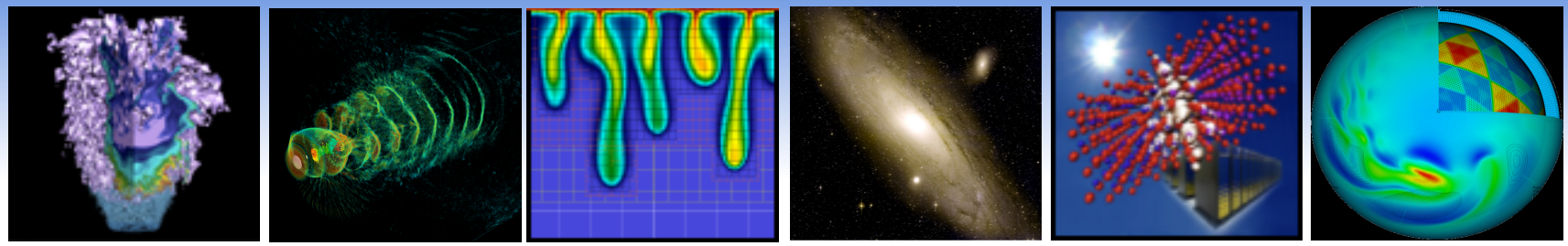

Hadoop Hands-On Exercises

Lawrence Berkeley National Lab

July 2011

U.S. DEPARTMENT OF ENERGY | Office of Science

NERSC

We will …

- Training accounts/User Agreement forms
- Test access to carver
- HDFS commands
- Monitoring
- Run the word count example
- Simple streaming with Unix commands
- Streaming with simple scripts
- Streaming “Census” example
- Pig Examples
- Additional Exercises

Login and Environment

ssh [username]@carver.nersc.gov

echo $SHELL

- should be bash

http://magellan.nersc.gov

(Go to Using Magellan -> Creating a SOCKS proxy)

Printed Handouts

Online http://tinyurl.com/6frxxur

/global/scratch/sd/lavanya/hadooptutorial

Environment Setup

$ ssh [username]@carver.nersc.gov
$ echo $SHELL

If your shell doesn't show /bin/bash please change your shell
$ bash

Setup your environment to use Hadoop on Magellan system
$ module load tig hadoop

Hadoop Command

hadoop command [genericOptions] [commandOptions]

Examples:-

command – fs, jar, job

[genericOptions] - -conf, -D, -files, -libjars, -archives

[commandOptions] - -ls, -submit

HDFS Commands [1]

$ hadoop fs –ls
If you see an error do the following where [username] is your training account username

$ hadoop fs -mkdir /user/[username]

$ vi testfile1 [Repeat for testfile2]
This is file 1
This is to test HDFS

$ hadoop fs -mkdir input

$ hadoop fs -put testfile* input

You can get help on commands -
$ hadoop fs -help

HDFS Commands [2]

```
$   hadoop fs -cat input/testfile1
$   hadoop fs -cat input/testfile*
```

Download the files from HDFS into a directory called input and check there is a input directory.

```
$  hadoop fs -get input input
$   ls input/
```

Monitoring

http://maghdp01.nersc.gov:50030/

http://maghdp01.nersc.gov:50070/

```
$ hadoop job -list
```

Wordcount Example

Input in HDFS

```
$  hadoop fs -mkdir wordcount-in
$  hadoop fs -put /global/scratch/sd/lavanya/
   hadooptutorial/wordcount/* wordcount-in/
```

Run example

```
$  hadoop jar /usr/common/tig/hadoop/
   hadoop-0.20.2+228/hadoop-0.20.2+228-examples.jar
   wordcount wordcount-in wordcount-op
```

View output

```
$  hadoop fs -ls wordcount-op
$  hadoop fs -cat wordcount-op/part-r-00000
$  hadoop fs -cat wordcount-op/p* | grep Darcy
```

Wordcount: Number of reduces

```
$  hadoop dfs -rmr wordcount-op

$  hadoop jar /usr/common/tig/hadoop/
   hadoop-0.20.2+228/hadoop-0.20.2+228-examples.jar
   wordcount -Dmapred.reduce.tasks=4 wordcount-in
   wordcount-op
```

http://maghdp01.nersc.gov:50030/

Wordcount: GPFS

Setup permissions for Hadoop user [ONE-TIME]

$ mkdir /global/scratch/sd/[username]/hadoop

$ chmod -R 755 /global/scratch/sd/[username]

$ chmod -R 777 /global/scratch/sd/[username]/hadoop/

Run Job

$ hadoop jar /usr/common/tig/hadoop

/hadoop-0.20.2+228/hadoop-0.20.2+228-examples.jar wordcount -Dfs.default.name=file:/// /global/scratch/sd/lavanya/hadooptutorial/wordcount/ /global/scratch/sd/[username]hadoop/wordcount-gpfs/

Set perms for yourself

$ fixperms.sh /global/scratch/sd/[username]/hadoop/wordcount-gpfs/

Streaming with Unix Commands

$ hadoop jar $HADOOP_HOME/contrib/streaming/hadoop*-streaming.jar -input wordcount-in -output wordcount-streaming-op -mapper /bin/cat -reducer /usr/bin/wc

$ hadoop fs -cat wordcount-streaming-op/p*

GPFS

$ hadoop jar $HADOOP_HOME/contrib/streaming/hadoop*-streaming.jar -Dfs.default.name=file:/// -input /global/scratch/sd/lavanya/hadooptutorial/wordcount/ -output /global/scratch/sd/[username]/hadoop/wordcount-streaming-op -mapper /bin/cat -reducer /usr/bin/wc

