Accès aux Données dans le Fog Computing : le cas des dispositifs de proximité

Luiz Angelo Steffenel, Manuele Kirsch-Pinheiro

Compas'2019 : Parallélisme/ Architecture / Système
LIUPPA - IUT de Bayonne, France, du 24 au 28 juin 2019

Accès aux Données dans le Fog Computing : le cas des dispositifs de proximité

Luiz Angelo Steffenel[1] and Manuele Kirsch-Pinheiro[2]

[1] Centre de Recherche en STIC, Université de Reims Champagne-Ardenne, Reims - France
luiz-angelo.steffenel@univ-reims.fr

[2] Centre de Recherche en Informatique, Université Paris 1 Panthéon-Sorbonne, Paris - France
manuele.kirsch-pinheiro@univ-paris1.fr

Résumé
Le Fog Computing étend le paradigme du Cloud Computing à la périphérie du réseau, en s'appuyant sur des services intelligemment distribués pour répondre au mieux aux besoins des applications. Souvent, ces services sont déployés sur des dispositifs "passerelle" caractérisés par une puissance de calcul relativement limitée, ce qui peut porter atteinte aux objectifs de qualité de service à l'origine de la démarche fog. Pour mieux comprendre les enjeux, cet article s'intéresse aux performances de l'accès aux données distribuées à partir de dispositifs de proximité tels que les "systèmes sur puce". Des résultats expérimentaux comparant les performances d'accès aux données viennent démontrer l'intérêt de la prise en compte de ces performances lors du déploiement d'une architecture fog.

Mots-clés : Fog computing ; Systèmes sur puce ; Accès aux données ; Evaluation de Performance ; Table de Hachage Distribuée

---

1. Introduction

La multiplication des dispositifs informatiques qui nous entourent (smartphones, tablettes, nano-ordinateurs) et des dispositifs de l'Internet des Objets (IoT) soulève plusieurs défis en ce qui concerne la gestion de l'information. Le plus souvent, les informations produites dans ces environnements sont traitées selon le modèle client-serveur, dans lequel l'agrégation et l'analyse des données sont effectuées sur des installations distantes telles que le cloud [10, 9]. Cela s'explique principalement par la flexibilité et la puissance de calcul offertes par ces plates-formes mais, malgré ces avantages, le fait de ne compter que sur des infrastructures distantes présente également plusieurs inconvénients. Par exemple, le transfert de données vers le cloud peut induire des retards non négligeables et ralentir le traitement des données ou la prise de décision [12, 18, 5]. De plus, les applications qui dépendent exclusivement de services distants peuvent se bloquer si la connexion réseau est instable ou défaillante.
Plusieurs alternatives à la philosophie du "tout vers le cloud" ont été proposées, dont l'edge computing et le fog computing. On peut considérer que l'edge computing vise à traiter les données à la périphérie d'un réseau, avant qu'il soit transmis à un cloud ou à un datacenter distant [8]. De son côté, le terme fog computing est utilisé notamment pour exprimer l'idée de services entourant les utilisateurs et les sources de données, par opposition aux ressources

distantes des clouds [3]. Quoi qu'il soit, on parle ici de l'usage de ressources à proximité de l'utilisateur au lieu de ressources distantes. Comme les deux concepts (fog et edge) sont très similaires, nous allons utiliser le terme *fog* dans le reste de l'article.
Afin d'offrir un service de qualité dans le fog, il est important de s'assurer que les services seront placés au plus près des équipements finaux et que les ressources seront attribuées en fonction de leurs capacités au moment de l'exécution. Pour rendre cela possible, deux principes deviennent primordiaux, la localisation des données et la prise en compte du contexte d'exécution. Selon [13], le maintien des données en local présenterait non seulement l'avantage de la réduction du trafic d'accès, mais aussi de réduire la dépendance par rapport au cloud. Utiliser sciemment la localité de données est alors l'un des objectifs clés des approches fog [15]. La prise en compte de la localité des données ainsi que du contexte d'exécution [2] représente à son tour une propriété clé pour gérer de manière appropriée l'hétérogénéité et la dynamique qui caractérisent les nœuds dans le fog.
Afin de mieux apprendre les effets de cette hétérogénéité, il est nécessaire d'étudier l'impact de le la gestion des données dans des équipements de faible capacité, par exemple les systèmes sur puce (System-on-a-Chip, SoC). Comme les travaux qui étudient la performance du stockage dans les environnements fog sont encore trop rares, nous proposons dans cet article une première évaluation des performances d'un système de stockage basé sur le principe des DHT (Tables de Hachage Distribuées) sur un scénario fog incluant des dispositifs de type SoC.
Cet article s'organise comme suit : la Section 2 présente des travaux associés dans le domaine du fog et discute les défis associés ; dans la Section 3 nous discutons sur la gestion de la localisation des données et la performance des systèmes de stockage DHT. Dans la Section 4 nous introduisons les scénarios de test, ainsi que les résultats et les analyses préliminaires obtenus. Finalement, la Section 5 conclut le présent travail.

