Dataflow Timeouts, slower runs, long duration etc

Last updated by | Anudeep Sharma | Mar 3, 2023 at 11:43 AM PST

Contents

Why timeout can happen?

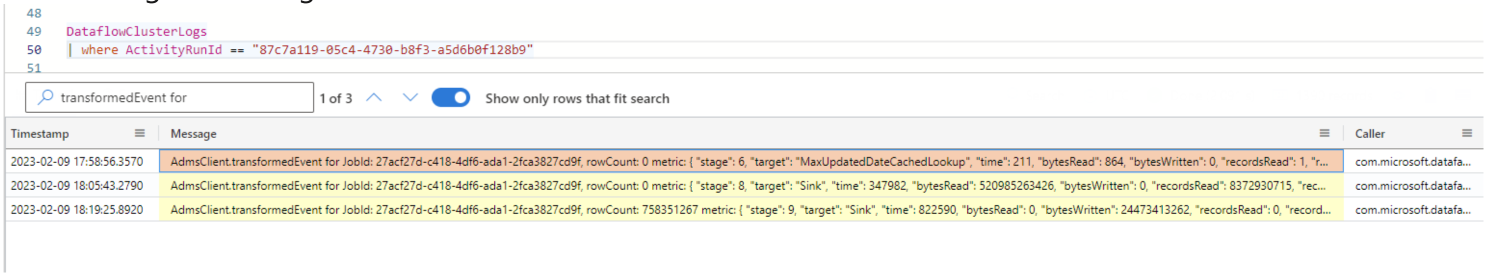
Timeout is the setting set by user (there are always defaults), so basically customer is expecting the run to be completed in certain duration otherwise they are instructing ADF to kill the activity. Root cause for timeouts are either longer duration or customer's wrong expectations.

Introduction to transformedEvent

In case you need more idea about the how dataflows run internally look at following link.
https://supportability.visualstudio.com/AzureDataFactory/_wiki/wikis/AzureDataFactory/857242/Introduction-to-different-concepts-in-Mapping-Dataflows?anchor=how-do-dataflows-run-internally%3F

Dataflows runs on spark and multiple transformations are combined together to create stages. For an example if there is a Source->Aggregate->Sink there will be 2 stages. Source->Aggregate will be a stage as for aggregate like count all rows, all rows much be processed before moving forward. Then there will be stage for output of the aggregate to sink. Every such stage completion is mentioned in the logs in DataflowClusterLogs marked with transformedEvent.

Like in image below logs are visible for the transformed event



```
DataflowClusterLogs
| where ActivityRunId == "87c7a119-05c4-4730-b8f3-a5d6b0f128b9"
```

transformedEvent for 1 of 3 Show only rows that fit search

Timestamp	Message	Caller
2023-02-09 17:58:56.3570	AdmsClient.transformedEvent for JobId: 27acf27d-c418-4df6-ada1-2fca3827cd9f, rowCount: 0 metric: { "stage": 6, "target": "MaxUpdatedDateCachedLookup", "time": 211, "bytesRead": 864, "bytesWritten": 0, "recordsRead": 1, "r...	com.microsoft.datafa...
2023-02-09 18:05:43.2790	AdmsClient.transformedEvent for JobId: 27acf27d-c418-4df6-ada1-2fca3827cd9f, rowCount: 0 metric: { "stage": 8, "target": "Sink", "time": 347982, "bytesRead": 520985263426, "bytesWritten": 0, "recordsRead": 8372930715, "rec...	com.microsoft.datafa...
2023-02-09 18:19:25.8920	AdmsClient.transformedEvent for JobId: 27acf27d-c418-4df6-ada1-2fca3827cd9f, rowCount: 758351267 metric: { "stage": 9, "target": "Sink", "time": 822590, "bytesRead": 0, "bytesWritten": 24473413262, "recordsRead": 0, "record...	com.microsoft.datafa...

TransformedEvents provide the number of records process and also name of the transformations processed till now. For OOM and for timeouts both transformed events are helpful to analyze the progress and what is remaining. There are helper function in the TSG below to get summary of stages, this helper function is built on top of transformedEvents only
https://supportability.visualstudio.com/AzureDataFactory/_wiki/wikis/AzureDataFactory/440661/How-to-get-number-of-rows-processed-time-taken-for-different-stages-for-Dataflow-activity

DataflowClusterLogs | where ActivityRunId == "xxxxxxxxxx"

In case there are no logs in DataflowClusterLogs activity id, make sure it is ActivityRunId only, you can make sure by running a query like ActivityRuns | where activityRunId == 'xxxxxxxxxx', if this query does not produce any row that means either this is very old activity run id (we don't have logs anymore), or right kusto cluster is not used or this is not an activity run id.

