11/25/2023, 8:12:11 AM

1/27

In this presentation we will talk about:

1. todo
2. todo

2/27

Comment here

3/27 What is Cloud Tasks?

Alternatives:

- BullMQ
- pg-boss
- Celery

https://www.draconianoverlord.com/2014/01/27/using-your-database-as-a-queue.html/
https://codeopinion.com/using-your-database-as-a-queue/ https://github.com/litements/litequeue

Example: send an email with sendgrid https://cloud.google.com/tasks/docs/samples/cloud-tasks-fun
https://cloud.google.com/tasks/docs/tutorial-gcf

Example: avoid HTTP 429 Too Many Requests https://cloud.google.com/workflows/docs/tutorials/buffer-workflows-executions

Dispatch flow control

(truncated) exponential backoff is a strategy we can use to schedule retries.
https://en.wikipedia.org/wiki/Exponential_backoff#Truncated_exponential_backoff

quotas and limits https://cloud.google.com/tasks/docs/quotas

https://cloud.google.com/tasks/pricing

Cloud Tasks runs in the App Engine internal infrastructure

PULL queus are just for backwards compatibility with the App Engine Task Queue SDK.
https://cloud.google.com/tasks/docs/reference/rest/v2beta3/projects.locations.queues.tasks#pullmessage

---

4/27 Queues in theory

https://cloud.google.com/tasks/docs/common-pitfalls#execution_order

> *Cloud Tasks aims for a strict "execute exactly once" semantic. However, in situations where a design trade-off must be made between guaranteed execution and duplicate execution, the service errs on the side of guaranteed execution. As such, a non-zero number of duplicate executions do occur.*

6/27 Buffering requests to comply with rate & concurrency limits from 3rd party APIs

Examples of a Cloud Tasks queue used to avoid rate limits:

- https://cloud.google.com/tasks/docs/tutorial-workflows
- https://towardsdatascience.com/gcp-serverless-design-pattern-adhering-to-rate-concurrency-limits-with-cloud-tasks-30aa756da763

https://tomorrowio.zendesk.com/hc/en-us/articles/5187600335124-What-is-the-difference-between-rate-limiting-and-concurrent-API-calls

---

8/27 Queues

Use the Cloud Tasks API to create/manage a queue. There is also the queue.yaml file, but I think it's a legacy method. It is strongly recommended that you use either the configuration file method or the Cloud Tasks API to configure your queues, but not both. https://cloud.google.com/tasks/docs/queue-yaml

https://cloud.google.com/tasks/docs/configuring-queues

https://www.pulumi.com/registry/packages/gcp/api-docs/cloudtasks/

`taskTtl` is the maximum amount of time that a task will be retained in this queue. After a task has lived for taskTtl, the task will be deleted regardless of whether it was dispatched or not. The minimum value is 10 days. The maximum value is 10 years. Queues created by Cloud Tasks have a default taskTtl of 31 days. https://cloud.google.com/tasks/docs/reference/rest/v2beta3/projects.locations.queues#resource:-queue

---

9/27 Tasks

https://cloud.google.com/tasks/docs/reference/rest/v2beta3/projects.locations.queues.tasks/create

https://cloud.google.com/tasks/docs/reference/rest/v2beta3/OidcToken

https://cloud.google.com/tasks/docs/reference/rest/v2beta3/OAuthToken

To create an App Engine task, replace httpRequest with appEngineHttpRequest.

10/27 IAM

https://cloud.google.com/tasks/docs/reference-access-control

To be precise, we need the IAM permission `cloudtasks.tasks.create`. We can obtain this permission by assigning the IAM role `roles/cloudtasks.enqueuer`.

You can create an IAM binding:

- at the GCP project level. It grants the service account permissions on ALL Cloud Tasks queues. With Pulumi this seems NOT to work.
- at the Cloud Tasks queue level. It grants the service account permissions on JUST THAT queue. With Pulumi this works and can be done using `gcp.cloudtasks.QueueIamBinding`

The first time I used Cloud Tasks I messed up because I changed the service account used by Cloud Tasks.

The IAM role `roles/cloudtasks.enqueuer` does NOT grant permission to run tasks. Just to enqueue them. To run tasks we need `roles/cloudtasks.taskRunner`.

Also, restrict access to your Cloud Tasks queues. https://cloud.google.com/tasks/docs/secure-queue-configuration

---

11/27 A basic example

Timeouts: for all HTTP Target task handlers the default timeout is 10 minutes, with a maximum of 30 minutes. https://cloud.google.com/tasks/docs/creating-http-target-tasks

https://stackoverflow.com/questions/58530361/how-increase-maximum-schedule-time-in-gcloud-tasks-api

https://cloud.google.com/tasks/docs/reference/rpc/google.cloud.tasks.v2#google.cloud.tasks.v2.Task.FIELD

12/27 A more advanced example

Consider making a ZenUML diagram https://mermaid.js.org/syntax/zenuml.html

---

13/27 Notifications with SSE

SSE data stream is UTF-8 encoded. The SSE server needs to send this HTTP response header: Content-Type: text/event-stream

With CPU always allocated you obviously pay more. https://cloud.google.com/run/docs/configuring/cpu-allocation I'm not entirely sure whether "CPU idle" means the exact same thing as "CPU throttled".

