PARTAGE EFFICACE DES RESSOURCES DE CALCUL DANS LE NUAGE INFORMATIQUE

par

Achraf LABIDI

MÉMOIRE PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
COMME EXIGENCE PARTIELLE À L'OBTENTION DE LA MAÎTRISE
AVEC MÉMOIRE EN GÉNIE CONCENTRATION : GÉNIE LOGICIEL
M. Sc. A.

MONTRÉAL, LE 11 JUILLET 2017

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

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

M. Kim Khoa Nguyen, codirecteur de mémoire
Département de génie électrique à l'École de technologie supérieure

M. Ghyslain Gagnon, président du jury
Département de génie électrique à l'École de technologie supérieure

M. Jia Yuan Yu, examinateur externe
Université de Concordia

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

LE 5 JUILLET 2017

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Mes remerciements s’adressent en premier lieu à mon directeur de mémoire, M. Mohamed Cheriet, pour son soutien et ses conseils qui m'ont été indispensables au long de mes recherches. Je remercie également mon codirecteur M. Kim Khoa Nguyen, pour ses efforts et son encadrement efficace impactant positivement le déroulement de mon projet de recherche.

Je tiens aussi à exprimer ma gratitude à Saida Khazri pour son aide précieuse et ses connaissances qu’elle a partagées avec moi durant ma maîtrise.

Je remercie mes collègues du laboratoire Synchromédia pour leur disponibilité ainsi que l'ensemble des membres du jury pour leur participation à l’évaluation de ce travail.

Je profite de cette occasion pour exprimer ma reconnaissance à ma famille et mes amis, en particulier à mes parents, Nour Elhouda et Hedi. Je les remercie profondément de m'avoir accordé leur support moral.

PARTAGE DES RESSOURCES DE CALCUL DANS LE NUAGE INFORMATIQUE

Achraf LABIDI

RÉSUMÉ

L'informatique en nuage ou l'infonuagique est apparue comme un nouveau paradigme capable de gérer une infrastructure informatique à grande échelle. Toutefois, la plupart des infrastructures infonuagiques existantes ne sont pas exploitées efficacement, et la sur-provision de ressources est un problème émergent.

En raison des exigences variables au cours du temps des ressources virtuelles, les plateformes physiques pourraient être utilisées de manière inadéquate, ce qui entraîne des coûts opérationnels supplémentaires. Des techniques de migration ont été proposées pour améliorer l'utilisation des ressources physiques, par exemple la consolidation des ressources virtuelles sur les ressources physiques. Les travaux antérieurs motivés par des objectifs énergétiques et d'équilibrage de charge sont souvent limités à une seule technique de migration. Ce mémoire présente un modèle de migration optimisé pour les machines virtuelles basé sur l'état courant et futur de l'utilisation des ressources physiques tout en considérant les différentes techniques de migration. Cette future utilisation s'appuie sur la prédiction de la charge de travail. Notre modèle a pour fin de minimiser le coût opérationnel lors du partage de l'infrastructure sous-jacente.

L'expérimentation menée dans le cadre de ce mémoire montre la capacité de notre modèle pour réaliser un meilleur partage des ressources physique et ce, en réduisant le coût opérationnel de 16% en comparaison à une solution existante. L'expérimentation montre également que l'intégration de différentes techniques de migration dans le même modèle implique une optimisation globale par rapport à l'intégration d'une seule technique.

Mots-clés: Infonuagique, partage de ressources, migration de VMs, Openstack, virtualisation.

SHARING COMPUTING RESOURCES IN CLOUD

Achraf LABIDI

ABSTRACT

Cloud computing has emerged as a new computational paradigm capable of managing large-scale IT infrastructure. However, most of the existing cloud infrastructures are not efficiently operated, and resource overprovisioning is an emerging issue.

Due to the time-varying requirements of virtual resources, physical platforms might be inefficiently used, resulting in additional operational costs. Migration techniques were proposed to improve physical resource utilization, e.g. consolidating virtual resources on physical ones. Prior work driven by energy aware and load balancing objectives are often restricted to a single migration technique. This thesis presents an optimized migration model for virtual machines based on current and future states of physical resource usage while considering multiple migration techniques. The future usage is based on workload prediction. Our model aims at minimizing the operational cost when sharing the underlying infrastructure.

The experimentations carried out in this work demonstrate the ability of our model to perform better physical resources sharing compared to other models based on one single migration technique and with no prediction techniques. With our model, we managed to reduce the operational cost by 16% on average compared to an existing solution.

Keywords: Cloud computing, resource sharing, VM migrations, Openstack, virtualization.

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ACF	Autocorrelation Function
API	Application Programming Interface
AR	Autoregressive
ARIMA	Autoregressive Integrated Moving Average
ARMA	Autoregressive and Moving Average
BLM	Bloc Live Migration
CM	Cold Migration
CPU	Central Processing Unit
DHCP	Dynamic Host Configuration Protocol
EC2	Elastic Compute Cloud
IPMI	Intelligent Platform Management Interface
IaaS	Infrastructure as a Service
KVM	Kernel-based Virtual Machine
LM	Live Migration
MA	Moving Average
MM	Minimization of Migration
NFS	Network File System
OVS	Open vSwitch
PACF	Partial Autocorrelation Function
QoS	Quality of Service
RMSE	Root-Mean-Square Deviation
SLA	Service Level Agreement

SSH	Secure Shell
VBLM	Volume Block Live Migration
VLAN	Virtual LAN
VM	Virtual Machine
VXLAN	Virtual Extensible LAN

CHAPITRE 1

INTRODUCTION

1.1 Contexte

Nous vivons dans une société de plus en plus connectée et automatisée, les environnements de nos jours concrétisent cette tendance à travers la liaison des ordinateurs et les différents équipements modernes à des tâches courantes (Alam, Reaz et Ali, 2012), ceci dit, l'avancement des réseaux informatiques et mobiles joue un rôle primordial pour établir un terrain commun dans lequel les appareils et les services sont parfaitement coopérés.

Nombreux sont les services que nous manipulons à une fréquence quotidienne, la nature de certains en termes de consommation des ressources de calculs, de réseau et de stockage exige un endroit plus puissant, tel que le nuage informatique qui offre des ressources ayant des capacités considérables ainsi que la migration. Par exemple, une application de vidéo streaming ou de réalité virtuelle nécessite des ressources performantes et une migration qui n'affecte pas son exécution telle que la migration à chaud. Ou encore, l'application de reconnaissance de visage déployé dans un contexte de sécurité (Jang et al., 2014) et qui nécessite d'être en état de sauvegarde périodiquement, d'où l'exigence d'un stockage volumineux et une migration adéquate telle que la migration avec stockage. D'autres applications telles que Google Maps requièrent des ressources de calcul et de stockage très performantes afin de trouver, calculer et afficher tous les chemins possibles à l'utilisateur. Ces dernières tolèrent généralement un temps d'indisponibilité dû à une migration à froid.

Le besoin en termes de puissance de calcul est en croissance continue. Les entreprises trouvent des difficultés techniques et financières afin d'établir une infrastructure physique capable de gérer convenablement leurs tâches. Lorsqu'il s'agit d'un environnement intelligent, les équipements locaux font souvent preuve de capacité de calcul et stockage limitée (Botta et al., 2014). Par ailleurs, plusieurs géants tels que Google, Amazone, IBM ont investi dans la

mise en place d'un environnement puissant, divers, connecté et virtualisé capable de répondre aux différentes exigences connu sous l'infonuagique (Sadiku, Musa et Momoh, 2014).

Le nuage informatique a récemment émergé comme étant un nouveau paradigme de l'informatique omniprésente (Mell et Grance, 2011), l'ultime avantage derrière l'utilisation de cette technologie est la grande puissance de calcul et de stockage offerte ainsi que la capacité de s'adapter rapidement aux changements. Il s'agit donc d'un moyen d'héberger des services dans un environnement puissant et élargir conséquemment leur portée, surtout ceux qui sont matériellement exigeants ou nécessitent un traitement bien particulier.

