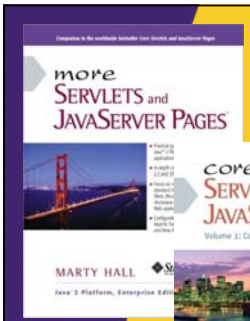
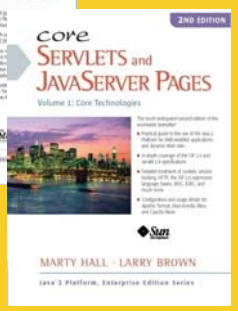
Map Reduce on YARN Overview

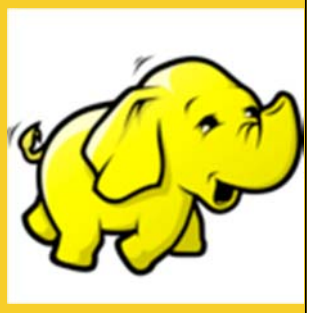
Originals of slides and source code for examples: http://www.coreservlets.com/hadoop-tutorial/
Also see the customized Hadoop training courses (onsite or at public venues) – http://courses.coreservlets.com/hadoop-training.html

Customized Java EE Training: http://courses.coreservlets.com/
Hadoop, Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Android.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

© 2012 coreservlets.com and Dima May







For live customized Hadoop training (including prep for the Cloudera certification exam), please email info@coreservlets.com

Taught by recognized Hadoop expert who spoke on Hadoop several times at JavaOne, and who uses Hadoop daily in real-world apps. Available at public venues, or customized versions can be held on-site at your organization.

- Courses developed and taught by Marty Hall
  - JSF 2.2, PrimeFaces, servlets/JSP, Ajax, jQuery, Android development, Java 7 or 8 programming, custom mix of topics
  - Courses available in any state or country. Maryland/DC area companies can also choose afternoon/evening courses.
- Courses developed and taught by coreservlets.com experts (edited by Marty)
  - Spring, Hibernate/JPA, GWT, Hadoop, HTML5, RESTful Web Services

Contact info@coreservlets.com for details

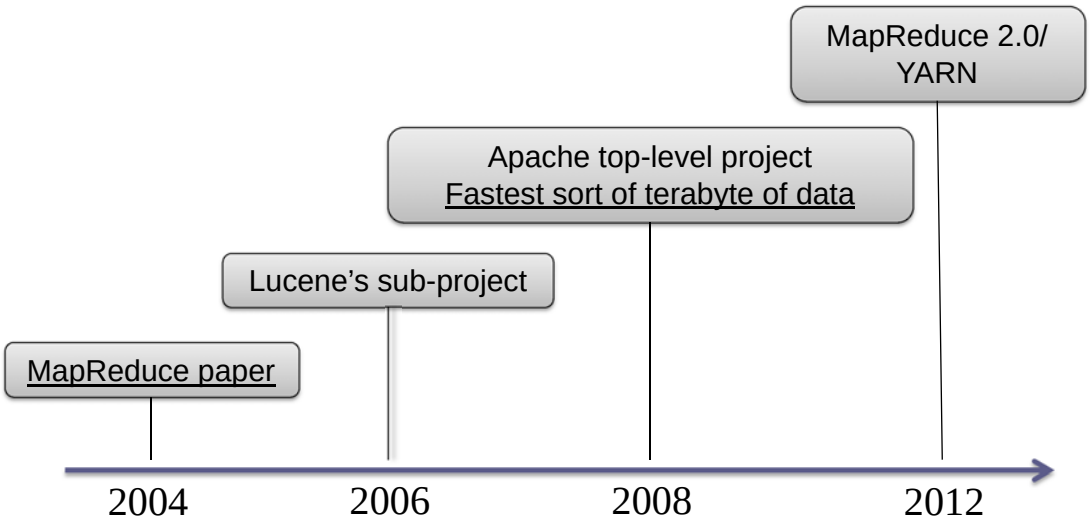
Agenda

- MapReduce Introduction
- MapReduce Model
- YARN and MapReduce 2.0 Daemons
- MapReduce on YARN single node installation
- MapReduce and YARN command line tools

Hadoop MapReduce

- Model for processing large amounts of data in parallel
  - On commodity hardware
  - Lots of nodes
- Derived from functional programming
  - Map and reduce functions
- Can be implemented in multiple languages
  - Java, C++, Ruby, Python (etc...)

