# RAPPORT

## Comparaison des clients Synchrones


*Réalisé par*

*Imaghri Aya et Rouibah Salma*

# 1 Configuration de la machine de test

## 1.1 Configuration matérielle 'Hardware'

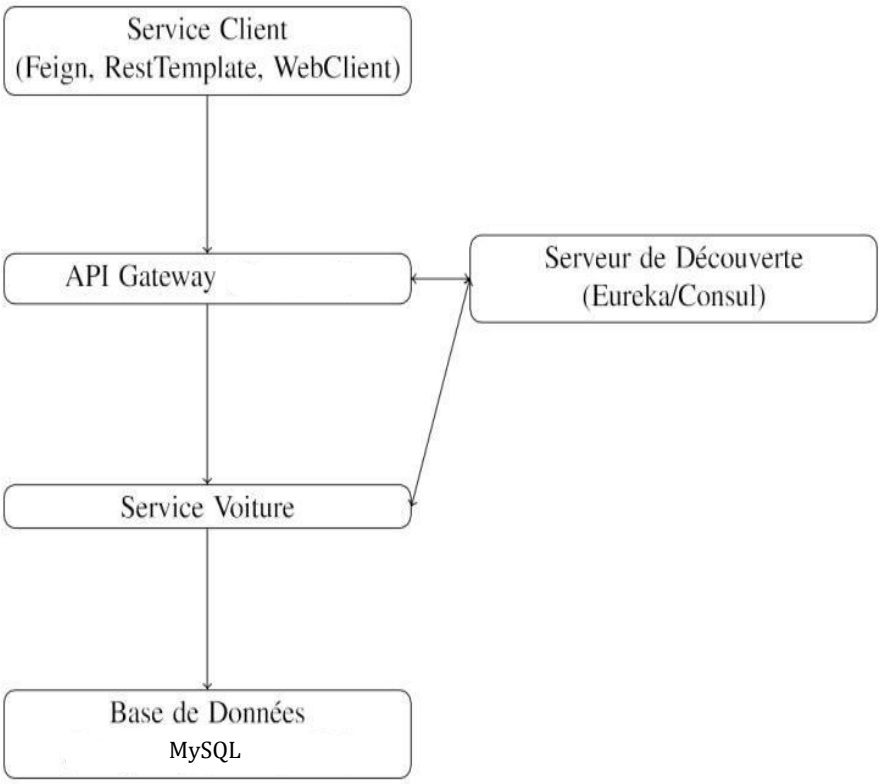
| Composant | Spécifications |
|---|---|
| Processeur CPU | *Intel(R) Core (TM) i7-1250U 1.10 GHz* |
| RAM | 16,0 Go |
| Carte Réseau (NIC) Ethernet | Intel(R) Wi-Fi 6E AX211 160MHz |
| Carte Graphique (GPU) | Intel(R) Iris(R) Xe Graphics |
| Système d'exploitation | Système d'exploitation 64 bits, processeur x64 |

*Tableau 1 : Spécifications matérielles de la machine de test*

## 1.2 Configuration Logicielle

| Composant | Version |
|---|---|
| Outil de Test de charge | Apache JMeter 5.6.3 |
| Simulateur de Panne | Permutus 3.2 |

