Exercise Preparation

Setup Hadoop, HDFS, Yarn

Get Ubuntu VM ready

1. Get Virtual Box:

Mac OSX:

```
wget https://download.virtualbox.org/virtualbox/5.2.18/VirtualBox-5.2.18-124319-OSX.dmg
```

Windows:

```
wget https://download.virtualbox.org/virtualbox/5.2.18/VirtualBox-5.2.18-124319-Win.exe
```

Linux: No need, you‘re lucky

2. Get Latest Ubuntu LTS Image File (18.04.):

```
wget http://releases.ubuntu.com/18.04.1/ubuntu-18.04.1-desktop-amd64.iso
```

Get Ubuntu VM ready

3. Install Virtual Box.

4. Create and install Ubuntu VM:

CPU Cores:	2++
RAM:	4096MB++
Disk Space:	20GB++ *(VDI, dynamic allocation)*

5. Insert and Install *Vbox_Guest_Additions.iso*

6. Restart VM.

Install and Setup Java

1. Install OpenJDK (JDK 8):

```
sudo apt-get install openjdk-8-jdk
```

2. Verify installation:

```
marcel@VirtualBox:~$ java -version
openjdk version "1.8.0_181"
OpenJDK Runtime Environment (build 1.8.0_181-8u181-b13-0ubuntu0.18.04.1-b13)
OpenJDK 64-Bit Server VM (build 25.181-b13, mixed mode)
```

2. SET *JAVA_HOME* and *JRE_HOME*:

```
sudo vi /etc/environment
```

```
JAVA_HOME="/usr/lib/jvm/java-8-openjdk-amd64"
JRE_HOME="/usr/lib/jvm/java-8-openjdk-amd64/jre"
```

Setup Hadoop User

1. Create User:

```
adduser hadoop
passwd hadoop
```

2. Switch To User:

```
su - hadoop
```


Setup SSH (needed by Hadoop components)

1. Install SSH and PDSH:

```
sudo apt-get install ssh
Sudo apt-get install pdsh
```

2. Create Private/Public Keypair (*without passphrase*):

```
ssh-keygen -t rsa -b 4096
```

3. Add Public Key To Authorized Keys file (to enable passwordless ssh login)

```
cat .ssh/id_rsa.pub >> .ssh/authorized_keys
chmod 0600 ~/.ssh/authorized_keys
```


Setup SSH (needed by Hadoop components)

4. Check If SSH Is Working

```
hadoop@marcel-VirtualBox:~$ ssh localhost
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-29-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage


 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

175 packages can be updated.
65 updates are security updates.

Last login: Sat Sep 29 22:56:15 2018 from 127.0.0.1
hadoop@marcel-VirtualBox:~$ exit
logout
Connection to localhost closed.
hadoop@marcel-VirtualBox:~$
```


Install Hadoop

1. Download Hadoop (v3.1.1):

```
wget http://apache.cs.utah.edu/hadoop/common/hadoop-3.1.1/hadoop-3.1.1.tar.gz
```

2. Extract Binaries:

```
tar -xzf hadoop-3.1.1.tar.gz
```

3. Move Binaries:

```
sudo mv hadoop-3.1.1 /home/hadoop/hadoop
```


Configure Hadoop

1. Set Up **UNIX** Environment Variables

```
vi .bashrc
```

```
export HADOOP_HOME=/home/hadoop/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
```

```
source .bashrc
```

Configure Hadoop

2. Add **Hadoop** Environment Variables *(hadoop-env.sh)*

```
vi /home/hadoop/hadoop/etc/hadoop/hadoop-env.sh
```

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

Configure Hadoop

3. Set Up **CORE** Variables *(core-site.xml)*

```
vi /home/hadoop/hadoop/etc/hadoop/core-site.xml
```

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


Configure Hadoop

4. Set Up **HDFS** Variables *(hdfs-site.xml)*

```
vi /home/hadoop/hadoop/etc/hadoop/hdfs-site.xml
```

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

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

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

Configure Hadoop

5. Set Up **MapReduce** Variables *(mapred-site.xml)*

```
vi /home/hadoop/hadoop/etc/hadoop/mapred-site.xml
```

```
<configuration>
      <property>
            <name>mapreduce.framework.name</name>
            <value>yarn</value>
      </property>
      <property>
            <name>yarn.app.mapreduce.am.env</name>
            <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
      </property>
      <property>
            <name>mapreduce.map.env</name>
            <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
      </property>
      <property>
            <name>mapreduce.reduce.env</name>
            <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
      </property>
</configuration>
```


