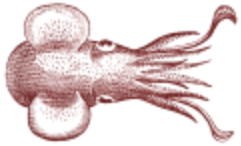
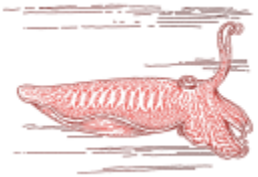
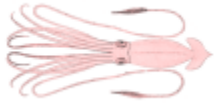
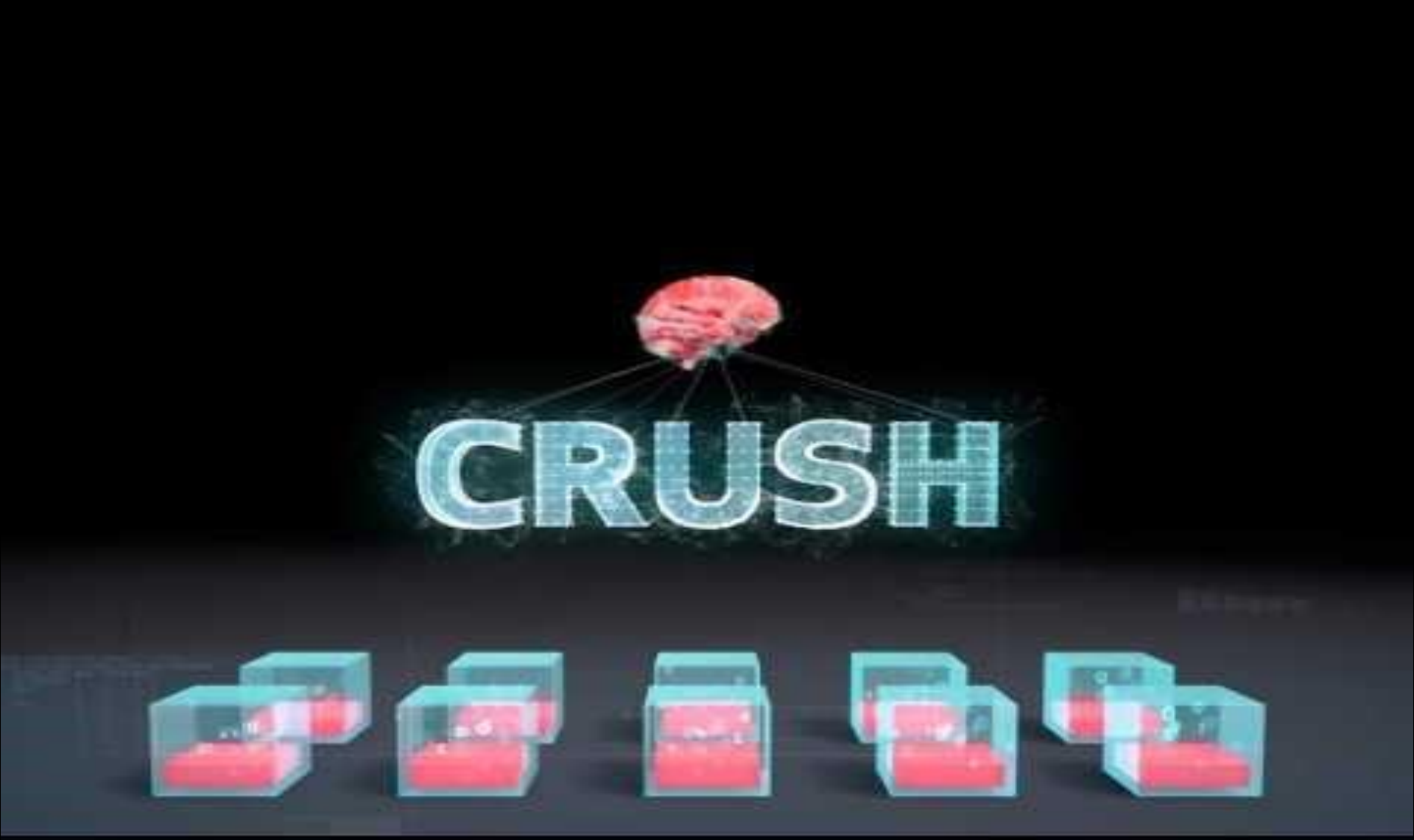




Ecole d'été ceph

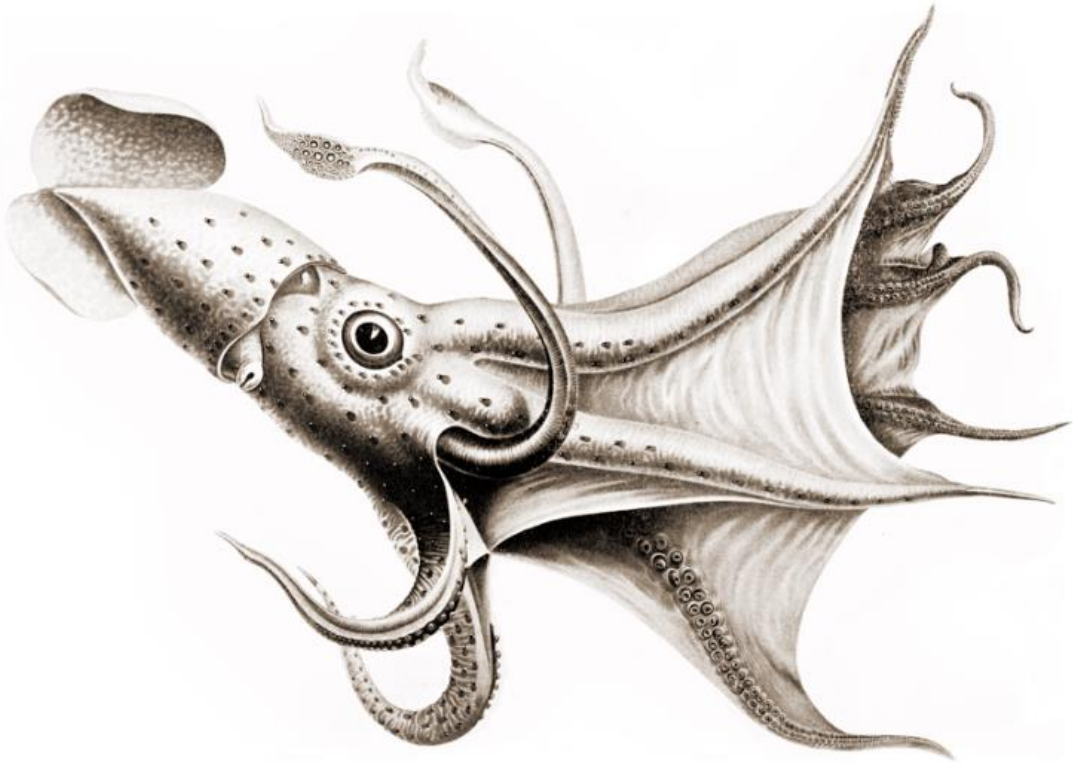
Concepts




CRUSH

Plan

- Concepts généraux CEPH
- définition du besoin
- Sécurité et CEPH (cephx, réseau interne) : qui accède aux machines et comment ?
- Présentation et première prise en main de l'infrastructure utilisée lors des TP



Concepts CEPH

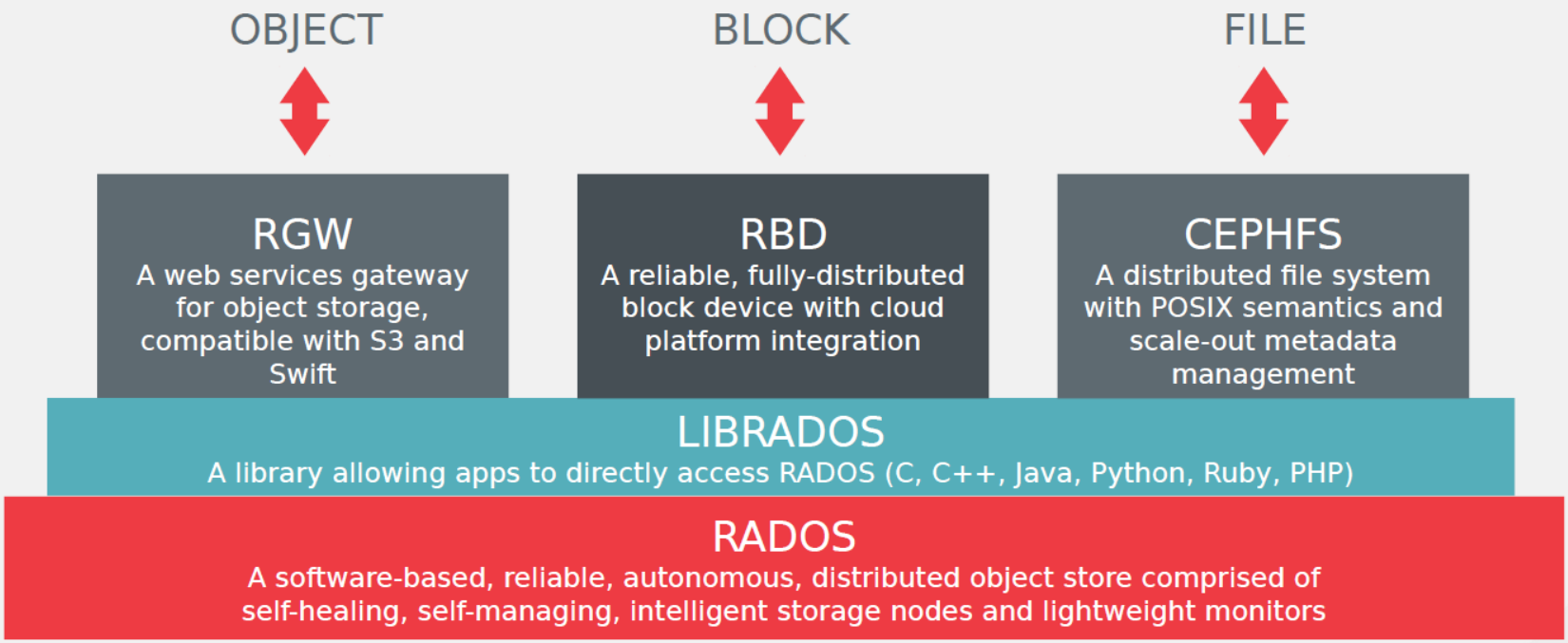
- Object, block, and file storage in a single cluster
- All components scale horizontally (1)
- No single point of failure
- Hardware agnostic, commodity hardware (2)
- Self-manage whenever possible
- Open source (LGPL)

1 (wikipedia) : L'évolutivité *horizontale* consiste à ajouter des ordinateurs pour faire face à une demande accrue d'un service. La méthode la plus courante est la répartition de charge (*load balancing*) par utilisation d'une grappe de serveurs (*cluster*). C'est une technique couramment utilisée par les serveurs web[15].

