Stratégies Multi-Échelles pour les Environnements Pervasifs et l'Internet des Objets

Luiz Angelo Steffenel, Manuele Kirsch Pinheiro

Stratégies Multi-Échelles pour les Environnements Pervasifs et l'Internet des Objets

Luiz Angelo Steffenel
Université de Reims Champagne-Ardenne
Laboratoire CReSTIC
Reims, France
luiz-angelo.steffenel@univ-reims.fr

Manuele Kirsch Pinheiro
Université Paris 1 Panthéon-Sorbonne
Laboratoire CRI
Paris, France
manuele.kirsch-pinheiro@univ-paris1.fr

ABSTRACT

Avec la dissémination des dispositifs portables (smartphones, tablets, nano-ordinateurs, etc.) et leur augmentation en puissance de calcul, il est devenu essentiel de mettre en place une plate-forme pour la coordination des dispositifs et la gestion de leurs informations. Afin de garantir l'efficacité et le passage à l'échelle dans des environnements tels que l'Internet of Things (IoT), nous devons repenser les stratégies de transmission et d'analyse des données de manière à éviter la centralisation des ressources de stockage et de traitement. Dans ce travail, nous présentons des directions pour le développement de plates-formes pervasives multi-échelles, capables de mieux exploiter les ressources et les capacités de ces dispositifs afin d'offrir une meilleure qualité de service aux applications.

CCS Concepts

•**Information systems** → **Distributed storage ;**
•**Human-centered computing** → **Ubiquitous and mobile computing systems and tools ;**
•**Computing methodologies** → *Distributed algorithms ;*
•**Computer systems organization** → *Fault-tolerant network topologies ;*

Keywords

Multi-échelle ; Internet des Objets ; Environnements pervasifs ; P2P

1. INTRODUCTION

Avec l'augmentation exponentielle du nombre de dispositifs informatiques de proximité (smartphones, tablettes, nano-ordinateurs, etc.), il est important de comprendre comment organiser ces ressources et comment gérer l'information de manière adaptée à ces environnements pervasifs. L'avènement de l'Internet des Objets (*Internet of Things* - IoT) représente une nouvelle tendance de l'industrie informatique où l'environnement physique est peuplé d'objets interconnectés et communicants qui interagissent les uns avec les autres et avec l'environnement lui-même.

Plusieurs facteurs ont contribué au développement de l'Internet des Objets, dont l'augmentation de la bande passante et de la puissance de calcul des appareils, couplés avec un coût décroissant de capteurs [11]. De nos jours, il est possible d'équiper la quasi-totalité des dispositifs quotidiens d'une interface sans fil, rendant ainsi possible leur interaction [16]. De telles capacités de communication ouvrent d'innombrables possibilités dans plusieurs domaines tels que la santé et les villes intelligentes, mais soulève également plusieurs défis concernant l'évolutivité, l'hétérogénéité et la dynamicité du réseau. Les principales limitations d'utilisation de ces dispositifs ne sont pas liées à leur capacité de communication (WiFi, Bluetooth, etc.), mais surtout à la difficulté d'une application à mieux tirer profit de cette interconnexion à d'autres appareils. Le potentiel de l'IoT ne sera pleinement atteint que si les données issues de l'IoT puissent être analysées et explorées convenablement par les applications. En effet, la force de ce concept réside dans l'intégration transparente des capteurs, des actionneurs et d'autres dispositifs, ce qui permet l'interaction et la collecte d'informations à grande échelle.

Malheureusement l'agrégation et l'analyse des données IoT sont souvent effectuées sur les infrastructures déportées de type *cloud computing*. Plusieurs travaux [13, 9, 7] comptent sur ces infrastructures car elles offrent de la puissance de calcul et de la flexibilité pour l'exécution de services et applications [21]. Malgré leurs avantages, les plates-formes de type *cloud* ont aussi quelques inconvénients importants. En effet, le transfert de données peut être considérablement coûteux et prendre du temps. En outre, les applications qui dépendent entièrement des services distants peuvent échouer si la connexion est défectueuse ou trop lente.

Par conséquent, nous devons repenser la façon de transmettre, de stocker et d'analyser les données au plus près des sources/consommateurs. Cette préoccupation a conduit les chercheurs à développer une série de solutions alternatives au *cloud computing*, telles que les grilles pervasives [15], le *mobile edge computing* [5, 6, 20], le *fog computing* [1] ou bien le *edge-centered computing* [8]. Toutes ces alternatives partagent un même objectif : utiliser la puissance de calcul des dispositifs environnants pour effectuer des tâches habituellement déléguées à une installation distante.

Pour répondre aux défis de l'IoT, ce travail préconise le développement de plates-formes de proximité capables d'assurer la coordination et la redistribution des tâches entre les

Ubimob 2016 Lorient, France

DOI : 10.475/123_4

appareils (en fonction de leurs capacités), améliorant ainsi l'utilisation des ressources et la qualité de service. Nous présentons des solutions architecturales et pratiques à mettre en œuvre pour la création d'une plate-forme multi-échelle reliée par un réseau P2P et capable de supporter le stockage et le traitement de données de l'IoT.

La suite de cet article est organisée comme suit : la Section 2 présente l'état de l'art sur le calcul dans les environnements pervasifs, ainsi que la définition de l'informatique multi-échelle. La Section 3 analyse les principales exigences pour un cadre multi-échelle et illustre comment une telle approche est mise en œuvre sur la plate-forme de calcul CloudFIT. Finalement, la Section 5 présente nos conclusions et travaux futurs.

2. ÉTAT DE L'ART

La diffusion des dispositifs de proximité avec des capacités de calcul non-négligeables (smartphones, tablettes, ordinateurs portables et nano-ordinateurs tels que le Raspberry Pi) encourage l'intégration de ces dispositifs dans le traitement des données.

