Synchronisation et partage de documents

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

Éric Laemmer
Direction Informatique, Université de Strasbourg
eric.laemmer@unistra.fr

Simon Piquard
Direction Informatique, Université de Strasbourg
simon.piquard@unistra.fr

Alain Heinrich
Direction Informatique, Université de Strasbourg
aheinrich@unistra.fr

Pascal Geoffroy
Direction Informatique, Université de Strasbourg
pascal.geoffroy@unistra.fr

Résumé

Cette année, l'Université de Strasbourg a mis en place un service de synchronisation et de partage de documents. Ce nouveau service se base sur le logiciel « Seafile » et héberge actuellement plus de 11000 utilisateurs et 2 To de données.
Cet article décrit la mise en place et l'évaluation de Seafile. En premier lieu, nous décrivons le fonctionnement de Seafile et son déploiement en détaillant les principaux mécanismes des clients, les échanges avec le serveur et l'architecture utilisée. De plus, nous présentons le processus de gestion de comptes. Notamment, la gestion des comptes invités utilisés pour les travaux de collaboration avec des membres extérieurs à l'Université. Enfin, nous évaluons Seafile en termes de fonctionnalités, de complexité de gestion et de performances et ce en nous appuyant sur des exemples d'usages et sur notre retour d'expérience.

Mots clefs

synchronisation de documents, stockage, gestion de comptes

1 Introduction

Depuis des années, l'échange de documents entre enseignants, chercheurs, étudiants et administratifs croît de façon considérable. Jusqu'à présent, l'Université de Strasbourg ne proposait pas d'outils dédiés à cet usage. En effet, l'absence d'un tel outil a généré l'apparition de nombreux usages détournés débouchant parfois sur de mauvaises pratiques comme par exemple, l'utilisation de la messagerie pour le stockage de documents, l'hébergement de données sensibles chez des hébergeurs externes, etc. Ainsi, en 2014, l'Université de Strasbourg a décidé de mettre en place un outil de synchronisation et de partage de documents

afin de proposer une alternative efficace, pratique et adaptable aux besoins de nos utilisateurs.
Cet article présente la première phase de la mise en place du logiciel Seafile pour plus de 11000 enseignants, chercheurs et personnels. La première partie détaille le fonctionnement général de Seafile en détaillant les principaux mécanismes tels que le modèle de données, le processus de déduplication ou encore de synchronisation. Puis, la seconde partie décrit la gestion des comptes, notamment la gestion des comptes invités permettant de travailler en collaboration avec des personnes extérieures à l'Université de Strasbourg. Ensuite, la troisième partie fournit une courte évaluation de Seafile. Enfin, nous concluons en présentant les différentes évolutions possibles telles que l'utilisation d'un stockage décentralisé utilisant « Ceph ».

2 Problématique

Aujourd'hui, nous disposons souvent de multiples appareils comme un ordinateur du bureau, un ordinateur personnel, un téléphone, une tablette, etc. Or, il est devenu fréquent de transférer des données de l'un à l'autre de ces appareils, afin de pouvoir consulter ou éditer un document aussi bien au travail que dans le train ou le soir à la maison. Outre ces synchronisations entre appareils personnels, le travail coopératif a aussi pris de l'ampleur et il est courant de travailler à plusieurs sur le même document.
Ainsi, nombreux sont ceux qui maintiennent manuellement à la fois une synchronisation entre différentes machines et une publication régulière pour diffuser la dernière version de documents. Face à ce constat, l'Université de Strasbourg a décidé de proposer un logiciel de synchronisation automatique et de partage de documents : le logiciel Seafile.
Seafile est composé d'une partie serveur qui centralise les données. Ces données sont consultées, téléchargées et mises à jour par des logiciels clients. Parmi ces logiciels clients, Seafile propose une interface web ainsi que des clients dédiés. Les clients dédiés s'installent sur les postes des utilisateurs et sont responsables :

— de récupérer les dernières versions des documents disponibles sur le serveur ;
— et de mettre en ligne les dernières modifications des documents effectuées sur le poste de l'utilisateur.

