REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université A.MIRA-BÉJAÏA

Faculté d'Électronique - Département d'Informatique

Mémoire

Présenté par : OUADI Sabira

Pour l'obtention du diplôme de Magistère

Filière : Informatique

Option : Cloud Computing

Thème :

Une fédération multi-cloud dans les environnements orientés *« grid services »*

Soutenu, le 04/10/2016, devant le jury composé de :

Nom et Prénom	**Grade**		
Mr BOUKERRAM Abdellah	Pr	UNIV. de Béjaia	Président
Mr TARI Abdelkamel	MCA	UNIV. de Béjaia	Rapporteur
Mr SLIMANI Hachem	MCA.	UNIV. de Béjaia	Examinateur
Mme EL-MAOUHAB Aouaouche	CR	CERIST	Invitée

Année Universitaire : 2015/2016

Remerciements

Je remercie en premier notre grand Dieu pour m'avoir donné le courage et la volonté pour terminer ce modeste travail.

J'exprime toute ma gratitude à mon directeur de thèse, Mr TARI Abdelkamel, Maître de Conférence A à l'Université de Béjaia de m'avoir fait confiance en acceptant d'encadrer ce travail.

Mes remerciements s'adressent aussi à Mme Aouaouache EL-MAOUHAB, chargée de recherche au CERIST, pour ses encouragements durant cette longue épreuve et pour son infinie patience.

Je remercie : Mr. BOUKERRAM Abdellah et Mr. SLIMANI Hachem, pour avoir accepté de juger ce modeste travail.

Un grand MERCI à Mes Parents, Mon Mari, Ma Soeur et Son Mari et Mes deux filles Meriem et Kenza pour m'avoir soutenu à fin que ce travail arrive à sa fin.

Je remercie mes amis pour leur aide appréciable, leurs encouragements continus et leur soutien moral ininterrompu.

J'exprime ma gratitude à touts mes collègues du CERIST, pour m'avoir poussé à faire le magister, encouragé, motivé. Et surtout Linda, Faiza et Amine qui m'ont aidé de manière bénévole et spontanée.

Et pour être sûr de n'oublier personne, que tous ceux, qui de près ou de loin, ont contribué par leurs conseils, leurs encouragements ou leur amitié, à l'aboutissement de ce modeste travail, trouvent ici l'expression de ma profonde reconnaissance.

Résumé

Avec l'augmentation du nombre de fournisseurs de services Cloud et la migration des applications depuis les grilles de calcul vers le Cloud, il est nécessaire de pouvoir tirer parti de ces nouvelles ressources. De plus, une large classe d'applications de calcul haute performance peuvent s'exécuter sur ces ressources sans modification (ou avec des modifications mineures). Ainsi le déploiement de ces applications dans un environnement *multi-cloud* évite la dépendance vis-à-vis des fournisseurs (en anglais : vendor lock-in). Dans cette thèse, nous introduisons une nouvelle plateforme *d'interopérabilité* aux services *IaaS* nommée *« DZ-CBroker »*. Cette plateforme permet d'utiliser les ressources qui proviennent de plusieurs Clouds *(multi-cloud)* comme des ressources classiques. De plus, elle permet de cacher à l'utilisateur final la difficulté et la complexité de sélectionner et d'utiliser ces nouvelles ressources lors de la création de nouvelles machines virtuelles. Finalement, nous procédons à la validation de notre approche par une implémentation.

Mots-clés : Multi-cloud, Interopérabilité, Grille de calcul

Abstract

With the increasing number of Cloud Services Providers and migration of applications from grid computing to the Cloud, it is necessary to be able to leverage these new resources. Moreover, a large class of High Performance Computing (HPC) applications can run these resources without (or with minor) modifications. And the deployment of these applications in a multi-cloud environment avoids dependence on a single cloud provider (vendor lock-in). In this thesis we introduce a new interoperability platform for IaaS named *DZ-CBroker*. This platform allows the use of the resources that come from an environment composed of multiple Clouds as well as classical resources. In addition, it allows to hide to the end user the difficulty and complexity of selecting and using these new resources when creating new virtuals machines. Finally, we validate our approach by an implementation.

Keywords : Multi-cloud, Interoperability, Grid Computing

Table des matières

Résumé ii

Abstract iii

Introduction générale xi

I Contexte et État de l'art 1

1 Grille de Calcul 2
- 1.1 Introduction . . . 2
- 1.2 Définition . . . 2
- 1.3 Classement et taxonomie . . . 3
  - 1.3.1 Grille d'information : . . . 3
  - 1.3.2 Grille de stockage : . . . 3
  - 1.3.3 Grille de calcul . . . 3
- 1.4 Les propriétés des grilles de calcul . . . 4
- 1.5 Architecture d'une grille de calcul . . . 4
  - 1.5.1 La couche réseau . . . 5
  - 1.5.2 La couche ressources . . . 5
  - 1.5.3 La couche Intergiciel . . . 5
  - 1.5.4 La couche application . . . 6
- 1.6 Éléments constitutif d'une grille de calcul . . . 6
- 1.7 Chemins d'un job . . . 7
- 1.8 Domaines d'application des grilles de calcul . . . 8
  - 1.8.1 Les sciences de la physique . . . 8
  - 1.8.2 Le biomédical . . . 9
  - 1.8.3 L'observation de la terre et climatologie . . . 9
  - 1.8.4 L'astronomie . . . 9
- 1.9 Conclusion . . . 9

2 Cloud Computing **11**
2.1 Introduction . . . 11
2.2 Historiques et définitions . . . 11
2.3 Caractéristiques de Cloud Computing . . . 12
2.4 Les Services du Cloud . . . 13
2.5 Les Modèles de déploiement du Cloud Computing . . . 14
2.6 Les éléments constitutifs du Cloud Computing . . . 15
2.7 la différence entre la virtualisation et le Cloud Computing . . . 17
2.8 Les Acteurs du Cloud Computing . . . 17
2.9 Les plates-formes cloud IaaS "Open-source" . . . 18
2.9.1 OpenNebula . . . 18
2.9.2 OpenStack . . . 20
2.9.3 Eucalyptus . . . 22
2.9.4 Synthèse . . . 24
2.10 Grid Computing et Cloud Computing . . . 25
2.10.1 L'intégration des grilles et des Clouds . . . 25
2.11 Conclusion . . . 26

3 Contexte Multi-Cloud **28**
3.1 Introduction . . . 28
3.2 L'interopérabilité dans le multi-cloud . . . 28
3.3 Le Cloud Broker . . . 29
3.3.1 Définition . . . 29
3.3.2 Scénario d'utilisation du Cloud Broker . . . 29
3.3.3 Les services d'un Cloud Broker . . . 29
3.3.4 Les fonctionnalités d'un cloud broker . . . 30
3.4 L'Inter-Cloud . . . 30
3.4.1 Les avantages d'un environnement Inter-Cloud pour les clients 30
3.4.2 Les avantages d'un environnement Inter-Cloud pour les fournisseurs . . . 31
3.5 L'Inter-Cloud broker . . . 32
3.6 Les taxonomies multi-clouds . . . 32
3.6.1 Taxonomie 1 . . . 33
3.6.2 Taxonomie 2 . . . 35
3.6.3 Discussion . . . 36
3.7 Solutions multi-clouds . . . 37
3.7.1 Les standards . . . 37
3.7.2 Vue d'ensemble des solutions existantes multi-Cloud . . . 39
3.7.3 Solutions d'abstraction (Librairies) . . . 44
3.8 Conclusion . . . 46

II Contribution **48**

4 Conception **49**
4.1 Introduction . . . 49
4.2 Problématique et méthodologie . . . 49
4.2.1 Problématique . . . 49
4.2.2 Objectifs . . . 50
4.3 Schéma Global proposé . . . 51
4.4 Architecture détaillée de la plateforme DZ-CBroker . . . 51
4.4.1 User (Utilisateurs) . . . 52
4.4.2 Grille de calcul . . . 52
4.4.3 User Interface . . . 53
4.4.4 La plateforme DZ-CBroker . . . 54
4.4.5 Plates-forme multi-cloud . . . 61
4.5 Diagramme de cas d'utilisation . . . 61
4.6 Diagrammes de séquence . . . 61
4.6.1 Diagramme de séquence d'authentification . . . 62
4.6.2 Diagramme de séquence de traitement d'une requête d'enregistrement d'une Image-grille . . . 63
4.6.3 Diagramme de séquence de traitement d'une requête d'approvisionnement d'une VM . . . 64
4.6.4 Diagramme de séquence de gestion d'une file d'attente . . . 65
4.7 Conclusion . . . 66

5 Mise en œuvre **68**
5.1 Introduction . . . 68
5.2 Environnements de développement . . . 68
5.2.1 Le choix des plateformes clouds . . . 68
5.2.2 Les API d'interopérabilité choisis . . . 68
5.2.3 La plateforme de développement . . . 69
5.2.4 Le serveur web . . . 69
5.3 Schéma général de la mise en œuvre . . . 70
5.3.1 Diagramme des composants du système proposé et leurs interactions . . . 72
5.3.2 Expérimentation et évaluation . . . 79
5.4 Conclusion . . . 83

Discussion et perspectives **86**

Conclusion générale **86**

Bibliographie **95**

A Interface utilisateur de la plateforme DZ-CBroker **97**
A.1 Authentification 97
A.2 Grid Manager interface 98
A.3 User grid interface 102

Table des figures

1.1 Architecture d'une grille de calcul 5
1.2 Infrastructure des services de la grille 7

2.1 Différents services du Cloud Computing 13
2.2 L'architecture de référence du Cloud Computing de l'organisme NIST 17
2.3 Architecture OpenNebula 18
2.4 Architecture OpenNebula 19
2.5 Architecture OpenNebula 20
2.6 Architecture d'OpenStack 21
2.7 Architecture OpenNebula 22
2.8 Architecture d'Eucalyptus 23

3.1 Scénario d'utilisation du cloud broker 29
3.2 Classification architecturale des Inter-Cloud 33
3.3 Fédération inter-cloud centralisé 34
3.4 Fédération inter-cloud pair à pair 34
3.5 Service Multi-Cloud 35
3.6 Librairie Multi-cloud 35
3.7 Relation entre les catégories multi-clouds 36
3.8 Architecture de gestion de sites RESEROIR 39
3.9 Architecture du CONTRAIL 40
3.10 Architecture du STRATOS 41
3.11 Architecture de CompatibleOne 42
3.12 Architecture d'OPTIMIS 43

4.1 Architecture globale de la solution 51
4.2 Architecture détaillée du *DZ-CBroker* 52
4.3 Exemple d'une Appliance Virtuelle 53
4.4 Modèle de configuration de type d'instance (Flavor) 55
4.5 Model de configuration d'une Template 55
4.6 Modèle de configuration *Manifest* 56

4.7 Algorithme de placement d'une machine virtuelle . . . 58
4.8 Processus de filtrage . . . 58
4.9 Algorithme de gestion de la file d'attente . . . 60
4.10 Cycle de vie d'une machine virtuelle . . . 60
4.11 Diagramme de cas d'utilisation . . . 61
4.12 Diagramme de séquence d'authentification . . . 62
4.13 Diagramme de séquence de traitement d'une requête d'enregistrement d'une Image-grille . . . 63
4.14 Diagramme de séquence de traitement d'une requête d'approvisionnement d'une VM . . . 64
4.15 Diagramme de séquence de gestion d'une file d'attente . . . 65

5.1 Schéma général de la mise en œuvre . . . 71
5.2 Diagramme des composants de système proposé et leurs interactions . . . 73
5.3 Fonctionnement de composant *Profil Manager* . . . 74
5.4 Modèle d'un fichier Manifest . . . 75
5.5 Fonctionnement de composant *Image Mananger* . . . 75
5.6 Fonctionnement de composant *Information Manager* . . . 76
5.7 Fonctionnement de composant *Virtual Machine Manager* . . . 78
5.8 Fonctionnement de composant *Cloud Middleware Connector* . . . 79
5.9 Déroulement de l'algorithme . . . 81
5.10 Taux d'utilisation des ressources de OpenStack après R1 . . . 82
5.11 Taux d'utilisation des ressources de OpenNebula après R1 . . . 82
5.12 Taux d'utilisation des ressources de OpenStack après R5 . . . 83
5.13 Taux d'utilisation des ressources de OpenNebula après R5 . . . 83

A.1 Page d'authentification . . . 97
A.2 Ajouter un utilisateur . . . 98
A.3 Gérer les utilisateurs . . . 99
A.4 L'infrastructure OpenNebula . . . 99
A.5 Ajouter une application scientifique . . . 100
A.6 Modèle d'un fichier Manifest . . . 100
A.7 Gérer les applications grid . . . 101
A.8 Gérer les machines virtuelles . . . 101
A.9 Créer une machine virtuelle . . . 102
A.10 gérer les machines virtuelles . . . 103

Liste des tableaux

1.1 description des différents nœuds de la grille de calcul [64] 7

2.1 Caractéristiques des plateformes open source 24

3.1 Tableau comparatifs 46

4.1 Définition des Attributs du fichier *Manifest* 56

5.1 Les ressources disponibles dans les clouds 80
5.2 Les besoins en ressources des applications gridifiées 80

Introduction générale

Contexte

La science d'aujourd'hui couvre des problèmes de plus en plus complexes et est poussée par des technologies de plus en plus puissantes. Elle est bien basée sur le calcul, l'analyse de données et la collaboration aussi bien que sur les efforts des théoriciens et expérimentalistes. A cet effet, la communauté scientifique a développé une large gamme de solutions pour résoudre le problème des besoins sans cesse croissants de puissance de calcul.

Les grilles offrent un paradigme efficace pour mutualiser un ensemble de ressources informatiques entre un certain nombre d'organisations, ce qui augmente l'utilisation globale des ressources. L'infrastructure de grille européenne (EGI) est un exemple éminent d'un environnement Grille, utilisé par des dizaines de milliers de scientifiques du monde entier. Récemment, le concept de Cloud Computing a émergé et se propage très largement dans presque tous les domaines des technologies de l'information, et cela grâce à la virtualisation. Ce nouveau paradigme est considéré comme une alternative aux grilles de calcul pour l'exécution des applications scientifiques. Il offre de nouvelles perspectives en termes de déploiement de logiciels dans les systèmes distribués. Il organise un pool partagé de serveurs dans les centres de données dans une infrastructure de cloud, qui peut fournir à la demande et à tout moment des ressources informatiques (par exemple, machine virtuelle, CPU, stockage, réseau, base de données, applications et services) aux utilisateurs.

En effet, Le cloud computing marque une réelle avancée vers l'infrastructure informatique dématérialisée. Cependant, l'adoption de ce modèle soulève un certain nombre de défis, notamment l'interopérabilité car on ne doit pas parler d'un seul cloud mais de plusieurs clouds avec autant de logins et d'interfaces. Afin de résoudre le problème d'interopérabilité, les services Cloud Broker ont émergé comme une solution pour fournir des services d'intégration des différents Clouds. C'est là que peut prendre place la solution d'authentification qui permet à un seul login d'accéder simultanément à plusieurs clouds, pour aider les entreprises à adopter facilement le cloud [54], les aider à gérer et maintenir leurs services et applications

dans le cloud, et aussi pouvoir migrer les données de l'entreprise d'un cloud à un autre Susceptible de mieux répondre à leurs besoins [98].

Problématique

Les grilles de calcul rencontrent certains problèmes de flexibilité. Parmi eux, la gestion des ressources dans l'infrastructure. Ce problème est causé par les applications scientifiques qui demandent beaucoup de ressources afin d'être exécutées. Dans ce cas-là, l'exploitation de la flexibilité du système d'administration du service cloud IaaS [1] peut faciliter et simplifier la gestion des ressources déployées pour les applications gridifiées, tout en diminuant la charge sur la grille de calcul. Mais l'utilisation d'un seul service cloud IaaS peut ne pas pallier le problème en cas de non disponibilité des ressources d'où la nécessité d'introduction d'un système multi-clouds. Mais l'un des problèmes du système multi-cloud est l'interopérabilité. En effet, l'interopérabilité nécessite la prise en charge de plusieurs aspects ; par exemple, d'une part, la capacité de déplacer facilement une infrastructure virtualisée entre les différents fournisseurs de cloud. D'autre part, l'utilisation simultanée de plusieurs clouds géographiquement distribués afin d'assurer la haute disponibilité et l'efficacité du service.

Dans notre problématique de fédération multi-cloud dans les environnements orientés "grid services", nous nous intéressons de migrer l'exécution des applications d'analyse gridifiées de la grille de calcul vers un système multi-cloud sur des machines virtuelles tout en essayant de résoudre les problèmes suivants :

- Éviter l'enfermement propriétaire (vendor lock-in) ;
- Accéder à un environnement Multi-Cloud en faisant abstraction de leurs différences de structures ;
- Assurer la haute disponibilité des applications gridifiées dans un environnement multi-cloud ;
- Choisir l'emplacement des machines virtuelles dans un environnement multi-cloud.

Contribution

L'approche que nous proposons a comme objectif de montrer l'importance du rôle que peuvent jouer les environnements multi-Clouds dans la gestion efficace et l'administration simplifiée des machines virtuelles contenant des applications scientifiques gridifiées. En effet, nous proposons une plateforme d'interopérabilité

1. Infrastructure as a Service

pour les services IaaS. Cette plateforme permet une seule authentification pour accéder à plusieurs clouds simultanément et elle permet d'un côté, à un administrateur d'une plateforme grille d'enregistrer les différentes images des machines virtuelles contenants des applications gridifiées (images grilles) dans les différents clouds, afin d'assurer la haute disponibilité des applications scientifiques gridifiées. D'un autre côté, notre plateforme permet aux utilisateurs l'instanciation de ces images gilles, le déploiement et la gestion des différentes machines virtuelles dans différents clouds d'une façon transparente. Pour cela nous proposons un algorithme de placement des machines virtuelles dans notre environnement multi-cloud.

Plan du document

Ce mémoire commence par une introduction générale qui sera suivie de deux parties présentant respectivement l'état de l'art et notre contribution et se termine avec une conclusion générale pour synthétiser notre travail ainsi qu'un ensemble de perspectives pour des travaux futurs.

Dans la première partie, l'état de l'art est présenté en trois chapitres. Le premier chapitre aborde une vision détaillée de la technologie des grilles de calcul : ses définitions, caractéristiques, domaines d'application, architecture, etc. Le deuxième explique quelques notions fondamentales et généralités à propos du Cloud Computing. Ainsi, nous avons mené une étude exhaustive et comparative de différentes plateformes cloud computing open source. Puis nous présentons les travaux de recherches sur l'intégration des deux technologies : grille et cloud. Dans le troisième chapitre nous présentons les concepts liés au multi-cloud. De plus, nous abordons quelques architectures multi-clouds existantes dans la littérature.

Dans la seconde partie, nous détaillons notre contribution en commençant par un chapitre où nous exposons notre approche. Puis un deuxième chapitre où nous présentons la mise en œuvre pour la valider.

Première partie

Contexte et État de l'art

Chapitre 1

Grille de Calcul

1.1 Introduction

Le terme Grille de calcul trouve son origine dans le vocabulaire anglo-saxon. Il est apparu il y a une trentaine d'années par analogie avec le réseau de distribution de l'électricité. En effet, cette technologie permet d'avoir accès à de nombreuses ressources informatiques telles que des serveurs de calcul, serveurs de données, éléments réseau, le tout avec la même facilité que le courant électrique lorsqu'un interrupteur est actionné. De manière plus formelle, la grille de calcul est une forme d'informatique distribuée, basée sur le partage dynamique des ressources entre des participants, des organisations et des entreprises dans le but de pouvoir les mutualiser, et faire ainsi exécuter des applications de calcul intensif ou des traitements de très gros volumes de données. Ces applications ne seraient pas possibles au sein d'un seul organisme ou d'une seule entreprise. Ce chapitre nous permet de faire une large revue d'ensemble sur le fonctionnement et les champs d'application de la technologie de la grille de calcul.

1.2 Définition

D'après les auteurs V.Louve[61] et A.Ahmar[1], le but de la grille de calcul est de partager les ressources d'un réseau (cycles CPU, stockage...) et de les rendre accessible à tout programme qui en aurait besoin, et ce, en tout points du réseau. La philosophie de la grille de calcul est d'abstraire les ressources d'un réseau hétérogène pour les partager et les rendre accessible et ainsi, bénéficier d'une puissance de calcul plus grande et disponible à la demande sans risques de goulots d'étranglements dus à une répartition déséquilibrée des charges.

Buyya définit la grille comme un type de système parallèle et distribué qui permet le partage, la sélection, et l'agrégation dynamiques de ressources autonomes,

géographiquement distribuées lors de leur exécution en fonction de leur disponibilité, capacité, performance, coût, et des besoins des utilisateurs en terme de qualité de service[16] [14] . Ian Foster présente trois critères de base pour distinguer une grille d'un autre système distribué [38] :

- Des ressources coordonnées dont leur administration n'est pas centralisée,
- Des méthodes utilisées qui sont standardisées (en utilisant des normes, ouvertes,des protocoles et des interfaces à des fins générales),
- Délivrer une Qualité de Service non négligeable (non triviale) (temps de réponse, disponibilité, sécurité, etc.).

1.3 Classement et taxonomie

Il existe une diversité de point de vu dans la littérature sur les catégories des grilles de calcul. D'après les auteurs P.Thierry[80], M.Quinson et F.Suter [82] on distingue 3 catégories :

1.3.1 Grille d'information :

Une grille d'information est une infrastructure générique qui facilite l'intégration de toute information n'importe où à travers des sources de données hétérogènes dans un environnement réseau. En particulier, une grille d'information constitue une virtualisation des sources de données pour permettre une intégration de toute information disponible. Ce type de grille permet de partager la connaissance. On a même considéré que le Web représente une application à succès du concept de Grille. Toutefois, la source de l'information n'est pas toujours connue (sources fiables ?).

1.3.2 Grille de stockage :

La grille de stockage est un nouveau modèle pour le déploiement et la gestion des ressources de stockage réparties sur plusieurs systèmes et réseaux, en utilisant efficacement les capacités de stockage disponibles [29] . Elle est généralement composée de ressources offrant une grande capacité de stockage, en mémoire et sur disque. Un autre exemple de grille de stockage est le Projet S3 (Simple Storage Service) d'Amazon [10].

1.3.3 Grille de calcul

Une grille de calcul est un dispositif logiciel qui offre aux utilisateurs des puissances quasi illimitées de calcul ou de stockage de données, grâce à un accès trans-

parent et facile (une simple connexion à un réseau à très haut débit de type Internet) à un vaste ensemble de ressources informatiques distribuées sur une grande échelle[20].

1.4 Les propriétés des grilles de calcul

Selon l'auteur A.Ahmar[1], les ressources de la grille de calcul sont potentiellement qualifiées de :

- **Partagées :** elles sont mises à la disposition des différents consommateurs de la grille et éventuellement pour différents usages applicatifs.
- **Distribuées :** elles sont situées dans des lieux géographiques différents.
- **Hétérogènes :** elles sont de toute nature, différentes par exemple par le système d'exploitation ou le système de gestion des fichiers.
- **Coordonnées :** les ressources sont organisées, connectées et gérées en fonction de besoins (objectifs) et contraintes (environnements). Ces dispositions sont souvent assurées par un ou plusieurs agents, qu'ils soient centralisés ou répartis.
- **Non-contrôlées** (ou autonomes) : les ressources ne sont pas contrôlées par une unité commune. Contrairement à un cluster, les ressources sont hors de la portée d'un moniteur de contrôle.
- **Délocalisées :** les ressources peuvent appartenir à plusieurs sites, organisations, réseaux et se situer à différents endroits géographiques.

1.5 Architecture d'une grille de calcul

D'après les auteurs A.Ahmar[1] et C.Therry[94] , bien que chaque grille présente son architecture elle-même, mais, en générale, l'architecture de toutes les grilles de calcul est modélisée sous le modèle en couche comme l'illustre la figure 1.1. Une couche est une abstraction représentant un ensemble de fonctions du système qui permettent des actions de même niveau. Chaque couche fait appel aux services de n'importe quelle couche inférieure. Nous retrouvons dans les couches supérieures les fonctionnalités proches de l'utilisateur alors que les couches inférieures, de bas niveau, représentent les éléments constituant l'infrastructure hardware et réseau.

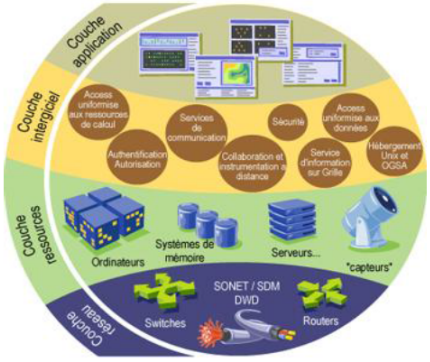

FIGURE 1.1: Architecture d'une grille de calcul[53]

1.5.1 La couche réseau

définit des protocoles de base pour faire des transactions sur la grille. L'utilisation de protocoles standards simplifie également le développement et le déploiement des solutions. Ainsi, pour la communication et la sécurité, de nombreux protocoles développés pour Internet ont été réutilisés. Pour la communication, les protocoles nécessaires sont ceux relatifs au transport, au routage et à la gestion des noms. Le choix se porte essentiellement sur les protocoles de la pile TCP/IP. Par exemple TCP et UDP sont utilisés pour le transport, IP pour le routage, ICMP pour la surveillance, et le DNS pour les applications. Le constructeur de la grille décide des protocoles sur la grille. Dans cette couche on implémente le système pour la sécurité comme GSI (Grid Security Infrastructure) [43].

1.5.2 La couche ressources

Représente les ressources de la grille. Il s'agit de l'ensemble des serveurs de calcul, de stockage, des grappes de calcul (clusters), des catalogues de données, des équipements spéciaux . . . etc.

1.5.3 La couche Intergiciel

Une grille de calcul intègre un middleware, qui permet le dialogue entre les différents composants d'une application répartie dont la gestion des ressources, et les ordonnancements (scheduling) des différents jobs pris en charge. Cette couche offre un ensemble de services tels que la sécurité, l'authentification, le service d'information, de communication, d'ordonnancement, etc. Il existe plusieurs middlewares, les plus connus, sont : Legion [57], Unicore [95], Condore [22], Globus [43] , g-Lite [41] et gLite-Emi [34]. Nous supposons que l'architecture de la grille est celle

mise en œuvre par la pile de middleware gLite-EMI, développé dans le cadre du projet EGI (European Grid Infrastructure). Le middleware gLite-EMI fournit une infrastructure de grille qui est accessible aux membres des communautés organisées dans les organisations virtuelles (Virtual Organization (VO)) ; tel que les personnes appartenant à une même VO ont généralement des droits et besoins communs, typiquement parce qu'elles travaillent dans une même discipline. **Une organisation virtuelle *(Virtual Organization (VO))*** est une entité logique, limitée dans le temps, créée dynamiquement dans le but de résoudre un problème spécifique, en fournissant et en allouant des ressources à la demande [51].

1.5.4 La couche application

C'est la couche « visible » par les utilisateurs. Cette couche donne l'accès aux ressources par l'intermédiaire d'interfaces graphiques GUI [1] ou en mode commande CUI [2], ou totalement intégrées dans les logiciels spécifiques.

1.6 Éléments constitutif d'une grille de calcul

Dans le tableau1.1, nous allons voir une brève description des différents éléments constitutifs de la grille EGEE basée sur le middleware g-Lite-EMI afin de mieux comprendre le fonctionnement de la grille.

1. Graphical User Interface
2. Command User Interface

Noeud	Nom	Rôle
WN	Worker Node	Est le serveur effectuant le calcul.
CE	Computing Element	Est le serveur interagissant directement avec le système de batch qui gère la soumission et le contrôle des jobs.
SE	Storage Element	Est l'élément gérant le stockage. Il permet de manipuler des fichiers de données volumineux.
UI	User Interface	Est le point d'accès à la grille. Il s'agit d'une machine contenant les différentes CLI et API qui permettent d'invoquer les services gLite.
WMS	Workload Management System	Est le serveur avec lequel interagit l'utilisateur lors de la soumission, le suivi et la récupération des résultats d'un calcul. Le serveur choisit le noeud de grille correspond au pré-requis du calcul (CE).
LFC	File Catalog	Est une base de données qui permet de garder la trace de la localisation des diverses copies des fichiers dans les différents SEs.
IS	Système d'information	A pour rôle de collecter des informations sur l'état des ressources.

TABLE 1.1: description des différents nœuds de la grille de calcul [64]

1.7 Chemins d'un job

La figure 1.2 nous montre le cheminement d'un job exécuté sur une grille de calcul [42][41].

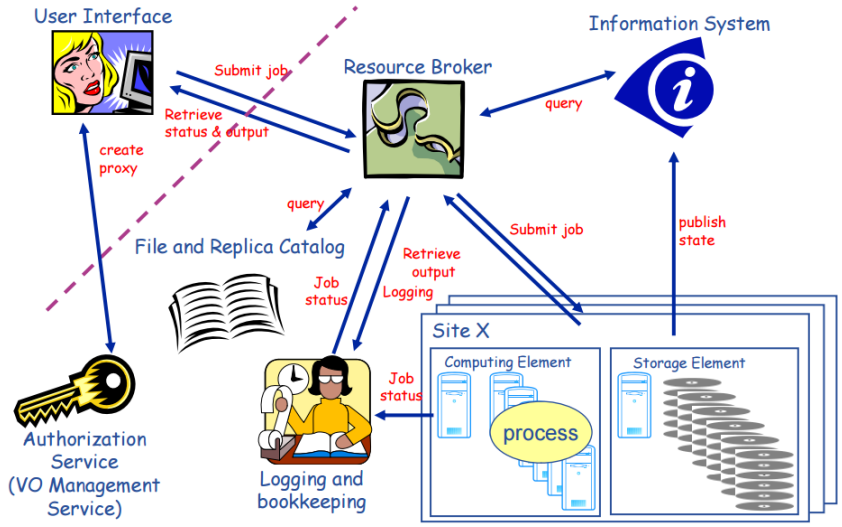

FIGURE 1.2: Infrastructure des services de la grille [41]

1. Après avoir obtenu un certificat numérique d'une autorité de certification digne de confiance, en vous s'inscrivant dans une VO et d'obtenir un compte sur l'UI, l'utilisateur est prêt à utiliser la grille. Il se connecte à l'UI et crée un proxy qui lui permet d'interagir en toute sécurité avec le système ;
2. L'utilisateur soumet le job au WMS via l'UI. Le WMS recherche le/les CE(s) pouvant prendre en charge l'exécution du job en consultant l'IS. L'utilisateur transmet les fichiers d'entrée dans l'InputSandBox [1] ;
3. Le job ainsi que l'InputSandBox sont transférés au CE qui prend en charge le job dans sa queue (liste d'attente) ;
4. Le CE envoie le job sur un ou plusieurs WN (s) disponibles ;
5. Lorsque le job est terminé, les fichiers produits par celui-ci sont disponibles sur le LRMS [2]. Le WMS est averti que le job s'est terminé ;
6. Le WMS récupère les fichiers de sortie dans l'OutputSandBox [3] ;
7. Le WMS envoie les résultats (l'outputSandBox) à l'utilisateur via l'UI ;
8. L'utilisateur peut interroger à tout moment l'état de son job par l'intermédiaire du Logging and Bookkeeping Service (LB). Le LB conserve une trace de l'exécution des jobs.

