Welcome!

WWCode San Francisco - Backend Study Group

Aug 3, 2023

- We'll start in a moment :)
- We are **RECORDING** tonight's event
- We may plan to take screenshots for social media
- If you are comfortable, turn the video ON. If you want to be anonymous, then turn the video off
- We'll introduce the hosts & make some time for Q&A at the end of the presentation
- Feel free to take notes
- Online event best practices:
    - Don't multitask. Distractions reduce your ability to remember concepts
    - Mute yourself when you aren't talking
    - We want the session to be interactive
    - Use the 'Raise Hand' feature to ask questions
- **By attending our events, you agree to comply with our Code of Conduct**

WOMEN WHO CODE /san-francisco

Introduction & Agenda

• Welcome from WWCode!
• Our mission: Empower diverse women to excel in technology careers
• Our vision: A tech industry where diverse women and historically excluded people thrive at any level
• About Backend Study Group

Harini Rajendran
Presentor
Senior Software Engineer, Confluent
Lead, WWCode SF

Ashwini Vasanth
Host
Product Builder
Architecture & API Platform, Devrev

- **Apache Spark 101**
  - **Need for Distributed Computing**
  - **Hadoop MapReduce**
  - **Disk based vs Memory based systems**
  - **What is Apache Spark?**
  - **Why Spark?**
  - **Architecture**
  - **Spark use cases**
  - **Spark APIs**
  - **Batch processing pipeline with Spark**
  - **Simple Demo**
  - **Q & A**

WOMEN WHO CODE /san-francisco

Need for Distributed Computing

- **Find the maximum number from [1,9,25,93,84,225,85,24,82,300]**
- **Count the number of occurrences of each word in a file with 100 words**

Need for Distributed Computing

- **Find the maximum number from a file that has trillions of numbers**
- **Count the number of occurrences of each word in a file of size worth multiple terrabytes**

Need for Distributed Computing

- **Problem**

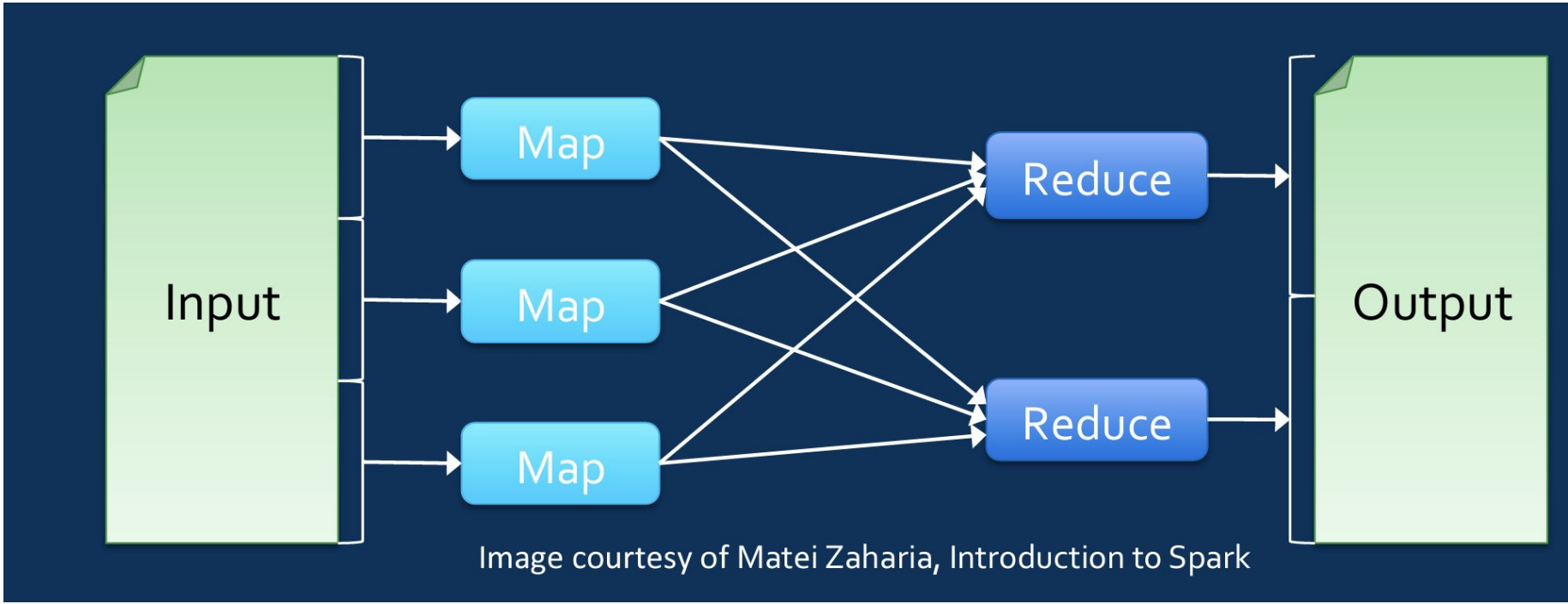
  Single machine is not enough anymore to complete the computation at hand.

