Conception d'un système de partage de données adapté à un environnement de Fog Computing

Bastien Confais

Rapporteurs :
- Marcelo Dias de Amorim, Directeur de Recherche CNRS, LIP6, Paris
- Frédéric Desprez, Directeur de Recherche Inria, Grenoble

Examinateur :
- Sara Bouchenak, Professeur des Universités, INSA Lyon

Directeurs :
- Benoît Parrein, Maître de Conférences titulaire de l'HDR, Université de Nantes
- Adrien Lebre, Chargé de Recherche titulaire de l'HDR, IMT-A, Nantes

Contexte

Aujourd'hui, grâce à la virtualisation, les infrastructures de Cloud Computing apportent aux utilisateurs :

- Des ressources de calcul et de stockage quasi-illimitées ;
- Accessibles à tout moment, depuis n'importe quel endroit de la planète.

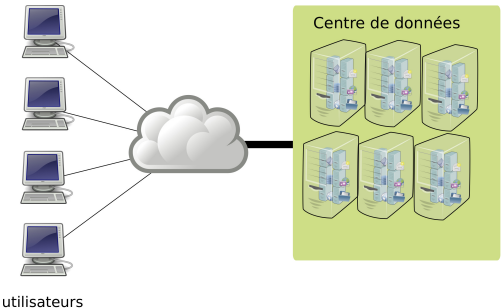


FIGURE 1 – Infrastructure de Cloud Computing.

Contexte - L'Internet des Objets

L'Internet des Objets est caractérisé par :

- un grand nombre d'objets ;
- souvent connectés par des liens radios ;
- qui peuvent être mobiles ;
- **qui ont des besoins de calcul à faible latence** ;
- malgré des ressources de calcul et de stockage limitées.

l'**Internet des Objets** est un nouveau défi :
plus de **8 milliards** [1] d'objets sont aujourd'hui connectés et ce chiffre ne fait qu'augmenter.

1. https://www.gartner.com/newsroom/id/3598917

Contexte - L'Internet des Objets

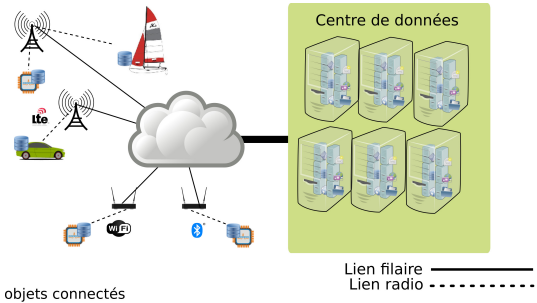


FIGURE 2 – Aperçu de l'Internet des Objets connecté à une infrastructure de Cloud Computing.

Les infrastructures de Cloud Computing font face à deux difficultés :

- Les centres de données sont situés **loin des utilisateurs** (impossible de réduire la latence) ;
- Trop de périphériques sont connectés à **quelques gros centres de données** (nécessite des liens réseaux avec des débits toujours plus élevés).

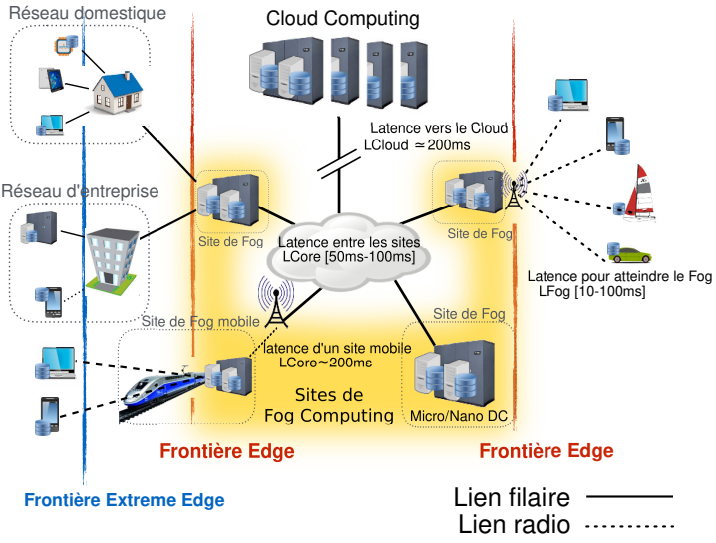


FIGURE 3 – Organisation générale des infrastructures de Cloud et de Fog Computing.

Introduction - Cas d'utilisation

- Ville intelligente
- Surveillance de l'état de santé
- Mise en cache de contenu
- Fonctions de virtualisation d'équipements réseau

Problème

Nous souhaitons créer une solution de stockage adaptée à un environnement de Fog Computing.

Les caractéristiques que nous attendons :

- Localité des données
- Confinement des trafics réseau
- Disponibilité des données en cas de partitionnement du réseau
- Support de la mobilité
- Passage à l'échelle

Plan de la soutenance

- **Contexte**
- **Introduction & problématique**
- **État de l'art**
- **Contributions** :
  1. Réduction des trafics réseau inter-sites pour les accès locaux
  2. Confinement des trafics réseau inter-sites pour les accès distants
  3. Utilisation simultanée des plateformes d'expérimentation Grid'5000 et FIT/IoT-Lab
- **Conclusion & perspectives**

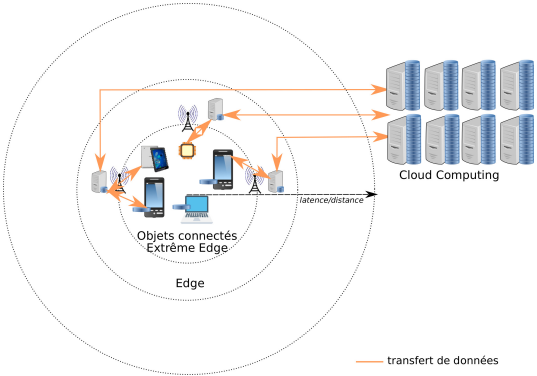
Informatique utilitaire : quelles infra-structures après le Cloud ?

Deux autres approches ont été proposées :

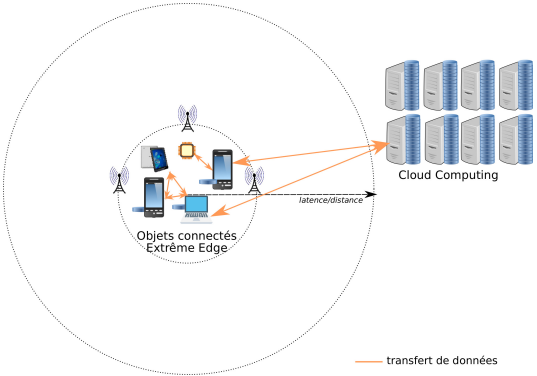
- Edge Computing (ou Cloudlets) [2]
- Extreme Edge Computing [3]

2. Niroshinie Fernando, Seng W. Loke et Wenny Rahayu (2013). « Mobile cloud computing : A survey ». Dans : *Future Generation Computer Systems* 29.1. Including Special section : AIRCC-NetCoM 2009 and Special section : Clouds and Service-Oriented Architectures, p. 84 -106

3. Ioana Giurgiu et al. (2009). « Calling the Cloud : Enabling Mobile Phones As Interfaces to Cloud Applications ». Dans : *Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware*. Middleware '09. Urbanna, Illinois : Springer-Verlag New York, Inc., 5 :1-5 :20



(a) – Approche Edge Computing



(b) – Approche Extrême Edge Computing

FIGURE 4 – Exemple de deux infrastructures montrant les interactions et les capacités de stockage de chaque équipement.

Modèle de stockage par objets

Les solutions de partage de données dans une infrastructure distribuée

La plupart des solutions de stockage distribué s'appuient sur **un serveur centralisé**.
(exemple : HDFS [4], Rozofs [5] ou encore Xtreemfs [6])

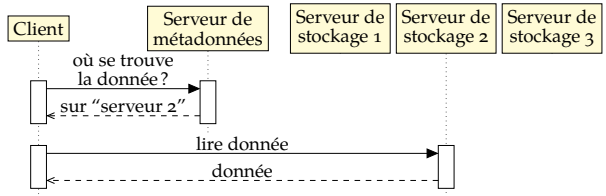


FIGURE 5 – Diagramme de séquence montrant le processus d'accès à une donnée.

4. K. SHVACHKO et al. (2010). « The Hadoop Distributed File System ». Dans : *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, p. 1-10

5. Dimitri PERTIN et al. (avr. 2014). « Distributed File System based on Erasure Coding for I/O Intensive Applications ». Dans : *4th International Conference on Cloud Computing and Service Science*. Barcelone, Spain

6. Felix HUPFELD et al. (déc. 2008). « The XtreemFS Architecture ; a Case for Object-based File Systems in Grids ». Dans : *Concurr. Comput. : Pract. Exper.* 20.17, p. 2049-2060

Modèle de stockage par objets

Le serveur de métadonnées limite fortement les performances :

- contacter le serveur de métadonnées pour chaque dossier traversé est coûteux ;
- le serveur de métadonnées est un point de défaillance du système ;
- dans un environnement multi-sites, cela implique de contacter un site distant.

Modèle de stockage par objets

Les solutions de stockage par objets (comme le service Amazon S3 [7]) :

- proposent un espace de nom « à plat » ;
- deux opérations simples :
  - **lecture** d'un objet ;
  - **écriture** d'un objet.
- pas de serveur de métadonnées.

7. Frederic P. Miller, Agnes F. Vandome et John McBrewster (2010). *Amazon Web Services*. Alpha Press

Positionnement des principales solutions de stockage distribué

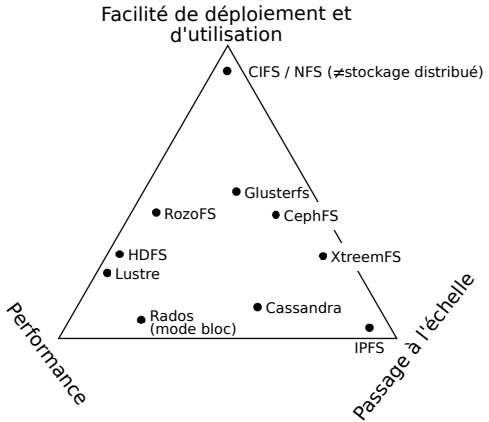


FIGURE 6 – Comparaison de différentes solutions de stockage distribué.

Quelques solutions de stockage par objets

Rados

Rados [8] est une solution de stockage par objets qui utilise la fonction de placement CRUSH (*Controlled Replication Under Scalable Hashing*) pour localiser les données.

CRUSH en quelques mots :

- utilise des fonctions de hachage ;
- s'appuie sur un arbre représentant la topologie ;
- prend en compte des contraintes de placement ;
- hachage consistant, lors d'un changement de topologie, seules les données stockées dans les branches impactées sont déplacées ;
- utilise le protocole Paxos pour maintenir l'arbre consistant.

8. Sage A. Weil et al. (2007). « RADOS : A Scalable, Reliable Storage Service for Petabyte-scale Storage Clusters ». Dans : *Proceedings of the 2Nd International Workshop on Petascale Data Storage : Held in Conjunction with Supercomputing '07*. PDSW '07. Reno, Nevada : ACM, p. 35-44

Cassandra

Cassandra [9] est une solution de stockage clé/valeur qui s'appuie sur une fonction de hachage et un protocole par *gossip*.

Au lieu d'avoir un moniteur, chaque nœud annonce à ses voisins la portion de l'espace de clés qu'il gère. Après propagation, chaque nœud a une vision de la topologie globale.

Comme dans Rados, le placement se fait par hachage (avec des contraintes).

9. Avinash Lakshman et Prashant Malik (avr. 2010). « Cassandra : A Decentralized Structured Storage System ». Dans : *SIGOPS Oper. Syst. Rev.* 44.2, p. 35-40

InterPlanetary FileSystem (IPFS)

IPFS [10] est une solution de stockage par objets qui s'appuie sur :

- Un protocole similaire à BitTorrent pour l'échange de données entre les nœuds ;
- Une table de hachage distribuée de type Kademlia pour stocker la localisation de chaque objet.

IPFS utilise des objets immuables et les objets sont stockés sur les serveurs contactés par les utilisateurs. Il n'y a donc pas de stratégie de placement.

