Lecture 02

Introduction to MapReduce

Csci E63 Big Data Analytics

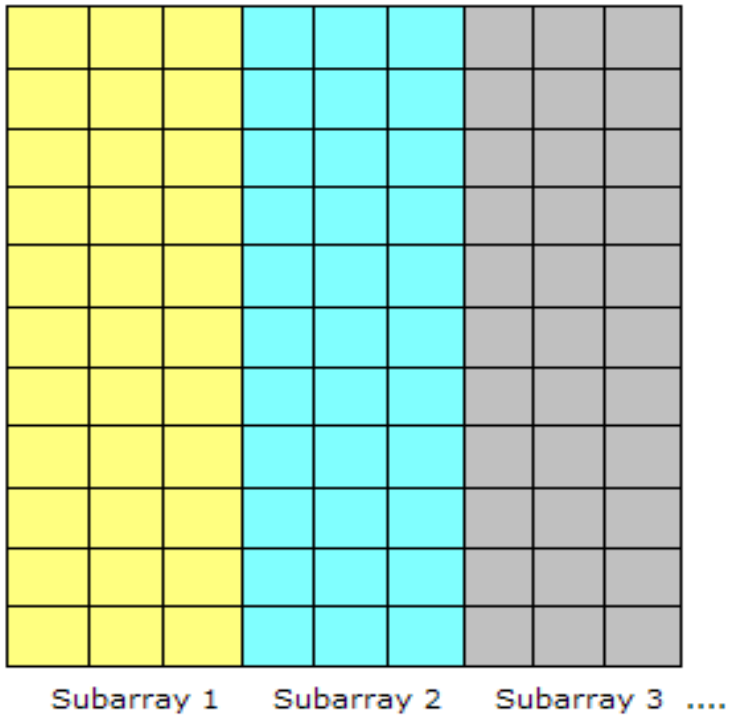
Zoran B. Djordjević

`Serial vs. Parallel` Programming Model

- Many or most of our programs are S*erial*.
  - A `Serial Program` consists of a sequence of instructions, where each instruction executes one after the other.
  - Serial programs run from start to finish on a single processor.
- *Parallel programming* developed as a means of improving performance and efficiency.
  - In a `Parallel Program`, the processing is broken up into parts, each of which could be executed concurrently on a different processor. Parallel programs could be faster.
  - Parallel Programs could also be used to solve problems involving large datasets and non-local resources.
  - Parallel Programs are usually ran on a set of computers connected on a network (a pool of CPUs), with an ability to read and write large files supported by a distributed file system.

Common Situation

- A common situation involves processing of a large amount of consistent data.
- If the data could be decomposed into equal-size partitions, we could devise a parallel solution. Consider a huge array which can be broken up into sub-arrays



If the same processing is required for each array element, with no dependencies in the computations, and no communication required between tasks, we have an ideal parallel computing opportunity, the so called *Embarrassingly Parallel problem.*
A common implementation of this approach is a technique called *Master/Worker*.

MapReduce Programming Model

- `MapReduce` programming model is derived as a technique for solving embarrassingly and not-so-embarrassingly parallel problems.
- The idea stems from the `map` and `reduce` combinators in Lisp programming language.
- In Lisp, a `map` takes as input a function and a sequence of values. It then applies the function to each value in the sequence. A `reduce` combines all the elements of a sequence using a binary operation. For example, it can use "+" to add up all the elements in the sequence.
- `MapReduce` was developed within Google as a mechanism for processing large amounts of raw data, for example, crawled documents or web request logs.

http://static.googleusercontent.com/media/research.google.com/en//archive/mapreduce-osdi04.pdf

- Google data is so large, it must be distributed across tens of thousands of machines in order to be processed in a reasonable time.
- The distribution implies parallel computing since the same computations are performed on each CPU, but with a different portion of data.

MapReduce Library

- `map` function, written by a user of the MapReduce library, takes `an input key/value` pair and produces `a set of intermediate key/value` pairs.
- The MapReduce library groups together all intermediate values associated with the same intermediate key and passes them to the `reduce` function.
- The `reduce` function, also written by the user, accepts `an intermediate key` and `a set of values` for that key. The `reduce` function merges together these values to form a possibly smaller set of values.

Why Does Google Need Parallel Processing

- Google's search mechanisms rely on several matrices of sizes that are really big: 10^{10} X 10^{10}
- For example, Google's Page Rank algorithm, which determines the relevance of various Web pages is a process which determines the largest eigen values of such big matrices.
- Google needs to spread its processing on tens or hundreds of thousands of machines in order be able to rank the pages of World Wide Web in "real time".
- Today we will just indicate some features of Google mechanisms.

Google Data Center Images

MapReduce Execution

- The `Map` invocations are distributed across multiple machines by automatically partitioning the input data into a set of M splits or ***shards*.**
- The input shards can be processed in parallel on different machines.
- `reduce` invocations are distributed by partitioning the intermediate key space into `r` pieces using a partitioning function (e.g., `hash(key) mod r`).
- The number of partitions (`r`) and the partitioning function are specified by the user.

Vocabulary and Number of Words in all Documents

- Consider the problem of counting the number of occurrences of each word in a large collection of documents

```
map(String documentName, String documentContent):
//key: document name, value: document content
for each word w in documentContent:
//key: word, value: number of occurances
  EmitIntermediate(w, wordCount);

reduce(String w, Iterator values):
// key: a word, // values: a list of counts
int result = 0;
for each v in values:
  result += v;
Emit(w, result));
```

- The map function emits each word plus an associated count of occurrences in a document.
- The reduce function sums all the counts for every word giving us the number of occurrances of each word in the entire set of documents.

Master and Workers

MapReduce algorithm is executed by two types of computer nodes: a MASTER and many WORKERS.

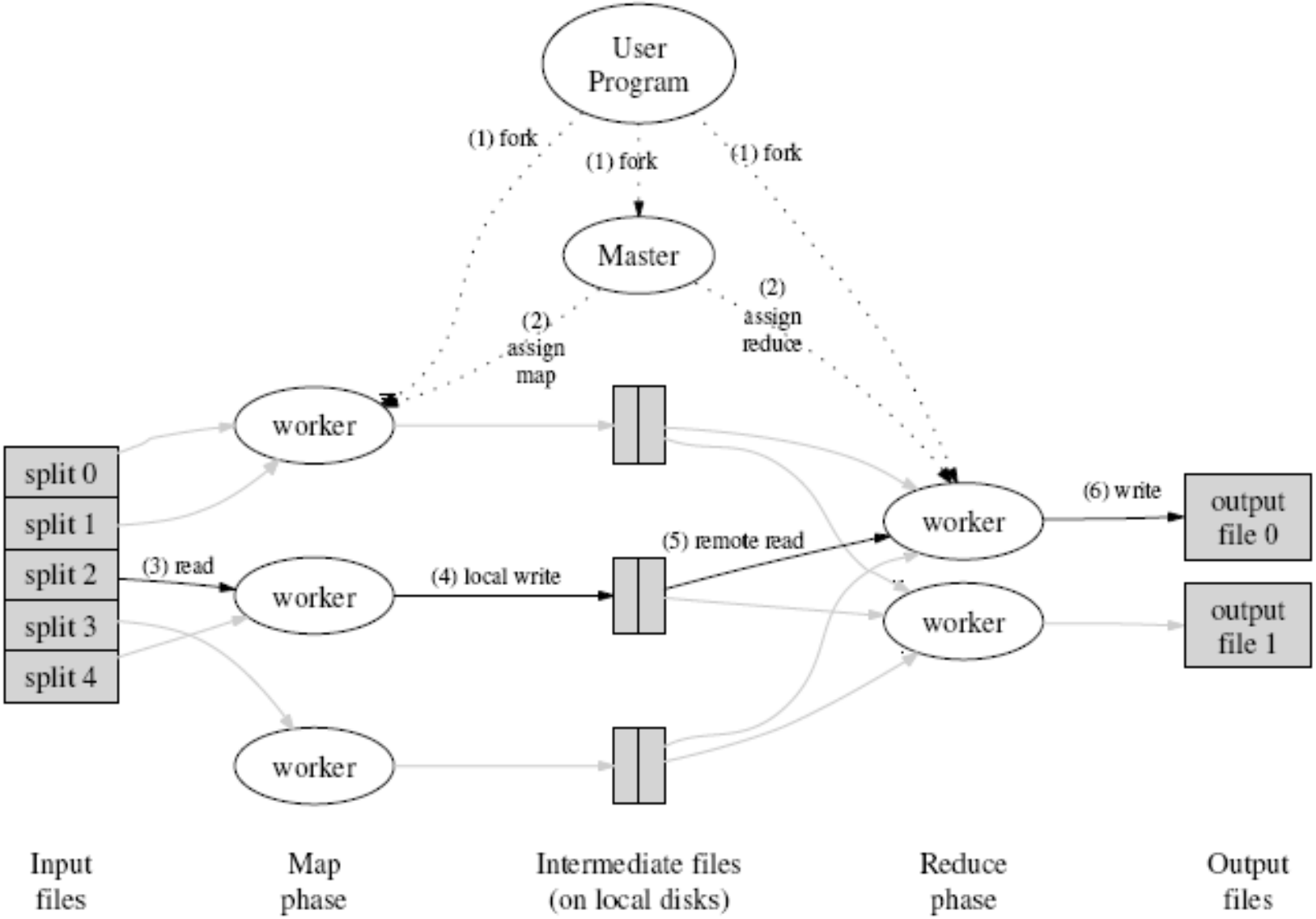
Role of the MASTER is to:

1. Initialize the data and splits it up according to the number of available WORKERS
2. Send each WORKER its portion of data.
3. Receive the intermediate results from each WORKER
4. Pass the intermediate results to other WORKERS
5. Collect results from the second tier WORKERS and perform some final calculations if needed.

Role of a WORKER is to:

- Receive portion of data from the MASTER
- Perform processing on the data
- Return results to the MASTER

Flow of Execution



MapReduce Steps

1. The MapReduce library in the user program first shards the input files into M pieces of typically 16 megabytes to 64 megabytes (MB) per piece. It then starts up many copies of the program on a cluster of machines.
2. One of the programs is special: the Master. The rest are workers are assigned work by the Master. There are M map tasks and R reduce tasks to assign. The master picks idle workers and assigns each one a map task or a reduce task.
3. A worker who is assigned a map task reads the contents of the corresponding input shard. It parses key/value pairs out of the input data and passes each pair to the user-defined Map function. The intermediate key/value pairs produced by the Map function are buffered in memory.
4. Periodically, the buffered pairs are written to local disk, partitioned into R regions by the partitioning function. The locations of these buffered pairs on the local disk are passed back to the master, who is responsible for forwarding these locations to the reduce workers.

