Tests de performances

Voici un court rapport résumant les tests de performances que nous avons effectués sur notre projet.

Veuillez noter que nous avions choisi au départ des valeurs de N (nombre de serveurs à contacter) assez petites (entre 1 et 10), et avions pris en considération l'entier de la liste de serveurs qui nous avait été mise à disposition. Cependant, nous avons constaté que plus de la majorité des serveurs ne répondaient pas. Nous avons alors pris la liberté de raccourcir cette liste, en sélectionnant, d'après nous, les serveurs les plus fiables (soit 41 sur 174). Cette réduction n'ayant pas été suffisante pour obtenir des résultats exploitables, nous avons augmenté la taille de nos N (entre 10 et 20), de manière à contacter plus de serveurs à chaque opération, et ainsi obtenir plus de succès lors de nos tentatives de put/get.

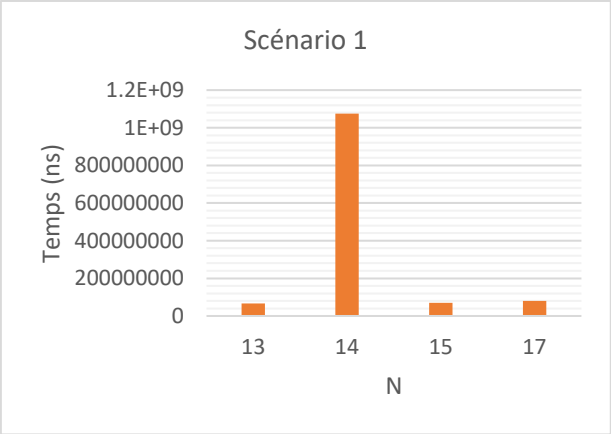
Veuillez également noter que nous avons réalisé tous nos tests depuis l'extérieur de l'école, et donc avec un VPN. Comme dit plus bas, cela a probablement légèrement affecté certains résultats.

Scénario 1 : Autant de puts que de gets, R = W = N/2 + 1, N variant de 10 à 20

Dans ce premier scénario, seulement 4 des 10 tentatives ont réussi, pour les valeurs suivantes :

- N = 13, R = W = 7
- N = 14, R = W = 8
- N = 15, R = 8, W = 8
- N = 17, R = 9, W = 9

Voici les temps analysés (en nanosecondes) :



Nous pouvons supposer que l'écart pour N = 14 est dû à un problème de connexion, étant donné le peu de différence entre les autres résultats. Cependant, avoir seulement 3 résultats à comparer n'est pas vraiment très intéressant…

Les raisons des échecs pour les autres valeurs sont les suivantes :

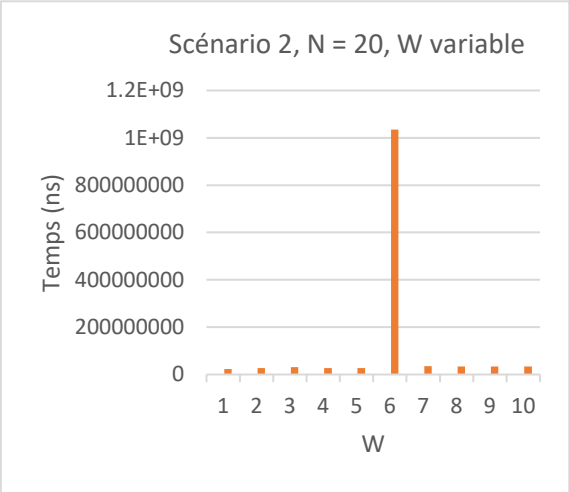
- N = 10, R = W = 6 → Echec du put
- N = 11, R = W = 6 → Echec du put
- N = 12, R = W = 7 → Echec du get
- N = 16, R = W = 9 → Echec du get
- N = 18, R = W = 10 → Echec du put et du get
- N = 19, R = W = 10 → Echec du get
- N = 20, R = W = 11 → Echec du put