Les travaux sur l'*edge computing* et le *fog computing* partagent souvent les mêmes définitions [24]. En effet, le *fog computing* a été défini par Cisco [4] comme "un paradigme qui étend le *cloud computing* et des services à la périphérie du réseau", tandis que le *(mobile) edge computing* vise à transformer les stations de base proches en "centres de services intelligents capables de fournir des services hautement personnalisés" [24]. Des exemples de *edge/fog* comprennent des services *fog* [1] et les *cloudlets* [20], tous les deux proposant le déploiement des serveurs de proximité offrant des services avec une latence réduite. A quelques exceptions près, comme [5], ces travaux considèrent que les dispositifs IoT ne contribuent pas à l'effort de calcul, restant dépendants d'un service tiers (à proximité ou à distance).

Garcia Lopez et al. [8] explorent une autre facette de l'*edge computing* en se concentrant sur le rôle de l'homme dans la boucle de contrôle. En effet, ces auteurs affirment la nécessité de recentrer le contrôle sur les équipements situés au bord du réseau, au lieu de simplement les considérer comme une première couche de calcul reliée à un réseau plus grand et plus puissant. Malheureusement, dans cette définition les dispositifs IoT ne contribuent pas non plus aux efforts de calcul, étant considérés comme des capteurs/actionneurs pilotés par les interactions entre l'homme et l'*edge*.

Bien que ces travaux préconisent la nécessité d'un environnement informatique de proximité, ils oublient souvent de détailler l'interconnexion ou les exigences de coordination entre les processus. Cela est particulièrement nécessaire dans l'optique de l'IoT, qui impose des défis importants pour l'évolutivité, la dynamicité et l'hétérogénéité des ressources. Afin de gérer efficacement les environnements pervasifs IoT, nous proposons l'utilisation de stratégies multi-échelles pour la gestion et la coordination des périphériques.

Dans la littérature nous trouvons, par exemple, la notion de grille pervasive [15], qui vise l'intégration des dispositifs de détection/d'actionnement ainsi que des systèmes de haute performance classiques. Ces grilles reposent sur l'utilisation des ressources habituellement sous-utilisées, composant ainsi une plate-forme de calcul dynamique [22]. En effet, les grilles pervasives offrent la possibilité d'intégrer les différentes ressources disponibles allant des petits appareils de type Raspberry Pi jusqu'aux machines virtuelles déployées sur les infrastructures d'un datacenter. Pour l'IoT, les grilles pervasives représentent une opportunité de déployer des tâches informatiques sur des ressources situées à proximité des dispositifs IoT, minimisant ainsi le transfert de données vers un réseau distant. De plus, selon les besoins, ces tâches peuvent être allouées aux ressources avec la capacité de calcul adéquate à chaque service, sans avoir à externaliser les données et les services. C'est par exemple le cas illustré par [17], où les dispositifs de calcul d'une maison (laptops, tablets ou nano-ordinateurs) sont utilisés en réseau afin de traiter les données relatives au déplacement des occupants de la maison et éventuellement détecter des chutes ou des situations nécessitant une intervention extérieure.

Un autre avantage des grilles pervasives est son indépendance par rapport à des architectures et services opérateur. Malgré l'appel à la décentralisation et à l'affranchissement du "tout cloud" prôné par les premiers travaux sur l'edge-computing et le fog-computing, nous observons une "appropriation" de ces concepts par les grands opérateurs du marché télécom et cloud tels que Cisco, Intel ou Microsoft, réunis par exemple au sein de l'Open Fog Consortium [1] afin de créer une architecture de référence pour le fog computing. Bien que de telles initiatives sont nécessaires pour la maturation d'une technologie, elles sont souvent source de contraintes au déploiement de plates-formes légères, ce qui est notre objectif principal.

Afin de répondre aux différents besoins des architectures et applications IoT, les grilles pervasives doivent être enrichies avec le concept de système multi-échelle. Les *systèmes multi-échelles* sont des systèmes distribués où les services sont organisés en couches à travers une ou plusieurs dimensions (dispositifs, réseau, localisation géographique, etc.), chaque couche fournissant un niveau de service supplémentaire qui peut être consulté en fonction du contexte de l'appareil [18, 19]. En vertu de cette approche, des actions primaires peuvent être décidées/interprétées à proximité, tandis qu'une analyse plus poussée de l'information peut être effectuée par des serveurs externes. Cette analyse stratifiée peut également être utilisée pour renforcer les aspects liés à la vie privée comme, par exemple, l'anonymisation des données qui seront externalisées. Nous croyons que ce concept offre plusieurs niveaux de granularité et d'interconnexion nécessaires à l'autonomie des dispositifs IoT et permet des services plus réactifs et de meilleure qualité. Ceci est notamment utile dans des domaines tels que la domotique, où l'adaptation au contexte et le respect de la vie privée sont des facteurs clé.

Dans la section suivante, nous énumérons certaines exigences pour le développement d'une plate-forme informatique multi-échelle.

3. CALCUL MULTI-ÉCHELLE

Comme indiqué précédemment, la plupart des travaux sur le *edge/fog computing* ont tendance à faire une distinction entre l'utilisateur final (ou les périphériques finaux) et les dispositifs qui se trouvent sur la frontière la plus proche de l'Internet/*cloud*. Dans de telles approches, les appareils IoT sont des simples clients des services déployés dans un voisinage proche. Contrairement au *fog/edge computing*, les grilles pervasives supposent que tous les dispositifs peuvent contribuer au calcul des tâches selon leurs propres capacités

1. http://www.openfogconsortium.org/

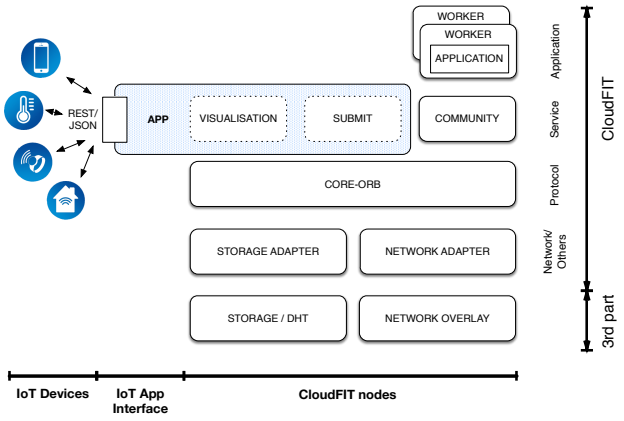

Figure 1 : La pile d'exécution CloudFIT et l'interface pour les dispositifs IoT

et ressources disponibles. Les dispositifs IoT peuvent alors devenir des acteurs de l'environnement et non seulement des clients passifs. Les concepts de grilles pervasives et de calcul multi-échelle peuvent ainsi être utilisés pour intégrer les dispositifs IoT dans l'environnement informatique.

