

# 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>
- Spark Data Engineering Code Examples:  
<https://github.com/medale/prez-spark-dataengineering>

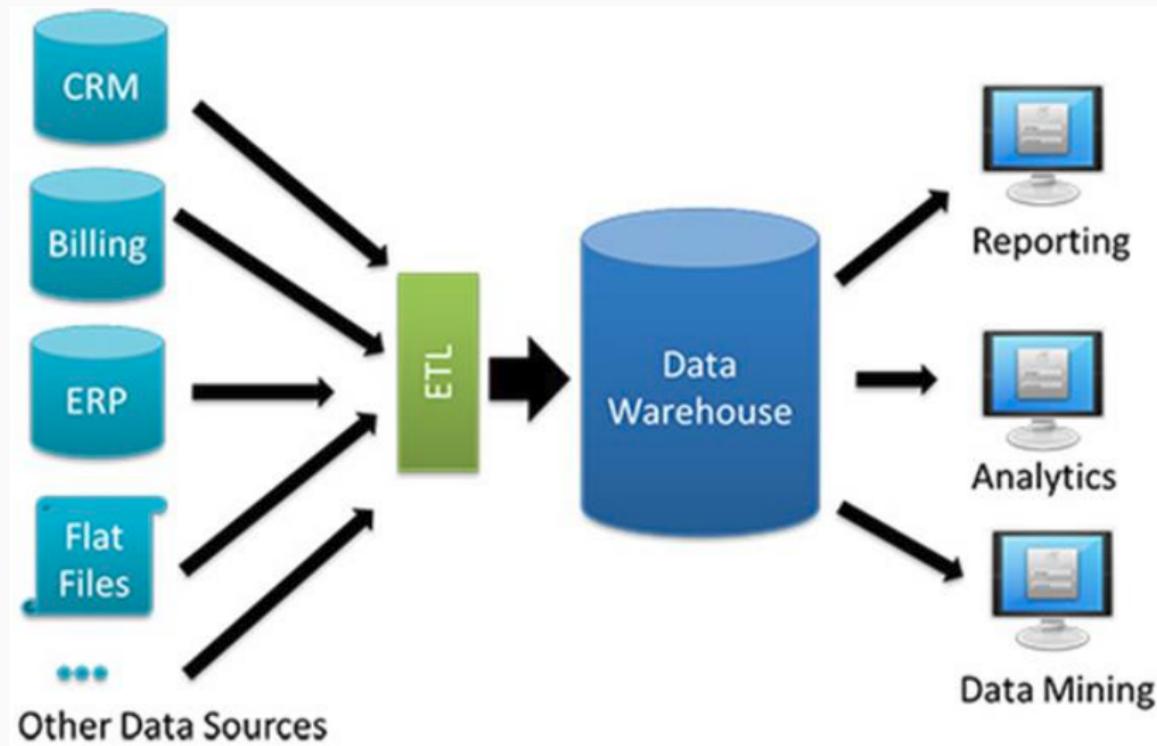
## Data Science Mission

- ID malicious GitHub Pull Requests?
- Source: <https://www.gharchive.org/>

# Data Engineering Mission

- <https://www.gharchive.org/>
  - 2/12/2011-12/31/2014 Timeline API (now deprecated).
  - From 1/1/2015 to now Events API.
  - Since 2015: About 20-40MB/hour \* 37900 hours ~1TB
- Store: Fast, time-based access when specifying yyyy, mm, dd, hh (or prefix combination)
- Side effect: Learn more about Spark batch processing

