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 GÉNIE
CONCENTRATION : RÉSEAUX DE TÉLÉCOMMUNICATIONS
M. Ing.

PAR
Ridha GADHGADHI

OPENICRA : VERS UN MODÈLE GÉNÉRIQUE DE DÉPLOIEMENT AUTOMATISÉ DES APPLICATIONS DANS LE NUAGE INFORMATIQUE

MONTRÉAL, LE 6 SEPTEMBRE 2013

©Tous droits réservés, Ridha Gadhgadhi, 2013

©Tous droits réservés

Cette licence signifie qu'il est interdit de reproduire, d'enregistrer ou de diffuser en tout ou en partie, le présent document. Le lecteur qui désire imprimer ou conserver sur un autre media une partie importante de ce document, doit obligatoirement en demander l'autorisation à l'auteur.

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

M. Mohamed Cheriet, directeur de mémoire
Département de génie de la production automatisée à l'École de technologie supérieure

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

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

M. Ali Kanso, examinateur externe
Ericsson Research, Ericsson Inc.

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

LE 23 AOÛT 2013

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Je tiens tout d'abord à remercier chaleureusement mon directeur de recherche, le professeur Mohamed Cheriet, pour son expertise, ses conseils et son soutien inconditionnel. Je remercie aussi Mathieu Lemay d'Inocybe Technologies Inc. pour ses conseils techniques.

Je tiens à remercier tous les membres du jury de m'avoir fait l'honneur d'accepter de juger mon mémoire :

- Monsieur Stéphane Coulombe, de l'ÉTS, Université du Québec;
- Monsieur Chamseddine Talhi, de l'ÉTS, Université du Québec;
- Monsieur Ali Kanso, d'Ericsson Montréal, Canada.

Mes remerciements s'adressent également à tous les membres du laboratoire Synchromedia avec qui il fut un plaisir de travailler sur les projets EcoloTIC et GreenStar Network.

Je remercie aussi l'organisme Canarie et le CRSNG pour leurs subventions et soutiens financiers. Sans le financement de ceux-ci, ce projet n'aurait pas été possible.

Finalement, je dédie ce travail à toute ma famille.

OPENICRA : VERS UN MODÈLE GÉNÉRIQUE DE DÉPLOIEMENT AUTOMATISÉ DES APPLICATIONS DANS LE NUAGE INFORMATIQUE

Ridha GADHGADHI

RÉSUMÉ

Le nuage informatique ou le « Cloud Computing » est une approche informatique qui fait référence à la mise à disposition de ressources informatiques à la demande via Internet ou en privé via le réseau interne de l'entreprise, avec un modèle de facturation à l'usage. Le paradigme IaaS (Infrastructure as a Service) consiste à fournir aux utilisateurs un accès, à la demande et en libre-service, à un parc informatique virtualisé, souvent composé de machines virtuelles sur lesquelles les utilisateurs peuvent installer, contrôler et personnaliser leurs applications. Alternativement, le modèle de service PaaS (Platform as a Service) offre aux utilisateurs un environnement de programmation disponible immédiatement et entièrement gérable pour créer et déployer des applications évolutives dans le nuage informatique et sans intervention de l'utilisateur. Bien que le choix d'un tel environnement puisse paraitre assez avantageux, plusieurs défis se lèvent devant l'utilisation efficace des systèmes du nuage informatique. Ainsi, les services du nuage informatique sont offerts à différents niveaux d'abstraction où les fournisseurs de nuage exposent l'accès à leurs services via des APIs propriétaires. Cela encourage l'enfermement propriétaire et limite l'interopérabilité des services de nuage informatique, ce qui constitue une barrière à l'entrée significative pour les utilisateurs de nuage. Plusieurs solutions ont été proposées en se basant sur l'utilisation des couches intermédiaires qui permettent d'isoler les applications de la variabilité de certains services offerts par les fournisseurs de nuages. Toutefois, ces approches constituent une solution partielle de ce problème lorsque de telles solutions utilisent certaines technologies propriétaires, car elles contribuent au risque de déplacer l'effet de l'enfermement propriétaire du fournisseur vers les outils de déploiement.

L'objectif principal de notre recherche consiste à concevoir et à développer un nouveau modèle générique de déploiement automatique des applications dans le nuage informatique, afin d'atténuer les effets de ces barrières à l'entrée, de réduire la complexité de développement des applications et de simplifier le processus de déploiement des services dans le nuage. De surcroît, supporter et déployer automatiquement des applications sur le nuage en assurant l'élasticité, la mise à l'échelle automatique et l'interopérabilité avec toutes les plateformes et en optimisant la gestion du stockage sont les objectifs primordiaux de ce mémoire. Notre modèle proposé OpenICRA met en œuvre une architecture en couches qui cache les détails de l'implémentation permettant d'avoir un processus de déploiement simple. En outre, contrairement aux autres solutions de nuages informatiques disponibles telles que Google App Engine, Windows Azure ou Amazon Elastic Beanstalk, les composants du modèle proposé se caractérisent par la liberté d'accès à leur code source. Ceci nous permet de garantir la portabilité des applications dans tout environnement d'exécution, d'éviter l'enfermement propriétaire et de faciliter le processus d'automatisation des applications dans le nuage. Les méthodes de redondance, de mise à l'échelle et l'intégration du système de fichiers distribué avec la couche IaaS permettent d'assurer la haute disponibilité, l'évolutivité

et l'extensibilité du modèle et des applications ainsi que l'optimisation de la gestion de stockage des disques durs virtuels des VMs dans tout environnement du nuage.

Nous avons réalisé deux cas d'études réels pour valider le modèle OpenICRA dont le premier consiste à automatiser le déploiement de l'intergiciel distribué OpenSAF dans un cluster de nœuds au sein de l'environnement Cloud du réseau GSN (Synchromedia, 2010), alors que le deuxième cas consiste à migrer l'application de travail collaboratif ICRA (Cheriet, 2012) vers le nuage EC2 d'Amazon (2012a). Nos résultats empiriques démontrent l'efficacité de notre modèle proposé pour déployer différents types d'applications sans apporter aucune modification dans leurs codes sources. En outre, nous démontrons comment notre proposition est capable d'automatiser et d'orchestrer le processus de déploiement des applications et d'optimiser leur exécution en fonction de la performance dans des environnements de nuage informatique hétérogènes.

Mots-clés : Nuage informatique, Modèle générique, Déploiement automatisé, Migration vers le nuage, Évolutivité, Enfermement propriétaire, Gestion du stockage dans le nuage.

OPENICRA: TOWARDS A GENERIC MODEL FOR AUTOMATIC DEPLOYMENT OF APPLICATIONS IN THE CLOUD COMPUTING

Ridha GADHGADHI

ABSTRACT

Cloud computing is a computing approach that refers to the on-demand provision of computational resources via the Internet or in private via the internal enterprise network, with a pay-as-you-use billing model. The paradigm IaaS (Infrastructure as a Service) provides users with an on-demand self-service access, to a virtualized IT infrastructure, often composed of virtual machines on which users can install, manage and customize their applications. Alternatively, PaaS (Platform as a Service) offers users a programming environment immediately available and fully manageable to create and deploy scalable applications in the cloud without user intervention. Although the choice of such an environment may seem advantageous enough, several challenges remain to effectively use cloud computing systems. Thus, cloud computing services are offered at different levels of abstraction where the cloud providers expose access to their services via proprietary APIs. This encourages vendor lock-in and limits the interoperability of cloud computing services, which constitutes a significant barrier to entry for cloud users. Several solutions have been proposed based on the use of intermediate layers that isolate applications from the variability of services offered by cloud providers. However, these approaches provide a partial solution to this problem when such solutions use certain proprietary technologies as they contribute to the risk of moving the effect of vendor lock-in from cloud providers to deployment tools.

The main objective of our research is to design and develop a new generic model for automated deployment of applications in the cloud to mitigate the effects of these barriers to entry, reduce the complexity of application development and simplify the process of deploying cloud services. In addition, support and automatically deploy applications on the cloud by providing elasticity, making automatic-scaling and interoperability with all platforms and optimizing storage management are the primary objectives of this memory. Our proposed model, OpenICRA, implements a layered architecture that hides the implementation details, allowing having a simple deployment process. Furthermore, unlike other cloud computing solutions available such as Google App Engine, Windows Azure or Amazon Elastic Beanstalk, the components of the proposed model is characterized by the freedom to access to their source code. This allows us to ensure application portability in any execution environment, avoid vendor lock-in and facilitate automation process of applications in the cloud. The methods of redundancy, scaling and integration of distributed file system with the IaaS layer are used to provide high availability, scalability and extensibility of the model and applications as well as optimizing storage management of VMs virtual hard disks in any cloud environment.

We conducted two real case studies to validate our proposed model, where the first consists to automate the deployment of a distributed middleware called OpenSAF across a cluster of nodes in the Cloud environment of GSN Network (Synchromedia, 2010), while the second

case is to ensure the migration of the collaborative work application, ICRA (Cheriet, 2012) to Amazon EC2 (Amazon, 2012a). Our empirical results demonstrate the effectiveness of our proposed model to deploy different types of applications without any change in their source code. In addition, we show how our proposal is able to automate and orchestrate the process of application deployment and optimize their implementation in terms of performance in heterogeneous cloud environments.

Keywords : Cloud computing, Generic Model, Automated Deployment, Cloud Migration, Scalability, Vendor lock-in, Cloud storage management.

TABLE DES MATIÈRES

Page

CHAPITRE 1 INTRODUCTION 1
1.1 Contexte 1
1.2 Problématique 4
1.3 Objectifs du mémoire.....6
1.4 Plan du mémoire 9

CHAPITRE 2 REVUE DE LA LITTÉRATURE 11
2.1 Introduction.....11
2.2 La virtualisation 11
2.2.1 Définitions..... 11
2.2.2 Les modèles d'utilisation de la virtualisation et ses avantages..... 13
2.2.2.1 Isolation des applications..... 14
2.2.2.2 Consolidation des serveurs 14
2.2.2.3 Migration des machines virtuelles 16
2.2.2.4 Test et développement 17
2.2.2.5 L'équilibrage de charge dynamique et la récupération après les sinistres..... 17
2.2.3 Les différentes techniques de la virtualisation..... 18
2.2.3.1 L'isolation 18
2.2.3.2 La virtualisation complète..... 19
2.2.3.3 La paravirtualisation 21
2.2.3.4 La virtualisation assistée par matériel..... 23
2.3 Le nuage informatique 24
2.3.1 Nuage informatique : qu'est-ce que c'est?..... 24
2.3.2 Les ancêtres du nuage 26
2.3.2.1 Grid Computing 26
2.3.2.2 Utility Computing 27
2.3.3 Caractéristiques du nuage 28
2.3.4 Modèles de service..... 29
2.3.5 Modèles de déploiement 31
2.4 Travaux connexes 34
2.5 Conclusion 40

CHAPITRE 3 DÉPLOIEMENT AUTOMATISÉ DES APPLICATIONS DANS LE NUAGE.....41
3.1 Introduction.....41
3.2 Enfermement propriétaire et portabilité.....41
3.3 Approche de redondance.....43
3.4 Mise à l'échelle automatique des applications.....44
3.5 Gestion du stockage dans le nuage (IaaS).....47
3.5.1 Architecture actuelle du système de stockage de GSN..... 47
3.5.2 Architecture de HDFS..... 48

3.5.3 Système de stockage proposé : Intégration du HDFS avec GSN 50
3.6 Processus de déploiement automatique des applications dans le nuage 53
3.6.1 Préparation du déploiement de l'application 54
3.6.2 Déploiement automatique des services et des applications 54
3.6.3 Surveillance et gestion du déploiement 56
3.7 Conclusion 56

CHAPITRE 4 ARCHITECTURE ET CONCEPTION DU MODÈLE DE DÉPLOIEMENT 59
4.1 Introduction.....59
4.2 Conception du modèle PaaS proposé : OpenICRA 60
4.2.1 Hypothèses..... 60
4.2.2 Architecture globale du modèle proposé 61
4.2.3 Contrôleur de nuage..... 63
4.2.4 Le gestionnaire de nuage 65
4.2.5 Les ressources du système 66
4.2.5.1 Les nœuds de données 66
4.2.5.2 Les conteneurs d'applications..... 66
4.2.6 Les Modules/Templates 67
4.2.6.1 Les modules 67
4.2.6.2 Les templates 68
4.2.7 Les mécanismes de communication du système..... 68
4.3 Choix des solutions technologiques.....69
4.3.1 Choix d'une plateforme en tant que service (PaaS)..... 69
4.3.1.1 Critères de choix 69
4.3.1.2 Sélection du PaaS..... 71
4.3.1.3 Description d'OpenShift 75
4.3.2 Choix d'outil de gestion de configuration 79
4.3.2.1 Comparaison 79
4.3.2.2 Choix de l'outil 81
4.3.3 Choix du système de fichiers distribué 81
4.3.3.1 Critères de choix 82
4.3.3.2 Comparaison qualitative des systèmes de fichiers distribués 84
4.3.3.3 Sélection du système de fichiers distribué 88
4.4 Conclusion 89

CHAPITRE 5 EXPÉRIMENTATION ET VALIDATION 91
5.1 Introduction.....91
5.2 1er cas d'étude : Automatisation du déploiement d'OpenSAF 91
5.2.1 Présentation du cas d'étude..... 91
5.2.2 Processus d'automatisation du déploiement d'OpenSAF..... 92
5.2.2.1 Préparation du déploiement 93
5.2.2.2 Déploiement automatique 95
5.2.2.3 Monitorage et gestion du déploiement..... 96
5.2.3 Résultats..... 98

5.3 2ème cas d'étude : Migration automatique du système ICRA vers EC2.....100
5.3.1 Présentation du cas d'étude..... 100
5.3.2 Processus de migration du système de travail collaboratif vers le nuage EC2 102
5.3.2.1 Préparation de la migration..... 103
5.3.2.2 Migration automatique vers EC2 107
5.3.2.3 Monitorage de la migration..... 109
5.3.3 Évaluation des performances globales du système de travail collaboratif..... 110
5.3.3.1 Comparaison des codecs et passage du G.711 à G.729 111
5.3.3.2 Test de l'évolutivité du système collaboratif..... 112
5.4 Conclusion 117

CONCLUSION.....119

LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES.....125

LISTE DES TABLEAUX

Page

Tableau 4.1 Critères de sélections des plateformes en tant que service 69

Tableau 4.2 Évaluation des PaaS par rapport aux critères de sélection établis 74

Tableau 4.3 Comparaison de Puppet, Chef et CFEngine 79

Tableau 4.4 Comparaison des différents systèmes de fichiers distribués.....85

Tableau 4.5 Les avantages et les inconvénients des DFSs comparés.....87

Tableau 5.1 Configuration des ressources d'OpenSAF.....94

Tableau 5.2 Durée de déploiement avant et après l'automatisation.....100

LISTE DES FIGURES

Page

Figure 1.1 Schéma synoptique de notre système et des différents éléments présentés dans ce mémoire....10

Figure 2.1 Hyperviseur de type 1....12

Figure 2.2 Hyperviseur de type 2....13

Figure 2.3 Isolation des applications....14

Figure 2.4 Consolidation des serveurs....15

Figure 2.5 Migration des machines virtuelles....16

Figure 2.6 Migration à chaud des machines virtuelles avec le module vMotion de VMware....17

Figure 2.7 Isolateur....19

Figure 2.8 Approche de la virtualisation complète....20

Figure 2.9 Approche de la paravirtualisation....22

Figure 2.10 Approche de la virtualisation assistée par matériel....23

Figure 2.11 Architecture du nuage informatique....30

Figure 2.12 Architecture du nuage hybride....32

Figure 3.2 Mise à l'échelle automatique : Communication TCP inter-conteneur....45

Figure 3.3 Système de stockage actuel de GSN....47

Figure 3.5 Intégration du système de fichiers distribué HDFS avec le réseau GSN....52

Figure 3.6 Processus de déploiement automatique en trois étapes....53

Figure 4.1 Architecture globale d'OpenICRA....62

Figure 4.2 Contrôleur de nuage....64

Figure 4.3 Gestionnaire de nuage....65

Figure 4.4 Relation entre les différentes ressources du système 67

Figure 4.5 Vue d'ensemble de la plateforme OpenShift 76

Figure 4.6 Des templates d'une application sur des gears 77

Figure 5.1 Processus de déploiement de l'intergiciel OpenSAF 93

Figure 5.2 État des nœuds de cluster dans le tableau de bord du système 97

Figure 5.3 Temps d'exécution du démon Puppet dans le nœud du contrôleur 97

Figure 5.4 Résultat du sondage de déploiement manuel d'OpenSAF 99

Figure 5.5 Déploiement réussi du contrôleur d'OpenSAF
en moins de 166 secondes 99

Figure 5.6 Architecture globale du système collaboratif ICRA 103

Figure 5.7 État global des conteneurs du système.....110

Figure 5.8 Test des performances des codecs G.711 et G.729 112

Figure 5.9 Taux d'utilisation du CPU du système ICRA.....114

Figure 5.10 Débit moyen utilisé 115

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

AEB	Amazon Elastic Beanstalk
AFS	Andrew File System
API	Application Programming Interface
AWS	Amazon Web Services
BIOS	Basic Input Output System
CLI	Command Line Interface
CPU	Central Processing Unit
CRM	Customer Relationship Management
CSCF	Call Session Control Function
CTAD	Customizable Template for Automatic Deployment
CTP	Coût Total de Possession
DFS	Distributed File System
DNS	Domain Name System
EC2	Elastic Compute Cloud
FUSE	Filesystem in Userspace
GFS	Google File System
GPFS	General Parallel File System
GSN	GreenStar Network
HDFS	Hadoop Distributed File System
HSS	Home Subscriber Server
IaaS	Infrastructure as a Service
ICRA	Internet based Collaboration and Resource Application
IDC	International Data Corporation
IMS	IP Multimedia Subsystem
IP	Internet Protocol
IPBX	IP Private Branch Exchange
KDC	Key Distribution Center
LDAP	Lightweight Directory Access Protocol
NFS	Network File System

NIST	National Institute for Standards and Technology
OCCI	Open Cloud Computing Interface
OS	Operating System
PaaS	Platform as a Service
PABX	Private Automatic Branch eXchange
PHP	Hypertext Preprocessor
POSIX	Portable Operating System Interface
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RAM	Random Access Memory
REST	REpresentational State Transfer
RPC	Remote Procedure Call
RSI	Retour Sur Investissement
S3	Simple Storage Service
SaaS	Software as a Service
SIP	Session Initiation Protocol
SOA	Service Oriented Architecture
SSH	Secure Shell
TCP	Transmission Control Protocol
TIC	Technologies de l'Information et de la Communication
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VE	Virtual Environment
VM	Virtual Machine
VMM	Virtual Machine Manager
VoIP	Voice over IP
VPN	Virtual Private Network
VPS	Virtual Private Server
WORM	Write Once Read Many

CHAPITRE 1

INTRODUCTION

1.1 Contexte

Au cours des dernières années, les ordinateurs sont devenus de plus en plus indispensables pour tout le monde. Comme cette technologie est largement disponible et son prix est très abordable, une vague de multiplication et une sous-utilisation des appareils électroniques accompagnée d'une augmentation exponentielle des coûts d'exploitation ont eu lieu. En effet, cette expansion incontrôlée a conduit à l'augmentation du personnel qualifié dans les centres de données, des coûts de la climatisation et de l'énergie, de l'achat et de la mise à jour des logiciels, etc.

En outre, le coût pour soutenir une infrastructure physique tentaculaire augmenta de façon spectaculaire. Ces augmentations ne furent pas uniquement appliquées au matériel, à l'énergie et à la climatisation, mais également à la gestion et à la maintenance. En conséquence, la sous-utilisation des serveurs est devenue un grand problème au sein des centres de données. D'ailleurs, International Data Corporation (IDC) (Boursas et al., 2008) a estimé que les serveurs en entreprise utilisent seulement 10 % à 15 % de la capacité totale de leurs ressources matérielles. Tous ces facteurs réduisent le retour sur investissement (RSI) et augmentent le coût total de possession (CTP).

Ainsi, l'utilisation de la technologie de la virtualisation est une réponse concrète afin de remédier à ces problèmes. Ce concept, qui a un impact considérable dans le domaine de l'informatique, offre une solution de valeur aux défis des technologies de l'information et de la communication (TIC) en minimisant la complexité des TIC et en réduisant les coûts d'exploitation. Selon Agarwal *et al.* (2012), la virtualisation permet souvent de déployer simultanément plusieurs systèmes d'exploitation sur une seule machine physique, sans perte de performance et avec une haute disponibilité, puisqu'elle conduit à avoir une redondance de services à moindre coût. De plus, elle permet aux entreprises de réduire les coûts de

matériel et d'exploitation de 50 % et les coûts énergétiques de 80 % en consolidant leurs ressources informatiques (VMware, 2012b).

De nouvelles technologies, où la virtualisation joue un rôle crucial, ont émergé récemment, en l'occurrence le nuage informatique qui est une technologie très prometteuse et en plein essor. Ce concept, que l'on évoque en tant que tremplin vers l'innovation technologique, se base principalement sur la grille informatique, la technologie SOA et évidemment, sur la virtualisation (Youseff, Butrico et Da Silva, 2008). En outre, il est en train de révolutionner le monde de l'informatique et est considéré comme l'architecture de la prochaine génération d'entreprise.

À l'heure actuelle, un grand nombre d'entreprises voit un intérêt notable dans le nuage. Il s'agit principalement d'un aspect financier, mais également de permettre à une société de se concentrer sur le cœur du métier. Avec l'apparition des solutions commerciales comme EC2 et S3 d'Amazon (Amazon, 2012a), Google App Engine (Google, 2012) et Microsoft Azure (Microsoft, 2012), le nuage est devenu plus qu'un simple concept, une réalité. En outre, la combinaison des dernières technologies du nuage informatique peut énormément contribuer à réduire les coûts, à assurer l'évolutivité et la flexibilité des services ainsi qu'à améliorer le rendement des applications. De plus, dans une infrastructure du nuage, l'utilisateur peut accéder aux applications à partir de n'importe quel appareil et n'importe où, et il ne paye que pour les services qu'il a réellement utilisés.

En raison de ses avantages, les entreprises sont de plus en plus nombreuses à étendre leur infrastructure technique en adoptant le nuage informatique. Cependant, le déploiement des applications dans le nuage informatique est un processus très complexe vu le grand nombre d'opérations nécessaires pour permettre un déploiement réussi telles que la restructuration de chaque couche de l'application pour le Cloud, la mise à l'échelle d'une façon automatisée en fonction de la demande, l'optimisation des différents services de l'application pour tirer parti des avantages du nuage et la gestion de stockage (Juve et Deelman, 2011). Selon HP (2012), ces activités de déploiement occupent jusqu'à 40% du temps de travail de l'opérateur alors

que la répétition des tâches de routines occupe jusqu'à 70% du temps des administrateurs de réseaux et de bases de données. Ceci est explicable par le fait que la majorité des applications utilisent plusieurs services et elles ont une architecture compliquée, souvent composée d'une base de données, d'un middleware et d'autres composants de configuration. Aussi, chaque service impose des exigences uniques sur l'environnement informatique qui l'héberge. De plus, pour déployer une application sur le nuage, une bonne préparation de l'environnement cible est requise pour qu'elle soit compatible avec son architecture, alors que pour migrer une application existante vers le nuage, une reconfiguration difficile de la connectivité des différentes entités de l'application et de la mise à l'échelle est souvent nécessaire.

En outre, contrairement aux solutions traditionnelles, où les services informatiques sont en bon contrôle physique, logique et personnel, le nuage informatique externalise les applications et les bases de données aux grands centres de données des prestataires spécialisés. Ainsi, l'adaptation d'applications existantes et le développement de nouveaux services de même que la gestion de données deviennent de plus en plus complexes. Ces processus de migration et de déploiement constituent un défi pour les utilisateurs de nuage, non seulement en ce qui a trait à la compatibilité et à l'interopérabilité des applications, mais encore en ce qui concerne la complexité du déploiement et de développement dans le nuage, où les principaux aspects tels que la mise à l'échelle des applications, la répétition des tâches récurrentes et la gestion du stockage de données n'ont pas bien été évalués.

Tel que rapporté dans (Chauhan et Babar, 2012), il n'y a pas assez de travaux dans la littérature sur le support de processus de déploiement des applications dans le nuage. Afin de rendre l'utilisation efficace d'un tel environnement, il est intéressant d'étudier comment nous pouvons franchir les défis et d'automatiser le déploiement des applications d'une manière reproductible tout en optimisant les ressources utilisées. Pour faire avancer ce domaine de recherche, notre travail consiste, dans la cadre de la présente recherche, à concevoir et à développer un nouveau modèle générique de déploiement automatisé des applications sur le nuage informatique, en tenant compte de la mise à l'échelle des applications et de leurs interopérabilités avec d'autres plateformes.

1.2 Problématique

Le nuage informatique est devenu une infrastructure importante pour le déploiement des applications. Il comporte de nombreux avantages indéniables. Un de ses aspects les plus attrayants est l'évolutivité à la demande des capacités et des ressources sans investissements initiaux. Ainsi, les utilisateurs et les organisations peuvent accéder à un large pool de ressources partagées basé sur le modèle de facturation à l'usage. Bien que le choix d'un tel environnement puisse paraître assez avantageux, les développeurs sont confrontés à plusieurs défis surtout en ce qui a trait à la migration et au déploiement des applications et des services dans le nuage.

Le déploiement de services dans le nuage est un processus souvent complexe en raison des difficultés rencontrées lors de la migration d'une application existante ou du développement d'un nouveau service. Lorsqu'une entreprise ou un établissement académique adopte le nuage, plusieurs défis doivent être relevés, et ce, peu importe l'offre en nuage choisie qu'il s'agisse de nuage privé, publique ou hybride. En effet, afin d'avoir un modèle générique de déploiement efficace et efficient, les questions auxquelles ce travail de recherche doit absolument répondre sont les suivantes :

Enfermement propriétaire et interopérabilité des applications : Selon l'enquête réalisée par North Bridge (2013), l'enfermement propriétaire et l'interopérabilité sont respectivement la troisième et la quatrième préoccupation majeures inhibant l'adoption du nuage informatique. Ce qui nous amène à la question, comment pouvons-nous éviter l'enfermement propriétaire? Comment pouvons-nous assurer l'interopérabilité et la portabilité des applications dans le nuage?

Complexité de déploiement des applications dans le nuage : Comment préparer l'environnement cible? Comment automatiser les principales tâches de routines et les tâches fastidieuses? Quelles sont les technologies qui doivent être supportées pour développer de nouveaux services sur le nuage? Quelles fonctionnalités intégrer pour réduire la complexité

du développement? Comment analyser les architectures des applications à migrer? Comment migrer les applications existantes vers le nuage? Comment traiter les erreurs de déploiement? Comment synchroniser le transfert automatique entre l'environnement source et cible? Comment gérer les noms de domaine des applications d'une façon automatisée ?

Mise à l'échelle automatique des applications : Comment automatiser l'élasticité ou la mise à l'échelle des applications déployées sur le nuage? Quelle décision faut-il prendre lorsque la demande augmente ou diminue? Quelle condition utiliser pour ajouter ou supprimer des instances de l'application? Quelle approche utiliser pour assurer l'évolutivité et l'extensibilité du système de déploiement?

Gestion de stockage au niveau IaaS[1] : Étant donné la nature dynamique de l'environnement du nuage, quel système de fichier choisir pour gérer le stockage de données? Quelles techniques utiliser pour gérer le stockage de métadonnées sur le nuage? Comment monter automatiquement les partitions des machines virtuelles d'une façon optimale?

Interrogation des utilisateurs : Comment traiter les requêtes des utilisateurs? La console web ou la ligne de commande est-il l'outil le plus efficace pour les développeurs? Comment modifier et mettre à jour les fichiers et le code source des applications? Comment surveiller les performances des applications déployées?

Nous avons présenté les grandes lignes de la problématique générale de déploiement et de la migration des applications dans le nuage informatique. Aussi, les objectifs à atteindre seront développés dans la prochaine section.

---

[1] IaaS : Infrastructure as a Service

1.3 Objectifs du mémoire

L'objectif principal de notre travail consiste à concevoir et à développer un nouveau modèle générique de déploiement automatique des applications dans le nuage informatique, afin de réduire la complexité de développement des applications, les coûts et les délais de mise en œuvre et de simplifier le processus de déploiement des services dans le nuage. De surcroît, supporter et déployer automatiquement des applications sur le nuage en assurant l'élasticité, la mise à l'échelle automatique et l'interopérabilité avec toutes les plateformes et en optimisant la gestion du stockage sont les objectifs primordiaux de ce mémoire.

Plus spécifiquement, nous devrions répondre à l'ensemble des problèmes soulevés dans la section précédente, à savoir i) l'enfermement propriétaire et l'interopérabilité des applications, ii) l'évolutivité du modèle de déploiement et la mise à l'échelle automatique des applications, iii) la réduction de la complexité de déploiement des applications dans le nuage, et iv) l'optimisation de la gestion de stockage au niveau IaaS.

Pour cela, nous commençons par le développement d'une plateforme en tant que service (PaaS) basée uniquement sur des technologies libres. Cette plateforme, que nous appelons OpenICRA, est un modèle de nuage qui devra mettre à la disposition des développeurs un environnement de programmation disponible immédiatement pour créer et déployer automatiquement des applications sur le nuage, dans un temps record et avec moins de complexité et de contraintes tant au niveau du développement qu'au niveau du déploiement. Ceci permet de répondre à l'ensemble des problèmes mentionnés dans la section précédente en offrant aux développeurs l'opportunité de se concentrer sur le langage de programmation et la satisfaction des clients au lieu de la configuration des systèmes d'exploitation, des serveurs et des outils de développement.

La concentration sur l'utilisation des piles logicielles et technologiques libres « open sources » et la non-intégration des solutions propriétaires nous permettent de construire une plateforme extensible, élastique et opérationnelle sur tout environnement de nuage. Ceci

garantit la liberté de choix, y compris la liberté de déployer toute application développée sur d'autres PaaS, l'indépendance et l'autonomie du modèle de déploiement. En isolant complètement l'application de l'environnement du nuage sous-jacent et en cachant les APIs et la complexité de configuration de l'environnement d'exécution, l'application peut être facilement déplacée d'un nuage à un autre sans aucune restriction ou modification du code source. Cela garantit la portabilité des applications à la fois sur et en dehors de la plateforme OpenICRA, empêchant ainsi l'enfermement propriétaire sur notre modèle proposé.

La technologie de la mise à l'échelle automatique est très nécessaire pour assurer l'évolutivité des applications déployées sur le nuage. En effet, cette technologie nous permet d'améliorer considérablement l'utilisation des ressources au niveau des serveurs et la qualité de service au niveau des utilisateurs finaux. Par conséquent, un développeur sera en mesure de configurer automatiquement l'augmentation ou la diminution des capacités des ressources selon la demande ou selon une configuration bien définie des règles de mise à l'échelle. Ceci est possible via une simple interface web d'administration de la plateforme. L'intégration de la technologie de l'équilibrage de charge avec la mise à l'échelle automatique contribue aussi à assurer la flexibilité et l'élasticité de la plateforme OpenICRA et ainsi, la garantie de la QoS.

Quant à l'automatisation de déploiement, le développement des modules et des classes extensibles et personnalisables décrivant le fonctionnement et la dépendance des différents composants et services de l'application à déployer sur le nuage permet de résoudre le problème de la répétition des tâches fastidieuses et d'automatiser le processus de déploiement des applications. Les classes développées doivent inclure la description de règles de la surveillance et de la mise à l'échelle automatique des services et des ressources de calcul et de stockage. Ceci permet de simplifier le déploiement, d'accélérer la mise en œuvre des applications et de réduire la complexité et les erreurs de déploiement ainsi que les coûts.

Pour la gestion de stockage, nous nous intéressons aux systèmes de fichiers distribués (DFS) pour résoudre le problème du rapport entre la croissance du volume des images des VMs

avec la performance des nœuds de calcul. L'intégration du DFS au niveau IaaS permet d'offrir un excellent environnement pour les machines virtuelles et les applications où l'évolutivité est un problème à la fois en termes de vitesse de calcul et de stockage. Aussi, l'utilisation d'un module de montage de volume comme interface entre le système de fichiers distribué et les nœuds de calcul résout le problème d'accès aux données par les machines virtuelles. Une telle intégration rend le système de stockage efficace, évolutif et capable de gérer les images volumineuses des machines virtuelles.

En ce qui concerne le déploiement des services de télécommunication, il est très utile d'intégrer des technologies de pointe comme les APIs de WebRTC (Bergkvist et al., 2012), les services de web-conférence de Red5 (Red5 Team, 2012), le partage d'écran, l'annotation en temps réel et les services d'interconnexion entre les réseaux PSTN et IP, afin d'élargir les services offerts par le modèle proposé. Ces intégrations ont pour but de faciliter la centralisation des outils de développement au même endroit et la création des applications de télécommunication. Également, il est important de développer une procédure d'affectation automatique des noms de domaines personnalisés aux espaces de stockage des utilisateurs avec des logiciels de gestion de DNS. L'utilisation des langages de programmation comme PHP et HTML5, surtout au niveau du développement de l'interface cliente, permet aux développeurs de créer des applications faciles à utiliser et qui n'exigent du client aucun logiciel ou plug-in à installer. De plus, il est très intéressant d'intégrer des outils de monitorage en temps réel pour surveiller l'évolution de déploiement et les performances des applications déployées par notre modèle sur différents environnements de nuage tels qu'Amazon web service (Amazon, 2012b), GSN (Synchromedia, 2010), etc., afin de vérifier le fonctionnement des services et de déclencher les méthodes de correction appropriées en cas du besoin.

1.4 Plan du mémoire

La suite de ce mémoire s'articule autour de quatre chapitres, en plus de l'introduction. Dans le deuxième chapitre, nous présentons une revue de littérature des concepts de la virtualisation et du nuage informatique, de leurs impacts sur les environnements TI ainsi qu'une critique des solutions de déploiement des applications dans le nuage déjà existantes. Le troisième chapitre présente les différentes méthodes suivies pour atteindre les objectifs du mémoire de recherche ainsi que les détails du processus de déploiement automatique des applications dans le nuage informatique. Le quatrième chapitre illustre les hypothèses et la conception du modèle proposé et détaille les différentes études comparatives selon des critères bien étudiés pour choisir les solutions technologiques aux niveaux PaaS et IaaS et qui seront utilisées dans la réalisation de ce projet de recherche. Le cinquième chapitre illustre deux cas d'études concrets d'automatisation de déploiement et de migration vers des environnements de nuage réels, et discute les expérimentations effectuées pour valider notre modèle proposé OpenICRA. Enfin, la conclusion de ce travail récapitule le bilan de ce qui est réalisé et les recommandations pour les futurs travaux tant pour les développeurs que pour les utilisateurs du nuage.

La figure 1.1 présente le schéma synoptique de notre système et les différents éléments présentés dans ce mémoire.