Cependant, afin que les applications puissent profiter pleinement de ces environnements, différents défis doivent être relevés. Dans ce travail, nous analysons un ensemble d'exigences qui nous paraissent essentielles pour cette intégration et nous illustrons leur prise en charge par une plate-forme de calcul multi-échelle basée sur CloudFIT [22]. CloudFIT est un middleware de calcul P2P basé sur le paradigme FIIT (*Finite Independent Irregular Tasks*). La version actuelle prend en charge le système TomP2P [2] ainsi que son API DHT (basée sur Kademlia), offrant des possibilités intéressantes pour la gestion des données. CloudFIT est développé en Java et peut donc être déployé sur une large gamme de dispositifs (même une version Android est en cours de développement). Afin de rendre son déploiement simple et rapide, nous avons pris soin de minimiser les dépendances envers les bibliothèques tiers, ce qui permet le lancement de la plate-forme avec un simple fichier *jar*.

La flexibilité de lancement de CloudFIT permet son exécution directement sur certains appareils IoT récents, souvent localisés au bord du *edge*. Dans le cas des capteurs isolés ou de microcontrôleurs de type Arduino, incapables d'exécuter un code Java, CloudFIT peut être utilisé en tant que *gateway* grâce à une interface front-end permettant le stockage des données et le déclenchement d'actions, comme illustré sur le côté gauche de la Fig. 1.

Les prochaines sections décrivent ainsi les défis pour la mise en place d'une plate-forme multi-échelle et comment nous avons adapté CloudFIT à ces fins.

3.1 Coordination et cloisonnement

L'un des principaux défis avec le calcul multi-échelle dans un environnement IoT est celui de permettre à la fois un contrôle précis sur les ressources et une interaction simplifiée avec le reste du réseau. En effet, les environnements IoT se caractérisent par leur échelle très variée, pouvant aller de quelques objets à plusieurs milliards. En concevant chaque couche comme un cluster, nous pouvons utiliser un réseau P2P soutenant le déploiement de chaque couche. Cependant, les réseaux P2P les plus connus organisent les nœuds indistinctement de leur emplacement réel, ce qui empêche l'établissement d'un regroupement de nœuds pour fournir des services à faible latence. Pour contourner ces inconvénients, nous considérons que le réseau P2P doit être enrichi par l'utilisation de techniques de cloisonnement. En effet, le regroupement des ressources sous la forme de *clusters* est une manière efficace pour organiser les couches de calcul multi-échelle et ainsi fournir une base de coordination pour le déploiement efficace des services.

Plusieurs approches de *clustering* sont proposées dans la littérature [10] et utilisées, par exemple, pour le routage dans les réseaux de capteur sans fil. La plupart des algorithmes de *clustering* utilisent des paramètres simples comme la densité du réseau environnant, choisissant un *cluster-head* en fonction de leurs identités uniques (ID). Malheureusement, ces métriques sont insuffisantes pour assurer le *clustering* dans un scénario hétérogène comme celui de l'IoT, vu qu'elles ne permettent pas d'exprimer les besoins du calcul multi-échelle. Au contraire, le regroupement des nœuds doit être défini à la fois de manière à co-localiser les données et les ressources de calcul nécessaires et aussi à définir des couches de calcul reliant les dispositifs proches et plus éloignés (jusqu'au cloud). Ainsi, afin de s'adapter à l'hétérogénéité des environnements pervasifs et l'IoT, les métriques de *clustering* doivent être étendues pour inclure des informations de contexte telles que la proximité, les capacités informatiques et de stockage des appareils, la fiabilité et même le niveau d'autorisation/confiance des nœuds collaborateurs.

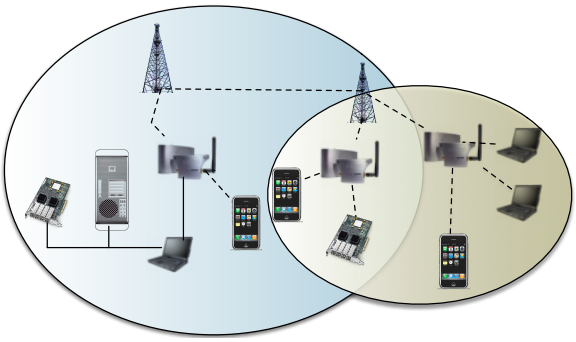

Figure 2 : Des communautés interconnectées dans une plate-forme multi-échelle

Dans le cas de CloudFIT, ce *clustering* peut être mis en œuvre grâce au concept de *communautés*, des sous-ensembles de nœuds qui peuvent être adressés séparément et donc utilisés pour répartir/cloisonner les opérations. Dans le cas d'un *clustering* automatique, la liste de membres est gérée par le *cluster-head*. Les membres de la liste peuvent donc être contactés en utilisant des procédures spécifiques telles que les diffusions structurées de TomP2P. Bien entendu, un nœud peut appartenir à plusieurs communautés, permettant à des données et des tâches de circuler entre les différentes couches multi-échelle, comme illustré en Fig. 2. Cette organisation à plusieurs niveaux, déjà utilisée par exemple par Lim et al. [12] pour la diffusion d'informations de contexte dans un environnement IoT, permettrait de mieux coordonner la communication entre ces ressources et d'ainsi mieux gérer la

2. http://tomp2p.net

variabilité d'échelle de ces environnements.

