Rapport de projet

Hashcode Google: Full time project 2017-2018

I. Analyse du problème

Ce projet doit commencer par la récupération des différentes informations passé dans le fichiers texte de google pour ensuite les créer dans notre structure : les caches, endpoints, vidéos, et requêtes. Ensuite, il faut créer des stratégies : c'est le principal problème de ce projet. Ces stratégies ont pour but de placer les vidéos sur les différents caches afin d'optimiser le temps d'accès à celles-ci depuis les différents endpoints.

Pour cela, il faut placer les vidéos de manière intelligente sur les différents caches, en fonction des requêtes de vidéo effectuées par les endpoints, la taille des caches, la taille des vidéos, et la latence de chacun des endpoints par rapport aux caches.

Nous pourrons ensuite donner un score à notre stratégie en fonction du temps total économisé sur toutes les requêtes de vidéo, par rapport à un schéma ou aucune vidéo n'est placée sur les caches, et dans lequel tous les endpoints doivent donc aller chercher leurs vidéos sur le serveur central, dont la latence est maximum.

Finalement, il faut mettre en place un benchmark, qui va permettre de récupérer les temps d'exécutions de nos différentes stratégies, et une visualisation, qui va permettre d'imager ces scores, et d'autres informations utiles.

Ce projet comprend plusieurs difficultés principales :

- Tout d'abord, la récupération les différentes informations (caches, endpoints, …) et les ranger de manière efficace, afin de pouvoir y accéder facilement pour le développement des stratégies, il faut donc une architecture efficace.
- Le développement de stratégies efficaces permettant d'obtenir un haut score, et qui se lance rapidement.
- La mise en place d'une visualisation en R, aucun membre ne connaissant ce langage, il est nécessaire de se documenter sur celui.
- De même pour la mise en place du benchmark, inconnu aussi par les différents membres, et dont la documentation est complexe.
- Le projet demande de gérer une dépendance entre modules avec maven qui est complexe.
- Le projet demande d'utiliser des modules imbriqués, ce qui a compliqué le pom.xml.

II. Les différentes stratégies

L'état final de notre projet comporte 8 stratégies. Voici leurs descriptions :

Stratégie 1 : Triviale
Stratégie de référence pour la suite, ne place aucune vidéo dans les caches.

Stratégie 2 : Triviale
Prend une vidéo aléatoire, sans prendre de doublons, et la place dans un cache aléatoire, jusqu'à ce qu'il n'y ait plus de place dans les caches, ou qu'il n'y ait plus de vidéos à placer. Cette stratégie sera mieux que la première, mettant à disposition des vidéos dans les caches.

Stratégie 3 : Triviale
Prend toutes les vidéos, une par une, et les places toutes dans le premier cache, puis le second, jusqu'à toutes les placer, ou remplir tous les caches. Comme la stratégie 2, cette stratégie permet de mettre à disposition des vidéos dans les caches plutôt que seulement sur le serveur principal.

Stratégie 4 : Intelligente
Rempli tous les caches avec la vidéo la plus demandée par tous, puis la seconde plus demandée, jusqu'à avoir placé toutes les vidéos, ou rempli tous les caches. Cela permet ainsi d'avoir les vidéos les plus demandées sur les caches en priorités, dans le cas où il n'y aura pas assez de place dans les caches pour toutes les placer.

Stratégie 5 : Intelligente
Met les vidéos les plus demandées par tout le monde dans le meilleur cache pour chacun des endpoints, puis dans le second meilleur cache, jusqu'à remplir tous les caches, ou avoir placé toutes les vidéos. Cette stratégie permettra de rendre disponible les vidéos les plus demandées dans un cache proche, pour chacun des endpoints.

