République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Superieur et de la Recherche Scientifique
Université Abderrahmane Mira de Bejaia
Faculté des Sciences Exactes Département d'Informatique

MÉMOIRE

Présenté par

Ikram ADLI [1]

Pour l'obtention du diplôme de Magister

Filière : Informatique
Option : Cloud Computing

Thème

Monitoring des "applications scientifiques" dans un environnement combiné "Cloud-Grid"

Soutenue le ....., devant le Jury composé de :

Mr. BOUKERRAM Abdellah	Professeur	Université de Bejaia	Président
Mr. TARI Abdelkamel	Professeur	Université de Bejaia	Rapporteur
Mr. SLIMANI Hachem	MCA	Université de Bejaia	Examinateur
Mme. EL-MAOUHAB Aouaouche	CR	CERIST, Alger	Examinateur
Mme. EL-MAOUHAB Aouaouche	CR	CERIST, Alger	Invitée

Promotion : 2012/2013

[1] adli@wissal.dz - CERIST

Remerciements

Tout d'abord, je tiens à remercier mon Directeur de thèse, Mr TARI Abdelkamel, Professeur à l'Université de Béjaia, de m'avoir fait confiance en acceptant d'encadrer ce travail.

Mes remerciements s'adressent aussi à Mme A. EL-MAOUHAB chargée de recherche au CERIST, pour la qualité de son encadrement, pour ses compétences scientifiques et techniques, pour ses encouragements durant cette longue épreuve, pour son infinie patience et sa disponibilité à tout moment.

Je remercie Mr BOUKERRAM Abdellah, Professeur à l'Université de Béjaia, qui me fait l'honneur de présider ce jury.

Je remercie Mr SLIMANI Hachem, Maître de Conférence A à l'Université de Béjaia, d'avoir bien voulu juger ce travail.

Mes remerciements seraient incomplets si je n'en adressais pas à l'ensemble du personnel de la Division Recherche et Développement en Réseaux pour leur soutien et leurs encouragements.

J'exprime ma gratitude à toutes mes amies notamment Hassina qui m'a beaucoup aidé, Houria et Ouafa qui m'ont aidée de manière bénévole et spontanée.

Ma reconnaissance va à ceux qui ont plus particulièrement assuré le soutien affectif : mes enfants, ma mère et ma famille.

Un grand merci à Rachid, mon mari, pour son soutien et appui inconditionnel.

Résumé

L'intégration entre l'infrastructure grille et Cloud est considérée comme un nouveau paradigme qui améliore la gestion de la grille dans son modèle actuel car le système d'administration Cloud IaaS fournit une fléxibilité pour faciliter et simplifier la gestion des ressources déployées pour l'exécution des applications scientifiques. Cependant, l'environnement "Grid-Cloud" rajoute une complexité dans l'architecture résultante, et la virtualisation des services de calcul entraîne inévitablement des pertes de performance par rapport à l'utilisation directe des ressources physiques, ainsi que l'imprévisibilité de la charge de travail des applications scientifiques qui conduit généralement soit à une infrastructure sous-chargée ou à une infrastructure surchargée.

L'une des problématiques d'un environnement "Grid-Cloud" est de garantir une certaine performance des applications scientifiques qui s'exécutent dans un environnement virtualisé avec une charge de travail imprévisible. Dans ce cas de figure, un monitoring multi niveaux pour surveiller le comportement des applications scientifiques, suivi d'une interprétation de l'utilisation des ressources pour prédire périodiquement la demande future sont nécessaires. Cela permet de déterminer les besoins en ressources pour assurer la haute disponibilité des applications scientifiques dans un environnement "Grid-Cloud".

Dans le présent travail, nous nous appuyons sur un monitoring multi niveaux et une approche de modélisation pour prédire la performance des applications scientifiques et effectuer une mise à l'échelle automatique dans un environnement combiné "Grid-Cloud". Pour ce faire, nous avons proposé une approche multi niveaux de monitoring basée modèle de performance des applications scientifiques, et qui s'exécutent sur des services grille déployées dans un environnement cloud. Notre modèle de performance se base sur les approches de modélisation prédictives, plus précisement "l'analyse des séries chronologiques".

Mots-clés: Cloud Computing, Grid Computing, Application scientifique, Monitoring, Performance, Modèle de performance, Time Series, Autoscaling.

Abstract

The integration between Grid and Cloud is considered as a new paradigm improving the Grid management in its current model because the Cloud Management system Iaas provide flexibility to expedite and simplify resources management deployed for the execution of scientific applications. However Grid-Cloud environment add complexity to resulting architecture and virtualization of calculation services inevitably leads performance losses compared to using physical resources and unpredictability of scientist application workload, generally leads either to an infrastructure sub-loaded or infrastructure overloaded.

One of the "Grid-Cloud" environment challenges is to guarantee scientific applications performance running on virtualized environment with unpredictable workload. In this case, a multi-level monitoring of scientific applications, followed by an interpretation the use of resources for periodically predict future demand are required. This determines resource required to ensure high availability of scientific applications in a "Grid-Cloud" environment.

In this Thesis, we highlight Multilevel Monitoring and Modeling Approach to predict Scientific Application performance and make an automatic scaling on a "Grid-Cloud" combined environment To do this we have to proposed an multi-layer monitoring approach based on scientific application performance model running on Grid Services in Cloud Environment. Our performance model based on predictive modeling approaches, specially "time series analysis".

Keywords: Cloud Computing, Grid Computing, Scientific application, Monitoring, Performance, performance Model, Time Series, Autoscaling.

Contents

Résumé ii

Abstract iii

Introduction Générale 1

I Contexte et État de l'art 5

CHAPTER 1 Grid et Cloud Computing 6

1.1 Introduction . . . 6
1.2 Grid Computing . . . 7
1.2.1 Historique et définitions . . . 7
1.2.2 Caractéristiques . . . 8
1.2.3 Organisation virtuelle . . . 9
1.2.4 Architecture d'une grille (modèle en couche) . . . 9
1.2.5 Les grid services selon le Middleware EMI . . . 10
1.2.5.1 Service d'accés (UI) . . . 10
1.2.5.2 Service de sécurité . . . 11
1.2.5.3 Service d'information (IS) . . . 12
1.2.5.4 Système de gestion de la charge de travail (WMS) . . . 12
1.2.5.5 L'èlèment de calcul (CE) . . . 12
1.2.5.6 Les Noeuds de calcul ("WN" Worker Node) . . . 13
1.2.5.7 L'élément de stockage "SE" . . . 13
1.2.6 Le langage de description de job (JDL) . . . 14
1.2.7 Cycle de vie d'un job dans le middleware EMI . . . 14
1.3 Cloud Computing . . . 15
1.3.1 Chronologie et définitions . . . 15
1.3.2 Caractéristiques principales . . . 16
1.3.3 Modèles de service . . . 17
1.3.3.1 "Software as a Service (SaaS)" . . . 17
1.3.3.2 "Platform as a Service (PaaS)" . . . 18

1.3.3.3 “Infrastructure as a Service (IaaS)” . . . 18
1.3.4 Modèles de déploiement . . . 18
1.3.4.1 Cloud privé . . . 19
1.3.4.2 Cloud communautaire . . . 19
1.3.4.3 Cloud public . . . 19
1.3.4.4 Cloud hybride . . . 19
1.3.5 Acteurs . . . 19
1.3.6 Virtualisation . . . 20
1.3.7 Les défis du “Cloud” . . . 20
1.4 Cloud vs Grid . . . 22
1.4.1 Grids et Clouds comme alternatifs . . . 22
1.4.2 Intégration Grid et Cloud . . . 23
1.4.2.1 La Grille sur le Cloud . . . 23
1.4.2.2 Le Cloud sur la Grille . . . 24
1.5 Conclusion . . . 24

CHAPTER 2 Monitoring dans le Cloud 25

2.1 Introduction . . . 25
2.2 Motivations pour le monitoring dans le Cloud . . . 26
2.3 Concepts de base . . . 27
2.3.1 Couches (layers) . . . 27
2.3.2 Niveaux d’abstraction . . . 28
2.3.3 Tests et métriques . . . 28
2.3.3.1 basé calcul (Computation-based) . . . 29
2.3.3.2 basé réseaux (network-based) . . . 29
2.3.4 Le monitoring dans “Grid” vs ”Cloud” . . . 30
2.4 Propriétés . . . 30
2.5 Les plates-formes de monitoring à usage général . . . 32
2.6 Les plates-formes de monitoring spécifique Cloud . . . 32
2.7 Défis et orientations de la recherche . . . 33
2.8 Conclusion . . . 34

CHAPTER 3 Le monitoring applicatif et la modélisation de la performance 36

3.1 Introduction . . . 36
3.2 Monitoring des applications Cloud basé sur un modèle de performance . . . 37
3.2.1 Généralités et définitions . . . 37
3.2.2 Sélection des paramètres . . . 38
3.2.3 Les approches de modélisation de performance des applications . . . 39
3.2.3.1 Théorie des files d’attente . . . 39

3.2.3.2 Théorie de contrôle . . . 40
3.2.3.3 Apprentissage par renforcement (Q-Learning) . . . 41
3.2.3.4 Analyse des time series . . . 41
3.2.4 Approche de garantie de performance basée sur un modèle de performance 44
3.3 Monitoring des applications Cloud basé sur les évènements . . . 46
3.3.1 Approache basée sur le paradigme de traitement d'évènement complexe 47
3.4 Approche mOSAIC . . . 48
3.5 GMonE: Approche complète pour le cloud monitoring à base de vision . . . 48
3.6 Conclusion . . . 49

II Contribution **50**

CHAPTER 4 Système muli-niveaux basé modèle de performance **51**

4.1 Introduction . . . 51
4.2 Approche d'intégration "Grid-Cloud" . . . 51
4.2.1 Objectifs d'intégration "Grid-Cloud" . . . 52
4.2.2 L'Appliance Virtuelle "Virtual Appliance" . . . 53
4.2.3 Le concept en couches de l'approche "grid sur le cloud" . . . 53
4.3 Le système de monitoring Multi-Level basé modèle de performance . . . 54
4.3.1 Problématique et motivations . . . 54
4.4 Approche proposée . . . 56
4.4.1 Vue d'ensemble de l'approche . . . 57
4.4.2 Modélisation de l'approche . . . 58
4.5 Architecture du système multi niveaux basé modèle de performance . . . 60
4.5.1 Le module "multi-level Monitor" . . . 61
4.5.1.1 Les composants du Module . . . 61
4.5.2 Le module "Data Collector and Selector" . . . 63
4.5.2.1 Les composants du Module . . . 63
4.5.2.2 Métriques de performance des applications cloud . . . 63
4.5.2.3 Fonctionnement interne du module . . . 65
4.5.3 Le module "modèle de communication" . . . 66
4.5.3.1 Les composants du Module . . . 67
4.5.4 Le module "modèle de performance" . . . 67
4.5.4.1 Motivations du modèle proposé . . . 67
4.5.4.2 Méthodologie . . . 68
4.5.4.3 Les composants du Module . . . 69
4.5.4.4 Fonctionnement interne du module . . . 70
4.5.4.5 Approche de modélisation . . . 71
4.5.4.6 La prise de décision . . . 72

4.5.5 Le module Mise à l'échelle et notification . . . 76
4.6 Conclusion . . . 77

CHAPTER 5 Mise en oeuvre et Expérimentation 78

5.1 Introduction . . . 78
5.2 Déploiement de l'approche d'intégration "Grid-Cloud" . . . 79
5.2.1 Environnement de déploiement . . . 79
5.2.2 Etapes de déploiement de l'approche d'intégration "Grid-Cloud" . . . 79
5.2.3 Mise en oeuvre du Cloud IaaS "OpenStack" . . . 81
5.2.3.1 Préparation des machines du Cloud d'OpenStack . . . 81
5.2.3.2 Définition des Appliances Virtuelles et Templates . . . 82
5.2.3.3 Instanciation des services de calcul . . . 83
5.2.4 Tests d'intégration du site "DZ-ARN-Stack" avec le Cloud "OpenStack" 83
5.2.4.1 Intégration du service grille "CE-Stack" . . . 83
5.2.4.2 Intégration des services grille "WN-Stack" . . . 84
5.3 Implémentation du système "Multi-Level basé modèle de Performance" . . . 85
5.3.1 Choix des logiciels et outils du monitoring multi-niveaux . . . 85
5.3.1.1 Les agents de monitoring de Ganglia . . . 85
5.3.1.2 L'agent de monitoring "Host sFlow" . . . 86
5.3.1.3 Le Framework de monitoring NAGIOS . . . 88
5.3.1.4 L'outil de monitoring APEL . . . 89
5.3.1.5 L'API JMS et le courtier "Apache ActiveMQ" . . . 90
5.3.2 Environnement de développement . . . 91
5.3.3 Mise en oeuvre du système "Multi-Level basé modèle de performance" 91
5.3.3.1 Implémentation du module "multi-level Monitor" . . . 91
5.3.3.2 Implémentation du module "Data Colector and Selector" . . . 92
5.3.3.3 Implémentation du module de communication . . . 92
5.3.3.4 Implémentation du module "Modèle de performance" . . . 93
5.3.3.5 Module mise à l'échelle . . . 96
5.4 Expérimentation et évaluation . . . 98
5.4.1 Scénario "Case Study" . . . 99
5.4.1.1 Choix des métriques de performance . . . 100
5.4.1.2 Définition des règles pour le module "Mise à l'échelle" . . . 100
5.4.1.3 Scénario de l'exécution des jobs séquentiels . . . 102
5.4.1.4 Scénario des jobs parallèles . . . 105
5.4.2 Analyse de performance de l'approche . . . 107
5.5 Synthèse . . . 108

Conclusion Générale 110

Appendices **123**

CHAPTER A OpenStack : Plate-forme cloud **124**

A.1 Introduction 124
A.2 Historique 124
A.3 Composants OpenStack 125
A.4 Architectures OpenStack 127
A.4.1 Architecture logique 127
A.4.2 Architecture conceptuelle 127
A.4.3 Architecture technique 129
A.5 Installation OpenStack 129
A.5.1 Lancer une instance sur le cloud OpenStack 129

List of Figures

1.1 Architecture en couches d'une grille de calcul [19] . . . 9
1.2 Les services du middleware EMI [106] . . . 11
1.3 The Information Service [4]. . . . 12
1.4 Structure interne du WMS [106] . . . 13
1.5 Structure d'un CE [106] . . . 13
1.6 Cycle de vie d'un job [4] . . . 14
1.7 Service XaaS . . . 18
1.8 Référence NIST 2012 [73] . . . 20
1.9 Machine virtuelle . . . 21

2.1 Cloud monitoring: motivations, properties, basic concepts and open issues [12] 29

3.1 Théorie des files d'attente [84] . . . 40
3.2 Théorie de contrôle [22] . . . 41
3.3 Analyse des time series [74] . . . 42
3.4 Modèle de performance pour les applications Cloud [100] . . . 46
3.5 Vue d'ensemble du monitoring [71] . . . 47

4.1 "Virtual Appliance" . . . 53
4.2 Processus de création d'une VM. . . . 54
4.3 Le concept en couches de l'approche "grille sur le cloud" . . . 55
4.4 Diagramme conceptuel de l'approche . . . 58
4.5 Niveaux de traitement de l'approche monitoring . . . 59
4.6 Modélisation de l'approche . . . 59
4.7 Architecture de l'approche de monitoring . . . 60
4.8 Architecture du module "Multi-Level Monitor" . . . 62
4.9 Diagramme d'Architecture du Collecteur Niveau Application . . . 64
4.10 Diagramme de séquence du Module "Collection et Sélection des métriques" . 65
4.11 Mécanisme de communication entre les modules . . . 66
4.12 Diagramme d'activité du modèle de performance . . . 68
4.13 Architecture du modèle de performance . . . 69

4.14 Diagramme de séquence du Module “modèle de performance” 70
4.15 Prédiction des valeurs futures 71
4.16 Architecture de mise à l’échelle 73

5.1 Architecture de l’approche d’intégration 81
5.2 Architecture de l’environnement de test 82
5.3 Liste des instances VM déployées suivant l’approche “Grid-Cloud” 85
5.4 Schéma général de la mise en oeuvre 86
5.5 Outil de monitoring Ganglia 87
5.6 Architecture de Ganglia avec le protocol sflow 87
5.7 L’outil de monitoring Nagios intégré dans OpenStack 89
5.8 Principe de Plug-in de fonctionnement Ganglia-Nagios 89
5.9 Architecture Apel Client 90
5.10 Architecture du modèle de performance 93
5.11 Paramètres du modèle de prédiction 96
5.12 Architecture Heat OpenStack 96
5.13 OpenStack Dashboard 98
5.14 Les queues “Active MQ” des métriques de performance. 102
5.15 Exécution des jobs sur les worker nodes en mode parallèle 106
5.16 Histogrammes des prédiction des valeurs futures 107

A.1 Architecture logique 127
A.2 Architecture technique 127
A.3 Architecture conceptuelle 128

List of Tables

3.1 Métrique pertinentes [53] 38
3.1 Métrique pertinentes [53] 39

4.1 Métrique pertinentes [53] 64
4.1 Métrique pertinentes [53] 65
4.2 Les Eléments de la phase de définition du modèle de performance 69

5.1 Environnement de déploiement de l'approche "Grid-Cloud" 80
5.2 Déploiement des instances sur l'infrastructure Cloud "OpenStack" 84
5.3 Caractéristiques des machines 98
5.3 Caractéristiques des machines 99
5.4 Application-Level Metrics for job batch 100
5.5 Résultat du scénario des jobs séquentiels sur 5 heures d'exécution 103
5.5 Résultat du scénario des jobs séquentiels sur 5 heures d'exécution 104
5.6 Résultat du scénario des jobs parallèles sur 3 heures d'exécution 105
5.6 Résultat du scénario des jobs parallèles sur 3 heures d'exécution 106

A.1 Versions d'OpenStack 124
A.1 Versions d'OpenStack 125

Listings

5.1 Partie de l'algorithme Moving Average 94
5.2 Programme de template WorkerNode.yaml 97
5.3 Programme d'un job parallèle whereami.jdl 105

Introduction Générale

Motivations et Problématique

Le Cloud Computing est un paradigme largement adopté pour la prestation de services sur Internet. Cela est dû à un certain nombre de raisons techniques, y compris: La virtualisation, l'optimisation de l'utilisation des ressources matérielles et logicielles, l'élasticité, l'isolement, la flexibilité et le schéma de service à la demande.

Initialement, l'idée de fournir des ressources informatiques à la demande a été développée dans le paradigme informatique de grille, qui vise à coordonner les ressources réparties sur différents domaines administratifs pour résoudre des problèmes de calcul [30].

Les grilles de calcul fournissent une large plateforme de calcul afin d'intégrer l'ensemble de toutes les ressources dans le but de délivrer un calcul de haute performance, mais l'architecture actuelle rencontre des problèmes majeurs [26] :

1. Avec la demande croissante des ressources par les utilisateurs et les calculs urgents des projets de recherche, certaines applications scientifiques peuvent ne pas obtenir les ressources nécessaires quand elles sont demandées;

2. La planification et la gestion des ressources dans l'infrastructure;

3. L'environnement pré-conditionné pour les applications qui seront exécutées conduit à des limitations sur les services offerts par l'environnement d'exécution.

Pour essayer de résoudre ces problèmes rencontrés dans les infrastructures existantes des grilles de calcul et offrir une architecture plus élastique, flexible et dynamique; des solutions *d'intégration entre l'infrastructure grille et Cloud* sont proposées [26, 46]. La combinaison des deux paradigmes améliore la gestion de la grille dans son modèle actuel car le système d'administration Cloud fournit une fléxibilité pour faciliter et simplifier la gestion des ressources déployées pour l'exécution des applications scientifiques.

Cependant, l'environnement "Grid-Cloud" rajoute une complexité dans l'architecture résultante, et la virtualisation des services de calcul entraîne inévitablement des pertes de performance par rapport à l'utilisation directe des ressources physiques, ainsi que l'imprévisibilité de la charge de travail des applications scientifiques qui conduit généralement soit à une infrastructure sous-chargée ou à une infrastructure surchargée [26, 46, 55]. A cet effet, une

prédiction de la performance, des outils de monitoring et des modèles pour l'analyse et le traitement des données du monitoring deviennent critiques et pertinentes.

La majorité des approches et plates-formes de monitoring proposées pour le scénario Grille ont été personnalisées pour les systèmes Cloud. Zanikolas et al. [11, 115] ont fait une étude dans le domaine de recherche du monitoring dans la grille en introduisant les concepts, les exigences et les phases impliquées ainsi que les activités de normalisation liées.

Le monitoring dans les environnements Cloud a été étudié dans la littérature [34, 31, 64, 43] pour la surveillance des ressources physiques et virtuelles des infrastructures de Cloud. Cependant, la majorité des solutions de monitoring se concentrent sur des métriques de bas niveau, telles que l'utilisation du CPU, la consommation de mémoire ou la bande passante du réseau. Nous soutenons que ces métriques, tout en étant sans doute pertinentes ne font pas saisir pleinement la performance réelle de l'application [71] et plus précisément les applications scientifiques qui ont des besoins spécifiques en matière de calcul haute performance pour une gestion efficace face à la qualité de service et les contraintes de la performance qui reste un défi.

Le monitoring de l'application est un défi difficile en raison des métriques surveillées de la plate-forme ou de la couche d'infrastructure, qui ne peuvent pas être facilement mises en correspondance avec les métriques nécessaires à la couche application [42].

A cet effet, des approches qui reposent sur des modèles de performance permettent d'analyser et traiter les données collectées afin de prédire les performances des applications pour la planification des capacités et la gestion des ressources [100, 67, 70]. Un modèle de performance rapporte quantitativement des métriques de qualité de service (QoS) au niveau de l'application (par exemple, le temps de réponse d'une application cliente) avec une ressource donnée [53].

Plusieurs approches de modélisation sont proposées dans la littérature, telles que la théorie des files d'attente, la théorie de contrôle, l'apprentissage par renforcement et l'analyse des times series. Les modèles de prédiction quantitatives tels que "l'analyse des times series" sont utilisées principalement pour prédire la charge du travail ou l'utilisation des ressources nécessaire pour traiter les demandes et satisfaire une certaine qualité de service. L'analyse des times series couvre un large éventail de méthodes proactives qui utilisent le passé pour prédir les valeurs futures et semble être un domaine de recherche prometteur [74].

L'objectif de notre travail consiste à proposer une approche multi niveaux de monitoring basée modèle de performance des applications scientifiques pour la prédiction de performance et la prise de décision pour une mise à l'échelle automatique dans un environnement "Grid-Cloud". Notre modèle de performance utilise "l'analyse des times series" comme approche de modélisation.

Contribution

Dans notre approche nous distinguons cinq principaux éléments qui sont, l'environnement d'intégration "Grid-Cloud", le module de monitoring multi-niveaux, le modèle de performance, le modèle de communication, et le module de mise à l'échelle à savoir:

Choix de l'approche d'intègration Grid-Cloud - l'une des approches proposées dans la littèrature concernant la combinaison des deux paradigmes "Grid" et "Cloud", qui a pour but de simplifier la gestion de la grille dans son modéle actuel en bénéficiant des avantages Cloud.

Un "système multi-niveaux de monitoring basé modèle de performance" - qui se compose de deux unités importantes:

1. Une unité de monitoring multi-niveaux capturant constamment des métriques et des indicateurs de performance en temps réel et à différents niveaux qui renseignent sur l'état des applications et de l'infrastructure (physique et virtuelle) et la compréhension et le choix précis des métriques quantitatives qui évaluent la performance des applications.
2. Proposer un modèle de performance qui traite et analyse les données collectées à partir de l'unité de monitoring pour décider de l'action de mise à l'échelle. Le but de notre modèle de performance est de fournir la capacité de prédire la performance des applications scientifiques pour un ensemble donné de ressources matérielles et *la planification des capacités et la gestion des ressources.*

Dans notre modèle de performance, le problème de modélisation de la perfomance d'une application scientifique est divisé en deux parties: La première partie, consiste à faire une estimation de la future charge de travail ou une utilisation de ressource en se basant sur des approches de modélisation de performance d'application plus précisement "l'analyse des séries chronologiques".

Sur la base de cette valeur prédite, la deuxième ètape consiste à décider de l'action appropriée à prendre à base des règles proactives et prédictives que nous avons définies, et qui déclenchent le redimensionement automatique (mise à l'échelle) ou des notifications aux administrateurs. Nous soulignons qu'une bonne utilisation des solutions de monitoring facilite énormement la prise de décision de mise à l'échelle.

Plan du document

Notre travail est présenté, dans le reste du document, organisé comme suit:

Chapitre 1, présente les deux paradigmes *Grid Computing* et *Cloud Computing*, des définitions, des comparaisons et des travaux de recherches sur leur intégration.

Dans le Chapitre 2, nous avons présenté une analyse dans le domaine du monitoring dans le Cloud, les principales propriétés et les concepts de base pour la construction d'un système de monitoring, ainsi que des solutions de monitoring en les classant en des catégories à usage général et spécifiques Cloud.

Dans le Chapitre 3, et dans la première partie de cette section, nous avons évoqué le monitoring applicatif basé sur un modèle de performance en présentant les approches de modélisation proposées dans la littérature.

Dans la seconde partie de cette section, nous avons évoqué des projets qui proposent de lever la complexité du monitoring applicatif en utilisant d'autre approches que la modélisation de la performance.

Chapitre 4, présente notre contribution qui consiste à choisir l'approche d'intégration "Grid-Cloud" et la conception du "système de monitoring multi niveaux basé modèle de performance", pour la prédiction et la prise de décision.

Chapitre 5, est la mise en oeuvre qui a servi à valider notre approche, ainsi qu'une évaluation et présentation des résultats réels.

Pour conclure, une discussion succède à ces parties où les perspectives donnent une vue future sur la continuation du présent travail avant de conclure le document.

Part I

Contexte et État de l'art

CHAPTER 1

Grid et Cloud Computing

1.1 Introduction

Initialement, l'idée de fournir des ressources informatiques à la demande a été développé dans le paradigme *"grille de calcul"*, mises en place dans le but de coordonner le partage des ressources réparties pour résoudre les problèmes intensifs à grande échelle. Le terme "grid" reflète l'analogie avec le réseau d'électricité, ce qui suggère que les infrastructures de grille devraient fournir la puissance de calcul à tout utilisateur, quelles que soient les détails des ressources sous-jacentes ou la complexité des problèmes. Comme les technologies Grid ont gagné une popularité croissante dans la communauté académique, des protocoles standards, middleware et services ont été proposés pour répondre aux diverses exigences des applications scientifiques [30]. Dans ce contexte, de nombreux efforts de recherche concentrés sur la fourniture d'une définition de la Grille [45, 47], la plus largement accepté est celle proposée par Ian Foster en 2002: "*un système qui coordonne des ressources informatiques dont l'administration n'est pas centralisée, des méthodes et des normes standardisées, ouvertes, pour des fins générales pour offrir une qualité de service non triviale* " [44].

Le Cloud Computing partage la même vision que le paradigme "Grid": il vise à fédérer les ressources de calcul distribué pour fournir des services à la demande à des clients externes. Cependant, les grilles étaient destinées à la communauté scientifique tandis que le cloud s'est tourné vers un modèle commercial où le client paye pour l'utilisation de ressources acquises à la demande [30]. Le Cloud Computing est un paradigme largement adopté pour la prestation de services sur Internet. Cela est dû à un certain nombre de raisons techniques, y compris: La virtualisation, l'optimisation de l'utilisation des ressources matériel et logicielles, l'élasticité, l'isolement de performance, la flexibilité et le schéma de service à la demande [16].

Dans ce que suit, nous présentons d'abord, les deux paradigmes grilles de calcul et Cloud computing. Enfin, nous présentons une étude comparative sur les deux technologies.

1.2 Grid Computing

1.2.1 Historique et définitions

L'idée serait venue de trois personnes, du Docteur en mathématiques et en informatique Ian Foster (Directeur du Laboratoire National Argonne à Chicago aux Etats-Unis), de Monsieur Carl Kesselman chercheur en Informatique à "The University of Southern California" et de Steve Tuecke ingénieur en informatique au Laboratoire National Argonne. Ces trois sont surnommés *"fathers of the Grid"* (Les Fondateurs du Grid), et sont à l'origine de l'une des plus importantes organisations pour le Grid "The Globus Alliance" [90].

Le mot *"grille"* vient de l'anglais *grid* qui a été choisi par analogie avec le système de distribution d'électricité américain (electric power grid), ce terme a été répandu en 1998 par l'ouvrage de Ian Foster et Carl Kesselman [45]. En effet, une grille peut être vue de la même manière que le réseau électrique fournit de la puissance électrique. La vision des inventeurs de ce terme est qu'il sera possible de se brancher sur une grille informatique pour obtenir de la puissance de calcul et/ou de stockage de données sans savoir ni où, ni comment cette puissance est fournie. Nous commençons par définir ce qu'est un site d'une grille, pour cela nous reprenons la définition de [68].

Definition 1.1. *Site : Un site est un ensemble de ressources informatiques localisées géographiquement dans une même organisation (campus universitaire, centre de calcul, entreprise ou chez un individu) et qui forment un domaine d'administration autonome, uniforme et coordonné.). Les ressources informatiques sont aussi bien des liens réseau, des machines (simples PC ou calculateurs parallèles) ou des éléments logiciels.*

Une grille informatique mutualise un ensemble de ressources informatiques géographiquement distribuées dans différents sites. Néanmoins, il n'existe pas de définitions trés précises des grilles de calcul.

Definition 1.2. *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 distribués lors de leur exécution en fonction de leur disponibilité, capacité, performance, coût, et des besoins des utilisateurs en terme de qualité de service [27].*

Mais la définition qui nous parait la plus complète est celle donnée par l'article "What is the Grid ? A Three Point Checklist" [44], où Ian Foster fait une synthèse de plusieurs définitions et sort avec une liste de trois caractéristiques pour un système de grille :

Definition 1.3.

1. *Des ressources coordonnées dont leur administration n'est pas centralisée,*
2. *Des méthodes utilisées qui sont standardisées (en utilisant des normes, ouvertes, des protocoles et des interfaces à des fins générales),*

3. *Délivrer une Qualité de Service non négligeable (non triviale) (temps de réponse, disponibilité, sécurité, etc.).*

Il existe différents types de grille informatique. On distingue notamment les grilles de données et les grilles de calcul.

- **Les grilles de données (data grid)** : Sont principalement utilisées pour le stockage de grandes masses de données. Elles sont généralement composées de ressources offrant une grande capacité de stockage, en mémoire et sur disque.

- **Les grilles de calcul (computational grid)** : Sont dédiées aux calculs intensifs. Une grande importance est alors donnée à la puissance des processeurs des noeuds qui la composent. Nous parlerons dans la suite de ce mémoire que de ce type de grilles car nous nous intéressons qu'a des applications scientifiques effectuant des calculs sur des grilles de calcul.

1.2.2 Caractéristiques

Une grille de calcul est une infrastructure virtuelle car les relations entre les entités qui la composent n'existent pas sur le plan matèriel mais d'un point de vue logique, elle est constituée d'un ensemble de ressources informatiques (tout élément qui permet l'exécution d'une tâche ou le stockage d'une donnée numérique : ordinateurs personnels, téléphones mobiles, calculatrices,etc) potentiellement [19], Ces ressources peuvent être :

1. **Coordonnées** : Elles sont organisées, connectées et gérées en fonction de besoins (objectifs) et contraintes (environnements). Il faut donc coordonner correctement ces différentes ressources.

2. **Partagées**: Elles sont à disposition de nombreux utilisateurs répartis en groupes appelés organisations virtuelles (Virtual Organization ou VO).

3. **Hétérogènes** : Elles peuvent être de natures complétement différentes aussi bien au niveau matériel que logiciel.

4. **Externalisées** : Elles peuvent être réparties sur un grand nombre de sites différents.

5. **Sans administration centralisé** : Il n'existe pas d'unité centralisée. Contrairement à un cluster, les ressources sont hors de la portée d'un moniteur de contrôle. (L'administration est distribuée sur chacun des nœuds.)

6. **Distribuées** : elles sont situées dans des lieux géographiques différents.

1.2.3 Organisation virtuelle

L'accés à distance à différentes ressources doit être fortement contrôlé afin de garantir un bon niveau de sécurité; Les fournisseurs de ressources et les utilisateurs doivent donc définir clairement ce qui est partagé, qui est autorisé à l'utiliser, quand et comment. Un ensemble d'individus et/ou d'institutions soumis à une telle charte de partage s'appelle *une organisation virtuelle* (VO).

1.2.4 Architecture d'une grille (modèle en couche)

Selon Baker et al [19] l'architecture des grilles de calcul peut être décrite en termes de couches. Les couches du haut sont orientées "utilisateur" tandis que les couches du bas, représentent les éléments constituant l'infrastructure hardware et réseau.

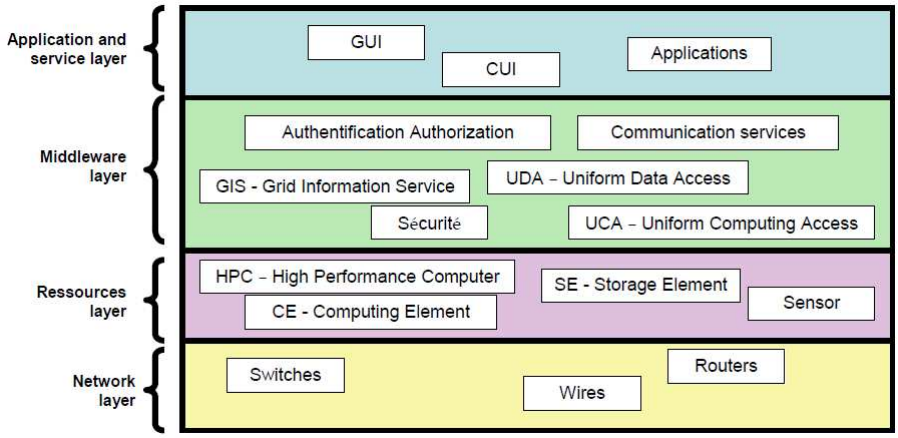

Figure 1.1: Architecture en couches d'une grille de calcul [19]

- **La couche applicative**

  C'est La couche "visible" par les utilisateurs. Elle contient les applications qui seront utilisées par les utilisateurs de la grille. Cette couche donne l'accés aux ressources soit par l'intermédiaire d'interfaces graphiques GUI (Graphical User Interface) ou en mode commande CUI (Command User Interface), elle peut être aussi intégrées dans les logiciels spécifiques.

- **La couche middleware ou intergiciel**

  L'ensemble des logiciels assurant la gestion de la grille est dénommé "l'intergiciel" (middleware) de la grille. Celui-ci représente en quelque sorte le "moteur" de grille, il gère toutes les ressources de la grille, et est informé en permanence de leur état : Unités de calcul et de stockage constitutives des nœuds, ne doit pas seulement trouver les données dont les scientifiques ont besoin, mais également utiliser les programmes et les ressources de calculs nécessaires pour les exécuter. Les tâches doivent être distribuées

n'importe où dans le monde dés qu'il y a une ressource suffisante disponible, puis retourner le résultat aux scientifiques.
Le middleware assure aux utilisateurs et aux applications les services de :

- Réservation et allocation des ressources nécessaires;
- Ordonnancement et lancement des travaux;
- Suivi de l'activité;
- Administration du système.

La grille Algérienne (le DZ-GRID) qui comprend jusque-là trois clusters installés sur trois sites : DZ-01 (au niveau du CERIST-Alger), DZ-02 (au niveau de l'université de Batna), DZ-03 (au niveau de l'université de Sétif) utilise le middleware EMI, nous allons par conséquent définir dans ce qui suit les composants d'une grille de calcul basée sur EMI.

- **La couche ressource**

  Elle est constituée des différentes ressources mises à la disposition de la grille, tels que les processeurs, les espaces de mémoire, les catalogues de données, les softwares modulaires ainsi que tout type d'appareils.

- **La couche réseau**

  C'est la couche de base. Elle propose les protocoles nécessaires à la communication entre les nœuds de la grille. Elle symbolise l'infrastructure de télécommunication.

1.2.5 Les grid services selon le Middleware EMI

Definition 1.4. *EMI "European Middleware Initiative" est un projet de collaboration entre plusieurs fournisseurs de Middlewares (ARC[1], gLite, UNICORE[2], dCache[3]), Financé par la Commission Européenne afin de produire une solution intergicielle cohérente et interopérable [4].*

L'architecture de la grille de calcul suivant *EMI* se compose d'un ensemble de services nommés aussi *"grid services"* de haut niveau nous présentons les plus importants [106] (voir figure 1.2): L'accés à la grille, la sécurité, le système d'information et de monitoring, le système de gestion de la charge de travail et la Gestion des données.

1.2.5.1 Service d'accés (UI)

Le point d'accés à la grille est l'interface utilisateur (UI). A partir D'une interface, un utilisateur peut être authentifié et autorisé à utiliser les ressources de la grille, et peut accéder à

[1] http://www.nordugrid.org/arc/
[2] http://www.unicore.eu/
[3] http://www.dcache.org/

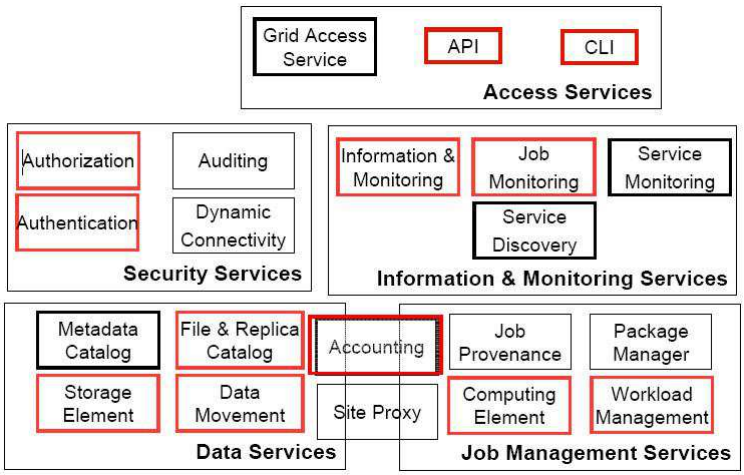

Figure 1.2: Les services du middleware EMI [106]

des fonctionnalités offertes par les systèmes de gestion de l'information, la charge de travail et des données. Une UI permet à travers une CLI (Command Line Interface) fournie par EMI d'effectuer les opérations de base sur la grille [4]:

- Lister toutes les ressources adéquates pour exécuter un job donné;
- Annuler des jobs;
- Interroger l'état des jobs et récupérer leur production;
- Copier, reproduire et supprimer des fichiers de la grille;
- présenter et gérer les jobs de transfert de fichiers;
- Récupérer l'état des différentes ressources du système d'information.

1.2.5.2 Service de sécurité

Pour pouvoir accéder aux ressources de la grille l'utilisateur doit être capable de s'authentifier auprés de ces ressources, il doit ainsi posséder un *"certificat numérique"* X.509 (passeport électronique) qui est une clé publique cryptée associée à une clé privée, le certificat est délivré par une autorité de certification *"CA"* (Certification Authority).
L'utilisateur doit ensuite être inscrit dans une "VO" (Virtual Organization). Muni de son certificat, il peut s'inscrire dans une ou plusieurs VOs. Sa demande sera ensuite validée par le "VO Manager". Le certificat d'utilisateur, dont la clé privée est protégée par un mot de passe, est utilisé pour générer et signer un certificat temporaire, appelé un "certificat proxy" (ou simplement un proxy) qui permet l'identification de l'utilisateur ainsi que ses jobs lors de leur exécution [4].

1.2.5.3 Service d'information (IS)

Le service d'information fournit des informations sur l'existence de ressources de la grille et des informations sur leur structure et leur état. Cette information est essentielle pour l'utilisation et l'exploitation de la grille.Les informations publiées sur la grille par le SI servent à la supervision et l'étude des performances des ressources [4].

Le SI utilisé par EMI est le BDII (Berkley Database Information Index), il existe deux types de machines BDII :

1. Site-BDII : installé au niveau de chaque site, il rapporte le statut à l'échelle du site lui même
2. Top-BDII : il représente le nœud supérieur de la grille, recueille les informations de tous les sites appartenant à la grille à travers les sites-BDII.

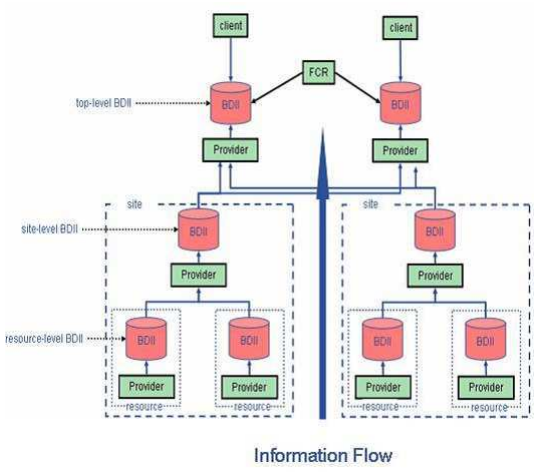

Figure 1.3: The Information Service [4].

1.2.5.4 Système de gestion de la charge de travail (WMS)

C'est le chef d'orchestre de la grille. Il permet d'accepter des jobs d'utilisateurs, de les affecter à l'élément de calcul (CE) le plus approprié, à enregistrer leur statut et de récupérer leur sortie [4]. La figure 1.4 nous montre la structure interne du WMS.

1.2.5.5 L'èlèment de calcul (CE)

Le CE est un service qui repésente une ressource informatique, il est le point d'entrée sur les fermes de calcul [4]. Les applications et logiciels dont ont besoin les utilisateurs y sont installées par un utilisateur ayant un rôle spécial dans la VO (software manager). Le CE est composé d'un (voir la figure 1.5):

- Système d'information

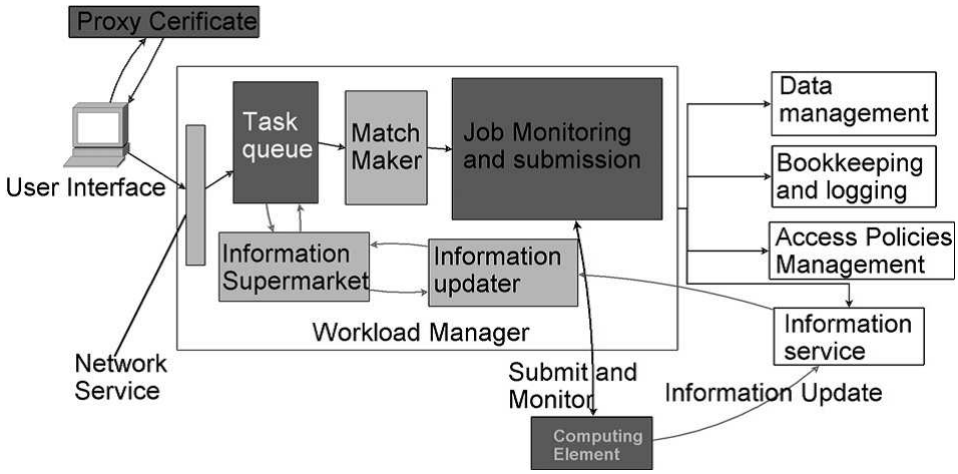

Figure 1.4: Structure interne du WMS [106]

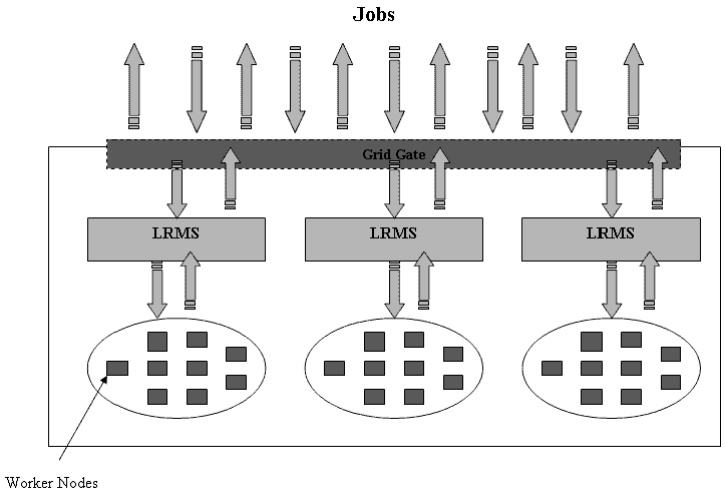

Figure 1.5: Structure d'un CE [106]

- Système de Batchs 4 local (Exemple : Torque / MAUI)
- Ensemble de Worker Nodes (machines homogènes en général).