3.2 Contexte et ordonnancement

Dans les environnements pervasifs, la sensibilité au contexte est essentielle pour la planification des tâches vu que la performance varie beaucoup d'un appareil à l'autre [5]. Les informations contextuelles telles que la puissance de calcul, la mémoire disponible et l'espace de stockage peuvent être utiles pour améliorer l'exécution en assignant des tâches aux dispositifs les plus adéquats. Avec d'autres paramètres tels que la proximité géographique, la latence réseau et le niveau de fiabilité d'un nœud, la sensibilité au contexte est un outil nécessaire à la clusterisation des nœuds et à la composition de services multi-échelles.

Bien que la collecte d'informations de contexte et sa mise en œuvre est au-delà de la portée de cet article, il faut noter que les algorithmes d'ordonnancement doivent néanmoins être assez permissifs, avec l'exclusion des nœuds uniquement lorsque les exigences spécifiques de l'application ne sont pas respectées (par exemple, un minimum de capacité mémoire). En effet, la dynamicité des environnements pervasifs due entre autres à la mobilité de certains appareils, rend particulièrement difficile d'anticiper quelles ressources seront disponibles et dans quelle mesure elles le seront. Dans ces conditions, il est préférable d'avoir des nœuds lents mais qui contribuent à l'effort que d'avoir un blocage par manque de ressources ou une surcharge trop importante des nœuds rapides [3, 25].

Les applications CloudFIT étant décrites par un ensemble fini de tâches qui peuvent être réparties entre les nœuds, des informations de base telles que le nombre de tâches parallèles qu'un nœud peut supporter sont déjà utilisées par l'ordonnanceur. Si l'ordonnanceur par défaut utilise une approche totalement distribuée (exécution *best-effort* associée à la diffusion des tâches finies), il peut être enrichi avec d'autres éléments tels que la mémoire disponible ou la vitesse relative des CPUs. La collecte de ces éléments peut se faire grâce au collecteur de contexte que nous avons proposé et utilisé dans [3], présenté sur la Fig. 3.

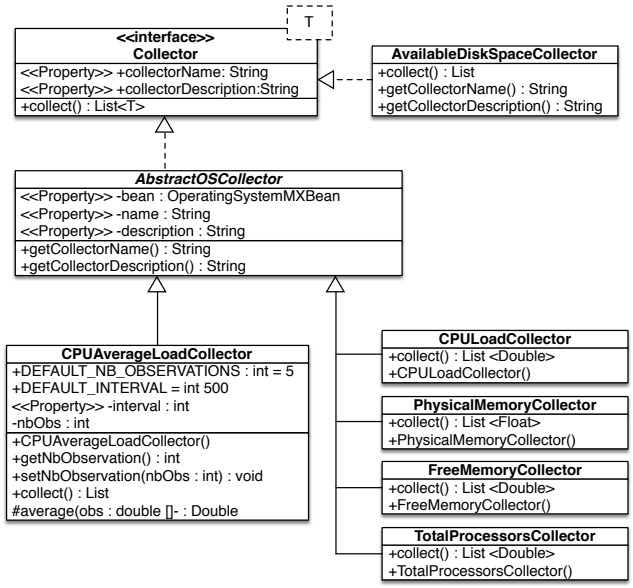

Figure 3 : Architecture pour un collecteur de contexte [2]

3.3 Accès aux données

Dans le cadre du traitement de données issues des dispositifs de l'IoT, un autre facteur à prendre en compte est celui de la performance liée à l'accès et à la gestion des données. En effet, la plupart des opérations impliquent la collecte, la transformation et l'analyse des données, et les plates-formes telles qu'Apache Hadoop se sont illustrées par leur capacité d'optimiser l'accès aux données en plaçant les tâches préférentiellement là où les données sont présentes (grâce au concept de la *data locality*).

Dans le passé [22] nous avons déjà mené des expérimentations avec CloudFIT démontrant que des performances similaires ou supérieures à celles d'Apache Hadoop pourraient être atteintes avec un système P2P, comme illustré en Fig. 4. De plus, n'étant pas limitées à un paradigme strict *map-reduce*, les applications développées sur CloudFIT peuvent en théorie s'aligner sur d'autres frameworks plus récents tels que Apache Spark. En attendant des comparaisons de performance avec ces frameworks dans des travaux futurs, nous avons pu identifier deux pistes pour améliorer la performance de CloudFIT : (*i*) mieux gérer la surcharge de gestion des données sur un nœud et (*ii*) prendre en charge de la *data locality*.

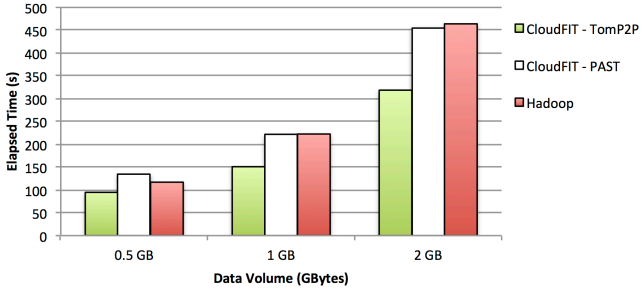

Figure 4 : Comparaison des temps d'exécution de WordCount avec CloudFIT et Hadoop [22]

Dans le premier cas, nous avons constaté un fort écart de performances d'accès aux données lorsque nous déployons CloudFIT dans un environnement hétérogène. Ces écarts de performance sont en effet une combinaison de la vitesse d'accès aux données et de la surcharge de gestion des systèmes de stockage (voir aussi les limitations physiques de stockage), et affectent notamment les dispositifs de faible capacité situés à la frontière du *edge*. Ainsi, par exemple, un Raspberry Pi est fortement pénalisé par la vitesse et la capacité de stockage de sa carte SD, malgré une bonne capacité de calcul. Afin de contourner cette limitation, nous avons modifié la couche de stockage afin que les nœuds puissent choisir d'agir seulement en tant que clients distants. Ces nœuds peuvent donc interroger la surcouche de stockage P2P via le réseau mais ils ne sont plus obligés à gérer le stockage, réduisant leur surcharge et aussi leur utilisation du disque.

