Beam Infrastructure & Dataflow Improvements - Load/Stress tests summary

PUBLIC DOCUMENT

Navigation

Useful information

Status: **in review**
Last updated: 2024-06-12

Doc Author(s): Akarys Shorabek (Akvelon), Vitaly Terentyev (Akvelon), Andrey Devyatkin (Akvelon)
Doc Reviewers and Contributors: Yi Hu
Project Contributors:

Discussion Log

Author	URL	Date

Reference documents:
[TDD] Beam Infrastructure & Dataflow Impro...

Overview

Context

Apache Beam is an open-source, unified model for defining parallel processing pipelines for batch and streaming data. It provides a portable API layer for building sophisticated data-parallel processing pipelines that may be executed across a diversity of execution engines, or runners.

This document outlines an overview of implemented Load and Stress tests for Apache Beam IOs to measure how a write operation behaves under load.

Load Tests

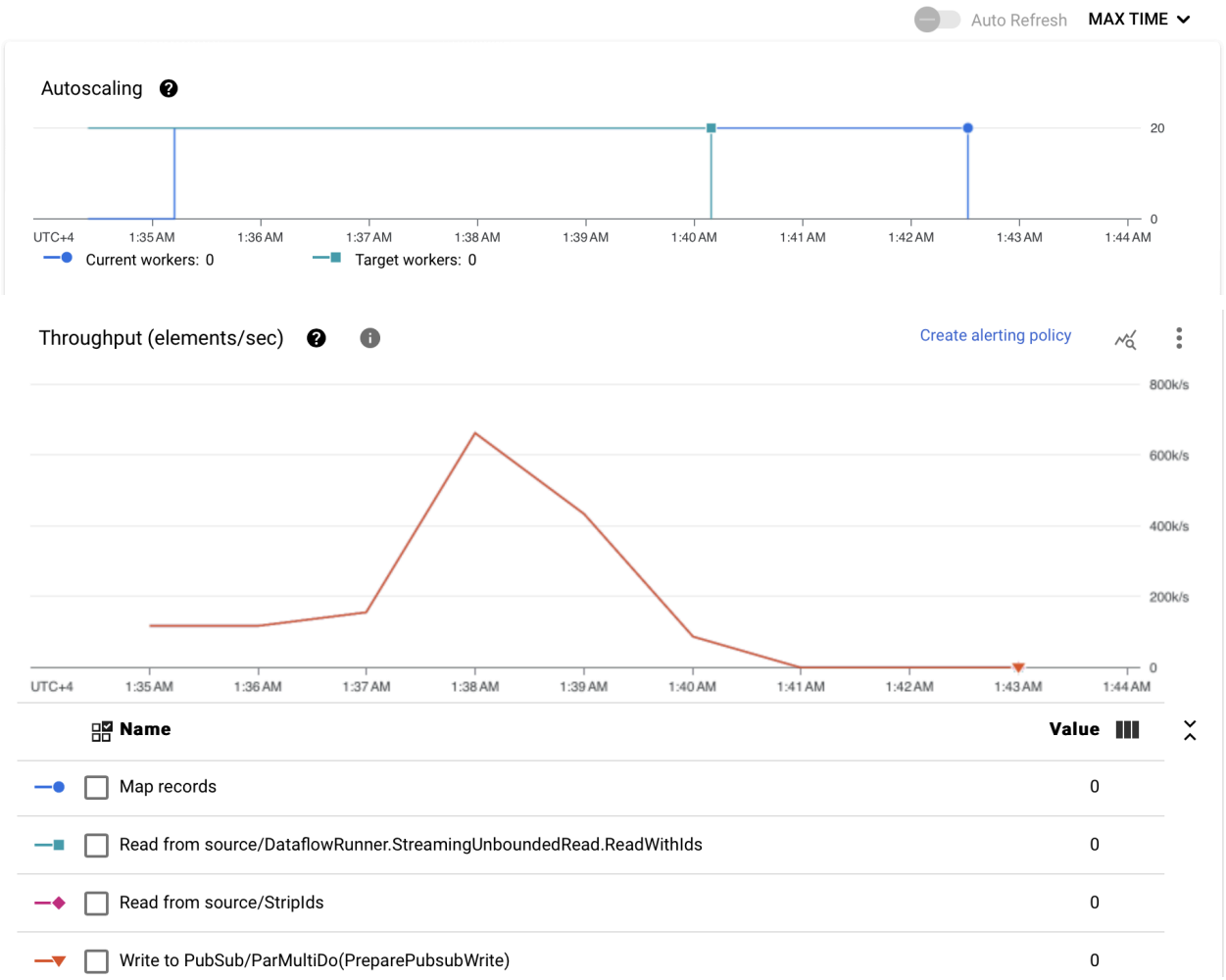
PubSubIO

Setup prerequisites:

- Number of workers: 20 (e2-standard-4)
- Average throughput: 500 MB/s
- Dataflow Runner: V1

