Passage à l'échelle de l'hébergement de messagerie par une approche décentralisée

Jean Benoit
Direction Informatique, Université de Strasbourg
jean@unistra.fr

Christophe Palanché
Direction Informatique, Université de Strasbourg
palanche@unistra.fr

Benjamin Collet
Direction Informatique, Université de Strasbourg
bcollet@unistra.fr

Vincent Lucas
Direction Informatique, Université de Strasbourg
lucas@unistra.fr

Fabrice Peraud
Direction Informatique, Université de Strasbourg
fperaud@unistra.fr

Résumé

La messagerie Osiris a une longue histoire. Partie d'une plate-forme de quelques centaines de comptes à l'aube des années 2000, elle héberge actuellement plus de 110 000 boîtes pour 26 To de données.
Cet article s'attachera à retracer les grandes étapes de la mise en place de cette nouvelle infrastructure de messagerie basée actuellement sur FreeBSD, ZFS, Cyrus IMAP et Nginx. Tout d'abord, nous présenterons les différentes solutions techniques étudiées et les raisons des choix effectués pour répondre aux besoins de performance et d'extensibilité requis pour les années à venir. Puis, nous ferons une description détaillée de l'architecture de la plate-forme, ainsi que de son implémentation, en incluant les méthodes utilisées pour le déploiement. Enfin, nous reviendrons sur le déroulement du projet et nous ferons un retour d'expérience des premiers mois d'exploitation de la plate-forme.

Mots clefs

messagerie, scale out

1 Contexte

La Direction Informatique est la structure en charge du réseau Osiris pour la communauté de l'enseignement supérieur à Strasbourg. Elle propose également un service d'hébergement mutualisé de messagerie.
La messagerie Osiris a une longue histoire [1]. Partie d'une plate-forme de quelques centaines de comptes

à l'aube des années 2000, elle héberge actuellement plus de 110 000 boîtes pour 26 To de données.
Jusqu'en avril 2015, les boîtes étaient hébergées sur une baie de stockage accédée en NFS et gérées par Dovecot [2]. Avec la croissance des usages, un certain nombre de problèmes sont apparus. Nous avons donc dû faire évoluer la plate-forme.

2 Démarche

2.1 Problèmes rencontrés

Les problèmes les plus importants auxquels nous avons fait face sur l'ancienne plate-forme étaient :

— des problèmes de performance

— des problèmes de coût d'évolution

— des incidents réguliers de corruption d'index

2.1.1 Performance

Les utilisateurs percevaient des lenteurs lors de l'accès à des dossiers contenant plusieurs milliers de message. La version de Dovecot utilisée (1.2) s'appuie sur la liste des fichiers contenus dans le repértoire. Cette opération peut être lente en NFS si le cache n'est pas convenablement dimensionné.

2.1.2 Problème de coût

La matériel utilisé pour le stockage (Netapp) présente un coût au gigaoctet très élevé, difficilement comparable avec le coût d'un espace disque équivalent sur du stockage local.

2.1.3 Corruption d'index

La fonction de verrouillage de fichier n'étant pas correctement implémentée sur NFSv3, des incidents réguliers de corruption d'index se produisaient, rendant la boîte inutilisable pour l'utilisateur. Ces incidents sont structurels, liés aux choix technologiques : les interactions de plusieurs instances de Dovecot sur les mêmes données en NFS sont à l'origine de ces corruptions.

2.2 Étude de Dovecot 2

En étudiant le fonctionnement des nouvelles versions de Dovecot, nous nous sommes rendus compte que certains problèmes y étaient résolus. Dovecot 2 propose des solutions pour ne plus avoir de corruptions d'index et pour améliorer les performances. Une étude détaillée avec maquettage a été entreprise pour évaluer la pertinence d'évoluer vers Dovecot 2.

