RFC 6392 : A Survey of In-network Storage Systems

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 26 octobre 2011

Date de publication du RFC : Octobre 2011

http://www.bortzmeyer.org/6392.html

Cette intéressante étude des systèmes de stockage de données en ligne (le RFC n'utilise pas le terme de "*cloud*", pourtant bien plus vendeur) fait le tour des principales solutions d'aujourd'hui. Son but est de préparer le travail ultérieur du groupe de travail IETF DECADE <http://tools.ietf.org/wg/decade>, dont le rôle est de normaliser une architecture pour l'accès au stockage en ligne, notamment pour le pair à pair. Ce RFC, le premier produit par DECADE, étudie l'offre existante, afin de voir dans quelle mesure elle est réutilisable pour DECADE.

Notons que tous les systèmes existants n'y sont pas : l'accent a été mis sur une sélection de systèmes typiques couvrant à peu près toutes les variantes possibles d'un tel service. C'est ainsi que, par exemple, Swift/OpenStack n'y figure pas. Par contre, le plus célèbre, le S3 d'Amazon est présent. Certains de ceux qui y figurent sont très expérimentaux, peu déployés, mais ont été ajoutés parce qu'ils illustrent un point important.

Il existe des tas de raisons pour stocker ses données dans le nuage. Cela peut être pour y accéder de partout, depuis n'importe quelle machine. Ou bien cela peut être pour rapprocher les données des utilisateurs. Si on distribue du contenu en pair à pair, il vaut mieux qu'il soit déjà dans le nuage, plus près des téléchargeurs, que sur une machine à la maison, connectée en ADSL, où le coût du dernier kilomètre à franchir est élevé.

La section 2 brosse un panorama historique du stockage en ligne. Par exemple, le goulet d'étranglement que représente le gros site Web, lorsqu'il a de très nombreux utilisateurs, a été partiellement levé avec l'introduction des CDN. Ceux-ci dupliquent le contenu statique des sites Web, et le rapprochent des utilisateurs, en plaçant des copies chez les principaux FAI. Mais les CDN ne sont pas accessibles à l'utilisateur normal. Si je voulais accélérer le temps de chargement des pages de http://www.bortzmeyer.org/, je ne pourrais certainement pas monter un CDN moi-même, un tel travail nécessite des moyens financiers et administratifs hors de ma portée.

Le pair à pair a ensuite fourni d'autres possibilités. Si on utilise les machines de tout le monde pour distribuer du contenu, pourquoi ne pas également s'en servir pour améliorer l'accès au contenu par une réplication et une copie proche des autres pairs ? Plusieurs systèmes de « caches P2P » ont ainsi été développés pour accélérer le pair à pair (cf. section 4.12). Malheureusement, tous sont spécifiques à une application particulière et ne peuvent donc pas servir aux nouvelles applications. Le monde du pair à pair évoluant vite, il ne peut donc pas bénéficier d'une infrastructure de stockage existante. Le but de DECADE est justement de permettre la création de cette infrastructure.

Pour pouvoir étudier de manière systématique les techniques existantes, la section 3 du RFC liste les composants d'une solution de stockage en ligne. Chaque technique sera alors décrite pour chacun de ces composants :

— Une interface d'accès aux données, permettant les opérations de base, notamment lire et écrire,
— Une interface de gestion des données, permettant des choses comme la suppression d'un bloc de données,
— Une interface de recherche permettant de s'y retrouver dans ses données,
— Un mécanisme de contrôle d'accès, permettant d'avoir des données publiques, des données restreintes à certains utilisateurs (par exemple seulement depuis certains pays), des données privées (accessibles uniquement à des clients identifiés, par exemple après paiement). L'accès peut dépendre de l'opération (lecture pour tous mais écriture seulement pour certains).
— Un mécanisme de découverte des mécanismes d'accès (le mode d'emploi du service),
— Un mode de stockage : les données sont-elles des fichiers (organisés en répertoires hiérarchiques), des objets (contrairement aux fichiers, ils ne sont pas organisés), ou simplement des blocs d'octets, comme le propose le disque dur local ?