2. État de l'Art

La dissémination de périphériques de proximité dotés de capacités de calcul favorise la production de grandes quantités de données mais aussi l'intégration de ces périphériques dans le traitement des données, une approche opposée au modèle de cloud pur dans lequel tout le stockage et le traitement des données est effectué sur des serveurs distants. Plusieurs alternatives à la philosophie du "tout vers le cloud" ont été proposées ces dernières années, parmi lesquelles les grilles pervasives [16], le mobile edge computing [7, 8], les cloudlets [17], l'opportunistic edge computing [15] ou le fog computing [3]. De manière générale, on peut considérer que ces approches visent à traiter les données à la périphérie d'un réseau plutôt que de déléguer ce traitement à un cloud ou à un datacenter distant. Elles permettraient le prétraitement des données et la génération de connaissances au plus près des sources des données, en particulier des appareils mobiles. De plus, sans avoir à transférer inutilement des données vers un datacenter, les analyses à la périphérie du réseau peuvent simplifier et accélérer considérablement l'analyse tout en réduisant les coûts [13].
Malgré les différentes variations, toutes ces alternatives partagent le même principe de base : exploiter la puissance de calcul d'appareils situés à la périphérie du réseau afin d'accomplir les tâches précédemment déléguées à un site distant, tel qu'un cloud ou un datacenter. Ces dispositifs n'interagissent plus seulement avec le cloud, ils peuvent également se connecter à divers appareils IoT pour détecter et reconnaître les situations propices à des décisions intelligentes [13], formant ainsi un réseau complexe d'appareils hétérogènes et interconnectés. Tous ces éléments sont potentiellement des sources d'informations, produisant ou stockant des données provenant de l'utilisateur, des applications ou de l'environnement lui-même.

Cependant, de nouveaux défis découlent des alternatives fog. Les nœuds à la périphérie du réseau sont par définition hétérogènes et avec des capacités limitées en mémoire, performance des processeurs, espaces de stockage ou bande passante du réseau. En outre, les ressources sont dispersées sur des plateformes personnelles et véhiculaires hétérogènes et souvent connectées par intermittence [13], ce qui fait que les dispositifs à la périphérie du réseau ne sont pas aussi fiables que les serveurs cloud [11].
Parmi les différentes approches middleware pour l'implémentation d'applications pour les environnements de type fog, nous considérons que les middleware pair-à-pair (P2P) sont parmi les plus adaptés du fait qu'ils sont intrinsèquement distribués, tolérants aux pannes et faciles à déployer. De plus, ces middleware comportent souvent des solutions de stockage distribués tels que les Tables de Hachage Distribuées (DHT), qui offrent des propriétés intéressantes vis-à-vis de la réplication des données et de leur pérennité.
Cependant, les travaux qui étudient la performance des DHT dans les environnements fog sont encore trop rares, comme par exemple le travail de [6]. On regrette toutefois que dans ce travail les auteurs aient utilisé un environnement simulé où la seule caractéristique modulée est la latence des communications, alors que les ressources de haute gamme (serveurs Dell powerEdge, Intel Xeon, 16 cores, 128 GB RAM, réseau d'interconnexion Ethernet 10Gbps) ne semblent pas correspondre à un scénario fog réaliste.
À notre avis, il est nécessaire d'étudier l'impact de la gestion des données dans des équipements de plus faible capacité comme les systèmes sur puce. Ceux-ci représentent une rupture avec l'architecture d'une machine traditionnelle car les SoC encapsulent la CPU, la GPU, la mémoire vive ainsi que d'autres composants sur une même puce [22]. La technologie SoC est généralement utilisée pour réduire le coût des ordinateurs à carte unique, tels que Raspberry Pi, Odroid ou Banana Pi. Ces systèmes sont actuellement utilisés pour une large gamme d'applications, de l'enseignement de l'informatique [1] à l'IoT [14]. Tout naturellement, les SoC jouent un rôle actif dans le fog computing mais à cause de leur faible performance, ils sont plus sensibles au surcoût de la gestion de la DHT, ce qui peut impacter la qualité de service (QoS) offerte aux utilisateurs finaux.

