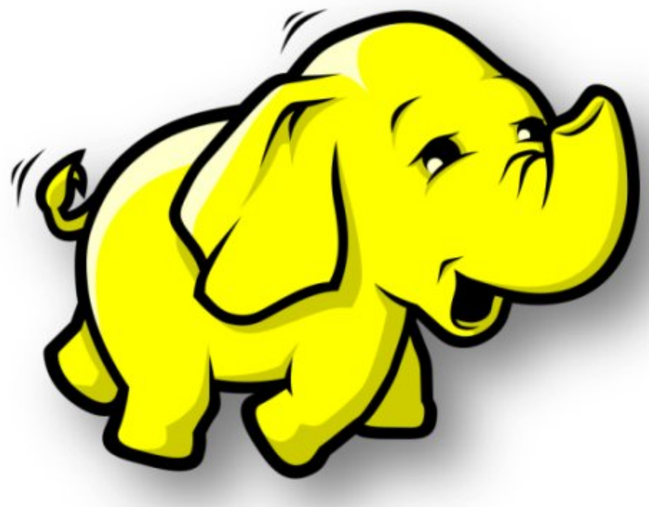
Chapter 2:

Hadoop Fundamentals



Lebanese University
Faculty of Information 1
hussein.hazimeh@ul.edu.lb
Dr. Hussein Hazimeh

Introduction

- Why to become a Big Data scientist?
  - Open domain (companies in USA ask for about 15 millions of jobs)
  - All skills required are available to learn
  - High salaries (about $100,000)

Big Data Challenges

- The major challenges associated with big data are:
  - Capturing data
  - Curation
  - Storage
  - Searching
  - Sharing
  - Transfer
  - Analysis
  - Presentation

Facts About Hadoop

- 2009: Hadoop sorts 1TB of data in 209 seconds
- 2009: Google sorts 1TB of data in 69 seconds
- 2010: Yahoo uses Hadoop to sort 1TB in 62 seconds
- 2017: A team uses Hadoop to sort 100TB in 1400 seconds (4TB/minute). They uses a cluster of 207 machines implemented on Spark.
- …

What is Apache Hadoop?

- **Hadoop** is an Apache open source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models.

- The **Hadoop** framework application works in an environment that provides distributed storage and computation across clusters of computers.

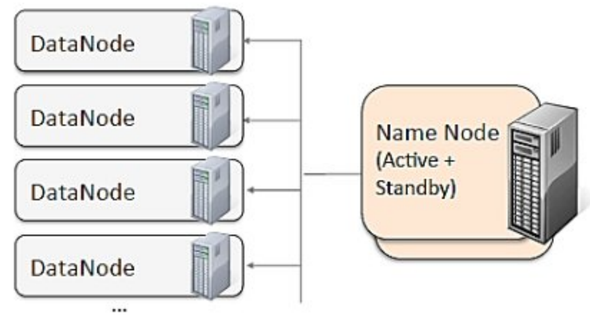
- **Hadoop** is designed to scale up from single server to thousands of machines, each offering local computation and storage.

Facts about Apache Hadoop

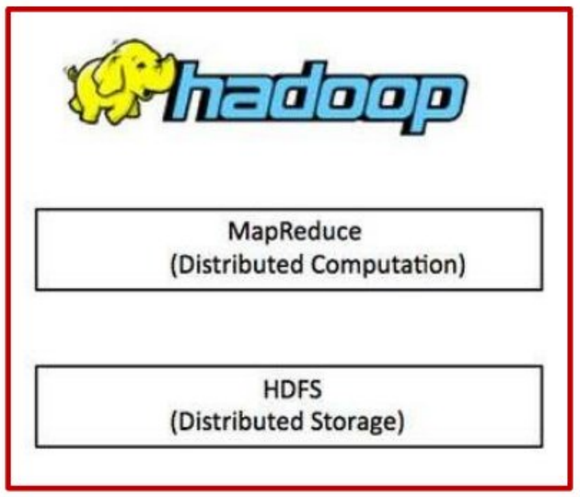
- Around 60 committers from ~10 companies
  - Cloudera, Yahoo!, Facebook, Apple, and more
- Hundreds of contributors writing features, fixing bugs
- Many related projects, applications, tools, etc.

Hadoop Components

- **Hadoop cluster:** master node (namenode) + worker nodes (slave nodes, data nodes)
- **Hadoop** core has two main components:
  - Hadoop Distributed File System (HDFS): Stores data on the cluster
  - MapReduce: Process data on the cluster