Les utilisateurs travaillent sur une copie locale des données permettant un usage hors ligne. Une fois connecté, le client Seafile va calculer les modifications et gérer les conflits. De son côté, le serveur gère les accès concurrents dans le cas où différents clients essayent de mettre à jour simultanément le même document. Seafile utilise également des mécanismes pour minimiser l'utilisation de la bande passante et l'espace de stockage.
Enfin, avec ces outils se posent les questions des performances et de passage à l'échelle en termes de gestion des comptes et de masse de données, ainsi que des mécanismes pour assurer la disponibilité du service.

3 Fonctionnement général de *Seafile*

Seafile est un service qui permet la récupération et le dépôt de documents. Chaque utilisateur de ce service a un compte lié a une adresse email. Avec ce compte, un utilisateur gère une collection de dépôts appelés bibliothèques. Chaque bibliothèque contient une arborescence de répertoires et de fichiers. Ces bibliothèques sont hébergées sur le serveur et peuvent être répliquées sur un ou plusieurs postes clients. Un utilisateur peut alors éditer sur son poste les fichiers, qui seront ensuite synchronisés sur le serveur grâce au client Seafile. Ceci permet à n'importe quel autre client de récupérer automatiquement la dernière version des fichiers.
De plus, les bibliothèques et répertoires peuvent être partagés avec d'autres utilisateurs. Ainsi, ces derniers disposent automatiquement de la dernière version des documents.
Enfin, chaque élément de l'arborescence peut être diffusé via un lien web. Cela permet à des utilisateurs n'ayant pas de compte pour ce service de pouvoir télécharger, ajouter ou remplacer des documents via l'interface web.

3.1 Côté client

Seafile peut être utilisé côté client, soit via une interface web, soit via une application dédiée et installée sur les postes utilisateurs. L'application dédiée permet de sauvegarder une configuration afin de se connecter automatiquement au serveur, ceci afin que les différentes synchronisations puissent se faire continuellement en tâche de fond. En outre, l'application permet de définir les répertoires sur le poste utilisateur hébergeant une copie locale des bibliothèques présentes sur le serveur. L'édition d'un fichier se fait donc sur la copie locale, permettant le travail hors-ligne. La synchronisation se fera une fois la connexion à nouveau active.
Du point de vue extensibilité le client dédié permet de répartir la charge des mécanismes de synchronisation, de gestion des conflits et de déduplication des données. Ainsi, le serveur économise de la bande passante et du temps de calcul.
Enfin, d'une façon générale, la gestion des données et leur synchronisation est une adaptation de certains modèles de données et mécanismes de « git ».

3.1.1 Modèle de données et mécanisme de synchronisation

Le modèle de données est basé sur la notion de dépôts git qui correspondent pour Seafile à une bibliothèque identifiée par un *Universally Unique IDentifier (UUID)*.
Le client travaille sur une branche *locale*, alors que le serveur contient la branche *master*. Les modifications sont enregistrées comme une version, ou *commit*, sur la branche *locale*. Le client va ensuite gérer la récupération de la dernière version de la branche *master* et la fusionner avec la branche *locale* avant de déposer sur le serveur le résultat de cette fusion (cf. [1] et figure 1).

