



MEMOIRE

Présenté à

L'École Nationale d'Ingénieurs de Sfax

en vue de l'obtention du

MASTERE

Nouvelles Technologies des Systèmes Informatiques Dédiés

Par

Meriam MAHJOUR

(Ingénieur Informatique)

**Étude et expérimentations du Cloud Computing pour le
monitoring des applications orientées services**

Soutenu le 03 septembre 2011, devant le jury composé de :

M.	Maher BEN JEMAA	Président
M.	Maher KHEMAKHEM	Rapporteur
M.	Mohamed JMAIEL	Encadreur
M.	Riadh BEN HALIMA	Invité
Mlle.	Afef MDHAFFAR	Invitée

A mes parents Zouhaier et Samia
A mes sœurs Wafa et Rania
A mon frère Mohamed
A tous ceux qui comptent pour moi
Je vous aime!

Remerciements

C'est avec un grand plaisir que je réserve cette page en signe de gratitude et de profonde reconnaissance à tous ceux qui ont bien voulu apporter l'assistance nécessaire au bon déroulement de ce travail.

Je veux d'abord remercier mon encadreur M. Mohamed Jmaiel, professeur à l'Ecole Nationale d'Ingénieurs de Sfax, pour la confiance qu'il m'a accordé en acceptant de diriger mes travaux de mastère. Qu'il trouve dans ce travail l'expression de ma profonde gratitude.

Mes vifs remerciements s'adressent également à M. Riadh Ben Halima, assistant à l'Ecole Nationale d'Ingénieurs de Sfax, pour la qualité de son co-encadrement, ses remarques pertinentes et ses encouragements continus. J'apprécie ses grandes qualités morales et son extrême modestie. Qu'il trouve dans ce travail l'expression de mon profond respect et mon infinie reconnaissance.

Je remercie vivement Mlle. Afef Mdhaffar pour son assistance et ses encouragements tout au long du mastère.

Je tiens à remercier M. Maher Ben Jemaa, maître de conférences à l'Ecole Nationale d'Ingénieurs de Sfax pour l'honneur qu'il m'a fait en acceptant de présider le jury de mon mastère.

Je remercie également M. Maher Khemakhem, maître de conférences à l'Institut Supérieur de Gestion de Sousse, d'avoir accepté de juger mes travaux de mastère.

Je n'oublierais pas à remercier les membres de l'unité de recherche *ReDCAD*, pour la bonne ambiance.

Je remercie particulièrement mes parents, mes sœurs et mon frère pour leur patience et leur soutien tout au long de ce mastère. Ainsi je tiens à leur dédier ce modeste travail.

Table des matières

Introduction générale	1
1 Etat de l'art	3
1.1 Introduction	3
1.2 Le Cloud Computing	4
1.2.1 Les types de Cloud Computing	4
1.2.2 Les services de Cloud Computing	5
1.3 La virtualisation	7
1.3.1 Définition	7
1.3.2 Techniques de virtualisation	7
1.3.3 Solutions de virtualisation	11
1.4 Solutions IaaS open source de Cloud Computing	14
1.4.1 Eucalyptus	15
1.4.2 OpenNebula	16
1.4.3 Nimbus	17
1.4.4 Xen Cloud Platform	19
1.4.5 AbiCloud	20
1.4.6 OpenStack	21
1.4.7 Synthèse	23
1.5 Les acteurs du marché du Cloud Computing	27
1.5.1 Amazon Web Services	27
1.5.2 Microsoft Azure	28
1.5.3 Google App Engine	28
1.5.4 Google Apps	29

1.5.5	Force.com	29
1.5.6	OVH	29
1.5.7	Synthèse	29
1.6	Cloud Computing et techniques de virtualisation	31
1.7	Conclusion	33
2	Le Cloud Computing et les services Web	34
2.1	Introduction	34
2.2	Les couches du Cloud Computing	34
2.3	Les services Web	35
2.3.1	Définition	36
2.3.2	Cycle de vie d'un service Web	36
2.3.3	Les concepts associés	36
2.4	Performance des services Web	37
2.5	Techniques de mesure de la QoS	39
2.5.1	Le Timer inséré dans le code	39
2.5.2	Utilisation de l'approche orientée aspect	39
2.5.3	Autres techniques	40
2.6	Architecture de mesure de la qualité de service	40
2.7	Conclusion	42
3	Expérimentations et validation d'AOP4CSM sur le Cloud	43
3.1	Introduction	43
3.2	Mise en place d'AOP4CSM	43
3.2.1	Couche infrastructure	43
3.2.2	Couche plateforme	45
3.2.3	Couche software	46
3.3	Environnement de déploiement	46
3.3.1	MiniCloud d'OVH	46
3.3.2	Configuration et déploiement	47
3.4	Expérimentations d'AOP4CSM	48
3.4.1	Résultats expérimentaux	48

3.4.2	Surcoût d'AOP4CSM	53
3.4.3	Synthèse	55
3.5	Conclusion	55
	Conclusion générale et perspectives	55
	Bibliographie	57
	annexe	61
.1	Etapes d'installation d'Eucalyptus	62
.1.1	Etape 1 : Préparation du front end	62
.1.2	Etape 2 : Préparation du nœud	63
.1.3	Etape 3 : Enregistrement et intégration des composants d'Eucalyptus	67
.1.4	Etape 4 : Utilisation d'Eucalyptus	67

Table des figures

1.1	Cloud Computing	3
1.2	Les différents niveaux des services du Cloud Computing	6
1.3	Isolation	8
1.4	Paravirtualisation	9
1.5	Virtualisation complète	9
1.6	Architecture d'Eucalyptus	16
1.7	Architecture d'OpenNebula	17
1.8	Architecture de Nimbus	18
1.9	Architecture de Xen Cloud Platform	19
1.10	Architecture d'AbiCloud	20
1.11	Architecture d'OpenStack	22
2.1	Architecture générale du Cloud Computing	35
2.2	Cycle de vie d'un service Web	37
2.3	Architecture de mesure de la QoS	38
2.4	Architecture de déploiement d'AOP4CSM	41
3.1	Composants d'Eucalyptus	45
3.2	Architecture de déploiement sur le Cloud	47
3.3	Evolution du temps de réponse	50
3.4	Evolution du temps d'exécution	51
3.5	Evolution du temps de communication	51
3.6	Représentation des requêtes servies en fonction du nombre de clients	52
3.7	Mesure de la disponibilité	53

3.8 Evolution du temps de réponse avec et sans AOP4CSM 54

Liste des tableaux

1.1	Techniques de virtualisation	10
1.2	Solutions de virtualisation	13
1.3	Solutions IaaS open source de Cloud Computing	24
1.4	Acteurs du Cloud Computing	30
1.5	Cloud Computing et techniques de virtualisation utilisées	32
3.1	Configuration matérielle nécessaire pour la mise en place d'un Cloud avec Eucalyptus	45
3.2	Configuration matérielle et logicielle	47
3.3	Les mesures de la QdS	49
3.4	Comparaison des résultats avec et sans AOP4CSM	54

Introduction Générale

Avec la généralisation d'Internet, le développement des réseaux haut débit, la location d'application et le paiement à l'usage résultent de l'apparition d'un nouveau concept : Le "*Cloud Computing*". Celui-ci consiste en une interconnexion et une coopération de ressources informatiques, situées dans diverses structures internes, externes ou mixtes et dont le mode d'accès est basé sur les protocoles et standards Internet. Le *Cloud Computing* est devenu ainsi, le sujet le plus débattu aujourd'hui dans le secteur des technologies de l'information. Le consensus qui se dégage est que le *Cloud Computing* jouera un rôle de plus en plus important dans les opérations informatiques des entreprises au cours des années à venir. C'est pour cela que ce travail de maîtrise s'intéresse au concept du *Cloud Computing*. Ainsi, nous avons mené une étude exhaustive sur les alternatives open-sources des plateformes du *Cloud Computing*. Ceci nous a permis d'avoir et de présenter une idée riche sur les techniques disponibles pour la création d'un environnement du Cloud.

En outre, le succès de la technologie d'Internet a apporté une nouvelle branche de composants logiciels, connue sous le nom de services Web. Les services Web demeurent aujourd'hui un acteur majeur dans la mise en œuvre des applications distribuées. Outre leur capacité d'interconnecter diverses données dans des milieux hétérogènes, ils sont caractérisés par leur réutilisabilité ainsi que leur indépendance de la plateforme de déploiement. Avec la prolifération du *Cloud Computing*, les fournisseurs de service ont tendance à déployer leurs services sur ce nouvel environnement.

Cependant, les services Web ne peuvent être considérés comme efficaces que s'ils répondent à temps aux contraintes des utilisateurs finaux tels que le temps de réponse. Ainsi, toute augmentation du temps de réponse d'un service Web est

considérée comme une dégradation de ses performances, voire même une panne.

Le monitoring de la Qualité de Service (QoS) constitue donc une étape primordiale pour la satisfaction des besoins des clients. Plusieurs solutions de monitoring existent actuellement. Parmi les quelles, nous citons AOP4CSM [39], une approche de monitoring développée au sein de l'équipe *ReDCAD*. Il s'agit d'une approche non invasive, conçue principalement pour les services du Cloud et permettant la collecte des paramètres de la QoS. L'objectif de ce travail est d'expérimenter sur les environnements du *Cloud Computing* l'approche AOP4CSM [39] et de montrer par la suite son efficacité.

Pour ce faire, nous avons opté pour une solution IaaS commerciale du *Cloud Computing* pour mettre en place notre propre environnement du Cloud à savoir le MiniCloud d'OVH. Ainsi, nous avons réalisé des expérimentations à l'échelle du *Cloud Computing* (MiniCloud d'OVH) pour valider AOP4CSM [39]. En se basant sur des mesures répétées plusieurs fois, nous avons dégagé les courbes de variation de la QoS qui peuvent être utilisées par la suite comme références soit lors de la découverte soit lors d'une analyse de l'état du service.

Le présent travail s'articule autour de trois chapitres : Dans le premier chapitre, nous expliquons, d'une part, quelques généralités à propos du *Cloud Computing* et d'autre part, étudions et présentons des solutions du *Cloud Computing* existantes. Le deuxième chapitre détaille la notion de la qualité de service dans les services Web ainsi que les techniques existantes pour les observer. De plus, il présente les détails de notre propre contribution à savoir notre architecture d'expérimentation sur le *Cloud Computing*. Le dernier chapitre est consacré à la description des expérimentations effectuées ainsi que l'analyse et l'interprétation des résultats obtenus. Ce rapport est clôturé par une conclusion et des perspectives.

1.1 Introduction

Le *Cloud Computing* est un concept qui regroupe plusieurs technologies servant à délivrer différents services. Il peut être schématisé par un ensemble indéterminé de ressources informatiques interconnectées dans un réseau bien défini (figure 1.1). L'accès aux services se fait à la demande par une application standard facilement disponible à savoir, un navigateur Web. Il est devenu un concept majeur faisant référence à l'utilisation de la mémoire et des capacités de calcul des ordinateurs et des serveurs répartis dans le monde entier.

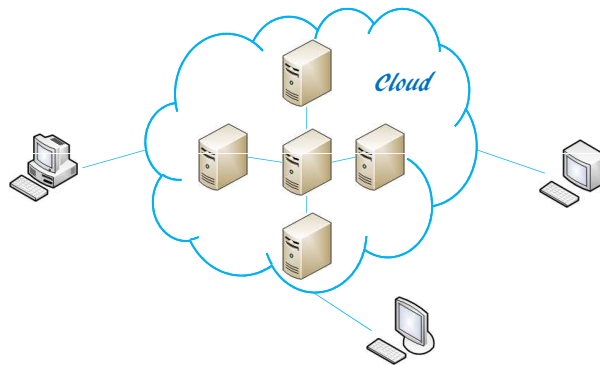


FIGURE 1.1 – Cloud Computing

Le *Cloud Computing* est un passage de l'informatique vers Internet. Les utilisateurs ne sont plus propriétaires de leurs serveurs informatiques mais peuvent accéder de manière évolutive à de nombreux services en ligne sans avoir à gérer l'infrastructure sous-jacente, souvent complexe. Les entreprises, dans ce cadre, n'auraient plus besoin de salles blanches ni de serveurs ni d'informaticiens. Toutes les applications

sont louées et exécutées au travers des navigateurs ou des serveurs d'applications.

1.2 Le Cloud Computing

1.2.1 Les types de Cloud Computing

Nous distinguons trois types de *Cloud Computing*. Le Cloud publique, également le premier apparu, le Cloud privé et le Cloud hybride qui est en fait la combinaison des deux premiers.

Le Cloud publique

Le principe est d'héberger des applications, en général des applications Web, sur un environnement partagé avec un nombre illimité d'utilisateurs. La mise en place de ce type de Cloud est gérée par des entreprises tierces (exemple Amazon, Google, etc.) et est accessible selon le modèle pay-as-you-go (payer selon la consommation) [15].

Les fournisseurs du Cloud publique les plus connus sont Google et Amazon.

Le Cloud privé

C'est un environnement déployé au sein d'une entreprise. Ainsi, elle doit gérer toute seule son infrastructure. Dans ce cas, implémenter un Cloud privé signifie transformer l'infrastructure interne en utilisant les technologies telles que la virtualisation pour enfin délivrer, plus simplement et plus rapidement, des services à la demande. L'avantage de ce type de Cloud par rapport au Cloud publique réside dans l'aspect de la sécurité et la protection des données [15]. En effet, l'ensemble du matériel est conservé au sein de votre propre emplacement. De ce fait, les ressources sont détenues et contrôlées par votre propre département informatique.

Eucalyptus, *OpenNebula* et *OpenStack* sont des exemples de solution pour la mise en place du Cloud privé.

Le Cloud hybride

En général, on entend par Cloud hybride la cohabitation et la communication entre un Cloud privé et un Cloud publique dans une organisation partageant des données et des applications (Par exemple, un Cloud dédié pour les données et un autre pour les applications) [15].

1.2.2 Les services de Cloud Computing

Le *Cloud Computing* permet aux entreprises de consommer des services à la demande. Les fournisseurs du Cloud distinguent trois services.

Infrastructure as a Service (IaaS)

Il s'agit d'une mise à disposition, à la demande, de ressources d'infrastructures (stockage, machines virtuelles, système d'exploitation (SE), etc.) dont la plus grande partie est localisée à distance dans des Datacenter¹. Dans ce type de service, seul le matériel est dématérialisé² [5].

Amazon EC2 (Amazon Elastic Compute Cloud) est un exemple d'IaaS qui permet de louer des machines virtuelles de tailles prédéfinies pour exécuter des applications.

Plateforme as a Service (PaaS)

Le PaaS dispose d'environnements spécialisés au développement comprenant les langages, les outils et les modules nécessaires. Dans ce type de service le matériel, l'hébergement et le Framework d'application sont dématérialisés [5].

Hadoop est un exemple de PaaS destiné aux applications distribuées et à la gestion intensive des quantités immenses de données.

1. Centre de traitement des données : lieu où se trouvent différents équipements électroniques, surtout des ordinateurs et des équipements de télécommunications

2. Plutôt que d'investir dans des serveurs, le *Cloud Computing* permet de disposer de tout ses serveurs virtuels hébergés sur Internet

Software as a Service (SaaS)

Sans aucune installation d'application sur l'ordinateur, le logiciel (Software) est directement utilisable à travers le navigateur Web comme service final. Ainsi, l'utilisation reste transparente pour les utilisateurs, qui ne se soucient ni de la plateforme ni du matériel. Le matériel, l'hébergement, le Framework d'application et le logiciel sont dématérialisés [5].

Amazon S3 (Amazon Simple Storage Service) est un exemple de SaaS qui est une plateforme de stockage en ligne. Il utilise une interface Web pour stocker et extraire les données.

La figure 1.2 montre les trois couches du *Cloud Computing* ainsi que leurs acteurs en donnant un compromis flexibilité/simplicité.

En Cloud, la flexibilité est obtenue grâce à la virtualisation des systèmes d'exploitation. La plateforme est exécutée via des machines virtuelles et les ressources peuvent être allouées et délibérées à la demande. Ainsi, l'IaaS est considéré le service le plus flexible. D'autre part, l'IaaS est le service le plus simple à mettre en place.

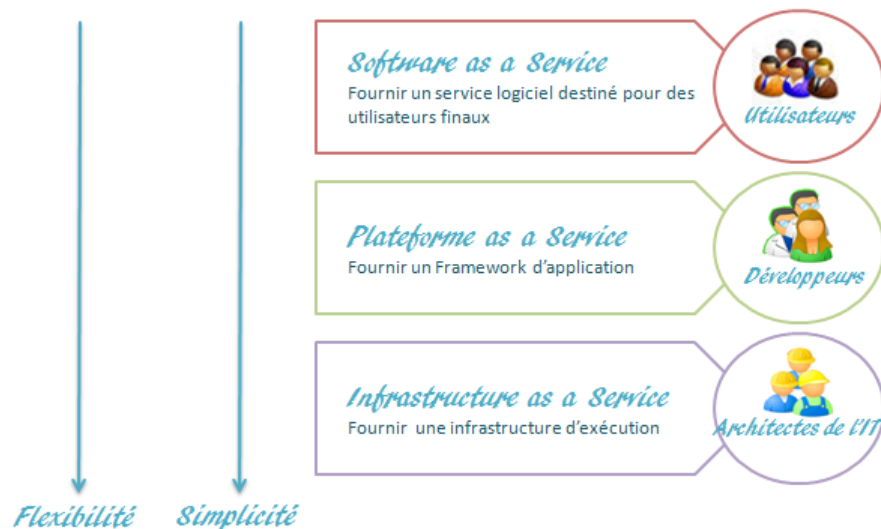


FIGURE 1.2 – Les différents niveaux des services du Cloud Computing

Les principes du *Cloud Computing* incluent :

- La virtualisation et l'automatisation ;

- Une administration centralisée des ressources.