We often use HTTP/1.1 in a Cloud Run service, since the HTTP/2 connection ends at the Google Frontend level. Our Cloud Run service is not directly deployed on the internet, but it's always behind GFE. Any internal service that must publish itself externally uses the GFE as a smart reverse-proxy frontend. The GFE provides public IP address hosting of its public DNS name, DoS protection, and TLS termination. https://cloud.google.com/docs/security/infrastructure/design#google-frontend-service

---

14/27 Notifications with WebSockets

Websockets with Cloud Run https://youtu.be/g6i-mb_3iWM?si=SN7x2Yyh10WN76C9

WebSocket server on Compute Engine https://cloud.google.com/pubsub/docs/streaming-cloud-pub-sub-messages-over-websockets

---

17/27 Make your tasks easy to identify

Cloud Tasks assigns each task a name automatically. But this names are UUIDs which are meaningless for you. You open the Cloud Tasks dashboard and don't understand who enqueued a task and what the task is about.

We recommend using a well-distributed prefix for task names, such as a hash of the contents.

These costs can be magnified significantly if tasks are named sequentially, such as with timestamps. I think this is similar to the index hotspotting problem of databases (see key visualizer in BigTable or Firestore).

18/27 Implement idempotent task handlers

PUT vs POST: PUT is idempotent, POST is not. https://restcookbook.com/HTTP%20Methods/put-vs-post/

PUT vs PATCH: PUT is idempotent, PATCH is not. PUT always requires the entire request payload. PATCH allows a subset of the payload.

20/27 Test your system

draconianoverlord usa il termine Cross-System Integration Testing perche' secondo lui ci sono:

- **Intra-system integration tests**. Si tratta di test di integrazione fra componenti di cui te hai il controllo. Ad esempio il tuo JS frontend, la tua API, il tuo database layer. **Questi tests secondo lui hanno senso** perche' te **controlli tutto l'environment**.
- **Inter-/cross-system integration tests**. Si tratta di test di integrazione fra una o piu' componenti di cui te hai il controllo (vedi sopra) e una o piu' componenti di un qualche vendor. Ad esempio quando te scrivi un API client per una API **non** tua, gli eventuali integration tests che scrivi sarebbero **cross-system** integration tests. **Questi tests secondo lui non hanno senso**. Ecco, ma questi direi che sono quelli che normalmente vengono chiamati system tests.

21/27 Cloud Tasks metrics

I think queue/depth in Cloud Tasks is similar to subscription/num_undelivered_messages (i.e. the message backlog) in Cloud Pub/Sub. But Pub/Sub allows N subscribers to a topic. Cloud Tasks only allows 1 Target for a Queue. https://cloud.google.com/pubsub/docs/monitoring#monitoring_the_backlog

---

22/27 Alert: queue overload

https://cloud.google.com/tasks/docs/manage-cloud-task-scaling#queue

https://cloud.google.com/tasks/docs/manage-cloud-task-scaling#target

Always define your alerting policies in JSON, keep them under version control and document them (e.g. in markdown) https://cloud.google.com/monitoring/alerts/policies-in-json

---

24/27 Cloud Tasks vs Cloud Pub/Sub (1/2)

https://cloud.google.com/tasks/docs/comp-pub-sub

Cloud Tasks supports only (?) push queues. PubSub supports pull queues AND push queues.

Does your service know which service to call?

- If yes, this is an explicit invocation => choose a push queue
- If no, this is an implicit invocation => choose a pull queue

Cloud Tasks is aimed at explicit invocation where the publisher retains full control of execution. In particular, a publisher specifies an endpoint where each message is to be delivered.

Pub/Sub aims to decouple publishers of events and subscribers to those events. Publishers do not need to know anything about their subscribers. Therefore, Pub/Sub gives publishers no control over the delivery of the messages save for the guarantee of delivery. In this way, Pub/Sub supports implicit invocation: a publisher implicitly causes the subscribers to execute by publishing an event.

There is also Pub/Sub Lite, which supports only pull queues.

In Cloud Tasks, enqueued task order is preserved on best-effort basis

https://cloud.google.com/pubsub/docs/choosing-pubsub-or-cloud-tasks

---

25/27 Cloud Tasks vs Cloud Pub/Sub (2/2)

https://cloud.google.com/tasks/docs/comp-pub-sub

https://cloud.google.com/pubsub/docs/choosing-pubsub-or-cloud-tasks

There is no equivalent of a Pub/Sub dead-letter topic in Cloud Tasks.

26/27 Cloud Tasks

https://cloud.google.com/tasks/docs/comp-tasks-sched

27/27

Cloud Tasks is a managed service that allows developers to create distributed task queues that can execute background jobs asynchronously.

In this talk we will describe how Cloud Tasks works, highlight its differences with Cloud Pub/Sub, and suggest a few guidelines we can adopt when creating our tasks and monitoring our queues.