

# CSX55: DISTRIBUTED SYSTEMS [MAPREDUCE/HADOOP]

## What's this hullabaloo about an elephant?

No, not the one named Horton  
Who has fun in the Jungle of Nool

This one's named Hadoop, and is just as cool  
Crunching through data and having fun

Shrideep Pallickara  
Computer Science  
Colorado State University

COMPUTER SCIENCE DEPARTMENT



COLORADO STATE UNIVERSITY

1

## Frequently asked questions from the previous class survey

- Why are intermediate map outputs not put in GFS?
- Since there are more mappers than reducers, will there be machines with only mapper tasks?
- Is there a standardized number of reducers that is typically chosen?



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.2

2

## Topics covered in today's lecture

- MapReduce
- Hadoop



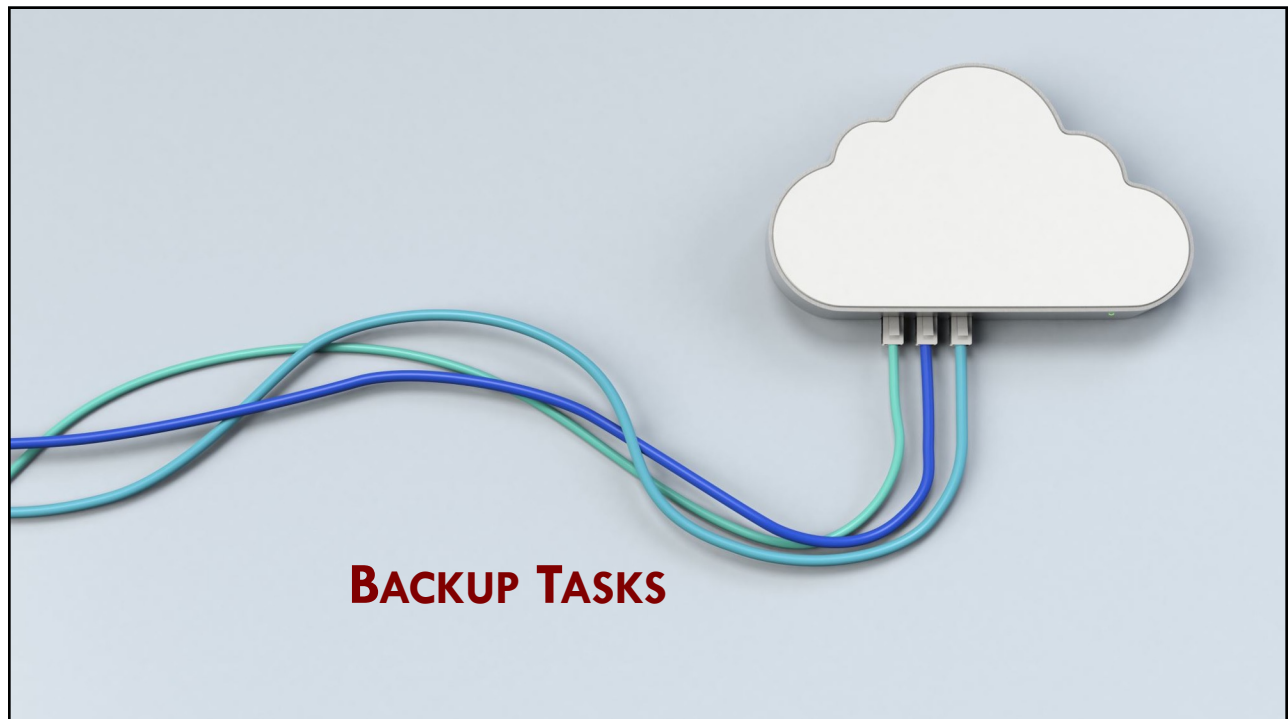
COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.3

3



4

## Stragglers

- Machine that takes an **unusually long time** to complete a map or reduce operation
- Can slow down entire computation



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.5

5

## How stragglers arise

- Machine with a **bad disk**
  - Frequent, correctable errors
  - Read performance drops from 30 MB/s to 1 MB/s
- Over **scheduling**
  - Many tasks executing on the same machine
  - **Competition** for CPU, memory, disk or network cycles
- **Bug** in machine initialization code
  - Processor caches may be disabled



