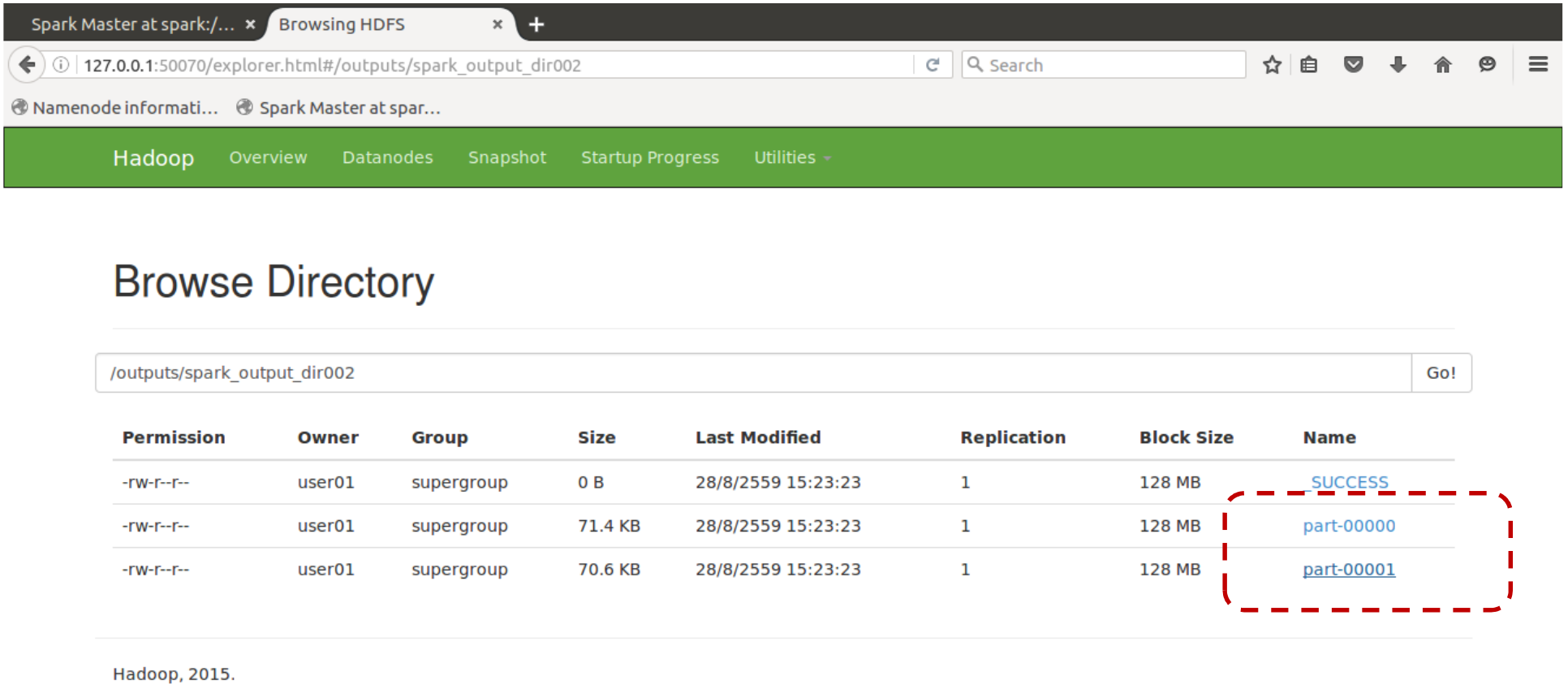
8. Test Core Spark RDD on HDFS Data

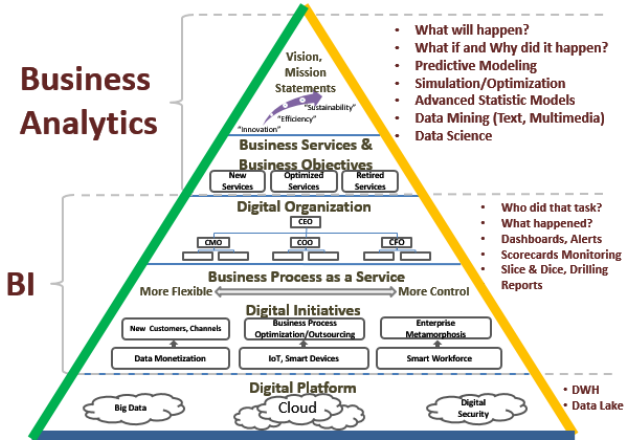
```
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.0.0
      /_/

Using Scala version 2.11.8 (OpenJDK 64-Bit Server VM, Java 1.8.0_91)
Type in expressions to have them evaluated.
Type :help for more information.

scala> var hFile = sc.textFile("hdfs://10.0.2.15:9000/inputs/*")
hFile: org.apache.spark.rdd.RDD[String] = hdfs://10.0.2.15:9000/inputs/* MapPartitionsRDD[1] at t
extFile at <console>:24

scala> val wc = hFile.flatMap(line => line.split(" ")).map(word => (word, 1)).reduceByKey(_ + _)
wc: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[4] at reduceByKey at <console>:26

scala> wc.take(10)
res0: Array[(String, Int)] = Array((someone,1), (6:24,2/23/09,1), (House,3), (8:23,Product1,1200,
Mastercard,Sarah,Floyds,1), (11:35,39.96111,-82.99889,1), (20:11,1/13/09,1), (order,2), (7:06,47.
45,18.9666667,1), (18:48,40.5945,-74.6244,1), (pigeon,1))

scala> wc.saveAsTextFile("hdfs://10.0.2.15:9000/outputs/spark_output_dir002")

scala>
```

