CaaS - Conversion as a Service

Architecture de l'Application

Mercredi 18 octobre 2017

SACC

Étudiants :
BELHASSEN Issam M2IFI AL
THIERNO Balde M2IFI AL
NAAMANE Ilias M2IFI WEB
EL MRIHY Mohamed M2IFI WEB

Plan

I.Sujet

Nous allons réaliser une API cloud native de transcodage de vidéo asynchrone, cette application de type CaaS (Conversion as a Service) propose plusieurs types de transcodage et plusieurs niveaux de qualité de service.
L'objectif de ce document est de présenter l'architecture globale de système hébergé sur la plateforme Google App Engine, les différents composants, les justifications des choix et d'estimations de coûts.

II.Scénarios et Diagramme de séquences

Scénario:

- En tant que client, je dois pouvoir déposer ma vidéo, choisis les conversions à réaliser parmi celles disponibles, la qualité de service attendu et un e-mail de contact.
- En tant que système je dois pouvoir envoyer un mail confirmant que l'opération est en cours.
- En tant que système je dois pouvoir envoyer un lien résumant le statut de chaque conversion.
- En tant que client, je dois pouvoir demander ma vidéo et la télécharger.

Ce diagramme de séquences illustre les différents interactions entre le Client, sa boîte Mail, et l'application afin de convertir une vidéo.

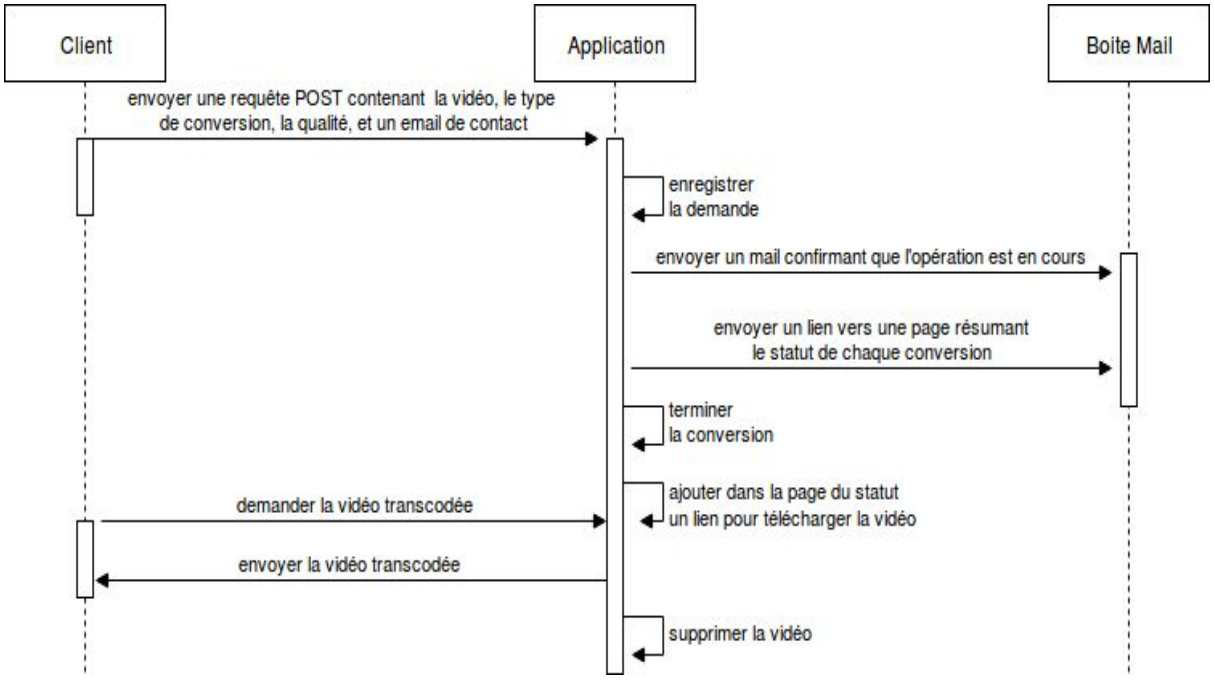

figure 3 :diagramme de séquences

III.Architecture global et description de composants

1.Schéma

En utilisant la plateforme PAAS Google App Engine, nous avons conçu une architecture qui garantit une flexibilité pour scaler nos services de manières indépendantes. En effet, les différents composants sur le schéma sont définis en tant que services.

