

A cosmic background image featuring a deep blue space filled with numerous stars of varying brightness. Two prominent, elongated galaxies with pinkish-purple dust lanes and blue star-forming regions are visible, one above and one below the central text.

NETFLIX COSMOS

Guillaume VACHERIAS - Eric HERQUÉ

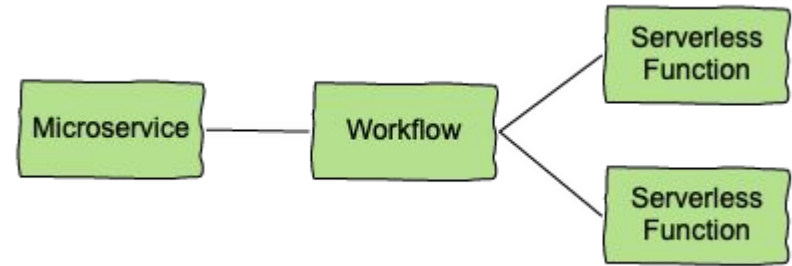
Introduction

Plateforme qui regroupe des microservices composés de :

- Workflow
- Fonction serverless

Plusieurs générations :

1. Architecture Monolithique qui permet les fonctions de streaming (2007)
2. Amélioration de la 1er génération
3. Reloaded (2017)



Architecture Monolithique

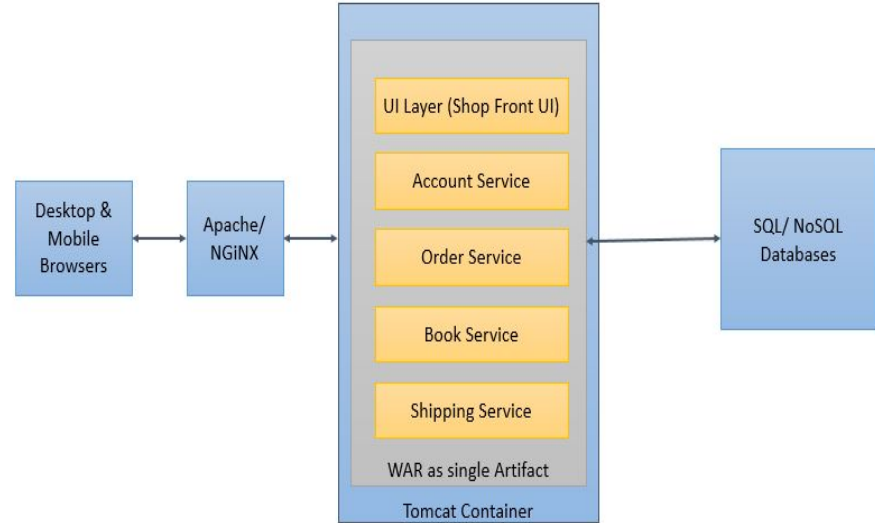
- Netflix Reloaded est basé sur un style d'architecture Monolithic



- Rapide à développer
- Simple à déployer (Java WAR File Format)
- Simple à mettre à grande échelle (scale up)



- Code source très lourd
- Temps de démarrage du container très long
- Problème de scaling development
- Difficile à changer d'outil de développement



Architecture Microservice

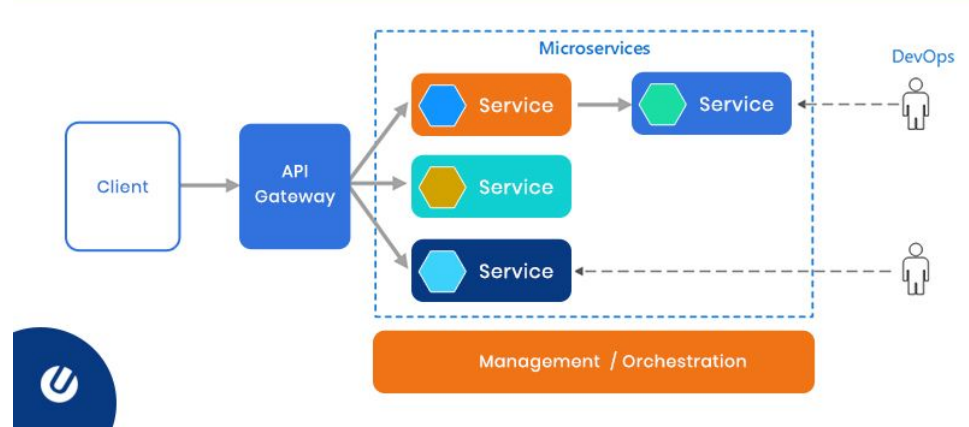
- Cosmos se base sur une architecture similaire à celui d'un microservice