2.2.1 Séparation données/index

Dovecot 2 implémente de nouveaux formats de boîte (sdbox, mdbox). Contrairement à l'ancien format Maildir, ces formats apportent une séparation claire entre la liste des messages et leur contenu. Il n'est donc plus nécessaire de lister les messages par des opérations sur le système de fichiers. Un simple accès à l'index suffit. De plus, l'architecture de Dovecot a évolué pour garantir des accès exclusifs aux boîtes grâce la fonction "director" implémentée dans le proxy de dovecot.

2.2.2 Index locaux

Si on utilise les nouveaux formats, les messages et les index peuvent être séparés sur des disques différents. Mais lors de notre étude, il n'existait pas de mécanisme de réplication suffisamment robuste, et cette architecture présentait un risque important de perte de données.

2.2.3 Choix de ne pas migrer vers Dovecot 2

Le passage de Dovecot 1 à Dovecot 2 nécessitait de migrer de toute façon l'ensemble des données, car le changement du format des boîtes aurait été obligatoire.
De plus, la solution Proxy/Director de Dovecot 2 nous semblait manquer de maturité. En effet, la fonctionnalité était sortie relativement récemment lorsque nous l'avions testé. Enfin, après avoir assisté à la présentation [3] de Timo Siraïnen, auteur principal de Dovecot, au FOSDEM 2014, l'orientation stratégique prise par le projet pour utiliser du stockage objet dans le Cloud et l'absence de système de réplication pour de gros volumes nous apparaissaient peu adaptés à nos besoins.

2.3 Étude de Cyrus IMAP

Nous avons testé Cyrus IMAP juste après avoir fait une maquette de Dovecot. La maturité de la solution, la présence d'un système de réplication éprouvé et sa richesse fonctionnelle nous a intéressés. Puis, en apprenant qu'un grand fournisseur d'hébergement de messagerie, Fastmail, utilise Cyrus sur des millions de boîtes, cela nous a conforté dans le sentiment que cette solution répondait à nos besoins.

3 Description de la nouvelle solution

3.1 Architecture

Notre nouvelle solution s'appuie sur les principes suivants :

— un stockage décentralisé : chaque backend héberge localement des boîtes, afin de mieux répartir les risques en cas de dysfonctionnement d'un serveur, tout en apportant de bonnes performances ;

— une réplication applicative : le serveur gère lui même la réplication des boîtes avec une sémantique applicative, au niveau des opérations IMAP. Étant de plus haut-niveau que la réplication des blocs au niveau d'un système de fichiers, elle offre une plus forte garantie que la boîte soit répliquée correctement. En effet, placer la réplication au niveau applicatif permet de rejouer plus simplement des modifications après une coupure. Auparavant, le risque de désynchronisation menaçait l'intégrité de la boîte ou d'un dossier : un échec de la synchronisation impactant des blocs dans le système de fichier pouvait engendrer une corruption d'un index. La synchronisation applicative a une granularité plus fine et le risque se réduit effectivement à une perte de message, que le protocole de synchronisation sait tout à fait restaurer ;

— la facilité d'extension : l'augmentation de la volumétrie est une opération simple qui consiste à ajouter des backends sans modification de l'architecture (selon le principe du « scaling » horizontal) ;

— un faible coût au Go/à la boîte : les disques locaux sur les backends ont de très bon temps d'accès pour un coût faible. Pour obtenir des performances équivalentes, il est nécessaire d'investir dans une plate-forme centralisée coûteuse, comprenant de nombreux disques rapides, des contrôleurs de disque puissants et une infrastructure réseau très performante ;

— le dimensionnement s'inspire du modèle décrit dans le blog de Fastmail [4].

3.1.1 Descriptions générale

La plate-forme se compose de frontends et de backends de messagerie (voir figure 1). Les clients se connectent en IMAP ou en POP sur le frontend, qui relaie ensuite le trafic vers le backend correspondant.