  Example: Computing max of trillions of numbers

- **Solution**

  Split the dataset into multiple chunks, parallelize the tasks and assign them to a network of machines

Hadoop MapReduce



Hadoop MapReduce

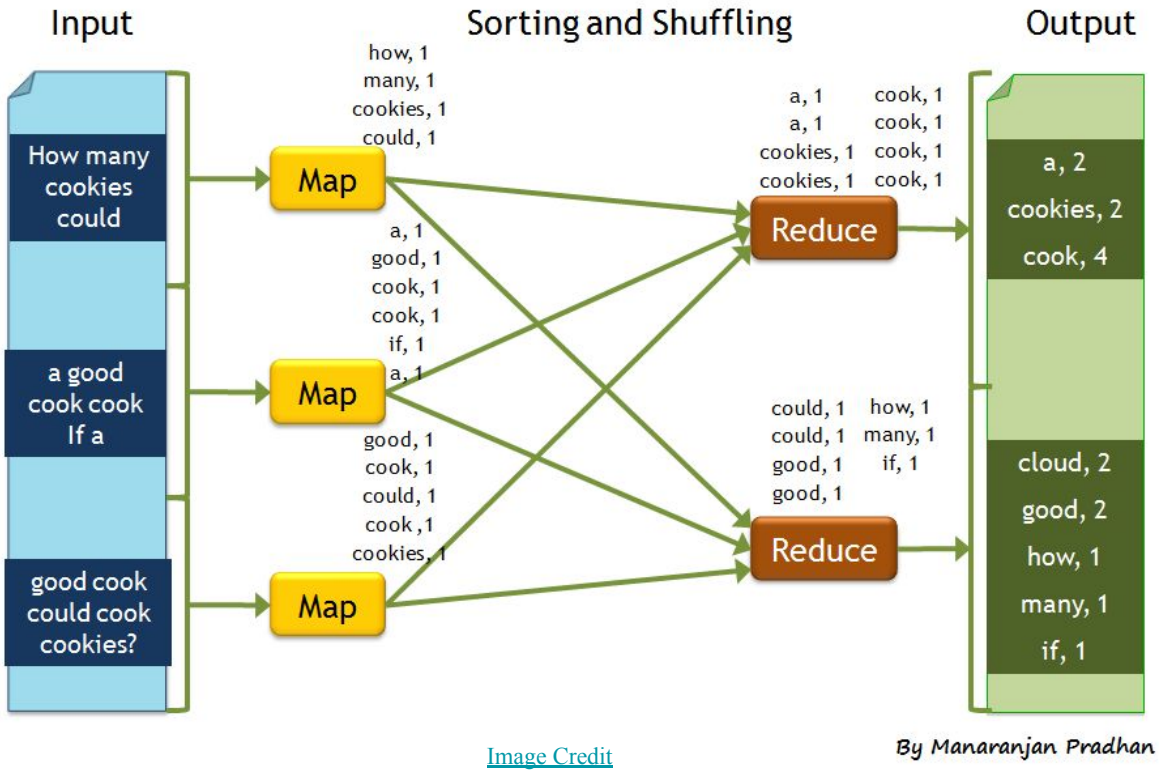


Image Credit

Hadoop MapReduce - Disk based system

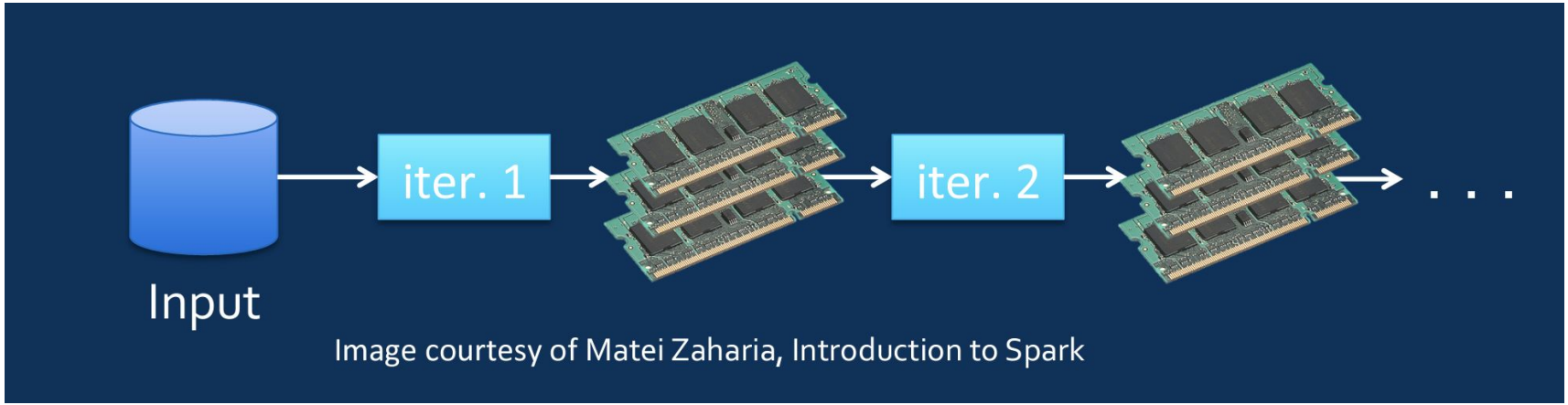
- Runs on Hadoop Distributed File System (HDFS)
- Disk based distributed computing framework
- 2 phases: Map and Reduce
- Input dataset is divided into small chunks and handed off to map tasks
- Intermediate output from map tasks are stored in disk
- Reduce tasks read the intermediate result and produces the final output
- Offline system: the entirety of the input dataset must be loaded at the beginning

Pros and Cons of Disk based systems

- Intermediate results are persisted in disk: High I/O cost
- Data is reloaded from disk for every stage
- Easy failure recovery

Memory based systems

- No heavy I/O cost as intermediate results are persisted in memory
- Best for iterative workloads
- Sensitive to availability of memory
- Possibility of data loss on node failures as data is stored in memory



What is Apache Spark?

