Distributed Computing: File Systems and MapReduce

CISC-525

Phil Grim

HARRISBURG UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Overview

- Distributed File Systems and Clustered Computing
  - What is a Distributed File System?
  - What is the Hadoop Distributed File System, and what are its goals?
  - What is an HDFS cluster?
  - What is rebalancing?
- The MapReduce Framework
  - The Mapper
  - The Reducer
- An Example of a MapReduce program

Distributed File Systems and Clustered Computing

- Traditional data storage consists of large computer systems with massive arrays of attached disk storage.
  - Expensive
  - Difficult to scale
- Distributed file systems are accessed over a network
  - Network File System (NFS)
  - Windows File Sharing (SMB/NMB)
  - Storage Area Networks (Netapp/EMC)
  - Still a single point of failure since all of these still rely on a single computer
- Clustered computing provides an alternative
  - Groups of commodity computers with local storage
  - Distributed file system spread across cluster
  - Google File System, Apache Hadoop most common examples
  - Much less expensive, easy to scale by adding more nodes

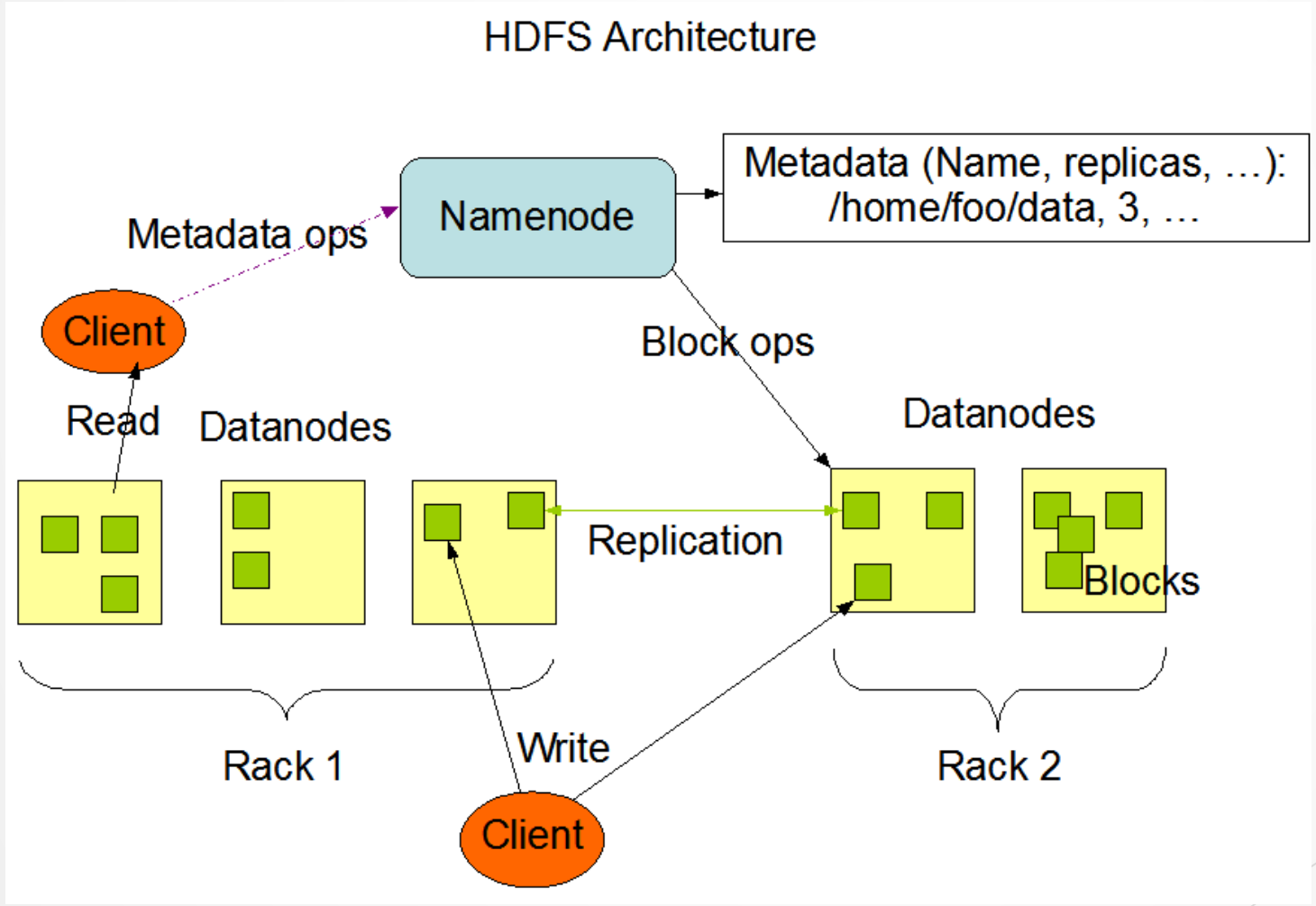
The Hadoop Distributed File System

- A file system based on a cluster of commodity computers working together to present high-performance, fault-tolerant disk storage
- An open-source implementation of the Google File System
- High performance for sequential reads of large files that are written once and read many times
- Uses large block-based storage and replication to ensure that files remain accessible even if one or more of the servers in the cluster go offline
- Not an ideal solution for small files or for random access within files