MapReduce Steps

5. When a reduce worker is notified by the Master about these locations, it uses remote procedure calls to read the buffered data from the local disks of the map workers. When a reduce worker has read all intermediate data, it sorts it by the intermediate keys so that all occurrences of the same key are grouped together. If the amount of intermediate data is too large to fit in memory, an external sort is used.
6. The reduce worker iterates over the sorted intermediate data and for each unique intermediate key encountered, it passes the key and the corresponding set of intermediate values to the user's Reduce function. The output of the Reduce function is appended to a final output file for this reduce partition.
7. When all map tasks and reduce tasks have been completed, the master wakes up the user program. At this point, the MapReduce call in the user program returns back to the user code.

After successful completion, the output of the MapReduce execution is available in the R output files

Usage of MapReduce at Google

- **Distributed Grep:** The map function emits a line if it matches a given pattern. The reduce function is an identity function that just copies the supplied intermediate data to the output.
- **Count of URL Access Frequency**: The map function processes logs of web page requests and outputs <URL, 1>. The reduce function adds together all values for the same URL and emits a <URL, total count> pair.
- **Reverse Web-Link Graph**: The map function outputs <target, source> pairs for each link to a target URL found in a page named "source". The reduce function concatenates the list of all source URLs associated with a given target URL and emits the pair: <target, list(source)>.
- Most of the rest of Google functionality also uses Map Reduce functionality.

Open Source MapReduce

- We will not build MapReduce frameworks.
- We will learn to use an open source MapReduce Framework called Hadoop which is offered as Apache project, as well by commercial vendors: Cloudera, Hortonworks, MapR, IBM, and many other vendors.
- Non commercial version of Hadoop source code is available at *apache.org*.
- Each vendor improves on original, open source design and sells its improved version commercially.
- **Hadoop makes massive parallel computing on large clusters (thousands of cheap, commodity machines) possible.**

Challenges

Most pronounced challenges a MapReduce framework addresses on our behalf are:

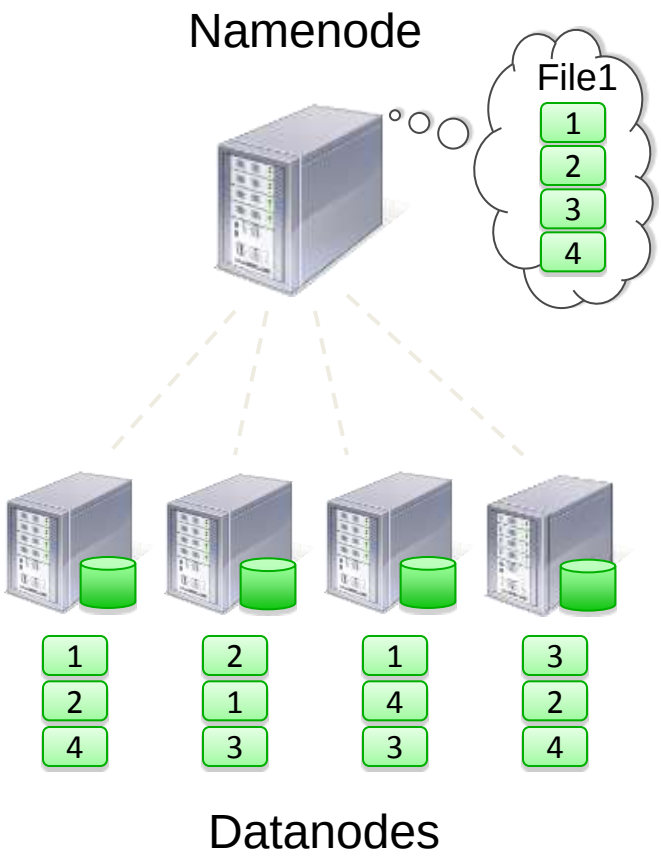
1. Cheap nodes fail, especially if you have many
   - Mean time between failures for 1 disk = 3 years
   - Mean time between failures for 1000 disks = 1 day
   - Solution: Build fault-tolerance into the system
2. Commodity network = low bandwidth
   - Solution: Push computation to the data
3. Programming distributed systems is hard
   - Solution: Distributed data-parallel programming model:
     - users write “map” & “reduce” functions.
     - MapReduce framework (system) distributes work and handles the faults.

Major Hadoop Components

- **Hadoop Common**: The common utilities that support the other Hadoop modules.
- **Hadoop Distributed file system** (HDFS)
  - Single namespace for entire cluster
  - Replicates data 3x for fault-tolerance
  - Allows writes, deletes and appends. Does not allow updates of data blocks. (Note, commercial implementations of Hadoop typically overcome this no-updates limitation)
- **Hadoop YARN**: A framework for job scheduling and cluster resource management.
- **Hadoop MapReduce**: A YARN-based system for parallel processing of large data sets.

Hadoop Distributed File System

- Files split into 64MB-128MB *blocks*
- Each block is replicated across several *datanodes* (usually 3)
- *Name node* stores metadata (file names, block locations, etc.)
- Optimized for large files, sequential reads
- Files are typically append-only



MapReduce Programming Model, Again

- Data types: key-value *records (Keys and Values could be of different data types: ints, dates, strings, etc.)*
- Map function:

$$(K_{in}, V_{in}) \rightarrow list(K_{inter}, V_{inter})$$

- Intermediate keys do not have to be related to the initial keys in any way.
- Reduce function is fed sorted collection of intermediate values for each intermediate key.

$$(K_{inter}, list(V_{inter})) \rightarrow list(K_{out}, V_{out})$$

- Reduce function transforms that collection into a final result, a list of key-value pairs.

Example: Word Count

You are given a document with many lines.

Define functions mapper and reducer:

```
def mapper(line):
    foreach word in line.split():
        output(word, 1)
```

- word is the key, 1 is the value

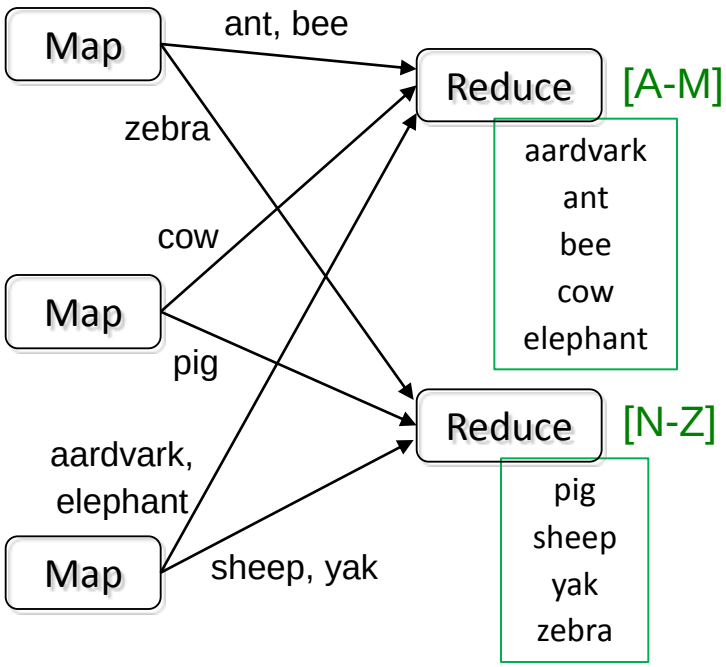
```
def reducer(key, list(values)):
    output(key, sum(values))
```

1. Sort

- **Input:** (key, value) records
- **Output:** same records, sorted by key

- **Map:** identity function
- **Reduce:** identify function

- **Trick:** Pick partitioning function h such that $k_1<k_2 \Rightarrow h(k_1)<h(k_2)$



2. Inverted Index

- **Input:** (filename, text) records
- **Output:** list of files containing each word

- **Map:**

```
foreach word in text.split():
    output(word, filename)
```

- **Combine:** uniquify filenames for each word

- **Reduce:**

```
def reduce(word, filenames):
    output(word, sort(filenames))
```

MapReduce Execution Details

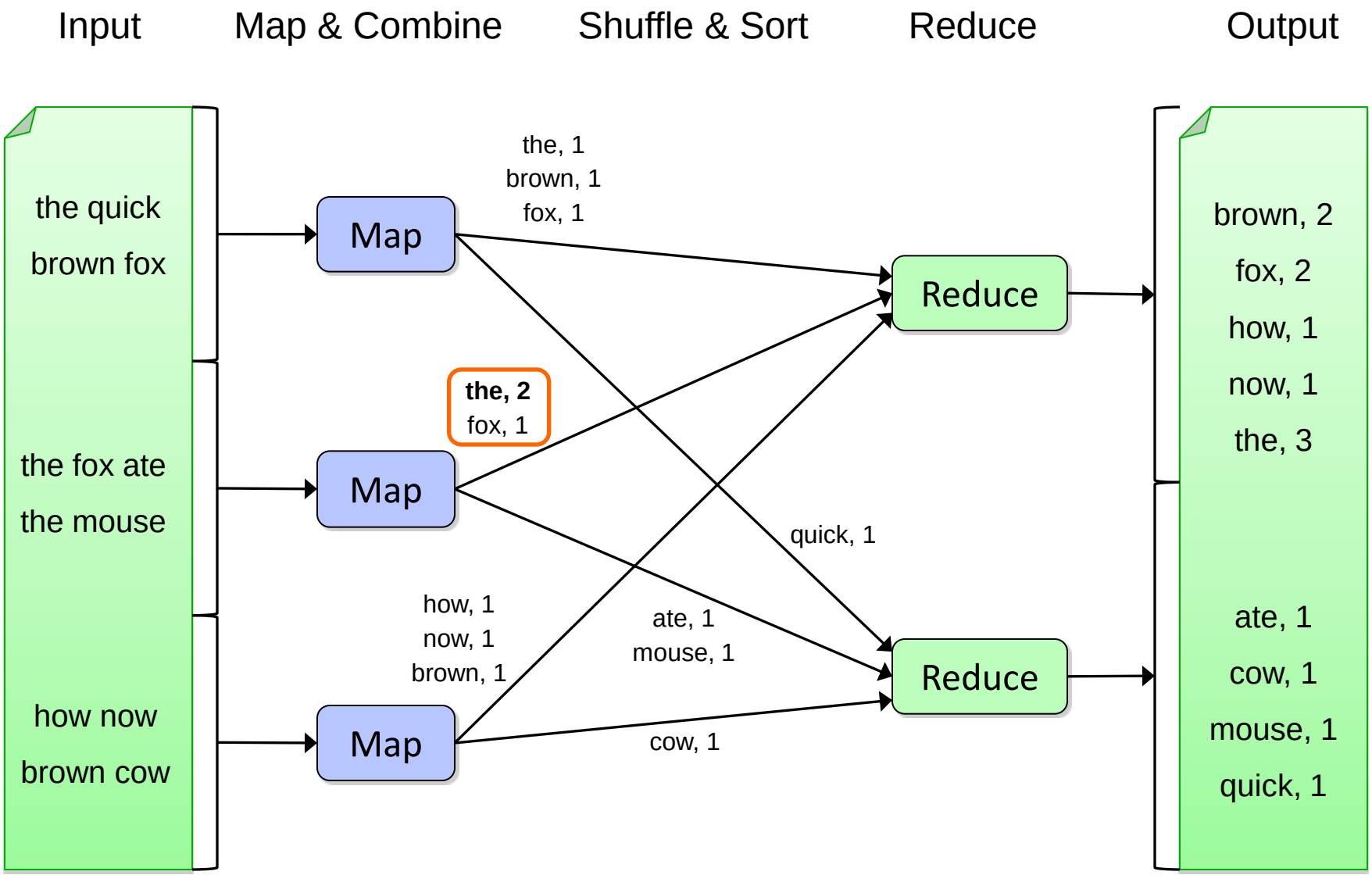
- Single *master* controls job execution on multiple *slaves*
- Mappers preferentially placed on same node or same rack as their input block
  - Minimizes network usage
- Mappers save outputs to local disk before they are served to reducers by Shuffle and Sort phase.
- Shuffle and Sort phase is important and could provide great speedup and other benefits.
  - Allows recovery if a reducer crashes
  - Allows having more reducers than nodes

