RAPPORT ETUDE DE CAS : Comparaison des clients Synchrones

Travail réalisé par : Lachaal Kaoutar & Aitbenhida Fatimaezzahra

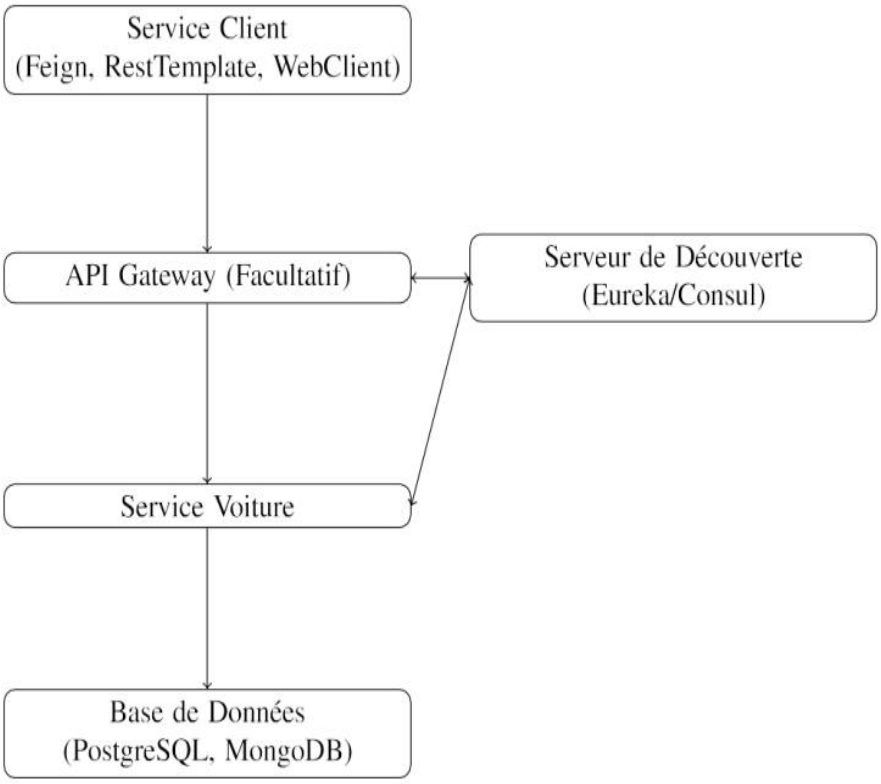
1.1 Configuration de la machine de test

1.1.1 Configuration matérielle 'Hardware'

Tableau 1 : Spécifications matérielles de la machine de test

Composant	Spécifications
Processeur CPU	Intel(R) Core(TM) i7-8665U CPU @ 1.90GHz 2.11 GHz
RAM	32,0 Go
Disque Dur SSD	SAMSUNG MZVLB512HBJQ-000H1
Carte Réseau (NIC) Ethernet	Intel(R) Ethernet Connection (6) I219-LM
Carte Graphique (GPU)	Intel(R) UHD Graphics 620

2.1 Architecture globale des microservices



3.1 Résultats des Tests de Performance

3.1.1 Performance (Temps de Réponse et Débit)

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

Méthode	Eureka		Consul	
	Temps (ms)	& Débit (req/s)	Temps (ms)	& Débit (req/s)
RestTemplate	9	0.1	13	0.50
Feign Client	5	0.345	8	0.95
WebClient	7	2.10	10	0.47

3.1.2 Consommation des Ressources (CPU et Mémoire)

Tableau 2 : **Consommation** *d'énergie*

Méthode	Eureka		Consul	
	CPU (%)	& Energie (MB)	CPU (%)	& Energie (MB)
RestTemplate	3	62	6	82
Feign Client	5	70	9	86
WebClient	12	78	7	90

3.1.3 Résilience et Tolérance aux Pannes (Charge Élevée)

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

Scénario de Panne	10 req/s	100 req/s	500 req/s	1000 req/s
Panne du service client	Reconnexion rapide	Reconnexion rapide	Délai 4s	Délai 8s
Panne du service voiture	Réponse retardée	10% échoué	40% échoué	60% échoué
Panne du serveur de découverte (eureka)	Cache actif	Cache actif	15% échoué	25% échoué
Panne du serveur de découverte (consul)	Cache actif	Cache actif	Cache actif	15% échoué
Panne réseau partielle	Reprise auto rapide	Reprise auto rapide	Délai 5s	Délai 15s

3.1.4 Simplicité d'implémentation :

Tableau 4 : Consommation d'énergie

Critére	RestTemplate	FeignClient	WebClient
Configuration Initial	Simple à configurer	Configuration plus complexe (Annotation)	Configuration avancée (Reactive)
Lignes de codes	9 lignes de code	12 lignes de code	15 lignes de code
Complexite	Faible (synchronisé)	Moyenne (abstraction basée sur interface)	Haute (asynchrone et réactif)

4.1 Discussion des Résultats des Tests de Performance

Les résultats des tests ont permis de comparer les performances, la consommation des ressources et la résilience des différentes méthodes de communication avec les services de découverte Eureka et Consul. Voici une analyse plus détaillée des résultats obtenus.