10. Juan Benet (2014). *IPFS - Content Addressed, Versioned, P2P File System*. Rapp. tech. Protocol Labs, Inc.

IPFS - Table de hachage distribuée

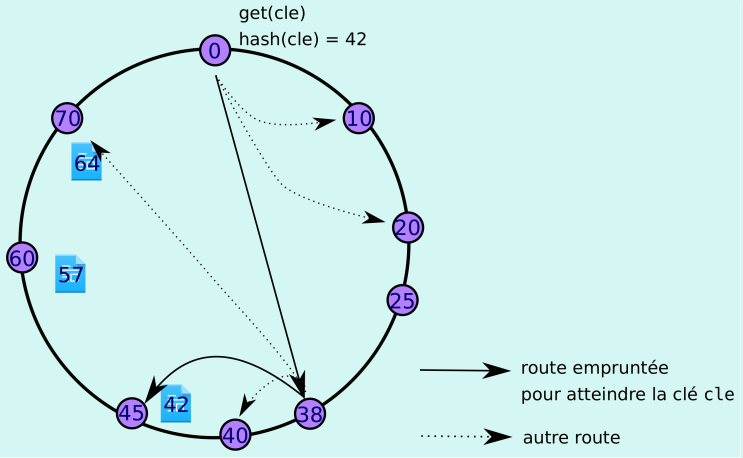


FIGURE 7 – Exemple de routage dans une table de hachage distribuée, lorsque le nœud d'identifiant 0 accède à l'enregistrement ayant pour clé `cle`.

IPFS - Diagramme de séquence

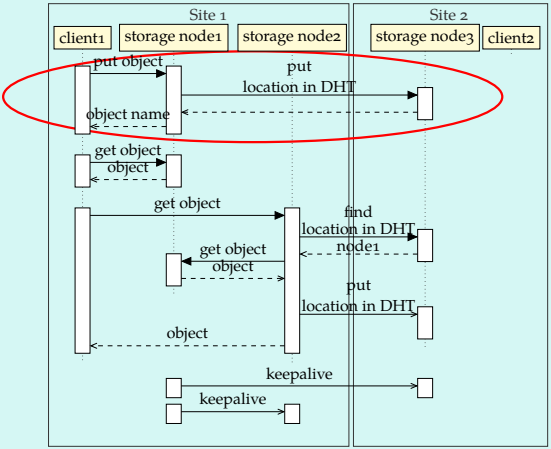


FIGURE 8 – Diagramme de séquence montrant le trafic réseau observé lorsque IPFS est déployé dans un environnement de Fog Computing.

IPFS - Diagramme de séquence

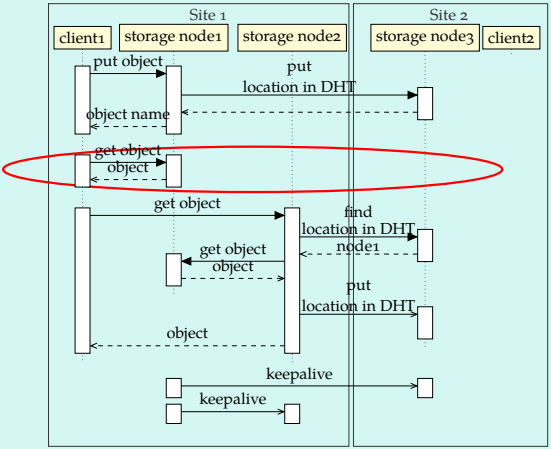


FIGURE 8 – Diagramme de séquence montrant le trafic réseau observé lorsque IPFS est déployé dans un environnement de Fog Computing.

IPFS - Diagramme de séquence

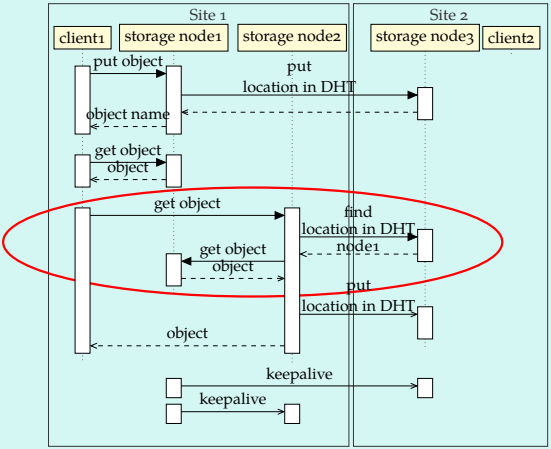


FIGURE 8 – Diagramme de séquence montrant le trafic réseau observé lorsque IPFS est déployé dans un environnement de Fog Computing.

IPFS - Diagramme de séquence

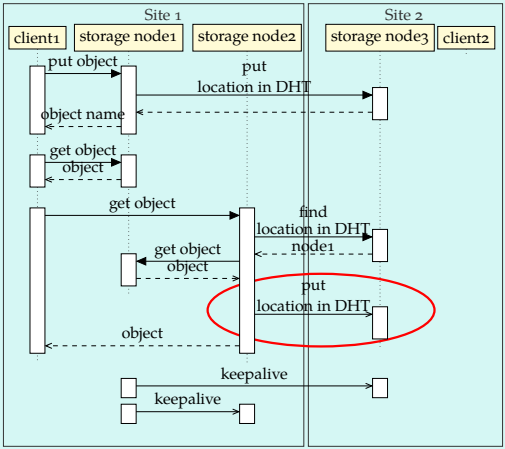


FIGURE 8 – Diagramme de séquence montrant le trafic réseau observé lorsque IPFS est déployé dans un environnement de Fog Computing.

Conclusion

	Rados	Cassandra	IPFS
Localité des données	**Oui***	**Oui***	Oui
Confinement du trafic réseau	**Partiellement**	Oui	**Non**
Fonctionnement en cas de partitionnement du réseau	**Partiellement**	Oui	**Partiellement**
Support de la mobilité	**Partiellement**	**Non**	Nativement
Passage à l'échelle	**Non**	Oui	Oui

* : l'utilisation des *pools* et des *keyspaces* peut devenir problématique face a de nombreux utilisateurs.

TABLE 1 – Caractéristiques satisfaites pour Rados, Cassandra et IPFS.

Ce travail préliminaire a été présenté dans la conférence IEEE CloudCom 2016 [11] et a fait l'objet d'une publication dans le journal Springer TLDKS 2017 [12].

11. Bastien CONFAIS, Adrien LEBRE et Benoît PARREIN (déc. 2016a). « Performance Analysis of Object Store Systems in a Fog/Edge Computing Infrastructures ». Dans : *CloudCom*. Luxembourg, Luxembourg, Décembre 2016

12. Bastien CONFAIS, Adrien LEBRE et Benoît PARREIN (août 2017c). « Performance Analysis of Object Store Systems in a Fog and Edge Computing Infrastructure ». Dans : *Transactions on Large-Scale Data- and Knowledge-Centered Systems XXXIII*. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 40-79

Réduction des trafics réseau inter-sites pour les accès locaux

Description du problème

La table de hachage distribuée, utilisée par IPFS est accédée lors d'accès à des objets stockés sur le site local.

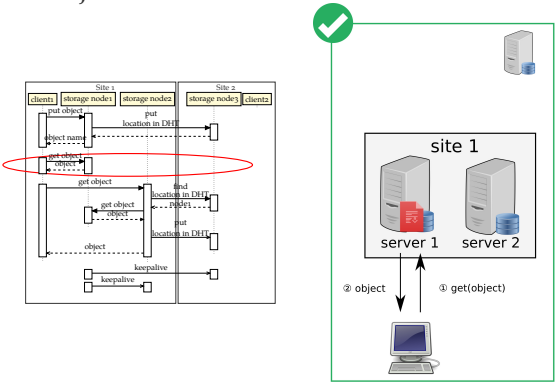


FIGURE 9 – Accès à un objet depuis un nœud qui le stocke.

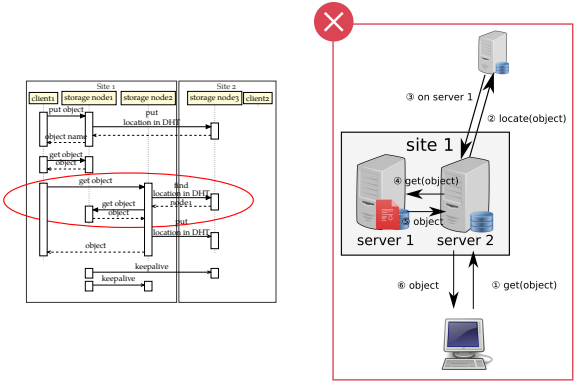


FIGURE 10 – Accès à un objet depuis un nœud qui ne le stocke pas.

Solution proposée

Nous proposons de coupler IPFS à un système de fichiers distribué déployé localement sur chaque site.

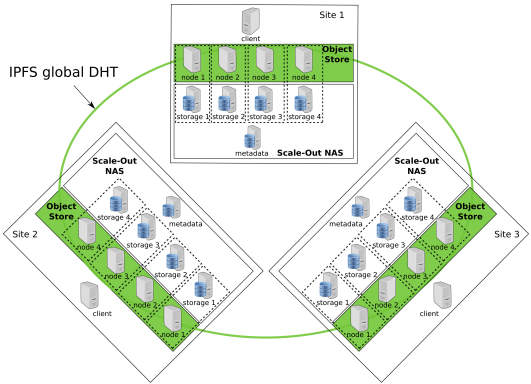


FIGURE 11 – Architecture générale de la solution.

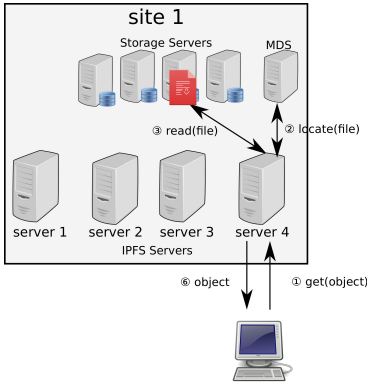


FIGURE 12 – Accès à un objet lorsque IPFS est couplé à un *Scale-Out NAS*.

Avantages / Inconvénients

Avantages :

- Supprime les trafics inter-sites lorsque l'objet est stocké sur le site local en évitant de consulter la table de hachage distribuée ;
- Meilleure disponibilité en cas de partitionnement du réseau.

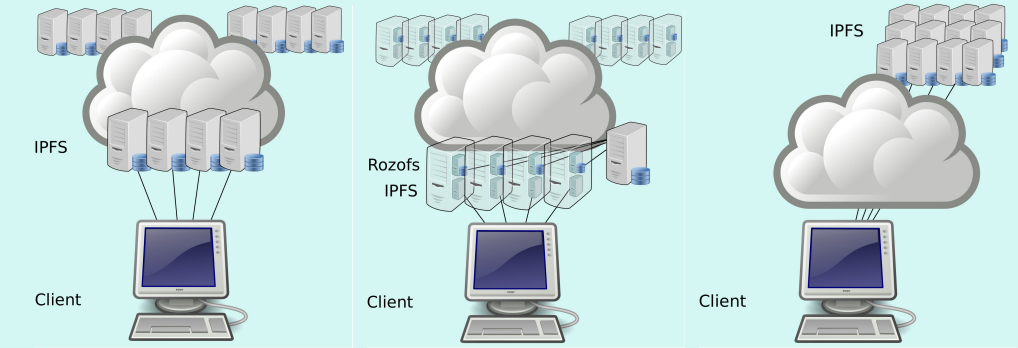
Inconvénients :

- Déploiement plus complexe ;
- Peut générer un goulet d'étranglement pour les accès distants.

Évaluation expérimentale

Nous évaluons notre approche avec le système de fichiers distribué **RozoFS**.
L'infrastructure de Fog Computing est émulée sur la plateforme Grid'5000.

Nous comparons 3 situations :



(a) – IPFS déployé seul (b) – Couplage IPFS+RozoFS (c) – IPFS dans le Cloud

FIGURE 13 – Différents environnements pour l'évaluation d'*InterPlanetary FileSystem*.

Évaluation expérimentale

- 3 sites chacun composé de :
  - 1 client ;
  - 4 serveurs de stockage IPFS/RozoFS colocalisés ;
  - 1 serveur de métadonnées RozoFS.
- Latences réseau :
  entre les clients et le site de Fog : $L_{Fog} = 10\,\text{ms}$
  entre les sites de Fog : $L_{Core} = 50\,\text{ms}$
  entre les serveurs d'un même site : $L_{Site} = 0,5\,\text{ms}$
  entre les clients et le Cloud : $L_{Cloud} = 200\,\text{ms}$
- Mesures :
  - Le temps pour lire et écrire un objet ;
  - La quantité de trafic réseau échangée entre les sites pendant les opérations de lecture ou d'écriture (grâce à `iptables`).

- Scénario :
  1. Chaque client écrit sur son site local ;
  2. Chaque client relit les objets qu'il vient d'écrire.
- Pour chaque objet, le nœud IPFS sollicité est sélectionné aléatoirement (loi uniforme) ;
- Tous les objets sont accédés en parallèle ;
- Un `tmpfs` est utilisé pour stocker les données de RozoFS ou d'IPFS ;
- L'exécution de chaque test est répétée 10 fois.

Évaluation expérimentale - Temps d'accès

Nous faisons varier la taille des objets écrits entre 256 Ko et 10 Mo.
Chacun des 3 clients écrit 100 objets sur son site (300 objets manipulés au total).

