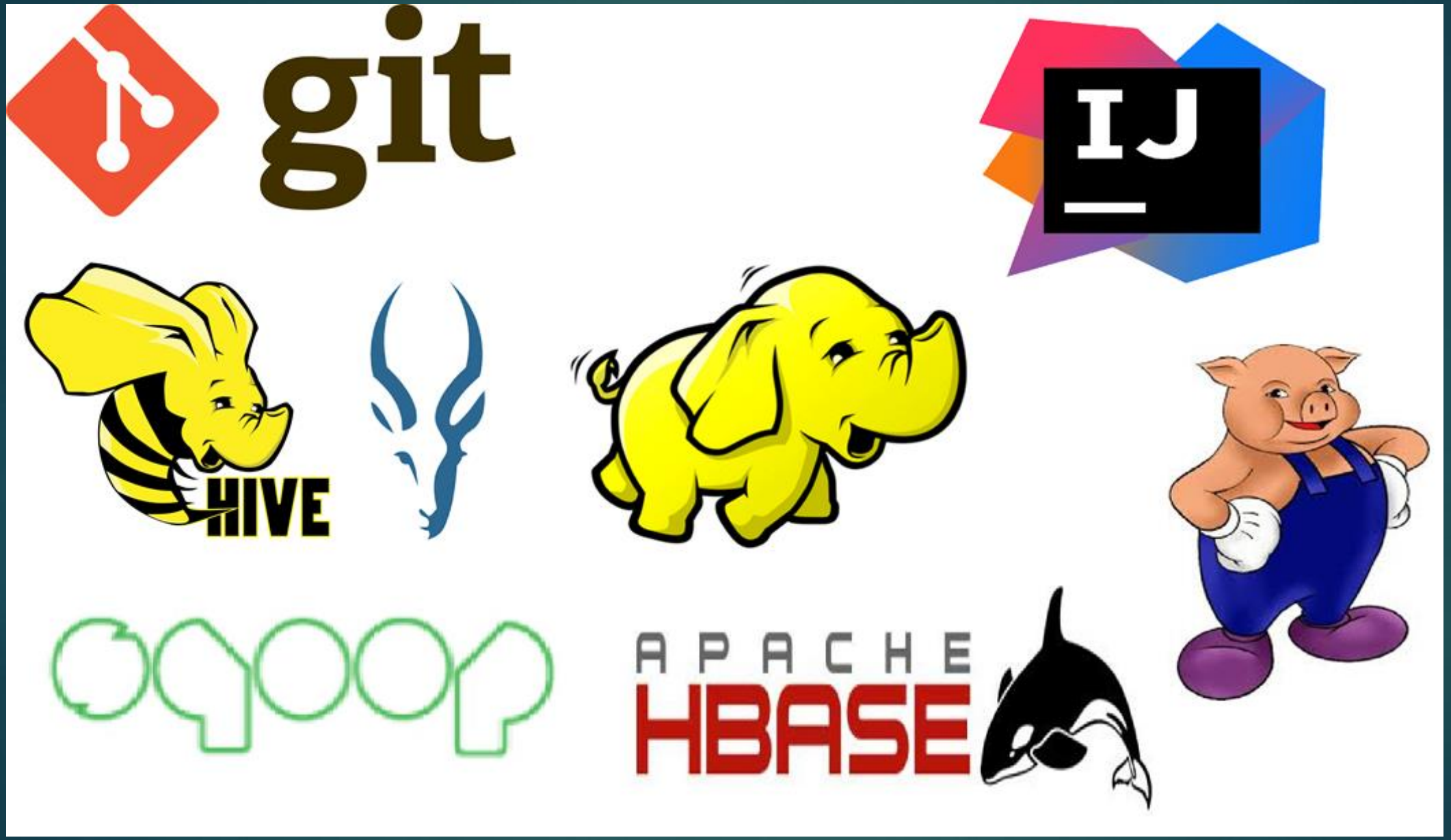

git
IJ
HIVE
APACHE
HBASE

Recap

- HDFS
- Map Reduce
- Pig
- Hive
- Impala
- Sqoop
- HBase

Hadoop

Processing nature	Tool
Batch processing on unstructured data	Pig
Batch processing on structured data	Hive
Ad-hoc analysis on structured data	Impala
Machine Learning	Apache Mahout
Graph processing	Apache Giraph
Stream processing	Apache Storm/Kafka

