Sparking Curiosity

@ Analytix Power

Who I am

- Data Engineer @ Expedia Group

- 😎 Data Engineer for the last 4-5 years
- 🤢 previously IT Audit and business continuity

- 🇳🇱 Moved to the Netherlands in August this year
- 🏄‍♂️ I like kitsurfing, climbing, cycling, skiing… adventure sports

Disclaimer: The views and opinions presented here are my own and not those of my employer

But not today…. Today, I am your colleague

- I will be sharing some learning from my last assignment, which involved creating an semantic layer table using Spark
- Lets make it a knowledge-sharing conversation about Spark. I will throw a question, and we can debate about it. Then, I tell you my findindings.
- **Please please please participate:** interrumpt, talk, ask, comment, challenge. Otherwise it will just be another borning presentation.

- THE OBJECTIVE: to get out saying that was an interesting conversation, and hopefully that we all learn something new.

First, lets look at the data (I)

transactions

```
*** bookingFactDF ***
+----------+-----------+---------------------+------------+--------+
|booking_id|customer_id|booking_date_time    |total_amount|currency|
+----------+-----------+---------------------+------------+--------+
|B001      |C001       |2023-09-14 19:20:16.468|100.0       |USD     |
|B002      |C002       |2023-09-15 19:20:16.468|200.0       |USD     |
|B003      |C003       |2023-09-16 19:20:16.468|150.0       |USD     |
|B004      |C004       |2023-09-14 19:20:16.468|100.0       |USD     |
|B005      |C005       |2023-09-15 19:20:16.468|200.0       |USD     |
|B006      |C005       |2023-09-16 19:20:16.468|150.0       |USD     |
|B007      |C001       |2023-09-14 19:20:16.468|100.0       |USD     |
|B008      |C002       |2023-09-15 19:20:16.468|200.0       |USD     |
|B009      |C003       |2023-09-16 19:20:16.468|150.0       |USD     |
|B010      |C001       |2023-09-14 19:20:16.468|100.0       |USD     |
|B011      |C002       |2023-09-15 19:20:16.468|200.0       |EUR     |
|B012      |C003       |2023-09-16 19:20:16.468|150.0       |GBP     |
+----------+-----------+---------------------+------------+--------+
```

datasets.scala

First, lets look at the data (II)

dimensions

*** clientDataDF ***

customer_id	country	age
C001	NLD	18
C002	ESP	19
C003	TUR	20
C004	CHN	21
C005	AUS	22

*** clientSurveyDataDF ***

customer_id	fav_color	fav_animal	nps_score
C001	red	chicken	7
C002	blue	pig	8
C003	green	carrot	9
C004	yellow	cat	7
C005	pink	shark	5

*** countryInformationDf

country_id	population
NLD	17.53
ESP	47.42
TUR	84.78
CHN	1425.533
AUS	25.69

*** currencyExchangesDf ***

currency_id	exchange_rate_usd
EUR	1.05
GBP	1.22

datasets.scala

In Spark, Does the order of joins impact performance?

In other words, are these two queries similar under the hood?

```
SELECT * FROM bookingFactDf booking
LEFT JOIN clientDataDF a ON booking.customer_id = a.customer_id
LEFT JOIN currencyExchangesDf b ON booking.currency = b.currency_id
LEFT JOIN clientSurveyDataDf c ON  booking.customer_id = c.customer_id
```

```
SELECT * FROM bookingFactDf booking
LEFT JOIN clientDataDF a ON booking.customer_id = a.customer_id
LEFT JOIN clientSurveyDataDf c ON  booking.customer_id = c.customer_id
LEFT JOIN currencyExchangesDf b ON booking.currency = b.currency_id
```

TIP: Look at the keys.

