

# Broadcast Joins



# Objective

What to do when joining a large DF with a small DF

Small & powerful technique: broadcasting



# Large-Small Join

```
val rows = sc.parallelize(List(Row(0, "zero"), Row(1, "first"), Row(2, "second"), Row(3, "third")))
val rowsSchema = StructType(Array(StructField("id", IntegerType), StructField("order", StringType)))

val table = spark.range(1, 100000000)
val lookupTable = spark.createDataFrame(rows, rowsSchema)
val joined = table.join(lookupTable, "id")

joined.show()
```

10 seconds?!

# Large-Small Join

```
val rows = sc.parallelize(List(Row(0, "zero"), Row(1, "first"), Row(2, "second"), Row(3, "third")))
val rowsSchema = StructType(Array(StructField("id", IntegerType), StructField("order", StringType)))

val table = spark.range(1, 100000000)
val lookupTable = spark.createDataFrame(rows, rowsSchema)
val joined = table.join(lookupTable, "id")

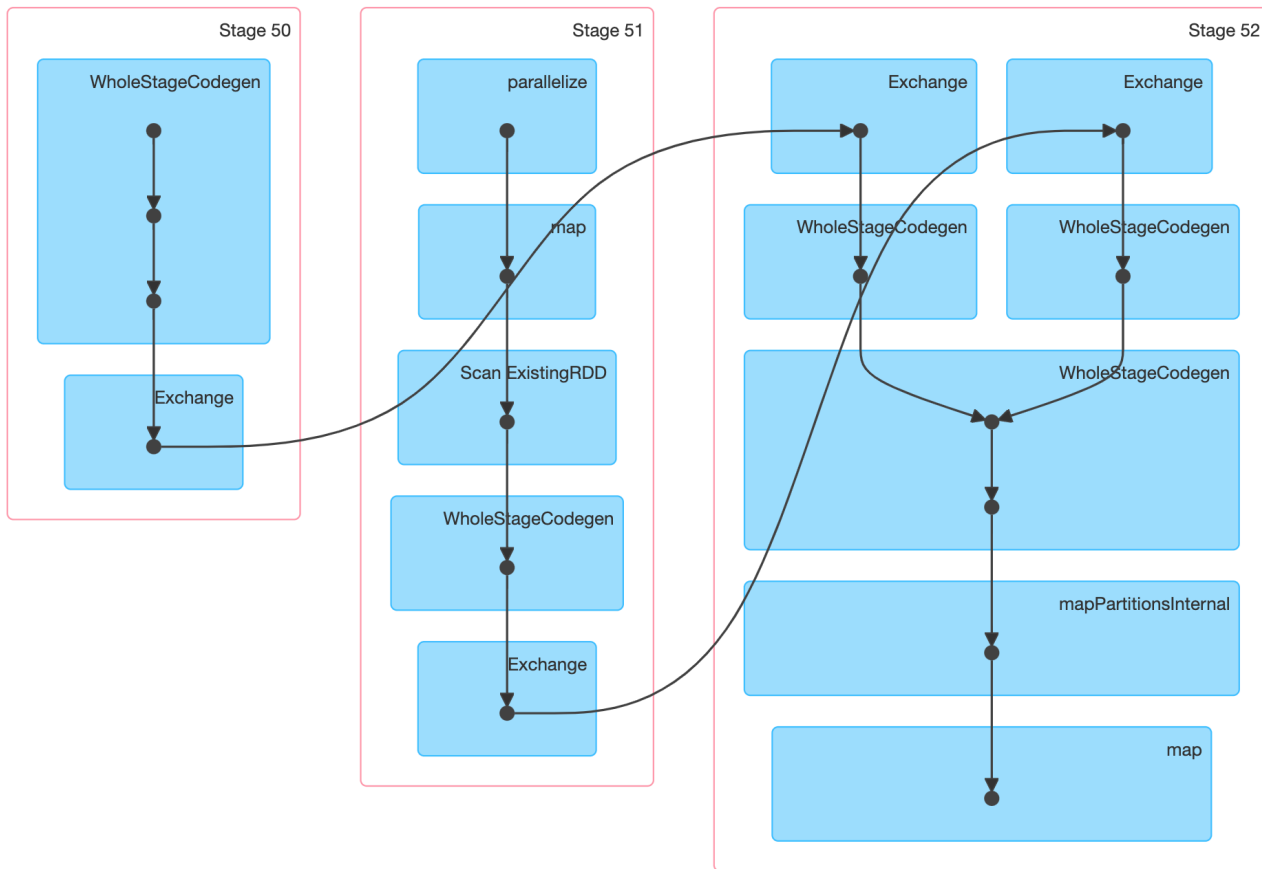
joined.show()
```

```
joined.explain()
```

```
scala> joined.explain
== Physical Plan ==
*(5) Project [id#259L, order#264]
+- *(5) SortMergeJoin [id#259L], [cast(id#263 as bigint)], Inner
   :- *(2) Sort [id#259L ASC NULLS FIRST], false, 0
      : +- Exchange hashpartitioning(id#259L, 200)
      :    +- *(1) Range (1, 100000000, step=1, splits=6)
   +- *(4) Sort [cast(id#263 as bigint) ASC NULLS FIRST], false, 0
      +- Exchange hashpartitioning(cast(id#263 as bigint), 200)
         +- *(3) Filter isnotnull(id#263)
            +- Scan ExistingRDD[id#263,order#264]
```