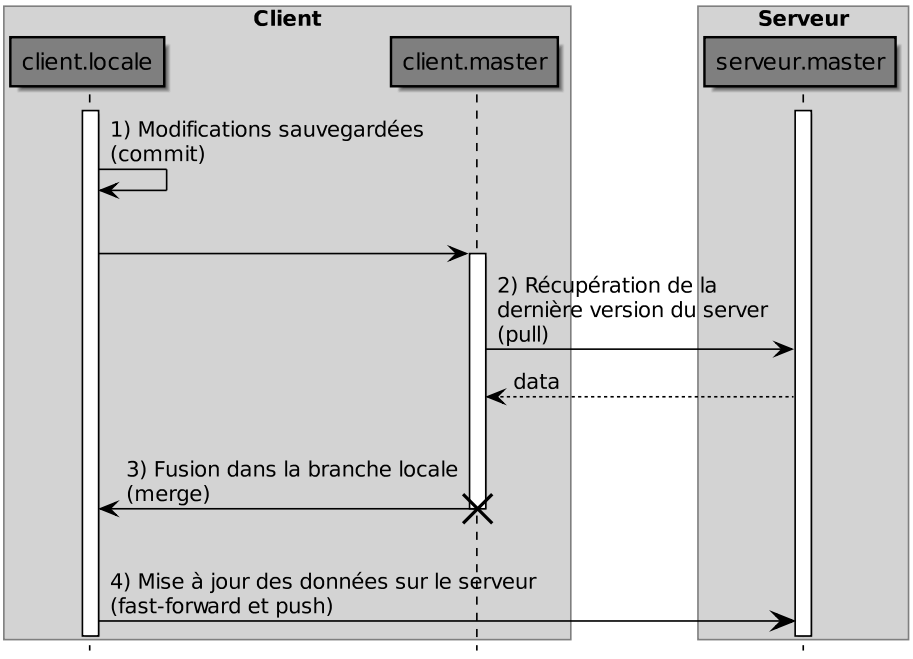

Figure 1: Mécanisme de synchronisation du client

Pour cela, l'application surveille les modifications des fichiers dans la sous-arborescence des bibliothèques synchronisées. Le client Seafile utilise des outils comme `inotify` sous Linux ou une vérification régulière sous Windows pour détecter ces modifications.
Même sans modifications locales, l'application vérifie régulièrement que les données locales correspondent bien à la dernière version présente sur le serveur et la télécharge si besoin.

3.1.2 Gestion des conflits

Contrairement à git, les « commit » se font automatiquement. Cela implique qu'une gestion des conflits doit également être automatisée afin de pouvoir au moins synchroniser les données. En cas de conflit, outre ceux déjà gérés par git, Seafile crée une copie du fichier édité et y ajoute un suffixe pour le différencier. Une fois cette nouvelle version synchronisée, reste à l'utilisateur le choix de conserver les différentes copies, de les supprimer ou de régler les conflits manuellement.

3.1.3 Déduplication des données

Avant de transmettre et stocker les données, Seafile utilise un mécanisme de déduplication afin d'économiser la bande passante et l'espace disque utilisé. Ce mécanisme découpe chaque fichier en blocs d'une moyenne de 1 Mo et utilise un algorithme de *Content Defined Chunking (CDC)* [2] pour déterminer quels blocs ont été modifiés entre deux versions du fichier.
Ainsi, seuls les blocs ayant été modifiés sont transmis et donc stockés sur le serveur (cf. figure 2).

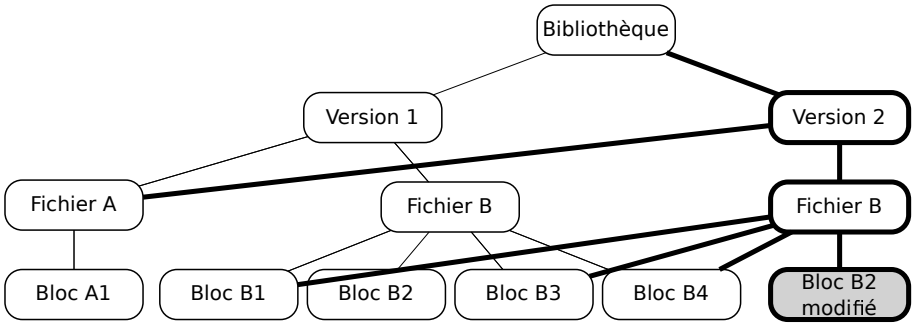

Figure 2: Déduplication des données

3.2 Côté serveur : service et architecture

Le service Seafile permet de gérer une base d'utilisateurs possédant une liste de bibliothèques et dont les données sont stockées et versionnées. Ce service est fourni par une infrastructure dont l'architecture (cf. figure 3) à l'Université de Strasbourg est la suivante :

Répartiteurs de charge : Un répartiteur de charge, redondé, basé sur « HAProxy » répartit la charge sur deux backends.