- Permet le déploiement en continue
 - Facilite les tests
 - Facilite le déploiement
 - Facilite la maintenance
- Isolation de fautes
- Possibilité de changement d'outil de développement

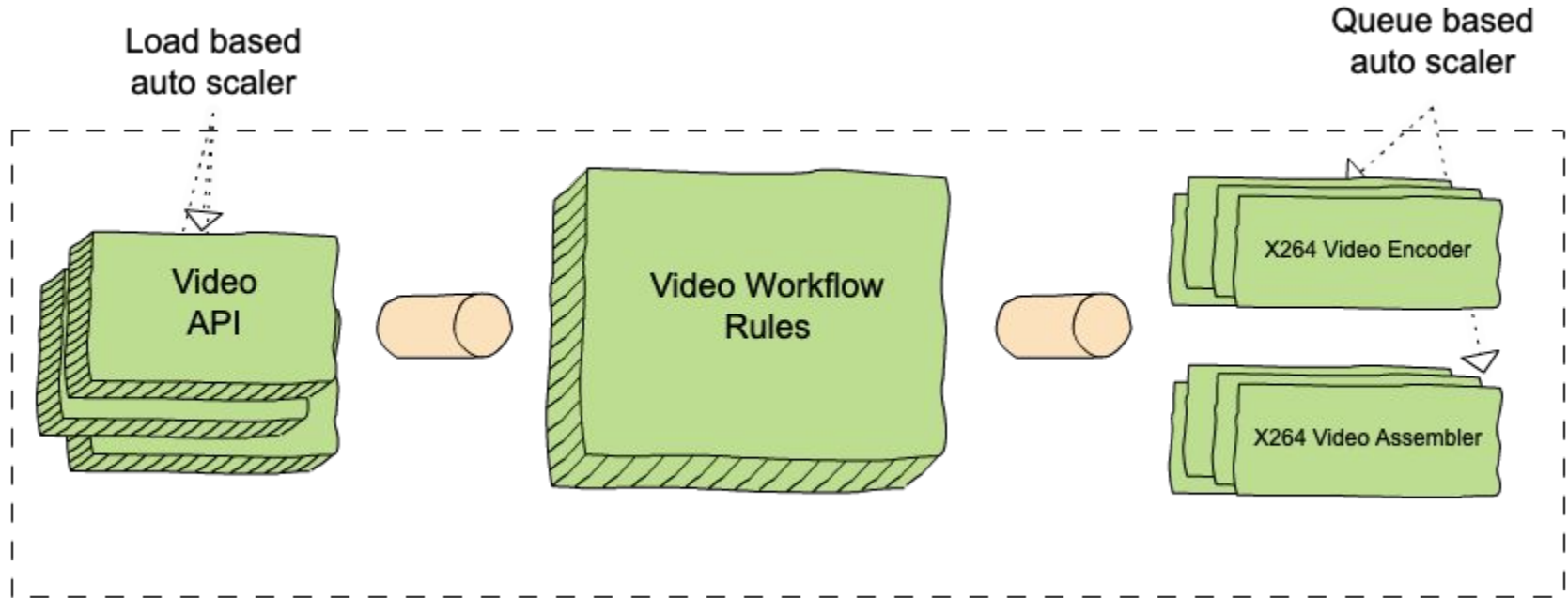


- Complexité des systèmes distribués



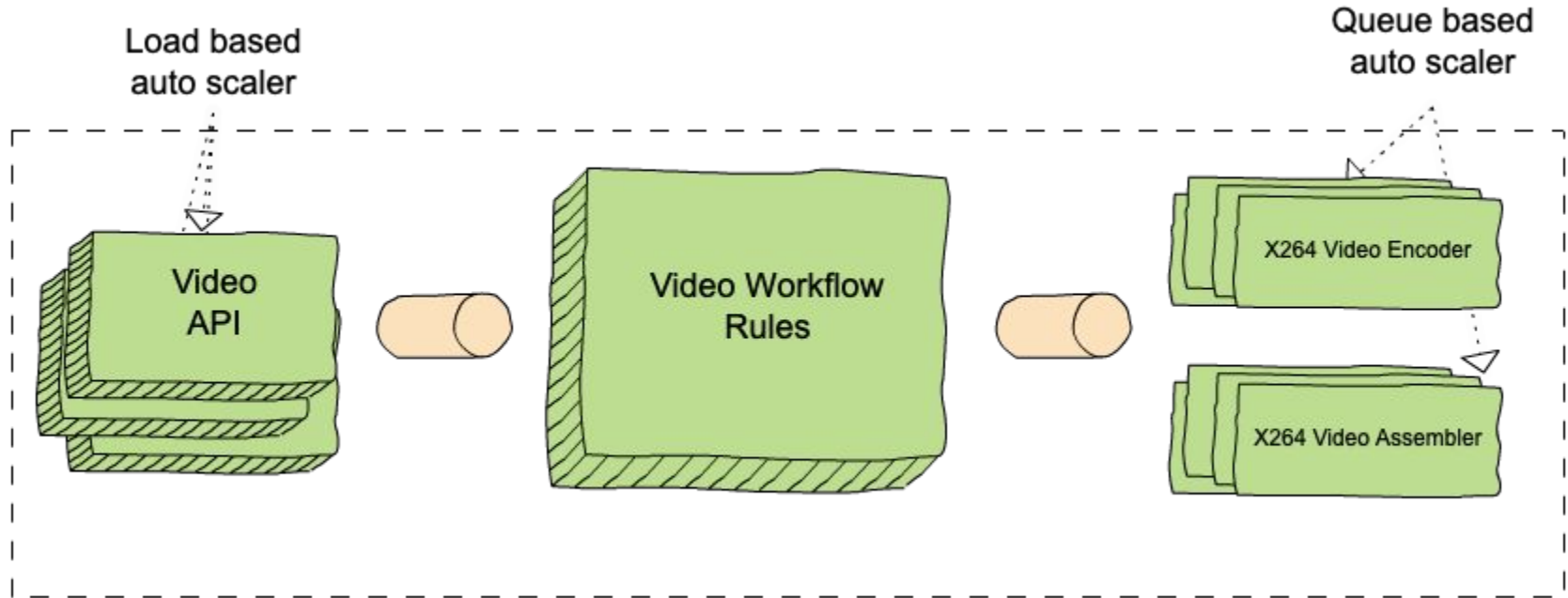
Le service Cosmos

- Même principe, mais ajout de composants
- Séparation du business logic

