Rapport de TD 2 : Évaluation et Analyse des Performances

Azevedo Gomes - Prud'homme Gateau

Table des Matières :

1. Introduction

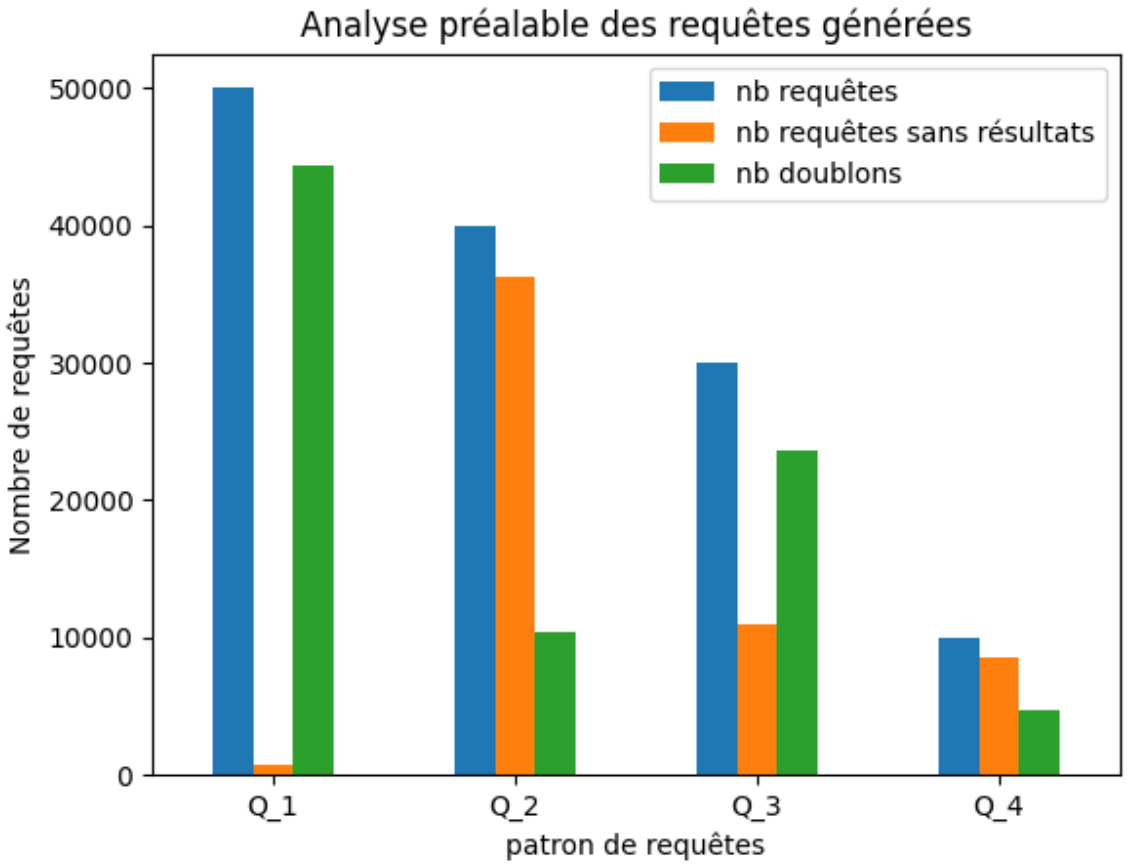
L'objectif de ce deuxième TD est d'évaluer et analyser les performances de notre Moteur de requêtes commencé lors du premier TD. Nous utiliserons le benchmark WatDiv pour générer nos données et requêtes ainsi que la dernière version de Jena pour vérifier la correction de nos résultats.

Le lien vers le code développé est le suivant : https://github.com/dneial/mini-rdf

2. Préparation des bancs d'essais

a. Analyse Préalable

Grâce à WatDiv nous avons pu générer des données et des requêtes en grande quantité et avec des patrons différents. Nous les avons toutes réunies en un seul fichier de requêtes que nous devons analyser préalablement pour vérifier qu'il correspond à nos attentes.



