CPSC 4770/6770

Distributed and Cluster Computing

Lecture 13: Introduction to Hadoop MapReduce

Key Steps to Use Hadoop on Palmetto

- Updating .bashrc on your Palmetto home directory
  - echo "module load openjdk/1.8.0_222-b10-gcc/8.3.1 hadoop/3.2.1-gcc/8.3.1" >> ~/.bashrc
- Requesting a compute node (e.g. node0001 ~ nodeXXXX) via qsub, e.g.:
  - qsub -I -l select=3:ncpus=8:mem=14gb,walltime=03:00:00
- Copying the myhadoop template from /zfs/citi
  - cp -R /zfs/citi/myhadoop/ ~/
  - cd ~/myhadoop
- Examining the myhadoop template
  - ls -l
  - ls -l bin/
- Launching myhadoop
  - ./init_hadoop.sh
- Testing myhadoop
  - ./test_hadoop.sh
- Hadoop main commands
  - hdfs
  - hdfs dfs
- Specifying configuration location
  - export HADOOP_CONF_DIR="/home/$USER/hadoop_palmetto/config/"
  - hdfs dfs -mkdir /user/
  - hdfs dfs -mkdir /user/$USER
  - hdfs dfs -ls /user/
  - hdfs dfs -ls /user/$USER
- Stopping myhadoop
  - ./stop_hadoop.sh

Home directory on HDFS

- In HDFS, the home directory is defaulted to be /user/$USER with $USER as your username
- hdfs dfs -ls /user/$USER
- hdfs dfs -ls
- hdfs dfs -ls .

List of HDFS Commands and Usages

- https://hadoop.apache.org/docs/r2.6.4/hadoop-project-dist/hadoop-hdfs/HDFSCommands.html
- http://fibrevillage.com/storage/630-using-hdfs-command-line-to-manage-files-and-directories-on-hadoop

```
hdfs
hdfs dfs
hdfs dfs -help
```

Create a Directory in HDFS

- Usage: hdfs dfs -mkdir <path>
- Example: Create a directory named **intro-to-hadoop** inside your user directory on HDFS
  - hdfs dfs -ls /user/$USER/intro-to-hadoop
  - hdfs dfs -ls /user/$USER

Upload/download a File to HDFS

- Copy single file, or multiple files from local file system to HDFS
  - Usage: hdfs dfs -put <localsrc> … <HDFS_dest_Path>
- Download single file, or multiple files from HDFS to local file system
  - Usage: hdfs dfs -get <HDFS_src_Path> … <localdest>
- Example**:** Upload a text file into the newly created *intro-to-hadoop* directory and copy it back
  - hdfs dfs **-put** /zfs/citi/complete-shakespeare.txt intro-to-hadoop/
  - hdfs dfs **-ls** intro-to-hadoop
  - hdfs dfs **-head** intro-to-hadoop/complete-shakespeare.txt
  - hdfs dfs **-get** intro-to-hadoop/complete-shakespeare.txt ~/shakespeare-complete.txt
  - head ~/shakespeare-complete.txt
  - diff /zfs/citi/complete-shakespeare.txt ~/shakespeare-complete.txt

Upload/download a directory to HDFS

- The put and get subsubcommands can also be used to move directories as well
- Example: upload a movielens directory into the intro-to-Hadoop directory
  - hdfs dfs **-put** /zfs/citi/movielens intro-to-hadoop/
  - hdfs dfs **-ls** intro-to-hadoop
  - hdfs dfs **-ls** intro-to-hadoop/movielens