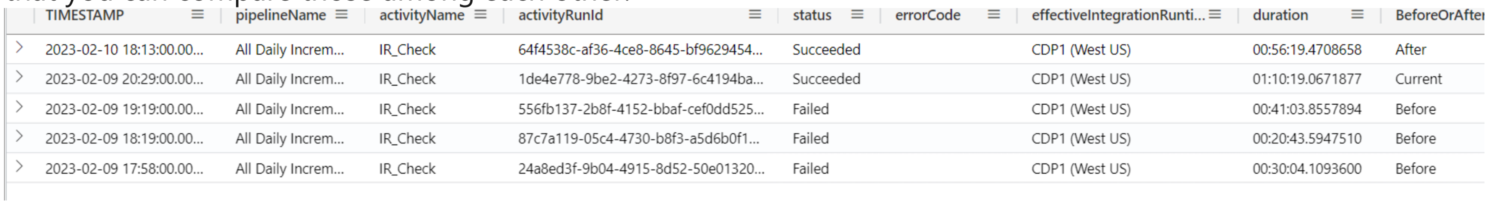
Analyzing transformedEvent

TransformedEvent contains the duration for the stage, number of rows, transformations involved for that stage, it also contains for which sink this stage was being processed. If a stage contains the sink transformations that means it is terminal stage for that sink.

In case you have faster run as well

It will be good to compare the stages (transformedEvents) to see if the number of rows are same for similar stages in both slow and faster runs. If scale of number of rows is different then it is obvious that there will be more time taken for more number of rows. Still customer can ask for how to make it faster, look at the public documentation or the TSG mentioned above to know more about the mapping dataflow and suggest user what are their options. You still might need to find the bottleneck, like pre/post queries on SQL could be slower because of more data, in that case bigger IR/Cluster, more partitions will not help.

Event without asking customer you can get the other runs for same activity by running following function GetDataflowActivityBeforAfterRunInformation(activityID) Above will provide more information of other runs, so that you can compare those among each other.



