Energy efficient service @Etat du Valais

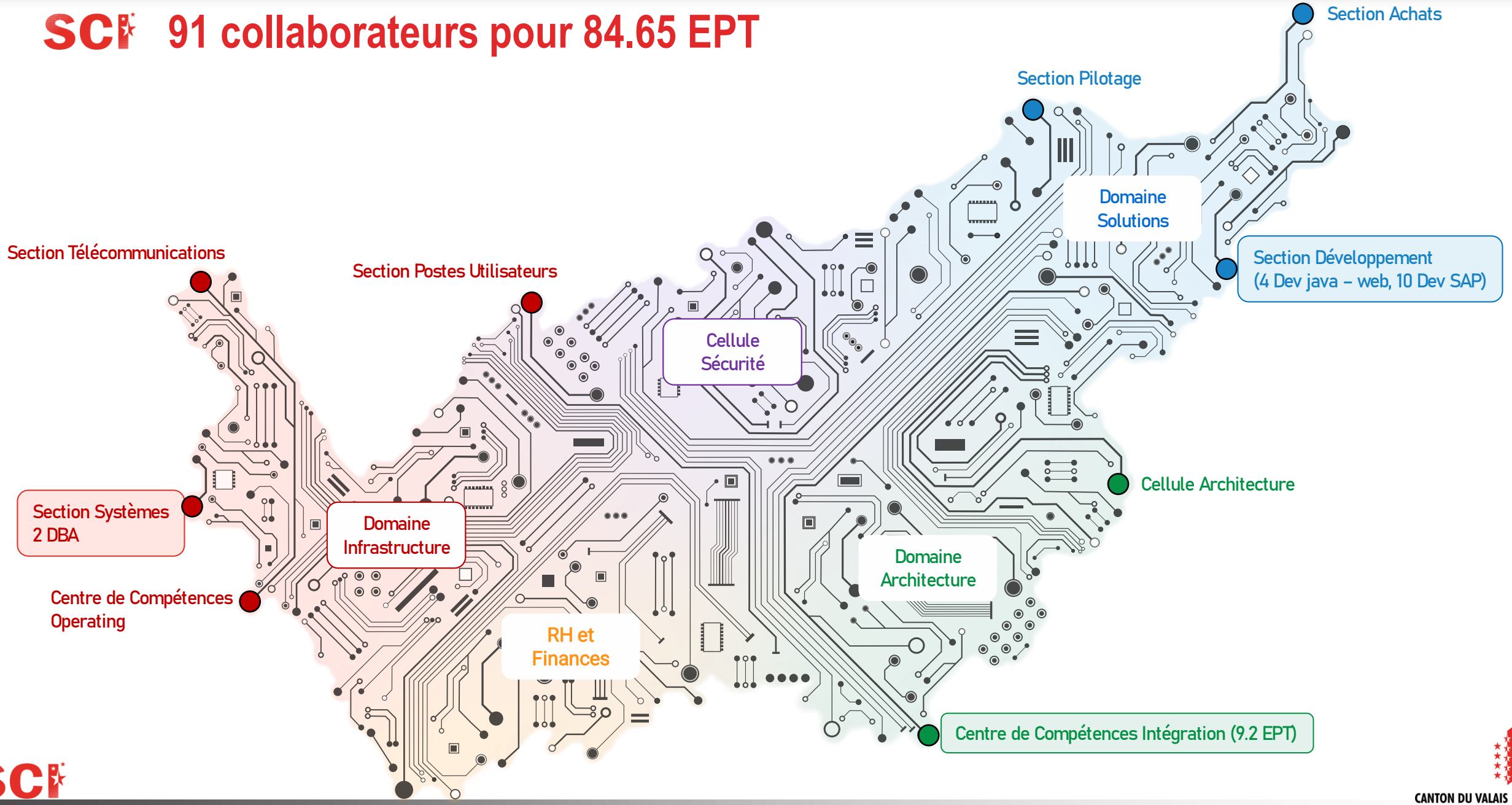
Un nouveau levier économique : l'efficience énergétique

Steve Favez : Architecte et chef du centre de compétences intégration

Agenda

- Aide ANS
- Contexte (SCI, plateforme applicative)
- Constats
- Remédiations possible
- Implémentation
- Conclusion
- Futur
- Q&A