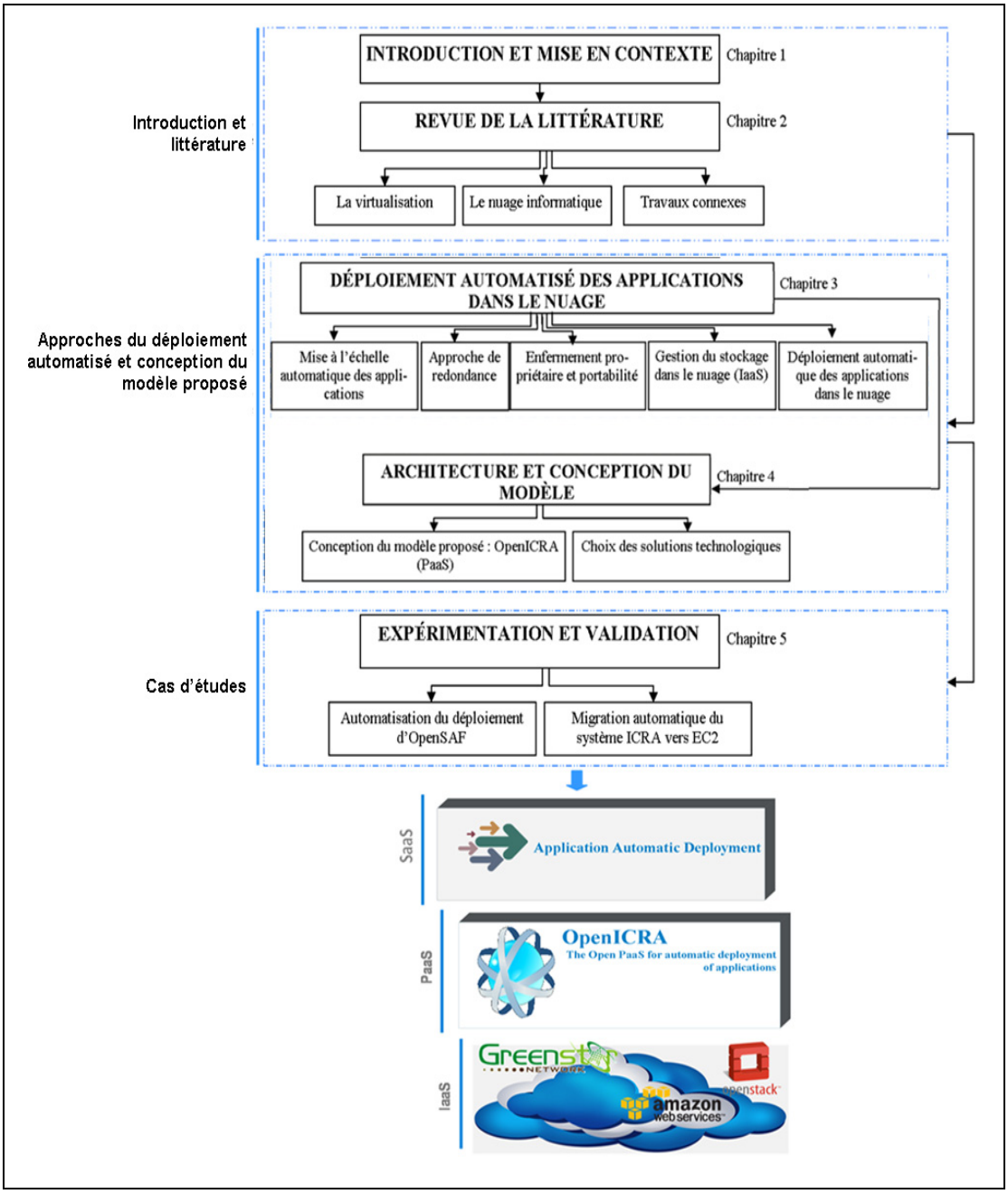

Figure 1.1 Schéma synoptique de notre système et des différents éléments présentés dans ce mémoire

CHAPITRE 2

REVUE DE LA LITTÉRATURE

2.1 Introduction

Ce chapitre vise à présenter les principales notions théoriques nécessaires à la compréhension du problème de ce projet de recherche. En effet, il s'articule autour de trois sections. Nous présentons dans la première, le concept de la virtualisation, ses modèles d'utilisation et ses différentes techniques. La seconde section expose le principe du nuage informatique et ses modèles de service et de déploiement. Enfin, la dernière est consacrée aux critiques des solutions déjà existantes.

2.2 La virtualisation

Cette section présente un aperçu général de la virtualisation incluant sa définition, ses modèles d'utilisation et ses avantages ainsi que ses différentes techniques.

2.2.1 Définitions

La virtualisation est une technologie qui fait référence à l'abstraction ou le masquage des ressources d'un serveur physique pour les faire apparaître logiquement différentes à ce qu'elles sont physiquement. Elle permet de faire fonctionner simultanément plusieurs systèmes d'exploitation qui peuvent être hétérogènes (par exemple Linux, Windows, Mac…) sur une seule machine. Un serveur virtualisé est communément appelé une machine virtuelle (VM). Ainsi, un système principal s'interpose entre la machine physique et l'ensemble des machines virtuelles selon le type de la virtualisation choisi. Ce dernier, qui peut être un système d'exploitation ou un logiciel spécifique, permet d'émuler le matériel de la machine physique pour le simuler aux systèmes virtualisés et de les gérer aisément (Nicolas et Julien, 2009).

Le virtualiseurs, qui créent et exécutent les machines virtuelles, sont appelés VMM (Virtual Machine Manager) ou aussi hyperviseur. Ces derniers sont classés actuellement en deux catégories :

- l'hyperviseur de **type 1** illustré à la figure 2.1, également connu comme natif ou « bare metal », est un logiciel qui s'exécute directement sur le matériel de la machine hôte. Les systèmes d'exploitation invités s'exécuteront sur un autre niveau au-dessus de l'hyperviseur. Ce type d'hyperviseur dispose d'un accès de bas niveau directement au matériel, offrant ainsi des meilleures performances. VMware ESX/ESXi, XenServer et KVM sont des exemples de l'hyperviseur de type 1.

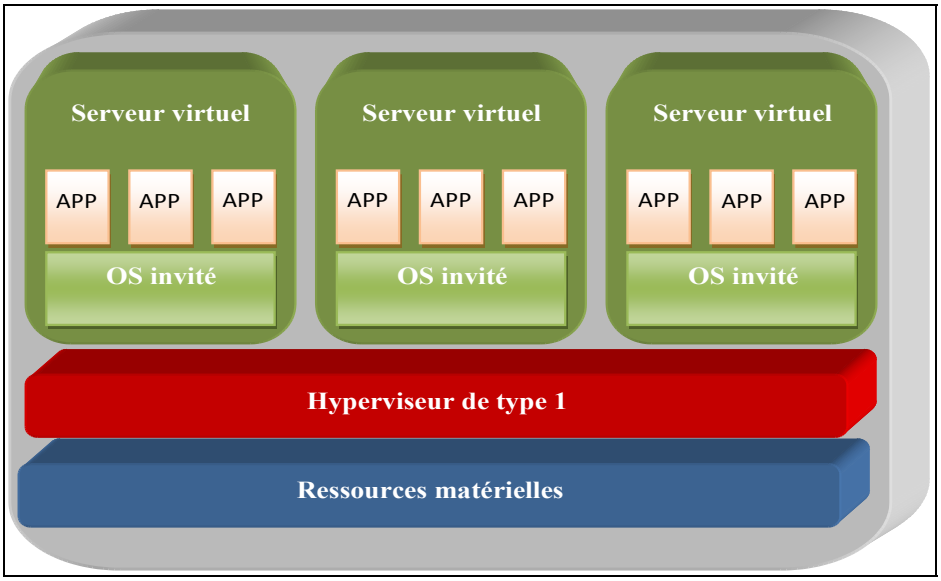

Figure 2.1 Hyperviseur de type 1

- l'hyperviseur de **type 2** illustré à la figure 2.2 est un logiciel qui s'exécute à l'intérieur d'un système d'exploitation préinstallé. Ce type d'hyperviseur est appelé également « host-based ». Avec la couche hyperviseur comme un second niveau logiciel distinct, les systèmes d'exploitation invités s'exécuteront au troisième niveau au-dessus du matériel. Les hyperviseurs de type 2 sont moins efficaces que ceux de type 1, mais la facilité d'utilisation de ce type de virtualisation représente un grand avantage.

VirtualBox, VMware Server et VMware Workstation sont des exemples de l'hyperviseur de type 2.

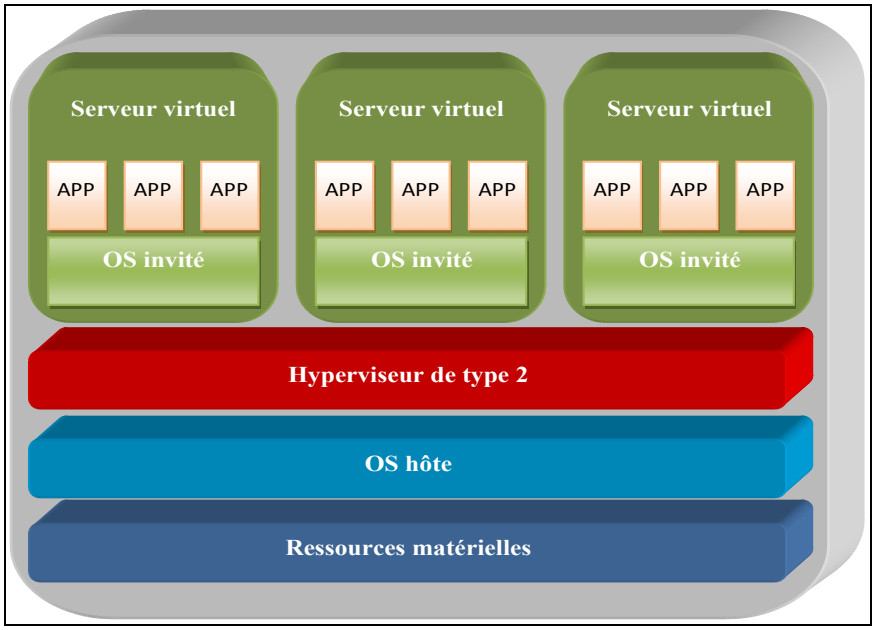

Figure 2.2 Hyperviseur de type 2

2.2.2 Les modèles d'utilisation de la virtualisation et ses avantages

L'optimisation de l'utilisation des ressources physiques, la gestion et la fiabilité des serveurs sont connues parmi les avantages classiques de la virtualisation. Cependant, il est devenu possible avec l'évolution de cette technologie de partager facilement un serveur virtualisé malgré les différentes exigences des systèmes d'exploitation, de les mettre à niveau à travers des machines virtuelles pour minimiser le temps d'indisponibilité, et d'isoler les défaillances des logiciels dans les machines virtuelles où elles se produisent. Bien que ces avantages soient traditionnellement considérés comme utiles dans les systèmes informatiques haut de gamme, la virtualisation a apporté de nouveaux modèles d'utilisation très avancés (Uhlig et al., 2005). Cette section illustre les différentes catégories des capacités fonctionnelles qui englobent plusieurs types d'usages de la virtualisation.

2.2.2.1 Isolation des applications

La virtualisation permet d'améliorer la sécurité globale et la fiabilité des systèmes en isolant les couches logicielles dans leurs propres machines virtuelles. En effet, comme le montre la figure 2.3, la sécurité peut être améliorée, car les intrusions peuvent se limiter à la machine virtuelle vulnérable, tandis que la fiabilité peut être renforcée, car les échecs logiciels dans une VM n'affectent pas les autres machines virtuelles.

L'isolation améliore aussi la disponibilité des machines virtuelles même si elles partagent les ressources physiques d'un même serveur. Ainsi, les applications exécutées sont plus disponibles dans un environnement virtuel que dans un système traditionnel, non virtualisé (VMware, 2012b).

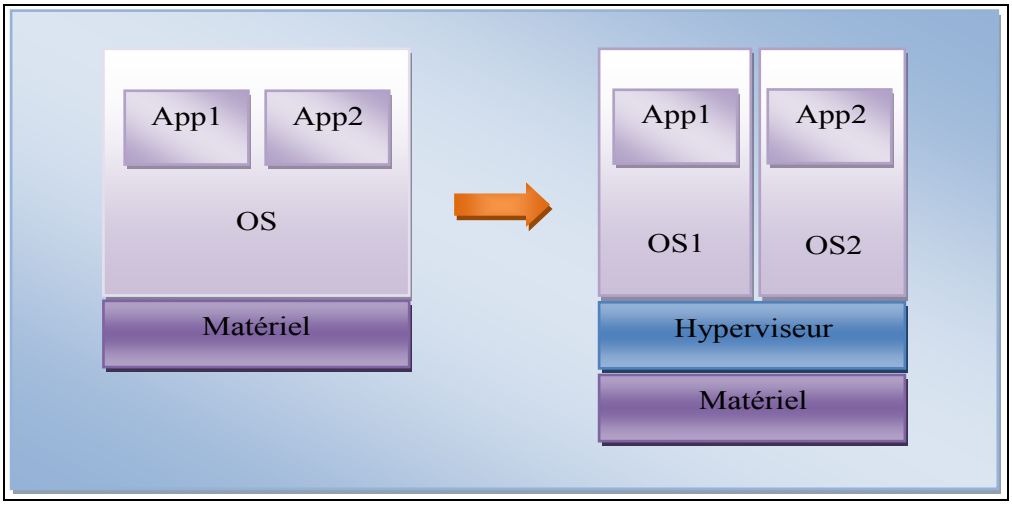

Figure 2.3 Isolation des applications

2.2.2.2 Consolidation des serveurs

La prolifération et la sous-utilisation des serveurs, telles que le déploiement d'un seul système d'exploitation et un serveur Web ou un serveur de téléchargement de fichier unique, sont très contestées par les centres de données (Uhlig et al., 2005). En conséquence, la virtualisation est une réponse à cette multiplication. Elle consiste à consolider les

applications individuelles sur un seul serveur physique en le partitionnant en plusieurs machines virtuelles indépendantes, comme illustré à la figure 2.4.

Grâce à la consolidation des serveurs, le taux d'utilisation augmente jusqu'à 80 % comparé au 5 à 15 % sur les machines non virtualisés (Boursas et al., 2008; VMware, 2012b). Cette augmentation des taux d'utilisations contribue aux réductions des coûts de 20 à 30 % par application (VMware, 2012b).

La consolidation permet aussi de résoudre le problème de manque d'espaces dans les locaux informatiques en minimisant le nombre des serveurs utilisés et les coûts de construction des centres de données. En ce qui concerne le côté environnemental, elle contribue à la réduction des serveurs recyclés lors de leur fin de vie et à la réduction de l'émission de gaz à effet de serre en minimisant l'énergie de l'électricité et de la climatisation. Également, elle aide à supprimer les risques de pertes économiques dues à des pannes des centres de données.

Ainsi, la gestion des mises à niveau présente une autre préoccupation pour les responsables informatiques. En effet, lorsqu'un nouveau matériel ou une nouvelle version de système d'exploitation devient disponible, les défis de support et d'incompatibilité des logiciels existants souvent nécessitent une entière mise à niveau de la plateforme de l'entreprise. La consolidation résout ce problème en permettant aux systèmes informatiques d'exécuter simultanément les applications existantes et les nouvelles versions des systèmes d'exploitation (Uhlig et al., 2005).

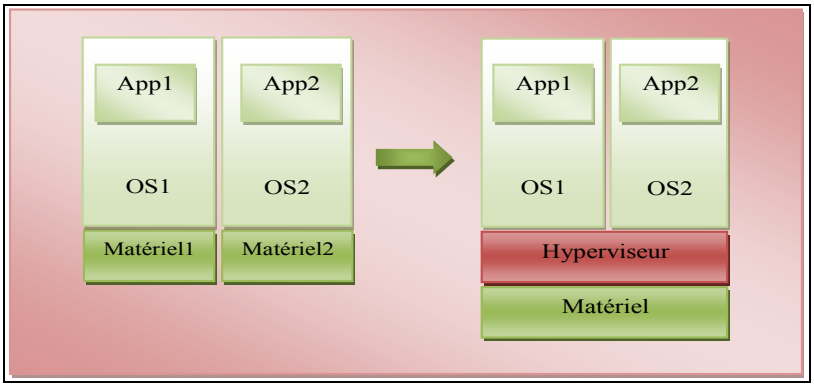

Figure 2.4 Consolidation des serveurs

2.2.2.3 Migration des machines virtuelles

En encapsulant l'état complet d'une machine virtuelle dans un ensemble de fichiers sur un système de stockage partagé, la virtualisation permet de détacher l'invité du matériel sur lequel il est en cours d'exécution et de faire le migrer à chaud vers une plate-forme physique différente. De plus, la migration des VMs facilite les opérations de maintenance du matériel et elle peut être déclenchée automatiquement par un équilibreur de charge ou par les agents de prédiction de défaillance. Cette capacité offre une meilleure qualité de service à un moindre coût d'exploitation. La figure 2.5 représente d'une façon simplifiée ce modèle d'utilisation de la virtualisation.

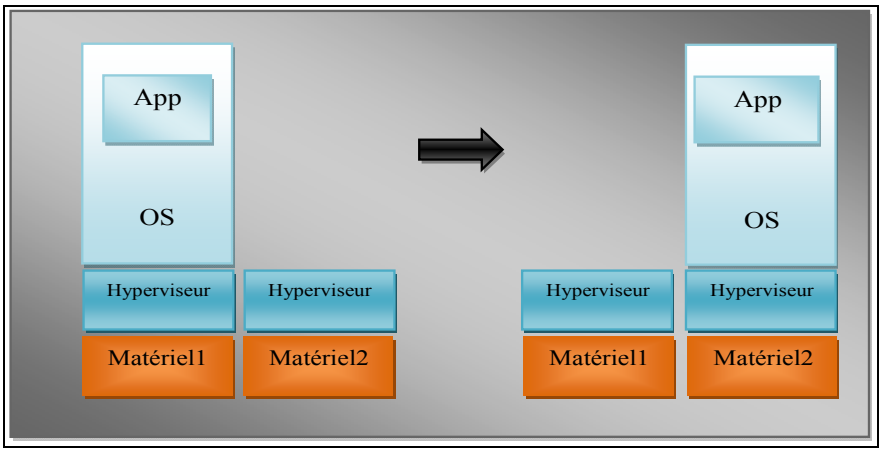

Figure 2.5 Migration des machines virtuelles

Le module vMotion de la suite vSphere de VMware permet de faire la migration à froid d'une machine virtuelle lorsqu'elle est éteinte. Également, comme illustré à la figure 2.6, cette technologie permet de migrer à chaud les machines virtuelles en cours d'exécution en les déplaçant instantanément d'un serveur physique à un autre sans interruption de service (Clark et al., 2005).

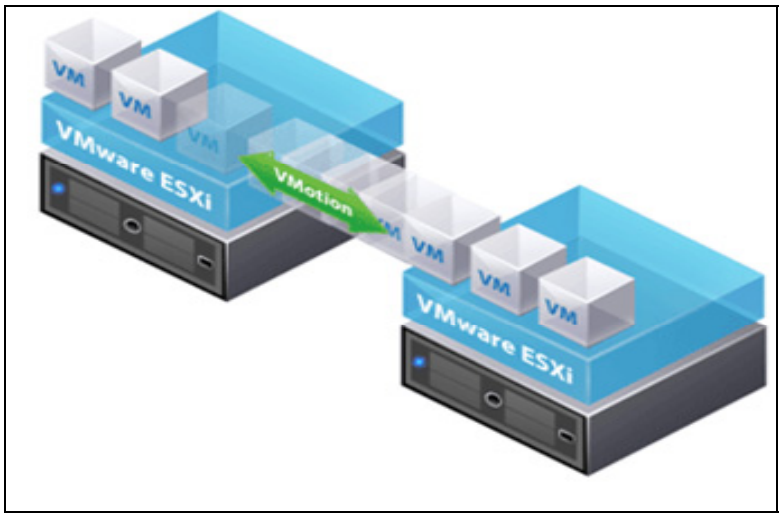

Figure 2.6 Migration à chaud des machines virtuelles avec le module vMotion de VMware
Tirée de Neoflow (2012)

2.2.2.4 Test et développement

Dans le secteur de développement, la virtualisation offre aux développeurs un environnement efficace de débogage de leurs programmes et de test de la portabilité de plusieurs systèmes d'exploitation sur un seul ordinateur. Ainsi, elle est très utile dans le domaine d'enseignement surtout lorsqu'on a besoin d'isoler les machines dont le but d'empêcher les étudiants de communiquer entre eux lors de devoirs. De plus, l'utilisation d'une machine virtuelle permet souvent un déploiement rapide en isolant l'application dans un environnement connu et contrôlé. En copiant simplement une image virtuelle de sauvegarde, une perte de temps de réinstallation peut être évitée au moment des accidents graves.

2.2.2.5 L'équilibrage de charge dynamique et la récupération après les sinistres

Comme les charges des serveurs varient, la virtualisation offre la possibilité aux machines virtuelles qui utilisent beaucoup des ressources d'un serveur d'être déplacées vers les serveurs sous-utilisés. Cet équilibrage de charge dynamique crée une utilisation efficace des ressources des serveurs.

La récupération après les sinistres est un élément essentiel pour un centre de données vu que les pannes des systèmes informatiques peuvent créer d'énormes pertes économiques. De plus, la migration à chaud améliore jusqu'à 85 % du délai de récupération après une panne et elle permet à une machine virtuelle d'être instantanément migrée vers un autre serveur lorsqu'une défaillance de machine virtuelle a été détectée (VMware, 2012b).

2.2.3 Les différentes techniques de la virtualisation

Différentes sont les techniques qui ont pour but la création d'un environnement virtualisé pour des applications et des systèmes d'exploitation. Ces techniques offrent diverses façons de mettre en œuvre la virtualisation. Dans cette section, nous expliquons les méthodes de la virtualisation les plus courantes pour mieux comprendre leurs spécificités.

2.2.3.1 L'isolation

L'isolation est une technique qui a pour but d'emprisonner l'exécution des applications les unes des autres dans des contextes indépendants (Donnette et Hannequin, 2007). Une des principales fonctions de l'isolateur est de faire tourner plusieurs instances d'une même application sur le même système hôte, même si elle n'était pas conçue pour être exécutée dans un mode multi-instance.

Cette technique, comme l'illustre la figure 2.7, isole les applications pour les rendre capables de communiquer qu'avec les processus et les ressources associés à leur propre environnement cloisonné. Cette solution, qui est uniquement liée aux systèmes Linux, offre des performances supérieures et elle permet de partitionner un serveur en plusieurs contextes. Aussi, elle permet de protéger les systèmes d'exploitation contre des intrusions utilisant les faiblesses des applications, car seul l'environnement cloisonné sera impacté par l'intrusion.

Quelques exemples courants d'isolateurs sont:

chroot : changement de répertoire racine d'un processus ;

Linux-VServer : isolation des processus en espace utilisateur ;

BSD Jail : isolation en espace utilisateur ;

OpenVZ : partitionnement au niveau noyau (Donnette et Hannequin, 2007).

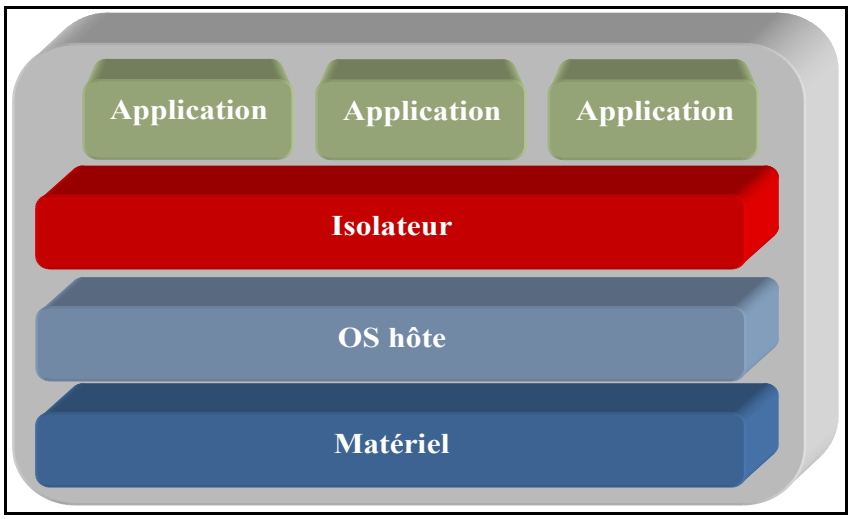

Figure 2.7 Isolateur

2.2.3.2 La virtualisation complète

L'émulation de matériel prend en charge de réels systèmes d'exploitation invités, et les applications tournant sur chacun de ces systèmes sont exécutées dans des environnements totalement isolés. De cette façon, plusieurs machines virtuelles peuvent être présentes sur une seule machine physique, chacune entièrement indépendante des autres. En effet, le moniteur de machine virtuelle (VMM) fournit une émulation complète du matériel pour le système d'exploitation invité, c'est pour cette raison que cette technique de virtualisation, illustrée à la figure 2.8, est appelée **virtualisation complète**.

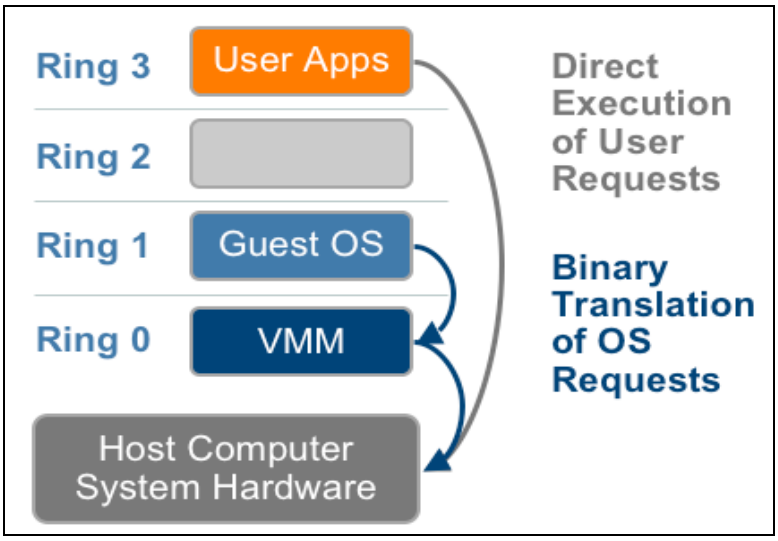

Figure 2.8 Approche de la virtualisation complète
Tirée de VMware (2007)

Ainsi, la technique de la virtualisation complète est utilisée pour supporter les systèmes d'exploitation invités **non modifiés**. Cela signifie que tous les logiciels, les systèmes d'exploitation et les applications qui s'exécutent en mode natif sur le matériel peuvent également s'exécuter sur la machine virtuelle. De surcroît, le terme non modifié fait référence aux noyaux des systèmes d'exploitation qui n'ont pas été modifiés pour tourner sur l'hyperviseur, par conséquent, ce dernier exécute leurs opérations privilégiées de la même façon qu'elles sont exécutées sur le ring 0 du CPU.

Cette approche permet de traduire le code du noyau pour remplacer les instructions non virtualisables par de nouvelles séquences d'instructions qui ont l'effet souhaité sur le matériel virtuel. Entre-temps, le code au niveau utilisateur, qui est considéré parmi les instructions non sensibles, est directement exécuté sur le processeur pour renforcer l'efficacité de la virtualisation et garantir la sécurité du système. L'hyperviseur fournit à chaque machine virtuelle tous les services du système physique, y compris un BIOS virtuel, des périphériques virtuels et une mémoire virtuelle.

La virtualisation complète combine la traduction binaire et l'exécution directe. Ainsi, le système d'exploitation invité n'a pas conscience d'être virtualisé et il n'exige donc aucune

modification. La virtualisation complète est la seule option qui ne nécessite ni l'assistance matérielle ni la paravirtualisation pour virtualiser les instructions sensibles et privilégiées. L'hyperviseur traduit toutes les instructions du système d'exploitation instantanément et il met en cache les résultats pour une utilisation future, tandis que les instructions au niveau utilisateur s'exécutent sans modification à vitesse native et sans dégradation de performance.

Selon VMware (2007), la virtualisation complète offre la meilleure façon d'assurer l'isolation et la sécurité pour les machines virtuelles, et elle simplifie la migration et la portabilité puisque la même instance du système d'exploitation invité peut s'exécuter dans un environnement virtuel ou bien sur un matériel natif.

2.2.3.3 La paravirtualisation

La paravirtualisation est une autre approche de virtualisation des serveurs. Elle consiste à modifier le noyau du système d'exploitation invité pour remplacer les instructions non virtualisable par des hyperappels qui communiquent directement avec la couche de virtualisation. L'hyperviseur fournit également des interfaces hyperappel pour d'autres opérations critiques du noyau comme la gestion mémoire, la gestion des interruptions et le chronométrage.

La paravirtualisation, où le système d'exploitation invité a conscience d'être virtualisé, est différente de la virtualisation complète, où le système d'exploitation non modifié n'a pas conscience d'être virtualisé et ses instructions sensibles sont remplacées par des trappes à l'aide de la traduction binaire. La réduction de la surcharge est parmi les points forts de la paravirtualisation, mais son avantage de performance par rapport à la technique de la virtualisation complète peut varier considérablement en fonction de la charge de travail.

Puisque la paravirtualisation ne peut pas supporter les systèmes d'exploitation non modifiés (par exemple, Windows 2000/XP), sa compatibilité et sa portabilité restent limitées. La paravirtualisation peut également introduire des problèmes importants de support et de mise à

jour dans les environnements de production, car elle nécessite de profondes modifications du noyau du système d'exploitation. La solution libre Xen est un exemple de paravirtualisation qui virtualise le processeur et la mémoire en utilisant un noyau Linux modifié et elle virtualise les entrées sorties à l'aide des pilotes personnalisés des périphériques du système d'exploitation invité.

Bien qu'il soit très difficile de construire un meilleur support de traduction binaire sophistiqué pour la virtualisation complète, la modification du système d'exploitation invité est relativement facile pour assurer la paravirtualisation. Ainsi, l'accélération de la traduction binaire pour améliorer les performances de la virtualisation est très difficile. Par conséquent, plusieurs logiciels tels que VMware, Xen et KVM ont utilisé cette technique de paravirtualisation pour remédier au problème de performance.

Comme l'illustre la figure 2.9, la paravirtualisation essaie de résoudre les problèmes de la virtualisation complète en permettant aux systèmes d'exploitation invités d'accéder directement au matériel sous-jacent, ce qui améliore la communication entre le système d'exploitation invité et l'hyperviseur pour renforcer la performance et l'efficacité.

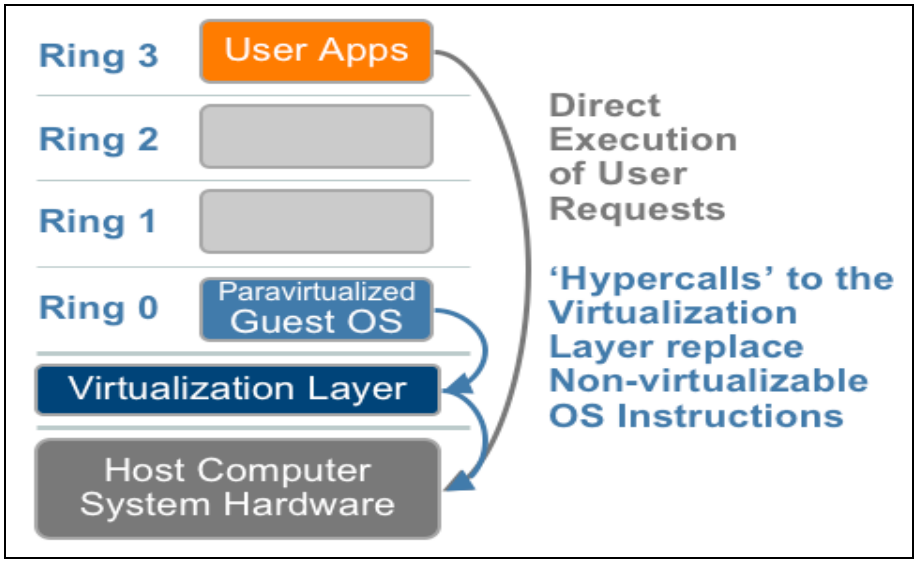

Figure 2.9 Approche de la paravirtualisation
Tirée de VMware (2007)

2.2.3.4 La virtualisation assistée par matériel

Comme la virtualisation assistée par matériel sert à intégrer le support de la virtualisation dans les processeurs, plusieurs vendeurs de matériel ont lancé rapidement des projets de développement de nouvelles fonctionnalités pour simplifier cette technique. Les améliorations de la première génération incluent la technologie VT-x (Virtualization Technology) d'Intel ainsi que l'AMD-V de la compagnie AMD. Ces deux technologies ciblent les instructions privilégiées en introduisant une nouvelle caractéristique de mode d'exécution de CPU, permettant au VMM de s'exécuter dans un nouveau mode racine (root) au-dessous du niveau d'exécution ring 0. Comme le montre la figure 2.10, les instructions privilégiées et sensibles sont configurées pour être virtualisées automatiquement par l'hyperviseur dans le matériel à l'aide d'un modèle classique *trapper-et-émuler*, supprimant ainsi la nécessité de passer par la traduction binaire ou la paravirtualisation.

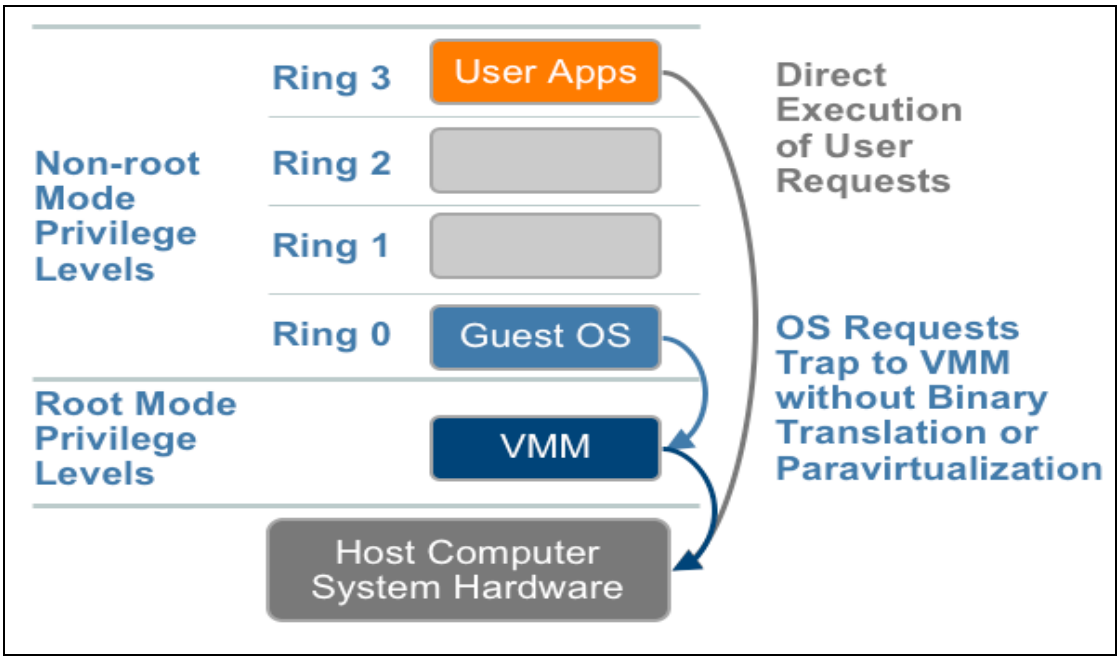

Figure 2.10 Approche de la virtualisation assistée par matériel
Tirée de VMware (2007)

L'état du système invité est stocké dans une structure de contrôle de machine virtuelle (VT-x) pour Intel ou dans un bloc de contrôle de machine virtuelle (AMD-V) pour AMD. Ces

processeurs d'Intel VT et AMD-V sont devenus disponibles en 2006, alors que seulement les systèmes les plus récents contiennent ces fonctionnalités d'assistance matérielle. Autrement, si la machine physique n'est pas équipée par un processeur récent, alors il est nécessaire d'avoir une machine qui supporte cette technique de virtualisation (Julien et al., 2008).

Ces nouvelles améliorations introduites par les vendeurs de matériel, tels qu'Intel et AMD ont rendu les hyperviseurs capables de se charger à l'intérieur de matériel, permettant aux applications de tourner sur le niveau d'exécution le plus élevé ring 3 et aux systèmes d'exploitation invités de fonctionner sans aucune modification sur ring 0. Les systèmes d'exploitation peuvent également accéder rapidement au CPU comme ils sont installés directement sur une machine physique. Cela a pour avantage d'avoir de meilleures performances.