Ailleurs, nous avons souvent tendance à dire que les services sont hébergés au nuage informatique, concrètement ces derniers s'exécutent sur des VMs (machines virtuelles, virtual machines en anglais), utilisent des routeurs virtuels et acheminent leurs trafics à travers des ports et des commutateurs virtuels. Ces ressources virtuelles, elles-mêmes, partagent les ressources physiques. Dépendamment des services qui y sont hébergés, les besoins de ces composants virtuels, en termes de l'infrastructure physique, sont variables dans le temps (Xiao, Song et Chen, 2013), ce qui implique des nouveaux défis à l'échelle du partage des ressources physiques. En effet, un mauvais partage mène forcément à un déséquilibre impactant la performance des ressources virtuelles et physiques, qu'à leur tour impactent les services qui s'y exécutent. Le partage optimal des ressources physiques devient ainsi un besoin fondamental au bon acheminement des services fournis et au maintien de la performance des ressources virtuelles et physiques. Toutefois, il est primordial de souligner qu'un tel partage est difficile à achever à cause de la nature hétérogène et très dynamique de l'infonuagique (Boutaba, Cheng et Zhang, 2012).

1.2 Problématique

Étant donné que les services accessibles via l'infonuagique ont un caractère évolutif, et en ayant conscience qu'ils sont hébergés dans un ensemble de VMs, le besoin de ces dernières en termes de ressources physiques devient variable au cours du temps, ainsi le risque d'une

mauvaise exploitation de la plateforme physique dû à un partage non optimal s'accroit, et en conséquence, son coût opérationnel.

Plusieurs approches (Manvi et Krishna Shyam, 2014), améliorent manifestement le partage des ressources dans le nuage informatique. Néanmoins, ces dernières ne tiennent pas en compte de multiples techniques de migration. En effet, la technique de migration à chaud (live migration) qui est majoritairement déployée, expose des limites dans certaines situations, par exemple lorsque le taux de changement des pages mémoires de la VM (dirty memory pages) est supérieur au débit d'envoi (Sv et al., 2011), d'où une nécessité de prendre en considération plus d'une seule technique. En outre, ces approches ont proposé des solutions qui traitent la consommation en énergie ou l'équilibrage de charge. De plus, à notre connaissance, elles (les approches) n'ont pas considéré, simultanément, l'état courant et futur des ressources dans le nuage informatique lors de l'établissement d'un plan de relocalisation dynamique.

Ainsi, les défis que nous éprouvons consistent à déceler une solution qui s'appuie sur la relocalisation dynamique et qui intègre un module de prédiction afin d'anticiper la future utilisation des ressources physiques. Cette solution doit alternativement prendre en compte plusieurs techniques de migration ainsi que la prédiction pour définir un plan de relocalisation permettant d'optimiser le partage de ressources physiques dans le nuage informatique et d'une manière conséquente, son coût opérationnel.

.

Par ailleurs, nous abordons précisément notre problématique en nous posant la question suivante : étant donné les exigences des VMs en ressources physiques qui varient au cours du temps, comment allouer ces dernières aux serveurs d'une manière optimale au moyen de différentes techniques de migration tout en anticipant leur consommation de ressources physiques?

1.3 Questions de recherche

Afin de bien cerner notre projet de recherche, nous abordons certaines questions de recherches :

RQ1 : L'hébergement et la migration au sein de l'infonuagique engendrent des coûts opérationnels. C'est à travers ces derniers que nous parvenons à optimiser le coût opérationnel total. Toutefois, vu que la migration inclut de différentes techniques, l'évaluation de son coût peut être très défiante. Ainsi, nous posons la question suivante: comment évaluer le coût de différentes migrations d'une manière globale afin de les utiliser efficacement dans l'optimisation du coût opérationnel?

RQ2: Les VMs ont des besoins qui varient au cours du temps, ceci implique une charge de travail variable. Ainsi, les ressources physiques courent le risque d'être surutilisées à plusieurs reprises. La prédiction est un moyen à travers lequel nous évitons cela. En conséquence, la question est: comment partager d'une manière efficace des ressources physiques en utilisant la prédiction lors du processus de relocalisation?

RQ3: L'utilisation des modèles de prédiction et de relocalisation des VMs est manifestement importante dans l'optimisation du partage des ressources physiques et donc le coût opérationnel. En conséquence, nous posons la question suivante: comment intégrer les nouveaux modèles de relocalisation et de prédiction de l'utilisation des ressources physiques dans la plateforme de gestion de nuage existante?

1.4 Objectifs

L'objectif principal de la présente recherche consiste à parvenir à un partage optimal des ressources physiques en ayant recours aux modèles statistiques pour la prédiction de la future utilisation des ressources ainsi qu'aux différentes techniques de migration.

Cet objectif peut se traduire en trois sous objectifs:

O1: L’intégration d'un modèle de prédiction s'appuyant sur des modèles statistiques ce qui permettra de renseigner la future utilisation des ressources physiques parmi les différentes VMs.

O2: L’intégration de plusieurs techniques de migration dans un même modèle d’optimisation de partage des ressources physiques, et éviter en conséquence la mauvaise utilisation de l'infrastructure physique.

O3: L'intégration des modèles précédents pour implémenter un modèle complet de partage qui est capable de générer des plans de relocalisation assurant un partage efficace.

Afin d’atteindre cet objectif, nous menons dans un premier temps une revue de littérature des méthodes de partage supportées par le nuage informatique. Notre contribution est un modèle d’optimisation qui intègre la prédiction s’appuyant sur les modèles statistiques. À travers cette dernière il devient possible de prévoir la variation de la charge de travail, et ainsi ajuster les ressources d'une manière efficace.

La deuxième contribution consiste à intégrer trois techniques de migration dans le modèle d’optimisation de cette manière nous pouvons assurer une optimisation globale du partage de l’infrastructure physique. Notre modèle mène à la satisfaction des besoins de VMs lorsqu'elles ont besoin de plus de ressources physiques, il permet également de mieux exploiter la structure physique de l'infonuagique. Nous résolvons le modèle proposé et l’implémentons dans un module logiciel pour prendre des décisions de relocalisation de ressources. Nous implémentons et intégrons par la suite le tout en une seule plateforme. Postérieurement des expérimentations seront menées dans un environnement expérimental afin de valider ce modèle au complet.

1.5 Hypothèse

L'hypothèse de ce projet de recherche présume qu'en construisant un modèle qui prédit la charge de travail et génère des plans de relocalisation basés sur de multiples techniques de migration le coût de partage de l'infrastructure physique sera réduit au minimum, tandis que les VMs seront satisfaites, d'où un partage optimal.

1.6 Plan du mémoire

Le présent mémoire s'articule autour de cinq chapitres :

Le premier chapitre est celui de l'introduction, il inclut la mise en contexte, la problématique et les objectifs, permettant ainsi de bien cerner notre projet de recherche et renseigner sur ce que nous comptons réaliser.

Le deuxième chapitre est une revue de littérature. À travers ce dernier, nous introduisons les notions de bases nécessaires dans la bonne compréhension de notre projet de recherches et son acheminement. De plus, nous présentons un ensemble de travaux connexes au nôtre, et nous discutons leurs limites. Ceci soulignera davantage la contribution de ce travail de recherche.

Le troisième chapitre détaille notre méthodologie, et ce, en soulignant le modèle proposé et décrit l'architecture du système et ses différents modules. En particulier, les modules associés à la prédiction de l'utilisation des ressources physiques par les VMs, le noyau et le module de prise de décision à travers lequel nous pouvons achever un plan de relocalisation dynamique ayant un coût opérationnel optimal. Nous introduisons dans ce chapitre un algorithme associé au module de prise de décision.