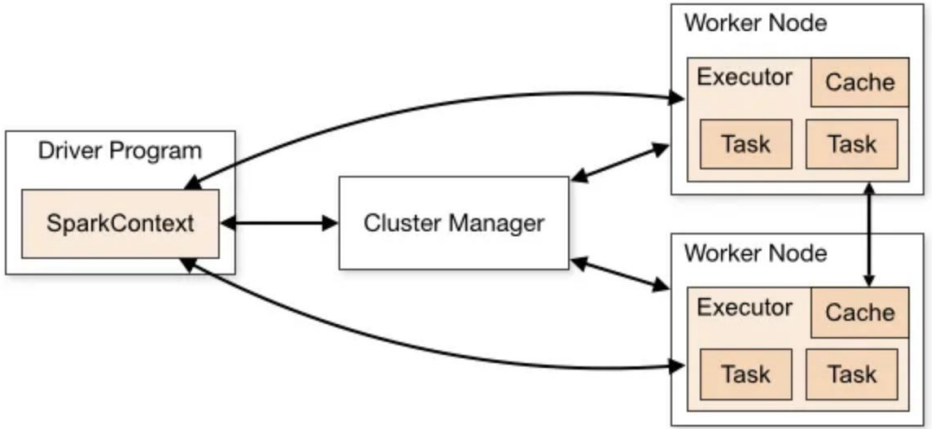
General purpose distributed data processing engine using in memory framework.

Why Spark?

- Addresses a lot of shortcomings of Hadoop MapReduce framework
- 10-100x faster than Hadoop MapReduce
- Operates on real time data. Don’t need all the data upfront
- 1 stop solution for different varieties of big data problems: batch, real-time (almost), machine learning, graphs, etc
- Supports **Scala, Java and Python**
- Can be run on existing Hadoop file system
- Keeps intermediate results in memory resulting in faster computation
- Multi-threaded tasks inside of JVM processes, whereas MapReduce runs as heavier weight JVM processes.
- Faster startup, better parallelism, and better CPU utilization.

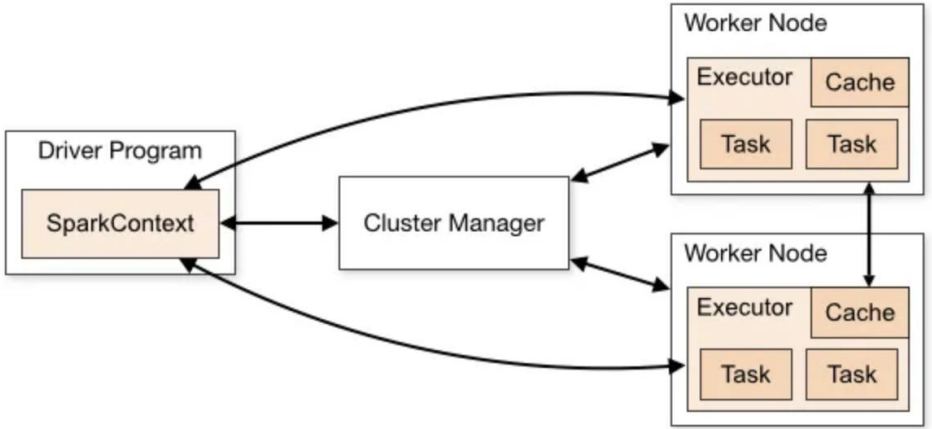
Spark Architecture

- **Master Slave architecture**
  - Master hosts the driver program
  - Slaves host the executors
  - Each executor runs multiple tasks in them
- **Driver**
  - Has spark context object
  - Orchestrator
  - JVM process that is responsible for the execution of the tasks.
- **Worker**
  - Workers are the instances where executors live to execute the user-written application code in the cluster.



Spark Architecture

- **Cluster manager**
  - Responsible for allocating resources across the spark application.
- **Executor**
  - A JVM process launched for an application on a worker node
  - Runs tasks as threads and keeps data in memory or disk storage across them.
  - Each worker node has its own executor.

