

Stockage distribué : retour d'expérience avec CEPH

Yann Dupont

DSIN / Université de Nantes
2, rue de la houssinière
44322 Nantes

Résumé

L'Université de Nantes s'est penchée depuis 2005 sur la haute disponibilité des systèmes, essentiellement pour des raisons pratiques : il fallait trouver un moyen d'assurer un service de qualité malgré les incidents à répétition dans ses salles machines.

Au fil des ans, différentes plates-formes ont été déployées pour s'acquitter de cette mission, assurant la haute disponibilité des services, mais faisant l'impasse sur la question du stockage. Celui-ci est toujours resté architecturé autour de SAN dont les nombreuses baies de stockage centralisées constituent pourtant un point unique de défaillance. Nos critères de choix étant fiabilité, simplicité, performance et coût, aucun autre système de stockage à disponibilité plus élevée ne répondait à ce cahier des charges.

CEPH (logiciel open source), faisait partie des candidats surveillés depuis quelques années. Il dispose de nombreux aspects techniques particulièrement originaux et a suffisamment mûri en 2012 pour engager une étape de pré-production, puis passer en production début 2013.

Ce retour d'expérience reviendra sur les différences fondamentales entre les solutions de stockage traditionnelles précédemment déployées et un système massivement distribué comme CEPH. Il détaillera également les gains, limitations et risques associés. Les quelques mésaventures subies en phase de pré-production, maintenant bien comprises, ont été mises à profit pour trouver des solutions. Le syndrome de « tous les œufs dans le même panier » est ainsi vaincu en déployant plusieurs clusters indépendants reposant sur une architecture virtuelle commune.

Mots-clefs

Stockage distribué, SAN, CEPH, open source, clusters, haute disponibilité, LINUX

1 Stratégie de stockage

La DSIN¹ de l'Université de Nantes a choisi en 2003 de gérer ses données en séparant nettement les aspects stockage (sur du matériel dédié) et traitement (sur des serveurs). La mise en place de plusieurs réseaux SAN² fibre channel étendus sur l'agglomération en a été la conséquence directe.

Conjointement à la virtualisation massive de nos services, qu'elle a contribué à rendre aisée, cette infrastructure dédiée de stockage, performante, extensible et fiable a été la base technique sous-jacente. Elle s'est aussi avérée économique grâce à l'usage de petites baies de stockage (16 à 24 disques), tolérantes à la panne³, capables d'un taux d'entrées/sorties élevé et pourtant peu onéreuses car simples et fonctionnellement pauvres, sans redevance logicielle. Nos serveurs fonctionnant tous sous LINUX, (les serveurs WINDOWS sont virtualisés) ils disposent de toutes les solutions pour gérer eux même efficacement ces services avancés (snapshot, synchronisation, déplacement à chaud). Nous sommes agnostiques quant aux baies de stockage déployées (quatre constructeurs différents au fil du temps et des marchés publics).

La croissance du volume géré a simplement consisté à multiplier les baies au fur et à mesure des besoins, ce qui a également augmenté la capacité totale de traitement de façon linéaire : le nombre de canaux et de contrôleurs dédiés croît en même temps que le nombre d'axes. Il est aussi possible de spécialiser des baies ou des parties de baies (type de disques, de RAID, taille de cache) lorsque des usages spécifiques sont envisagés, ou au contraire, agréger virtuellement

1. Direction Des Systèmes d'Information et du Numérique.

2. Storage Area Network. Réseau dédié spécialisé pour le stockage, utilisant des protocoles spécifiques historiquement incompatibles avec l'*Ethernet*, tels que le *Fibre Channel*, ou compatible tels que l'*iSCSI*, l'*AOE* (Ata Over Ethernet), le *FcoE* (Fibre Channel Over Ethernet)...

3. Contrôleurs doubles dotés de nombreux canaux *fibre channel* permettant des chemins indépendants via deux fabricques SAN disjoints.

plusieurs baies en un seul ensemble logique. Les données sont accessibles au travers d'un périphérique de type bloc à accès exclusif, exactement comme un disque dur local. Pour les besoins d'accès multiples et concurrents, des services de système de fichiers en réseau (*NFS*, *CIFS*) couplés aux annuaires *LDAP* sont déployés et font office d'interfaces *NAS* spécialisées.

La construction de cette solution hybride multipliant les éléments permet d'isoler des flux et garantir une qualité de service réelle, nécessité due à la diversité des besoins gérés. Ainsi, une charge induite par des TP d'informatique ne perturbent pas les performances du courrier électronique du personnel, les dossiers (*homedirs*) des étudiants et le courrier électronique étant stockés sur des baies différentes (empruntant des canaux séparés).

Il a ainsi été possible d'accompagner toutes les évolutions de besoins de stockage de l'université et de réaliser de très nombreux projets fédérateurs d'importance (stockage de fichiers pour les étudiants, personnels et équipes de recherche, messageries, espaces collaboratifs, bases de données, bibliothèques vidéos, support de projets de recherche, etc) tout en maîtrisant les investissements.

1.1 Haute disponibilité et stockage centralisé

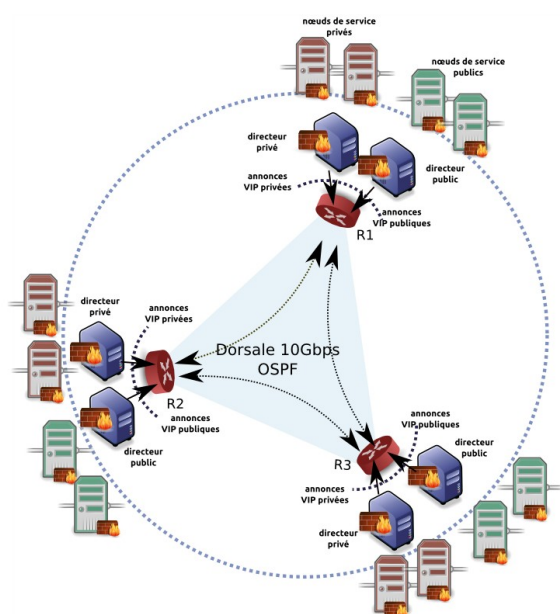


Figure 1 - Plate-forme à haute disponibilité de l'Université de Nantes

patrimoine de l'établissement, le risque majeur était avéré ; en cas de sinistre important (incendie, etc.), outre l'interruption de service, c'était surtout la survie des données qui était en jeu. La problématique était alors partiellement adressée par des sauvegardes déportées et des synchronisations de volume graduellement mises en place, mais une solution plus globale devait être envisagée, d'autant que l'aspect sûreté des données n'était pas le seul point à considérer.