	Temps moyens en **écriture** (secondes)			Temps moyens en **lecture** (secondes)		
Taille / Déploiement	256 Ko	1 Mo	10 Mo	256 Ko	1 Mo	10 Mo
Déployé seul						
Couplé avec RozoFS						
Déployé dans le Cloud						

TABLE 2 – Temps d'accès moyens pour écrire et lire un objet lorsque la charge est de 100 objets stockés sur chaque site (topologie de 3 sites). Les écarts types sont d'environ 0,1 s.

Évaluation expérimentale - Temps d'accès

Nous faisons varier la taille des objets écrits entre 256 Ko et 10 Mo.
Chacun des 3 clients écrit 100 objets sur son site (300 objets manipulés au total).

	Temps moyens en **écriture** (secondes)			Temps moyens en **lecture** (secondes)		
Taille / Déploiement	256 Ko	1 Mo	10 Mo	256 Ko	1 Mo	10 Mo
Déployé seul	0,33	1,07	3,92	0,29	0,50	1,98
Couplé avec RozoFS	0,33	1,08	3,97	0,19	0,36	1,83
Déployé dans le Cloud	2,29	5,55	27,58	1,57	2,62	11,24

TABLE 2 – Temps d'accès moyens pour écrire et lire un objet lorsque la charge est de 100 objets stockés sur chaque site (topologie de 3 sites). Les écarts types sont d'environ 0,1 s.

Évaluation expérimentale - Temps d'accès

Nous faisons varier la taille des objets écrits entre 256 Ko et 10 Mo.
Chacun des 3 clients écrit 100 objets sur son site (300 objets manipulés au total).

	Temps moyens en **écriture** (secondes)			Temps moyens en **lecture** (secondes)		
Taille / Déploiement	256 Ko	1 Mo	10 Mo	256 Ko	1 Mo	10 Mo
Déployé seul	0,33	1,07	3,92	0,29	0,50	1,98
Couplé avec RozFSs	0,33	1,08	3,97	0,19	0,36	1,83
Déployé dans le Cloud	2,29	5,55	27,58	1,57	2,62	11,24

TABLE 2 – Temps d'accès moyens pour écrire et lire un objet lorsque la charge est de 100 objets stockés sur chaque site (topologie de 3 sites). Les écarts types sont d'environ 0,1 s.

Évaluation expérimentale - Temps d'accès

Nous faisons varier la taille des objets écrits entre 256 Ko et 10 Mo.
Chacun des 3 clients écrit 100 objets sur son site (300 objets manipulés au total).

	Temps moyens en **écriture** (secondes)			Temps moyens en **lecture** (secondes)		
Taille / Déploiement	256 Ko	1 Mo	10 Mo	256 Ko	1 Mo	10 Mo
Déployé seul	0,33	1,07	3,92	0,29	0,50	1,98
Couplé avec RozoFS	0,33	1,08	3,97	0,19	0,36	1,83
Déployé dans le Cloud	2,29	5,55	27,58	1,57	2,62	11,24

TABLE 2 – Temps d'accès moyens pour écrire et lire un objet lorsque la charge est de 100 objets stockés sur chaque site (topologie de 3 sites). Les écarts types sont d'environ 0,1 s.

Évaluation expérimentale - Temps d'accès

Nous faisons varier la taille des objets écrits entre 256 Ko et 10 Mo.
Chacun des 3 clients écrit 100 objets sur son site (300 objets manipulés au total).

	Temps moyens en **écriture** (secondes)			Temps moyens en **lecture** (secondes)		
Taille / Déploiement	256 Ko	1 Mo	10 Mo	256 Ko	1 Mo	10 Mo
Déployé seul	0,33	1,07	3,92	0,29	0,50	1,98
Couplé avec RozoFS	0,33	1,08	3,97	0,19	0,36	1,83
Déployé dans le Cloud	2,29	5,55	27,58	1,57	2,62	11,24

TABLE 2 – Temps d'accès moyens pour écrire et lire un objet lorsque la charge est de 100 objets stockés sur chaque site (topologie de 3 sites). Les écarts types sont d'environ 0,1 s.

Évaluation expérimentale - Temps d'accès

Nous faisons varier la taille des objets écrits entre 256 Ko et 10 Mo.
Chacun des 3 clients écrit 100 objets sur son site (300 objets manipulés au total).

	Temps moyens en **écriture** (secondes)			Temps moyens en **lecture** (secondes)		
Taille / Déploiement	256 Ko	1 Mo	10 Mo	256 Ko	1 Mo	10 Mo
Déployé seul	0,33	1,07	3,92	0,29	0,50	1,98
Couplé avec RozoFS	0,33	1,08	3,97	0,19	0,36	1,83
Déployé dans le Cloud	2,29	5,55	27,58	1,57	2,62	11,24

TABLE 2 – Temps d'accès moyens pour écrire et lire un objet lorsque la charge est de 100 objets stockés sur chaque site (topologie de 3 sites). Les écarts types sont d'environ 0,1 s.

Évaluation expérimentale - Temps d'accès

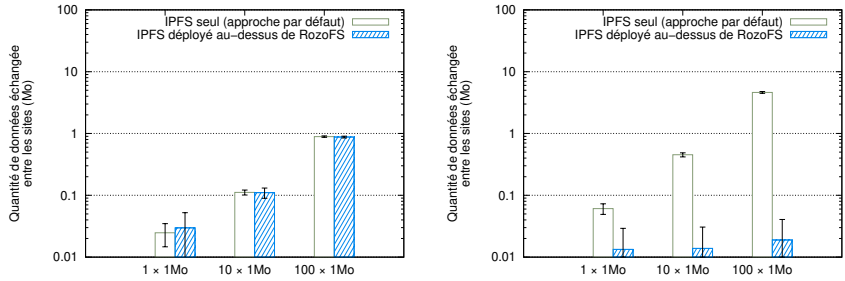
Nous faisons varier la taille des objets écrits entre 256 Ko et 10 Mo.
Chacun des 3 clients écrit 100 objets sur son site (300 objets manipulés au total).

	Temps moyens en **écriture** (secondes)			Temps moyens en **lecture** (secondes)		
Taille / Déploiement	256 Ko	1 Mo	10 Mo	256 Ko	1 Mo	10 Mo
Déployé seul	0,33	1,07	3,92	0,29	0,50	1,98
Couplé avec RozoFS	0,33	1,08	3,97	0,19	0,36	1,83
Déployé dans le Cloud	2,29	5,55	27,58	1,57	2,62	11,24

TABLE 3 – Temps d'accès moyens pour écrire et lire un objet lorsque la charge est de 100 objets stockés sur chaque site (topologie de 3 sites). **Les écarts types sont d'environ 0,1 s**.

Évaluation expérimentale - Quantité de trafic inter-sites

Nous faisons varier le nombre d'objets entre 1 et 100.



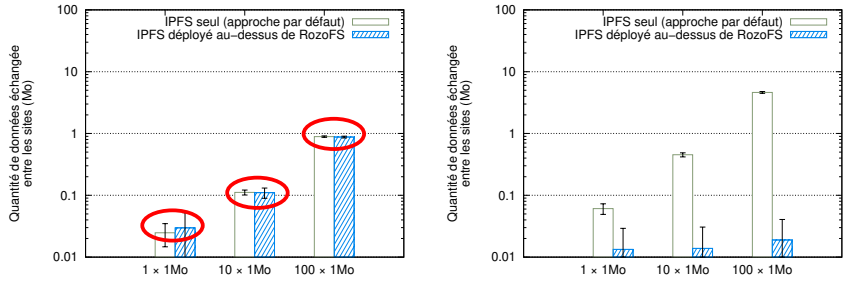
(a) – Écriture

(b) – Lecture

FIGURE 14 – Quantité de données moyenne échangée entre les sites pendant l'écriture et la lecture.

Évaluation expérimentale - Quantité de trafic inter-sites

Nous faisons varier le nombre d'objets entre 1 et 100.



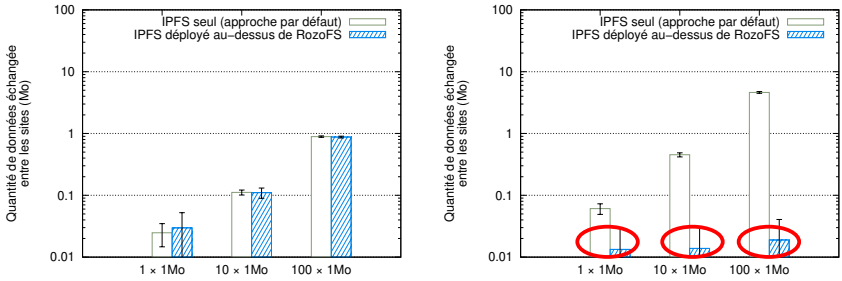
(a) – Écriture

(b) – Lecture

FIGURE 14 – Quantité de données moyenne échangée entre les sites pendant l'écriture et la lecture.

Évaluation expérimentale - Quantité de trafic inter-sites

Nous faisons varier le nombre d'objets entre 1 et 100.



(a) – Écriture

(b) – Lecture

FIGURE 14 – Quantité de données moyenne échangée entre les sites pendant l'écriture et la lecture.

Conclusion

Cette évaluation a montré que :

- L'ajout d'un système de fichiers distribué **ne dégrade pas les performances en écriture**, malgré l'utilisation d'un code à effacement et le surcoût d'espace de stockage consommé ;
- Le couplage permet de réduire de façon plus importante les temps de lecture des **petits objets** (réduction de 34%) ;
- La réduction des trafics réseaux inter-sites permet de **confiner le trafic**.

Ce travail a été présenté dans la conférence IEEE ICFEC 2017 [13].

---

13. Bastien Confais, Adrien Lebre et Benoît Parrein (mai 2017b). « An Object Store Service for a Fog/Edge Computing Infrastructure based on IPFS and Scale-out NAS ». Dans : *1st IEEE International Conference on Fog and Edge Computing - ICFEC'2017*. Madrid, Spain, Mai 2017

Confinement des trafics réseau inter-sites pour les accès distants

Description du problème

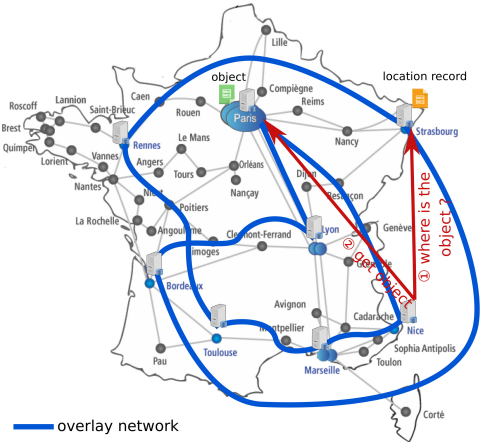


FIGURE 15 – Exemple de réseau de recouvrement d'une table de hachage distribuée.

L'utilisation de la table de hachage distribuée pose deux difficultés :

1. Le réseau de recouvrement ne suit pas le réseau physique sous-jacent ;
2. Il peut être nécessaire de contacter un **site lointain** pour localiser une donnée stockée sur un **site « voisin »**.

Description du problème

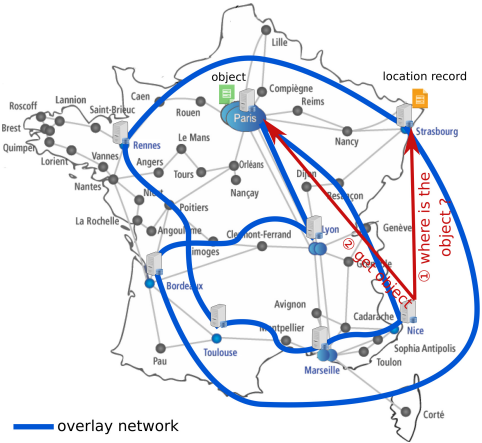


FIGURE 15 – Exemple de réseau de recouvrement d'une table de hachage distribuée.

L'utilisation de la table de hachage distribuée pose deux difficultés :

1. Le réseau de recouvrement ne suit pas le réseau physique sous-jacent ;
2. Il peut être nécessaire de contacter un **site lointain** pour localiser une donnée stockée sur un **site « voisin »**.

Description du problème

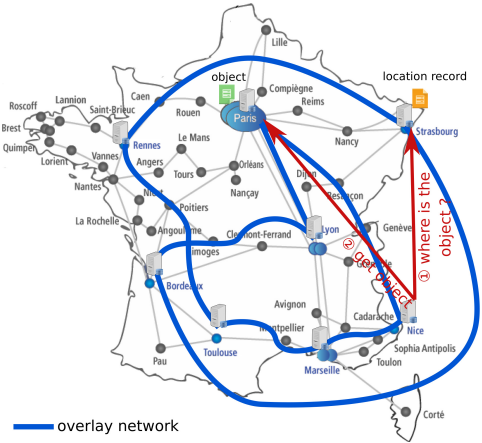


FIGURE 15 – Exemple de réseau de recouvrement d'une table de hachage distribuée.

