CEPH

Yann Dupont

CCIPL / DSIN Université de Nantes

www.univ-nantes.fr

14 Décembre 2016

UNIVERSITÉ DE NANTES

Ceph ?

Système de stockage complet et versatile,
utilisable de nombreuses façons.

Architecturé sur un cluster de serveurs standards.
Fournit nativement des interfaces objet, système de fichiers et bloc.

Très utilisé dans le monde open-source : couvre de nombreux cas d'usage.

Historique des versions

inktank

47 Préversions (!)

07 / 2012 Argonaut (v 0.48)

01 / 2013 Bobtail (v 0.56)

05 / 2013 Cuttlefish (v 0.61)

08 / 2013 Dumpling (v0.72) LTS

11 / 2013 Emperor (v 0.67)

Rachat

Majeure

05 / 2014 Firefly (v0.80) LTS

10 / 2014 Giant (v0.87)

04 / 2015 Hammer (v0.94) LTS

renumérotation

11 / 2015 Infernalis (v9.2.z)

04 / 2016 Jewel (v10.2.z) LTS

Base de RedHat CEPH storage v2

?? / 2017 Kraken (v11.2.z)

?? / 2017 Luminous (v12.2.z) LTS

Opensource
https://github.com/ceph

Tous les 6 mois
LTS tous les ans

Numéros de versions

Version X.Y.Z

X pair = version à long support (LTS)

Y=0 development / preview
Y=1 beta
Y=2 stable

Z = version mineure

Version actuelle :

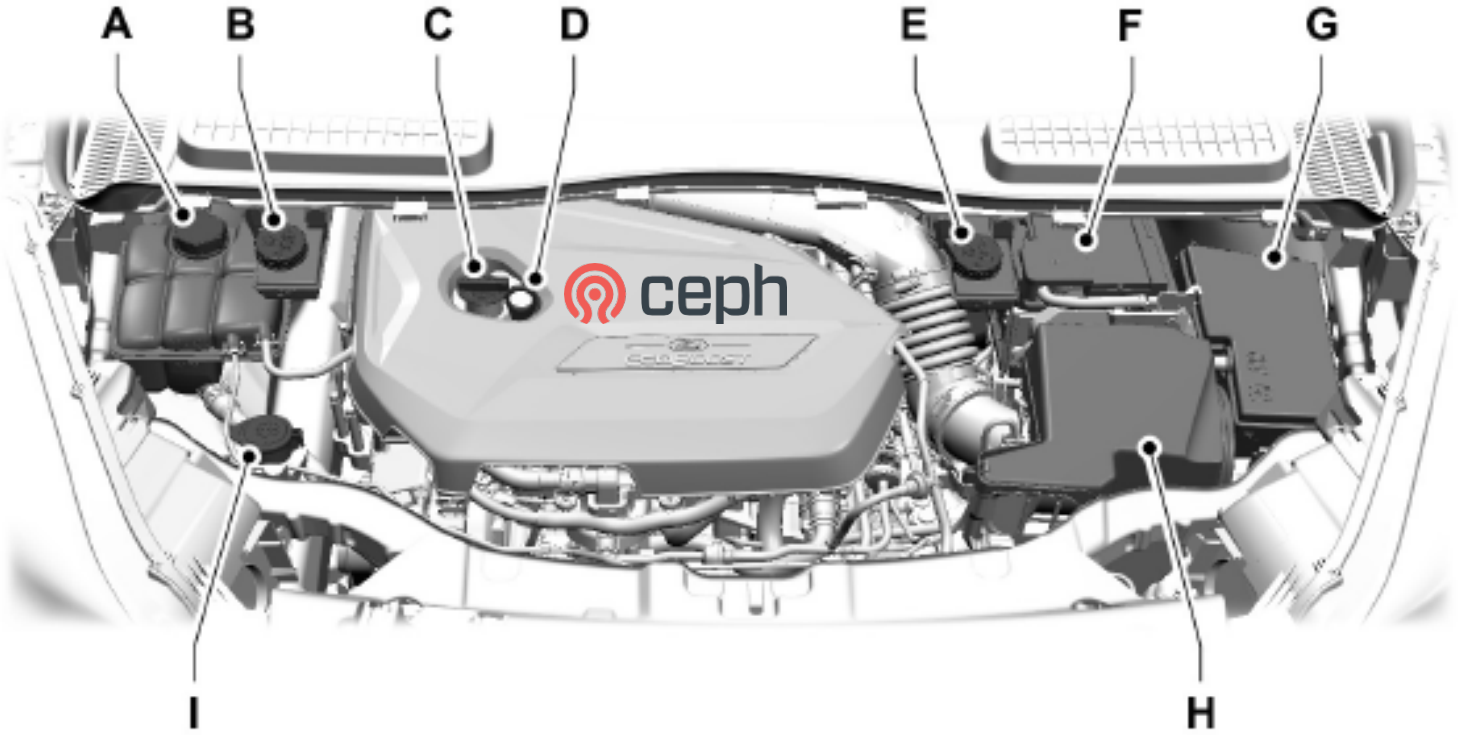
v10.2.5 (sortie le 10 dec 2016)
(Nom de code Jewel)

Support : (open source)
Jewel 04/2016 ⟶ 11/2017

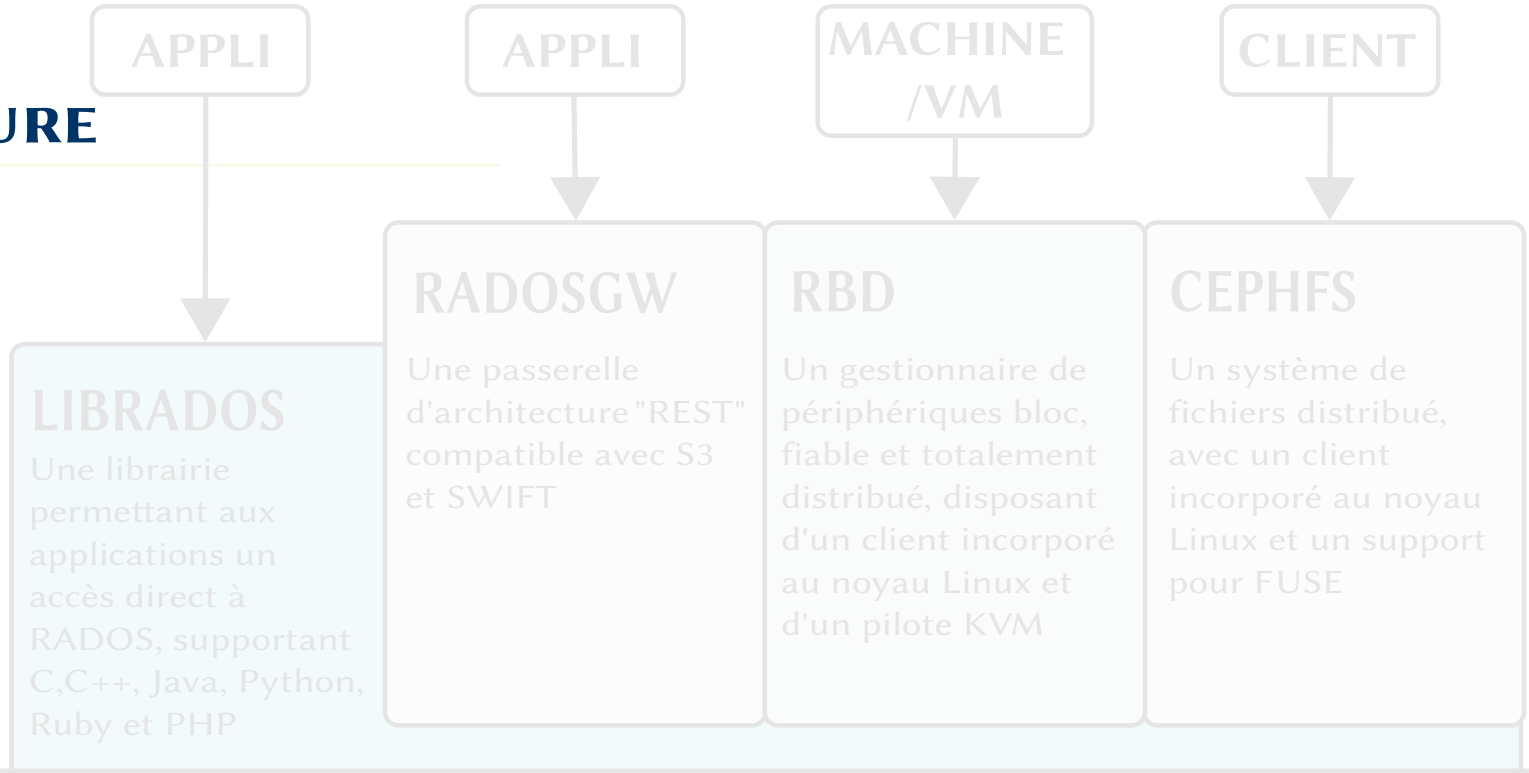
Linux Seulement
(développements BSD en cours)

v11.1.0 (beta) : 13 dec 2016

Sous le capot ...