1.8 Domaines d'application des grilles de calcul

Il existe plusieurs projets qui utilisent la technologie de la grille de calcul pour exécuter leurs applications. Nous citons quelques applications [63] :

1.8.1 Les sciences de la physique

Les sciences de la physique est un domaine en plein essor avec le développement de la technologie grille. Le projet LHC [55] du CERN a été lancé en 2007 pour fournir un débit de données massives (de 10Gb/s à 100Gbit/s) où chaque événement de la physique peut être traité indépendamment, résultant de parallélisme de billions de voies. Ce grand défi ne peut être résolu par un seul ordinateur ou cluster. Par conséquent, les scientifiques essaient toujours de trouver des moyens pour résoudre ce problème avec les nouvelles possibilités offertes par la grille de calcul. A cet effet, plusieurs projets de grille sont créés ou impliqués : LCG [55], EGEE [53], DataGrid [26], GridPP [45] et INFN [4].

1. L'InputSendBox contient le chemin de l'exécutable et les paramètres de l'application. Il les transfère depuis l'UI au WNs
2. LRMS (Local Resource Management System) est le système de gestion de ressources locale
3. L'OutputSendBox contient les fichiers qui sont retournés par la grille, une fois l'exécution du job terminée. Il les transfère depuis les WNs à l'UI

1.8.2 Le biomédical

La biologie de calcul a changé d'une science traditionnelle de calcul intensif à une science de haut débit, pilotée par les données. Beaucoup d'expérimentations et d'équipements de mesure sont directement liés aux ressources informatiques pour obtenir et traiter rapidement les données. Parmi les projets impliqués dans ce domaine de recherche : La technologie DataGrid [26], Le projet de grille japonais nommé BioGRID [13], le projet EGEE [53]. En fait, de nombreux pays ont établi des projets de grille de calcul spéciale pour la recherche biologique, en vue du développement pharmaceutique et l'élucidation biodynamique.

1.8.3 L'observation de la terre et climatologie

Les satellites d'observation terrestre retournent 100 giga-octets d'images de données par jour. L'équipement de stockage sur le terrain a déjà stocké plus de 1 000 000 giga octets de données. Ces données peuvent être utilisées pour analyser les profils d'ozone ou à détecter du pétrole. La grille permet à ces données d'être facilement partagées entre les différents producteurs et consommateurs. Par exemple, DataGrid [26] a réalisé le stockage de données réparties dans toute l'Europe.

1.8.4 L'astronomie

L'astronomie est un domaine dynamique de la recherche car l'être humain est toujours curieux de l'espace. À partir d'images traitées de planètes à d'énormes quantités de données brutes, il y a une grande quantité de données basées sur l'astronomie disponibles sur Internet. Il en résulte des exigences de stockage importantes. Les projets de grille D-Grid [25], DutchGrid [31], OSG [76], ChinaGrid[50] et SDG [90] sont tous impliqués dans ce domaine de recherche.

Il existe encore d'autres applications et dans d'autres domaines qui sont produites par des scientifiques et des chercheurs qui s'intéressent aux grilles de calcul.

1.9 Conclusion

Dans ce chapitre nous avons présenté la technologie grille de calcul. La grille peut être avantageuse pour beaucoup d'applications axées sur le traitement, qui peuvent combiner de puissantes ressources de calcul pour traiter des problèmes qui ne pourraient pas être résolus sur un système unique car il peut prendre un temps d'exécution énorme par rapport à leur exécution sur une grille de calcul. Avec l'évolution rapide de la technologie des grilles de calcul en terme de matériel et logiciel à cause de l'augmentation des besoins en puissance de calcul pour la

recherche scientifique fondamentale et appliquée, parmi les défit des grilles de calcul sont :

- la gestion de leurs ressources.
- l'hétérogénéité des plates-formes (Systèmes d'exploitation), ceci à un impact négatif sur l'exécution d'une application sous Windows dans un noeud distant sous Linux.
- la grille de calcul ne permet pas aux utilisateurs de personnaliser l'environnement (infrastructure) de la grille afin d'exécuter leur expérimentation, car les sites de la grille sont statiques.

Le ressort du nouveau paradigme « Cloud Computing » peut remplacer les grilles de calcul car il fournit une flexibilité et une gestion simplifiée des ressources de l'infrastructure Cloud. Ainsi il permet aux utilisateurs de personnaliser l'environnement d'exécution de leurs applications, grâce à la virtualisation. Dans le chapitre suivant nous allons décrire en détail la technologie du Cloud Computing.

Chapitre 2

Cloud Computing

2.1 Introduction

Dans le monde hyper-connecté d'aujourd'hui, qui nécessite une disponibilité permanente, la technologie est devenue un enjeu critique pour le succès des entreprises de toutes tailles. Elle constitue désormais un avantage économique indiscutable : elle permet de stimuler l'engagement client, d'augmenter la productivité, d'innover et de se différencier des concurrents. Au sein des entreprises, les budgets alloués aux solutions informatiques sont habituellement restreints. Ainsi, ces structures ont souvent du mal à suivre le rythme des évolutions technologiques nécessaires au maintien des performances de leurs activités. Les solutions apportées par le Cloud computing peuvent être une réponse à cette double contrainte de réduction de moyens et de croissance de production et de valeur ajoutée.

les solutions de cloud computing sont bien souvent mises en avant comme étant la solution à tous les problèmes. Elles offrent d'innombrables possibilités. Elles sont synonymes de souplesse et de coûts réduits pour les entreprises. Elles sont une manne financière pour les fournisseurs[52].

2.2 Historiques et définitions

Le mot Cloud est apparu au début des années 90 pour désigner des réseaux disposant d'un mode de transfert asynchrone, mais depuis une dizaines d'années l'expression « cloud computing » a pris de plus en plus d'importance. Salesforce.com fut le premier hébergeur de cloud en 1999, suivi en 2002 par Amazon qui propose un ensemble d'hébergement d'application, de stockage et d'offre d'emploi. Il a développé ses services Amazon Web Services (AWS) et Elastic Compute Cloud (EC2) en 2005 et 2006 respectivement. Ce dernier fut le premier service de « Cloud » réellement accessible.

En 2007, Google, IBM et des universités lancèrent un projet de recherche sur le « Cloud » qui permet de lui faire gagner en popularité et en consistance. C'est en 2009 que la réelle explosion du « cloud » survint avec l'arrivée sur le marché de sociétés comme Google (Google App Engine), Microsoft (Microsoft Azure), IBM (IBM Smart Business Service), Sun (Sun Cloud) et Canonical Ltd (Ubuntu Enterprise Cloud) [9].

Comme toute nouvelle technologie, il est difficile de trouver une définition universelle au Cloud Computing. Il s'agit de faire un choix parmi de nombreuse définitions existantes que ce soit dans la littérature scientifique ou sur internet.

- «Le Cloud Computing est un modèle qui offre aux utilisateurs du réseau un accès à la demande, à un ensemble de ressources informatiques partagées et configurables, et qui peuvent être rapidement mises à la disposition du client sans interaction direct avec le prestataire de service.» [62] .
- CISCO : «Le Cloud Computing est une plateforme de mutualisation informatique fournissant aux entreprises des services à la demande avec l'illusion d'une infinité de ressources» [11]

Donc le Cloud Computing est un concept qui a le but de pousser les entreprises à externaliser les ressources qu'elles stockent. Ces ressources offrant des capacités de stockage et de calcul, des logiciels de gestion de messagerie, et d'autres services sont mises à disposition par des sociétés tierces et accessibles, grâce à un système d'identification, via un PC et une connexion à Internet.

L'idée principale à retenir est que le Cloud n'est pas un ensemble de technologies, mais un modèle de fourniture, de gestion et de consommation de services et de ressources informatiques localisé dans des Datacenter.

2.3 Caractéristiques de Cloud Computing

Globalement, nous distinguons cinq caractéristiques principales du Cloud Computing [101] :

- **Service à la demande :** le service fournit au client au moment de son besoin et de sa demande ;
- **Accès réseau, clients variés** : plusieurs clients peuvent accéder au service à partir de plusieurs réseaux d'accès selon le contrat SLA [1] avec le fournisseur de service du Cloud Computing ;

1. le Service Level Agreement (SLA), est un document qui définit contractuellement la qualité du service à attendre.

- **Mise en commun des ressources (pooling)** : le même service et les mêmes ressources sont consommés par plusieurs clients ;
- **Élasticité** : service évolutif avec une certaine souplesse dans sa modification et déploiement ;
- **Service mesuré et facturation à l'usage** : le client va payer ce qui a consommé selon le type et le temps d'utilisation du service.

2.4 Les Services du Cloud

La Figure 2.1 représente les différentes couches du Cloud Computing de la couche la moins visible pour les utilisateurs finaux à la plus visible. Ces services dans l'industrie sont respectivement référencés comme l'IaaS (Infrastructure as a Service) qui est plutôt gérée par les architectes réseaux. La couche PaaS(Plateforme as a Service) qui est destinée aux Développeurs d'applications et finalement le logicielle comme un service (SaaS) est le « produit final » pour les utilisateurs :

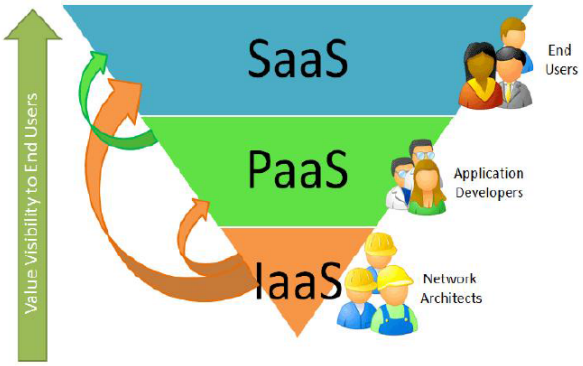

FIGURE 2.1: Différents services de Cloud Computing [35]

- **IaaS (Infrastructure as a Service / Infrastructure fournie en mode service) :** seul le serveur est dématérialisé. Un prestataire propose la location de composants informatiques comme des espaces de stockages, une bande passante, des unités centrales et des systèmes d'exploitation. Les utilisateurs d'une IaaS peuvent donc utiliser à la demande des serveurs virtuels situés dans des datacenters sans avoir à gérer les machines physiques (coûts de gestion, remplacement de matériel, climatisation, électricité...etc). L'IaaS offre une grande flexibilité, avec une administration à distance, et permet d'installer tout type de logiciel. En revanche, cette solution nécessite la présence d'un administrateur système au sein de l'entreprise, comme pour les solutions serveur classiques [35].

- **PaaS (Platform as a Service / Plate-forme fournie en mode service)** le matériel (serveurs), l'hébergement et le framework [1] d'application sont dématérialisés. L'utilisateur loue une plateforme sur laquelle il peut développer, tester et exécuter ses applications. Le déploiement des solutions PaaS est automatisé et évite à l'utilisateur d'avoir à acheter des logiciels ou d'avoir à réaliser des installations supplémentaires, mais ne conviennent qu'aux applications Web [35].
- **SaaS (Software as a Service / Logiciel fourni en mode service)** le matériel, l'hébergement, le framework d'application et le logiciel sont dématérialisés et hébergés dans un des datacenters du fournisseur. Les utilisateurs consomment les logiciels à la demande sans les acheter, avec une facturation à l'usage réel. Il n'est plus nécessaire pour l'utilisateur d'effectuer les installations, les mises à jour ou encore les migrations de données. Les solutions SaaS constituent la forme la plus répandue du Cloud Computing [35].

Il est à noter également dans la littérature informatique d'autres termes utilisés dans le cadre du Cloud Computing qui peuvent ajouter un peu plus de confusion sur le sujet [12] :

- **Hardware as a Service (HaaS)** peut être considéré comme un équivalent à IaaS.
- **Database as a Service (DaaS)** concerne plus précisément les bases de données et peut s'apparenter à une composante de PaaS.
- **Development as a Service** peut se comprendre comme un environnement de développement et de test intégré à la plateforme PaaS.
- **Integration as a Service** est un bus d'échange intégré à la plateforme PaaS.

2.5 Les Modèles de déploiement du Cloud Computing

Le NIST [2] [65]définit plusieurs modèles de déploiement du Cloud Computing que sont

- **Cloud privé :** L'infrastructure du Cloud est réservée à l'usage exclusif d'une seule organisation. Elle peut être possédée, gérée et opérée par cette organisation, un intervenant extérieur ou une combinaison des deux. Elle est située dans les locaux de l'organisation ou dans ceux d'un hébergeur externe.

1. Un framework est un kit de composants logiciels structurels, qui servent à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel.
2. NIST : National Institute of Standards and Technology

- **Cloud public :** L'infrastructure du Cloud est destinée à un usage public. Elle peut être possédée, gérée et opérée par un organisme privé, public, académique ou une combinaison de ceux-ci. Elle est située chez un hébergeur.
- **Cloud communautaire :** L'infrastructure du Cloud est réservée à l'usage d'une communauté spécifique de consommateurs partageant des intérêts communs : missions, exigences de sécurité, partage d'informations et ou d'applications, Elle peut être possédée, gérée et opérée par un ou plusieurs organismes participant à la communauté, un intervenant extérieur ou une combinaison d'entre eux. Elle est située dans les locaux de l'organisation ou dans ceux d'un hébergeur externe.
- **Cloud hybride :** L'infrastructure du Cloud est composée d'au moins deux infrastructures différentes (privée, publique ou communautaire) qui conservent leur autonomie mais qui sont liées entre elles par des technologies (propriétaires ou non) assurant la portabilité des données et des applications.

2.6 Les éléments constitutifs du Cloud Computing

Dans cette section nous jetons un regard de haut niveau sur les éléments technologiques qui constituent les bases de l'infrastructure informatique du Cloud

a. **L'Infrastructure :** L'infrastructure informatique du Cloud est un assemblage de serveurs, d'espaces de stockage et de composants réseau organisés de manière à permettre une croissance incrémentale supérieure à celle que l'on obtient avec les infrastructures classiques. Ces composants doivent être sélectionnés pour leur capacité à répondre aux exigences d'extensibilité, d'efficacité, de robustesse et de sécurité. Les serveurs d'entreprise classiques ne disposent pas des capacités réseau, de la fiabilité ni des autres qualités nécessaires pour satisfaire efficacement et de manière sécurisée les accords de niveau de service SLA [100].

b. **La Virtualisation :** La virtualisation est la principale technologie dans le Cloud. C'est une méthode pour partitionner une ressource physique en plusieurs ressources virtuelles, par exemple : un serveur, un espace de stockage ou un réseau lors de la création des machines virtuelles. Elle permet d'intégrer les différents serveurs de façon plus flexible pour faciliter l'utilisation. Le but de la virtualisation est de faire la transparence d'utilisation et l'efficacité d'exploitation des ressources ainsi que d'assurer le fonctionnement des différents services et la séparation entre de multiples utilisateurs impliqués dans un matériel physique [100].

c. **Réseaux IP :** Dans une infrastructure de Cloud, le réseau non seulement connecte les utilisateurs au Cloud, mais sert également à l'interconnexion interne du Cloud. Le modèle mis en œuvre dans un réseau d'entreprise ne répond pas aux besoins d'efficacité et de sécurité associés à l'acquisition et au fonctionnement du Cloud. À l'échelle du Cloud, le réseau doit s'orienter vers un système Carrier-Grade, avec des stratégies réseau optimisées. Les multiples commutateurs disséminés tout au long des chemins de données deviennent des points uniques de défaillance (SPOF, single points of failure) et ajoutent des coûts de différentes manières. Bien que l'optimisation puisse conduire à un seul réseau unifié, la sécurité nécessite un partitionnement ou une virtualisation du réseau pour une séparation réelle entre les différentes classes du trafic. Le réseau sera probablement plus plat, mais il faut s'attendre à plusieurs réseaux en parallèle pour une meilleure sécurité [100].

d. **Les Interfaces de service :** L'interface de service placée entre le fournisseur et le client est un élément de différenciation du Cloud. Elle représente un contrat qui fait respecter la proposition de valeur décrite par des SLA et des conditions tarifaires. Si le Cloud semble nouveau, c'est principalement en raison de cette interface. Elle représente la valeur d'un fournisseur et sert de base à la concurrence. Par l'ajout d'interfaces de service libre, d'autres optimisations sont obtenues. Les clients du Cloud sont en mesure d'engager des ressources de manière automatisée sans que le service informatique soit un obstacle. L'espace de stockage et les ressources sont présentés à travers une interface graphique que l'utilisateur peut manipuler de manière à obtenir et à instancier une infrastructure informatique virtuelle [100].

e. **Le Centre de données (Datacenters) :** Un centre de traitement de données en anglais « data center » est un site physique sur lequel se regroupent des équipements qui constituent le système d'information de l'entreprise (mainframes, serveurs, baies de stockage, équipements réseaux et de télécommunications, etc.). Il peut être interne ou externe à l'entreprise, exploité ou non avec le soutien de prestataires. Il comprend en général un contrôle sur l'environnement (climatisation, système de prévention contre l'incendie, etc.), une alimentation d'urgence et redondante, ainsi qu'une sécurité physique élevée. Cette infrastructure peut être propre à une entreprise et utilisée par elle seule ou à des fins commerciales. Ainsi, des particuliers ou des entreprises peuvent venir y stocker leurs données suivant des modalités bien définies[70].

2.7 la différence entre la virtualisation et le Cloud Computing

La virtualisation permet aux entreprises d'utiliser une seule pièce de matériel physique, pour effectuer le travail d'un grand nombre de machines. Plusieurs instances d'un système d'exploitation s'exécutant sur un seul dispositif matériel, sont beaucoup plus économiques qu'une pièce de matériel pour chaque tâche du serveur. Par contre, le Cloud Computing est l'accès par Internet aux applications d'entreprise fonctionnant dans un environnement non-locale. Néanmoins, le Cloud Computing peut certainement tirer des avantages de la virtualisation [70].

2.8 Les Acteurs du Cloud Computing

L'architecture de référence du Cloud Computing de l'organisme NIST définit cinq acteurs majeurs : le consommateur du cloud, le fournisseur du cloud, l'auditeur de cloud, le cloud broker et le transporteur de cloud [59](voir la figure 2.2).

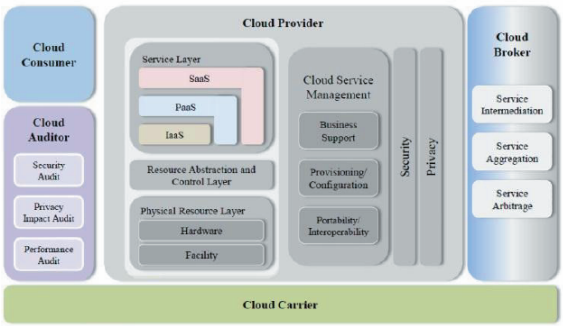

FIGURE 2.2: L'architecture de référence du Cloud Computing de l'organisme NIST [59]

1. **Consommateur de cloud.** Une personne ou un organisme qui entretient une relation d'affaires avec le fournisseur de cloud et utilise le service fourni.
2. **Fournisseur de cloud.** Une personne, un organisme ou une entité responsable de mettre à disposition des consommateurs des services en cloud.
3. **Auditeur de cloud.** Un tiers qui peut procéder à une évaluation des services du cloud comuting tout en restant indépendant.
4. **Cloud broker.** Une entité qui gère l'utilisation, la performance et la prestation de services du cloud tout en négociant les relations entre les fournisseurs et le consommateur du cloud.
5. **Transporteur du cloud.** L'intermédiaire qui fournit la connectivité et transporte les services du fournisseur du cloud vers le consommateur.

2.9 Les plates-formes cloud IaaS "Open-source"

A l'heure actuelle, différentes plates-formes permettent la gestion des datacenters qui fournissent l'Infrastructure-as-a-Service. Ces plates-formes de gestion sont par conséquent des outils très utiles dans le domaine du cloud computing. Elles nécessitent néanmoins des connaissances techniques afin de créer des machines virtuelles comme par exemple savoir quelle est la capacité (puissance CPU,mémoire RAM) nécessaire pour faire tourner une application, ou savoir quel OS installer. Il faut également configurer les différentes machines afin qu'elles puissent communiquer entre elles ou avec le monde extérieur (Internet). Dans cette section nous allons présenté les plates-formes IaaS : OpenNebula[73], OpenStack [74] et Eucalyptus[36] qui sont des frameworks open source.

2.9.1 OpenNebula

FIGURE 2.3: Logo OpenNebula

OpenNebula [73] est une plate-forme de gestion d'infrastructure cloud open source. Elle a été conçu pour permettre aux gestionnaires de gérer de manière efficace des infrastructures cloud complexes et hétérogènes. Cette plate-forme gère les ressources virtuelles de clouds publics et hybrides. Elle présente une architecture en couches, ce qui permet la gestion centralisée des datacentres, et fournit un niveau détaillé de personnalisation. Au sommet de la pile, elle expose de multiples API à fin de communiquer avec AWS EC2 [1] [32] et L'Open Cloud Computing Interface (OCCI) [71] écrite par l'Open Grid Forum (OGF)[72].

2.9.1.1 Architecture d'OpenNebula

L'architecture d'OpenNebula est décrite visuellement dans la figure 2.4. Elle est organisée en couches :

1. Amazon Web Services, Amazon Elastic Compute Cloud

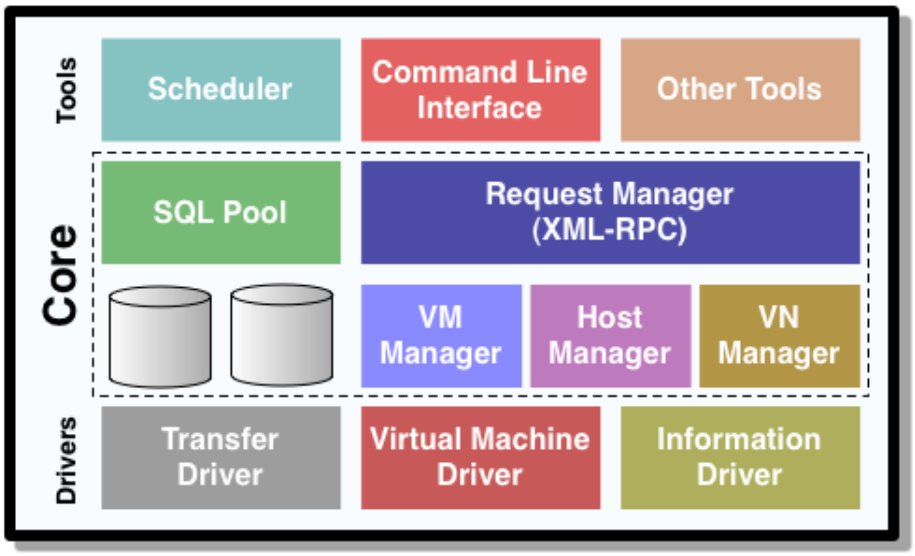

FIGURE 2.4: Architecture OpenNebula

1. **Tools** : Cette couche contient des outils utilisés pour interagir avec une instance d'OpenNebula et pour gérer les machines virtuelles. L'outil "scheduler" est un composant qui recherche automatiquement les hôtes physiques afin de déployer des machines virtuelles nouvellement créés. L'outil "CLI" est utilisée pour émettre des commandes pour gérer le cycle de vie complet des machines virtuelles instanciées.
2. **Core** : Cette couche intermédiaire contient les composants de base de l'architecture OpenNebula.
   - **Le VM Image management** :chaque image disque est enregistrée et gérée par un catalogue d'images centralisé.
   - **Le "VM Management"** :chaque machine virtuelle est caractérisée par plusieurs caractéristiques par exemple : CPU, Mémoire, disque et réseau virtuelle et peut être lancé sur un hyperviseur d'hôte physique disponible.
   - **Le "Virtual Network Management"** :il est possible de créer plusieurs réseaux virtuels en liaison avec différentes interfaces physiques ; en attribuant différentes adresses IP statiques ou dynamiques.
   - **Le "Host Management"** :chaque hôte physique est caractérisé par plusieurs caractéristiques dont le CPU, mémoire, disque et le type d'hyperviseur (KVM, XEN, VMware)et peut héberger plusieurs instances de machines virtuelles.

3. **Drivers** : C'est la couche inférieure de l'architecture, elle contient les pilotes utilisés pour communiquer avec les plates-formes de cloud computing à différents niveaux :
   - **Transfer Driver (Pilotes de transfert)** s'occupent du traitement de l'image tel que le clonage, la suppression et la création de lieux d'échange.
   - **Virtual Machine Drivers** gèrent concrètement le cycle de vie d'une machine virtuelle par des commandes émises par la CLI. Les responsabilités de ces conducteurs comprennent le déploiement, l'arrêt, et la migration des machines virtuelles.
   - **Information drivers (Pilote d'information)** exécutent des scripts dans des hôtes physiques pour recueillir des informations sur eux, tels que la mémoire totale, la mémoire libre, CPU totales, CPU consommés, et ainsi de suite.

2.9.2 OpenStack

FIGURE 2.5: Logo OpenStack

OpenStack est une plate-forme open Source de Cloud Computing privés et publics. Elle est très dynamique, présentant plusieurs nouvelles fonctionnalités à chaque nouvelle version du logiciel. Cependant, elle est fragmentée en plusieurs modules logiciels (projets de OpenStack) avec les bibliothèques d'interface dédiées[74]. Cette fragmentation durcit le processus d'installation, la gestion de la plate-forme et augmente la complexité du système. D'autre part, elle interagit avec plusieurs applications tierces, utilise des interfaces REST et offre des bibliothèques d'interface comme OCCI[71] , AWS EC2[32] et S3[10].

2.9.2.1 Architecture d'OpenStack

L'architecture d'OpenStack est composée de plusieurs éléments que sont :

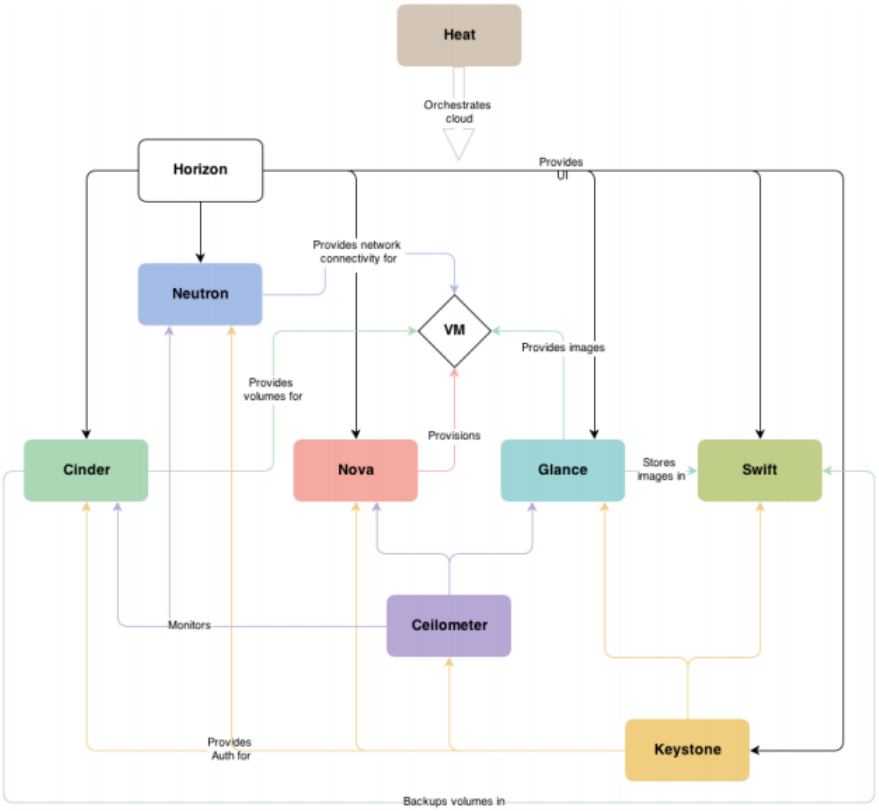

FIGURE 2.6: Architecture d'OpenStack[74]

1. **Swift** : Permet de gérer et de stocker une large capacité de données dans le Cloud avec une redondance pour assurer la tolérance aux pannes, ainsi que le basculement entre les différents objets de stockage.
2. **Nova Compute** : Est le contrôleur de toutes les activités liées aux machines virtuelles, il permet de gérer et configurer le réseau de ces dernières.
3. **Glance** : Est un système qui fournit les services de stockages, de découvertes, d'enregistrements, et de distributions des images disques des machines virtuelles.
4. **OpenStack Horizon (OpenStack Dashboard)** : Fournit une interface graphique aux administrateurs et aux utilisateurs à fin de pouvoir accéder aux différents services, il facilite l'intégration avec les ressources OpenStack.
5. **OpenStack Keystone (OpenStack Identity)** : Est un projet OpenStack, qui fournit le service d'identité pour l'authentification, l'autorisation, jeton et aussi les services catalogue et politique pour une utilisation spécifique par les autres projets d'OpenStack. L'authentification dans Keystone, est la

confirmation de l'identité d'un utilisateur. Keystone fait l'authentification à travers un ensemble des affirmations spéciale telle que le nom d'utilisateur et le mot de passe. Après la fin de l'authentification, keystone donne à l'utilisateur un jeton pour démontrer que son identité a été authentifiée.

