ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE EN CONCENTRATION GÉNIE LOGICIEL
M. Sc. A.

PAR
Abderrahmane EL BARDAI

VIRTUALISATION D'UNE PLATEFORME DE GESTION DE CONTEXTE

MONTRÉAL, LE 20 OCTOBRE 2015

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

Mme Nadjia Kara, directrice de mémoire
Département de génie logiciel et des TI à l'École de technologie supérieure

Mme May El Barachi, codirectrice de mémoire
Professeure assistante à Collège d'innovation technologique, Université de Zayed

M. Sègla Kpodjedo, président du jury
Département de génie logiciel et des TI à l'École de technologie supérieure

M. Abdelouahed Gherbi, membre du jury
Département de génie logiciel et des TI à l'École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 14 SEPTEMBRE 2015

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Mes premiers remerciements vont d'abord à madame Nadjia Kara, professeure au département de génie logiciel à l'école technologie supérieure et directeur du mémoire. Qui m'a chaleureusement accueillie dans son laboratoire de recherche.

Je remercie aussi madame May El Barachi, codirectrice de mémoire professeure à l'Université Zayed d'Abu Dahbi pour son aide précieuse, de l'attention et de soutien qu'elle a portés à mon travail.

Je dédie ce mémoire de fin d'études à mon très cher père EL BARDAI DRISS et ma très chère mère REGRAGUI NAJIBA en témoignage de ma reconnaissance envers le soutien, les sacrifies et les efforts qu'ils ont faits pour mon éducation ainsi que ma formation.

Que ce travail soit le témoignage de notre respect et de notre gratitude pour la bienveillance avec laquelle vous m'avez toujours entourée.

VIRTUALISATION D'UNE PLATEFORME DE GESTION DE CONTEXTE

Abderrahmane EL BARDAI

RÉSUMÉ

Depuis son apparition, le Cloud Computing a motivé différents acteurs du secteur des technologies de l'information à créer de nouveaux services virtuels. Le Cloud Computing apporte beaucoup d'avantages comme la personnalisation, le paiement au besoin ou encore l'effet d'avoir une élasticité infinie.

Les systèmes CA sont des systèmes intelligents qui peuvent changer de comportement dépendamment du contexte dans lequel ils se trouvent. Ce type de système est le genre de système que l'utilisateur final préfère, car il anticipe ses besoins et s'y adapte. Pour avoir, un bon système CA, il faut avoir une bonne base de source d'informations. Ces informations de base permettent à ces systèmes d'être plus précis pour identifier leur contexte actuel. Les réseaux de capteurs sans fil sont une riche source d'information, car ils offrent une information diversifiée et en grand nombre.

Notre projet consiste à virtualiser les systèmes CA en utilisant les réseaux de capteurs sans fil. Notre première contribution consiste à décomposer les systèmes CA en blocs fonctionnels. La deuxième contribution est la proposition d'une architecture de gestion de contexte qui permet la composition et le déploiement de services CA dans le nuage. Cette architecture est validée à travers le prototypage et les tests. Notre dernière contribution est la proposition d'une méthodologie de dimensionnement de ces blocs fonctionnels

Mots-clés : Cloud Computing, réseaux de capteurs sans fil, Sensibilité au contexte.

VIRTUALISATION OF A CONTEXT MANAGEMENT PLATFORM

Abderrahmane EL BARDAI

ABSTRACT

Since its appearance, Cloud Computing has motivated the different IT actors to create novel virtual services. Cloud Computing paradigm provide great advantages such as personalisation of services, pay-only when needed and seemingly infinite system scalability.

Context aware systems are smart systems that can change their behaviour depending on the existing context. This type of system is the kind of systems that the end user wants to interact with because it anticipates the needs of the end user and adapts to them. In order to have a good context aware system, this system needs to have a good source of information. This raw information allows context aware systems to be more precise in identifying the current context. Wireless sensor networks provide a rich amount of raw information and thus can be integrated with context aware systems.

Our project tries to virtualise context aware systems using the information coming from wireless sensor networks. Our first contribution is that we portioned context aware systems to functional blocs named substrates. The second contribution is the proposition of an architecture of a context management platform that allows the composition and the deployment of context aware systems in the cloud. This architecture was validated via prototyping and testing. Our last contribution is that we presented a methodology that helps to dimension the substrates.

Keywords : Context awareness, Wireless Sensor Networks, Cloud Computing.

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

BDD	Base de données
CA	Context aware
CC	Nuage Computing
CPU	Central Processing Unit
IAAS	Infrastructure As a Service
I/O	Input/Output
PAAS	Platform As a Service
RCSF	Réseau de capteur sans fils
SAAS	Software As a Service
SAC	CA
SGBD	Système de gestion de base de données

INTRODUCTION

Depuis leur apparition, les réseaux de communication sans fil ont connu un large succès. Grâce à leurs nombreux avantages, ces réseaux ont pu s'imposer comme acteurs incontournables dans les architectures réseau actuelles. Les réseaux sans fil offrent en effet des propriétés intéressantes, qui peuvent être résumées en trois points : la facilité du déploiement, l'omniprésence de l'information et le coût réduit d'installation. Un réseau de capteurs sans fil est un réseau sans fil composé de nœuds capteurs distribués spatialement dans un environnement donné. Ces capteurs surveillent différents types de données environnementales comme la température et l'humidité ou physiques comme la pression sanguine et la température du corps humain, etc. Par ailleurs, la collecte des données n'est pas en elle-même une finalité, mais juste un moyen pour d'autres systèmes qui traitent les données.

Après l'explosion de la bulle internet, l'informatique occupe une partie importante de notre quotidien. L'intégration des circuits électroniques dans les véhicules, l'omniprésence de la téléphonie mobile intelligente, l'arrivée sur le marché des ordinateurs portables ultra-performants donnent une idée de ce qui peut nous attendre en informatique dans les années à venir : de plus en plus de mobilité, de communication entre machines qui ressemble de plus en plus au langage humain et des utilisateurs s'attendant à ce que leurs appareils répondent à des attentes motivées aussi par leur environnement. Les systèmes CA sont là pour répondre à ces besoins. Puisque ces systèmes tirent leur information d'une source de contexte, l'utilisation des réseaux de capteurs comme source de contexte devient un volet de recherche intéressant. Cependant, avec tant d'informations fournies par les réseaux de capteurs, l'intégration de ces réseaux dans d'autres systèmes peut créer plusieurs défis. Les différents protocoles de communications utilisés et les différentes représentations des données capturées créent un problème d'hétérogénéité. La puissance de traitement nécessaire est accrue dû à la quantité des données et le stockage des données devient difficile à cause des différents formats de données et à cause de la vitesse de stockage/extraction des données.

Tous ces facteurs ont eu pour effet de freiner le déploiement et l'évolution commerciale de ce type de systèmes.

Problématique

Les systèmes CA sont des systèmes qui réagissent au contexte dans lequel ils se trouvent. Ce contexte est la combinaison d'une ou plusieurs données simples qui sont à la disposition de ces systèmes. Les données fournies par les réseaux capteurs sont de bonnes données à utiliser pour construire le contexte. Il serait donc intéressant de pouvoir intégrer les réseaux sans fil dans les systèmes CA. Ces systèmes sont monolithiques se présentant comme un bloc logiciel devant le client. Le fournisseur de ce type de systèmes offre un service prêt à être utilisé cependant, il n'est pas flexible et personnalisable, car il se peut que le système fournisse plusieurs services. Par ailleurs, le client n'apprécie pas un service particulier dans ce système ou bien tout simplement n'est pas intéressé par un service donné parmi les autres ou bien le client apprécie les différents services, mais la performance du système en cas de forte charge est mauvaise. Par contre, le Cloud Computing offre cette flexibilité et cette élasticité. La principale question est comment rendre ces systèmes CA plus personnalisé, plus puissant et plus optimal en termes de ressources (matérielles et financières).

Avec l'apparition du paradigme du Cloud Computing, qui consiste à fournir différents types de services à travers internet, l'idée d'intégrer les systèmes CA dans le Nuage devient intéressante; car elle permettra de pallier aux défis et limitations rencontrées précédemment. Ainsi, en rendant les systèmes CA modulaires, on peut fournir chaque module comme service dans le nuage et créer des systèmes CA composés de ces blocs fonctionnels. En faisant comme suit le client peut obtenir un système personnalisé, car il le compose lui-même et ne va payer que sur les services dont il a besoin (propriété du nuage) et ne se souciera pas des problèmes reliés aux ressources matérielles (élasticité : propriété du nuage). Donc, la principale question qui se pose est comment rendre les applications CA modulaires tout en intégrant les réseaux de capteurs? Comment peut-on fournir au client une plateforme qui lui permet de créer des services CA à partir des blocs fonctionnels fournis aussi comme service?

Objectifs

Les principaux objectifs de ce projet sont :

- La réalisation d'une analyse détaillée sur les réseaux de capteurs sans fil, des systèmes CA et de leur intégration dans le Nuage;
- Définir les principaux blocs fonctionnels qui composent les systèmes CA et proposer une architecture dans le Nuage qui permet la composition et le déploiement dynamique de services CA;
- Développer une plateforme de gestion de contexte;
- Tester les performances de la plateforme de gestion de contexte et les différentes substrates;
- Proposer un algorithme de gestion des ressources physiques pour une plateforme de gestion de contexte.

Plan du mémoire

Ce document commence par une introduction générale du projet et de son contexte. Il contient aussi la problématique du mémoire et ses objectifs.

Le premier chapitre est la revue de la littérature. Nous commençons tout d'abord par présenter les réseaux des capteurs sans fil. Ensuite, nous décrirons en détail les systèmes CA. Puis, nous analyserons le paradigme du Nuage. Enfin, nous passons en revue quelques travaux de recherche reliés aux systèmes CA implémentés dans le nuage.

Dans le deuxième chapitre, nous proposons des scénarios concrets qui peuvent être implémentés et qui motivent notre travail de recherche. Nous décrirons ensuite les différentes entités qui constituent les systèmes CA. Nous poursuivons en présentant une vue globale de

l'architecture. Ensuite, nous détaillerons l'architecture logicielle de la plateforme gestion de contexte proposée et les différents diagrammes de séquences reliés.

Le troisième chapitre décrit un prototype d'une plateforme de gestion de contexte et présente l'analyse des performances d'une telle infrastructure. D'abord, nous décrirons les différents diagrammes de classe de notre plateforme. Ensuite, nous expliquerons, l'implémentation des différents substrates développés.

Dans le quatrième chapitre, nous présentons l'environnement de test, les résultats des tests effectués et l'analyse de ces résultats. Puis, nous présenterons un algorithme de gestion de ressources physiques pour la plateforme de gestion de contexte.

Enfin, nous terminons avec une conclusion générale sur notre travail de recherche et présenterons quelques travaux futurs.

CHAPITRE 1

REVUE DE LA LITTÉRATURE

1.1 Les réseaux de capteurs

Les réseaux de capteurs sans fil (RCSF) sont composés d'un large nombre de nœuds capteurs déployés d'une façon dense dans un environnement cible.

1.1.1 Architecture des RCSFs

L'architecture classique des RCSFs est illustrée dans la figure 1-1. Les nœuds capteurs sont dispersés dans le domaine dans lequel la capture est souhaitée. Les nœuds capteurs possèdent une portée de transmission faible et communiquent avec les nœuds adjacents pour acheminer les captures vers la passerelle. La passerelle est connectée à un médium longue distance et sert de lien entre les nœuds capteurs et les clients. Le nœud capteur est l'entité de base des RCSFs et sont déployés en grand nombre. Un nœud capteur comporte au moins ces quatre composants :

- Unité de détection : cette unité peut comporter plusieurs capteurs pour détecter différents types de phénomènes et d'un convertisseur qui convertiront les différents signaux analogiques en signaux numériques pour les envoyer à l'unité de traitement;
- Unité de traitement : cette unité gère en collaboration avec les autres unités de traitement la mission de mener à bien les tâches de détection. Cette unité contient en général un composant qui permet le stockage des données;
- Unité de transmission : ce composant est responsable de connecter le nœud capteur au réseau;
- Unité d'alimentation : ce composant assure les besoins énergétiques des différentes unités du nœud capteur.

D'autres unités facultatives peuvent aussi être présentes dans un nœud capteur par exemple un générateur de courant qui transforme une énergie non électrique en courant utilisable par l'unité d'alimentation.

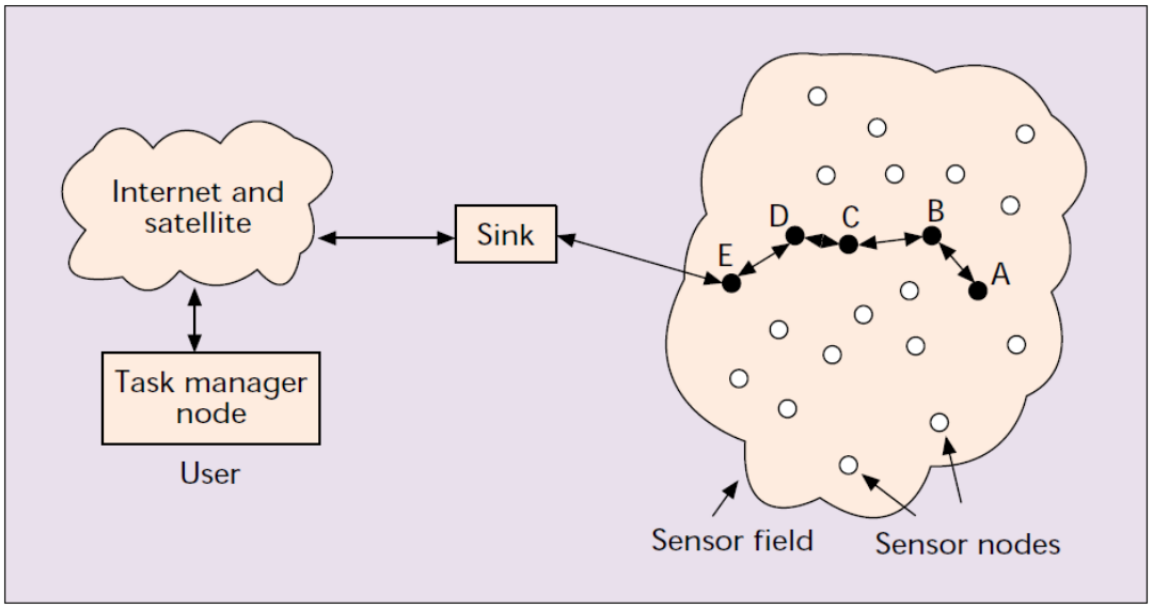

Figure 1-1 Architecture d'un RCSF.
Tirée de Jason Lester Hill (2003)

Pour concevoir et déployer un RCSF, plusieurs facteurs doivent être pris en considération. Dans cette partie, nous essayerons de citer les plus importants :

- La fiabilité : c'est la capacité du réseau à réussir la tache de détection sans interruption malgré la défaillance de certains nœuds capteurs au sein du réseau. On désigne par défaillance toute mesure ou capture considérée fausse et non pas seulement l'arrêt du fonctionnement du nœud capteur;
- L'évolutivité : c'est la capacité du système à subvenir de manière dynamique aux besoins évolutifs en termes de ressources;
- Les coûts de production : puisque les RCSFs exigent par leur nature un large nombre de nœuds capteurs alors le prix unitaire de ces nœuds doit être justifiable. Un coût élevé aura systématiquement comme résultat l'abandon d'une solution RCSF;

- La consommation d'énergie : les nœuds de capteurs possèdent des ressources limitées en termes d'énergie. Certains scénarios d'applications peuvent trop utiliser les ressources du nœud capteur et se montrent très gourmant vis-à-vis la consommation d'énergie au point que la batterie ne puisse plus maintenir les besoins de l'application;
- L'environnement de déploiement : les RCSFs ont un large domaine d'applicabilité ce qui confère une grande variété d'environnement de déploiements. Le choix de nœuds capteurs adaptés à l'environnement de déploiement est donc primordial;
- Les médias de transmission : les différents médiums utilisés doivent être fiables dans le sens où la probabilité d'acheminer des informations fausses à partir de données correctes est faible, résistants aux différents types d'environnements (les médiums hertziens sont très sensibles aux détériorations climatiques) et efficaces dans la consommation d'énergie.

1.2 Les systèmes CA

Les systèmes CA (SAC) sont des systèmes qui possèdent la capacité de changer de comportement dépendamment de la situation qui entoure l'environnement d'exécution. Ces systèmes donnent naissance à un nouveau type d'applications : plus intelligentes, plus intuitives et tendent à comprendre les intentions de l'utilisateur et agissent en fonction de ces intentions. La situation ou le contexte n'est pas limité seulement à l'utilisateur final, son profil et ses préférences, mais aussi à d'autres éléments par exemple le lieu dans lequel il se trouve, un facteur de temps, l'état du matériel, etc.

1.2.1 Exemples d'application

Dans cette partie, nous essayerons de donner deux scénarios qui illustrent les avantages des systèmes CA.

Dans le domaine de la santé, on peut considérer une application SAC qui permet de surveiller à distance l'état des patients. Les capteurs envoient régulièrement des informations sur l'état du patient par exemple la fréquence cardiaque, la vitesse de déplacement, la localisation. L'application se chargera en temps normal à l'aide d'un pilulier intelligent de rappeler la personne responsable du patient par SMS ou autre type de média (e-mail, alarme…) des médicaments qu'il doit prendre. En cas de comportement jugé anormal comme par exemple une crise cardiaque, l'application se chargera de contacter l'entité la plus proche de lui qui peut lui venir en aide.

Une autre application intéressante est dans le domaine social. Un assistant de vie intelligent inspiré de l'application lancée par Nike nommée Nike+ training App. Les chaussures sportives de la nouvelle génération comportent des capteurs et fournissent différents types de données : saut vertical, la vitesse, le nombre de sauts effectués lors d'un entraînement, la durée en l'air durant les sauts, la fréquence cardiaque et le taux de glucose dans le sang. Toutes ces informations sont envoyées à un mobile intelligent dans lequel l'utilisateur peut visionner ses performances, les partager dans un réseau et rivaliser et se comparer à d'autres utilisateurs. L'application peut aussi fournir des indications et des conseils pour améliorer les performances et elle peut aussi proposer des programmes d'entraînements pour atteindre des objectifs fixés.

1.2.2 Fonctionnalités

Les systèmes SAC sont des systèmes complexes et leur compréhension est essentielle pour le bon déroulement du projet. Dans ce qui suit, nous essayerons d'éclaircir les principales fonctionnalités des systèmes SaC.

1.2.2.1 Acquisition du contexte

L'acquisition du contexte est une fonctionnalité initiale et de base. Dans cette phase nous collectons de l'information contextuelle depuis les différentes sources de contexte. Nous définissons le contexte par « toute information qui peut être utilisée pour caractériser la

situation d'une entité. Une entité peut être une personne, une place, ou un objet qui s'avère pertinent à l'interaction entre l'utilisateur et l'application » (Devajaru, Hoh et Hartle, 2007). L'information qui constitue le contexte peut provenir de deux types de sources :

- Sources physiques : ce sont les mesures qui proviennent des différents nœuds capteurs des RCSFs. Ils représentent des mesures associées à des phénomènes physiques par exemple la température, la pression, l'humidité…
- Sources logiques : ce sont des mesures qui représentent des états logiques d'entités contextuelles. Par exemple : la bande passante du réseau, la fiabilité des mesures des nœuds capteurs…

Il existe trois méthodes d'acquisition de contexte : soit par accès directs aux ressources physiques. Dans ce cas les capteurs et le module d'acquisition se trouvent sur le même appareil. Cette méthode est pratique, mais le contexte collecté est limité et n'est pas adapté aux systèmes distribués. Soit à travers un intergiciel ou bien à travers une base de données.
L'acquisition du contexte est une grande fonctionnalité et peut être composée de plusieurs sous-fonctionnalités :

- La détection des données : c'est la capacité du module à recevoir les mesures des nœuds capteurs et aussi envoyer des commandes vers ces nœuds;
- Le balayage : cette fonctionnalité permet la découverte dynamique des ressources du réseau et permet aussi d'identifier l'ensemble de nœuds capteurs optimaux pour effectuer une tâche de détection;
- L'agrégation : consiste à fusionner un grand nombre de mesures pour en tirer des valeurs significatives par exemple une moyenne de valeurs ou une variance. Cette opération a pour effet d'alléger l'effort de traitement et de stockage. Et donc, optimise l'utilisation des ressources;
- L'adaptation : on désigne par l'adaptation l'obtention de l'information que lorsque l'on a vraiment besoin de celle-ci.

À cause de l'aspect propriétaire et la diversité des technologies des RCSFs, les développeurs de systèmes SaC développent leurs propres modules d'acquisition de contexte

dépendamment de la technologie implémentée dans le RCSF. Dans ce sens, plusieurs travaux de recherches ont été réalisés proposant des solutions qui effectuent l'acquisition du contexte indépendamment de la technologie utilisée dans les RCSFs. On cite par exemple les travaux de Devajaru, Hoh et Hartle (2007) et de Gigan et Atkinson (2007). La solution proposée par Gigan et Atkinson est particulièrement intéressante, car elle introduit une couche d'abstraction qui cache les détails et les spécifications matérielles des nœuds capteurs. La figure 1-2 illustre l'architecture de cette solution décomposée en trois niveaux.

La couche agent supérieure est responsable de communiquer avec les clients de la solution. Elle reçoit les paquets, les analyse puis fait appel aux méthodes appropriées de la couche inférieure et envoie les réponses de requêtes aux clients. La couche suivante est la couche de communication. Cette couche fournit des APIs et des fonctions génériques qui permettent le contrôle et la gestion des différents capteurs indépendamment de la technologie des RCSFs.

Cette couche a aussi pour rôle d'identifier le réseau et les capteurs cibles des requêtes des clients. La couche inférieure est responsable de traduire la requête générique envoyée par la couche de communication en requêtes natives compréhensibles par le RCSF correspondant. Cette couche est étroitement liée à la partie matérielle du RCSF et son implémentation varie suivant le protocole de communication et l'encodage des données du RCSF.

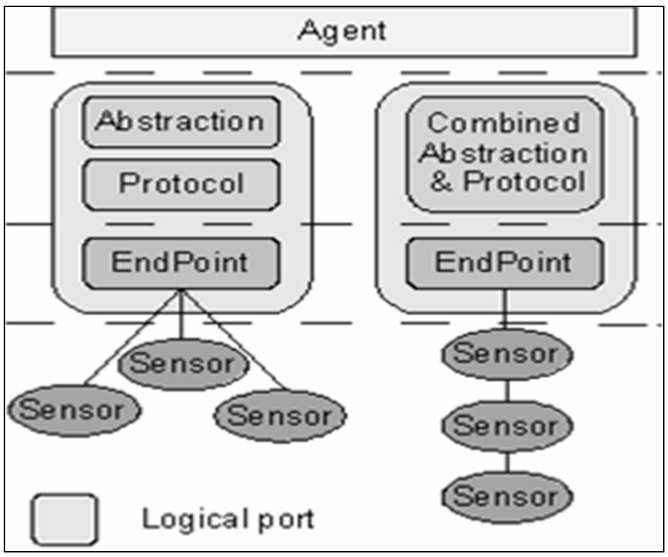

Figure 1-2 Couche d'abstraction de capteur
Tirée de Gigan et Atkinson (2007)

1.2.2.2 Modélisation du contexte

Après l'obtention du contexte, il est indispensable de formater et représenter ce contexte le plus efficacement possible. La modélisation joue un rôle important dans les systèmes SaC, car une bonne modélisation permettra d'améliorer l'efficacité des traitements, des raisonnements, du stockage et de l'échange des données contextuelles. Plusieurs sujets de recherche se sont intéressés à l'aspect modélisation. Nous citerons d'abord les exigences qui doivent être prises en considération lors de la modélisation du contexte puis présenterons les différentes approches de modélisation qui existent.

Exigences

- La rapidité : certains systèmes SaC sont temps réel et donc il est nécessaire dans ce cas que l'approche de modélisation permette la récupération des données requises le plus rapidement possible;

- La gestion des dépendances : le modèle doit prendre en considération les différentes relations qui existent entre les contextes. Un changement dans la valeur d'un contexte peut affecter un autre ou plusieurs contextes;
- Support du raisonnement : le modèle choisi doit donner l'opportunité au client d'effectuer plus de traitements sur le contexte collecté pour pouvoir l'exploiter au mieux;
- Formalisme : le formalisme permet de décrire les informations de manière précise et permet de supprimer les ambiguïtés qui peuvent être générées lors de l'échange des informations contextuelles entre les différentes parties prenantes;
- Interopérabilité : c'est la capacité du modèle à pouvoir communiquer et interagir avec d'autres systèmes pour échanger des données;
- Support de la qualité : le modèle doit prendre en considération, en dehors des valeurs de captures, des métriques de la qualité du contexte.

Approches

- Les modèles de type clé-valeur : ces modèles consistent à associer à chaque classe de contexte (la clé) les mesures acquises des capteurs (la valeur). Cette méthode de modélisation est simpliste, mais s'avère très performante et simple à implémenter. Elle est très pratique pour les systèmes simples et non complexes;
- Les modèles basés sur des profils XML;
- Les modèles graphiques qui sont des modèles basés sur des organigrammes. C'est-à-dire que l'on va essayer de représenter de manière graphique les données contextuelles. On note dans ce sens le CML (context modeling language);
- Les modèles basés sur la logique qui sont des modèles qui représentent le contexte sous forme de règles. Ce sont des modèles caractérisés par leur haut degré de formalité et leur capacité à effectuer du raisonnement;
- Les modèles basés sur les ontologies : Ce sont des modèles très formels et très descriptifs et offrent la possibilité de raisonner et échanger le contexte. Cependant,

ces modèles sont complexes et très couteux et consomment beaucoup des ressources de traitement;

- Les modèles hybrides : Ce sont des modèles qui incorporent au moins deux des approches citées avant. Le but de ce mélange est d'essayer de bénéficier des avantages qu'offrent les différentes approches de modélisation.

1.2.2.3 Autres fonctionnalités

Le raisonnement du contexte est une autre fonctionnalité intéressante et qui doit être prise en considération. Le raisonnement consiste à traiter les données et les rendre plus utiles. On distingue deux types de raisonnement. Un raisonnement déductif qui va permettre de générer et d'inférer de nouvelles connaissances à partir d'informations contextuelles. Le deuxième type raisonne sur l'incertitude du contexte et a pour finalité d'améliorer la qualité du contexte. La logique floue, la logique probabilistique, les réseaux bayésiens, les modèles cachés de Markov sont des approches adoptées pour ce type de raisonnement.

L'échange du contexte est une autre fonctionnalité importante. Elle représente la capacité du système à interagir avec d'autres systèmes SaC. L'échange du contexte est un moyen efficace pour augmenter la quantité des informations contextuelles. Un système SaC ouvert vers l'extérieur est plus puissant et plus efficace. Cependant, une bonne définition des acteurs, du contenu à échanger et des protocoles utilisés lors des échanges est cruciale pour éviter d'obtenir des incohérences et des effets inverses.

Le stockage des données contextuelles est important. La quantité et la qualité du contexte sont un actif précieux qu'il faut gérer intelligemment. Le stockage du contexte doit être optimal en termes d'efficacité et la manière avec laquelle s'effectuera le stockage dépendra étroitement de la modélisation du contexte. Le stockage des données devra donc permettre le stockage et la récupération efficace des données et la mise à jour des données pour éliminer tout contenu désuet. Pour certaines applications temps réel, il est nécessaire que le stockage des données permette la récupération rapide des données requises.

Certains systèmes SaC peuvent exiger des fonctionnalités spécifiques par exemple un support de la sécurité dans le sens où le système possède des données jugées critiques et qu'il veut les protéger. Dans ce cas le système doit assurer les trois propriétés de base de la sécurité : la confidentialité, l'intégrité et la disponibilité des données. Assurer une bonne qualité peut s'avérer nécessaire dans certains scénarios. La fonctionnalité de qualité devra alors être présente et couvrira la qualité du contexte et la qualité de service.

D'autres fonctionnalités qui ne sont pas spécifiques à la nature des RCSFs peuvent aussi prendre part d'un système SaC par exemple une fonctionnalité qui prédit de futurs contextes à partir de données historiques, une fonctionnalité qui permet la visualisation des données ou une fonctionnalité d'analyse des données.

1.3 Le Cloud computing

Le Cloud Computing (CC) est un modèle qui permet un accès pratique, omniprésent et sur demande à un ensemble de ressources informatiques configurables (réseaux, serveurs, stockage, applications et services). Ces ressources peuvent être rapidement provisionnées et libérées avec un effort minime de gestion et d'interaction avec les fournisseurs de services.
Le modèle du Nuage est composé de cinq caractéristiques essentielles, trois modèles de service, et quatre modèles de déploiement.

1.3.1 Les caractéristiques du Nuage

Les principales caractéristiques du Nuage sont :

- Libre-service à la demande : Un consommateur peut être provisionné par des capacités informatiques, telles que le temps de serveur et de stockage en réseau, au besoin automatiquement sans nécessiter une interaction humaine avec chaque fournisseur de services;

- L’accès à un réseau large : Les capacités informatiques sont disponibles sur le réseau et accessibles via mécanismes normalisés qui favorisent l’utilisation par les plateformes client hétérogènes (par exemple, téléphones mobiles, tablettes, ordinateurs portables et stations de travail);
- Mise en commun des ressources : Les ressources informatiques (ex. ressources de stockage, de traitement, de mémoire et de bande passante réseau) du fournisseur sont regroupées pour desservir plusieurs consommateurs en utilisant un modèle multilocataire, avec différentes ressources physiques et virtuelles dynamiquement assignée et réassignée selon la demande des consommateurs. Le client n’a généralement pas de contrôle ou de connaissances sur l’emplacement exact des ressources fournies, mais peut être en mesure de préciser l’emplacement à un niveau plus élevé d’abstraction (par exemple, pays, état, ou datacenter);
- Élasticité rapide : Les capacités peuvent être élastiquement provisionnées et libérées, dans certains cas automatiquement, dépendamment de la demande. Du point de vue du consommateur, les capacités disponibles pour provisionner souvent semblent être illimitées et peuvent être appropriées en tout moment;
- Service mesuré : Les systèmes du Nuage contrôlent automatiquement et optimisent l’utilisation des ressources en tirant parti de la capacité de mesure à un certain niveau d’abstraction approprié pour le type de service (par exemple, stockage, le traitement, la bande passante, et les comptes d’utilisateurs actifs). L’utilisation des ressources peut être surveillée, contrôlée et rapportée en assurant la transparence à la fois pour le fournisseur et consommateur du service utilisé.

1.3.2 Les services du Nuage

Le modèle du Nuage offre trois principaux services illustrés dans la figure 1-3.

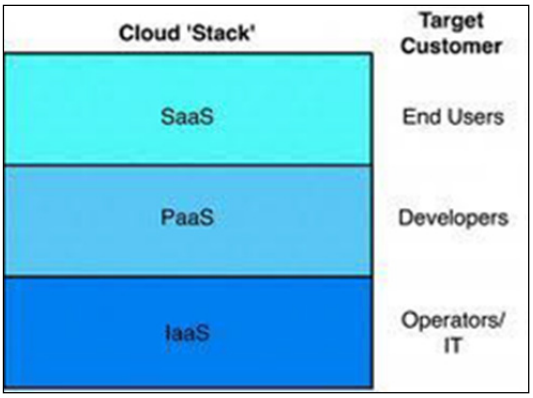

Figure 1-3 Architecture du Nuage.
Tirée de Randy Bias (2009)

- Software as a Service (SaaS). Le service prévu pour le consommateur est d'utiliser les applications du fournisseur exécutées sur une infrastructure de Cloud. Les applications sont accessibles à partir de divers dispositifs de clients à travers soit une interface client léger, comme un navigateur Web ou une interface de programme. Le consommateur ne gère ni contrôle l'infrastructure Cloud sous-jacente, y compris le réseau, les serveurs, les systèmes d'exploitation, le stockage, ou même les capacités d'application individuelles;
- Platform as a Service (PaaS). Le service prévu pour le consommateur est de déployer sur le nuage les infrastructures consommateur créées ou les applications créées à l'aide des langages de programmation, les bibliothèques, les services acquis, et des outils supportés par le fournisseur. Le consommateur n'a pas à gérer ou à contrôler l'infrastructure infonuagique sous-jacente, y compris le réseau, les serveurs, les systèmes d'exploitation, ou le stockage, mais il a le contrôle sur les applications déployées et, éventuellement, les paramètres de configuration de l'environnement d'application d'hébergement;
- Infrastructure as a Service (IaaS). Le service prévu pour le consommateur est le provisionnement des ressources de traitement, de stockage, de réseaux et autres ressources informatiques fondamentales où le consommateur est en mesure de déployer et d'exécuter un logiciel arbitraire, qui peut inclure les systèmes d'exploitation et les applications. Le consommateur ne gère ni contrôle le Nuage

sous-jacent, mais a le contrôle sur les systèmes d'exploitation, le stockage et les applications déployées; et le contrôle éventuellement limité de certaines composantes du réseau (par exemple, pare-feu).

1.3.3 Les modèles de déploiement du Nuage

Il existe quatre principaux modèles de déploiement de Clouds :

- Le nuage privé. L'infrastructure infonuagique est provisionnée pour une utilisation exclusive par une seule organisation comprenant plusieurs consommateurs (par exemple, les unités d'affaires). Elle peut être détenue, gérée, et exploitée par l'organisation, un tiers, ou une combinaison d'entre eux;
- Le nuage communautaire. L'infrastructure infonuagique est provisionnée pour une utilisation exclusive par une particulière communauté de consommateurs de différentes organisations qui ont des préoccupations communes (par exemple, une mission, des exigences de sécurité, des politiques et des considérations de conformité). Elle peut être détenue, gérée et exploitée par un ou plusieurs des organismes de la communauté, une tierce partie, ou une combinaison d'entre eux, et elle peut exister sur ou en dehors des locaux;
- Le nuage public. L'infrastructure infonuagique est provisionnée pour une utilisation ouverte par le grand public. Elle est détenue, gérée et exploitée par une entreprise, ou une organisation universitaire, gouvernementale ou une combinaison d'entre eux. Elle existe dans les locaux du fournisseur du Nuage;
- Le nuage hybride. L'infrastructure infonuagique est une composition de deux ou plusieurs nuages d'infrastructures distinctes (privé, communautaire ou public) qui restent des entités uniques, mais sont liées ensemble par une technologie normalisée ou propriétaire qui permet aux données et applications la portabilité.

1.4 Les plateformes de gestion de contexte

Plusieurs travaux ont été réalisés et qui avaient pour but de fournir des plateformes de gestion de contexte dans le nuage. L'article de Hyun Jung, Moon et Soo (2012) présente un Framework qui fournit des fonctionnalités primaires de gestion de contexte dans le nuage. En particulier, des fonctionnalités de raisonnement et de visualisation du contexte collecté. L'architecture est composée en deux parties : une partie qui sera installée au niveau du client qui se chargera de la collecte et la transmission des données contextuelles dans le Nuage et une partie au niveau du Nuage qui se chargera réceptionner les contextes, les stocker et enfin fournir les fonctionnalités aux clients à partir du stock de contexte. Le principal défaut de cette approche est qu'elle est monolithique. En d'autres termes, on ne va pas allouer à chaque requête du client une instance qui se chargera de traiter sa requête. C'est le serveur central qui se chargera de traiter toutes les requêtes des clients. Cette approche n'emploie pas la réutilisabilité des composants qui est un des grands avantages du Nuage. Par contraste à cette approche, Chihani, Bertin et Crespi (2012) proposent une architecture pour la gestion de contexte qui se focalise principalement sur le raisonnement du contexte qui le fournit comme service (reasoning as service). L'architecture se base sur le fait qu'il existe plusieurs fournisseurs qui peuvent fournir des moteurs de raisonnement et que la plateforme se chargera à partir de la requête de l'utilisateur de solliciter le fournisseur le plus approprié et créer une instance qui se chargera de raisonner sur le contexte collecté. Ce que l'on peut reprocher à cette plateforme est le fait que les travaux se sont focalisés sur la fonctionnalité du raisonnement du contexte et ont négligé les autres fonctionnalités qui sont elles aussi primordiales.