Architecture



1

RADOS

Un ensemble d'objets distribués, fiable et autonome, géré par des nœuds de stockage capables d'auto gestion et auto réparation

Architecture

APPLI

2

LIBRADOS

Une librairie permettant aux applications un accès direct à RADOS, supportant C,C++, Java, Python, Ruby et PHP

APPLI

MACHINE /VM

CLIENT

RADOSGW

Une passerelle d'architecture "REST" compatible avec S3 et SWIFT

RBD

Un gestionnaire de périphériques bloc, fiable et totalement distribué, disposant d'un client incorporé au noyau Linux et d'un pilote KVM

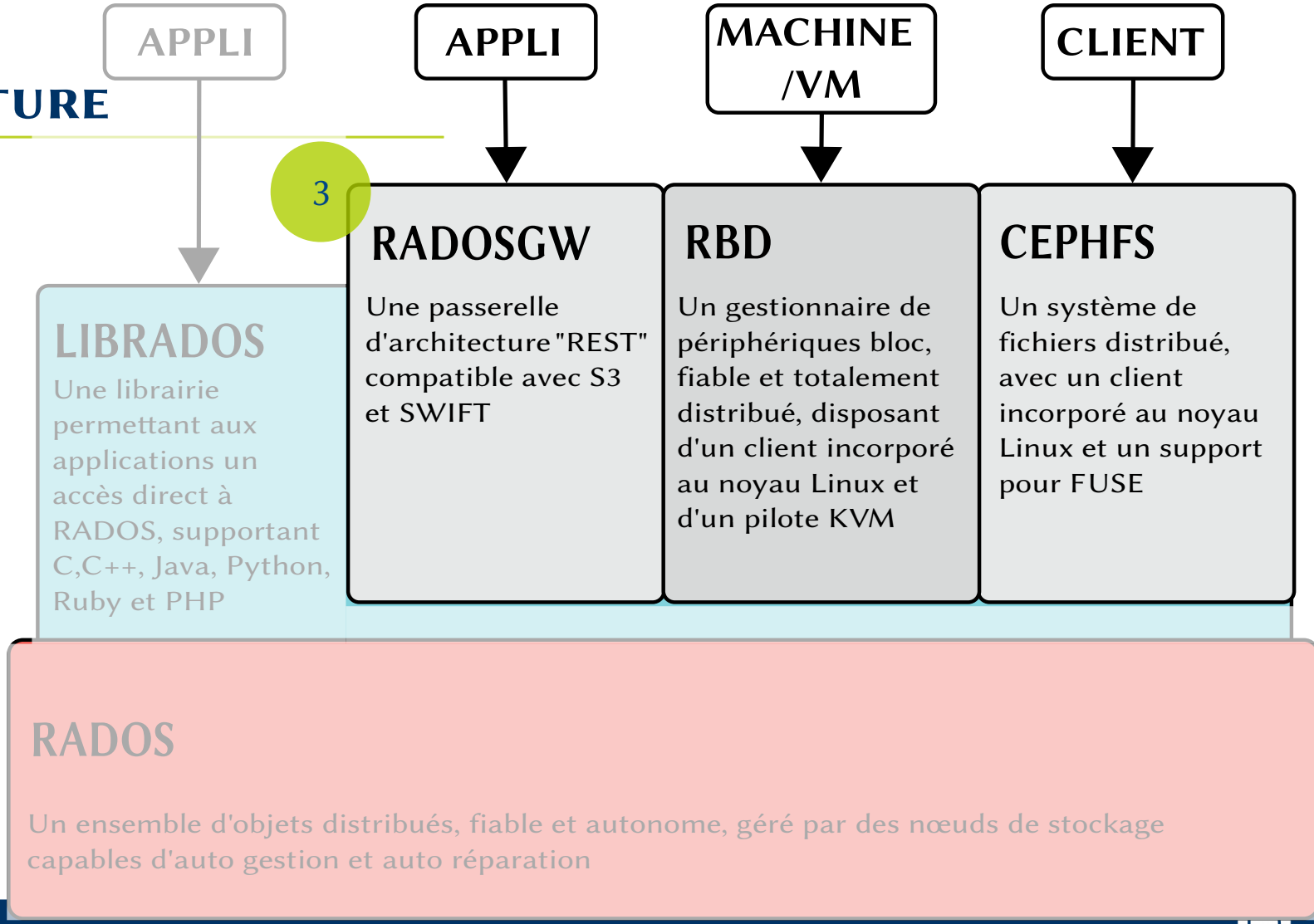
CEPHFS

Un système de fichiers distribué, avec un client incorporé au noyau Linux et un support pour FUSE

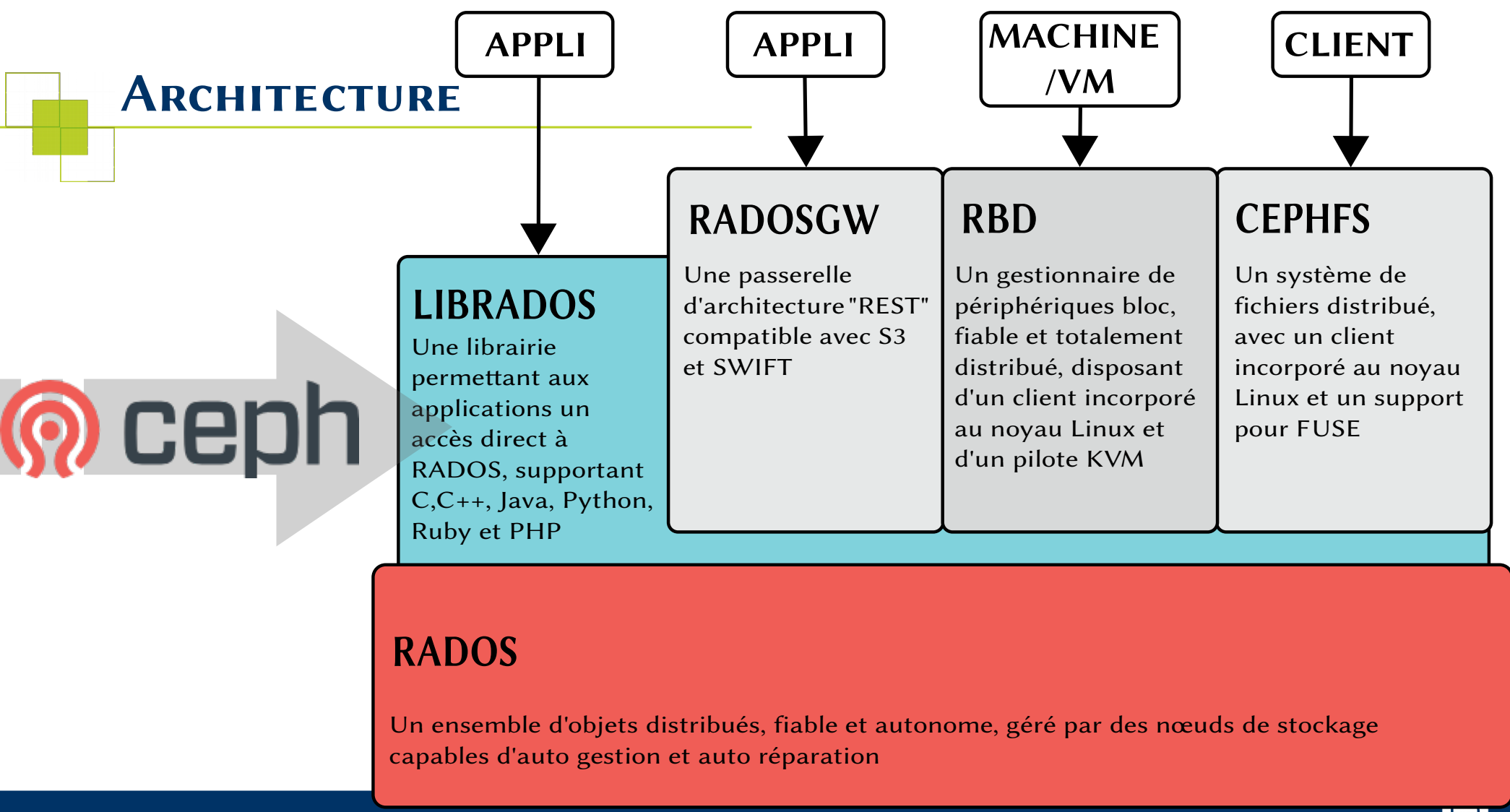
RADOS

Un ensemble d'objets distribués, fiable et autonome, géré par des nœuds de stockage capables d'auto gestion et auto réparation

