

Summary	1
Introduction of the two solutions	1
Performance comparison	2
Conclusion	2
Implementation Details	3
Environment	3
Scripts	3
Experiment Details	3
1. Original Query	3
1.1 With BroadcastJoin used (by default)	3
1.2. original query with BroadcastJoin distabled	5
2. Solution 1 – rewrite source data and set sjuffle partition numbe	7
2.1 Performance with ' Broadcast join' used by default	8
2.2 Solution 1 with Broadcast disabled	10
3. Solution 2 – using bucketing	11
3.1 With 'broadcast join' enable by default	12
3. 2. With 'broadcast join' disabled:	14

Summary

Introduction of the two solutions

To improve the performance, we have tried following two solutions:

1. Solution 1 – reducing shuffle partition number
 - a) Rewrite the source data to 4 parquet files using bucketing
 - b) set “spark.sql.shuffle.partitions” to 4 and driver number to 4

As a result, there is reduced shuffle/write. Also we gain good performance when broadcast join is disable

2. Solution 2 – using bucketing
 - a) Rewrite the source data to 4 parquet files using bucketing
 - b) Read the data to DataFrame as tables
 - c) set “spark.sql.shuffle.partitions” to 4 and driver number to 4

As a result, there is shuffle read/write in the execution plan. Also we gain good performance when broadcast join is disable

Two solutions are similar. Maybe the only difference is that, solution 2 reads the data as a table and solution reads the data not as table. But the results are quite different. For solution 2 , 'bucketing' is achieved, which means, there is no shuffle in the execution plan . However strangely, the performance of solution 2 is slightly worse than solution 1

Performance comparison

As these two tables are both small, the Broadcast join is automatically used for the query by Spark. So we also do experiments on the scenarios that Broadcast join can not be used. This will happen when the two tables are large, say, bigger than 10Mb (the default value "spark.sql.autoBroadcastJoinThreshold").

	With broadcast Join	With broadcast Join Disabled
Original query	3.45 3.17	19.48 18.13
Solution 1	2.17 2.15	2.30 2.22
Solution 2	3.08 3.08	3.23 3.27

The experiments also show:

- For solution 1, there is reduced shuffle;
- For solution 2 . there is no shuffle

Conclusion

The experiment results have confirmed that:

- Our solutions have better performance than the original query.
- More importantly, our solutions are stable – even when broadcast join can not be used, they still gain good performance

Implementation Details

1. Environment

We set up Spark on a local machine (Windows 10) and ran the experiments on it.

2. Scripts

Here are the scripts we used. Note:

1. We slightly changed the original script by adding timing measurement logic
2. For user cases where 'broadcast join' is disabled, we basically using the same script and just adding one line:

```
spark.conf.set("spark.sql.autoBroadcastJoinThreshold",-1)
```

Original query	p20_optim_10.py
Solution 1	p20_write1.py: re-write the source data to 4 files p20_optim_21.py: improved solutio
Solution 2	p20_write2.py: re-write the source data to 4 files p20_optim_31.py: improved solution
Other file	Ut_spr.py: we are using a decorator method 'timer()' from this module to measure the time.