3. Gérer la localité des données dans le DHT

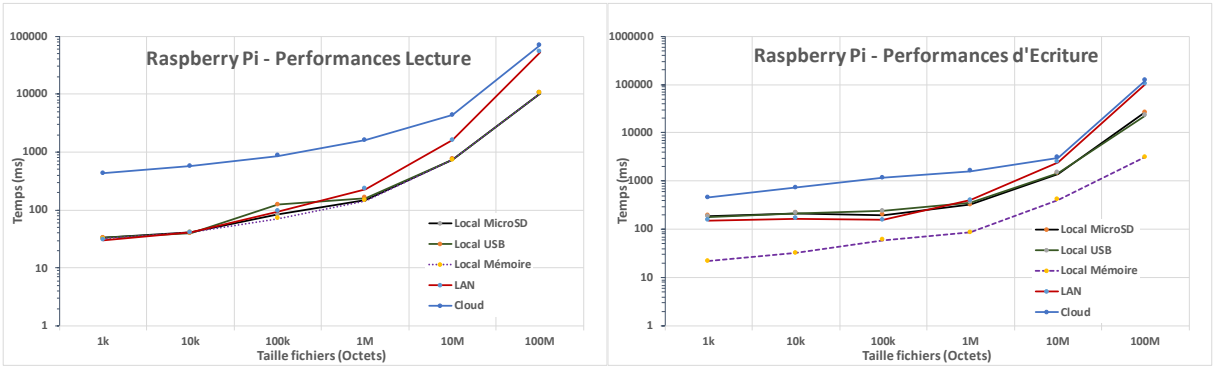
Bien que le Big data s'appuie fortement sur la localité de données pour améliorer ses performances, souvent les ressources et réseaux sont homogènes et de hautes performance [4], telles que dans un datacenter ou une infrastructure cloud. Cependant, en ce qui concerne le fog, la localisation des données a un impact encore plus important, car non seulement les communications et les transferts de données peuvent s'étendre du réseau local au cloud, mais aussi parce que les périphériques sont intrinsèquement hétérogènes.
Ainsi, le premier facteur à prendre en compte est l'endroit où se trouvent les données, ce qu'on appelle la *data-locality*. Bien que certaines applications reposent sur des serveurs de stockage externes (par exemple, un service de stockage sur le cloud), cette solution n'est pas toujours adaptée car elle induit des latences ou des frais de transfert supplémentaires. Dans l'idéal, une solution fog doit pouvoir contrôler l'emplacement de stockage des données.
Cependant, la plupart des DHT souffre d'une perte d'information sur la localisation des données, causée par le mécanisme de distribution fondé sur des fonctions de hachage [23, 6]. En effet, le hachage DHT typique utilise MD5 ou SHA pour calculer les ID de ressources et de nœuds répartis sur le réseau, dans le but d'avoir une répartition uniforme sur l'espace des identifiants et ainsi mieux répartir les données et les charges. Malheureusement, cela complexifie l'optimisation du transfert de données car les informations sur la localité des données sont perdues