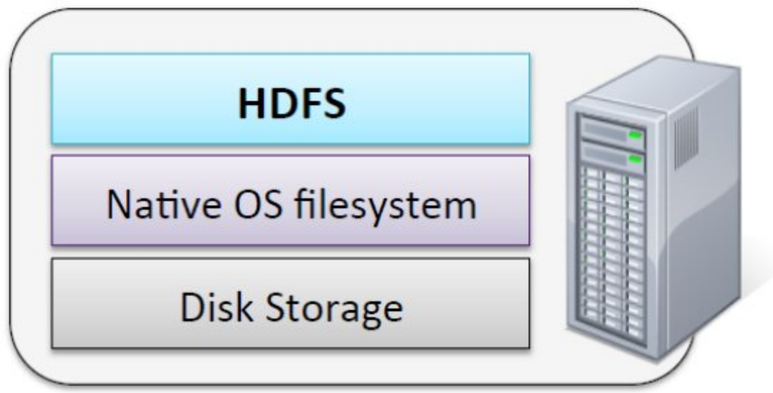


- **Other projects around Hadoop:**
  - Referred to as the 'Hadoop Ecosystem'
  - Pig, Hive, HBase, Flume, Oozie, Sqoop, etc

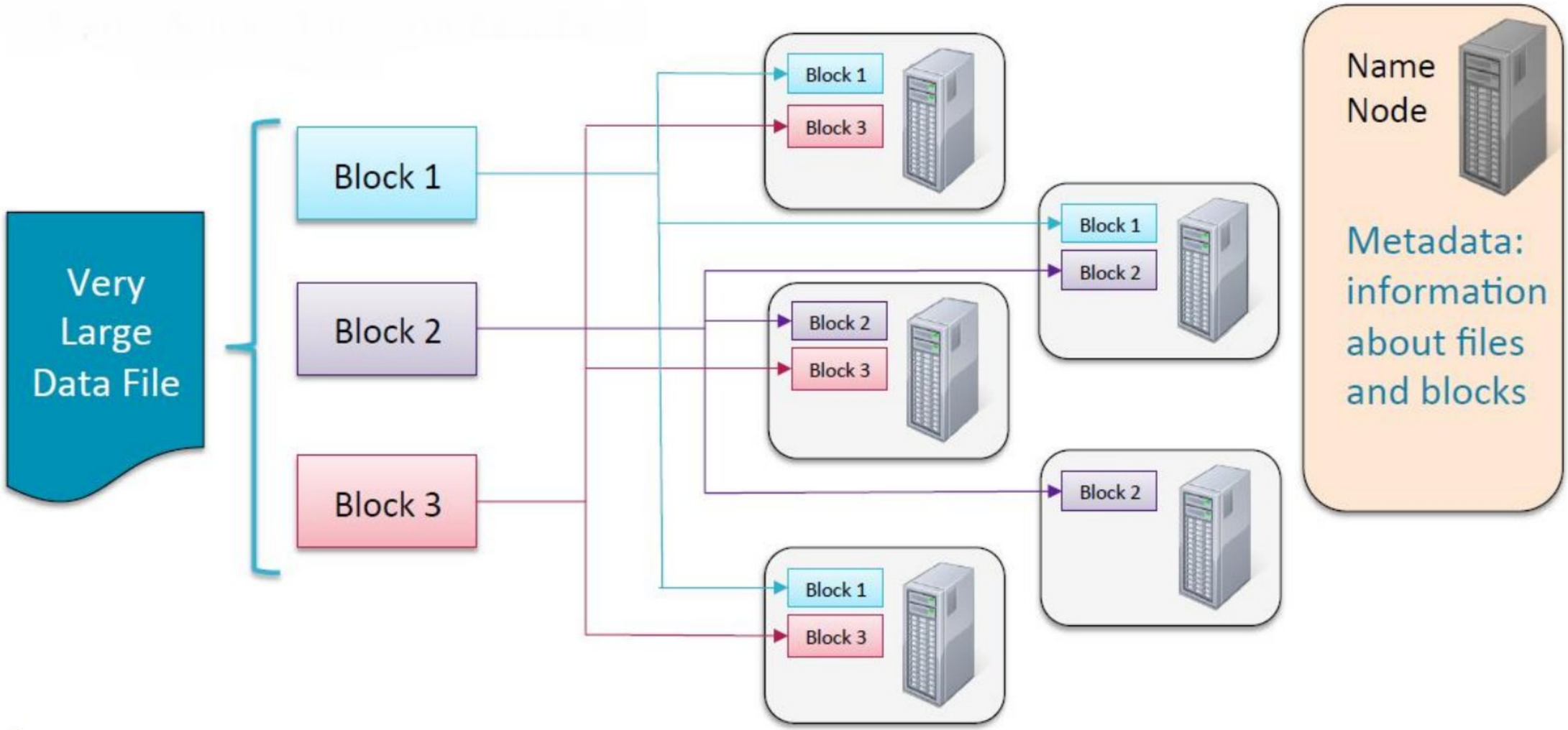


HDFS

- **HDFS** is based on the Google File System (GFS)
- Responsible for storing data on the cluster
- 3 characteristics: storing big data (petabytes), data streaming access (write once, read many), run in commidity hardware.
- Files are split into blocks (64MB or 128 MB)
- Data is distributed across many machines
- Each block is replicated multiple times (3 by default)
- A master node called the NameNode & keeps track of which blocks make up a file, and where those blocks are located
  - Known as the metadata