Le CE soumet les jobs aux worker nodes via le système de batch local.

1.2.5.6 Les Noeuds de calcul ("WN" Worker Node)

Ce sont les noeuds qui fournissent les services de calcul et d'exécution des jobs envoyés par le CE, chaque site doit en posséder plusieurs.

1.2.5.7 L'élément de stockage "SE"

Ces noeuds permettent d'accéder aux disques de stockages massifs. La gestion de l'espace disque est assurée par une solution légère à l'aide du DPM (Disk Pool Manager) [4].

1.2.6 Le langage de description de job (JDL)

Un job (fichier de description d'un job) est construit avec une séquence d'attributs dans un langage appelé JDL (Job Description Language). Il y a deux grandes catégories d'attributs :

1. Ceux concernant le job : Définissent le job lui-même;
2. Ceux concernant les ressources.

- Ils sont utilisées par le WMS pour déterminer les ressources néecessaires au job;
- Ils permettent de préciser les caractéristiques de calcul requises (librairies disponibles, etc.);
- Ils permettent de définir les caractéristiques liées aux données : données entrantes, SE utilisée pour les données, les protocoles daccés aux données, etc.

1.2.7 Cycle de vie d'un job dans le middleware EMI

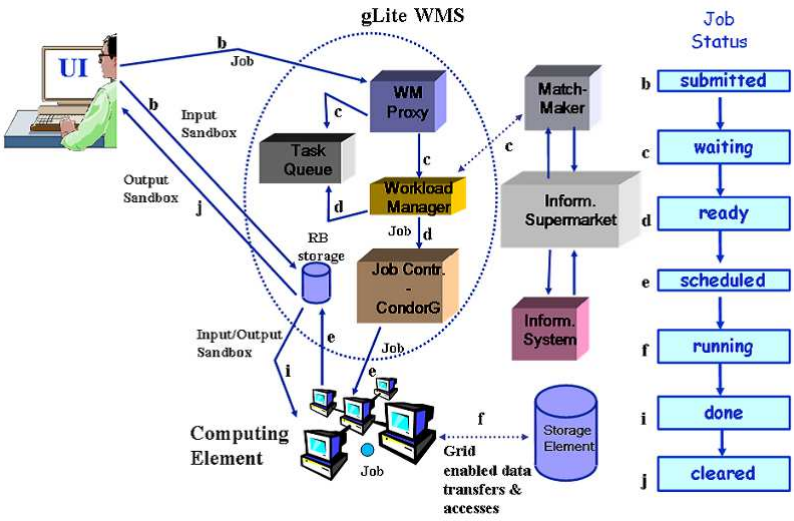

Figure 1.6: Cycle de vie d'un job [4]

Le cycle de vie d'un job exécuté sur une grille de calcul basé sur le middleware EMI peut être schématisé comme suit [4] :

1. S'inscrire dans une organisation virtuelle (VO) pour posséder un certificat numérique d'une autorité de certification digne de confiance et obtenir un compte UI.L'utilisateur se connecte et crée un proxy qui lui permet d'interagir en toute sécurité avec le système.
2. Le job est soumis par l'utilisateur via le User Interface (UI), et transféré au WMS mais pas encore été traité par le serveur WMProxy, il est considéré comme *"SUBMITTED"*.

3. Le job est accepté par le serveur mais pas encore été pris en charge par le WMS, il est considéré comme *"WAITING"*.

4. 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 . Lorsque le job est assigné à un Computing Element (CE) approprié mais n'y est pas encore transféré, il est considéré comme *"READY"*.

5. Le job ainsi que l'InputSandBox sont transférés au CE qui prend en charge le job dans sa queue (liste d'attente), le job et est dans l'état *"SCHEDULED"*.

6. Le CE envoie le job sur un ou plusieurs WN (s) disponibles, le job est en cours d'exécution sur le WN il prend alors l'état *"RUNNING"*.

7. Lorsque le job est terminé, les fichiers produits par celui-ci sont disponibles sur le LRMS (Local Resource Management System). Le WMS est averti que le job s'est terminé, le job devient *"DONE (OK)"* s'il se termine avec succés.

8. Le WMS récupère les fichiers de sortie dans l'OutputSandBox et envoie les résultats (l'OutputSandBox) à l'utilisateur via l'UI, le job est *"CLEARED"*.

9. Si l'exécution du job échoue alors l'état du job retourne *"DONE(Failed)"*.

10. A n'importe quel moment l'utilisateur peut arrêter son job sur la grille, et le job est considéré comme *"CANCELED"*.

11. Comme le job peut rester trop longtemps sur la grille jusqu'à ce que le proxy expire et dans ce cas le job est automatiquement arrêté par le WMS et il se met à l'état *"ABORTED"*.

12. 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.

1.3 Cloud Computing

1.3.1 Chronologie et définitions

L'informatique en nuage, informatique dématérialisée, ou encore infonuagique (Cloud computing) est un nouveau modèle informatique qui consiste à proposer les services informatiques sous forme des services à la demande accessibles de n'importe où, n'importe quand et par n'importe qui. L'informatique en nuage consiste à dématérialiser les ressources informatiques et d'y accéder à l'aide d'un service, un concept qui est emprunté à l'architecture orientée services (SOA).

- **Chronologie** : Le concept remonte à 1960, lorsque John McCarthy a estimé que le calcul peut un jour être organisé comme un service d'utilité publique [49].

  Le Cloud est la 5e génération d'infrastructures informatiques [28] : mainframe (1970), client-serveur (1980), web (1990), SOA (2000) et Cloud (20 ? ?). Il n'y a pas de date-clé à laquelle nous puissions dire que le Cloud computing est né. Amazon a dévoilé sa version d'essai d'Elastic Computing Cloud (EC2) le 24 août 2006. Une telle date peut être considérée comme date de naissance. L'expression est devenue populaire en 2007.Le Cloud computing met en oeuvre l'idée d'informatique utilitaire du type service public, proposé par John McCarthy. Un tel concept peut être comparé au cluster de calcul ou encore à l'informatique en grille.

- **Définition** : Le Cloud Computing est une métaphore désignant un réseau de ressources informatiques (logicielles et/ou matérielles) accessibles à distance par le biais des technologies Internet [28, 82, 16, 110, 116, 17]. C'est un modèle informatique selon lequel des ressources informatiques sont fournies sous la forme d'un *"service à la demande"*. Ainsi les services de l'informatique en nuage reposent sur un concept de *"paiement à l'utilisation"*.

  De nombreuses définitions ont été données du Cloud Computing. Nous citons deux définitions : la première de NIST [18, 81] et la deuxième de Buyya [28] :

  Definition 1.5. *Un* ***modèle pay-per-use*** *pour permettre un accès réseau pratique et disponible* ***à la demande*** *à un ensemble de ressources informatiques (réseaux, serveurs, stockage, applications, services) configurables qui peuvent être* ***rapidement provisionnées et publiées*** *avec un minimum d'effort de gestion ou d'interaction du fournisseur des services.*

  Definition 1.6. *Le Cloud est* ***un système de calcul réparti et parallèle*** *comprenant un ensemble d'ordinateurs* ***virtualisés*** *et interconnectés qui sont* ***dynamiquement provisionnés*** *et présentés comme des ressources informatiques basées sur l'accord de service à niveau SLA "Service Level Agreement" établi entre le fournisseur et le client.*

1.3.2 Caractéristiques principales

Le Cloud Computing est une nouvelle façon de délivrer les ressources informatiques, et non une nouvelle technologie. Il se caractérise par [82] :

- **Un accès en libre service à la demande** : Un client pourra commander des ressources informatiques en fonction de ses besoins. Les ressources informatiques sont fournies d'une manière entièrement automatisée et c'est le client, au moyen d'une interface, qui met en place et gère la configuration à distance.

- **Un accès ubiquitaire au réseau** : Les capacités sont disponibles sur le réseau et accessibles par des mécanismes standards, qui favorisent l'accès au service par des clients lourds ou légers via des plates-formes hétérogènes.

- **Une mise en commun des ressources** : Les ressources informatiques sont mises à la disposition des clients sur un modèle multi-locataires, avec une attribution dynamique des ressources physiques et virtuelles en fonction de la demande. Le client n'a généralement ni contrôle, ni connaissance sur l'emplacement des ressources, mais est en mesure de le spécifier à un plus haut niveau d'abstraction (pays, état ou centre de données).Ce partage des ressources est la caractéristique qui différencie le Cloud Computing de l'infogérance.

- **Une élasticité rapide** : Les capacités informatiques mises à disposition du client peuvent être ajustées (augmenter ou diminuer) rapidement (quelques minutes voire quelques secondes) en fonction des besoins et/ou de la charge et de façon automatique dans certains cas. L'utilisateur a l'illusion d'avoir accès à des ressources illimitées bien que le fournisseur définit toujours un seuil (par exemple : 20 instances par zone est le maximum possible pour Amazon EC2).

- **Un service mesuré en permanence** : Les ressources consommées sont contrôlées et communiquées au client et au fournisseur de service de façon transparente. Cela garantit un niveau de disponibilité adapté aux besoins spécifiques des clients.

Pour conclure, la somme de ces cinq caractéristiques permet aujourd'hui de dire si un service proposé est vraiment de type Cloud computing ou non.

1.3.3 Modèles de service

XaaS (X as a Service) est à la base du paradigme de l'informatique en nuage. Nous dénombrons 3 modèles de service [82] comme illustré dans la Figure 1.7 :

1.3.3.1 "Software as a Service (SaaS)"

Logiciels en tant que service "SaaS" (Software as a Service) où le matériel, l'hébergement, le framework d'application et le logiciel sont dématérialisés.La capacité fournie au client est d'utiliser les applications du fournisseur s'exécutant sur une infrastructure cloud.Le client accède à l'application à distance via une interface disponible, ou via des APIs fournies et ne paye pas pour posséder l'application en elle-même mais plutôt pour l'utiliser.

Le client ne peut ni gérer ou contrôler l'infrastructure cloud sous-jacente, y compris le réseau, les serveurs, les systèmes d'exploitation, le stockage, voire des capacités d'application individuels, à l'exception possible des paramètres de configuration d'applications spécifiques à l'utilisateur.Parmi les exemples les plus utilisés, on trouve des messageries comme Gmail, ou

des applications telles que la CRM (Customer Relationship Management), voire les réseaux sociaux tels que le Facebook.

1.3.3.2 "Platform as a Service (PaaS)"

Plateforme en tant que service "PaaS" (Platform as a Service) où le matériel, l'hébergement et le framework d'application sont dématérialisés. Le "PaaS" est un modèle qui permet la mise à disposition pour les clients (essentiellement des développeurs) d'environnement technique modulaire composé de tous les outils et les langages de développement nécessaires afin de soutenir la construction, le développement et le déploiement des applications et des services accessibles à distance. Le client ne peut pas gérer ou contrôler l'infrastructure cloud sous-jacente, y compris réseau, serveurs, systèmes d'exploitation, ou le stockage, mais a le contrôle sur les applications développées et déployées et leurs configurations. Les principaux acteurs sont Google (Google App Engine) [5] et Microsoft (Windows Azure) [6].

1.3.3.3 "Infrastructure as a Service (IaaS)"

Infrastructure en tant que service "IaaS" (Infrastructure as a Service) où seul le matériel (serveurs) est dématérialisé. Le IaaS s'adresse aux ingénieurs systèmes. Il désigne l'ensemble des composants d'une infrastructure technique déportée dans les nuages. Le IaaS se traduit principalement par des environnements d'exécution virtualisés. Le principle acteur de IaaS est actuellement Amazon EC2 [3].

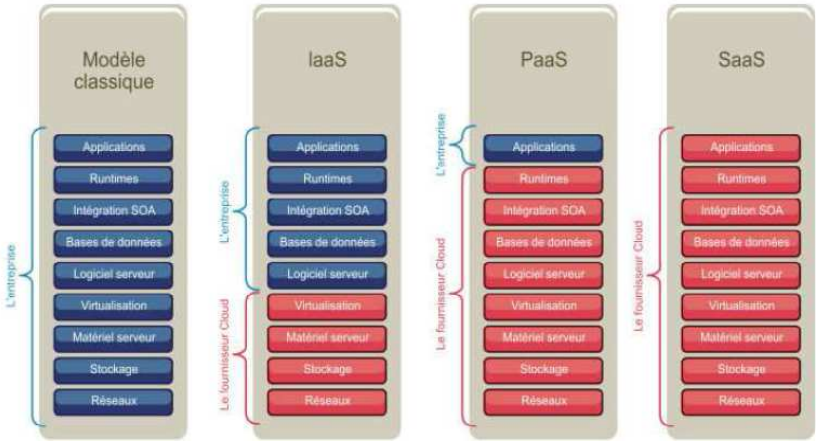

Figure 1.7: Service XaaS

1.3.4 Modèles de déploiement

L'informatique en nuage marque une nouvelle avancée vers l'infrastructure informatique élastique. Il existe quatre principaux modèles de déploiement des services en nuage : nuage privé, nuage communautaire, nuage public et nuage hybride [82].

1.3.4.1 Cloud privé

Nuage réservé à l'usage exclusif d'une seule organisation. Il peut être possédé, géré et opéré par cette organisation, un intervenant extérieur ou une combinaison des deux. Généralement il est situé dans les locaux de l'organisation.

1.3.4.2 Cloud communautaire

Nuage réservé à l'usage d'une communauté spécifique de consommateurs partageant des intérêts communs. Il peut être possédé, géré et opéré par un ou plusieurs organismes participant à la communauté, un intervenant extérieur ou une combinaison d'entre eux. Il est situé dans les locaux des organismes participant ou dans ceux d'un hébergeur externe.

1.3.4.3 Cloud public

Nuage externe à l'entreprise, accessible via Internet ou un réseau privé, géré par un opérateur externe propriétaire des infrastructures, avec des ressources totalement partagées entre tous ses clients.

1.3.4.4 Cloud hybride

Nuage résultat d'une conjonction de deux ou plusieurs Clouds différents (public, privé ou communautaire) amenés à "coopérer", à partager entre eux les applications et les données.

1.3.5 Acteurs

La Figure 1.8 présente un aperçu de l'architecture de référence de l'informatique en nuage selon NIST [73]. Il identifie les principaux acteurs et leurs activités. Il y a cinq acteurs :

- **Consommateur (Cloud Consumer)** : Une entité qui utilise un service fourni par un fournisseur,
- **Fournisseur (Cloud Provider)** : Une entité chargée de rendre un service à la disposition des parties intéressées,
- **Auditor (Cloud Auditor)** : Une entité qui peut procéder à une évaluation indépendante des services : la performance et la sécurité du nuage,
- **Courtier (Cloud Broker)** : Une entité qui gère l'usage, la performance et le provisionnement de services, et qui négocie la relation entre les fournisseurs et les consommateurs,
- **Carrier (Cloud Carrier)** : Un intermédiaire qui fournit la connectivité et le transport des services des fournisseurs aux consommateurs.

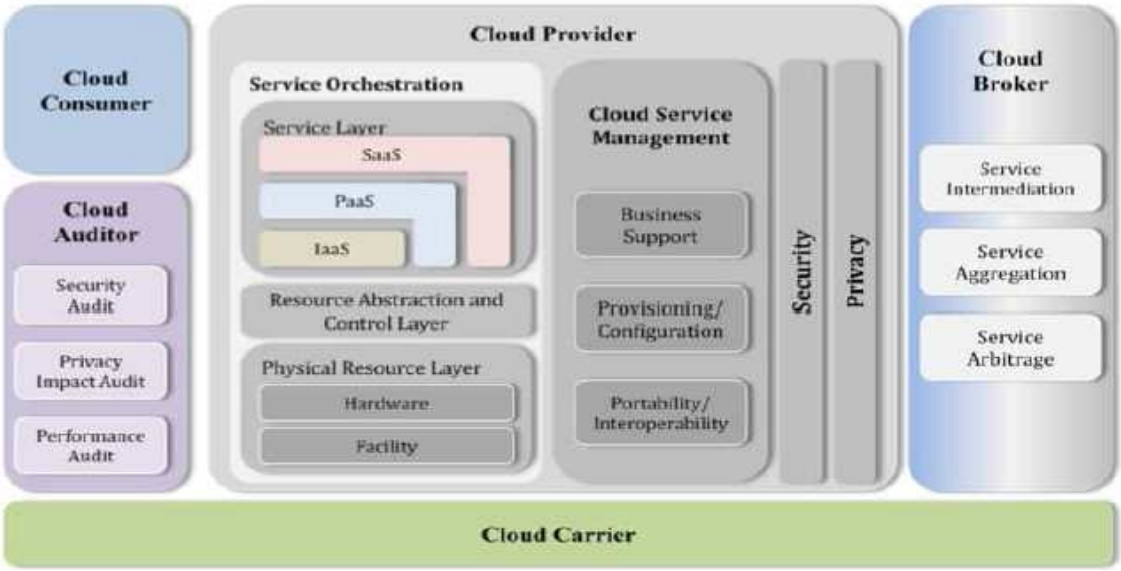

Figure 1.8: Référence NIST 2012 [73]

1.3.6 Virtualisation

Le concept de virtualisation [101, 50] a vu le jour pendant les années 1960 et permet de faire fonctionner nombreux systèmes d'exploitation et/ou plusieurs applications sur un seul serveur. Ces derniers doivent être indépendants et autonomes. La virtualisation apporte de très nombreux avantages tels que :

- L'utilisation optimale des ressources existantes et
- L'économie sur le matériel par mutualisation. Une infrastructure virtuelle permet de réduire les coûts informatiques tout en augmentant l'efficacité, le taux d'utilisation et la flexibilité des actifs existants.

Plus concrètement, la virtualisation est l'ingrédient indispensable du Cloud Computing. Elle offre l'abstraction, l'encapsulation, l'isolation et la consolidation des applications et des serveurs[46].

Une VM (voir Figure 1.9) est un conteneur de logiciels totalement isolé, capable d'exécuter ses propres systèmes d'exploitation et applications, à l'instar d'un ordinateur physique. Une machine virtuelle se comporte exactement comme un ordinateur physique. Elle contient un processeur, une mémoire RAM, un disque dur et une carte d'interface réseau virtuels (autrement dit, basés sur des logiciels) qui lui sont propres [101].

1.3.7 Les défis du "Cloud"

Le cloud computing est un domaine de recherche actif, qui évolue constamment vers de nouveaux types de services et de nouveaux modèles d'application.comme toute nouvelle technologie qui immerge, il a apporté avec lui des promesses; certains se sont réalisés alors que

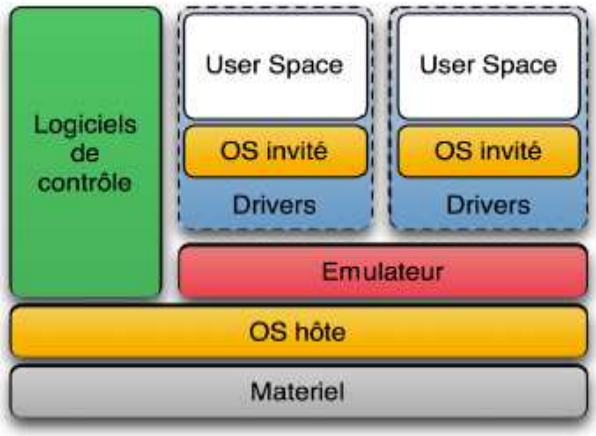

Figure 1.9: Machine virtuelle

d'autres sont confrontés à des obstacles auxquels il faudra encore faire face.Certains aspects clés qui ont un impact sur le taux d'adoption large des Cloud IaaS, en particulier dans le contexte des applications scientifiques, sont détaillés ci-dessous [30, 107]:

- **Gestion efficace des données**. Les efforts de recherche dans ce domaine visent à trouver des modèles de stockage de données appropriées qui répondent aux besoins des applications de données intensives et conformes avec les restrictions des environnements cloud.

- **Elasticité**. Le cloud computing est basé sur la capacité de fournir des ressources à la demande. De plus, l'ensemble des ressources louées doit évoluer à mesure que la charge de l'application augmente, et à diminuer automatiquement lorsque la charge de traitement du pic a été surmontée, afin de minimiser les coûts d'utilisation. Les services élastiques doivent fournir un compromis acceptable, en évitant les pénalités de performance, ainsi que l'utilisation inefficace des ressources. Dans ce contexte, le défi réside dans l'identification des mécanismes spécifiques pour permettre une corrélation automatique entre l'état du système et la dimension de location.

- **Sécurité**. Ceci est un sujet de recherche essentiel pour le cloud computing. Les problèmes de sécurité qui se posent dans les environnements de cloud sont de deux ordres.Tout d'abord, la sécurité des données impacts grandement les applications qui veulent compter sur les services de Cloud pour le stockage ou le traitement de données sensibles.Deuxièmement, les fournisseurs de Cloud sont également confrontés à des problèmes de sécurité, car ils ont besoin de contrebalancer les attaques malveillantes en vue de soutenir une disponibilité et performance de service constante.En conséquence, les fournisseurs de Cloud ont à mettre en œuvre des mécanismes de détection efficaces pour protéger et réparer leurs systèmes en cas d'attaques de sécurité.

- **Interopérabilité et la portabilité** . Face à la multiplication des plateformes de

Cloud, les clients pourrons être confrontés à la migration d'une application d'un Cloud vers un autre et à l'utilisation de plusieurs Clouds en même temps (Hybridation de Clouds) pour l'hébergement d'une application. Ainsi, la même application s'exécute dans deux environnements de Cloud différents appartenant à des fournisseurs distincts. Dans ces deux cas, la mise en place d'une API standardisée pour le problème d'interopérabilité entre les plateformes de Clouds est nécessaire.

- **Administration**. Fournisseurs et clients sont confrontés quotidiennement à plusieurs tâches d'administration Cloud. Cependant les plateformes de cloud doivent prendre en compte et faciliter les tâches d'administration de ces deux utilisateurs.

1.4 Cloud vs Grid

La question que nous nous posons : "A quel niveau se situe la différence entre le cloud et les grilles (si elle existe) ?". Autrement dit, le terme "Cloud Computing" n'est-il pas une autre appellation de "Grid Computing"? En effet et aprés consultation de la littérature [46, 26], beaucoup pensent que le cloud est une évolution naturelle de la grille. Certains considèrent les grilles et les Clouds comme une option alternative pour exercer le même travail de deux façons différentes. Cependant, il ya très peu de nuages sur lequel on peut construire, tester, ou exécuter des applications de calcul intensif.

1.4.1 Grids et Clouds comme alternatifs

"Grid" et "Cloud" sont des technologies qui ont été conçu pour fournir aux utilisateurs des ressources informatiques accessible en fonction de leurs besoins spécifiques.

La grille a été conçu avec une approche "bottom-up" (ascendente) [60, 24, 25, 17]. Le but est de partager des logiciels et matérielles par plusieurs organisations par le moyen de protocoles et politiques communs. L'idée est de déployer des services interopérables afin de permettre l'accès à des ressources physiques et de logiciels utilitaires disponibles. Les ressources grille sont gérées par leurs propriétaires. Les utilisateurs autorisés peuvent invoquer des grid services sans payer et sans garantie au niveau du service.

D'autre part, la technologie cloud a été conçu selon une approche "top-down"(descendente). Le cloud vise à fournir à ses utilisateurs une fonctionnalité spécifique de haut niveau, sans avoir accès à l'infrastructure physique, puisque ils interagissent seulement à travers la plateforme virtualisée.

D'un point de vue technologique, la virtualisation est exploitée dans le Cloud pour fournir un environnement isolé permettant ainsi aux clients de rencontrer les exigences demandées à travers une configuration et une sauvegarde faciles. De ce fait, le fournisseur est le propriétaire de la plateforme physique du Cloud, tandis que le client est propriétaire des ressources virtualisées dont il paye pour.

Les partisans du Cloud affirment que le Cloud est facile à utiliser, est évolutive [60], et donne toujours aux utilisateurs exactement ce qu'ils veulent. D'autre part, la grille est difficile à utiliser, ne donne pas des garanties de performance, est utiliseé par les communautés étroites de scientifiques pour résoudre des problèmes spécifiques, et ne supporte pas réellement l'interopérabilité [60].

Cependant, Les fans de la grille répondent que les utilisateurs de grille n'ont pas besoin d'une carte de crédit, que partout dans le monde il existe de nombreux exemples de projets réussis, et qu'un grand nombre de nœuds de calcul sont connectés à travers le net et exécutent des applications scientifiques à grande échelle.

1.4.2 Intégration Grid et Cloud

Pour comprendre pourquoi les grilles et les clouds devraient être intégrés, nous devons commencer par examiner ce que les utilisateurs veulent et ce que ces deux technologies peuvent fournir. Ensuite, nous pouvons essayer de comprendre comment le cloud et la grille peuvent se compléter mutuellement et pourquoi leur intégration est le but des activités de recherche intensives [99, 26]. Nous savons qu'un supercalculateur fonctionne plus vite qu'une ressource virtualisée. Par exemple, un benchmark LU sur EC2 (la plate-forme de cloud fourni par Amazon) est plus lent, et une certaine surcharge est ajoutée pour démarrer une VM [26]. D'autre part, la probabilité pour exécuter une application en temps fixe sur une ressource grille dépend de nombreux paramètres et ne peut pas être garantie.

afin de définir comment chacune des deux technologies peut compléter l'autre. Quelque problématiques ont été exposées à l'issue de ces recherches :

- 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.

à la lumière de ce qui précède, l'intégration des deux environnements est une question débattue [60]. Dans l'état de l'art, deux approches principales ont été proposées:

1.4.2.1 La Grille sur le Cloud

Une approche cloud IaaS (Infrastructure as a Service) est adoptée pour construire et gérer un système de grille flexible [32]. Ce faisant, le middleware de grille fonctionne sur une machine virtuelle. Ainsi, le principal inconvénient de cette approche est la performance. La virtualisation entraîne inévitablement des pertes de performance par rapport à l'utilisation directe de ressources physiques.

1.4.2.2 Le Cloud sur la Grille

Une autre approche consiste à utiliser l'infrastructure stable de la grille pour construire un environnement Cloud [113], dans ce cas, un ensemble de composants grille est utilisé pour administrer les machines virtuelles du Cloud. A titre d'exemple, nous pouvons citer l'utilisation du "Middleware Globus Toolkit" dans le "Cloud Nimbus[4]".

1.5 Conclusion

Dans ce chapitre nous avons présenté les deux paradigmes "Grid Computing" et "Cloud Computing". En effet, beaucoup pensent que le cloud est une évolution naturelle de celles-ci, car en regardant de près il est possible de constater que les contraintes qui ont limité l'adoption des grilles ont été levées pour former l'offre actuelle du cloud computing. D'un point de vue technologique nous n'identifions pas de réelle différence entre les plateformes de grille et de cloud. Cependant, la complexité, la gestion difficile et le coût des ressources matériels requis pour le "Grid Computing", l'ouverture du cloud aux utilisateurs de différents niveaux (pas toujours la communauté scientifique) et éventuellement son caractére commercial où le client paye pour l'utilisation de ressources acquises à la demande sont les potentielles différences que nous identifions.

Comme toujours, la vérité est au milieu. Certains utilisateurs préfèrent payer car ils ont besoin d'un service spécifique avec des exigences strictes et nécessitent une qualité de service garantie que le cloud peut fournir. Tandis que, beaucoup d'utilisateurs de la communauté scientifique recherchent une sorte d'architecture de supercalculateur pour résoudre des calculs intensifs qui traitent une énorme quantité de données, et ils ne se soucient pas d'obtenir un niveau de garantie de performance et la grille peut fournir ça.

Cependant, L'intégration pourrait simplifier la tâche de l'utilisateur pour sélectionner, configurer, et gérer les ressources selon les besoins de l'application. elle ajoute la flexibilité d'exploiter les ressources disponibles, mais l'intégration présente de sérieux problèmes pour la gestion globale du système, en raison de la complexité des architectures résultantes. La prédiction de la performance, le réglage des applications, et le benchmarking sont quelques-unes des activités pertinentes qui deviennent critiques et qui ne peut être effectuée en l'absence d'évaluation de performance des clouds.

[4] http://www.nimbus.org

CHAPTER 2

Monitoring dans le Cloud

2.1 Introduction

Le nombre de services du Cloud Computing a augmenté rapidement et fortement dans les dernières années, et ainsi a augmenté la complexité des infrastructures derrière ces services. Cependant, des activités de *"monitoring"* précises et à grains fins sont nécessaires pour fonctionner efficacement les plates-formes du Cloud Computing et gérer leurs compléxités croissantes et leurs besoins en sécurité [11].

En informatique, le "Monitoring" traduit littéralement par "la supervision", est une technique de suivi, qui permet de surveiller, d'analyser, de rapporter et d'alerter les fonctionnements anormaux des systèmes informatiques. Pour le "Cloud Computing", le monitoring est nécessaire afin d'administrer, gérer le mieux les ressources, surveiller la performance des applications et la disponibilité des services. Ainsi, informé par Kornaros et al [65] et Mansouri et al [76], nous définissons le monitoring comme:"*Un processus qui identifie pleinement et précisément la cause principale d'un événement en capturant les informations correctes au bon moment et au moindre coût afin de déterminer l'état du système et à la surface le status d'une manière opportune et significative".*

Dans la littérature, il y a un grand nombre de travaux proposant des études et des taxonomies sur le Cloud Computing en général [51, 46], sur les technologies de virtualisation [33, 88], et sur la sécurité dans les Clouds [82, 118, 63], Cependant, il y a peu d'études spécifiques sur les plates-formes, les techniques et les outils de monitoring des infrastructures, des services et des applications Cloud [12]. Dans ce qui suit, nous allons fournir une analyse sur le "Monitoring dans le Cloud". Plus précisément, nous discutons les principales motivations, concepts de base et définitions , et souligner les questions de recherche ouvertes et les orientations futures dans le domaine du monitoring dans le Cloud.

2.2 Motivations pour le monitoring dans le Cloud

Le monitoring dans le Cloud est une tâche d'une importance capitale pour les fournisseurs de services Cloud (appelés fournisseurs dans ce qui suit) et les consommateurs de services Cloud (appelés consommateurs dans ce qui suit). D'un côté, il est un outil clé pour le contrôle et la gestion des infrastructures matérielles et logicielles; de l'autre côté, il fournit des informations et *"Indicateurs clés de performance"* (KPIs) pour les plates-formes et les applications [12]. Le monitoring continue du Cloud et de ses SLAs (par exemple, en termes de disponibilité, délai, etc.) fournit aux fournisseurs et consommateurs des informations telles que la charge de travail générée par ce dernier, la performance et la qualité de service offerte par le Cloud, permettant également de mettre en oeuvre des mécanismes visant à prévenir ou récupérer des violations (à la fois pour les fournisseurs et les consommateurs). Le Cloud Computing implique de nombreuses activités pour lesquelles le monitoring est une tâche essentielle. Les plus importantes sont [11, 12]:

- **Planification des ressources et capacités** : L'une des tâches les plus difficiles pour les développeurs d'applications et services, avant l'adoption à grande échelle du Cloud Computing, a toujours été la planification des ressources et capacités (e.g. web services [83]). Le monitoring devient essentiel pour les fournisseurs de Service de Cloud pour prévoir et garder une trace de l'évolution de tous les paramètres intervenant dans le processus d'assurance de la qualité de service, afin de planifier correctement leur infrastructure et ressources en respectant les contrats SLAs [54].

- **Gestion des ressources et capacités** : La première étape pour gérer un système complexe comme le cloud consiste à disposer d'un système de monitoring capable de capturer avec précision son état [111]. Au fil des ans, la virtualisation est devenue un élément clé pour la mise en œuvre du Cloud Computing. Cacher la forte hétérogénéité des ressources de l'infrastructure physique, les technologies de virtualisation introduit un autre niveau de complexité pour le fournisseur d'infrastructure, qui doit gérer les ressources physiques et virtualisées.

- **Gestion du data center** : Les services Clouds sont fournit à travers de larges centres de données (infrastructure physique), dont la gestion est une activité très importante. La gestion du centre de données comprend deux tâches fondamentales : le monitoring, qui sert à garder trace des métriques du matériel et logicielles désirées, et l'analyse de données, qui traite ces métriques pour déduire les états du système ou l'application, le provisionnement des ressources, le dépannage, ou d'autres mesures de gestion.

- **La gestion du contrat "Service Level Agreement (SLA)"**: Les ressources et les services Cloud sont offerts par le fournisseur aux clients Cloud selon le "Service level Agreement (SLA)" basé sur les conditions d'utilisations en termes de performance,

qualité de service et les sanctions en cas de violation [42]. Le monitoring est indispensable pour garantir le niveau SLA des applications, et détecter la consommation des clients et les offres du fournisseur.

- **Billing (facturation)** : Afin d'offrir des *services mesurés* permettant au consommateur de payer proportionnellement à un paramètre mesuré, le *monitoring* est fondamental, non seulement pour le fournisseur , mais aussi pour le consommateur, afin de vérifier son utilisation effective des services Cloud, et également de comparer les prix sur les différents prestataires.

- **Dépannage** : L'infrastructure complexe d'un cloud représente un grand défi pour le dépannage (par exemple, l'analyse d'une cause principale), comme la cause du problème doit être recherchée dans plusieurs composants possibles (par exemple le réseau, la machine, etc) et chacun d'entre eux est composé de plusieurs couches. Une plate-forme de monitoring complète et fiable est donc nécessaire pour les fournisseurs pour comprendre où situer le problème dans leur complexe infrastructure et pour les consommateurs pour comprendre si le problème de performance ou de défaillance est causé par le Fournisseur lui-même ou par d'autres causes.

- **Gestion de performance** : Etant donné que la maintenance de l'infrastructure physique est déléguée aux fournisseurs, le modèle Cloud Computing est attrayant pour la plupart des consommateurs (principalement les moyennes entreprises et les groupes de recherche). Cependant, en dépit de l'attention portée par les fournisseurs, certains noeuds du cloud peuvent atteindre des ordres de performance de magnitude plus que d'autres noeuds [17]. Le monitoring est nécessaire car il peut considérablement améliorer la performance des applications réelles et affecter la planification des activités.

- **Gestion de la sécurité** : La sécurité est un aspect très importent dans le Cloud. Elle est considérée comme l'un des obstacles les plus importants à la propagation du Cloud Computing. Spécialement, pour certains types d'applications.

2.3 Concepts de base

Dans cette section, nous présentons un certain nombre de concepts [12] à la base du monitoring dans le Cloud et qui sont utilisés pour définir le contexte des sections suivantes. La figure 2.1 présente ces concepts dans une taxonomie proposée pour les principaux aspects du monitoring dans le Cloud présenté dans ce chapitre.

2.3.1 Couches (layers)

Selon les travaux du Cloud Security Alliance, un cloud peut être modélisé en sept couches [102, 103, 23] ("Facility", "Network", "Hardware", "OS", "Middleware", "Application", "User"): Ces couches peuvent être contrôlées soit par le fournisseur du Cloud soit par le client. Dans

le context du monitoring dans le Cloud, ces couches peuvent être considérées comme l'endroit où placer les "probes" (les agents) du système de monitoring. En effet, la couche à laquelle les agents sont situées a des conséquences directes sur les phénomènes qui peuvent être surveillés et observés.Cependant, pour observer ou superviser un phénomène dans Le Cloud, on doit collecter toutes les informations relatives à ce phénomène dans les différentes couches du Cloud et qui ont des conséquences directes sur ce phénomène [11].

2.3.2 Niveaux d'abstraction

Le monitoring dans le Cloud est important des deux côtés : Côté client et côté fournisseur. Ainsi, on peut distinguer deux niveaux d'abstraction : le monitoring du niveau haut "Hight-level monitoring" et le monitoring du niveau bas "Low-level Monitoring", et les deux sont nécessaires [12, 40].

- **Le monitoring d'infrastructure "Low-level Monitoring" :** La collecte d'informations concerne toute l'infrastructure physique du Cloud et ces informations sont intéressantes du point de vue fournisseur des services Cloud et ne doivent en aucun cas être visibles pour le client.Plus précisément [102], pour le monitoring du niveau bas les utilitaires spécifiques collectent les informations à la couche matérielle (par exemple, en termes de CPU, la mémoire, la température, la tension, la charge de travail, etc), à la couche système d'exploitation et à la couche de middleware (par exemple, bug et les vulnérabilités des logiciels), à la couche réseau (par exemple, sur la sécurité de l'ensemble de l'infrastructure à travers le pare-feu, IDS et IPS), et à la couche d'installation (par exemple sur la sécurité physique des installations concernées par le monitoring des données sur les salles de Data Center à l'aide de la vidéo-surveillance et des systèmes d'authentification).

- **Le monitoring des services applicatifs "Hight-level monitoring" :** Il est lié à l'information sur l'état de la plate-forme virtuelle. Cette information est collectée au niveau des couches middleware, applications et utilisateurs par les fournisseurs ou les consommateurs à travers des plates-formes et services exploités par eux-mêmes ou par des tiers. Dans le cas du modèle SaaS, les informations de monitoring de niveau haut ont généralement plus d'intérêt pour le consommateur que pour le fournisseur.

2.3.3 Tests et métriques

Les tests de monitoring peuvent être divisés en deux catégories principales [80, 12]: Basés sur calcul et basés sur le réseau. Les tests à base de Calcul sont liées aux activités de monitoring visant à acquérir des connaissances et déduire l'état des plates-formes réelles ou virtualisés exécutant des applications de Cloud.

2.3.3.1 basé calcul (Computation-based)

Les tests sont liés aux métriques suivantes: Débit du serveur, défini comme le nombre de requête (par exemple la récupération de la page Web) par seconde;Vitesse du CPU;Temps CPU par exécution, défini comme le temps CPU d'une seule exécution; L'utilisation du CPU, définie comme l'occupation CPU de chaque machine virtuelle (utile pour surveiller l'utilisation simultanée d'une seule machine par plusieurs machines virtuelles); l'échanges de page mémoire par seconde, défini comme le nombre de pages mémoire par seconde échangés à travers des E/S;l'échanges de page mémoire par exécution, définis comme le nombre de pages de mémoire utilisés lors d'une exécution; débit disque/mémoire; débit/délai de passage des messages entre les processus; durée de tâches prédéfinies spécifiques; temps de réponse; temps de démarrage d'une machine virtuelle; temps d'acquisition /libération d'une machine virtuelle; temps d'exécution/accès; up-time.

Chacun d'entre eux peut être évalué en termes d'indicateurs statistiques classiques (moyenne, médiane, etc.) ainsi que sur le plan de la caractérisation temporelle et donc stabilité, variabilité et prévisibilité [12].

2.3.3.2 basé réseaux (network-based)

Les tests sont liés au monitoring des métriques de la couche réseau. Cet ensemble inclut le temps d'aller-retour (RTT), la gigue, le débit, la perte de paquets/données, la bande passante disponible, la capacité, volume de trafic, etc [12, 104]. En utilisant ces métriques, plusieurs études expérimentales de la littérature comparent l'hébergement Web classique à celui du Cloud [12, 13].

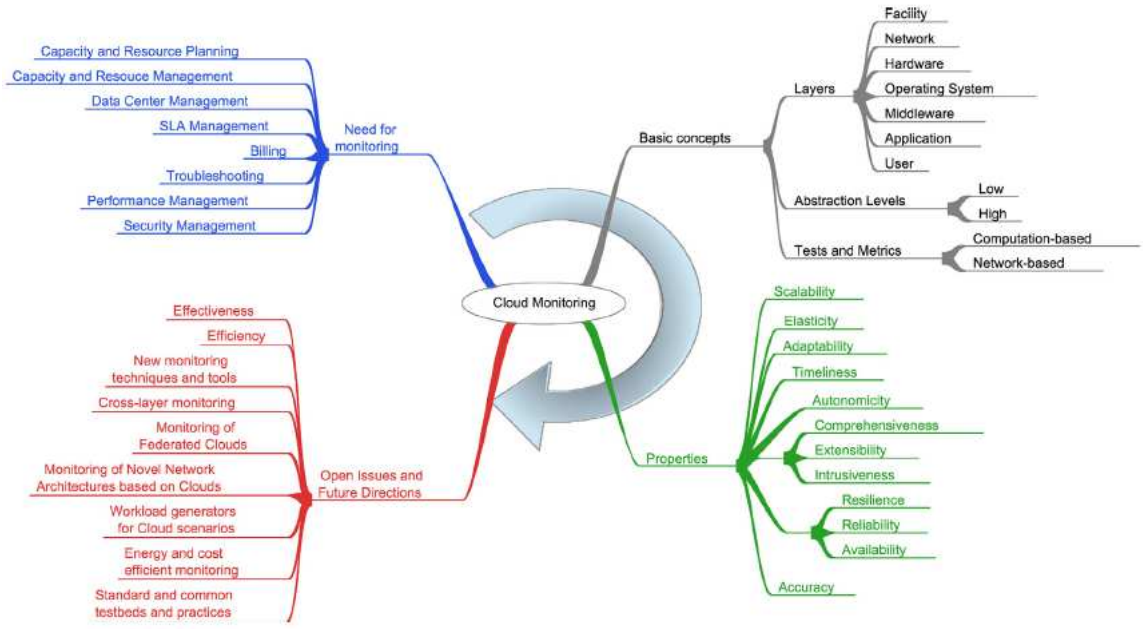

Figure 2.1: Cloud monitoring: motivations, properties, basic concepts and open issues [12]

2.3.4 Le monitoring dans “Grid” vs ”Cloud”

Des Similitudes et des chevauchements de propriétés entre le Cloud Computing et les paradigmes distribués précédents ont conduit à une discussion approfondie sur la définition du Cloud Computing et ses caractéristiques particulières [12, 82, 43]: Ici nous considérons les différences du point de vue monitoring [12, 43].

En comparaison avec le “Grid Computing”, le monitoring dans le Cloud est plus complexe en raison des différences dans le modèle de confiance et la vue sur la ressource/services présentée à l'utilisateur [46]. En effet, le principal objectif d'une Grille est le partage des ressources entre les multiples organisations,impliquant de simples critères de comptabilité et une abstraction de ressources limitée, ce qui crée une relation simple entre les paramètres de montoring et l'état de la ressource physique. D'autre part, pour le Cloud, la présence de multiples couches et les paradigmes de services conduit à une forte abstraction de ressources, résultant en une relation plus opaque entre les couches ou Service observables spécifique et les ressources sous-jacentes.

Cet écart d'objectifs et de transparence doit être rempli lors de l'adoption d'un système de monitoring d'un Cloud provenant du terrain Grid Computing. Enfin, le paradigme service “à la demande” pose des défis supplémentaires.

La plupart des méthodes de monitoring et plates-formes proposées pour le cas de la grille [12] ont été personnalisées pour les systèmes de Cloud. Cependant, ces aspects doivent être pris en compte lorsque on envisage Ganglia [78], Nagios [2] MonaLisa [89], R-GMA [38] et GridICE et des systèmes similaires pour surveiller un Cloud [12, 43] .

2.4 Propriétés

Pour fonctionner correctement, un système de monitoring distribué est tenu d'avoir plusieurs *propriétés*[12, 11]:

- **évolutivité “Scalability”** : Un système de monitoring est *“évolutif”* s'il peut faire face à un grand nombre de probes(agents) [12][35]. Cette propriété est très importante dans les scénarios du Cloud Computing en raison du grand nombre de paramètres à surveiller sur un grand nombre de ressources. Cette importance est amplifiée par l'adoption des technologies de virtualisation, qui permettent d'allouer beaucoup de ressources virtuelles sur une seule ressource physique. Les mesures nécessaires pour obtenir une vue d'ensemble sur l'état du Cloud conduit à la production d'un très grand nombre de volume de données provenant de multiples emplacements répartis. Par conséquent, un système de monitoring évolutif devrait être en mesure de collecter efficacement, transférer et analyser un tel volume de données sans nuire à l'exploitation normale de Cloud.

