



République Tunisienne

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique



المعهد العالي للدراسات التكنولوجية برادس
Institut Supérieur Des Etudes Technologiques De Rades

Rapport Sur le Framework GRPC



M2-MPDAM - ISET Rades

Internet of Things

Réalisé par : Mabaouj Aya

Email : mabaoujaya@gmail.com

Année Universitaire : 2023/2024

Sommaire

INTRODUCTION GRPC.....	5
1. Histoire gRPC :.....	5
2. Définition grpc :	5
CONCEPTS DE BASE DE GRPC	6
ARCHITECTURE GRPC	26
AVANTAGES & INCONVENIENTS GRPC.....	35
1. Avantages GRPC :.....	35
2. Inconvénients GRPC :.....	36
CAS D'UTILISATIONS GRPC	37
CONCLUSION	50

Liste des figures

Figure 1 - logo gRPC.....	5
Figure 2-Protocol Buffers	6
Figure 3-Streaming.....	7
Figure 4-Http/2.....	7
Figure 5-Canaux.....	8
Figure 6-Architecture	26

Liste des tableaux

Tableau 1-Avantages GRPC 35

Tableau 2-Inconvénients GRPC..... 36

Introduction GRPC

1. Histoire gRPC :

En 2015, Google a développé gRPC comme extension du framework RPC pour relier de nombreux microservices créés avec différentes technologies. Initialement, il était étroitement associé à l'infrastructure interne de Google, mais plus tard, il a été rendu open source et standardisé pour une utilisation communautaire. Au cours de la première année de sa sortie, les plus grandes organisations l'ont exploité pour alimenter des cas d'utilisation allant des microservices au Web, au mobile et à l'IoT. Et en 2017, il est devenu le projet d'incubation de la Cloud Native Computing Foundation (CNCF) en raison de sa popularité croissante.

2. Définition grpc :

gRPC est un framework RPC (Remote Procedure Call) open source robuste utilisé pour créer des API évolutives et rapides. Il permet aux applications client et serveur de communiquer de manière transparente et de développer des systèmes connectés. De nombreuses grandes entreprises technologiques ont adopté gRPC, telles que Google, Netflix, Square, IBM, Cisco et Dropbox. Ce framework s'appuie sur HTTP/2, des tampons de protocole et d'autres piles technologiques modernes pour garantir une sécurité, des performances et une évolutivité maximales des API.



Figure 1 - logo gRPC

Concepts De Base De GRPC

L'évolution et la réussite de gRPC s'expliquent par l'adoption de technologies de pointe offrant des performances supérieures à celles de JSON et XML, tout en renforçant la sécurité des API. Les principaux concepts de Grpc sont les suivants :

- **Tampons de Protocole (Protobuf) :** Les tampons de protocole, ou Protobuf, sont un protocole de sérialisation/désérialisation développé par Google. Ils permettent de définir des services et de générer automatiquement des bibliothèques clientes. gRPC utilise les fichiers .proto pour définir des services et des messages, et le compilateur Protobuf (protoc) génère le code client et serveur en fonction de ces fichiers. Cette approche offre des avantages en termes de performances, de légèreté des messages et de facilité d'utilisation.

