View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by HAL Mines Nantes

Quel système de stockage pour les architectures Fog ?

Bastien Confais, Adrien Lèbre, Benoît Parrein

To cite this version:

Bastien Confais, Adrien Lèbre, Benoît Parrein. Quel système de stockage pour les architectures Fog ?. Compas'2016, Jul 2016, Lorient, France. <hal-01376292>

HAL Id: hal-01376292

https://hal.archives-ouvertes.fr/hal-01376292

Submitted on 4 Oct 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Compas'2016 : Parallélisme/ Architecture / Système
Lorient, France, du 5 au 8 juillet 2016

Quel système de stockage pour les architectures Fog ?

Bastien Confais[1], Adrien Lebre[2] et Benoît Parrein[1]

[1]Université de Nantes, IRCCyN UMR CNRS 6597,
Polytech Nantes, rue Christian Pauc BP 50609, 44306 Nantes Cedex 3, France

[2]INRIA, LINA UMR CNRS 6241,
Mines de Nantes, 4 Rue Alfred Kastler, 44300 Nantes, France

Résumé
Dans ce papier, nous nous intéressons au stockage distribué dans un contexte de « Fog Computing ». Nous définissons un ensemble de critères (localité des données, confinement du trafic, fonctionnement en mode déconnecté, mobilité des utilisateurs) que doit satisfaire un système de stockage adapté à ce contexte. Nous vérifions *in vivo* sur la plateforme Grid'5000 si les deux systèmes de stockage en mode objet Rados et IPFS respectent en particulier le confinement du trafic réseau entre les sites de Fog. Ce critère a un impact majeur sur les performances en écriture et en lecture. Par l'expérience, nous concluons que les deux systèmes comportent des lacunes et nécessitent des améliorations, notamment au niveau de la gestion des métadonnées. La méthode peut être étendue à l'ensemble des critères énoncés ainsi qu'à d'autres solutions logicielles de stockage.

1. Introduction

L'« Internet des objets » apporte de nouvelles contraintes informatiques : un grand nombre de périphériques ayant des ressources limitées et un besoin de calculs intensifs à faible latence. L'informatique en nuage, plus communément appelée « Cloud Computing » n'est pas adaptée pour répondre aux problématiques de latence car elle repose sur quelques centres de données. L'approche « nuage bas » connue sous le nom de « Fog Computing » a été proposée en 2012 par Cisco [4]. L'idée est de déployer des ordinateurs (serveurs) répartis géographiquement en bordure du réseau pour effectuer les calculs à faible latence et d'utiliser le Cloud (« nuage haut ») pour les calculs n'ayant pas cette contrainte. Les serveurs composant le Fog sont répartis sur différents sites. Chaque site pouvant être par exemple un point de présence du réseau ou une station de base (*Base Station Controller*) d'un réseau cellulaire.
Cette idée a été reprise dans plusieurs articles : Aazam *et al.* [1] proposent d'utiliser une passerelle entre les objets connectés et l'Internet pour agréger les données avant de les envoyer dans le Cloud. Byers *et al.* [5] proposent également d'utiliser plusieurs niveaux intermédiaires de Fog pour optimiser les calculs, le stockage et l'utilisation du réseau par les objets connectés. En termes d'applications, Dubey *et al.* [9] proposent d'utiliser le Fog pour traiter des données de santé.
Dans ce travail, nous cherchons à savoir si des solutions de stockage existantes peuvent être utilisées dans ce contexte ou bien s'il faut les améliorer voire en développer de nouvelles. Pour cela, nous vérifions expérimentalement sur la plateforme de tests Grid'5000, si Rados [16], le