Les travaux de Hyung Jung et Soo Dong (2010) présentent un Framework qui permet l'adaptation dynamique des services. Le Framework capture le contexte et l'analyse pour déterminer l'action à effectuer puis réalise et adapte le service. Le Framework se base sur un « arbre d'adaptation » pour déterminer à partir du contexte collecte le type d'adaptation de service à réaliser.

Le travail de Babidi et Taleb (2011) propose une plateforme de gestion de contexte basée sur le courtier. Le courtier dans ce cas sert d'intermédiaire entre les consommateurs de contextes qui sont des Web services SaC et les fournisseurs de contextes. Le courtier a pour objectif de détacher les consommateurs des fournisseurs de contextes. Il permet aux fournisseurs de publier leur contexte et aux abonnés de s'abonner aux contextes désirés.

Belqasmi et al (2012) présente une plateforme de gestion d'application IVR dans le Nuage. La plateforme permet aux fournisseurs d'applications IVR la possibilité de créer et composer leurs services à partir de modules fournis appelés ressources physiques (substrates). La plateforme fournit des interfaces graphiques et des APIs qui permettent au client de créer et composer son service. La plateforme s'occupe de gérer la requête et instancier les différentes ressources physiques nécessaires pour réaliser la requête du client. Elle permet aussi de gérer, opérer et exécuter les différentes ressources physiques instanciées. L'avantage de cette solution est que l'on profite des avantages offerts par le Nuage.

1.5 Conclusion

Pour résumer, les réseaux de capteurs sont des réseaux vastes et offrent un grand nombre de données. Par ailleurs, les systèmes CA restent encore monolithiques et n'utilisent qu'une quantité limitée d'information. Malgré les efforts de recherche effectués, il n'existe aucun système CA distribué qui utilise la richesse des données des capteurs sans fil.

CHAPITRE 2

ARCHITECTURE PROPOSÉE

2.1 Scénarios motivants

Pour illustrer les services que nous pouvons offrir avec des systèmes CA en utilisant des capteurs, nous avons choisi dans notre projet le magasinage intelligent (Smart shopping). Longues files d'attente, produits en rupture de stock, location inconnue de certains produits, des rabais ratés sont des facteurs parmi d'autres qui rendent l'expérience de magasinage un peu énervante et plutôt un fardeau. Mais qu'en est-il d'une nouvelle méthode de magasinage? Une nouvelle méthode qui rendra l'expérience de magasinage plus intéressante et dans laquelle les clients prennent plus de plaisir. On présente dans ce qui suit un scénario de magasinage intelligent en utilisant les RCSF (réseaux de capteurs sans fil) et les systèmes CA.

Définition des différents acteurs (utilisateurs finaux)

Client : c'est le consommateur des produits. Il est là pour profiter au maximum de son expérience de magasinage. L'application du magasinage intelligent doit anticiper avec le plus de précision possible les actions de l'utilisateur. Les détails de la manière dont le client peut utiliser cette application sont expliqués après la définition des différents acteurs.

Employé du magasin : il peut interagir avec l'application et bénéficier de plusieurs services comme :

- Des informations à propos des produits, leurs localisations, les quantités de stock exposées aux clients, et les quantités de stock dans l'inventaire. L'application peut aider et assister les clients en recherche d'aide ou de conseil;
- Des notifications concernant des ruptures de stock ou des produits expirés;

- Des mises à jour d'informations reliées aux stocks des produits comme le prix ou la quantité disponible;
- Des offres pour des rabais du vendeur aux consommateurs.

Gestionnaire ou le superviseur du magasin : il peut utiliser l'application pour visualiser et analyser les données du magasin. Par exemple, l'application peut lui fournir des informations de son stock de provisions, des articles les plus populaires, des périodes du jour où le magasin vend le plus, etc. L'application fournit aux gestionnaires des informations utiles et des rétroactions par rapport à l'expérience de magasinage des clients. Ainsi cela permet aux gestionnaires de prendre des décisions efficaces et stratégiques par rapport à leur politique de réapprovisionnement de stock et les choix des produits à étaler.

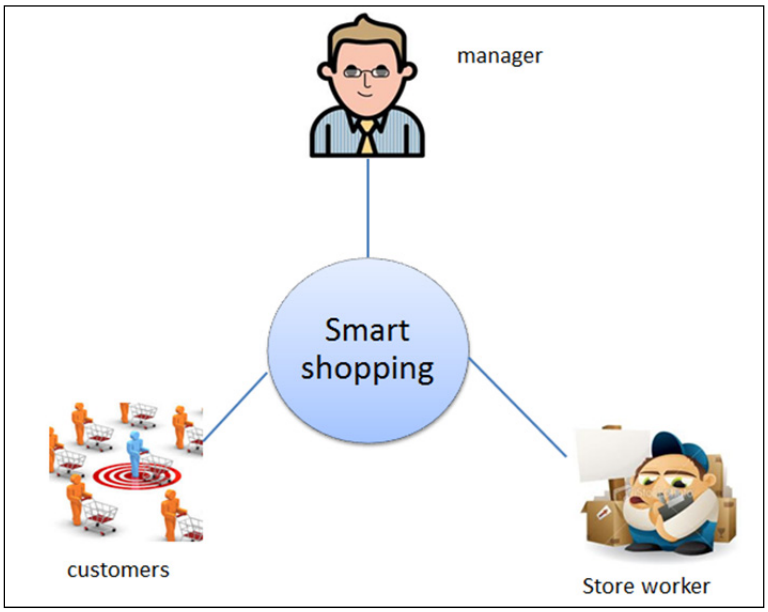

Figure 2-1 Les acteurs du magasinage intelligent

Scénarios typiques

Le comportement du consommateur varie en fonction de la nature du magasinage. On peut diviser ce comportement en deux catégories : magasinage actif et magasinage passif.

En mode passif, le système construit un profil du consommateur et envoie des notifications et des suggestions au client à propos d'articles, de rabais ou bien d'évènements qui pourraient l'intéresser.

En mode actif, le client montre un intérêt pour le magasinage en utilisant l'application intelligente ou bien en visitant directement le magasin, dans ce cas, le système se comportera dépendamment de la nature du magasinage. Nous avons sélectionné deux activités de magasinage et nous essayons de montrer quels sont les différents services que l'application peut fournir.

Magasinage de l'épicerie : Une famille modèle composée d'un père, une mère et deux enfants va partir au supermarché pour effectuer leur magasinage hebdomadaire habituel. Une fois sur place, l'application identifie les quatre personnes et les associe comme étant un groupe. Lorsque la mère a téléchargé gratuitement l'application, elle s'est inscrite et a configuré son profil et celui des membres de sa famille. Puisque la mère est le créateur du groupe, elle est désignée comme leader. Les enfants resteront à l'aire de jeu offerte par le magasin tandis que les parents vont s'occuper du magasinage. Avant leur arrivée, la mère avait déjà préparé une liste d'épicerie. L'application se charge de transférer automatiquement cette liste vers « le chariot intelligent ». Le chariot crée un plan du supermarché et montre la localisation des produits sélectionnés de la liste et le chemin optimal pour les atteindre. L'application peut aussi proposer les rabais du supermarché et les produits qui pourraient intéresser la famille en se basant sur le profil de chacun des membres. La mère peut effectuer son magasinage tout en ayant l'esprit tranquille puisque l'application offre la localisation exacte de chaque membre du groupe à tout moment à l'aide des capteurs des cellulaires des membres du groupe. Le père quant à lui, n'est pas en charge de la liste d'épicerie, mais il est plutôt intéressé par un complément alimentaire spécifique. Il demande alors à l'application de lui fournir plus d'informations à propos de l'article. Ne se contentant pas du produit, il demande en plus un régime basé sur ce complément alimentaire qui conviendrait à sa condition médicale et qui ne contient pas des aliments auxquels il est allergique, ce que l'application est en mesure de lui fournir en consultant son profil. Après un certain temps,

l'application envoie un message de notification qui informe la mère qu'ils ont atteint la limite des dépenses, car la famille a fixé un seuil de dépenses dans leurs préférences. À la caisse, le chariot affiche la somme des articles scannés automatiquement lors du magasinage et demande une confirmation de paiement. Puisqu'ils sont des clients fidèles, les membres de la famille bénéficient d'un rabais personnel. Le compte est bon; la transaction est effectuée. Entre temps, l'application envoie un message de notification aux enfants leur disant que le temps de jeu est terminé et qu'il est temps de rentrer à la maison.

Magasinage de vêtements : l'application peut fournir un catalogue contenant une sélection de vêtements qui convient aux préférences de l'utilisateur. Le catalogue peut être personnalisé en fonction des critères du client : prix, tendances de mode, rabais, marque, matière de fabrication, etc. Lorsque le client entre dans un magasin, il peut se connecter à l'application de son propre appareil mobile ou bien de l'appareil du magasin. L'application va fournir les fonctionnalités citées ci-dessus tout en l'adaptant aux spécifications de chaque magasin. Elle peut aussi fournir un « outil d'essai virtuel » qui permet aux clients d'essayer virtuellement les vêtements exposés dans le magasin. C'est une méthode simple, rapide et non encombrante pour essayer des vêtements. L'application permet aussi de montrer la localisation des articles sélectionnés. Elle peut aussi fournir un « outil de personnalisation de vêtements » qui permet aux clients de concevoir leurs propres vêtements basés sur des modèles ou bien de leur propre création. La commande est soumise et une réponse est envoyée au client contenant les détails de sa commande : faisabilité, prix, temps pour la production, estimation du temps de livraison, etc.

Magasinage de mobilier : le client peut interagir avec l'application et utiliser ces fonctionnalités :

- Conseiller de décor : l'application peut utiliser les préférences de l'utilisateur pour fournir des conseils sur le décor intérieur de la maison;
- Simulateur de maison : l'application peut créer une réplique virtuelle de la maison du client (en utilisant des capteurs installés à la maison). Le client peut alors naviguer à

travers les produits et simuler comment les différents meubles peuvent être disposés dans les pièces de la maison et voir en général à quoi ressemblera sa maison de l'intérieur. Ceci permet d'avoir une vue plus réaliste et plus claire des articles à acheter et permet d'éviter certains problèmes comme les dimensions des produits, le style et la compatibilité té avec le décor existant;

- Assistant d'assemblage : un guide interactif peut assister le client lors la phase d'assemblage du meuble acheté. Le client peut utiliser par exemple la caméra du téléphone au cours de son assemblage pour valider avec le guide les bonnes pièces à utiliser et l'allure du meuble durant son montage;
- Livraison personnalisée : le magasin peut utiliser les informations contextuelles du client (emploi du temps/agenda) pour programmer le temps de livraison optimal et convenable pour le client (ex. quand le client est à la maison).

2.2 Présentation des substrates reliés aux systèmes CA

Un substrate est un bloc logiciel qui remplit une fonctionnalité déterminée d'un système CA. Pour avoir un bon système CA, il est nécessaire et requis que certaines fonctionnalités soient présentes dans ce système. D'autres systèmes peuvent aussi demander certaines fonctionnalités qui restent reliées à la nature du système. C'est dans cette optique que nous avons divisé les substrates nécessaires pour créer un système CA en deux catégories. Des substrates primaires qui regroupent substrates nécessaires et obligatoires pour faire fonctionner un système CA. Pour qu'un système soit CA, tous ces substrates doivent être implémentés. Ce sont des fonctionnalités de base. Les substrates secondaires, par contre, sont des fonctionnalités qui peuvent être ajoutées aux substrates primaires pour servir un but précis qui dépend de la nature du système visé. Ce sont donc des fonctionnalités complémentaires. Dans notre étude nous avons recensé au total sept substrates primaires et cinq substrates secondaires. Dans la suite de cette section, nous allons définir et décrire chaque substrate.

Les substrates primaires

Suite à notre étude de l'état de l'art sur la sensibilité du contexte dans les réseaux de capteurs sans fil, nous avons recensé au total sept substrates primaires :

- La détection du contexte (Sensing) : consiste à détecter et capturer les valeurs et les données recueillies à partir des nœuds capteurs;
- Agrégation : consiste à combiner données capturées pour avoir des valeurs représentatives des données recueillies de la même nature. Par exemple, on prend la moyenne de toutes les données de température capturées par les différents capteurs durant une durée déterminée;
- Classification & modélisation : consiste à représenter le contexte de manière à pouvoir effectuer un stockage, un raisonnement et un échange efficaces. Deux approches différentes de modélisation influent sur l'efficacité de l'échange des données. Un modèle basé sur XML définit un schéma sur lequel les données doivent se conformer. La validation des données XML est plus simple et efficace que pour un modèle clé-valeur où les données sont textuelles et leur validation se fait manuellement et donc moins efficace. Au niveau du stockage, encore parcourir un format texte est nettement moins performant que de parcourir du XML;
- Validation : consiste à effectuer du prétraitement sur le contexte collecté. Les données collectées par les capteurs ne sont pas toujours exactes. Une température ambiante supérieure à 100 Celcius est surement une valeur fausse. Ce substrate permet d'enlever ces mauvaises valeurs qui compromettent la véracité du contexte;
- Inférence : consiste à inférer un nouveau contexte à partir du contexte collecté. Ce substrate utilisera une combinaison de valeurs de contexte pour générer une information contextuelle de niveau supérieur que l'on ne peut pas avoir directement à partir d'un capteur spécifique. Elle apporte une richesse d'information au niveau du contexte;

- Stockage : ce substrate permet de stocker les informations contextuelles efficacement et élimine les données redondantes et désuètes;
- Échange : ce substrate permet d'échanger le contexte avec les d'autres systèmes CA.

Les substrates secondaires

De la même manière, nous avons recensé et gardé à la fin cinq ressources physiques secondaires décrites comme suit :

- Prédiction : consiste à prédire de futures modifications de l'information contextuelle à partir de données historiques afin d'agir suivant ce contexte. Par exemple, la prédiction sur le contexte de la pression sanguine du corps humain peut sauver une vie humaine si on arrive à prédire correctement l'arrivée d'une hypotension;
- Support de la qualité : ce substrate permet de garantir des propriétés de qualité de contexte et de qualité de service;
- Support de la sécurité : ce substrate permet d'assurer la confidentialité et l'intégrité des données jugées critiques;
- Visualisation des données : ce substrate fournit des outils pour visualiser l'information contextuelle;
- Analyse des données : ce substrate fournit des outils pour analyser les données contextuelles.

2.3 La vue globale

D'après l'analyse de l'état de l'art, nous proposons un Framework qui respecte le concept infonuagique. Le nuage offre trois types de services : le logiciel comme service, la plateforme comme service et l'infrastructure comme service. Dans un niveau supérieur, nous trouverons la couche qui fournit les logiciels comme services. Pour les systèmes qui nous intéressent, ce sont des applications CA. La richesse du contexte que l'on peut acquérir des

réseaux de capteurs sans fil permet de concevoir une grande variété d'application dans différents domaines.
La couche qui suit est la couche de plateforme dans laquelle le fournisseur de service trouve tout ce dont il a besoin pour créer son application CA. Cette couche fournira des interfaces graphiques et des APIs qui constitueront les points d'accès au fournisseur de service. Cette couche peut aussi fournir des blocs fonctionnels qui entrent dans le développement des systèmes CA, mais ne sont pas spécifiques à ceux-ci. Ce sont des fonctionnalités génériques que l'on peut trouver présentes dans d'autres types de systèmes. Nous citons la fonctionnalité de prédiction, de visualisation des données et d'analyse des données. Le lien commun de ces blocs fonctionnels est qu'ils acceptent comme entrées des données sous format standard Independent du système qui lui fait appel.

Dans la couche inférieure qui est l'infrastructure, on retrouve les différents blocs fonctionnels qui constituent la base d'un système CA. Les blocs fonctionnels retrouvés dans cette couche sont spécifiques et liés à la nature des réseaux de capteurs sans fil. Le principal avantage de cette architecture est la simplicité avec laquelle il devient possible de développer des applications CA. Les développeurs d'applications classiques pourront migrer facilement les applications CA, car l'architecture cache toute la complexité qui se cache derrière la sensibilité au contexte. Mais, en même temps, elle fournit des fonctionnalités avancées pour les développeurs expérimentés dans le domaine de la sensibilité au contexte.

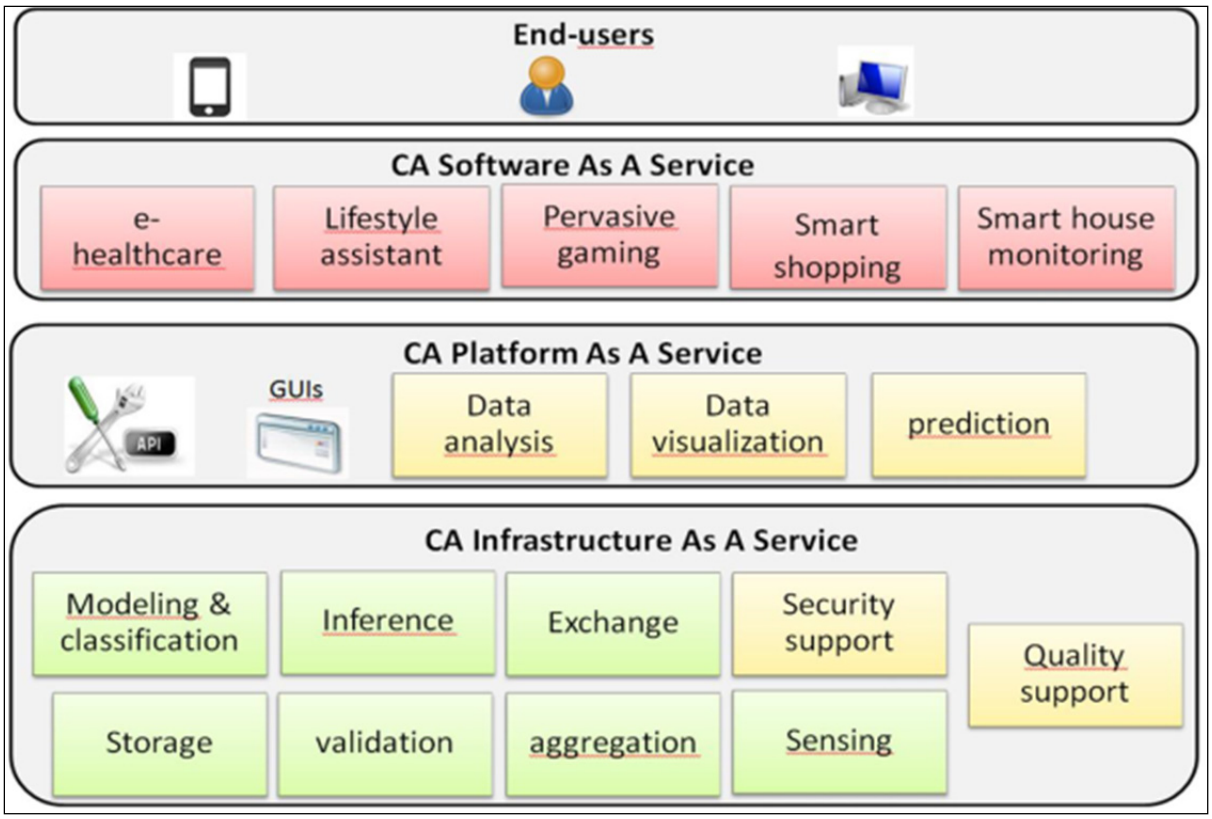

Figure 2-2 La vue générale

2.4 Architecture logicielle

La figure 2-3 illustre l'architecture logicielle globale. Cette architecture est composée de deux couches : plateforme et l'infrastructure ainsi que d'un dépôt de données (Broker).

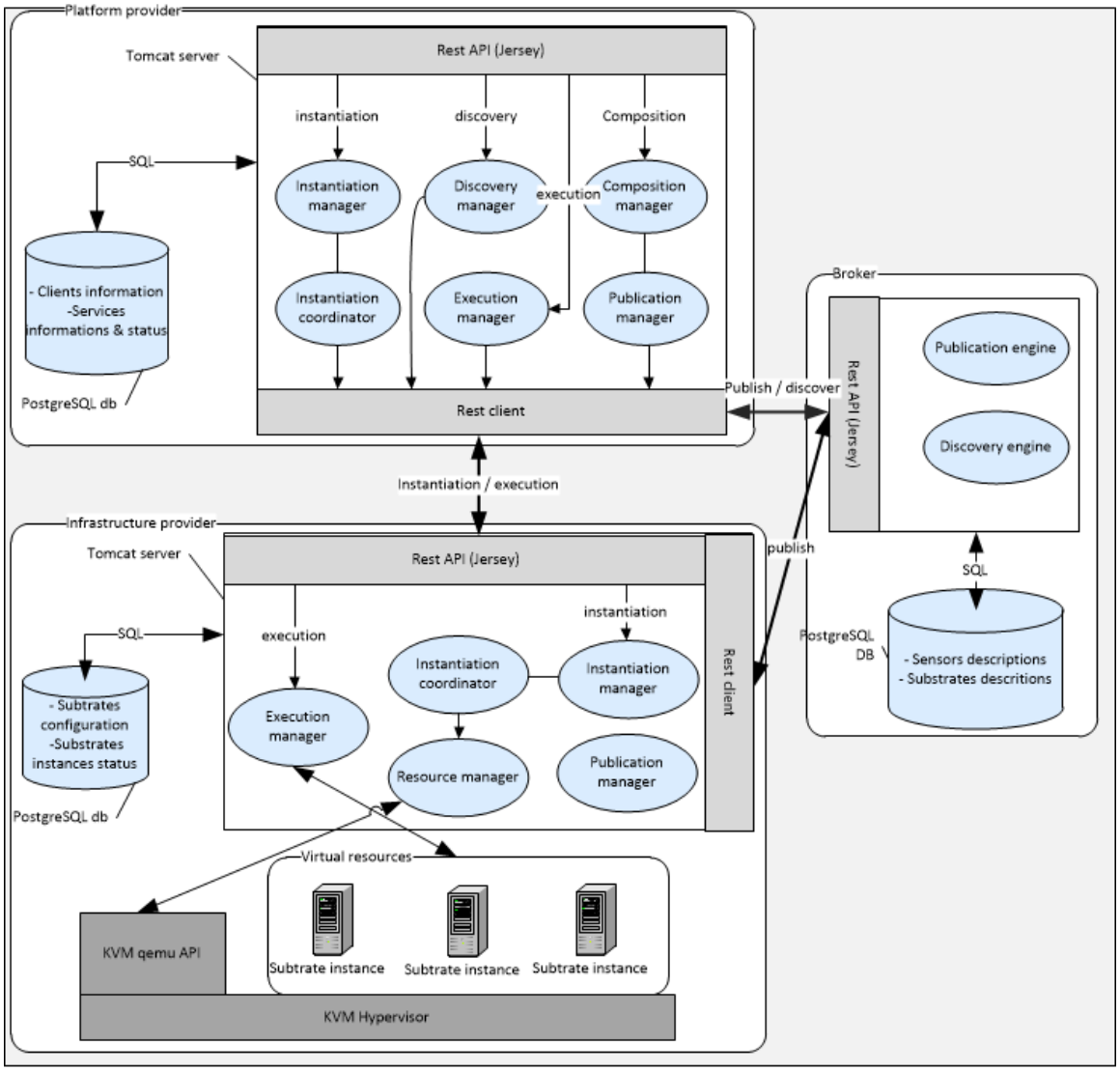

Figure 2-3 Architecture logicielle de la plateforme de gestion de contexte

2.4.1 La couche de plateforme

Cette couche offre aux fournisseurs de services un ensemble d'outils nécessaires (ex, interfaces graphiques, API, etc.) pour développer des services CA. Elle représente une couche d'abstraction entre les applications contextuelles et les infrastructures physiques. La couche de plateforme offre aux fournisseurs de services des substrates et des capteurs. En réalité, les différentes ressources sont seulement exposées par la couche de plateforme. Ce sont en fait les fournisseurs d'infrastructure qui fournissent et gèrent ces ressources.

La couche plateforme fournit les blocs fonctionnels clés suivants :

- Interfaces graphiques : elles sont utilisées par les fournisseurs de services. Ces interfaces sont les points de contact entre la plateforme comme service et le fournisseur de service. À travers ces interfaces graphiques, le client (fournisseur de service) peut découvrir les différentes ressources offertes par la plateforme. Il peut aussi, les utiliser pour composer et créer un service. Enfin les interfaces graphiques peuvent être utilisées pour voir le statut de ses services et même pour tester leur bon fonctionnement;
- API : ces librairies sont destinées à chaque fournisseur de service. Elles sont destinées à être consommées et utilisées lors de la phase de développement du service par son fournisseur. Ces APIs doivent utiliser des protocoles standardisés (Interfaces REST dans notre cas);
- Gestionnaire de découverte (Discovery manager) : il sert à traiter les différentes requêtes de découvertes qu'il reçoit de la part des fournisseurs de service. Ce gestionnaire valide le format des données de la requête, puis il l'analyse selon les différents critères de découverte et il compose une nouvelle requête destinée au courtier. La réponse du courtier est envoyée alors au fournisseur de service;
- Gestionnaire de composition (Composition manager) : ce gestionnaire permet de traiter les requêtes de composition provenant des clients. Ces requêtes sont envoyées lorsque le client a l'intention de composer un service. Le client sélectionne depuis l'interface graphique un ensemble de substrates et de capteurs et soumet la requête de composition. La principale tâche de ce gestionnaire est de créer un fichier descriptif du service composé nommé le « fichier de composition de service ». Ce fichier contient toutes les informations nécessaires pour déployer un service CA suivant le choix du client;
- Gestionnaire de publication (Publication manager) : lorsque le fournisseur de service décide de publier son service au public ou à un ensemble spécifique d'utilisateurs. Il

envoie une requête de publication au gestionnaire de publication. Ce dernier compose une nouvelle requête destinée au courtier pour que ce dernier la publie;

- Le gestionnaire d'instanciation (Instantiation manager) : lorsque le client est satisfait de sa composition de son service. Il valide en envoyant une requête d'instanciation. Cette requête permet de déployer physiquement les ressources sélectionnées. Ce gestionnaire a pour principale fonctionnalité de valider cette requête, d'identifier les différents fournisseurs d'infrastructure qui offrent les ressources dont le service a besoin. Enfin, il envoie ces informations au coordinateur d'instanciation;
- Le coordinateur d'instanciation (Instantiation coordinator) : ce nœud fonctionnel est responsable de gérer les diverses requêtes d'instanciation provenant des différents fournisseurs de services. Il utilise l'information du gestionnaire d'information pour créer différentes requêtes d'instanciation destinée à des fournisseurs d'infrastructure ciblés. Ces requêtes sont mises dans une file d'attente et sont traitées selon leurs priorités. Enfin, le coordinateur garde trace de l'instanciation du service, et dépendamment des réponses reçues des requêtes d'instanciation, il génère un message indiquant le statut générique de la requête et de son déroulement, puis l'envoie au client (fournisseur de service). Si la requête générique d'instanciation s'est bien exécutée, le coordinateur crée au niveau de la base de données un descriptif du service et expose au client les différentes API générées par le déploiement des ressources;
- Le gestionnaire d'exécution (Execution manager) : le gestionnaire d'exécution permet de traiter les requêtes d'exécution de service envoyées par le client. Le but de ces requêtes est de connaitre le statut des services déployés et aussi de tester le bon fonctionnement de ses services. Le gestionnaire d'exécution identifie alors les fournisseurs d'infrastructures qui offrent et qui déploient les ressources et envoie des requêtes d'instanciation à ces fournisseurs.

2.4.2 La couche d'infrastructure

Cette couche est responsable de fournir les différentes ressources nécessaires pour créer un service CA. Ainsi, les différents blocs fonctionnels relatifs aux systèmes CA discutés dans le chapitre précédent sont implémentés à ce niveau. L'accès aux données collectées par les capteurs est aussi implémenté au niveau de cette couche. Cette couche assure la gestion efficace des ressources pour assurer la qualité de service aux usagers. Elle offre les blocs fonctionnels suivants :

- APIs : elles sont principalement des librairies de communication qui permettent à cette couche de s'interfacer avec d'autres entités externes (le fournisseur de Plateforme comme service, autres fournisseurs d'infrastructure comme service, dépôt de données, etc.);
- Le gestionnaire d'instanciation (Instantiation manager) : ce gestionnaire traite les requêtes d'instanciation qui proviennent des différents fournisseurs de plateforme. Le rôle principal de ce gestionnaire est d'identifier les différentes ressources à déployer et les grouper sous forme de ressources appartenant au même service, puis envoyer ces informations vers le coordinateur d'instanciation;
- Le coordinateur d'instanciation (Instantiation coordinator) : cette entité priorise et traite les requêtes d'instanciation en se basant sur l'information fournie par le gestionnaire d'instanciation. La deuxième fonctionnalité qu'offre cette entité est l'envoi des demandes d'instanciations des ressources tout en gardant trace de l'état d'instanciation des autres ressources appartenant au même service. Le coordinateur a toujours une vue globale de l'état d'avancement général de l'instanciation d'un service donné;
- Le gestionnaire de ressources (Resource manager) : il est responsable de la gestion des ressources logicielles et physiques. Ce bloc a pour rôle de créer/supprimer les ressources (substrates), les configurer et les connecter entre elles. Le gestionnaire des ressources envoie régulièrement des messages de notifications au coordinateur pour

que ce dernier puisse juger de l'état d'avancement du déploiement du service demandé;

- Le gestionnaire de publication (publication manager) : lorsqu'un fournisseur d'infrastructure développe une nouvelle ressource physique (substrate) ou bien possède de nouveaux capteurs et veut les exposer et les publier, il fait appel au gestionnaire de publication qui s'occupe de cette opération. Ce dernier va formater les informations concernant la nouvelle ressource suivant les exigences du courtier et envoie la requête à ce dernier;
- Le gestionnaire d'exécution (Execution manager) : ce gestionnaire traite les requêtes d'exécution provenant des différents fournisseurs de plateforme. Il identifie qu'elle est la ressource physique déployée concernée par cette requête puis il compose et envoie une requête de test spécifique à cette ressource. La réponse de cette requête est envoyée jusqu'au fournisseur de service. Il est à noter que les requêtes d'exécution destinée à l'utilisateur final ne passent pas par ce gestionnaire; elles sont envoyées directement aux substrates déployés.

2.4.3 Le courtier (Broker)

Le courtier est une entité unique qui relie les différents fournisseurs de plateforme et d'infrastructure. Cette couche permet aux différentes parties prenantes de découvrir ce que leurs compétiteurs offrent. De plus, elle leur permet d'exposer leurs produits/services à travers des environnements infonuagiques. Le courtier contient trois éléments fonctionnels clés :

- API : Elle permet au courtier d'accepter et de recevoir les requêtes des différents fournisseurs de service;
- Le moteur de découverte (Discovery Engine) : ce moteur traite les différentes requêtes de découvertes que le courtier reçoit. Les ressources à découvrir sont les substrates et les capteurs. Donc la réponse d'une requête de découverte contient une

liste qui contient des descriptions de substrates et de capteurs sélectionnés selon les critères inscrits dans la requête;

- Le moteur de publication : ce moteur traite les différentes requêtes de publication que le Courtier reçoit. La requête de publication contient une description de la ressource à publier. Le format de description de ressources, auquel doivent se conformer les fournisseurs de services, est préalablement spécifié par le Courtier.

2.5 Diagrammes de séquences

Dans notre architecture on distingue quatre principales fonctionnalités qui sont la découverte des ressources, la publication des ressources, la composition/instanciation d'un service, et l'exécution d'un service. Ces fonctionnalités sont verticales, car elles demandent l'interaction entre différentes couches (fournisseur de plateforme comme service, fournisseur d'infrastructure comme service et le dépôt de données) pour qu'elles soient réalisées. Les diagrammes suivants décrivent les différentes phases.

2.5.1 La séquence de découverte

La figure 2-4 illustre la séquence de la découverte des ressources par le client.

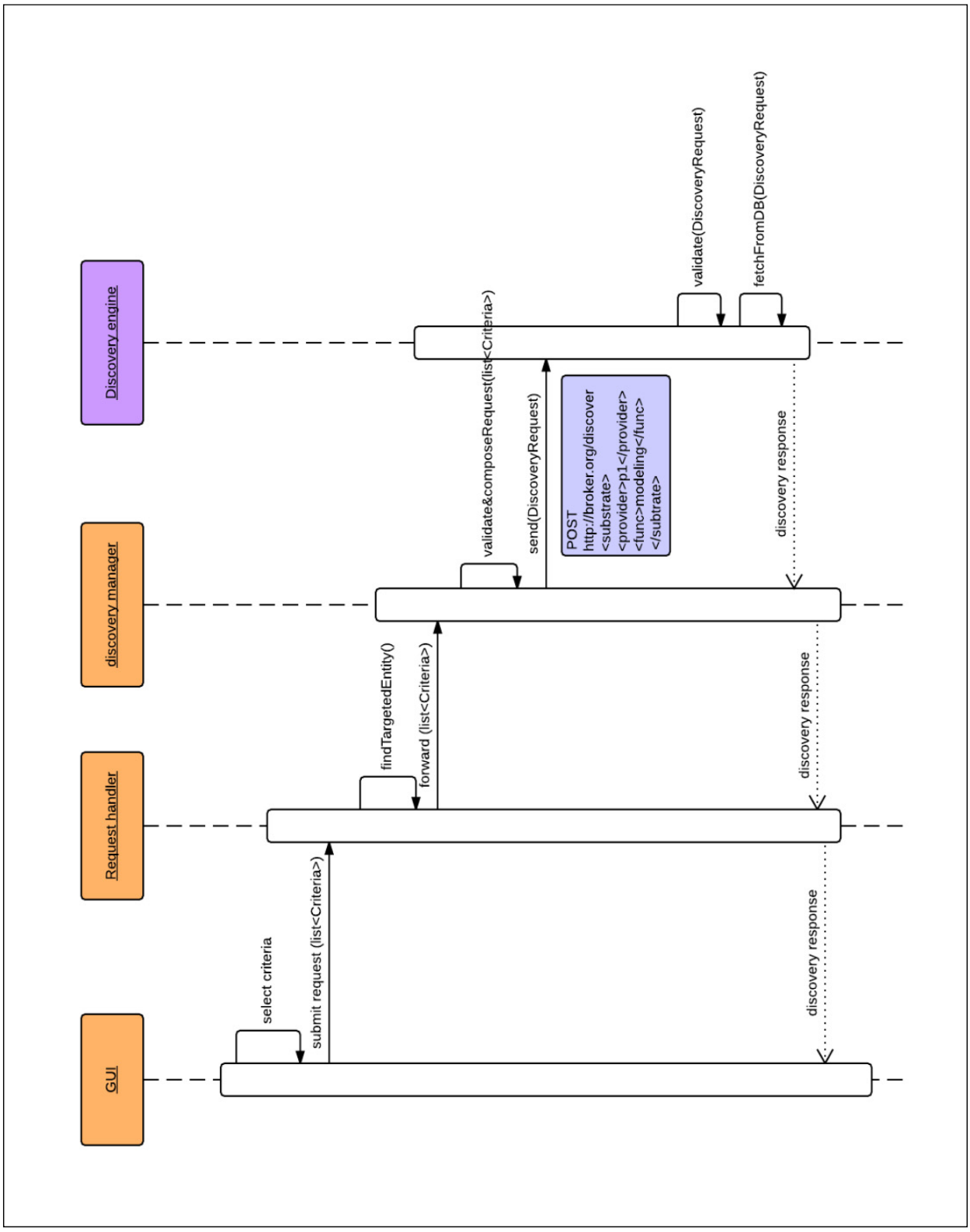

Figure 2-4 Diagramme de séquence de la phase découverte des ressources

À travers l'interface graphique, l'utilisateur sélectionne les critères pour effectuer sa recherche. Dans notre implémentation, nous avons sélectionné comme critères pour les substrates la recherche par type (ex. modeling,inference), par fournisseur d'infrastructure, par domaine d'application (ex : magasinage intelligent, e-healthcare). Pour les capteurs, nous avons sélectionné la recherche par type (ex. : température, humidité), par fournisseur d'infrastructure, par domaine d'application (ex : magasinage intelligent, e-healthcare). Une fois ses critères sélectionnés, l'utilisateur soumet son choix. La requête est traitée par le request handler situé au niveau de l'API REST au niveau de la couche plateforme. Cette entité reconnait le type de requête et la renvoie au gestionnaire de découverte de la couche plateforme. Le gestionnaire valide les données entrées par l'utilisateur et le format des critères sélectionnés. Une fois la validation terminée, il compose une requête REST destinée au Courtier. Au niveau du Courtier, le request handler reçoit la requête puis l'envoie au moteur de découverte. Ce dernier filtre les critères de recherche, compose une requête SQL qui cible la base de données du courtier. Le résultat de cette requête SQL est une liste de descriptions de substrates et de capteurs. Cette liste est formatée et mise dans la réponse de la requête REST envoyée par la plateforme. Une fois la réponse obtenue au niveau de la plateforme, on affiche le résultat final à l'utilisateur.

