

# Data Engineering with Apache Spark

---

Markus Dale, medale@asymmetrik.com

May 2019

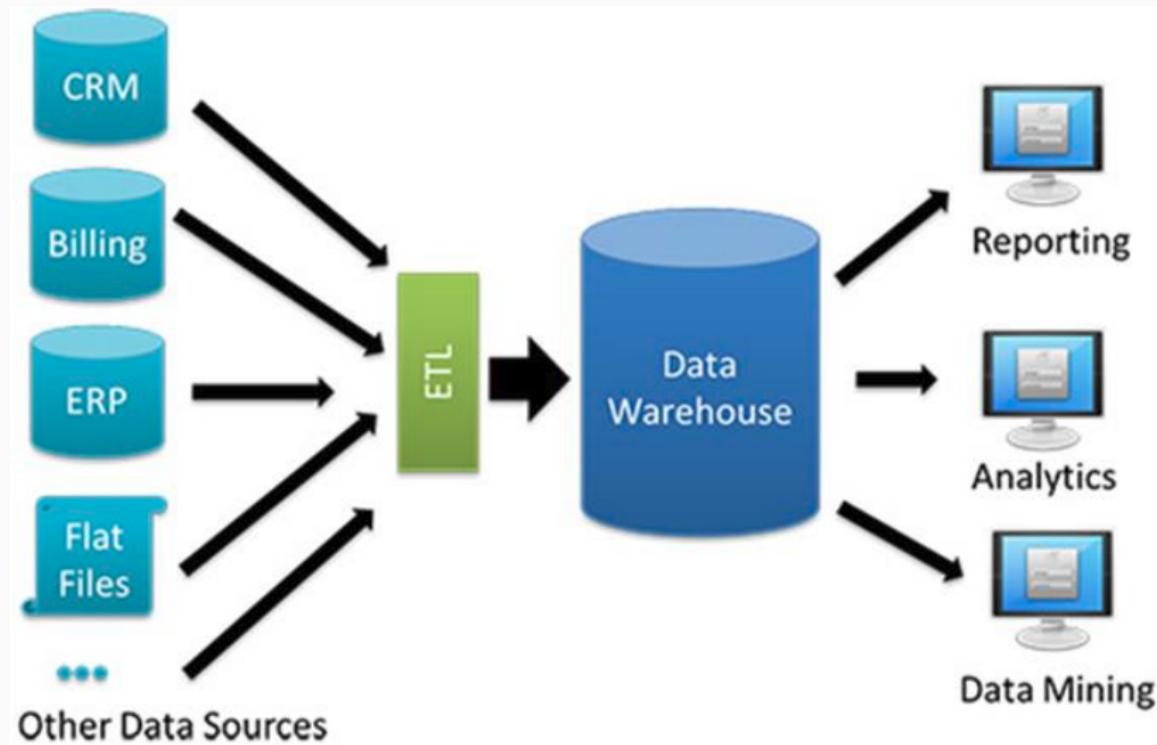
## Intro, Slides And Code

- Slides: <https://github.com/medale/prez-spark-dataengineering/blob/master/presentation/SparkDataEngineering.pdf>
- Scala Spark Code Examples:  
<https://github.com/medale/prez-spark-dataengineering>

# Goals

- Intro to Spark Scala Dataset API for data engineering
  - At scale data exploration
  - At scale ETL (Extract Transform Load)
  - Storage formats
  - Spark abstractions