Les résultats obtenus nous montrent qu'une quantité importante de doublons est présente parmi nos requêtes. On constate aussi qu'une quantité significative de requêtes n'ont pas de réponse.

Ces deux aspects ne sont pas forcément souhaitables pour un système censé simuler un cas réel d'utilisation de moteur de requêtes, nous devons donc faire des choix sur le contenu de notre fichier de benchmark.

b. Choix des requêtes

Pour assembler un benchmark pertinent nous avons pris la décision de réduire le nombre de doublons et de requêtes sans réponses. Cependant retirer drastiquement tous ces éléments auraient eu pour conséquence de limiter la variété des patrons utilisés, notamment les patrons de requêtes contenant 4 conditions. Nous avons donc conservé à peu près 30% de requêtes sans réponse pour ce patron particulier.

À la fin de cette étape nous avons un jeu de requêtes contenant un total de 8600 requêtes qui correspondent à nos attentes pour un benchmark

3. Hardware et Software

Notre benchmark à été testé sur une machine ayant les spécifications suivantes

- Modèle : HP Notebook - 17-bs105nf
- Processeur : Intel Core i5-8250U CPU @ 1.60GHz ;
  6 Mo de mémoire cache, 4 cœurs
- RAM : 12,0 GB,
- Disques : SATA 1 To, 5400 tr/min
  SSD M.2 128 Go

Le système d'exploitation suivant :

- OS : XUbuntu 23.04 Lunar

L'application réalisée est codée en Java, puis compilée et exécutée en ligne de commande avec le jdk : openjdk 17.0.9 2023-10-17
Ce système n'est cependant pas représentatif d'un environnement d'exécution adéquat pour ce type d'activité, un réel serveur de données aurait de bien meilleures spécifications et un système optimisé pour la tâche du traitement de données et de requêtes.

4. Métriques, Facteurs et Niveaux

a. Métriques

Pour évaluer son environnement d'exécution, un benchmark doit faire état de différentes métriques. Dans notre cas on étudiera :

- Le temps de lecture des données
- Le temps de lecture des requêtes
- Le temps d'exécution des requêtes
- Le temps total de l'application

b. Facteurs

Plusieurs éléments peuvent influencer ces différentes mesures, nous avons choisi de nous concentrer sur la mémoire allouée au programme et le nombre de données dans la base.

Nos tests sont donc réalisés avec 2 et 4 Go de mémoire allouée pour le tas de la machine virtuelle Java via la commande : *-Xmx{qté de mémoire}g*

Et nous testons deux jeux de données générés via WatDiv de 500.000 et 2.000.000 de triplets RDF.

Nous avons donc 4 configurations de tests possibles :