L'évolutivité *verticale* consiste à utiliser un ordinateur qui offre de nombreuses possibilités d'ajout de pièces, sur lequel il est possible de mettre une grande quantité de mémoire, de nombreux processeurs, plusieurs cartes mères et de nombreux disques durs.

2: WYBIWYG : what you buy is what you get (tm)

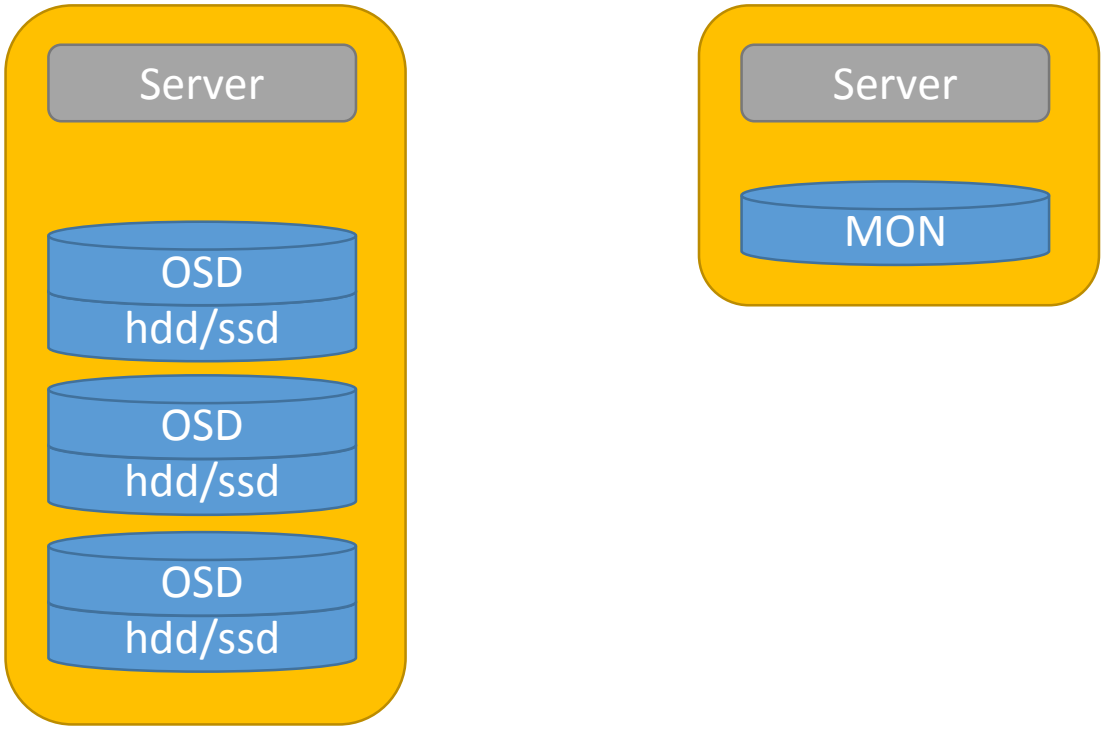
CEPH COMPONENTS



- RGW = Rados Gateway
- RBD = Rados Block Device

Ceph : RADOS

- Reliable Autonomic Distributed Object Store
- Contient 2 types de démons au minimum

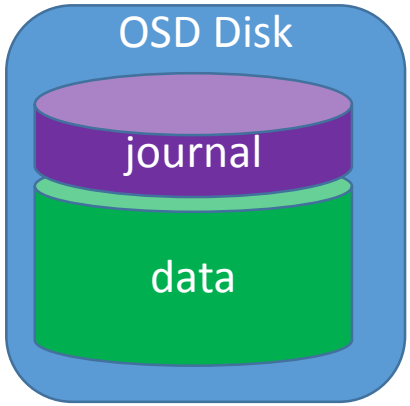


Ceph daemon : OSD

- **Object Storage Device**
- Démon responsable du stockage d'objets RADOS
- Fait appel à un backend de stockage :
  - FileStore : le + connu, le plus utilisé, le plus stable
  - BlueStore : en phase de stabilisation
  - Avant Bluestore, KeyValueStore
  - MemStore (intérêt ??)

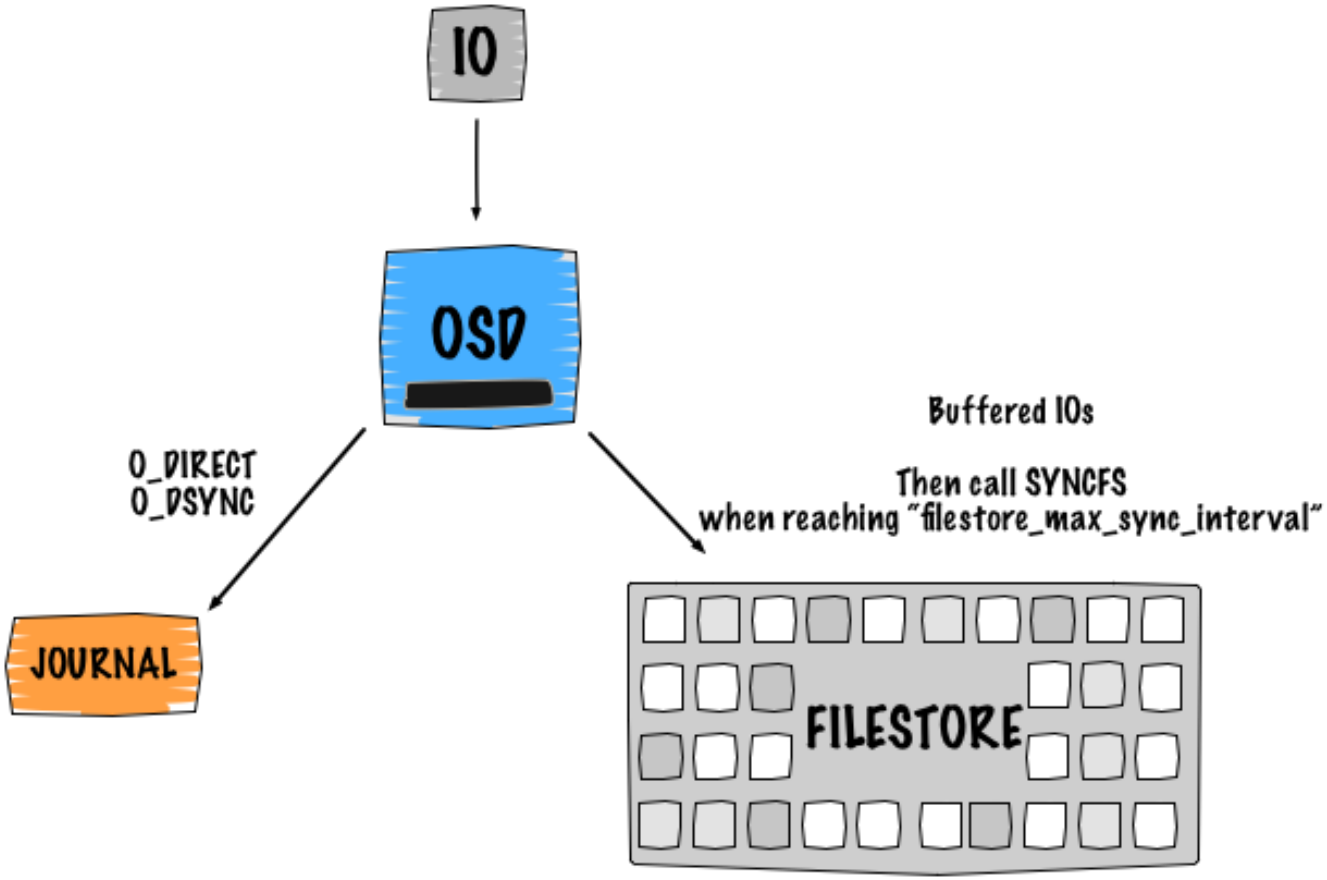
Ceph OSD : FileStore

- utilisation d'un filesytem POSIX pour stocker les données (XFS recommandé, btrfs, ext4 et zfs possibles)
  - ext4 plus recommandé : limitations taille extattrs (*)
  - btrfs : si vous êtes joueurs...
- Utilisation des attributs étendus pour stocker des metadata



https://www.sebastien-han.fr/blog/2013/12/02/ceph-performance-interesting-things-going-on/
* : http://docs.ceph.com/docs/jewel/rados/configuration/filesystem-recommendations/

Ceph OSD : FileStore

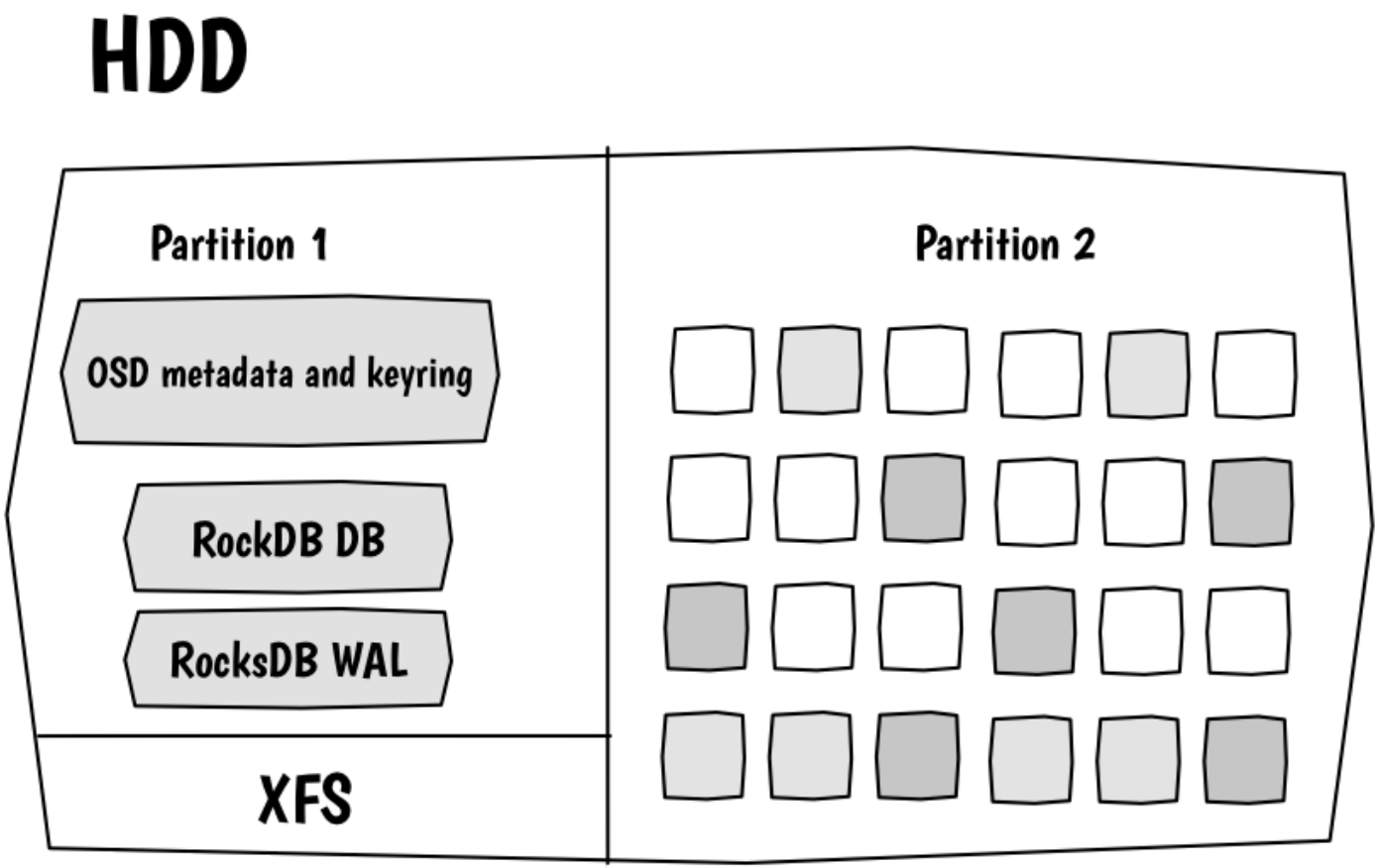


https://www.sebastien-han.fr/blog/2013/12/02/ceph-performance-interesting-things-going-on/