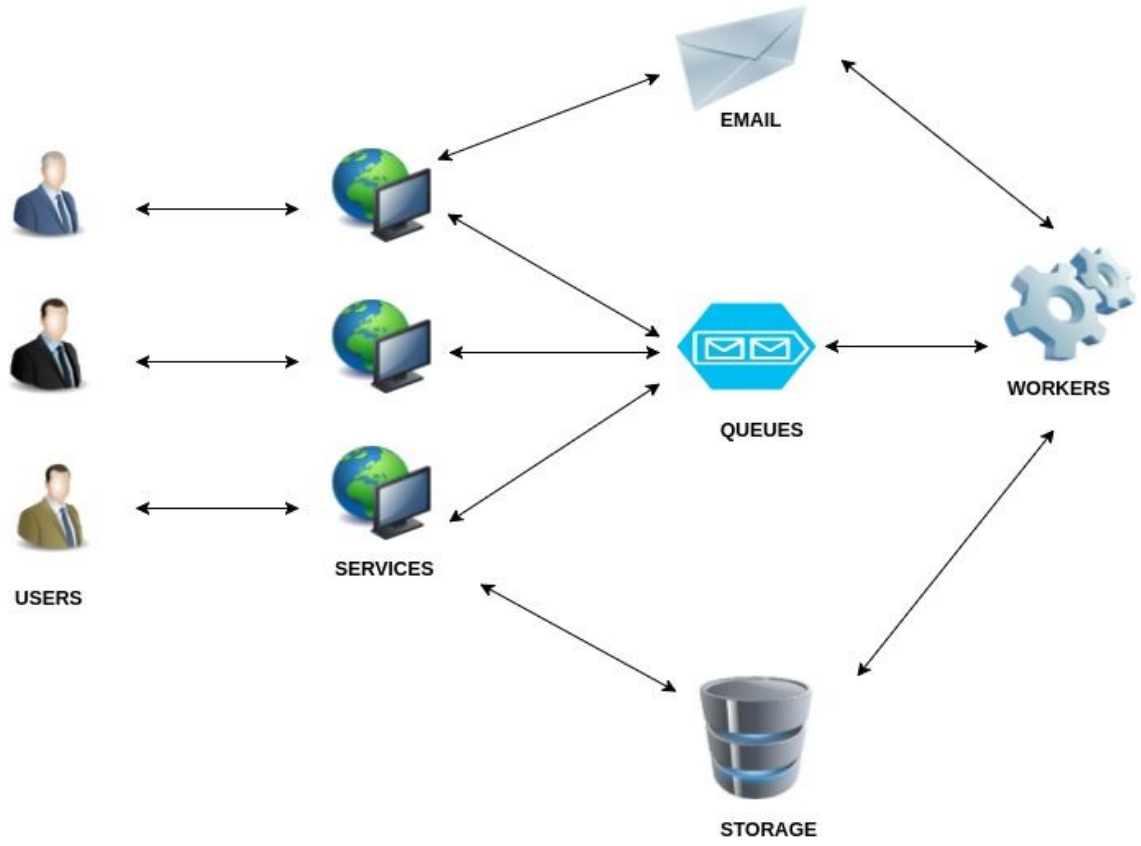

figure 1 : Architecture cloud de projet

2.Description de composants

a. SERVICES

Ce composant joue le rôle d'un load balancer, il permet de répartir les requêtes clients entre plusieures instances. Afin d'utiliser un load balancer en Round-Robin, ce composant doit être Stateless, il nous permet de garantir une meilleure performance en terme de latence et de sensibilité (réagir rapidement et positivement) et de faire des mises à jour à chaud sans que le service ne soit interrompu.

Ce composant suivra l'offre et la demande, on procédera donc à du scaling horizontal nous assurant ainsi l'élasticité de ce dernier.

Pour assurer la Résilience et le Fault-Tolerance, on mettra en place un système de reprise sur erreur basé sur les états d'une tâche géré dans le service de base de donnée. On utilisera des Cron

Job pour gérer la notion de TTL au niveau de la gestion de fichier mais aussi pour faire assurer la délivrance des mails.
Utilisation d'élasticité : Oui (automatic-scaling, min instance : 1, max instance : automatic)

b.Queues

Les requêtes provenant des utilisateurs sont enfilées dans les files d'attentes, nous utilisons un service de Queue afin de :

- garantir que les tâches de conversion seront bien traités à un moment donné.
- mettre en place un système de priorisation en fonction des grades utilisateurs.
- gérer les piques de charges sur les workers et de différer naturellement le traitement des tâches dans le temps en l'absence de worker disponible.

On interagit avec ce composant en mode asynchrone.

