POLYTECH
NICE-SOPHIA

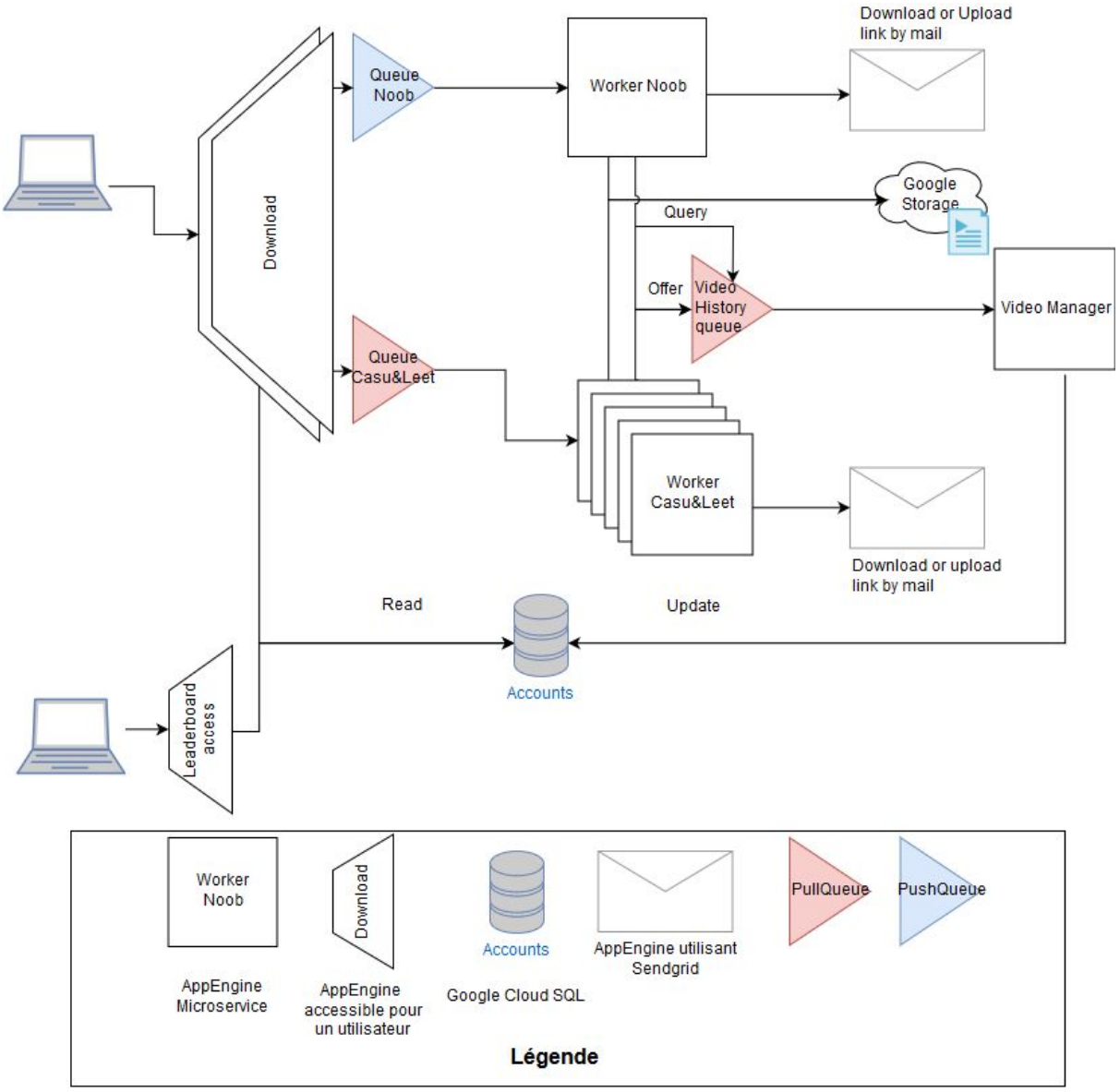
Université Nice Sophia Antipolis

Évaluation du coût de notre application PolyShare

CANAVA Thomas, GARDAIRE Loïc, JUNAC Jérémy, MELKONIAN François

Architecture logicielle pour le cloud computing
19/10/2018

Architecture de notre projet



<u>Figure n°1 :</u> Architecture de l'application

Nous avons découpé notre application en 5 microservices, chacun ayant une responsabilité différente.
De plus, notre application dispose de deux App Engines qui reçoivent les requêtes des clients.

L' App Engine *Download*

Cette app engine a la responsabilité d'empiler la requête de l'utilisateur sur une des deux queues, selon son statut : si l'utilisateur est un Noob, nous envoyons sa requête dans la *QueueNoob.* Sinon, sa requête est empilée dans la *QueueCasuLeet.*

La queue *QueueNoob*

Cette push queue contient les informations sur une demande de téléchargement ou de mise en ligne. Une seule instance consomme ces événements au rythme de 1 par minute par compte.
Pour le téléchargement, le message envoyé contient les informations sur un client et l'identifiant de la vidéo.
Pour la mise en ligne, l'ensemble de la vidéo ainsi que les informations sur le client seront compris dans la requête. Ce n'était pas le cas lorsque nous avons fait le diagramme et le pricing. Les informations sur ce point précis ne correspondront donc pas à la solution implémentée dans le futur.