91 collaborateurs pour 84.65 EPT



2000+ applications hébergées par le SCI

- 5 Départements pour 50+ Services

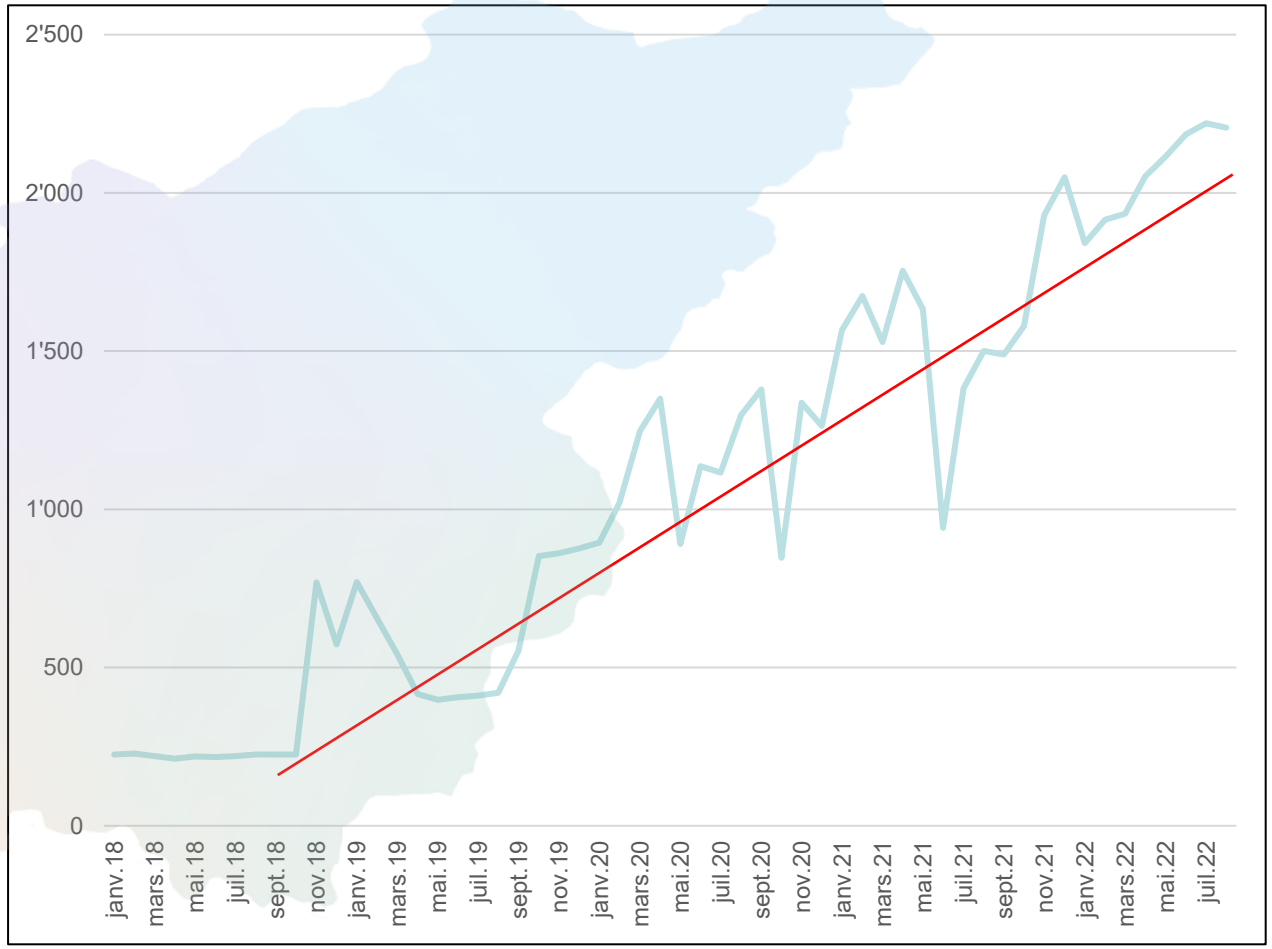
- 1300+ Logiciels métiers spécifiques
- 240+ Logiciels de bureau
- 500+ Logiciels d'infrastructure
- 220+ Projets informatiques par année
- 54'000+ Demandes de support

3600+ pods déployés sur OpenShift

- 100+ Applications développées et déployées
- 50+ Solutions externes déployées
- 4 Clusters
- 60+ VMs
- 5 Environnements
- 300+ NamesSpaces

Plateforme applicative – containerisation sur OpenShift

- Fondations technologiques
  - Orientation micro-service en se basant sur JEE, puis SpringBoot
    Développer et déployer plus rapidement
  - Utilisation intensive de la containerisation avec Docker, puis passage à RH OpenShift
- Résultats métiers
  - Capacité de fournir beaucoup plus rapidement des prestations métiers
  - Plateforme permettant d'embarquer aussi bien des équipes internes que des prestataires / solutions externes.

