MapReduce/Hadoop

A Short Setup Guide

Introduction

- Hadoop could be configured to run in three modes:
    - Standalone mode (single node cluster)
    - Pseudo-distributed mode (single node cluster)
    - Fully-distributed mode (multi node cluster)
- We use "Pseudo Distributed Mode" in this assignment.
    - Each Hadoop daemon runs in a separate Java process;
    - Using different port to simulate different node;

Requirements

- GNU/Linux (Debian/Ubuntu/Fedora...)
- JavaTM
- SSH
- Hadoop

Step 1: GNU/Linux

- Hadoop runs on GNU/Linux or on Windows
  - GNU/Linux is prefered;
  - Windows + Cygwin (including *openssh*);
- If you do not have a working GNU/Linux environment:
  - Choose one of the Installation type (using Ubuntu for example):
    - Desktop CD Installation
      - http://www.ubuntu.com/download/help/install-desktop-latest
    - Virtualbox Installation
      - http://www.psychocats.net/ubuntu/virtualbox
    - Inside Windows Installation
      - http://www.psychocats.net/ubuntu/wubi
- We assume you are using Ubuntu (Debian-based GNU/Linux) in the following slides.

Step 2: JavaTM

- Java 1.6+ (either Sun Java or Open Java) is recommended for Hadoop.
  - Sun Java is prefered;
  - OpenJDK is handy in most of the GNU/Linux distributions
    - Also applicable in this assignment;
  - http://wiki.apache.org/hadoop/HadoopJavaVersions;
- Installation:

  -
    ```
    $ sudo apt-get install openjdk-7-jdk
    ```
- A quick check of Java installation:

  -
    ```
    $ java -version
    ```

Step 3: SSH

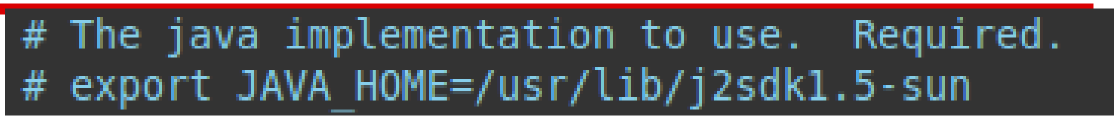
- sshd must be running to use the Hadoop scripts that manage remote Hadoop daemons.
- Installation:
  - $ sudo apt-get install ssh openssh-server openssh-client
- A quick check of ssh installation:
  - $ ssh localhost
  - If you want to SSH to localhost without password:
    - $ ssh-keygen -t dsa -P '' -f ~/.ssh/id_dsa
    - $ cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys

Step 4: Hadoop

- Download a recent stable release from:
  - http://apache.communilink.net/hadoop/common/
  - 1.2.X - current stable version, 1.2 release;
  - We use version 1.2.1 in this assignment
  - http://apache.communilink.net/hadoop/common/hadoop-1.2.1/hadoop-1.2.1.tar.gz
- We assume that the hadoop is in folder “hadoop”
  - The home directory is /home/youraccount/hadoop
- Unpack Hadoop package downloaded:
  - `$ tar xzf hadoop-1.2.1.tar.gz -C ~/hadoop/`

Step 4: Hadoop (Cont'd)

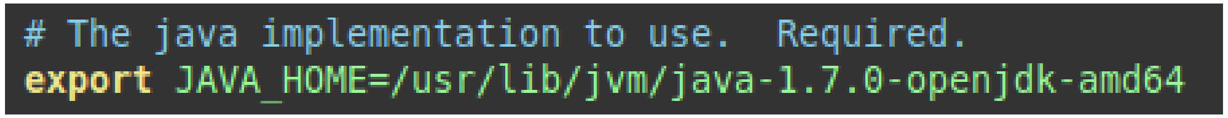
- Configure JAVA:



```
# The java implementation to use.  Required.
# export JAVA_HOME=/usr/lib/j2sdk1.5-sun
```

  - Edit file ~/hadoop/hadoop-1.2.1/conf/hadoop-env.sh
    - find the line with # export JAVA_HOME=xxxx

Change it to point to the correct location of JAVA (Depending on which system you are using):



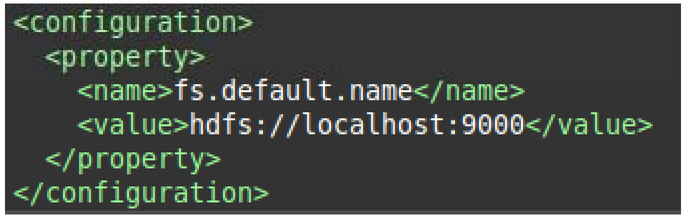
```
# The java implementation to use.  Required.
export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-amd64
```

update-alternatives --config java
JAVA_HOME is everything before /jre/bin/java

- Configure Hadoop component:
  - Each component in Hadoop is configured using an XML file;
  - Three files: core-site.xml, hdfs-site.xml, mapred-site.xml;