Check a File State in HDFS

- Check a file/directory state in HDFS
- Usage: hdfs fsck <path>
- Example: Check the health status of the directories */intro-to-hadoop* in HDFS using **fsck**
  - hdfs fsck intro-to-hadoop/ -files -blocks -locations

```
Status: HEALTHY
 Number of data-nodes:  2
 Number of racks:               1
 Total dirs:                    2
 Total symlinks:                0

Replicated Blocks:
 Total size:    1029304096 B
 Total files:   8
 Total blocks (validated):      21 (avg. block size 49014480 B)
 Minimally replicated blocks:   21 (100.0 %)
 Over-replicated blocks:        0 (0.0 %)
 Under-replicated blocks:       21 (100.0 %)
 Mis-replicated blocks:         0 (0.0 %)
 Default replication factor:    3
 Average block replication:     2.0
 Missing blocks:                0
 Corrupt blocks:                0
 Missing replicas:              21 (33.333332 %)

Erasure Coded Block Groups:
 Total size:    0 B
 Total files:   0
 Total block groups (validated):        0
 Minimally erasure-coded block groups:  0
 Over-erasure-coded block groups:       0
 Under-erasure-coded block groups:      0
 Unsatisfactory placement block groups: 0
 Average block group size:      0.0
 Missing block groups:          0
 Corrupt block groups:          0
 Missing internal blocks:       0
FSCK ended at Fri Sep 18 09:09:07 EDT 2020 in 29 milliseconds


The filesystem under path '/user/jin6/intro-to-hadoop' is HEALTHY
```

See Contents of a File in HDFS

- Usage: hdfs dfs -cat <path[filename]>
- Example: display the content of *complete-shakespeare.txt* in HDFS
  - hdfs dfs **-cat** intro-to-hadoop/complete-shakespeare.txt

Top 10 Hadoop HDFS Commands

Source: https://data-flair.training/blogs/top-hadoop-hdfs-commands-tutorial/

Reality of Working with Big Data

- Hundreds or thousands of machines to support big data
  - Distribute data for storage (HDFS)
  - Parallelize data computation (Hadoop MapReduce)
  - Handle failure (HDFS and Hadoop MapReduce)

MapReduce Programming Paradigm

- What is “map”?
  - A function/procedure that is applied to every individual elements of a collection/list/array/…
  - e.g.,

```
int square(x) { return x*x;}
map square [1,2,3,4] -> [1,4,9,16]
```

- What is “reduce”?
  - A function/procedure that performs an operation on a list. This operation will “fold/reduce” this list into a single value (or a smaller subset)
  - e.g.,

```
reduce ([1,2,3,4]) using sum -> 10
reduce ([1,2,3,4]) using multiply -> 24
```

- In HDFS, **map** tasks are performed on top of individual data blocks; **reduce** tasks are performed on intermediate results from map tasks to calculate the final results.

Implementation of MapReduce Programming Paradigm in Hadoop MapReduce

- **Programmers implement:**
  - Map function: Take in the input data and return a *key,value* pair
  - Reduce function: Receive the *key,value* pairs from the mapper and provide a final output as a reduction operation on the pairs
  - Optional functions:
    - Partition function: determines the distribution of mappers' *key,value* pairs to the reducers
    - Combine functions: initial reduction on the mappers to reduce network traffics
- **The MapReduce Framework handles everything else**

What is “everything else”?

- **What is "everything else"?**
  - Scheduling
  - Data distribution
  - Synchronization
  - Error and Fault Handling
- **The cost of "everything else"?**
  - All algorithms must be expressed as a combination of mapping, reducing, combining, and partitioning functions
  - No control over execution placement of mappers and reducers
  - No control over life cycle of individual mappers and reducers
  - Very limited information about which mapper handles which data block
  - Very limited information about which reducer handles which intermediate key
- **Additional challenge**
  - Large scale debugging on big data programming is difficult

