POC FaaS et Orchestration

du traitement des images d'Observations de la Terre

Sébastien Aucouturier

comet SIL

Orchestration de traitements distribués
mardi 31 mai 2022

cnes
CENTRE NATIONAL D'ÉTUDES SPATIALES

Qui suis-je ?

s.aucouturier@accenture.com

Consultant Opensource et Devops Senior

- Libriste pragmatique
- Contributeur occasionnel
- Formateur dans l'âme
- Ancien Dev qui a mal tourné (qui aime bien les Ops et la Sécu)
- Mon plaisir : prendre une idée, créer une preuve de concept et industrialiser une solution fiable.
- Mon premier ordinateur ? un sinclair ZX81 en 1986.

RTFaaS4EO

Pilotage de Pierre-Marie Brunet

Face à la croissance des données spatiales d'OT couplées à la complexité des algorithmes, le CNES doit faire évoluer ses infrastructures systèmes et ses services tant sur l'optimisation et la robustesse du traitement de ces données que sur l'accès et la manipulation de celles-ci.

L'objectif de cette étude : permettre aux utilisateurs finaux (communautés scientifiques ou industrielles) de se recentrer sur leur coeur de métier à savoir valoriser, exploiter et faire parler les données d'OT en s'affranchissant de toutes les contraintes qui en découlent. Ainsi ils s'affranchissent du déploiement de framework de traitements, leur administration, et toutes les difficultés qui en découlent tout en utilisant de manière optimale les ressources de calculs

- Comment relever ce défi ?
- Est-ce qu'une architecture serverless ou FaaS va tenir toutes ses promesses ?
- Quels sont les avantages et les inconvénients d'une telle architecture avec les cas d'usages du CNES ?

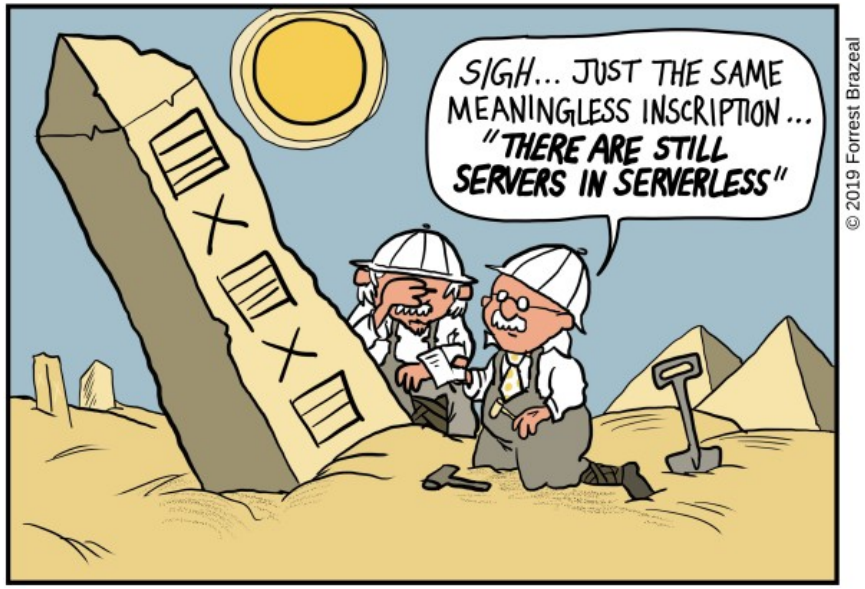
FaaS

What is it?