1.3 La virtualisation

1.3.1 Définition

La virtualisation est l'ensemble de techniques et d'outils permettant de faire tourner plusieurs systèmes d'exploitation sur une même machine physique afin de délivrer une meilleure utilisation des ressources. Cette technologie vient pour répondre à certains problèmes tels que :

- **La sous exploitation des serveurs physiques** : il est estimé que dans un Datacenter privé, le taux d'utilisation moyen est de 10%. Celui-ci passe à 35% sur une architecture virtuelle [27].
- **La croissance du nombre de serveurs physiques** : les ressources physiques d'un serveur seront partagées entre différents serveurs virtuels ce qui permet de ne pas acheter plusieurs serveurs physiques.
- **La sécurité et la fiabilité** : isoler les services sur des serveurs différents.

Dans les systèmes de virtualisation, il faut noter les notions suivantes :

- **SE hôte** : le système d'exploitation installé sur la machine physique ;
- **SE invité** : les systèmes d'exploitation des machines virtuelles ;
- **Partage des ressources physiques** : les différentes machines virtuelles installées sur le serveur partagent ces ressources à savoir le processeur, les disques durs et d'autres périphériques ;
- **Isolation** : les machines virtuelles sont considérées comme des ordinateurs physiques et donc possèdent chacune sa propre adresse IP ;
- **Manipulation des machines virtuelles** : une machine virtuelle est un fichier situé sur un disque du serveur.

1.3.2 Techniques de virtualisation

La virtualisation permet de cohabiter plusieurs systèmes d'exploitation complètement isolés dans un même hôte [9].

On distingue plusieurs techniques de virtualisation à savoir : l'isolation, la paravirtualisation et la virtualisation complète. Ces trois techniques sont détaillées dans ce qui suit.

L'isolation

L'isolation permet de diviser un système d'exploitation en plusieurs espaces mémoires ou encore contextes. Chaque contexte est géré par le SE hôte. Cette isolation permet de faire tourner plusieurs fois la même application prévue pour ne tourner qu'une seule fois par machine.

Les programmes de chaque contexte ne sont capables de communiquer qu'avec les processus et les ressources associées à leur propre contexte [9]. L'isolation est uniquement liée aux systèmes Linux.

La figure 1.3 présente l'architecture de la technique d'isolation.

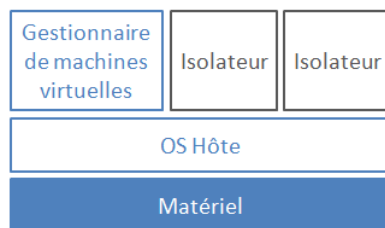


FIGURE 1.3 – Isolation

La paravirtualisation (virtualisation type 1)

La paravirtualisation est une technique de virtualisation qui présente à la machine invitée une interface logicielle similaire mais non identique au matériel réel. Ainsi, elle permet aux systèmes d'exploitation invités d'interagir directement avec le système d'exploitation hôte et donc ils seront conscients de la virtualisation [9].

La figure 1.4 présente l'architecture d'une virtualisation type 1.

La virtualisation complète (virtualisation type 2)

La virtualisation complète (en anglais full-virtualization) est une technique de virtualisation qui permet de créer un environnement virtuel complet. En utilisant

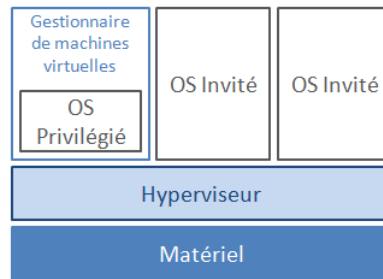


FIGURE 1.4 – Paravirtualisation

cette technique, le système d'exploitation invité n'interagit pas directement avec le système d'exploitation hôte et donc il croit s'exécuter sur une véritable machine physique.

Cette technique de virtualisation ne permet de virtualiser que des SE de même architecture matérielle que l'hôte [9].

La figure 1.5 présente l'architecture d'une virtualisation type 2.

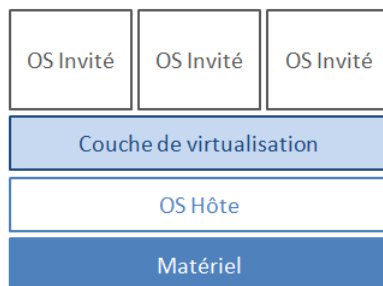


FIGURE 1.5 – Virtualisation complète

Dans certaines architectures matérielles, le support de virtualisation est intégré avec le processeur. Les exemples les plus connus du marché sont : AMD-V et Intel VT.

Synthèse

Nous venons d'étudier trois techniques de virtualisation à savoir l'isolation, la paravirtualisation et la virtualisation complète. Chaque technique possède ses points forts et ses points faibles. De ce fait, le choix d'un système de virtualisation adapté

à ses besoins apparaît assez complexe. Pour essayer de guider l'utilisateur à choisir la solution la plus convenable à son contexte, nous avons dressé le tableau 1.1.

TABLE 1.1 – Techniques de virtualisation

	Isolation	Paravirtualisation	Virtualisation complète
Type des SE invités	Même type (Linux)	Type différent mais avec une architecture identique (doit être adaptée à la couche de virtualisation -> conscient d'être virtualisés)	Type différent mais avec une architecture identique (n'est pas adaptée à la couche de virtualisation -> croit dialoguer directement avec le matériel)
Performance	++ (Faible surcoût)	+++ (SE invités fonctionnent en ayant conscience d'être virtualisés)	+ (L'unité centrale de calcul, c'est-à-dire le CPU, la RAM ainsi que la mémoire de stockage sont directement accessibles aux machines virtuelles)
Simplicité	+++	+	++
Exemples	OpenVZ	Xen, HyperV	KVM, VirtualBox

En conclusion, nous pouvons dire que :

- L'isolation est considérée comme la solution la plus performante. Cependant son inconvénient réside sur le fait qu'elle est incomplète ainsi, le SE doit être le même et de type Linux.
- La paravirtualisation suppose que le noyau du SE invité soit légèrement modifié. De ce fait, si le système ne dispose pas de fonctions dédiées à la paravirtualisation dans son noyau, cette technique devient inutilisable. L'objectif majeur de cette technique est d'offrir un accès quasi identique aux ressources matérielles entre système hôte et système invité.
- La virtualisation complète permet d'exécuter le SE invité de manière native sans modification. En revanche, cette solution est considérée comme la moins performante puisque le système invité ne participe pas au processus de virtualisation et doit traverser la couches de virtualisation pour accéder aux ressources matérielles.

1.3.3 Solutions de virtualisation

Dans cette section, nous présentons les outils de virtualisation les plus utilisés qui sont : *OpenVZ*, *Xen*, *KVM*, *VirtualBox*, *VMware* et *HyperV*.

OpenVZ

OpenVZ est une solution de virtualisation de type isolation basée sur le noyau linux et permettant à un serveur physique d'exécuter plusieurs instances de systèmes d'exploitation isolés. *OpenVZ* offre moins de flexibilité dans le choix du système d'exploitation. En effet, le système d'exploitation invité et hôte doivent être de type Linux.

Xen

Xen est une solution libre de virtualisation permettant de faire tourner plusieurs systèmes d'exploitation sur une même machine physique. Il est de type hyperviseur, c'est à dire qu'il vient s'insérer entre le matériel et le noyau. *Xen* est considéré comme une solution à base de paravirtualisation, car les systèmes invités doivent être modifiés pour cohabiter [28].

KVM

KVM est un projet de virtualisation complète qui est actuellement en développement pour un module de paravirtualisation. Il est intégré depuis le noyau *Linux* 2.6.20 et permettant une virtualisation matérielle des processeurs. Ainsi, il ne fonctionne que sur un processeur de type Intel VT ou AMD-V [12].

VirtualBox

VirtualBox est une solution de virtualisation permettant d'émuler le comportement d'un ordinateur réel auprès du système invité [24]. Il est considéré comme une solution de virtualisation complète.

VMware Server

VMware est une société qui offre des produits propriétaires liés à la virtualisation d'architectures x86. Elle est leader dans le marché de la virtualisation pour PC. Son produit de virtualisation VMware Server est de type virtualisation complète pour serveur sous GNU/Linux et/ou Microsoft Windows [25].

HyperV

HyperV est une solution de virtualisation basée sur la virtualisation 64 bits pour Microsoft. Il est considéré comme une solution de paravirtualisation [14].

Synthèse

Le tableau 1.2 montre une étude comparative de ces différentes solutions de virtualisation.

TABLE 1.2 – Solutions de virtualisation

Développé par	OpenVZ [3] [20] [36]	Xen [23] [28] [3] [36]	KVM [12] [3] [36]	VirtualBox [3] [24] [35]	VMware Server [24] [32]	HyperV [32]
	SWsoft	Université de Cambridge	Qumranet en Israël	Innotek (racheté par Sun)	VMware	Microsoft
Systèmes d'exploitation hôtes	Toutes les distributions Linux	Linux (certaines versions spécifiques)	Linux (intégrer dans le noyau)	Toutes les versions Windows et Linux, Macintosh et OpenSolaris	La plupart des versions Windows, Linux et MacOS	<ul style="list-style-type: none"> - Windows XP, Vista, 7 - SUSE Linux Enterprise Server - RedHad Linux Enterprise
Systèmes d'exploitation invités compatibles	Linux (même type que celui du système hôte)	Toutes les versions Windows, Linux et Solaris	Tout système d'exploitation	<ul style="list-style-type: none"> - Toutes les versions Windows et Linux - Solaris - Windows XP - OpenSolaris - OpenBSD 	<ul style="list-style-type: none"> - Windows 2000, XP, Vista, 7 - Linux (2.4 and 2.6) - Solaris - FreeBSD 	<ul style="list-style-type: none"> - Windows XP, Vista, 7 - SUSE Linux Enterprise Server - RedHad Linux Enterprise Linux
Type de virtualisation	Isolateur	<ul style="list-style-type: none"> - Paravirtualisation (SE modifié) - Virtualisation complète (sur des processeurs supportant les technologies InetVT et AMD-V) 	<ul style="list-style-type: none"> - Paravirtualisation (SE modifié) - Virtualisation complète (en cours de développement) 	Virtualisation complète (SE non modifié)	Virtualisation complète (SE non modifié)	Paravirtualisation
Plateforme	<ul style="list-style-type: none"> - x86 (processeur 32 bits) - AMD64/Intel64 (processeur 64 bits) 	<ul style="list-style-type: none"> - x86 (processeur 32 bits), x64 (processeur 64 bits) - PowerAMC (processeur à architecture power) - ARM 	X86 (processeur 32 bits)	<ul style="list-style-type: none"> - x86 (processeur 32 bits) - AMD64/Intel64 (processeur 64 bits) 	x64 (processeur 64 bits)	x64 (processeur 64 bits)
Contrainte matérielle (virtualisation enabled)	Avec/sans	Avec/sans	Avec	Avec/sans	Avec/sans	Avec
Migration à chaud de machines virtuelles	✓	✓	✓	✓	✓	✗
Utilisation dans le Cloud		Amazon, Cloud.com, GoGrid, Rackspace			Google, Salesforce	Microsoft Azure
Gratuit-Payant	Gratuit (Licence GPL)	Gratuit (Licence GPL)	Gratuit (Licence GPL)	Gratuit (Licence GPL)	Payant	Payant

Ces solutions utilisent des techniques de virtualisation variées. Certaines parmi elles permettent de faire cohabiter plusieurs SE dans un seul serveur physique (Xen, KVM, VMware, VirtualBox). D'autres solutions permettent à un unique système d'être exécuté plusieurs fois dans un mode multi-instance (OpenVZ). Certaines s'appuient sur les capacités du matériel (KVM) alors que d'autres nécessitent un système d'exploitation modifié pour cohabiter avec la solution de virtualisation (Xen).

Parmi les solutions étudiées :

- Xen s'avère la solution la plus complexe de point de vue technique mais c'est la solution la plus performante et la plus complète du marché. Elle est aussi la solution la plus utilisée dans le Cloud.
- KVM est plus simple à utiliser que Xen mais, il est moins performant. Le point faible de cette solution de virtualisation est le fait qu'il exige un matériel dédié à la virtualisation.
- OpenVZ est une solution performante mais elle présente des problèmes de compatibilité. En effet, OpenVZ ne fonctionne que sur des machines supportant un système d'exploitation Linux.
- VirtualBox est une solution complète qui peut fonctionner sur tout système d'exploitation y compris les systèmes Macintosh.
- VMware Server est la première solution utilisée dans le domaine de *Cloud Computing*.

1.4 Solutions IaaS open source de Cloud Computing

Le *Cloud Computing* représente un nouveau défi dans le monde informatique. Plusieurs solutions sont proposées : des solutions propriétaires et des solutions open-sources. La mise en place d'un environnement du *Cloud Computing* pour des buts de recherche nécessite initialement le choix d'une solution open source et puis son installation. La décision est souvent très difficile à prendre puisque chaque solution possède ses propres caractéristiques.

Une comparaison des solutions du *Cloud Computing* représente donc un bon point de départ. Voici un panorama de six outils open-sources de *Cloud Compu-*

ting offrant une infrastructure comme service et permettant une gestion simplifiée d'architectures matérielles complexes.

1.4.1 Eucalyptus

Présentation

Eucalyptus est un outil open source issue d'un projet de recherche de l'université de Californie. Cette solution est la plus connue, car elle est intégrée dans les distributions *Ubuntu Server* et *Debian* [8].

Eucalyptus est écrit en C, Java et Python et permet de créer des Clouds IaaS de type privé ou hybride. Il supporte les machines virtuelles Linux ainsi que les hyperviseurs Xen et KVM. Son avantage majeur est le fait qu'il est compatible avec Amazon *EC2*. Il possède également une version entreprise (payante) de la société *Eucalyptus Systems* qui apporte des fonctionnalités supplémentaires comme le support de *VMware*.

Architecture

Eucalyptus comporte cinq composants principaux (figure 1.6) :

- **Cloud Controller** : c'est l'unique point d'entrée (*Front end*) pour tous les utilisateurs et les administrateurs d'*Eucalyptus*. Il est responsable de la gestion de tout le système.
- **Cluster Controller** : il conserve les informations relatives aux ressources physiques.
- **Node Controller** : il interagit avec l'hyperviseur pour gérer les machines virtuelles. Lors d'une demande d'allocation de ressources physiques par le *Cloud Controller*, c'est le *Cluster Controller* qui alloue les ressources en sollicitant les *Nodes Controllers*.
- **Walrus** : c'est le composant qui gère l'accès aux services de stockage.
- **Storage Controller** : ce composant fonctionne avec le composant *Walrus* et permet de stocker les images des machines virtuelles et les données des utilisateurs.

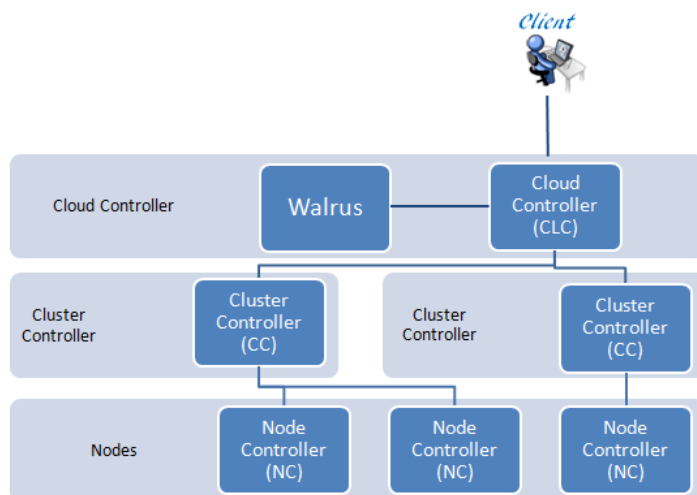


FIGURE 1.6 – Architecture d'Eucalyptus

1.4.2 OpenNebula

Présentation

Il s'agit d'une plateforme purement open-source permettant de déployer des Clouds privés, hybrides et publiques [18]. Elle est écrite en C++, Ruby et Shell et elle supporte les hyperviseurs *Xen*, *KVM* et *VMware*. Le support de *Virtualbox* est prévu à partir de la version 4.0 de *VirtualBox*. Sa puissance consiste dans ses connecteurs vers des fournisseurs d'IaaS sur les Clouds publics tels que : *Amazon EC2 Web Service*, *Nimbus WSRF*, *ElasticHosts REST*, etc.

OpenNebula est soutenu par le projet européen *RESERVOIR*³, qui propose une architecture complète pour la gestion de Datacenter et la création de services Cloud.

