École Polytechnique de Montréal

Département de Génie Informatique et Génie Logiciel

INF8480

Système répartis et infonuagique

Travail Pratique #1

Appels de méthodes à distance

Soumis par :

Hardy Vodounou
1708514

Jason Li
1746843

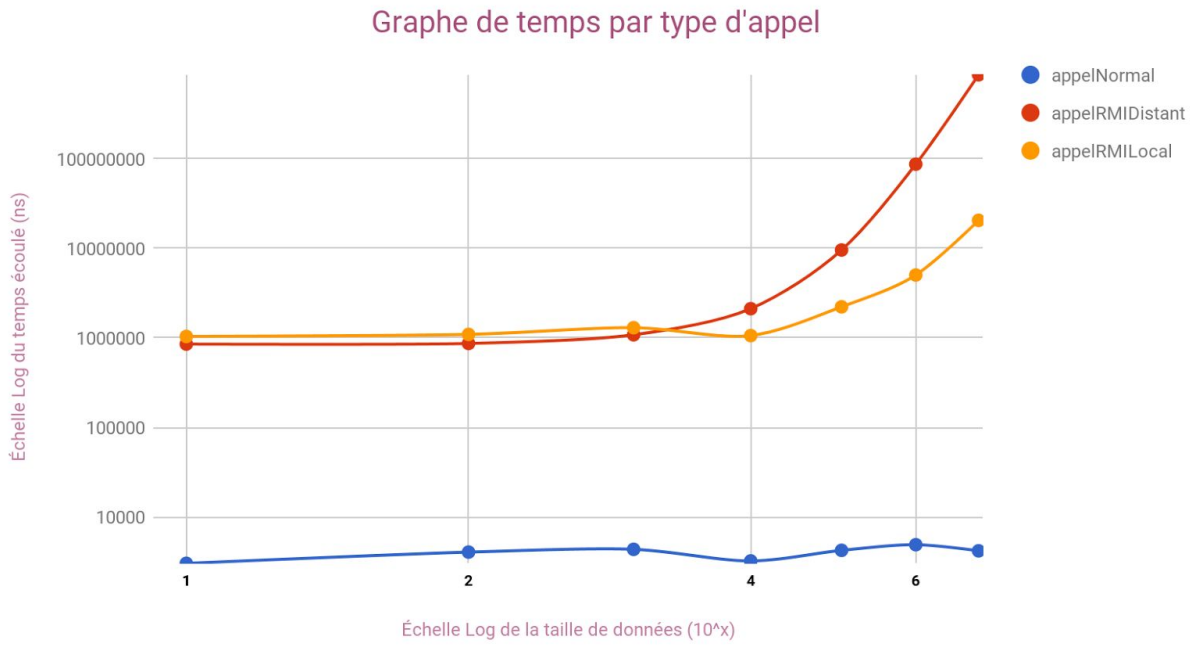
Date de remise

13 février 2018

Partie 1

Résultats de l'impact de la taille des arguments sur la performance des appels RMI.

Nous avons créé un script afin de rouler le client pour recevoir 10 échantillons par exemplaires, dont un exemplaire est le cas d'un envoi de message avec la taille de paramètre de 10^x octets. Donc, nous avons compilé la moyenne de dix échantillons de chacun des appels pour assurer une certaine précision des données. Ensuite nous avons établie le graphe ci-dessous montrant le temps consacré à chaque appel en fonction de la taille de données envoyés.



Comme que l'on peut le remarquer, un appel normal n'est pas vraiment influencé par la taille de données comparativement aux deux autres appels.

Pour les appels RMI local et distant, nous voyons une croissance prononcée voire même potentiellement polynomiale. Cela est justement dû à l'augmentation des données transférées entre le client et le serveur utilisant Java RMI, car il nécessite un processus de communication de transfert de donnée plus long que celui localement.

Avantages de Java RMI

Les méthodes et l'interface pour construire un système RPC est simple et intuitif, peu de lignes de code sont nécessaires pour l'implémenter. Une abstraction de la complexité au niveau de la connexion et des manipulations des fils d'exécution facilite la tâche à l'utilisateur ou au développeur, ce qui accélère le processus de développement du produit logiciel. La consolidation de l'application du client ou du serveur peut être indépendante entre eux. Donc, nous pouvons compiler le client sans à compiler à nouveau le côté du serveur, vice-versa, et la connection s'établit sous un niveau de complexité qui ne nous concerne plus. Il supporte aussi beaucoup de librairies qui viennent enrichir notre code.

Désavantages de Java RMI

Il y a une limite de taille de données pouvant être transportées par chaque appel de méthode. Nous avons remarqué que, lorsque la taille de l'argument passé dans la fonction dépasse 10^7 octets, cela nous donne une erreur. La technologie nous restreint seulement à un seul langage de programmation : Java. Les cas de tests demandent beaucoup de travaux, surtout lorsque les changements sont partout, car nous devons recompiler pour debugger en console ou surveiller les changements d'états et de comportements. La sécurité n'est pas le point le plus fort dans ce type de système car il n'y a aucune encryption supportée pour protéger la confidentialité des messages. Donc, une personne qui intercepte le message peut facilement déchiffrer le contenu si aucune mesure n'est faite de notre part pour y protéger.

Explication de l'interaction

Premièrement, le serveur charge et compile les fichiers de classes du code. Ensuite, le serveur charge ces instances du code dans le *rmiregistry*. Lorsque le client tente d'invoquer une méthode du serveur, le registre lui renvoie l'objet concerné, créant ainsi son «Stub». Donc, Il existe le «Stub object» au niveau du client qui rassemble les paramètres de fonctions et les conduit au «Skel Objet» au niveau serveur qui va à son tour invoquer les vraies méthodes résidant au serveur et puis retransmettre le message de retour, s'il y en a, au «Stub Objet», dont le client. Bien sûr, à chaque transfert de message, le «Stub» ou le «Skel» devra transformer ou rétablir les données reçues et envoyées. Ces manipulations de paquetages des informations dans ces cas s'appellent «marshalling» et «unmarshalling».