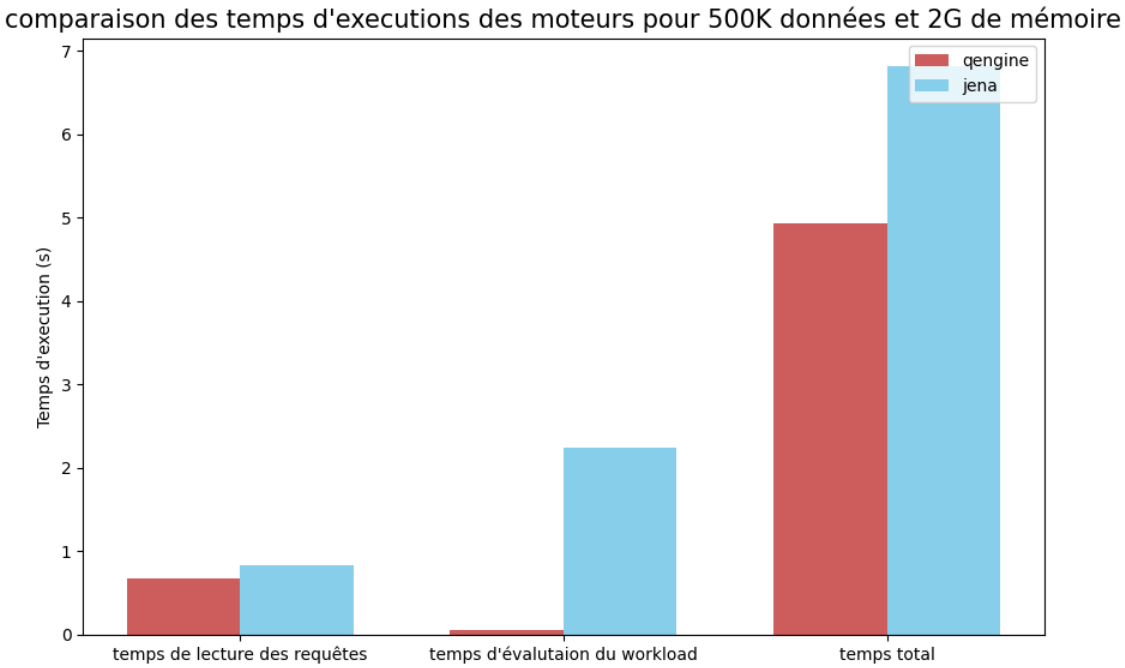
- 2Go de mémoire; 500K données
- 4Go de mémoire; 500K données
- 2Go de mémoire; 2M données
- 4Go de mémoire; 2M données

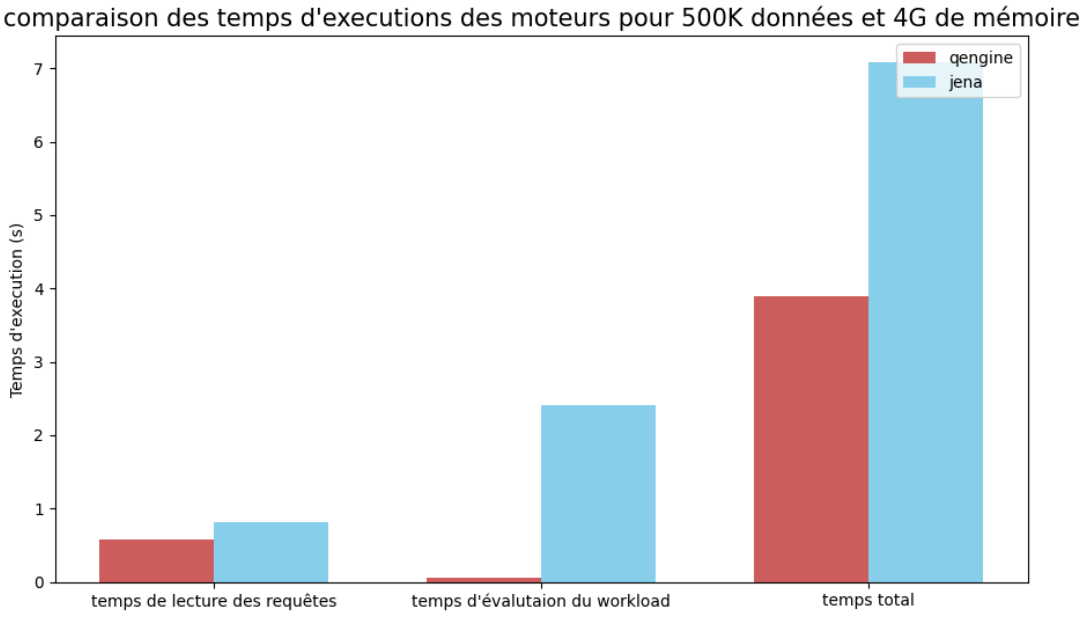
5. Vérification de la correction et de la complétude