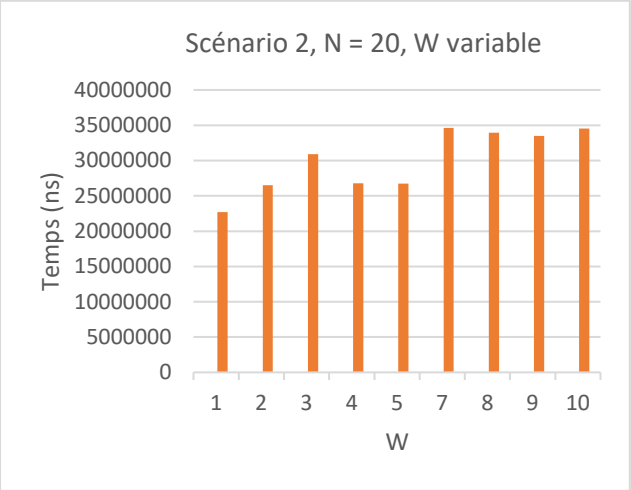
Ici, un échec uniquement du put mais pas du get était possible étant donné que nous avons ajouté à chaque tentative la même paire clé-valeur. Comme nous pouvons le constater, il n'est arrivé qu'une seule fois que le put et le get échouent en même temps. Penchons-nous donc sur le scénario suivant.

Scénario 2 : Puts uniquement, N = 20, W variant de 1 à 10

Dans ce scénario, nous avons tenté de faire uniquement des puts, à savoir 10 de suite, pour un W variant de 1 à 10, avec l'espoir d'obtenir plus de succès, ce qui fût chose faite. En effet, cette fois-ci 10 opérations sur 10 furent un succès ! Voici les temps analysés :

La valeur à W = 6 semble être due à un problème de connexion, et nous empêche d'analyser correctement le reste des données. Nous avons donc décidé de supprimer cette valeur. Voici le résultat :

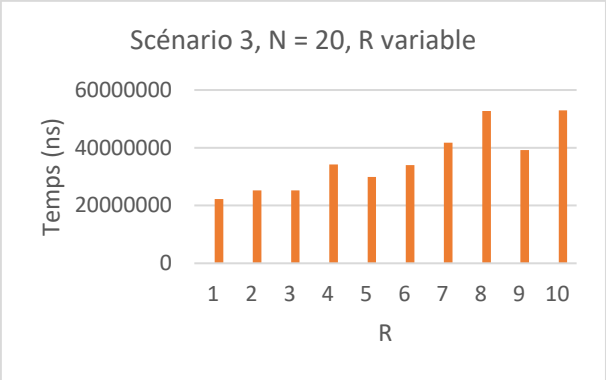




Nous pouvons constater une légère augmentation du temps en corrélation avec le W, mais globalement le système absorbe bien ce changement.

Scénario 3 : Gets uniquement, N = 20, R variant de 1 à 10

De manière similaire au scénario précédent, nous avons cette fois tenté 10 gets de suite (les serveurs contenant bien entendu une valeur pour la clé demandée), avec R variant de 1 à 10, et tout autant de succès : 10 sur 10. Voici les temps reportés, sans valeur parasite cette fois-ci :



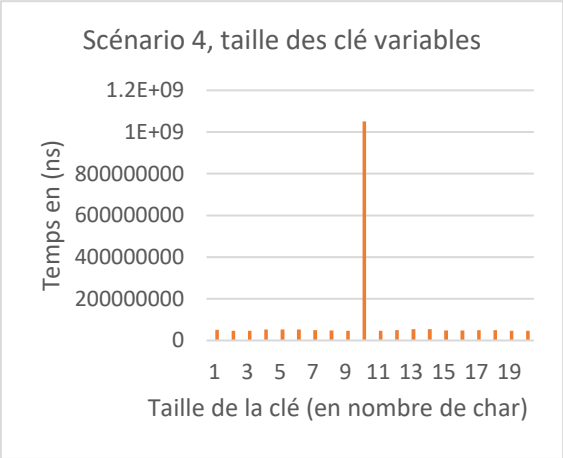
Ici aussi, nous constatons une augmentation, mais plus marquée que lors du scénario avec les puts. Cela est totalement sensé, car en faisant un get, nous attendons une réponse plus grande du serveur (une valeur) que lorsque nous faisons un put (un datagramme vide).