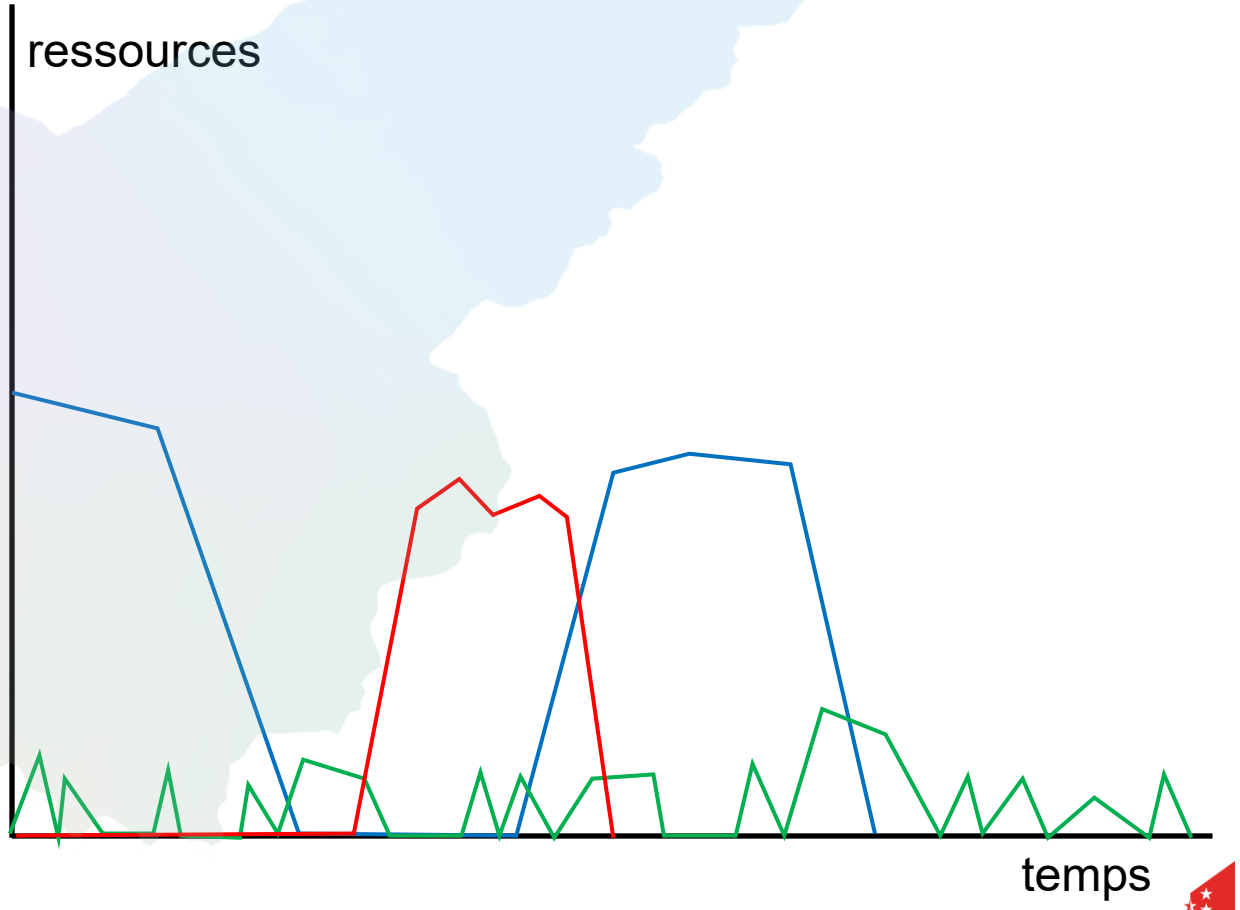


Constats – un besoin grandissant de CPU et de mémoire

- Stack Java
  - En s'appuyant sur les technologies java, chaque application demande pas mal de mémoire et de CPU.
    - Un container JBoss eap requiert en Moyenne 2 GB de mémoire
    - Un container SpringBoot 1 GB de mémoire
  - Le tout, généralement sur 4 environnements (int, tst, qas et prd)
- Faiblesses
  - Consommation mémoire et CPU élevée, même avec SpringBoot
  - Une manière peu efficiente de gérer les ressources, très peu d'automatisation de l'élasticité

Constats – utilisation non intensive des applications

- L’accès aux prestations / applications ne se fait pas de manière continue.
  - Horaires de bureau pour les employés
  - Par période pour certaines prestations (pêche, chasse, paiements directs, etc.)
  - Aléatoirement, de jour comme de nuit, mais pas une charge excessive pour nos systèmes
  - Beaucoup d’applications, mais population et sollicitation limitée
- Et répliqué sur plusieurs environnements. Soit en moyenne annuelle, 60% de nos ressources non utilisées.



