cea irfu

Aperçu des nouveautés Ceph Quincy

(et un peu Reef)

Benjamin MARE - 10/10/2023

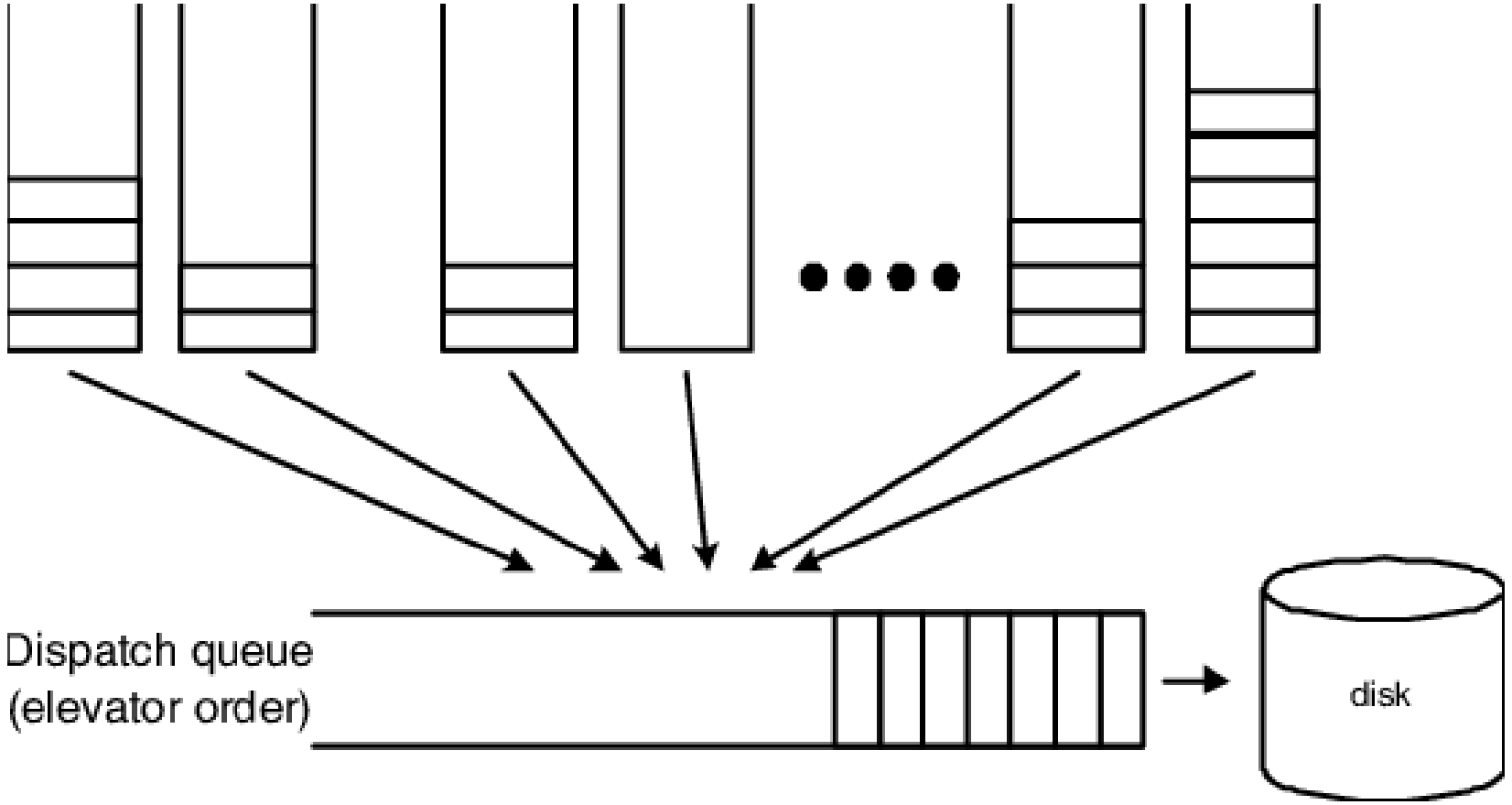
cea

Sommaire

1. Ordonnanceur: mClock
2. OSD
3. CephFS
4. Divers

1. Ordonnanceur: mClock

Rappel : ordonnanceur d'un disque



Wu, Joel & Brandt, Scott. (2023). Qos support in object-based storage devices.

mClock

La QoS dans Ceph est basée sur l'algorithme dmClock, cependant, actuellement l'implémentation est mClock. [1]

➔ **d**mClock est la version distribué de mClock [2]. Il faut donc s'attendre à l'avenir à une évolution de mClock vers dmClock.

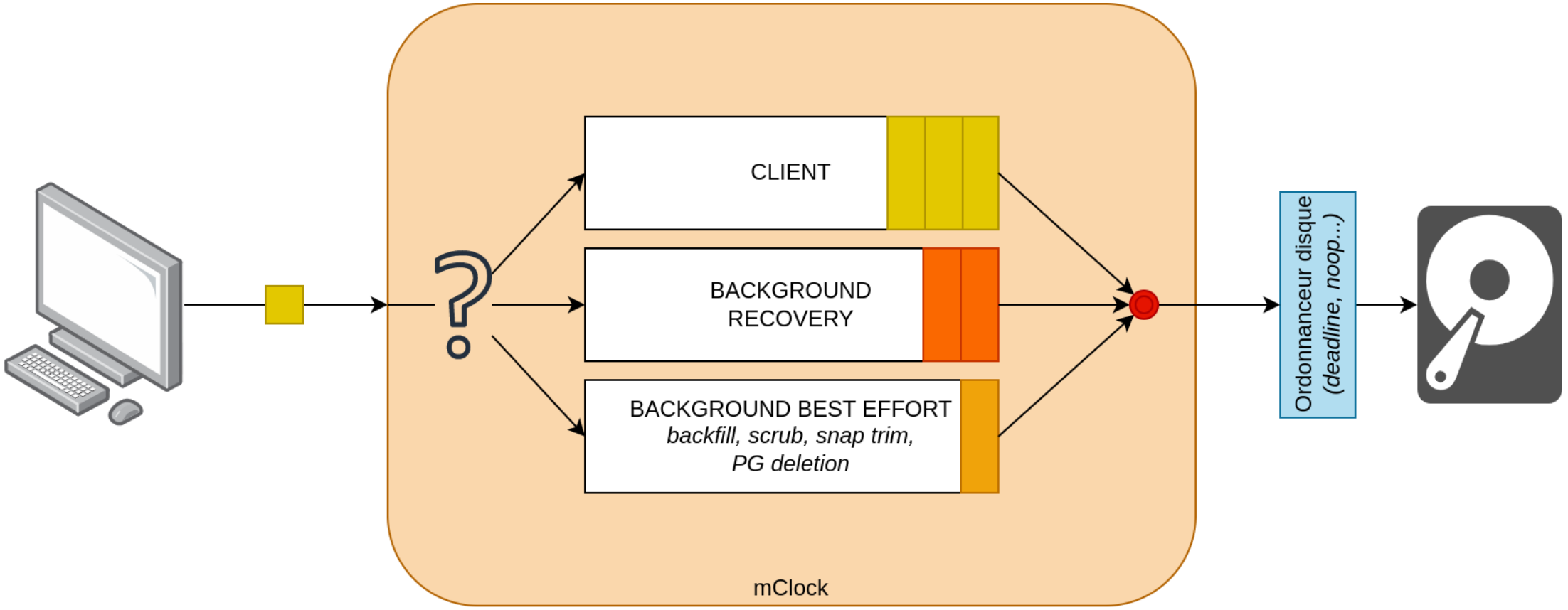
➔ Cela sous entend donc que les différents mClock (aussi bien des différents OSD que les différents sur un même OSD, car il y a un mClock par shard) n'échangent pas. Cela doit pouvoir être la source de comportement qui différent légèrement par rapport à la théorie qui sera présentée.