L'utilisation de la table de hachage distribuée pose deux difficultés :

1. Le réseau de recouvrement ne suit pas le réseau physique sous-jacent ;
2. Il peut être nécessaire de contacter un **site lointain** pour localiser une donnée stockée sur un **site « voisin »**.

Différentes approches pour localiser des données

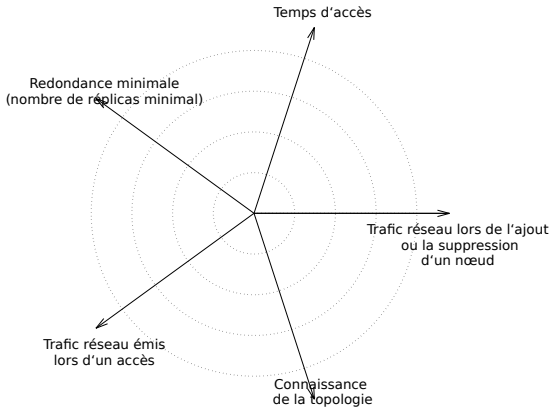


FIGURE 16 – Diagramme en étoile résumant les caractéristiques de différentes approches.

Différentes approches pour localiser des données

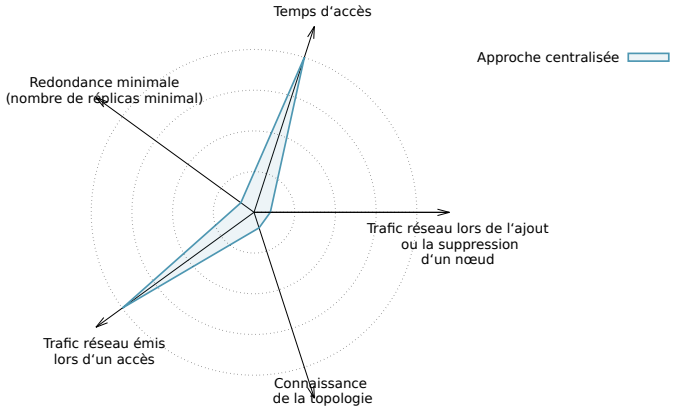


FIGURE 17 – Diagramme pour les **approches centralisées** (ex : RozoFS, Lustre, ...).

Différentes approches pour localiser des données

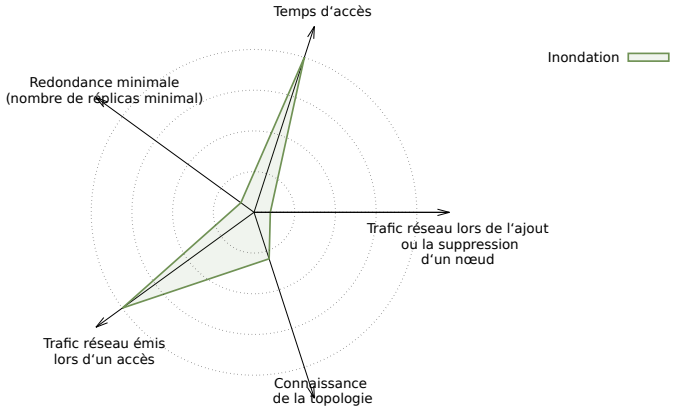


FIGURE 18 – Diagramme pour les **approches par inondation** (ex : Gnutella).

Différentes approches pour localiser des données

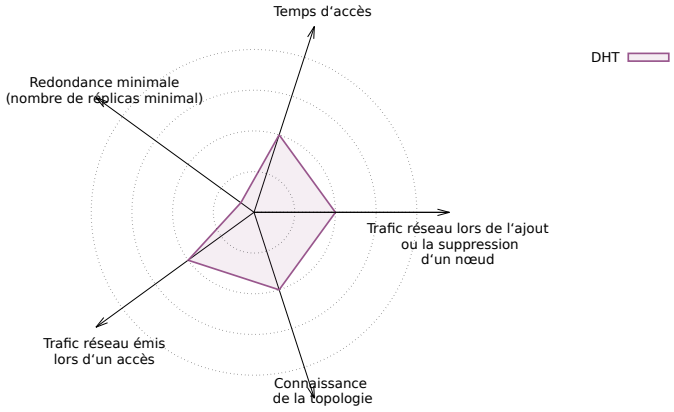


FIGURE 19 – Diagramme pour les **tables de hachage distribuées** (ex : IPFS).

Différentes approches pour localiser des données

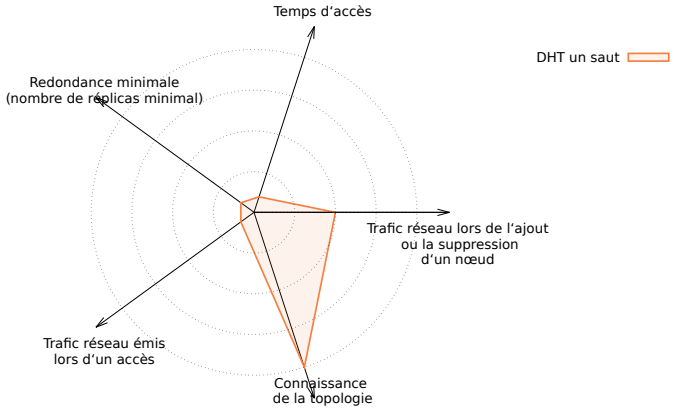


FIGURE 20 – Diagramme pour les **tables de hachage distribuées à un saut** (ex : Cassandra).

Différentes approches pour localiser des données

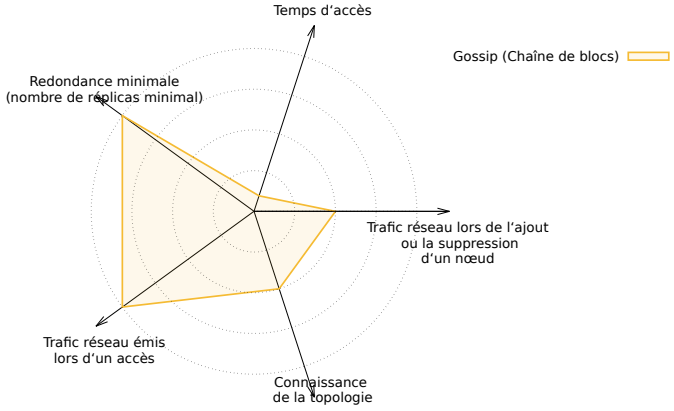


FIGURE 21 – Diagramme pour les **approches par gossip** (ou par chaînes de blocs) (ex : Dynamo).

Différentes approches pour localiser des données

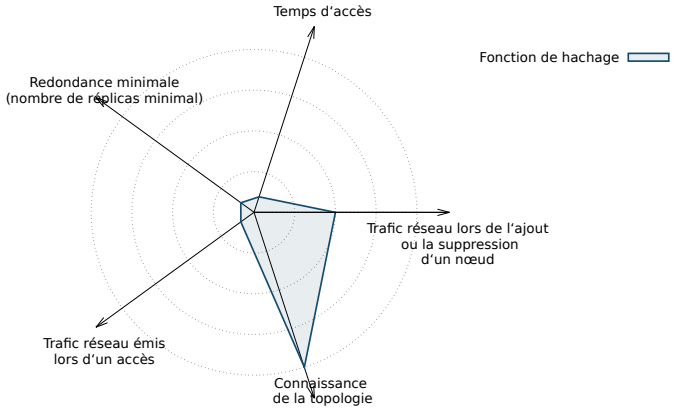


FIGURE 22 – Diagramme pour les **approches utilisant une fonction de hachage** (ex : Glusterfs, Rados).

Différentes approches pour localiser des données

Aucune des solutions présentées ne convient.

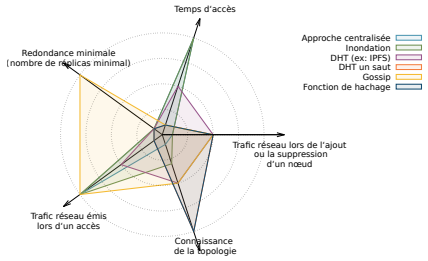


FIGURE 23 – Diagramme pour l'ensemble des approches présentées précédemment.

Nous proposons une solution ayant les caractéristiques suivantes :

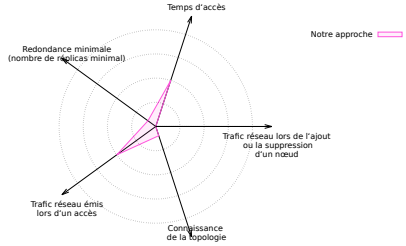


FIGURE 24 – Diagramme pour l'approche proposée.

Attention :l'ajout ou la suppresion d'un nœud est un défi qui reste à relever.

Le protocole DNS

Nous proposons un protocole similaire au protocole Domain Name System (DNS) [14].

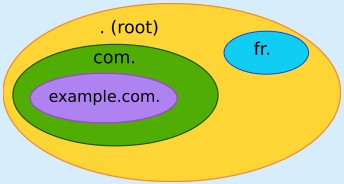


Figure 25 – Découpage de l'espace de noms du DNS.

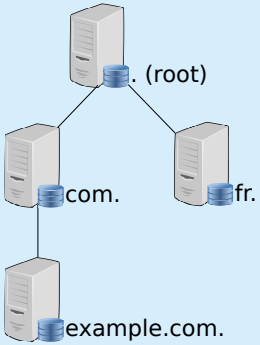


Figure 26 – Organisation du DNS en arbre.

14. P. Mockapetris (nov. 1987). *Domain Names - Concepts and Facilities*. RFC 1034. Network Working Group

Le protocole DNS

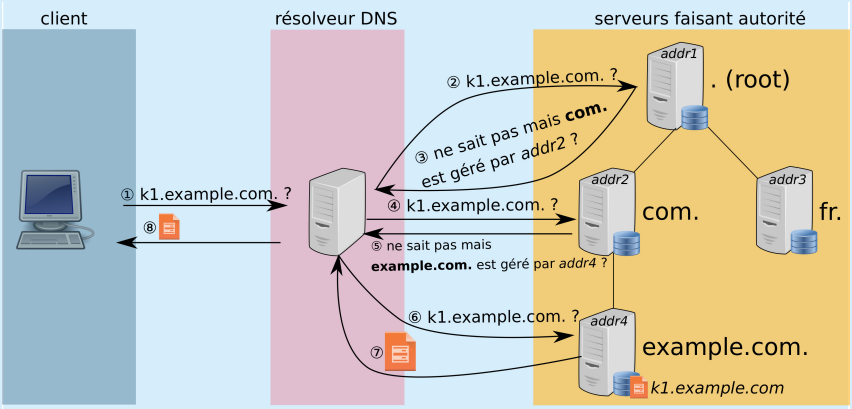


FIGURE 27 – Processus de résolution.

Solution proposée

Architecture :

- Nous construisons un arbre qui respecte la topologie physique ;
- Les requêtes sont transmises du site courant en interrogeant successivement les nœuds parents, jusqu'à trouver la localisation recherchée ou à atteindre la racine ;

Espace de noms :

- Les noms des objets sont suffixés par le nom du site sur lequel ils sont écrits en premier ;
- Perte de la transparence de nommage.

Solution proposée

Architecture :

- Nous construisons un arbre qui respecte la topologie physique ;
- Les requêtes sont transmises du site courant en interrogeant successivement les nœuds parents, jusqu'à trouver la localisation recherchée ou à atteindre la racine ;

Espace de noms :

- Les noms des objets sont suffixés par le nom du site sur lequel ils sont écrits en premier ;
- Perte de la transparence de nommage.

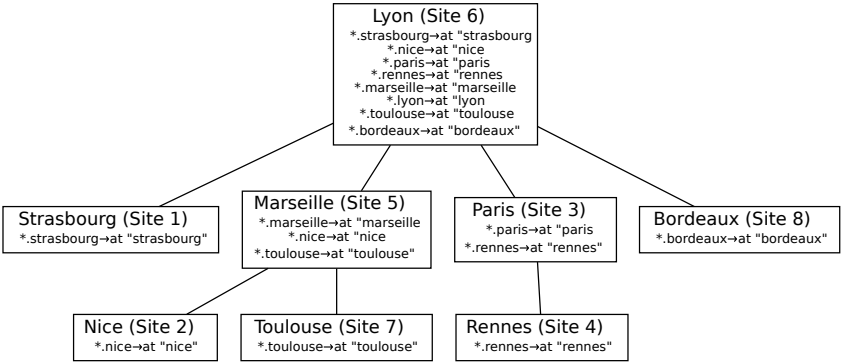


FIGURE 28 – Exemple d'arbre avec les enregistrements de localisation stockés au départ.

