Join Mechanics

Objective

Recap essential join concepts

Understand why joins are (usually) slow and expensive

Learn how partitioning plays a critical role in joins

Joins

Combine the data in multiple DataFrames/RDDs

- rows are combined
- join condition: only the rows passing the condition are kept

Join types

- inner: combine only the rows passing the condition
- left_outer: inner + all rows in the left "table", with nulls in the corresponding fields of the other "table"
- right_outer: same, for the right "table"
- full_outer = same, for both "tables"

More join types (DFs only)

- left_semi = all the rows in the left DF for which there is a row in the right DF passing the condition
- left_anti = all the rows in the left DF for which there is NO row in the right DF passing the condition

* "table" = DataFrame or RDD

Why Are Joins Slow?

If DFs or RDDs don't have a known partitioner, <u>a shuffle is needed</u>

- data transfer overhead
- potential OOMs
- limited parallelism

Co-located RDDs

- have the same partitioner
- reside in the same physical location in memory (on the same executor)
- can be joined <u>without any network transfer</u>

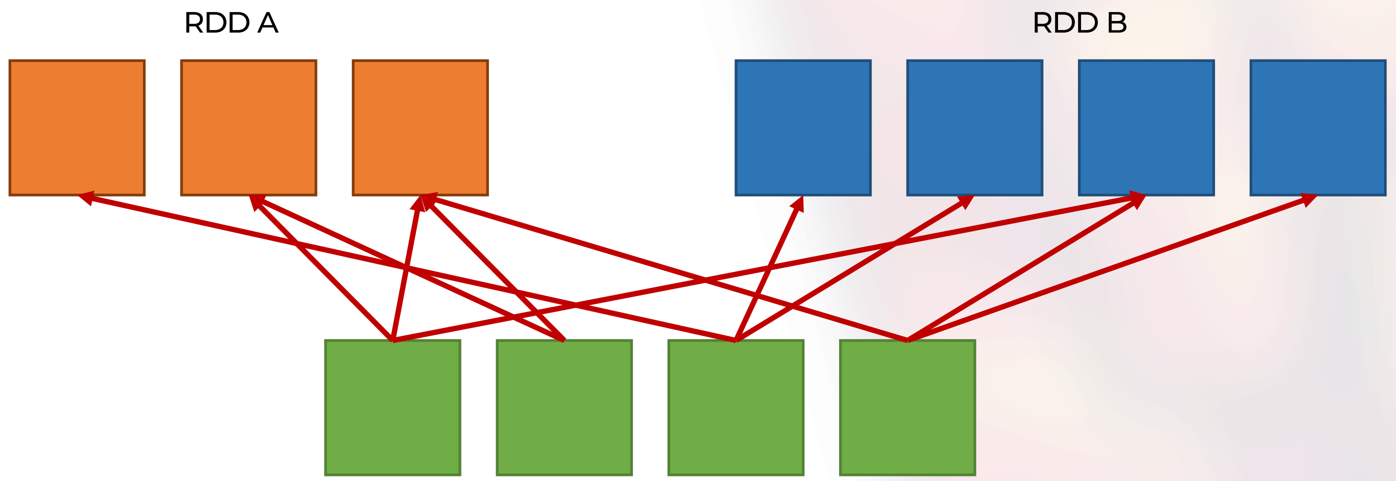
Co-partitioned RDDs

- have the same partitioner
- may be on different executors
- will (in general) be joined <u>with</u> network traffic
    - although much less than without the partitioning information

Shuffled Join

No partitioner is known

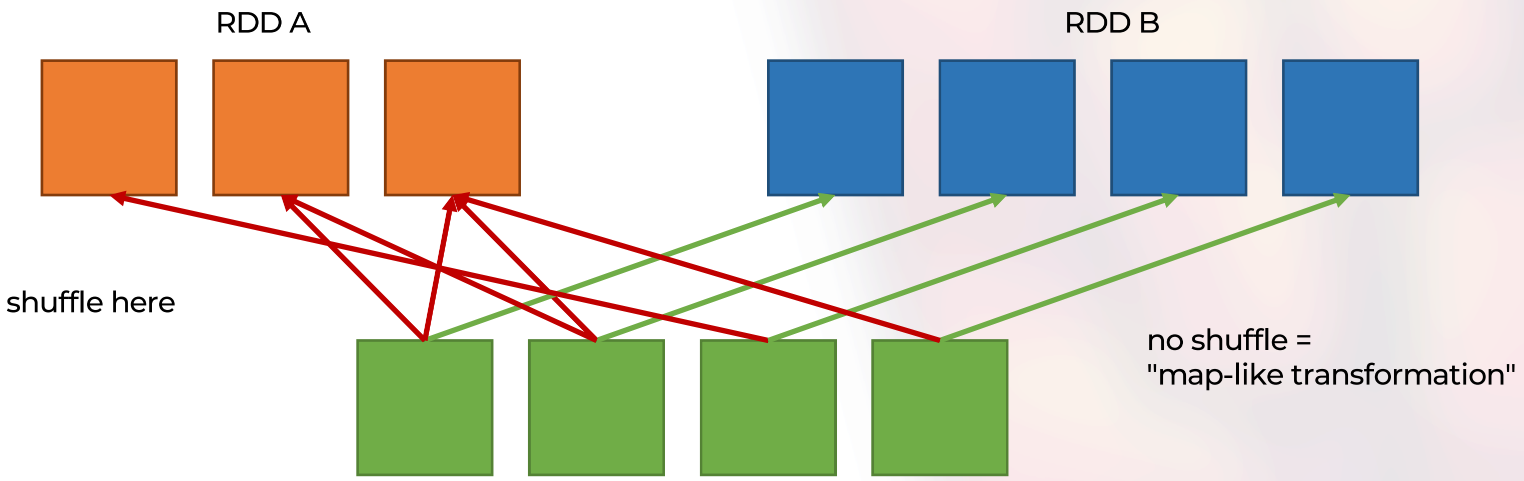
- Rows with the same key must be on the same partition
- Spark needs to shuffle both RDDs
- VERY expensive