  Dans la littérature un tel problème a été principalement abordé en proposant des architectures dans lesquelles les données et les événements de monitoring sont propagés vers

l'application de contrôle après leur agrégation et filtrage, afin de réduire leur volume de données de monitoring [11].

- **Elasticité "Elasticity" :** Un système de monitoring est *"élastique"* si il peut s'adapter aux changements dynamiques des entités surveillées, de sorte que les ressources virtuelles créées et détruites par l'expansion et la contraction sont surveillées correctement [11].

  Le principal défi dans la fourniture de l'élasticité est liée au fait que c'est une nouvelle propriété fondamentale introduite par le monitoring dans le Cloud et pas considérée auparavant comme une exigence pour le monitoring des systèmes distribués génériques [11].

- **Adaptabilité "Adaptability" :** Un système de monitoring est *"adaptable"* s'il peut s'adapter à des charges variables de calcul et de réseau sans être invasif (c'est à dire empêcher d'autres activités) [12]. Cependant, la capacité de régler les activités de surveillance selon des politiques appropriées est d'une importance significative pour atteindre les objectifs de gestion du Cloud.

- **Opportunité "Timeliness" :** Un système de monitoring est "opportun" si les évènements détectés sont disponibles aux moments de leur utilisation prévue.

- **Autonomicité "Autonomicity" :** Un système de monitoring est *"autonome"* s'il est en mesure de gérer lui-même ses ressources distribuées par réaction automatique aux changements imprévisibles, tout en cachant la complexité intrinsèque aux fournisseurs et consommateurs [12, 86].

- **Compréhensivité,Extensibilité et Intrusion** : Un système de monitoring est *"compréhensif"'* si il prend en charge différents types de ressources (physiques et virtuelles), plusieurs types de données de monitoring, et plusieurs locataires [11, 54]; il est *"extensible"* si un tel support peut facilement être étendu (par exemple, grâce à des plug-ins ou modules fonctionnels); il est *"intrusif"'* si son adoption nécessite une modification importante dans le Cloud [12, 62].

- **Résilience, Fiabilité et Disponibilité :** Un système de monitoring est *"résilient"* lorsque la persistance à la prestation de services peut légitimement être digne de confiance face aux changements, ce qui signifie essentiellement résister à un certain nombre de défaillances de composants, tout en continuant à fonctionner normalement; il est *"fiable"* quand il peut accomplir une fonction requise dans des conditions données pour une période de temps déterminée; il est *"disponible"* si il fournit des services selon la conception du système lorsque les utilisateurs les demandent [12].

- **Précision "Accuracy"** : Un système de monitoring est *"précis"* lorsque les mesures qu'il fournit sont exactes, c'est à dire qu'ils sont aussi proches que possible de la valeur réelle à mesurer [11].Il existe deux principaux problèmes liés à la précision des systèmes

de monitoring du Cloud: la charge de travail utilisée pour effectuer les mesures, et l'impact des systèmes de virtualisation qui ajoutent des couches supplémentaires entre les applications et les ressources matérielles.

2.5 Les plates-formes de monitoring à usage général

Avant l'arrivée du Cloud Computing, un certain nombre d'outils ont été déjà disponibles dans le but de surveiller divers ressources de l'infrastructure telles que les réseaux et les nœuds de calcul. Certains se spécialisent dans des domaines particuliers tels que les clusters HPC et les grilles. Plusieurs de ces outils continuent d'être développés, et pourrait être adoptés dans les Clouds pour un monitoring à différents niveaux d'abstraction. Dans cette section, nous discutons les caractéristiques des outils de monitoring de l'infrastructure et de l'application à usage général [115].

Les outils de *monitoring de l'infrastructure à usage général* utilisent généralement un modèle client-serveur en installant un agent dans chaque système à surveiller. Les agents de monitoring mesurent les valeurs des métriques des composants surveillés et les envoient au serveur de surveillance. Le serveur stocke les métriques collectées dans une base de données, les analyse et envoie des alertes. Il peut générer des graphiques, des rapports sur les tendances et des rapports SLA sur la base des paramètres surveillés extraites de la base de données.

La plupart des systèmes de monitoring utilisent l'e-mail et le SMS en tant que mécanismes d'alerte. Par exemple, Nagios [21], Opsview [66], Zabbix [91] et Open NMS [93]. Certains outils de monitoring, tels que Cacti [72], utilisent des alertes sonores. D'autres, comme collectd [39] et Ganglia [78], n'ont pas de mécanismes d'alerte.

Peu d'outils sont disponibles pour le *monitoring des applications à usage général* [115]. La technique de monitoring d'application la plus appropriée dépend de la nature de l'application. Par exemple,"Kiwi Application Monitor" [48] surveille les applications centralisées en cours d'exécution dans une seule machine en contrôlant les processus du système d'exploitation. D'autres outils, tels que "Remoting-Open Services Gateway initiative (ROSGi) Development Tool (RDT)" [98] et "Distributed Application Monitoring System" (DAMS) [61], qui visent à surveiller les applications réparties impliquant les communications réseau.

2.6 Les plates-formes de monitoring spécifique Cloud

L'avènement du Cloud Computing a donné naissance au développement des outils de monitoring spécifiques au Cloud. Actuellement, les fournisseurs de Cloud offrent divers services en utilisant des techniques de gestion et logiciels propre à eux. En outre, plusieurs de ces fournisseurs utilisent des outils de monitoring qui dépendent d'eux et qui complètent leurs offres. Par exemple, Amazon Cloud Watch [14] surveille Amazon Web Services (AWS) tel que Amazon EC2, les instances Amazon RDS DB et des applications s'éxécutant sur AWS. Azure Watch [8], d'autre part, surveille des ressources basées sur Azure, les instances Win-

dows Azure, les bases de données SQL Azure, sites Web, des applications Web, et le stockage Windows Azure. Ces deux outils permettent à l'utilisateur de définir les métriques surveilées. En revanche, les outils de surveillance indépendents du fournisseur qui existent peuventt être utilisés pour surveiller plusieurs plates-formes de Cloud. Par example, Nimsoft [79] peut surveiller Amazon EC2, S3 Web Services, Rackspace Cloud, Microsoft Azure, Google App Engine, Google Apps et Salesforce CRM;Monitis [9] peut surveiller Amazon EC2 / AWS et Rackspace Cloud; CloudKick [57] peut surveiller Amazon EC2, GoGrid et Rackspace Cloud. Enfin, certains outils de surveillance, tels Private Cloud Monitoring Systems (PCMONS) [31] surveillent uniquement les Clouds privés.

La plupart des outils de monitoring Cloud, y compris Amazon Cloud Watch, Azure Watch, Nimsoft et Monitis, sont capables de surveiller les niveaux infrastructure et application. Cependant, certains outils sont en mesure de surveiller seulement le niveau infrastructure (CloudKick, PCMONS) ou le niveau application (Boundary application monitor, mOSAIC [96] and Cloud Application SLA Violation Detection (CASViD) [42].

2.7 Défis et orientations de la recherche

Le Cloud Computing est une technologie prometteuse qui a actuellement gagner une large acceptation dans l'industrie. La gestion efficace de cette technologie reste un défi et sera un domaine de recherche important pour les années à venir [43]. Cependant, elle doit surmonter plusieurs défis dont [12, 11, 43]:

- **Efficacité :** Le défi réside dans la possibilité d'avoir une vision claire du Cloud et d'identifier les causes à l'origine des phénomènes observés. Pour atteindre cet objectif, des améliorations sont nécessaires en termes de:
  1. Algorithmes et techniques personnalisés qui fournissent des résumés efficaces, le filtrage et la corrélation des informations provenant de différents agents;
  2. des techniques d'analyse de la cause principale capables de tirer les causes des phénomènes observés, de repérer le fil droit dans le tissu complexe de l'infrastructure Cloud; et
  3. Des mesures précises dans un environnement dominé par les ressources virtualisées.
- **Capacité :** Des techniques et des algorithmes efficaces sont nécessaires pour pouvoir gérer rapidement et continuellement la quantité de données de surveillance utilisée pour avoir une vue compréhensive du Cloud. Le système de surveillance doit donc être capable de faire plusieurs opérations sur les données (collecte, filtrage, agrégation, corrélation, découpage, stockage) en respectant les exigences strictes en terme de temps, la puissance de calcul, et surcoût des communications.

Outre les améliorations rapportées ci-dessus, dans le prochain avenir, différentes directions de recherche sont possibles pour le monitoring Cloud:

1. **Nouvelles techniques et outils de monitoring :** Des techniques de surveillance efficaces devraient être en mesure de fournir, d'une part, des mesures à grains très fins, et, d'autre part, une vision synthétique du Cloud, impliquant toutes les variables qui affectent la qualité de service et d'autres exigences. En même temps, les techniques ne devraient pas ajouter une charge de performance au système (pensez, par exemple, au mobile Cloud). Enfin, elles doivent être intégrés à une méthode de contrôle qui gère les performances du système d'entreprise.

2. **Cross-layer monitoring** : La structure complexe du Cloud est composée de plusieurs couches pour permettre la séparation fonctionnelle, la modularité et donc la gestion. Cependant, une telle superposition forte pose plusieurs limites sur le système de surveillance, en termes de types d'analyses et actions conséquentes qui peuvent être effectuées. Cependant, les consommateurs et les fournisseurs font leurs décisions en se fondant sur un horizon limité.

3. **Monitoring des Clouds fédérés :** La collaboration standardisée à travers de multiples infrastructures de cloud est appelée fédération de ressources. Cependant, un tel processus de normalisation est encore à un stade précoce [1]. La forte hétérogénéité entre les différentes infrastructures de surveillance conteste la possibilité d'obtenir une solution de monitoring complète pour les Clouds fédérés, et cela n'a pas encore été traités correctement dans la littérature.

4. **Energie et coût effectif** : Les activités de monitoring peuvent être très exigeantes en termes de ressources informatiques et de communication, et donc en termes d'énergie et de coût. Un autre défi important pour les systèmes de monitoring Cloud des prochaines générations est celle d'effectuer des activités de surveillance répondant à leurs besoins de base (exactitude, compréhensivité, la fiabilité, etc), mais en minimisant la consommation d'énergie et les coûts liés

2.8 Conclusion

Le Cloud Computing est une technologie prometteuse qui gagne actuellement une large acceptation dans l'industrie. La *gestion efficace* de cette technologie reste encore un défi et sera un important domaine de recherche pour les prochaines années. Cependant, le monitoring dans le Cloud est un domaine qui n'a pas encore été pleinement réalisé.

Dans ce chapitre, nous avons fourni une analyse dans le domaine du monitoring dans le Cloud. La Figure 2.1 a montré une taxonomie contenant un aperçu rapide des principaux aspects que nous avons examiné dans le présent chapitre. De manière plus détaillée, nous avons discuté les principales activités dans un environnement cloud qui ont un besoin effectif d'un monitoring. Pour contextualiser et étudier le monitoring dans le Cloud, nous avons fourni les principales propriétés d'un système de monitoring dans le Cloud, les concepts

de base pour la construction d'un tel système ainsi que les contributions connexes prévues dans la littérature. Nous avons présenté quelques solutions de monitoring en les classant en des catégories à usage général et spécifiques Cloud afin de mieux comprendre les outils et analyser historiquement leur évolution. Plusieurs outils de monitoring à usage général sont actuellement adaptés au monitoring dans les Clouds, et ce processus d'adaptation continue dans l'avenir. Plus précisément, La majorité des approches et plates-formes de monitoring proposées pour le scénario Grille ont été personnalisées pour les systèmes Cloud. Zanikolas et al. [11, 115] ont fait une étude dans le domaine de recherche du monitoring dans la grille en introduisant les concepts , les exigences et les phases impliqués ainsi que les activités de normalisation liées. Par ailleurs, la majorités des issues et défis restent à accomplir, notamment en ce qui concerne l'introduction de nouvelles techniques afin de rencontrer les exigences demandées.

CHAPTER 3

Le monitoring applicatif et la modélisation de la performance d'applications dans le Cloud Computing

3.1 Introduction

Le monitoring est un outil clé pour gérer et administrer les ressources et les applications de cloud computing; d'autre part, il fournit des informations et *des indicateurs clés de performance* pour les plates-formes et les applications [108].

Dans le monitoring des applications de cloud computing, nous pouvons distinguer deux niveaux distincts: Le monitoring niveau infrastructure et le monitoring niveau application. Le monitoring des ressources au niveau de l'infrastructure [36] vise à la mesure et la déclaration des paramètres du système liée aux services d'infrastructure réelle ou virtuelle offerte à l'utilisateur (exemple CPU, RAM, ou des paramètres de stockage de données). Au niveau de l'application, la nature des paramètres surveillés, et la façon dont leur valeurs doivent être récupérées dépendent du comportement de l'application [96]. Cependant, surveiller les applications de cloud et plus précisment les applications scientifiques qui nécessitent de grandes capacités de calcul n'est pas une tâche triviale, nécessitant à la fois pour les applications et la plate-forme, d'être constamment surveillées, en capturant des informations à différents niveaux (monitoring multi-niveaux) et une granularité de temps. Plus concrètement, le domaine du monitoring applicatif a besoin de plus de recherches et plus précisément les applications scientifiques , tels que des outils et des modèles pour surveiller les applications de cloud computing, en particulier le *monitoring de la performance des applications* et services cloud pour assurer qu'ils répondent aux besoins des utilisateurs.

D'autre part, le monitoring de l'application est un défi difficile en raison des métriques surveillées de la plate-forme ou de la couche d'infrastructure, qui ne peuvent pas être facilement mises en correspondance avec les métriques nécessaires à la couche application [42].

Il existe des travaux de recherche qui ont des points de similitude avec les concepts de notre étude, dans cette section notre objectif est de positionner le présent travail par rapport aux autres. Les différents articles de notre bibliographie, nous fait constater que deux

catégories de recherches peuvent nous intéresser. La première concerne les approches ayant employé *la modélisation de la performance* de l'application tel que les modèles de prédictions quantitatives, comme *l'analyse des séries chronologiques*. Les modèles de performance fournissent la capacité de prédire la performance de l'application pour un ensemble donné de ressources matérielles et sont utilisés pour *la planification des capacités et la gestion des ressources*. L'autre catégorie concerne les travaux qui se sont penchés sur le monitoring applicatif d'une autre façon (tel que l'aspect multi-niveaux). Nous procèderons donc par présenter ces travaux dans l'ordre dans lequel nous venons de les citer tout en nous efforçons d'être, sans le prétendre, les plus exhaustifs possible.

Ce chapitre est donc réparti comme suit, nous commençons par clarifier le concept du monitoring applicatif basé sur un modèle de performance et enchaînerons par une approche en relation avec l'objet de notre étude. Nous présentons également, quelques exemples illustrant l'implémentation de cette fonctionnalité dans les environnements Cloud. Néanmoins, nous avons réservé la dernière partie aux travaux ayant étudié des problématiques liées au monitoring applicatif autre que la modélisation de la performance dans le Cloud Computing.

3.2 Monitoring des applications Cloud basé sur un modèle de performance

Prédire précisément la performance de l'application pour un ensemble donné de ressources aide dans la gestion efficace de l'infrastructure informatique de haute performance (exemple : application scientifique) en permettant une allocation de ressources optimisée. L'importance de modéliser correctement le comportement de la performance des applications est prononcé dans les centres de données virtualisés, où plusieurs applications partagent les mêmes ressources physiques (ressources de l'hôte). Une mauvaise allocation des ressources à une VM spécifique a le potentiel d'affecter non seulement la machine virtuelle cible, mais d'autre ou toute les machines virtuelles fonctionnant sur le même hôte.

3.2.1 Généralités et définitions

Un modèle de performance rapporte quantitativement des métriques de qualité de service (QoS) au niveau de l'application (par exemple, le temps de réponse d'une application cliente) avec une ressource donnée [53]. Les modèles de performance fournissent la capacité de prédire les performances des applications pour un ensemble donné de ressources matérielles et sont utilisés pour la planification des capacités et la gestion des ressources [67]. Pour cela, des paramètres des ressources, des critères de qualité de service (QOS) de haut niveau sont prédits via un modèle de performance[114].

Les modèles de performance traditionnels assurent la disponibilité du matériel dédié à l'application. Avec la croissance du déploiement de l'applications sur un matériel virtualisé, les ressources matérielles sont de plus en plus partagées entre plusieurs machines virtuelles.

Construire des modèles de performance pour des applications dans des environnements virtualisés exige d'identifier un ensemble clé de paramètres indépendants de l'architecture de virtualisation et qui influencent la performance de l'application pour un ensemble diversifié et représentatif d'applications. Parmi les définitions d'un modèle de performance on trouve celle de Ghanbari [52]:

Definition 3.1. *: Un modèle de performance est une abstraction qui prend la spécification de couche physique d'une application et sa charge de travail et la mappe à divers attributs de qualité (par exemple, temps de réponse, débit).*

Plusieurs défis doivent être abordés dans l'élaboration d'une telle solution [67]:

1. L'identification des paramètres d'un système virtualisé qui peuvent affecter la performance d'une application virtualisée au bon niveau d'abstraction et qui sont suffisantes pour prédire le comportement de l'application avec une grande précision.
2. L'identification des techniques soit pour l'observation ou le contrôle de ces paramètres dans un environnement virtualisé.
3. Construction d'un modèle de performance pour une application virtualisée basé sur les paramètres ci-dessus.

3.2.2 Sélection des paramètres

Identifier l'ensemble idéal des paramètres influençant la performance d'une VM necessite de s'adresser aux préoccupations suivantes [67]: Tout d'abord, les paramètres doivent mapper directement ou indirectement, et refléter un comportement d'utilisation des ressources connue des processus, et ils doivent être faciles à contrôler et/ou observer. En second lieu, le comportement de l'application dans une seule VM peut dépendre de la nature de l'activité dans les autres machines virtuelles et l'influence de cette concurrence doit être pris en compte et caractérisée avec précision. Par exemple, une application intensive en E/S s'exécutant dans une VM peut affecter les opérations d'E/S d'une autre application fonctionnant sur une autre VM. Le troisième défi est d'identifier l'ensemble minimal des paramètres du modèle qui capture efficacement les performances des applications de haute précision.

Tout modèle de performance a besoin d'un bon système de monitoring qui collecte différentes métriques décrivant l'état actuel du système et de l'application, et à une granularité approprié (par exemple par seconde, par minute). Ghanbari et al [53] propose une liste de ***métriques de performance***. Le tableau 3.1 résume les métriques utilisées:

Table 3.1: Métrique pertinentes [53]

Catégorie	Métriques	Détails
	CPU Utilization	Utilisation moyenne du CPU en pourcentage par codes d'application

Hardware

Table 3.1: Métrique pertinentes [53]

Catégorie	**Métriques**	**Détails**
	CPU Max	Utilisation pic du CPU en pourcentage pour le total
	CPU Total	Utilisation moyenne du CPU en pourcentage pour le total
	RAM used	Percentage of used space in physical memory and paging space
	RAM libre	Pourcentage de la mémoire libre
	RAM total	La mémoire totale
	DISK used	Pourcentage d'espace disque utilisé
	Network interface access	Accès à l'interface réseaux
OS Process	Cpu-time	Temps CPU
	Real-memory (resident set) size	Mémoire occupée par le processus (quantité de mémoire physique utilisée par le processus)
Application server	Total threads count	Nombre de threads total
	Active threads count	Nombre de threads actifs
	Used memory	Mémoire utilisée
	Processed requests, pending requests	requêtes traités, requêtes en attente
	Dropped requests	Requêtes abandonnées
	Response time	Temps de réponse
Message Queue	Average number of jobs in the queue	Nombre moyen de jobs dans la file d'attente
	Average jobs queing time	Moyenne de temps d'attente de job.

3.2.3 Les approches de modélisation de performance des applications

La prédiction est le processus de faire des déclarations au sujet d'événements dont les résultats réels n'ont pas encore été observés. Plusieurs méthodes sont proposées dans la littérature pour la prévention ou la prédiction des performances. Etant donné l'éventail des comportements des applications virtualisées, l'identification des techniques qui peuvent adéquatement les modéliser est un challenge [114]. Parmi les approches utilisées, nous pouvons citer : la théorie des files d'attente, la théorie de contrôle, l'apprentissage par renforcement et l'analyse des times series [114].

3.2.3.1 Théorie des files d'attente

La théorie des files d'attente (voir figure 3.1) est une théorie mathématique relevant du domaine des probabilités [84]. Un modèle analytique basé sur la théorie des files d'attente peut prédire le comportement du système. La méthode d'analyse par valeur moyenne (Mean Value Analysis - MVA) est la plus utilisée. Une telle méthode permet d'estimer la latence, la

disponibilité et le temps de réponse pour une charge de travail. Dans le contexte de Cloud, un service ou une application peuvent être représentés par une ou plusieurs files d'attente. Chaque modèle de file d'attente est une ressource.

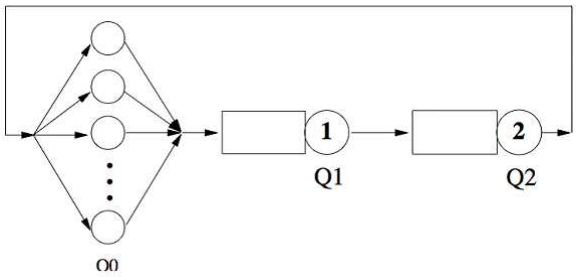

Figure 3.1: Théorie des files d'attente [84]

Plusieurs travaux de recherche ont été présentés dans le domaine de modélisation de la performance d'application dans les environements virtualisés ainsi que le Cloud Computing basés sur le modèle des *files d'attente*, nous allons citer les différents travaux que nous avons pu rassembler dans la phase recherche bibliographique:

Doyle et al [41] ont étudié les modèles internes pour estimer le temps de réponse du service sous différentes charges et l'allocation des ressources. L'approche utilise des modèles de files d'attente pour la prédiction du temps de service et du serveur and is application-aware since it employs performance models based on knowledge of application load.

Bennani et al [22] ont également enquêté sur des réseaux de files d'attente ouvert multi classe d'une application-aware pour prédire le temps de réponse et le débit de la charge de travail batch et en ligne.

Urgaonkar et al [109] ont utilisé un réseau de files d'attente G/G/1 (un par serveur). Un modèle de file d'attente est utilisé pour la planification de la capacité pour chaque niveau de l'application.

3.2.3.2 Théorie de contrôle

La théorie de contrôle (voir figure 3.2) est une branche de l'ingénierie et le mathématique, qui traite du comportement des systèmes dynamiques [22]. L'objectif du contôleur est de maintenir la performance de l'application à un niveau souhaité en ajustant l'entrée des ressources. La théorie de contrôle a montré des résultats intéressants sur les plates-formes de cloud computing [74].

Plusieurs travaux de recherche ont été présentés dans le domaine de modélisation de la performance d'application en utilisant la *Théorie de contrôle*. On peut citer:

Padala et al [92] ont fourni un contrôleur adaptatif qui utilise un ARMA de second ordre (Auto régressive et la moyenne mobile). Le contrôleur ajuste l'utilisation CPU et les E/S disque. Les auteurs dans cette solution utilise le CPU et les E/S disque niveau bas et le temps de réponse moyen comme mesure de haute performance.

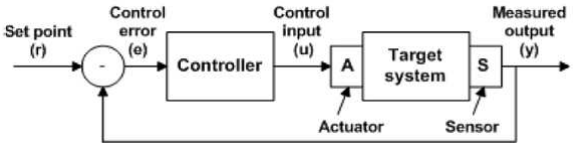

Figure 3.2: Théorie de contrôle [22]

Bodik P et al [95] ont utilisé une machine d'apprentissage statistique qui rend le contrôle automatique pratique pour les centres de données Internet.

Lim et al [56] décrit une technique de seuil via un contrôleur intégral proportionnel pour déterminer la taille du stockage du cluster. Le système ne répond que lorsque les valeurs du monitoring échantillonnées sont hors de la plage des seuils. Une fonction de coût est utilisée dans cette étude. L'inconvénient de cette solution est que la définition des seuils nécessite une bonne connaissance de la charge de travail et dépend du choix des métriques comme les variables de performance (par ex. CPU).

3.2.3.3 Apprentissage par renforcement (Q-Learning)

L'apprentissage par renforcement fait référence à une classe de problèmes d'apprentissage automatique [105]. Le but est d'apprendre, à partir d'expériences, ce qu'il convient de faire en différentes situations, de façon à optimiser une réutilisation numérique au cours du temps. A l'exclusion de la charge imprédictible, un système d'apprentissage peut calculer la capacité demandée en se basant sur son historique. Parmi ses méthodes de modélisation nous citons : le processus de décision markovien (Markov Decision Process) et l'algorithme Q-learning. Parmi les travaux de recherche pour cette méthode:

Rao et al [97] ont proposé une approche fondée sur un apprentissage par renforcement (RL), nommée VCONF. La mise à l'échelle est effectuée par le (re) configuration automatique d'ordinateurs virtuels basée sur un modèle de RL. Les meilleurs résultats sont obtenus avec des charges de travail homogènes. L'inconvénient de cette solution est qu'il n'y a aucune garantie d'optimalité pour les configurations dérivées.

Barrett et al [20] ont proposé une approche Qlearning qui utilise des agents d'apprentissage parallèles. Cette approche réduit le temps nécessaire pour déterminer les politiques optimales tandis que l'apprentissage à temps réel. Cependant, elle ne résout pas la nécessité de bonnes politiques dans les premiers stades de l'apprentissage.

3.2.3.4 Analyse des time series

Les séries chronologiques "time series" (voir figure 3.3) sont utilisées dans de nombreux domaines, y compris les finances, l'ingénierie, l'économie et la bio-informatique, généralement pour représenter le changement d'une mesure dans le temps. Une série chronologique est une séquence de points de données, généralement mesurée à des instants temporels successifs espacés à des intervalles de temps uniformes. Un exemple est le nombre de requête

qu'une application peut atteindre, prise à des intervalles d'une minute. L'analyse des séries chronologiques pourrait être utilisée pour trouver des motifs répétitifs dans la charge de travail ou de tenter de prévoir les valeurs futures.

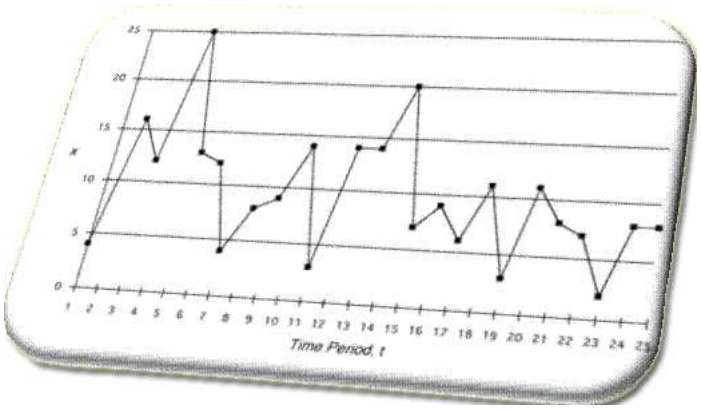

Figure 3.3: Analyse des time series [74]

Définition de la Technique certaines métriques de performance, comme la charge CPU moyenne ou la charge de travail d'entrée, seront échantillonnées périodiquement à des intervalles fixes (par exemple chaque minute). Le résultat sera une time-series X contenant une séquence des dernières observations w:

$$X = x_t + x_{t-1} + x_{t-2} + ... + x_{t-w+1}$$

Le problème de modélisation de la perfomance d'une application peut être divisé en deux parties: L'analyse des séries chronologiques est appliquée seulement dans la première partie, qui consiste à faire une estimation de *la future charge de travail* ou une *utilisation de ressource*. Sur la base de cette valeur prédite, la deuxième étape consiste à décider de l'action appropriée à prendre pour garantir une certaine performance. Plusieurs approches peuvent être utilisées dans la prise de décision comme un ensemble de règles prédéfinies ou de résoudre un problème d'optimisation de l'allocation des ressources [74]. Dans ce qui suit, nous allons nous concentrer sur la première étape qui utilise des techniques basées sur des séries chronologiques.

Comme indiqué précédemment, il existe deux principaux objectifs de l'analyse des séries chronologiques:

1. La prévision des valeurs futures de la série chronologique, basée sur les dernières observations,

2. Identification du modèle (si il est présent) que suit la série chronologique, puis l'extrapoler pour prédire les valeurs futures.

Dans les deux cas, l'information requise est une liste des dernières observations w de la série chronologique, que nous noterons fenêtre d'entrée "*input window*" ou fenêtre de l'histoire "*history window*".

Les techniques de prévision peuvent être appliquées soit pour la prédiction de charge de travail ou l'utilisation des ressources. Basées sur les dernières observations consécutives w $(x_t, x_{t-1}, x_{t-2}, ..., x_{t-w+1})$, une valeur future y_{t+r} est prévue, qui est r intervalles à venir de la fenêtre d'entrée "input window". Certaines des techniques utilisées à cet effet dans la littérature [74] sont la moyenne mobile, l'Auto-régression, ARMA (combinant les deux), le lissage exponentiel et différentes approches basées sur la machine d'apprentissage. Dans ce qui suit, nous nous concentrons sur les techniques de prévision de *"La moyenne mobile"* nommées aussi *"Moving Average"* .

Les méthodes de la moyenne mobile : Elles peuvent être utilisées pour lisser une série chronologique afin de supprimer le bruit ou pour faire des prédictions. La valeur de prévision y_{t+1} est calculée comme la moyenne pondérée des dernières valeurs w consécutives. La formule générale est la suivante: $y_{t+1} = a_1 x_t + a_2 x_{t-1}, ...,$ où $a_1, a_2, ..., a_w$ sont un ensemble de facteurs de pondération positifs qui doivent sommer la valeur 1. Selon la façon de déterminer ces poids, plusieurs méthodes sont définies:

- **Moving average MA(q):** La simple moyenne mobile ou MA est la moyenne arithmétique des dernières valeurs w ou q, c'est à dire, il attribue des poids égaux à toutes les observations.
- **Moyenne mobile pondérée WMA(q):** des poids différents sont affectés à chaque observation. Typiquement, plus de poids est donné aux termes les plus récents de la time-series et moins de poids aux données les plus anciennes.
- **Lissage exponentiel (Exponential smoothing):** Il affecte de façon *exponentielle* la diminution du poids au fil du temps. Un nouveau paramètre est introduit, un facteur α de lissage qui affaiblit l'influence des données passées. La formule de prédiction pour le lissage exponentiel simple est:

$$
\begin{aligned}
& y_{t+1} = \alpha x_t + (1-\alpha) y_t \\
&= \alpha x_t + (1-\alpha)[\alpha x_{t-1} + (1-\alpha) y_{t-1}] \\
&= \alpha x_t + \alpha(1-\alpha) x_{t-1} + (1-\alpha)^2 [\alpha x_{t-2} + (1-\alpha) y_{t-2}] \\
&= \alpha x_t + \alpha(1-\alpha) x_{t-1} + (1-\alpha)^2 x_{t-2} + ... + (1-\alpha)^w - 1 x_{t-w+1}
\end{aligned}
$$

où y_{t+1} représente la valeur de prédiction pour la période $t+1$, x_t est la valeur à l'instant t, et y_t est la prévision faite pour la période t. *Le lissage exponentiel simple* est adapté pour les séries chronologiques qui ont aucun changement de tendance significatif, alors que *le double lissage* peut être appliquée à des séries chronologiques avec une tendance linéaire existante.

D'autre part, *Le triple lissage exponentiel* peut être utilisé pour des séries chronologiques avec la tendance et la saisonnalité. Enfin, le double et le triple lissage exponentiel sont dérivés en appliquant le lissage exponentiel pour les données déjà lissées.

Méthodes des motifs fréquents dans les séries chronologiques: Plusieurs méthodes sont utilisées pour trouver les motifs fréquents à savoir, Pattern matching, Signal processing techniques (Fast Fourier Transform (FFT) et l'auto-correlation. Une technique de base pour la représentation des séries chronologiques est un histogramme. Il consiste à diviser la série chronologique en plusieurs bandes de même largeur, et représentant la fréquence pour chaque bande. Il a été utilisé dans la littérature pour représenter la structure et la distribution de l'utilisation des ressources, puis prédire des valeurs futures. La méthode utilisée dans notre approche est le "Pattern matching".

La précision des algorithmes de séries temporelles dépendent de différents paramètres tels que la longueur de l'intervalle de monitoring, la taille de la fenêtre historique qui détermine la sensibilité de l'algorithme pour les tendances locales versus les tendances globales, et la fenêtre d'adaptation qui détermine dans quelle mesure dans l'avenir le modèle peut s'étendre.

Travaux de recherche Dans la littérature, les techniques des séries chronologiques ont été appliquées principalement pour les prédictions de la charge de travail ou l'utilisation de ressource. Une moyenne mobile simple pourrait être utilisée à cette fin, mais avec des résultats médiocres [117]. Pour cette raison, les auteurs ont appliqué cette méthode pour supprimer uniquement le bruit de la série chronologique [94], ou tout simplement pour avoir une bonne comparaison (aune). Par exemple, Huang et al. [58] ont présenté un modèle de prédiction de la ressource (pour l'utilisation CPU et l'utilisation de la mémoire) sur la base d'un lissage exponentiel double (double exponential smoothing), et ils l'ont comparé avec une simple moyenne mobile et une moyenne mobile pondérée (WMA). Le lissage exponentiel obtient clairement de meilleurs résultats, car il prend en compte les données actuelles et les archives de l'histoire pour la prédiction. Mi et al. [85] ont également utilisé un lissage exponentiel quadratique par rapport à des traces de charge de travail réelle (Coupe du monde 98 et ClarkNet), et ont montré de bons résultats précis, avec une petite quantité d'erreur (erreur relative d'une moyenne de 0,064 pour le meilleur des cas).

3.2.4 Approche de garantie de performance basée sur un modèle de performance

Shao et al. [100] présentent une garantie de performance pour les applications de cloud basée sur *un modèle de performance* et sur le monitoring. Extrait à partir des données surveillées en exécution en utilisant des techniques datamining, le modèle peut être utilisé pour diriger comment ajuster la stratégie de la fourniture des ressources sous une exigence de performance donnée. Un framework de monitoring est mis en oeuvre pour la collecte des données surveillées

en temps d'exécution sur un véritable Cloud qui fournit un service PaaS. Les contributions de cette approche sont répertoriées comme suit:

- Un ensemble pratique de métriques de performance pour la qualité de Service (QoS) dans les Clouds, qui peut être utilisé pour évaluer la performance des applications du cloud computing.
- Un modèle pour la garantie de performance et ajustement de la provision des ressources adaptative est construit en utilisant des techniques datamining basées sur la masse des données collectées lors de l'exécution (voir la figure 3.4).
- Un framework de surveillance pour la collecte de données de performance de l'application Cloud et les attributs qui peuvent affecter la performance.

Les Attributs QOS pour les applications Clouds Deux attributs QOS (qualité de service) sont considérés pour mesurer la performance des applications.

- *Disponibilité (Availability)*: La disponibilité est utilisée pour mesurer l'état de préparation pour la prestation de services d'une application cloud. Il est l'attribut de qualité le plus important dans le QOS (la qualité de service). Il est utilisé pour évaluer si l'application du Cloud peut offrir un service tout le temps.
- *Le temps de réponse (Response time)* : On définit le temps de réponse en tant que temps passé à partir de l'utilisateur final pour envoyer une demande, le contenu demandé est téléchargé complètement sur le côté client.

Conception du modèle de performance Le résultat du modèle est la performance d'une application cloud spécifique, qui est sur la base d'un certain nombre d'attributs pertinents. La valeur de ces attributs et la performance correspondante peuventt être collectées par un monitoring continu.

Le but de ce modèle de performance est de faire la prévision numérique, en appliquant l'algorithme de régression linéaire [112] sur l'ensemble de données collectées. La métrique de performance P est le résultat du modèle et qui peut être exprimé sous la forme d'une combinaison linéaire des attributs, avec des poids prédéterminés, comme représenté dans la formule (1).

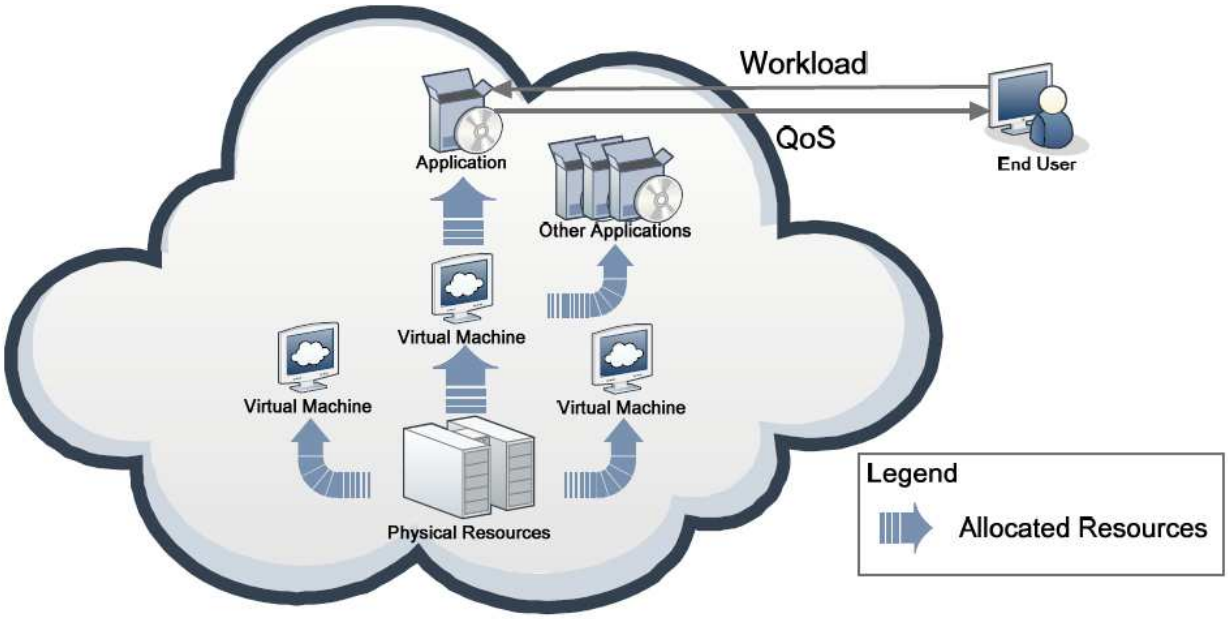

Figure 3.4: Modèle de performance pour les applications Cloud [100]

$$P = w_0 + a_1.w_1 + a_2.w_2 + ... + a_k.w_k \ (1)$$

Dans la formule (1), P est la métrique de performance; $a_1, a_2, ..., a_k$ sont les valeurs des attributs et $w_1, w_2, ..., w_k$ sont les poids. Les poids sont calculés à partir des données d'apprentissage. Aprés que tous les poids sont calculés, le modèle est conçu.

3.3 Monitoring des applications Cloud basé sur les évènements

Dans la plus part des systèmes de monitoring, les agents communiquent deux sortes de messages : des informations d'état (exemple : l'utilisation actuelle du CPU est 2.0) et des informations d'évènement (exemple : l'utilisation du disque pour une partition particulière dépasse le seuil donné).

Les évènements peuvent être liés à un changement significatif dans l'état du système d'exploitation ou d'applications, ils peuvent être générés non seulement pour les problèmes mais pour achèvements avec succès des tâches planifiées (par exemple : un hôte en cours de redémarrage, se reconnecter en tant qu'administrateur, etc). A cet effet une approche de monitoring basée sur l'obtention d'informations sur la survenue d'événements intéressants, peut être utilisée. C'est une vision plus dynamique du système de monitoring où seules les modifications importantes dans le système sont recueillies par [7] :

1. La détection d'évènements et la transmission pour traitement.
2. Le pré-filtrage ou la corrélation d'évènements.
3. La corrélation plus complexe des évènements d'état.

4. L'action par rapport aux évènements traités.

5. L'archivage des évènements traités.

3.3.1 Approache basée sur le paradigme de traitement d'évènement complexe

L'approche Event-Based Monitoring [71] présente un framework pour la collecte des métriques de performance au niveau de l'application. L'infrastructure de monitoring proposée utilise le paradigme de traitement des événements complexes(CEP) [75].

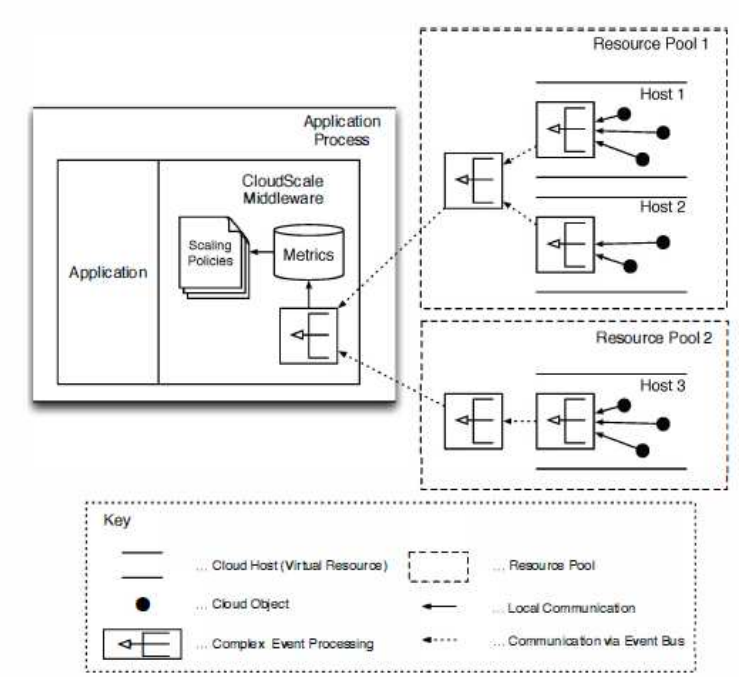

Figure 3.5: Vue d'ensemble du monitoring [71]

Dans l'ensemble, cette approche est basée sur les notions de monitoring et CEP à base d'événements. A cet effet, l'idée est que les différents composants pertinents dans le système émettent des événements, qui indiquent leur statut actuel (par exemple, qu'un nouvel objet Cloud a été déployé, ou qu'un objet de Cloud a commencé à exécuter une méthode donnée). Nous nous référons à ces éléments comme des *émetteurs d'événements*. Les émetteurs d'événements importants incluent des objets de Clouds et les hôtes de Clouds , mais, en principe, tout code Java exécuté dans le Cloud peut agir comme *un émetteur d'événement* en écrivant dans le flux d'événements respectifs. Les événements produits par les émetteurs d'événements sont ensuite transformés en une connaissance de niveau supérieur en utilisant des techniques CEP (un processus que nous appelons la corrélation des événements). Enfin, les métriques de monitoring (de surveillance) sont définies à partir de ces événements complexes de haut niveau. Pour des raisons de performances, les événements sont corrélés en plusieurs étapes. La Figure 3.5 présente l'architecture du système de surveillance proposée.

3.4 Approche mOSAIC

Rak et al. [96] proposent le monitoring de l'application Cloud en utilisant l'approche mOSAIC. Dans une première étape, les auteurs décrivent le développement des applications personnalisées en utilisant l'API mOSAIC pour être déployé sur les environnements de cloud. Pour ces applications, ils proposent dans une seconde étape des techniques de surveillance. Leur intérêt est seulement de collecter des informations qui peuvent être utilisées pour effectuer l'équilibrage de charge manuel ou automatique, augmenter / diminuer le nombre de machines virtuelles ou de calculer le coût total de l'exécution de l'application. Leur approche ne tient pas compte de la détection des violations SLA pour éviter le coût de pénalité SLA et d'ailleurs, elle n'est pas générique, car elle surveille seulement les applications développées en utilisant l'API mOSAIC.

L'approche mOSAIC, dispose d'une API et d'un framework mOSAIC qui visent à offrir une solution pour le développement d'applications Cloud interopérables, portables et indépendantes des fournisseurs Cloud. Elle dispose aussi de composants de surveillance mOSAIC qui facilitent la construction des systèmes de monitoring personnalisés pour des applications Cloud utilisant *l'API mOSAIC*.

