Introduction to Spark

Ahmed Eldawy

Distributed Data Processing

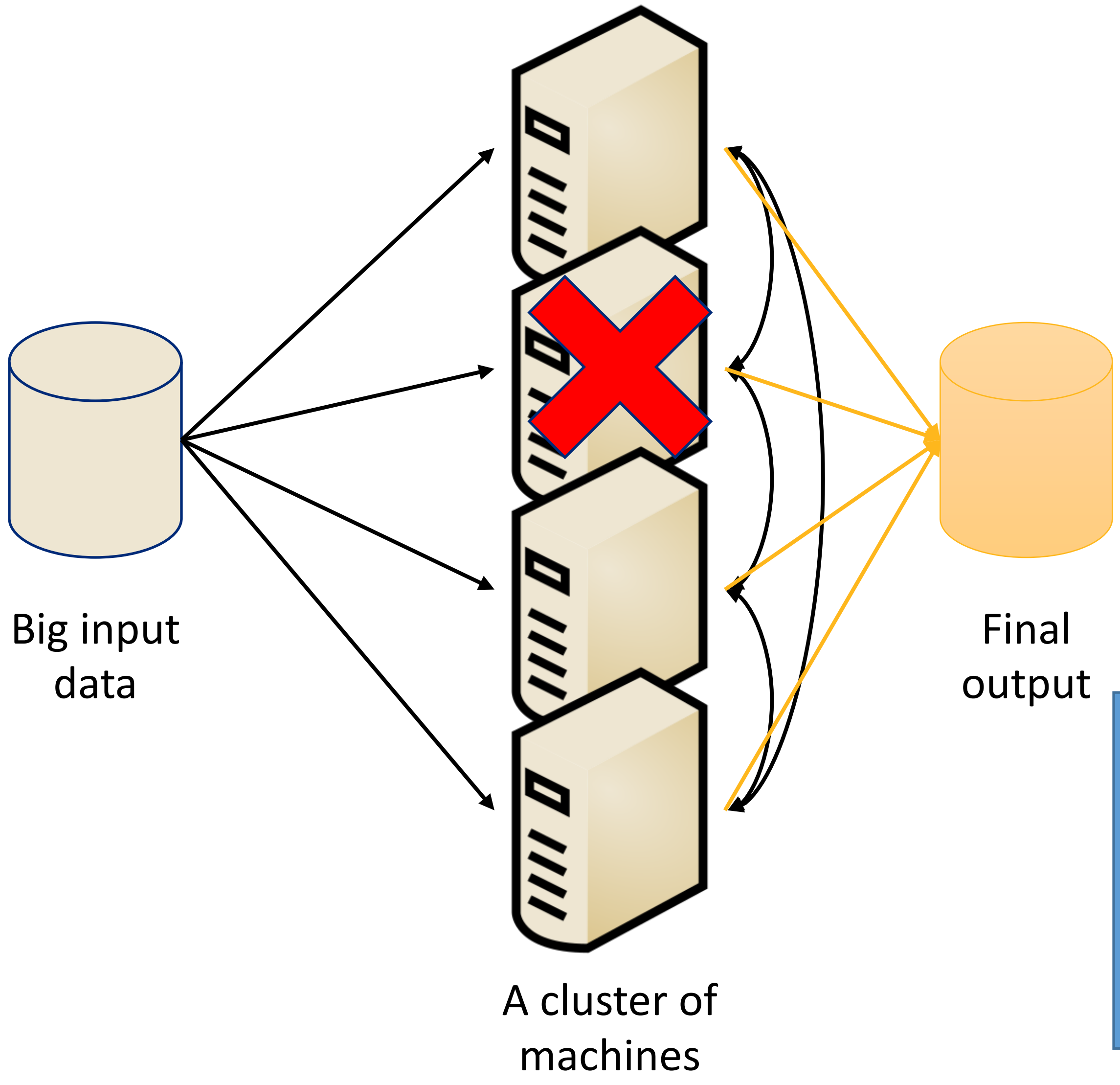
- The idea of distributed databases is older than you might think

  Richard Peebles, Eric G. Manning: A Computer Architecture for Large (Distributed) Data Bases. **VLDB 1975**: 405-427

- Distributed data structures and algorithms have always been around

- So, what is new?