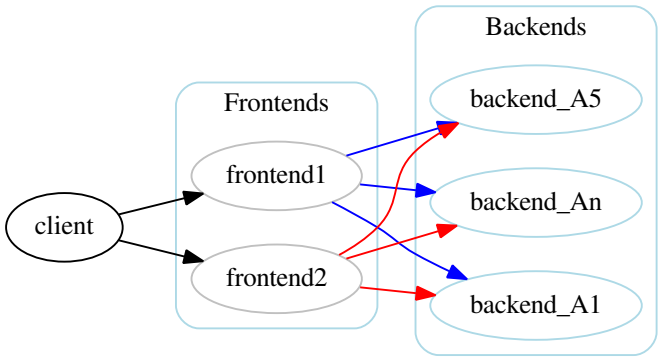

Figure 1 - Architecture générale

3.1.2 Redondance et réplication du service de messagerie

Chaque backend est maître pour un ensemble d'utilisateurs donné. Le backend maître envoie les modifications des boîtes qu'il héberge vers un backend esclave (voir figure 2). Si le backend maître, appelons-le A1, devient indisponible, une bascule est faite manuellement vers le backend esclave, A2. Ainsi, les prochaines connexions des utilisateurs hébergés sur le backend A1 sont relayées vers A2. La logique de la bascule manuelle se base sur le constat fait par les ingénieurs de Fastmail : le type de serveur déployé chez Fastmail est de très bonne qualité, avec des disques redondés, et la fiabilité matérielle est très élevée. De plus, la métrologie donne des indications qui permettent de déplacer les données proactivement. Leur expérience montre que le nombre d'interventions sur les serveurs est très faible. La bascule manuelle est un événement exceptionnel.

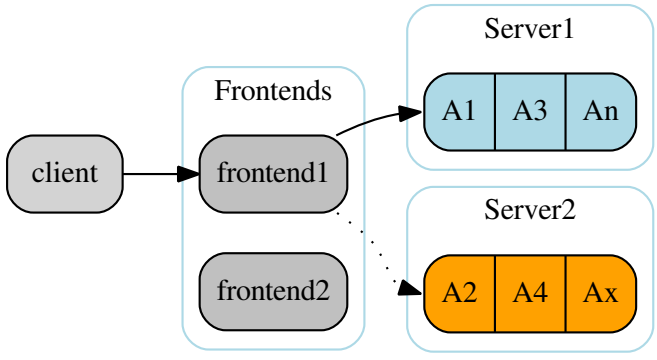

Figure 2 - Redondance et réplication du service de messagerie

3.1.3 Frontend

Load-balancer En amont de la plate-forme, un équilibreur de charge redondant, fonctionnant au niveau IP, reçoit les connexions TCP des clients et les redirige vers l'ensemble des frontaux en "round-robin".

Fonctionnement du frontend Sur chaque frontend, le module "Mail proxy" de Nginx relaie les connexions IMAP et POP. L'authentification du client se fait via un web service écrit en Lua. Ce web service est implémenté sous forme d'un script exécuté dans NGINX. Il effectue l'authentification de l'utilisateur, soit sur l'annuaire LDAP, soit sur le serveur CAS pour le webmail SOGo.
Sur chaque frontend, un serveur de cache Redis enregistre de manière chiffrée les mots de passe et les tickets CAS afin de réduire le nombre de requêtes à l'annuaire et au serveur CAS.

Base de donnée Cyrusroles L'aiguillage de la connexion depuis le frontend vers le backend qui héberge la boîte s'appuie sur les données fournies par le web-service d'authentification. Ce dernier interroge une base de données, Cyrusroles, qui contient l'association entre l'identifiant d'un utilisateur et le backend. La base est hébergée et synchronisée entre les frontends.

3.1.4 Backends

Chaque boîte est hébergée sur un backend donné, dit backend maître (voir figure 3). La boîte est stockée par Cyrus IMAP. Elle est répliquée au niveau applicatif sur un backend esclave. Le protocole de réplication, CSYNC, est intégré à Cyrus.