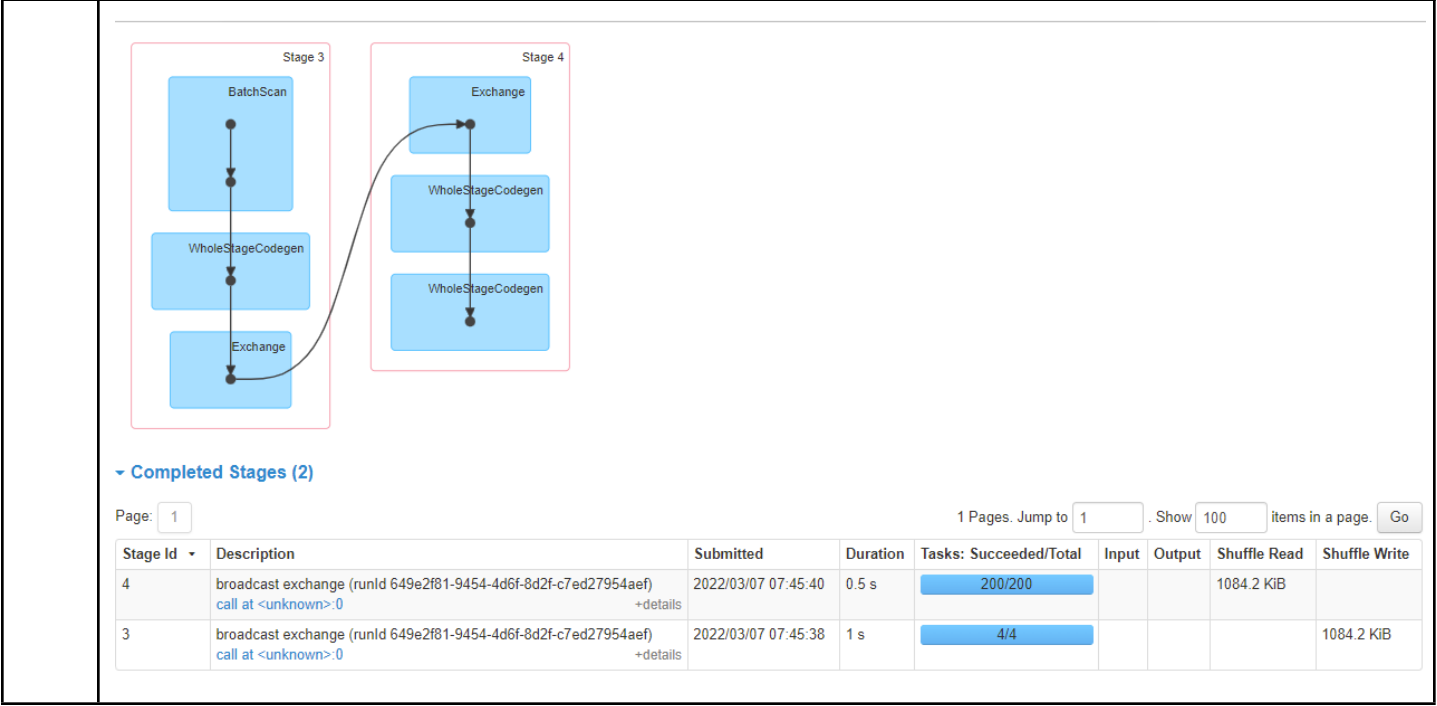
Experiment Details

1. Original Query

1.1 With BroadcastJoin used (by default)

Note	<p>It is noticed:</p> <ol style="list-style-type: none"> 1. BroadcastJoin has been used as the two dataset are small (less than 10Mb) and 'spark.sql.autoBroadcastJoinThreshold' is default to 10M. So the performance is not bad 2. 200 partitions are used for shuffle
Run time	<p>Run 1: 3.45 Run 2: 3.17</p> <p>Run 1: 3.45 sec</p> <pre> 156025 2015-01-01 04:32:... Deriving the Cano... 1 1 156026 2015-01-01 04:49:... Does Bell's inequ... 1 3 +-----+-----+-----+-----+-----+ only showing top 20 rows Elapsed:3.457381 </pre> <p>Run 2: 3.17 sec</p> <pre> >>> with ut1.timer(): ... resultDF.orderBy('question_id', 'month').show(3) ... +-----+-----+-----+-----+-----+ question_id creation_date title month cnt +-----+-----+-----+-----+-----+ 155989 2014-12-31 17:59:... Frost bubble form... 2 1 155989 2014-12-31 17:59:... Frost bubble form... 12 1 155990 2014-12-31 18:51:... The abstract spac... 1 1 +-----+-----+-----+-----+-----+ only showing top 3 rows Elapsed:3.174859 </pre>
Expl ain plan	<pre> >>> resultDF.orderBy('question_id', 'month').explain() == Physical Plan == *(4) Sort [question_id#44L ASC NULLS FIRST, month#60 ASC NULLS FIRST], true, 0 +- Exchange rangepartitioning(question_id#44L ASC NULLS FIRST, month#60 ASC NULLS FIRST, 200), true, [id=#222] +- *(3) Project [question_id#44L, creation_date#46, title#47, month#60, cnt#76L] +- *(3) BroadcastHashJoin [question_id#44L], [question_id#32L], Inner, BuildRight :- *(3) Project [question_id#44L, creation_date#46, title#47] : +- *(3) Filter isnotnull(question_id#44L) : +- *(3) ColumnarToRow : +- BatchScan[question_id#44L, creation_date#46, title#47] ParquetScan Location: InMemoryFileIndex[file:/C:/demo/c17-spark-ex1/01-HelloSpark-and-other-app/data/questions], ReadSchema: struct<question_id:bigint,creation_date:timestamp,title:string>, PushedFilters: [IsNotNull(question_id)] +- BroadcastExchange HashedRelationBroadcastMode(List(input[0, bigint, true])), [id=#217] +- *(2) HashAggregate(keys=[question_id#32L, month#60], functions=[count(1)]) </pre>

	<pre> +- Exchange hashpartitioning(question_id#32L, month#60, 200), true, [id=#213] +- *(1) HashAggregate(keys=[question_id#32L, month#60], functions=[partial_count(1)]) +- *(1) Project [question_id#32L, month(cast(creation_date#34 as date)) AS month#60] +- *(1) Filter isnotnull(question_id#32L) +- *(1) ColumnarToRow +- BatchScan[question_id#32L, creation_date#34] ParquetScan Location: InMemoryFileIndex[file:/C:/demo/c17-spark-ex1/01-HelloSpark-and-other-app/data/answers], ReadSchema: struct<question_id:bigint,creation_date:timestamp>, PushedFilters: [IsNotNull(question_id)] </pre>
--	--



1.2. original query with BroadcastJoin distabled

We added `"spark.conf.set("spark.sql.autoBroadcastJoinThreshold",-1)"` to the script so the Broadcast Join was disabled

Observation	<ul style="list-style-type: none"> • 200 partitions took part in shuffle process • There are shuffles. Shuffle read/write data size is large (each 9.5 M in total) • Worst performance – running time is 6 times of when broadcast is used
-------------	---

Run time:
19.48 sec
18.13 sec.