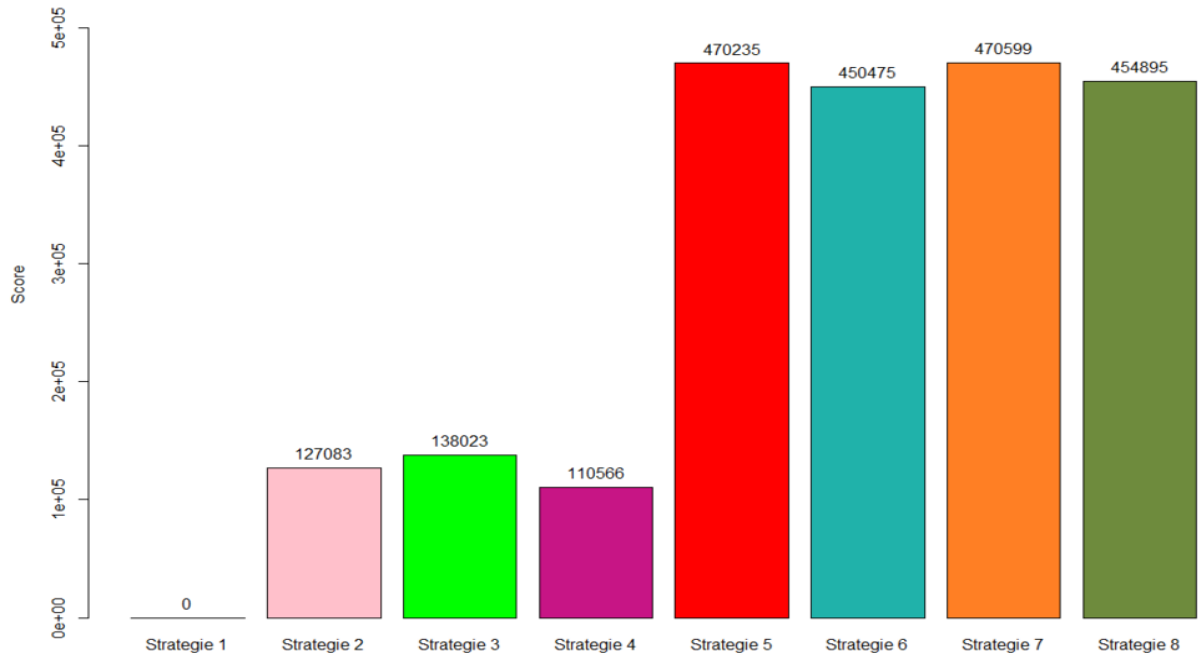
Stratégie 6 : Intelligente
Met la vidéo la plus demandée par un endpoint dans ses meilleurs caches, puis la seconde vidéo, jusqu'à avoir placé toutes les vidéos.Contrairement à la précédente, nous prenons en compte les requêtes de chacun des endpoints, et place les plus demandés sur leur cache le plus proche, permettant un placement optimisé pour chacuns des endpoints.

Stratégie 7 : Intelligente
Dans cette stratégie nous parcourons tous les caches puis nous parcourons tous les endpoints reliés au cache et pour chaque endpoint nous parcourons leurs requetes afin de calculer quel serait le temps gagné si la vidéo était déplacée dans le cache. Ensuite nous stockons toutes ses informations dans une treeMap afin de pouvoir la trier facilement et mettre uniquement les vidéos les plus rentables dans le cache.

Stratégie 8 : Intelligente
Cette stratégie commence par les endpoints qui font le plus de requêtes totals, puis il prend le cache ayant le moins de latence pour ce endpoint et enfin commence par le plus petit ratio taille/nombre de requête pour cette vidéo. Cette technique permet d'optimiser l'espace dans les caches, en prenant en compte la taille de la vidéo ainsi que son nombre de fois où elle a été appelée.