HDFS

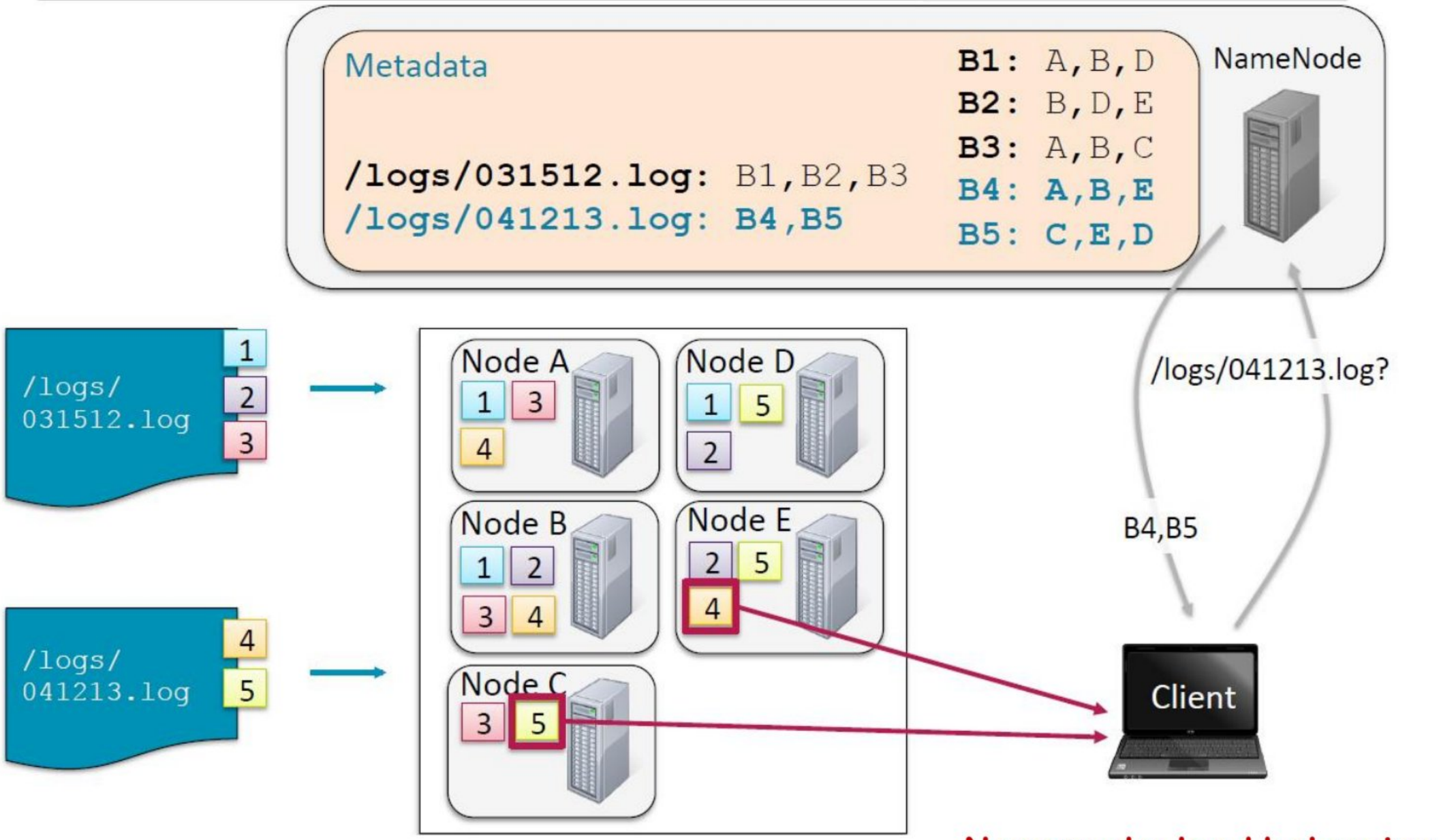


- Blocks division advantages: fits limited resources of commidity hardware, regenerate block when a node fails.
- Small data files => more metadata => HDFS becomes inefficient
- Store file of size 129 Mb?

Accessing HDFS

- 3 methods:
  - Fs Shell Command line: hadoop fs
  - Java API
  - Ecosystem Projects

Storing and Retrieving Files



Namenode: load balancing of client requests

HDFS NameNode Availability

- If the NameNode stops, the cluster becomes inaccessible
- Solution 1: backup
  - Problem: loss data between two backups
- Solution 2: Standby Name node that updates periodically metadata of Active namenode
  - Problem: loss data between two updates
- Solution 3: Secondary name node that update continuously metadata of active name node
  - Shared edit log between Secondary and active name nodes
  - Fail over controller: translation between active and secondary name node
  - Fencing: how is working? Secondary or active name nodes

MapReduce

- **MapReduce** is a method for distributing a task across multiple nodes
- Each node processes data stored on that node
- Consists of two phases:
  - Map
  - Reduce
- Terminology
  - **Job:** A complete execution of Mappers and Reducers over a dataset (application in MapReduce 2)
  - **Task:** is the execution of a single Mapper or Reducer over a slice of data