Architecture



Architecture

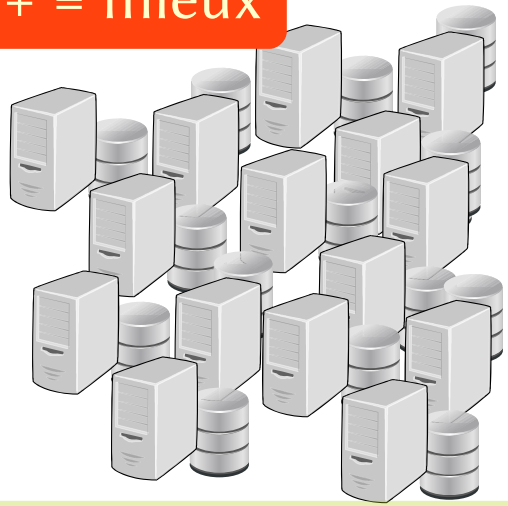


Architecture de Ceph

Nombre impair

Nombre impair

+ = mieux



MON
Monitor

Vérifie le bon état du cluster
Assure la communication initiale avec les clients
Vérifie les droits d'accès
Machine (VM) dédiée conseillée.

MDS
Meta Data Server

Gère les méta données
OPTIONNEL
(uniquement pour CephFS)

OSD
Object Storage Daemon

Stocke les objets
sur filesystem local (XFS)
Communique avec les clients
Débit disque et réseau

Architecture de Ceph

pas de SPOF
scalable

MON

Pas de SPOF
Passif/Actif :
pas (encore) scalable

MDS

Pas de SPOF
scalable

OSD

Démons en espace utilisateur Linux

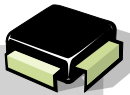
Dimensionner son architecture

Réplication synchrone
(la partie la plus lente ralentit le reste)

Débit != Latence
Performance : tous les éléments comptent

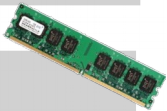
Beaucoup d'OSD
Être vigilant sur la configuration

Réseau :
Bonne carte et bon driver (éviter bnx2x :)
Iperf3 entre les nœuds pour vérifier le débit
Bon switch pour baisser les latences
Mlx5 ?? Rdma pas encore supporté



>=1 thread / OSD
cœurs rapides : utiles pour Erasure coding
Utiles pour OSD sur SSD, (cpu facteur limitant)

Utiliser des contrôleurs non bloquants
1 SSD sur 4 pour les OSD (journaux)
OSD sur 7,2k, 15k ou SSD : dépend du profil
PAS DE RAID



Utilisé pour les processus OSD (512 Mo ⟶ 4 Go)
Et cache I/O par les kernel Linux

Machine typique OSD

1 OSD = 1 Disk
plus de RAID Hardware
Des SSD SLC partagés (Journaux)
12 OSD / machines
Capacité Brute : ~ 100 To

Stocker un objet

Ceci est un objet
(utile)

Stocker un objet dans RADOS

Où ?
Comment ?

Un cluster CEPH présente des pools de stockage.
Ils sont créés au besoin. (1 de base).

Chaque pool dispose de ses caractéristiques,
ses droits d'accès,
sa règle de placement d'objets (CRUSH).

Stockage dans
pool objets_utiles

Pools

```
GLOBAL:
SIZE       AVAIL      RAW USED     %RAW USED
43483G     16986G       26496G         60.94
POOLS:
NAME                ID     USED       %USED     MAX AVAIL     OBJECTS
mirrors             3       3341G      7.68         7646G      863776
cloud-perso         4       1296G      2.98         5097G      465044
os-patrons          6        209G      0.48         5097G       67193
os-dsin-prv         7        907G      2.09         5097G      262322
logs                9       2583G      5.94         5097G      666408
data-dsin           11       432G      1.00         5097G      111371
data-tiers          12       180G      0.41         5097G       46168
objets-utiles       13     21234M      0.05         5097G        5379
esxi-backup         14      8973M      0.02         5097G        2916
```

CRUSH

- Où et comment placer les objets dans le pool ?
- Le faire simplement et rapidement ?
- Qui fait le choix ?

Règles CRUSH

Description de la hiérarchie du matériel

```
root default
     datacenter lmb
          room lombarderie-ltp
               host abouriou
                    vm ceph-osd-lmb-C-1-1
                         osd.0     up   1
                    vm ceph-osd-lmb-C-1-2
                         osd.1     up   1
               host magon
                    vm ceph-osd-lmb-C-3-1
                         osd.8     up   1
                    vm ceph-osd-lmb-C-3-2
                         osd.9     up   1
     datacenter loi
          [...]
```

Règles de placement des objets

```
rule rbd {
        ruleset 2
        type replicated
        min_size 1
        max_size 10
        step take default
        step chooseleaf firstn 0 type datacenter
        step emit
}
```

Tout est stocké globalement dans le cluster
Tout le monde (y compris le client) dispose de la dernière version.

Groupes de placement

Un pool est découpé
en groupes de placement (PG).

1 PG contient la liste des
OSD contenant les copies
de l'objet.

Carte des PG change lors de la
modification de la topologie
(algorithme).

Le client PLACE ses objets.

```
ceph osd pool get objets-utiles pg_num
pg_num: 1024
```

Le pool objets-utiles dispose de 1024 PG

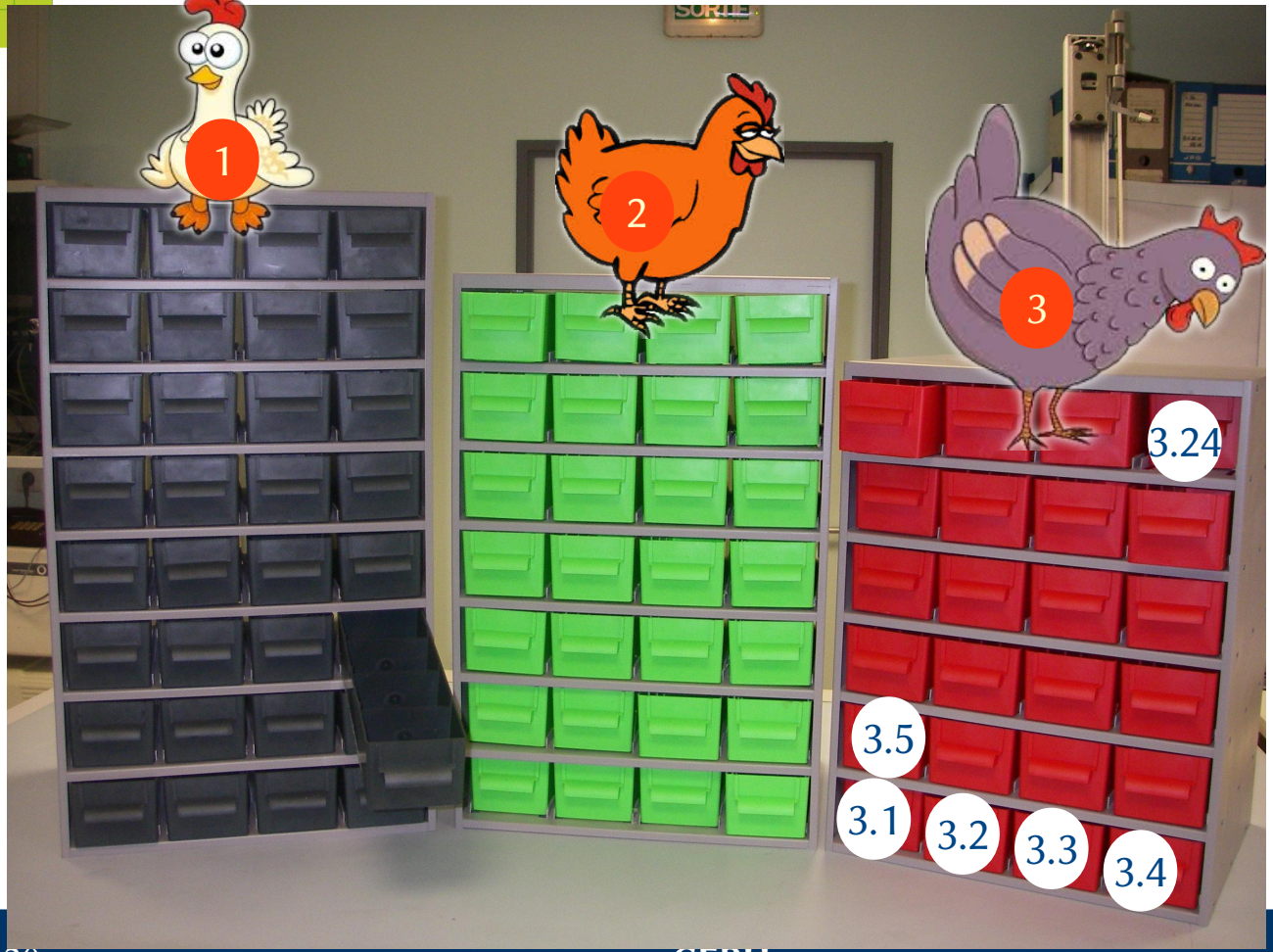
```
ceph osd pool get objets-utiles size
Size: 3
```