[23]. Si certains systèmes permettent de manipuler cette répartition de manière globale [6], nous avons opté par l'utilisation d'un mécanisme de cartographie (*mapping*) décrit dans [19] et qui a l'avantage d'être indépendant du système de DHT sous-jacent et de permettre un partitionnement des données orienté application (i.e., les données seront prioritairement sauvegardés dans les nœuds participant à l'exécution d'une application).
Pour mener à bien ces tests expérimentaux, nous nous appuyons sur la plate-forme distribuée CloudFIT [20], développée en tant que middleware expérimental pour le fog et qui intègre un middleware P2P (TomP2P) et un système de stockage DHT. Nous avons développé un ensemble de tests de performance permettant l'évaluation des temps d'accès aux données stockés à différents endroits d'une DHT. En choisissant différentes localisations pour les données, on souhaite identifier le comportement des dispositifs SoC, permettant ainsi une meilleure compréhension de leurs avantages et limitations. Ceci pourra permettre une meilleure planification des accès au stockage, de manière à garantir les exigences en QoS attendues des solutions fog.

4. Évaluation et Résultats

4.1. Description des expériences

Afin de mener cette expérience, nous avons déployé CloudFIT sur un réseau composé d'un nœud Raspberry Pi 3 (ARM Cortex-A53 à 4 cœurs, 1GB RAM, 100Mbps Ethernet), d'un serveur (AMD Opteron 6164HE à 12 cœurs, 48 Go de RAM, 1Gbps Ethernet) situé sur le même réseau local du Raspberry Pi, et de machines virtuelles dans la plate-forme Google Cloud (GCP). Les machines GCP sont du type *n1-standard-2* (2vCPU, 7,5 Go de RAM) situées aux USA.
Les premières expériences correspondent à des opérations d'écriture et de lecture sur le DHT à partir du Raspberry Pi, avec des fichiers de taille 1 Ko, 10 Ko, 100 Ko, 1 Mo, 10 Mo et 100 Mo. En effet, ceci couvre les tailles de stockage typiques du Big Data (par exemple, Hadoop HDFS utilise des blocs de données de 64 Mo [21]). Trois scénarios ont été envisagés pour vérifier l'impact de la localisation des données : (i) opérations sur la même machine, (ii) sur le même réseau et (iii) sur des nœuds situés dans le cloud. Le premier cas représente les situations dans lesquelles la localité de données est utilisée pour stocker des données directement à la machine qui les traitera (le SoC). Le deuxième scénario représente le cas intermédiaire où les données sont situés dans un équipement à proximité de la machine qui traitera les données. Enfin, le dernier scénario représente les situations où les données se trouvent loin du nœud.
Afin d'éviter des effets de cache, la plate-forme CloudFIT était arrêtée et les fichiers supprimés après chaque exécution. De plus, l'ordre des fichiers a été mélangé pour éviter des effets de pipeline sur le réseau, et au moins 10 mesures ont été effectuées pour chaque scénario.
Nous avons également expérimenté trois supports de stockage différents : la carte mémoire MicroSD intégrée au Raspberry Pi (Samsung Evo, 16GB, classe 10), un disque dur externe en connexion USB 2.0 et un stockage uniquement en mémoire. La comparaison entre la carte mémoire et le lecteur de disque USB externe est motivée par la différence de performances relative entre ces deux supports : une carte MicroSD de classe 10 supporterait au moins 10 Mo/s en écriture (avec une vitesse de lecture dépendant du constructeur), tandis qu'un disque externe USB pourrait atteindre environ 20 Mo/s (et cela uniquement parce que le chipset USB 2.0 du Raspberry Pi l'empêche d'aller plus vite). Comme les deux méthodes de stockage souffrent de goulots d'étranglement en raison de la configuration du chipset Raspberry Pi, nous avons également inclus des tests de stockage où les données sont gardées exclusivement en mémoire. Cette dernière option pourrait profiter aux applications déployées dans des nœuds "passerelles", censés stocker des données intermédiaires et les prétraiter avant de les envoyer à d'autres nœuds ou au cloud.