8. Test Core Spark RDD on HDFS Data



8. Test Core Spark RDD on HDFS Data





Business Analytics vs. Business Intelligence

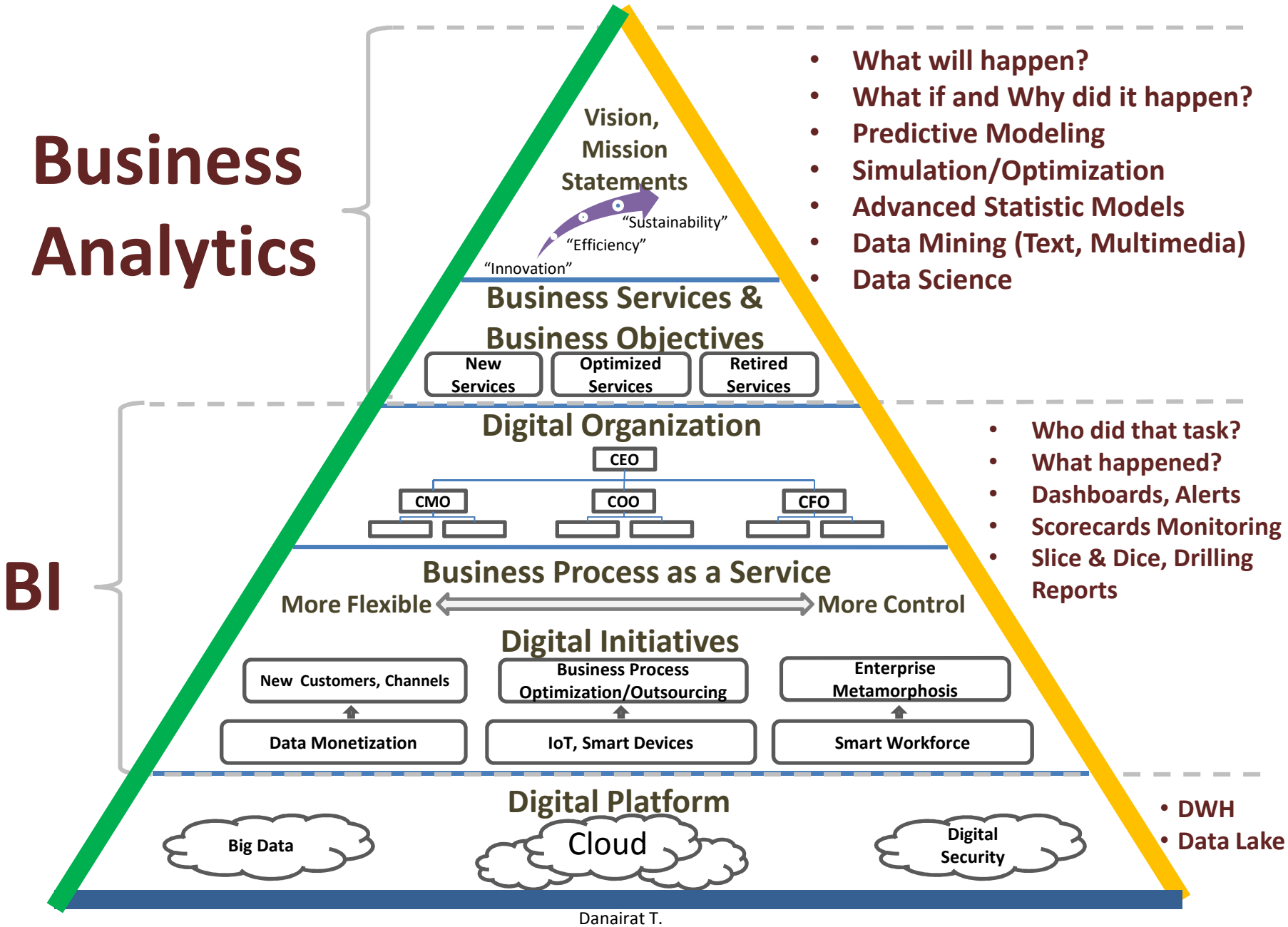
อ.ดนัยรัฐ ธนบดีธรรมจารี

Line ID: Danairat

FB: Danairat Thanabodithammachari

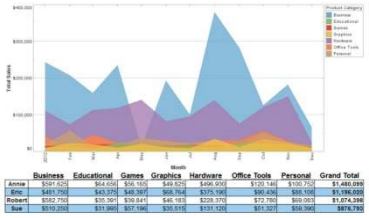
+668-1559-1446

Business Analytics and Business Intelligence



Danairat T.

What is Business Intelligence (BI)?

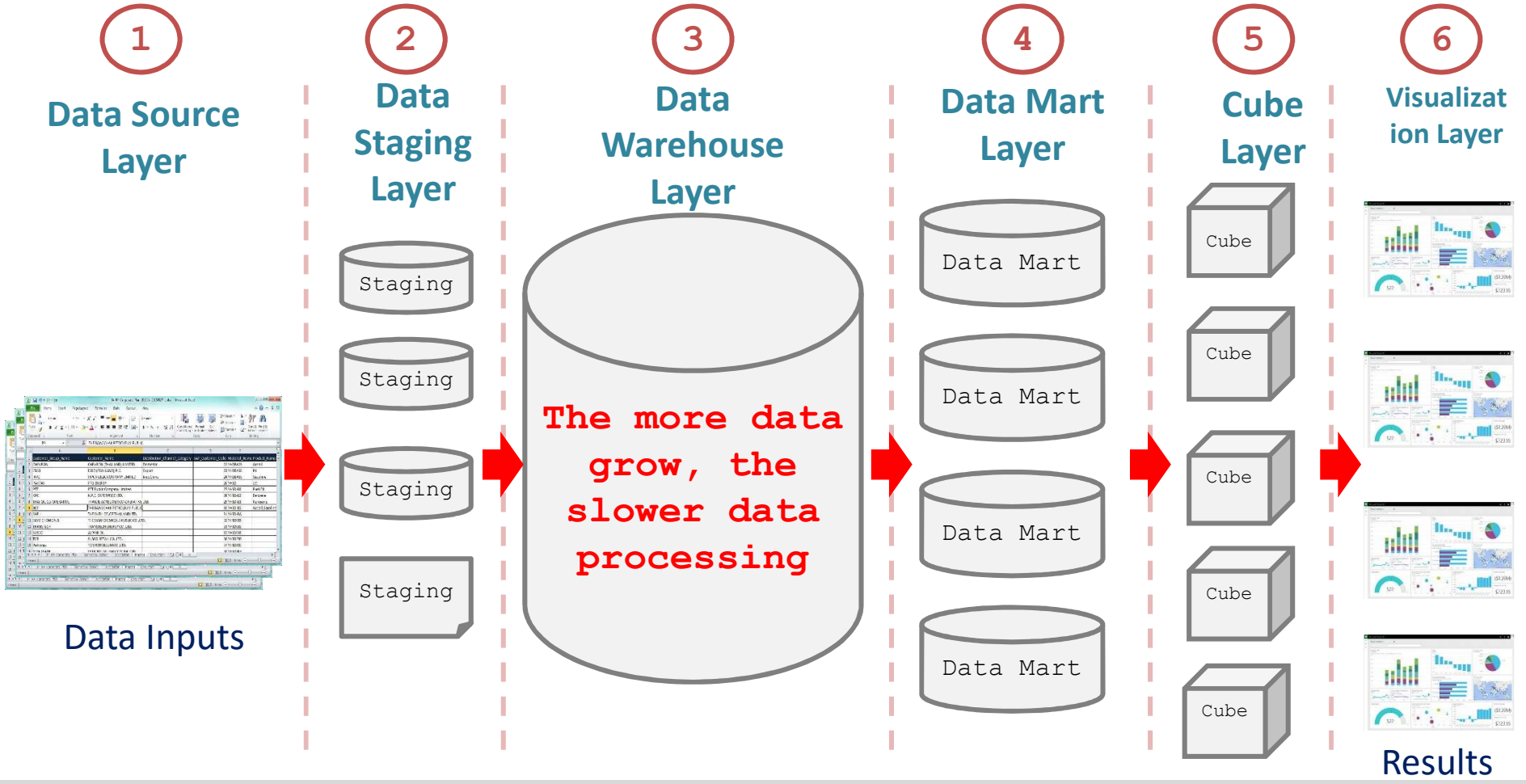


Business intelligence systems are used to maintain, optimize and streamline current operations. BI improves and maintains operational efficiency and helps businesses **increase organizational productivity**. Business intelligence software confers many benefits, notably **powerful reporting and data analysis** capabilities. Using BI's rich visualization mechanisms, managers are able to generate intuitive, readable reports that contain relevant, actionable data.

Popular business intelligence solutions include; SAP BusinessObjects, QlikView, IBM Cognos, Microstrategy, etc.

https://selecthub.com/business-intelligence/business-intelligence-vs-business-analytics/

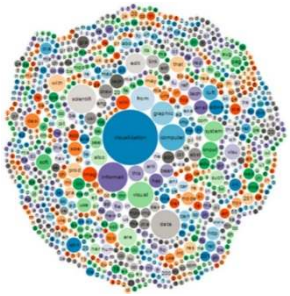
Traditional Data Warehouse

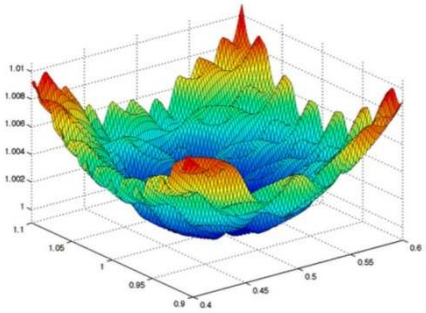
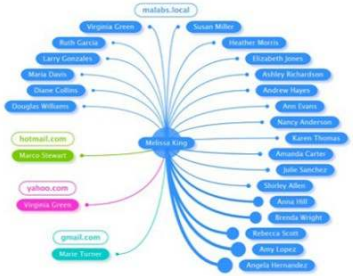


Top Concerns from Traditional Data Warehouse Architecture

1. A lot of data duplication lead to cost of data store/storage issue
2. Very slow of data processing and need to restart/roll back the job if any failed
3. Data security issue due to keep data too many copies and various formats

What is Business Analytics (BA)?



