(Big) Data Engineering In Depth

From Beginner to Professional

Moustafa Alaa

Senior Data Engineer at Onfido, London, UK

The Definitive Guide to Big Data Engineering Tasks

Previous video recap!

Hadoop Core Concepts

- HDFS.

Hadoop Core Concepts

- HDFS.
- YARN.

Hadoop Core Concepts

- HDFS.
- YARN.
- Map-Reduce.

Hadoop Map Reduce

Introduction To Hadoop Map Reduce API

The basic idea of MapReduce

We break this into three stages

- Map.

[1] This example taken from https://reberhardt.com/cs110/summer-2018/lecture-notes/lecture-14/

The basic idea of MapReduce

We break this into three stages

- Map.
- Shuffle/Group (Mapper Intermediates).

[1] This example taken from https://reberhardt.com/cs110/summer-2018/lecture-notes/lecture-14/

The basic idea of MapReduce

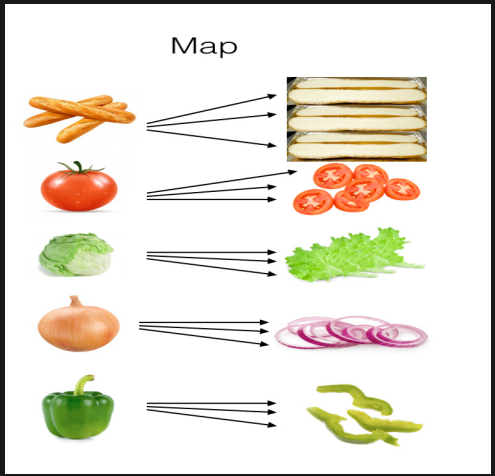
We break this into three stages

- Map.
- Shuffle/Group (Mapper Intermediates).
- Reduce

[1] This example taken from https://reberhardt.com/cs110/summer-2018/lecture-notes/lecture-14/

Map

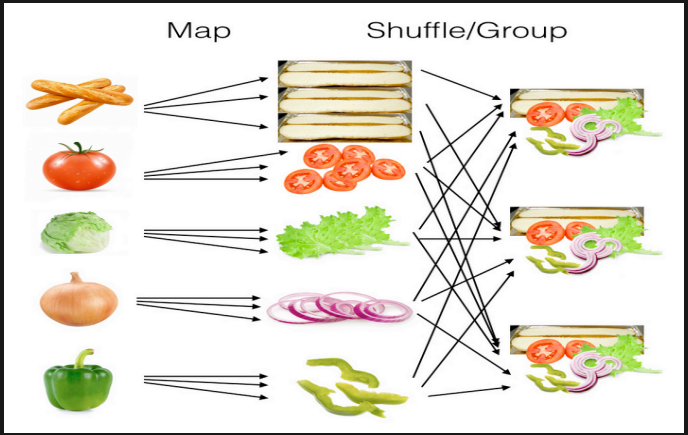
We distribute our raw ingredients amongst the workers.



[1] This example taken from https://reberhardt.com/cs110/summer-2018/lecture-notes/lecture-14/

Shuffle/Group

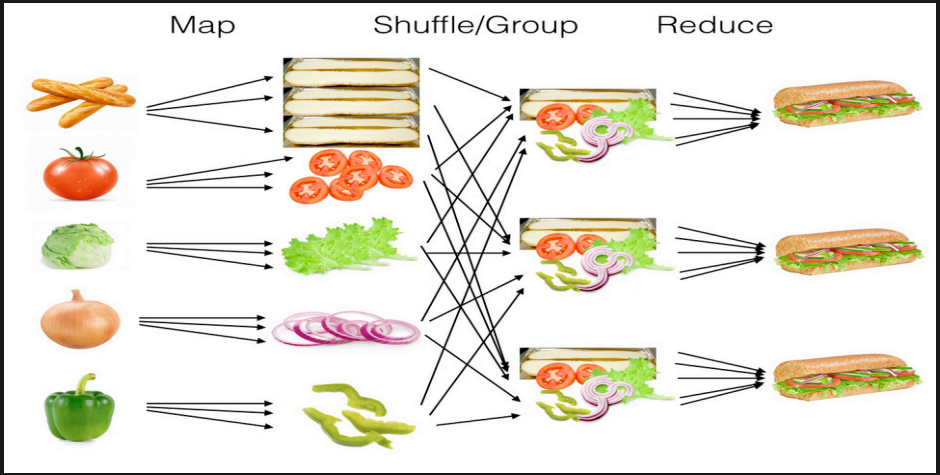
We will organise and group the processed ingredients into piles, so that making a sandwich becomes easy.