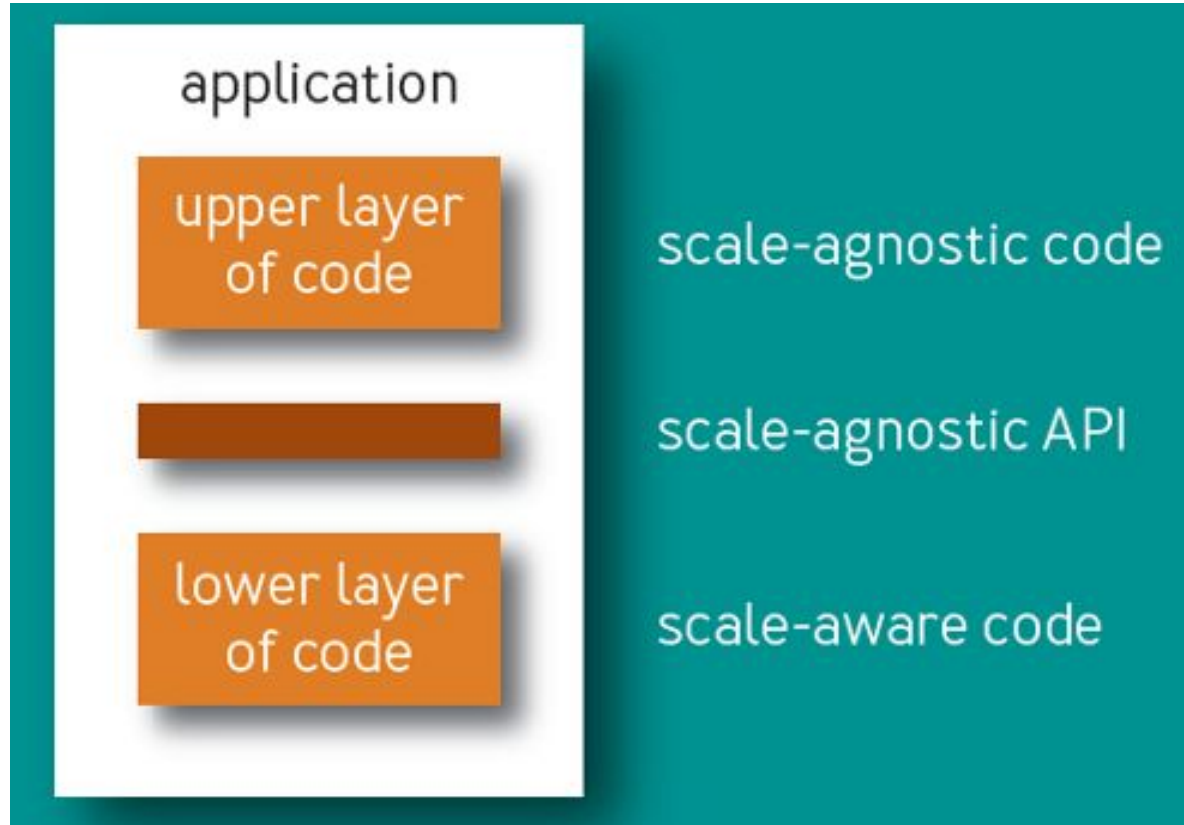


Séparation du business logic

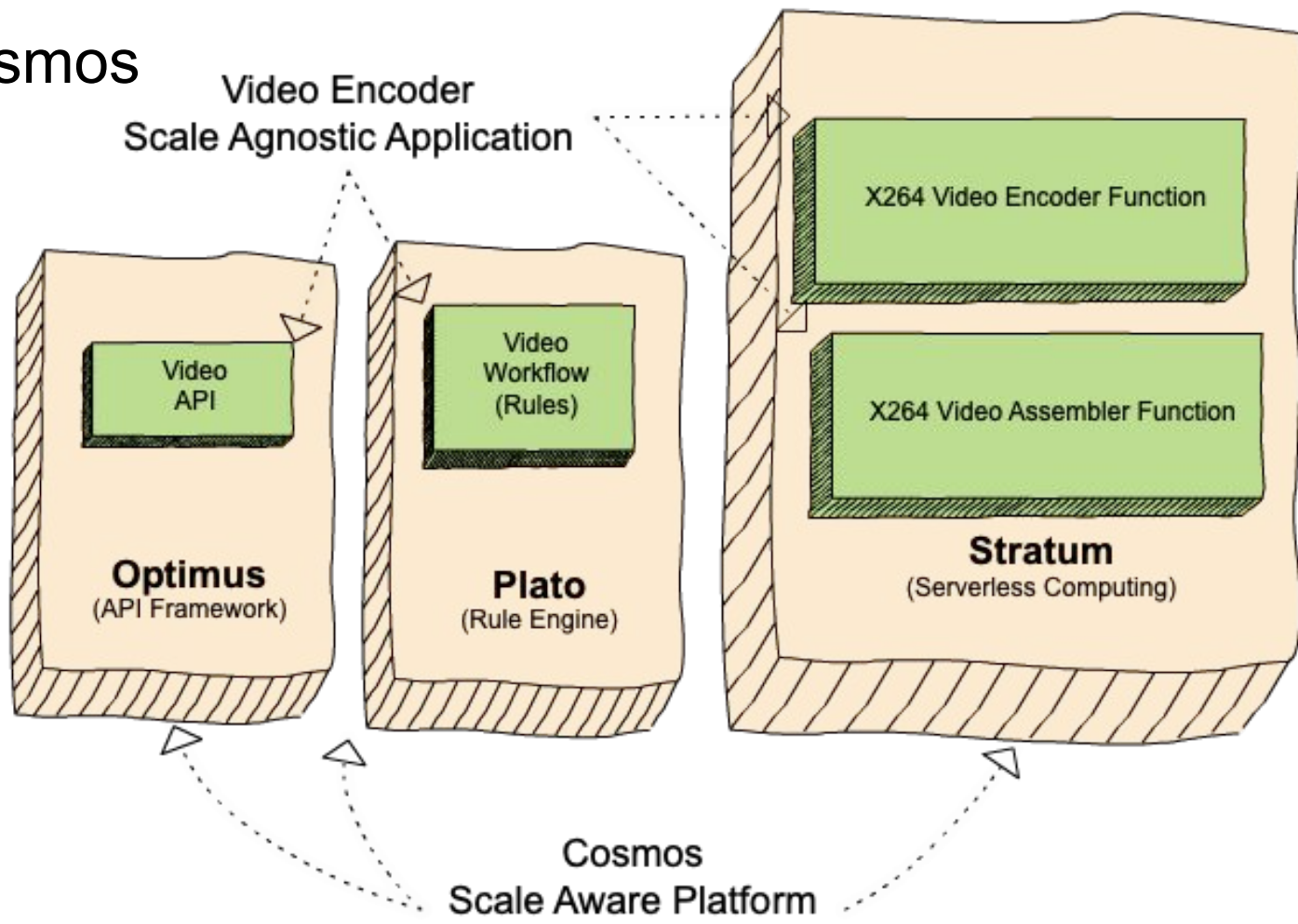
- Plateforme API fournit une abstraction
- Composants scale-agnostic et sous-systèmes scale-aware



