Interactions Matlab / RadosGW pour l'analyse de données d'imagerie mécanique

David Grimbichler
david.grimbichler@uca.fr

Université Clermont Auvergne, Mésocentre

Mercredi 28 novembre 2018

Sommaire

UNIVERSITÉ Clermont Auvergne

Sommaire

Mésocentre Clermont Auvergne

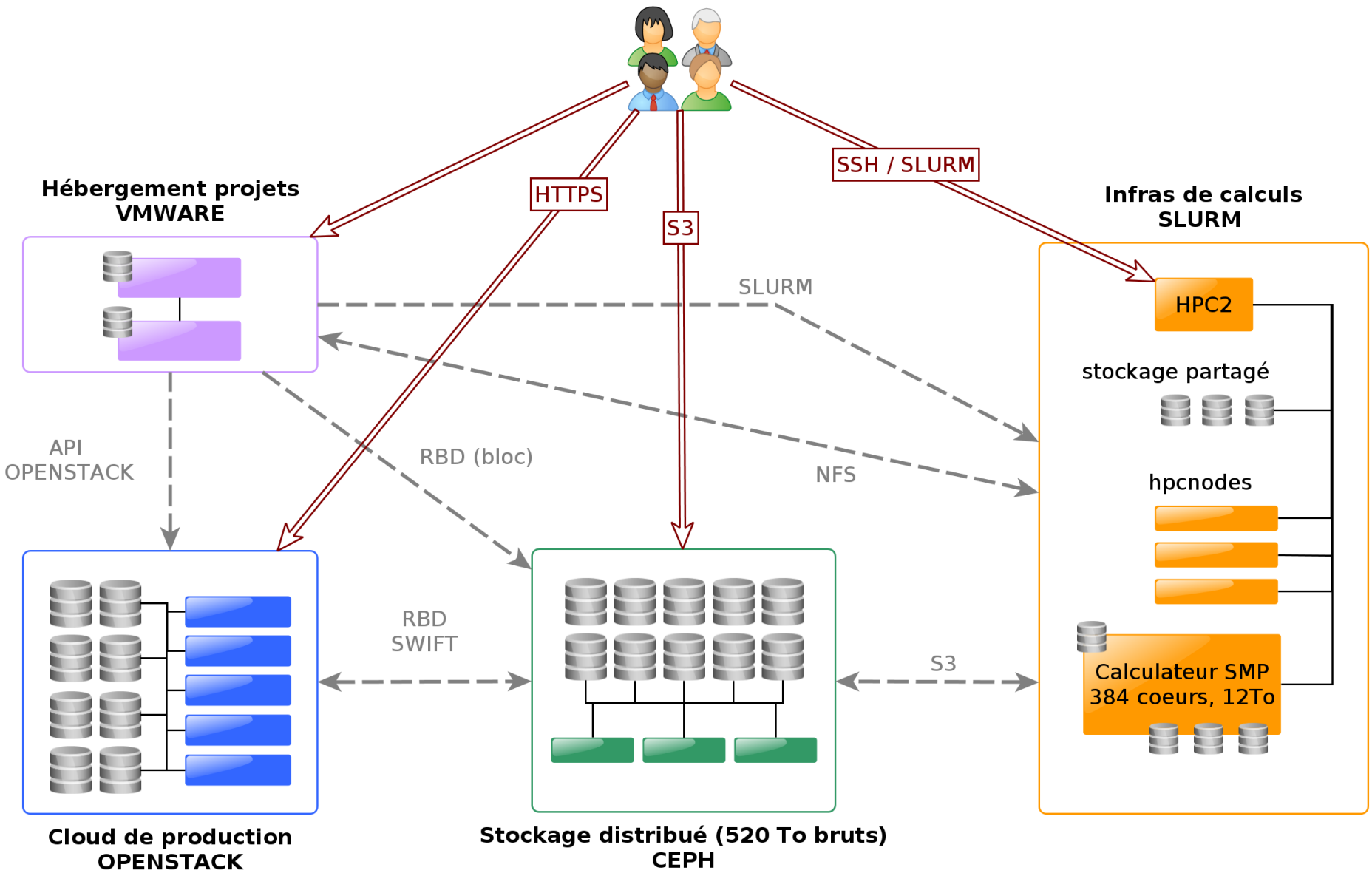
Mésocentre Clermont Auvergne :

- Université Clermont Auvergne
- $\approx$ 35000 étudiants, $\approx$ 35 laboratoires
- Mésocentre : service qui mutualise des ressources pour les différents laboratoires académiques
- incluant les laboratoires de l'université et des établissements partenaires

