INF4410 – Systèmes répartis en infonuagique
TP2 – Services distribués et gestion des pannes

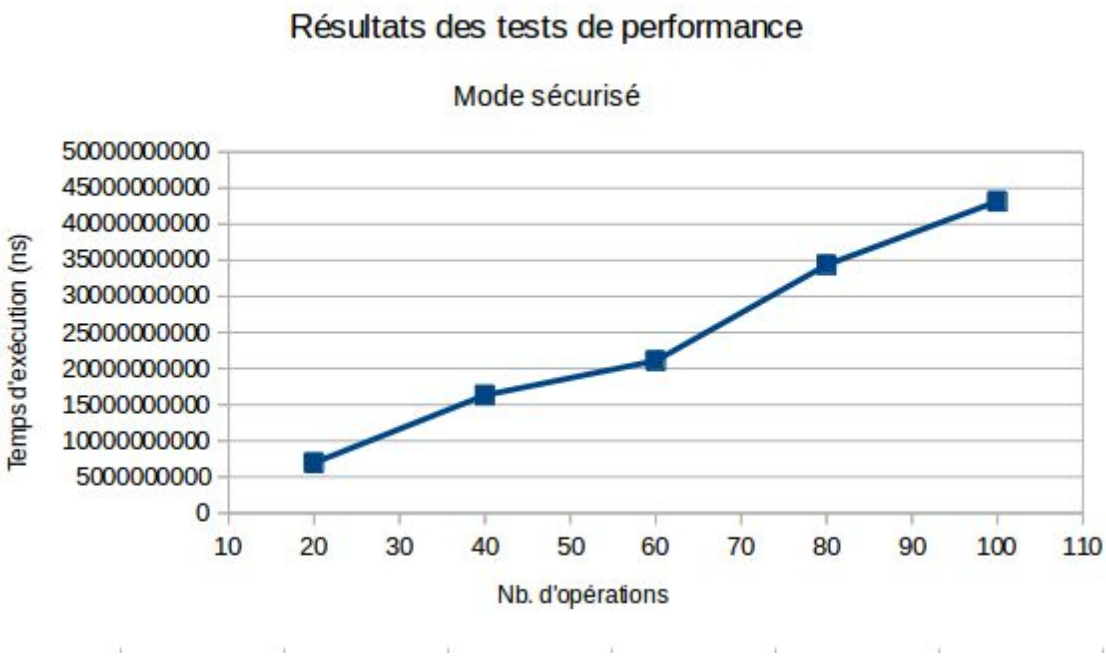
Louis Pépin – 1651555
Pier-Luc Morissette – 1588511

Question 1)

Une autre architecture possible serait d'avoir un serveur dédié au répartiteur auquel se connecte un client. Le client envoie le fichier d'opérations au répartiteur qui lui renvoie le résultat final. Le répartiteur pourrait rouler sur une machine virtuelle comme quoi si la machine physique sur laquelle il est présent plante, il sera facile de le migrer. Bien sûr, si tous les serveurs disponibles plantent en même temps ou s'il n'existe aucune machine qui possède assez de ressources pour rouler le répartiteur, le programme ne fonctionnera pas bien non plus.

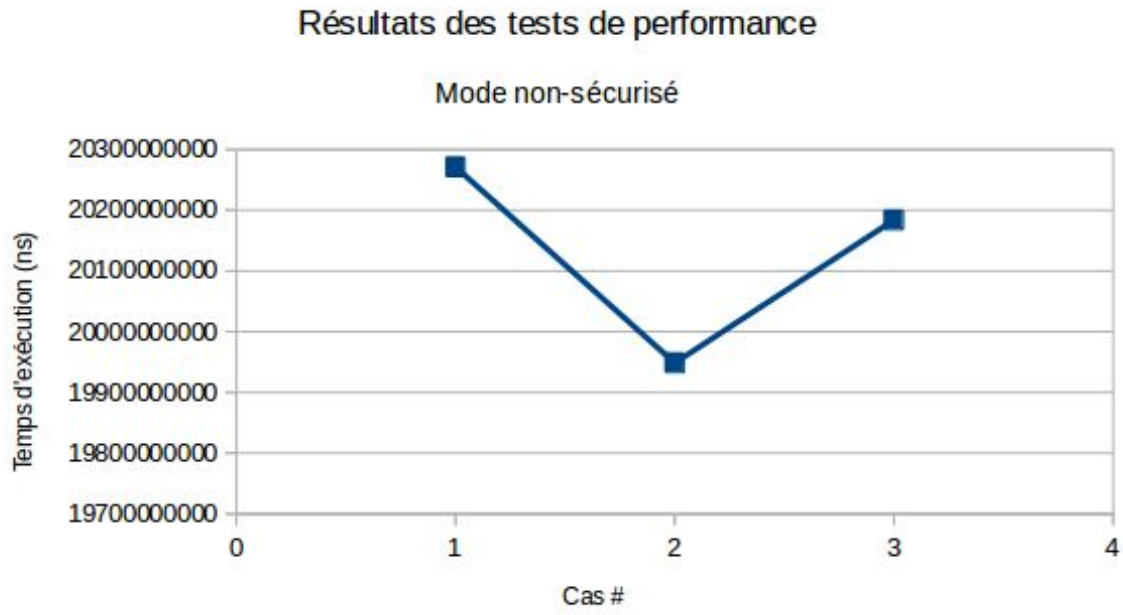
Résultats des tests de performance

Le graphique ci-dessous représente le rapport entre le temps d'exécution et la taille de tâches selon les résultats des tests de performances en mode sécurisé (voir annexe 1) :