**Le pool os-patrons est configuré
pour dupliquer 3 fois l'objet**

`ceph pg dump` (13.3fb → [3,4,8])

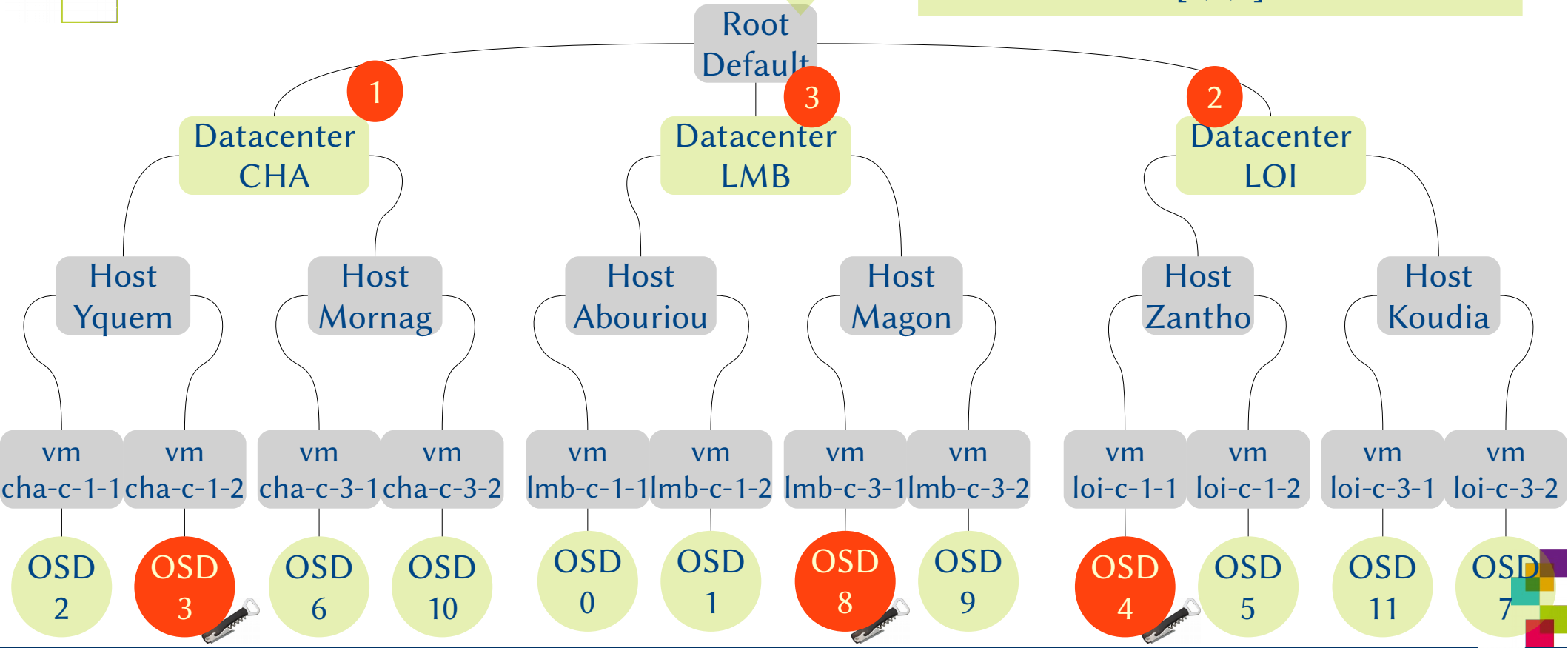
**Le PG 3fb du pool 13 (objets-utiles)
utilisera les serveurs de stockage (OSD)
3, 4 et 8**

Pools & PG



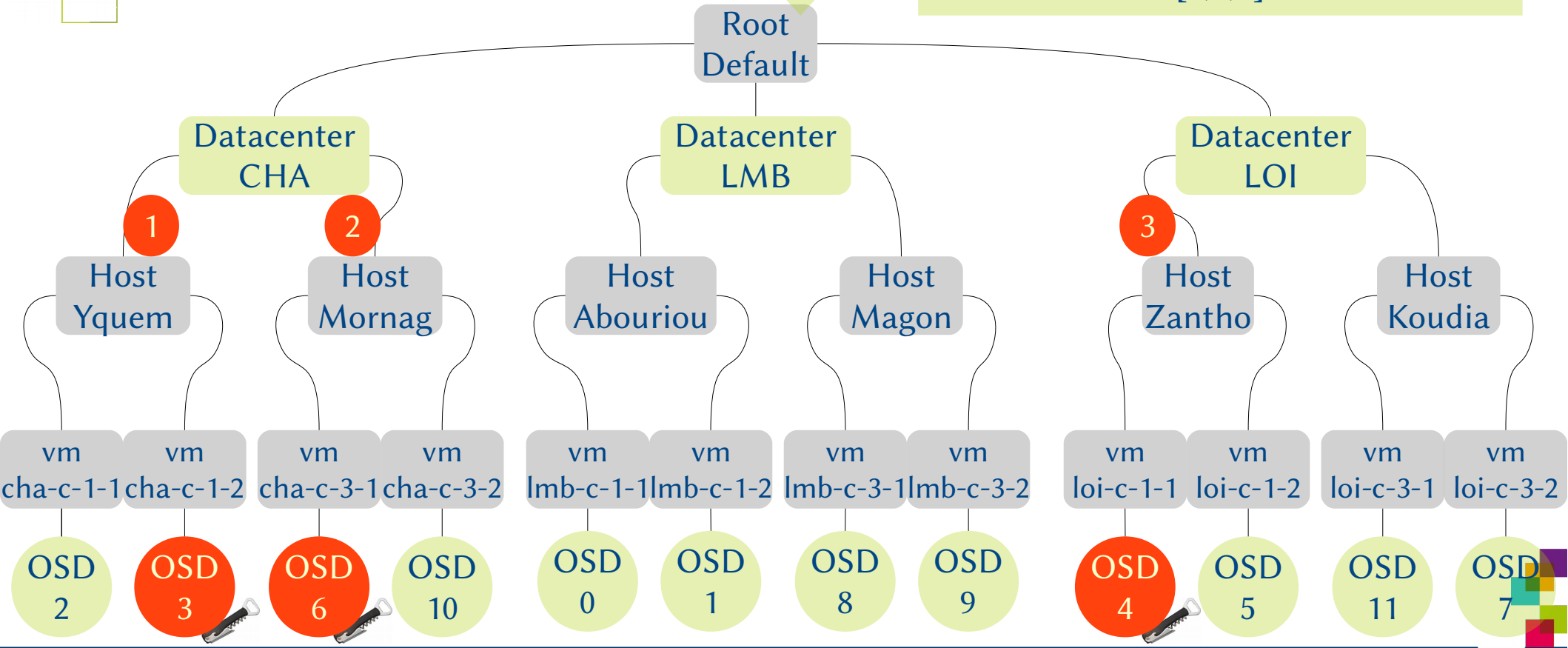
Règles CRUSH

Pool 13 : objets_utiles (***3 copies***)
PG dump 13.3fb
[3,4,8]