➔ Si vous avez touché au nombre de shards, cela pourrait avoir un effet plus ou moins voulu sur le comportement final des OSD.

Il remplace l'ancien ordonnanceur WPQ.

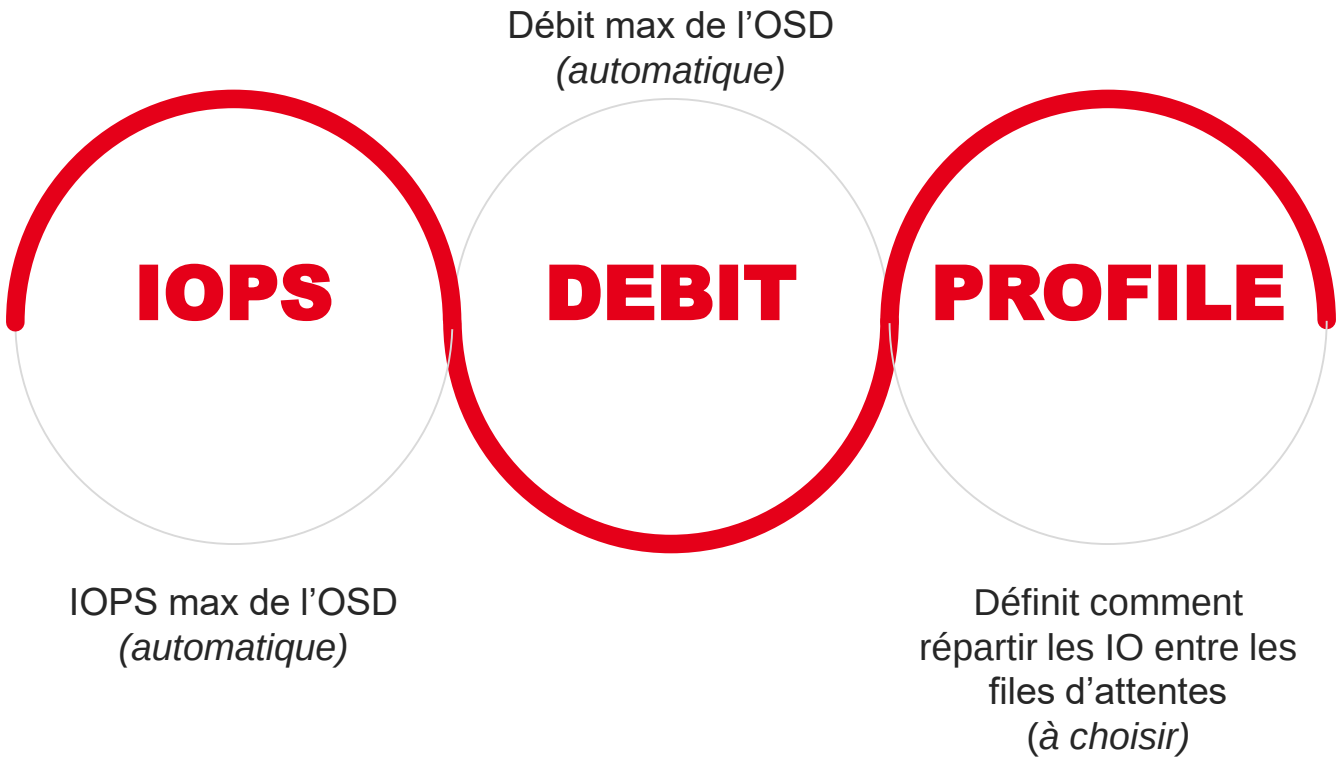
[1] *https://docs.ceph.com/en/quincy/rados/configuration/mclock-config-ref/*
[2] *https://docs.ceph.com/en/latest/rados/configuration/osd-config-ref/#caveats*

mClock


CLIENT
BACKGROUND
RECOVERY
BACKGROUND BEST EFFORT
backfill, scrub, snap trim,
PG deletion
mClock
Ordonnanceur disque
(deadline, noop...)

mClock

L’algorithme nécessite 3 arguments pour fonctionner et mettre en place la QoS :



mClock

Exemple de profile :

Avec un OSD à 315 IOPS

Combien d'IOPS **minimum** ?

Service Type	Reservation	Weight	Limit
client	50%	1	MAX
background recovery	50%	1	MAX
background best-effort	MIN	1	90%

157 IOPS

0 IOPS

⇒ S'il y a beaucoup de recovery & d'IO clients, alors impossible de faire du scrub, suppression de PG...

mClock

Exemple de profile :

Avec un OSD à 315 IOPS

Combien d'IOPS **maximum** ?

Service Type	Reservation	Weight	Limit
client	50%	1	MAX
background recovery	50%	1	MAX
background best-effort	MIN	1	90%

315 IOPS

283 IOPS

⇒ S'il n'y a pas d'autres IO, le client peut utiliser le disque à 100 %. Par contre, impossible d'utiliser le disque à 100 % pour du scrubbing

mClock

Exemple de profile :

Avec un OSD à 315 IOPS

Comment répartir le « surplus » d'IOPS ?

Service Type	Reservation	Weight	Limit
client	50%	1	MAX
background recovery	50%	1	MAX
background best-effort	MIN	1	90%

Tous un poids de « 1 », on répartit équitablement

*⇒ Imaginons qu'on cherche à répartir les 315 IOPS entre background best-effort & client. On obtiendrait alors pour le client : 315*0,5 (**réservation**)+ 315*0,5*0,5 (**le surplus**) = 157,5 + 78,75 ≈ **236** IOPS*
*Background Best Effort aura : 315*0 + 315*0,5*0,5 ≈ **79** IOPS*
Et on respecte la condition LIMIT : 236 ≤ 315

REX – *Avant*

Avec un cron, on changeait les paramètres suivants :

```bash
osd_scrub_priority # priorité de la queue des scrubs
osd_max_scrubs # combien de scrub max en parallèle sur un même OSD ?
osd_recovery_op_priority # priorité de la queue du recovery
osd_recovery_max_single_start # combien d'opération de recovery max en parallèle sur un même OSD ?
osd_max_backfills # combien de backfill max en parallèle sur un même OSD ?
osd_recovery_max_active # combien de requête de recovery max en parallèle sur un même OSD ?
osd_scrub_begin_hour # à quelle heure commencer à autoriser les scrubs ?
osd_scrub_end_hour # à quelle heure finir d'autoriser les scrubs ?
```

REX – « *Hier* »

Avec un cron, on change les paramètres suivants :

bash

osd_scrub_priority # priorité de la queue des scrubs
osd_max_scrubs # combien de scrub max en parallèle sur un même OSD ?
osd_recovery_op_priority # priorité de la queue du recovery
osd_recovery_max_single_start # combien d'opération de recovery max en parallèle sur un même OSD ?
~~***osd_max_backfills*** # combien de backfill max en parallèle sur un même OSD ?~~
~~***osd_recovery_max_active*** # combien de requête de recovery max en parallèle sur un même OSD ?~~
osd_scrub_begin_hour # à quelle heure commencer à autoriser les scrubs ?
osd_scrub_end_hour # à quelle heure finir d'autoriser les scrubs ?
osd_mclock_profile # *quel profile mClock utiliser ?*

REX – « *Hier* »

osd_mclock_profile *⇒ quel profile mClock utiliser ?*

- high_client_ops semble adéquat pour la journée !
- Balanced pour la nuit ?
  - Pas si efficace que ça, changer les autres paramètres a plus d'impact sur le scrub/recovery !

