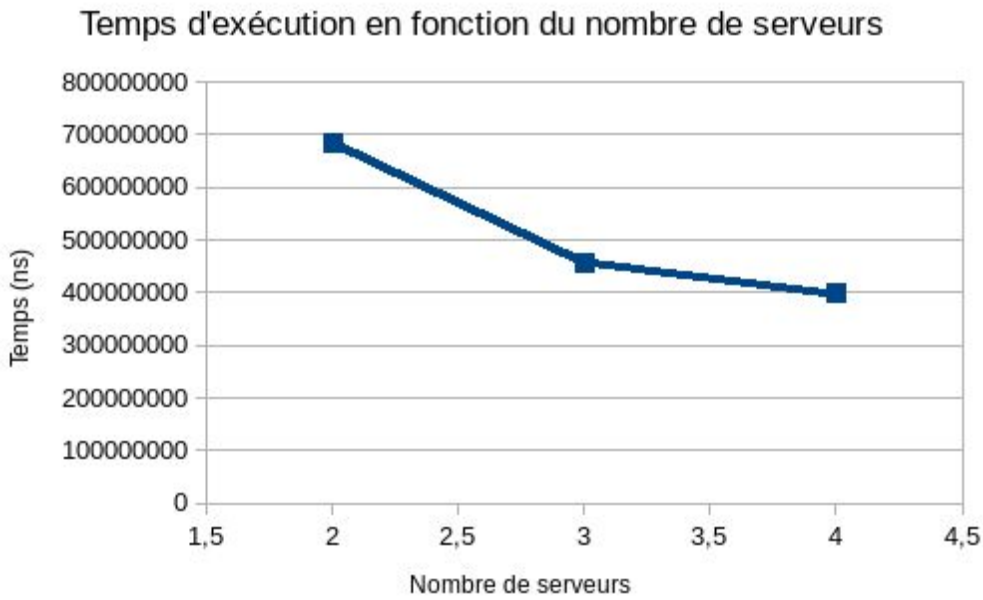
Guillaume Lavoie-Harvey (1688951)
Charles Tremblay (1687959)

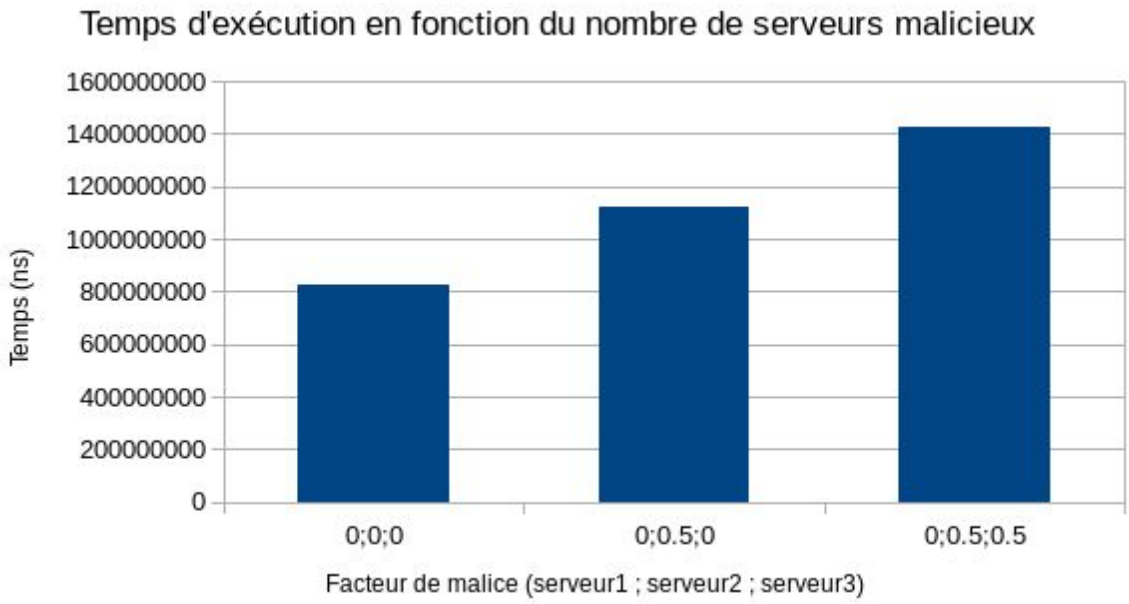
Rapport TP2 - Services distribués et gestion des pannes
INF4410 - Systèmes répartis et infonuagique

Tests de performance



(Client = répartiteur)
(Serveur = serveur de calcul)

Il est facile de remarquer que le temps avoir d'obtenir un résultat est constamment réduit en ajoutant des serveurs. En effet, avoir de plus en plus de serveurs permet de former plus de threads pour ainsi accomplir plus de tâches plus rapidement, puisque le Client réparti ses tâches aux autres serveurs.