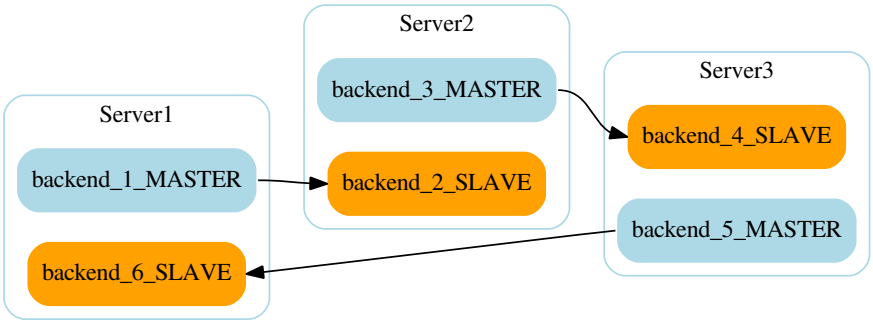

Figure 3 - Backends

3.1.5 Dépot des messages

Pour le dépôt des messages dans les boîtes, le chemin diffère légèrement de l'accès IMAP à la boîte. Sur le frontend, un serveur Postfix reçoit les messages en SMTP et les relaie vers les backends en LMTP. Au moment du dépôt, les filtres de messagerie de la boîte sont exécutés.

3.2 Implémentation

3.2.1 Hébergement des backends

Répartition des boîtes Dans le dimensionnement de Fastmail, on ne compte pas plus de 2000 boîtes par backend. Leur expérience montre que la gestion d'un petit nombre de boîtes sur un backend est plus simple et plus efficace en terme de ressources. Chaque serveur physique héberge plusieurs containers gérant chacun 2000 utilisateurs.

Nous avons suivi ce modèle : les 110 000 boîtes sont réparties de façon homogène sur 64 backends. Avec la redondance, nous avons donc déployé 128 backends au total sur 8 machines physiques, répartis équitablement sur deux sites disctincts. Nous avons donc 16 backends par machine.
La stratégie pour réduire les risques en cas d'incident sur un serveur est la suivante : un serveur physique se trouvant sur le premier site, héberge 8 backends « maître ». Les backends « esclave » correspondant sont répartis par paire sur 4 serveurs physiques distincts se trouvant sur l'autre site.

Virtualisation Le procédé de virtualisation utilisé est basé sur des « jails » sous FreeBSD. Ce sont des containers similaires à LXC sous Linux. Le choix de FreeBSD est lié à la disponibilité native du système de fichier ZFS. Ce dernier a une grande maturité et une grande richesse fonctionnelle : il nous permet notamment de gérer des volumes importants (13 To octets par machine dans notre cas), sans nécessiter de vérifier la cohérence du système de fichier (FSCK) lors d'un redémarrage du serveur.

Organisation des répertoires Chaque jail comporte les montages suivants :

- `/mail` & `/metadata` : contiennent les données et les métadonnées des boîtes ;
- `/config` : la configuration active de Cyrus ;
- `/base` : commun à toutes les jails d'un serveur physique, il contient le système ;
- `/` : squelette contenant essentiellement des liens symboliques vers les répertoires systèmes (`/etc` -> `/base/etc`, etc.).

3.2.2 Déploiement et exploitation

Déploiement initial L'outil de gestion de configuration et de déploiement automatisé, Opscode Chef, est de plus en plus utilisé dans les projets d'infrastructure de la Direction Informatique. Aussi avons-nous décidé de déployer l'ensemble des éléments de la plate-forme avec Chef.
La principale difficulté était de déployer les jails : Chef ne configure que la machine physique et non les containers. À l'origine, chaque jail était configurée à l'aide d'une fonction Chef (un provider) acceptant différents paramètres. La fragilité et la lenteur de cette méthode nous ont poussées à revoir tout le système de déploiement. Chef n'est desormais plus utilisé pour déployer les jails mais pour pré-approvisionner les fichiers de configuration de ces dernières. De plus, le disque système a été mutualisé entre toutes les jails.
Toutes les jails accèdent aux mêmes répertoires qui sont communs (`/usr`, `/etc`, etc.) dans l'image appelée "base". Les parties spécifiques à chaque jail sont stockées dans des volumes différents. Cette mise en commun permet de simplifier le processus de mise à jour des systèmes : seul un système par machine physique est mis à jour.