1.2 La fin d'une logique

L'augmentation incessante du volume des espaces de stockage (plus de 500 To utiles, répartis sur une trentaine de baies de disques, avec une utilisation parfois peu rationnelle) a augmenté la complexité de manœuvre de cette plate-forme sans que des moyens humains supplémentaires dévolus à cette tâche ne soient dégagés. Les pressions budgétaires toujours plus fortes invitent aussi à une convergence du monde du stockage et celui du réseau traditionnel.

De façon assez claire, le constat est que nous sommes allés au bout de notre logique décidée il y a déjà longtemps. Il était devenu nécessaire d'évoluer progressivement vers une technologie distribuée, résistant mieux au facteur d'échelle, plus tolérante aux pannes et mieux intégrée.

4. *Ou non politiquement correct* : « salles non initialement prévues à cet usage, à l'instabilité chronique, sources continues de soucis en tout genres ».

5. Pas forcément plus fiables. La construction d'un datacenter viendra clore ce chapitre malheureux début 2015.

6. De nombreuses pannes, vécues comme autant de tests grandeur nature pour la plateforme, permettent d'assurer que le terme n'est pas galvaudé.

2 Évoluer en douceur

Notre recherche s'est orientée vers un remplacement quasi transparent de l'ancienne plate-forme, retenant ses qualités et gommant ses défauts ; la nouvelle solution se devant de rester compatible avec les services déployés (compatibilité POSIX totale requise) et les logiciels de virtualisation⁷ existants, être simple à utiliser, sans nécessiter de logiciels ou pilotes propriétaires (voire exotiques) et bénéficier d'une intégration directe dans les distributions LINUX⁸.

Pour s'intégrer à notre plan continu d'activité, l'architecture du stockage doit être structurée similairement à la plate-forme d'hébergement à haute disponibilité et tirer parti de nos trois points de présence géographique de façon multi-active et sans point unique de faiblesse. En cas de panne d'une salle machine, les deux salles restantes doivent assurer le service sans intervention humaine, tout comme le retour à l'état nominal qui doit s'effectuer de façon automatique. Les données doivent donc être placées de façon intelligente.

La solution doit être capable d'un passage à l'échelle important (au delà du pétaoctet), tout en simplifiant l'administration au quotidien (mise à disposition de nouveaux volumes), en optimisant le volume de données utilisé, et autant que possible, sans dégrader les performances, préserver une qualité de service et être économiquement viable.

L'ensemble de ces pré-requis représente une belle lettre au père Noël.

2.1 Recherche de solutions adéquates

Eu égard aux contraintes économiques et de volumétrie, une solution propriétaire a rapidement semblé hors de portée ; toutefois, la recherche d'une réponse au travers des solutions *open source* a pris des allures de quête du Graal.

Généralement, l'offre se structure autour de système de fichiers distribués, de type NAS, permettant un accès concurrent et multiple, autorisant une bonne souplesse de fonctionnement, mais pas forcément des performances élevées (complexité de gestions des verrous, synchronisation des méta-données...). *LUSTRE*[2], *GLUSTERFS*[3], *HEKAFS*[4], *MOOSEFS*[5], *ROZOFs*[6] et *CEPHFS* font partie de solutions faisant l'objet de veille technologique depuis des années. Elles ont chacune leur intérêt propre, mais présentaient aussi des problèmes de maturité, de fonctionnalités ou de performances, et aucune n'avait la dimension globale recherchée.

Il s'avère que pour une bonne partie de nos applications hébergées, un accès exclusif est suffisant. Il a le mérite de se calquer sur notre ancien mode de fonctionnement (mise à disposition d'un *LUN* pour chaque nouveau besoin), de rester simple, d'avoir moins de latences (moins de serveurs traversés) et donc d'être globalement plus performant.

Mais les solutions orientées « bloc » ne sont pas légion. *OCFS2*[7], *GFS2*[8] sont des systèmes de fichiers en grappe (de type « cluster »). Ils permettent un accès multiple de type actif/actif pour de multiples serveurs partageant une ressource commune, telle une baie *SAN*. En soi, cela n'adresse pas la problématique de la sécurisation des données, sauf si la baie sous-jacente est répliquée de façon synchrone en deux lieux distants, ou en utilisant un système de réplication de périphériques blocs, tel *DRBD*⁹[9] pour arriver à ses fins.

CEPH[10], quant à lui, autorise plusieurs modes de fonctionnement, dont un en mode bloc, constituant en cela une de ses originalités et ayant de fait, suscité tout notre intérêt. Nous l'avons surveillé pendant quelques années et attendu patiemment sa maturation.

3 ceph

Ce logiciel open source (Licence LGPL) fait suite aux travaux de Sage Weil durant sa thèse[11]. Il fonctionne actuellement uniquement sous *LINUX*¹⁰. Il est composé d'une partie client, jouissant d'une intégration native dans le noyau standard depuis mars 2010, et d'une partie serveur, plus complexe, basée sur un ensemble de démons fonctionnant en espace utilisateur. Depuis 2011, une société, (*INKTANK*), a été créée pour assurer le support et rétribuer les développeurs.

Il est temps plonger dans les entrailles de *CEPH*, en se limitant au strict nécessaire pour comprendre notre implémentation.

