Rapport SACC

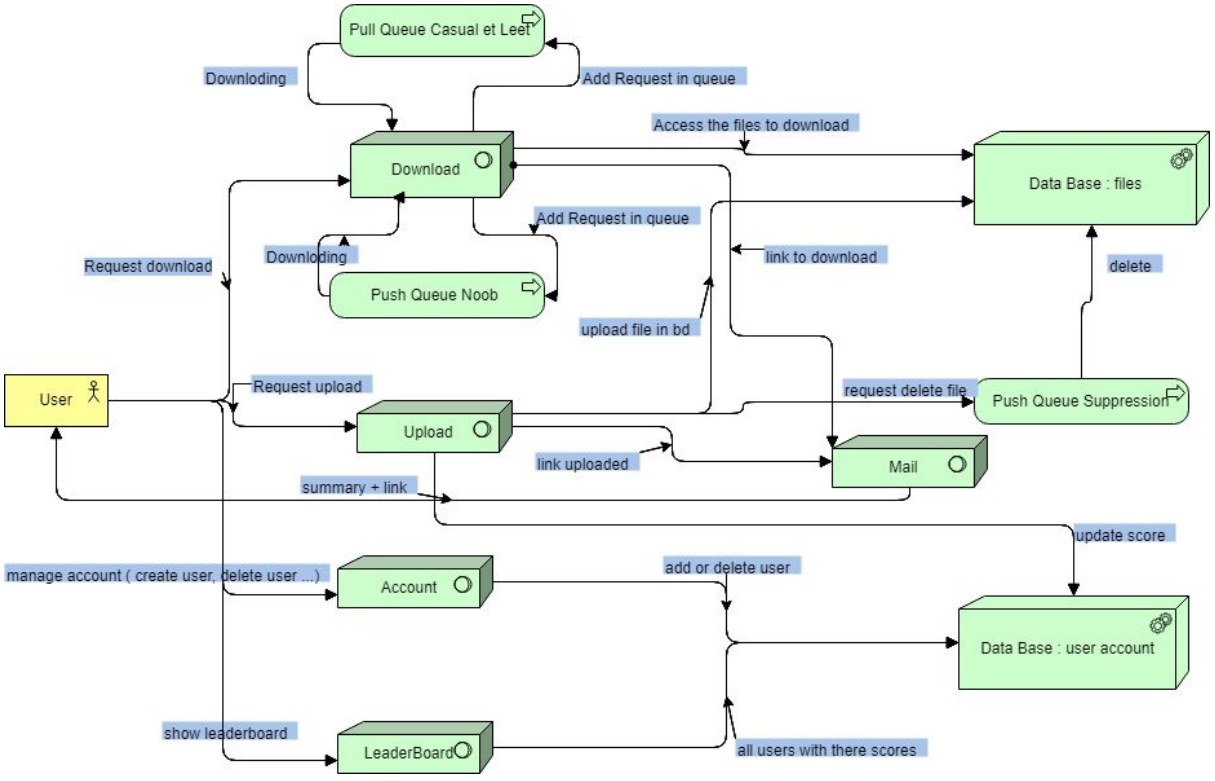
Mohamed Chennouf, Meersman Rudy, Duminy Gaétan et Ivan Picard Marchetto

Google Cloud

User Story

- Bob veut upload un fichier et possède le statut de Noob
- Il se connecte au service avec son identifiant
- Il fait une requête au système avec le path de la vidéo
- Le Système prend la requête en compte et calcul le nombre de point que Bob à gagner grâce à la taille du fichier
- Le système upload la vidéo sur le Google Cloud
- Une fois uploadé, le Système envoie à Bob un mail de confirmation avec le lien vers son fichier

Architecture Globale



PDF full size en annexe

Dans cette architecture, nous avons 5 composants :

Upload: Ce composant se charge d'upload les fichiers. Ce module possède une API qui prend l'id de l'utilisateur(email), la taille du fichier à upload, le path de la vidéo et la date actuel (qui sera supprimer au bout de 5/10/30 min) . Le fichier demandé est chargé dans la base de données.

Une fois uploadé, le module appelle le composant Mail pour envoyer le récapitulatif ainsi que le lien à télécharger. Ce module met aussi à jour la base de données des utilisateurs en incrémentant le score de la personne en fonction de la taille du fichier uploadé.

Les messages contenu dans les queues sont au format JSON afin de pouvoir identifier plus facilement les différentes requêtes. Un JSON va donc posséder un champ Action qui permet de déterminer la requête ainsi qu'un Body contenant les informations nécessaire à l'exécution de celle-ci. En voici un exemple:

```
{Action: "Upload",
 Body: {
        userID: Bob@gmail.com
        videoPath: "Bob/myVideos/video5.mp4"
        videoSize: 34
        dateRequest: timeNow()
        }
}
```

Download: Ce module télécharge les fichiers demandés par l'utilisateur. Il contient une API prenant l'email de l'utilisateur et l'identifiant du fichier à télécharger. Suivant le type d'utilisateur (Noob, Casual, Leet). Celui-ci va mettre les requêtes dans une queue appropriée. Une fois la tâche exécutée par les workers, le module download appelle le module Mail pour envoyer un mail avec le lien du fichier à télécharger.