6. **Networking :Reseau (Neutron)** : Assure plusieurs rôles :
   - gestion d'un réseau virtuel au sein d'OpenStack ;
   - fournit un bloc complet pour la gestion de réseaux complexes dans les infrastructures Cloud ;
   - fournit une connectivité réseau entre une sélection d'interfaces réseaux (interface virtuelle d'un service Compute, interface sur un service de loadbalancing, etc.) ;
   - expose des API décrivant la connectivité réseau et la configuration entre les interfaces.
7. **OpenStack Storage (cinder)** : Fournit des services qui permettent de gérer et d'accéder aux volumes de stockage de blocs pour une utilisation par les instances de machines virtuelles.
8. **Telemetry Service (Ceilometer)** : Ce projet vise à devenir l'infrastructure nécessaire pour recueillir des mesures au sein d'OpenStack, de telle sorte qu'il n'aurait pas besoin de deux agents à écrire pour recueillir les mêmes données. Son principale objectif est la surveillance. À cet effet, le ceilometer devrait être en mesure de partager les données recueillies avec une variété de consommateurs [74].

2.9.3 Eucalyptus

FIGURE 2.7: Logo Eucalyptus

Eucalyptus[36] est également une plateforme open source de cloud computing. Il permet d'exécuter des VMs dans un IaaS virtualisé. Actuellement, Eucalyptus prend en compte des IaaS munis des systèmes de virtualisation Xen, kvm, VMware. Il organise l'IaaS de façon hiérarchique : les machines au niveau des feuilles, les

clusters (groupe de machines) au niveau intermédiaire et le cloud (ensemble de clusters) à la racine. Son fonctionnement est organisé de la même façon. Il associe à chaque niveau de l'IaaS un composant précis

2.9.3.1 Architecture de Eucalyptus

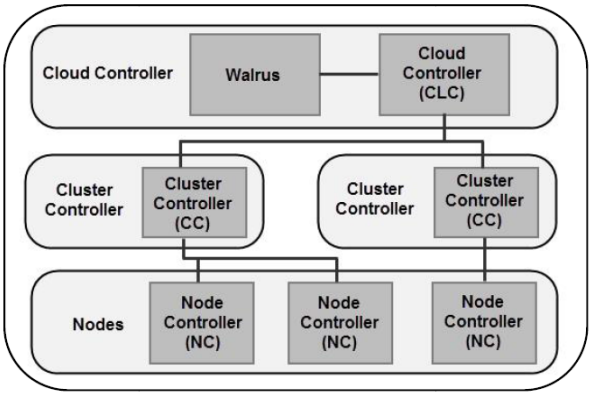

FIGURE 2.8: Architecture d'Eucalyptus

Une configuration de cloud fondée sur Eucalyptus se compose de cinq types de composants principaux.

1. **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. Il permet de surveiller la disponibilité des ressources sur les différentes composantes de l'infrastructure du cloud.
2. **Node Controler** : Le rôle du node est d'héberger KVM, il sert ainsi d'hyperviseur pour les machines virtuelles qui sont déployées. Les machines virtuelles fonctionnant sur l'hyperviseur sont appelées des instances. Eucalyptus permet aussi d'utiliser d'autres types d'hyperviseurs comme XEN. Le contrôleur de nœud fonctionne sur chaque nœud et est chargé de vérifier le cycle de vie des instances en cours d'exécution sur le nœud.
3. **Cluster Controller** : Ce contrôleur sert à déployer et gérer les différents contrôleurs de nœud. Il sert également à gérer la mise en place du réseau entre les instances des différents nœuds. C'est lui qui communique l'ensemble des informations au contrôleur du cloud. Il reçoit les requêtes de déploiement des instances, décide sur quel contrôleur de nœud les instances seront déployé aussi il contrôle le réseau virtuel entre les instances.
4. **Walrus** : Il assure 3 fonctions principales :
   - le stockage des images de machines virtuelles ;

- le stockage des images prises en fonctionnement à un instant précis ;
- le stockage des fichiers et les services.

5. **Storage Controller** : Ce composant fonctionne avec le composant Walrus et permet de stocker les images des machines virtuelles et les données des utilisateurs.

2.9.4 Synthèse

Chacune des plateformes de Cloud Computing "Open-source" cités ci-dessus a ces propres caractéristiques. Afin de mieux les comprendre, nous allons résumer les plus importantes caractéristiques dans le tableau 2.1 :

	OpenNebula	**OpenStack**	**Eucaliptus**
Type de service	IaaS	IaaS	IaaS
Modèle de déploiement	Privé/public/hybride	Privé/public	Privé/public
Architecture	Centralisé ; Trois composants, Minimum deux serveur	Intégration des deux composants OpenStackobject et OpenStack-compute	Hiérarchiqu ; cinq composants ; Supporte multiple cluster ; Minimum deux serveurs
Technologie de virtualisation supportée	Xen, KVM, VMware	Xen, KVM, VMware, LXC, UML	Xen, KVM, VMware(ESX/ESXi)
Stockage(Transfert image)	- Systèm, Image et fichier de stochage de donnés - SAN/NAS Server, vmfs, LVM, Ceph	Swift(http/s), Unix File System(ssh)	Walrus (http/s)
Authentification	User/Password, SSH, X 509, LDAP	X 509, LDAP	LDAP, CHAP
Compatibilité et interopérabilité	Compatible avec les API Amazon EC2, S3, libvirt et les API OCCI (multi-plateforme)	compatible avec les API Amazon EC2, S3 et OCCI	compatible avec les API Amazon EC2 et S3
Langage de programmation	Java, Ruby, C++	Python	Java, C, et Python
Flexibilité	Très fléxible	Flexible	Moins fléxible
Support de l'OS d'installation Linux	linux	linux	linux

TABLE 2.1: Caractéristiques des plateformes open source

Tout en analysant les différentes plates-formes de gestion de cloud open source, nous observons qu'ils impliquent différentes philosophies lors de la conception de leurs plates-formes de gestion de cloud. Nous constatons que les différences sont aussi liés à la façon de stocker l'image, la méthode d'authentification et le langage

de programmation. Cette diversité est causée par l'absence de normes architecturales bien définis pour la banalisation des systèmes IaaS. Chaque plateforme IaaS tend à fournir des fonctionnalités distinctes et être compatible avec des services spécifiques de tiers afin de monopoliser le marché et imposer ses technologies standards.

2.10 Grid Computing et Cloud Computing

Le Grid Computing ressemble au Cloud Computing dans certains concepts comme : un grand nombre de serveurs, communication réseau très importante, stockage généralement distribué, etc. Mais ces deux techniques sont assez différentes.

Dans le cas du Grid Computing il s'agit surtout de mettre à disposition de quelques équipes des ressources informatiques très puissantes pour des périodes de temps données. Par exemple le projet Grid 5000 en France [44], qui est utilisé pour faire des calculs parallèles sur de très gros volumes de données.

Dans le cas du Cloud Computing, sont souvent notées les notions d'élasticité, de disponibilité et de virtualisation de la ressource. Avec le Cloud Computing l'utilisateur ne sait pas trop sur quel(s) serveur(s) fonctionne son application. Il paye en fonction de l'utilisation[23].

2.10.1 L'intégration des grilles et des Clouds

Les travaux de recherches [[86],[6],[69],[83]] ont exposé quelques problématiques sur l'intégration des deux technologies que sont :

- L'intégration de la virtualisation dans des infrastructures existantes de grilles.
- Le déploiement des services grille sur des infrastructures virtualisées existantes.
- L'utilisation des composants open-sources pour la construction des Clouds.
- Les technologies grilles pour les Clouds fédérés.

A l'issue de ces préoccupations citées ci-dessus, il existe plusieurs approches dans le domaine de l'intégration de la grille et le cloud[87]

2.10.1.1 La Grille sur le Cloud (Grid over Cloud)

Une approche de Cloud IaaS est adaptée pour construire et administrer un système grille flexible (Exemple : Le projet "StratusLab"[60]). Dans cette approche, le "Middleware" des services grille s'exécute sur des machines virtuelles de l'infrastructure Cloud. Le seul problème est la performance, puisque la virtualisation entraine des pertes de performance comparé à l'utilisation directe des ressources physiques.

2.10.1.2 Le Cloud sur la Grille (cloud over grid)

Une des solutions utilisée dans cette approche, est l'utilisation de la soumission des jobs de la grille et la gestion des ressources pour le lancement des images d'exécution de jobs. Cette solution est utilisée par le centre de données "CNAF [1]" d' "INFN [2]" en Italie pour leur service "WNoDeS [3]" . Une autre solution consiste à utiliser l'infrastructure stable de la grille pour construire un environnement Cloud [103], dans ce cas, un ensemble de composants grille est utilisé pour administrer les machines virtuelles du Cloud.

2.10.1.3 Grille de clouds (grid of clouds)

Cette approche poursuivie par le projet EGI. Il a pour objectif de créer une fédération des différents sites de Cloud en exécutant différentes piles de logiciels, qui doivent interopérer la publication d'informations de découverte de ressources et la publication d'information de comptabilité/contrôle des données pour les services de EGI centrales. Cette approche étudie des solutions pour un marché central des images, pour une authentification unifiée et d'autorisation et pour un système de cloud broker [87].

2.11 Conclusion

Le Cloud computing marque une réelle avancée vers l'infrastructure informatique dématérialisée. Il fournit des ressources informatiques, logicielles ou matérielles, accessible à distance en tant que service.

Dans ce chapitre nous avons présenté les différentes définitions et caractéristiques du Cloud Computing afin de montrer ce qu'il a apporté de nouveau dans tous les domaines Informatiques.

Nous avons présenté trois différentes plateformes de Cloud IaaS "Opensource". Dans les Cloud "Open-source" (contrairement aux Cloud Commerciaux), les détails d'implémentation et de technologie ne sont pas cachés à l'utilisateur, facilitant ainsi aux chercheurs d'exploiter ces informations pour leur expérimentations et calculs. Pour faciliter le choix entre ces plateformes aux chercheurs, nous avons fait une comparaison entre elles.

Nous avons aussi vu que la Grille et le Cloud sont deux technologies relativement récentes et en pleine expansion et que beaucoup de travaux de recherche sont intéressés d'intégrer les deux technologies. Malgré cela, il reste encore de nombreuses contraintes qui freinent leur évolution au sein d'Internet. La Grille,

1. https ://www.cnaf.infn.it/
2. http ://web2.infn.it/
3. http ://web2.infn.it/wnodes/

son principal frein réside dans la complexité et le prix du matériel requis. D'autre part le Cloud-Computing rencontre de nombreux défis liés à la gestion des donnés, la sécurité, l'élasticité et l'interopérabilité entre les clouds [89][88].

Les fournisseurs du cloud sont devenus très nombreux sur le marché dans ces dernières années. Chacun d'eux favorise sa propre infrastructure du cloud, et des standards et les formats incompatibles pour accéder au cloud, ce qui les empêche de se mettre d'accord afin de supporter les applications du cloud computing d'une manière standardisée. Cela crée le problème d'interopérabilité entre clouds. Et comme le fournisseur du cloud ne pourrait pas être en mesure de satisfaire tous les besoins de l'utilisateur et que ce dernier se trouve lié un cloud offert par un seul fournisseur. Les clients de cloud computing sont probablement à la recherche d'un cloud interopérable, où il peuvent avoir le plein contrôle sur l'endroit où déployer leurs applications et migrer leurs services facilement quand il est nécessaire, sans investissement de développement supplémentaire. Donc un scénario de fédération de clouds doit être envisagé. Dans notre thèse nous nous somme intéressés de résoudre le problème d'interopérabilité dans un système multi-cloud. Donc dans le prochaine chapitre, nous allons voir en détail la technologie « multi-cloud ».

Chapitre 3

Contexte Multi-Cloud

3.1 Introduction

Avec le nombre de fournisseurs de cloud en pleine expansion, les utilisateurs se voient confronter au défi de sélectionner un cloud et/ou des ressources appropriées, pour exécuter leurs applications. Ceci est une tâche difficile, parce que les clouds proposent une vaste gamme de ressources. En outre, ces ressources sont généralement adaptées à des fins différentes, et elles sont souvent soumises à plusieurs contraintes. Puisque les clouds peuvent tomber en panne, ou connaitre des limites en matière d'évolutivité, un scénario d'un cloud interopérable « système multi-cloud » doit être envisagé afin de régler le problème de verrouillage propriétaire «vondor lock-in» et satisfaire toutes les requêtes des utilisateurs.

3.2 L'interopérabilité dans le multi-cloud

L'interopérabilité désigne généralement la capacité de différents systèmes hétérogènes de fonctionner et d'interagir ensemble. Dans un système multi-cloud, l'interopérabilité peut être définie comme la capacité de comprendre les formats d'application de chacun, par exemple : le modèles SLA, les formats d'authentification, jeton d'autorisation, les données d'attributs et d'autres [48] ,afin de permettre aux différents clouds de coopérer ou de s'interopérer [81].

Dans le Cloud Computing, l'interopérabilité, la compatibilité et la portabilité sont des termes étroitement liés et peuvent souvent être confondus. L'auteur dans [21] précise les similitudes et les différences entre ces termes :

- **L'interopérabilité dans le Cloud** permet aux multiples fournisseurs de Cloud de travailler ensemble. Alors que La compatibilité dans le Cloud et la portabilité répondent à la question «comment ?».

- **La compatibilité dans le Cloud** signifie que l'application et les données peuvent travailler de la même manière quel que soit le fournisseur de Cloud.
- **la portabilité dans le Cloud** est la capacité d'avoir facilement déplacé et réutilisé les données et les composants d'application dans le Cloud, quel que soit le choix du fournisseur de Cloud, système d'exploitation, le format de stockage ou l'API.

Pour assurer l'interopérabilité dans un système multi-cloud, un nouveau service qui s'appelle « service cloud broker » est nécessaire.

3.3 Le Cloud Broker

3.3.1 Définition

Le cloud broker a pour mission d'aider le client à choisir les fournisseurs et les technologies de Cloud Computing tout en garantissant la performance des architectures proposées, la flexibilité et la cohérence du Système d'Information, la sécurité et la réversibilité des solutions et les architectures choisies. Tout ceci au moyen d'outils qui facilitent la sélection des solutions technologique [46].

3.3.2 Scénario d'utilisation du Cloud Broker

FIGURE 3.1: Scénario d'utilisation du cloud broker

Un consommateur de cloud peut demander le service d'un cloud broker au lieu de contacter directement un fournisseur de cloud. Le cloud broker peut créer un nouveau service en combinant plusieurs services ou en améliorant un service existant. Dans la figure 3.1 , les fournisseurs de cloud réels sont invisibles pour le consommateur de cloud et le consommateur de cloud interagit directement avec le cloud broker.

3.3.3 Les services d'un Cloud Broker

Selon les auteurs W.Bimlesh, J.Aditi et S.Bharti[98], les services principales que peut fournir un cloud broker sont :

- **Le service d'intermédiation** :un cloud broker améliore un service donné par l'amélioration de certaines capacités spécifiques et la fourniture de certains services à valeur ajoutée pour le consommateur.
- **Le service d'agrégation** :le cloud broker combine et intègre de multiples services dans un ou plusieurs services.
- **Le service d'arbitrage** :le cloud broker intègre les services qui ne sont pas fixés, et il permet la flexibilité de choisir parmi plusieurs fournisseurs.

3.3.4 Les fonctionnalités d'un cloud broker

Selon l'auteur V.Venturi[96], le cloud broker assure plusieurs fonctionnalités entre autres :

- **Interopérabilité / Abstraction** :les utilisateurs du cloud n'ont pas besoins d'utiliser plusieurs API ou clients, ou d'obtenir plusieurs informations d'identification d'accès.
- **La liaison(Matchmaking)** :le Cloud broker choisit les services de cloud computing qui correspondent aux besoins des utilisateurs, de sorte qu'ils n'ont pas besoins de passer par plusieurs catalogues.
- **Composition / Agrégation / Arbitrage** :Le Cloud broker combine et intègre multiples services dans un ou plusieurs nouveaux services. Il fournit l'intégration de données et assure le transfert sécurisé des données entre les consommateurs et les fournisseurs. Le Cloud broker peut choisir des services provenant de plusieurs fournisseurs de cloud.

3.4 L'Inter-Cloud

L'Inter-Cloud computing a été officiellement défini comme [18] : Un modèle de cloud qui a le but de garantir la qualité de service, tels que la performance et la disponibilité de chaque service. Il permet la réaffectation à la demande des ressources et le transfert de la charge de travail à travers un inter-fonctionnement des systèmes de cloud computing de différents fournisseurs de cloud. Il se base sur la coordination de chacune des exigences de qualité de service des clients avec chaque SLA des fournisseurs. Ainsi il utilise une interface standard.

3.4.1 Les avantages d'un environnement Inter-Cloud pour les clients

Les avantages d'un environnement Inter-Cloud pour les clients de cloud sont nombreux et peuvent être largement résumés comme suit [47] :

- **Diverses régions géographiques** : les principaux fournisseurs de services de cloud computing ont créé des datacenters dans le monde entier. Cependant, il est peu probable que n'importe quel fournisseur serait en mesure d'établir des datacenters dans tous les pays et les région administratives [15]. De nombreuses applications ont des exigences législatives de l'endroit où les données sont stockées. Ainsi, un centre de données dans une seule région de pays peut ne pas être suffisant, et les développeurs d'applications devront faire un contrôle précis (pays ou état) que d'où les ressources sont positionnées. Seulement en utilisant plusieurs Clouds, on peut avoir accès aux ressources si largement distribuées et fournir la performance et les services de législation conformes aux clients.
- **Une meilleure résistance de l'application** : au cours des dernières années, il ya eu plusieurs cas de pannes de services Cloud, y compris les principaux fournisseurs de Cloud [5][66].Les implications de manquement de l'un des datacenters par Amazon étaient très graves pour les clients qui comptaient uniquement sur cet emplacement. Dans une analyse post-mortem, Amazon a conseillé à leurs clients de concevoir leurs applications pour utiliser plusieurs datacenters pour la tolérance aux pannes [5]. Par ailleurs, dans le rapport de Berkeley sur le cloud computing, Armbrust et al. soulignent que l'éventuelle indisponibilité du service est l'inhibiteur numéro un à l'adoption de Cloud computing [40]. Ainsi, ils conseillent l'utilisation de plusieurs fournisseurs. Outre la tolérance aux pannes,l'utilisation des ressources de différents fournisseurs représente une assurance contre un fournisseur arrêté pour des raisons réglementaires ou juridiques.
- **Éviter l'enfermement propriétaire (vendor lock-in)** : en utilisant plusieurs Clouds en étant capable de transiter librement la charge de travail entre eux, un client de Cloud peut facilement éviter le vendor lock-in. Dans le cas où un fournisseur change une politique ou les prix qui nuisent à ses clients, ces derniers pourraient facilement migrer ailleurs.

3.4.2 Les avantages d'un environnement Inter-Cloud pour les fournisseurs

- **Étendre la demande** : être capable de décharger sur d'autres Clouds. Un fournisseur peut évoluer en termes de ressources. Un cloud devrait être prêt d'utiliser suffisamment de ressources pour répondre à sa charge attendue, et si la charge de travail augmente au-delà de ces limites, il peut louer des ressources provenant d'autres Clouds[15].par exemple les applications hébergées dans un Cloud s'exécutaient dans un autre Cloud.

- **Meilleur contrat de niveau de service SLA aux clients *(Le service-level agreement)*** : dans le pire sénario, s'il y'a une panne ou un manque de ressources, la charge de travail entrante peut être déplacée vers un autre Cloud. Donc un fournisseur de Cloud peut offrir de meilleurs SLA aux clients [47].
- Générer des revenus additionnels des clients existants par la vente de services au-delà d'empreinte géographique propre au fournisseur de services[99].
- Générer de nouveaux revenus provenant des autres fournisseurs de services en louant la capacité inutilisée selon les besoins[99].

Toutefois, la réalisation de tous ces avantages à la fois pour les fournisseurs de Cloud et les clients doit être faite sans violer les exigences des applications. Une application appropriée Cloud broker (composé de provisionnement et d'ordonnancement) devrait honorer les exigences en termes de performance, de réactivité et de considérations juridiques. Les approches existantes pour atteindre cet objectif varient en termes d'architecture, de mécanismes et de flexibilité.

3.5 L'Inter-Cloud broker

Tout au long de la littérature, le terme Inter-Cloud broker a été utilisé avec des significations différentes [47]. Dans la plupart des cas, cela signifie un service qui agit pour le compte du client pour l'approvisionnement des ressources et le déploiement des composants d'application [15][37][91]. Nous adhérons à cette idée générale pour définir un broker de gestion d'infrastructure comme une entité automatisée avec les responsabilités suivantes :

- L'approvisionnement automatique des ressources et la gestion de plusieurs clouds pour une demande de virtualisation donnée. Typiquement, cela inclurait l'allocation et la libération des ressources (par exemple une machine virtuelle (VM) et le stockage).
- La planification et l'équilibrage de charge sur les demandes entrantes vers les ressources allouées.

3.6 Les taxonomies multi-clouds

Dans la littérature, il existe plusieurs taxonomies des multi-clouds, où nous pouvons confondre les thèmes suivants : multi-cloud, inter-cloud et le cloud fédéré. Dans cette section nous allons discuter quelques taxonomies des multi-cloud afin de mieux comprendre la classification des solutions multi-clouds existantes.

3.6.1 Taxonomie 1

Les auteurs G.Nikolay et B.Rajkumar de l'article [47] proposent une classification en fonction de l'architecture des Inter-cloud. Dans cette classification, on note que l'Inter-Cloud est subdivisé en deux grandes familles que sont : la fédération et le multi-cloud. Et chaque famille est elle-même subdivisée en deux sous-familles.(Voir figure 3.2)

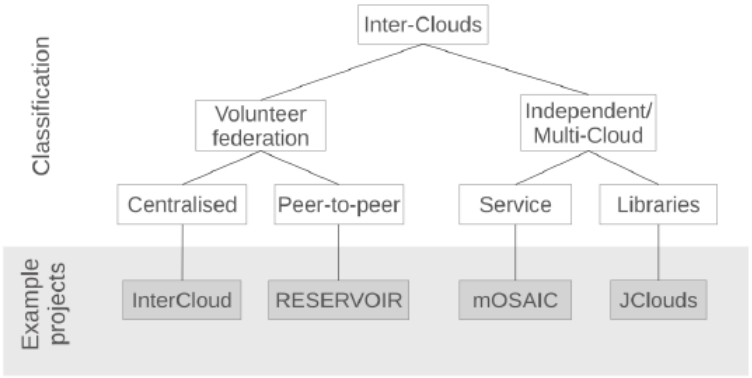

FIGURE 3.2: Classification architecturale des Inter-Cloud[47]

3.6.1.1 fédération de clouds (Volunteer federation)

Une fédération est atteinte lorsqu'un ensemble de fournisseurs de Cloud interconnectent volontairement leurs infrastructures pour permettre le partage des ressources entre eux [84][37][3] afin d'obtenir des avantages économiques [99]. La fédération de Cloud permet aux fournisseurs de Cloud d'utiliser (louer) les ressources de l'extérieur lorsque la demande dépasse l'offre. Elle permet également aux fournisseurs de développer leurs empreintes géographiques sans avoir à déployer leurs propres ressources informatiques dans le monde entier. Il ya beaucoup de défis redoutables inhérentes dans la fédération de cloud. Sur le plan opérationnel, il est difficile de déployer et migrer des machines virtuelles partout dans le monde avec souplesse et rapidité. Il est assez difficile de le faire dans un seul fournisseur de services, et encore moins entre les fournisseurs de services. De plus il est difficile pour les fournisseurs de services de déterminer et partager la charge automatiquement.

Du point de vu architecturale les fédérations de Clouds peuvent être classées comme suit :

a. **Fédération Centralisée :** dans ce groupe d'architectures, il y'a une entité centrale qui effectue et facilite l'affectation des ressources. Habituellement, cette entité centrale agit comme un référentiel où les ressources de Cloud dis-

ponibles sont enregistrées, mais elle peut aussi avoir d'autres responsabilités en agissant comme une place de marché pour les ressources.(voir figure 3.3)

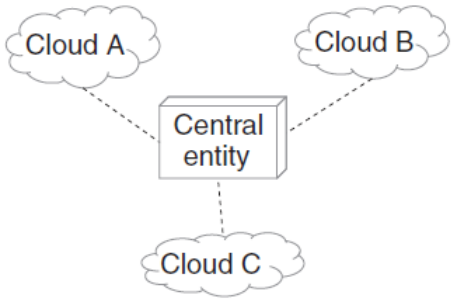

FIGURE 3.3: Fédération inter-cloud centralisé

b. **Fédération pair-à-pair :** dans les architectures de ce groupe, les Clouds communiquent et négocient directement entre eux sans l'intervention des médiateurs. (voir figure 3.4)

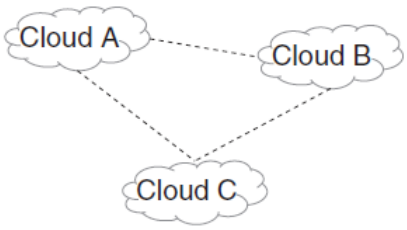

FIGURE 3.4: Fédération inter-cloud pair à pair

3.6.1.2 Multi-Cloud

L'approche multi-Cloud désigne l'utilisation de plusieurs Clouds. Elle est indépendante du fournisseur de cloud et utilise des ressources provenant de différents clouds. Contrairement à une fédération de Clouds, un environnement multi-Cloud n'implique pas l'interconnexion des bénévoles et le partage des infrastructures des fournisseurs. Les clients ou leurs représentants sont directement responsables de la gestion de l'approvisionnement des ressources et de la planification [37].

Les multi-Clouds peuvent être classées comme suit :

a. **Service multi-Cloud :** l'approvisionnement des applications est réalisé par un service Cloud broker qui se charge de placer et d'exécuter l'application dans plusieurs Clouds. (voir figure 3.5)

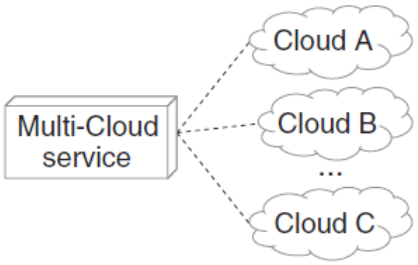

FIGURE 3.5: Service Multi-Cloud

b. **Librairies multi-Cloud :** souvent des applications cloud broker personnalisées prennent directement en charge l'approvisionnement et l'ordonnancement des composants d'application à travers les clouds nécessaires. Typiquement, ces approches font usage des librairies Inter-Cloud qui facilitent l'utilisation de plusieurs clouds et de manière uniforme.(voir figure 3.6)

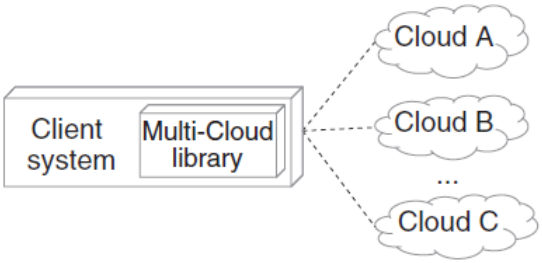

FIGURE 3.6: Librairie Multi-cloud

3.6.2 Taxonomie 2

L'auteur P.Dana de l'article [78] introduit une autre classification basée sur les différentes catégories de multi-Clouds (voir figure 3.7). Ces catégories sont créées selon le modèle de déploiement. Au premier niveau, la catégorie multi-clouds est subdivisée en trois familles : fédération de Clouds, multi-Clouds et Inter-Clouds. Le deuxième niveau est lié à l'organisation des différentes interactions entre les Clouds impliqués. Les troisième et quatrième niveaux font référence à l'organisation de l'architecture. Le terme horizontal dans les sous-familles fédérations horizontales et multi-clouds horizontales se réfèrent au niveau du déploiement des

services de Cloud. Par exemple, on agrège plusieurs clouds au même niveau de déploiement SaaS ou PaaS ou IaaS. Tandis que le terme hiérarchie dans les sous-familles fédérations hiérarchiques et multi-Clouds hiérarchiques utilise l'agrégation de plusieurs Clouds indépendamment de leur niveau de déploiement. Par exemple, on peut agréger des ressources provenant de IaaS et PaaS.