Configure Hadoop

6. Set Up **YARN** Variables *(yarn-site.xml)*

```
vi /home/hadoop/hadoop/etc/hadoop/yarn-site.xml
```

```
<configuration>
        <property>
                <name>yarn.nodemanager.aux-services</name>
                <value>mapreduce_shuffle</value>
        </property>
</configuration>
```

Configure Hadoop

7. Clear HDFS

```
hdfs namenode -format
```

8. Start HDFS:

```
start-dfs.sh
Maybe requires: export PDSH_RCMD_TYPE=ssh
```

9. Start YARN:

```
start-yarn.sh
```


Check Hadoop/HDFS

10. Run Admin Status Report

```
hdfs dfsadmin -report
```

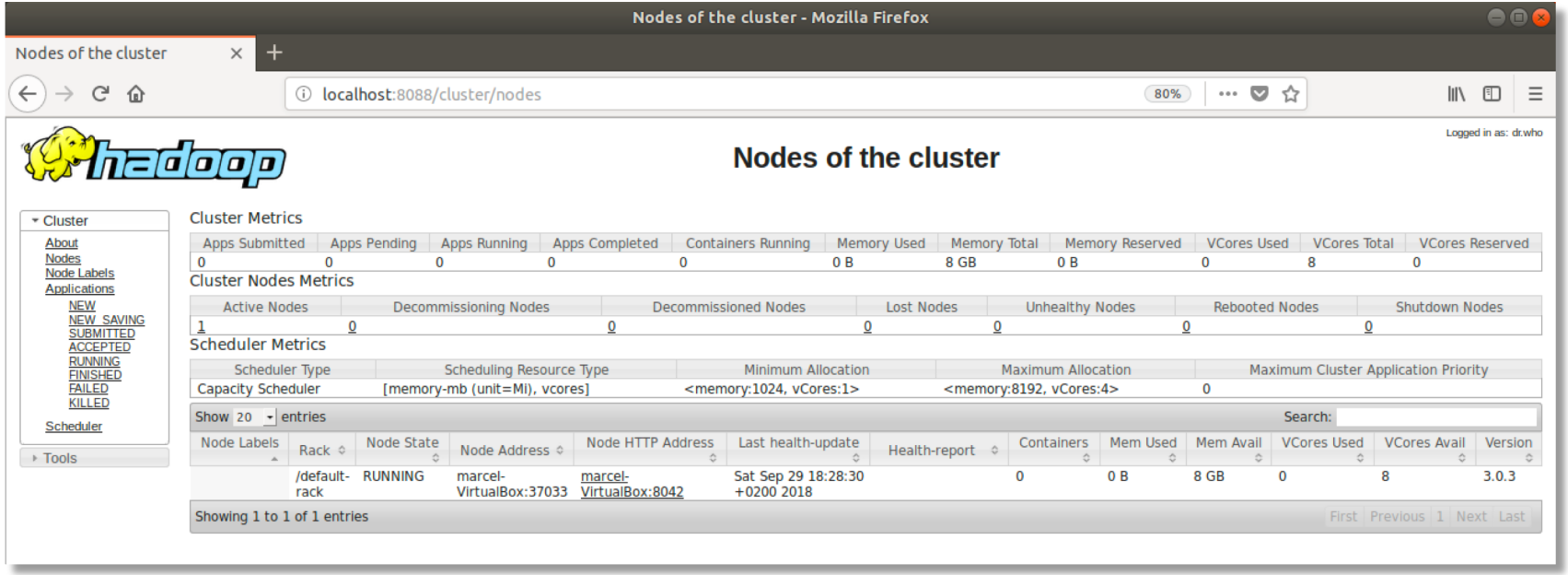
```
Configured Capacity: 20852596736 (19.42 GB)
Present Capacity: 12480020480 (11.62 GB)
DFS Remaining: 12479995904 (11.62 GB)
DFS Used: 24576 (24 KB)
DFS Used%: 0.00%
Replicated Blocks:
                        Under replicated blocks: 0
                        Blocks with corrupt replicas: 0
                        Missing blocks: 0
                        Missing blocks (with replication factor 1): 0
                        Pending deletion blocks: 0
Erasure Coded Block Groups:
                        Low redundancy block groups: 0
                        Block groups with corrupt internal blocks: 0
                        Missing block groups: 0
                        Pending deletion blocks: 0

-------------------------------------------------
Live datanodes (1):

Name: 127.0.0.1:9866 (localhost)
Hostname: marcel-VirtualBox
Decommission Status : Normal
Configured Capacity: 20852596736 (19.42 GB)
DFS Used: 24576 (24 KB)
Non DFS Used: 7289712640 (6.79 GB)
DFS Remaining: 12479995904 (11.62 GB)
DFS Used%: 0.00%
DFS Remaining%: 59.85%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Sat Sep 29 18:04:04 CEST 2018
Last Block Report: Sat Sep 29 18:01:52 CEST 2018
```