`https://mesocentre.uca.fr`

UNIVERSITÉ Clermont Auvergne

Infrastructures Mésocentre

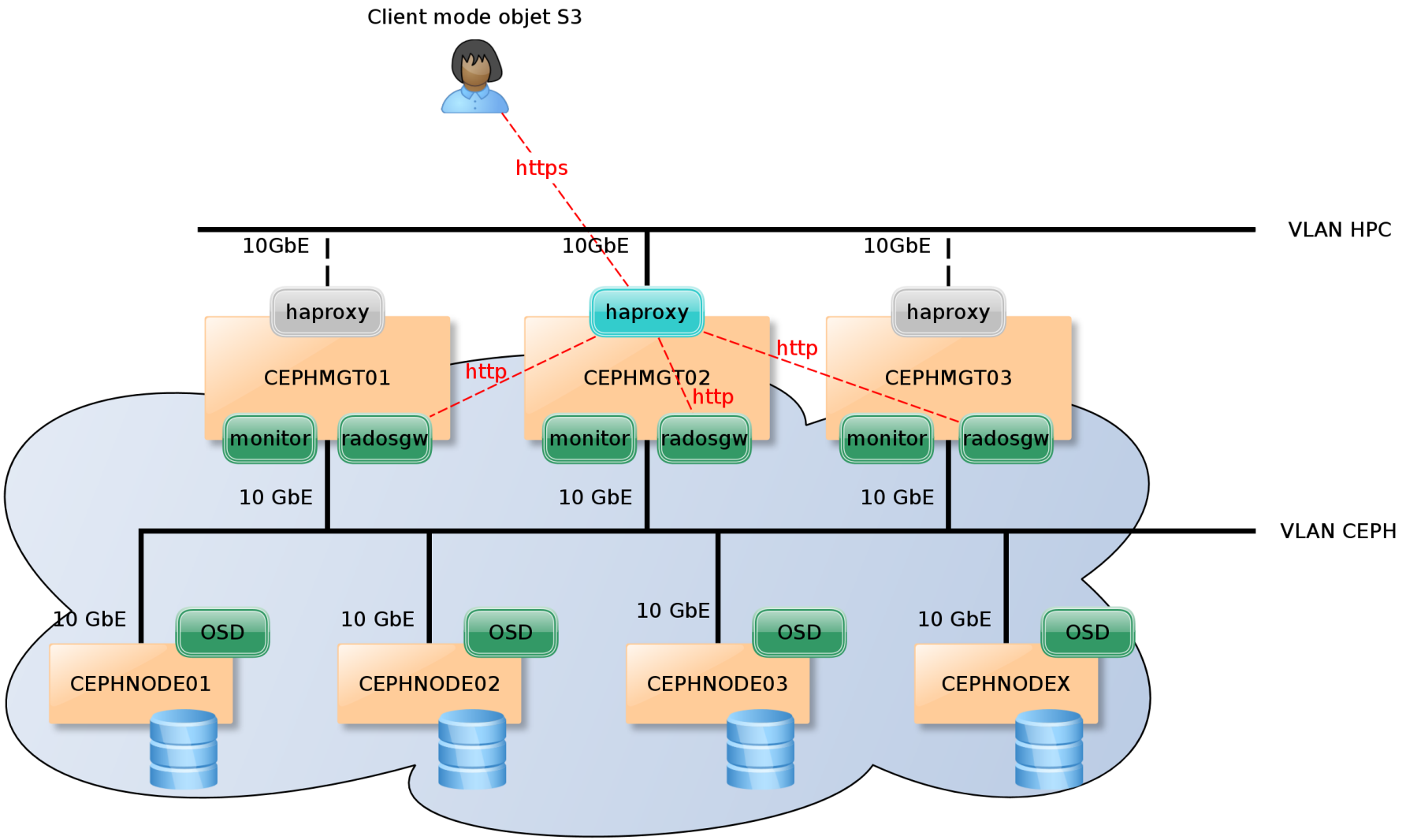


Infrastructure actuelle pour la recherche

Stockage : un cluster Ceph de 520 To (bruts)

- 3 contrôleurs redondants (monitors + radosgw + HAProxy - VRRP)
- 13 noeuds de stockage de 40 To
- 10 OSD de 4To + 2 SSD 200Go (journaux) par noeud, filestore
- interconnexion 10G
- 3 réplicats
- Ceph Jewel en production depuis début 2016 - tests préalables en Ceph Hammer en 2015 (pas de CephFS ☹)

Infrastructure Ceph Mésocentre



Ceph pour la recherche

Usages de notre cluster Ceph pour la recherche :

