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 181 DISCUSSION

Actual exam question from Google's Professional Data Engineer

Question #: 181

Topic #: 1

[All Professional Data Engineer Questions]

You need to give new website users a globally unique identifier (GUID) using a service that takes in data points and returns a GUID. This data is sourced from both internal and external systems via HTTP calls that you will make via microservices within your pipeline. There will be tens of thousands of messages per second and that can be multi-threaded. and you worry about the backpressure on the system. How should you design your pipeline to minimize that backpressure?

A. Call out to the service via HTTP.

B. Create the pipeline statically in the class definition.

C. Create a new object in the startBundle method of DoFn.

D. Batch the job into ten-second increments.

Show Suggested Answer

by ducc at *Sept. 4, 2022, 3:19 a.m.*

Comments

Type your comment...

Submit

John_Pongthorn Highly Voted 2 years, 1 month ago

Selected Answer: D

D: I have insisted on this choice all aling.
please read find the keyword massive backpressure
https://cloud.google.com/blog/products/data-analytics/guide-to-common-cloud-dataflow-use-case-patterns-part-1

if the call takes on average 1 sec, that would cause massive backpressure on the pipeline. In these circumstances you should consider batching these requests, instead.

upvoted 20 times

NicolasN 1 year, 12 months ago

Thanks for sharing, you found exactly the same problem!
The document defitely proposes batching for this scenario.

I'm quoting another part from the same example that would be useful for a similar question with different conditions:
- If you're using a client in the DoFn that has heavy instantiation steps, rather than create that object in each DoFn call:
* If the client is thread-safe and serializable, create it statically in the class definition of the DoFn.
* If it's not thread-safe, create a new object in the startBundle method of DoFn. By doing so, the client will be reused across all elements of a bundle, saving initialization time.

upvoted 6 times

Atnafu 1 year, 11 months ago

By the way if you see the shared Pseudocode, it's talking about start bundle and finish bundle of DoFn. The question is which one to choose to avoid back pressure?
you can see why you need to choose bundle instead of batching in below link
Batching introduces some processing overhead as well as the need for a magic number to determine the key space.
Instead, use the StartBundle and FinishBundle lifecycle elements to batch your data. With these options, no shuffling is needed.
https://cloud.google.com/dataflow/docs/tutorials/ecommerce-java#micro-batch-calls

upvoted 2 times

NicolasN 1 year, 11 months ago

Valid points. but I don't change my mind, regarding the requirements of this particular question:
- multi-threaded ability
- no mention of heavy initialization steps or a lot of disk I/O (where shuffling might be a problem).
And especially the excerpt:
"if the call takes on average 1 sec, that would cause massive backpressure on the pipeline. In these circumstances you should consider batching these requests, instead"
It's like the guys that authored the question had this sentence in front of their eyes.

upvoted 2 times

John_Pongthorn Highly Voted 2 years, 1 month ago

Selected Answer: D

D
All guys ,pls read carefully on Pattern: Calling external services for data enrichment
https://cloud.google.com/blog/products/data-analytics/guide-to-common-cloud-dataflow-use-case-patterns-part-1
A , B , C all of them are solution for norma case but if you need to stand for backpressure,
in last sector in Note : Note: When using this pattern, be sure to plan for the load that's placed on the external service and any associated backpressure. For example, imagine a pipeline that's processing tens of thousands of messages per second in steady state. If you made a callout per element, you would need the system to deal with the same number of API calls per second. Also, if the call takes on average 1 sec, that would cause massive backpressure on the pipeline. In these circumstances, you should consider batching these requests, instead.

Anyone can share ideas to debate with me.

upvoted 8 times

MaxNRG Most Recent 10 months, 2 weeks ago

Selected Answer: D

Option D is the best approach to minimize backpressure in this scenario. By batching the jobs into 10-second increments, you can throttle the rate at which requests are made to the external GUID service. This prevents too many simultaneous requests from overloading the service.

Option A would not help with backpressure since it just makes synchronous HTTP requests as messages arrive. Similarly, options B and C don't provide any inherent batching or throttling mechanism.

