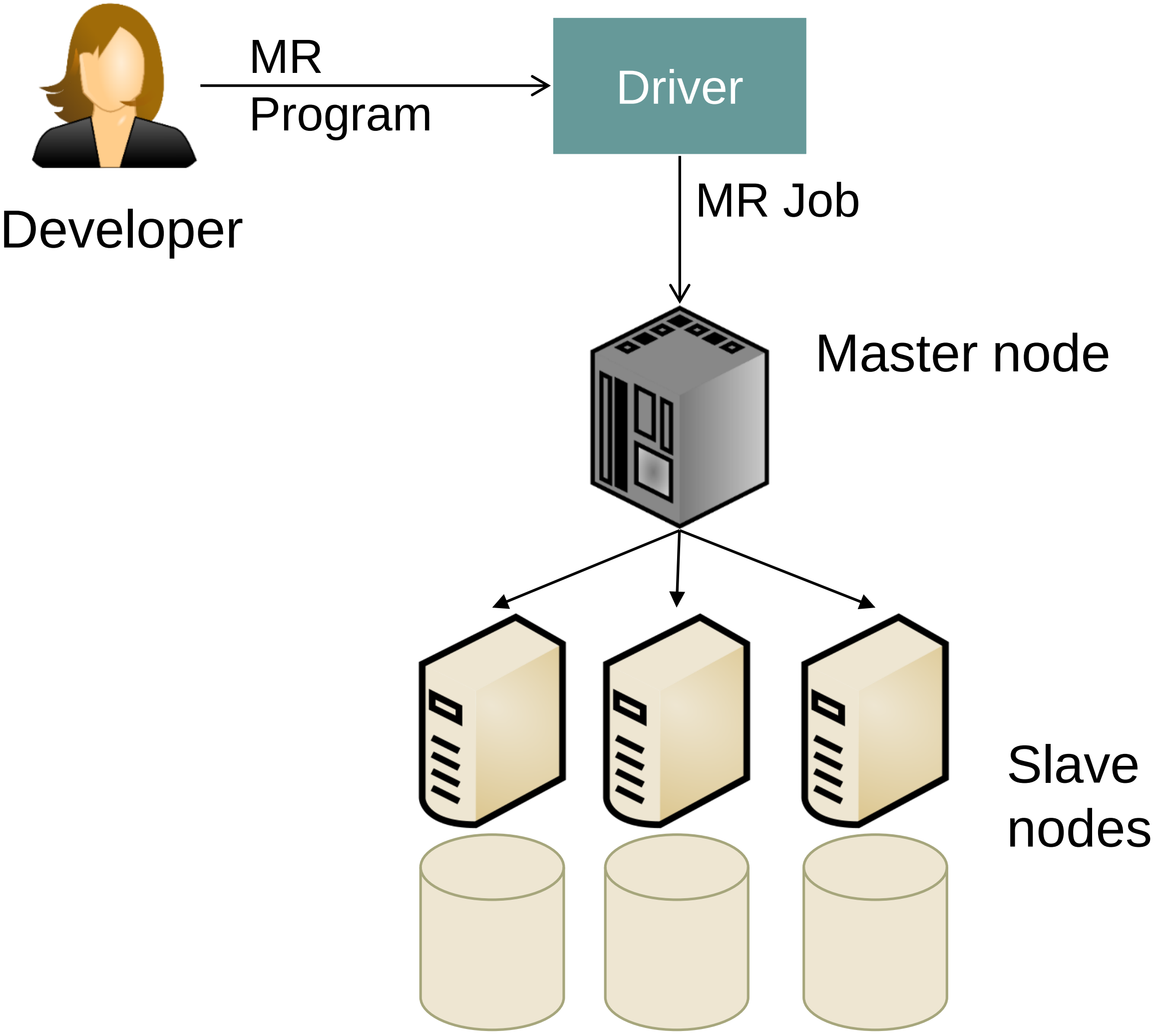
Hadoop Map Reduce

MapReduce

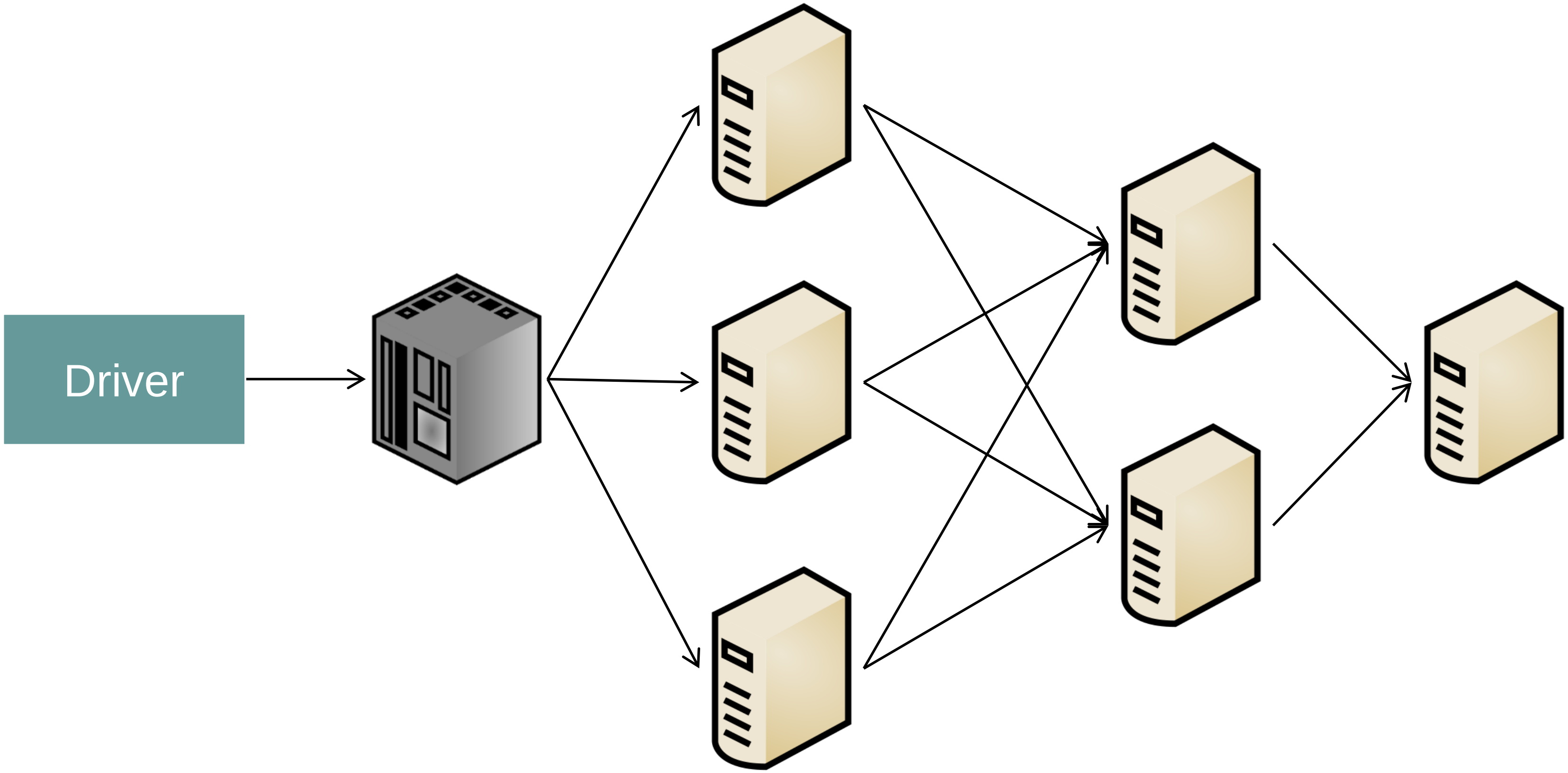
- 2-in-1
  - A programming paradigm
  - A query execution engine
- A kind of functional programming
- We focus on the MapReduce execution engine of Hadoop through YARN

Overview



Code Example

Job Execution Overview



Job submission

Job preparation

Map

Shuffle

Reduce

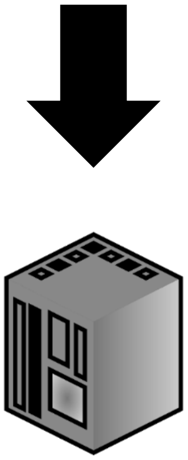
Cleanup

Job Submission

- Execution location: Driver node
- A driver machine should have the following
  - Compatible Hadoop binaries
  - Cluster configuration files
  - Network access to the master node
- Collects job information from the user
  - Input and output paths
  - Map, reduce, and any other functions
  - Any additional user configuration
- Packages all this in a Hadoop Configuration