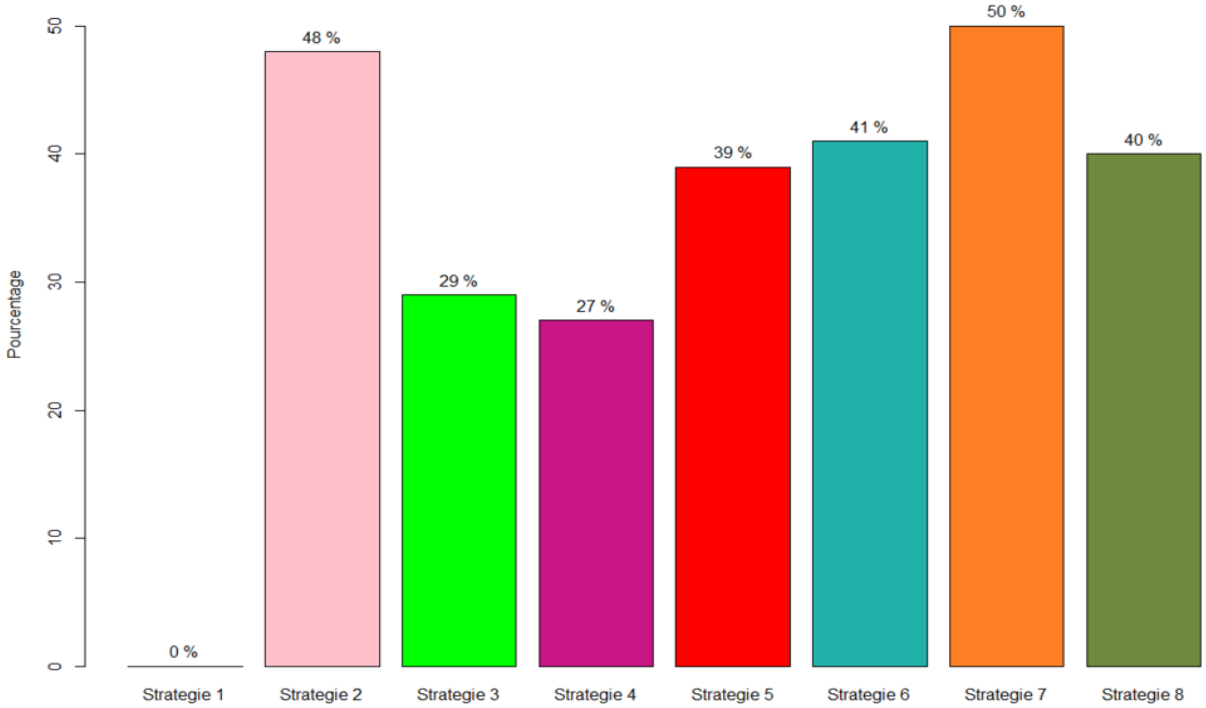
III. Analyse de la complexité de notre solution

Voici le graphique du score de chaque stratégie avec en entré le fichier me_at_the_zoo.in :