Le quatrième chapitre focalise sur le côté expérimental de notre projet. Il s'agit d'un chapitre qui inclut les résultats de nos expérimentations à travers lesquelles nous pouvons valider notre hypothèse.

Le cinquième et dernier chapitre est un bilan et une synthèse de notre contribution. Nous présentons, similairement, les perspectives à considérer pour un futur travail.

CHAPITRE 2

REVUE DE LITTÉRATURE

2.1 Introduction

Ce chapitre souligne les différents éléments et travaux de recherche nécessaires à l'acquisition d'une connaissance solide du domaine de l'infonuagique généralement et le partage de ressources physiques particulièrement.

En premier, nous présentons les architectures traditionnelles et virtualisées des centres de données, ensuite nous étudions les plateformes infonuagiques à savoir Openstack. Postérieurement, différentes techniques de partage des ressources en infonuagique sont définies. Nous en finissons avec une section dédiée aux travaux connexes utiles dans le bon acheminement de notre projet de recherche.

2.2 Plateforme physique

Une plateforme physique inclut un ensemble d'équipements à travers lesquels une entreprise offre ses propres services. Il s'agit d'une infrastructure très importante, sa bonne gestion résulte en des services ayant une meilleure performance. Toutefois, la concurrence entre ces derniers concernant les ressources physiques mène fréquemment à une insatisfaction autant du client que le fournisseur (Zhang et al., 2013). De ce fait, la progression sur le plan matériel et logiciel a contribué largement dans la mise en œuvre des techniques permettant d'une part de mieux partager ces ressources et d'autre part de combler aux besoins des services qui s'y exécutent. Dans cette section, nous examinons en détail les architectures traditionnelles et virtualisées.

2.2.1 Architecture traditionnelle

Les plateformes traditionnelles hébergent les applications dans des serveurs physiques n'ayant aucune couche de virtualisation. En effet l'application s'exécute au niveau du système d'exploitation et utilise les ressources qui lui sont disponibles, par ailleurs, la majorité des serveurs d'un centre de données opèrent pour des applications liées à des domaines spécifiques. Ceci a principalement conduit à la non-élasticité et l'inefficacité de l'utilisation des ressources physiques. Mis à part les contraintes de sauvegarde qui imposent le déploiement d'un serveur entier ou plus pour cette affaire, ces plateformes ne sont pas en mesure d'exécuter des techniques efficaces destinées à la migration des applications et services. En conséquence, des troubles tels que le gaspillage de ressources et d'énergie s'affichent. Le problème de partage des ressources a longuement motivé les chercheurs à creuser plus pour révéler des techniques fiables améliorant d'une manière appropriée le partage de l'infrastructure physique, et ce, dans l'objectif de l'exploiter convenablement et satisfaire simultanément les services offerts.

2.2.2 Architecture virtualisée

Contrairement aux plateformes traditionnelles, cette architecture s'appuie sur la virtualisation pour achever un partage convenable du matériel sous-jacent. Il devient ainsi possible de fournir plusieurs services ayant des exigences matérielles et logicielles différentes depuis le même serveur tout en assurant une isolation propice et un partage juste.

2.2.2.1 Virtualisation des hôtes

La virtualisation consiste à abstraire un système d'exploitation invité du matériel sous-jacent (Daniels, 2009). Un composant appelé hyperviseur est responsable de l'émulation du matériel et constitue une couche intermédiaire se trouvant entre le système natif et invité. À l'exception de certains cas, le système invité n'a pas généralement conscience de la couche de virtualisation et opère comme un système natif.

D’un autre côté, l’hyperviseur est un composant logiciel qui permet de créer et gérer des VMs. Il présente ainsi la couche de virtualisation responsable à l'émulation du matériel et essentielle au partage d'une même ressource. Il existe deux types d'hyperviseurs, le type 1 qui s'exécute directement sur l'infrastructure physique garantissant aux systèmes invités une performance semblable à celle d'un système natif (Guérineau, 2008). Le type 2 consiste à un composant logiciel s'installant et s'exécutant dans le système natif. Selon Hugo dans (Guérineau, 2008), les systèmes invités doivent passer par deux couches logicielles pour atteindre le matériel ce qui réduit la performance. Toutefois ce type d'hyperviseurs présente l'avantage d'une installation et configuration simple. Les hyperviseurs les plus connus dans l’industrie ainsi que la recherche sont QEMU/KVM (Kernel-based Virtual Machine), Xen, Hyper-V, VMware ESX.

Il existe différentes techniques de virtualisation qui se diffère essentiellement par leur niveau d'abstraction.

- L'isolation établit une séparation forte entre différents contextes logiciels ou plus précisément des espaces utilisateurs, cependant cette méthode n’assure pas une isolation complète (Besson, 2012);
- La para-virtualisation est une technique qui exige des interfaces spéciales, ces dernières sont intégrées dans les systèmes invités en tant que pilotes ou de modifications liées au noyau. En effet, selon Maxime B. dans (Besson, 2012), il s'agit d'un compromis entre un niveau d'abstraction élevé et un niveau de performance satisfaisant;
- La virtualisation complète, comme mentionnée dans (Besson, 2012), consiste à un hyperviseur qui intercepte d'une manière transparente tous les appels issus des systèmes invités demandant des ressources matérielles. Cette technique supporte les systèmes invités non modifiés;
- Le partitionnement matériel permet de séparer les ressources matérielles au niveau de la carte mère de la machine (Besson, 2012). Nous pouvons ainsi avoir plusieurs systèmes sur la même infrastructure physique.

2.2.2.2 Virtualisation des réseaux

La virtualisation ne se limite aucunement aux serveurs, de nos jours, elle fait ses preuves au niveau des réseaux. Selon Alexandre dans (Arrivé-Cornec, 2014), la virtualisation dans ce cas élimine l'adhérence entre les réseaux logiques et physiques, un concept semblable au cas d'un serveur et un système invité. Généralement, lorsque nous évoquons le sujet des réseaux virtuels, des protocoles participants à la virtualisation sont mentionnés tels que le VLAN (virtual LAN) et VXLAN (Virtual Extensible LAN). VMware de sa part propose un hyperviseur qui virtualise les différentes couches réseau, la totalité des règles de gestion lui est ainsi confiée (Arrivé-Cornec, 2014). D'une autre part, une technologie libre appelée OVS (open vswitch) participe considérablement à la virtualisation des réseaux. Ce composant logiciel permet de rediriger des flux entre les hôtes et se comporte comme un commutateur physique. Par ailleurs, il utilise des ponts (bridges) connectés aux interfaces des VMs. Le nuage informatique utilise les réseaux virtuels pour permettre aux différents clients de créer et gérer leurs propres réseaux de manière isolée, ainsi nous obtenons des réseaux privés et virtuels à la fois.

2.3 Plateformes infonuagiques

Une plateforme infonuagique constitue un ensemble de ressources physiques et virtuelles, interconnectées et fonctionnant en toute collaboration. Le nuage peut offrir des services pertinents grâce à la puissance de calcul et l'exploitation adéquate de l'infrastructure sous-jacente. Le nuage est considéré comme un modèle informatique dans lequel les ressources sont fournies en tant que services généraux et peuvent être allouées et libérées sur demande (Abbasov, 2014). Parmi les caractéristiques du nuage informatique:

- L'élasticité: les ressources sont allouées et libérées d'une manière adaptée aux besoins de l'utilisateur ce qui évite de longues procédures traditionnellement requises;
- L'accessibilité: tout service offert à partir du nuage est accessible de partout tant que l'utilisateur dispose d'une connectivité internet;

- Le paiement à l'usage: des outils de mesure et de collecte des données sont mis en œuvre afin de calculer le montant qui correspond à l'utilisation des ressources.