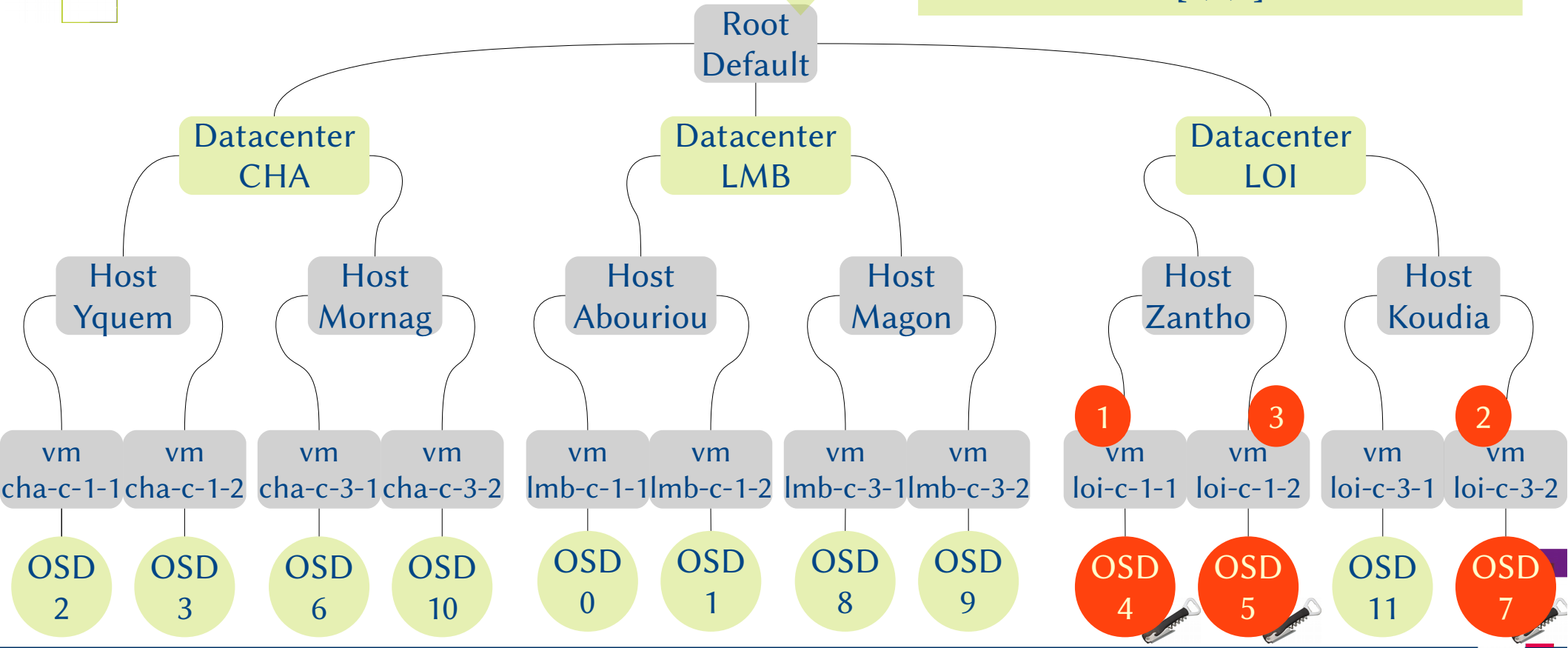
Règles CRUSH

Si le choix se fait sur host
PG dump 13.3fb
[3,6,4]



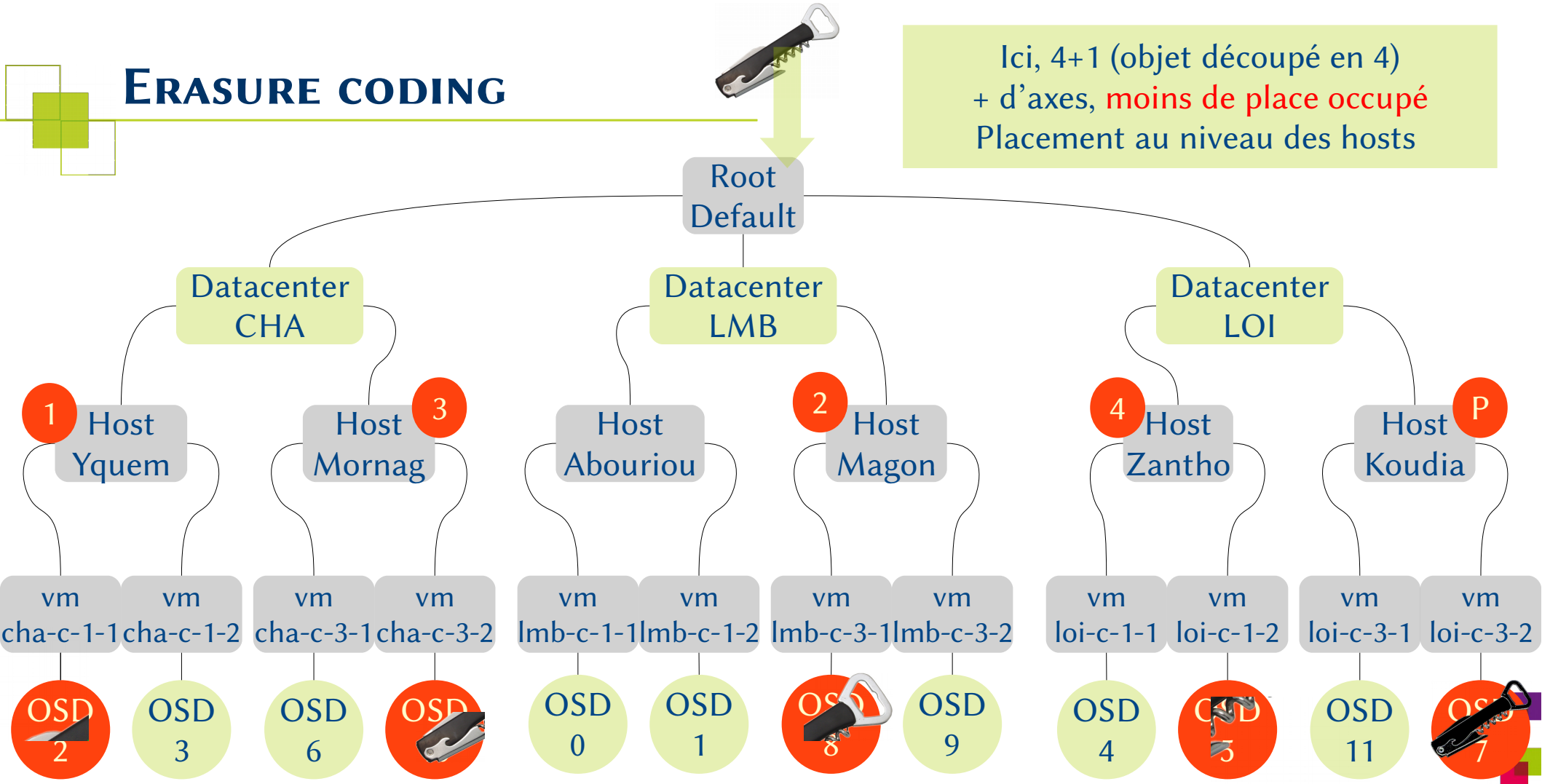
Règles CRUSH

Si le choix se fait sur vm
PG dump 13.3fb
[4,7,5]