Gestion de l'exploitation La gestion des 110000 comptes et la multiplicité des backends a nécessité la réalisation d'outils spécialisés. Ainsi, nous avons développé un logiciel basé sur IMAP et des commandes spécifiques à Cyrus permettant d'analyser l'état des boîtes de messagerie, de corriger des erreurs courantes ou encore de simplifier la restauration à la demande des courriers supprimés.

Supervision et métrologie La métrologie de la messagerie est assurée par l'outil Graphite [5] qui enregistre l'évolution de l'espace disque et les performances système 4. De plus, notre plateforme de supervision Nagios surveille le processus de synchronisation entre les backends, l'état des disques et la disponibilité générale du service.

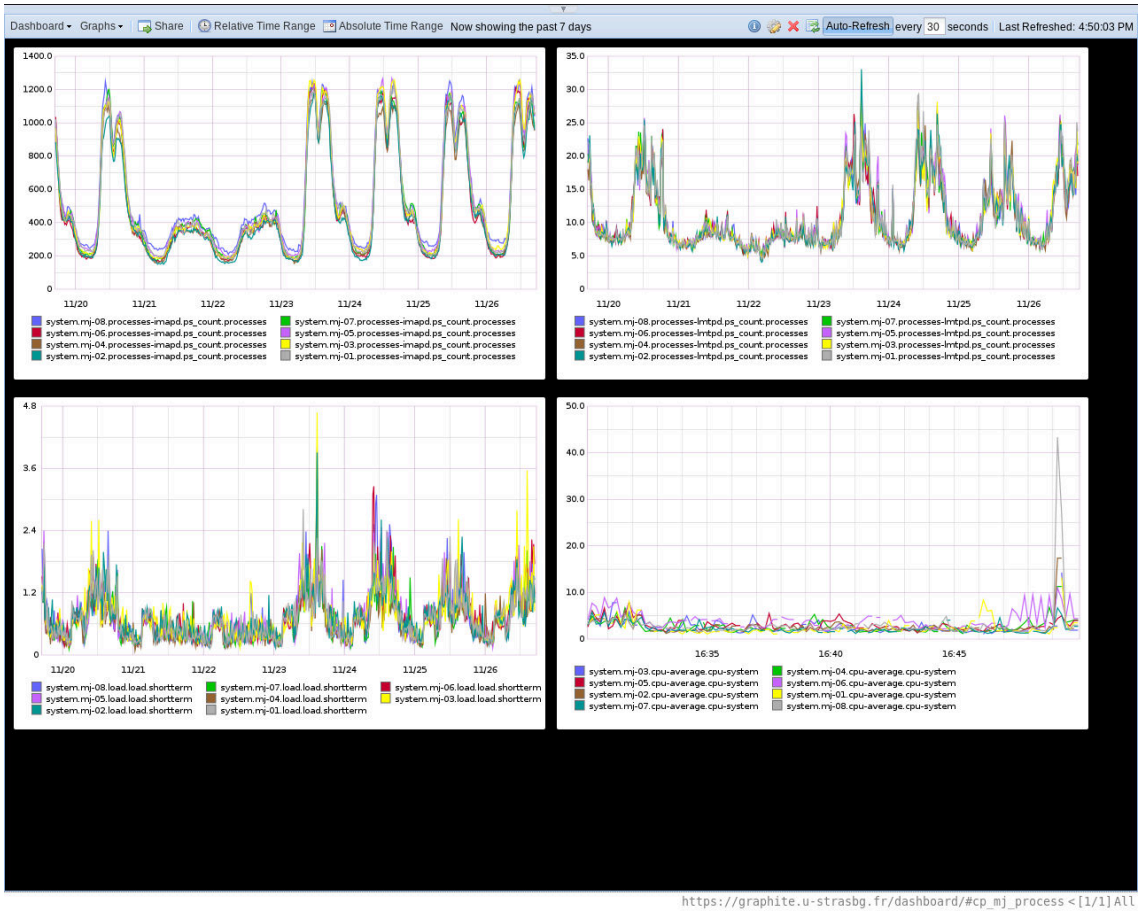

Figure 4 - Métrologie avec Graphite

4 Déroulement du projet

4.1 Démarrage du projet

Après une phase de pré-étude, le projet a véritablement démarré en avril 2014. Nous avons fait des maquettes de certains aspects de la plate-forme, et notamment de ZFS (les différents niveaux de raidZ, les clone de dataset etc.), du déploiement avec Chef, des jails et de la configuration de Cyrus. Dans cette phase, nous avons mis au point l'architecture et la stratégie de migration. Nous avons commencé à écrire des recettes de déploiement. Parallèlement, nous avons dimensionné la plate-forme et commandé le matériel.

4.2 Proxy de migration

Les phases de conception et de codage se sont poursuivies jusqu'à l'été 2014, où une opération de maintenance préparatoire a été effectuée. Nous avons mis en place des élements d'infrastructure temporaires :

— un proxy imap/pop de migration, basé sur Nginx, capable d'aiguiller les connexions vers l'ancienne ou la nouvelle plate-forme en fonction de la valeur d'un attribut dans l'annuaire pour un utilisateur donné ;

— un relais de messagerie spécifique capable de router vers l'ancienne ou la nouvelle plate-forme.