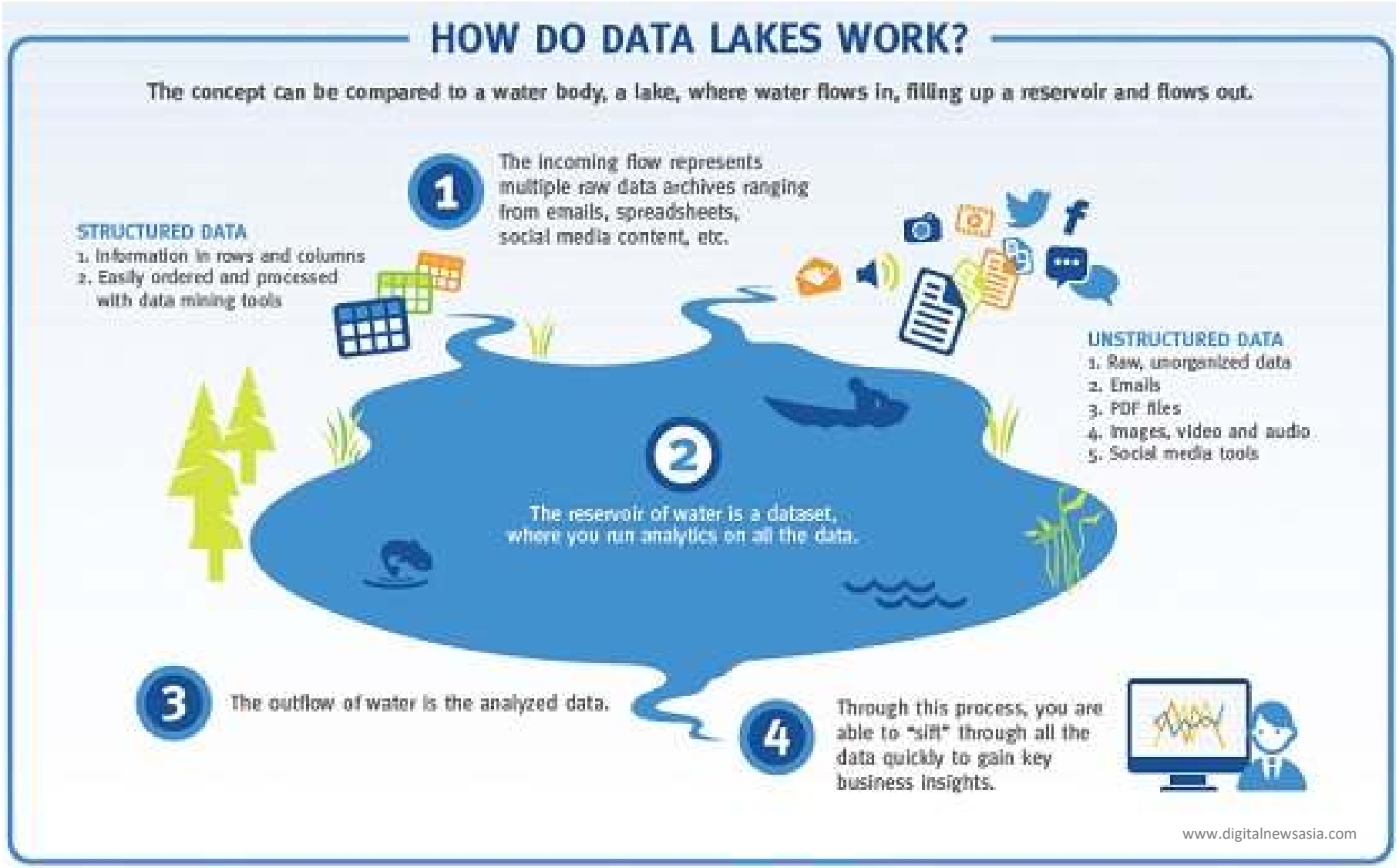



Like business intelligence, BA collects and analyzes data, employs **predictive analytics** and generates **richly visualized reports**, helping identify and address an organization's weak points. That's where similarities end. Business analytics software is used to explore and analyze historical and current data. It utilizes **statistical analysis, data mining** and quantitative analysis to identify past business trends.

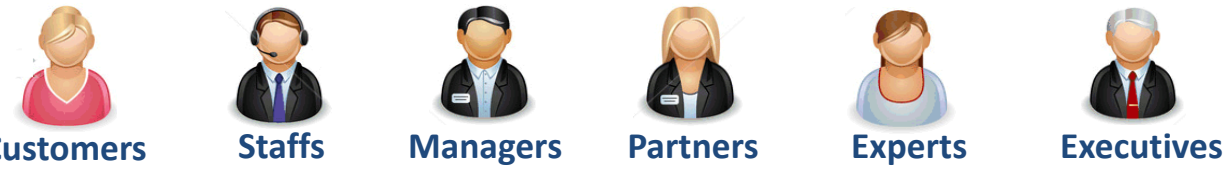
Popular business analytics solutions include; SAP Business Analytics Suite, Pentaho BA, Birst BI and Tableau BIg Data Analytics.

https://selecthub.com/business-intelligence/business-intelligence-vs-business-analytics/

Data Lake


HOW DO DATA LAKES WORK?
The concept can be compared to a water body, a lake, where water flows in, filling up a reservoir and flows out.
1
The incoming flow represents multiple raw data archives ranging from emails, spreadsheets, social media content, etc.
STRUCTURED DATA
1. Information in rows and columns
2. Easily ordered and processed with data mining tools
UNSTRUCTURED DATA
1. Raw, unorganized data
2. Emails
3. PDF files
4. Images, video and audio
5. Social media tools
2
The reservoir of water is a dataset, where you run analytics on all the data.
3
The outflow of water is the analyzed data.
4
Through this process, you are able to "sift" through all the data quickly to gain key business insights.
www.digitalnewsasia.com

Big Data for Business Analytics Platform



Big Data Applications

Next Best Action | BI/Report | Fraud Analysis | Cyber Security | Talent Search

Big Data Platform

Monitoring, Resource Management and Metadata Framework

Descriptive/ Diagnose Analytics | Predictive Analytics | Prescriptive Analytics

Parallel Data Processing, Refinery

Distributed Data Store, Data Lake

Ingestion and Acquisition

Security and Control Framework

Big Data Infrastructure

Compute Storage, Network

Example of Big Data Deployment Architecture

Access Channels

System Monitoring

App1

App2

Executive Reports

Advanced Analytics

System Security

Application Server

NoSQL with Maps DB

Hadoop

HDFS, YARN, Spark, Storn, MRv2, Hive, Mahout, Zookeeper, Mesos, Etc.

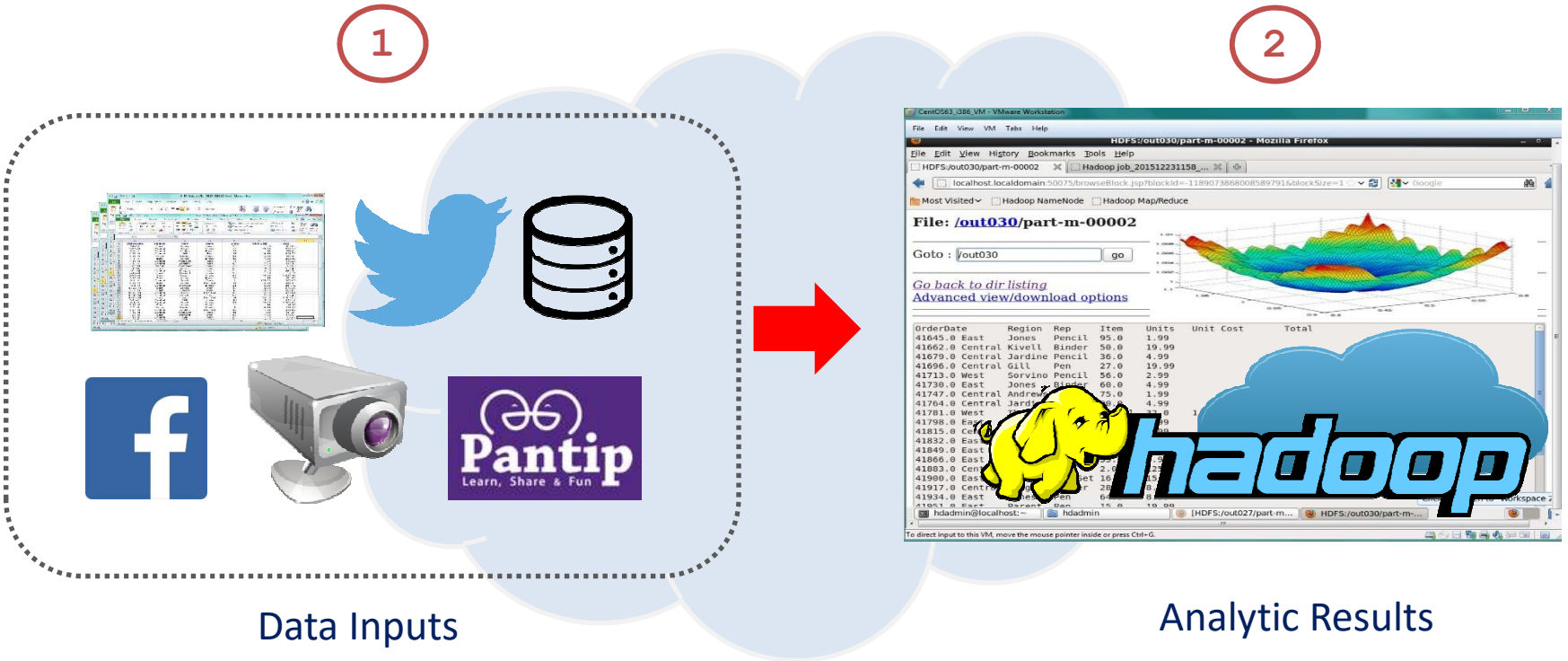
Storage Server and Network Infrastructure

Physical Data Center

Access | Apps | Platform | Infra

Core Hadoop processing

NO data staging transformation and NO data move required!!