REX – « *Hier* »

En journée, le matin, le Gitlab devient systématiquement inutilisable… Pourquoi ?

Petit coup de iostat et on voit le problème… Les disques sont utilisés à fond !

```
CENSORED:~# iostat -x 1 120 /dev/mapper/ceph--* | awk '{if ($23 >= 80) print $1}' | sort| uniq -c | sort -n
      3 dm-8
      3 dm-9
      5 dm-13
     31 dm-1
     33 dm-3
     33 dm-4
     38 dm-11
     52 dm-2
     71 dm-7
     82 dm-15
     82 dm-16
     87 dm-12
    100 dm-14
    103 dm-0
    114 dm-6
    119 dm-10
```

REX – « *Hier* »

Pourquoi que le matin ? Il y a effectivement des scrubs qui se finissent (à cause d'un « pgs not deep-scrubbed in time»):

```
pgs:     919 active+clean
         26  active+clean+scrubbing+deep
```

Mais d'après la doc ça devrait le faire :

Service Type	Reservation	Weight	Limit
client	60%	2	MAX
background recovery	40%	1	MAX
background best-effort	MIN	1	70%

Si je calcul, ça fait :

- Client = 315*0,6 + 315*0,4*2/3 ≈ **273** IOPS
- Scrub ≈ **42** IOPS

REX – « *Hier* »

Mauvais calcul des IOPS de mes disques ? Probablement en partie vrai, mais le garde-fou Ceph doit limiter la casse. [1]

En regardant le code source Ceph, on remarque alors que la doc Quincy n'est pas bonne ! [2]

Un tableau juste serait:

Service Type	Reservation	Weight	Limit
client	50 %	2	MAX
background_recovery	25 %	1	100 %
Background best-effort	25 %	2	MAX

Ce qui donne :

- Client : 315*0,5 + 315*0,25*0,5 ≈ **197** IOPS
- Scrub : 315*0,25 + 315*0,25*0,5 ≈ **118** IOPS

Ce qui semble déjà mieux expliquer la situation…

[1] *https://docs.ceph.com/en/quincy/rados/configuration/mclock-config-ref/#mitigation-of-unrealistic-osd-capacity-from-automated-test*
[2] *https://github.com/ceph/ceph/blob/v17.2.6/src/osd/scheduler/mClockScheduler.cc#L277*

REX – *Aujourd'hui*

On crée un profile custom selon le tableau suivant :

Service Type	Reservation	Weight	Limit
client	70 %	2	MAX
background_recovery	30 %	1	MAX
Background best-effort	MIN	1	50 %

Pour cela, on touche à *:*

bash

```
osd_mclock_scheduler_{client,background_recovery,background_best_effort}_{res,wgt,lim}
```

*Contrairement à ce que dit la documentation, « res » et « lim » ne sont pas en pourcentage Quincy (c'est le cas pour **Reef** par contre), mais en nombre d'IOPS « brute ».*

mClock

Profile par défaut Quincy et avant: high_client_ops

➔ Reef: passage à balanced, s'il y a recovery alors lui laisser bien plus de place, minimiser les opérations de maintenance de fond.

Service Type	Reservation	Weight	Limit
client	50 %	~~2~~ 1	MAX
background_recovery	~~25 %~~ 50%	1	~~100%~~ MAX
Background best-effort	~~25 %~~ MIN	~~2~~ 1	~~MAX~~ 90%

➔ Reef: Le re-placement (misplaced) a été dé-priorisé par rapport au recovery (données dont la redondance n'est pas celle attendue, il manque un réplica par exemple)

Mon retour: Avec mClock, lors d'un recovery l'impact pour les clients (surtout les VM) a changé. Il va re-changer avec Reef. Je vous invite à vérifier que le recovery d'un simple disque perdu ne rende pas inutilisable le Ceph pour les clients impactés. Dans mon cas, la nuit c'est le cas (mais avec mes modifications, le jour ça semble aller)

Aller plus loin

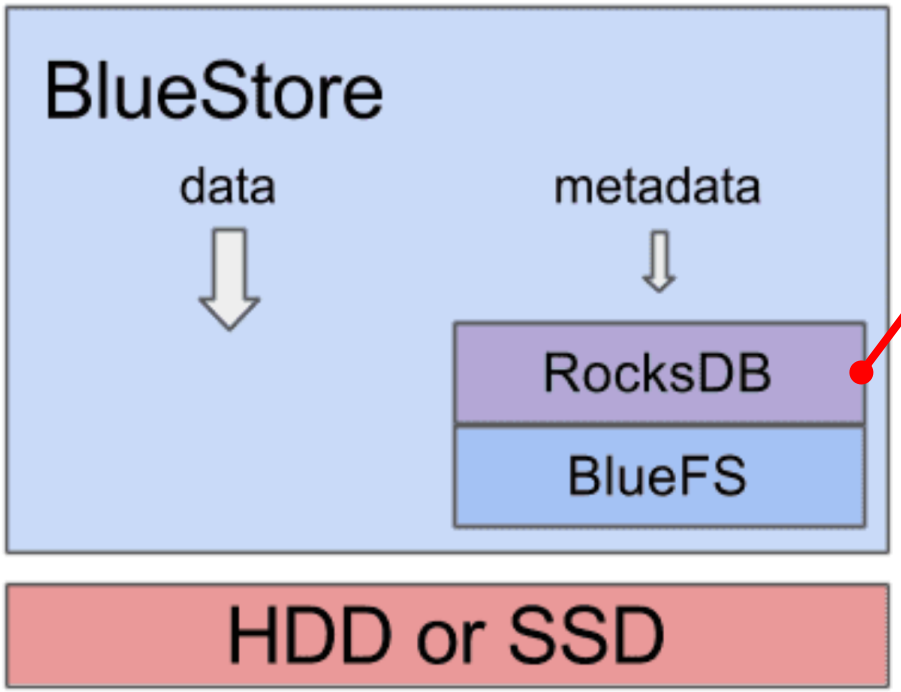
Bon à savoir :

- Les paramètres de backfill et recovery (https://docs.ceph.com/en/quincy/rados/configuration/mclock-config-ref/#steps-to-modify-mclock-max-backfills-recovery-limits)
- Pour basculer d'un profile custom à un builtin, ce n'est pas si simple que ça (https://docs.ceph.com/en/quincy/rados/configuration/mclock-config-ref/#steps-to-switch-from-the-custom-profile-to-a-built-in-profile)
  - Pour passage de custom => builtin, la doc dit :
    1) Changer le profile
    2) Enlever les configurations spécifiques au profile custom
  - Dans les faits, ça fonctionne seulement si je fais dans le sens contraire (??). Sinon c'est comme si le profile custom reste celui appliqué. *(à tester avec Reef pour voir si ça reste le cas)*
- Tester et mettre le bon nombre d'IOPS pour chaque OSD (https://docs.ceph.com/en/quincy/rados/configuration/mclock-config-ref/#steps-to-manually-benchmark-an-osd-optional) *Jamais testé, je suis preneur de retours !*
- mClock prend en compte les paramètres « osd_*_priority » (??)

“Et vous, des soucis, remarques sur mClock ?