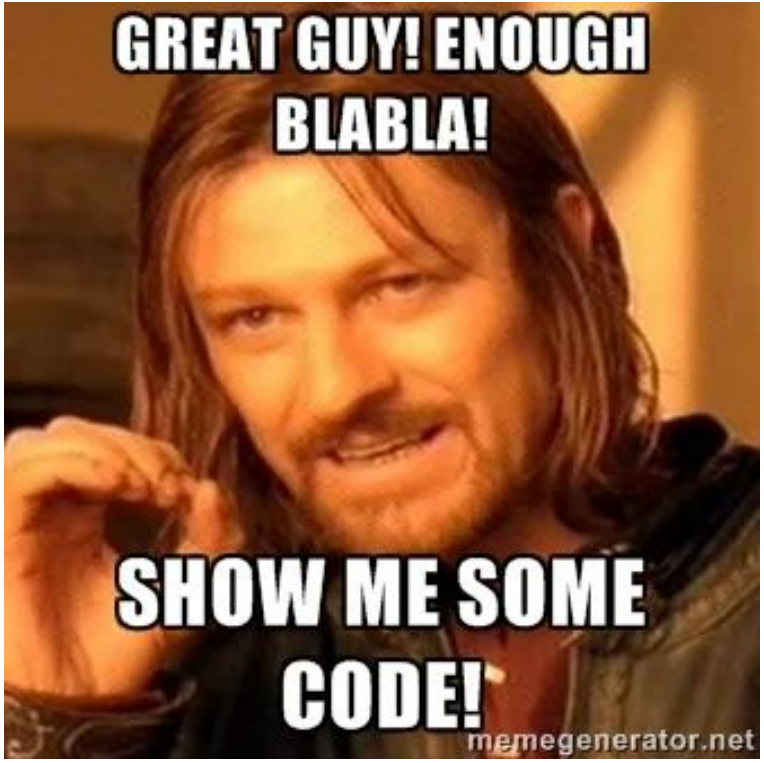


Spark use cases

- **Real-time data processing**
  - Data streams are generated by various sources like log files, sensors, stock trading, bank transactions, etc
  - To get the best result, we have to act and process this data as it arrives.
- **Machine learning**
  - Spark stores intermediate results in-memory and hence can run iterative queries on data faster.
  - This speedy processing is a real need in machine learning, where we need to feed algorithms with well-structured data sets.
  - This helps to train machine learning algorithms.
- **Interactive Analytics**
  - Spark helps in interactive query processing as it can adapt and respond quickly.
  - As a result, data scientists can explore data on the go instead of running pre-defined queries or using static dashboards.

Spark use cases

- **Data integration**
  - Data from different systems across business need to be combined for analysis and reporting
  - Data is often unclean
  - For this purpose, ETL processing (Extract, transform, and load) is performed on this unclean data where they are often pulled from different systems, then cleansing and standardization are performed on it.
  - Finally, the clean data is loaded and analyzed into a separate system.
  - Spark is increasingly being used for this purpose, which reduces the time and cost required for this ETL process.



Spark APIs

- **RDD (Resilient Distributed Datasets)**
  - Most basic data abstraction in spark
  - Fault tolerant collection of elements
  - Immutable
  - An RDD can be distributed across nodes and can be operated on in parallel
  - You can do transformations and take actions on RDD using low-level APIs
  - 2 ways to create RDD
    - Parallelizing an existing collection
    - Reading data from external storage like S3, HDFS, etc
  - Note: RDD is more like a low-level abstraction which we don't use often these days

```
val sc = spark.sparkContext
val myRdd: RDD[(String, Int)] =
sc.parallelize(
Seq(("Jon", 1),
("Tyrion", 2),
("Bronn", 3)))
```

Spark APIs

- **RDD (Continued)**
  - Transformations (Converts an RDD to another)
    - Filter
    - Union
    - Intersection
    - Distinct
    - groupByKey
    - joinByKey, etc
  - Actions (Operates on a RDD to give an answer)
    - Collect
    - Count
    - First
    - SaveAsFile, etc

```
//Transformations
val rdd:RDD[String] =
spark.sparkContext.textFile("src/main/scala/test.txt")
val rdd2 = rdd.flatMap(f=>f.split(" "))
val rdd3:RDD[(String,Int)]= rdd2.map(m=>(m,1))
val rdd4 = rdd3.filter(a=> a._1.startsWith("a"))
println("Count : "+rdd4.count())


//Actions
val firstRec = rdd4.first()
println("First Record : "+firstRec._1 + ","+
firstRec._2)    rdd4.saveAsTextFile("/tmp/wordCount")
```