En outre, cette technologie s'affiche avec d'autres caractéristiques telles que le coût faible d'exploitation des ressources, la grande évolutivité et la minimisation des dépenses d'entretien ce qui la rende encore plus intéressante en particulier à l'échelle industrielle. Des géants tels qu'Amazone, Google, IBM et Microsoft se sont investis dans cette technologie et disposent de leurs propres plateformes (Armbrust et al., 2010).

2.3.1 Applications hébergées

Contraintes par les limites des ressources physiques locales, de nombreuses applications sont hébergées dans l'infonuagique afin de répondre à leurs besoins. Quand nous évoquons le sujet de l'infrastructure physique et sa puissance de calcul, la performance des VMs et conséquemment celle des applications qui y résident est au rendez-vous. La solution vers l'amélioration de la performance consiste donc à migrer vers l'infonuagique, cela permettra également d'éviter le coût élevé du matériel.

Il existe plusieurs types d'applications qui sont aujourd'hui accessibles depuis l'infonuagique, le commerce mobile en est un bon exemple, ce dernier doit faire face à de multiples défis essentiellement relatifs à la sécurité, la faible bande passante et la complexité élevée. Ces défis ont été adressés grâce à l'infonuagique (Dinh et al., 2013). D'une manière similaire, les applications déployées dans les environnements intelligents et ayant le concept d'apprentissage automatique sont exigeantes en terme de ressources de calculs et de stockages. Les applications servant le domaine médical et celles des jeux vidéo ont, de même, besoin de ressources puissantes. Il est important de mentionner que les applications hébergées dans le nuage informatique ont une consommation variable dans le temps, autrement, le nuage est la meilleure solution pour leur accorder plus ou moins de ressources dépendamment des besoins (Lorido-Botran, Miguel-Alonso et Lozano, 2014).

2.3.2 Openstack

Openstack est une plateforme infonuagique libre et ouverte (open-source) mise en œuvre en tant que IaaS (infrastructure as a service). Il est composé de plusieurs services interconnectés et collaboratifs ayant pour objectif le contrôle efficace de l'infrastructure physique sous-jacente, incluant les serveurs, les espaces de stockages et les équipements réseau. Les utilisateurs peuvent gérer cette plateforme de différentes manières, soit à travers des interfaces web, la ligne de commande ou un API (application programming interface). La solution Openstack est sous un développement actif depuis 2010 et une nouvelle version voit le jour chaque six mois. Selon (Sefraoui, Aissaoui et Eleuldj, 2012), c'est une plateforme qui a largement du potentiel grâce à son architecture, sa communauté et le support de ses partenaires. Développé à l'origine par NASA, Openstack est dédié aux infrastructures massives, mais peut fonctionner convenablement avec une infrastructure limitée.

2.3.2.1 Services

Les services forment la base à travers laquelle Openstack opère, ils fournissent l'ensemble des fonctionnalités, telles que la création des VMs, leur contrôle, la gestion du stockage et encore. Ces services sont généralement distribués et utilisent le réseau pour s'échanger des informations, toutefois il est possible de les avoir tous dans une même machine dépendamment de l'architecture déployée. Étant libres et ouverts (open-source), plusieurs services étaient des projets individuels, ensuite ont été rassemblés sous le même toit appelé Openstack surmontant tout problème de compatibilité, d'où leur performance s'est vue améliorée. La figure 2.1 décrit les différents services existants à l’heure actuelle dans Openstack, et les fonctionnalités offertes.

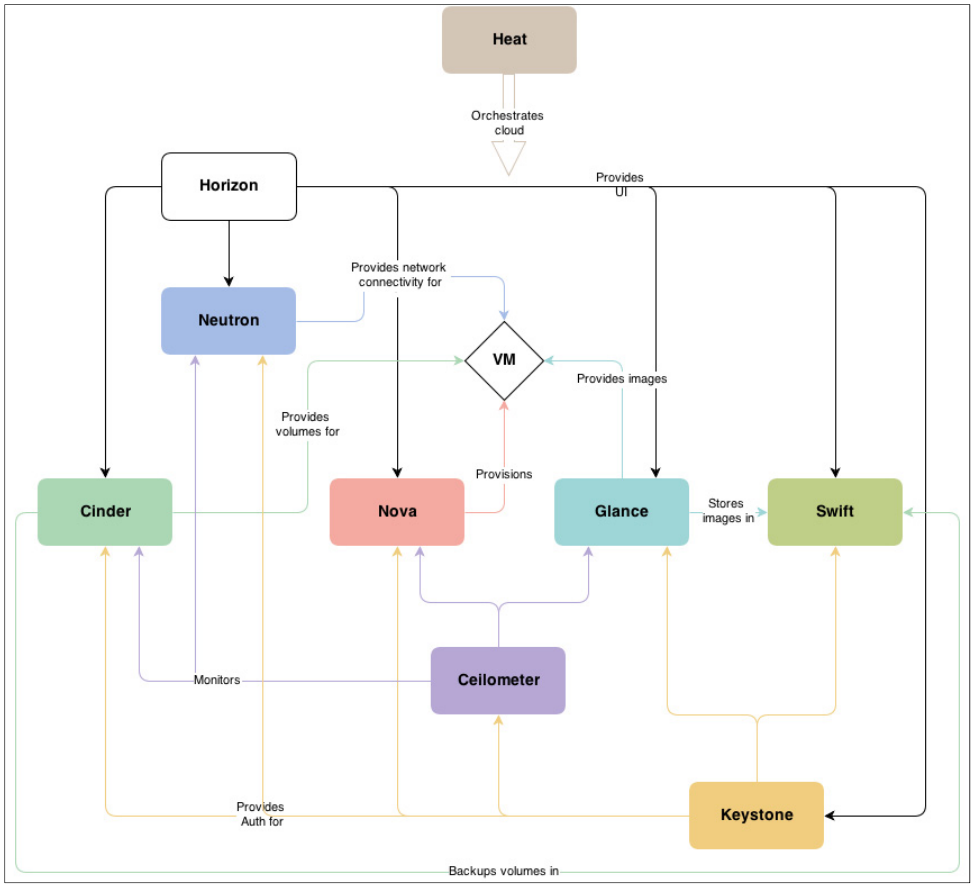

Figure 2.1 Services Openstack

Les services Nova :

Nova est un groupe de services indispensables dans Openstack, il permet la mise en œuvre et la gestion des VMs sur un nombre d'hôtes hébergeant le service nova-compute. Il est aussi à l'origine de l'évolutive dont Openstack est caractérisé. Ce composant est analogue à Amazon EC2 (elastic compute cloud), et est géré semblablement. Selon (Linusnova, 2013), nova est le principal service d'Openstack et le plus complexe à la fois, il est formé par un ensemble de processus qui communiquent à travers une file et s'échangent des informations pour assurer la gestion des divers nœuds de calculs.

Les services de réseaux :

Openstack offre la possibilité de créer et gérer des infrastructures de réseaux virtuels incluant les sous-réseaux, commutateurs et routeurs. Cela permet aux différents utilisateurs de créer leurs propres réseaux virtuels d'une manière isolée sur une même infrastructure physique. De part et d'autre, Openstack suggère deux services de réseautage, le premier est un démon Nova appelé nova-network, et le deuxième est un service à part appelé Neutron, ce dernier se révèle d'être beaucoup plus complexe et performant à la fois que nova-network (Kumar et al., 2014).

Les services de collecte des données :

La collection des données est une fonctionnalité importante dans les plateformes infonuagiques, tel est le cas pour Openstack. Ce dernier présente un service de télémétrie appelé Ceilometer, capable de collecter les données d'une manière efficace afin qu'elles soient utilisées pour des fins de facturation ou d'analyses (Kumar et al., 2014). La collection peut être stockée ensuite sous forme d'échantillons ou d'événements dans une base de données.