Run 1

```
>>> spark.conf.set("spark.sql.autoBroadcastJoinThreshold",-1)
>>> with ut1.timer():
...     resultDF.orderBy('question_id', 'month').show(3)
...
+-----+-----+-----+-----+-----+
|question_id|creation_date|title|month|cnt|
+-----+-----+-----+-----+-----+
|155989|2014-12-31 17:59:...|Frost bubble form...|2|1|
|155989|2014-12-31 17:59:...|Frost bubble form...|12|1|
|155990|2014-12-31 18:51:...|The abstract spac...|1|1|
+-----+-----+-----+-----+-----+
only showing top 3 rows

Elapsed:19.479335
```

Run 2:

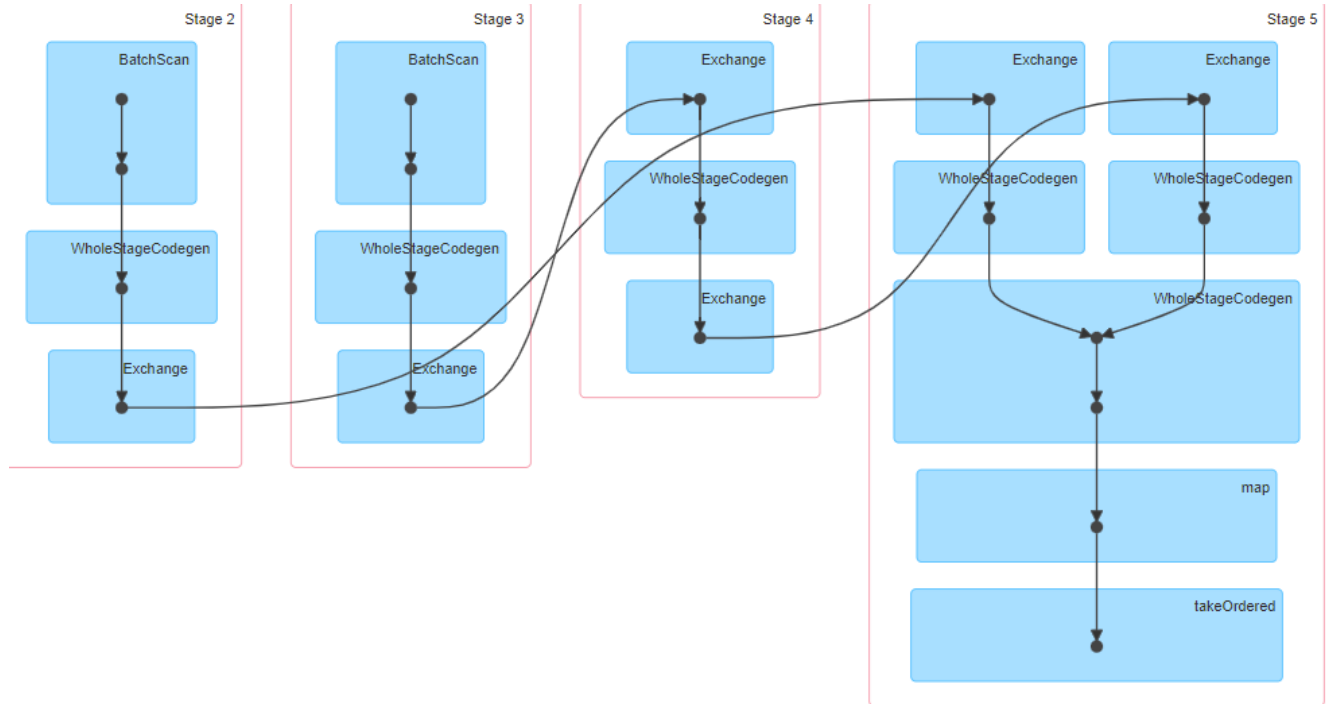
```
only showing top 3 rows

Elapsed:18.129344
```

Plan

```
>>> resultDF.orderBy('question_id', 'month').explain()
== Physical Plan ==
*(7) Sort [question_id#44L ASC NULLS FIRST, month#60 ASC NULLS FIRST], true, 0
+- Exchange rangepartitioning(question_id#44L ASC NULLS FIRST, month#60 ASC NULLS FIRST, 200),
true, [id=#407]
  +- *(6) Project [question_id#44L, creation_date#46, title#47, month#60, cnt#76L]
    +- *(6) SortMergeJoin [question_id#44L], [question_id#32L], Inner
      :- *(2) Sort [question_id#44L ASC NULLS FIRST], false, 0
      : +- Exchange hashpartitioning(question_id#44L, 200), true, [id=#384]
      :   +- *(1) Project [question_id#44L, creation_date#46, title#47]
      :     +- *(1) Filter isnotnull(question_id#44L)
      :     +- *(1) ColumnarToRow
      :     +- BatchScan[question_id#44L, creation_date#46, title#47] ParquetScan Location:
InMemoryFileIndex[file:/C:/demo/c17-spark-ex1/01-HelloSpark-and-other-app/data/questions], ReadSchema:
struct<question_id:bigint,creation_date:timestamp,title:string>, PushedFilters: [IsNotNull(question_id)]
    +- *(5) Sort [question_id#32L ASC NULLS FIRST], false, 0
    +- Exchange hashpartitioning(question_id#32L, 200), true, [id=#399]
      +- *(4) HashAggregate(keys=[question_id#32L, month#60], functions=[count(1)])
      +- Exchange hashpartitioning(question_id#32L, month#60, 200), true, [id=#395]
        +- *(3) HashAggregate(keys=[question_id#32L, month#60], functions=[partial_count(1)])
        +- *(3) Project [question_id#32L, month(cast(creation_date#34 as date)) AS month#60]
        +- *(3) Filter isnotnull(question_id#32L)
        +- *(3) ColumnarToRow
        +- BatchScan[question_id#32L, creation_date#34] ParquetScan Location:
InMemoryFileIndex[file:/C:/demo/c17-spark-ex1/01-HelloSpark-and-other-app/data/answers], ReadSchema:
```

```
struct<question_id:bigint,creation_date:timestamp>, PushedFilters: [IsNotNull(question_id)]
```