Pour garantir les SLA demandé on propose l'utilisation de trois queues(push queue) distinctes où chacune s'occupera d'un niveau de qualité de service :

- Pour la qualité de service bronze, on utilisera une push queue pour laquelle on spécifie un nombre maximum de worker parallèle de un. on spécifie aussi un manual scaling d'une seule instance et un rate de 1/1 afin de réduire les coûts. Cela revient à garantir que toutes les vidéos sont mises dans un pot commun et transcodé correctement une à une dans l'ordre de soumission.
- Pour la qualité de service silver, on utilisera une queue pour laquelle on spécifie un nombre de worker suffisamment grand pour garantir une rapidité de traitement décente. Afin d'empêcher un utilisateur de dépasser 3 conversion parallèles, on enregistre l'état de chaque conversion (Pending, Queued, Processing, Terminated) sur le service de base de données afin qu'un worker puisse dépilé une tâche.
  Il y a deux possibilitées :
  - ❖ L'utilisateur à moins de 3 conversion en cours, auquel cas le worker commence le traitement.
  - ❖ L'utilisateur a 3 conversion en cours, auquel cas la tâche est remise dans la queue et pour éviter que cette tâche soit dé-piler plusieurs fois rien, on utilisera la possibilité de définir un countdown (dans les TaskOptions) qui revient à définir un délai avant d'essayer de dépiler cette tâche de nouveau.
- Pour la qualité de service gold, on suivra le même principe que pour la qualité de service silver à ceci près que l'on prévoit un pool de worker plus important et que l'on autorise les utilisateurs à avoir jusqu'à 5 conversion en cours.

c.Workers

Les workers s'occupent de la validation des quotas de conversion et de la logique métier traitant de conversion des vidéos, on disposera de trois pools de workers (un par queue), qui procéderons de la manière suivante : *dequeue process delete,* ce qui nous donne l'assurance que le message sera toujours traité au moins une fois et vu que notre service est idempotent process plusieures fois ne pose pas de problème en cas de crash d'un worker.

Ce composant suivra l'offre et la demande, on procédera donc à du scaling horizontal nous assurant ainsi l'élasticité de ce dernier.

Pour le worker associé à l'utilisateur :

- Bronze : un 'manuel-scaling' d'une instance.

- Silver : un ‘automatic-scaling’ avec un maximum de 10 instances.
- Gold : un ‘automatic-scaling’ avec un maximum de 20 instances.

voici un exemple d’une implémentation possible de workers: on aura 3 type de worker(gold,silver,bronze) qui hérite d'une classe principale workerServlet, on suppose que chaque worker a deux parametres: concurrent (nombre de tâches concurrent) et expiration (durée d’expiration de tâches).

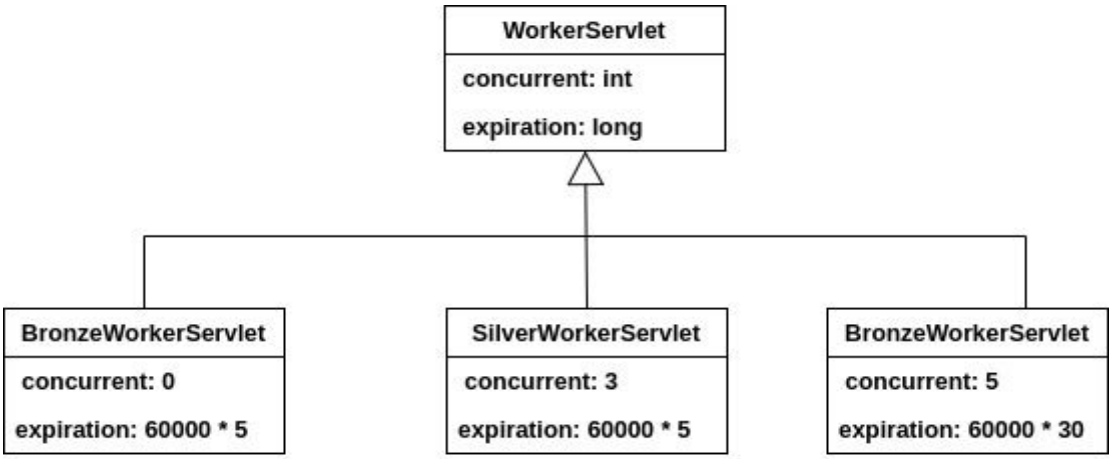

figure 2 :Exemple d’une implémentation de workers

d. Storage

le storage représente notre système de stockage qui est se constitue de deux type qui sont :

1. Stockage des données :

Ce sont les données de type mail, identifiant de l’utilisateur, la qualité de service (bronze, silver,gold) liées à la demande de conversion, les données des vidéos c’est à dire le nom et la durée en seconde et l’état des tâches sur le service.