Backends : Deux backends hébergent les services de Seafile dont le service web et le service de gestion des données. Ces services gèrent la base d'utilisateurs et vérifient l'authentification des utilisateurs en connexion avec l'annuaire LDAP de l'université. Le service Seafile gère également les accès concurrents des clients et enregistre les modifications d'abord sur l'espace de stockage dédié, puis les métadonnées complémentaires dans la base de données. Les accès sont améliorés par un système de cache via le service « memcache ».
Les backends des notre plate-forme sont des machines vituelles composées de quad-cores à 2.2 GHz et 16 Go de mémoire vive.

Bases de données : Elles stockent les métadonnées : les droits et liens de partage, les groupes, les quotas des utilisateurs, etc.

Stockage : Les serveurs de stockage hébergent les données et les métadonnées les plus critiques comme le propriétaire et l'historique des blocs de données.

En attendant la prochaine évolution, le stockage est effectué en « NFS ». Un total de 14 To a été alloué sur des baies de stockage FreeNAS.

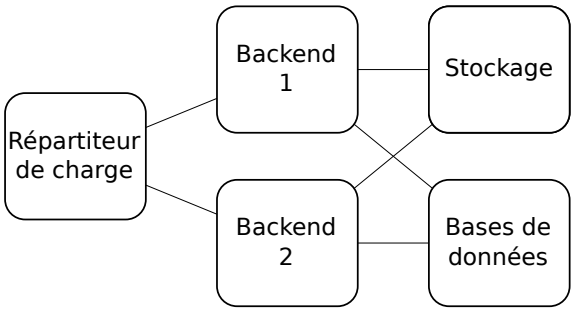

Figure 3: Infrastructure mise en place

3.2.1 Stockage : données et métadonnées

D'un point de vue volume, la base de données est assez légère comparativement au volume du stockage. En effet, la base de données fait actuellement 85 Mo pour 2 To de données stockées.
Nous ne détaillons ici que la structure des données et métadonnées de l'espace de stockage. Les métadonnées sont enregistrées dans le répertoire `seafile-data` où le sous-répertoire `storage` est lui-même subdivisé en 3 parties [3] :

commits : Contient une description au format `json` de chaque modification effectuée. Cette description comprend notamment un identifiant de *commit*, la date de cette modification, l'*UUID* de la bibliothèque ou encore l'identifiant du *commit* précédent.
Cette description contient également l'identifiant de l'entrée de la structure *fs* décrite ci-dessous.

fs : Contient une description en données binaires de l'arborescence permettant d'associer les blocs de données au *commit*.

blocks : Contient les données découpées en blocs d'en moyenne 1Mo. Un bloc est stocké dans un répertoire dont le nom correspond à l'empreinte « SHA1 » de ses données. Cela permet entre autre de vérifier l'intégrité des données en cas de corruption.
Les blocs contiennent les données en clair, sauf pour les bibliothèques chiffrées.

Nous avons déjà vu que le client utilise un algorithme de *Content Defined Chunking (CDC)* pour ne transmettre que les blocs modifiés. De même au niveau stockage, lorsqu'une nouvelle version d'un fichier est commitée, seuls les blocs modifiés sont écrits sur le disque. L'ancienne version du bloc est conservée afin de garder un historique des modifications. Cette technique permet de limiter l'espace disque utilisé.