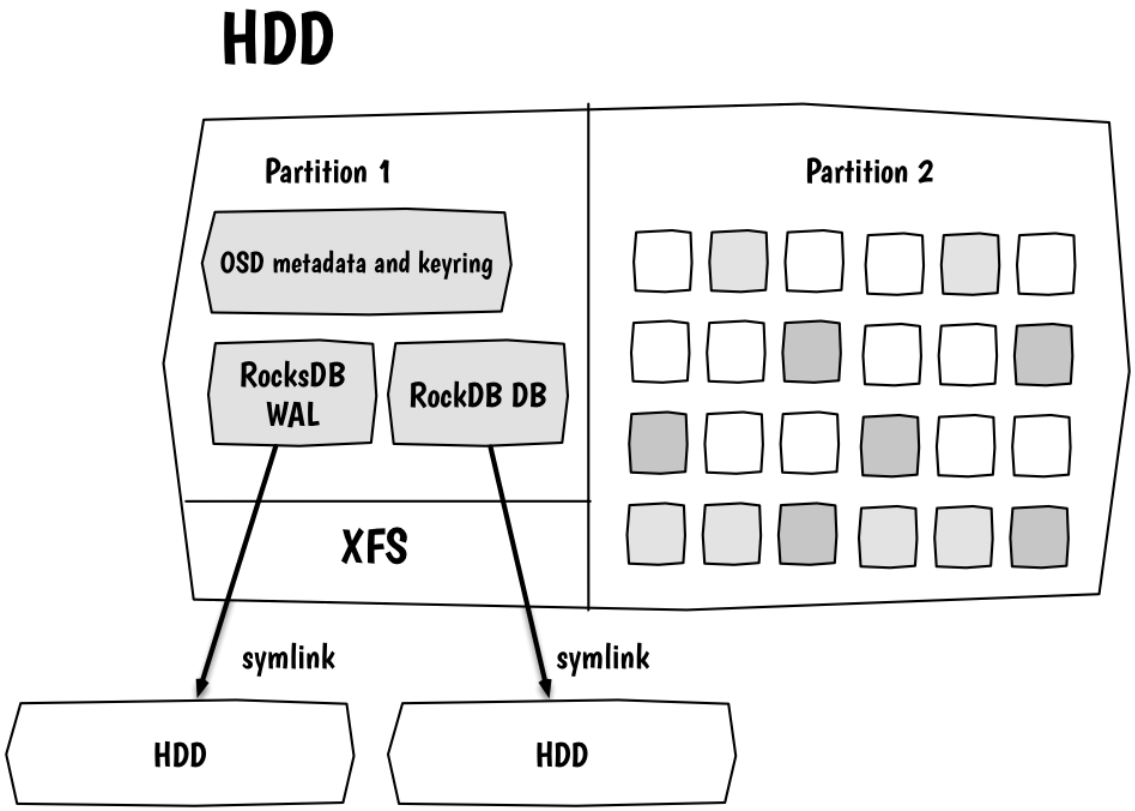
Ceph OSD : Bluestore

- Utilisation de rocksdb : base de données transactionnelle
  - Objects metadata
  - Write-ahead log
  - Ceph omap data
  - Allocator metadata : determine l'emplacement des doneness
- Plus de nécessité de journal ceph
- Plus de « double write penalty »
- Utilisation interne d'un filesytem « maison » (bluefs) qui implémente juste les interfaces nécessaires au stockage des données rocksdb
- Utilisation d'un blockdev brut pour stocker les données

Ceph OSD : Bluestore



Ceph OSD : Bluestore

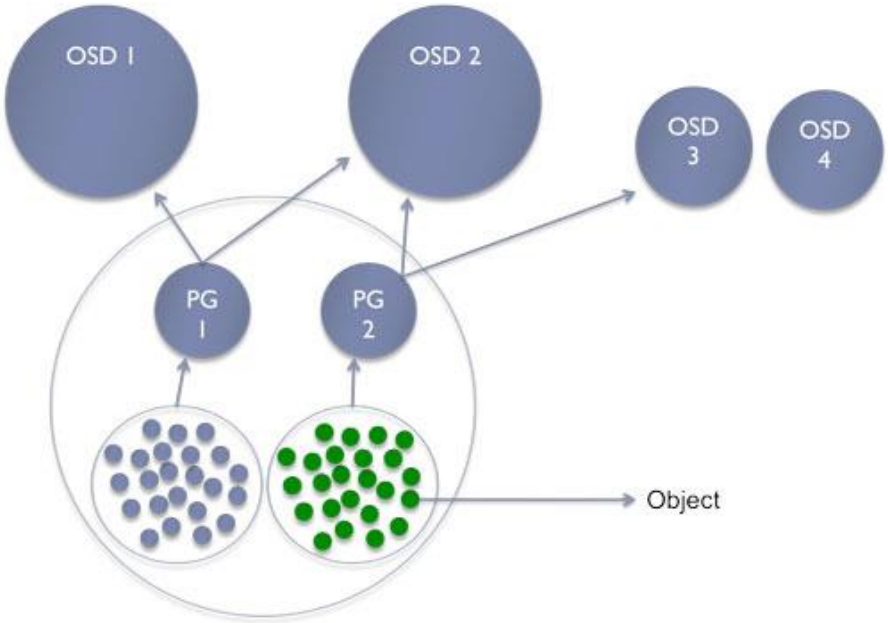


Ceph PG

- Un Placement Group (PG) agrège des objets à l'intérieur d'un pool
- Raisons d'être des PG:
  - monitorer le placement d'objets et leurs metadata dans des OSD individuellement est cher en CPU
  - irréaliste lorsque l'on a des millions, voire milliards d'objets

Ceph : PG

- PG = groupe d'OSD où chaque OSD possède 1 replica d'un objet RADOS
- Nombre d'OSD d'un PG = nombre de replica du pool
- Chaque PG a un OSD primaire désigné
- Un objet rados atteint en 1er l'OSD primaire, puis est répliqué vers les OSD secondaires



Ceph PG

- Plus on a de PG (par pool), plus on consomme de RAM et de CPU
- Les PG (leur nombre) déterminent la distribution des objets sur les OSD utilisés par un pool
- Il est possible d'augmenter le nombre de PG d'un pool
- mais **impossible** de le diminuer
- Crush ne peut pas prendre en compte la taille des objets : la distribution des données sur les OSD peut être non uniforme si un faible nombre de très gros objets sont stockés dans un pool de petits
- Si un PG est en erreur, des données sont inaccessibles...

Différents états des PG

état	signification
creating	Ceph is still creating the placement group.
active	Ceph will process requests to the placement group
clean	Ceph replicated all objects in the placement group the correct number of times
down	A replica with necessary data is down, so the placement group is offline
replay	The placement group is waiting for clients to replay operations after an OSD crashed
splitting	Ceph is splitting the placement group into multiple placement groups
scrubbing	Ceph is checking the placement group for inconsistencies
degraded	Ceph has not replicated some objects in the placement group the correct number of times yet
inconsistent	Ceph detects inconsistencies in one or more replicas of an object in the placement group (e.g., objects are the wrong size, objects are missing from one replica after recovery is finished.)
peering	The placement group is undergoing the peering process
repair	Ceph is checking the placement group and repairing any inconsistencies it finds (if possible)
recovering	Ceph is migrating/synchronizing objects and their replicas

Différents états des PG

état	signification
backfill	Ceph is scanning and synchronizing the entire contents of a placement group instead of inferring what contents need to be synchronized from the logs of recent operations. Backfill is a special case of recovery
wait-backfill	The placement group is waiting in line to start backfill
backfill-toofull	A backfill operation is waiting because the destination OSD is over its full ratio
incomplete	Ceph detects a placement group is missing information about writes that may have occurred, or does not have any healthy copies. If you see this state, try to start any failed OSDs that may contain the needed information, or temporarily adjust min_size to allow recovery
stale	The placement group is in an unknown state — the monitors have not received an update for it since the placement group mapping changed
remapped	The placement group is temporarily mapped to a different set of OSDs from what CRUSH specified
undersized	The placement group has fewer copies than the configured pool replication level
peered	The placement group has peered, but cannot serve client IO due to not having enough copies to reach the pool's configured min_size parameter. Recovery may occur in this state, so the pg may heal up to min_size eventually

Ceph PG

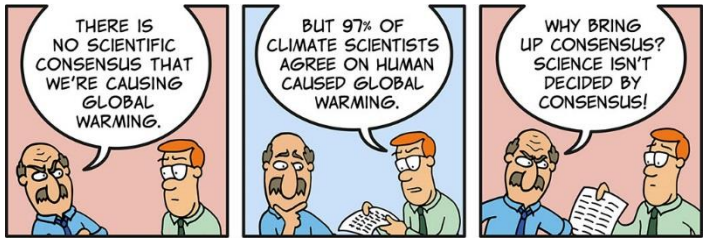
- Certaines règles sont à respecter pour le dimensionnement des PG **(qui déterminent la bonne distribution des données)**
- Le nombre de PG dans un pool détermine la durabilité des données (probabilité de perte)
  - Cas extrême : 1 unique PG pour 10 OSD, pool taille 3
    - => seuls 3 OSD utilisés. Les autres : vides.
  - Cas extrême 2 : 512 PG, pool size 3, 200 OSD
    - chaque OSD contiendra max (512*3)/200 ~ 7 PG
    - Si un OSD tombe, il faudra reconstruire 7 PG dont les réplicas se trouvent sur MAX 3*7 = 21 OSD
    - La reconstruction ne sera pas rapide (n'utilise pas 200 OSD !)
  - La règle à respecter est de prendre la puissance de 2 la plus proche de :

$$N = \frac{100 * OSDs}{pool\ size}$$

  - Dans notre cas, N=100*200/3=6666.66..., donc Npg = 8192
    - Chaque OSD contiendrait alors 8192*3/200=~ 122 PG
    - La reconstruction d'un OSD utiliserait alors max 122*3 OSD

http://docs.ceph.com/docs/master/rados/operations/placement-groups/

Ceph MON



- Partie clé d'un cluster ceph
- Utilise l'algorithme Paxos pour construire un quorum entre les différentes instance de ceph-mon
- Requiert N MON pour fonctionner
  - nécessite int(N/2)+1 MON up pour obtenir un consensus
    - 2 pour 3 MON, 3 pour 4, 3 pour 5, 4 pour 6...
  - Les MON majoritaires permettent de s'assurer de la consistance du cluster
  - Les erreurs/latences transitoires sont ainsi évitées et un MON out of date peut récupérer grâce au quorum

Ceph MON

- Les Mons maintiennent une “master copy” de la “cluster map”, y compris des membres du cluster, leur état, changements, et de l’état général du cluster
- La “cluster map” est un ensemble de “sub-maps”
- Chaque “sub-map” contient un historique itératif de ses changement d’état opérationnels.