Scale-agnostic et Scale-aware

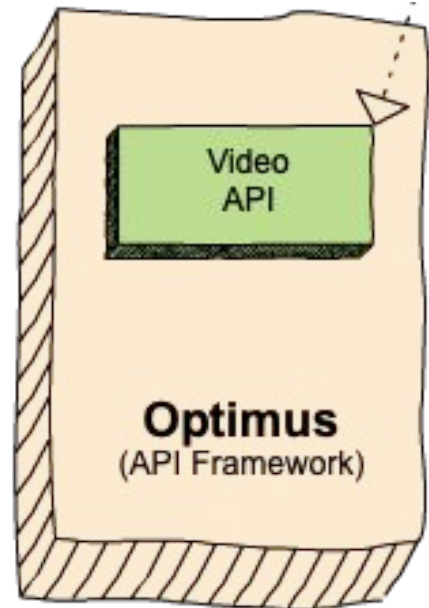


Le service Cosmos



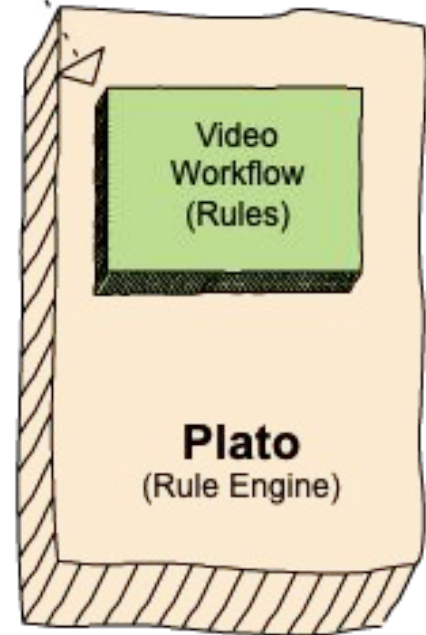
Optimus

- Mapping entre requêtes et business logic



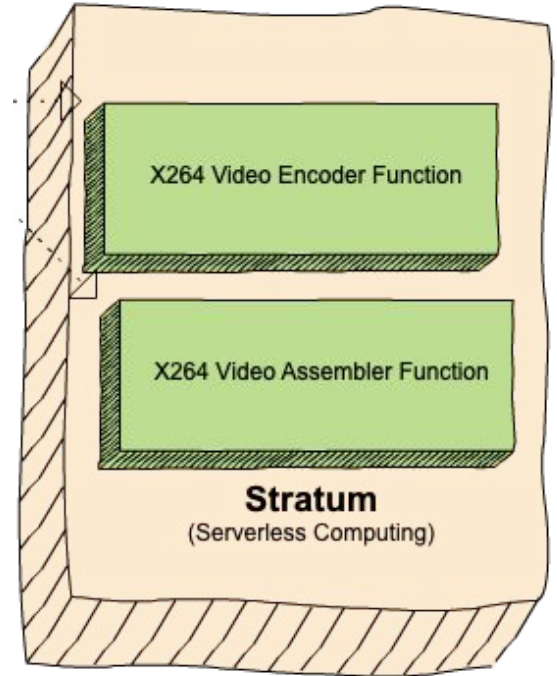
Plato

- Orchestration du workflow
- Définit par le développeur
- 1 règle = 4 parties:
 - Statement
 - Action
 - Reaction
 - Error



Stratum

- Couche serverless
- Regroupe les fonctions asynchrones



Exemple pour une requête



Nirvana

Recent

Fact Search

More

quick search (by project or request ID)