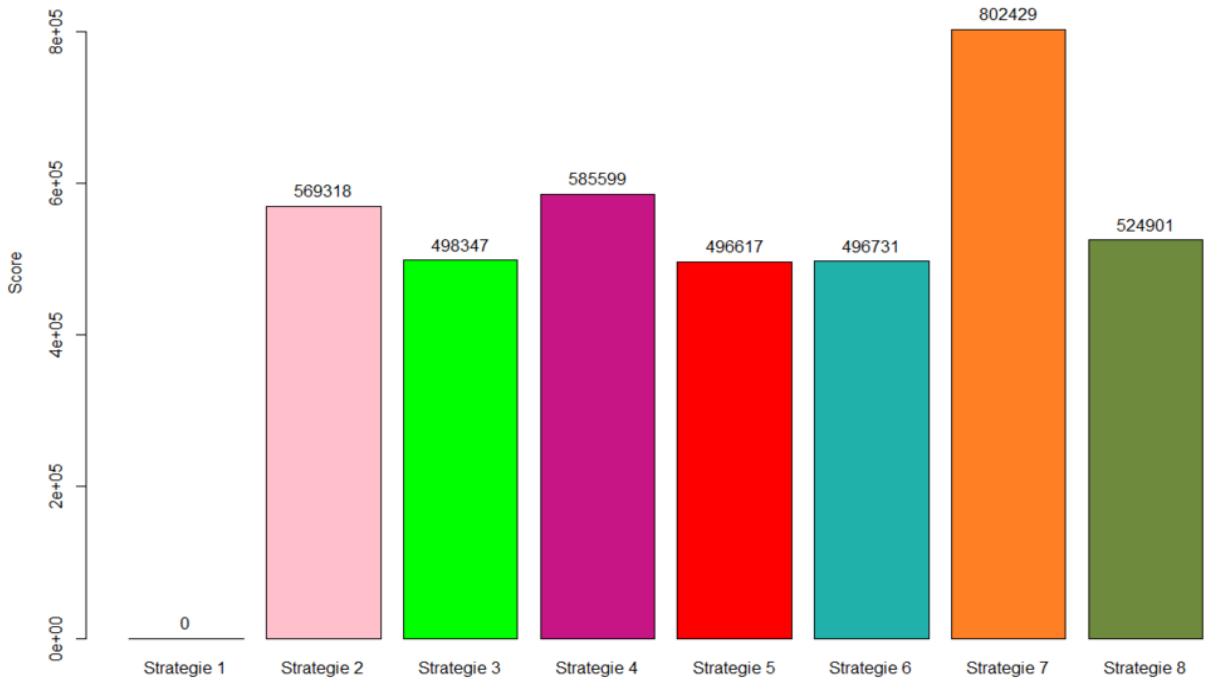


Nous pouvons voir que les 4 stratégies intelligentes donnent un bien meilleur score que les stratégies triviales. Le fichier d'entrée zoo a plus de places dans les caches que de vidéo à ranger, schéma qui profite aux stratégies 5 et 7, leur permettant d'obtenir un meilleur score que les 6 et 8. La stratégie 5, plaçant les vidéos les plus demandées par tous, dans les caches les plus optimisés pour chaque endpoint, peut ainsi placer toutes les vidéos les plus demandées dans les caches des endpoints. La stratégie 7, elle place les vidéos qui feront gagner le plus temps dans les caches.

Voici le graphique du pourcentage de remplissage des caches par stratégie avec le fichier me_at_the_zoo.in :

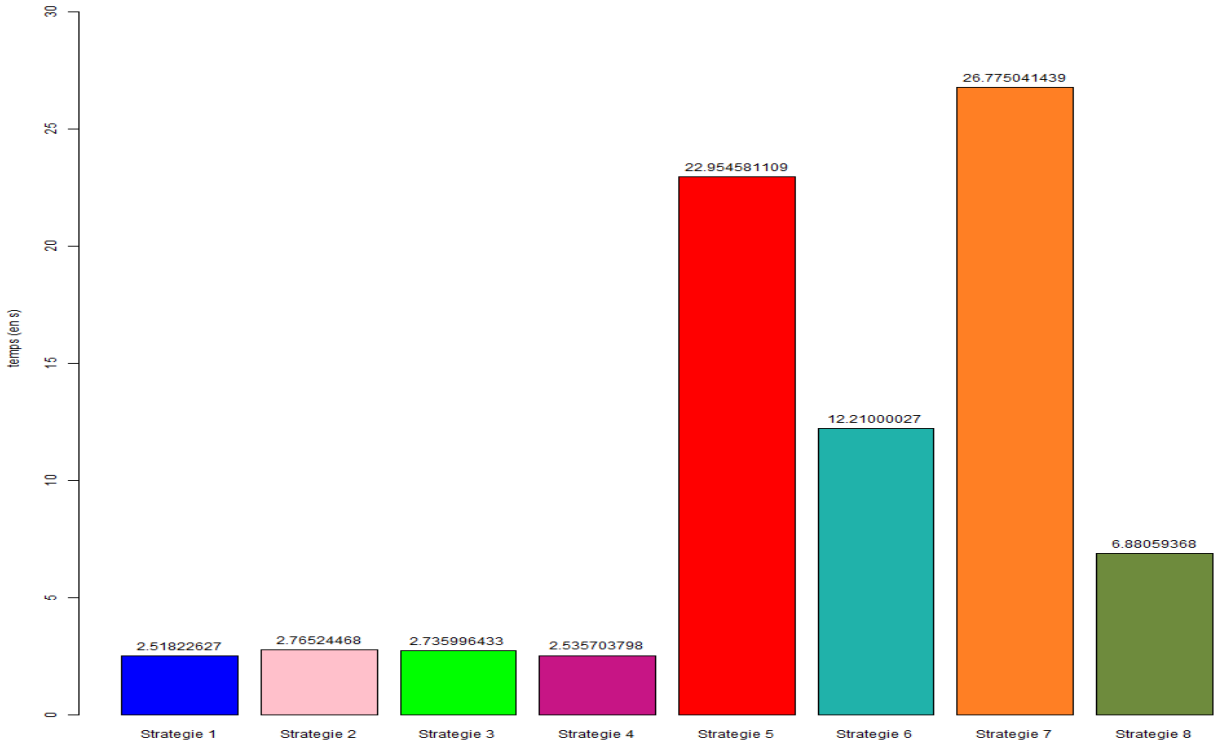


Voici le graphique du score de chaque stratégie avec le fichier data.in :