Hadoop Configuration

Key: String	Value: String
Input	hdfs://user/eldawy/README.txt
Output	hdfs://user/eldawy/wordcount
Mapper	edu.ucr.cs.cs226.eldawy.WordCount
Reducer	…
JAR File	…
User-defined	User-defined

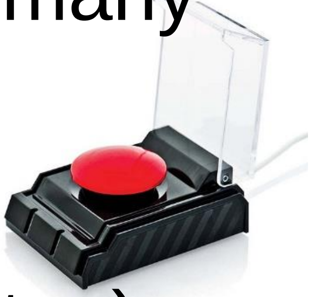


Serialized over network

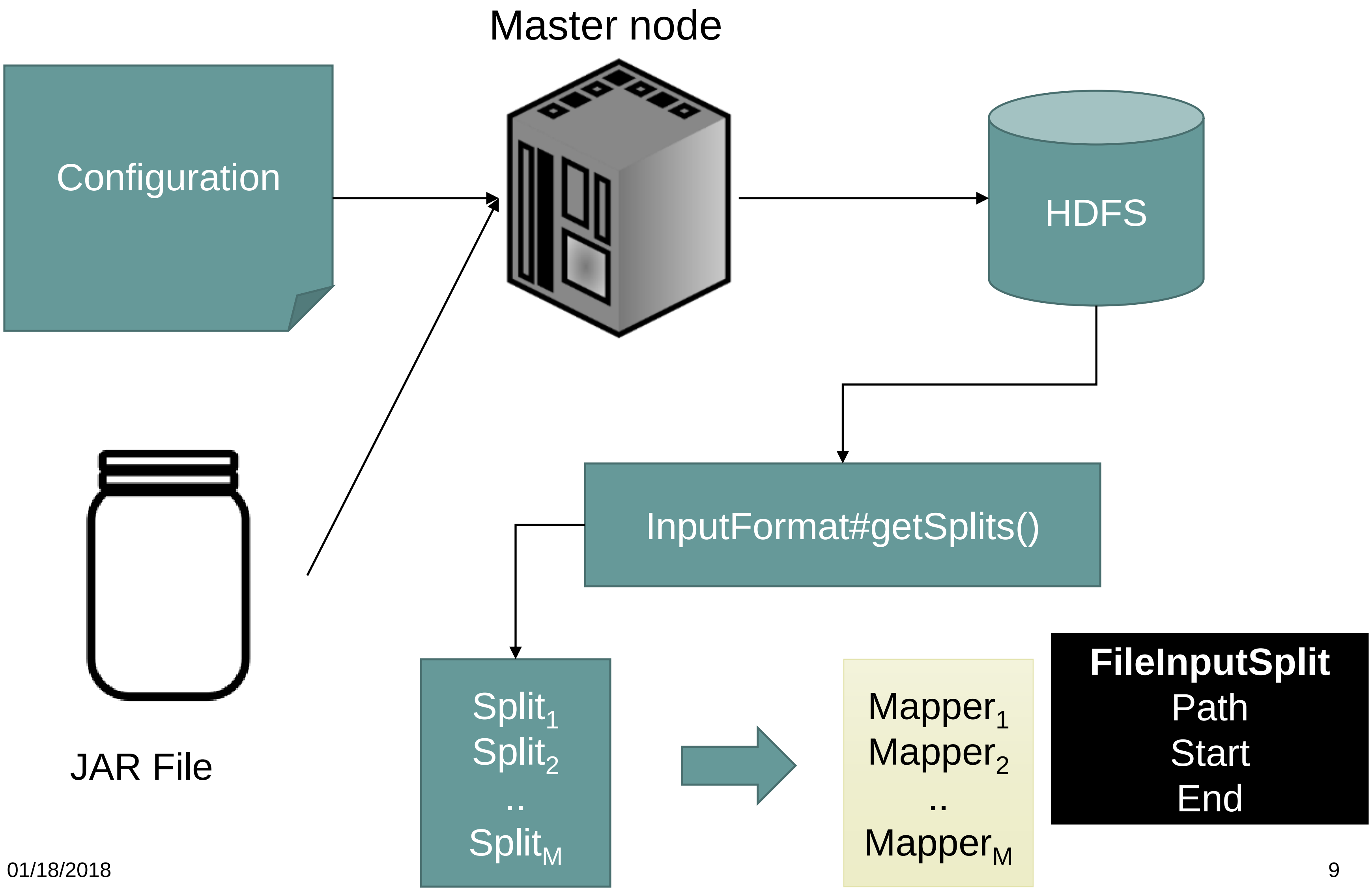
Master node

Job Preparation

- Runs on the master node
- Gets the job ready for parallel execution
- Collects the JAR file that contains the user-defined functions, e.g., Map and Reduce
- Writes the JAR and configuration to HDFS to be accessible by the executors
- Looks at the input file(s) to decide how many map tasks are needed
- Makes some sanity checks
- Finally, it pushes the BRB (Big Red Button)



Job Preparation



Master node

Configuration

JAR File

HDFS

InputFormat#getSplits()

Split$_1$
Split$_2$
..
Split$_M$

Mapper$_1$
Mapper$_2$
..
Mapper$_M$

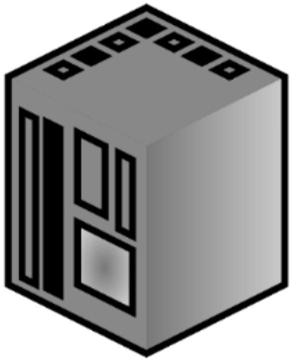