est un nouveau modèle de cloud computing

On-Premises	IaaS	CaaS	PaaS	FaaS
Functions	Functions	Functions	Functions	Functions
Application	Application	Application	Application	Application
Runtime	Runtime	Runtime	Runtime	Runtime
Containers	Containers	Containers	Containers	Containers
Operating System	Operating System	Operating System	Operating System	Operating System
Hardware	Hardware	Hardware	Hardware	Hardware

managed by you managed by platform provider

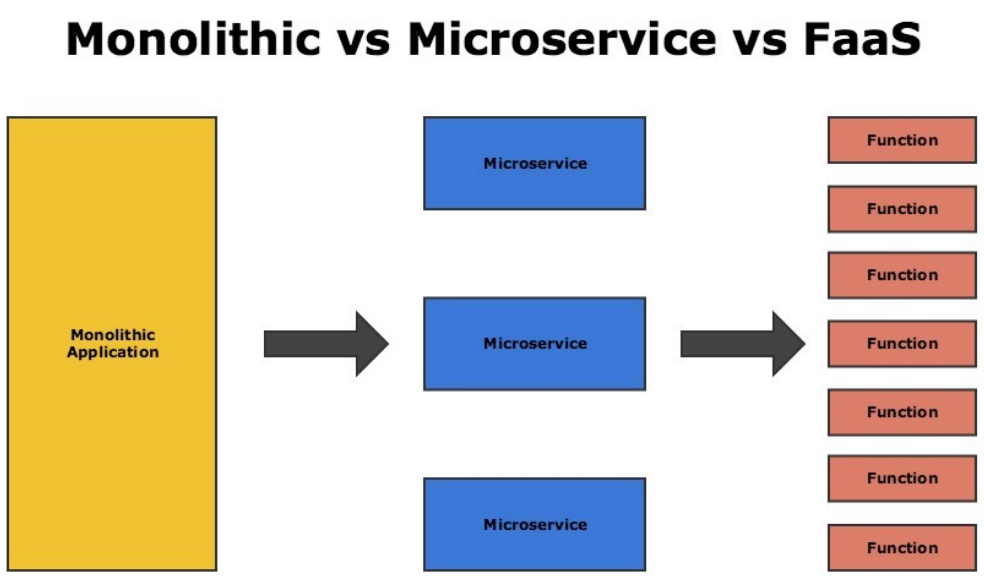


FaaS utilise une architecture serverless pour permettre aux développeurs de déployer, mettre à jour et faire évoluer facilement des applications sans avoir à gérer de serveurs.

Fonction

What is it?

Définition : Une fonction est une tâche ou une opération spécifique écrite sous la forme de code discret et exécutée indépendamment.



Note : Si un microservice n'exécute qu'une seule action en réponse à un événement, il peut être considéré comme une fonction.

Orfeo Toolbox (OTB) est une bibliothèque de **fonctions** de traitement des images des satellites d'observation de la Terre.

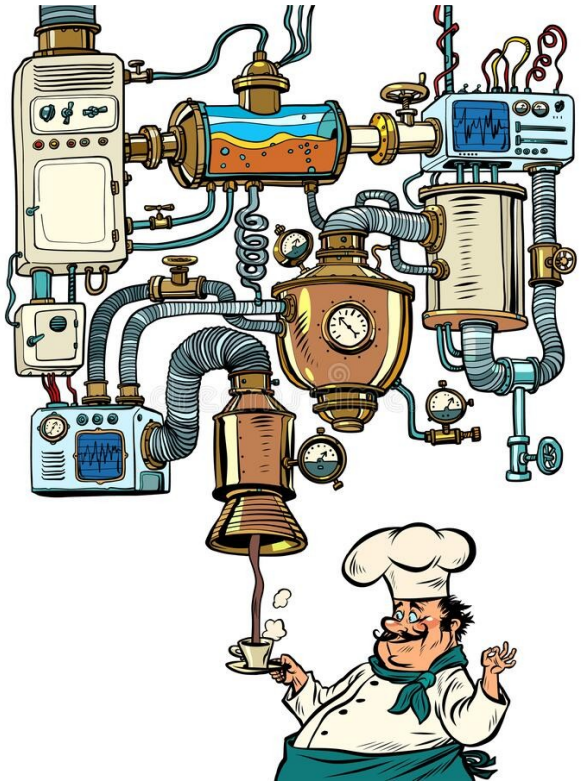
Les solutions FaaS existantes

Avec l'essor de Kubernetes, et sa facilité pour conditionner, déployer et exécuter des conteneurs à grande échelle, plusieurs solutions FaaS vu le jour :