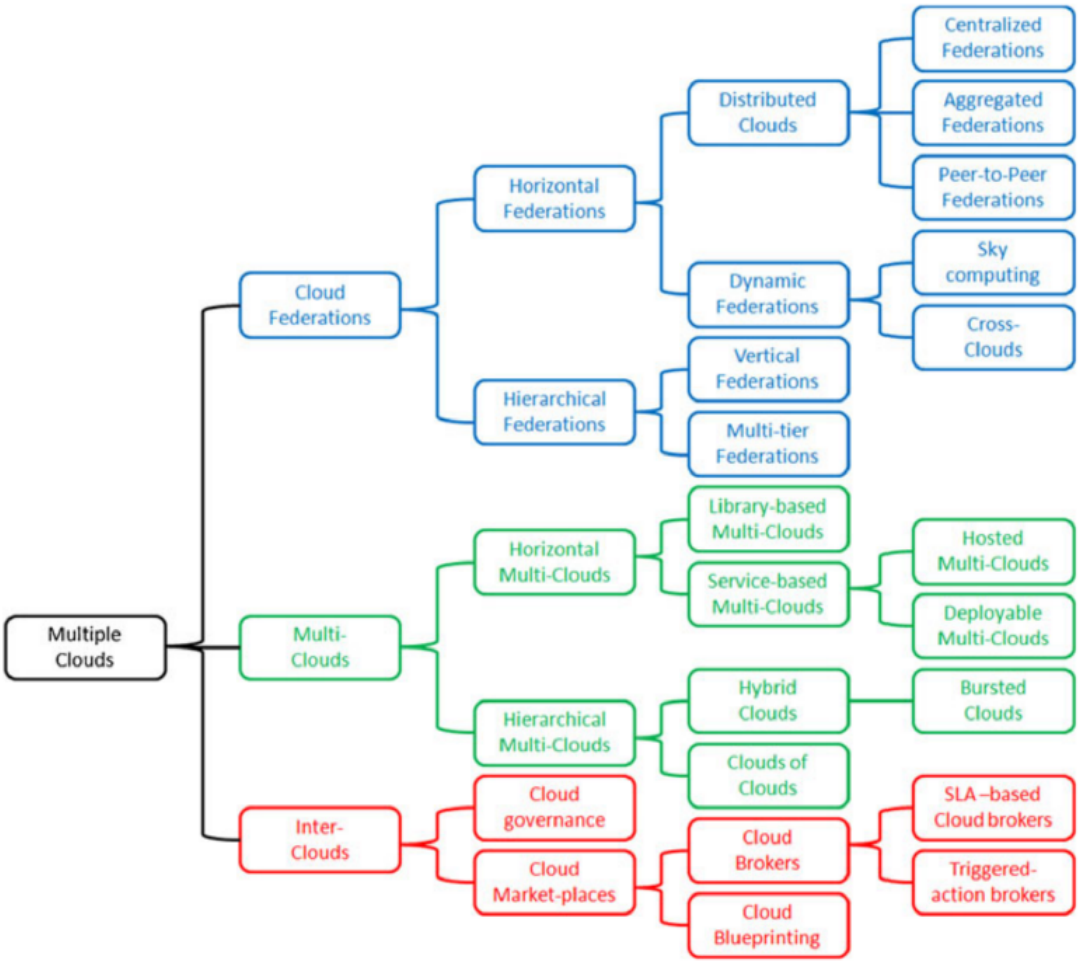

FIGURE 3.7: Relation entre les catégories multi-clouds[78]

3.6.3 Discussion

Les deux travaux [78] et [47] font tous deux une classification multi-Clouds. Les travaux de l'auteur [78] mettent la fédération, les multi-Clouds et les Inter-Clouds au même niveau (voir figure 3.7). Tandis que les auteurs [47] placent les Inter-Clouds au-dessus de la fédération et les multi-Clouds qui se trouvent au même

niveau (voir figure 3.2).

Les auteurs de l'article [47] soulignent dans leur classification que les fournisseurs de Cloud collaborent volontairement (fédération de Cloud) ou pas (multi-Clouds). Les deux catégories fédération de Cloud et multi-Clouds peuvent être distinguées par le type de modèle de déploiement [37]. La différence est faite par le degré de collaboration entre les clouds en question et par la façon par laquelle l'utilisateur interagit avec les clouds. Le premier modèle, appelé cloud fédéré, suppose un accord entre les fournisseurs de cloud. Dans ce modèle, le consommateur de service n'est pas au courant du fait que le fournisseur de cloud qu'il paie l'utilisation des services des autres fournisseurs. Le second modèle, appelé multi-clouds, suppose qu'il n'y a pas d'accord a priori entre les fournisseurs de clouds et un tiers (consommateur de cloud) qui est responsable des services.

Dans les multi-clouds l'un des principaux problèmes est la portabilité des applications à travers les clouds. Tandis que le principal problème dans la fédération de clouds est l'interopérabilité [97] [79].

Notre solution **"DZ-CBroker"** se positionne dans la sous-famille "service" de la famille "multi-clouds" dans la taxonomie 1.

3.7 Solutions multi-clouds

3.7.1 Les standards

Il y'a plusieurs efforts de standardisation dans le domaine du Cloud computing qui ont été établis selon les modèles de services SaaS, PaaS et IaaS. Dans cette section nous allons citer quelques standards qui ont fait les solutions de modèle de service IaaS :

3.7.1.1 VMAN

Le standard de la gestion de la virtualisation (Virtualization Management) (VMAN)[30] du Distributed Management Task Force (DMTF) [1] a effectué une extension du standard DMTF pour la gestion de ressources réelles pour couvrir la gestion de ressources virtuelles. Il comprend le Open Virtualization Format (OVF) [2] qui est un standard pour les formats des images qui vont être chargées dans l'IaaS en gérant le système.

1. DMTF : https ://www.dmtf.org/about
2. OVF : https ://www.dmtf.org/standards/ovf

3.7.1.2 CIMI

Le Cloud Infrastructure Management Interface (CIMI)[27] est un groupe de travail au sein du consortium DMTF qui vise à gérer des ressources IaaS. Il implémente une interface REST et définit des APIs rendus aux formats XML et JSON. CIMI vise à fournir un support au standard OVF.

3.7.1.3 OCCI

L'Open Grid Forum (OGF)[33] est une autre initiative ouverte qui vise à la création d'une solution pratique pour interfacer les Clouds de type IaaS existants. OGF a défini Open Cloud Computing Interface (OCCI) pour fournir un accès uniforme aux ressources IaaS existantes. L'objectif principal de l'OCCI est la création de cloud hybride exploitant les environnements de clouds indépendants des fournisseurs et des intergiciels de plateformes.. OCCI définit des spécifications pour les éléments de base de cloud et des interfaces y compris un modèle utilisant l'API REST pour accéder, utiliser et gérer les clouds.

3.7.1.4 CDMI

Le Cloud Data Management Interface (CDMI)[92] est un standard proposé par l'organisme Storage Networking Industry Association (SNIA). CDMI définit une API REST pour effectuer des opérations CRUD (Create Read Update Delete) sur des données à partir d'un environnement de stockage dans le cloud. Il définit également une API pour la découverte des stockages dans le cloud et la gestion des conteneurs de données.

3.7.1.5 Etat actuel

Si nous regardons l'histoire de l'informatique, nous nous rappelons de l'importance de l'ouverture par rapport au coût élevé des plateformes propriétaires. Dans un monde parfait, lorsqu'on développe une application en Cloud, la meilleure façon pour lutter contre le coût élevé et la dépendance vis-à-vis d'un seul fournisseur est d'utiliser les standards ouverts. La plupart des efforts de standardisation effectués dans le domaine de Cloud computing sont destinés aux niveaux IaaS et PaaS. En résumé, un tel idéal n'est pas encore adopté dans le jeune marché de Clouds. Le problème est que la plupart des vendeurs rendent le processus d'adoption de standard difficile, frustrant et tellement long que certaines organisations abandonnent.

3.7.2 Vue d'ensemble des solutions existantes multi-Cloud

Nous allons présenter et analyser dans cette section certains travaux qui ont été effectués dans le domaine du multi-cloud.

3.7.2.1 RESERVOIR

Le projet européen de recherche RESERVOIR[85] vise à développer une infrastructure orientée services permettant l'interopérabilité dynamique entre les fournisseurs de Cloud. RESERVOIR sépare le rôle du fournisseur de service de celui du fournisseur d'infrastructure. En effet, le fournisseur de service interagit avec les utilisateurs finaux et répond à leur besoins. Les fournisseurs de service ne possèdent pas les unités de calcul mais les louent auprès des fournisseurs d'infrastructures qui interagissent les uns avec les autres de manière transparente pour créer des pools de ressources. La figure 3.8 illustre une architecture de gestion des sites RESERVOIR.

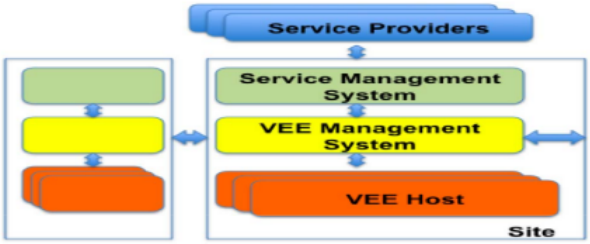

FIGURE 3.8: Architecture de gestion de sites RESEROIR[85]

Les fournisseurs d'infrastructures exploitent des sites RESERVOIR qui possèdent et gèrent l'infrastructure physique sur laquelle les applications sont exécutées. La fédération des sites RESERVOIR qui collaborent forme un cloud de RESERVOIR.

3.7.2.2 CONTRAIL

Le projet européen de recherche CONTRAIL propose une plateforme qui vise à mettre en œuvre une infrastructure IaaS et une plateforme PaaS. Il permet aux fournisseurs de Clouds d'intégrer de manière transparente les ressources des autres Clouds avec leur propre infrastructure pour fédérer plusieurs Clouds en favorisant la portabilité des applications d'un Cloud à l'autre. L'architecture de CONTRAIL est illustrée dans la figure 3.9.

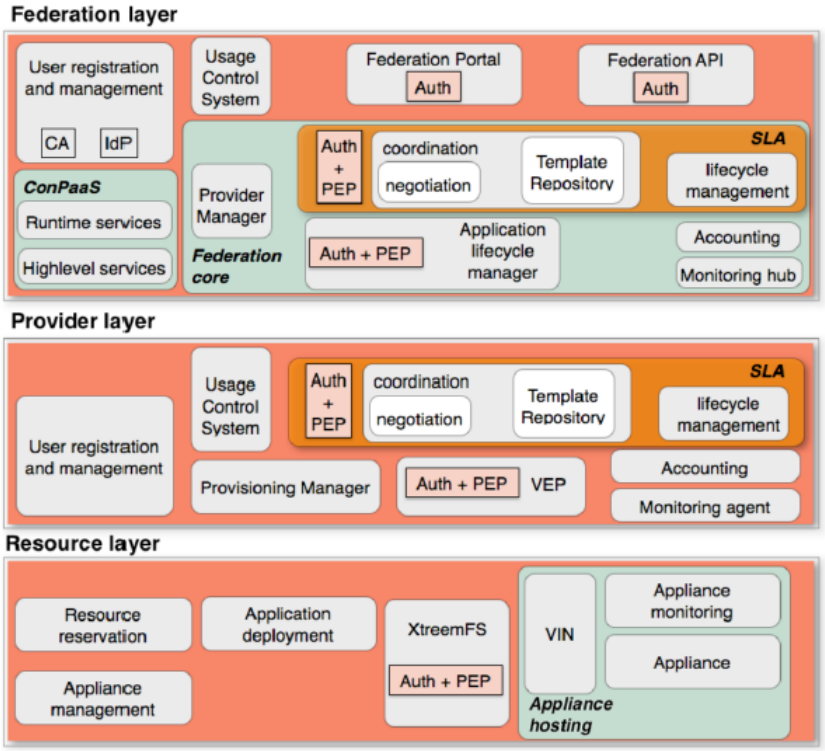

FIGURE 3.9: Architecture du CONTRAIL [17]

3.7.2.3 STRATOS

STRATOS[77] est un projet de recherche supporté par le Engineering Research Council of Canada (NSERC), Ontario Centre of Excellence (OCE) et Amazon Web Service. Le but de ce projet est de fournir un Cloud Broker au niveau IaaS permettant d'approvisionner des ressources provenant de plusieurs Clouds. Il supporte le déploiement et la gestion des applications à l'exécution. STRATOS permet de déployer des applications, tout en définissant sur ces dernières des exigences qui sont importantes en termes d'indicateurs de performance (Key Performance Indicators) ou KPI. Ces exigences sont transformées en un ensemble de requêtes de demande de ressources qui sont évaluées en fonction de l'offre des fournisseurs de cloud, afin de sélectionner la meilleure offre. Apache Stratos est testé sur les fournisseurs IaaS suivantes : AWS EC2, OpenStack et vClouds. Cependant, il est très facile d'étendre Apache Stratos à soutenir tout IaaS qui est pris en charge par jclouds Apache. L'architecture de STRATOS est illustrée dans la figure 3.10.

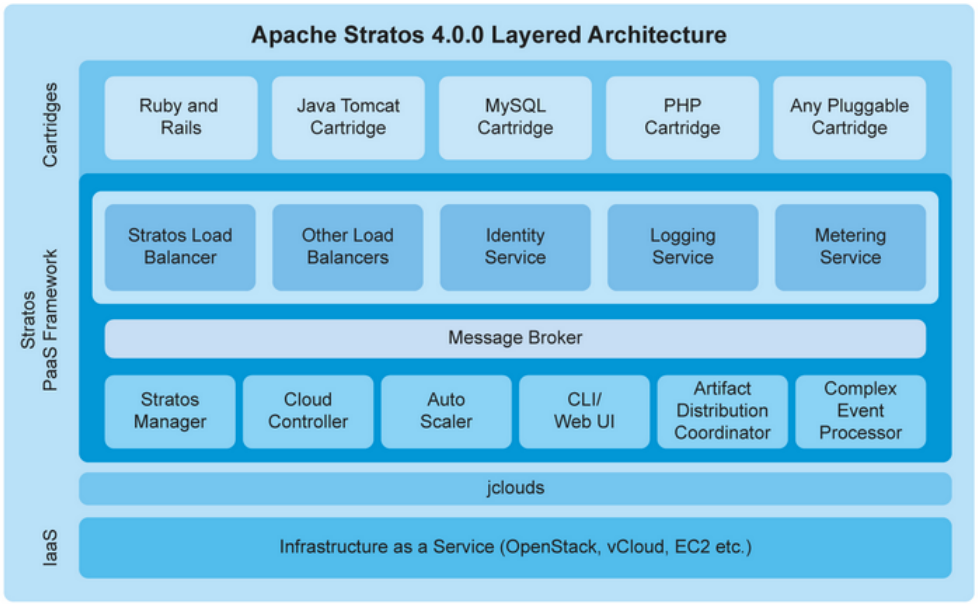

FIGURE 3.10: Architecture du STRATOS [93]

3.7.2.4 CompatibleOne

CompatibleOne[85]est un projet libre initié entre l'industrie et la recherche. CompatibleOne est un Cloud Broker qui fournit un middleware interopérable permettant de décrire et de fédérer des ressources provenant de clouds hétérogènes. Il permet aux développeurs de combiner différents services provenant de plusieurs fournisseurs. Il supporte différents types de ressources qui peuvent être fournies aux niveaux IaaS et PaaS. CompatibleOne[102] est une plateforme d'exécution qui fournit un modèle permettant de décrire les ressources. Ce modèle qui est nommé CompatibleOne Resource Description System (CORDS), permet de décrire une application à base d'objets. La plateforme d'exécution nommée Advanced Capabilities for CORDS (ACCORDS) est un système pour approvisionner et déployer des applications dans le cloud. La plateforme CompatibleOne est basée sur deux Standards CDMI et OCCI. Elle utilise OCCI comme modèle de communication via REST et CDMI pour l'accès aux données de stockage. La figure 3.11 illustre l'architecture de la plateforme CompatibleOne.

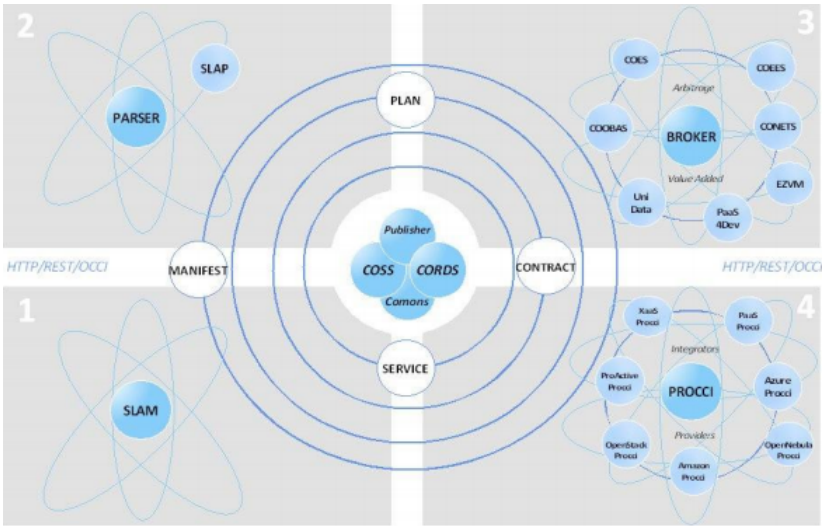

FIGURE 3.11: Architecture de CompatibleOne[85]

3.7.2.5 OPTIMIS

Le projet européen de recherche OPTIMIS[37] a pour objectif de fournir une plateforme ouverte, évolutive et fiable pour la fourniture de services flexibles. Il fournit des services auto-adaptatifs et des infrastructures non seulement basés sur la performance, mais aussi sur les aspects tels que la confiance, le risque et le coût. Un des principaux résultats du projet OPTIMIS est sa boîte à outils qui est constituée d'un ensemble de logiciels qui peuvent être assemblés de manière arbitraire et donc permettre la construction de diverses architectures de cloud. Il s'agit notamment des architectures pour fédérer plusieurs clouds et les architectures multi-clouds. L'architecture d'OPTIMIS est illustré dans la figure 3.12.

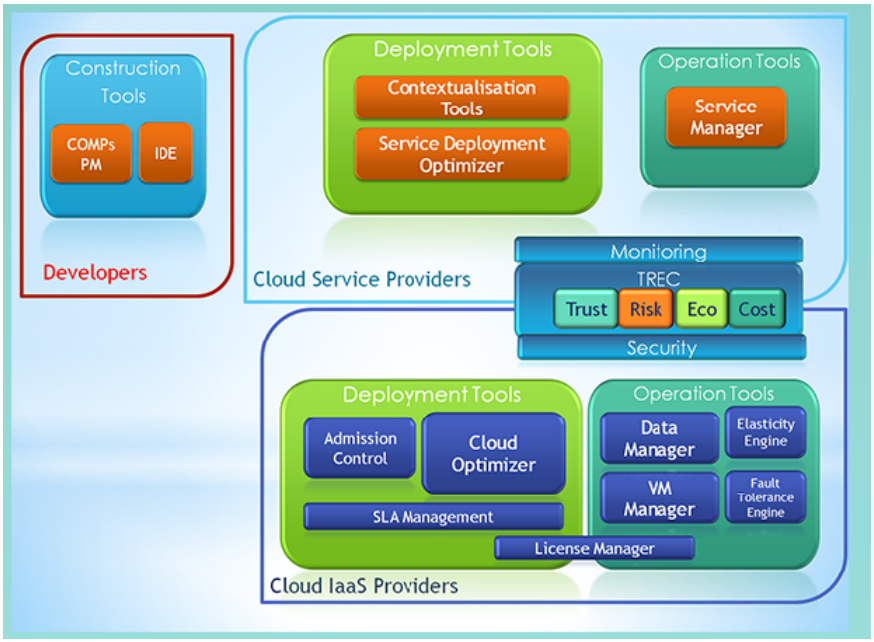

FIGURE 3.12: Architecture d'OPTIMIS [75]

3.7.2.6 Discussion

Il existe un certain nombre de travaux sur l'approvisionnement dynamique de ressources matérielles (machines physiques ou machines virtuelles) et les grappes (cluster) de machines[7]. Le travail sur l'approvisionnement dynamique de ressources dans le cloud computing peut être classifié en deux catégories. Les auteurs de l'article[67] ont adressé le problème d'approvisionnement de ressources à la granularité des machines virtuelles. D'autres auteurs [19] ont considéré l'approvisionnement de ressources à une granularité plus fine (cpu, mémoire, dique, réseau). Dans le contexte de notre travail, nous considérons l'approvisionnement à la granularité des machines virtuelles.

RESERVOIR [85] supporte deux modes d'approvisionnement : explicite et implicite. Le mode explicite exige qu'au moment du déploiement, le fournisseur de service spécifie précisément les besoins en capacité de l'application dans des conditions de charge de travail spécifiques. Le mode implicite couvre les besoins en capacité pour les applications de services non calibrés. Dans ce mode, le fournisseur de services ne peut avoir que des estimations de dimensions initiales de son service ou ne pas avoir d'estimation de dimensionnement du tout. En mode explicite, en utilisant les méta-données, le développeur a la possibilité d'exprimer le type de ressources, le nom du fournisseur de ressources, et l'emplacement de ces dernières. Cette méta-donnée peut être exprimée à différentes granularités (application ou composant). CompatibleOne[102] fournit deux modèles différents pour

approvisionner des ressources aux niveaux IaaS et PaaS. Ainsi, CompatibleOne offre la possibilité de déployer une application sur un IaaS et PaaS. Toutefois, CompatibleOne n'a pas un modèle uniforme pour approvisionner des ressources à la fois au niveau IaaS et PaaS. CompatibleOne[56] permet d'approvisionner des ressources à travers plusieurs fournisseurs de cloud par l'implantation du standard Open Cloud Computing Interface (OCCI).

Parmi les solutions existantes, il n'existe pas une couche d'abstraction de haut niveau qui cache la complexité d'approvisionnement de ressources sur plusieurs clouds. Les auteurs de l'article [39] ont abordé le problème de déploiement d'une grappe de machines virtuelles avec des configurations de ressources données à travers un ensemble de machines physiques. Tandis que d'autres auteurs de l'article [24] définissent une API Java permettant aux développeurs de contrôler et de gérer une grappe de machines virtuelles Java et de définir les politiques d'allocation de ressources pour ces grappes.

3.7.3 Solutions d'abstraction (Librairies)

Les librairies d'interface d'abstraction fournissent une collection d'implémentations pour le développement de systèmes de couches intermédiaires abstraites pour accéder à chaque composant de la plateforme sous-sous-jacente IaaS. Nous allons présenter quelque exemples existants d'abstraction de cloud tel que ; Deltacloud[28], jclouds[49] et Libcloud[58].

3.7.3.1 DeltaCloud

Deltacloud[8] est un projet open-source Apache qui vise à définir une API REST pour accéder à tout type de plate-forme de cloud qui exposent ses services au niveau IaaS. Entièrement écrit en Ruby (mais avec la possibilité d'exploiter les librairies clientes d'utiliser différentes langues), il offre à l'utilisateur la possibilité de tirer parti de Deltacloud classique, DMTF, CIMI et les API EC2 représentant l'API d'abstraction Deltacloud afin de gérer différentes plates-formes de Cloud computing. Cette API fonctionne comme un wrapper autour de ces plates-formes. Elle fait abstraire et cache leurs différences. Chaque fournisseur de cloud, a un pilote qui est capable d'interpréter l'API native de ce fournisseur de Cloud, afin que l'utilisateur n'ait pas besoin de traiter avec l'API spécifique directement. Cela signifie également que, si une API notamment publiée par un prestataire a été mise à jour et modifiée, le programme utilisateur ne serait pas affectée, car les spécifications Deltacloud sont garantis de rester stable. L'utilisateur peut aussi développer une application ayant une plate-forme spécifique, puis décider de migrer vers une autre plateforme compatible. Une liste des fournisseurs pris en charge par Deltacloud et des API relatives sont fournies en ligne. Deltacloud comprend

de nombreuses plates-formes de cloud comme Amazon EC2, IBM SmartCloud, GoGrid et OpenStack . Le fait d'être un projet open-source, les contributeurs peuvent ajouter librement leurs pilotes personnalisés à la liste et modifier ceux qui existent.

3.7.3.2 Apache Libcloud

Apache Libcloud est une librairie, entièrement écrite en Python, ce qui permet des interactions entre plusieurs fournisseurs de services de cloud populaires. En particulier, Libcloud fournit une API unifiée qui masque les différences entre différents fournisseurs grâce à une abstraction de haut niveau des services et des API exposées par les différents fournisseurs. Son principal objectif est de simplifier les efforts des développeurs à des produits de construction qui sont en mesure de travailler avec des services provenant de différents fournisseurs de Cloud computing, facilitant ainsi l'interopérabilité. La librairie Libcloud supporte un groupe étendu de plates-formes IaaS[58], y compris OpenNebula et OpenStack, et permet aux utilisateurs de gérer les ressources de cloud comme le stockage et le réseau.

3.7.3.3 Apache jclouds

Apache jclouds [49] est une librairie développée par Apache Software Foundation en Java et Python. Elle partage de nombreuses caractéristiques et concepts avec ceux de l'API Libcloud, les deux ayant le même objectif de définir une API de haut niveau pour accéder et gérer les services fournis par différents fournisseurs de cloud computing de manière uniforme. En particulier, jclouds propose une librairie open-source, utilisable via Java ou Clojure, permettant la création d'applications qui sont portables sur différentes plates-formes de cloud computing, donnant à l'utilisateur la possibilité d'utiliser encore des caractéristiques spécifiques aux clouds. les utilisateurs peuvent interagir avec les services offerts par les fournisseurs de cloud sans avoir à traiter directement avec les API REST et,en même temps, en utilisant une représentation abstraite des ressources cibles. La liste des fournisseurs pris en charge par jCloud est importantes comme Amazon, Azure et il a une liste compatibles avec des plates-formes IaaS, y compris OpenStack.

3.7.3.4 Discutions

Le tableau 3.1 résume les différentes caractéristiques des différentes API que sont : Deltacloud, jclouds et Libcloud. Elles sont parmi les plus représentatives des solutions d'abstraction de Cloud IaaS et elles sont utilisées dans plusieurs travaux existant qui entame les problèmes d'interopérabilité dans le cloud par exemple, Aeolus [2]et Mosaic[68]. Deltacloud est un framework qui comprend un client Ruby, un tableau de bord Web et un environnement de développement de

drivers pour soutenir l'intégration de nouvelles plates-formes IaaS. jclouds et Libcloud supportent un large éventail de plates-formes, même si elles sont limitées à des services spécifiques, elles sont des librairies de programmation standard, contrairement au Deltacloud, elles n'intègrent pas les outils de développement supplémentaires. En général, les API wrapper accessibles dans les librairies qui peuvent être directement importées par les développeurs dans leurs projets offrent un support plus large en termes de services et plates-formes gérables. La possibilité d'étendre les API proposées afin d'inclure de nouvelles plates-formes et services est une caractéristique commune, et les développeurs sont encouragés à contribuer. C'est également approuvé par la licence open-source qui caractérise ces API. Malgré le bon nombre de plates-formes et services gérés par les différentes API multi-plateformes, la tendance générale est de supporter l'IaaS seulement. Les offres PaaS et SaaS ne sont pas encore envisagés par ces API, même si une série de projets ont essayé d'étendre les librairies existantes. Cela est probablement dû à l'absence d'une norme mature pour les interfaces PaaS, pour lesquelles ces API multi-plateformes pourraient adhérer.

Caractéristiques	**Deltacloud**	**jCloud**	**Libcloud**
La licence	open source Apache licence Version 2.0	open source Apache licence Version 2.0	open source Apache licence Version 2.0
Language de programation	Ruby	java	Python
Clouds supportés	17 clouds	30 cloud	38 cloud
Integration de plate-forme	Drivers	Maven, Dependencies	Drivers
Les standards et Api	OCCI, REST, CIMI, AWS		
Autre interfaces	Interface web, Client Ruby	API	API
Le choix du fournisseur	Manual	Manual	Manual
Les operations supportés	Compute, storage, network	Compute, storage	Compute, storage, network

TABLE 3.1: Tableau comparatifs

3.8 Conclusion

Le Cloud computing est un concept jeune et en constante évolution. Une des évolutions les plus prometteuses d'après la communauté scientifique est le déploiement d'applications distribuées sur un système multi-cloud. En effet, ce mode de déploiement permet, entre autres, d'atténuer le risque de verrouillage propriétaire, de stimuler la concurrence des différents fournisseurs d'infrastructure dans le cloud, et de controler une partie de confidentialité des données utilisées par les applications déployées dans le cloud. Néanmoins, la gestion des ressources informatiques

provenant de plusieurs clouds est un défi technologique et scientifique d'actualité. En effet, la grande taille des systèmes multi-clouds et l'hétérogénéité des ressources qui les composent sont des aspects difficilement pris en compte par les solutions de cloud computing d'aujourd'hui.

Deuxième partie

Contribution

Chapitre 4

Conception

4.1 Introduction

Comme nous avons pu le voir dans la première partie, le cloud computing est considéré comme une alternative aux grilles de calcul pour les applications scientifiques, car le système d'administration du service cloud IaaS fournit une flexibilité pour faciliter et simplifier la gestion des ressources déployées pour les applications gridifiées. Cependant, dans le marché du cloud computing, il n'existe pas un seul cloud homogène qui répond à tous les besoins des utilisateurs cloud mais plusieurs clouds disparates. D'où l'utilisation d'un système multi-cloud est devenue très nécessaire afin de satisfaire les besoins des utilisateurs du cloud computing. Nous avons vu aussi que l'un des défis posés par le système multi-cloud est l'interopérabilité entre ces différents cloud computing parce que chacun d'eux favorise ses propres infrastructure, standards et formats qui sont incompatibles pour accéder à d'autres clouds. D'où différents travaux se penchent sur la question d'interopérabilité dans ce système en proposant un middleware *cloud broker* qui fournit une plateforme interopérable entre plusieurs plateformes clouds.

4.2 Problématique et méthodologie

4.2.1 Problématique

Nous rappelons que dans notre problématique de fédération multi-cloud dans les environnements orientés "grid services", nous nous intéressons de migrer l'exécution des applications d'analyse gridifiées de la grille de calcul vers un système multi-cloud sur des machines virtuelles. tout en essayant de résoudre les problèmes suivants :

1. L'accès à un environnement multi-cloud de manière transparente tout en faisant abstraction de leurs différences de structures.
2. La haute disponibilité des applications gridifiées dans un environnement multi-cloud.
3. Choisir le placement des machines virtuelles dans un environnement multi-cloud et permettre à l'utilisateur de créer et gérer ses machines virtuelles.

4.2.2 Objectifs

Nous nous rappelons que le but de notre étude est d'exécuter des applications scientifiques d'analyse gridifiées dans un environnement multi-cloud afin de diminuer la charge sur la grille de calcul et proposer une plateforme d'interopérabilité pour les services IaaS. A cet effet, nous présentons dans ce chapitre, notre contribution qui consiste à implémenter un cloud Broker *DZ-CBroker* qui permet de gérer un système multi-cloud spécifique au projet *DZ e-Science GRID*, et qui assure les fonctionnalités suivantes :

1. Permettre une seul authentification pour accéder à plusieurs clouds.
2. Permettre à un administrateur grille d'enregistrer les différentes images des machines virtuelles contenants des applications gridifiées (images grilles) dans les différents clouds, afin d'assurer la haute disponibilité des applications scientifiques gridifiées.
3. Permettre aux utilisateurs l'instanciation des images grilles et le déploiement et la gestion des différentes machines virtuelles dans différents clouds d'une façon transparente.
4. Proposer un algorithme de placement des machines virtuelles dans notre environnement multi-cloud.