3.1 Concepts

7. Utilisation de libvirt avec les pilotes KVM et LXC, avec pour chacun un accès direct à des volumes et non un fichier image.

8. La compatibilité avec une version de Debian doit être assurée.

9. Mode actif/actif possible à partir de la version 8, mais qui n'est pas foncièrement conseillé sur leur site.

10. Des portages sont envisagés sous *BSD*, pour d'autres OS une utilisation indirecte à travers des couches *CIFS*, *NFS* voire *ISCSI* reste possible.

À la base de *CEPH* se trouve *RADOS*, un système de stockage d'objets répartis, à l'intégrité auto régulée, géré par des serveurs indépendants.

Une API ainsi qu'une librairie, *LIBRADOS*, permet à une application d'interagir directement avec ce stockage d'objets. *CEPH* utilise cette librairie pour fournir de base, trois services principaux à l'utilisateur :

Figure 2 - : Architecture de CEPH

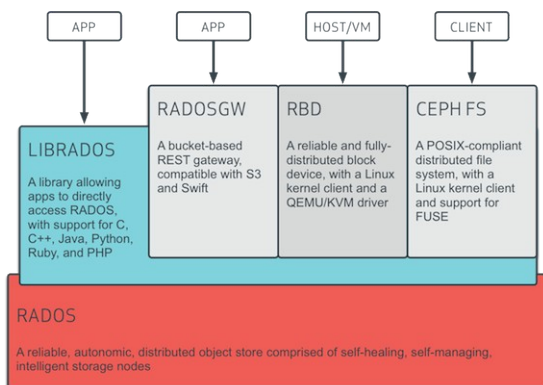


Figure 3 - : Architecture de CEPH

- une interface directe au stockage objet, *RADOSGW*, compatible avec les protocoles de stockage *S3* et *SWIFT*¹¹,
- une interface de type block, nommée *RBD*, directement utilisable par un client *LINUX*. Il s'agit d'un véritable *SAN* virtuel,
- une interface de type « système de fichiers en réseau », appelée *CEPHFS*, utilisable par un client *LINUX*. Il s'agit d'une interface de type *NAS*.

Ici réside une des originalité de *CEPH*. Sa versatilité est forte, puisqu'il peut être utilisé au sein de projets spécifiquement développés pour du stockage massif de données, mais aussi en remplacement direct de services de stockage traditionnels, quelle que soit l'ancienne technologie utilisée. C'est cet usage

basique qui nous intéresse - en premier lieu.

3.2 Stabilité et état des développements

Le développement du produit, très actif, jouit d'un rythme de publication bien établi : une version expérimentale figée toutes les deux semaines et tous les 3 mois, une version de référence, stabilisée, faisant l'objet d'un support étendu. Celle-ci, affublée d'un nom de céphalopode¹² dont la première lettre indique la génération, est indiquée pour tout usage sérieux. « *Emperor* », version 0.72, sortie le 9 novembre 2013, constitue donc la 5ème itération de la solution.

Le numéro de version est inférieur à 1.0 car tous les éléments ne sont pas au même stade de maturité : *RADOS*, *RADOSGW* et *RBD* sont stables depuis la version 0.48 (« *Argonaut* », sortie en juillet 2012). *CEPHFS* est moins mature, ses performances sont considérées comme non optimales et le fonctionnement des serveurs de métadonnées en mode multi-actif est connu pour être encore parfois problématique. *CEPHFS* est donc actuellement déconseillé en support d'applications sensibles.

3.3 Sous le capot...

La partie serveur de *CEPH* est assurée par 3 types de démons différents à déployer :

- *OSD* (Object Storage Daemon): son rôle est simplement de stocker efficacement des objets. Il utilise le disque local du serveur, via un système de fichiers local classique (*XFS*, *BTRFS*, *EXT4*).
- *MON* (MONitor daemon) : son rôle est de vérifier le bon état du cluster, assurer la communication initiale avec les clients et vérifier les droits d'accès.
- *MDS* (MetaData Server daemon) : son rôle est de gérer les méta données pour *CEPHFS*.

L'ensemble de ces démons, déployés en nombre sur des machines, constitue un cluster *CEPH*. Celui-ci ne comporte **aucun point de faiblesse unique**, tous les éléments étant redondés à volonté et fonctionnent en mode multi-actif. Toutes les évolutions de topologie modifient automatiquement des « *maps* » stockées dans le cluster et versionnées. Véritables garanties de la cohérence de celui-ci, elles sont distribuées à tous ses éléments constitutants.

Il est souhaitable d'installer un nombre impair¹³ de *MON* et autant d'*OSD* que nécessaire, puisque la volumétrie totale en est directement dépendante. L'installation de *MDS* est optionnelle si l'utilisation de *CEPHFS* n'est pas envisagée. Enfin, des clés de sécurité doivent être déployées pour sécuriser les échanges. Un script, *ceph-deploy*, permet désormais de simplifier l'ensemble des étapes d'installation pour l'administrateur. L'ajout de *MON* et *MDS* est réalisable *a posteriori*, à tout moment et de manière dynamique.

11. *S3* : Amazon Simple Storage Service, interface très populaire. *SWIFT* : stockage simple d'objet développé par Openstack.

12. D'où le nom de *CEPH*. Le premier logo représentait une pieuvre, désormais remplacée par une version stylisée.

13. Pour éviter le phénomène de « *split brain* », qui, dans le cas d'une panne réseau, divise le cluster en deux parties strictement égales, empêchant ainsi un quorum permettant de décider de façon sûre quelle partie est réellement inaccessible et donc, en panne. Dans les faits, 3 *mon* suffisent.

L'espace disponible s'accroît automatiquement à chaque nouvel *OSD* ajouté et intégré activement. Pour ce faire, ce dernier doit être doté d'un poids¹⁴, et inséré dans une arborescence. L'administrateur peut y définir finement son placement grâce aux notions de *datacenter*, salle, etc. En l'absence d'action de l'administrateur, tous les *OSD* sont placés dans un « rack inconnu », au même niveau. Plusieurs arborescences peuvent coexister et le critère géographique n'est pas le seul possible (la capacité d'entrées/sorties pouvant en constituer une autre).