- FaaS **Managé**:
  - Lambda AWS
  - Azure Functions
  - Google Cloud Functions
  - Scaleway Elements Serverless Platform

- FaaS **Opensource** :
  - **OpenFaaS**
  - Kubeless
  - **Knative**
  - **Nuclio**
  - **Fission**
  - Fn Projects
  - Dispatch
  - OpenWhisk
  - OpenLambda

- **Data Processing** as Function **Framework** :
  - Actinia
  - Lithops
  - Pesto



Les solutions FaaS Managées

ne sont pas adaptées aux défis du traitement des données scientifiques

	Function Timeout	Memory	Temporary disk space
AWS Lambda	900s	Up to 3GB	500 Mb
Azure Functions	300s	Up to 1.5 Gb	512 Mb
Google Cloud Functions	540s	Up to 2Gb	tmpfs : it consume memory
Scaleway (Beta)	No limitation	Up to 2Gb	unknown

L’heure du choix

Dans le cadre de cette étude, le choix pour construire une preuve de concept, s'est porté sur :

OpenFaaS

Pourquoi ?

- Possibilité de réutiliser des dockerfiles existants (avec quelques ajouts) et développés par le CNES pour les images FaaS

- Suffisamment fonctionnel sans être complexe (mais avec des écueils, nous reviendrons dessus)

- Ne nécessite pas un énorme Cluster K8s pour fonctionner

- Présence d'une forte communauté open source avec un leader actif : Alex Ellis

Les écueils

- Contourner la problematique de l’autoscaling natif de kubernetes (HPA)

Comment ? Extrait d'un échange que j'ai eu avec Sebástian Risco, chercheur de l'Université Polytechnique de Barcelone :
" we found that Kubernetes Jobs is the best approach to handle long-running and heavy memory consumption functions." https://github.com/seb-835/openfaas-job-worker

- Gérer les “long time running functions”

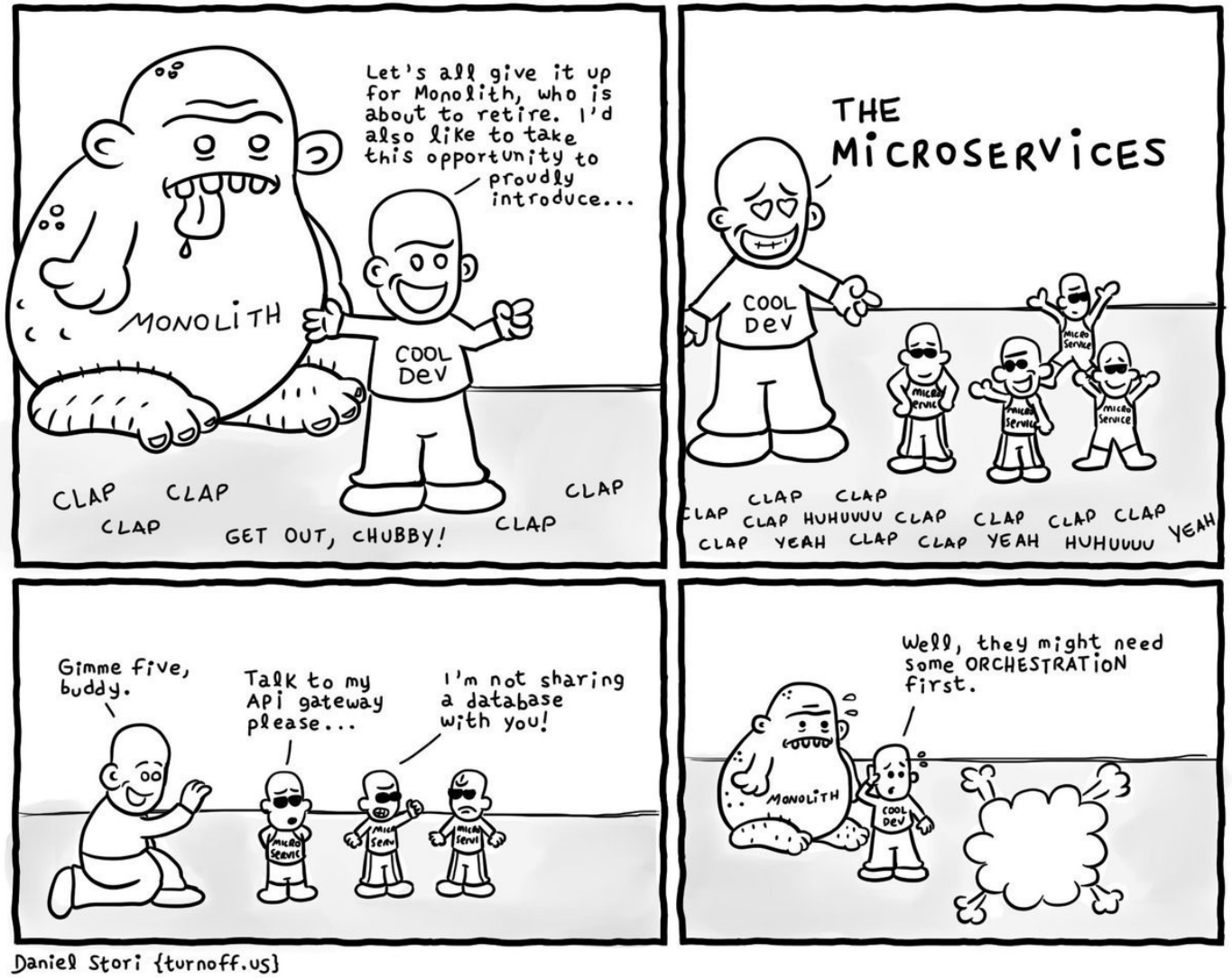
Comment ? Il est préferable, fortement recommandé, **Obligatoire**, d'utiliser un mode de requête Asynchrone.