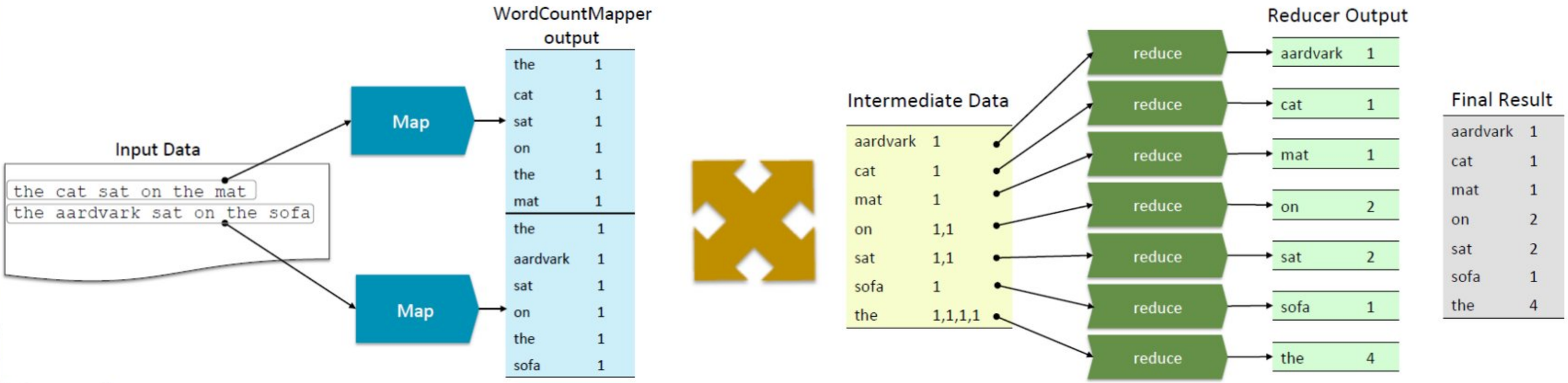
MapReduce

- **The Mapper**
  - Each Map task (typically) operates on a single HDFS block
  - Map tasks (usually) run on the node where the block is stored

- **Shuffle and Sort**
  - Sorts and consolidates intermediate data from all mappers
  - Happens as Map tasks complete and before Reduce tasks start

- **The Reducer**
  - Operates on shuffled/sorted intermediate data (Map task output)
  - Produces final output

Example: Word Count



Example: Analyzing Log Data

Input Data

```
...
2013-03-15 12:39 - 74.125.226.230 /common/logo.gif 1231ms - 2326
2013-03-15 12:39 - 157.166.255.18 /catalog/cat1.html 891ms - 1211
2013-03-15 12:40 - 65.50.196.141 /common/logo.gif 1992ms - 1198
2013-03-15 12:41 - 64.69.4.150 /common/promoex.jpg 3992ms - 2326
...
```

Map

FileTypeMapper output

...	...
gif	1231
html	891
gif	1992
jpg	3992
html	788
gif	3997
...	...

Intermediate Data after Shuffle and Sort

html	891,788,344,2990...
gif	1231,1992,3997,872...
jpg	3992,7881,2999...
png	919,890,3441,444...
txt	344,325,444,421...

Reduce

AverageReducer output

html	888.6
gif	1886.4
jpg	888.6
png	1201.0
txt	399.1

Hadoop: Environment Setup

- Hadoop is supported by GNU/Linux platform.
- We have to install a Linux operating system for setting up Hadoop environment.
  - If any, use a Virtualbox software
- Hadoop Operation Modes: 3 modes
  - Local/Standalone Mode: After downloading Hadoop in your system, by default, it is configured in a standalone mode and can be run as a single java process.
  - Pseudo Distributed Mode: It is a distributed simulation on single machine. Each Hadoop daemon such as hdfs, yarn, MapReduce etc., will run as a separate java process. This mode is useful for development.
  - Fully Distributed Mode: This mode is fully distributed with minimum two or more machines as a cluster.

Hadoop: Environment Setup

- **Steps to install Hadoop:**

1) Create a user
2) SSH Setup and Key Generation
3) Installing Java
4) Downloading Hadoop
5) Hadoop Configuration

   - **core-site.xml:** contains information about the port number used for Hadoop instance, memory allocated for the file system, memory limit for storing the data, etc.
   - **hdfs-site.xml:** contains information such as the value of replication data, namenode path, and datanode paths of your local file systems.