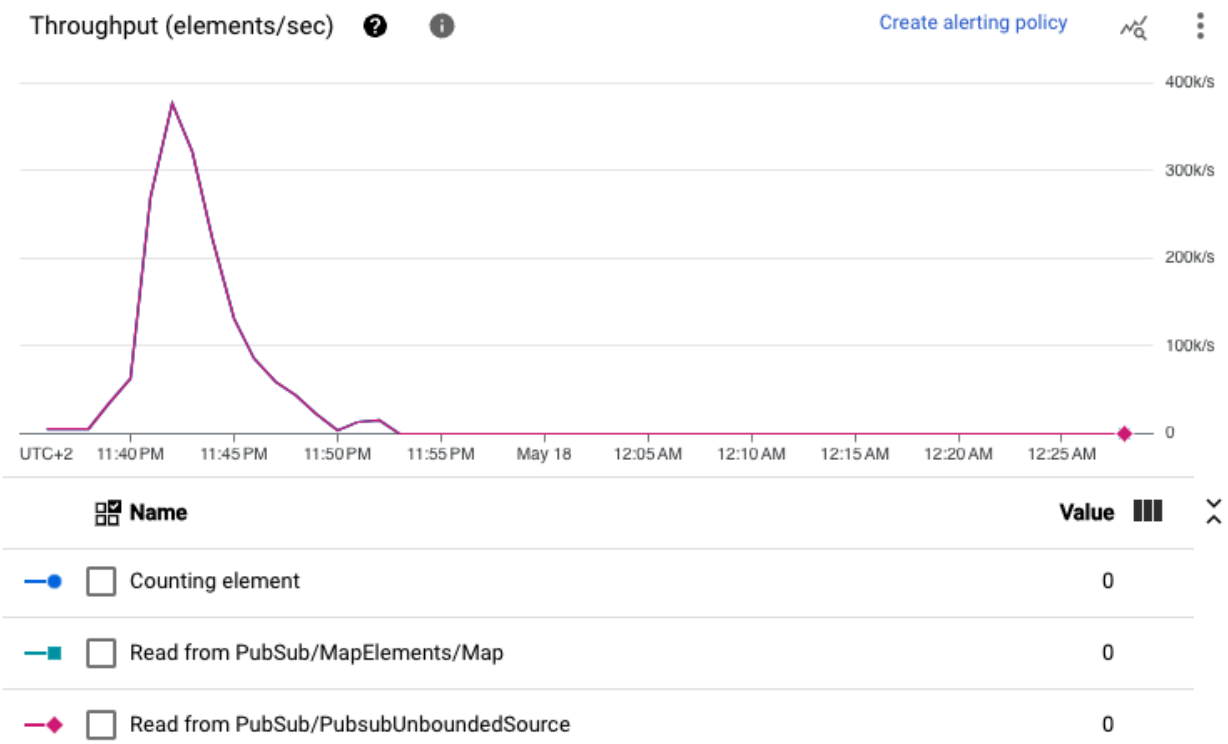
[Link to PR]
[Link to write job]

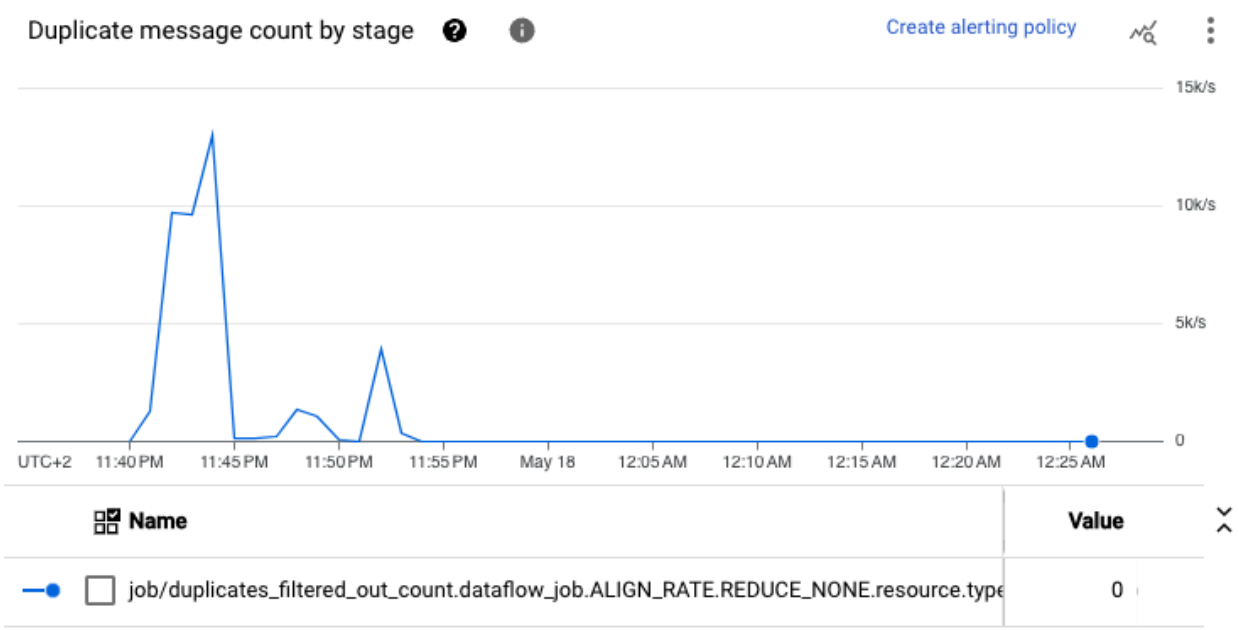
At the beginning of the test, throughput rises to a certain value (for example, 600 MB/s), and then decreases to zero by the end of the test. The number of workers remains unchanged.



During the test writing, a bug with SyntheticUnboundedSource was discovered where the number of lost records in the write pipeline equals the number of initial bundles in SyntheticUnboundedSource, which SyntheticUnboundedSource automatically sets depending on the number of records. Therefore, it was necessary to specify a certain value for initial bundles at the beginning of the test for convenience of checking.