Project

Requests

Tracing

Trace ID

e90db3b8cdbe4f7c9c28f10703f5aea4

Project ID

1308484440208641280



Project-related spans only

Duration: 8 minutes

5

Total Spans: 73 | Project Spans: 68

videox264encodeapi

1

37 ms

46 ms

videox264encode_workflow

55 ms

47 ms

268 ms

181 ms

c.n.v.v.f.x.encodex264

2

6.6 m

c.n.v.v.f.x.assemblex264

3

36 s

c.n.v.v.f.x.generateindex

4

22 s

12:12

12:13

12:14

12:15

12:16

12:17

12:18

12:19

12:20

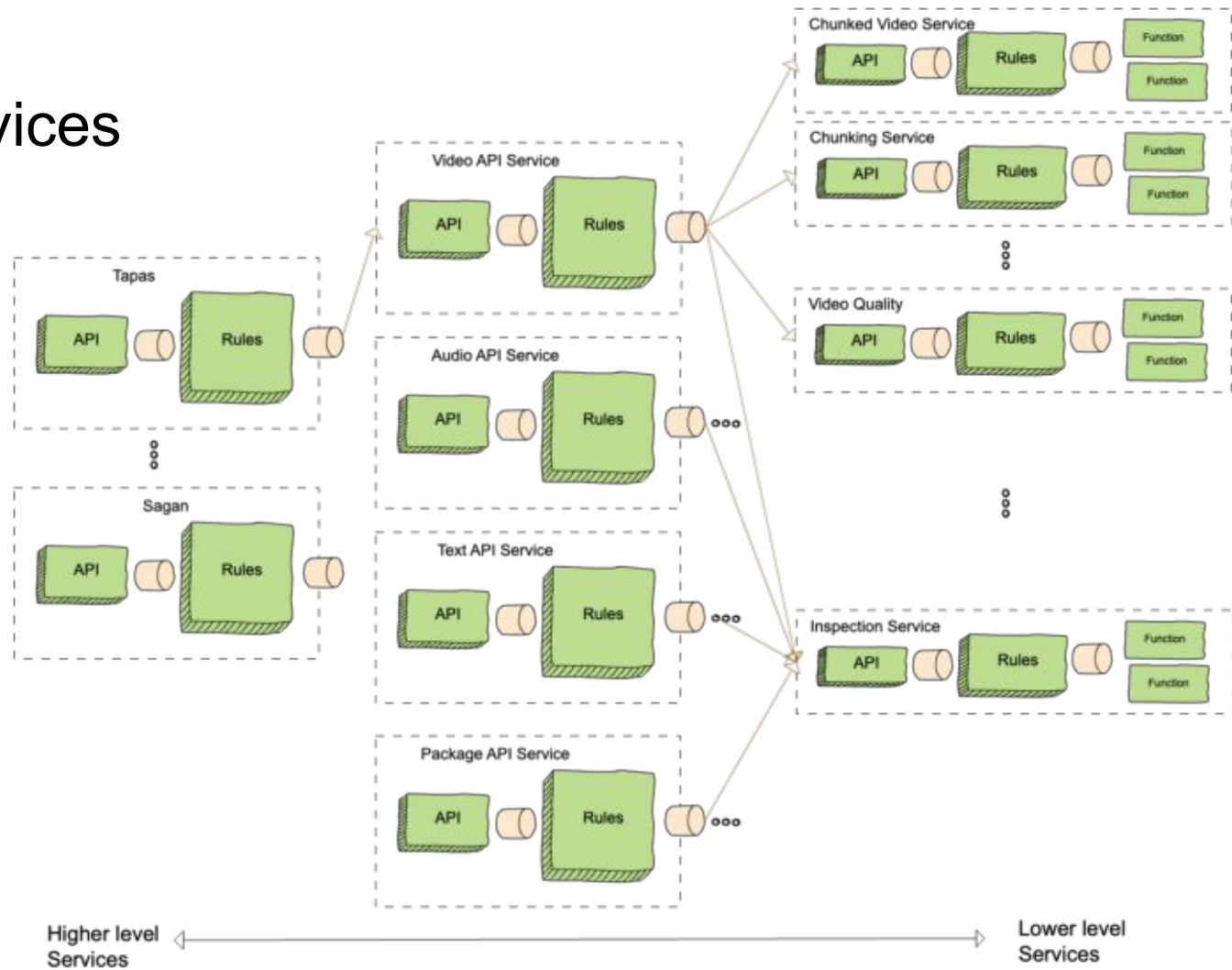
12:21

Layering de services

- Décomposition & superposition de services => architecture modulaire
- Niveaux des services
- Les services sont orchestrés par des services de plus haut niveaux