4.3 Schéma Global proposé

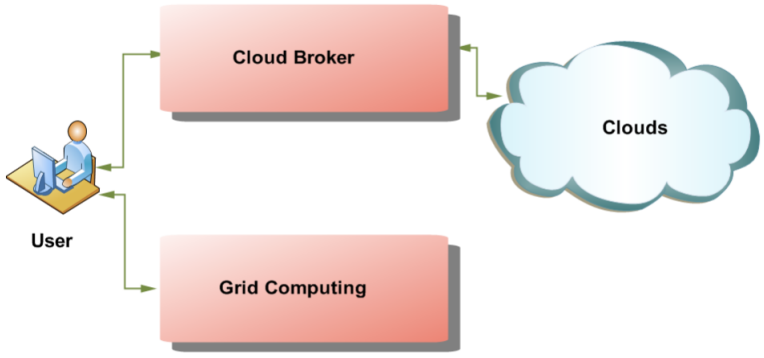

FIGURE 4.1: Architecture globale de la solution

Nous présentons ici une vue globale de l'architecture de la plateforme *DZ-CBroker* illustrée dans la figure 4.1. L'architecture de la plateforme *DZ-CBroker* est basée sur quatre acteurs principaux qui sont : l'utilisateur, la grille de calcul, le cloud broker et un ensemble de plates-formes clouds. Cette architecture est une représentation simplifiée de la plateforme *DZ-CBroker*.

Lorsqu'un utilisateur lance une requête sur la grille, puis récupère les résultats, il aura besoin dans la plupart des cas d'analyser ces résultats, et ce à l'aide d'une ou plusieurs applications gridifiées. Dans notre cas, ces applications sont déployées sur un environnement multi-cloud. Pour cela l'utilisateur va utiliser la plateforme *DZ-CBroker* pour invoquer les applications appropriées.

4.4 Architecture détaillée de la plateforme DZ-CBroker

L'architecture de la plateforme *DZ-CBroker* proposée permet d'assurer l'interopérabilité entre les infrastructures des plateformes cloud. Elle offre à l'utilisateur un accès transparent à plusieurs plateformes clouds simultanément pour l'approvisionnement des machines virtuelles indépendamment des plateformes clouds. L'architecture détaillée de la plateforme *DZ-CBroker* est représentée dans la figure suivante 4.2.

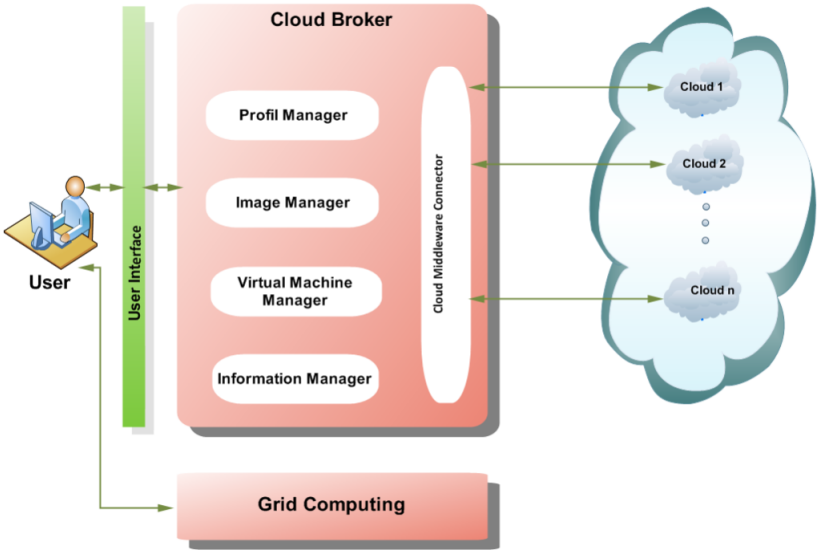

FIGURE 4.2: Architecture détaillée du *DZ-CBroker*

Dans les sections suivantes nous verrons le fonctionnement interne de chaque composant en mettant en évidence l'interaction entre ces différents composants :

4.4.1 User (Utilisateurs)

nous distinguons deux types d'utilisateurs :

1. **Grid Manager** : C'est un administrateur grille qui a tout les privilèges pour gérer la plateforme *DZ-CBroker*, il peut gérer les images grid, les machines virtuelles, ainsi que les comptes utilisateur.
2. **User Grid** : C'est un utilisateur qui veut exécuter des applications gridifiées dans une machine virtuelle dans le cloud. D'où la plateforme *DZ-CBroker* propose à cet utilisateur une liste d'applications gridifiées pour qu'il choisisse l'application qui répond à ses besoins. La plateforme *DZ-CBroker* crée la machine virtuelle dans un environnement multi-cloud et donne la main à l'utilisateur pour manipuler l'état (démarrée, arrêtée, suspendue, supprimée) de la machine virtuelle.

4.4.2 Grille de calcul

Afin d'alléger la gestion des ressources disponibles sur la grille tels que les CPUs, la taille de mémoire, les capacités de stockage, les logiciels accessibles, etc. l'administrateur de la grille de calcul *Grid Manager* crée pour chaque application scientifique gridifiée une Appliance Virtuelle qui va être utilisée pour créer

une image disque de VM (image grille). Dans notre approche, une Appliance Virtuelle est définie comme étant un paquetage logiciel composé d'une image d'un système d'exploitation, et d'un ensemble logiciel qui permet d'avoir un environnement d'exécution fiable correspondant à une application scientifique de la grille de calcul. Ce paquetage n'est pas spécifique à un hyperviseur donné mais supporté par la majorité des hyperviseurs (Xen, Kvm, VMware), (voir figure 4.3).

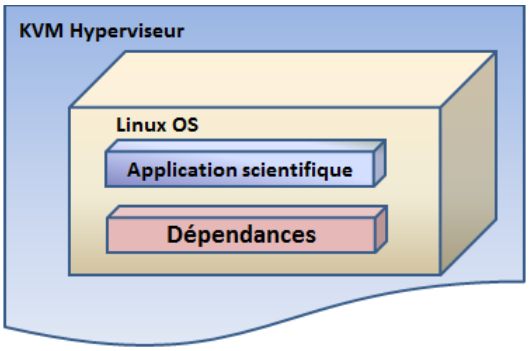

FIGURE 4.3: Exemple d'une Appliance Virtuelle

L'administrateur de la grille de calcul *Grid Manager* sollicite la plateforme *DZ-CBroker* afin de sauvegarder les images grilles créées dans plusieurs plateformes clouds. Par la suite ces images grilles sont instanciées à la demande de l'utilisateur par le service *Virtual Machine Manager* de la plateforme *DZ-CBroker* afin de créer une Machine Virtuelle (VM) spécifique à une application gridifiée.

4.4.3 User Interface

L'interface utilisateur est une application web qui permet aux utilisateurs d'interagir avec la plateforme *DZ-CBroker* pour exécuter leurs requêtes. Elle fournit deux types d'interfaces :

1. **User Grid Interface** : est destinée à l'utilisateur *User Grid*. Elle permet l'authentification, l'approvisionnement des machines virtuelles et la gestion des états des machines virtuelles.
2. **Grid Manager Interface** : est destiné à l'utilisateur *Grid Manager*. Elle permet l'authentification, la création et la gestion des comptes utilisateurs, l'enregistrement des images des machines virtuelles, l'approvisionnement et la gestion des états des machines virtuelles.

4.4.4 La plateforme DZ-CBroker

La plateforme *DZ-CBroker* est un intermédiaire entre l'utilisateur et les plateformes cloud. Elle est composée de cinq services présentés ci-après :

4.4.4.1 Profil Manager

Ce service prend en charge l'authentification des utilisateurs du *DZ-CBroker*. Il permet la création du compte utilisateur et la gestion de son profil. Il a pour rôle aussi l'ouverture de la connexion avec les différentes plateformes cloud.

4.4.4.2 Information Manager

Ce service est chargé de collecter des informations concernant les métriques des ressources disponibles sur toutes les plateformes clouds à l'instant courant, via le service *Cloud Middleware Connector*. Donc pour chaque cloud il va récupérer les informations sur les métriques suivants :

- Le nombre de machines virtuelles allouées.
- Le nombre de CPU disponibles non encore alloués par une machine virtuelle.
- La taille de la mémoire disponible non encore réservé par une machine virtuelle.
- L'espace de stockage disponible non encore alloué par une machine virtuelle.

4.4.4.3 Image Manager

Le Service *Image Manager* a la responsabilité d'enregistrer les images-grille dans les *repositories* des différentes plateformes cloud IaaS ou de les retirer.

Chaque fournisseur IaaS propose un nombre limité de types d'instance (parfois nommées *Flavors*) des machines virtuelles. Chaque type d'instance a un profil de performance spécifique (nombre de CPU, quantité de mémoire, le stockage, type de stockage, etc.) (voir figure 4.4), Où un type d'instance est utilisé dans la définition du modèle de configuration (Template) de création d'une machine virtuelle (voir figure 4.5).

```
<flavor>
<id> 5 <id>
<name> Gold </name>
<vcpu> 4 </vcpu>
<disk> 80 </disk>
<ram> 2048 </ram>
</flavor>
```

FIGURE 4.4: Modèle de configuration de type d'instance (Flavor)

Définition d'une Template : c'est un modèle de configuration disponible qui spécifie les propriétés de configuration comme l'image utilisée pour créer l'instance de la machine virtuelle et du matériel sur lequel cette instance sera exécutée. L'utilisation d'une *Template* rend l'exécution d'une machine virtuelle plus rapide et plus cohérente.

```
<template>
        <id> 1 <id>
        <name> Application Analyse physiques </name>
        <flavor_id> 5 </flavor_id>
        <image_id> 3 </image_id>
</template>
```

FIGURE 4.5: Model de configuration d'une Template

L'exécution d'une application gridifiée nécessite parfois un type d'instance de machine virtuelle spécifique qui n'existe pas dans toutes les plateformes cloud IaaS puisqu'une application gridifiée demande beaucoup de ressources. Dans notre approche nous proposons une solution simple qui consiste à concevoir un modèle qui permettrait de faire abstraction de tout système d'approvisionnement d'une machine virtuelle. Un tel modèle permettrait une description détaillée de la configuration de l'environnement d'exécution afin de l'approvisionner de manière automatisée sur des fournisseurs cloud hétérogènes. Ce modèle que nous avons proposé s'appelle *Manifest*, il est défini sous format xml comme le montre la figure 4.6.

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest xmlns="http://www.dzcbroker.arn.dz" name="TEMPLATE_SPSS">
  - <node name="vm_spss">
        <id_app>1</id_app>
        <app_name>SPSS</app_name>
        <description>SPSS (Statistical Package for the Social Sciences): permet de décrire les caractéristiques d'une
            population donnée, de comparer deux groupes de données ou d'étudier la corrélation entre deux évènements. Ce
            logiciel est très complet et possède toutes les fonctionnalités utiles pour une étude statistique bien posée.
            </description>
      - <infrastructure>
            <cpu>1</cpu>
            <ram>512</ram>
            <disk>300</disk>
        </infrastructure>
        <image_name>SPSS</image_name>
        <user_name>grid_manager</user_name>
    </node>
</manifest>
```

FIGURE 4.6: Modèle de configuration *Manifest*

Dans le tableau 4.1 nous allons présenter les attributs du fichier Manifest. Ces attributs sont définis pour spécifier la capacité d'une machine virtuelle.

Attributs	**Description**
Node_name	Le nom de la machine virtuelle à créer
id_app	L'identifiant de l'application
app_name	Spécifier le nom de l'application gridifiée déployée sur cette machine virtuelle
description	décrire l'application
infrastructure	Cet attribut donne l'environnement de configuration, il spécifie trois sous attributs
sub_attributs :	
CPU	Le nombre de cpu
RAM	La taille de mémoire en Méga octet
DISCK	La taille du disque en Giga octet
image_name	Spécifier le nom de l'image grille utilisée
user_name	Spécifier l'utilisateur qui a le droit de modifier de fichier

TABLE 4.1: Définition des Attributs du fichier *Manifest*

Après la création du fichier *Manifest* il sera sauvegardé dans une base de données pour qu'il soit plus tard utilisé pour l'approvisionnement de la machine virtuelle.

Le service *Image Manager* fournit trois services à savoir, la duplication et le stockage, la configuration et la découverte des images grilles sur plusieurs plateformes cloud ; et ce d'une manière automatique et transparente afin de faciliter le stockage et l'utilisation des images dans la création des machines virtuelles sur différents clouds.

Ces services sont détaillés ci-après :

- **Service duplication et stockage** : offre la haute disponibilité des applications gridifiées où est appliquée la stratégie de réplication des images grilles dans les différentes plateformes clouds. Plus précisément, il utilise la redondance pour s'assurer qu'une défaillance d'une de ses images grilles déployées dans une plateforme cloud n'affecte pas le fonctionnement global du système *DZ-CBroker*. Donc ce service crée un fichier *Manifest* pour chaque image grille selon les besoins de l'application gridifiée en ressources et il le sauvegarde dans la base de données. Puis il enregistre les images grilles dans les *repositories* des différentes plateformes cloud via le service *Cloud Middleware Connector*.
- **Service découverte** : ce service est responsable de récupérer toutes les informations des images grilles stockées dans les différentes plateformes afin de proposer à l'utilisateur une variété d'applications gridifiées déployées sur des images grilles pour créer des machines virtuelles.
- **Service de configuration** : ce service permet à l'utilisateur **Grid Manager** de gérer ces images grilles enregistrées dans les différentes plateformes d'une manière transparente et facile. Il peut ajouter ou supprimer une image grille et donc modifier son environnement d'exécution.

4.4.4.4 Virtual Machine Manager

Il permet l'approvisionnement et la gestion du cycle de vie des machines virtuelles.

- **Service d'approvisionnement** : ce service offre à la plateforme*DZ-CBroker* la possibilité d'exploiter les ressources provenant de plusieurs clouds en fonction des besoins de l'utilisateur. L'efficacité de l'allocation des machines virtuelles dépend de la disponibilité des ressources et l'équilibrage de charge sur les différentes plateformes cloud. D'où le service d'approvisionnement se base sur la comparaison de métriques collectées par le composant *Information Manager* afin de sélectionner le cloud approprié qui va prendre en charge l'allocation d'une machine virtuelle. Pour ce faire nous proposons un algorithme de placement d'une machine virtuelle voir figure 4.7.

```
START
 INPUT: requête;
   Satisfait <- false;
   Pour chaque requête
    Member_list <- null;
    Priority_cloud_list <- null;
    Cloud_list <- All Cloud;
    Filter_list <- (CoreFilter, RamFilter, DiskFilter, ComputeFilter);
    Priority_cloud_list <-(CPU, RAM, Disk, RUN_VM)
          POUR CHAQUE cloud ϵ Cloud_list FAIRE
               SI satisfait_req(cloud, requête,Filter_list) ALORS ajouter( Member_list,cloud);
          FIN;
   FIN;
   SI (member_list not null) ALOR
          Priority_cloud_list <- Trier_Clouds(Member_list, Priority_cloud_list)
          Satistfait <- ture;
          Allouer_VM <- Priority_cloud_list(1);
   SINON fille_attente <- requête ;

   FIN

FIN
```

FIGURE 4.7: Algorithme de placement d'une machine virtuelle

Lorsqu'une requête d'approvisionnement d'une machine virtuelle arrive au service d'approvisionnement elle se déroule comme suit (voir figure 4.8) :

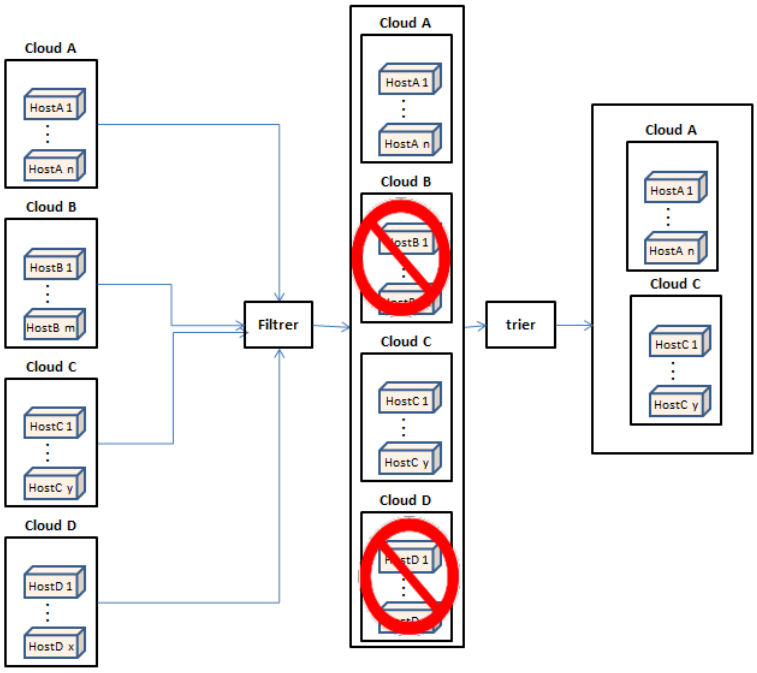

FIGURE 4.8: Processus de filtrage

1. Tout d'abord le service d'approvisionnement demande au service *Information Manger* de lui donner toutes les informations des métriques de toutes les plateformes cloud.

2. Puis il utilise les propriétés de filtrage prédéfinies dans la liste *filter-list* pour éliminer les clouds qui ne satisfont pas les exigences de la requête, et garder ceux qui répondent aux critères dans une liste *member_list*. La liste des filtres *filter-list* est composée de :
   - **cloudFilter** : passe tous les clouds qui sont opérationnels et activés.
   - **CoreFilter** : le filtre se base sur l'utilisation du CPU. Il passe les clouds ayant un nombre suffisant de cœurs de processeurs.
   - **RamFilter** : filtre les clouds par leur RAM. Seuls les clouds avec suffisamment de RAM pour héberger l'instance sont passés.
   - **DisckFilter** : filtre les clouds par leur allocation de disque. Seuls les clouds ayant un espace disque suffisant pour héberger l'instance de la machine virtuelle sont passés.
3. Ensuite il trie les clouds qui satisfont la requête selon leur priorité en donnant un rang pour chaque cloud. Pour calculer le rang de chaque cloud nous comparons leurs capacités en terme de ressources, tout en donnant la priorité au nombre maximum de CPU libre, ensuite à la plus grande taille de RAM puis à l'espace disque libre et dans le cas où ils sont égaux nous mettons en priorité les clouds qui ont moins de machines virtuelles en cours d'exécution afin d'équilibrer la charge sur les clouds.
4. Le service d'approvisionnement choisit le cloud ayant la priorité (le rang) la plus élevée pour créer la machine virtuelle. Dès que le cloud est choisi, le service d'approvisionnement récupère le fichier *Manifest* pour créer le *template* d'approvisionnement approprié à cette plateforme cloud, car dans un environnement hétérogène de multi-cloud, les types de *templates* existants fournis par les différentes infrastructures peuvent avoir des différences. Puis il envoie la requête d'approvisionnement au *Cloud Middleware Connector* qui va déployer l'instance de la machine virtuelle dans la plateforme appropriée.
5. Dans le cas où la liste *member_list* est vide la requête sera enregistrée dans une fille d'attente. Le service d'approvisionnement parcourt périodiquement la file d'attente pour exécuter les requêtes non satisfaites en utilisant la stratégie FIFI [1] (voir figure 4.9).

1. First In First Out