2 shuffles (one on 100M elements!)

# Large-Small Join



# Large-Small Join

Unnecessary shuffles

Tiny partitions for the small table

## ▼ Completed Stages (3)

Stage Id ▾	Description		Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
52	<a href="#">show at &lt;console&gt;:37</a>	<a href="#">+details</a>	2020/04/11 09:20:42	0.1 s	<div>1/1</div>			2.4 MB	
51	<a href="#">show at &lt;console&gt;:37</a>	<a href="#">+details</a>	2020/04/11 09:20:35	76 ms	<div>6/6</div>				284.0 B
50	<a href="#">show at &lt;console&gt;:37</a>	<a href="#">+details</a>	2020/04/11 09:20:35	7 s	<div>6/6</div>				479.8 MB

Multiple jobs with skipped stages

Stage Id ▾	Description		Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
55	<a href="#">show at &lt;console&gt;:37</a>	<a href="#">+details</a>	2020/04/11 09:20:42	0.3 s	<div>4/4</div>			9.6 MB	

Stage Id ▾	Description		Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
58	<a href="#">show at &lt;console&gt;:37</a>	<a href="#">+details</a>	2020/04/11 09:20:42	3 s	<div>20/20</div>			48.0 MB	

Stage Id ▾	Description		Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
61	<a href="#">show at &lt;console&gt;:37</a>	<a href="#">+details</a>	2020/04/11 09:20:46	3 s	<div>100/100</div>			239.9 MB	

Stage Id ▾	Description		Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
64	<a href="#">show at &lt;console&gt;:37</a>	<a href="#">+details</a>	2020/04/11 09:20:49	2 s	<div>75/75</div>			179.9 MB	