FIRST QUERY

```
#### smallUnorderedJoins
== Physical Plan ==
* SortMergeJoin LeftOuter (19)
:- * Sort (15)
:  +- Exchange (14)
:     +- * SortMergeJoin LeftOuter (13)
:        :- * Sort (9)
:        :  +- Exchange (8)
:        :     +- * SortMergeJoin LeftOuter (7)
:        :        :- * Sort (3)
:        :        :  +- Exchange (2)
:        :        :     +- * Scan ExistingRDD (1)
:        :        +- * Sort (6)
:        :           +- Exchange (5)
:        :              +- * Scan ExistingRDD (4)
:        +- * Sort (12)
:           +- Exchange (11)
:              +- * Scan ExistingRDD (10)
+- * Sort (18)
   +- Exchange (17)
      +- * Scan ExistingRDD (16)
```

Exchange: 6 Sort: 6

SECOND QUERY

```
#### smallOrderedJoins
== Physical Plan ==
* SortMergeJoin LeftOuter (17)
:- * Sort (13)
:  +- Exchange (12)
:     +- * SortMergeJoin LeftOuter (11)
:        :- * SortMergeJoin LeftOuter (7)
:        :  :- * Sort (3)
:        :  :  +- Exchange (2)
:        :  :     +- * Scan ExistingRDD (1)
:        :  +- * Sort (6)
:        :     +- Exchange (5)
:        :        +- * Scan ExistingRDD (4)
:        +- * Sort (10)
:           +- Exchange (9)
:              +- * Scan ExistingRDD (8)
+- * Sort (16)
   +- Exchange (15)
      +- * Scan ExistingRDD (14)
```

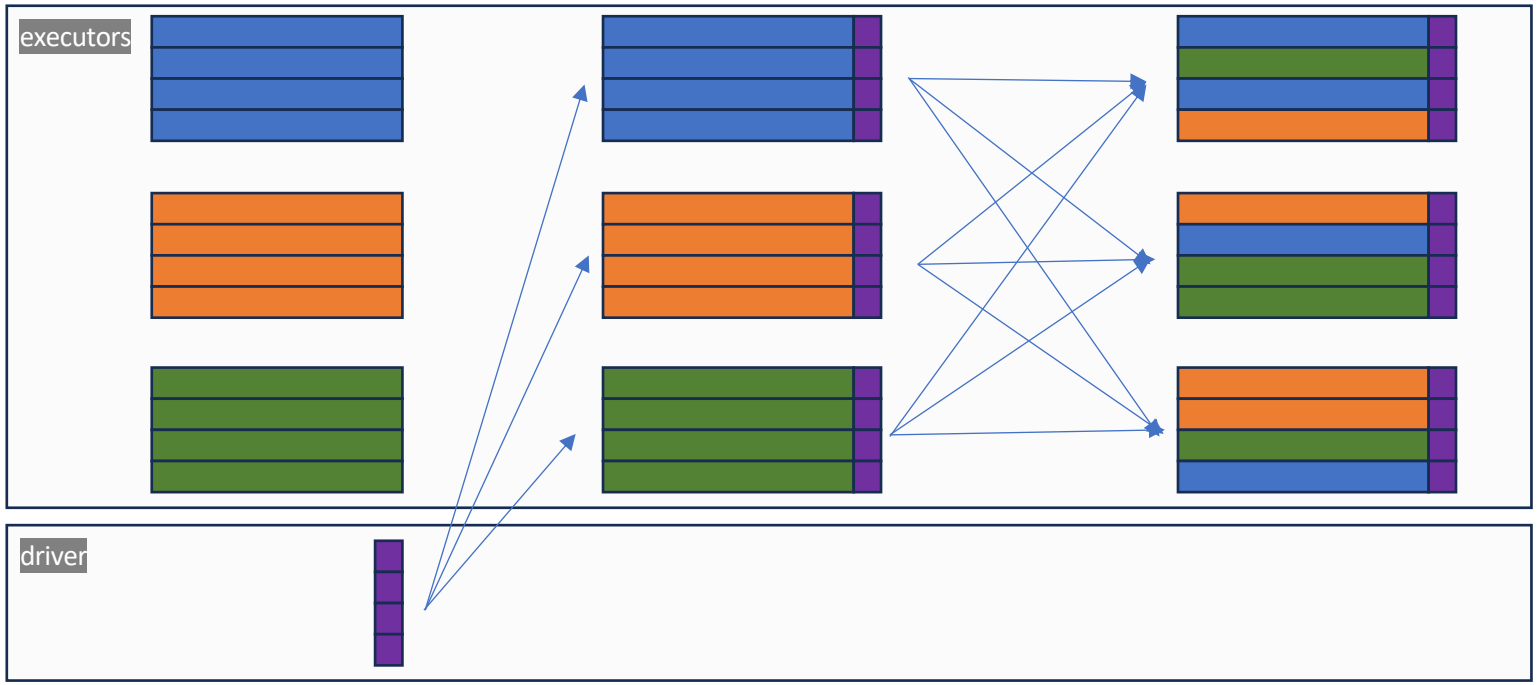
Exchange: 5 Sort: 5

orderOfJoins.scala

Can Adaptive Query Execution change this?