## 2.1 Architecture globale des microservices

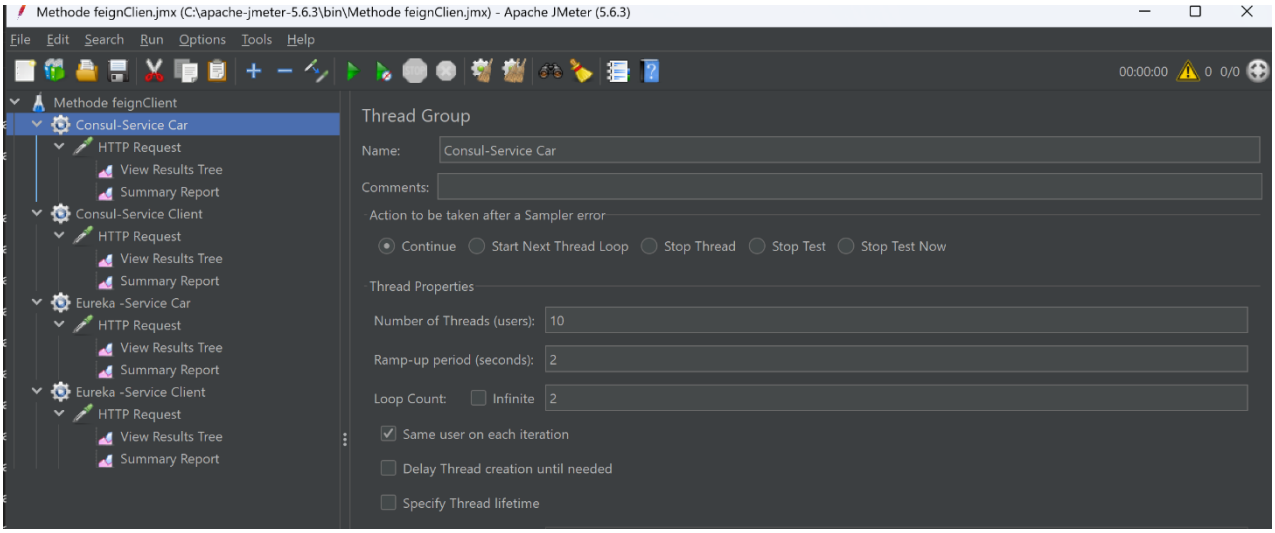


## 2 Scenario de test et outils utilise

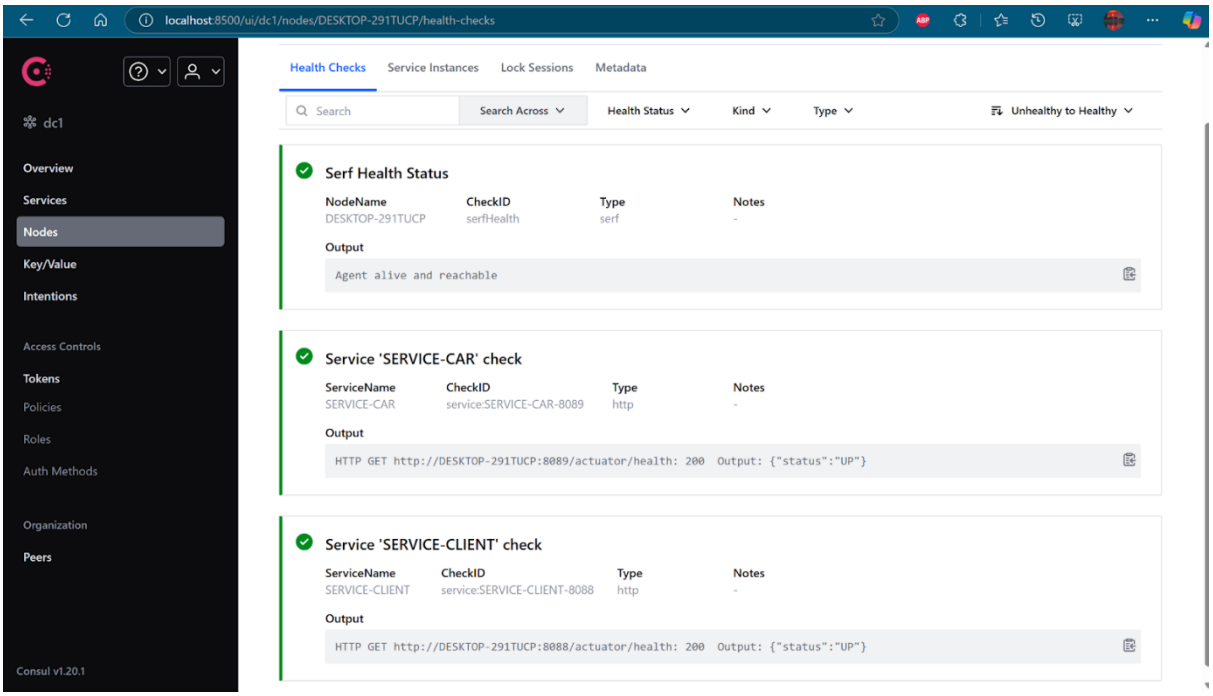
Des requêtes sont envoyées depuis le service client vers le service voiture. Les tests de performance sont réalisés avec :

**Jmeter :** Simulation de 10 à 1000 requêtes simultanées pour observer le comportement du système sous différentes charges.

### Plan de test pour méthode feignClient




## Interface ui Consul



# 3 Résultats des Tests de Performance

## 3.1 Performance (Temps de Réponse et Débit)

| Méthode | Eureka | | Consul | |
|---|---|---|---|---|
| | Temps (ms) | & Débit (req/s) | Temps (ms) | & Débit (req/s) |
| **RestTemplate** | 9 | 0.25 | 13 | 0.55 |
| **Feign Client** | 5 | 0.57 | 8 | 0.95 |
| **WebClient** | 7 | 3.03 | 11 | 0.48 |

*Tableau 1 : Comparaison des performances (Temps moyen et débit)*

## 3.2 Consommation des Ressources (CPU et Mémoire)

| Méthode | Eureka | | Consul | |
|---|---|---|---|---|
| | CPU (%) | & Memoire (MB) | CPU (%) | & Memoire (MB) |
| **RestTemplate** | 4 | 65 | 7 | 83 |
| **Feign Client** | 6 | 69 | 8 | 85 |
| **WebClient** | 11 | 75 | 6 | 91 |

*Tableau 2 :* Consommation *de ressources*

## 3.3 Résilience et Tolérance aux Pannes (Charge Élevée)

| Scénario d e Panne | 10 req/s | 100 req/s | 500 req/s | 1000 req/s |
|---|---|---|---|---|
| Panne du service client | Reconnexion rapide | Reconnexion rapide | Délai 6s | Délai 11s |
| Panne du service voiture | Réponse Lente ou échec | 11% échoué | 35% échoué | 60% échoué |
| Panne du serveur de découverte (eureka) | Cache actif | Cache actif | 17% échoué | 26% échoué |
| Panne du serveur de découverte (consul) | Cache actif | Cache actif | Cache actif | 14% échoué |
| Panne de réseau partielle | Reprise rapide | Reprise rapide | Délai 6s | Délai 20s |

