Spark Optimization Mini Project

Original: Counts the number of questions grouped by question_id and month

```
'''
Answers aggregation

Here we : get number of answers per question per month
'''
Run Cell | Run Above | Debug Cell | Go to [9]
#%%
answers_month = answersDF.withColumn('month', month
('creation_date')) \
    .groupBy('question_id', 'month') \
    .agg(count('*') \
    .alias('cnt'))

resultDF = questionsDF.join(answers_month, 'question_id') \
    .select('question_id', 'creation_date', 'title',
    'month', 'cnt')

resultDF.orderBy('question_id', 'month').show()
```

```
+-----------+-------------------+--------------------+-----+---+
|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|
|     155990|2014-12-31 18:51:...|The abstract spac...|   12|  1|
|     155992|2014-12-31 19:44:...|centrifugal force...|   12|  1|
|     155993|2014-12-31 19:56:...|How can I estimat...|    1|  1|
|     155995|2014-12-31 21:16:...|Why should a solu...|    1|  3|
|     155996|2014-12-31 22:06:...|Why do we assume ...|    1|  2|
|     155996|2014-12-31 22:06:...|Why do we assume ...|    2|  1|
|     155996|2014-12-31 22:06:...|Why do we assume ...|   11|  1|
|     155997|2014-12-31 22:26:...|Why do square sha...|    1|  3|
|     155999|2014-12-31 23:01:...|Diagonalizability...|    1|  1|
|     156008|2015-01-01 00:48:...|Capturing a light...|    1|  2|
|     156008|2015-01-01 00:48:...|Capturing a light...|   11|  1|
|     156016|2015-01-01 02:31:...|The interference ...|    1|  1|
|     156020|2015-01-01 03:19:...|What is going on ...|    1|  1|
|     156021|2015-01-01 03:21:...|How to calculate ...|    2|  1|
|     156022|2015-01-01 03:55:...|Advice on Major S...|    1|  1|
|     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
```

From the Spark UI at localhost:4040 we can examine the individual jobs and see runtime duration. Total time of all jobs is 4.4s.

Spark Jobs (?)

User: Margaret
Total Uptime: 20 s
Scheduling Mode: FIFO
Completed Jobs: 5

▸ Event Timeline

▾ **Completed Jobs (5)**

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

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
4	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2024/10/25 17:29:59	0.8 s	1/1 (1 skipped)	1/1 (4 skipped)
3	$anonfun$withThreadLocalCaptured$1 at FutureTask.java:264 $anonfun$withThreadLocalCaptured$1 at FutureTask.java:264	2024/10/25 17:29:57	0.7 s	1/1	4/4
2	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2024/10/25 17:29:55	2 s	1/1	4/4
1	load at NativeMethodAccessorImpl.java:0 load at NativeMethodAccessorImpl.java:0	2024/10/25 17:29:52	0.1 s	1/1	1/1
0	load at NativeMethodAccessorImpl.java:0 load at NativeMethodAccessorImpl.java:0	2024/10/25 17:29:51	0.8 s	1/1	1/1

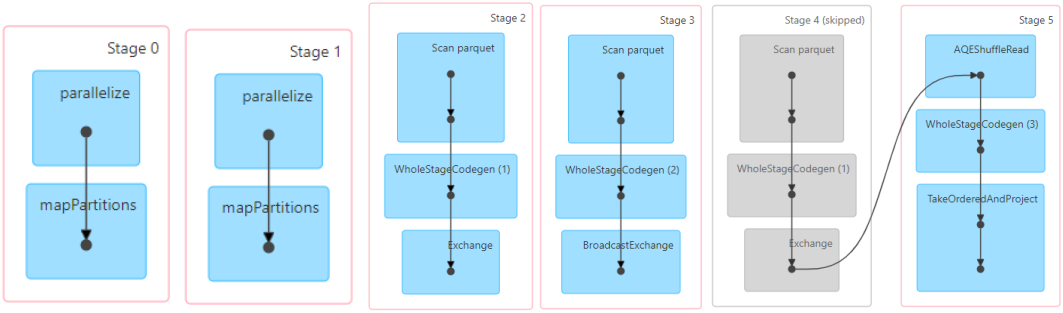
Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

▾ **Completed Stages (5)**

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
5	showString at NativeMethodAccessorImpl.java:0	+details	2024/10/25 17:29:59	0.7 s	1/1			1084.0 KiB	
3	$anonfun$withThreadLocalCaptured$1 at FutureTask.java:264	+details	2024/10/25 17:29:57	0.7 s	4/4	4.6 MiB			
2	showString at NativeMethodAccessorImpl.java:0	+details	2024/10/25 17:29:55	2 s	4/4	1840.4 KiB			1084.0 KiB
1	load at NativeMethodAccessorImpl.java:0	+details	2024/10/25 17:29:52	76 ms	1/1				
0	load at NativeMethodAccessorImpl.java:0	+details	2024/10/25 17:29:51	0.5 s	1/1				