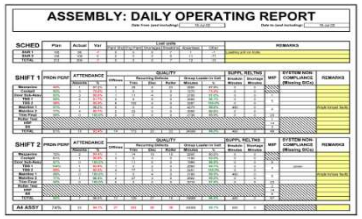


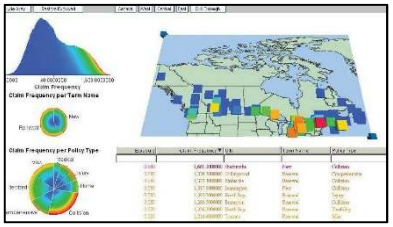
Top Benefits

1. Cloud and IoT ready architecture roadmap
2. No data duplication with reduce cost of data store/storage
3. Fast data processing and all processing are built-in fault tolerant
4. Align with unify data architecture and data governance
5. Less steps of data processing comparing with traditional DWH

The Effort Investment:-

1. Learn core Hadoop





Big Data Analytics

อ.ดนัยรัฐ ธนบดีธรรมจารี

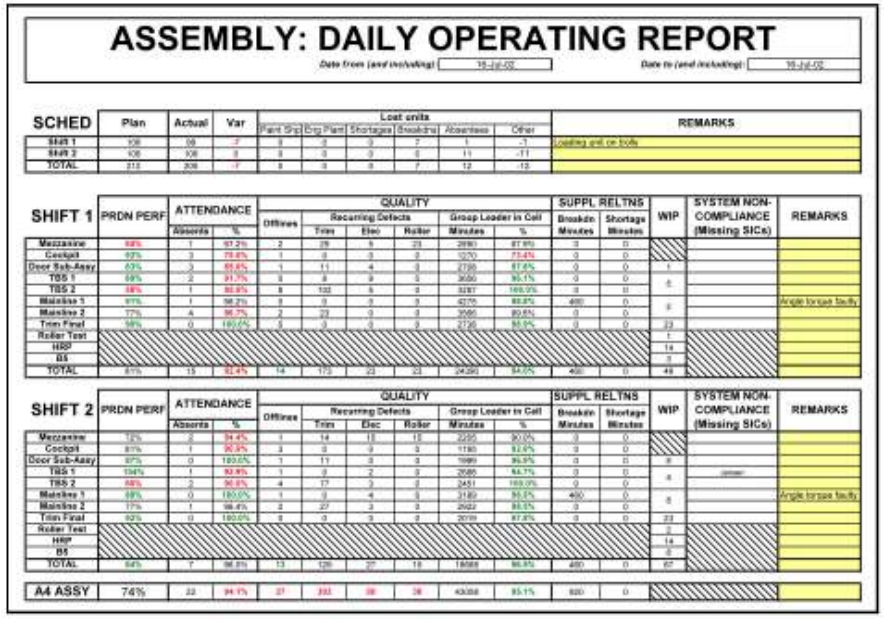
Line ID: Danairat

FB: Danairat Thanabodithammachari

+668-1559-1446

Descriptive/Diagnostic analytics

Descriptive/Diagnostic analytics answers the question, "What happened in the business?" It looks at data and information to describe the current business situation in a way that trends, patterns and exceptions become apparent. This takes the form of reports, dashboards, MIS, etc.



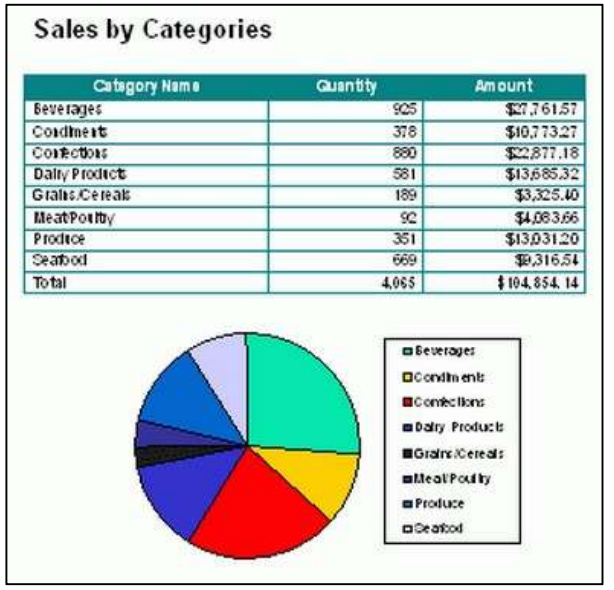
ASSEMBLY: DAILY OPERATING REPORT

SCHED	Plan	Actual	Var	Lost units	REMARKS
Shift 1	108	108	-7	8 0 0 3 7 1 -7	Loading grid on trolls
Shift 2	108	108	0	9 0 0 0 0 11 -11	
TOTAL	216	208	-7	8 0 0 3 7 12 -18	

SHIFT 1	PRDN PERF	ATTENDANCE		QUALITY					SUPPL RELTNS		WIP	SYSTEM NON-COMPLIANCE (Missing SICs)	REMARKS
		Absents	%	Offlines	Recurring Defects: Trim	Recurring Defects: Elec	Recurring Defects: Roller	Group Leader in Cell: Minutes	Breakdn Minutes	Shortage Minutes			
Mezzanine	68%	1	97.2%	2	29	5	23	2990	0	0			
Cockpit	93%	3	79.6%	1	0	3	0	1070	0	0			
Door Sub-Assy	63%	3	85.6%	1	11	4	0	2700	0	0	1		
TBS 1	89%	2	91.7%	3	8	9	0	3650	0	0	6		
TBS 2	88%	1	80.8%	8	102	5	0	3287	0	0			
Mainline 1	91%	1	98.2%	3	0	0	0	4275	480	0	6		Angle torque faulty
Mainline 2	77%	4	96.2%	2	22	0	0	3060	0	0			
Trim Final	98%	0	100.0%	5	0	0	0	2726	0	0	23		
Roller Test											1		
HRP											14		
05											2		
TOTAL	87%	15	92.4%	14	176	26	23	24,995	480	0	49		

SHIFT 2	PRDN PERF	ATTENDANCE		QUALITY					SUPPL RELTNS		WIP	SYSTEM NON-COMPLIANCE (Missing SICs)	REMARKS
		Absents	%	Offlines	Recurring Defects: Trim	Recurring Defects: Elec	Recurring Defects: Roller	Group Leader in Cell: Minutes	Breakdn Minutes	Shortage Minutes			
Mezzanine	72%	2	94.4%	1	14	18	18	2295	0	0			
Cockpit	87%	1	90.3%	3	0	9	0	1190	0	0			
Door Sub-Assy	97%	0	100.0%	1	11	3	3	1999	0	0	8		
TBS 1	104%	1	92.8%	1	0	2	3	2685	0	0	6		
TBS 2	88%	2	90.8%	4	17	3	0	2451	0	0			
Mainline 1	88%	0	100.0%	1	0	4	5	3180	480	0	6		Angle torque faulty
Mainline 2	77%	1	98.4%	2	27	3	0	2622	0	0			
Trim Final	93%	0	100.0%	5	0	3	0	2019	0	0	23		
Roller Test											2		
HRP											14		
05											3		
TOTAL	84%	7	96.8%	13	126	27	19	18668	480	0	67		