2.5.2 Séquence d'instanciation

La figure 2-5 illustre la séquence d'instanciation par le client.

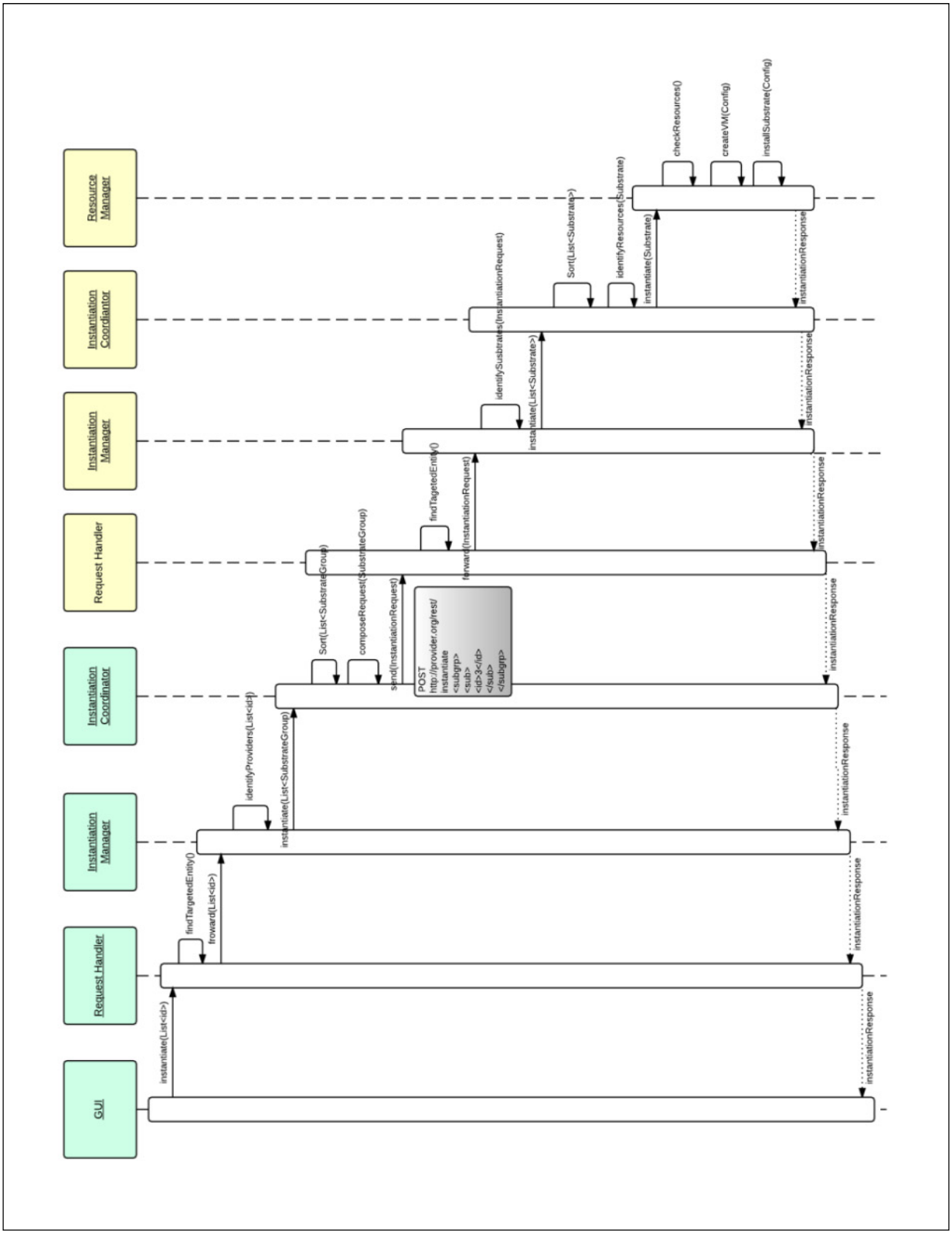

Figure 2-5 Diagramme de séquence de la phase d'instanciation

La séquence d'instanciation commence quand l'utilisateur soumet la demande d'instanciation. L'utilisateur sélectionne le fichier de composition de service puis soumet la demande d'instanciation. La requête est reçue par le Request handler de la couche plateforme qui renvoie le contenu de la requête au gestionnaire d'instanciation situé dans la même couche. À partir du fichier de composition de service, le gestionnaire d'instanciation détermine quels sont les différents fournisseurs d'infrastructures offrant les différentes ressources nécessaires pour l'instanciation du service. Une fois l'identification terminée, le gestionnaire demande à la base de données locale de la couche plateforme de fournir les informations nécessaires (ex : identifiant du fournisseur d'infrastructure, urls pour communiquer avec ces fournisseurs,etc.) pour accéder aux différents fournisseurs et comment communiquer avec eux. Après l'obtention de la réponse de la base de données, ce gestionnaire groupe et structure ces informations et les envois au coordinateur d'instanciation. Le coordinateur d'instanciation de la couche plateforme reçoit les nouvelles ressources à instancier et les met dans la file d'attente selon leur priorité (FIFO : premier arrivé premier servi). Le coordinateur crée une requête d'instanciation ciblant un fournisseur d'infrastructure donné et comportant les informations nécessaires qui identifient les différentes ressources puis il envoie cette requête à ce fournisseur d'infrastructure à travers le client REST de la couche plateforme. Au niveau de la couche infrastructure, le request handler reçoit la requête d'instanciation et la renvoie directement au gestionnaire d'instanciation de cette couche. Ce dernier identifie à partir de la requête les différentes ressources à instancier (la configuration nécessaire pour instancier ces ressources, besoin en ressources, dépendances entre les ressources, emplacement physique ou virtuel des ressources, etc.). Une fois l'identification terminée, pour chaque ressource, le coordinateur d'instanciation de l'infrastructure envoie au gestionnaire des ressources l'ordre de créer ces ressources. Le gestionnaire de ressources vérifie qu'effectivement les ressources (CPU, mémoire vive, bande passante, espace disque…) pour déployer la ressource sont disponibles. Si les ressources sont disponibles, le gestionnaire se chargera de créer une machine virtuelle suivant la configuration spécifiée par le coordinateur et la connecte au réseau pour qu'elle soit accessible. Ensuite le gestionnaire se chargera de configurer cette ressource pour qu'elle puisse fournir la fonctionnalité demandée. Pour clarifier ce processus, nous prenons comme

exemple la phase d'instanciation du substrate modeling intégrée dans le prototype décrit dans le chapitre suivant. Pour déployer un susbtrate modeling 1) une VM vierge est créée; 2) le gestionnaire des ressources analyse les informations envoyées par le coordinateur. Parmi ces informations le fichier d'extension « war » qui est l'implémentation du modèle et aussi le fichier de configuration qui contient les ressources logicielles et l'environnement logiciel à préparer pour cette implémentation (ex. installation de java, besoin d'une base de données, installation de protocoles de communication (ssh), installation de serveurs dédiés…). Cet environnement est fortement lié à l'implémentation du substrate; 3) à la fin de son traitement, le gestionnaire de ressource envoie un message de statut sur le déroulement de l'instanciation au coordinateur qui lui de son côté garde une trace des différentes ressources impliquées dans le même service. Le coordinateur envoie un statut contenant l'état d'instanciation des ressources au fournisseur de la plateforme. Enfin au niveau de la couche plateforme, son coordinateur d'instanciation génère un statut plus général sur le déroulement total de l'instanciation et l'envoie pour l'utilisateur (fournisseur de service).

2.5.3 Séquence d'exécution

La figure 2-6 illustre la séquence d'exécution d'un service par le client.

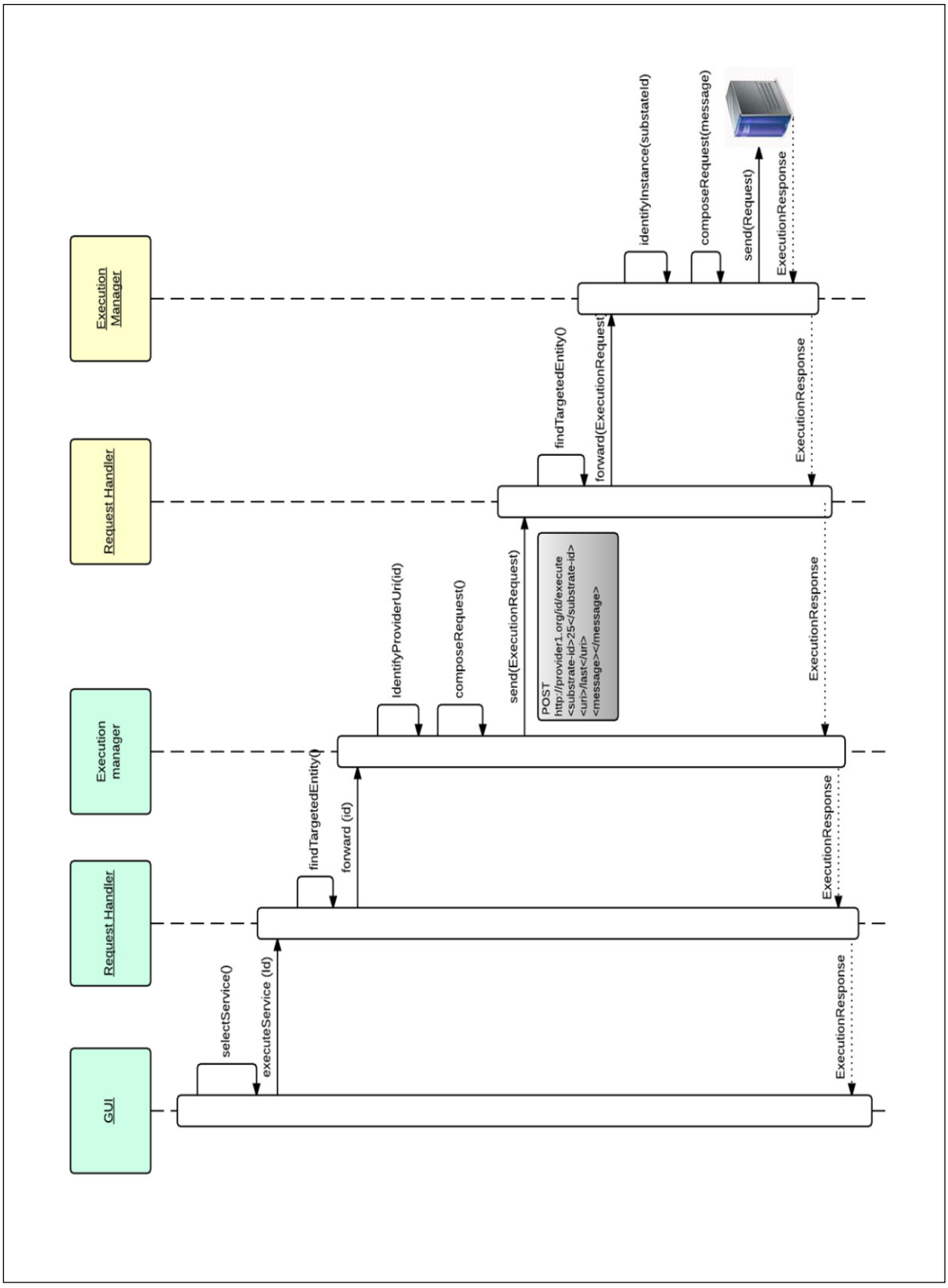

Figure 2-6 Diagramme de séquence de la phase d'exécution

La séquence d'exécution commence lorsque l'utilisateur a créé son service et essaie de tester son bon fonctionnement. Par cela, on ne veut pas tester la disponibilité du service. Mais, nous voulons voir si les valeurs en sortie correspondent aux attentes du client. Si mon service est composé juste du substrate modeling. Si je lui donne en entrée de l'information contextuelle brute par exemple la température ambiante d'une pièce, mon attente en tant que client est qu'il me retourne la même information, mais imbriquée dans le modèle proposé par le substrate suivant le format proposé aussi. C'est-à-dire transformer une information brute en une information modélisée. La séquence d'exécution commence lorsque le fournisseur sélectionne une fonctionnalité à tester. La requête est reçue par le request handler au niveau plateforme et la renvoie directement au gestionnaire d'exécution. Ce dernier identifie le fournisseur d'infrastructure qui a déployé le service et lui envoie une requête d'exécution. Au niveau du fournisseur d'infrastructure, son RequestHandler reçoit la requête et l'envoie au gestionnaire d'exécution du fournisseur d'infrastructure. Ce dernier identifie le substrate ciblé par la requête et envoie une requête qui cible l'interface REST du substrate. La réponse fournie par le substrate et renvoyée jusqu'au client.

2.6 Conclusion

Pour résumer, l'introduction de la notion de susbtrate a permis de décentraliser les systèmes CA et dans le nuage peut être considéré comme un service. Ainsi, à travers ces substrates nous avons réussi à proposer l'architecture d'une plateforme qui permet la composition et le déploiement de services CA dans le nuage.

CHAPITRE 3

IMPLÉMENTATION DE LA PLATEFORME DE GESTION DE CONTEXTE

3.1 Diagramme de classe

Dans cette section, nous présentons les deux diagrammes de classes relatifs à l'implémentation de notre plateforme de gestion de contexte. En premier lieu, nous présentons le diagramme de classe de la couche plateforme, nous enchainons ensuite par celui de la couche infrastructure.

3.1.1 Le fournisseur de plateforme

La figure 3-1 représente le diagramme de classes implémenté au niveau du PaaS. Le diagramme de classe comporte quatre principales classes : *InstantiationManager*, *InstantiationCoordinator*, *CompositionManager* et *ExecutionManager*. Ces trois classes offrent les mêmes fonctionnalités décrites dans l'architecture logicielle. Les autres classes sont des classes bean. C'est-à-dire des classes qui contiennent de l'information par exemple la classe *SCFfile* qui est une représentation objet mappé par JAXB du fichier de composition de service. De même par exemple pour la classe *SensorDescription* qui comporte la description des capteurs qui proviennent comme réponse d'une requête de découverte du Courtier. L'autre type de classe qui existe est représenté par les classes *RequestHandler* et *NotificationEngine* qui sont des classes de communication. La première implémente des interfaces pour traiter les différents types de requêtes provenant des différentes entités du PaaS (ex. gestionnaire et coordinateur) et la deuxième implémente des clients reste pour envoyer des requêtes vers l'extérieur de a plateforme. Enfin le dernier type de classe appelé utilitaire qui sert à fournir des informations et des traitements non reliés à la logique métier de l'implémentation. La classe *InstantiationResponse* ou *Status* sont l'incarnation de ce type puisqu'ils fournissent des informations sur les statuts d'exécution des requêtes.

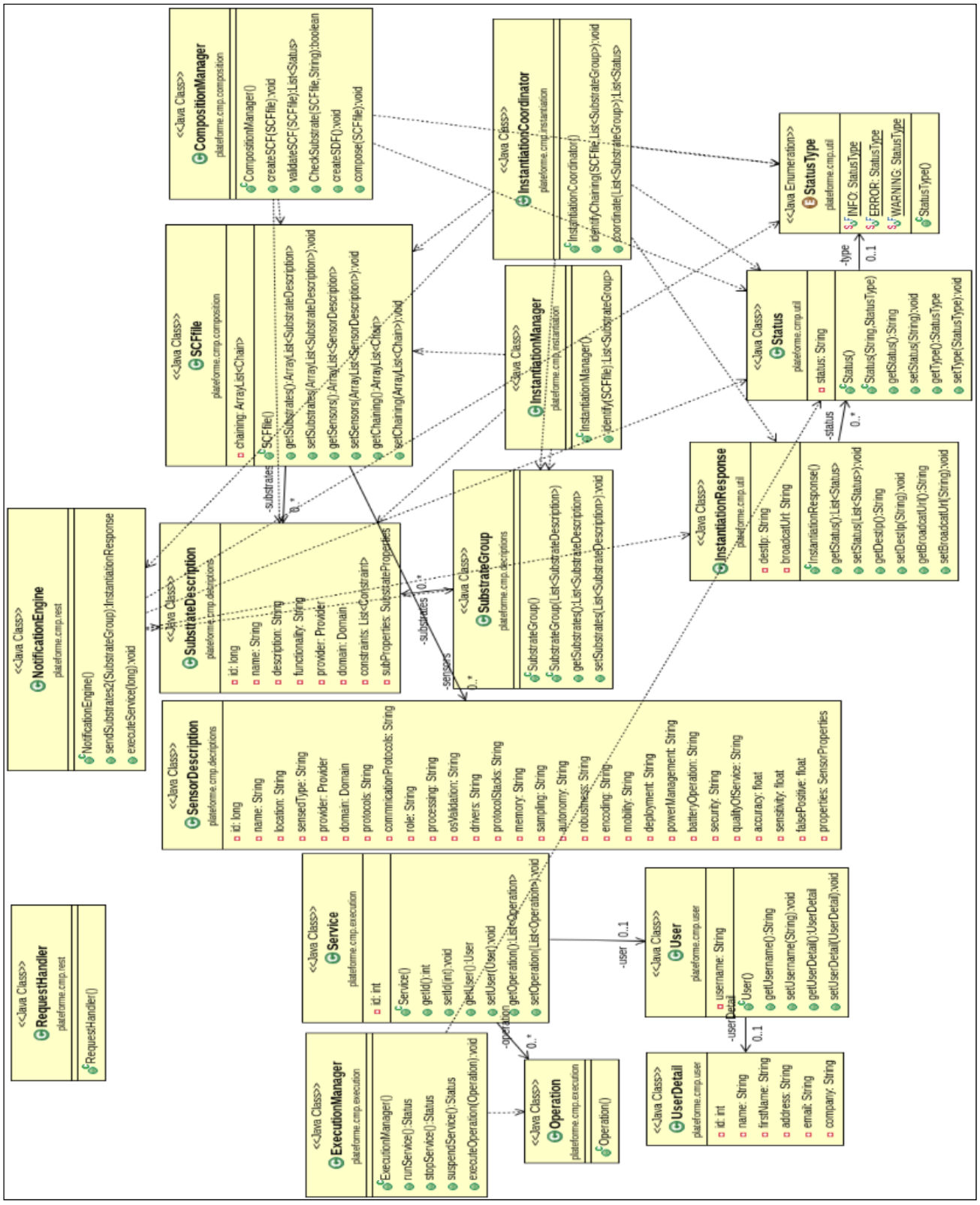

Figure 3-1 Diagramme de classe de la couche plateforme

3.1.2 Le fournisseur d'infrastructure

La figure 3-2 illustre le diagramme de classes de la couche infrastructure. Les principales classes métiers sont : *InstantiationManager*, *InstantiationCoordinator*, *ResourceManager*, *ExecutionManager*, *Instantiate*. Les classes beans sont : *Substrate*, *SubstrateInstance*, *ConfigFile*, *Operation*. Les classes de communications sont : *RequestHandler* et *NotificationEngine*. Et enfin, les classes de support sont : *InstantiationResponse*, *Status*, *StatusType*.

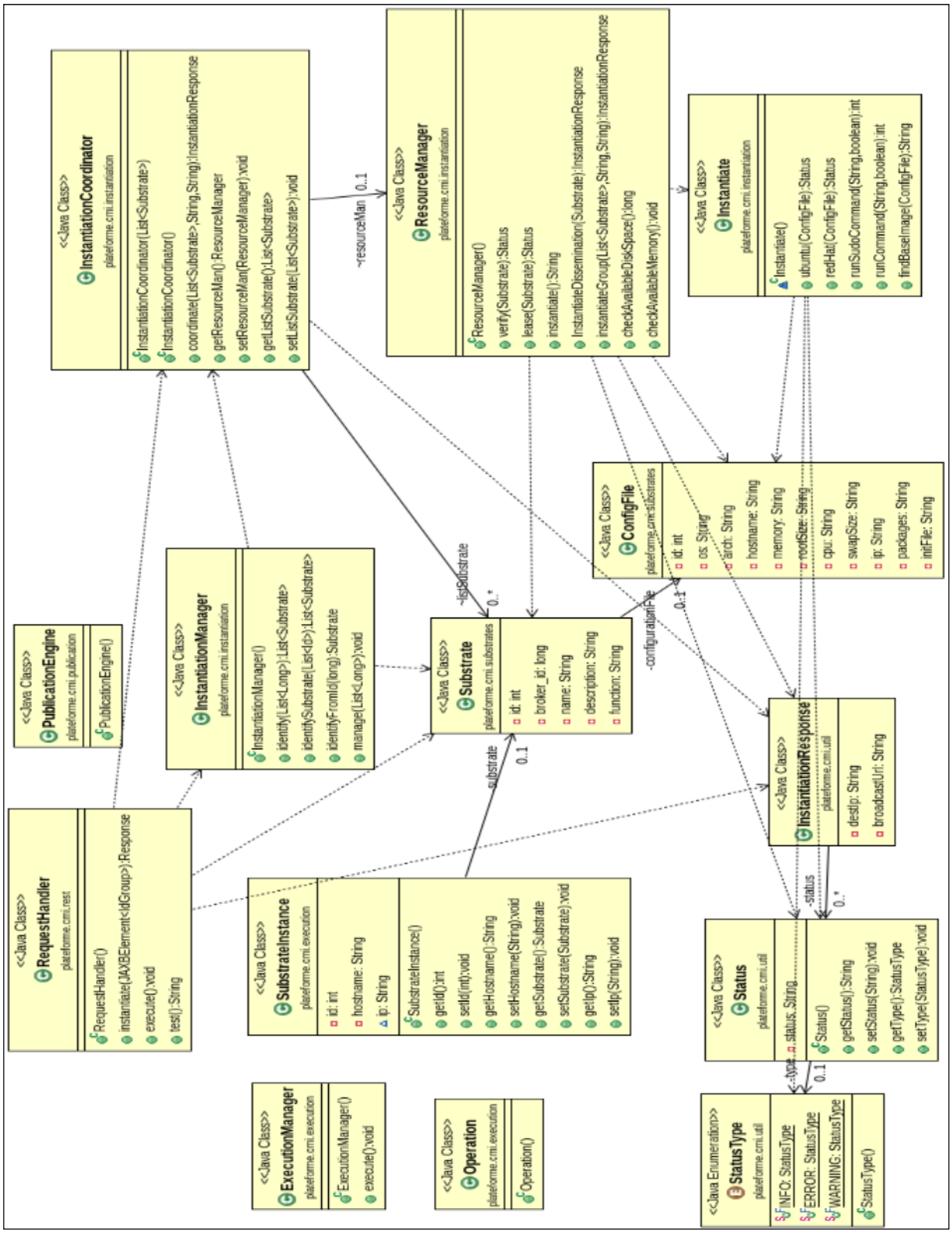

Figure 3-2 Diagramme de classe de la couche infrastructure

3.2 Les substrates

Nous avons aussi développé les principaux blocs fonctionnels présents en général dans un service CA. Ces substrates sont : acquisition, modeling, dissemination, storage et inference. Dans ce qui suit, nous décrirons chacun de ces blocs.

3.2.1 Acquisition

Ce substrate est utilisé pour l'acquisition et la collecte des données des capteurs. Elle possède deux principales fonctionnalités. La première permet de communiquer avec les capteurs et stocker l'information dans sa base de données. La deuxième permet de recevoir les données envoyées par les capteurs. La différence entre les deux fonctionnalités est qui initie la requête. En d'autres termes, dans la première fonctionnalité le substrate joue le rôle d'un client et dans la deuxième elle joue le rôle d'un serveur. Ce substrate est une application java contenue dans un serveur Tomcat qui offre une interface REST assurant la deuxième fonctionnalité. Au sein de cette application, nous avons utilisé Wildcat. WildCAT est un cadre générique pour les applications CA. Il permet le suivi des applications à grande échelle en permettant aux développeurs de facilement organiser et accéder aux capteurs avec le langage de type SQL (MySQL ou PostgreSQL) pour consulter les données de capteurs. Nous avons choisi d'utiliser le SGBD PostgreSQL pour stocker les données collectées depuis les capteurs. Notre substrate assure deux types de collecte :

- Pull : Ce type de collecte est généré par le client qui cherche à trouver à un instant donné quelle est la valeur mesurée par un capteur;
- Time driven : c'est un type de collecte géré dans le temps. Le substrate collecte périodiquement les valeurs mesurées par un capteur donné.

Un troisième type de collecte peut être supporté qui est « event driven ». C'est-à-dire on garde un listener sur le capteur qui va réagir à un évènement particulier (ex : valeur mesurée supérieure à un seuil fixé). Et dès que la condition de l'évènement est réalisée, le capteur envoie la valeur mesurée à notre substrate.

Le tableau 3-1 montre les différentes interfaces REST de ce substrate :

Tableau 3-1 Liste des interfaces REST du substrate acquisition

URL	méthode	Description
http://localhost:8080/acquisition/get	GET	Fournit la dernière valeur capturée
http://localhost:8080/acquisition/get?type =température	GET	Fournit la dernière valeur capturée du type spécifié comme argument

Les données envoyées par les capteurs sont en général de petite taille sous un format texte simple de type : « <sensor-id>;<timestamp>;<mesured-value> ». Ce format est une chaine de caractères qui contient l'identifiant du capteur, l'instant de la capture et la valeur mesurée. Le tout séparé par un séparateur donné (ex. un point-virgule). Les données reçues sont stockées sous format XML. La figure 3-3 montre un exemple de ce type de formatage.

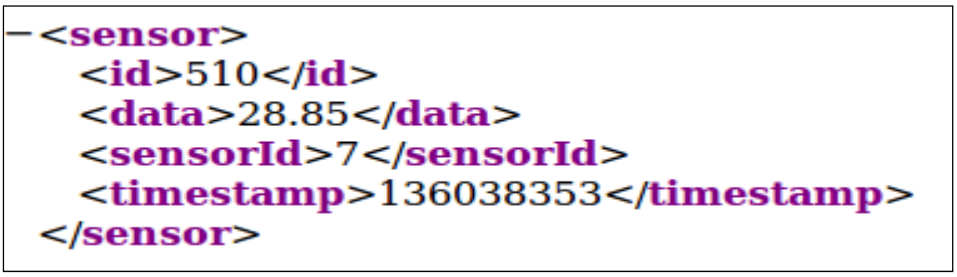

Figure 3-3 Exemple de format des données du substrate acquisition

Comme le montre cette figure, les champs du fichier XML sont les mêmes que ceux du format texte. La seule différence est que l'on ajoute l'identifiant la mesure ou l'entrée pour des raisons d'unicité.

3.2.2 Modélisation

Le substrate modeling est un substrate responsable de transformer les données brutes provenant des capteurs en données structurées suivant un modèle donné. Ce substrate offre donc une interface REST permettant d'accéder à cette fonctionnalité. Le tableau ci-dessous

décrit les interfaces REST supportées par ce substrate. Comme on l'a vu dans la revue de littérature, il y a plusieurs façons de modéliser les données. Dans le cadre de notre projet, nous avons opté pour un modèle basé sur XML pour des raisons de simplicité. Les modèles proposés en général dépendent fortement du domaine d'application. C'est pour cela que nous avons choisi de nous concentrer que sur les domaines d'application qui nous intéressent et de ne modéliser que l'information contextuelle reliée aux domaines de l'e-healthcare et du smartshopping. La figure 3-4 illustre un exemple du schéma XML du modèle proposé dans notre implémentation. Le schéma XML complet se trouve en annexe VI

```
<xs:complexType name="tuple">
  <xs:sequence>
    <xs:element name="timestamp" type="xs:dateTime" minOccurs="0"/>
    <xs:element name="physiologicalData" type="tns:physiology" minOccurs="0"/>
    <xs:element name="environmentalData" type="tns:environment" minOccurs="0"/>
    <xs:element name="identificationData" type="tns:identification" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<!-- physiology  specifictions -->
<xs:complexType name="physiology">
  <xs:sequence>
    <xs:element name="pulseRate" type="tns:pulse" minOccurs="0"/>
    <xs:element name="bodyTemperature" type="tns:bodyTemperature" minOccurs="0"/>
    <xs:element name="respiratoryRate" type="tns:respiration" minOccurs="0"/>
    <xs:element name="bloodPressure" type="tns:pressure" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="pulse">
  <xs:sequence>
    <xs:element name="pulseValue" type="tns:pulseValue" minOccurs="1"/>
    <xs:element name="pulseDescription" type="tns:pulseDescription" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="unit" type="xs:string" use="required" fixed="bpm"/>
</xs:complexType>
<xs:simpleType name="pulseValue">
  <xs:restriction base="xs:unsignedShort">
    <xs:minInclusive value="50"/>
    <xs:maxInclusive value="150"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="pulseDescription">
  <xs:restriction base="xs:string">
    <xs:enumeration value="normalRange"/>
    <xs:enumeration value="tachycardia"/>
    <xs:enumeration value="bradycardia"/>
  </xs:restriction>
</xs:simpleType>
```

Figure 3-4 Modèle de données XML implémenté

Il est à noter que ce modèle n'est qu'une proposition parmi plusieurs qui peuvent exister. La tâche du fournisseur d'infrastructure est de, lors de la phase de publication, décrire au maximum les avantages de son modèle et de son implémentation. De son côté, c'est le fournisseur de service qui choisit parmi les différents substrates de modeling celle qui correspond le mieux à ses besoins. Le tableau 3-2 fournit l'interface REST offerte par ce substrate.

Tableau 3-2 Interface REST du substrate modeling

URL	Méthode	Description
http://localhost:8080/modeling/model	POST	Transforme les données fournies par les capteurs en données contextuelles de haut niveau

3.2.3 Dissemination

Ce substrate est le nœud central des systèmes CA. Son principal rôle est d'échanger les données entre les différents substrates. Elle permet de connecter les substrates entre elles. Toutes les communications qui doivent être transmises entre les autres nœuds passent par ce substrate. Dans notre implémentation de ce substrate, nous avons proposé deux interfaces REST. La première permet de gérer les messages internes. Cette interface reçoit le message d'un substrate, identifie la source et l'envoie au destinataire. Si le message n'est pas valide, on envoie un message d'erreur au substrate source sans informer le substrate de destination. La deuxième interface gère les requêtes externes. Ce sont les requêtes qui viennent des clients. Pour accéder au service, le client utilise cette interface REST. Par exemple, si le client veut de l'information contextuelle par rapport à la température ambiante d'un environnement, il envoie une requête à travers cette interface. La température est incluse comme argument dans le corps de la requête. Alors, le substrate dissemination se chargera dans ce cas d'envoyer la requête au substrate storage pour chercher l'information reliée à la température. Le tableau 3-3 résume les interfaces REST qu'offre ce substrate.

Tableau 3-3 Interfaces REST du substrate dissemination

URL	Méthode	Description
http://localhost:8080/dissemination/forward	POST	Valide et transmet la requête au substrate ciblée
http://localhost:8080/dissemination/execute	POST	Exécute une requête qui provient du client

3.2.4 Stockage

Storage est un substrate de persistance de l'information contextuelle collectée. Ces informations bas niveau c'est-à-dire l'information qui vient directement des capteurs ou des informations de haut niveau c'est-à-dire l'information modélisée qui provient du substrate modeling. Ce substrate propose deux principales fonctionnalités. La première est le stockage des données. On a utilisé la base de données relationnelle PostgreSQL pour gérer la persistance des données. Ce choix est avantageux puisque ce SGBD offre le type XML. En d'autres termes, on peut stocker des fichiers XML comme type dans une entrée de la table ce qui va parfaitement avec notre susbtrate modeling dans laquelle on utilise un modèle basé sur XML. La deuxième fonctionnalité est d'extraire l'information contextuelle des tables de la base de données comme voulu. Dans certains cas le client aimera tirer de l'information de haut niveau d'un certain capteur dans un certain intervalle de temps. Le substrate storage devra fournir dans ce cas la réponse au client. Nous avons développé cette interface de manière à ce qu'elle puisse prendre comme argument le type de données à récupérer. Le tableau 3-4 liste les interfaces REST offertes par ce substrate :

Tableau 3-4 Interface REST du substrate storage

URL	Méthode	Description
http://localhost:8080/storage/store	POST	Stocke les données contextuelles
http://localhost:8080/storage/get	GET	Récupère les données contextuelles

3.2.5 Inférence

Ce substrate permet de créer d'autres informations sur le contexte à partir des données contextuelles de bas niveau. La principale fonctionnalité qu'offre notre implémentation de ce substrate est l'inférence. Dans notre cas, cette fonctionnalité est « stateful ». C'est-à-dire qu'elle dépend des états et des résultats précédents des requêtes traitées. Lorsqu'une nouvelle requête arrive à ce substrate, elle contient alors une nouvelle information contextuelle. Notre

substrate d'inférence dans ce cas traite cette information en faisant appel à un ensemble de règles. Trois cas se présentent, soit on crée une nouvelle information contextuelle haut niveau, soit on met à jour une information contextuelle existante, soit on annule/supprime une information contextuelle obsolète. Le tableau 3-5 décrit l'interface REST offerte par ce substrate.

Tableau 3-5 Interfaces REST du substrate inference

URL	Méthode	Description
http://localhost:8080/inference/infer	POST	Infère de nouveaux contextes à partir de données de haut ou de bas niveau.

CHAPITRE 4

ANALYSE DES PERFORMANCES DE LA PLATEFORME DE GESTION DE CONTEXTE

4.1 Technologies et outils de développement

Java : C'est un langage de programmation général. Java possède des avantages qui lui permettent de se distinguer des autres langages de programmation :

- Java est orienté objet
- Java est indépendante de la plateforme (portabilité)
- Java est distribuée
- Java est sécurisée
- Java est robuste
- Java est multithread

Nous avons utilisé ce langage de programmation comme langage de base pour développer notre plateforme de gestion de contexte et les différents substrates.