La télémétrie est un service composé d'un ensemble d'agents, nous en citons, l'agent de notification qui est responsable de la consommation des notifications envoyées par les différents services d'Openstack (Kumar et al., 2014). Ces notifications sont transformées par l'agent en des événements et échantillons de mesure et publiées en conséquence. Il est possible de personnaliser le service de telle manière à collecter des métriques spécifiques. D'autre part, ce service doit avoir une image complexe de l'infrastructure. Cet objectif nécessite des informations supplémentaires que celles fournies par les événements et les notifications publiées par chaque service, d'où Ceilometer déploie une autre technique pour collecter ces données en interrogeant l'infrastructure, incluant les API des différents services Openstack et les hyperviseurs, cette technique est appelée Polling, il existe trois agents qui la supportent, l'agent de calcul, central et IPMI (intelligent platform management interface).

Les services de stockage :

Le stockage sous Openstack figure parmi les fonctionnalités principales, les services qui y sont dédiés traitent toute demande de gestion des espaces de stockage. Aujourd'hui, il en existe deux types, stockage bloc et objet, le choix se fait généralement suivant le besoin, cependant, dans les déploiements courants, les deux types sont fréquemment utilisés, d'où la raison pour laquelle Openstack inclut deux services appelés « Cinder » détaillé par (Rosado et Bernardino, 2014) et « Swift » détaillé par (Pepple, 2011) qui correspondent au stockage bloc et objet.

2.3.2.2 Architecture

Comme il a été déjà mentionné, Openstack est une plateforme infonuagique qui s'appuie sur un ensemble de services ayant pour objectif la gestion des ressources physiques et virtuelles. L'architecture de cette plateforme peut être composée d'un seul nœud ou plusieurs, ses services sont ainsi installés relativement à sa distribution. Généralement l'utilisation d'un seul nœud revient à des objectifs de tests, la plupart des infonuagiques basées sur Openstack sont distribuées sur de nombreux nœuds. Nous expliquons, dans ce qui suit chacune de ces architectures.

Une architecture à plusieurs nœuds telle indiquée dans la figure 2.2, consiste à distribuer les fonctions sur plusieurs serveurs, nous aurons donc différents types de nœuds dont nous détaillons dans ce qui suit (Beloglazov et al., 2012):

- Nœud de contrôle: cet hôte est capable de gérer les ressources virtuelles et physiques de l'infonuagique. Il s'agit d'un composant primordial et admet des services importants tels que nova (nova API, nova scheduler...etc.), serveur Neutron qui permet le contrôle du nœud de réseautage, ainsi que des services de stockage...etc;
- Nœud de réseautage: à travers un nœud totalement dédié, il devient possible de gérer des réseaux virtuels ayant des topologies complexes. De plus, toute communication à

l'externe ou entre des sous-réseaux est gérée par ce composant. Il inclut des services tels qu'un serveur DHCP (dynamic host configuration protocol), agent L2 et agent L3, ce qui rend faciles la création et la manipulation de différents routeurs, commutateurs et réseaux virtuels;

- Nœud de calcul: cet élément est essentiel pour exécuter des VMs, pour ce faire, deux services sont obligatoires, la nova-compute ainsi qu'un agent Neutron qui permet de connecter la VM aux réseaux virtuel et externe via le nœud de réseautage;
- Nœud de stockage: il est possible d'avoir des nœuds de stockage à travers lesquelles des espaces de stockages sont exploités. Ce nœud comporte un ensemble de services, par exemple le stockage bloc cinder-volume.

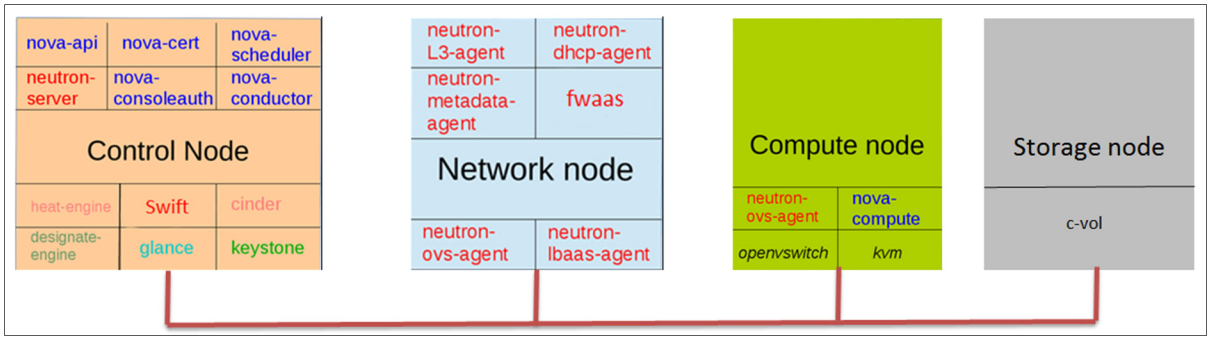

Figure 2.2 Architecture multi-nœuds d'Openstack
Tirée de (Trezeciak, 2016)

De toute évidence, nous pouvons avoir des nœuds dédiés à d'autres fonctions telles que la collecte des données, l'équilibrage de charge…etc.

Dans le cas d'une architecture à un nœud, tous les services Openstack sont installés sur un seul serveur, ce dernier se comporte ainsi comme étant un nœud de contrôle, de calcul, de réseautage et même de stockage. Les VMs sont créées dans ce serveur ainsi que tous les réseaux virtuels, en conséquence, la performance se dégrade à cause de la grande charge appliquée sur ce nœud unique. Cette architecture n'est en aucun cas pratique, toutefois elle peut être utile pour des fins de tests.

2.3.2.3 Migration des VMs

En tant que technologie infonuagique, Openstack est apparu pour contrôler et bien partager de vastes réservoirs de ressources physiques. Ce dernier suggère la migration comme un outil assurant un partage idéal de l'infrastructure matérielle sous-jacente. En effet, il est possible de mener la migration d'une VM en s'appuyant sur différentes techniques (Kashyap, Chaudhary et Jat, 2014). La figure 2.3 illustre ces techniques.