Ce service nous permettra d’assurer la reprise sur erreur dans le cas d’une indisponibilité de service en reconstituant le référentiel des tâches donc il est hautement disponible, évolutif, simple et offre une flexibilité.

2. Stockage de Fichiers :

Ceci représente le système de stockage de fichiers qui vont être générés et téléchargés à la suite de la conversion.

e.Email

L’email représente le service qui se charge d'envoyer les notifications(prise en compte et l'état de la demande , et les liens de téléchargement des vidéos après conversions) liées à la demande de conversion aux différents utilisateurs.

IV.Estimation du coût de la plateforme (sans stockage)

Le coût de la plateforme pour chaque utilisateur dépend du nombre d’instances utilisés par heure, du load-balancer, et du coût queue.

Dans notre cas, on possède 3 types d'utilisateurs, dont on a pu effectuer quelques scénarios qui pourront nous aider à faire l'estimation:

- **Utilisateur Bronze:**

  Pour une vidéo d'une minute, la durée de conversion est entre 66s et 150s. Supposons que: la moyenne des durées de conversion pour ce scénario est: 108s : 0.03hr

- **Pour une faible charge** (une instance de classe B1 0.05$/hr) :
  Coût total = 0.05*0.03 = $0.0015
- **Pour une charge moyenne** (une instance de classe B1 0.05$/hr) :
  Coût total =0.05*0.03 = $0.0015
- **Pour une charge forte** (une instance de classe B1 0.05$/hr)=
  Coût total =0.05*0.03 = $0.0015

On estime que notre application convertira 1000 vidéos de type bronze par jour dont 200 sont en faible charge 200 en charge forte et 400 en charge moyenne avec un coût total de 1.5$/jr qui est équivalent à $45/mois.

Cout-total-estimé-bronze = 45$/month

- **Utilisateur Silver:**

Pour **3 vidéos** d'un utilisateur silver avec une durée moyenne de 500s chacune :

Pour une faible charge (une instance/conversion de classe B1 0.05$/hr) :
Coût total = 3*0.05*0.13 = $0.02
Pour une charge moyenne (2 instances/conversion de classe B1 0.05$/hr) :
Coût total =6*0.05*0.13 = $0.039
Pour une charge forte (3 instance/ conversion de classe B1 0.05$/hr)=
Coût total =12*0.05*0.13 = $0.078

On estime que notre application convertira 1500 videos de type silver par jour avec 150 vidéos en faible charge et 150 videos en charge forte et 1200 en charge moyenne.

Cout-total-estimé-silver = $21.5/jr → 645$/month

- **Utilisateur Gold:**

Pour **5 vidéos** d'un utilisateur gold avec une durée moyenne de 500s chacune:
Pour une faible charge (une instance/conversion de classe B1 0.05$/hr) :
Coût total = 5*0.05*0.13 = $0.0325
Pour une charge moyenne (2 instances/conversion de classe B1 0.05$/hr) :
Coût total =10*0.05*0.13 = $0.065
Pour une charge forte (3 instances/conversion de classe B1 0.05$/hr)=
Coût total =15*0.05*0.13 = $0.0975

On estime que notre application convertira 1200 vidéos de type gold par jour avec 510 videos en faible charge et 90 videos en charge forte et 300 en charge moyenne.

Cout-total-estimé-gold = $14.95/jr -> $448.5/month

D'où le coût total de la plateforme est :

Cout-total-estimé-platform = 1138.5/month

V.Modèle de coût pour l'application

Scénario : l'utilisateur envoie ses données(sa vidéo, son adresse mail,...) la conversion se fait, puis l'utilisateur récupère sa vidéo en considérant une instance par conversion.

- à faible charge :
  Services: 1 instance
  worker : 1 instance
  queues : 3 queues
- à moyenne charge
  Services: automatic
  worker : bronze 1 instance, silver 6 instances, gold 10 instances
  queues : 3 queues
- à forte charge
  Services: automatic
  worker : bronze instance, silver 10 instances, gold 15 instances
  queues : 3 queues

Système de facturation:

Utilisateur bronze: Gratuit
Utilisateur silver: $0.5/video -> $750/jr
Utilisateur gold: $1/video-> $1200/jr

Estimation-Total: $1950/jr →** **$/58500/month

Estimation-Gain: 58500-1138.5 → **$57361.5/month**