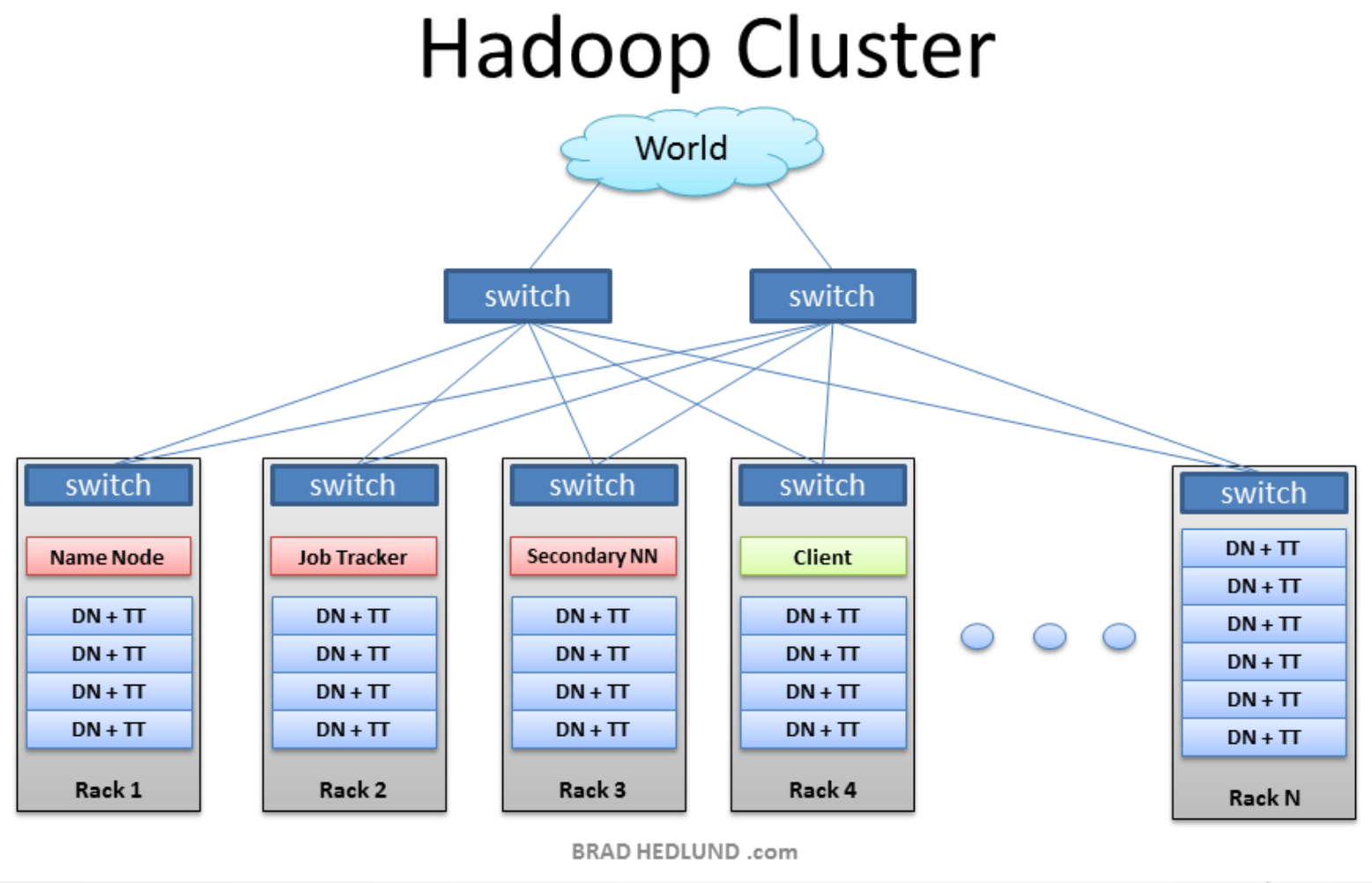
HDFS Cluster

- An HDFS cluster consists of a group of commodity computers working together
- The Name Node serves as the table of contents of the file system much like the File Allocation Table of a traditional file system
- Data nodes serve a portion of the local disk drive as part of the distributed file system
- Clients query the name node to discover the locations of blocks in a file
- Data transfers are handled directly between clients and Data Nodes, reducing overhead on the Name Node
- Secondary Name Nodes provide failover if the Name Node should fail
- Data Nodes generally co-located with Hadoop MapReduce Task Trackers to provide data locality to MapReduce jobs