1 Pages. Jump to . Show items in a page.

Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
showString at <unknown>:0 +details	2022/03/07 11:50:43	3 s	<div>200/200</div>			8.4 MiB	
showString at <unknown>:0 +details	2022/03/07 11:50:30	12 s	<div>200/200</div>			1084.2 KiB	2.9 MiB
showString at <unknown>:0 +details	2022/03/07 11:50:29	1 s	<div>4/4</div>				1084.2 KiB
showString at <unknown>:0 +details	2022/03/07 11:50:29	1 s	<div>4/4</div>				5.4 MiB

2. Solution 1 – rewrite source data and set sjuffle partition numbe

We made following improvements:

1) Rewrite the source data to 4 parquet files using bucketing

Questions	<pre>questionsDF\ .coalesce(1)\ .write\ .mode('overwrite')\ .bucketBy(4, 'question_id')\ .sortBy("question_id")\</pre>
-----------	--

	<pre>.option("path","data/questions_n")\ .saveAsTable('questions',format='parquet')</pre>
Answers	<pre>answersDF\ .coalesce(1)\ .write\ .mode('overwrite')\ .bucketBy(4, 'question_id')\ .sortBy("question_id")\ .option("path","data/answers_n")\ .saveAsTable('answers',format='parquet')</pre>

- 2) Set `spark.sql.shuffle.partitions` to 4 and set number of drivers to 4 (`master("local[4])`)

2.1 Performance with 'Broadcast join' used by default

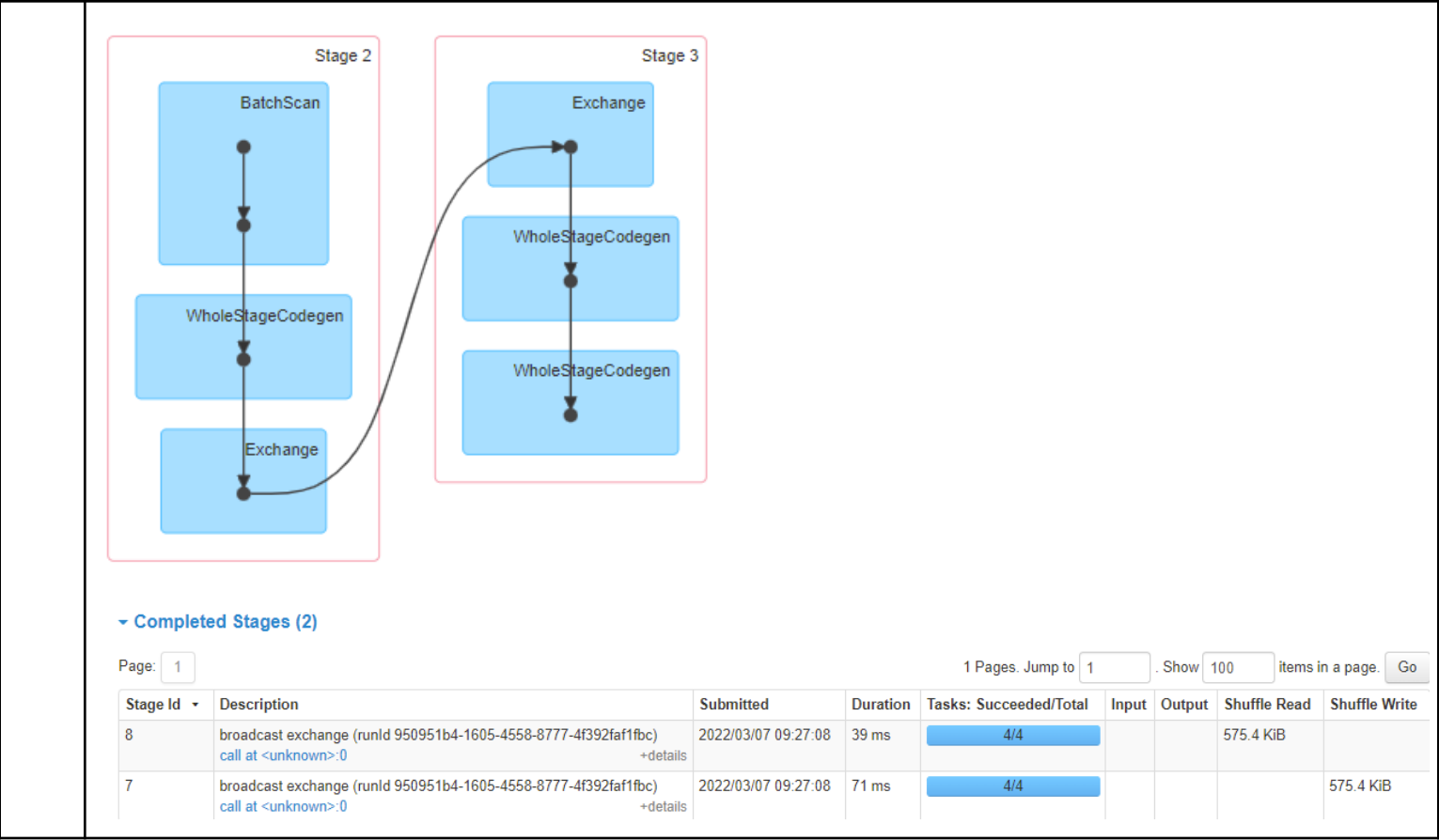
Note	<ul style="list-style-type: none"> There are shuffle read/write, but shuffle data size is reduced((575k) 4 partitions are used for shuffle Gained best performance (shortest time of running query)
	<p>Run 1: 2.17 sec</p> <pre>>>> with ut1.timer(): ... resultDF.orderBy('question_id', 'month').show(3) ... +-----+-----+-----+-----+ question_id creation_date title month cnt +-----+-----+-----+-----+ 155989 2014-12-31 17:59:... Frost bubble form... 2 1 155989 2014-12-31 17:59:... Frost bubble form... 12 1 155990 2014-12-31 18:51:... The abstract spac... 1 1 +-----+-----+-----+-----+ only showing top 3 rows Elapsed:2.165312</pre> <p>Run 2: 2.15 sec</p>


```
>>>
>>> with ut1.timer():
...     resultDF.orderBy('question_id', 'month').show(3)
...
+-----+-----+-----+-----+-----+
|question_id|creation_date|title|month|cnt|
+-----+-----+-----+-----+
|155989|2014-12-31 17:59:...|Frost bubble form...|2|1|
|155989|2014-12-31 17:59:...|Frost bubble form...|12|1|
|155990|2014-12-31 18:51:...|The abstract spac...|1|1|
+-----+-----+-----+-----+
only showing top 3 rows

Elapsed:2.146522
```

