CS 696 Intro to Big Data: Tools and Methods
Fall Semester, 2017
Doc 10 Word Count, Hadoop
Oct 4, 2017

Word Count - Hello World of Hadoop

Given a text file

- Count the number of times each word occurs in the file

Examples do not worry about

- What a word is
- Different endings to same word
- Words hyphenated at end of sentences

Sequential Algorithm

“a cat a bat a cat hat”

“a", "cat", "a", "bat", "a", "cat", “hat"

a -> 1	a -> 1 cat -> 1	a -> 2 cat -> 1	a -> 2 cat -> 1 bat -> 1	a -> 3 cat -> 1 bat -> 1	a -> 3 cat -> 2 bat -> 1	a -> 3 cat -> 2 bat -> 1 hat -> 1

Spark/Hadoop Algorithm

“a cat a bat a cat hat”

M1	M2	M3
“a cat a”	“bat a”	“cat hat”

split

“a", "cat", "a"	"bat", "a"	"cat", “hat"

map

(“a”, 1), (“cat", 1), ("a", 1)	("bat", 1), ("a", 1)	("cat", 1), (“hat", 1)

reduceByKey

$$\begin{tikzcd}
(\text{“a”},2),\ (\text{“cat"}, 1) \arrow[dr] \arrow[d] & (\text{"bat"}, 1),\ (\text{"a"}, 1) \arrow[dr] \arrow[d] & (\text{"cat"}, 1),\ (\text{“hat"}, 1) \arrow[dll] \arrow[d, "\text{shuffle}"] \\
(\text{“cat"}, 2) & (\text{"a"}, 3) & (\text{"bat"}, 1),\ (\text{“hat"}, 1)
\end{tikzcd}$$

Word Count

Scala

```
val textFile = sc.textFile("words.txt")
val counts = textFile.flatMap(line => line.split(" ")).
                map(word => (word, 1)).
                reduceByKey(_ + _)
counts.saveAsTextFile("counts")
```

Java

```
JavaRDD<String> textFile = sc.textFile("hdfs://...");
JavaPairRDD<String, Integer> counts = textFile
    .flatMap(s -> Arrays.asList(s.split(" ")).iterator())
    .mapToPair(word -> new Tuple2<>(word, 1))
    .reduceByKey((a, b) -> a + b);
counts.saveAsTextFile("hdfs://...");
```

Why Flatmap

TextFile

“A cat in the
hat returns”

Read as Lines

(“A cat in the”,
“hat returns”)