Dag for each job:



Running .explain() on resultDF shows the Physical Plan:

```
✓ resultDF.explain() ···

== Physical Plan ==
AdaptiveSparkPlan isFinalPlan=false
+- Project [question_id#24L, creation_date#26, title#27, month#158, cnt#174L]
   +- BroadcastHashJoin [question_id#24L], [question_id#12L], Inner, BuildRight, false
      :- Filter isnotnull(question_id#24L)
      :  +- FileScan parquet [question_id#24L,creation_date#26,title#27] Batched: true, DataFilters: [isnotnull(question_id#
      +- BroadcastExchange HashedRelationBroadcastMode(List(input[0, bigint, true]),false), [plan_id=489]
         +- HashAggregate(keys=[question_id#12L, month#158], functions=[count(1)])
            +- Exchange hashpartitioning(question_id#12L, month#158, 200), ENSURE_REQUIREMENTS, [plan_id=486]
               +- HashAggregate(keys=[question_id#12L, month#158], functions=[partial_count(1)])
                  +- Project [question_id#12L, month(cast(creation_date#14 as date)) AS month#158]
                     +- Filter isnotnull(question_id#12L)
                        +- FileScan parquet [question_id#12L,creation_date#14] Batched: true, DataFilters: [isnotnull(questi
```

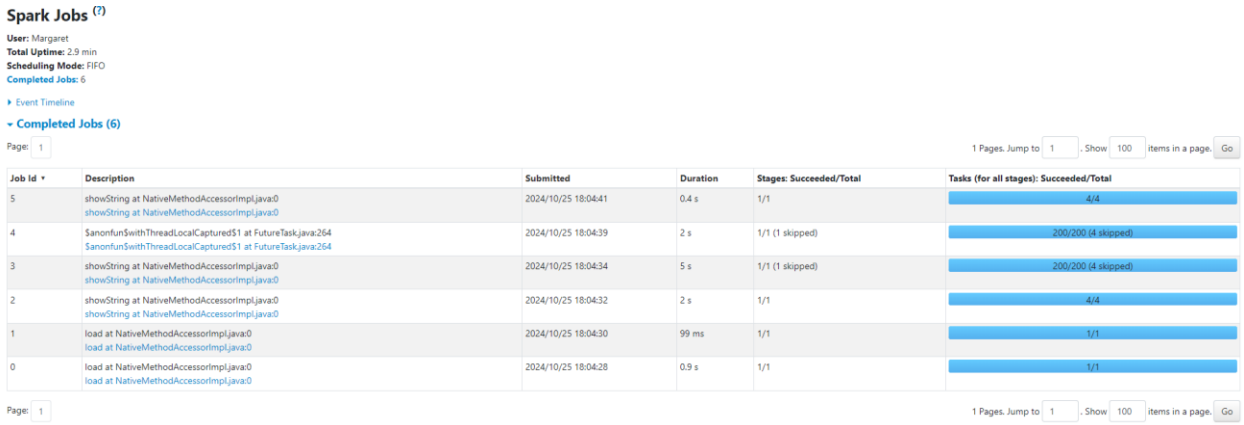
Code refactoring to improve run speed:

Caching:
A common method of improving performance is persisting data into cache so it can be reused. Here I cache answers_month.

```
#Caching answers_month
answers_month.cache()
resultDF_cached = questionsDF.join(answers_month,
'question_id') \
    .select('question_id', 'creation_date', 'title',
    'month', 'cnt')

resultDF_cached.orderBy('question_id', 'month').show()
answers_month.unpersist()
```

Result – Caching:



Spark Jobs (?)