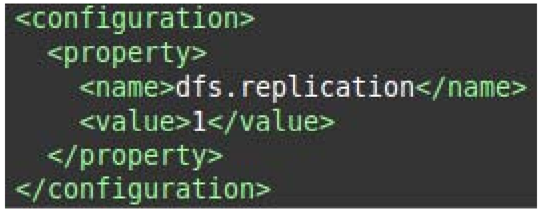
Step 4: Hadoop (Cont'd)

- Edit ~/hadoop/hadoop-1.2.1/conf/core-site.xml



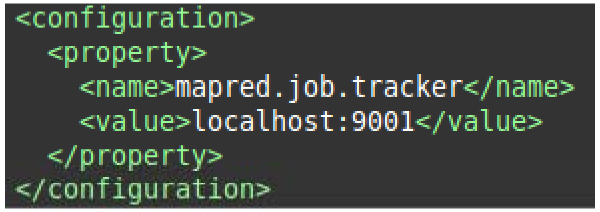
```
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

- Edit ~/hadoop/hadoop-1.2.1/conf/hdfs-site.xml



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

- Edit ~/hadoop/hadoop-1.2.1/conf/mapred-site.xml



```
<configuration>
  <property>
    <name>mapred.job.tracker</name>
    <value>localhost:9001</value>
  </property>
</configuration>
```

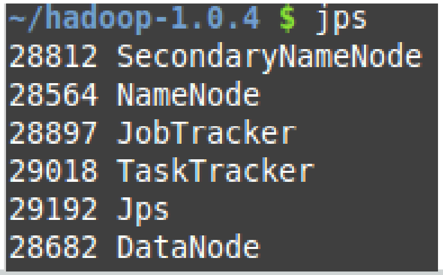
Start Hadoop

- A brand-new HDFS installation needs to be formatted before it can be used:
  - Using the binary file hadoop in bin directory
    - $ cd ~/hadoop/hadoop-1.2.1/
    - $ bin/hadoop namenode -format
- Start Hadoop:
  - Using the start-all.sh script file in bin directory
    - $ bin/start-all.sh

A quick check of Hadoop initialization:

jps - Java Virtual Machine Process Status Tool

- $ jps



```
~/hadoop-1.0.4 $ jps
28812 SecondaryNameNode
28564 NameNode
28897 JobTracker
29018 TaskTracker
29192 Jps
28682 DataNode
```

Browse the web interface

- Web based Interface for NameNode
  - http://localhost:50070
- Web based Interface for JobTracker
  - http://localhost:50030
- Web based Interface for TaskTracker
  - http://localhost:50060

HDFS shell commands

- Hadoop Shell Commands, http://hadoop.apache.org/docs/r0.18.3/hdfs_shell.html
  - Browsing Your HDFS Folder
    - $ bin/hadoop fs -ls
  - Upload Files or Folder to HDFS
    - $ bin/hadoop fs -put <localsrc> <hdfsdst>
  - Download HDFS Files or Folder to Local
    - $ bin/hadoop fs -get <hdfssrc> <localdst>
  - Remove Files or Folder
    - $ bin/hadoop fs -rm HDFSfile
    - $ bin/hadoop fs -rmr HDFSfolder
  - ...

1. You might encounter the "ls: Cannot access .: No such file or directory." error if there is no files/directories in your HDFS folder. You can use the "-mkdir" command to create a folder (or "-put" command to upload something) first and then run this command again, then you will get correct output.

Test WordCount program on Hadoop

- Copy some input files into the HDFS
  - We use all the files in conf directory as input:
  - $ bin/hadoop fs -put conf input
- Run wordcount program in the example
  - $ bin/hadoop jar hadoop-examples-1.2.1.jar wordcount input output
- You can copy the output from HDFS to local file system by "-get" command or examine them directly in HDFS by "-cat" command.

$ bin/hadoop fs -get output/part-r-00000 ~/hadoop/result/
  - $ bin/hadoop fs -cat output/part-r-00000
- HDFS can not update files already exist.
  - You have to remove/rename the previous output;

$bin/hadoop fs -rmr output

Stop Hadoop

- Shutdown the Hadoop:
  - Using the stop-all.sh script file in bin directory
    - $ bin/stop-all.sh

Build your MapReduce program

- Suppose you are currently in directory ~/hadoop/assignment3/
  - Your .java files are store in src folder in the current working directory;
- Create a new folder bin to store the class files
  - `$ mkdir bin`
- Compile your .java files

```
$javac -classpath ~/hadoop/hadoop-1.2.1/hadoop-core-1.2.1.jar -d bin src/*
```

- Create jar file
  - `$ jar -cvf a3.jar -C bin/ .`

Run your MapReduce program

- Run the program
  - $ bin/hadoop jar ~/hadoop/assignment3/a3.jar xxx.yyy <input> <output>
  - Here xxx is the package name and yyy is the main class of your mapreduce program; <input> and <output> are the arguments of the main class.