La première version du proxy de migration avait des problèmes de performance qui nous ont obligés à réécrire la logique d'aiguillage en Lua. En effet, Nginx intègre un module qui embarque un interpréteur Lua capable de faire la compilation à la volée (Lua Jit). Comparée à d'autres techniques d'appel de script, l'exécution de scripts en Lua Jit offre le plus haut niveau de performances. Cette technique s'est révélée si efficace qu'elle a été adoptée ensuite pour d'autres services de la plate-forme de messagerie.

4.3 Système unique et premiers déploiements

Les recettes de déploiement étaient quasiment terminées en début d'été mais les interactions entre Chef et les jails étaient peu satisfaisantes. Nous avons alors modifié l'architecture et réécrit en partie les recettes de déploiement pour fonctionner sur un répertoire système partagé entre toutes les jails d'une même machine physique.
Un fois ce point d'architecture réglé, la plate-forme a été complètement déployée à la rentrée 2014 :

— les serveurs physiques ;

— les jails ;

— le partitionnement ZFS ;

— des frontends basés sur le proxy IMAP/POP de Cyrus, incluant les fonctions Cyrus Murder (base centrale de localisation des boîtes sur les backend) et Mupdate (synchronisation des frontends).

4.4 Modification des frontends

Nous avons fait les premiers tests de charge avec des milliers de comptes, ce qui a fait apparaître des phénomènes inexpliqués de lenteur. Nous pensions que ces comportements pathologiques étaient liés à Cyrus Murder. Lors de la conception initiale, nous avions envisagé d'utiliser Nginx pour réaliser la fonction de proxy. Nous avons alors redéployé les frontends en remplaçant Cyrus Murder et Cyrus Mupdate par Nginx et une infrastructure de localisation et d'authentification que nous avons développée en propre. La seule fonctionalité perdue par la suppression de Cyrus Murder était les dossiers partagés (*shared folders*), qui n'étaient pas requis.