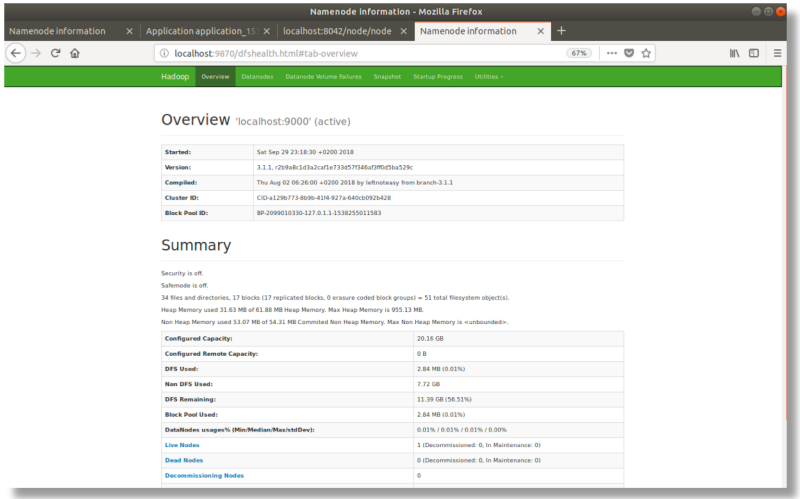
Check Hadoop/HDFS

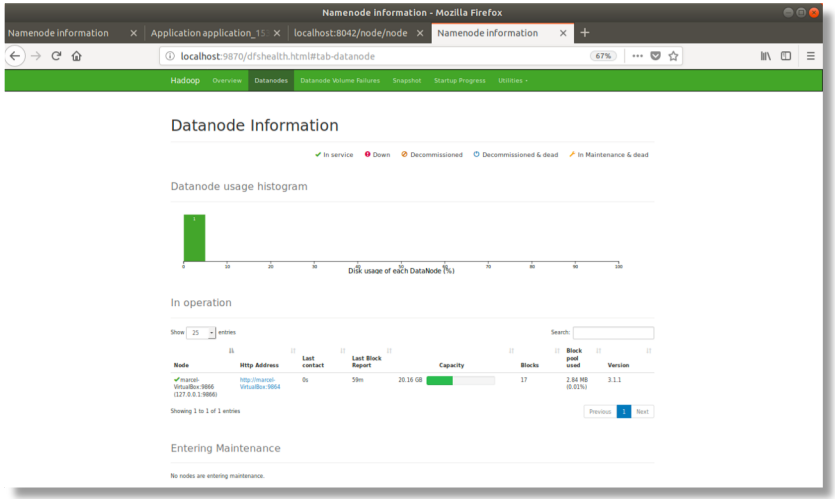
11. Check Ressource Manager Landing Page (http://localhost:8088/cluster):