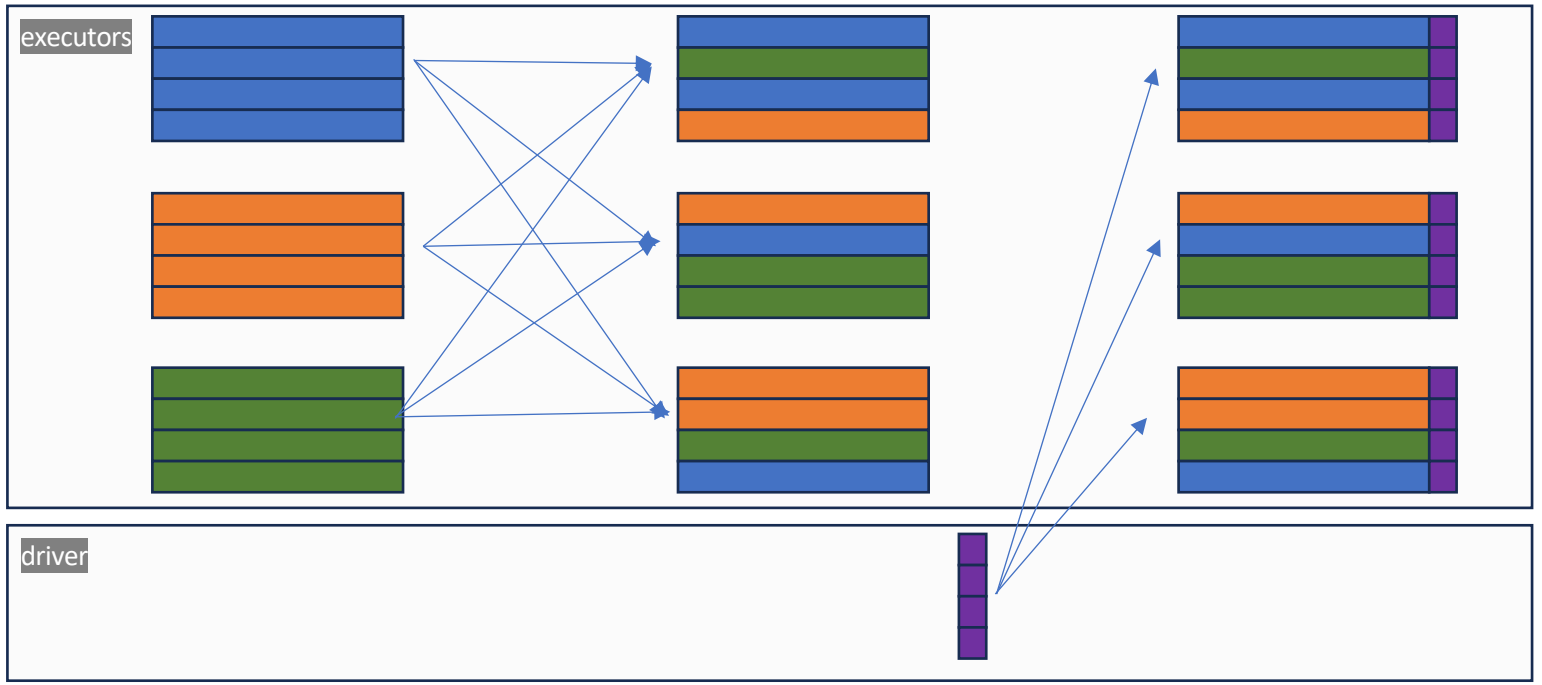
Is there a missing configuration parameter?

Does the order of a joins Borcasted tables impact pefromance?