# Data engineering



# Apache Spark - Big data tooling



Figure 1: Swiss Army Knife for Big Data

# Apache Spark: Data engineering on small dataset



Figure 2: Laptop

## Apache Spark: Data engineering for larger dataset (Vertical Scaling)

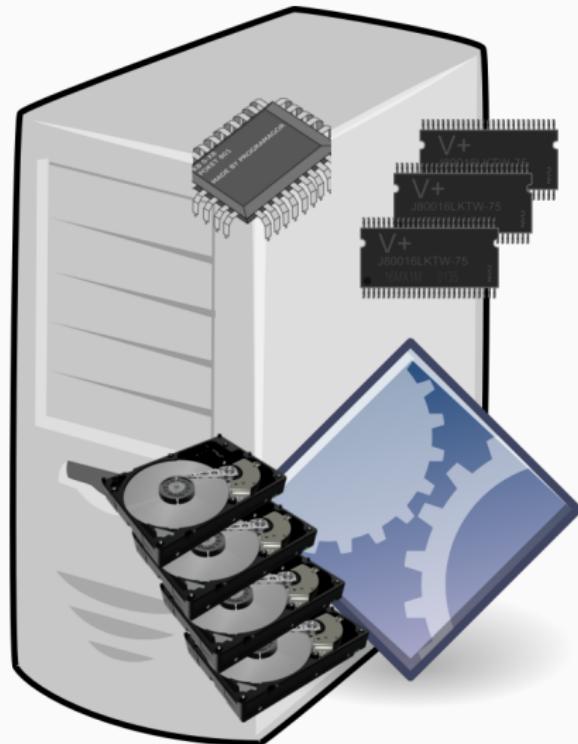


Figure 3: Beefed-up Server

# Apache Spark: Data engineering for large datasets (Horizontal Scaling)

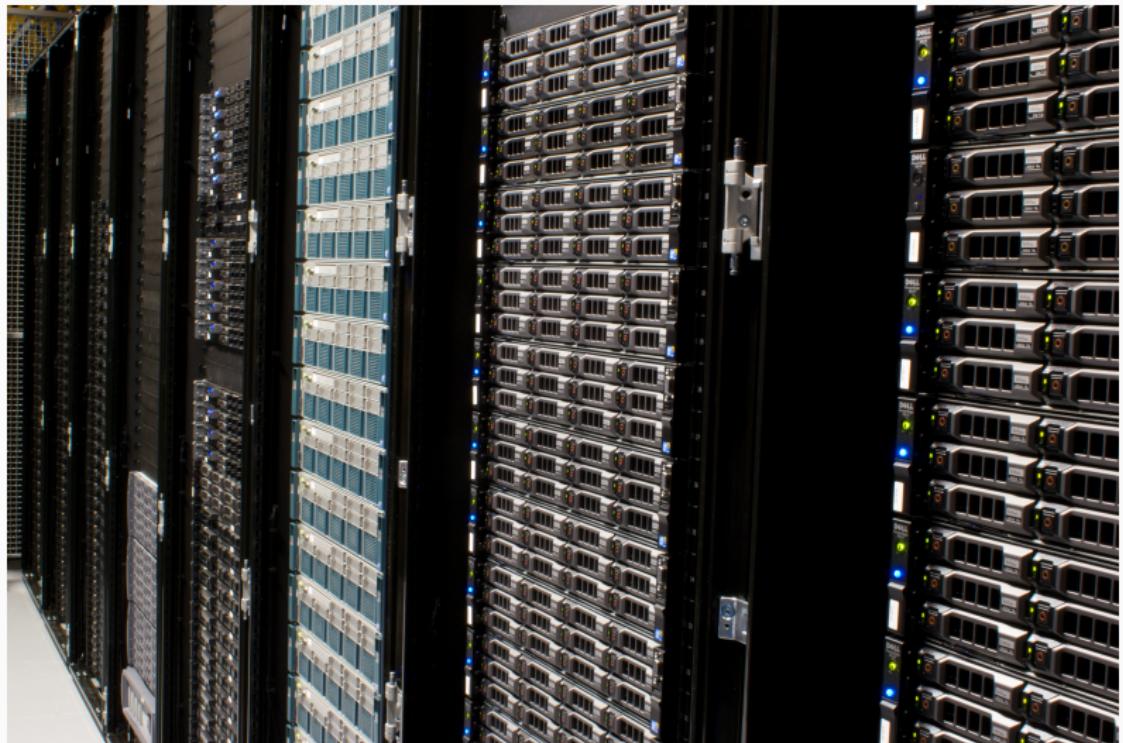


Figure 4: Multiple cooperating Servers

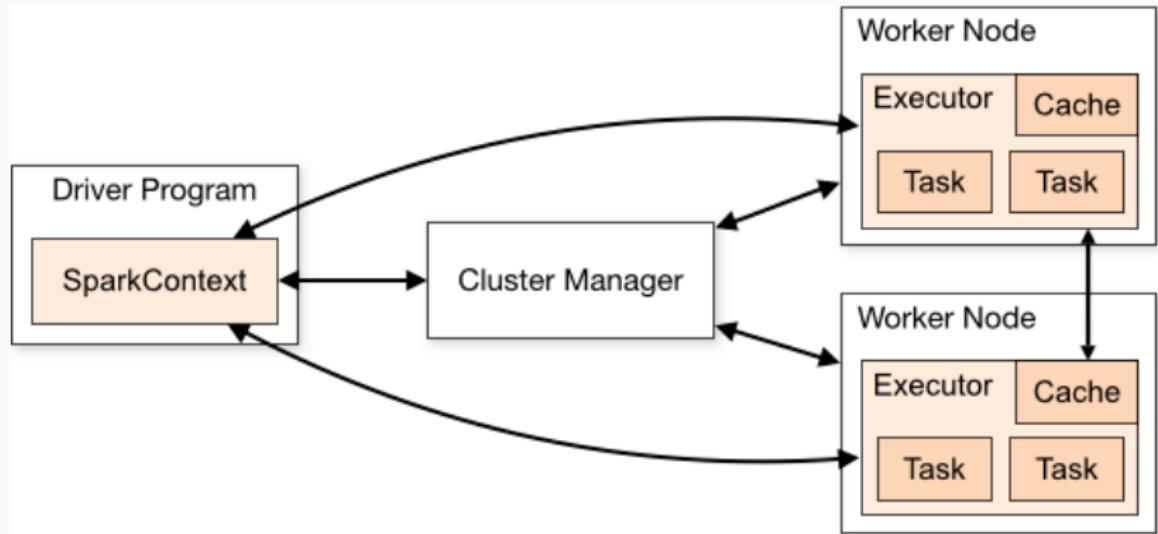
Cluster Manager - Manage cores, memory, special capabilities



Amazon EMR



# Anatomy of a Spark Application



Source: Apache Spark website

# Hello, Spark World!

```
import org.apache.spark.sql.SparkSession

object HelloSparkWorld {

    val RecordsUrl = "file:///datasets/github/data"

    def process(spark: SparkSession): (Long, Long) = {
        val records = spark.read.json(RecordsUrl)
        val totalEventCount = records.count()
        val prs = records.where(records("type") === "PullRequestEvent")
        val pullRequestEventCount = prs.count()
        (totalEventCount, pullRequestEventCount)
    }

    def main(args: Array[String]): Unit = {
        val spark = SparkSession.builder().appName("HelloSparkWorld").getOrCreate()
        val (total, prs) = process(spark)
        println(s"Total events: ${total}, pr events: ${prs}.")
        spark.stop()
    }
}
```

## SparkSession - Gateway to the Cluster

```
def main(args: Array[String]): Unit = {  
    val spark = SparkSession.builder().  
        appName("HelloSparkWorld").  
        getOrCreate()  
    ...  
    spark.stop()  
}
```

# API - SparkSession

SparkSession

org.apache.spark.sql

## SparkSession

Related Docs: [class Sp](#)

object **SparkSession** extends [Logging](#) with [Serializable](#)

Annotations @Stable()  
Source [SparkSession.scala](#)

► Linear Supertypes

Ordering [Alphabetic](#) By Inheritance

Inherited [SparkSession](#) [Serializable](#) [Serializable](#) [Logging](#) [AnyRef](#) [Any](#)  
[Hide All](#) [Show All](#)