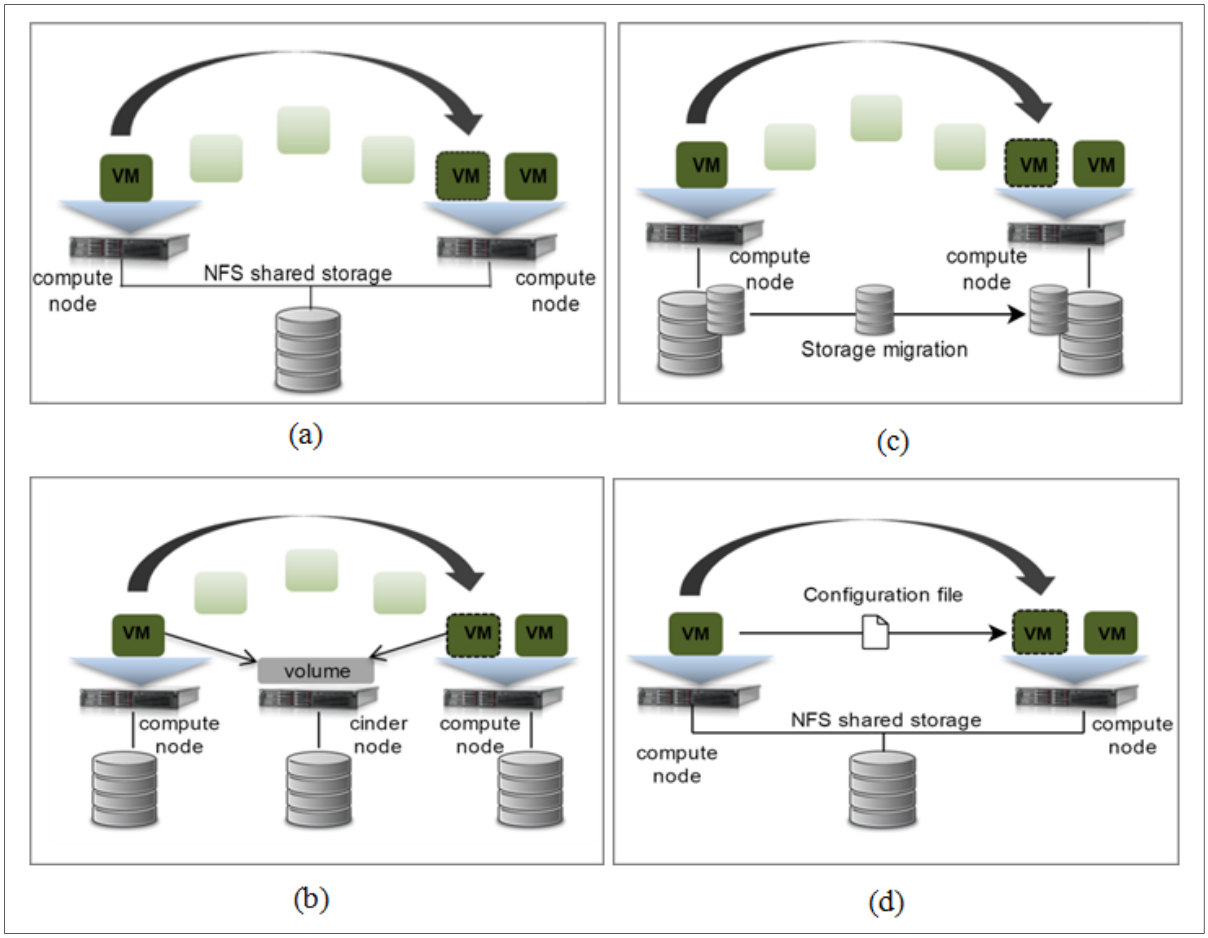

Figure 2.3 les techniques de migration, (a) LM, (b) VBLM, (c) BLM, et (d) CM

Lors de la migration à chaud (live migration), seuls la mémoire et l'état de l'unité centrale sont transférés, cette technique nécessite la configuration d'un stockage partagé entre les

hôtes sources et cibles. La mémoire est transférée en pré-copiant ses pages vers la destination pendant que les applications continuent à fonctionner normalement (figure 2.3(a)). Cependant, le taux de pages modifiées (dirty-pages) devrait impérativement demeurer sous le taux de transfert disponible (Svärd et al., 2011). Par défaut, Openstack définit un seuil spécifique au-dessus duquel la migration à chaud sera interrompue.

La migration à chaud supportée par les volumes (volume-backed live migration) se comporte similairement à la migration à chaud ordinaire, sauf qu'elle ne nécessite aucune configuration de stockage partagé (figure 2.3(b)). En effet, le service de stockage en bloc (cinder) crée automatiquement un seul back-end partagé entre chaque hôte participant.

D'autre part, la migration à chaud avec stockage (bloc live migration) supporte la migration du stockage, de la mémoire et l'état de l'unité centrale, par conséquent, le temps de migration augmente relativement à la quantité utilisée de stockage (figure 2.3(c)). Cette technique n'implique pas une configuration de stockage partagé.

Une autre technique appelée migration à froid (cold migration) est supportée par Openstack. Dans cette technique aucune configuration supplémentaire n'est requise. En effet, un fichier contenant les spécifications de la VM est envoyé à la cible, là où elle sera rétablie. Cette machine est suspendue avant le processus, ensuite reprise à la destination (figure 2.3(d)). Il est à noter que la migration à froid sous Openstack, implique l'envoi du stockage même s'il est partagé.

Toute technique de migration implique un temps de transfert, ce dernier est calculé généralement à travers le temps de pré-copie et d'arrêt. Le temps de pré-copie représente une phase dans laquelle une partie importante de mémoire/stockage (dépendamment de la technique de migration) est envoyée à la destination. Le temps d'arrêt (down time) est la phase dans laquelle la VM est indisponible. Le travail de (Pang, 2010) a proposé une manière avec laquelle nous pouvons estimer ces derniers. D'autre part, la recherche de (Moghaddam

et Cheriet, 2010) nous a permis de comprendre en détail les techniques de compression et de pré-copie utilisées pour améliorer la migration à chaud, en particulier le temps d'arrêt.

2.4 Partage des ressources en infonuagique

Selon (Moreno-Vozmediano, Montero et Llorente, 2012), un composant clé d'un centre de données moderne est la gestion de l'infrastructure physique et virtuelle par la plateforme infonuagique. Le partage optimal des ressources physiques offre impérativement un meilleur mode opérationnel du centre de données. L'optimisation du partage des ressources physiques est motivée par différents critères tels que la consolidation des serveurs, l'équilibrage de charge, ou l'équilibrage thermique. Les objectifs qui leur correspondent se traduisent par la réduction du nombre de serveurs et donc minimiser l'énergie consommée, l'équilibrage de charge et conséquemment éviter la dégradation de la performance, ou encore l'équilibrage de la température entre les serveurs et ainsi éviter la surchauffe tout en réduisant le besoin en refroidissement (Moreno-Vozmediano, Montero et Llorente, 2012). D'autres critères peuvent s'imposer, par exemple la réduction des coûts opérationnels d'hébergement et de relocalisation, l'objectif étant la minimisation du coût total de partage de l'infrastructure physique.

Généralement, s'appuyer seulement sur la virtualisation améliore le partage des ressources physiques mais ne l'optimise pas (voir inconvénients de la virtualisation dans (Semnanian et al., 2011)). En effet, les services qui s'exécutent sur les ressources virtuelles, précisément sur les VMs, ont une nature dynamique, d'une autre manière, leurs besoins sont changeables au cours du temps, d'où ceux des VMs. Il peut s'avérer ainsi intéressant de leur attribuer plus ou moins de ressources à travers la relocalisation dynamique ou encore en anticipant leurs besoins. Achever un partage optimal implique des techniques fondamentales que nous explorons dans ce qui suit.

2.4.1 Relocalisation dynamique

La relocalisation dynamique s'intéresse au déplacement d'un composant virtuel d'une source vers une destination, pour ça, elle s'appuie fortement sur le mécanisme de migration offert par la plupart des plateformes infonuagiques. Son utilité se concrétise par la bonne utilisation des ressources physiques. Plus précisément, la migration des VMs évite d'une part la surutilisation et sous-utilisation des serveurs physiques, et d'autre part l'incapacité d'accorder plus de ressources physiques à une VM suite à leur insuffisance. Nous avons déjà souligné les différentes techniques de migration sous Openstack, chacune dispose de ses propres avantages et limites, de ce fait, il est encore plus intéressant de mettre en place un système décisionnel permettant la sélection automatique de la meilleure technique de migration selon l'état actuel de la plateforme. La ressource la plus utilisée par la relocalisation dynamique est la bande passante, en conséquence, la vigilance lors de son utilisation est primordiale afin d'éviter des problèmes de latence causés par la congestion. Dans la section des travaux connexes, nous introduisons des travaux qui ont déployé la relocalisation dynamique afin d'optimiser l'exploitation des ressources physique.

2.4.2 Prédiction de la charge de travail

Nous avons souligné dans les sous-sections précédentes que les relocalisations dynamiques sont d'une grande utilité lors de l'optimisation de partage de ressources de l'infrastructure physique. Toutefois, il peut s'avérer problématique si nous menons ces opérations en ignorant l'état futur des ressources physiques et virtuelles. Dans le cas de la relocalisation dynamique, la prédiction joue un rôle important, et ce, en prévoyant d'éventuelles surutilisations des ressources physiques, en conséquence nous évitons de migrer les VMs vers des destinations qui seront probablement surchargées.