Ceph MON : MAPS

- Les clients et OSD dépendent de la connaissance de la topologie du cluster
- Celle ci est connue grâce à 5 cartes « maps » qui forment la « cluster map »:
  - The Monitor Map: Contains the cluster fsid, the position, name address and port of each monitor. It also indicates the current epoch, when the map was created, and the last time it changed. To view a monitor map, execute ceph mon dump.
  - The OSD Map: Contains the cluster fsid, when the map was created and last modified, a list of pools, replica sizes, PG numbers, a list of OSDs and their status (e.g., up, in). To view an OSD map, execute ceph osd dump.
  - The PG Map: Contains the PG version, its time stamp, the last OSD map epoch, the full ratios, and details on each placement group such as the PG ID, the Up Set, the Acting Set, the state of the PG (e.g., active + clean), and data usage statistics for each pool.
  - The CRUSH Map: Contains a list of storage devices, the failure domain hierarchy (e.g., device, host, rack, row, room, etc.), and rules for traversing the hierarchy when storing data.
    - To view a CRUSH map, execute ceph osd getcrushmap -o {filename};
    - then, decompile it by executing crushtool -d {comp-crushmap-filename} -o {decomp-crushmap-filename}.
  - The MDS Map: Contains the current MDS map epoch, when the map was created, and the last time it changed. It also contains the pool for storing metadata, a list of metadata servers, and which metadata servers are up and in.
    - To view an MDS map, execute ceph fs dump.
- Note : certaines notions seront abordées plus tard.

Ceph : retour à RADOS

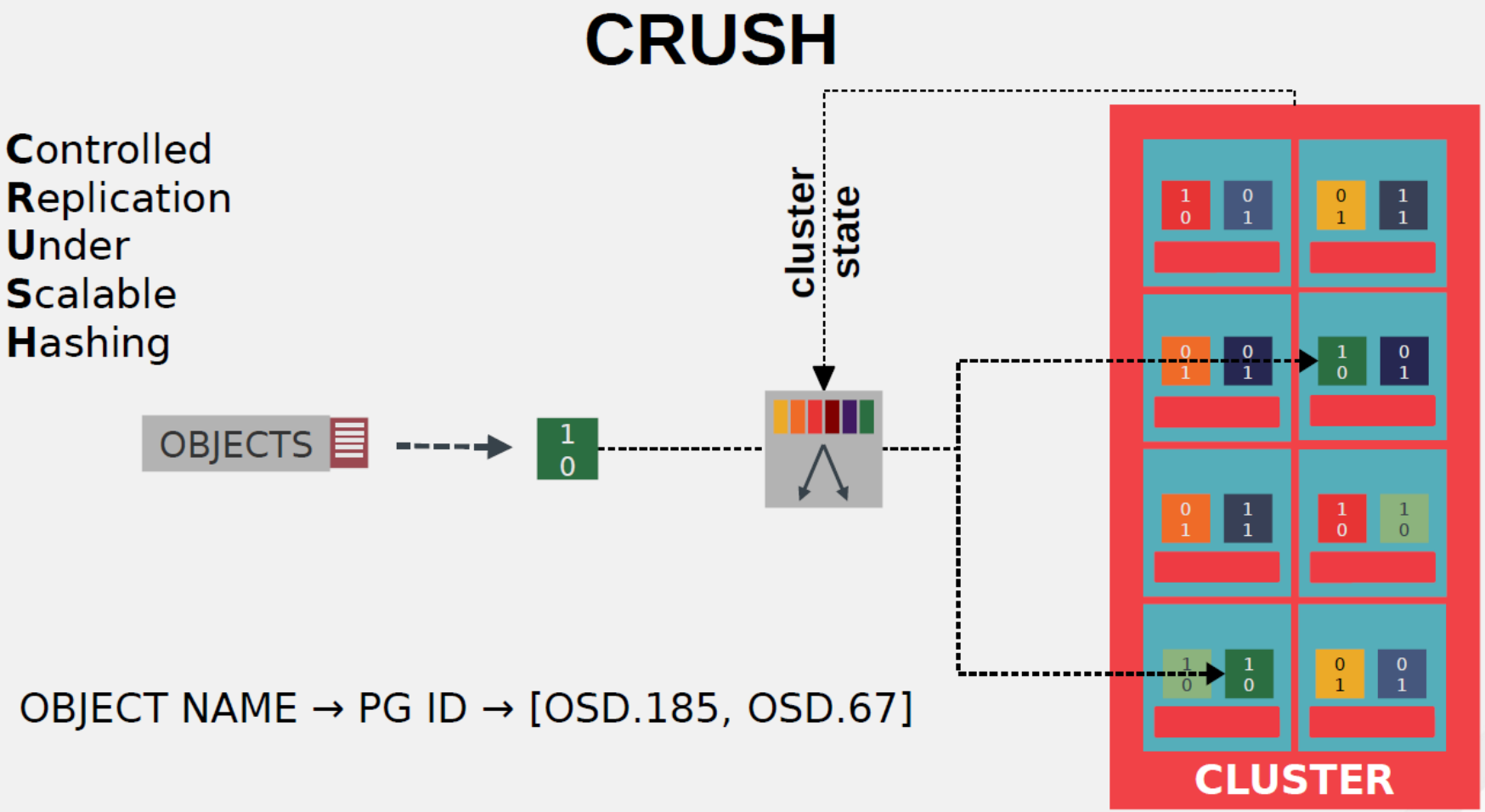
- A Ceph Monitor maintains a master copy of the cluster map. A cluster of Ceph monitors ensures high availability should a monitor daemon fail. Storage cluster clients retrieve a copy of the cluster map from the Ceph Monitor.
- A Ceph OSD Daemon checks its own state and the state of other OSDs and reports back to monitors.
- Storage cluster clients and each Ceph OSD Daemon use the CRUSH algorithm to efficiently compute information about data location, instead of having to depend on a central lookup table. Ceph's high-level features include providing a native interface to the Ceph Storage Cluster via librados, and a number of service interfaces built on top of librados.

Ceph : crush

- Controlled Replication Under Scalable Hashing

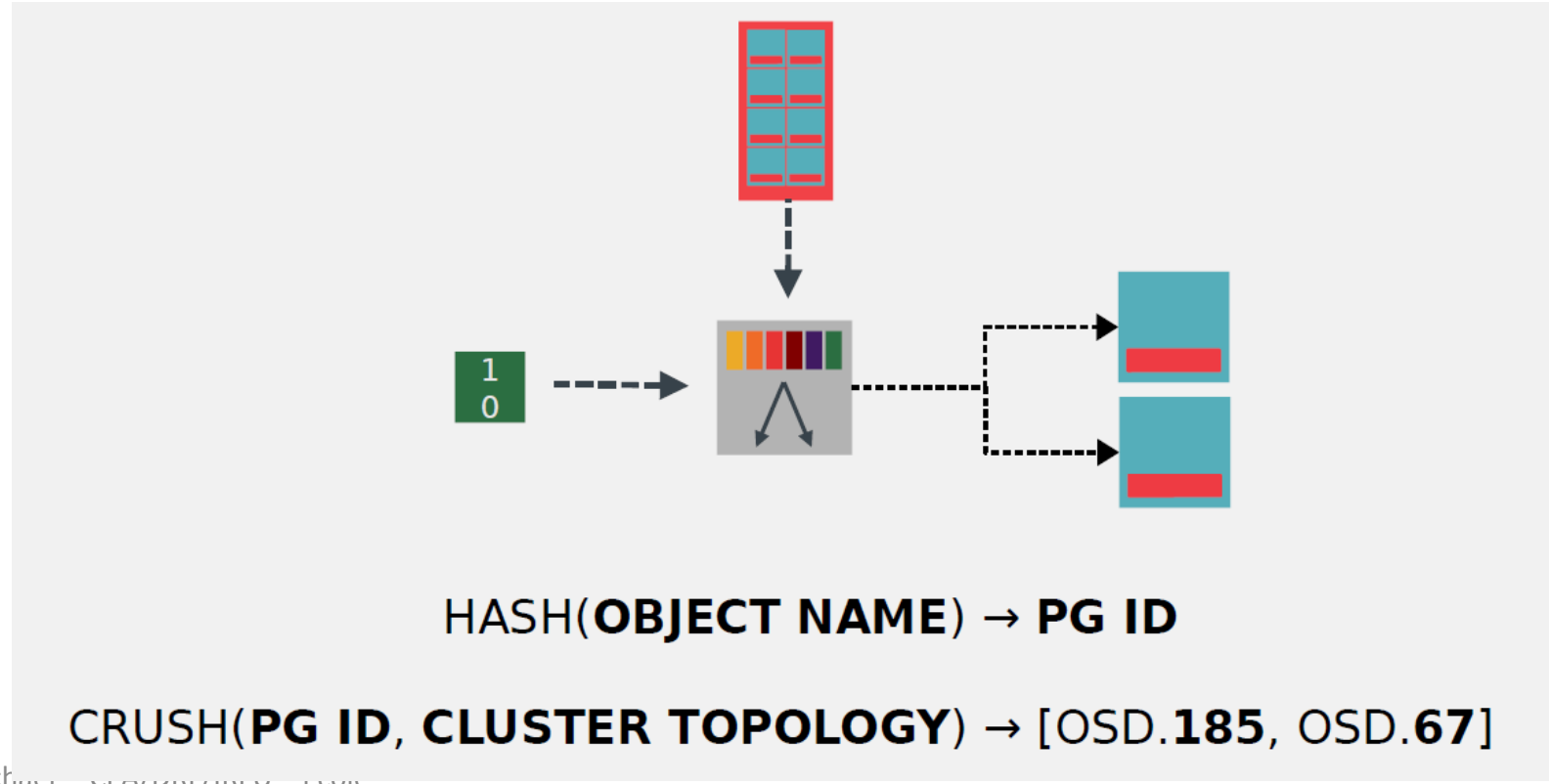
- « Ceph's CRUSH algorithm liberates storage clusters from the scalability and performance limitations imposed by centralized data table mapping. It replicates and re-balance data within the cluster dynamically—elminating this tedious task for administrators, while delivering high-performance and infinite scalability. »