Hadoop MapReduce History



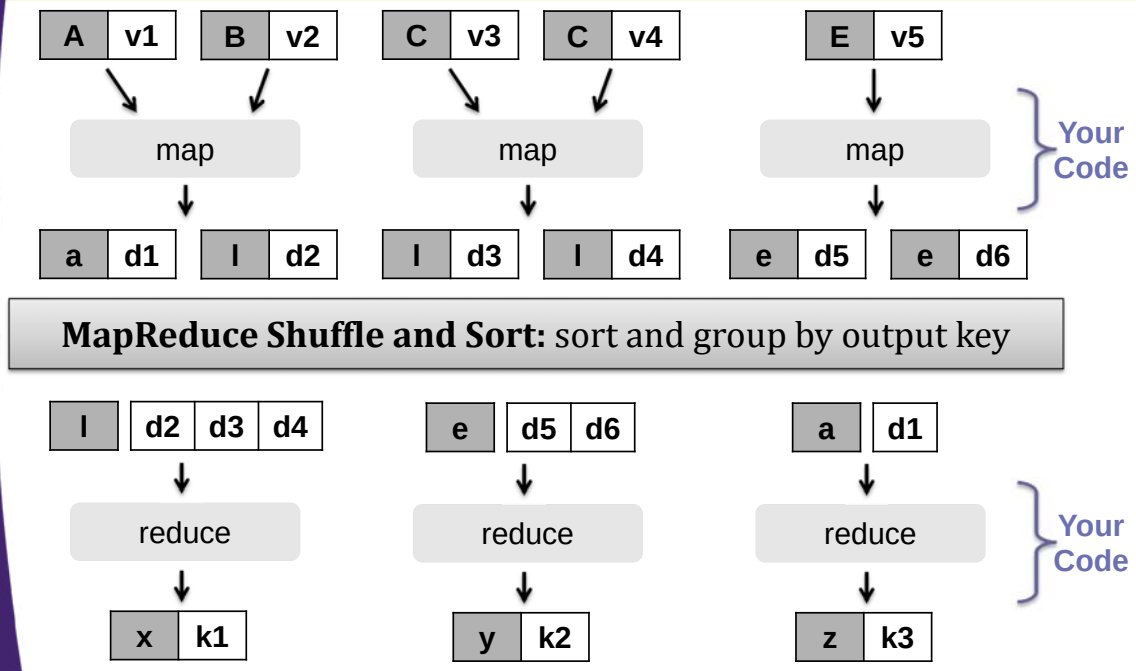
MapReduce Model

- **Imposes key-value input/output**
- **Defines map and reduce functions**

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

1. Map function is applied to every input key-value pair
2. Map function generates intermediate key-value pairs
3. Intermediate key-values are sorted and grouped by key
4. Reduce is applied to sorted and grouped intermediate key-values
5. Reduce emits result key-values

MapReduce Model/Framework



Source: Jimmy Lin, Chris Dyer. Data-Intensive Text Processing with MapReduce. Morgan & Claypool. 2010

MapReduce Framework

- **Takes care of distributed processing and coordination**
- **Scheduling**
  - Jobs are broken down into smaller chunks called tasks. These tasks are scheduled
- **Task Localization with Data**
  - Framework strives to place tasks on the nodes that host the segment of data to be processed by that specific task
  - Code is moved to where the data is

Source: Jimmy Lin, Chris Dyer. Data-Intensive Text Processing with MapReduce. Morgan & Claypool. 2010

MapReduce Framework

- **Error Handling**
  - Failures are an expected behavior so tasks are automatically re-tried on other machines
- **Data Synchronization**
  - Shuffle and Sort barrier re-arranges and moves data between machines
  - Input and output are coordinated by the framework

Map Reduce 2.0 on YARN

- **Yet Another Resource Negotiator (YARN)**
- **Various applications can run on YARN**
  - MapReduce is just one choice (the main choice at this point)
  - http://wiki.apache.org/hadoop/PoweredByYarn
- **YARN was designed to address issues with MapReduce1**
  - Scalability issues (max ~4,000 machines)
  - Inflexible Resource Management
    - MapReduce1 had slot based model