Pour ce qui est de la prise en charge de la *data-locality*, ceci est un problème plus général qui affecte la majorité des architectures de stockage P2P. En effet, les API de stockage P2P sont souvent basées sur les tables de hachage distribuées (DHT). Ces DHTs sont conçues de manière à répartir les données sur le réseau et les répliquer lorsque cela est possible, notamment afin d'éviter la perte de données en cas de désabonnement (*churn*). Un inconvénient de cette procédure est qu'on observe une perte d'information concernant la lo-

calisation des données [26], rendant difficile l'optimisation des transferts réseau.

A l'instar d'autres systèmes de P2P, La bibliothèque Tom P2P utilisée par CloudFIT ne permet pas encore à un nœud de faire la différence entre les données locales ou distantes. Nous sommes en contact avec l'équipe de développement TomP2P afin d'intégrer une telle fonctionnalité dans les futures versions, mais en attendant nous avons développé une stratégie pour renforcer la proximité des données grâce à un calcul personnalisé de la clé de localisation des ressources.

Ceci est possible car contrairement à la plupart des systèmes de P2P qui ont seulement une clé de hachage unique, TomP2P identifie les ressources par quatre clés différentes $\{k_l, k_d, k_c, k_v\}$, selon la hiérarchie suivante :

— k_l - clé de localisation, utilisée pour la localisation d'une ressource dans la DHT
— k_d - clé de domaine, fonctionne comme une clé d'authentification, permet la séparation des données
— k_c - clé de contenu, permet d'identifier une ressource. Par défaut est identique à la clé de localisation
— k_v - clé de version, permet la gestion de versions multiples d'une ressource

La clé de localisation est celle qui s'approche le plus des clés DHT traditionnelles, ayant par fonction l'association d'une ressource (copie primaire ou index) au nœud avec l'ID le plus proche. Sans aucune instruction supplémentaire la clé de localisation et la clé de contenu sont les mêmes, mais peuvent être différentes, par exemple, en cas de collision (deux ressources avec la même clé de localisation).

La clé de domaine est liée à un mécanisme d'authentification simple de TomP2P, son but étant de renforcer le cloisonnement des données des différents clients. Dans le cas de CloudFIT, la clé de domaine est utilisée comme un *namespace* pour la séparation des données de différentes communautés ou jobs de calcul. Si les applications nécessitent une plus grande confidentialité, il suffit de chiffrer les données au niveau des applications, ceci n'étant pas généralisé au niveau de CloudFIT pour des raisons de performance.

Finalement, la clé de version permet la coexistence de différentes versions d'une ressource, ce qui permet une meilleure gestion des données "mutables", avec par exemple un accès à l'historique des modifications ou l'écriture en parallèle d'une ressource par plusieurs nœuds. Cette clé de version est utilisée dans les nouvelles versions de l'application MapReduce développée sur CloudFIT.

Ainsi, afin de renforcer la *data locality*, nous travaillons sur la manipulation des clés de localisation. En effet, il est possible de forcer la clé de localisation d'une ressource et ainsi obliger le stockage d'au moins une copie dans un nœud indiqué. Nous avons donc implémenté une procédure pour le découplage entre la clé de localisation et la clé de contenu grâce à une double fonction de hachage. Dans un premier moment, la clé de contenu est obtenue avec une méthode de hachage classique. Ensuite, la clé de localisation est calculée en faisant une association limitée aux ID des nœuds d'une communauté. La Fig. 5 montre l'exemple de cette cartographie en calculant la clé de localisation d'une ressource r_3 par rapport à une communauté $Comm_1$.

Cette cartographie augmente la probabilité que la copie primaire d'une donnée se trouve parmi les membres de la communauté, sans pour autant empêcher la réplication des données sur d'autres nœuds (même en dehors de la communauté). La seule contrainte de cette approche est que les

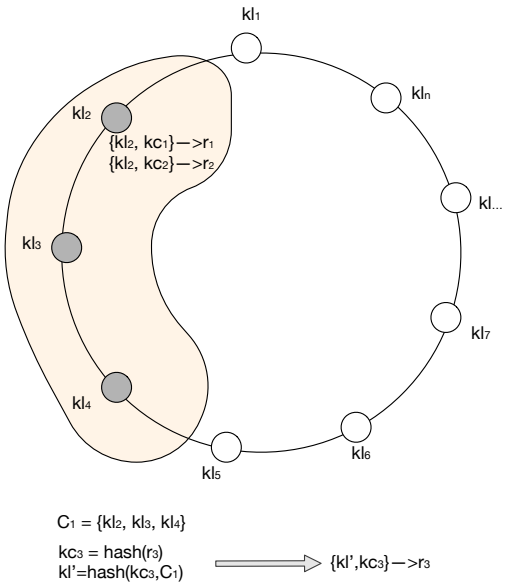

Figure 5 : Calcul des clés de localisation renforçant la *data-locality*

applications doivent garder en mémoire les clés de localisation utilisées au lieu de les régénérer selon la demande. En effet, toute modification de la liste de membres d'une communauté risque de confondre la recherche d'une ressource car la clé k_l est liée à la taille et à la composition de la liste.

4. CAS D'UTILISATION

Dans cette section nous décrivons un cas d'utilisation de CloudFIT avec une application réelle dans le domaine de la géophysique, ce qui nous a permis d'explorer l'hétérogénéité des ressources et l'organisation multi-échelle de communautés. Pour rappel, lorsque CloudFIT a été développé dans le cadre du projet PER-MARE [3] et fut utilisé notamment pour des *benchmark* de type Map Reduce, comme démontré en Fig. 4. Bien que ces *benchmarks* ont permis de valider le *middleware*, le cadre synthétique des tests ne se prêtait pas à la composition d'une application complexe.