Lots of tools

Image Ref: http://hunteryoung.com/wp-content/uploads/2017/05/Too-many-Tools-1160x770.jpg

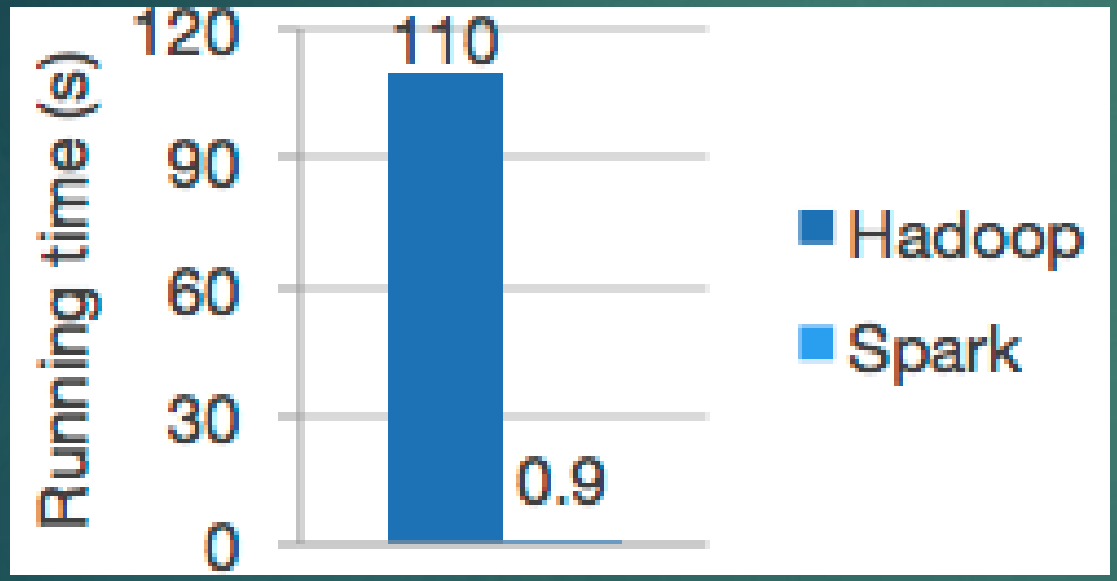

Spark

Why Spark?

Processing nature	Spark
Batch processing on unstructured data	Spark RDD API
Batch processing on structured data	SparkSQL
Ad-hoc analysis on structured data	SparkSQL
Machine Learning	MILib
Graph processing	Graphax
Stream processing	Spark Streaming

