Évaluation et Analyse des Performances

Benchmark memo

Hayaat Hebiret et Denis Beauget

Introduction

Attention ce memo n'est là que pour avoir une idée de la démarche mise en place, un rendu plus complet et détaillés (Respectant le déroulement du sujet) sera rendu pour le 17 décembre. Ce memo ne reprend que les éléments clé de notre rapport et permet au correcteur d'avoir une idée globale du système pour nous faire un retour clair et rapide de l'état du projet.

Dans le cadre de l'UE HAI914I nous devions, dans un premier temps, mettre en place un évaluateur de requêtes en étoile fonctionnel. Dans l'objectif de pouvoir tester notre système et de pouvoir le comparer à d'autre systèmes nous allons à travers ce memo mettre en place une démarche de test, un "Benchmark", pour tester notre système de la manière la plus précise possible tout en pouvant extraire un maximum de métrique.

Préparation des bancs d'essai

Pour la génération des données brut pour notre système nous avons utilisé **WatDiv** [a], il permet de générer des données en respectant des templates et des règles de génération simple (comme le nombre de patrons utilisés par exemple). La limite de WatDiv est due au fait que nous ne pouvons pas imposer certaines conditions comme le nombre de requêtes sans réponses ou le nombre de doublons généré ce qui dans le cadre d'un benchmark souhaitant se rapprocher au maximum d'une utilisation réaliste du système n'est pas souhaitable. (En effet, il y a peu de chances que X utilisateurs envoient la même requête dans un laps de temps très cours.)

Chiffre clés et mise en place

Nous devions maintenant faire des "choix" sur les données générés avec WatDiv pour obtenir des données "propre" et prête à être exécuté sur nos environnements :

1. Seulement **10%** du jeu de données sera des requêtes sans réponses.
2. Chaque requête du système n'aura le droit qu'à **un seul doublon** (un doublons est définis par la similitude des critères de séléction, il est donc possible que des requêtes de tailles différentes soit considéré comme les même si, par exemple, la requête la plus longue à deux fois le m.

L'objectif est d'obtenir un jeu de données se rapprochant au maximum d'une situation réelle, nous avons jugé que 10% de requêtes sans réponses était une moyenne correcte et un bon compromis entre complexité et réalité. L'idée est la même pour les doublons en permettant à chaque requête de n'avoir qu'un seul doublon associé à la génération de WatDiv nous obtenons suivant les cas entre 5% et 10% de doublon après nos traitements.
Ces traitements annexe s'exprime à travers des fonctions directement intégrées à notre code.

Rappel du Git : `https://github.com/Beauget/HAI914I_Projet`

Environnement utilisé

Pour l'exécution du benchmark, nous avons décidé d'utiliser 3 machines et plus précisément d'en comparer 2. Nous utiliserons 2 machines personnel sous Linux (les informations des 2 systèmes sont disponibles ci-dessous.). Dans une démarche plus expérimentale et, car nous en avons l'occasion nous lancerons nos tests sur une machine sous Windows 11 sans mélanger les résultats avec les 2 machines précédentes (car ça n'aurait pas de sens).

Processor	Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz
Memory	3896MB (2223MB used)
Machine Type	Notebook
Operating System	Ubuntu 20.04.3 LTS
User Name	hayaat (Hayaat)

FIGURE 1 – Première machine Linux.

Processeur	Intel(R) Pentium(R) Silver N5000 CPU @ 1.10GHz
Mémoire	7626MB (2025MB utilisé)
Machine Type	Notebook
Systeme d'exploitation	Ubuntu 20.04.3 LTS
Utilisateur	leo (Léo)

FIGURE 2 – Deuxième machine Linux.

```
[Résumé système]
Nom du système d'exploitation   Microsoft Windows 11 Famille
Modèle  Inspiron 15 3511
Type    PC à base de x64
Référence (SKU) du système      0AB0
Processeur      11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz, 2419 MHz, 4 cœur(s), 8 processeur(s) logique(s)
Mémoire physique (RAM) installée : 8,00 Go
```

FIGURE 3 – Machine Windows.

Premier histogramme

Dans le sujet il était conseillé de créer un premier histogramme avec les requêtes issus de **WatDiv** en utilisant 500K et 2M de triples.
Cependant nous utilisons pour l'instant que 100k et 500k.
Après avoir généré nos requêtes, nous appliquons les critères énoncés précedemment sur plusieurs volumes de données afin de déterminé qu'elle combinaison est la meilleur.
Il est important de noter que seulement 4 templates sur 13 sont correctement généré par WatDiv : Q_1_includes, Q_1_likes,Q_1_subscribes et Q_2_subscribes_like. Dû à la taille des requêtes, on à une forte probabilité de duplications

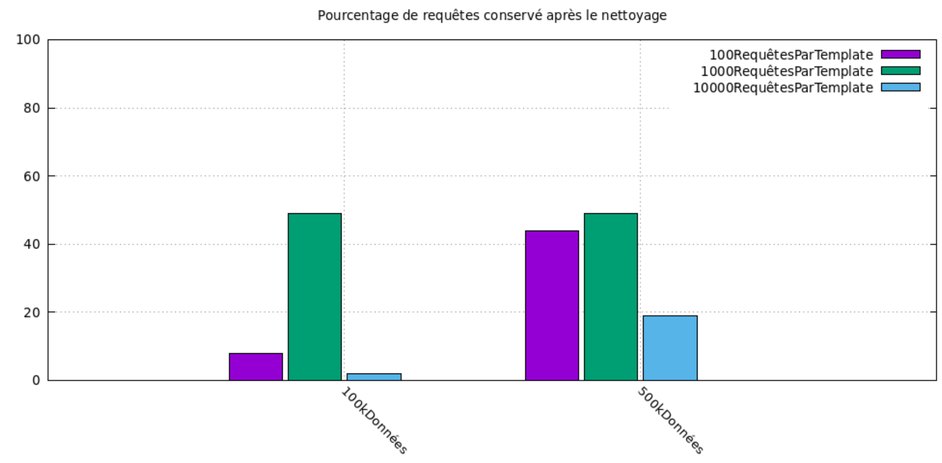

FIGURE 4 – Histogramme obtenu.

On peut supposer qu'un grand volume de données combiné à un volume raisonnable de requêtes permet de limiter la redondance et les requêtes sans réponse.

Résumé

L'intégralité de notre benchmark est maintenant prêt :
Nous avons définis des règles pour la génération des requêtes avec **WatDiv**

1. **10%** de requêtes sans réponses
2. **1 seul doublon** par requête

Nous avons choisi les environnements à comparer, au nombre de 3 dont 1 n'est qu'un "bonus" qui ne sera pas intégré au résultat final car incomparable avec un environnement Linux. Ces environnements permettent de mettre en avant nos différents facteurs et niveaux (principalement le CPU et la mémoire, mais également le système d'exploitation en facteur secondaire).

1. 2 machines Linux distinctes pour le comparatif général.
2. 1 machine Windows 11 non intégré au comparatif général.

Maintenant, en cas de validation de notre démarche décrite précédemment nous passerons notre fichier de requêtes générés avec **WatDiv** à notre implémentation Jena. Nous pourrons comparer les résultats et en déduire une interprétation graphique dans le rendu final du projet.

a. https://dsg.uwaterloo.ca/watdiv/