$ fixperms.sh /global/scratch/sd/[username]/hadoop/wordcount-streaming-op

Streaming with Scripts

$ mkdir simple-streaming-example

$ cd simple-streaming-example

$ vi cat.sh

cat

Now let us test this

$ hadoop fs -mkdir cat-in

$ hadoop fs -put /global/scratch/sd/lavanya/hadooptutorial/cat/in/* cat-in/

$ hadoop jar /usr/common/tig/hadoop/hadoop-0.20.2+228/contrib/streaming/hadoop*streaming*.jar -mapper cat.sh -input cat-in -output cat-op -file cat.sh

Streaming with scripts – Number of reducers and mappers

```
$ hadoop jar /usr/common/tig/hadoop/
  hadoop-0.20.2+228/contrib/streaming/
  hadoop*streaming*.jar -Dmapred.reduce.tasks=0 -
  mapper cat.sh -input cat-in -output cat-op -file cat.sh
```

```
$  hadoop jar /usr/common/tig/hadoop/
  hadoop-0.20.2+228/contrib/streaming/
  hadoop*streaming*.jar -
  Dmapred.min.split.size=91212121212  -mapper cat.sh
  -input cat-in -output cat-op -file cat.sh
```

Census sample

```
$  mkdir census
$  cd census
$  cp /global/scratch/sd/lavanya/hadooptutorial/
   census/censusdata.sample .
$  mkdir census
$  cd census
$  cp /global/scratch/sd/lavanya/hadooptutorial/
   census/censusdata.sample .
```

Mapper

```
#The code is available in
    $   vi mapper.sh
    while read line; do
    if [[ "$line" == *Alabama* ]]; then
        echo "Alabama 1"
    fi
    if [[ "$line" == *Alaska* ]]; then
        echo -e "Alaska\t1"
    fi
    done
$   chmod 755 mapper.sh
$   cat censusdata.sample | ./mapper.sh
```

Census Run

```
$  hadoop fs -mkdir census
$  hadoop fs -put  /global/scratch/sd/lavanya/
   hadooptutorial/census/censusdata.sample census/
$  hadoop jar /usr/common/tig/hadoop/
   hadoop-0.20.2+228/contrib/streaming/
   hadoop*streaming*.jar -mapper mapper.sh -input
   census -output census-op -file mapper.sh –reducer /
   usr/bin/wc
$  hadoop fs -cat census-op/p*
```

Census Run: Mappers and Reducers

```
$ hadoop fs -rmr census-op

$  hadoop jar /usr/common/tig/hadoop/
   hadoop-0.20.2+228/contrib/streaming/
   hadoop*streaming*.jar -Dmapred.map.tasks=10 -
   Dmapred.reduce.tasks=2 -mapper  mapper.sh -input
   census -output census-op/ -file mapper.sh -reducer /
   usr/bin/wc
```

Census: Custom Reducer

```
$ vi reducer.sh
   last_key="Alabama"
   while read line; do
      key=`echo $line | cut -f1 -d' '`
            val=`echo $line | cut -f2 -d' '`
      if [[ "$last_key" = "$key" ]];then
               let "count=count+1";
      else
               echo "**" $last_key $count
              last_key=${key};
              count=1;
      fi
   done
   echo "**" $last_key $count
```

Census Run with custom reducer

```
$  hadoop fs -rmr census-op

$  hadoop jar /usr/common/tig/hadoop/
   hadoop-0.20.2+228/contrib/streaming/
   hadoop*streaming*.jar -Dmapred.map.tasks=10 -
   Dmapred.reduce.tasks=2 -mapper mapper.sh -input
   census -output census-op -file mapper.sh -reducer
   reducer.sh -file reducer.sh
```

Pig – Basic Operations

LOAD – loads data into a relational form

FOREACH..GENERATE – Adds or removes fields (columns)

GROUP – Group data on a field

JOIN – Join two relations

DUMP/STORE – Dump query to terminal or file

There are others, but these will be used for the exercises today

Pig Example

Find the number of gene hits for each model in an hmmsearch (>100GB of output, 3 Billion Lines)

bash# cat * |cut –f 2|sort|uniq -c

```
> hits = LOAD '/data/bio/*' USING PigStorage() AS
   (id:chararray,model:chararray, value:float);
> amodels = FOREACH hits GENERATE model;
> models = GROUP amodels BY model;
> counts = FOREACH models GENERATE group,COUNT
   (amodels) as count;
> STORE counts INTO 'tcounts' USING PigStorage();
```

Pig - LOAD

Example:

```
hits = LOAD 'load4/*' USING PigStorage() AS
   (id:chararray, model:chararray,value:float);
```

Pig has several built-in data types (chararray, float, integer)

PigStorage can parse standard line oriented text files.

Pig can be extended with custom load types written in Java.

Pig doesn't read any data until triggered by a DUMP or STORE

Pig – FOREACH..GENERATE, GROUP

Example:

```
amodel = FOREACH model GENERATE hits;
models = GROUP amodels BY model;
counts = FOREACH models GENERATE group,COUNT
   (amodels) as count;
```

Use FOREACH..GENERATE to pick of specific fields or generate new fields. Also referred to as a projection

GROUP will create a new record with the group name and a “bag” of the tuples in each group

You can reference a specific field in a bag with <bag>.field (i.e. amodels.model)

You can use aggregate functions like COUNT, MAX, etc on a bag

Pig – Important Points

- Nothing really happens until a DUMP or STORE is performed.
- Use FILTER and FOREACH early to remove unneeded columns or rows to reduce temporary output
- Use PARALLEL keyword on GROUP operations to run more reduce tasks

Pig - Exercise

Using the census data (path), compute the number of records for each state.