Hadoop: Environment Setup

- **core-site.xml**

```
<configuration>
   <property>
      <name>fs.default.name</name>
      <value>hdfs://localhost:9000</value>
   </property>
</configuration>
```

- **hdfs-site.xml**

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

   <property>
      <name>dfs.name.dir</name>
      <value>file:///home/hadoop/hadoopinfra/hdfs/namenode </value>
   </property>

   <property>
      <name>dfs.data.dir</name>
      <value>file:///home/hadoop/hadoopinfra/hdfs/datanode </value>
   </property>
</configuration>
```

Hadoop: Environment Setup

- **Steps to install Hadoop: cont.**

  6) **Verifying Hadoop Installation**

  - **Verifying Name Node Setup**

    $ hdfs namenode -format

  - **Verifying Hadoop dfs**

    $ start-dfs.sh

  - **Verifying Yarn Script**

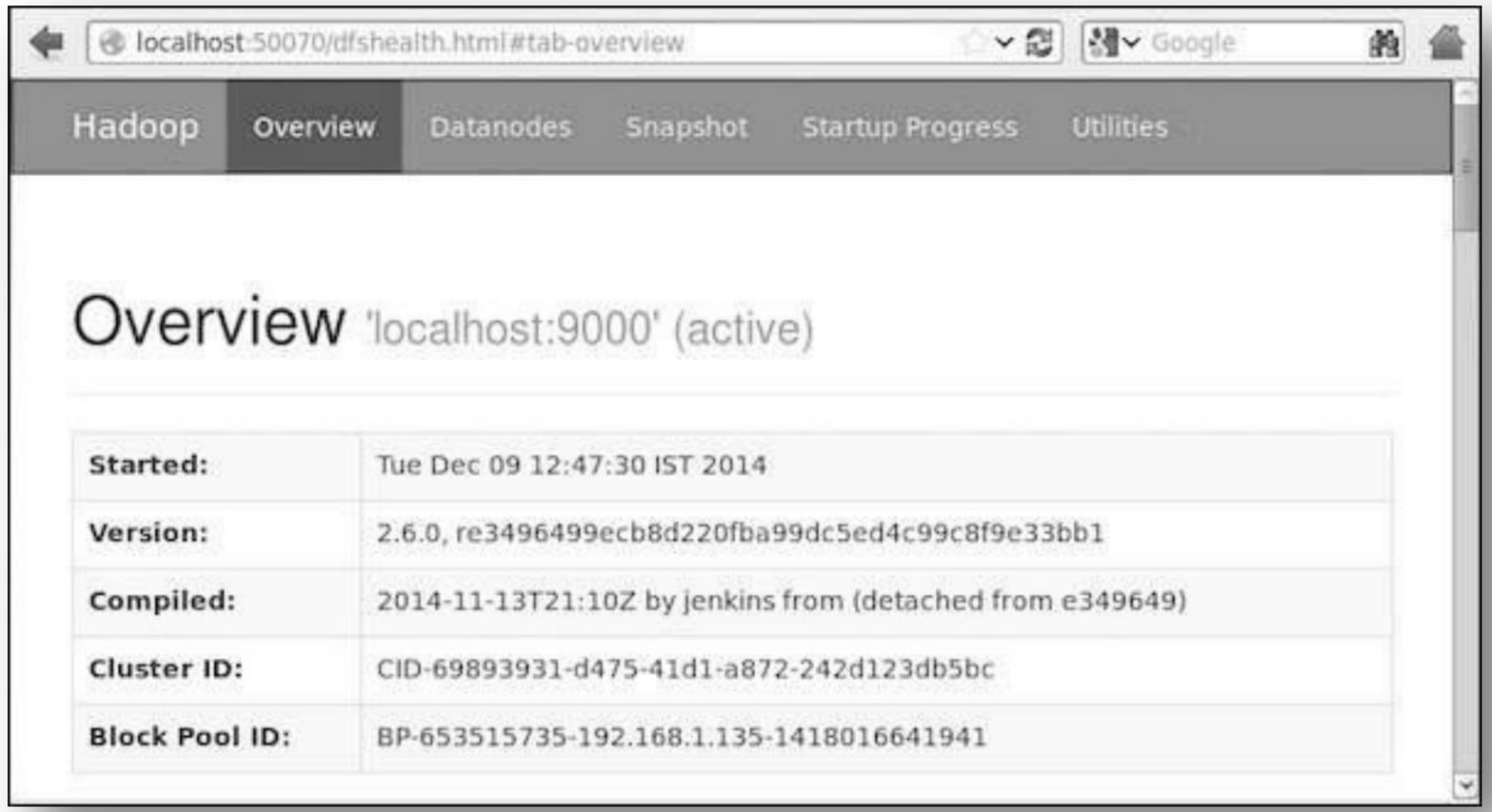
    $ start-yarn.sh

Hadoop: Environment Setup

- **Steps to install Hadoop: cont.**

  7) **Accessing Hadoop on Browser**

  http://localhost:50070/

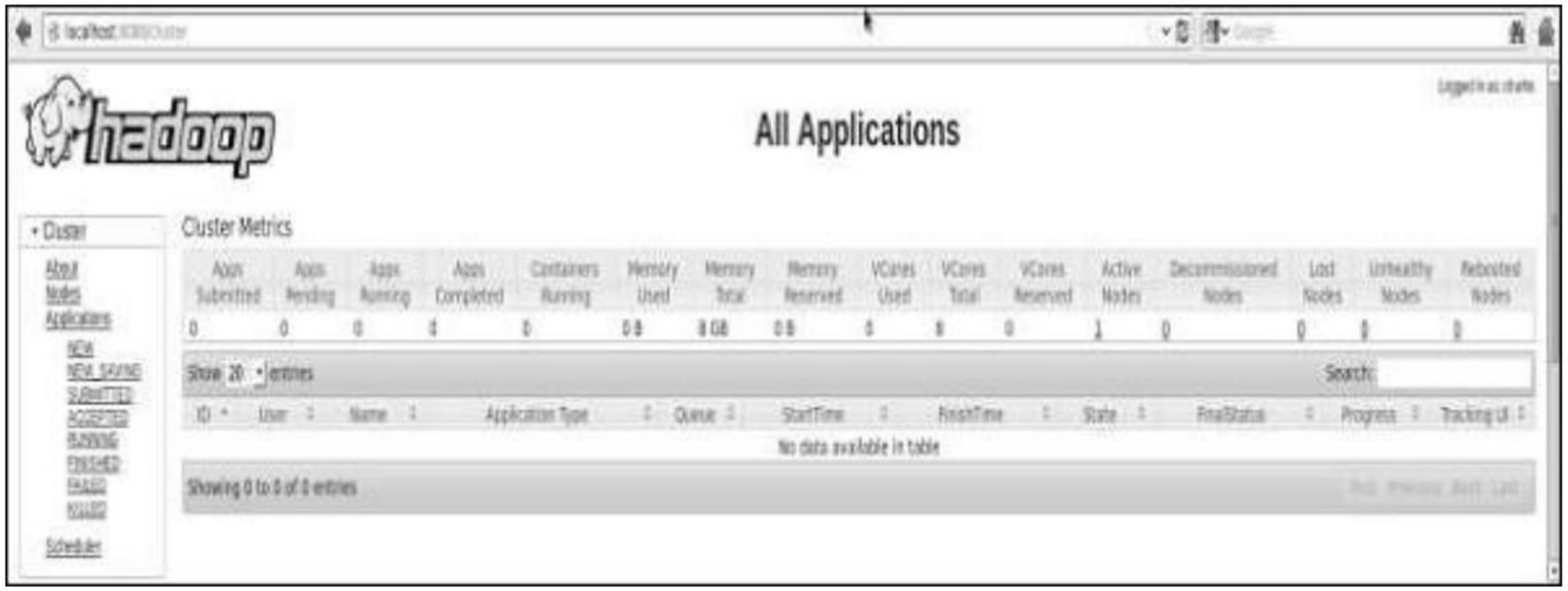


Hadoop: Environment Setup

- **Steps to install Hadoop: cont.**

  8) **Verify All Applications for Cluster**

  http://localhost:8088/



HDFS Operations

- **Starting HDFS**
  - **Format the namenode**

    $ hadoop namenode -format
  - **Start the dfs (namenode and data nodes)**

    $ start-dfs.sh
- **Listing Files in HDFS**

  $ $HADOOP_HOME/bin/hadoop fs -ls <args>