4 Gestion des comptes

Pour la première année de service, la décision a été prise de ne créer que 11000 comptes, chacun pourvu de 15 Go de quota. La provision de ces comptes se fait via un filtre LDAP.
Cependant, il est important de pouvoir travailler en commun sur des documents avec des personnes externes : par exemple pour des collaborations de recherche. Ainsi, nous avons défini et implémenté un mécanisme de gestion des comptes invités.

4.1 Comptes invités

Les comptes invités sont des comptes externes au système d'information de l'université. Ils sont créés pour permettre à des utilisateurs internes de partager de façon synchrone des bibliothèques avec des collaborateurs externes à l'université : par exemple dans le cadre de travaux de recherche.
En étudiant la possibilité de créer des comptes invités, deux prérequis majeurs sont apparus :

Ouvert à tous : Tous les utilisateurs internes doivent pouvoir soumettre et obtenir de façon totalement automatique la création de comptes invités.
Pour cela, nous avons développé un programme qui reçoit ces demandes via un formulaire web et crée les comptes demandés via l'*API REST* de Seafile.

Maîtriser la volumétrie des disques : Ces créations de comptes ne doivent pas générer une augmentation de l'espace de stockage prévu initialement. Ainsi, les comptes invités sont créés avec le quota minimum : soit 1 octet.
Cela empêche donc l'utilisateur d'un compte invité de créer des bibliothèques qui lui sont propres et donc d'abuser de ce mécanisme.
Cependant, cela n'empêche pas à l'utilisateur d'un compte invité de travailler sur une bibliothèque partagée par un utilisateur interne. En effet, dans ce cas, l'espace utilisé sera décompté sur le quota du propriétaire de la bibliothèque.

Cette stratégie présente l'avantage de déléguer à la fois la création des comptes invités et le contrôle de l'espace disque utilisé. En effet, il revient à chaque utilisateur interne :

— de créer ses comptes invités ;

— de partager ses bibliothèques ;

— et de surveiller l'usage fait avec ces partages.

À tout moment, un utilisateur peut supprimer selon les besoins, soit son partage, soit entièrement sa bibliothèque.

5 Évaluation

Dans cette partie, nous présentons une brève évaluation de Seafile afin d'en montrer les principales caractéristiques en termes de fonctionnalités, de complexité de gestion et de performances.

5.1 Fonctionnalités

D'un point de vue fonctionnalité Seafile répond aux principales attentes pour ce type de service :

— synchronisation automatique de documents ;

— partage de documents et de bibliothèques ;

— publication de documents via des liens web publiques ou protégés par mot-de-passe ;

— multiples clients : site web, clients pour ordinateurs (Linux, Mac et Windows), téléphones portables et tablettes (Android et iOS) ;

— actions automatisables via l'API REST. Il faut noter que celle-ci n'est pas complète mais s'enrichit de versions en versions.

5.2 Complexité de gestion

La complexité de gestion des comptes et quotas n'est pas simple à évaluer. Si nous avons créé sans problème un mécanisme de gestion des comptes invités, il manque à Seafile une gestion de rôles pour affecter des quotas et des droits spécifiques à certains groupes d'utilisateurs.
Par contre, concernant la gestion des données, il est appréciable que Seafile permette à l'utilisateur de non seulement créer, éditer et supprimer ses données, mais aussi de disposer d'une corbeille à partir de laquelle il peut restaurer les répertoires et fichiers supprimés ou restaurer une ancienne version présente dans l'historique. Le seul point qui nécessite l'intervention de l'administrateur est la restauration entière d'une bibliothèque.

5.3 Performances

Pour évaluer les performances de Seafile, nous décrivons ici deux scenarii :

— un scenario théorique en comparant la masse de données transférées via Seafile et via la messagerie électronique ;

— un scenario pratique analysant les temps de calcul et de transfert de données en fonction du nombre et de la taille des fichiers.