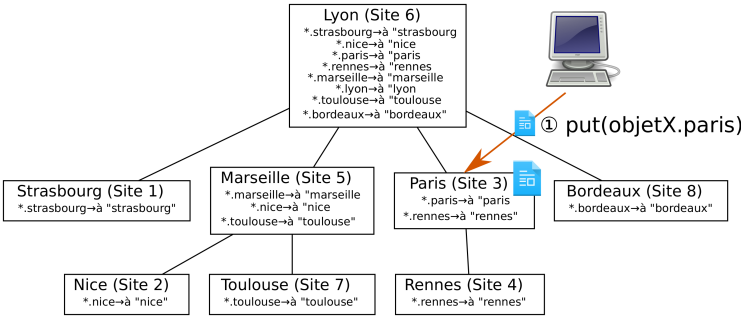


FIGURE 29 – Processus d'écriture, ne nécessitant pas de mettre à jour les enregistrements de localisation.

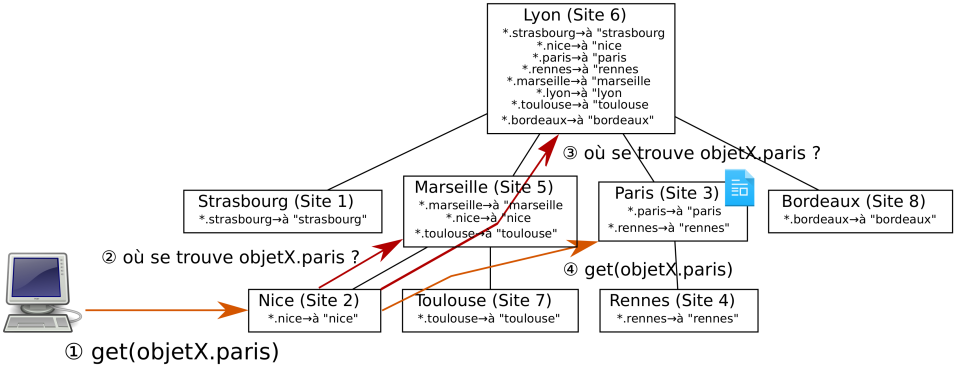


FIGURE 30 – Lecture de l'objet stocké à Paris depuis le site situé à Nice.
En rouge, les échangent liés au processus de localisation.

Solution proposée

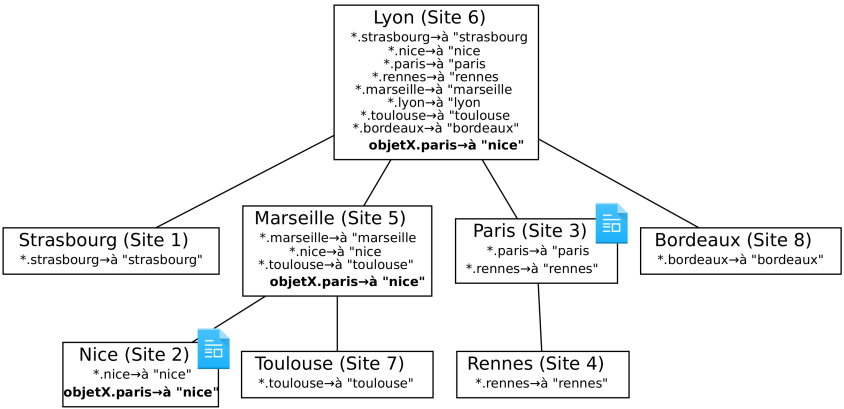


FIGURE 31 – État de l'arbre une fois qu'un nouveau réplica a été créé à Nice.

Solution proposée

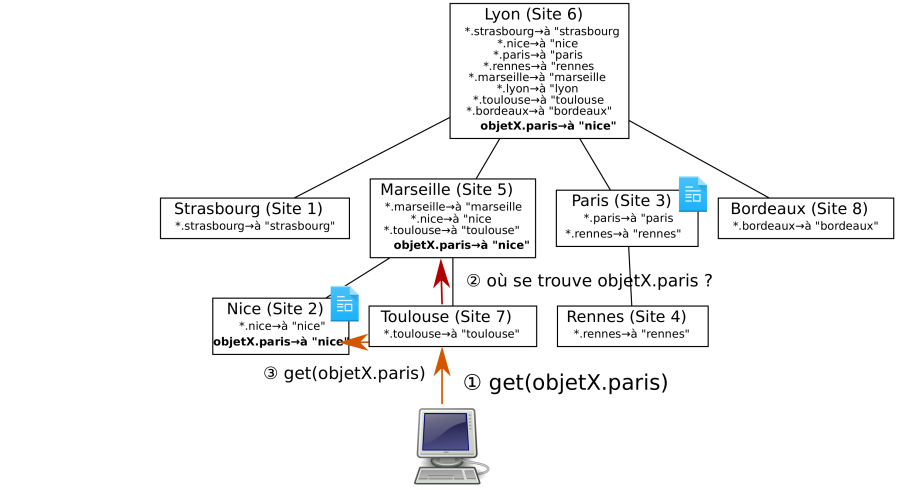


FIGURE 32 – Lecture de l'objet depuis Toulouse après que l'objet ait été accédé depuis Nice.

Construction de l'arbre

La construction de l'arbre est importante pour les performances du système

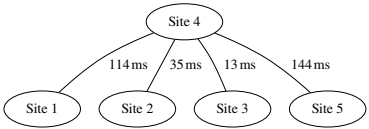


FIGURE 33 – Avec un arbre « plat », le nœud racine devient fortement sollicité.

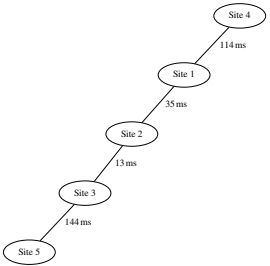


FIGURE 34 – Avec un arbre « profond » nécessite, dans le pire des cas, de faire de nombreux sauts.

Construction de l'arbre

Nous proposons de réutiliser l'algorithme de Dijkstra [15] appliqué à la topologie physique pour générer notre arbre.
Nous considérons la distance $d(x, y)$, comme étant la latence réseau du lien entre les nœuds x et y.

- Nous exécutons l'algorithme avec chaque nœud source et choisissons l'arbre ayant le plus faible poids ;
- Nous adaptons la fonction de coût à notre protocole itératif :

$$f_c = \left(\sum_{i=racine}^{parent(\text{nœud})} d\left(i, parent\,(\text{i})\right) \times \text{profondeur}\,(i) \right) + d\left(parent\,(\text{nœud}), \text{nœud}\right) \times \text{profondeur}\,(\text{nœud}) \qquad (1)$$

15. E. W. Dijkstra (déc. 1959). « A Note on Two Problems in Connexion with Graphs ». Dans : *Numer. Math.* 1.1, p. 269-271

Exemple sur une sous-partie du réseau RENATER

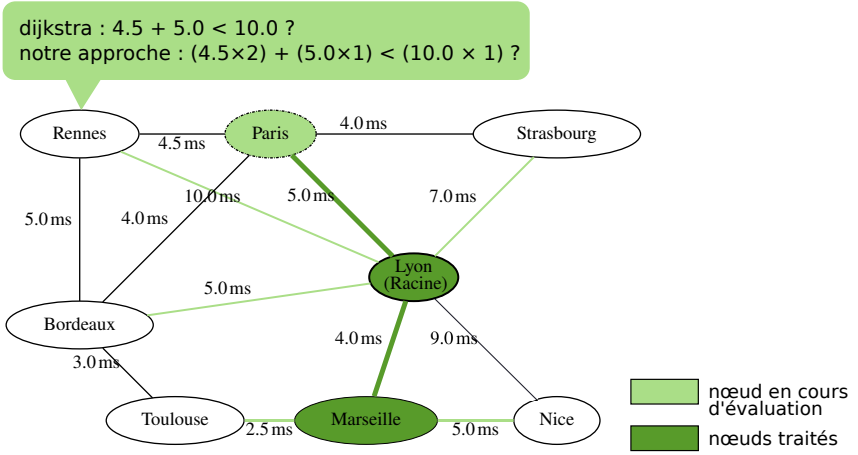


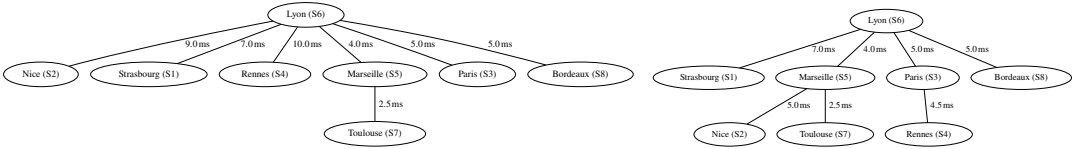
FIGURE 35 – Comparaison entre la fonction de coût de Dijkstra et la métrique proposée. Lyon est le nœud source (la racine de l'arbre) et Paris est le nœud actuellement sélectionné.

Construction de l'arbre

Problème : un arbre trop « plat » ne permet pas de bénéficier des nouveaux enregistrements de localisation puisque la racine est immédiatement atteinte.

Solution proposée : utiliser certains liens qui augmentent légèrement la latence s'ils augmentent la profondeur d'un nœud.

L'arbre ainsi construit semble mieux adapté à notre protocole.



(a) – Arbre de poids minimal.

(b) – Arbre avec un poids augmenté de 20%

FIGURE 36 – Arbres construits en utilisant notre métrique seulement (a) et en autorisant une dégradation des latences (b).

Avantages / Inconvénients

Avantages :

- Pas d'enregistrements de localisation à stocker pour les objets qui ne sont jamais accédés ;
- Les sites proches sont contactés en premier ;
- Meilleure disponibilité en cas de partitionnement du réseau ;
- Plus un objet est accédé, plus la localisation pourra être trouvée avec un nombre réduit de sauts.

Inconvénient :

- Complexité pour construire l'arbre.

Évaluation expérimentale

Nous évaluons notre approche en émulant sur Grid'5000 l'arbre obtenu précédemment. Nous comparons notre approche à la table de hachage distribuée utilisée par IPFS.

- Les objets sont écrits sur le Site 1 ;
- Ils sont ensuite lus sucessivement depuis les autres sites. Nous nous assurons qu'un objet n'est pas lu plusieurs fois depuis le même site ;
- Nous mesurons le temps mis pour **localiser** chaque objet et non le temps d'accès total.

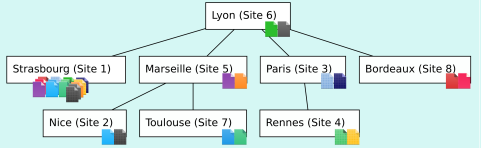


FIGURE 37 – Exemple d'affectation pour la première lecture.

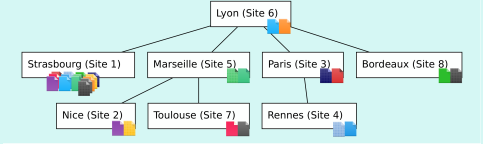
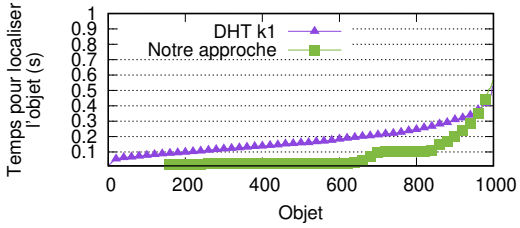
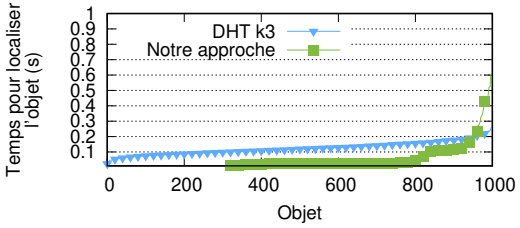


FIGURE 38 – Exemple d'affectation pour la seconde lecture.

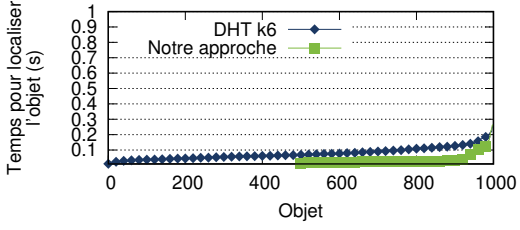
Évaluation expérimentale - Temps de localisation



(a) – Première lecture



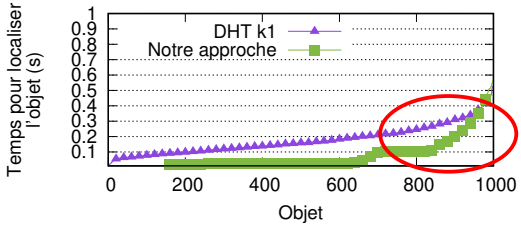
(b) – Troisième lecture



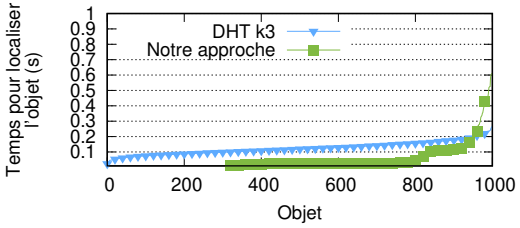
(c) – Sixième lecture

FIGURE 39 – Temps moyens pour localiser les objets lors de la première (a), troisième (b) et sixième (c) lecture.