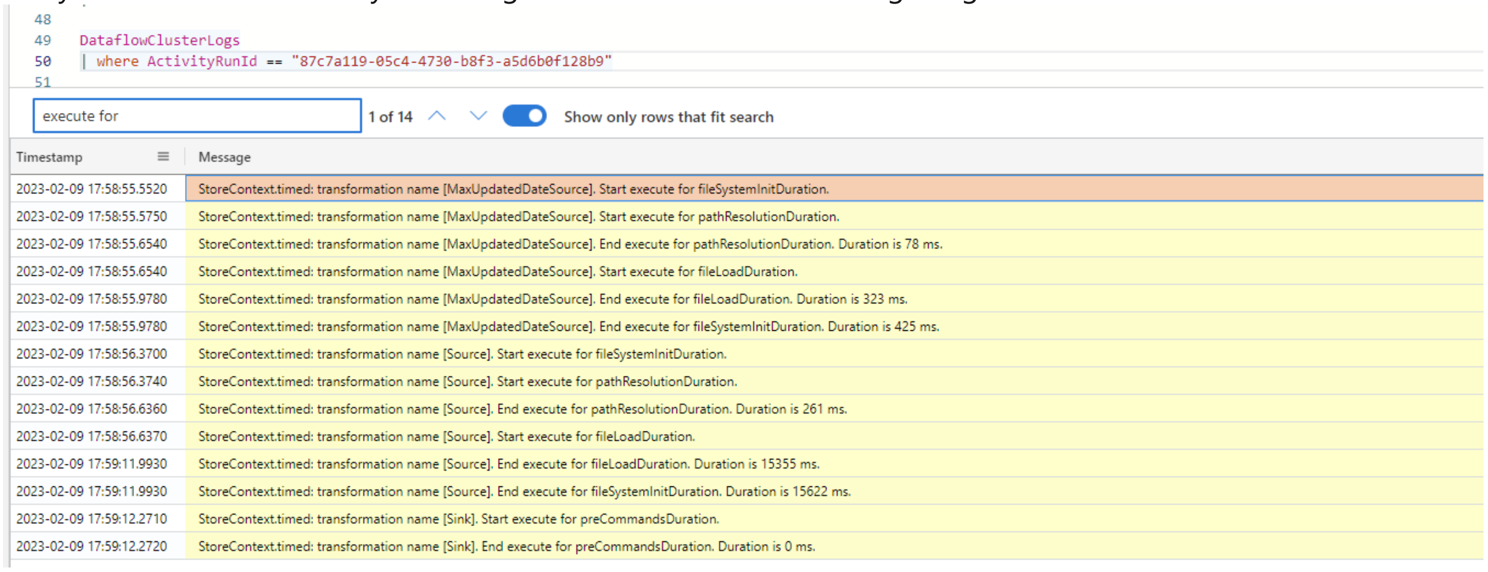
	TIMESTAMP	pipelineName	activityName	activityRunId	status	errorCode	effectiveIntegrationRunti...	duration	BeforeOrAfter
>	2023-02-10 18:13:00.00...	All Daily Increm...	IR_Check	64f4538c-af36-4ce8-8645-bf9629454...	Succeeded		CDP1 (West US)	00:56:19.4708658	After
>	2023-02-09 20:29:00.00...	All Daily Increm...	IR_Check	1de4e778-9be2-4273-8f97-6c4194ba...	Succeeded		CDP1 (West US)	01:10:19.0671877	Current
>	2023-02-09 19:19:00.00...	All Daily Increm...	IR_Check	556fb137-2b8f-4152-bbaf-cef0dd525...	Failed		CDP1 (West US)	00:41:03.8557894	Before
>	2023-02-09 18:19:00.00...	All Daily Increm...	IR_Check	87c7a119-05c4-4730-b8f3-a5d6b0f1...	Failed		CDP1 (West US)	00:20:43.5947510	Before
>	2023-02-09 17:58:00.00...	All Daily Increm...	IR_Check	24a8ed3f-9b04-4915-8d52-50e01320...	Failed		CDP1 (West US)	00:30:04.1093600	Before

There are no transformedEvents or it came a long time after the first log appears in DataflowClusterLogs

In case there are no transformedEvent or it came late, then problem is on reader transformation. Either wild card path or query on SQL is taking long time. Look for what kind of source is being read and dig for more information accordingly. There are TSGs to know about wild card path and sql sources. More is mentioned below in this TSG about searching for 'execute for' in the DataflowClusterLogs.

There is long delay after last transformedEvent

First thing to cross check if this supposed to be the last transformedEvent. Easy way to check is compare with the completed runs or if this a successful run then it is last transformedEvent. In case this is timedout run and there is no successful run available then you need to look note down all the sinks processed till now with each transformedEvent and in that event transformations should contains the sink name as well. If it is last sink then processing is taking time in the post actions like in case data is written to SQL, then we write the data to staging table (you can learn more from the concepts TSG), then data is moved within the SQL server through SQL commands from staging to target tables, that might be taking time. Or pre/post SQL can take time. You can analyze more about these by searching 'execute for' like in following image.



```
DataflowClusterLogs
| where ActivityRunId == "87c7a119-05c4-4730-b8f3-a5d6b0f128b9"
```