L'occasion d'utiliser CloudFIT pour la création d'une application multi-échelle est issue d'un projet de collaboration avec le Laboratoire LACESM de l'Universidade Federal de Santa Maria (UFSM) au Brésil. Ce laboratoire de géophysique et météorologie s'est spécialisé dans l'étude de la dynamique et des effets du trou d'ozone antarctique. Le projet de collaboration en question est lié à la détection d'Événements Secondaires de la couche d'Ozone Antarctique (*Ozone Secondary Events* - OSE) [23].

Les OSE sont caractérisés par une réduction de la colonne totale d'Ozone dans les latitudes moyennes en raison d'un apport de masses d'air pauvres en Ozone qui se détachent du vortex polaire, comme illustré en Fig. 6. Contrairement aux mouvements d'expansion et contraction des frontières du vortex polaire antarctique (le "trou d'ozone"), les OSE sont moins connus et peuvent se déclencher à tout moment selon les courants atmosphériques prédominants. Pour cette raison, la détection et le suivi de tels événements sont né-

3. http://cosy.univ-reims.fr/PER-MARE

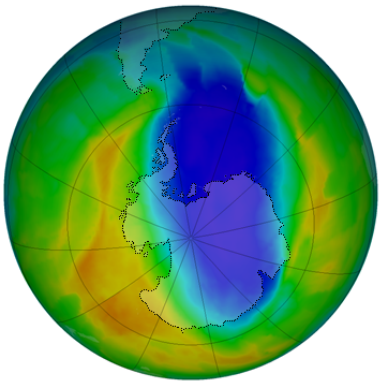

Figure 6 : Couche d'Ozone pour le 18 Oct 2013[14]

cessaires pour alerter la population et les autorités locales des risques liés à l'augmentation de l'irradiation ultra-violet, notamment celle de type UV-C, la plus nocive.

Pour la détection des OSE, différentes sources d'information peuvent être intégrées et explorées : des données satellite, des mesures d'équipements au sol ou portés par des ballons atmosphériques, etc. Ces mesures ont un profil proche de celles issues d'autres dispositifs IoT : une quantité de données relativement faible par unité (quelques ko pour des équipements au sol ou des balayages satellite) mais dont le volume cumulé au fil des années et le traitement nécessaire peuvent s'avérer important. Il faut aussi remarquer que les différentes étapes de la détection des OSE requièrent des efforts de calcul variés : le stockage des données et l'analyse ponctuel des phénomènes ne sont pas très gourmands en ressources, mais l'analyse d'une zone plus élargie ou la prédiction des événements nécessitent bien plus de ressources de calcul.

Afin d'accommoder les différentes étapes et mieux découper les besoins de cette application, nous avons structuré l'application sous la forme de cinq tâches principales, toutes exécutées sur CloudFIT :

— Stockage et pré-traitement des entrées
— Filtrage et agrégation
— Analyse des séries temporelles
— Détection d'événements
— *Prédiction des événements* (pas encore implémentée)

Chaque tâche peut être parallélisée et leur exécution coordonnée selon un DAG, comme illustré en Fig. 7. L'expression des calculs en tant que des tâches multi-échelles permet à chaque étape d'utiliser les ressources les plus adaptés, selon les capacités des dispositifs et leur position vis-à-vis des sources de données. Ainsi, par exemple, des dispositifs périphériques organisés dans une communauté C1 (cf. Fig. 7) peuvent être utilisés pour pré-traiter les données des spectrophotomètres Dobson et Brewer et des capteurs satellite OMI afin de les stocker dans la DHT. Cette étape démande peu de ressources si exécutée régulièrement (par exemple, avec l'inclusion des mesures journalières dans la DHT).

Grâce aux données entrées dans la DHT, il est possible de déclencher la détection des OSE sur le nouvel ensemble de données ou, dans le cas d'une analyse plus poussée, lancer une procédure d'analyse historique visant la détection de patrons connus utiles à une prédiction des événements. Cette étape, qui inclut le filtrage et l'analyse des séries temporelles, nécessite des ressources de calcul plus importants, surtout lorsque la zone d'études est plus large ou s'il faut étudier les événements sur une longue période de temps. Ainsi, une communauté C2 composée de machines un peu plus puissantes peut se charger de ces tâches (cf. la description des performances plus bas).

La Fig. 7 inclut aussi deux autres communautés C3 et C4, utilisées pour l'exécution de la cinquième étape liée à la prédiction des événements futurs. Cette étape, pas encore implémentée, requiert à la fois le traitement d'autres données atmosphériques telles que les courants stratosphériques et en haute troposphère, l'identification de patrons de corrélations entre les vents et les OSE et finalement la prédiction des événements. L'identification de corrélations est une tâche lourde, nécessitant l'analyse de grandes masses de données historiques et du *machine-learning*, et ainsi la communauté C3 nécessitera bien plus de ressources que les communautés précédentes. Toutefois, on estime que la communauté C3 ne sera appelée qu'occasionnellement car, après une première exécution, on pourra obtenir des patrons utilisables pour les prédictions : une nouvelle analyse des corrélations ne sera nécessaire que lorsque le volume de nouvelles données s'apprête à l'amélioration des modèles. Finalement, la communauté C4 se chargera de la prédiction des événements, on estime que les besoin de ressources seront similaires à ceux de la communauté C2.

Un exemple du résultat obtenu peut être vu en Fig. 8, représentant la progression du front OSE autour du 20 Octobre 2013, un OSE bien connu ayant servi à la validation de notre application. Nous pouvons observer la progression des masses d'air pauvre en Ozone sur des zones avec une population importante mais aussi la dissipation de celles-ci, ce qui peut prendre plusieurs jours. Les résultats positifs obtenus nous encouragent à développer ce système de surveillance et d'alerte, et à poursuivre l'analyse de la corrélation entre les OSE et les courants atmosphériques afin de mettre en place un système de prédiction d'OSE.