[1] This example taken from https://reberhardt.com/cs110/summer-2018/lecture-notes/lecture-14/

Reduce

we'll combine the ingredients into a sandwich



[1] This example taken from https://reberhardt.com/cs110/summer-2018/lecture-notes/lecture-14/

Case Study Example 1

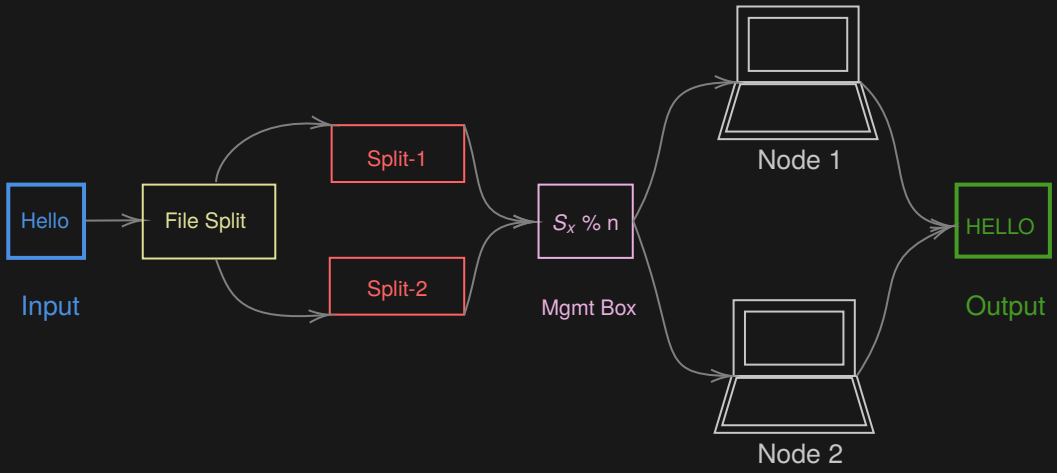
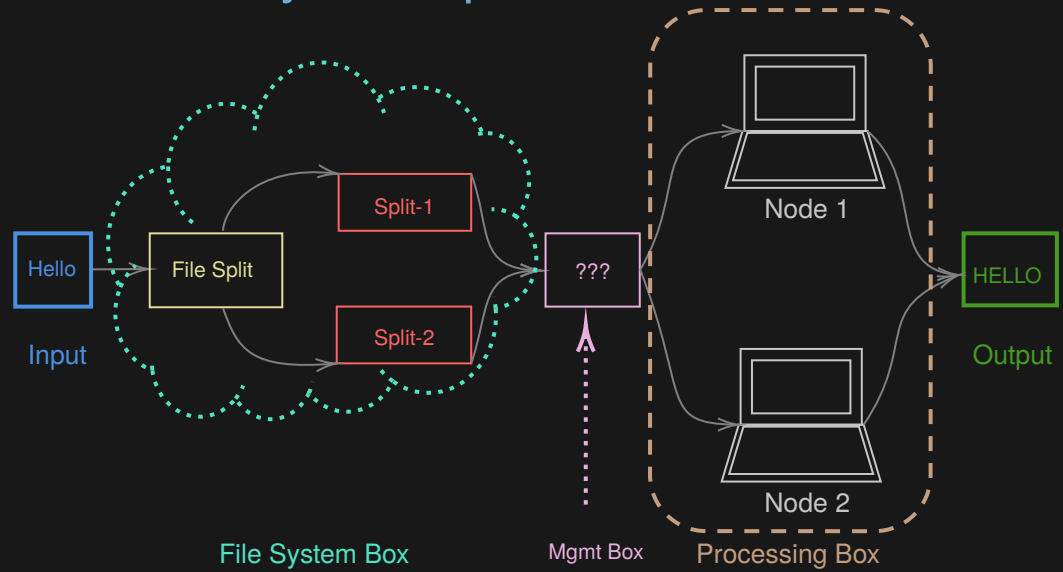
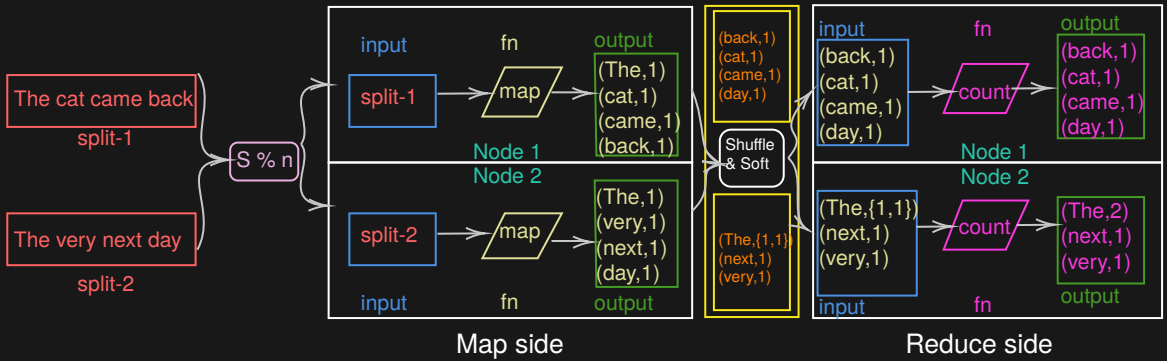


