Equipe Collet wa mou Mosseiru

Younes Abdennadher
Alexandre Bolot
Camille Leroux
Hugo François

Rapport Projet semestre 5 – Equipe B

I. Analyse du problème

Le problème énoncé par le Hashcode de Google est un problème NP complet. C'est-à-dire que les éventuelles solutions sont facilement vérifiables mais il est beaucoup plus complexe d'en trouver de nouvelles efficacement. D'autre part, tous les algorithmes destinés à résoudre un problème NP-complet ont un temps d'exécution exponentiel en fonction de la taille des données, ce qui rend quasiment impossible le fait de trouver la solution la plus optimale. Tester toutes les possibilités serait bien trop long. On pourrait comparer le Hashcode au problème du sac à dos (https://fr.wikipedia.org/wiki/Probl%C3%A8me_du_sac_%C3%A0_dos#Sac_%C3%A0_dos_multi-objectif) puisque nous avons un certain nombre d'objets (vidéos) à répartir entre plusieurs sections (caches). Les caches sont assez similaires à plusieurs sacs à dos. Mais dans notre cas, en plus de les remplir, nous devions prendre en compte d'autres critères comme les temps de latence. Nous faisons ainsi face à un problème d'optimisation combinatoire ; parmi un ensemble de solutions, nous devons déterminer la solution la plus efficace tout en prenant en compte plusieurs critères.

Pour résoudre ce problème, il a donc fallu que nous testions plusieurs algorithmes différents afin de trouver celui qui permet d'obtenir le meilleur score. Le problème étant NP-complet, il est par définition assez difficile à résoudre. Il vaut donc mieux chercher des solutions approchées et non pas tester toutes les possibilités pour déterminer la plus efficace.

II. Nos solutions

A. Stratégies implémentées

1) AllInDataCenterStrategy

Cette stratégie est la plus basique. Le DataCenter est le seul possesseur des vidéos et envoie toutes les vidéos aux endpoints qui en ont fait la requête sans passer par les caches. Les caches sont donc constamment vides quand cette stratégie est appliquée. Logiquement, son score est de 0.

2) LightestsInCache

Cette stratégie permet au DataCenter d'envoyer aux serveurs caches les vidéos les plus légères d'abord afin de mettre le plus de vidéos possibles dans les caches. Elle trie donc la liste de toutes les vidéos par taille de vidéo (de la plus petite à la plus grande) puis parcourt cette liste et envoie les vidéos aux serveurs cache jusqu'à ce que ceux-ci soient remplis. La stratégie vérifie également que le serveur cache est connecté à un endpoint ayant besoin de ladite vidéo. Si cette vérification n'était pas faite,

les serveurs caches pourraient être saturées de vidéos inutiles pour les endpoints auxquels ils sont connectés.

3) AverageStrategy

Cette stratégie calcule dans un premier temps le temps de latence moyen cache-endpoint pour chaque cache et trie les caches par temps de latence (du plus petit au plus grand). Elle parcourt ensuite chaque endpoint en fonction de cette liste et les remplit un par un avec des vidéos. Si cette stratégie est aussi bas dans le classement, c'est parce que l'équipe n'a pas eu le temps de l'optimiser en triant également les vidéos par popularité (c'est-à-dire par nombre de fois où elles sont demandées). Cela aurait permis de ne mettre dans les endpoints les plus *rapides* que les vidéos qui sont les plus populaires. Ces vidéos seraient alors privilégiées par rapport à celles qui sont le moins vues. En appliquant cela, cette stratégie pourrait même être meilleure que la suivante (BestForEachCache) puisque le critère du temps de latence est aussi pris en compte.

4) BestForEachCacheStrategy

Cette stratégie parcourt les serveurs caches et et les remplit avec les vidéos les plus demandées par les endpoints reliés.

Pour chaque cache, elle crée une liste de vidéos demandées par les endpoints reliés au cache et trie ces listes par nombre de requêtes faites pour cette vidéo. La stratégie remplie ainsi les caches avec les vidéos les plus populaires pour les caches.

Assez logiquement, cette stratégie fait un meilleur score que la dernière puisque ce sont les vidéos les plus demandées qui sont priorisées.

5) RandomStrategy

Le fonctionnement de cette stratégie est simple. On parcoure les endpoints un par un et on les remplit de vidéos tirées aléatoirement.

6) ProbaTegy

La ProbaTegy trie les serveurs cache par nombre de connexions à des endpoints puis remplit les caches un à un dans l'ordre de la liste triée. On a ainsi d'abord les caches qui ont le plus de connexion (et donc la plus grande probabilité d'être relié à un endpoint qui a besoin des vidéos dans le cache) qui sont remplis. Cette stratégie est adaptée à n'importe quel type d'architecture de fichier d'input, c'est ce qui explique son score élevé.

7) CacheIfQueryStrategy

Dans un premier temps, la stratégie calcule nombre moyen de requêtes de vidéos par endpoint puis calcule la moyenne de cette moyenne.

Dans un second temps, la stratégie parcourt les caches et y place les vidéos qui sont demandées plus de fois que la moyenne calculée.