A streaming pipeline is used to verify the read data. During the verification of large data, duplicates may appear.





Stress Tests

All stress tests were written to assess the capabilities of specific IOs in streaming mode. For verifying the written data, the following methods were used:

- A separate pipeline for reading in streaming mode.
- The capabilities of the ResourceManager of the specific IO to obtain the amount of written data directly if streaming mode was not supported for reading by the IO.
- A separate pipeline for reading in batch mode if the first two options were not supported.

Stress tests operate on the principle of dynamically increasing the load over time. Test results are exported to BigQuery or InfluxDB, depending on the configuration.

All tests use SyntheticUnboundedSource as a source for the write pipeline.

BigTableIO

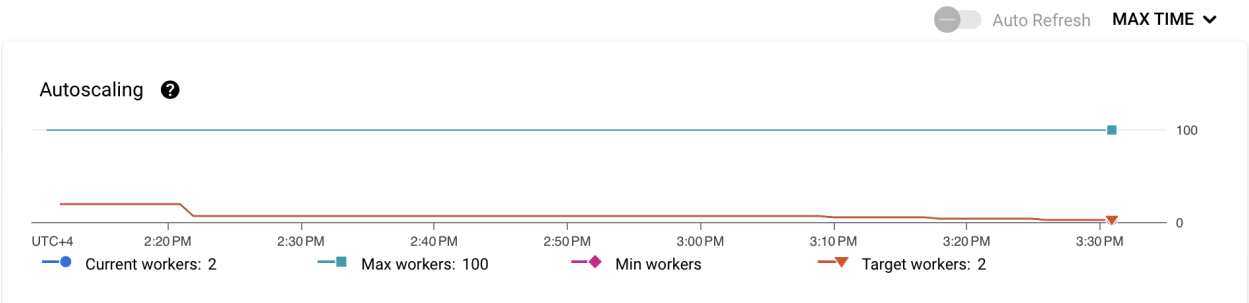
Setup prerequisites:

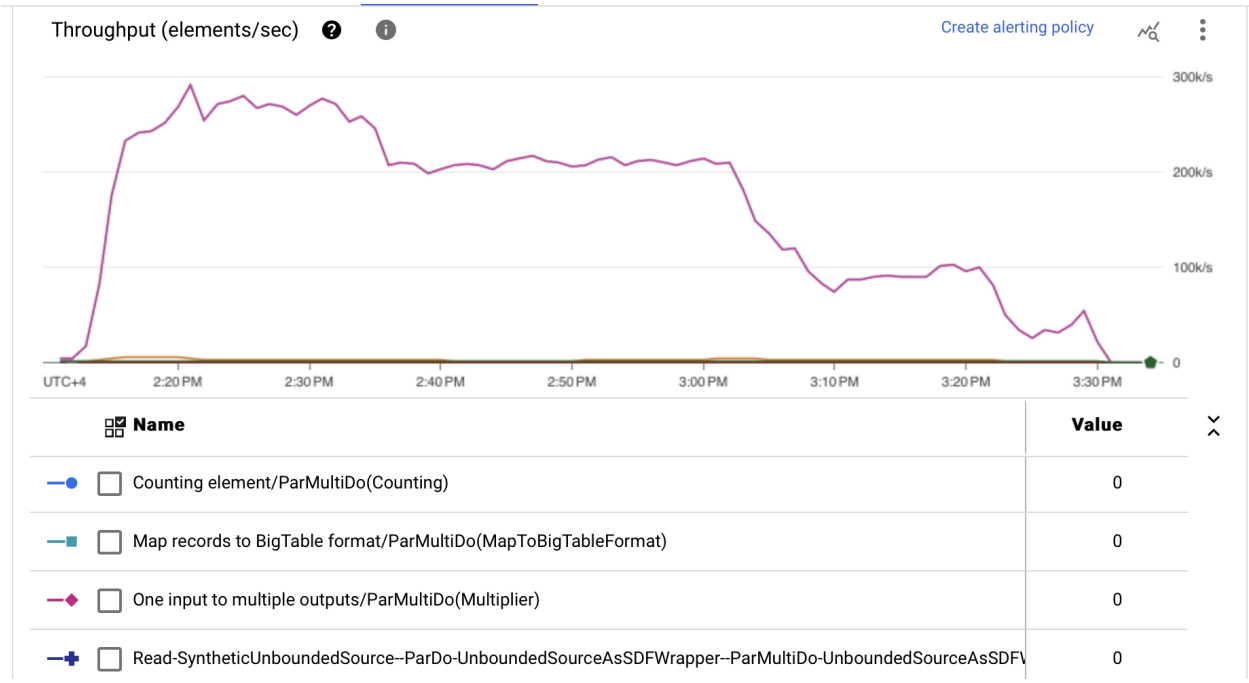
- Number of instance nodes: 10
- Number of workers: 7
- Average throughput: 170MB/s

- Dataflow Runner: V2

[Link to PR]
[Link to write job]

At the beginning of the test, the throughput rises to a certain value. It may then drop slightly, rise again, and remain stable at that level until the end of the test.





It was also found that one node of the instance could not handle the load, so it was increased to 10 nodes. Further increases in the number of nodes did not show a significant difference in efficiency. This suggests BigtableIO write has a hard throughput limit no matter how many resources are allocated.

