Delay in Pipeline Runs Troubleshooting

Last updated by | Ranjith Katukojwala | Feb 1, 2022 at 11:13 AM PST

Contents

Issue

The customer might raise concerns that their pipeline might run for a longer period with delays.

Cause

There might be some scenarios that might cause delays

Service Issue:

- As we know that ADF has a dependency on Logic apps, there might be chances due to a partner service outage or any other partner team-related issues the delays may occur. Can refer to this ICM for more details on this scenario.

By design behavior:

- A single pipeline run has multiple For Each activity and inside these activities may configure with Copy activities. Copy activities to be more precise, inside of a single pipeline run. Due to this, the Orchestration machine gets overloaded and occasionally delays activity completion and throttling issues on logic apps.

Recommendation

In order to prevent the delays happens due to:

Service Issue:

- For the LogicApps issue, you may leverage ASC or outage hub to check the related outages. If there is an outage and it get resolved, the workflow is for the customer to stop the runs and try it again.

By design behavior:

Customers must redesign the use of Azure Data Factory's pipelines. Could be achievable in the following ways:

- Split the current pipeline into multiple pipelines, they can be identical, but by having multiple instances, the load would be split across multiple machines and thus have a much less chance of overloading.
- Set a concurrency for the number of Copy activities that can be running in parallel. When there isn't a limit, the pipeline will attempt to schedule all the Copy activities in parallel thus causing the overload.

Important information you need to know.

Note: #1 is specifically for our internal knowledge and #2 , #3 could share with customers but recommended not to share directly.

1. **LogicApps is an implementation detail that should never be shared with customers**.

Our customers are ADF customers. When we partner with another azure service and something goes bad, this is completely on us. We will work with the partner (or move away from the partnership) and make sure that everything works. We share the implementation details with CSS so you understand what is happening, but sharing this with customers will only create confusion. ADF is responsible for the E2E customer experience.

2. **ADF is a batch service (not a streaming service) -> Small delays are by design**.

This is an extremely important concept, and not sure everyone understands what it means. Imagine you are streaming a movie and one of the frames is stored in a bad sector. The service provider could keep multiple copies of the movie, identify the bad frame, read for a secondary location, and properly display the frame. Obviously, this would take a bit of time and ruin your entertainment experience. A much better option is to skip the frame and move forward. Retry logic makes no sense for streaming services. This is a very important design decision. ADF was designed as a batch service. Reliability is the most important metric for us. We want customers to trust that every pipeline that is scheduled to run, will succeed. This means that we have complex retry logic. Our design assumes that some delay in the activity execution is acceptable to ensure that the activity will succeed. We could have built a streaming service, but we felt like a batch service would be a better choice for our customers.

Obviously, we also want pipelines to complete as scheduled. We can’t simply assume that failures will happen all the time and we’ll keep on retrying. Every time an activity changes state (Queued, In Progress, Completed), we are invoking several REST APIs. We have defined a 4min transition SLA so we can retry these API calls. This design significantly increases the chances of success for an activity.

3. **ADF is a 99.9% SLA service -> Some activities will violate the 4min transition SLA.**