How many times the broadcasted table (purple) was shuffled / moved around?

```
SELECT /*+ BROADCAST(b) */  * FROM bookingFactDf booking
LEFT JOIN clientDataDF a ON booking.customer_id = a.customer_id
LEFT JOIN currencyExchangesDf b ON booking.currency = b.currency_id
LEFT JOIN clientSurveyDataDf c ON  booking.customer_id = c.customer_id
```



How many times the broadcasted table (purple) was shuffled?

```
SELECT /*+ BROADCAST(b) */  * FROM bookingFactDf booking
LEFT JOIN clientDataDF a ON booking.customer_id = a.customer_id
LEFT JOIN clientSurveyDataDf c ON  booking.customer_id = c.customer_id
LEFT JOIN currencyExchangesDf b ON booking.currency = b.currency_id
```

If you have to join the following, what strategy would be better?

A

```
SELECT * FROM bookingFactDF bookings

LEFT JOIN clientDataDF  client
ON  bookings. customer_id = client. customer_id

LEFT JOIN countryInformationDf  country
ON client.country = country.country_id
```

B

```
WITH tt AS (
  SELECT * FROM ClientProfile
  LEFT JOIN RiskDimension rd
  ON cl. risk_id = rd. risk_model_id
)

SELECT * FROM bookingFactDF bookings
LEFT JOIN tt  cl
ON bookings.customer_id= cl.customer_id
```

Assume bookingFactDF is BILLIONS of rows
clientDataDF and countryInformationDf are in the millions. Broadcast is not possible

QUERY A

```
== Physical Plan ==
* SortMergeJoin LeftOuter (13)
:- * Sort (9)
:  +- Exchange (8)
:     +- * SortMergeJoin LeftOuter (7)
:        :- * Sort (3)
:        :  +- Exchange (2)
:        :     +- * Scan ExistingRDD (1)
:        +- * Sort (6)
:           +- Exchange (5)
:              +- * Scan ExistingRDD (4)
+- * Sort (12)
   +- Exchange (11)
      +- * Scan ExistingRDD (10)
```

QUERY B

```
== Physical Plan ==
* SortMergeJoin LeftOuter (13)
:- * Sort (3)
:  +- Exchange (2)
:     +- * Scan ExistingRDD (1)
+- * Sort (12)
   +- Exchange (11)
      +- * SortMergeJoin LeftOuter (10)
         :- * Sort (6)
         :  +- Exchange (5)
         :     +- * Scan ExistingRDD (4)
         +- * Sort (9)
            +- Exchange (8)
               +- * Scan ExistingRDD (7)
```

preJoins.scala

What could be the the purpose of the following code?

```
SELECT * FROM bookingFactDF
LEFT JOIN currencyExchangesDf
ON
coalesce(nullif(bookingFactDF.currency,”USD”), bookingFactDF.customer_id) = currencyExchangesDf.currency_id
```

Tip: look at the data

*** bookingFactDF ***

booking_id	customer_id	booking_date_time	total_amount	currency
B001	C001	2023-09-14 19:20:16.468	100.0	USD
B002	C002	2023-09-15 19:20:16.468	200.0	USD
B003	C003	2023-09-16 19:20:16.468	150.0	USD
B004	C004	2023-09-14 19:20:16.468	100.0	USD
B005	C005	2023-09-15 19:20:16.468	200.0	USD
B006	C005	2023-09-16 19:20:16.468	150.0	USD
B007	C001	2023-09-14 19:20:16.468	100.0	USD
B008	C002	2023-09-15 19:20:16.468	200.0	USD
B009	C003	2023-09-16 19:20:16.468	150.0	USD
B010	C001	2023-09-14 19:20:16.468	100.0	USD
B011	C002	2023-09-15 19:20:16.468	200.0	EUR
B012	C003	2023-09-16 19:20:16.468	150.0	GBP