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.6

6

## Alleviating the problem of stragglers

- When a MapReduce operation is *close to completion*
- Schedule **backup** executions of *remaining* in-progress tasks
- Task completed when
  - ▣ Primary or backup finishes execution
- Significantly reduces time to complete large MapReduce operations



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.7

7



8

## Partitioning Function

- Users simply specify R
  - ▣ The number of output files
- Default partitioning
  - ▣  $\text{hash}(\text{key}) \bmod R$
- Sometimes output keys are URLs
  - ▣ Entries from a host must go to same output file
  - ▣  $\text{hash}(\text{Hostname}(\text{urlkey})) \bmod R$



## Ordering Guarantees

- Intermediate *key/pairs* are processed in **increasing** key order
- Easy to generate sorted output file



## The **Combiner** function

- There is significant **repetition** in intermediate keys produced by each map task
- For word-frequencies
  - Each map may produce 100s or 1000s of <the, "1">
- All of these counts sent over the network
- Combiner: Does **partial merging** of this data
  - *Before* it is sent to reducer



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.11

11

## Combiner function

- Executed on each machine that performs map task
- Code implementing combiner & reduce function
  - *Usually* the same ... [We will see an example where this is not true.]
- Difference?
  - COMBINE: Output written to *intermediate* file
  - REDUCE: Output written to *final output* file



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.12

12

## Input/Output Types: Support for reading input data in different formats

- Text mode treats every line as a  $\langle key, value \rangle$  pair
  - ▣ Key: Offset in the file
  - ▣ Value: Contents of the line
- $\langle key, value \rangle$  pairs are sorted by *key*
- Each input type *knows how to split itself* for
  - ▣ Processing as separate map tasks
  - ▣ Text mode splitting occurs only at line boundaries



## Side-effects

- Besides intermediate files, other auxiliary files may be produced
  - ▣ Side effects
- No atomic commits for multiple auxiliary files that are produced



## Skipping Bad Records

[1/3]

- Bugs in user code cause Map or Reduce functions to crash
  - ▣ Deterministically: On certain records
- Fix the bug?
  - ▣ Yes, but not always feasible
- Acceptable to ignore a few records



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.15

15

## Skipping Bad Records

[2/3]

- Optional mode of operation
  - ① **Detect** records that cause *deterministic crashes*
  - ② **Skip** them
- Each worker installs a **signal handler** to catch segmentation violations and bus errors



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.16

16



## Skipping Bad Records

[3/3]

- Signal handler sends *last gasp* UDP packet to the Master
  - ▣ Contains sequence number
- When Master sees more than 1 failure at that record
  - ▣ Indicates record should be skipped during next execution



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.17

17

## Local Execution

- Support for **sequential execution** of MapReduce operation on a single machine
  - ▣ Helps with debugging, profiling, and testing
- Controls to *limit* computation to a particular map
- Invoke programs with a special flag
  - ▣ Use debugging and testing tools



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.18

18

## Status Information

- Master runs internal HTTP Server
- Exports pages for viewing
- Show the progress of a computation
  - ▣ Number of tasks in progress
  - ▣ Number of tasks that completed
  - ▣ Bytes of input
  - ▣ Bytes of intermediate data
  - ▣ Processing rate



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.19

19



20

## Hadoop

- Java-based open-source implementation of MapReduce
- Created by Doug Cutting
- Origins of the name Hadoop
  - ▣ Stuffed yellow elephant
- Includes HDFS [Hadoop Distributed File System]



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.21

21

## Hadoop timelines

- Feb 2006
  - ▣ Apache Hadoop project officially started
  - ▣ Adoption of Hadoop by Yahoo! Grid team
- Feb 2008
  - ▣ Yahoo! Announced its search index was generated by a 10,000-core Hadoop cluster
- May 2009
  - ▣ 17 clusters with 24,000 nodes



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.22

22

## Hadoop Releases

- There are two active releases at the moment
  - ▣ 2.10.x
  - ▣ 3.4.x



## Hadoop Evolution

- 0.20.x series became 1.x series
- 0.23.x was forked from 0.20.x to include some major features
- 0.23 series later became 2.x series
- 2.8.0 is branched off from 2.7.3
- 2.9.0 is branched off from 2.8.2
- 3.0.0 series is branched off from 2.7.0
- 3.1.0 series is branched off from 3.0.0
- 3.2.0 is branched off from 3.1.0