- **Inserting Data into HDFS**
  - **Create an input directory**

    $ $HADOOP_HOME/bin/hadoop fs -mkdir /user/input
  - **Transfer and store a data file from local systems to the Hadoop file system**

    $ $HADOOP_HOME/bin/hadoop fs -put /home/file.txt /user/input
  - **Verify the file**

    $ $HADOOP_HOME/bin/hadoop fs -ls /user/input

HDFS Operations

- **Retrieving Data from HDFS (Assume we have a file in HDFS called outfile)**
  - **View the data from HDFS**

    $ $HADOOP_HOME/bin/hadoop fs -cat /user/output/outfile

  - **Get the file from HDFS to the local file system**

    $ $HADOOP_HOME/bin/hadoop fs -get /user/output/ /home/hadoop_tp/

- **Shutting Down the HDFS**

  $ stop-dfs.sh

Hadoop Example

- Copy file foo.txt from local disk to the user's directory in HDFS
  - $ hadoop fs -put foo.txt foo.txt

  This"will"copy"the"file"to"/user/username/foo.txt
- Get a directory listing of the user's home directory in HDFS
  - $ hadoop fs –ls
- Get a directory listing of the HDFS root directory
  - $ hadoop fs –ls /
- Display the contents of the HDFS file /user/fred/bar.txt
  - $ hadoop fs -cat /user/fred/bar.txt
- Copy that file to the local disk, named as baz.txt
  - hadoop fs -get /user/fred/bar.txt baz.txt

Hadoop Example

- Create a directory called input under the user's home directory
  - $ hadoop fs -mkdir input
- Delete the directory input_old and all its contents
  - $ hadoop fs -rm -r input_old