Why Spark: cont…

- Faster



Logistic regression in Hadoop and Spark

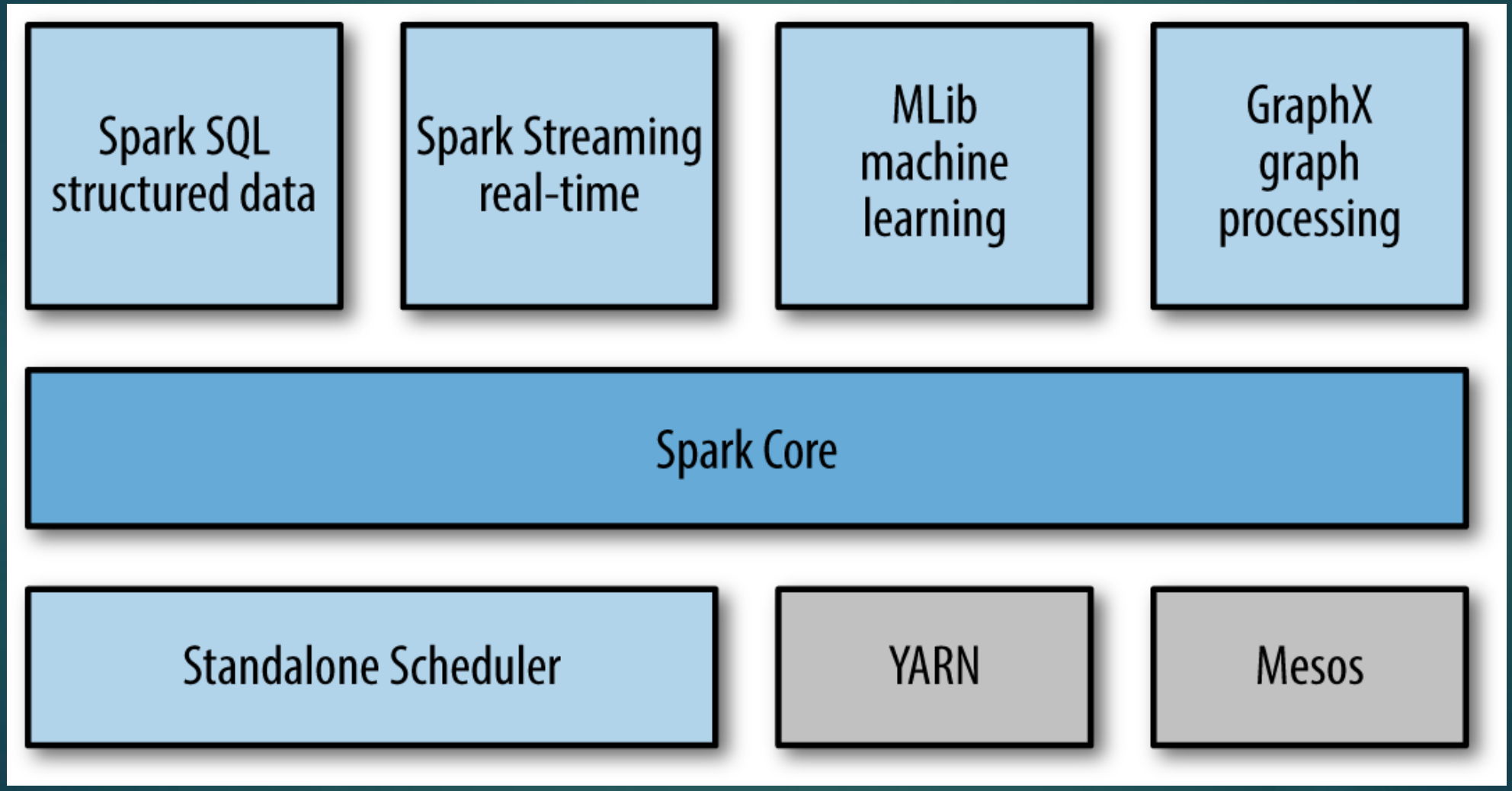
Why Spark: cont…

- Ease of use
- Support for multiple languages
- REPL for development and ad-hoc analysis
- Fewer lines of code

Why Spark: cont…

- Inter operability with other platforms
  - Hadoop
  - Mesos
  - HBase
  - Cassandra

Components


Spark SQL
structured data
Spark Streaming
real-time
MLib
machine
learning
GraphX
graph
processing
Spark Core
Standalone Scheduler
YARN
Mesos

Core of Spark: RDD

- **R**esilient **D**istributed **D**ataset

  Fault tolerant

  Distributed across multiple processes

  Source could be a file or program generated

- Immutable collection of elements, partitioned across multiple processes to operate in parallel