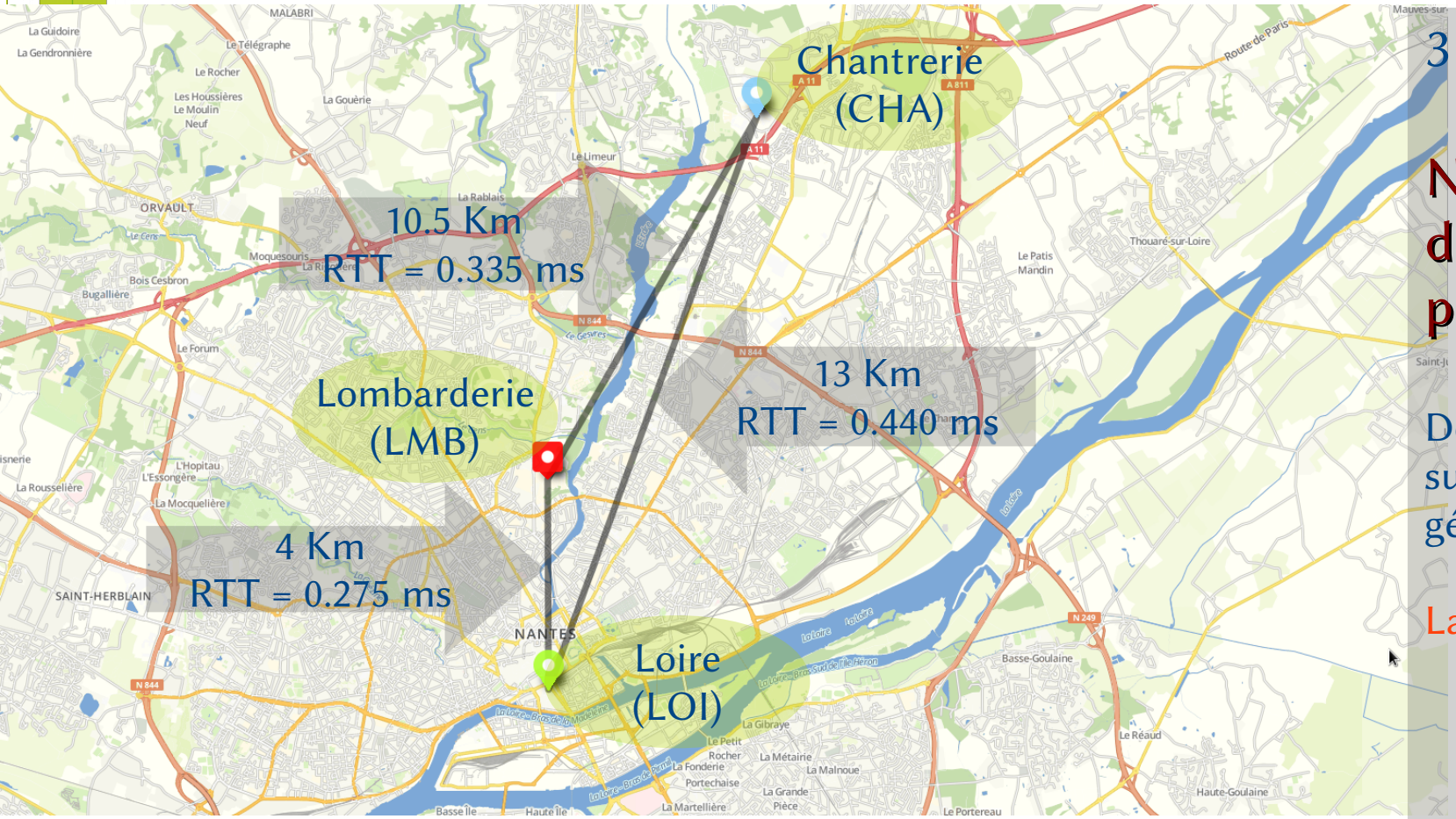


Erasure coding

Ici, 4+1 (objet découpé en 4)
+ d'axes, moins de place occupé
Placement au niveau des hosts



Sécurité des données.



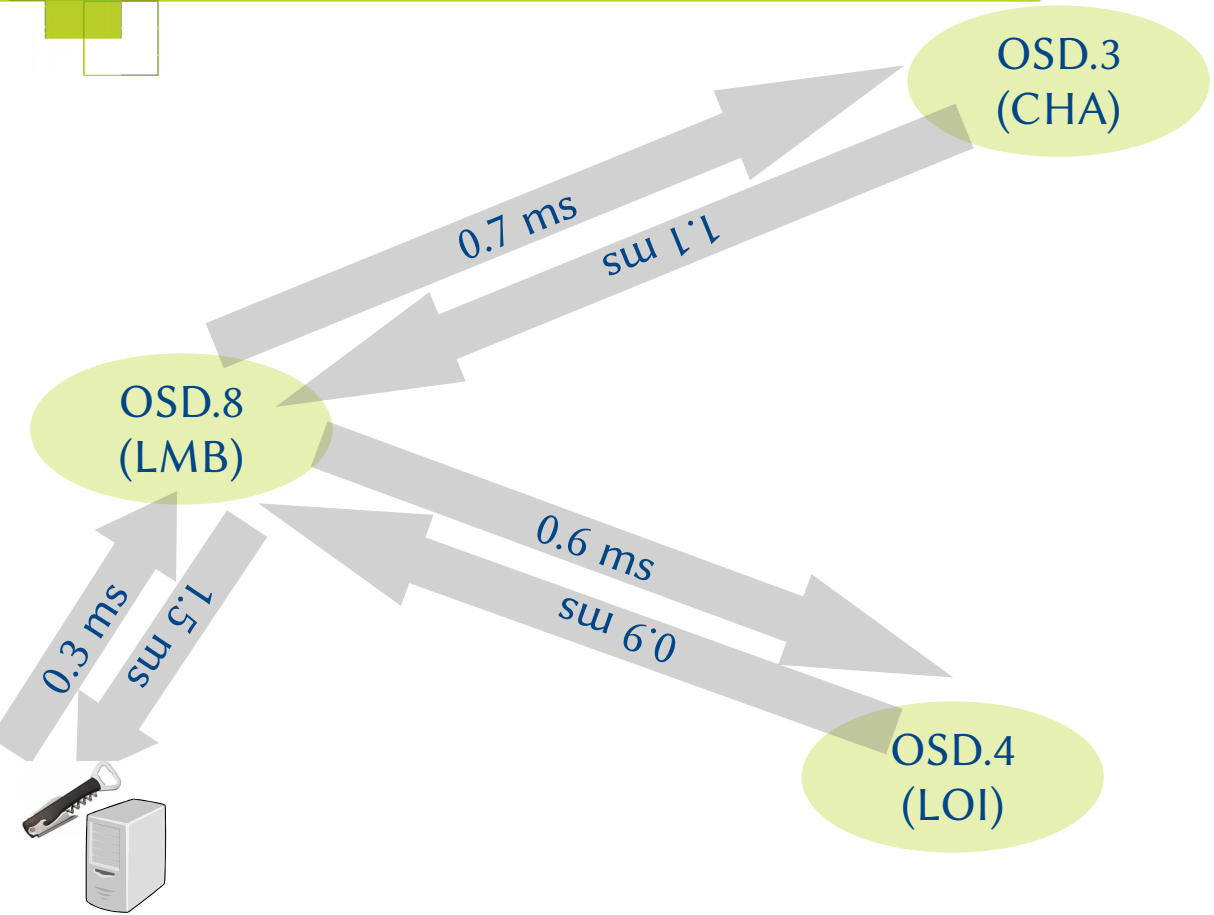
3 salles machines

Notion de domaine de panne

Données stockées sur 3 points géographiques

Latence vs Performance

Latence



>1,5 ms par transaction

~ 600 iops

Mais par thread.

Plus de threads = plus d'I/O mécaniquement

Récapitulatif

Cluster

Contient Des

Pool

Contrôle

Authentif.

Composé de

PG

Liste Des

OSD

Stockent Sur

Disque et FS local

Assignés à

Objets

contient des

Applique

Règles de Placement (CRUSH)

Conditionne le placement

Interface Objet (LibRados, RadosGW)

Nécessite Mon, OSD

Rados, librados = mode natif.

Objet stocké de façon monolithique ou découpé en blocs d'une certaine taille.

Utilitaire rados pour intéragir directement (cf TP)
Clients : outils cloud, applications WWW

RESTful (web) service: Swift, S3
implémenté via l'outil RadosGW (couche de translation utilisant un serveur WWW embarqué)
Une grande partie de S3 est implanté.

Interface bloc RBD (Rados Block Device)

Nécessite Mon, OSD

Émule des périphériques de type bloc. San Virtuel.
Gère les snapshots, copy on write, import, export, miroir asynchrone, thin provisionning

Performant, peut remplacer des baies SAN.

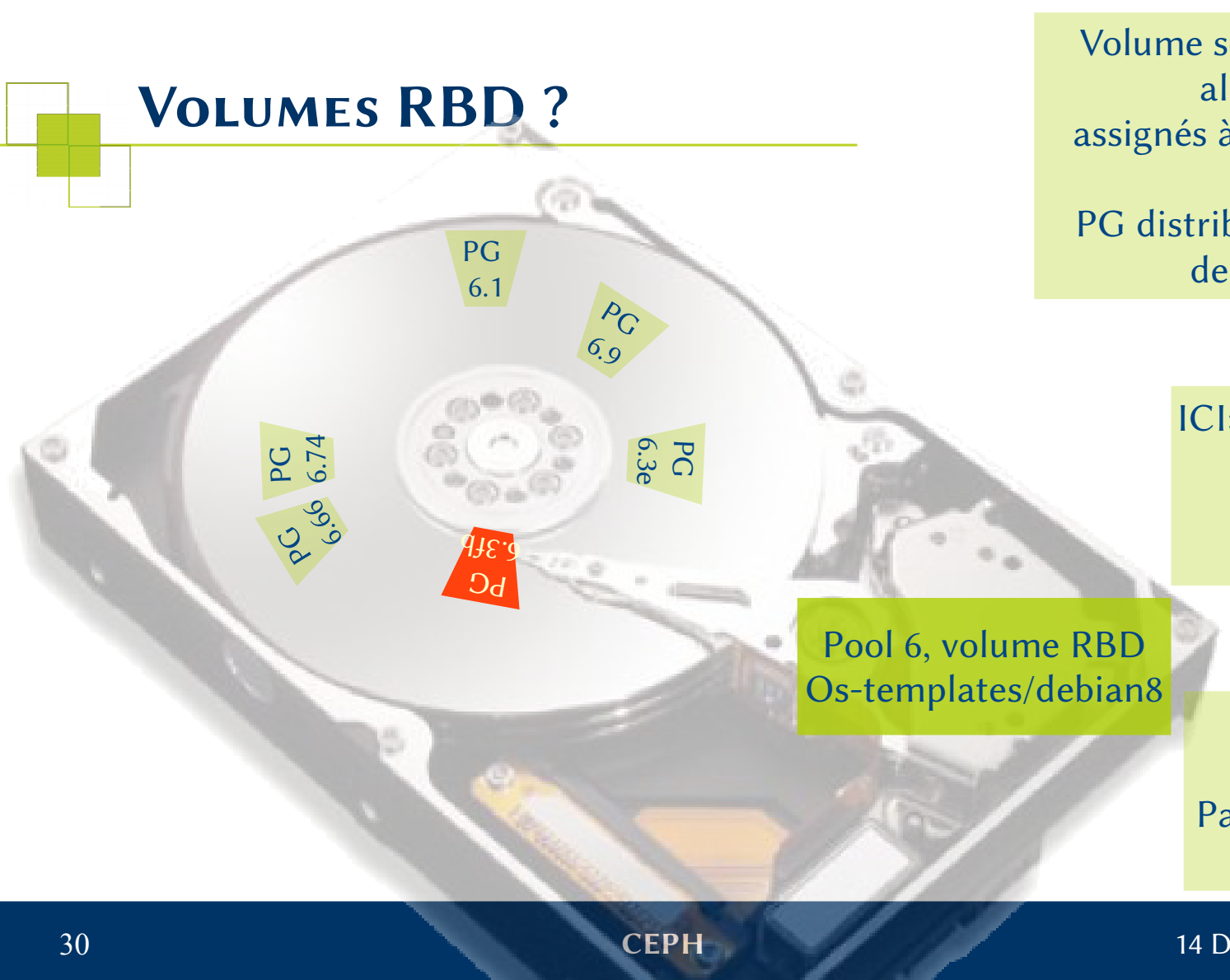
Clients :