Hadoop: Command Reference

- **-ls <path>**

  Lists the contents of the directory specified by path, showing the names, permissions, owner, size and modification date for each entry.

- **-du <path>**

  Shows disk usage, in bytes, for all the files which match path; filenames are reported with the full HDFS protocol prefix.

- **-mv <src><dest>**

  Moves the file or directory indicated by src to dest, within HDFS.

- **-cp <src> <dest>**

  Copies the file or directory identified by src to dest, within HDFS.

- **-rm <path>**

  Removes the file or empty directory identified by path.

- **-put <localSrc> <dest>**

  Copies the file or directory from the local file system identified by localSrc to dest within the DFS.

- **-cat <filen-ame>**

  Displays the contents of filename on stdout.

Hadoop: Command Reference

- **-mkdir <path>**

  Creates a directory named path in HDFS.

- **-stat [format] <path>**

  Prints information about path. Format is a string which accepts file size in blocks (%b), filename (%n), block size (%o), replication (%r), and modification date (%y, %Y).

- **-help <cmd-name>**

  Returns usage information for one of the commands listed above. You must omit the leading '-' character in cmd.

Reading Data from a Hadoop Using Java

- Displaying files from a Hadoop filesystem using a *URLStreamHandler*

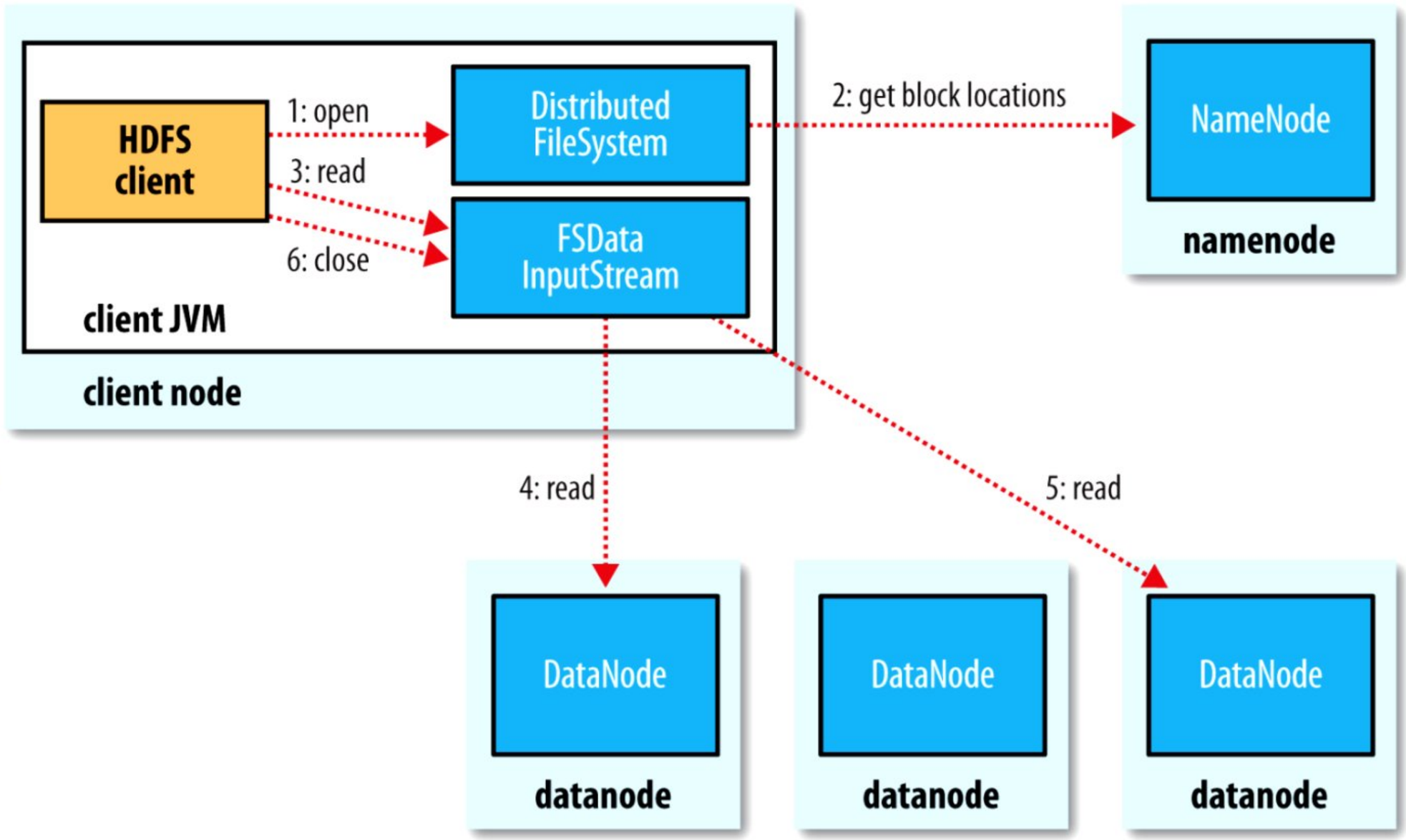
```
public class URLCat {