La queue *QueueCasuLeet*

Cette pull queue gère les demandes des Casual et des Leet. En fonction de la demande, plusieurs instances pourront traiter ces messages. Le format des messages empilés est le même que *QueueNoob.* Notre utilisation de cette queue ne nécessite pas forcément une pull queue ou une push queue.

Les App Engines *Workers*

Ces AppEngines ont la responsabilité de créer un lien de téléchargement ou d'enregistrer une vidéo. Pour l'instance unique dédiée aux Noobs, *WorkerNoob*, elle traitera une demande par minute.
L'App Engine *WorkerCasu&Leet* possède un mécanisme de vérification du respect des quotas des utilisateurs, utilisant la pull queue *VideoHistory*. Après avoir traité une demande, un email récapitulatif est envoyé au client.

La pull queue *Video History Queue*

La pull queue regroupe les requêtes faites par le client. Ces messages ont deux vies:

- Tout d'abord, le message servira à la requête des app engines. En effet, lorsqu'un utilisateur fait une requête, cette queue est interrogée pour savoir si l'utilisateur a le droit de la faire.
- Ensuite, le message sera consommé par *Video History* queue qui supprimera la vidéo après le temps défini par le rang de l'utilisateur

Avant de traiter la requête d'un client, les app engines *Workers* effectuent une requête sur cette pull queue pour vérifier le nombre d'appels liés au compte de l'utilisateur et rejeter les requêtes en trop.
La pull queue sera défilée par l'app engine *Video Manager*.

L'AppEngine *Video Manager*

La responsabilité de ce service est d'incrémenter le score des utilisateurs en traitant les messages arrivant à destination. Ce module, commun aux Noob, Casual et Leet, met à jour la base de donnée. Il est aussi chargé de supprimer les vidéos une fois le temps écoulé.

Utilisation de l'élasticité dans notre projet

Notre architecture en microservices nous offre la possibilité de mettre à l'échelle horizontalement en fonction de la répartition de la charge.
La pull queue *VideoHistory* nous permet de garantir la cohérence de notre système de droit, tout en préservant le leaderboard d'un important nombre de vidéos ajoutées.

Ratio des niveaux sur 1 000 utilisateurs

Nous avons catégorisé nos potentiels utilisateurs en groupe nous semblant cohérent.
Noob : 600 utilisateurs

- 50% : Uploader exceptionnel (1-2 vidéos) qui regarde peu de vidéos (1h / semaine)
- 50% : Uploader exceptionnel (1-2 vidéos) qui regarde beaucoup de vidéo (7h / semaine)

Casual : 300 utilisateurs

- 50% : Uploader occasionnel (4-5 vidéos) et regarde peu de vidéos (1h / semaine)
- 50% : Uploader occasionnel (4-5 vidéos) et regarde beaucoup de videos (7h / semaine)

Leet : 100 utilisateurs

- 50% : Uploader expérimenté (~6 vidéos) et regarde peu de vidéos (1h / semaine)
- 50% : Uploader expérimenté (~6 vidéos) et regarde beaucoup de vidéos (7h / semaine)

Nous avons pris une moyenne de 6 vidéos mises en ligne par semaine pour les Leet. En effet, les personnes mettant en ligne plus de 1 vidéo par jour restent très rares, donc 6 nous semblait être une moyenne convenable.

Évaluation du coût

Pour les calculs, nous avons fait les suppositions suivantes :

- 1 heure de visionnage équivaut à 8 vidéos visionnées
- Le temps nécessaire pour générer un lien d'upload/download de vidéo est de 15 secondes

Coût des téléchargements et des mises en ligne des Noob

Téléchargement

Avec ces valeurs, nos calculs donnent les résultats suivants :

$$8 * 15 * 300 \ + \ 8 * 7 * 15 * 300 \ = \ 288\,000 \ s \ / \ semaine$$

Il y a 604 800 secondes dans une semaine. Il y a cependant un problème : les utilisateurs ne sont pas uniformément répartis en fonction du temps, car différents facteurs rentrent en compte : la nuit, les heures travaillées, etc. Si on se base sur les plateformes existantes, par exemple YouTube, elles observent un pic entre 17-18h et 22-23h.

Nous supposons qu'il va y avoir un pic entre 18h et 23h, en se basant sur les statistiques de plateformes vidéo comme Youtube.
Nous avons donc 50% du trafic en 18 000 secondes. Pour gérer ce pic, nous aurions besoin d'au moins 8 instances. Or, il est dit dans le sujet que les noobs n'ont qu'une seule instance allouée. Il est donc absolument impossible de traiter le trafic des noobs.
Il faut cependant noter un fait très important. L'estimation de 15 secondes pour traiter une requête est une valeur qui, a priori, serait bien au delà de la réalité. On peut raisonnablement penser que nous arriverons à gérer le trafic des noobs en pratique.