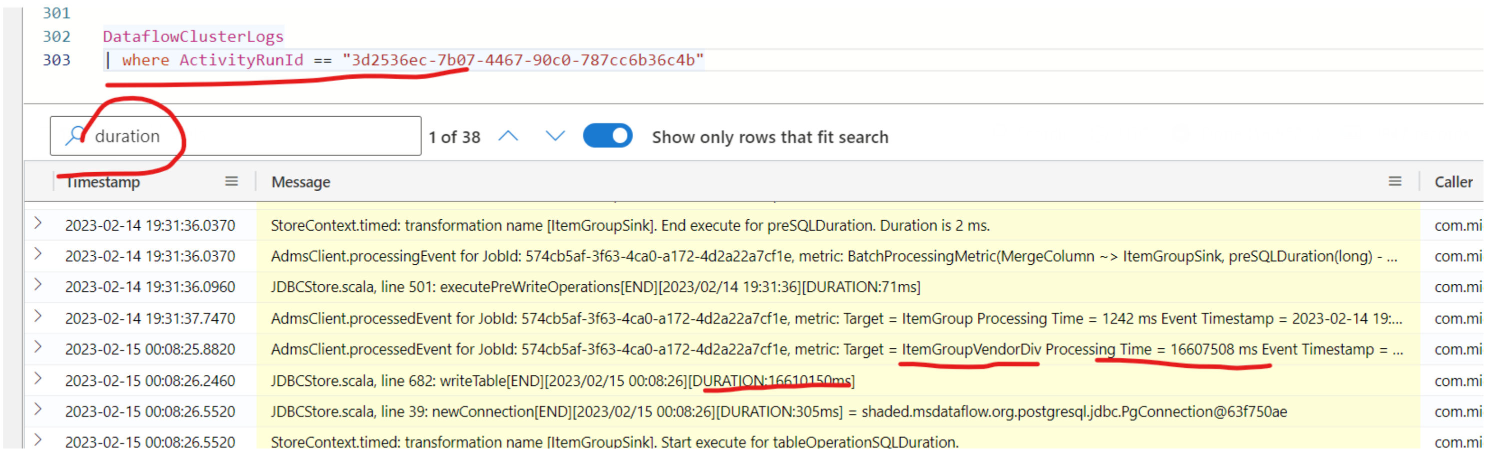
execute for 1 of 14 Show only rows that fit search

Timestamp	Message
2023-02-09 17:58:55.5520	StoreContext.timed: transformation name [MaxUpdatedDateSource]. Start execute for fileSystemInitDuration.
2023-02-09 17:58:55.5750	StoreContext.timed: transformation name [MaxUpdatedDateSource]. Start execute for pathResolutionDuration.
2023-02-09 17:58:55.6540	StoreContext.timed: transformation name [MaxUpdatedDateSource]. End execute for pathResolutionDuration. Duration is 78 ms.
2023-02-09 17:58:55.6540	StoreContext.timed: transformation name [MaxUpdatedDateSource]. Start execute for fileLoadDuration.
2023-02-09 17:58:55.9780	StoreContext.timed: transformation name [MaxUpdatedDateSource]. End execute for fileLoadDuration. Duration is 323 ms.
2023-02-09 17:58:55.9780	StoreContext.timed: transformation name [MaxUpdatedDateSource]. End execute for fileSystemInitDuration. Duration is 425 ms.
2023-02-09 17:58:56.3700	StoreContext.timed: transformation name [Source]. Start execute for fileSystemInitDuration.
2023-02-09 17:58:56.3740	StoreContext.timed: transformation name [Source]. Start execute for pathResolutionDuration.
2023-02-09 17:58:56.6360	StoreContext.timed: transformation name [Source]. End execute for pathResolutionDuration. Duration is 261 ms.
2023-02-09 17:58:56.6370	StoreContext.timed: transformation name [Source]. Start execute for fileLoadDuration.
2023-02-09 17:59:11.9930	StoreContext.timed: transformation name [Source]. End execute for fileLoadDuration. Duration is 15355 ms.
2023-02-09 17:59:11.9930	StoreContext.timed: transformation name [Source]. End execute for fileSystemInitDuration. Duration is 15622 ms.
2023-02-09 17:59:12.2710	StoreContext.timed: transformation name [Sink]. Start execute for preCommandsDuration.
2023-02-09 17:59:12.2720	StoreContext.timed: transformation name [Sink]. End execute for preCommandsDuration. Duration is 0 ms.

You will see how fileSystemInitDuration, pathResolutionDuration, preCommand etc has start and end both. In case there is something stated does not end that is taking more time, or if it is ended there is time noted for that in the image. Above example is for the file source and sink thats why you don't see pre/post sql etc. in case of sql sinks you will see sql related durations like this.

Searching duration

You can also search for just duration, like in following example there is duration to write to table is more.



```
DataflowClusterLogs
| where ActivityRunId == "3d2536ec-7b07-4467-90c0-787cc6b36c4b"
```