- Accéder aux Données avec OpenFaaS

Comment ? En utilisant un operateur comme l'Operateur Datashim : https://github.com/datashim-io/datashim

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 825061.

Orchestrer le FaaS



Choisir le bon orchestrateur

- Comment gérer les fonctions de longue durée (LongTime Running)

- Comment gérer les "Failure" et les "Retry"

- Construire une application Stateful à partir de fonctions Stateless

- Comment l'interfacer à une infrastructure FaaS executée dans un environnement cloud natif (kubernetes)

Nous avons restreint notre étude aux solutions opensources suivantes :

Zeebe
NetflixConductor
Uber Cadence
Airflow ETL

Des approches logicielles différentes

- décrire la planification des tâches/micro-services/fonctions à l'aide d'un graphique BPMN.

(Zeebe, Netflix Conductor)

- implémenter l'ordonnancement As Code des Tâches/Micro-Services/Fonctions.

(Uber Cadence, Air Flow ETL)

Orchestrateur vs DAG (Directed Acyclic Graphs)

La conduite de micro-services ou d'une architecture événementielle ne peut être gérée que par un orchestrateur.

(Gestion de Boucle, de reprise sur incident, ...,)

mais parfois nous n'avons besoin d'exécuter qu'un seul pipeline ETL (Extract, Transform , Load),

un DAG est suffisant alors pour concevoir le workflow.

L’heure du choix, le retour ...

Zeebe

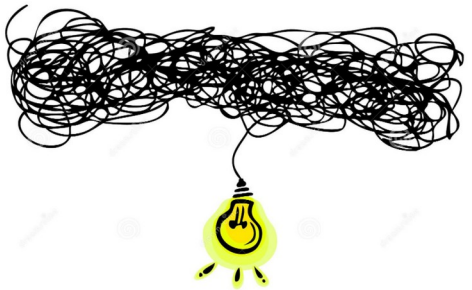
Pourquoi ?

- Solution déjà mise en oeuvre au CNES dans le cadre d’un projet

- Possibilité de tester différentes approches graphiques : Boucles, Tests , Map Reduce
- Présence d'une forte communauté open source :

https://github.com/camunda-community-hub/
https://camunda.com/developers/

Les difficultés



- Comment appeler nos FaaS depuis Zeebe :

développement d’un worker zeebe2FaaS en python, utilisant la bibliotheque pyzeebe.