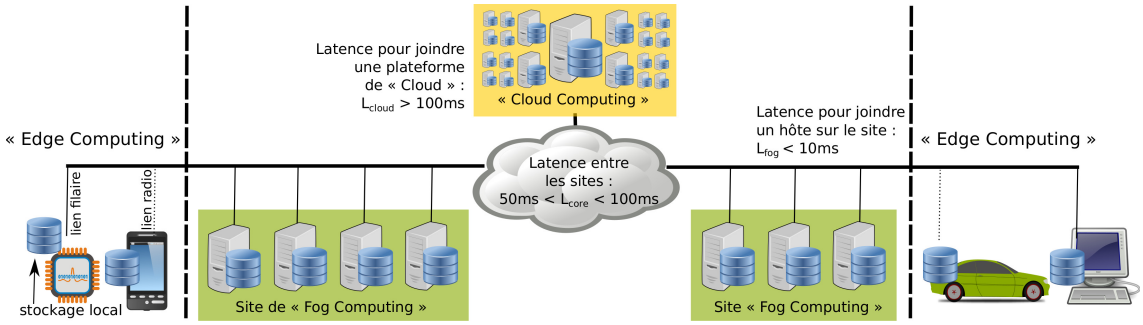


FIGURE 1 – Interconnexion des architectures de type Cloud, Fog et Edge avec les latences correspondantes.

module de la solution Ceph dédié au stockage d'objets et IPFS (InterPlanetary File System) [2], un système de stockage objet réutilisant les concepts de BitTorrent, confinent le trafic réseau entre les sites. Ce critère est pour nous important car cela signifie que les lectures et écritures sont indépendantes sur chaque site : si un site a une charge plus importante que les autres, le temps pour écrire ou lire un objet localisé sur un autre site ne doit pas augmenter. Dans la suite de l'article, les « objets » désignerons l'unité de stockage (et non les périphériques de « l'Internet des objets » qui seront désignés sous le terme d'« objets connectés »).
Après avoir défini un modèle de stockage en mode fog computing et donné une liste de cinq critères (localité des données, confinement du trafic réseau, fonctionnement en mode déconnecté, support de la mobilité et passage à l'échelle) que doit satisfaire un système de stockage adapté au Fog dans la partie 2, et après avoir introduit Rados et IPFS dans la partie 3, nous allons vérifier expérimentalement dans la partie 4 si les deux systèmes respectent la contrainte de confinement du trafic réseau.

2. Modèle de stockage en mode Fog et définitions

La figure 1 présente les éléments de l'infrastructure générale : Cloud, Fog et Edge. La plateforme de Fog est répartie par définition sur un nombre important de sites (seuls deux sites sont représentés sur le schéma). Ces sites comportent un faible nombre de serveurs et gèrent un grand nombre de clients (situés pour la plupart dans la zone Edge). Les clients se connectent au site le plus proche en matière de latence. La latence entre les sites (notée L_{core}) est d'environ 50 ms et correspond à la latence d'un lien de type WAN [1] tandis que la latence entre les clients et les sites (notée L_{fog}) est inférieure à 10 ms et correspond à la latence d'un lien radio. La latence (notée L_{cloud}) pour contacter une plateforme de Cloud Computing est élevée (de l'ordre de 200 ms) [8][10] et imprévisible [17]. Ces valeurs seront utilisées dans la partie expérimentale de ce travail.

2.1. Hiérarchie des systèmes de stockage

Les latences entre les différents éléments permettent de voir l'infrastructure comme un système hiérarchique. Cette hiérarchie est également proposée par Bonomi *et al.* [3]. Les « objets connectés » font partie du bord du réseau (l'« Edge Computing »). Ces périphériques ont une capacité de stockage limitée mais qui est accessible immédiatement. Le Fog propose une capacité plus importante mais avec une latence plus élevée et le Cloud propose au prix d'une latence très élevée une capacité de stockage quasi infinie [17]. Comme proposé par Byers *et al.* [5] ou encore Dubey *et al.* [9], il est possible d'avoir plusieurs couches de Fog entre les « objets connec-

1. Wide Area Network

tés » et le Cloud. Luan *et al.* proposent que les données ayant une portée géographique sont les données qui devraient être déplacées du Cloud vers le Fog [13]. Bonomi *et al.* rappellent que nous devons tenir compte du type de mémoire [4]. Nous éviterons de stocker des données temporaires sur un support qui ne peut être écrit qu'un nombre très limité de fois. Un cas d'utilisation du stockage dans le Fog est le « Swap en tant que service ». Un périphérique mobile ayant une quantité de mémoire limitée peut utiliser l'espace de stockage fourni par le Fog comme « swap ». Dans ce cas de figure, il est déconseillé de stocker le swap sur la mémoire du périphérique situé à l'Edge.