Layering de services

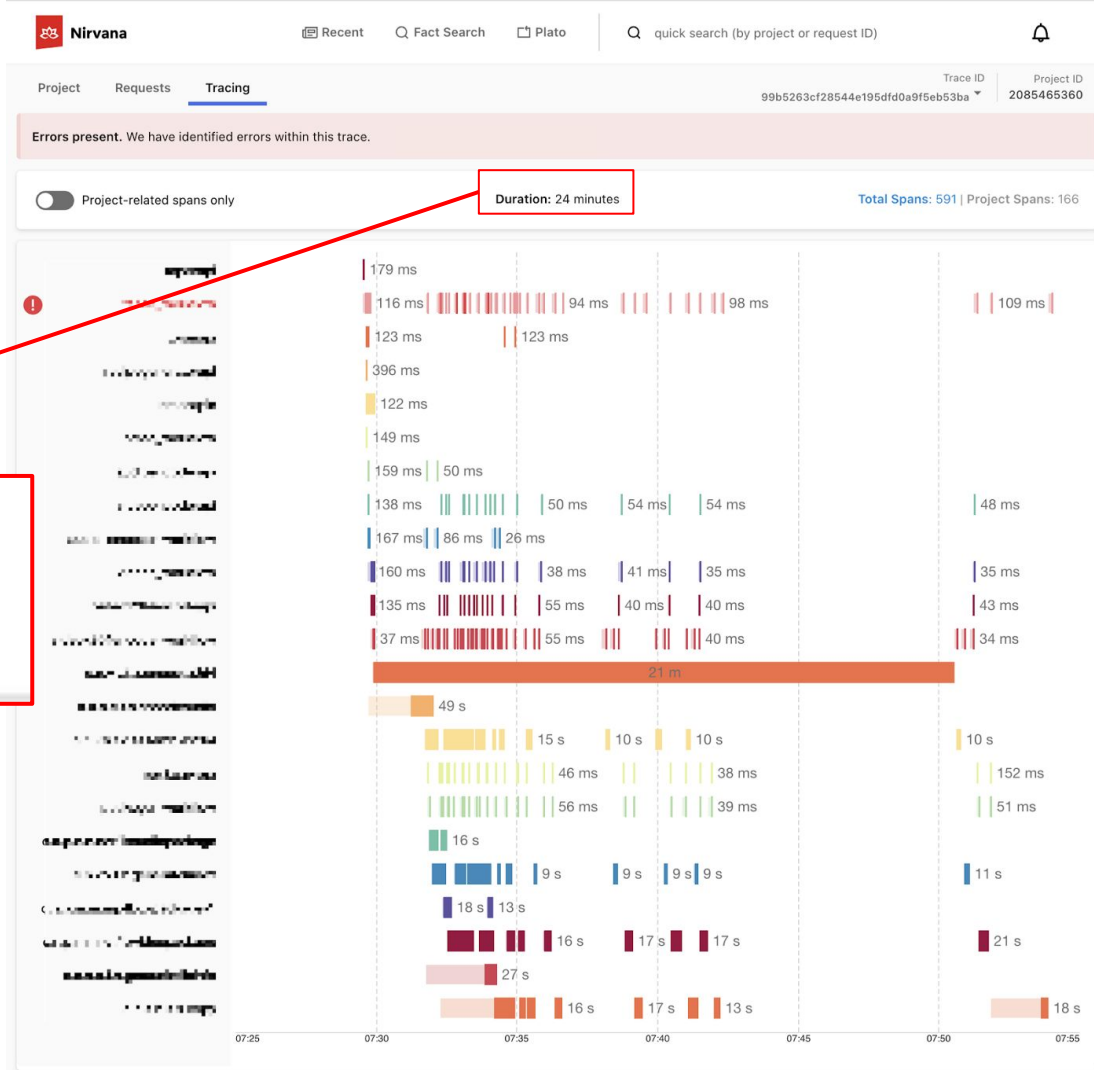
Exemple avec Tapas



Layering de services

Exemple avec Tapas

Duration: 24 minutes



Netflix Video Quality Service

Mesure de qualité pour les vidéo en streaming :

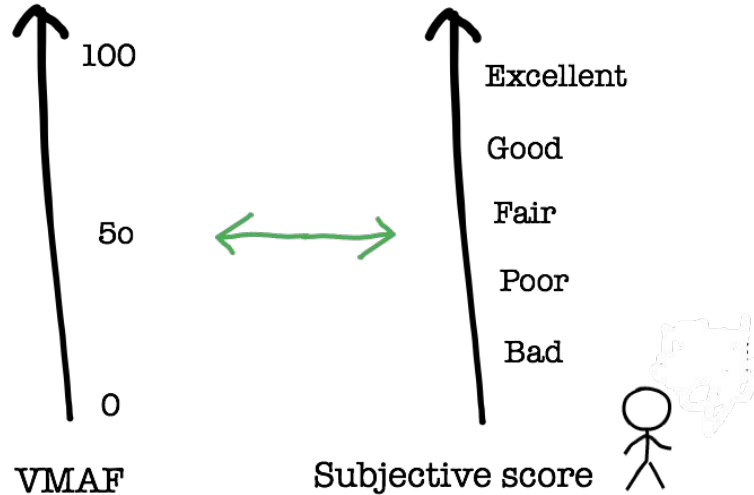
- Video codec comparison
- Amélioration de la Qualité d'Expérience (QoE)
- Video Multimethod Assessment Fusion (VMAF)