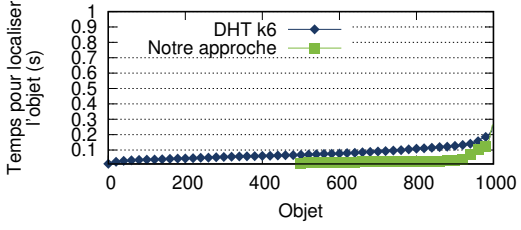
Évaluation expérimentale - Temps de localisation



(a) – Première lecture



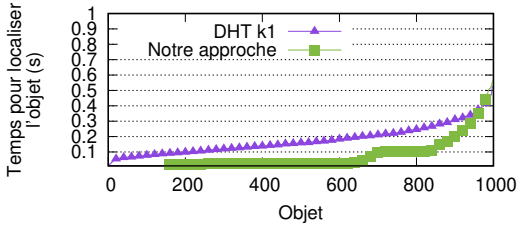
(b) – Troisième lecture



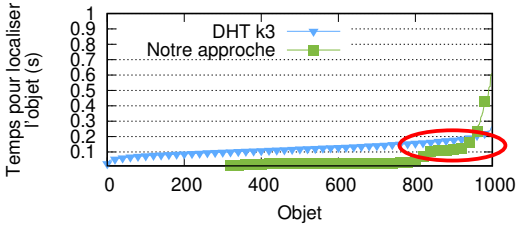
(c) – Sixième lecture

FIGURE 39 – Temps moyens pour localiser les objets lors de la première (a), troisième (b) et sixième (c) lecture.

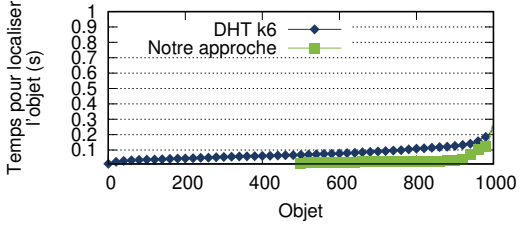
Évaluation expérimentale - Temps de localisation



(a) – Première lecture



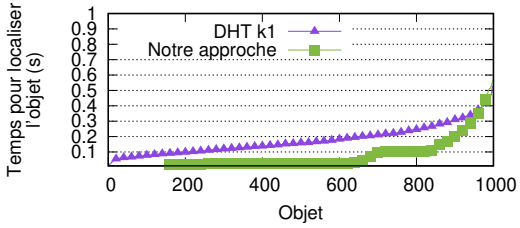
(b) – Troisième lecture



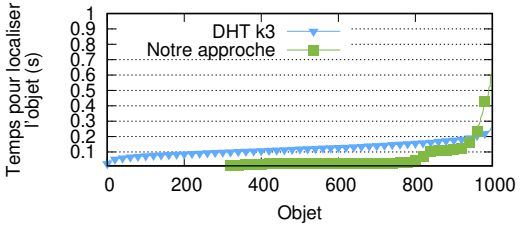
(c) – Sixième lecture

FIGURE 39 – Temps moyens pour localiser les objets lors de la première (a), troisième (b) et sixième (c) lecture.

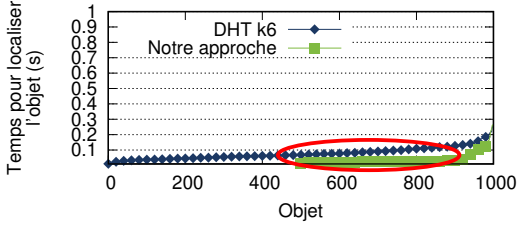
Évaluation expérimentale - Temps de localisation



(a) – Première lecture



(b) – Troisième lecture



(c) – Sixième lecture

FIGURE 39 – Temps moyens pour localiser les objets lors de la première (a), troisième (b) et sixième (c) lecture.

Conclusion

Notre approche permet :

- De réduire les temps de localisation d'environ 20% par rapport à une table de hachage distribuée ;
- De confiner le trafic réseau ;
- D'accéder au réplica le plus proche.

Ce travail a été présenté à RESCOM 2018 [16] et est en soumission pour la conférence IEEE GlobeCom 2018 [17].

16. Bastien Confais, Adrien Lebre et Benoît Parrein (jan. 2018b). « Adaptation of the Dijkstra's algorithm for metadata management in Fog Computing ». Dans : *Journées non thématiques, RESCOM 2018*. Toulouse, France

17. Bastien Confais, Adrien Lebre et Benoît Parrein (2018a). « A Tree-Based Approach to locate Object Replicas in a Fog Storage Infrastructure (en soumission) ». Dans : *IEEE Global Communications Conference (GlobeCom 2018)*

Utilisation simultanée des plate-formes d'expérimentation Grid'5000 et FIT/IoT-Lab

FIT/IoT-Lab

Nous souhaitons vérifier comment notre solution se comporte dans un environnement réaliste, dans un cas d'utilisation de collecte de données issues de capteurs.

Nous proposons d'utiliser simultanément :

- La plateforme Grid'5000 pour l'émulation du site de Fog ;
- La plateforme FIT/IoT-Lab qui met à disposition des objets connectés (capteurs).

Description de la plateforme FIT/IoT-Lab

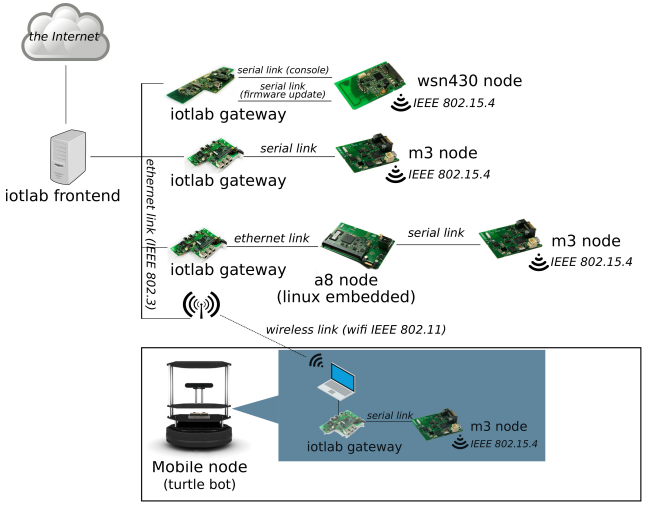


FIGURE 40 – Architecture générale d'un site de la plateforme FIT/IoT-Lab.

Scénario

Nous voulons écrire et lire des objets depuis un nœud m3 connecté par un lien radio.

Nous implémentons un client IPFS pour le système d'exploitation RIOT (malgré un support de TCP limité).

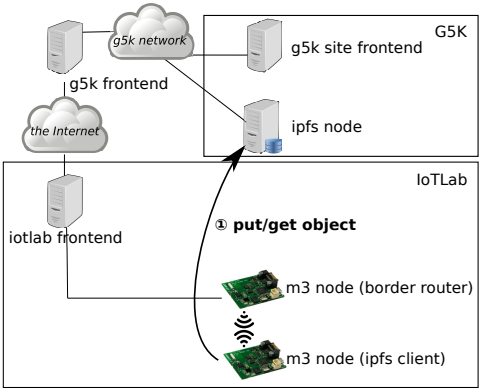


FIGURE 41 – Description du scénario.

Difficultés d'interconnexion des plateformes Grid'5000 et FIT/IoT-Lab

L'interconnexion des deux plateformes n'est pas simple :

- Pas de réseau globalement routable de bout en bout :
  - adresses IPv4 partagées entre plusieurs nœuds (NAT) ;
  - pas d'IPv6 sur la plateforme Grid'5000.
- Non localité du routage : le chemin réseau entre un site G5K et un site IoT-Lab situés dans la même ville n'est pas direct.

Tunnels entre les plateformes Grid'5000 et FIT/IoT-Lab

Nous proposons d'utiliser des tunnels SSH.

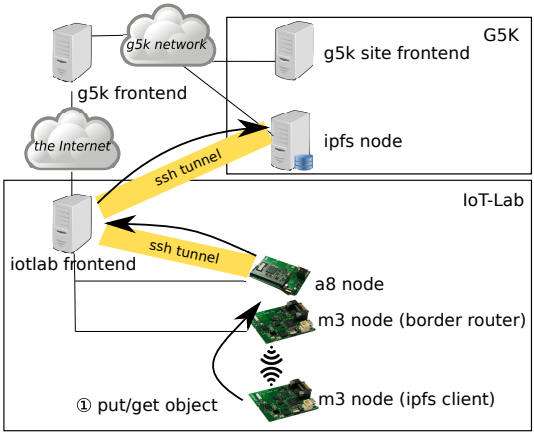


FIGURE 42 – Tunnels établis entre les plateformes FIT/Iot-lab et Grid'5000.

Conclusion

Nous retiendrons que le déploiement d'une infrastructure de Fog Computing dans un environnement réel n'est pas simple.

Les interconnexions entre les opérateurs ne garantissent pas la localité des trafics réseau.

Conclusion générale

Notre objectif était de développer une solution de stockage adaptée aux infrastructures de Fog Computing.

Ce que nous avons fait :

- Proposition d'un couplage entre IPFS et un système de fichiers distribué.
  Objectif : limiter les trafics inter-sites lors d'accès à des objets locaux.
- Proposition d'un système de gestion des localisations.
  Objectif : limiter les trafics inter-sites lors d'accès à des objets stockés sur un site distant.
- Utilisation simultanée des plateformes Grid'5000 et FIT/IoT-Lab.
  Objectif : réaliser une expérimentation dans un environnement de Fog Computing plus réaliste.

Conclusion générale

Quelques cas d'utilisation pour notre solution :

- Partage de vidéos dans un stade ;
- Partage de données dans un contexte maison/travail ;
- Collecte de données issues de capteurs.

Perspectives

Amélioration de l'implémentation :

- Définir un protocole plus efficace (récursif avec un réseau de recouvrement décentralisé). *Alexandre Van Kempen, post-doctorant, équipe STACK, LS2N* a été recruté pour approfondir ces aspects.

Améliorations nécessitant un apport théorique :

- Intégrer le calcul de l'arbre dans le protocole ;
- Supporter la dynaminicité du réseau ;
- Gérer les sites de Fog avec des capacités hétérogènes ;
- Déplacer les données de façon préventive ;
- Répartir les données entre les infrastructures de Cloud Computing et celles du Fog.

Production scientifique I

Journaux

Confais, Bastien, Adrien Lebre et Benoît Parrein (août 2017c). « Performance Analysis of Object Store Systems in a Fog and Edge Computing Infrastructure ». Dans : *Transactions on Large-Scale Data- and Knowledge-Centered Systems XXXIII*. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 40-79.

Conférences internationales avec comité de lecture

Confais, Bastien, Adrien Lebre et Benoît Parrein (déc. 2016a). « Performance Analysis of Object Store Systems in a Fog/Edge Computing Infrastructures ». Dans : *CloudCom*. Luxembourg, Luxembourg, Décembre 2016.

– (mai 2017b). « An Object Store Service for a Fog/Edge Computing Infrastructure based on IPFS and Scale-out NAS ». Dans : *1st IEEE International Conference on Fog and Edge Computing - ICFEC'2017*. Madrid, Spain, Mai 2017.

– (2018a). « A Tree-Based Approach to locate Object Replicas in a Fog Storage Infrastructure (en soumission) ». Dans : *IEEE Global Communications Conference (GlobeCom 2018)*.

Production scientifique II

Conférences nationales avec comité de lecture

- Confais, Bastien, Adrien Lebre et Benoît Parrein (juil. 2016c). « Quel système de stockage pour les architectures Fog ? » Dans : *Compas'2016*. Lorient, France.
- (juin 2017a). « An object store for Fog infrastructures based on IPFS and a Scale-Out NAS ». Dans : *RESCOM 2017*. RESCOM 2017 École d'été, journées thématiques (virtualisation dans les réseaux et le Cloud). Le Croisic, France, 2, (session poster).
- (avr. 2018c). « Improving locality of an object store working in a Fog environment ». Dans : *1st Grid'5000-FIT school*. Nice, France.

Autres publications

- Confais, Bastien (avr. 2017). « Un système de stockage par objets pour les architectures Fog s'appuyant sur IPFS et un système de fichiers distribué de type Scale-Out NAS ». Dans : *Journée des doctorants 2017, ED STIM*. Nantes, France.
- Confais, Bastien, Adrien Lebre et Benoît Parrein (juin 2016b). « Quel système de stockage distribué pour le Fog ? » Dans : *Journées Scientifiques de l'Université de Nantes*. Nantes, France.