Sur la base de ce découpage en composants, on peut décrire les systèmes existants (section 4, mais rappelez-vous que le RFC ne prétend pas les présenter tous et que je n'ai pas repris ici tous ceux listés dans le RFC). Notez que l'ordre de la présentation est celui du RFC, même s'il peut sembler incohérent. D'autre part, un bon nombre des descriptions ressemblent à du copier-coller depuis les descriptions officielles (y compris dans leur côté marketing).

Commençons par le plus connu, Amazon S3. Les utilisateurs créent des containers, les **seaux** (*"buckets"*) puis y déposent des objets. Les accès se font en REST.

Pour montrer les capacités de S3, testons un peu avec un client Unix en ligne de commande, s3cmd <http://s3tools.logix.cz/s3cmd>. On lui indique les clés avec `s3cmd --configure` puis on peut créer un seau (le nom est hiérarchique car un nom de seau doit être unique entre **tous** les utilisateurs ; on peut aussi préfixer avec l'ID Amazon) :

```
% s3cmd  mb s3://org-bortzmeyer-essais
```

(où `mb` signifie *"make bucket"*, créer un seau.) Puis on peut mettre un fichier dans ce seau :

```
% s3cmd  put tartiflette1.jpg s3://org-bortzmeyer-essais
/home/stephane/Downloads/tartiflette1.jpg -> s3://org-bortzmeyer-essais/tartiflette1.jpg  [1 of 1]
...
 35267 of 35267   100% in    0s    37.21 kB/s  done
```

Il est ensuite possible de récupérer cet objet, par exemple depuis une autre machine :

```
% s3cmd  get s3://org-bortzmeyer-essais/tartiflette1.jpg
...
s3://org-bortzmeyer-essais/tartiflette1.jpg -> ./tartiflette1.jpg  [1 of 1]
 35267 of 35267   100% in    0s    74.04 kB/s  done
```

On a donc vu les deux commandes les plus importantes, `put` et `get` (lire et écrire des objets).

S3 est devenu très populaire, entre autres en raison de sa simplicité. Pas d'espace de nommage hiérarchique, pas de possibilité de modifier les données existantes, accès en REST... L'interface d'accès permet de lire et d'écrire (les deux opérations illustrées plus haut), on peut détruire les objets, on peut lire le contenu d'un seau (c'est la seule possibilité de recherche), S3 fournit des mécanismes de contrôle d'accès riches, il n'y a pas de vraie découverte (l'utilisateur doit connaître l'URL où envoyer les requêtes, dans l'exemple ci-dessus, on utilise celle par défaut), et enfin, on l'a vu, S3 stocke des objets (aucun mécanisme de dossiers ou répertoires).

Il existe d'autres systèmes analogues à S3 comme Dropbox.

Un exemple d'un service très différent est le BranchCache de Windows, assez répandu en raison de la diffusion de ce système d'exploitation. Les accès ne sont pas explicites comme avec S3, BranchCache vise à stocker et récupérer les données automatiquement dans un cache. Il y arrive en se mettant sur le trajet des requêtes HTTP et SMB. Il intercepte la requête normale, et accède au fichier depuis le cache, si c'était une opération de lecture et que le fichier était déjà disponible.

Autre exemple très différent de S3, le CNF ("*Cache-and-Forward*"), une architecture de recherche visant à doter les réseaux d'un mécanisme efficace d'accès aux données. Dans CNF, les routeurs ne sont plus simplement des routeurs qui opèrent au niveau 3 mais des moteurs d'accès des données ("*store and forward*") qui font suivre les requêtes d'accès et gardent le résultat localement, pour les accès ultérieurs. Le contenu très populaire est ainsi petit à petit copié dans les routeurs les plus proches. Un mélange de Usenet et de cache opportuniste (tout le contenu n'est pas copié, contrairement à Usenet ou aux CDN, seulement celui qui est demandé).

Plus dans l'idée d'un accès explicite aux données, le standard CDMI vise à résoudre un problème récurrent du "*cloud*" : l'absence de normes pour l'accès aux données, qui font qu'une fois choisi un fournisseur, on en est prisonniers. Si vous bâtissez votre service autour d'un accès à S3, vous ne pouvez plus quitter Amazon. Certes, l'interface de S3 est tellement populaire qu'il existe désormais des offres compatibles (hébergées, ou bien à construire soi-même comme celle d'Eucalyptus) mais ces offres sont à la merci d'un changement unilatéral de l'interface S3 par Amazon. CDMI est donc une tentative pour concevoir et populariser, au sein d'un consortium industriel, la SNIA, une interface standard. Elle ressemble dans ses principes à S3 et repose sur les modes actuelles (REST et JSON).