L'espace de stockage global généré par les *OSD* est adressable via des « *pools* », initialement au nombre de trois à la création d'un nouveau cluster. Chacun associe une des arborescences d'*OSD* à des règles de placement et de réplication de données au sein de la « *CRUSH map* » stockée au sein du cluster. Celle-ci ressemble à ce qui est illustré Figure 4 -. Elle peut être extraite et éditée à la main, ou manipulée directement au travers d'outils en ligne de commande, afin de modifier le poids ou le placement d'un *OSD*, par exemple. Elle est nommée d'après l'algorithme de placement des objets *CRUSH*[13], au sein de *CEPH*. Un système de permissions basé sur des clés partagées protège les ressources. Seul un client possédant la bonne clé pourra accéder à un pool de stockage donné.

```
root default
datacenter loi
  room loire-presidenc
    rack karuizawa
      host ceph-osd-loi-A-1-1
        osd.12 up
    rack hazelburn
      host ceph-osd-loi-A-2-1
        osd.15 up
  datacenter lmb
    room lombarderie-ltc
      rack kavalan
        host ceph-osd-lmb-A-1-1
          osd.6 up
```

Figure 4 - Extrait d'une hiérarchie d'*OSD*

3.4 Réplication des données

Chaque pool est divisé en de nombreux groupes de placement (*PG*). L'algorithme *CRUSH* permet, en fonction des règles définies dans la « *CRUSH map* » courante, de déterminer automatiquement pour un *PG* donné, les *OSD* de stockage (primaire et secondaires). Chaque client du pool est capable de réaliser ce calcul.

Toute donnée stockée dans *CEPH* est découpée en blocs de taille fixe.

Pour un volume *RBD*, cette taille (4 Mo par défaut) est dépendante du pool auquel il appartient. Ces fonctions de découpage sont assurées par le client. Chacun de ces fragments est ensuite assigné à un *PG*. Grâce à l'algorithme *CRUSH*, le client qui veut lire ou écrire un morceau de la donnée connaît les *OSD* concernés et va contacter directement le primaire. En cas d'écriture, l'*OSD* primaire stocke les données dans son dépôt d'objets local, puis s'occupe d'envoyer les données sur les *OSD* secondaires et attend un acquittement. Une fois celui-ci obtenu, lui même envoie un acquittement au client. Le processus est donc **synchrone**.

Ici se trouve une autre originalité de *CEPH*. Tous les « clients » sont intelligents et assurent, en autonomie, une grosse partie du travail. Ils contactent directement les *OSD* sans intermédiaire assurant à la fois de bonnes performances, une parallélisation importante et, par voie de conséquence, une forte capacité de résistance au facteur d'échelle.

3.5 Gestion des pannes

En cas de panne ou dysfonctionnement de l'un des *OSD*, une détection par les *MON* ainsi que les autres *OSD* va exclure temporairement le fautif du cluster ; certains *PG* se trouvent alors dégradés, amputés, soit de l'*OSD* primaire, soit d'un des secondaires. Dans le cas le moins favorable (réplicat du pool fixé à 2), ces *PG* ne disposent plus que d'une seule copie valide des données, considérées alors en danger. Si l'*OSD* fautif n'est pas réapparu au delà d'une courte période (5 minutes par défaut), un processus de reconstruction automatique est engagé, consistant à redupliquer les données impactées sur des *OSD* de substitution.

Chaque *OSD* assure également un auto diagnostic et vérifie l'intégrité des *PG* qu'il gère une fois par jour de façon légère et une fois par semaine de façon complète. Ces vérifications ne s'opèrent que lorsque l'*OSD* est peu sollicité.

Il est maintenant temps de refermer le capot et de se concentrer sur l'utilisation de *CEPH* en tant que client.

3.6 Interface avec l'utilisateur

CEPH, intégré au noyau *LINUX* depuis 2010, bénéficie d'un support natif au sein des distributions les plus répandues, simplifiant en cela son utilisation.

Un client nécessite deux éléments : un fichier de configuration, *ceph.conf* qui doit, au minimum contenir la liste des serveurs *MON*, ainsi qu'un trousseau de clefs, contenant le nécessaire pour accéder aux *pools* souhaités.

14. Qui correspond en général à la quantité de données qu'il est capable de gérer (taille du volume local).

Ici, l'exemple d'un client appelé CIE. Le gestionnaire du cluster lui a donné la possibilité d'accéder au pool INFO:

```
[client.CIE]
key:AQCAfXXX6FrhIRAAZgtXXXdbeRmcBmfbSjY0tg==
caps: [mon] allow r
caps: [osd] allow * pool INFO
```

```
[global]
auth cluster required = cephx
auth service required = cephx
auth client required = cephx
[mon]
[mon.a]
host = ceph-mon-cha-B-1
mon addr = 172.20.106.84:6789
```

Ici, pour référence, les permissions associées au client, côté cluster. Un client aura seulement besoin des 2 premières lignes (la clef) `ceph.conf` allégé, mais suffisant côté client .

Figure 1 - Trousseau de clé et fichier `ceph.conf`, côté client

```
## Montage de type CEPHFS
mount -t ceph -o name=CIE,secretfile=/etc/ceph/secret-CIE 172.20.106.84:/prj /mnt/
## Acces a RBD
rbd map INFO/grosvolume --id CIE ## mappe le volume « grosvolume » du pool INFO
mount /dev/rbd/INFO/grosvolume /mnt2 ## le volume peut être utilisé
```

L'exemple ci-dessus détaille les accès à une ressource *CEPHFS*, et à un volume *RBD*. La syntaxe d'accès à *CEPHFS* est similaire à celle de *NFS* (à l'identification près).

L'accès à une ressource *RBD* est différente mais reste toutefois simple. Puisqu'il s'agit d'un volume de type bloc, celui-ci aura été préalablement formaté avant utilisation. Ces volumes sont aussi directement utilisables depuis l'hyperviseur *KVM*, constituant en cela un choix intéressant dès lors que la migration à chaud de machines virtuelles est considérée.