Le graphique démontre que le temps d'exécution croit de façon linéaire en fonction de la taille de la tâche. Cela est tout à fait normal puisque notre répartiteur calcule le nombre d'opérations de la tâche et il distribue ces opérations à part égale aux différents serveurs. De plus, la capacité maximale des serveurs n'affecte pas significativement notre temps d'exécution puisque si la tâche est refusée par un serveur, le répartiteur lui envoie simplement les opérations de la tâche une à la fois. Le temps d'une requête étant significativement plus court que le temps d'effectuer l'opération, nous conservons tout de même une linéarité entre le temps d'exécution d'une tâche et sa taille.

Le graphique ci-dessous représente le rapport entre le temps d'exécution et la taille de tâches selon les résultats des tests de performances en mode non-sécurisé (voir annexe 2):



On voit que la performance ne varie pas beaucoup en fonction du serveur affecté. Cela n'est pas une surprise, car le nombre total d'instructions calculées est à chaque fois le même.

Architecture du code

Le répartiteur commence par créer autant de threads qu'il y a de serveurs de calcul disponible. Ces threads s'occupent d'alimenter en opérations les serveurs dont ils sont responsables. Le répartiteur assigne à chacun des threads une quantité égale d'opérations qu'elles devront livrer à leur serveur avant de terminer leur exécution. Une fois que tous les threads ont terminé, les résultats qu'ils ont accumulés sont compilés et le résultat final est retourné.

Le fait que les tâches enregistrent les résultats de chaque calcul rend facile la gestion des serveurs malicieux. Puisqu'on sait que chaque serveur a exécuté les mêmes opérations dans le même ordre, on peut facilement comparer les résultats de chaque serveur et de n'utiliser que celui qui est le plus fréquent.

Afin de détecter les pannes intempestives, nous utilisons deux types d'exception dans notre interface RMI. ComputingServerOverloadException qui indique que le serveur de calcul est trop occupé pour recevoir la liste d'opérations proposée, et RemoteException qui nous permet d'attraper tous les types d'exceptions RMI. Puisqu'une opération ne peut être envoyée que si le "lookup" et "bind" ont été accompli avec succès, on sait qu'une RemoteException signifie un crash du serveur en question. Malheureusement, nous n'avons pas eu le temps d'implémenter une bonne façon de gérer ces cas, mais nous arrivons à les détecter.

Annexe

Annexe 1: Résultats des tests de performance - mode sécurisé

q1 = 20 ---- q2 = 10 ---- q3 = 5

######### 20 ops #########
Found 3 computing servers
Computing in safe mode...
Starting workers...
Waiting for workers...
Compiling results...
Final result: 374, Elapsed time: 6947406916 ns.

######### 40 ops #########
Found 3 computing servers
Computing in safe mode...
Starting workers...
Waiting for workers...
Compiling results...
Final result: 2984, Elapsed time: 16318228874 ns.

######### 60 ops #########
Found 3 computing servers
Computing in safe mode...
Starting workers...
Waiting for workers...
Compiling results...
Final result: 2607, Elapsed time: 21090646640 ns.

######### 80 ops #########
Found 3 computing servers
Computing in safe mode...
Starting workers...
Waiting for workers...
Compiling results...
Final result: 887, Elapsed time: 34354121381 ns.

######### 100 ops #########
Found 3 computing servers
Computing in safe mode...
Starting workers...
Waiting for workers...
Compiling results...
Final result: 2317, Elapsed time: 43160080452 ns.

Annexe 2: Résultats des tests de performance - mode non-sécurisé

Tous les serveurs ont une capacité de 20 opérations

######### 1er cas #########
[1]   running   rmiregistry 5000
[2]   running   ./computingServer Server1 5015 20 5
[3] - running   ./computingServer Server2 5016 20 0
[4] + running   ./computingServer Server3 5017 20 0

Found 3 computing servers
Computing in safe mode...
Starting workers...
Waiting for workers...
Compiling results...
Final result: 2607, Elapsed time: 20271221719 ns.

######### 2e cas #########
[1]   running   rmiregistry 5000
[2]   running   ./computingServer Server1 5015 20 0
[3] - running   ./computingServer Server2 5016 20 5
[4] + running   ./computingServer Server3 5017 20 0

Found 3 computing servers
Computing in safe mode...
Starting workers...
Waiting for workers...
Compiling results...
Final result: 2607, Elapsed time: 19948579253 ns.

######### 3e cas #########
[1]   running   rmiregistry 5000
[2]   running   ./computingServer Server1 5015 20 0
[3] - running   ./computingServer Server2 5016 20 0
[4] + running   ./computingServer Server3 5017 20 5

Found 3 computing servers
Computing in safe mode...
Starting workers...
Waiting for workers...
Compiling results...
Final result: 2607, Elapsed time: 20183883650 ns.

######## COMMAND USED ########
java -jar dispatcher.jar
/home/pierluc/Polytechnique/Hiver2016/INF4410/carteblanche/code_tp2/donnees/benchmark/donnees-60.txt 1 localhost 5000 >> donnees/benchmark/benchmark-unsafemode-result.txt 2>&1