Maven : C'est un « outil de gestion de construction ». Il consiste à définir la façon, dont vos fichiers. Java seront compilées pour. Class, emballé en. Jar (ou .war ou .ear). Il est semblable à Apache Ant ou Gradle ou Makefiles en C /C ++, mais il tente d'être complètement autonome dans le sens où nous n'avons pas besoin d'outils ou de scripts supplémentaires pour réaliser des tâches courantes comme le téléchargement et l'installation de bibliothèques nécessaires, etc. Nous avons utilisé Maven dans notre projet pour effectuer la compilation et le déploiement automatique des substrates.

JEE : Java Platform, Enterprise Edition ou Java EE est une extension de la plateforme Java, Standard Edition (Java SE). La plateforme fournit un ensemble d'API qui comprend les services de réseau et internet, de sécurité et de mapping objet-relationnel, etc. La plateforme

comporte une conception basée en grande partie sur des composants modulaires s'exécutant sur un serveur d'applications. Les API que nous avons trouvé intéressantes et avec lesquelles nous avons travaillé sont :

- Javax.servlet : C'est l'implémentation du contrôleur dans le modèle MVC
- Javax.faces : Ce package définit la racine du JavaServer Faces (JSF) API. JSF est une technologie pour construire des interfaces utilisateurs à partir de composants.
- Javax.el : Ce package définit les classes et interfaces pour de Java EE Expression Language. L'Expression Language (EL) est un langage simple à l'origine conçu pour satisfaire les besoins spécifiques des développeurs d'applications Web. Il est utilisé en particulier avec JSF au niveau des interfaces graphiques.
- Javax.persistance : Ce paquet contient les contrats conclus entre un fournisseur de persistance et les classes gérées et les clients de la Java Persistence API (JPA).

Jersey RESTful Web Services framework : est un cadre pour le développement de services RESTful dans Java. Il fournit un support pour JAX-RS API et sert comme implémentation de référence pour JAX-RS (JSR 311 et JSR 339). JAX-RS est une norme qui le rend facile de créer un service RESTful qui peut être déployé sur n'importe quel conteneur de servlet ou serveur d'applications Java : GlassFish, WebLogic, WebSphere, JBoss,Tomcat, etc.

JAXB : Ce package est utilisé pour convertir les POJO de/vers XML (en GlassFish il peut également être utilisé pour convertir le POJO à/de JSON). JAX-RS gère par défaut l'interaction avec la mise en œuvre JAXB.

Primefaces : PrimeFaces est une suite de composants open source et une bibliothèque de composants qui étend les applications JavaServer Faces (JSF). Parmi les points forts de Primefaces :

- Riche ensemble de composants (htmleditor, saisie semi-automatique, support des graphiques).

- Support d'Ajax basé sur le standard JSF 2.0 API Ajax.
- Léger, sans configuration et aucune dépendance requise.
- Kit d'interface utilisateur mobile pour créer des applications Web mobiles pour les appareils portables.
- Une documentation exhaustive, grande communauté, vivante et active utilisateur.

Nous avons utilisé Primefaces pour développer nos interfaces graphiques

Choix de la base de données : nous présentons dans cette section une comparaison technique qui permet de choisir un SGBD de sous lequel le système peut fonctionner avec un niveau de performance fiable.

En termes de bases de données libres, deux produits se démarquent grâce à leur niveau de performance et de fiabilité : MySQL et Postgres. Les deux SGBD sont open source. Mysql a été développé au départ dans le seul but d'être une base de données rapide, alors que Postgres était orienté sur le respect des standards et les fonctionnalités avancées (comme les transactions et les sous-sélections). Du côté propriétaire, le produit Oracle se distingue par sa performance et sa richesse fonctionnelle.

Le tableau 4-1 présente un comparatif entre les trois SGBD :

Tableau 4-1 Comparatif entre les différents SGBD

Critère	Postgres	Mysql	Oracle
Coût	Gratuit	Gratuit	Payant
Conformité à ANSI SQL	Plus proche des standards ANSI SQL	Suit quelques aspects des standards ANSI SQL	Conforme au standard ANSI SQL
Sous requêtes	Supportés	Supportés depuis la version 5	Supportés
Transactions	Oui	Supportés avec le moteur InnoDB	Oui
Réplication	Oui	Oui	Oui via un produit spécifique
Support des clés étrangères	Oui	Supportés avec le moteur InnoDB	Oui
Vues	Oui	Oui	Oui
Procédures stockées	Oui	Non	Oui
Triggers	Oui	Oui, depuis la version 5.0.2	Oui
Unions	Oui	Non	Oui
Full joins	Oui	Non	Oui
Contraintes	Oui	Non	Oui
Vacuum	Oui	Non	Oui
ODBC	Oui	Oui	Oui
JDBC	Oui	Oui	Oui
Support XML	Oui	Non	Oui

Nous privilégions ainsi l'usage de SGBD open source PostgreSQL pour gérer la persistance des données.

Tomcat : Apache Tomcat est un serveur Web et conteneur de servlets open source développé par Apache Software Foundation (ASF). Tomcat implémente plusieurs spécifications Java EE, y compris Java Servlet, JSP (JavaServer Pages), Java EL. Tomcat présente les avantages suivants :

- Gratuité : Tomcat est un conteneur Web du projet open source Apache. Tomcat est un logiciel libre.

- Performance : Tomcat présente les garanties de performances nécessaires pour l'exploitation du système
- Portabilité : Tomcat est portable sur différentes plateformes (Unix, Windows…)

Hibernate : Hibernate est un Framework de gestion de la persistance, permettant de gérer les communications avec la BDD en se basant sur un fichier de mapping XML facile à déployer. L'utilisation, de Hibernate sur les applications, rend ces dernières indépendantes des bases de données puisque toutes les modifications interviennent sur les fichiers de mapping et de configuration. L'utilisation de Hibernate permet de garantir une indépendance de la solution logicielle par rapport à système de gestion de base de données adopté.

4.2 Environnement de test

La figure 4-1 illustre l'environnement de tests qe nous avons utilisé pour tester les performances de la plateforme de gestion de contexte. Nous possédons deux machines physiques connectées au même réseau local. Nous avons installé KVM sur les deux machines physiques. La première machine (PM1) peut supporter jusqu'à quatre machines virtuelles qui sont acquisition, dissemination, modeling et storage. La deuxième machine (PM2) supporte jusqu'à trois machines qui sont : inference, le dépôt de données et le serveur de monitorage Zabbix. Toutes les machines virtuelles ont été créées à l'aide d'un pont qui les connecte au même réseau local que celui des machines physiques. Toutes les machines virtuelles utilisent comme système d'exploitation Linux Server 12.04 connu pour sa stabilité et ses besoins minimes en termes de ressources.

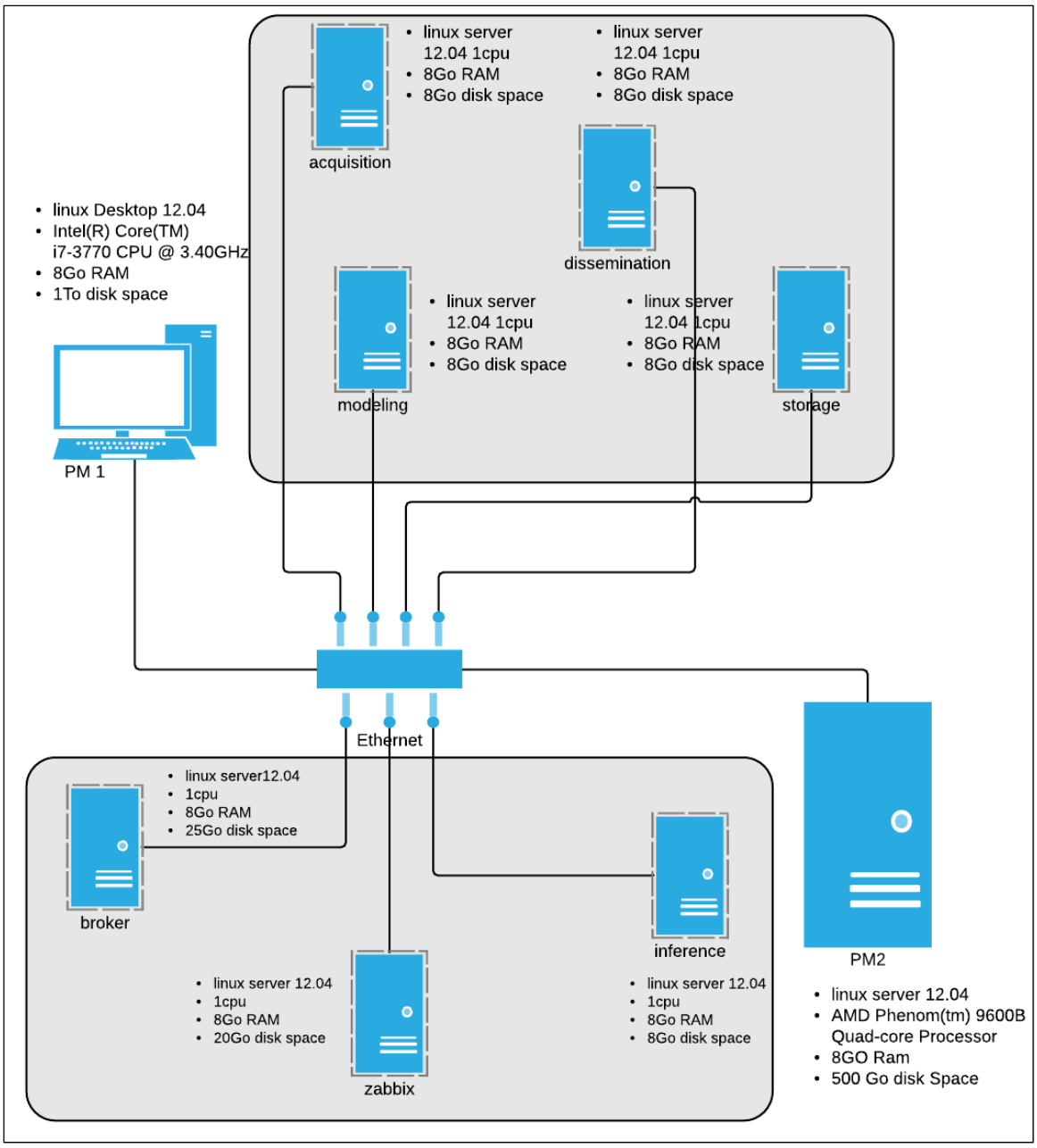

Figure 4-1 Environnement des tests de performance

4.3 Tests de la plateforme

Dans cette section, nous analysons les performances de la plateforme développée et les différents substrates pour valider leur bon fonctionnement au niveau fonctionnel. Par ailleurs, nous proposons de mesurer de manière quantitative l'utilisation des différentes ressources

dépendamment des différentes configurations disponibles. Tout d'abord, nous commencerons par réaliser les différents tests fonctionnels. Ensuite, nous aborderons les tests qui fournissent les temps de réponse des différentes requêtes qui circulent au sein de la plateforme. Puis, nous exécuterons des tests en changeant le type de données et le contenu des requêtes. Enfin nous réaliserons des tests de stress pour voir quelles sont les limites de notre système.

4.3.1 Validité et fiabilité des outils de mesure

Une analyse de performance efficace repose sur des outils de mesure fiables et que pour choisir le bon outil, nous proposons d'analyser la validité et la fiabilité de deux outils : Zabbix et Collectd. Le choix de ces deux outils est dû au fait qu'ils sont largement utilisés auprès de la communauté scientifique.

Lorsqu'il n'y a pas de Traffic, la consommation de base en termes de CPU est en moyenne de 1,21 % pendant une durée de 20 minutes pour Zabbix, et de 1.22 % pour Collectd. On peut dire alors que les deux outils capturent les mêmes valeurs avec 0,01 % de différence en précision.

Effet de Collectd sur la consommation de CPU

On arrête le service de monitorage de Collectd et on garde Zabbix seul faire le monitorage. On remarque que la nouvelle valeur moyenne de la consommation du CPU (capturée entre 13h15 et 13h45) est de 0,12 %. On peut dire que la consommation moyenne du service de monitorage Collectd est de 1,21 – 0,12= 0,09 % du CPU. Cependant lorsqu'on relance le service de monitorage Collectd. On remarque que la valeur ne revient pas à l'état d'origine que l'on s'y attend, mais reste dans une valeur faible à 1,21 %. Ceci montre que la consommation que la baisse de consommation du CPU n'est pas reliée seulement à l'arrêt de collectd, car le processeur ne revient pas à son état original après démarrage du service collectd.

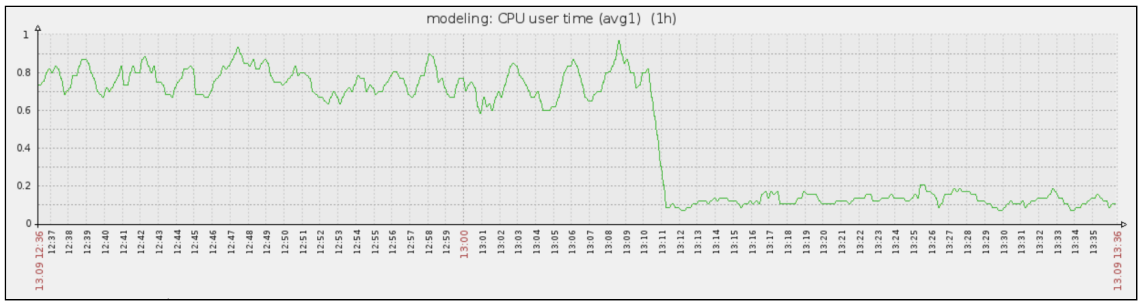

Figure 4-2 Utilisation du CPU du substrate modeling en état inactif
sur une courte durée

De même la consommation de base en termes de CPU est en moyenne de 0,6 % étalée sur presque 4 jours montre que la consommation de base est de 0,6 % sachant que les deux services de monitorages sont en marche.

La figure 4-3 montre que la consommation du CPU de collectd et de zabbix ne dépasse pas les 0.6 % lorsqu'on ne stresse pas la machine.

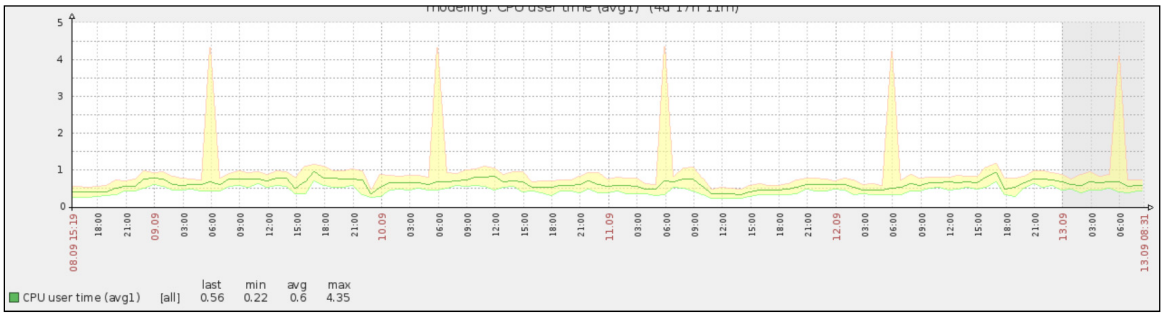

Figure 4-3 Utilisation du CPU du substrate modeling en état inactif
sur une durée longue

Un autre test intéressant est de comparer le comportement de la machine lorsqu'elle est stressée par un flux externe de données lorsque collectd est en marche ou bien à l'arrêt.
La figure 4-4 montre le résultat de ce test. La première partie où l'on remarque il y a une forte utilisation des ressources correspond à une première batch de requêtes qui stressent le service de modeling. Dans cette durée Collectd est en marche et la consommation moyenne du CPU est de 90 %. Après la fin d'exécution de cette lot, on stoppe le service collectd et on

attend que le système se stabilise puis on relance la même batch de requêtes. Durant cette deuxième partie de stress, on remarque que la consommation du CPU est en moyenne de 90 %.

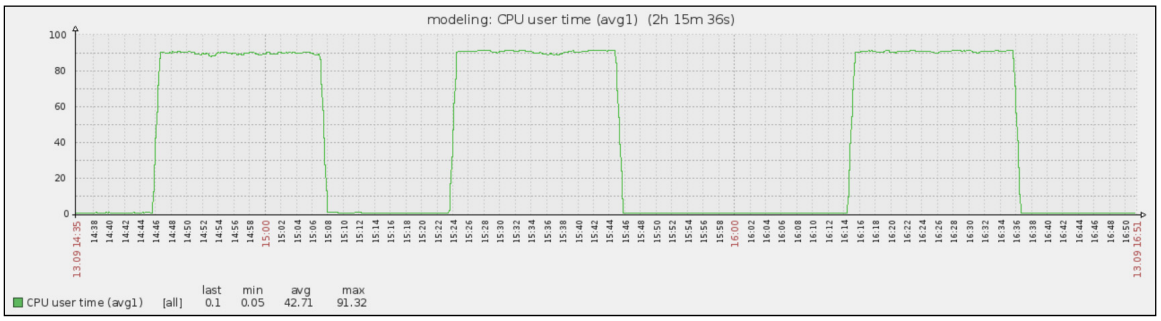

Figure 4-4 Effet de Collectd sur l'utilisation du CPU du substrate modeling

La différence entre les deux scénarios de montre que l'impact de collectd sur l'utilisation des ressources est négligeables. Dans les tests suivants, on peut négliger la quantité de CPU consommée par les outils Collectd et Zabbix et nous ne garderons que Zabbix pour comme outil de mesure.

4.3.2 Temps de réponse

Ce test permet d'avoir une idée sur la responsivité de notre système. Les différentes requêtes prises en considération sont des requêtes entre le client et la plateforme. On néglige les requêtes intra-plateforme, car elles sont des fois incluses dans les requêtes du client et aussi elles ne sont pas représentatives du temps qui impacte le client. On cherche à voir si notre système est capable de servir rapidement ou non le client. Les requêtes prises en considération sont : la découverte des ressources, la publication des ressources, l'instanciation d'un service dépendamment du nombre de composants et l'exécution d'un service. Le tableau 4-2 résume les temps de réponse pour les différents cas :

Tableau 4-2 Temps de réponse des requêtes des clients

Opération	Temps de réponse
Découverte	130 ms
Publication	104 ms
Composition avec 1 substrate	98 s
Composition avec 2 substrates	142 s
Composition avec 3 substrates	218 s
Composition avec 4 substrates	296 s

Ces temps de réponse sont acceptables pour les requêtes de découvertes et de publications. Pour les requêtes de compositions, le temps nécessaire pour créer une machine virtuelle est le facteur qui rend ce temps de réponse élevé, mais acceptable.

4.3.3 Tests de changement de données

Dans ce test, nous essayons de voir quel est l'impact du changement du contenu de la requête sur la consommation des ressources. Ce test ne peut être exécuté que sur deux substrates ou l'on peut dire qu'il y a en effet un changement de contenu. La première est le substrate modeling dans laquelle on peut augmenter la taille de la donnée à modéliser. Ceci peut être appliqué aux capteurs qui fournissent des données de types texte et qui n'ont pas de restrictions au niveau taille. Dans notre cas, nous avons utilisé le capteur de code QR pour augmenter à volonté la taille de ce capteur. On peut aussi utiliser d'autres types de capteurs pour réaliser ce type de test par exemple des capteurs d'images ou des enregistreurs vidéos dont la taille de la donnée fournie peut varier grandement dans un scénario réel. La figure ci-dessus montre l'effet de ce changement sur la consommation du CPU.

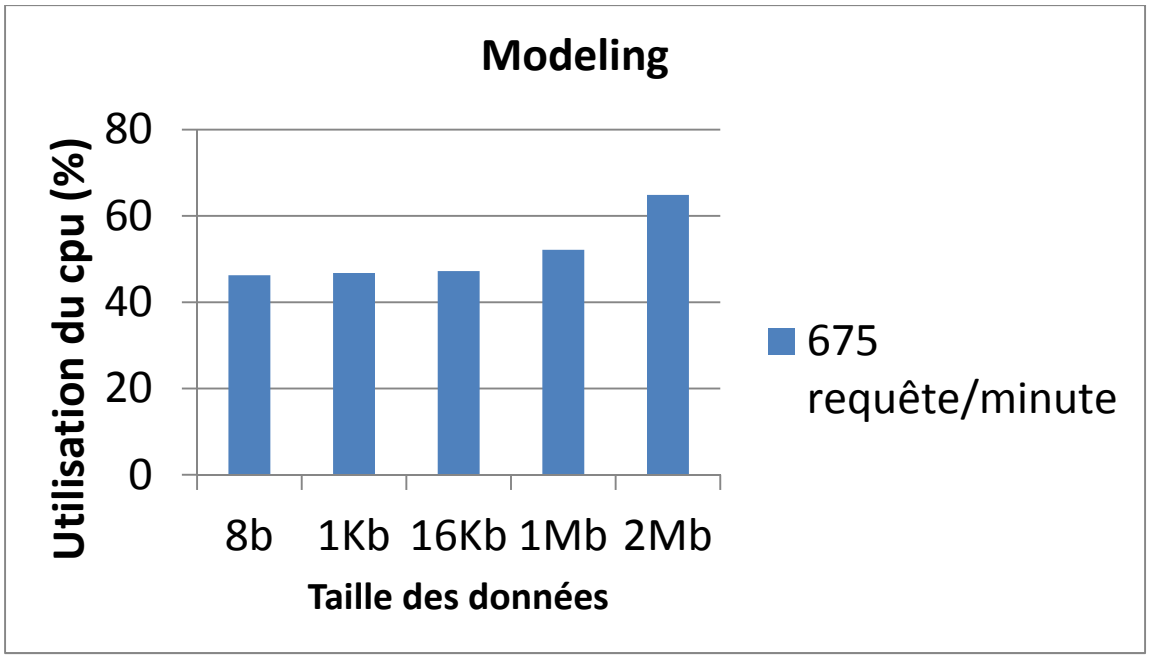

Figure 4-5 Utilisation du CPU du substrate modeling en fonction de la taille de la requête

On remarque que la consommation de CPU ne change pas pour des tailles de données ne dépassant pas 16Kb. Dès qu'on dépasse ce seuil, on commence à voir une augmentation dans la consommation du CPU où elle atteint jusqu'à 17 % d'augmentation. Cependant, il faut noter que les données fournies par les capteurs sont majoritairement très petites (dans l'ordre de 30 octets).

Au niveau du substrate modeling aussi, on peut changer le nombre de paramètres à modéliser et voir quel est leur impact sur la consommation des ressources. Ce scénario est à envisager, car, de plus en plus, les capteurs deviennent multifonctionnels et ne capturent plus qu'un seul type de données. Le capteur avec lequel on a travaillé fournit 5 types de données. La figure ci-dessous montre quel est l'effet du changement du nombre de paramètres sur la consommation du CPU.

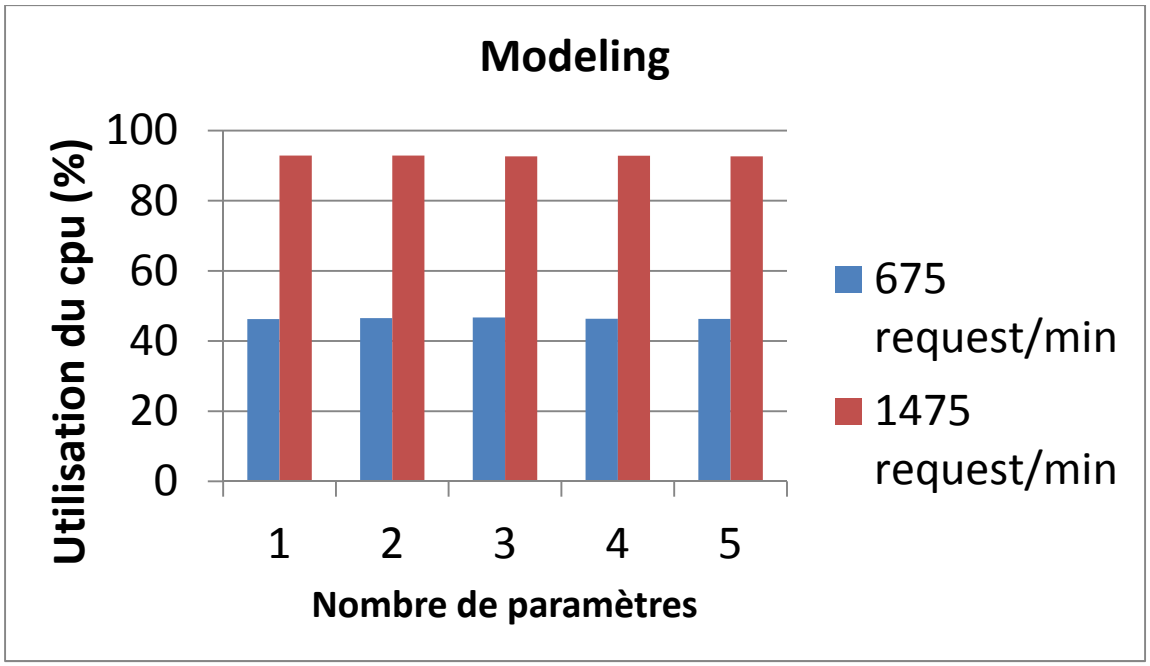

Figure 4-6 Utilisation du CPU du substrate modeling en fonction du nombre paramètres à modéliser

On remarque que le nombre de paramètres passés n'a pas d'impact sur la consommation du CPU.

Le deuxième substrate est inference. Dans ce cas, nous avons essayé de changer le nombre de règles testées durant le traitement de la requête. Dans ce test, nous fournissons des données en entrée de manière à activer les règles d'inférence qu'on a ciblée. En contrôlant l'entrée, on contrôle le nombre de règles à tester. La figure ci-dessous montre quel l'effet de ce changement sur la consommation du CPU.

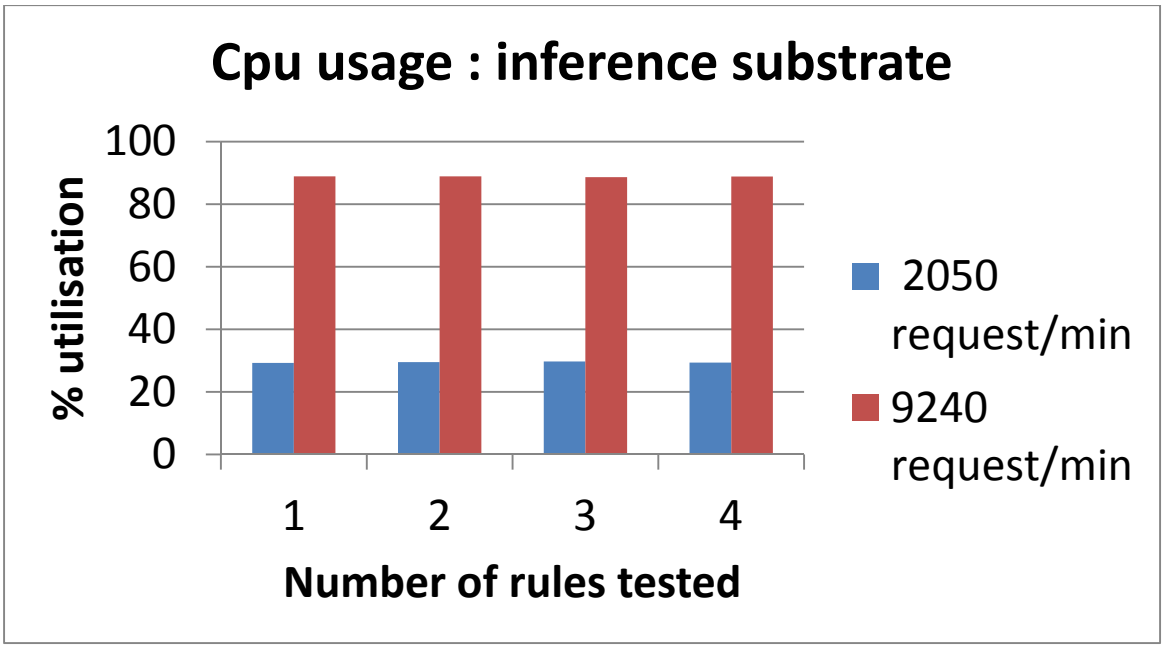

Figure 4-7 Utilisation du CPU du substrate inference en fonction du nombre de règles testées

On remarque que la consommation du CPU n'est pas affectée pas le nombre de règles qu'on teste. Normalement, dans les implémentations de substrate d'inference on devrait avoir des moteurs d'inference comme Jena qui gèrent cette fonctionnalité. Dans notre implémentation nous n'avons pas utilisé de moteur d'inference et nous avons utilisé un nombre limité de règles. La raison derrière cela est que c'est en dehors de notre scope du projet, car c'est un domaine d'expertise est très vaste. La structure simple de notre implémentation est la raison pour laquelle la consommation du CPU n'est pas affectée par le nombre de règles à vérifier.

4.3.4 Tests de stress

Les tests de stress consistent à voir quel est l'impact de la charge sur la consommation des ressources. Les ressources à surveiller sont : l'utilisation du processeur, l'utilisation de la mémoire vive et le trafic réseau.

Le CPU

Le but de ce test est de voir le changement de l'utilisation du CPU en fonction de la charge fournie. Pour mesurer cela, on envoie une charge constante de requêtes vers le substrate pendant une durée fixée (en moyenne entre 10 et 15 min) et on mesure la moyenne de consommation du CPU durant cette période. On refait le test en augmentant ou en diminuant la charge (nombre de requêtes par minutes) envoyée pour essayer de couvrir au maximum les valeurs possibles de l'utilisation du CPU. Les figures ci-dessous illustrent cette variation en fonction du nombre de requêtes par minute et du nombre de cœurs pour les cinq substrates développés.

Les figures 4-8, 4-9, 4-10 et 4-11 montrent la variation du CPU pour le substrate modeling.

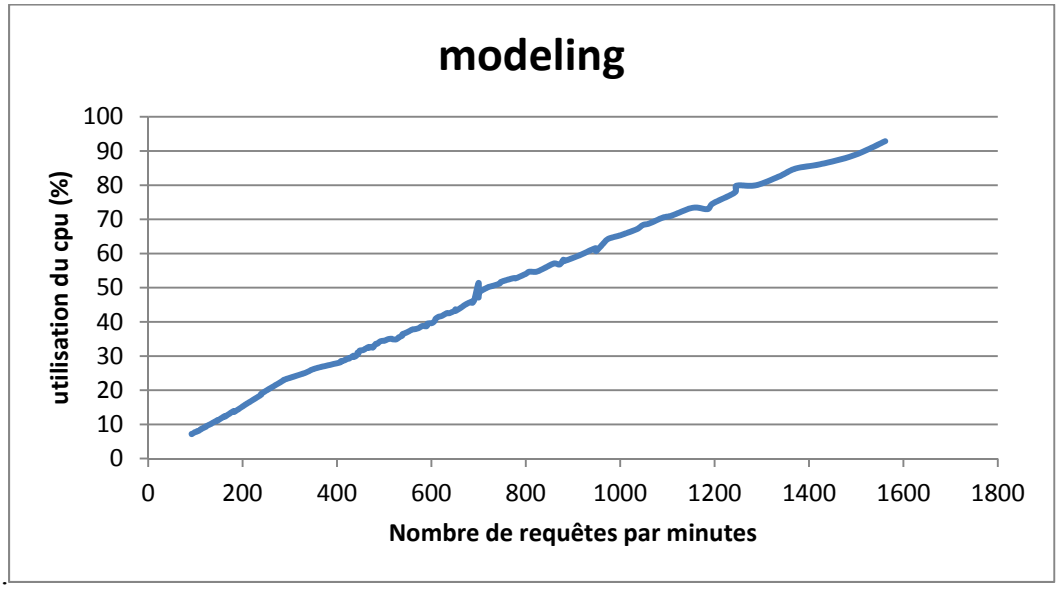

Figure 4-8 Utilisation du CPU du substrate modeling avec un seul coeur

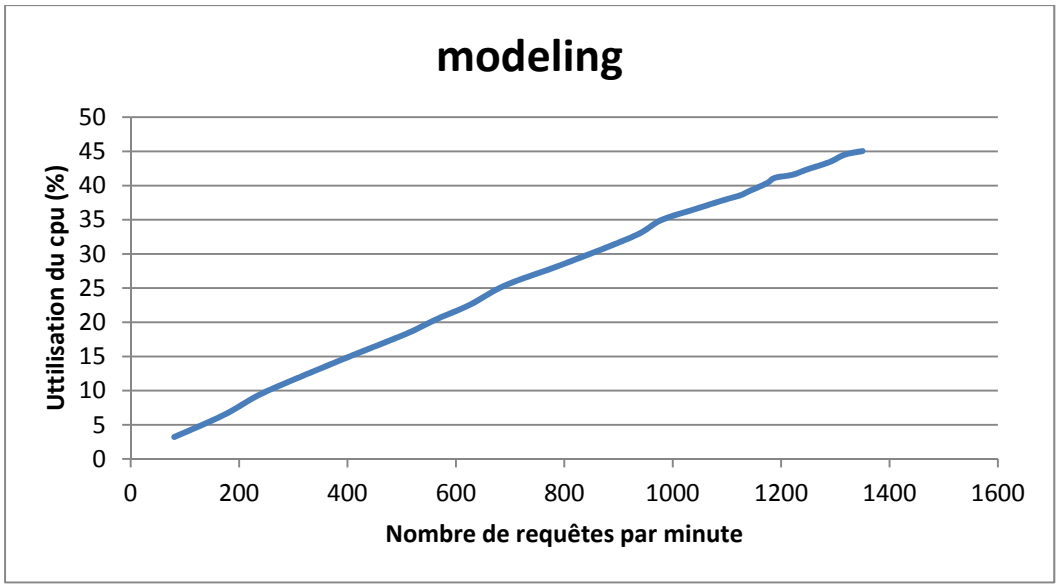

Figure 4-9 Utilisation du CPU du substrate modeling en utilisant deux cœurs

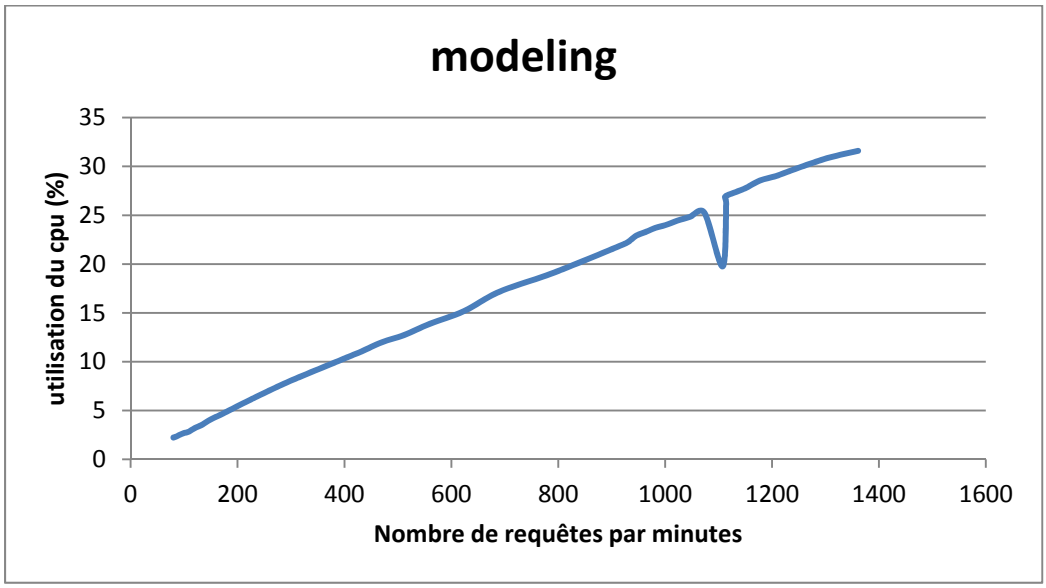

Figure 4-10 Utilisation du CPU du substrate inference avec trois cœurs

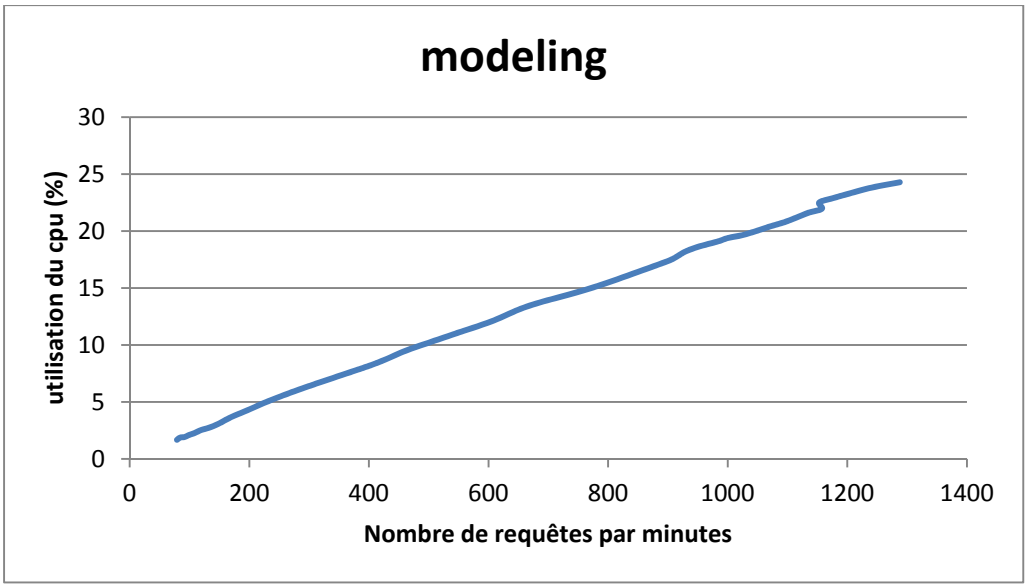

Figure 4-11 Utilisation du CPU du substrate modeling avec quatre cœurs

Les figures 4-12, 4-13, 4-14 et 4-15 montrent la variation du CPU pour le substrate acquisition.