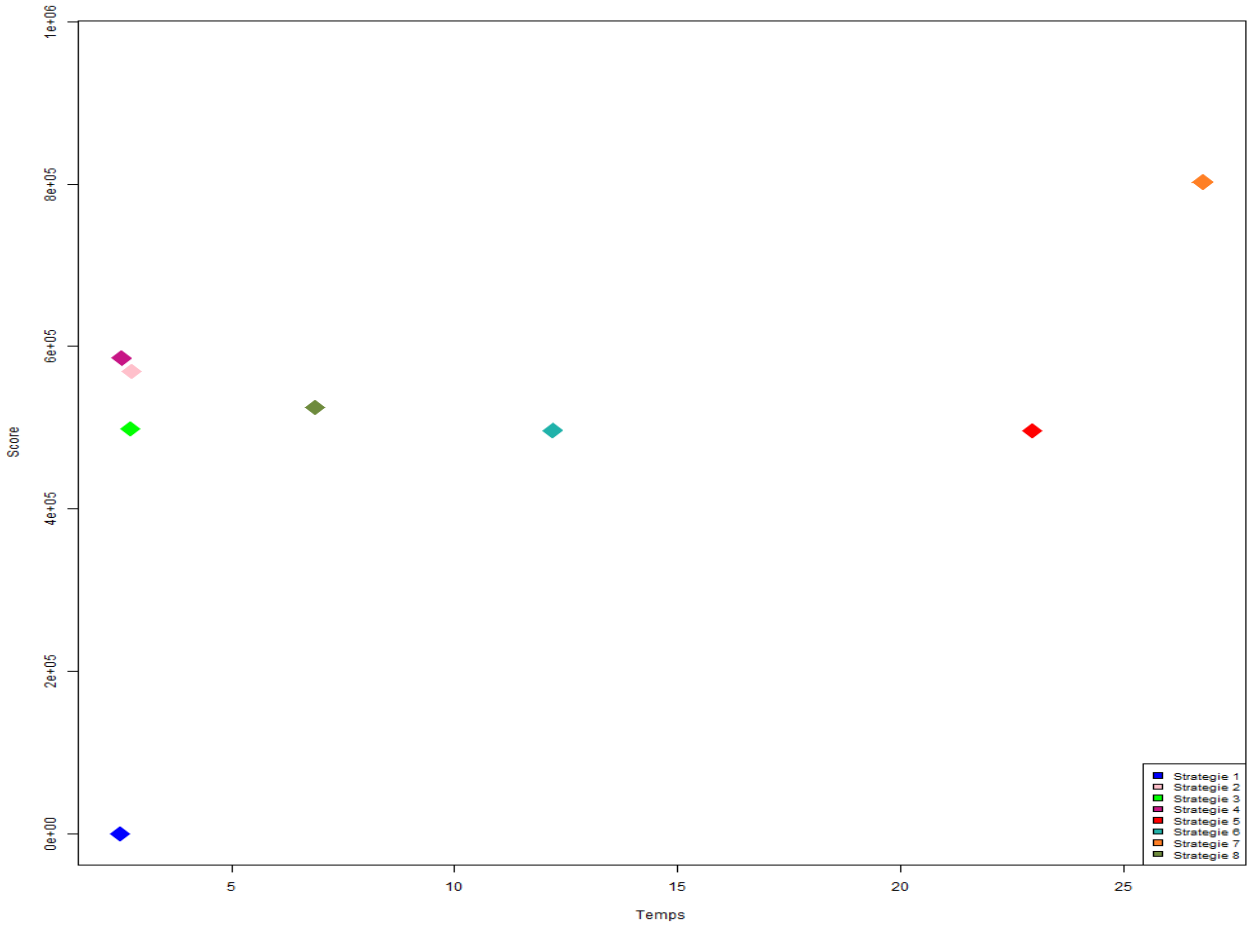
Dans le cas de kitten.in, toutes les stratégies intelligentes ne sont pas mieux que les stratégies triviales sauf pour la stratégie 7. Le fichier d'entrée kitten.in a plus de vidéos à placer qu'il n'y a de place dans les caches disponibles et les caches sont connecté a énormément de endpoint. Contrairement aux stratégies, la stratégie 7 adapte le contenue des caches qu'elle parcoure par rapport au contenue des caches déjà remplie. C'est pourquoi elle arrive à obtenir un score aussi élevé par rapport aux autres.

Voici le graphique du temps d'exécution par stratégie du fichier data.in :



Dans ce graphique, nous pouvons voir le temps d'exécution pour chaque stratégie sachant que ce temps prend en compte la lecture du fichier kitten.in.

Voici le graphique entre le score et le temps d'exécution par stratégie avec le fichier data.in :



Ce graphe permet de trouver un compromis entre le score et le temps d'exécution, comme par exemple la stratégie 4. De plus, elle permet aussi de choisir une stratégie en fonction des besoins du client. Si le client veut une stratégie ayant le meilleur score, sans tenir compte du temps d'exécution, il choisira la 7. Au contraire, si le client demande un très bon temps d'exécution il prendra la stratégie 4.