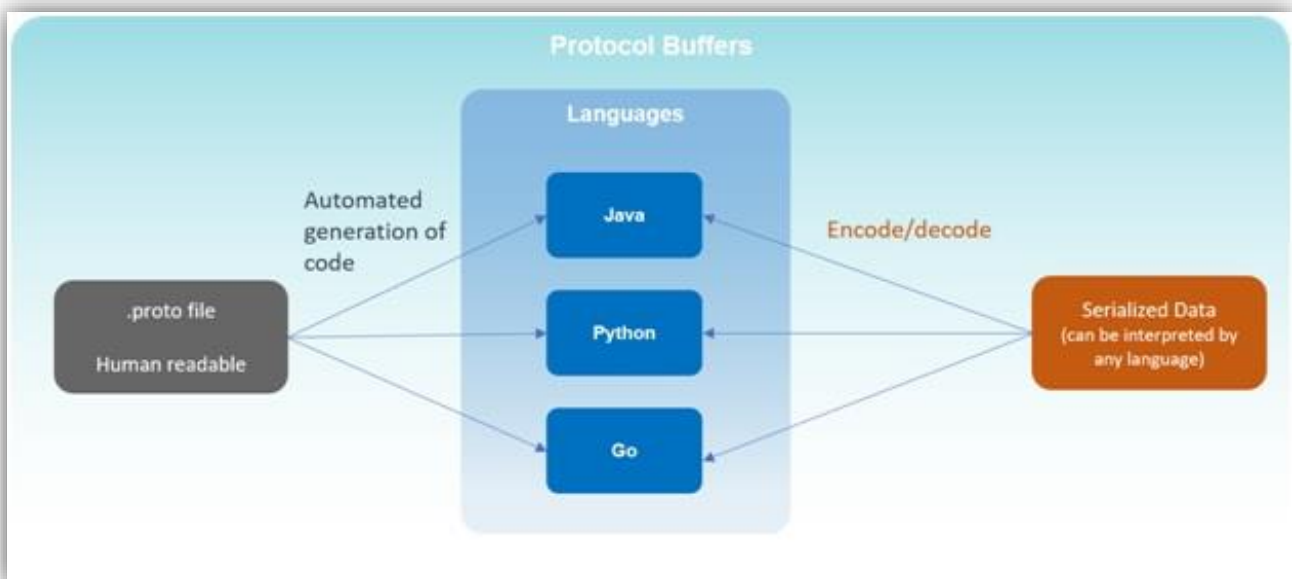


Figure 2-Protocol Buffers

- **Streaming :** Le streaming est un concept clé de gRPC, permettant plusieurs processus au sein d'une seule requête. Cela est possible grâce à la capacité de multiplexage de HTTP/2, où plusieurs réponses peuvent être envoyées ou plusieurs requêtes reçues simultanément via une seule connexion TCP.

Il existe trois types de streaming : RPC de streaming côté serveur, RPC de streaming côté client, et RPC de streaming bidirectionnel, chacun permettant des flux de données dans des directions spécifiques.