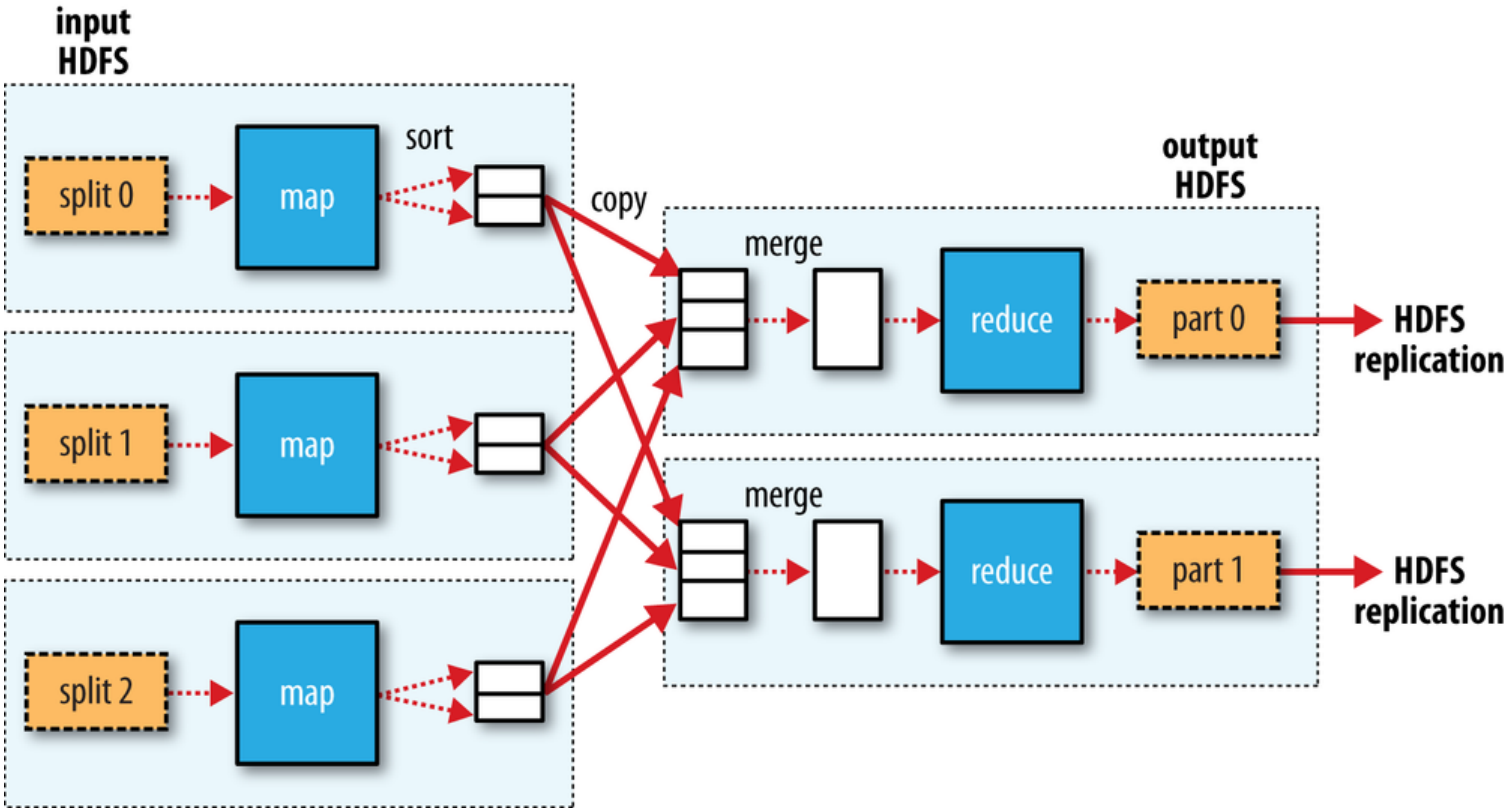
map(line => line.split(" "))

((“A”, “cat”, “in”, “the”),
(“hat”, “returns”))

flatMap(line => line.split(" "))

(“A”, “cat”, “in”, “the”, “hat”, “returns”)

Shuffle


input HDFS
split 0
map
sort
split 1
map
split 2
map
copy
output HDFS
merge
reduce
part 0
HDFS replication
merge
reduce
part 1
HDFS replication

What is Hadoop

Framework for distributed storage & distributed processing of very large data sets

Hadoop Common

Utilities

Hadoop Distributed File System (HDFS)

Hadoop YARN

Manage computing resources in clusters & schedule users' applications

Hadoop MapReduce

Implementation of the MapReduce programming model

What is Hadoop

Java program + native C code + shell scripts

Java Jar file

Native Libraries

For performance some components of hadoop have native libraries

- Compression (bzip2, lz4, snappy, zlib)
- Native io utilities
- CRC32 checksum

Only on GNU/Linux

- RHEL4/Fedora
- Unbuntu
- Gentoo

On other systems uses Java implementation

```
16/11/02 09:12:16 WARN util.NativeCodeLoader: Unable to load native-hadoop library
for your platform... using builtin-java classes where applicable
```

Hadoop Distributed Filesystem HDFS

Parts of a file are distributed on different machine

Large files - 100 MB, GB or TB
- File block size - 128MB or larger for efficient transfer

Streaming data access
- Copy to HDSF once
- Read many times

Handles node failure

High-latency access

Single Writer, append only

Namenode & Datanodes

- Namenode
  - master
  - Manages filesystem
    - Filesystem tree & metadata for files * directories
  - Clients interact with namenode
  - Cluster may contain multiple namenodes
    - Federation
      - Divide namespace up if too many files
    - High Availability
      - Backup if main namenode fails

- Datanode
  - worker
  - Reads file blocks
  - Reports to name node which blocks it contains