2.2. Propriétés d'un système de stockage en mode Fog

Les caractéristiques que doit vérifier un système de stockage en mode Fog sont de notre point de vue (i) une localité des données, (ii) un confinement du trafic réseau, (iii) une continuité de fonctionnement lorsqu'un site est déconnecté, (iv) un support de la mobilité des utilisateurs ainsi que (v) un passage à l'échelle.
Pour la localité, un utilisateur doit toujours écrire sur le site le plus proche pour réduire au maximum le temps d'accès aux données. Le confinement du trafic est le fait qu'une écriture ou une lecture effectuée sur un site n'impacte pas les temps d'accès sur les autres sites. Le fonctionnement en mode déconnecté est le fait que les données d'un site déconnecté des autres soient accessibles aux utilisateurs du site. Le support de la mobilité est le fait que les données d'un utilisateur puissent être déplacées d'un site à un autre. Dans les grilles de calculs, les données sont relocalisées lorsque le client y accède. Dans le cas du Fog, il faut être capable de relocaliser les données en amont. Enfin, le critère de passage à l'échelle consiste à avoir un système qui supporte à la fois un très grand nombre de sites de Fog, un grand nombre de clients par site et un nombre important d'objets stockés.
Dans le cas de notre application cible (« Swap en tant que service »), les données devront être placées sur le site le plus proche, par exemple situé au niveau de l'antenne couvrant la cellule. Lorsque l'utilisateur se déplace, pour éviter que l'accès aux données ne devienne trop lent, celles-ci doivent être relocalisées sur le site de Fog situé dans la même cellule radio que le périphérique de l'utilisateur. Les données doivent rester accessibles à l'utilisateur même si la cellule radio est déconnectée des autres. Un système de stockage adapté au Fog doit trouver des compromis pour satisfaire ces caractéristiques contradictoires, notamment au niveau de la gestion des métadonnées. Par exemple, un grand nombre de sites doit pouvoir être supporté tout en restant performant. Cela ne permet pas, comme dans les grilles de calculs, l'utilisation, d'un serveur centralisé pour gérer les métadonnées.

3. Systèmes de stockage distribués candidats pour une architecture de type Fog

Nous avons choisi Rados [16] et IPFS [2] de par leur conception qui ne nécessite pas de serveur centralisé. Les systèmes de fichiers distribués traditionnels comme PVFS [6] ou HDFS [14] n'ont pas été concu pour un contexte multisites en plus de souvent reposer sur un serveur centralisé pour le stockage des métadonnées.

3.1. Rados

Rados [16] est une solution de stockage en mode objet réputée pour son algorithme CRUSH [15] permettant de localiser les données sans recourir à une communication distante.
Il utilise deux types de nœuds : des serveurs de stockage et des moniteurs. Les moniteurs ont pour rôle de maintenir et de distribuer aux serveurs de stockage et aux clients un arbre appelé « clustermap » qui contient la topologie du système. Les moniteurs utilisent l'algorithme

	Rados	IPFS
Localité des données	oui (règles de placement)	oui (nativement)
Fonctionnement en mode déconnecté	non (Paxos)	partiellement
Support de la mobilité	oui	oui
Passage à l'échelle	moyennement (Paxos)	oui

TABLE 1 – Caractéristiques satisfaites par Rados et IPFS.