Execution plan:

```
>>> resultDF.orderBy('question_id', 'month').explain()
== Physical Plan ==
*(4) Sort [question_id#12L ASC NULLS FIRST, month#28 ASC NULLS FIRST], true, 0
+- Exchange rangepartitioning(question_id#12L ASC NULLS FIRST, month#28 ASC NULLS FIRST, 4),
true, [id=#149]
  +- *(3) Project [question_id#12L, creation_date#14, title#15, month#28, cnt#44L]
    +- *(3) BroadcastHashJoin [question_id#12L], [question_id#0L], Inner, BuildRight
      :- *(3) Project [question_id#12L, creation_date#14, title#15]
        : +- *(3) Filter isnotnull(question_id#12L)
          :   +- *(3) ColumnarToRow
            :     +- BatchScan[question_id#12L, creation_date#14, title#15] ParquetScan Location:
InMemoryFileIndex[file:/C:/demo/c17-spark-ex1/01-HelloSpark-and-other-app/data/questions_n],
ReadSchema: struct<question_id:bigint,creation_date:timestamp,title:string>, PushedFilters:
[IsNotNull(question_id)]
      +- BroadcastExchange HashedRelationBroadcastMode(List(input[0, bigint, true])), [id=#144]
    +- *(2) HashAggregate(keys=[question_id#0L, month#28], functions=[count(1)])
      +- Exchange hashpartitioning(question_id#0L, month#28, 4), true, [id=#140]
        +- *(1) HashAggregate(keys=[question_id#0L, month#28], functions=[partial_count(1)])
          +- *(1) Project [question_id#0L, month(cast(creation_date#2 as date)) AS month#28]
            +- *(1) Filter isnotnull(question_id#0L)
              +- *(1) ColumnarToRow
                +- BatchScan[question_id#0L, creation_date#2] ParquetScan Location:
InMemoryFileIndex[file:/C:/demo/c17-spark-ex1/01-HelloSpark-and-other-app/data/answers_n],
ReadSchema: struct<question_id:bigint,creation_date:timestamp>, PushedFilters: [IsNotNull(question_id)]
```