(a) (b)

FIGURE 1 – Performances de (a) lecture et (b) écriture d'un Raspberry Pi

4.2. Résultats et Analyse

La Figure 1a représente les différents temps pour les opérations de lecture des données, selon la taille des données et le scénario (même nœud, même réseau, sur le cloud). Nous pouvons observer que pour la lecture des données, les scénarios "même réseau" et local présentent des performances similaires pour la lecture de messages de moins de 1 Mo. Cependant, pour les messages plus volumineux, le temps de lecture entre deux nœuds devient beaucoup plus important, passant à des niveaux presque comparables à ceux d'accès aux nœuds distants, qui pourtant sont les plus pénalisés par la latence des communications. Ce ralentissement met en évidence non seulement les limites du réseau (réseau à 100 Mbits/s), mais également le temps système nécessaire pour demander une ressource par le biais du DHT, qui finit par masquer l'impact de la latence sur les transmissions.

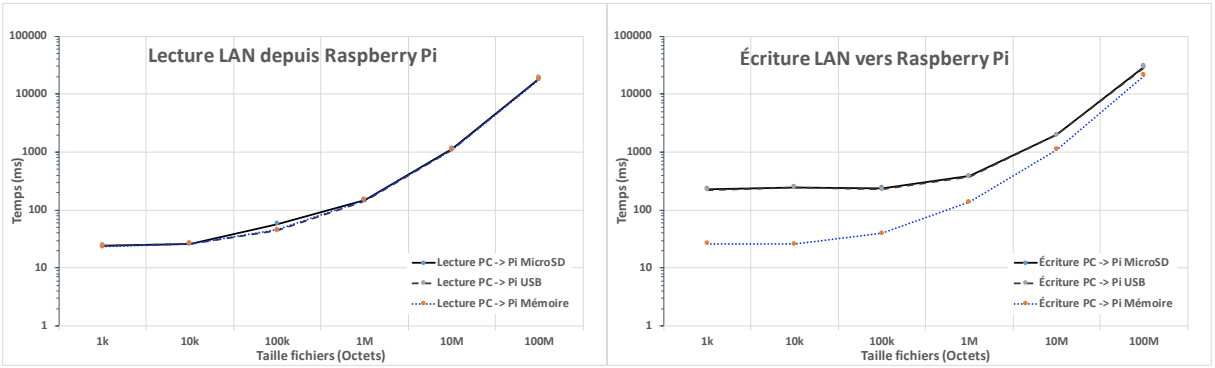
En ce qui concerne l'accès à la DHT locale, les trois options de stockage affichent des performances presque similaires. Une explication possible est que l'implémentation DHT conserve les données récentes dans la mémoire, masquant ainsi le coût supplémentaire d'accès aux unités de stockage. Des améliorations futures sur le protocole d'expérimentation pourront aider à mieux comprendre ce résultat.

Pour ce qui est de l'écriture des données sur la DHT, la Figure 1b démontre, sans surprise, que l'option de stockage en mémoire locale est beaucoup plus rapide, car elle ne paie pas la surcharge d'accès au stockage persistant. En ce qui concerne les options de stockage MicroSD et de stockage sur disque USB, leurs performances sont au même ordre de grandeur, mais l'échelle logarithmique des images ne permet pas une analyse plus détaillée. Un examen plus attentif montre que le disque USB devrait être préféré, car il faut en moyenne 22,7 secondes pour écrire 100 Mo sur le DHT, contre 27,3 secondes pour le stockage MicroSD (une amélioration de 16%).

Dans le cas du scénario "même réseau", nous retrouvons des comportements presque similaires à ceux observés en lecture, où le temps nécessaire pour le stockage de petits blocs de données (jusqu'à 1MB) affiche des performances similaires au stockage en local, alors que pour des données de plus grande taille le coût tend vers celui d'un stockage sur le cloud.

Les résultats présentés sur ces deux figures semblent démontrer qu'un accès en lecture locale à partir d'un Raspberry Pi ne présente un avantage considérable que si les données sont de plus grande taille. Le même constat peut se faire sur l'écriture des données à partir du Raspberry, à l'exception du cas où le stockage se fait directement en mémoire.