# 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)

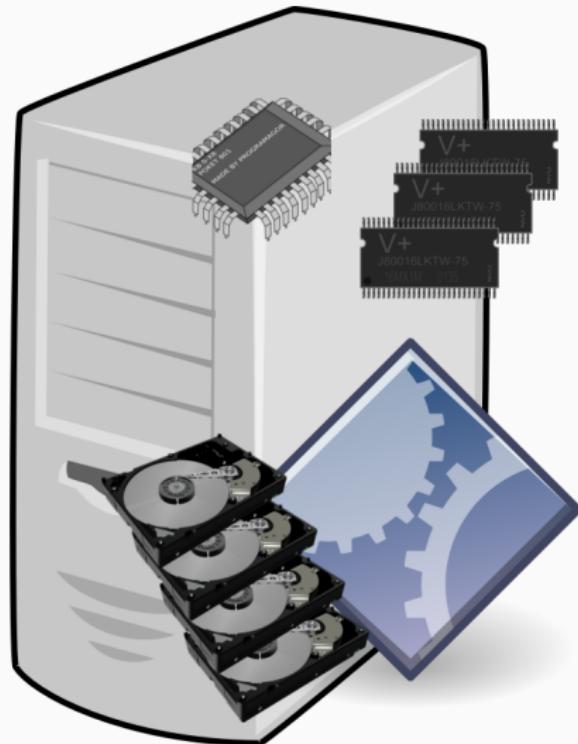


Figure 3: Beefed-up Server

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

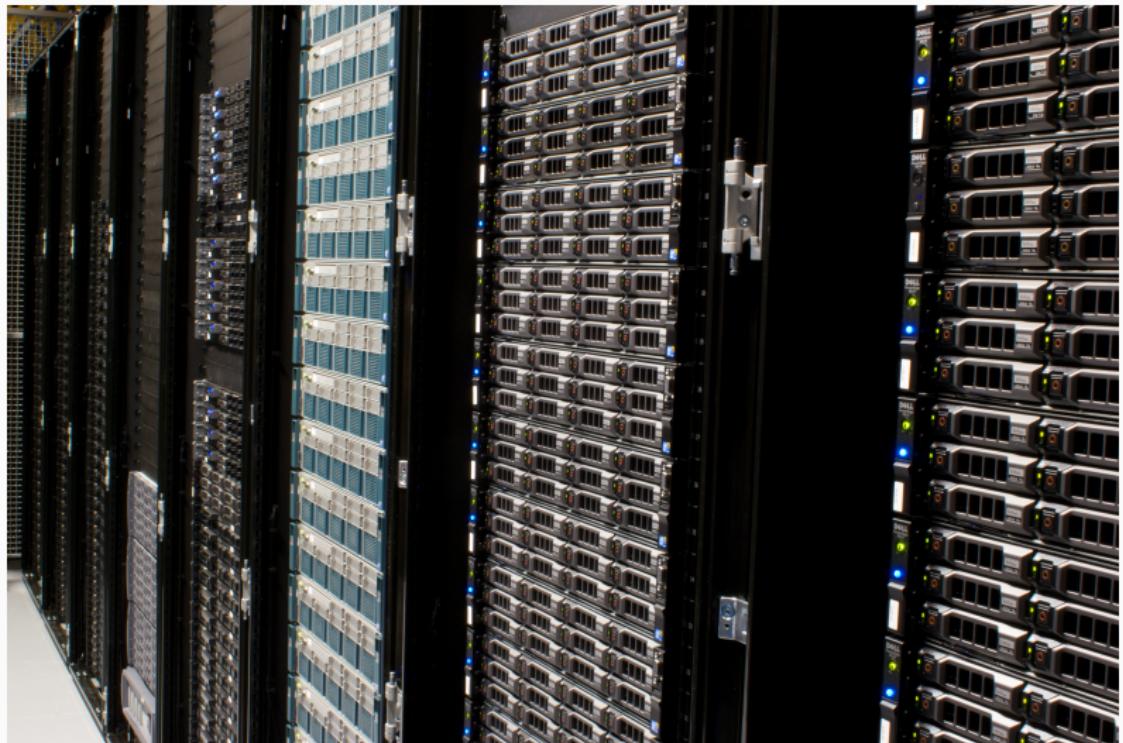


Figure 4: Multiple cooperating Servers

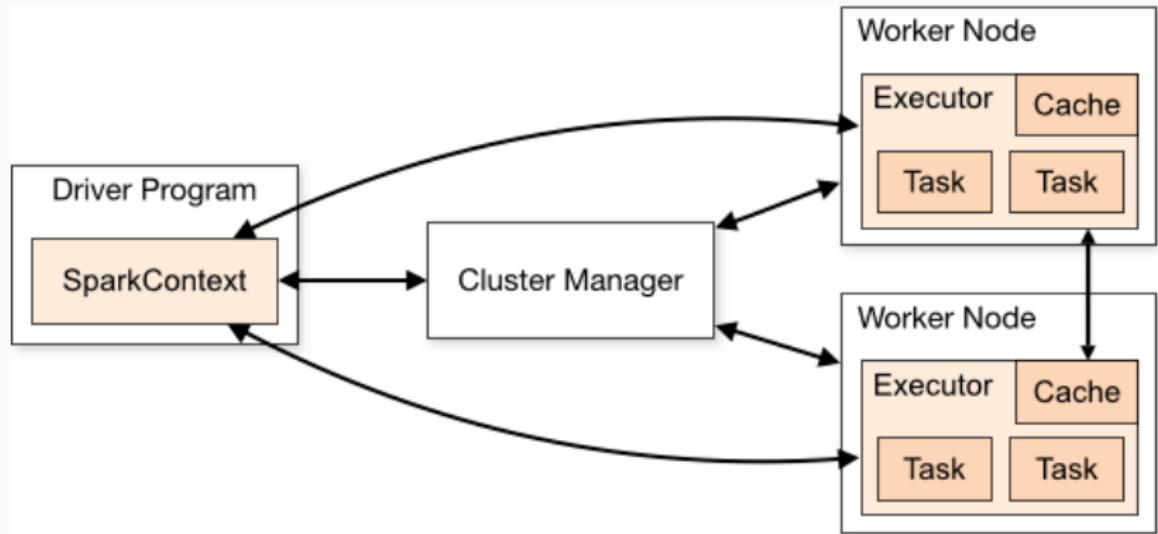
Cluster Manager - Manage cores, memory, special capabilities



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 Object

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](#).

## API - SparkSession Class

```
//spark.read - DataFrameReader  
val records: DataFrame = spark.read.json(RecordsUrl)
```

## spark.read: DataFrameReader - Input

- CSV
- json
- jdbc
- parquet
- text - DataFrame with “value” column
- textFile - Dataset[String]
- Third party:
  - <https://spark-packages.org>: Avro, Redshift, MongoDB...
  - Spark Cassandra Connector (DataStax github)

DataFrame = Dataset[Row]

```
val records: DataFrame = spark.read.json(RecordsUrl)
```

## DataFrame Schema

```
scala> records.printSchema
root
|-- actor: struct (nullable = true)
|   |-- display_login: string (nullable = true)
|   |-- id: long (nullable = true)
...
|-- created_at: string (nullable = true)
|-- id: string (nullable = true)
|-- payload: struct (nullable = true)
|   |-- comment: struct (nullable = true)
|   |   |-- body: string (nullable = true)
...
|-- public: boolean (nullable = true)
|-- repo: struct (nullable = true)
|   |-- id: long (nullable = true)
|   |-- url: string (nullable = true)
|-- type: string (nullable = true)
```

## GitHub Event Data

```
cd /datasets/github/data
wget http://data.gharchive.org/2019-04-28-0.json.gz
wget http://data.gharchive.org/2019-04-28-1.json.gz
wget http://data.gharchive.org/2019-04-28-13.json.gz
```

## Preliminary Exploration

```
# creates 2019-04-28-0.json  
gunzip 2019-04-28-0.json.gz
```

```
# 95865  
wc -l 2019-04-28-0.json
```