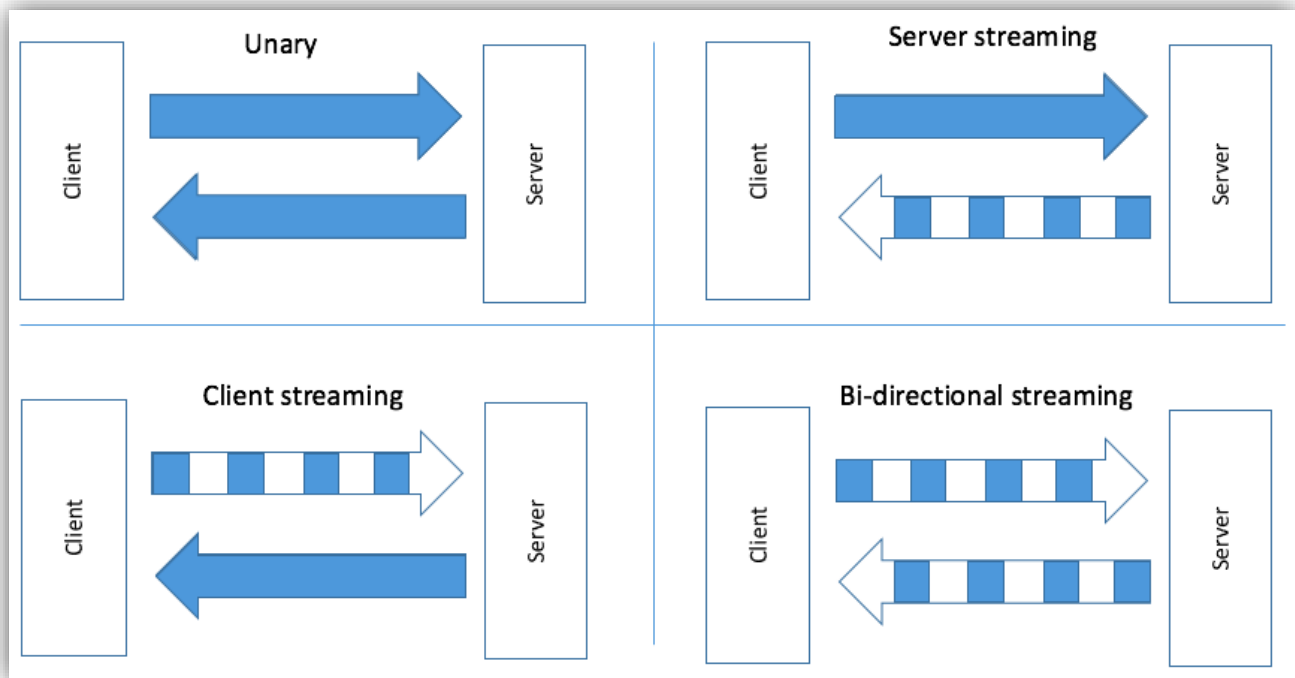


Figure 3-Streaming

- **HTTP/2 :** gRPC est construit sur le protocole HTTP/2, qui apporte de nombreuses améliorations par rapport à HTTP/1.1. Il utilise un encodage binaire, permet le streaming bidirectionnel en duplex intégral, offre un contrôle de flux et une compression d'en-tête. HTTP/2 améliore les performances globales et réduit la consommation de ressources. C'est le fondement sur lequel gRPC offre une communication efficace entre les clients et les serveurs.