3.5 GMonE: Approche complète pour le cloud monitoring à base de vision

Montes et al. [87] proposent GMonE, une solution qui peut être classée comme "Cloud monitoring vision". D'un point de vue général, le montoring côté client et côté fournisseur de service Cloud peuvent être distingués. GMonE est un outil de monitoring Cloud à usage général applicable à toutes les couches des Clouds. Elle permet aux instances de monitoring d'être déployées sur n'importe quel niveau du Cloud et fournit un modèle de plug-in où les utilisateurs peuvent injecter leur propres scripts de collecte des métriques personnalisées à des instances de monitoring.

En résumant, GMonE est une plate-forme de monitoring Cloud visant au monitoring des différents niveaux de service (de SaaS jusqu'a IaaS) pour soutenir le cycle de vie des services déployés dans le Cloud "Monitoring multiple levels". Cependant, leur solution nécessite la mise en œuvre de plug-ins qui sont fortement couplés à l'infrastructure de chaque fournisseur de cloud, qui le rend impropre pour les fédérations de cloud computing.

La principale contribution de cette approche est de concevoir un framework de monitoring Cloud à usage général, couvrant tous les différents scénarios du monitoring des Clouds. Ceci est une étape importante dans le monitoring du Cloud, créant la base d'une solution optimale de surveillance des Clouds. Plus près de cette approche est Lattice [35], un framework de monitoring pour les réseaux virtuels. Ce framework utilise des sources de données et des probes (agents) pour collecter diverses données de surveillance pour les machines physiques et virtuelles. Cette approche fournit également une telle fonctionnalité en permettant à

l'utilisateur de définir des plug-ins. Cependant, un peu plus loin elle fournit une vision double incluant des visions pour le fournisseur et le client du Cloud respectivement.

3.6 Conclusion

Dans la première partie de cette section, nous avons évoqué le monitoring applicatif basé sur un modèle de performance. La modélisation de la performance d'une application dans un environement virtualisé et plus précisément dans le Cloud Computing est une tâche difficile et nécessaire. Les modèles de performance fournissent la capacité de prédire les performances des applications pour un ensemble donné de ressources matérielles et sont utilisés pour la planification des capacités et la gestion des ressources. Plusieurs approches de modélisation sont proposées dans la littérature, telles que la théorie des files d'attente, la théorie de contrôle, l'apprentissage par renforcement et l'analyse des times series. Les modèles de prédictions quantitatives sont utilisées principalement pour prédire la charge du travail ou l'utilisation des ressources. Plusieurs méthodes sont proposées dans la littérature pour la prévision des valeurs. Nous distinguons les méthodes basées sur la moyenne, l'apprentissage automatique et le filtrage par motif. Ces méthodes sont souvent combinées pour obtenir diverses variantes de prédiction. Il est donc intéressant d'essayer, avec la même approche et plus précisément par "l'analyse des times series", de résoudre d'autres problématiques, comme la garantie de performance des applications scientifiques dans un environement Cloud. L'analyse des times series couvre un large éventail de méthodes qui suivent une approche prédictive et semble être un domaine de recherche prometteur.

Dans la seconde partie de cette section, nous avons évoqué des projets qui proposent de lever la complexité du monitoring applicatif en utilisant d'autre approches que la modélisation de la performance. Ces travaux ont adressé les problématiques, telles que le monitoring basé sur des applications développées en utilisant des APIs spécifiques, le monitoring à base d'événements, et le monitoring multi-niveaux. Ces travaux de recherche ont des points de similitude avec les concepts de notre étude. Plus précisément, notre approche se base sur les approches basées sur le monitoring "multi niveaux".

Part II

Contribution

CHAPTER 4

Système muli-niveaux basé modèle de performance

4.1 Introduction

Nous avons présenté en premier lieu dans les chapitres précédents, le contexte de ce travail de thèse en fournissant un état de l'art sur le concept des deux paradigmes "Grid" et "Cloud" et sur l'étude comparative des deux paradigmes. Notre étude a montré que la combinaison des deux paradigmes améliore la gestion de la grille dans son modèle actuel en bénéficiant des avantages Cloud comme l'organisation autonome, la gestion simplifiée des ressources à la demande pour rencontrer les exigences des applications de hautes performances et une architecture plus élastique, flexible et dynamique. Pour adresser cet objectif, nous avons suivi l'approche "la grille sur le Cloud", l'une des approches proposées dans la littérature concernant la combinaison des deux paradigmes.

En deuxième lieu, nous avons fait une étude sur le monitoring dans le Cloud et un état de l'art sur le monitoring applicatif et la modélisation de la performance des applications dans le Cloud Computing. A cet effet, nous présentons dans la deuxième partie de ce chapitre, notre contribution qui consiste a proposer un système de monitoring multi-niveaux pour surveiller le comportement des applications scientifiques, et un modèle de performance qui interprète l'utilisation des ressources pour prédire périodiquement la demande future. Cela permet de déterminer les besoins en ressources et décider de l'action appropriée pour assurer la haute disponibilité des applications scientifiques dans un environnement "Grid-Cloud".

4.2 Approche d'intégration "Grid-Cloud"

Les grilles de calcul fournissent une large plateforme de calcul afin d'intégrer l'ensemble de toutes les ressources dans le but de délivrer un calcul de haute performance, mais l'architecture actuelle rencontre des problèmes majeurs :

1. Avec la demande croissante des ressources par les utilisateurs et les calculs urgents des projets de recherche, certaines applications scientifiques peuvent ne pas obtenir les

ressources nécessaires quand elles sont demandées;

2. Le problème de planification et de gestion des ressources dans l'infrastructure.

Pour essayer de résoudre ces problèmes rencontrés dans les infrastructures existantes des grilles de calcul et offrir une architecture plus élastique, flexible et dynamique, des solutions *d'intégration entre l'infrastructure grille et Cloud* sont proposées.

En effet, l'introduction des capacités flexibles du système d'administration Cloud, et l'ajout des technologies de virtualisation à l'infrastructure grille existante peut simplifier la planification et la gestion des ressources à la demande pour rencontrer les exigences des applications de hautes performances qui s'exécutent sur la grille.

Les différentes approches proposées dans la littérature [29, 15] concernant la combinaison des deux paradigmes sont :

- L'approche 1 : "La grille sur le Cloud", Il s'agit d'utiliser et d'adapter le système Cloud IaaS pour construire et administrer des services grille. Dans cette approche, les services grille sont exécutés sur des machines virtuelle d'une infrastructure Cloud et administrées par le système flexible du Cloud lui-même.
- L'approche 2 : "Le Cloud sur la grille" [113], qui consiste en l'utilisation des ressources de calcul de la grille pour la construction d'infrastructure Cloud.

Notre objectif est d'adapter un système de monitoring à plusieurs niveaux basé modèle de performance pour un environnement d'intégration "Grid-Cloud" qui suit l'approche 1 : "*La grille sur le Cloud*".

4.2.1 Objectifs d'intégration "Grid-Cloud"

L'intégration entre l'infrastructure grille et Cloud est considérée comme un nouveau paradigme qui améliore la gestion de la grille dans son modèle actuel et assure les fonctionnalités suivantes:

- Par l'introduction des capacités de gestion de cloud, les grilles peuvent devenir dynamiques.
- Simplifier la tâche aux utilisateurs de grille pour sélectionner, configurer et gérer les ressources selon les exigences de l'application.
- Permettre aux utlisateurs grille d'exploitater et d'utiliser des ressources de calcul via deux interfaces : Une vue d'abstraction des ressources basée "services" à travers les APIs grille ou une vue d'abstraction basée "machine virtuelle" à travers les APIs Cloud.
- Séparation des responsabilités d'administration et de gestion : La gestion de l'infrastructure physique et matérielle est déléguée aux administrateurs Cloud, tandis que la gestion des logiciels et middlewares est déléguée aux administrateurs grille.

4.2.2 L'Appliance Virtuelle "Virtual Appliance"

Dans un environnement cloud computing, un élément clé de l'interaction entre les forunisseurs de services et l'infrastructure est le mécanisme de ***définition de service***. Dans un Cloud IaaS on peut le définir comme étant un paquetage logiciel (système d'exploitation, middleware et composants service) nommée aussi ***Virtual Appliance***.

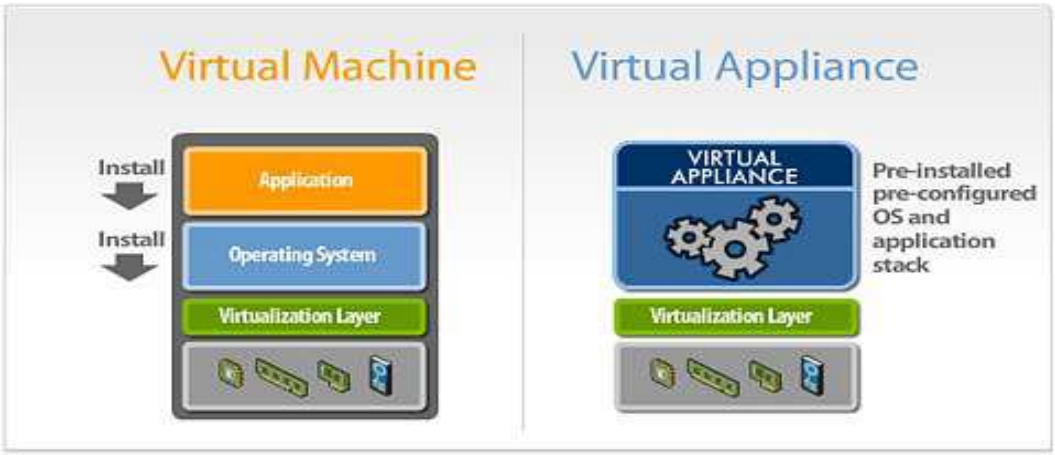

Figure 4.1: "Virtual Appliance"

Une appliance virtuelle est une image de machine virtuelle pré-configurée, prête à fonctionner sur un hyperviseur (voir figure 4.1). Les appliances virtuelles sont destinées à éliminer les coûts d'installation, de configuration et de maintenance associés à la gestion des piles de logiciels complexes. Une appliance virtuelle n'est pas une plate-forme de machine virtuelle complète, mais plutôt une image logicielle contenant une pile logicielle conçu pour fonctionner sur une plate-forme de machine virtuelle.

Dans notre cas, une Appliance virtuelle est une description de service, et peut être définit comme étant un paquetage logiciel composé d'une image d'un système d'exploitation, d'un ensemble logiciels du Middleware de grille "EMI" et des composants du service requis pour la mise en fonction d'un service grille donné (VOMS, CE, WNs...etc). A cet effet, chaque service de la grille possède une Appliance virtuelle qui le décrit. L'Appliance virtuelle est utilisée pour créer une image disque de VM pour chaque service grille. Les images disques créés sont ensuite sauvegardées dans un espace spécifique appelé "VM Image Storage". Les images disques sont ensuite instanciées afin de créer une Machine Virtuelle (VM) spécifique à un service grille approprié (figure 4.2). Une stratégie d'appliances virtuelles permet également de déployer et d'ajuster rapidement des ressources.

4.2.3 Le concept en couches de l'approche "grid sur le cloud"

Suivant le modèle proposé par [69] et qui s'appuit sur l'approche d'intégration "grille sur le cloud", l'environement est divisé en trois couches d'abstraction essentielles:

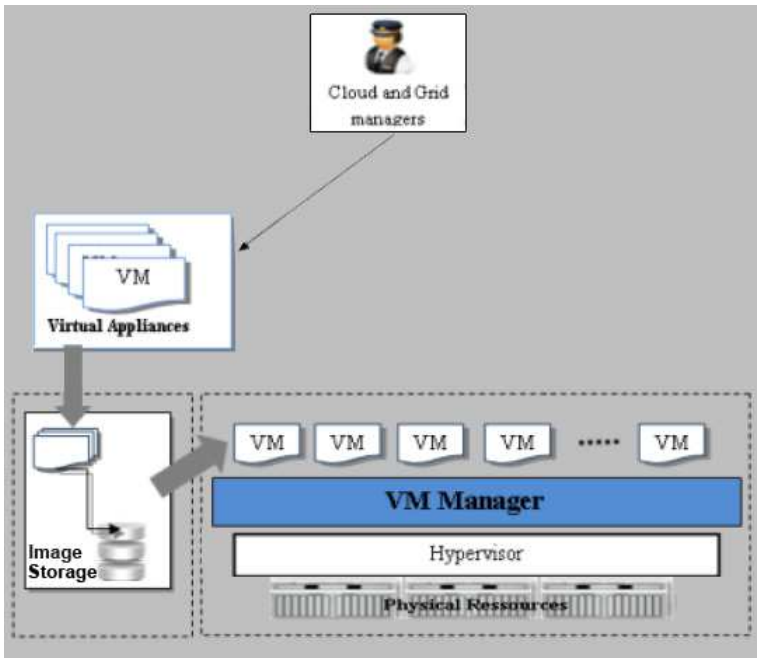

Figure 4.2: Processus de création d'une VM.

La couche utilisateur "User Layer" qui présente les utilisateurs de l'environement "Grid-Cloud", la couche service "Service Layer" qui comporte les services grille "Grid Services" et "architecture de grille" selon le Middleware "EMI" et, la couche Infrastructure "Infrastructure Layer" qui représente les niveaux de stockage, de calcul, et le réseau.

Nous avons conçu un système de monitoring à plusieurs niveaux basé modèle de performance, et qui s'étale sur toutes les couches du modèle d'intégration "Grid-Cloud" (voir le schéma 4.3). Le système de monitoring à plusieurs niveaux basé modèle de performance collectent différentes mesures et des indicateurs de performance qui renseignent sur l'état des applications et de l'infrastructure (physique et virtuelle). Les métriques collectées seront utilisées comme entrée dans notre modèle de performance qui se base sur une approche de modélisation et utilise des techniques de prévisions pour prédire des valeurs futures. Sur la base de ces valeurs prédites, des règles de décision doivent être définies afin de garantir une certaine performance.

4.3 Le système de monitoring Multi-Level basé modèle de performance

4.3.1 Problématique et motivations

La problématique de ce travail de thèse fut le monitoring multi-niveaux basé modèle de performance des applications scientifiques qui s'exécutent sur des services grille déployées sur l'infrastructure Cloud IaaS. Dans le chapitre deux, nous avons fourni les principales propriétés d'un système de monitoring dans le Cloud, les concepts de base pour la construction d'un tel

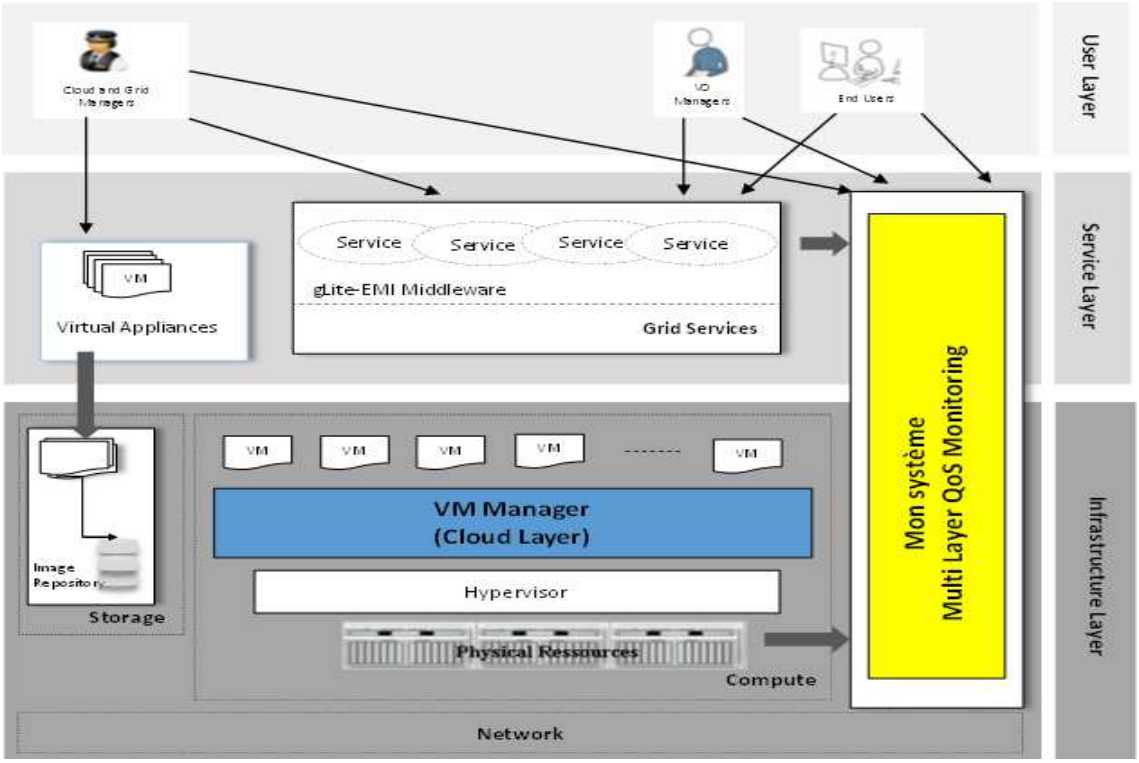

Figure 4.3: Le concept en couches de l'approche "grille sur le cloud"

système ainsi que les contributions connexes prévues dans la littérature. Nous avons aussi, présenté quelques solutions de monitoring en les classant dans des catégories *à usage général* et *spécifiques Cloud* afin de mieux comprendre les outils et analyser historiquement leur évolution. Plusieurs outils de monitoring à usage général sont actuellement adaptés au monitoring dans les Clouds, et ce processus d'adaptation continue dans l'avenir. Plus précisément, plusieurs des approches et plates-formes de monitoring proposées pour le scénario Grille ont été personnalisées pour les systèmes Cloud. Zanikolas et al. [11, 115] ont fait une étude dans le domaine de recherche du monitoring dans la grille en introduisant les concepts, les exigences et les phases impliqués ainsi que les activités de normalisation liées.

Le monitoring dans les environnements Cloud a été étudié dans la littérature [34, 31, 64, 43] pour la surveillance des ressources physiques et virtuelles des infrastructures de Cloud. Cependant, la majorité des solutions de monitoring se concentrent sur des métriques de bas niveau, tel que l'utilisation du CPU, la consommation de mémoire ou la bande passante du réseau. Nous soutenons que ces métriques, tout en étant sans doute pertinentes ne font pas saisir pleinement la performance réelle de l'application [71] et plus précisément les applications scientifiques qui ont des besoins spécifiques en matière de calcul haute performance pour une gestion efficace face à la qualité de service et les contraintes de la performance qui reste un objectif.

Plus concrètement, le domaine du monitoring des applications a besoin de plus de recherches et plus précisément pour les applications scientifiques, notamment quand elles s'éxécutent sur

un environnement “Grid-Cloud” qui rajoute une complexité dans l’architecture résultante. Dans ce cas de figure, une prédiction de la performance, des outils et des modèles pour surveiller ces applications (monitoring de la performance des applications) deviennent critiques et pertinentes.

A cet effet, de nouvelles approches qui reposent sur des modèles de performance permettent d’analyser et traiter les données collectées afin de prédire les performances des applications pour un ensemble donné de ressources matérielles et sont utilisés pour la planification des capacités et la gestion des ressources [100, 67, 70]. Un modèle de performance rapporte quantitativement des métriques de qualité de service (QoS) au niveau de l’application (par exemple, le temps de réponse d’une application cliente) avec une ressource donnée [53].

Plusieurs approches de modélisation sont proposées dans la littérature, telles que la théorie des files d’attente, la théorie de contrôle, l’apprentissage par renforcement et l’analyse des times series. Notre approche se base sur les modèles de prédictions quantitatives tel que “l’analyse des times series”, et qui sont utilisés principalement pour prédire la charge du travail ou l’utilisation des ressources. Plusieurs méthodes sont proposées dans la littérature pour la prévision des valeurs. Nous distinguons les méthodes basées sur la moyenne, la machine d’apprentissage (machine learning), réseaux de neurone (Artificial Neural Networks), et la régression. Ces méthodes sont souvent combinées pour obtenir diverses variantes de prédiction. Notre approche se base sur les techniques de prévision de *“La moyenne mobile”*.

Dans cette partie, nous allons présenter une nouvelle approche multi niveaux de monitoring des applications scientifiques en se basant sur un modèle de performance. Ces applications scientifiques s’exécutent sur des services grille déployées dans un environnement cloud (plateforme grille virtualisée).

4.4 Approche proposée

Pour évaluer la performance des applications scientifiques dans un environnement “Grid-Cloud”, qui sont exigentes en matière de puissance de calcul et sensibles au temps, il faut définir des métriques quantitatives qui expriment le comportement de l’application. Les deux plus importants critères sont la disponibilité et la performance [114]. Pour pouvoir atteindre ces mesures fines, il faut trouver la bonne combinaison ou relation entre les métriques; mais trouver la bonne combinaison est un problème difficile à maîtriser.

Pour cela concevoir un modèle de mesure de performance d’applications permet de prendre en compte les différentes mesures et aider à détecter la source de dégradation de performance afin de prendre des actions correctives. Grace à ce modèle il serait alors possible de gérer et planifier les ressources nécessaires afin de rencontrer des niveaux de services ciblés.

A cet effet, notre approche consiste à intégrer un ensemble d’outils de monitoring multi niveaux capable de générer des mesures à plusieurs niveaux et un modèle de performance en fournissant la capacité de prédire les performances des applications pour un ensemble donné de ressources matérielles. L’importance de modéliser correctement le comportement de la

performance des applications est prononcée dans les plates-formes virtualisées où de multiples applications partagent les mêmes ressources physiques (ressources de l'hôte). L'objectif à travers cette approche est:

- La collecte des métriques quantitatives sur plusieurs couches (métriques de bas niveau et haut niveau);
- La compréhension des métriques quantitatives qui évaluent la performance des applications.
- Construire un modèle de performance d'application scientifique qui prédit le nombre de ressources nécessaire pour garantir une certaine performance dans un evironnement "Grid-Cloud";
- Prédire périodiquement la demande future et déterminer les besoins en ressources en utilisant le modèle de performance qui se base sur la modélisation et la prédiction;
- Allouer automatiquement des ressources à l'aide des besoins en ressources prévues.

4.4.1 Vue d'ensemble de l'approche

L'objectif principal de notre approche est de garantir une certaine performance et une haute disponibilité pour les applications scientifiques dans l'environnement "Grid-Cloud". Pour ce faire, nous avons catégorisé notre approche comme montré dans le diagramme conceptuel en cinq sujets principaux (voir figure 4.4): Domaines et applications, attributs de qualité, gestion affiliée, modélisation et prédiction, et enfin le niveau d'autoscaling dans le Cloud.

1. **Domaines et applications**: Notre approche s'applique sur les applications scientifiques qui nécessitent une puissance de calcul et une disponibilité de ressource.
2. **Attributs de qualité** : Notre approche est basée sur des métriques quantitatives qui expriment le comportement de l'application. Les deux plus importants critères sont la disponibilité et la performance [114].
3. **Gestion affiliée**: Notre approche exige un système de monitoring qui mesure et génére les métriques à plusieurs niveaux de toute l'infrastructure (physique, virtuelle et applicatifs).
4. **Modélisation et prédiction**: Pour trouver la bonne combinaison entre les métriques, il faut modéliser le comportement de l'application en se basant sur une approche de modélisation et des techniques de prédiction.
5. **Niveau d'autoscaling dans le Cloud**: L'autoscaling dans notre approche est appliquée au niveau IaaS de l'environnement Cloud.

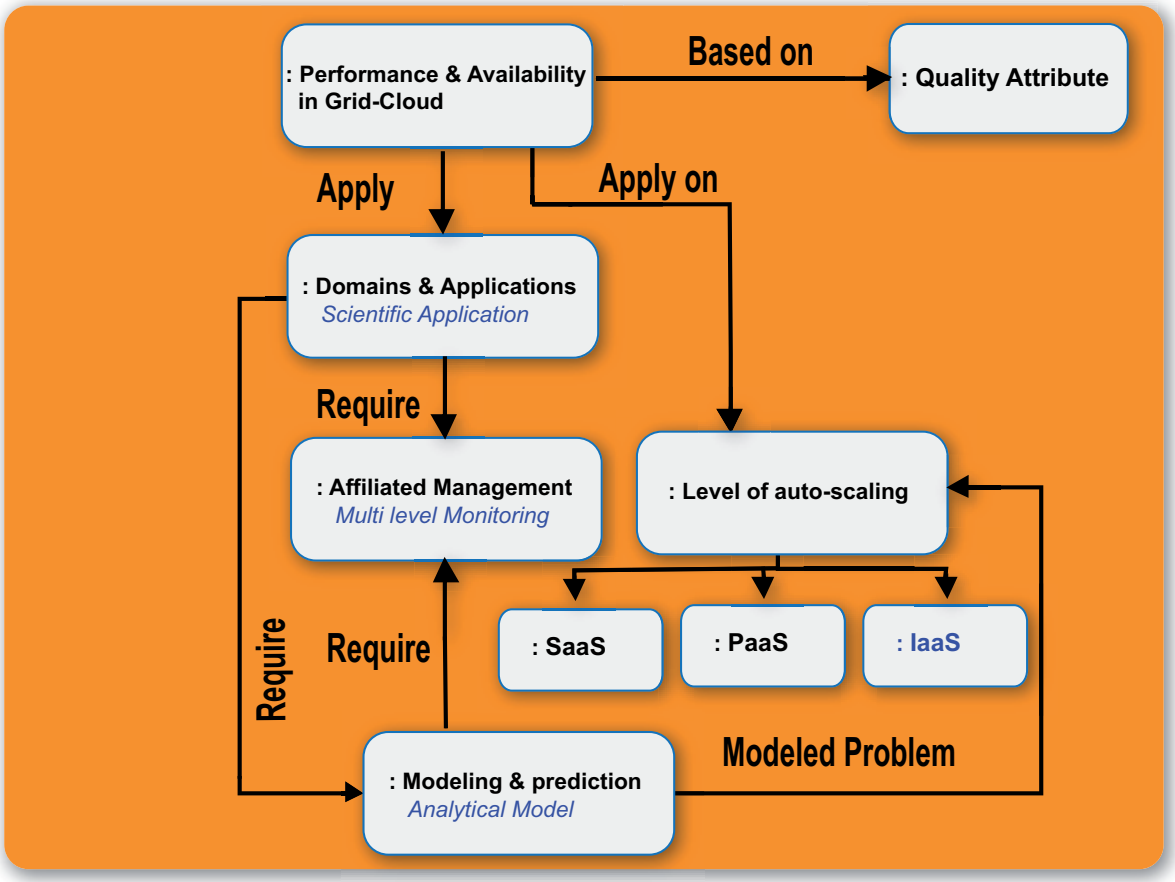

Figure 4.4: Diagramme conceptuel de l'approche

4.4.2 Modélisation de l'approche

Notre système de monitoring consiste à mesurer et générer des métriques à plusieurs niveaux de toute l'infrastructure (physique , virtuelle et applicatifs), ensuite de collecter et selectionner les données mesurées afin de les envoyer au modèle de performance capable de les traiter, les corréler et les analyser pour une prise de décision permettant une prédiction en besoin de ressources et un redimensionnement automatique (mise à l'echelle) (voir figure 4.5).

A cet effet, notre "système multi-niveaux de monitoring basé modèle de performance", peut être modélisé en deux unités importantes: (voir figure 4.6):

1. Une unité de monitoring multi-niveaux qui collectent différentes mesures et des indicateurs de performance qui renseignent sur l'état des applications et de l'infrastructure (physique et virtuelle) et la compréhension et le choix précis des métriques quantitatives qui évaluent la performance des applications. L'architecture de cette unité suit une ***approche producteur-consommateur basée agent***. Cette approche fournit un système de monitoring Cloud en temps réel, évolutive, et qui est utilisé pour collecter des métriques de monitoring à partir de plusieurs couches de l'infrastructure sous-jacente,

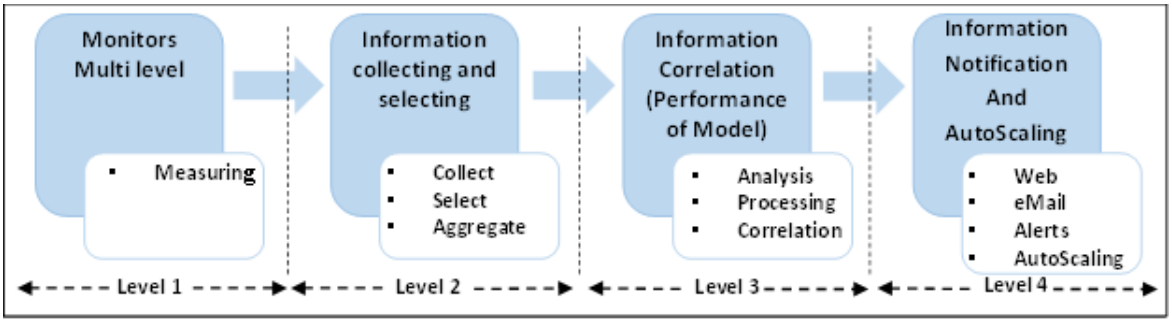

Figure 4.5: Niveaux de traitement de l'approche monitoring

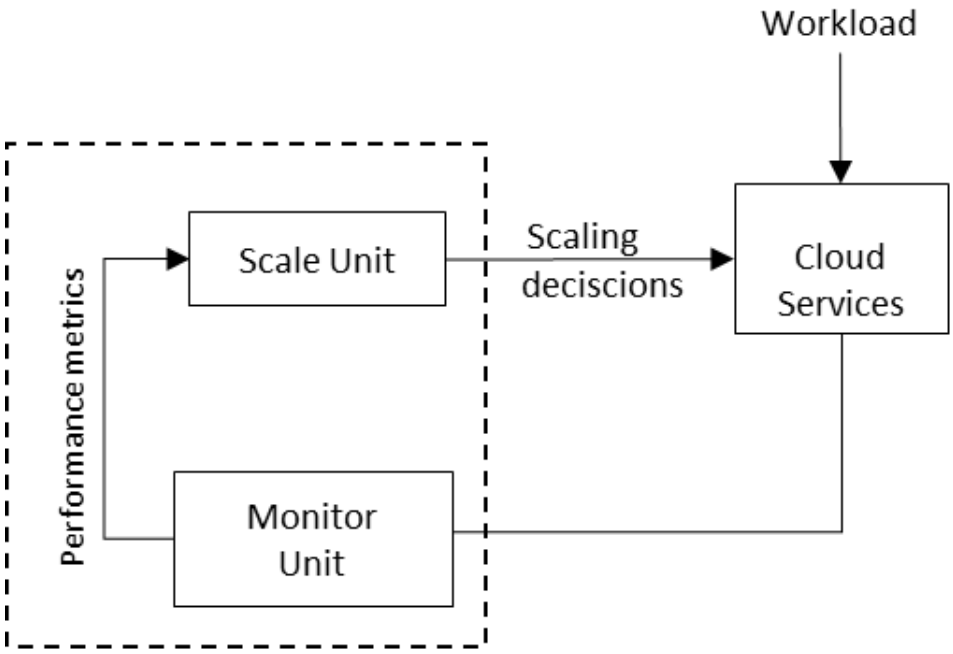

Figure 4.6: Modélisation de l'approche

ainsi que des indicateurs de performance des applications déployées.

2. Un modèle de performance qui est divisé en deux parties: La première partie, consiste à faire une estimation de la future charge de travail ou une utilisation de ressource en se basant sur des approches de modélisation de performance d'application plus précisement "l'analyse des séries chronologiques". Sur la base de cette valeur prédite, la deuxième étape consiste à décider de l'action appropriée à prendre avec des règles de décision qui doivent être définies afin de faire des notifications aux administrateurs ou bien déclencher le dimensionement automatique (ajout des VMs).

Nous soulignons qu'une bonne utilisation des solutions de monitoring facilite énormement la prise de décision pour un redimensionnement dynamique de ressources [74].

4.5 Architecture du système multi niveaux basé modèle de performance

Notre système de monitoring multi-niveaux basé modèle de performance (voir la figure 4.7) est constitué de cinq modules principaux qui communiquent entre eux et qui sont adaptés à un environnement Cloud et à n'importe quel environnement distribué.

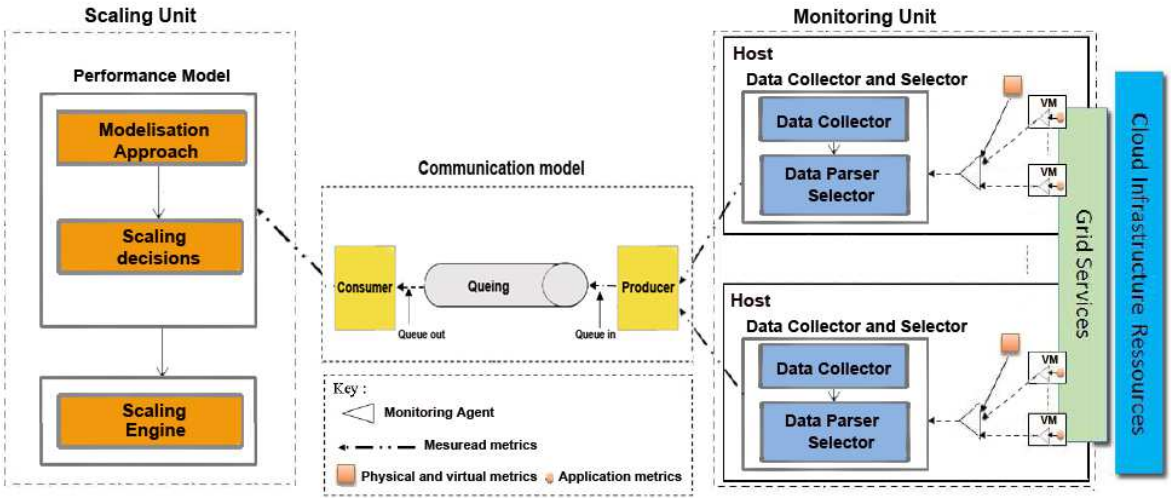

Figure 4.7: Architecture de l'approche de monitoring

1. **Module de monitoring multi niveaux "multi-level Monitor":** (niveau 1 génération de métriques quantitatives): Il se compose d'un ensemble d'outils de monitoring situés sur plusieurs niveaux dans un environnement Cloud, à savoir des outils de monitoring qui génèrent des métriques de bas niveau et qui sont situés sur chaque nœud physique de l'environnement Cloud (compute node) et des outils de monitoring qui génèrent des métriques de haut niveau situées sur les machines virtuelles de l'infrastruture cloud pour fournir des informations sur le comportement des applications s'y exécutant (eg: Temps de réponse). Ce module sera détaillé dans la section 4.5.1

2. **Module de collecte et sélection de métriques "Data Collector and Selector"** (niveau 2 collecte et sélection de métriques): Situé sur chaque source d'information de monitoring à savoir les nœuds physiques de l'environnement Cloud et les machines virtuelles de l'infrastruture cloud. Il est responsable de la collecte des métriques de monitoring issus de plusieurs sources et la sélection des métriques collectées. La compréhension et le choix précis des métriques quantitatives qui évaluent la performance des applications, sont utiles pour améliorer le processus de prise de décision concernant la quantité de resources necessaires pour assurer la QOS d'une application hébergée dans le cloud. Ce module sera détaillé dans la section 4.5.2.

3. **Module de communication**: Les différents modules "Data collector and selector"

situés sur les noeuds physiques de l'environnement Cloud et le module "modèle de performance" échangent un volume considérable de métriques entre eux, pour cela un mécanisme de communication fiable est nécessaire. Ce module sera détaillé dans la section 4.5.3.

4. **Module modèle de performance** (niveau 3 analyse et traitement pour prise de décision): Notre modèle de performance détermine quels types de relations existent entre les différentes métriques sélectionnées à partir du module précédent et qui reflètent mieux le concept de performance pour l'application et planifie les ressources nécessaires afin de rencontrer des niveaux de services ciblés. Une approche de modélisation et des techniques de prédiction sont utilisées pour pouvoir prendre des décisions de redimensionnement automatique et notifications. Ce module sera détaillé dans la section 4.5.4.

5. **Module Mise à l'échelle automatique des ressources et notification** (niveau 4 : redimensionnement automatique et notification): Il est responsable de la collecte des informations générées par le module "modèle de performance" afin de déclencher (envoyer) des alertes et des notifications aux administrateurs Cloud via plusieurs stratégies de distribution d'alertes (messagerie, Interface Web, logs). En plus, redimensionner automatiquement des ressources via les composants de l'environnement Cloud. Ce module sera détaillé dans la section 4.5.5.

4.5.1 Le module "multi-level Monitor"

Il est composé de plusieurs outils de monitoring situés sur plusieurs couches de la plate-forme du cloud. A savoir des outils situés sur chaque nœud physique et qui génèrent et mesurent des ressources physiques et virtuelles du Cloud sur lesquelles les services grille sont déployés (exemple : CPU, RAM) et des outils situés sur les machines virtuelles et qui génèrent et mesurent des métriques de haut niveau liées au comportement de l'application (exemple : Temps de réponse,..). La figure 4.8 illustre l'architecture du module.

4.5.1.1 Les composants du Module

- **Des agents de monitoring "Probes de bas niveau"** : L'agent est responsable de la mesure et la capture continuelle des métriques de monitoring de bas-niveau (CPU, Mémoire, Disque, Réseaux). Chaque agent est localisé sur un nœud physique et il est capable de mesurer les ressources physiques de ce nœud, et les ressources virtuelles des machines virtuelles qui s'exécutent sur ce nœud. les métriques capturées par l'agent sont envoyées au deuxième module responsable de la collecte et la sélection pertinente des métriques.

- **Des agents de monitoring "Probes de disponibilité" :** L'agent est responsable du contrôle des images des machines virtuelles et les niveaux de service disponibles. Un service est disponible seulement si les utilisateurs peuvent y accéder comme prévu.Cet

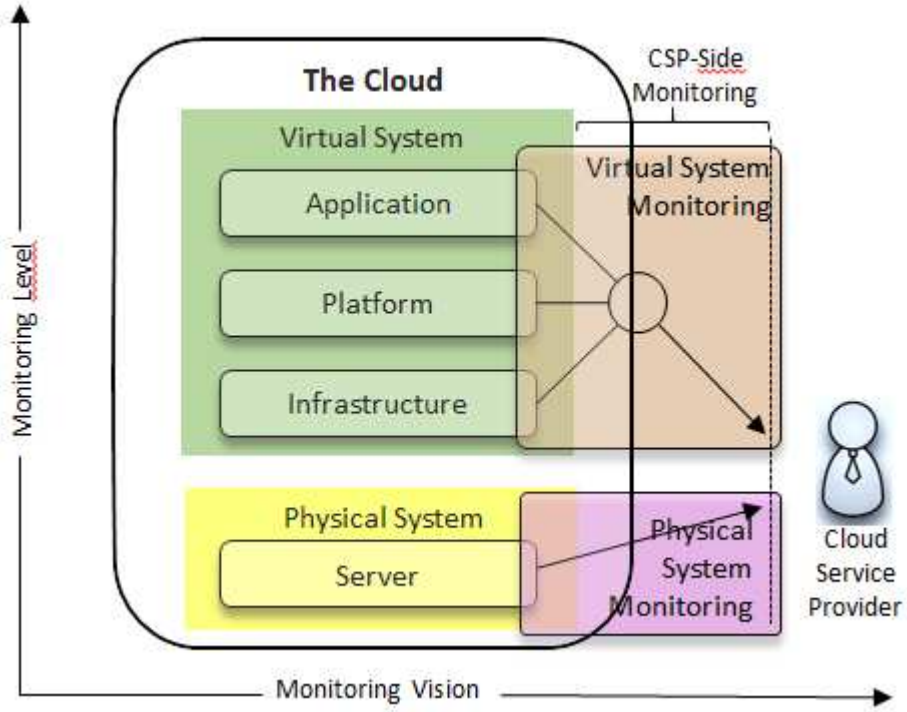

Figure 4.8: Architecture du module “Multi-Level Monitor”

agent est localisé sur un nœud physique et il est doté d'un outil de configuration qui lui permet de fournir le monitoring des services installés sur les machines virtuelle sans intégrer un agent sur chaque machine virtuelle (visualisation du nœud physique avec le groupe des VMs déployées sur ce nœud) . La mise en œuvre d'un monitoring de virtualisation efficace avec l'agent offre les avantages suivants:

- Visibilité de la disponibilité du serveur, des services, et de l'application;
- Détection rapide des serveurs et des défaillances du système d'exploitation;
- Détection rapide des défaillances de services et d'applications;
- Capacité de surveiller les indicateurs suivants : VM Statut, services,etc..
- Notification.

- **Des agents de monitoring “Probes de haut niveau”**: l'agent est responsable de la mesure et la capture continuelle des métriques de monitoring de haut-niveau (temps de réponse, nombre de requêtes, charge de travail). Chaque agent est localisé sur une machine virtuelle VM et il est capable de mesurer les métriques liées au comportement de l'application, les métriques capturées par l'agent sont envoyées aux deuxième module responsable de la collecte et la sélection pertinente des métriques.

4.5.2 Le module "Data Collector and Selector"

Il est situé au niveau le plus proche de la source des métriques, c.à.d. sur chaque noeud physique (et chaque machine virtuelle) de l'infrastructure du Cloud. Il est responsable de la collecte des métriques des ressources virtuelles et physiques du Cloud sur lesquelles les services grille sont déployés ainsi que les métriques liées au comportement des applications, puis de la sélection des métriques qui seront envoyées pour un traitement sur le prochain module.

4.5.2.1 Les composants du Module

Le module "*Data Collector and Selector*" est constitué de plusieurs composants travaillant en collaboration pour effectuer les tâches, ces composants sont :

- **Collecteur de métriques "Data Collector"** : Le programme spécifique "Data Collector" permet de collecter périodiquement les métriques mesurées par les agents situés sur plusieurs niveaux dans le cloud. On peut distinguer deux types de collecteurs:

  1. **Application Level Data Collector**: Situé dans les machines virtuelles de l'environnement Cloud (voir figure 4.9). Ce composant collecte les informations de monitoring du niveau haut des applications qui s'exécutent dans l'environnement virtuel de l'infrastructure Cloud.
  2. **Virtual and Infrastructure Level Data Collector** : Principalement, ce composant collecte les données du monitoring de l'infrastructure physique et virtuelle. Les données sont retournées par le système dans un document XML.

- **Sélection des métriques "Data Selector"**: C'est un programme qui nous permet de sélectionner les métriques et que le modèle de performance a besoin pour la prise de décision (voir la section 4.5.4.6).

4.5.2.2 Métriques de performance des applications cloud

Pour évaluer la performance des applications cloud, il faut définir des métriques quantitatives qui expriment la QOS.

1. **Métriques de disponibilité**: La disponibilité est l'attribut de qualité le plus important dans la QOS. Un service est disponible seulement si les utilisateurs peuvent y accéder comme prévu.

2. **Métriques de performance**: Un service est caractérisé par sa performance qui indique la rapidité du traitement des requêtes. Nous distinguons deux catégories de métriques de performance:

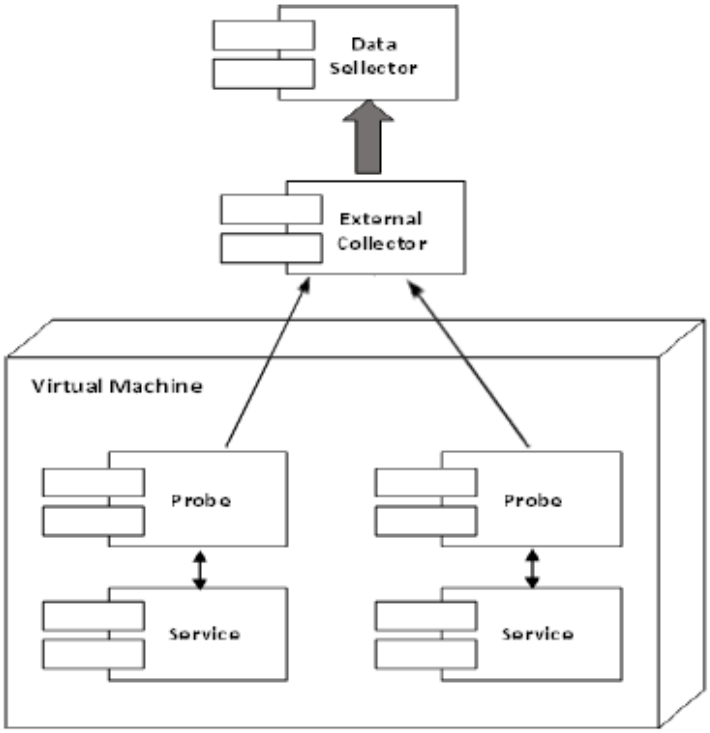

Figure 4.9: Diagramme d'Architecture du Collecteur Niveau Application

- **Les métriques de bas niveau** (métriques de ressources): Concernent les métriques de performance de niveau infrastructure: CPU, RAM, disque et de bande passante et réseau, elles mesurent la performance des composants matériels.
- **Les métriques de haut niveau** (métriques de service) : concernent la performance au niveau de l'application: Le débit, la charge de travail (workload), le temps de réponse et la latence.