An Optimization: The Combiner

- Shuffle and Sort phase typically introduces combiner, a local aggregation function, for repeated keys produced by same map
- Combiner works with associative functions like sum, count, max
- Decreases size of intermediate data
- Example: map-side aggregation for Word Count:

```
def combiner(key, values):
    output(key, sum(values))
```

Word Count with Combiner



Fault Tolerance in MapReduce

1. If a task crashes:
   - Retry on another node
     - OK for a map because it has no dependencies, just do it again, duplicate data are saved somewhere else
     - OK for reduce because map outputs are on disk, or several disks, new reduce could start over.
   - If the same task fails repeatedly, fail the job or ignore that input block. Quite often tasks are statistical in nature, no one would notice a slight error, anyway.
2. If a node crashes:
   - Re-launch its current tasks on other nodes
   - Re-run any maps the node previously ran
     - Necessary because their output files were lost along with the crashed node

Fault Tolerance in MapReduce

3. If a task is going slowly (straggler):
   - Launch second copy of task on another node ("speculative execution")
   - Take the output of whichever copy finishes first, and kill the other

- Surprisingly important in large clusters
  - Stragglers occur frequently due to failing hardware, software bugs, misconfiguration, etc
  - Single straggler may noticeably slow down a job

- For these fault tolerance features to work, *your map and reduce tasks must be side-effect-free*

Takeaways

- By providing a data-parallel programming model, MapReduce can control job execution in useful ways:
  - Automatic division of job into tasks
  - Automatic placement of computation near data
  - Automatic load balancing
  - Recovery from failures & stragglers
- User focuses on application, not on complexities of distributed computing

Run Hadoop on Your Own Machine or Cluster

- Typically one wants to install a recent release of Hadoop on a recent release of a popular Linux OS.
- $30 or $60 for the academic license for a Red Hat OS is not bad. Major software vendors, like IBM, like Red Hat.
- Fedora is an open source, free version of Red Hat OS that is ahead of commercial Red Hat and serves as a development platform for what will eventually be packaged as Red Hat.
- CentOS 6.x is a “repackaged” Red Hat 6.x for free non-commercial use.
- Other options are Ubuntu, and Oracle Linux which are free.
- SUSE is yet another popular Linux, but is not free.
- There are several other Linux versions.
- Those not faint-hearted could download all Hadoop code from Apache.org

Welcome to Apache™ Hadoop®!

What Is Apache Hadoop?

The Apache™ Hadoop® project develops open-source software for reliable, scalable, distributed computing.

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers usi is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardwar library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of con prone to failures.

The project includes these modules:

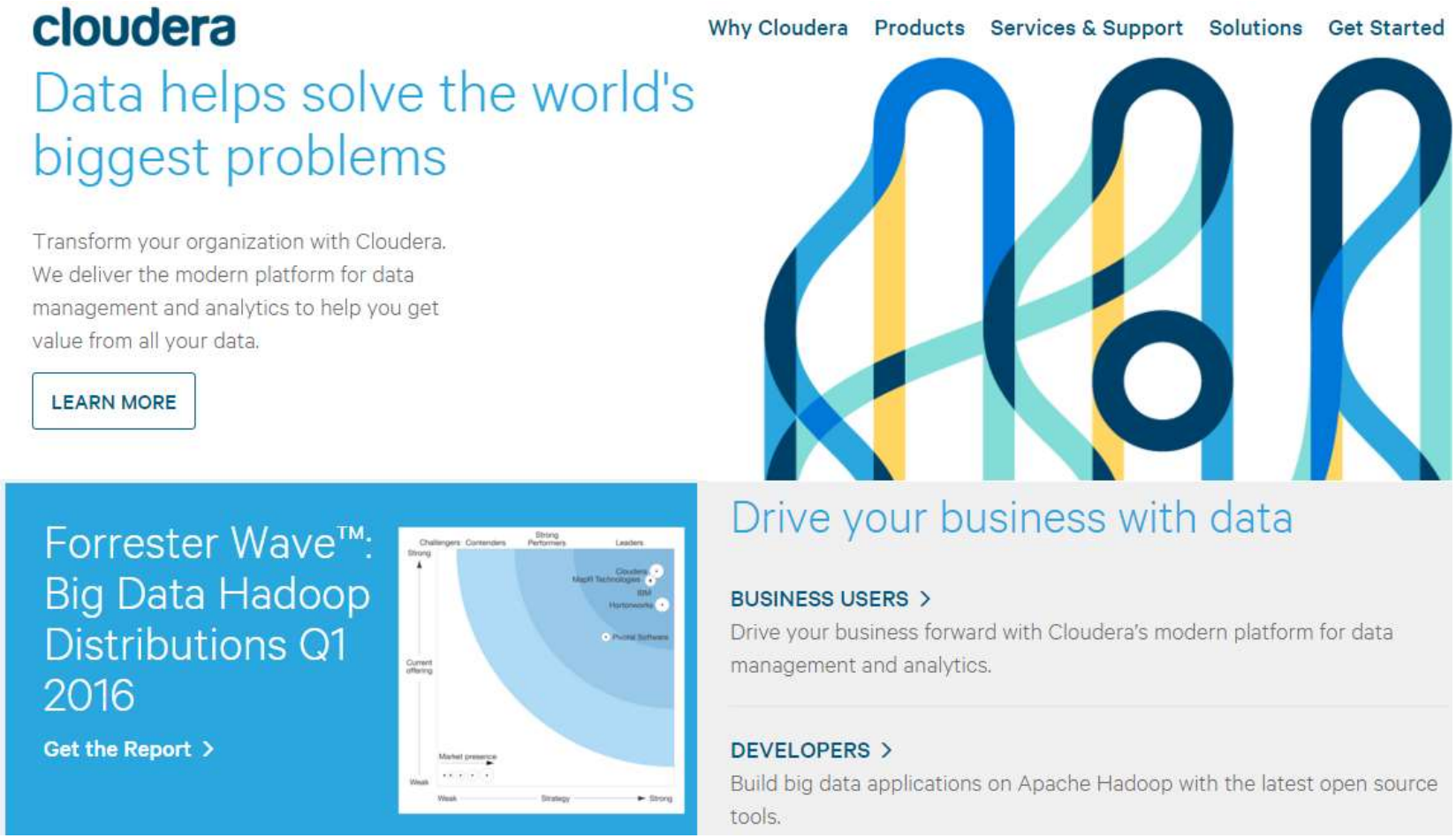
- **Hadoop Common**: The common utilities that support the other Hadoop modules.
- **Hadoop Distributed File System (HDFS™)**: A distributed file system that provides high-throughput access to application data.
- **Hadoop YARN**: A framework for job scheduling and cluster resource management.
- **Hadoop MapReduce**: A YARN-based system for parallel processing of large data sets.

Other Hadoop-related projects at Apache include:

- **Ambari™**: A web-based tool for provisioning, managing, and monitoring Apache Hadoop clusters which includes support for Hadoop HDFS, HCatalog, HBase, ZooKeeper, Oozie, Pig and Sqoop. Ambari also provides a dashboard for viewing cluster health such as heatmaps and ab and Hive applications visually alongwith features to diagnose their performance characteristics in a user-friendly manner.
- **Avro™**: A data serialization system.
- **Cassandra™**: A scalable multi-master database with no single points of failure.
- **Chukwa™**: A data collection system for managing large distributed systems.
- **HBase™**: A scalable, distributed database that supports structured data storage for large tables.
- **Hive™**: A data warehouse infrastructure that provides data summarization and ad hoc querying.
- **Mahout™**: A Scalable machine learning and data mining library.
- **Pig™**: A high-level data-flow language and execution framework for parallel computation.
- **Spark™**: A fast and general compute engine for Hadoop data. Spark provides a simple and expressive programming model that supports a including ETL, machine learning, stream processing, and graph computation.
- **Tez™**: A generalized data-flow programming framework, built on Hadoop YARN, which provides a powerful and flexible engine to execute a process data for both batch and interactive use-cases. Tez is being adopted by Hive™, Pig™ and other frameworks in the Hadoop ecosyste commercial software (e.g. ETL tools), to replace Hadoop™ MapReduce as the underlying execution engine.
- **ZooKeeper™**: A high-performance coordination service for distributed applications.

Cloudera

- Doug Cutting who invented Hadoop is at Cloudera.



Cloudera

- Cloudera thinks big and acts fast. Cloudera's scope and pace of innovation is astounding, but not surprising for the first commercial Hadoop startup, founded in 2008. Cloudera started the SQL-for-Hadoop craze with Impala. It offered the first visual cluster management tool and continues to put significant effort into key features such as security, high availability, governance, and administration.
- The vendor is anything but shy when it comes to making strategic acquisitions and partnerships to fill enterprise gaps in security, data management, and analytics. Cloudera's customers value the commercial add-on tools like Cloudera Manager, Cloudera Navigator, and Impala, as well as the company's overall vision for an enterprise big data platform.

From The Forrester Wave™: Big Data Hadoop Distributions, Q1 2016

Hortonworks

- Hortonworks made significant contribution most importantly YARN and TAZ.