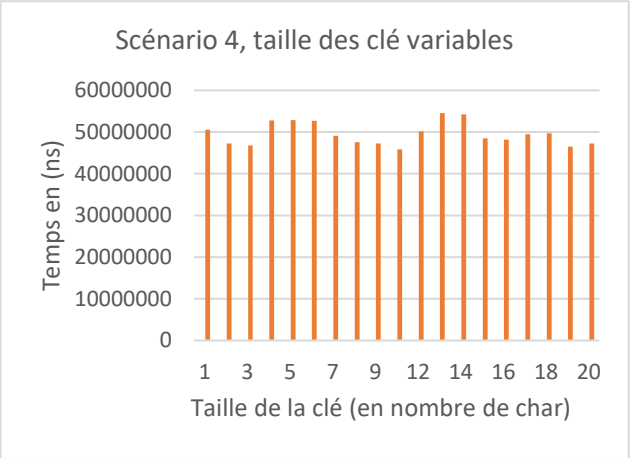
Scénario 4 : Clés de taille variable (valeurs de taille fixe), N = 20, W = R = 2

Ici, nous avons fixé N = 20, W = R = 2, et effectué 20 puts et gets de suite, pour une clé de taille variable (entre 1 et 20 caractères), et une valeur de taille fixe.

Pour éviter de reproduire les mêmes résultats peu intéressants qu'au scénario 1, nous avons ici volontairement choisi de garder W et R à 2, malgré le N à 20. En effet, nous nous sommes dit que pour bien pouvoir analyser l'impact de la variation de la taille de la clé, il serait judicieux d'avoir des opérations qui fonctionnent… ce qui fût chose faite, avec 10 succès sur 10 ! Analysons donc les temps :

Une fois encore, une valeur parasite est apparue. Voici donc les résultats avant et après l'avoir supprimée :

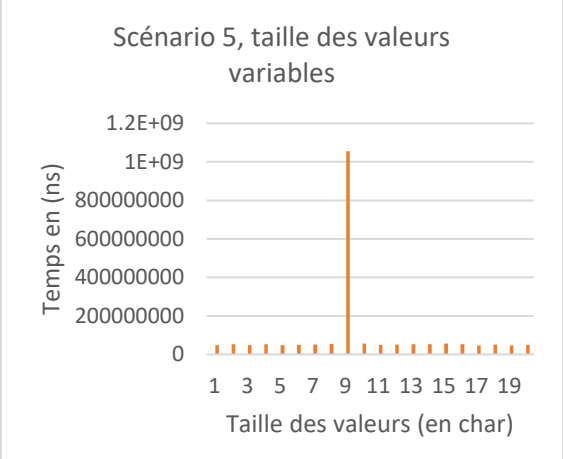


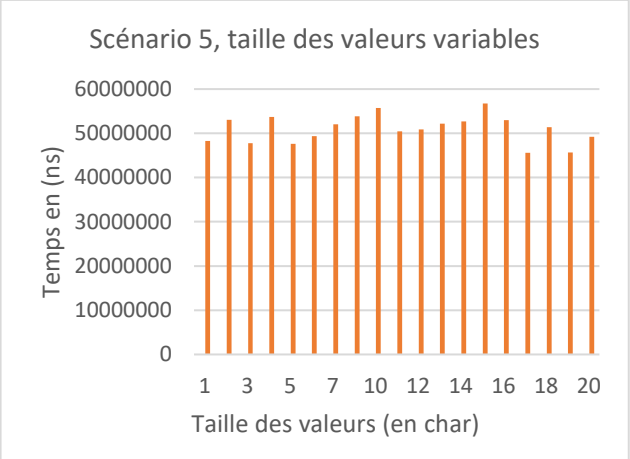


Les temps de réponse semblent très stables, ce qui peut sembler normal étant donné que le trafic induit sur le réseau ne change pas en fonction de la taille de la clé (même nombre de messages UDP).

Scénario 5 : Valeurs de taille variable (clés de taille fixe), N = 20, W = R = 2

Finalement, et toujours de manière similaire aux scénario précédent avec N = 20 et W = R = 2, nous avons cette fois-ci choisi de faire varier la taille des valeurs (également entre 1 et 20 caractères), pour une clé de taille fixe. Avec 10 succès sur 10 également, voici les résultats :





Ici aussi, des résultats cohérents avec le scénario précédent : même nombre de messages UPD → temps stable.

Conclusion

Notre première remarque, critique, pourrait être le peu de données effectivement analysables pour le scénario 1. Nous supposons que le nombre d'échecs est dû en grande partie à des timeouts, et donc une connexion trop lente. En effet, dans 90% des cas, au moins une des deux opérations (put ou get) a fonctionné. Ce problème aurait peut-être pu être réglé en augmentant le temps de timeout des sockets.

Autrement, en passant outre les petits sursauts vraisemblablement dus au VPN, les résultats semblent cohérents. Le bottleneck de notre système est en effet le réseau, et non pas le traitement des chaînes de caractères, d'où la faible influence des scénarios 4 et 5 comparé aux scénarios 2 et 3.