Distributed Data Processing



Data partitioning
Load balancing
Fault tolerance
Synchronization

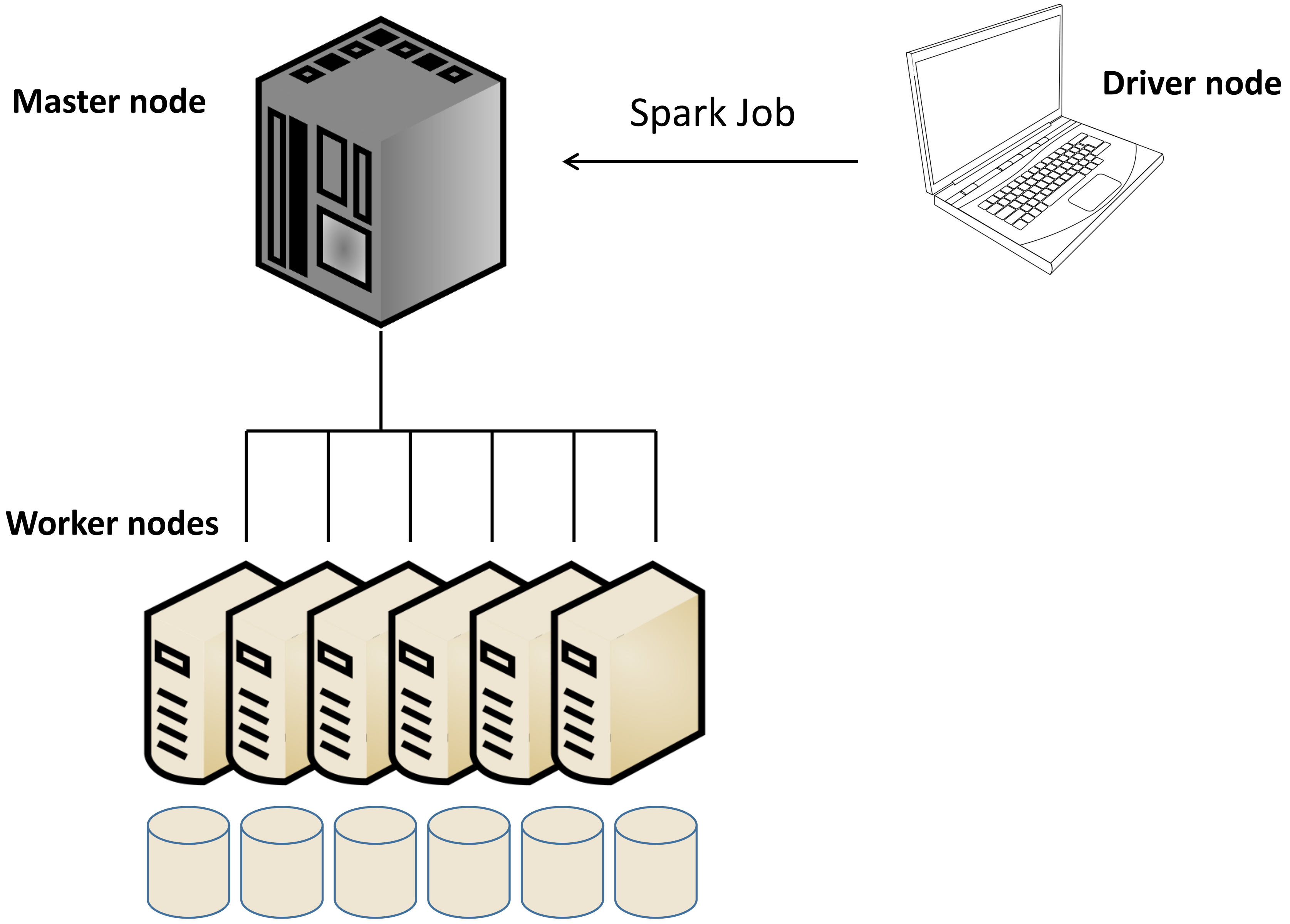
MapReduce

- A programing paradigm for expressing distributed algorithms
- Introduced by Google in 2004
  - Google File System for distributed storage
  - Google MapReduce for distributed processing
- Hadoop is the open source counterpart released in 2007 and contributed mainly by Yahoo!
  - HDFS
  - Hadoop MapReduce