2.3 Le nuage informatique

Dans cette section, nous présentons tout d'abord le concept du nuage informatique et ses technologies connexes. Ensuite, nous détaillons ses caractéristiques et, finalement, nous exposons ses modèles de services et de déploiement.

2.3.1 Nuage informatique : qu'est-ce que c'est?

Le nuage informatique connu par le terme anglais «Cloud Computing» représente aujourd'hui une véritable révolution pour l'industrie informatique. Il est donc défini comme un concept qui fait référence à l'utilisation des outils technologiques et des capacités de calcul et de stockage des serveurs répartis dans le monde entier, fournis en tant que service auxquels on peut recourir sur Internet indépendamment de l'appareil qui est utilisé pour y avoir accès (Patrick, 2011).

Avec le nuage, la puissance de calcul et la capacité de stockage de l'information sont disponibles à la demande et facturées en fonction de l'utilisation réelle permettant une

réduction drastique des coûts. Par rapport à ceux qui installent leur propre infrastructure, les utilisateurs du nuage peuvent énormément économiser leurs dépenses et tirer profit de la fiabilité et de l'évolutivité du système. De plus, les utilisateurs peuvent tirer parti de cette puissance des ressources informatiques potentiellement illimitées pour réaliser des tâches de calcul intensif et accéder aux systèmes de stockage de masse (Chaisiri, Lee et Niyato, 2012).

De ce fait, le nuage informatique consiste à externaliser les applications et les bases de données aux grands centres de données des prestataires spécialisés. Ces derniers fournissent aux utilisateurs les services quelque soit leur nature (logiciel, matériel, plate-forme, ou stockage) via Internet par une application standard, facilement disponible, souvent un navigateur web (Ahmed, Chowdhury et Rafee, 2012).

Même si les ancêtres du nuage sont assez anciens tels que la virtualisation qui était déjà apparue depuis 1960 et la grille informatique depuis 1990, le nuage informatique est né officiellement le 24 août 2006, date à laquelle Amazon lance sa version d'essai d'Elastic Compute Cloud (EC2). Cette offre, qui se caractérisait par sa capacité de calcul flexible, était destinée aux développeurs qui ne voulaient pas déployer leur propre infrastructure et qui louaient l'infrastructure existante d'Amazon via Internet pour mettre en œuvre leurs applications (Singh et Nain, 2012).

Le terme « Cloud Computing » est devenu populaire en 2007, lorsque cette expression est apparue pour la première fois dans l'édition anglaise Wikipédia, le 3 mars 2007. Cette année-là, Dell a tenté de déposer la marque « Cloud Computing ». Sa demande a été acceptée en juillet, mais l'autorisation est révoquée quelques jours plus tard (Singh et Nain, 2012).

Après plusieurs années de débats et la publication d'une quinzaine de brouillons, le National Institute for Standards and Technology (NIST) a publié, en septembre 2011, la version finale de la définition du nuage informatique.

Selon la définition officielle du NIST, « Le nuage informatique est un modèle pratique, à la demande, permet d'établir un accès, par le réseau, à un réservoir partagé de ressources informatiques configurables (réseau, serveurs, stockage, applications et services) qui peuvent être rapidement approvisionnées et libérées en minimisant les efforts de gestion ou les interactions avec le fournisseur de service » (Mell et Grance, 2011).

Aujourd'hui, cette technologie a pris une réelle ampleur et plusieurs grandes entreprises du secteur IT telles qu'Amazon, IBM, Microsoft, Google… se sont investies dans la maturation du nuage informatique en proposant la location de leur infrastructure.

La définition du NIST énumère cinq caractéristiques essentielles du nuage informatique : accès via le réseau, à la demande et en libre-service, ressources informatiques virtualisées et mutualisées, élasticité ou mise à l'échelle rapide, et service mesuré. Elle énumère également les trois modèles de services (logiciel en tant que service, plate-forme en tant que service et infrastructure en tant que service), et les quatre modèles de déploiement (privé, communautaire, public et hybride) (Mell et Grance, 2011).

2.3.2 Les ancêtres du nuage

Les ancêtres les plus directs du nuage informatique sont la virtualisation, le Grid Computing et l'Utility Computing. À ce niveau, le nuage, qui se base principalement sur ces technologies, est souvent comparé avec ses concepts dont chacun a certaines caractéristiques communes avec le Cloud (Claranet, 2013). Comme le concept de la virtualisation a été détaillé dans la première section de ce chapitre, nous présentons dans cette sous-section seulement les deux autres technologies.

2.3.2.1 Grid Computing

Le « *Grid Computing* » est un terme faisant référence à la combinaison d'un ensemble de ressources informatiques distribuées, hétérogènes, voire délocalisées et autonomes. Ces

ressources ont un objectif commun se représente dans la constitution d'une infrastructure virtuelle performante offrant une très grande capacité de calcul. En utilisant ce concept, des grandes et des complexes opérations de calcul peuvent être réalisées en divisant la charge de travail en des sous-tâches qui sont traitées en parallèle par des ordinateurs dispersés géographiquement.

Dans sa forme la plus simple, le grid computing peut être représenté comme un « super ordinateur virtuel» composé de plusieurs ordinateurs interconnectés en réseau qui agissent ensemble pour accomplir des tâches faramineuses.

Un des projets les plus célèbres du Grid Computing est Folding@Home. Ce projet consiste à utiliser les puissances de calcul inutilisées des milliers d'ordinateurs pour traiter un problème scientifique complexe. Son but est de simuler le repliement de protéine et d'autres phénomènes biomoléculaires afin de découvrir de nouveaux médicaments, notamment contre la maladie d'Alzheimer, la drépanocytose et certains types de cancers (Beberg et al., 2009).

2.3.2.2 Utility Computing

L'« *Utility Computing* » est un modèle qui consiste à fournir sur demande des ressources informatiques matérielles, logicielles ainsi que d'autres services, et ce en offrant un mode de facturation à l'usage plutôt que sur une base forfaitaire. Comme d'autres modèles informatiques à la demande tels que le grid computing, l'utility computing vise à maximiser l'utilisation efficace des ressources et à réduire les coûts associés.

Le mot « Utility » est utilisé pour faire une analogie avec d'autres services, comme l'énergie électrique, qui visent à satisfaire les besoins des clients. Cette approche, connue également par le mode de paiement à l'usage « pay-per-use » ou service mesuré, est devenue de plus en plus un terme courant dans l'informatique d'entreprise et est parfois utilisé pour le marché de la consommation ainsi pour le service Internet, l'accès au site Web, le partage de fichiers et d'autres applications.

2.3.3 Caractéristiques du nuage

Selon Mell et Grance (2011), le service de nuage informatique proposé par le prestataire doit répondre à un certain nombre de critères. Les cinq caractéristiques exigées sont les suivantes :

Libre-service à la demande : L'utilisateur doit pouvoir allouer et libérer des capacités informatiques (espace de stockage, bande passante, temps de serveur) suivant ses besoins, automatiquement sans requérir une interaction humaine avec le fournisseur de service (par exemple via des API).

Accès étendu au réseau : Les ressources sont disponibles sur le réseau et accessibles par le biais des mécanismes standards, qui favorisent l'accès au service par des clients lourds ou légers à travers des plates-formes hétérogènes (par exemple : téléphone mobile, ordinateurs portables et stations de travail).

Mise en commun des ressources : Les ressources informatiques (stockage, traitement, mémoire et bande passante réseau) du fournisseur sont mises en commun pour servir plusieurs utilisateurs, avec des attributions et libérations dynamiques des différentes ressources physiques et virtuelles en fonction de la demande. Le client ne contrôle pas l'emplacement réel des ressources fournies et il ne sait pas où elles se trouvent physiquement. Cependant, il peut parfois avoir la possibilité de choisir une localisation à un niveau plus haut d'abstraction (par exemple, le pays, l'état, ou le centre de données).

Élasticité ou mise à l'échelle rapide : Les ressources sont allouées et libérées de façon élastique, parfois automatique, pour dimensionner rapidement la capacité, de l'augmenter et de la diminuer en fonction des modifications des besoins de calcul. Pour le client, cette allocation de ressources s'avère souvent illimitée et peut être faite à tout moment et dans la quantité voulue.

Service mesuré : Les systèmes du nuage contrôlent automatiquement et optimisent l'utilisation des ressources en fonction des informations remontées par des fonctionnalités de mesures, selon le type de service (par exemple, stockage, traitement, bande passante, et comptes d'utilisateurs actifs). L'utilisation de ces ressources peut être surveillée et contrôlée en toute transparence pour le fournisseur et le consommateur du service utilisé.

2.3.4 Modèles de service

Le nuage informatique est un modèle d'entreprise qui fournit des ressources informatiques (physiques et virtuelles) en tant que service d'une façon flexible et en fonction de la demande. La figure 2.11 illustre l'architecture de ce concept qui peut être divisée en quatre couches : la couche matérielle, la couche infrastructure, la couche plateforme et la couche application. Conceptuellement, chaque couche de l'architecture illustrée dans cette figure peut être implémentée comme un service de la couche supérieure. Inversement, chaque couche peut être perçue comme un client de la couche inférieure. Cependant, dans la pratique, le nuage offre des services qui peuvent être regroupés en trois catégories: Logiciel en tant que service (SaaS), Plate-forme en tant que service (PaaS) et Infrastructure en tant que service (IaaS) (Zhang, Cheng et Boutaba, 2010).

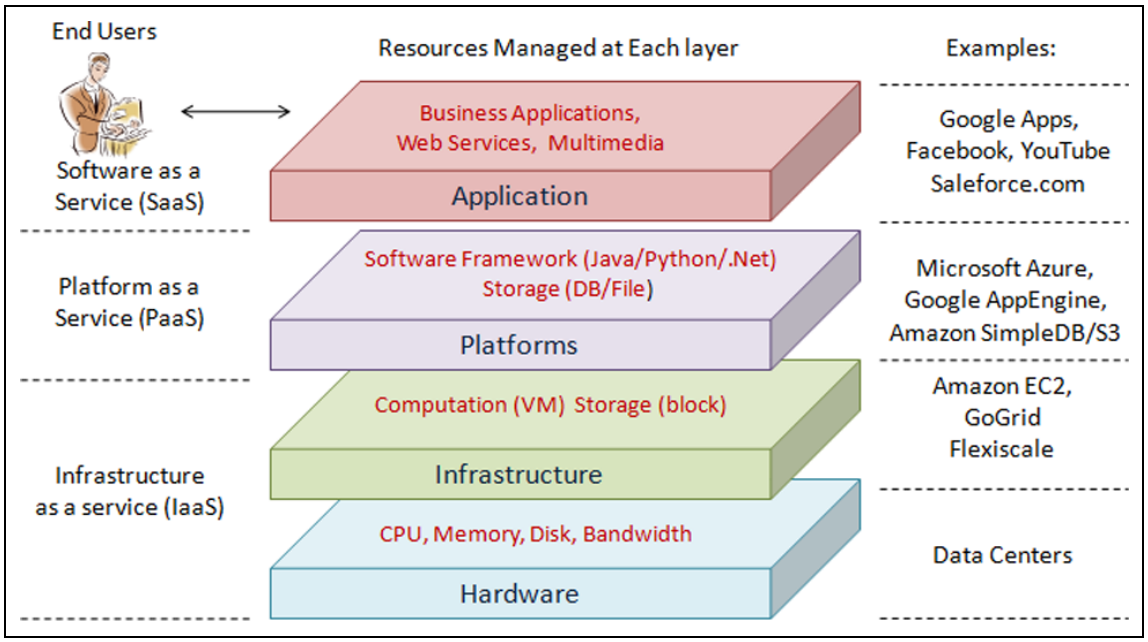

Figure 2.11 Architecture du nuage informatique
Tirée de Zhang, Cheng et Boutaba (2010)

Logiciel en tant que service : Ce modèle connu par le terme anglais SaaS (Software as a Service) fait référence à fournir des offres logicielles complètes sur le nuage. Le consommateur peut accéder souvent à travers un navigateur aux applications logicielles du fournisseur qui s'exécutent sur l'infrastructure du nuage à la demande via Internet. Il ne gère ni contrôle l'infrastructure du nuage sous-jacente, y compris le réseau, les serveurs, les systèmes d'exploitation et le stockage. Cependant, le prestataire est le responsable qui s'occupe de la maintenance et de la mise à jour de l'infrastructure, la plate-forme et le logiciel. Salesforce.com (Salesforce, 2013) est le pionnier dans ce domaine avec son outil de gestion de la relation client (CRM). Rackspace (Rackspace, 2013) et SAP Business ByDesign (SAP, 2013) sont d'autres fournisseurs des solutions en mode SaaS.

Plateforme en tant que service : Connu par son appellation anglaise PaaS (Platform as a Service), ce modèle consiste à offrir un environnement d'hébergement et de développement sur le nuage permettant de créer des applications à l'aide des langages de programmation et des outils pris en charge par le fournisseur. Le PaaS permet aussi de fournir aux utilisateurs plus de flexibilité tout en réduisant la complexité de développement et en évitant la perte de

temps de déploiement de leurs applications. L'utilisateur ne gère ni contrôle l'infrastructure du nuage sous-jacente (réseau, serveurs, systèmes d'exploitation ou stockage), mais il peut contrôler les applications déployées et, éventuellement, les configurations d'environnement d'hébergement d'applications. Parmi les solutions PaaS, nous citons Google App Engine (Google, 2012), Microsoft Windows Azure (Microsoft, 2012) et Force.com (Salesforce, 2013).

Infrastructure en tant que service : Ce modèle, connu par le terme anglais IaaS (Infrastructure as a Service), consiste à fournir à la demande de ressources matérielles telles que des unités de puissance de calcul, de traitement et des capacités de stockage, souvent en termes des machines virtuelles. Le consommateur peut déployer et contrôler les systèmes d'exploitation, le stockage, les applications, et éventuellement la possibilité de configurer certains composants réseau (par exemple, pare-feu). Des exemples des principaux fournisseurs d'Infrastructure en tant que service sont Amazon EC2 (Amazon, 2012a), GoGrid (GoGrid, 2013) et Flexiscale (Flexiscale, 2013).

2.3.5 Modèles de déploiement

Il y a plusieurs facteurs à prendre en considération lors de la migration d'applications d'entreprise vers un environnement de nuage. Par exemple, certains fournisseurs de services sont souvent intéressés à baisser les coûts de fonctionnement, tandis que les consommateurs peuvent préférer ceux qui offrent un niveau élevé de sécurité, de fiabilité et de performance à des prix variés. Par conséquent, il existe différents types de nuage, chacun ayant ses propres avantages et inconvénients (Dillon, Wu et Chang, 2010). La figure 2.12 montre ces différents types.

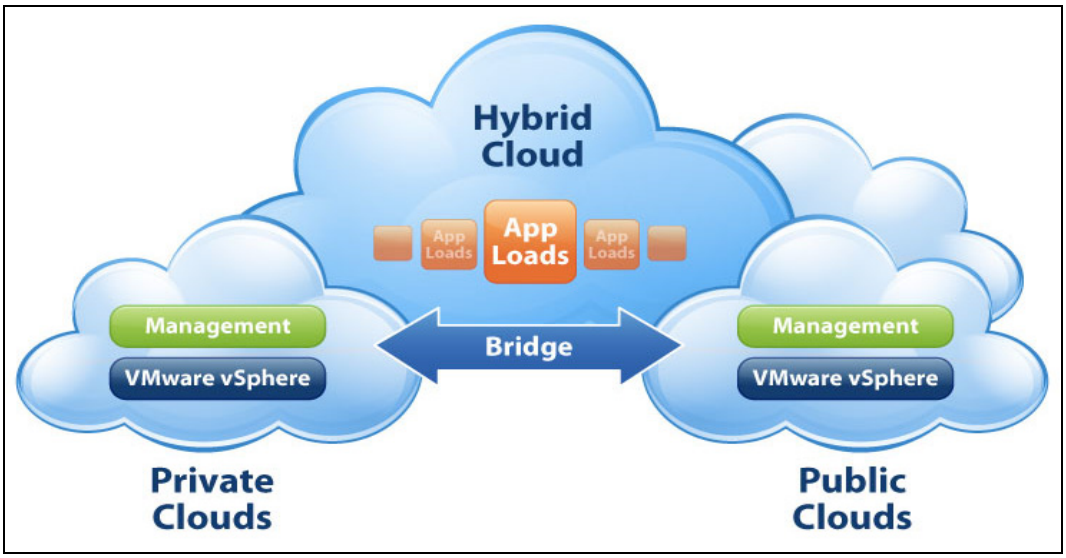

Figure 2.12 Architecture du nuage hybride
Tirée de VMware (2012a)

Nuage public : L'infrastructure du nuage est mise à la disposition du public en général ou d'un grand groupe d'entreprises via le réseau. Le fournisseur gère l'infrastructure et il peut partager les ressources entre plusieurs entités clientes. Ce type de nuage offre plusieurs avantages aux clients, y compris la réduction d'investissement initial avec une évolutivité pratiquement infinie, l'optimisation des coûts d'exploitation et le transfert des risques technologiques vers le fournisseur d'infrastructure. Toutefois, le nuage public n'offre pas une disponibilité permanente ou un contrôle précis sur les données, le réseau et les paramètres de sécurité, ce qui laisse la majorité des entreprises préférant le nuage privé.

Nuage privé : L'infrastructure est conçue pour une utilisation exclusive par une seule organisation. Le nuage privé peut être hébergé et géré par l'organisation ou par un tiers. Dans le cas d'un fournisseur tiers, l'organisation peut utiliser des réseaux sécurisés de types VPN pour accéder à l'infrastructure. Ce modèle de nuage offre le plus haut degré de contrôle sur les performances, la fiabilité et la sécurité. Cependant, le directeur des technologies d'Amazon (Amazon, 2012b) Werner Vogels reproche au nuage privé son manque d'élasticité et ses coûts cachés (Bouzereau, 2010). En effet, l'infrastructure du nuage privé est souvent similaire aux serveurs propriétaires traditionnels et ne fournit pas les bienfaits du nuage recensés tels que la réduction des coûts des investissements initiaux.

Nuage communautaire : Il consiste à partager l'infrastructure informatique entre des organismes d'une même communauté qui ont des intérêts communs. Par exemple, tous les organismes gouvernementaux dans la province de Québec peuvent partager une même infrastructure informatique sur le nuage pour gérer les données relatives aux résidants du Québec. Également, la solution du nuage communautaire peut être adaptée pour résoudre la complexité de gestion des processus de réservations utilisés non seulement pour le trafic aérien, mais aussi pour les hôtels et les locations de voitures (Claranet, 2013). Ce nuage peut être géré par un tiers ou par les organisations elles-mêmes par l'intermédiaire d'Internet ou de réseaux privés. En outre, les coûts peuvent être répartis sur un nombre moindre d'utilisateurs ce qui permet de réaliser des économies plus importantes que le nuage privé, tout en offrant certaines de ses caractéristiques de sécurité.

Nuage hybride : Le nuage hybride est une combinaison de deux ou plusieurs modèles de nuage (privé, communautaire ou public) qui tente de remédier aux limitations de chaque approche et permettre la portabilité des données et des applications entre les nuages. Ces nuages sont liés par l'entremise des technologies normalisées ou propriétaires permettant la portabilité des données et des applications, mais demeurent des entités indépendantes. Dans un nuage hybride, une partie de l'infrastructure de service peut tourner dans un nuage privé tandis que l'autre partie s'exécute dans un nuage public. Le nuage hybride offre plus de flexibilité que les nuages publics et privés. Spécifiquement, il fournit des meilleurs contrôles et sécurités sur les données d'application par rapport au nuage public, tout en offrant la flexibilité et la mise à l'échelle des services par l'augmentation ou la diminution des capacités de calcul selon les modifications des besoins. Cependant, la conception d'un nuage hybride nécessite de déterminer soigneusement la meilleure répartition entre les composantes de nuage public et privé.

2.4 Travaux connexes

L'utilisation et le développement de services de nuage qui n'ont pas été standardisés engendrent de nombreuses difficultés de migration, de déploiement et de portabilité des applications dans le nuage. Il existe différentes initiatives de standardisation visant à normaliser cette situation, mais aucun consensus n'a été atteint (Miranda et al., 2012). Par conséquent, plusieurs solutions ont été proposées en se basant sur l'utilisation des couches intermédiaires qui permet d'isoler les applications de la variabilité de certains services offerts par les fournisseurs de nuages. Lorsque ces outils sont propriétaires, ils peuvent créer des applications qui ne fonctionnent que sur une infrastructure spécifique. Ces approches constituent donc une solution partielle de ce problème, car elles contribuent au risque de déplacer l'effet de l'enfermement propriétaire du fournisseur vers les outils de déploiement.

Dans la littérature, il y a plusieurs travaux existants qui tentent d'aborder les problèmes de la migration et de déploiement d'applications dans le nuage en adoptant différentes approches.

Tran et al. (2011) ont proposé une approche de migration d'une application .Net d'architecture n-tiers vers la plateforme Windows Azure (Microsoft, 2012). Dans ce travail, les auteurs ont proposé une classification des différentes tâches de migration sous forme de catégories en indiquant les facteurs importants qui influent sur le coût de la migration. Cette proposition est une bonne initiative pour simplifier aux développeurs les procédures de migration vers le nuage. Cependant, elle offre seulement des techniques de migration d'une application particulière pour être déployée dans un environnement spécifique, Windows Azure. De plus, elle n'assure aucune réduction de la complexité de déploiement et elle exige une maîtrise de l'application à migrer et beaucoup d'effort pour la reconfigurer dans l'environnement cible.

Le projet mOSAIC (Di Martino et al., 2011), une initiative de référence financée par la commission européenne dans le cadre du programme FP7-ICT, consiste à proposer une solution permettant d'offrir aux utilisateurs du nuage la liberté de choix au niveau des

langages de programmation ainsi qu'au niveau des ressources informatiques. Il traite principalement la normalisation des mécanismes de développement et de déploiement des applications sur le nuage. Ce travail se caractérise par sa robustesse et se base sur une API et une plate-forme libre pour le développement des applications dans le nuage. Cependant, cette approche vise à supporter seulement les nouvelles applications compatibles avec les environnements du nuage et elle n'offre pas des services de migration des applications traditionnelles existantes. De plus, elle n'intègre ni des APIs ni des outils pour la gestion et le déploiement automatique d'applications sur le nuage pour esquiver les tâches d'administration récurrentes.

La solution Elastic-R (Chine, 2010) est une plateforme en tant que service qui permet aux enseignants et aux étudiants d'allouer des machines virtuelles sur le Cloud via un navigateur Web standard pour utiliser les environnements de statistiques et mathématiques les plus couramment utilisés tels que R et Scilab. Elastic-R offre aussi des services de collaboration et de partage des ressources, ainsi que des techniques d'enseignement interactives. Ses services distribués permettent aux éducateurs de créer sur le Cloud leurs propres outils e-learning. Toutefois, cette solution est compatible seulement avec EC2 d'Amazon (2012a) et elle se limite uniquement à offrir des services de calcul mathématique et statistique basés sur les environnements R (Ross et Robert, 2013) et Scilab (Campbell, Chancelier et Nikoukhah, 2006). De plus, elle n'est pas extensible pour supporter d'autres services et n'offre pas la portabilité des services créés puisqu'ils ne peuvent fonctionner qu'avec Amazon web services.

Tian, Su et Lu (2010) ont développé une plateforme PaaS pour gérer les ressources informatiques des laboratoires d'universités dans un nuage privé. L'objectif de ce travail est de résoudre le problème d'accès limité aux logiciels. Grâce à la virtualisation et les mécanismes de partage de ressources, cette solution a permis de répondre aux besoins des étudiants pour accéder aux applications à distance, mais elle est implémentée d'une manière spécifique et sur mesure pour un environnement particulier. De plus, son modèle

architectural n’est pas générique et ne peut pas être mis en œuvre pour d’autres infrastructures informatiques.

Dans (Salih et Zang, 2012), une enquête est présentée dont l’objectif principal est de comprendre la manière d’utiliser efficacement les plates-formes PaaS dans un environnement de nuage. Une comparaison détaillée de différentes plateformes du Cloud a été exposée en se basant sur plusieurs aspects tels que l'architecture, les caractéristiques, les applications supportées, etc. Salih et Zang (2012) ont montré à travers ce travail l’importance de l’utilisation des plateformes open source. En effet, ils ont expliqué qu’un PaaS a le potentiel de démocratiser le développement web en permettant à tous ceux qui peuvent utiliser un fureteur à programmer, tester et évoluer aisément leurs applications Web. En outre, ils ont prouvé qu’une solution PaaS open source s’adapte avec les standards de l'industrie et permet aux applications d'être déployées à travers de multiples fournisseurs du Cloud. L’enquête encourage et motive les développeurs d’utiliser les PaaS Open source pour bénéficier de leurs caractéristiques de portabilité et de flexibilité. Toutefois, aucune des solutions présentées dans cette enquête ne se base sur une approche d’automatisation de déploiement d’applications sur le Cloud.

D'autres travaux sont orientés vers l'analyse des difficultés existantes et la recherche des solutions pour remédier aux problèmes de l'interopérabilité et de la portabilité des applications dans le nuage. Différentes solutions sont proposées dans ce domaine :

Open Cloud Computing Interface (OCCI, 2012) est un ensemble de spécifications permettant le développement des outils pour effectuer des tâches communes de gestion des systèmes de nuage incluant le déploiement, la mise à l'échelle automatique, le monitorage et la gestion du réseau. L’API proposée par OCCI est appuyée par divers projets de nuage informatique comme Eucalyptus, OpenNebula, et OpenStack (Favoritemedium.com, 2011; Khazret, 2013; OpenStack, 2013). Cette approche de standardisation vise à assurer à la fois l’interopérabilité qui permet aux différents fournisseurs de nuage de travailler ensemble sans aucune restriction ni réserve, et la portabilité des applications permettant aux clients de basculer facilement

entre les fournisseurs en fonction des objectifs commerciaux (par exemple, coût) afin de favoriser la concurrence. Il convient de mentionner qu'OCCI a commencé comme une API pour la gestion des infrastructures Cloud. Comme il est indiqué dans son site officiel (OCCI, 2012), l'Open Cloud Computing Interface servira aussi d'autres modèles de nuage en plus de l'IaaS, tels que PaaS et SaaS, cependant Mike (2012) a affirmé que la norme OCCI n'est encore prête pour le modèle PaaS.

Miranda et al. (2012) ont montré qu'il n'y a pas suffisamment d'efforts qui ont été faits dans la normalisation des méthodologies de développement pour assurer la portabilité et la compatibilité d'applications dans les environnements de nuage. Par ailleurs, les auteurs ont exploré une solution alternative basée sur l'utilisation des techniques d'adaptation logicielles (SA) dont différentes lignes directrices ont été présentées comme la base d'une approche qui favorise l'interopérabilité et la migration des applications vers le nuage, évitant ainsi l'enfermement propriétaire « vendor lock-in ». Cette approche peut être appliquée pour identifier et résoudre les sources de discordance du code de déploiement d'applications dans les environnements de nuage. Malgré l'importance de l'adaptation des composants logiciels dans différents environnements d'exécution, ce travail manque d'être intégré dans une plateforme PaaS pour que les développeurs puissent utiliser son algorithme.

Une autre proposition de développement des applications basée sur l'adaptation logicielle est présentée dans (Guillén et al., 2013). Elle permet aux applications d'être déployées indifféremment sur tout environnement de nuage. La segmentation de l'application en différents modules sert à faciliter le déploiement et la redistribution de l'application dans des environnements hétérogènes. L'interopérabilité entre les composantes interdépendantes déployées dans différents nuages est réalisée en générant automatiquement les services de communication nécessaires. Cette solution assure aux développeurs la portabilité de leurs applications et l'interopérabilité des services associés. Toutefois, elle n'offre pas des services d'orchestration des tâches malgré qu'elle utilise un mode de déploiement où les composants des applications sont distribués dans différents environnements de nuage. Ceci cause une perte de temps spécifiquement au niveau de la configuration et l'établissement de la

communication des services puisqu'ils proviennent de différentes sources et zones géographiques. Ainsi, cette approche complique le déploiement de l'application sur le nuage et perturbe la concentration du développeur au détriment de l'amélioration de son code source.

Dans (Dowell et al., 2011), les auteurs décrivent les défis du nuage informatique, examinant les efforts industriels et proposant un modèle d'interopérabilité entre les environnements de nuage. Parmi ces défis, nous citons la nécessité de réaliser l'intégration des services de nuage à travers les principes de la technologie SOA. Les auteurs proposent un service d'orchestration en tant qu'une couche middleware intermédiaire. En effet, la portabilité est considérée comme un défi particulier de l'interopérabilité. Vue sous cet angle, l'utilisation incohérente du terme interopérabilité pour désigner les deux concepts peut être justifiée. Toutefois, Petcu (2011) a proposé deux définitions en termes d'interopérabilité. La première est l'interopérabilité horizontale qui peut être définie comme la capacité de communiquer l'un avec l'autre entre deux services du même modèle de nuage (IaaS/PaaS/SaaS). La deuxième est l'interopérabilité verticale qui a été définie comme la capacité de déployer un service sur un autre d'un modèle de nuage inférieur tel est le cas d'une application SaaS portée sur différentes plateformes PaaS.

Le facteur commun de tous ces travaux est le désir de créer une couche d'abstraction intermédiaire qui découple les applications et les outils propriétaires imposés par les prestataires de nuage. Dans ce cas, les développeurs créent leurs applications en utilisant cette couche intermédiaire qui est une plateforme indépendante permettant de masquer les APIs propriétaires des vendeurs. Ainsi, la couche intermédiaire empêche les développeurs d'être liés aux langages de programmation, formats de fichiers ou stockage de données spécifiques.

En somme, certains travaux ont proposé une abstraction des particularités des infrastructures spécifiques afin de favoriser la migration des applications d'un nuage à un autre, mais ils n'ont pas abordé les problèmes de portabilité et d'interopérabilité. D'autres ont traité le

problème de l'interopérabilité en proposant des solutions de standardisation des services de nuage, cependant aucun consensus n'a été établi. Les difficultés rencontrées pour adopter ces standards sont la conséquence du manque d'acceptation des fournisseurs comme Google, Microsoft, Amazon et d'autres puisqu'ils veulent offrir des services différenciés pour attirer plus de clients. De plus, la définition de normes au niveau de l'infrastructure, qui n'est pas une tâche facile, ne résout pas le problème de la portabilité des applications vu que chaque plateforme propose sa propre interopérabilité avec ses outils et frameworks. De même, la plupart des propositions ont pour objectif de créer et développer de nouvelles applications dans le nuage, mais aucune de ces propositions n'offre des services de migration d'applications existantes. Également, aucun de ces travaux ne prend en charge l'automatisation de déploiement d'applications dans le nuage en plus qu'il n'y a pas de service d'orchestration offert pour éviter les tâches répétitives et fastidieuses.

Dans ce contexte et étant construit sur des technologies open source, notre modèle proposé OpenICRA est conçu pour assurer la liberté de choix, y compris la liberté de se déplacer d'un environnement traditionnel à un autre de nuage et vice-versa. À cette fin, seulement les langages et outils open source standards sont utilisés et aucune API, technologie ou ressource propriétaire n'est intégrée dans notre modèle. Cela garantit l'interopérabilité entre les environnements de nuage et la portabilité des applications à la fois sur et hors de la plateforme OpenICRA, empêchant ainsi l'enfermement propriétaire sur notre plateforme. OpenICRA comprend un ensemble complet d'outils de ligne de commande qui fournissent un accès complet à l'interface développeur. Ces outils sont faciles à utiliser et aussi scriptable pour les interactions automatisées. Notre solution proposée comprend également une riche interface Web console pour la gestion et l'orchestration en parallèle et en temps réel des nœuds d'un cluster, permettant ainsi de réduire la complexité des tâches de développement et de gestion des applications.

En plus, notre modèle générique est capable de migrer et de déployer les applications existantes sur un environnement Cloud et il offre des services d'automatisation des tâches récurrentes en déclenchant en temps réel des actions dans un cluster de nœuds. Pour

bénéficier pleinement de l'élasticité du nuage, OpenICRA fournit automatiquement une mise à l'échelle horizontale des ressources en fonction de l'augmentation ou de la diminution de la charge d'application éliminant ainsi le besoin des opérations manuelles d'ajout des nouvelles instances des machines virtuelles.

2.5 Conclusion

Le *Cloud Computing* se positionne actuellement en tête de liste des nouvelles technologies. Il se caractérise par son extensibilité et élasticité et son exploitabilité par un grand nombre d'utilisateurs dans le monde entier. Les ressources informatiques sont disponibles à la demande, en libre-service et facturées en fonction de l'utilisation. Ce concept se base principalement sur le *Grid Computing*, l'*utility Computing* et évidemment, sur la virtualisation que nous avons détaillée dans ce chapitre. Ainsi, comme toute innovation technologique qui se respecte, le nuage informatique non seulement fait économiser de l'argent, mais aussi permet de réduire le temps du déploiement des applications évolutives en utilisant un processus simplifié d'approvisionnement de ressources informatiques et de mise à l'échelle.

CHAPITRE 3

DÉPLOIEMENT AUTOMATISÉ DES APPLICATIONS DANS LE NUAGE

3.1 Introduction

À la lumière des problématiques et objectifs décrits dans les sections 1.2 et 1.3, nous avons proposé un modèle générique de déploiement automatisé des applications dans le nuage. Ce modèle permet à la fois de réduire les efforts de programmation, le temps et les coûts nécessaires à la mise en œuvre des applications à l'aide des fonctions de gestion et d'automatisation. Ceci permet d'accélérer le processus de déploiement et d'assurer la flexibilité et l'évolutivité des applications déployées avec une haute disponibilité.

Dans ce qui suit, nous allons présenter les différentes méthodes suivies pour atteindre les objectifs du mémoire de recherche ainsi que les détails du processus de déploiement automatique des applications dans le nuage.

3.2 Enfermement propriétaire et portabilité

Dans le but d'assurer la liberté de choix, y compris la liberté de déployer des applications développées sur d'autres PaaS, notre modèle proposé doit être conçu et construit en utilisant des piles logicielles et technologiques libres « open sources ». À cette fin, seuls les frameworks et les technologies libres seront utilisés et aucune API, technologie, ou ressource propriétaire ne sera intégrée dans la construction de notre modèle de déploiement automatisé.

Sur demande du développeur, le modèle PaaS proposé doit fournir un niveau beaucoup plus élevé de contrôle, car il aura l'accès à l'infrastructure sous-jacente. Sinon, il cache la complexité de la configuration des ressources en utilisant une couche d'abstraction et de meilleures pratiques permettant de satisfaire les besoins des utilisateurs ayant des exigences moins avancées. En outre, l'intégration des outils d'approvisionnement des ressources de calcul tels que Boxgrinder et VMBuilder avec le gestionnaire de nuage (Cloud Manager, voir

section 4.2.4) permet au modèle proposé de supporter les différents nuages publics (GSN, Amazon EC2, Windows Azure, Rackspace...) et les nuages privés (OpenStack, VMWare vCenter, Citrix XenServer...). En isolant complètement l'application de l'environnement du nuage sous-jacent et en cachant les APIs et la complexité de configuration de l'environnement d'exécution, l'application peut être facilement déplacée d'un nuage à un autre sans aucune restriction et à n'importe quel moment. Cela nous permet d'atteindre notre objectif de minimisation de l'enfermement propriétaire.

La plateforme proposée comprend des langages de programmation et des frameworks libres non modifiés. Cela signifie que les applications développées sur cette plateforme peuvent être facilement déplacées vers d'autres environnements supportant les mêmes langages de programmation. Par exemple, les applications Ruby ou JBoss fonctionnant sur notre plateforme peuvent être déplacées vers des implémentations autonomes de Ruby ou JBoss dans les centres de données. Le développeur peut déployer son application dans le nuage sans apporter des modifications du code source quelle que soit la pile logicielle utilisée (Java/Spring, Java EE, Ruby on Rails, PHP...), la base de données (relationnelles comme MySQL ou non relationnelles telles que MongoDB ou Apache Cassandra), ou tout autre composant que l'application a utilisé.