Constats – le syndrome de la dinde de Thanksgiving

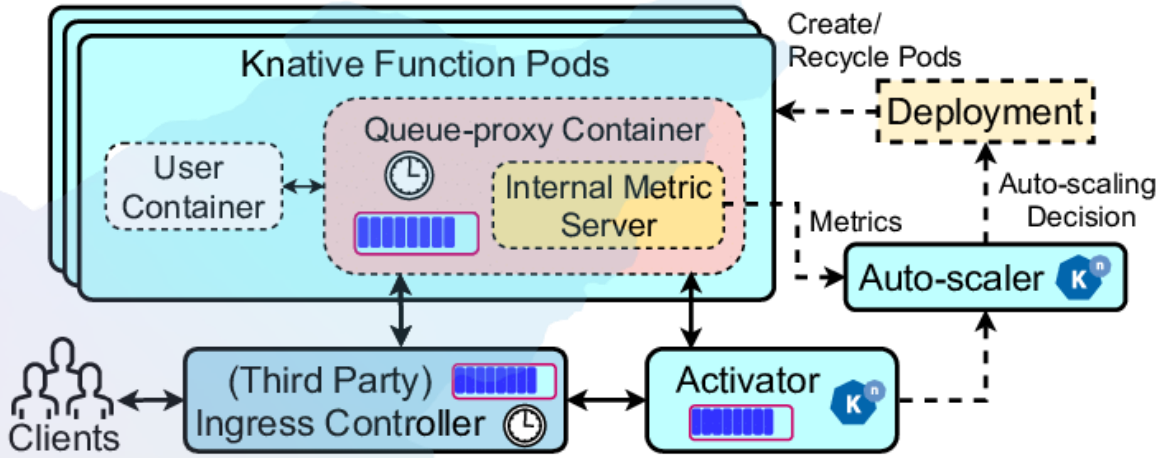
- Pour fournir une prestation.
  - Une application containerisée
  - Qui est exécutée sur un nœud “k8s” (un serveur
  - Qui est exécuté dans une VM sur un cluster ESX
  - Qui est exécuté sur un serveur physique
- Utilisation de ressources virtuelles pour exécuter un container - soit, en moyenne, 30% des ressources en plus et les coûts de licences inhérentes

Remédiations

- Différentes solutions qui sont complémentaires
  - Utilisation de l'élasticité k8s : Knative
  - Utilisation de frameworks de développement plus légers en ressources
  - Passage à OpenShift sur bare metal
- Buts escomptés
  - Diminuer l'utilisation des ressources physiques (énergétique)
  - Diminuer la taille requise des clusters (énergétique et économique)
  - Supprimer les couches inutiles (énergétique et économique)