Ghanbari et al [53] propose une liste de métriques de performance. Le tableau 4.1 résume les métriques utilisées:

Table 4.1: Métrique pertinentes [53]

Catégorie	**Métriques**	**Détails**
Hardware	CPU Utilization	Utilisation moyenne du CPU en pourcentage par codes d'application
	CPU Max	Utilisation pic du CPU en pourcentage pour le total
	CPU Total	Utilisation moyenne du CPU en pourcentage pour le total
	RAM used	Percentage of used space in physical memory and paging space
	RAM libre	Pourcentage de la mémoire libre
	RAM total	La mémoire totale

Table 4.1: Métrique pertinentes [53]

Catégorie	**Métriques**	**Détails**
	DISK used	Pourcentage d'espace disque utilisé
	Network interface access	Accès à l'interface réseaux
OS Process	Cpu-time	Temps CPU
	Real-memory (resident set) size	Mémoire occupée par le processus (quantité de mémoire physique utilisée par le processus)
Application server	Total threads count	Nombre de threads total
	Active threads count	Nombre de threads actifs
	Used memory	Mémoire utilisée
	Processed requests, pending requests	requêtes traités, requêtes en attente
	Dropped requests	Requêtes abandonnées
	Response time	Temps de réponse
Message Queue	Average number of jobs in the queue	Nombre moyen de jobs dans la file d'attente
	Average jobs queing time	Moyenne de temps d'attente de job.

4.5.2.3 Fonctionnement interne du module

Le fonctionnement du module "Data Collector and Selector" peut suivre les étapes suivantes pour la collecte et la sélection des métriques mesurées. Le diagramme de séquence 4.10 montre les étapes du fonctionnement de ce module:

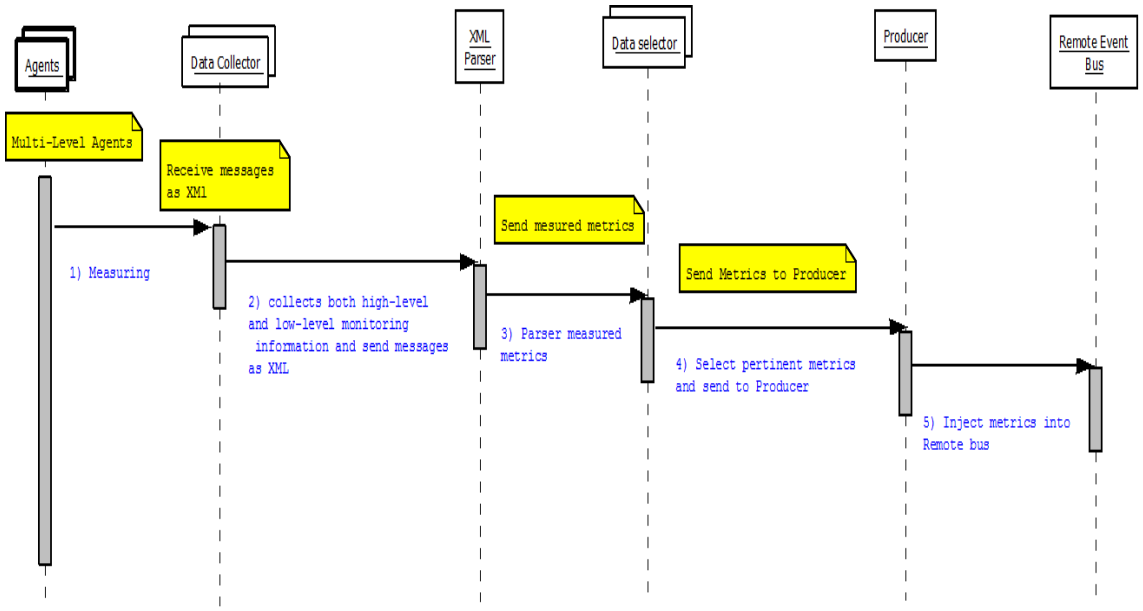

Figure 4.10: Diagramme de séquence du Module "Collection et Sélection des métriques"

1. **Etape 1** : Collecter par le programme “Data Collector” périodiquement les métriques mesurées à partir des agents de monitoring multi couches et les envoyer en tant que message XML.

2. **Etape 2** : Analyse des messages XML reçus et extraction des informations (valeurs de monitoring) par un analyseur XML spécifique et les envoyer pour une sélection pertinentes.

3. **Etape 3** : Faire une sélection périodiquement des métriques mésurées par le programme “Data selector”,et qui sert comme entrée dans le module de performance, qui sera détaillé dans la section 4.5.4.

4. **Etape 4** : injection des métriques traitées par le programme “Producer” sur le bus de communication vers le module “modèle de performance” pour un traitement.

4.5.3 Le module “modèle de communication”

Les différents modules “Data Collector and Selector” situés sur les noeuds physiques de l’environnement Cloud et le module “modèle de performance” échangent un volume considérable de métriques entre eux, pour cela un mécanisme de communication fiable est nécessaire. Ce module est doté d’un modèle de communication basé sur un bus fournissant la communication qui permet au module “modèle de performance” de recevoir des métriques des différents modules “Data Collector and Selector”. Il permet de contrôler, de dispatcher les métriques mesurés et filtrées. La figure 4.11 montre un scénario de communication entre les deux modules via le bus de communication. Le bus de communication fait passer les métriques à l’autre module en utilisant un processus de file d’attente “Queuing”, qui permet un mécanisme d’envoi fiable et efficace.

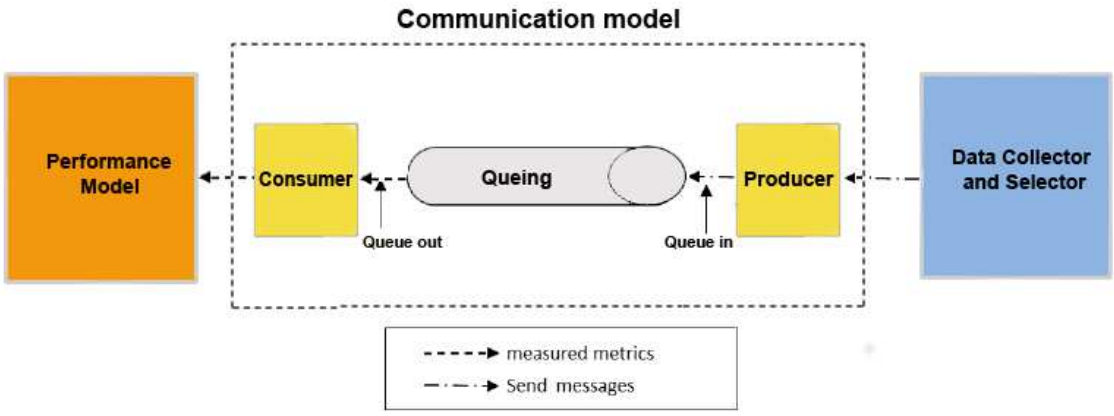

Figure 4.11: Mécanisme de communication entre les modules

4.5.3.1 Les composants du Module

Ce module est doté d'un modèle de communication, qui suit une approche producteur-consommateur. Il est constitué de deux composants à savoir :

Envoi des métriques et mécanisme de communication "Producer" - Sur chaque noeud physique où les modules "Data Collector and Selector" sont situés, un composant *Producer* est situé, qui agit comme un producteur de messages, envoi les métriques collectées, parsées et filtrées, chacune sur une file d'attente spécifique sur le bus de communication qui fait passer les métriques au module "Modèle de performance" en utilisant un processus de file d'attente "Queuing".

Réception des métriques et mécanisme de communication "Consumer" - Pour recevoir les métriques de monitoring envoyées par le module "Data Collector and Selector", un composant *Consumer* est situé, agit comme un consommateur de messages, en consommant les messages dans l'ordre de leur réception à partir du bus de communication qui utilise un processus de file d'attente "Queuing".

4.5.4 Le module "modèle de performance"

Les modèles de performance fournissent la capacité de prédire la performance de l'application pour un ensemble donné de ressources matérielles. Prédire précisément la performance de l'application pour un ensemble donné de ressources aide dans la gestion efficace de l'infrastructure informatique de haute performance, dans notre cas l'application scientifique.

Tout modèle de performance a besoin d'un "*bon système de monitoring*" qui collecte différentes métriques décrivant l'état actuel du système et de l'application, et à une granularité appropriée (par exemple par seconde, par minute).

Notre modèle de performance se base sur les approches ayant employé la modélisation de la performance de l'application tels que les modèles de prédictions quantitatives, plus précisement "l'analyse des séries chronologiques" (vue dans la section 3.2.3.4). Plusieurs méthodes sont proposées dans la littérature pour la prévision des valeurs [74, 114]. Nous distinguons les méthodes basées sur la moyenne, l'apprentissage automatique et le filtrage par motif. Dans ce qui suit, nous nous concentrons sur les techniques de prévision de *"La moyenne mobile"* nommées aussi *"Moving Average"* (vue dans la section 3.2.3.4). La figure 4.12, présente un diagramme d'activité de notre modèle de performance.

4.5.4.1 Motivations du modèle proposé

Les objectifs de ce module se concentre sur la façon de développer un modèle de performance pour les applications scientifiques déployées sur un Cloud Computing, qui définit les concepts de performance et leur relation pour concevoir un modèle d'analyse pour une prise de décision de mise à l'echelle . Plus spécifiquement, les objectifs de notre modèle sont:

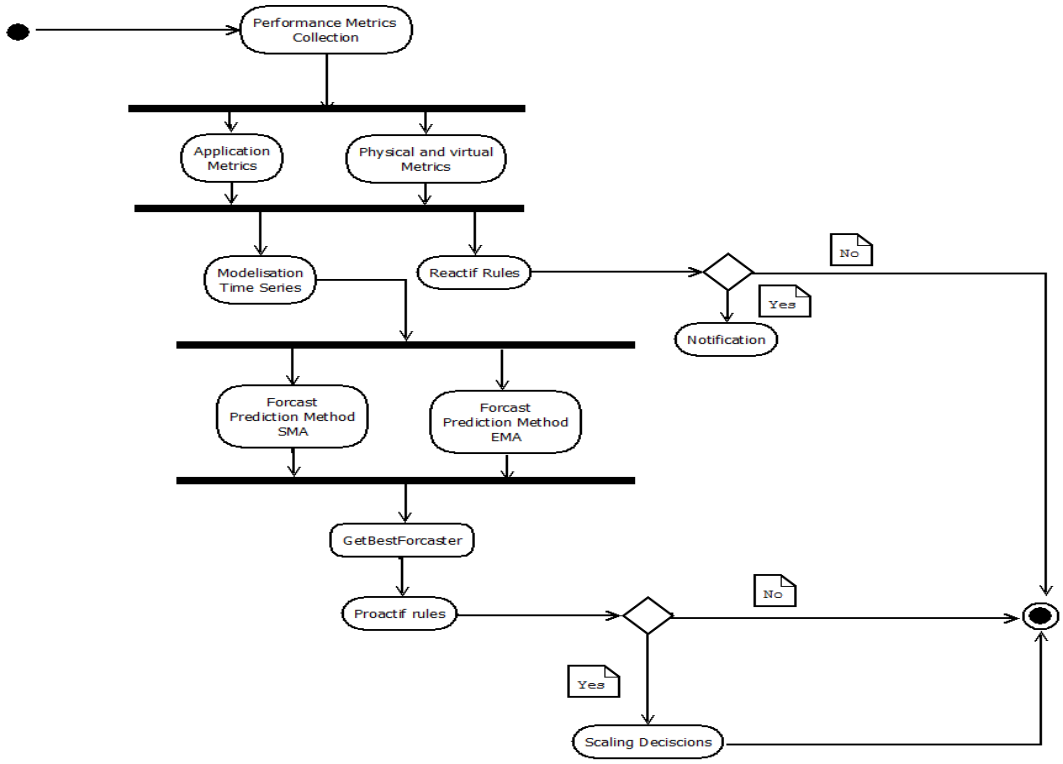

Figure 4.12: Diagramme d'activité du modèle de performance

- Quel est le processus de mesure pour analyser la performance de l'application dans un environement Cloud?
- Quelle est l'approche de modélisation qui répond mieux à nos besoins?
- Quelles sont les caractéristiques du systéme Cloud Computing qui sont liés à la performance de l'application?
- Comment le modèle de performance peut être utilisé dans la pratique pour analyser la performance, afin d'améliorer l'application dans un environnement Cloud au sein d'une organisation?

4.5.4.2 Méthodologie

Une phase de définition consiste à identifier le problème de notre modèle, et les solutions possibles à explorer. Le tableau 4.2 montre les éléments de la phase de définition du probléme.

Table 4.2: Les Eléments de la phase de définition du modèle de performance

Motivation	Objectif	Proposition
- Explorez comment mesurer la performance des applications dans le cadre du Cloud Computing	- Proposer un modèle de performance qui peut identifier les principaux facteurs qui affectent la performance des applications Cloud - Utiliser une méthode de mesure qui permet de quantifier les caractéristiques de qualité de service qui sont liées à la performance	- Faire une estimation de la future charge de travail ou une utilisation de ressource en se basant sur des approches de modélisation de performance d'application et décider de l'action appropriée pour garantir une certaine performance.

4.5.4.3 Les composants du Module

Le problème de modélisation de la perfomance d'une application peut être divisé en deux parties: La première partie, consiste à faire une estimation de *la future charge de travail* ou une *utilisation de ressource* en se basant sur des approches de modélisation de performance d'application (voir la section 3.2.3). Sur la base de cette valeur prédite, la deuxième étape consiste à décider de l'action appropriée à prendre pour garantir une certaine performance. Le module "modèle de performance" est constitué de plusieurs composants (voir la figure 5.10) travaillant en collaboration pour l'analyse et le traitement des métriques mesurées, afin de prédire des valeurs futures et prendre des décisions, ces composants sont :

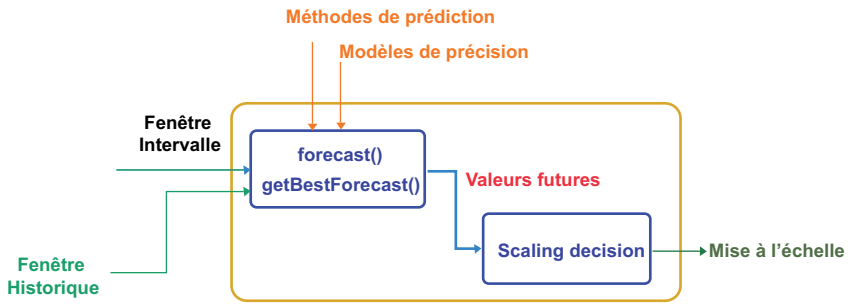

Figure 4.13: Architecture du modèle de performance

- **Approche de modélisation**: L'étude d'une série chronologique permet d'analyser, de décrire et d'exploiter un phénomène au cours du temps. Dans la littérature, les techniques des séries chronologiques ont été appliquées principalement pour la prévision de la charge de travail ou l'utilisation des ressources [74], ce qui répond mieux à notre

objectif. La prédiction se base sur des méthodes de prédiction (ForecastingMethod) et un modèle de précision (ForecastingAccuracy) qui permet de sélectionner les meilleures valeurs prédites issues des méthodes de prédiction. Parmi les méthodes de prédiction *"La moyenne mobile"* nommées aussi *"Moving Average"* . L'approche sera détaillée dans la section 4.5.4.5.

- **La prise de décision**: Aprés la prédiction, des régles de décision doivent être définies afin de déclencher le dimensionement automatique (ajout des VMs) et faire des notifications aux administrateurs. Dans notre approche, nous avons combiné entre deux types de règle, à savoir des règles proactives et des règles réactives, afin d'améliorer la performance de notre approche. Cette partie sera détaillée dans la section 4.5.4.6.

4.5.4.4 Fonctionnement interne du module

Le fonctionnement du module "modèle de performance" est expliqué dans le diagramme de séquence 4.14, et il suit les étapes suivantes:

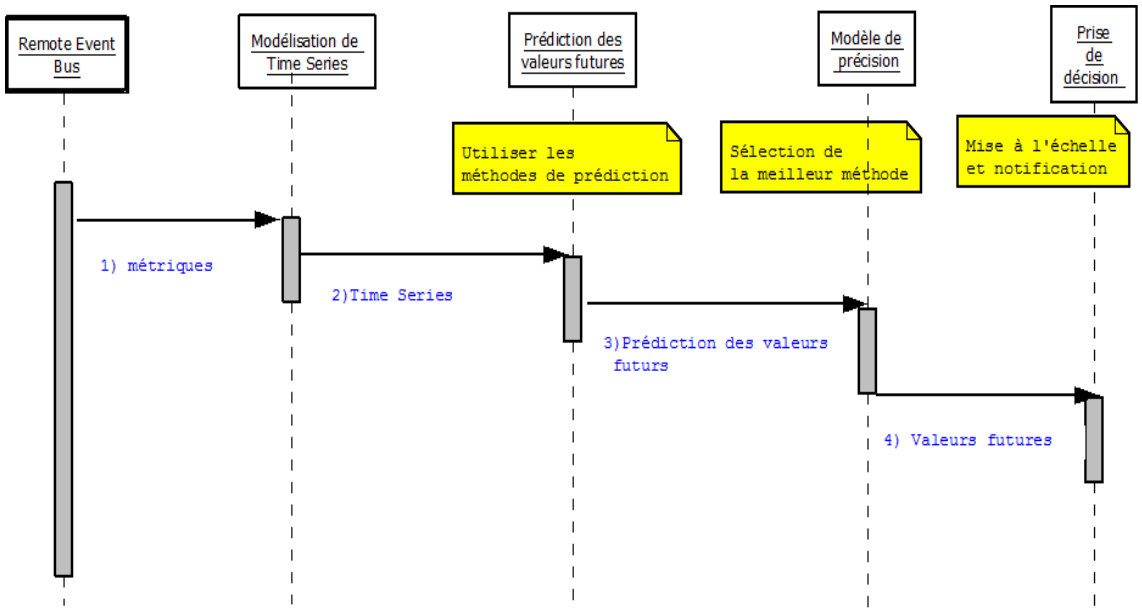

Figure 4.14: Diagramme de séquence du Module "modèle de performance"

1. **Etape 1**: Modélisation d'une série chronologique qui a comme fenêtre de prédiction (fenêtre historique) les métriques collectées du monitroing;

2. **Etape 2** : Prédiction des valeurs futures en utilisant les méthodes de prédiction. L'indicateur utilisé est "la moyenne mobile".

3. **Etape 3**: **Sélection de méthode de prédiction:** Elle consiste à appliquer les différentes méthodes de prédiction puis sélectionner la meilleure méthode qui a la plus petite valeur d'erreur de prédiction en se basant sur des modèles de précision.

4. **Etape 4** : Après la prédiction, des règles de décision sont définies pour déclencher un dimensionement automatique (ajout des VMs) et des notifications aux administrateurs.

4.5.4.5 Approche de modélisation

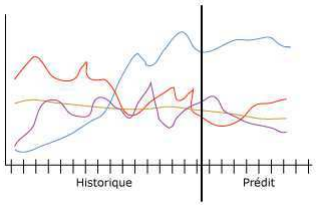

Figure 4.15: Prédiction des valeurs futures

Modélisation d'une série chronologique: Nous utilisons les séries chronologiques pour la prédiction des valeurs futures (intervalle de prédiction) à partir des valeurs observées (fenêtre de prédiction). Cette analyse est illustrée par la Figure 4.15. Les informations sur l'historique apparaissent à gauche du trait vertical alors que les prévisions effectuées apparaissent à droite du trait vertical.

Dans notre approche les métriques colléctées issues de plusieurs sources , seront échantillonnées périodiquement à des intervalles fixes (par exemple chaque minute). Le résultat pour chaque métrique sera une time-series X contenant une séquence des dernières observations w:

$$X = x_t + x_{t-1} + x_{t-2} + ... + x_{t-w+1}$$

Des techniques de prévision peuvent être appliquées soit pour la prédiction de charge de travail ou l'utilisation des ressources. Basées sur les dernières observations consécutives w $(x_t, x_{t-1}, x_{t-2}, ..., x_{t-w+1})$, une valeur future y_{t+r} est prévue, qui est r intervalles à venir de la fenêtre d'entrée "input window". Certaines des techniques utilisées à cet effet dans la littérature [74, 114] sont la moyenne mobile.

Les méthodes de la moyenne mobile: Elles peuvent être utilisées pour lisser une série chronologique afin de supprimer le bruit ou pour faire des prédictions. La valeur de prévision y_{t+1} est calculée comme la moyenne pondérée des dernières valeurs w consécutives. La formule générale est la suivante: $y_{t+1} = a_1x_t + a_2x_{t-1}, ...,$ où $a_1, a_2, ..., a_w$ sont un ensemble de facteurs de pondération positifs qui doivent sommer la valeur1. Selon la façon de déterminer ces poids, plusieurs méthodes sont définies. Pour notre approche, nous avons choisi deux indicateurs techniques : SMA (Simple Moving Average "Moyenne Mobile") et EMA (Exponentiel Moving Average-Moyenne Mobile Exponentielle) que nous avons évalués.

- Simple Moving Average $MA(q)$ (SMA): qui est la simple moyenne mobile ou MA est la moyenne arithmétique des dernières valeurs w ou q, c'est à dire, il attribue des poids $1/w$ égaux à toutes les observations. Le principe de calcul de cette méthode peut être

décrit comme suit :

```
SMA = somme glissante (Valeur Métrique(i), Période)/période de la SMA
Exemple pour une SMA3:
la valeur en t de la SMA3 = (Valeur Métrique en t-2+ Valeur Métrique en t-1 + Valeur Métrique
en t )/3
```

- Exponential Smoothing (EMA): Il affecte de façon *exponentielle* la diminution du poids au fil du temps. Un nouveau paramètre est introduit, un facteur α de lissage qui affaiblit l'influence des données passées. La formule de prédiction pour le lissage exponentiel simple est:

$$
\begin{aligned}
&y_{t+1} = \alpha x_t + (1-\alpha) y_t \\
&= \alpha x_t + (1-\alpha)[\alpha x_{t-1} + (1-\alpha) y_{t-1}] \\
&= \alpha x_t + \alpha(1-\alpha) x_{t-1} + (1-\alpha)^2 [\alpha x_{t-2} + (1-\alpha) y_{t-2}] \\
&= \alpha x_t + \alpha(1-\alpha) x_{t-1} + (1-\alpha)^2 x_{t-2} + ... + (1-\alpha)^w - 1 x_{t-w+1}
\end{aligned}
$$

où y_{t+1} représente la valeur de prédiction pour la période $t+1$, x_t est la valeur à l'instant t, et y_t est la prévision faite pour la période t.

Le principe de calcul de cette méthode s'effectue en trois étapes :

1. **Le calcul initiale** :

```
SMA = somme (Valeur Métrique(i), période)/période
```

2. **Le calcul du coefficient** :

```
M = (2 / (période + 1))
```

3. **Le calcul de la moyenne exponentielle courante :**

```
EMA = (Valeur Métrique(i)* M)+(EMA(i-1)*(1-M)) ou bien
EMA = Valeur Métrique(i) - EMA(i-1)* M + EMA(i-1).
```

Les modèles de précision: Les modèles de précision permettent dévaluer les méthodes de prédiction. Intuitivement, plus l'erreur de prévision est petite et meilleure est la méthode. Parmi les modèles de précision, l'erreur absolue moyenne (Mean absolute error - MAE):

Mean Absolute error : $MAE = mean(|e_i|)$

4.5.4.6 La prise de décision

Plusieurs approches peuvent être utilisées dans la prise de décision [74]. Notre approche se base sur une prise de décision à partir d'un ensemble de règles prédéfinies. Pour ce faire, nous

avons combiné entre deux types de règle, à savoir des règles proactives et des règles réactives, afin d'améliorer la performance de notre approche.

Les règles proactives - Elles se basent sur le résultat des approches prédictives et les valeurs futures des métriques de ressources et sont utilisées pour déclencher une mise à l'échelle automatique.

Les règles réactives - Elles se réfèrent à l'ensemble des méthodes qui réagissent au système actuel et / ou l'état de l'application, et cela sur la base des valeurs obtenues à partir d'un ensemble de métriques issues du monitoring en temps réel. Les règles réactives sont des contraintes qui expriment des stratégies de notifications aux administrateurs afin de diminuer le taux d'erreurs issus du calcul des algorithmes de prédiction.

La mise à l'échelle (voir la figure 4.16), consiste à déterminer la quantité de ressources d'un système. En agissant sur la quantité de ressources du système, on contrôle sa capacité de traitement. Il peut être présenté par le quadruplet : (Quand, Comment, Combien, Où). Le "Quand" indique le moment de la mise en oeuvre : en amont (proactif), en aval (réactif) ou hybride (réactif-proactif). Le "Comment" présente le type de dimensionnement : horizontal, vertical ou les deux. Le "Combien" indique la quantité de ressources concernées (planification de capacité). Le "Où" spécifie la cible à savoir le calcul, le stockage ou la répartition de charge.

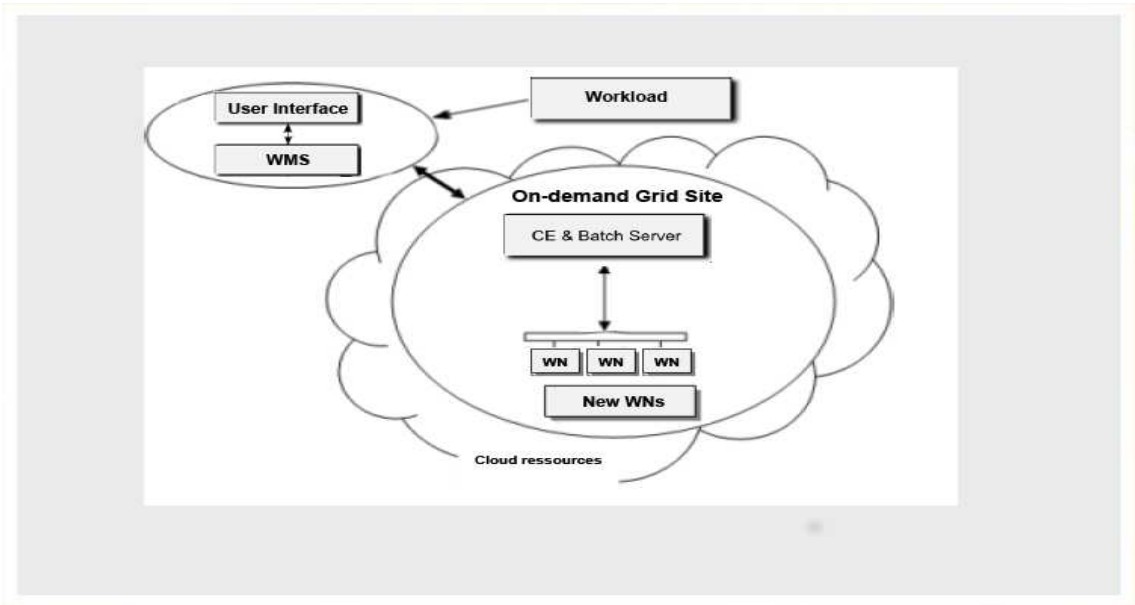

Figure 4.16: Architecture de mise à l'échelle

- **Quand** : Le dimensionnement est basé sur une prédiction de la charge dans le futur. En utilisant un service de monitoring, le système peut émettre des éléments déclencheurs pour prendre les mesures nécessaires pour augmenter ou réduire la quantité de ressources en fonction de métriques.

- **Comment** : Notre modèle de performance se base sur un dimensionnement horizontale. Un dimensionnement horizontal est la possibilité d'ajuster (augmenter (scale out) ou diminuer (scale in)) le nombre d'instances (VMs) du système à la demande.

- **Combien** : La quantité de ressources à ajouter ou supprimer applique la planification meilleur effort (Best-effort). Une maitrise est nécessaire pour utiliser cette solution. Nous identifions les règles à base des seuils. cette approche dépend fortement de choix des variables de performance à savoir les métriques.

- **Où** : présente la portée de l'action du dimensionnement. Une telle portée est définie par la nature de ressources (calcul, stockage ou répartition de charge). Notre modèle de performace se base sur les ressources de calcul et la répartition de charge.

Les règles à base des seuils: Les règles ou les politiques à base de seuil sont très populaires parmi les fournisseurs de cloud. La simplicité et la nature intuitive de ces politiques, les rendent très attrayant pour les utilisateurs. Toutefois, la fixation des seuils est une tâche par application, et nécessite une compréhension profonde des tendances des métriques et de la charge de travail.

Le nombre d'instances du système varie en fonction d'un ensemble de règles. Ces règles sont divisées en deux types : les règles pour augmenter les ressources et les règles pour diminuer les ressources. Les règles suivent en général le template suivant : une règle peut utiliser une ou plusieurs métriques telles que CPU ou le temps de réponse. Pour une règle (basée sur une métrique m) plusieurs paramètres sont impliqués : un seuil supérieur $thr_{upper,m}$, un seuil inférieur $thr_{lower,m}$ et deux périodes de temps $temps_{upper}$ et $temps_{lower}$ qui définissent la durée pendant laquelle la ***condition*** doit être remplie pour ***déclencher une action*** de dimensionnement. Les seuils supérieurs et inférieurs doivent être définis pour chaque métrique de performance x. La règle associée est définie de la manière suivante :

$$\begin{cases} si & m > thr_{upper,m} & pour & temps_{upper} & alors & faire & n = & n + k_{add} \\ si & m < thr_{lower,m} & pour & temps_{lower} & alors & faire & n = & n - k_{remove} \end{cases}$$

Où m est une métrique (CPU par exemple), k_{add} et k_{remove} sont respectivement la capacité à ajouter ou à supprimer.

Exemples de règles: L'idée dans notre approche est de fixer des objectifs de performance ou des objectifs de planification selon les besoins des applications scientifiques qui s'exécutent sur l'infrastructure Cloud. Pour ce faire, nous allons définir un ensemble de règles en utilisant soit une valeur de métrique issue directement du système de monitoring ou bien une combinaison de valeurs de métriques avec des seuils supérieur et inférieur pour chaque valeur de métrique, soit en utilisant des valeurs futures de métriques calculées à partir des algorithmes de prédiction (voir la section 4.5.4.5).

Définition de la règle 1 :

Supposons qu'on a les objectifs suivants : Ajouter une instance lorsque la future valeur moyenne de la métrique "CPU utilisation" sur une machine virtuelle VMx (qui exécute par exemple une application grille y), et qui est calculée à partir des algorithmes de prédiction, est supérieure à 70% pendant plus de 15 minutes, et libérer une instance si la valeur est Inférieur à 30%.

En se concentrant sur la mise à l'échelle horizontale, l'utilisateur peut définir un maximale $nMax$ et un minimum $nMin$ pour un certain nombre de VMs, afin de contrôler le coût global et garantir un niveau minimum de disponibilité.

A chaque itération, si la métrique de performance x atteint thr_{upper} pour $temps_{upper}$ secondes, s VMs sont demandées et ajoutées dans l'infrastructure Cloud. Si la métrique de performance x passe en dessous de thr_{lower} pour $temps_{lower}$ secondes, s VMs sont enlevées et leurs ressources sont libérées.

$$\begin{cases} if \quad Future\_VMx.vcpu\_util > 70\% \quad pour \quad temps_{upper} \quad alors \quad faire \quad n = \quad n+1 \\ if \quad Future\_VMx.vcpu\_util < 30\% \quad pour \quad temps_{lower} \quad alors \quad faire \quad n = \quad n-1 \end{cases}$$

Où : Future_VMx.vcpu_util est la valeur moyenne prédite de métrique sur l'utisation cpu de la VM

Définition de la règle 2 :

Supposons qu'on a les objectifs suivants: Signaler si un "Worker Node (WN)" n'est pas disponible pour les 15 dernières minutes, ou bien le "Load Average" du "Worker Node WN" sur les 15 dernières minutes est supérieur à 1.6. Dans ce cas de figure, la régle pourra nous indiquer si le "Worker Node" est incapable d'exécuter des jobs (soit il n'est pas disponible, soit son "Load Average" dépasse 1.6). Cette règle se base sur les valeurs issues directement du monitoring :

$$if \quad VMx.available = 0 \quad or \quad VMx.load\_five \geq 1.6 \quad then \quad Notification$$

Où : VMx.available est la valeur de métrique sur l'état de démarrage du noeud, VMx.load_five est la valeur de métrique sur le ''Load Average" du Worker Node et qui signifie une moyenne de la charge système sur une certaine période.

Définition de la règle 3 :

Les conditions dans les règles sont généralement basées sur un ou au plus deux métriques de performance, les plus populaires sont "Average CPU Load" des VMs, le temps de réponse, ou le taux de demande d'entrée.

Les objectifs dans cette règle sont comme suit: Ajouter une instance si la future valeur moyenne de "Average CPU load" sur les 30 dernières minutes est supérieur à 1.6, et la future valeur moyenne du temps de réponse d'un job est supérieur à 2 minutes.

$if \quad Future\_VMx.load\_five > 1.6 \quad et \quad FutureJob\_Response\_Time \geq 120 \quad alors \quad faire \quad n = \quad n+1$

Où : Future_VMx.load_five est la valeur moyenne prédite de métrique sur le ''Load Average" du Worker Node et qui signifie une moyenne de la charge système sur une certaine période. FutureJob_Response_Time est la valeur moyenne prédite du temps de réponse qu'un job peut prendre dans un temps futur.

Définition de la règle 4 :

Les objectifs dans cette règle sont comme suit: Ajouter une instance si la future valeur moyenne de "CPU utilisation" est supérieur à 70%, et la future valeur moyenne du temps de réponse d'un job est supérieur à 2 minutes.

$if \quad Future\_VMx.cpu\_util > 1.6 \quad et \quad FutureJob\_Response\_Time \geq 120 \quad alors \quad faire \quad n = \quad n+1$

Où : Future_VMx.cpu_util est la valeur moyenne prédite de métrique sur l'utisation cpu de la VM.
FutureJob_Response_Time est la valeur moyenne prédite du temps de réponse qu'un job peut prendre dans un temps futur.

4.5.5 Le module Mise à l'échelle et notification

Dans l'approche à base de règles que nous avons adopté, nous avons défini des régles de type ***"Event-Condition-Action"*** dans la partie prise de décision de notre approche. Une action pourrait impliquer la réalisation d'une décision de gestion globale (par exemple, ajoutant ou supprimant des worker nodes) comme montré dans la figure 4.16 , une décision de gestion locale (par exemple augmenter la mémoire d'un processeur), ou l'émission d'un événement, par exemple une alerte ou une notification.

La mise à l'échelle des ressources en réponse à des variations de charge imprévisibles est une tâche difficile. Le Cloud Computing est vu comme une grande opportunité pour lever cet obstacle, ce qui résout partiellement le problème, puisqu'il reste à pourvoir les systèmes qui les gèrent de fonctionnalités supplémentaires comme l'autoscaling qui permet de supporter ce besoin d'adaptation.

L'événement d'alerte ou de notification permet aux administrateurs de recevoir l'information avec le détail du problème et les réactions à effectuer. Pour ce faire, nous avons intégré un système de notification qui nous permet de :

1. Visualiser les alertes à travers une "interface web" dédiée via un plugin;

2. Envoyer les notifications aux administrateurs par messagerie via un programme.

3. Envoyer les graphes sous format d'histogramme issus des algorithmes de prédiction, aux administrateurs par messagerie.

4.6 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 adopté pour sa résolution. Nous avons également présenté le cas d'étude qui a motivé notre travail. Nous avons abordé tout de suite après, l'architecture globale retenue pour la solution où il est possible de distinguer quatre principaux éléments qui sont, l'environnement d'intégration "Grid-Cloud", le module de monitoring multi-niveaux, le modèle de performance et le modèle de communication.

A cet effet nous avons organisé ce chapitre en décrivant les principaux modules, approches, analyse et fonctionnement de notre solution à savoir:

1. Choix de l'approche d'intègration "Grid-Cloud", l'une des approches proposèes dans la littèrature concernant la combinaison des deux paradigmes "Grid" et "Cloud", qui a pour but de simplifier la gestion de la grille dans son modéle actuel en bénéficiant des avantages Cloud comme l'organisation autonome, la gestion simplifiée des ressources à la demande pour rencontrer les exigences des applications de hautes performances et une architecture plus èlastique, flexible et dynamique.

2. De compléter cette architecture d'intégration par un "système multi-niveaux de monitoring basé modèle de performance", et qui se compose de deux unités importantes:

   (a) Une unité de monitoring multi-niveaux qui collectent différentes mesures et des indicateurs de performance qui renseignent sur l'état des applications et de l'infrastructure (physique et virtuelle) et la compréhension et le choix précis des métriques quantitatives qui évaluent la performance des applications.

   (b) Un modèle de performance qui est divisé en deux parties: La première partie, consiste à faire une estimation de la future charge de travail ou une utilisation de ressource en se basant sur des approches de modélisation de performance d'application plus précisément "l'analyse des séries chronologiques". Sur la base de cette valeur prédite, la deuxième ètape consiste à décider de l'action appropriée à prendre avec des règles de décision qui doivent être définies afin de faire des notifications aux administrateurs ou bien déclencher le dimensionnement automatique (ajout des VMs).

Dans le chapitre suivant, nous allons présenter la validation de l'approche d'intégration "Grid-Cloud", ainsi que le système de monitoring multi-niveaux basé modèle de performance, par l'expérimentation sur un environnement réel, l'implémentation des modules composant notre système, et enfin, l'évaluation et l'analyse des résultats d'expérimentation du système sur l'environnement d'intégration "Grid-Cloud".

CHAPTER 5

Mise en oeuvre et Expérimentation

5.1 Introduction

Dans le présent chapitre, nous développons la mise en oeuvre de notre approche. La première partie est consacrée au déploiment de l'approche d'intégration "Grid-Cloud". Pour cela, Nous décrivons les environnements matériels et logiciels sur lesquels nous nous sommes appuyés pour la réalisation de nos expérimentations. Dans la deuxième partie de ce chapitre, nous présentons l'implémentation de notre système de monitoring "multi-niveaux" basé sur un modèle de performance, et qui se compose de deux unités importantes : Une unité de monitoring multi-niveaux qui renseignent sur l'état des applications et de l'infrastructure, et un modèle de performance, qui consiste à faire une estimation de la future charge de travail ou une utilisation de ressource en se basant sur des approches de modélisation de performance d'application, et décider de l'action appropriée pour une mise à l'échelle .

Pour ce faire, nous avons décrit les outils et logiciels utilisés, ainsi que l'environnement de développement dans lequel nous avons travaillé avant d'entamer les détails de cette mise en oeuvre. Nous avons choisi comme approche de modélisation "l'analyse des times series", et pour les techniques de prédiction "la moyenne mobile". Pour la prise de décision nous avons combiné des règles proactives qui se basent sur le résultat des approches prédictives et les valeurs futures des métriques de ressources, et des règles réactives qui se basent sur un ensemble de métriques issues du monitoring à temps réel, afin d'améliorer la performance de notre approche. En dernier, nous exposons l'évaluation du système "multi-niveaux basé modèle de performance" pour le monitoring des ressources, services et applications dans l'environnement d'intégration "Grid-Cloud" déployé.

5.2 Déploiement de l'approche d'intégration "Grid-Cloud"

5.2.1 Environnement de déploiement

Le déploiement de l'approche d'intégration "Grid-Cloud" choisie (l'approche grille sur le Cloud) est une intégration des services grille sur une plate-forme Cloud type IaaS. Nous avons choisi de tester l'intégration des services grille, plus précisément les services de calcul de la grille nationale "DZ e-Science GRID"[1] avec Cloud IaaS "OpenStack".

La grille Algérienne "DZ e-Science GRID" est une infrastructure qui gère des ressources de stockage et de calcul et tous les services de base (VOMS, WMS, IS, CE, WNs,...etc) en utilisant le middleware EMI. Elle est administrée et gérée par l'équipe de la Technologie des Systèmes Collaboratifs et Intégrés de la division Réseaux du Centre de Recherche sur l'information Scientifique et Technique. La grille "DZ e-Science GRID" participe au projet européen "Eumedgrid [2]", qui vise à développer dans le bassin Méditerranéen une infrastructure de grid pour la Recherche. Actuellement, les services grille dans la grille "DZ e-Science GRID" sont soit sous forme de ressources virtuelles ou sous forme de ressources physiques.

Le projet OpenStack est une plate-forme Cloud Computing open source qui prend en charge tous les types d'environnements de cloud et vise à la mise en œuvre simple, une extensibilité massive et un riche ensemble de fonctionnalités. D'autre part, OpenStack fournit une solution Infrastructure-as-a-Service (IaaS) à travers une variété de services complémentaire et il est hautement configurable pour répondre aux différents besoins avec divers options de calcul, mise en réseau, et de stockage. Nous avons eu recours à différents composants logiciels et environnementales afin de pouvoir intégrer les composants de l'approche d'intégration choisie. Le tableau suivant 5.1 résume cet environnement de déploiement:

5.2.2 Etapes de déploiement de l'approche d'intégration "Grid-Cloud"

Une approche de Cloud IaaS est adaptée pour construire et administrer un système grille flexible, et dans cette approche, le "Middleware" des services grille s'exécute sur des machines virtuelles de l'infrastructure Cloud. Le déploiement de l'approche d'intégration "Grid-Cloud", consiste dans notre cas à intégrer des services de calcul (CE et Worker Nodes) avec le système d'administration Cloud "OpenStack". Ces services de calcul intégrés, "Computing Element (CE)" nécessaires à la gestion des "Worker Nodes(WN)" pour l'exécution des "jobs", sont liés à la grille "DZ e-Science GRID" (voir figure 5.1), et peuvent être gérés par le "Grid Site Manager" à travers la grille, et par le "Cloud Manager" à travers l'infrastructure Cloud. Afin d'intégrer cette approche, nous avons suivi les étapes suivantes :

[1] http://www.grid.arn.dz
[2] http://www.eumedgrid.eu

Table 5.1: Environnement de déploiement de l'approche "Grid-Cloud"

		Approche d'intégration	**Environnement de déploiement**
User Layer		- Administrateur Cloud "Cloud Manager"	- Administrateur Cloud "OpenStack"
		- Administrateur Grille "Grid Manager" et Administrateur VO "VO Manager"	- Administrateur de la grille "DZ-eScience GRID" et Administrateur VO "DZ-Grid"
		- Utilsateurs finaux "End Users"	- Utilisateurs finaux de la grille nationale "DZ-eScience GRID"
Multi-Level Monitoring System	**Service Layer**	Les services Grille	Les services de la grille nationale "DZ-eScience GRID"
		Modèle d'image	- Système d'exploitation Linux (Redhat Entreprise Linux) - Hyperviseur de virtualisation (KVM) - Middleware (gLite-EMI) - Ensemble d'applications requit pour la mise en fonction d'un service grille donné (CE, WNs,...etc)
	Infrastructure Layer	VMs + "VM Manager" (Cloud layer)	Le système de gestion Cloud IaaS "OpenStack" + définition de la template de VM sur "OpenStack"
		Hyperviseur (Compute)	Hyperviseur "KVM" + Nova (application)
		Object Storage	Swift (stockage d'objet)
		Image Service	Glance (service d'image)
		Dashboard	Horizon (interface Web de paramétrage et gestion)
		Réseau (Network)	Neutron (gestion des réseaux à la demande): Réseau public pour l'infrastructure physique + réseau privé et public pour l'infrastructure virtuelle
		Storage	Cinder (service de disques persistants pour les machines virtuelles)
		Orchestration	Heat (service d'orchestration à base de template)

1. **La mise en place du Cloud IaaS "OpenStack"** : Comme première étape, un Cloud de type IaaS privé "OpenStack" est mis en place pour permettre la gestion des nouveaux services de calcul liés à la grille "DZ e-Science GRID" et déployés sur l'infrastructure Cloud selon l'approche d'intégration "Grid-Cloud". La mise en place de cet environnement sera détaillée dans la section 5.2.3.