The verification of the read records is conducted using a separate pipeline for reading in batch mode, as streaming is not supported. The amount of written data typically matches exactly with the read data.

When using PeriodicImpulse as a source, average throughput is lower than with SyntheticUnboundedSource by 3-5 times. At the beginning of the test, throughput increases to a certain value, then it rises by approximately 1.5 times where it remains for most of the time, and at the end, throughput decreases to the same value as at the beginning due to dynamic load increase.

BigQueryIO

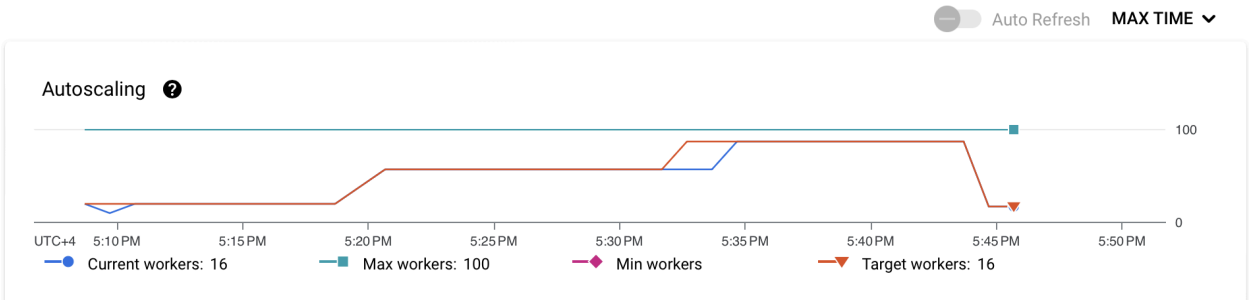
STORAGE_WRITE_API

Setup prerequisites:

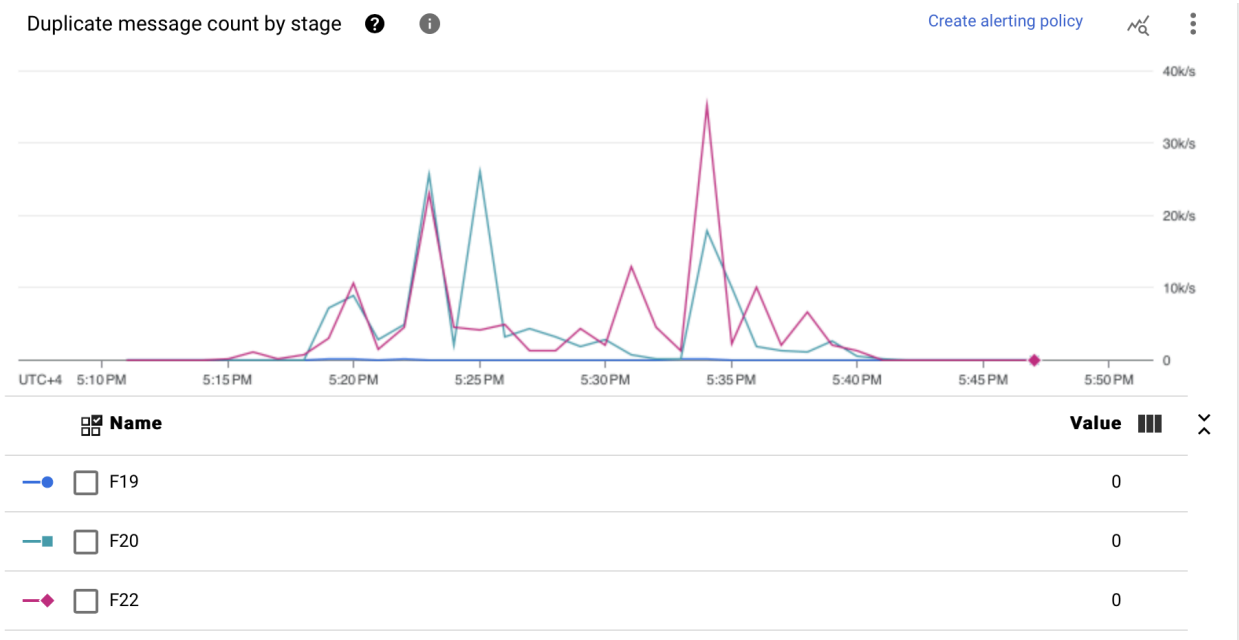
- Number of workers: 87
- Dataflow Runner: V1, Streaming Engine enabled
- Average throughput: 420 MB/s

[Link to PR]
[Link to write job]

For STORAGE_WRITE_API mode the throughput rises to a specific value (for example, 500 MB/s), stays there for some time, and then increases by 1.5 times, holding steady for a while before it continues to rise (up to 1M/sec) and gradually falls until the end of the test.



The number of workers during the test doubles. Duplicates may appear in a write pipeline with large volumes of data, especially during worker autoscaling.