5.3.1 Seafile et messagerie électronique

Définissons le scenario où Alice envoie un document de 5 Mo à Bob.
Pour la messagerie électronique, si on ne prend pas en compte l'encodage MIME, le coût de ce transfert peut-être approximé à :

10 Mo transmis sur le réseau : 5 Mo entre Alice et le serveur et 5 Mo entre le serveur et Bob.

20 Mo d'espace disque utilisés : 10 Mo de fichiers sauvegardés (5 Mo chez Alice et 5 Mo chez Bob) et 10 Mo de données sur les serveurs (5 Mo dans les messages envoyés pour Alice, et 5 Mo dans les messages reçus pour Bob).

Pour Seafile, si on ne prend pas en compte les métadonnées, le coût de ce transfert peut-être approximé à :

10 Mo transmis sur le réseau : 5 Mo entre Alice et le serveur et 5 Mo entre le serveur et Bob.

15 Mo d'espace disque utilisés : 10 Mo de fichiers sauvegardés (5 Mo chez Alice et 5 Mo chez Bob) et 5 Mo de données sur les serveurs (une seule copie partagée).

Si maintenant Alice envoie une nouvelle version du document avec une correction ne provoquant le changement que d'une ligne.
Pour la messagerie électronique :

10 Mo transmis sur le réseau : 5 Mo entre Alice et le serveur et 5 Mo entre le serveur et Bob.

10 Mo supplémentaires d'espace disque utilisés : 10 Mo de données sur les serveurs (5 Mo dans les messages envoyés pour Alice, et 5 Mo dans les messages reçus pour Bob). 0 Mo pour les données locales, car nous pouvons espérer qu'Alice et Bob ont écrasé le fichier précédent.

Pour Seafile :

2 Mo transmis sur le réseau : 1 Mo entre Alice et le serveur et 1 Mo entre le serveur et Bob. On suppose ici que la modification ne concerne qu'un bloc de données.

1 Mo supplémentaires d'espace disque utilisés : 1 Mo de données sur les serveurs (seul le bloc modifié est dupliqué).

Au total, la messagerie aura utilisé 20 Mo de volume réseau et 30 Mo d'espace disque, alors que Seafile se limitera à 12 Mo de volume réseau et 16 Mo d'espace disque. Ainsi, cet exemple décrit bien l'avantage en termes de performances, d'utiliser un service comme Seafile pour un travail coopératif. De plus, si l'économie effectuée est efficace dès le premier échange et s'amplifie avec les versions du document, il est notable que ces économies sont également proportionnelles au nombre de collaborateurs se partageant le fichier.

5.3.2 Analyse du temps de calcul et de transfert

Pour cette évaluation, nous avons mis en ligne des séries de fichiers afin de comparer les performances de Seafile en fonction du nombre et de la taille des fichiers.
Ainsi, notre jeu de tests se compose de fichiers de 1 Ko, 10 Ko, 100 Ko, 1 Mo, 10 Mo, 100 Mo, 1 Go et 5 Go. Ces fichiers sont transmis par groupe de 1, 10, 100, 1000, 10000 et 100000. Parmi ces combinaisons, seuls les tests générant moins de 10 Go ont été joués.
Les résultats montrent principalement que Seafile est moins efficace en terme de données transmises par seconde pour des petits fichiers. Par exemple (cf. figure 4), transmettre un fichier de 1 Go prend 125 secondes, alors que transmettre 100000 fichiers de 10 Ko, soit un total de 1 Go, prend 1846 secondes : soit un passage de 64Mb/s à environ 4Mb/s.
En effet, quand un grand nombre de petits fichiers sont transmis, cela génère plus de métadonnées et requiert également plus d'inodes sur le système de fichiers du serveur de stockage. Ce point est important car ces petits fichiers constituent la majorité des fichiers de nos utilisateurs.