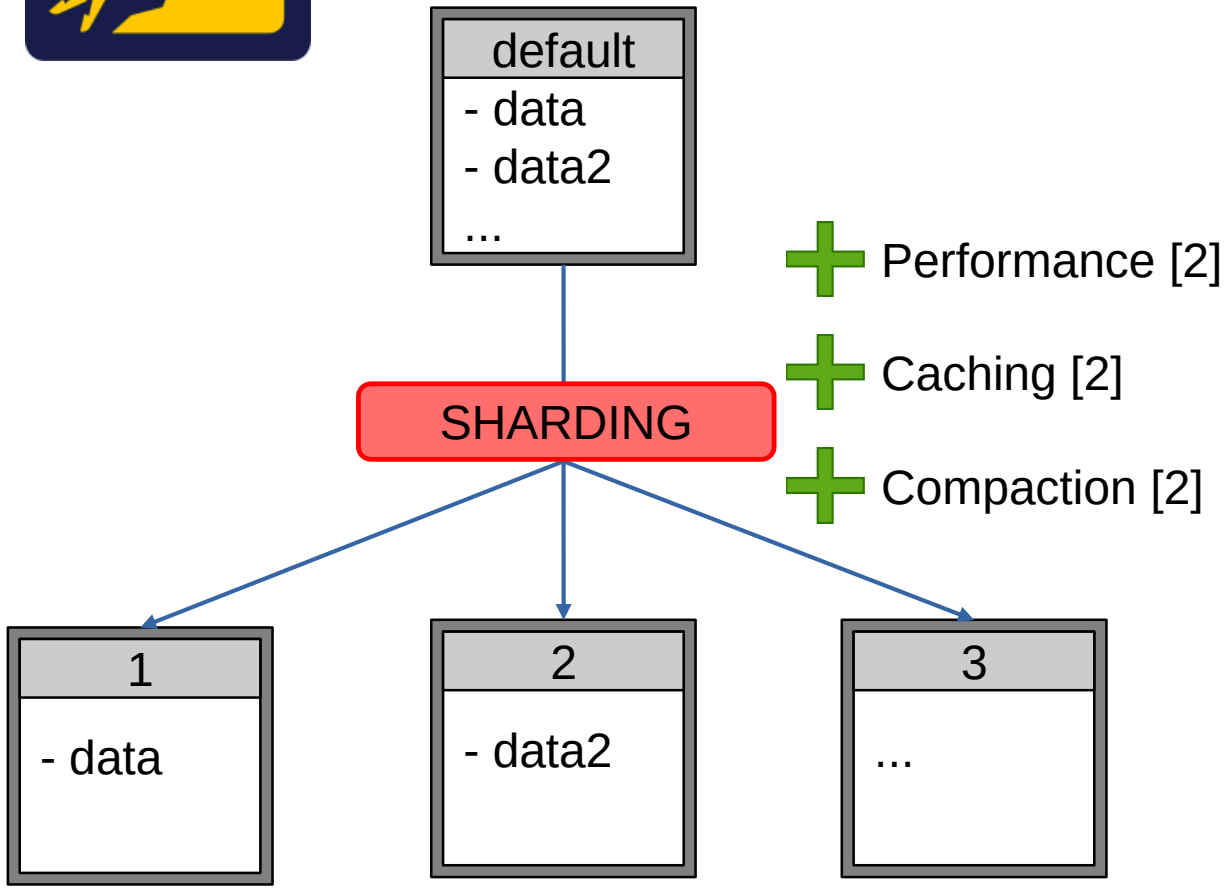
2. OSD

RocksDB sharding



[1]

Base de données clef-valeur



[1] https://ceph.io/en/news/blog/2017/new-luminous-bluestore/

[2] https://docs.ceph.com/en/latest/rados/configuration/bluestore-config-ref/#rocksdb-sharding

RocksDB sharding - cadeau

bash

```
function shard_rocksdb () {
  local osd="$1"
  ceph orch daemon stop $osd
  echo "waiting for $osd to become DOWN"
  # wait until OSD is down
  until ceph osd info $osd  | grep -E '[[:space:]]down[[:space:]]' >/dev/null ; do
    sleep 10
  done
  # We need an extra wait for the lock to disappear
  sleep 10
  echo "$osd DOWN, sharding rocksdb"
  ceph-bluestore-tool --path /mnt/$osd --sharding="m(3) p(3,0-12) O(3,0-13)=block_cache={type=binned_lru} L P"
reshard
  echo "sharding rocksdb for $osd DONE"
  ceph orch daemon start $osd
  until ceph osd info $osd  | grep -E '[[:space:]]up[[:space:]]' >/dev/null ; do
    sleep 10
  done
}
export -f shard_rocksdb
# Paralellize 2 OSD at the same time. We could try 3 maybye ?
ls -d /mnt/osd* |  xargs -n 1 basename | xargs -P 2 -I {} bash -c "shard_rocksdb {}"
```

RocksDB sharding

bash

```
function shard_rocksdb () {
  local osd="$1"
  ceph orch daemon stop $osd
  echo "waiting for $osd to become DOWN"
  # wait until OSD is down
  until ceph osd info $osd  | grep -E '[[:space:]]down[[:space:]]' >/dev/null ; do
    sleep 10
  done
  # We need an extra wait for the lock to disappear
  sleep 10
  echo "$osd DOWN, sharding rocksdb"
  ceph-bluestore-tool --path /mnt/$osd --sharding="m(3) p(3,0-12) O(3,0-13)=block_cache={type=binned_lru} L P"
reshard
  echo "sharding rocksdb for $osd DONE"
  ceph orch daemon start $osd
  until ceph osd info $osd  | grep -E '[[:space:]]up[[:space:]]' >/dev/null ; do
```

Si on en croit la variable « bluestore_rocksdb_cfs » qui définit le sharding lors de la création d'un nouvel OSD, il faudrait refaire la manipulation lors du passage en ***Reef****.*
« m(3) p(3,0-12) O(3,0-13)=block_cache={type=binned_lru}
*L=**min_write_buffer_number_to_merge=32** P=**min_write_buffer_number_to_merge=32** »*

RocksDB defaults - *Reef*

Suite à des benchmark, on devrait voir une amélioration des performances de RocksDB (metadata OSD).

➔Jusqu'à 13.59% d'augmentation d'IOPS 4k random write pour du RGW.

Si vous avez touché à « bluestore_rocksdb_options », lisez le blog ci-dessous. Peut-être devriez vous à nouveau modifier cette variable.

➔ Pour appliquer une modification de ce paramètre, il faut redémarrer l'OSD (ce qui aura lieu lors du sharding / lors de la mise à jour).