Figure: Convert text to upper text, for example, The -> THE

Case Study Example 1



Case Study Example 2



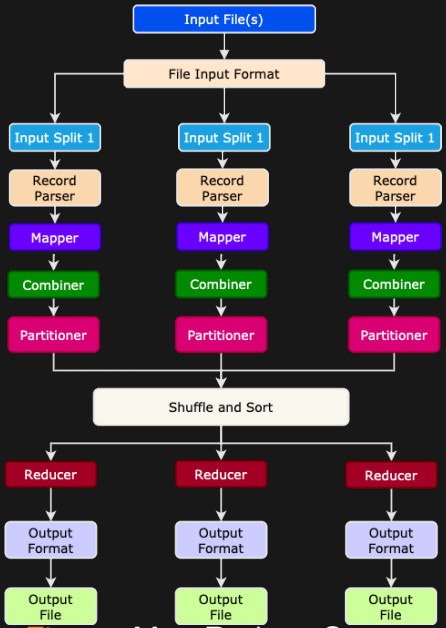


Figure: Map Reduce Stages

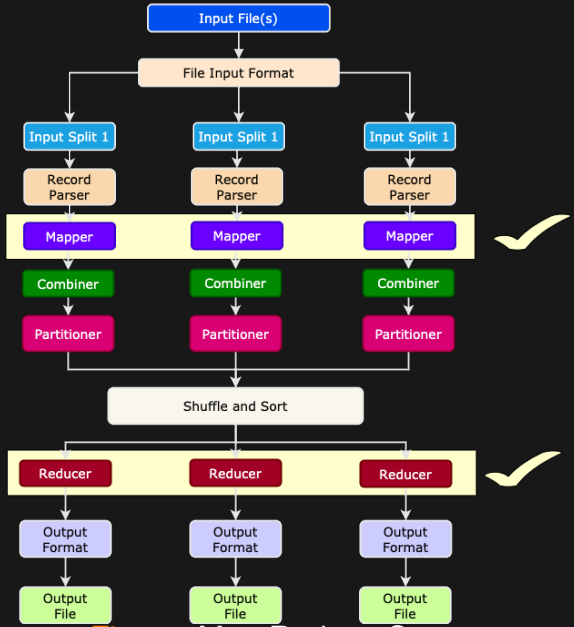


Figure: Map Reduce Stages

Map Reduce (word count) Deep Dive

The Map-Reduce consists of three ”main” parts

- The Driver.

Map Reduce (word count) Deep Dive

The Map-Reduce consists of three ”main” parts

- The Driver.
- The Mapper.

Map Reduce (word count) Deep Dive

The Map-Reduce consists of three ”main” parts

- The Driver.
- The Mapper.
- The Reducer.

Hadoop Map Reduce API

Hadoop Map Reduce API Deep Dive

The Driver

- The code that runs on the client machine configures the job details by creating an object from the `Job` class, which implements the `JobContext` interface.

The Driver

- The code that runs on the client machine configures the job details by creating an object from the `Job` class, which implements the `JobContext` interface.

- It submits the job to the cluster.

The Driver

- The code that runs on the client machine configures the job details by creating an object from the `Job` class, which implements the `JobContext` interface.
- It submits the job to the cluster.
- It parses job arguments to identify job parameters, for example, input/output directories..

The Driver: Job Configuration

The `Job` object allows you to set configuration for your `Map-Reduce` job:

- You can configure the `Mapper` & the `Reducer` classes.

The Driver: Job Configuration

The `Job` object allows you to set configuration for your `Map-Reduce` job:

- You can configure the `Mapper` & the `Reducer` classes.
- Set the `Mapper` input/output key & value data types.