  static {
    URL.setURLStreamHandlerFactory(new FsUrlStreamHandlerFactory());
  }

  public static void main(String[] args) throws Exception {
    InputStream in = null;
    try {
      in = new URL(args[0]).openStream();
      IOUtils.copyBytes(in, System.out, 4096, false);
    } finally {
      IOUtils.closeStream(in);
    }
  }
}
```

Writing Data to Hadoop Using Java

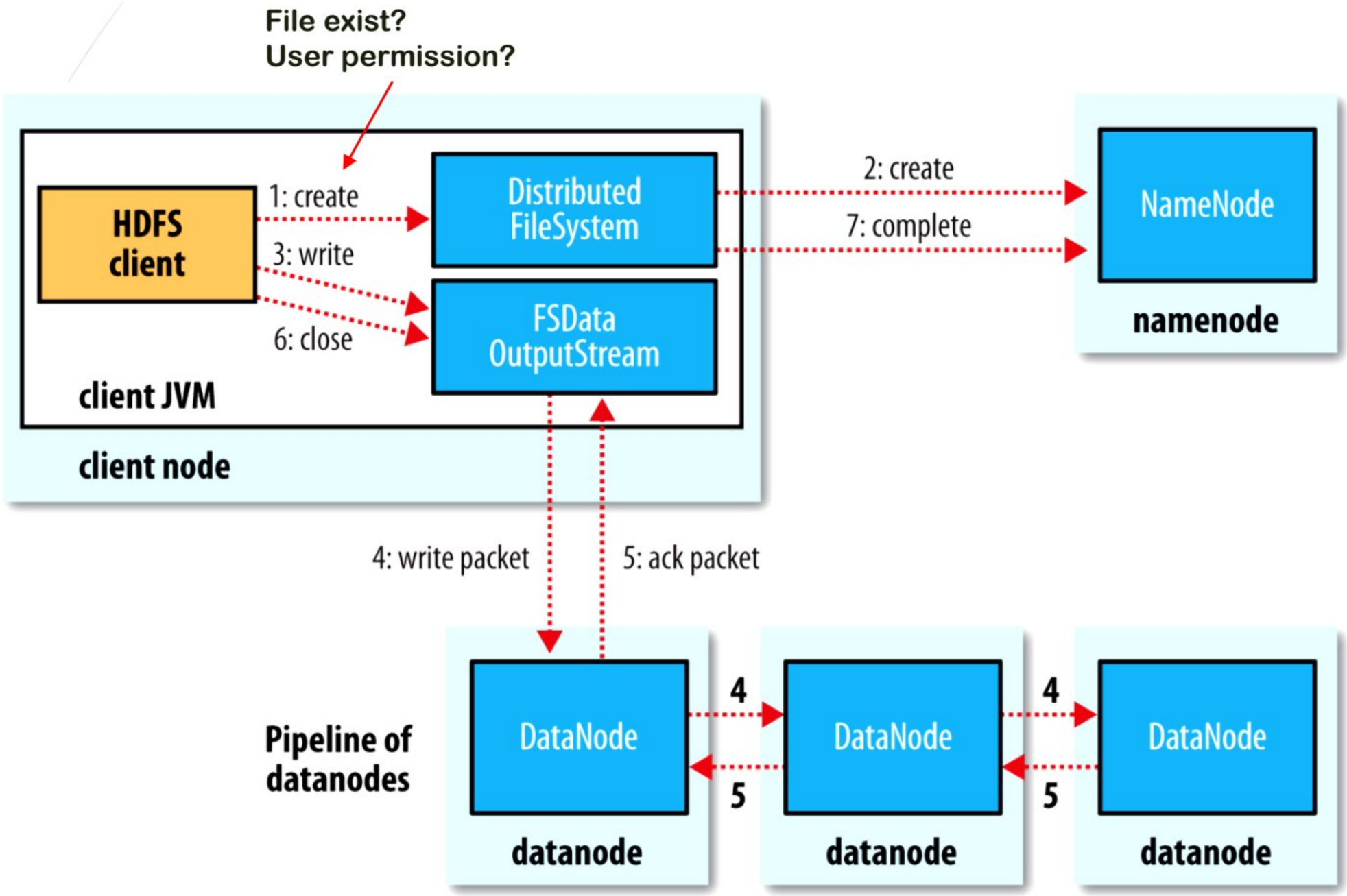
- Copying a local file to a Hadoop filesystem

```
public class FileCopyWithProgress {
  public static void main(String[] args) throws Exception {
    String localSrc = args[0];
    String dst = args[1];

    InputStream in = new BufferedInputStream(new FileInputStream(localSrc));

    Configuration conf = new Configuration();
    FileSystem fs = FileSystem.get(URI.create(dst), conf);
    OutputStream out = fs.create(new Path(dst), new Progressable() {
      public void progress() {
        System.out.print(".");
      }
    });

    IOUtils.copyBytes(in, out, 4096, true);
  }
}
```

Data Flow: File Read



Data Flow: File Write



How namenode choose datanode? Reliability and read/write bandwidth