A4 ASSY	74%	22	94.1%	27	332	38	26	43068	85.1%	820	0		



Sales by Categories

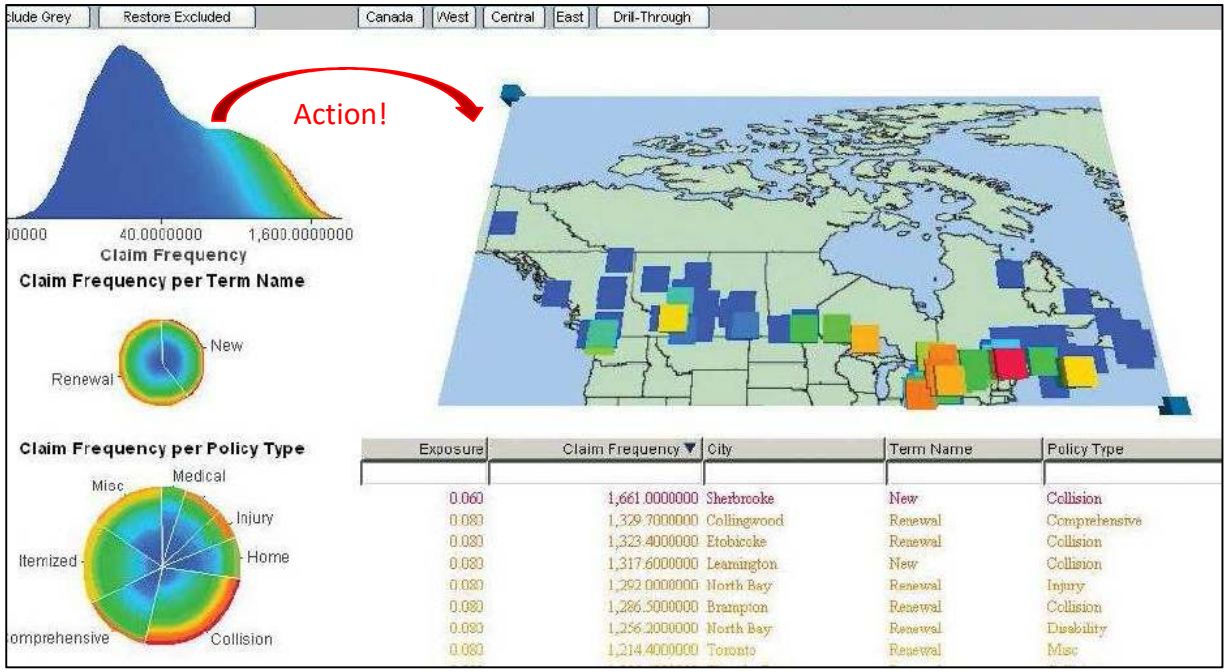
Category Name	Quantity	Amount
Beverages	925	$27,761.57
Condiments	378	$10,773.27
Confections	880	$22,877.18
Dairy Products	581	$13,685.32
Grains/Cereals	189	$3,325.40
Meat/Poultry	92	$4,083.66
Produce	351	$13,031.20
Seafood	669	$9,316.54
Total	4,065	$104,854.14

Predictive analytics

Predictive analytics answers the question, "What is likely to happen in the future?" Here data modeling and forecasting are used to determine future possibilities

Prescriptive analytics

Prescriptive analytics is the combination of the above to provide answers to the "So what?" and the "Now what?" For example, what should a business do to retain key customers? How can businesses improve their supply chain to enhance service levels while reducing costs?



Big Data Project Life Cycle Management

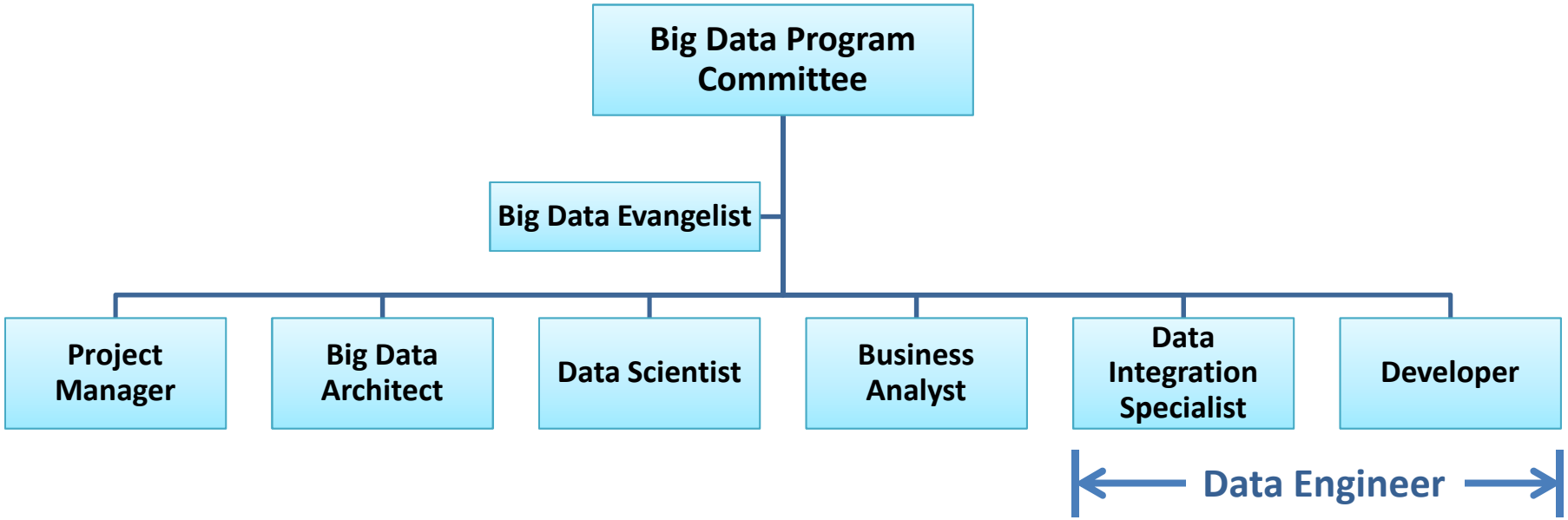
อ.ดนัยรัฐ ธนบดีธรรมจารี

Line ID: Danairat

FB: Danairat Thanabodithammachari

+668-1559-1446

Big Data People and Team Structure



Big Data Team Structure

No.	Roles	Description
1	**Big Data Program Committee**	The Team to develop Big Data initiative and deliver solution results
2	**Big Data Evangelist**	The business evangelist must have a combination of good communication and presentation skills and deep contextual business knowledge, as well as a clear understanding of technology in general and big data techniques.
3	**Project Manager**	The project manager “owns” the development schedule and is expected to ensure that the right architects, designers, and developers are brought into the project at the right times.
4	**Big Data Architect**	The person who has background in parallel and distributed computing architecture. This person is knowledgeable about fundamental performance “gotchas” that will impede the speed, scalability, and extensibility of any application requiring massive data volumes.

Big Data Team Structure

No.	Roles	Description
5	**Data Scientist**	The data scientist combines knowledge of computer science, the use of high-performance applications, and statistics, economics, mathematics, and probabilistic analysis skills.
6	**Business Analyst**	The person who engages with the business process owners and solicits their needs and expectations. Business analysts who are able to effectively translate business expectations into specific data analysis results.
7	**Data Integration Specialist**	The person who has experience in data extraction, transformation, loading, and data transformations in preparation for cleansing and delivery to target systems. Seek people with experience with data federation and virtualization, data quality, and metadata analysis.
8	**Application Developer**	The technical resources with the right set of skills for programming and testing parallel and distributed applications.

Big Data Project Life Cycle

1. Identify Targeted Users
2. Identify Target Opportunities / Key Measurements
3. Identify Data Sources/Types
4. Identify Data Capturing Approaches
5. Identify Data Processing and Visualization Planning
6. Identify Big Data Platform
7. Identify Security
8. Identify Governance and Change Control for Operation
9. Identify Team Structure
10. Identify Phasing, Budget and Cost

1. Develop Use Cases
2. Develop Requirements Definition
3. Develop Big Data Solution Framework
4. Develop the Development and Test Environment
5. Develop Data Capture
6. Develop Analytics
7. Integrate Visualization
8. Manage Assets and Configurations

1. Monitor Big Data Platform Availability, Utilization and Capacity Planning
2. Manage Operation Tasks (Admin. Scripts, Commands), Data Capturing System, Upgrading or Patching
3. Manage Service Requests and Incidents
4. System admin. Training
5. Helpdesk Training
6. End-User Training (Analytic Results)

1. Adoption Rates for each analytics results
2. No. of Missing Analytic Results
3. No. of Missing Data
4. Lost hours per month
5. Avg. of each Analytic Result Response Time
6. No. of Technology System Failure per month
7. Evaluate Activity Conformance with Policies

Key Activities, People and Deliverables

No.	Phases	Activities	People	Deliverables
1	Planning	Identify Targeted Users	Big Data Program Committee	Big Data Discovery Worksheet
2	Planning	Identify Target Opportunities	Big Data Program Committee	Big Data Discovery Worksheet
3	Planning	Identify Team Structure	Big Data Program Committee	Team Organization Chart
4	Planning	Identify Data Sources/Types	Big Data Architect, Data Scientist, Data Integration Specialist	Data Sources Types Information
5	Planning	Identify Data Capturing Approaches	Data Integration Specialist, Data Scientist	Data Capturing Information
6	Planning	Identify Data Processing and Visualization Planning	Business Analyst, Big Data Architect, Data Scientist, Developer	Data Processing Framework and User Interface Summary
7	Planning	Identify Big Data Platform	Big Data Architect, Project Manager	Big Data Platform Summary
8	Planning	Identify Security	Corporate Information Security, Big Data Architect, Project Manager	Security Scope Summary
9	Planning	Identify Governance and Change Control for Operation	Internal Control Team, Corporate Information Security, Big Data Architect, Project Manager	Governance, RACI, Change Procedures Summary
10	Planning	Identify Phasing Budget and Cost	CIO, CFO, Project Manager, Business Analyst	Project Investment Summary

Key Activities, People and Deliverables

No.	Phases	Activities	People	Deliverables
1	Development	Develop Use Cases	Business Users, Business Analyst, Big Data Architect, Big Data Evangelist	Use Cases Summary
2	Development	Develop Requirements Definition	Business Users, Business Analyst, Big Data Architect	Requirements Summary
3	Development	Develop Big Data Solution Framework	Big Data Architect	Solution Component Diagram
4	Development	Develop the Development and Test Environment	Big Data Architect, Data Integration Specialist, Developer	Development and Test Environment
5	Development	Develop Data Capture	Data Integration Specialist, Developer	Data Capturing Module
6	Development	Develop Analytics	Data Integration Specialist, Developer	Data Analytic Module
7	Development	Integrate Visualization	Data Integration Specialist, Developer	User Interface and Visualization Results
8	Development	Manage Assets and Configurations	Project Manager, Big Data Architect, Corporate Information Security, Head of IT Operation	Assets Inventory and Configurations Change Procedure

Agile Methodology may apply in Big Data Development Phase.