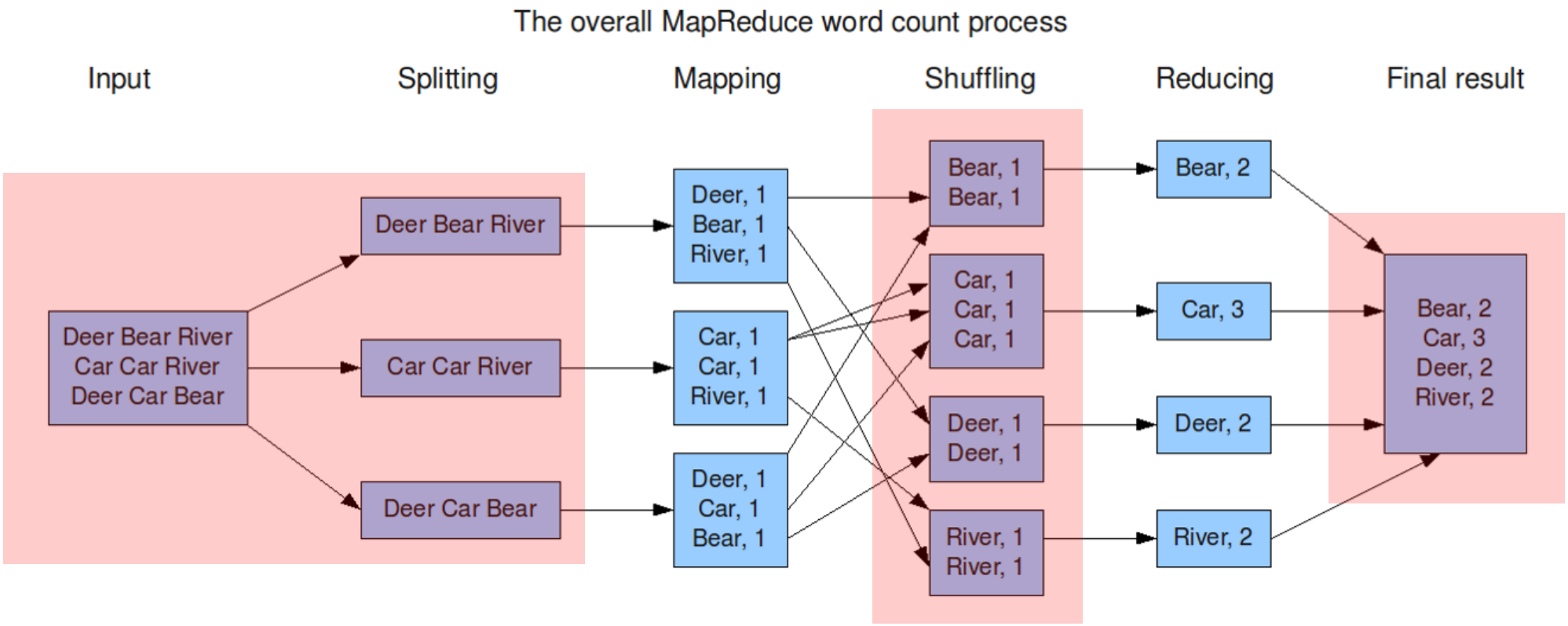
Applications of MapReduce

- Text tokenization, indexing, and search
  - Web access log stats
  - Inverted index construction
  - Term-vector per host
  - Distributed grep/sort
- Graph creation
  - Web link-graph reversal (Google's PageRank)
- Data Mining and machine learning
  - Document clustering
  - Machine learning
  - Statistical machine translation

WordCount: the *Hello, World* of Big Data

- Count how many unique words there are in a file/multiple files
- Standard parallel programming approach:
  - Count number of files
  - Set number of processes
  - Possibly setting up dynamic workload assignment
  - A lot of data transfer
  - Significant coding effort

MapReduce WordCount Example



WordCount (wcMapper.py)

- hdfs dfs -cat intro-to-hadoop/complete-shakespeare.txt 2>/dev/null | head -n 20
- cd ~/myhadoop
- cat -n codes/wcMapper.py

```
#!/usr/bin/env python
import sys
for oneLine in sys.stdin:
    oneLine = oneLine.strip()
    for word in oneLine.split(" "):
        if word != "":
            print ('%s\t%s' % (word, 1))
```

- module load anaconda3/2020.07-gcc/8.3.1
- hdfs dfs -cat intro-to-hadoop/ complete-shakespeare.txt 2>/dev/null | head -n 20 | python ./codes/wcMaper.py
- hdfs dfs -cat intro-to-hadoop/ complete-shakespeare.txt 2>/dev/null | head -n 20 | python ./codes/wcMapper.py | sort

WordCount (wctReducer.py)

```
#!/usr/bin/env python
import sys

current_word = None
total_word_count = 0

for line in sys.stdin:
    line = line.strip()
    word, count = line.split("\t", 1)
    try:
        count = int(count)
    except ValueError:
        continue

    if current_word == word:
        total_word_count += count
    else:
        if current_word:
            print ("%s\t%s" % (current_word, total_word_count))
        current_word = word
        total_word_count = 1

if current_word == word:
    print ("%s\t%s" % (current_word, total_word_count))
```