Comme CDMI est indépendant d'un vendeur particulier, et repose sur des protocoles simples et bien connus comme HTTP, elle est une approche tentante pour le groupe DECADE.

J'ai déjà parlé des CDN. Ils représentent aujourd'hui une des approches les plus populaires, d'accès aux données en ligne. Un bon tour d'horison complet des CDN figure dans « "*A Taxonomy and Survey of Content Delivery Networks*" `<http://www.cloudbus.org/reports/CDN-Taxonomy.pdf>` » de Pathan, A.K. et Buyya, R. Le principe est de copier à l'avance le contenu vers des serveurs situés chez les FAI et d'utiliser divers trucs (par exemple fondés sur le DNS) pour que les utilisateurs soient redirigés vers ces serveurs. Le plus connu des CDN est Akamai mais il en existe bien d'autres comme Limelight ou CloudFront. Pour DECADE, l'intérêt des CDN est que c'est une technique très répandue et éprouvée. Mais elle nécessite une relation d'affaires existante entre l'opérateur du CDN et le fournisseur de contenu (contrairement aux caches classiques).

Le CDN ne fournit typiquement d'accès qu'en lecture (l'écriture est réservée au fournisseur de contenu, via le mécanisme que lui offre l'opérateur du CDN).

Plus proche du CNF, le système DTN vise à créer un environnement adapté aux cas difficiles, notamment aux missions spatiales. Dans ces cas où la connectivité est très intermittente (une seule tempête solaire peut tout interrompre pendant des semaines), et où les délais sont énormes, les protocoles TCP/IP classiques ne conviennent plus. D'où le DTN, normalisé dans le RFC 4838 [1], une architecture "*store and forward*" qui ressemble davantage à UUCP.

Le DTN repose sur le protocole "*Bundle*" (RFC 5050), qui permet de stocker les données en attendant que le destinataire soit à nouveau disponible. Bien sûr, IP est lui aussi "*store and forward*" dans la mesure où le routeur garde le paquet en mémoire avant de le passer au routeur suivant. Mais le temps de rétention des données dans IP est typiquement de quelques milli-secondes au maximum, alors qu'il atteint couramment plusieurs jours dans Bundle, imposant le recours à un stockage persistant (au cas où le routeur redémarre). Contrairement à IP, où le routeur a le droit de jeter des paquets, Bundle est plus proche du courrier électronique : le routeur doit prendre soin des données jusqu'à ce qu'il ait pu les transmettre. Il n'est donc pas exagéré de l'inclure dans cette liste des services de stockage en ligne.

Dans la catégorie futuriste, une autre famille de candidats intéressants est celle qu'on va appeler, pour simplifier. les « réseaux fondés sur les objets nommés » (c'est le nom de la première conférence scientifique sur le sujet <http://www.ieee-infocom.org/2012/nomen/>). C'est une vaste famille, comprenant uniquement des projets de recherche fondamentale, qui ont en commun le fait que toute l'architecture du réseau repose sur des objets de données, ayant un nom, et que le réseau est conçu pour accéder rapidement et de manière sûre à ces objets. Cette famille privilégie donc l'accès au contenu. On y trouve des systèmes comme CCN <http://www.bortzmeyer.org/van-jacobson-ccn.html>, ou comme NDN ("*Named Data Networking*") et NetInf ("*Network of Information*"), traités plus en détail dans ce RFC.

NDN <http://www.named-data.net/> est un tel système. Le réseau gère des objets nommés et signés cryptographiquement. Les routeurs qui transmettent ces objets ont la possibilité de les garder en cache. Les objets les plus demandés vont donc avoir plus de chance de se retrouver dans un cache proche, réalisant ainsi automatiquement ce que les CDN font manuellement. Comme les autres propositions de la famille "*Name-oriented networking*", tout repose sur l'idée que « "*data delivery has become the primary use of the network*" ».

NetInf <http://www.netinf.org/> reprend les mêmes affirmations minitéliennes et y ajoute des fausses explications sur ce que représente un URL (le deuxième champ d'un URL, après le plan, n'identifie **pas** une machine).