Key Activities, People and Deliverables

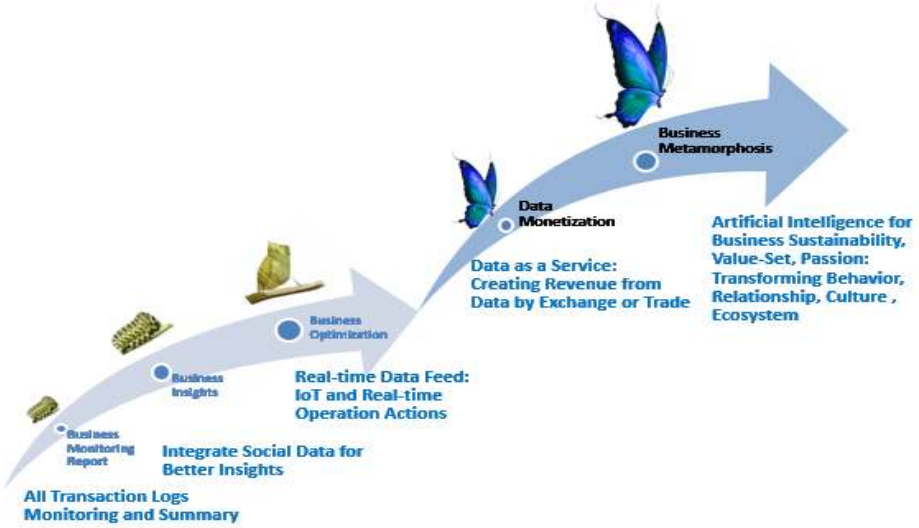
No.	Phases	Activities	People	Deliverables
1	Operation and Support	Monitor Big Data Platform Availability, Utilization and Capacity Planning	IT Operation Team	Availability, Utilization and Capacity Planning Report
2	Operation and Support	Manage Operation Tasks (Admin. Scripts, Commands), Data Capturing System, Upgrading or Patching	IT Operation Team, Big Data Architect	Schedule or Ad-Hoc Operation Activities
3	Operation and Support	Manage Service Requests and Incidents	IT Operation Team	Service Requests and Incidents Procedures
4	Operation and Support	System Administration Training	Big Data Architect, Data Integration Specialist, Developer, IT Administration, IT Operation	System Administration and Operation Training Activity
5	Operation and Support	Helpdesk Training	IT Administration, IT Operation, IT Helpdesk	Helpdesk Training Activity
6	Operation and Support	End-User Training (Analytic Results)	Business Analyst, Business Users	End-User Training Activity

Key Activities, People and Deliverables

No.	Phases	Activities	People	Deliverables
1	Evaluation	Create Adoption Rates for each analytics Results	IT Operation	Percent of user adoption
2	Evaluation	Create No. of Missing Analytic Results	Big Data Program Committee	No. of Missing Analytics Report
3	Evaluation	Create No. of Missing Data Results	Big Data Program Committee	No. of Missing Data Report
4	Evaluation	Create Lost hours per month Results	Big Data Architect, Data Scientist, Data Integration Specialist	Lost hours per month Report
5	Evaluation	Create Avg. of each Analytic Processing and Response Time Results	Data Integration Specialist, Data Scientist	Analytic Processing and Response Time Report
6	Evaluation	Create No. of Technology System Failure per month Results	Business Analyst, Big Data Architect, Data Scientist, Developer	Technology System Failure per month Report
7	Evaluation	Evaluate Activity Conformance with Policies	Big Data Architect, Project Manager	Change Control Log Report

Big Data Governance Worksheet

No.	Master/ Transactional/ Summary Data	Data Name	Owner	Used by Critical Business Service (Y/N)	Volume (MB/GB/TB)	Varieties of Types (Text, XML, JSON, Image, Sound, VDO, etc.)	Velocity (How fast data change in minutes)	Change Control (Y/N), Change Procedure	Current Issues