En ce qui concerne la performance de cette application, nous avons exécuté le scénario de la Fig. 8 sur différents équipements, allant d'un Raspberry Pi 2 à un serveur Intel Xeon 2x12 cœurs. Individuellement, Le Raspberry Pi nécessite plus de 30 min pour exécuter les étapes de filtrage, agrégation, génération des séries temporelles et la détection, alors que le serveur Xeon n'a besoin que de 7 minutes. Dans le cas d'une exécution en mode cluster pervasif (communauté C2) entre un Raspberry Pi, un Macbook Air (Intel i7-4650U, 2 cœurs) et un Lenovo U110 (Intel Core2 Duo), le temps moyen est de l'ordre de 10 minutes. Il faut toutefois remarquer que cette exécution ne concernait qu'une zone géographique limitée au sud de l'Amérique du Sud (couvrant une zone de $30° \times 55°$ ou environ 2400 points) alors qu'une couverture globale requiert l'analyse de presque 65000 points. La possibilité d'ajouter des nœuds à CloudFIT et de deployer les tâches en communautés confère l'élasticité nécessaire au traitement de tâches de calcul avec des besoins différents. De plus, l'organisation multi-échelle des nœuds permet l'utilisation de machines "bas de gamme" pour les opérations initiales (stockage et prétraitement des données), pour qu'ensuite d'autres communautés de machines plus puissantes puissent prendre le relais du calcul, une approche cohérente avec le *edge-computing*.

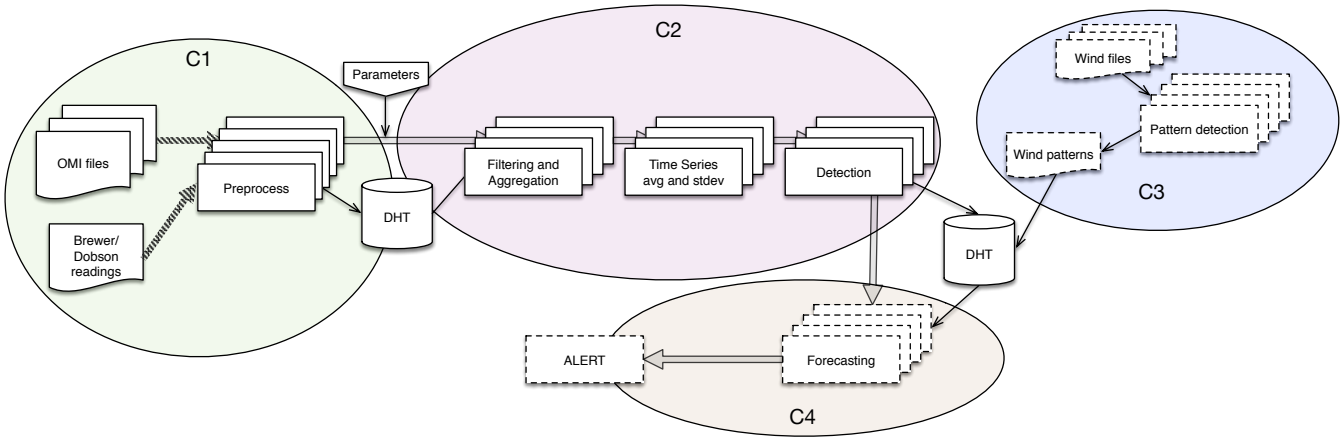

Figure 7 : DAG des tâches dans le framework de détection des OSE

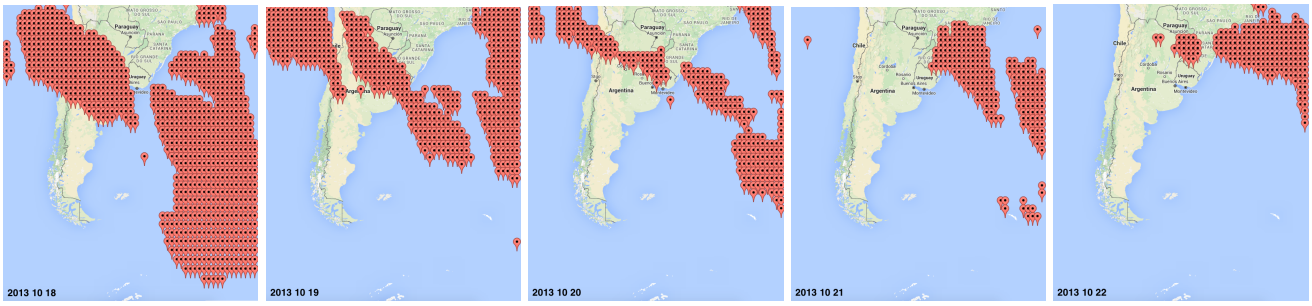

Figure 8 : Évolution des OSE entre le 18 et le 22 Octobre 2013

5. CONCLUSIONS ET TRAVAUX FUTURS

L'Internet des Objets est une tendance IT qui s'est rapidement développée et a ouvert de nouvelles opportunités à la fois de recherche et commerciales. Elle apporte également des défis intéressants liés à l'évolutivité et la dynamicité de l'environnement. Dans cet article, nous considérons les exigences IoT menant à une organisation multi-échelle des ressources sur les environnements pervasifs. Nous discutons des solutions architecturales et pratiques afin de mettre en œuvre une plate-forme de calcul pervasive multi-échelle adaptée au stockage et traitement de données issues des dispositifs IoT. En outre, nous montrons comment le middleware CloudFIT peut être adapté pour interconnecter les dispositifs, les regrouper et les coordonner.

Nos travaux futurs incluent la poursuite de l'implémentation des solutions identifiées dans ce texte. Bien que nous travaillions actuellement sur des stratégies pour une meilleure localisation des données, celles-ci devront obligatoirement être associées à des politiques d'ordonnancement sensibles au contexte, de manière à mieux adapter le traitement des tâches aux ressources disponibles dans les environnements pervasifs multi-échelles.

6. REFERENCES

[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the 1st MCC Workshop on Mobile Cloud Computing*, MCC '12, pages 13–16, New York, NY, USA, 2012. ACM.