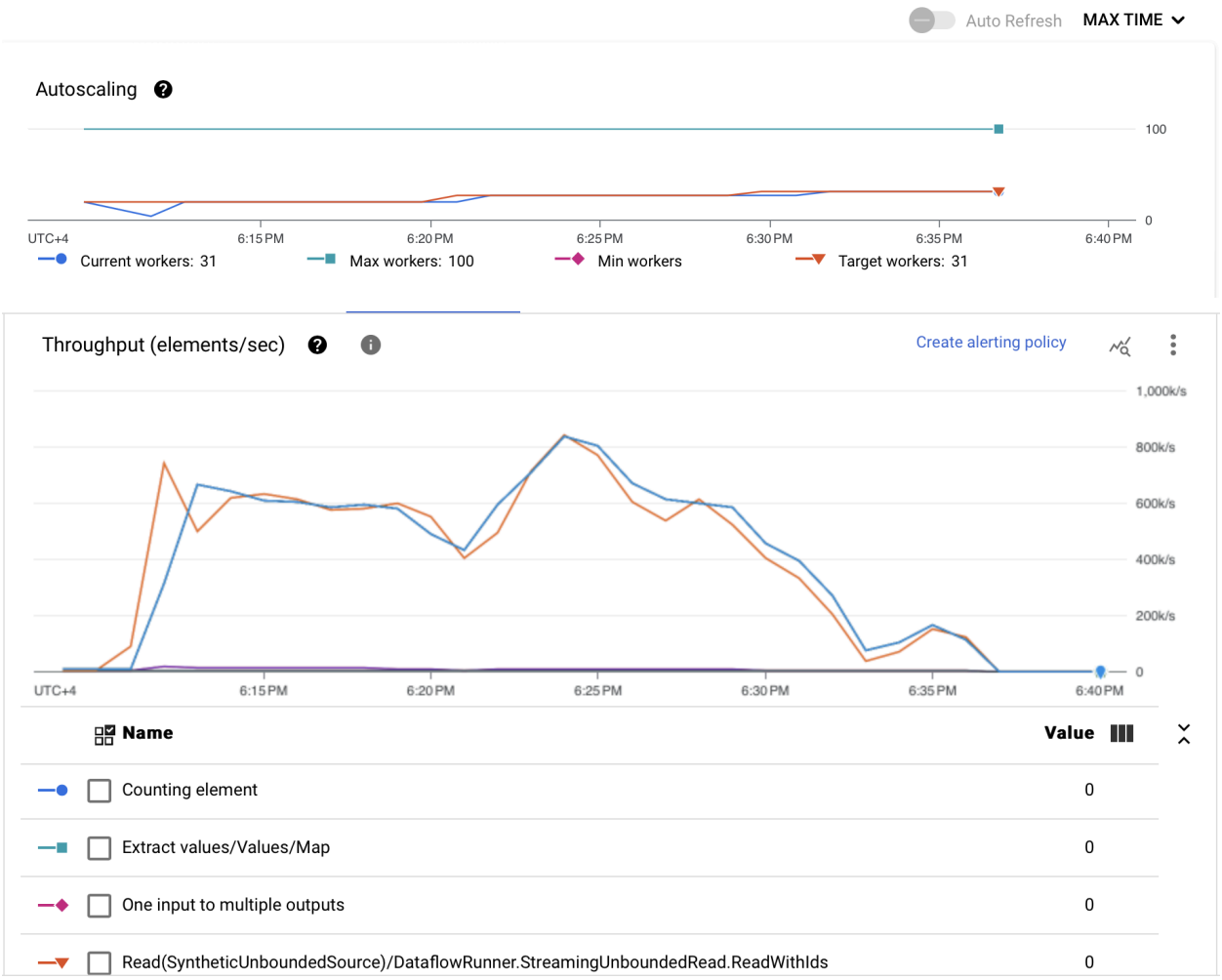
STORAGE_API_AT_LEAST_ONCE

Setup prerequisites:

- Number of workers: 31
- Dataflow Runner: V1, Streaming Engine enabled
- Average throughput: 470 MB/s

Link to PR
Link to write job

For the STORAGE_API_AT_LEAST_ONCE mode, the throughput rises to a certain level (for example, 600 MB/s) and remains there to the end.



Further increases in load (*rowsPerSecond*) do not result in much improvement in average throughput, and for the STORAGE_API_AT_LEAST_ONCE method, intermittently (in 1 out of 5 runs) the following error appears in the console: "*Error message from worker: java.lang.RuntimeException: More than 10 attempts to call AppendRows failed*". (Example job)

Data verification is conducted directly through the ResourceManager. The missing records issue was found during the test.