The Driver: Job Configuration

The `Job` object allows you to set configuration for your `Map-Reduce` job:

- You can configure the `Mapper` & the `Reducer` classes.
- Set the `Mapper` input/output key & value data types.
- Set the `Reducer` input/output key & value data types.

The Driver: Job Configuration

- We can configure file input directory and output.

The Driver: Job Configuration

- We can configure file input directory and output.
- We configure the output path using `FileOutputFormat.setOutputPath()` to specify the reducers' directory to write the output data.

The Driver: Job Configuration

- We configure the input path using `FileInputFormat.setInputPaths()`, and by default, it will read all the files in the specified directories and send them to the mappers.

[1] For more details, please read HTDG. Ch.3 File patterns and PathFilter sections.

The Driver: Job Configuration

- We configure the input path using `FileInputFormat.setInputPaths()`, and by default, it will read all the files in the specified directories and send them to the mappers.

- We can use **`Hadoop glob patterns`** to read directory patterns, for example, */warehouse/public/sales**.

[1] For more details, please read HTDG. Ch.3 File patterns and PathFilter sections.

The Driver: Job Configuration

- We configure the input path using `FileInputFormat.setInputPaths()`, and by default, it will read all the files in the specified directories and send them to the mappers.
- We can use `Hadoop glob patterns` to read directory patterns, for example, */warehouse/public/sales**.
- We can call `FileInputFormat.addInputPath()` to multiple times by specifying a single file or directory.

[1] For more details, please read HTDG. Ch.3 File patterns and PathFilter sections.

Hadoop Map Reduce API

Please read HTDG. Ch.3 The Java Interface

The Driver: Job Configuration

– You could set driver configurations globally using Hadoop configurations.

The Driver: Job Configuration

- You could set driver configurations globally using Hadoop configurations.
- Any options not specified in the job configuration will use the Hadoop default values.

The Driver: Job Configuration

- You could set driver configurations globally using Hadoop configurations.
- Any options not specified in the job configuration will use the Hadoop default values.
- We use the `Job` object to specify the job name and check its state..

The Driver: Job Configuration

- It is optional to set the mapper and reducer classes.

The Driver: Job Configuration

- It is optional to set the mapper and reducer classes.
- Hadoop uses its default `IdentityMapper` and `IdentityReducer`.

The Driver: Job Configuration

Lunch a Map-Reduce job:

- The `waitForCompletion()` method in the `Job` class launches the job and polls for progress. In addition, it writes the logs and summarizing the Map-Reduce job progress and changes.

The Driver: Job Configuration

Lunch a Map-Reduce job:

- The `waitForCompletion()` method in the `Job` class launches the job and polls for progress. In addition, it writes the logs and summarizing the Map-Reduce job progress and changes.

- When the job completes successfully, the job counters are displayed. Otherwise, the error that caused the job to fail is logged to the console.

InputFormat

- TheThe driver defines the `InputFormat`; then the `InputFormat` creates a `RecordReader"` object that parses the input data into key/value pairs passed to the mapper.

InputFormat

- TheThe driver defines the `InputFormat`; then the `InputFormat` creates a `RecordReader"` object that parses the input data into key/value pairs passed to the mapper.
- For example: `TextInputFormat`:

InputFormat

- TheThe driver defines the `InputFormat`; then the `InputFormat` creates a `RecordReader"` object that parses the input data into key/value pairs passed to the mapper.
- For example: `TextInputFormat`:
  - It is the default.

InputFormat

- TheThe driver defines the `InputFormat`; then the `InputFormat` creates a `RecordReader"` object that parses the input data into key/value pairs passed to the mapper.
- For example: `TextInputFormat`:
  - It is the default.
  - It creates `LineRecordReader` objects.

InputFormat

- TheThe driver defines the `InputFormat`; then the `InputFormat` creates a `RecordReader"` object that parses the input data into key/value pairs passed to the mapper.
- For example: `TextInputFormat`:
  - It is the default.
  - It creates `LineRecordReader` objects.
  - Key: is the line offest in the file.

InputFormat

- TheThe driver defines the `InputFormat`; then the `InputFormat` creates a `RecordReader"` object that parses the input data into key/value pairs passed to the mapper.
- For example: `TextInputFormat`:
  - It is the default.
  - It creates `LineRecordReader` objects.
  - Key: is the line offest in the file.
  - Value: is the line which terminated by ”\n”.

Keys and Values

- Keys and Values in Hadoop are java `Objects` not `Java primitives types`.