*** currencyExchangesDf ***

currency_id	exchange_rate_usd
EUR	1.05
GBP	1.22

```
SELECT * FROM bookingFactDF
LEFT JOIN currencyExchangesDf
ON
bookingFactDF.currency = currencyExchangesDf.currency_id
```

```
+---------+-----+
|partition|count|
+---------+-----+
|22       |1    |
|63       |1    |
|181      |10   |
+---------+-----+
```

```
SELECT * FROM bookingFactDF
LEFT JOIN currencyExchangesDf
ON
coalesce(nullif(bookingFactDF.currency,”USD”), bookingFactDF.customer_id)
= currencyExchangesDf.currency_id
```

```
+---------+-----+
|partition|count|
+---------+-----+
|22       |1    |
|58       |2    |
|63       |1    |
|89       |1    |
|168      |2    |
|180      |3    |
|193      |2    |
+---------+-----+
```

skewJoins.scala

Would this be better?

```
SELECT bf.*, null as exchange_rate_usd, null as currency_id FROM bookingFactDF bf where currency = 'USD’
UNION

(SELECT * FROM bookingFactDF
LEFT JOIN currencyExchangesDf ON bookingFactDF.currency = currencyExchangesDf.currency_id
WHERE currency != 'USD')
```

skewJoins.scala

ORIGINAL QUERY

```
== Physical Plan ==
* SortMergeJoin LeftOuter (7)
:- * Sort (3)
:  +- Exchange (2)
:     +- * Scan ExistingRDD (1)
+- * Sort (6)
   +- Exchange (5)
      +- * Scan ExistingRDD (4)
```

USING COMMON TABLE EXPRESSIONS

```
== Physical Plan ==
* HashAggregate (16)
+- Exchange (15)
   +- * HashAggregate (14)
      +- Union (13)
         :- * Project (3)
         :  +- * Filter (2)
         :     +- * Scan ExistingRDD (1)
         +- * SortMergeJoin LeftOuter (12)
            :- * Sort (7)
            :  +- Exchange (6)
            :     +- * Filter (5)
            :        +- * Scan ExistingRDD (4)
            +- * Sort (11)
               +- Exchange (10)
                  +- * Filter (9)
                     +- * Scan ExistingRDD (8)
```

skewJoins.scala

How do you spot skewness without looking at the data?

Details for Job 9

Status: SUCCEEDED
Submitted: 2023/08/30 14:10:03
Duration: 2.6 h
Associated SQL Query: 0
Completed Stages: 16
Skipped Stages: 2

▸ Event Timeline
▸ DAG Visualization

▾ Completed Stages (16)

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
26	save at HiveDataFrameWriter.scala:217 +details	2023/08/30 16:34:32	14 min	7500/7500		261.9 GiB	1028.6 GiB	
25	save at HiveDataFrameWriter.scala:217 +details	2023/08/30 16:13:24	21 min	7500/7500			959.7 GiB	1028.6 GiB
24 (retry 1)	save at HiveDataFrameWriter.scala:217 +details	2023/08/30 15:53:47	20 min	7139/7139			963.8 GiB	911.2 GiB
23 (retry 1)	save at HiveDataFrameWriter.scala:217 +details	2023/08/30 16:13:22	1 s	8/8	29.3 MiB			100.6 MiB
22 (retry 2)	save at HiveDataFrameWriter.scala:217 +details	2023/08/30 15:48:51	4.9 min	1100/1100			143.0 GiB	139.0 GiB
20 (retry 2)	save at HiveDataFrameWriter.scala:217 +details	2023/08/30 15:47:34	1.3 min	186/186			18.3 GiB	24.2 GiB
19 (retry 3)	save at HiveDataFrameWriter.scala:217 +details	2023/08/30 15:39:23	0.5 s	131/131				
17 (retry 2)	save at HiveDataFrameWriter.scala:217 +details	2023/08/30 15:39:23	10 s	16/16	1129.4 MiB			1262.2 MiB
16 (retry 2)	save at HiveDataFrameWriter.scala:217 +details	2023/08/30 15:43:08	4.4 min	161/161			10.5 GiB	17.2 GiB
15 (retry 2)	save at HiveDataFrameWriter.scala:217 +details	2023/08/30 15:39:23	0.7 s	8/8	1631.7 KiB			
14 (retry 2)	save at HiveDataFrameWriter.scala:217 +details	2023/08/30 15:39:23	0.7 s	7/7	1418.9 KiB			
13 (retry 2)	save at HiveDataFrameWriter.scala:217 +details	2023/08/30 15:42:10	58 s	149/149			6.2 GiB	7.1 GiB
12 (retry 2)	save at HiveDataFrameWriter.scala:217 +details	2023/08/30 15:39:23	10 s	9/9	67.2 MiB			114.9 MiB
11 (retry 2)	save at HiveDataFrameWriter.scala:217 +details	2023/08/30 15:39:23	21 s	74/74	3.1 GiB			5.6 GiB
10 (retry 2)	save at HiveDataFrameWriter.scala:217 +details	2023/08/30 15:41:57	13 s	158/158			388.0 MiB	403.2 MiB
9 (retry 2)	save at HiveDataFrameWriter.scala:217 +details	2023/08/30 15:39:23	2.6 min	215/215	390.4 MiB			392.2 MiB

Spark Jobs (?)