[2] G. W. Cassales, A. S. Charao, M. Kirsch-Pinheiro, C. Souveyet, and L. A. Steffenel. Context-aware scheduling for apache hadoop over pervasive environments. *Procedia Computer Science*, 52 :202 – 209, 2015. The 6th International Conference on Ambient Systems, Networks and Technologies (ANT-2015).

[3] G. W. Cassales, A. Schwertner Charão, M. Kirsch-Pinheiro, C. Souveyet, and L.-A. Steffenel. Improving the performance of apache hadoop on pervasive environments through context-aware scheduling. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–13, 2016.

[4] Cisco Research Center Requests for Proposals (RFPs). Fog computing, ecosystem, architecture and applications. http://www.cisco.com/web/about/ac50/ac207/crc_new/university/RFP/rfp13078.html.

[5] S. Dey, A. Mukherjee, H. S. Paul, and A. Pal. Challenges of using edge devices in iot computation grids. In *Int. Conf. on Parallel and Distributed Systems*, pages 564–569, 2013.

[6] ETSI. Mobile-edge computing - introductory technical white paper. https://portal.etsi.org/Portals/0/TBpages/MEC/Docs/Mobile-edge_Computing_-_Introductory_Technical_White_Paper_V1%2018-09-14.pdf.

[7] M. Fazio, A. Celesti, A. Puliafito, and M. Villari. Big data storage in the cloud for smart environment monitoring. *Procedia Computer Science*, 52 :500 – 506,

2015. The 6th International Conference on Ambient Systems, Networks and Technologies (ANT-2015).

[8] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere. Edge-centric computing : Vision and challenges. *SIGCOMM Comput. Commun. Rev.*, 45(5) :37–42, Sept. 2015.

[9] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. Internet of things (iot) : A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7) :1645 – 1660, 2013.

[10] C. Johnen and F. Mekhaldi. Self-stabilization versus robust self-stabilization for clustering in ad-hoc network. In *Proceedings of the 17th International Conference on Parallel Processing - Volume Part I*, Euro-Par'11, pages 117–129. Springer, 2011.

[11] M. Jones. Internet of things : Shifting from proprietary to standard. http://www.valuewalk.com/2014/07/internet-of-things-iot/.

[12] L. Lim and D. Conan. Distributed event-based system with multiscoping for multiscalability. In *Proceedings of the 9th Workshop on Middleware for Next Generation Internet Computing*, MW4NG '14, pages 3 :1–3 :6, New York, NY, USA, 2014. ACM.

[13] D. Miorandi, S. Sicari, F. D. Pellegrini, and I. Chlamtac. Internet of things : Vision, applications and research challenges. *Ad Hoc Networks*, 10(7) :1497 – 1516, 2012.

[14] NASA. Nasa ozone hole watch web site. http ://ozonewatch.gsfc.nasa.gov/.

[15] M. Parashar and J.-M. Pierson. Pervasive grids : Challenges and opportunities. In K.-C. Li, C.-H. Hsu, L. T. Yang, J. Dongarra, and H. Zima, editors, *H. of Research on Scalable Computing Technologies*, pages 14–30. IGI Global, 2010.

[16] K. Paridel, E. Bainomugisha, Y. Vanrompay, Y. Berbers, and W. D. Meuter. Middleware for the internet of things, design goals and challenges. *ECEASST*, 28, 2010.

[17] A. Ramakrishnan, D. Preuveneers, and Y. Berbers. Enabling self-learning in dynamic and open IoT environments. In E. Shakshuki and A. Yasar, editors, *The 5th International Conference on Ambient Systems, Networks and Technologies (ANT-2014)*, volume 32 of *Procedia Computer Science*, pages 207–214. Elsevier, June 2014.

[18] S. Rottenberg, S. Leriche, C. Lecocq, and C. Taconet. Vers une définition d'un système réparti multi-échelle. In *UBIMOB'12 - 8èmes Journées Francophones Mobilité et Ubiquité*, pages 178–183, 2012.

[19] S. Rottenberg, S. Leriche, C. Taconet, C. Lecocq, and T. Desprats. Musca : A multiscale characterization framework for complex distributed systems. In *Proceedings of the 2014 Federated Conference on Computer Science and Information Systems*, pages 1657–1665, 2014.

[20] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, 8(4) :1423, Dec. 2009.

[21] M. Serrano, D. Le-Phuoc, M. Zaremba, A. Galis, S. Bhiri, and M. Hauswirth. Resource optimisation in iot cloud systems by using matchmaking and self-management principles. In *The Future Internet*, volume 7858 of *LNCS*, pages 127–140. Springer, 2013.

[22] L. A. Steffenel and M. Kirsch-Pinheiro. When the cloud goes pervasive : approaches for IoT PaaS on a mobiquitous world. In *EAI International Conference on Cloud, Networking for IoT systems (CN4IoT 2015)*, Rome, Italy, Oct. 2015.

[23] L. A. Steffenel, M. Kirsch-Pinheiro, D. Kirsch-Pinheiro, and L. V. Peres. Using a pervasive computing environment to identify secondary effects of the antarctic ozone hole. In *2nd Workshop on Big Data and Data Mining Challenges on IoT and Pervasive (Big2DM)*, Madrid, Spain, May 2016. Procedia Computer Science, Elsevier.

[24] O. Vermesan, P. Friess, P. Guillemin, R. Giaffreda, H. Grindvoll, M. Eisenhauer, M. Serrano, K. Moessner, M. Spirito, L.-C. Blystad, and E. Z. Tragos. Internet of things beyond the hype : Research, innovation and deployment. In *Internet of Things - From Research and Innovation to Market Deployment.* River Publishers, 2014.

[25] W. Wang, M. Barnard, and L. Ying. Decentralized scheduling with data locality for data-parallel computation on peer-to-peer networks. In *Proceedings of Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Monticello, IL, USA, 2015.

[26] D. Wu, Y. Tian, and K.-W. Ng. 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, pages 1–8. Springer, 2005.