2. Définition des Appliances Virtuelles et Templates pour les services de calcul de la grille "DZ e-Science GRID";

3. **Virtualisation des services de calcul** : La troisième étape consiste au déclenchement du processus de démarrage de plusieurs instances de VM à partir de ces Appliances Virtuelles à travers le système de gestion Cloud IaaS "OpenStack".

Actuellement les services de calcul (CE et Worker Nodes) résident dans la grille "DZ e-Science GRID" sur des ressources physiques et sont gérés par le middleware grille EMI.

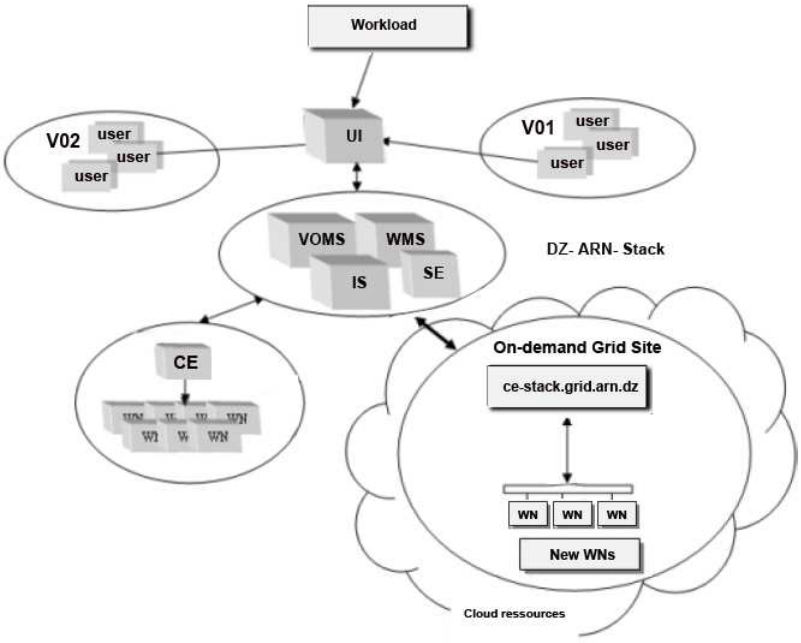

Figure 5.1: Architecture de l'approche d'intégration

5.2.3 Mise en oeuvre du Cloud IaaS "OpenStack"

5.2.3.1 Préparation des machines du Cloud d'OpenStack

Pour la mise en oeuvre du Cloud IaaS "OpenStack", nous avons utilisé l'architecture à "plusieurs noeuds[3]" avec OpenStack Networking (neutrons) et les nœuds optionnels pour les services "Block Storage" and "Object Storage". Pour cela, nous avons installé un noeud controller et deux noeuds compute (voir figure 5.2), afin de fonctionner tous les services OpenStack, y compris les services de contrôle, de Réseau, de calcul et de stockage. Notons que, la plateforme du Cloud "OpenStack" installée est "Juno[4]".

Pour l'installation du noeud Controller, nous avons utilisé une machine "openstack.grid.arn.dz". La machine est un serveur HP disposant de 5GB de mémoire, de 300Giga de disque et un système d'exploitation "Centos 7 [5]". La machine contrôleur gère le service Identity, le service Image, des parties de gestion du noeud compute, une partie de gestion Networking, divers agents Networking, le service dashboard, ainsi que le service SQL database.

Pour l'installation des deux noeuds compute, nous avons mis en place deux noeuds physiques (compute1.grid.arn.dz et compute2.grid.arn.dz), compute1 dispose de 4GB de mémoire, de 2 processeurs CPU de type Intel Core 2 Duo de 3.00GHz et compute2 est

[3] http://docs.openstack.org/liberty/install-guide-rdo/overview.html
[4] http://docs.openstack.org/juno/install-guide/install/apt/content/ch_overview.html
[5] https://www.centos.org/

un serveur HP disposant de 10GB de mémoire, de 2TB de disque et de 2 processeurs CPU (4 cores). Le nœud Compute exécute une partie de l'hyperviseur du Compute qui fonctionne les instances et utilise l'hyperviseur KVM. Le nœud Compute gère également un agent de service Networking qui relie les instances à des réseaux virtuels.

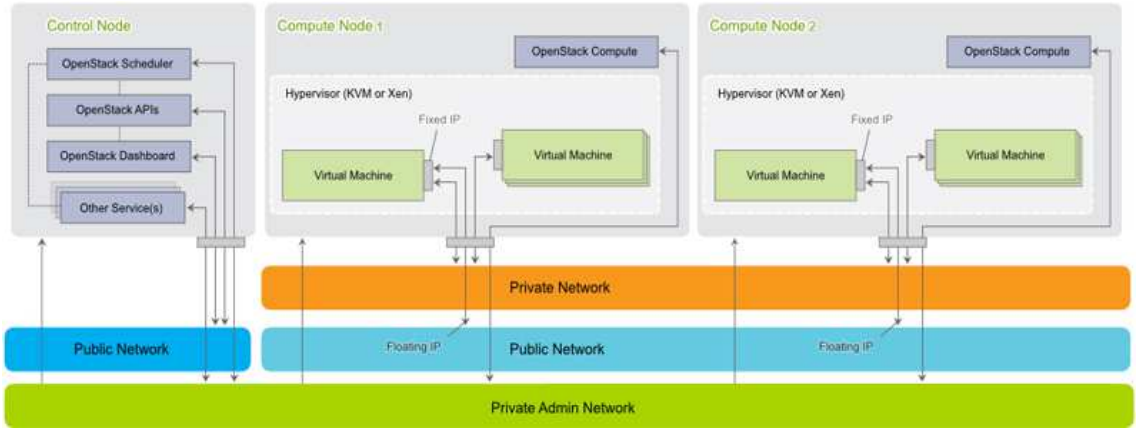

Figure 5.2: Architecture de l'environnement de test

5.2.3.2 Définition des Appliances Virtuelles et Templates

Nous avons définit une appliance virtuelle pour chaque service de calcul grille (CE et WNs). Chaque appliance virtuelle contient une image disque de VM configurée avec le système d'exploitation "Linux", le middleware grille "EMI" utilisé par tous les services de la grille "DZ e-Science GRID" ainsi qu'un ensemble d'applications requit pour la mise en fonction d'un service de calcul.

Le Service Image d'OpenStack (glance), nous permet d'enregistrer et de récupérer des images des machines virtuelles. A cet effet, il dispose d'une API REST qui permet d'interroger les métadonnées d'image virtuelle de la machine et de récupérer une image réelle. D'autre part, on peut stocker des images de machines virtuelles mises à disposition par le Service Image et par le service Object Storage d'OpenStack.

Nous illustrons dans cet exemple les étapes à suivre pour une création d'une image d'un service grille "Worker Node" après la préparation de l'appliance virtuelle dans notre environnement Cloud "OpenStack" :

1. Création et modification dans un répertoire local temporaire;
2. Téléchargement de l'image dans un répertoire local temporaire;
3. Source des informations d'identification d'administration pour avoir accès à l'admin uniquement par les commandes CLI;
4. Ajout de l'image avec le service Image.

5.2.3.3 Instanciation des services de calcul

Aprés la définition des appliances virtuelles pour chaque service de calcul et la création de l'image disque, vient la phase de virtualisation qui consiste à lancer une instance à partir de l'appliance. Le lancement d'une instance se fait avec OpenStack Networking (neutron). Une instance est une machine virtuelle que OpenStack provisionne sur un nœud Compute.

L'exécution d'une instance passe par plusieurs étapes, notons que cette opération peut se faire soit par l'interface "Dashboard" d'OpenStack ou bien par ligne de commandes, ou par le lancement d'une template déjà définie à partir du service "Heat":

1. Activation de SSH et ICMP sur notre groupe de sécurité par défaut en utilsant le service horizon via l'interface graphique ou par ligne de commande;
2. Création ou importation d'une paire de clés;
3. Lancer l'instance: Pour lancer une instance, on doit au moins préciser un gabarit de VM (flavor), le nom de l'image, le réseau, le groupe de sécurité, la clé, et le nom de l'instance. Le flavor spécifie un profil d'attribution de ressources virtuel qui comprend un processeur, la mémoire et le stockage. Cette opération se fait soit via le dashboard ou bien par ligne de commande;
4. Associer une adresse IP flottante.

5.2.4 Tests d'intégration du site "DZ-ARN-Stack" avec le Cloud "OpenStack"

Pour les tests d'intégration "Grid-Cloud", nous avons travaillé sur un nouveau site grille "DZ-ARN-Stack" qui contient un service "CE" et plusieurs services grille "WNs" (worker nodes) associés. Le site est intégré avec le cloud "OpenStack" suivant l'approche d'intégration "Grid-Cloud". L'intégration de service CE et les WNs associés passe par les étapes d'intégration déja expliquée dans les sections précédentes 5.2.3.

5.2.4.1 Intégration du service grille "CE-Stack"

Avec le déploiement actuel de la grille "DZ e-Science GRID", le service grille CE réside sur des ressources physiques. L'intégration du service "CE-Stack" du nouveau site "DZ-ARN-Stack" lié à la grille "DZ e-Science GRID" dans l'infrastructure Cloud "OpenStack" passe par les étapes suivantes :

1. Définition de l'appliance virtuelle et la template pour le service grille "CE-Stack": Pour cela nous avons créé une image VM avec l'installation d'un système d'exploitation "Scientific Linux 6", l'installation du Middleware "EMI3[6]" de la grille "DZ e-Science GRID", et l'installations des paquetages spécifiques au service grille CE.

[6] https://twiki.cern.ch/twiki/bin/view/EMI/GenericInstallationConfigurationEMI3

2. Ajout de l'image du service "CE-Stack" dans l'environnement Cloud "OpenStack" avec le service Image (glance).

3. Déclenchement du processus de démarrage d'une machine virtuelle sur le Cloud "openstack.grid.arn.dz" à partir de l'image VM en précisant un modèle de VM (flavor), le nom de l'image, le réseau, le groupe de sécurité, la clé, et le nom de l'instance.

4. Configuration du service "CE-Stack" intégré dans le cloud, et le lier à la grille "DZ e-Science GRID" pour l'environnement hybride.

5.2.4.2 Intégration des services grille "WN-Stack"

Aussi, les services grille "Worker Nodes (WNs)" associés au "CE-Stack" doivent suivre des étapes basées sur l'infrastructure Cloud "OpenStack" :

1. Définition d'une appliance virtuelle pour le service grille "WN-Stack": Pour cela nous avons créé une image VM avec l'installation du système d'exploitation "Scientific Linux 6", l'installation du Middleware "EMI" (EMI 3) et l'installations des paquetages spécifiques au service grille WN.

2. Ajout de l'image du service "WN-Stack" dans l'environnement Cloud "OpenStack" avec le service Image (glance).

3. déclenchement du processus de démarrage d'une machine virtuelle sur le Cloud "openstack.grid.arn.dz" à partir de l'image VM en précisant un modèle de VM (flavor), le nom de l'image, le réseau, le groupe de sécurité, la clé et le nom de l'instance.

4. Configuration du service "WN-Stack" intégré dans le cloud et le lier au "CE-Stack" pour l'environnement hybride.

Aprés la création du nouveau site "DZ-ARN-Stack" ("CE-Stack" avec les "WNs-Stack" associés), nous avons obtenu une instance de machine virtuelle pour le service grille CE et plusieurss instances de machines virtuelles pour le service grille WN (voir le tableau 5.2), et qui sont déployées sur les noeuds computes. Ces instances représentent le nouveau site "DZ-ARN-Stack" des services grille directement déployés sur le Cloud (figure 5.3).

Table 5.2: Déploiement des instances sur l'infrastructure Cloud "OpenStack"

VM Image	Instance	Storage	Host
CE-Appliance	CE-Stack	20GB	c1.grid.arn.dz
WN-Appliance	WN01-Stack	20GB	c1.grid.arn.dz
WN-Appliance	WN02-Stack	20GB	c2.grid.arn.dz

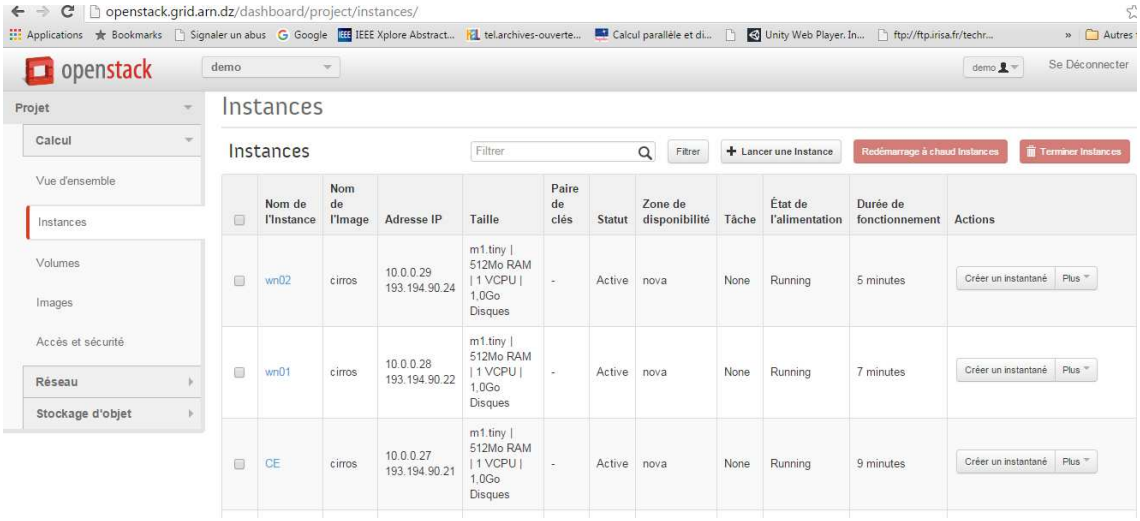

Figure 5.3: Liste des instances VM déployées suivant l'approche "Grid-Cloud"

5.3 Implémentation du système "Multi-Level basé modèle de Performance"

Dans cette section, nous présentons l'implémentation de notre système de monitoring "Multi-Level" basé sur un modèle de performance et qui est composé de trois principaux modules: Un module de monitoring multi-niveaux, un modèle de performance et le modèle de communication. La figure 5.4 présente le schéma général de la mise en oeuvre de notre approche.

Pour cela, nous décrivons les principaux outils et logiciels Open-source que nous avons intégré ensemble afin d'implémenter un monitoring multi-niveaux (ressources, services et applications), l'approche de modélisation de la performance des applications "analyse des times series", les techniques de prédiction pour la prise de décision, et l'approvisionnement des ressources en utilisant le service d'autoscaling que propose "OpenStack", afin de garantir un calcul de haute performance des applications scientifiques, et qui s'exécutent dans un environnement "Grid-Cloud".

5.3.1 Choix des logiciels et outils du monitoring multi-niveaux

5.3.1.1 Les agents de monitoring de Ganglia

Pour le fonctionnement de notre système de monitoring "multi-level basé modèle de performance" nous avons installé "Ganglia[7]" [77] qui est un système de surveillance largement utilisé pour les systèmes de calcul de haute performance, telles que les clusters ou les grilles de calcul. Cependant Ganglia n'est pas destiné à être utilisé pour la surveillance de ressources virtuelles et plus précisement dans un environement Cloud, pour cela nous l'avons combiné avec le protocol "sFlow[8]".

[7] http://ganglia.sourceforge.net/
[8] http://host-sflow.sourceforge.net/

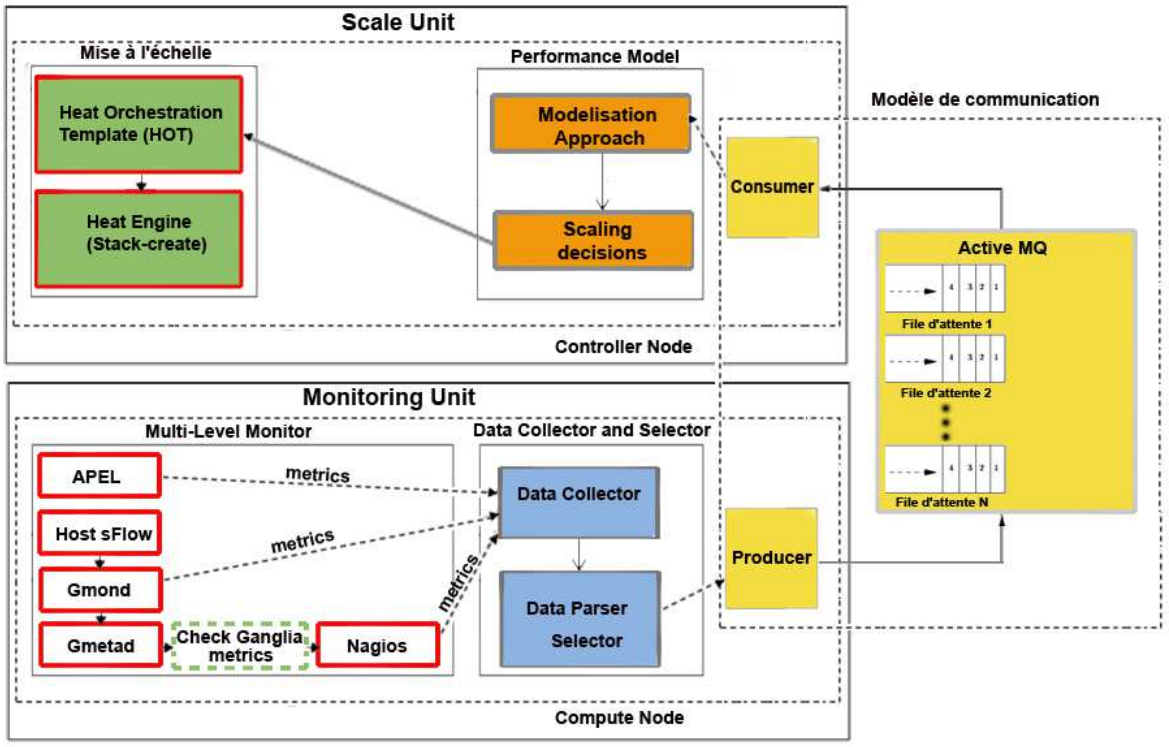

Figure 5.4: Schéma général de la mise en oeuvre

Ganglia est architecturalement composé de trois daemons: gmond, gmetad, et une interface web, généralement appelée ganglia-web "gweb" (voir figure 5.5). Pour cela, nous avons installé sur chaque noeud physique de l'infrastructure "Grid-Cloud" le daemon "Gmond" et sur un serveur centralisé le daemon "Gmetad" et le daemon "gweb":

1. L'agent "Gmond" de "Ganglia" est installé sur chaque noeud physique, et nous permet de collecter des métriques de performance physiques, telles que CPU, mémoire, disque, réseau, et des données sur les processus actifs.

2. Gmetad (Ganglia Meta Daemon) est le service qui collecte des métriques d'autres sources gmond et stocke leur états sur le disque au format RRD.

3. Après avoir collecter plusieurs métriques, nous avons certainement besoin d'une représentation visuelle, de préférence en utilisant des graphiques dans le Web. Gweb assure cette fonction.

5.3.1.2 L'agent de monitoring "Host sFlow"

L'agent "Host sFlow" [10], un outil Open-source permettant d'exporter d'une source, des métriques de performance physiques et virtuelles en utilisant le protocole sflow. L'agent permet un monitoring de performance évolutif, multi-vendeurs, multi-OS. L'agent "Host sflow" exporte les métriques de monitoring CPU, mémoire, disque et réseaux du noeud physique et

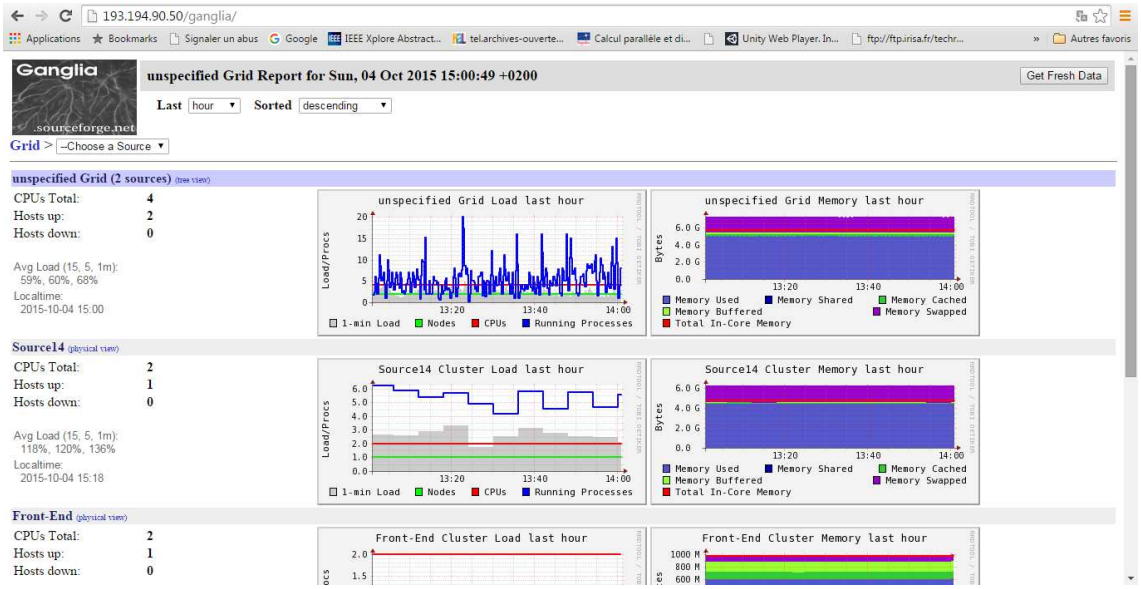

Figure 5.5: Outil de monitoring Ganglia

exporte également les mêmes métriques par machine virtuelle s'exécutant sur l'hyperviseur de l'hôte. L'agent gmond est configuré sur le noeud physique en mode collecteur seulement en écoutant les données sflow envoyées par l 'agent "Host sflow". Les métriques collectées par "gmond" peuvent être ensuite retirées sous format XML[9] (voir figure 5.6).

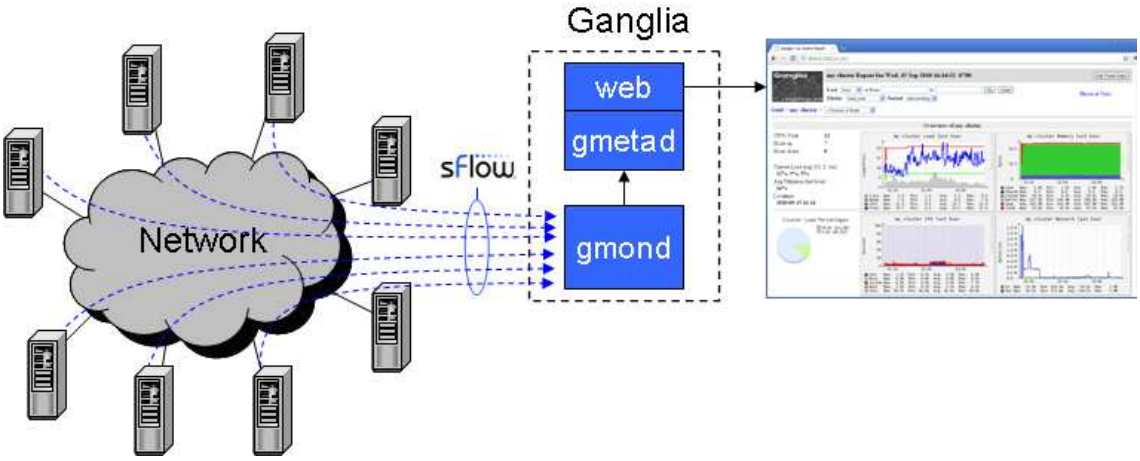

Figure 5.6: Architecture de Ganglia avec le protocol sflow

D'autre part des règles d'autorisation doivent être ajoutées sur le par feu Open vSwitch intégré dans la plate-forme OpenStack pour que l'agent "Gmond" de "Ganglia" se trouvant avec l'agent "Gmetad" sur le serveur centralisé collecte les métriques générées par l'agent "Host sFlow".

[9] eXtensible Markup language http://www.w3.org/XML/

5.3.1.3 Le Framework de monitoring NAGIOS

Nagios [59] est un framework open source ayant comme objectif la surveillance des hôtes du réseau et des services dans le but de détecter les pannes. Dans le contexte de Nagios, le service peut être un service réel du réseau, hôte ou un service métrique (par exemple la charge du serveur) ou un test fonctionnel (tel que le test d'exécution d'un job).

Nagios se compose de services de base et de capteurs. Le noyau de service Nagios effectue la planification et l'exécution des contrôles d'état de service et des opérations supplémentaires, tels que le lancement des notifications et des mécanismes de récupération automatique.

Nagios fournit une interface Web avancée qui permet aux utilisateurs d'afficher des informations de surveillance et modifier les paramètres de surveillance (par exemple désactiver les notifications, l'exécution des contrôles de services ou de programmer les temps d'arrêt d'une entité, etc.).

Dans la mise en oeuvre de notre système "multi-level basé modèle de performance" nous avons déployé le Framework Nagios intégré avec l'environnement Cloud "OpenStack" et installé un Framework Nagios spécifique à un environnement de grille. Les chercheurs et développeurs de projets Nagios pour les Grilles ont conçu et mis en œuvre plusieurs extensions et approches qui répondent à des problèmes spécifiques. Pour notre installation nous nous sommes basé sur la documentation de "SAMUpdate23 - EGIWiki[10]".

nagios-virt - Le Framework Nagios intégré avec l'environnement Cloud "OpenStack" considère toutes les machines de l'infrastructure comme étant des machines physiques, ce qui n'est pas pratique dans un environnement Cloud qui est basé sur la virtualisation, pour cela nous avons additionné l'outil de configuration "nagios-virt[11] " pour notre infrastructure virtuelle (voir Figure 5.7).

Ganglia avec Nagios -

L'installation de Ganglia comprend un certain nombre de métriques intégrées telles que la charge moyenne, l'utilisation de la CPU, disque, etc. Cependant, on peut également ajouter des métriques personnalisées. Evidemment, il serait très utile d'être en mesure d'utiliser ces métriques dans des alertes Nagios (voir figure 5.8). Il ya plusieurs façons d'intégrer le système de monitoring Ganglia avec les systèmes d'alerte tels que Nagios. Parmi les projet, nous avons utilisé "check-ganglia-metric[12]", qui est un plugin Nagios qui collecte les données de Ganglia de l'agent "gmetad" et les stocke de manière transparente dans un fichier de cache local. Quelques arguments de ligne de commande doivent être ajouté dans la configuration de nagios.

[10] NagiosSam23 https://wiki.egi.eu/wiki/SAMUpdate23
[11] https://people.redhat.com/~rjones/nagios-virt/
[12] http://vuksan.com/linux/nagios_scripts.html/

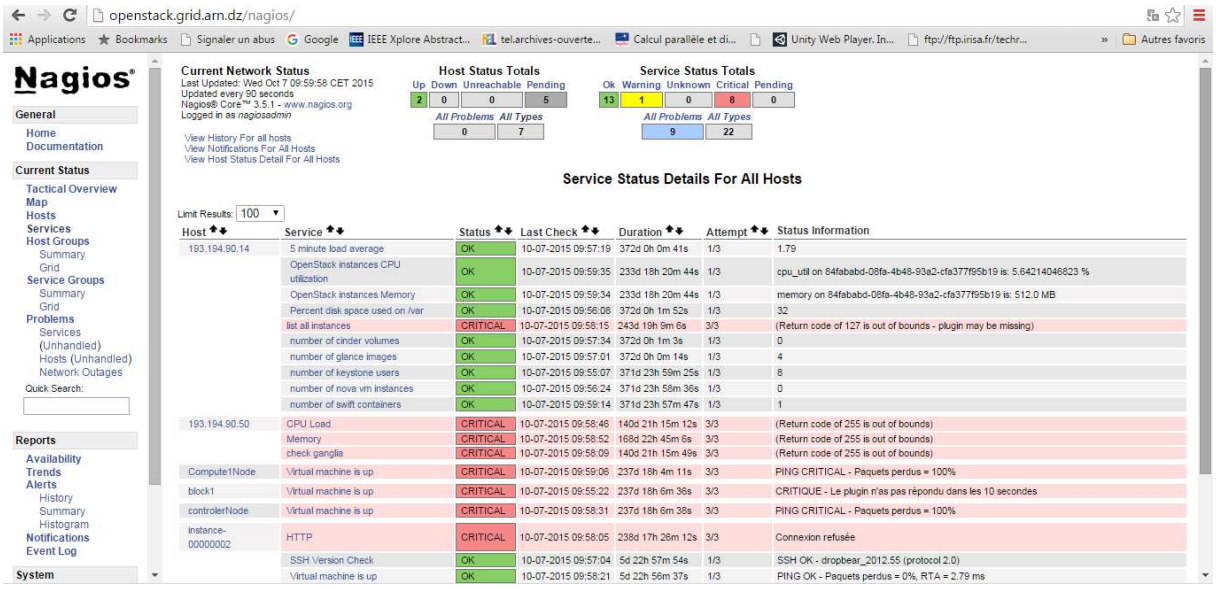

Figure 5.7: L'outil de monitoring Nagios intégré dans OpenStack

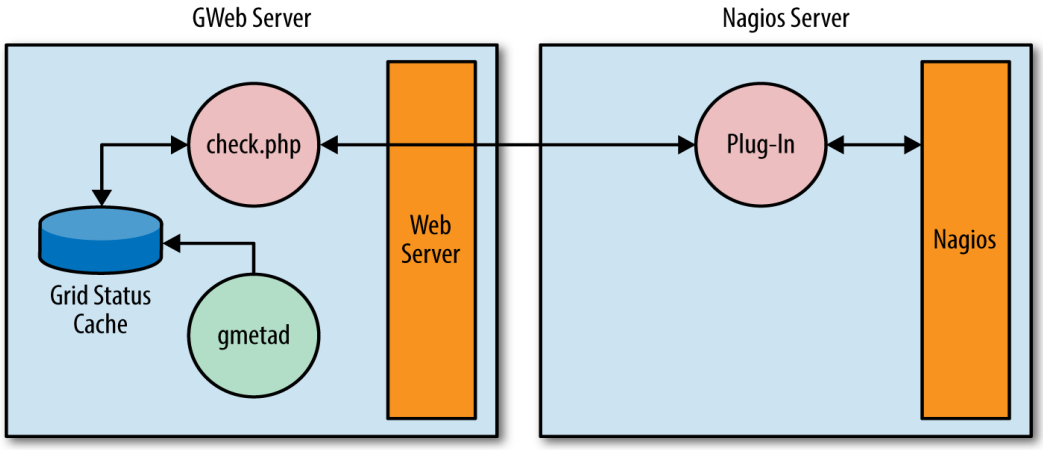

Figure 5.8: Principe de Plug-in de fonctionnement Ganglia-Nagios

5.3.1.4 L'outil de monitoring APEL

Pour le monitoring niveau application, nous avons intégré l'outil de monitoring "APEL[13]" (Accounting Processor for Event Logs). Apel est un outil qui rassemble des informations sur l'utilisation CPU en analysant les fichiers logs des systèmes batch et blah accounting et insére les données dans une base de données MySQL locale. APEL publie ensuite les données dans un dépôt centralisé.

Dans notre implémentation du système "Multi-Level" nous avons installé le "client APEL[14]" sur le service grille "CE-Stack" (Computing Element) déjà déployé sur l'environnement "Grid-Cloud". Le client APEL se compose de deux éléments principaux, "APEL parser" et "APEL

[13] https://twiki.cern.ch/twiki/bin/view/EMI/EMI3APELClient
[14] https://wiki.egi.eu/wiki/APEL

publisher" (voir figure 5.9).

Les Parsers APEL interprètent les fichiers logs pour extraire des informations sur le job. Plus précisément, le traitement des fichiers logs produites par un système batch et les fichiers logs accounting blahd/grid-jobmap, qui sont produites par un CE. Tandis que APEL Publisher est utilisé pour décrire le client Apel.

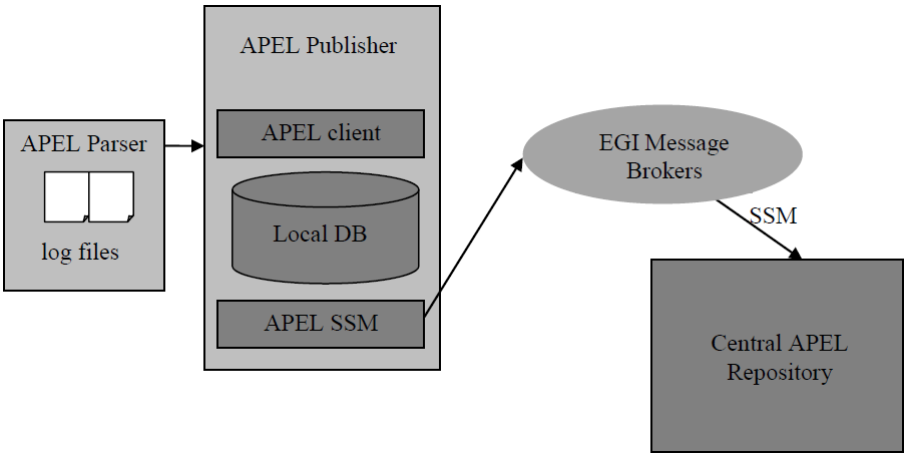

Figure 5.9: Architecture Apel Client

5.3.1.5 L'API JMS et le courtier "Apache ActiveMQ"

Les composants de notre système échangent un grand nombre de métriques collectées à partir de plusieurs niveaux de monitoring. Cependant, il ya un besoin de moyens fiables et évolutives de communication. Afin de répondre à ce besoin, nous avons conçu et mis en place un modèle de communication basé sur l'API Java Messaging Service "JMS[15]", qui est une API Middleware orientée message java (MOM), pour envoyer des messages entre deux ou plusieurs clients. Afin d'utiliser JMS, nous avons besoin d'un fournisseur JMS qui gère les sessions et les files d'attente. A cet effet, nous avons utilisé l'open source bien établie "Apache ActiveMQ[16]".

Notre modèle de communication mis en œuvre est basé sur un mécanisme de file d'attente. Nous l'utilisons pour réaliser une communication inter-processus pour faire passer nos métriques collectées à partir de différents oultis de monitorirng multi-level entre deux composants du système, en raison du fait que les composants peuvent fonctionner sur des machines différentes et à différents endroits. Pour cela, nous avons installé "Apache ActiveMQ 5.11.1" avec l'API "JMS 1.1"

1. L'interface de programmation Java Message Service (JMS) permet d'envoyer et de recevoir des messages de manière asynchrone entre applications (producteurs et consommateurs de messages) ou composants Java. JMS permet l'échange de messages entre deux systèmes ou plus. Ce service supporte le modèle publication/abonnement

[15] http://java.sun.com/products/jms/
[16] http://activemq.apache.org/

(publish/subscribe) et le modèle point à point. Dans notre implémentation, nous avons utilisé le "modèle point à point", où le producteur publie les messages dans une file (queue) et le consommateur lit les messages de la file. Dans ce cas le producteur connaît la destination des messages et poste les messages directement dans la file du consommateur. Pour utiliser ce modèle, le consommateur doit invoquer la méthode receive() qui est bloquante.

2. "Apache ActiveMQ" est un courtier de message open source écrit en Java avec un client Java Message Service (JMS) complet. C'est un fournisseur de service compatible avec l'API JMS, qui gère les connexions, les sessions, les destinations et les messages. Il fournit des "fonctionnalités d'entreprise" qui signifie dans ce cas favoriser la communication de plus d'un client ou un serveur.

5.3.2 Environnement de développement

Vu nos besoins en développement qui sont variés (modélisation, prédiction, communication, développement d'applications), nous avons eu recours à différents outils de développement.

Eclipse JDT - Concernant les besoins en développement d'application, nous les avons réalisé avec "Eclipse[17]JDT". C'est un environnement de développement intégré principalement écrit en JAVA; il est libre et extensible grâce à son architecture totalement axée sur l'utilisation des plugins.

Le langage XML - Utilisé pour la représentation des métriques collectées à partir des outils de monitoring niveau infrastructure et virtuelle (gmond, gmetad et hsflowd)

Le langage SQL - Utilisé pour intérroger la base de données Apel et récupérer les métriques niveau applicatif comme le temps de réponse d'une exécution d'un job.

5.3.3 Mise en oeuvre du système "Multi-Level basé modèle de performance"

5.3.3.1 Implémentation du module "multi-level Monitor"

Il se compose de l'ensemble d'outils de monitoring ("Gmond", "Gmetad","Host Sflow", "Nagios" et "Apel") (voir la figure 5.4) installé sur chaque niveau de notre environnement Cloud OpenStack:

1. "Gmond", "Gmetad", et "Host Sflow" responsables de la mesure et la capture continuelle des métriques de monitoring de bas-niveau (CPU, Mémoire, Disque, Réseaux), situés sur chaque nœud physique de l'environnement Cloud (nœud compute).

2. "Nagios" responsable du contrôle des images des machines virtuelles et les niveaux de service disponibles sur l'environnement Cloud.

[17] http://www.eclipse.org/

3. “Apel” responsable de la mesure et la capture continuelle des métriques de monitoring de haut-niveau (temps de réponse, nombre de requêtes, charge de travail) situé sur la machine virtuelle du service de calcul “CE-Stack” de l’environnement Cloud.

Les métriques capturées par les différents agents sont envoyées aux deuxième module “Data Colector and Selector” responsable de la collecte et la sélection

5.3.3.2 Implémentation du module “Data Colector and Selector”

L’implémentation du module “Data Colector and Selector” est basée sur des programmes JAVA que nous avons implémenté pour collecter et filtrer les métriques issues de plusieurs sources de monitoring. Nous présentons dans les sections suivantes la mise en oeuvre des principales fonctions du module :

collecte des métriques Pour recevoir les métriques de monitoring collectées par “Gmond de ganglia” et générées par l’agent “Host sflow” sur le noeud physique, nous avons implémenté un programme en Java “DataCollector” permettant d’interroger périodiquement le daemon “Gmond” sur le socket “TCP”, de saisir l’état du noeud surveillé en format XML. Le fichier XML contient des informations d’état du noeud physique (OS, Date démarrage, architecture, ...etc), mais aussi toutes les métriques de monitoring par défaut que “Gmond” collecte sur les ressources physiques du noeud (disk_total, disk_free, mem_total, mem_free, cpu_num, cpu_speed, ...etc), ainsi que les métriques par machine virtuelle s’exécutant sur le noeud, comme par exemple (<VM name>.vcpu_util, <VM name>.vmem_total, <VM name>.vdisk_total, ...etc).

parser et filtrer Nous avons implémenté un programme en Java “DataParserSelector”, qui nous permet d’analyser le fichier “Ganglia_XML” collecté par “Gmond” sous format XML. Le programme nous permet de parcourir le fichier XML par le parser “SAXParser”, d’en extraire les éléments représentants les métriques de monitoring et leur valeur en se basant sur leur “XPath[18]”.

5.3.3.3 Implémentation du module de communication

Ce module est doté d’un modèle de communication, qui suit une approche producteur-consommateur. Le modèle de communication est basé sur un bus fournissant la communication entre les différents modules. Le bus de communication fait passer les métriques entre les différents modules en utilisant un processus de file d’attente “Queuing”, qui permet un mécanisme d’envoi fiable et efficace. Nous présentons dans cette section, la mise en oeuvre des principales fonctions du module :

[18] http://xmlfr.org/w3c/TR/xpath/

Envoi des métriques et mécanisme de communication - Sur chaque noeud physique où les agents de monitoring sont installés ("Gmond", "Host Sflow", Nagios et APEL sur le CE), nous avons implémenté un programme JAVA qui agit comme un producteur de messages JMS (figure dessin), en se connectant à distance aux queues JMS que nous avons défini sur le composant "Apache Active MQ" sur la machine centrale et envoi les métriques collectées, parsées et filtrées, chacune sur une file d'attente spécifique. Le bus de communication fait passer les métriques au module "Modèle de performance" en utilisant un processus de file d'attente "Queuing".

Réception des métriques et mécanisme de communication - Pour recevoir les métriques de monitoring envoyées par le module "Data collector and selector", nous avons implémenté un programme JAVA qui agit comme un consommateur de messages JMS, en écoutant les queues des métriques prédéfinies sur "Apache ActiveMQ" et en consommant les messages dans l'ordre de leur réception.

5.3.3.4 Implémentation du module "Modèle de performance"

Notre modèle de performance se base sur les approches ayant employé la modélisation de la performance de l'application tels que les modèles de prèdictions quantitatives. Nous reposons sur les méthodes des séries chronologiques pour fournir un moyen de prédiction en se basant sur l'historique.

Dans ce qui suit, nous introduisons, d'abord, la modélisation des séries chronologiques. Ensuite, la mise en oeuvre des techniques de prédiction et la prise de décision (voir figure 5.10).

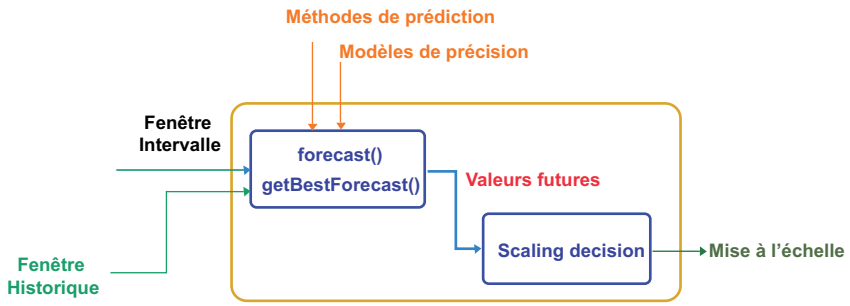

Figure 5.10: Architecture du modèle de performance

Modélisation d'une série chronologique: L'étude d'une série chronologique permet d'analyser, de décrire et d'exploiter un phénomène au cours du temps. Nous utilisons les séries chronologiques pour la prédiction des valeurs futures (intervalle de prédiction) de la demande à partir des valeurs observées (fenêtre de prédiction).

Dans notre implémentation les valeurs observées (fenêtre de prédiction) sont les métriques colléctées issues de plusieurs sources et échantillonnées périodiquement à des intervalles fixes (chaque minute). Le résultat pour chaque métrique sera une time-series X contenant une séquence des dernières observations w:

$$X = x_t + x_{t-1} + x_{t-2} + ... + x_{t-w+1}$$

Le problème de modélisation de la perfomance d'une application est divisé en deux parties:

1. La prédiction des valeurs futures (en utilisant l'analyse de séries chronologiques), qui consiste à faire une estimation de la future charge de travail ou une utilisation de ressource;

2. La prise de décision: L'approche utilisée est une combinaison de règles proactives et réactives.

En premier lieu, nous avons développé un programme JAVA "TimeSeries", qui consiste à modéliser les valeurs de chaque métrique échantillonnées périodiquement à des intervalles fixes dans une TimeSeries.