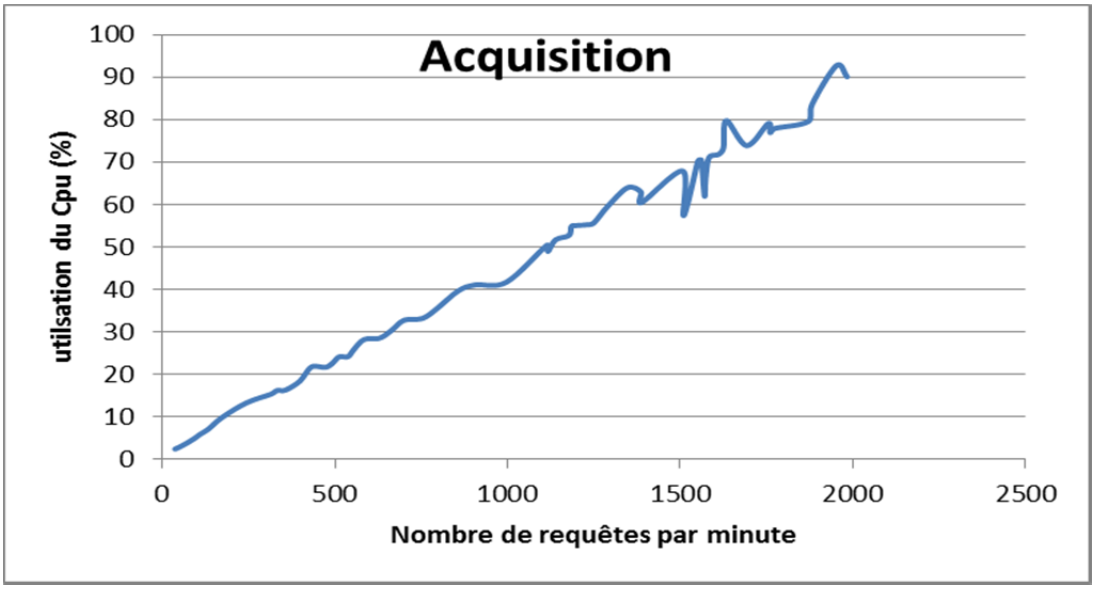

Figure 4-12 Utilisation du CPU du substrate acquisition avec un seul coeur

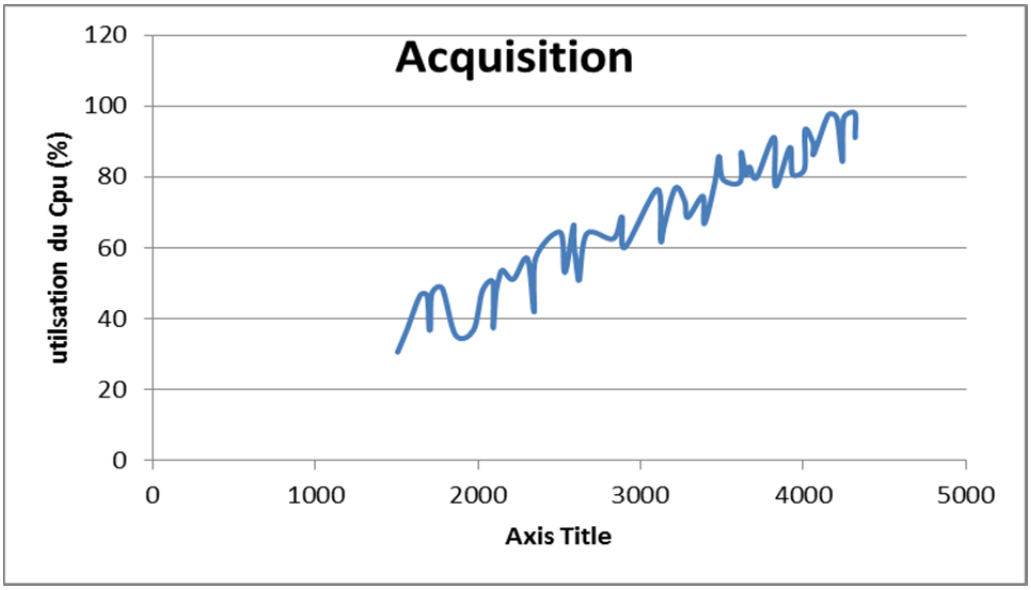

Figure 4-13 Utilisation du CPU du substrate modeling avec deux cœurs

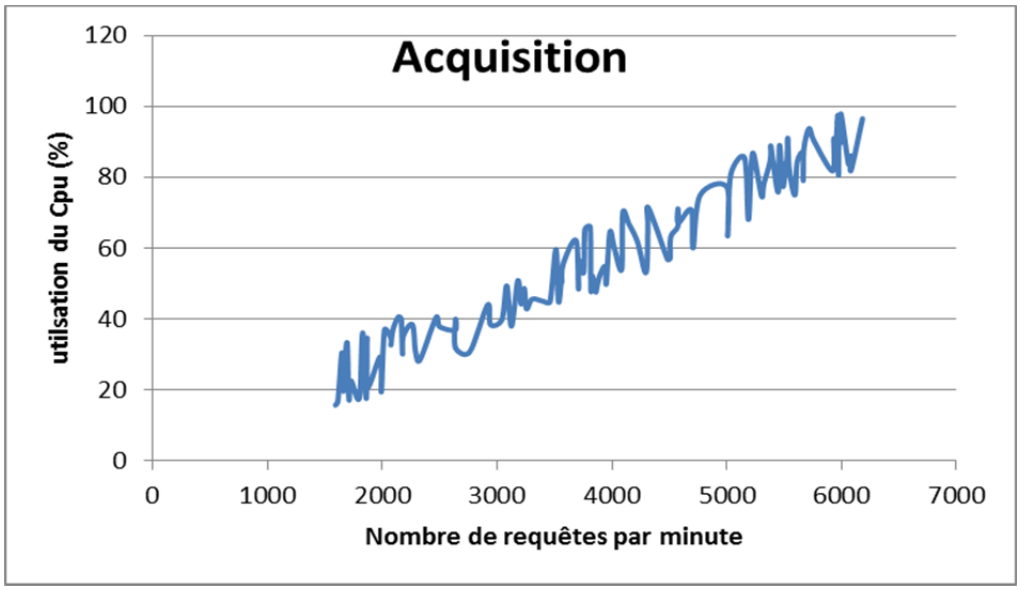

Figure 4-14 Utilisation du CPU du substrate acquisition avec trois coeurs

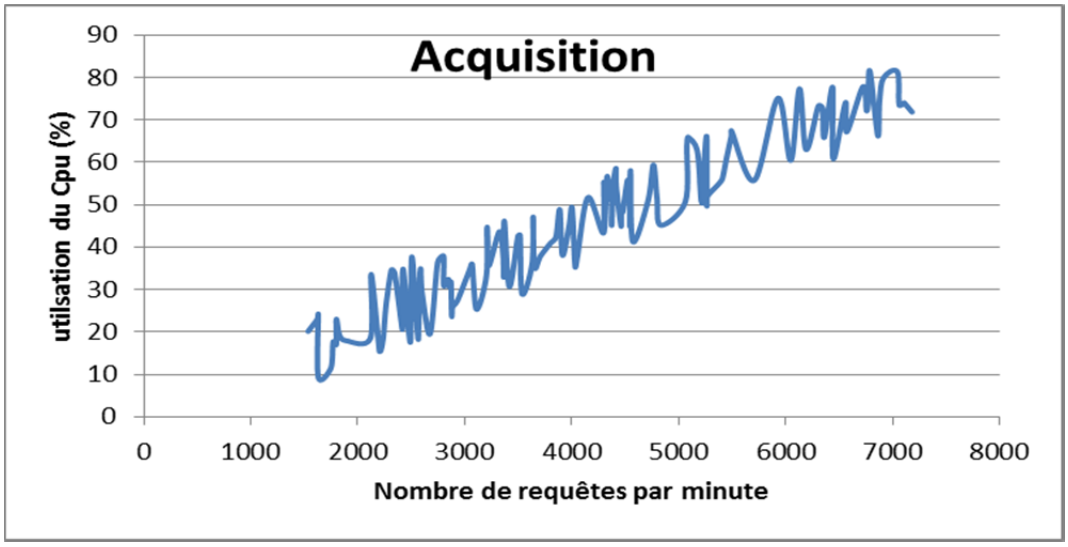

Figure 4-15 Utilisation du CPU du substrate acquisition avec quatre cœurs

Les figures 4-16, 4-17, 4-18 et 4-19 montrent la variation du CPU pour le substrate dissemination.

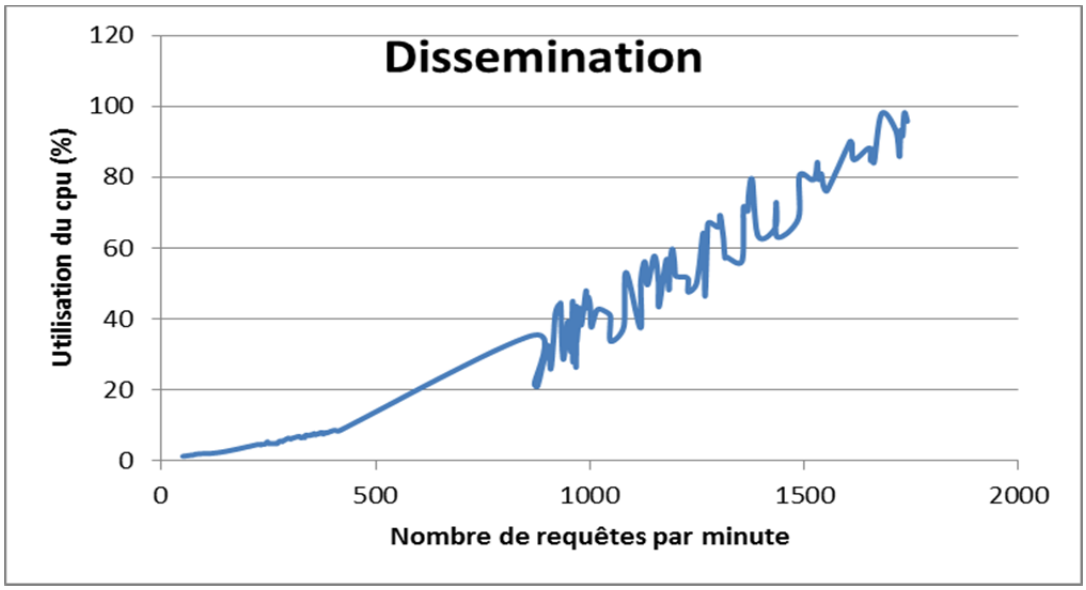

Figure 4-16 Utilisation du CPU du substrate modeling avec un seul coeur

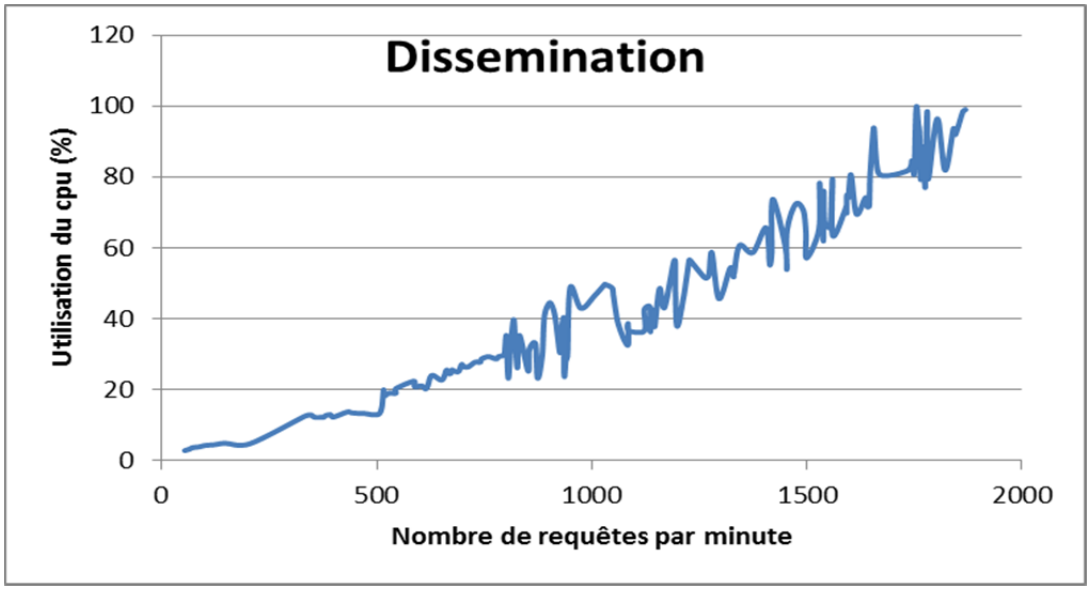

Figure 4-17 Utilisation du CPU du substrate dissemination avec deux cœurs

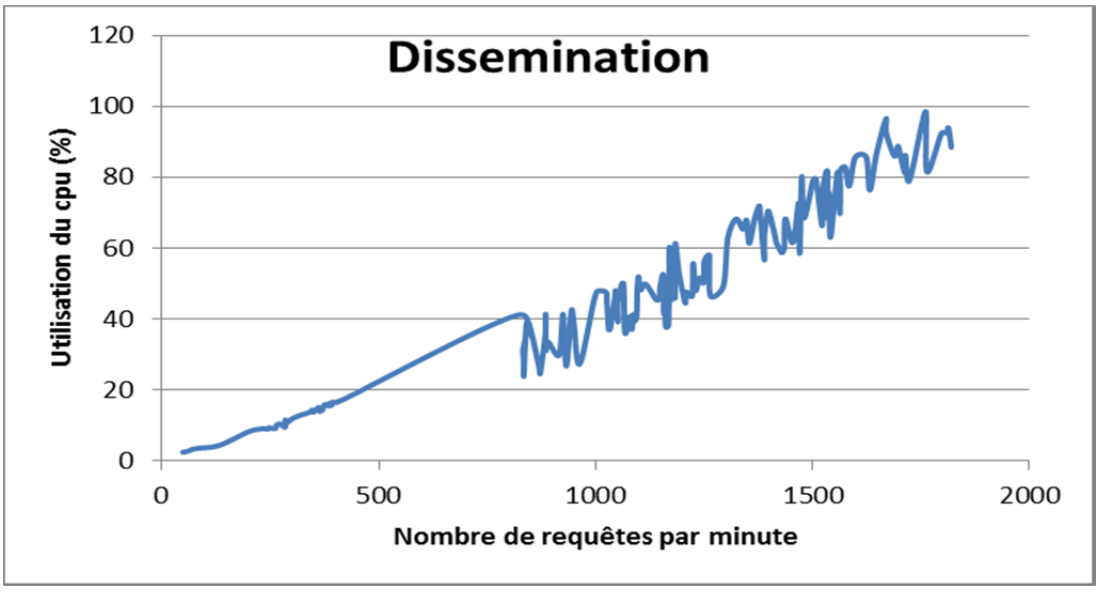

Figure 4-18 Utilisation du CPU du substrate dissemination avec trois cœurs

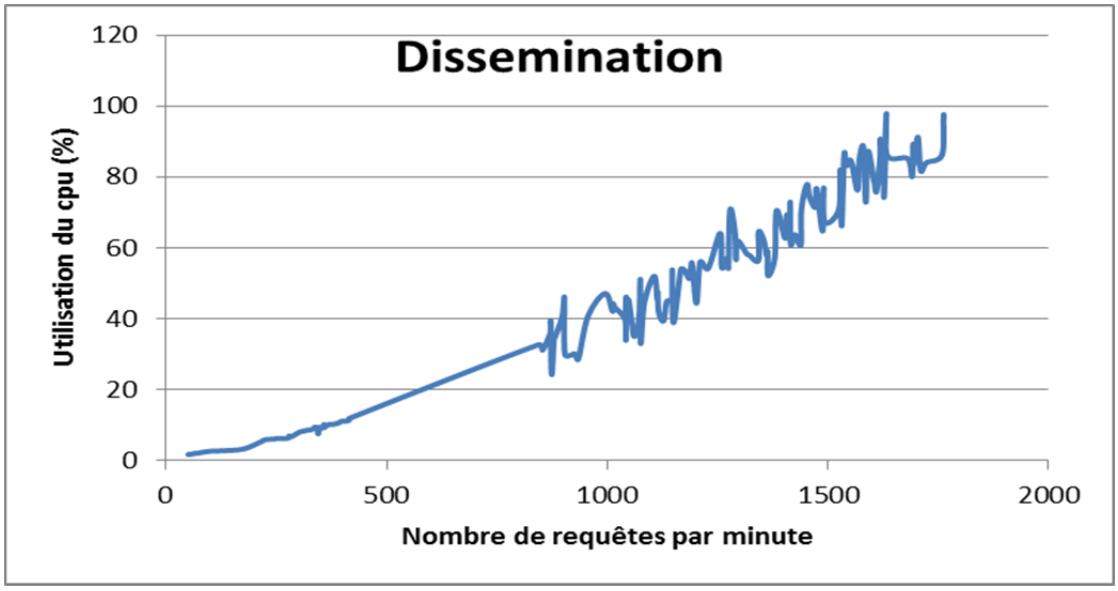

Figure 4-19 Utilisation du CPU du substrate dissemination avec quatre cœurs

Les figures 4-20, 4-21, 4-22 et 4-23 montrent la variation du CPU pour le substrate storage.

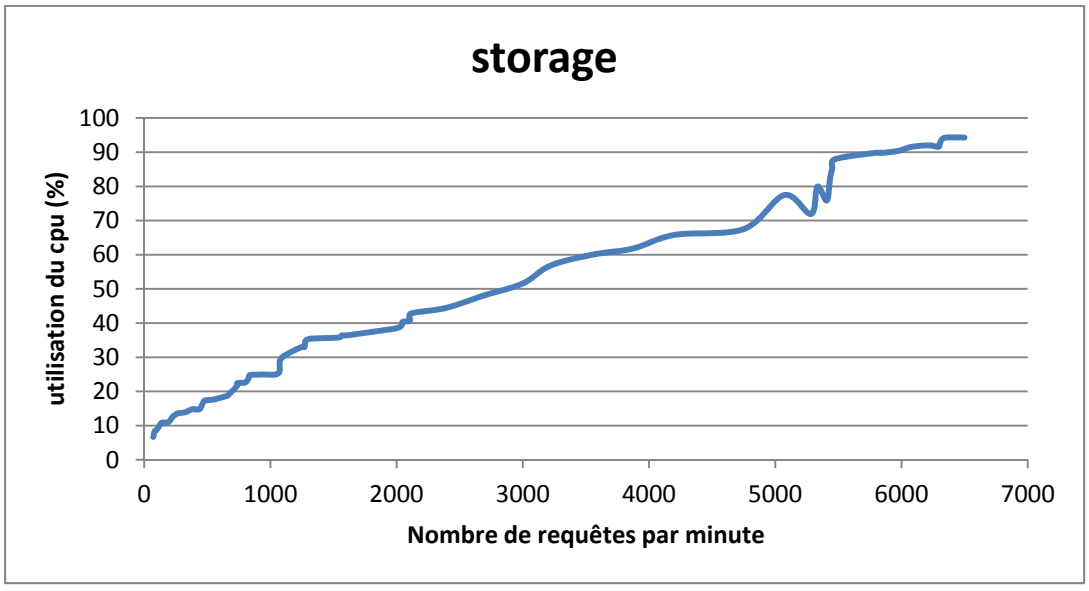

Figure 4-20 Utilisation du CPU de la subtrate storage avec un seul cœur

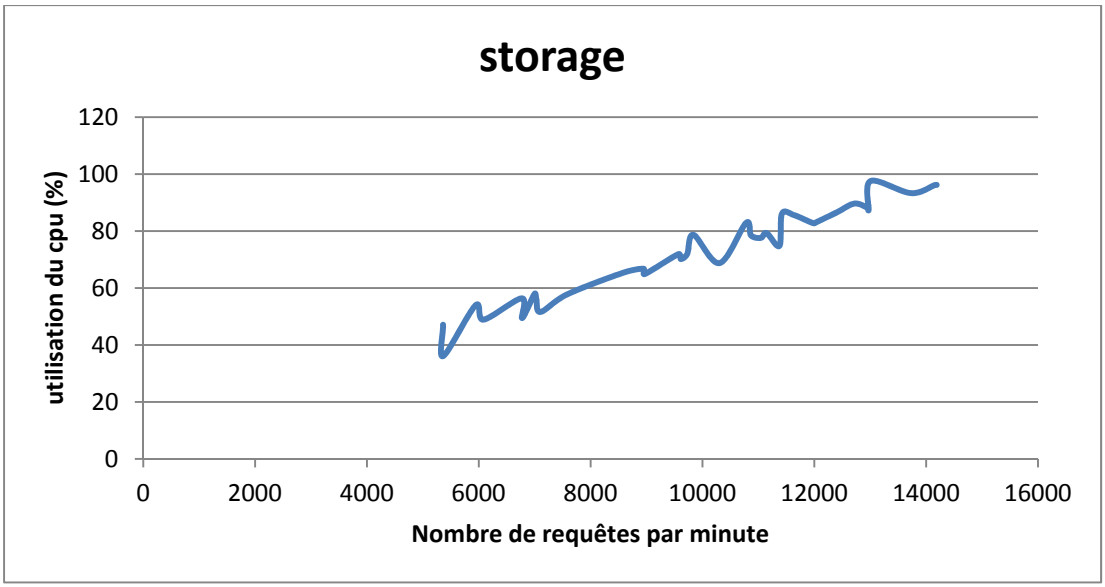

Figure 4-21 Utilisation du substrate storage avec deux cœurs

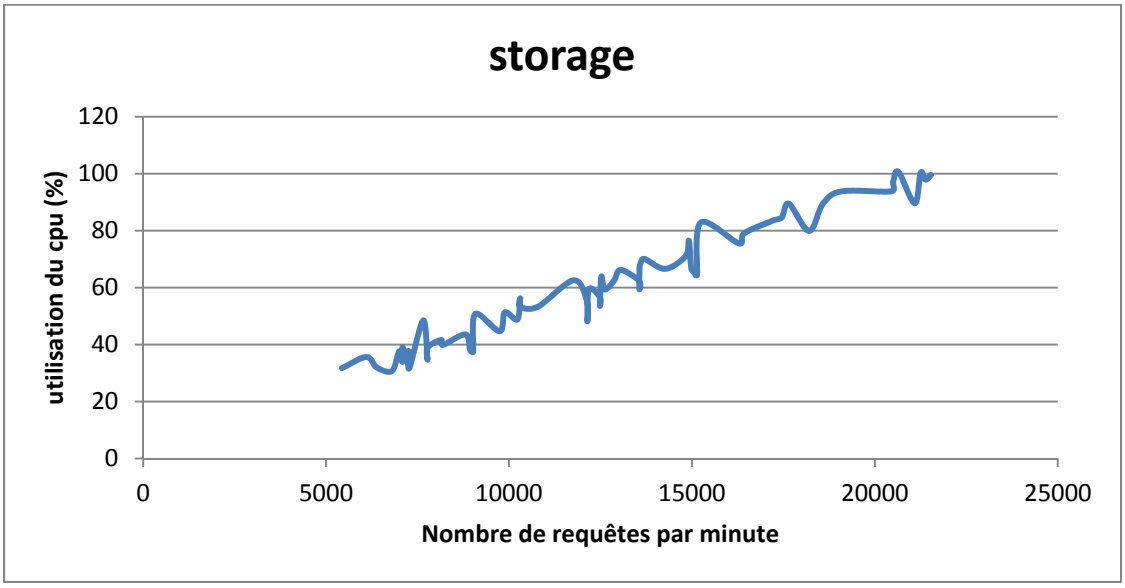

Figure 4-22 Utilisation du substrate storage avec trois cœurs

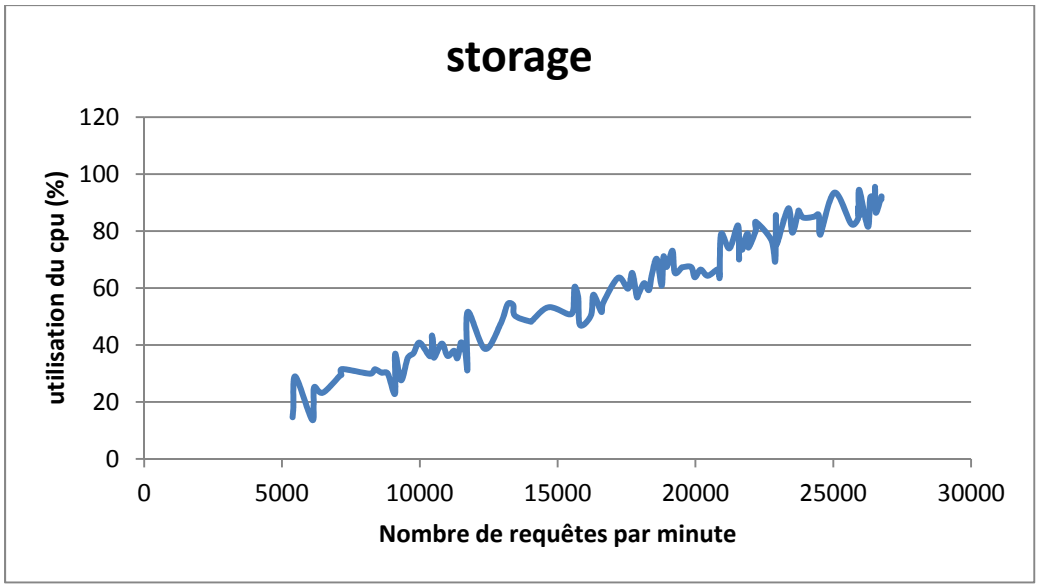

Figure 4-23 Utilisation du substrate storage avec quatre cœurs

Les figures 4-24, 4-25, 4-26 et 4-27 montrent la variation du CPU pour le substrate inference.

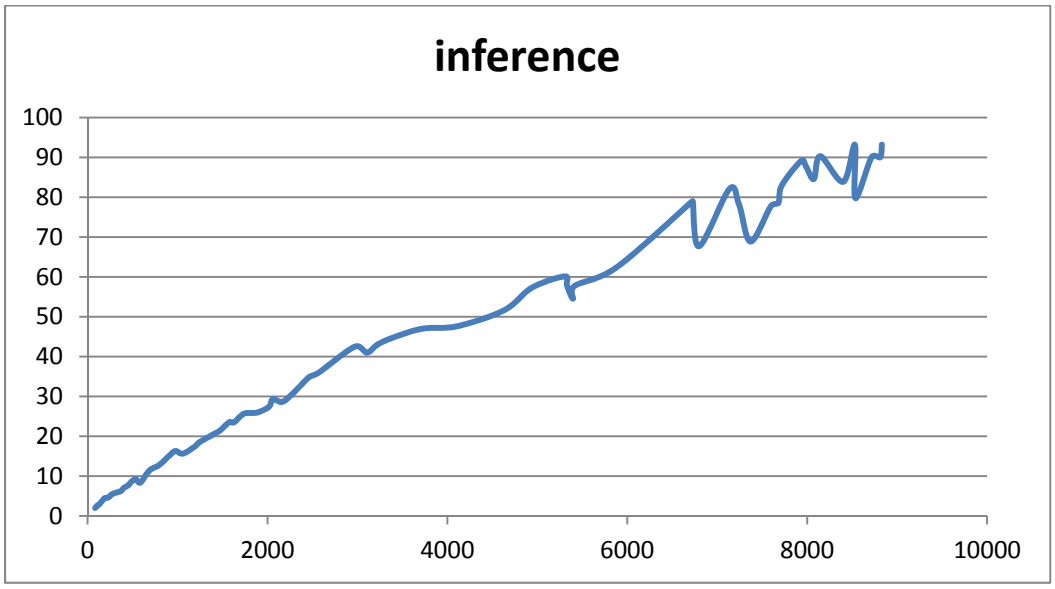

Figure 4-24 Utilisation du CPU du substrate inference avec un seul coeur

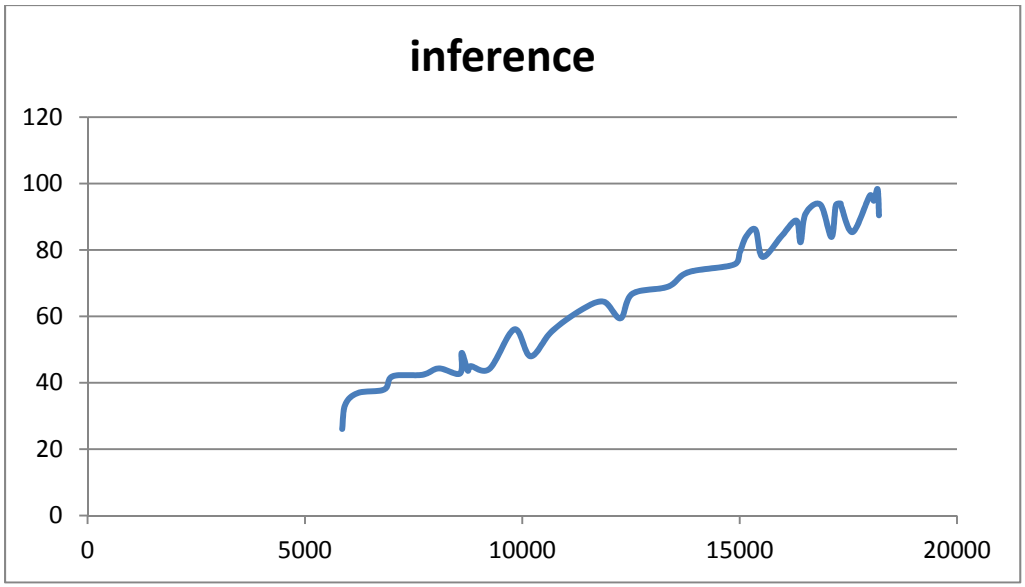

Figure 4-25 Utilisation du CPU du substrate inference avec deux cœurs

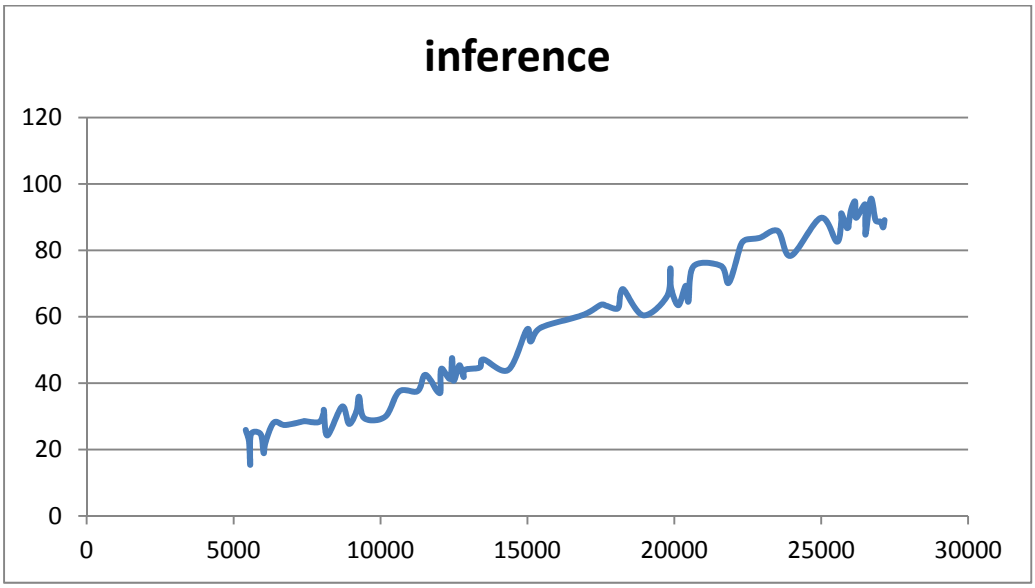

Figure 4-26 Utilisation du CPU du substrate inference avec trois cœurs

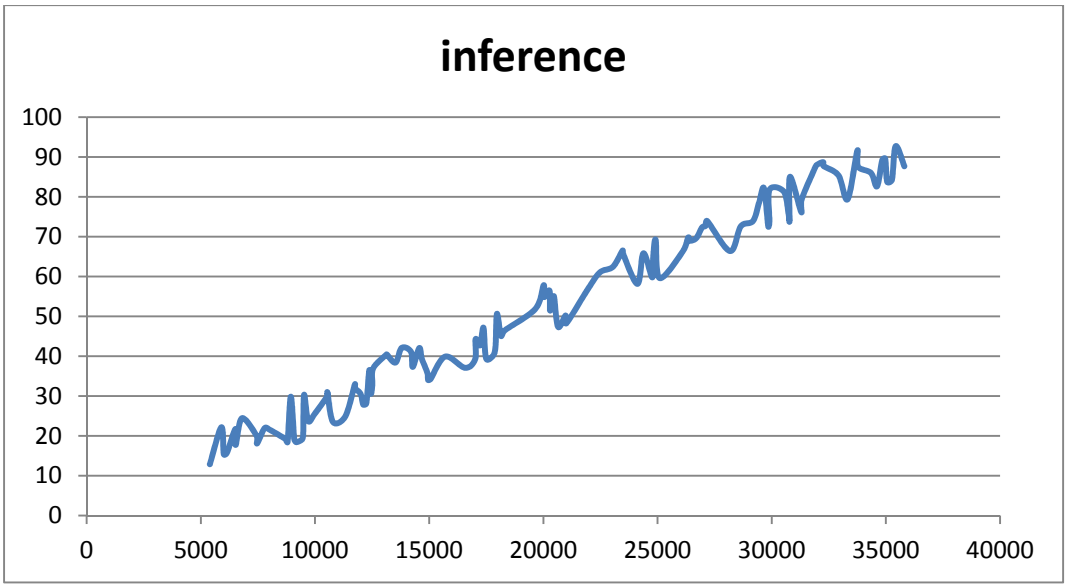

Figure 4-27 Utilisation du substrate inference avec quatre cœurs

La deuxième série de tests consiste à analyser l'impact du changement du nombre de requêtes à traiter sur l'utilisation de la bande passante. On a utilisé la même stratégie que celle utilisée pour la mesure du CPU. La figure 4-28 illustre l'utilisation de la bande passante en fonction du nombre de requêtes par minute du substrate inference. Les figures pour les autres substrates n'ont pas été présentées, car les résultats sont similaires. La courbe en bleu représente le flux entrant et la courbe en rouge représente le flux sortant.

