INF8480 – Systèmes répartis et infonuagique

Hiver 2019

TP2
Services distribués et gestion des pannes

Groupe 2

1791314 – Simon Nadeau
1846732 – Mathieu Châteauvert

Soumis à : Daniel Capelo & Adel Belkhiri

18/03/2019

Choix de conception et d'implémentation

Pour la division des tâches, la gestion des tâches échouées, la détection de pannes et la gestion d'erreurs d'authentification et d'accès au serveur de noms, nous avons choisi d'implémenter toute ces fonctionnalités directement dans le répartiteur. La première chose que le répartiteur fait lors du calcul est donc de faire l'acquisition des informations utiles sur les serveurs de calcul fonctionnels.

Ensuite, nous avons rapidement compris que le répartiteur devrait avoir une structure lui permettant d'envoyer les tâches sur les différents serveurs de calcul sans attendre que ceux-ci effectuent leurs calculs et renvoient leurs résultats. De cette manière, on peut tirer profit des serveurs de calculs pour exécuter les différentes tâches en parallèle et améliorer les performances. Pour implémenter ce choix de conception, nous avons utilisé un Executor et un ExecutorCompletionService permettant de lancer plusieurs Callable. C'est l'exécution de ses Callable qui permet d'obtenir un résultat (TaskInfo) dans lequel nous avons inclus les différentes informations utiles pour le calcul du résultat final, la gestion des tâches échouées, la détection de pannes et la gestion d'erreurs d'authentification et d'accès au serveur de noms. À partir du statut contenu dans le résultat de l'exécution d'une tâche, nous pouvons donc adapter le travail à effectuer en fonction de la situation.

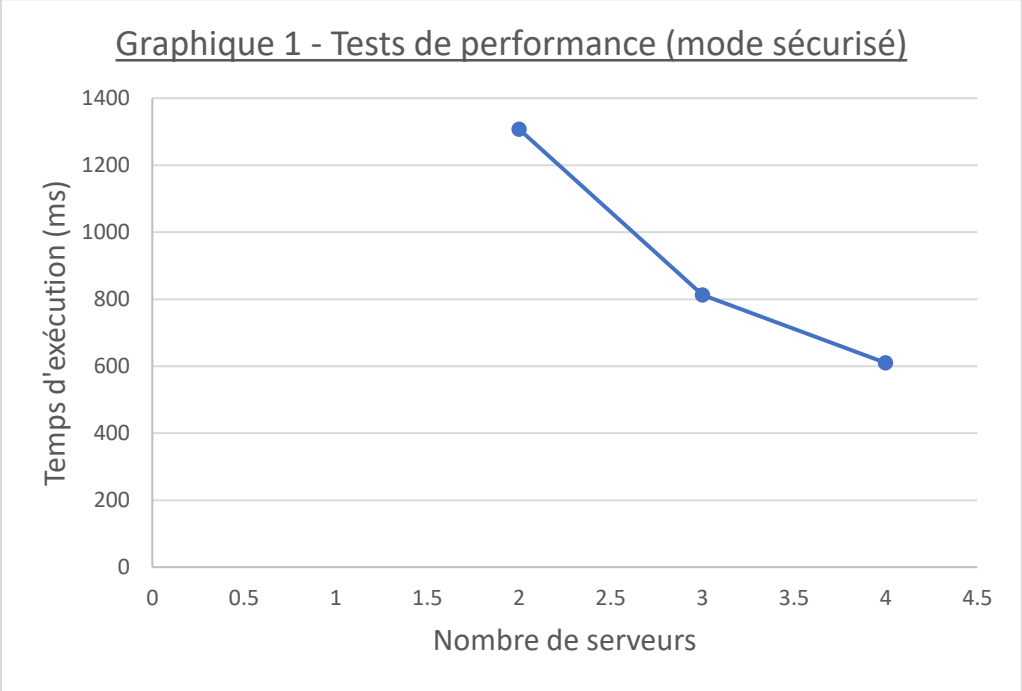
Pour la gestion du cas d'une tâche refusée, d'une panne, d'une erreur d'authentification ou d'une erreur d'accès au serveur de noms, le répartiteur l'effectue de la même façon en mode sécurisé et non-sécurisé. Pour la gestion du cas d'une tâche refusée, le répartiteur ne fait que renvoyer la tâche sur le même serveur de calcul, puisqu'éventuellement elle sera acceptée. Pour la gestion d'un serveur de calcul qui tombe en panne, le répartiteur met à jour sa liste d'informations sur les serveurs de calcul fonctionnels auxquels il a accès. Pour la gestion d'une erreur d'authentification ou d'une erreur d'accès au serveur de noms, le répartiteur cesse l'exécution du programme. Comme toutes les informations nécessaires se retrouvent dans l'objet TaskInfo créé à cet effet et retourné à la suite de l'exécution d'une tâche, ces manières de faire la gestion de différents statuts nous semblaient être les plus simples.

Pour la gestion du cas où le calcul s'effectue complètement sur un serveur de calcul, l'implémentation diffère entre le mode sécurisé et le mode non-sécurisé. Pour le mode non-sécurisé, le répartiteur ne fait qu'accumuler le résultat puis lancer une nouvelle tâche. Pour le mode sécurisé, le répartiteur emmagasine le résultat et relance la tâche sur un serveur différent jusqu'à ce qu'il obtienne deux résultats identiques.

En ce qui concerne l'implémentation des serveurs de noms et de calcul, nous l'avons fait la plus simple possible. Le serveur de noms ne fait vraiment que recueillir les informations sur les serveurs de calcul et authentifier le répartiteur et les serveurs de calcul ne font qu'effectuer les calculs des opérations qui leurs sont fournies.

