Partitioners

Objective

Customize partitioning logic on RDDs

Be aware of partitioning schemes used with DFs

Partitioners

Decide which record stays on which partition (key-value RDDs only)

- hash partitioning = same hash, same partition
- range partitioning = same range, same partition
- custom partitioning = you decide where each key stays, for custom computations

Partitioning has advantages and does not incur shuffles

- hash partitioning for joins and by-key functions
- range partitioning for sorts

DFs cannot control partitioning logic, but follow rules

- sort/orderBy => RangePartitioning
- aggregation by key => HashPartitioning
- join => both DFs obey HashPartitioning
- repartition with a number => RoundRobinPartitioning
- repartition by column => HashPartitioning

Joins Speedup

Make sure the same keys are on the same partition

- RDDs must have the same partitioner
- otherwise, Spark will pick one

Co-partitioning: RDDs share the same partitioner

- no shuffle involved for joins

Colocation: RDD partitions are already loaded in memory

- fastest join possible

Spark rocks