*Tableau 3 : Comportement du système sous charge élevée lors de pannes*

### 3.4 Simplicité d'implémentation :

| Critéres | RestTemplate | FeignClient | WebClient |
|---|---|---|---|
| Configuration Initial | Facile à configurer | Configuration plus complexe | Configuration avancée |
| Lignes de codes | 10 lignes de code | 13 lignes de code | 16 lignes de code |
| Complexite | Faible (synchronisé) | Moyenne (abstraction basée sur interface) | Haute (asynchrone et réactif) |

*Tableau 4 : Simplicité d'implémentation*

## 4 Discussion des Résultats des Tests de Performance

Les tests ont comparé les performances, la consommation des ressources et la résilience des méthodes de communication avec Eureka et Consul. Voici une analyse détaillée des résultats.

### 4.1 Temps de Réponse et Débit

Apres une analyse on remarque un écart significatif entre les méthodes en matière de temps de réponse et de débit :

- **RestTemplate avec Eureka** : Temps de réponse de 9 ms et débit de 0,25 req/s, performant pour des charges légères mais limité en débit.
- **Feign Client avec Eureka** : Temps de réponse amélioré à 5 ms et débit de 0,57 req/s, montrant une meilleure gestion des requêtes avec une latence réduite.
- **WebClient** : Temps de réponse de 7 ms et débit élevé pour Eureka (3,03 req/s), mais des performances inférieures avec Consul (0,48 req/s), probablement en raison de sa nature asynchrone exigeant plus de ressources.

Conclusion :

Les tests de débit révèlent que WebClient offre les meilleures performances avec Eureka, mais Feign Client se démarque avec Consul en affichant un débit supérieur. Cela suggère que Feign Client est mieux adapté pour gérer des charges importantes sous Consul, tout en maintenant une efficacité équilibrée en termes de latence et de consommation de ressources.

## 4.2 Consommation des Ressources

Les tests de **consommation des ressources** (CPU et mémoire) révèlent des écarts significatifs entre les méthodes :

- **RestTemplate** : Avec une consommation minimale de 4 % de CPU et 65 MB de mémoire sous Eureka, il s'impose comme l'option la plus légère. Cela en fait un choix idéal pour les applications simples ou à faible charge.
- **WebClient** : Plus gourmand, il consomme 11 % de CPU et 75-91 MB de mémoire, en raison de sa nature réactive nécessitant davantage de ressources pour gérer les connexions asynchrones.
- **Feign Client** : Affichant une consommation modérée de 6-8 % de CPU et 69-85 MB de mémoire selon le service de découverte utilisé, il offre un bon équilibre entre performances et ressources.

Ces observations montrent que **RestTemplate** convient mieux aux environnements à ressources limitées, tandis que **Feign Client** et **WebClient** privilégient les performances, au prix d'une consommation accrue.

## 4.3 Résilience et Tolérance aux Pannes

À faible charge (10-100 req/s), les services affichent une reconnexion rapide ou utilisent efficacement le cache pour limiter les impacts. Cependant, à des charges plus élevées (500-1000 req/s), des délais importants ou des taux d'échec élevés apparaissent, notamment pour les pannes du service voiture (60 % d'échecs à 1000 req/s) et du serveur de découverte Eureka (26 % d'échecs à 1000 req/s). En revanche, Consul se distingue par une meilleure gestion grâce à son cache actif, même sous forte charge, avec seulement 14 % d'échecs. Cela met en évidence la nécessité de choix stratégiques en fonction de la tolérance aux pannes et des charges prévues.

## 4.4 Simplicité d'implémentation

Les tests de simplicité d'implémentation ont révélé des différences importantes entre les méthodes :

- **RestTemplate** : La méthode la plus simple à configurer, nécessitant seulement 10 lignes de code pour une configuration de base. Grâce à son approche synchrone, elle est idéale pour les applications simples ou les environnements peu complexes, avec une configuration minimale.
- **WebClient** : La méthode la plus complexe, demandant jusqu'à 16 lignes de code et une bonne maîtrise des paradigmes réactifs. Son approche asynchrone permet de gérer efficacement les flux non bloquants et les requêtes simultanées. Cependant, cette flexibilité accrue s'accompagne d'une courbe d'apprentissage plus élevée en raison des concepts de programmation réactive.
- **Feign Client** : Bien que légèrement plus complexe (environ 13 lignes de code), il s'intègre parfaitement avec Spring Cloud. Idéal pour les environnements de microservices, il simplifie les appels aux services tout en rendant le code des applications distribuées plus lisible. Malgré une configuration un peu plus longue, il est bien adapté aux projets nécessitant une gestion efficace des appels HTTP.