Architecture

L'architecture interne d'*OpenNebula* peut être divisée en trois couches (figure 1.7) :

- **Tools** : c'est l'ensemble des outils de gestion pour *OpenNebula* ;
- **Core** : il se compose d'un ensemble de composants pour contrôler les machines virtuelles, le stockage et le réseau virtuel ;
- **Drivers** : l'interaction entre *OpenNebula* et l'infrastructure de Cloud est ef-

3. <http://www.reservoir-fp7.eu/>

fectuée par des pilotes spécifiques qui sont les drivers.

Les machine *Front end* et *Node* sont reliés entre eux à travers un réseau privé.

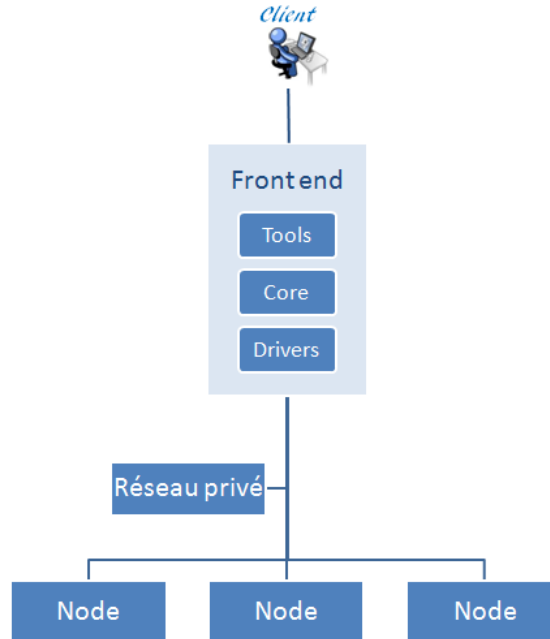


FIGURE 1.7 – Architecture d’OpenNebula

1.4.3 Nimbus

Présentation

Issu du monde de la recherche, *Nimbus* permet de déployer un Cloud de type IaaS [17]. Cette plateforme supporte les hyperviseurs Xen et KVM, et peut s’interfacer avec le Cloud Amazon EC2. Elle est associée au projet *Cumulus*⁴, qui permet de déployer des services de stockage en Cloud, compatible avec le Cloud Amazon S3.

Architecture

Nimbus comprend plusieurs composants qui peuvent être regroupés selon trois contextes comme le montre la figure 1.8. Ces composants sont reliés entre eux à travers un réseau privé.

4. <http://www.roehampton.ac.uk/cumulus/>

- **Client-supported modules** : il est utilisé pour gérer les clients du Cloud. Il comprend le *context client module*, le *Cloud client module*, le *reference client module* et l'*EC2 client module*.
- **Service-supported modules** : il fournit tous les services du Cloud. Il comprend le *context broker module*, le *Web service resource Framework module*.
- **Background resource management modules** : son rôle est de gérer les ressources physiques du Cloud. Il comprend différents modules à savoir : *workspace service manager module*, *IaaS gateway module*, *EC2 module*, *workspace pilot module*, *workspace resource management module* et *workspace controller*.

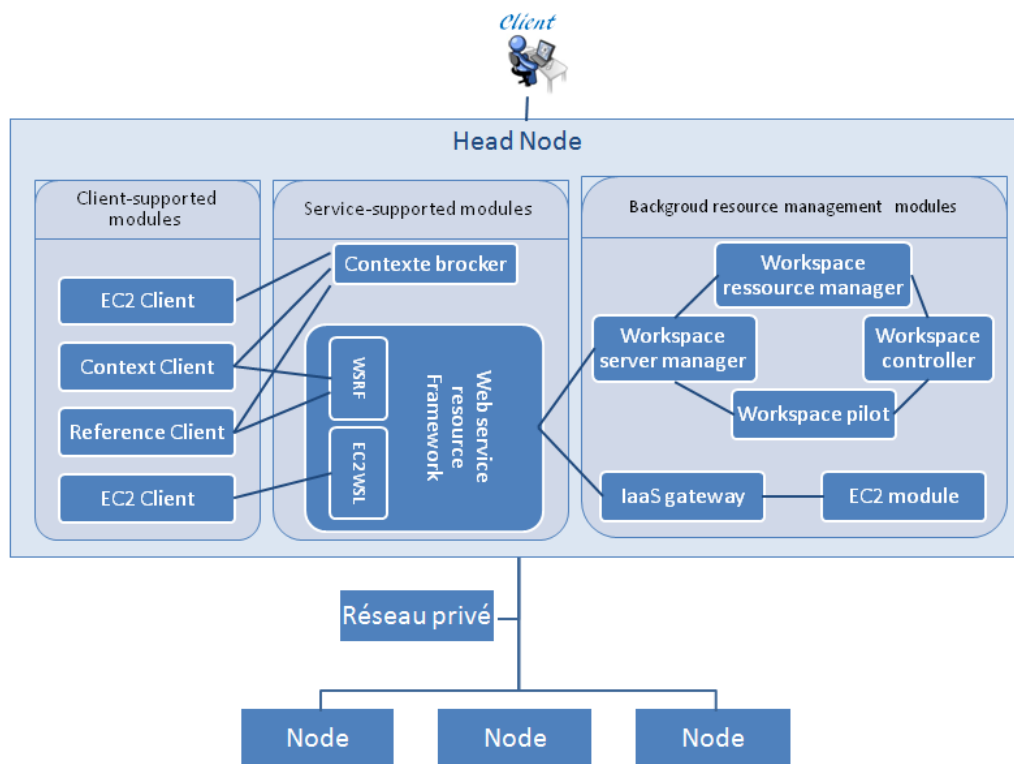


FIGURE 1.8 – Architecture de Nimbus

1.4.4 Xen Cloud Platform

Présentation

Xen Cloud Platform (XCP) est une plateforme open-source du *Cloud Computing* développée par la communauté Xen et distribuée sous licence GPL⁵. Elle a pour but de proposer une plateforme open-source gratuite pour construire et faire dialoguer des services du *Cloud Computing* [29].

Architecture

Les composants principaux de Xen Cloud Platform sont :

- **XCP Host** : il consiste en un système d'exploitation Xen.
- **Master XCP Host** : il gère les *XCP hosts*.
- **Shared storage** : c'est un composant optionnel où sont stockées les machines virtuelles. Ce composant permet aux administrateurs de déplacer des machines virtuelles d'un *XCP host* à un autre.

La figure 1.9 montre l'architecture de *Xen Cloud Platform*.

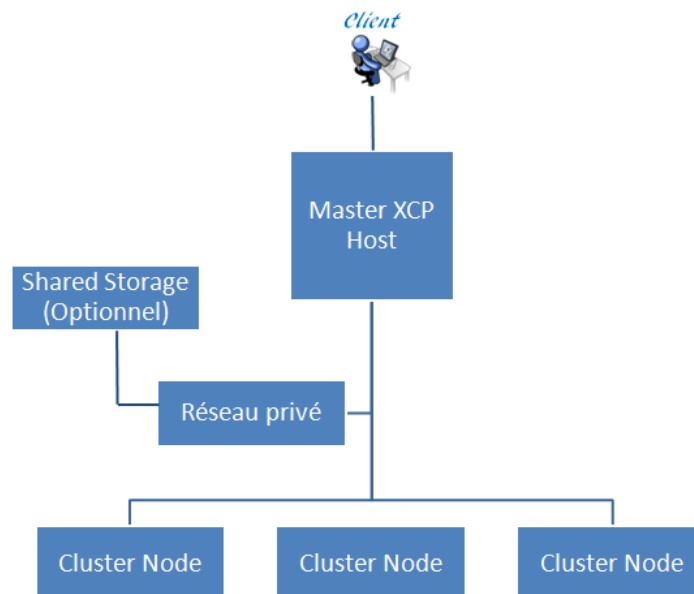


FIGURE 1.9 – Architecture de Xen Cloud Platform

5. Une licence qui fixe les conditions légales de distribution des logiciels libres du projet GNU

1.4.5 AbiCloud

Présentation

AbiCloud, principalement développé par *Abiquo*, est la plateforme du *Cloud Computing* permettant de créer et de gérer des Cloud publiques, privés et hybrides [1].

Architecture

AbiCloud se compose principalement de trois éléments (figure 1.10) :

- **AbiCloud server** : il consiste en un système d'exploitation Xen.
- **AbiCloud WS (AWS)** : il est responsable de la gestion des machines virtuelles.
- **Virtual System Monitor (VSM) WS** : c'est le composant qui permet de suivre toute l'infrastructure virtuelle du Cloud.

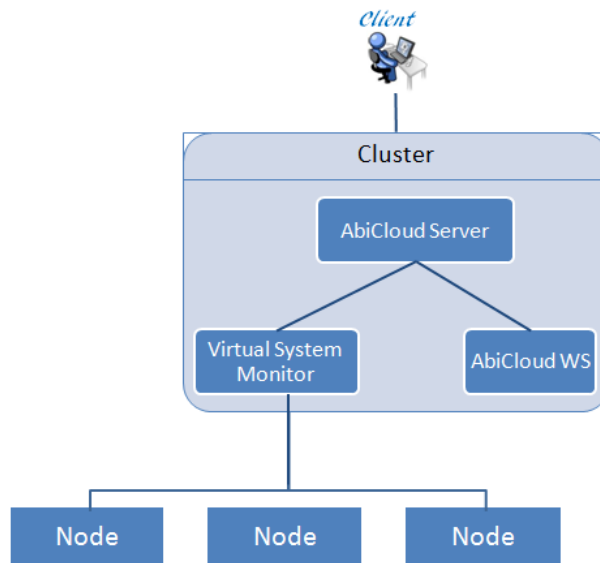


FIGURE 1.10 – Architecture d'AbiCloud

1.4.6 OpenStack

Présentation

Créé en juillet 2010 par la *NASA* et l'hébergeur américain *Rackspace*, *OpenStack* est une offre d'IaaS 100% open-source encore en développement qui a livré son code source récemment et qui permet aux sociétés de développer leurs propres solutions d'infrastructure du *Cloud Computing* [19].

Plus que trente fournisseurs soutiennent ce projet tels que : *AMD*, *Intel*, *Dell* et *Citrix*. *OpenStack* devrait également être intégré dans les prochaines versions d'*Ubuntu* comme c'est le cas pour *Eucalyptus*. Il comprend le logiciel *OpenStack Compute* pour la création automatique et la gestion de grands groupes de serveurs privés virtuels et le logiciel *OpenStack Storage* pour optimiser la gestion de stockage, répliquer le contenu sur différents serveurs et le mettre à disposition pour une utilisation massive de données.

Architecture

- **OpenStack Compute** : c'est le contrôleur du Cloud. Il est utilisé pour démarrer des instances virtuelles et configurer le réseau pour chaque instance.
- **OpenStack Object Storage** : c'est un système pour stocker des objets de capacité massivement évolutive.
- **OpenStack Imaging Service** : il permet de récupérer les machines virtuelles.

La figure 1.11 montre les relations fondamentales entre ses différents composants.

Afin de caractériser chaque solution et permettre le choix adéquat pour la construction d'un Cloud, nous avons réalisé une étude comparative de ses six solutions en se basant sur différents critères de classification :

- **PRODUIT PAR** : la société qui a produit la solution.
- **BUT** : l'objectif à viser par la création de la solution.
- **LE DOMAINE D'UTILISATION** : le *Cloud Computing* est présent dans plusieurs domaines d'utilisation. Ainsi, il serait important de présenter le contexte de chaque solution afin d'aider les fournisseurs à bien choisir celle qui répond le plus à leurs besoins.

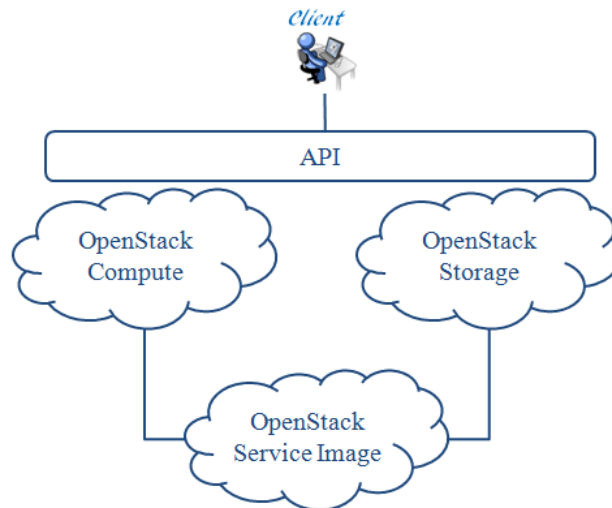


FIGURE 1.11 – Architecture d'OpenStack

- L'ARCHITECTURE : c'est la structure du système qui comprend les ressources matérielles et logicielles et l'interaction entre eux.
- SYSTÈMES D'EXPLOITATION SUPPORTÉS : pour construire un *Cloud Computing*, on doit disposer d'un système d'exploitation qui gère les ressources physiques en permettant leur allocation et leur partage.
- LANGAGE DE PROGRAMMATION : il représente le langage de programmation avec lequel la solution est implémentée.
- STOCKAGE : pour gérer des grands volumes de données, la solution la plus économique, qui connaît également à l'heure actuelle un vif succès, est le *Cloud Computing*.
- RÉSEAU : dans le *Cloud Computing*, nous entendons par réseau, le réseau des machines virtuelles.
- L'INTERFACE UTILISATEUR : l'interface utilisateur représente l'interface de communication entre l'utilisateur et le Cloud.
- SÉCURITÉ : le *Cloud Computing* apparaît comme une opportunité formidable pour les entreprises mais, pose logiquement la question de sécurité. Il est donc essentiel de protéger les ressources et de contrôler l'accès des utilisateurs et des administrateurs au système du *Cloud Computing*.

- EQUILIBRAGE DE CHARGE : un des aspects les plus importants est la capacité de répondre aux requêtes dans un temps acceptable, même en cas d'affluence massive. L'équilibrage de charge consiste alors à distribuer une tâche à un ensemble de serveurs ou de périphériques afin de répartir la charge globale vers différents équipements et de s'assurer de leur disponibilité.
- TOLÉRANCE AUX PANNES : c'est la capacité d'un système à fonctionner malgré une défaillance d'un de ses composants.
- MIGRATION DES MACHINES VIRTUELLES : c'est le fait de déplacer des machines virtuelles d'un serveur physique à un autre. On distingue deux types de migration, à savoir la migration à chaud qui se fait lors de l'exécution des machines virtuelles et la migration à froid qui ne peut être réalisée qu'après l'arrêt de la machine virtuelle à déplacer.
- EMBLACEMENT DES MACHINES VIRTUELLES : savoir l'emplacement des machines virtuelles permet de garantir une utilisation optimale des ressources.
- COMPATIBILITÉ AVEC AMAZON EC2 : Amazon EC2 est la solution d'infrastructure du *Cloud Computing* la plus connue et la plus utilisée. Ainsi, il serait intéressant d'étudier la compatibilité des solutions d'IaaS du *Cloud Computing* par rapport à Amazon EC2.
- UTILISATION DANS LE MARCHÉ : l'ensemble des solutions du *Cloud Computing* du marché qui utilisent chacune des solutions IaaS du Cloud déjà étudié. Cette information permet de bien mettre en valeur ces solutions IaaS.

1.4.7 Synthèse

Le tableau 1.3 montre une étude comparative entre les différentes solutions IaaS open-sources de *Cloud Computing*.