Paxos [11] entre eux pour maintenir la consistance de ce « clustermap ».
Rados permet de créer des ensembles d'objets (« pools ») et des règles de placement permettant dans notre cas de placer les données d'un « pool » sur un seul site. Cela vérifie le critère de localité. En revanche, Rados ne permet pas le fonctionnement d'un site déconnecté des autres à cause des moniteurs. Un moniteur isolé ne peut ni être élu ni contacter le moniteur élu afin de distribuer le « clustermap ». Pour le critère de mobilité, Rados déplace automatiquement les données lorsqu'une règle de placement est modifiée. Enfin, le critère de passage à l'échelle a été évalué précédemment avec le système de fichiers Cephfs qui utilise Rados [7]. Ceci est résumé dans le tableau 1.

3.2. InterPlanetary File System

InterPlanetary File System [2] [2] est une solution de stockage en mode objet s'appuyant sur une table de hachage distribué (DHT [3]) Kademlia et sur le protocole BitTorrent [12], un protocole d'échange de données en mode pair à pair réputé pour sa simplicité et sa capacité à passer à l'échelle. La principale différence avec BitTorrent est que la liste des pairs est globale alors qu'elle est spécifique à chaque contenu dans BitTorrent. Utiliser une DHT pour stocker la localisation des objets a été proposé par Zhang *et al.* [17] dans un contexte semblable au nôtre mais n'a pas été évalué expérimentalement.
IPFS respecte le critère de localité puisque les données sont placées sur le serveur de stockage auquel est connecté le client. Si un site est déconnecté, les nœuds du site isolé ne peuvent plus localiser tous les objets. Toutefois, comme nous le verrons dans la section 4.2, la DHT n'est pas consultée lorsqu'un client envoie la requête directement au nœud stockant la donnée. Dans cette situation, il est possible pour un client d'accéder aux données en interrogeant tous les nœuds du site. Le critère de mobilité semble respecté puisqu'il est possible de demander explicitement à un nœud de stocker un objet. L'objet est téléchargé en utilisant le protocole BitTorrent. L'utilisation d'une DHT semble fournir à IPFS la capacité de passer à l'échelle.

4. Évaluation des systèmes de stockage

Dans cette partie, nous proposons d'évaluer si les systèmes de stockage Rados et IPFS respectent le critère de confinement de trafic, *i.e.* les lectures et les écritures effectuées sur un site n'impactent pas les temps d'accès aux données sur les autres sites : si un site est plus fortement sollicité, il ne faut pas que les temps d'accès aux données localisées sur les autres sites augmentent. Par ailleurs, seul le critère de confinement a été évalué dans ce travail.

4.1. Matériel et méthodes

La topologie utilisée pour l'évaluation est celle présentée dans la figure 2 : chaque site de Fog est composé d'un client et de deux serveurs de stockage (et d'un moniteur pour Rados). Les systèmes sont configurés pour satisfaire la localité des données : Pour Ceph, nous créons un

2. `http://ipfs.io/`
3. Distributed Hash Table

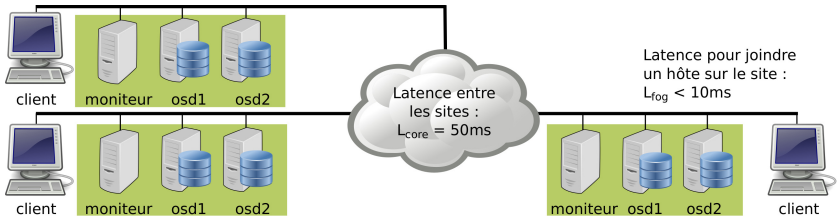


FIGURE 2 – Topologie utilisée pour le déploiement de Rados et d'IPFS. Un « OSD » (Object Storage Daemon) est un serveur de stockage. Les moniteurs ne sont utilisés que par Rados.

« pool » par site associé à une règle de placement pour stocker les données sur le site. IPFS stocke les données sur le serveur sur lequel elles sont envoyées. Comme le critère de localité est vérifié, la question du confinement du trafic entre les sites peut être posée.