## 0.23 included several major features

- New MapReduce runtime, called MapReduce 2, implemented on a new system called YARN
  - ▣ YARN: Yet Another Resource Negotiator
  - ▣ Replaces the “classic” runtime in previous releases
- HDFS federation
  - ▣ HDFS namespace can be dispersed across multiple name nodes
- HDFS high-availability
  - ▣ Removes name node as a single point of failure; supports standby nodes for failover

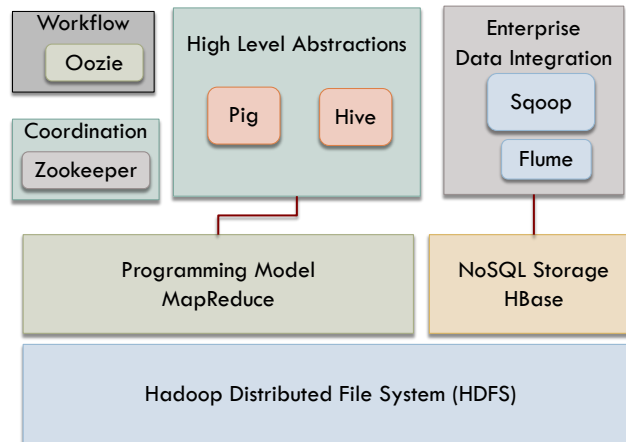


## 3.x includes major features

- Hadoop Submarine support
  - ▣ Hadoop Submarine is a new project that orchestrates Tensorflow programs without modifications on Yarn and provide access to data stored on HDFS
  - ▣ Support for GPUs and Docker images
- Erasure coding in HDFS
- New/Improved storage connectors
  - ▣ ADLS (Azure Datalake Generation 2), Amazon S3, and Amazon DynamoDB
- HDFS storage policies
  - ▣ Hierarchical storage – Archival, Disk (default), SSD, and RamDisk
  - ▣ Users can define the type of storage when storing data
  - ▣ Blocks can be moved between different storage types



## The Hadoop Ecosystem



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.27

27

## MapReduce Jobs

- A MapReduce **Job** is a unit of work
- Consists of:
  - Input Data
  - MapReduce program
  - Configuration information
- Hadoop runs the jobs by dividing it into **tasks**
  - Map tasks
  - Reduce tasks



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.28

28

## Types of nodes that control the job execution process [Older Versions]

- **Job tracker**
  - *Coordinates all jobs* by scheduling tasks to run on task trackers
  - Records overall progress of each job
    - If task fails, reschedule on a different task tracker
- **Task tracker**
  - Run tasks and reports progress to job tracker



## Types of nodes that control the job execution process [Newer Versions]

- Resource Manager
- Application Manager
- Node manager






31

## Processing a weather dataset

- The dataset is from NOAA
- Stored using a line-oriented format
  - ▣ Each line is a record
- Lots of elements being recorded
- We focus on temperature
  - ▣ Always present with a fixed width

 **COLORADO STATE UNIVERSITY** Professor: SHRIDEEP PALICKARA  
COMPUTER SCIENCE DEPARTMENT HADOOP L24.32

32



## Format of a record in the dataset

```
0057
332130      # USAF weather station identifier
99999      # WBAN weather station identifier
19500101   # Observation date
300        # Observation time
4
+51317     # latitude (degrees x 1000)
+028783    # longitude (degrees x 1000)
FM-12
+0171      # elevation (meters)
99999
V020
320        # wind direction (degrees)
1          # quality code
...
-0128     # air temperature (degrees Celsius x 10)
1          # quality code
-0139     # dew point temperature (degree Celsius x 10)
```



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.33

33

## Analyzing the dataset

- What's the highest recorded temperature for each year in the dataset?
- See how programs are written
  - Using Unix tools
  - Using MapReduce



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.34

34

## Using awk Tool for processing line-oriented data

```
#!/usr/bin/env bash
for year in all/*
do
  echo -ne 'basename $year .gz' "\t"
  gunzip -c $year | \
    awk '{ temp=substr($0, 88, 5) + 0;
          q=substr($0, 93, 1);
          if (temp !=9999 && q ~ /[01459]/ &&
              temp > max) max = temp }
        END {print max}'
done
```



## Sample output that is produced

```
% ./max_temperature.sh
1901      317
1902      244
1903      289
1904      256
1905      283
...
```



## To speed things up, we need to be able to do this processing on multiple machines

- STEP 1: **Divide** the work and execute concurrently on multiple machines
- STEP 2: **Combine** results from independent processes
- STEP 3: **Deal with failures** that might take place in the system