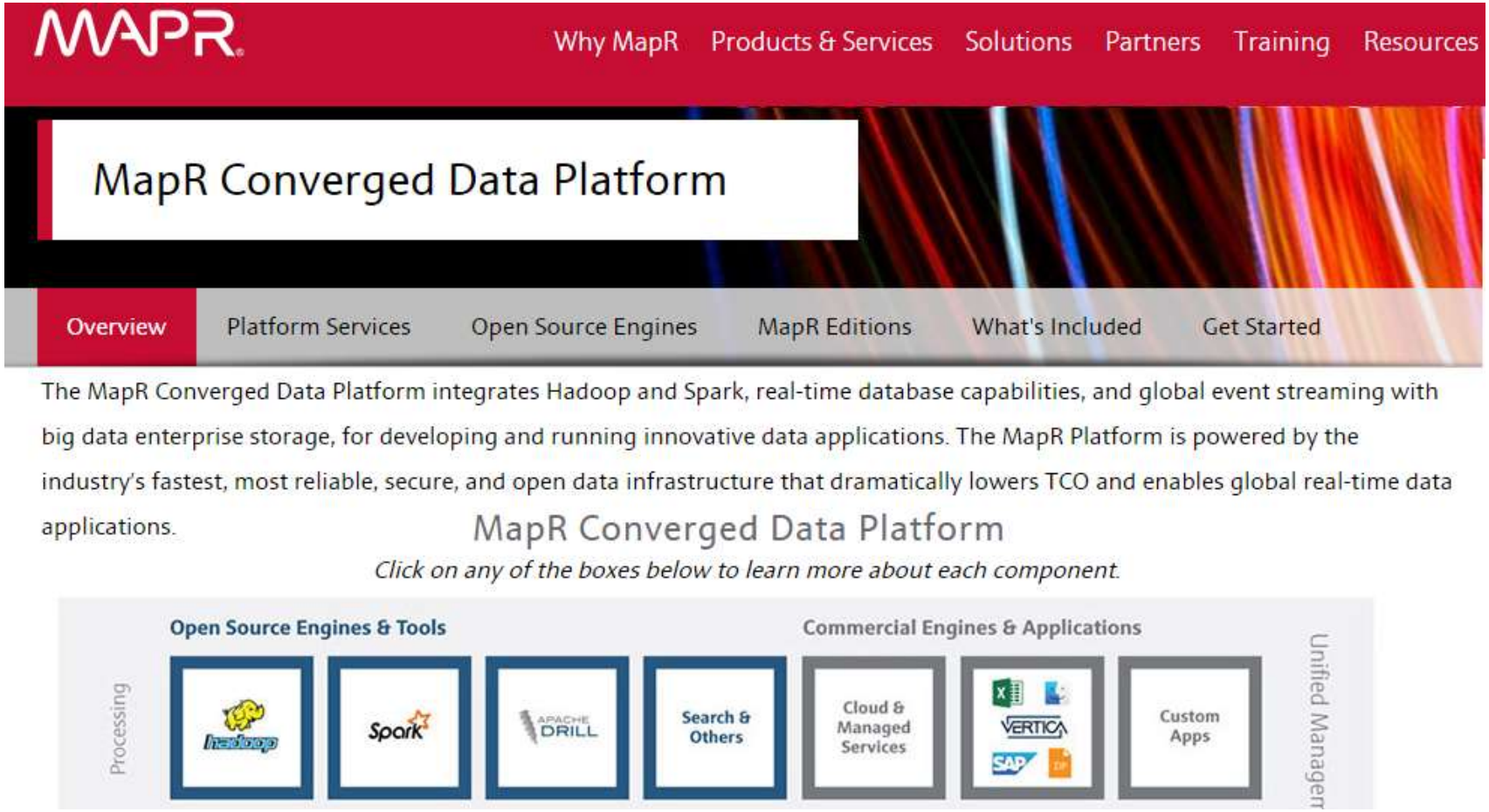
Hortonworks

- Hortonworks doubles-down on inclusive, broad community innovation. Hortonworks is a rock when it comes to its promise to offer a 100% open source distribution. All of the technology built into HDP is an Apache open source project.
- Hortonworks will acquire companies to fill enterprise gaps and immediately contributes the code to an Apache project for the good of the community. For example, Hortonworks acquired XA Secure, a company with a commercially licensed security solution, and contributed the code to Apache as Apache Ranger.
- Hortonworks is also an important member of the Open Data Platform initiative (ODPi) formed earlier this year with IBM, Pivotal Software, and 12 other technology vendors, because the group has adopted Hortonworks-initiated projects such as Apache Ambari.6 Customers value Hortonworks' approach to open source innovation.

From The Forrester Wave™: Big Data Hadoop Distributions, Q1 2016

MapR

- If you are the Government of India and want to store and use biometric information of 1.2 billion people, you use MapR Hadoop



MapR

- MapR Technologies innovates to deliver extreme performance and reliability at scale. From day one, MapR Technologies' strategy has been to engineer a distribution that would allow Hadoop to reach its full performance and scale potential with minimal effort.
- MapR Technologies' linchpin — the MapR filesystem, which implements the HDFS API, is fully read/write, and can store trillions of files (versus the complex configuration for HDFS that requires separated namespaces).
- MapR has also done more than any other distribution vendor under the covers of Hadoop to deliver a reliable and efficient distribution for large-cluster implementations. Its customers typically have or are planning large, mission-critical Hadoop clusters and want to use MapR-DB and MapR Streams (which implement the HBase and Kafka APIs, respectively).

From The Forrester Wave™: Big Data Hadoop Distributions, Q1 2016

IBM BigInsight

IBM's BigInsights: Smart Analytics for Big Data

© 2016 IBM Corporation

IBM BigInsight

- IBM differentiates BigInsights with end-to-end advanced analytics. IBM integrates a wealth of key data management components and analytics assets into the open source core of its Hadoop distribution.
- Enterprises using IBM's data management stack will find BigInsights a natural extension to their existing data platform. The company has also launched an ambitious open source project, Apache SystemML, for machine learning on Apache Spark from its newly minted Spark Technology Center.
- IBM's customers value the maturity and depth of its Hadoop extensions, such as BigSQL, which is one of the fastest and most SQL-compliant of all the SQL-for-Hadoop engines. In addition, BigQuality, BigIntegrate, and IBM InfoSphere Big Match provide a mature and feature-rich set of tools that run natively with YARN to handle the toughest Hadoop use cases.

From The Forrester Wave™: Big Data Hadoop Distributions, Q1 2016

Cloudera CDH

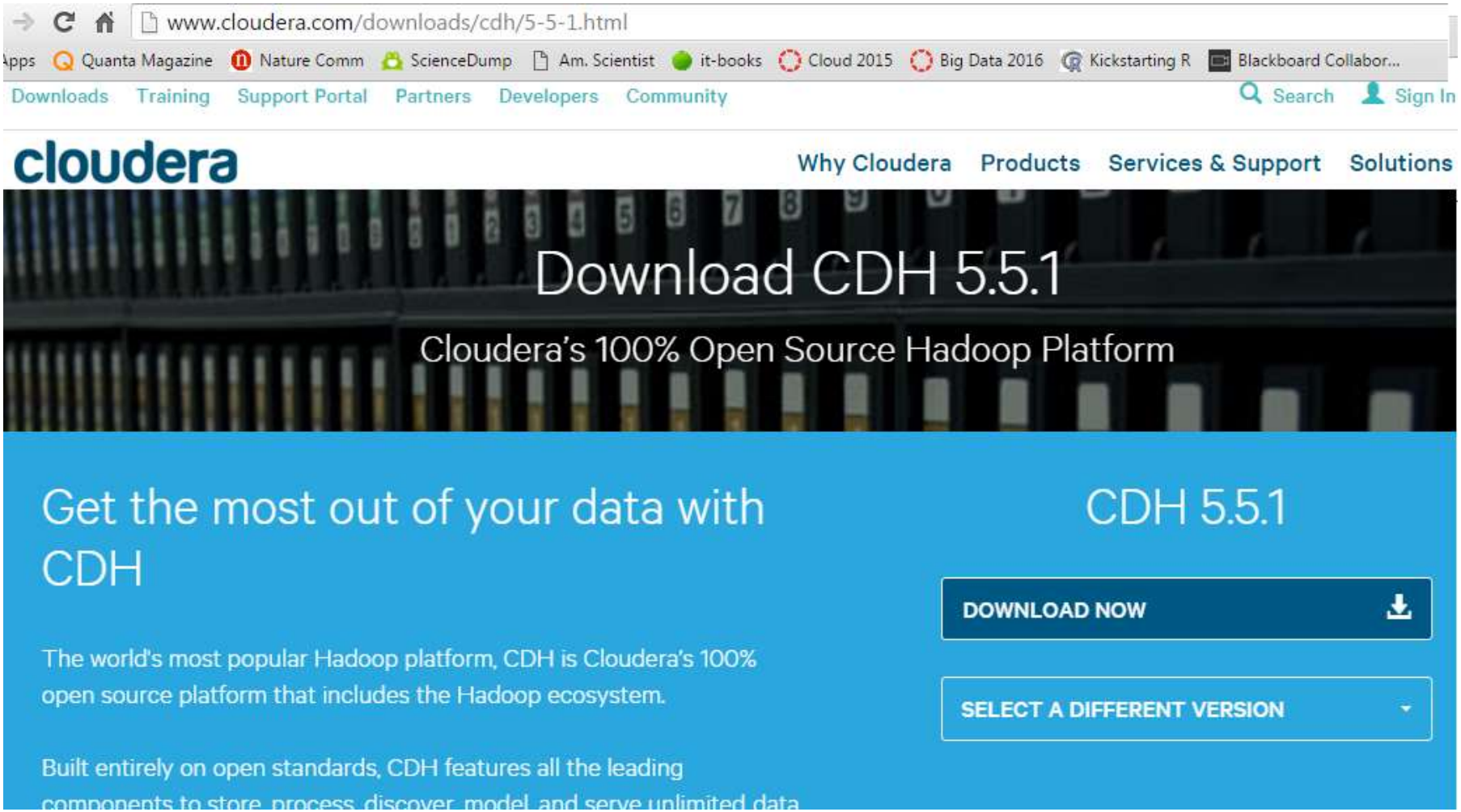
- If you really want to install Hadoop yourself, one option is to use Cloudera's CDH (Cloudera Distribution of Hadoop).
- The latest, non-beta, appears to be CDH 5.5.1

- You read documentation first. You always do!
- If you have a 32 bit machine, you better do. You need a new machine or a new OS, at least.
- Cloudera tools are available for 64 bit machines, only.
- The RAM that these people ask you to have on your machine is getting larger every year.
- This year, they mention 4GB+ and recommend 8GB+. Next year, it will be 8GB+ and 16GB+
- To install CDH go to:

www.cloudera.com/downloads/cdh/5-5-1.html

Cloudera CDH

www.cloudera.com/downloads/cdh/5-5-1.html



CDH, Some of Supported Linux Versions

`CDH 5.5` is supported on many versions of Red Hat, Oracle Linux, CentOS, SUSE, and Ubuntu Linux.