Visibility [Public](#) All

### Type Members

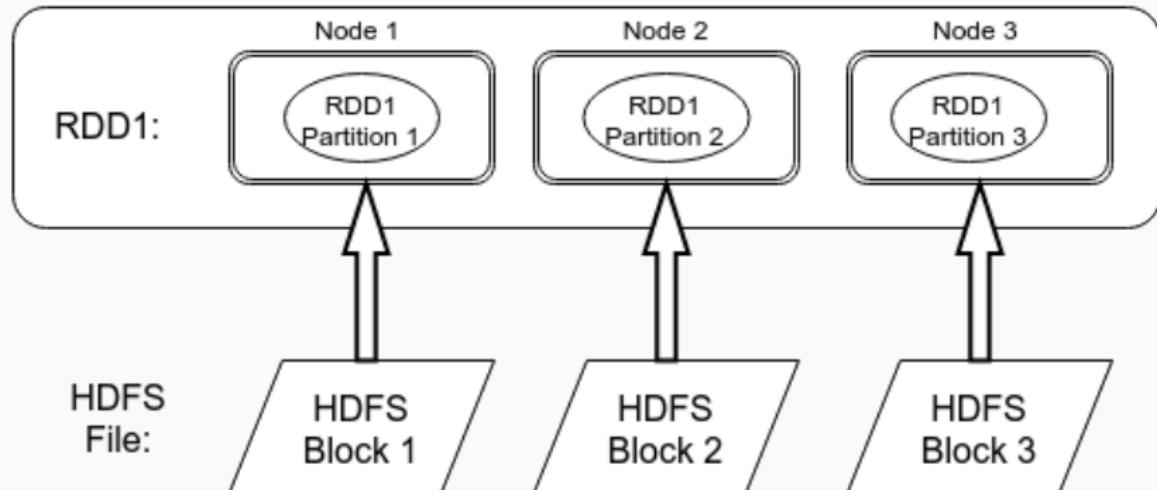
► class **Builder** extends [Logging](#)  
Builder for [SparkSession](#).

### Value Members

► def **active**: [SparkSession](#)  
Returns the currently active SparkSession, otherwise the default one.

► def **builder()**: [Builder](#)  
Creates a [SparkSession.Builder](#) for constructing a [SparkSession](#).

## Resilient Distributed Datasets (RDDs)



## Starting Spark Standalone Cluster Manager

```
# Start on master
$SPARK_HOME/sbin/start-master.sh --host 192.168.1.232

# Start one or more workers
$SPARK_HOME/sbin/start-slave.sh spark://192.168.1.232:7077
```

# Spark Standalone Cluster Manager UI - idle



## Spark Master at spark://192.168.1.230:7077

URL: spark://192.168.1.230:7077

Alive Workers: 1

Cores in use: 8 Total, 0 Used

Memory in use: 30.4 GB Total, 0.0 B Used

Applications: 0 Running, 0 Completed

Drivers: 0 Running, 0 Completed

Status: ALIVE

### Workers (1)

Worker Id	Address	State	Cores	Memory
worker-20190430220608-192.168.1.230-37667	192.168.1.230:37667	ALIVE	8 (0 Used)	30.4 GB (0.0 B Used)

### Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration

### Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration

## Running spark-shell in cluster

```
spark-shell --master spark://192.168.1.232:7077 \
--driver-memory 1g \
--executor-memory 2g \
--total-executor-cores 4 \
--executor-cores 2 \
--jars /tmp/dataset-0.9.0-SNAPSHOT-fat.jar
```

## Spark Shell Startup

# Spark Standalone Cluster Manager - 1 running application



## Spark Master at spark://192.168.1.230:7077

URL: spark://192.168.1.230:7077

Alive Workers: 1

Cores in use: 8 Total, 4 Used

Memory in use: 30.4 GB Total, 4.0 GB Used

Applications: 1 [Running](#), 0 [Completed](#)

Drivers: 0 Running, 0 Completed

Status: ALIVE

### ▼ Workers (1)

Worker Id	Address	State	Cores	Memory
worker-20190430220608-192.168.1.230-37667	192.168.1.230:37667	ALIVE	8 (4 Used)	30.4 GB (4.0 GB Used)