Plus sérieux, OceanStore <http://oceanstore.cs.berkeley.edu/> est une architecture où un ensemble de machines coopèrent en mettant en commun des espaces de stockage. OceanStore met l'accent sur la résilience et l'auto-organisation. Son intérêt pour DECADE est surtout cette capacité à résister aux pannes, et même à une certaine proportion de participants malhonnêtes (qui modifieraient les données, ou bien les effaceraient). OceanStore fournit une interface permettant lecture et écriture des objets mais reste encore très expérimental.

Un des domaines où il y a eu le plus d'utilisations du stockage en ligne est évidemment le partage de photos. Une section est donc consacrée à ce cas particulier, où on voit apparaître des noms connus comme Flickr ou ImageShack, sans compter des services qui, sans être spécialisés dans les photos, sont néanmoins largement utilisés pour cela, comme Tumblr. Une particularité de ces services est

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc4838.txt

l'accent mis sur les fonctions de recherche (qui sont absentes de beaucoup de services de stockage), notamment par le biais d'étiquettes <http://www.bortzmeyer.org/tagging.html> attachées aux photos. On peut ainsi retrouver, parmi des myriades de photos, celles qui concernaient saint-quentin-en-yvelines <http://www.flickr.com/search/?q=saint-quentin-en-yvelines&f=hp> ou lascaux <http://imageshack.us/photos/lascaux>. Ces services ont en général également des moyens d'organisation des photos (en « albums » ou « galeries », analogues à ce que fournit un système de fichiers).

Ces services fournissent la plupart du temps des moyens de désigner les photos comme privées (personnelles) ou publiques et, dans certains cas, comme limitées à un ensemble de personnes. À noter que le RFC n'évoque pas un problème sérieux du stockage en ligne : le fait que rien n'est privé pour l'opérateur du service. On peut toujours marquer ses données comme privé, l'opérateur y a quand même accès (ainsi que la police et la justice de son pays).

Le RFC n'hésite pas à mélanger technologies très nouvelles (et souvent très expérimentales) et très anciennes. Usenet a ainsi sa section. C'est un système de distribution de messages, rangés hiérarchiquement par sujet (cf. RFC 5537). Ces messages sont automatiquement recopiés entre tous les serveurs Usenet qui participent à ce réseau. L'utilisateur final peut alors y accéder via un protocole comme NNTP. Usenet était donc du P2P longtemps avant que ce terme ne soit à la mode. Si Usenet semble nettement sur le déclin aujourd'hui, il reste un des plus importants exemples de distribution efficace de contenu sur le réseau.

En tant que système de stockage en ligne, Usenet permet de lire et d'écrire (on dit "*to post*"), et, dans une mesure très limitée, de détruire des messages (avec l'opération "*cancel*").

S'il y a les caches pair à pair, un autre type de cache largement déployé est le cache Web, qui fait l'objet d'une analyse dans ce RFC. L'idée est surtout d'économiser de la capacité réseau (et de gagner en latence), et le cache Web est donc en général mis en œuvre par le FAI ou par les administrateurs du réseau local. Le protocole HTTP dispose de tout un tas d'options pour contrôler finement le niveau de « cachage ». Le logiciel libre le plus répandu pour cette tâche est Squid.

Ce service ne donne accès aux utilisateurs qu'en lecture, l'écriture est faite automatiquement, comme conséquence de la lecture.

Après ce long catalogue de systèmes très variés, la section 4.15 fait le point : est-ce que DECADE a trouvé ce qu'il cherchait ? La plupart de ces services sont nettement plus client-serveur que pair à pair. Et beaucoup sont spécifiques au « cachage », pas au stockage à long terme (un cache peut toujours perdre des données, par exemple, elles seront récupérées sur le site originel). Pour ces raisons et pour d'autres, il est probable que DECADE devra développer son propre service.

Après l'examen des services, la seection 5 du RFC porte sur un examen des protocoles. Quels protocoles existants pourraient être utiles pour DECADE ? Le premier examiné est bien sûr HTTP (RFC 7230). S'il est surtout connu pour ses fonctions de lecture (téléchargement de ressources Web), HTTP peut servir à bien d'autres choses, à développer des applications REST ou à écrire du contenu nouveau sur le serveur. Un des gros avantages de HTTP est évidemment que les logiciels clients sont déjà disponibles, du navigateur Web au programme en ligne de commande comme cURL, en passant par des bibliothèques pour tous les langages de programmation.