```
START
    TANT QUE (File_attente non null ) FAIRE
            -Récupérer le premier élément (requête) de la file d'attente
            -Exécuter Algorithm1(requête) ;
        SI (satisfait = true)
            supprimer(requête);
        FIN
    FIN
FIN
```

FIGURE 4.9: Algorithme de gestion de la file d'attente

- **Service de surveillance de cycle de vie** : ce service est en charge de gérer le cycle de vie des machines virtuelles, car une machine virtuelle peut passer par différents états (voir figure 4.10). Donc l'utilisateur est capable de démarrer, arrêter, suspendre ou détruire une machine virtuelle.

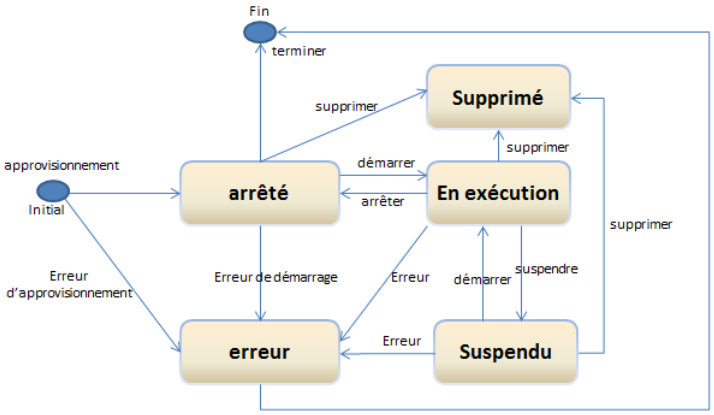

FIGURE 4.10: Cycle de vie d'une machine virtuelle

4.4.4.5 Cloud Middleware Connector

Le déploiement des applications gridifiées dans des environnements multi-cloud évite l'enfermement propriétaire *vendor lock-in* et la dépendance à une seule plateforme cloud. Mais l'environnement multi-cloud cause le problème d'interopérabilité entre les différents cloud et afin de résoudre le problème d'interopérabilité nous avons proposé une couche d'abstraction qui est *Cloud Middleware Connector*. Et ce en fournissant une interface commune et unifiée qui agrège plusieurs interfaces afin de permettre au *DZ-CBroker* de se connecter avec les différentes plateformes cloud d'une manière simultanée et facile. Il fournit des méthodes distinctes pour la recherche d'information des ressources, la gestion des images, l'approvisionnement et la gestion des machines virtuelles ; ces méthodes utilisent des APIs fournies par

chaque plateforme cloud afin de permettre l'interaction avec le cloud approprié. L'utilisation des API fournies par les spécifications des plateformes cloud permet à notre système de rajouter facilement une nouvelle plateforme cloud.

4.4.5 Plates-forme multi-cloud

C'est un ensemble de plateforme cloud IaaS qui sont accessibles par des utilisateurs à travers un intermédiaire qui est la plateforme *DZ-CBroker*. Chaque plateforme cloud dispose d'un ensemble de machines physiques (Hosts) et chaque machine physique contient un hyperviseur pour déployer plusieurs machines virtuelles dans une même machine physique.

4.5 Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation (use case) représente les cas d'utilisations d'un système, les acteurs et les relations entre eux. Comme nous l'avons déjà décrit dans la section 4.4.1, nous avons deux acteurs dans notre système qui sont le **Grid Manager** et le **User Grid**. nous présentons dans la figure 4.11 notre diagramme de cas cas d'utilisation.

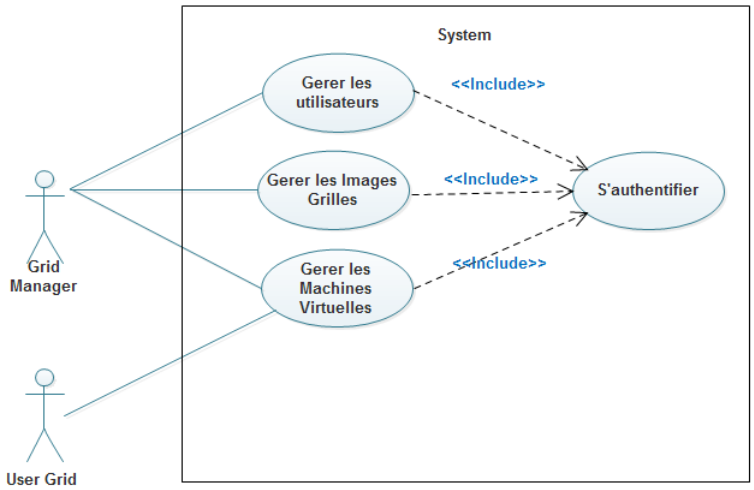

FIGURE 4.11: Diagramme de cas d'utilisation

4.6 Diagrammes de séquence

Un diagramme de séquence est un ensemble d'objets et de relations ainsi que des messages qui peuvent circuler entre eux. Le diagramme de séquence insiste sur

le classement des messages par ordre chronologique.

Dans ce qui suit, nous allons modéliser le fonctionnement de notre système sous forme des diagrammes de séquence. Pour se faire une idée du déroulement des actions, nous allons présenter quatre diagrammes de séquences des principaux cas d'utilisation, ils correspondent respectivement à la phase d'authentification, d'enregistrement des Images grilles, d'approvisionnement d'une machine virtuelle et de la gestion de la file d'attente.

4.6.1 Diagramme de séquence d'authentification

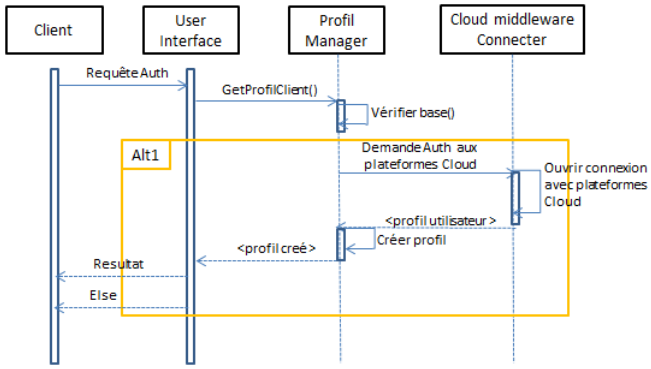

FIGURE 4.12: Diagramme de séquence d'authentification

Lorsque l'utilisateur veut s'authentifier à la plateforme *DZ-CBroker*,

1. L'utilisateur fait une demande d'authentification ;
2. Le service *Profil Manager* vérifie l'authentification et l'autorisation d'accès en utilisant les informations de l'utilisateur disponibles dans la base de données et il transmet la demande au service *Cloud Middleware Connector* afin d'ouvrir une connexion avec les différentes plateformes ;
3. Si l'utilisateur est bien authentifié, le service *Profil Manager* affiche à l'utilisateur son profil détaillé (par exemple les VM déjà crées avec leurs états) ;
4. Sinon le service *Profil Manager* informe l'utilisateur qu'il ne dispose pas d'un compte dans la plateforme *DZ-CBroker*.

4.6.2 Diagramme de séquence de traitement d'une requête d'enregistrement d'une Image-grille

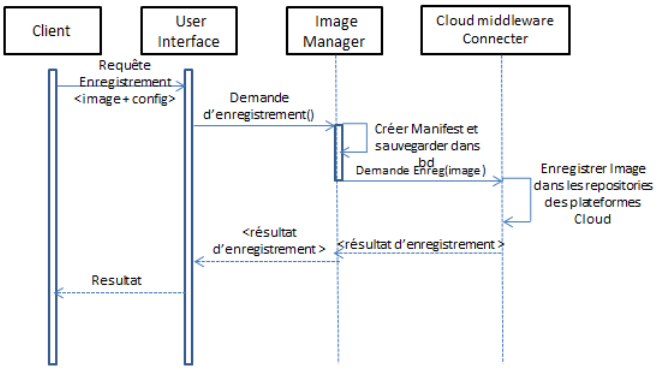

FIGURE 4.13: Diagramme de séquence de traitement d'une requête d'enregistrement d'une Image-grille

Lorsque l'utilisateur *Grid Manager* veut enregistrer une image grille dans l'environnement multi-cloud, il doit utiliser la plateforme *DZ-CBroker* :

1. L'utilisateur *Grid Manager* s'authentifie d'abord dans la plateforme *DZ-CBroker* puis il lance une requête de demande d'enregistrement d'une image et il spécifie l'environnement d'exécution de la VM à travers l'interface *Grid Manager Interface* ;
2. L'interface *Grid Manager Interface* transmet la requête au service *Image Manager*. Ce dernier crée un fichier *Manifest* et le sauvegarde dans la base de données ;
3. Le service *Image Manager* envoie une requête d'enregistrement d'une image au service *Cloud Middleware Connector* pour que ce dernier s'occupe de l'enregistrement de l'image dans les *repositories* des différentes plateformes cloud spécifiés ;
4. Puis le service *Cloud Middleware Connector* transmet le résultat d'enregistrement au service *Image Manager* ;
5. Si l'enregistrement de l'image a été fait avec succès alors le *Grid Manager Interface* affiche au *grid manager* que sa requête a été bien traitée. Sinon il lui affiche un message d'erreur d'enregistrement.

4.6.3 Diagramme de séquence de traitement d'une requête d'approvisionnement d'une VM

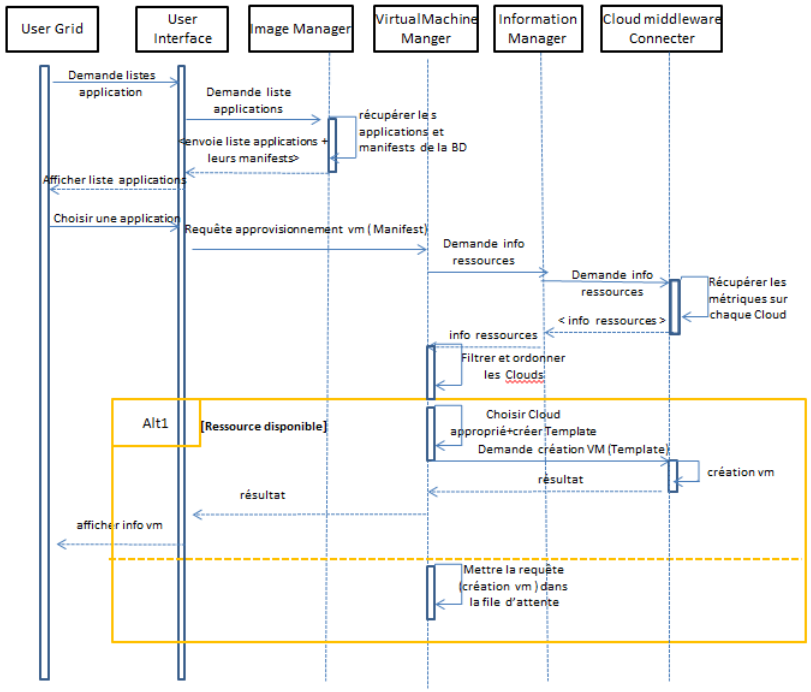

FIGURE 4.14: Diagramme de séquence de traitement d'une requête d'approvisionnement d'une VM

Lorsque l'utilisateur veut exécuter une application scientifique dans un environnement multi-cloud il doit d'abord s'authentifier à la plateforme *DZ-CBroker* puis :

1. L'utilisateur demande la liste des applications via l'interface *User Grid Interface* ;
2. L'interface *User Grid Interface* demande au service *Image Manager* la liste des applications gridifiée existantes ;
3. Le service *Image Manager* envoie à l'interface *User Grid Interface* la listes des applications avec leurs fichiers de configuration *Manifest*. Puis l'interface affiche à l'utilisateur une variété d'applications ;
4. L'utilisateur choisit une application ;
5. L'interface *User Grid Interface* envoie une requête d'approvisionnement d'une VM au service *Virtual Machine Manager*, puis ce dernier demande au service *Information Manager* de lui récupérer les métriques des ressources de chaque plateforme cloud ;

6. Le service *Information Manager* récupère les métrique des ressources de chaque Cloud via le service *Cloud Middleware Connector* ;
7. Le service *Virtual Machine Manger* exécute l'algorithme d'approvisionnement de VM. S'il y a assez de ressources disponibles il sélectionne le meilleur cloud approprié qui répond aux besoins de la requête de l'utilisateur sinon il met la requête dans la file d'attente ;
8. Dans le cas positif, le service *Virtual Machine Manger* crée le *Template* approprié à la plateforme choisie à l'aide du fichier *Manifest*, puis il envoie une requête d'approvisionnement au service *Cloud Middleware Connector* ;
9. Le service *Cloud Middleware Connector* s'occupe de la création de la machine virtuelle dans le cloud approprié ;
10. Si la VM a été bien créée le service *Virtual Machine Manager* affiche à l'utilisateur des informations sur la nouvelle VM créée et lui donne la main de gérer le cycle de vie de cette VM, sinon il met la requête dans la file d'attente.

4.6.4 Diagramme de séquence de gestion d'une file d'attente

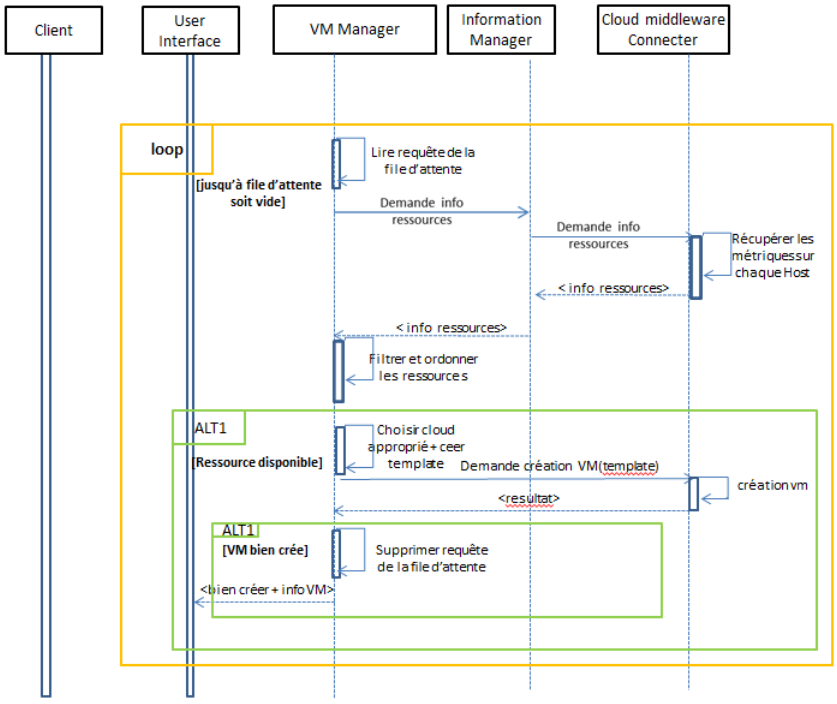

FIGURE 4.15: Diagramme de séquence de gestion d'une file d'attente

Comme nous l'avons déjà mentionné, lorsque la requête d'approvisionnement n'a pas été traitée, le service *Virtual Machine Manager* met la requête dans la fille d'attente.

1. Le service *Virtual Machine Manager* récupère le premier élément (requête) de la file d'attente tant qu'elle n'est pas vide puis il demande au service *Information Manager* de lui récupérer les informations sur les métriques des ressources de chaque plateforme cloud ;
2. Le service *Information manager* récupère une liste de clouds avec leurs informations sur les états des ressources via le service *Cloud Middleware Connector* ;
3. S'il n'y a pas assez de ressources, le service *Virtual Machine Manager* laisse la requête dans la file d'attente sinon il exécute l'algorithme d'approvisionnement de VM. Ensuite il fait une sélection du cloud approprié qui répond mieux aux besoins de la requête ;
4. Après le choix du cloud, le service *Virtual Machine Manager* crée le *Template* approprié à la plateforme choisi et envoie au service *Cloud Middleware Connector* une demande d'approvisionnement d'une VM ;
5. Le service *Cloud Middleware Connector* s'occupe de la création de la machine virtuelle dans le cloud approprié ;
6. Si la VM a été bien créée, le service *Virtual Machine Manager* supprime la requête de la file d'attente et restitue à l'utilisateur des informations sur la nouvelle VM créée et lui donne la main de gérer le cycle de vie de cette VM, sinon il continue de parcourir périodiquement la file d'attente jusqu'à ce qu'elle soit vide.

4.7 Conclusion

Dans ce chapitre nous venons de donner les détails de notre approche. En premier, nous avons défini la problématique ainsi que la méthodologie que nous avons adoptons pour sa résolution. Nous avons abordé tout de suite après, l'architecture globale retenue pour la solution, qui est basée sur quatre acteurs principaux qui sont : l'utilisateur, la grille de calcul, le cloud broker et un ensemble de plates-formes clouds. Ensuite nous avons décrit en détail chaque composant de l'architecture de la plateforme *DZ-CBroker* que nous avons proposé en tant que service multi-cloud qui permet dans un environnement multi-cloud :

1. l'accès uniforme ;
2. la haute disponibilité des images grilles contenant des applications scientifiques gridifiées ;

3. le déploiement, l'exécution et la gestion des machines virtuelles.

Nous avons aussi proposé un algorithme de placement des machines virtuelles et proposer d'utiliser la fille d'attente, en cas de non disponibilité des ressources pour mieux répondre aux besoins des utilisateurs et traiter toutes les requêtes d'approvisionnement des machines virtuelles.

Dans le chapitre suivant, nous allons présenter la validation de notre plateforme *DZ-CBroker* par une implémentation.

Chapitre 5

Mise en œuvre

5.1 Introduction

Ce chapitre est organisé comme suit : dans un premier temps, nous décrivons les environnements logiciels et matériels sur lesquels nous nous appuyons pour la réalisation de nos expérimentations. Ensuite, nous présentons le schéma général de la mise en œuvre. Ce schéma montre les étapes de l'exécution des requêtes des utilisateurs : *Grid Manager* et *User Grid*. Puis nous présentons l'implémentation de notre plateforme *DZ-CBroker*, par la description du fonctionnement de ses composants et leurs interactions, et enfin nous mettons en évidence l'impact de l'algorithme de placement que nous avons proposé par une évaluation de son utilisation.

5.2 Environnements de développement

5.2.1 Le choix des plateformes clouds

Dans la section 2.9 du chapitre 2 nous avons présenté trois clouds open source. Dans notre implémentation (évaluation), nous avons choisi OpenStack et OpenNebula comme plateforme cloud car ils garantissent l'interopérabilité en fournissant des standards ouverts, des APIs partagées afin de se connecter à leurs plateformes facilement.

5.2.2 Les API d'interopérabilité choisis

Pour permettre à la plateforme « DZ-CBroker » de communiquer avec les système de gestion des deux plateformes clouds sélectionnées (OpenNebula et OpenStack) nous avons choisi d'utiliser les APIs : **OpenNebula Cloud API « OCA**

» et **OpenStack4j** respectivement.

1. **OpenNebula Cloud API « OCA »** : est un client open source de OpenNebula qui fournit des fonctionnalités complètes. il est codé en Ruby, Python et Java.
2. **OpenStack4j** : est un client open source de OpenStack qui permet l'approvisionnement et le contrôle du système d'OpenStack.

5.2.3 La plateforme de développement

Nous avons choisis d'utiliser le langage de programmation Java car il est à usage général, évolué et orienté objet dont la syntaxe est proche du C++. Ainsi il est devenu aujourd'hui une direction incontournable dans le monde de la programmation. Aussi, une des principales raisons de ce choix est que les APIs des clouds choisis sont développées avec ce langage. Et comme plateforme de développement notre choix s'est porté sur la plateforme Eclipse. En effet parce que la fondation Eclipse s'est démarquée dans la communauté de l'open source avec un nombre important de sous-projets. Dans notre cas nous nous intéressons au projet *Web Tools Platform (WTP)*. Ce dernier a pour objectif de fournir les outils nécessaires au développement d'applications Web basées sur J2EE. Il se se décline en trois sous-projets :

- Web Standard Tools : cible les outils permettant la manipulation des différents standards rencontrés dans les architectures Web (HTML, CSS, Javascript, SVG, XML, XSD, SOAP, WSDL, UDDI ...)
- J2EE Standard Tools : outillage pour le développement d'applications J2EE. Ces outils permettent les développements aux normes servlets, JSP et EJB. Les applications peuvent être testées et déboguées à partir d'Eclipse en utilisant les fonctionnalités de lancement de serveurs J2EE comme le serveur Apache Tomcat.
- JSF Tools : outillage complémentaire pour le développement d'applications utilisant les JavaServer Faces.

5.2.4 Le serveur web

Nous avons opté pour l'utilisation du serveur Apache Tomcat qui est une implémentation open source d'un conteneur web, il permet donc d'exécuter des applications web reposant sur les technologies Servlets et JSP.

5.3 Schéma général de la mise en œuvre

Pour mettre en œuvre notre approche, nous avons opté pour le modèle *MVC (Model - View - Controller)*, qui est un pattern destiné à la conception d'applications GUI [1]. Son principe de base est la séparation des composants suivants :

1. **Le modèle** : il conserve toutes les données relatives à l'application (sous quelque forme que ce soit : base de données, fichiers. . .) et contient la logique métier de l'application.
2. **La vue** : elle a pour rôle d'offrir une présentation du modèle (IHM [2]). On peut avoir de nombreuses vues pour un même modèle, chacune présentant les informations de manière différente.
3. **Le contrôleur** : c'est le composant qui répond aux actions de l'utilisateur. Il traduit les évènements de l'IHM en modifications du modèle. Il définit également la manière dont l'IHM doit réagir face aux interactions de l'utilisateur.

Donc notre plateforme DZ-CBroker est une application web basée sur le modèle MVC comme le montre la figure 5.1

1. Grafic User Interface
2. Interface Homme Machine

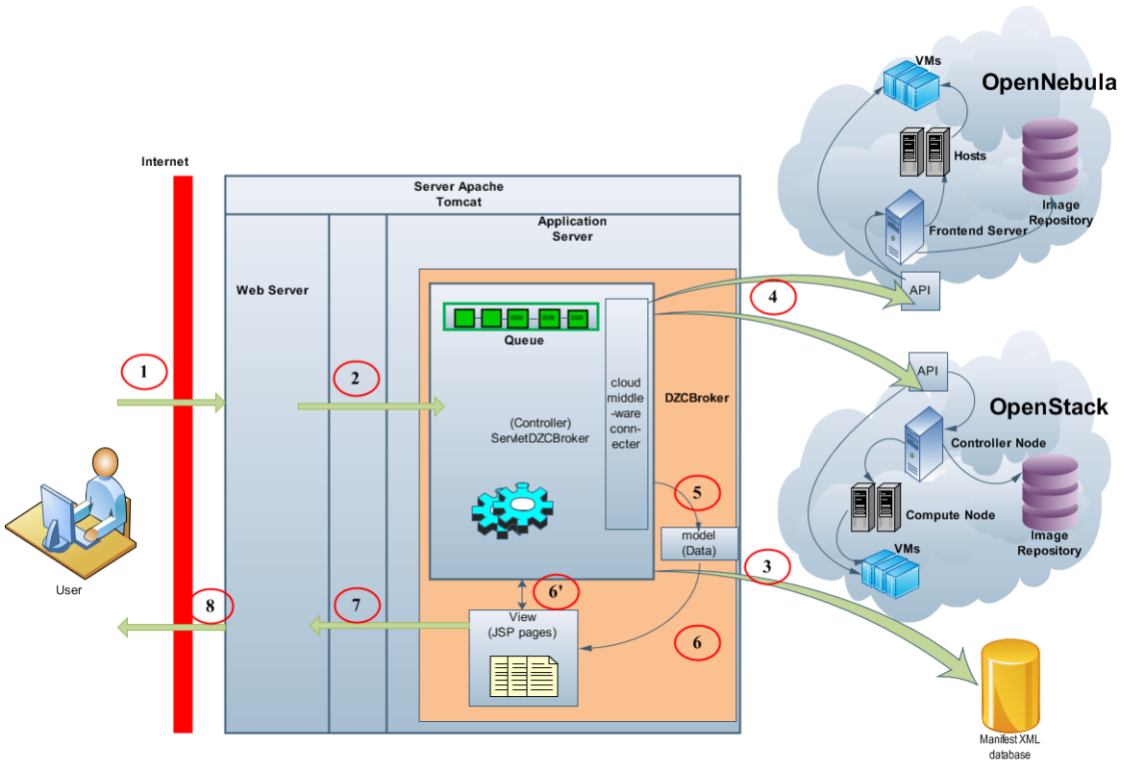

FIGURE 5.1: Schéma général de la mise en œuvre

Lorsque l'utilisateur de la plateforme émet une requête HTTP au serveur web « Apache », ce dernier va la traiter et il s'aperçoit que la requête reçue est destinée au serveur d'applications « Tomcat », donc il la lui transmet -Sachant que les deux serveurs sont reliés par un canal, nommé connecteur [1]-. Le serveur d'applications (Tomcat) reçoit la requête à son tour. Il exécute la servlet « ServletDZCBroker » de l'application DZ-CBroker pour laquelle est destinée la requête. Le serveur « Tomcat » fournit à la servlet deux objets Java exploitables : le premier représente la requête, et le deuxième représente la réponse. Puis, la *servlet* exécute les fonctions correspondantes au traitement de la requête entrante. En fait, ces fonctions se connectent aux plateformes OpenNebula et OpenStack en utilisant les APIs OpenNebula Cloud API « OCA » et OpenStack4j, respectivement. La*servlet* conserve toutes les données relatives à l'application dans le composant model et génère la réponse à l'utilisateur dans un composant *model*. Puis, elle renvoie la réponse (*model*) dans la vue (*page jsp*) correspondante. Ainsi, le serveur d'applications la renvoie, par le biais du connecteur, au serveur web. En fin, le serveur

1. http ://www-igm.univ-mlv.fr/~dr/XPOSE2003/tomcat/tomcat.php ?rub=5&id=3

web affiche la réponse à l'utilisateur sous format HTML.

5.3.1 Diagramme des composants du système proposé et leurs interactions

Nous avons déjà présenté dans le chapitre précédent, dans la section 4.3, les composants de notre plateforme *DZ-CBroker*, qui permettent d'assurer l'interopérabilité entre les infrastructures des plateformes clouds. Dans cette section nous expliquons le fonctionnement de ces composants et leurs interactions, comme le montre la figure 5.2.

L'utilisateur de la plateforme *DZ-CBroker* utilise des vues générées par le Contrôleur, qui représentent le *User Interface*, afin de pouvoir communiquer avec les différents composants de la plateforme. Le composant *User Interface* interagit avec le composant *Profil Manger* pour permettre l'authentification. Il interagit avec le composant *Image Manager* pour permettre l'enregistrement et l'utilisation des images grilles. Et il peut aussi interagir avec le composant *Virtual Machine Manager* afin de permettre la création et la gestion des machines virtuelles. Le composant *Virtual Machine Manager* peut interagir avec le composant *Information Manger* pour récupérer les information relatives aux métriques des ressources disponibles sur toutes les plateformes clouds afin de permettre l'approvisionnement d'une machine virtuelle. Tous les composants cités au-dessus doivent interagir avec le composant *Cloud Middleware Connector* pour permettre l'interaction avec les différentes plateformes clouds d'une manière simultanée et facile.

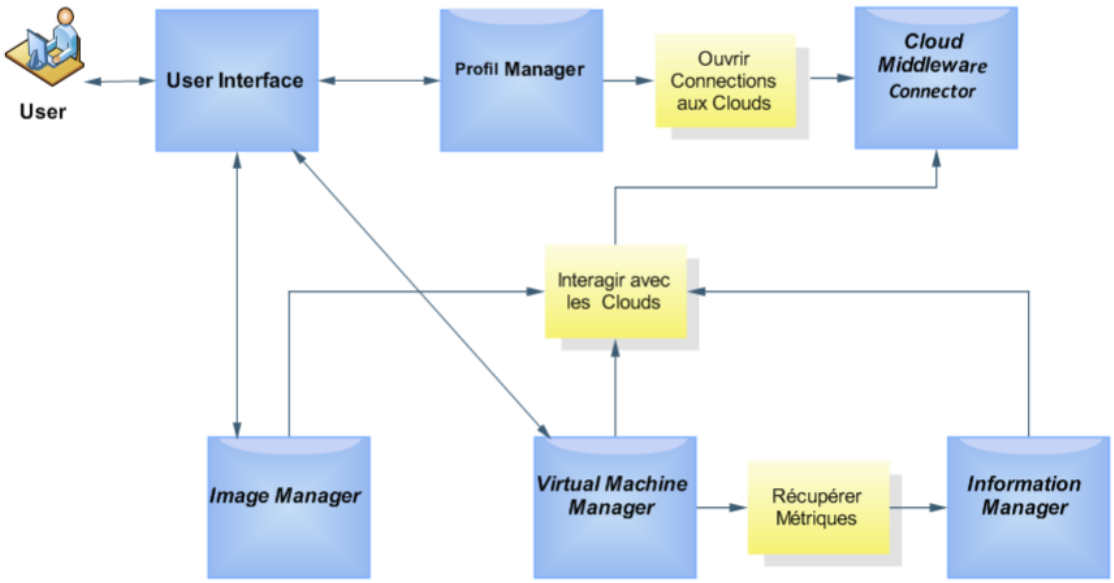

FIGURE 5.2: Diagramme des composants de système proposé et leurs interactions

Dans les sections suivantes, nous allons décrire en détail le fonctionnement interne de chaque composant de l'application *DZ-CBroker* sur le plan implémentation.

5.3.1.1 Profil Manager