Recommended Java versions are:

Minimum	Recommended
1.7.0_25	1.7.0_80
1.8.0_31	1.8.0_60

CentOS	6.4	64-bit
	6.5	64-bit
	6.5 in SE Linux mode	64-bit
	6.6	64-bit
	6.6 in SE Linux mode	64-bit
	6.7	64-bit
	7.1	64-bit
Oracle Linux with default kernel and Unbreakable Enterprise Kernel	5.6 (UEK R2)	64-bit
	6.4 (UEK R2)	64-bit
	6.5 (UEK R2, UEK R3)	64-bit
	6.6 (UEK R3)	64-bit
	7.1	64-bit
Ubuntu/Debian		
Ubuntu	Precise (12.04) - Long-Term Support (LTS)	64-bit
	Trusty (14.04) - Long-Term Support (LTS)	64-bit
Debian	Wheezy (7.0, 7.1)	64-bit

Supported Databases

- CDH supports various Databases and Connectors

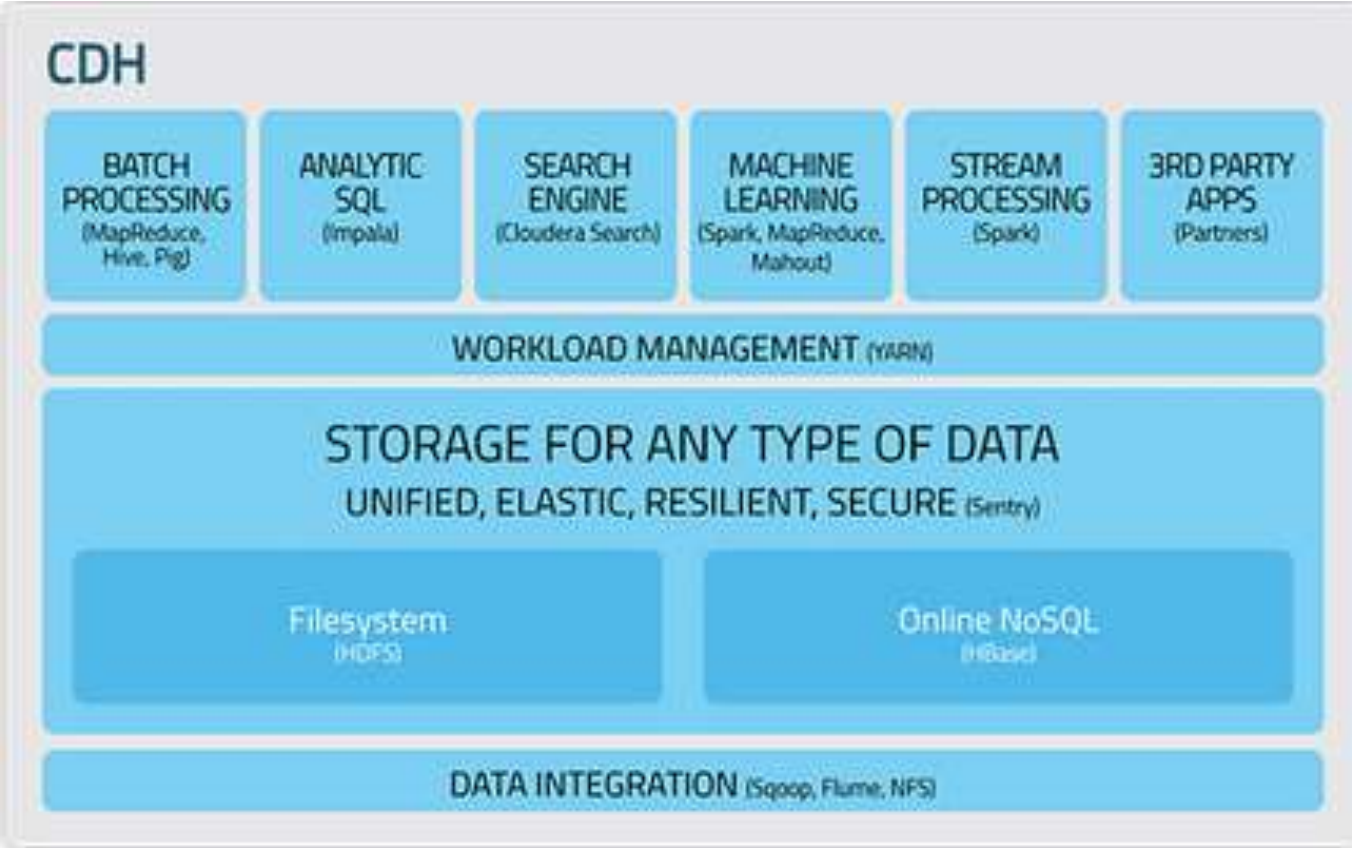
Component	MariaDB	MySQL	SQLite	PostgreSQL	Oracle	Derby
Oozie	5.5	5.5, 5.6	–	9.2, 9.3, 9.4See Note 3	11gR2, 12c	Default
Flume	–	–	–	–	–	Default (for the JDBC Channel only)
Hue	5.5	5.1, 5.5, 5.6See Note 7	Default	9.2, 9.3, 9.4See Note 3	11gR2, 12c	–
Hive/Impala	5.5	5.5, 5.6See Note 1	–	9.2, 9.3, 9.4See Note 3	11gR2, 12c	Default
Sentry	5.5	5.5, 5.6See Note 1	–	9.2, 9.3, 9.4See Note 3	11gR2, 12c	–
Sqoop 1	5.5	See Note 4	–	See Note 4	See Note 4	–
Sqoop 2	5.5	See Note 5	–	See Note 5	See Note 5	Default

Download VM for now

- Cloudera VM is a 64-bit VM, and requires a 64-bit host OS and a virtualization product that can support a 64-bit guest OS.
- This VM uses 4 GB of total RAM. The total system memory required varies depending on the size of your data set and on the other processes that are running.
- The demo VM file is in a 7-zip format and is approximately 4 GB. Feel free to mirror internally or externally to minimize bandwidth usage.
- To use the VMware VM, you must use a player compatible with WorkStation 8.x or higher: Player 4.x or higher, ESXi 5.x or higher, or Fusion 4.x or higher. Older versions of WorkStation can be used to create a new VM using the same virtual disk (VMDK file), but some features in VMware Tools won't be available.

Cloudera CDH

- Cloudera offers CDH (Cloudera Distribution of Apache Hadoop), an open source software distribution of Apache Hadoop and additional key packages from Hadoop ecosystem. CHD makes installation and deployment of Hadoop ecosystem almost easy.
- Mutual compatibility of various packages is taken care off for you.
- CDH offers unified querying options (including batch processing, interactive SQL, text search, and machine learning) and enterprise security features.



CDH 5.5.1 Packaging and Tarball Information

http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/cdh_vd_cdh_package_tarball.html

CDH 5 packages are labeled using the following format:

component-base_version+cdh_version+patch_level

URL where you could find 5.5.1 tarballs is on top of the page

Component	Package Version
Apache Avro	avro-1.7.6+cdh5.5.1+111
Apache Crunch	crunch-0.11.0+cdh5.5.1+77
DataFu	pig-udf-datafu-1.1.0+cdh5.5.1+17
Apache Flume	flume-ng-1.6.0+cdh5.5.1+29
Apache Hadoop	hadoop-2.6.0+cdh5.5.1+924
Apache HBase	hbase-1.0.0+cdh5.5.1+274
HBase-Solr	hbase-solr-1.5+cdh5.5.1+57
Apache Hive	hive-1.1.0+cdh5.5.1+327
Hue	hue-3.9.0+cdh5.5.1+333
Cloudera Impala	impala-2.3.0+cdh5.5.1+0
Kite SDK	kite-1.0.0+cdh5.5.1+116
Llama	llama-1.0.0+cdh5.5.1+0

CDH 5.5.1 Packaging and Tarball Information

- CDH5.X has a rich set of features and packages.
- Each package /framework has its own tarball.

Component	Version
Apache Mahout	mahout-0.9+cdh5.5.1+26
Apache Oozie	oozie-4.1.0+cdh5.5.1+223
Parquet	parquet-1.5.0+cdh5.5.1+176
Parquet-format	parquet-format-2.1.0+cdh5.5.1+12
Apache Pig	pig-0.12.0+cdh5.5.1+72
Cloudera Search	search-1.0.0+cdh5.5.1+0
Apache Sentry (incubating)	sentry-1.5.1+cdh5.5.1+106
Apache Solr	solr-4.10.3+cdh5.5.1+325
Apache Spark	spark-1.5.0+cdh5.5.1+94
Spark Netlib	spark-netlib-master.3
Apache Sqoop	sqoop-1.4.6+cdh5.5.1+29
Apache Sqoop2	sqoop2-1.99.5+cdh5.5.1+33
Apache Whirr	whirr-0.9.0+cdh5.5.1+17
Apache ZooKeeper	zookeeper-3.4.5+cdh5.5.1+91

Objectives

- In order to create a machine that runs Hadoop software we will do the following:
  - Download a supported version of CentOS Linux system. We will work with CentOS 6.7.
  - If you know what you are doing, you can work with any other supported OS and any other version.
  - Use VMWare Workstation 11 (Fusion 7) to create a virtual machine (VM) with selected Linux OS.
  - Install CDH5.5 and contained packages.

As user`root`, Check for Java

- Check for version of Java. Hadoop needs Java 7 or 8.

```
[root@localhost ~]# which java
/usr/bin/which: no java in
(/usr/local/sbin:/usr/sbin:/sbin:/usr/local/bin:/usr/bin:/bin:/root/bin
)
```

- Since Java is not there, we need to install it. I will use Java 8
- CDH5.5 recommends using `jdk1.8_60`. You can fetch it at:

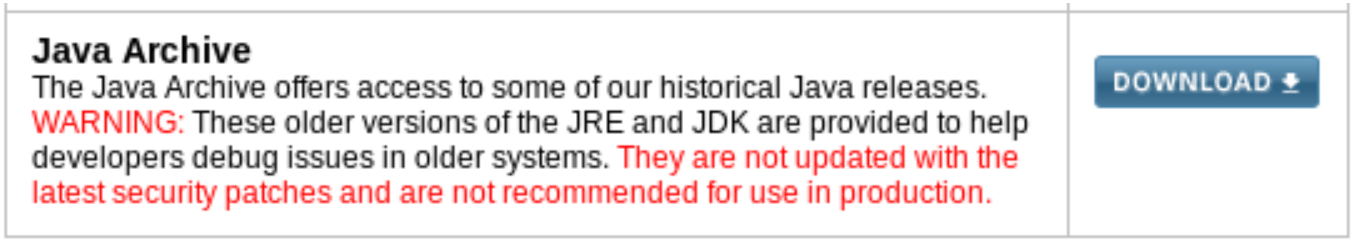
http://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase7-521261.html#jre-8u60-oth-JPR

- Select Linux x64 `jdk-8u60-linux-x64.rpm`, (for a 64 bit Linux).
- When you click on the download link, you will be asked to save the file.
- You will also be asked for a username and password. If you do not have them, register on the spot. Registration and downloads are free.
- On your VM, you are the user `cloudera`, and the file will go into the directory `/home/cloudera/Downloads`
- Suffix `rpm` stands for **R**ed Hat **P**ackage **M**anager. `RPM` is a file format and a package management system used on several Linux distributions.

Find Java SDK

- In order to adhere to the recommended release of Java 8_60 we need to go to Java SE Downloads:

- And then scroll to the bottom of the page to Java Archives



Hit Download

Download

- On the next pages select Java SE 8 and then Java SE Development Kit 8u60.
- Download `rpm` file for Linux x64 version 8u60.
- You should first accept license agreement and might have to provide Oracle Tech Net username and password. If you do not have them, register. Registration is free.