IV. Milestone

L'emploi du temps créé en début de projet nous a beaucoup aidé dans notre organisation, et a été bien respecté, nous avons chaque jour bien atteint les objectifs initialement définis. Nous n'avons pas eu besoin de le changer au cours du projet.

Lundi :

- Architecture générale du projet (Créations des trois modules)
- Mise en place de l'architecture du module Engine
- Traitement du fichier donné en entrée
- Documentation sur le langage R
- Création de stratégies triviales

Mardi :

- Mise en place du module Benchmark
- Script pour la visualisation en langage R
- Fin du développement des stratégies triviales
- Tests unitaires des différentes méthodes
- Développement de stratégies intelligentes

Mercredi :

- Fin de la mise en place du Benchmark
- Fin du développement du script de traitement pour la visualisation
- Développement de stratégies intelligentes
- Préparation de la soutenance du projet

Jeudi :

- Tests unitaires des différentes méthodes
- Développement de stratégies intelligentes

Vendredi :

- Amélioration de la visualisation
- Développement de stratégies intelligentes
- Rédaction du rapport de projet

Si nous devions modifier quelque chose sur l'organisation de ce projet, nous consacrerions plus de temps à la création de l'architecture du projet, à son départ.

Les feedbacks journaliers ont permis d'améliorer de nombreux éléments :

- La construction de notre architecture,
- La mise en place du Benchmark dont la mise en place a été complexe,
- La préparation de notre soutenance orale,
- Mise en place d'un emploi du temps intelligent,
- Corrections méthodes de tri

V. Utilisations d'algorithmes, de Maven, de git et de la présentation

Pour ce projet, les classes java ont permis de récupérer les informations données par les fichiers d'entrée, de les stocker par objet, et de pouvoir y accéder facilement.
Ensuite, nous avons fait des algorithmes pour les stratégies, afin d'obtenir les meilleurs scores possibles, en un temps de lancement faible. L'algorithme ici permet de développer des stratégies très complexe.

Git est très utile, voir même indispensable pour un projet de ce genre : il permet de travailler en groupe, de partager son travail, et d'avancer avec une version commune. Par contre, nous avons rencontré un problème avec git à propos des majuscules et minuscules : la modification du nom de dossier n'est pas prise en compte par git. Exemple : renommage "teamA" en "teama".

Maven est un outil qui nous a été très utile car il permet de gérer toutes les dépendances entres les différents modules. Ils nous étaient indispensable pour ce projet puis que les modules benchmark et visualiser avait tout deux besoins d'interagir avec le module engine. De plus, il nous permet d'effectuer des tests unitaires afin d'en assurer sa solidité.

Grâce au feedback de la présentation nous avons pu améliorer certain point de notre projet. Comme par exemple de passer toutes les méthodes des classes stratégie en protected afin de sécuriser l'accès à ces méthodes et diversifier les graphiques générés par le module visualiser.

VI. Répartition des tâches

Pour le projet, nous nous sommes réparti différentes tâches au fur et à mesure du projet. Les derniers jours, nous nous sommes tous concentrés sur le développement des stratégies.

Voici notre répartition des tâches pour le projet :

Théos :

- Création de l'architecture du projet : les classes Controller, Cache, Request, Endpoint, et Vidéo, permettant de conserver et manipuler les différents objets.
- Documentation et mise en place du benchmark.
- Développement de stratégies intelligentes.

Alexis :

- Mise en place du git et des 3 modules dans le projet.
- Création de la classe permettant d'obtenir le fichier de sortie.
- Développement de stratégies triviales et intelligentes.

François :

- Documentation et mise en place de la visualisation R.
- Développement de stratégies triviales et intelligentes.

Florent :

- Création de la classe Input Splitter permettant la récupération du fichier d'entrée, et d'en récupérer toutes les informations utiles (caches, requêtes, vidéos, …)
- Tests unitaires Maven des classes du projet.
- Développement de stratégies intelligentes.

Notre répartition des scores sur l'investissement des membres dans le projet est le suivant :

Noms	Points
Théos	103
Alexis	99
François	99
Florent	99

Tous nos membres ont investi beaucoup de temps pour ce projet durant cette semaine, de manière régulière et équitable. Théos a été le membre ayant investi le plus temps, en apportant son aide plusieurs fois aux autres membres sur leurs parties.