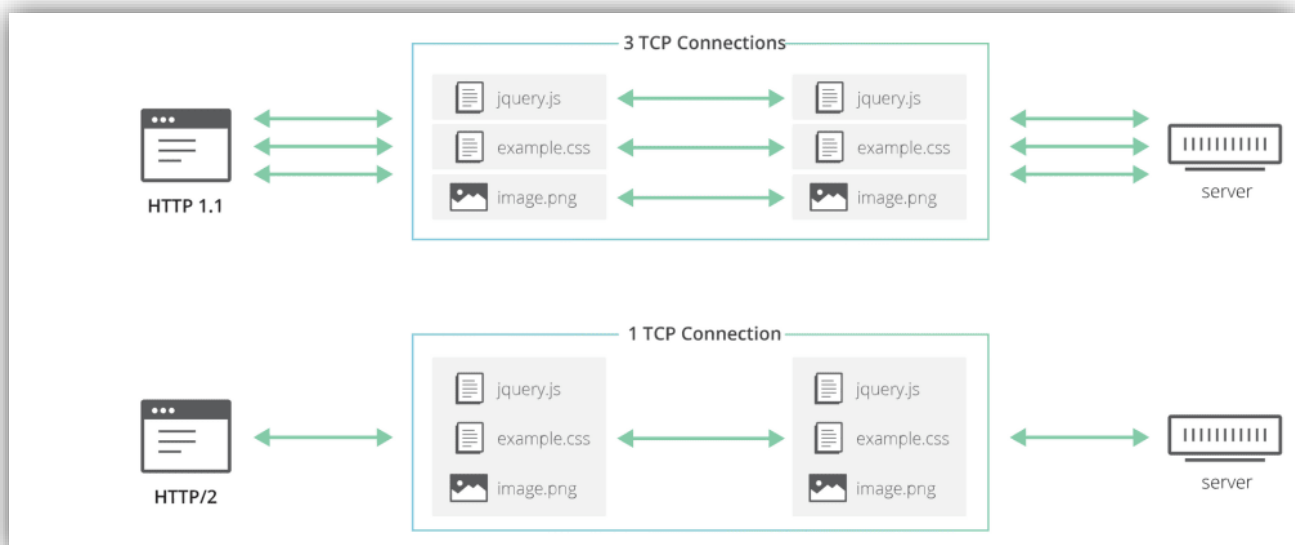


Figure 4-Http/2

- **Canaux :** Les canaux sont essentiels dans gRPC. Ils étendent le concept de flux HTTP/2 en prenant en charge plusieurs flux sur plusieurs connexions simultanées. Les canaux permettent de se connecter au serveur gRPC sur une adresse et un port spécifiques, facilitant la création de stubs clients pour l'interaction avec le serveur.

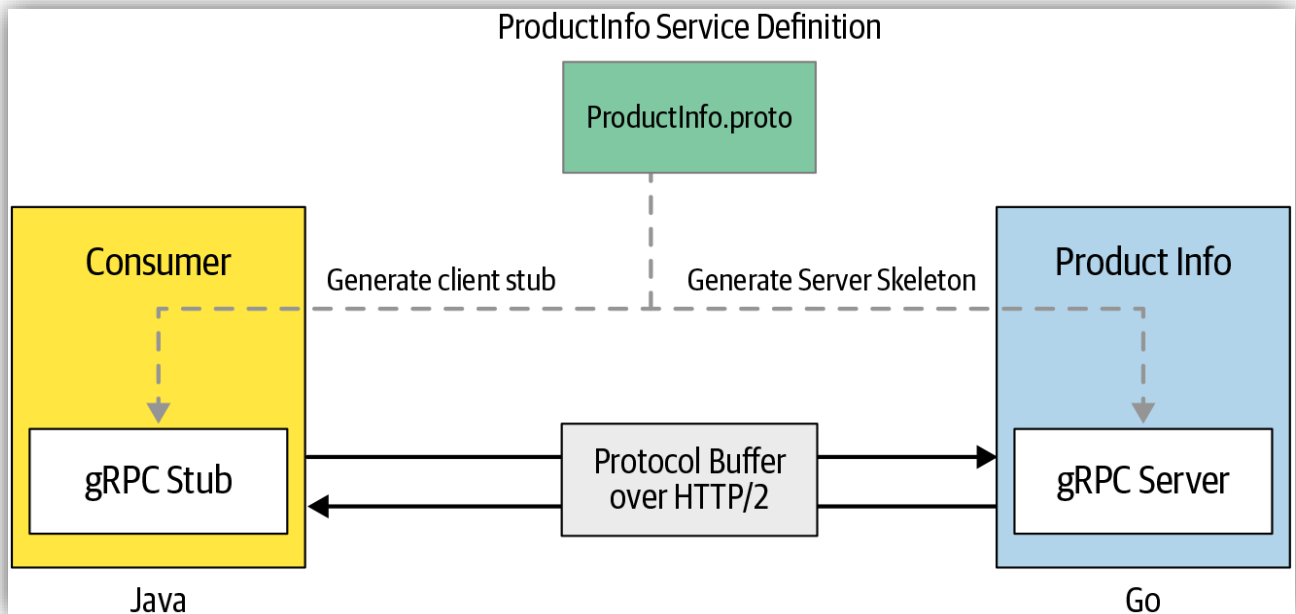


Figure 5-Canaux

Architecture gRPC

Dans le schéma d'architecture gRPC ci-dessous, on distingue les aspects client et serveur de gRPC. Au sein de gRPC, chaque service client est doté d'un "stub" (généré automatiquement), qui agit comme une interface regroupant les opérations distantes disponibles. Le client gRPC initie un appel de procédure locale vers le stub, en transmettant les paramètres destinés au serveur. Le stub client prend ensuite ces paramètres, les sérialise via le processus de marshaling à l'aide de Protobuf, et transmet la requête à la bibliothèque cliente locale, s'exécutant sur la machine du client.