3.7 Fonctions avancées

La création instantanée de volumes sur un pool donné à l'aide d'une technique de sur-réservation (dite de « *thin provisioning* ») permet une utilisation rationnelle et économique des ressources : l'espace n'est pas alloué immédiatement mais au fil de l'eau, à l'utilisation du volume.

Les « *snapshots* » de volumes à volonté sont gérés, avec d'une part, la possibilité de retour arrière mais d'autre part la possibilité de les utiliser comme image maître de copies indépendantes, utilisables ultérieurement. Une utilité immédiate apparaît dans le cadre de la création de patrons (*masters*) pour machines virtuelles. Le processus utilise un mode « *copy on write* » où seules les modifications apportées à la copie par rapport à l'original consommeront de l'espace dans le cluster, autorisant un gain de place particulièrement important. La nature délocalisée de *CEPH* fait qu'il est possible de gérer ce processus de façon globale à l'ensemble d'un établissement.

4 Genèse de l'implémentation et retour d'expérience

L'adoption de solution logicielle open source encore jeune expose *de facto* à des aléas de fonctionnement. Toutefois, au fur et à mesure que l'équipe commence à mieux intégrer la solution, et que celle-ci, de son côté, évolue et se stabilise, un design itératif nourri des problèmes rencontrés se construit. Il est alors important de ne pas brûler les étapes, de bien évaluer les risques et valider la solution sous peine de gros désagréments. Plusieurs points sont restés communs au fil des déploiements ; citons le placement des éléments (cf Figure 5 -), resté globalement identique à celui de notre plate-forme d'hébergement de services et nécessaire pour atteindre le niveau de tolérance à la panne requis : ces trois points de présence, symétriques, étant indépendants d'un point de vue électrique et vis à vis du matériel réseau et des raccordements optiques associés.

Ne faisant pas l'objet d'achats spécifiques, la plate-forme a été montée avec le matériel existant. Pour chaque point

géographique, des serveurs connectés à des baies *locales*, offrant des disques de 2 To agrégés en RAID5 ont été mis à contribution pour assurer un espace confortable à cette plate-forme. Pour ne pas trop restreindre la quantité de données à stocker, il a été décidé de se limiter à seulement deux copies, divisant tout de même l'espace utile par deux.

4.1 Premier essai

Un nombre très limité d'*OSD* était déployé par plaque géographique, chacun gérant plus de 10 To au moyen de *BTRFS*, le système de fichiers alors préconisé. Mais, rapidement des problèmes de performance et de stabilité sont apparus. Puis, une panne (encore !) d'onduleur a privé d'alimentation tous les serveurs d'un même site. Au redémarrage, 2 des 3 *OSD* ont vu leurs données irrémédiablement détruites. Il s'est avéré que ces différents soucis étaient imputables à *BTRFS*, alors trop jeune. L'erreur principale de ce premier design a été de garder des réglages proches des standards, dont une « *CRUSH map* » ne définissant pas l'arborescence des placements. La perte simultanée de 2 *OSD* dans ce contexte a signifié la perte irrémédiable de données, tous les *OSD* étant alors considérés équivalents, sans critère de localité.

4.2 Second essai

BTRFS a été abandonné au profit de *XFS*¹⁵, le nombre d'*OSD* a été augmenté, pour gérer moins de données (4 To) et les règles de placement de données ont été révisées, un *PG* devant répliquer ses données sur deux plaques géographique séparées.

Après un fonctionnement parfait pendant plus de quatre mois et une résistance avérée à de nombreuses incidents (provoqués ou involontaires), un autre crash a toutefois été déploré.

Afin de réaliser des tests de charge et de performance, un noyau *LINUX* récent (3.6) venait d'être installé sur les serveurs. Pour ces mêmes raisons, notre espace de stockage était bien rempli. L'algorithme de placement des *PG* n'a pas réparti les données de façon complètement uniforme¹⁶ et deux *OSD* étaient quasiment pleins.

Pendant un week-end, une machine a connu des problèmes de stabilité, essentiellement pour des raisons de bugs d'allocation mémoire, imputables au noyau mentionné. Ceci a déclenché la reconstruction des 4 To dégradés. Ce processus stressant a généré un plantage identique de deux nœuds supplémentaires, et la recopie d'une masse de données encore plus importante, aboutissant *in fine*, à un effet boule de neige dévastateur. À la fin du week-end, les *OSD* étaient soit tous pleins, soit inopérants et donc, inaccessibles.

Contre toute attente, le redémarrage des machines plantées s'est extrêmement mal passé. Cette fois, un bug de *XFS*¹⁷ a définitivement corrompu les données d'une partie de nos serveurs. Contacté suite à notre mésaventure, Dave Chinner, développeur *XFS*, a corrigé très rapidement le bug, mais trop tard pour nos données de test.

Bien qu'il s'avère que *CEPH* ne soit en rien coupable de cet incident, il a mis en lumière que la stratégie utilisée était à revoir. Les risques inhérents à de tels déploiements étaient nouveaux, complexes à appréhender, et avaient potentiellement un impact dévastateur. Il fallait donc pouvoir les éviter au mieux.