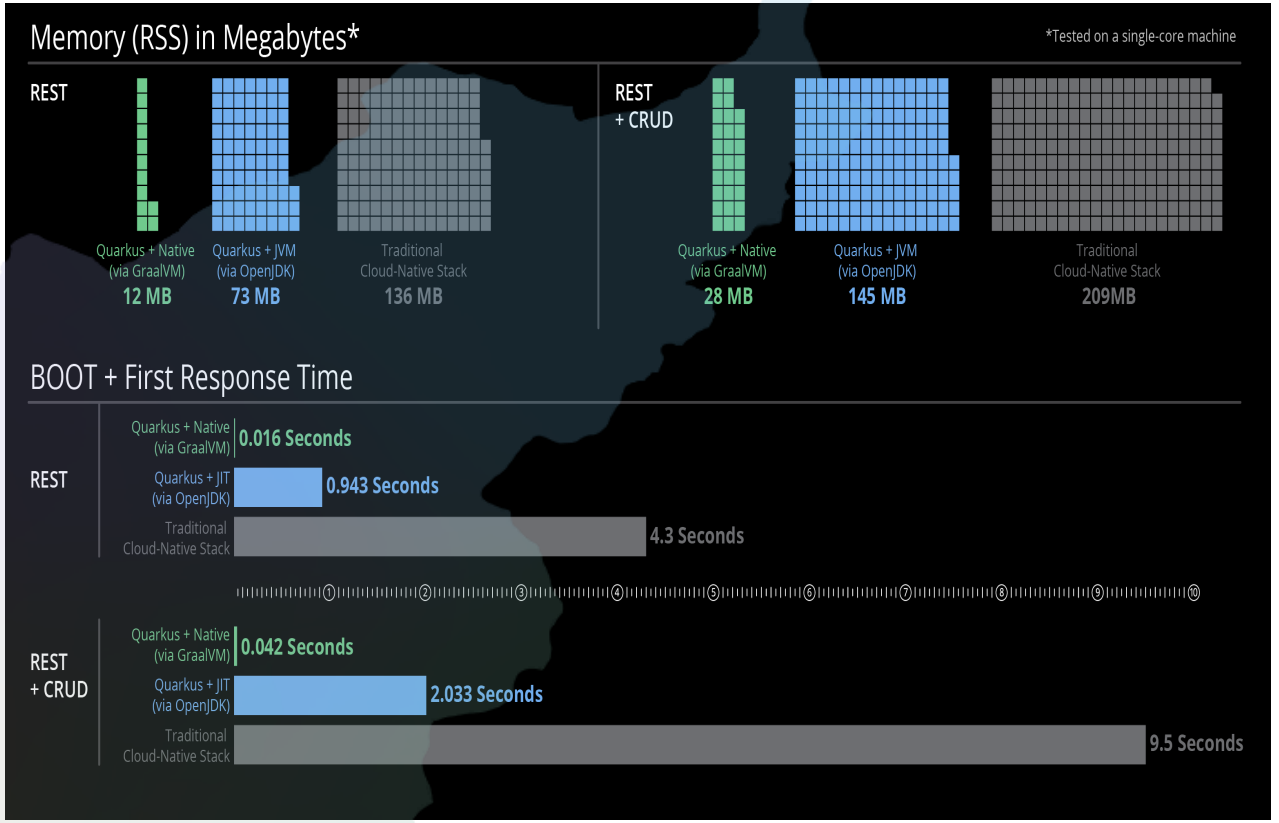
Remédiations – Knative

- Pour des applications “event driven” et “serverless” : https://knative.dev/docs
  - Démarrage et Arrêt des pods en fonction de la charge – événements externes (requête HTTP / évènement)
  - Totalement intégré au monde Kubernetes fonctionne avec des “sidecar” / “proxy” qui interceptent les événements pour “scaler” les pods.
- Mais…
  - Requiert des applications serverless, soit capables de démarrer et de se stopper très rapidement.