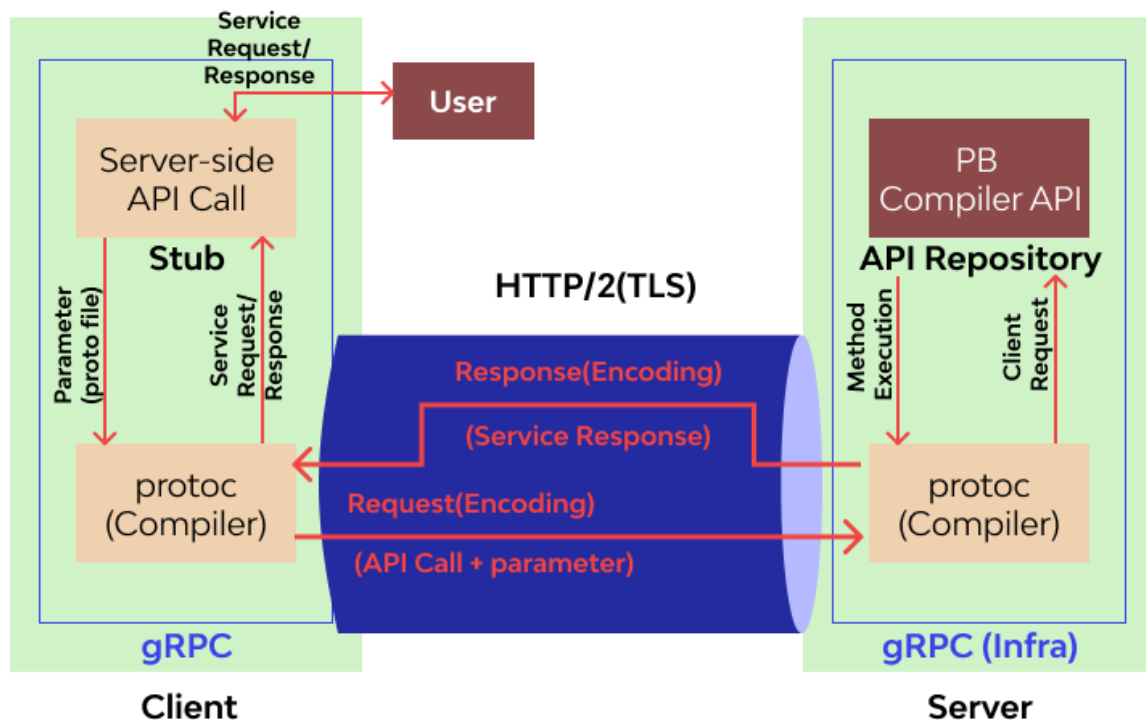


Figure 6-Architecture

Le système d'exploitation du client contacte le serveur distant en utilisant le protocole HTTP/2. Le système d'exploitation du serveur reçoit ces paquets et invoque la procédure correspondante dans le stub du serveur. Ce dernier décode les paramètres reçus et effectue l'appel de procédure approprié à l'aide de Protobuf. Le stub du serveur renvoie ensuite la réponse, préalablement encodée, à la couche de transport du client. Le stub client récupère le message de résultat, décompresse les paramètres renvoyés, et le contrôle est renvoyé à l'appelant.

Avantages & Inconvénients

GRPC

1. Avantages GRPC :



Caractéristique	Description
Performance	Utilisation de Protobuf et HTTP/2 pour des performances supérieures, avec des messages plus petits et un chargement plus rapide.
Streaming	Sémantique de streaming côté client/serveur, permettant divers types de streaming, y compris bidirectionnel.
Génération de code	Génération automatique de code natif accélérant le développement en produisant des squelettes côté serveur et des stubs côté client.
Interopérabilité	Fonctionne sur plusieurs plates-formes et langages grâce au format binaire Protobuf, offrant une prise en charge multiplateforme complète.
Sécurité	Utilisation de TLS de bout en bout sur HTTP/2 pour garantir la sécurité de l'API, encourage l'authentification et le chiffrement des données.
Convivialité et productivité	Solution RPC transparente multi-langages, réduisant le temps de développement grâce à la génération automatique de code.
Caractéristiques intégrées	Offre une prise en charge intégrée de nombreuses fonctionnalités de base, simplifiant le développement d'applications robustes.