RDD Operations

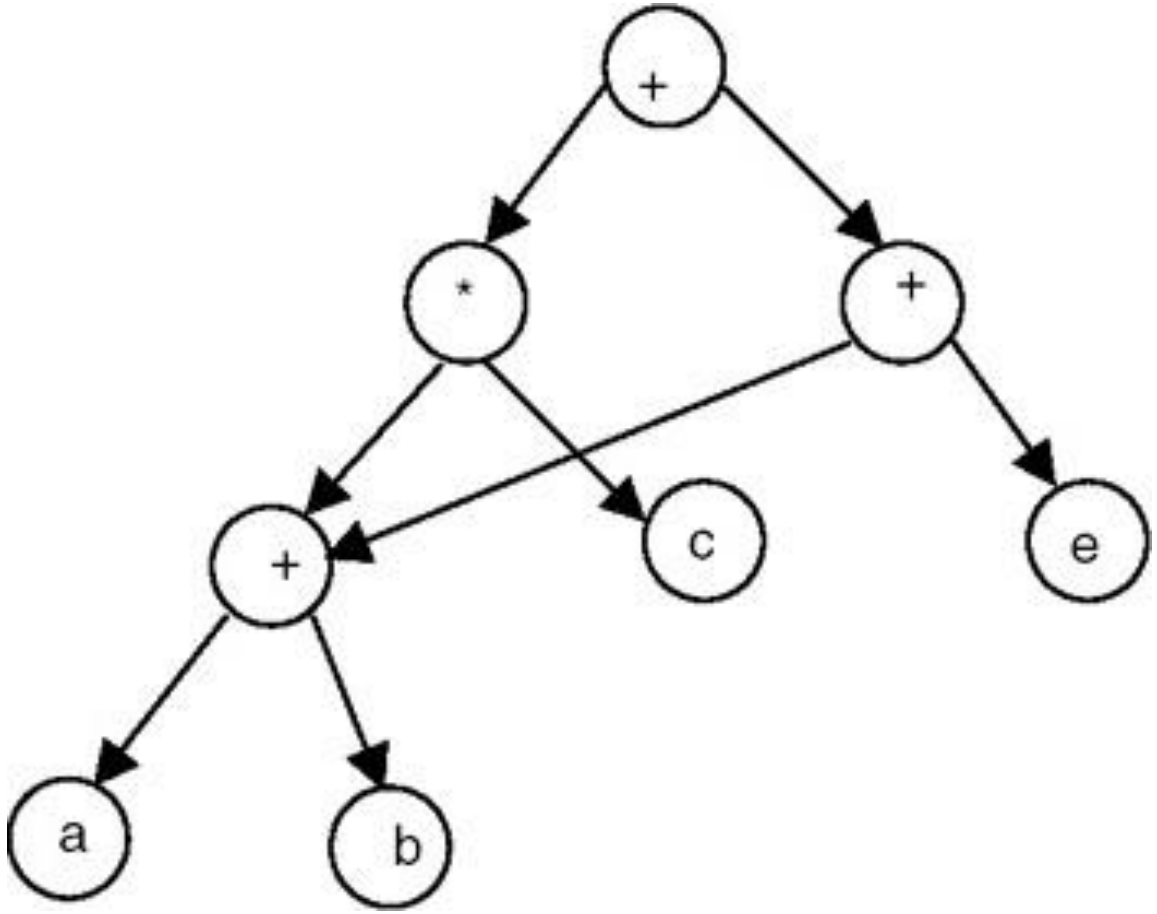
- Transformation
- Action

RDD Operation: Transformation

- Can be applied on any RDD
- Generates another RDD as result
- Example: map, flatMap, filter, reduceByKey…
- Evaluated lazily