MapReduce1 vs. YARN

- **MapReduce1 runs on top of JobTracker and TaskTracker daemons**
  - JobTracker schedules tasks, matches task with TaskTrackers
  - JobTracker manages MapReduce Jobs, monitors progress
  - JobTracker recovers from errors, restarts failed and slow tasks
- **MapReduce1 has inflexible slot-based memory management model**
  - Each TaskTracker is configured at start-up to have N slots
  - A task is executed in a single slot
  - Slots are configured with maximum memory on cluster start-up
  - The model is likely to cause over and under utilization issues

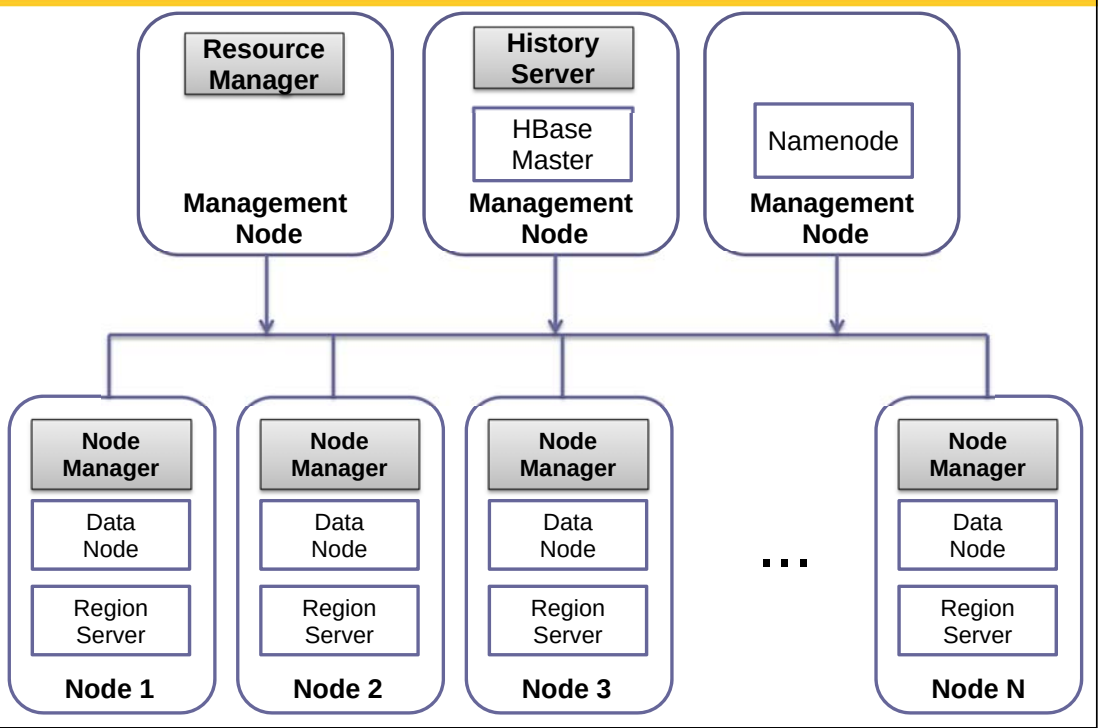
MapReduce1 vs. YARN

- **YARN addresses shortcomings of MapReduce1**
  - JobTracker is split into 2 daemons
    - ResourceManager - administers resources on the cluster
    - ApplicationMaster - manages applications such as MapReduce
  - Fine-Grained memory management model
    - ApplicationMaster requests resources by asking for “containers” with a certain memory limit (ex 2G)
    - YARN administers these containers and enforces memory usage
    - Each Application/Job has control of how much memory to request

Daemons

- **YARN Daemons**
  - Node Manger
    - Manages resources of a single node
    - There is one instance per node in the cluster
  - Resource Manager
    - Manages Resources for a Cluster
    - Instructs Node Manager to allocate resources
    - Application negotiates for resources with Resource Manager
    - There is only one instance of Resource Manager
- **MapReduce Specific Daemon**
  - MapReduce History Server
    - Archives Jobs' metrics and meta-data

Sample YARN Daemons Deployments with HDFS and HBase



Old vs. New Java API

- **There are two flavors of MapReduce API which became known as Old and New**
- **Old API classes reside under**
  - org.apache.hadoop.mapred
- **New API classes can be found under**
  - org.apache.hadoop.mapreduce
  - org.apache.hadoop.mapreduce.lib
- **We will use new API exclusively**
- **New API was re-designed for easier evolution**
- **Early Hadoop versions deprecated old API but recently deprecation was removed**
- **Do not mix new and old API**