Java SE Development Kit 8u60

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.
Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux ARM v6/v7 Hard Float ABI	77.69 MB	jdk-8u60-linux-arm32-vfp-hflt.tar.gz
Linux ARM v6/v7 Hard Float ABI	74.64 MB	jdk-8u60-linux-arm64-vfp-hflt.tar.gz
Linux x86	154.66 MB	jdk-8u60-linux-i586.rpm
Linux x86	174.83 MB	jdk-8u60-linux-i586.tar.gz
Linux x64	152.67 MB	jdk-8u60-linux-x64.rpm

`rpm` file will go to directory `Downloads` under `/home/cloudera`

Installing Java

1. If not `root`, become `root` by running the `$ su` command and entering `root`'s password.

2. Install JDK by running `rpm` command

```
$ rpm -ivh jdk-8u60-linux-x64.rpm
```

- To uninstall this package type:

  ```
  $ rpm -e jdk-8
  ```

- On some installations, the script displays a binary license agreement, which you are asked to agree to before installation can proceed. Once you have accepted the license, `rpm` runs the file `jdk-8u60-linux-x64.rpm` in the current directory.
- **NOTE:** `rpm` process sends `jdk-8` files to `/usr/java/jdk1.8.0_60` directory.
- Java executable is sent to `/usr/bin`, as you could see by running `$ which java`

3. Delete the `rpm` file if you want to save disk space.

4. To see which Linux packages are installed, type:

`$ rpm -qa | grep jdk` or just `$ rpm -qa`

Set JAVA_HOME for user root

- If you are root, go to root's home directory (~ represents current user's home)

```
$ cd ~
```

- As root open file /root/.bash_profile using vi editor and add JAVA_HOME.
- Before adding a line in vi, hit Esc(ape) key and then lower case i, for inserting. Your .bash_profile should at least have the following lines:

```
JAVA_HOME=/usr/java/jdk1.8.0_60
export JAVA_HOME
PATH=$PATH:$HOME/bin:$JAVA_HOME/bin
export PATH
```

- When done editing .bash_profile, exit by typing Esc and then :wq , for write and quit.
- In order for your session to become aware of new values, or new variables, you need to source .bash_profile file.
- In the directory where .bash_profile resides (e.g. /root) type:

```
$ source .bash_profile
```

- To verify new values, type

```
$ echo $PATH
$ cd $JAVA_HOME
```

Set JAVA_HOME for user cloudera

- If you are root, type exit and verify that you are user cloudera

```
$ whoami
cloudera
```

- As user cloudera open file /home/cloudera/.bash_profile using vi editor and add JAVA_HOME.
- cloudera's .bash_profile file should at least have the following lines:

```
JAVA_HOME=/usr/java/jdk1.8.0_60
export JAVA_HOME
PATH=$PATH:/$HOME/bin:$JAVA_HOME/bin
export PATH
```

- When done editing .bash_profile, exit by typing Esc and then :wq for write and quit.
- In order for your session to become aware of new values, or new variables, you need to source .bash_profile file.
- In the directory where .bash_profile resides (e.g. /home/cloudera). Type:

```
$ source .bash_profile
```

- To verify that new values are visible, type

```
$ echo $PATH
$ cd $JAVA_HOME
```

Versions of Hadoop, MapReduce

- It appears that Apache (Cloudera) Hadoop has two major versions, older Hadoop 1 and more modern Hadoop 2.
- CDH5.5 supports Hadoop 2, release 2.5
- Hadoop 2 is more scalable and more reliable than Hadoop 1. For example, in Hadoop 2 we could have more than one name node.
- The major new component of Hadoop 2 is called Yarn.
- **YARN** is a framework that facilitates writing arbitrary distributed processing frameworks and applications, and not only MapReduce.
- YARN provides the daemons and APIs necessary to develop generic distributed applications of any kind, and not only MapReduce, handles and schedules resource requests (such as memory and CPU) from such applications, and supervises their execution.
- YARN's execution model (MRv2) is more generic than the earlier MapReduce implementation.
- YARN can run applications that do not follow the MapReduce model, unlike the original Apache Hadoop MapReduce (also called MR1).

MRv2

What is new in MR2 or MRv2?

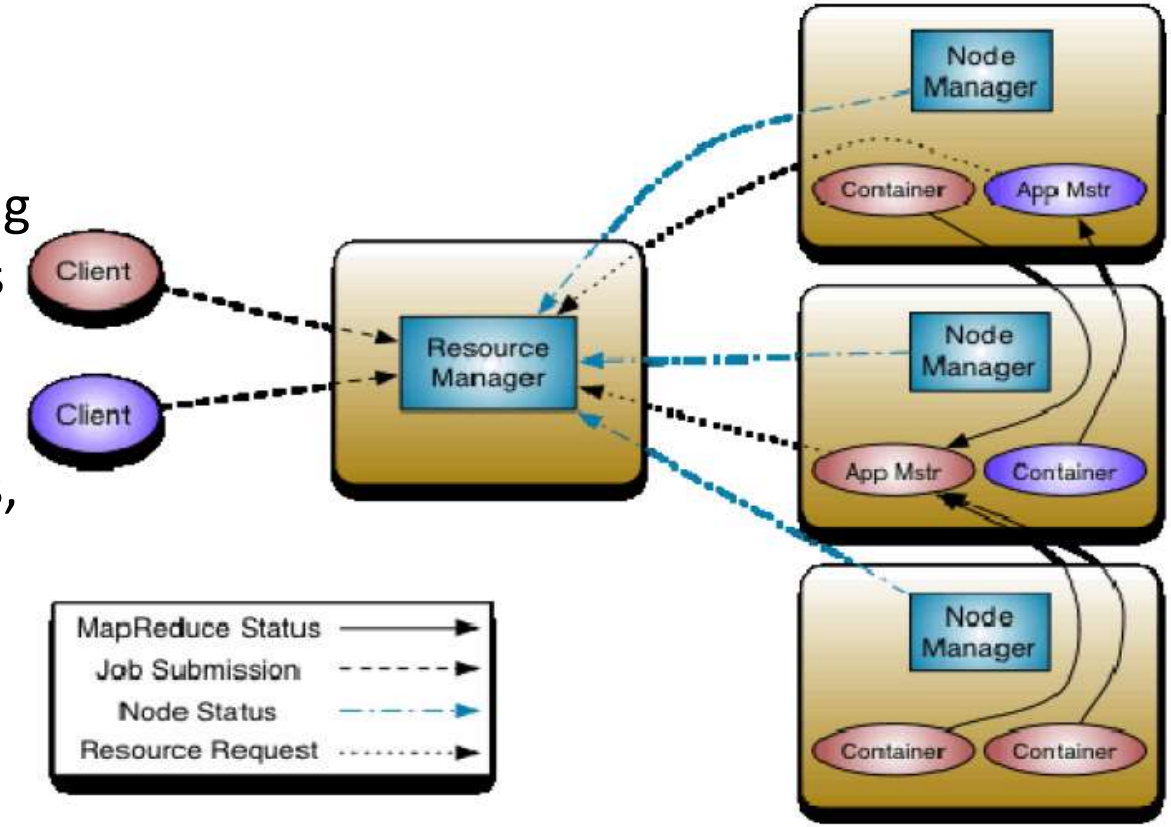
- With YARN, there is no longer a single JobTracker to run jobs and a TaskTracker to run tasks of the jobs.
- MR2 utilizes the familiar MapReduce execution underneath, except that each job now controls its own destiny via its own ApplicationMaster taking care of execution flow (such as scheduling tasks, handling speculative execution and failures, etc.).
- MR2 is a more isolated and scalable model than the MR1 system where a singular JobTracker does all the resource management, scheduling and task monitoring work.
- **If MRv2 is so much better , why mention MRv1?**
  - There is a large number of applications written in MRv1 and you want to use them and not rewrite them without a good reason.
  - A large number of existing books are written with examples in MRv1 Java API. All of those examples are useful. The difference between MRv1 and MRv2 APIs is not huge, still you would have to make changes to the code to run those examples on YARN.
- You should do all new development on YARN (MRv2)

MRv2 (YARN)

- The fundamental idea of MRv2's YARN architecture is to split up the two primary responsibilities of the JobTracker — resource management and job scheduling/monitoring — into separate daemons: a global ResourceManager (RM) and per-application ApplicationMasters (AM).
- With MRv2, the ResourceManager (RM) and per-node NodeManagers (NM), form the data-computation framework.
- The ResourceManager service effectively replaces the functions of the JobTracker, and NodeManagers run on slave nodes instead of TaskTracker daemons.
- The per-application ApplicationMaster is, in effect, a framework specific library and is tasked with negotiating resources from the ResourceManager and working with the NodeManager(s) to execute and monitor the tasks.

YARN Architecture (from hadoop.apache.org)

- The ResourceManager has two main components: Scheduler and ApplicationsManager.
- The Scheduler is responsible for allocating resources to the various running applications subject to familiar constraints of capacities, queues etc.
- The Scheduler is pure scheduler in the sense that it performs no monitoring or tracking of status for the application.



Installation of CDH 5.5.x with Cloudera Manager

- Hadoop eco system comes in many packages. For the basic Map Reduce we need Hadoop itself and could download it separately.
- For other tools like Hive, Pig, Hbase, etc., we need additional packages.
- If you know what you are doing, you could download the source code for all of those packages, improve the source code and then deploy it.
- The main product of Cloudera is the Cloudera Hadoop Distribution or CDH. Currently CDH is at version 5.5.1
- All installations of Cloudera provided packages could be done using a tool called Cloudera Manager.
- Cloudera Manager is a graphical tool and is quite convenient, and much safer (in the sense of avoiding mistakes). Cloudera Manager could also effortlessly install Java and all desired packages on all machines in you cluster.
- In production, at your work, use Cloudera Manager.
- Cloudera Manager requires 10GB of RAM on all machines in your cluster. Since our laptops rarely have more than 4 or 8 GB we will install CDH using a different process.