DAG

- Directed Acyclic Graph prepared to indicate task dependencies



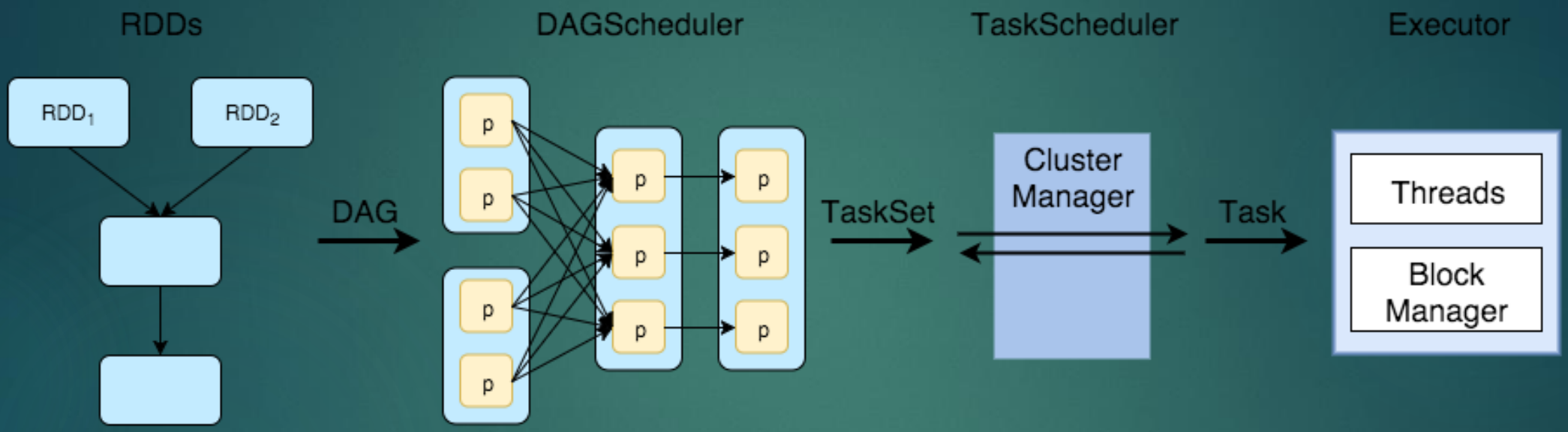
RDD Operation: Action

- Call for evaluation of complete DAG
- Can be applied on any RDD
- May generate result on driver program
- Example: count, take, saveAsTextFile, collect…

Parallelism

- Tasks within a stage will be executed when stage is ready to execute
- Shuffle operation is the stage boundary

Execution and Coordination



Ref: http://datastrophic.io/core-concepts-architecture-and-internals-of-apache-spark/

Launch Spark shell

- Standalone:

./bin/spark-shell --master spark://IP:PORT

- YARN

./bin/spark-shell --master yarn

- Mesos

./bin/spark-shell --master mesos://host:5050

Submit Application

- Scala/Java

spark-submit --class WebHitCount --master local --deploy-mode client --executor-memory 1g --name WebHitCount --conf "spark.app.id=WebHitCount" SparkWebHitCount.jar <other parameters to JAR file>

- Python

spark-submit --master yarn --deploy-mode client --executor-memory 1g --name WebHitCount --conf "spark.app.id=WebHitCount" webhitcount.py <Other parameters>

REPL

- Scala

spark-shell

- Python

pyspark

Language comparison matrix

Metrics	Scala	Java	Python	R
Type	Compiled	Compiled	Interpreted	Interpreted
JVM based	Yes	Yes	No	No
Verbosity	Less	More	Less	Less
Code Length	Less	More	Less	Less
Productivity	High	Less	High	High
Scalability	High	High	Less	Less
OOP Support	Yes	Yes	Yes	Yes


Spark
SQL

Why SparkSQL?