Ceph: crush


CRUSH
Controlled
Replication
Under
Scalable
Hashing
cluster
state
OBJECTS
1
0
OBJECT NAME → PG ID → [OSD.185, OSD.67]
CLUSTER

Ceph CRUSH = fonction

- connue de tous
- params connus des MON et distribués via les MAPS



Ceph POOLS

- Les pools permettent de séparer les données
  - a des fin administratives (quotas, statistiques,...)
  - pour des raisons de sécurité (restrictions utilisateurs)
- Des pools différents permettent aussi de gérer différemment des données
  - nombre de réplicas
  - erasure coding
  - machines ou types de disques utilisés
- Un pool peut en cacher un autre (;))
  - cf plus tard

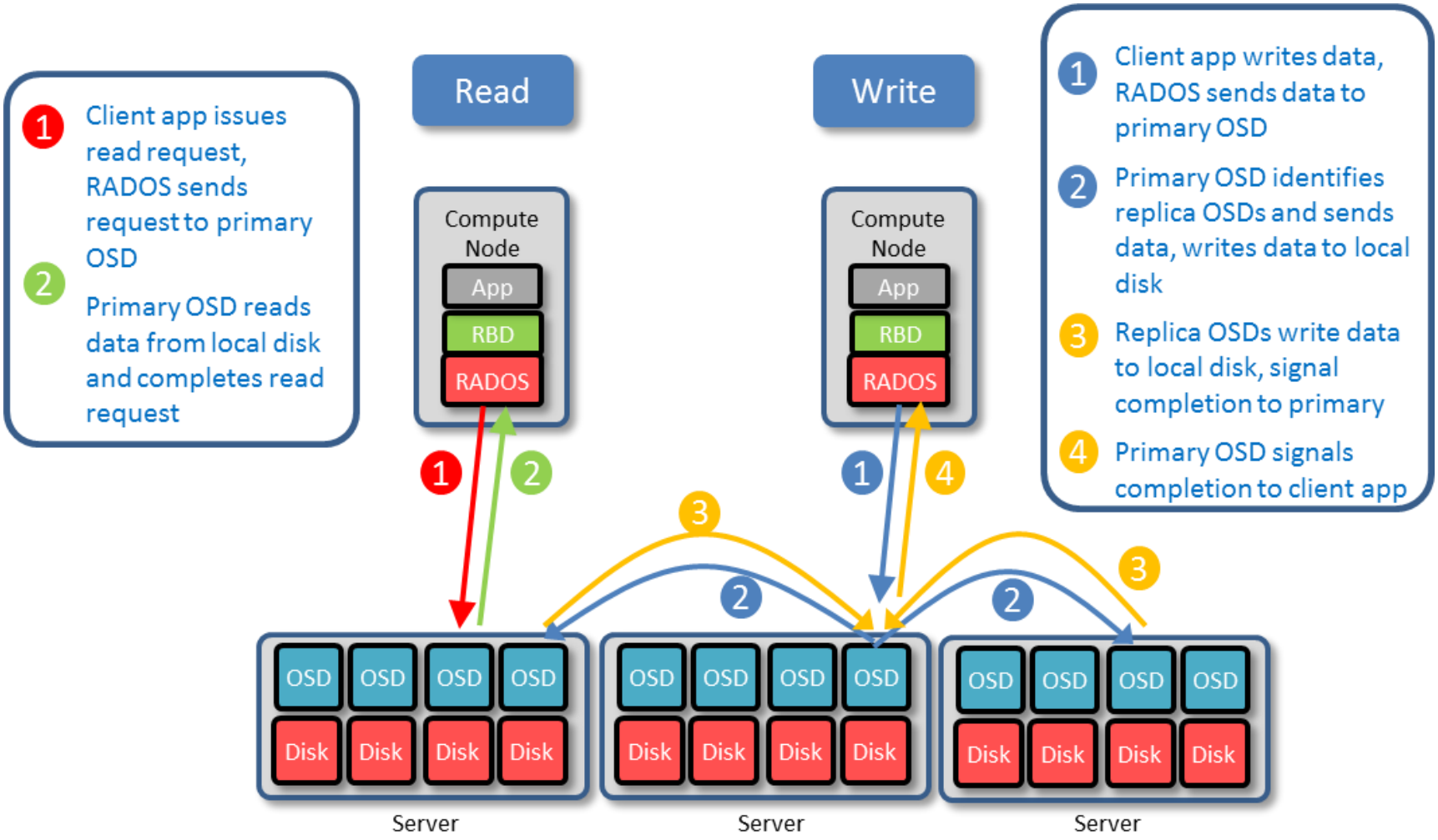
Ceph POOLS : replicated

- Le type de pool le plus utilisé
- Chaque donnée est répliquée N fois
- Il est (était ?) fortement déconseillé d'avoir MOINS de 1+2 replicas
  - Avec 2, possibilité de split head
    - si une donnée est déteriorée, comment déterminer celle des deux qui est exacte ?
  - ceph ne stock**ait** pas de checksums au début
    - mais maintenant si, et le deep scrubbing permet de détecter les données corrompues

Ceph POOLS : replicated

- A chaque écriture client, les données sont transférées par l'OSD primaire vers les replicas
- N réplicas => N transferts, N IO
  - terme commun : write amplification
    - Chaque réplication passera aussi par le cache ceph
    - et les journaux XFS
- un réseau interne dédié permet de ne pas impacter la bande passante utilisateur dispo lors de la réplication

Ceph POOLS : replicated flow

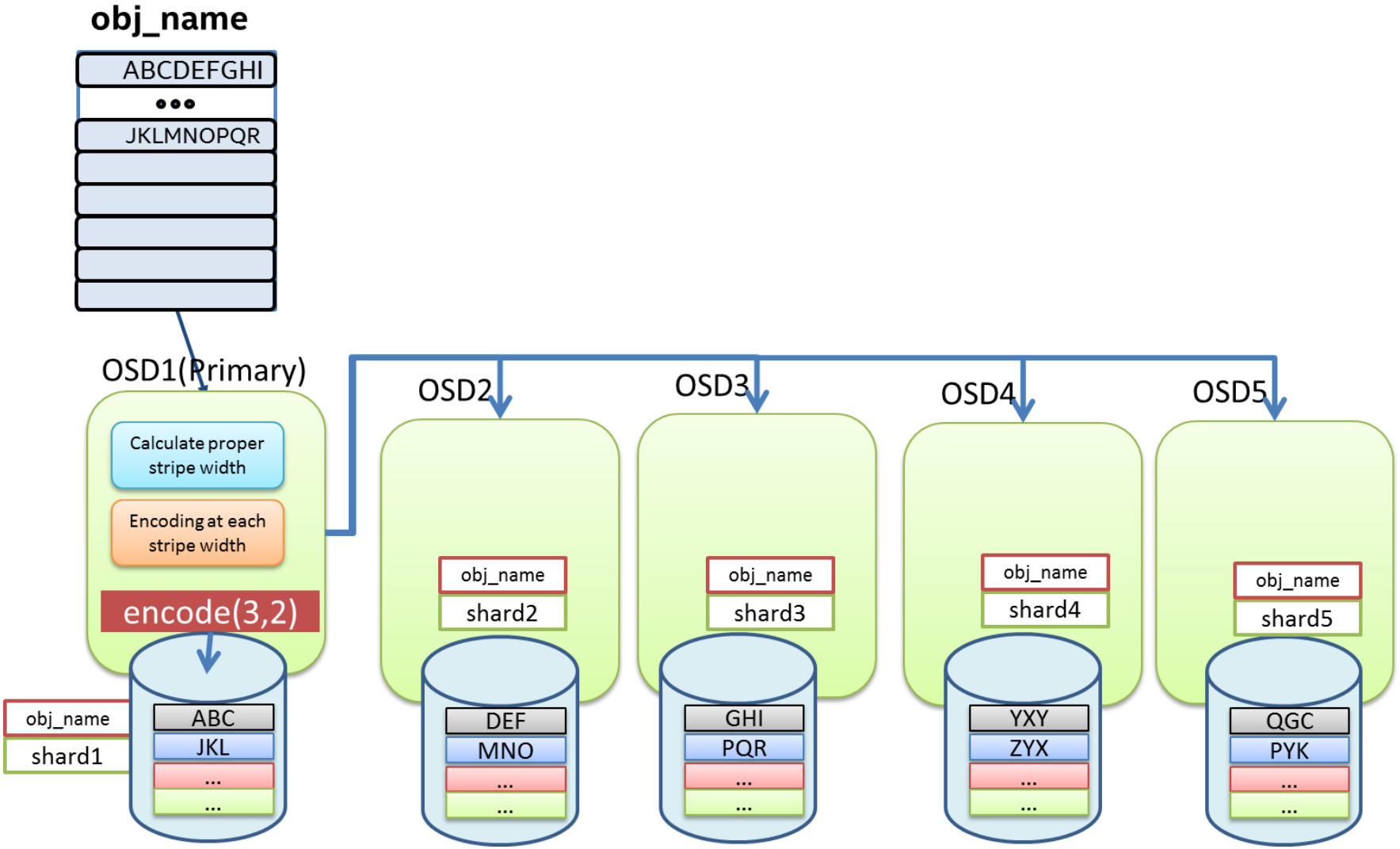


https://software.intel.com/en-us/blogs/2015/04/06/ceph-erasure-coding-introduction