TABLE 1.3 – Solutions IaaS open source de Cloud Computing

	<i>Eucalyptus</i> [41] [40] [42] [43] [37] [34]	<i>OpenNebula</i> [41] [40] [42] [43] [37] [34]	<i>Nimbus</i> [41] [40] [42] [43] [37] [34]	<i>Xen Cloud Platform</i> [28] [29] [31] [40] [42]	<i>AbiCloud</i> [1] [41] [31] [43] [37] [34]	<i>OpenStack</i> [19]
Produit par	- Apparçu au début par l'université Santa Barbara de l'université de Californie - <i>Eucalyptus</i> System Company	L'union Européenne	Chercheurs de l'université de Chicago	Citrix XenServer	Abico	Rackspace, NASA, Dell, Citrix, Cisco, Canonical et plus que 50 autres organisations
But	Une réponse open source pour le Cloud commerciale EC2	Un Cloud privé pure	Solution scientifique du <i>Cloud Computing</i>	- Evolution de Citrix XenServer - Configuration automatique et maintenance des <i>Cloud Computing</i>	La gestion des Clouds publics et privés dans des environnements hétérogènes	Créer et offrir des fonctionnalités de <i>Cloud Computing</i> en utilisant un logiciel open-source fonctionnant sur du matériel standard
Architecture	- Hiérarchique - Cinq composants - Supporte multiple cluster - Minimum deux serveurs	- Centralisé - Trois composants - Minimum deux serveurs	- Centralisé - Trois composants - Minimum deux serveurs	- Centralisé - Trois composants - Minimum deux serveurs	- Centralisé - Trois composants - Minimum deux serveurs	Intégration des deux composants OpenStack objet et OpenStack compute
Domaine d'utilisation	Les entreprises	Les chercheurs dans le domaine de <i>Cloud Computing</i> et de la virtualisation	Les communautés scientifiques (moins intéressés par les techniques internes du système)	Les entreprises qui veulent construire leurs propre Cloud et de le rendre interoperable avec ceux des principaux fournisseurs de services Cloud	Les entreprises qui veulent créer leur propre <i>Cloud Computing</i>	Les sociétés, les fournisseurs de services, les chercheurs et les centres de données mondiaux qui cherchent à déployer à grande échelle leurs Cloud privés ou publics
Systèmes d'exploitation supportés	Linux (Ubuntu, Fedora, CentOS, OpenSUSE et Debian)	Linux (Ubuntu, RedHat Enterprise Linux, Fedora et SUSE Linux Enterprise Server)	Linux (Ubuntu, RedHat Enterprise Linux)	- Linux (Fedora, RedHat, CentOS et Suse Linux Enterprise Server) - Windows 7	- Linux (Ubuntu et CentOS) - Mac OS - Windows XP	- Linux et récemment Windows - Exige x86 processor
Langage de programmation	Java, C, et Python	Java, Ruby, C++	Python, Java	Caml	Java, Ruby, C++, Python	Python
Stockage	Walrus	- GridFTP, Comulus (version récente de GridFTP) - XCP	- SCP - SQLite3	VastSky	HDFS	<i>OpenStack Storage</i>
Réseau	Serveur DHCP installé sur le cluster controller	Configuration manuelle par l'administrateur	Serveur DHCP installé sur chaque nœud	Open vSwitch	WSManagement	<i>OpenStack Compute</i>
Interface utilisateur	✓ EC2 WS API ✓ Outils tel que : HybridFox, ElasticFox	✓ EC2 WS API ✓ OCCT API	✓ EC2 WS API ✓ <i>Nimbus WSRF</i>	✗ Ligne de commande "XE" ✓ XenCenter et Versiera (application commerciale pour Windows)	✓ Interface Web implémentée avec Adobe Flex	✓ Interface Web
Sécurité niveau utilisateur	- Fichier zip téléchargeable à travers l'interface Web qui contient certificats - Connexion HTTPS	- Authentification	- Certificat X509	- Authentification - Connexion SSH	- Authentification (mois de passe stockés en format MD5)	- Certificat X509
Sécurité niveau administrateur	- Connexion SSH - Root exigé	- Root seulement si nécessaire (selon les droits d'accès)	- Connexion SSL - Intègre Globus (certification)	- Connexion SSH	- Authentification	- Connexion SSH
Equilibrage de charge	Le Cloud Controller	Nginx	Le context broker	XAPI	AbiServer	Le Cloud Controller
Tolérance aux pannes	Séparation des clusters controllers	Database backend (enregistre les informations des machines virtuelles)	Vérification périodique des nœuds du Cloud	Synchronisation transactionnelle à chaud des états des machines virtuelles entre serveurs physiques	✗	Replication
Migration à froid des VMs	✓	✓	✓	✓	✓	✓
Migration à chaud des VMs	✗	✓ Shared FS	✗	✓ - Open Virtualization Format - Shared Storage	✗	✗
Emplacement des VMs	Node controller	Cluster node	Physical nodes	XCP Host	Clouds nodes	<i>OpenStack Compute</i>
Compatibilité avec EC2	✓	✓	✓	✓	✗	✗
Utilisation dans le marché	NASA	Le projet <i>RESERVOIR</i> , NUBA	STAR		Principalement actif en Espagne	

- LE DOMAINE D'UTILISATION : *OpenNebula* semble être la solution la plus convenable pour les chercheurs dans le domaine du Cloud vu qu'elle est conçue d'être une solution scientifique du *Cloud computing*. *Eucalyptus*, *XCP* et *Abi-Cloud* intéressent plutôt les fournisseurs des entreprises qui souhaitent construire leur propre Cloud.

OpenStack est principalement mis en place pour intégrer ces deux services : *OpenStack Compute* et *OpenStack Storage* afin de permettre à différentes organisations (chercheurs, fournisseurs, sociétés, etc.) de déployer à grande échelle leurs Clouds privés ou publics.

Nimbus est utilisé souvent par les communautés scientifiques qui pourraient être moins intéressés par les techniques internes du système.

- L'ARCHITECTURE : dans le Cloud, il n'y a pas un type d'architecture mais plusieurs. Généralement, chaque fournisseur propose la sienne (centralisé ou hiérarchique).

- LE SYSTÈME DE STOCKAGE : stocker des données dans le Cloud est géré par des outils spécifiques :

Walrus : le service de stockage inclus dans *Eucalyptus* qui permet de stocker les données persistantes ;

SCP (Secure Copy) : une ligne de commande utilisée pour transférer des fichiers à travers le réseau en s'appuyant sur le protocole SSH ;

SQLite : une bibliothèque écrite en C qui propose un moteur de bases de données relationnelles accessibles par le langage SQL ;

GridFTP : un protocole de transfert de données performant et sécurisé ;

VastSky : un système de stockage pour *XCP* ;

HDFS (Hadoop Distributed File System) : un système de fichiers qui propose un accès haut débit pour les données et est adapté pour les applications qui ont des données massives ;

OpenStack Storage : le service de stockage d'*OpenStack* qui permet d'optimiser la gestion de stockage, répliquer le contenu sur différents serveurs et le mettre à disposition auprès d'un grand nombre d'utilisateur.

- LE RÉSEAU :

- Avec *OpenNebula*, il faut configurer manuellement le réseau de machines virtuelles. Cette tâche de configuration est totalement automatisée avec *Eucalyptus* et *Nimbus* et est gérée par le serveur *DHCP*.
- Xen Cloud Platform s'appuie sur le logiciel *Open vSwitch* proposé par [28] et qui permettra la création des commutateurs virtuels.
- *WS-Management* est le protocole de communication permettant l'administration des serveurs utilisés avec *AbiCloud*.
- *OpenStack Compute* est le service de gestion de grands groupes de serveurs privés virtuels d'*OpenStack*.
- L'INTERFACE UTILISATEUR : la plupart des solutions offrent une interface Web pour assurer une interaction facile avec le Cloud. *Xen Cloud Platform* offre plutôt la ligne de commande "*XE*". Toutefois, il existe des solutions commerciales telles que *Versiera* et *XenCenter* (seulement pour Windows) qui sont proposées pour gérer *XCP* à l'aide d'une interface graphique.
- L'ÉQUILIBRAGE DE CHARGE : chaque solution possède sa façon à mettre en œuvre cet aspect :
 - *Eucalyptus* : le *Cloud Controller*, le contrôleur principal du Cloud, est responsable de la gestion de tout le système ;
 - *Nimbus* : le *Context Broker*, un composant de *Nimbus*, qui permet aux clients de coordonner de grands lancements de machines virtuelles automatiquement et de façon répétée. *AbiServer* est le responsable de la gestion de tout le système ;
 - *OpenNebula* : *Nginx*, un est un logiciel de serveur Web, permet de gérer plusieurs connexions en même temps ;
 - *Xen Cloud Platform* : *XAPI* est une interface qui assure la configuration et le contrôle des instances virtualisées ;
 - *OpenStack* : le *Cloud Controller* du composant *OpenStack Compute* est le responsable de la gestion des tous les composants du système.
- LA TOLÉRANCE AUX PANNES : plusieurs méthodes sont mises en œuvre pour assurer cette propriété :
 - La séparation des clusters controllers pour *Eucalyptus* réduit la chance de

- défaillance. En effet, ce composant gère plusieurs *Nodes Controllers* et il est responsable de la gestion du cycle de vie (création, utilisation, suppression, etc.) des instances virtuelles ;
- *OpenNebula* utilise une base de données persistante où les informations sur les machines virtuelles sont stockées. Cette solution garantit une grande protection des données mais, elle nécessite un personnel spécialisé (administrateur de données, administrateur du *SGBD*) et un coût d'acquisition assez élevé ;
 - *Nimbus* assure un contrôle régulier des nœuds (Checkpointing). Ainsi, une image des VMs en cours d'exécution sera stockée périodiquement afin qu'elle puisse être reprise en cas d'une défaillance ;
 - *Xen Cloud Platform* prend en charge la synchronisation transactionnelle à chaud des états des machines virtuelles entre les serveurs physiques ce qui permet de garantir un grand degré de fiabilité ;
 - La tolérance aux pannes par réplication est le principe utilisé avec *OpenStack*. Cette approche permet de masquer les pannes éventuelles : lorsqu'une machine tombe en panne, une des copies de cette machine prend sa place dans le système. Cependant, cette approche nécessite la disponibilité d'un nombre important de ressources.

1.5 Les acteurs du marché du Cloud Computing

Dans cette section, nous citons les solutions les plus connues dans le monde du *Cloud Computing*.

1.5.1 Amazon Web Services

Amazon a été la première société à proposer une plateforme du *Cloud Computing* avec *Amazon Web Services* [2]. Il s'agit d'un outil simple et facile à manipuler qui permet le développement des applications dans un environnement du *Cloud Computing*. Il est à noter que la technologie utilisée est issue à l'origine des systèmes de virtualisation de Xen. Les services d'*Amazon Web Services* sont :

- *Amazon Elastic Compute Cloud (EC2)* : il fournit un stockage basé sur les services Web ;
- *Amazon Simple Storage Service (S3)* : Amazon S3 offre une simple interface de services web à utiliser pour stocker et récupérer n'importe quelle quantité de données, à tout moment, de n'importe quel endroit sur le web ;
- *Amazon CloudFront* : il permet de distribuer des objets stockés sur S3 vers un emplacement proche de l'appelant ;
- *Amazon SimpleDB* : il permet aux développeurs d'exécuter des requêtes sur des données structurées ;
- *AWS Management Console (AWS Console)* : c'est une interface Web pour gérer et surveiller les infrastructures Amazon, incluant *Amazon EC2* et *Amazon S3*.

Selon [7], Amazon exploite quatorze Datacenter : huit aux États-Unis, quatre en Europe, et deux en Asie.

1.5.2 Microsoft Azure

Windows Azure, la seconde offre commerciale proposée par Microsoft après SQL Azure, peut être vue comme la base d'une solution PaaS [13]. Avec Windows Azure, Microsoft assure une couche qui aura comme rôle de structurer l'ensemble des technologies associées, que ce soit SQL, .NET, etc. Mais encore et toujours, Windows Azure reste "clairement destiné aux entreprises" selon [6].

1.5.3 Google App Engine

Google App Engine est une plateforme de développement et d'hébergement d'applications Web. Il permet d'exécuter des applications Web sur l'infrastructure de Google [10]. Il est lancé en 2008 et est disponible seulement en Cloud publique sous la forme d'une offre gratuite. Sa grande force réside dans le fait qu'il soit purement gratuit [6].

1.5.4 Google Apps

Une suite bureautique (Gmail, Google Agenda, Google Documents, etc.) de la société Google qui est accessible par les particuliers, les entreprises, les établissements d'enseignement et les organisations [11]. Il existe différentes versions de Google Apps :

- *Google Apps for Groups* : pour tout le monde mais limité à 50 utilisateurs ;
- *Google Apps for Education* : pour le primaire, le secondaire et l'université ;
- *Google Apps for Gouvernement* : pour les offices gouvernementaux ;
- *Google Apps for Business* : il s'agissait anciennement de Google Apps Premier Edition ;

1.5.5 Force.com

C'est une plateforme du *Cloud Computing* destinée pour les applications d'entreprise [22]. Force.com est un acteur PaaS qui fournit une plateforme pour créer et déployer des applications à la demande telle que la planification des ressources d'entreprise (ERP), gestion des ressources humaines (GRH) et la gestion de la chaîne d'approvisionnement (SCM). Force.com interopère avec la plateforme de Google App Engine.

1.5.6 OVH

OVH est une société française indépendante, orienté grand public. Elle est le numéro un de l'hébergement en France et se positionne comme le deuxième hébergeur européen (d'après [16]). Elle propose des serveurs privés dédiés, des ressources de stockage, des ressources de bande passante, des ressources, de CPU, de RAM, de SQL, du HTTP, des Emails etc. En 2010, OVH démarre une offre du *Cloud Computing* assez agressive se qui fait son succès [21].

1.5.7 Synthèse

Le tableau 1.4 est un comparatif de ses acteurs du *Cloud Computing*.

TABLE 1.4 – Acteurs du Cloud Computing

	OVH Cloud Computing [21] IaaS	Amazon EC2 (Iaas)	Web Services S3 (PaaS)	Force.com PaaS	Google App Engine [10] [42] PaaS	Microsoft Azure [13] [42] SQL Azure (SaaS)	Google Apps [11] [42] SaaS																																						
Service	Une infrastructure Cloud privée externalisée	<table border="1"> <tr> <td>Petite Instance (par défaut)</td> <td>Grande Instance</td> <td>Très grande Instance</td> </tr> <tr> <td>Stockage RAM 1,7 GO</td> <td>RAM 7,5 GO</td> <td>RAM 15 GO</td> </tr> <tr> <td>Stockage 160 GO</td> <td>Stockage 850 GO</td> <td>Stockage 1690 GO</td> </tr> <tr> <td>Plateforme</td> <td>Plateforme</td> <td>Plateforme</td> </tr> <tr> <td>32 GO</td> <td>64 GO</td> <td>64 GO</td> </tr> </table>	Petite Instance (par défaut)	Grande Instance	Très grande Instance	Stockage RAM 1,7 GO	RAM 7,5 GO	RAM 15 GO	Stockage 160 GO	Stockage 850 GO	Stockage 1690 GO	Plateforme	Plateforme	Plateforme	32 GO	64 GO	64 GO	Applications pour les RH, applications de stock, applications iPhone, iPad, Android et BlackBerry.	Générer des applications Web conçues pour un trafic élevé sans avoir à gérer l'infrastructure correspondante	Permet de réaliser puis exploiter des applications Web - Comprend des services Cloud d'hébergement (Azure Compute) et de stockage (Azure Storage)	<ul style="list-style-type: none"> - Permet de réaliser puis exploiter des applications Web - Comprend des services Cloud d'hébergement (Azure Compute) et de stockage (Azure Storage) 	<ul style="list-style-type: none"> - Gmail - Google Documents - Google Groupes - Google Agenda - Google Sites - Google Vidéos - Google Talk 																							
Petite Instance (par défaut)	Grande Instance	Très grande Instance																																											
Stockage RAM 1,7 GO	RAM 7,5 GO	RAM 15 GO																																											
Stockage 160 GO	Stockage 850 GO	Stockage 1690 GO																																											
Plateforme	Plateforme	Plateforme																																											
32 GO	64 GO	64 GO																																											
Coût	<table border="1"> <tr> <td>256 MO</td> <td>0,014\$ par heure</td> </tr> <tr> <td>512 MO</td> <td>0,028\$ par heure</td> </tr> <tr> <td>768 MO</td> <td>0,042\$ par heure</td> </tr> <tr> <td>1224 MO</td> <td>0,057\$ par heure</td> </tr> <tr> <td>1280 MO</td> <td>0,071\$ par heure</td> </tr> <tr> <td>1536 MO</td> <td>0,085\$ par heure</td> </tr> <tr> <td>1792 MO</td> <td>0,099\$ par heure</td> </tr> <tr> <td>2048 MO</td> <td>0,11\$ par heure</td> </tr> </table>	256 MO	0,014\$ par heure	512 MO	0,028\$ par heure	768 MO	0,042\$ par heure	1224 MO	0,057\$ par heure	1280 MO	0,071\$ par heure	1536 MO	0,085\$ par heure	1792 MO	0,099\$ par heure	2048 MO	0,11\$ par heure	<table border="1"> <tr> <td>Windows</td> <td>Windows</td> <td>Windows</td> </tr> <tr> <td>0,12\$ par heure</td> <td>0,48\$ par heure</td> <td>0,96\$ par heure</td> </tr> <tr> <td>Linux 0,085\$ par heure</td> <td>Linux 0,34\$ par heure</td> <td>Linux 0,68\$ par heure</td> </tr> </table>	Windows	Windows	Windows	0,12\$ par heure	0,48\$ par heure	0,96\$ par heure	Linux 0,085\$ par heure	Linux 0,34\$ par heure	Linux 0,68\$ par heure	<table border="1"> <tr> <td>Edition gratuite</td> <td>Edition d'entreprise</td> <td>Edition illimitée</td> </tr> <tr> <td>Gratuit</td> <td>75,0438\$ par mois</td> <td>111,176\$ par mois</td> </tr> </table>	Edition gratuite	Edition d'entreprise	Edition illimitée	Gratuit	75,0438\$ par mois	111,176\$ par mois	Gratuit si pas de dépassement de pourcentage (stockage < 1 GO, taille du code d'une application : 150 MO)	<table border="1"> <tr> <td>Compute</td> <td>Web Edition</td> </tr> <tr> <td>0.12\$ par mois</td> <td>(1 GO) : 9,99\$</td> </tr> <tr> <td>Storage</td> <td>Business Edition</td> </tr> <tr> <td>0.15\$ par mois</td> <td>(10 GO) : 99,99\$</td> </tr> </table>	Compute	Web Edition	0.12\$ par mois	(1 GO) : 9,99\$	Storage	Business Edition	0.15\$ par mois	(10 GO) : 99,99\$	Gratuit sauf pour google Apps for business 40\$ par personne et par an
256 MO	0,014\$ par heure																																												
512 MO	0,028\$ par heure																																												
768 MO	0,042\$ par heure																																												
1224 MO	0,057\$ par heure																																												
1280 MO	0,071\$ par heure																																												
1536 MO	0,085\$ par heure																																												
1792 MO	0,099\$ par heure																																												
2048 MO	0,11\$ par heure																																												
Windows	Windows	Windows																																											
0,12\$ par heure	0,48\$ par heure	0,96\$ par heure																																											
Linux 0,085\$ par heure	Linux 0,34\$ par heure	Linux 0,68\$ par heure																																											
Edition gratuite	Edition d'entreprise	Edition illimitée																																											
Gratuit	75,0438\$ par mois	111,176\$ par mois																																											
Compute	Web Edition																																												
0.12\$ par mois	(1 GO) : 9,99\$																																												
Storage	Business Edition																																												
0.15\$ par mois	(10 GO) : 99,99\$																																												
Fournisseur	Société OVH	Amazon	Salesforce.com	Google	Microsoft	Google																																							
Sécurité	Authentification	Chaque fichier stocké sera identifié par une clé	- Certification ISO 27001 - Hébergement redondant et répliqués des données	Authentification	Contrôle d'accès et gestion des utilisateurs	Authentification																																							
Langage de programmation	Java	Java, Python	Langage Apex (Supporte C#, .NET, Java, C++)	Java, Python	.NET, Ruby, Python, Java ou PHP	supporte Java, Python																																							

Des acteurs issus du Web, fort de leur expérience en environnement distribué ont décidé d'ouvrir ces savoir faire en proposant leur propre offre du *Cloud Computing*.