Voici un message type pour l'action Download :

```
{Action: "Download",
 Body: {
        userID: Bob@gmail.com
        fileID: "Bob/myVideos/video5.mp4"
        }
}
```

Account: Ce module permet des créer, supprimer ou modifier un compte utilisateur.

LeaderBoard: Ce module récupère tous utilisateurs dans la base de donnée et les classent par score.

Mail: Ce module se charge de créer et d'envoyer des mails en fonctions de la requête exécutée.

Queues

Nous avons 3 queues pour gérer les différentes requêtes :

- Une push queue pour les Noobs :

Les files d'attente Push exécutent des tâches en transmettant des demandes aux "workers".

Elles envoient leurs demandes à un rythme fiable et régulier tout en garantissant l'exécution des tâches. Ces tâches sont traitées les unes après les autres, dans l'ordre de leur arrivée. Il est possible de contrôler leur taux d'envoi à partir de la file d'attente, nous permettant de minimiser le nombre de "worker". Ce contrôle du nombre de worker nous permet de minimiser les coûts. C'est pourquoi nous avons choisi une push queue pour les ***Noobs*** car leurs demandes sont traités séquentiellement et chaque minute les utilisateurs ne peuvent faire qu'une seule demande.

On a décidé de ne pas mettre en place d'élasticité pour la push queue car les requête sont traité séquentiellement et notre queue est capable de gérer un nombre suffisamment important d'utilisateurs ***Noobs***.

- Une pull queue pour les Casuals et Leets :

Les files d'attente Pull donnent plus de flexibilité quant au moment où les tâches sont traitées, de plus ces files d'attente fonctionnent bien lorsque l'on doit regrouper des tâches pour une exécution efficace. Les pulls queue permettent aussi de prioriser certaines tâches car contrairement à la push queue, on peut choisir l'ordre auquel les tâches vont être exécuté.

Grâce aux pull queue, on peut grouper deux tâches (download 1+download 2) et les exécuter en même temps. Un worker du pull queue pourrait se réveiller périodiquement toutes les 1 min pour exécuter les deux opérations des ***Casuals***. De même, un autre worker pourrait toutes les min exécuter les 4 opérations des ***Leets***. Ajouter à cela, le traitement des ***Causals*** et ***Leets*** est similaire. La seul différence est le nombre opérations en parallèle.

La pull queue nous permet aussi de mettre en place des tags sur les messages et ainsi différencier les messages venant des ***Casuals*** et ceux venant des ***Leets***. En différenciant ces messages par tags, nous pouvons associer des workers pour les

Leets (qui gère 4 requêtes en parallèle) et d'autre pour les ***Casuals*** (qui gère 2 requêtes en parallèle) . Cette pull queue a de l'élasticité horizontal qui nous permettra d'avoir plus de workers lorsque le système est surchargé.

- Une pull queue pour la suppression :

Cette pull queue sert à supprimer les fichiers contenus dans notre base de données. A chaque fois qu'un fichier est uploadé, un message est envoyé à la pull queue avec un temps (5 / 10 / 30 min suivant l'utilisateur) pour que les workers supprime le fichier après le délai.

Elasticité

Nous allons mettre en place une élasticité horizontale, par augmentation du nombre de serveurs . Cela ne pose généralement pas de problème pour les infrastructures Web. Pour l'instant, nous utiliserons le scaling automatique proposé par google.Cela nous permettra de gérer les multiples requêtes de téléchargement qui risque rapidement de saturer nos queues. Les autres besoins en élasticité sont délégués et expliqués dans la partie queue.

Coût de la plateforme

Le site de Google Cloud nous indique que le coût d'une instance *F2* est de **$0.10/h** pour les machines situées en Belgique (les plus proches).
Nous avons donc une base de 4 instance *F2* qui tourne 24 heures sur 24 et 7 jours sur 7. Le coût par mois au maximum est donc de: 0.10 * 4 * 24 * 31 = **$297.60/mois**.
Lorsqu'un surplus d'utilisateur surchargent une instance, on scale et on ajoute une machine pour supporter la charge. On a un scaling qui peut au pire des cas doubler soit **$595.20/mois**.
On considère que nous avons 1000 utilisateurs, on souhaite être rentable à partir de ce nombre minimum estimé, donc cela nous donne : **$0.60/mois** au minimum par utilisateur.

Références:

1. https://cloud.google.com/appengine/pricing?hl=fr
2. https://github.com/Rmeersman/Cloud_Computing_GDRMIMMC/blob/master/image/archi.pdf