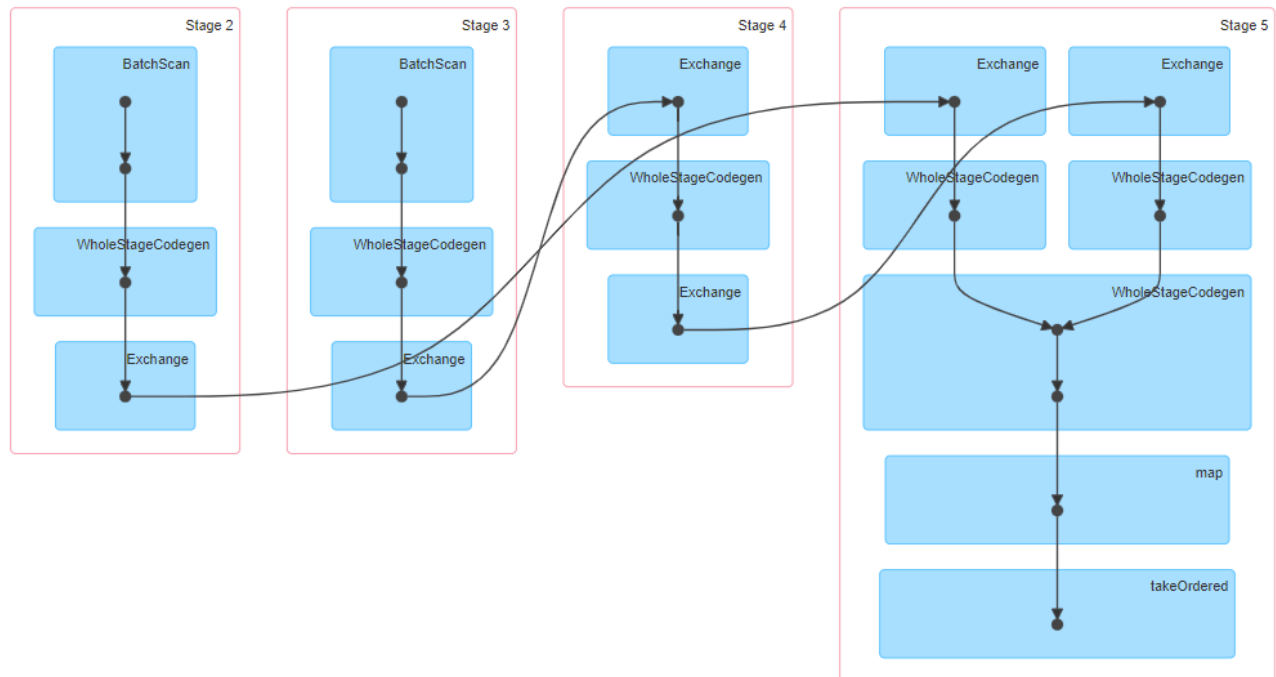
2.2 Solution 1 with Broadcast disabled

Observations	<ul style="list-style-type: none">The execution plan is very similar to the original query when broadcast join is disable4 partitions are used and shuffle data size is reducedWith our improvement, the performance is comparable to when broadcast join is used, which means performance is stable
	<div>Run time: 2.30 sec, 2.22 sec</div> <div>Run 1: <div>only showing top 3 rows</div><div>Elapsed:2.297134</div><div>>> resultDF2.explain()</div></div> <div>Run 2: <div>Elapsed:2.222535</div></div>

```

>>> resultDF2.explain()
== Physical Plan ==
*(7) Sort [question_id#12L ASC NULLS FIRST, month#28 ASC NULLS FIRST], true, 0
+- Exchange rangepartitioning(question_id#12L ASC NULLS FIRST, month#28 ASC NULLS FIRST, 4),
true, [id=#181]
  +- *(6) Project [question_id#12L, creation_date#14, title#15, month#28, cnt#44L]
    +- *(6) SortMergeJoin [question_id#12L], [question_id#0L], Inner
      :- *(2) Sort [question_id#12L ASC NULLS FIRST], false, 0
        : +- Exchange hashpartitioning(question_id#12L, 4), true, [id=#158]
        :   +- *(1) Project [question_id#12L, creation_date#14, title#15]
        :     +- *(1) Filter isnotnull(question_id#12L)
        :     +- *(1) ColumnarToRow
        :     +- BatchScan[question_id#12L, creation_date#14, title#15] ParquetScan Location:
InMemoryFileIndex[file:/C:/demo/c17-spark-ex1/01-HelloSpark-and-other-app/data/questions_n],
ReadSchema: struct<question_id:bigint,creation_date:timestamp,title:string>, PushedFilters:
[IsNotNull(question_id)]
      +- *(5) Sort [question_id#0L ASC NULLS FIRST], false, 0
        +- Exchange hashpartitioning(question_id#0L, 4), true, [id=#173]
        +- *(4) HashAggregate(keys=[question_id#0L, month#28], functions=[count(1)])
          +- Exchange hashpartitioning(question_id#0L, month#28, 4), true, [id=#169]
            +- *(3) HashAggregate(keys=[question_id#0L, month#28], functions=[partial_count(1)])
              +- *(3) Project [question_id#0L, month(cast(creation_date#2 as date)) AS month#28]
                +- *(3) Filter isnotnull(question_id#0L)
                +- *(3) ColumnarToRow
                +- BatchScan[question_id#0L, creation_date#2] ParquetScan Location:
InMemoryFileIndex[file:/C:/demo/c17-spark-ex1/01-HelloSpark-and-other-app/data/answers_n],
ReadSchema: struct<question_id:bigint,creation_date:timestamp>, PushedFilters: [IsNotNull(question_id)]

```



<div> <div>Completed Stages (4)</div> <div> <div>Page: 1</div> <div>1 Pages. Jump to 1. Show 100 items in a page. Go</div> </div> </div>									
Stage Id	Description		Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
5	showString at <unknown>:0	+details	2022/03/07 11:18:22	0.2 s	4/4			5.4 MiB	
4	showString at <unknown>:0	+details	2022/03/07 11:18:22	0.1 s	4/4			575.4 KiB	575.4 KiB
3	showString at <unknown>:0	+details	2022/03/07 11:18:21	0.6 s	4/4				575.4 KiB
2	showString at <unknown>:0	+details	2022/03/07 11:18:21	0.6 s	4/4				4.8 MiB