Batching into time windows is a common strategy in stream processing to deal with high velocity data. The 10-second windows allow some buffering to happen, rather than making a call immediately for each message. This provides a natural throttling that can be tuned based on the external service's capacity.

upvoted 2 times

MaxNRG 10 months, 2 weeks ago

To design a pipeline that minimizes backpressure, especially when dealing with tens of thousands of messages per second in a multi-threaded environment, it's important to consider how each option affects system performance and scalability. Let's examine each of your options:

upvoted 1 times

MaxNRG 10 months, 2 weeks ago

A. Call out to the service via HTTP: Making HTTP calls to an external service for each message can introduce significant latency and backpressure, especially at high throughput. This is due to the overhead of establishing a connection, waiting for the response, and handling potential network delays or failures.

upvoted 1 times

MaxNRG 10 months, 2 weeks ago

Given these considerations, option D (Batch the job into ten-second increments) seems to be the most effective strategy for minimizing backpressure in your scenario. By batching messages, you can reduce the strain on your pipeline and external services, making the system more resilient and scalable under high load. However, the exact batch size and interval should be fine-tuned based on the specific characteristics of your workload and the capabilities of the external systems you are interacting with.

Additionally, it's important to consider other strategies in conjunction with batching, such as implementing efficient error handling, load balancing, and potentially using asynchronous I/O for external HTTP calls to further optimize performance and minimize backpressure.

upvoted 1 times

MaxNRG 10 months, 2 weeks ago

B. Create the pipeline statically in the class definition: While this approach can improve initialization time and reduce overhead during execution, it doesn't directly address the issue of backpressure caused by high message throughput.

upvoted 1 times

MaxNRG 10 months, 2 weeks ago

C. Create a new object in the startBundle method of DoFn: This approach is typically used in Apache Beam to initialize resources before processing a bundle of elements. While it can optimize resource usage and performance within each bundle, it doesn't inherently solve the backpressure issue caused by high message rates.

upvoted 1 times

MaxNRG 10 months, 2 weeks ago

D. Batch the job into ten-second increments: Batching messages can be an effective way to reduce backpressure. By grouping multiple messages into larger batches, you can reduce the frequency of external calls and distribute the processing load more evenly over time. This can lead to more efficient use of resources and potentially lower latency, as the system spends less time waiting on external services.

upvoted 1 times

izekc 1 year, 6 months ago

Selected Answer: D

Option C is not correct because it does not address the issue of backpressure. Creating a new object in the startBundle method of DoFn will not help to reduce the number of calls that are made to the service, which can lead to backpressure.

Here are some reasons why C is not correct:

Creating a new object in the startBundle method of DoFn is not a scalable solution. As the number of messages increases, the number of objects that need to be created will also increase. This can lead to performance problems and memory usage issues.
Creating a new object in the startBundle method of DoFn does not address the issue of backpressure. The service may still experience backpressure if the number of messages exceeds the service's capacity.
A better solution would be to use batching to reduce the number of calls that are made to the service. This can help to improve performance and reduce backpressure.

upvoted 1 times

Oleksandr0501 1 year, 6 months ago

gpt: Option C is a better approach as it allows for object creation to occur in a more controlled manner within the DoFn, potentially reducing the pressure on the system. However, it could still create a large number of objects depending on the rate of incoming messages.

Option D of batching the job into ten-second increments can also be a good solution to reduce backpressure on the system. This way, you can limit the number of messages being processed at any given time, which can help prevent bottlenecks and reduce the likelihood of backpressure.

Therefore, the best approach would be to combine options C and D, creating a new object in the startBundle method of a DoFn, and batching the job into smaller time increments, such as 10 seconds. This way, you can control the rate of object creation and processing, which can help minimize backpressure on the system.

upvoted 1 times

Oleksandr0501 1 year, 6 months ago

another vague question, as we see...
so, i`ll choose D... if i get this test
"However, depending on the specifics of your use case, one option may be better suited than the other. For example, if you have a high volume of incoming messages with occasional spikes, option D of batching the job into smaller time increments may be more effective in managing the load. On the other hand, if the incoming messages are more evenly distributed over time, option C of creating a new object in the startBundle method of DoFn may be a better option. Ultimately, it may be necessary to experiment with both approaches and determine which one works best for your specific use case."