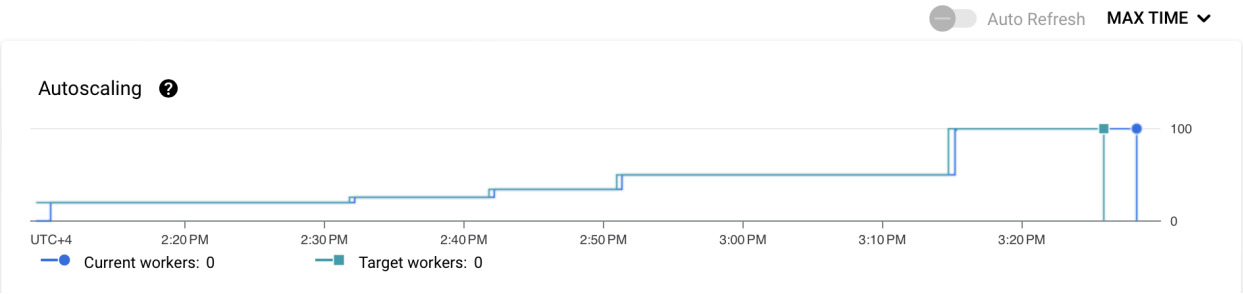
SpannerIO

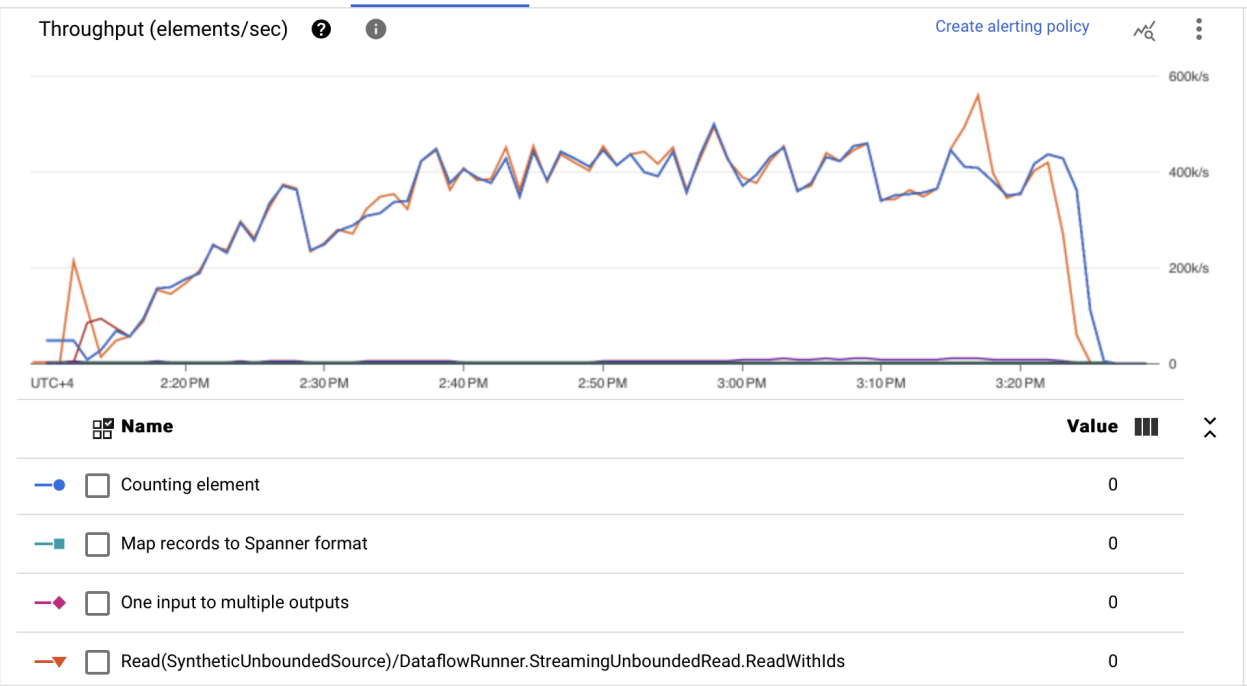
Setup prerequisites:

- Number of instance nodes: 30
- Number of workers: 100
- Average throughput: 330 MB/s
- Dataflow Runner: V1

Link to PR
Link to write job

The throughput consistently grows (for example, up to 400 MB/s) and falls depending on the dynamic increase in load. The number of workers goes up to 100 during the test. Duplicates may appear in a write pipeline when testing large volumes of data.