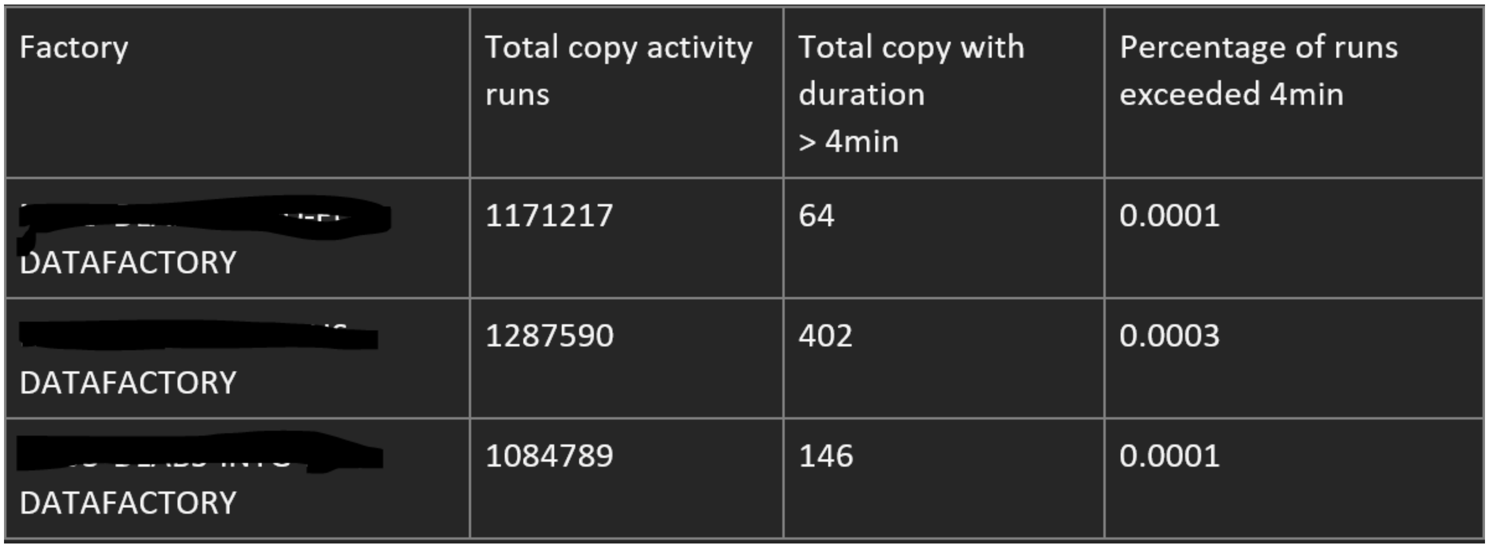
If we were building solutions for an airplane, we would need to build 100% SLA solutions. To begin with, we would not have the flexibility of relying on other 99.9% SLA Azure services. We could have done it, but the solution would be much more expensive. We made the business decision to assume 99.9% SLA, as this works for most customers.

Of course, we are always proactively identifying failures on our side and fixing them. We are constantly working towards a better service, but there should be an expectation that some failures (or 4min transition SLA misses) will happen. We will never be a 100% SLA service. This is by design.

Troubleshooting

We have recently got a case from an S500 customer where their pipelines experienced delays especially with one specific activity (Copy) with a lot of runs.

The below numbers include 3 transitions (Queued, In Progress, Completed) + actual copy execution. As you can see in the table, we are completing these activities in less than 4min 99.99% of the time. This means that, obviously, we are respecting our 99.9% SLA for the 4min transition SLA. This customer is executing a lot of activities, so failures will be inevitable. We need to put these failures in context and understand that they are a very small percentage of the total.



Factory	Total copy activity runs	Total copy with duration > 4min	Percentage of runs exceeded 4min
[redacted] DATAFACTORY	1171217	64	0.0001
[redacted] DATAFACTORY	1287590	402	0.0003
[redacted] DATAFACTORY	1084789	146	0.0001

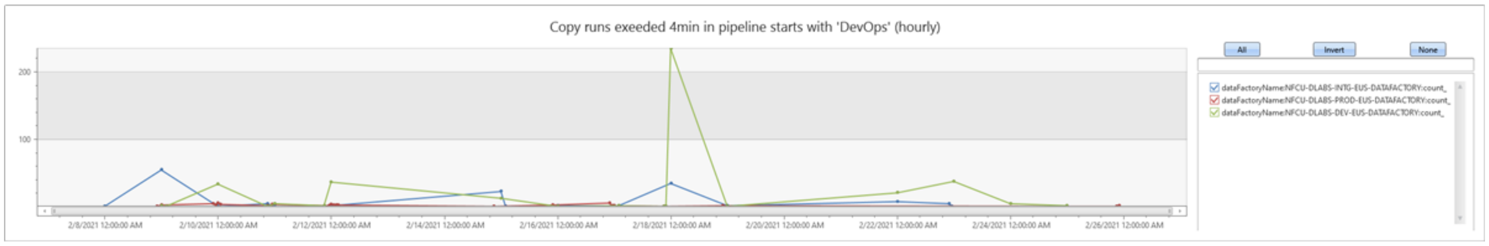
Queries to get the above data

```
//Query used for the above total runs:

ActivityRuns
| where TIMESTAMP > datetime(2021-02-08 00:00:00.0000000)
| where TIMESTAMP < datetime(2021-02-26 00:00:00.0000000)
| where location == "eastus"
| where dataFactoryName in ()
| where activityType == "" //Copy
| where status in ("Succeeded", "Failed", "Cancelled")
| extend duration = end - start
//| where duration > time(00:04:00)
| where pipelineName startswith ""
| project TIMESTAMP, duration, dataFactoryName, pipelineName, pipelineRunId, activityName, activityRunId, star
| order by duration asc
| summarize count() by dataFactoryName
```