Il faut aussi considérer que, dans la plupart des cas d'usage du fog, les sources de données se situent sur le réseau proche des machines "passerelle" (dispositifs IoT, etc.). Ainsi, nous avons procédé à un deuxième ensemble de mesures où une machine située sur le réseau local lit/écrit



(a) (b)

FIGURE 2 – Lecture (a) et écriture (b) sur le Raspberry Pi à partir d'une autre machine

des données sur la DHT à un emplacement qui correspond au nœud Raspberry Pi. Les Figures 2a et 2b illustrent les performances d'écriture et de lecture lorsque l'autre nœud accède au Raspberry Pi. Si le comportement semble similaire au cas précédent (avantage du stockage mémoire en écriture, performances similaires en lecture), il faut remarquer que le temps d'écriture total est 4 fois plus long lorsque le Raspberry Pi doit héberger les données par rapport au temps nécessaire lorsque le Raspberry Pi écrit des données sur autre nœud. Connaître ces informations est important car cela pourrait aider à identifier le meilleur rôle des périphériques de type Raspberry Pi dans le fog. Par exemple, nos observations suggèrent que ces périphériques bas de gamme sont mieux adaptés pour collecter lentement des données à partir de périphériques proches (capteurs, IoT), effectuer de petits calculs et transmettre les données à d'autres nœuds, plutôt que de recevoir de gros volumes de données provenant d'ailleurs.

5. Conclusions

Le Fog et le Edge Computing étendent le paradigme du Cloud Computing à la périphérie du réseau, en s'appuyant sur des services intelligemment distribués pour répondre aux besoins des applications. De manière générale, ces approches visent à traiter les données à la périphérie d'un réseau plutôt que de déléguer ce traitement à un cloud ou à un datacenter distant. Elles permettraient le prétraitement des données et la génération de connaissances au plus près des sources des données et ainsi améliorer la qualité de service.
Souvent, ces services sont déployés sur des dispositifs "passerelle" caractérisés par une puissance de calcul relativement limitée, ce qui peut porter atteinte aux objectifs de qualité de service à l'origine de la démarche fog. Pour mieux comprendre les enjeux, cet article s'intéresse aux performances de l'accès aux données distribuées. Plus exactement, on évalue l'utilisation des DHT comme support de stockage pour les environnements de type fog computing. En observant le comportement des applications et des périphériques, nous sommes en mesure d'identifier des situations potentiellement en mesure de gêner la performance.
Nous pensons que l'analyse préliminaire proposée ici contribue à améliorer la compréhension des contraintes des flux de données dans le fog, ce que peut se traduire pour des améliorations dans la gestion des données et dans le déploiement des nœuds. Nos directions de travaux futurs comprennent l'analyse de la performance d'applications utilisant beaucoup de données, l'élaboration de stratégies de lecture/écriture sensibles aux différences de performance des dispositifs, et la comparaison avec d'autres plateformes fog.