upvoted 1 times

juliobs 1 year, 7 months ago

Selected Answer: D

D works.
Could be C, but who said that the pipeline is in Dataflow/Beam?

upvoted 1 times

musumusu 1 year, 8 months ago

Answer C
batch increment in 10 sec, can improve load balancing, but overall back pressure (messages are generating more than consuming or publishing) in this case startBundle in DoFn or find other options in future like
caching,
load shedding (prioritising message flow),
messege queuing
These options handle backpressure..
If your cpu is performing bad then go with change in batch increment timing

upvoted 1 times

maci_f 1 year, 9 months ago

Selected Answer: D

I was hesitating between C and D, but then I realised this: https://cloud.google.com/blog/products/data-analytics/guide-to-common-cloud-dataflow-use-case-patterns-part-1
Here is says "If it's not thread-safe, create a new object in the startBundle method of DoFn." The task explicitly says "There will be tens of thousands of messages per second and that can be multi-threaded."
Correct me if I'm wrong, but multi-threaded == thread-safe. Therefore, no need to go for the C approach.

upvoted 3 times

zellck 1 year, 11 months ago

Selected Answer: D

D is the answer.

https://cloud.google.com/blog/products/data-analytics/guide-to-common-cloud-dataflow-use-case-patterns-part-1
For example, imagine a pipeline that's processing tens of thousands of messages per second in steady state. If you made a callout per element, you would need the system to deal with the same number of API calls per second. Also, if the call takes on average 1 sec, that would cause massive backpressure on the pipeline. In these circumstances you should consider batching these requests, instead.

upvoted 3 times

AzureDP900 1 year, 10 months ago

D. Batch the job into ten-second increments.

upvoted 1 times

Atnafu 1 year, 11 months ago

C
C is an answer because
First of all, no doubt that we should avoid single call of element that's why we use multi-threading else it overwhelm an external service endpoint.To avoid this issue, batch calls to external systems.
Batch calls has also issue:GroupByKey transform or Apache Beam Timer API.
these approaches both require shuffling, which introduces some processing overhead as well as the need for a magic number to determine the key space.
Instead, use the StartBundle and FinishBundle lifecycle elements to batch your data. With these options, no shuffling is needed.
Source:
https://cloud.google.com/dataflow/docs/tutorials/ecommerce-java#micro-batch-calls
https://cloud.google.com/blog/products/data-analytics/guide-to-common-cloud-dataflow-use-case-patterns-part-1
Summary:
StartBundle and FinishBundle do batch with no shuffling

upvoted 1 times

AHUI 2 years, 1 month ago

Ans C: reference https://cloud.google.com/architecture/e-commerce/patterns/batching-external-calls

upvoted 1 times

SMASL 2 years, 1 month ago

Selected Answer: C

Based on the answers in this discussion thread, I would go for C. The most important link to support this choice is as following: https://cloud.google.com/architecture/e-commerce/patterns/batching-external-calls

upvoted 2 times

Thobm 2 years, 1 month ago

Selected Answer: D

Beam docs recommend batching
https://beam.apache.org/documentation/patterns/grouping-elements-for-efficient-external-service-calls/

upvoted 1 times

John_Pongthorn 2 years, 1 month ago

C , It is straight foward , You can take a look at https://cloud.google.com/blog/products/data-analytics/guide-to-common-cloud-dataflow-use-case-patterns-part-1
Pattern : Calling external services for data enrichmen

upvoted 1 times

TNT87 2 years, 1 month ago

Answer C
https://cloud.google.com/blog/products/data-analytics/guide-to-common-cloud-dataflow-use-case-patterns-part-1

upvoted 1 times

> **TNT87** 2 years, 1 month ago
>
> https://cloud.google.com/architecture/e-commerce/patterns/batching-external-calls
> To support choice C
>
> upvoted 1 times

YorelNation 2 years, 2 months ago

Selected Answer: C

i think you are right gg

upvoted 1 times

nwk 2 years, 2 months ago

How about C?
https://cloud.google.com/architecture/e-commerce/patterns/batching-external-calls

upvoted 2 times

Load full discussion...