Distributed File System Architecture



HDFS Cluster



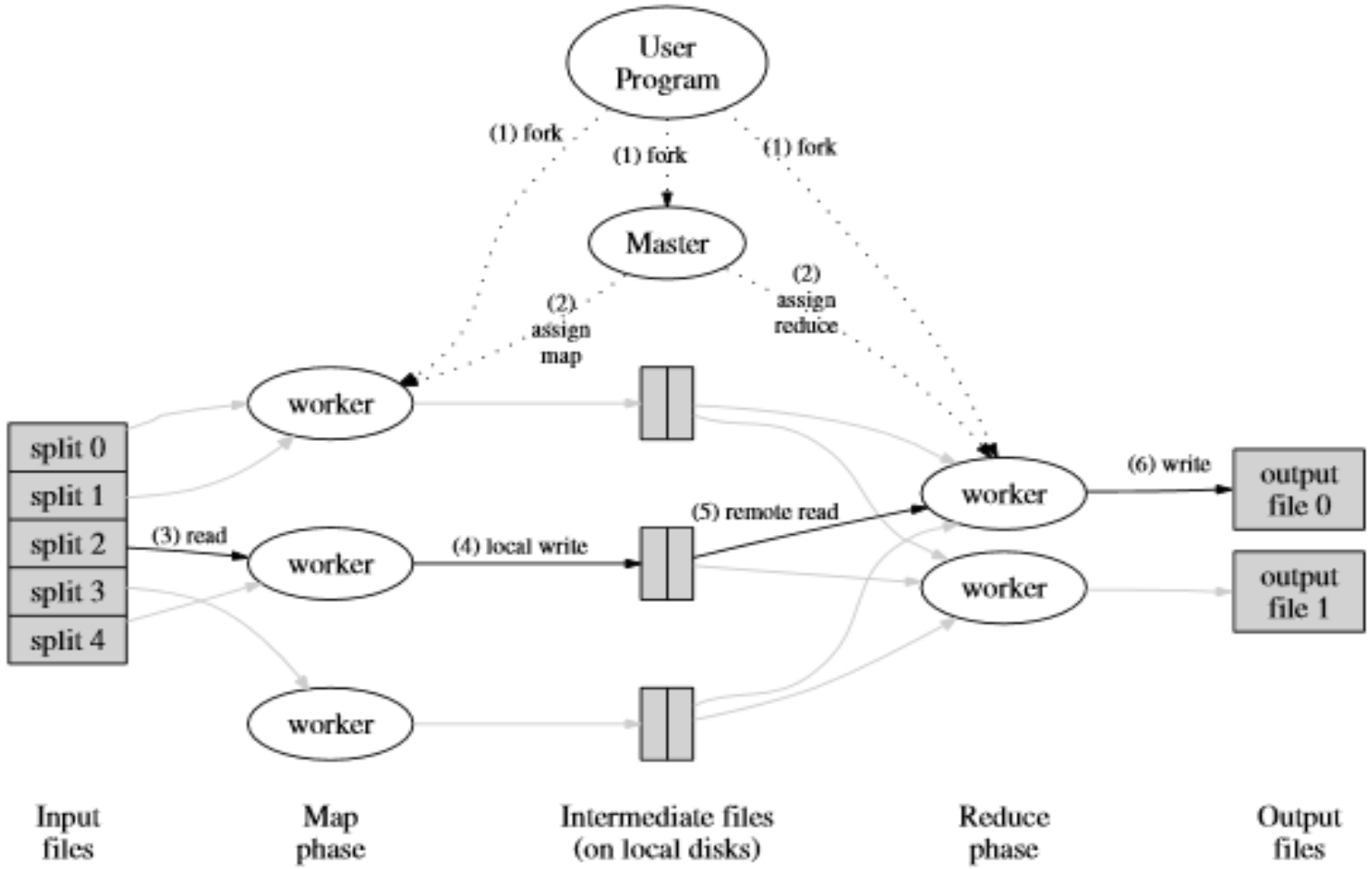
Rebalancing

- Files in HDFS are stored in blocks of 64mb which are spread across multiple data nodes, and replicated across other data nodes
- If a data node has too many blocks, it may receive an inordinate amount of traffic
- If a particular block is under-replicated, the fault tolerance of that file is reduced
- HDFS will in both cases work to alleviate the problem by moving or copying blocks between data nodes so that each block is properly replicated and each data node is serving a similar number of blocks and amount of network traffic.

The MapReduce Framework

- Invented by Google, the MapReduce framework leverages clusters of commodity computers and a distributed file system to enable massively parallel processing of data.
- A MapReduce job consists of two functions, Map and Reduce.
- The Map function runs locally to the data
- The Reduce function combines the output of the Map functions and produces the final output
- A Job Tracker accepts jobs and distributes the Map and Reduce tasks to Task Trackers on each compute node.

The MapReduce Architecture



The Map Function

- The Map function receives key/value pairs as inputs.
- Can be lines from a file, cells from a NoSQL data store, or other implementations of an input format.
- The Map function emits key/value pairs after processing the data. One call to the map function can result in zero, one, or many outputs.
- The MapReduce framework attempts to localize the inputs to the map functions, that is, each mapper gets data that is stored on the compute node it is running on
- Output from the Map function is sorted and delivered to the Reduce phase by the framework

The Reduce Function

- The Reduce function receives a key and a list of values as input.
- Every value that is emitted by a map function and associated with a particular key is delivered to a single reducer.
- The reducer performs its user-specified operation and emits key/value pairs to the final destination, which can be a file, a NoSQL datastore, or other custom implementation of an output format.
