Grenoble INP ENSIMAG

Serverless

Serverless c'est quoi ?

Quand vous n'avez pas à définir et gérer des serveurs pour fournir des services
  ils sont toujours présents quand même!

Exemple : le RPC
  local ou distant, pas moyen de savoir

Des degrés de serverless-ness
  BD Oracle: serverful (3 instances)
  RDS Aurora, DynamoDB : serveur transparent
  DynamoDB on-Demand mode : server less

EC2 instance : serverfull
Docker orchestré par Kubernetes, presque server less
Lambda fonction à exécuter : serverless (FaaS function as a service)

Intérêts du serverless

économie : vous ne payez que pour les vrais traitements que vous effectuez

serveur web : vous payez même s'il ne répond pas

exemples : 90% d'économie pour financial engines

sécurité : la sécurisation (surveillance, mise à jour, etc.) est faite par le fournisseur

élasticité : pas besoin de dimensionner l'architecture, fait par le fournisseur

qualité du code (plus facile ?)

pas suffisamment de recul pour se prononcer

être sans état (fonctionnel) est clé

microservices connectés par des bus de message asynchrones

une certitude : cela permet de consacrer tous ses moyens à l'écriture et la maintenance du code

Limites du serverless

Pas adapté à toutes les applications
dès qu'il y a un état global le serverless peut être problématique

Serverless adapté à des opérations courtes (ou pouvant s'exprimer sous forme de combinaison d'opérations courtes) -> microservices

état global possible s'il est accédé peu souvent
opérations d'écriture exceptionnelles

Souvent applications structurées en deux éléments
1/ data plane : données persistantes avec logique métiers propres => serverfull
2/ control plane : accès à ces données, traitement et exceptionnellement écriture => serverless
Le data plane est dimensionné statiquement, le control plane dynamiquement en fonction des sollicitations

Latence du serverless

Critique récurrente : combien de temps prend l'invocation d'une méthode en serverless ?

Routage des invocations : bus de message haut débit

Fonction de la date de dernière invocation
- récente : latence courte
- problème du « cold start »
- Java a un coût élevé en cold start, optimisé pour runtime

Chargement du contexte
- sticky sessions (conservation du contexte entre 2 invocations)
  - avec routage intelligent
  - problème pour fournisseur, pas pour le client
    - mais le problème de latence impactera le client!

Pas de garantie de latence absolue donnée par le fournisseur
- Souvent pour des pourcentages d'invocation

Étendue du spectre serverlesss

Définition pour AWS (Lambda)
- exécution à la demande
- gratuit si pas utilisé
- totalement géré par le fournisseur de cloud
- limité à 5 minutes par invocation

Function as a Service (FaaS)
- partie où le code est spécifique à la logique business
- peut être implémenté avec des containers
- pas forcément dans un cloud public

Pas forcément event-driven
- les flux de données traditionnels peuvent s'appliquer

passage du serverfull au serverless

Service full + calcul éphémère
- l'infrastructure de calcul n'existe pas tant qu'elle ne calcule pas
  - latence du déploiement du calcul est critique
- le service en charge de la persistence ne peut pas être serverless
- exemple : transcodage de vidéo

Lien étroit entre ressources utilisées et ressources facturées
- exemple : AWS batch qui déploie des containers sur un cluster EC2
  - pour un {ensemble de} traitement donné

Control plane plus restreint et plus abstrait
- on passe de l'expression de ressources à une qualité de service voulue

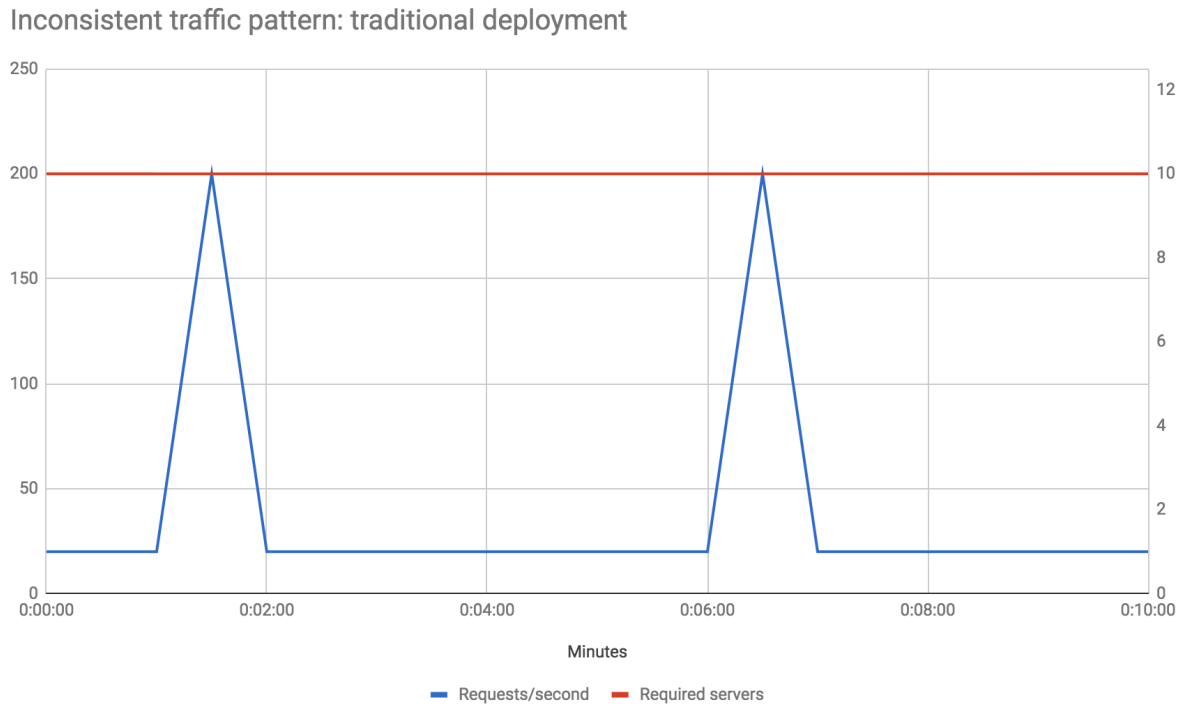
Risque de vendor lock-in

Exemple d'un serveur web

A priori pas serverless du tout

En fait
- site web stocké sur S3
- servir une page est une invocation de fonction

Informations de routage à ajouter (et coût associé)



Le futur du server less

Amélioration de la gestion des états
- garder des invocations de fonctions actifs plus longtemps
- diminuer la latence d'accès aux données nécessaires (cache)
  - proxy avant les serveurs FaaS ?

Réduction de la latence
- payer pour avoir toujours des fonctions toujours disponibles ?
- (Durable Functions chez Microsoft)

Comment répartir ses logiciels entre Serverless et serveurs persistents ?

Références

https://martinfowler.com/articles/serverless.html

Le code d'un serveur de mail en server less :
https://github.com/0x4447/0x4447_product_s3_email

https://vlfig.me/posts/microservices (Microservices — architecture nihilism in minimalism's clothes)

et le TP à venir…

Plateforme OpenFaaS

Volonté de standardiser le serverless

Basé sur cluster Kubernetes

Invocation de fonction = démarrage d'un pod
légèreté incompatible avec portabilité ?