Ceph POOLS : Erasure Coding

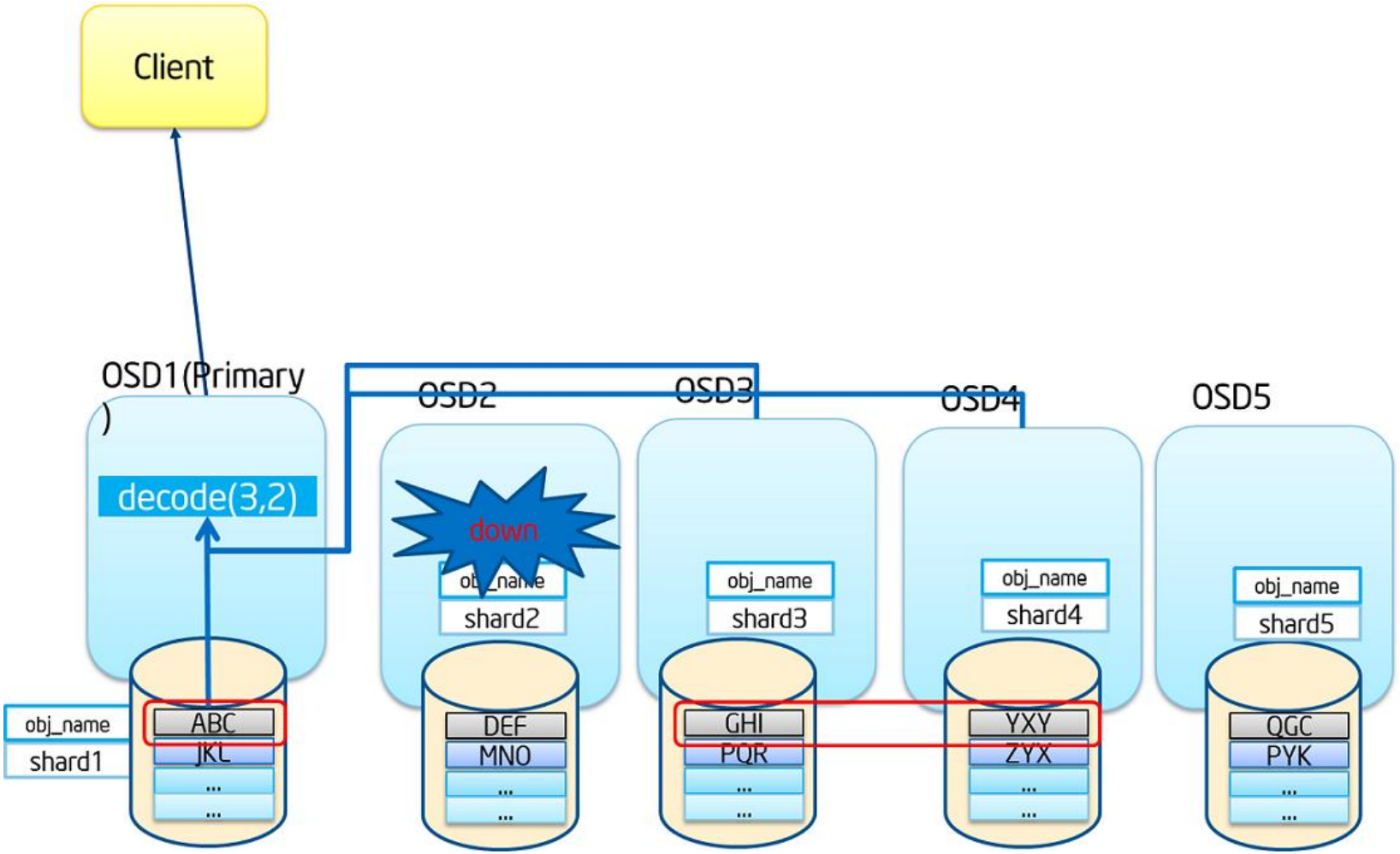
- Permet de définir la tolérance à la panne
- On choisit :
  - le nombre de morceaux de données (K)
  - le nombre de métadonnées (M)
  - On peut alors « perdre » jusqu'à M morceaux, quels qu'ils soientt, et réussir à reconstruire les morceaux perdus
- Exemple :
  - deux RAID6 : 2*(K=8, M=2) (M est fixe) : perte de donnée si 3 disques HS dans 1 raid
  - avec CEPH : K=16, M=4 : perte si *5* disques HS, même ratio metadata
- Un algorithme (code correcteur d'erreur) convertit les données utilisateur à la volée en K+M morceaux
  - plugable : différents plugins existent
  - Jerasure, ISA-l (Intel Storage Acceleration library) , LRC
- Le PG associé à l'objet contient K+M OSD
- Chaque OSD contient un morceau de la donnée transformée (K ou M)

Ceph POOLS : Erasure Coding flow - WRITE



https://software.intel.com/en-us/blogs/2015/04/06/ceph-erasure-coding-introduction

Ceph POOLS : Erasure Coding flow – READ when OSD down



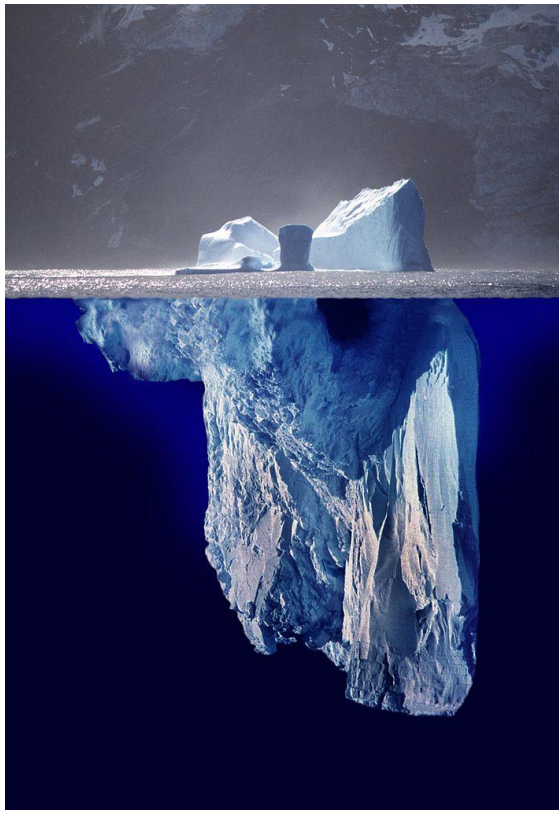
https://software.intel.com/en-us/blogs/2015/04/06/ceph-erasure-coding-introduction

Ceph POOLS : Erasure Coding

- Avantage :
  - espace utilisé très inférieur à la réplication
  - Et donc coût/TCO inférieur (à priori)
  - Allez répliquer 3 fois 3PiB...
- Inconvénient :
  - très gourmand en CPU et en mémoire !
  - Les calculs sont effectués sur l'OSD primaire
  - Chaque IO nécessite de décoder l'ensemble des données
    - 1ko ? => on décode/encode (N+M)x taille de bloc ceph (par défaut : 4MiB)
  - Accès aléatoires peu performants
  - ET la ré-écriture partielle n'était pas possible jusqu'à présent
    - impossible donc d'utiliser un pool EC pour y stocker des VMs directement

Ceph POOLS : cache tiering

- le tiering permet de mettre un pool en overlay d'un autre
- On peut ainsi combiner un pool de stockage ultra performant (et cher) avec un pool moins coûteux mais plus gros
- Les données sont migrées d'un pool à l'autre
  - différentes politiques possibles



Ceph POOLS : cache tiering

- Attention !

- **Known Bad Workloads**
  - The following configurations are *known to work poorly* with cache tiering.
  - *RBD with replicated cache and erasure-coded base*: This is a common request, but usually does not perform well
  - *RBD with replicated cache and base*: RBD with a replicated base tier does better than when the base is erasure coded, but it is still highly dependent on the amount of skew in the workload, and very difficult to validate

- M'enfin : il faut essayer et voir SON workload
- pour en savoir plus :

http://docs.ceph.com/docs/master/rados/operations/cache-tiering/

Ceph MDS

- Meta Data Server
- Contient les meta-données de la partie CephFS
- Jusqu'à présent (*) :
  - plusieurs MDS étaient possibles mais en configuration ACTIF/PASSIF
  - Les configurations ACTIF/ACTIF causaient des corruptions de données
  - 1 seul cephfs était possible par cluster
- Cependant :
  - plusieurs MDS actifs sont possibles : un load balancing est alors automatique
  - Il possible de séparer la charge MDS par « rangs » et d'attribuer certains répertoires à ces rangs
  - Il est possible de créer plusieurs cephfs

  => ceph fs flag set enable_multiple true

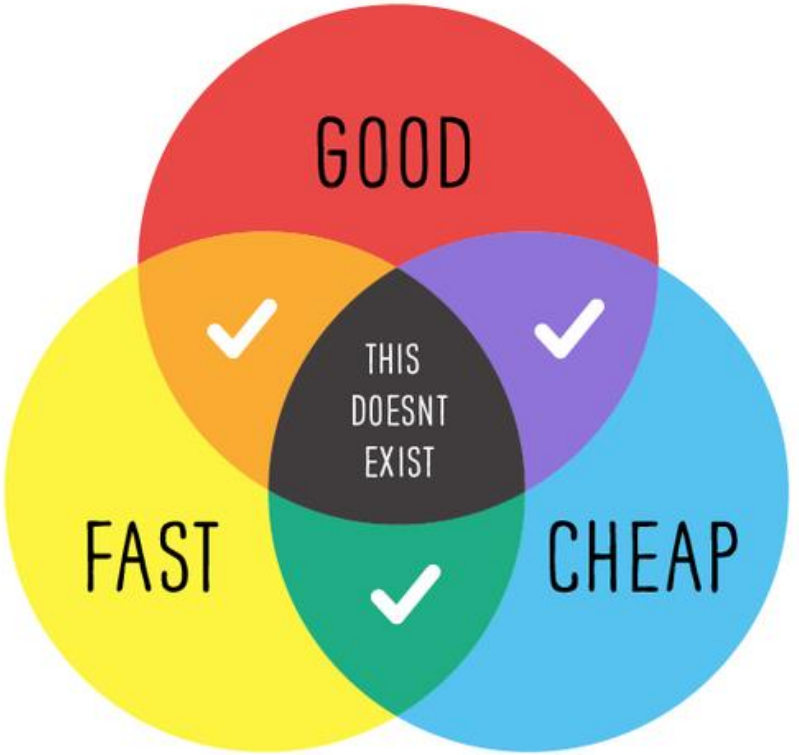
* : http://docs.ceph.com/docs/master/cephfs/multimds/

MAIS : http://docs.ceph.com/docs/master/start/quick-ceph-deploy/#add-a-metadata-server : no commerial support

Définition du besoin

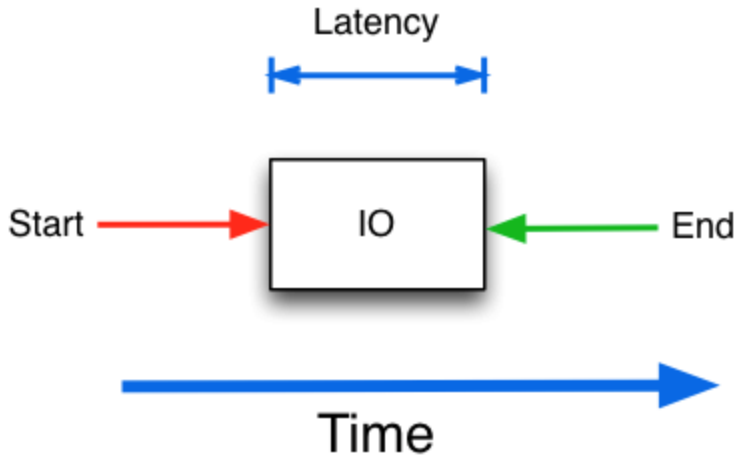
Définition du besoin :

