

# Hadoop MapReduce

Claudio Scheer<sup>1</sup>   Gabriell Araujo<sup>1</sup>

<sup>1</sup>Master's Degree in Computer Science  
Pontifical Catholic University of Rio Grande do Sul - PUCRS

High Performance for Big Data Applications

# Table of Contents

- 1 Hadoop
- 2 General
- 3 Examples

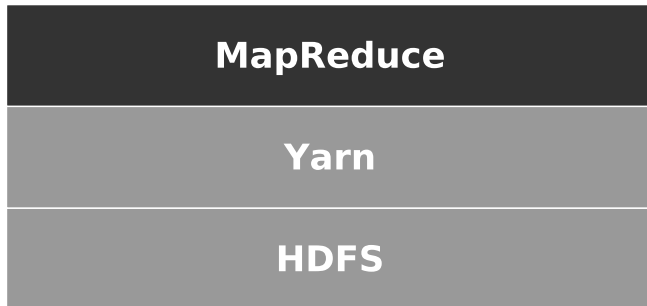
# Table of Contents

1 Hadoop

2 General

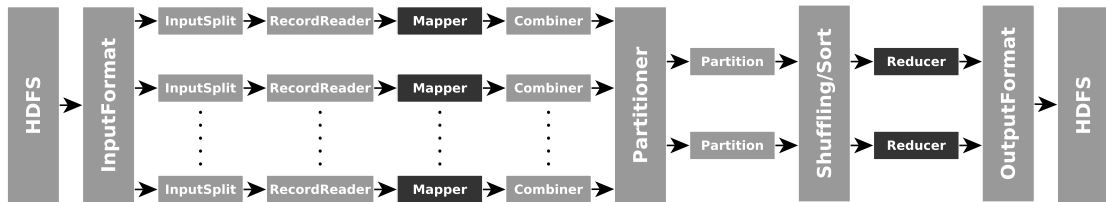
3 Examples

# What we mean by Hadoop



<https://data-flair.training/blogs/hadoop-ecosystem-components>

# MapReduce execution flow



<https://data-flair.training/blogs/hadoop-ecosystem-components>

# Custom data types

- LongWritable = long;
- IntWritable = int;
- Text = String;
- Other data types ([link](#));

```
public class IntWritable implements WritableComparable<IntWritable> {
    private int value;

    public IntWritable(int value) { set(value); }

    public void set(int value) { this.value = value; }

    public int get() { return value; }

    @Override
    public void readFields(DataInput in) throws IOException {
        value = in.readInt();
    }

    @Override
    public void write(DataOutput out) throws IOException {
        out.writeInt(value);
    }

    @Override
    public int compareTo(IntWritable o) {
        int thisValue = this.value;
        int thatValue = o.value;
        return (thisValue < thatValue ? -1 : (thisValue == thatValue ? 0 : 1));
    }
}
```

► Source

- TextInputFormat: <LongWritable, Text>
- KeyValueTextInputFormat: <Text, Text>
  - Key splitted by \t;
- NLineInputFormat: <LongWritable, Text>
  - `config.setInt(NLineInputFormat.LINES_PER_MAP, 256);`
- Custom InputFormat must implement `getSplits` and `getRecordReader`;

<https://hadoop.apache.org/docs/stable/api/org/apache/hadoop/mapred/InputFormat.html>

- Defines the level of parallelism;
- Splitted by the size of the block;
  - $10\text{TB}/128\text{MB} = 82000$
- It is a logical division of the input data;

<https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html>



- Split InputSplit into <key, value> pairs;
  - Custom implementations can read values from outside the InputSplit;
- Lines greater than the max record length are ignored:
  - `config.setInt(LineRecordReader.MAX_LINE_LENGTH, Integer.MAX_VALUE);`
- Vanilla example:

```
public void run(Context context) {  
    setup(context);  
    while (context.nextKeyValue()) {  
        map(context.setCurrentKey(), context.getCurrentValue(), context);  
    }  
    cleanup(context);  
}
```

- Maps  $\langle \text{key1}, \text{value1} \rangle$  to  $\langle \text{key2}, \text{value2} \rangle$ ;
- Vanilla example:

```
public class SimpleMapper extends Mapper<LongWritable, Text, Text, IntWritable> {  
    private IntWritable one = new IntWritable(1);  
    private Text word = new Text();  
  
    @Override  
    public void map(LongWritable key, Text value, Context context) {  
        StringTokenizer itr = new StringTokenizer(value.toString());  
        while (itr.hasMoreElements()) {  
            word.set(itr.nextToken());  
            context.write(word, one);  
        }  
    }  
}
```

<https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html>

# Combiner

- Reduces data transfer between mapper and reducer;
- Reduces the amount of data to be processed in the reducer;