8) FirstInStrategy

Cette stratégie ajoute les vidéos dans les caches dans l'ordre dans lesquelles elle les trouve. Il n'y a pas vraiment d'algorithmique dans cette stratégie mais nous avons découvert qu'elle parvenait à obtenir des scores relativement élevés. Nous avons donc décidé de la conserver dans ce classement.

B. Structure du projet

Pour ce qui est de sa structure, le projet est séparé en 3 sous-modules Maven nommés *engine*, *benchmark*, et *visualizer*.

Ce découpage en module a permis à l'équipe de bien séparer les tâches les unes des autres pour éviter au maximum les conflits d'implémentation.

Dans le module *engine*, nous avons utilisé le polymorphisme pour les stratégies où les toutes les classes qui héritent de la classe *Strategy* implémentent une méthode utilisée par le controlleur pour calculer le score et générer les fichiers de sortie.

III. Organisation de l'équipe

A. Planification

Au début de la semaine, nous avons prévu de livrer un MVP (*Minimal Valuable Product*) en deux jours afin de garder du temps en marge en cas d'imprévu, de retard, ou autre. Mais cette planification était ambitieuse étant donné que nous ne connaissions pas les technologies à utiliser pour le benchmark et la visualisation.

Le deuxième jour, nous avons donc réévalué notre planning et consommé une partie du temps que nous avions gardé en réserve. Cela nous a permis de mieux optimiser et répartir notre temps sur la semaine et de proposer un MVP pour le jour de la soutenance et d'être légèrement en avance pour la suite des évènements.

B. Répartition du travail

Après avoir fait un diagramme de classes, nous avons utilisé l'outil de kanban de Trello afin de nous répartir les tâches dans le groupe et dans le temps.

Ainsi, nous nous sommes répartis les tâches comme ceci : Hugo s'est chargé de mettre en place l'architecture du projet Maven et a implémenté le Benchmark ; Alexandre lui, s'est chargé de créer le *visualiseur* et les graphiques correspondants ; Younes s'est occupé du parseur et de ses tests et Camille a fait l'implémentation initiale des classes et l'algorithme de calcul de score (avec ses tests).

En ce qui concerne les stratégies, chacun des membres en a implémenté un certain nombre.

C. Bénéfices de la planification initiale

How have you been able to benefit from the milestones you defined? How would you modify them, now that you are at the end of the project? How have you integrated the feedback given every day?

Tout d'abord, le diagramme de classe fait le premier jour, nous a permis pour avoir une meilleure vision de l'implémentation de nos classes et de l'architecture globale que le projet allait avoir.

De plus, la manière dont nous nous sommes répartis les tâches entre membres était assez optimale. Nous avons rencontré très peu de conflits lors de l'implémentation.

Si la planification qui était assez ambitieuse le premier jour a vite été corrigée et revue c'est surtout grâce aux feedbacks que nous avons eu des professeurs encadrants. La nouvelle planification a permis d'avoir une vision globale du projet, de voir vers où l'équipe avançait.

Si nous devions changer quelque chose lors du déroulement du projet, nous rajouterions une deuxième phase de planification en milieu de semaine (après la soutenance par exemple). La planification initiale prévoyait l'implémentation de la livraison d'un produit minimal pour la démonstration et ne prévoyait que de manière générale les phases de relecture et de finalisation du code.

Une deuxième phase de planification en milieu de projet aurait permis de préciser ce qu'il restait à faire et à améliorer, de finaliser plus rapidement le projet en répartissant les nouvelles tâches et d'éviter certaines confusions sur les tâches à réaliser en fin de semaine.

IV. Déroulement du projet

A) Avis sur les interventions

Nous n'avons pas toujours tous pu assister aux sessions de démo et de présentations durant la semaine. Mais au moins un membre était présent à chaque présentation. La présentation de M. Papazian le mercredi nous a permis d'avoir une nouvelle source d'idées pour la création des stratégies et nous avons pu revoir les stratégies déjà implémentées avec un point de vu plus théorique afin de les optimiser.

L'idée générale est que ces ateliers étaient utiles pour ceux qui rencontraient des problèmes avec les technologies utilisées et que les informations apprises dans ces modules étaient détaillées et claires.

Lors de la présentation git du jeudi après-midi, nous étions trop avancés dans le projet pour mettre en place efficacement l'architecture et l'organisation proposée par le professeur. En revanche, nous avons tous bien compris les intérêts que cela apporte et nous pensons réutiliser ces bonnes pratiques dans nos prochains projets, autant que pour nos projets personnels.

B. Implication des membres

Comme dit plus haut, au cours de la semaine chacun des membres de l'équipe avait des tâches bien déterminées et tout le monde a implémenté des stratégies. Nous avons donc décidé, étant donné la répartition équitable du travail et des efforts fournis au sein de l'équipe, d'allouer de façon équilibrée le nombre de points disponibles.

99 points sont donc attribués à Younes, Alexandre et Camille et nous avons attribué à Hugo 103 points car il a mis en place toute l'architecture du projet et a travaillé sur plusieurs parties différentes.