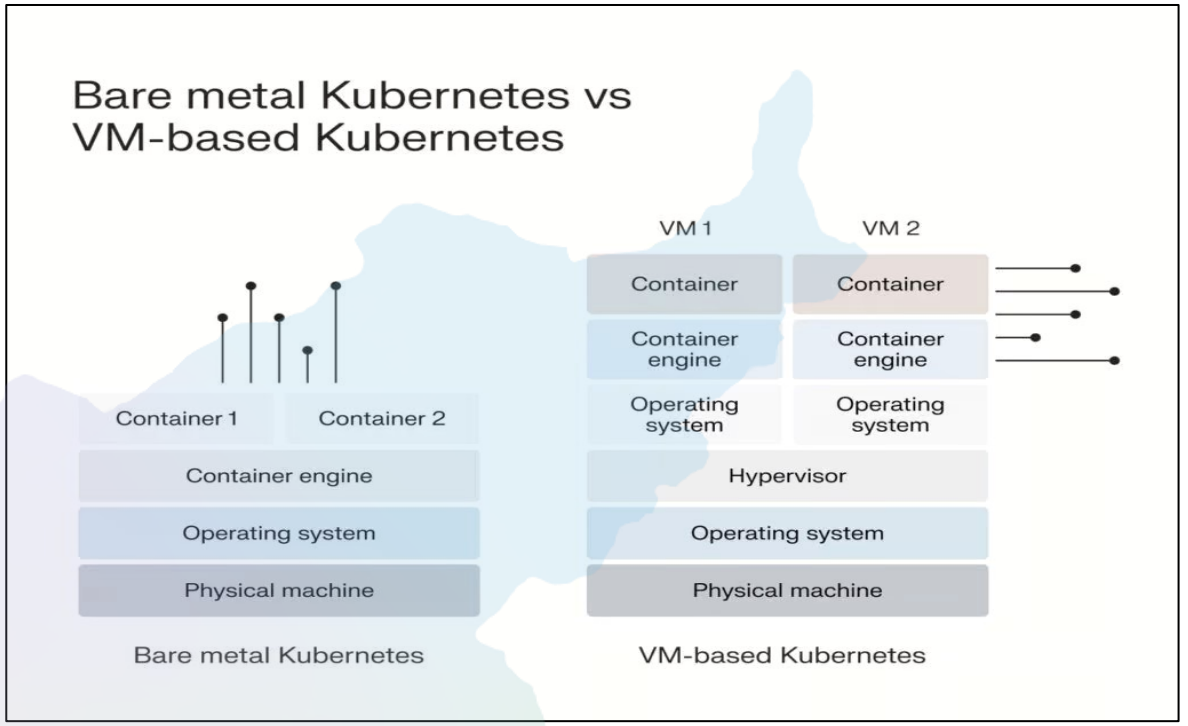
Remédiations – Quarkus

- Sans quitter le monde Java
  - Applications plus légères, plus rapides et consommant moins de mémoire
  - Compilation AOT
  - Gains sur le papier
    - 3 à 4 fois plus rapide au démarrage qu'une app standard et 100 fois plus rapide en native.
    - Consommant entre 30% et 90% moins de mémoire.
    - Image beaucoup plus légère.



Remédiations – K8s sur de baremetal

- K8s/OpenShift fonctionnent depuis quelques années très bien sur du baremetal
- Les performances sont clairement meilleures
- https://thenewstack.io/does-kubernetes-really-perform-better-on-bare-metal-vs-vms/



K8s cluster type	CPU		RAM latency (write/read)	Storage		Network	
	Speed (execution time)	Utilization		TPC (transactions per second)	Latency	Bandwidth	Latency
VM	47.07 sec	86.81%	174.53 / 173.75 ms	4,636	55.21 ms	6.52 MB/sec	145 us
Bare metal	21.46 sec	43.75%	62.02 / 47.33 ms	12,029	21.28 ms	31 MB/sec	24.5 us