- IOPS/Latence
- Taille d'IO
- Bande passante
- Nombre de clients



https://www.host1plus.com/blog/articles/black-friday-post-incident-report/

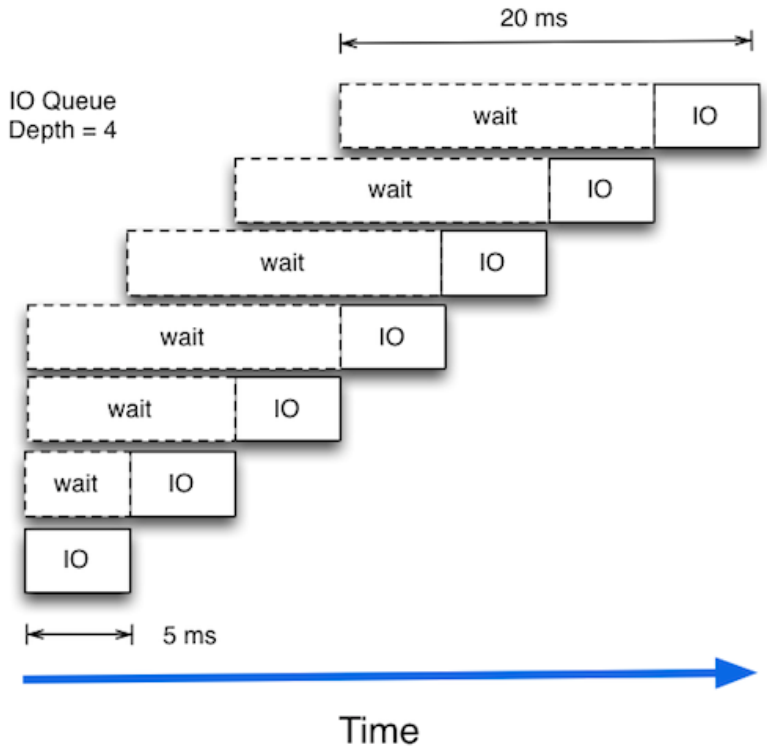
C’est quoi, des IOPS



- Terme à la mode
- Ne signifie rien sans contexte
- Dépend de la latence
  - « vitesse à laquelle une requête d’IO est traitée »
  - 1000 IOPS avec une latence de 10ms PEUT être mieux que 5000 IOPS avec une latence de 50ms
- dépend de nombreux autres facteurs
- Mais 1000 IOPS != 1000 IOPS parallèles
  - Si 2 IO arrivent en même temps, la seconde doit attendre la fin de la 1ere : sa latence augmente

IOPS et queue depth

- Les IOPS peuvent être réordonnées : + IOPS
- Mais + de latence en moyenne
- Ici, 20ms au lieux de 5



IOPS = f(x,y,z,...)

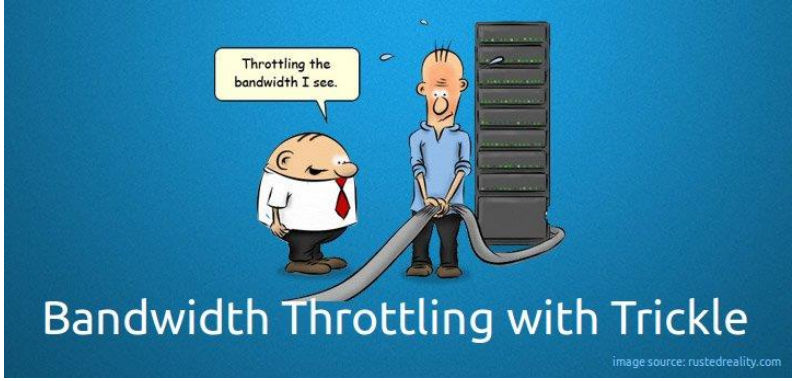
- IOPS dépendent donc de :
  - latence
  - queue depth
  - taille du cache
  - taille des données à traiter
  - type de travail : R ? W ? RW (10% read ?) ? R sequential ? W seq ?...
  - de la manière de bencher ET de l'outil de bench

Latence

- La latence est critique pour de nombreux workloads
  - une DB peut se bloquer tant que ses IO ne sont pas terminées
- Analogie latence/voiture
  - 2 voitures allant à 100 km/h
    - #1 met 50s à atteindre 100 km/h
    - #2 met 200s
    - Qui dirait que #2 est la plus rapide ?

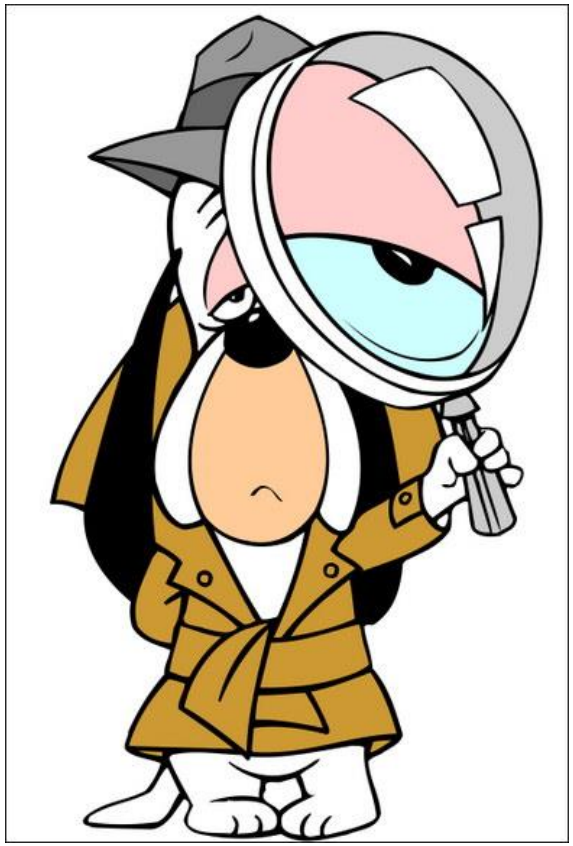
  - 2 voitures
    - #1 met 30s à atteindre 100km/h
    - #2 met 50s à atteindre 100km/h, puis 200s pour 150km/h
    - Quelle voiture *semblera* plus rapide à son conducteur ?

Bande passante



- Quelle bande passante ?
  - agrégée sur l'ensemble des clients ?
    - 30Gbits/s ! (mais pour 10000 clients)
  - Par client ?
  - Pour combien de clients ?
    - et on en revient alors aux IOPS...
- Pour quelle taille de données ?
  - peut on espérer obtenir 100 mbits/TiB/s pour 1000 clients et 2PiB, en lectures 4MB, avec une latence de 20ms ?

Sécurité



E84AC2C0460F3994 ?

⇒ http://ceph.com/releases/important-security-notice-regarding-signing-key-and-binary-downloads-of-ceph/

⇒ Intrusion sur des serveurs, September 17, 2015

⇒ Clés de signement des RPMs changée

Sécurité des données

- Comment sécuriser 2 ou 10 PiB de données ?
  - duplication / backups ?
  - codes correcteurs d'erreurs ?
  - gestion rigoureuse d'un hardware garanti
  - validation de chaque pièce de stockage ?
    - ex. @ IRFU : nouveau stockage (non ceph) mis en production un jeudi
    - 1 VD de 56TiB en RAID6 (16x4TB)
    - le lundi suivant : 4 HDD dead. VD en erreur. ~1TB utilisé...
      - La cause : le backplane du serveur (pas les disques !) causant des erreurs SAS, le firmware faisant le reste
  - Mise à jour (et validation ?) des firmwares ?
    - fw disques permettent de corriger des erreurs de timeouts sur IO, et déclaration parfois à tort de secteurs ou HDD défectueux
  - Matériel entreprise
    - oublier les SSD consumers

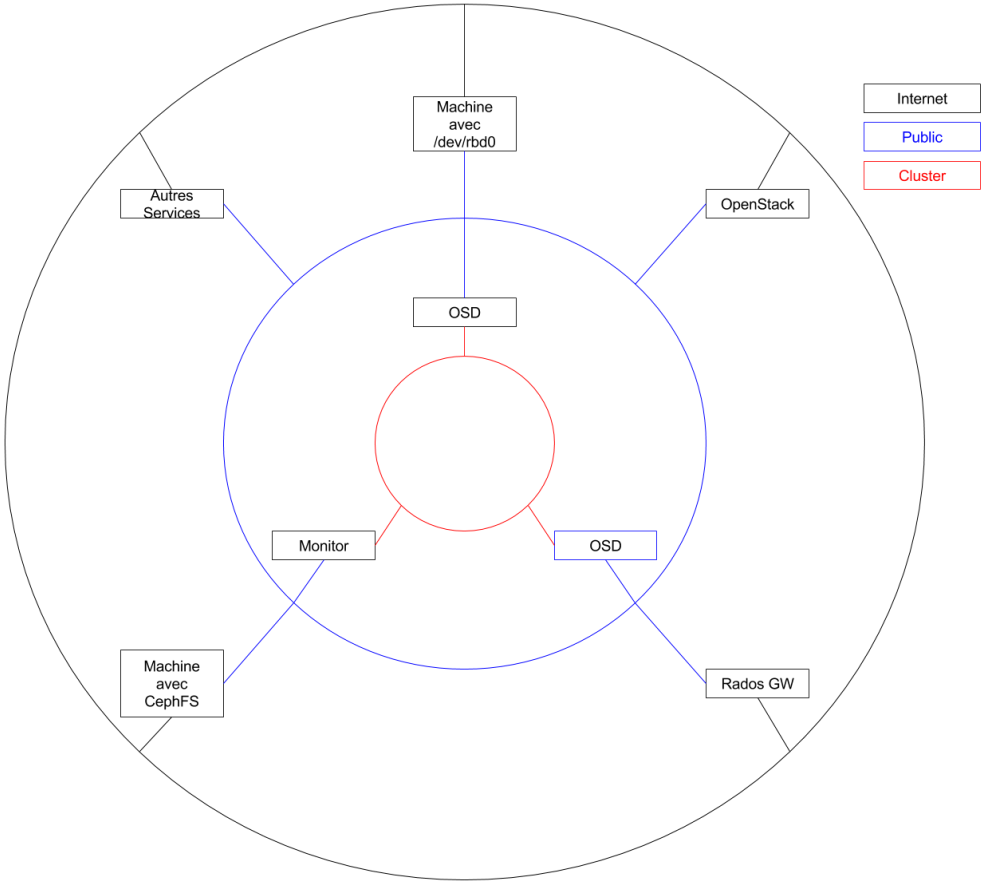
https://www.kingston.com/us/ssd/enterprise/best_practices/enterprise_versus_client_ssd

Sécurité des systèmes

- Il ne suffit pas de maintenir les systèmes à jour
  - les configurer correctement
  - les isoler
- Ceph préconise l'utilisation de 2 interfaces réseau
  - WARN : 1 réseau interne qui écoute sur 0.0.0.0 permettra à tout paquet provenant de tout réseau de tenter un DoS
  - Config ***DEFAULT*** jusque luminous si cluster network non spécifié
  - permet de dissocier le traffic interne du traffic client (QoS ET sécurité)
- Ceph permet d'activer la signature des messages
  - http://docs.ceph.com/docs/master/rados/configuration/auth-config-ref/#signatures
  - permet d'éviter le spoofing
- Ceph permet de donner des droits utilisateurs ET système via « cephx »