Tableau 1-Avantages GRPC








2. Inconvénients gRPC :



Limitations	Descriptions
Prise en charge limitée du navigateur	gRPC ne peut pas être directement appelé depuis un navigateur Web en raison de sa forte dépendance à HTTP/2. Des couches de proxy et gRPC-web sont nécessaires pour convertir entre HTTP/1.1 et HTTP/2, ce qui ajoute de la complexité
Format non lisible par l'homme	Les messages gRPC sont compressés dans un format non lisible par l'homme grâce à Protobuf. Les développeurs ont besoin d'outils supplémentaires pour analyser les charges utiles, écrire des requêtes manuelles et effectuer le débogage
Pas de mise en cache périphérique	Les appels gRPC utilisent la méthode POST, ce qui empêche la mise en cache périphérique. Cela peut poser des problèmes de sécurité des API, et la spécification gRPC ne prévoit pas de mécanisme de cache entre serveur et client
Apprentissage plus difficile	gRPC peut être difficile à apprendre pour certaines équipes, en particulier lorsqu'il s'agit de se familiariser avec Protobuf et de gérer les aspects spécifiques d'HTTP/2. Cette courbe d'apprentissage plus raide peut inciter certains à préférer REST pour plus longtemps

Tableau 2-Inconvénients gRPC

Cas d'utilisations GRPC

-  **Communication entre microservices :** C'est l'un des cas d'utilisation les plus répandus. Les microservices sont des petits services autonomes qui communiquent les uns avec les autres pour fournir des fonctionnalités plus larges. gRPC facilite la communication entre ces services grâce à ses performances élevées et sa prise en charge de nombreux langages de programmation.
-  **Applications web modernes :** gRPC est adapté pour la communication entre le frontend et le backend des applications web modernes. Il prend en charge la communication bidirectionnelle, ce qui est utile pour les fonctionnalités en temps réel.
-  **IoT (Internet des objets) :** Les appareils IoT, tels que les capteurs et les dispositifs embarqués, peuvent utiliser gRPC pour communiquer avec un serveur central de manière efficace, en tirant parti du protocole HTTP/2 pour minimiser la surcharge réseau.
-  **Services de streaming :** gRPC prend en charge les services de streaming, ce qui signifie que vous pouvez établir des flux de données bidirectionnels entre les clients et les serveurs. Cela convient aux applications de diffusion en continu, aux jeux multijoueurs, aux tableaux de bord en temps réel, etc.
-  **Développement multi-langages :** Vous pouvez utiliser gRPC pour développer des applications dans différents langages de programmation tout en maintenant la communication entre elles.
-  **Contrôle d'accès et sécurité :** gRPC offre des fonctionnalités de sécurité avancées, notamment l'authentification, l'autorisation et le chiffrement de bout en bout, ce qui est essentiel pour protéger les données et les communications sensibles.
-  **Traitement asynchrone :** gRPC permet de traiter les appels de manière asynchrone, améliorant ainsi les performances des applications à haute charge.

Conclusion

En conclusion, gRPC est un framework RPC polyvalent et puissant qui offre une communication rapide, efficace et évolutive entre les applications client et serveur. En s'appuyant sur des technologies telles que les tampons de protocole, HTTP/2 et le streaming, gRPC répond aux besoins de développement d'API modernes, avec une adoption croissante par de grandes entreprises et une variété de cas d'utilisation, allant des microservices à l'IoT. Son potentiel pour améliorer les performances et la sécurité des applications en fait une solution de choix dans le paysage technologique actuel.