Parmi ces acteurs nous trouvons :

- **Amazon** : avec son offre *Amazon Web Services* et plus précisément son produit IaaS *Amazon Elastic Compute Cloud (EC2)* qui permet le déploiement de machines virtuelles chez Amazon afin de tirer partie de leurs capacités de traitement.
- **Salesforce.com** : avec sa plateforme *Force.com* qui permet d'accéder à un large catalogue d'applications SaaS tels que les applications de stocks, iPhone, iPad, Android, etc.
- **Google** : un acteur bien connu à destination à la fois des particuliers, à titre gratuit, et des entreprises. Avec son offre *App Engine*, il fournit une plateforme qui s'adresse aux développeurs en leur proposant un ensemble de services (hébergement, base de données, etc.) pour pouvoir faire tourner leurs applications sur les infrastructures de Google.
Google propose aussi *Google Apps*, une offre SaaS pour les applications en ligne (Gmail, Google Calendar, Google Talk, Google Documents, Google Sites, etc.).
- **Microsoft** : avec son offre Azure qui permet de développer et déployer des applications serveur, accessibles par Internet. Azure est constitué d'un ensemble de services hébergés dans les Datacenter de Microsoft. Cela comprend notamment : Windows Azure (infrastructure de calcul, de stockage, de réseau, etc.) et SQL Services (base de données en ligne).

1.6 Cloud Computing et techniques de virtualisation

Dans le cadre des offres du *Cloud Computing*, les techniques de virtualisation sont très présentes. Le tableau 1.5 donne un aperçu sur les techniques de virtualisation des différents Clouds open-sources et propriétaires déjà cités.

TABLE 1.5 – Cloud Computing et techniques de virtualisation utilisées

	Hyperviseur	Xen	KVM	VirtualBox	VMware	HyperV
Open-source	Outils	✓ version => 3.0.x (Plus utilisé) Debian, CentOS, OpenSUSE	✓(virtualisation en-abled)		✓ Pour la solution commerciale	✓
	<i>OpenNebula</i>	✓	✓(virtualisation en-abled)	Prévu	✓Libvirt 0.8.3 re-commandé	
	<i>Nimbus</i>	✓ La plupart des déploiements actuels utilisent Xen	✓(virtualisation en-abled)			
	Xen Cloud Platform	✓ Evolution de XenServer				
	<i>AbiCloud</i>	✓ <i>AbiCloud</i> 0.8.0 et 0.7.0 => avec libvirt 0.6.5 (non en-core testé)	✓ <i>AbiCloud</i> 0.8.0 et 0.7.0 => avec libvirt 0.6.5	✓ <i>AbiCloud</i> 0.8.0 et 0.7.0 => Virtual-box 2.2.x <i>AbiCloud</i> 0.6.0 => Virtual-box 2.1.x		
	<i>OpenStack</i>	✓	✓ (virtualisation enabled)	✓Récentement		
	Amazon Web Services	✓				
	Google Apps				✓	
	Microsoft Azure					✓
	Google App Engine				✓	
Force.com				✓		
OVH				✓		

Comme le montre le tableau 1.5, Xen et KVM s'avèrent les solutions les plus utilisées pour les offres open-sources du *Cloud Computing*. Cependant, VMware est clairement la solution la plus présente dans le domaine de la virtualisation de Cloud propriétaires. De ce fait, VMware est également considéré le premier fournisseur en infrastructures du *Cloud Computing*.

1.7 Conclusion

Nous avons présenté dans ce chapitre une idée générale sur le *Cloud Computing*, son architecture et ses différents services. Nous avons mené par la suite une étude sur les alternatives open-sources des plateformes du *Cloud Computing* ainsi que les techniques de virtualisation utilisées. Ceci nous permet de choisir la solution avec laquelle nous créons notre propre environnement du Cloud.

2.1 Introduction

Comme déjà cité dans le premier chapitre, le *Cloud Computing* comporte trois niveaux de service : Infrastructure comme service, Plateforme comme service et Software comme service. Nous détaillons dans un premier temps, ses services. Comme la qualité de service devient un facteur de plus en plus important pour les clients de service, il serait intéressant qu'elle soit mesurée et maintenue. Ainsi, nous réalisons dans un second temps, une étude des environnements de mesure et de test que nous utilisons par la suite dans le *Cloud Computing*.

2.2 Les couches du Cloud Computing

Une fois nous disposons de la couche infrastructure du *Cloud Computing*, il sera important de passer aux autres couches. Ainsi, la couche plateforme nous permet de bénéficier d'un Framework de développement, d'où la possibilité de développer des outils spécifiques pour notre activité. Dans la couche software, nous fournissons des services informatiques par le biais du Web. On parle donc des services Web dans le Cloud.

La figure 2.1 résume l'architecture générale du *Cloud Computing*.

De manière générale, l'architecture du *Cloud Computing* peut être divisée en quatre couches, comme indiqué dans la figure 2.1 : la couche matériels, la couche infrastructure, la couche plateforme et la couche application.

- **La couche matériels** : elle est responsable de la gestion des ressources physiques (serveurs, routeurs, commutateurs, etc). La couche matérielle est géné-

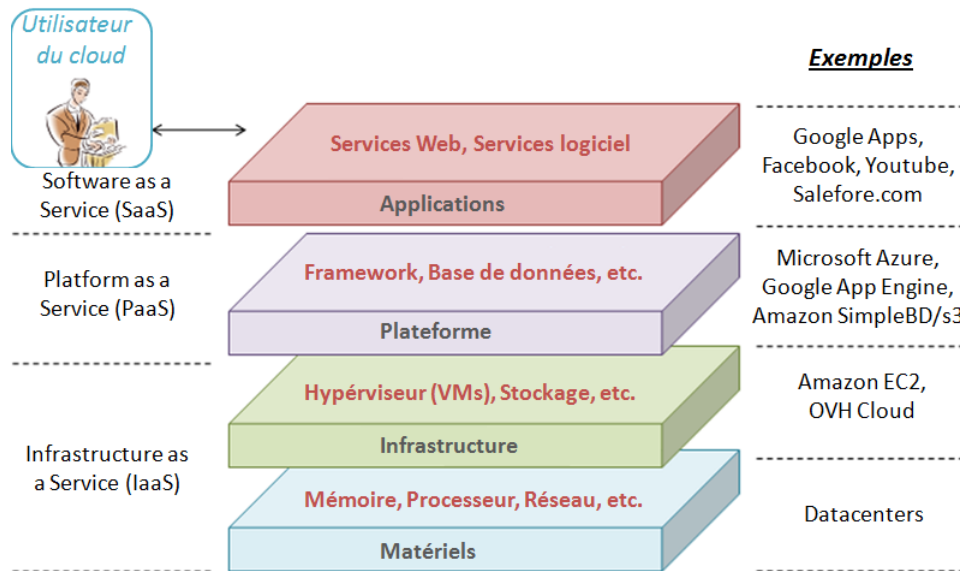


FIGURE 2.1 – Architecture générale du Cloud Computing

ralement mise en œuvre dans des Datacenter.

- **La couche infrastructure** : c'est un composant essentiel du *Cloud Computing*. Elle permet de créer un pool de stockage par le partitionnement des ressources physiques en utilisant la technique de virtualisation.
- **La couche plateforme** : elle comporte le SE ainsi qu'un ensemble de Framework d'application. Son but est de minimiser la charge du déploiement d'applications. En effet le déploiement sera effectué dans des machines virtuelles.
- **La couche applications** : elle est située plus haut niveau de la hiérarchie et elle comporte les applications du Cloud.

2.3 Les services Web

Dans ce qui suit nous donnons un aperçu sur les services Web tout en présentant les différentes techniques inhérentes.

2.3.1 Définition

Le groupe de travail du W3C [26] a défini le service web comme étant "un système logiciel identifié par une URI dont les interfaces publiques et les associations sont définies et décrites en XML. Sa définition peut être découverte par d'autres systèmes logiciels. Ces systèmes peuvent alors interagir avec le service web selon les modalités indiquées dans sa définition, en utilisant des messages XML transmis par des protocoles internet" [30].

Afin d'utiliser un service Web, ce dernier doit passer par un ensemble d'étapes qui constituent en fait son cycle de vie.

2.3.2 Cycle de vie d'un service Web

Un service Web suit un ensemble d'étapes pour être mis en place et utilisé par un client final. La première étape consiste à déployer le service Web dans un conteneur adéquat. Dans la deuxième étape, le service Web sera enregistré dans l'annuaire de service (UDDI) et décrit par le langage WSDL pour avoir enfin un service Web prêt à l'utilisation.

Lorsqu'un client veut demander un service Web, il démarre par le chercher dans l'annuaire UDDI. Dès qu'il découvre le service Web adéquat, il commence par comprendre le contrat WSDL imposé par le serveur et finalement, il l'invoque. Le serveur Web retourne ainsi la réponse de la requête au client.

Le schéma de la figure 2.2 résume le cycle de vie d'un service Web.

Dans ce qui suit, nous détaillons les concepts associés aux services Web, à savoir le SOAP, le WSDL et l'UDDI.

2.3.3 Les concepts associés

a. SOAP

SOAP est l'abréviation de "*Simple Object Access Protocol*". Il s'agit d'un protocole léger permettant l'échange d'information dans des environnements distribués [30]. Il est basé sur XML.

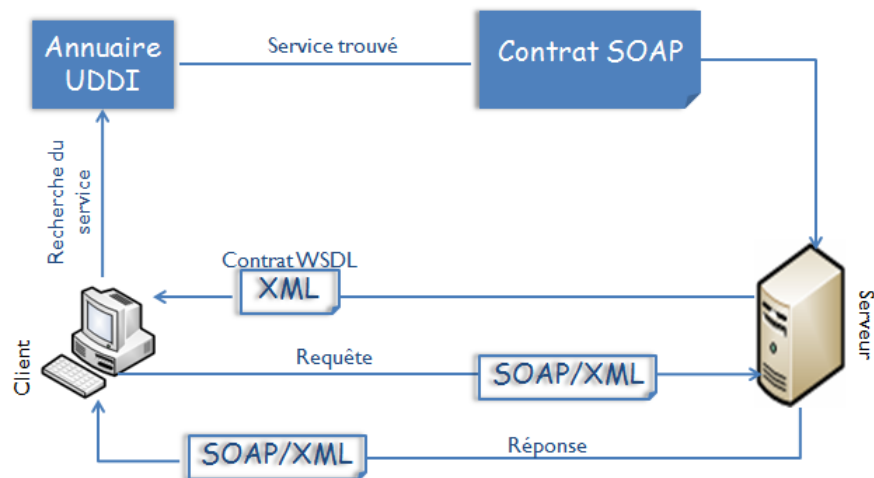


FIGURE 2.2 – Cycle de vie d'un service Web

b. WSDL

WSDL est l'abréviation de *Web Services Description Language*. Comme son nom l'indique, c'est un langage de description des services Web.

Il s'agit d'un format XML dédié à la représentation des services par un ensemble d'opérations et de messages abstraits reliés à des protocoles et des serveurs réseaux [30].

c. UDDI

UDDI est l'abréviation de *Universal Description Discovery and Integration*. C'est un annuaire de services fondé sur XML et plus particulièrement destiné aux services Web. UDDI est une spécification mise au point par l'OASIS [30]. Il permet de localiser sur le réseau le service Web recherché.

2.4 Performance des services Web

Plusieurs définitions de la performance des services Web existent dans la littérature. Nous utilisons la définition fournie par le groupe du W3C [26] comme une fondation pour notre propre définition. D'après le W3C [26], la performance est définie en termes de débit, temps de réponse, temps de latence et temps d'exécution.

Dans le cadre de ce mastère, nous définissons la performance en termes de temps

de réponse et ses sous concepts (temps de communication et temps d'exécution).

- Le temps de réponse est le temps nécessaire pour traiter une requête dès l'instant de son envoi jusqu'au moment de sa réception. Le temps de réponse peut être divisé en temps de communication et temps d'exécution.
- Le temps de communication est le temps nécessaire pour le transport de la requête et de sa réponse.
- Le temps d'exécution mesure le temps mis par le serveur pour exécuter la requête.

La figure 2.3 montre une invocation simple d'un service.

Nous optons pour cette architecture afin de réaliser nos expérimentations sur le *Cloud Computing*.

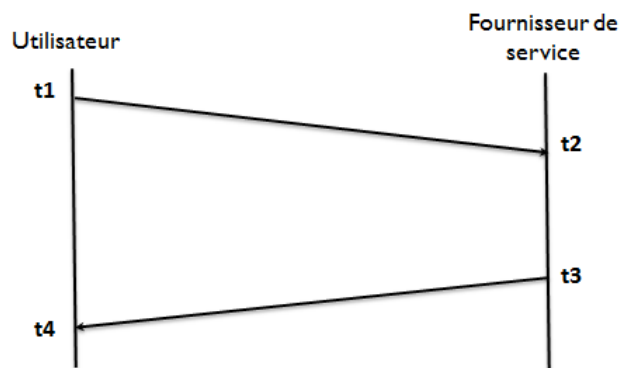


FIGURE 2.3 – Architecture de mesure de la QoS

- t1 : temps d'envoi de requête du client
- t2 : temps de réception de la requête par le service
- t3 : temps d'envoi de la réponse
- t4 : temps de réception de la requête

Ainsi :

- $(t2-t1)+(t4-t3)$: temps de communication
- $t2-t3$: temps d'exécution

2.5 Techniques de mesure de la QoS

Nous présentons dans cette partie les deux techniques que nous utilisons à savoir le *Timer* et l'orienté aspect pour mesurer la qualité de service ainsi que d'autres techniques existantes.

2.5.1 Le Timer inséré dans le code

Une méthode simple pour la mesure des caractéristiques de performance des services Web peut être développée en ajoutant des fonctionnalités dans le code client du service Web [38]. Ces fonctionnalités sont insérées avant et après l'invocation du service Web.

2.5.2 Utilisation de l'approche orientée aspect

Le code de mesure de performance peut être implémenté en utilisant la Programmation Orientée Aspect (POA). Celle-ci est un paradigme de programmation qui permet de réduire fortement les couplages entre les différents aspects techniques d'un logiciel. L'aspect définit des points d'action qui représentent les points de jonction satisfaisants aux conditions d'activation de l'aspect responsable de la mesure de performance. Par exemple, le temps de réponse est mesuré en définissant un point de mesure qui détecte le temps avant et après l'invocation d'une méthode du service.

AOP4CSM [39] est une approche de monitoring des services du Cloud. Elle se base sur la programmation orientée aspect (AOP). AOP4CSM [39] permet la collecte et la mesure des paramètres de performance des services du Cloud tels que le temps de réponse et le temps de communication, ceci étant sans avoir à modifier le code source du service et du client [39]. Nous avons tout d'abord téléchargé l'outil AOP4CSM [39] depuis son site Web [4]. Ensuite, nous avons déployé AOP4CSM [39] au sein de la plateforme du *Cloud Computing* que nous avons mise en place.

2.5.3 Autres techniques

Approche du proxy

Dans cette approche, un proxy est utilisé comme médiateur de communication entre le client et le serveur. Les messages échangés entre le client et le serveur seront donc visibles par le proxy et les attributs de performance seront mesurés par ce proxy [38].

Modification de la bibliothèque du SOAP

Dans cette approche, la bibliothèque du parsing du message SOAP est modifiée pour enregistrer l'information nécessaire pour la mesure de performance [38]. Ces informations sont envoyées par la suite à une troisième entité responsable du stockage et de la mise à jour des mesures.

La modification de bibliothèque a besoin d'être disponible sur les différentes implémentations et plateformes. C'est une modification qui doit être établie notamment dans les bibliothèques SOAP des clients ainsi que des fournisseurs.

Approche basée sur le monitoring de paquets

L'idée principale derrière cette approche est de capturer les paquets des messages SOAP entrants et sortants [38]. Cette approche utilise particulièrement les paramètres de la couche transport pour dériver les métriques et les mesures de la performance.

2.6 Architecture de mesure de la qualité de service

La mesure de la qualité de service d'un service Web consiste à calculer en fonction des valeurs récupérées et de nombre de requêtes invoquées un ou plusieurs attributs de QoS.

Pour mesurer et évaluer le temps de réponse du service Web, nous avons adopté l'architecture présentée par la figure 2.4 [4] dans un environnement du *Cloud Computing*. Cette architecture est basée sur l'approche de monitoring AOP4CSM.