Aussi, le développeur peut utiliser ses propres langages de programmation et les différents types des bases de données en tirant parti de la puissante de l'extensibilité et de la possibilité de personnalisation des templates si le support offert par la plateforme des langages préconfigurés ne lui convient pas. En utilisant le template personnalisable de déploiement automatisé CTAD (Customizable Template for Automatic Deployment) et l'intégration du gestionnaire de configuration avec l'outil de gestion de versions Git, le développeur peut déployer à peu près n'importe quel programme dans le nuage. Ainsi, il peut déployer n'importe quel binaire qui s'exécute sur une plateforme Linux ou même Windows. Cela nous permet d'atteindre notre objectif de déploiement des applications dans le nuage sans aucun changement du code et avec garantit de la portabilité des applications à la fois sur et en

dehors de notre plateforme, empêchant ainsi l'enfermement propriétaire sur notre modèle générique de déploiement automatisé des applications dans le nuage.

3.3 Approche de redondance

Les deux modes "avec état" (Stateful) et "sans état" (stateless) permettent de décrire si une machine ou un service est conçu pour noter et retenir un ou plusieurs des événements précédents dans une séquence d'interactions donnée avec un utilisateur, un programme, ou tout autre élément extérieur. De surcroît, "avec état" signifie que l'ordinateur ou le programme conserve une trace de l'état de l'interaction, en définissant des valeurs dans un champ de stockage conçu à cet effet. Alors que le terme "sans état" signifie qu'il n'existe aucune trace des interactions passées et que chaque demande doit être manipulée ou traitée en se basant seulement sur l'information qui vient avec cette demande.

C'est dans ce contexte que notre modèle proposé doit être conçu avec une approche de redondance à l'esprit en utilisant des composants sans état (stateless). La redondance sert généralement à deux objectifs qui sont la haute disponibilité et la mise à l'échelle. Les principaux composants du système peuvent certainement tous être séparés dans des hôtes distincts et multipliés pour la redondance.

Ainsi, chaque élément architectural doit être mis en place de manière redondante. Les nœuds de données sont par défaut mis à l'échelle et peuvent fonctionner dans une architecture redondante. Les applications de contrôleur et de gestionnaire de nuage doivent être sans état (stateless) et nécessitent une mise en place derrière un simple équilibreur de charge (load balancer). Le service de messagerie est également sans état (stateless), et Puppet/MCollective doit être configurée pour utiliser plusieurs points de terminaison ActiveMQ. Des instances multiples de MongoDB doivent être combinées dans un cluster de serveurs (jeu de réplicas) qui met en œuvre la réplication maître-esclave et le basculement automatisé pour la haute disponibilité et la tolérance aux pannes. La figure 3.1 présente l'approche de redondance de notre plateforme proposée.

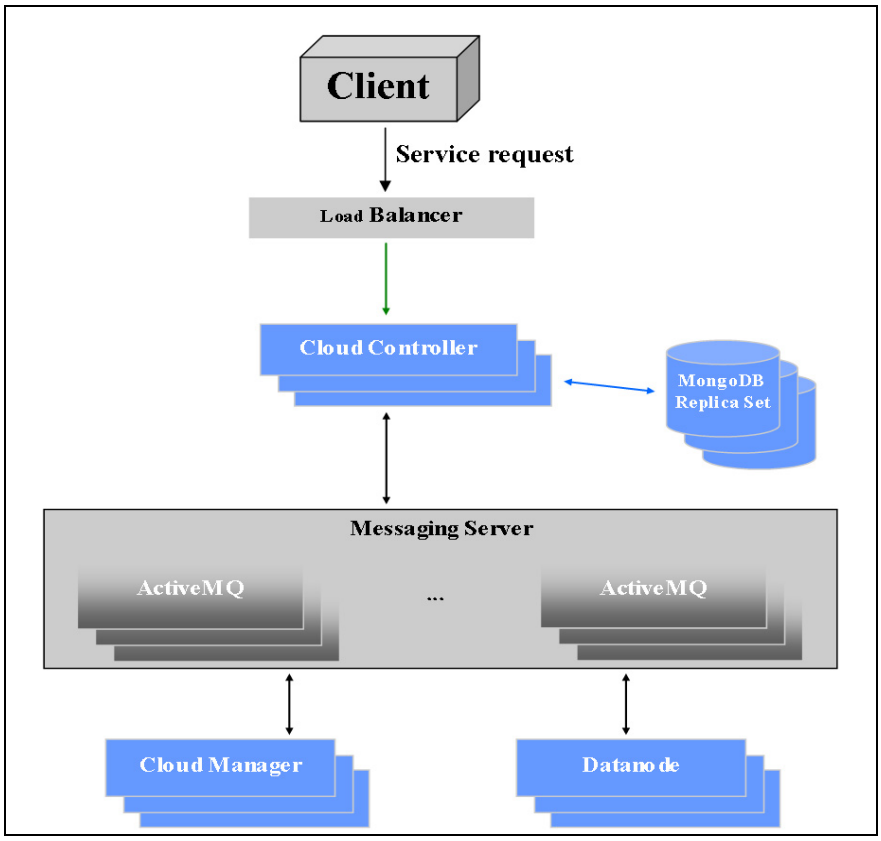

Figure 3.1 Architecture proposée avec une redondance des principaux composants

3.4 Mise à l'échelle automatique des applications

La mise à l'échelle horizontale permet à une application de réagir à l'évolution du trafic et d'allouer automatiquement les ressources nécessaires en fonction de la demande. Ceci est accompli en utilisant l'équilibreur de charge (load balancer).

Au moment de la création ou de déploiement, le développeur doit spécifier si l'application est mise à l'échelle (évolutive) ou non. Si l'application déployée ou créée n'est pas mise à l'échelle, elle occupe uniquement un seul conteneur dans un nœud de données et tout le trafic sera envoyé vers ce conteneur. Toutefois, lorsque le développeur crée une application avec mise à l'échelle, elle approvisionne deux conteneurs, un pour l'équilibreur de charge, et l'autre pour l'application réelle. Si le développeur ajoute d'autres templates comme MongoDB ou MySQL à son application, ils seront installés dans leurs propres conteneurs.

Comme le montre la figure 3.2, l'équilibreur de charge hébergé dans un conteneur dédié se positionne entre le conteneur de l'application et le réseau Internet et il achemine le trafic Web vers le template de l'application du développeur.

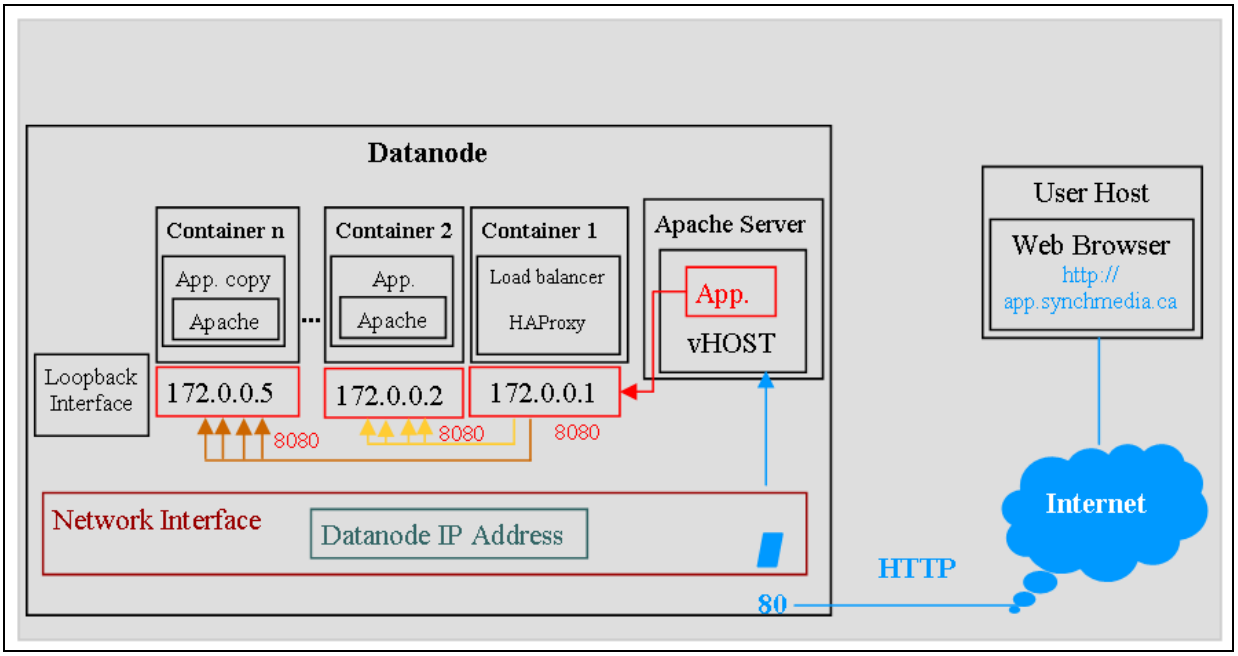

Figure 3.2 Mise à l'échelle automatique : Communication TCP inter-conteneur

Lorsque le trafic augmente, l'équilibreur de charge avertit le contrôleur de nuage de notre plateforme PaaS qu'il a besoin des ressources informatiques supplémentaires. Le contrôleur de nuage vérifie s'il y a des ressources disponibles dans le système de la plateforme PaaS, puis il envoie une requête au gestionnaire de nuage pour créer au développeur une autre copie du son application dans un nouveau conteneur (voir la conception du PaaS proposé à la section 4.2.2). Dans le cas où il n'a pas trouvé de ressources libres, le contrôleur envoie quand même une requête au gestionnaire de nuage pour lui demander d'approvisionner un ou plusieurs nœuds de données. Après, le gestionnaire doit créer un nouveau conteneur pour la copie de l'application. Afin de synchroniser les mises à jour faites par le développeur avec toutes les copies de l'application, le code source dans le dépôt de l'outil de gestion de versions décentralisé Git sera copié dans chaque nouveau conteneur. Lorsque la nouvelle copie de l'application démarre en invoquant les hameçons « hooks », l'équilibreur de charge

commence le routage des requêtes web vers cette nouvelle instance. Si le développeur pousse une modification du code à son application via l'outil Git, toutes ses instances en cours d'exécution obtiendront cette mise à jour.

L'algorithme de mise à l'échelle peut être basé sur le temps de réponse moyen, sur l'utilisation de CPU, sur l'utilisation réseau ou celle de la mémoire. Il permet d'augmenter ou diminuer automatiquement la capacité de ressources selon les conditions définies par le développeur. Avec la méthode de mise à l'échelle automatique, le développeur peut s'assurer que le nombre d'instances de ses applications augmente de façon continue durant les pics de demandes pour maintenir la performance, et diminue automatiquement en supprimant les conteneurs supplémentaires durant la baisse des demandes pour minimiser les coûts.

Par exemple, le développeur peut définir une condition pour ajouter de nouvelles instances de ses applications en incrémentant d'une instance vers le cluster d'application quand l'utilisation moyenne de CPU atteint ou dépasse 80 %; de même, il peut définir une condition pour supprimer les instances de ses applications quand l'utilisation moyenne de CPU chute en dessous de 20 %. Le développeur peut aussi définir une condition de mise à l'échelle qui se base sur le temps de réponse moyenne. Ainsi, l'équilibreur de charge ajoute un autre conteneur au cluster de l'application si le temps de réponse moyenne dépasse par exemple le seuil de 0.5 milliseconde. Si le temps de réponse moyenne chute en dessous de ce seuil pendant plusieurs minutes, il supprime ce conteneur. Ce processus se répète et l'équilibreur de charge continue d'ajouter ou supprimer des conteneurs en fonction des besoins du trafic de l'application.

Il est important de noter qu'il y a d'autres approches de mise à l'échelle automatique qui se basent sur l'utilisation de la mémoire et le nombre de requêtes simultanées pouvant être utilisées par l'équilibreur de charge. Aussi, les seuils d'ajout et de suppression des conteneurs sont configurables et peuvent être ajustés en fonction du besoin du développeur.

3.5 Gestion du stockage dans le nuage (IaaS)

Dans un environnement du nuage informatique tel que le réseau GreenStar Network (GSN) (Synchromedia, 2010), il est nécessaire d'avoir une haute vitesse de calcul et un système de stockage évolutif et efficace. Le système de fichier distribué Hadoop (HDFS) choisi pour gérer le stockage dans le nuage de GSN (voir section 4.3.3) a démontré plusieurs caractéristiques désirables. Avec l'outil de gestion de la distribution des données en temps réel et les techniques d'approvisionnement et d'optimisation des ressources virtualisées, GSN offre un environnement hautement efficace pour les applications à haute performance et E/S intensives. Par conséquent, une intégration du HDFS avec GSN offrirait un excellent environnement pour les machines virtuelles et les applications où l'évolutivité est un problème à la fois en termes de vitesse de calcul et de stockage.

3.5.1 Architecture actuelle du système de stockage de GSN

Le réseau GreenStar utilise un système de stockage partagé côté serveur et le système SSHFS (Secure Shell File System) côté client. Au niveau serveur, le réseau GSN utilise un bloc de stockage unique pour stocker les images des machines virtuelles (images en cours d'exécution et des templates de VMs). La figure 3.3 montre le système de stockage actuel du réseau GreenStar.

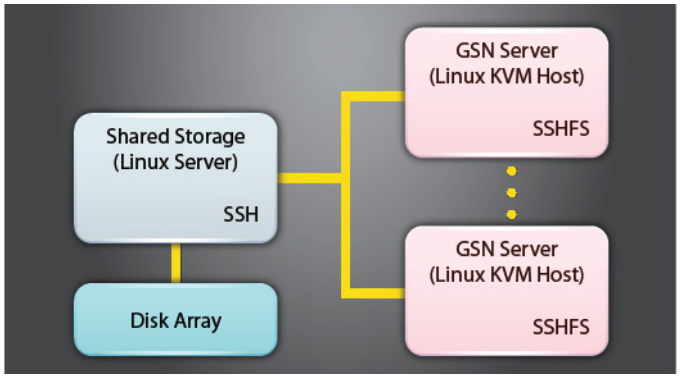

Figure 3.3 Système de stockage actuel de GSN
Tirée de GSN Middleware Team (2011)

Selon Small Tree (2013), un système de stockage partagé se réfère généralement à un disque de stockage partagé à travers le réseau permettant à quiconque sur le même réseau d'accéder aux fichiers du serveur. Ceci signifie que l'architecture du système partagé ne permet pas la mise à l'échelle horizontale en additionnant de nouveaux serveurs de stockage. Par contre, ce système permet seulement la mise à l'échelle verticale dans un seul serveur. De plus, il sauvegarde tous les fichiers de l'infrastructure du réseau entier ce qui limite le nombre de fichiers à stocker. Aussi, un système de stockage partagé ne peut pas entreposer la totalité ou une partie d'un fichier volumineux qui dépasse la capacité du disque partagé.

Le stockage n'était pas une priorité du projet GSN lors de son lancement. C'est pour cette raison que GSN possède certaines limitations au niveau de son système actuel. En effet, il n'est pas évolutif et ne répond pas aux besoins d'un environnement basé sur le concept de nuage informatique. En raison d'absence des solutions dynamiques pour la gestion du système de stockage, le contrôleur OpenGSN peut rencontrer des goulots d'étranglement dans le cas du manque d'espace pour héberger une image volumineuse d'une machine virtuelle ou du traitement massif des données. Ainsi, dans une telle situation le contrôleur ne peut pas ajouter de nouvelles instances jusqu'à ce que le système de stockage soit mis à l'échelle verticalement.

3.5.2 Architecture de HDFS

Hadoop Distributed File System (HDFS) est un système de fichiers distribué, évolutif et conçu pour stocker de manière fiable des fichiers très volumineux sur différentes machines dans un large cluster de nœuds. Il est inspiré par les publications de GoogleFS (Ghemawat, Gobioff et Leung, 2003).

La figure 3.4 présente l'architecture de HDFS qui se compose des éléments suivants:

Namenode : Le nœud de nom « namenode » stocke les informations des métadonnées de la grappe (cluster). Il contient les métadonnées ainsi que l'énumération des blocs de HDFS et la liste des nœuds de données du cluster.

Datanode : Les nœuds de données « datanodes » sont les serveurs de stockage où les blocs de HDFS sont stockés. Ils sont connectés au nœud de nom et envoient périodiquement des signaux de pulsation pour l'informer de leur état.

Namenode secondaire : Le nom de nœud secondaire contient une instance du nœud de nom primaire et il est utilisé en cas de défaillance du nœud de nom.

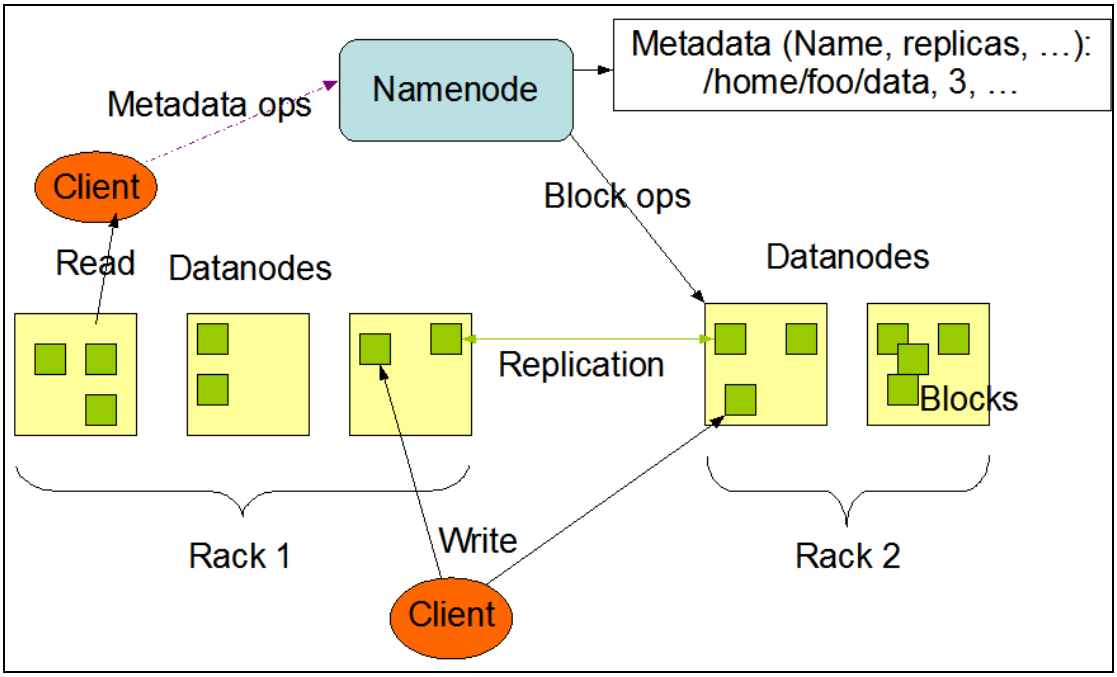

Figure 3.4 Architecture de HDFS
Tirée de Borthakur (2008)

HDFS est conçu pour être déployé sur un matériel moins coûteux et il fournit un accès haut débit aux données applicatives et est adapté aux applications qui utilisent de gros fichiers de données. Il permet d'obtenir une utilisation efficace de l'espace de stockage à travers la compression et la distribution de données stockées dans son cluster de nœuds. Son architecture tolérante aux pannes fournit également un accès rapide aux grandes quantités de

données réparties dans plusieurs serveurs et elle offre une évolutivité élevée (voir section 4.3.3).

Son intégration avec GSN fournira donc une infrastructure idéale pour les applications qui ont des exigences du grand espace de stockage et une très haute vitesse de calcul. Des exemples de telles applications sont les hyperviseurs, les applications d'analyse des fichiers journaux et les applications de haute disponibilité. Ainsi, HDFS peut être un meilleur choix pour une utilisation avec d'autres applications sur un système de fichiers ordinaire afin d'avoir un stockage distribué et tolérant aux pannes.

3.5.3 Système de stockage proposé : Intégration du HDFS avec GSN

Le système de stockage proposé consiste à intégrer le système de fichiers distribué HDFS avec le réseau GSN pour résoudre le problème du rapport entre la croissance du volume des images des VMs avec la performance des nœuds de calcul et du gestionnaire du nuage. Dans notre architecture proposée, nous configurons le cluster HDFS et mettons en œuvre l'extension fuse-dfs du module FUSE, système de fichier en espace utilisateur, sur le nœud de calcul. Ce module fuse-dfs fournit une interface pour le transfert de données entre les nœuds de calcul et les nœuds de données de HDFS. En outre, les machines virtuelles créées par le gestionnaire du nuage peuvent accéder directement à leurs disques durs virtuels stockés dans le cluster HDFS en utilisant le module fuse-dfs.

N'étant pas un véritable système de fichiers compatible avec la norme POSIX, HDFS ne peut pas être monté directement par le système d'exploitation et l'accès aux fichiers de HDFS se fait via ses commandes Shell. Cependant, le module FUSE pour HDFS traite cette limitation en exposant ce dernier en tant qu'un point de montage sur le nœud de calcul. Cela signifie que nous pouvons monter les fichiers à partir de HDFS vers n'importe quel service POSIX. Également, nous pouvons utiliser à distance un client visuel, comme WinSCP, pour explorer les fichiers HDFS. En effet, en utilisant fuse-dfs, qui est l'une des applications basées sur le

système de fichier Fuse, nous pouvons monter HDFS comme s'il s'agissait d'un système de fichiers Linux traditionnel.

Afin d'intégrer HDFS avec GSN, une interface commune pour l'échange de données est nécessaire. Cette interface permettra au réseau GSN de récupérer et de stocker les données dans le cluster de HDFS. La solution consiste donc à utiliser le module fuse-dfs comme un canal de communication et qui relie la machine namenode de HDFS pour monter en entier le système de fichiers distribué. Ainsi, HDFS peut être monté sur les nœuds de calcul en tant que système de fichiers Linux traditionnels tels qu'ext3 ou ext4 et peut être utilisé pour stocker et récupérer des fichiers en utilisant fuse-dfs.

Le module FUSE fournit un «pont» à l'interface noyau des nœuds de calcul. Au lieu d'utiliser l'outil client de HDFS, le terminal standard du nœud de calcul agira en tant que client de HDFS pour accéder à ses données. Aussi, GSN sera en mesure d'interagir directement avec HDFS pour extraire et mettre à jour les données des applications.

Dans notre système de stockage proposé, nous configurons le nœud du nom, les nœuds de données du cluster HDFS et le gestionnaire du nuage sur des serveurs différents. Ceci est fait pour une meilleure performance et optimisation d'utilisation des ressources. Étant donné que le module fuse-dfs doit interagir directement avec le gestionnaire de nuage, ces deux composants doivent fonctionner sur le même système pour éviter les problèmes de performance et de lui permettre d'accéder aux fichiers du système HDFS. En conséquence, le module fuse-dfs est configuré sur le même nœud que le gestionnaire de nuage (voir section 4.2.4) en supposant que JAVA JDK est déjà installé sur l'hôte. Une telle configuration permet une meilleure utilisation des ressources.

L'architecture de la solution de gestion de stockage proposée est illustrée à la figure 3.5. En effet, le Namespace gère les répertoires, les fichiers et les blocs et il prend en charge les opérations du système de fichiers telles que la création, la modification, la suppression et le listage des fichiers et des répertoires. Le Block Storage comporte deux parties qui sont le

Block Management et le Physical Storage (datanodes). Le Block Management maintient l'adhésion des datanodes dans le cluster et il prend en charge les opérations liées aux blocs, comme la création, la suppression, la modification et l'obtention de l'emplacement des blocs. Il prend également en charge la réplication des blocs. Alors que le Physical Storage s'occupe du stockage et de l'accès en lecture et écriture aux blocs.

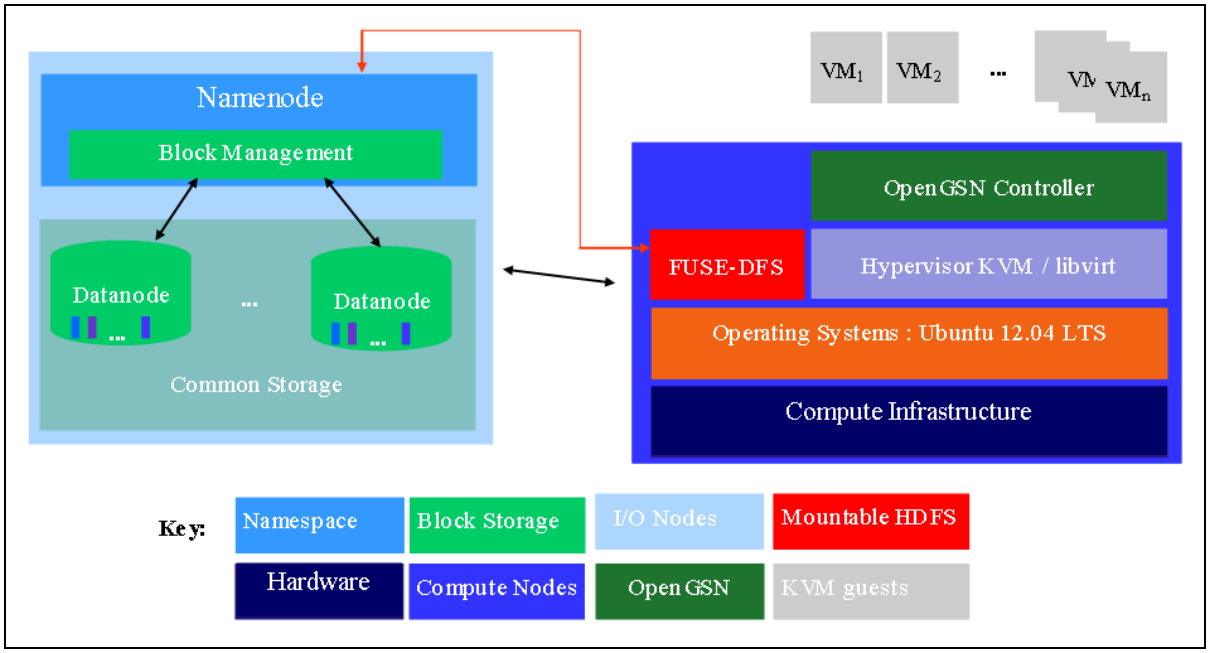

Figure 3.5 Intégration du système de fichiers distribué HDFS avec le réseau GSN

Le module *fuse-dfs* est le composant responsable du montage de HDFS. Il utilise les APIs de la librairie *libhdfs* et FUSE pour apparaitre le système de fichiers HDFS comme un système de fichiers ordinaire sur la machine hôte. Il permet également aux programmes arbitraires d'accéder aux données stockées dans HDFS. Le *Compute node* est le serveur sur lequel le contrôleur OpenGSN et l'hyperviseur KVM s'exécutent.

Le système de stockage proposé peut être facilement mis à l'échelle en ajoutant ou en retirant des nœuds en fonction des besoins. Il n'a besoin d'aucune modification à apporter aux composants existants du réseau GSN. L'ajout ou la suppression d'un nœud ou d'un espace de stockage est transparent pour l'infrastructure du réseau GSN. Le système proposé est donc de

nature élastique. Ainsi, il peut être utilisé comme un entrepôt pour les images volumineuses des machines virtuelles.

3.6 Processus de déploiement automatique des applications dans le nuage

Afin de réduire la complexité et les erreurs de déploiement, d'accélérer la mise en œuvre des applications et de réduire les coûts, nous utilisons des modules et des classes extensibles et personnalisables pour automatiser le déploiement des applications dans le nuage, permettant aux développeurs de décrire leurs applications, leurs services et leurs interdépendances ainsi que la surveillance et la mise à l'échelle de leurs services et leurs ressources de calcul et de stockage. En conséquence, le déploiement et la gestion de la configuration d'une application dans le nuage en utilisant notre modèle PaaS deviennent un processus simple qui se représente en trois étapes principales comme le montre la figure 3.6.

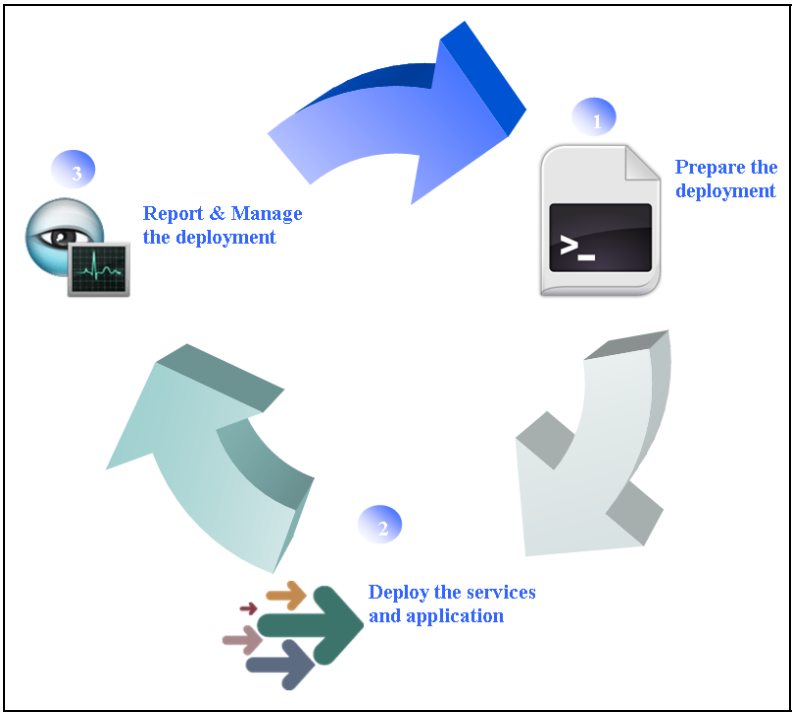

Figure 3.6 Processus de déploiement automatique en trois étapes

3.6.1 Préparation du déploiement de l'application

La première étape du processus de déploiement consiste à :

- décrire l'approvisionnement des ressources informatiques nécessaires pour le déploiement de l'application, dans un environnement de nuage (public ou privé);
- préparer les fichiers binaires ou le code source requis pour les services de l'application;
- décrire le cycle de vie de l'application et de ses services dans des modules et des classes;
- décrire les règles de monitorage et de la mise à l'échelle de l'application et de ses services.

En ce qui concerne la description de l'approvisionnement des ressources et du cycle de vie de l'application et de ses services ainsi que les règles de la surveillance et de la mise à l'échelle, le développeur peut réduire le temps du développement des modules et des classes en utilisant des modules existants et réutilisables et les personnaliser selon ses besoins. Alternativement, s'il a certaines connaissances préalables, il peut construire un module typique en utilisant le langage de configuration déclaratif du gestionnaire de configuration de notre modèle PaaS. Ces modules permettent à définir l'état de déploiement désiré de l'application et de ses services d'une façon automatique. En outre, le développeur peut réutiliser ces modules pour déployer des applications distribuées dans un cluster de nœuds physiques ou virtuels ainsi que dans des environnements traditionnels ou de nuages.

3.6.2 Déploiement automatique des services et des applications

La deuxième étape est celle de l'automatisation du déploiement des applications dans le nuage. Elle consiste à déclencher automatiquement le déploiement par une seule action d'un clic sur un bouton ou en tapant une simple commande sur l'outil en ligne de commande « synchro ». Parmi les commandes utilisées, nous citons « *synchro run X -I rgadhgadhi@synchromedia.ca -p qwerty* », « *synchro app create -s icra ctad mysql-5.1 phpmyadmin-3.4 -I rgadhgadhi@synchromedia.ca -p qwerty* », etc. La première commande est utilisée pour déclencher le déploiement automatique d'une application en lançant le script

X créé par le développeur, alors que la deuxième commande consiste à créer un template évolutif et personnalisable de type *CTAD* avec les applications MySQL et PHPMyAmin pour la création de la base de données. Le type de template *CTAD* permet au développeur d'installer toute application compatible avec le système Linux. Les options « -I » et « -p » sont utilisées pour l'authentification du développeur par le contrôleur de nuage.

En bref, cette étape consiste à :

- approvisionner automatiquement les ressources de calcul et de stockage du nuage choisi en utilisant le gestionnaire de nuage (Cloud Manager);
- auto-installer des agents Puppet dans les conteneurs ou les machines virtuelles approvisionnées;
- déclencher le déploiement de l'application par le contrôleur de nuage (Cloud Controller) en utilisant l'application MCollective.

Après le déclenchement du déploiement par le contrôleur de nuage, chaque client Puppet (agent) contacte le serveur (master) pour télécharger la dernière configuration et l'appliquer afin d'atteindre l'état désiré et défini par le développeur. Les clients Puppet sont les agents installés par le *Cloud Manager* dans les ressources approvisionnées (conteneurs ou machines virtuelles), alors que le contrôleur de nuage est celui qui orchestre l'ensemble des composants de la plateforme à travers MCollective.

Lorsque les clients commencent à appliquer la nouvelle configuration, ils créent le template qui correspond au type de l'application à déployer et installent ses frameworks et ses services. Par la suite, ils configurent l'application, les règles de monitorage et de mise à l'échelle et ils s'assurent que cette configuration est synchronisée avec celle désirée par le développeur. Une fois terminé, chaque client commence à envoyer un rapport vers le serveur indiquant s'il y a des erreurs à corriger ou le déploiement est réussi.

Même pour le développement de nouvelles applications ou de nouveaux composants d'une application existante, la mise à jour peut se faire d'une façon automatique à travers les

techniques de synchronisation différentielle entre les versions locales « *on-premise* » et celles dans le nuage.

3.6.3 Surveillance et gestion du déploiement

Cette dernière étape consiste à surveiller et gérer le déploiement à l'aide de l'interface web console ou le terminal (CLI) de la plateforme. Ainsi, le développeur peut évaluer l'automatisation et surveiller ses services applicatifs pour la disponibilité et la performance en utilisant l'outil de monitorage par défaut. Il est possible d'intégrer des outils de monitorage efficaces avec l'API libre de Puppet afin de générer des rapports détaillés sur le résultat de déploiement pour garantir la gouvernance, la conformité et le contrôle.

3.7 Conclusion

Dans ce chapitre nous avons présenté un aperçu de l'architecture de notre modèle générique de déploiement automatique des applications dans un environnement de nuage. Le modèle est constitué de trois composants architecturaux clés : le Cloud Manager, le Cloud Controller et les Datanodes. Le Cloud Manager est le responsable de l'approvisionnement des ressources de calcul et de stockage ainsi que l'exécution des tâches demandées par le Cloud Controller. Ce dernier est le cerveau du modèle et il s'occupe de l'automatisation du déploiement des applications, de la mise à l'échelle, de la gestion et du contrôle du système. Les datanodes sont les nœuds qui hébergent les templates des frameworks et les données des applications. Tous ces composants sont interconnectés à travers le serveur de queues ActiveMQ et le client et les serveurs du logiciel d'orchestration MCollective.

Les méthodes de redondance, de mise à l'échelle et l'intégration du système de fichiers distribué avec la couche IaaS permettent d'assurer la haute disponibilité, l'évolutivité et l'extensibilité du modèle et des applications déployées par notre plateforme ainsi que l'optimisation de la gestion de stockage des disques durs virtuels des VMs dans un environnement du nuage. L'utilisation d'un écosystème de déploiement libre et l'intégration

des frameworks et des modules extensibles permettent aussi au modèle d'assurer la portabilité des applications et d'éviter l'enfermement propriétaire dans le nuage.

CHAPITRE 4

ARCHITECTURE ET CONCEPTION DU MODÈLE DE DÉPLOIEMENT

4.1 Introduction

Notre travail de recherche consiste à concevoir et à développer un nouveau modèle générique de déploiement automatisé des applications dans le nuage. Pour atteindre cet objectif, nous nous proposons de nous servir de trois éléments architecturaux essentiels qui sont les suivants :

- un PaaS permettant la migration et le déploiement d'applications dans le nuage ;
- un outil de gestion de configuration pour automatiser le déploiement et éviter la répétition des tâches fastidieuses ;
- un système de fichier distribué pour faciliter le partage et la gestion des données dans le nuage.