- Passage des Paramètres des Tâche aux FaaS :

mise en place de templating

- Gestion de l'asynchronisme des fonctions avec Zeebe :

au travers des objets “received task”

Preuve de Concept

Automatiser une chaîne de pré-traitement d'images

Chaine de traitement couple (P_, MS_) :
(d'aprés un notebook de Carole Amiot)

tache 1 : OTB BundleToPerfectSensor
tache 2: OTB Orthorectification
tache 3: Python call_normalize_product (algo de P.Lassale)

Preuve de Concept : FaaS

Faas Asynchrone : OTBCLI et normalize-product-function

tache1:
function : otbcli
cmdline: BundleToPerfectSensor -inp {P_} -inxs {MS_} -out {S3PATH}/bundle.tif -elev.dem /mnt/datasets/nfs-datalake-job

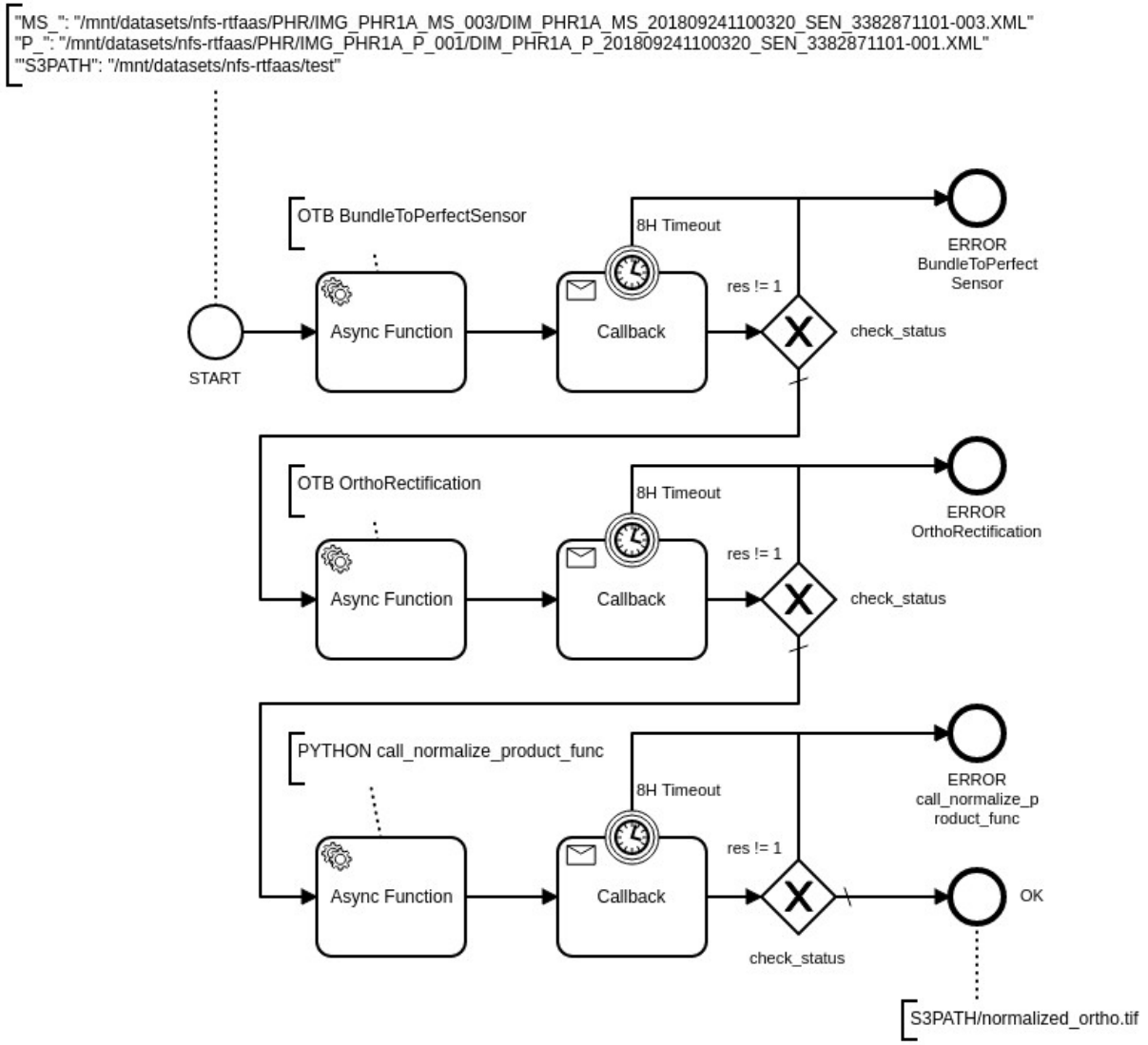
tache 2:
function : otbcli
cmdline: OrthoRectification -io.in {S3PATH}/bundle.tif -elev.dem /mnt/datasets/nfs-datalake-job -elev.geoid /mnt/datasets/nfs-rtfaas-job/egm96.grd -io.out {S3PATH}/ortho.tif