2.4.2.1 Les séries temporelles

Les modèles de prédictions diffèrent selon le besoin et le type de données, nous visons en particulier les modèles statistiques appliqués sur des données capturées sur la base des séries

temporelles. Les séries temporelles sont un ensemble de valeurs numérique qui mettent en évidence le comportement d'une entité particulière au cours du temps, généralement ces valeurs sont enregistrées dans des intervalles bien réguliers. La prédiction appliquée sur ces séries se traduit par un mécanisme dans lequel les observations du passé sont analysées afin d'élaborer un modèle qui décrit la relation sous-jacente, ce modèle est ensuite déployé dans le but d'extrapoler des séries temporelles dans le futur. Par exemple, la collection du taux d'utilisation de mémoire d'un serveur chaque minute, et l'application d'un modèle de prédiction statistique sur ces données peut être très pertinent pour prédire la prochaine valeur et agir en conséquence.

2.4.2.2 Les modèles statistiques

Il existe plusieurs modèles statistiques utilisés généralement pour prédire le futur comportement d'une entité, ils sont déployés dans plusieurs domaines (financier, météorologique, informatique…etc.) et ont fait preuve d'une bonne précision. Nous expliquons ci-dessous quelques-uns, tel décrit par (Chatfield, 2016):

- Le modèle autorégressif (AR pour autoregressive) est un modèle dans lequel les valeurs individuelles des séries temporelles peuvent être décrites par des modèles linéaires basés sur des observations précédentes;
- Dans le modèle de moyenne mobile (MA pour moving average), les valeurs des séries temporelles peuvent être exprimées en fonction des erreurs des estimations précédentes. De ce fait, les estimations passées ou les erreurs de prévision sont prises en compte lors de l'estimation de la prochaine valeur de la série;
- En combinant les modèles AR et MA, le modèle autorégressif à moyenne mobile (ARMA pour autoregressive and moving average) est obtenu. Ce modèle prédit sur la base des anciennes observations et erreurs d'estimations, ce qui lui rajoute de la pertinence, toutefois, un tel modèle ne peut s'appliquer sur des données non stationnaires;
-

- ARIMA (autoregressive integrated moving average) rajoute l'étape de différenciation au modèle ARMA (Chatfield, 2016). Une étape qui permet de transformer les données en données stationnaires.

Les séries temporelles sont souvent non-stationnaires, cela veut dire qu'elles incluent de la saisonnalité et de la tendance (Jain, 2016). La différenciation permet de s'en débarrasser le temps d'appliquer ces modèles statistiques. Cette technique est requise car la régression linéaire assume que les observations sont toutes indépendantes (Jain, 2016).

2.4.2.3 Étapes de prédiction : cas d'ARIMA

Afin d'obtenir une meilleure prédiction, il est impératif de suivre certaines étapes. Nous prenons l'exemple du modèle ARIMA comme décrit dans la figure 2.4, nous détaillons dans ce qui suit ces étapes (Abu, 2016):

- Étape 1 : visualiser les séries temporelles afin d'analyser à l'œil la variation des données, cette étape n'est pas obligatoire, mais peut se révéler importante dans certains cas;
- Étape 2 : stationner les séries afin d'éliminer la saisonnalité et la tendance;
- Étape 3 : dresser la fonction d'autocorrélation (ACF pour autocorrelation function) et la fonction d'autocorrélation partielle (PACF pour partial autocorrelation function) et ce, afin de déterminer les paramètres de configuration convenables dans le modèle ARIMA;
- Étape 4 : construire le modèle pour pouvoir prédire des séries temporelles futures;
- Étape 5 : Prédire en appelant le modèle construit dans l'étape précédente.

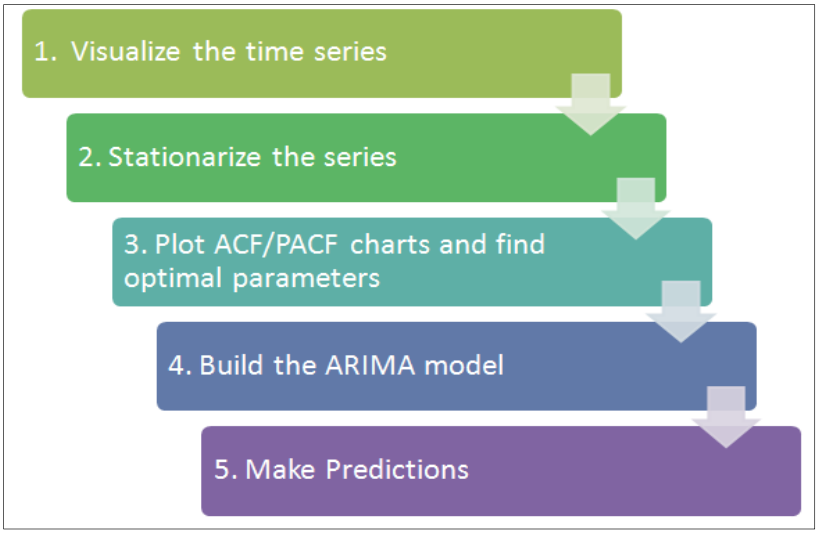

Figure 2.4 Les étapes de prédiction: cas d'ARIMA
Tirée de (Abu, 2016)

2.5 Travaux connexes

Plusieurs chercheurs se sont intéressés au problème de partage de l'infrastructure physique dans l'infonuagique. Les solutions proposées pour l'optimisation du partage des ressources se focalisent sur le coût opérationnel, la consolidation des serveurs, l'équilibrage de charges et l'amélioration de la qualité des services. Nous étudions ainsi les modèles qu'ils ont adoptés afin de s'en inspirer dans notre travail, ensuite nous présentons leurs limites et ce que nous proposons pour les surmonter.

2.5.1 Solutions basées sur la relocalisation

Le travail de (Beloglazov, Abawajy et Buyya, 2012) a suggéré la relocalisation dynamique, en particulier la migration à chaud, comme moyen pour assurer une meilleure consolidation des serveurs et en conséquence la minimisation de l'énergie. Ils ont proposé deux stratégies complémentaires, une qui choisit les emplacements des VMs (VM placement) et une autre qui sélectionne les machines candidates (VM selection).

La première stratégie consiste à placer chaque VM dans un hôte impliquant la moindre augmentation d'énergie suite à cette allocation. Le problème est semblable à un problème de « bin packing » avec des tailles et des prix variables. D'un autre côté, la deuxième stratégie consiste à sélectionner le minimum de VMs à déplacer (MM pour minimization of migration). Cette dernière est considérée lorsque le taux total d'utilisation de CPU (unité centrale, central processing unit en anglais) est soit supérieur à un seuil maximal ou inférieur à un seuil minimal. Soit V_j l'ensemble des VMs allouées au serveur j, $P(V_j)$ est donc l'ensemble de puissances qui correspond à V_j. La politique MM essaie de trouver R tel exprimé dans (2.1), avec T_u le seuil maximal, T_l le seuil minimal, u_j l'utilisation totale de CPU au serveur j et $u_a(v)$ la fraction de CPU allouée à la VM v.

$$R = \begin{cases} \left\{ S | S \in P(V_j), u_j - \sum_{v \in S} u_a(v) < T_u,\ |S| \to min \right\}, & if\ u_j > T_u; \\ V_j, & if\ u_j < T_l; \\ \emptyset, & otherwise \end{cases} \tag{2.1}$$

Cette politique permet de sélectionner les VMs qui contribuent le plus au dépassement du seuil maximal et ce, pour chaque serveur dont ce seuil est violé. Dans le cas contraire, toutes les VMs sont sélectionnées. Dans le cadre de ce travail, d'autres politiques de sélections sont proposées. Une des limites de cette recherche est l'utilisation d'une seule technique de migration (migration à chaud) ce qui peut être très coûteux dans certains cas, en particulier, lorsque le taux des pages mémoire modifiées est proche du taux de transfert. De plus, la future utilisation des ressources n'est pas anticipée, et donc les serveurs doivent atteindre le point de surcharge avant qu'ils soient traités.