L'expérience est conduite sur la plateforme de tests Grid'5000 en ajoutant des latences entre les machines à l'aide du programme `tc`[4]. Nous utilisons les latences définies précédemment, à savoir $L_{fog} = 10\,ms$ et $L_{core} = 50\,ms$. Les liens réseau entre les machines ont un débit de 10 Gbps.

Pour vérifier si le confinement du réseau est respecté, nous effectuons des opérations locales sur chaque site de Fog : les client écrivent de manière indépendante des objets sur leur site. Une fois les écritures réalisées, les caches sont vidés et ces mêmes clients relisent ce qu'ils ont écrits. Les objets ne sont pas répliqués. Ils ne sont stockés qu'en un seul exemplaire.

Nous relevons le temps mis pour effectuer l'opération sur chaque site ainsi que la quantité de données échangées entre les sites. Le but est d'étudier dans quelle mesure ces trafics et leur quantité impactent les autres sites. Dans le cas de l'écriture, les valeurs sont relevées après que les données aient été écrites sur le disque (`sync`). Nous posons l'hypothèse suivante : si le confinement est respecté, les temps et les quantités de données transitant entre les sites devraient être indépendants du nombre de sites.

Le scénario est exécuté plusieurs fois en faisant varier le nombre de sites ainsi que le nombre et la taille des objets. Nous utilisons 7 sites puis 11 sites avec 1 objet de 10Mo écrit par chaque client, 10 objets de 1 Mo, 100 objets de 1 Mo et 10 objets de 10 Mo. Chaque client écrit les objets en parallèle. Ces choix permettent de vérifier si écrire 10 Mo en un ou en 10 objets est équivalent. Nous testons également l'écriture de 10 objets de 1 Mo écrits séquentiellement par chaque client. Écrire les objets séquentiellement permet de vérifier que la latence est un problème puisque lors de l'écriture, toutes les latences sont cumulées.

Pour équilibrer la charge des serveurs avec IPFS, les client choisissent aléatoirement le serveur du site qu'ils vont utiliser pour écrire ou lire les données. Il est possible qu'un objet soit écrit sur un serveur et lu depuis l'autre serveur du même site.

IPFS insère de la redondance dans la DHT : pour chaque objet, la localisation est stockée sur 10 nœuds différents de la DHT. Afin de placer Rados et IPFS dans une situation comparable, nous modifions IPFS pour éviter cette réplication de métadonnées.

4.2. Résultats

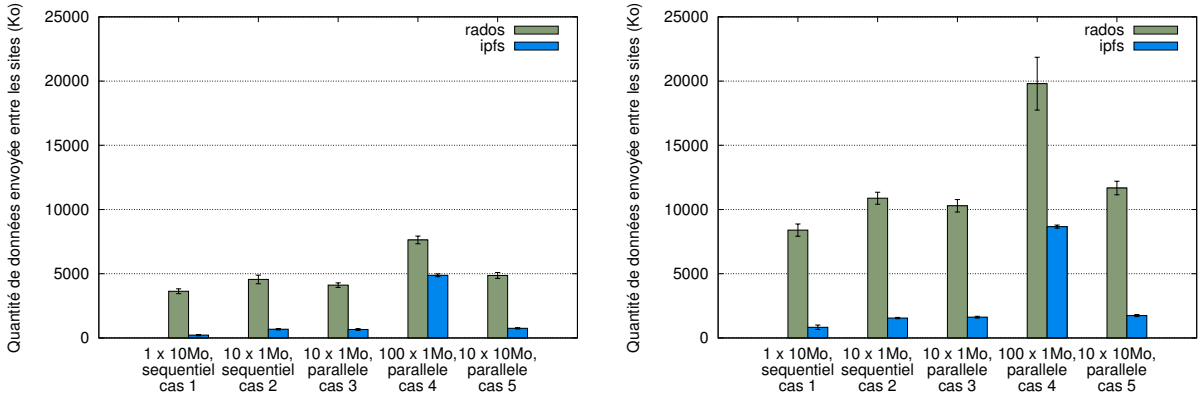
Le tableau 2 contient les temps moyens d'écriture et de lecture pour un site calculé selon la formule suivante :

$$\text{temps moyen} = \frac{\sum_{j=1}^{\#\text{itérations}} \frac{\sum_{i \in \text{sites}} \text{temps}_{ij}}{\#\text{sites}}}{\#\text{itérations}} \quad (1)$$