HTTP permet donc les fonctions d'accès aux données (opérations `GET`, `PUT`, etc), de gestion des données (le RFC fait une erreur <http://www.rfc-editor.org/errata_search.php?eid=3006> en disant qu'il n'y en a pas, ce qui oublie `DELETE`, section 4.3.5 du RFC 7231) et de contrôle d'accès (accès

public, ou accès restreint après authentification). Les données sont nommées « ressources » et sont souvent des fichiers. HTTP étant client-serveur, il ne convient pas parfaitement à DECADE, qui met plutôt l'accent sur le pair à pair.

HTTP a servi de base à d'autres protocoles. Pour l'accès aux données en ligne, on pense évidemment à WebDAV (RFC 4918). C'est une extension de HTTP qui lui ajoute des fonctions comme le verrouillage ou comme le regroupement des ressources en collections, qu'on peut gérer collectivement. Ces collections font que le modèle de données de WebDAV devient très proche de celui d'un modèle « système de fichiers ».

WebDAV a également une extension pour avoir des ACL, le RFC 3744. Par contre, il n'est pas évident que ce mécanisme de gestion des droits passe à l'échelle, dans le contexte de l'Internet entier.

Autre protocole d'accès à du stockage en ligne, iSCSI, normalisé dans le RFC 7143. En gros, il s'agit d'un protocole permettant l'envoi de commandes SCSI au dessus de TCP, vers des périphériques de stockage comme des disques durs. iSCSI est typiquement un outil pour réaliser des SAN, bien qu'il puisse aussi servir au dessus de l'Internet (par exemple avec les services de nommage du RFC 4171).

iSCSI fournit des opérations de lecture et d'écriture sur des blocs de données (qui correspondent en général aux blocs du disque dur). iSCSI est de très bas niveau et aucun développeur d'applications n'aurait envie de travailler directement avec ce protocole. En réseau local, on utilise plutôt NFS, dont les dernières versions (RFC 5661) disposent de ce qu'il faut pour travailler sur l'Internet.

NFS fournit des opérations de lecture, d'écriture, de gestion (destruction, renommage) de fichiers. Il a des mécanismes de contrôle d'accès qui se sont particulièrement perfectionnés avec sa version 4, qui a amené les ACL.

Par contre, il n'est pas du tout évident qu'il puisse servir dans un service pair à pair de grande taille, notamment vue la rigidité de certains de ses mécanismes de sécurité (même problème qu'avec WebDAV).

Le dernier protocole envisagé, OAuth (RFC 5849) n'est pas vraiment un protocole d'accès à du stockage en ligne. S'il est mentionné ici, c'est parce qu'il fournit un modèle intéressant pour le contrôle d'accès à ces services de stockage. Les modèles traditionnels n'avaient que deux acteurs, le client et le serveur. Le serveur autorisait (ou pas) le client à accéder aux données. OAuth introduit un troisième acteur, le titulaire des données. Celui-ci peut donner une autorisation d'accès à un client tiers, sans lui fournir pour autant toutes ses lettres de créance (mots de passe, clés privées, etc). OAuth permet donc de déléguer l'accès.

Que peut-on synthétiser de l'examen de ces protocoles ? Ils sont surtout conçus pour du client-serveur (bien que certains puissent être adaptés au pair à pair). Et HTTP est tellement répandu et bien connu qu'il peut fournir un bon point de départ à DECADE. Par contre, aucun de ces protocoles ne prend en compte la nécessité d'accès avec faible latence, qui est nécessaire pour la diffusion de vidéo en "*streaming*".

Enfin, si presque tous les protocoles fournissent les mécanismes de base, comme lire et écrire, la grande majorité n'a aucun mécanisme de contrôle de l'allocation des ressources, un des objectifs de DECADE (il s'agit de permettre un accès à ses ressources, sans que les pairs ne puissent tout consommer).

Bref, résume la section 6, conclusion de ce RFC, s'il existe de nombreux systèmes de stockage en ligne, ils ont été conçus avec un cahier des charges très différent de celui de DECADE (l'exemple le plus net étant leur conception client-serveur et pas pair à pair). Aucun ne peut donc être choisi tel quel pour le projet.