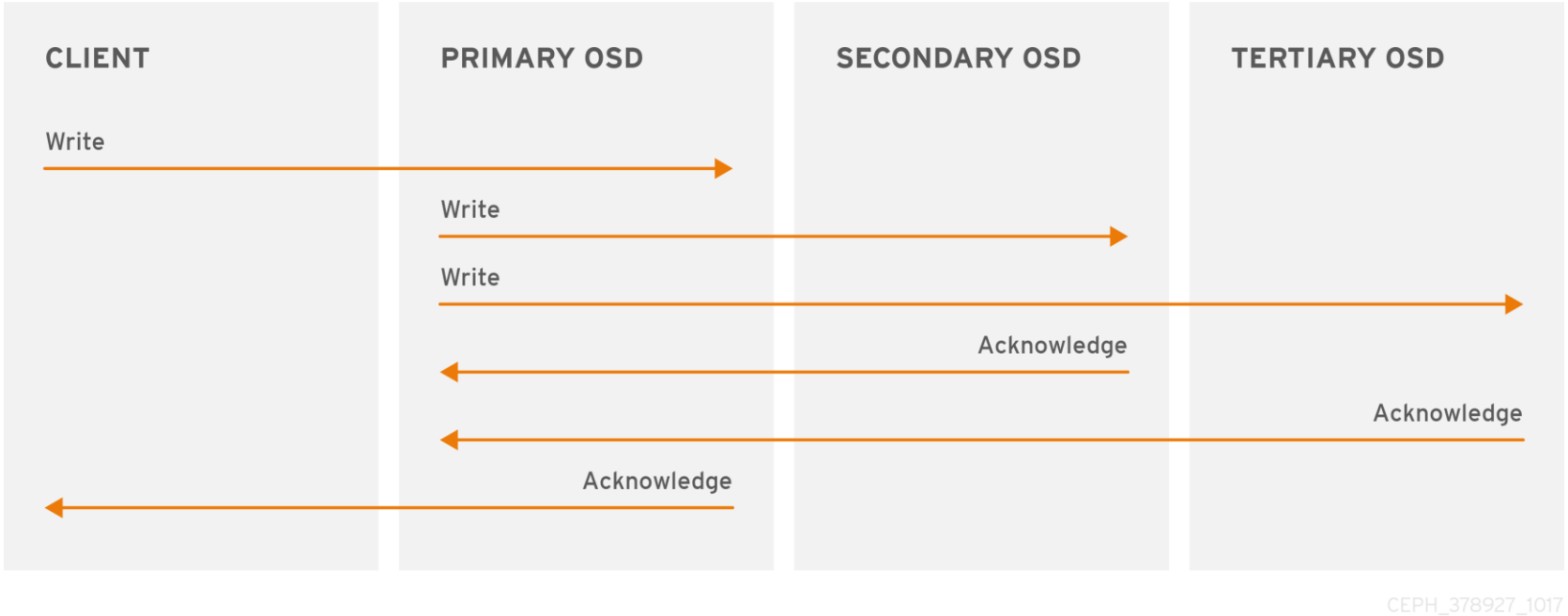
D'après ma compréhension du blog et de la note de mise à jour, cette option part du principe que le sharding a été fait sur les OSD. Pour en tirer profit il faut donc veiller à avoir fait le sharding.

[1] https://ceph.io/en/news/blog/2022/rocksdb-tuning-deep-dive/

[2] https://docs.ceph.com/en/latest/releases/reef/#v18-2-0-reef

Read Balancing

Comment se passe une écriture (et lecture) sur un pool réplica ?

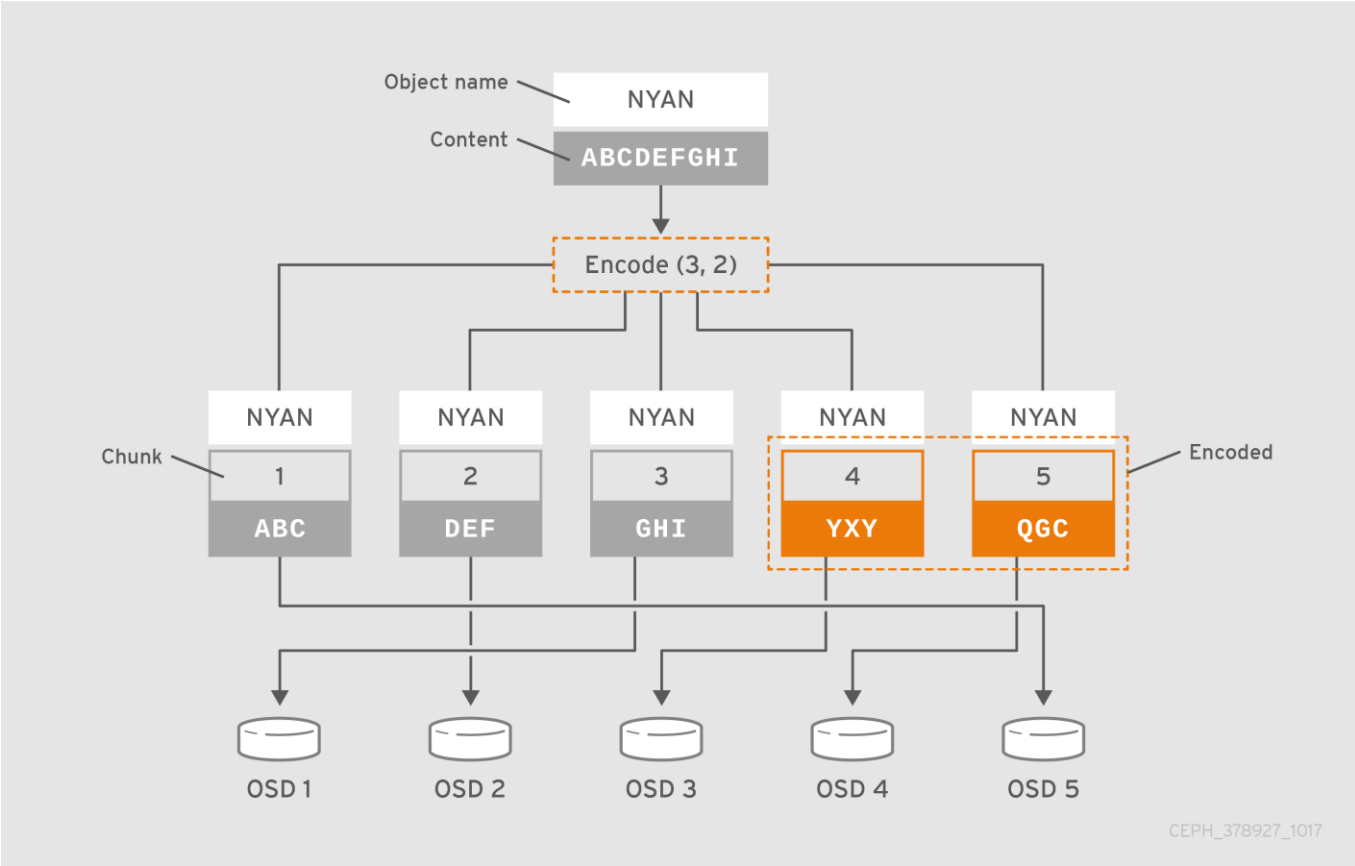


L'OSD primaire a donc plus de traitements à faire que les autres. Il est le **seul** à faire les lectures.

[1] https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/4/html/architecture_guide/the-core-ceph-components

Read Balancing

Comment se passe une écriture (et lecture) sur un pool EC ?



L'OSD primaire a donc plus de traitements à faire que les autres. Il doit faire des opérations pour reformer les données.

[1] https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/4/html/architecture_guide/the-core-ceph-components

Read Balancing

Optimiser qui est l'OSD primaire est surtout utile si:

- Vous avez des disques de tailles différentes
- Vous avez des disques de technologies différentes (SATA, SAS, NVME…)

On peut espérer une optimisation de 15% dans certaines situations [1] (surtout replicas). Mais dans l'idée, on répartit mieux l'utilisation CPU pour l'EC (impact à mesurer cependant).