Optimized Join

One RDD doesn't have a known partitioner, or partitioners are different

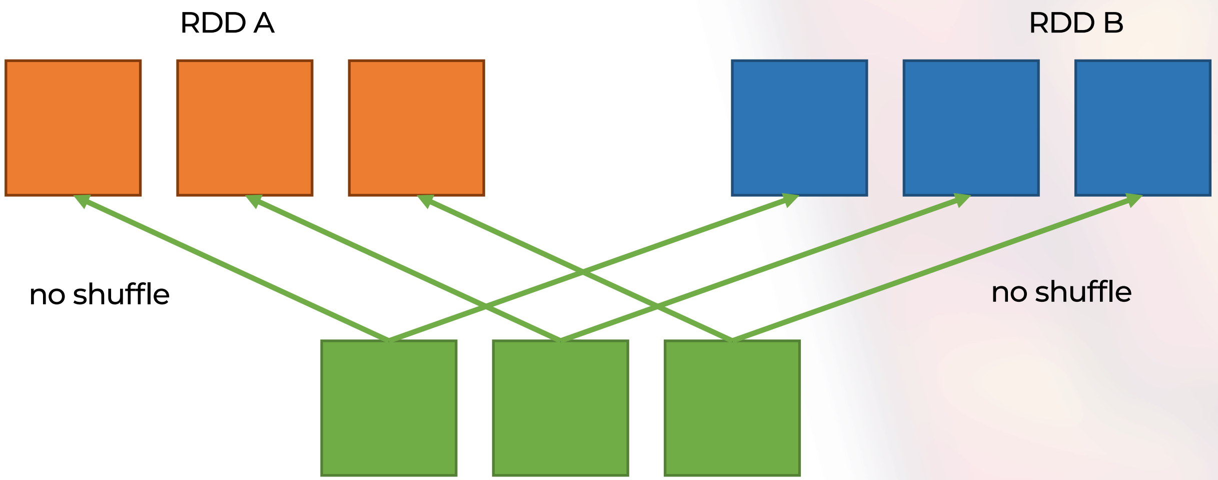
- we can force the other RDD to obey the same partitioner
- one shuffle instead of two



Optimized Join +

Both RDDs have the same partitioner (co-partitioned)

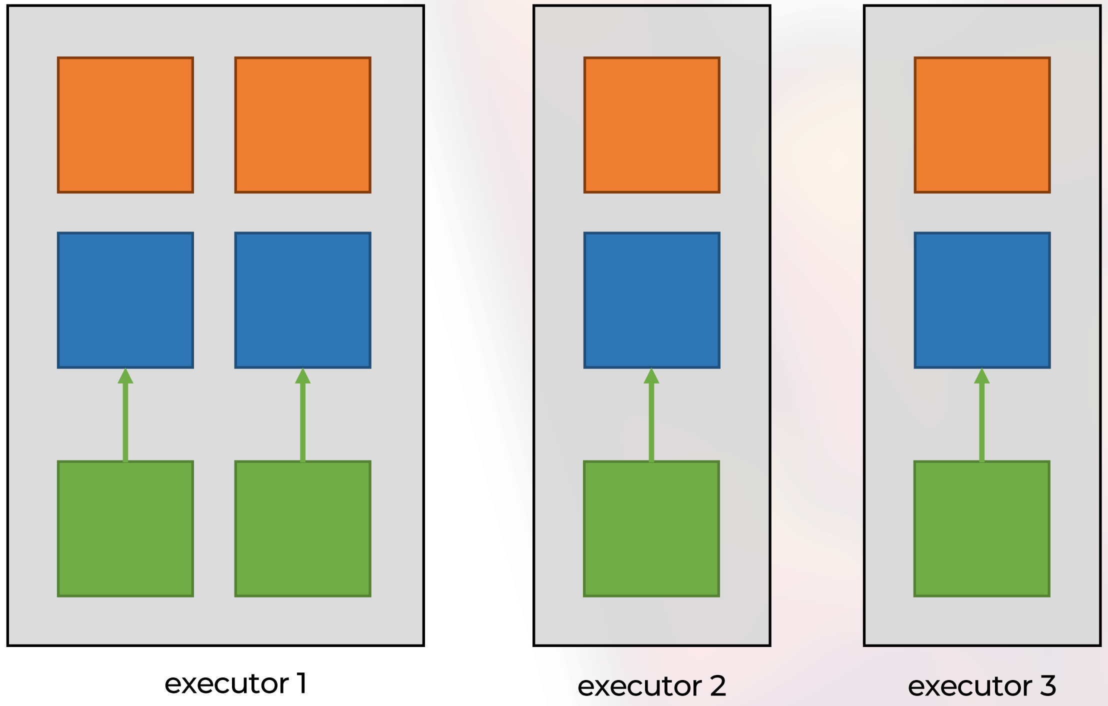
- just fetch the existing partitions and do the join
- no shuffles
- narrow dependency



Optimized Join ++

Same partitioners, partitions loaded in memory (co-located)

- no partition fetching
- no shuffle
- no network transfer
- fastest joins possible



Join Mechanics

Shuffling, colocation & copartitioning apply to RDDs and DFs

This chapter: joins on DFs

Techniques apply to grouping as well

Next chapter: joins on RDDs

Spark rocks