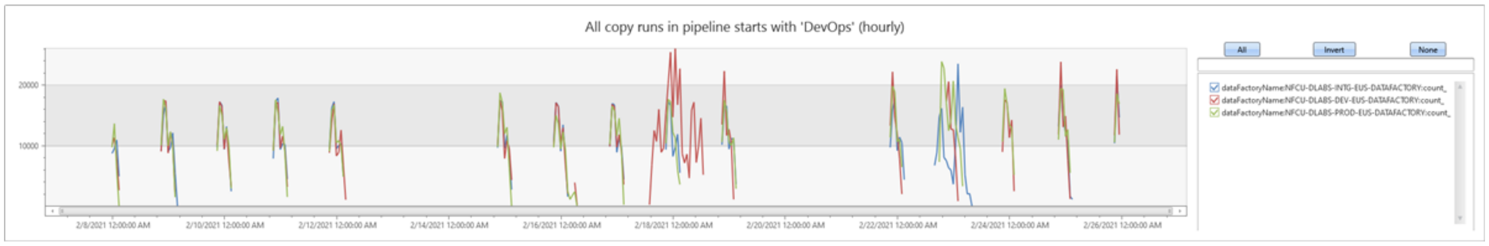
```
//Query to get the >4min hourly run chart:

ActivityRuns
| where TIMESTAMP > datetime(2021-02-08 00:00:00.0000000)
| where TIMESTAMP < datetime(2021-02-26 00:00:00.0000000)
| where location == "eastus"
| where dataFactoryName in ()
| where activityType == "" //copy
| where status in ("Succeeded", "Failed", "Cancelled")
| extend duration = end - start
| where duration > time(00:04:00) //time(00:02:30)
| where pipelineName startswith ""
| project TIMESTAMP, duration, dataFactoryName, pipelineName, pipelineRunId, activityName, activityRunId, star
| order by duration asc
| summarize count() by bin(TIMESTAMP, 1h), dataFactoryName
| render timechart with (title ="Copy runs exeeded 4min in pipeline starts with 'DevOps' (hourly)")
```



```
//Query to get the total hourly run chart:

ActivityRuns
| where TIMESTAMP > datetime(2021-02-08 00:00:00.0000000)
| where TIMESTAMP < datetime(2021-02-26 00:00:00.0000000)
| where location == "eastus"
| where dataFactoryName in ()
| where activityType == "" //copy
| where status in ("Succeeded", "Failed", "Cancelled")
| extend duration = end - start
//| where duration > time(00:04:00) //time(00:02:30)
| where pipelineName startswith ""
| project TIMESTAMP, duration, dataFactoryName, pipelineName, pipelineRunId, activityName, activityRunId, star
| order by duration asc
| summarize count() by bin(TIMESTAMP, 1h), dataFactoryName
| render timechart with (title ="All copy runs in pipeline starts with 'DevOps' (hourly)")
```



Additional Information:

- **Icm References:** 210383621, 207574522, 204308781, 201351950
- **Author:** rakatuko

- **Reviewer:** rakatuko
- **Keywords:**

How good have you found this content?