Manual Installation of CDH 5.5.x with Yarn (Mrv2)

- We want to install Hadoop with YARN on CentoOS6.7
- We will install CDH on a single Linux node in what is called a **pseudo-distributed mode**. This mode utilizes all cores (processors) of your machine and all processes (services) communicate over local TCP sockets for inter-process communications.
- The following text follows Cloudera Quick Start Guide for CDH5.5.x
- PDF version (if you find it) is easier to print and read.
- HTML version has working links, though. You can find it at:

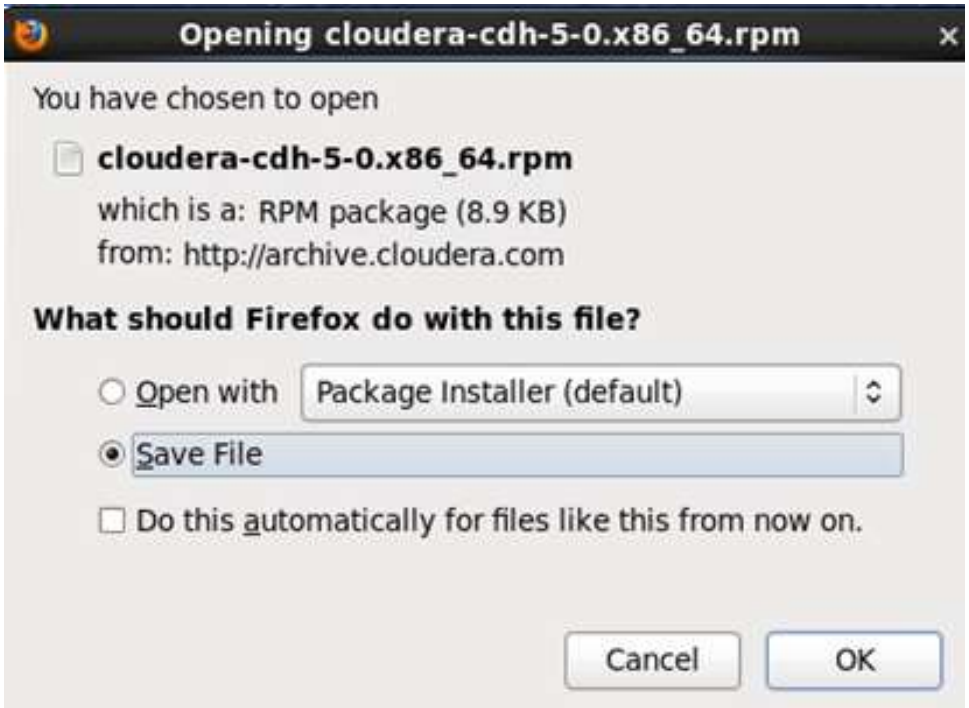
http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/cm_qs_quick_start.html#../topics/cdh_qs.html

Installing CDH5.5.1 on a Single Pseudo-Distributed Node

- In order to download CDH5.5.1 package, in your CentOS VM, open Mozilla, and for 64 bit Redhat (CentOS) 6.X systems, go to

http://archive.cloudera.com/cdh5/one-click-install/redhat/6/x86_64/cloudera-cdh-5-0.x86_64.rpm

Download file `cloudera-cdh-5-0.x86_64.rpm`



- File will end up in `/home/cloudera/Downloads` directory

In case you need to stop and uninstall Hadoop

- In the case you have YARN (MRv2) already installed and want to uninstall it, stop all the daemons first by typing:

```
$ for x in `cd /etc/init.d ; ls hadoop-hdfs-*`; do sudo service
$x stop ; done      # all on one line is fine
$ for x in 'cd /etc/init.d ; ls hadoop-mapreduce-* ; do sudo
service $x stop ; done
$ for x in 'cd /etc/init.d ; ls hadoop-yarn-* ; do sudo service
$x stop ; done
```

- To remove Hadoop on Red Hat-compatible systems type:

```
$ sudo yum remove hadoop-conf-pseudo hadoop-mapreduce-*
$ sudo yum remove hadoop-conf-pseudo hadoop-hdfs-*
$ sudo yum remove hadoop-conf-pseudo hadoop-yarn-*
```

Install the RPM (Install CDH5.5)

- To install RPM type:

```
$ sudo yum --nogpgcheck localinstall cloudera-cdh-5-0.x86_64.rpm
```

- Accept all defaults when asked. After a while, you will get:

```
Installed:   cloudera-cdh.x86_64 0:5-0
```

- Optionally, we could add the Cloudera Public GPG Key (GNU Privacy Guard) to the local repository by executing, all on one line:

```
$ sudo rpm --import
http://archive.cloudera.com/cdh5/redhat/6/x86_64/cdh/RPM-GPG-KEY-
cloudera
```

- That key will be used for encrypting traffic between Hadoop nodes.
- Next, install Hadoop with YARN in pseudo-distributed mode:

```
$ sudo yum install hadoop-conf-pseudo
```

- `yum` will list a fairly large number of packages and their combined download size and ask a few times whether you want to proceed.
- You should always say `y(es)`

Yum

- The **Yellowdog Updater, Modified** (**yum**) is an open-source command-line package-management utility for Linux operating systems using the RPM Package Manager.
- Yum has a command-line interface only.
- Yum allows automatic updates, package and dependency management, on RPM-based distributions.
- Yum works with software repositories (collections of packages), which can be accessed locally or over a network connection.
- Under the hood, yum depends on RPM, which is a packaging standard for digital distribution of software, which automatically verifies the authorship and integrity of manipulated software.
- Yum nor RPM provide built-in support for proprietary restrictions on copying of packages by end-users.
- Yum is implemented as libraries in the Python programming language, with a small set of programs that provide a command-line interface.

Verifying Pseudo-Distributed Configuration, YARN

- Pseudo-distributed YARN installation has a single node running five daemons called: `namenode, secondarynamenode, resourcemanager, datanode` and `nodemanager.`
- To view configuration files for YARN on Red Hat (CenOS), type

```
$ rpm -ql hadoop-conf-pseudo
/etc/hadoop/conf.pseudo
/etc/hadoop/conf.pseudo/README
/etc/hadoop/conf.pseudo/core-site.xml
/etc/hadoop/conf.pseudo/hadoop-env.sh
/etc/hadoop/conf.pseudo/hadoop-metrics.properties
/etc/hadoop/conf.pseudo/hdfs-site.xml
/etc/hadoop/conf.pseudo/log4j.properties
/etc/hadoop/conf.pseudo/mapred-site.xml
/etc/hadoop/conf.pseudo/yarn-site.xml
[cloudera@localhost Downloads]$
```

- The configration of our Hadoop installation is all contained in `/etc/hadoop/conf.pseudo` directory.
- All Hadoop components in pseudo distributed system search for the Hadoop configurations in `/etc/hadoop/conf.pseudo` directory

Format NameNode

- Before starting the NameNode for the first time we must format the HDFS file system. The installation process did not do that for us.
- Formatting of namenode must be performed as user `hdfs`. You can do that using command `hdfs namenode -format` with an additional `sudo -u hdfs` as the initial part of the command string, as below:

```
$ sudo -u hdfs hdfs namenode -format
15/03/05 11:57:08 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = localhost.localdomain/127.0.0.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 2.5.0-CDH5.5.2
STARTUP_MSG:   classpath = /etc/hadoop/conf:/. . .
15/02/05 11:25:18 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at
localhost.localdomain/127.0.0.1
************************************************************/
```

- In earlier releases of CHD `hadoop-config-pseudo` package performed this formatting automatically. Two `hdfs hdfs` strings in the instruction above are needed. The first is the name of the user, the second is the command itself.

`hdfs` script and its commands

```
root@wkst-72-157 etc]# hdfs
Usage: hdfs [--config confdir] COMMAND  where COMMAND is one of:
  dfs                  run a filesystem command on the file systems supported in Hadoop.
  namenode -format     format the DFS filesystem
  secondarynamenode    run the DFS secondary namenode
  namenode             run the DFS namenode
  journalnode          run the DFS journalnode
  zkfc                 run the ZK Failover Controller daemon
  datanode             run a DFS datanode
  dfsadmin             run a DFS admin client
  haadmin              run a DFS HA admin client
  fsck                 run a DFS filesystem checking utility
  balancer             run a cluster balancing utility
  jmxget               get JMX exported values from NameNode or DataNode.
  mover                run a utility to move block replicas across storage types
  oiv                  apply the offline fsimage viewer to an fsimage
  oiv_legacy           apply the offline fsimage viewer to an legacy fsimage
  oev                  apply the offline edits viewer to an edits file
  fetchdt              fetch a delegation token from the NameNode
  getconf              get config values from configuration
  groups               get the groups which users belong to
  snapshotDiff         diff two snapshots of a directory or diff the current directory contents with a snapshot
  lsSnapshottableDir   list all snapshottable dirs owned by the current user
  portmap              run a portmap service
  nfs3                 run an NFS version 3 gateway
  cacheadmin           configure the HDFS cache
  crypto               configure HDFS encryption zones
  storagepolicies      list/get/set block storage policies
  version              print the version
```

hdfs dfsadmin

- `hdfs dfsadmin` is an administrative client. The command allows you to quickly review status of your daemons.
- For example, if daemons are not up, you get something like:

```
$ sudo -u hdfs hdfs dfsadmin -report
Configured Capacity: 0 (0 B)
Present Capacity: 0 (0 B)
DFS Remaining: 0 (0 B)
DFS Used: 0 (0 B)
DFS Used%: NaN%
Under replicated blocks: 0
Blocks with corrupt replicas: 0
Missing blocks: 0
```

Start HDFS

- Before we start HDFS, list all hadoop executables in `/etc/init.d` directory that contain `hadoop-` and then start them as services.

```
[cloudera@localhost init.d]$ ls -la hadoop-*
-rwxr-xr-x. 1 root root 4335 Nov 20 15:27 hadoop-hdfs-datanode
-rwxr-xr-x. 1 root root 4469 Nov 20 15:27 hadoop-hdfs-namenode
-rwxr-xr-x. 1 root root 4202 Nov 20 15:27 hadoop-hdfs-secondarynamenode
-rwxr-xr-x. 1 root root 4221 Nov 20 15:27 hadoop-mapreduce-historyserver
-rwxr-xr-x. 1 root root 4138 Nov 20 15:27 hadoop-yarn-nodemanager
-rwxr-xr-x. 1 root root 4182 Nov 20 15:27 hadoop-yarn-resourcemanager
```

```
$ for x in `cd /etc/init.d ; ls hadoop-hdfs-*` ;
  do sudo service $x start ;
  done
```

- It is very important that start all hadoop processes as Linux services. Services ensure that environments are set properly and the executables have all needed dependencies.
- To verify that services have started, we could visit http://localhost:50070/ where `NameNode` provides a web report on Distributed File System `(DFS)` capacity, number of `DataNodes` and logs.

NameNode, DFS Status Page, localhost:50070

- NameNode has its status reporting web site at port 50070, where one could inquire about the health of the nameNode and HDFS file

localhost:50070/dfshealth.html#tab-overview

Hadoop | Overview | Datanodes | Snapshot | Startup Progress | Utilities

Overview 'localhost:8020' (active)

Started:	Thu Mar 05 17:31:19 PST 2015
Version:	2.5.0-cdh5.3.2,
Compiled:	2015-02-24T20
Cluster ID:	CID-45cc6750-
Block Pool ID:	BP-363293384-

Heap Memory used 37.22 MB of 57.75 MB Heap Memory. Max Heap Memory is 966.69 MB.

Non Heap Memory used 26.66 MB of 28.19 MB Commited Non Heap Memory. Max Non Heap Memory is 130 MB.

Summary

Security is off.

Safemode is off.