Tests de performance – mode sécurisé

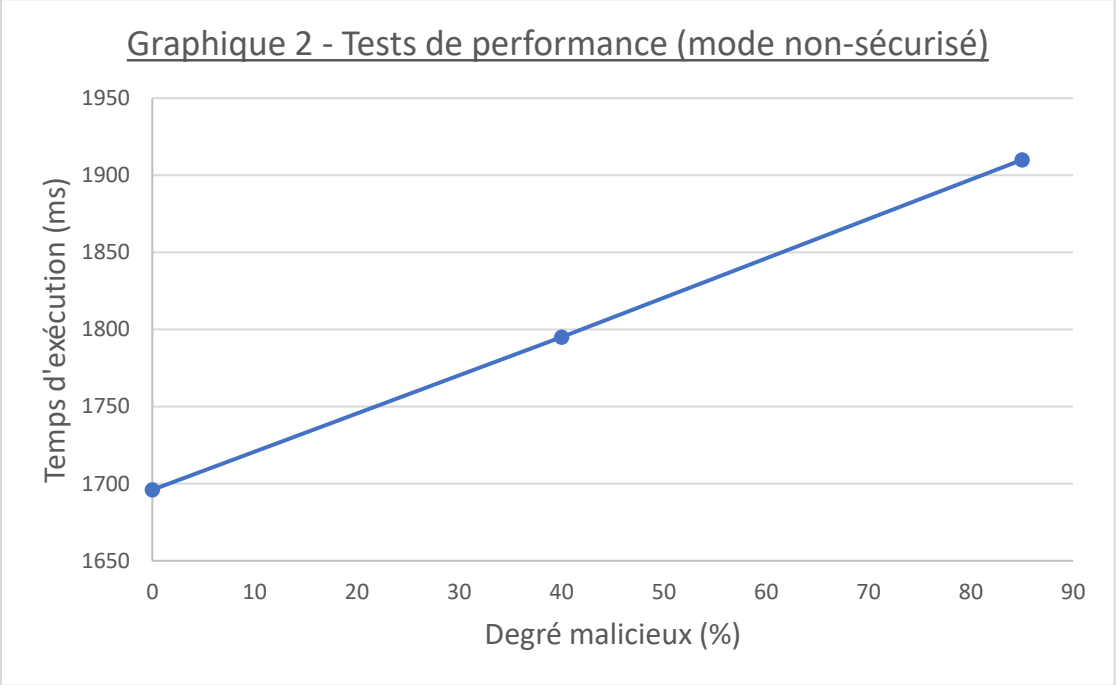
Voici le graphique du temps d'exécution (pour un fichier de 500 lignes d'opérations) en fonction du nombre de serveurs :



On remarque que le temps d'exécution du calcul complet diminue plus le nombre de serveurs augmente. Cela est normal, car, comme nous lançons plusieurs tâches de calculs partiels s'exécutant en parallèle, plus il y a de serveurs de calculs, plus il peut y avoir de tâches en parallèle. Toutefois, cette diminution n'est pas nécessairement linéaire. Cela s'explique par le fait que la récupération du résultat de chaque calcul partiel et le lancement de nouvelles tâches se fait de manière séquentielle. Cette limitation du répartiteur fait donc en sorte que plus le nombre de serveurs de calcul sera grand, moins l'ajout d'un serveur de calcul supplémentaire améliorera les performances.

Tests de performance – mode non-sécurisé

Voici le graphique du temps d'exécution (pour un fichier de 500 lignes d'opérations) en fonction du paramètre m du serveur malicieux :



On remarque que le temps d'exécution du calcul complet augmente plus le serveur malicieux renvoie de résultats erronés. Cela est tout à fait normal, car plus le taux de résultats erronés est élevé, plus la validation s'effectue difficilement. De plus, il semble normal que cette augmentation soit pratiquement linéaire, car le nombre de tâches supplémentaires qui doivent être effectuées varie directement en fonction du nombre de résultats erronés générés. On remarque également que tous les temps d'exécution sont plus élevés qu'en mode sécurisé. Cela est normal, car chaque tâche doit être effectuée au moins deux fois pour valider le résultat. D'ailleurs, à trois serveurs 0% malicieux, le temps d'exécution est environ deux fois celui du même calcul en mode sécurisé avec trois serveurs.

Question de réflexion

Pour améliorer la résilience du répartiteur, une solution serait de le fait rouler sur un serveur accessible par l'intermédiaire d'une interface. Du côté serveur, un gestionnaire de répliques s'occuperait de maintenir des répliques du répartiteur à jour pour être en mesure de récupérer les données de celui-ci en cas de panne. Les requêtes du côté du client seraient alors passé par l'intermédiaire de l'interface du répartiteur qui rendrait le tout transparent.

Le principal avantage d'une telle solution est qu'elle améliore la disponibilité du répartiteur en cas de panne. En effet, il est ainsi possible d'offrir un service pratiquement 100% du temps, puisqu'un serveur répliqué peut prendre la relève au cas où un serveur tombe en panne. Un autre avantage est la transparence pour le client. En effet, l'utilisation d'une interface comme intermédiaire du côté client et d'un gestionnaire de réplication du côté serveur permettent d'assurer un système transparent.

Toutefois, un tel type de système est lourd à implémenter pour assurer la cohérence des données. De plus, un tel système affecte négativement les performances d'un point de vue d'utilisation de la mémoire à cause de la réplication des données.

Certains scénarios pourraient tout de même causer une panne du système. Par exemple, une erreur de connexion entre le client et le serveur du répartiteur empêcherait tout le système de fonctionner. Également, une erreur de coordination dans la réplication du serveur du répartiteur entraînerait des erreurs du système.