Production scientifique III

Confais, Bastien, Adrien Lebre et Benoît Parrein (nov. 2016d). « Which storage system for FOG Computing ? (conférence invitée) ». Dans : *WOS6 : 6th INRIA/Technicolor workshop on Big Data and Analytics*. Rennes, France.

– (jan. 2018b). « Adaptation of the Dijkstra's algorithm for metadata management in Fog Computing ». Dans : *Journées non thématiques, RESCOM 2018*. Toulouse, France.

Production logicielle I

Adaptations de logiciels existants (1.8 K lignes de code)

- Module pour *Yahoo Cloud System Benchmark* (YCSB) afin d'évaluer les performances d'*InterPlantery FileSystem* (IPFS), 160 lignes en Java, `https://github.com/bconfais/YCSB/blob/master/ipfs/src/main/java/com/yahoo/ycsb/db/IPFSClient.java`
- Modification d'IPFS pour stocker les objets au sein d'un système de fichiers distribué, 250 lignes en Go, `https://github.com/bconfais/go-ipfs/tree/common_backend_v1a`
- Implémentation de notre protocole de localisation des objets au sein d'IPFS, 500 lignes en Go, `https://github.com/bconfais/go-ipfs/tree/dns`
- Implémentation d'un client IPFS pour le système d'exploitation RIOT, 900 lignes en C, `https://github.com/bconfais/RIOT/tree/master/examples/ipfs_client`, soumis pour être intégré à la version officielle du système d'exploitation : `https://github.com/RIOT-OS/RIOT/pull/8889`

Production logicielle II

Scripts de déploiement sur Grid'5000 (3.5 K lignes de code)

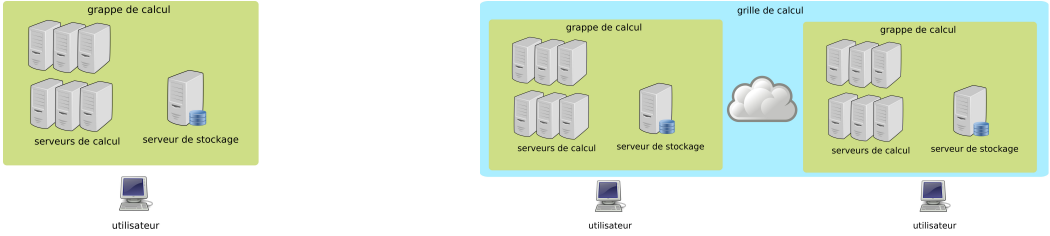
- Script permettant de déployer Rados, Cassandra et IPFS, 1000 lignes en Python
  `https://github.com/bconfais/benchmark/tree/master/Cloudcom2016`
- Script permettant de déployer IPFS et RozoFS et permettant également d'évaluer notre couplage, 800 lignes en Python, `https://github.com/bconfais/benchmark/tree/master/FEC2017`
- Script pour déployer IPFS ainsi qu'un service DNS, afin d'évaluer notre protocole de localisation des objets, 900 lignes en Python, `https://github.com/bconfais/benchmark/blob/master/dns`
- Script pour établir un tunnel entre les plateformes Grid'5000 et FIT/IoT-Lab, déployer IPFS sur Grid'5000 et demander à un nœud sur la plateforme FIT d'écrire un objet, 600 lignes en Python, `https://github.com/bconfais/benchmark/tree/master/iot-lab`

Merci pour votre attention

Backup

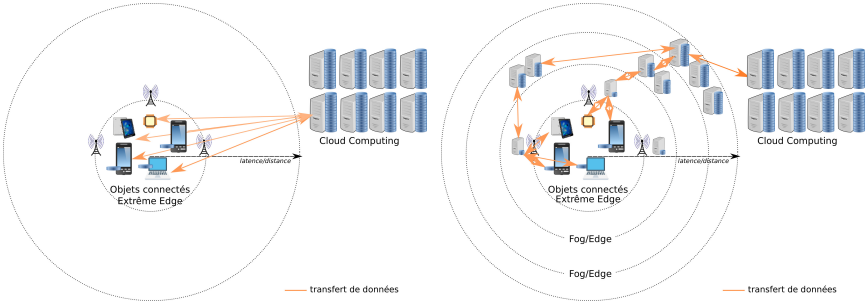
Plusieurs architectures historiques :

- Les grappes de calcul ;
- Les grilles de calcul ;
- Les infrastructure d'informatique en nuage.



(a) – Grappe de calcul (b) – Fédération de grappes dans une grille de calcul

FIGURE 43 – Évolution des infrastructures pour le calcul réparti.



(a) – Approche Cloud Computing (b) – Approche Fog Computing

FIGURE 44 – Exemple d'infrastructure de Cloud Computing (a) et de Fog Computing (b) montrant les interactions et les capacités de stockage de chaque équipement.

Quelques solutions de stockage

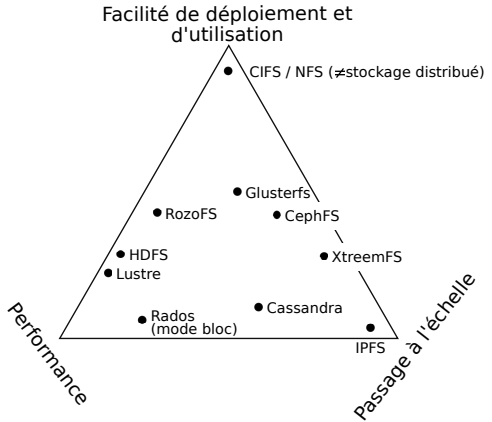


FIGURE 45 – Comparaison de différentes solutions de stockage distribués.

Rados - La fonction de placement CRUSH

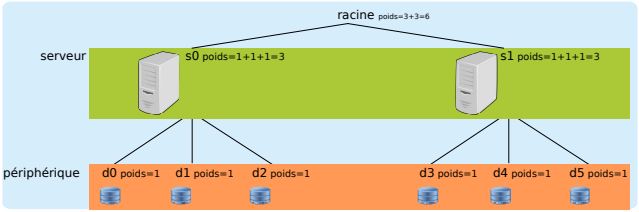


FIGURE 46 – Exemple de *clustermap*.

1. sélectionner(racine)
2. choisir(2, serveur)
3. choisir(1, périphérique)
4. fin

FIGURE 47 – Exemple de règle pour le placement.

$\vec{i} = ()$

Rados - La fonction de placement CRUSH

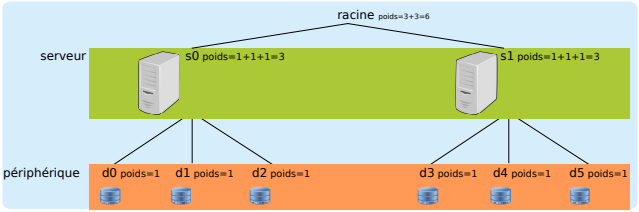


FIGURE 46 – Exemple de *clustermap*.

1. **sélectionner(racine)**
2. choisir(2, serveur)
3. choisir(1, périphérique)
4. fin

FIGURE 47 – Exemple de règle pour le placement.

$\vec{i}$ = (racine)

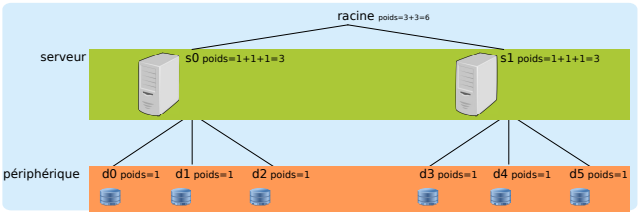


FIGURE 46 – Exemple de *clustermap*.

1. sélectionner(racine)
2. **choisir(2, serveur)**
3. choisir(1, périphérique)
4. fin

FIGURE 47 – Exemple de règle pour le placement.

$$c(1, \text{nom objet}) = hash(\text{nom objet}) + p \ mod \ 2 = 1$$
$$c(2, \text{nom objet}) = hash(\text{nom objet}) + 2p \ mod \ 2 = 0$$
$$\vec{i} = (\text{s0 s1})$$

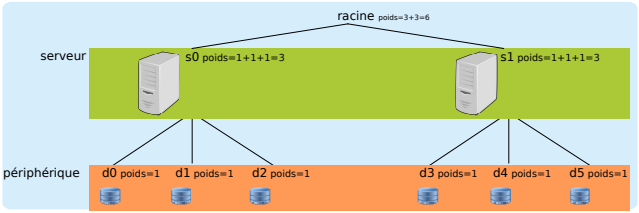


FIGURE 46 – Exemple de *clustermap*.

1. sélectionner(racine)
2. choisir(2, serveur)
3. **choisir(1, périphérique)**
4. fin

FIGURE 47 – Exemple de règle pour le placement.

pour s0 : $c(1, \text{nom objet}) = hash(\text{nom objet}) + p \ mod \ 3 = 1$

pour s1 : $c(1, \text{nom objet}) = hash(\text{nom objet}) + p \ mod \ 3 = 1$

$\vec{i}$ = (d1 d4)

Rados - La fonction de placement CRUSH

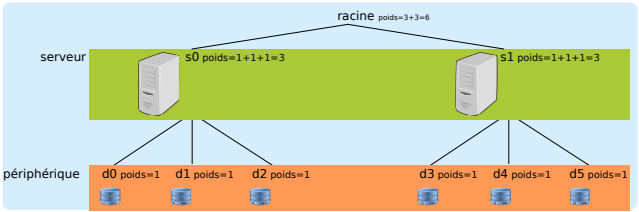


FIGURE 46 – Exemple de *clustermap*.

1. sélectionner(racine)
2. choisir(2, serveur)
3. choisir(1, périphérique)
4. **fin**

FIGURE 47 – Exemple de règle pour le placement.

$\vec{i}$ = (d1 d4)

CRUSH est un algorithme coûteux. Rados regroupe donc les objets.

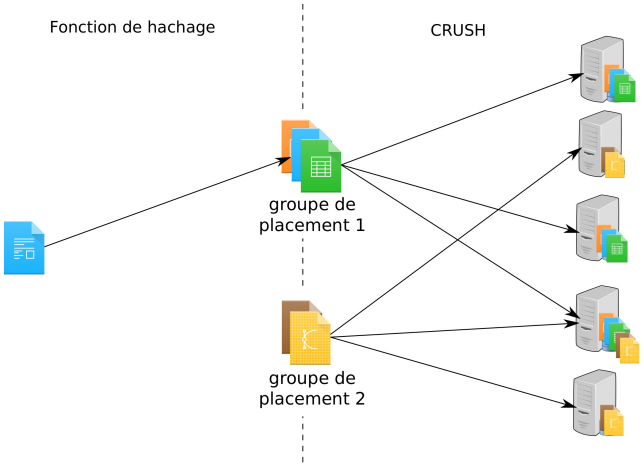


FIGURE 48 – Processus de placement utilisé par Rados.

Diagramme de séquence

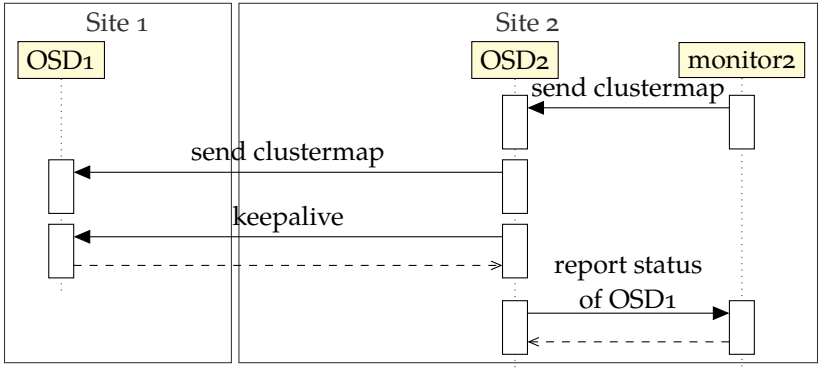


FIGURE 49 – Trafic réseau observé lorsque Rados est déployé dans un environnement de Fog Computing (point de vue d'un **serveur de stockage**).

Évaluation expérimentale des solutions de stockage Rados, Cassandra & IPFS

Conclusion

Rados : la quantité de trafic dépend du nombre de moniteurs mais aussi du nombre d'objets accédés (accès rapportés aux moniteurs) ;

Cassandra : la quantité de trafic est linéaire au nombre de sites ;

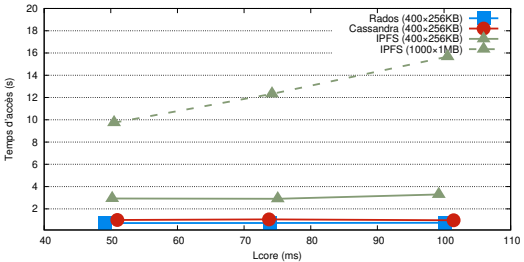
IPFS : la quantité de trafic est linéaire au nombre d'objets accédés.