4.5 Préparation et migration

À partir de novembre 2014, il a été décidé que la Direction Informatique migrerait les données. Nous avons étudié la méthodologie et la planification permettant de rendre cette opération transparente pour les utilisateurs. Cette réflexion nous a amené à développer des scripts de synchronisation incrémentale. Ainsi, après une synchronisation initiale de 26 To, une mise à jour des données était réalisée quotidiennement. Par conséquent, la migration d'un utilisateur n'a nécessité que trois étapes :

— coupure des connexions à l'ancien serveur ;

— dernière synchronisation incrémentale ;

— modification du routage vers les nouveaux serveurs sur les frontaux.

Entre janvier et avril 2015 les comptes étudiants ont été migrés sur la nouvelle infrastructure. De même la migration des comptes personnels a eu lieu entre juin et juillet 2015. Cette bascule a été divisée en vagues dont la plus importante concernait 25000 comptes. Finalement, sur les 110000 comptes, moins d'une centaine d'utilisateurs ont fait face à des difficultés.

5 Retour d'expérience

L'expérience acquise après les premiers mois d'utilisation en production montre que la plateforme est bien adaptée à la charge et que cette dernière se répartit correctement sur les différents backends : environ 150 connexions concomitantes sur chacun des 64 backends.
D'un point de vue exploitation, ce changement d'architecture a permis de supprimer les problèmes liés au stockage centralisé : index de messagerie corrompus et latence lors de l'accès aux données.
Le changement de plateforme nous a imposé le développement de nouveaux outils spécifiques à cette architecture, ce qui nous a permis d'ajouter des fonctionnalités permettant d'automatiser la majorité des tâches d'exploitation depuis une console centrale ou via des web-services.

6 Conclusion

Notre nouvelle plate-forme d'hébergement de messagerie donne entière satisfaction, tant au niveau des performances que des fonctionnalités. Sa facilité d'extension par l'ajout de backends rendra son fonctionnement pérenne pour plusieurs années sans modification de l'architecture.
En terme de perspectives d'évolution, Cyrus, pierre angulaire de notre plate-forme, est en pleine mutation : une fondation [6] a été mise en place pour organiser le développement de nouvelles fonctions, et notamment des extensions de calendrier et une réplication actif-actif. De plus, les développeurs de Fastmail ont conçu une API web très prometteuse permettant d'accéder aux données de Cyrus, JMAP [7], sur laquelle pourront s'appuyer des webmails et d'autres outils. La fondation est soutenue par de grands utilisateurs de Cyrus, dont Fastmail et Netmail.
De notre côté, il reste de nombreux aspects à développer. Nous envisageons notamment de proposer un service simple de restauration de boîte directement accessible à l'utilisateur sous la forme d'une interface web. Nous allons également nous intéresser à l'évolution de notre webmail. Nous pouvons donc espérer des évolutions très riches à l'avenir.

Bibliographie

[1] Pierre David, Alain Zamboni, Philippe Pegon, et Jean Benoit. Evolution de l'architecture de messagerie d'osiris. Dans *Actes de la conférence JRES 2009*, Nantes. https://2009.jres.org/planning_files/article/pdf/124.pdf.

[2] Dovecot. http://www.dovecot.org.

[3] Timo Sirainen. Dovecot's way of scaling to millions of users. FOSDEM 2014. https://archive.fosdem.org/2014/schedule/event/dovecots_way_of_scaling_to_millions_of_users/.

[4] Bron Gondwana. Fastmail 2014 advent calendar. FastMail Blog. http://blog.fastmail.com/2014/12/01/fastmail-advent-2014/.

[5] Graphite. http://graphite.wikidot.com/.

[6] Announcing the cyrus foundation and development plans for 2015. http://asg.andrew.cmu.edu/archive/message.php?mailbox=archive.cyrus-announce&msg=223.

[7] Dec 23 : Jmap — a better way to email. http://blog.fastmail.com/2014/12/23/jmap-a-better-way-to-email/.