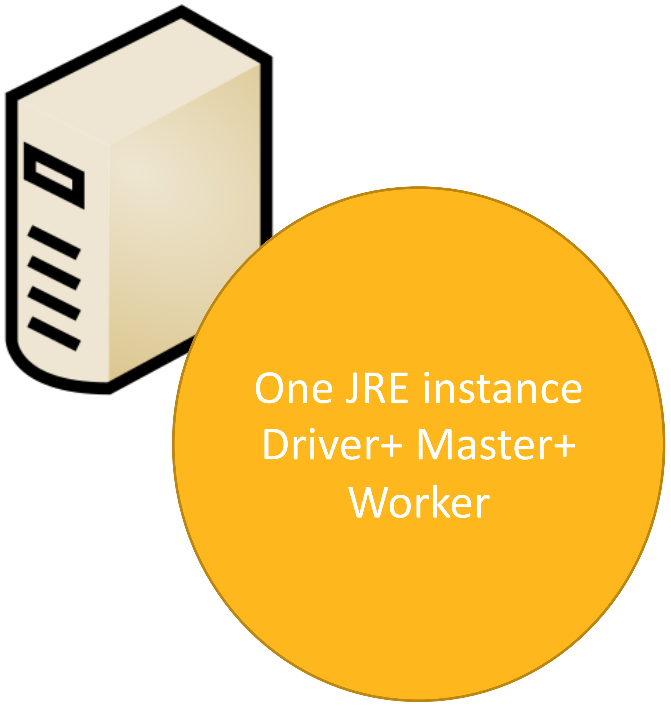
Spark

- Hadoop and MapReduce were a perfect research vehicle
- They helped in framing what we really want in a big data system
- Spark came as a new system designed from scratch to satisfy the real need of big data
- A distributed shared-nothing system
- Uses a functional programming paradigm