3. Solution 2 – using bucketing

With this solution, we are using ‘Bucketing’ technology :

- 1) Rewrite the source data to 4 parquet files using bucketing by specifying following options, which is similar to Solution 1


```
bucketBy(4, 'question_id')\
.sortBy("question_id")\
.saveAsTable
```
- 2) Also set “spark.sql.shuffle.partitions” to 4 and driver number to 4, similar to solution 1
- 3) Read the data to DataFrame as tables

As a result, there is no shuffle in the execution plan

3.1 With ‘broadcast join’ enable by default

obser serva tions	<ul style="list-style-type: none"> There is broadcast join but no shuffle read/write (from the execution plan) The performance is slightly better than original query but is worse than solution 1
	<div>Run 1: 3.08 sec</div> <pre>>>> with ut1.timer(): ... resultDF.orderBy('question_id', 'month').show(3) ... +-----+-----+-----+-----+-----+ question_id creation_date title month cnt +-----+-----+-----+-----+-----+ 155989 2014-12-31 17:59:... Frost bubble form... 12 1 155989 2014-12-31 17:59:... Frost bubble form... 12 1 155990 2014-12-31 18:51:... The abstract spac... 1 1 +-----+-----+-----+-----+-----+ only showing top 3 rows Elapsed:3.078451</pre> <div>Run 2: 3.08 sec</div>

```
>>> with ut1.timer():
..     resultDF.orderBy('question_id', 'month').show(3)
..
```

question_id	creation_date	title	month	cnt
155989	2014-12-31 17:59:...	Frost bubble form...	2	1
155989	2014-12-31 17:59:...	Frost bubble form...	12	1
155990	2014-12-31 18:51:...	The abstract spac...	1	1

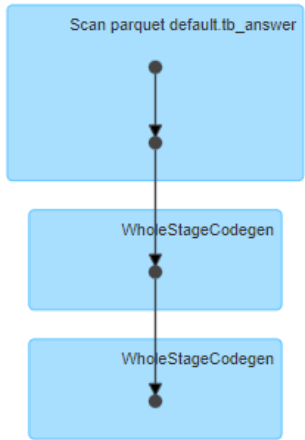
only showing top 3 rows

Elapsed:3.084175

```
>>> resultDF.orderBy('question_id', 'month').explain()
== Physical Plan ==
*(3) Sort [question_id#0L ASC NULLS FIRST, month#28 ASC NULLS FIRST], true, 0
+- Exchange rangepartitioning(question_id#0L ASC NULLS FIRST, month#28 ASC NULLS FIRST, 4),
true, [id=#135]
  +- *(2) Project [question_id#0L, creation_date#2, title#3, month#28, cnt#44L]
    +- *(2) BroadcastHashJoin [question_id#0L], [question_id#16L], Inner, BuildRight
      :- *(2) Project [question_id#0L, creation_date#2, title#3]
      : +- *(2) Filter isnotnull(question_id#0L)
      :   +- *(2) ColumnarToRow
      :     +- FileScan parquet default.tb_question[question_id#0L,creation_date#2,title#3] Batched: true,
DataFilters: [isnotnull(question_id#0L)], Format: Parquet, Location:
InMemoryFileIndex[file:/C:/demo/c17-spark-ex1/01-HelloSpark-and-other-app/data/tb_question],
PartitionFilters: [], PushedFilters: [IsNotNull(question_id)], ReadSchema:
struct<question_id:bigint,creation_date:timestamp,title:string>, SelectedBucketsCount: 4 out of 4
    +- BroadcastExchange HashedRelationBroadcastMode(List(input[0, bigint, true])), [id=#130]
      +- *(1) HashAggregate(keys=[question_id#16L, month#28], functions=[count(1)])
        +- *(1) HashAggregate(keys=[question_id#16L, month#28], functions=[partial_count(1)])
          +- *(1) Project [question_id#16L, month(cast(creation_date#18 as date)) AS month#28]
            +- *(1) Filter isnotnull(question_id#16L)
              +- *(1) ColumnarToRow
                +- FileScan parquet default.tb_answer[question_id#16L,creation_date#18] Batched: true,
DataFilters: [isnotnull(question_id#16L)], Format: Parquet, Location:
InMemoryFileIndex[file:/C:/demo/c17-spark-ex1/01-HelloSpark-and-other-app/data/tb_answer],
PartitionFilters: [], PushedFilters: [IsNotNull(question_id)], ReadSchema:
struct<question_id:bigint,creation_date:timestamp>, SelectedBucketsCount: 4 out of 4
```

Job 0 – read 'question' data; there is broadcastExchange; no shuffle

Stage 0



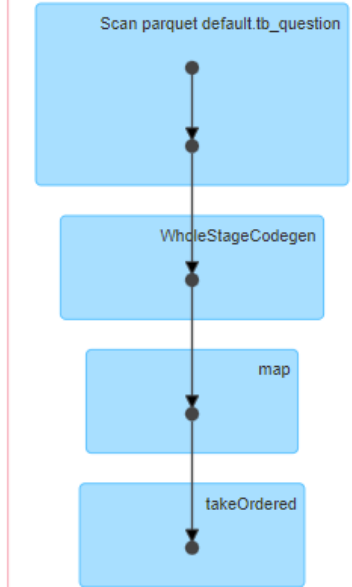
Page: 1

1 Pages. Jump to 1. Show 100 items

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read
0	broadcast exchange (runId b60ff2dc-a301-4cd0-9ea3-e38840733dc9) call at <unknown>:0 +details	2022/03/07 10:48:43	1 s	4/4	1598.8 KiB		