- Integrated

```
context = HiveContext(sc)
results = context.sql(
  "SELECT * FROM people")
names = results.map(lambda p: p.name)
```

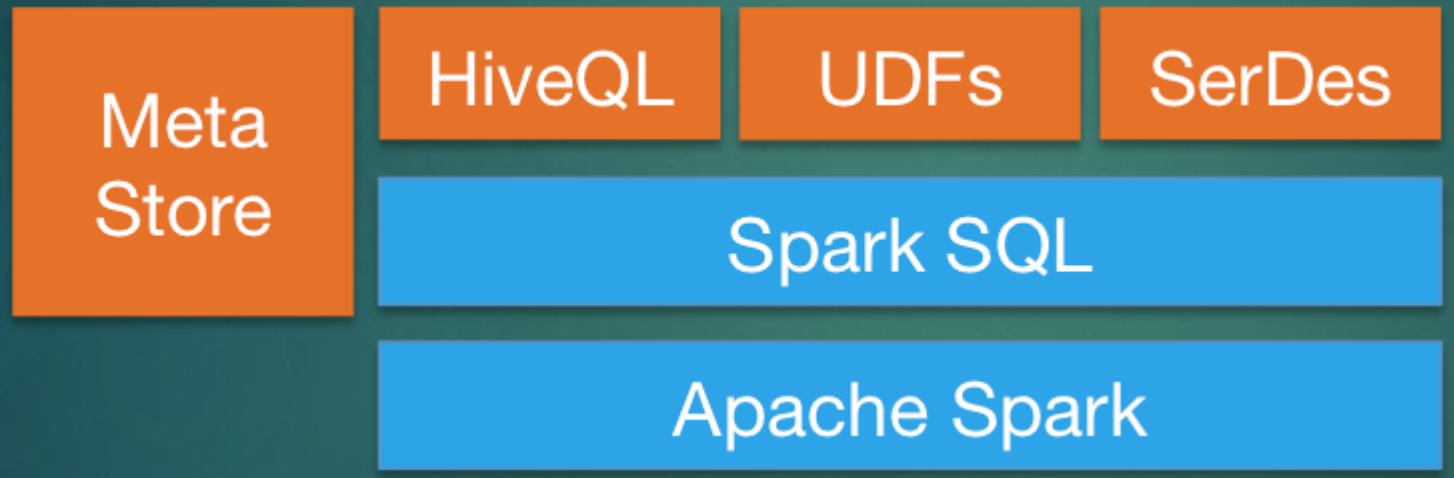
Why SparkSQL: cont…

- Uniform Data access

```
context.jsonFile(“filename.json”)
  .registerTempTable(“json”)
results = context.sql(
  """SELECT *
     FROM people
     JOIN json ..."""")
```

