Rapport de projet : Google Hashcode

TeamS : Calvin Audier, Taoufiq Kounaidi, Benjamin Feray, Soufiane Aourinmouche

Analyse théorique du problème :

D’un point de vue théorique, la complexité du problème se représente principalement dans la complexité à trouver une solution qui se rapproche de la meilleure solution tout en ayant un temps d'exécution acceptable. Nous étions amenés dans ce problème à chercher des solutions qui sont à la fois pas très coûteuses en terme de complexité et (et donc en temps d’exécution) et qui sont efficace; d’où la nécessité de les optimiser et de l’utilisation du benchmark. Une autre difficulté moins complexe était le traitement d’une grande masse de données, en prenant comme entrée des grands fichiers (kittens.in par exemple), ce qui rend le traitement plus complexe, mais qui montre les limites de la solution utilisée et montre si elle est efficace ou pas.

Description de nos stratégies :

Stratégie 0 : Cette stratégie ne fait rien. Elle avait pour but de tester l'intégrité de notre projet en début de semaine.

Stratégie 1 : Notre première stratégie prend une video aléatoire et la mets dans un cache qui lui aussi est choisi au hasard. Si la vidéo est trop volumineuse pour le cache, alors la stratégie ne fait rien.

Stratégie 2 : Cette stratégie va choisir la vidéo la plus demandée parmis tout les endpoints, et la place dans tout les caches (si la vidéo n'est pas trop volumineuse).

Stratégie 3 : En ce qui concerne la stratégie 3, elle va, pour chaque endpoint, placer la vidéo la plus demandée dans le cache où le délai de latence est le plus faible (si ce dit cache a assez de place pour l'accueillir, sinon la vidéo n'est pas placée).

Stratégie 4 : Mieux encore, la stratégie 4 va , pour chaque endpoint, extraire les vidéos les plus demandées sous la forme d'une pile (en haut de la pile se trouve la vidéo la plus demandé, en bas la vidéo la moins demandée, les vidéos non-demandées ne sont pas placées), puis va les stocker dans le cache dont le délai de latence est le plus faible, sauf si ce dit cache est totalement rempli, dans ce cas on passe au endpoint suivant.

Stratégie 5 : C'est une amélioration de la stratégie 4. Elle consiste à ordonner, pour chaque EndPoint, une liste des serveurs caches auxquels il a accès par latence (celui avec le moins de latence est le meilleur), et ordonner les vidéos qu'il demande par nombre de demandes. Ainsi nous stockons les meilleurs vidéos dans le meilleur cache serveur possible jusqu'à ce qu'il ne peut plus stocker la vidéo suivante, nous la stockons alors dans le meilleur cache suivant *(au lieu de s'arrêter comme elle fait la stratégie 4).*

Stratégie 6 : Cette stratégie est la même que la stratégie 5,sauf que les vidéos ne sont pas ordonnées par nombre de requêtes, mais par le rapport nombre de demandes / poids de la vidéo. Cela nous permet d'un peu plus prioriser les vidéos plus légères même si elles sont moins demandées et permettre de ne pas remplir un cache par une seule vidéo volumineuse.

Analyse de la complexité de nos solutions :

Critique de la stratégie 0 : Vue que cette stratégie ne fait rien, elle n'utilise aucun algorithme qui peut être critiqué.

Critique de la stratégie 1 : Cette stratégie ne repose que sur l'aléatoire et n'utilise aucune sorte de boucle, la complexité est en O(1).

Critique de la stratégie 2 : Cette stratégie repose sur la recherche de la vidéo la plus demandée par tous les endpoints pour la stocker dans tous les caches. Une boucle parcourant tous les endpoints pour construire l'association (video, somme de nombre de requêtes par tous les endpoints), est d'une complexité O(N) , ce qui n'est pas très complexe, mais trop cher vu le résultat de la stratégie qui n'est pas très efficace, parce qu'elle centralise son intelligence à choisir la vidéo sur tous les endpoints, ce qui n'est pas intéressant pour certains (un endpoint qui ne demande pas la vidéo l'aura dans le cache le plus proche)

Critique de la stratégie 3 : Cette stratégie est moins complexe que la précédente, pourtant plus intelligente et plus efficace en stockant pour chaque endpoint sa vidéo la plus demandée dans son cache le plus proche.

Critique de la stratégie 4 : La stratégie est une imbrication de deux boucles (un while dans un for) afin de stocker les vidéos les plus demandées pour chaque endpoint. Elle est d'une complexité supérieure à O(2N), mais elle se limite à l'utilisation d'un seul cache serveur. Pour un endpoint dont la majorité des vidéos les plus demandées sont de petite taille, elle peut être très efficace en stockant beaucoup de vidéos dans un cache, mais pour un autre qui demande des vidéos de grandes tailles, elle va stocker peu de vidéos dans le cache ce qui n'est pas très intéressant.