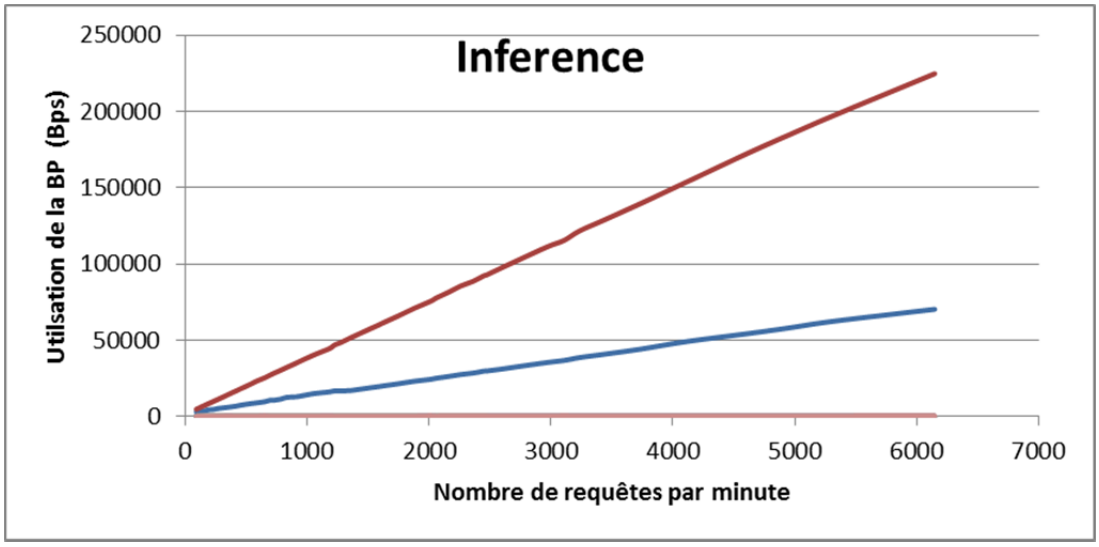

Figure 4-28 Effet de la charge sur le trafic réseau

On remarque que l'utilisation de la bande passante est linéaire par rapport au nombre de requêtes traitées. Il faut noter que dans le cas réel, cette courbe ne sera pas linéaire, car dans ce cas le contenu de la requête donc sa taille n'est pas constant (ce qui est le cas dans notre expérimentation).

La troisième série de tests consistait à analyser la variation de l'utilisation de la mémoire en fonction du nombre de requêtes traitées. Nous avons essayé d'appliquer la même stratégie qu'avec l'utilisation du CPU. Mais malheureusement, on n'a pas réussi. Le premier problème est qu'on ne s'aperçoit pas de l'effet du stress sur la consommation de la mémoire. Par exemple quand la machine est inactive, l'usage moyen de base du CPU est autour de 0,6 % et de la mémoire 9 %. Alors, lorsqu'on envoie une charge spécifique au substrate on remarque une augmentation dans l'utilisation du CPU et reste presque constante durant la période de stress. Après le stress, la valeur de l'utilisation du CPU retourne à la valeur initiale de base. Ce type d'effet nous a permis de dire avec confiance que l'augmentation de l'usage du CPU est due au stress appliqué. Le comportement de la mémoire est différent. Lorsqu'on répète le même test multiple fois, on observe différents comportements. Des fois, il se peut qu'il y ait une augmentation dans l'utilisation de la mémoire puis une stagnation dans cette valeur. Des fois, il n'y a aucune augmentation. Et des fois encore, on remarque l'augmentation puis une

diminution à une valeur intermédiaire différente de la valeur initiale. Avec toutes ces situations, on ne peut pas arriver à une conclusion sur l'effet du stress sur l'utilisation de la mémoire. Mais ce que l'on peut tirer de ces tests est la valeur maximale de l'utilisation de la mémoire dont le substrate a besoin pour un bon fonctionnement. En d'autres termes, nous avons pris la valeur maximale de l'utilisation de la mémoire lorsqu'on collectait les mesures sur l'utilisation du CPU. La table ci-dessous décrit ces valeurs.

Tableau 4-3 Utilisation maximale de la RAM par les différents substrates

Substrate	Memory consumption out of 1Gb(%)
Acquisition	15 %
Modeling	21 %
Dissemination	16 %
Storage	15 %
Inférence	18 %

Les tests réalisés nous ont permis de lister les observations et conclusions suivantes :

- Nous remarquons que l'utilisation du trafic réseau est linéaire et plutôt déterministe. Cet aspect facilite le dimensionnement et la prédiction de cette métrique. Puisque c'est linéaire à la charge, il suffit de multiplier la pente par la charge pour prédire l'utilisation de la bande passante. Nous constatons aussi que la valeur maximale de l'utilisation de la bande passante est de l'ordre de 240Kbps. Cette valeur est très petite comparée à la capacité des cartes réseaux actuelles. On peut dire donc que la bande passante ne pose pas de problème dans le dimensionnement de notre système;
- Nous remarquons que l'utilisation de la mémoire possède un comportement complètement opposé à celui de la bande passante. Les tests sur l'utilisation de la mémoire ne sont pas répétitifs et ne possèdent pas des motifs récurrents. Ceci rend la prédiction et le dimensionnement de cette métrique très difficile. La cause derrière cette irrégularité dans les valeurs mesurées réside dans le fait que le gestionnaire de mémoire du système d'exploitation est imprévisible. Sous l'environnement Linux,

c'est le gestionnaire de mémoire qui décide quand il faut allouer/libérer de la mémoire pour les différentes applications;

- Nous remarquons que les tests sur la consommation du CPU possèdent une allure. Contrairement à la bande passante et à la mémoire, l'utilisation du CPU se trouve au milieu. Elle n'est pas linéaire, mais aussi elle n'est pas imprédictible. Ceci est dû au fait que notre substrate roule dans un environnement Linux qu'on ne peut totalement contrôler. En d'autres termes, même si nous allouons un cœur à notre machine virtuelle, hôte d'un substrate, l'ordonnanceur partagera toujours ce cœur « dédié » avec d'autres processus. Par ailleurs, au niveau de la machine virtuelle, Linux est installé et ajoute un surcout au niveau de l'utilisation du CPU.

Dans le cadre de notre projet, nous proposons de trouver une tendance de l'utilisation du CPU. Tel qu'illustré dans les figures précédentes, le CPU représente un goulot d'étranglement pour une telle plateforme. En effet, il n'est pas possible de dépasser un certain nombre de requêtes par minute que si nous augmentons le nombre de cœurs de la machine virtuelle. Ainsi, pour identifier le besoin en CPU, nous recourons à la régression linéaire pour trouver la tendance de l'utilisation du CPU en fonction de la charge et du nombre de cœurs de la machine virtuelle. Les figures ci-dessous montrent les résultats de l'analyse de la régression linéaire effectuée sur nos modèles les plus précis qu'on a pu trouver pour chaque substrate.

4.4 Dimensionnement de la plateforme de gestion de contexte

4.4.1 Régression linéaire

En statistiques, la régression linéaire est une approche pour la modélisation de la relation entre une variable dépendante y (variable à expliquer) et une ou plusieurs variables explicatives (ou variables indépendantes) notées X. Le cas d'une variable explicative est appelé régression linéaire simple. Pour plus d'une variable explicative, le processus est appelé la régression linéaire multiple.

En régression linéaire, les données sont modélisées à l'aide des fonctions prédictives linéaires, et les paramètres inconnus du modèle sont estimés à partir des données. Ces modèles sont appelés modèles linéaires. Le plus souvent, la régression linéaire se réfère à un modèle dans lequel la moyenne conditionnelle de Y sachant la valeur de X est une fonction affine de X. Comme toutes les formes de l'analyse de régression, la régression linéaire est axée sur la distribution de probabilité conditionnelle de y donnée X, plutôt que sur la probabilité conjointe distribution d'Y et X, qui est le domaine de l'analyse multivariée.

La régression linéaire a été le premier type d'analyse de régression qui a été étudiée rigoureusement, et d'être largement utilisé dans les applications pratiques. Ceci est parce que les modèles qui dépendent linéairement sur leurs paramètres inconnus sont plus faciles à monter que les modèles non linéaires à leurs paramètres et parce que les propriétés statistiques des estimateurs résultants sont plus faciles à déterminer.
La régression linéaire a de nombreuses utilisations pratiques. La plupart des applications tombent dans l'une des deux grandes catégories suivantes :
Si l'objectif est la prédiction, ou de prévision, ou la réduction, la régression linéaire peut être utilisée pour ajuster un modèle prédictif à un ensemble de valeurs Y et X données observées. Après l'élaboration d'un tel modèle, si une valeur supplémentaire de X est alors donné sa valeur sans accompagnement de y, le modèle ajusté peut être utilisé pour faire une prédiction de la valeur de y.

Étant donné une variable y et un certain nombre de variables X1, ..., Xp qui peut être lié à y, une analyse de régression linéaire peut être appliquée pour quantifier la force de la relation entre y et la Xj.

Des modèles de régression linéaires sont souvent montés en utilisant la méthode des moindres carrés, mais ils peuvent aussi être munis d'autres moyens, par exemple en réduisant au minimum le « manque d'ajustement » d'une autre norme (comme avec moins écarts absolus régression), ou en minimisant une autre version de la fonction moins de perte carrés comme dans la régression ridge et lasso. À l'inverse, l'approche des moindres carrés peut

être utilisée pour ajuster les modèles qui ne sont pas des modèles linéaires. Ainsi, bien que les termes « moindres carrés » et « modèle linéaire » sont étroitement liés, mais ne sont pas synonymes.

Nous décidons dans notre projet d'utiliser l'outil d'analyse fourni dans Microsoft Excel pour réaliser la régression linéaire sur les données collectées. Il existe deux indicateurs qui permettent d'évaluer la qualité de la régression linéaire effectuée :

- Coefficient de détermination R2 : C'est un chiffre compris entre 0 et 1 et est considéré le principal indicateur de la régression linéaire. Dans la régression linéaire multiple, nous devons prendre en considération le coefficient de détermination ajusté. Plus la valeur est proche de 1 plus le modèle est conforme aux données mesurées;
- P-value : cet indicateur est relatif à chaque variable indépendante. Il est utilisé pour déterminer quelles sont les variables indépendantes que nous devons garder dans notre modèle de régression linéaire. Si la valeur de P-value est inférieure à 0,05 (nous prenons en général 95% comme seuil de confiance) alors cette variable indépendante est statistiquement significative et donc doit être gardée au niveau du modèle de régression.

Les tableaux 4-4 à 4-13 illustrent les résultats de la régression linéaire pour les différents substrates.

Modeling

Tableau 4-4 Résultats de la régression linéaire pour le subsrate modeling

Regression Statistics	
Multiple R	0,998918
R Square	0,997837
Adjusted R Square	0,99778
Standard Error	0,955864

Observations	231

Tableau 4-5 Modèle de régression linéaire pour le substrate Modeling

Variable	Coefficients	Standard Error	t Stat	P-value
Intercept	-19,882	2,299458	-8,6464	1,03E-15
N	5,037606	0,975868	5,162178	5,38E-07
X	0,036861	0,000984	37,46381	1,97E-98
x*n	0,016952	0,000598	28,32371	5,59E-76
Logx	11,20641	1,05056	10,66708	9,52E-22
Nlogx	-2,81328	0,459668	-6,12023	4,13E-09
Xlogn	-0,14179	0,002055	-69,0027	6,6E-153

Dissemination

Tableau 4-6 Résultats de la régression linéaire pour le subsrate dissemination

Regression Statistics	
Multiple R	0,983587
R Square	0,967443
Adjusted R Square	0,967142
Standard Error	5,078506
Observations	655

Tableau 4-7 Modèle de régression linéaire pour le substrate dissemination

Variable	Coefficients	Standard Error	t Stat	P-value
Intercept	-16,8437	9,870216	-1,70652	0,088391
n	5,859753	3,162581	1,852839	0,06436
X*n	0,006274	0,000948	6,618728	7,6E-11
Log x	1,87E-05	1,59E-06	11,75348	4,76E-29
X²	11,62991	4,547525	2,557416	0,010772
N* log x	-3,92809	1,37895	-2,84861	0,00453
x	0,00182	0,005421	0,335823	0,0737113

Acquisition

Tableau 4-8 Résultats de la régression linéaire pour le subsrate acquisition

Regression Statistics	
Multiple R	0,989702
R Square	0,97951
Adjusted R Square	0,979233
Standard Error	3,195136
Observations	376

Tableau 4-9 Modèle de régression linéaire pour le substrate acquisition

Variable	Coefficients	Standard Error	t Stat	P-value
Intercept	6,864277	1,703491	4,029536	6,79E-05
X	0,031059	0,000589	52,76762	2,9E-174
N	-8,40246	4,074058	-2,06243	0,039864
x*n	0,008972	0,000396	22,63646	8E-72
N* logx	1,883181	1,180178	1,595675	0,111415
X * log(n)	-0,09402	0,00277	-33,9439	1,1E-115

Storage

Tableau 4-10 Résultats de la régression linéaire pour le subsrate storage

Regression Statistics	
Multiple R	0,98484902
R Square	0,9699276
Adjusted R Square	0,96948698
Standard Error	4,2863772
Observations	278

Tableau 4-11 Modèle de régression linéaire pour le substrate storage

	Coefficients	Standard Error	t Stat	P-value
Constante	-5,13693889	4,91722335	-1,04468285	0,297093711
x*n	0,00011603	2,9139E-05	3,9819351	8,77107E-05
Log x	8,10077716	2,05704282	3,93806928	0,000104345
N* log x	-1,54341225	0,22976253	-6,71742365	1,07227E-10
X * log n	0,01130716	0,00045669	24,7591925	9,39864E-72

Inférence

Tableau 4-12 Résultats de la régression linéaire pour le subsrate inference

Regression Statistics	
Multiple R	0,989962
R Square	0,980026
Adjusted R Square	0,979764
Standard Error	3,842786
Observations	311

Tableau 4-13 Modèle de régression linéaire pour le substrate inference

	Coefficients	**Standard Error**	**t Stat**	**P-value**
Constante	-4,84674255	3,05991785	-1,58394531	0,114242461
X	-0,00024234	0,00020914	-1,15872958	0,047472861
n	-3,2601085	0,59939683	-5,4389819	1,10056E-07
X*n	8,217E-05	3,8992E-05	2,10733712	0,035904443
Log x	4,83299065	1,26934889	3,80745649	0,000169729
x * logn	0,00943858	0,00035673	26,4583938	5,75016E-81

Les résultats de la régression linéaire sont satisfaisants puisque le R2 ajusté est très proche de 1 et le P-value est inférieure à 0.05 pour les différents substrates.

Les figures de 4-29à 4-48 illustrent les courbes comparatives des valeurs mesurées et des valeurs prédites de l'utilisation du CPU en fonction du nombre de requêtes par minute et du nombre de cœurs pour les différents substrates.