YARN with MapReduce 2.0 Installation

- **Single Node Installation**
  - Mimics Fully-Distributed Cluster but runs on just one host
  - Good for testing, debugging and prototyping
  - Not production use or performance benchmarking!
  - Used in this class
- **Cluster**
  - Run YARN on many nodes
  - Great for production and development clusters

Set Up Pseudo-Distributed Mode

1. **Verify Prerequisites**
   - Java, SSH, etc...
2. **Configure YARN**
   - Settings inside of yarn-site.xml and yarn-env.sh
3. **Configure MapReduce**
   - Settings inside of mapred-site.xml
4. **Start YARN and MapReduce History Server**

1: Verify Prerequisites

- **Java version 6+**
  - It is written in Java after all
  - Oracle JDK is the most popular

- **Linux is the only production platform**
- **Supports all other Operating Systems for development**
  - Windows, OS X
  - Cygwin is required for Windows development
- **Password-less SSH is setup**
  - Already had to do this for HDFS

2: Configure YARN

- **Configuration Files are discovered by looking in the directory specified in YARN_CONF_DIR environment property**
  - export YARN_CONF_DIR=$HADOOP_CONF_DIR

2: Configure YARN

Config File	Description
yarn-env.sh	A bash script where YARN environment variables are specified. For example, configure log directory here.
yarn-site.xml	Hadoop configuration file where majority of properties are specified for YARN daemons. Configures Resource Manager, Node Manager and History Server.
slaves	A list of nodes where Node Manager daemons are started; one host per line.
mapred-site.xml	MapReduce specific properties go here. This is the application specific configuration file; an application is MapReduce in this case.

Note: YARN will also utilize core-site.xml and hadoop-env.sh which were covered in HDFS lecture

2: Configure YARN - yarn-env.sh

- **Bash Script executed by YARN start script**
- **Sets things like daemons'**
  - heap size
  - log location(s)
  - pid file locations
  - etc...

```
export YARN_LOG_DIR=/home/hadoop/Training/logs/yarn
export YARN_PID_DIR=$TRAINING_HOME/hadoop_work/pids
```


2: Configure YARN - yarn-site.xml

- **Configure host and ports for daemons, data output locations, auxiliary services, security, etc....**
  - Configure hosts and ports for Resource Manager, Node Manager and MapReduce History Server
  - Configure local storage directories for Node Managers
  - Configure any application specific services

2: Configure YARN - yarn-site.xml

```
<property>
  <name>yarn.resourcemanager.address</name>
  <value>localhost:10040</value>
  <description>In Server specified the port that Resource
Manager will run on. In client is used for connecting to
Resource Manager</description>
</property>

<property>
  <name>yarn.resourcemanager.resource-tracker.address</name>
  <value>localhost:8025</value>
  <description>Utilized by Node Manager for communication
with Resource Manager</description>
</property>
```


2: Configure YARN - yarn-site.xml

```
<property>
  <name>yarn.resourcemanager.scheduler.address</name>
  <value>localhost:8030</value>
  <description>Utilized by Application Masters to
communicate with Resource Manager; in our case for
MRAppMaster (MapReduce Application Master) to communicate
with Resource Manager</description>
</property>

<property>
  <name>yarn.resourcemanager.admin.address</name>
  <value>localhost:8141</value>
  <description>Utilized by administrative clients ($yarn
rmadmin) to communicate with Resource Manager</description>
</property>
```

2: Configure YARN - yarn-site.xml

```
<property>
   <name>yarn.nodemanager.local-dirs</name>
   <value>/home/hadoop/Training/hadoop_work/mapred/nodemanager</value>
   <final>true</final>
   <description>Comma separated list of directories, where
local data is persisted by Node Manager</description>
</property>

<property>
   <name>yarn.nodemanager.aux-services</name>
   <value>mapreduce.shuffle</value>
   <description>Long running service which executes on Node
Manager(s) and provides MapReduce Sort and Shuffle
functionality.</description>
</property>
```


2: Configure YARN - yarn-site.xml

```
<property>
  <name>yarn.log-aggregation-enable</name>
  <value>true</value>
  <description>Enable log aggregation so application logs are
moved onto hdfs and are viewable via web UI after the
application completed. The default location on hdfs is '/log'
and can be changed via yarn.nodemanager.remote-app-log-dir
property</description>
</property>

<property>
  <name>hadoop.security.authorization</name>
  <value>false</value>
  <description>Disable authorization for development and
clusters that do not require security</description>
</property>
```

