HDFS Installation and Shell

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.

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

• Pseudo-Distributed Installation
• Namenode Safemode
• Secondary Namenode
• Hadoop Filesystem Shell

Installation - Prerequisites

• JavaTM 1.6.x
  – From Oracle (previously Sun Microsystems)
• SSH installed, sshd must be running
  – Used by Hadoop scripts for management
• Cygwin for windows shell support
Installation

• Three options
  – Local (Standalone) Mode
  – Pseudo-Distributed Mode
  – Fully-Distributed Mode

Installation: Local

• Default configuration after the download
• Executes as a single Java process
• Works directly with local filesystem
• Useful for debugging
• Simple example, list all the files under /
  – $ cd <hadoop_install>/bin
  – $ hdfs dfs -ls /
Installation: Pseudo-Distributed

- Still runs on a single node
- Each daemon runs in its own Java process
  - Namenode
  - Secondary Namenode
  - Datanode
- Location for configuration files is specified via `HADOOP_CONF_DIR` environment property
- Configuration files
  - `core-site.xml`
  - `hdfs-site.xml`
  - `hadoop-env.sh`

Installation: Pseudo-Distributed

- `hadoop-env.sh`
  - Specify environment variables
    - Java and Log locations
  - Utilized by scripts that execute and manage hadoop

```bash
export TRAINING_HOME=/home/hadoop/Training
export JAVA_HOME=$TRAINING_HOME/jdk1.6.0_29
export HADOOP_LOG_DIR=$TRAINING_HOME/logs/hdfs
```
Installation: Pseudo-Distributed

• $HADOOP_CONF_DIR/core-site.xml
  – Configurations for core of Hadoop, for example IO properties
  
• Specify location of Namenode

```xml
<property>
  <name>fs.default.name</name>
  <value>hdfs://localhost:8020</value>
  <description>NameNode URI</description>
</property>
```

Installation: Pseudo-Distributed

• $HADOOP_CONF_DIR/hdfs-site.xml
  – Configurations for Namenode, Datanode and Secondary Namenode daemons

```xml
<property>
  <name>dfs.namenode.name.dir</name>
  <value>/home/hadoop/Training/hadoop_work/data/name</value>
  <description>Path on the local filesystem where the NameNode stores the namespace and transactions logs persistently.</description>
</property>

<property>
  <name>dfs.datanode.data.dir</name>
  <value>/home/hadoop/Training/hadoop_work/data/data</value>
  <description>Comma separated list of paths on the local filesystem of a Datanode where it should store its blocks.</description>
</property>
```
Installation: Pseudo-Distributed