4. `http://www.linuxfoundation.org/collaborate/workgroups/networking/iproute2`

	Temps moyens d'écriture sur un site (en secondes)					**Temps moyens de lecture** sur un site (en secondes)				
Objets	1 × 10 Mo	10 × 1 Mo séquentiel	10 × 1 Mo parallèle	100 × 1 Mo parallèle	10 × 10 Mo parallèle	1 × 10 Mo	10 × 1 Mo séquentiel	10 × 1 Mo parallèle	100 × 1 Mo parallèle	10 × 10 Mo parallèle
	7 sites									
Rados	7,59	9,96	7,45	10,26	11,25	4,39	6,89	4,17	7,10	7,62
IPFS	7,48	11,52	7,01	12,18	8,91	4,19	8,39	3,65	8,28	6,18
	11 sites									
Rados	10,88	14,15	12,21	13,95	14,19	6,25	8,72	5,78	8,75	9,03
IPFS	8,79	12,95	8,38	14,60	10,75	4,50	8,73	4,07	8,81	6,95

TABLE 2 – Temps moyens **en secondes** pour écrire et lire l'ensemble des objets sur un site avec 7 sites et 11 sites. Les écarts types ne sont pas renseignés ici (tous inférieurs à 2 secondes).



(a) – 7 sites

(b) – 11 sites

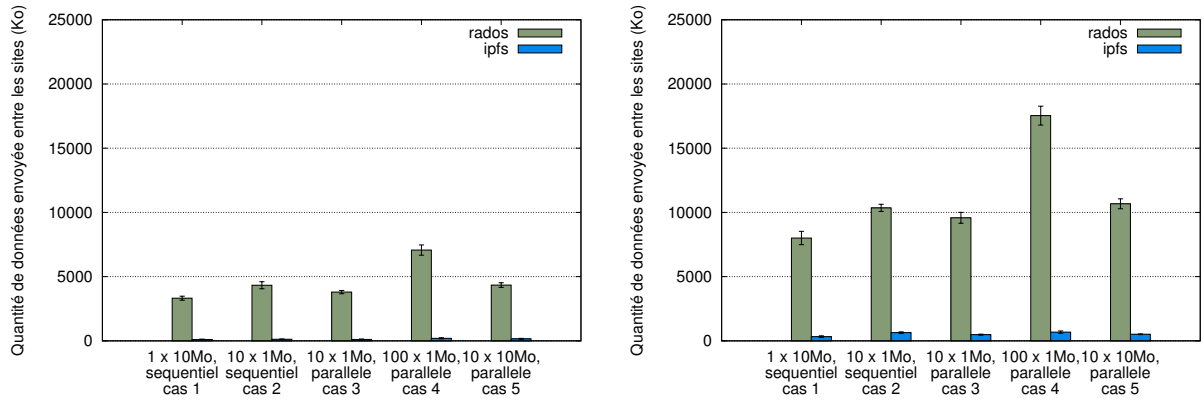
FIGURE 3 – Quantités de données moyennes échangées entre l'ensemble des sites pendant les opérations **d'écriture** (10 exécutions de chaque scénario).

La variable « $temps_{ij}$ » est le temps cumulé relevé sur le site i pour lire ou écrire l'ensemble des objets lors de l'itération j. Le tableau montre que Rados et IPFS ont des temps de d'écriture et de lecture similaires bien que les deux systèmes soient assez peu performants au vu des quantités manipulées. Comme pressenti, les temps d'accès augmentent avec le nombre de sites. Nous pouvons noter toutefois que ces dernières sont plus significatives pour Rados. Cela est dû au trafic entre les différents sites comme illustré sur les figures 3 et 4 et discuté ci-après.