User: smeschiari
Total Uptime: 1.1 min
Scheduling Mode: FIFO
Active Jobs: 1
Completed Jobs: 146, only showing 145

▸ Event Timeline

Active Jobs (1)

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
145	checkpoint at DatasetUtils.scala:464 checkpoint at DatasetUtils.scala:464 (kill)	2019/02/17 17:05:10	20 min	2/5	521/808 (1 running)

Completed Jobs (146, only showing 145)

Page: 1 2 > 2 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
144	run at ThreadPoolExecutor.java:1142 run at ThreadPoolExecutor.java:1142	2019/02/17 17:05:09	1 s	2/2 (2 skipped)	400/400 (208 skipped)
143	count at FeatureExtract.scala:150 count at FeatureExtract.scala:150	2019/02/17 17:05:05	2 s	4/4	409/409

Details for Stage 25 (Attempt 0)

Resource Profile Id: 0
Total Time Across All Tasks: 6.6 h
Locality Level Summary: Process local: 1000
Shuffle Read Size / Records: 58.7 GiB / 1694528788
Shuffle Write Size / Records: 190.2 GiB / 502081312
Spill (Memory): 129.0 GiB
Spill (Disk): 36.7 GiB
Associated Job Ids: 17

▸ DAG Visualization
▸ Show Additional Metrics
▸ Event Timeline

Summary Metrics for 1000 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	13 s	17 s	19 s	21 s	1.3 h
GC Time	0.0 ms	16.0 ms	24.0 ms	36.0 ms	7 s
Spill (memory)	0.0 B	0.0 B	0.0 B	0.0 B	129 GiB
Spill (disk)	0.0 B	0.0 B	0.0 B	0.0 B	36.7 GiB
Shuffle Read Size / Records	54.1 MiB / 1642587	54.5 MiB / 1648407	54.5 MiB / 1649698	54.6 MiB / 1650964	5.5 GiB / 46503178
Shuffle Write Size / Records	175.9 MiB / 452455	177.1 MiB / 456350	177.4 MiB / 457175	177.7 MiB / 458159	17.2 GiB / 45310020

Showing 1 to 6 of 6 entries

Aggregated Metrics by Executor

Show 20 entries Search:

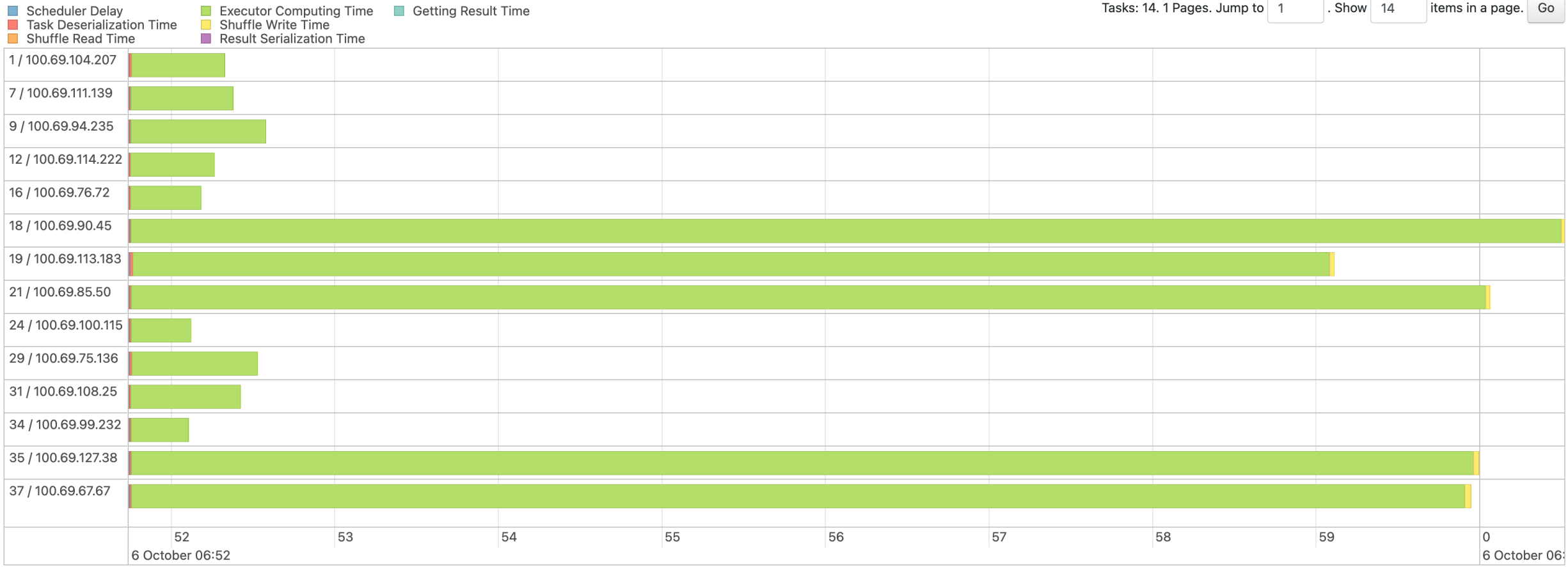
Executor ID	Logs	Address	Task Time	Total Tasks	Failed Tasks	Killed Tasks	Succeeded Tasks	Excluded	Shuffle Read Size / Records	Shuffle Write Size / Records	Spill (Memory)	Spill (Disk)
22		100.69.90.204:43441	1.4 h	21	0	0	21	false	6.5 GiB / 79500404	20.6 GiB / 54454841	129 GiB	36.7 GiB
39		100.69.104.181:39031	7.6 min	30	0	0	30	false	1.6 GiB / 49491731	5.2 GiB / 13715005	0.0 B	0.0 B
16		100.69.76.72:39031	8.2 min	29	0	0	29	false	1.5 GiB / 47847867	5 GiB / 13265382	0.0 B	0.0 B
20		100.69.97.56:34137	7.9 min	28	0	0	28	false	1.5 GiB / 46188414	4.8 GiB / 12802323	0.0 B	0.0 B
31		100.69.108.25:38431	8.3 min	27	0	0	27	false	1.4 GiB / 44555607	4.7 GiB / 12359890	0.0 B	0.0 B
[illegible]		[illegible]	8.3 min	27	0	0	27	false	1.4 GiB / 44543888	4.7 GiB / 12347470	0.0 B	0.0 B

▸ DAG Visualization

▸ Show Additional Metrics

▾ Event Timeline

☐ Enable zooming



Summary Metrics for 14 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	0.3 s	0.5 s	0.8 s	8 s	9 s
GC Time	0.0 ms	0.0 ms	0.0 ms	0.0 ms	24.0 ms
Input Size / Records	2.2 KiB / 0	2.2 KiB / 0	2.2 KiB / 0	43.7 MiB / 5962238	43.9 MiB / 5962241
Shuffle Write Size / Records	0.0 B / 0	0.0 B / 0	0.0 B / 0	55.8 MiB / 5962238	55.8 MiB / 5962241