- **$HADOOP_CONF_DIR/hdfs-site.xml**

  ```xml
  <property>
    <name>dfs.namenode.checkpoint.dir</name>
    <value>/home/hadoop/Training/hadoop_work/data/secondary_name</value>
    <description>Determines where on the local filesystem the DFS secondary name node should store the temporary images to merge. If this is a comma-delimited list of directories then the image is replicated in all of the directories for redundancy.
    </description>
  </property>

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

Installation: Pseudo-Distributed

- **$HADOOP_CONF_DIR/slaves**
  - Specifies which machines Datanodes will run on
  - One node per line

- **$HADOOP_CONF_DIR/masters**
  - Specifies which machines Secondary Namenode will run on
  - Misleading name
Installation: Pseudo-Distributed

• Password-less SSH is required for Namenode to communicate with Datanodes
• In this case just to itself
• To test:
  – $ ssh localhost
• To set-up
  – $ ssh-keygen -t dsa -P " -f ~/.ssh/id_dsa
  – $ cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys

• Prepare filesystem for use by formatting
  – $ hdfs namenode -format
• Start the distributed filesystem
  – $ cd <hadoop_install>/sbin
  – $ start-dfs.sh
• start-dfs.sh prints the location of the logs

$ ./start-dfs.sh
Starting namenodes on [localhost]
localhost: starting namenode, logging to /home/hadoop/Training/logs/hdfs/hadoop-
hadoop-namenode-hadoop-laptop.out
localhost: 2012-07-17 22:17:17,054 INFO  namenode.NameNode
(StringUtils.java:startupShutdownMessage(594)) - STARTUP_MSG:
localhost: ******************************************
localhost: STARTUP_MSG: Starting NameNode
...
Installation: logs

- Each Hadoop daemon writes a log file:
  - Namenode, Datanode, Secondary Namenode
  - Location of these logs are set in $HADOOP_CONF_DIR/hadoop-env.sh
  - export HADOOP_LOG_DIR=$TRAINING_HOME/logs/hdfs

- Log naming convention:

  hadoop-dima-namenode-hadoop-laptop.out

  product, username, daemon, hostname

Installation: logs

- Log locations are set in $HADOOP_CONF_DIR/hadoop-env.sh
  - Specified via $HADOOP_LOG_DIR property
  - Default is <install_dir>/logs
  - It's a good practice to configure log directory to reside away from the installation directory

  export TRAINING_HOME=/home/hadoop/Training
  export HADOOP_LOG_DIR=$TRAINING_HOME/logs/hdfs
Management Web Interface

• Namensnode comes with web based management
  – http://localhost:50070
• Features
  – Cluster status
  – View Namensnode and Datanode logs
  – Browse HDFS
• Can be configured for SSL (https:) based access
• Secondary Namensnode also has web UI
  – http://localhost:50090

Management Web Interface

• Datanodes run management web server also
• Browsing Namensnode will re-direct to Datanodes' Web Interface
• Firewall considerations
  – Opening <namenode_host>:50070 in firewall is not enough
  – Must open up <datanode(s)_host>:50075 on every datanode host
  – Best scenario is to open the browser behind firewall
    • SSH tunneling, Virtual Network Computing (VNC), X11, etc..
  – Can be SSL enabled
Management Web Interface

Namenode's Safemode

- HDFS cluster read-only mode
- Modifications to filesystem and blocks are not allowed
- Happens on start-up
  - Loads file system state from fsimage and edits-log files
  - Waits for Datanodes to come up to avoid over-replication
- Namenode's Web Interface reports safemode status
- Could be placed in safemode explicitly
  - for upgrades, maintenance, backups, etc....
Secondary Namenode

- Namenode stores its state on local/native file-system mainly in two files: edits and fsimage
  - Stored in a directory configured via dfs.name.dir property in hdfs-site.xml
  - edits: log file where all filesystem modifications are appended
  - fsimage: on start-up namenode reads hdfs state, then merges edits file into fsimage and starts normal operations with empty edits file
- Namenode start-up merges will become slower over time but ...
  - Secondary Namenode to the rescue

Secondary Namenode

- Secondary Namenode is a separate process
  - Responsible for merging edits and fsimage file to limit the size of edits file
  - Usually runs on a different machine than Namenode
  - Memory requirements are very similar to Namenode’s
  - Automatically started via start-dfs.sh script
Secondary Namenode

• Checkpoint is kicked off by two properties in hdfs-site.xml
  – `fs.checkpoint.period`: maximum time period between two checkpoints
    • Default is 1 hour
    • Specified in seconds (3600)
  – `fs.checkpoint.size`: when the size of the edits file exceeds this threshold a checkpoint is kicked off
    • Default is 64 MB
    • Specified in bytes (67108864)

Secondary Namenode

• Secondary Namenode uses the same directory structure as Namenode
  – This checkpoint may be imported if Namenode's image is lost