Check Hadoop/HDFS

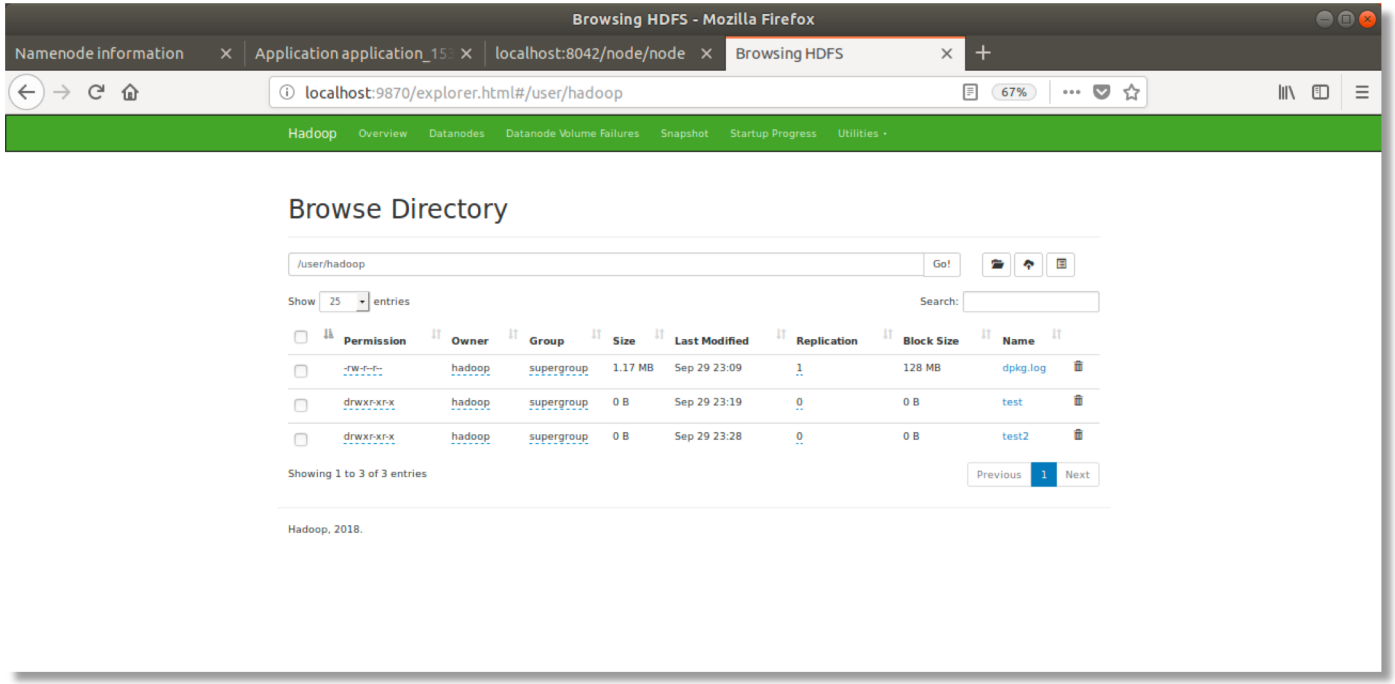
12. Check NameNode Landing and Status Page (http://localhost:9870*):*





Check Hadoop/HDFS

13. Check HDFS File Browser (http://localhost:9870/explorer.html#/):



Working with HDFS

1. Create User Directory (*on HDFS*):

```
hadoop fs -mkdir /user
hadoop fs -mkdir /user/hadoop
```

2. List Directories (*on HDFS*):

```
hadoop@marcel-VirtualBox:~$ hadoop fs -ls /
Found 2 items
drwx------   - hadoop supergroup          0 2018-09-29 23:11 /tmp
drwxr-xr-x   - hadoop supergroup          0 2018-09-29 23:09 /user
hadoop@marcel-VirtualBox:~$
```