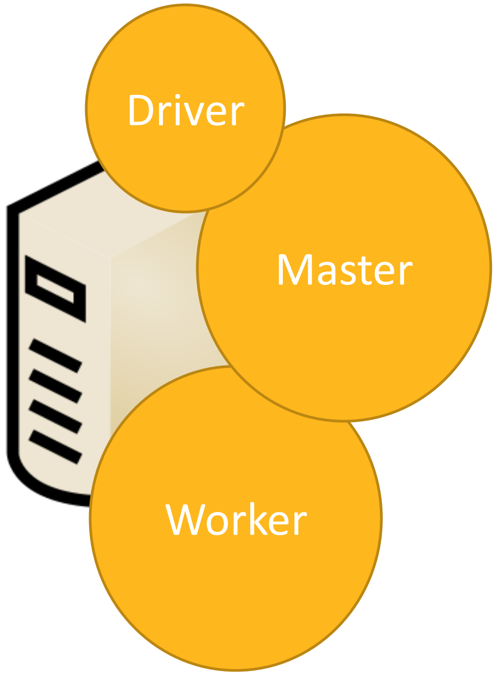
Spark Overview



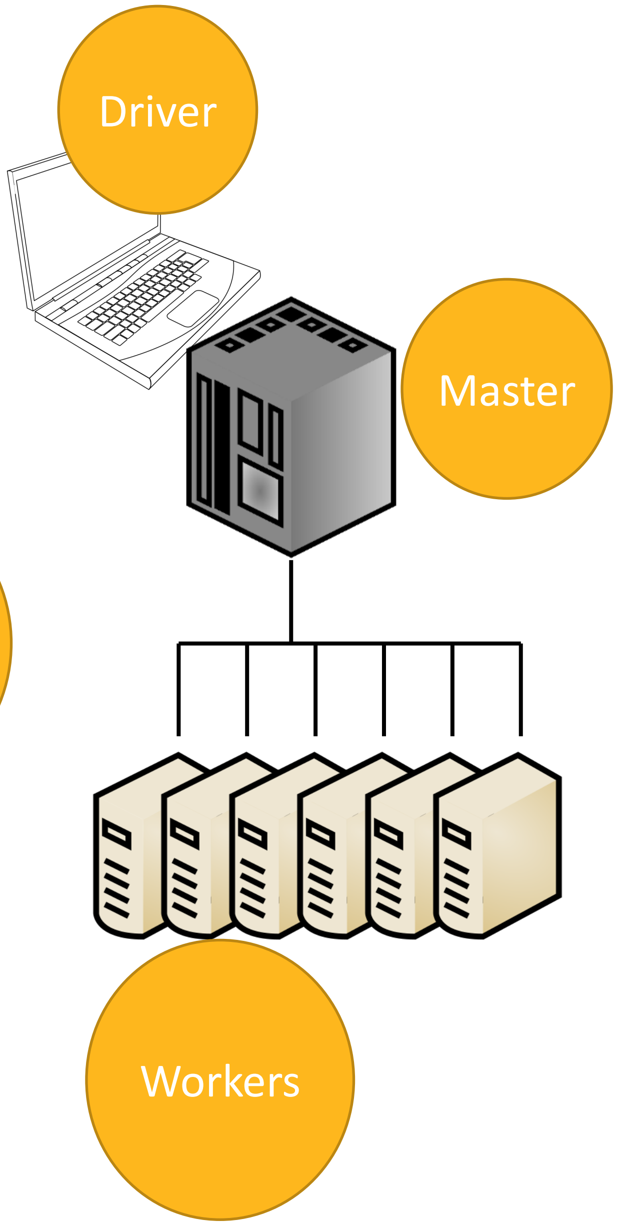
Spark Operation Modes



Local mode



Stand-alone mode



Cluster mode

Examples

CELEBRATING 30 YEARS
Marlan and Rosemary Bourns
College of Engineering

Examples

host	time	method	url	response	bytes
pppa006.compuserve.com	807256800	GET	/images/launch-logo.gif	200	1713
vcc7.langara.bc.ca	807256804	GET	/shuttle/missions/missions.html	200	8677

```
# Initialize the Spark context
JavaSparkContext spark =
    new JavaSparkContext("local", "CS226-Demo");
```

Examples

```
// Initialize the Spark context
JavaSparkContext spark =
    new JavaSparkContext("local", "CS226-Demo");


# Hello World! Example. Count the number of lines in the file
JavaRDD<String> textFileRDD =
                  spark.textFile("nasa.tsv");
long count = textFileRDD.count();
System.out.println("Number of lines is "+count);
```

Examples

```
// Count the number of OK lines
JavaRDD<String> okLines = textFileRDD.filter(new
Function<String, Boolean>() {
    @Override
    public Boolean call(String s) throws Exception {
        String code = s.split("\t")[5];
        return code.equals("200");
    }
});
long count = okLines.count();
System.out.println("Number of OK lines is "+count);
```

Examples

```
// Count the number of OK lines
// Shorten the implementation using lambdas (Java 8 and above)
JavaRDD<String> okLines =
 textFileRDD.filter(s -> s.split("\t")[5].equals("200"));


long count = okLines.count();
System.out.println("Number of OK lines is "+count);
```

Examples

```
// Make it parametrized by taking the response code as a
command line argument
String inputFileName = args[0];
String desiredResponseCode = args[1];
...
JavaRDD<String> textFileRDD = spark.textFile(inputFileName);
JavaRDD<String> okLines = textFileRDD.filter(
    s -> s.split("\t")[5].equals(desiredResponseCode));
```

Examples

```
// Count by response code
// Important! Not all operations are on the getting started
guide
JavaPairRDD<Integer, String> linesByCode =
textFileRDD.mapToPair(s -> {
        String code = s.split("\t")[5];
        return new Tuple2<>(Integer.valueOf(code), s);
    });
Map<Integer, Long> countByCode = linesByCode.countByKey();
System.out.println(countByCode);
```

Further Reading

- Spark home page: http://spark.apache.org/
- Quick start: http://spark.apache.org/docs/latest/quick-start.html
- RDD documentation: http://spark.apache.org/docs/latest/rdd-programming-guide.html
- RDD Paper: Matei Zaharia *et al*. "Resilient Distributed Datasets: A Fault-tolerant Abstraction for In-memory Cluster Computing." NSDI’12