Keys and Values

- Keys and Values in Hadoop are java `Objects` not `Java primitives types`.
- Values are objects which implement `Writable`.

Keys and Values

- Keys and Values in Hadoop are java `Objects` not `Java primitives types`.
- Values are objects which implement `Writable`.
- Keys are objects which implement `WritableComparable`.

What is Writable?

- `Writable` is an interface in Hadoop.

What is Writable?

- `Writable` is an interface in Hadoop.
- `Writables` are used for data type ”serialization” in Hadoop to translate/serialize ”primitive java data types” to ”Hadoop data types”, Ex: int to IntWritable and String to Text.

What is Writable?

- `Writable` is an interface in Hadoop.
- `Writables` are used for data type ”serialization” in Hadoop to translate/serialize ”primitive java data types” to ”Hadoop data types”, Ex: int to IntWritable and String to Text.
- Hadoop uses the `Writable` interface for data transfer in the cluster and network.

What is WritableComparable?

- A `WritableComparable` is a `Writable` which is also `Comparable`.

What is WritableComparable?

- A `WritableComparable` is a `Writable` which is also `Comparable`.

- We can compare two `WritableComparables` against each other to determine their ***order***, for example, we could need to compare the order of two Text ”Apple vs. Cat or numbers ordering” to understand the ordering mechanism.

What is WritableComparable?

- A `WritableComparable` is a `Writable` which is also `Comparable`.
- We can compare two `WritableComparables` against each other to determine their ***order***, for example, we could need to compare the order of two Text ”Apple vs. Cat or numbers ordering” to understand the ordering mechanism.
- Obviously, the reason we have Keys to be `WritableComparable` is that they are passed to the reducer in ***sorted order***.

What is WritableComparable?

- A `WritableComparable` is a `Writable` which is also `Comparable`.
- We can compare two `WritableComparables` against each other to determine their ***order***, for example, we could need to compare the order of two Text ”Apple vs. Cat or numbers ordering” to understand the ordering mechanism.
- Obviously, the reason we have Keys to be `WritableComparable` is that they are passed to the reducer in ***sorted order***.
- Note: All Hadoop implemented types are both `Writable` and `WritableComparable`.

Map Reduce (word count) Deep Dive

The Map-Reduce example consists of three main parts

- ~~The Driver~~.

Map Reduce (word count) Deep Dive

The Map-Reduce example consists of three main parts

- ~~The Driver~~.
- The Mapper.

The Mapper

- The mapper class deals with a single input split.

The Mapper

- The mapper class deals with a single input split.
- All mapper classes must extend the `Mapper` base class.

The Mapper

- The mapper class deals with a single input split.
- All mapper classes must extend the `Mapper` base class.
- All mapper must specify the key and values for input and output.

The Mapper

- The mapper class deals with a single input split.
- All mapper classes must extend the `Mapper` base class.
- All mapper must specify the key and values for input and output.
- All mappers must override the `map` method and pass the key, value, and `Context`.

The Mapper

- The mapper class deals with a single input split.
- All mapper classes must extend the `Mapper` base class.
- All mapper must specify the key and values for input and output.
- All mappers must override the `map` method and pass the key, value, and `Context`.
- The `Context` is used to write intermediate data and all information about the job's configurations.

Map Reduce (word count) Deep Dive

The Map-Reduce example consists of three main parts

- ~~The Driver~~.

Map Reduce (word count) Deep Dive

The Map-Reduce example consists of three main parts

- ~~The Driver~~.
- ~~The Mapper~~.

Map Reduce (word count) Deep Dive

The Map-Reduce example consists of three main parts

- ~~The Driver~~.
- ~~The Mapper~~.
- The Reducer.

The Reducer

- The Reducer receives a Key and an Iterable collection of Writable objects. It also receives a Context object.

The Reducer

- The Reducer receives a Key and an Iterable collection of Writable objects. It also receives a Context object.
- All reducers classes must extend the `Reducer` base class.

The Reducer

- The Reducer receives a Key and an Iterable collection of Writable objects. It also receives a Context object.
- All reducers classes must extend the `Reducer` base class.
- All mapper must specify the key and values for intermediate input and final (or intermediate) output.

The Reducer

- The Reducer receives a Key and an Iterable collection of Writable objects. It also receives a Context object.
- All reducers classes must extend the `Reducer` base class.
- All mapper must specify the key and values for intermediate input and final (or intermediate) output.
- All reducers must override the ”reduce” method and pass the key, `Iterable` and ”Context”.

Hadoop Map Reduce API

Map Reduce Demo

Thank you for watching!

See you in the next video ☺