## Editor: one JSON per line

```
[{"id": "9493263421", "type": "PushEvent", "actor": {"id": 25541826, "login": "mainh", "display_login": "mainh", "gravatar_id": "", "url": "https://api.github.com/users/mainh", "avatar_url": "https://avatars.githubusercontent.com/u/25541826?"}, "repo": {"id": 150551511, "name": "Car-eye-team/JTT1078-media-server", "url": "https://api.github.com/repos/Car-eye-team/JTT1078-media-server"}, "payload": {"push_id": 3536163318, "size": 1, "distinct_size": 1, "ref": "refs/heads/master", "head": "fa8714446ae5c83c16f3125c33d51388aa6e797c", "before": "6edb3514ce440ed16d8e4cc99b8f829211f4e8a8", "commits": [{"sha": "fa8714446ae5c83c16f3125c33d51388aa6e797c", "author": {"name": "Main", "email": "a3a6910872d9e29b8a7d713ee6dd76d73f01233c@qq.com"}, "message": "Add files via upload\n\n更新架构图", "distinct": true, "url": "https://api.github.com/repos/Car-eye-team/JT1078-media-server/commits/fa8714446ae5c83c16f3125c33d51388aa6e797c"}]}, "public": true, "created_at": "2019-04-23T12:00:00Z", "org": {"id": 36096525, "login": "Car-eye-team", "gravatar_id": "", "url": "https://api.github.com/orgs/Car-eye-team", "avatar_url": "https://avatars.githubusercontent.com/u/36096525?"}}]  
{"id": "9493263424", "type": "WatchEvent", "actor": {"id": 6560033, "login": "voby", "display_login": "voby", "gravatar_id": "", "url": "https://api.github.com/users/voby", "avatar_url": "https://avatars.githubusercontent.com/u/6560033?"}, "repo": {"id": 74293321, "name": "sveltejs/svelte", "url": "https://api.github.com/repos/sveltejs/svelte"}, "payload": {"action": "started"}, "public": true, "created_at": "2019-04-23T12:00:00Z", "org": {"id": 23617963, "login": "sveltejs", "gravatar_id": "", "url": "https://api.github.com/orgs/sveltejs", "avatar_url": "https://avatars.githubusercontent.com/u/23617963?"}}}  
{"id": "9493263426", "type": "PullRequestEvent", "actor": {"id": 300805, "login": "D4nte", "display_login": "D4nte", "gravatar_id": "", "url": "https://api.github.com/users/D4nte", "avatar_url": "https://avatars.githubusercontent.com/u/300805?"}, "repo": {"id": 145489725, "name": "coblox/bitcoinrpc-rust-client", "url": "https://api.github.com/repos/coblox/bitcoinrpc-rust-client"}, "payload": {"action": "opened", "number": 54, "pull_request": {"url": "https://api.github.com/repos/coblox/bitcoinrpc-rust-client/pull/54?"}}}
```

## Pretty Print One Record?

```
# default 1000 lines - xaa, xab...
split 2019-04-28-0.json
mkdir temp
cd temp

# 1 file per line
split -1 .../xaa

# xac's a PullRequestEvent
python -m json.tool < xac > pretty.json
```

## Open pretty.json in Atom - PullRequestEvent

```
        "organizations_url": "https://api.github.com/users/D4nte/orgs",
        "received_events_url": "https://api.github.com/users/D4nte/received_events",
        "repos_url": "https://api.github.com/users/D4nte/repos",
        "site_admin": false,
        "starred_url": "https://api.github.com/users/D4nte/starred{/owner}",
        "subscriptions_url": "https://api.github.com/users/D4nte/subscriptions",
        "type": "User",
        "url": "https://api.github.com/users/D4nte"
    }
},
"public": true,
"repo": {
    "id": 145489725,
    "name": "cloblox/bitcoinrpc-rust-client",
    "url": "https://api.github.com/repos/cloblox/bitcoinrpc-rust-client"
},
"type": "PullRequestEvent"
}
```

## 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

## spark-shell auto-imports

```
scala> :imports
1) import org.apache.spark.SparkContext._
2) import spark.implicits._
3) import spark.sql
4) import org.apache.spark.sql.functions._
```

## Data Exploration - schema and counting

```
val RecordsUrl = "file:///datasets/github/data"
val records = spark.read.json(RecordsUrl)
records.printSchema
//...
// |-- repo: struct (nullable = true)
// |   |-- id: long (nullable = true)
// |   |-- name: string (nullable = true)
// |   |-- url: string (nullable = true)
// |   |-- type: string (nullable = true)

records.count()
//147,374
```

# Spark Application UI - Jobs, stages, tasks



Spark shell application UI

Jobs    Stages    Storage    Environment    Executors    SQL

## Spark Jobs (?)

User: medale  
Total Uptime: 2.5 min  
Scheduling Mode: FIFO  
Completed Jobs: 2

▶ Event Timeline

▼ Completed Jobs (2)