User: Margaret
Total Uptime: 2.9 min
Scheduling Mode: FIFO
Completed Jobs: 6

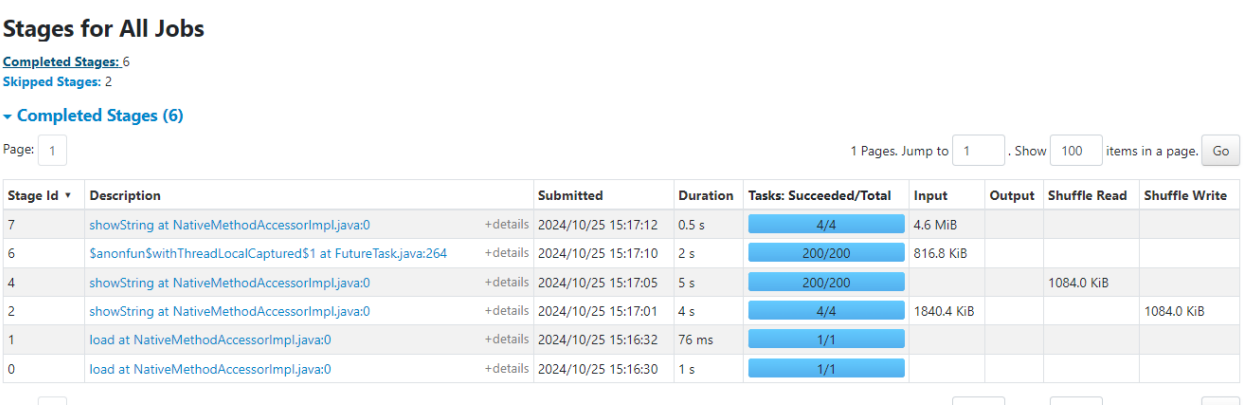
Event Timeline

Completed Jobs (6)

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

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
5	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2024/10/25 18:04:41	0.4 s	1/1	4/4
4	$anonfun$withThreadLocalCaptured$1 at FutureTask.java:264 $anonfun$withThreadLocalCaptured$1 at FutureTask.java:264	2024/10/25 18:04:39	2 s	1/1 (1 skipped)	200/200 (4 skipped)
3	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2024/10/25 18:04:34	5 s	1/1 (1 skipped)	200/200 (4 skipped)
2	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2024/10/25 18:04:32	2 s	1/1	4/4
1	load at NativeMethodAccessorImpl.java:0 load at NativeMethodAccessorImpl.java:0	2024/10/25 18:04:30	99 ms	1/1	1/1
0	load at NativeMethodAccessorImpl.java:0 load at NativeMethodAccessorImpl.java:0	2024/10/25 18:04:28	0.9 s	1/1	1/1

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



Stages for All Jobs

Completed Stages: 6
Skipped Stages: 2

Completed Stages (6)

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
7	showString at NativeMethodAccessorImpl.java:0 +details	2024/10/25 15:17:12	0.5 s	4/4	4.6 MiB			
6	$anonfun$withThreadLocalCaptured$1 at FutureTask.java:264 +details	2024/10/25 15:17:10	2 s	200/200	816.8 KiB			
4	showString at NativeMethodAccessorImpl.java:0 +details	2024/10/25 15:17:05	5 s	200/200			1084.0 KiB	
2	showString at NativeMethodAccessorImpl.java:0 +details	2024/10/25 15:17:01	4 s	4/4	1840.4 KiB			1084.0 KiB
1	load at NativeMethodAccessorImpl.java:0 +details	2024/10/25 15:16:32	76 ms	1/1				
0	load at NativeMethodAccessorImpl.java:0 +details	2024/10/25 15:16:30	1 s	1/1				

The final step (Job ID 5) decreased from 0.8s to 0.5s, but the additional step of caching the data increased the overall time (5s for Job ID 3). Total time of all jobs is 10.4s.

Repartition:

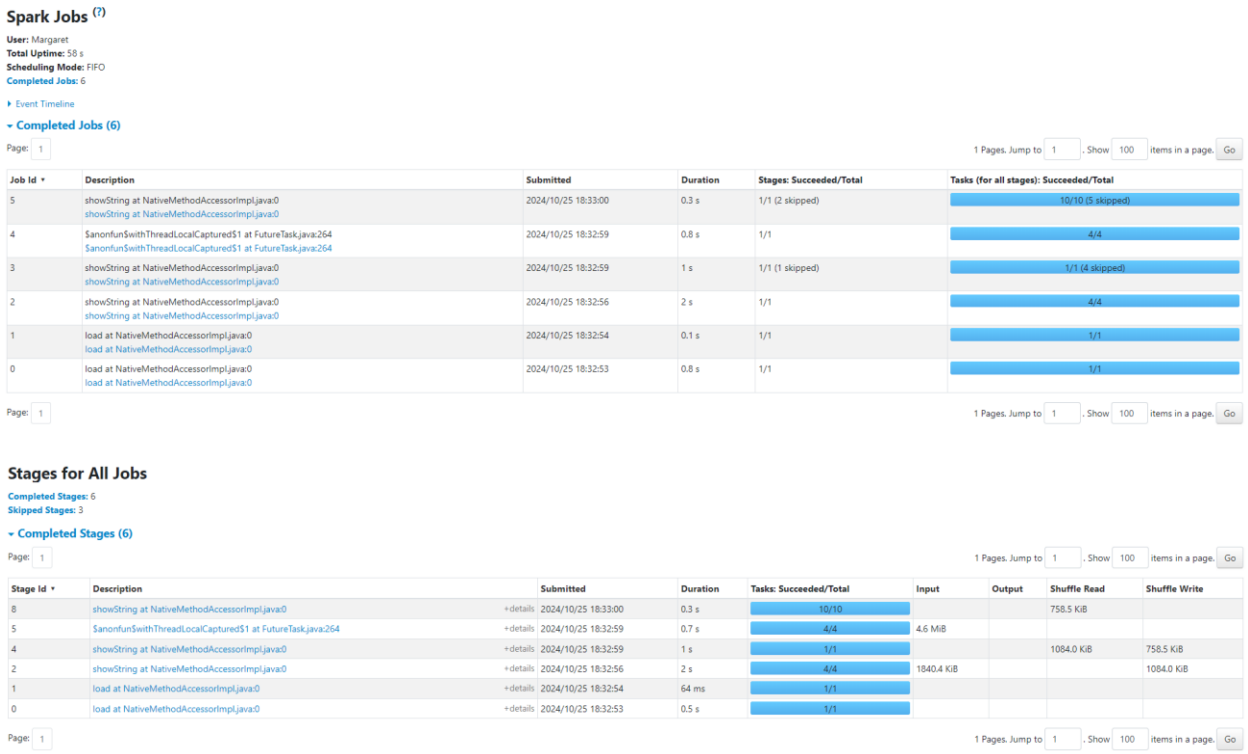
Repartitioning divides the data and helps prevent skew. Here I repartition answers_month into 10 partitions by question_id.

```
#Repartitioning answers_month
r_answers_month = answers_month.repartition(10,
'question_id')

resultDF_r = questionsDF.join(r_answers_month,
'question_id') \
    .select('question_id', 'creation_date', 'title',
    'month', 'cnt')

resultDF_r.orderBy('question_id', 'month').show()
```

Result:



Spark Jobs (?)

User: Margaret
Total Uptime: 58 s
Scheduling Mode: FIFO
Completed Jobs: 6

Event Timeline

Completed Jobs (6)

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

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
5	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2024/10/25 18:33:00	0.3 s	1/1 (2 skipped)	10/10 (5 skipped)
4	$anonfun$withThreadLocalCaptured$1 at FutureTask.java:264 $anonfun$withThreadLocalCaptured$1 at FutureTask.java:264	2024/10/25 18:32:59	0.8 s	1/1	4/4
3	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2024/10/25 18:32:59	1 s	1/1 (1 skipped)	1/1 (4 skipped)
2	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2024/10/25 18:32:56	2 s	1/1	4/4
1	load at NativeMethodAccessorImpl.java:0 load at NativeMethodAccessorImpl.java:0	2024/10/25 18:32:54	0.1 s	1/1	1/1
0	load at NativeMethodAccessorImpl.java:0 load at NativeMethodAccessorImpl.java:0	2024/10/25 18:32:53	0.8 s	1/1	1/1

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

Stages for All Jobs

Completed Stages: 6
Skipped Stages: 3

Completed Stages (6)

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
8	showString at NativeMethodAccessorImpl.java:0 +details	2024/10/25 18:33:00	0.3 s	10/10			758.5 KiB	
5	$anonfun$withThreadLocalCaptured$1 at FutureTask.java:264 +details	2024/10/25 18:32:59	0.7 s	4/4	4.6 MiB			
4	showString at NativeMethodAccessorImpl.java:0 +details	2024/10/25 18:32:59	1 s	1/1			1084.0 KiB	758.5 KiB
2	showString at NativeMethodAccessorImpl.java:0 +details	2024/10/25 18:32:56	2 s	4/4	1840.4 KiB			1084.0 KiB
1	load at NativeMethodAccessorImpl.java:0 +details	2024/10/25 18:32:54	64 ms	1/1				
0	load at NativeMethodAccessorImpl.java:0 +details	2024/10/25 18:32:53	0.5 s	1/1				

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

The job is small, so we still do not see much improvement. This time, the final stage has decreased from 0.8s to 0.3s, however, the timing for jobs in the middle have slightly increased. Total time of all jobs is 5s.

Conclusion:

Both repartitioning and caching improved the performance of the final step but had too high of an overhead cost to be beneficial overall. However, if the dataset was larger and the last step took a larger percentage of the total run time, then both these methods would work well in reducing the time spent.