Nous avons ensuite approximer la taille d'une vidéo à une moyenne de 40 Mo.
Pour la bande passante nécessaire pour gérer ce trafic, nous avons fait les calculs suivants :

- Nous avons estimé que 50% des noobs regarderont peu de vidéos, et que les autres 50% en regarderont beaucoup.
- Donc $8 * 300 \ + \ 8 * 7 * 300 \ = \ 19\,200 \ vidéos \ / \ semaine$
- Soit 768,000 Mo par semaine

Pour ce qui est du nombre de requêtes qui vont être gérées par la queue, nous avons estimé leur taille à 1 Mo. En effet, ces requêtes sont petites car elles ne contiennent pas le fichier. Encore une fois, cette valeur de 1 Mo est une borne supérieure. On arrive donc à 19 200 Mo par semaine pour la queue.

Mise en ligne

Nous supposons que la mise en ligne des vidéos est le plus souvent bien répartie dans le temps. Le nombre d'instances reste cependant inchangé, car la mise en ligne et le téléchargement des noobs sont gérés par la même instance de notre application.

Pour la bande passante nécessaire pour gérer ce trafic, nous avons fait les calculs suivants :

- 1.5 vidéo de 40 Mo par semaine en moyenne
- $1.5 * 15 * 300 \ = \ 6\,750 \ s \ / \ semaine$
- $1.5 * 300 \ = \ 450 \ vidéos \ / \ semaine$
- $1.5 * 40 * 300 \ = \ 18\,000 \ Mo$

Pour les queues, cela nous donne donc 450 Mo par mois à traiter par les queues (car requêtes de 1 Mo).

Coût des téléchargements venant des Casual et des Leet

Avec ces valeurs, nos calculs donnent les résultats suivants :

- Pour les Casual : $8 * 15 * 150 + 8 * 7 * 15 * 150 = 144\,000\ s\ /\ semaine$
- Pour les Leet : $8 * 15 * 50 + 8 * 7 * 15 * 50 = 48\,000\ s\ /\ semaine$
- **Total :** $192\,000\ s\ /\ semaine$

Nous supposons qu'il va y avoir un pic entre 18h et 23h, en se basant sur les statistiques de plateformes vidéo comme Youtube.
Nous avons donc 50% du trafic en 18 000 secondes. Pour gérer ce pic, nous avons besoin de 5 instances et 2 instances sinon.
Nous arrivons à une moyenne de 2.625 instances par heure.

Pour la bande passante, nous arrivons donc aux calculs suivants :

- Pour les Casual : $8 * 150 + 8 * 7 * 150 = 9\,600\ vidéos\ /\ semaine$
- Pour les Leet : $8 * 50 + 8 * 7 * 50 = 3\,200\ vidéos\ /\ semaine$
- **Total :** 12 800 vidéos par semaine ce qui donne 512 000 Mo

Pour les queues, 12 800 Mo par mois à traiter par les queues (car requêtes de 1 Mo)

Coût des mises en ligne venant des Casual et des Leet

Nous supposons que la mise en ligne des vidéos est le plus souvent bien répartie dans le temps. Nous avons donc besoin de 2 instances, la seconde étant là en cas de pic exceptionnel. La moyenne sera donc de 1.1 instance.

- Pour les Casual : $4 * 15 * 300 = 18\,000\ s\ /\ semaine$
- Pour les Leet : $6 * 15 * 100 = 9\,000\ s\ /\ semaine$
- **Total :** $27\,000\ s\ /\ semaine$
- Nombre de vidéos mises en ligne : $4 * 300 + 6 * 100 = 1\,800\ vidéos\ /\ semaine$
- ↳1 800 Mo pour les queues (car 1 Mo par requête)
- Bande passante : $1\,800 * 40 = 7\,200\ Mo$

Coût de l'envoi d'email

Avec la gestion de la mise en ligne et du téléchargement, nous arrivons à un nombre d'emails à envoyer de $39\,650\ mails\ /\ mois$.

Sendgrid offre 12 000 emails gratuits par mois. Nous aurons donc des coûts supplémentaires liés à l'envoi de mail.
D'autre part, nous savons que d'autre alternatives comme MailJet nous permettent d'envoyer assez d'emails pour 21,95€ (60 000 emails par mois), si le coût vient à exploser.

Coût total de notre application :

Total (toutes catégories confondues) :

- 4.725 instances par heure en moyenne
- Bande passante: 1 305.2 Go
- Queues: 39.650 Go

$287.31 pour 1 000 utilisateurs sur 1 mois

↳ Soit environ $0.29 par utilisateur sur 1 mois

Résultat obtenu ici :
https://cloud.google.com/products/calculator/#id=9ee42b91-0557-427c-a0a1-3cc5ebd3b22d