Big Data Maturity Model

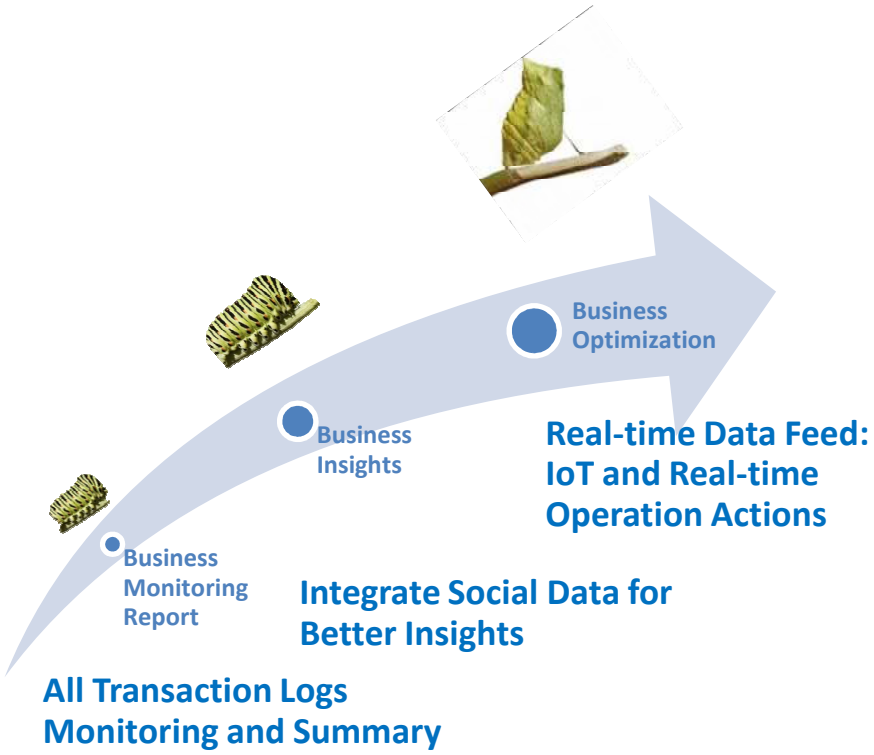
อ.ดนัยรัฐ ธนบดีธรรมจารี

Line ID: Danairat

FB: Danairat Thanabodithammachari

+668-1559-1446

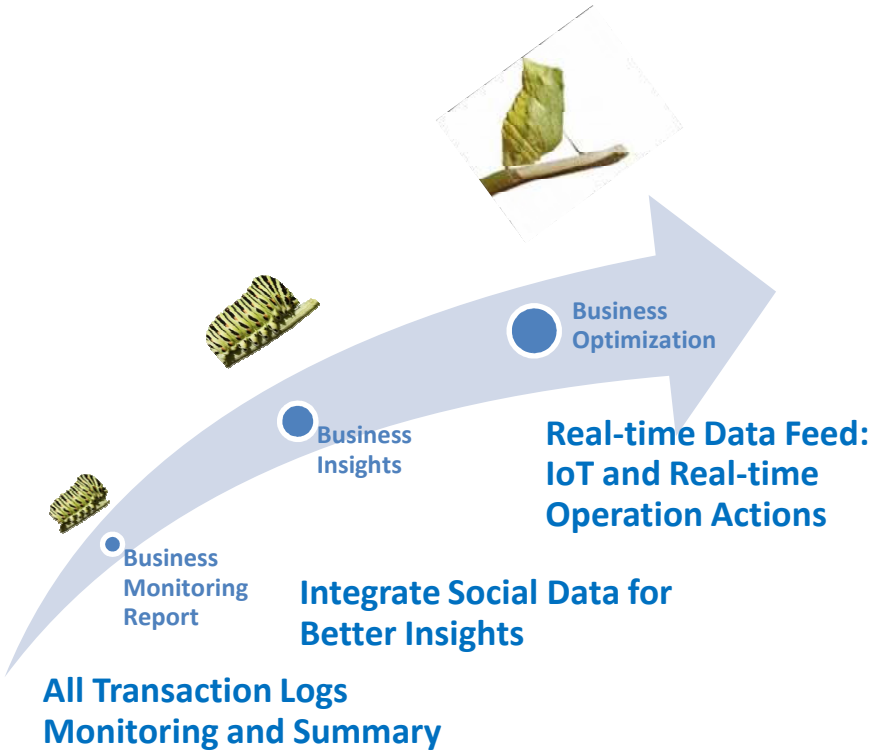
Big Data Maturity Model



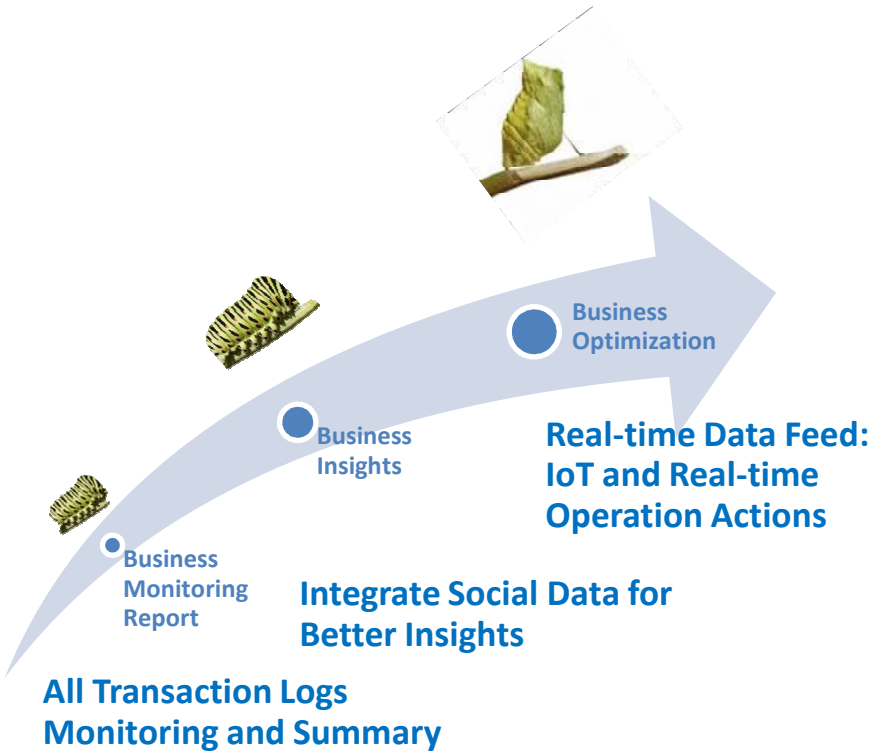
Big Data Maturity Model

Business Monitoring Report

Mine all the transactional data at the lowest levels of detail



Big Data Maturity Model



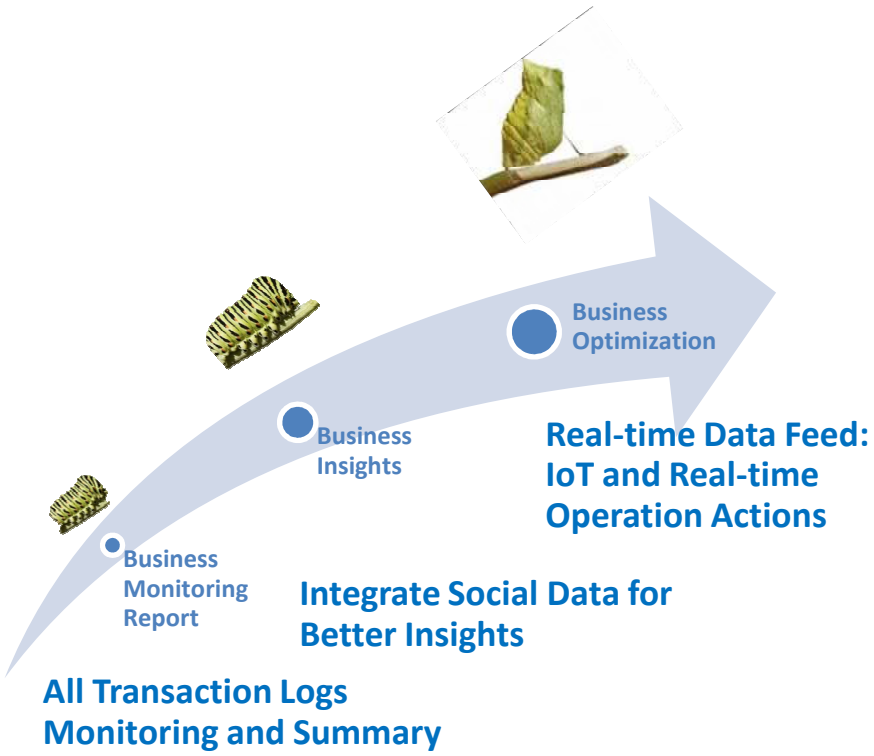
Business Monitoring Report

Mine all the transactional data at the lowest levels of detail

Business Insights

Integrate unstructured data with detailed structured (transactional) data to provide new metrics and new dimensions against which to monitor and optimize key business processes.

Big Data Maturity Model



Business Monitoring Report

Mine all the transactional data at the lowest levels of detail

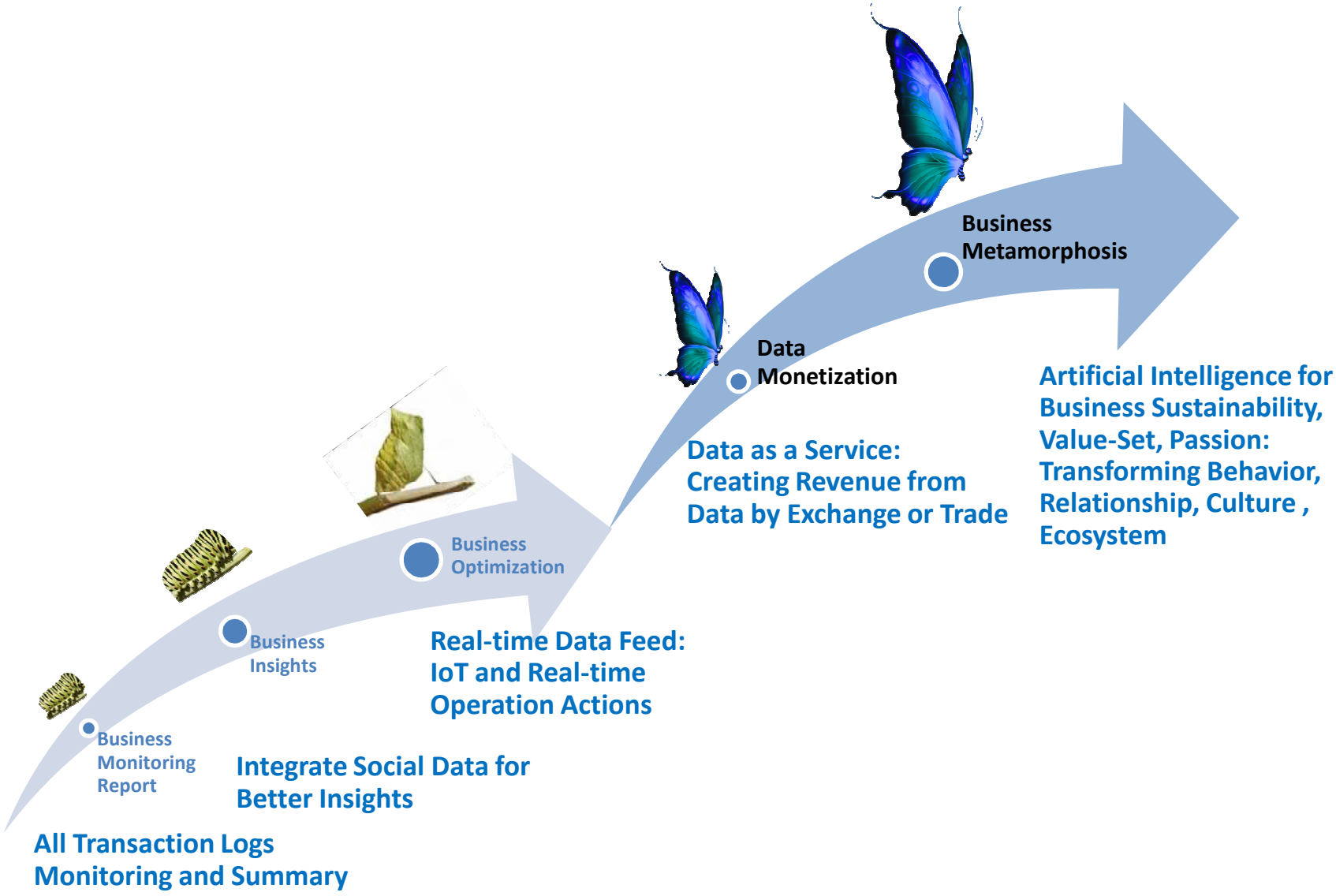
Business Insights

Integrate unstructured data with detailed structured (transactional) data to provide new metrics and new dimensions against which to monitor and optimize key business processes.