Bibliographie

1. Ali (M.), Vlaskamp (J. H. A.), Eddin (N. N.), Falconer (B.) et Oram (C.). – Technical development and socioeconomic implications of the Raspberry Pi as a learning tool in developing countries. – In *Computer Science and Electronic Engineering Conf. (CEEC)*, pp. 103–108. IEEE, 2013.
2. Baldauf (M.), Dustdar (S.) et Rosenberg (F.). – A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 2, n4, 2007, pp. 263–277.
3. Bonomi (F.), Milito (R.), Zhu (J.) et Addepalli (S.). – Fog computing and its role in the internet of things. – In *Proceedings of the 1st MCC Workshop on Mobile Cloud Computing*, *MCC '12*, MCC '12, pp. 13–16, New York, NY, USA, 2012. ACM.
4. Cassales (G. W.), Charao (A. S.), Kirsch-Pinheiro (M.), Souveyet (C.) et Steffenel (L. A.). – Improving the performance of apache hadoop on pervasive environments through context-aware scheduling. *Springer Journal of Ambient Intelligence and Humanized Computing*, vol. 7, n3, 2016, pp. 333–345.
5. Chen (S.), Zhang (T.) et Shi (W.). – Fog computing. *IEEE Internet Computing*, vol. 21, n2, Mar 2017, pp. 4–6.
6. Confais (B.), Lebre (A.) et Parrein (B.). – *Performance Analysis of Object Store Systems in a Fog and Edge Computing Infrastructure*, pp. 40–79. – Berlin, Heidelberg, Springer Berlin Heidelberg, 2017.
7. Dey (S.), Mukherjee (A.), Paul (H. S.) et Pal (A.). – Challenges of using edge devices in iot computation grids. – In *Int. Conf. on Parallel and Distributed Systems*, pp. 564–569, 2013.
8. ETSI. – Mobile-edge computing - introductory technical white paper. – https ://portal.etsi.org/.
9. Fazio (M.), Celesti (A.), Puliafito (A.) et Villari (M.). – Big data storage in the cloud for smart environment monitoring. *Procedia Computer Science*, vol. 52, 2015, pp. 500 – 506. – The 6th International Conference on Ambient Systems, Networks and Technologies (ANT-2015).
10. Gubbi (J.), Buyya (R.), Marusic (S.) et Palaniswami (M.). – Internet of things (iot) : A vision, architectural elements, and future directions. *Future Generation Computer Systems*, vol. 29, n 7, 2013, pp. 1645 – 1660.
11. Hao (Z.), Novak (E.), Yi (S.) et Li (Q.). – Challenges and software architecture for fog computing. *IEEE Internet Computing*, vol. 21, n2, 2017, pp. 44–53.
12. Hofmann (P.) et Woods (D.). – Cloud computing : The limits of public clouds for business applications. *Internet Computing, IEEE*, vol. 14, n6, Nov 2010, pp. 90–93.
13. Huang (D.) et Wu (H.) (édité par). – *Mobile Cloud Computing*. – Morgan Kaufmann, 2018.
14. Molano (J. I. R.), Betancourt (D.) et Gómez (G.). – Internet of things : A prototype architecture using a Raspberry Pi. *In : Knowledge Management in Organizations*, pp. 618–631. – Springer, 2015.
15. Olaniyan (R.), Fadahunsi (O.), Maheswaran (M.) et Zhani (M. F.). – Opportunistic edge computing : Concepts, opportunities and research challenges. *Future Generation Computer Systems*, vol. 89, 2018, pp. 633 – 645.
16. Parashar (M.) et Pierson (J.-M.). – Pervasive grids : Challenges and opportunities. *In : Handbook of Research on Scalable Computing Technologies*, éd. par Li (K.), Hsu (C.), Yang (L.), Dongarra (J.) et Zima (H.), pp. 14–30. – IGI Global, 2010.
17. Satyanarayanan (M.), Bahl (P.), Caceres (R.) et Davies (N.). – The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, vol. 8, n4, décembre 2009, pp. 14–23.
18. Schadt (E. E.), Linderman (M. D.), Sorenson (J.), Lee (L.) et Nolan (G. P.). – Computational solutions to large-scale data management and analysis. *Nature Reviews Genetics*, vol. 11, n9,

Sept 2010, pp. 647–657.

19. Steffenel (L. A.). – Improving the performance of fog computing through the use of data locality. – In *30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD 2018)*, Lyon, France, 2018.
20. Steffenel (L. A.) et Kirsch-Pinheiro (M.). – When the cloud goes pervasive : approaches for IoT PaaS on a mobiquitous world. – In Springer (édité par), *EAI International Conference on Cloud, Networking for IoT systems (CN4IoT 2015)*, *LNICST*, number 169 in LNICST, pp. 347–356, Rome, Italy, octobre 2015.
21. White (T.). – *Hadoop : The definitive guide*. – Yahoo Press, O'Reilly, 2010, 2nd edition édition.
22. Wolf (W.), Jerraya (A. A.) et Martin (G.). – Multiprocessor system-on-chip (MPSoC) technology. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 27, n 10, 2008, pp. 1701–1713.
23. Wu (D.), Tian (Y.) et Ng (K.-W.). – Aurelia : Building locality-preserving overlay network over heterogeneous p2p environments. – In *Proc. of the International Conference on Parallel and Distributed Processing and Applications*, *ISPA'05*, ISPA'05, pp. 1–8. Springer, 2005.