

[Google Discussions](#)

### Exam Professional Data Engineer All Questions

View all questions & answers for the Professional Data Engineer exam

[Go to Exam](#)

## EXAM PROFESSIONAL DATA ENGINEER TOPIC 1 QUESTION 178 DISCUSSION

Actual exam question from Google's Professional Data Engineer

Question #: 178

Topic #: 1

[\[All Professional Data Engineer Questions\]](#)

You are testing a Dataflow pipeline to ingest and transform text files. The files are compressed gzip, errors are written to a dead-letter queue, and you are using SideInputs to join data. You noticed that the pipeline is taking longer to complete than expected; what should you do to expedite the Dataflow job?

- A. Switch to compressed Avro files.
- B. Reduce the batch size.
- C. Retry records that throw an error.
- D. Use CoGroupByKey instead of the SideInput.

[Show Suggested Answer](#)

by [AWSandeep](#) at *Sept. 2, 2022, 9:02 p.m.*

### Comments

[Submit](#)

 [John\\_Pongthorn](#) [Highly Voted](#) 2 years, 1 month ago

**Selected Answer: D**


D: it is most likely.

There are a lot of reference doc to tell about comparison between them

[https://cloud.google.com/architecture/building-production-ready-data-pipelines-using-dataflow-developing-and-testing#choose\\_correctly\\_between\\_side\\_inputs\\_or\\_cogroupbykey\\_for\\_joins](https://cloud.google.com/architecture/building-production-ready-data-pipelines-using-dataflow-developing-and-testing#choose_correctly_between_side_inputs_or_cogroupbykey_for_joins)

<https://cloud.google.com/blog/products/data-analytics/guide-to-common-cloud-dataflow-use-case-patterns-part-2>

<https://stackoverflow.com/questions/58080383/sideinput-i-o-kills-performance>

   upvoted 16 times

  **zelck** **Highly Voted**  1 year, 11 months ago

**Selected Answer: D**

D is the answer.

[https://cloud.google.com/architecture/building-production-ready-data-pipelines-using-dataflow-developing-and-testing#choose\\_correctly\\_between\\_side\\_inputs\\_or\\_cogroupbykey\\_for\\_joins](https://cloud.google.com/architecture/building-production-ready-data-pipelines-using-dataflow-developing-and-testing#choose_correctly_between_side_inputs_or_cogroupbykey_for_joins)

The CoGroupByKey transform is a core Beam transform that merges (flattens) multiple PCollection objects and groups elements that have a common key. Unlike a side input, which makes the entire side input data available to each worker, CoGroupByKey performs a shuffle (grouping) operation to distribute data across workers. CoGroupByKey is therefore ideal when the PCollection objects you want to join are very large and don't fit into worker memory.

Use CoGroupByKey if you need to fetch a large proportion of a PCollection object that significantly exceeds worker memory.

   upvoted 11 times

  **MaxNRG** **Most Recent**  10 months, 2 weeks ago

**Selected Answer: D**

To expedite the Dataflow job that involves ingesting and transforming text files, especially if the pipeline is taking longer than expected, the most effective strategy would be:

D. Use CoGroupByKey instead of the SideInput.

   upvoted 1 times

  **MaxNRG** 10 months, 2 weeks ago

Here's why this approach is beneficial:

1. Efficiency in Handling Large Datasets: SideInputs are not optimal for large datasets because they require that the entire dataset be available to each worker. This can lead to performance bottlenecks, especially if the dataset is large. CoGroupByKey, on the other hand, is more efficient for joining large datasets because it groups elements by key and allows the pipeline to process each key-group separately.
2. Scalability: CoGroupByKey is more scalable than SideInputs for large-scale data processing. It distributes the workload more evenly across the Dataflow workers, which can significantly improve the performance of your pipeline.
3. Better Resource Utilization: By using CoGroupByKey, the Dataflow job can make better use of its resources, as it doesn't need to replicate the entire dataset to each worker. This results in faster processing times and better overall efficiency.

   upvoted 1 times

  **MaxNRG** 10 months, 2 weeks ago

The other options may not be as effective:

- A (Switch to compressed Avro files): While Avro is a good format for certain types of data processing, simply changing the file format from gzip to Avro may not address the underlying issue causing the delay, especially if the problem is related to the way data is being joined or processed.
- B (Reduce the batch size): Reducing the batch size could potentially increase overhead and might not significantly improve the processing time, especially if the bottleneck is due to the method of data joining.
- C (Retry records that throw an error): Retrying errors could be useful in certain contexts, but it's unlikely to speed up the pipeline if the delay is due to inefficiencies in data processing methods like the use of SideInputs.

   upvoted 1 times

  **musumusu** 1 year, 8 months ago

Answer: B,

reducing the batch size improve the speed performance also improve cpu utilisation.

Dead letter queues are generated for messages that are errorrly acknowledged and its good to use sideinputs for that to check small amount of errors in memory.

Cogroupbykey is not necessary for error messages.



I see only batch size that can be customized to improve the performance.

In npractical use case:

in practical use case.

you check these tools Stackdriver Monitoring and Logging, Cloud Trace, and Cloud Profiler. and try to find the cause, if its file type issue in compression, or batch size.



   upvoted 1 times

  **Atnafu** 1 year, 10 months ago

D

Flatten will just merge all results into a single PCollection. To join them you can use CoGroupByKey

   upvoted 1 times

  **TNT87** 2 years ago

**Selected Answer: A**

When optimizing for load speed, Avro file format is preferred. Avro is a binary row-based format which can be split and read in parallel by multiple slots including compressed files.

   upvoted 2 times

  **devaid** 2 years ago



that is for Big Query isn't?

   upvoted 1 times

  **TNT87** 1 year, 9 months ago

datflow can use avro format sir. streaming or batching to bigquery in avro format it can

   upvoted 1 times

  **TNT87** 2 years ago

<https://cloud.google.com/blog/topics/developers-practitioners/bigquery-explained-data-ingestion>

   upvoted 1 times

  **devaid** 2 years, 1 month ago

**Selected Answer: D**

D probably, side inputs have to fit in memory. If the p-collection in the side input doesn't fit well in memory it's better to use CoGroupByKey.

   upvoted 1 times

  **TNT87** 2 years, 1 month ago

**Selected Answer: A**

Answer A



the same question is in number 70 you transform the files to Avro using Dataflow

   upvoted 1 times

  **KC\_go\_reply** 1 year, 4 months ago

Avro requires the data to be at least semi-structured, because it wants a fixed schema. Text files are unstructured data, therefore it doesn't make sense to use Avro files for them

   upvoted 3 times

  **csd1fggfghgvh234** 2 years, 1 month ago

A switching to avro. No serialisation

   upvoted 1 times

  **TNT87** 2 years, 1 month ago

Switch to Avro format

Answer A

   upvoted 2 times

  **TNT87** 2 years, 1 month ago

<https://docs.confluent.io/platform/current/schema-registry/serdes-develop/serdes-avro.html>

   upvoted 1 times

  **YorelNation** 2 years, 2 months ago

**Selected Answer: D**

D probably, side inputs have to fit in memory. If the p-collection in the side input doesn't fit well in memory it's better to use CoGroupByKey.

   upvoted 3 times

  **AWSandeep** 2 years, 2 months ago

**Selected Answer: B**

B. Reduce the batch size.

   upvoted 4 times



## Platform

> [Home](#)

> [Examtopics PRO](#)

> [All Exams](#)

> [Training Courses](#)