Il faut tout d'abord noter, qu'en programmation orientée aspect :

- Un *pointcut* est un point de coupure qui donne les positions dans le programme ou va être introduit le code fonctionnel de l'aspect. Pour cela le pointcut va définir un type d'action, principalement l'appel d'une méthode.
- Un *advice* associé à un *pointcut* le code devant être exécuté. On peut sélectionner différentes position pour l'exécution du code de l'*advice* : avant ou après le *pointcut*.

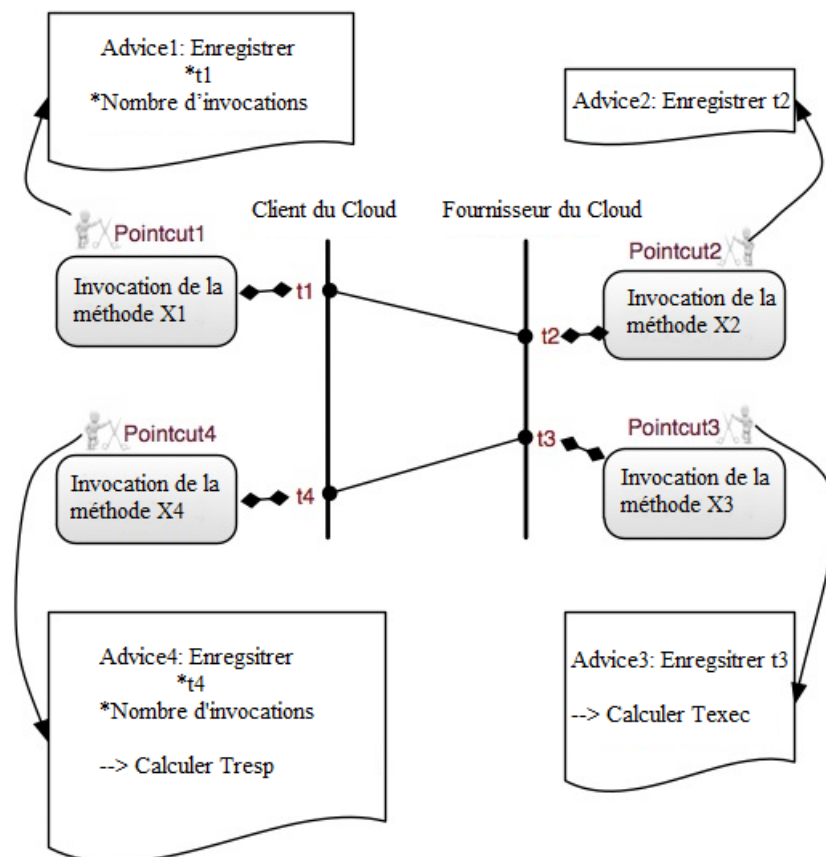


FIGURE 2.4 – Architecture de déploiement d'AOP4CSM

Entre le client du Cloud et le serveur, AOP4CSM met en place quatre points de mesure (*pointcut*), deux côté client et deux côté serveur, dans le but de mesurer et évaluer les paramètres de la QoS.

Le client envoie une requête pour invoquer un service Web déployé sur un serveur. Le pointcut côté client invoque la méthode X1 à l'instant t1 (le temps d'invocation du service par le client). Ce dernier sera enregistré par l'aspect. Par la suite, le

pointcut côté serveur invoque la méthode X2 à l'instant t2 (temps de réception de la requête par le fournisseur de service) et de même t2 sera enregistré par l'aspect. La même démarche est faite pour la réponse envoyée par le serveur.

L'étape suivante est la mesure du temps d'exécution et du temps de communication en se basant sur l'information calculée par l'aspect. Cette étape est assurée, d'une part, par le pointcut3 côté serveur qui permet de calculer le temps d'exécution du service Web et d'autre part, par le pointcut4 côté client qui calcule le temps de communication de l'invocation du service Web. Par la suite, l'aspect exécute un thread qui envoie les paramètres de la QdS mesurés à un gestionnaire de logging. Ce dernier est un service Web qui permet d'enregistrer les données dans une base de données MySQL.

Ainsi :

- Le Temps d'exécution : le temps que le service a pris pour exécuter la requête.

$$\text{Texec} = t3 - t2$$

- Le Temps de réponse : Temps d'exécution + Temps de communication.

$$\text{Tresp} = t4 - t1$$

2.7 Conclusion

Notre architecture de mesure de la qualité de service dans le Cloud est basée sur les aspects. Ces derniers assurent la collection d'informations des côtés client et fournisseur de service du Cloud. Ceci permet de garantir plus de précision lors de la mesure de la QdS. Une fois l'architecture est prête, il reste à passer à l'étape de validation en effectuant des expérimentations à grande échelle tout en suivant l'évolution et le changement de la QdS.

Dans le chapitre suivant, nous détaillerons notre démarche pour établir les expérimentations de la mesure de la QdS.

Expérimentations et validation d'AOP4CSM sur le Cloud

3.1 Introduction

Après avoir présenté l'approche de mesure adoptée, nous passons dans ce chapitre à détailler la procédure d'expérimentation que nous avons adoptée pour valider nos travaux. Ainsi, ce chapitre s'articule autour de deux axes principaux. Dans la première partie, nous détaillons les étapes de mise en place de l'architecture définie. Quant à la deuxième partie, nous nous intéressons à présenter les résultats de nos expérimentations tout en donnant nos interprétations.

3.2 Mise en place d'AOP4CSM

Comme déjà mentionné dans le chapitre précédent, notre architecture du *Cloud Computing* se compose de trois couches. Dans cette section, nous présentons les étapes de mise en place de chaque couche.

3.2.1 Couche infrastructure

Pour la couche infrastructure, nous avons choisi de mettre en place un Cloud open-source de type IaaS. L'étude comparative des outils open-sources du *Cloud Computing* faite au premier chapitre, nous a permis de choisir de mettre en place le Cloud avec *Eucalyptus*. En effet, c'est la solution la plus répandue du *Cloud Computing* qui est intégrée dans les distributions *Ubuntu* en tant qu'outil du *Cloud Computing*. Son avantage majeur est sa compatibilité avec Amazon EC2.

Eucalyptus

Eucalyptus ne fonctionne qu’avec des serveurs sous Linux, équipés de l’hyperviseur Xen, KVM ou HyperV pour les solutions open-sources et il est intégré dans les versions récentes d’*Ubuntu* à partir de la version 9.04.

Pour mettre en place une infrastructure minimale du *Cloud Computing*, nous aurons besoin d’au moins deux machines :

- o Front end
- o Nœud

Ainsi, afin de construire notre Cloud privé, il nous faut à disposition au minimum :

- Deux ordinateurs personnels ;
- Un serveur : Eucalyptus fonctionne uniquement avec des serveurs type Linux et plus précisément *Debian Squeeze*, *Fedora*, *OpenSuse*, *CentOS* et *Ubuntu*. Puisque nos machines ne supportent pas la virtualisation matérielle, nous avons utilisé *Debian Squeeze* ;
- Quatre paquets : *eucalyptus-cc* (*Cluster Controller*), *eucalyptus-sc* (*Storage Controller*), *eucalyptus-walrus* (interfaçage avec *Amazon Web Services*) et *eucalyptus-cloud* (*Cloud Controller*) pour le front end ;
- Le paquet *eucalyptus-nc* (*Node Controller*) et un pont réseau (bridge) pour le nœud ;
- Un hyperviseur pour la création et la gestion des machines virtuelles qui sera installé sur la machine nœud. Le meilleur choix pour l’hyperviseur dépend du support du matériel. Nos machines ne supportent pas la virtualisation matérielle, ainsi nous avons choisi d’utiliser l’hyperviseur *Xen*.

La figure 3.1 décrit les différents composants d’*Eucalyptus*. Nous disposons d’un serveur *Debian Squeeze* sur chacune des machines *front end* et *node*.

Pour des contraintes de temps et de matériels, nous avons utilisé pour la validation et l’expérimentation le Cloud privé d’OVH. En effet, nous visons à effectuer des expérimentations à grande échelle, ce qui ne peut pas être fait avec deux machines. D’autre part, il nous faut des machines plus puissantes avec une configuration comme c’est décrit par le tableau 3.1. Nous notons que pour la mise en place de notre

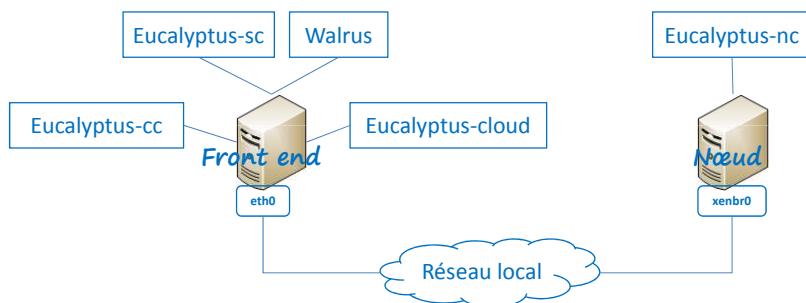


FIGURE 3.1 – Composants d’Eucalyptus

Cloud privé avec *Eucalyptus*, nous avons utilisé des machines avec une configuration minimale selon ce qui est décrit dans le tableau 3.1.

TABLE 3.1 – Configuration matérielle nécessaire pour la mise en place d’un Cloud avec Eucalyptus

Matériel	Configuration minimale		Configuration suggérée	
	Front end	Node	Front end	Node
CPU	1GHz	Supporte la technologie de virtualisation (VT)	2 x 2GHz (Dual core)	Supporte la technologie de virtualisation (VT), 64-bit, Multi-core
Mémoire	2GO	1GO	4GO	1GO
Disque	5400rpm IDE	5400rpm IDE	7200rpm SATA	7200rpm SATA ou SCSI
Espace disque	40GO	40GO	200GO	100GO

OVH Cloud Computing

OVH vient d’annoncer une offre du Cloud privé, basée sur vSphere et vCenter de VMware. Elle consiste en une solution d’infrastructure à la demande (*Infrastructure as a Service*, ou IaaS), et permet d’externaliser les coûts liés aux équipements IT de l’entreprise, en les faisant baisser de manière conséquente.

3.2.2 Couche plateforme

Pour la couche plateforme, nous avons utilisé :

- Apache Tomcat comme serveur Web ;

- Axis comme implémentation de service Web ;
- Java comme langage de programmation ;
- MySQL comme système de gestion de base de données.

3.2.3 Couche software

Pour mettre en œuvre l'architecture proposée dans la figure 2.4, nous l'avons appliquée sur une application distribuée : la Revue Coopérative qui est une publication périodique (hebdomadaire, mensuelle, etc.) publiant des articles et des travaux de recherche pour des spécialistes dans divers domaines (scientifiques, littéraires, juridiques, économiques, etc.). Il s'agit d'un système de gestion de conférences développé dans notre équipe *ReDCAD* et permet l'automatisation des différentes phases de la revue coopérative. Comme notre but étant d'observer les valeurs des paramètres de qualité de service, nous avons choisi de mesurer la QoS du système de gestion de conférences basé sur les services Web et déployé sur le Cloud. Nous nous sommes intéressés principalement de la mesure de la QoS pour le scénario "recherche de conférence". Ce dernier permet au client de recevoir la liste de toutes les conférences répondant aux critères de recherche déjà définis.

3.3 Environnement de déploiement

Afin de lancer un nombre assez important de clients et être le plus proche du contexte grande échelle, nous avons choisi d'effectuer les mesures de la QoS sur le MiniCloud d'OVH.

3.3.1 MiniCloud d'OVH

MiniCloud est un ensemble de briques intégrant la virtualisation sur des machines tournant sous *Linux*, le stockage de données de référence basé sur Solaris et bien entendu avec toutes les fonctionnalités de sauvegarde à distance et agencées par une base de données MySQL.

3.3.2 Configuration et déploiement

Pour réaliser les mesures de paramètres de la QdS du service "recherche de conférences", nous avons utilisé plusieurs nœuds fonctionnant sous MiniCloud OVH et comportant le système d’exploitation *Linux Ubuntu*.

La figure 3.2 schématise notre architecture d’expérimentation à grande échelle.

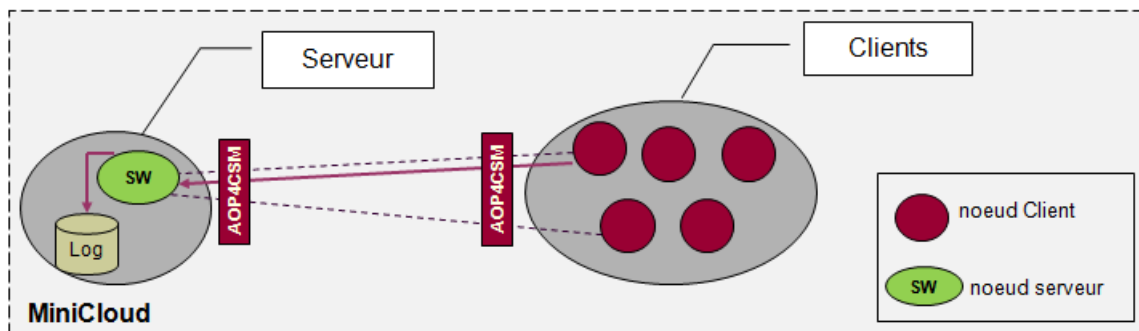


FIGURE 3.2 – Architecture de déploiement sur le Cloud

Comme le montre la figure 3.2, nous avons utilisé six nœuds. Le premier nœud servira de serveur Web, il comporte notre base de données MySQL ainsi que notre service Web de recherche de conférences. Les cinq autres nœuds hébergent un ou plusieurs clients qui invoquent plusieurs fois et simultanément le service Web cible. Pour chaque requête d’un client, l’aspect envoie une requête d’insertion à la base de données contenant des valeurs de la QdS.

La configuration matérielle et logicielle de chacune de ces machines est résumée dans le tableau 3.2.

TABLE 3.2 – Configuration matérielle et logicielle

Composant	Configuration matérielle			Configuration logicielle
Serveur Web	Processeur	RAM	Disque	Ubuntu server 10.04
	8 GHz	1025 Mo	5 Go	
Client Web	Processeur	RAM	Disque	Ubuntu server 10.04
	8 GHz	512 Mo	5 Go	

3.4 Expérimentations d'AOP4CSM

Afin de mesurer la qualité de service de notre service Web de recherche de conférences, nous avons réalisé un ensemble d'expérimentations sur le *Cloud Computing*. En utilisant des scripts shell Linux, nous avons pu exécuter plusieurs clients en même temps et chaque client envoie de multiples requêtes pendant une durée de 60 secondes. Ainsi, chaque client établit une connexion avec le service Web et envoie des requêtes SOAP au service "recherche de conférences". Les expérimentations sont répétées plusieurs fois et les valeurs obtenues sont stockées dans une base de données MySQL pour pouvoir calculer les paramètres de la QoS et dresser des courbes d'évaluation.

Dans la suite de cette section, nous présentons les résultats de nos expérimentations.

3.4.1 Résultats expérimentaux

Nous précisons que pour nos expérimentations, nous avons exécuté respectivement 1, 5, 10, 25, 50, 75 et 100 clients. Chaque client invoque le service Web continuellement pendant une minute.

Le tableau 3.3 montre l'évolution de la qualité de service en fonction du nombre de clients.

TABLE 3.3 – Les mesures de la QoS

Nombre de clients	Temps de réponse (ms)			Temps d'exécution (ms)			Temps de communication (ms)			Nombre total de requêtes	Nombre de requêtes réussites
	Min	Max	Moyenne	Min	Max	Moyenne	Min	Max	Moyenne		
1 client	72	122	99,25	60	111	88	11	12	11,25	294	284
5 clients	102	117	110	90	104	97,25	12	13	12,75	1208	1194
10 clients	213	225	218	197	219	204	13	21	14	1249	1201
20 clients	476	540	504,75	458	521	482,25	18	29	22,5	1577	1123
25 clients	613	787	664,5	597	741	640	15	30	24,5	2026	1067
50 clients	1043	1376	1244,25	1021	1350	1219,25	22	27	25	2613	1189
75 clients	1865	2472	2170,5	1838	2439	2142,5	26	33	28	3182	1304
100 clients	2024	2532	2274,25	2001	2503	2245,25	23	39	29	3721	1212

L'augmentation du nombre de clients engendre une dégradation de la QoS. Cette dégradation est bien visible sur le temps de réponse ainsi que le temps d'exécution. En effet, la charge sur le serveur devient lourde et la performance baisse.

Pour illustrer davantage ces résultats, nous représentons l'évolution du temps de réponse dans la figure 3.3 ainsi que le temps d'exécution dans la figure 3.4 et de communication dans la figure 3.5 par rapport au nombre de clients. Ainsi, pour un nombre de clients inférieur à 5, l'augmentation des temps de réponse et d'exécution est légère. Cependant, pour un nombre de clients variant entre 25 et 75, l'augmentation devient de plus en plus importante. A partir de 75 clients jusqu'à 100 clients, la variation se stabilise.