## The Hollywood principle **Don't call us, we'll call you.**

- Useful software development technique
- Object's (or component's) initial condition and **ongoing life cycle** is handled by its *environment*, rather than by the object itself
- Typically used for implementing a class/component that must fit into the **constraints of an existing framework**



## Doing the analysis with Hadoop

- Break the processing into two phases
  - ▣ Map and Reduce
  - ▣ Each phase has  $\langle key, value \rangle$  pairs as input and output
- Specify two functions
  - ▣ Map
  - ▣ Reduce



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.39

39

## The map phase

- Choose a Text input format
  - ▣ Each line in the dataset is given as a text *value*
  - ▣ *key* is the *offset of the beginning of the line* from the beginning of the file
- Our map function
  - ▣ Pulls out year and the air temperature
  - ▣ Think of this as a data preparation phase
    - ▣ Reducer will work on data generated by the maps



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.40

40

## How the data is represented in the actual file

```
0067011990999991950051507004...9999999N9+00001+9999999999...  
0043011990999991950051512004...9999999N9+00221+9999999999...  
0043011990999991950051518004...9999999N9-00111+9999999999...  
0043012650999991949032412004...0500001N9+01111+9999999999...  
0043012650999991949032418004...0500001N9+00781+9999999999...
```



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.41

41

## How the lines in the file are presented to the map function by the framework

keys: Line offsets within the file

```
(0, 0067011990999991950051507004...9999999N9+00001+9999999999...)  
(106, 0043011990999991950051512004...9999999N9+00221+9999999999...)  
(212, 0043011990999991950051518004...9999999N9-00111+9999999999...)  
(318, 0043012650999991949032412004...0500001N9+01111+9999999999...)  
(424, 0043012650999991949032418004...0500001N9+00781+9999999999...)
```

The lines are presented to the map function as key-value pairs



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.42

42

## Map function

- **Extract** year and temperature from each record and **emit** output

```
(1950, 0)
(1950, 22)
(1950, -11)
(1949, 111)
(1949, 78)
```



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.43

43

## The output from the map function

- Processed by the MapReduce framework **before** being sent to the reduce function
  - **Sort** and **group**  $\langle \text{key}, \text{value} \rangle$  pairs by key
- In our example, each year appears with a list of all its temperature readings

```
(1949, [111, 78])
(1950, [0, 22, -11])
...
```



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.44

44

## What about the reduce function?

- All it has to do now is iterate through the list supplied by the maps and pick the max reading
- Example output at the reducer?

```
(1949, 111)  
(1950, 22)  
...
```



## What does the actual code to do all of this look like?

- ① Map functionality
- ② Reduce functionality
- ③ Code to run the job



## The map function is represented by an abstract Mapper class

- Declares an abstract `map()` method
- Mapper class is a generic type
  - 4 formal type parameters
  - Specifies input key, input value, output key, and output value



## The Mapper for our example

```
public class MaxTemperatureMapper extends
    Mapper<LongWritable, Text, Text, IntWritable> {
    private final int MISSING = 9999;
    public void map(LongWritable key, Text value, Context context)
        throws IOException, InterruptedException {
        String line = value.toString();
        String year = line.substring(15, 19);
        int airTemperature;
        if (line.charAt(87) == '+' ) {
            airTemperature = Integer.parseInt(line.substring(88, 92));
        } else {
            airTemperature = Integer.parseInt(line.substring(87, 92));
        }
        String quality = line.substring(92, 93);
        if (airTemperature != MISSING && quality.matches("[01459]")) {
            context.write(new Text(year), new IntWritable(airTemperature));
        }
    }
}
```





## Rather than use built-in Java types, Hadoop uses its own set of basic types

- Optimized for **network serialization**
- These are in the `org.apache.hadoop.io` package
  - ▣ `LongWritable` corresponds to Java `Long`
  - ▣ `Text` corresponds to Java `String`
  - ▣ `IntWritable` corresponds to Java `Integer`



## But the `map()` method also had `Context`

- You use this to write the output
- In our example
  - ▣ Year was written as a `Text` object
  - ▣ Temperature was wrapped as an `IntWritable`



## More about Context

- A context object is available at any point of the MapReduce execution
- Provides a convenient mechanism for exchanging required system and job-wide information
- Context coordination happens only when an appropriate phase (driver, map, reduce) of a MapReduce job starts.
  - ▣ Values set by one mapper are not available in another mapper but is available in any reducer