- There are generally many mappers and few reducers in a MapReduce job, depending on the distribution of data.

An Example of MapReduce

- Example input data for a classification analytic.

Dell	Precision M6700	i7	3.2GHz	32GB	1TB
Dell	Precision M6600	i7	2.8GHz	24GB	1TB
Dell	Precision M6500	i7	2.4GHz	16GB	500GB
Acer	A300	i5	2.8GHz	12GB	500GB
Acer	A100	i5	2.4GHz	8GB	500GB
Lenovo	LR770	i7	3.0GHz	16GB	750GB
Lenovo	LR550	i5	2.4GHz	8GB	500GB
Acme	AC55	i5	2.4GHz	8GB	500GB
Acme	AC22	i3	2.2GHz	4GB	250GB
Asus	T600	17	3.2GHz	16GB	1TB
Asus	T400	i5	2.4GHz	8GB	500GB
Asus	T200	i3	2.2GHz	4GB	250GB

The Map function

- Each mapper will receive one line of the text file. The key/value pair consists of the line number from the file as the key, and the text of the line as the value.
- The mapper performs feature extraction on the data using a Naïve Bayes Classifier.
- The mapper outputs key/value pairs consisting of the laptop model and classification value for the processor speed, hard drive size, and memory size of each laptop model – i.e. three key/value pairs for each line of input.

```
// Output variables are declared static to avoid an object creation each time
map is called.

protected static Text modelOut = new Text();

protected static Text classOut = new Text();


// Map is called once per line in the file.  It emits a key/value pair for each
feature extracted.

public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException

{

  String[] tokens = value.toString().split("\t");

  if (tokens.length > 2)

  {

     modelOut.set(tokens[0] + " " + tokens[1]);

     Collection<Classification<String, String>> classifications =
bayesClassifier.classifyDetailed(Arrays.asList(tokens));


    classOut.set(getCategory(MEM_PREFIX, classifications));

    context.write(modelOut, classOut);

    classOut.set(getCategory(PROC_PREFIX, classifications));

    context.write(modelOut, classOut);

    classOut.set(getCategory(HD_PREFIX, classifications));

    context.write(modelOut, classOut);

  }

}
```