Techniques de prédiction: En deuxième lieu, nous avons utilisé des méthodes de prédiction (ForecastingMethod) et modèles de précision (ForecastingAccuracy). Les méthodes de prédiction utilisées sont les ***méthodes de la moyenne*** à savoir: Simple Moving Average *SMA* et Exponentiel Moving Average *EMA*. La valeur future (de prévision) y_{t+r} est calculée comme la moyenne pondérée des dernières valeurs w consécutives. La formule générale est la suivante: $y_{t+r} = a_1 x_t + a_2 x_{t-1}, ...$, où $a_1, a_2, ..., a_w$ sont un ensemble de facteurs de pondération positifs qui doivent sommer 1.

Listing 5.1: Partie de l'algorithme Moving Average

```
public static TimeSeries createMovingAverage(TimeSeries source,
                                              String name,
                                              int periodCount,
                                              int skip) {

        // check arguments
        if (source == null) {
            throw new IllegalArgumentException(``Null source.");
        }

        if (periodCount < 1) {
            throw new IllegalArgumentException(
                ``periodCount must be greater than or equal to 1."
            );

        }

        TimeSeries result = new TimeSeries(name, source.getTimePeriodClass());

        if (source.getItemCount() > 0) {

            // if the initial averaging period is to be excluded, then
            // calculate the index of the
            // first data item to have an average calculated...
```

```
long firstSerial
    = source.getDataItem(0).getPeriod().getSerialIndex() + skip;
```

Pour l'implémentation et l'évaluation des algorithmes de prédiction nous avons suivi les étapes suivantes:

- **Les librairies JfreeChart et OpenForecast**: Nous utilisons les librairies "JfreeChart[19]" et "OpenForecast[20]" pour l'implémentation des méthodes de prédiction. Nous nous intéressons à ces librairies qui sont utilisées pour des méthodes de prévision à court terme. Parmi ces méthodes : La simple Moving Average (SMA) et Exponentiel Moving Average (EMA).

- **Choix des paramètres des méthodes de prédictions:** La précision des algorithmes dépendent de différents paramètres tels que la longueur de l'intervalle de monitoring, la taille de la fenêtre d'historique qui détermine la sensibilité de l'algorithme pour les tendances locales versus les tendances globales et le pas. Notons que pour ces choix de paramètres, on s'est basé sur les travaux de recherche effectués [37]:

  1. **Monitoring-interval length**: La longueur d'intervalle du monitoring dans chaque time series est d'une durée de 1 minute. Elle représente l'intervalle entre les valeurs de métriques collectées.
  2. **Size of the history window**: La longueur "history window" est d'une durée de 1 heure et représente la longueur de la time series.
  3. **Sliding Windows**: Elle représente un bloc de valeurs de la time series et qui avance avec un pas (skip) prédéfini. Dans notre cas l'intervalle est d'une durée de 14 minutes.
  4. **skip** : Représente la longueur de la période de saut. Dans notre cas nous l'avons fixé à 1.

- **Prédire les valeurs futures (forecast)**: Nous avons appliqué la méthode forecast() de la librairie OpenForecast sur les deux méthodes de prédiction (SMA et EMA) pour prédire les valeurs futures.

- **Sélection de méthode de prédiction (getBestForecaster):** Dans cette partie nous avons utilisé un modèle de précision qui permet d'évaluer les deux méthodes de prédiction afin de sélectionner la meilleure méthode qui a la plus petite valeur d'erreur de prédiction. Parmi les modèles de précision, nous avons utilisé l'*erreur absolue moyenne (Mean absolute error - MAE)* de la librairie OpenForecast. Le modèle de précision "MAE" est utilisé lorsque on compare les méthodes de prévision sur un ensemble de données unique, ce qui est notre cas.

[19] http://www.ktipsntricks.com/data/ebooks/java/jfreechart-1.0.13.pdf
[20] http://www.stevengould.org/software/openforecast/index.shtml

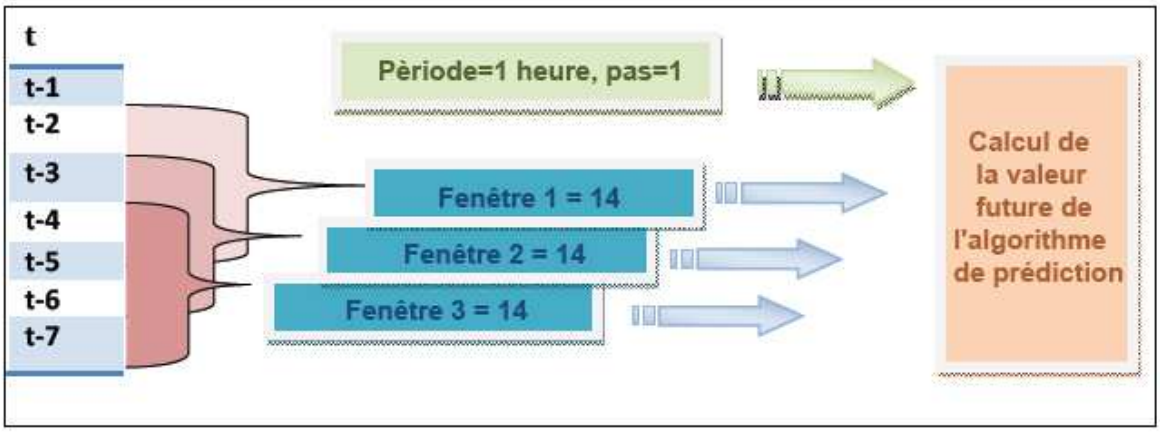

Figure 5.11: Paramètres du modèle de prédiction

La prise de décision: Dans l'approche à base de règles que nous avons adopté, nous avons défini des régles de type ***"Event-Condition-Action"***. Pour cela, nous avons défini un ensemble de règles, à savoir des règles proactives pour la mise à l'échelle et des règles réactives pour les notifications, afin d'améliorer la performance de notre approche.

5.3.3.5 Module mise à l'échelle

Nous avons définis des règles proactives et réactives qui nous permettent de déclencher le re-dimensionnement automatique et des notifications aux administrateurs. Notre solution doit permettre de réaliser un autoscaling sur la plate-forme Cloud à chaque fois que cela est nécessaire, c'est-à-dire qu'elle ajuste le nombre d'instances des worker nodes à la demande.

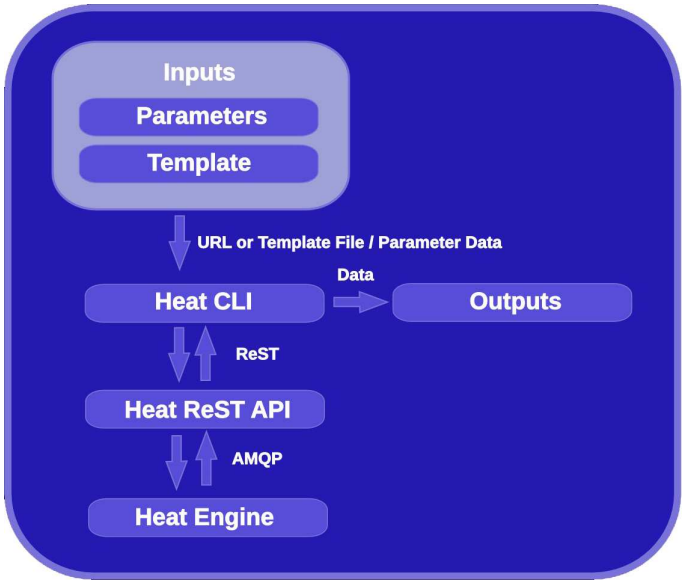

Figure 5.12: Architecture Heat OpenStack

Pour cela, nous avons utilisé le moteur d'autoscaling que nous propose OpenStack du

service ***Heat*** voir figure 5.12.

Heat est le moteur d'orchestration d'OpenStack, qui nous permet de lancer des applications cloud basées sur des templates de type HOT (Heat Orchestration Templates). Les templates sont écrites en YAML, ce qui nécessite une étude et une programmation de ces templates. La template décrit le processus ou la logique avec laquelle l'instanciation est décrite et gérée.

En premier lieu, nous avons développé des templates d'environnement pour une image WorkerNode. La template instance va créer une instance basée sur l'image WorkerNode, configurer un volume, ajouter une adresse IP du réseau privé, ajouter IP flottante du réseau public, ajouter un groupe de sécurité, et une clé ssh privée, le listing 5.2 présente la template.

Listing 5.2: Programme de template WorkerNode.yaml

```
1
2  heat_template_version: 2014-10-16
3  description: A base WorkerNode server
4
5  resources:
6    server:
7      type: OS::Nova::Server
8      properties:
9        block_device_mapping:
10         - device_name: vda
11           delete_on_termination: true
12           volume_id: { get_resource: volume }
13       flavor: m1.nano
14       key_name: admin
15       networks:
16         - port: { get_resource: port }
17
18   port:
19     type: OS::Neutron::Port
20     properties:
21       network: private
22       security_groups:
23         - all
24
25   floating_ip:
26     type: OS::Neutron::FloatingIP
27     properties:
28       floating_network: public
29
30   floating_ip_assoc:
31     type: OS::Neutron::FloatingIPAssociation
32     properties:
33       floatingip_id: { get_resource: floating_ip }
34       port_id: { get_resource: port }
35
36   volume:
37     type: OS::Cinder::Volume
38     properties:
39       image: 'WorkerNode'
40       size: 1
```

Maintenant que nous avons un modèle de l'environnement, nous avons créé un type de ressource Heat et l'associé au fichier WorkerNode.yaml. A ce stade, nous sommes prêts à exécuter notre auto scaling Heat stack. Le résultat est une instance WorkerNode1 lancée.

5.4 Expérimentation et évaluation

Dans cette section, nous présentons l'expérimentation et l'évaluation des résultats de notre approche de monitoring "multi-level basé modèle performance", sur un environnement réel d'intégration "Grid-Cloud" que nous avons mis en œuvre.

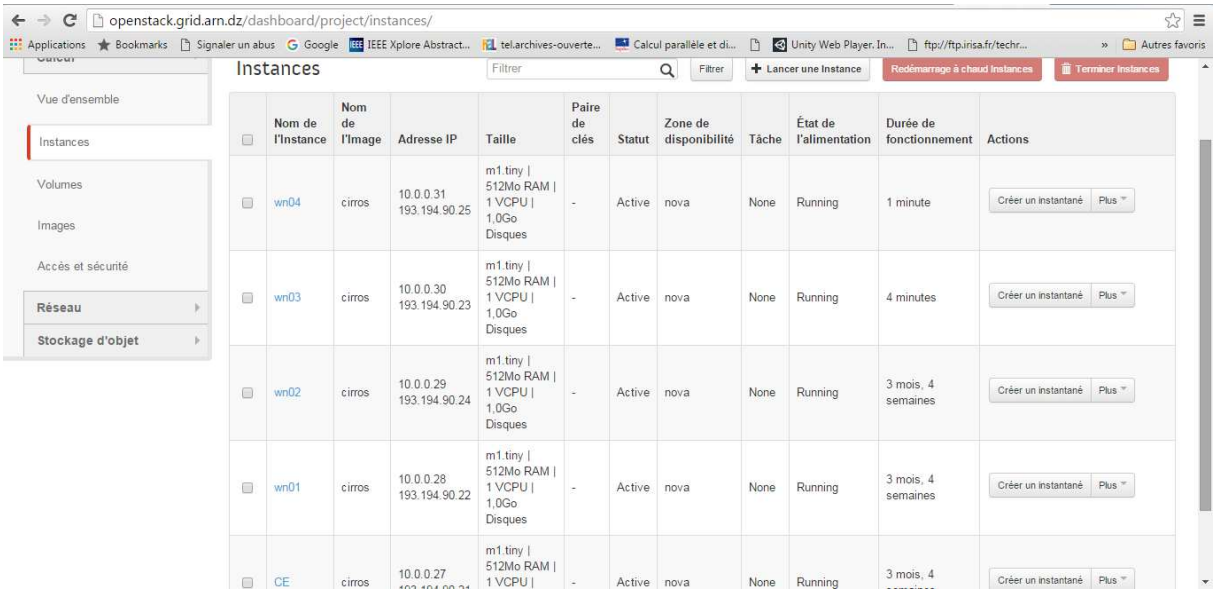

Figure 5.13: OpenStack Dashboard

L'environnement est composé d'une machine (openstack.grid.arn.dz) d'administration Cloud "OpenStack", et de deux autres machines physiques constituant les nœuds compute de l'infrastructure du Cloud. Le premier nœud compute (compute1.grid.arn.dz), comporte trois machines virtuelles, une machine virtuelle représentant le service grille élément de calcul "CE-Stack.grid.arn.dz" qui est responsable de soumettre les jobs aux worker nodes via le système de batch local, une machine virtuelle qui représente le service nagios "nagios.grid.arn.dz" et une autre machine virtuelle qui représente le worker node "wn01-stack.grid.arn.dz". Tandis que le deuxième nœud compute (compute2.grid.arn.dz), comporte une machine worker node "wn02-stack.grid.arn.dz" (voir figure 5.13). Les caractéristiques de chaque machine sont résumées sur le tableau 5.3. Notons que cet environement de test est lié à la grille "DZ e-Science GRID" et la soumission des jobs va se faire à partir du service "ui.grid.arn.dz" de la grille.

Table 5.3: Caractéristiques des machines

Machine	Type	CPU	Cores	Memory	Storage
openstack.grid.arn.dz	physique	Intel(R) Core(TM)2 Duo CPU	4	4GB	300GB
compute1.grid.arn.dz	physique	Intel(R) Xeon(R) 2 CPU	4	2GB	300GB

Table 5.3: Caractéristiques des machines

Machine	**Type**	**CPU**	**Cores**	**Memory**	**Storage**
compute2.grid.arn.dz	physique	Intel(R) Xeon(R) 4 CPU	8	10GB	2TB
ce-stack.grid.arn.dz	VM	Intel(R) Xeon(R) 2 CPU	2	1GB	20GB
wn01-stack.grid.arn.dz	VM	Intel(R) Xeon(R) 2 CPU	1	1GB	20GB
wn02-stack.grid.arn.dz	VM	Intel(R) Xeon(R) 2 CPU	1	1GB	20GB
nagios.grid.arn.dz	VM	Intel(R) Xeon(R) 2 CPU	2	1GB	50GB

La machine virtuelle "ce-stack.grid.arn.dz" a été instanciée à partir de l'appliance virtuelle définie pour le service de calcul grille "CE" et du modèle de VM commun (voir la section 5.2.3.2). Les machines virtuelles Worker nodes ont été instanciées à partir de l'appliance virtuelle définie pour le service grille worker node et le modèle du VM commun, et la machine virtuelle "nagios.grid.arn.dz" est une machine qui contient l'outil nagios installé et configuré pour la grille "DZ e-Science GRID".

La machine physique "openstack.grid.arn.dz" dispose du daemon "Gmetad" responsable de la collecte des métriques à partir des agents "Gmond" et le daemon "gweb", pour la représentation visuelle. Les noeuds (compute1.grid.arn.dz, compute2.grid.arn.dz) disposent de l'hyperviseur de virtualisation "KVM" et des agents de monitoring "Host Sflowd" et "Gmond" de "Ganglia", ainsi que le client "nagios" pour les alertes, et la machine virtuelle "ce-stack.grid.arn.dz", dispose de l'outil "Apel" pour les métriques niveau application (job).

Le module "Data Collector and selector" est installé sur chaque noeud compute. Tandis que le module "modèle de performance" est installé sur la machine d'administration Cloud "openstack.grid.arn.dz" pour l'analyse et le traitement des données et le module de communication qui relit les deux autres modules.

5.4.1 Scénario "Case Study"

Pour valider notre approche de monitoring multi-level basé modèle de performance, nous avons déroulé deux scénarios : Un scénario pour l'exécution de jobs séquentiels et un scénario pour l'exécution de jobs parallèles sous une variation de charge de travail, afin de tester la faisabilité et la performance de notre approche. Nous avons aussi évaluer les deux techniques de prédiction "SMA" et "EMA". Les scénarios consistent à suivre les étapes suivantes :

1. Collecter périodiquement (la période est d'une minute pour chaque métrique) les différentes métriques issues de notre système de monitoring multi-niveaux (Host Sflowd, Gmond, Nagios et Apel).

2. Modéliser l'ensemble des valeurs échantillonnées de chaque métrique sous une time de series avec une fenêtre de prédiction de 60 observations (1 heure de temps).

3. Prédire périodiquement les valeurs futures en utilisant les deux méthode de prédiction “SMA” et “EMA”.

4. Evaluer les deux méthodes de prédiction en utilisant le modèle de précision “MAE” puis sélectionner la meilleure méthode qui a la plus petite valeur d’erreur de prédiction.

5. Décider de l’action de mise à l’échelle à réaliser (mise à l’échelle horizontale) à partir des règles à base de seuil proactives issues des algorithmes et réactives issues du monitoring en temps réel.

Nous avons combiné entre deux types de règles à base de seuil afin d’améliorer la performance de notre approche: des règles proactives et des règles réactives. Les règles proactives se basent sur le résultat des approches prédictives et les valeurs futures des métriques de ressources. Pour cela, nous avons utilisé deux méthodes de prédiction : Simple Moving Average (SMA) et Exponential Smoothing (EMA) (les techniques de prédiction sont détaillées dans la section 4.5.4.5), et les deux s’exécutent sur les mêmes time seris. Tandis que, les techniques réactives, comme son nom l’indique, se réfèrent à l’ensemble des méthodes qui réagissent au système actuel et / ou l’état de l’application.

5.4.1.1 Choix des métriques de performance

Pour évaluer la performance des applications scientifiques (jobs), nous avons sélectionné des métriques quantitatives, à travers le module “Data Collector and Selector”, et qui sont présentées dans le tableau suivant 5.4:

Table 5.4: Application-Level Metrics for job batch

Metric Name	**Description**	**Monitoring level**
VMx.available	Disponibilité d’une VM	VM
Workload	Nombre de jobs par minute	Application
Response time	Temps d’exécution d’un job	Application
VMx.vcpu_util	Valeur de l’utisation cpu de la VM	VM
Computex.cpu_idle	Inactivité CPU du noeud compute	Machine physique

5.4.1.2 Définition des règles pour le module “Mise à l’échelle”

Définition des règles proactives En se basant sur les valeurs futures prédites à partir de l’analyse des times series, en utilisant les algorithmes cités ci-dessus, nous allons définir des règles proactives pour gérer la quantité de ressources affectées à une application s’exécutant sur notre environnement “Grid-Cloud”, et pour la prise de décision d’un redimensionnement automatique et une notification pour l’administrateur. Supposons qu’on a les objectifs suivants :

- **Règle proactive 1**: Ajouter une instance lorsque la future valeur moyenne de la

métrique "CPU utilisation" sur une machine virtuelle VMx (qui exécute une application grille y), et qui est calculée à partir des algorithmes de prédiction, est supérieure à 18, et libérer une instance si la valeur est Inférieur à 2.

$$\begin{cases} if \quad Future\_VMx.vcpu\_util > 18 \quad alors \quad faire \quad n = \quad n+1 \\ if \quad Future\_VMx.vcpu\_util < 2 \quad alors \quad faire \quad n = \quad n-1 \end{cases}$$

Où : Future_VMx.vcpu_util est la valeur moyenne prédite de métrique sur l'utisation cpu de la VM

- **Règle proactive 2**: Ajouter une instance si la future valeur moyenne du temps de réponse d'un job est supérieur à 2 minutes.

$$if \quad FutureJob\_Response\_Time \geq 120 \quad alors \quad faire \quad n = \quad n+1$$

Où : FutureJob_Response_Time est la valeur moyenne prédite du temps de réponse qu'un job peut prendre dans un temps futur $(t+r)$.

- **Règle proactive 3**: Ajouter une instance si la future valeur moyenne de "CPU utilisation" est supérieur à 18, et la future valeur moyenne du temps de réponse d'un job est supérieur à 2 minutes.

$$if \quad Future\_VMx.cpu\_util > 18 \quad et \quad FutureJob\_Response\_Time \geq 120 \quad alors \quad faire \quad n = \quad n+1$$

Où : Future_VMx.cpu_util est la valeur moyenne prédite de métrique sur l'utisation cpu de la VM.
FutureJob_Response_Time est la valeur moyenne prédite du temps de réponse qu'un job peut prendre dans un temps futur $(t+r)$.

Définition des règles réactives En se basant sur les valeurs des métriques collectées, nous allons définir des règles réactives qui servent à notifier l'adminstrateur sur l'utilisation des ressources.

1. **Règle réactive 1**: Notifier l'administrateur pour les compute nodes inactifs, dans ce cas la deux cas de figures se présentent: une alerte de notification pour un ajout d'instance worker node sur un noeud compute en cas de sur-utilisation, et une alerte de notification pour une suppression d'instance worker node en cas de sous-utilisation.

$$\begin{array}{l} if \quad Computex.cpu\_idle < 50 \quad pour \quad 15 \quad minutes \quad alors \quad faire \quad notification \\ if \quad Computex.cpu\_idle > 90 \quad pour \quad 15 \quad minutes \quad alors \quad faire \quad notification \end{array}$$

Où : Computex.cpu_idle est la valeur de métrique de l'inactivité CPU du noeud compute.

2. **Règle réactive 2**: Signaler si un "Worker Node (WN)" n'est pas disponible pendant 10 minutes.

if VMx.availble = 0" pour 10 minutes alors faire notification

Où : VMx.availble est la valeur de métrique de disponibilité d'une VM.

5.4.1.3 Scénario de l'exécution des jobs séquentiels

Nous avons sélectionné un ensemble de jobs de type séquentiel pour ce scénario. Les jobs suivent une charge de travail (workload) qui va varier dans des périodes de temps durant l'exécution.

Pour ce faire, nous avons lancé le script (workload-seq.sh) à partir du service "ui.grid.arn.dz". Le script travail comme un générateur de charge qui éxécute des jobs séquentiels pendant des périodes prédéfinies.

Le module "Data Collector", va collecter les métriques des ressources Cloud sur lesquelles les jobs s'exécutent. Les métriques sont transmises par la suite en format XML via le module de communication qui suit une approche producteur-consommateur pour les mettre dans des files d'attente préalablement définies dans le ActiveMQ (figure 5.14).

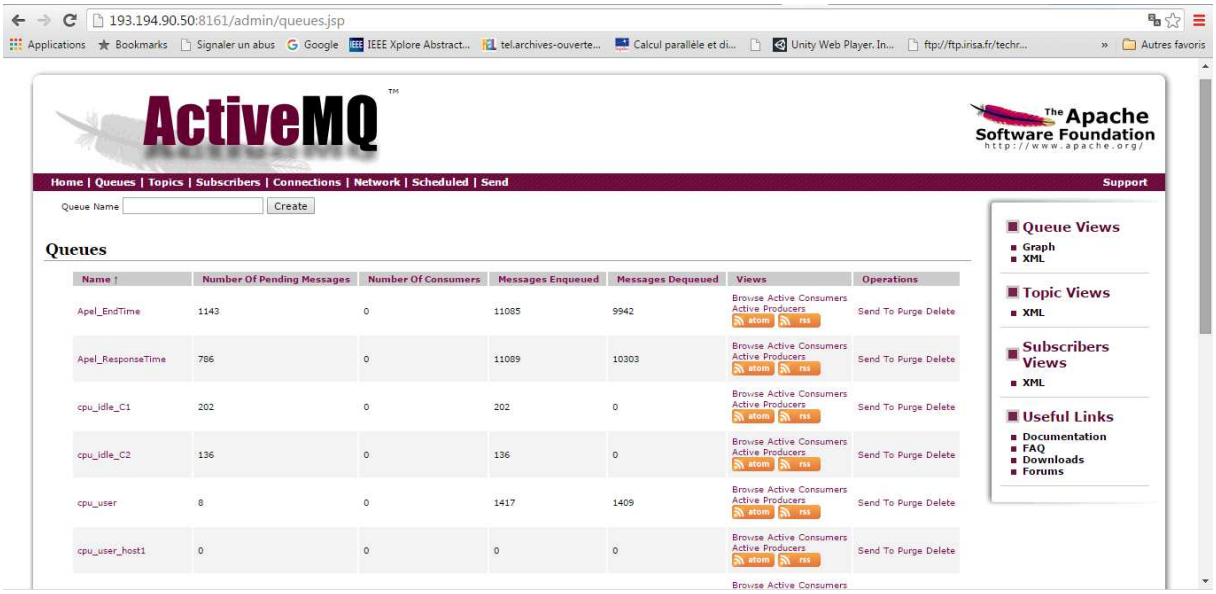

Figure 5.14: Les queues "Active MQ" des métriques de performance.

L'éxécution du scénario sur le module "modèle de performance" est montré sur le tableau 5.5.

Table 5.5: Résultat du scénario des jobs séquentiels sur 5 heures d'exécution

Workload jobs/minute	**méthode de prédiction**	**Future Value Response Time**	**MAE**	**Future Value Cpu Util WN01**	**MAE**	**Future Value Cpu Util WN02**	**MAE**	**Avg C1.cpu_idle /15 minutes**	**Avg C2.cpu_idle /15 minutes**
2	SMA	7	45.78	9.45	42.56	11.93	30.14	74.32 % 78.05 % 76.51 % 75.98 %	88.77 % 89.55 % 87.86 % 89.25 %
	EMA	1	21.41	2.45	20.19	11.79	25.36	79.80 % 80.10 % 83.51 % 82.88 %	90.47 % 90.06 % 89.46 % 89.59 %
3	SMA	6	33.78	2.45	32.11	14.26	37.25	82.73 % 61.46 % 68.22 % 79.97 %	89.57 % 86.92 % 87.92 % 87.89 %
	EMA	1.5	19.12	3.75	26.15	11.95	18.13	80.15% 81.23% 80.54 % 79.16 %	90.36 % 88.73 % 87.51 % 88.92 %
4	SMA	1.5	51.02	2.4	53.12	14.66	30.54	67.58 % 72.33 % 80.54 % 80.15 %	90.54 % 88.20 % 87.88 % 87.52 %
	EMA	6	11.24	9.81	12.16	13.49	22.18	81.47 % 78.22 % 79.13 % 77.12 %	89.41 % 88.83 % 86.98 % 86.40 %
5	SMA	1	63.91	16.49	23.14	13.86	22.15	81.00 % 80.25 % 79.16 % 78.12 %	89.05 % 85.18 % 84.90 % 84.87 %
	EMA	9	22.27	17.04	15.88	12.59	17.21	65.24 % 63.90 % 64.57 % 66.78 %	89.01 % 87.25 % 88.63 % 86.69 %
6	SMA	1	60.71	16.59	16.12	10.82	31.05	78.44 % 78.55 % 78.74 % 78.90 %	89.67 % 86.92 % 86.88 % 84.34 %
	EMA	10	12.97	17.5	13.07	13.94	17.26	69.01 % 47.42 % 65.75 %	87.31 % 84.66 % 84.90 %

Table 5.5: Résultat du scénario des jobs séquentiels sur 5 heures d'exécution

Workload jobs/minute	méthode de prédiction	Future Value Response Time	MAE	Future Value Cpu Util WN01	MAE	Future Value Cpu Util WN02	MAE	Avg C1.cpu_idle /15 minutes	Avg C2.cpu_idle /15 minutes
								64.26 %	84.67 %

Discussion des résultats L'exécution du premier scénario sur les différents modules, nous a permis d'évaluer notre approche. Le tableau 5.5 montre les résultats du scénario 1. Le tableau présente chaque méthode de prédiction avec la prédiction des valeurs futures des métriques "temps de réponse" et "cpu utilisation" pour chaque worker node, ainsi que la moyenne de la métrique "cpu idle" issue du monitoring en temps réel et cela sous une variation de workload. Les valeurs prédites des deux méthodes de prédiction utilisent le même ensemble de données (time series).

Nous avons aussi calculé la mesure de précision des prévisions pour les valeurs futures afin de sélectionner la meilleur méthode de prédiction qui va être utilisée dans la prise de décision pour une mise à l'échelle. Pour le calcul de la mesure de précision des prévisions, nous avons utilisé le modèle de précision "MAE" (Mean Absolute error). Les modèles de précision permettent d'évaluer les méthodes de prédiction. Intuitivement, plus l'erreur de prévision est petite et meilleure est la méthode. Notons que pour le calcul de la précision pour chaque méthode de prédiction (SMA et EMA), nous avons utilisé la même time series.

D'aprés le calcul du modèle de précision "MAE" appliqué sur les deux méthodes de prédiction (SMA et EMA), nous remarquons que la méthode de prédiction "EMA" donne de meilleurs résultats que la méthode de prédiction "SMA". Cela permet de sélectionner la méthode de prédiction "EMA" et utiliser ses valeurs prédites dans les règles de décision pour une mise à l'échelle. Notons que dans ce scénario, aucune valeur prédite n'est arrivée au seuil pour déclencher une mise à l'échelle automatique.

On remarque aussi que le CPU utilisation du "wn02-stack.grid.arn.dz" parait plus élevée que celui du "wn01-stack.grid.arn.dz". Ceci est dû au fait que les caractéristiques "Hardware" des machines computes qui hébergent les deux worker nodes diffèrent largement.

5.4.1.4 Scénario des jobs parallèles

Dans le scénario 2, nous avons sélectionné un ensemble de jobs de type parallèle (voir l'exemple 5.3ce). Les jobs dans scénario suivent une charge de travail (workload) qui va varier dans des périodes de temps durant l'exécution en utilisant le script "workload-par.sh".

Listing 5.3: Programme d'un job parallèle whereami.jdl

```
1
2  [
3  JobType = ``Normal";
4  CpuNumber = 4;
5  Executable = ``mpi-whereami.sh";
6  Arguments = ``whereami openmpi";
7  StdOutput = ``mpi-test.out";
8  StdError = ``mpi-test.err";
9  InputSandbox = {"whereami.c", "mpi-whereami.sh"};
10 OutputSandbox = {"mpi-test.err","mpi-test.out", "whereami"};
11 ]
```

L'éxécution du scénario des jobs parallèles est montrée sur le tableau 5.6.

Table 5.6: Résultat du scénario des jobs parallèles sur 3 heures d'exécution

Workload jobs/minute	**méthode de prédiction**	**Future Value Response Time**	**MAE**	**Future Value Cpu Util WN01**	**MAE**	**Future Value Cpu Util WN02**	**MAE**	**Avg C1.cpu_idle /15 minutes**	**Avg C2.cpu_idle /15 minutes**
2	SMA	12	38.27	2.45	18.23	11.43	18	76.57 % 63.55 % 71.69 % 74.36 %	83.77 % 84.55 % 82.86 % 83.25 %
	EMA	3	11.27	2.5	15.09	11.32	17.5	79.52 % 78.83 % 78.46 % 79.27 %	88.87 % 85.39 % 84.54 % 84.25 %
4	SMA	9	34.48	2.1	27.97	10.69	20.88	71.39 % 71.93 % 73.97 % 72.75 %	88.38 % 83.39 % 83.51 % 82.41 %
	EMA	4	19.88	7.72	12.97	12.8	14.97	71.86 % 74.49 % 73.68 % 73.94 %	86.67 % 81.67 % 82.85 % 80.43 %
	SMA	8	21.97	15.64	22.16	11.62	22.86	77.23 % 77.58 % 77.05 %	82.02 % 83.31 % 80.01 %

Table 5.6: Résultat du scénario des jobs parallèles sur 3 heures d'exécution

Workload jobs/minute	méthode de prédiction	Future Value Response Time	MAE	Future Value Cpu Util WN01	MAE	Future Value Cpu Util WN02	MAE	Avg C1.cpu_idle /15 minutes	Avg C2.cpu_idle /15 minutes
								77.49 %	78.95 %
	EMA	10	15.11	**18.7**	15.43	13.82	16.11	69.01 % 47.42 % 65.75 % 64.26 %	78.31 % 74.66 % 70.90 % 66.67 %

Discussion des résultats L'exécution du scénario des jobs parallèles sur les différents modules, nous a permis d'évaluer notre approche. Le tableau présente chaque méthode de prédiction avec la prédiction des valeurs futures des métriques "temps de réponse" et "cpu utilisation", la moyenne de la métrique "cpu idle" issue du monitoring en temps réel, ainsi que la mesure de présicion "MAE" pour les valeurs prédites.

D'aprés le calcul du modèle de précision "MAE" appliqué sur les deux méthodes de prédiction (SMA et EMA), nous remarquons aussi que dans ce scénarion, la méthode de prédiction "EMA" donne de meilleurs résultats que la méthode de prédiction "SMA". Cela permet de sélectionner la méthode de prédiction "EMA" et utiliser ses valeurs prédites dans les règles de décision pour une mise à l'échelle.

La figure 5.15 montre que la file d'attente du service de calcul "ce-stack.grid.arn.dz", contient beaucoup de jobs en attente, ce qui necessite l'ajout d'autres worker nodes pour l'exécution de ces jobs. Cela est montré aussi sur le tableau 5.6 pour un workload égal à 6 jobs et une valeur future "CPU utilisation" pour le WN01 qui est arrivée au seuil défini dans la règle proactive. A cet effet, la règle proactive de la "cpu utilisation" s'est déclenchée automatiquement pour lancer un autoscaling d'ajout d'une intance worker node et l'envoi d'un email de notification à l'administrateur (voir la figure 5.16).

```
[root@ce-stack ~]# bjobs -u all
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
12130   dzgrid0 RUN   dzgrid     ce-stack.gr 1*wn01-stac *416578293 Jan 25 09:32
                                             2*ce-stack.grid.arn.dz
                                             1*wn02-stack.grid.arn.dz
12131   dzgrid0 PEND  dzgrid     ce-stack.gr             *301949998 Jan 25 09:32
12132   dzgrid0 PEND  dzgrid     ce-stack.gr             *450814473 Jan 25 09:32
12133   dzgrid0 PEND  dzgrid     ce-stack.gr             *074782922 Jan 25 09:32
12134   dzgrid0 PEND  dzgrid     ce-stack.gr             *265252752 Jan 25 09:32
12135   dzgrid0 PEND  dzgrid     ce-stack.gr             *295179922 Jan 25 09:32
12136   dzgrid0 PEND  dzgrid     ce-stack.gr             *010638007 Jan 25 09:32
12137   dzgrid0 PEND  dzgrid     ce-stack.gr             *228386506 Jan 25 09:32
12138   dzgrid0 PEND  dzgrid     ce-stack.gr             *935984367 Jan 25 09:32
```

Figure 5.15: Exécution des jobs sur les worker nodes en mode parallèle

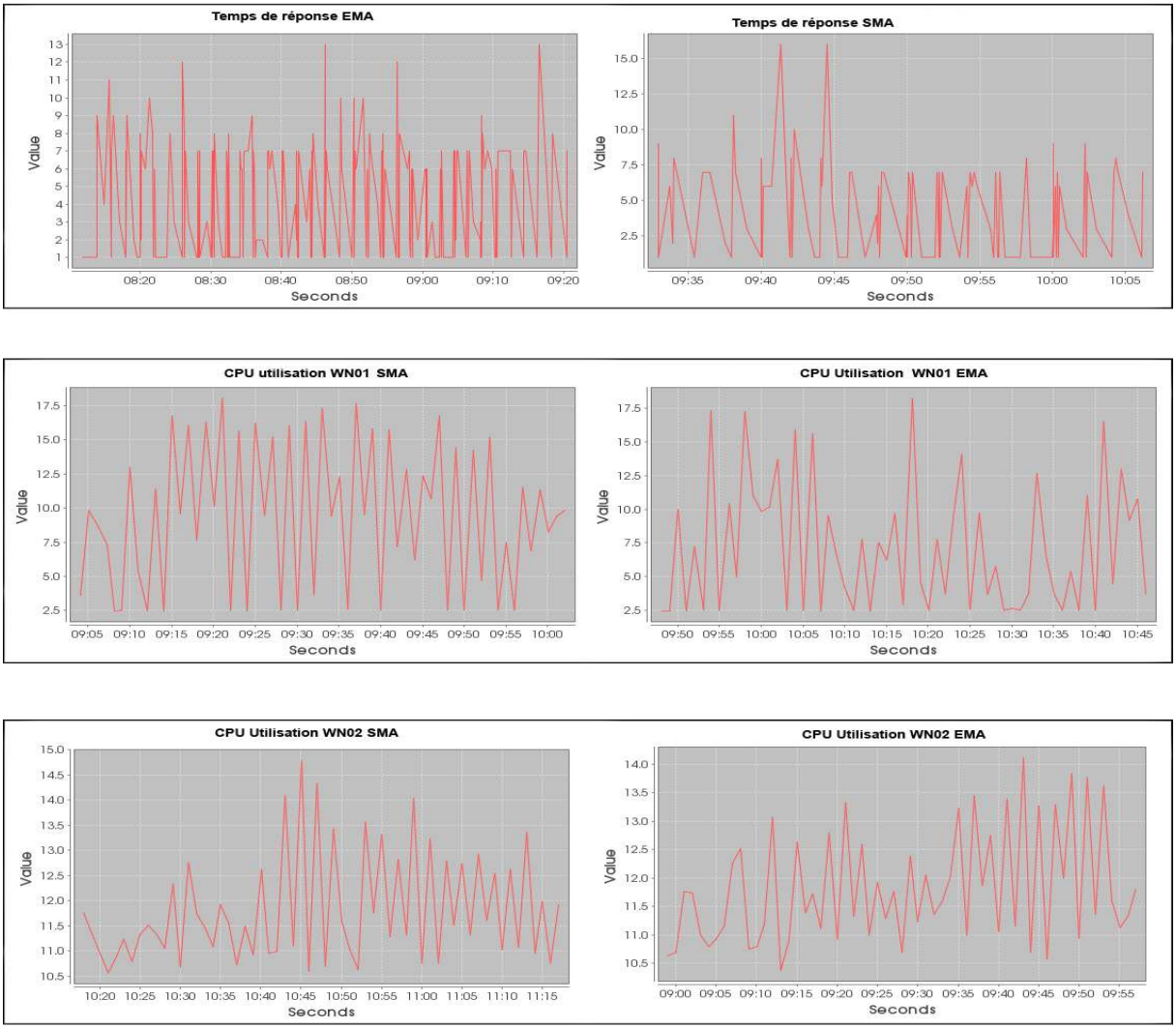

Figure 5.16: Histogrammes des prédiction des valeurs futures

5.4.2 Analyse de performance de l'approche

Nous avons sélectionné un ensemble de jobs de type séquentiel et parallèle qui s'exécutent pendant des périodes de temps selon des variations de workload (nombre de jobs/unité de temps). Nous avons choisi d'utiliser deux méthode de prédiction (SMA et EMA) pour prédire les valeurs futures de "CPU utilisation" et le "temps de réponse" puis sélectionner la meilleure méthode qui a la plus petite valeur d'erreur de prédiction pour la prise de décision et enfin une mise à l'échelle automatique. Les résultats de nos expérimentations (scénario des jobs séquentiels et scénario des jobs parallèles) et l'uilisation du modèle de précision "MAE" pour comparer les deux méthodes, ont montré que la méthode EMA (appelé aussi simple lissage exponentiel) a donné de meilleurs résultats que la méthode SMA dans un environnement "Grid-Cloud".

D'autre part, la méthode de prédiction EMA donne plus d'importance aux poids de la

time series les plus récents, ce qui la rend intéréssante dans le cas d'un court terme [37]. Ce cas de figure est similaire à nos scénarios, puisqu'on a choisi une fenêtre de prédiction de 1 heure de temps pour nos time series.

D'un autre côté, la précision des algorithmes dépendent de différents paramètres tels que la longueur de l'intervalle de monitoring , la taille de la fenêtre d'historique (fenêtre de prédiction) qui détermine la sensibilité de l'algorithme pour les tendances locales versus les tendances globales. Pour le choix de la fenêtre glissante (14 dans notre cas) et le pas (1 dans notre cas), on s'est basé sur les travaux de recherche effectués (voir le choix des paramètres 5.3.3.4). Le fonctionnement des différents modules de notre système de monitoring multi level installés sur les noeuds physiques et le module de performance installé sur la machine d'administraiton Cloud, nous a permis d'évaluer notre approche sur plusieurs aspects : la faisabilité de l'approche et sa performance.

5.5 Synthèse

Le travail présenté dans ce document consiste en la proposition d'une approche multi niveaux de monitoring basé modèle de performance des applications scientifiques qui s'exécutent sur des nouveaux services de calcul grille déployés dans un environnement cloud (plateforme grille virtualisée) selon l'approche "Grid-Cloud".

La mise en œuvre de notre solution a été détaillée dans ce chapitre. Nous avons commencé par présenter l'environnement dans lequel nous avons travaillé et les modules de notre approche. Nos besoins étaient multiples : Créer un site grille "DZ-ARN-Stack" sur une infrastructure de Cloud (plus précisément les services de calcul, "CE" et worker node) selon l'approche "Grid-Cloud", mettre en place un système de monitoring multi niveaux, modélisation de la performance des applications scientifiques, prédire l'utilisation des ressources en se basant sur des techniques de prédiction quantitatives "analyse des times series", prise de décision à base de règles proactives et réactives, et enfin le redimensionnement automatique par l'utilisation du moteur d'autoscaling que nous propose Openstack. Nous avons montré par la suite l'implémentation des différentes parties de notre travail.

Dans le déploiement réel de l'approche d'intégration "Grid-Cloud", nous avons atteint les objectifs suivants:

- Intégration de la virtualisation dans des infrastructures grille existantes.
- Déploiement des services grille au dessus des infrastructures virtuelles.
- Intégration des services cloud dans les infrastructures grille.
- Amélioration de la durée d'approvisionnement des ressources pour les services grille, par l'utilisation des appliances virtuelles (service glance) et les templates HOT (service heat).

Nous avons constaté que l'approche d'intégration simplifie la tâche de l'utilisateur, pour sélectionner, configurer, et gérer les ressources selon les besoins de l'application. Elle ajoute la flexibilité d'exploiter les ressources disponibles. Cependant, l'architecture résultante ajoute une complexité pour gérer le système globale, et la virtualisation des workers node entraîne inévitablement des pertes de performance par rapport à l'utilisation directe de ressources physiques. A cet effet, un monitoring capturant les informations à différents niveaux et une granularité de temps pour surveiller constamment, les ressources et le comportement des applications, suivi d'une prédiction de la performance de ces applications scientifiques sont nécessaires.

Dans l'ordre de valider la fonctionnalité et la performance de l'approche proposée, nous avons déployé notre approche qui se compose de deux unités importantes, l'unité de monitoring multi niveaux et l'unité du modèle de performance sur l'infrastructure de l'approche "Grid-Cloud".

Le déploiement de l'unité de monitorinbg multi niveaux est modulaire. Au niveau de chaque nouveau nœud compute physique, on peut installer un module "Data collector and selector". Il permet de fournir un "multilevel monitoring", qui surveille des types d'information hétérogènes, à partir des métriques de système de bas niveau (par exemple l'utilisation CPU, le trafic réseau, l'allocation de mémoire, etc.), jusqu'à des métriques de haut niveau (par exemple temps de réponse, débit, la latence), et qui sont collectées sur plusieurs niveaux (niveau physique, VM, et application) du Cloud et à différents intervalles de temps.

Quant au déploiement de l'unité du modèle de performance, il consiste en la modélisation de performance des applications scientifiques sur la base des méthodes des séries chronologiques pour fournir un moyen de prédiction et permettre une mise à l'echelle automatique. Pour cela, nous avons modélisé des séries chronologiques, utiliser deux méthodes de prédiction "SMA", "EMA" puis sélectionner la meilleure méthode en utilisant le modèle de précision "MAE". Une fois, la méthode de prédiction sélectionnée, la prédiction de ses valeurs futures est utilisée dans la prise de décision qui se base sur des règles proactives et réactives.

Les résultats de nos expérimentations et l'utilisation du modèle de précision "MAE" pour comparer les deux méthodes, ont montré que la méthode "EMA" a donné de meilleurs résultats que la méthode "SMA" dans un environnement "Grid-Cloud".

L'évaluation de l'approche à travers les tests effectués montre que le modèle de performance permet d'atteindre l'objectif qui est la planification et la gestion de ressources pour garantir une certaine performance en se basant sur des approches de modélisation et des techniques de prédiction. En plus, la combinaison des règles réactives et les règles proactives améliore la performance de l'approche dans la prise de décision pour l'ajustement des ressources.