Showing 1 to 4 of 4 entries

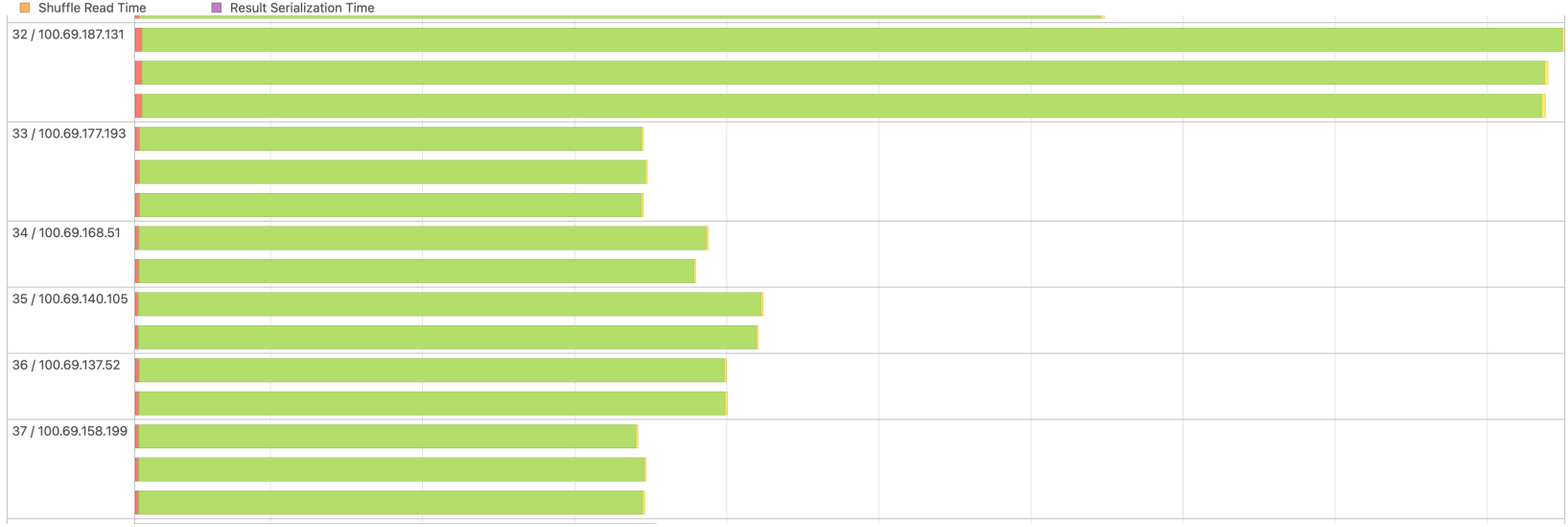
Details for Stage 25 (Attempt 0)

Resource Profile Id: 0
Total Time Across All Tasks: 7.9 h
Locality Level Summary: Process local: 1000
Shuffle Read Size / Records: 58.7 GiB / 1696992160
Shuffle Write Size / Records: 190.4 GiB / 502657029
Spill (Memory): 129.1 GiB
Spill (Disk): 36.7 GiB
Associated Job Ids: 17

▸ DAG Visualization
▸ Show Additional Metrics
▸ Event Timeline
☐ Enable zooming

Scheduler Delay · Task Deserialization Time · Shuffle Read Time · Executor Computing Time · Shuffle Write Time · Result Serialization Time · Getting Result Time

Tasks: 1000. 10 Pages. Jump to 1 . Show 100 items in a page. Go



Summary Metrics for 1000 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	13 s	18 s	21 s	24 s	1.4 h
GC Time	0.0 ms	9.0 ms	20.0 ms	31.0 ms	11 s
Spill (memory)	0.0 B	0.0 B	0.0 B	0.0 B	129.1 GiB
Spill (disk)	0.0 B	0.0 B	0.0 B	0.0 B	36.7 GiB
Shuffle Read Size / Records	54.2 MiB / 1645073	54.5 MiB / 1650801	54.6 MiB / 1652104	54.7 MiB / 1653359	5.5 GiB / 46559546
Shuffle Write Size / Records	176.1 MiB / 453073	177.3 MiB / 456856	177.6 MiB / 457693	177.9 MiB / 458686	17.2 GiB / 45364518

Showing 1 to 6 of 6 entries

Aggregated Metrics by Executor

Show 20 entries Search:

Executor ID	Logs	Address	Task Time	Total Tasks	Failed Tasks	Killed Tasks	Succeeded Tasks	Excluded	Shuffle Read Size / Records	Shuffle Write Size / Records	Spill (Memory)	Spill (Disk)
1		100.69.128.245:33931	9.6 min	24	0	0	24	false	1.3 GiB / 39645727	4.2 GiB / 10992742	0.0 B	0.0 B
2		100.69.134.77:38339	9.6 min	24	0	0	24	false	1.3 GiB / 39647296	4.2 GiB / 10981500	0.0 B	0.0 B
3		100.69.191.229:35005	9.9 min	24	0	0	24	false	1.3 GiB / 39639559	4.2 GiB / 10982438	0.0 B	0.0 B

THANK YOU