

RDD Joins



Objective

Various techniques to optimize key-value RDD joins

Free-for-all techniques

- Spark doesn't optimize our jobs as with DFs/SparkSQL
- it's our job to make them run well
- optimized versions often run better than SparkSQL



Scenario

We are running a nation-wide standard exam (> 1M students)

- every candidate can attempt the exam 5 times
- each attempt is scored 0 - 10
- final score is the max of all attempts

Goal: the number of candidates who passed the exam

- passed = at least one attempt above 9.0

Data:

- candidates: candidate ID (long), candidate name (string)
- exam scores: candidate ID (long), attempt score (double)

To Remember

Spark doesn't optimize our operations the same as it did with SQL

- no query plans
- no column pruning
- no pre-filtering

We're in control:

- pre-filter the data before the join
- pre-aggregate (even partial) results before the join
- co-partition: assign partitioners to RDDs so we can avoid shuffles
- combine all the above!

Spark rocks

