Rapport du projet

Systèmes repartis

Réalisé par :

Ghouma Mayez

Section : IDS3

Introduction

Dans ce rapport, nous allons comparer trois technologies de communication distribuée en Java : Java RMI, GRPC et les sockets. L'objectif est d'évaluer leurs performances, leur facilité de mise en œuvre, leur flexibilité ainsi que leurs avantages et limitations dans des scénarios d'utilisation réelle.

Concepts

1.Java RMI :

Gestion d'une liste de tâches

**Java RMI offre une abstraction haut niveau pour la communication entre les processus Java.*

**La mise en œuvre des services RMI est relativement simple, nécessitant principalement la définition d'une interface distante et l'implémentation de cette interface sur le serveur.*

**La transmission d'objets complexes peut être réalisée sans effort grâce à la sérialisation Java.*

**Cependant, la configuration requise pour le registre RMI peut être fastidieuse et l'interopérabilité avec d'autres langages est limitée.*

2.GRPC :

Service de messagerie

*GRPC est une technologie RPC (Remote Procedure Call) basée sur HTTP/2 et protobuf pour la sérialisation des données.

*La définition des services et des messages dans un fichier. proto est simple et claire.

*GRPC offre une polyvalence avec le support de nombreux langages de programmation.

*Les performances de gRPC sont généralement élevées grâce à l'utilisation de HTTP/2 pour le transport des données.

*Cependant, la courbe d'apprentissage peut être plus raide en raison de la complexité de certains concepts comme les flux et les canaux.

3.Sockets :

Service de chat

**Les sockets offrent un contrôle fin sur la communication réseau, permettant une personnalisation approfondie.*

**L'implémentation des services avec des sockets est flexible et peut être adaptée à divers besoins.*

**Les performances des sockets dépendent en grande partie de l'implémentation spécifique et peuvent être très bonnes dans les scénarios adaptés.*

**Cependant, la gestion manuelle de la communication réseau peut être complexe et sujette aux erreurs.*

Comparaison

****Facilité de mise en œuvre :**

-Java RMI : Simple, mais la configuration du registre peut être fastidieuse

-GRPC : Requiert une compréhension approfondie, mais offre une facilité de mise en œuvre élevée.

-Sockets : Contrôle maximal, mais gestion manuelle complexe de la communication réseau.

****Performances :**

-GRPC : Performances élevées grâce à HTTP/2 et à la sérialisation protobuf.

-Java RMI et Sockets : Performances variables selon l'implémentation et la configuration.

****Flexibilité :**

-GRPC : Grande flexibilité avec support multi-langage et types de données via protobuf.

-Java RMI : Limité au monde Java, potentielles limitations d'interopérabilité.

-Sockets : Flexibilité maximale, mais nécessite plus de travail manuel pour des fonctionnalités avancées.

Avantages et limitations

Java RMI :

Offre une intégration transparente avec Java et une simplicité d'implémentation, mais peut être limité en termes d'interopérabilité et de performances.

GRPC :

Combine performances élevées, facilité de mise en œuvre et flexibilité, mais peut avoir une courbe d'apprentissage plus raide.

Les sockets :

Offrent une personnalisation maximale, mais nécessitent plus de travail manuel et peuvent être sujets à des erreurs de bas niveau.

Conclusion

Dans ce projet, nous avons étudié trois technologies de communication distribuée en Java : Java RMI, gRPC et les sockets. Chacune a ses avantages et limitations. GRPC se distingue par ses performances élevées et sa polyvalence, Java RMI par sa simplicité d'intégration, tandis que les sockets offrent une flexibilité maximale. Le choix dépend des besoins spécifiques du projet, comme les performances et la facilité de développement.