Étant donné les énormes variétés des scénarios de déploiement, des plates-formes, des configurations et des applications au sein des environnements de production, il est rare de découvrir un outil complètement personnalisable et qui répond à nos besoins. Une solution alternative pour un usage personnel et commercial est l'utilisation des logiciels libres et gratuits. Ces logiciels à code source ouvert offrent deux avantages clés pour les organisations:

- ils sont ouverts et extensibles;
- ils sont gratuits.

Dans ce qui suit, nous commençons par présenter les hypothèses et la conception du modèle proposé que nous avons baptisé OpenICRA. Dans la section 4.3.1, nous comparons les

différentes solutions libres des différentes plateformes en tant que service (PaaS). Cette comparaison nous aidera à choisir le système le plus approprié pour le déploiement d'applications selon les critères souhaités. La section 4.3.2 expose la comparaison des outils libres d'automatisation et d'orchestration des tâches pour sélectionner la meilleure solution qui va être améliorée et intégrée dans le PaaS choisi. Finalement, nous comparons dans la section 4.3.3 les systèmes de fichier distribués pour choisir la solution de gestion de stockage adéquate.

4.2 Conception du modèle PaaS proposé : OpenICRA

4.2.1 Hypothèses

Cette section expose en détail nos hypothèses de la conception du modèle proposé.

- le système OpenICRA est construit seulement à partir des composants libres tels que HAProxy, Puppet, OpenShift, etc. Ainsi, aucune technologie, API ou ressource propriétaire n'est intégrée dans le système.
- le système utilise les ressources du projet GreenStar Network (GSN) comme une infrastructure en tant que service contrôlée par le logiciel de gestion du nuage OpenGSN. Le système est également compatible avec tout autre environnement de nuage tel qu'Amazon EC2, Rackspace et GoGrid.
- le système utilise un système de fichiers distribué au niveau IaaS pour répondre aux besoins de la croissance des demandes de traitement de données de l'infrastructure sous-jacente.
- le module de télécommunication développé dans OpenICRA interagit avec des fournisseurs VoIP externes tels que Google Voice pour les appels sortants et IPKall pour les appels entrants. Il utilise aussi un serveur PABX local basé sur Asterisk pour assurer les appels SIP. Ces serveurs ont pour but d'interconnecter l'application déployée avec les réseaux VoIP et PSTN.

4.2.2 Architecture globale du modèle proposé

Le modèle proposé OpenICRA est une plateforme en tant que service (PaaS) qui permet d'automatiser et d'orchestrer autant que possible le déploiement des applications dans le nuage. Il intègre une grande variété de services et des technologies avancées pour assurer la mise à l'échelle des applications et une meilleure qualité de service pour l'utilisateur final.

Ce modèle se positionne entre l'application et l'environnement de nuage choisi en s'occupant de l'infrastructure, de la gestion et de la configuration des ressources nécessaires, laissant le développeur concentré seulement avec le développement et l'amélioration de ses applications.

OpenICRA met en œuvre une architecture en couches qui cache les détails de l'implémentation permettant d'avoir un processus de déploiement simple comme le montre la figure 4.1. Cette architecture se compose d'un contrôleur de nuage (Cloud Controller), un gestionnaire de ressources (Cloud Manager) et plusieurs nœuds de données (Datanodes). Chaque composant est une machine Linux configurée avec le module de sécurité SELinux et qui exécute des processus de communications permettant de constituer un cluster de nœuds.

Les gabarits des frameworks de différents langages de programmation sont hébergés dans les datanodes. Ils se caractérisent par leurs extensibilités permettant aux développeurs d'ajouter de nouveaux frameworks afin de supporter le déploiement de tout type d'application. Les modules qui sont hébergés dans le Cloud Controller permettent aux développeurs de migrer leurs applications vers le nuage sans apporter aucune modification dans le code, indépendamment du langage de programmation (PHP, Java/Spring, Python, Ruby on Rails...), des bases de données (relationnelles telles que MySQL, ou NoSQL telles que Apche Cassandra et MongoDB) ou de toute autre pile logicielle utilisée.

Les applications sont constituées d'au moins un framework qui est contenu dans un gabarit et s'exécute dans un ou plusieurs conteneurs d'applications. Des gabarits supplémentaires

peuvent être ajoutés à une application dans un ou plusieurs conteneurs différents. En utilisant SELinux, les utilisateurs ont seulement la permission de modifier leurs propres fichiers d'applications. Ainsi, les comptes des utilisateurs sont accessibles via SSH permettant de sécuriser la communication entre la machine du développeur et l'environnement d'OpenICRA.

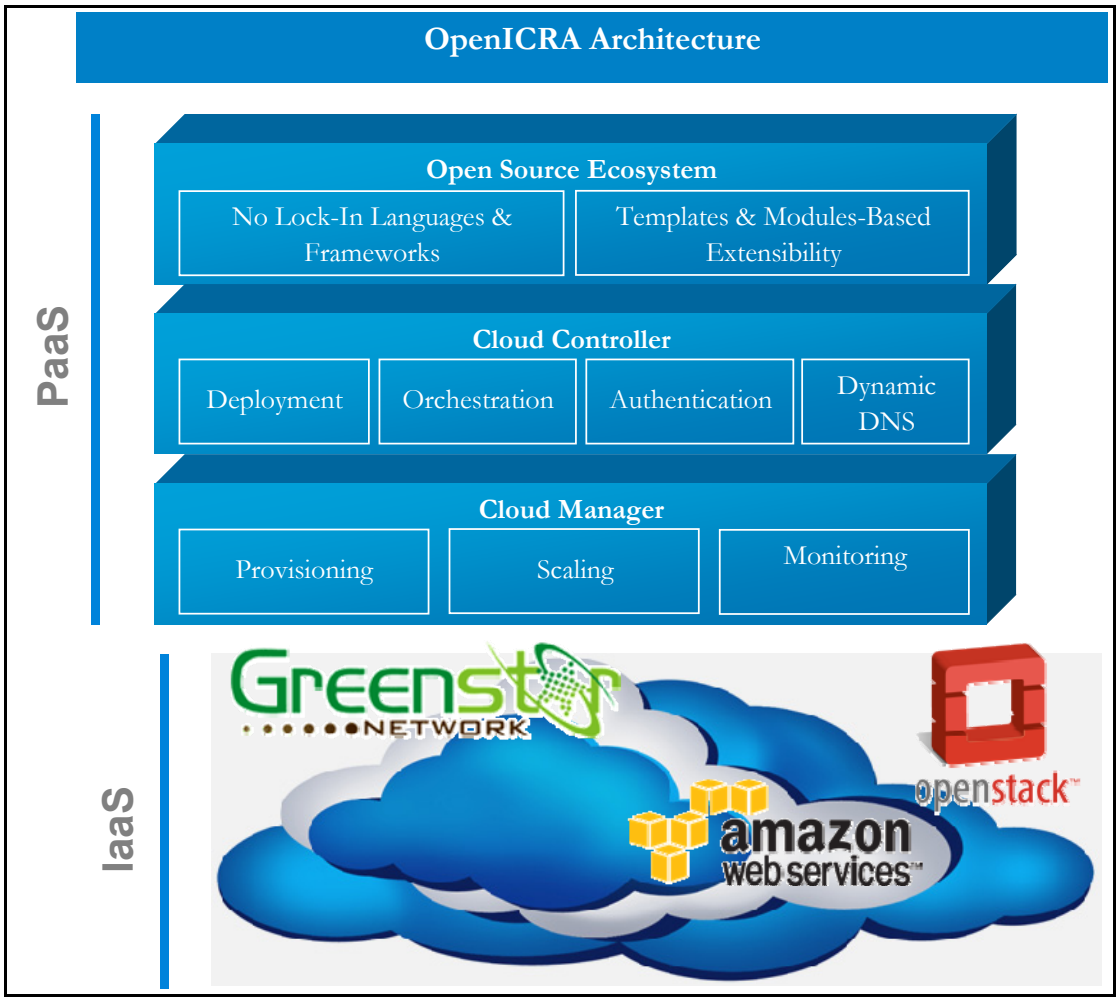

Figure 4.1 Architecture globale d'OpenICRA

Il est important de noter que notre modèle proposé OpenICRA se base principalement sur des composantes libres «open sources» et n'utilise aucune technologie propriétaire. En conséquence, il supporte la portabilité lui permettant de fonctionner sur toute infrastructure de nuage tel que GSN, EC2 d'Amazon ou avec le logiciel de gestion du Cloud OpenStack.

De plus, l'application d'utilisateur peut être exportée à partir de la plateforme OpenICRA pour s'exécuter sur tout autre environnement.

4.2.3 Contrôleur de nuage

Le contrôleur ou le Cloud Controller est le cerveau du système. Il manipule la création, la gestion et l'orchestration générale du déploiement automatique des applications, y compris l'authentification des utilisateurs et la communication avec le Cloud Manger et les nœuds de données. Il gère aussi le service DNS et les métadonnées des applications. Ainsi, les utilisateurs n'ont pas un contact direct avec le contrôleur, mais ils utilisent l'interface web console ou les outils de CLI pour interagir avec ce dernier via l'API du style d'architecture REST ou SSH.

Le contrôleur doit gérer lui-même les services offerts aux applications et aux utilisateurs. Ces services sont classés en trois sections qui sont l'authentification, les métadonnées et les noms de domaines des applications (DNS). En outre, il utilise trois interfaces distinctes afin de s'interagir avec les différents systèmes comme le montre la figure 4.2.

Authentification : Cette interface est utilisée pour gérer l'authentification et l'autorisation des utilisateurs via LDAP ou Kerberos KDC.
Métadonnées : Les métadonnées des applications, utilisateurs et datanodes sont sauvegardées dans une base de données NoSQL MongoDB.
DNS : Le serveur BIND est le responsable de la gestion dynamique des URLs des applications déployées par le développeur.

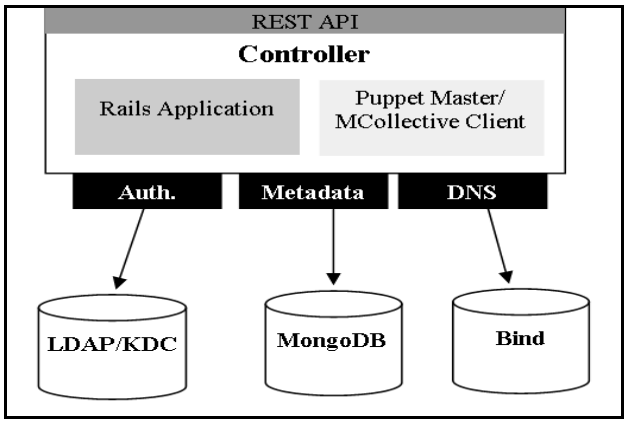

Figure 4.2 Contrôleur de nuage

Tous ces services peuvent être personnalisés selon les besoins. En effet, le service de stockage de métadonnées, d'authentification et le serveur DNS peuvent être implémentés en tant que des plug-ins. Ils peuvent aussi résider dans la même machine que le contrôleur ou dans des machines physiques ou virtuelles séparées.

L'application Rails hébergée dans la machine du contrôleur permet de transférer les demandes des utilisateurs provenant de l'API REST vers le gestionnaire de configuration automatisée Puppet. Ce dernier est le responsable d'automatisation de déploiement des applications. Ainsi, l'utilisateur peut transférer le module de déploiement de son application à Puppet à travers SSH ou via l'interface web console. Puis, il peut utiliser son outil de CLI pour interagir avec MCollective afin de déclencher le processus d'automatisation de déploiement de l'application. Par la suite, Puppet analyse le module de l'utilisateur, approvisionne les ressources nécessaires via le Cloud Manager et finalement lance d'une façon automatique le déploiement de l'application dans le cluster des nœuds de données (datanodes). Selon le type d'approvisionnement de ressources demandé, l'utilisateur peut avoir un contrôle total de la machine virtuelle ou seulement des droits de modifications des fichiers de ses applications.

4.2.4 Le gestionnaire de nuage

Le Cloud Manager est le gestionnaire des ressources de nuage. Il s'agit d'une couche d'abstraction pour l'environnement de nuage sous-jacent. Il communique avec le contrôleur de nuage à travers MCollective pour répondre aux besoins des utilisateurs en termes d'approvisionnement des ressources de calcul et de stockage. La figure 4.3 montre une vue logique de ses différents composants.

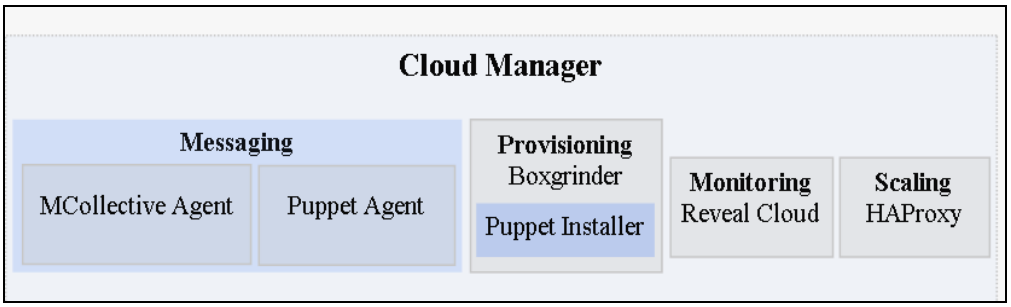

Figure 4.3 Gestionnaire de nuage

Le Cloud Manager est responsable de l'interfaçage avec l'infrastructure de nuage pour approvisionner les ressources de calcul à la demande et qui sont nécessaires pour le fonctionnement des applications. Il utilise l'outil Boxgrinder pour créer de nouveaux conteneurs ou machines virtuelles qui hébergent les fichiers des applications. Ainsi, il se connecte via SSH à la ressource approvisionnée pour installer automatiquement le gestionnaire de configuration Puppet. Ce dernier se connecte au Puppet Master pour vérifier s'il y a de nouveaux manifests à compiler.

Le Cloud Manager assure aussi la surveillance permanente des applications en utilisant l'API de l'outil de monitorage. Les résultats de suivi des applications peuvent être affichés sur un tableau de bord à travers un simple fureteur. Au moment de l'approvisionnement de ressource, le Cloud Manager fournit un template d'un équilibreur de charge avec la machine virtuelle ou le conteneur demandé. Les processus de l'application s'exécutent dans un conteneur autre que celui de l'équilibreur de charge. Ce dernier se positionne entre l'application et le réseau Internet et il route les demandes vers le conteneur de l'application.

Lorsque la demande augmente, il déclenche des règles de mise à l'échelle automatique en créant une copie de l'application dans un autre conteneur et il commence à équilibrer la charge entre les différents conteneurs de l'application.

4.2.5 Les ressources du système

4.2.5.1 Les nœuds de données

Les nœuds de données ou datanodes sont des machines physiques ou virtuelles. Ces ressources sont représentées par l'écosystème open source dans la figure 4.1. Ainsi, ces nœuds sont partitionnés en des conteneurs en utilisant le module de sécurité SELinux. Cette méthode de partitionnement est inspirée de la technique utilisée par la plateforme OpenShift pour créer leur gears (conteneurs). Les nœuds de données hébergent les templates des frameworks utilisés pour le déploiement des applications dans le nuage. Ils sauvegardent aussi les fichiers des applications et exécutent leurs processus.

4.2.5.2 Les conteneurs d'applications

Les conteneurs des applications sont des partitions virtuelles ou des environnements virtuels (VE) gérés par le module de sécurité SELinux et ils sont connus aussi par le nom anglais *Virtual Private Server* (VPS).

Les conteneurs fournissent une ressource informatique limitée pour exécuter un ou plusieurs modules/templates. Ils limitent la quantité de la mémoire RAM et de l'espace de stockage disponible pour les applications.

La figure 4.4 illustre la relation entre les différentes ressources du système. Ainsi, le système OpenICRA crée plusieurs conteneurs dans chaque machine virtuelle et les distribue dynamiquement à travers les différents nœuds de données.

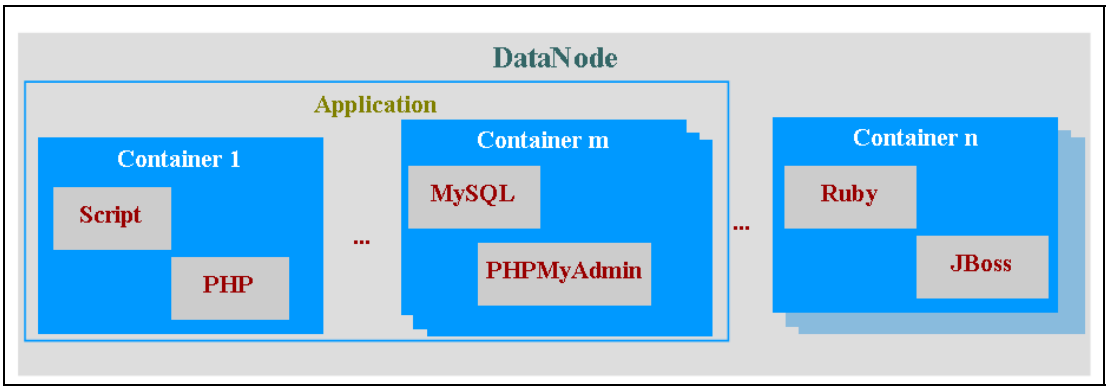

Figure 4.4 Relation entre les différentes ressources du système

En général, une application se compose d'au moins un framework qui est contenu dans un template et qui s'exécute dans un ou plusieurs conteneurs dans le cas d'un cluster ou de mise à l'échelle. Des templates supplémentaires peuvent être ajoutés à la demande dans les mêmes ou différents conteneurs.

4.2.6 Les Modules/Templates

Les modules sont utilisés par l'outil Puppet pour automatiser le déploiement des applications, alors que les templates sont hébergés dans les nœuds de données et généralement utilisés pour créer ou développer une nouvelle application dans le nuage ou pour déployer une application existante.

4.2.6.1 Les modules

Les modules sont des répertoires utilisés par Puppet Master pour déclencher des déploiements automatisés dans les nœuds de données à travers Puppet Agent et MCollective Server. Ils sont aussi utilisés pour migrer des applications existantes vers le nuage. Leur structure de base est typiquement la suivante :

- mymodule/ (le nom du module)
  - manifests/

- init.pp (contient la classe mymodule)
- public.pp (contient la classe mymodule**::**public)

Chaque fichier (.pp) du répertoire manifest contient une seule classe. Les classes de Puppet sont des morceaux de code de Puppet qui permettent de déployer des applications ou configurer des aspects spécifiques d'une machine ou d'un cluster de nœuds.

4.2.6.2 Les templates

Les templates sont des conteneurs ou des répertoires qui hébergent les framewoks ou les composants qui peuvent être utilisés pour déployer ou créer une application. Ils sont sauvegardés dans les nœuds de données et ils supportent aussi le déploiement de toute application binaire sans aucune modification du code source. Les templates fournissent la fonctionnalité réelle nécessaire pour exécuter l'application de l'utilisateur. Les langages de programmation supportés sont PHP, Ruby, Java, Python, etc., et plusieurs types de bases de données tels que Postgres, Mysql Mongo, etc.

4.2.7 Les mécanismes de communication du système

La communication à partir de tout client externe (par exemple, SSH, Web Console) vers le PaaS OpenICRA s'effectue à travers l'API du style d'architecture REST qui est hébergée par le contrôleur de nuage. Ce dernier communique avec les nœuds de données et le gestionnaire de nuage (Cloud Manager) à travers le serveur de queues ActiveMQ. Mcollective est utilisée pour faciliter l'interrogation et l'orchestration de l'ensemble de nœuds de données et du Cloud Manager ainsi que la communication avec les nœuds individuels d'une manière sécurisée.

4.3 Choix des solutions technologiques

Afin de faciliter l'intégration et l'implémentation des divers composants du modèle proposé, il est donc nécessaire d'analyser et de comparer les différentes solutions technologiques utilisées.

4.3.1 Choix d'une plateforme en tant que service (PaaS)

La plateforme en tant que service (PaaS) est le composant principal de notre projet de recherche. En effet, un PaaS permet aux utilisateurs de créer, déployer et gérer des applications dans le nuage. Selon Khazret (2013), il existe autour de 70 PaaS différents (mars 2013) et le nombre reste en croissance continue. Dans cette situation, il est intéressant d'établir une liste de critères souhaités afin de comparer les différents PaaS et choisir la solution la plus adaptée à nos besoins.

4.3.1.1 Critères de choix

Tableau 4.1 Critères de sélections des plateformes en tant que service

Critère	Poids	Description
Richesse en langages de programmation et technologies supportés	3	La plateforme PaaS doit supporter les langages de programmation les plus utilisés pour le développement des applications tels que Java, PHP et Ruby. Seulement les langages et outils libres doivent être utilisés et le PaaS ne doit pas avoir ni des APIs ni des technologies propriétaires. Cela garantit l'interopérabilité entre les environnements de nuage et la portabilité des applications à la fois sur et hors de la plate-forme, empêchant ainsi l'enfermement propriétaire.

Critère	**Poids**	**Description**
Licence	2	Elle doit être libre et distribuée sous une licence permettant la modification de son code source et de l'adapter selon nos besoins.
Disponibilité du code source	2	La plateforme doit avoir un code source disponible et accessible sans aucune restriction.
Extensibilité	2	Elle doit avoir une architecture modulaire permettant d'intégrer de nouveaux modules et d'autres outils pour la faire évoluer en fonction des besoins demandés.
Gestion automatique des noms de domaine	1	La plateforme doit être capable de gérer et d'affecter automatiquement des noms de domaine pour chaque utilisateur.
Équilibrage de charge automatique	1	Elle doit supporter la technologie de l'équilibrage de charge automatique en ajoutant plus de ressources s'il y a une augmentation de la demande, permettant d'éviter le goulet d'étranglement et de garantir une haute disponibilité des services
Outils de gestion	1	La plateforme doit disposer d'un ensemble complet d'outils de programmation et de gestion tels qu'une interface en ligne de commande et une interface web utilisateur. La combinaison de ces deux types d'interfaces permet aux développeurs d'utiliser des scripts pour les interactions automatisées à travers la ligne de commande et de gérer facilement leurs applications via l'interface web tout en réduisant la complexité des tâches de développement.

4.3.1.2 Sélection du PaaS

La liste ci-dessous présente les solutions PaaS existantes les plus populaires. Intentionnellement, cette liste contient divers types de systèmes de nuage (PaaS) qui supportent des technologies et des services différents. Ils ont été sélectionnés en mettant l'accent sur leur popularité (Jeff, 2013) et leur potentiel futur en tant que produits soutenus par des sociétés bien connues.

Google App Engine (Google, 2012) est une plate-forme de nuage informatique pour développer et héberger des applications Web dans les centres de données gérés par Google. Elle virtualise les applications sur plusieurs serveurs. Dans le cas d'une augmentation de la demande, App Engine traite le surcroît par une mise à l'échelle automatique pour les applications web en attribuant automatiquement davantage de ressources pour répondre à la demande supplémentaire. Les principaux langages supportés sont Java, Python, Ruby, Groovy, etc. Les applications développées par App Engine peuvent profiter des mêmes technologies évolutives que celles utilisées par Google telles que Google File System (GFS) et BigTable.

Cloud Foundry (VMware, 2013) est une plateforme en tant que service libre offrant aux utilisateurs la possibilité de choisir librement le prestataire de nuage, les plateformes de développement et les services applicatifs. Initié par VMware, avec le vaste soutien de l'industrie, Cloud Foundry rend plus rapide et plus facile à construire, à tester, à déployer et à mettre à l'échelle les applications. Elle supporte plusieurs technologies incluant Java, Ruby, Node.js, Groovy, etc. Cloud Foundry est distribuée sous la licence Apache 2.0 et a été principalement écrite en langage de programmation Ruby. La plateforme Cloud Foundry est un projet disponible à travers une variété de distributions de nuage privé et des instances de nuage public, y compris CloudFoundry.com.

Amazon Elastic Beanstalk est un PaaS offert par Amazon (2012b) pour déployer et gérer des applications sur le nuage AWS. AEB simplifie le déploiement d'applications dans le

nuage en supportant différentes technologies telles que Java, PHP, Python et .NET. Comme le PaaS est basé sur l'IaaS sous-jacente d'Amazon, le client garde le contrôle complet sur les ressources AWS et tire profit des services connectés tels que l'équilibrage de charge, la mise à l'échelle automatique et la surveillance de l'état des applications. Par conséquent, les services sont tarifés en fonction des ressources IaaS que l'AEB consomme.

Microsoft Windows Azure (Microsoft, 2012) est une plateforme en tant que service construit sur le nuage de Microsoft. Il s'agit d'une offre d'hébergement et de services. Cette plateforme permet de déployer des applications à travers le réseau de centres de données gérés par Microsoft. Elle permet aussi aux développeurs de créer des applications à l'aide de plusieurs technologies supportées telles que Java, PHP, .Net, Python et Node.js.

OpenShift de Red Hat (2013) est une plateforme en tant que service (PaaS) libre. OpenShift offre aux développeurs une plateforme pour créer, tester, déployer et exécuter des applications sur un environnement de nuage. Elle prend en charge toutes les infrastructures, les middlewares et les outils de gestion de configuration, laissant au développeur la liberté de se concentrer seulement sur la conception et le développement de ses applications. Les développeurs peuvent utiliser le logiciel de gestion de version décentralisé Git pour déployer des applications en différents langages sur la plateforme. De plus, elle supporte une large liste de technologies d'applications incluant Python, PHP, Perl, Java, MySQL, PostgreSQL et MongoDB. L'infrastructure OpenShift permet aussi de surveiller le trafic web entrant pour réagir dans le cas de l'augmentation de la demande en allouant automatiquement les ressources nécessaires afin de traiter les demandes entrantes.

Heroku (Wauters, 2010) est un service de nuage informatique de type plateforme en tant que service. Il était créé en 2007 puis racheté par salesforce.com en 2010. Heroku a soutenu initialement le langage de programmation Ruby avec Rack et Ruby on Rails. Ensuite, l'offre s'est étendue pour supporter d'autres langages tels que node.js, java, spring, ruby, php, scala, etc. La plateforme permet de déployer de façon très rapide des applications web dans le

nuage avec une gestion de mise à l'échelle horizontale très souple au travers d'un modèle de gestion des processus adaptable.

Force.com est la plateforme en tant que service de Salesforce (2013). Elle permet aux développeurs de créer des applications d'entreprise et les héberge sur l'infrastructure de Salesforce.com. Ces applications sont construites en utilisant Apex (un langage de programmation propriétaire de type Java pour la plateforme Force.com) et Visualforce (une syntaxe de type XML pour créer des interfaces utilisateurs en HTML, Ajax ou Flex).

DotCloud est un service de nuage informatique de type PaaS pour les développeurs qui offre un support pour plusieurs technologies telles que PHP, Ruby, Python, Perl, Java, Node.JS, MySQL, Redis, RabbitMQ, Solr, MongoDB et PostgreSQL. La plateforme permet aux développeurs et aux organisations de déployer, gérer et faire évoluer leurs applications avec une grande facilité en assemblant et personnalisant des services et des piles logicielles préconfigurées.

CloudBees (2013) vise à fournir une plate-forme Java en tant que service (PaaS) couvrant à la fois le développement des services et l'exécution des produits en Java. Le PaaS est axé sur la simplification et l'accélération du cycle de vie entier d'une application Java, ce qui le rend plus facile et plus rapide pour construire, exploiter et gérer les applications Java dans le nuage. Il prend en charge tous les langages basés sur JVM tels que Java, Spring, JRuby, Grails, Scala, Groovy et autres. Il n'y a pas d'enfermement propriétaire avec CloudBees puisqu'il n'y a pas des APIs spécifiques ou des composants java non standard. Ainsi, si le développeur n'est pas satisfait par les services de Cloudbees, il peut migrer facilement vers un autre environnement sans aucune restriction.

CumuLogic (2013) est un service de nuage informatique de type plateforme en tant que service (PaaS) pour les entreprises. Cette plateforme évolutive permet de développer et déployer facilement des applications web et mobiles sur le nuage. Java et PHP sont les seuls langages supportés par CumuLogic. Elle permet donc aux entreprises et fournisseurs de

nuage informatique de construire des services de type AWS d'Amazon compatibles dans leurs propres centres de données sur leur nuage IaaS ou sur un environnement virtualisé.

Le tableau suivant compare les différents PaaS identifiés ci-dessus selon les critères établis:

Tableau 4.2 Évaluation des PaaS par rapport aux critères de sélection établis

Critère / Plateforme	Richesse en langages de programmation et technologies supportés	Licence	Disponibilité du code source	Gestion automatique des noms de domaine	Extensibilité	Équilibrage de charge automatique	Outils de gestion
Cloud Foundry	+	+	+	-	+	-	+/-
Amazon Elastic Beanstalk	+	-	-	+	-	+	+
Heroku	+	-	-	+	-	+	+
Google App Engine	+	-	-	+	-	+	+
Microsoft Windows Azure	+/-	-	-	+	-	+	+
Force.com	-	-	-	+	-	+	+
DotCloud	+	-	-	+	+/-	+	+
CloudBees	+/-	-	-	+/-	-	+	+
CumuLogic	+/-	-	-	+	-	+	+
OpenShift	+	+	+	+	+	+	+

Le but de cette comparaison est de trouver la plateforme en tant que service la plus appropriée pour le déploiement d'applications dans le nuage. Le processus de sélection a été basé sur plusieurs critères techniques et non techniques qui sont décrits ci-dessus. En plus de ces critères, le PaaS cible doit fournir aussi un environnement de développement pour les

applications afin de simplifier le processus de développement et d'accélérer le processus de déploiement dans le nuage.

Ainsi, le tableau 4.2 permet de constater qu'OpenShift est la solution la plus appropriée au contexte de la recherche. En effet, elle offre un PaaS libre avec une large liste de fonctionnalités telles que la gestion de stockage, l'équilibrage de charge automatique, la gestion automatique des noms DNS et supporte que des technologies libres permettant d'éviter l'enfermement propriétaire et d'assurer la portabilité en toute simplicité des applications déployées.

La sous-section suivante décrit les principales caractéristiques d'OpenShift ainsi que son architecture globale.

4.3.1.3 Description d'OpenShift

OpenShift est une plateforme en tant que service libre permet de créer, déployer et gérer des applications dans le nuage. Elle fournit de l'espace disque, des ressources CPU, mémoire, connectivité réseau et un serveur web. Selon le type d'application à déployer, OpenShift donne accès à un gabarit du système de fichier pour ce type (par exemple, PHP, Python et Ruby/Rails). Elle gère également les configurations DNS des applications.

Les principales caractéristiques de la plateforme OpenShift sont les suivantes :

Minimisation de l'enfermement propriétaire: OpenShift est construite sur des piles technologiques libres et elle est conçue pour fournir la liberté de choisir le fournisseur de nuage ou le PaaS qui satisfait le besoin de l'utilisateur. À cette fin, la plateforme utilise seulement les langages de programmation et *frameworks* libres et aucune technologie propriétaire n'est utilisée. Cela garantit la portabilité des applications et empêche l'enfermement propriétaire sur la plateforme OpenShift.

Plateforme polyglotte: La plateforme offre aux développeurs la possibilité de choisir le bon outil pour chaque projet en fournissant une large liste de langages de programmation tels que Java, Ruby, Node.js, Python, PHP et Perl. En plus des langages de programmation ouverts, la plupart des frameworks populaires libres comme Rails, Django, EE6, Spring, Play, Sinatra et Zend, sont aussi inclus dans OpenShift.

Services de base de données intégrés: OpenShift offre aux développeurs des technologies de base de données permettant d'avoir des instances de bases de données connectées automatiquement dans les piles d'applications selon les besoins. Donc, les développeurs peuvent choisir selon leurs besoins une base de données de type relationnel classique ou NoSQL moderne.

Extensibilité : Les développeurs peuvent ajouter un autre langage, base de données, ou des composants middleware dont ils ont besoin via le système personnalisable de gabarit. Cette extensibilité permet aux développeurs d'étendre facilement le PaaS pour supporter d'autres normes ou exigences spécifiques.

L'architecture globale d'OpenShift est présentée dans la figure 4.5. Les deux unités fonctionnelles de base de la plate-forme sont l'intergiciel '*brocker*', qui assure l'interface avec les autres composants, et les gabarits '*cartridges*', qui offrent les cadres des applications.

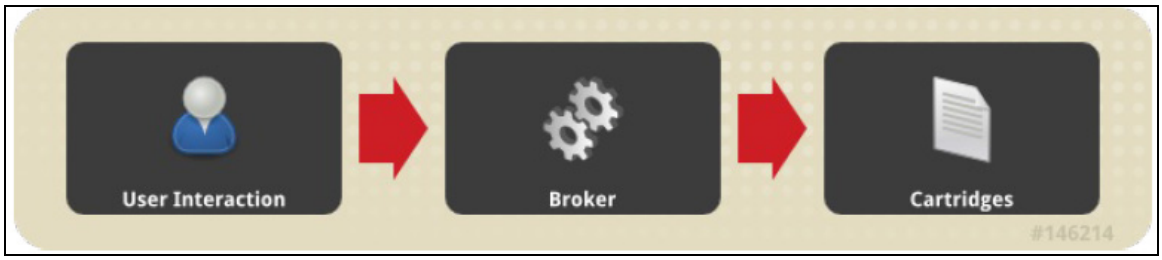

Figure 4.5 Vue d'ensemble de la plateforme OpenShift
Tirée de Muhammad (2012)

Brocker : Le *brocker* est le point de contact unique pour toutes les activités de gestion des applications. Il est chargé de gérer les connexions des utilisateurs, les comptes, le DNS, les états des applications et l'orchestration générale de toutes les applications. L'interaction de l'utilisateur avec le brocker est généralement réalisée en utilisant soit la console Web ou l'outil de CLI via une API basée sur le style d'architecture pour les systèmes distribués REST.

Cartridges : Les *cartridges* nommées aussi les gabarits fournissent les fonctionnalités nécessaires pour exécuter les applications des utilisateurs. Ainsi, de nombreuses *cartridges* sont actuellement disponibles dans OpenShift pour soutenir différents langages de programmation tels que Perl, PHP et Ruby, et d'autres *cartridges* de bases de données comme PostgreSQL, MySQL et Mongo.

En plus du brocker et cartridges, OpenShift a d'autres composants qui sont les partitions et qui s'appellent aussi 'gears'.

Gears : Elles fournissent des conteneurs de ressources limités où on peut exécuter une ou plusieurs *cartridges*. De plus, elles permettent de limiter la quantité de RAM et de l'espace disque disponible pour chaque *cartridge*.

La figure 4.6 illustre la relation entre l'application, les cartridges et les gears dans un environnement d'OpenShift.

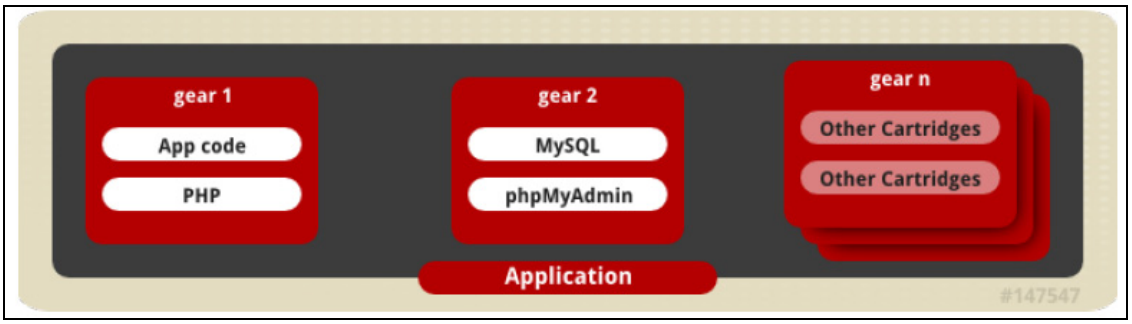

Figure 4.6 Des templates d'une application sur des gears
Tirée de Muhammad (2012)

Il est important de noter que chaque application peut utiliser une ou plusieurs partitions avec différents gabarits. Il est aussi possible que chaque partition soit assignée pour une application séparée. Également, plusieurs partitions ou ‘gears’ s’exécutent sur une seule machine physique ou virtuelle qui s’appelle hôte ou nœud.