Nous pouvons observer que plus il y a de serveurs malicieux, plus le temps avant d'obtenir une réponse augmente. Ce phénomène s'explique par l'algorithme qui exige une validation avec un autre serveur avant de confirmer qu'une réponse est adéquate. Si deux serveurs ont des réponses différentes (ou que l'un des deux serveurs échoue quelque part), les tâches doivent être re-exécutées. Par conséquent, avoir plus de serveurs malicieux augmente notamment la variance dans nos résultats et exige plus de validation.

On peut aussi observer que le temps d'attente sans utiliser la validation est notamment plus petite, puisque chaque tâche ne necessite pas de confirmation par un autre serveur.

Question 1

Une approche à considérer serait d'avoir plusieurs répartiteurs synchronisés entre eux. En d'autres mots, chacun des répartiteurs aurait la liste complète des tâches à compléter et informe les autres répartiteurs des tâches qu'il a envoyés aux serveurs. Si un répartiteur plante, les autres répartiteurs pourraient théoriquement continuer sans problème, puisque le système de synchronisation n'est qu'en place pour déterminer ce que sont les tâches déjà envoyées par autrui. Les forces de cette architecture serait qu'on pourrait permettre à des répartiteurs de planter, car on aurait au moins un répartiteur pouvant prendre la relève (à l'exception du cas que tous les répartiteurs meurent). Par contre, le système de synchronicité pourrait demander plus de vérifications que la version de notre TP. De plus, on ne gérerait pas le cas où tous les répartiteurs plantent.

Choix de conception:

Tout d'abord, la répartition des tâches entre les différents serveurs est faite de manière entièrement aléatoire. Bien sûr, il ne s'agit pas d'une stratégie très efficace, mais elle fonctionne sans problème. De plus, une telle stratégie tend, dans le long terme, à répartir la tâche équitablement entre tous les serveurs disponibles. Toutefois, notre algorithme pourrait être plus efficace si la répartition de tâches prenait en compte la capacité mesurée des différents serveurs de calcul ainsi que leur occupation actuelle.

Pour ce qui est de déterminer la capacité de chaque serveur, le répartiteur commence son exécution en présumant que la capacité de chaque serveur disponible est de 3. Ce chiffre a été choisi arbitrairement pour correspondre aux données des tests demandés mais pourrait être ajustée pour une situation réelle. À chaque fois qu'un serveur de calcul renvoie une réponse valide, on augmente sa capacité présumée de 1. Si le serveur renvoie une erreur, on diminue sa capacité de 1. Cette technique offre 3 avantages: Premièrement, elle permet d'Ajuster dynamiquement la capacité des serveurs, si jamais elle pouvait changer pendant l'exécution. Deuxièmement, avec la formule utilisée pour simuler le refus de tâche, cette technique obtient un taux de refus de 50% lorsque les tâches sont 3 fois la taille maximale acceptée, ce qui signifie que la taille de nos tâche tendra vers cette valeur, et que bien que 50% seront refusées, la moitié de 300% représente quand même une taille de tâche moyenne de 150% effectuée par le serveur. Troisièmement, cette technique permet de gérer les pannes de serveur puisqu'une interruption du serveur est considérée comme une erreur, ce qui diminue la capacité de ce dernier. Ceci fera rapidement descendre la capacité d'un serveur non-disponible à 0, ce qui le rendra inaccessible au répartiteur.

Ces choix sont assez clairement non-optimaux, mais ils ont été implémentés car ils offrent un compromis entre efficacité et simplicité d'implémentation. Il aurait toutefois été possible d'implémenter quelques techniques plus avancées, telles qu'une prise en compte de la charge des serveurs ou un système de réputation afin de détecter les serveurs malicieux.