Spark

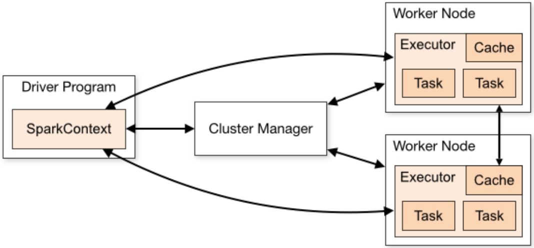
Schwartz

July 14, 2017

indeed.com “New York City, New York” keyword search

Spark, the successor to Hadoop MapReduce, is designed to work in-memory and have a simpler API that allows use of Python, R, Scala and Java code in a distributed computing framework

Search	Jobs
“Data Science”	8,720
“Statistics”	5,315
“Big Data”	4,654
“Time Series”	1,523
“Machine Learning”	1,321
“A/B Testing”	504
“Multi-Armed Bandit”	1



“You don’t have to be an engineer to be be a racing driver, but you do have to have Mechanical Sympathy.” – Jackie Stewart, F1

This is especially true for big data. If you don’t have sympathy for the system, you’ll write inefficient code that will tax your hardware and take forever to run. You need to know a little bit about how big data tools work so that you can use them correctly.

Outline

- Spark (vs Hadoop)
- Resilient Distributed Dataset (RDD)
- Sample Functions

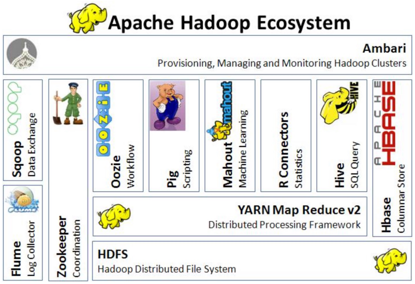
Spark ~~and~~ VERSUS Hadoop

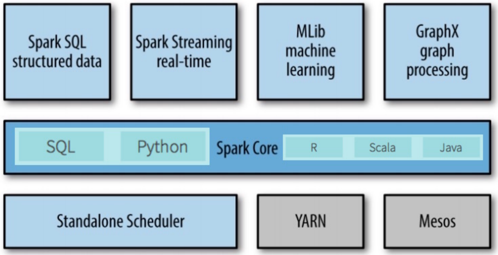
HadOOPS!!!

- <key,value> paradigm
- ecosystem interaction programming
- on (slow) hard disk
- no iterative (machine learning) tools

spAR!!!k

- other workflows provided
- no interaction programming
- memory (up to 100x faster)
- does machine learning





Resilient Distributed Dataset (RDD)

Abstractions:

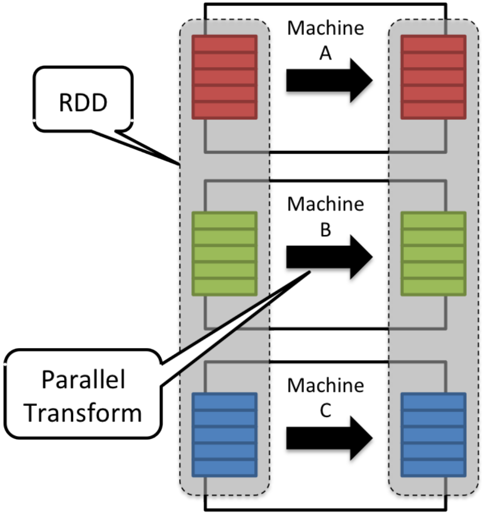
- History-Based Fault-Tolerance*

*No built in redundancy

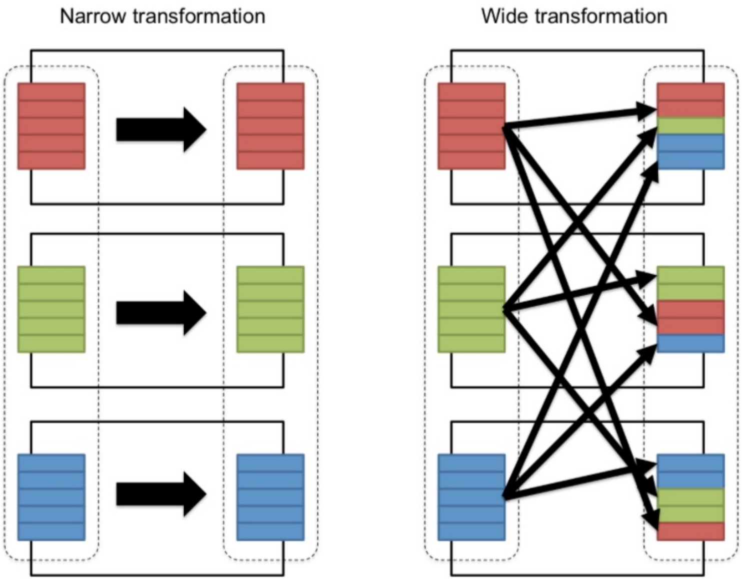
- Distributed Data Partitioning
- Partitioned Parallel Operation

Characteristics:

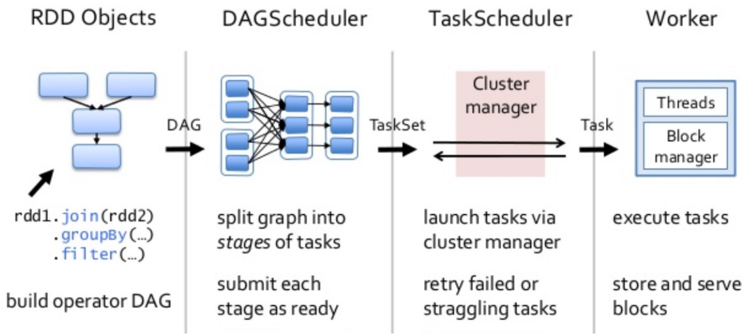
- Immutable
- Lazy
- Cachable



Transactional Complexity



Execution



Characteristics:

- Immutable
- Lazy
- Cachable

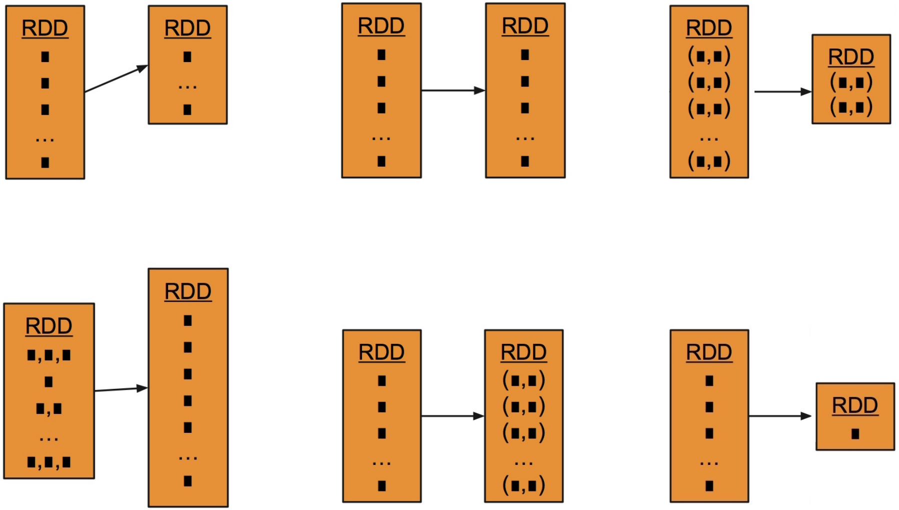
Transform (Lazy):

- Filter
- Map
- FlatMap
- ReduceByKey* (f)

Action (Execute):

- First/Take (n)
- Collect (all)
- Count
- Reduce* (f)

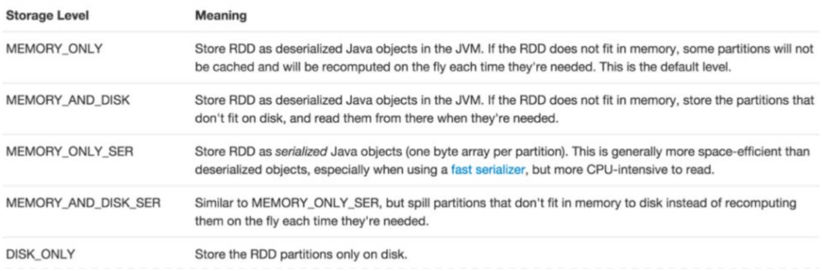
Quiz



Other Important Things

- cache()
- persist()
- saveAsTextFile()

- Cache *wide* transformations
- Not *Narrow* transformations
- Prototype on small data sets



Storage Level	Meaning
MEMORY_ONLY	Store RDD as deserialized Java objects in the JVM. If the RDD does not fit in memory, some partitions will not be cached and will be recomputed on the fly each time they're needed. This is the default level.
MEMORY_AND_DISK	Store RDD as deserialized Java objects in the JVM. If the RDD does not fit in memory, store the partitions that don't fit on disk, and read them from there when they're needed.
MEMORY_ONLY_SER	Store RDD as *serialized* Java objects (one byte array per partition). This is generally more space-efficient than deserialized objects, especially when using a fast serializer, but more CPU-intensive to read.
MEMORY_AND_DISK_SER	Similar to MEMORY_ONLY_SER, but spill partitions that don't fit in memory to disk instead of recomputing them on the fly each time they're needed.
DISK_ONLY	Store the RDD partitions only on disk.

- parallelize(data, 16), textFile(data, 16), repartition(16)
- Spark recommends 2-4 partitions per core in cluster
- partition by key for GroupByKey, ReduceByKey, etc.

Terminology

- *Job*: code that reads input, performs some computation on it,and produces some output data
- *Stages*: parts of *jobs* distinguished based on sequential computational map or reduce boundaries
- *Tasks*: parts of a *stage* that are done on a partition of one *executor*(machine)
- *DAG*: Directed Acyclic Graph
- *Executor*: process responsible for executing a *task*
- *Driver*: The program/process responsible for running the *job* over the Spark Engine
- *Master*: machine on which the *driver* runs
- *Slave/Worker*: a machine on which the *executor* program runs

Odds and Ends

- HDFS, S3, HBase, JSON, text, local

- Accumulator (stored on master – accessible on workers)
- Broadcast (read only made available to all workers)

- Lin/Log Reg, SVM, NB, RF, GBT
- NMF, SVD, PCA, Kmeans, LDA
  - LabeledPoint(Target, Feature)

- `https://github.com/point0five/spark-install`
- `https://github.com/gSchool/dsi-spark`
- `https://github.com/gSchool/spark-shortcourse`