duration 1 of 38 Show only rows that fit search

Timestamp	Message	Caller
2023-02-14 19:31:36.0370	StoreContext.timed: transformation name [ItemGroupSink]. End execute for preSQLDuration. Duration is 2 ms.	com.mi
2023-02-14 19:31:36.0370	AdmsClient.processingEvent for JobId: 574cb5af-3f63-4ca0-a172-4d2a22a7cf1e, metric: BatchProcessingMetric(MergeColumn ~> ItemGroupSink, preSQLDuration(long) - ...	com.mi
2023-02-14 19:31:36.0960	JDBCStore.scala, line 501: executePreWriteOperations[END][2023/02/14 19:31:36][DURATION:71ms]	com.mi
2023-02-14 19:31:37.7470	AdmsClient.processedEvent for JobId: 574cb5af-3f63-4ca0-a172-4d2a22a7cf1e, metric: Target = ItemGroup Processing Time = 1242 ms Event Timestamp = 2023-02-14 19:...	com.mi
2023-02-15 00:08:25.8820	AdmsClient.processedEvent for JobId: 574cb5af-3f63-4ca0-a172-4d2a22a7cf1e, metric: Target = ItemGroupVendorDiv Processing Time = 16607508 ms Event Timestamp = ...	com.mi
2023-02-15 00:08:26.2460	JDBCStore.scala, line 682: writeTable[END][2023/02/15 00:08:26][DURATION:16610150ms]	com.mi
2023-02-15 00:08:26.5520	JDBCStore.scala, line 39: newConnection[END][2023/02/15 00:08:26][DURATION:305ms] = shaded.msdataflow.org.postgresql.jdbc.PgConnection@63f750ae	com.mi
2023-02-15 00:08:26.5520	StoreContext.timed: transformation name [ItemGroupSink]. Start execute for tableOperationSQLDuration.	com.mi

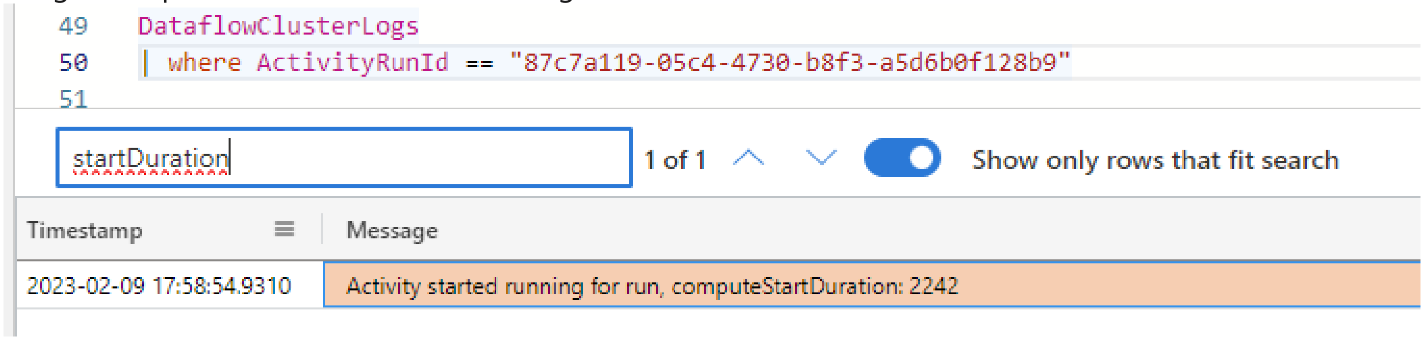
There is a long gap between the transformedEvent

If there is a long gap between the 2 transformedEvents look for what is the happening between those. There could be other sources being processed then it become case like previous one. Every transformedEvent is suggesting end of the stage, but there is **transformingEvent** which is suggesting if the stage is progressing. If after the transformedEvent there are transformingEvent right away then processing is going on, if not then it is reading some sources or stuck. In this case basically above sections are applicable.

Rare events

In rare cases you see see

- Longer compute start duration, like in image below. Duration is in milliseconds.



```
DataflowClusterLogs
| where ActivityRunId == "87c7a119-05c4-4730-b8f3-a5d6b0f128b9"
```

startDuration 1 of 1 Show only rows that fit search

Timestamp	Message
2023-02-09 17:58:54.9310	Activity started running for run, computeStartDuration: 2242

- Long gap between start of activity from orchestration or ADMS side. You can compare the start time of logs for same activity run id in ActivityRuns table (orchestration) with CustomLogEvent table (ADMS) and then with DataflowClusterLogs (actual processing of dataflows). If start times of logs are way apart like Orchestration to ADMS few minutes that is a problem. ADMS to DataflowCluster logs can be longer about 5 minutes as Spark cluster is starting and that should be approximately matching with compute start duration as pointed in previous bullet point.

Follow this TSG for high File init durations

How good have you found this content?