1 files and directories, 0 blocks = 1 total files

Configured Capacity:	26.29 GB
DFS Used:	24 KB
Non DFS Used:	4.71 GB
DFS Remaining:	21.58 GB
DFS Used%:	0%
DFS Remaining%:	82.08%
Block Pool Used:	24 KB
Block Pool Used%:	0%
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	1 (Decommissioned: 0)
Dead Nodes	0 (Decommissioned: 0)
Decommissioning Nodes	0
Number of Under-Replicated Blocks	0

Create /tmp, /var and /var/log HDFS directories

- For a variety of users: `hadoop-yarn, hbase, benchmarks, jenkins, hive, root, hue,` and `oozie`, we need to create user home directories of the form `/user/$USER` (e.g. `/user/hbase`) and a series of directories of the type: `/tmp, /tmp/hbase, /var, /var/log, var/log/$USER` (e.g. `/var/log/hbase`).
- These fundamental directories are created by the special HDFS administrative user `hdfs` which then changes the mode (access to those directories to `777`, typically) and transfers ownership of those directories to above mentioned users.
- On YARN (MRv2) and MRv1 these tasks are accomplished by running script init-hdfs.sh which resides in directory `/usr/lib/hadoop/libexec`, i.e.

```
$ sudo /usr/lib/hadoop/libexec/init-hdfs.sh
```

Examples of commands used inside `init-hdfs.sh` script are given on the next slide. Do not run those commands if you ran `init-hdfs.sh`

init-hdfs.sh

- If directory `/tmp` exists, one has to recursively remove it

```
$ sudo -u hdfs hadoop fs -rm -r /tmp
```

- For Hadoop with YARN, `init-hdfs.sh` creates new `/tmp` directory and sets permissions

```
$ sudo -u hdfs hadoop fs -mkdir -p /tmp/hadoop-yarn/staging/done_intermediate
$ sudo -u hdfs hadoop fs -chown -R mapred:mapred /tmp/hadoop-yarn/staging
$ sudo -u hdfs hadoop fs -chmod -R 1777 /tmp
```

- It is necessary to create `/var/log/hadoop/yarn` because it is the parent of `/var/log/hadoop-yarn/apps` which is explicitly configured in `yarn-site.xml`.

Verify HDFS File Structure and start YARN

- Run the following command

```
[cloudera@localhost ~]$ sudo -u hdfs hadoop fs -ls -R /
```

The last "/" in the above command is the root of the file system.

- On YARN, we will see a list of directories

```
drwxrwxrwt - hdfs supergroup  0 2016-02-05 09:13 /tmp
drwxrwxrwt - hdfs supergroup  0 2016-02-05 09:13 /tmp/hadoop-yarn
drwxrwxrwt - mapred mapred 0 2016-02-05 09:13 /tmp/hadoop-yarn/staging
drwxrwxrwt - mapred mapred 20. .09:13 /tmp/hadoop-yarn/staging/done_intermediate
drwxr-xr-x - hdfs   supergroup   0 20. . 09:21 /var
drwxr-xr-x - hdfs   supergroup   0 20. . 09:21 /var/log
drwxr-xr-x - yarn   mapred       0 20. . 09:21 /var/log/hadoop-yarn
```

- In order to start YARN services, from /etc/init.d directory type the following:

```
$ sudo service hadoop-yarn-resourcemanager start
$ sudo service hadoop-yarn-nodemanager start
$ sudo service hadoop-mapreduce-historyserver start
```

`hadoop` scrip

- On the previous side user `hdfs` invoked a `hadoop` command. If you type:

```
$ which hadoop
/usr/bin/hadoop
```

- If you open `/usr/bin/hadoop` file, you will see that it invokes another file

```
/usr/lib/hadoop/bin/hadoop
```

- That other `hadoop` file is also a script which runs various Java programs depending on the options you pass to it.
- To get those options invoke `hadoop` by itself:

```
$ hadoop
```

Options of hadoop script

```
[root@wkst-72-157 etc]# hadoop
Usage: hadoop [--config confdir] COMMAND
       where COMMAND is one of:
  fs                   run a generic filesystem user client
  version              print the version
  jar <jar>            run a jar file
  checknative [-a|-h]  check native hadoop and compression
libraries availability
  distcp <srcurl> <desturl> copy file or directories recursively
  archive -archiveName NAME -p <parent path> <src>* <dest> create
                                a hadoop archive
  classpath            prints the class path needed to get the
  credential           interact with credential providers
                       Hadoop jar and the required libraries
  daemonlog            get/set the log level for each daemon
  trace                view and modify Hadoop tracing settings or
  CLASSNAME            run the class named CLASSNAME

Most commands print help when invoked w/o parameters.
```

Distributed File System, fs command

- Hadoop has access to both local, Linux, file system, and its own distributed file system (HDFS Hadoop Distributed File System)
- We access HDFS through Hadoop distributes file system shell, `fs`.
  `$ hadoop  fs`
- Will get a long list of options. Some of those resemble Unix (Linux) commands. Some are different.
- We use those commands to create directories in the HDFS, copy files between HDFS and the local file system, Internet and AWS S3 buckets.
- When you use `fs`, you always prefix it with `hadoop`.

File system shell `fs`

```
hadoop@domU-12-31-39-00-69-A7:~$ hadoop fs
Usage: java FsShell
           [-ls <path>]
           [-lsr <path>]
           [-du <path>]
           [-dus <path>]
           [-count[-q] <path>]
           [-mv <src> <dst>]
           [-cp <src> <dst>]
           [-rm [-skipTrash] <path>]
           [-rmr [-skipTrash] <path>]
           [-expunge]
           [-put <localsrc> ... <dst>]
           [-copyFromLocal <localsrc> ... <dst>]
           [-moveFromLocal <localsrc> ... <dst>]
           [-get [-ignoreCrc] [-crc] <src> <localdst>]
           [-getmerge <src> <localdst> [addnl]]
           [-cat <src>]
           [-text <src>]
           [-copyToLocal [-ignoreCrc] [-crc] <src> <localdst>]
           [-moveToLocal [-crc] <src> <localdst>]
```

File system shell `fs`

```
            [-moveToLocal [-crc] <src> <localdst>]
            [-mkdir <path>]
            [-setrep [-R] [-w] <rep> <path/file>]
            [-touchz <path>]
            [-test -[ezd] <path>]
            [-stat [format] <path>]
            [-tail [-f] <file>]
            [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
            [-chown [-R] [OWNER][:[GROUP]] PATH...]
            [-chgrp [-R] GROUP PATH...]
            [-help [cmd]]
Generic options supported are
-conf <configuration file>     specify an application configuration file
-D <property=value>            use value for given property
-fs <local|namenode:port>      specify a namenode
-jt <local|jobtracker:port>    specify a job tracker
-files <comma separated list of files>    specify comma separated files to
be copied to the map reduce cluster
-libjars <comma separated list of jars>    specify comma separated jar
files to include in the classpath.
```

Create new User, User Directory

- In order to create a new Linux user account `chuck` logged in as user `root`, or type; `$su -` and enter `root`'s password. Then, type:

```
$ useradd -g mapred chuck
```

- The above will create new account/user `chuck`, as a member of group `mapred`. Please note that a user running MapReduce programs must be a member of `mapred` group. To create password for new user, type:

```
$ passwd chuck
```

- At the `New password:` prompt , enter a password for user `chuck`, press [Enter].
- At the `Retype new password:` prompt, enter the same password to confirm.
- Next create HDFS home directory for a new MapReduce user, e.g. `chuck`. By the way on a truly distributed cluster you will do all of this on the NameNode. Type:

```
$ sudo -u hdfs hadoop fs -mkdir /user/chuck
$ sudo -u hdfs hadoop fs -chown chuck /user/chuck
```

- Alternatively, if the Linux user already exist, you can login as that user and create the home directory as follows:

```
$ sudo -u hdfs hadoop fs -mkdir /user/$USER
$ sudo -u hdfs hadoop fs -chown $USER /user/$USER
```

- To remove a Linux user type:

```
$ sudo userdel chuck
```

- We should create HDFS directories for user `cloudera` as well.

Running a MapReduce Example on YARN

- Login as user `chuck` or switch the account `chuck` by typing:

```
$ su - chuck
Password: ********
```

- Subsequently make a directory in HDFS called `input` and copy some XML files into it by running the following commands:

```
$ hadoop fs -mkdir input
$ hadoop fs -put /etc/hadoop/conf/*.xml input
$ hadoop fs -ls input
-rw-r--r--   1 chuck mapred  1458 2016-02-05 10:33 input/core-site.xml
-rw-r--r--   1 chuck mapred  1875 2016-02-05 10:33 input/hdfs-site.xml
-rw-r--r--   1 chuck mapred  1549 2016-02-05 10:33 input/mapred-site.xml
-rw-r--r--   1 chuck mapred  2361 2016-02-05 10:33 input/yarn-site.xml
```

- Set `HADOOP_MAPRED_HOME` for user `chuck`. Best, enter it into `.bash_profile` file.

```
$ export HADOOP_MAPRED_HOME=/usr/lib/hadoop-mapreduce
```

Running an example on YARN

- As user `chuck`, run Hadoop `grep` example job with a regular expression on files in the `input` directory:

```
$ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar grep input output23 'dfs[a-z.]+'
```

- After a lot of console output, the job completes.
- We can find the output in the HDFS directory named `output23` because we specified that output directory to Hadoop. Type:

```
$ hadoop fs -ls
```

- You will see directories `input` and `output23`. We list the content of `output23`

```
$ hadoop fs -ls output23
Found 2 items
-rw-r--r-- 1 joe supergroup 1068 2016-02-05 10:33 output23/part-r-00000
-rw-r--r- 1 joe supergroup 0 2016-02-05 10:33     output23/_SUCCESS
```

- The content of the output file `part-00000` can be seen using `fs -cat` command:

```
$ hadoop fs -cat output23/part-r-00000 | head
1 dfs.datanode.data.dir
1 dfs.namenode.checkpoint.dir
1 dfs.namenode.name.dir
1 dfs.replication
1 dfs.safemode.extension
1 dfs.safemode.min.datanodes
```

Removing YARN CDH5 Installation

- Stop the daemons:

```
$ for x in `cd /etc/init.d ; ls hadoop-hdfs-*`
  do sudo service $x stop
  done
$ for x in 'cd /etc/init.d ; ls hadoop-mapreduce-* ;
  do sudo service $x stop
  done
```

- Remove hadoop-conf-pseudo:

- On Red Hat-compatible systems:

```
$ sudo yum remove hadoop-conf-pseudo hadoop-mapreduce-*
```

Create new Linux user `joe`

- First create a Linux user group and assign to it a name and group id (gid). Group id has to be a number greater than 500.
- To see existing group ids, do `$ cat /etc/group` .
- To create new group `hadoopusers` with `group id 505`, type:

```
$ groupadd hadoopusers - - gid 505
```

- Next add a user `joe` who is a member of that group

```
$ useradd -u 504 -g 505 -m -s /bin/bash joe
```

- Check that `joe is` created

```
$ cat /etc/passwd | grep joe
joe:x:504:505::/home/joe:/bin/bash
```

- Add a password to `joe`

```
$ passwd joe
New password:xxxxxxxxx
Retype new password:xxxxxxxxx
```

- Add `joe` to sudo users, just like you did it for user `clodera`.
- Switch the user. Become `joe`

```
# su - joe
[joe@localhost root]$
```

References

- Hadoop the Definitive Guide, 3rd Edition, by Tom White, O'Reilly 2012
- Hadoop in Action, by Chuck Lam, Manning 2011
- Hadoop in Practice, by Alex Holmes, Manning 2012
- Cloudera, CDH5 Quick Start Guide