Working with HDFS

3. Copy File (just a *random log file*) from local directory to HDFS:

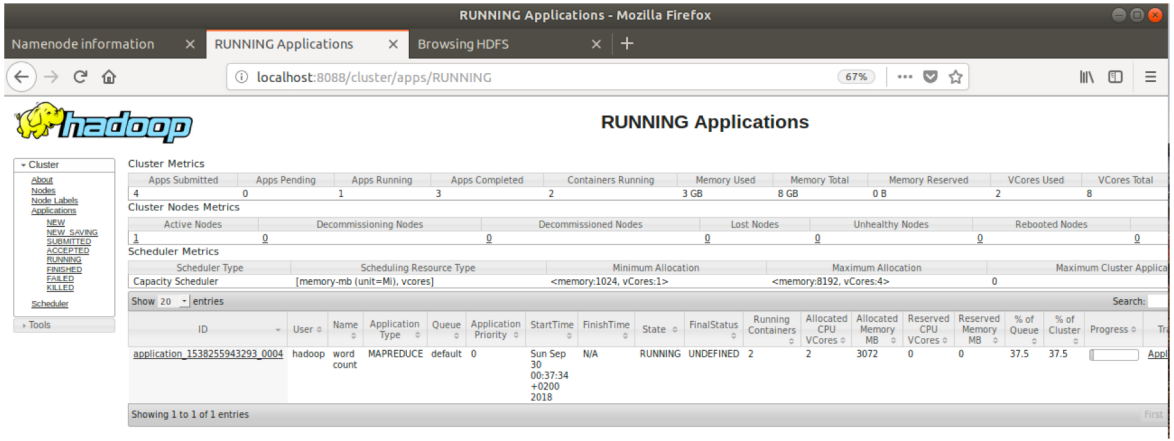
```
hadoop fs -put /var/log/dpkg.log /user/hadoop/dpkg.log
```


Run Example MapReduce Job

1. Using MapReduce WordCount Jar provided by Hadoop

```
hadoop jar hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.1.1.jar
wordcount /user/hadoop/dpkg.log /user/hadoop/test_output
```

