Spark Basics

onTarget
TRAINING SOLUTIONS

Agenda

- Hello World in Spark
- RDDs overview
- RDDs operations
-

Simple Spark Example

```
val conf = new SparkConf()
  .setAppName("standalone app")

val sc = new SparkContext(conf)

val count = sc
  .textFile("README.md")
  .filter(l => l.contains("Python"))
  .count()

println(“Lines containing python: “ + count)
```

Spark Basics - RDDs

- RDDs - Resilient Distributed Dataset
  - Immutable collection of objects, distributed across the cluster
  - Split into partitions to support parallelism
  - Create from external sources (HDFS, S3, etc) or in code
- RDD transformations
  - Return a new RDD (RDDs are immutable)
  - Lazily executed
  - Example: map, flatMap, filter
- RDD actions
  - Output to external systems (HDFS), or return to driver program
  - Force execution of (lazy) transformations
- Spark code just **builds** lineage graph (DAG of RDDs)
- RDDs can be recomputed and persisted

onTarget
TRAINING SOLUTIONS

Spark Basics - RDDs (cont)

- Transformations
  - Basic
    - map - could return a different type
    - flatMap - fn returning iterator, "merges" the iterators across the elements
    - filter
    - sample(withRepl, frac, seed)
  - Set-like
    - distinct - slow, requires shuffle
    - union - fast, doesn't remove dupes
    - intersection - slow, also removes dupes (like distinct)
    - subtract
  - cartesian - slow!

onTarget TRAINING SOLUTIONS

Spark Basics - RDDs (cont)

- Actions
  - count : return total number elements in RDD
  - collect : just take all of it and return to "driver"
  - take(n) : minimizes num partitions it accesses => could have bias
  - takeSample : like take but randomized
  - top(n) : like take, but sorts first
  - topOrdered(n, ordering) : custom ordering fn
  - reduce : fold without initial
  - fold : initial value + op
  - aggregate : fold with combiner
  - foreach : just run code, without data going to driver

Spark - Caching

- Example:

```
val result = input.map(x => x * x)
// result.persist(StorageLevel.DISK_ONLY)
println(result.count())
println(result.collect().mkString(","))
```

- Levels
  - MEMORY_ONLY
  - MEMORY_ONLY_SER
  - MEMORY_AND_DISK (spill to disk if mem is not enough, does not Serialize in memory)
  - MEMORY_AND_DISK_SER (spill to disk if mem is not enough, does Serialize in memory)
  - DISK_ONLY
- on eviction, it uses LRU, del or disk
- unpersist -- manually evict something from cache

Spark Basics - RDDs

- Pair transformations (**PairRDDFunctions**)
  - reduceByKey : like reduce, but acts per key
  - foldByKey : like fold, but acts per key
  - combineByKey(createCombiner, mergeValue, mergeCombiners) : like aggregate
    - paralelism/partitions can be configured on all combine* transformations
  - mapValues : preserves partition
  - flatMapValues
  - keys
  - values
  - sortByKey
  - join, leftOuterJoin, rightOuterJoin, cogroup
  - repartition
  - Coalesce : like repartition but only reduces

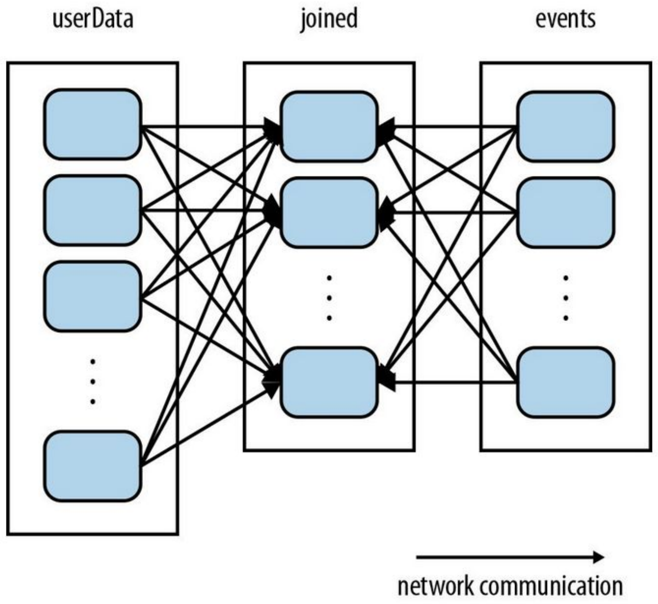
Spark Basics - Pair RDD partitioning

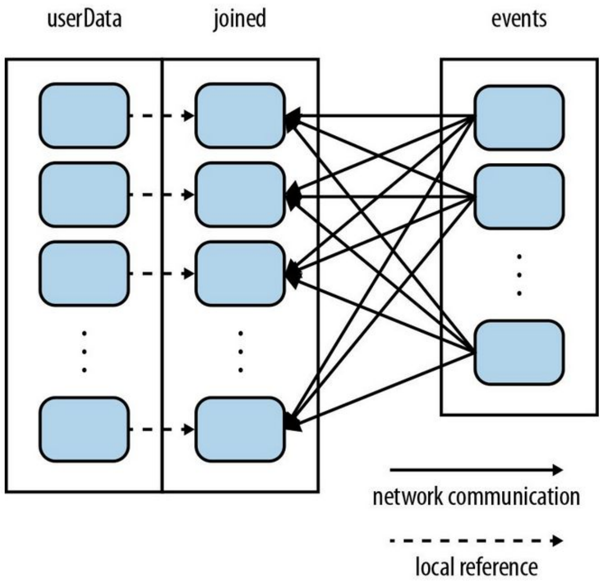
- Defines parallelism
- Defines data locality
- Crucial for performance optimizations

- Use persist() after partitionBy, otherwise it will be recomputed
- sortByKey results in range-partitioned RDD
- groupByKey results in hash-partitioned RDD
- map() creates RDD without partition info (i.e "erases" it)

onTarget
TRAINING SOLUTIONS

Spark Basics - Partitioning example





Spark Basics - Partitioning details

- benefit from partitioning:
  - cogroup, groupWith, join*, groupByKey, combineByKey*, lookup
- will set a partitioner:
  - cogroup, groupWith, join*, groupByKey, combineByKey*, partitionBy, sort
  - mapValues, flatMapValues, filter (only if parent had partition)
- binary partitioners will use the partition of one of their operands, or the first one, if both have partitioners
- PageRank example

Links

Repo with code and other lectures: https://github.com/ichernev/spark-course