Comme nous avons déjà décrit l'architecture du composant *Profil Manger* dans le chapitre précédant dans la section 4.4.4.1, nous allons maintenant détailler son fonctionnement. Afin que l'utilisateur de l'application *DZ-CBroker* puisse utiliser cette plateforme d'interopérabilité, il doit s'authentifier en tant qu'utilisateur *User Grid* ou administrateur *Grid Manager* à travers une interface d'authentification. Une fois qu'il introduit son 'user' et son 'password', le *Profil Manger* exécute la fonction d'authentification qui va récupérer les 'users' et les 'passwords' d'Opennebula et d'Openstack afin d'ouvrir les connexions vers ces deux clouds simultanément à travers le composant *Cloud Middleware Connector*, comme le montre la figure 5.3.

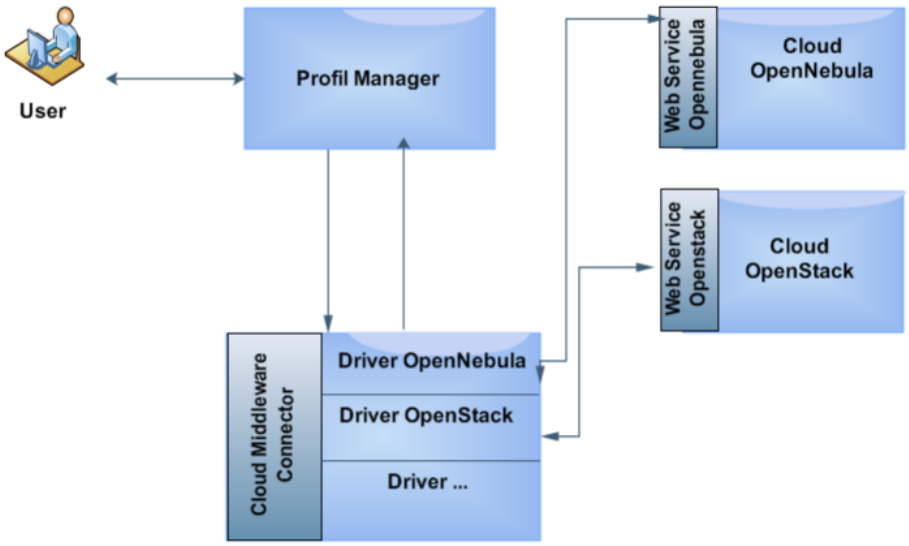

FIGURE 5.3: Fonctionnement de composant *Profil Manager*

Dès que le profil de l'utilisateur est récupéré et que les connexions vers les clouds sont ouvertes, le *Profil Manger* affiche la page d'accueil (vue) selon le type d'utilisateur ; car chaque type d'utilisateur a des permissions et des tâches différentes comme expliqué dans le chapitre précédant au niveau de la section 4.4.3.

5.3.1.2 Image Mananger

Dans la section 4.4.4.3 du chapitre précédent, nous avons expliqué l'architecture du composant *Image Mananger*, nous allons maintenant décrire son fonctionnement qui assure plusieurs tâches :

1. **Ajouter une application gridifiée** : tout d'abord, l'administrateur de la grille de calcul *Grid Manager* crée pour chaque application gridifiée une image grille ; il enregistre ensuite cette image à travers la vue *Ajouter-application.jsp*. Le composant *Image Manager* importe l'image grille et les données relatives à l'application gridifiée de cette image, puis il crée le fichier *Manifest.xml* (voir la figure 5.4) et l'enregistre dans la base de données. Ensuite il enregistre l'image-grille dans les *repositories* des différentes plateformes clouds à travers le composant *Cloud Middleware Connecter* (voir figure5.5).
2. **Gérer les applications gridifiées** : le composant *Image Manger* fournit des fonctions de gestion des applications gridifiées enregistrées dans le système, de manière à permettent à l'utilisateur *Grid Manger* de :

- modifier l'environnement de configuration d'exécution des images grille ;
- activer ou désactiver l'exécution d'une application gridifiée ;
- supprimer une application gridifiée, en supprimant son image grille correspondante dans les différents clouds à travers le composant *Cloud Middleware Connector* ainsi que son fichier manifest.xml qui va être utilisé lors de l'approvisionnement d'une machine virtuelle contenant cette application gridifiée.

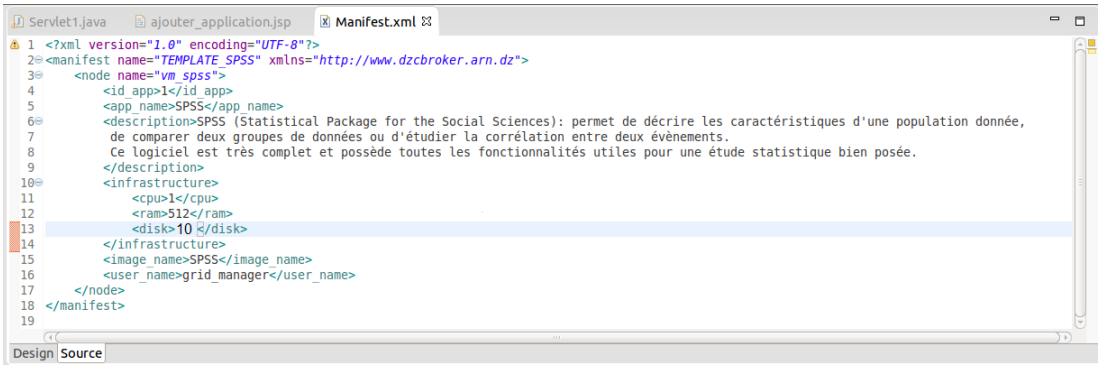

FIGURE 5.4: Modèle d'un fichier Manifest

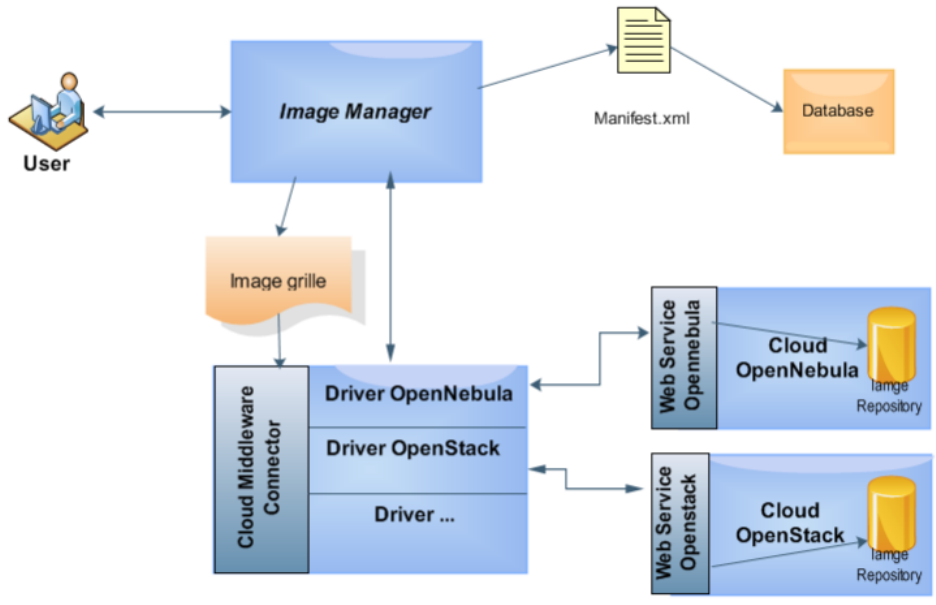

FIGURE 5.5: Fonctionnement de composant *Image Mananger*

5.3.1.3 Information Manager

Le composant *Information Manager* fournit pour chaque cloud participant au système multi-cloud une fonction de récupération des informations relatives aux métriques des ressources disponibles à travers le composant *Cloud Middleware Connector* comme, le montre la figure5.6, ces informations sont :

- Le nombre de machines virtuelles allouées.
- Le nombre de CPU disponibles non encore alloués par une machine virtuelle.
- La taille de la mémoire disponible non encore réservée par une machine virtuelle.
- L'espace de stockage disponible non encore alloué par une machine virtuelle.

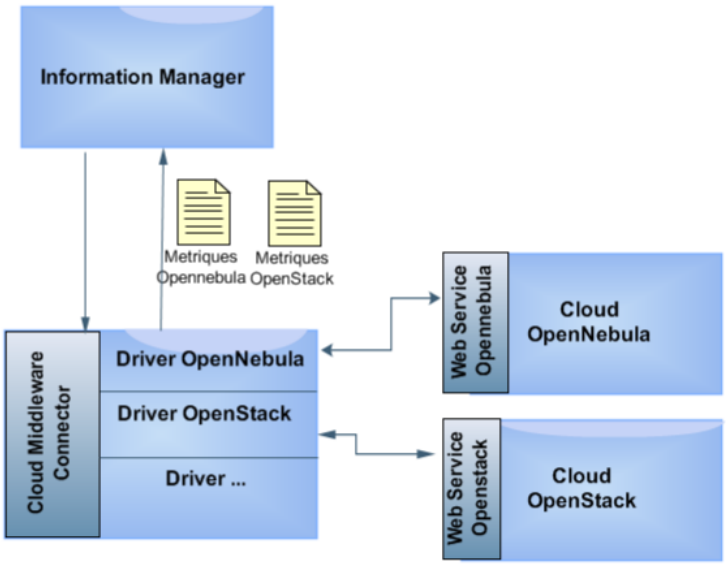

FIGURE 5.6: Fonctionnement de composant *Information Manager*

5.3.1.4 Virtual Machine Manager

Le composant *Virtual Machine Manager*, il a été déjà décrit dans le chapitre précédent dans la section 4.4.4.4, fournit deux services qui sont : le service d'approvisionnement et le service de gestion du cycle de vie des machines virtuelles. Dans la présente section nous décrivons le fonctionnement de chaque service comme le montre la figure 5.7

1. **Service d'approvisionnement** : Lorsque l'utilisateur *User Grid* choisit l'application gridifiée qui répond à ses besoins à travers la *User Interface*

sur la vue *afficher-application.jsp*, cette dernière récupère le fichier *manifest.xml* de l'application gridifiée à travers le composant *Image Manager*, puis transmet la requête de l'utilisateur contenant le fichier *manifest.xml* au composant *Virtual Machine Manager*. Ce dernier demande au composant *Information Manger* de lui donner toutes les informations des métriques des ressources disponibles, cités dans la section précédente 5.3.1.3, concernant les deux plateformes clouds OpenNebula et OpenStack. Il lit ensuite le fichier *manifest.xml* de la requête pour récupérer les besoins en ressources afin de créer la VM correspondante à l'application gridifiée sélectionnée. Ensuite, il exécute l'algorithme de placement d'une machine virtuelle (voir la figure 4.7 dans le chapitre précédent). S'il y a une disponibilité de ressource, le composant *Virtual Machine Manager* crée une *Template* correspondante au cloud approprié afin d'approvisionner la VM de l'application gridifiée dans ce cloud et ce à travers le composant *Cloud Middleware Connector*. Après l'approvisionnement de la VM, le composant *Cloud Middleware Connector* transmet les informations relatives à cette VM au composant *Virtual Machine Manager*. Ce dernier les affiche à l'utilisateur afin de lui permettre d'accéder à cette VM, il pourra ainsi exécuter l'application gridifiée choisie, afin d'analyser les résultats récupérés suite à l'exécution d'une application sur la gille *DZ eScience Grid*. Dans le cas de non disponibilité de ressources, la requête de l'utilisateur est enregistrée dans une file d'attente, le service d'approvisionnement parcourt périodiquement cette file d'attente pour exécuter les requêtes non satisfaites suivant une stratégie FIFO ce qui nous permet d'éviter les cas de famine.

2. **Service de surveillance de cycle de vie** : Ce service permet à l'utilisateur *User Grid* de voir et de gérer toutes ses machines virtuelles à travers la vue *afficher-vms.jsp*. Il lui fournit les fonctions de démarrage, d'arrêt, de suspension et de destruction d'une machine virtuelle. Ce service permet aussi à l'utilisateur *Grid Manager* de voir et de gérer toutes les machines virtuelles créées par tous les utilisateurs dans les deux clouds OpenNebula et OpenStack.

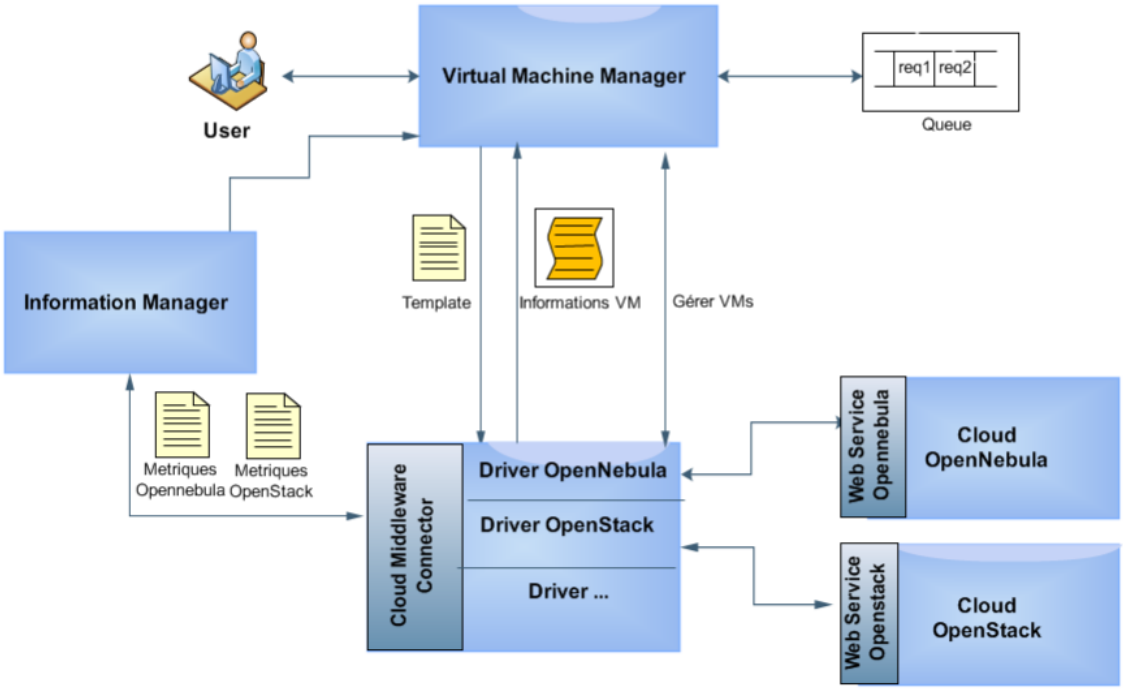

FIGURE 5.7: Fonctionnement de composant *Virtual Machine Manager*

5.3.1.5 Cloud Middleware Connector

Tous les composants de la plateforme *DZ-CBroker* communiquent avec le composant *Cloud Middleware Connector* pour pouvoir accéder aux différentes plateformes clouds d'une manière simultanée et facile. Pour cela, il fournit une interface commune et unifiée qui regroupe plusieurs interfaces, où il crée pour chaque cloud un *Driver* qui utilise l'API de la plateforme correspondante. Par exemple l'API *OpenNebula Cloud API « OCA »* pour la plateforme OpenNebula et l'API *OpenStack4j* pour la plateforme OpenStack. cette approche nous permet d'étendre facilement le système en y ajoutant de nouvelles interface pour d'autres plateformes cloud. Chaque API invoque le service web correspondant à une plateforme cloud donnée, ce service web interagit avec le système de gestion de cette plateforme cloud pour collecter des informations sur les ressources, enregistrer et gérer des images grilles et enfin approvisionner et gérer des machines virtuelles.

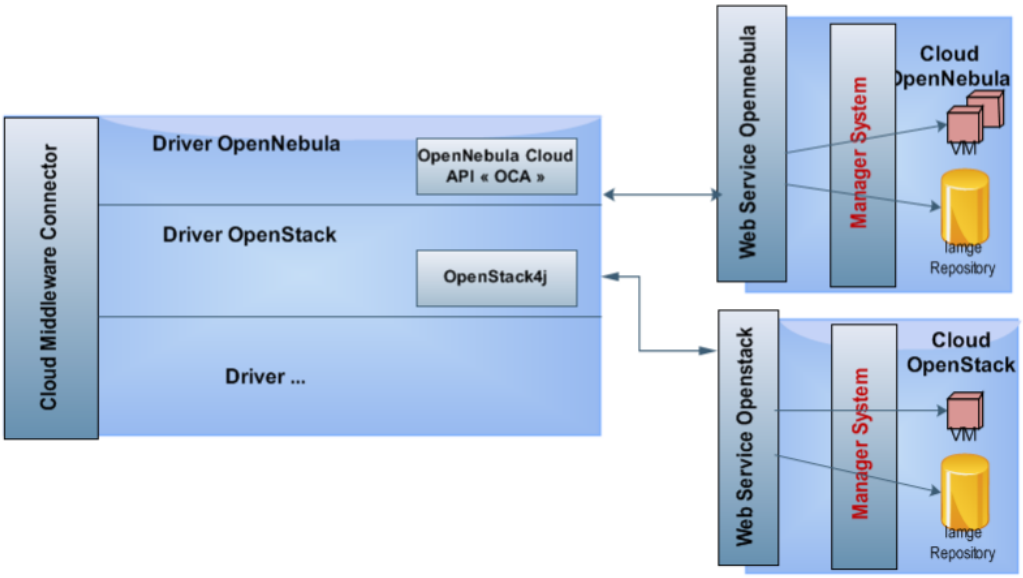

FIGURE 5.8: Fonctionnement de composant *Cloud Middleware Connector*

5.3.2 Expérimentation et évaluation

Dans cette section, nous présentons les résultats de l'expérimentation de notre plateforme d'interopérabilité *DZ-CBroker* que nous avons mis en œuvre. Nous allons dérouler l'algorithme de placement à l'arrivée des requêtes d'approvisionnement de machines virtuelles.

- L'architecture du système multi-cloud sur lequel les tests ont été effectués est composée de deux plateformes clouds :
  - **OpenNebula** : nous avons installé le cloud OpenNebula sur deux machines physiques ; une machine utilisée comme machine frontale pour le système d'administration cloud "Opennebula", et l'autre machines physique constitue le noeud de l'infrastructure du cloud où nous exécutons les machines virtuelles.
  - **OpenStack** : pour la plateforme OpenSatck nous avons utilisé une plateforme de test dédié sur internet qui s'appelle : trystack [1] afin de pouvoir exploiter les ressources offertes. au début nous avons créé un compte sur cette plateforme puis nous avons récupéré les configurations spécifiques pour pouvoir se connecter à cette plateforme en utilisant l'API *Openstack4j* à partir de notre plateforme *DZ-CBroker*.

1. trystack.org

le tableau 5.1 résume les ressources disponibles dans les deux clouds à l'état initial :

Cloud	CPU	RAM	DISK	Nombre d'instance
OpenStack	6	8.192G	60G	0
OpenNebula	2	4.096G	454.1G	0

TABLE 5.1: Les ressources disponibles dans les clouds

- Les applications scientifiques d'analyse gridifiées que nous avons utilisées pour créer leurs *images grilles* sont :

  1. **SPSS** [1] : permet de décrire les caractéristiques d'une population donnée,de comparer deux groupes de données ou d'étudier la corrélation entre deux évènements. ce logiciel est très complet et possède toutes les fonctionnalités utiles pour une étude statique bien posée.
  2. **Scilab** : contient des centaines de fonctions mathématiques,des fonctions graphiques 2D et 3D et un environnement de programmation. Il fournit des outils pour l'analyse et la modification de données.

  Le tableau 5.2 résume les besoins en ressources des deux applications afin de pouvoir s'exécuter sur une machine virtuelle.

Application	CPU	RAM	DISK
SPSS	1	512M	10G
Scilab	2	2048M	15G

TABLE 5.2: Les besoins en ressources des applications gridifiées

Pour démontrer la faisabilité de notre algorithme de placement des machines virtuelles, nous avons exécuté un scénario représentatif sur lequel on peut observer les résultats représentés dans le tableaux 5.9.

1. Statistical Package for the Social Sciences

Requête (déploiement/libération)				Ressources P1 (OpenStack)				Ressources P2 (OpenNebula)				Placement de la VM
	Cpu	Ram	Disk	Cpu	Ram	Disk	Nbr Instance	Cpu	Ram	Disk	Nbr Instance	
R1:déploiement scilab	2	2048M	15G	6	8,192G	60G	0	2	4.096G	454.1G	0	P1
R2 : déploiement SPSS	1	512M	10G	4	6,144G	45G	1	2	4.096G	454.1G	0	P1
R3 : déploiement scilab	2	2048M	15G	3	5,632G	35G	2	2	4.096G	454.1G	0	P1
R4 : déploiement SPSS	1	512M	10G	1	3,584G	20G	3	2	4.096G	454.1G	0	P2
R5 : déploiement SPSS	1	512M	10G	1	3,584G	20G	3	1	3,584G	444.1G	1	P2
R6 : déploiement scilab	2	2048M	15G	1	3,584G	20G	3	0	3,072G	434.1G	2	Mise en file d'attente (R6)
R7 : déploiement SPSS	1	512M	10G	1	3,584G	20G	3	0	3,072G	434.1G	2	Mise en file d'attente (R7)
Libération SPSS (p2)	1	512	10G	1	3,584G	20G	3	1	3,584G	444.1G	1	X
Libération SPSS (p1)	1	512M	10G	2	4.096G	30G	2	1	3,584G	444.1G	1	X
Déploiement scilab de la file d'attente (R6)	2	2048M	15G	2	4.096G	30G	2	1	3,584G	444.1G	1	P1
Déploiement SPSS de la file d'attente (R7)	1	512	10G	0	2,048G	15G	3	1	3,584G	444.1G	1	P2
R8 : déploiement scilab	2	2048M	15G	0	2,048G	15G	3	0	3,072G	434.1G	2	Mise en file d'attente (R8)

FIGURE 5.9: Déroulement de l'algorithme

En fait nous avons deux types de requêtes d'approvisionnement de machines virtuelles ; une pour le déploiement de l'application *Scilab* et l'autre pour le déploiement de l'application *SPSS*. Nous remarquons que lors du déploiement de la requête R1 la machine virtuelle est créée sur la plateforme Openstack(P1) car cette plateforme a plus de CPU par rapport la plateforme OpenNebula(P2). Les figures 5.10 et 5.11 capturées par les interfaces web Horison de OpenStak et Sunstone de OpenNebula montrent respectivement le Taux d'utilisation des ressources après l'exécution de la requête R1.

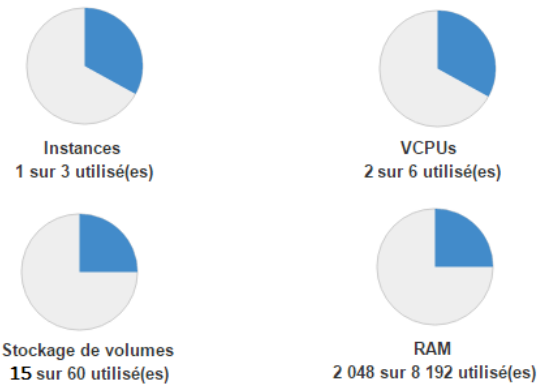

FIGURE 5.10: Taux d'utilisation des ressources de OpenStack après R1

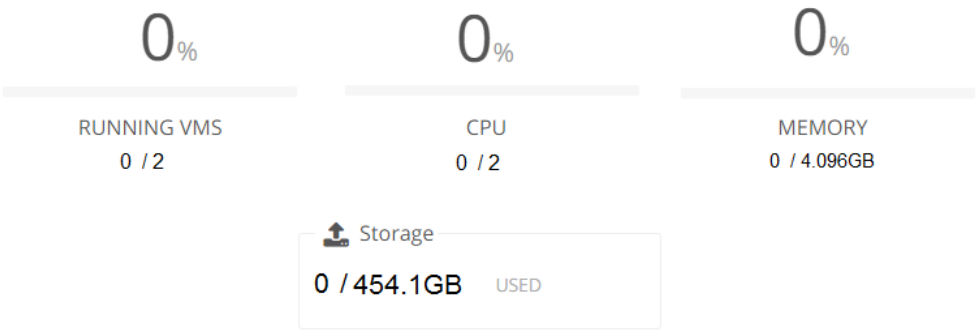

FIGURE 5.11: Taux d'utilisation des ressources de OpenNebula après R1

Nous remarquons que les requêtes R2, R3 et R4 s'exécutent sur le cloud qui a le plus grand nombre de CPU disponibles, par contre R5 s'exécute sur le cloud qui a la plus grande taille disque car il y' a une égalité sur le nombre de CPU et la taille de la RAM disponible sur les deux clouds. Les figures 5.12 et 5.13 montrent le taux d'utilisation des ressources après l'exécution de la requête R5.

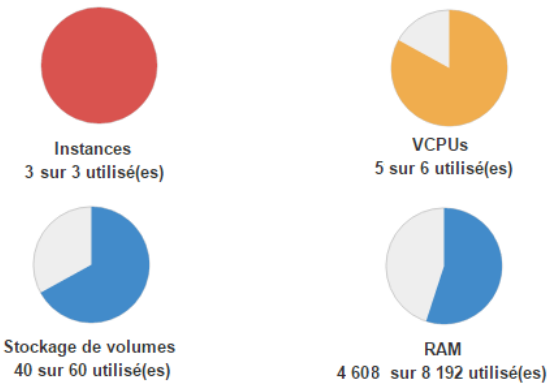

FIGURE 5.12: Taux d'utilisation des ressources de OpenStack après R5

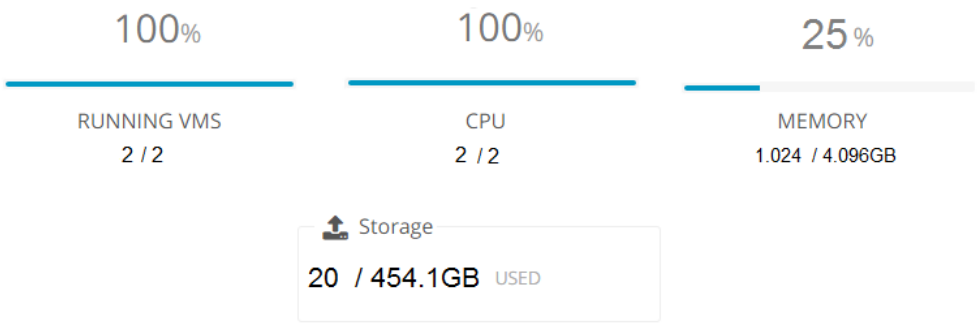

FIGURE 5.13: Taux d'utilisation des ressources de OpenNebula après R5

La requête R6 est mise dans la file d'attente parce qu'il n'y a pas assez de ressources. A l'arrivage de la requête R7 le système la met dans la file d'attente car le système utilise la stratégie FIFO [1] pour l'exécution des requêtes.

Nous observons que dès que les deux clouds libèrent quelque ressources (suppression des deux machines SPSS de P2 et SPSS de P1), le système déploie les deux requêtes mises en attente R6 puis R7 successivement. A la fin, la requête R8 est mise en attente pour une éventuelle libération de ressources.

5.4 Conclusion

Nous avons présenté dans ce chapitre l'implémentation de notre plateforme *DZ-CBroker* qui permet l'interopérabilité entre les deux plateformes OpenNebula

1. First In FIRST Oout

et OpenStack. Comme nous l'avons montré, notre solution offre à l'utilisateur un accès transparent, facile et convivial pour la manipulation de ses machines virtuelles dans un environnement multi-cloud. De plus, nous avons constaté que notre algorithme de placement des machines virtuelles est satisfaisant car il permet d'exécuter toutes les requêtes tout en évitant la famine.

Conclusion et Perspectives

Conclusion générale et Perspectives

La technologie de grille de calcul est adoptée dans un grand nombre de domaines scientifiques, notamment, la physique, l'astronomie, le génie biomédical, l'observation de la terre et et la climatologie, la nanotechnologie, le domaine géospatial, et bien d'autres. Mais l'un des problèmes actuellement rencontré par la grille de calcul est le problème de gestion des ressources dans l'infrastructure.

Récemment, le concept de cloud computing a émergé et se propage très largement dans presque tous les domaines des technologies de l'information, et cela grâce à la virtualisation. Ce nouveau paradigme est considéré comme une alternative aux grilles de calcul pour les applications scientifiques, parce que le système d'administration cloud IaaS fournit une flexibilité pour faciliter et simplifier la gestion des ressources déployées pour les applications grille. Comme le cloud computing est un concept jeune il est en constante évolution. Une des évolutions les plus prometteuses d'après la communauté scientifique est le déploiement d'applications distribuées sur un système multi-cloud. En effet, ce mode de déploiement permet, entre autres, d'atténuer le risque de verrouillage propriétaire et de stimuler la concurrence des différents fournisseurs d'infrastructure dans le cloud. Néanmoins, le problème d'interopérabilité entre ces différentes infrastructures clouds est un défi technologique et scientifique d'actualité. En effet, la grande taille des systèmes multi-cloud et l'hétérogénéité des ressources qui les composent sont des aspects difficilement pris en compte par les solutions de cloud computing d'aujourd'hui.

La contribution majeure de notre travail est la mise en place d'une plateforme d'interopérabilité au niveau IaaS spécifique à notre grille nationale « DZ eScience Grid ». Cette plateforme offrira aux utilisateurs de notre grille nationale « DZ eScience Grid », plus de souplesse et de flexibilité pour exécuter les applications d'analyses scientifiques.

Pour mettre en place notre solution, nous avons fait une étude détaillée sur les technologies de la grille de calcul et le cloud computing, ainsi que sur les systèmes multi-cloud. Nous avons passé en revue les travaux de recherche sur les solutions proposées pour résoudre le problème l'interopérabilité dans les systèmes

multi-cloud. Enfin nous avons proposé un cloud broker qui est une plateforme d'interopérabilité au niveau IaaS « DZ-CBroker ». Cette plateforme permet l'accès unifier à plusieurs clouds hétérogènes « système multi-cloud » afin de gérer les machines virtuelles contenants des applications gridifiées. Pour la mise en œuvre et le déploiement de la plateforme d'interopérabilité « DZ-CBroker », nous avons implémenté les différents composants de ce dernier selon une architecture modulaire. Pour tester notre plateforme, nous avons effectué une étude des différents cloud IaaS open source pouvant être exploités dans un système multi-cloud. Après, nous avons choisi les clouds IaaS « OpenNebula » et « OpenStack ». Nous avons pu montrer à travers l'implémentation et l'expérimentation de la plateforme *DZ-CBroker* sur un environnement « multi-cloud » que nous avons atteint les objectifs que nous avons fixés :

- L'accès uniforme à un environnement multi-cloud d'une manière transparente tout en faisant abstraction de leurs différences de structures.
- Permettre à un administrateur grille d'enregistrer les différentes images grilles dans les différents clouds, afin d'assurer la haute disponibilité des applications scientifiques gridifiées.
- Permettre aux utilisateurs de déployer et gérer les différentes machines virtuelles dans différents clouds d'une façon transparente.
- Proposer un algorithme de placement des machines virtuelles dans notre environnement multi-cloud.

Perspectives

La continuité de ce travail portera principalement sur les points suivants :

- Etendre le système multi-cloud tout en intégrant de nouvelles plates-formes cloud, comme par exemple, *Eucalyptus* dans la plateforme *DZ-CBroker* ;
- Permettre l'exécution distribuée d'une application gridifiée sur plusieurs machines virtuelles déployées dans différents clouds ;
- Améliorer l'algorithme de placement des machines virtuelles, tout en traitant le temps d'exécution et l'économie de l'énergie.

Bibliographie

[1] Ahmar Abbas. *Grid Computing : A Practical Guide to Tech. and App.* Firewall Media, 2004.

[2] Aeolus. Aeolus : Manage Your Cloud Deployments with Ease. https ://github.com/aeolusproject/aeolusproject.github.com/wiki, 2015.

[3] Arjuna Agility. What Is Federation, 2014.

[4] Roberto Alfieri, Roberto Barbera, Patrizia Belluomo, Alessandro Cavalli, Roberto Cecchini, Andrea Chierici, Vincenzo Ciaschini, Luca Dell'Agnello, Flavia Donno, Enrico Ferro, et al. The infn-grid testbed. *Future Generation Computer Systems*, 21(2) :249–258, 2005.

[5] AWS Amazon. Summary of the amazon ec2 and amazon rds service disruption in the us east region, 2015.

[6] Brian Amedro, Françoise Baude, Fabrice Huet, and Elton Mathias. Combining grid and cloud resources by use of middleware for spmd applications. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 177–184. IEEE, 2010.