Spark APIs

- **Dataframe**
  - Distributed collection of data organized into named columns
  - Like tables in MySql
  - Imposes a structure onto the collection of data

```
val dfWithColumnNames: DataFrame =
spark.createDataFrame(myRdd).toDF("name", "ids")
dfWithColumnNames.show(truncate = false)
+-------+---+
|name   |id |
+-------+---+
|Jon    |1  |
|Tyrion |2  |
|Gandalf|3  |
+-------+---+

df.filter($"id" > 1).show()
+-------+---+
|name   |id |
+-------+---+
|Tyrion |2  |
|Gandalf|3  |
+-------+---+
```

Spark APIs

- **Dataframe (Continued)**
  - Dataframes can be read from lot of data sources like json, s3, parquet, etc

```
spark.read.csv("users.txt")
```

  - Data scheme can be inferred

```
dfWithColumnNames.printSchema()
root
 | — name: string (nullable = true)
 | — id: integer (nullable = false)
```

  - Sql can be executed

```
df.createOrReplaceTempView("people")
dfWithColumnNames.createOrReplaceTempView("got")
val sqlDF = spark.sql("SELECT * FROM got where name = 'Jon' ")
sqlDF.show()
+----+---+
|name| id|
+----+---+
| Jon|  1|
+----+---+
```