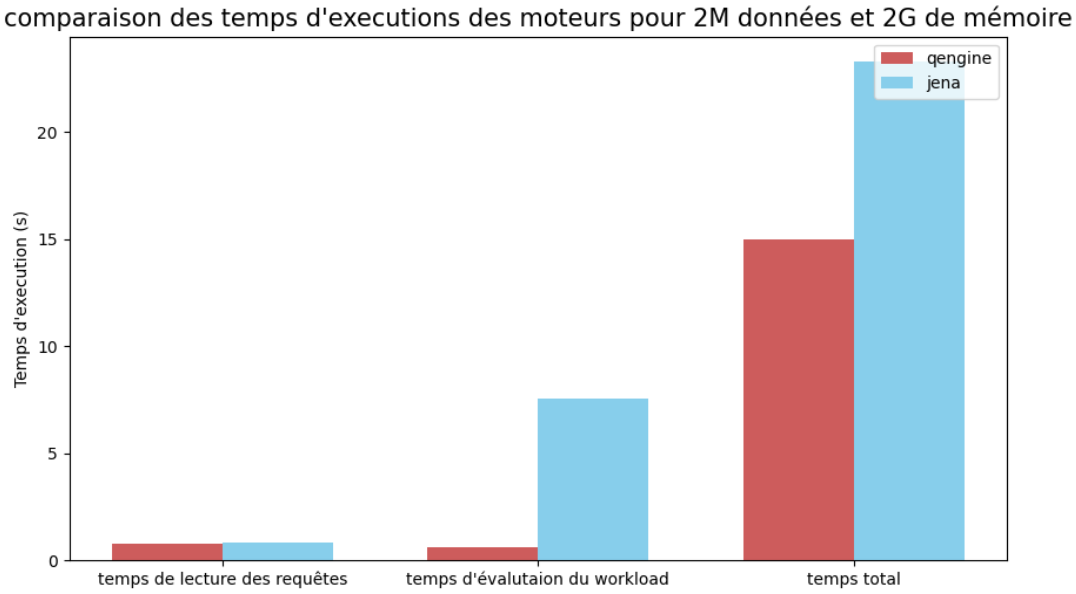
Au cours du TP précédent nous avons pu vérifier la correction de notre moteur en comparant ses résultats avec ceux d'un autre moteur de requêtes : "Apache Jena" que nous utilisons lors des tests de notre benchmark.
Un script à été élaboré afin d'exécuter à la suite notre moteur et celui de Jena, avec les 4 configurations définies dans la section Facteurs. Les métriques d'évaluation des deux systèmes sont également enregistrées dans un fichier csv.