4.3.2 Choix d'outil de gestion de configuration

Le système de gestion de configuration constitue un élément essentiel du modèle proposé. Son intégration dans notre solution de recherche permet d'automatiser les tâches répétitives, de déployer rapidement des applications critiques et de gérer de façon proactive les changements d'infrastructure sur site (on-premises) ou dans le nuage.

Il existe trois solutions majeures libres de gestion de configuration qui sont Puppet (Turnbull et McCune, 2011), Chef (Nelson-Smith, 2011) et CFEngine (Ressman et Valdés, 2000). Dans ce qui suit, nous présentons une comparaison de ces trois solutions pour choisir l'outil le plus adapté à nos besoins et faciliter son intégration dans la plateforme en tant que service (PaaS).

4.3.2.1 Comparaison

Puppet, Chef et CFEngine sont trois outils majeurs libres désignés spécialement pour la gestion de configuration des systèmes informatiques. Ils partagent la même idée qui consiste à automatiser les tâches de configuration dans un cluster de nœuds et éviter ainsi les tâches répétitives.

Le tableau 4.3 compare ces trois solutions présentant en détail les caractéristiques de chaque outil.

Tableau 4.3 Comparaison de Puppet, Chef et CFEngine

Outil / **Critère**	**Puppet**	**Chef**	**CFEngine**
Licence	Apache	Apache	GPL
Technologie	Ruby	Ruby	C
Architecture	- Client/serveur - Modèle autonome	Client/serveur	Client/serveur

Outil / Critère	Puppet	Chef	CFEngine
Langage de configuration	Langage déclaratif	Ruby	Langage déclaratif
Méthode de configuration	- Fichier plein texte - Outil CLI - Interface web	- Fichier plein texte - Outil CLI - Interface web	Fichier plein texte
Systèmes d'exploitation supportés	- Linux/Unix - Windows (expérimental)	- Linux/Unix - Windows (expérimental)	- Linux/Unix - Windows (expérimental)
Cloud	Oui	Oui	Oui
Orchestration	Oui (MCollective)	Non	Non
Commentaire	- Créé en 2006 par Puppet Labs ; - Le plus populaire - La solution la plus simple.	- Créé en 2009 par OpsCode - Évolué à partir du Puppet ; - Très récent par rapport à Puppet et CFEngine mais très prometteur.	- Le premier outil de gestion de configuration créé par Cfengine en 1993 ; - L'outil le plus complexe.

La liste des outils de gestion de configuration cités dans le tableau ci-dessus n'est pas une liste exhaustive puisqu'il y a d'autres logiciels qui sont disponibles, toutefois, ils ne répondent pas aux besoins du projet.

4.3.2.2 Choix de l'outil

Le tableau 4.3 permet de constater que Puppet est l'outil le plus approprié pour notre projet de recherche. En effet, Puppet est l'outil le plus simple à intégrer dans notre PaaS et le plus complet. De plus, il est l'unique outil qui offre des services d'orchestration en parallèle et en temps réel dans un cluster de nœuds à travers une interface utilisateur web centralisée ou en ligne de commande permettant la gestion de la configuration des systèmes en quelques minutes plutôt qu'en plusieurs heures ou jours.

4.3.3 Choix du système de fichiers distribué

Dans un environnement de production où il y a plusieurs serveurs voulant accéder aux mêmes fichiers, il est toujours difficile d'établir une architecture adéquate de gestion du stockage. Cependant, il existe divers moyens permettant le partage des fichiers avec des serveurs multiples. Un de ces moyens consiste à implémenter un système de fichiers distribué (DFS) offrant aux utilisateurs d'ordinateurs distribués physiquement la possibilité de partager des données et des ressources de stockage.

Ainsi, les systèmes de fichiers distribués sont constitués de différents composants logiciels exécutés sur plusieurs ordinateurs, mais fonctionnent comme un système unique. Ils peuvent être avantageux, car ils facilitent la distribution des documents à de multiples clients et ils fournissent un système de stockage centralisé.

Afin de faciliter l'intégration du système de fichiers distribué choisi dans notre modèle proposé, il est important d'établir une liste de critères de sélection souhaités. Par la suite, nous utilisons cette liste pour choisir le système de fichiers le plus adapté à nos besoins en tenant compte de son implémentation dans des systèmes informatiques du réseau à très grande échelle tels que le nuage informatique, la grille informatique, etc.

4.3.3.1 Critères de choix

Évolutivité

L'évolutivité désigne la capacité d'un système de fichier de maintenir ses fonctionnalités et ses performances en cas de forte demande. Pour ce critère, nous allons évaluer les systèmes de fichiers selon le nombre de machines supporté par chaque cluster de DFS.

Architecture

Il existe différents types d'architecture de systèmes de fichiers distribués :

- architecture asymétrique dont le système de fichier est maintenu par des gestionnaires de métadonnées dédiés.
- architecture symétrique qui est basée sur la technologie pair-à-pair où tous les nœuds hébergent le code du gestionnaire de métadonnées et comprennent la structure du système de stockage.
- architecture client-serveur qui offre une vue normalisée de son système de fichiers local.
- architecture en grappe qui est composée de deux types de nœuds. Un nœud unique maître « master » qui gère le système de fichier et plusieurs nœuds esclaves qui gèrent les données enregistrées.
- architecture parallèle dont les blocs de données sont rayés, en parallèle, à travers plusieurs périphériques de stockage sur divers serveurs. Cette architecture permet également un accès au même fichier en même temps, et prend en charge les capacités de lecture/écriture simultanément.

Processus

Il y a deux types de processus :

- un processus est dit avec état (Stateful) si chaque opération de fichier est considérée comme faisant partie d'une session. L'avantage de cette approche supportée par la plupart des systèmes de fichiers est qu'elle prend en charge la mise en cache des informations des fichiers ce qui permet l'accélération d'accès aux données.
- un processus est dit sans état (Stateless) s'il n'y a aucune information maintenue sur le client. Ainsi, le verrouillage de fichiers n'est pas possible. Quand un accès reprend, le

processus recommence une nouvelle demande de fichier. L'avantage principal de l'approche sans état réside dans sa simplicité.

Communication

Les DFSs utilisent l'appel de procédures à distance (RPC) comme un mode de communication leur permettant d'être indépendants du système d'exploitation, des réseaux et des protocoles de transport sous-jacents. Dans l'approche RPC, il existe deux protocoles de communication à prendre en compte qui sont TCP et UDP.

- TCP est pratiquement utilisé par tous les DFSs.
- UDP est utilisé dans Hadoop pour améliorer ses performances.

Nommage

Les approches communes utilisées par les DFSs sont les suivantes:

- un serveur central de métadonnées pour gérer l'espace de nom de fichier. Par conséquent, le découplage des métadonnées et des données améliore l'espace de noms de fichiers et résout le problème de synchronisation.
- des métadonnées distribuées dans tous les nœuds qui résultent une compréhension de la structure du disque.

Synchronisation

La synchronisation est obtenue par l'utilisation de mécanismes de verrouillage des fichiers et des baux. Ces techniques de verrouillage diffèrent en fonction du but des applications. Dans les solutions de stockages existantes, la technique *écrire une fois et lire autant de fois,* connue sous le terme anglais WORM « Write Once Read Many », est la plus utilisée. Certains systèmes choisissent de donner les verrous sur les objets pour les clients et certains choisissent d'effectuer toutes les opérations de façon synchrone sur le serveur. D'autres systèmes choisissent d'appliquer une solution hybride pour leur système de verrouillage de fichiers. L'utilisation de système de bail, où le maître donne au serveur pour une durée limitée le droit d'effectuer des écritures, est la méthode la plus courante pour contrôler l'accès parallèle aux DFSs.

La cohérence et la réplication

La plupart des DFSs utilisent la somme de contrôle pour valider les données après l'envoi via le réseau de communication. La mise en cache et la réplication joue un rôle important quand elles sont configurées pour fonctionner sur un réseau étendu. La plupart des méthodes utilisées sont les suivantes:

- mise en cache côté client lui permettant de communiquer avec le serveur pour vérifier la disponibilité des mises à jour des données mises en cache;
- réplication côté serveur lui permettant d'être informé pour chaque ouverture. Si un fichier est ouvert en écriture, alors les autres clients désactivent automatiquement la mise en cache pour ce fichier.

Tolérance aux pannes

Cette caractéristique est très liée à la fonction de réplication parce que son but est de fournir la disponibilité et de supporter la transparence des défaillances aux utilisateurs. Il existe deux approches pour la tolérance aux pannes:

- défaillance en tant qu'exception qui isole le nœud défaillant ou restaurer le système à partir du dernier état de fonctionnement normal.
- défaillance en tant que norme qui consiste à répliquer toutes sortes de données.

Sécurité

L'authentification et le contrôle d'accès sont parmi les problèmes de sécurité importants dans les DFS qui doivent être analysés. La plupart des DFSs emploient l'authentification, l'autorisation et la vie privée. Cependant, certains DFS tels que GFS et Hadoop sont basés sur la confiance entre tous les nœuds et les clients pour de fins spécifiques.

4.3.3.2 Comparaison qualitative des systèmes de fichiers distribués

Le processus de comparaison considère les systèmes de fichiers distribués les plus communs et les plus utilisés qui sont GFS de Google (Ghemawat, Gobioff et Leung, 2003), HDFS d'Apache (Borthakur, 2008), GPFS d'IBM (Schmuck et Haskin, 2002) et NFS de Sun

(Haynes et Noveck, 2013). Dans la littérature, il y a plusieurs autres systèmes de fichiers tels qu'AFS (Miloushev et Nickolov, 2011), Lustre (Zhao et al., 2010), S3 d'Amazon (2012b), etc. Cependant, ils ne satisfont pas les critères d'évaluation établis. Le résumé de la comparaison est présenté dans le tableau ci-dessous :

Tableau 4.4 Comparaison des différents systèmes de fichiers distribués

DFS / **Critère**	**GFS**	**HDFS**	**GPFS**	**NFS**
Évolutivité	10 000+ machines	10 000+ machines	1 000+ machines	10-20 machines
Architecture	Architecture en grappe, asymétrique parallèle	Architecture en grappe, asymétrique, parallèle	Architecture en grappe, symétrique, parallèle	Client Server, asymétrique
Processus	Avec état	Avec état	Avec état	Sans état
Communication	RPC/TCP	RPC/TCP &UDP	RPC/TCP & UDP	RPC/UDP ou TCP
Nommage	Serveur central de métadonnées	Serveur central de métadonnées	Métadonnées distribuées dans tous les nœuds	Métadonnées distribuées dans tous les noeuds
Synchronisation	WORM, verrouillage des objets par les clients, utilisation du système de bail	WORM, verrouillage des objets par les clients, utilisation du système de bail	WORM, verrouillage des objets par les clients, utilisation du système de bail	WORM, verrouillage des objets par les clients, utilisation du système de bail
Tolérance aux pannes	Défaillance en tant que norme	Défaillance en tant que norme	Défaillance en tant qu'exception	Défaillance en tant que norme

DFS / Critère	GFS	HDFS	GPFS	NFS
Sécurité	Pas de mécanisme de sécurité dédié	Pas de mécanisme de sécurité dédié	Listes de contrôle d'accès	Intégration de Kerberos, Authentification de toutes les opérations
Cohérence et réplication	Réplication côté serveur, réplication asynchrone, somme de contrôle, cohérence entre les réplications des objets de données	Réplication côté serveur, réplication asynchrone, somme de contrôle	Réplication côté serveur, Réplication de métadonnées seulement	Réplication seulement dans NFSv4

Les commentaires et les interprétations de la comparaison des différents systèmes de fichiers distribués sont présentés dans le tableau 4.5.

Tableau 4.5 Les avantages et les inconvénients des DFSs comparés

DFS	**Avantages**	**Inconvénients**
GFS	- Totalement distribué - Offre une haute disponibilité et une réelle évolutivité	- Système de gestion de cluster non standardisé. - Compatible seulement avec les distributions Linux - Logiciel propriétaire
GPFS	- Hautement évolutive - Réplication des données - Bonne performance - Stable	- Disponible seulement en version commerciale
HDFS	- Offre une formidable opportunité pour les applications qui utilisent des données massives. - Offre une grande flexibilité et puissance pour assurer le passage à l'échelle jusqu'à des milliers de machines. - Offre un accès haut débit aux données applicatives - Hautement disponible et résistant aux pannes - Évolution continue du système - Logiciel libre	- Nécessite d'importantes ressources pour réaliser l'avantage réel du système - Confus de l'aspect technique causé par les différentes directions de nombreuses organisations

DFS	Avantages	Inconvénients
NFS	- Adoption par l'IETF - Méthodes de récupération des données améliorées - Prend en charge le partage de fichiers de Windows	- L'implémentation dans Linux est relativement récente - Des implémentations maturent sur d'autres systèmes d'exploitation, mais elles ne sont pas encore largement déployées. - N'est pas évolutif

4.3.3.3 Sélection du système de fichiers distribué

Un système de fichier distribué est un moyen important qui peut offrir des avantages considérables à l'infrastructure du projet EcoloTIC (Ericsson et al., 2012), y compris une meilleure utilisation des ressources de stockage, un approvisionnement plus rapide et un meilleur positionnement pour faire face aux besoins croissants de stockage de données.

Dans cette section, nous avons comparé et analysé les systèmes de fichiers distribués les plus populaires et les plus communs. Cette étude comparative des DFSs vise à choisir la solution de stockage la plus adaptée à nos besoins en exposant les caractéristiques, les avantages et les inconvénients de chacun des DFSs dans des tableaux comparatifs. Les tableaux 4.4 et 4.5 permettent de constater que HDFS est le meilleur choix pour mettre en œuvre un système de gestion de stockage distribué, en termes de performance, de fiabilité et d'adaptabilité. Soutenu par des acteurs majeurs tels que Facebook, Yahoo, Twitter et Cloudera, HDFS est apparu comme une solution solide qui utilise une technologie hautement évolutive pour répondre aux enjeux des données massives. Ainsi, il est un système libre compatible avec l'environnement de nuage informatique qui peut être facilement intégré sans aucune

restriction avec des middlewares de gestion des ressources de nuage et il peut être aussi très utile pour gérer les données des applications déployées par des plateformes en tant que service dans un environnement de nuage.

4.4 Conclusion

En guise de conclusion à ce chapitre, nous notons que les choix des solutions technologiques pour la conception de notre modèle générique de déploiement des applications sont basés sur des comparaisons qualitatives de haut niveau. Ces solutions constituent les principaux composants architecturaux de notre modèle proposé, offrant ainsi de hauts niveaux de flexibilité, d'extensibilité et de performances. À la différence de solutions commerciales, ces composants se caractérisent par la liberté d'accès à leur code source. Ceci nous permet de garantir la portabilité des applications déployées dans tout environnement d'exécution, d'assurer la mise à l'échelle du modèle, d'éviter l'enfermement propriétaire des vendeurs et de faciliter le processus d'automatisation des applications dans le nuage.

CHAPITRE 5

EXPÉRIMENTATION ET VALIDATION

5.1 Introduction

Pour tester et valider notre modèle proposé OpenICRA, nous illustrons deux cas d'études concrets dont le premier consiste à automatiser le déploiement de l'intergiciel distribué OpenSAF dans un cluster de nœuds au sein de l'environnement de production du réseau GSN (Synchromedia, 2010), et le deuxième cas vise à migrer le système de travail collaboratif ICRA (Cheriet, 2012) vers le nuage EC2 d'Amazon (2012a). Pour cela, nous allons suivre le processus de déploiement automatique des applications décrit dans la section 3.6 pour réaliser les deux cas d'études et exposer les résultats obtenus à partir des environnements de nuage réel.

5.2 1er cas d'étude : Automatisation du déploiement d'OpenSAF

5.2.1 Présentation du cas d'étude

OpenSAF est un intergiciel libre activement soutenu par les principales compagnies de télécommunications au monde telles qu'Ericsson, HP, Nokia Siemens Networks, Sun Microsystems, etc. Cet intergiciel permet de développer un environnement d'exécution à haute disponibilité pour les applications et les équipements télécoms compatibles avec les spécifications du SAF (Service Availability Forum) et qui exigent un service ininterrompu. Il est devenu une norme pour les intergiciels qui offrent des services de haute disponibilité depuis 2010 lorsque Ericsson a annoncé officiellement son déploiement commercial au sein de l'environnement des services multimédia ECE (Ericsson Composition Engine) (Business Wire, 2011; OpenSAF team, 2013).

Dans le cadre du projet EcoloTIC, le partenaire industriel a proposé de déployer OpenSAF dans un large cluster de nœuds dans un environnement de nuage afin de bénéficier des

avantages de ce concept et de fournir un meilleur service de haute disponibilité pour développer des applications distribuées dans le nuage. Cependant, en raison de la complexité et les énormes variétés de scénarios de déploiements d'OpenSAF et les tâches répétitives et fastidieuses de configuration des services et des applications, le partenaire industriel a décidé d'automatiser son déploiement afin de réduire considérablement les erreurs et les coûts, d'accélérer la mise en œuvre des applications et de garantir la conformité, le contrôle et la gouvernance.

C'est dans ce contexte que nous allons utiliser notre modèle proposé pour automatiser le déploiement de l'intergiciel OpenSAF dans un environnement de production réel du réseau GSN.

5.2.2 Processus d'automatisation du déploiement d'OpenSAF

Avec notre modèle générique conçu pour supporter de différents scénarios de déploiement, l'automatisation de déploiement d'OpenSAF peut être réalisée dans un processus de trois étapes tel que décrit dans la section 3.6. La figure 5.1 illustre les étapes du processus d'automatisation du déploiement d'OpenSAF.

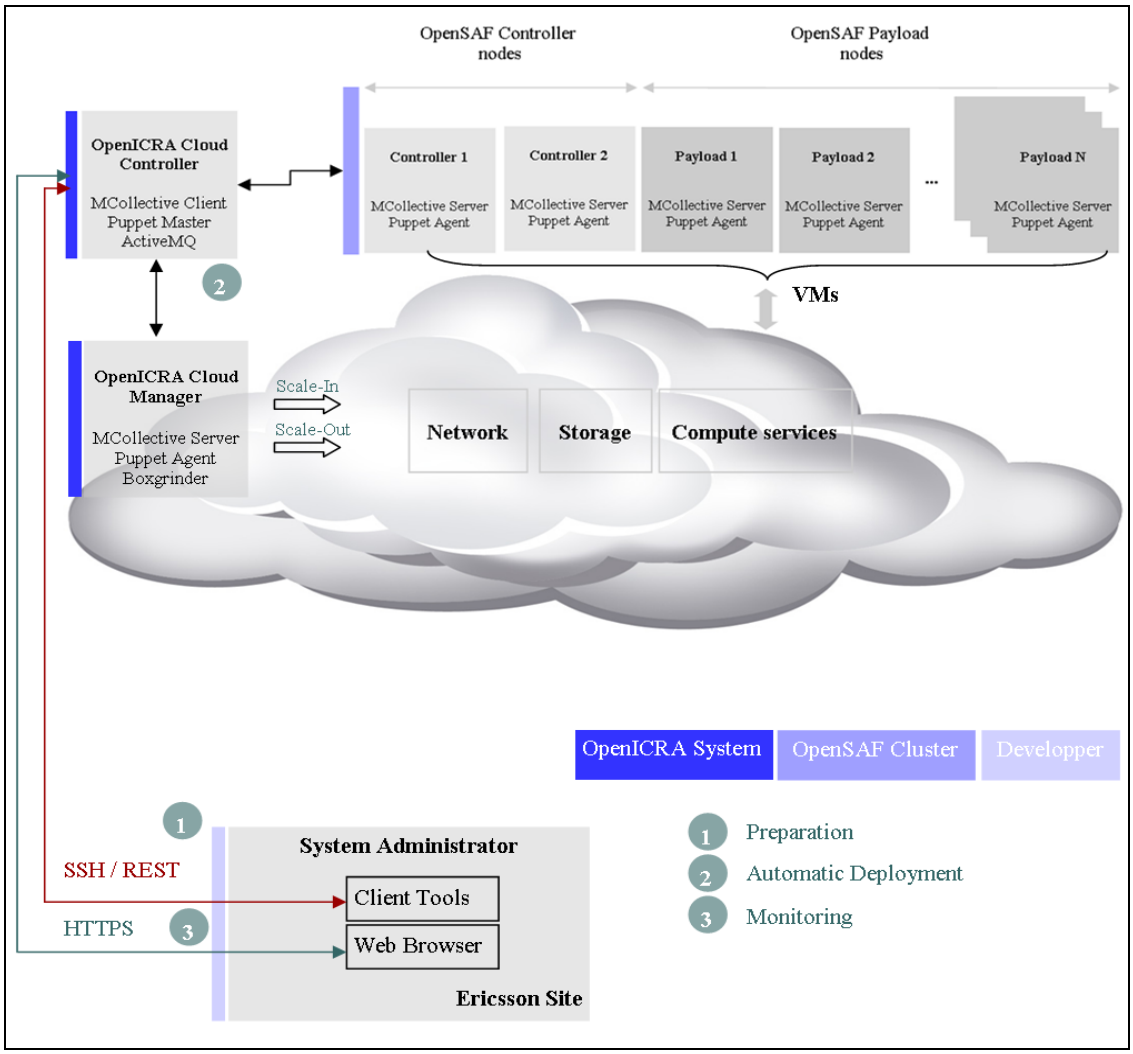

Figure 5.1 Processus de déploiement de l'intergiciel OpenSAF

5.2.2.1 Préparation du déploiement

Selon sa documentation officielle, la configuration d'OpenSAF recommandée permettant de démontrer ses diverses capacités de la haute disponibilité (HA) avec de meilleures performances nécessite au moins quatre serveurs, dont deux machines pour agir en tant que contrôleurs configurés en mode Active/Veille et au moins deux autres machines pour les payloads (entités où OpenSAF gère l'exécution de ses applications). Dans un cluster d'OpenSAF, un nœud ou un serveur peut être un contrôleur ou un payload et un maximum de

deux contrôleurs est permis. Tout autre nœud additionnel doit être configuré en tant que payload dans le cluster (Business Wire, 2011; OpenSAF team, 2013).

Pour assurer un déploiement réussi, nous devons comprendre la spécification et la caractérisation de l'application. Dans notre cas, l'architecture d'OpenSAF doit être soigneusement étudiée. En outre, selon son site officiel, les solutions des erreurs communes doivent être configurées dans chaque machine virtuelle. De plus, il est possible de déployer OpenSAF sur Red Hat Enterprise Linux et OpenSUSE à partir des dépôts de paquets habituels, mais cela n'est pas possible sur Ubuntu. Toutefois, OpenSAF peut être déployé par la compilation de son code source directement sur Ubuntu 12.04. Ainsi, nous avons opté pour la deuxième méthode afin de bénéficier de la souplesse d'Ubuntu malgré qu'il n'y ait pas de dépôt officiel pour OpenSAF qui peut être utilisé afin de télécharger ses paquets binaires à travers l'outil de gestion de paquets *apt-get*.

Pour les exigences des ressources de calcul et de stockage, nous avons utilisé la configuration des ressources présentée dans le tableau suivant :

Tableau 5.1 Configuration des ressources d'OpenSAF

	Nœud de données (datanode)	**Machine virtuelle**
Système d'exploitation	Ubuntu 12.04 LTS	Ubuntu 12.04 LTS
Matériel du serveur	HP	
Mémoire	4 GB	512 MB
Hyperviseur		KVM
Processeur	AMD Phenom(tm) 9600B Quad-Core Processor × 4	2 virtual cores
Espace de stockage	100 GB	8 GB
Domaine		synch.local
Hostname/Adresse IP		controller1/192.168.122.6
		controller2/192.168.122.7
		payload1/192.168.122.8
		payload2/192.168.122.8

Pour automatiser le déploiement d'OpenSAF, nous avons développé un module appelé « opensaf » qui décrit toutes les instructions nécessaires à l'automatisation du déploiement. Ce module est écrit en langage déclaratif et il utilise des classes, des fichiers de configurations et des fichiers binaires. Le code source de la classe « opensaf » est utilisé pour l'automatisation du déploiement de l'intergiciel OpenSAF en respectant toutes les exigences mentionnées ci-dessus.

Les fichiers binaires sont utilisés pour installer les dépendances lorsqu'il n'y a pas de connexion Internet dans l'environnement de déploiement. Dans le cas contraire, ils seront ignorés et les dépendances seront installées à partir des dépôts officiels des paquets. Les fichiers de configuration sont des gabarits préconfigurés utilisés pour assurer le bon fonctionnement, la compatibilité et la communication entre les différents composants d'OpenSAF.

Toutefois, les classes sont utilisées pour :

- demander au Cloud Manager d'approvisionner les ressources de calcul et de stockage indispensables pour le déploiement et installer automatiquement le gestionnaire de configuration Puppet;
- exécuter les mises à jour et les installations nécessaires des paquetages de dépendances ;
- orchestrer les nœuds du cluster ;
- appliquer la configuration recommandée par l'administrateur système.

5.2.2.2 Déploiement automatique

Après la préparation, la deuxième étape consiste à déclencher le script d'approvisionnement des ressources et de déploiement automatique d'OpenSAF. Ce script doit être lancé par le développeur ou l'administrateur à partir de sa machine locale via l'outil en ligne de commande « synchro » ou une session SSH pour déclencher le déploiement.

L'outil d'approvisionnement utilisé est vmbuilder, l'hyperviseur est KVM et le système d'exploitation invité est Ubuntu 12.04 LTS. En outre, l'approvisionnement des machines virtuelles et l'installation automatique de l'outil de gestion de configuration Puppet dans chaque machine sont faits à travers un script Shell par le gestionnaire du nuage. De même, les configurations des adresses IP, des noms du domaine, des hostnames, des utilisateurs ainsi que les droits d'exécution et la signature des certificats des agents par le maître sont ajustés automatiquement par le script d'approvisionnement.

L'installation des agents du Puppet se fait d'une façon automatique au moment du premier démarrage de chaque machine virtuelle approvisionnée par le Cloud Manager. Ce dernier utilise un script d'installation automatique avec un fichier de réponse préconfiguré pour éviter l'interaction avec l'utilisateur.

Tel qu'illustré à la figure 5.1, le contrôleur du nuage est le point central du processus de déploiement. Il orchestre en parallèle et en temps réel tous les composants du système OpenICRA et il gère toutes les principales tâches de déploiement. En effet, il contacte le Cloud Manager pour approvisionner les ressources de calcul et de stockage nécessaires. Ce dernier installe et exécute séquentiellement et automatiquement l'agent Puppet dans chaque machine virtuelle créée. Au moment du démarrage de la machine virtuelle, chaque agent Puppet contacte le contrôleur (Puppet master) pour télécharger la configuration décrite dans la classe « opensaf » et l'appliquer localement. La classe « opensaf » est développée de telle sorte qu'il soit capable de déployer OpenSAF et de résoudre automatiquement tous les problèmes courants sans aucune intervention humaine. Une fois le déploiement est terminé, chaque agent Puppet envoie un rapport au contrôleur.

5.2.2.3 Monitorage et gestion du déploiement

Quand les nœuds de données (agents Puppet) s'exécutent pour chercher de nouvelles configurations ou mises à jour à partir du contrôleur, ils renvoient les données d'inventaire et un rapport de leur exécution. Ces rapports et statistiques peuvent être visualisés par le

développeur dans la page web de chaque nœud à travers l'interface web console. La figure 5.2 illustre l'état des nœuds du cluster d'OpenSAF dans le tableau de bord du système.

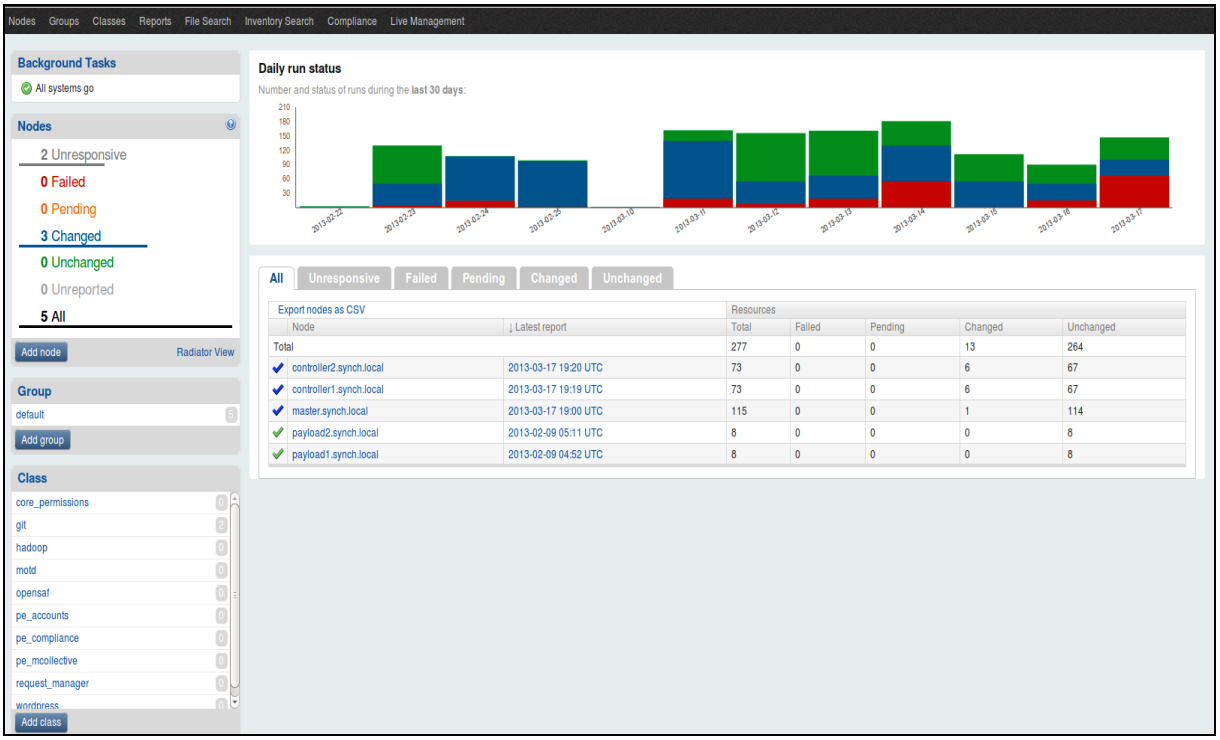

Figure 5.2 État des nœuds de cluster dans le tableau de bord du système

La page web de chaque nœud présente aussi la courbe de temps d'exécution que le nœud a pris tel qu'illustré à la figure 5.3.

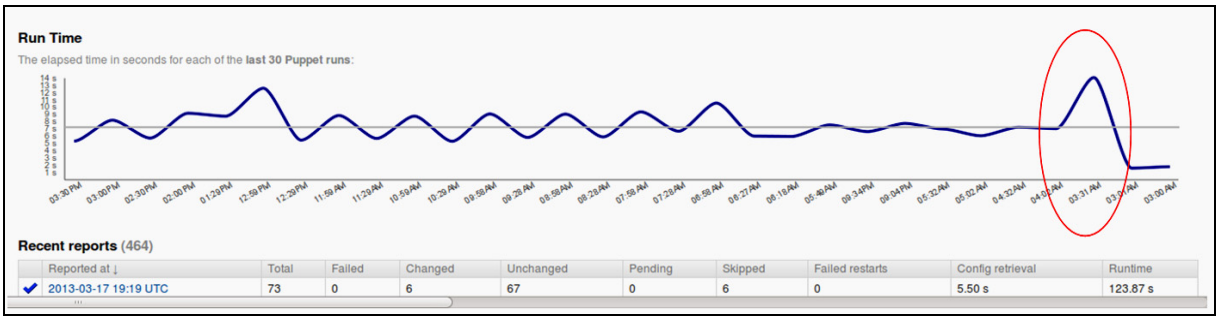

Figure 5.3 Temps d'exécution du démon Puppet dans le nœud du contrôleur

La courbe du temps d'exécution montre combien de temps chacune des 30 dernières exécutions du Puppet a pris pour terminer la configuration ou le déploiement souhaité. Une exécution plus longue signifie généralement qu'il y a eu des modifications ont été apportées, mais pourrait également indiquer que la charge du serveur était lourde ou quelque autre circonstance. Dans notre cas, la dernière exécution, la plus longue, est celle qui correspond au déploiement d'OpenSAF. Ceci est dû au fait qu'il n'y a pas des configurations à faire pour les autres exécutions.

Le développeur peut utiliser les services de monitorage à partir de l'interface web du contrôleur pour vérifier l'état de l'exécution du processus de déploiement. Ainsi, il peut utiliser les tableaux de fichiers journaux, les tableaux des métriques et ceux des événements qui sauvegardent et récapitulent toutes les évènements enregistrés pendant le déploiement d'OpenSAF de chaque nœud au même endroit.

5.2.3 Résultats

Dans cette section, nous présentons les résultats du processus de déploiement automatique d'OpenSAF dans un cluster de nœuds.

La contribution principale de ce cas d'étude est de soulager les utilisateurs de la lecture des guides d'installations complexes et de la résolution manuelle des problèmes en les automatisant afin d'éviter les tâches répétitives à faire sur chaque serveur. Selon un récent sondage ayant interrogé environ 95 utilisateurs de plusieurs compagnies telles que Oracle, Connectem, Ericsson, etc., les utilisateurs d'OpenSAF souffrent de la complexité et de la longue durée du déploiement manuel d'OpenSAF (Synchromedia, 2013). Ce sondage a permis de constater que la durée moyenne nécessaire pour déployer manuellement OpenSAF dans un cluster est environ 138.95 minutes. La figure 5.4 illustre les résultats de ce sondage et montre que 63.16% des répondants dépassent la durée moyenne du déploiement manuel de l'ensemble des utilisateurs.

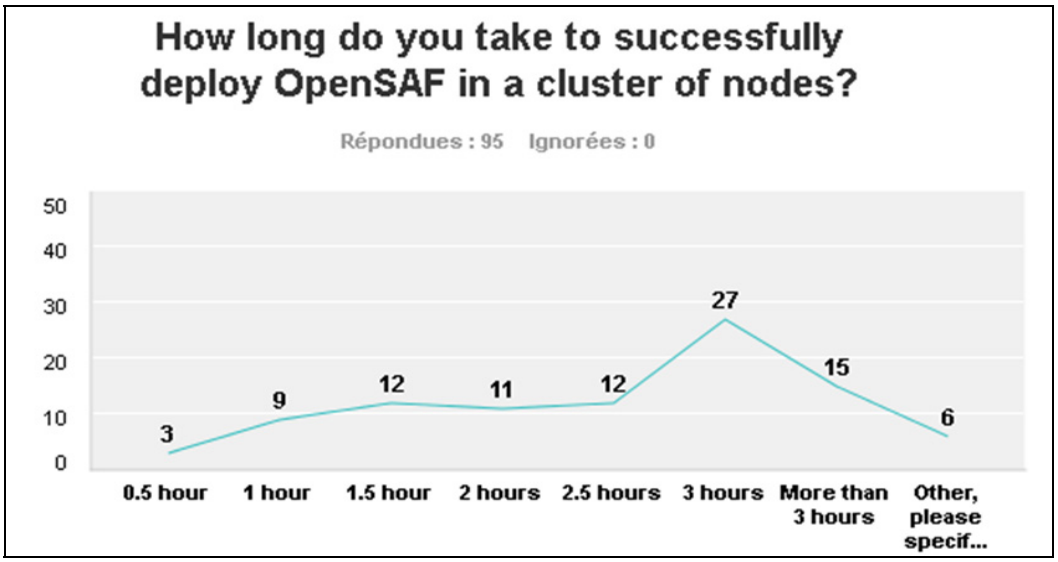

Figure 5.4 Résultat du sondage de déploiement manuel d'OpenSAF

Avec notre modèle proposé, nous avons largement réduit le temps nécessaire de déploiement d'OpenSAF en automatisant les principales tâches de configuration, d'installation, de mises à jour et de résolutions des problèmes courants. La figure 5.5 illustre le résultat du processus de déploiement, la durée et l'état de démarrage d'OpenSAF dans le contrôleur.