Avant:

```bash
ceph osd primary-affinity <osd-id> <weight>
```

Depuis Reef:

```bash
ceph osd getmap -o om
osdmaptool om --read out.txt --read-pool <pool name> [--vstart]
source out.txt
```

[1] https://docs.ceph.com/en/quincy/rados/operations/crush-map/#primary-affinity

[2] https://docs.ceph.com/en/reef/rados/operations/read-balancer/

Read Balancing - *REX*

Ayant mis (via un script) des primary affinity sur l'ensemble de mes OSD, en raison des tailles variés, je vais devoir tester cette fonctionnalité et voir comment ils interagissent ensemble…

Pour l'instant c'est du offline, mais puisqu'ils comptent passer en online, ça peut valoir le coup de creuser…

Quelqu'un a testé ? Le mapping « manuel » de Reef overwrite le primary affinity ?

Des remarques ?

- Le RockDB sharding a-t-il plus d'effet avec les NVME que pour les HDD ? Ou au contraire ?
- Avez-vous une infrastructure non « homogène » qui vous a amené à regarder les OSD primaires ?

3. CephFS

Multi MDS (entre autre) par hôte

Actuellement, le MDS est mono thread & « CPU-bound ». Il est possible de mettre plusieurs MDS sur une même machine (je ne m'y suis jamais risqué).

Mais Cephadm permet maintenant de déployer plusieurs MDS par serveur [1]. On peut le faire pour:

- Les MDS
- Les RGW
- Les MGR *(utile seulement pour les environnement de test avec un seul nœud ?)*

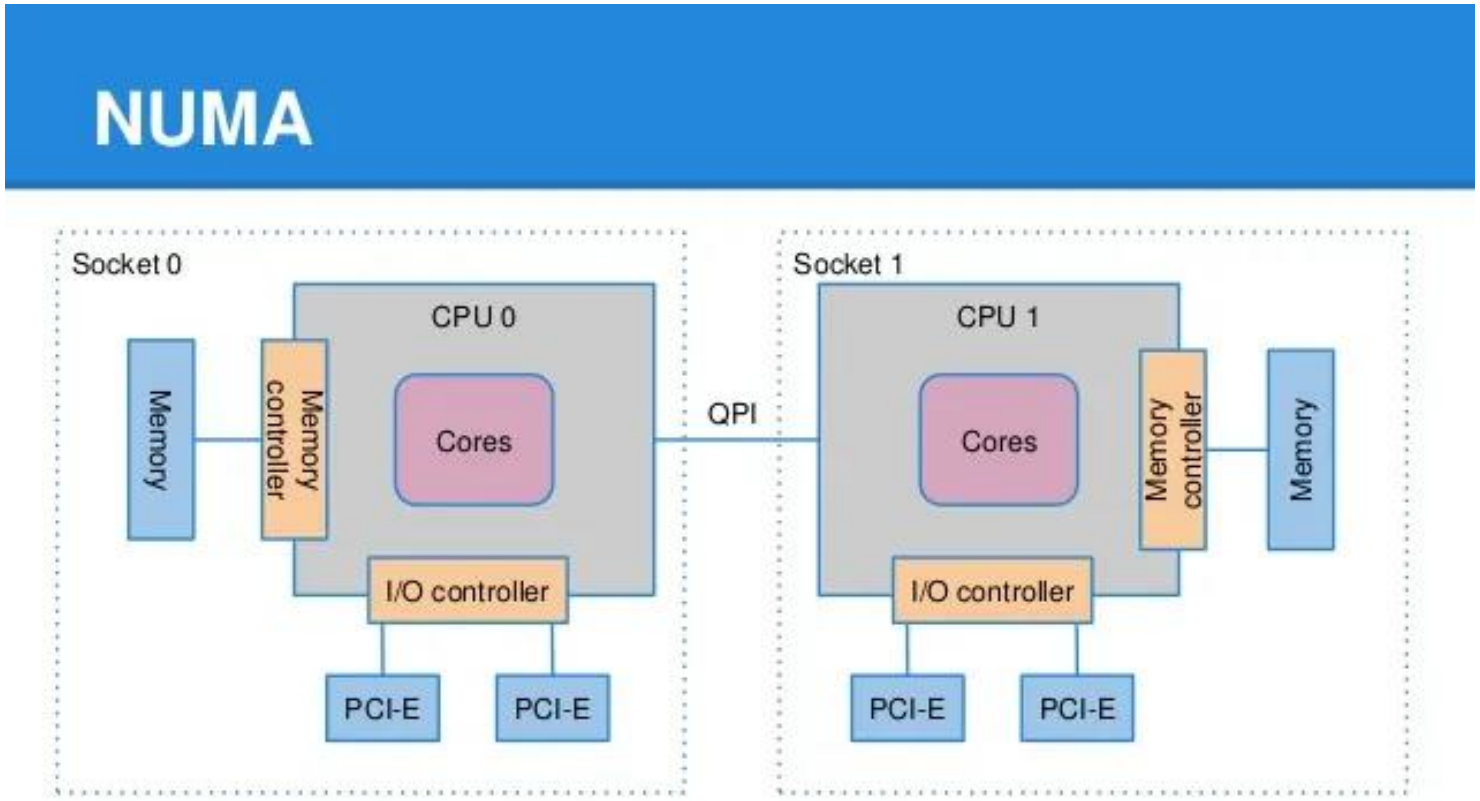
La doc précise qu'un gain en performances est envisageable pour les RGW & MDS.

YAML

```yaml
service_type: rgw
placement:
  label: rgw
  count_per_host: 2
```

[1] https://docs.ceph.com/en/quincy/cephadm/services/#co-location-of-daemons

Numa pinning MDS- *Rappel*



[1] *https://fatmin.com/2016/06/10/numa-node-to-pci-slot-mapping-in-red-hat-enterpise-linux/*

Numa pinning MDS

On peut donc actuellement pin:

- Les OSD
  - Automatique pour les NVME si on détecte l'interface réseau sur le bon noeud NUMA.
- Les MDS
- Les RGW

D'après Redhat, permet de rendre les performances plus déterministes. [1]

De préférence, pin sur la socket avec la carte réseau de connecté.

bash

```
osd_numa_node # Nœud numa de l'OSD
osd_numa_auto_affinity # Si stockage et carte réseau sur même nœud numa, mettre automatiquement le nœud numa
osd_numa_prefer_iface # Préférer l'IP sur le même nœud numa que le stockage
rgw_numa_node # Nœud numa du RGW
mds_numa_node # Nœud numa du MDS
```

[1] https://access.redhat.com/documentation/fr-fr/red_hat_openstack_platform/11/html/hyper-converged_infrastructure_guide/resource-isolation#resource-isolation-numa

Activation des fonctionnalités MDS

Auparavant:

```bash
cephf fs set cephfs min_compat_client nautilus
```

Maintenant, on peut forcer en fonction du besoin chaque fonctionnalité:

```bash
cephf fs required_client_features <fs name> [add|rm] <feature_name>
```

Attention à ne pas forcer une fonctionnalité disponible que pour les clients FUSE si vous avez des clients noyau !

Montage asynchrone MDS

Seulement pour le montage noyau. Permet la création et suppression de fichier dans cephfs de façon asynchrone.

Amélioration des performances des commandes *rm*, *tar* ou encore *rsync*.