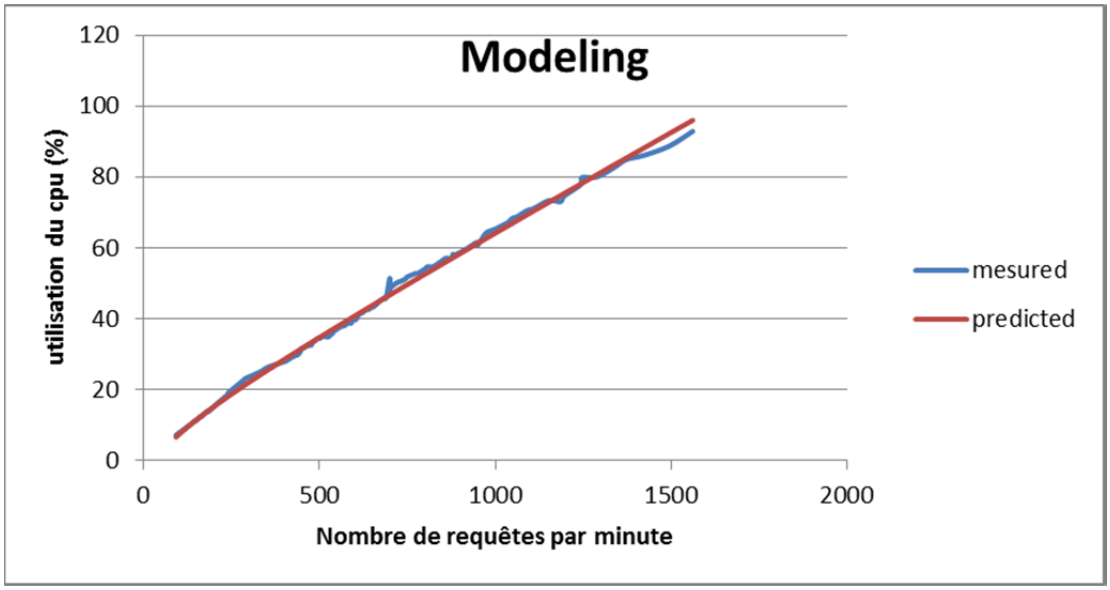

Figure 4-29 Utilisation du CPU du substrate modeling avec un seul cœur

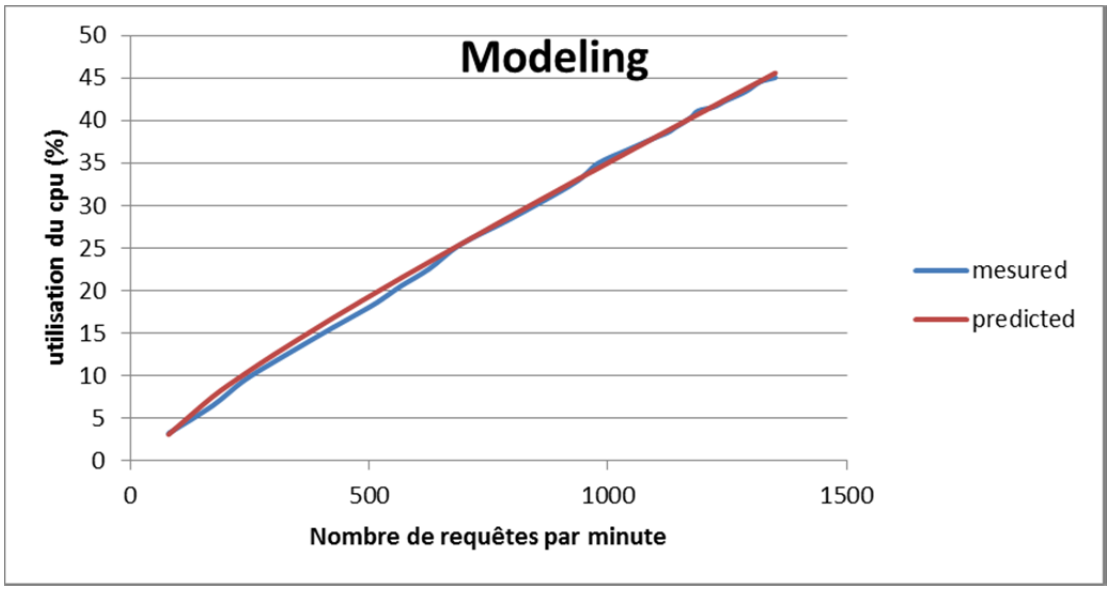

Figure 4-30 Utilisation du CPU du substrate modeling avec deux cœurs

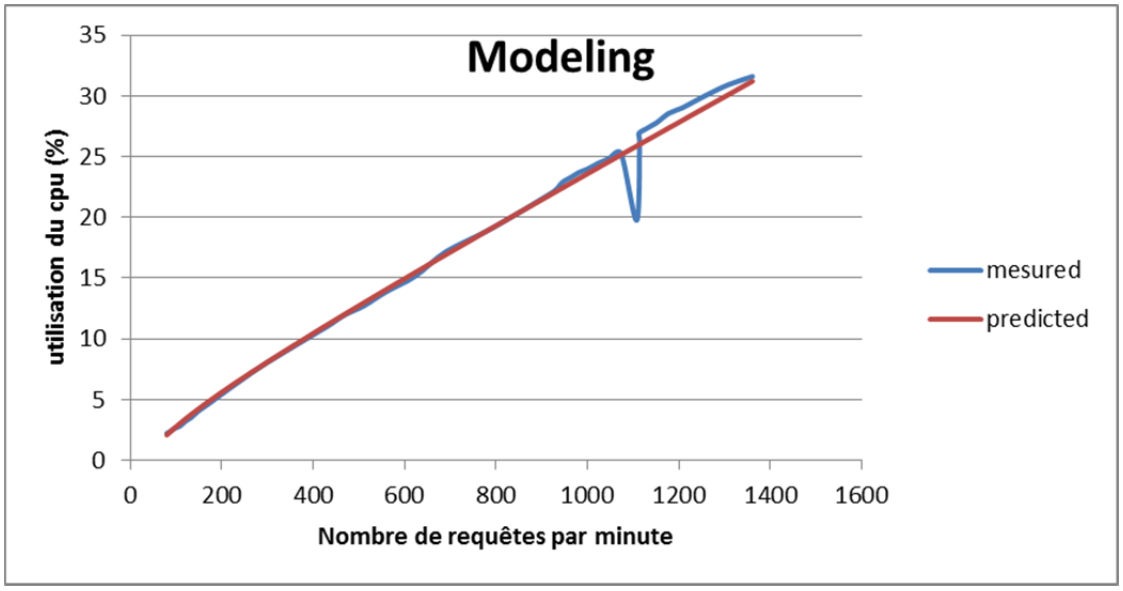

Figure 4-31 Utilisation du CPU du substrate modeling avec trois cœurs

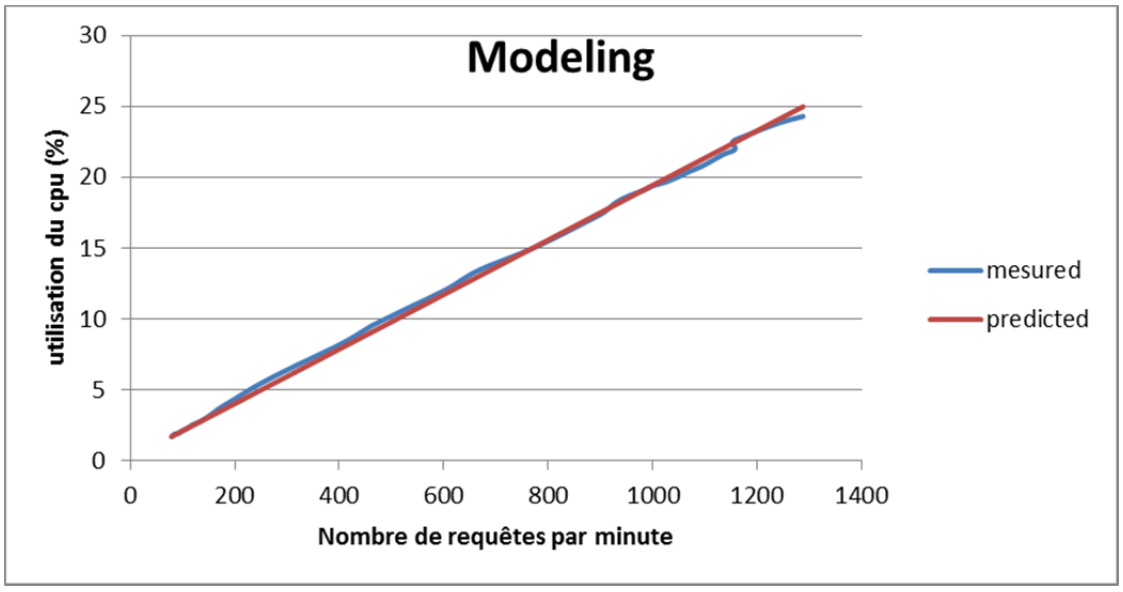

Figure 4-32 Utilisation du CPU du substrate modeling avec quatre cœurs

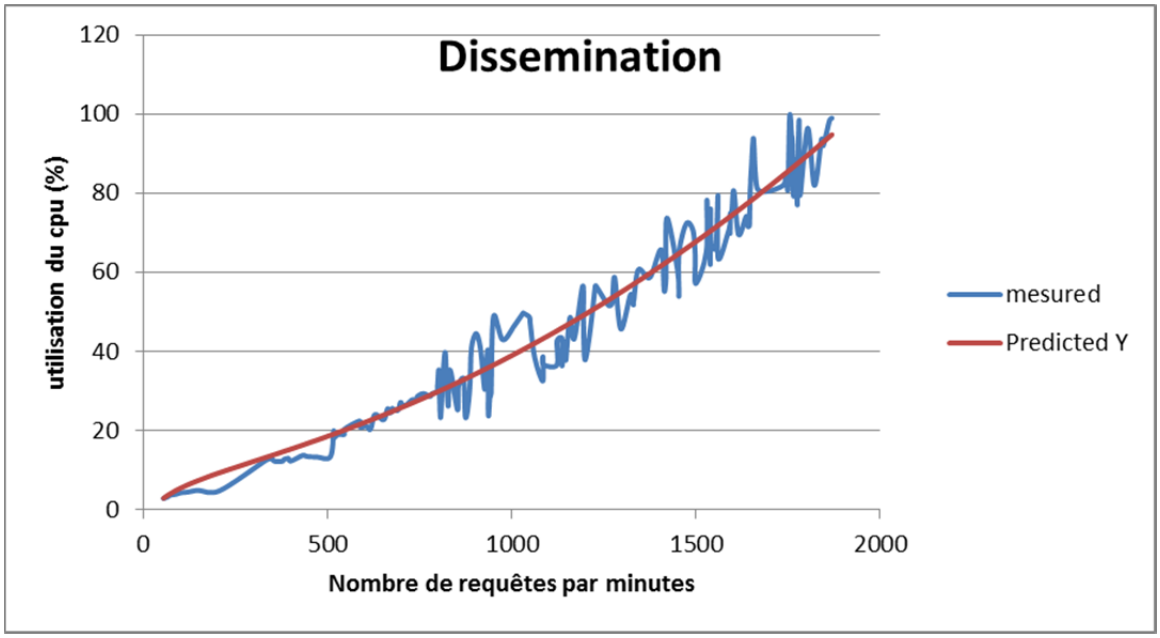

Figure 4-33 Utilisation du CPU du substrate dissemination avec un seul cœur

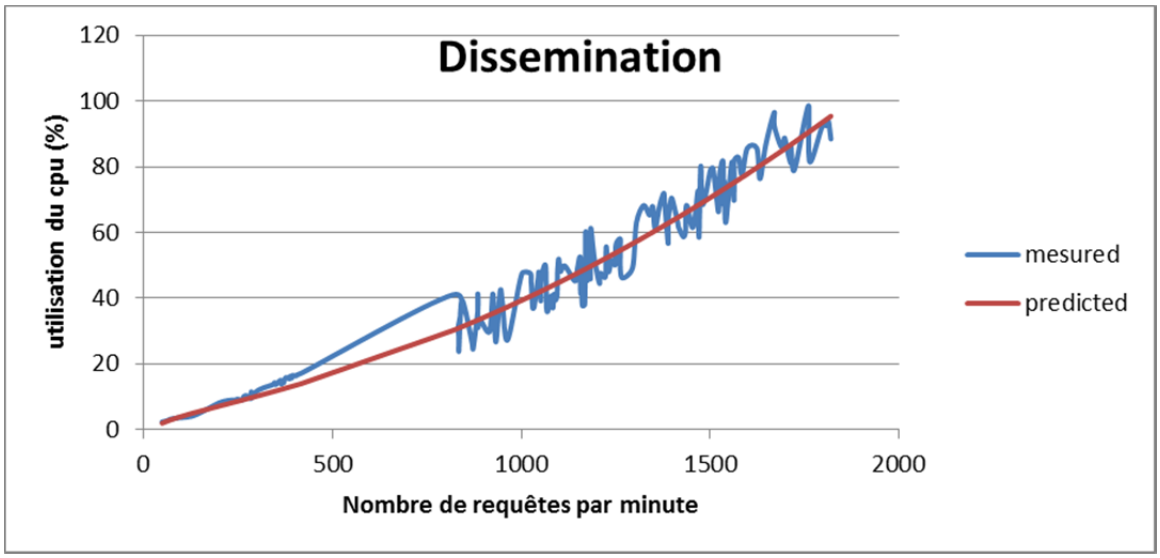

Figure 4-34 Utilisation du CPU du substrate dissemination avec deux cœurs

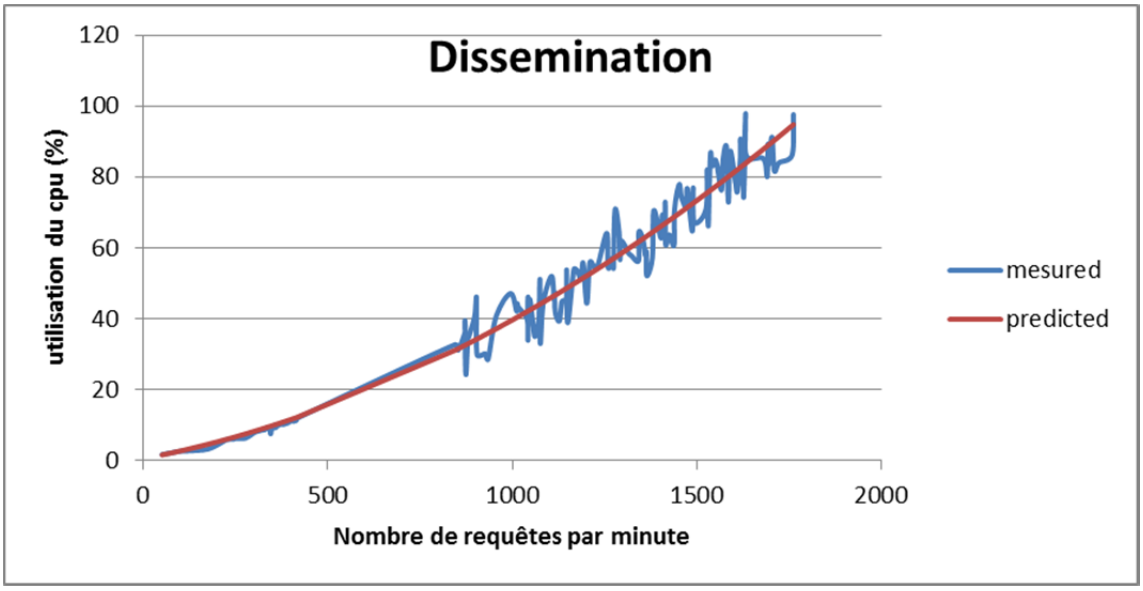

Figure 4-35 Utilisation du CPU du substrate dissemination avec trois cœurs

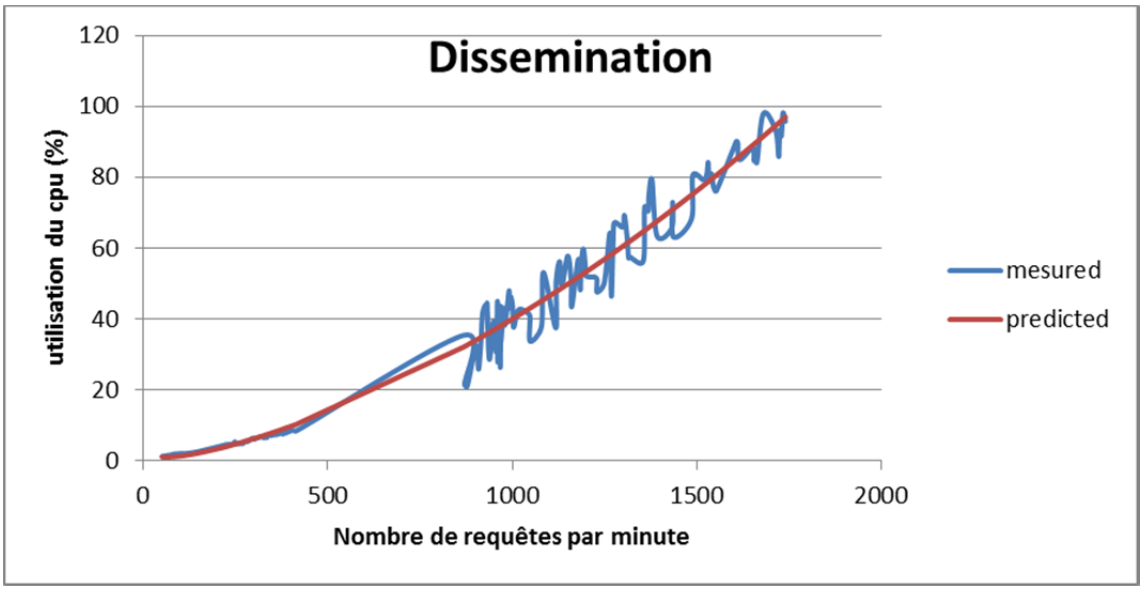

Figure 4-36 Utilisation du CPU du substrate dissemination avec quatre cœurs

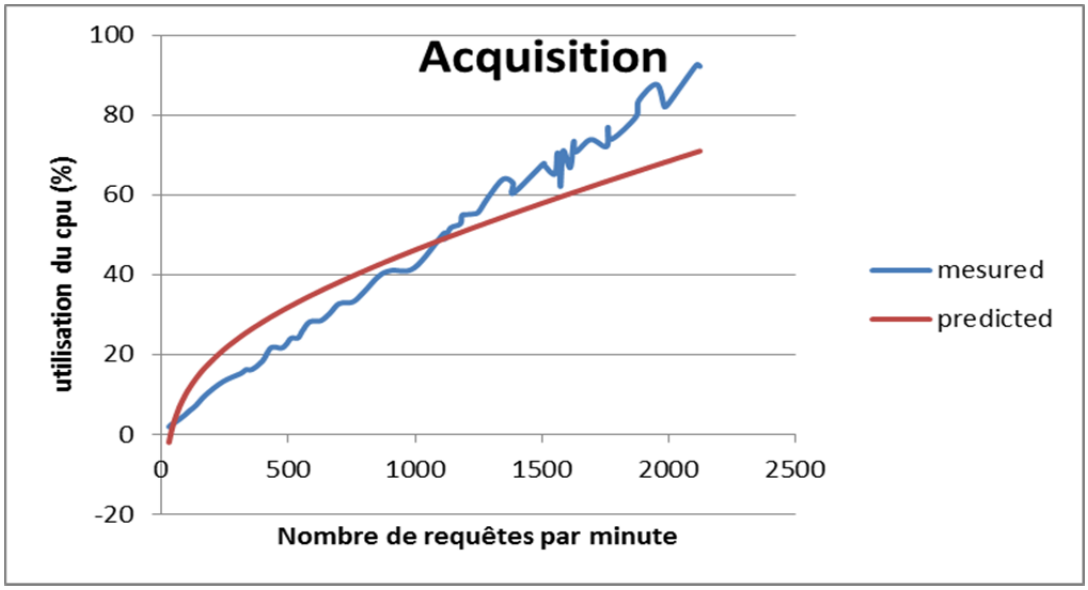

Figure 4-37 Utilisation du CPU du substrate acquisition avec un seul cœur

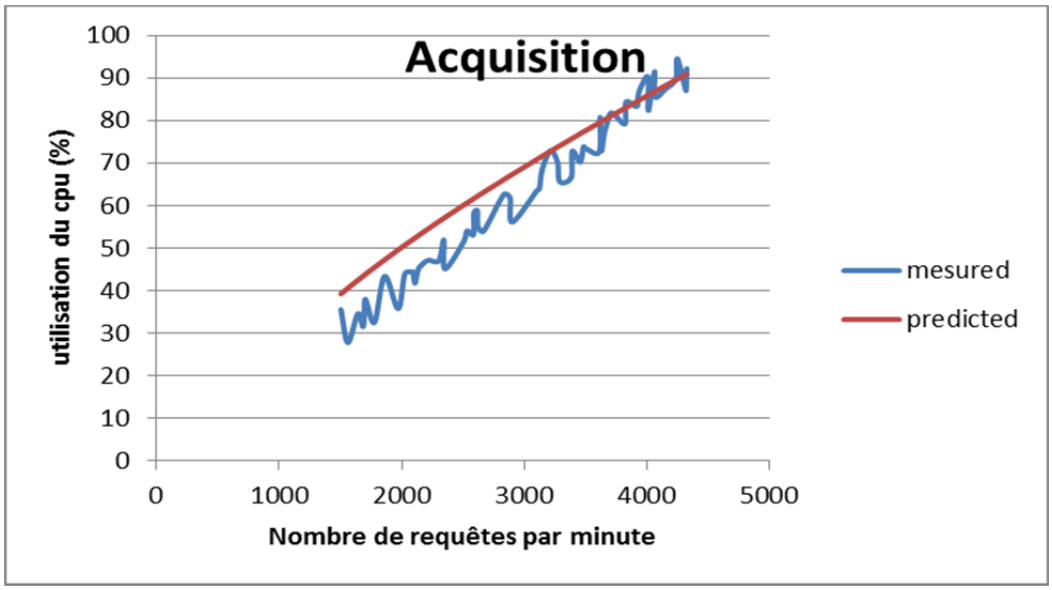

Figure 4-38 Utilisation du CPU du substrate acquisition avec deux cœurs

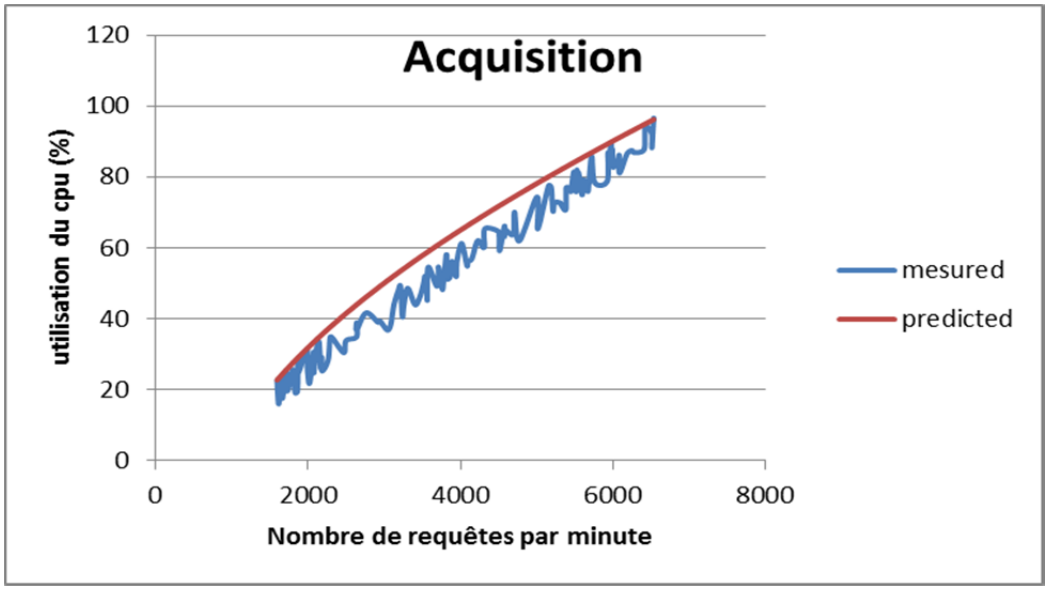

Figure 4-39 Utilisation du CPU du substrate acquisition avec trois cœurs

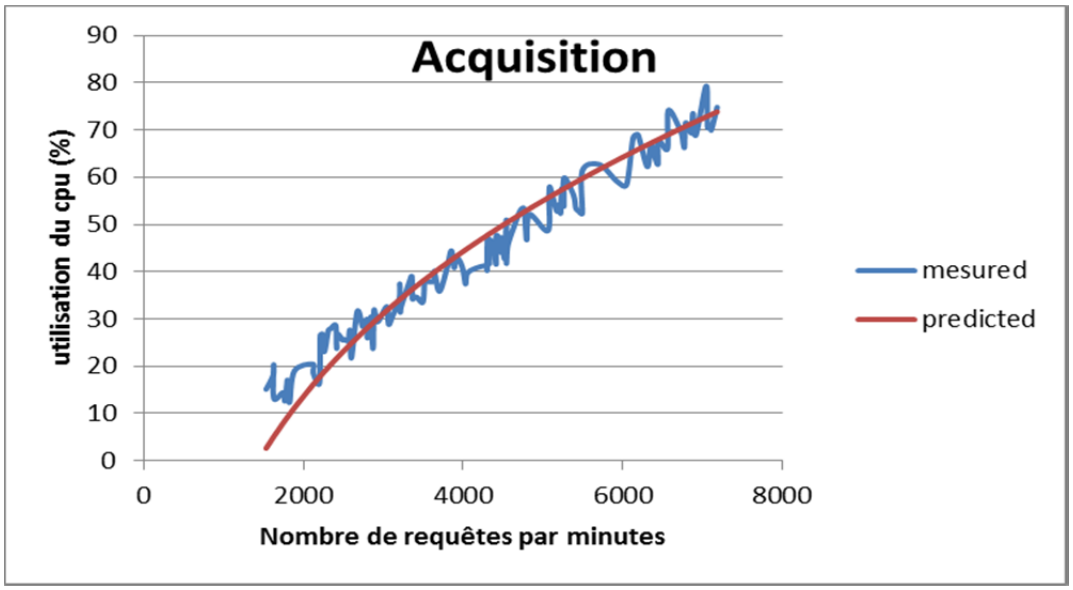

Figure 4-40 Utilisation du CPU du substrate acquisition avec quatre cœurs

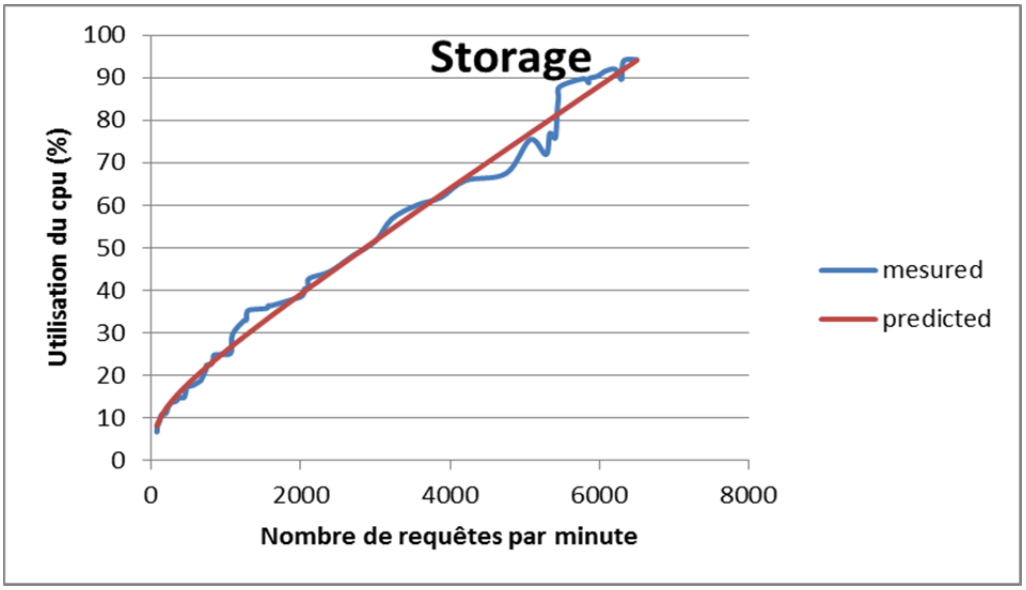

Figure 4-41 Utilisation du CPU du substrate storage avec un seul cœur

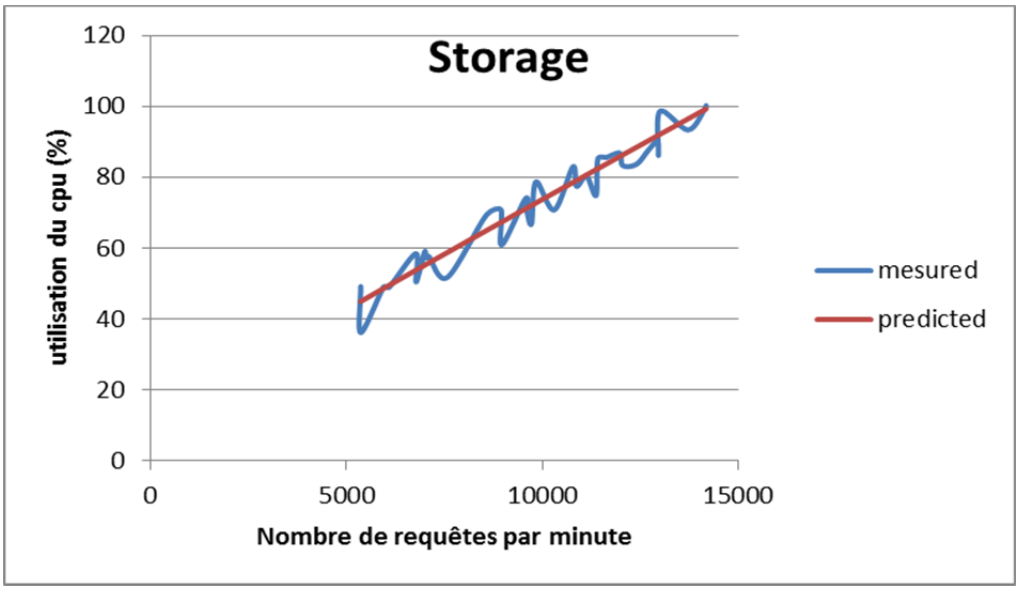

Figure 4-42 Utilisation du CPU du substrate storage avec deux cœurs

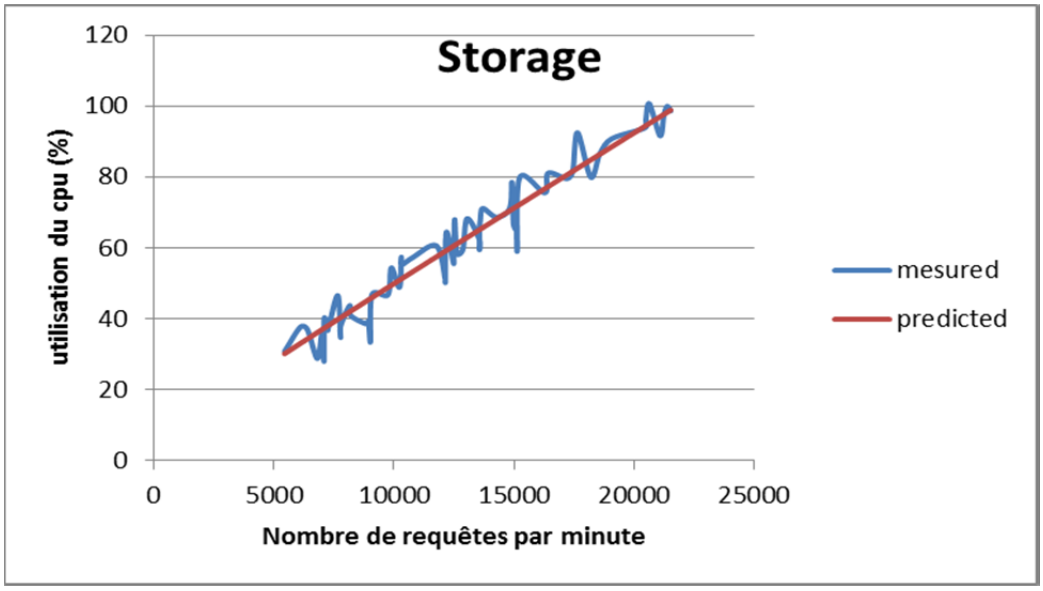

Figure 4-43 Utilisation du CPU du substrate storage avec trois cœurs

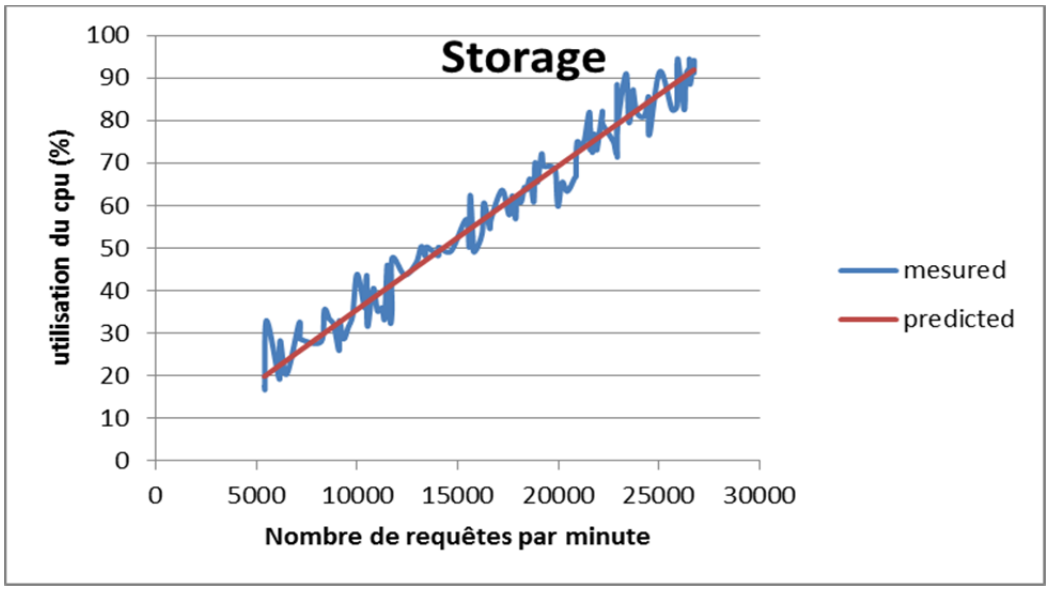

Figure 4-44 Utilisation du CPU du substrate storage avec quatre cœurs

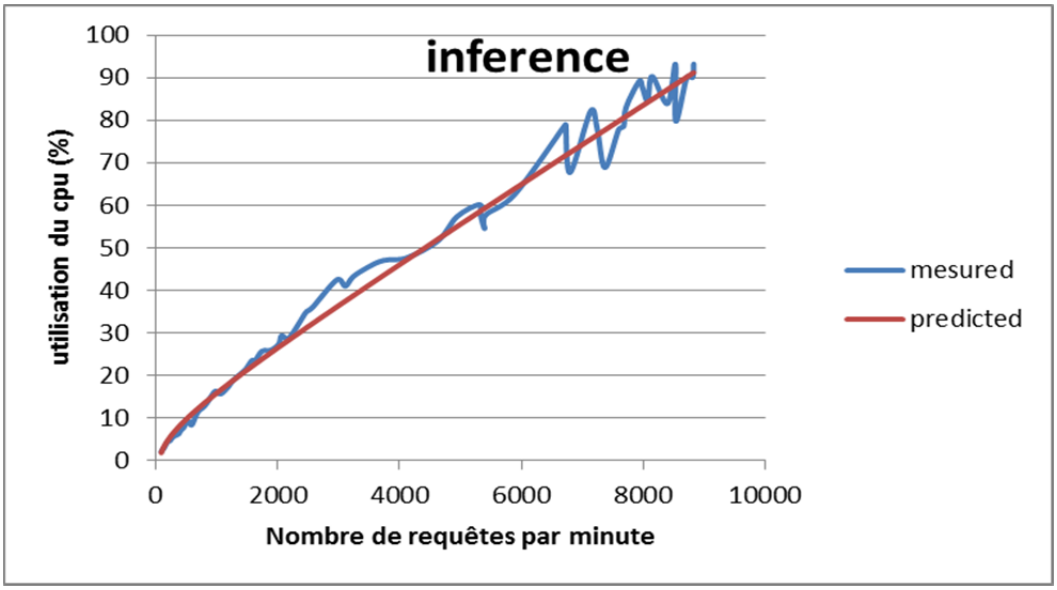

Figure 4-45 Utilisation du CPU du substrate inference avec un seul cœur

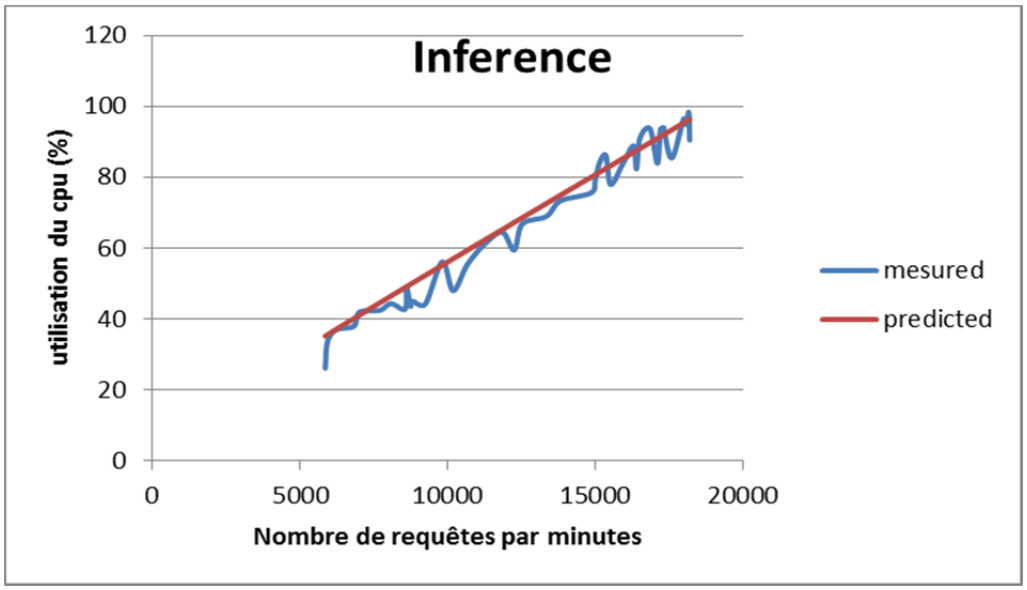

Figure 4-46 Utilisation du CPU du substrate inference avec deux cœurs

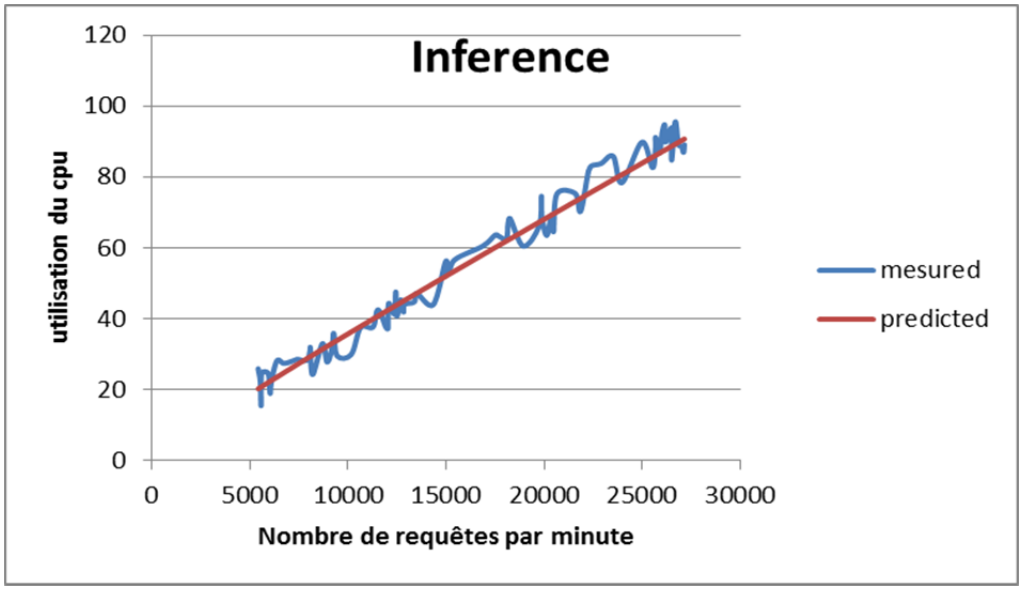

Figure 4-47 Utilisation du CPU du substrate inference avec trois cœurs

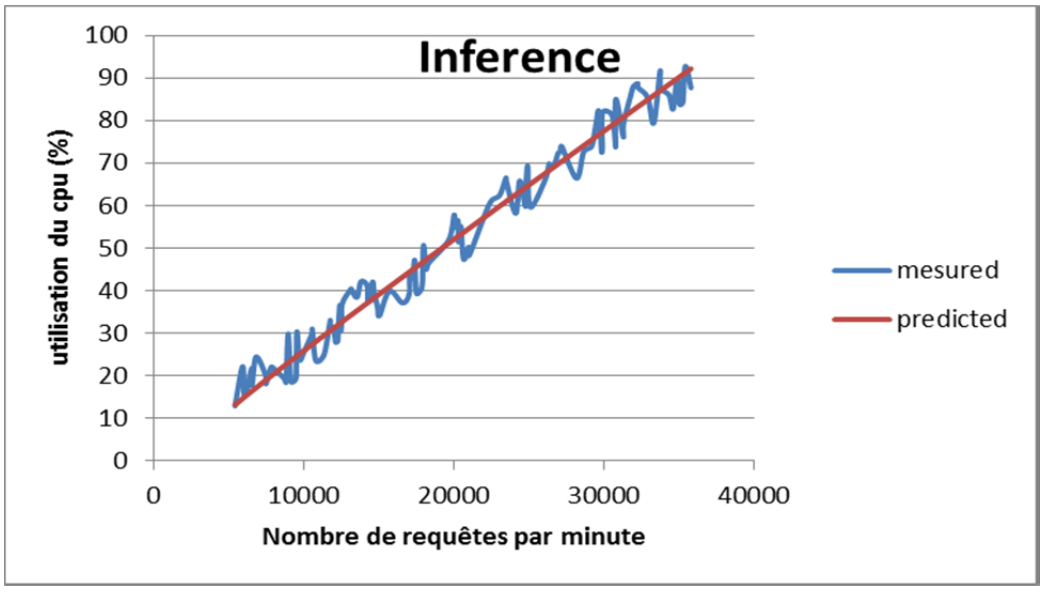

Figure 4-48 Utilisation du CPU du substrate inference avec quatre cœurs

4.4.2 Méthodologie de dimensionnement

En nous basant sur les résultats obtenus aux sections précédentes, nous proposons une méthodologie de dimensionnement de substrates pour les fournisseurs d'infrastructures.

Avant de publier le substrate au niveau du Courtier, le fournisseur d'infrastructure devra suivre les étapes suivantes pour dimensionner son substrate :

- Posséder/installer un système de monitorage qui récupère les valeurs de la métrique à dimensionner et les différentes valeurs des variables qu'il considère comme variables explicatives de la métrique. . Dans notre cas par exemple, on s'est intéressé à la consommation du CPU comme métrique et comme variables explicatives on a pris la charge que doit traiter le substrate et le nombre de cœurs alloués au substrate;
- Effectuer différents tests sur le substrate et récupérer les données de la métrique et des variables explicatives. Dans ces tests on doit faire varier au maximum les variables explicatives pour couvrir le plus grand intervalle de valeurs possibles. Le nombre de tests à effectuer dépend du fournisseur d'infrastructure. Mais plus ce nombre est grand plus c'est bénéfique pour le dimensionnement;
- Effectuer une régression linéaire sur la métrique à dimensionner en fonction d'une combinaison des variables explicatives. Le résultat de la régression linéaire pourra nous dire à quel point le modèle considère est précis à l'aide de l'indicateur R^2 multiple. Le choix du modèle vient en général d'une intuition auprès de celui qui propose le modèle de régression. Mais pour un fournisseur non expérimenté, il suffit de réaliser la régression linéaire avec plusieurs modèles et choisir après le plus précis d'entre eux;
- La dernière étape consiste à implémenter la fonction modèle au niveau de l'infrastructure comme service.

Le modèle est une fonction mathématique de la métrique à dimensionner en fonction des variables explicative (ex. nombre de requêtes par minute et nombre de cœurs). Ce modèle peut être implémenté en deux niveaux. Le premier lors du déploiement du service de gestion de contexte. Dans cette étape le modèle aidera à trouver les valeurs des variables explicatives de base pour un bon fonctionnement du service. Le deuxième niveau est lors de la gestion d'élasticité du substrate. Le modèle dans ce cas permettra à la prise de décision dans l'augmentation ou la diminution des ressources allouées au substrate.

Plus spécifiquement, nous avons défini comme métrique la consommation du CPU et comme variable explicative : la charge et le nombre de cœurs. Ce que nous aimerions savoir est qu'elle est le nombre de cœurs que nous devons allouer à ce substrate? Pour déterminer le nombre de cœurs, nous considérons la règle suivante : si la consommation du CPU est au-dessus de 70 % alors les ressources sont insuffisantes. Si à l'instant t, nous relevons une consommation du CPU C supérieure à 70%,et nous notons que le nombre de cœurs actuellement alloués est N et que la charge traitée par le substrate L. alors, nous calculons la consommation C(N+1,L). Cette consommation est obtenue grâce à la régression linéaire, si la valeur relevée est inférieure à 70 % alors on ajoute un cœur à notre substrate sinon nous incrémentons le N jusqu'à atteindre une consommation inférieure à 70%. Une fois atteinte, nous allouons le nombre N de cœurs à notre substrate correspondant à ce nouveau seuil. De la même manière, nous pouvons procéder pour déterminer si nous devons diminuer le nombre de cœurs alloués. L'algorithme 1 décrit la procédure de gestion dynamique du CPU.

Algorithme 1 : Gestion dynamique de la consommation du CPU

```
1  SET threshold
2  SET subtrateID
3  PROCEDURE cpu_management
4  N <- getNumberCores
5  C <- getCpuUsage
6  L <- getLoad
7  IF C GREATER THAN threshold THEN
8      WHILE C GREATER THAN  OR EQUAL threshold
9          INCREMENT N
10         C <- CALL  model_function WITH N AND L
11     ENDWHILE
12 Else
13     While C LESS THAN threshold
14         DECREMENT N
15         C <- CALL  model_function WITH N AND L
16     ENDWHILE
17     INCREMENT N
18 END IF
19 CALL Allocate WITH N AND subtrateID
```

Les lignes 1 et 2 permettent d'initialiser deux variables : 1) la variable « threshold » qui détermine le seuil sur lequel nous allons nous baser pour prendre une décision. La deuxième variable « substrateId » qui permet d'identifier la machine virtuelle du substrate. À la ligne 3, nous déclarons notre fonction principale. Les lignes 4,5 et 6 nous permettent de récupérer respectivement la valeur actuelle du nombre de cœurs alloués à la machine virtuelle du substrate, l'utilisation actuelle du CPU et la charge traitée par le substrate. À la ligne 7, nous testons si l'utilisation du CPU est supérieure au seuil. Si oui, alors nous exécutons le code des lignes 8 à 11. Tant que la consommation du CPU est supérieure ou égale au seuil (ligne 8) nous incrémentons le nombre de cœurs N par 1 (ligne 9) et nous calculons la nouvelle valeur de la consommation en utilisant le modèle obtenu et nous l'ajoutons à la consommation du CPU C (ligne 9)… Sinon nous exécutons le code des lignes 13 à 18. Tant que la consommation du CPU C est inférieure au seuil (ligne 13), nous décrémentons le nombre de cœurs N par 1 (ligne 14) et nous ajoutons à C la prédiction du modèle avec le nouveau N

(ligne 15). À la sortie de la boucle (ligne 17), nous incrémentons le nombre de cœurs par 1. Enfin, nous allouons N cœurs à la machine virtuelle de notre substrate identifiée par substrateId (ligne 19).

La figure 4-49 décrit une implémentation en python de l'algorithme proposé

```
1  import threading
2  import math
3
4  ## function that allows to manage the number of cores of a virtual machine each
5  ## "period" seconds.
6  def cpu_management(id,period,threshold) :
7      N= getNumberCores()
8      C= getCpuUsage()
9      L= getLoad()
10     If C>threshold:
11         While C>=threshold:
12             N=N+1
13             C= model(N,L)
14     Else :
15         While C < threshold:
16             N=N-1
17             C =  model(N,L)
18         N=N+1
19     ## function that allocates N cores to the virtual machine identified by id
20     Allocate(N, id)
21     threading.timer(cpu_management,period,[id,period,threshold]).start()
22
23 def model(N,X):
24     return -19,882 +(0,037+0,017*N -  0,142 math.log(N))*X + (11,206  - 2,813*N)*math.log(X) + 5,038*N
25
26 ##calling the function
27 cpu_management("modeling",30,0.7)
```

Figure 4-49 Implementation en Python de l'algorithme de gestion du CPU

Ce code peut être exécuté au niveau de l'hôte du substrate et permettra donc de gérer de manière dynamique le nombre de cœurs alloués au substrate.

4.5 Conclusion

Pour résumer, le prototype développé a permis de valider les différentes composantes de l'architecture proposée de la plateforme. De plus, les tests effectués ont permis de connaitre les limites de notre système et aussi de trouver et proposer une stratégie pour dimensionner les différents substrates.

CONCLUSION

Pour résumer, les principales contributions de ce projet sont, premièrement, la séparation des systèmes classiques CA en modules fonctionnels nommés substrates. En réalisant cette séparation, nous montrons que l'on peut concevoir des systèmes CA modulaires et distribués. À l'opposé des systèmes classiques qui sont des systèmes monolithique et centralisé. Les substrates offrent aussi l'avantage de la réutilisabilité et la maintenabilité de ceux-ci.

La deuxième contribution est la proposition de l'architecture d'une plateforme de gestion de contexte dans le Nuage. Cette plateforme permet la composition et le déploiement dynamique de services CA tout en offrant des interfaces ouvertes et standards.

La troisième contribution est la validation de ces substrates et l'architecture à travers le prototypage. En effet, nous avons réussi à implémenter cinq principaux substrates des systèmes CA à savoir : acquisition, dissemination, modeling, storage et inference. Nous avons aussi fourni une implémentation de la plateforme de gestion de contexte et l'avons déployé dans un environnement virtuel. La plateforme développée permet la découverte et la publication des ressources qui sont les capteurs et les substrates. Elle permet aussi la composition de ces ressources et leur déploiement pour créer des services CA. Finalement, elle permet d'exécuter les services déployés.

La quatrième et dernière contribution de ce mémoire est la proposition d'une méthodologie de dimensionnement des substrates. Nous avons réussi à atteindre ce résultat à l'aide des différents tests de stress effectués sur les différents substrates. Les tests de stress nous ont permis d'analyser les performances de la plateforme et des différents substrates et surtout de connaitre leurs limitations. La méthodologie proposée peut être utilisée par les différents fournisseurs d'infrastructure pour déterminer les besoins en termes de ressources physiques (CPU, mémoire, bande passante…) des substrates choisies par les clients lors d'une composition d'un service de gestion de contexte. Un dimensionnement adéquat permet une bonne gestion et une optimisation des ressources des fournisseurs d'infrastructures. Aussi,

nous avons proposé un algorithme qui permet la gestion dynamique et en temps réel de ces ressources (le CPU dans notre cas). Cet algorithme peut être implémenté au niveau de la plateforme pour gérer l'élasticité des susbtrates selon le besoin du client.

Autant que nous sachions, ce travail présente une première initiative de la virtualisation des systèmes CA. Les travaux de recherches dans ce domaine sont encore à leur début. Nous avons démontré qu'il est possible de virtualiser les systèmes CA. Nous avons abordé cette problématique de recherche en traitant les aspects conception d'une telle architecture distribuée et la gestion des ressources dans ce type d'infrastructures. Cette étude nous a permis d'identifier plusieurs problématiques de recherche que nous proposons comme travaux futurs. Parmi ceux-ci, nous pouvons citer l'amélioration de la plateforme de gestion de contexte en proposant des options plus détaillées sur la manière de composition des ressources et aussi ajouter des validations sur la compatibilité des ressources qui composent le service. Un autre onglet de recherche serait de se concentrer sur l'implémentation des substrates. Ainsi, proposer une substrate modeling basée sur des modèles ontologiques apporterait de la richesse au substrate. De même, on peut améliorer le substrate inference en intégrant un moteur d'inférence avec des règles d'inférences étudiées minutieusement intrinsèque au domaine d'application. Ce qui nous ramène à étudier aussi d'autres domaines d'application comme le « pervasive gaming » de les modéliser et de trouver les différentes règles d'inférence qui y peuvent s'appliquer.

ANNEXE I

INSTALLATION DE ZABBIX SUR UBUNTU 12.04 LTS

Zabbix est un outil de monitorage open source. Il permet de mesurer une grande variété de données ou métriques reliées au réseau ou aux applications. Il utilise une base de données pour stocker les données et son backend est écrit en C. Zabbix possède une architecture client-serveur. Dans ce qui suit, nous décrirons comment installer cet outil sur une machine Linux Ubuntu.

Installation de Zabbix-server

Le serveur Zabbix peut être installé sur une machine virtuelle Ubuntu en insérant les commandes suivantes :

- Sudo apt-get install zabbix-server-mysql
- Sudo apt-get install zabbix-frontend-php

Pour accéder à l'interface graphique de Zabbix, il suffit à partir d'un navigateur d'entrer l'URL suivante :

- http://<zabbix-server-ip-address>/zabbix

une authentification est demandée alors par l'interface. Par défaut, un super utilisateur est fourni « Admin ». Son mot de passe est « zabbix ».

On peut après ajouter des utilisateurs avec différents privilèges à travers l'interface graphique.

Installation de Zabbix-agent

L'agent de Zabbix est la partie cliente de Zabbix. Il est installé sur la machine cible du monitorage. Pour installer l'agent, il suffit de taper la ligne de commande suivante au niveau de la machine client.

- Sudo apt-get install zabbix-agent

Ensuite, il faut configurer cet agent. Taper :

- Sudo vi /etc/zabbix/zabbix_agentd.conf

Localiser les deux lignes qui contiennent Server= et Hostname=. Si elles sont commentées par #, enlever ces commentaires et modifier :

- Server=<Zabbix-server-ip-address>
- Hostname=<zabbix-agent-hostname>

Enfin, redémarrer le service pour prendre en compte la nouvelle configuration :

- Sudo /etc/init.d/zabbix_agent restart

Une fois les deux parties installées, il reste juste à les connecter. Sur l'interface graphique, allez vers l'onglet « configuration » puis « hosts » puis « create host ».
Un formulaire s'affiche. Les informations essentielles à remplir sont :

- L'adresse IP du client
- Le nom du client (celui rempli dans le fichier de configuration de l'agent)
- ajouter un Template. Dans notre cas, nous avons ajouté Linux Template, car on utilise des machines clientes Ubuntu.
- Affecter à un groupe (facultatif) pour bien organiser votre monitorage.