FileInputSplit
Path
Start
End

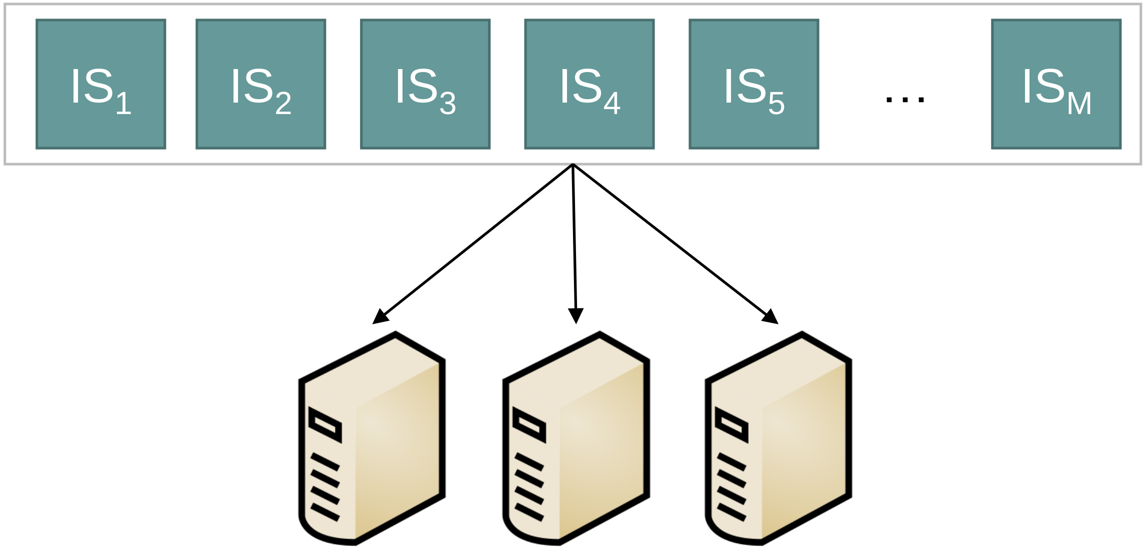
Map Phase

- Runs in parallel on worker nodes
- M Mappers:
  - Read the input
  - Apply the map function
  - Apply the combine function (if configured)
  - Store the map output
- There is no guaranteed ordering for processing the input splits

Map Phase

Master node





Mapper

- Reads the job configuration and task information (mostly, InputSplit)
- Instantiates an object of the Mapper class
- Instantiates a record reader for the assigned input split
- Calls Mapper#setup(Context)
- Reads records one-by-one from the record reader and passes them to the map function
- The map function writes the output to the context

MapContext

- Keeps track of which input split is being read and which records are being processed
- Holds all the job configuration and some additional information about the map task
- Materializes the map output