Conclusion Générale

Dans ce mémoire, nous nous sommes penchés sur l'apport que pourrait avoir l'utilisation d'un modèle de performance à base de modélisation et prédiction dans un environnement "Grid-Cloud" afin de garantir une certaine performance aux applications scientifiques qui s'exécutent.

Le domaine de modélisation de la performance des applications pour la prédiction et la mise à l'échelle automatique dans un environnement virtualisé, où plusieurs machines virtuelles partagent les mêmes ressources physiques, est un sujet d'actualité et attractif pour les chercheurs et les scientifiques. Cependant, chacun d'eux propose son propre modèle de performance spécifique à ses besoins.

Dans notre projet nous avons proposé un système multi niveaux de monitoring basé modèle de performance dans un environnment "Grid-Cloud". Le système proposé permet de prédire la performance des applications scientifiques spécifiques à l'infrastructure DZ e-Science Grid et une mise à l'échelle automatique afin d'assurer aux utilisateurs de la grille, la haute disponibilité de leurs applications scientifiques, et offrir aux adminsitrateurs grille et Cloud la possibilité de planifier la gestion des ressources.

Dans le contexte de ce travail nous avons procédé à une étude comparative des deux paradigmes "Grid", "Cloud" qui a montré que la combinaison "Grid-Cloud" améliore la gestion de la grille dans son modèle actuel en bénéficiant des avantages Cloud. Pour adresser cet objectif, nous avons suivi l'approche "grille sur le Cloud" l'une des approches proposées dans la littérature.

Aprés le choix de l'approche d'intégration nous avons proposé un "système multi-niveaux de monitoring basé modèle de performance" pour la prédiction de performance et la prise de décision. Pour ce faire, nous avons étudié le monitoring dans le Cloud, nous avons fourni les principales propriétés d'un système de monitoring dans le Cloud, les concepts de base pour la construction d'un tel système ainsi que les contributions connexes prévues dans la littérature. Nous avons aussi présenté quelques solutions de monitoring en les classant en des catégories à usage général et spécifiques Cloud afin de mieux comprendre les outils et analyser historiquement leur évolution. Nous avons constaté que les solutions existantes se concentrent sur des métriques de bas niveau, et ne font pas saisir pleinement la performance réelle de l'application. Tandis que les solutions de monitoring applicatif sont des solutions spécifiques à des applications personnalisées et ne sont pas génériques.

A cet effet, nous avons évoqué le monitoring applicatif basé sur un modèle de performance qui traite et analyse les données. La modélisation de la performance d'une application dans un environement virtualisé et plus précisément dans le Cloud Computing est une tâche difficile et nécessaire. Les modèles de performance fournissent la capacité de prédire les performances des applications pour un ensemble donné de ressources matérielles. Les modèles de prédictions quantitatives sont utilisées principalement pour prédire la charge du travail ou l'utilisation des ressources. L'analyse des times series couvre un large éventail de méthodes qui suivent une approche prédictive et semble être un domaine de recherche prometteur.

Notre contribution a été de proposer un "système multi-niveaux de monitoring basé modèle de performance" qui traite et analyse les données collectées pour prédire la performance d'une application scientifique, et qui se compose de deux unités importantes:

Une unité de monitoring multi-niveaux - capturant constamment des métriques et des indicateurs de performance en temps réel et à différents niveaux qui renseignent sur l'état des applications et de l'infrastructure (physique et virtuelle) et la compréhension et le choix précis des métriques quantitatives quiévaluent la performance des applications.

Un modèle de performance - qui traite et analyse les données collectées, et qui est divisé en deux parties: La première partie, consiste en la modélisation de performance des applications scientifiques sur la base des méthodes des séries chronologiques pour fournir un moyen de prédiction et déterminer les besoins en ressources. Pour cela, nous avons modélisé des séries chronologiques, utiliser deux méthodes de prédiction "SMA", "EMA" puis sélectionner la meilleure méthode en utilisant le modèle de précision "MAE". Sur la base de ces valeurs prédites, la deuxième ètape consiste à décider de l'action appropriée à prendre en combinant des règles próactives et réactives que nous avons définies afin de faire une mise à l'échelle automatique et des notifications aux administrateurs.

Les résultats de nos expérimentations (scénario des jobs séquentiels et scénario des jobs parallèles) et l'utilisation du modèle de précision "MAE" pour comparer les deux méthodes, ont montré que la méthode "EMA" a donné de meilleurs résultats que la méthode "SMA" dans un environnement "Grid-Cloud". D'autre part, la combinaison des règles réactives qui se basent sur le monitoring en temps réel et les règles proactives qui se basent sur des valeurs prédites améliore la performance de l'approche dans la prise de décision pour l'ajustement des ressources.

La mise en oeuvre de l'approche proposée a montré la validation des objectifs que nous avons fixé:

- La virtualisation et la gestion des services de calcul (CE et Worker Nodes) dans l'infrastructure Cloud;

- La Création d'un site grille "DZ-ARN-Stack" sur une infrastructure Cloud;
- La collecte des métriques quantitatives hétérogènes sur plusieurs niveaux (métriques de bas niveau et haut niveau);
- Le filtrage des données de monitoring circulant sur le réseau par la compréhension et le choix précis des métriques quantitatives qui évaluent la performance des applications;
- L'analyse et traitement d'informations de monitoring en se basant sur des approches de modélisation;
- La prédiction des valeurs futurs pour l'utilisation des ressources en utilisant des techniques de prédictions;
- La mise à l'échelle automatique via des règles de décision proactives et réactives pour une haute disponibilité.

Perspectives

Au terme du travail que nous venons d'accomplir, des perspectives se précisent pour constituer d'éventuelles continuations. Ces perspectives peuvent être classés en deux catégories : les travaux réalisables dans un futur proche, et ceux réalisables dans un futur plus lointain.

A court terme, nous proposons :

- L'utilisation de plusieurs méthodes de prédiction pour avoir plus de choix dans la sélection de la meilleure méthode de prédiction afin d'améliorer la prise de décision pour la mise à l'échelle.
- L'amélioration de notre modèle de performance par rapport au paramètres des méthodes de prédiction, par exemple la taille de la fenêtre d'historique. Cela permet de donner de meilleures valeurs prédites.
- L'utilisation de plusieurs modèles de précision afin de mieux évaluer les méthodes de prédiction.
- L'exploration de plus de métriques par le modèle de performance pour mieux comprendre le comportement des applications scientifiques.
- La mise à l'échelle verticale (augmenter les ressources des machines virtuelles).

A long terme, nous souhaitons :

- Combiner plusieurs techniques prédictives pour améliorer la précision des valeurs prédites;
- Adapter le modèle de performance sur les Clouds fédérés (Clouds hybrides ou hétérogènes), par exemples sur différents Cloud IaaS privés participants.

Bibliography

[1] Compatibleone.

[2] Nagios.

[3] Amazon ec2, 2013.

[4] Emi. emi architecture and technology developement plan. Apr 2013.

[5] Google app engine, 2013.

[6] Windows azure, 2013.

[7] Sci flex project. role of a cep system in cloud computing, Aug 2008.

[8] Azurewatch, Jan. 2014.

[9] Monitis, Jan. 2014.

[10] Sflow, Oct. 2010.

[11] Giuseppe Aceto, Alessio Botta, Walter de Donato, and Antonio PescapÃÍ. Cloud monitoring: Definitions, issues and future directions. pages 63–67, 2012.

[12] Giuseppe Aceto, Alessio Botta, Walter De Donato, and Antonio Pescapè. Survey cloud monitoring: A survey. *Comput. Netw.*, 57(9):2093–2115, June 2013.

[13] A. M. Aljohani, D. R. W. Holton, Irfan-Ullah Awan, and J. S. Alanazi. Performance evaluation of local and cloud deployment of web clusters. In Leonard Barolli, Fatos Xhafa, and Makoto Takizawa, editors, *NBiS*, pages 274–278. IEEE Computer Society, 2011.

[14] Amazon. Amazon cloudwatch: Developer guide, 2009.

[15] Brian Amedro, Françoise Baude, Fabrice Huet, and Elton Mathias. Combining Grid and Cloud Resources by Use of Middleware for SPMD Application. In *2nd International Conference on Cloud Computing Technology and Science*, pages 177–184, Indianapolis, IN, United States, November 2010.

[16] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, April 2010.

[17] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. Above the clouds: A berkeley view of cloud computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009.

[18] L. Badger, T. Grance, R. Patt-Corner, and J.Voas. Draft cloud computing synopsis and recommendations. Technical Report 800-146, National Institute of Standards and Technology, 2011.

[19] R. Baker, M.and Buyya and D. Laforenza. Grids and grid technologies for wide-area distributed computing. *International Journal of Software : Practice and Experience (SPE)*, 32(15), Nov 2012.

[20] Enda Barrett, Enda Howley, and Jim Duggan. Applying reinforcement learning towards automating resource allocation and application scalability in the cloud. *Concurrency and Computation: Practice and Experience*, 25(12):1656–1674, 2013.

[21] W. Barth. Nagios: system and network monitoring. *No Starch Pr*, 2008.

[22] Mohamed N. Bennani and Daniel A. MenascÃľ. Resource allocation for autonomic data centers using analytic performance models. In *ICAC*, pages 229–240. IEEE Computer Society, 2005.

[23] G. Brunette and R. Mogull. Security Guidance for critical areas of focus in Cloud Computing V2. 1. *CSA (Cloud Security Alliance), USA. Online: http://www. cloud-securityalliance. org/guidance/csaguide. v2*, 1, 2009.

[24] R. Buyya, D. Abramson, J. Giddy, and G. Nimrod. An architecture for a resource management and scheduling system in a global computational grid. In *Proceedings of the 4th International Conference on High Performance Computing*, Asia Pacific Region, Beijing, China, 2000.

[25] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger. Economic models for resource management and scheduling in grid computing, concurrency and computation: Practice and experience. 14(13-15):1507–1542, 2002.

[26] R. Buyya, J. Broberg, and A. Goscinski. *Cloud Computings: Principles and Paradigms, chapter Performance Prediction For HPC on Clouds.* John Wiley, 2011.

[27] R. Buyya and S. Venugopal. A gentle introduction to grid computing and technologies. *Computer Society of India*, 2005.

[28] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud computing and emerging it platforms : Vision, hype, and reality for delivering computing as the 5th utility. *Elsevier : Future Generation Computer Systems*, 25(6):599âĂŞ616, June 2009.

[29] Eddy Caron, Frédéric Desprez, Adrian Muresan, and Luis Rodero-Merino. Using Clouds to Scale Grid Resources: An Economic Model. *Future Generation Computer Systems*, Volume 28(Issue 4):633–646, 2012.

[30] Alexandra Carpen-Amarie. Blobseer as a data-storage facility for clouds : self-adaptation, integration, evaluation, 2012.

[31] S.A. Chaves, R.B. Uriarte, and C.B. Westphall. Toward an architecture for monitoring private clouds, ieee communications magazine. *in: W. Gentzsch, L. Grandinetti, G. Joubert (Eds.), High Speed and Large Scale Scientific Computing, IOS*, 49:130–137, 2011.

[32] L. Cherkasova, D. Gupta, E. Ryabinkin, R. Kurakin, V. Dobretsov, and A. Vahdat. Optimizing grid site manager performance with virtual machines. In *Proceedings of the 3rd Conference on USENIX Workshop on Real, Large Distributed Systems*, WORLDS'06, Seattle, 2006. USENIX Association.

[33] N.M. Mosharaf Kabir Chowdhury and Raouf Boutaba. A survey of network virtualization. *Comput. Netw.*, 54(5):862–876, April 2010.

[34] S. Clayman, A. Galis, C. Chapman, G. Toffetti, L. Rodero-Merino, L.M. Vaquero, K. Nagin, and B. Rochwerger. Monitoring service clouds in the future internet. *ACM Trans*, 2010.

[35] S. Clayman, A. Galis, and L. Mamatas. Monitoring virtual networks with lattice. *in: Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP*, pages 239–246, 2010.

[36] Stuart Clayman, Alex Galis, Clovis Chapman, Giovanni Toffetti, Luis Rodero-Merino, Luis Miguel Vaquero, Kenneth Nagin, and Benny Rochwerger. Monitoring service clouds in the future internet. In Georgios Tselentis, Alex Galis, Anastasius Gavras, Srdjan Krco, Volkmar Lotz, Elena Paslaru Bontas Simperl, Burkhard Stiller, and Theodore B. Zahariadis, editors, *Future Internet Assembly*, pages 115–126. IOS Press, 2010.

[37] A. Coghlan. A little book of r for time series. 2015.

[38] A.W. Cooke, A.J.G. Gray, L. Ma, W. Nutt, J. Magowan, M. Oevers, P. Taylor, R. Byrom, L. Field, S. Hicks, J. Leake, M. Soni, A.J. Wilson, R. Cordenonsi, L. Cornwall, A. Djaoui, S. Fisher, N. Podhorszki, B.A. Coghlan, S. Kenny, and D. O'Callaghan. R-GMA: An information integration system for grid monitoring. In *CoopIS/DOA/ODBASE 2003*, volume 2888 of *LNCS*, pages 462–481, Catania, Italy, 2003. Springer.

[39] B. Cowie. Building a better network monitoring system. 2012.

[40] Frèdèric Desprez, Eddy Caron, Luis Caron, and Adrian Caron. Auto-scaling, load balancing and monitoring in commercial and open-source clouds. *Cloud Computing: Methodology, System and Applications, CRC Press*, pages 66–68, 2011.

[41] R. P. Doyle, J. S. Chase, O. M. Asad, W. Jin, and A. Vahdat. Model-based resource provisioning in a web service utility. *In USENIX Symposium on Internet Technologies and Systems*, 2003.

[42] Vincent C. Emeakaroha, Tiago C. Ferreto, Marco AurÃľlio Stelmar Netto, Ivona Brandic, and CÃŀsar A. F. De Rose. Casvid: Application level monitoring for sla violation detection in clouds. In Xiaoying Bai, Fevzi Belli, Elisa Bertino, Carl K. Chang, Atilla ElÃği, Cristina Cerschi Seceleanu, Haihua Xie, and Mohammad Zulkernine, editors, *COMPSAC*, pages 499–508. IEEE Computer Society, 2012.

[43] Kaniz Fatema, Vincent C. Emeakaroha, Philip D. Healy, John P. Morrison, and Theo Lynn. A survey of cloud monitoring tools: Taxonomy, capabilities and objectives. *J. Parallel Distrib. Comput.*, 74(10):2918–2933, 2014.

[44] I. Foster. *What is Grid ? A Three Point Checklist.* PhD thesis, Laboratoire National d'Argonne et l'Université de Chicago, Etats Unis, Juillet 2002.

[45] I. FOSTER and C. KESSELMAN. The grid : Blueprint for a new computing infrastructure. *Morgan Kaufmann,1ère édition*, Juillet 1998.

[46] I. Foster, Y. Zhao, I. Raicu, and S. Lu. Cloud computing and grid computing 360-degree compared. *in: Grid Computing Environments Workshop, 2008. GCE039;08, IEEE*, pages 1–10, 2008.

[47] Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *Int. J. High Perform. Comput. Appl.*, 15(3):200–222, August 2001.

[48] Michael P. Frank, Liviu Oniciuc, Uwe H. Meyer-Baese, and Irinel Chiorescu. A space-efficient quantum computer simulator suitable for high-speed fpga implementation, 2009.

[49] Simson Garfinkel and Harold Abelson. *Architects of the Information Society: 35 Years of the Laboratory for Computer Science at Mit*. MIT Press, Cambridge, MA, USA, 1999.

[50] R. P. Goldberg. Architecture of virtual machines. In *Proc. ACM International Workshop on virtual computer systems*, pages 74–112, New York,NY, USA, 1973.

[51] Chunye Gong, Jie Liu, Qiang Zhang, Haitao Chen, and Zhenghu Gong. The characteristics of cloud computing. In *Proceedings of the 2010 39th International Conference on Parallel Processing Workshops*, ICPPW '10, pages 275–279, Washington, DC, USA, 2010. IEEE Computer Society.

[52] Ghanbari Hamoun. Autonomic mechanisms in cloud computing ecosystems. August 2011.

[53] Ghanbari Hamoun, Simmons Bradley, Litoiu Marin, and Iszlai Gabriel. Exploring alternative approaches to implement an elasticity policy. In *Proceedings of the Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 716 – 723, july 2011.

[54] P. Hasselmeyer and N. dâĂŹHeureuse. Towards holistic multi-tenant monitoring for virtual data center. *in: Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/ IFIP*, pages 350–356, 2010.

[55] Z. Hill and M. Humphrey. A quantitative analysis of high performance computing with amazon's ec2 infrastructure: The death of the local cluster? In *Grid Computing, 2009 10th IEEE/ACM International Conference on*, pages 26–33, Oct 2009.

[56] H.Lim, B.Shivnath, and C.Jeffrey. Automated control for elastic storage. In *Proceedings of the 7th International Conference on Autonomic Computing*, pages 1–10, New York, NY, USA, 2010.

[57] Bastian Hoßbach, Bernd Freisleben, and Bernhard Seeger. Reaktives cloud monitoring mit complex event processing. *Datenbank-Spektrum*, 12(1):33–42, 2012.

[58] Jinhui Huang, Chunlin Li, and Jie Yu. Resource prediction based on double exponential smoothing in cloud computing. In *Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference on*, pages 2056–2060, April 2012.

[59] Emir Imamagic and Dobrisa Dobrenic. Grid infrastructure monitoring system based on nagios. In *Proceedings of the 2007 Workshop on Grid Monitoring*, GMW '07, pages 23–28, New York, NY, USA, 2007. ACM.

[60] S. Jha, Andre Merzky, and Geoffrey C. Fox. Using clouds to provide grids with higher levels of abstraction and explicit support for usage modes. *Concurrency and Computation: Practice & Experience*, 21(8):1087–1108, 06/2009 2009.

[61] Haihua Jiang, Hai Lv, Nan Wang, and Ruihua Di. A performance monitoring solution for distributed application system based on jmx. In *Grid and Cooperative Computing (GCC), 2010 9th International Conference on*, pages 124–127, Nov 2010.

[62] G. Katsaros, R. KÃijbert, and G. Gallizo. Building a service-oriented monitoring framework with rest and nagios. *in:2011 IEEE International Conference on Services Computing (SCC)*, pages 426–431, 2011.

[63] Md. Tanzim Khorshed, A.B.M. Shawkat Ali, and Saleh A. Wasimi. A survey on gaps, threat remediation challenges and some thoughts for proactive attack detection in cloud computing. *Future Generation Computer Systems*, 28(6):833 – 851, 2012. Including Special sections SS: Volunteer Computing and Desktop Grids and SS: Mobile Ubiquitous Computing.

[64] B. Konig, J.M. Alcaraz Calero, and J. Kirschnick. Elastic monitoring framework for cloud infrastructures. *ACM Trans*, 2012.

[65] Georgios Kornaros and Dionisios Pnevmatikatos. A survey and taxonomy of on-chip monitoring of multicore systems-on-chip. *ACM Trans. Des. Autom. Electron. Syst.*, 18(2):17:1–17:38, April 2013.

[66] J. Krizanic, A. Grguric, M. Mosmondor, and P. Lazarevski. Load testing and performance monitoring tools in use with AJAX based web applications. 2010.

[67] Sajib Kundu, Raju Rangaswami, Kaushik Dutta, and Ming Zhao. Application performance modeling in a virtualized environment. In Matthew T. Jacob, Chita R. Das, and Pradip Bose, editors, *HPCA*, pages 1–10. IEEE Computer Society, 2010.

[68] S. LACOUR. *Contribution à l'automatisation du déploiement d'applications sur les grilles de calcul*. PhD thesis, Université de Rennes 1, IRISA, Rennes, France, Décembre 2005.

[69] H. Ladour. Plateforme de monitoring des "cloud services " orientée "grid services". Master's thesis, Université Abderrahmane Mira de Bejaia Faculté des Sciences Exactes Département d'Informatique, 2013.

[70] R. Lee. Smart system resource monitoring in cloud. *iBusiness*, 4(1):34–42, 2012.

[71] Philipp Leitner, Christian Inzinger, Waldemar Hummer, Benjamin Satzger, and Schahram Dustdar. Application-Level Performance Monitoring of Cloud Services Based

on the Complex Event Processing Paradigm. In *IEEE International Conference on Service-Oriented Computing and Applications (SOCA'12)*, 2012.

[72] D. liang Lee, S.-Y. Yang, and Y.-J. Chang. Developing an active mode of network management system with intelligent multi-agent techniques. *in: Pervasive Computing (JCPC), 2009 Joint Conferences on*, pages 77–82, 2009.

[73] Fang Liu, Jin Tong, Jian Mao, Robert Bohn, John Messina, Lee Badger, and Dawn Leaf. *NIST Cloud Computing Reference Architecture: Recommendations of the National Institute of Standards and Technology (Special Publication 500-292).* CreateSpace Independent Publishing Platform, USA, 2012.

[74] T. Lorido-Botran, J.Miguel-Alonso, and J.A. Lozano. Auto-scaling techniques for elastic applications in cloud environments. Technical Report EHU-KAT-IK, Technical Report,Department of Computer Architecture and Technology, UPV/EHU, Basque Contry, sept 2012.

[75] D. Luckham. The power of events: An introduction to complex event processing in distributed enterprise systems. *Addison-Wesley Professional*, May 2002.

[76] M. Mansouri-Samani and M. Sloman. Monitoring distributed systems. *Network, IEEE*, 7(6):20–30, Nov 1993.

[77] M. Massie, B. Li, and V. Vuksan. *Monitoring with Ganglia.* Oreily, November 2012.

[78] M. L. Massie, B. N. Chun, and D. E. Culler. The Ganglia Distributed Monitoring System: Design, Implementation, and Experience. *Parallel Computing*, 30(7), July 2004.

[79] A. Meddeb. Internet qos: Pieces of the puzzle. *Communications Magazine, IEEE*, 48(1):86–94, January 2010.

[80] Y. Mei, L. Liu, X. Pu, and S. Sivathanu. Performance measurements and analysis of network i/o applications in virtualized cloud. *in: 2010 IEEE 3rd International Conference on Cloud Computing (CLOUD)*, pages 59–66, 2010.

[81] P. Mell and T. Grance. The nist definition of cloud computing. Technical Report 15, National Institute of Standards and Technology, 2009.

[82] P. Mell and T. Grance. The nist definition of cloud computing. *Nist Special Publication*, 145, 2011.

[83] D.A. MenascÃĺ and V.A.F. Almeida. Capacity planning for web services: metrics, models, and methods. *Prentice Hall*, 2002.

[84] Daniel A. Menasce, Lawrence W. Dowdy, and Virgilio A. F. Almeida. *Performance by Design: Computer Capacity Planning By Example*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.

[85] Haibo Mi, Huaimin Wang, Gang Yin, Yangfan Zhou, Dianxi Shi, and Lin Yuan. Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers. In *Services Computing (SCC), 2010 IEEE International Conference on*, pages 514–521, July 2010.

[86] Rizwan Mian, Patrick Martin, and Jose Luis Vazquez-Poletti. Provisioning data analytic workloads in a cloud. *Future Generation Computer Systems*, 2012.

[87] Jesùs Montes, Alberto Sànchez, Bunjamin Memishi, Marîa S. Pérez, and Gabriel Antoniu. Gmone: A complete approach to cloud monitoring. *Future Generation Computer Systems*, 29(8):2026 – 2040, 2013. Including Special sections: Advanced Cloud Monitoring Systems and amp; The fourth {IEEE} International Conference on e-Science 2011 âĂŤ e-Science Applications and Tools and amp; Cluster, Grid, and Cloud Computing.

[88] Susanta Nanda and Tzi cker Chiueh. A survey of virtualization technologies. Technical report, 2005.

[89] Harvey B. Newman, Iosif Legrand, Philippe Galvez, Ramiro Voicu, and Catalin Cirstoiu. Monalisa : A distributed monitoring service architecture. *CoRR*, cs.DC/0306096, 2003.

[90] The globus alliance, 2012.

[91] Rihards Olups. *Zabbix 1.8 Network Monitoring*. Packt Publishing, 2010.

[92] Pradeep Padala, Kai-Yuan Hou, Kang G. Shin, Xiaoyun Zhu, Mustafa Uysal, Zhikui Wang, Sharad Singhal, and Arif Merchant. Automated control of multiple virtualized resources. In *Proceedings of the 4th ACM European Conference on Computer Systems*, EuroSys '09, pages 13–26, New York, NY, USA, 2009. ACM.

[93] C. Pape and R. Trommer. Monitoring vmware-based virtual infrastructures with opennms. 2012.

[94] Sang-Min Park and Marty Humphrey. Self-tuning virtual machines for predictable escience. In Franck Cappello, Cho-Li Wang, and Rajkumar Buyya, editors, *CCGRID*, pages 356–363. IEEE Computer Society, 2009.

[95] BodÂťÄśk Peter, Griffith Rean, Sutton Charles, Fox Armando, Jordan Michael, and Patterson David. Statistical machine learning makes automatic control practical for internet datacenters. In *Proceedings of the Hot Topics in Cloud Computing (HotCloud 09), Usenix Assoc.*, page 75âĂŞ80, 2009.

[96] M. Rak, S. Venticinque, T. Mahr, G. Echevarria, and G. Esna. Cloud application monitoring: the mosaic approach. *in: 2011 IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom), IEEE*, pages 758–763, 2011.

[97] Jia Rao, Xiangping Bu, Cheng-Zhong Xu, Leyi Wang, and George Yin. Vconf: A reinforcement learning approach to virtual machines auto-configuration. In *Proceedings of the 6th International Conference on Autonomic Computing*, pages 137–146, New York, NY, USA, 2009. ACM.

[98] Jan S. Rellermeyer, Gustavo Alonso, and Timothy Roscoe. Building, deploying, and monitoring distributed applications with eclipse and r-osgi. In *Proceedings of the 2007 OOPSLA Workshop on Eclipse Technology eXchange*, eclipse '07, pages 50–54, New York, NY, USA, 2007. ACM.

[99] T. Rings, G. Caryer, J. Gallop, J. Grabowski, T. Kovacikova, S. Schulz, and I. Stokes-Rees. Opportunities for integration with the next generation network. *Journal of Grid Computing.*, 2009.

[100] J. Shao and Q. Wang. A performance guarantee approach for cloud applications based on monitoring. *in: Computer Software and Applications Conference Workshops (COMPSACW), 2011 IEEE 35th Annual*, pages 25–30, 2011.

[101] J.E. Smith and R. Nair. The architecture of virtual machines. *Computer*, 38(5):32–38, 2005.

[102] J. Spring. Monitoring cloud computing by layer, part 1. *IEEE Security and Privacy*, 9(2):66–68, 2011.

[103] J. Spring. Monitoring cloud computing by layer, part 2. *IEEE Security and Privacy*, 9(3):52–55, 2011.

[104] Srikanth Sundaresan, Walter de Donato, Nick Feamster, Renata Teixeira, Sam Crawford, and Antonio Pescapè. Broadband internet performance: A view from the gateway. In *Proceedings of the ACM SIGCOMM 2011 Conference*, SIGCOMM '11, pages 134–145, New York, NY, USA, 2011. ACM.

[105] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[106] K. Tanr. Introduction to grid computing. *CRC Press 2009, ISBN: 978-1-4200-7406-2*, 2009.

[107] A. Tchana. *Système d'Administration Autonome Adaptable : application au Cloud*. PhD thesis, Mathématiques Informatique Télécommunications (MITT), Nov 2011.

[108] D. Trihinas, G. Pallis, and M.D. Dikaiakos. Jcatascopia: Monitoring elastically adaptive applications in the cloud. In *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on*, pages 226–235, May 2014.

[109] Bhuvan Urgaonkar, Giovanni Pacifici, Prashant Shenoy, Mike Spreitzer, and Asser Tantawi. Analytic modeling of multitier internet applications. *ACM Trans. Web*, 1(1):45, 46, 137, may 2007.

[110] Luis M. Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: Towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, 39(1):50–55, December 2008.

[111] A. Viratanapanu, A. K. A. Hamid, Y. Kawahara, and T. Asami. On demand fine grain resource monitoring system for server consolidation. *Kaleidoscope: Beyond the Internet? âĂŞ IFNS 2010 ITU-T. IEEE*, pages 1 – 8, 2010.

[112] I.H. Witten and E. Frank. Data mining: Practical machine learning tools and techniques. *(Second Edition)*, 2005.

[113] Lamia Youseff, Richard Wolski, Brent C. Gorda, and Chandra Krintz. Paravirtualization for HPC systems. In *Frontiers of High Performance Computing and Networking - ISPA 2006 Workshops, ISPA 2006 International Workshops, FHPCN, XHPC, S-GRACE, GridGIS, HPC-GTP, PDCE, ParDMCom, WOMP, ISDF, and UPWN, Sorrento, Italy, December 4-7, 2006, Proceedings*, pages 474–486, 2006.

[114] K. Yousri. *SLA-driven cloud elasticity management approach.* PhD thesis, Ecole des Mines de Nantes, France, Dec 2013.

[115] Serafeim Zanikolas and Rizos Sakellariou. A taxonomy of grid monitoring systems. *Future Gener. Comput. Syst.*, 21(1):163–188, January 2005.

[116] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. *J. Internet Services and Applications*, 1(1):7–18, 2010.

[117] Gong Zhenhuan and Wilkes Xiaohui, Gu andJohn. Press : Predictive elastic resource scaling for cloud systems. In *Proceedings of the 6th IEEE/IFIP International Conference on Network and Service Management (CNSM 2010)*, pages 9–16, Canada, 2010.

[118] Dimitrios Zissis and Dimitrios Lekkas. Addressing cloud computing security issues. *Future Gener. Comput. Syst.*, 28(3):583–592, March 2012.

Appendices

APPENDIX A

OpenStack : Plate-forme cloud

A.1 Introduction

OpenStack est un ensemble de logiciels open source permettant de déployer des infrastructures de cloud computing (infrastructure en tant que service). La technologie possède une architecture modulaire composée de plusieurs projets corrélés (Nova, Swift, Glance...) qui permettent de contrôler les différentes ressources des machines virtuelles telles que la puissance de calcul, le stockage ou encore le réseau inhérents au centre de données sollicité.

Le projet est porté par la Fondation OpenStack, une organisation non-commerciale qui a pour but de promouvoir le projet OpenStack ainsi que de protéger et d'aider les développeurs et toute la communauté OpenStack.

A.2 Historique

En juillet 2010, Rackspace Hosting et la NASA ont lancé conjointement un nouveau projet open source dans le domaine du cloud computing sous le nom d'OpenStack. L'objectif du projet OpenStack est de permettre à toute organisation de créer et d'offrir des services de cloud computing en utilisant du matériel standard. La première version livrée par la communauté, dont le surnom est Austin, fut disponible seulement quatre mois après. Il est prévu de livrer régulièrement des mises à jour logicielles à quelques mois d'intervalle. De nombreux membres rejoignent le projet, En 2014, la communauté OpenStack compte 5 600 membres et 850 organisations.

Table A.1: Versions d'OpenStack

Nom	**Date**	**Composants inclus**
Austin	21 octobre 2010	Nova, Swift
Bexar	3 février 2011	Nova, Glance, Swift
Cactus	15 avril 2011	Nova, Glance, Swift
Diablo	22 septembre 2011	Nova, Glance, Swift
Essex	5 avril 2012	Nova, Glance, Swift, Horizon, Keystone

Table A.1: Versions d'OpenStack

Nom	**Date**	**Composants inclus**
Folsom	27 septembre 2012	Nova, Glance, Swift, Horizon, Keystone, Quantum, Cinder
Grizzly	4 avril 2013	Nova, Glance, Swift, Horizon, Keystone, Quantum, Cinder
Havana	22 octobre 2013	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Heat, Ceilometer
Icehouse	17 avril 2014	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Heat, Ceilometer, Trove
Juno	16 octobre 2014	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Heat, Ceilometer, Trove, Sahara
Kilo	30 avril 2015	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Heat, Ceilometer, Trove, Sahara, Ironic
Liberty	26 octobre 2015	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Heat, Ceilometer, Trove, Sahara, Ironic, SearchLight, Designate, Zaqar, Barbican, Manila

A.3 Composants OpenStack

OpenStack possède une architecture modulaire, comme l'illustre la figure, qui comprend de nombreux composants, que nous détaillons brièvement ci-dessous :

Calcul : Nova - Nova est une des briques principales d'Openstack. Son but est de gérer les ressources de Calcul des infrastructures. Pour cela, nova contrôle les hyperviseurs par l'intermédiaire de la libvirt ou directement par les API de certains hyperviseurs.

Stockage objet : Swift - Le stockage objet d'OpenStack s'appelle Swift. C'est un système de stockage de données redondant et évolutif. Les fichiers sont écrits sur de multiples disques durs répartis sur plusieurs serveurs dans un Datacenter. Il s'assure de la réplication et de l'intégrité des données au sein du cluster. Le Cluster Swift évolue horizontalement en rajoutant simplement de nouveaux serveurs.

Stockage bloc : Cinder - Le service de stockage en mode bloc d'OpenStack s'appelle Cinder. Il fournit des périphériques persistants de type bloc aux instances OpenStack. Il gère les opérations de création, d'attachement et de détachement de ces périphériques sur les serveurs.

Le réseau : Neutron - Le service Neutron d'Openstack (anciennement Quantum) permet de gérer et manipuler les réseaux et l'adressage IP au sein d'OpenStack. Grâce à Neutron, les utilisateurs peuvent créer leurs propres réseaux, contrôler le trafic à travers des groupes de sécurité (security groups) et connecter leurs instances à un ou plusieurs réseaux. Neutron gère aussi l'adressage IP des instances en leur assignant des adresses IP statiques ou par l'intermédiaire du service DHCP. Il fournit aussi un service d'adresse IP flottante que l'on peut assigner aux instances afin d'assurer une connectivité depuis Internet. Ces adresses IP flottantes peuvent être réassignées à d'autres instances en cas de maintenance ou de défaillance de l'instance originelle.

Tableau de bord : Horizon - OpenStack fournit un tableau de bord qui s'appelle Horizon. Il s'agit d'une application web qui permet aux utilisateurs et aux administrateurs de gérer leurs Clouds à travers d'une interface graphique. Comme toutes les briques d'OpenStack cette application est libre et il n'est donc pas rare de voir des versions modifiées par les fournisseurs de Cloud ou par d'autres sociétés commerciales ne seraitce que pour y faire apparaitre leur nom et logo, mais aussi pour y intégrer leurs systèmes de métrologie ou de facturation par exemple.

Service d'identité : Keystone - Le service d'identité d'OpenStack s'appelle Keystone. Il fournit un annuaire central contenant la liste des services et la liste des utilisateurs d'Openstack ainsi que leurs rôles et autorisations. Au sein d'Openstack tous les services et tous les utilisateurs utilisent Keystone afin de s'authentifier les uns avec les autres. Keystone peut s'interfacer avec d'autre service d'annuaire comme LDAP. Il supporte plusieurs formats d'authentification comme les mots de passe et autres.

Service d'image : Glance - Le service d'image d'OpenStack s'appelle Glance. Il permet la découverte, l'envoi et la distribution d'image disque vers les instances. Les images stockées font office de modèle de disque. Le service glance permet aussi de stocker des sauvegardes de ces disques. Glance peut stocker ces images disques de plusieurs façons : dans un dossier sur serveur, mais aussi à travers le service de stockage objet d'OpenStack ou dans des stockages décentralisés comme Ceph. Glance ne stocke pas seulement des images, mais aussi des informations sur celles-ci, les métadonnées. Ces métadonnées sont par exemple le format du disque (comme QCOW2 ou RAW) ou les conteneurs de celles-ci (OVF par exemple).

Télémétrie : Ceilometer - Le service de télémétrie d'OpenStack s'appelle Ceilometer. Il permet de collecter différentes métriques sur l'utilisation du Cloud. Par exemple il permet de récolter le nombre d'instances lancé dans un projet et depuis combien de temps. Ces métriques peuvent être utilisées pour fournir des informations nécessaires à un système de facturation par exemple. Ces métriques sont aussi utilisées dans les applications ou par d'autres composants d'Openstack pour définir des actions en fonction de certains seuils comme avec le composant d'orchestration.

Orchestration : Heat - Heat est le composant d'orchestration d'Openstack. Il permet de décrire une infrastructure sous forme de modèles. Dans Heat, ces modèles sont appelés des "stack". Heat consomme ensuite ces modèles pour aller déployer l'infrastructure décrite sur Openstack. Il peut aussi utiliser les métriques fournies par ceilometer pour décider de créer des instances supplémentaires en fonction de la charge d'une application par exemple.

A.4 Architectures OpenStack

A.4.1 Architecture logique

D'un point de vue logique, OpenStack peut être représenté a minimum avec trois services : un bloc de compute, un bloc de Networking et un bloc de Storage. Ces services utilisent un bus de communication pour communiquer entre eux. Ils exposent publiquement leurs fonctionnalités via des API. Un dashboard permet de se connecter aux services avec un portail présentant une IHM (interface homme-machine).

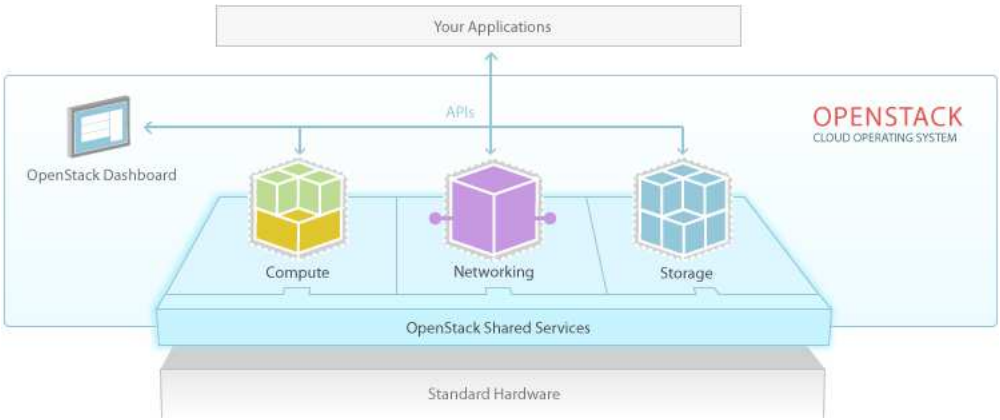

Figure A.1: Architecture logique

A.4.2 Architecture conceptuelle

D'un point de vue conceptuel, OpenStack est composé de services reliés les uns aux autres autour de deux services qui communiquent avec tous les autres modules : Horizon et Keystone.

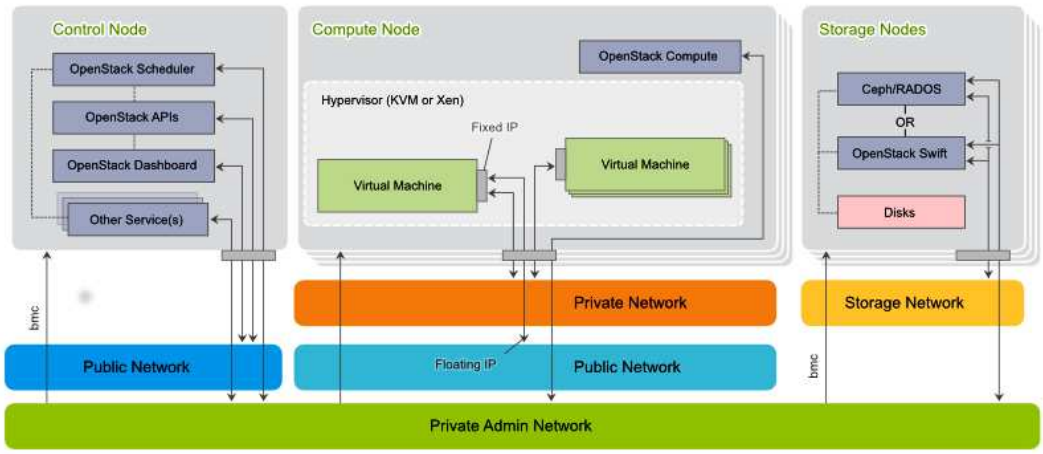

Figure A.2: Architecture technique

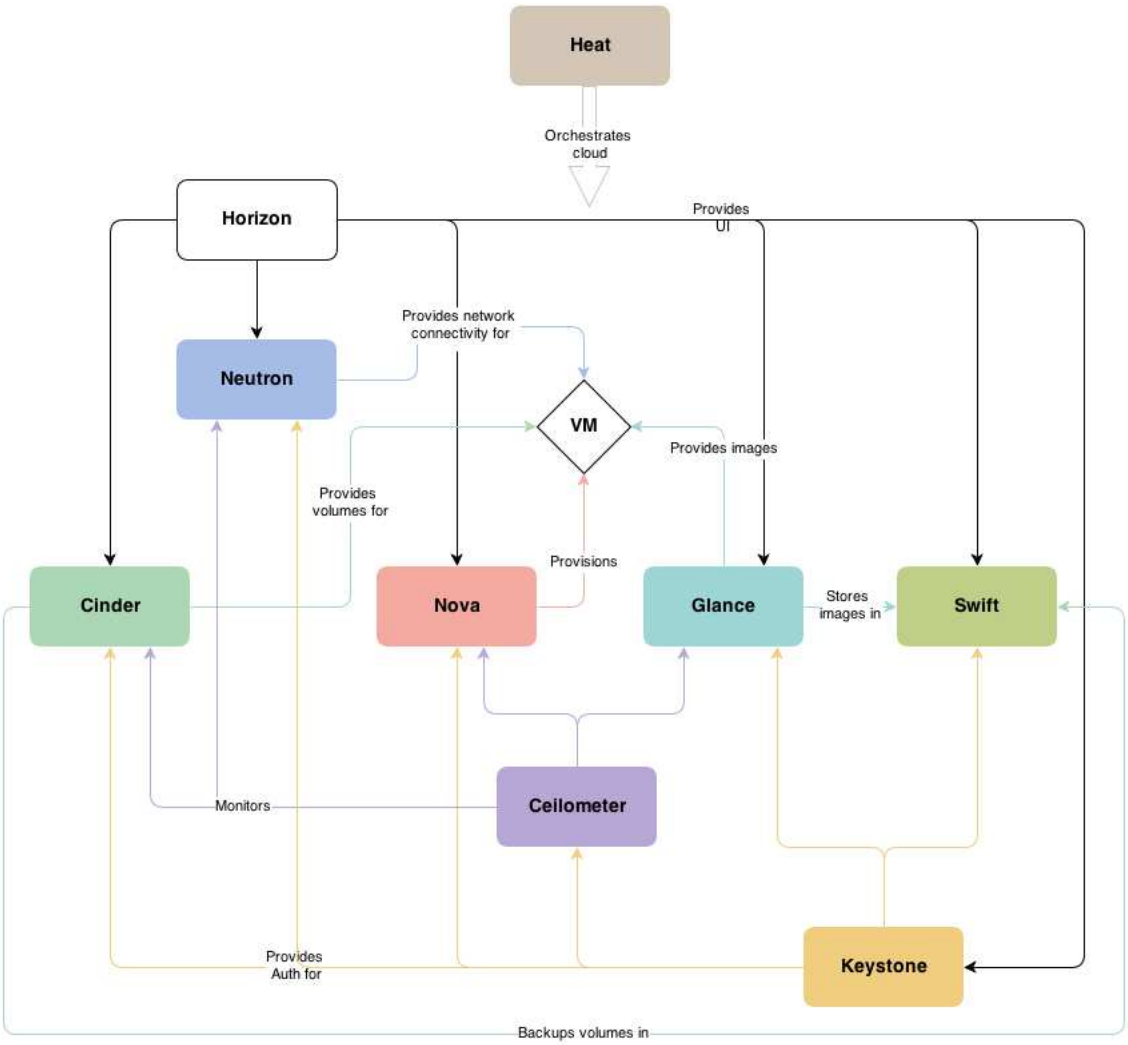

Figure A.3: Architecture conceptuelle

A.4.3 Architecture technique

D'un point de vue technique, OpenStack fonctionne soit en environnement virtuel, soit en environnement Bare-Metal ou les deux en fonction des services installés.

A.5 Installation OpenStack

A.5.1 Lancer une instance sur le cloud OpenStack

Pour lancer une instance il faut,