Krbd,Fuse,Librbd.
Pas de client Windows ou Vmware.

Support librbd pour KVM → Windows peut quand même en bénéficier.
Support iSCSI via un gateway (ne faisant pas partie de CEPH) : il existe plusieurs solutions.

Comme n'importe quel périphérique bloc, peut aussi être réexporté par samba ou NFS.

Volumes RBD ?



Volume segmenté en blocs de 4 Mo,
alloués à la demande
assignés à un PG (Placement Group)

PG distribués selon règles (CRUSH)
de placement du pool

ICI: règles de placement simples

Root =datacenter
Size = 3

Pool 6, volume RBD
Os-templates/debian8

"map" pré-calculée,
Distribuée aux clients
Pas de serveur de metadonnées
Accès simple & direct

Volumes RBD ?

On continue à stocker des objets dans un pool.

Les objets = blocs de 4 Mo de données.

(1000 secteurs de type AF, 8000 de type standard)

Allocation de l'objet lors de sa 1ere écriture

Possible de créer autant de volumes que nécessaire

Attention à la sur-réservation !

Interface file system distribué: CephFS

Nécessite Mon, OSD, ***MDS***

Très longtemps considéré comme instable.
Stable depuis Jewel.
Outil de check/réparation disponible.

Mais pas forcément scalable : Support multi-mds déconseillé (mode actif passif : cf TP)

Et pas forcément performant : benchmarks ?

Fichiers découpés en blocs de 4 Mo (comme RBD).

Support Linux kernel ou fuse.

Côté serveur

Jewel : grosse amélioration SSD / Cache Tier
Noyau Linux 4.4 : Amélioration réseau, blkmq (SSD)
Noyau 4.9 à tester (sorti 11 dec 2016)
Paramétrer ceph.conf (enlever debugs, augmenter thread, prioriser les tâches)
Cartes réseau avec un bon driver (certaines ne saturent pas le 10 Gb/s)
Séparer réseau côté public et cluster
Configurer MTU à 9000 au moins côté cluster
Bonding : utiliser un mode actif/actif (lacp, xmit_hash_policy=layer3+4)

Systèmes de fichiers pas conçus pour gérer ce flux de données et méta données
BlueStore ⟶ Kraken (mode preview actuellement)

RBD : DISCARD, VIRTIO, VIRTIO-SCSI

Allocation à la volée de l'espace.

Il est possible d'allouer des volumes de 20 To sans les avoir physiquement.
Consommation à l'utilisation.

Pb : Un FS comme XFS va s'étaler sur tout l'espace.

Même soucis que les SSD, les baies SAN avancées : envoyer TRIM
pour récupérer espace.
(Pour les SSD MLC/TLC, important pour la performance)

VIRTIO standard ne le supporte pas.

RBD : Usage direct avec KVM (librbd, virtio-scsi)

```
<disk type='network' device='disk'>
    <driver name='qemu' type='raw' cache='writeback' discard='unmap'/>
    <auth username='nfsgw'>
      <secret type='ceph' uuid='260ee1cc-c10a-44c0-a708-6f466c8adb2b'/>
    </auth>
    <source protocol='rbd' name='NFS/IRTS'>
      <host name='172.20.106.86' port='6789'/>
      <host name='172.20.107.86' port='6789'/>
      <host name='172.20.108.86' port='6789'/>
    </source>
    <target dev='sdb' bus='scsi'/>
<controller type='scsi' index='0' model='virtio-scsi'>
</controller>
```

Support du trim/discard
Pour l'invité

Possibilité de fstrim
ou mount -o discard

Synchronisation de
l'allocation par le FS
Et l'allocation par la
couche bloc.

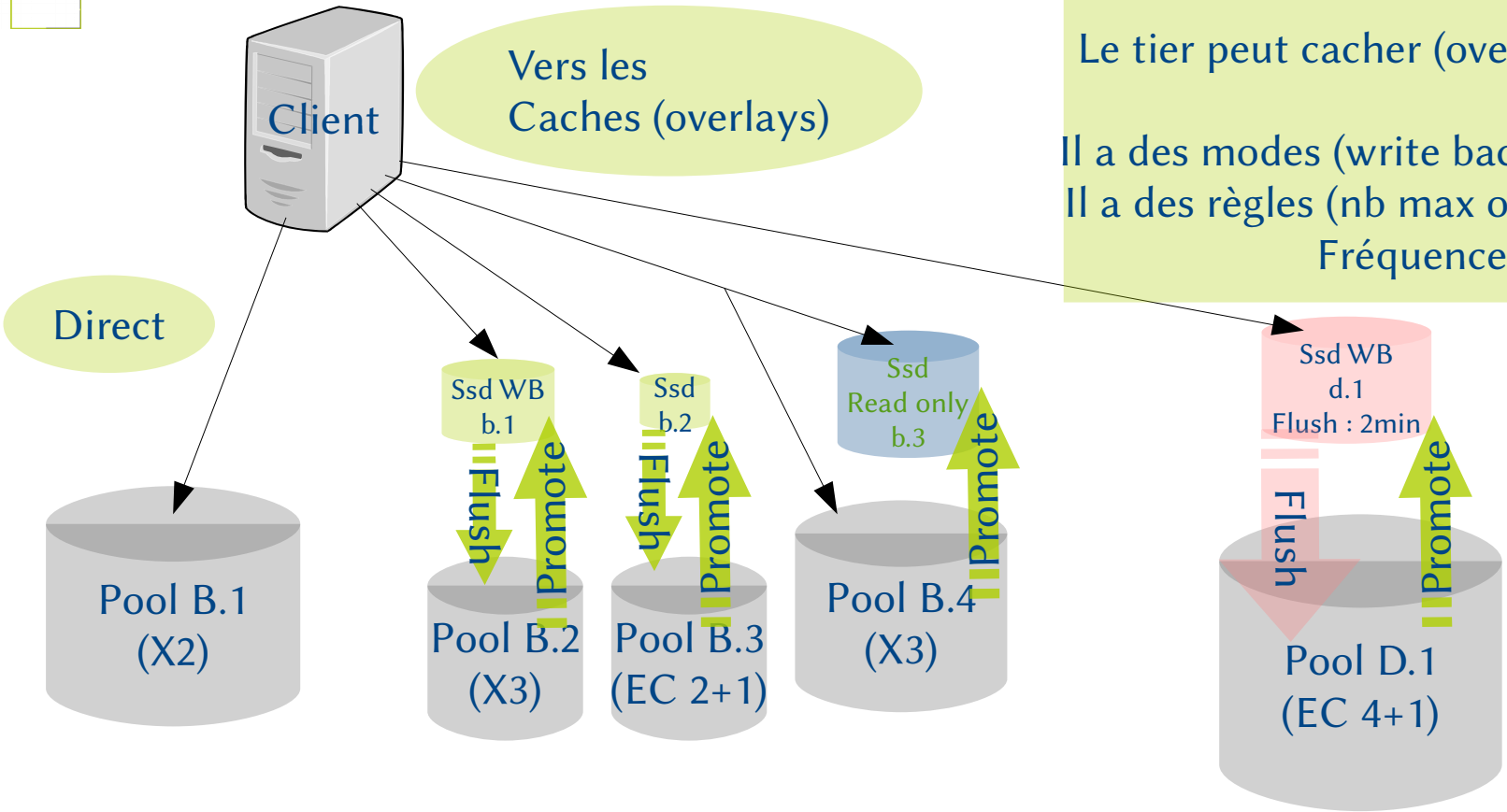
Virtio-scsi + lent que virtio
(10% ?)