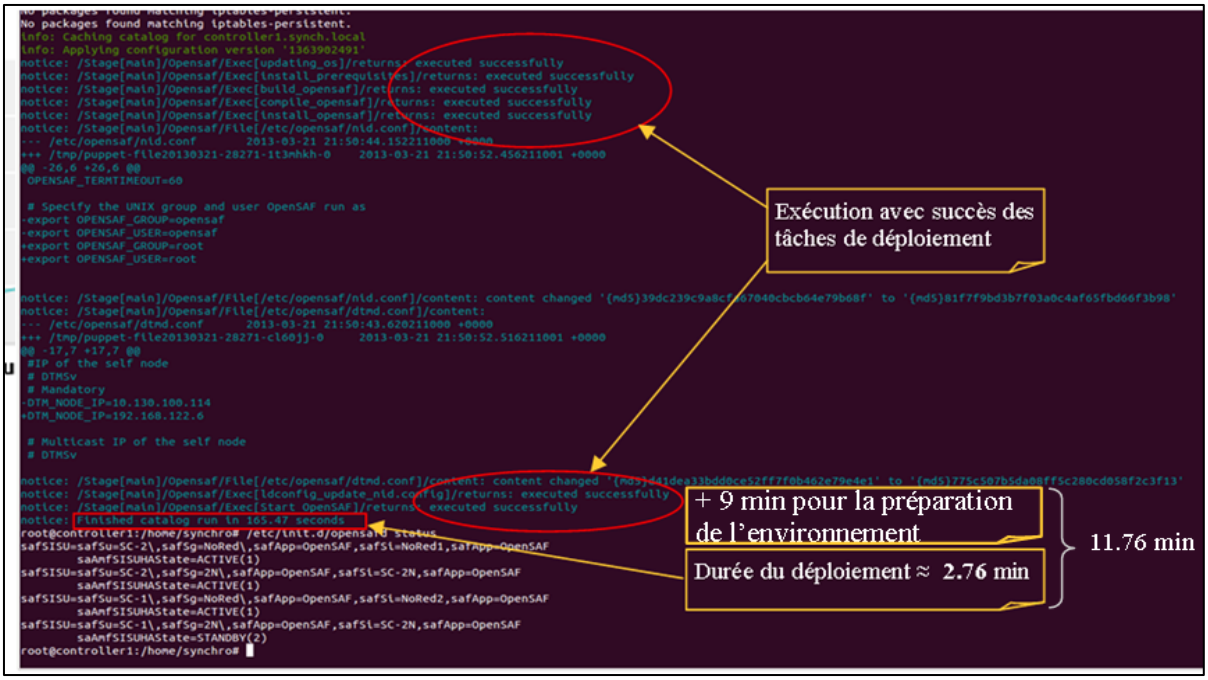

Figure 5.5 Déploiement réussi du contrôleur d'OpenSAF en moins de 166 secondes

L'approche d'automatisation de déploiement d'OpenSAF a produit des résultats quantifiables où l'amélioration de l'efficacité atteint 91.51%. Ces résultats sont exposés dans le tableau 5.2 qui récapitule la comparaison de la durée de déploiement avant et après l'automatisation ainsi que l'amélioration de l'efficacité.

Tableau 5.2 Durée de déploiement avant et après l'automatisation

Tâche	**Avant l'automatisation**	**Après l'automatisation**	**Amélioration de l'efficacité**
Déploiement d'OpenSAF	≈ 2.31 h	≈ 11.76 min[2]	91,51 %

En guise de conclusion à ce cas d'étude, nous remarquons que OpenICRA a permis de réduire de 91.59% l'effort et le temps de déploiement d'OpenSAF et a assuré un déploiement automatisé sans aucune intervention humaine après le déclenchement du processus. Ceci permet de réduire les coûts et d'accélérer efficacement la mise en œuvre de l'intergiciel OpenSAF dans le nuage.

5.3 2ème cas d'étude : Migration automatique du système ICRA vers EC2

5.3.1 Présentation du cas d'étude

ICRA (Internet based Collaboration and Resource Application) est une application de travail collaboratif basée sur le web développée par le groupe de recherche de Synchromedia. Elle permet aux utilisateurs d'établir des réunions virtuelles et des vidéoconférences à partir des endroits éloignés à travers le monde pour faciliter leurs travaux de recherche et d'équipe. En outre, l'application est équipée des technologies de télécommunications interactives de pointe, permettant aux utilisateurs d'avoir des discussions productives et de partager

[2] La durée de déploiement après l'automatisation est égale à la durée de déploiement automatique d'OpenSAF plus la durée de la préparation automatique de l'environnement (machines virtuelles, configuration réseau, etc.) à l'aide d'OpenICRA.

facilement des documents numériques, des données et d'annotations en temps réel (Cheriet, 2012).

ICRA et Asterisk (Sun et al., 2013) sont parfois considérées comme des applications concurrentes pour divers services, alors qu'en réalité, Asterisk peut être complémentaire à l'application ICRA. L'intégration et le déploiement des deux applications ensemble dans un environnement de nuage créent essentiellement un système de travail collaboratif plus évolutif et plus performant.

Le choix d'un PBX IP libre n'est à la base pas compliqué puisqu'aujourd'hui l'acteur majeur des PBX IP libres est le logiciel Asterisk. Le but de ce cas d'étude n'étant pas d'innover sur la technologie des PBX IP. Nous avons tout naturellement décidé d'utiliser celui-ci étant donné qu'il est considéré l'un des PBX gratuits les plus répandus au monde. Il est aussi un des éléments clés du service VoIP et il devra permettre la gestion des appels voix vers l'extérieur.

Cette intégration permet d'offrir aux utilisateurs une opportunité de tester et d'être familiarisés avec la technologie de la voix sur IP dans un environnement de nuage et de travail collaboratif, telle que l'annotation en temps réel, le partage de documents et de présentations, la messagerie instantanée et le web-conférence. Elle offre aussi la possibilité d'utiliser facilement et rapidement des services VaaS (Voip as a Service) et CaaS (Collaboration as a Service) avec un coût réduit et sans logiciels requis côté client.

Cependant, l'une des complexités associées à l'intégration d'Asterisk avec ICRA est reliée au grand nombre des composants à configurer et à migrer vers le nuage et le manque d'outils cohérents de gestion. En effet, ces deux applications utilisent plusieurs services différents et nécessitent une reconfiguration difficile lors de la migration pour qu'elles soient compatibles avec l'environnement d'exécution cible.

Ce cas d'étude a donc pour objectif principal d'intégrer l'autocommutateur téléphonique Asterisk avec l'application de travail collaboratif ICRA. Le tout devant être déployé automatiquement dans l'environnement Cloud EC2 d'Amazon. Ainsi, nous allons utiliser notre modèle proposé OpenICRA pour automatiser l'intégration et le déploiement des deux applications dans le nuage. Ceci permet de simplifier le processus de migration et d'assurer l'évolutivité, le contrôle et la gouvernance de tous les éléments du système.

5.3.2 Processus de migration du système de travail collaboratif vers le nuage EC2

Conformément au fonctionnement du modèle proposé décrit à la section 3.6, le processus d'intégration et de migration des deux applications, ICRA et Asterisk, peut aussi être accompli en trois étapes simples. La figure 5.6 illustre ce processus d'automatisation de la migration du système de travail collaboratif.

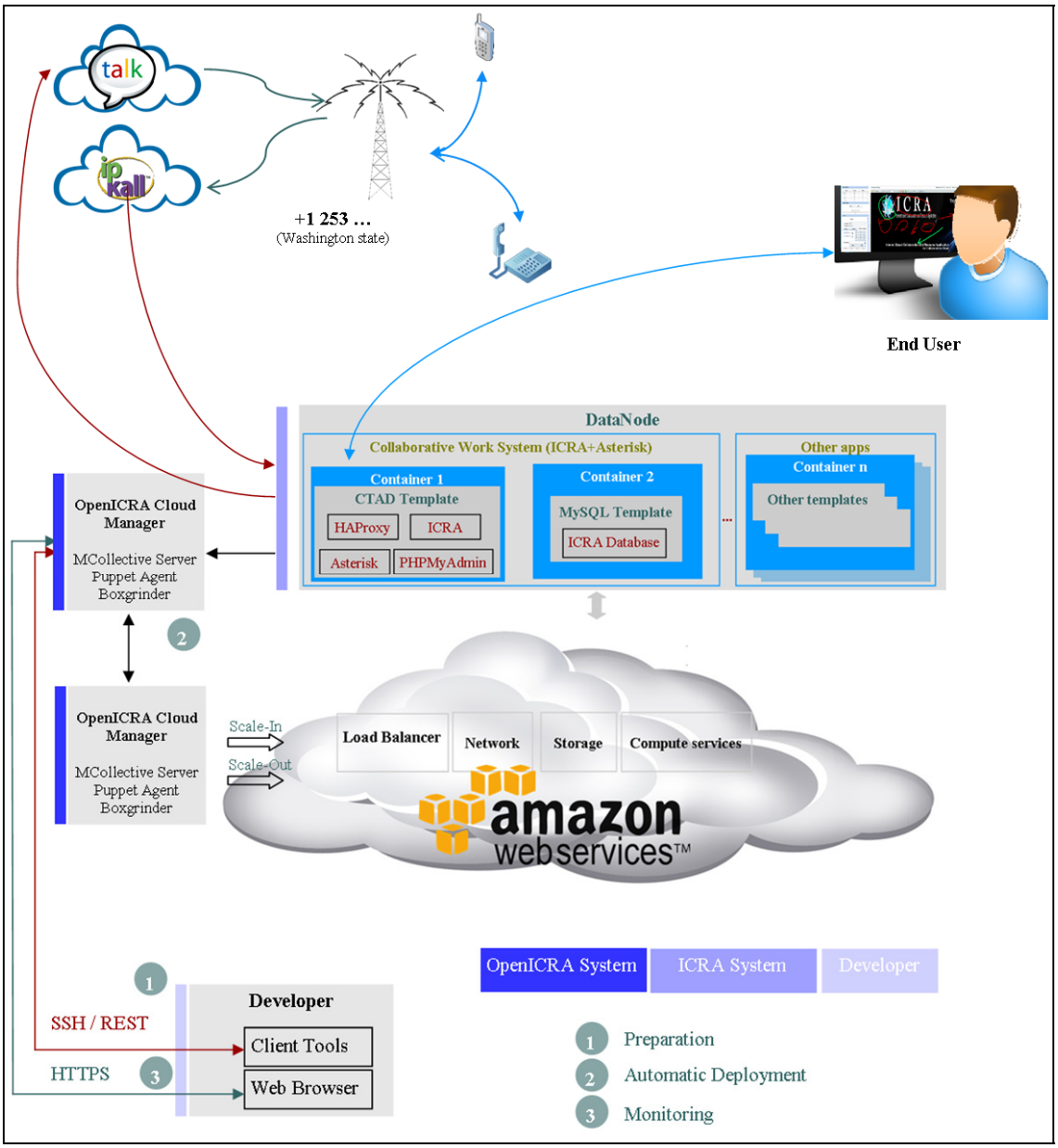

Figure 5.6 Architecture globale du système collaboratif ICRA

5.3.2.1 Préparation de la migration

Cette étape consiste à comprendre l'architecture et le fonctionnement des différents services du système de travail collaboratif à migrer vers le nuage pour assurer un déploiement réussi.

Dans notre cas, l'objectif est d'intégrer le serveur VoIP Asterisk avec l'application de travail collaboratif ICRA pour permettre aux utilisateurs de pouvoir effectuer des appels entrants et sortants depuis l'interface web cliente centralisée de l'application ICRA vers tous les réseaux téléphoniques tels que les réseaux IP, PSTN, cellulaire, etc. Le tout devant être déployé sur le nuage. Pour cela, nous avons déterminé qu'Amazon est le fournisseur le plus approprié en raison de sa flexibilité et de son élasticité. Il offre des machines virtuelles avec plein de services hautement disponibles qui utilisent une technologie de pointe et un accès sécurisé aux ressources via des clés RSA, ainsi que des services de monitorage tels que CloudWatch. Toutefois, il est important de mentionner que le processus de migration du système de travail collaboratif vers le Cloud ne se limite pas seulement à l'infrastructure d'Amazon, mais il est générique et peut être implémenté dans des environnements de nuage hétérogènes tels qu'OpenGSN, GoGrid, Rackspace, etc.

Pour l'interconnexion de l'autocommutateur Asterisk avec les réseaux PSTN et cellulaire, nous recherchons donc des opérateurs nous permettant de recevoir et d'émettre des appels, le tout gratuitement. La solution a été de trouver un fournisseur regroupant les deux caractéristiques. Aucun n'a répondu à nos attentes. Nous avons décidé de répartir les tâches des opérateurs. Pour interconnecter le réseau IP avec le PSTN, nous cherchons un opérateur nous fournissant un numéro de téléphone. Nous avons facilement déterminé quel serait notre fournisseur de ligne. IPKall (2013) répond à nos besoins, il fournit un numéro de téléphone des États-Unis (état de Washington) gratuit, mais fonctionnant au Canada. Pour les appels sortants, notre attention portera donc sur le fournisseur Google Voice puisqu'il nous offre la possibilité d'appeler les numéros des États-Unis et du Canada gratuitement. Il nous faudra donc utiliser IPKall pour les appels entrants et Google Voice pour les appels externes (sortants).

En ce qui concerne l'architecture de l'application ICRA, elle se base principalement sur des composants libres à savoir MySQL, SWFTools, ImageMagick, OpenOffice.org, Red5, middleware java (jar), une interface web cliente, etc. Tous ces composants s'installent en

ligne des commandes sur tout système Linux. Donc, leur installation peut être automatisée par les hameçons de déploiement d'OpenICRA.

L'interface web cliente d'ICRA est le point d'interconnexion de tous les composants du système. Elle se connecte au serveur de base de données et au Middleware ICRA à travers les web services et le serveur de queue ActiveMQ. Également, elle se connecte au serveur de streaming via le fichier de configuration « conf.xml » pour assurer les services de la vidéoconférence, la messagerie instantanée, l'annotation en temps réel, etc.

Mizu de Mizutech (2013) étant un webphone, nous l'avons intégré à l'interface web cliente d'ICRA. Le webphone n'a aucun impact sur le portail ICRA, il s'agit de deux applications complètement indépendantes, mais réunies sur la même interface web. Son intégration est très bien déroulée, celui-ci fonctionne au sein de l'application ICRA en tant qu'interface de connexion entre le portail ICRA et les réseaux de téléphonie via Asterisk. Elle permet de fournir aux utilisateurs un accès direct au serveur VoIP pour effectuer des conversations téléphoniques internationales vers et en provenance du réseau PSTN, ainsi des messages SMS et des conversations entre des comptes SIP. Cette intégration permettra aussi de placer le module VoIP dans une interface centralisée et d'avoir un service de collaboration sur le Cloud combiné avec différents types des technologies avancées.
Pour automatiser l'intégration des différents services et le déploiement du système de travail collaboratif, nous avons développé un QuickStart qui décrit toutes les exigences et les instructions nécessaires à l'automatisation de la migration du système collaboratif vers EC2. Ce hook « hameçon » sera utilisé à l'étape suivante « déploiement automatique » par le script de déploiement automatique pour déclencher l'intégration et le déploiement des applications ICRA et Asterisk.

En ce qui concerne la mise en place d'Asterisk, elle est assez simple et se fait en 3 étapes :

- téléchargement du paquet
- installation du paquet
- configuration du logiciel

L'étape de configuration étant la plus délicate. Dans un premier temps nous utiliserons des clients SIP (Session Iinitiation Protocol). Pour notre cas, nous configurerons plusieurs comptes SIP dans le fichier de configuration sip.conf dont un est pour la réception des appels provenant du réseau PSTN.

Après avoir configuré le fichier sip.conf, il faudra permettre aux utilisateurs de communiquer entre eux. Pour cela, nous utiliserons le fichier extensions.conf. Ce fichier contient les informations de communication entre les différents clients SIP ainsi que le dial plan nécessaire pour acheminer les appels internationaux.

Gtalk est un service de Google Apps (Google, 2013). Son utilisation nous permet de passer des appels gratuitement vers le Canada et les États-Unis ainsi que d'envoyer des SMS à l'international. Pour cela nous passerons par Internet qui transportera nos flux jusqu'à notre service Asterisk sur Amazon. La partie de communication entre Gtalk et Asterisk est l'une des parties les plus intéressantes, mais aussi complexes. Asterisk sera configuré pour transmettre la communication au service Gtalk. Ce dernier nous permettra de passer des appels vers les réseaux PSTN. Pour effectuer ceci, il faudra configurer les trois fichiers de l'interfaçage entre Asterisk et Gtalk qui sont :

- jabber.conf
- gtalk.conf
- extensions.conf

Le premier permet de nous authentifier sur le serveur de Google Talk. Le second est pour autoriser toute personne connectée à Asterisk à téléphoner vers l'extérieur via Gtalk. Alors que le troisième permet la gestion des appels et détermine les chemins utilisés pour effectuer des appels sortants.

IPKall (2013) est un service qui agit comme une passerelle (un proxy) de redirection des appels de PSTN ou de réseau cellulaire vers le réseau VoIP. Les principaux codecs supportés par IPKall sont G.711A-law, GSM et G.729 A. Ce service nous permettra d'avoir

gratuitement un numéro de téléphone de l'état de Washington (+253) afin de pouvoir contacter notre système collaboratif hébergé sur le Cloud. Il utilise le protocole SIP pour dialoguer. Nous avons obtenu le numéro suivant +1 253 242 2225. Il nous a fallu configurer dans un premier temps les informations SIP demandées par IPKall. L'autre partie de configuration se passe sur le serveur Asterisk. Les fichiers de configuration à modifier sont les suivants :

- sip.conf
- extensions.conf

Le premier servira à créer un compte SIP correspondant à IPKAll. Tandis que le second sert à rediriger les flux arrivant d'IPKall vers un compte SIP.

Donc, selon la configuration conçue par le développeur, le modèle générique OpenICRA peut migrer le système collaboratif automatiquement vers EC2 et atteindre l'état de déploiement désiré à partir d'une seule commande via l'outil en ligne de commande « synchro ».

5.3.2.2 Migration automatique vers EC2

L'architecture d'OpenICRA permet au développeur de se connecter directement au contrôleur en s'authentifiant via l'outil en ligne de commande « synchro » et aux nœuds de données via SSH. Le contrôleur est la seule entité capable de communiquer directement avec le Cloud Manager du PaaS. Même si on connaît son adresse IP et les paramètres de connexions (user/password) de la machine hébergent le Cloud manager, on ne peut pas se connecter, car seul le contrôleur a ce droit. Ceci est configuré avec une politique de sécurité stricte pour empêcher l'approvisionnement aléatoire des ressources.

Les outils d'approvisionnement utilisés sont Boxgrinder (JBoss Community, 2013) et SELniux. Le premier est pour créer des instances de machines virtuelles (nœuds de données)

en local et sur le nuage d'Amazon, alors que le deuxième est pour créer des conteneurs des applications dans les datanodes.

Le développeur peut donc déployer ses applications à partir de sa machine locale en utilisant l'outil en ligne de commande d'OpenICRA ou parfois via SSH. Dans notre cas, le développeur doit utiliser l'outil « synchro » pour déclencher le déploiement. Pour cela, il suffit de lancer le script de migration automatique du système collaboratif vers EC2. Une fois le déploiement déclenché, le PaaS OpenICRA intervient à travers ses composants Cloud Controller et Cloud Manager pour assurer le bon déroulement du déploiement.

Dans le script de migration, nous avons utilisé la commande suivante « *synchro app create -s icra ctad mysql-5.1 phpmyadmin-3.4 -I rgadhgadhi@synchromedia.ca -p qwerty* » qui permet de créer un template évolutif et personnalisable de type CTAD avec les applications MySQL et PHPMyAmin pour la création de la base de données. Ce type de template CTAD permet au développeur d'installer toute application compatible avec le système Linux. Pour notre cas d'étude, ce template est créé dans un conteneur indépendant pour héberger les applications ICRA, Asterisk, PHPMyAdmin et l'équilibreur de charge HAProxy qui assure la mise à l'échelle automatique. La base de données MySQL est hébergée dans un autre conteneur. Ceci permet d'assurer l'évolutivité du système collaboratif en cas d'augmentation ou de diminution de la demande. Les options « -I » et « -p » sont utilisées pour l'authentification du développeur par le contrôleur de nuage.

Les deux principaux composants du système collaboratif, ICRA et Asterisk, sont reliés entre eux par deux processus en front end et en back end. Le processus de front end est le serveur web qui héberge en même temps les fichiers web du portail et ceux du webphone. Il permet d'assurer l'interconnexion des deux applications déployées via un web service. Cependant, le processus back end de liaison des deux applications est assuré par le service d'équilibrage de charge et celui de mise à l'échelle automatique « auto scaling » offerts par HAProxy. En effet, auto scaling est un service qui permet de gérer la gestion de démarrage et de terminaison des instances des applications en fonction de la demande. La maintenance d'un

certain nombre d'instances d'application en exécution simultanément est l'une des fonctions les plus importantes de l'auto scaling. Nous avons donc adopté cette méthode pour garder le même taux de réponse quel que soit le volume du trafic reçu par le système collaboratif. Ceci est accompli en raison de l'élasticité du nuage et du service de l'équilibrage de charge automatique permettant de livrer le service de collaboration sur le Cloud à l'utilisateur final, avec de meilleures performances en approvisionnant d'une façon dynamique de nouvelles ressources en fonction de la demande.

En outre, l'hameçon ou le hook de déploiement d'ICRA et d'Asterisk installe d'une façon automatisée les prérequis et les services du système, configure l'application comme souhaité par le développeur et résout automatiquement les problèmes courants.

Une fois le déploiement du système collaboratif terminé, le hook de déploiement renvoie un rapport au Cloud Controller qui sera disponible pour le développeur via l'interface web de l'outil de surveillance.

5.3.2.3 Monitorage de la migration

Pendant cette étape, le développeur peut utiliser les outils de monitorage fourni par la plateforme PaaS, OpenICRA, pour s'assurer du bon déroulement de déploiement. En fait, avec les outils de monitorage il peut facilement détecter s'il y a des erreurs ont eu lieu ou vérifier si le déploiement est réussi.

La figure 5.7 illustre l'état global des ressources du système collaboratif après leur déploiement sur EC2 et montre que les deux instances sont saines avec une variation d'utilisation du CPU entre 0.6% et 8.59%.

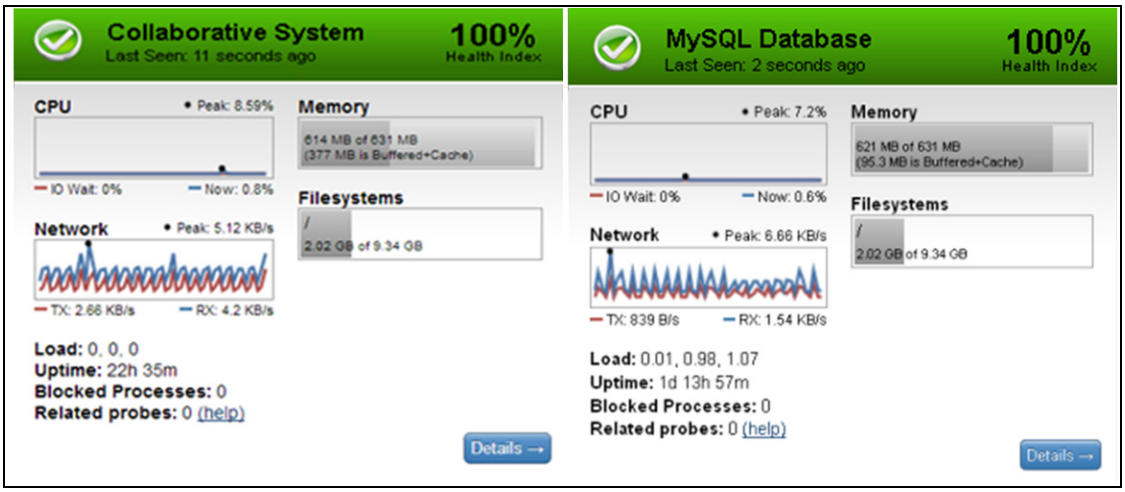

Figure 5.7 État global des conteneurs du système

Pour un déploiement sur Amazon EC2, OpenICRA offre aux développeurs la possibilité d'utiliser à la fois le service de monitorage CloudWatch d'EC2 (Amazon, 2013) et RevealCloud de CopperEgg (2013). Ces derniers permettent de fournir en temps réel des informations détaillées sur l'état du processus de la migration et des services du système collaboratif. Ainsi, le développeur peut utiliser les services de ces outils à partir de l'interface web du contrôleur pour vérifier avec plus de détail l'état de l'exécution du processus de la migration en consultant les tableaux journaux des différents services.

5.3.3 Évaluation des performances globales du système de travail collaboratif

Pour évaluer la stabilité et l'évolutivité du système collaboratif déployé par OpenICRA sur Amazon EC2, nous allons présenter en premier lieu une comparaison quantitative des codecs G.711 et G.729 du module VoIP intégré et un test de résistance du système global en second lieu.

5.3.3.1 Comparaison des codecs et passage du G.711 à G.729

Dans cette section, nous comparons les codecs G.711 et G.729 pour déterminer lequel est optimal pour le système de travail collaboratif. Les tests ont été réalisés sous WireShark et son module d'analyse des flux RTP, une des machines clientes capturait le trafic RTP. Les deux captures ont été réalisées à 10 minutes d'intervalle, on peut donc affirmer que le trafic et la saturation des différents liens intervenant entre les deux clients SIP étaient constants.

L'analyse montre un excellent gain au niveau de la diminution de la perte de paquets puisque que l'on passe de 8-9% de perte pour le G.711 à 0% avec le G.729 tel qu'illustré à la figure 5.8. Pour rappel G.711 utilise une grande bande passante (56- 64 Kbit/s) par rapport à G.729 optimisé pour les réseaux WAN (8kbit/s).

Pour ce qui est de la latence et du jitter, il n'y a pas d'évolution notable. Notre architecture dépend de plusieurs fournisseurs et est difficilement optimisable à ce niveau. On peut quand même noter les performances correctes, généralement en dessous des 100ms avec une moyenne autour de 20ms (valeur mesurée non présente dans ce schéma).

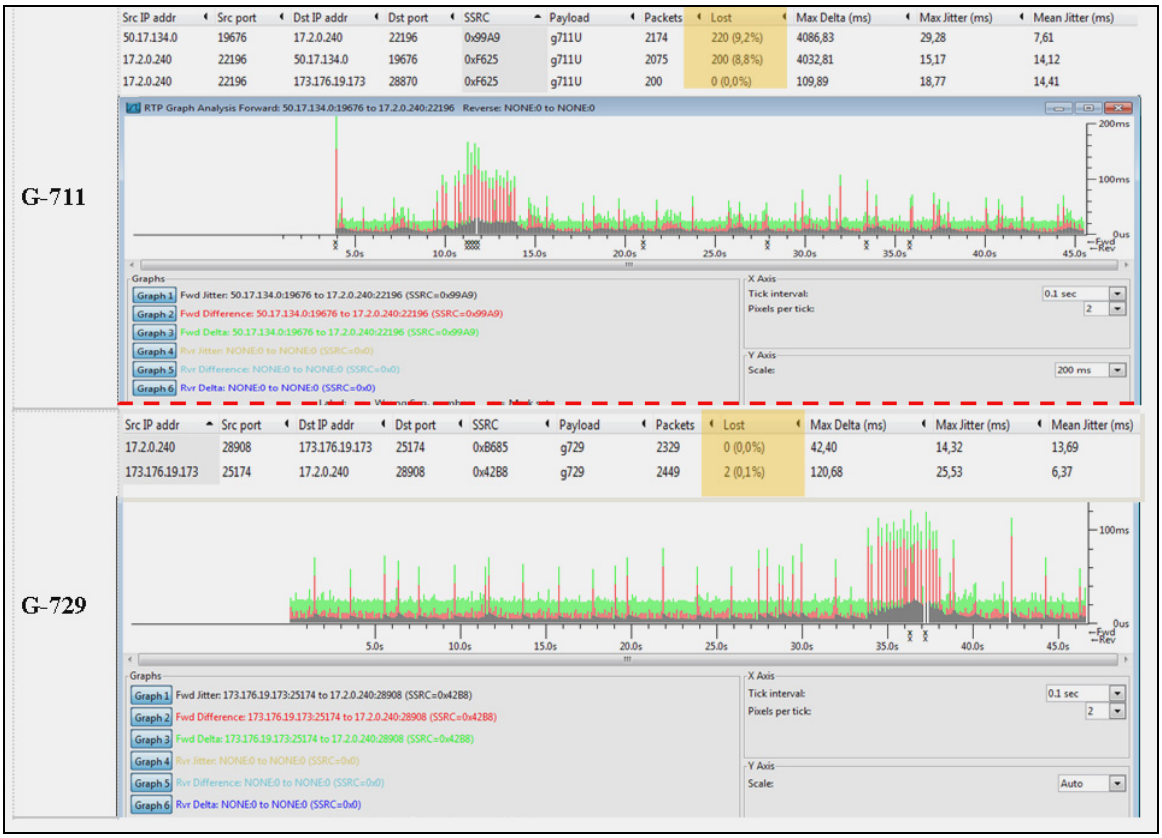

Figure 5.8 Test des performances des codecs G.711 et G.729
Tirée de Gadhgadhi, Allmendinger et Prost (2012)

5.3.3.2 Test de l'évolutivité du système collaboratif

Pour évaluer les capacités de la mise à l'échelle automatique du modèle OpenICRA ainsi que l'évolutivité et la stabilité du système collaboratif lorsque la demande augmente avec le nombre des utilisateurs, nous avons réalisé un test de résistance afin de mesurer sous des conditions de fonctionnement sévères les performances globales des conteneurs hébergés dans un nœud de données d'OpenICRA sur Amazon.

En outre, nous présentons dans cette section les étapes suivies pour réaliser ce test de résistance et les différents résultats obtenus.

En fait, un bon test de résistance pour le système de travail collaboratif d'ICRA consiste à générer une grande charge avec plusieurs utilisateurs. Pour cela, nous avons choisi de réaliser ce test au sein du laboratoire Synchromedia (2012) en utilisant ses équipements et un script Shell personnalisable (Fred, 2013) qui permet de générer un grand nombre de clients afin de surcharger le système collaboratif. Sur chaque poste, nous avons lancé plusieurs clients (entre 10 et 50 clients) connectés à la même conférence pour surcharger le serveur et surveiller son comportement. Ainsi, la mesure de l'utilisation du CPU, du débit et de la mémoire du serveur fut effectuée grâce aux outils de surveillances Cloudwatch (Amazon, 2013) et RevealCloud (CopperEgg, 2013).

Dans ce qui suit, nous présentons la configuration du « Stress Test » et les résultats des taux d'utilisation du CPU, de la mémoire et du débit moyen.

En outre, la configuration des nœuds de données de notre système est la suivante :

- 613 Mo de Mémoire ;
- 2 unités de calcul EC2 (1 cœur virtuel avec 2 unités de calcul EC2 Compute Units) ;
- stockage EBS ;
- plateforme 64-bit Ubuntu 12.04 LTS ;
- 71 Mbit/s de débit maximum (saturation du réseau lorsqu'on dépasse 70% d'utilisation de l'interface) ;
- mise à l'échelle automatique en incrémentant par une instance lorsque le taux d'utilisation du CPU dépasse 90 % (règle configurée dans l'équilibreur de charge).

Concernant les résultats du test de résistance, la figure 5.9 illustre en détail l'évolution du taux d'utilisation du CPU durant toute la période du test. Par ailleurs, il s'est augmenté du 6.5% lorsqu'on a 5 clients passant par 17.49% à 60 clients jusqu'à 86% à 105 clients. Avec 115 clients, le taux d'utilisation du CPU a atteint 92% ce qui est dépassé 90%, l'équilibreur de charge a détecté ce dépassement et il s'est réagi en créant une nouvelle instance de l'application ICRA.

En même temps, le nombre des clients continu à évoluer et le taux d'utilisation du CPU aussi. À 22h46 UTC, le taux du CPU commence à diminuer. Ceci est en raison de la répartition de charge entre les deux instances du système collaboratif après le déclenchement de la mise à l'échelle automatique.

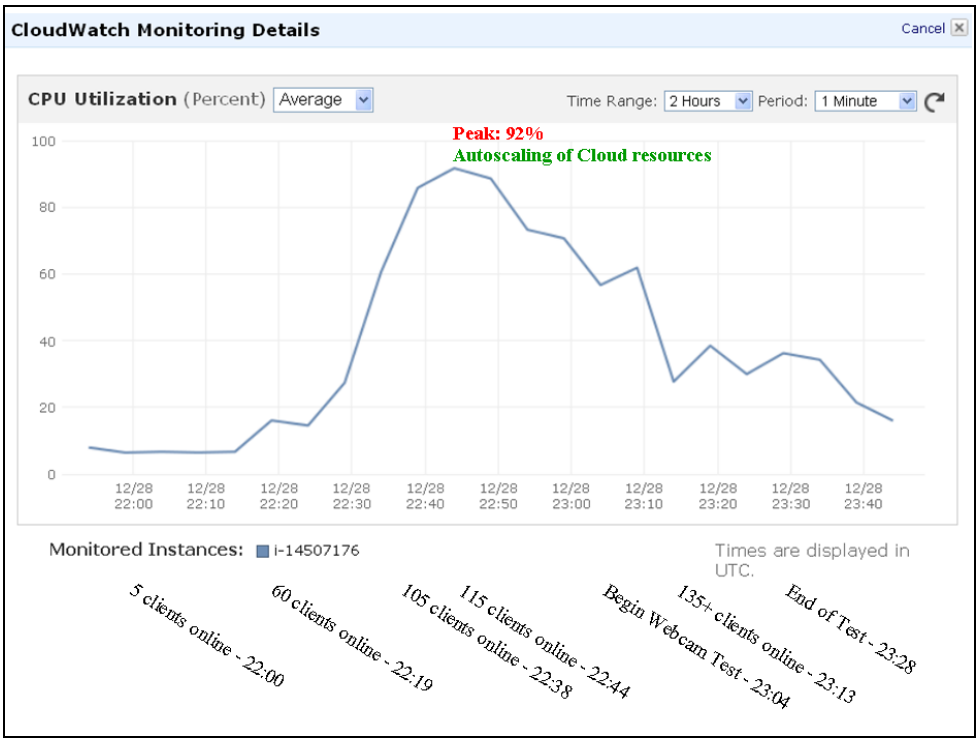

Figure 5.9 Taux d'utilisation du CPU du système ICRA

Au niveau de la performance du réseau, la figure 5.10 illustre l'évolution du débit de transfert de données utilisé auquel les octets sont envoyés et reçus via l'interface réseau du conteneur hébergé dans le nœud de données. Nous remarquons que ce dernier a fait un pic avec une vitesse de téléchargement en amont de 5.76 MB/s alors que celle en aval est autour de 120 KB/s. Par la suite, le débit en amont et en aval a été diminué graduellement en raison de la mise à l'échelle automatique déclenchée par l'équilibreur de charge de la plateforme OpenICRA. En plus du seuil 90% du CPU, l'équilibreur de charge est configuré avec une règle d'incrémentation d'une instance d'application lorsque le taux d'utilisation de l'interface réseau dépasse un seuil de 70% du débit maximum. Comme la valeur du pic du débit utilisé

(5.76MB/s en amont) n'a pas dépassé la valeur du seuil (70% du débit maximum ≈ 6.21MB/s) , le réseau n'est pas considéré en situation de saturation. C'est pour cette raison que nous n'avons pas eu un déclenchement d'une deuxième mise à l'échelle automatique lorsque le débit en amont a atteint sa valeur maximale.