- Mode bloc RBD :
  - disque sur machines virtuelles (VMWare ESX)
  - volume Linux monté sur machines (virtuelles ou non) avec gros besoins de stockage
- Pools spécifiques pour un Seafile (Dropbox-like) pour les chercheurs
- Mode objet (S3) pour les grands volumes de données de recherche (plusieurs To)
- Usage majoritaire : mode objet via outils existants (s3cmd + s3fs pour les jobs de calcul, Cyberduck / Dragondisk pour la gestion, ...)
- Utilisation de l'API S3 directement dans des applications par les chercheurs : ***néant***

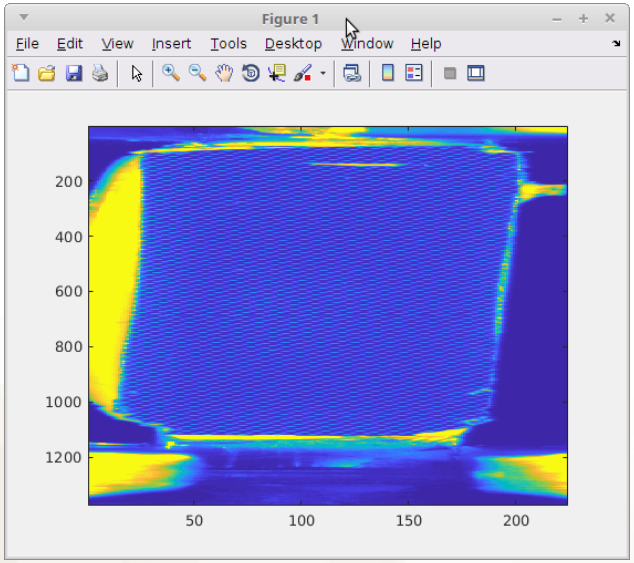
Sommaire

Situation

Problème des chercheurs (début 2017) :

- Beaucoup de fichiers : 20 millions
- Images TIFF (d'un format particulier) : vidéo haute fréquence d'imagerie mécanique
- Traitements avec Matlab (calcul de déformations) sur leurs postes
- 30 To de données :
  - Non sauvegardées
  - sur 6 disques externes USB3 de 6 To chacun
  - en FAT32
  - Certains jeux de données coupés sur deux disques
  - Accès aux données sur disque : lent (liste du contenu des répertoires)

Démarche



Pourquoi les chercheurs s'adressent à nous :

- expérimentations non reproductibles : sécurisation des données
- stockage secondaire, réplication
- équipe de plusieurs personnes : impossible de travailler à plusieurs avec les disques
- à terme : calculs sur les données via le cluster hpc

Upload

Upload des données en S3 (radosgw) avec s3cmd :

- via une machine reliée en 1G, disques en USB 3
- temps prévu : moins de 4 jours
- Au départ : upload moyen à 110Mo/s
- Débit moyen $\searrow$ + Temps d'accès $\nearrow$ avec nombre d'objets
- Avec s4cmd (multithread) : mieux, mais idem temps d'accès $\nearrow$

Sharding

⇒ Découverte du **sharding** ☺ :

- re-sharding de l'index du bucket (bug quotas)

```
radosgw-admin bucket reshard ...
```

```
radosgw-admin metadata get bucket.instance:mybucket:12.7 | grep num_shards
"num_shards": 20,
```

- changement configuration Ceph pour le sharding par défaut
- reprise de l'envoi avec s4cmd : ça va beaucoup mieux !

Sommaire

Accès aux données

Objectifs :

- fournir l'accès aux données pour les chercheurs de l'équipe
- ne pas changer leur façon de travailler
- sous Windows, Linux, MacOS
- il faut que ce soit plus facile et plus pratique qu'avec les disques !
- leur permettre de gérer les données et les droits : Cyberduck, Dragondisk (moins s3cmd)
- $\Rightarrow$ donc avoir accès aux données depuis Matlab

Matlab

Matlab doit accéder aux données de façon transparente.

Idée 1 : s3fs pour Linux/MacOS

- ne fonctionne pas sous Windows
- beaucoup d'objets dans le bucket $\Rightarrow$ déconnexions sporadiques
- temps d'accès et de listage très longs

Idée 2 : un script Matlab qui encapsule l'utilisation de s3cmd

- s3cmd ls : récupère plein de données annexes (autres que nom)
- s3cmd get + lecture Matlab
- Problème : très lent à cause du nombre d'objets

Matlab

Idée 3 : Matlab dispose d'un module d'accès S3 (FileDataStore)

- uniquement à partir de la version 2017 (pas de license pour cette version)
- impossible de spécifier le endpoint (Amazon AWS uniquement)

Idée 4 : développement d'un module Matlab (mex) compilable sous Windows, Linux, MacOS. Ajout de fonctions permettant de :

- lister le contenu d'un bucket
- récupérer un objet sur disque
- récupérer un objet *directement en mémoire Matlab* (en matrice)
  $\Rightarrow$ possibilité d'utiliser les fonctions de lecture d'image de Matlab

Matlab S3 mex

Pour le développement de l'**idée 4** :

- utilisation de la bibliothèque Ceph libs3 : `S3_get_object, S3_list_bucket, ...`
- Linux, MacOS : pas de difficulté particulière
- Windows : qqs modifications pour compilation correcte avec MinGW
- Windows : compilation préalable des bibliothèques dont dépend libs3 (libcurl, libiconv, libssl, ...)