Spark APIs

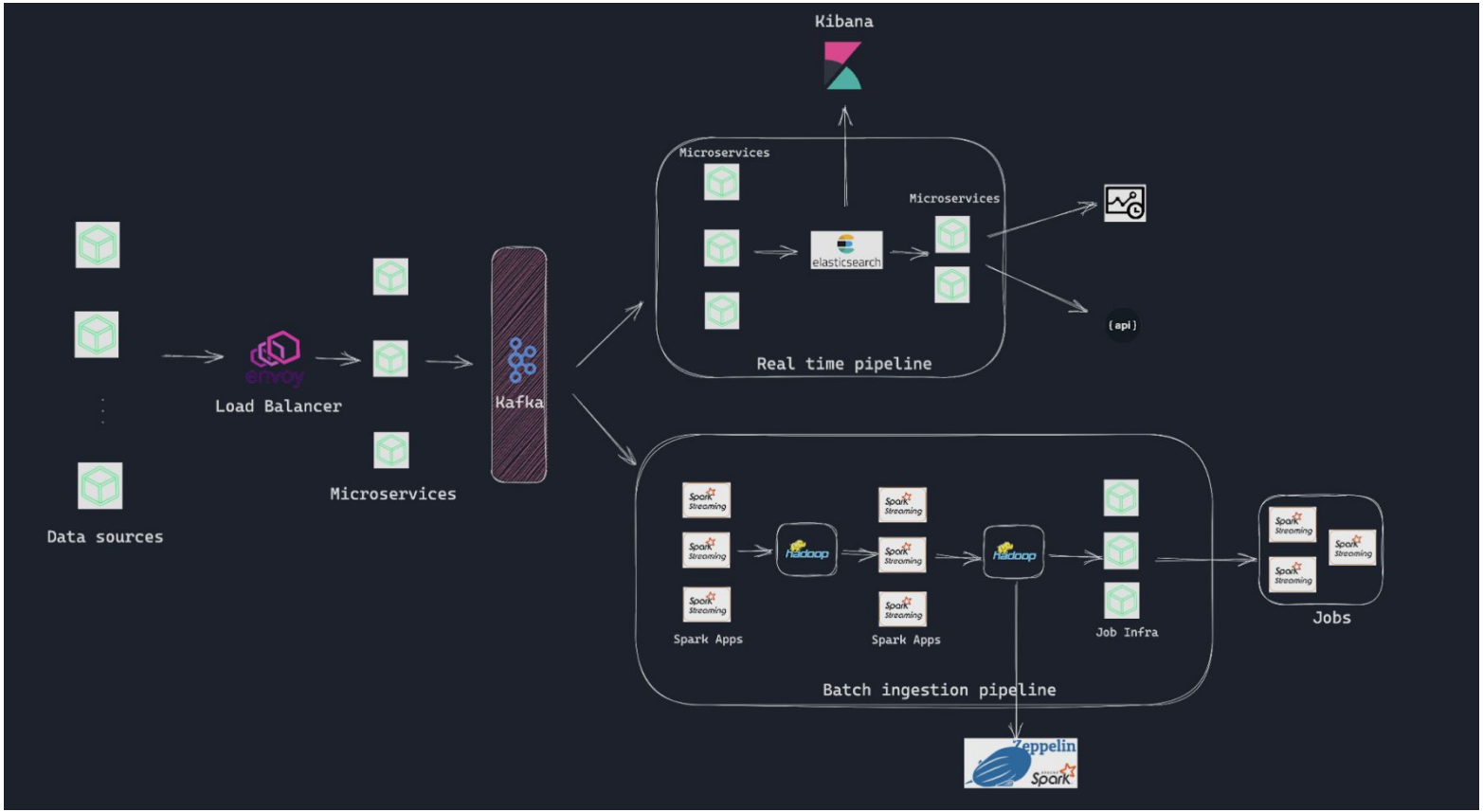
- **Dataset**
  - Strongly typed domain specific objects
  - Can be transformed in parallel
  - Main advantage: Type safety
  - Syntax errors and analytic errors can be caught during compile time

```
case class Characters(name: String, id: Int)//Note this
class should be outside your main object.

import spark.implicits._
val data = Seq(Characters("Jon", 21),
Characters("Tyrion", 22), Characters("Gandalf", 19))
val ds = spark.createDataset(data)


//To read data from a Datasource
val path: = "examples/src/main/resources/Characters.csv"
val peopleDS = spark.read.csv(path).as[Characters]
peopleDS.filter(_.id%2 ==0).show()
val ds1 = dfWithColumnNames.as[Characters]
ds1.filter(_.name.startsWith("J")).show()
```

Data Pipeline with Spark



Simple Demo

https://www.machinelearningplus.com/pyspark/install-pyspark-on-mac/

Backend Study Group

References:

- https://courses.cs.duke.edu/spring15/compsci516/Lectures/LanceIntroSpark.pdf
- http://pulasthisupun.blogspot.com/2016/06/apache-hadoop-detailed-word-count.html
- https://towardsdatascience.com/apache-spark-101-3f961c89b8c5
- https://medium.com/@achilleus/apache-spark-101-971aaf5d4334
- https://developer.hpe.com/blog/spark-101-what-is-it-what-it-does-and-why-it-matters/
- https://contenteratechspace.com/blog/apache-spark-101-what-is-it-and-why-it-matters/
- https://www.machinelearningplus.com/pyspark/install-pyspark-on-mac/
- https://sparkbyexamples.com/spark-rdd-tutorial/?expand_article=1
-

Backend Study Group:

- Presentations on GitHub and session recordings available on WWCode YouTube channel
- Upcoming sessions:
  - Aug 10th, 2023- Domain Driven Design
  - Sep 21st, 2023- Web Development 101
  - Oct 12th, 2023- DevOps 101

Women Who Code:

- Technical Tracks and Digital Events for more events
- Join the Digital mailing list for updates about WWCode
- Contacts us at: contact@womenwhocode.com
- Join our Slack workspace and join *#backend-study-group*!

You can unmute and talk or use the chat

WOMEN WHO CODE /san-francisco