Taille des fichiers	1 Ko	10 Ko	100 Ko	1 Mo	10 Mo	100 Mo	1 Go	5 Go
Pour 1 seul fichier								
Précalcul du commit (s)	8	3	3	2	4	4	30	134
Transmission des données (s)	2	2	2	2	3	14	125	580
Total (s)	10	5	5	4	7	18	156	715
Pour 10 fichiers								
Précalcul du commit (s)	3	3	4	7	5	35	312	
Transmission des données (s)	2	2	2	2	14	122	1217	
Total (s)	5	5	6	10	19	157	1530	
Pour 100 fichiers								
Précalcul du commit (s)	6	3	4	5	37	301		
Transmission des données (s)	3	4	4	14	122	1186		
Total (s)	9	7	8	20	159	1487		
Pour 1000 fichiers								
Précalcul du commit (s)	5	6	11	51	339			
Transmission des données (s)	13	13	23	132	1212			
Total (s)	18	20	34	184	1551			
Pour 10000 fichiers								
Précalcul du commit (s)	65	77	139	531				
Transmission des données (s)	106	114	214	1484				
Total (s)	171	191	353	2016				
Pour 100000 fichiers								
Précalcul du commit (s)	886	873	1597					
Transmission des données (s)	1052	1846	3349					
Total (s)	1920	2721	4948					

Figure 4: Résultats des tests de synchronisation

6 Comparaison avec d'autres solutions

Pendant l'étude préalable du projet, nous avons envisagé l'utilisation d'autres logiciels tels que Owncloud (testé en version 6.0.2) et Pydio (testé en version 5.0.3). Le choix de Seafile (testé en version 2.1.4) a été fait notamment sur la base :

— d'une enquête sur l'ergonomie auprès d'utilisateurs ;

— de la facilité d'administration ;

— et d'un test de performances.

Critère subjectif par excellence, les utilisateurs ont privilégié l'ergonomie de Seafile et Owncloud, écartant Pydio car étant souvent décrit comme confus.
Concernant la facilité d'administration, même s'il y a peu de différences vraiment marquantes, le point le plus sensible concerne la gestion des groupes :

Pydio **:** La gestion des groupes est quasiment inexistante : un individu ne peut faire partie que d'un seul groupe.

Owncloud **:** L'interface de gestion des groupes devient lente dès qu'on dépasse les 50 personnes et est inutilisable pour la partie concernant le LDAP.

Seafile **:** Pas de problèmes spécifiques : à noter que chaque usager gère ses propres groupes [1].

Enfin, une évaluation sommaire des performances basée sur la synchronisation de 9999 fichiers de 57,5 Ko montre que sur une architecture de tests Seafile nécessite 3 minutes, Owncloud 3 heures et Pydio 11 heures. Même succincte, cette étude montre des résultats et préférences principalement au bénéfice de Seafile, motivant ainsi notre choix.

7 Conclusion

Pour échanger et travailler de façon coopérative sur des documents, disposer d'un outil de synchronisation automatique et de publication est devenu une nécessité. En choisissant de mettre en place le logiciel Seafile, l'Université de Strasbourg fournit un tel outil à ses 11000 enseignants, chercheurs, et personnels.
Nous avons montré que ce service était adapté à de tels besoins aussi bien en termes de fonctionnalités, de gestion que de performances. Nous avons également mis en place un mécanisme de gestion des comptes invités afin de permettre des collaborations avec des personnes extérieures à l'université.
L'évolution de ce service passera certainement par une ouverture à un public plus large, avec une gestion automatisée des groupes. Cette ouverture nécessitera sûrement une évolution de l'architecture, notamment par un stockage distribué utilisant « Ceph ».

Bibliographie

[1] Synchronization algorithm | seafile server manual. http://manual.seafile.com/develop/sync_algorithm.html.

[2] Athicha Muthitacharoen, Benjie Chen, et David Mazieres. A low-bandwidth network file system. Dans *Symposium on Operating Systems Principles*, pages 174–187, 2001.

[3] Data model | seafile server manual. http://manual.seafile.com/develop/data_model.html.

1. Depuis la version 4.1.0, *Seafile* supporte la synchronisation des groupes via le LDAP.