"unified analytics engine"

Мотивация

- Проблемът с обема на данните се задълбочава в началото на XXI век (2003-4 година)
- Гугъл са едни от първите, които го адресират:
  - MapReduce:

https://static.googleusercontent.com/media/research.google.com/en//archive/mapreduce-osdi04.pdf

  - GFS:

https://static.googleusercontent.com/media/research.google.com/en//archive/gfs-sosp2003.pdf

HDFS и Hadoop

- Появява се HDFS(hadoop file system) дистрибутирана файлова система, която работи върху “commodity computers”.
- MapReduce programmatic framework - на различни езици, предимно Java, операции, които се случват на диска.
- Дълго време Hadoop == datalake
- Hadoop има двойнствено значение HDFS || екосистема(Hive, Impala, Kafka, ZooKeeper, Flink, Flume, Pig, Hbase…)
- Необходимост от по-бърз и унифициран подход към обработката на данни

Data Lake

Spark

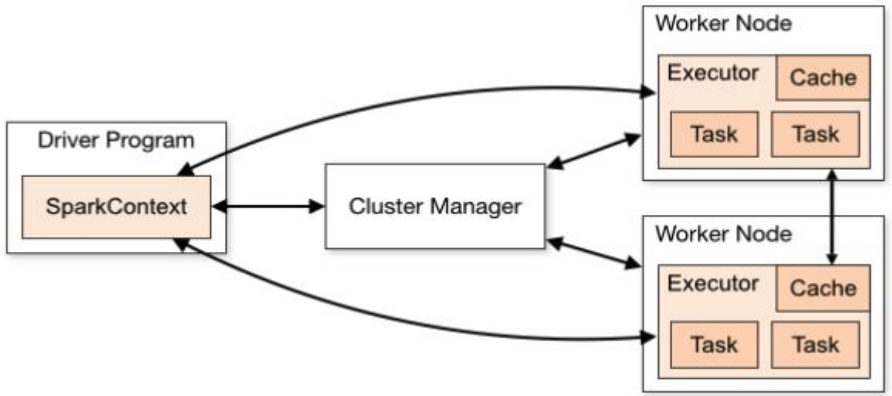
- Проектът е стартитан от Матей Захария през 2009 година като докторска дисертация в Бъркли.
- През 2014 година е дарен на Apache Software Foundation
- По-късно М.З. създава DataBricks(~40B market cap) - основна движеща сила на Spark
- “Apache Spark™ is a multi-language engine for executing data engineering, data science, and machine learning on single-node machines or clusters”
- “Unified engine for large-scale data analytics”

Преимущества на Apache Spark

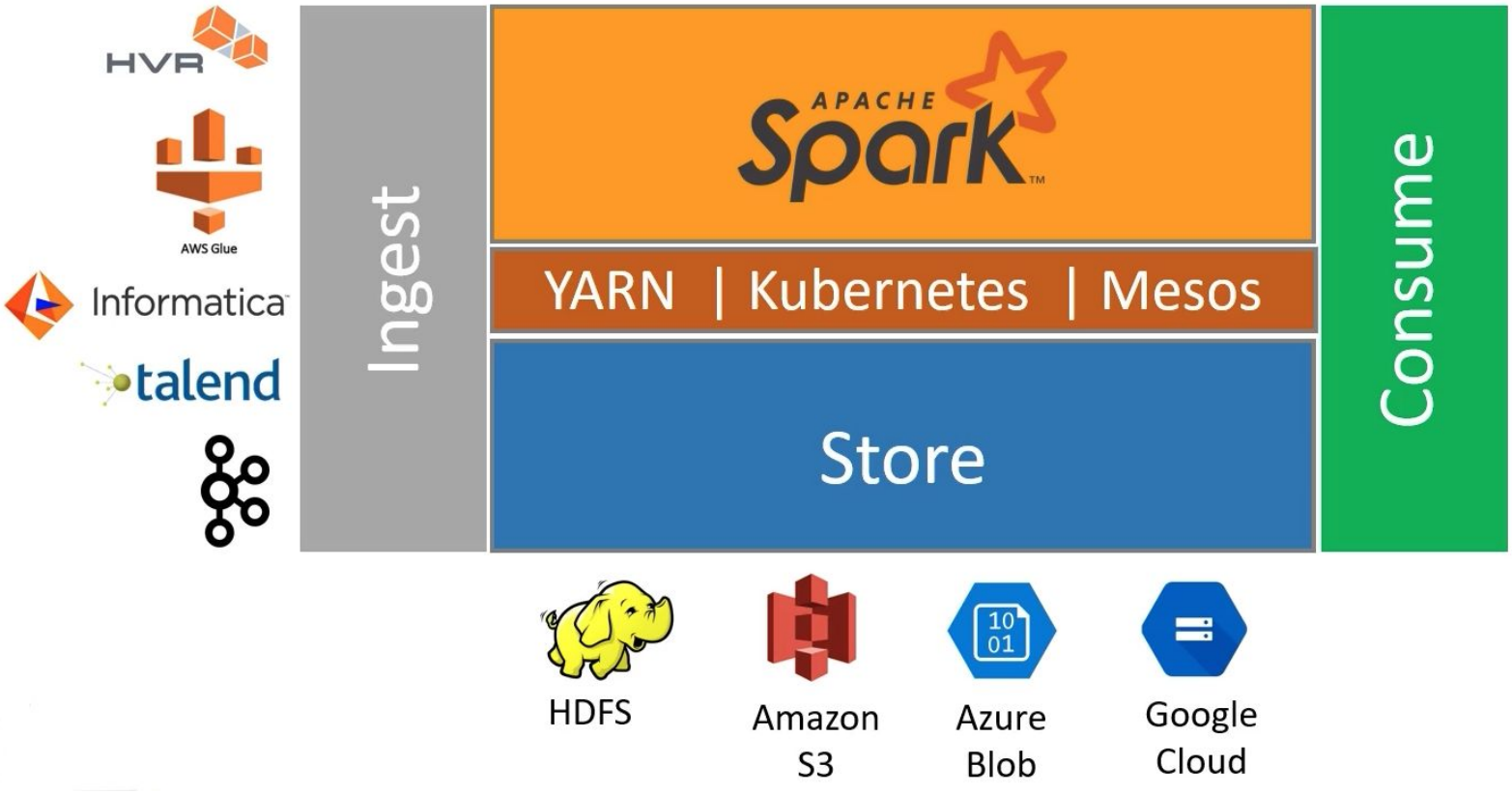
- За разлика от MapReduce, работи с in-memory computation(~100x по-бърз)
- Fault-tolerant
- Разпределена обработка чрез паралелизъм
- Унифицирана платформа
- Работи с много клъстер-мениджъри(абстрактно):
    - YARN(default)
    - Mesos
    - K8s(popular)
    - Spark Standalone