Tier, cache

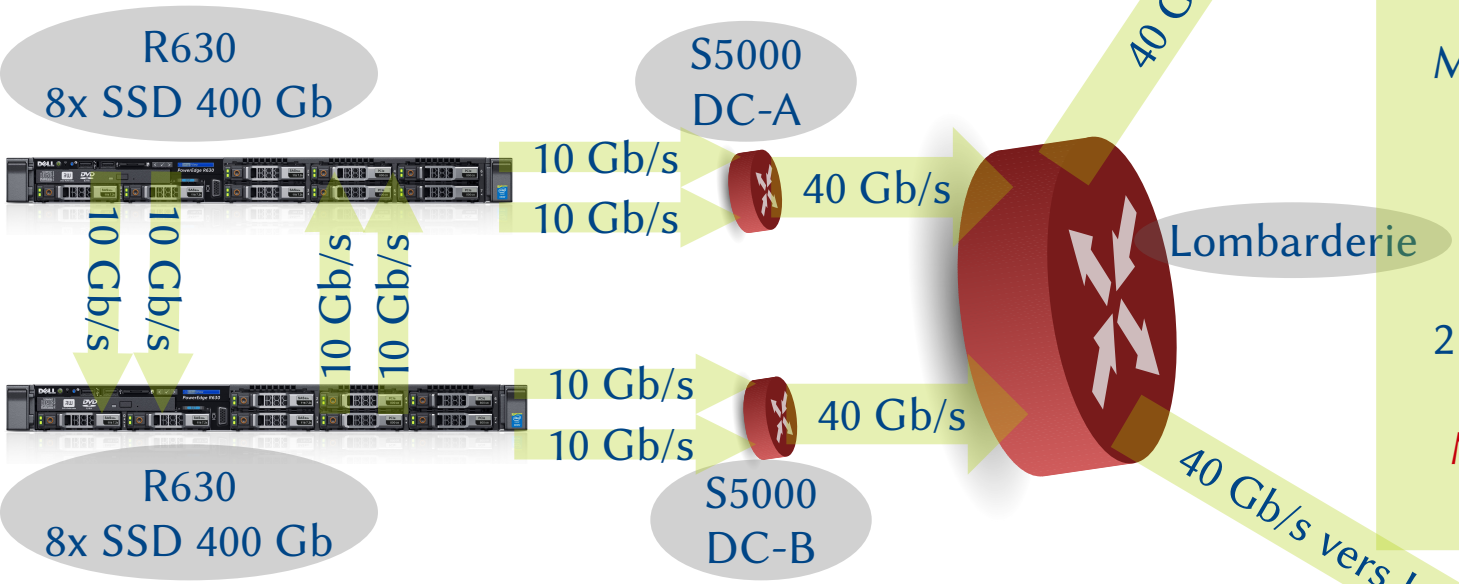
Tout pool peut être tier d'un autre.
sans nécessité de similarité
(taille ≠, réplicat X2 ≠ EC 4+1, racine ≠)

Le tier peut cacher (overlay) le pool d'origine

Il a des modes (write back, forward, read-only)
Il a des règles (nb max objet, taux remplissage
Fréquence flush...)



Setup machines SSD



2 machines au sein du datacenter
Latence minimale (- de 0,1ms)

Réplicat x2 (!!)

Découpage en clusters/LXC comme autres machines

Machines orientées performance
(E5 2637 , 3,5 Ghz
SSD write intensive)

4 ports 10 Gb/s
2 bondings 2x10 Gb/s mode actif

Moins intéressant qu'un setup complet 40 Gb/s (Coût !)

Risques associés

Adhérence des objets au cache : jamais flushé si mauvais réglage des règles.

En cas de crash du cache, peut-on récupérer les pools sous-jacents ?

Erasure Coding : Pas mieux que 2+1 pour garantir une tolérance à la panne. (sur notre setup)

PAS à mettre sur tous les POOLS.

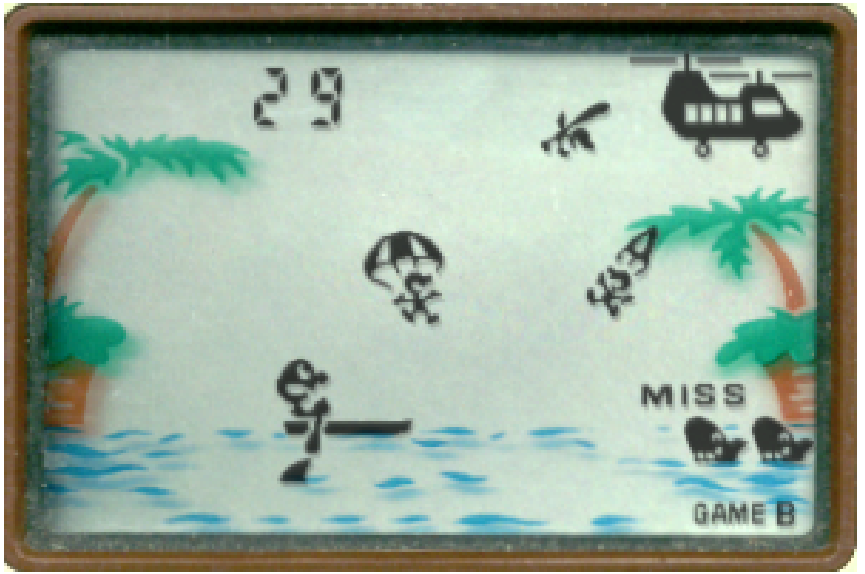
Éviter le désastre

SCRUB automatique pour le cluster
(patrouille sur les PG et fait des tests de cohérence)
Ne PAS utiliser size=2

Utiliser les outils sous-jacent du FS des OSD
(mkfs.xfs -m crc=1,finobt=1)

Utiliser des noyaux éprouvés
Une distribution à jour
Des versions de Ceph à jour (mais pas trop)

Monitoring
Lire les logs
Avoir une bonne connaissance du système sous-jacent
Lire les listes de diffusion



SAUVEGARDER !

Ceph status

Monitoring

Cluster OSD Pool PG Radios Gateway MDS Misc.

Cluster health at 10:16:14

Read : 5.3 MB/s , Write : 24.7 MB , Recovery : 0.0 bytes/s

OK

history

2015-11-18 10:14:33 : OK

Monitors status

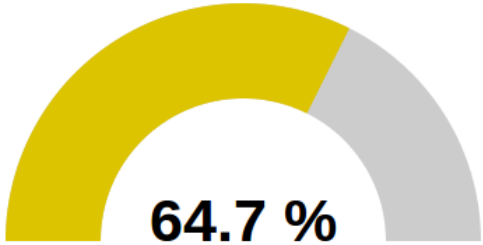
b a d

10432 Placement groups

active+clean+scrubbing+deep
active+clean
active+clean+scrubbing

100%

Avail. capacity : 115.3 TB



64.7 %

211.6 TB used on 326.8 TB

117 OSD

	UP	DOWN
IN	117	0
OUT	0	0

Full	near	full

pools

clean	unclean
9	0

MDSs

up:in	1	1
up:standby		
max		1

Vue inkscope

Répliqua x3

~75 To utiles

9 pools

UNIVERSITÉ DE NANTES

Quelques mots de sagesse

Une volumétrie importante !

Démarrer plusieurs clusters CEPH (soit physique, soit virtuels)

Exemple : cluster de sauvegarde

Des besoins différents
Des administrateurs différents
Des versions différentes

Kraken

Format disque bluestore stabilisé
Erasure coding sans Tiers pour RBD
Scrub en pause pendant les phases de reconstruction / rééquilibrage des données
...

V11.1.0 ⟶ Beta
Kraken = support limité

TP.

Création d'un cluster CEPH
Administration
Création de pools
L'utiliser en RADOS
L'utiliser en RBD (kRBD)
L'utiliser en CephFS
Tolérance à la panne
Monitoring
Tuning
...

Merci !

Questions ?

Crédits : Opencliparts / Openstreetmap / Ceph.com