Datanode fails

Each block of a file is stored on multiple machines

This is set in conf file

- For standalone & Pseudo distributed set to 1

Hadoop WordCount

Map
    Function in subclass of Mapper

Reduce
    Function is subclass of Reducer

Main
    Configures and runs hadoop job

Map

```
public static class TokenizerMapper
     extends Mapper<Object, Text, Text, IntWritable>{

  private final static IntWritable one = new IntWritable(1);
  private Text word = new Text();

  public void map(Object key, Text value, Context context
                  ) throws IOException, InterruptedException {
    StringTokenizer itr = new StringTokenizer(value.toString());
    while (itr.hasMoreTokens()) {
      word.set(itr.nextToken());
      context.write(word, one);
    }
  }
}
```

Reduce

```
public static class IntSumReducer
     extends Reducer<Text,IntWritable,Text,IntWritable> {
  private IntWritable result = new IntWritable();

  public void reduce(Text key, Iterable<IntWritable> values,
                     Context context
                     ) throws IOException, InterruptedException {
    int sum = 0;
    for (IntWritable val : values) {
      sum += val.get();
    }
    result.set(sum);
    context.write(key, result);
  }
}
```

Main - Driver Program

```
public static void main(String[] args) throws Exception {
  Configuration conf = new Configuration();
  Job job = Job.getInstance(conf, "word count");
  job.setJarByClass(WordCount.class);
  job.setMapperClass(TokenizerMapper.class);
  job.setCombinerClass(IntSumReducer.class);
  job.setReducerClass(IntSumReducer.class);
  job.setOutputKeyClass(Text.class);
  job.setOutputValueClass(IntWritable.class);
  FileInputFormat.addInputPath(job, new Path(args[0]));
  FileOutputFormat.setOutputPath(job, new Path(args[1]));
  System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```

Hadoop Ecosystem

Hadoop

- HDFS
- MapReduce
- YARN

Tez

Pig

Hive

Hbase

Sqoop

Oozie

Falcon

Spark

ZooKeeper

Mahout

Phoenix

BigTop

+ others

Apache Pig

Programming Map-Reduce can be low level

Apache Pig - high-level platform for creating programs for Hadoop

Pig Latin

```
input_lines = LOAD '/tmp/my-copy-of-all-pages-on-internet' AS (line:chararray);
words = FOREACH input_lines GENERATE FLATTEN(TOKENIZE(line)) AS word;
filtered_words = FILTER words BY word MATCHES '\\w+';
word_groups = GROUP filtered_words BY word;
word_count = FOREACH word_groups GENERATE COUNT(filtered_words) AS
                            count, group AS word;

ordered_word_count = ORDER word_count BY count DESC;
STORE ordered_word_count INTO '/tmp/number-of-words-on-internet';
```

Apache Hive

SQL is common way to interact with data

Hive provides SQL like query language for HDFS, Amazon S3 data

HiveQL - converted into MapReduce

```
DROP TABLE IF EXISTS docs;
CREATE TABLE docs (line STRING);
LOAD DATA INPATH 'input_file' OVERWRITE INTO TABLE docs;
CREATE TABLE word_counts AS
SELECT word, count(1) AS count FROM
 (SELECT explode(split(line, '\s')) AS word FROM docs) temp
GROUP BY word
ORDER BY word;
```

Apache HBase

BigTable for Hadoop

Non-relational distributed database

Fault-tolerant way of storing large quantites of sparse data

Apache Sqoop

People have data in non-hadoop databases

Sqoop

- Transferring data between relational databases & Hadoop

Apache Phoenix

But SQL is common

Phoenix
- Massively parallel relational database for Hadoop
- Uses HBase to store data

Apache Spark

Hadoop has latency issues - reads data from disk

MapReduce is not conducive to solving all problems

Spark

- Uses distributed shared memory: Resilient distributed dataset (RDD)
- Iterative algorithms
- Implemented in Scala

Spark Core

Spark SQL

- Dataframes & SQL

Spark Streaming

Spark MLlib

- Machine learning

Apache Mahout

Hadoop does not have machine learning libraries

Mahout

- Environment for quickly creating scalable machine learning applications
- Samsara - R-line syntax & environment

Apache Flink, Apache Storm

Hadoop does batch jobs

Spark streaming has delays

Fling & Storm

- Each calin to have high throughput and low latency streaming