Job Id ▼	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
1	count at <console>:26 count at <console>:26	2019/05/02 10:01:16	0.6 s	2/2	4/4
0	json at <console>:25 json at <console>:25	2019/05/02 10:00:58	4 s	1/1	3/3

# Spark Application UI - Stages



Spark shell application UI

Jobs    Stages    Storage    Environment    Executors    SQL

## Stages for All Jobs

Completed Stages: 3

▼ Completed Stages (3)

Stage Id ▾	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
2	count at <console>:26 +details	2019/05/02 10:01:16	76 ms	1/1			177.0 B	
1	count at <console>:26 +details	2019/05/02 10:01:16	0.5 s	3/3	46.6 MB			177.0 B
0	json at <console>:25 +details	2019/05/02 10:00:58	4 s	3/3	46.6 MB			

# Spark Application UI - Stage details

Jobs Stages Storage Environment Executors SQL

## Details for Stage 1 (Attempt 0)

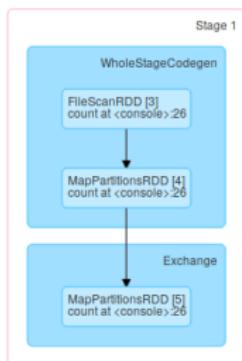
Total Time Across All Tasks: 1 s

Locality Level Summary: Process local: 3

Input Size / Records: 46.6 MB / 147374

Shuffle Write: 177.0 B / 3

▼ DAG Visualization



► Show Additional Metrics

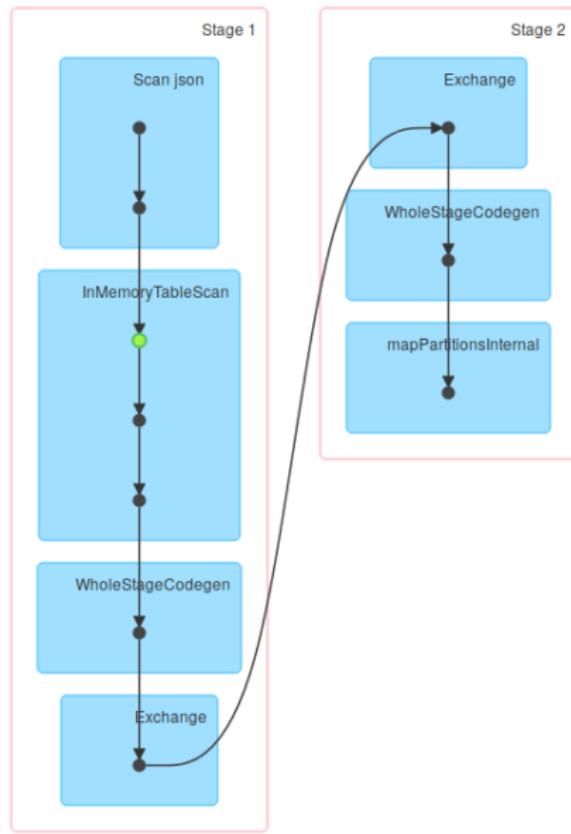
► Event Timeline

### Summary Metrics for 3 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
--------	-----	-----------------	--------	-----------------	-----

# Job 1 - Stages 1 and 2 DAG

## DAG Visualization



## One Job = n lazy transformations, 1 action

```
//job0 (read twice - once for schema, once for count)
val records = spark.read.json(RecordsUrl)
//lazy transformation
records.cache()
//action 1 - end of job 1
records.count()
records.printSchema

//2 lazy transformations - immutable datasets
val types = records.select("type")
val distinctTypes = types.distinct()

//one eager action - job 2
distinctTypes.show()
+-----+
|      type |
+-----+
|      PushEvent |
...
|      PullRequestEvent |
+-----+
```

# Spark Application UI - Storage (caching)

APACHE  2.4.0

Jobs Stages Storage Environment Executors SQL

Spark shell application UI

## Storage

### ▼ RDDs

ID	RDD Name	Storage Level	Cached Partitions	Fraction Cached	Size in Memory	Size on Disk
11	FileScan json [actor#6,created_at#7,id#8,org#9,payload#10,public#11,repo#12,type#13] Batched: false, Format: JSON, Location: InMemoryFileIndex[file:/datasets/github/data], PartitionFilters: [], PushedFilters: [], ReadSchema: struct<actor:struct<avatar_url:string,display_login:string,gravatar_id:string,id:bigint,login:str...]	Memory Deserialized 1x Replicated	3	100%	312.1 MB	0.0 B