3: Configure MapReduce - mapred-site.xml

```
<configuration>
   <property>
    <name>mapreduce.cluster.temp.dir</name>
    <value>/home/hadoop/Training/hadoop_work/mapred/temp</value>
    <final>true</final>
  </property>

  <property>
    <name>mapreduce.cluster.local.dir</name>
    <value>/home/hadoop/Training/hadoop_work/mapred/local</value>
    <final>true</final>
  </property>

  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
    <description>Use YARN as the servicer of MapReduce, if
not specified Local Job Runner is used</description>
  </property>
</configuration>
```


4: Start YARN and MapReduce History Server

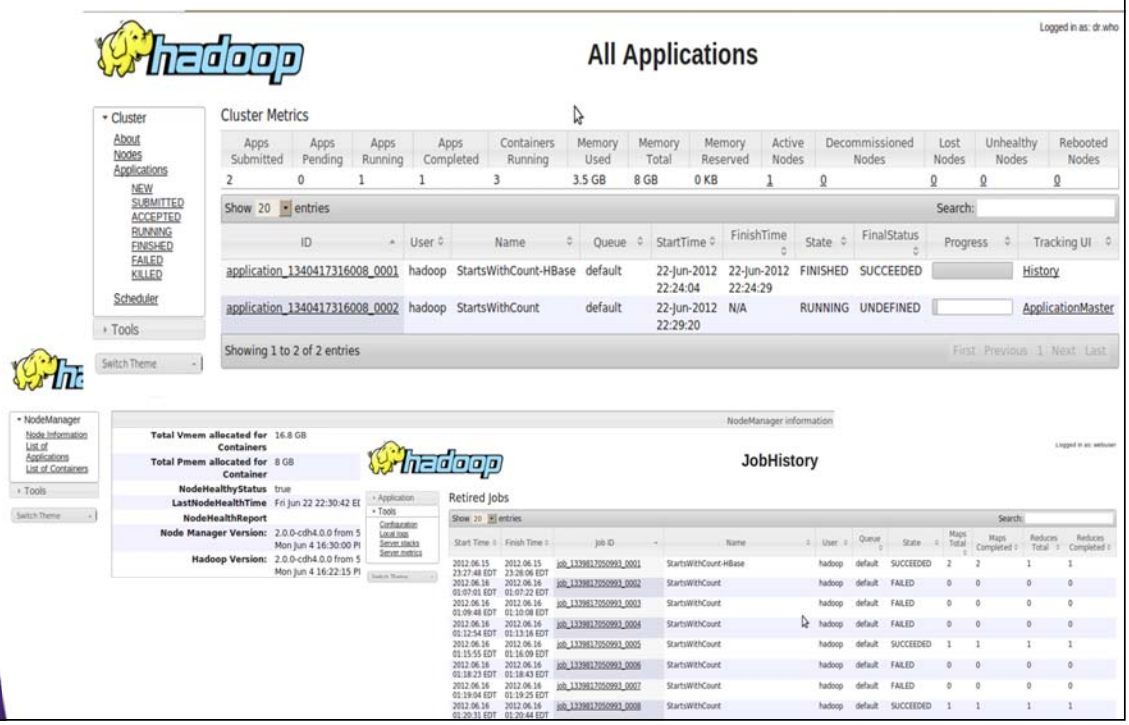
```
$ cd $HADOOP_HOME/sbin
$ ./start-yarn.sh
$ ./mr-jobhistory-daemon.sh start historyserver
```

- **start-yarn.sh script starts all the YARN related daemons on the cluster**
  - Resource Manager is started on the machine that start-yarn.sh script is executed on
    - Delegates to $yarn-daemon.sh start resourcemanager
  - Node Manager is started for each machine specified in the <YARN_CONF_DIR>/slaves
    - Delegates to $yarn-daemon.sh start nodemanager

YARN Web-UI

- **Resource Manager Web-UI**
  - Cluster resource usage, job scheduling, and current running jobs
  - Runs on port 8088 by default
- **Application Proxy Web-UI**
  - Provides information about the current job
  - Runs as a part of Resource Manager Web-UI by default
  - After completion, jobs get exposed by History Server
- **Node Manager Web-UI**
  - Single Node information and current containers being executed
  - Runs on port 8042 by default
- **MapReduce History Server Web-UI**
  - Provides history and details of past MapReduce jobs
  - Runs on port 19888 by default

MapReduce 2.0 Web-UI



Command Line Tools

- **<hadop_install>/bin/yarn**
  - Execute code with a jar
  - $yarn jar jarFile [mainClass] args...
  - Print out CLASSPATH: $yarn classpath
  - Resource Manager admin: $yarn rmadmin
- **<hadop_install>/bin/mapred**
  - $mapred job
    - Get information about jobs
    - Kill Jobs

$ yarn jar jarFile [mainClass] args...

