INF8430 – Systèmes répartis et infonuagique

Automne 2018

TP1 - Appels de méthodes à distance

[1739632] – [Vincent Leduc]

[1584742] – [François-Xavier Dueymes]

Remis à Houssem Daoud

[02-10-2018]

PARTIE 1

Question 1

Après avoir modifié le code de *ResponseTime_analyser*, le client est programmé pour faire 7 appels de chaque méthode (appel direct, appel RMI local et appel RMI distant) et d'afficher le temps écoulé à chaque retour d'appel. Les tableaux suivants présentent les résultats.

Tableau 1: Temps écoulé pour les appels normaux

Taille (bytes)	Temps écoulé (ns)
10	2481
100	287
1000	272
10000	1784
100000	377
1000000	536
10000000	3167

Tableau 2: Temps écoulé pour les appels RMI locaux

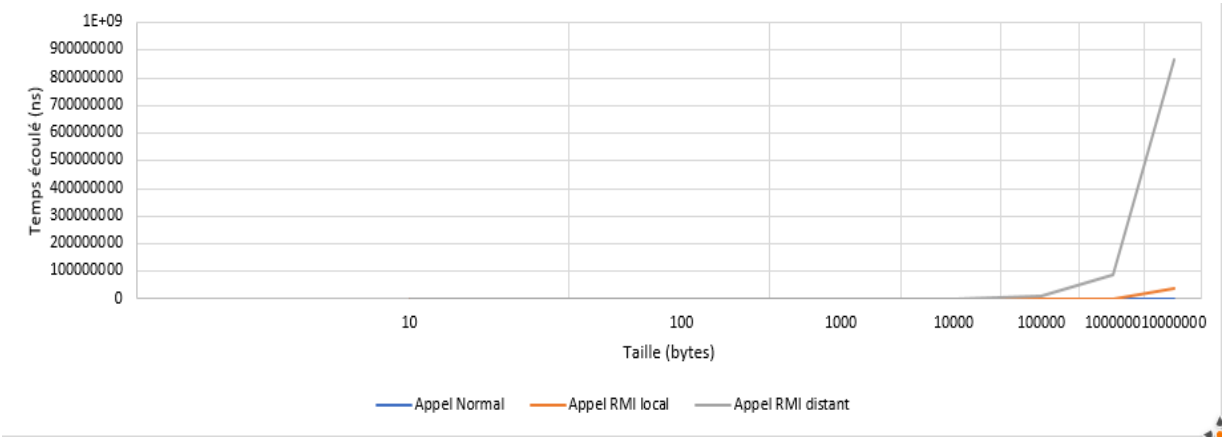
Taille (bytes)	Temps écoulé (ns)
10	1014163
100	548421
1000	520444
10000	693407
100000	679565
1000000	1792372
10000000	36896605

Tableau 3: Temps écoulé pour les appels RMI distants

Taille (bytes)	Temps écoulé (ns)
10	1930087
100	1166592
1000	1332457
10000	1992691
100000	9519631
1000000	86739925
10000000	866079776

La figure 1 est une représentation graphique des résultats à échelle logarithmique.

Figure 1: Graphique du temps écoulé de chaque appel en fonction de la taille du paramètre



Comme prévu, les appels normaux sont bien plus rapides que les appels RMI. Leurs temps d'exécution sont assez petits comparés à ceux des appels RMI pour être difficilement visualisable sur la figure 1.

Les appels RMI sont exécutés dans des temps comparables qu'ils soient effectués vers le serveur local que celui distant pour une taille d'argument de 0 à 100000 bytes. Au-delà, le temps d'appel RMI distant se multiplie pour être jusqu'à 23 fois plus élevé qu'un appel RMI local pour 10000000 bytes en argument.

Ainsi l'utilisation de la technologie RMI permet, pour un certain coût en temps supplémentaire par rapport à un appel sur un objet local, d'interagir avec un serveur sans ressentir de différence de temps de transfert pour des petits objets de 10ko ou moins. Un grand avantage est de pouvoir créer un objet sur une machine et de le rendre accessible à des machines distantes à l'aide d'une interface et d'une représentation locale de l'objet. [1] Par contre cette technique n'est pas compatible avec des objets venant d'autres langages et donc impose des contraintes de conception.

Question 2

Après avoir lancé le processus rmiregistry dans le répertoire ou se trouvent l'implémentations des interfaces, le serveur de fichiers peut être démarré. *ServerInterface* est instancié et associé à la clé "*server*" dans le registre RMI. Il peut maintenant recevoir des appels.

À son lancement, *client* tente de récupérer l'instance *ServerInterface* en en faisant la requête au registre RMI (*lookup("server"*). Client appelle alors une méthode (*execute(a, b)*) sur le proxy de *ServerInterface.* Celui-ci sérialise l'identifiant et les arguments a, b de la méthode pour que le tout soit transmis sur le réseau. Arrivé au serveur, les arguments sont désérialisés par *Remaote* et la méthode est appelée sur l'objet correspondant de *Server.* Le retour de la méthode est ensuite sérialisé et envoyé au *ServerInterface* de *Client* qui le désérialise et retourne le résultat comme un appel normal de fonction.

Références

[1] Chapitre 1 : Remote Method Invocation (RMI), G.Latu, 2003. Tiré de: http://website2k3.free.fr/cours/sd/cours_sd2_chap1.htm#_Toc55741811