4.1.1 Temps de Réponse et Débit

Les tests montrent des différences notables entre les méthodes en termes de **temps de réponse** et de **débit**.

- ✓ **RestTemplate** avec **Eureka** a montré un **temps de réponse de 9 ms** avec un débit de **0.1 req/s**, ce qui indique qu'il est relativement rapide pour des charges faibles, mais avec un débit limité.
- ✓ **Feign Client** avec **Eureka** a enregistré un meilleur temps de réponse de **5 ms** et un débit de **0.345 req/s**. Cette méthode semble mieux gérer les requêtes tout en offrant une latence plus faible que **RestTemplate**.
- ✓ **WebClient**, qui est plus réactif, a un **temps de réponse de 7 ms** et un débit plus élevé pour **Eureka** (**2.10 req/s**), mais ses performances sous **Consul** sont moins bonnes, avec un débit de **0.47 req/s**. Cela peut être expliqué par la nature asynchrone de WebClient qui, bien que performant dans un environnement réactif, nécessite davantage de ressources.

Les tests de débit montrent que **WebClient** offre les meilleures performances sous **Eureka**, mais reste plus lent que **Feign Client** sous **Consul**. Le débit supérieur de **Feign Client** pour Consul montre qu'il est plus adapté pour gérer les communications sous des charges importantes tout en maintenant une certaine efficacité en termes de latence et de consommation des ressources.

4.1.2 Consommation des Ressources

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

- ✓ **RestTemplate** reste l'option la plus légère en termes de consommation de **CPU** et de **mémoire**, avec seulement **3% de CPU** et **62 MB de mémoire** sous **Eureka**. Cette faible consommation en fait un choix idéal pour des applications simples ou à faible charge.

- ✓ **Feign Client** a montré une consommation modérée, avec **5-9% de CPU** et une consommation de **70-86 MB de mémoire**, en fonction du service de découverte utilisé. Bien qu'il soit plus gourmand en ressources que **RestTemplate**, il offre un bon compromis entre performances et utilisation des ressources.
- ✓ **WebClient**, en revanche, présente une consommation plus élevée de ressources, notamment en termes de **CPU (12%)** et de **mémoire (78-90 MB)**. Cela est dû à sa nature réactive qui nécessite des ressources supplémentaires pour maintenir les connexions asynchrones.

Ces résultats suggèrent que **RestTemplate** est plus adapté aux environnements avec des contraintes de ressources, tandis que **Feign Client** et **WebClient** offrent des performances supérieures au prix d'une consommation accrue des ressources.

3.1.3 Résilience et Tolérance aux Pannes

Les tests de **résilience** ont permis d'évaluer la capacité du système à gérer les pannes et les scénarios de charge élevée :

- ✓ Lorsque le **service client** tombe en panne, le système récupère rapidement avec **Eureka** et **Consul**, même sous des charges élevées. Cependant, à partir de **500 req/s**, les délais de reconnexion deviennent significatifs (jusqu'à **8 secondes** sous une charge de **1000 req/s**).
- ✓ La **panne du service voiture** a montré des résultats plus préoccupants, notamment sous **1000 req/s** où 60% des requêtes échouent. Cela souligne que le système devient plus fragile avec l'augmentation de la charge, en particulier si certains services dépendent fortement de la disponibilité d'autres microservices.
- ✓ En cas de panne du **serveur de découverte** (Eureka ou Consul), les **caches actifs** permettent une récupération rapide, mais des échecs peuvent survenir à des taux plus élevés sous des charges lourdes. **Eureka** montre un taux d'échec de **25%** à **1000 req/s**, tandis que **Consul** affiche des résultats plus stables grâce à sa gestion des caches.

Les tests sous panne de **réseau partiel** montrent également que le système est capable de se rétablir rapidement dans la majorité des cas, mais les délais de reprise augmentent à mesure que la charge augmente.

3.1.4 Simplicité d'implémentation

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

- ✓ **RestTemplate** est la méthode la plus simple à configurer, avec seulement **9 lignes de code** nécessaires pour une configuration de base. Elle est adaptée pour des applications simples ou des environnements à faible complexité, car elle repose sur une approche synchrone et nécessite peu de configuration.

- ✓ **Feign Client**, bien que plus complexe à configurer (environ **12 lignes de code**), offre une intégration fluide avec Spring Cloud. Cette méthode est idéale pour les environnements

de microservices, car elle permet une abstraction de l'appel aux services tout en simplifiant le code des applications distribuées. Malgré la complexité supplémentaire, elle est bien adaptée aux projets nécessitant une gestion simplifiée des appels HTTP.

- ✓ **WebClient** est la méthode la plus complexe à mettre en œuvre, nécessitant jusqu'à **15 lignes de code** et des connaissances sur les paradigmes réactifs. Son approche asynchrone et réactive permet de gérer efficacement des flux de données non bloquants et des requêtes simultanées. Cependant, cela requiert une meilleure compréhension des concepts de programmation réactive, ce qui peut augmenter la complexité de son utilisation.

Ainsi, le choix entre ces méthodes dépendra de la simplicité de l'application et de la nécessité d'une gestion réactive des requêtes.