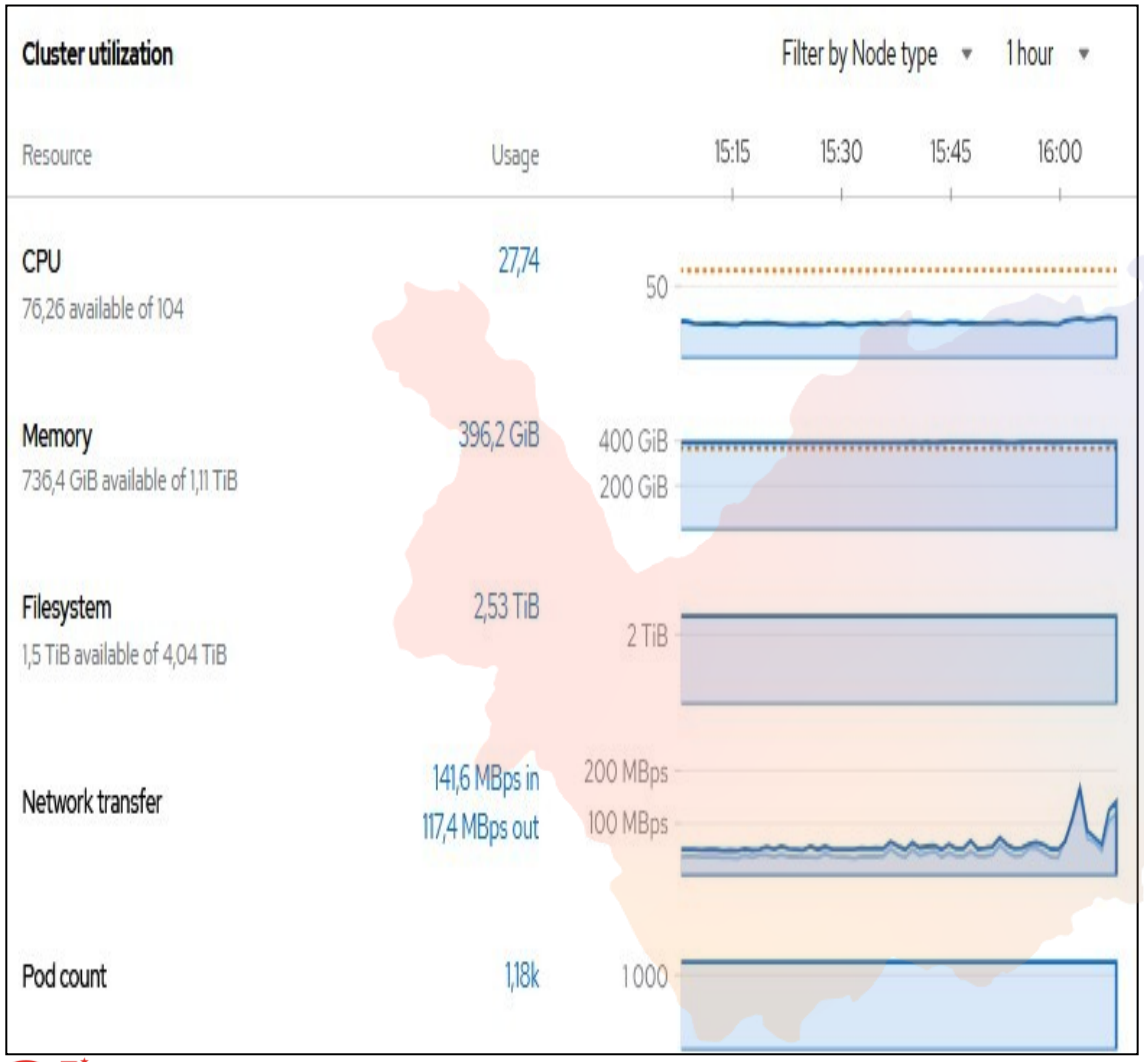
Implémentation Knative – serviceMesh

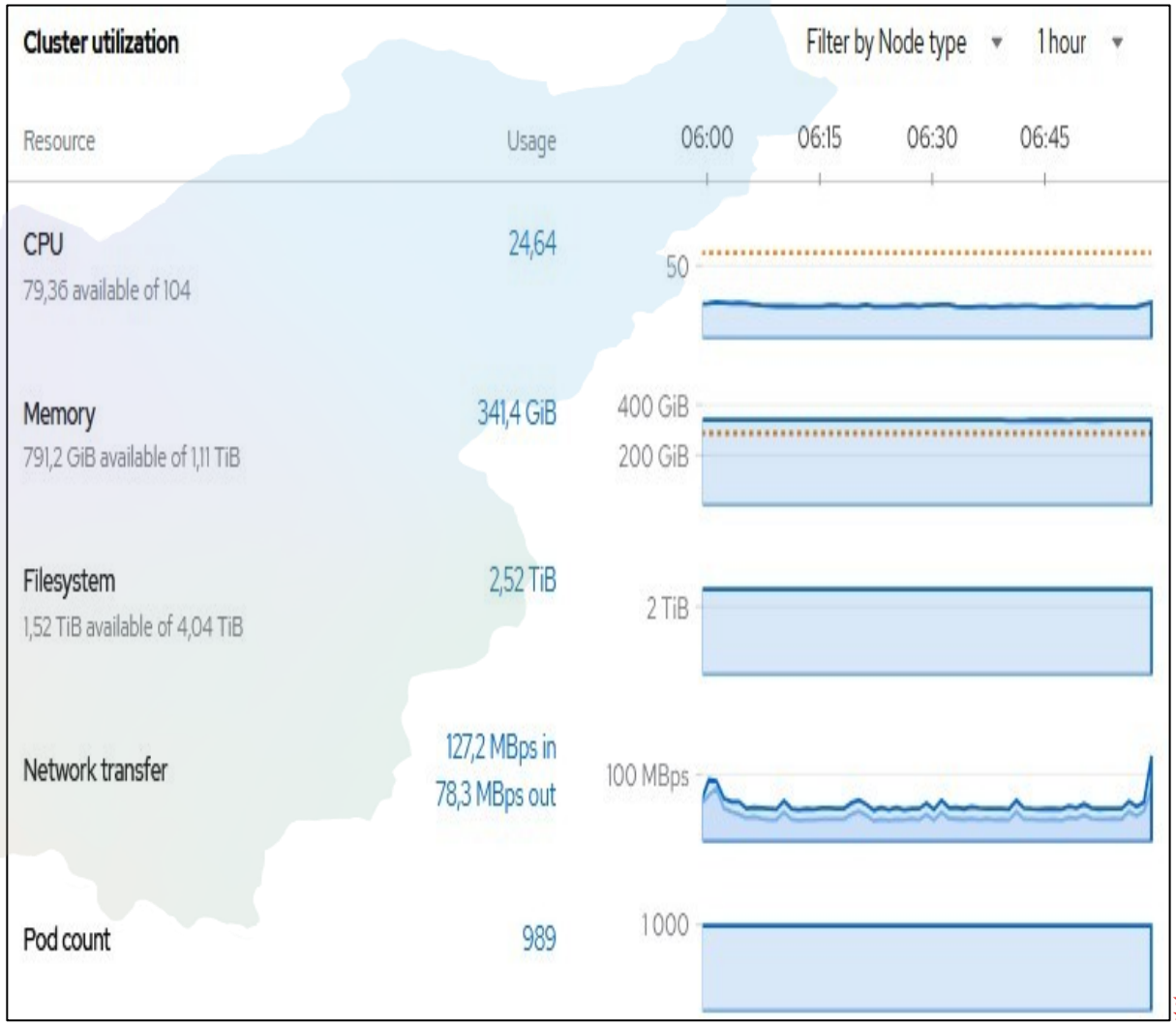
- Relativement simple à mettre en place sur un cluster k8s
- Pas de changement au niveau des images
- Par contre – version de base ne supporte pas les “paths”

myapps-svc	**service myapps-svc**	**ingress host & path myapps.vs.ch/svc**
myapps-front	service myapps-front	ingress host & path myapps.vs.ch