4.3 Troisième essai et plate-forme de production

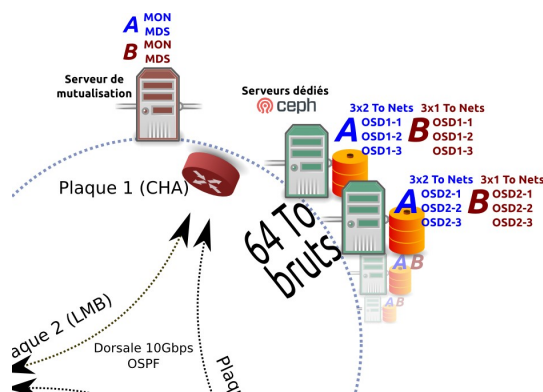


Figure 5 - Une partie de l'infrastructure CEPH

risquée. En cas de panne ou dysfonctionnement, (qu'elle soit matérielle, logicielle), une étape de vérification du

Afin d'éviter l'effet domino, la stratégie « ne pas mettre tous ses œufs dans le même panier » a été appliquée :

- il y aura désormais **plusieurs** clusters, fonctionnant de façon parallèle sur des machines physiques communes. Ceci permettra aussi d'avoir des versions de *CEPH* différentes, une administration séparée et une portée des risques limités. Chaque *MON*, *MDS*, *OSD* aura ses propres machines virtuelles et volumes de données séparés,
- pour chaque cluster, la multiplication des *OSD* pour gérer des volumes plus petits (1 à 2 To maximum) afin de limiter les dommages collatéraux. Faire fonctionner un système de fichiers comprenant plusieurs millions d'entrées et occupant plusieurs To restant une opération

15. N'étant pas les seuls à avoir subi une telle déconvenue, *XFS* est maintenant le système de fichiers local recommandé.

16. C'est toujours vrai, même si les algorithmes ont été améliorés depuis.

17. Bug introduit dans la version 3.0 du noyau, lié à une mauvaise gestion des journaux et capable de détruire le système de fichiers.

disque doit être engagée. En dépassant la dizaine de millions de fichiers, la structure du système de fichiers devient extrêmement complexe, et le résultat de l'étape de vérification devient aléatoire par la faute d'un bug logiciel, d'une saturation mémoire, ou simplement d'un temps d'accomplissement incompatible avec les contraintes d'exploitation,

- toujours veiller à garder un espace libre, pour éviter les *OSD* pleins et pour pallier d'éventuelles reconstructions,
- considérer les nœuds du cluster comme des boîtes noires et limiter les changements logiciels au *strict* minimum.

200 To bruts de stockage ont été mis à disposition (3 sites de 2x32 To en Raid5), soit 52 To utiles par site (2x26 To).

Sur chaque site géographique, 3 à 4 machines physiques hébergent les *OSD* de chaque cluster. N'ayant pas de besoins spécifiques, les *MON* et *MDS* sont eux accueillis sur des serveurs de mutualisation.

Les clusters suivants ont été démarrés :

- « *A* » : production. Uniquement des versions 'stables et long terme' de *CEPH*. Actuellement en version « *Dumpling* » (v0.67.4), constitué de 39 *OSD*, gérant 78 To bruts, pour 39 To utiles (2 répliquas) ;
- « *B* » : expérimental. Les versions de développements sont autorisées, et permettent de tester les nouvelles fonctionnalités. Les données stockées sur ce cluster ne peuvent être considérées comme vitales à l'exploitation. Il est actuellement en v0.71, constitué de 21 *OSD*, pour 10,5 To utiles (2 répliquas).

À peine plus de la moitié de l'espace de stockage physique est alloué, nous permettant, en cas de besoin, d'augmenter les espaces accessibles aux *OSD* de chaque cluster. Avec ces changements, la plate-forme s'est révélée très stable sur plusieurs mois, nous conduisant à démarrer la production début janvier 2013. Elle fonctionne sans soucis depuis.

5 Bilan de l'utilisation de la plate-forme en production

Eu égard aux problèmes rencontrés, la confiance initiale dans la plate-forme était modérée. Le premier usage a été de stocker des duplications de sauvegarde. Au fil du temps, la confiance est revenue et les sauvegardes sont désormais stockées, de manière primaire, sur des volumes *RBD*. Celles-ci occupent le cluster « *A* », et des dizaines de millions de fichiers sont sauvegardés quotidiennement. Nous avons aussi profité de l'importante capacité disponible pour des processus de migration (gros volumes de baies de stockage traditionnelles, déplacement de données de vieux serveurs non reliés au *SAN*, création d'images de disques durs temporaires, etc)

Des services expérimentaux ont aussi été initiés, en particulier un service de stockage de fichiers personnels sur le cluster *B*. Le système est actuellement basé sur *CEPHFS*¹⁸, mais une utilisation de *RADOSGW* (via l'API *S3*) est envisagée. Des systèmes de gestion et d'analyse massif de journaux (*logs*) ont également été installés. Ce cluster sert aussi à valider certains paramètres et réglages.

5.1 Bénéfices à l'utilisation

Si la mise en place de la solution reste une opération relativement complexe, surtout dans un contexte de déploiement multiples de clusters, son utilisation est en revanche très simple. Mais plus important encore, l'administration des espaces est également nettement plus simple que sur un *SAN* : plus d'opérations de zoning ou de *LUN masking* à effectuer, plus de maintenances de commutateurs *fibres channel*. La sécurisation des données est intrinsèque et offre une haute tolérance de panne (sinistre majeur), en parfaite adéquation avec nos objectifs de PCA.

L'aspect global d'une telle solution permet une utilisation du stockage à n'importe quel endroit de l'Université, et ne devient presque plus qu'une « simple » question de débit d'accès de nos différents sites universitaires.

Le gain en terme de souplesse est particulièrement important ; la mise à disposition de volumes de plusieurs To pour un projet spécifique demande quelques secondes, de même que la mise à disposition d'images de serveurs « prêts à l'emploi ».

5.2 Performance

La performance des systèmes de stockage distribué fait l'objet d'après comparatifs et de nombreux tests. Nous en réalisons nous-même régulièrement pour valider nos choix. Cet article n'en exposera pas, tant par manque de place que parce que de nombreuses études de performances sont publiées. Enfin, les chiffres publiés seraient peu pertinents, notre plate-forme d'hébergement n'étant pas homogène (plusieurs générations de serveurs, les plus lents ralentissant l'ensemble). Les copies étant synchrones, la répartition sur trois sites distants de plus jusqu'à 13 kilomètres induit

18. Owncloud et Pydio ont été testés, mais aucune solution ne semble encore tout à fait mature pour un déploiement à large échelle.