[7] Paolo Anedda, Simone Leo, Simone Manca, Massimo Gaggero, and Gianluigi Zanetti. Suspending, migrating and resuming hpc virtual clusters. *Future Generation Computer Systems*, 26(8) :1063–1072, 2010.

[8] Apache. The Apache Software Foundation. http ://www.apache.org/foundation, 2015.

[9] Maurice. Audin. Etat de l'art du cloud computing et adaptation au logiciel libre. *111*, 2009.

[10] AWS. AWS : Amazon S3. https ://aws.amazon.com/s3, 2015.

[11] Kapil Bakshi. Cisco cloud computing-data center strategy, architecture, and solutions point of view white paper for us public sector 1st edition, 2009.

[12] Steve Bennett, Mans Bhuller, and Robert Covington. Architectural strategies for cloud computing. *Oracle White Paper in Enterprise Architecture*, 2009.

[13] BioGrid. Construction of a supercomputer network, 2015.

[14] Rajkumar Buyya. Grid computing info centre (grid infoware). *Aquired at : http ://www.gridcomputing.com/9pages*, 2005.

[15] Rajkumar Buyya, Rajiv Ranjan, and Rodrigo N Calheiros. Intercloud : Utility-oriented federation of cloud computing environments for scaling of application services. In *Algorithms and architectures for parallel processing*, pages 13–31. Springer, 2010.

[16] Rajkumar Buyya and Srikumar Venugopal. A gentle introduction to grid computing and technologies. *database*, 2 :R3, 2005.

[17] Emanuele Carlini, Massimo Coppola, Patrizio Dazzi, Laura Ricci, and Giacomo Righetti. Cloud federations in contrail. In *Euro-Par 2011 : Parallel Processing Workshops*, pages 159–168. Springer, 2012.

[18] Use Cases. Functional requirements for inter-cloud computing. In *Global Inter-Cloud Technology Forum, GICTF White Paper*, 2010.

[19] Jeffrey S Chase, Darrell C Anderson, Prachi N Thakar, Amin M Vahdat, and Ronald P Doyle. Managing energy and server resources in hosting centers. In *ACM SIGOPS Operating Systems Review*, volume 35, pages 103–116. ACM, 2001.

[20] CNRS. Grille de calcul : l'internet du calcul intensif, 2014.

[21] R Cohen. Examining cloud compatibility, portability and interoperability. *ElasticVapor : Life in the Cloud, DOI : http ://www. elasticvapor. com/2009/02/examining-cloudcompatibility. html*, 2009.

[22] Condor. High thorughput computing, 2014.

[23] Nicolas Cynober. Différence entre Grid et Cloud Computing. http ://www.haute-disponibilite.net/2008/07/15/difference-entre-grid-et-cloud-computing, 2008.

[24] Grzegorz Czajkowski, Michal Wegiel, Laurent Daynes, Krzysztof Palacz, Mick Jordan, Glenn Skinner, and Ciaran Bryce. Resource management for clusters of virtual machines. In *Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on*, volume 1, pages 382–389. IEEE, 2005.

[25] D-Grid. D-Grid initiative, 2015.

[26] DataGrid. The DataGrid project, 2015.

[27] D Davis and G Pilz. Cloud infrastructure management interface (cimi) model and rest interface over http. *vol. DSP-0263, May*, 2012.

[28] Deltacloud. Deltacloud Framework. http ://deltacloud.apache.org, 2014.

[29] Yuhui Deng and Frank Wang. A heterogeneous storage grid enabled by grid service. *ACM SIGOPS Operating Systems Review*, 41(1) :7–13, 2007.

[30] DMTF. DMTF to develop standards for managing a cloud computing environment. http ://www.dmtf.org/standards/cloud/, Juin 2011.

[31] DutchGrid. Large-scale distributed computing in the Netherlands, 2015.

[32] EC2. Amazon : Amazon Web Services : Amazon EC2. http ://aws.amazon.com/ec2, 2015.

[33] Andy Edmonds, Thijs Metsch, Alexander Papaspyrou, and Alexis Richardson. Toward an open cloud standard. *Internet Computing, IEEE*, 16(4) :15–25, 2012.

[34] EMI. European Middleware Initiative, 2015.

[35] Kalalo Etienne, Lafargue Flore, Martins Diana, Pagnon Stéphane, and Pons Jérôme. le cloud computing une nouvelle filière fortement structurante. *ÎLE-DE-FRANCE Direction régionale des entreprises, de la concurrence, de la consommation du travail et de l'emploi*, 2012.

[36] Eucalyptus. Eucalyptus. http ://open.eucalyptus.com, 2015.

[37] Ana Juan Ferrer, Francisco HernáNdez, Johan Tordsson, Erik Elmroth, Ahmed Ali-Eldin, Csilla Zsigri, RaüL Sirvent, Jordi Guitart, Rosa M Badia, Karim Djemame, et al. Optimis : A holistic approach to cloud service provisioning. *Future Generation Computer Systems*, 28(1) :66–77, 2012.

[38] Ian Foster. What is the grid ? a three point checklist, july 2002. *ThreePointCheck. pdf*, 2006.

[39] Ian Foster, Timothy Freeman, Kate Keahy, Doug Scheftner, Borja Sotomayer, and Xuehai Zhang. Virtual clusters for grid communities. In *Cluster Computing and the Grid, 2006. CCGRID 06. Sixth IEEE International Symposium on*, volume 1, pages 513–520. IEEE, 2006.

[40] Armando Fox, Rean Griffith, Anthony Joseph, Randy Katz, Andrew Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, and Ion Stoica. Above the clouds : A berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, 28 :13, 2009.

[41] gLite. Overview of gLite grid middleware, 2014.

[42] gLite3.0. Glite 3.2 user guide. *CERN. https ://edms.cern.ch/file/722398/1.4/gLite-3-UserGuide.pdf*, 2010.

[43] Globus. Globus Alliance, 2013.

[44] Grid5000. Grid5000. https ://www.grid5000.fr/, 2014.

[45] GridPP. Grid for Particle Physics. UK computing for particle physics, 2015.

[46] Nikolay Grozev and Rajkumar Buyya. EI-Cloud. https ://www.linkedin.com/company/ei-cloud, 2014.

[47] Nikolay Grozev and Rajkumar Buyya. Inter-cloud architectures and application brokering : taxonomy and survey. *Software : Practice and Experience*, 44(3) :369–390, 2014.

[48] Piyush Harsh, Florian Dudouet, Roberto G Cascella, Yvon Jegou, and Christine Morin. Using open standards for interoperability issues, solutions, and challenges facing cloud computing. In *Network and service management (cnsm), 2012 8th international conference and 2012 workshop on systems virtualiztion management (svm)*, pages 435–440. IEEE, 2012.

[49] jClouds. The Java Multi-Cloud Toolkit. http ://jclouds.apache.org, 2015.

[50] Hai Jin. Chinagrid overview, 2004.

[51] Joshy Joseph and Craig Fellenstein. *Grid computing*. Prentice Hall Professional, 2004.

[52] Jean-Philppe Kalfon. Les nouveaux modèles économiques des courtiers du cloud. *http ://bfmbusiness.bfmtv.com/01-business-forum/les-nouveaux-modeles-economiques-des-courtiers-du-cloud-582993.html*, 2012.

[53] Erwin Laure and Bob Jones. Enabling grids for e-science : The egee project. *Grid computing : infrastructure, service, and applications*, page 55, 2009.

[54] Chad M Lawler. Cloud service broker. *Hitachi : Green IT Cloud Summit*, 2012.

[55] LCG. Worldwide lhc computing grid, 2015.

[56] Laurent Lefevre, Olivier Mornard, Jean-Patrick Gelas, and Maxime Morel. Monitoring energy consumption in clouds : the compatibleone experience. In *Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on*, pages 794–795. IEEE, 2011.

[57] Legion. A worldwild virtual computer, 2014.

[58] Apache Libcloud. One Interface To Rule Them All. http ://j-clouds.apache.org, 2015.

[59] Fang Liu, Jin Tong, Jian Mao, Robert Bohn, John Messina, Lee Badger, and Dawn Leaf. Nist cloud computing reference architecture. *NIST special publication*, 500 :292, 2011.

[60] Charles Loomis, Mohammed Airaj, Marc-Elian Bégin, Evangelos Floros, Stuart Kenny, and David O'Callaghan. Stratuslab cloud distribution. *European Research Activities in Cloud Computing*, page 271, 2012.

[61] V Louve. Grille de calcul. http ://www.mathrice.org/rencontres/octobre.2004/grilles.pdf, 2004.

[62] Slaheddine. Maaref. Cloud computing en afrique situation et perspectives. *222*, 2012.

[63] Frédéric Magoulès, Jie Pan, Kiat-An Tan, and Abhinit Kumar. *Introduction to grid computing*. CRC Press, 2009.

[64] MaGrid. La Grille de Calcul Marocaine. 2015.

[65] Peter Mell and Tim Grance. The nist definition of cloud computing. 2011.

[66] Microsoft. Windows Azure Service Disruption Update. http ://blogs.msdn.com/b/windowsazure/archive/2012/03/01/windows-azure-service-disruption-update.aspx, 2013.

[67] Ralph Mietzner and Frank Leymann. Towards provisioning the cloud : On the usage of multi-granularity flows and services to realize a unified provisioning infrastructure for saas applications. In *Services-Part I, 2008. IEEE Congress on*, pages 3–10. IEEE, 2008.

[68] mOSAIC. mOSAIC : Open source API and platform for multiple clouds . *http ://www.mosaic-cloud.eu*, 2015.

[69] Michael A Murphy and Sebastien Goasguen. Virtual organization clusters : Self-provisioned clouds on the grid. *Future Generation Computer Systems*, 26(8) :1271–1281, 2010.

[70] Landry Fossouo Noumsi. Etude et mise en place d'une solution "cloud computing " privée dans une entreprise moderne : cas de camtel. *Ecole nationale supérieure des postes et télécommunications - Ingénieur des travaux des télécommunications*, 2012.

[71] OCCI. OpenStack : OCCI. `https://wiki.openstack.org/wiki/Occi#Summary`, 2015.

[72] OGF. Open Grid Forum : Open Forum - Open Standards. https ://www.ogf.org/dokuwiki/doku.php, 2015.

[73] OpenNebula. OpenNebula. http ://opennebula.org/, 2015.

[74] OpenStack. OpenStack. http ://www.openstack.org/, 2015.

[75] OPTIMIS. OPTIMIS Toolkit components. http ://optimistoolkit.com/features.html, 2015.

[76] OSG. Open Science Grid, 2015.

[77] Przemyslaw Pawluk, Bradley Simmons, Michael Smit, Marin Litoiu, and Serge Mankovski. Introducing stratos : A cloud broker service. In *2012 IEEE Fifth International Conference on Cloud Computing*, pages 891–898. IEEE, 2012.

[78] Dana Petcu. Multi-cloud : expectations and current approaches. In *Proceedings of the 2013 international workshop on Multi-cloud applications and federated clouds*, pages 1–6. ACM, 2013.

[79] Dana Petcu, Ciprian Crăciun, Marian Neagul, Silviu Panica, Beniamino Di Martino, Salvatore Venticinque, Massimiliano Rak, and Rocco Aversa. Architecturing a sky computing platform. In *Towards a Service-Based Internet. ServiceWave 2010 Workshops*, pages 1–13. Springer, 2011.

[80] Thierry Priol. Défis et perspectives scientifiques des grilles informatiques. *Club des déci*, 2005.

[81] Han Qi and Abdullah Gani. Research on mobile cloud computing : Review, trend and perspectives. In *Digital Information and Communication Technology and it's Applications (DICTAP), 2012 Second International Conference on*, pages 195–202. ieee, 2012.

[82] M. Quinson and F. Suter. Grilles informatiques et algorithmique distribuée avancée. *www.loria.fr/ suter/files/SDR08-09.pdf*, 2008-2009.

[83] Thomas Rings, Geoff Caryer, Julian Gallop, Jens Grabowski, Tatiana Kovacikova, Stephan Schulz, and Ian Stokes-Rees. Grid and cloud computing : opportunities for integration with the next generation network. *Journal of Grid Computing*, 7(3) :375–393, 2009.

[84] Benny Rochwerger, David Breitgand, Eliezer Levy, Alex Galis, Kenneth Nagin, Ignacio M Llorente, Ruben Montero, Yaron Wolfsthal, Erik Elmroth, Juan Caceres, et al. The reservoir model and architecture for open federated cloud computing. *IBM Journal of Research and Development*, 53(4) :4–1, 2009.

[85] Benny Rochwerger, David Breitgand, Eliezer Levy, Alex Galis, Kenneth Nagin, Ignacio M Llorente, Ruben Montero, Yaron Wolfsthal, Erik Elmroth, Juan Caceres, et al. The reservoir model and architecture for open federated cloud computing. *IBM Journal of Research and Development*, 53(4) :4–1, 2009.

[86] Luis Rodero-Merino, Eddy Caron, Adrian Muresan, and Frédéric Desprez. Using clouds to scale grid resources : An economic model. *Future Generation Computer Systems*, 28(4) :633–646, 2012.

[87] Davide Salomoni, Elisabetta Ronchieri, Gianni Dalla Torre, and Vincenzo Ciaschini. WNoDeS "Grid / Cloud Integration Framework" . http ://wnodes.github.io/WNoDeS/documentation/architectural_framework.html, 2014.

[88] L Schubert and K Jeffery. Advances in clouds research in future cloud computing, report from the cloud computing expert working group meeting. *Cordis (Online), BE : European Commission*, 2012.

[89] Lutz Schubert, Keith G Jeffery, and Burkard Neidecker-Lutz. *The Future of Cloud Computing : Opportunities for European Cloud Computing Beyond 2010 :–expert Group Report.* European Commission, Information Society and Media, 2010.

[90] SDG. Scientific Data Grid CA, 2015.

[91] Jose Luis Lucas Simarro, Rafael Moreno-Vozmediano, Ruben S Montero, and Ignacio Martín Llorente. Dynamic placement of virtual machines for

cost optimization in multi-cloud environments. In *High Performance Computing and Simulation (HPCS), 2011 International Conference on*, pages 1–7. IEEE, 2011.

[92] D Slik, M Siefer, E Hibbard, C Schwarzer, A Yoder, LN Bairavasundaram, S Baker, M Carlson, H Nguyen, and R Ramos. Cloud data management interface (cdmi) v1. 0, 2011.

[93] Stratos. Apache Stratos. https ://c-wiki.apache.org/confluence/display/STRATOS/4.1.0+Architecture, 2015.

[94] C. Therry. Le projet européen de grille de calcul (DataGrid) : Concept de grilles de calcul. Objectifs du projet et état d'avancement. *Conservatoire National des Arts et Métiers, Centre d'enseignement de Lyon*, Mai 2003.

[95] Unicore. Distributed computing and data resources, 2014.

[96] Valerio Venturi. Ccr giornata di formazione dedicata al cloud computing. Feb 2013.

[97] Massimo Villari. *Achieving Federated and Self-Manageable Cloud Infrastructures : Theory and Practice : Theory and Practice*. IGI Global, 2012.

[98] Bimlesh Wadhwa, Aditi Jaitly, and Bharti Suri. Cloud service brokers : an emerging trend in cloud adoption and migration. In *Software Engineering Conference (APSEC, 2013 20th Asia-Pacific*, volume 2, pages 140–145. IEEE, 2013.

[99] Rebecca Wetzel. Cloud federation primer : The coming intercloud. *Internet : http ://searchtelecom.techtarget.com/feature/Cloud-federation-primer-The-coming-Intercloud*, Mar. 2015.

[100] Vic (J.R.) Winkler. La sécurité dans le cloud. *Pearson Education France.*, 2011.

[101] Vic JR Winkler. *Securing the Cloud : Cloud computer Security techniques and tactics*. Elsevier, 2011.

[102] Sami Yangui, Iain-James Marshall, Jean-Pierre Laisne, and Samir Tata. Compatibleone : The open source cloud broker. *Journal of Grid Computing*, 12(1) :93–109, 2014.

[103] Lamia Youseff, Rich Wolski, Brent Gorda, and Chandra Krintz. Paravirtualization for hpc systems. In *Frontiers of High Performance Computing and Networking–ISPA 2006 Workshops*, pages 474–486. Springer, 2006.

Annexes

Annexe A

Interface utilisateur de la plateforme DZ-CBroker

A.1 Authentification

Afin que l'utilisateur de l'application *DZ-CBroker* puisse utiliser la plateforme d'interopérabilité *DZ-CBroker*, il doit mettre l'Url de l'application : `http://www.dzcbroker.arn.dz/CloudBroker/main` dans le navigateur pour s'authentifier soit autant qu'un utilisateur *User Grid* ou qu'un administrateur *Grid Manager* (voir la figure A.1).

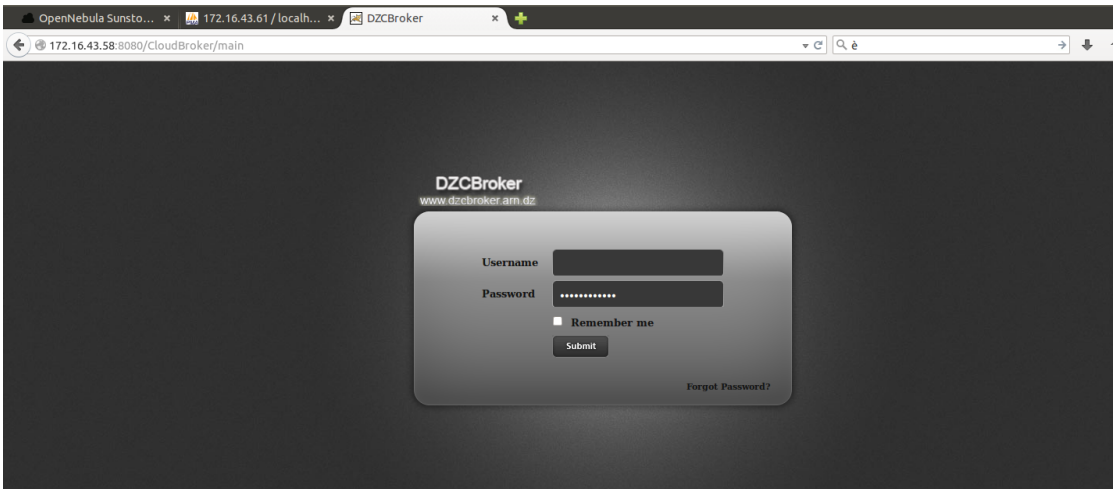

FIGURE A.1: Page d'authentification

Nous allons détailler dans la suite les types d'interfaces utilisateur :

A.2 Grid Manager interface

Dès que l'utilisateur *Grid Manager* s'authentifie, il aura son propre menu, qui assure plusieurs fonctionnalités afin de gérer l'application *DZ-CBroker*. Ces fonctionnalités sont détaillées comme suit :

1. **Ajouter un nouveau utilistaeur** :

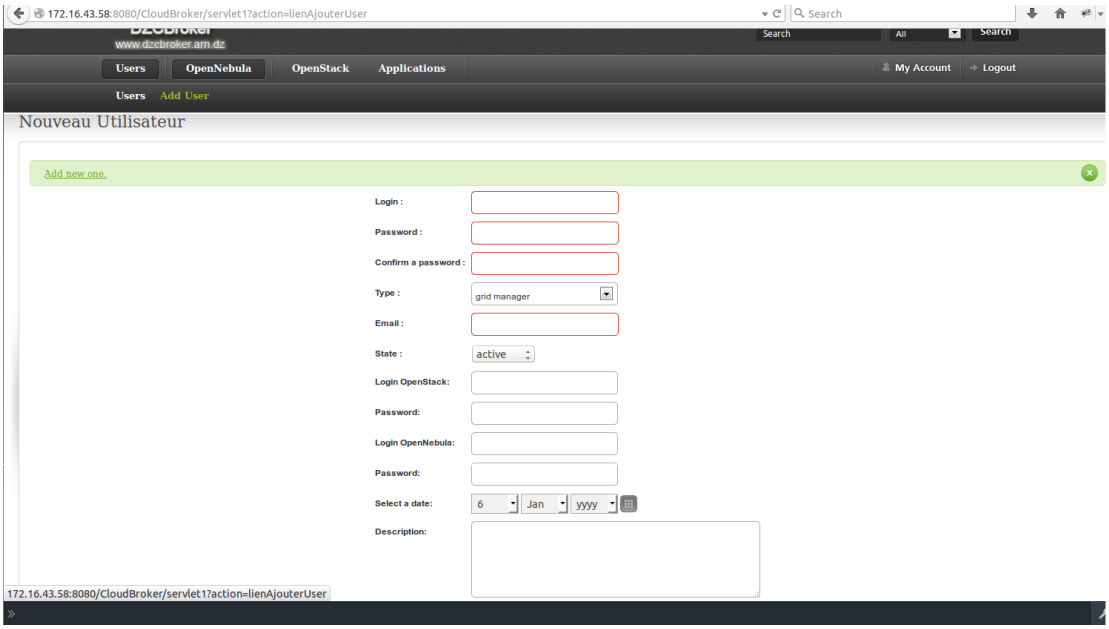

FIGURE A.2: Ajouter un utilisateur

Dans le sous menu *Add User* le *Grid Manager* doit créer un compte pour chaque utilisateur de l'application *DZ-CBroker*. Il faut noter qu'avant la création d'un compte au niveau de *DZ-CBroker*, l'administrateur devrait avoir déjà créé les comptes utilisateurs au niveau des plateformes cloud OpenNebula et OpenStack.

2. **Gérer les utilisateurs**

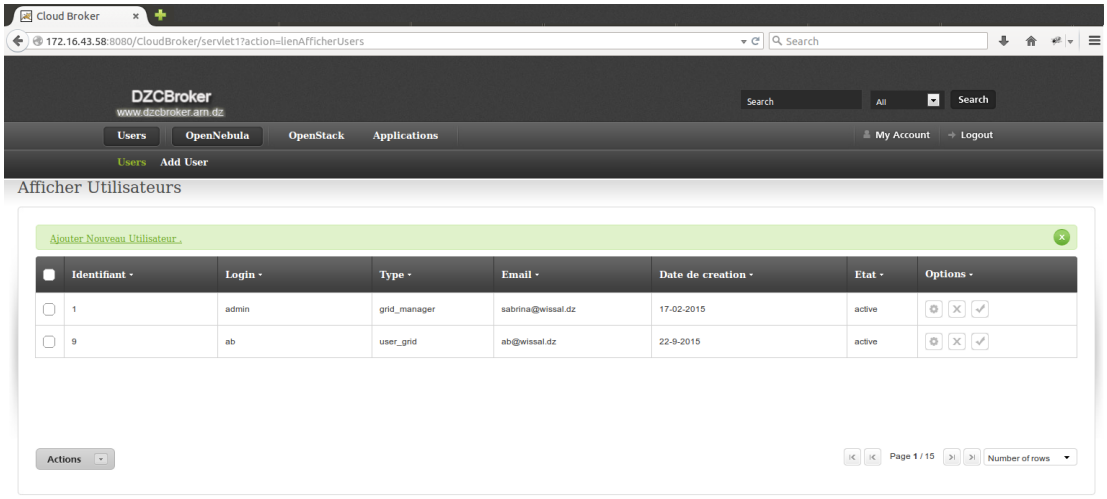

FIGURE A.3: Gérer les utilisateurs

Dans le sous menu *Users*, le *Grid Manager* peut voir tous les utilisateurs de la plateforme *DZ-CBroker*. Où il peut gérer ces utilisateurs en modifiant, activant/désactivant ou supprimant leurs comptes.

3. **Consulter les infrastructures**

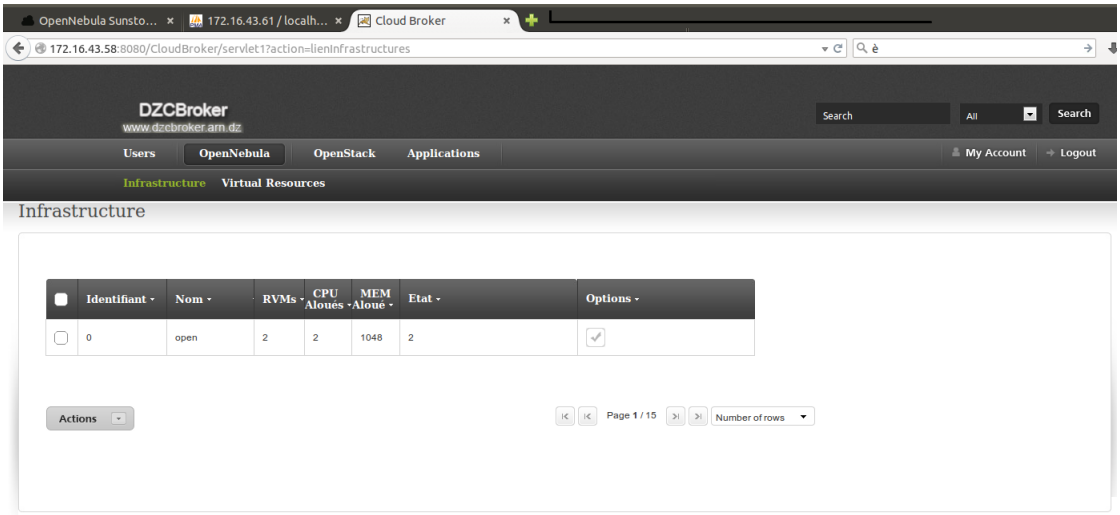

FIGURE A.4: L'infrastructure OpenNebula

Dans le menu OpenNebual et le menu OpenStack le *Grid Manager* peut consulter l'état des machines physiques de chaque Cloud en terme de disponibilité de ressource.

4. Ajouter une application scientifique

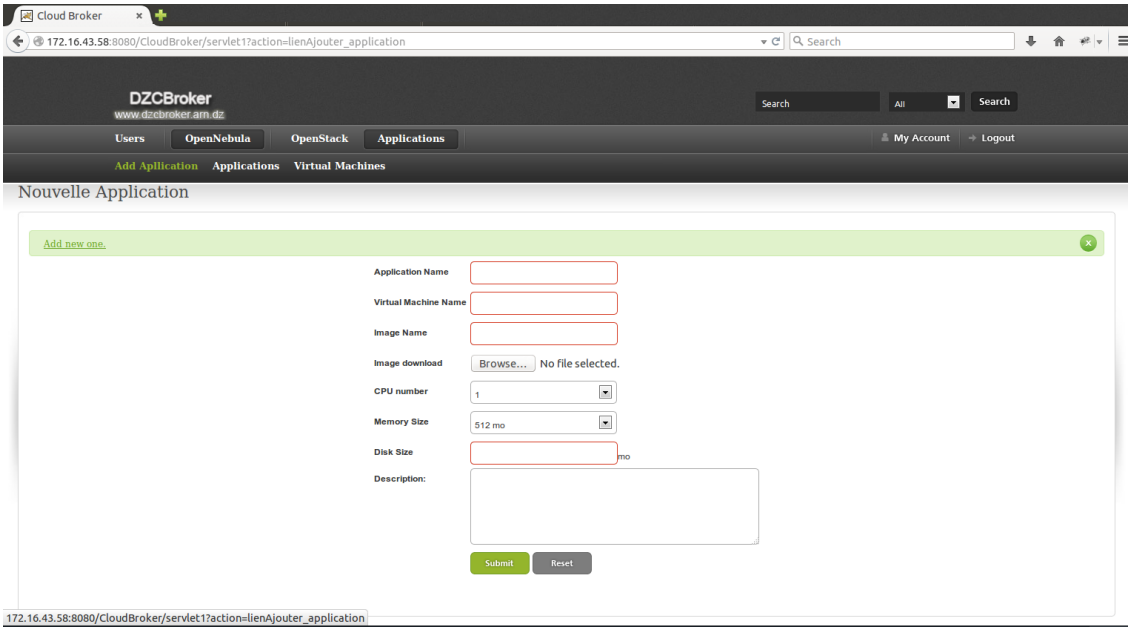

FIGURE A.5: Ajouter une application scientifique

Dans le sous-menu *Add Application* le *Grid Manager* peut ajouter une application scientifique, en introduisant les données relatives à cette dernière et importer son *Image grille*, pour que le système remplisse le fichier *Manifest* (voir la figure A.6) et enregistre l'image grille dans les *repositories* des différentes plateformes Clouds.

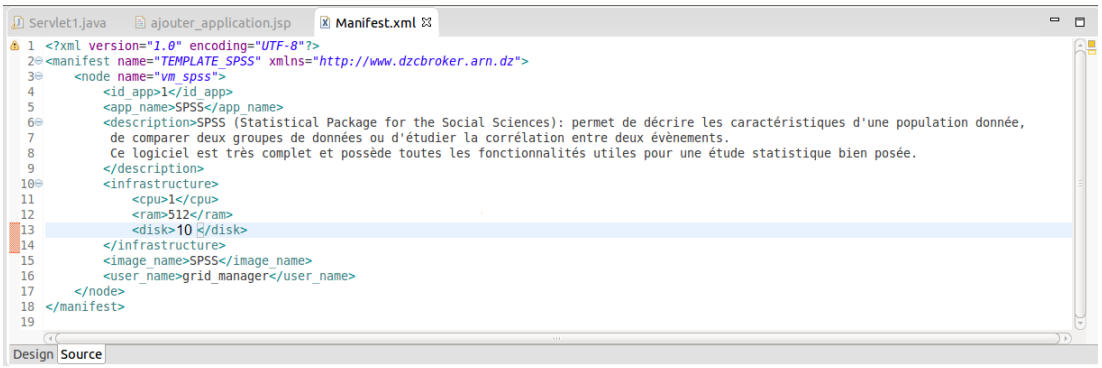

FIGURE A.6: Modèle d'un fichier Manifest

5. Gérer les applications grid

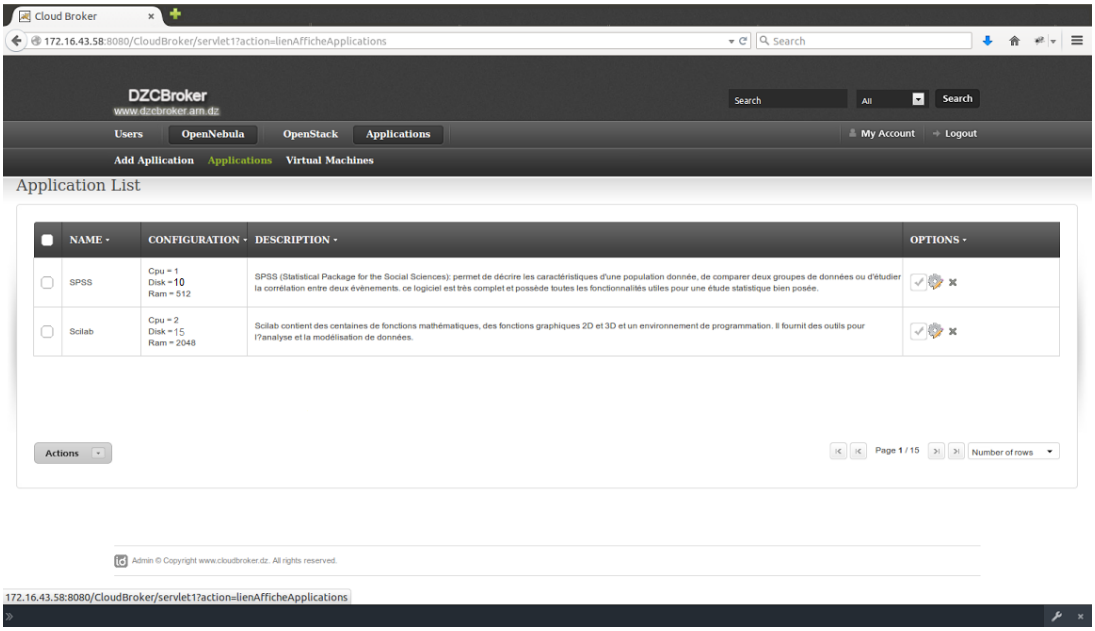

FIGURE A.7: Gérer les applications grid

Dans le sous-menu *Applications* le *Grid Manager* peut gérer toutes les applications enregistrées dans le système.

6. **Gérer les machines virtuelles**

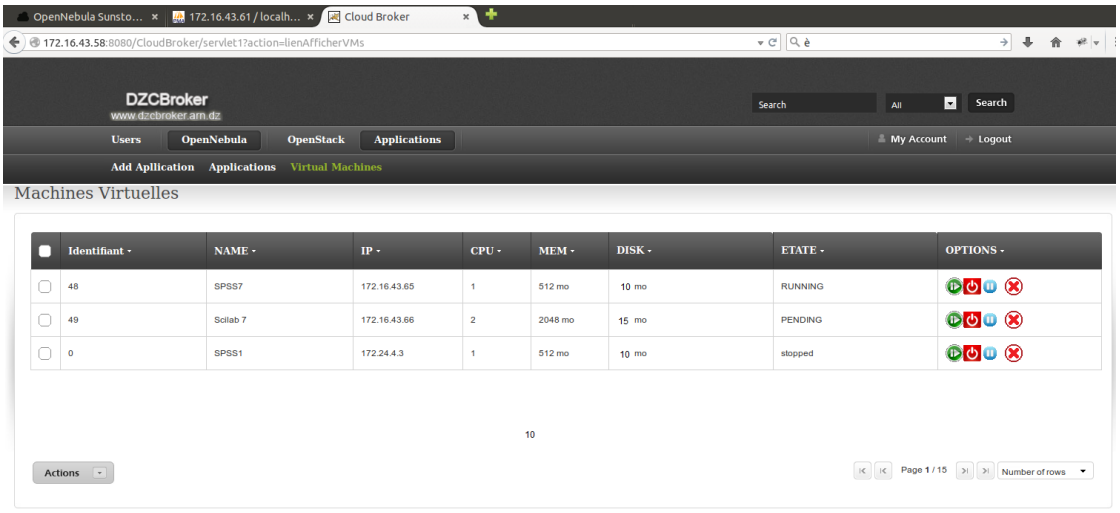

FIGURE A.8: Gérer les machines virtuelles

Dans le sous-menu *Virtual Machines*, le *Grid Manager* peut voir toutes les

machines virtuelles créées par tous les utilisateurs. Il a aussi toutes les permissions de gestion de ces différentes machines virtuelles.

A.3 User grid interface

Après que l'utilisateur *User Grid* a récupéré les données d'exécution d'une application scientifique qui s'est exécutée sur la grille *DZ eScience Grid*, l'administrateur *Grid Manager* lui crée un compte dans la plateforme *DZ-CBroker*. Puis l'utilisateur *Grid User* s'authentifie dans cette plateforme pour choisir une application scientifique s'exécutant dans une machine virtuelle dans un système multi-cloud afin d'analyser ses données.

1. **Créer une machine virtuelle**

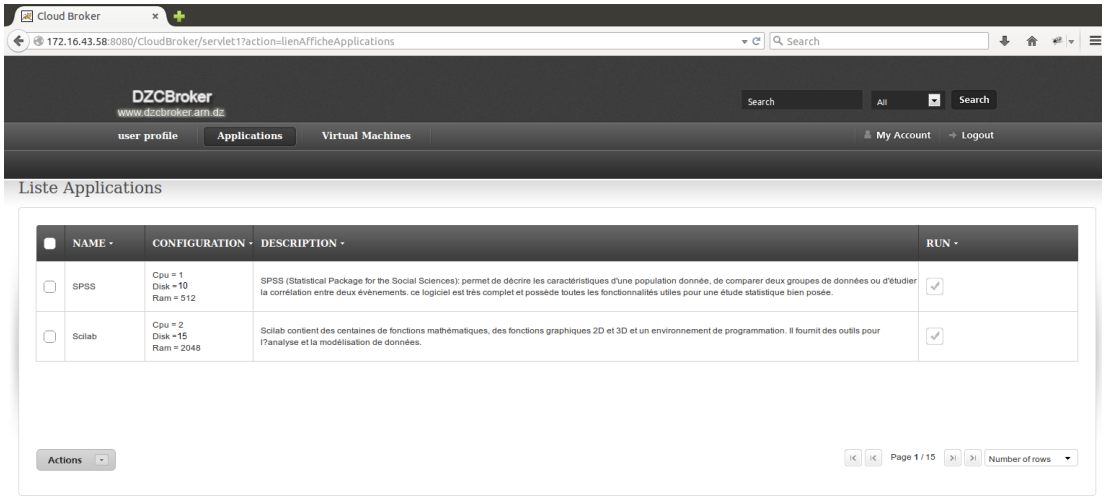

FIGURE A.9: Créer une machine virtuelle

Dans le menu *Applications* l'utilisateur trouve une liste des applications scientifiques avec leurs descriptions. Il choisit l'application convenable à ses besoins, puis il clique sur le bouton *exécuter* afin de permettre à la plateforme *DZ-CBroker* d'exécuter l'algorithme de placement pour créer la machine virtuelle contenant cette application d'une manière transparente dans un environnement multi-cloud.

2. **Gérer les machines virtuelles**

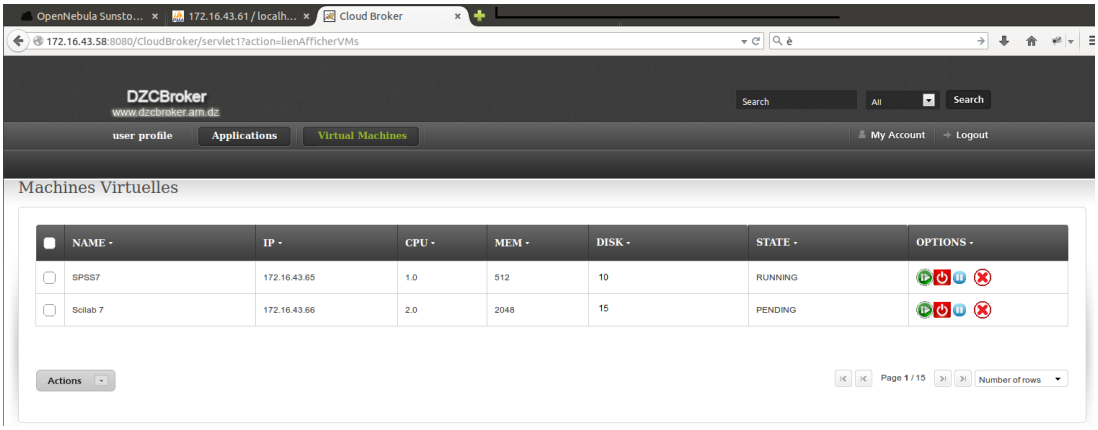

FIGURE A.10: gérer les machines virtuelles

Dans le menu *Virtual Machines*, l'utilisateur *Grid User* peut voir toutes ses machines virtuelles. Il peut aussi gérer le cycle de vie de ses machines virtuelles facilement en exploitant les fonctions suivantes : lancer, suspendre, arrêter et supprimer.