Toujours au niveau des temps d'accès du tableau 2, nous pouvons également observer des différences entre les manipulations séquentielles et parallèles avec un surcoût du simple au double pour IPFS dans le cas des lectures (8,39 s vs 3,65 s et 8,73 s vs 4,07 s respectivement pour 7 et 11 sites). Ceci s'explique par le fait que pour chaque requête, IPFS doit interroger la DHT et donc potentiellement contacter un noeud distant.

Notons également que l'implémentation d'IPFS testée ne permet pas un traitement parallèle efficace. En effet, écrire 100 objets de 1 Mo par site avec IPFS prend 1,4 fois plus de temps que l'écriture de 10 objets de 10 Mo. (14,60 s vs 10,75 s pour 11 sites) Après recherche, nous avons découvert que l'implémentation ne permet de traiter que deux requêtes simultanément. En utilisant 10 objets de 10 Mo, l'écart entre Rados et IPFS se creuse au profit de ce dernier (par exemple pour 7 sites : 11,25 s vs 8,91 s).

Les figures 3 et 4 montrent le trafic échangé entre les sites. Ces trafics concernent les métadonnées car ils ne dépendent pas de la taille des objets. Ils sont principalement liés aux échanges entre les moniteurs pour Rados et à l'utilisation de la DHT pour IPFS.



(a) – 7 sites

(b) – 11 sites

FIGURE 4 – Quantités de données moyennes échangées entre l'ensemble des sites pendant les opérations **de lecture** (10 exécutions de chaque scénario).

Une analyse par régression nous indique pour Rados que les quantités transmises augmentent linéairement avec le nombre d'objets pour l'écriture et la lecture (et de façon quadratique avec le nombre de sites) [5]. Régulièrement, tous les moniteurs envoient une statistique d'utilisation au moniteur élu. Cette statistique comporte une partie de taille fixe et une partie dont la taille est proportionnelle au nombre d'objets écrits ou lus. Le moniteur élu agrège ces statistiques et les rediffuse. Cette stratégie semble clairement coûteuse lorsque le nombre de sites devient important. La stratégie des moniteurs semble être inadéquate pour le contexte du Fog. Ce phénomène explique également la différence observée entre les modes d'accès.Dans le cas d'un accès séquentiel, le test dure plus longtemps (9,96 s vs 7,45 s dans le tableau 2 pour 7 sites en écriture). Les statistiques échangées sont plus nombreuses ce qui explique la légère augmentation de trafic (4554 Ko vs 4115 Ko).
Pour IPFS, le trafic entre les sites pendant l'écriture semble être proportionnel au nombre de sites et au nombre d'objets manipulés ce qui correspond à un comportement normal pour une DHT. En lecture, le trafic entre les sites est moins important car la DHT n'est consultée que lorsque le serveur interrogé ne stocke pas l'objet. Avec deux serveurs par site, la DHT n'est consultée en moyenne que pour une requête sur deux.

5. Conclusion

Nous avons présenté une liste de critères pour un système de stockage en mode Fog puis nous avons cherché à vérifier si Rados et IPFS vérifiaient le critère du confinement du trafic réseau. Ces expériences montrent que Rados, avec le trafic entre les moniteurs et IPFS, avec sa DHT globale, entraînent des trafics entre les sites, même si pour IPFS, ces trafics sont moindres. Des mécanismes semblent nécessaires pour limiter le trafic entre les sites. Rados pourrait être déployé indépendamment sur chaque site avec des passerelles entre les sites [6]. Pour IPFS, l'ajout de tables de hachage distribuées locales à chaque site permettrait de limiter le contact avec les autres sites. Par ailleurs, dans un souci d'amélioration des temps d'accès d'autres solutions de stockage comme Cassandra pourront être testées et ce, au travers de l'ensemble des critères énoncés dans ce travail.