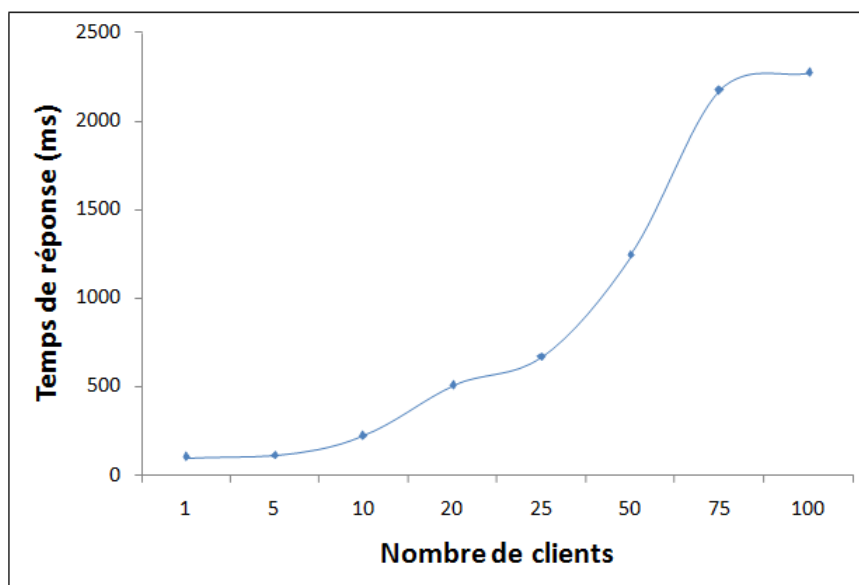


FIGURE 3.3 – Evolution du temps de réponse

Nous avons enregistré, d'une part, un nombre d'exceptions levées et donc un nombre de requêtes important qui a échoué et d'autre part, un temps de réponse qui ne cesse d'augmenter. Nous avons dressé par la suite la courbe qui montre les proportions des requêtes servies en fonction du nombre de clients (voir figure 3.6). Comme le montre cette figure, l'augmentation du nombre de clients engendre une diminution du nombre de requêtes servies. Cette faiblesse est due à plusieurs aspects à savoir la capacité de traitement du fournisseur et l'efficacité de l'accès de la base

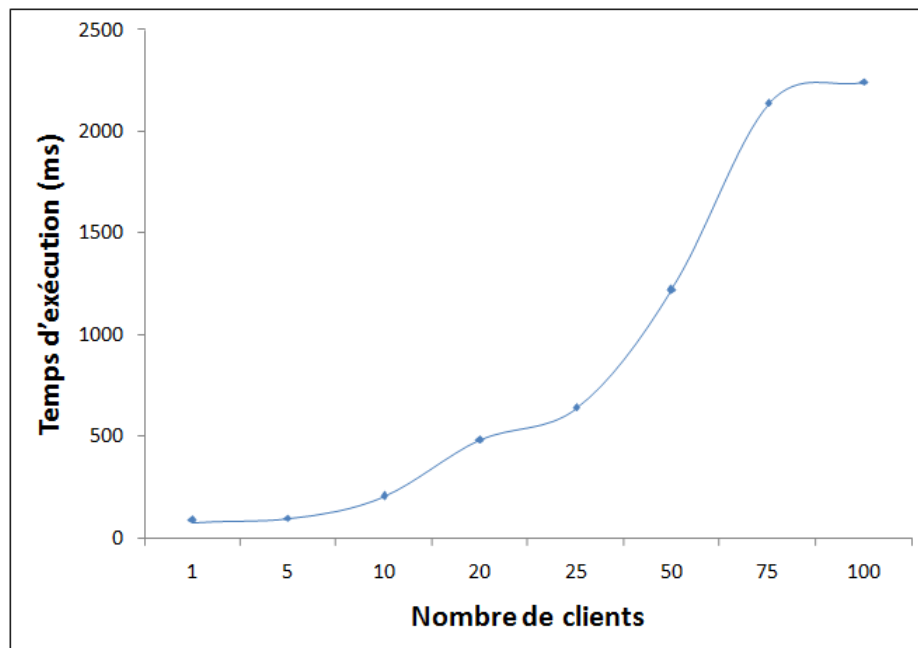


FIGURE 3.4 – Evolution du temps d'exécution

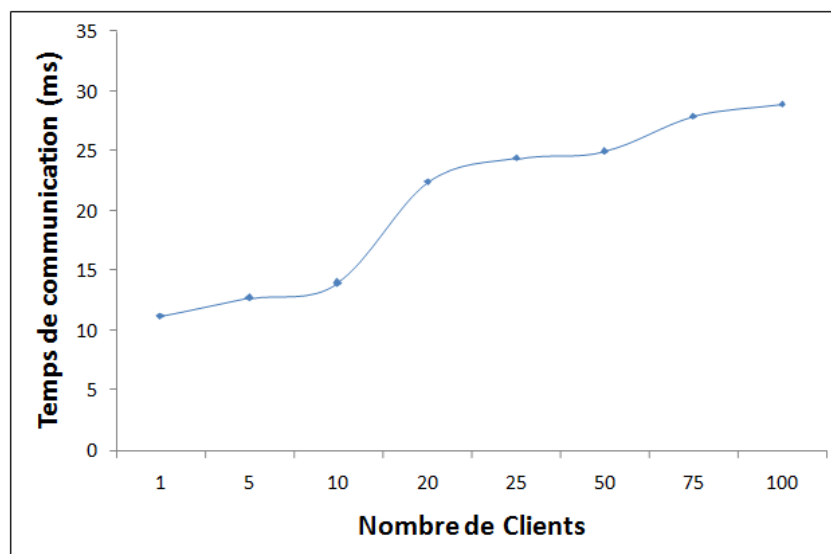


FIGURE 3.5 – Evolution du temps de communication

de données.

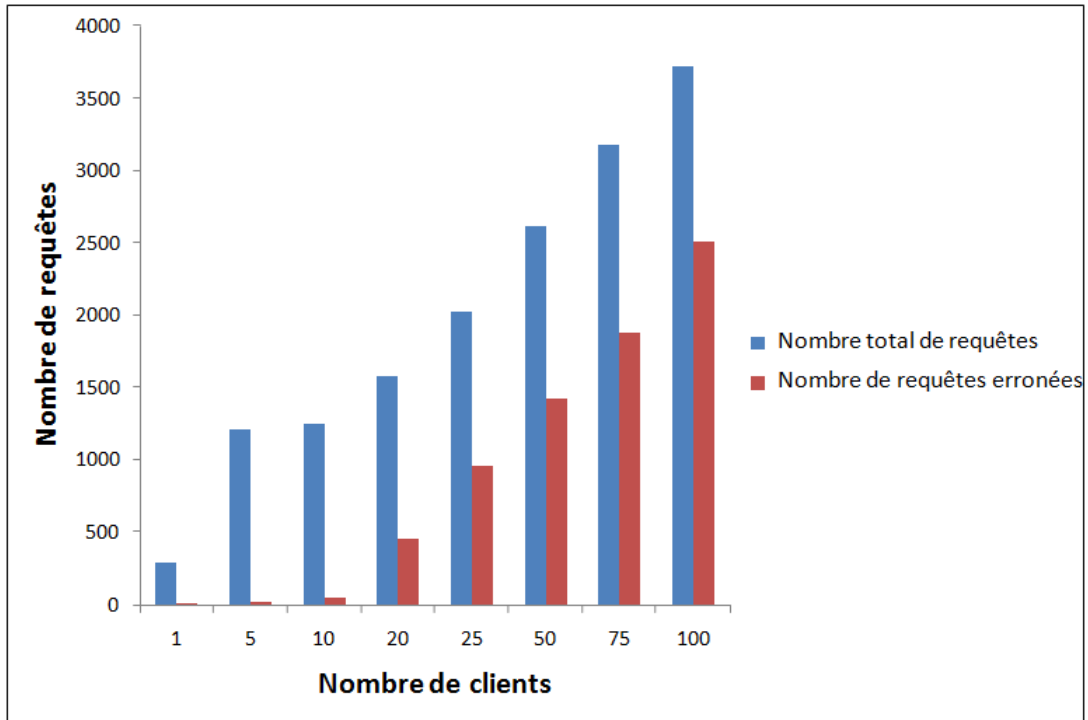


FIGURE 3.6 – Représentation des requêtes servies en fonction du nombre de clients

Nous avons remarqué au cours des expérimentations, que la plupart des requêtes erronées sont du au type d'exception : "connection refused" qui signifie que la capacité du serveur a être dépassée. Cependant, à partir de 25 clients, le nombre de requêtes erronées devient de plus en plus remarquable et cela est expliqué par le fait que le nombre de requêtes envoyées au serveur a dépassé la valeur maximale des connexions simultanées permises.

Nous reportons sur la figure 3.7 l'évolution de la disponibilité du service Web en fonction du nombre de clients. Les résultats ont montré que pour un nombre de clients inférieur à 10 il n'y avait pas trop de réponses erronées pendant la période de test et donc le service est toujours disponible. Cependant, nous avons remarqué qu'à peut près 70% des requêtes ne pourraient pas être accompli par le service Web pour 100 clients simultanées.

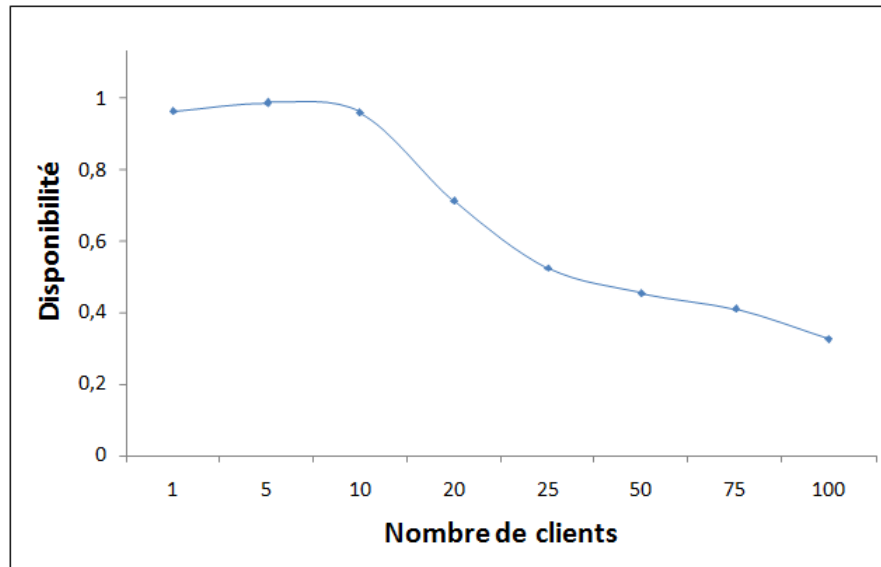


FIGURE 3.7 – Mesure de la disponibilité

3.4.2 Surcoût d’AOP4CSM

Les résultats expérimentaux déjà présentés montrent bien l'évolution du service Web dans le Cloud. En effet, notre but était de calculer le surcoût d’AOP4CSM [39]. Pour ce faire, nous avons réservé le même nombre de nœuds avec lesquels nous avons effectué les expérimentations précédentes mais, cette fois-ci, nous avons utilisé l'approche de monitoring avec *Timer* (Voir section 3.4 de ce chapitre) dans le service Web ainsi que dans le client. Nous avons d'autre part, utilisé le même programme client qui invoque le même service, à savoir, le service de "recherche de conférences". Nous avons obtenu ainsi, les résultats montrés dans le tableau 3.4.

Afin de bien étudier ces résultats, nous avons tracé les courbes de mesure du temps de réponse pour chacun des deux cas. Dans le premier cas, la mesure est accomplie avec l'approche de monitoring AOP4CSM [39]. Dans le deuxième cas, la mesure est réalisée dans le code du client en intégrant des *Timers* et sans utilisation de l'approche AOP4CSM [39].

Comme le montre la figure 3.8, l'utilisation d’AOP4CSM [39] n'a pas engendré de coût supplémentaire pour un nombre de clients inférieur à 20. Cependant, au-delà de 20 clients la mesure basée sur l'approche de monitoring AOP4CSM [39] a donné

TABLE 3.4 – Comparaison des résultats avec et sans AOP4CSM

Nombre de client	Temps de réponse avec AOP4CSM (ms)	Temps de réponse sans AOP4CSM (ms)
1 client	99,25	86,5
5 clients	110	100,25
10 clients	218	193,75
20 clients	504,75	389,75
25 clients	664,5	579,5
50 clients	1244,25	974,5
75 clients	2170,5	1879,5
100 clients	2274,25	2060

un coût supplémentaire négligeable pour le temps de réponse (X secondes).

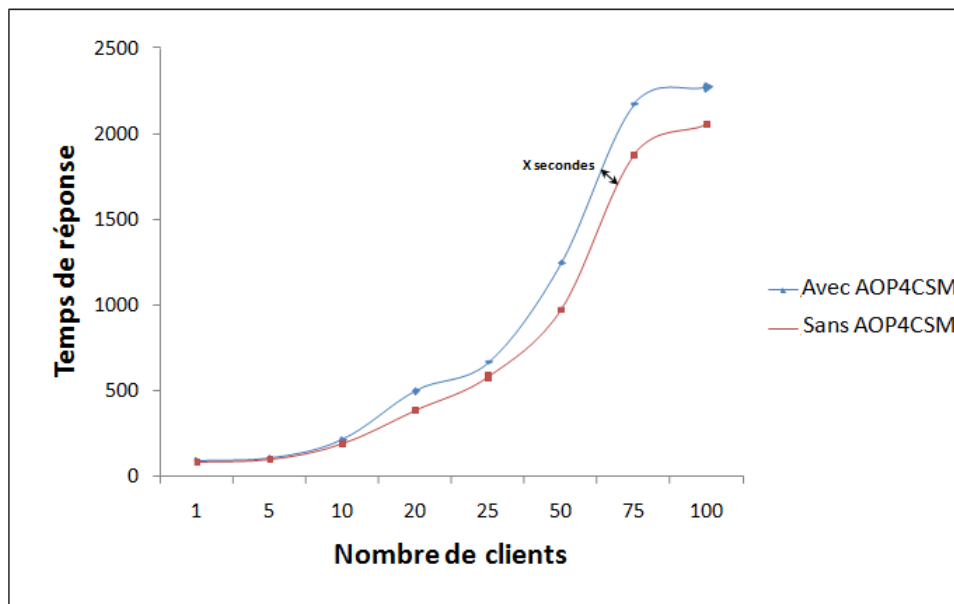


FIGURE 3.8 – Evolution du temps de réponse avec et sans AOP4CSM

3.4.3 Synthèse

L'ensemble des expérimentations déjà présentées dans ce chapitre nous a permis de :

- * Avoir une idée riche et maîtriser l'environnement du *Cloud Computing* ;
- * Expérimenter AOP4CSM à grande échelle ;
- * Montrer l'efficacité d'AOP4CSM. En effet, l'utilisation d'AOP4CSM n'a pas engendré un coût supplémentaire important pour le temps de réponse ;
- * Extraire différentes courbes qui vont être exploitées par la suite comme référence soit pour l'analyse de l'état du service, soit pour la sélection du service Web.

3.5 Conclusion

Dans ce chapitre, nous avons mis en place notre architecture de monitoring sur le niveau IaaS du Cloud. La mesure et l'évolution de la qualité de service sont validées à travers une expérimentation à grande échelle. A travers cet ensemble d'expérimentations, nous avons pu dégager des valeurs moyennes de QoS pour le service de recherche de conférence. A base de ces mesures nous avons pu générer des courbes de référence.

Conclusion générale et perspectives

En conclusion, le travail de recherche effectuée dans le cadre de ce mastère aborde principalement le monitoring des applications de la couche IaaS du *Cloud Computing*. Nous avons donné une idée générale sur le *Cloud Computing*, son architecture et ses différents services. Nous avons fait par la suite une étude comparative des différentes solutions open source et propriétaires du Cloud tout en précisant les techniques de virtualisation utilisés dans chacune d'entre elles. Ceci nous a permis d'avoir une idée riche sur les techniques de virtualisation ainsi que les différentes solutions disponibles du *Cloud Computing* et surtout de bien maîtriser le concept du Cloud. Nous avons ensuite opté pour une solution commerciale pour mettre en place notre propre environnement du *Cloud Computing* à savoir le MiniCloud d'OVH vu les contraintes matérielles.

La principale motivation qui a régi ce travail est d'une part la mesure de la qualité de service des services Web et d'autre part la validation de l'approche AOP4CSM [39] dans un environnement *Cloud Computing* pour enfin montrer son efficacité et générer des courbes de référence. Pour ce faire, nous avons tout d'abord choisi notre environnement de déploiement dans le Cloud. Nous avons passé par la suite à l'étape de validation en effectuant des expérimentations à grande échelle sur un service d'une application à base de services Web (la Revue Coopérative) et plus précisément le scénario "Recherche de conférences". Les résultats des expérimentations ont été enregistrés dans une base de données afin de pouvoir les exploiter par la suite pour effectuer les calculs de la qualité de service. A base de ces mesures, nous avons généré des courbes de référence et nous avons montré que l'approche AOP4CSM [39] ajoute un coût supplémentaire négligeable à la QoS.

Certes, nos expérimentations permettent la mesure de la QoS dans un environ-

nement du *Cloud Computing*. Ainsi, nous pourrions étendre notre travail tout en implémentant une architecture pour le diagnostic et la réparation d'applications basées sur les services Web.