Critique de la stratégie 5 : La stratégie contient deux boucles "while" imbriquées ce qui est coûteux en terme de complexité. La deuxième boucle pour remplir le cache choisi tant qu'il y'a de l'espace suffisant pour la vidéo à stocker, la première boucle tourne tant qu'il y a un serveur à utiliser et tant que les vidéos ne sont pas toutes stockées. Alors l'omission des vidéos qui ne sont pas beaucoup demandées, nous aurait épargné plusieurs itérations et calculs à faire, et aurait diminué la complexité sans trop influencer le score.

Critique de la stratégie 6 : Cette stratégie est conçue pour remédier au problème de la stratégie 5 qui est de remplir vite les caches par de grandes vidéos, donc un cache contient à la fin une,deux ou trois vidéos très volumineuses. Sauf que son exécution sur des grands fichiers donne un résultat moins bon que celui de la 5ème stratégie, parce que en général les vidéos les plus demandées sont les plus volumineuses , la stratégie n'a pas donné ce que nous attendions d'elle (avec les fichiers .in fournits en tout cas).

Organisation au cours de la semaine :

Les milestones nous ont permis d'avoir une idée sur les objectifs à atteindre sur le court terme. Ils étaient découpés en demie journées, et étaient alors beaucoup plus détaillés.
Le premier jour de la semaine, nous avons décidé de faire que la conception du projet, mais nous étions beaucoup dans le risque (*"Parmi tous les groupes que j'ai vu, vous êtes les plus risqués"* , comme nous a dit notre facilitateur Benjamin Benni). Nous avons dû alors changer nos milestones pour diminuer nos risques.

Nous avons décidé alors de changer nos milestones comme suivant :
. Faire le mvp Mardi (le deuxième jour) plutôt que le Mercredi
. Fixer des objectifs plus précis que ce qu'on avait, comme "générer un graphe vide" plutôt que "commencer le visualisateur"

Le milestone du premier jour était de finir la conception, ce que nous avons réussi à faire.
Les premières lignes de code ont étaient écrites le deuxième jour, et grâce au changement au niveau des milestones, nous avons beaucoup diminué les risques (*"Vous avez un luxe, c'est que vous n'avez plus aucun risque"* ,nous a dit Benjamin Benni).
Les feedback donnés chaque jour par le facilitateur étaient d'une grande utilité pour nous, l'alerte du grand risque que nous avions au début nous a mis sur le bon chemin. Son feedback sur les graphes que nous comptions faire à partir du fichier data.in, nous a épargné du travail qui n'allait avoir aucune valeur pour le client vu que les données traitées sont fournis par le client lui même.

Opinions sur :

Maven : Le concept du POM nous a permis de :

- Automatiser la gestion et la construction du projet
- Avoir une compréhension plus simple et plus complète de l'amélioration du projet
- Gérer les dépendances entre les trois modules du projet

Git : Tout au long de ce semestre, nous avions remarqué qu'avec Git, on a pu travailler sur plusieurs projets (Dojo, Takenoko, Hashcode) sans se marcher sur les pieds. De plus, quand un fichier a été modifié par plusieurs personnes en même temps, Git sait s'adapter et choisir un algorithme qui fusionne intelligemment les lignes du fichiers qui ont été modifiées. À cela s'ajoute la rapidité de Git, les données sont empaquetées, compressées, et les mises à jour sont fusionnées rapidement.

La session de la démo : Pour la démonstration, notre objectif était de valoriser la structure et l'architecture du projet, montrer que nous avons une union (engine - benchmark - visualiser) bien solide, et de montrer que nous avons pu avoir une version assez complète des trois modules du projet, avec des stratégies assez basiques et pas très améliorées. Ainsi nous avons consacré le temps qui restait après le jour de la démo à l'amélioration du moteur pour le rendre moins lourd et plus opérationnel, et surtout à l'amélioration de nos algorithmes pour mettre en place des stratégies plus intelligentes. La démo nous a permis d'avoir un feedback technique, contrairement aux feedback du facilitateur au cours de la semaine qui étaient plutôt des feedback client. Ainsi, nous nous avons posé plus de questions sur nos choix (l'utilité de l'héritage dans quelques cas).

Distribution des 400 points :

Calvin Audier : 100

Taoufiq Kounaidi : 50

Benjamin Feray : 150

Soufiane Aourinmouche : 100