Résultat : création d'un module MEX pour Matlab.

`http://forge.clermont-universite.fr/projects/s3-matlab-mex-module`

Utilisation

Au final, quasiment aucun changement dans le code des chercheurs :

```
s3Init();

S3_SRVR = 's3.mesocentre.uca.fr';
S3_AKEY = 'MYACCESSKEY';
S3_SKEY = 'MYSECRETKEY';
S3_BCKT = 'mybucket';

[L, E] = s3List(S3_SRVR, S3_AKEY, S3_SKEY, S3_BCKT,
                'path/to/objects/');

for k = 1:size(L,1)
  disp(L{k})
  [data, err] = s3Get(S3_SRVR, S3_AKEY, S3_SKEY, S3_BCKT, L{k});
  ...
end

s3Cleanup();
```

Sommaire

Performances comparatives

Avec s3cmd :

```
$ time s3cmd ls s3://bucket/data01/Img_1100001_to_1200000/
  [...]
real 3m9.400s
user 2m55.891s
sys  0m3.020s
```

Avec s3fs :

```
$ mkdir /tmp/s3fs
$ s3fs bucket /tmp/s3fs -o url=https://s3.mesocentre.uca.fr,uid=$UID
$ time ls /tmp/s3fs/data01/Img_1100001_to_1200000/
[...]
real 15m17.940s
user 0m3.359s
sys  0m5.392s
```

Avec s3fs, souvent erreurs "transport endpoint is not connected" après quelques dizaines de minutes

Performances comparatives

Avec module Matlab :

```
>> s3Init();
>> tic;
   [L, E] = s3List(S3_SRVR, S3_AKEY, S3_SKEY, S3_BCKT,
                   'data01/Img_1100001_to_1200000/');
   toc

Elapsed time is 4.537611 seconds.

>> E
E =
     'OK'

>> size(L,1)
ans =
       100000
```

Sommaire

Conclusion

Les chercheurs sont contents :

- plus facile pour eux de travailler
- données sécurisées
- accès aux données de n'importe où
- très peu de changement dans leur code Matlab
- impact du temps de transfert des images invisible devant le temps de traitement de chaque image

Le Mésocentre est content :

- Acquisition d'expérience sur Ceph
- Meilleure compréhension de la structure des objets rados

Perspectives

Mésocentre :

- Nouveau Ceph Mimic mis en place (800 To)
- Bluestore
- avec un OpenStack
- avec CephFS
- Fusion du Ceph existant dans cette nouvelle infrastructure

Merci !

Merci de votre attention ☺

Sommaire

Mésocentre Clermont Auvergne

Missions du Mésocentre :

- Mutualiser des ressources de calcul et de stockage à l'échelle du site
- Accompager les utilisateurs et les communautés des utilsateurs
- Développer une expertise autour du calcul scientifique (calcul intensif, calcul GPU, cloud computing) et du stockage
- Favoriser les échanges et les collaborations entre équipes de recherche
- Collaborer avec les mésocentres en région (initiative *CIDRA*^{2}) et à l'échelle nationale (GENCI, Groupe Calcul)

Calcul

Infrastructure actuelle de calcul pour la recherche :

- Cluster de calcul avec SLURM
- 40 noeuds, 720 coeurs physiques, 6 To de RAM (17.8 TFlops)
- + 1 noeud avec 2 GPU (2 Tesla P100 16G), d'autres noeuds GPU en cours d'acquisition
- + 1 calculateur à mémoire partagée (SMP) :
  - fullmesh
  - 384 coeurs physiques - 16 × Xeon E7-8890 v4 (13.5 TFlops)
  - 12 To de RAM

Performances comparatives

Avec goofys :

```
$ mkdir /tmp/goofys
$ goofys -o ro --stat-cache-ttl 3600s --type-cache-ttl 3600s
      -uid $UID --dir-mode=0777 --file-mode=0777 --cheap
      --endpoint https://s3.mesocentre.uca.fr mybucket /tmp/goofys
$ time ls /tmp/goofys/data01/Img_1100001_to_1200000/
[...]
real 0m58.855s
user 0m3.122s
sys  0m6.579s
```

Avec goofys, aucun problème de déconnexion.