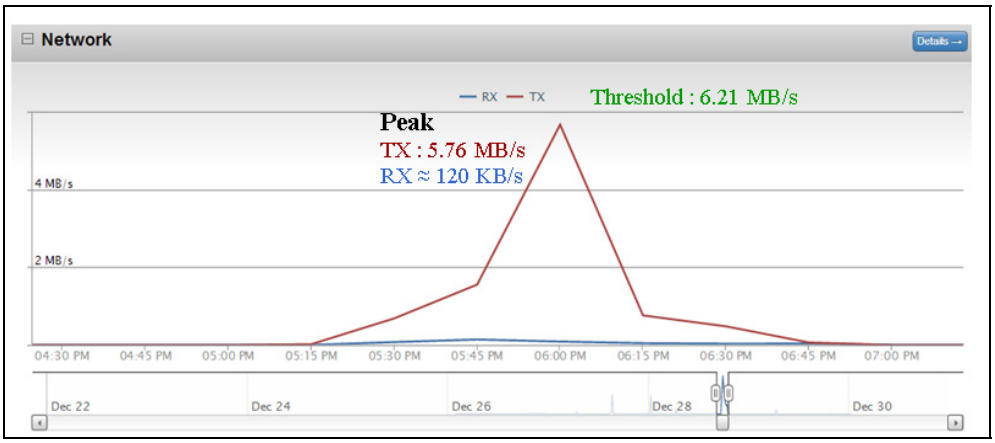

Figure 5.10 Débit moyen utilisé

En outre, le taux d'utilisation de la mémoire est illustré à la figure 5.11. La mémoire active a été au début 31.6% pour le premier groupe de 5 utilisateurs. Puis, lorsqu'on a atteint 60 utilisateurs dans le groupe, le taux d'utilisation de la mémoire active a été augmenté vers 37.8% et on a remarqué de même une augmentation de la mémoire cache du 30.9% à 34.34% pour atteindre une valeur totale de 91.02% en incluant le 18.88% du tampon. Avec 115 utilisateurs, la valeur de la mémoire totale utilisée est autour de 605 Mo ce qui équivaut à 98% (somme de la mémoire active, cache et du tampon). Même si la taille de la mémoire active reste stable, nous constatons une augmentation de la mémoire cache et celle du tampon ce qui concorde parfaitement avec les diminutions de l'utilisation du CPU juste après le déclenchement de la mise à jour automatique par l'équilibreur de charge d'OpenICRA. Pour rappel la mémoire tampon est une zone de la RAM utilisée pour stocker temporairement des données entre deux processus pendant leur déplacement afin de compenser la différence de débit et de vitesse de traitement entre deux processus ne fonctionnant pas au même rythme, tandis que le cache ne contient que des duplications des données afin de diminuer le temps d'accès à ces données.

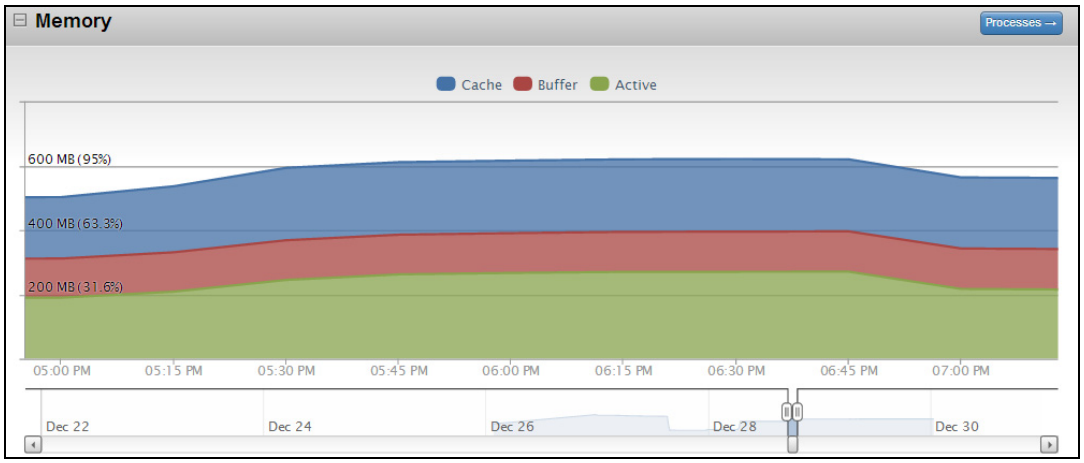

Figure 5.11 Taux d'utilisation de la mémoire du serveur ICRA

Selon ce test, avec 135 utilisateurs connectés en même temps au serveur, le taux de l'utilisation du CPU a atteint 92%, ce qui induit à un approvisionnement automatique de ressources supplémentaires par l'équilibreur de charge pour réagir face à l'augmentation massive de requêtes des clients. Suite à cette réaction, le système collaboratif a rétabli son fonctionnement habituel par le biais de la répartition de la charge, ce qui a contribué à atténuer la surcharge de certains conteneurs du système d'ICRA. Ceci montre bien l'évolutivité et la flexibilité de notre système OpenICRA. Il est important de mentionner qu'on a arrêté d'ajouter plus que 135 utilisateurs pour des raisons de manque de ressources matérielles.

En somme, grâce à ce cas d'étude complexe qui combine plusieurs services décentralisés dépendent de plusieurs fournisseurs tels que Google Talk, IPKall, et AWS, notre modèle générique de déploiement a prouvé ses capacités de déploiement automatique sur le Cloud de tout type d'application avec un processus de trois étapes simples à savoir la préparation, le déploiement et le monitorage. De plus, il a montré son habilité de réagir avec précision en cas d'augmentation de la demande et dépassement des seuils préconfigurés en déclenchant une mise à l'échelle automatique pour rétablir le fonctionnement normal de l'application déployée et équilibrer la charge.

5.4 Conclusion

Dans ce chapitre, nous avons présenté deux cas d'études réels pour tester et valider notre modèle de déploiement. Les résultats prometteurs obtenus démontrent l'efficacité de notre modèle pour déployer différents types d'applications sans apporter aucune modification du code source de l'application. Ils démontrent aussi la supériorité de notre modèle par rapport aux autres modèles de services PaaS discutés dans la section 2.4. En effet, notre modèle OpenICRA permet de déployer d'une façon automatique tout type d'application dans tout environnement de nuage en assurant sa mise à l'échelle, sa portabilité et en évitant l'enfermement propriétaire dans le nuage.

Dans le premier cas d'étude, nous avons détaillé le processus d'automatisation du déploiement de l'intergiciel OpenSAF dans un cluster de nœuds. Cette technique peut aider les utilisateurs à économiser du temps et d'efforts dans le déploiement d'OpenSAF et leur permettre de se concentrer uniquement sur l'intégration et l'optimisation de fonctionnement des services. En outre, le module développé permet de réduire le temps de déploiement en quelques minutes au lieu des heures d'installation et de configuration. Comme prouvée dans la section des résultats (voir section 5.2.3), l'amélioration de l'efficacité de cette méthode du déploiement automatique a atteint 91.51%. De surcroît, le processus d'automatisation s'occupe de l'approvisionnement des ressources nécessaires, de la configuration de l'environnement cible, du déploiement de l'application et de la mise à l'échelle ainsi que le reportage des fichiers journaux (logs) dans une interface web centralisée. Ceci laisse le développeur suivre aisément l'avancement de son déploiement et avec une précision accrue.

Dans le deuxième cas d'étude, nous avons exposé le processus de migration automatisé du système de travail collaboratif ICRA intégré avec l'autocommutateur Asterisk vers l'environnement du nuage EC2 d'Amazon. En utilisant les APIs des technologies avancées que l'application ICRA utilise et en les intégrant automatiquement via le PaaS OpenICRA, la migration a contribué énormément à réduire le temps de déploiement sur le nuage. Ceci permet d'économiser de l'argent et de simplifier le processus de la migration tout en cachant

la complexité de configuration de l'environnement du nuage. De plus, l'automatisation de la migration a assuré la mise à l'échelle automatique en fonction de la demande du système collaboratif ainsi que la portabilité et l'interopérabilité avec toutes les plateformes permettant d'éviter l'enfermement propriétaire.

Parmi les avantages les plus importants de ce deuxième cas d'étude est qu'il est devenu possible grâce à la migration du système de travail collaboratif vers le nuage d'offrir la VoIP en tant que service (VaaS) et la collaboration en tant que service (CaaS) qui sont à nos yeux des technologies d'avenir, le tout gratuit, via un fureteur, et aucun logiciel n'est requis côté client pour accéder aux différents services proposés. De plus, OpenICRA offre une interface utilisateur unique et conviviale qui a pour objectif de centraliser tous les services dans un seul endroit et de faciliter l'accès. Aussi, la téléphonie en tant que service combiné avec le partage des fichiers de différents formats et l'annotation en temps réel est offerte par le système collaboratif ICRA gratuitement vers toute l'Amérique du Nord.

Pour résumer, ces cas d'étude nous permettent de valider notre modèle proposé en déployant d'une façon automatique deux types d'applications différents. La première était sur un cluster de nœuds avec un contrôle total au niveau infrastructure alors la seconde était déployée sur un cluster de conteneurs avec des fonctionnalités complètes et plus de flexibilités au niveau applicatif, le tout sur une nouvelle architecture informatique en plein essor qui est le Cloud Computing. Suite à des mesures de surveillance avec Cloudwatch d'amazon et RevealCloud de copperegg.com, ainsi que d'autres outils de test, des résultats prometteurs ont été enregistrés. Ceci explique que notre système est bel et bien opérationnel, stable et évolutif dans le nuage informatique.

CONCLUSION

Ce mémoire aborde les problèmes majeurs reliés au déploiement des applications dans le nuage informatique, qui relève un défi important pour les développeurs lors de la création d'une nouvelle application ou la migration d'une application existante vers un environnement de nuage informatique. Dans le but d'éviter l'enfermement propriétaire, de réduire la complexité de développement des applications et les délais de mise en œuvre, de simplifier le processus de déploiement des services et d'assurer l'évolutivité automatique des applications dans le nuage, nous avons étudié, conçu et développé, dans ce mémoire, un nouveau modèle générique de déploiement automatisé des applications dans le nuage informatique.

Étant construit sur des technologies open source, notre modèle proposé, que nous avons baptisé OpenICRA, est conçu pour assurer la liberté de choix, y compris la liberté de choisir de se déplacer d'un environnement traditionnel à un autre de nuage et vice-versa. À cette fin, seulement les langages et outils open source standard ont été utilisés dans notre modèle et aucune API, technologie ou ressource propriétaire n'a été intégrée. Cela garantit l'interopérabilité avec toutes les plateformes de nuage et la portabilité des applications à la fois sur et hors de la plate-forme OpenICRA, empêchant ainsi l'enfermement propriétaire sur notre plate-forme. OpenICRA comprend un ensemble complet d'outils de ligne de commande qui fournissent un accès complet à l'interface du développeur. Ces outils sont faciles à utiliser et aussi scriptable pour les interactions automatisées. Notre solution proposée comprend également une riche interface Web console pour la gestion et l'orchestration en parallèle et en temps réel des nœuds d'un cluster, permettant ainsi de réduire la complexité des tâches de développement et de gestion des applications.

Deux cas d'études concrets ont été réalisés pour tester et valider notre modèle de déploiement automatique OpenICRA. Le premier cas d'étude consiste à automatiser le déploiement de l'intergiciel distribué OpenSAF dans un cluster de nœuds avec un contrôle total au niveau infrastructure au sein de l'environnement Cloud du réseau GSN (Synchromedia, 2010), et le

deuxième cas consiste à migrer le système de travail collaboratif ICRA (Cheriet, 2012) vers le nuage EC2 d'Amazon (2012a) avec moins de contrôle au niveau infrastructure, mais plus de flexibilité au niveau applicatif. Les résultats prometteurs obtenus démontrent la supériorité et l'efficacité de notre modèle pour déployer différents types d'applications sans apporter aucune modification du code source de l'application.

En plus, notre modèle générique a montré ses capacités de migrer et de déployer les applications sur un environnement Cloud et d'offrir des services d'automatisation des tâches récurrentes en invoquant en temps réel des actions dans un cluster de nœuds. Pour bénéficier pleinement de l'élasticité du nuage, OpenICRA fournit également une mise à l'échelle automatique de ressources d'une façon horizontale en fonction de l'augmentation ou de la diminution de la charge de l'application éliminant ainsi le besoin des opérations manuelles d'ajout de nouvelles instances des applications. L'intégration de la technologie de l'équilibrage de charge avec la mise à l'échelle automatique contribue aussi à assurer l'évolutivité et l'élasticité de la plateforme OpenICRA et ainsi, la garantie de la QoS.

En outre, contrairement aux autres solutions de nuages informatiques disponibles telles que Cloud Foundry, OpenShift, ou CloudBees, cette plateforme ne se limite pas au déploiement manuel de nouvelles applications dans le nuage. Elle peut être en effet utilisée pour automatiser le déploiement des applications existantes dans le nuage informatique en utilisant des modules personnalisables et extensibles et aisément adaptables aux différentes exigences architecturales des applications.

Dans un environnement de nuage, la manière dont nous stockons les données continue à évoluer d'une façon impressionnante tant au niveau de la vitesse qu'au niveau du volume. Les entreprises sont donc à la recherche de moyens plus efficaces de gérer les données volumineuses telles que les images de disques durs virtuels des VMs, etc. Dans ce mémoire, nous avons proposé un système de gestion de stockage basé sur le système de fichier distribué HDFS pour résoudre le problème du rapport entre la croissance du volume des images des VMs avec la performance des nœuds de calcul. L'intégration du DFS au niveau

IaaS a permis d'offrir un excellent environnement pour les machines virtuelles et les applications où l'évolutivité est un problème à la fois en termes de vitesse de calcul et de stockage. Au niveau l'accessibilité au pool de stockage, l'intégration du HDFS avec le module de montage de volume fuse-hdfs a résolu le problème d'accès aux données (non compatibilité totale du HDFS avec la norme POSIX) par les machines virtuelles et a simplifié la gestion des images volumineuses des disques durs virtuels en rendant le système de stockage plus efficace et plus évolutif.

Ce travail offre donc aux utilisateurs de nuage informatique l'opportunité de tester, de déployer leurs applications et d'être familiarisés avec les services de nuage. Il offre aussi la possibilité de créer automatiquement et rapidement des applications dans le nuage. De surcroît, il contribue non seulement à la familiarisation avec le nuage informatique, qui est considéré comme l'architecture de la prochaine génération de l'informatique d'entreprise, mais également à la protection de l'environnement. En effet, il est compatible avec les ressources et les technologies écoresponsables de GSN, premier projet international qui a pour mission de construire le premier réseau vert à zéro Carbone (Moghaddam, Cheriet et Nguyen, 2011). Ainsi, ce modèle supporte l'intégration des services des fournisseurs VoIP comme Gtalk, IpKall, etc., et le déploiement des applications de télécommunication, en particulier les IPBX d'asterisk et Freeswicth qui permettent l'interconnexion entre les réseaux IP et PSTN, OpenIMS et les webphones tels que Mizu, ozeki, etc. De ce fait, il contribue à la vision d'avenir et à l'amélioration des modèles de migration des applications de télécommunications du projet EcoloTic (Ericsson et al., 2012).

Aussi, le présent travail a donné lieu à deux publications dont le premier article a été publié dans Gadhgadhi, Nguyen et Cheriet (2012), et le deuxième a été soumis à l'International Journal of Cloud Computing and Services Science (IJ-CLOSER).

Dans le futur, plusieurs autres travaux de recherches pourraient être empruntés. Premièrement, étant donné que l'objectif du mémoire est de développer un modèle générique de déploiement automatique des applications dans le nuage informatique et dans le cadre de

la vision du projet EcoloTIC, des efforts significatifs ont y été investis pour supporter la migration automatique des applications de télécommunications telles que les composants d'OpenIMS (HSS, CSCF, etc.), les APIs de WebRTC ainsi que des services VoIP pour interconnecter les réseaux PSTN et IP. Un des objectifs de l'intergiciel OpenGSN est de concevoir un système de gestion de nuage autonome et capable d'assurer la migration des services et des applications vers le nuage. Ainsi, une intégration de notre modèle de déploiement générique avec OpenGSN pourrait contribuer à atteindre cet objectif. Ceci rend ce dernier plus autonome et plus efficace. Également, cette étape permet de stabiliser notre modèle de déploiement en l'utilisant dans des cas d'études concrets tels que la migration de l'implémentation OpenIMS vers un environnement de nuage au sein du projet EcoloTIC. Aussi, une telle intégration ouvrira, indubitablement, la voie vers d'autres besoins qui nécessitent un système de déploiement plus avancé en termes de rapidité de la mise en œuvre des applications dans le nuage et d'efficacité de déploiement.

Finalement, au niveau du stockage, l'architecture par défaut du système de fichiers distribué HDFS permet d'avoir un seul nœud de nom (*namenode*) pour l'ensemble du cluster. Cette décision architecturale faite du HDFS un système plus simple à mettre en œuvre, mais elle entraine certaines limitations telles que l'impossibilité de la mise à l'échelle horizontale du nœud de nom, la limite de performance du système de stockage à un seul espace de nom dans le cluster et l'impossibilité d'isoler un nœud de nom à une organisation spécifique pour éviter sa surcharge par une application particulière ou pour séparer les différentes catégories d'applications (par exemple, HBase) à un espace de nom indépendant. Ces problèmes pourraient être traités par la Fédération HDFS en supportant de multiples nœuds de noms dans le même cluster. Ceci permet d'améliorer l'architecture existante de HDFS par une séparation claire des nœuds de noms et du pool de stockage, permettant d'avoir un bloc de stockage plus générique pour sauvegarder de différentes catégories d'applications. Dans la perspective d'obtenir un système de stockage puissant et évolutif et automatiquement déployable qui combine les forces de processus d'automatisation de déploiement des applications dans le nuage informatique et celles d'entreposage fiable dans le nuage des fichiers de données très volumineux et de l'accès à grande vitesse aux données, nous

suggérons de développer un modèle de stockage générique pour les environnements de nuage qui utilise HDFS, GlusterFS, Lustre, etc. comme un système de fichiers distribué et OpenICRA comme un modèle PaaS de déploiement automatique des applications dans le nuage informatique.

LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES

Agarwal, A., Luniya, R., Bhatnagar, M., Gaikwad, M. et Inamdar, V. 2012. « Reviewing the World of Virtualization ». In *Intelligent Systems, Modelling and Simulation (ISMS), 2012 Third International Conference on*. (8-10 Feb. 2012), p. 554-557.

Ahmed, M., Chowdhury, A. S. M. R. et Rafee, M. M. H. 2012. « An Advanced Survey on Cloud Computing and State-of-the-art Research Issues ».

Amazon. 2012a. « Amazon Elastic Compute Cloud (Amazon EC2) ». In *Amazon Web Services*. En ligne. < http://aws.amazon.com/ec2 >. Consulté le 17 novembre 2012.

Amazon. 2012b. « Amazon web service (AWS) ». In *Amazon Global Infrastructure*. En ligne. < http://aws.amazon.com/ >. Consulté le 20 novembre 2012.

Amazon. 2013. « Amazon CloudWatch ». In *AWS Products & Solutions*. En ligne. < http://aws.amazon.com/cloudwatch/ >. Consulté le 13 mai 2013.

Beberg, A. L., Ensign, D. L., Jayachandran, G., Khaliq, S. et Pande, V. S. 2009. « Folding@ home: Lessons from eight years of volunteer distributed computing ». In *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*. p. 1-8. IEEE.

Bergkvist, A., Burnett, D. C., Jennings, C. et Narayanan, A. 2012. « WebRTC 1.0: Real-time communication between browsers ». *Working draft, W3C*.

Borthakur, D. 2008. « HDFS architecture guide ». *Hadoop Apache Project. http://hadoop. apache. org/common/docs/current/hdfs_design. pdf*.

Boursas, L., Carlson, M., Hommel, W., Sibilla, M. et Wold, K. 2008. *Systems and Virtualization Management: Standards and New Technologies*, 18. Springer.

Bouzereau, O. 2010. « Amazon voit le Cloud privé comme une arnaque ». In. (19 Octobre 2010), sous la dir. de ZDNet. < http://www.zdnet.fr/actualites/amazon-voit-le-cloud-prive-comme-une-arnaque-39755491.htm >. Consulté le 7 janvier 2013.

Business Wire. 2011. « OpenSAF Reports Large Gains in Market Traction ». In. (March 15, 2011). En ligne. < http://python.sys-con.com/node/1753543 >. Consulté le 6 mai 2013.

Campbell, S. L., Chancelier, J. P. et Nikoukhah, R. 2006. *Modeling and Simulation in SCILAB*. Springer.

Chaisiri, S., Lee, B. S. et Niyato, D. 2012. « Optimization of resource provisioning cost in cloud computing ». *Services Computing, IEEE Transactions on,* vol. 5, n° 2, p. 164-177.

Chauhan, M. A., et Babar, M. A. 2012. « Towards Process Support for Migrating Applications to Cloud Computing ». In *Cloud and Service Computing (CSC), 2012 International Conference on*. p. 80-87. IEEE.

Cheriet, M. 2012. « ICRA ». In *Synchromedia's Collaborative platform*. En ligne. < http://tokra.synchromedia.ca/icra/ >. Consulté le 20 novembre 2012.

Chine, K. 2010. « Learning math and statistics on the cloud, towards an EC2-based Google Docs-like portal for teaching/learning collaboratively with R and Scilab ». In *Advanced Learning Technologies (ICALT), 2010 IEEE 10th International Conference on*. p. 752-753. IEEE.

Claranet. 2013. « le cloud computing est une vraie révolution dans le milieu informatique ». In *Cloud – Définition et historique - Claranet*. En ligne. < http://www.claranet.fr/cloud.html >. Consulté le 7 janvier 2013.

Clark, C., Fraser, K., Hand, S., Hansen, J. G., Jul, E., Limpach, C., Pratt, I. et Warfield, A. 2005. « Live migration of virtual machines ». In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*. p. 273-286. USENIX Association.

CloudBees. 2013. « CloudBees: The Java PaaS Company ». In. En ligne. < http://www.cloudbees.com/ >. Consulté le 24 mars 2013.

CopperEgg. 2013. « Real-time insight into server performance and services deployed in public and private clouds. ». In *Server Performance – RevealCloud*. En ligne. < http://www.copperegg.com/revealcloud-server-monitoring/ >. Consulté le 13 mai 2013.

CumuLogic. 2013. « Java and PHP PaaS - Platform as a service ». In. En ligne. < http://www.cumulogic.com/cumulogic-software-portfolio/paas/ >. Consulté le 24 mars 2013.

Di Martino, B., Petcu, D., Cossu, R., Goncalves, P., Máhr, T. et Loichate, M. 2011. « Building a mosaic of clouds ». In *Euro-Par 2010 Parallel Processing Workshops*. p. 571-578. Springer.

Dillon, T., Wu, C. et Chang, E. 2010. « Cloud computing: Issues and challenges ». In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*. p. 27-33. IEEE.

Donnette, B., et Hannequin, D. 2007. « LA VIRTUALISATION ». In *Groupe LINAGORA*. (Paris, France).

Dowell, S., Barreto, A., Michael, J. B. et Shing, M. T. 2011. « Cloud to cloud interoperability ». In *System of Systems Engineering (SoSE), 2011 6th International Conference on*. p. 258-263. IEEE.

Ericsson, CGI, FUJITSU, IBM, MIRANDA et DALSA, T. 2012. « A major GreenICT Initiative ». In *Equation project*. En ligne. < http://www.equationict.com/ >. Consulté le 20 novembre 2012.

Favoritemedium.com. 2011. « KVM host with gateway guest using port-forwarding ». In. (NOVEMBER 25, 2011). < http://tech.favoritemedium.com/2011/11/kvm-host-with-gateway-guest-using-port.html >. Consulté le February 15, 2013.

Flexiscale. 2013. « Cloud Software and Cloud Hosting Provider ». In. En ligne. < www.flexiscale.com >. Consulté le 4 janvier 2013.

Fred, D. 2013. « Stress Testing the BigBlueButton Server with Many Clients ». In. < https://github.com/bigbluebutton/bigbluebutton/blob/master/labs/stress-testing/bbb-test >. Consulté le 22 mai 2013.

Gadhgadhi, R., Allmendinger, P.-H. et Prost, P. 2012. « Intégration d'un service de VoIP pour une plateforme de travail collaboratif sur le Cloud Computing ». In *Mobilité et téléphonie IP MGR840*. (ETS, 24/02/2012).

Gadhgadhi, R., Nguyen, K.-K. et Cheriet, M. 2012. « Automated intrusion attack with permanent control: Analysis and countermeasures ». In *Information Science, Signal Processing and their Applications (ISSPA), 2012 11th International Conference on*. p. 1440-1441. IEEE.

Ghemawat, S., Gobioff, H. et Leung, S.-T. 2003. « The Google file system ». In *ACM SIGOPS Operating Systems Review*. Vol. 37, p. 29-43. ACM.

GoGrid. 2013. « Cloud Hosting, CLoud Computing and Hybrid Infrastructure from GoGrid, ». In. En ligne. < http://www.gogrid.com/ >. Consulté le 4 janvier 2013.

Google. 2012. « Google App Engine ». In *Google Developers*. En ligne. < https://developers.google.com/appengine/ >. Consulté le 20 novembre 2012.

Google. 2013. « Google Talk software ». In. En ligne. < http://www.google.com/talk/ >. Consulté le 13 mai 2013.

GSN Middleware Team. 2011. « GSN - Middleware Architecture description & Implementation ». In *Greenstar Network*. (12/31/2011). < http://www.greenstarnetwork.com >.

Guillén, J., Miranda, J., Murillo, J. M. et Canal, C. 2013. « A service-oriented framework for developing cross cloud migratable software ». *Journal of Systems and Software*.

Haynes, T., et Noveck, D. 2013. « Network File System (NFS) version 4 Protocol ». *Network*.

HP. 2012. « Four steps to better application management and deployment ». In *Business white paper*. (April 2012).

IPKall. 2013. « Free Washington state phone number to your Internet phone ». In *Receive CALLS FREE via IP*. En ligne. < http://www.ipkall.com/ >. Consulté le 13 mai 2013.

JBoss Community. 2013. « BoxGrinder is a set of projects that help you grind out appliances for multiple virtualization and Cloud providers ». In. En ligne. < http://www.boxgrinder.org/ >. Consulté le 13 mai 2013.

Jeff, V. 2013. « Ten Cloud Computing Leaders ». In. En ligne. < http://www.datamation.com/netsys/article.php/3884311/Ten-Cloud-Computing-Leaders.htm >. Consulté le 21 mars 2013.

Julien, B., Emeric, A., Jean-charles, D. et Mickaël, V. 2008. « Windows Server 2008 : Hyper-V ». In *Microsoft*. (19-05). < http://www.labo-microsoft.org/articles/Hyper-v/01/ >.

Juve, G., et Deelman, E. 2011. « Automating application deployment in infrastructure clouds ». In *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*. p. 658-665. IEEE.

Khazret, S. 2013. « A Comprehensive List of Cloud Platforms (PaaS) ». In. En ligne. < http://www.dzone.com/news/comprehensive-list-cloud >. Consulté le 13 mars 2013.

Mell, P., et Grance, T. 2011. « The NIST Definition of Cloud Computing ». In *Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg*. (September 2011), sous la dir. de 800-145, NIST Special Publication. < http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf >.

Microsoft. 2012. « Microsoft's Cloud Platform ». In *Windows Azure*. En ligne. < http://www.windowsazure.com >. Consulté le 20 novembre 2012.

Mike, K. 2012. « Adopt emerging specifications for cloud resource control ». In *Red Hat*. En ligne. < https://openshift.redhat.com/community/content/adopt-emerging-specifications-for-cloud-resource-control-occi >. Consulté le 18 février 2013.

Miloushev, V. I., et Nickolov, P. A. 2011. « Aggregated opportunistic lock and aggregated implicit lock management for locking aggregated files in a switched file system ». In. Google Patents.

Miranda, J., Murillo, J. M., Guillén, J. et Canal, C. 2012. « Identifying adaptation needs to avoid the vendor lock-in effect in the deployment of cloud SBAs ». In *Proceedings of the 2nd International Workshop on Adaptive Services for the Future Internet and 6th International Workshop on Web APIs and Service Mashups*. p. 12-19. ACM.

Mizutech. 2013. « WebSIPPhone - VoIP phone software for the browser ». In. En ligne. < http://www.mizu-voip.com/Products/WebSIPPhone.aspx >. Consulté le 13 mai 2013.

Moghaddam, F. F., Cheriet, M. et Nguyen, K. K. 2011. « Low carbon virtual private clouds ». In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*. p. 259-266. IEEE.

Muhammad, T. 2012. « OpenShift 2.0.22 User Guide ». In *Using OpenShift to manage your applications in the cloud*. (USA). Red Hat.

Nelson-Smith, S. 2011. *Test-Driven Infrastructure with Chef*. O'Reilly Media.

Neoflow. 2012. « Migration à chaud de machines virtuelles avec VMware vSphere et le vMotion ». In *Virtualisation VMware*. En ligne. < http://www.neoflow.fr/tutoriels.item.168/migration-%C3%A0-chaud-de-machines-virtuelles-avec-vmware-vsphere-et-le-vmotion.html >. Consulté le 29 novembre 2012.

Nicolas, F., et Julien, V. 2009. « La virtualisation ». In *IUT Charlemagne de NANCY*. licence pro. A.S.R.A.L.L.

North Bridge. 2013. « 2012 Future of Cloud Computing Survey Exposes Hottest Trends in Cloud Adoption ». In *2012 Future of Cloud Computing Survey*. (SAN FRANCISCO, USA). En ligne. < http://northbridge.com/2012-future-cloud-computing-survey-exposes-hottest-trends-cloud-adoption >. Consulté le 28 janvier 2013.

OCCI. 2012. « Open Cloud Computing Interface ». In. En ligne. < http://occi-wg.org/ >. Consulté le 17 février 2013.

OpenSAF team. 2013. « Getting Started with OpenSAF ». In. < http://devel.opensaf.org/wiki/GettingStartedWithOpenSAF >. Consulté le 11 février 2013.

OpenStack. 2013. « Open source software for building private and public clouds ». In *OpenStack Open Source Cloud Computing Software*. En ligne. < http://www.openstack.org/ >. Consulté le 27 janvier 2013.

Patrick, J. 2011. « cloud computing tentative de définition ». In., sous la dir. de Informatique, ABISSA. En ligne. < http://www.abissa.ch/data/fichiers/tec_cloud_computing.pdf >. Consulté le 3 janvier 2013.

Petcu, D. 2011. « Portability and interoperability between clouds: challenges and case study ». *Towards a Service-Based Internet*, p. 62-74.

Rackspace. 2013. « Dedicated Server, Managed Hosting, Web Hosting by Rackspace Hosting ». In. En ligne. < http://www.rackspace.com/ >. Consulté le 4 janvier 2013.

Red5 Team. 2012. « Red5 Media server ». In *The Open Source Media Server*. En ligne. < http://www.red5.org/ >. Consulté le 20 novembre 2012.

Red Hat. 2013. « DEVELOP AND SCALE APPS IN THE CLOUD ». In. En ligne. < https://www.openshift.com/ >. Consulté le 23 mars 2013.

Ressman, D., et Valdés, J. 2000. « Use of CFengine for automated, multi-platform software and patch distribution ». In *Proceedings of the 14th LISA Conference*. p. 207-218.

Ross, I., et Robert, G. 2013. « The R Project for Statistical Computing ». In. < http://www.r-project.org/ >.

Salesforce. 2013. « Social & mobile Application Development Platform ». In. En ligne. < http://www.force.com/ >. Consulté le 4 janvier 2013.

Salih, N. K., et Zang, T. 2012. « Survey and comparison for Open and closed sources in cloud computing ». *arXiv preprint arXiv:1207.5480*.

SAP. 2013. « SAP Business ByDesign ». In. En ligne. < http://www54.sap.com/solutions/tech/cloud/software/business-management-bydesign/overview/index.html >. Consulté le 4 janvier 2013.

Schmuck, F. B., et Haskin, R. L. 2002. « GPFS: A Shared-Disk File System for Large Computing Clusters ». In *FAST*. Vol. 2, p. 19.

Singh, B., et Nain, P. 2012. « Bottleneck Occurrence in Cloud Computing ». In *IJCA Proceedings on National Conference on Advances in Computer Science and Applications (NCACSA 2012)*. Foundation of Computer Science (FCS).

Small Tree. 2013. « Shared Storage ». In. En ligne. < https://www.small-tree.com/shared_storage_a/212.htm >. Consulté le 10 avril 2013.

Sun, L., Mkwawa, I.-H., Jammeh, E. et Ifeachor, E. 2013. « Case Study 1—Building Up a VoIP System Based on Asterisk ». In *Guide to Voice and Video over IP*. p. 193-213. Springer.

Synchromedia. 2010. « GreenStar Network | Building a Zero Carbon Network ». In *GreenStar Network*. En ligne. < http://www.greenstarnetwork.com/ >. Consulté le 20 novembre 2012.

Synchromedia. 2013. « Manual deployment duration of OpenSAF ». In. En ligne. < http://www.surveymonkey.com/s/Y5Q9DSL >. Consulté le 7 mai 2013.

Tian, W., Su, S. et Lu, G. 2010. « A framework for implementing and managing platform as a service in a virtual cloud computing lab ». In *Education Technology and Computer Science (ETCS), 2010 Second International Workshop on*. Vol. 2, p. 273-276. IEEE.

Tran, V., Keung, J., Liu, A. et Fekete, A. 2011. « Application migration to cloud: a taxonomy of critical factors ». In *Proceedings of the 2nd International Workshop on Software Engineering for Cloud Computing*. p. 22-28. ACM.

Turnbull, J., et McCune, J. 2011. *Pro Puppet*. Apress.

Uhlig, R., Neiger, G., Rodgers, D., Santoni, A. L., Martins, F. C. M., Anderson, A. V., Bennett, S. M., Kagi, A., Leung, F. H. et Smith, L. 2005. « Intel virtualization technology ». *Computer,* vol. 38, n° 5, p. 48-56.

VMware. 2007. « Understanding Full Virtualization, Paravirtualization, and Hardware Assist ». In *White paper.* (November 11, 2007).

VMware. 2012a. « Créer un datacenter efficace et flexible à l'aide de VMware vSphere ». In *VMware vSphere*. En ligne. < http://www.vmware.com/ca/fr/products/datacenter-virtualization/vsphere/compute.html >. Consulté le 29 novembre 2012.

VMware. 2012b. « Optimize IT Ressources with virtual technology ». In *Virtualization Basics*. En ligne. < http://www.vmware.com/virtualization/what-is-virtualization.html >. Consulté le 21 novembre 2012.

VMware. 2013. « Deploy and scale applications in seconds on your choice of clouds. ». In. En ligne. < http://www.cloudfoundry.com >. Consulté le 23 mars 2013.

Wauters, R. 2010. « Salesforce. com Buys Heroku For $212 Million In Cash ». *Luettu,* vol. 16, p. 2012.

Youseff, L., Butrico, M. et Da Silva, D. 2008. « Toward a unified ontology of cloud computing ». In *Grid Computing Environments Workshop, 2008. GCE'08*. p. 1-10. IEEE.

Zhang, Q., Cheng, L. et Boutaba, R. 2010. « Cloud computing: state-of-the-art and research challenges ». *Journal of Internet Services and Applications,* vol. 1, n° 1, p. 7-18.

Zhao, T., March, V., Dong, S. et See, S. 2010. « Evaluation of a performance model of Lustre file system ». In *ChinaGrid Conference (ChinaGrid), 2010 Fifth Annual*. p. 191-196. IEEE.