• Secondary Namenode is NOT
  – Fail-over for Namenode
  – Doesn't provide high availability
  – Doesn't improve Namenode's performance
Shell Commands

- Interact with FileSystem by executing shell-like commands
- Usage: $hdfs dfs -<command> -<option> <URI>
  - Example $hdfs dfs -ls /
- URI usage:
  - HDFS: $hdfs dfs -ls hdfs://localhost/to/path/dir
  - Local: $hdfs dfs -ls file:///to/path/file3
  - Schema and namenode host is optional, default is used from the configuration
    - In core-site.xml - fs.default.name property

Hadoop URI

scheme://authority/path

hdfs://localhost:8020/user/home

Scheme and authority determine which file system implementation to use. In this case it will be HDFS
Path on the file system
Shell Commands

- Most commands behave like UNIX commands
  - ls, cat, du, etc..
- Supports HDFS specific operations
  - Ex: changing replication
- List supported commands
  - $ hdfs dfs -help
- Display detailed help for a command
  - $ hdfs dfs -help <command_name>

Shell Commands

- Relative Path
  - Is always relative to user's home directory
  - Home directory is at /user/<username>
- Shell commands follow the same format:
  $ hdfs dfs -<command> -<option> <path>
- For example:
  - $ hdfs dfs -rm -r /removeMe
Shell Basic Commands

- **cat – stream source to stdout**
  - entire file: $hdfs dfs -cat /dir/file.txt
  - Almost always a good idea to pipe to head, tail, more or less
  - Get the first 25 lines of file.txt
    - $hdfs dfs -cat /dir/file.txt | head -n 25
- **cp – copy files from source to destination**
  - $hdfs dfs -cp /dir/file1 /otherDir/file2
- **ls – for a file displays stats, for a directory displays immediate children**
  - $hdfs dfs -ls /dir/
- **mkdir – create a directory**
  - $hdfs dfs -mkdir /brandNewDir

Moving Data with Shell

- **mv – move from source to destination**
  - $hdfs dfs -mv /dir/file1 /dir2/file2
- **put – copy file from local filesystem to hdfs**
  - $hdfs dfs -put localfile /dir/file1
  - Can also use copyFromLocal
- **get – copy file to the local filesystem**
  - $hdfs dfs -get /dir/file localfile
  - Can also use copyToLocal
Deleting Data with Shell

- `rm` – delete files
  - `$hdfs dfs -rm /dir/fileToDelete`
- `rm -r` – delete directories recursively
  - `$hdfs dfs -rm -r /dirWithStuff`

Filesystem Stats with Shell

- `du` – displays length for each file/dir (in bytes)
  - `$hdfs dfs -du /someDir/`
- **Add `-h` option to display in human-readable format instead of bytes**
  - `$hdfs dfs -du -h /someDir$
    - 206.3k /someDir`
Learn More About Shell

- More commands
  - tail, chmod, count, touchz, test, etc...
- To learn more
  - $hdfs dfs -help
  - $hdfs dfs -help <command>

- For example:
  - $ hdfs dfs -help rm

fsck Command

- Check for inconsistencies
- Reports problems
  - Missing blocks
  - Under-replicated blocks
- Doesn't correct problems, just reports (unlike native fsck)
  - Namenode attempts to automatically correct issues that fsck would report
- $ hdfs fsck <path>
  - Example : $ hdfs fsck /
HDFS Permissions

• Limited to File permission
  – Similar to POSIX model, each file/directory
  – has Read (r), Write (w) and Execute (x)
  – associated with owner, group or all others

• Client's identity determined on host OS
  – Username = `whoami`
  – Group = `bash -c groups`

HDFS Permissions

• Authentication and Authorization with Kerberos
  – Hadoop 0.20.20+
  – Earlier versions assumed private clouds with trusted users
  – Hadoop set-up with Kerberos is beyond the scope of this class

• To learn about Hadoop and Kerberos
  – http://hadoop.apache.org/common/docs/r0.23.0/hadoop-yarn/hadoop-yarn-site/ClusterSetup.html
  – CDH4 and Keberos:
    - "Hadoop: The Definitive Guide" by Tom White
DFSAdmin Command

- HDFS administrative operations
  - `$hdfs dfsadmin <command>`
  - Example: `$hdfs dfsadmin -report`
- `-report`: displays statistic about HDFS
  - Some of these stats are available on Web Interface
- `-safemode`: enter or leave safemode
  - Maintenance, backups, upgrades, etc..

Rebalancer

- Data on HDFS Clusters may not be uniformly spread between available Datanodes.
  - Ex: New nodes will have significantly less data for some time
  - The location for new incoming blocks will be chosen based on status of Datanode topology, but the cluster doesn't automatically rebalance
- Rebalancer is an administrative tool that analyzes block placement on the HDFS cluster and re-balances
  - `$ hdfs balancer`
Wrap-Up

Summary

• We learned about
  – Pseudo-Distributed Installation
  – Namenode Safemode
  – Secondary Namenode
  – Hadoop FS Shell
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/JSF-2.2.xu/ – JSF 2.2.xu
http://www.coreservlets.com/JSF-Tutorial/PrimeFaces/PrimeFaces.html
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

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.