### ▼ Running Applications (1)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
app-20190430221543-0000	(kill) <a href="#">Spark shell</a>	4	2.0 GB	2019/04/30 22:15:43	medale	RUNNING	14 min

### ▼ Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration

## HelloSparkWorld in spark-shell

```
scala> import com.uebercomputing.HelloSparkWorld  
import com.uebercomputing.HelloSparkWorld  
  
scala> HelloSparkWorld.process(spark)  
res0: (Long, Long) = (147374,6699)
```

# Spark Application UI - Jobs, stages, tasks



Spark shell application UI

Jobs    Stages    Storage    Environment    Executors    SQL

## Spark Jobs (?)

User: medale  
Total Uptime: 7.5 min  
Scheduling Mode: FIFO  
Completed Jobs: 3

▶ Event Timeline

▼ Completed Jobs (3)

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
2	count at HelloSparkWorld.scala:16 count at HelloSparkWorld.scala:16	2019/04/30 22:54:02	0.3 s	2/2	4/4
1	count at HelloSparkWorld.scala:13 count at HelloSparkWorld.scala:13	2019/04/30 22:53:57	4 s	2/2	4/4
0	json at HelloSparkWorld.scala:11 json at HelloSparkWorld.scala:11	2019/04/30 22:53:53	3 s	1/1	3/3

## Job - n lazy transformations, 1 action

```
//job 0 - list files, infer schema
val records = spark.read.json("file:///datasets/github/data")
//transformation
records.cache()
//action - job 1
val totalEventCount = records.count()

//transformation - datasets are immutable!
val prs = records.where(records("type") === "PullRequestEvent")
//action - job 2
val pullRequestEventCount = prs.count()
```

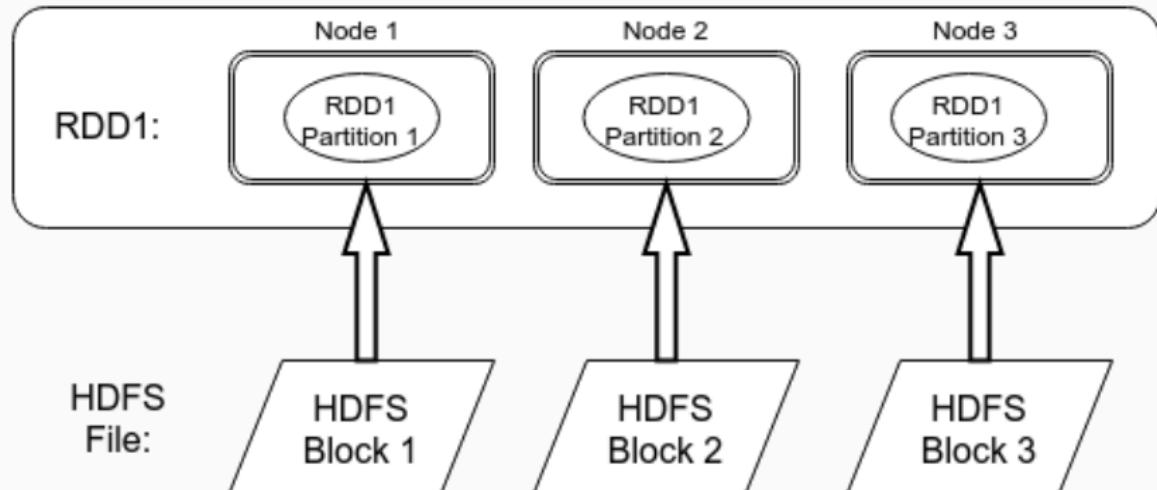
## Job 0 - Stages, tasks, partitions

Job Id ▼	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
1	count at HelloSparkWorld.scala:13 count at HelloSparkWorld.scala:13	2019/04/30 22:53:57	4 s	2/2	4/4
0	json at HelloSparkWorld.scala:11 json at HelloSparkWorld.scala:11	2019/04/30 22:53:53	3 s	1/1	3/3