Pour visualiser vos données, dirigez-vous vers l'onglet « monitoring » =>
« latest data », puis sélectionner votre client et la métrique à afficher.

ANNEXE II

INSTALLATION DE KVM ET CRÉATION DE MACHINES VIRTUELLES

Pour exécuter KVM, vous avez besoin d'un processeur qui prend en charge la virtualisation matérielle. Intel et AMD ont tous deux développé des extensions pour leurs processeurs, considérés respectivement Intel VT-x (nom de code Vanderpool) et AMD-V (nom de code Pacifica). Pour voir si votre processeur prend en charge l'un de ces, vous pouvez consulter la sortie de cette commande :

- Egre

Si 0, cela signifie que votre CPU ne prend pas en charge la virtualisation matérielle.
Si 1 ou plusieurs alors il supporte. Mais vous devez toujours vous assurer que la virtualisation est activée dans le BIOS.
Par défaut, si vous avez démarré dans le noyau XEN il n'affichera pas le drapeau « svm » ou « vmx » en utilisant la commande grep. Pour voir s'il est activé ou pas de Xen, entrez :

- Kvm-ok

Il affichera alors si les accélérations de KVM sont activées ou non. Si elles sont désactivées, on peut toujours créer des machines virtuelles. C'est juste qu'elles seront lentes et moins performantes qu'en utilisant les accélérations. Il est préférable d'utiliser KVM avec une architecture à 64 bits.

Installation de KVM

Les étapes suivantes permettent d'installer KVM. Tout d'abord installer les dépendances :

- $ sudo apt-get install qemu-kvm libvirt-bin ubuntu-vm-builder bridge-utils

Ensuite, vous devez vous assurez que votre nom d'utilisateur est ajouté au groupe libvirtd :

- $ sudo adduser `id -un` libvirtd

Après ça vous avez besoin de vous reconnecter (relogin) pour que votre utilisateur devienne effectivement un membre de ce groupe et puisse rouler des machines virtuelles.
Vous pouvez tester si votre installation est réussie en tapant la commande suivante :

- $ virsh -c qemu:///system list

Le résultat devra afficher

- ```
  Id Name                 State
  ----------------------------------
  ```

```
$
```

Optionnel : Installer virt-manager (graphical user interface)

Si vous travaillez avec un ordinateur bureau, vous aimerez peut être installé une interface graphique pour gérer vos machines virtuelles.

```
$ sudo apt-get install virt-manager
```

Le gestionnaire de machine virtuelle apparaitre dans le menu « Applications » -> « Outils système ». Par défaut, la connexion au serveur local est présente par défaut. Vous pouvez ajouter de nouvelles instances QEMU en passant par le menu « fichier » -> « ajouter connexion ».

Avant de créer une machine virtuelle, il ne vous reste qu'à configurer un pont « bridge » au niveau de votre configuration réseau. Pour cela :

- Sudo vi /etc/network/interfaces

Si votre principale interface est eth0. Alors ajouter les lignes suivantes :

- auto br0
  iface br0 inet dhcp
  bridge_ports eth0
  bridge_stp off

Pour prendre en considération les changements effectués, redémarrez le service du réseautage :

- sudo/etc/init.d/networking restart

Il est à noter que cette commande ne marche plus sur Linux 14.04. Il suffit d'utiliser la commande ifup dans ce cas.
```

Maintenant, vous pouvez créer votre machine virtuelle bridgée à l'aide de Virtual Machine Manager qui est sous l'onglet Applications -> Outil Système. Depuis l'interface graphique, cliquez sur l'icône en haut à gauche pour créer une nouvelle machine virtuelle.

ANNEXE III

PRÉSENTATION DE L'INTERFACE GRAPHIQUE DE LA COUCHE PLATEFORME

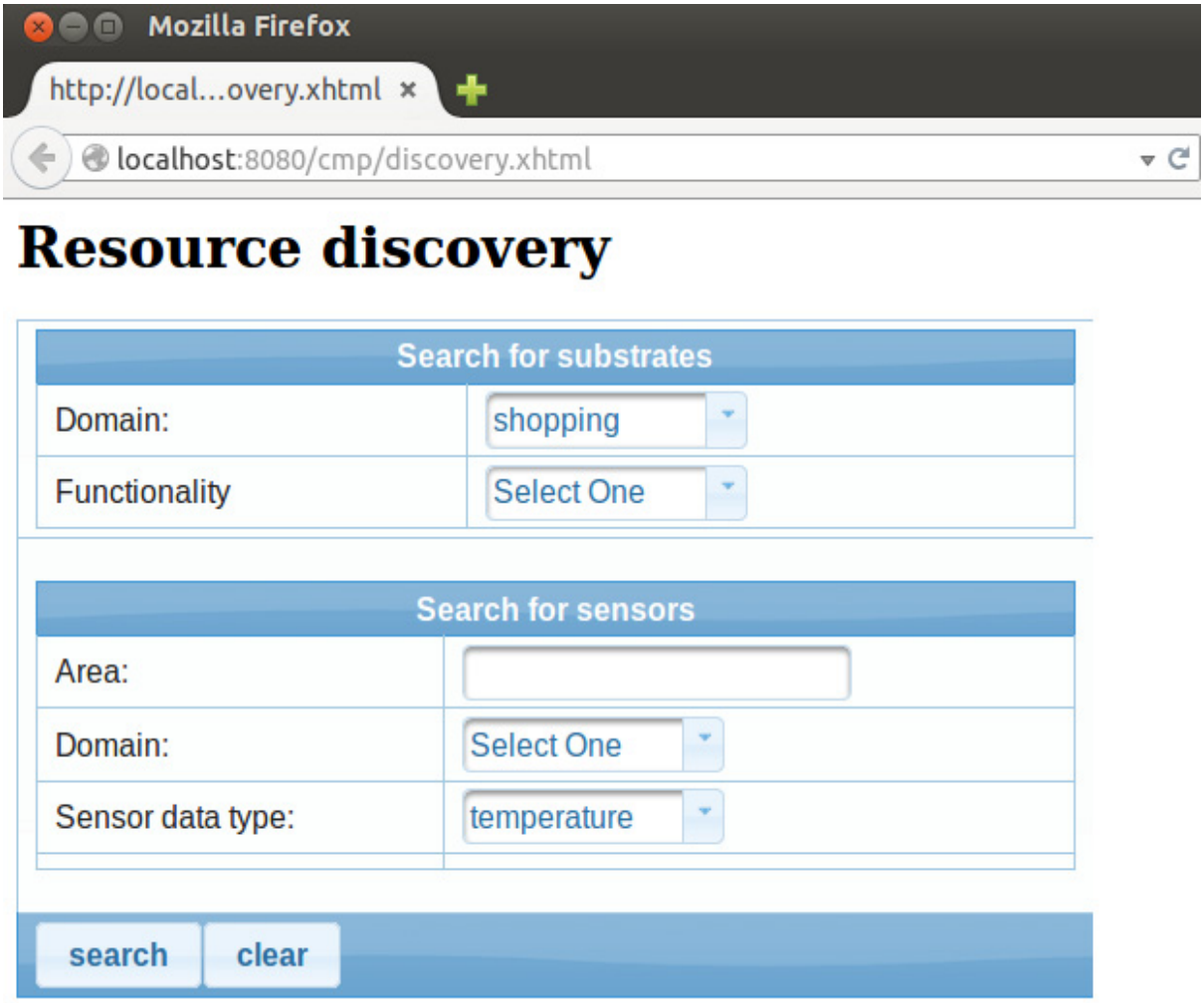

Figure-A III-1 Interface graphique pour effectuer la découverte des ressources

Substrates search results

Id	Name	Description	Provider	functionality
1	dissemination substrate	this is a dissemination substrate	ericson	dissemination
3	modeling substrate	this is a modelign substrate	provider 1	modeling
4	store substrate	this is a storing substrate	ericson	storage
5	inference substrate	this is an inference substrate	ericson	inference
2	acquisition substrate	this is a testing acquisition substrate	provider 1	acquisition

Sensors search results

Id	Name	Location	Sensed types	Provider	Mobility	Uri	Sensing Mode	Sensing Value
3	temperature sensor	montreal	temperature	ericson		http://broker/#temp	periodic	
10	Intersemap temperature sensor	montreal	temperature	ericson		http://localhost:8080/qrcodewebapp/sensors/temperature2	periodic	
6	supra temperature sensor	laval	temperature	provider 1		http://10.180.121.83:8080/qrcodewebapp/sensors/temperature	periodic	
12	temperature 3	montreal	temperature	ericson		http://10.180.121.83:8080/qrcodewebapp/sensors/temperature3	periodic	
13	temperature 4	montreal	temperature	ericson		http://10.180.121.83:8080/qrcodewebapp/sensors/temperature4	periodic	
14	temperature 5	montreal	temperature	ericson		http://10.180.121.83:8080/qrcodewebapp/sensors/temperature5	periodic	
15	temperature 6	montreal	temperature	ericson		http://10.180.121.83:8080/qrcodewebapp/sensors/temperature6	periodic	
16	temperature 7	montreal	temperature	ericson		http://10.180.121.83:8080/qrcodewebapp/sensors/temperature7	periodic	

Add Compose service

Figure-A III-2 Interface graphique pour effectuer la composition d'un service

ANNEXE IV

FICHIER DE CONFIGURATION DES SUBSTRATES

```
-<configuration>
  <disseminationuri>http://10.180.121.45:8080/dissemination/broadcast</disseminationuri>
 -<sensors>
  -<sensor-config>
     <id>1</id>
     <name>codebar sensor</name>
    -<uri>
       http://10.180.121.83:8080/qrcodewebapp/sensors/qrcode
     </uri>
     <capture-mode>periodic</capture-mode>
     <capture-value>30</capture-value>
   </sensor-config>
  -<sensor-config>
     <id>2</id>
     <name>cisco webcam vt II</name>
    -<uri>
       http://10.180.121.83:8080/qrcodewebapp/sensors/qrcode
     </uri>
     <capture-mode>request</capture-mode>
     <capture-value/>
   </sensor-config>
  </sensors>
 </configuration>
```

Figure-A IV-1 Exemple d'un fichier de configuration d'une substrate

ANNEXE V

FICHIER DE COMPOSITION DE SERVICE

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<scFfile>
    <substrates>
        <substrate-description>
            <id>1</id>
            <name>dissemination substrate</name>
            <description>this is a dissemination substrate</description>
            <functionality>dissemination</functionality>
            <provider>
                <id>2</id>
                <instantiationUri>http://localhost:8080/cmi/rest/instantiate</instantiationUri>
                <name>ericson</name>
            </provider>
            <domain>shopping</domain>
            <subProperties xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="disseminationSubstrate"/>
        </substrate-description>
        <substrate-description>
            <id>3</id>
            <name>modeling substrate</name>
            <description>this is a modelign substrate</description>
            <functionality>modeling</functionality>
            <provider>
                <id>1</id>
                <instantiationUri>http://localhost:8080/cmi/rest/instantiate</instantiationUri>
                <name>provider 1</name>
            </provider>
            <domain>shopping</domain>
            <subProperties xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="modelingSubstrate"/>
        </substrate-description>
        <substrate-description>
            <id>4</id>
            <name>store substrate</name>
            <description>this is a storing substrate</description>
            <functionality>storage</functionality>
            <provider>
```

```
            <id>2</id>
            <instantiationUri>http://localhost:8080/cmi/rest/instantiate</instantiationUri>
            <name>ericson</name>
        </provider>
        <domain>shopping</domain>
        <subProperties xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="storageSubstrate"/>
    </substrate-description>
    <substrate-description>
        <id>5</id>
        <name>inference substrate</name>
        <description>this is an inference substrate</description>
        <functionality>inference</functionality>
        <provider>
            <id>2</id>
            <instantiationUri>http://localhost:8080/cmi/rest/instantiate</instantiationUri>
            <name>ericson</name>
        </provider>
        <domain>shopping</domain>
        <subProperties xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="inferenceSubstrate"/>
    </substrate-description>
    <substrate-description>
        <id>2</id>
        <name>acquisition substrate</name>
        <description>this is a testing acquisition substrate</description>
        <functionality>acquisition</functionality>
        <provider>
            <id>1</id>
            <instantiationUri>http://localhost:8080/cmi/rest/instantiate</instantiationUri>
            <name>provider 1</name>
        </provider>
        <domain>shopping</domain>
        <subProperties xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="acquisitionSubstrate">
            <sensorConfig>
                <id>42</id>
                <name>environmental sensor v5 </name>
```

```
71                      <uri>http://10.180.121.83:8080/qrcodewebapp/sensors/environmental/v5</uri>
72                      <capture-mode>periodic</capture-mode>
73                      <capture-value></capture-value>
74                  </sensorConfig>
75                  <sensorConfig>
76                      <id>41</id>
77                      <name>environmental sensor v4 </name>
78                      <uri>http://10.180.121.83:8080/qrcodewebapp/sensors/environmental/v4</uri>
79                      <capture-mode>periodic</capture-mode>
80                      <capture-value></capture-value>
81                  </sensorConfig>
82                  <sensorConfig>
89                  <sensorConfig>
96                  <sensorConfig>
103             </subProperties>
104         </substrate-description>
105     </substrates>
106     <sensors>
107         <sensor-description>
108             <id>42</id>
109             <name>environmental sensor v5 </name>
110             <location>montreal</location>
111             <sensedType>temperature,humidity,light,soundmotion</sensedType>
112             <provider>
113                 <id>2</id>
114                 <instantiationUri>http://localhost:8080/cmi/rest/instantiate</instantiationUri>
115                 <name>ericson</name>
116             </provider>
117             <domain>shopping</domain>
118             <accuracy>0.0</accuracy>
119             <sensitivity>0.0</sensitivity>
120             <falsePositive>0.0</falsePositive>
121             <sensingMode>periodic</sensingMode>
122             <sensingValue></sensingValue>
123             <uri>http://10.180.121.83:8080/qrcodewebapp/sensors/environmental/v5</uri>
```

```
124         </sensor-description>
125         <sensor-description>
126             <id>41</id>
127             <name>environmental sensor v4 </name>
128             <location>montreal</location>
129             <sensedType>temperature,humidity,light,sound</sensedType>
130             <provider>
131                 <id>2</id>
132                 <instantiationUri>http://localhost:8080/cmi/rest/instantiate</instantiationUri>
133                 <name>ericson</name>
134             </provider>
135             <domain>shopping</domain>
136             <accuracy>0.0</accuracy>
137             <sensitivity>0.0</sensitivity>
138             <falsePositive>0.0</falsePositive>
139             <sensingMode>periodic</sensingMode>
140             <sensingValue></sensingValue>
141             <uri>http://10.180.121.83:8080/qrcodewebapp/sensors/environmental/v4</uri>
142         </sensor-description>
143         <sensor-description>
161         <sensor-description>
179         <sensor-description>
197     </sensors>
198 </scFfile>
```

Figure-A V-1 : Exemple d'un fichier de composition de service (SCF)

ANNEXE VI

MODÈLE DE DONNÉES XML IMPLÉMENTÉ COMPLET

```xml
<xs:complexType name="tuple">
  <xs:sequence>
    <xs:element name="timestamp" type="xs:dateTime" minOccurs="0"/>
    <xs:element name="physiologicalData" type="tns:physiology" minOccurs="0"/>
    <xs:element name="environmentalData" type="tns:environment" minOccurs="0"/>
    <xs:element name="identificationData" type="tns:identification" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<!-- physiology  specifictions -->
<xs:complexType name="physiology">
  <xs:sequence>
    <xs:element name="pulseRate" type="tns:pulse" minOccurs="0"/>
    <xs:element name="bodyTemperature" type="tns:bodyTemperature" minOccurs="0"/>
    <xs:element name="respiratoryRate" type="tns:respiration" minOccurs="0"/>
    <xs:element name="bloodPressure" type="tns:pressure" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="pulse">
  <xs:sequence>
    <xs:element name="pulseValue" type="tns:pulseValue" minOccurs="1"/>
    <xs:element name="pulseDescription" type="tns:pulseDescription" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="unit" type="xs:string" use="required" fixed="bpm"/>
</xs:complexType>
<xs:simpleType name="pulseValue">
  <xs:restriction base="xs:unsignedShort">
    <xs:minInclusive value="50"/>
    <xs:maxInclusive value="150"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="pulseDescription">
  <xs:restriction base="xs:string">
    <xs:enumeration value="normalRange"/>
    <xs:enumeration value="tachycardia"/>
    <xs:enumeration value="bradycardia"/>
  </xs:restriction>
</xs:simpleType>
```

```
38  <xs:complexType name="bodyTemperature">
39    <xs:sequence>
40      <xs:element name="temperatureValue" type="tns:tempValue" minOccurs="1"/>
41      <xs:element name="tempDescription" type="tns:tempDescription" minOccurs="0"/>
42    </xs:sequence>
43    <xs:attribute name="unit" type="xs:string" use="required" fixed="Celsius"/>
44  </xs:complexType>
45  <xs:simpleType name="tempValue">
46    <xs:restriction base="xs:unsignedShort">
47      <xs:minInclusive value="30"/>
48      <xs:maxInclusive value="42"/>
49    </xs:restriction>
50  </xs:simpleType>
51  <xs:simpleType name="tempDescription">
52    <xs:restriction base="xs:string">
53      <xs:enumeration value="normalRange"/>
54      <xs:enumeration value="hypothermia"/>
55      <xs:enumeration value="fever"/>
56    </xs:restriction>
57  </xs:simpleType>
58  <xs:complexType name="respiration">
59    <xs:sequence>
60      <xs:element name="respirationRate" type="tns:respirationRate" minOccurs="1"/>
61      <xs:element name="respirationDescription" type="tns:respirationDescription" />
62    </xs:sequence>
63    <xs:attribute name="unit" type="xs:string" use="required" fixed="resp. per minute"/>
64  </xs:complexType>
65  <xs:simpleType name="respirationRate">
66    <xs:restriction base="xs:unsignedShort">
67      <xs:minInclusive value="15"/>
68      <xs:maxInclusive value="40"/>
69    </xs:restriction>
70  </xs:simpleType>
71  <xs:simpleType name="respirationDescription">
72    <xs:restriction base="xs:string">
73      <xs:enumeration value="normalRange"/>
74      <xs:enumeration value="hypoventilation"/>
```

```
75        <xs:enumeration value="hyperventilation"/>
76      </xs:restriction>
77    </xs:simpleType>
78    <xs:complexType name="pressure">
79      <xs:sequence>
80        <xs:element name="pressureValue" type="tns:pressureValue" minOccurs="1" />
81        <xs:element name="pressureDescription" type="tns:pressureDescription" minOccurs="0" />
82      </xs:sequence>
83      <xs:attribute name="unit" type="xs:string" use="required" fixed="mm HG" />
84    </xs:complexType>
85    <xs:complexType name="PressureValue">
86      <xs:sequence>
87        <xs:element name="systolic" type="tns:systolic" nimOccurs="1" />
88        <xs:element name="diastolic" type="tns:diastolic" nimOccurs="1" />
89      </xs:sequence>
90    </xs:complexType>
91    <xs:simpleType name="systolic">
92      <xs:restriction base="xs:unsignedShort">
93        <xs:minInclusive value="100" />
94        <xs:maxInclusive value="200" />
95      </xs:restriction>
96    </xs:simpleType>
97    <xs:simpleType name="diastolic">
98      <xs:restriction base="xs:unsignedShort">
99        <xs:minInclusive value="60" />
100       <xs:maxInclusive value="100" />
101     </xs:restriction>
102   </xs:simpleType>
103   <xs:simpleType name="pressureDescription">
104     <xs:restriction base="xs:string">
105       <xs:enumeration value="normalRange" />
106       <xs:enumeration value="hyperTension" />
107       <xs:enumeration value="hypoTension" />
108     </xs:restriction>
109   </xs:simpleType>
110   <!-- Environment specifications-->
111   <xs:complexType name="environment">
```

```
112     <xs:sequence>
113       <xs:element name="ambientTemperature" type="tns:temperature" minOccurs="0" />
114       <xs:element name="soundLevel" type="tns:sound" minOccurs="0" />
115       <xs:element name="lightIntensity" type="tns:light" minOccurs="0" />
116       <xs:element name="relativeHumidity" type="tns:humidity" minOccurs="0" />
117     </xs:sequence>
118   </xs:complexType>
119   <xs:complexType name="temperature">
120     <xs:sequence>
121       <xs:element name="tempValue" type="xs:int" minOccurs="1" />
122     </xs:sequence>
123     <xs:attribute name="unit" type="tns:tempUnit" use="required" />
124   </xs:complexType>
125   <xs:simpleType name="tempUnit">
126     <xs:restriction base="xs:string">
127       <xs:enumeration value="Celsius" />
128       <xs:enumeration value="Fahrenheit" />
129       <xs:enumeration value="Kelvin" />
130     </xs:restriction>
131   </xs:simpleType>
132   <xs:complexType name="sound">
133     <xs:sequence>
134       <xs:element name="soundlevel" type="xs:int" minOccurs="1" />
135     </xs:sequence>
136     <xs:attribute name="unit" type="xs:string" use="required" fixed="decibels" />
137   </xs:complexType>
138   <xs:complexType name="light">
139     <xs:sequence>
140       <xs:element name="intensity" type="xs:int" minOccurs="1" />
141     </xs:sequence>
142     <xs:attribute name="unit" type="xs:string" use="required" fixed="candelas" />
143   </xs:complexType>
144   <xs:complexType name="humidity">
145     <xs:sequence>
146       <xs:element name="humidityLevel" type="tns:humidityLevel" minOccurs="1" />
147     </xs:sequence>
148     <xs:attribute name="unit" type="xs:string" use="required" fixed="%" />
```

```
149     </xs:complexType>
150     <xs:simpleType name="humidityLevel">
151       <xs:restriction base="xs:unsignedShort">
152         <xs:minInclusive value="30" />
153         <xs:maxInclusive value="100" />
154       </xs:restriction>
155     </xs:simpleType>
156     <!-- identification specifications -->
157     <xs:complexType name="identification">
158       <xs:sequence>
159         <xs:element name="location" type="tns:location" minOccurs="0"/>
160         <xs:element name="idTag" type="tns:idTag" minOccurs="0"/>
161         <xs:element name="codebar" type="tns:codebar" minOccurs="0"/>
162         <xs:element name="motion" type="tns:motion"
163           </xs:sequence>
164     </xs:complexType>
165     <xs:complexType name="location">
166       <xs:sequence>
167         <xs:element name="latitude" type="tns:float" minOccurs="0"/>
168         <xs:element name="longitude" type="tns:float" minOccurs="0"/>
169         <xs:element name="coordinateType" type="tns:coordinateType" minOccurs="0"/>
170       </xs:sequence>
171     </xs:complexType>
172     <xs:simpleType name="coordinateType">
173       <xs:restriction base="xs:string">
174         <xs:enumeration value="gps"/>
175       </xs:restriction>
176     </xs:simpleType>
177     <xs:complexType name="idTag">
178       <xs:sequence>
179         <xs:element name="tagValue" type="xs:string" minOccurs="0"/>
180         <xs:element name="itemType" type="tns:itemType" minOccurs="0"/>
181         <xs:element name="positionX" type="xs:float" minOccurs="0"/>
182         <xs:element name="positionY" type="xs:float" minOccurs="0"/>
183       </xs:sequence>
184     </xs:complexType>
185     <xs:simpleType name="itemType">
```

```
186     <xs:restriction base="xs:string">
187       <xs:enumeration value="identified"/>
188       <xs:enumeration value="unknown"/>
189     </xs:restriction>
190   </xs:simpleType>
191   <xs:complexType name="codebar">
192     <xs:sequence>
193       <xs:element name="codeValue" type="xs:string" minOccurs="0"/>
194       <xs:element name="codeType" type="tns:codeType" minOccurs="0"/>
195     </xs:sequence>
196   </xs:complexType>
197   <xs:simpleType name="codeType">
198     <xs:restriction base="xs:string">
199       <xs:enumeration value="scanned"/>
200       <xs:enumeration value="added"/>
201       <xs:enumeration value="removed"/>
202       <xs:enumeration value="unknown"/>
203     </xs:restriction>
204   </xs:simpleType>
205   </xs:schema>
206   <xs:complexType name="motion">
207     <xs:sequence>
208       <xs:element name="motionValue" type="xs:string" minOccurs="0"/>
209       <xs:element name="motionType" type="tns:motionType" minOccurs="0"/>
210     </xs:sequence>
211   </xs:complexType>
212   <xs:simpleType name="motionType">
213     <xs:restriction base="xs:string">
214       <xs:enumeration value="noMotion"/>
215       <xs:enumeration value="ingoing"/>
216       <xs:enumeration value="outgoing"/>
217     </xs:restriction>
218   </xs:simpleType>
```

Figure-A VI-1 Modèle de données XML implémenté complet

ANNEXE VII

INSTALLATION ET CRÉATION D'UNE BDD AVEC POSTGRESQL 9.1

Installer les paquets nécessaires

$ sudo apt-get install postgresql postgresql-contrib

Ceci installera la partie serveur et cliente. Si l'on veut juste la partie cliente il suffit de taper

$ sudo apt-get install postgres-client

Pour changer le mot de passe par défaut du super utilisateur :

$ sudo –u postgres template1

Puis

ALTER USER postgres with encrypted password '<nouveau mot de passe'>

Cliquer sur CTRL+d puis editer le fichier/etc/postgresql/9.1/main/pg_hba.conf. changer la ligne : local all postgres peer par local all postgres md5.

Redémarrer le service par :

/etc/init.d/postgresql restart

Après, créer votre super utilisateur au niveau de la base de données

Createuser –U postgres –d –e –E –l –P –r –s <nom d'utilisateur>

Entrer ensuite votre mot de passe. Maintenant vous pouvez accéder à PostgreSQL et la manipuler.

Commandes de base pour commencer à utiliser PostgreSQL

pour créer une nouvelle base de données :

$ createdb <nom bdd>

Pour manipuler une base de données « test »

$ psql test

Pour créer une nouvelle table

Test=# Create table profile (id serial, name varchar(20), age int, birthday date, description text);

Ceci créera une table nommée profile qui contient cinq champs. :

Un identifiant id de type serial qui va s'incrémenter automatiquement.

Le nom « name » qui est de type chaine de caractères ne dépassant pas vingt caractères.

L’âge qui est de type nombre entier.

La date de naissance « birthday » qui est de type date.

Une description qui est de type chaine de caractères sans restriction sur sa taille.

BIBLIOGRAPHIE

A. K. Dey, » Understanding and Using Context", Personal and Ubiquitous Computing, vol. 5, no. 1, pp. 4-7, 2001.

A. K. Dey, » Providing Architectural Support for Building Context-Aware Applications", PhD Thesis, College of Computing, Georgia Institute of Technology, December 2000.

A. K. Dey, G. D. Abowd and D. Salber, » A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications", Human-Computer Interaction, vol. 16, no. 2, pp. 97-166, 2001.

Anusuriya Devaraju, Simon Hoh, and Michael Hartley. 2007. A context gathering framework for context-aware mobile solutions. In Proceedings of the 4th international conference on mobile technology, applications, and systems and the 1st international symposium on Computer human interaction in mobile technology (Mobility « 07). ACM, New York, NY, USA, 39-46.

Chihani, B.; Bertin, E.; Crespi, N., « Enhancing M2M Communication with Cloud-Based Context Management, » in Next Generation Mobile Applications, Services and Technologies (NGMAST), 2012 6th International Conference on, vol., no., pp.36-41, 12-14 Sept. 2012

B. Jennings, S. V. Meer, S. Balasubramaniam, D. Botvich, M. O'Foghlu, W. Donnelly and J. Strassner, "Towards Autonomic Management of Communications Networks", IEEE Commun. Mag., vol.45, no. 10, pp. 112-121, 2007.

Claudio Bettini, Oliver Brdiczka, Karen Henricksen, Jadwiga Indulska, Daniela Nicklas, Anand Ranganathan, and Daniele Riboni. 2010. A survey of context modelling and reasoning techniques. Pervasive Mob. Comput. 6, 2 (April 2010), 161-180.

C. Anagnostopoulos, A. Tsounis and S. Hadjiefthymiades, "Context Awareness in Mobile Computing Environments: a Survey", Springer Wireless Personal Communications, vol. 42, no. 3, pp. 445-464, 2007.

Charith Pereray, Arkady Zaslavskyy, Peter Christen, Michael Compton. « Context-aware Sensor Search, Selection and Ranking Model for Internet of Things Middleware »

D. Franklin and J. Flaschbart, "All Gadget and No Representation Makes Jack a Dull Environment », in Proc. AAAI Spring Symposium on Intelligent Environments, Palo Alto, 1998, pp. 155-160.

Dineshbalu Balakrishnan, May El Barachi, Ahmed Karmouch, and Roch Glitho. 2005. Challenges in modeling and disseminating context information in ambient networks.

In Proceedings of the Second international conference on Mobility Aware Technologies and Applications (MATA'05), Thomas Magedanz, Ahmed Karmouch, Samuel Pierre, and Iakovos Venieris (Eds.). Springer-Verlag, Berlin, Heidelberg, 32-42.

Badidi, E.; Taleb, I., "Towards a cloud-based framework for context management," in Innovations in Information Technology (IIT), 2011 International Conference on , vol., no., pp.35-40, 25-27 April 2011

Belqasmi, F.; Azar, C.; Glitho, R.; Soualhia, M.; Kara, N., « A case study on IVR applications' provisioning as cloud computing services," in Network, IEEE , vol.28, no.1, pp.33-41, January-February 2014

G. Chen and D. Kotz, ”Solar: A Pervasive-Computing Infrastructure for Context-Aware Mobile Applications”, Technical Report TR2002-421, Dartmouth College, 2002.

Gilles Gigan , Ian Atkinson. « Sensor Abstraction Layer: a unique software interface to effectively manage sensor networks », 2007

H. W. Gellersen, A. Schmidt and M. Beigl, » Multi-Sensor Context-Awareness in Mobile Devices and Smart Artifacts », Mobile Networks and Applications (MONET), vol. 7, no. 5, pp. 341-351, 2002.

Hyun Jung La; Soo Dong Kim, "A Conceptual Framework for Provisioning Context-aware Mobile Cloud Services," in Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on , vol., no., pp.466-473, 5-10 July 2010

Hyun Jung La; Moon Kwon Kim; Soo Dong Kim, "A Cloud Service Framework for Visualizing and Reasoning with Mobile Contexts," in Mobile Services (MS), 2012 IEEE First International Conference on , vol., no., pp.25-32, 24-29 June 2012

I.F., Akyildiz; Weilian Su; Sankarasubramaniam, Y.; Cayirci, E., « A survey on sensor networks, » in Communications Magazine, IEEE , vol.40, no.8, pp.102-114, Aug 2002

J. Belschner et al., » Optimisation of Radio Access Network Operation Introducing Self-x Functions: Use Cases, Algorithms, Expected Efficiency Gains”, IEEE 69th Vehicular Technology Conference, VTC Spring 2009.

J. Strassner and D. O’Sullivan, » Knowledge Management for Context-Aware Policy-based Ubiquitous Computing Systems », in Proc. 6th International Workshop on Managing Ubiquitous Communications and Services, Spain, 2009, pp. 67-76.

J. Strassner, S. V. Meer, D. O'Sullivan and S. Dobson, » The Use of Context-Aware Policies and Ontologies to Facilitate Business-Aware Network Management", Journal of Network and Systems Management, vol. 17, no. 3, pp. 255-284, 2009.

K. E. Kjr, » A Survey of Context-Aware Middleware », in Proc. 25th conference on International Multi-Conference : Software Engineering, Innsbruck, Austria, 2007, pp. 148-155.

K. Henricksen, "A Framework for Context-Aware Pervasive Computing Applications", PhD thesis, School of Information Technology and Electrical Engineering, The University of Queensland, Australia, 2003.

Karen Henricksen, Jadwiga Indulska, Ted McFadden, and Sasitharan Balasubramaniam. 2005. Middleware for distributed context-aware systems. In Proceedings of the 2005 Confederated international conference on On the Move to Meaningful Internet Systems - Volume >Part I (OTM'05), Robert Meersman and Zahir Tari (Eds.), Vol. >Part I. Springer-Verlag, Berlin, Heidelberg, 846-863.

K. Henricksen and J. Indulska, "Modelling and Using Imperfect Context Information », in Proc. 2nd IEEE Annual Conference on Pervasive Computing and Communications Workshops (PERCOMW), 2004, pp. 33-37.

L. Sarakis, G. Kormentzas and F. M. Guirao, "Seamless Service Provision for Multi Heterogeneous Access", IEEE Wireless Communications, vol. 16, no. 5, pp. 32-40, 2009.

May El Barachi and Nadjia Kara, 'Contextual Information acquisition and management solutions for IP-based Networks: A survey', Recent Patents on Telecommunications Journal, March 2013.

M. Charalambides, G. Pavlou, P. Flegkas, J. R. Loyola, A. K. Bandara, E. C. Lupu, A. Russo, N. Dulay and M. Sloman, "Policy Conflict Analysis for DiffServ Quality of Service Management", IEEE Trans. Network Service Management, vol. 6, no. 1, pp. 15-30, 2009.

N. Agoulmine, S. Balasubramaniam, D. Botvitch, J. Strassner, E. Lehtihet and W. Donelly, "Challenges for Autonomic Network Management », in Proc. 1st conference on Modelling Autonomic Communication Environment (MACE), Ireland, 2006.

P. Debaty, P. Goddi and A. Vorban, "Integrating the Physical World with the Web to Enable Context-Enhanced Services", Springer Mobile Networks and Applications, vol. 10, no. 4, pp. 385-394, 2005.

P. Gray and D. Salber, ”Modelling and Using Sensed Context Information in the Design of Interactive Applications », in Proc. 8th IFIP International Conference on Engineering for Human-Computer Interaction, LNCS 2254, 2001, pp. 317-336.

P. J. Brown, J. D. Bovey and X. Chen, ”Context-aware applications: From the laboratory to the marketplace”, IEEE Pers. Commun., vol. 4, no. 5, pp. 58-64, 1997.

Prodromos Makris, Dimitrios N. Skoutas, Charalabos Skianis . “A Survey on Context-Aware Mobile and Wireless Networking: On Networking and Computing Environments Integration »

Randy Bias, 2009, http://www.cloudscaling.com/blog/cloud-computing/cloud-standards-are-misunderstood/

R. Hull, P. Neaves and J. Bedford-Roberts, » Towards situated computing », in Proc. 1st International Symposium on Wearable Computers (ISWC), Cambridge, 1997, pp. 146-153.

R. Sterritt, M. Mulvenna and A. Lawrynowicz, ”Dynamic and Contextualised Behavioural Knowledge in Autonomic Communications”, Springer Verlag Berlin Heidelberg, LCNS 3457, pp. 217-228, 2005.

R. Schmohl and U. Baumgarten, ”Context-aware Computing: a Survey Preparing a Generalized Approach », in Proc. International Multi-Conference of Engineers and Computer Scientists, Hong Kong, 2008.

R. Winter, J. H. Schiller, N. Nikaein and C. Bonnet, » CrossTalk: Cross- Layer Decision Support based on Global Knowledge”, IEEE Commun. Mag., vol. 44, no. 1, pp. 93-99, 2006.

S. Dobson, S. Denazis, A. Fernandez, D. Gaiti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt and F. Zambonelli, » A Survey of Autonomic Communications”, ACM Trans. Autonomous and Adaptive Systems, vol. 1, no. 2, pp. 223-259, 2006.