# Partitioner

- Redirects the Combiner output to a specific reducer;
- Has the same number as the number of reducers;
- Used only when there is more than one reducer;
- Vanilla example:

```
public class StupidPartitioner extends Partitioner<Text, IntWritable> {  
    public int getPartition(Text key, IntWritable value, int numPartitions) {  
        if (value.get() < 35) {  
            return 0;  
        } else {  
            return 1;  
        }  
    }  
}
```

<https://hadoop.apache.org/docs/stable/api/org/apache/hadoop/mapreduce/Partitioner.html>

- Collects output from the mappers to the reducers using HTTP requests;
- Sorts the collected  $\langle \text{key}, \text{value} \rangle$  pairs based on the key;
  - `job.setSortComparatorClass(StupidSortComparator.class);`

<https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/PluggableShuffleAndPluggableSort.html>

<https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/EncryptedShuffle.html>

- Reduces  $\langle \text{key2}, \text{list}(\text{value2}) \rangle$  to  $\langle \text{key3}, \text{value3} \rangle$ ;
- Change the number of reducers:
  - `job.setNumReduceTasks(2);`
- Vanilla example:

```
public static class SimpleReducer extends Reducer<Text, IntWritable, Text, IntWritable> {  
    private IntWritable result = new IntWritable();  
  
    @Override  
    public void reduce(Text key, Iterable<IntWritable> values, Context context) {  
        int sum = 0;  
        for (IntWritable val : values) {  
            sum += val.get();  
        }  
        result.set(sum);  
        context.write(key, result);  
    }  
}
```

<https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html>

[https://docs.cloudera.com/HDPDocuments/HDP2/HDP-2.6.5/bk\\_command-line-installation/content/determine-hdp-memory-config.html](https://docs.cloudera.com/HDPDocuments/HDP2/HDP-2.6.5/bk_command-line-installation/content/determine-hdp-memory-config.html)

- Specifies how reducer output will be written;
  - `config.set(TextOutputFormat.SEPARATOR, “,”);`
- Allows to write multiple outputs;
  - <https://hadoop.apache.org/docs/stable/api/org/apache/hadoop/mapred/lib/MultipleOutputs.html>

# Table of Contents

1 Hadoop

2 General

3 Examples



# How many maps?

- Usually defined by the size of the input;
- Around 10-100 maps per-node;

# Change the number of maps

- Change the block size;
  - Describe how to change;

# How many reducers?

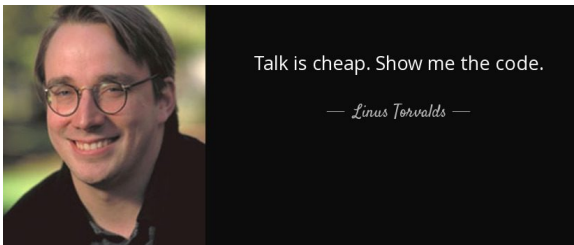
- 0.95 or 1.75 multiplied by ( $\text{<number of nodes> * <number of maximum containers per node>}$ );
  - 1.75 has a much better load balancing;
  - Scale factor are not using whole numbers to reserve a few slots for speculative-tasks and failed tasks;

[https://docs.cloudera.com/HDPDocuments/HDP2/HDP-2.6.5/bk\\_command-line-installation/content/determine-hdp-memory-config.html](https://docs.cloudera.com/HDPDocuments/HDP2/HDP-2.6.5/bk_command-line-installation/content/determine-hdp-memory-config.html)

- Change the block size;
  - Describe how to change;

- how to interpret log output - how to monitoring - how to know which nodes are being used

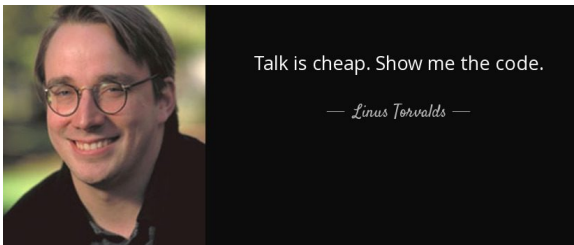
Configs - `mapreduce.job.running.map.limit` - The maximum number of simultaneous map tasks per job. There is no limit if this value is 0 or negative. - `mapreduce.job.max.map` - Limit on the number of map tasks allowed per job. There is no limit if this value is negative. - `mapreduce.input.fileinputformat.split.maxsize` - split size



# Table of Contents

- 1 Hadoop
- 2 General
- 3 Examples

# Unquestionable!





This is a text in first column.

$$E = mc^2$$

- First item
- Second item

This text will be in the second column and on a second thought this is a nice looking layout in some cases.