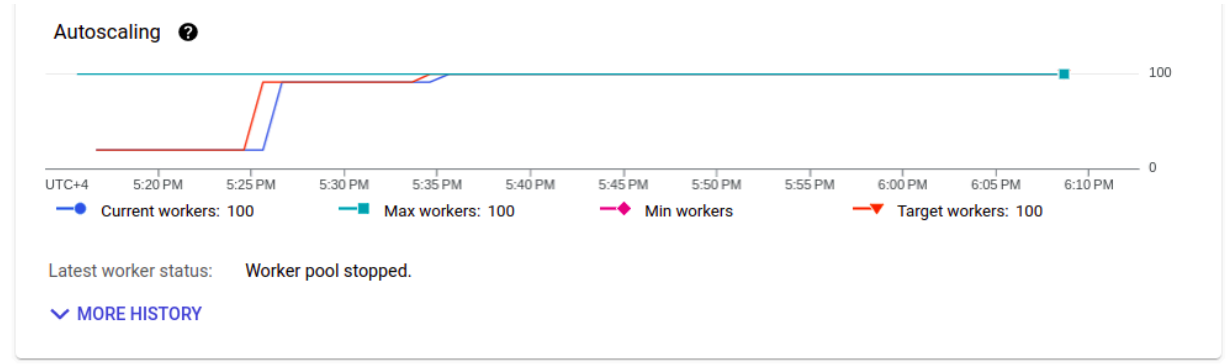


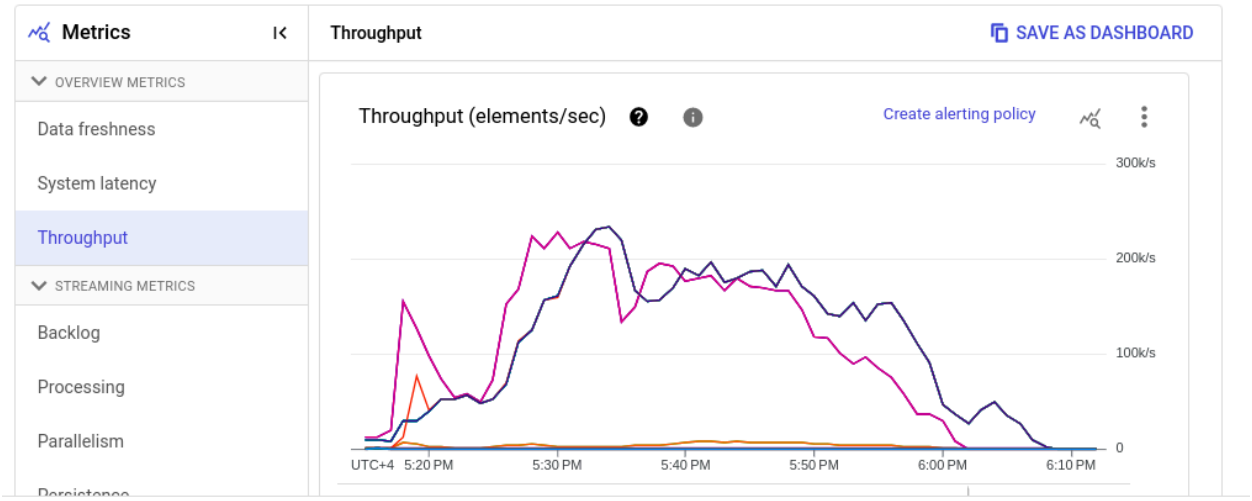
A batch read pipeline is used for verification.

During the test development, it was found that the default number of nodes of the Spanner instance (1 node) could not handle the load, so it was increased to 30 nodes, as there was no significant difference after reaching this number.

Additionally, it was discovered that using Dataflow runner_v2, the writing throughput drops by 2 times, so it was decided to use the legacy version:

- Number of instance nodes: 30
- Number of workers: 100
- Average throughput: 200 MB/s
- Dataflow Runner: V2





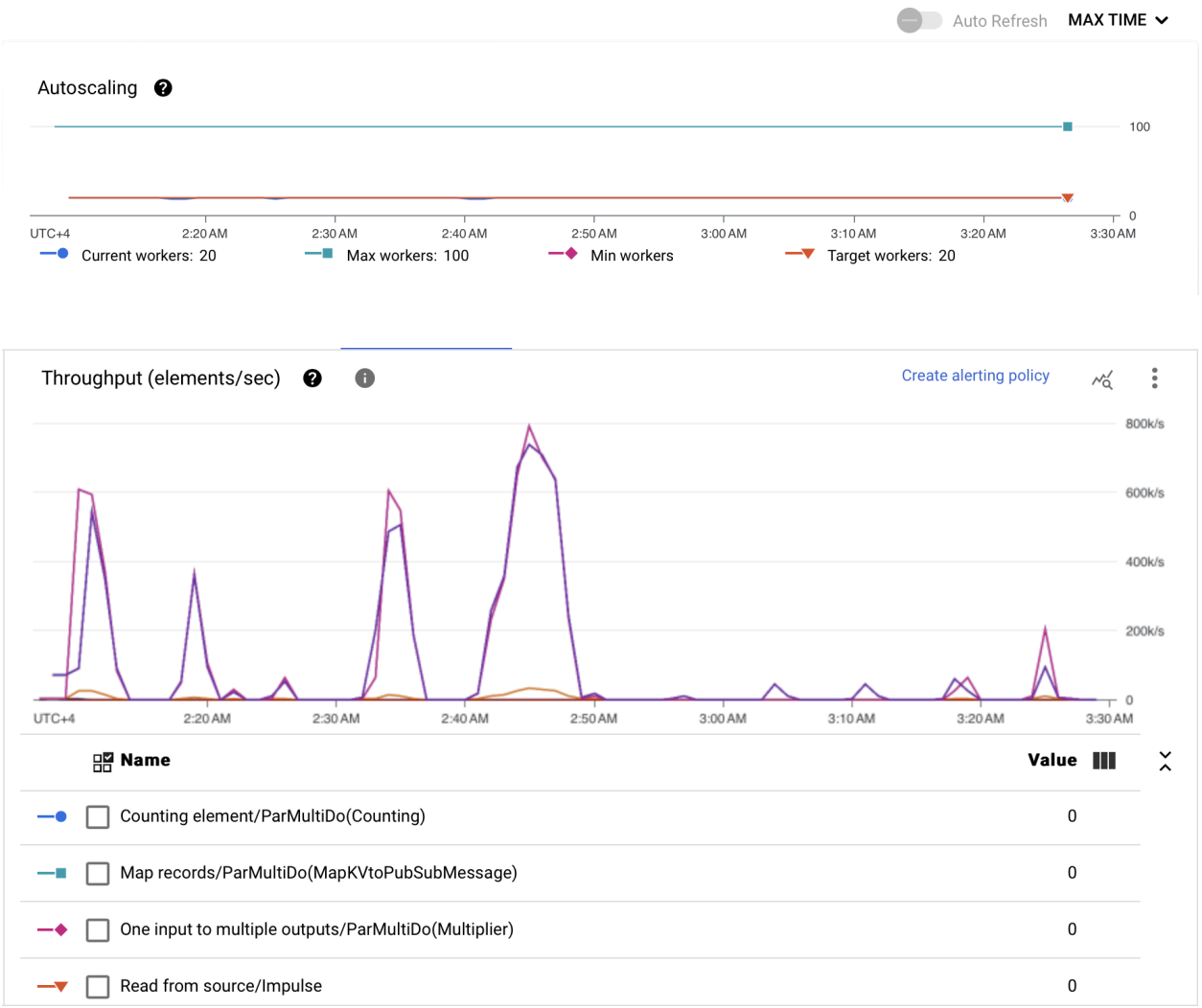
PubSubIO

Setup prerequisites:

- Number of workers: 20
- Average throughput: 300 MB/s
- Dataflow Runner: V2

Link to PR
Link to write job

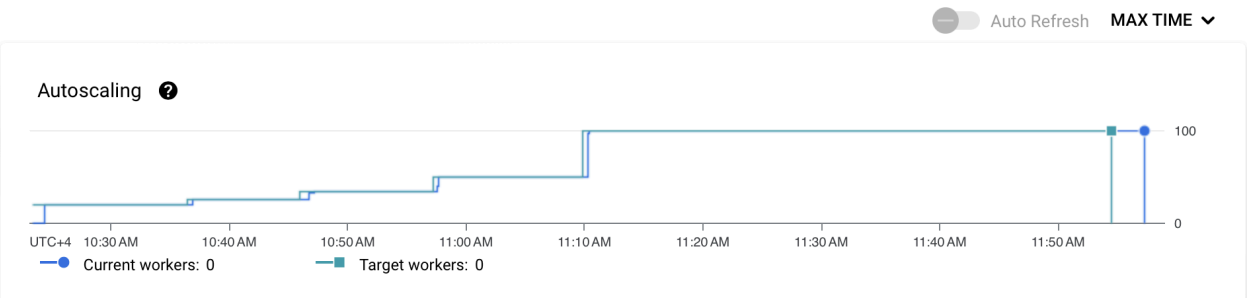
Typically, the throughput quickly rises to a high level (for example, 400 MB/s) and then rapidly drops to zero; this pattern repeats until the end of the test. The number of workers may occasionally decrease.

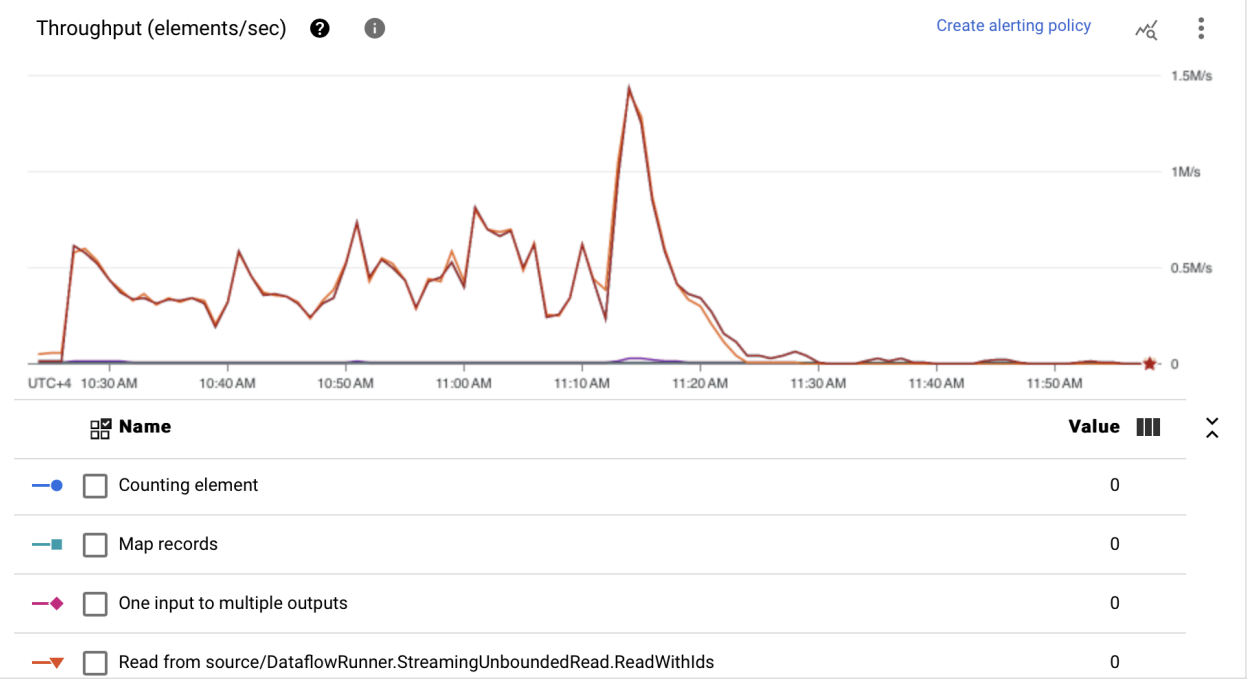


Duplicates may appear when testing large volumes of data. A streaming pipeline is used to verify the read data.

- Number of workers: 100
- Average throughput: 235 MB/s
- Dataflow Runner: V1

[Link to write job]





KafkaIO

Setup prerequisites:

- Number of kafka brokers: 3
- Number of workers: 3
- Average throughput: 200 MB/s
- Dataflow Runner: V2

Link to PR
Link to write job

When using PeriodicImpulse, the throughput remains stable for most of the test duration, although it's relatively low (for example, 30MB/s). However, when using

SyntheticUnboundedSource, the average throughput increases (for example, up to 200 MB/s). Yet, it is not always stable and can occasionally drop to a lower level.

To run a test with a large volume of data, it is currently necessary for the Kafka disk to have at least 500GB of space. Otherwise, the servers may crash, which could lead to errors in the test.

It has been found that testing with large volumes of data can result in duplicates.

Appendix

Technical Design Document -

[TDD] Beam Infrastructure & Dataflow Improvements - Performance, load, and stress test…