également une latence sensible (0,44 ms).

Évaluer la performance d'un système ne se limite pas à lancer quelques *benchmarks* surtout s'ils sont *mono threadés*, alors qu'une des forces du système est sa montée en charge lors d'opérations parallèles. Des tests en situation réelle ont ainsi démontré l'intérêt d'outils *multi-threadés* tels que *mcp*¹⁹[14] pour augmenter très fortement les débits.

Disons simplement qu'avec un peu de réglage fin, la partie *RBD* peut saturer un lien à 10 Gb/s et dans la plupart des tests, les performances obtenues sont proches à moins de 30 % de la plupart de nos baies classiques. Pour le dire simplement, la performance est suffisamment bonne pour nos usages et permet, tant en terme de fonctionnalités qu'en terme de performances, d'envisager le remplacement des baies de stockage *SAN* dans la majorité des cas.

5.3 Limitations

- Le paragraphe 3.4 a montré que les écritures entre *OSD* sont synchrones, limitant les performances mais dont l'impact peut être réduit grâce au cache local des serveurs et une gestion optimisée du « *read-ahead* ». Ainsi, les performances restent très correctes sur un plan métropolitain. Dans un cadre régional ou national, il est probable que le problème deviendrait prégnant. Des développements vers la mise à disposition d'une réplication asynchrone sont en cours.
- Chaque client communique directement avec les *OSD*, *MON* et *MDS* déployés, impliquant un routage et une gestion des filtrages réseaux en entrée de site appropriés. Ceux-ci doivent être adaptés à 10 Gb/s et capables d'absorber le flux induit. Dans le cas de machines faisant fonctionner des applications critiques, des vlans bien identifiés (isolés) et restreints à l'accès de la plate-forme *CEPH* doivent être définis.
- En terme de sécurité et contrôle des accès, la granularité se situe au niveau du pool et non pas des volumes, ce qui aurait été préférable pour nos besoins.
- Il n'est pas possible de déployer de réels mécanismes de *QoS*. Une gestion fine des *pools*, conjointement à du *shaping* au niveau réseau permettrait d'y arriver, mais la situation est loin d'être équivalente à notre existant.
- Le *thin provisioning* des volumes n'est pas dénué de risques. Une idée naïve est que l'allocation des *PG* est fonction directe de l'usage du système de fichiers ; pourtant un usage cumulatif se traduisant par le remplissage progressif des *PG* du *pool* est constaté. En réalité, lorsque des données sont effacées du système de fichiers, le périphérique de type bloc n'est pas averti : les *PG* ne sont alors pas libérés. Pour faire le lien et informer la couche sous-jacente, il faut le support des fonctions « *TRIM* ». *KVM* supporte cette fonctionnalité, mais pas encore le pilote *RBD* du noyau ; dans ce cas, il ne faut pas sur-réserver les espaces.

Enfin, le vrai problème est celui de l'appropriation de la technologie par l'équipe, eu égard l'importance stratégique du projet. Une technologie open source aussi complexe ne peut être maîtrisée que par une seule personne. Une montée en compétence de l'équipe et un suivi important sont requis. Le vrai passage à l'échelle se situe à ce niveau.

5.4 Évolution de la plate-forme

Les serveurs physiques utilisés (équipés de 2 disques internes agrégés en *RAID1*) ne fournissent pas de performances optimales pour *CEPH*. Ils gèrent de nombreux *OSD* virtualisés pour différents clusters. En phase d'écriture, chacun d'eux gère dans un premier temps le remplissage d'un journal local stocké sur l'agrégat *RAID1* du serveur physique, puis ensuite le transfert sur le volume issu d'un agrégat *RAID5* unique de la baie associée, qui sert aussi aux lectures.

Pour maximiser le parallélisme de la solution, chaque *OSD* devrait être indépendant, mais dans ce schéma, il n'en est rien, tous sont hébergés sur une même machine physique et partagent les mêmes périphériques. Ils se gênent mutuellement, en particulier dans les phases d'écriture sollicitant l'agrégat *RAID1*, aux performances médiocres. Cela est particulièrement visible lorsqu'une reconstruction est en cours, les performances en étant fortement impactées.

Début 2013, trois machines dimensionnées pour *CEPH* (une par plaque géographique) ont été déployées. Elles sont équipées de deux disques *SSD* pour stocker les journaux et disposent de 12 disques locaux SATA de 3 To configurés en 3 agrégats *RAID5* dédiés à un *OSD* par cluster. Le résultat est sans appel, avec un facteur 3 à 4 d'amélioration pour ces serveurs qui ne représentent malheureusement qu'un tiers de notre parc.

5.4.1 Gestion des réplicats

Trois nouvelles machines du même type vont renforcer notre ensemble en novembre 2013, avec comme objectif de disposer d'autant *OSD* indépendants que de disques pour renforcer le parallélisme de la solution. Il sera fait l'impasse sur l'organisation des données en *RAID*. Il s'agit d'une autre approche de la sécurité, une étape psychologique est franchie et

19. Modification de cp pour fonctionner en mode multi-thread. Redoutable en tandem avec ceph !

nécessite une confiance absolue dans *CEPH* : la sécurité des données repose désormais intégralement sur celui-ci.

Pour diminuer les risques, ces machines seront intégrées dans un pool dont la réplication est placée à 3. En cas de panne d'un disque, il n'y a plus de sécurité liée au *RAID*, mais la donnée est encore disponible à deux endroits dans le cluster, ce qui assure une bonne sécurité. En cas d'une panne ou isolation complète d'une plaque géographique, toutes les données sont encore disponibles en deux endroits, et ne sont donc pas en danger. Il n'y a pas reconstruction dans ce cas, ce qui évite les pertes de performance et les éventuels « effets domino », déplorés par le passé. De surcroît, cette stratégie nous protège mieux en cas de corruption due à un système de fichiers éventuellement défaillant.