5. Une expérience avec 3 et 5 sites a été également réalisée.
6. `http://docs.ceph.com/docs/master/radosgw/federated-config/`

Bibliographie

1. Aazam (M.) et Huh (E.-N.). – Fog computing and smart gateway based communication for Cloud of Things. – In *Proceedings of the 2014 International Conference on Future Internet of Things and Cloud*.
2. Benet (J.). – *IPFS - Content Addressed, Versioned, P2P File System*. – Rapport technique, 2014.
3. Bonomi (F.), Milito (R.), Natarajan (P.) et Zhu (J.). – Fog computing : A platform for Internet of Things and Analytics. *In : Big Data and Internet of Things : A Roadmap for Smart Environments*, éd. par Bessis (N.) et Dobre (C.), pp. 169–186. – Springer International Publishing, 2014.
4. Bonomi (F.), Milito (R.), Zhu (J.) et Addepalli (S.). – Fog computing and its role in the Internet of Things. – In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*.
5. Byers (C. C.) et Wetterwald (P.). – Fog computing distributing data and intelligence for resiliency and scale necessary for iot : The Internet of Things. *Ubiquity*, vol. 2015, novembre 2015, pp. 4 :1–4 :12.
6. Carns (P. H.), Ligon, III (W. B.), Ross (R. B.) et Thakur (R.). – PVFS : A parallel file system for linux clusters. – In *Proceedings of the 4th Annual Linux Showcase & Conference - Volume 4*.
7. Confais (B.), Van Kempen (A.), David (S.), Parrein (B.) et Nachouki (M.-P.). – Distributed Filesystems comparison on the GRID'5000 cluster. – In *Third Sino-French Workshop on Information and Communications Technology (SFWICT)*, Nantes, France, juin 2015.
8. Couto (R. D. S.), Secci (S.), Campista (M. E. M.) et Costa (L. H. M. K.). – Network design requirements for disaster resilience in iaas clouds. *IEEE Communications Magazine*, vol. 52, n° 10, October 2014, pp. 52–58.
9. Dubey (H.), Yang (J.), Constant (N.), Amiri (A. M.), Yang (Q.) et Makodiya (K.). – Fog data : Enhancing telehealth big data through fog computing. – In *Proceedings of the ASE BigData & SocialInformatics 2015*.
10. Firdhous (M.), Ghazali (O.) et Hassan (S.). – Fog computing : Will it be the future of cloud computing ? – In *Third International Conference on Informatics & Applications, Kuala Terengganu, Malaysia*, pp. 8–15, 2014.
11. Lamport (L.). – Paxos made simple, fast, and byzantine. – In *Procedings of the 6th International Conference on Principles of Distributed Systems. OPODIS 2002, Reims, France, December 11-13, 2002*, pp. 7–9, 2002.
12. Legout (A.), Urvoy-Keller (G.) et Michiardi (P.). – *Understanding BitTorrent : An Experimental Perspective*. – Rapport technique, INRIA, Institut Eurecom, 2005.
13. Luan (T. H.), Gao (L.), Li (Z.), Xiang (Y.) et Sun (L.). – *Fog Computing : Focusing on Mobile Users at the Edge*. – Rapport technique, 2015.
14. Shvachko (K.), Kuang (H.), Radia (S.) et Chansler (R.). – The hadoop distributed file system. – In *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies*.
15. Weil (S. A.), Brandt (S. A.), Miller (E. L.) et Maltzahn (C.). – Crush : Controlled, scalable, decentralized placement of replicated data. – In *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*.
16. Weil (S. A.), Leung (A. W.), Brandt (S. A.) et Maltzahn (C.). – Rados : A scalable, reliable storage service for petabyte-scale storage clusters. – In *Proceedings of the 2nd International Workshop on Petascale Data Storage : Held in Conjunction with Supercomputing '07*.
17. Zhang (B.), Mor (N.), Kolb (J.), Chan (D. S.), Goyal (N.), Lutz (K.), Allman (E.), Wawrzynek (J.), Lee (E.) et Kubiatowicz (J.). – The Cloud is not enough : Saving IoT from the Cloud. – In *Proceedings of the 7th USENIX Conference on Hot Topics in Cloud Computing*.