Serverless - Faas

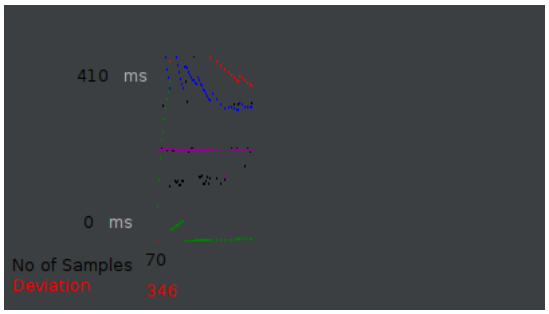
Blancy Antoine, Pertuzati Gustavo, Dagier Thomas

D1:

AWS:

Cold et Warm start respectivement:

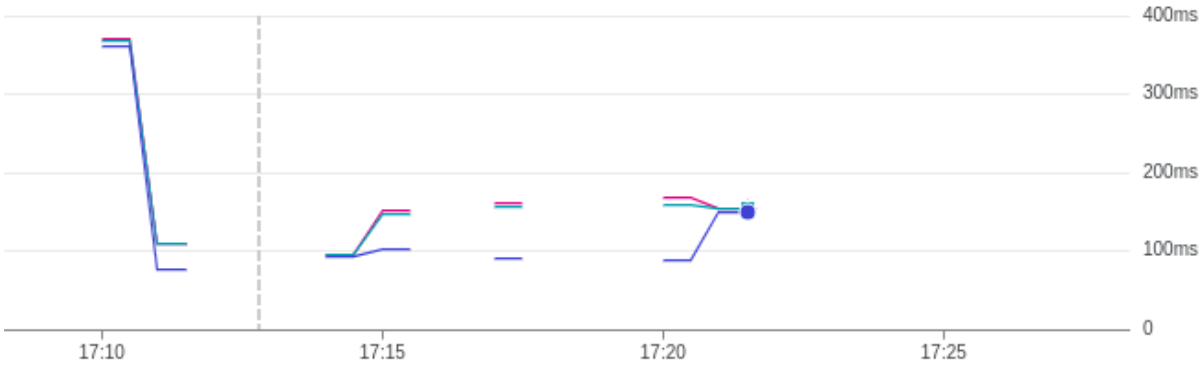
On voit que les temps d'exécution ne sont pas si différents que ça pour AWS entre cold et warm start, mais le warm start est légèrement plus rapide, sur plusieurs tests différents. On peut aussi voir le temps de réponse qui est plus élevé sur un cold start dans JMeter (Au début on a le cold start, et après le warm start. Le bleu est le temps de réponse moyen). Ça doit être dû au chargement des dépendances et de l'environnement d'exécution



Un total d'une 80aine d'invocations ont été faites pour ces tests.
AWS n'a pas de free tier, donc non, les invocations effectuées ne sont pas comprises dans le free tier.

Google:

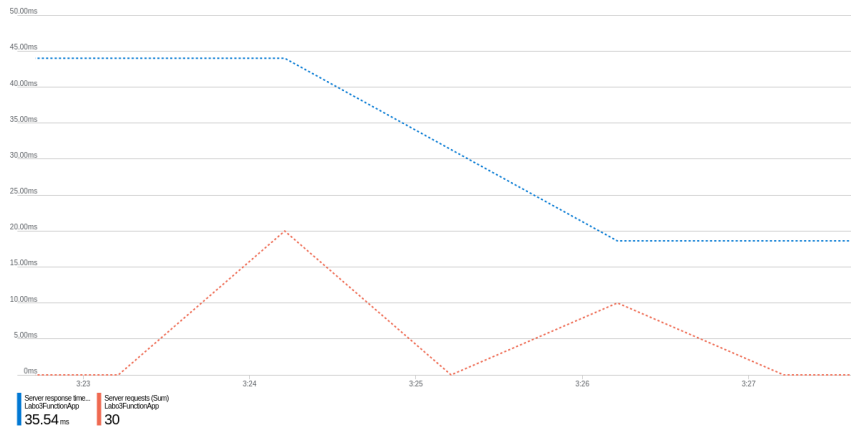
Cold/Warm Start Metrics View:



Cold/Warm Start JMeter Graph View:

On peut voir que les temps ne sont pas tout à fait similaires mais qu'il n'y a pas de grandes différences entre le cold et le warm start. Un total de 20 invocations ont été effectuées pour ces tests.

Azure:



Cold / warm start (Metric view)

Cold / warm start (JMeter view)

On observe que les temps d'exécutions ne sont pas très différents entre le cold et le warm start. Le cold start est logiquement plus lent que le warm start car il y a tout à charger (dépendances, fonctions...) alors que l'opération est déjà faite au moment où on mesure le warm start. En outre, l'environnement d'exécution peut aussi avoir changé.
On peut aussi voir le temps de réponse qui est plus élevé sur un cold start dans JMeter (Au début on a le cold start, et après le warm start. Le bleu est le temps de réponse moyen).

Pour ces tests, nous avons réalisé une centaine d'invocations. Azure possède un option permettant de ne pas payer avant d'avoir fait un minimum de 1'000'000 d'exécutions (puis 20 centimes par millions d'exécution). De ce fait, les invocations sont comprises dans le service free-tier.

D2:

	Concurrency	Asynchronous Calls	Programming Flexibility	Cold/Warm Start
AWS	Quand on test avec 100k utilisateurs, le délai monte à 10s, et pour 10k - 5s. Donc ce n'est pas très performant.	AWS permet d'avoir des invocations synchrones ET asynchrones	-Supporte 7 langages de prog. -Possible d'importer runtime si langage souhaité pas disponible. -Création de couches et de triggers	Pas de grande différence entre les deux quant-au temps d'exécution (~2ms). Le délai est un peu plus long au cold start sur JMeter mais pas de beaucoup (~70ms)
Google	Quand on test avec 400 clients qui envoient trois requêtes par seconde, il est possible de paramétrer la simultanéité. On peut donc avoir besoin de beaucoup moins d'instances pour gérer un même volume de requêtes.	Il est possible de mettre des tâches en file d'attente de façon sécurisée afin qu'elle soit traitée de manière synchrone ET asynchrone.	-Infrastructure personnalisable -Options de performances -Compatibilité des fonctionnalités natives -Machines virtuelles gérées	Les démarrages à froid peuvent prendre beaucoup de temps. Il est recommandé d'éviter les démarrages à froid inutiles et de rationaliser le processus de démarrage à froid dans la mesure du possible.
Azure	Quand on test avec 100k utilisateurs, le délai monte à 16s et pour 10k - 12s. Donc ce n'est pas très solide et performant. Par contre, ce n'est pas limité ce qui peut être très pratique.	Azure met à disposition la possibilité d'avoir des invocations asynchrones et synchrones. D'après la doc l'asynchrone est tout de même fastidieux	-interface peu pratique -on est presque contraint d'utiliser vscode pour le déploiement (ca reste facile à utiliser) -les invocations sont comprises dans le free-tier -disponibles dans 7 languages -possibilité d'avoir des custom runtime - le timeout maximum est de 30min (le plus élevé des providers)	Pas de grande différence entre les deux quant-au temps d'exécution (~10ms). Le délai est un peu plus long au cold start sur JMeter mais pas de beaucoup (~20ms)