Job 1: read 'answer' data, no shuffle

Stage



Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read
1	showString at <unknown>:0 +details	2022/03/07 10:48:45	0.1 s	4/4	4.6 MiB		

3. 2. With 'broadcast join' disabled:

- There is no shuffle read/write (from the execution plan)
- The performance is slightly slower than using this solution when broadcast join is used, and a lot better than original query when broadcast join is disabled

Running time:

Run 1: 3.23 sec

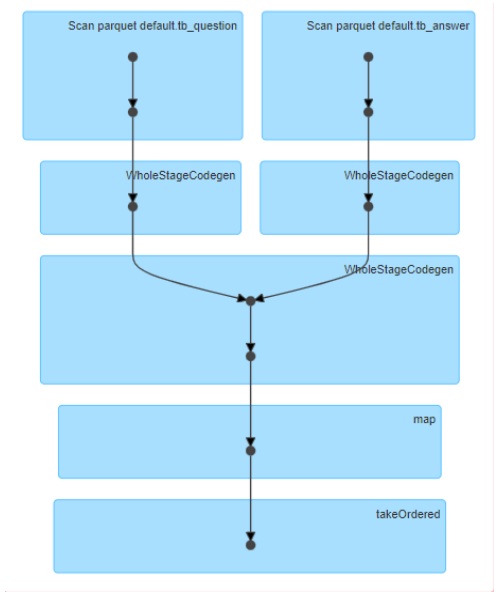
```
----- resultDF2.show -----
>>> with ut1.timer():
...   resultDF.orderBy('question_id', 'month').show(3)
...
+-----+-----+-----+-----+-----+
|question_id|creation_date|title|month|cnt|
+-----+-----+-----+-----+-----+
|155989|2014-12-31 17:59:...|Frost bubble form...|2|1|
|155989|2014-12-31 17:59:...|Frost bubble form...|12|1|
|155990|2014-12-31 18:51:...|The abstract spac...|1|1|
+-----+-----+-----+-----+-----+
only showing top 3 rows
Elapsed:3.226743
```

Run 2: 3.27 sec

```
-----+-----+-----+-----+-----+
only showing top 3 rows
Elapsed:3.274235
```

```
>>> resultDF.orderBy('question_id', 'month').explain()
== Physical Plan ==
*(4) Sort [question_id#0L ASC NULLS FIRST, month#28 ASC NULLS FIRST], true, 0
+- Exchange rangepartitioning(question_id#0L ASC NULLS FIRST, month#28 ASC NULLS FIRST, 4), true,
[id=#147]
   +- *(3) Project [question_id#0L, creation_date#2, title#3, month#28, cnt#44L]
      +- *(3) SortMergeJoin [question_id#0L], [question_id#16L], Inner
         :- *(1) Sort [question_id#0L ASC NULLS FIRST], false, 0
            : +- *(1) Project [question_id#0L, creation_date#2, title#3]
            :    +- *(1) Filter isnotnull(question_id#0L)
            :       +- *(1) ColumnarToRow
            :          +- FileScan parquet default.tb_question[question_id#0L,creation_date#2,title#3] Batched: true,
DataFilters: [isnotnull(question_id#0L)], Format: Parquet, Location:
InMemoryFileIndex[file:/C:/demo/c17-spark-ex1/01-HelloSpark-and-other-app/data/tb_question],
PartitionFilters: [], PushedFilters: [IsNotNull(question_id)], ReadSchema:
struct<question_id:bigint,creation_date:timestamp,title:string>, SelectedBucketsCount: 4 out of 4
         +- *(2) Sort [question_id#16L ASC NULLS FIRST], false, 0
            +- *(2) HashAggregate(keys=[question_id#16L, month#28], functions=[count(1)])
            +- *(2) HashAggregate(keys=[question_id#16L, month#28], functions=[partial_count(1)])
            +- *(2) Project [question_id#16L, month(cast(creation_date#18 as date)) AS month#28]
            +- *(2) Filter isnotnull(question_id#16L)
            +- *(2) ColumnarToRow
               +- FileScan parquet default.tb_answer[question_id#16L,creation_date#18] Batched: true,
DataFilters: [isnotnull(question_id#16L)], Format: Parquet, Location:
InMemoryFileIndex[file:/C:/demo/c17-spark-ex1/01-HelloSpark-and-other-app/data/tb_answer],
PartitionFilters: [], PushedFilters: [IsNotNull(question_id)], ReadSchema:
struct<question_id:bigint,creation_date:timestamp>, SelectedBucketsCount: 4 out of 4
```

It has only one job:



Completed stages (1)

Page: 1 1 Pages. Jump to 1 . Show 100 items in a page. Go

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
0	showString at <unknown>:0 +details	2022/03/07 10:36:19	2 s	4/4	6.2 MiB			