RPC & gRPC

Remote Procedure Call

Les concepts de base du RPC

Interface Description Language (IDL)

- Définir l’interface entre un client et un serveur
- Contrat entre le client et le serveur
- Information sur la signature des appels de méthodes/procédures distantes
- Spécification des messages d’entrée et de sortie

Génération d'un "Stub"

- Code **généré automatiquement** depuis la définition de l'interface avec l'IDL.
- C'est un **proxy** côté client et côté serveur pour gérer les paramètres d'entrée et les sorties de l'appel de méthode/procédure distant
  - On appelle aussi cela du **"Marshalling"/"Unmarshalling"**
  - Mémoire distribuée et non pas une mémoire partagée, adressages différents, paramètres et sorties non connues par l'autre machine
  - Transformer les paramètres en mémoire locale en formats de données prêts à être transmis sur le réseau
  - On peut aussi parler de **sérialisation/désérialisation**
- Le stub gère aussi les messages envoyés sur le réseau et les protocoles utilisés.

Diagramme de fonctionnement

z = function_call(x,y)

Diagramme de fonctionnement

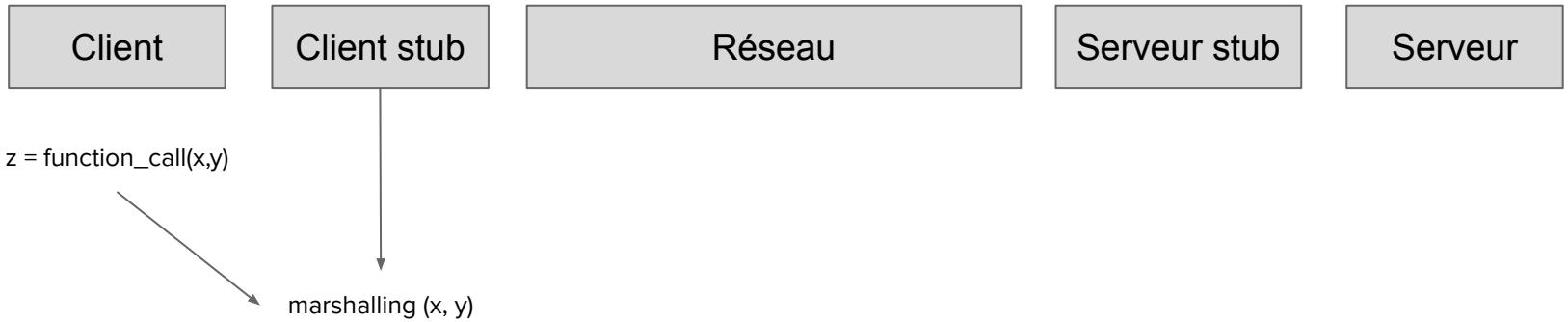


Diagramme de fonctionnement

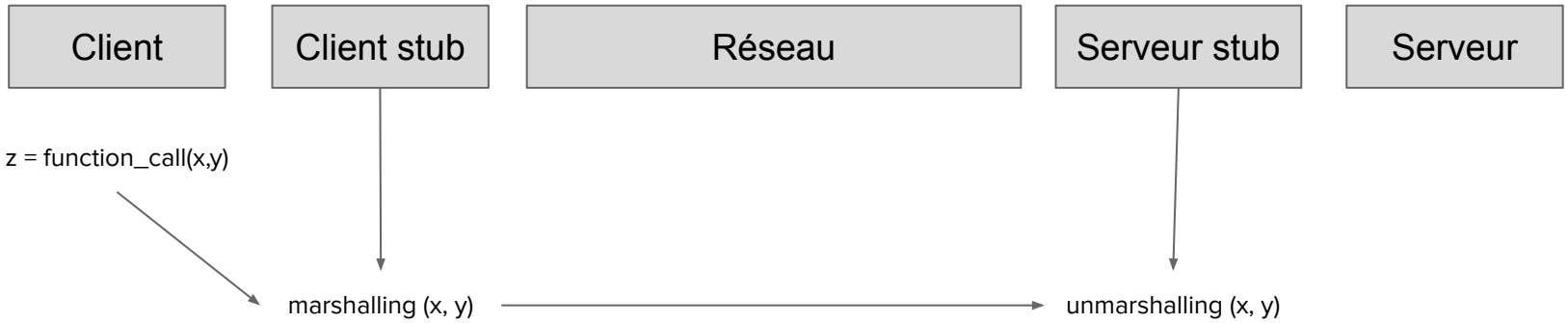


Diagramme de fonctionnement

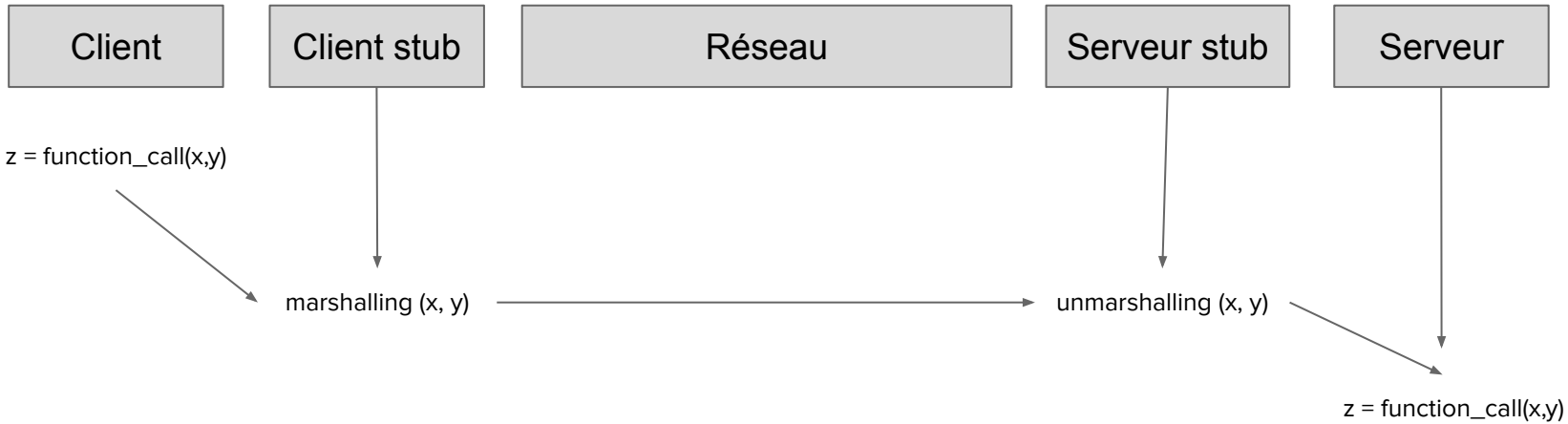


Diagramme de fonctionnement

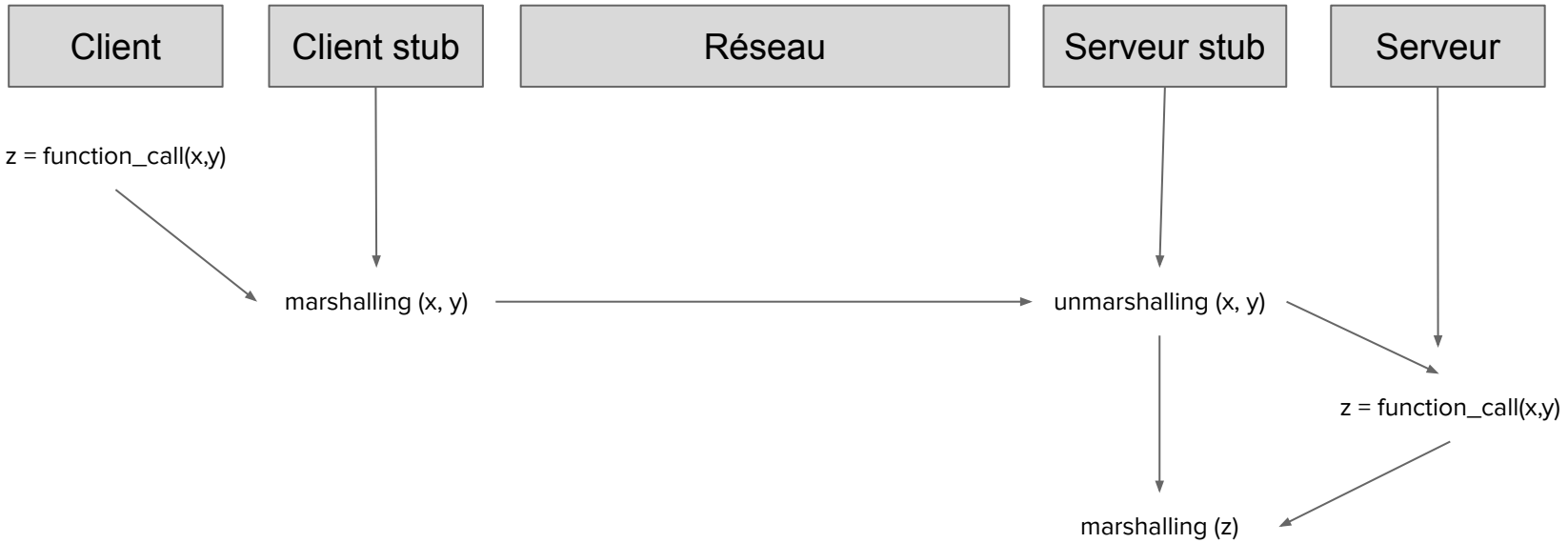
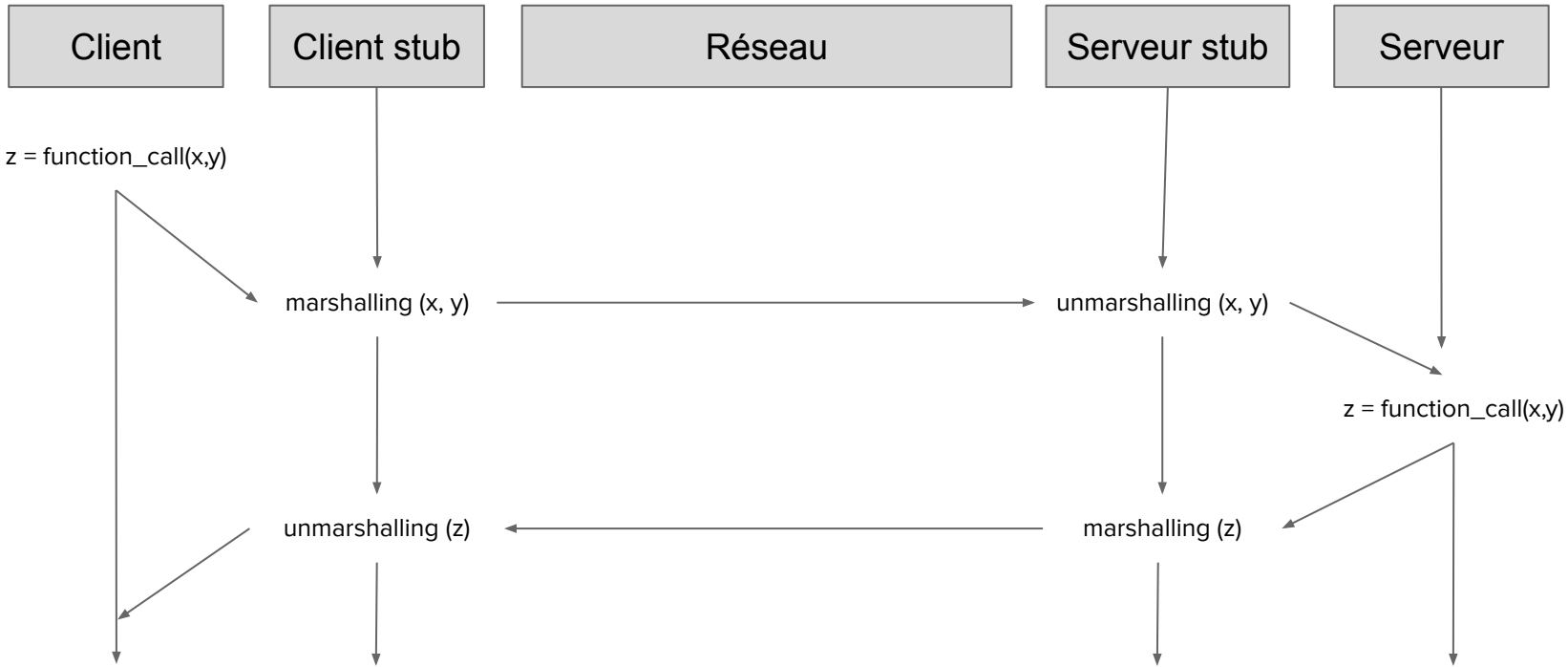