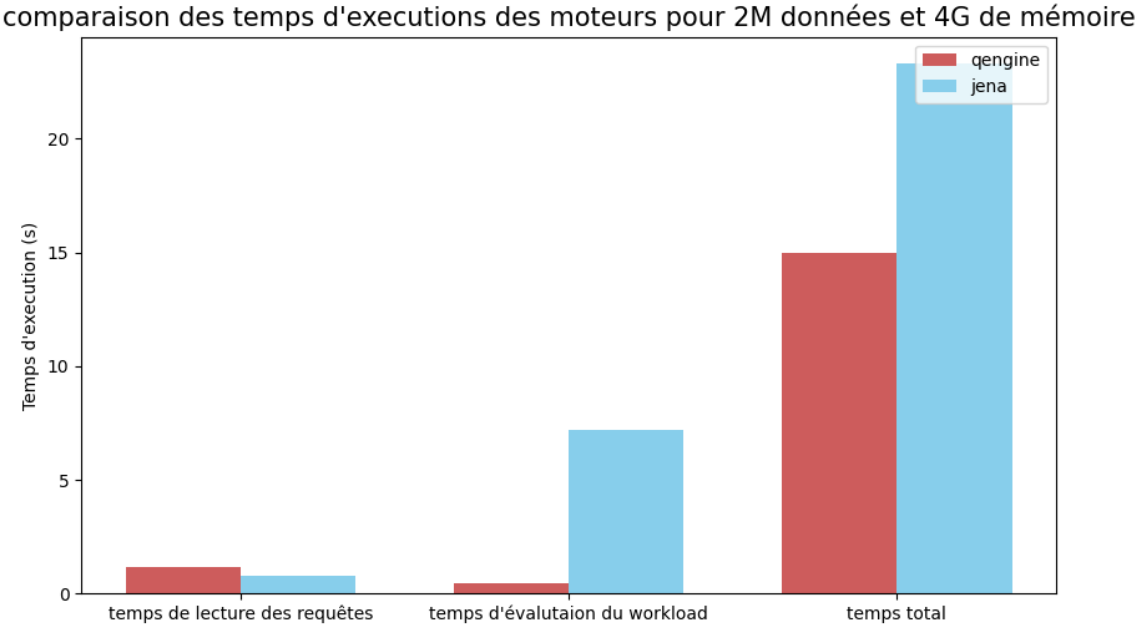
6. Comparaison avec Jena

Les résultats de la comparaison des métriques sont les suivants :









Dans la mesure du temps total est également compris le temps de lecture des données. Ces analyses nous permettent de mettre en avant la vitesse de notre moteur par rapport à celui de Jena, on constate aussi une légère augmentation des temps avec l'augmentation de la quantité de mémoire allouée et une diminution drastique en

augmentant la taille des données. Ces deux aspects seront étudiés plus en profondeur dans les sections suivantes.

Si les temps de Jena sont plus élevés celà peut s'expliquer par le fait que ce moteur est plus complet dans sa conception tandis que le nôtre est voué à une application ciblée. De plus, lors de la mise en place des mécanismes de mesure on peut être extrêmement précis dans notre moteur en ayant accès au code source, chose que l'on à pas en utilisant Jena.

7. Protocole de test

Le lancement de notre benchmark se fait par l'intermédiaire du script "benchmark.py". Ce dernier demande à l'utilisateur la quantité de mémoire qu'il souhaite allouer au tas de la JVM ainsi que le fichier de données à utiliser parmi les deux que nous avons généré. L'exécution de notre moteur exporte ensuite les résultats obtenus dans le fichier "time_monitoring.csv"

8. Importance des facteurs avec un modèle de régression non-linéaire

Impact des tailles de la mémoire et du nombre de triplets sur les performances		
Données\Mémoire	2Go RAM	4Go RAM
500.000 triplets	4166 ms	3913 ms
2.000.000 triplets	18512 ms	14984 ms

Pour interpréter les résultats de ce tableau nous utilisons un modèle de régression non-linéaire.

On définit la variable Xa:

- Xa = -1 si 2Go de RAM
- Xa = 1 si 4Go de RAM

Et la variable Xb:

- Xb = -1 si 500000 triplets
- Xb = 1 si 2000000 triplets

A partir de l'équation de base:

$$y = q_0 + q_a X_a + q_b X_b + q_{ab} X_a X_b$$

Et des valeurs de la table, on obtient le système de 4 équations à 4 variables suivant :

- $4166 = q_0 - q_A - q_b + q_{ab}$
- $3913 = q_0 + q_A - q_b - q_{ab}$
- $18512 = q_0 - q_A + q_b - q_{ab}$
- $14964 = q_0 + q_A + q_b + q_{ab}$

La résolution de ce système nous donne le modèle de régression suivant:

$$y = 10388.8 - 950.25X_a + 6349.25X_b - 823.75X_{ab}$$

Que l'on peut interpréter de la manière suivante :

- Le temps de réponse de base est de 10388.8ms
- L'augmentation de la taille de mémoire améliore le temps de réponse de 950.25ms
- Une augmentation des données abaisse les performances de 6349.25ms
- L'interaction entre la mémoire et les données affecte les performances à hauteur de 823.75ms