Enfin, un rapide calcul montre qu'en considérant 3 machines équipées de 12 disques de 4 To, soit 144 To bruts, la solution de 3 copies offre 48 To utiles avec 36 *OSD* indépendants contre 54 To utiles pour 9 *OSD* indépendants, ce qui est acceptable, même si dans les deux cas, une très grosse partie de l'espace reste gaspillée.

5.4.2 Erasure coding

Le système de réplication de données est donc bien plus onéreux qu'un système *RAID* traditionnel, même si la sécurité offerte est incomparable. Dans un système *RAID5*, la sécurité des données est assurée par une parité supplémentaire gérée bit à bit, et seul l'équivalent d'un disque est consommé par ce processus. Ainsi, la diminution de l'espace global de stockage est limitée. Il est possible d'appliquer ce principe aux systèmes de fichiers distribués, en modifiant le codage de l'information, grâce à une technique dite d'« *erasure coding* » qui apporte une redondance à l'information. Celle-ci est distribuée géographiquement par petits blocs. La redondance doit être suffisante pour permettre le restitution de la donnée d'origine malgré la perte d'un ou plusieurs morceaux. Cette technique peut se révéler très économe à partir du moment où une forte répartition est possible. Le système *ROZFS* utilise déjà ce principe grâce à une fonction de transformation appelée *mojette*[12], issue de travaux de recherches menés à l'Université de Nantes par J.P Guedon.

Dans notre cas précis, l'expérience et les pannes subies nous ont montré qu'il ne fallait pas raisonner sur le nombre de serveurs déployés, mais seulement sur nos sites, seuls éléments à être complètement indépendants. En utilisant l'*erasure coding*, la panne complète d'une plaque géographique doit laisser 100 % des données disponibles, signifiant que chaque point géographique doit détenir au moins 50 % des données. Ce qui donne au mieux un facteur 1.5 (au lieu du facteur 2 actuellement en cours). le gain est limité mais non négligeable, la disponibilité de points géographiques indépendants supplémentaires permettrait d'abaisser ce facteur, mais n'est actuellement pas possible.

L'incorporation d'une librairie d'*erasure coding* est en cours dans *CEPH*, elle a été initiée par Loïc Dachary et est maintenant supportée officiellement par l'équipe de développement et prévue pour la version 'F' : Firefly (Février 2014)

5.4.3 Tiering

Cette même version devrait aussi intégrer un mécanisme de « *tiering* ». Le principe est de définir plusieurs types de pools selon des critères (rapidité, sécurité, coût du matériel...). Le mécanisme de *tiering* permettra de calculer la fréquence d'utilisation des objets et de les déplacer de façon transparente entre pools en fonction de la politique choisie.

Ainsi, les objets auxquels on accède très fréquemment, tels que les index de bases de données, seront promus dans les pools les plus rapides, potentiellement hébergés sur des machines disposant de SSD, alors que ceux moins utilisés seront relégués dans les pools les moins rapides, y compris éventuellement sur des volumes potentiellement moins sécurisés mais plus économiques.

5.5 Développements futurs, stratégie

Le système de clusters multiples fonctionnant bien, le rôle actuel des clusters est affiné et deux nouveaux clusters vont être démarrés pour 2014.

- « *C* » : « production ». Il accueillera les besoins des machines virtuelles et servira pour les interactions avec *openstack*, le cluster *A* étant désormais exclusivement dévolu aux sauvegardes. Le passage à *openstack* est un sujet à part et fera l'objet d'un travail important dans les années qui viennent, mais il est clair que *CEPH* est complémentaire de cette technologie et est supporté officiellement.
- « *D* » : « tiers », permettra des stockages volumineux pour des projets spécifiques.

6 Conclusion

Nous pouvons désormais affirmer que la plate-forme est fonctionnelle, robuste et répond en très grande partie à nos besoins. La vitalité du projet, sa communauté importante et dynamique, les fonctionnalités importantes, sont autant de signes sur la réussite de *CEPH*. En l'état, ses fonctionnalités et sa performance nous permettent actuellement un remplacement progressif d'une partie de notre stockage traditionnel sous certaines conditions. Les fonctionnalités en

cours de développement, en particulier le *tiering*, ne font qu'abonder vers cette solution, dont le spectre d'utilisation ne cesse d'augmenter. Il est toutefois dépendant des investissements en machines dédiées que nous serons capables d'effectuer et surtout des moyens humains à mettre en œuvre pour pérenniser cette solution.

Bibliographie

- [1] Yann Dupont, Yoann Juet et Jean-philippe Menil. Plateforme d'hébergement de services tolérante à la panne, redondée géographiquement. Dans Actes du congrès JRES2011, Toulouse, Novembre 2011. https://2011.jres.org/archives/116/paper116_article.pdf
- [2] LUSTRE. <http://www.lustre.org>
- [3] GLUSTERFS. <http://www.gluster.org>
- [4] HEKAFS. <http://hekafs.org/>
- [5] MOOSEFS. <http://www.moosefs.org/>
- [6] ROZOFs. <http://www.fizians.com>
- [7] OCFS2. <http://oss.oracle.com/projects/ocfs2/>
- [8] GFS2. <http://www.redhat.com/products/enterprise-linux-add-ons/resilient-storage/>
- [9] DRBD. <http://www.drbd.org/>
- [10] CEPH. <http://www.ceph.com/>
- [11] Sage Weil : Ceph: reliable, scalable, and high performance distributed storage, University of california, Santa cruz. Décembre 2007. <http://ceph.com/papers/weil-thesis.pdf>
- [12] Jean-Pierre Guedon. The Mojette Transform: Theory and Applications. ISTE / WILEY. ISBN : **978-1848210806**
- [13] Sage A. Weil Scott A. Brandt Ethan L. Miller Carlos Maltzahn. CRUSH: Controlled, Scalable, Decentralized Placement of Replicated Data. Storage Systems Research Center, University of California, Santa Cruz. <http://ceph.com/papers/weil-crush-sc06.pdf>
- [14] MCP. <http://ti.arc.nasa.gov/opensource/projects/mcp>