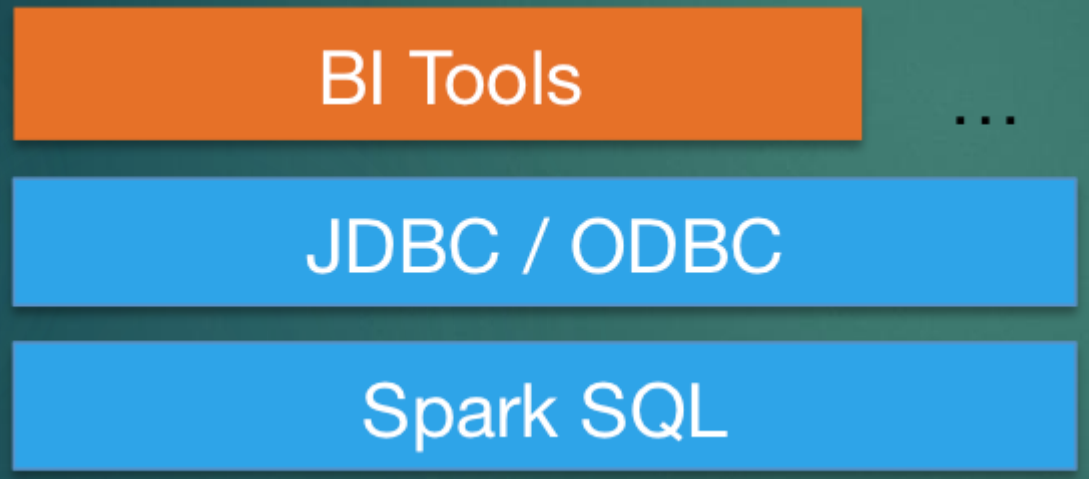
Why SparkSQL: cont…

- Hive Integration



Why SparkSQL: cont…

- Standard Connectivity



Data holders

- RDD
- Dataframe: RDD with schema
- Dataset:
  introduced in 1.6 version
  provides strong type over RDD

Data source

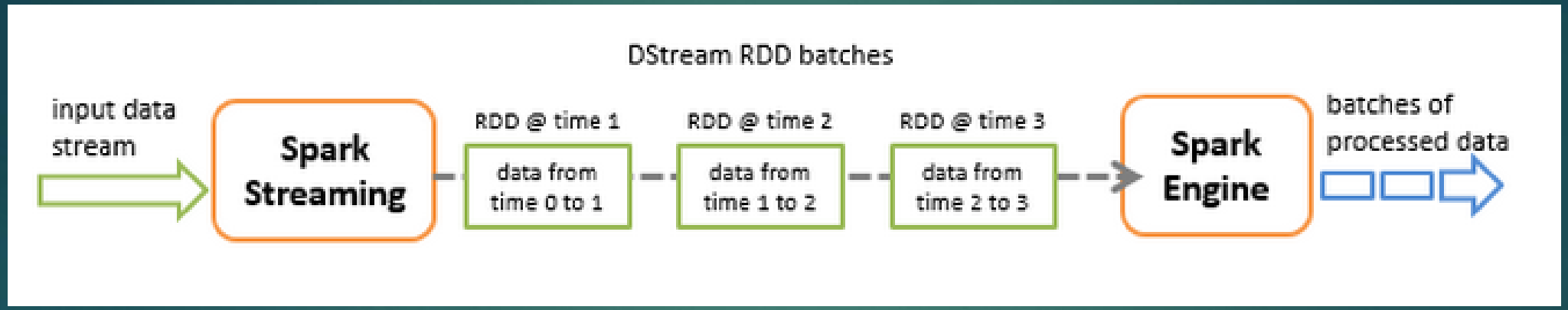
- Hive existing table
- Structured files. Json file for example
- RDD

Hive integration

- Can use most of the SQL features available in Hive
- Insert data through Spark and read in Hive
- Executes DDL statements
- Refers Hive metastore for metadata


Spark
Streaming

DStream


DStream RDD batches
input data stream
Spark Streaming
RDD @ time 1
data from time 0 to 1
RDD @ time 2
data from time 1 to 2
RDD @ time 3
data from time 2 to 3
Spark Engine
batches of processed data

Source

- Kafka
- Flume
- HDFS/S3
- Amazon Kinesis
- Twitter

Storage/Target

- HDFS
- Databases
- Dashboard

Program flow

- Set streaming context
- Define source for the streaming context
- Apply all transformations of Dstream
- Start the streaming context

Persist/Cache data

- Helpful to reuse the same dataset
- Multiple storage levels:

https://spark.apache.org/docs/latest/rdd-programming-guide.html

- How to check current storage level:
  <Object name>.getStorageLevel

Examples

- https://github.com/apache/spark/tree/master/examples/src/main/scala/org/apache/spark/examples

References

- http://spark.apache.org/docs/1.3.0/cluster-overview.html
- Hadoop: the definitive guide 4th edition
- https://www.cloudera.com/documentation/enterprise/5-6-x/PDF/cloudera-spark.pdf
- https://databricks.com/product/getting-started-guide/quick-start
- Cloud hosted community spark setup

  https://community.cloud.databricks.com/