tache 3:
function : normalize_product_function
cmdline: call_normalize_function --in_ms {MS_} --in_ortho {S3PATH}/ortho.tif -out {S3PATH}/normalized_ortho.tif

FaaS Synchrone : callback-ack

passé en paramètre et appelé par OpenFaaS en fin de traitement d'une tâche

Preuve de Concept : workflow

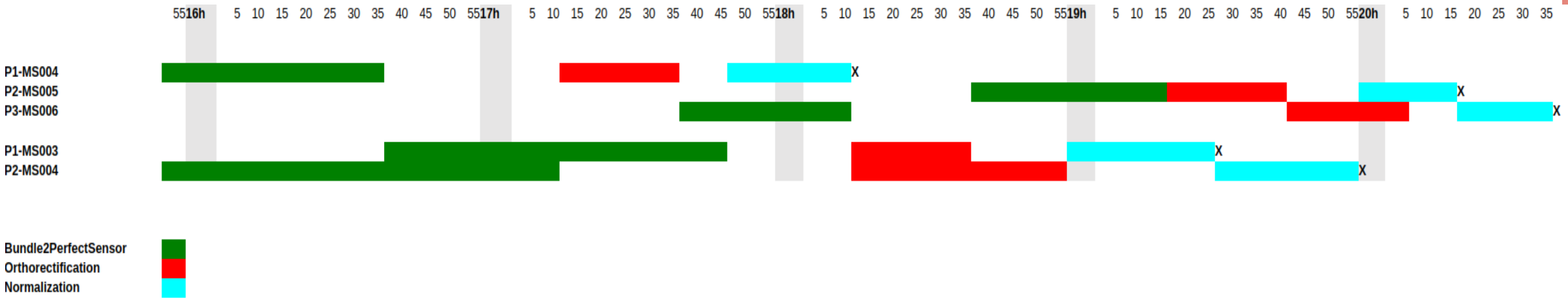


RUN

Sur le node553, l'orchestrateur de K8s va temporiser l'exécution des tâches selon les ressources disponibles et les ressources requises.

La normalisation d'une image consomme jusqu’a 49 Go de mémoire :(et j'ai assigné un minimum de 4 coeurs par FaaS.

L'exécution en parallèle de 2 produits : 1 stereo et un tri-stereo sur ce run de test a pris : 4h40



Conclusion

La preuve de concept est fonctionnelle ! et plutôt stable.
Même si nous avons du trouver et mettre en place des solutions alternatives dans le cadre du traitement d'images scientifiques par des FaaS

Mais il reste encore beaucoup de points à gérer avant une industrialisation :

- Gestion des erreurs, reprise sur incidents
- Sécurisation des échanges zeebe / OpenFaaS
- Amélioration des performances (S3, NFS)
- Gestion de l'InfiniBand et des GPUs avec le cluster K8s

En tant que DevOps, je la trouve assez intéressante, mais vous ? peut -elle vous aider au quotidien ?

Des Questions ? Des Interrogations ? A vous …

Thank You

>