Diagramme de fonctionnement

Diagramme de fonctionnement

Diagramme de fonctionnement



RPC mis au goût du jour - gRPC

De retour dans les années 90 ?

- Framework Open Source proposé par Google
- HTTP/2 = allier la flexibilité de HTTP avec l'efficacité
- Compatible avec de nombreux langages
- Utilisé par des acteurs incontournables (NetFlix, CoreOS, CISCO etc.)
    - assure un maintien dans le temps

“Protocol buffers” l’IDL de gRPC

- Spécification de l’interface du service
- Structure des messages échangés entre le client et le serveur

```
// The greeter service definition.
service Greeter {
  // Sends a greeting
  rpc SayHello (HelloRequest) returns (HelloReply) {}
}

// The request message containing the user's name.
message HelloRequest {
  string name = 1;
}

// The response message containing the greetings
message HelloReply {
  string message = 1;
}
```

4 types de méthodes possible à invoquer

- Unary RPCs
  - Le client envoie une seule requête au serveur et obtient une unique réponse
  - Appel de fonction normal
- Server streaming RPCs
  - Le client envoie une seule requête mais reçoit une séquence de messages en réponse
- Client streaming RPCs
  - Le client envoie une séquence de messages et reçoit une unique réponse
- Bidirectional streaming RPCs
  - Le client envoie une séquence de messages et reçoit une séquence de messages en réponse

Appel synchrone ou asynchrone

- Synchrone : l’appel de méthode distante est bloquant
- Asynchrone : l’appel de méthode distante est non bloquant

Tuto

gRPC

Sujet sur Moodle !

gRPC vs REST/OpenAPI

Avantages et désavantages

	Performance API	Debug API	Découverte API
REST / OpenAPI			
gRPC			

	gRPC	**REST/OpenAPI**
Flexibilité chgt d'API	modifier fichiers protos re-générer code modifier les codes	modifier les requêtes modifier les codes
Dépendances	installation gRPC pour tous les langages souhaités	installation de bibliothèques de type Flask / FastAPI
Travail	explosion nb de fonctions	explosion nb points d'entrée

Exemple

Un exemple d’application fournie par Google qui utilise gRPC pour les communications entre les micro-services internes, avec plusieurs langages à la fois.

https://github.com/GoogleCloudPlatform/microservices-demo

TP

gRPC

Sujet sur Moodle !