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.51

51

## The reduce function is represented by an abstract Reducer class

- Declares an abstract `reduce ()` method
- `Reducer` class is a generic type
  - ▣ 4 formal type parameters
  - ▣ Used to specify the input and output types of the reduce function
  - ▣ The input types should **match** the *output types of the map function*
    - In the example, `Text` and `IntWritable`



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.52

52

## The Reducer

```
public class MaxTemperatureReducer extends
    Reducer <Text, IntWritable, Text, IntWritable> {

    public void reduce(Text key, Iterable<IntWritable> values,
        Context context)
        throws IOException, InterruptedException {

        int maxValue = Integer.MIN_VALUE;
        for (IntWritable value : values) {
            maxValue = Math.max(maxValue, value.get());
        }
        context.write(key, new IntWritable(maxValue));
    }
}
```



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.53

53

## The code to run the MapReduce job

```
public class MaxTemperature {
    public static main(String[] args) throws Exception {
        Job job = Job.getInstance();
        job.setJarByClass(MaxTemperature.class);
        job.setJobName("Max temperature");

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        job.setMapperClass(MaxTemperatureMapper.class);
        job.setReducerClass(MaxTemperatureReducer.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

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



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.54

54

## Details about the Job submission

[1/3]

- Code must be packaged in a JAR file for Hadoop to distribute over the cluster
  - `setJarByClass()` causes Hadoop *to locate relevant JAR file* by looking for JAR that contains this class
- Input and output paths must be specified next
  - `addInputPath()` can be *called more than once*
  - `setOutputPath()` specifies the output directory
    - Directory *should not exist* before running the job
    - Precaution to prevent data loss



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.55

55

## Details about the Job submission

[2/3]

- The methods `setOutputKeyClass()` and `setOutputValueClass()`
  - Control the output types of the map and reduce functions
  - If they are different?
    - Map output types can be set using `setMapOutputKeyClass()` and `setMapOutputValueClass()`



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.56

56

## Details about the Job submission.

[3/3]

- The `waitForCompletion()` method **submits** the job and **waits** for it to complete
  - The `boolean` argument is a *verbose* flag; if set, progress information is printed on the console
- Return value of `waitForCompletion()` indicates success (`true`) or failure (`false`)
  - In the example this is the program's exit code (0 or 1)



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.57

57

## API DIFFERENCES

COMPUTER SCIENCE DEPARTMENT



COLORADO STATE UNIVERSITY

58

## The old and new MapReduce APIs

- The new API favors abstract classes over interfaces
  - ▣ Make things easier to evolve
- New API is in `org.apache.hadoop.mapreduce` package
  - ▣ Old API can be found in `org.apache.hadoop.mapred`
- New API makes use of context objects
  - ▣ Context unifies roles of `JobConf`, `OutputCollector`, and `Reporter` from the old API



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.59

59

## The old and new MapReduce APIs

- In the new API, job control is done using the `Job` class rather than using the `JobClient`
- Output files are named slightly differently
  - ▣ Old API: Both map and reduce outputs are named `part-nnnn`
  - ▣ New API: Map outputs are named `part-m-nnnn` and reduce outputs are named `part-r-nnnn`



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.60

60

## The old and new MapReduce APIs

- The new API's `reduce()` method passes values as `Iterable` rather than as `Iterator`
  - Makes it **easier to iterate** over values using the `for-each` loop construct

```
for (VALUEIN value: values) {  
    ...  
}
```



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.61

61

## The contents of this slide-set are based on the following references

- Jeffrey Dean, Sanjay Ghemawat: *MapReduce: Simplified Data Processing on Large Clusters*. OSDI 2004: 137-150
- Jeffrey Dean, Sanjay Ghemawat: *MapReduce: simplified data processing on large clusters*. *Commun. ACM* 51(1): 107-113 (2008)
- Tom White. *Hadoop: The Definitive Guide. 3<sup>rd</sup> Edition. Early Access Release*. O'Reilly Press. ISBN: 978-1-449-31152-0. Chapters 1 and 2.
- Boris Lublinsky, Kevin Smith, and Alexey Yakubovich. *Professional Hadoop Solutions*. Wiley Press. Chapter 3.



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA  
COMPUTER SCIENCE DEPARTMENT

HADOOP

L24.62

62