Certains trafic sont asynchrones et n'impactent pas les temps d'accès :

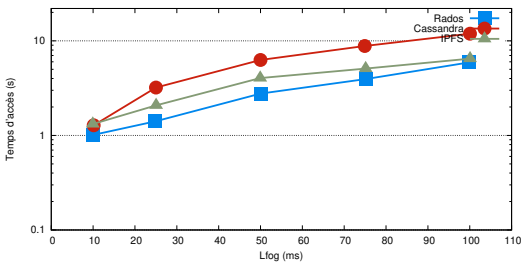
- Le trafic entre les moniteurs pour Rados ;
- Le `gossip` dans Cassandra ;
- La mise à jour de la table de hachage distribuée (après une écriture ou après qu'un nouveau réplica soit créé) dans IPFS.

Impact des latences L_{Core} et L_{Fog}

Nous proposons de faire varier L_{Core} pendant que la latence $LFog$ est fixée à 10 ms. De la même façon, nous faisons ensuite varier L_{Fog} lorsque que L_{Core} est fixé à 50 ms. Le test a été effectué avec 7 sites.



(a) – Temps de lecture en fonction de la latence L_{Core}.



(b) – Temps de lecture en fonction de la latence L_{Fog}.

FIGURE 50 – Temps de lecture en fonction de la latence inter-sites (a) et de la latence pour joindre le site de Fog (b).

Impact des latences L_{Core} et L_{Fog}

Observations sur la variation de L_{Core} :

- IPFS est grandement impacté L_{Core} ;
- Rados ne fonctionne plus après une latence inter-site supérieure à 100 ms (impossible de réaliser l'élection Paxos).

Observation sur la variation de L_{Fog} :

- Cassandra est très sensible à la variation de la latence L_{Fog}... ... mais nous ne comprenons pas pourquoi.

Impact de la latence inter-sites lors d'accès distants

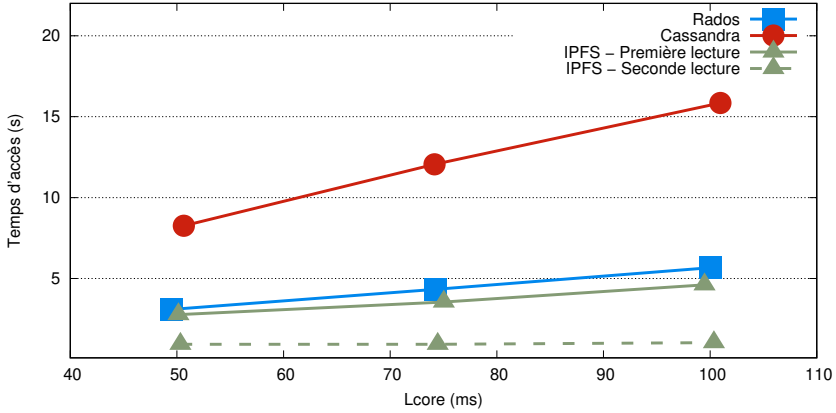


FIGURE 51 – Temps d'accès en fonction de la latence inter-sites L_{Core}. Une charge de 400×256 Ko est utilisée avec 7 sites.

IPFS est le moins sensible à la variation de latence.

Évaluation expérimentale - Rozofs en quelques mots

Rozofs [18] est un système de fichiers distribué avec les caractéristiques suivantes :

- Un serveur de métadonnées détermine le placement de chaque projection ;
- Un code à effacement Mojette permet la tolérance aux pannes : 2 projections sur 3 sont nécessaires pour reconstruire le bloc de données ;
- Un surcoût de 50% en espace de stockage.

18. Dimitri Pertin et al. (avr. 2014). « Distributed File System based on Erasure Coding for I/O Intensive Applications ». Dans : *4th International Conference on Cloud Computing and Service Science*. Barcelone, Spain

Approches pour localiser des données

Approche simple qui consiste à interroger le voisinage.

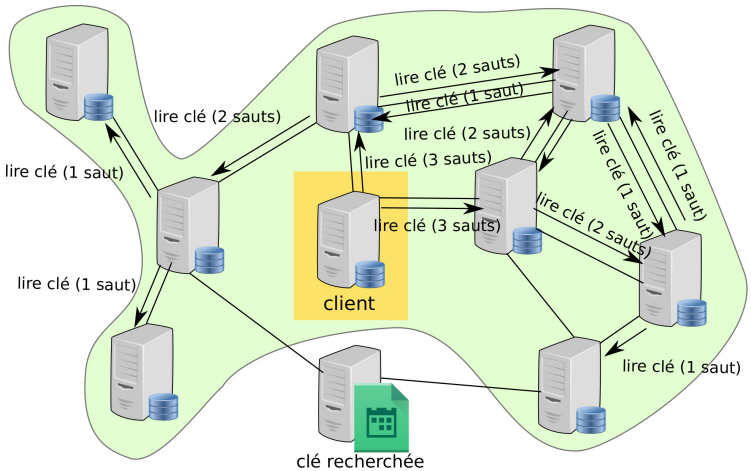


FIGURE 52 – Exemple de `flooding` pour localiser un objet.

Chaque nœud envoie à ses voisins la liste des objets qu'il stocke. Chaque nœud se construit un annuaire pour localiser les objets.

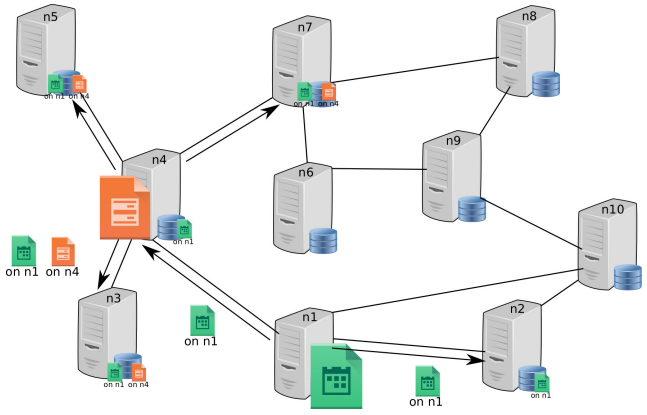


FIGURE 53 – Exemple d'une approche par gossip.

Chaîne de blocs

La localisation peut être stockée dans une chaîne de blocs. Le principal défaut de cette approche est que les chaînes de blocs sont des structures de données dans lequel seule l'opération d'ajout est acceptée.

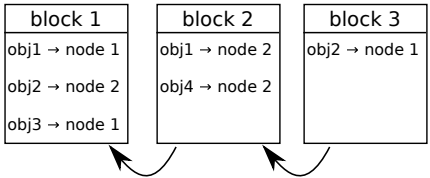


Figure 54 – Exemple d'une chaîne de blocs stockant la localisation des objets.

Table de hachage distribuée

Une DHT peut être utilisée si :

- Le réseau de recouvrement suit la topologie physique ;
- La stratégie de routage limite le nombre de sauts ;
- Être capable de choisir le nœud stockant un enregistrement donné.

Le routage de type `Plaxton` peut être utilisé pour fournir de la localité.

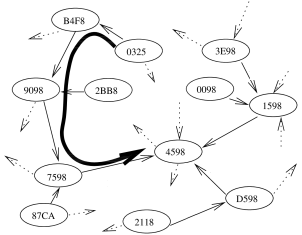
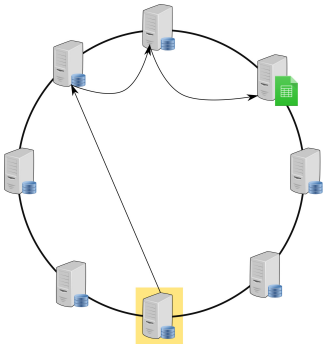
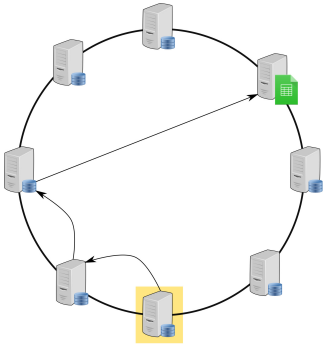


FIGURE 55 – Routage de type Plaxton dans lequel les requêtes sont routées selon leur préfixe.

Réplication le long du chemin dans une table de hachage distribuée



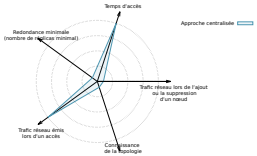
(a) – Plus long saut en premier.



(b) – Plus petit saut en premier.

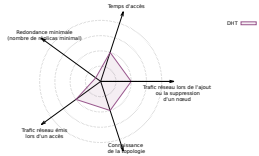
FIGURE 56 – Deux exemples de stratégies pour router les requêtes dans une table de hachage distribuée.

Synthèse des approches permettant de localiser des données



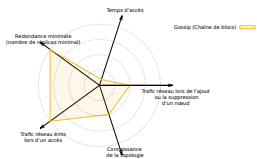
(a) – Approche centralisée.

(b) – Inondation.

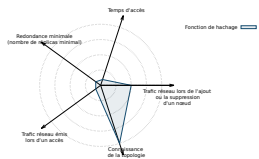


(c) – DHT.

(d) – DHT à un saut.



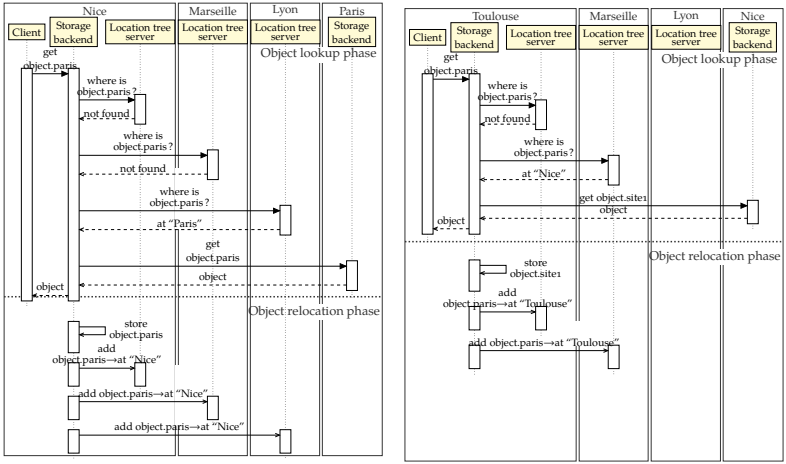
(e) – Gossip.



(f) – Fonction de hachage.

FIGURE 57 – Diagramme en étoile résumant les caractéristiques de plusieurs mécanismes permettant de localiser des données.

Solution proposée pour limiter les échanges inter-sites



(a) – Lecture depuis Nice d'un objet stocké à Paris.

(b) – Lecture depuis Toulouse d'un objet précédemment relocalisé à Nice.

FIGURE 58 – Diagrammes de séquence montrant le processus de lecture lorsque l'objet est

Perspectives

Ce que je ferais différement aujourd'hui :

- Meilleure caractérisation des cas d'utilisation pour les évaluations expérimentales ;
- Meilleure caractérisation de l'infrastructure (échelle différente selon le point de vue d'un opérateur ou d'un fournisseur de services) ;

Quelques pistes de recherche :

- Développer une solution de stockage hybride impliquant le Fog Computing et l'Extrême Edge Computing ;
- Prendre en compte des considérations énergétiques dans le stockage.

Le temps d'écriture d'un objet de 80 octets est d'environ **0,72** seconde
(à comparer à **0,13** seconde lorsque le client se trouve sur la plateforme Grid'5000).

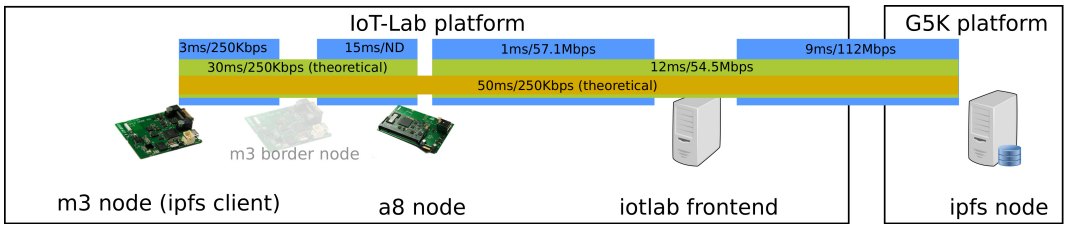


FIGURE 59 – Latences et débits TCP mesurés avec `iperf` lors de l'utilisation des tunnels entre la plateforme FIT/Iot-Lab et Grid'5000.