## Default file system/file system URLs

```
val hc = spark.sparkContext.hadoopConfiguration  
hc.get("fs.defaultFS")  
//res4: String = file:///
```

```
spark.read.json("/datasets/github/data")  
spark.read.json("file:///datasets/github/data")
```

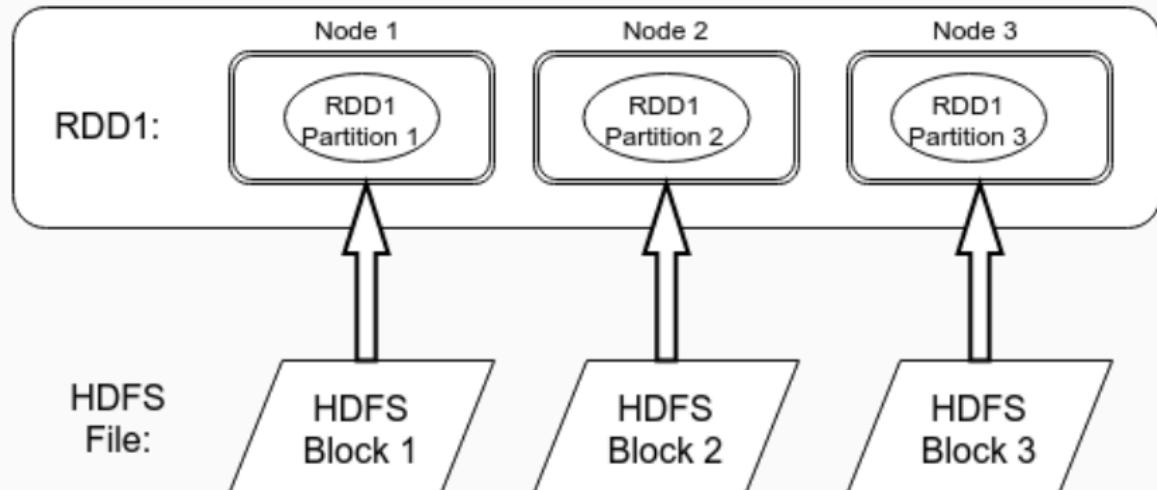
```
//hdfs://<name-node>:port  
//s3a://<bucket-name>
```

## Input partitions - splittable file?

- Splittable: bzip2, parquet, avro
- Non-splittable: gzip (1 task per file)

```
hc.get("dfs.block.size")
//res6: String = 134217728
```

## Resilient Distributed Datasets (RDDs)



## 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()  
    }  
}
```

## Datasets/DataFrames compiled to RDDs

- Catalyst query optimizer for built-in functions
- Project Tungsten - memory management
  - Row storage (Apache Arrow)
  - Encoders for Dataset objects (`spark.implicits._`)

## Data Exploration - event type distribution

```
val prs = records.where(records("type") === "PullRequestEvent")
val pullRequestEventCount = prs.count()

import spark.implicits._
val typeCounts = records.groupBy($"type").
    count.orderBy($"count".desc)
typeCounts.show
+-----+-----+
|      type | count |
+-----+-----+
| PushEvent | 82518 |
| CreateEvent | 18313 |
| WatchEvent | 16868 |
| PullRequestEvent | 6699 |
...
...
```

## Narrow vs. wide transformations

- narrow: all data from one partition
- wide: data from multiple parent partitions (shuffle)

## Shuffle Partitions - Default

Job Id ▼	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
4	show at <console>:26 show at <console>:26	2019/05/02 10:54:53	0.8 s	2/2	203/203

## Setting Shuffle Partitions

```
spark.conf.set("spark.sql.shuffle.partitions",  
    "10")
```

## Shuffle Partitions Optimized

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
5	<a href="#">show at &lt;console&gt;:26</a> <a href="#">show at &lt;console&gt;:26</a>	2019/05/02 11:04:14	0.1 s	2/2	13/13

## 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>