Fonctionnement :

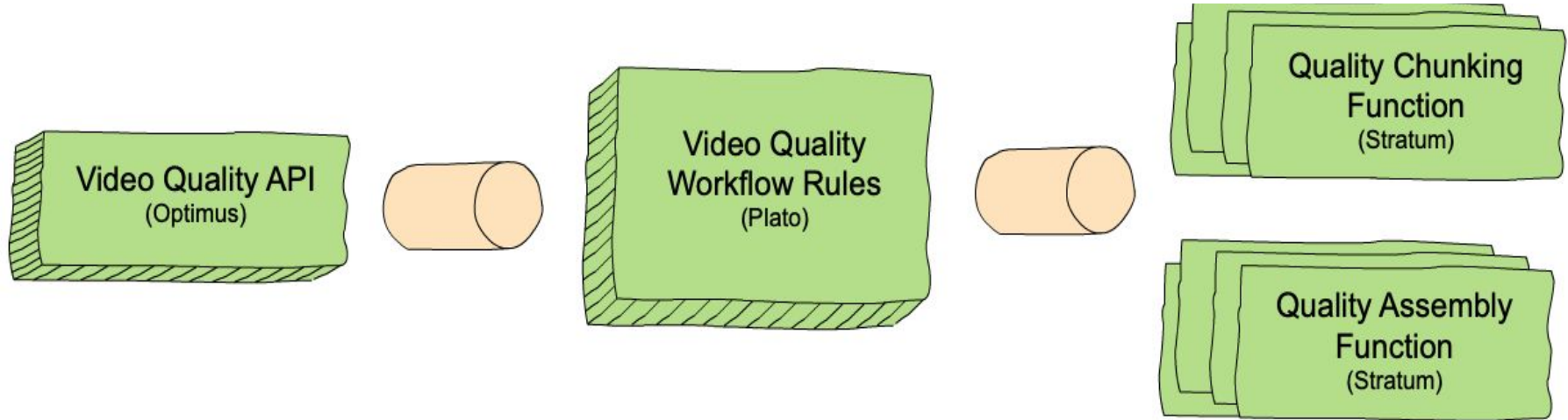
- Découpage de la vidéo en chunk
- Calcule du VMAF sur chaque chunk
- Fusion des valeurs VMAF pour obtenir une mesure VQS

Video Multimethod Assessment Fusion (VMAF)

- Mesure de la qualité du vidéo ressenti



VQS fonctionnement



Points de relais :

- *Request*
measureQuality(Video source, Video derivative)
- *VQS getQuality()*

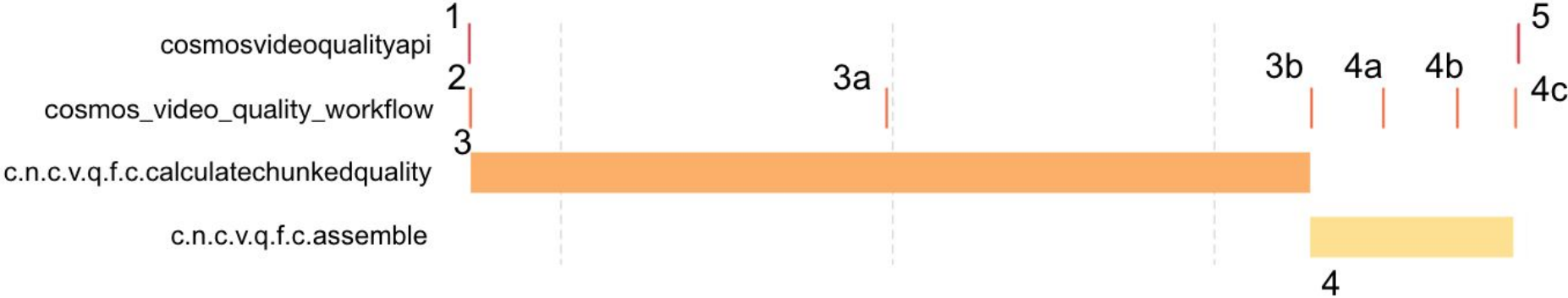
Workflow :

- Découpage en N chunk
- Appelle des fonctions pour le calcul de VMAF
- Appelle de la fonction fusion pour fusionner ces mesures

Couche des fonctions :

- *VMAF*
computeChunkQuality(chunk c)
- *VQS assembleChunk(VMAF vmafC[])*

Workflow VQS



Conclusion

- Technologie récente (2019)
- 40 services pour l'instant
- Création d'un nouveau paradigme
- Efficace sur les systèmes complexes
- Perspectives futures