## Job 0 - Stage 0, tasks, partitions

Stage Id ▾	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
0	<a href="#">Json at HelloSparkWorld.scala:11 +details</a>	2019/04/30 22:53:53	3 s	3/3	46.6 MB			

## Input partitions - splittable file?



## Job 1 - count

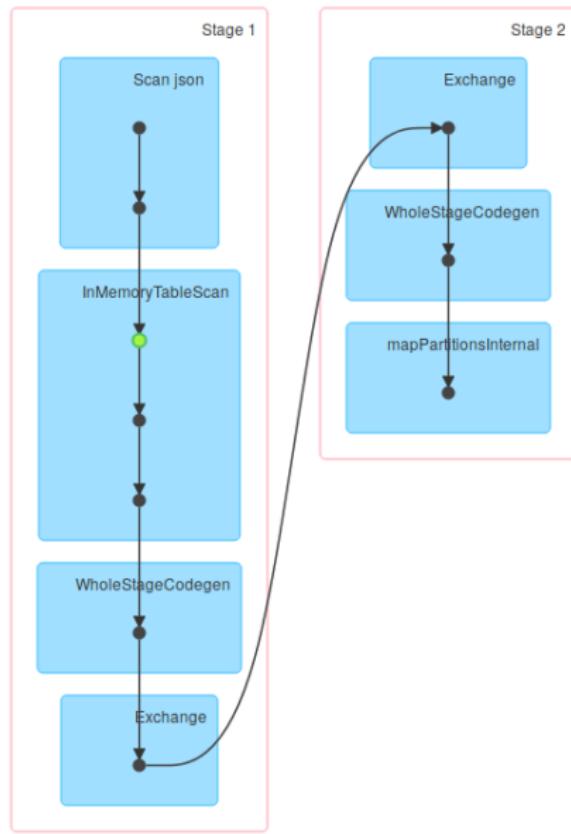
Job Id ▼	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
1	count at HelloSparkWorld.scala:13 <a href="#">count at HelloSparkWorld.scala:13</a>	2019/04/30 22:53:57	4 s	2/2	4/4
0	json at HelloSparkWorld.scala:11 <a href="#">json at HelloSparkWorld.scala:11</a>	2019/04/30 22:53:53	3 s	1/1	3/3

## Job 1 - count Stages 1 and 2

Stage Id ▾	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
2	count at <a href="#">HelloSparkWorld.scala:13</a> +details	2019/04/30 22:54:02	69 ms	1/1			177.0 B	
1	count at <a href="#">HelloSparkWorld.scala:13</a> +details	2019/04/30 22:53:57	4 s	3/3	46.6 MB			177.0 B

# Job 1 - Stages 1 and 2 DAG

## DAG Visualization



## RDDs - Not deprecated!

```
object RddProcessor {  
  
    val DefaultEventInputUrl = "file:///datasets/github/data"  
  
    def process(sc: SparkContext, inputUrl: String): (Long, Long) = {  
        val records: RDD[String] = sc.textFile(inputUrl)  
        println(s"We have a total of ${records.partitions.size} partitions.")  
        val total = records.count()  
        val prs = records.filter(r => r.contains("PullRequestEvent"))  
        val totalPrs = prs.count()  
        (total, totalPrs)  
    }  
  
    def main(args: Array[String]): Unit = {  
        val spark = SparkSession.builder().  
            appName(name = "RddProcessor").  
            getOrCreate()  
  
        val inputUrl = if (args.size > 0) {  
            args(0)  
        } else {  
            DefaultEventInputUrl  
        }  
        process(spark.sparkContext, inputUrl)  
        spark.stop()  
    }  
}
```

## And now for something completely different: Colon Cancer



- Screening saves lives!
  - Colonoscopy - talk to your doc
  - Dave Barry: A journey into my colon — and yours
- Colorectal Cancer Alliance

# Questions?



- [medale@asymmetrik.com](mailto:medale@asymmetrik.com)
- <https://github.com/medale/prez-spark-dataengineering/blob/master/presentation/SparkDataEngineering.pdf>