A court terme et en se basant sur l'étude technique réalisée sur le *Cloud Computing*, nous pourrions mettre en place un environnement Cloud si les contraintes matérielles seront relaxées. Ceci nous permettra l'utilisation de notre propre environnement pour mener à terme nos expérimentations à tous les niveaux du Cloud.

Bibliographie

- [1] Abicloud. <http://community.abiquo.com/>.
- [2] Amazon web srrvice. <http://aws.amazon.com/fr/>.
- [3] Ambitwire. <http://ambitwire.com/>.
- [4] Aop4csm. <http://www.redcad.org/members/mdhaffar/aop4csm/>.
- [5] Cloud comuting consulting. <http://www.cloudconsulting.fr/index.php/fr/le-cloud-computing/iaas-paas-et-saas>.
- [6] Comprendre le cloud computing. <http://pro.clubic.com/it-business/cloud-computing/article-376690-4-cloud-computing.html>.
- [7] Datacenterknowledge. <http://www.datacenterknowledge.com/>.
- [8] Eucalyptus. <http://www.eucalyptus.com/>.
- [9] Fedora. <http://doc.fedora-fr.org/wiki/Virtualisation>.
- [10] Google app engine. <http://code.google.com/intl/fr-FR/appengine/>.
- [11] Google apps. <http://www.google.com/apps/>.
- [12] Kvm. <http://www.linux-kvm.org/>.
- [13] Microsoft. <http://www.microsoft.com/>.
- [14] Microsoft hyperv.
<http://www.microsoft.com/hyper-v-server/en/us/default.aspx>.
- [15] My saas.
<http://mysaas.fr/2010/10/04/private-cloud-publique-cloud-et-hybrid-cloud/>.
- [16] Netcraft. <http://news.netcraft.com/>.
- [17] Nimbus. <http://www.nimbusproject.org/>.

- [18] Opennebula. <http://opennebula.org/>.
- [19] Openstack. <http://www.openstack.org/>.
- [20] Openvz.
<http://www.prolibre.com/documentation/virtualisation/openvz.html>.
- [21] Ovh. <https://www.ovh.com/fr/index.xml>.
- [22] Salesforce. <http://www.salesforce.com/fr/>.
- [23] Tnr global. <http://www.tnrglobal.com/blog/2009/05/migrating-an-openvz-virtual-machine>.
- [24] Virtualbox. <http://www.virtualbox.org/>.
- [25] Vmware vsphere. <http://www.arumtec.net/fr/outils-virtualisation/outils-de-virtualisation/vmware-vsphere-4.1/presentation-de-vmware-vsphere-4.1>.
- [26] W3c. <http://www.w3.org/>.
- [27] Xen.
<http://aldevar.free.fr/data/VeilleTechno/VeilleTechno-Virtualisation.pdf>.
- [28] Xen. <http://www.xen.org/>.
- [29] Xen. <http://wiki.xen.org/>.
- [30] *Understanding Web Services : XML, WSDL, SOAP, and UDDI*.
Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.
- [31] J. Kelner . T. Endo, G. E. Goncalves and D. Sadok. A survey on open-source cloud computing solutions. *Brazilian Symposium on Computer Networks and Distributed Systems*, May 2010.
- [32] Peter Baer. Vmware vsphere vs. microsoft hyper-v : A technical analysis a cti strategy. November 2009.
- [33] Tom Bellwood, Luc Clément, David Ehnebuske, Andrew Hately, Maryann Hondo, Yin Leng Husband, Karsten Januszewski, Sam Lee, Barbara McKee, Joel Munter, and Claus von Riegen. Uddi version 3.0. juillet 2002.
- [34] Ignacio M. Llorente Borja Sotomayor, Ruben S. Montero and Ian Foster.

- [35] Jianhua Che, Qinming He, Qinghua Gao, and Dawei Huang. Performance measuring and comparing of virtual machine monitors. In *Proceedings of the 2008 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing - Volume 02*, EUC '08, pages 381–386, Washington, DC, USA, 2008. IEEE Computer Society.
- [36] Jianhua Che, Congcong Shi, Yong Yu, and Weimin Lin. A synthetical performance evaluation of openvz, xen and kvm. In *Proceedings of the 2010 IEEE Asia-Pacific Services Computing Conference*, APSCC '10, pages 587–594, Washington, DC, USA, 2010. IEEE Computer Society.
- [37] Thiago Damasceno Cordeiro, Douglas Brito Damalio, Nadilma Cintra Valenca Nunes Pereira, Patricia Takako Endo, Andre Vitor de Almeida Palhares, Glauco Estacio Goncalves, Djamel Fawzi Hadj Sadok, Judith Kelner, Bob Melander, Victor Souza, and Jan-Erik Mangs. Open source cloud computing platforms. In *Proceedings of the 2010 Ninth International Conference on Grid and Cloud Computing*, GCC '10, pages 366–371, Washington, DC, USA, 2010. IEEE Computer Society.
- [38] Riadh Ben Halima. *Conception, implantation et expérimentation d'une architecture en bus pour l'auto-réparation des applications distribuées à base de services web*. PhD thesis, 2009.
- [39] Afef Mdhaffar, Riadh Ben Halima, Ernst Juhnke, Mohamed Jmaiel, and Bernd Freisleben. AOP4CSM : An Aspect-Oriented Programming Approach for Cloud Service Monitoring. In *Proceedings of the 11th IEEE International Conference on Computer and Information Technology*, 2011.
- [40] Junjie Peng, Xuejun Zhang, Zhou Lei, Bofeng Zhang, Wu Zhang, and Qing Li. Comparison of several cloud computing platforms. In *Proceedings of the 2009 Second International Symposium on Information Science and Engineering*, ISISE '09, pages 23–27, Washington, DC, USA, 2009. IEEE Computer Society.
- [41] Marvin Rambhadjan and Arthur Schutijser. *SURFnet cloud computing solutions*.

- [42] Bhaskar Prasad Rimal, Eunmi Choi, and Ian Lumb. A taxonomy and survey of cloud computing systems. In *Proceedings of the 2009 Fifth International Joint Conference on INC, IMS and IDC*, NCM '09, pages 44–51, Washington, DC, USA, 2009. IEEE Computer Society.
- [43] Peter Sempolinski and Douglas Thain. A comparison and critique of eucalyptus, opennebula and nimbus. In *Proceedings of the 2010 IEEE Second International Conference on Cloud Computing Technology and Science*, CLOUDCOM '10, pages 417–426, Washington, DC, USA, 2010. IEEE Computer Society.

Annexe

.1 Etapes d'installation d'Eucalyptus

.1.1 Etape 1 : Préparation du front end

Pré requis :

- Vérifier que les locales ¹ sont correctement configurés.

```
# dpkg-reconfigure locales
```

- Les ports 8443, 8773, 8774 et 9001 doivent être ouverts.

Installation

- Les paquets eucalyptus-cc, eucalyptus-cloud, eucalyptus-nc et walrus sont disponibles à partir du dépôt de Debian. Pour les installer avec leurs dépendances, il faut ajouter le dépôt d'Eucalyptus au fichier `/etc/apt/sources.list` :

```
# vim /etc/apt/sources.list
# deb http ://eucalyptussoftware.com/downloads/repo/eucalyptus/$VERSION/
debian/ squeeze main
# apt-get update
```

- Installation d'Eucalyptus sur le front end.

```
# aptitude install eucalyptus-common eucalyptus-cloud eucalyptus-walrus
eucalyptus-sc eucalyptus-cc
```

1. Les paramètres régionaux permettent d'afficher les données selon les attentes culturelles et linguistiques propres à la langue et au pays de l'utilisateur

- Activer les services du front end.

```
# /etc/init.d/eucalyptus-cc start # /etc/init.d/eucalyptus-cloud start
```

.1.2 Etape 2 : Préparation du nœud

Pré requis :

- Vérifier que les locales sont correctement configurés.

```
# dpkg-reconfigure locales
```

Le node controller est entièrement installé et configuré sur un hyperviseur de type Xen qui permettra de contrôler les machines virtuelles. Ainsi, il faut tout d'abord installer Xen sur le nœud.

* Dom0 (machine hôte)

- Commençons par installer l'hyperviseur : noyau xen et xen-tools.

```
# aptitude -P install xen-hypervisor-4.0-amd64 linux-image-xen-amd64
```

- Modifier le grub de telle façon que xen devient le système par défaut.

```
# mv -i /etc/grub.d/10_linux /etc/grub.d/50_linux # update-grub2
```

- Désactiver l'OS prober² pour ne pas mettre à jour le menu grub lorsqu'on installe des machines virtuelles.

2. Paquet qui détecte les systèmes installés sur une machine. Lors de la mise à jour, de l'installation ou de la suppression d'un noyau, ce paquet mettra à jour le menu de grub pour les autres systèmes.

```
# echo "" /etc/default/grub
# echo "# Disable OS prober to prevent virtual machines on logical volumes from
appearing in the boot menu." /etc/default/grub
# echo "GRUB_DISABLE_OS_PROBER=true" /etc/default/grub
# update-grub2
```

- Redémarrer la machine.

Les modifications apportées à Xen sont effectuées sur le fichier `xend-config.sxp`

```
# vim /etc/xen/xend-config.sxp
(xend-unix-server yes)
(network-script network-bridge)
# /etc/init.d/xend restart
```

* **DomU (machine invitée)**

Nous pouvons également utiliser l'outil `xen-tools` permettant l'installation facile des machines virtuelles.

```
# apt-get install xen-tools
```

Pour configurer `xen-tools`, il faut éditer le fichier `/etc/xen-tools/xen-tools.conf` qui contient les valeurs nécessaire pour utiliser le script `xen-createimage`.

```
lvm = universe
dir = /home/xen
install-method = debootstrap
size = 4Gb # Disk image size.
memory = 256Mb # Memory size
swap = 1Gb # Swap size
fs = ext3 # use the EXT3 filesystem for the disk image.
dist = 'xt-guess-suite-and-mirror -suite' # Default distribution to install.
dhcp = 1
passwd = 1
mirror_maverick = http://nl.archive.ubuntu.com/ubuntu/
ext3_options = defaults
pygrub=1
gateway = 192.168.1.1
netmask = 255.255.255.0
broadcast = 192.168.1.255
mirror_lenny = http://cdn.debian.net/debian
ext3_options = defaults
pygrub=1
gateway = 192.168.1.1
netmask = 255.255.255.0
broadcast = 192.168.1.255
```

- Créer une ou plusieurs machines virtuelles :

```
# xen-create-image --hostname <hostname> --dist lenny
```

- Le port 8775 doit être ouvert.

Installation :

- Ouvrir Xen
- Le paquet eucalyptus-nc est disponible à partir du dépôt. Pour l'installer avec ses dépendances, il faut ajouter le dépôt d'Eucalyptus au fichier `/etc/apt/sources.list` :

```
# vim /etc/apt/sources.list Deb http ://eucalyptussoft-  
ware.com/downloads/repo/eucalyptus/$VERSION/debian/ squeeze main #  
apt-get update
```

- Installation d'Eucalyptus sur le nœud.

```
# aptitude install open-iscsi libcrypt-openssl-random-perl libcrypt-openssl-rsa-perl  
libcrypt-x509-perl eucalyptus-nc
```

La dernière étape d'installation d'Eucalyptus est de s'assurer que le nœud est configuré pour interagir avec l'hyperviseur par l'intermédiaire de libvirt. Ainsi, sur le nœud, nous devons vérifier que le fichier `/etc/libvirt/libvirtd.conf` contient les lignes suivantes :

```
unix_sock_group = "libvirt"  
unix_sock_ro_perms = "0777"  
unix_sock_rw_perms = "0770"  
auth_unix_ro = "none"  
auth_unix_rw = "none"
```

- Redémarrer libvirtd.

```
# /etc/init.d/libvirt-bin stop  
# /etc/init.d/libvirt-bin start  
# chown root :libvirt /var/run/libvirt/libvirt-sock  
# chown root :libvirt /var/run/libvirt/libvirt-sock-ro
```

- Activer le service node controller.

```
# /etc/init.d/eucalyptus-nc start
```

.1.3 Etape 3 : Enregistrement et intégration des composants d'Eucalyptus

Sur la machine front end :

```
# /usr/sbin/euca_conf -register-walrus (front end IP address)
# /usr/sbin/euca_conf -register-cluster (clustername) (front end IP address)
# /usr/sbin/euca_conf -register-sc (clustername) (front end IP address)
# /usr/sbin/euca_conf -register-nodes (node IP address)
```

.1.4 Etape 4 : Utilisation d'Eucalyptus

Se connecter au Cloud à travers l'URL [https ://<ip front end> :8443/](https://<ip front end>:8443/)

- Se connecter en tant qu'administrateur : Username = admin, Password = admin.
- Obtenir les certificats à partir de l'onglet *Credentials*.
- Enregistrer les certificats dans la machine front end.

```
# mkdir ~/.euca
# cd ~/.euca
# unzip (nom du fichier zip).zip
# chmod 0700 ~/.euca
# chmod 0600 ~/.euca/*
# . eucarc
```

Afin de gérer Eucalyptus (gestion des images, des instances, du stockage, du réseau, etc.), nous avons utilisé l'outil euca2ools.

```
# vim /etc/apt/sources.list
deb http://www.eucalyptussoftware.com/downloads/repo/euca2ools/1.3.1/debian
squeeze main
# aptitude update
# aptitude install euca2ools python-boto
```



Étude et expérimentations du Cloud Computing pour le monitoring des applications orientées services

Meriam MAHJOUB

الخلاصة:

تمثل سحابة الحوسبة بيئة برمجيات قوية ومرنة، والتي تمثل إدارة المواد، وتدفع وفقاً للاستهلاك. هجرة تطبيقات المؤسسات على هذه البيئة في تزايد مستمر. الكثير من هذه التطبيقات متاحة في شكل خدمات ويب والتي توحد الوصول إلى منطق الأعمال عبر الإنترنت. بالرغم من ذلك، لا يمكن اعتبارها فعالة إلا إذا استجابت في الوقت لطلبات الزبائن. بالتالي، يعتبر أي ارتفاع في زمن الإجابة من خدمة ويب تدهوراً للأداء أو حتى فشلاً. إذاً فإن رصد جودة الخدمة يمثل خطوة مهمة لتلبية حاجيات الزبائن. الهدف من هذا العمل هو التجربة على بيئة سحابة الحوسبة منهج *AOP4CSM*. هو منهج لرصد جودة خدمات الويب على بيئة سحابة الحوسبة والذي تم تصميمه في فريق بحثنا. هذا العمل يمكن من (1) فهم بيئة سحابة الحوسبة، (2) إظهار جدوى *AOP4CSM*، (3) تجربته على نطاق واسع واختبار متانته، (4) إنشاء رسوم بيانية لتستخدم كمرجع لجودة الخدمة التي يمكن استعمالها سواء للتحليل أو لتحديد خدمة الويب.

المفاتيح: سحابة الحوسبة، خدمات الويب، جودة الخدمات، رصد، *AOP4CSM*، رسوم بيانية مرجعية

Résumé

Le *Cloud Computing* est un environnement logiciel puissant et flexible qui délègue la gestion du matériel, et qu'on paie selon la consommation. La migration des applications d'entreprise sur cet environnement ne cesse de s'accroître. Une grande partie de ces applications est offerte sous forme de services Web, qui standardisent l'accès à la logique métier via Internet. Cependant, ils ne peuvent être considérés comme efficaces que s'ils répondent à temps aux contraintes des utilisateurs finaux tels que le temps de réponse. Ainsi, toute augmentation du temps de réponse d'un service Web est considérée comme une dégradation de ses performances, voire même une panne. Le *monitoring* de la qualité de service (QoS) constitue donc une étape primordiale pour la satisfaction des besoins des clients. L'objectif de ce travail est d'expérimenter sur un environnement de *Cloud Computing* l'approche *AOP4CSM*. Il s'agit d'une approche de *monitoring* de la QoS des services Web du Cloud qui est conçue au sein de notre équipe de recherche. Le travail réalisé permet de (1) appréhender l'environnement du *Cloud Computing*, (2) montrer la faisabilité de *AOP4CSM*, (3) de l'expérimenter à large échelle et de tester sa robustesse, (4) générer des courbes de référence de la QoS qui peuvent être exploitées soit pour l'analyse de l'état, soit pour la sélection du service Web.

Mots clés: *Cloud Computing*, Services Web, Qualité de Service, *Monitoring*, *AOP4CSM*, courbes de référence

Abstract:

Cloud computing is a powerful and flexible software environment, which delegates the material's management and you pay as you go. The migration of enterprise applications on the Cloud is increasing. The majority of these applications are available as Web services that standardize access to business logic via the Internet. However, their efficiency depends on the rapidity of their answer to users constraints such as end-user response time. Thus, every increase in the response time of a Web service is considered as a degradation of its performances or even failure. As a result, monitoring quality of service (QoS) is a key feature in order to satisfy the client requirement. This project main objective is to experiment on Cloud Computing environments the *AOP4CSM* approach. It is an approach which monitors the QoS of Web services in the Cloud and it is designed in our research team. Our work allows (1) understanding the environment of the Cloud, (2) demonstrating the feasibility of *AOP4CSM*, (3) experimenting *AOP4CSM* on a large scale and test its robustness, (4) generating reference QoS curves that can be used either for the analysis or the selection of the Web service.

Key-words: Cloud Computing, Web Service, Quality of Service, *Monitoring*, *AOP4CSM*, reference curves