The Reduce function

- The reduce function is called with each laptop model key emitted by a mapper, and the list of values associated with that key.
- The reducer combines the three values into a single line that is emitted and written to the output file

```
/**
  * The reducer class gathers the output of the mappers and creates a single
line of output for each input
  * key with the features in alphabetical order.
  */
public static class FeatureExtractionReducer extends Reducer<Text, Text, Text,
Text>
{
  // Output variables are static to avoid object creation on each call.
  protected static Text out = new Text();
  protected static SortedSet<String> outSet = new TreeSet<String>();
  protected static StringBuilder sb = new StringBuilder();

  // The reduce method is called once for each key emitted by the mapper, with
an iterable collection of
  // each value emitted for that key.
  public void reduce(Text key, Iterable<Text> values, Context context) throws
IOException, InterruptedException
  {
    outSet.clear();
    sb.setLength(0);
    for (Text t : values)
    {
      outSet.add(t.toString());
    }
    for (String s : outSet)
    {
      sb.append(s).append("\t");
    }
    out.set(sb.toString().trim());
    context.write(key, out);
  }
}
```

MapReduce Job

Hadoop job_201404161948_0002 on pbj-dev-centos

User: pgrim
Job Name: SentimentAnalysis
Job File: hdfs://pbj-dev-centos:9000/tmp/hadoop-mapred/mapred/staging/pgrim/.staging/job_201404161948_0002/job.xml
Submit Host: pbj-dev-centos.data-tactics-corp.com
Submit Host Address: 192.168.30.98
Job-ACLs: All users are allowed
Job Setup: Successful
Status: Running
Started at: Thu Apr 17 10:12:40 GMT-05:00 2014
Running for: 2hrs, 27mins, 59sec
Job Cleanup: Pending

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	0.00%	1	0	1	0	0	0 / 0
reduce	0.00%	1	1	0	0	0	0 / 0

	Counter	Map	Reduce	Total
Job Counters	SLOTS_MILLIS_MAPS	0	0	1,182
	Launched map tasks	0	0	1
	Data-local map tasks	0	0	1
File Input Format Counters	Bytes Read	0	0	0
FileSystemCounters	HDFS_BYTES_READ	98	0	98
	FILE_BYTES_WRITTEN	59,217	0	59,217
Map-Reduce Framework	Map output materialized bytes	0	0	0
	Combine output records	0	0	0
	Map input records	0	0	35
	Physical memory (bytes) snapshot	0	0	5,882,322,944
	Spilled Records	0	0	0
	Map output bytes	0	0	95,573
	Total committed heap usage (bytes)	0	0	5,690,621,952
	CPU time spent (ms)	0	0	8,479,470
	Virtual memory (bytes) snapshot	0	0	7,330,136,064
	SPLIT_RAW_BYTES	98	0	98
	Map output records	0	0	1,516
	Combine input records	0	0	0

Hadoop Ecosystem

- Tools and frameworks for manipulating data stored in Hadoop HDFS
  - Hive – SQL-like query language
  - Pig – Simple query language
  - Oozie – Job scheduling and workflows
  - Hbase – Columnar data store
  - Accumulo – Columnar data store with cell-level security
  - Drill – Unified optimized query language
  - Hue – Web based user interface

Additional Reading

References

Apache Software Foundation. (2013, 08 04). *HDFS Architecture Guide.* Retrieved from Hadoop: http://hadoop.apache.org/docs/stable1/hdfs_design.html

Dean, J., & Ghemawat, S. (2004). *MapReduce: Simplified Data Processing on Large Clusters.* Retrieved from Google Research: http://static.googleusercontent.com/media/research.google.com/en/us/archive/mapreduce-osdi04.pdf

Ghemawat, S., Gobioff, H., & Leung, S.-T. (2003). *The Google File System.* Retrieved from Google Research: http://static.googleusercontent.com/media/research.google.com/en/us/archive/gfs-sosp2003.pdf

Rajaraman, A., Leskovec, J., & Ullman, J. D. (2013). *Mining of Massive Datasets.* Palo Alto, CA: Stanford University.