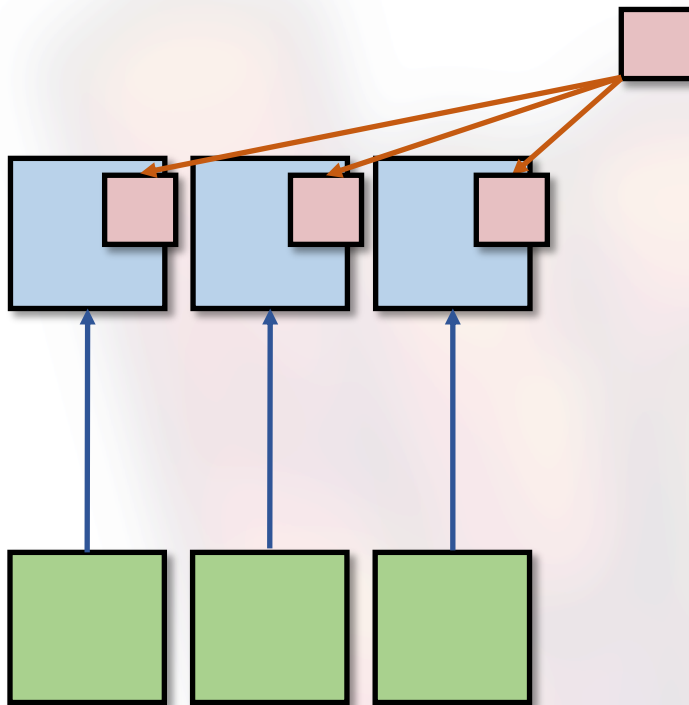
# Large-Small Join

Solution: broadcast join

Smaller DF/RDD is sent to all executors

All joins are done in memory

Tiny overhead



# Broadcast Join

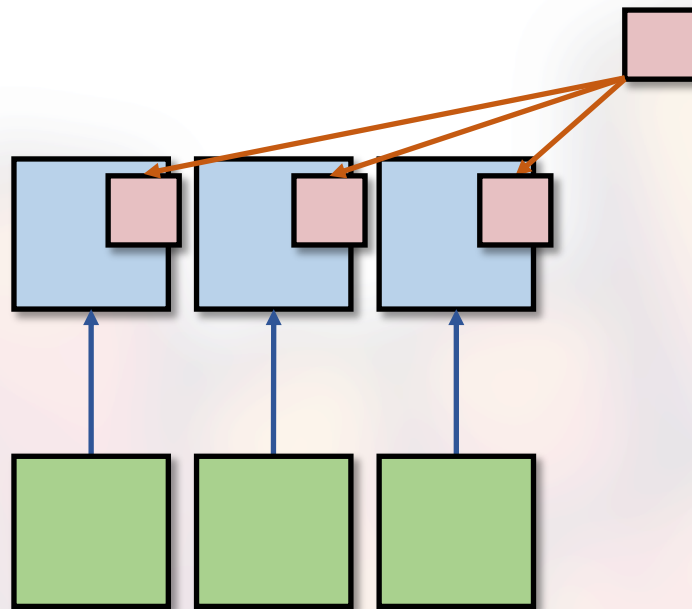
```
val rows = sc.parallelize(...)
val rowsSchema = StructType(...)

val table = spark.range(1, 100000000)
val lookupTable = spark.createDataFrame(rows, rowsSchema)
val joined = table.join(lookupTable, "id")

joined.show()
```

```
joined.explain()
```

```
== Physical Plan ==
*(2) Project [id#294L, order#299]
+- *(2) BroadcastHashJoin [id#294L], [cast(id#298 as bigint)], Inner, BuildRight
   :- *(2) Range (1, 100000000, step=1, splits=6)
   +- BroadcastExchange HashedRelationBroadcastMode(List(cast(input[0, int, false] as bigint))) ← not a shuffle!
      +- *(1) Filter isNotNull(id#298)
         +- Scan ExistingRDD[id#298,order#299]
```





# Broadcast Join

```
val rows = sc.parallelize(...)
val rowsSchema = StructType(...)

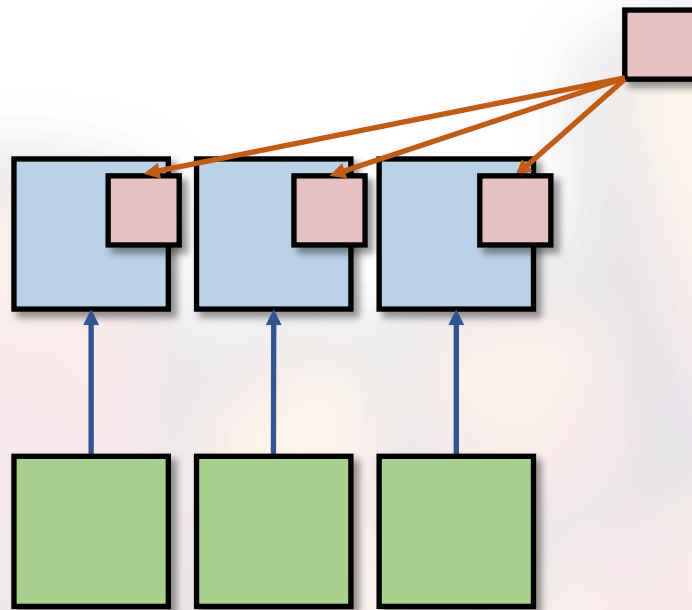
val table = spark.range(1, 100000000)
val lookupTable = spark.createDataFrame(rows, rowsSchema)
val joined = table.join(lookupTable, "id")

joined.show()
```

100ms

100x perf!

This ratio is also seen in prod



# Broadcast Join

Scenario: join with a lookup table

Share the smaller DF/RDD across *all* executors

- tiny overhead
- all other operations done in memory

## Pros

- shuffles avoided
- insane speed

## Risks

- not enough driver memory
- if smaller DF is quite big – large overhead
- if smaller DF is quite big – OOMing executors

Broadcasting can be done automatically by Spark

- finds one DF smaller than a threshold

```
spark.sql.autoBroadcastJoinThreshold = 10485760
```

size in bytes

set to -1 to disable

**Spark rocks**