- **Execute code in the provided jar**

```
$ yarn jar
$HADOOP_HOME/share/hadoop/mapreduce/hadoop-
mapreduce-examples-2.0.0-cdh4.0.0.jar pi 5 5
```

Examples jar files shipped with hadoop

pi is the program that computes pi

Specify number of mappers

Number of samples; artifact of pi application

$ yarn rmadmin

- **Runs ResourceManager admin client**
- **Allows to refresh and clear resources**

```
$ yarn rmadmin -refreshNodes
```

Resource Manager will refresh its information about all the Node Managers

$mapred job

- **Command line interface to view job's attributes**
- **Most of the information is available on Web-UI**

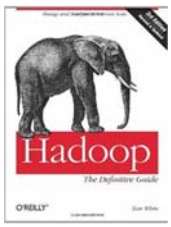
List Jobs that are currently running

```
$ mapred job -list

$ mapred job -status job_1340417316008_0001
```

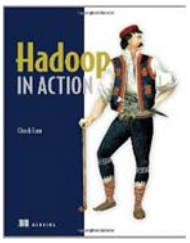
Retrieve job's status by Job ID

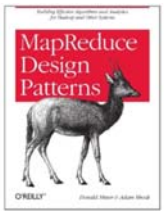
Resources: Books



Hadoop: The Definitive Guide
Tom White (Author)
O'Reilly Media; 3rd Edition (May6, 2012)

Hadoop in Action
Chuck Lam (Author)
Manning Publications; 1st Edition (December, 2010)



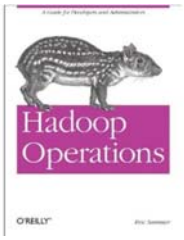


MapReduce Design Patterns
Donald Miner (Author), Adam Shook (Author)
O'Reilly Media (November 22, 2012)

Resources: Books

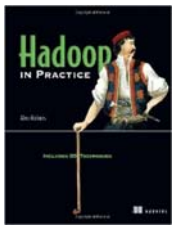
HBase: The Definitive Guide
Lars George (Author)
O'Reilly Media; 1 edition (September 20, 2011)

Hadoop Operations
Eric Sammer (Author)
O'Reilly Media (October 22, 2012)



Data-Intensive Text Processing with MapReduce
Jimmy Lin and Chris Dyer (Authors) (April, 2010)
Download for FREE:
http://lintool.github.com/MapReduceAlgorithms/index.html

Resources: Books



Hadoop in Practice
Alex Holmes (Author)
Manning Publications; (October 10, 2012)

Resources

- **Home Page**
  - http://hadoop.apache.org
- **Mailing Lists**
  - http://hadoop.apache.org/mailing_lists.html
- **Wiki**
  - http://wiki.apache.org/hadoop
- **Documentation:**
  - http://hadoop.apache.org/docs/r2.0.2-alpha

Wrap-Up

Summary

- **We learned about**
  - MapReduce Model
  - YARN/MapReduce 2.0 Daemons
  - MapReduce on YARN single node installation
  - MapReduce and YARN command line tools

Questions?

More info:

http://www.coreservlets.com/hadoop-tutorial/ – Hadoop programming tutorial

http://courses.coreservlets.com/hadoop-training.html – Customized Hadoop training courses, at public venues or onsite at *your* organization

http://courses.coreservlets.com/Course-Materials/java.html – General Java programming tutorial

http://www.coreservlets.com/java-8-tutorial/ – Java 8 tutorial

http://www.coreservlets.com/JSF-Tutorial/jsf2/ – JSF 2.2 tutorial

http://www.coreservlets.com/JSF-Tutorial/primefaces/ – PrimeFaces tutorial

http://coreservlets.com/ – JSF 2, PrimeFaces, Java 7 or 8, Ajax, jQuery, Hadoop, RESTful Web Services, Android, HTML5, Spring, Hibernate, Servlets, JSP, GWT, and other Java EE training