

# **Rapport Comparatif des Technologies de Communication Distribuée en Java**

## **Introduction**

Ce rapport présente une comparaison entre trois technologies de communication distribuée en Java : Java RMI, gRPC, et les sockets. L'objectif est de comprendre et d'explorer les fonctionnalités, les avantages, et les limitations de chaque technologie à travers le développement de services spécifiques.

## **Description des Technologies**

### **Java RMI**

Java Remote Method Invocation (RMI) permet l'exécution d'appels de méthode à distance entre les machines virtuelles Java. RMI facilite le développement d'applications distribuées en masquant la complexité de la communication réseau.

### **gRPC**

gRPC est un framework de communication interprocessus haute performance, supportant plusieurs langages de programmation. Il utilise HTTP/2 pour la communication et Protocol Buffers comme langage d'interface.

### **Sockets**

Les sockets permettent la communication entre applications à travers des réseaux en utilisant les protocoles TCP ou UDP. Ils fournissent un canal de communication de bas niveau entre les processus.

## **Implémentation**

### **Java RMI : Gestion d'une Liste de Tâches**

Le service implémenté permet d'ajouter, supprimer, et récupérer des tâches dans une liste. L'utilisation de RMI facilite la gestion d'état côté serveur et la synchronisation des appels clients.

```
ServerMain [Java Application] C:\Users\User\p2\pool  
Serveur RMI prêt.
```

```
Menu:  
A - Ajouter une tâche  
D - Supprimer une tâche  
S - Afficher la liste des tâches  
Q - Quitter  
Choisissez une option: A  
Entrez la tâche à ajouter: faire les courses  
Tâche ajoutée.  
  
Menu:  
A - Ajouter une tâche  
D - Supprimer une tâche  
S - Afficher la liste des tâches  
Q - Quitter  
Choisissez une option: A  
Entrez la tâche à ajouter: reviser java RMI  
Tâche ajoutée.  
  
Menu:  
A - Ajouter une tâche  
D - Supprimer une tâche  
S - Afficher la liste des tâches  
Q - Quitter  
Choisissez une option: D  
Entrez la tâche à supprimer: faire les courses  
Tâche supprimée.  
  
Menu:  
A - Ajouter une tâche  
D - Supprimer une tâche  
S - Afficher la liste des tâches  
Q - Quitter  
Choisissez une option: S  
Liste des tâches:  
reviser java RMI
```

## gRPC : Service de Messagerie

Le service de messagerie permet l'envoi et la récupération de messages textuels. gRPC offre une efficacité accrue grâce à l'utilisation de HTTP/2, permettant des appels simultanés et bidirectionnels.

```
[INFO] --- install:3.1.1:install (default-install) @ GrpcProject ---
[INFO] Installing C:\Users\User\Downloads\mini projet SR\GrpcProject\pom.xml to C:\Users\User\.m2\repository\messagerie\GrpcProject\pom.xml
[INFO] Installing C:\Users\User\Downloads\mini projet SR\GrpcProject\target\GrpcProject-0.0.1-SNAPSHOT.jar to C:\Users\User\.m2\repository\messagerie\GrpcProject\GrpcProject-0.0.1-SNAPSHOT.jar
[INFO] BUILD SUCCESS
[INFO] Total time: 6.038 s
[INFO] Finished at: 2024-03-29T01:55:33+01:00
[INFO]
```

```
ChatServer [Java Application] C:\Program Files\Java\jdk1.8.0_202\bin\javaw.exe (29 mars 2024, 2:09:34 AM) [pid: 2968]
Server started, listening on port: 9090
```

```
ChatClient [Java Application] C:\Program Files\Java\jdk1.8.0_202\bin\javaw.exe (29 mars 2024, 2:10:05 AM) [pid: 16756]
```

```
--- Chat started --- at port: 9090

To send a message press 'S'
To retrieve messages press 'R'
To exit press 'E':
S
Your user ID: 1
Recipient's user ID: 2
Enter your message: hello
Your message has been sent successfully. Status: Message sent successfully

To send a message press 'S'
To retrieve messages press 'R'
To exit press 'E':
R
Enter your user ID to retrieve messages: 2
Messages for user 2:
Message ID: 5c44a7fa-121e-4661-bef7-190e31d2b407
Sender ID: 1
Message: hello
```

## Sockets : Service de Chat

Le chat utilise des sockets pour permettre l'échange de messages en temps réel dans un salon commun. Cette implémentation met en évidence le contrôle granulaire offert par les sockets sur la communication réseau.

```
ChatClient (1) [Java Application] C:\Users\User\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.9.v20231011-1920\bin\javaw.exe (29 mars 2024, 2:10:05 AM) [pid: 16756]
Connected to server.
Enter your name: wiem
Received: Welcome to the chat!
Received: You are connected as wiem
Received: wiem has joined the chat.

Sending:
hello everybody!
Sending: hello everybody!

Sending:
Received: user2 has joined the chat.
Received: user2:
Received: user2: hey wiem welcome to the chat
Received: user2: ù
```

# **Comparaison des Technologies**

## **Facilité de Mise en Œuvre**

Java RMI: Simplifie le développement distribué avec des objets Java, mais est limité à l'écosystème Java.

GRPC: Nécessite la définition des services via Protocol Buffers, introduisant une étape supplémentaire, mais offre une compatibilité interlangages.

Socket: Demande une gestion manuelle de la communication réseau, augmentant la complexité du développement.

## **Performances**

Java RMI et GRPC bénéficient d'optimisations dans la gestion des connexions, gRPC ayant l'avantage avec HTTP/2.

Sockets Offrent des performances potentiellement supérieures en raison du contrôle direct sur la communication, mais au prix d'une complexité accrue.

## **Conclusion**

La sélection d'une technologie de communication distribuée en Java dépend largement du contexte d'application spécifique, des exigences en matière de performances, et de l'environnement de développement. Java RMI offre une intégration transparente dans les applications Java, gRPC excelle dans les environnements hétérogènes avec des besoins de performance élevés, tandis que les sockets sont adaptés aux situations nécessitant un contrôle complet sur la communication réseau. Ce projet a démontré que chaque technologie a ses avantages distincts et ses cas d'usage optimaux, guidant les développeurs dans le choix de l'outil adapté à leurs besoins spécifiques.