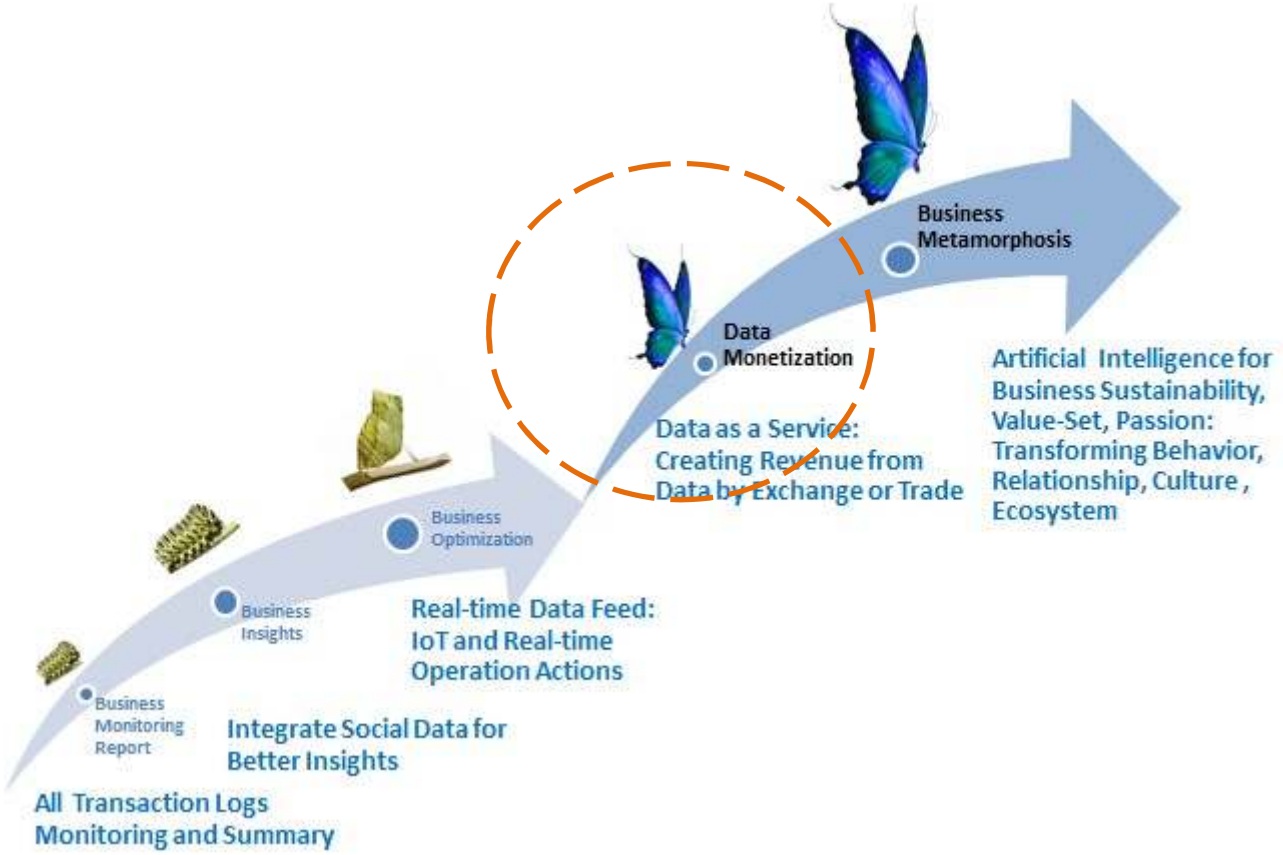
Business Optimization

Leverage real-time (or low-latency) data feeds to accelerate the organization's ability to identify and act upon business and market opportunities in a timely manner.

Big Data Maturity Model

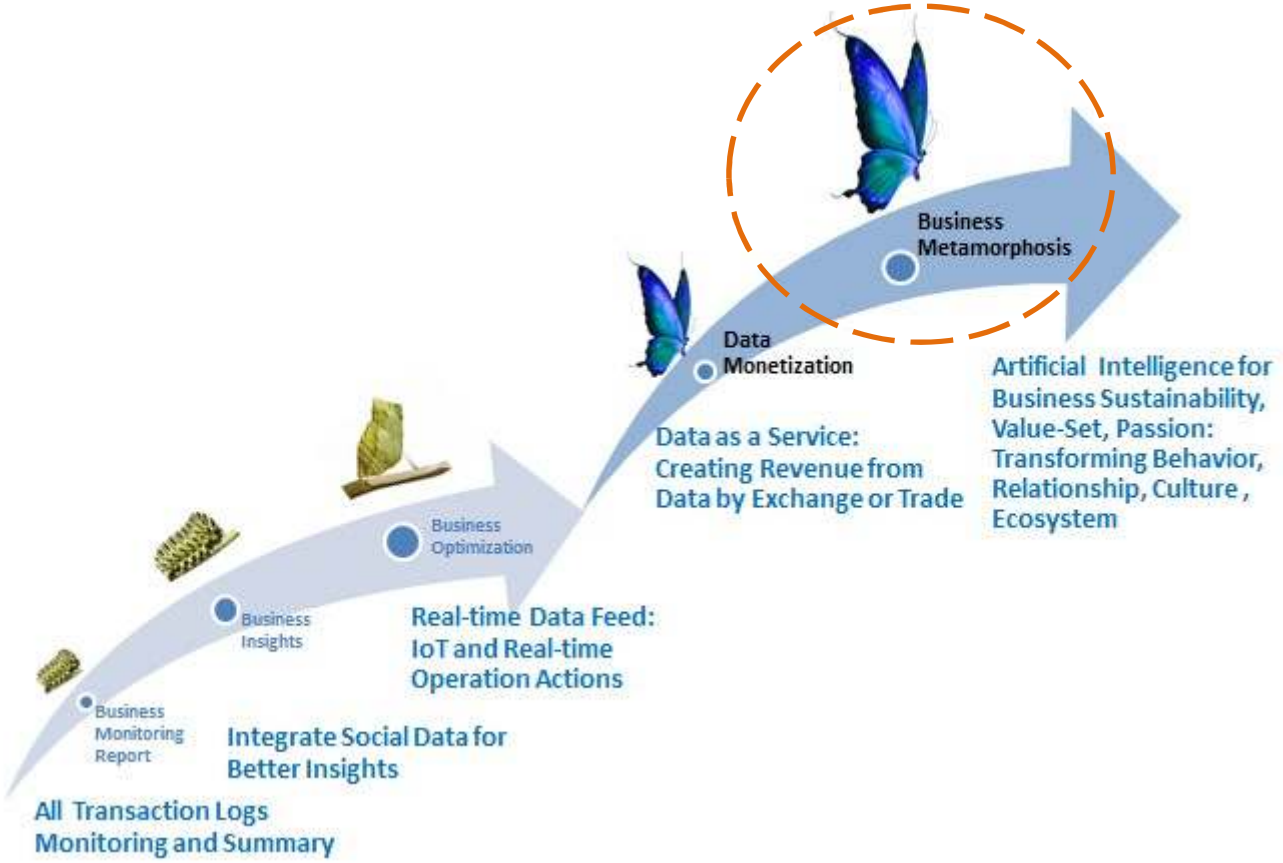


Big Data Maturity Model



Integrate predictive analytics into your key business processes to uncover insights buried in the massive volumes of detailed structured and unstructured data. (Note: having business users slice and dice the data to uncover insights worked fine when dealing with gigabytes of data, but doesn't work when dealing with terabytes and petabytes of data.)

Big Data Maturity Model



Driving new business models, new processes, more meaningful business interactions, innovation, improved and faster decision making, and a more agile organization

A digital ecosystem is a business community of organizations and individuals transacting across a distributed, adaptive, open, social, technical system with collaboration, transparency, constant evolution, self-organization, scalability and sustainability.

Thank you.

Together we can!

อ.ดนัยรัฐ ธนบดีธรรมจารี
+668-1559-1446 Line ID: danairat
FB: https://www.facebook.com/tdanairat