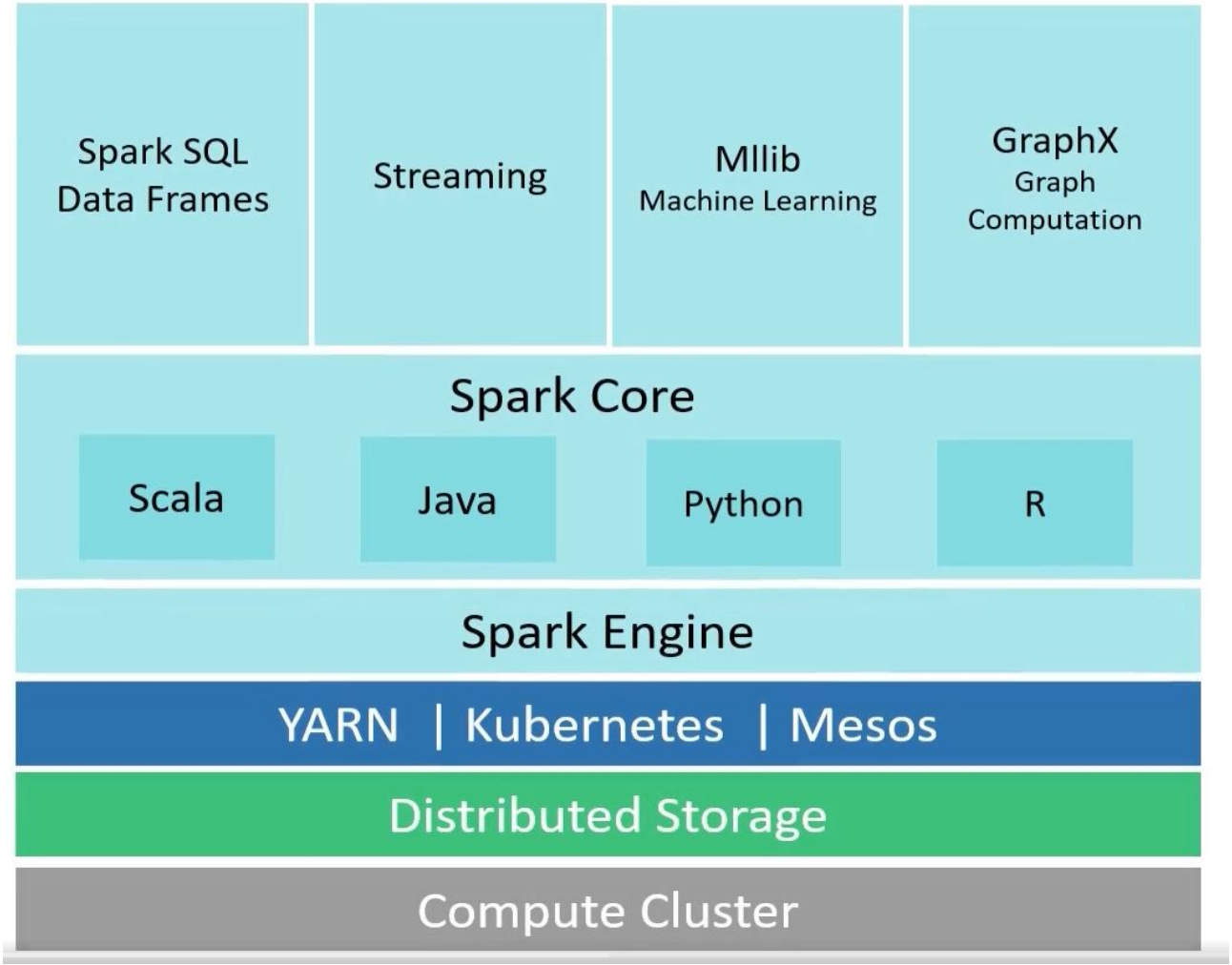
Архитектура

- Работи в master-slave(driver-worker) архитектура. Драйвърът създава контекст - “entry point” към програмата(spark application) и всички операции се изпълняват от worker-ите, докато ресурсите се менажират от клъстер-мениджъра



Source: https://spark.apache.org/


HVR
AWS Glue
Informatica
talend
Ingest
APACHE Spark
YARN | Kubernetes | Mesos
Store
Consume
HDFS
Amazon S3
Azure Blob
Google Cloud


Spark SQL
Data Frames
Streaming
Mllib
Machine Learning
GraphX
Graph
Computation
Spark Core
Scala
Java
Python
R
Spark Engine
YARN | Kubernetes | Mesos
Distributed Storage
Compute Cluster

Resilient Distributed Dataset(RDD) and Spark DataFrame(DF)

- RDD:
  - Основен spark обект
  - Съхранява се в паметта
  - Разделен на partitions
  - Read-only(не се променя immutable)
  - Resilient

DF

1. Абстракция върху RDDs
2. Замества RDD като начин за боравене с данни в Spark
3. Колони и редове

```
root
 |-- age: integer (nullable = true)
 |-- sex: integer (nullable = true)
 |-- cp: integer (nullable = true)
 |-- trestbps: integer (nullable = true)
 |-- chol: integer (nullable = true)
 |-- fbs: integer (nullable = true)
 |-- restecg: integer (nullable = true)
 |-- thalach: integer (nullable = true)
 |-- exang: integer (nullable = true)
 |-- oldpeak: double (nullable = true)
 |-- slope: integer (nullable = true)
 |-- ca: integer (nullable = true)
 |-- thal: integer (nullable = true)
 |-- target: integer (nullable = true)
```

Spark Job

- Състои се от spark stages и всеки stage е множество от задачи(spark tasks)
- Стъпките, които обикновено се изпълняват са зареждане на данни, трансформиране, съхраняване
- Представени са в spark DAG, който представлява логическа репрезентация на spark job-а