Pour l'activer:

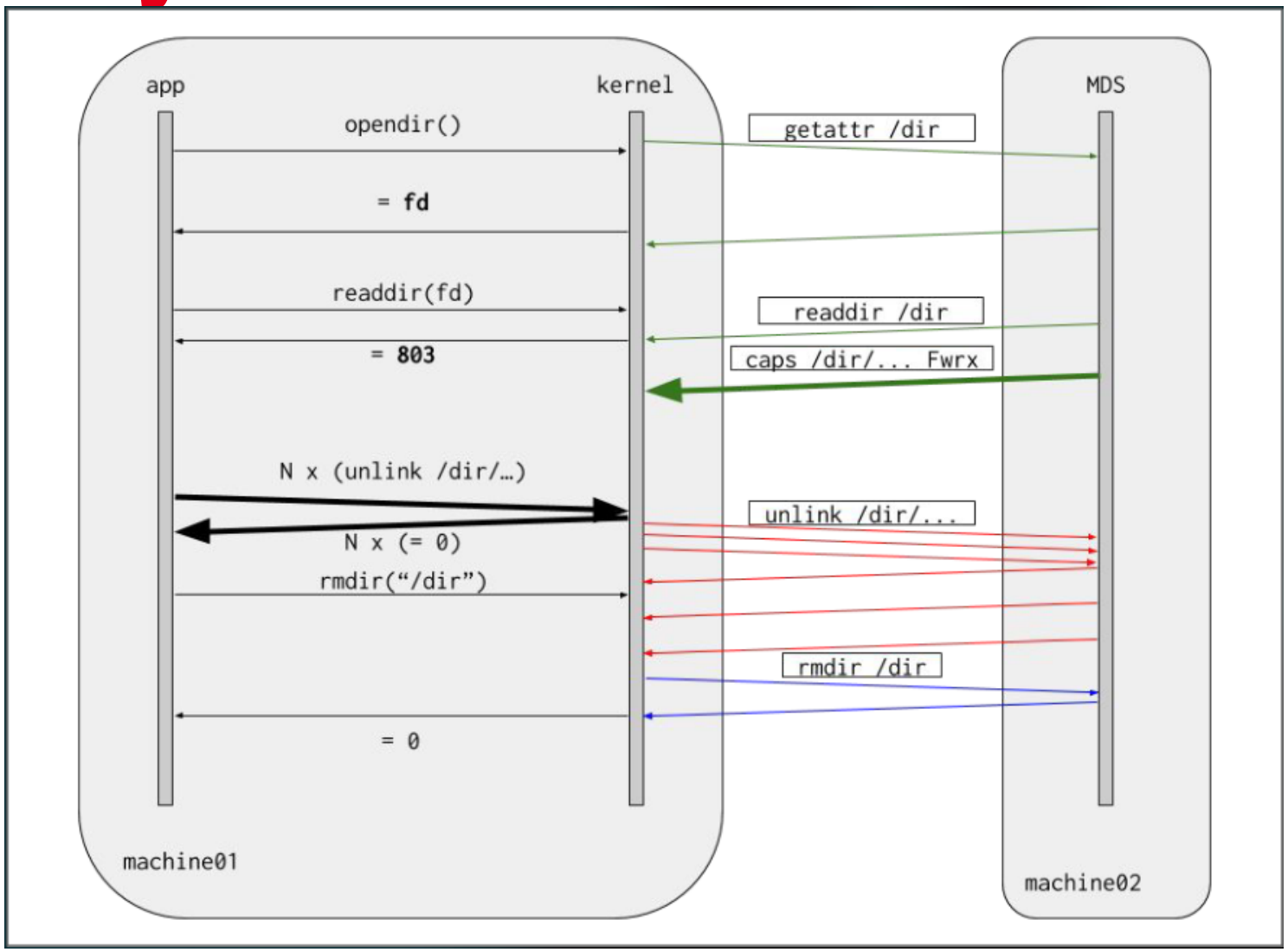
bash

```
mount –o nowsync –t ceph …
```

Cette option devient l'option par défaut avec le noyau 5.16.