Sécurité ceph et utilisateurs

- les users CEPH ne doivent pas être des utilisateurs standard, mais des utilisateurs de services eux même chargés d'une gestion fine des droits
- Le(s) réseau(x) ceph ne doit pas être exposé directement au client final - « l'utilisateur »



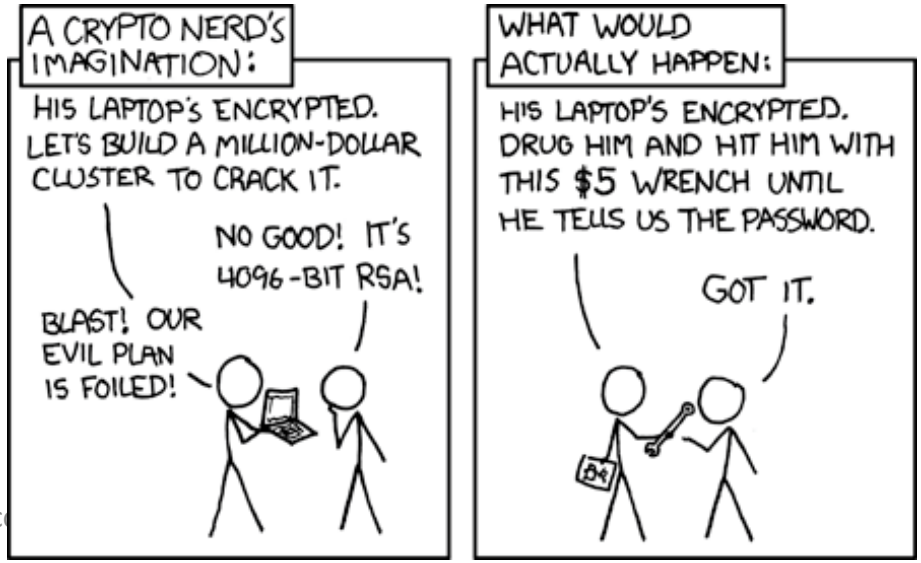
Cephx

- gestion +/- fine des droits utilisateurs
  - gestion basique par pool
  - possibilité de gérer les droits à l'intérieur des pools via des namespaces, mais UNIQUEMENT pour les outils utilisant librados (doc datant de firefly...)
  - http://docs.ceph.com/docs/master/rados/operations/user-management/
  - possibilité de DESACTIVER cephx
    - mais alors...

➔

CephX

- No SPOF
- autorise et authentifie chaque interlocuteur
  - les serveurs Ceph (MDS, MON, OSD...)
  - les utilisateurs (openstack, rados gw...)
- autorisation via ceph « caps » (capabilities)
- authentification via clés de chiffrement partagées
- sécurise ceph tant que les clefs restent secrètes



Cephx : CAPS

- Le format général des droits d'un user ceph est :

$$\{daemon-type\}\ 'allow\ \{capability\}'\ [\{daemon-type\}\ 'allow\ \{capability\}']$$

- Pour les MON, les caps sont au format :
  - mon 'allow rwx'
  - mon 'allow profile osd'
- Pour les OSD, les caps sont au format :
  - osd 'allow {capability}' [pool={poolname}] [namespace={namespace-name}]
- Pour les MDS, les caps sont au format (*) :
  - mds 'allow {capability} [path=/*the_path*]'
- Si un utilisateur a des droits OSD sans restriction de pool, ce client aura accès à TOUS les POOLS
- Exemples :

```
ceph auth add client.john mon 'allow r' osd 'allow rw pool=liverpool'
ceph auth get-or-create client.paul mon 'allow r' osd 'allow rw pool=liverpool'
ceph auth get-or-create client.george mon 'allow r' osd 'allow rw pool=liverpool' -o george.keyring
ceph auth get-or-create-key client.ringo mon 'allow r' osd 'allow rw pool=liverpool' -o ringo.key
ceph auth get-or-create client.foo mon 'allow r' mds 'allow r, allow rw path=/bar' osd 'allow rw pool=data'
ceph auth get-or-create client.foo mon 'allow r' mds 'allow rw path=/bar' osd 'allow rw pool=data'
```


http://docs.ceph.com/docs/master/rados/operations/user-management/

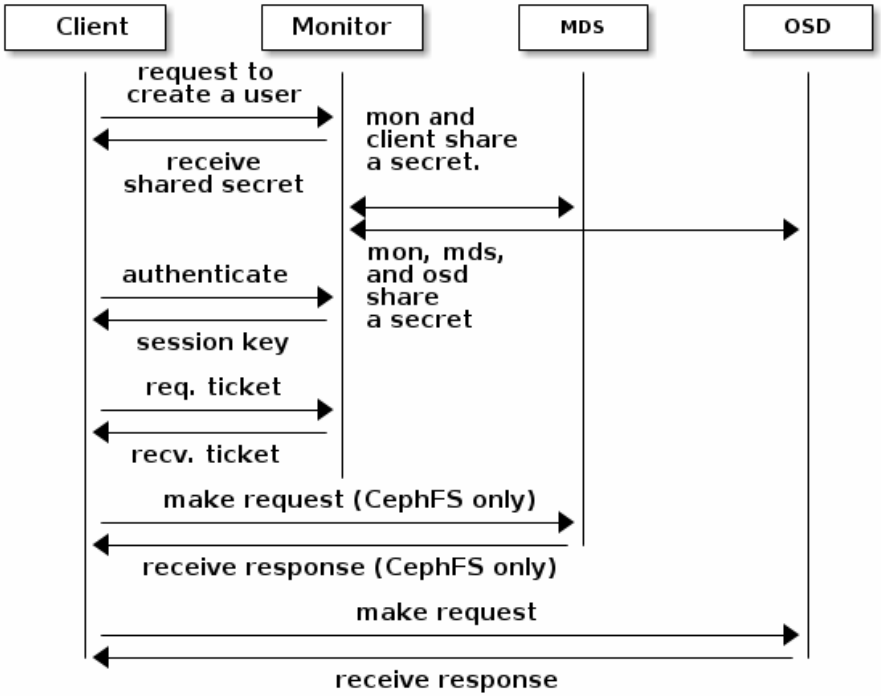
* : http://docs.ceph.com/docs/master/cephfs/client-auth/

Cephx : CAPS

cap	description
allow	Precedes access settings for a daemon. Implies rw for MDS only.
r	Gives the user read access. Required with monitors to retrieve the CRUSH map
w	Gives the user write access to objects.
x	Gives the user the capability to call class methods (i.e., both read and write) and to conduct auth operations on monitors.
class-read	Gives the user the capability to call class read methods. Subset of x.
class-write	Gives the user the capability to call class write methods. Subset of x.
*	Gives the user read, write and execute permissions for a particular daemon/pool, and the ability to execute admin commands.
profile osd	Gives a user permissions to connect as an OSD to other OSDs or monitors. Conferred on OSDs to enable OSDs to handle replication heartbeat traffic and status reporting.
profile mds	Gives a user permissions to connect as a MDS to other MDSs or monitors.
profile bootstrap-osd	Gives a user permissions to bootstrap an OSD. Conferred on deployment tools such as ceph-disk, ceph-deploy, etc. so that they have permissions to add keys, etc. when bootstrapping an OSD.
profile bootsrap-mds	Gives a user permissions to bootstrap a metadata server. Conferred on deployment tools such as ceph-deploy, etc. so they have permissions to add keys, etc. when bootstrapping a metadata server.

CephX : fonctionnement

- Le client s'authentifie via les MON
- les MON connaissent
  - les clefs de tout le monde
  - les autorisations
- Le client demande aux MON l'accès à un serveur S
- les MON donnent un « ticket » au client pour accéder au serveur S
- RAPPEL : un client écrit uniquement dans l'OSD primaire qui s'occupe de la réplication (ou du dispatch en EC)



Commandes usuelles

(mardi)

Ceph health

- ceph -s
- ceph -w
- ceph df
- ceph health detail
- ceph pg dump
- ceph pg X.Y query
- ceph pg dump_stuck inactive

PG

- rados list-inconsistent-obj 12.127 --format=json-pretty
- ceph pg repair
- ceph pg scrub x.y
- ceph pg deep-scrub x.y

OSD

- ceph osd tree : liste tous les OSD
- ceph --admin-daemon /var/run/ceph/ceph-osd.43.asok config show
- ID=XYZ ; ceph osd out $ID ; (ceph osd crush remove osd.$ID) ; ceph auth del osd.$ID ; ceph osd rm $ID

Ceph OSD : gestion manuelle

- le plus simple, sur un cluster fonctionnel : ceph-disk
- ceph-disk <commande>
  - list : liste les disques système et l'éventuel état de disques ceph pré-existants
  - prepare : crée un OSD sur un disque
  - activate : démarre l'OSD si celui ci ne l'est pas déjà
  - activate-all : démarre tous les OSD (@boot par exemple)
  - zap : !! détruit un disque ainsi que ses partitions GPT
- filestore (default): ceph-disk prepare /dev/sdX
- bluestore : ceph-disk prepare --bluestore ...

RBD

- rbd ls un_pool : voir les volumes rbd d'un pool
- rbd snap ls os-patrons/windows2012server-patron : regarde les snapshots d'un volume donné
- rbd children os-patrons/patron-invite-kvm-2@2015-11-16-update2 : liste tous les "enfants" d'un snapshot donné

Operation du cluster

- ceph osd set|unset noout
- ceph osd set noscrub / nodeep-scrub : peut résoudre des pb d'IO
- watch ceph osd pool stats : regarder les IO, opérations sur le tiering
- ceph osd getcrushmap -o crushmap ; crushtool -d crushmap -o crushmap.txt ; edit crushmap.txt ; crushtool -c crushmap.txt -o crushmap.new ; ceph osd setcrushmap -i crushmap.new : editer se crushmap
- ceph osd pool get <nom pool> <param> : vérifier un param d'un pool
- ceph tell osd.* version : vérifier la version des OSD

Workarounds

- rbd feature disable mon_pool/mon_volume deep-flatten,fast-diff,object-map,exclusive-lock : pour rendre les images créées avec des versions récentes de ceph (jewel) compatibles avec le modukes krbd du noyau Linux (le soucis ne se pose pas si on utilise librbd et kvm, par contre)

- ceph tell osd.* injectargs '--bluestore_cache_size 104857600' : pour réduire la taille par défaut de cache bluestore, gigantesque

sources

- http://docs.ceph.com/docs/master/architecture/
- http://recoverymonkey.org/2012/07/26/an-explanation-of-iops-and-latency/
- http://louwrentius.com/category/storage9.html
- https://www.kingston.com/us/ssd/enterprise/best_practices/enterprise_versus_client_ssd
- http://docs.ceph.com/docs/master
- https://thenewstack.io/software-defined-storage-ceph-way/
- Note aux googles : attention, on arrive souvent sur des docs ceph « non master »...