Операции в Spark

- Трансформации:
  - Случват се в worker
  - Narrow
    - Изпълняват се паралелно в partitions
    - Примери: select(), filter(), withColumn(), drop()...
  - Wide
    - Изпълняват се след като се групират данните от множество partitions
    - Примери: groupBy(), join(), cube(), agg()....

partitions в Spark по подразбиране == cpu cores | machines

Операции в Spark

- Действие(Action)
  - Стартират изпълнението на job(workers <=> driver)
  - Примери:
    - read()
    - write()
    - collect()
    - take()
    - count()

Работа със Spark

- spark-submit | cluster mode
  - Изпълнява се на клъстер, до които потребителя има достъп(напр. през ssh)

- Interactive mode - shell | notebook

Demo

Set master:

spark-class org.apache.spark.deploy.master.Master --host 127.0.0.1

Set worker:

spark-class org.apache.spark.deploy.worker.Worker spark://127.0.0.1:7077 --host 127.0.0.1

Execute:

spark-submit --master spark://127.0.0.1:7077 --name SparkDemo --total-executor-cores 4 --executor-memory 2g --executor-cores 2 <script name>

Spark in the cloud

- Databricks
- AWS
  - Glue
  - EMR on EC2
  - EMR Serverless

- Azure
  - Azure DB
- GCP
  - DataProc

Some alternatives

- Ballista
  - https://github.com/apache/arrow-ballista
- Apache Flink
  - https://flink.apache.org/
- Pandas
  - https://pandas.pydata.org/
- PolaRS
  - https://pola.rs/

Links

- https://spark.apache.org/docs/latest/index.html
- https://www.databricks.com/glossary/what-is-apache-spark
- https://github.com/apache/spark
- https://spark.apache.org/docs/latest/api/python/index.html
- https://spark.apache.org/examples.html
- https://docs.delta.io/latest/index.html
- https://www.dremio.com/blog/introduction-to-apache-iceberg-using-spark/
- https://github.com/rsradev/distc