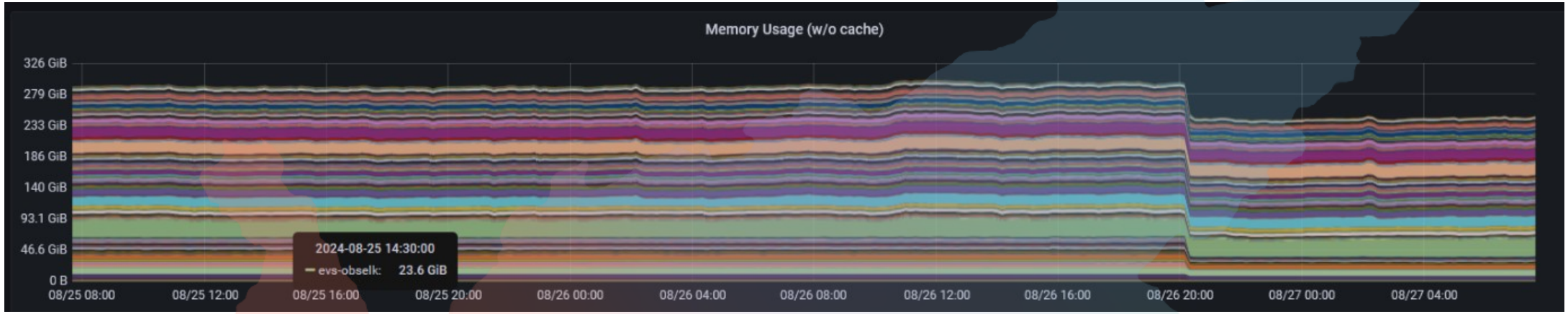
- Solution alternative pour tester à l’échelle :
  - Batch stoppant les Pods inutiles de manière journalière – sur serveur de test (int – tst et qas)

Implémentation Knative – serviceMesh





Implémentation Knative – serviceMesh



	Gain brut	Utilisé avant	Utilisé après	Gain en %
Pods	191	1180	989	16%
CPU	3.1	27.74	24.64	11%
Mémoire	54.8	341.4	341.4	16%

SCI

CANTON DU VALAIS
KANTON WALLIS

Implémentation – baremetal

- Prochaines étapes
  - Passage à un clustering baremetal (accélérateur Broadcom)
  - Capitaliser sur ServiceMesh (Istio) pour avoir une version complète de Knative (support des paths)
  - Demande un changement au niveau de notre architecture OpenShift pour optimiser le hardware

Baremetal – aller plus loin

- K8s / Openshift – MachineSet pour autoscaler le hardware
  - https://medium.com/@wintonjkt/machinesets-and-auto-scaling-openshift-cluster-a24c458a200a
  - Mais : le hardware ne démarre pas dans la seconde – encore réservé à de grosses infrastructures
- Néanmoins, optimisation possible en diversifiant l'utilisation
  - Applications utilisées la journée
  - Batchs utilisés surtout la nuit et jours fériés
- Solution - Passage de la charge "batch" sur Openshift
  - Totalement supporté par la solution StoneBranch
  - Lissage de la charge sur la journée
  - Amélioration de la sécurité et la scalabilité au niveau des batchs.

Conclusion

- Une vision d'efficience énergétique sur les services déployés permet :
  - Des économies d'énergie
  - Des économies de licences
  - Des économies "hardware"
- Néanmoins
  - Beaucoup de solutions existantes ne sont pas adaptées au modèle Knative
  - C'est une responsabilité de tous – aussi bien des administrations que des fournisseurs. Il faut commencer à penser "cloud natif" dès le début du développement.
  - Les coûts induits par des applications non cloud natives sont maintenant calculables – licences / consommation énergétique – ceci doit faire partie des métriques
- On ne peut continuer à parler d'administration numérique sans réfléchir à son efficience énergétique et économique.
- L'efficience énergétique n'est pas du green washing – c'est un levier économique.

Futur

- Mise en place d'une infrastructure Baremetal (projet démarrant cette année) avec ServiceMesh / Knative
- Migration en continu de nos applications JEE / Springboot vers Quarkus
- Passage de notre solution de gestion de batch Stonebranch sur Openshift.