[1] https://docs.ceph.com/en/latest/rados/operations/user-management/#namespace

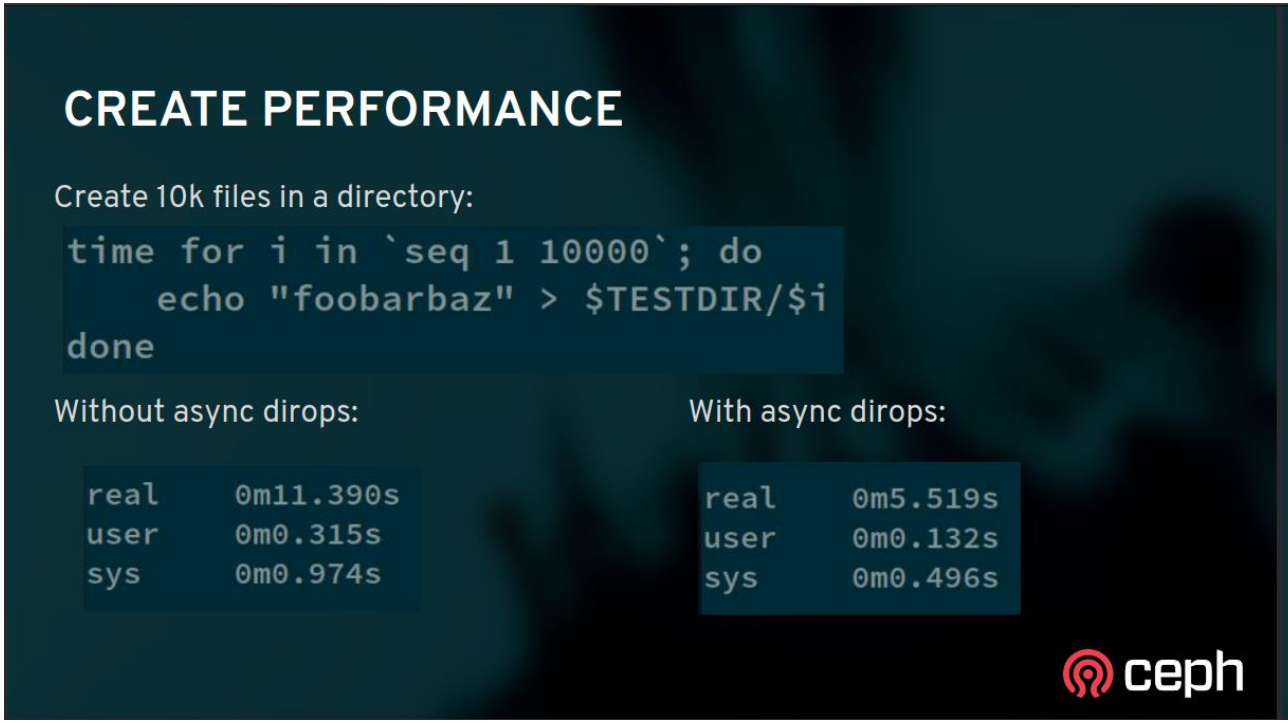
Montage asynchrone MDS

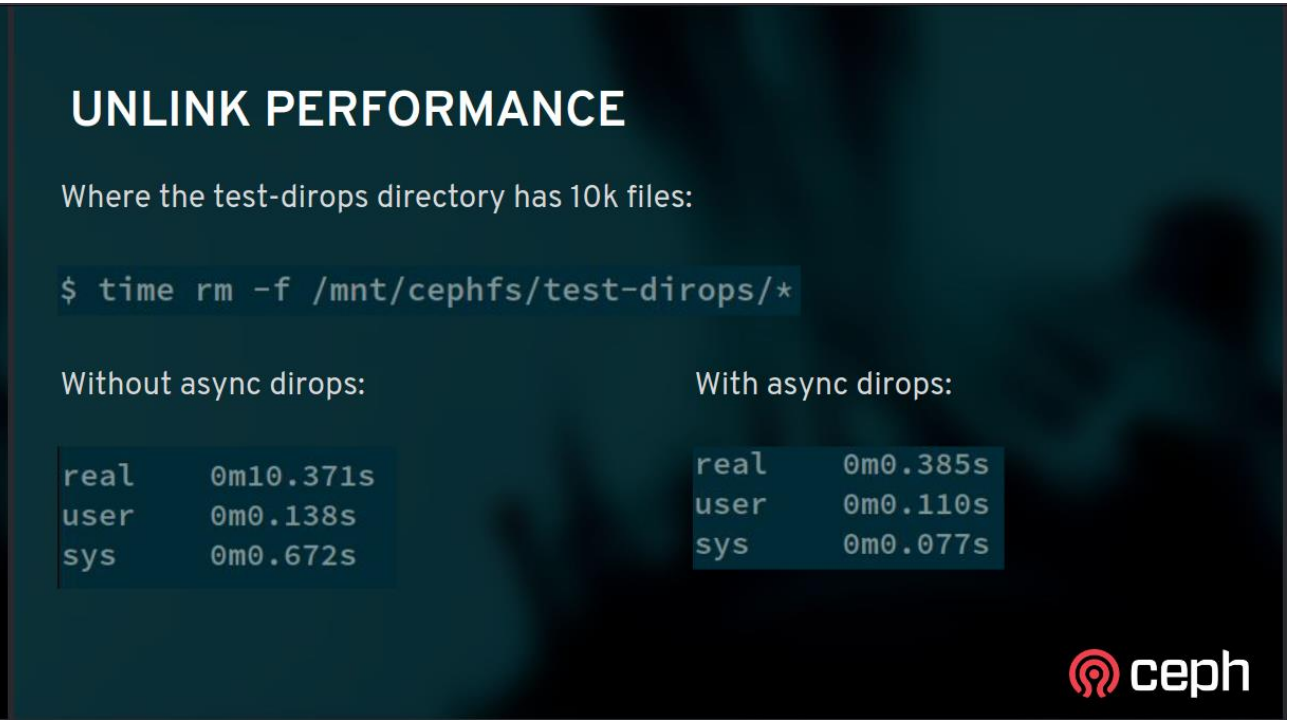


[1] https://www.usenix.org/sites/default/files/conference/protected-files/vault20_slides_layton.pdf

Montage asynchrone MDS

Quelques exemples donnés par Redhat :





[1] https://www.usenix.org/sites/default/files/conference/protected-files/vault20_slides_layton.pdf

Montage asynchrone MDS

Quelques exemples donné par Redhat :

[1] https://www.usenix.org/sites/default/files/conference/protected-files/vault20_slides_layton.pdf

“Avez-vous testé certaines de ces nouveautés ?

- Le multi MDS/RGW est-il un vrai gain ?
- Le numa pinning a t-il aidé ?

4. Divers

Cache MGR

Les modules MGR partagent un cache qui doit être activé.

Il est conseillé de l'activer dès 500+ OSD ou 10k+ PG.

On a moins d'OSD & de PG, mais on l'a quand même activé (et c'est utile 2/3 du temps):

bash

```
$ ceph daemon mgr.${MGR} perf dump
[…]
 "mgr": {
        "cache_hit": 916460,
        "cache_miss": 574250
    },
[…]
```

Après une minute d'attente, augmentation de 2 cache_hit & 4 cache_miss. Avec un OSDMAP de ~200Ko le MGR ne *semble* effectivement pas surchargé.

[1] https://docs.ceph.com/en/quincy/mgr/administrator/#performance-and-scalability

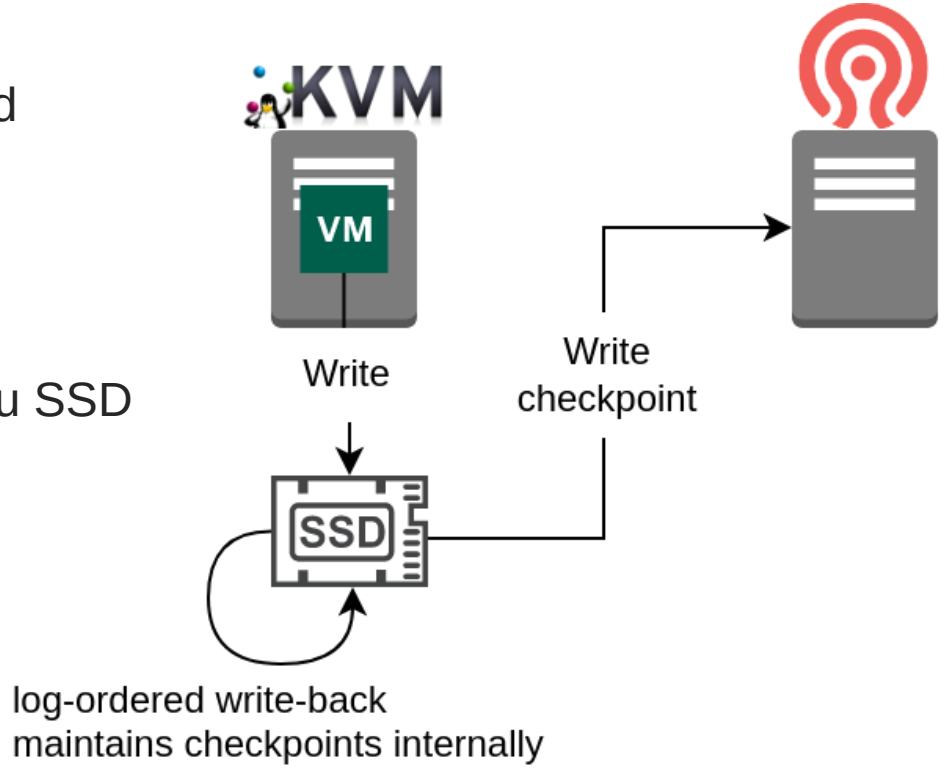
RBD Persistent Write Log (PWL) cache

Un moyen de profiter d'un disque local rapide pour les VM (PMEM ou NVME) et ne rien perdre au cache en cas de coupure impromptue.

Probablement de taille plus importante que le cache librbd (32Mio par défaut)

Sans en avoir trop ce qui pourrait devenir embêtant (KRBD avec writeback)

Si bien dimensionné, on profite juste des performances du SSD local. Bon moyen d'allier performances / sûreté / coût ?



[1] https://docs.ceph.com/en/quincy/rbd/rbd-persistent-write-log-cache/#rbd-persistent-write-log-cache

Le Cache-tiering c’est finis !

Le cache-tiering passe en deprecated pour Reef.

J’aurais aimé une solution de tiering du stockage (de base mettre sur du « chaud », puis basculer sur du « tiède/froid » si inutilisé). Cela ne semble pas la priorité / l’objectif de Ceph ?

➔ J’en avais auparavant pour faire du RBD avec de l’Erasure Coding. Mais puisque c’est maintenant possible de faire du presque full EC j’ai détruit ces pools il y a plusieurs années.

[1] https://docs.ceph.com/en/quincy/rbd/rbd-persistent-write-log-cache/#rbd-persistent-write-log-cache

Avez-vous testé certaines de ces nouveautés ?

- Êtes vous impacté par la fin du cache-tiering ? Quelle est votre solution ?
- Avez-vous testé le PWL ? Votre avis sur son utilité ?

irfu

Merci !

Benjamin MARE

benjamin.mare@cea.fr

01 69 08 99 25