- cat -n codes/wcReducer.py
- hdfs dfs -cat intro-to-hadoop/complete-shakespeare.txt 2>/dev/null | head -n 20 | python wcMapper.py | sort | python ./codes/wcReducer.py
- Or you can write in a more readable format:

```
hdfs dfs -cat intro-to-hadoop/ complete-shakespeare.txt \
      2>/dev/null \
      | head -n 20 \
      | python wcMapper.py \
      | sort \
      | python ./codes/wcReducer.py
```

Run a MapReduce Program by Streaming

- hdfs dfs -rm -R intro-to-hadoop/output-wordcount

- mapred --config ~/hadoop_palmetto/config streaming \
  -input intro-to-hadoop/text/gutenberg-shakespeare.txt \
  -output intro-to-hadoop/output-wordcount \
  -file wordcountMapper.py \
  -mapper wordcountMapper.py \
  -file wordcountReducer.py \
  -reducer wordcountReducer.py

- hdfs dfs -cat intro-to-hadoop/output-wordcount/part-00000 2>/dev/null | head -n 100

Challenge

- Modify *wcMapper.py* so that punctuations and capitalization are no longer factors in determining unique words

```
#!/usr/bin/env python
import sys
for oneLine in sys.stdin:
    oneLine = oneLine.strip()
    for word in oneLine.split(" "):
        if word != "":
            print ('%s\t%s' % (word, 1))
```

Partial Solution (wcEnahancedMapper.py)

```
#!/usr/bin/env python
import sys
import string

translator = str.maketrans('', '', string.punctuation)

for oneLine in sys.stdin:
    oneLine = oneLine.strip()
    for word in oneLine.split(" "):
        if word != "":
            newWord = word.translate(translator).lower()
            print ('%s\t%s' % (_______, 1))
```

hdfs dfs -cat intro-to-hadoop/complete-shakespeare.txt 2>/dev/null | head -n 20 | python wcEnhancedMapper.py | sort | python wcReducer.py

Basic Anatomy of a Java Hadoop MapReduce Application (Example: WordCount.java)

```java
import java.io.IOException;
import java.util.*;

import org.apache.hadoop.conf.*;
import org.apache.hadoop.fs.*;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.*;
import org.apache.hadoop.mapreduce.lib.output.*;
import org.apache.hadoop.util.*;
```

```java
/* MAIN */
public class WordCount extends Configured implements Tool {

  public static void main(String args[]) throws Exception {
    int res = ToolRunner.run(new WordCount(), args);
    System.exit(res);
  }

  public int run(String[] args) throws Exception {
    Path inputPath = new Path(args[0]);
    Path outputPath = new Path(args[1]);

    Configuration conf = getConf();
    Job job = new Job(conf, this.getClass().toString());

    FileInputFormat.setInputPaths(job, inputPath);
    FileOutputFormat.setOutputPath(job, outputPath);

    job.setJarByClass(WordCount.class);
    job.setInputFormatClass(TextInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);
    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(IntWritable.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    job.setMapperClass(Map.class);
    job.setCombinerClass(Reduce.class);
    job.setReducerClass(Reduce.class);

    return job.waitForCompletion(true) ? 0 : 1;
  }
```

WordCount.java (Map)

```
46   /* MAP */
47   public static class Map extends Mapper<Object, Text, Text, IntWritable> {
48     private Text word = new Text();
49
50     public void map(Object key, Text value, Context context) throws IOException, InterruptedException {
51       StringTokenizer tokenizer = new StringTokenizer(value.toString());
52       while (tokenizer.hasMoreTokens()) {
53         word.set(tokenizer.nextToken());
54       context.write(word, new IntWritable(1));
55     }
56   }
```

WordCount.java (Reduce)

```
58   /* REDUCE */
59   public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable>{
60     public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {
61       int sum = 0;
62       for (IntWritable value : values) {
63         sum += value.get();
64       }
65       context.write(key, new IntWritable(sum));
66     }
67   }
68 }
```