

INF8480 Systèmes répartis et infonuagique

TP1 : Appel de méthodes à distance

**Gabriel Côté-Jones : 1771119**

**Aladin Riabi 1717705**

Polytechnique Montréal

11 Février 2019

Question 1

Le tableau 1 représente les temps d’exécution pour les appels Normal, local RMI et distant RMI en fonction de la taille de son paramètre et de sa valeur de retour. La taille est déterminée par 10x et est en octets.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Appel normal (ns)** | **Appel RMI local (ns)** | **Appel RMI distant (ns)** |
| **X = 1** | 2678 | 1048962 | 12801636 |
| **X = 2** | 2878 | 1274939 | 4810543 |
| **X = 3** | 2750 | 1278008 | 4131606 |
| **X = 4** | 3233 | 1340325 | 3508733 |
| **X = 5** | 2735 | 1328493 | 18666866 |
| **X = 6** | 2593 | 3593108 | 175752702 |
| **X = 7** | 4174 | 24542161 | 1754593288 |

Tableau 1 : Temps d’exécution de chaque méthode avec paramètre de taille 10 puissance x octets.

La Figure 1 est une version graphique du tableau 1. Dans ce graphe, l’axe des y représente le temps d’exécution en ns et l’axe des x représente la valeur de x.

Fig. 1 : Temps d’exécution des appels Normal, RMI local et RMI distant en fonction de x.

On voit avec le tableau 1 et la figure 1 que tous les appels ont un temps d’exécution semblable pour les valeurs de x comprises entre 1 et 4. Cependant, le temps d’exécution de l’appel RMI distant augmente de façon exponentielle à partir de x = 4.

Java RMI comporte plusieurs avantages. On voit surtout lors de la réalisation de la partie 2 du TP, que l’on peut changer l’implémentation des méthodes du serveur comme on le veut sans impact sur le client tant que la définition de la méthode ne différencie pas de celle du stub. De plus, ajouter des classes ou des méthodes au stub et au serveur n’a pas d’impact sur le client.

Cependant, RMI comporte quelques désavantages. Tous les objets qui circulent entre le serveur et le client (en paramètre ou en méthode de retour) doivent implémenter l’interface *Serializable* pour pouvoir être « mashaller ». De plus, Java RMI ne fonctionne que pour des communications Java à Java.

Question 2

1. Au lancement du serveur (objet de classe Server), le serveur instancie et envoi au registre RMI le stub (un objet de classe ServerInterface).
2. À son départ le client (objet de classe Client) utilise la méthode loadServerStub(addr) avec l’adresse IP du serveur distant en paramètre. Cette méthode permet au client de demander le stub au registre RMI.
3. Si le registre RMI, a retourné au client le stub, le client peut maintenant l’utiliser pour faire des appels aux méthodes du serveur.
4. Le client utilise la méthode execute() du stub (rappel : stub = objet de classe ServerInterface).
5. Le stub « mashalle » les informations sur les paramètres spécifiés par le client. C’est-à-dire, il envoie les informations nécessaires pour recréer les paramètres au serveur.
6. Le serveur reconstruit le paramètre, invoque la bonne méthode (dans ce cas, la méthode execute() du serveur) puis « mashalle » la réponse pour retourner l’information nécessaire au client.