2. View Running MapReduce Job:



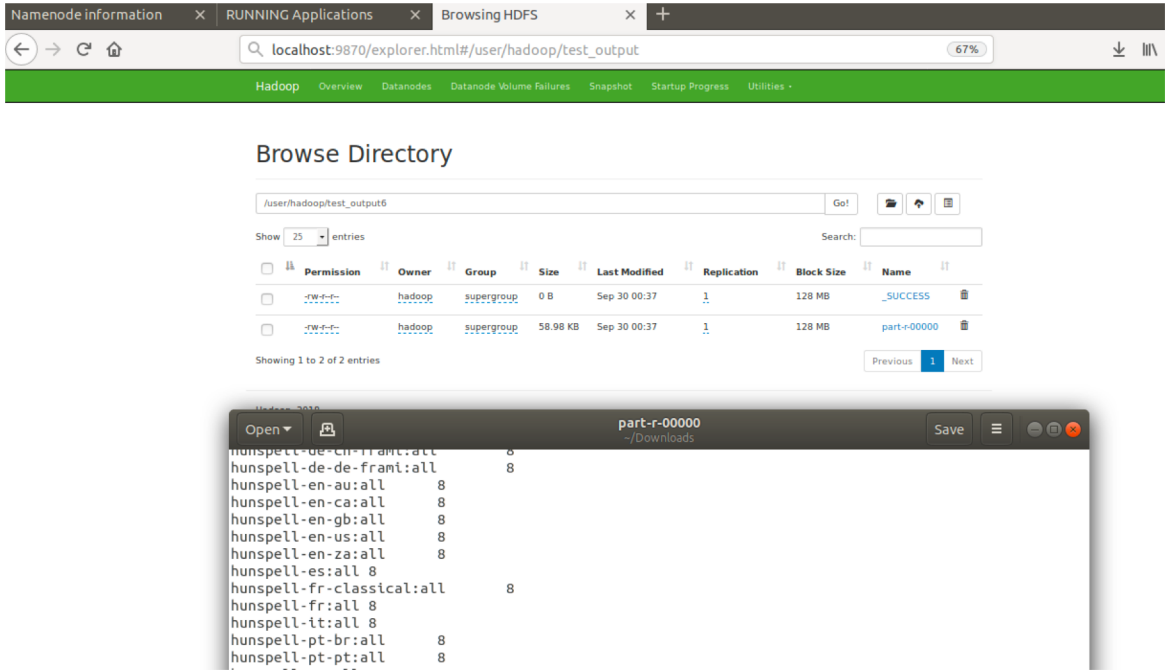
Run Example MapReduce Job

3. Take A Look At The Output/Result (*via Bash*):

```
hadoop@marcel-VirtualBox:~$ hadoop fs -cat /user/hadoop/test_output/part-r-00
000
hunspell-de-de-frami:all      8
hunspell-en-au:all  8
hunspell-en-ca:all  8
hunspell-en-gb:all  8
hunspell-en-us:all  8
hunspell-en-za:all  8
hunspell-es:all     8
hunspell-fr-classical:all     8
hunspell-fr:all     8
hunspell-it:all     8
hunspell-pt-br:all  8
hunspell-pt-pt:all  8
…
```


Run Example MapReduce Job

4. Take A Look At The Output/Result (*via Web HDFS File Browser*):



Exercises

Hadoop, HDFS, Yarn

Exercises

1. **Clone git repo** (to get sample data):

```
git clone https://github.com/marcelmittelstaedt/BigData.git
```

2.

- **Copy sample file** (*/BigData/exercises/01_hadoop/sample_data/Faust_1.txt*) from Git Repo **to HDFS**.
- Use and **run** default **MapReduce** Jar (*hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.1.1.jar*) **to calculate wordcount** for text file.
- **Copy result** of MapReduce job **back to local** ubuntu **filesystem**.

3.

- Use and **run** default **MapReduce** Jar (*hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.1.1.jar*) to **get the count of occurences** of the exact string ***‚Faust‘*** within text file.
- **Copy result** of MapReduce job **back to local** ubuntu **filesystem**.
- **Tip:** don‘t use *wordcount* part of jar.

Help

MapReduce Examples within *hadoop-mapreduce-examples-3.1.1.jar*:

aggregatewordcount:	An Aggregate based mapreduce program that counts the words in the input files.
aggregatewordhist:	An Aggregate based mapreduce program that computes the histogram of the words in the input files.
bbp:	A mapreduce program that uses Bailey-Borwein-Plouffe to compute exact digits of Pi.
dbcount:	An example job that counts the pageview logs stored in a database.
distbbp:	A mapreduce program that uses a BBP-type formula to compute exact bits of Pi.
grep:	A mapreduce program that counts the matches of a regex in the input.
join:	A job that performs a join over sorted, equally partitioned datasets.
multifilewc:	A job that counts words from several files.
pentomino:	A mapreduce tile laying program to find solutions to pentomino problems.
pi:	A mapreduce program that estimates Pi using a quasi-Monte Carlo method.
randomtextwriter:	A mapreduce program that writes 10 GB of random textual data per node.
randomwriter:	A mapreduce program that writes 10 GB of random data per node.
secondarysort:	An example defining a secondary sort to the reduce phase.
sort:	A mapreduce program that sorts the data written by the random writer.
sudoku:	A sudoku solver.
teragen:	Generate data for the terasort.
terasort:	Run the terasort.
teravalidate:	Checking results of terasort.
wordcount:	A mapreduce program that counts the words in the input files.
wordmean:	A mapreduce program that counts the average length of the words in the input files.
wordmedian:	A mapreduce program that counts the median length of the words in the input files.
wordstandarddeviation:	A mapreduce program that counts the standard deviation of the length of the words in the input files.