Dans un autre travail, un modèle proposé par (He, Guo et Guo, 2011) assure une meilleure utilisation des ressources matérielles à travers la réduction du coût opérationnel. En effet, le travail a examiné deux cas, le cas des demandes de VMs volumineuses, et celui des demandes de redimensionnement, quoique, la relocalisation de ces machines en se basant sur la migration est exigée dans les deux cas. Quelques étapes ont été étudiées, la première

consiste à introduire des propriétés et les analyser au niveau d'un serveur de ressources. Typiquement, ces propriétés incluent la disponibilité de la mémoire, le type d'hyperviseur et l'objectif du nœud (hébergement, sauvegarde...etc.). La deuxième étape tente de découvrir les nœuds convenables à la relocalisation des ressources virtuelles. Quant à la troisième étape, elle consiste à sélectionner le nœud le plus approprié entre ceux qui sont ciblés dans la deuxième étape.

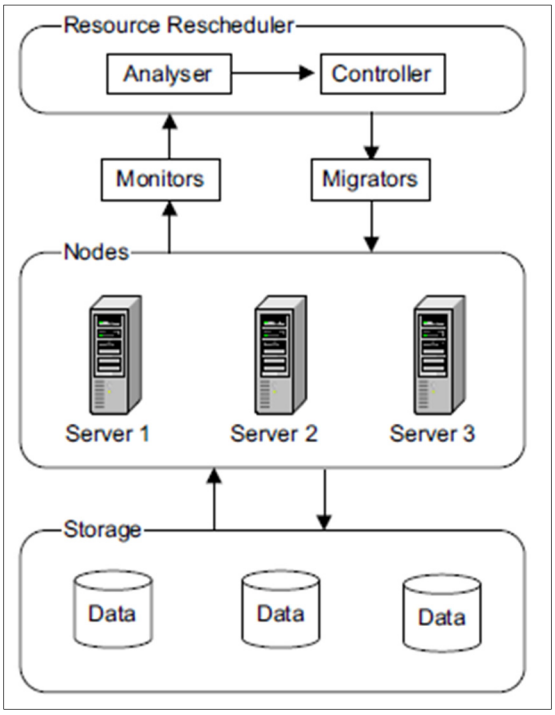

Figure 2.5 Partage des ressources
physiques - modèle 1
Tirée de (He, Guo et Guo, 2011)

Le système repose sur un ensemble de modules comme l'indique la figure 2.5, un module pour effectuer le suivi et sélectionner les données pertinentes, un analyseur pour suggérer le plan de relocalisation le plus approprié, un contrôleur pour répartir les tâches de migration selon le plan et un exécuteur qui assurera l'exécution de la migration. Ce travail ne cible pas les centres de données à grande échelle et utilise en priorité la migration à froid comme un moyen de relocalisation.

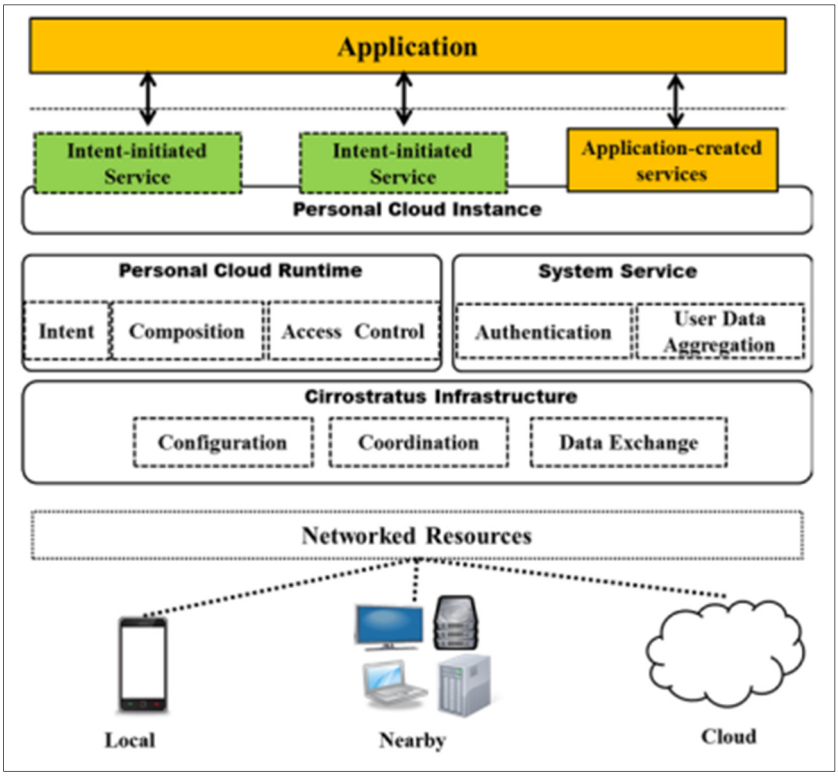

Figure 2.6 Partage des ressources physiques - modèle 2
Tirée de (Jang et al., 2014)

Le travail présenté dans (Jang et al., 2014) a conduit à la mise en œuvre d'un Framework appelé PCloud. Ce dernier permet l'amélioration de l'expérience utilisateur à travers le partage des ressources physiques, plus précisément, celles à proximité et à distance, en conséquence, un périphérique autonome peut voir ses capacités renforcées par le processus de partage. En outre, ce Framework est caractérisé par une infrastructure en couches comme l'indique la figure 2.6 où le module Cirrostratus est la couche permettant la gestion des ressources physiques à proximité et à distance. Le module runtime de sa part assigne le service à la ressource la mieux adaptée. Les deux modules sont collaboratifs afin de répondre convenablement aux requêtes utilisateurs.

De leur côté fonctionnel, 'Cirrostratus' calcule un indice de performance (PI) à travers lequel le 'runtime' décide ensuite où attribuer ces services. Cet index est une heuristique qui change

dynamiquement et utilise des données récapitulatives sur les capacités des ressources. Il est calculé en fonction de la mémoire, de la fréquence du processeur et de son utilisation actuelle. De ce fait, quand une application exige un service, le runtime interagit avec Cirrostratus et obtient le PI de chaque ressource, le service est par la suite affecté à celle ayant le plus grand PI. De plus, des temps de calcul et de réseau sont calculés régulièrement, le Framework effectue ensuite une relocalisation lorsque ces durées de service dépassent de façon substantielle et constante les valeurs précédemment observées.

Les expérimentations ont montré une meilleure performance des applications lorsque nous incluons des ressources à proximité et distantes. Néanmoins, la stratégie de relocalisation des ressources n'est pas très efficace, puisque la qualité de service doit diminuer à plusieurs reprises pour lancer une éventuelle relocalisation.

Une recherche menée par (Moses et al., 2011) représente le problème de partage de ressource à travers une approche appelée MIMe (Monitor, Identify et Migrate) comme indiqué dans la figure 2.7. Au premier, les machines sont surveillées et donc celles qui sont en manque de ressources seront identifiées. Les candidats à la migration sont déterminés sur la base d'une technique qui prend en considération la qualité de service des applications. Cette étude souligne en particulier le partage du cache, toutefois, le partage de la mémoire, le processeur et la bande passante sont essentiels pour achever un partage optimal de l'infrastructure physique.

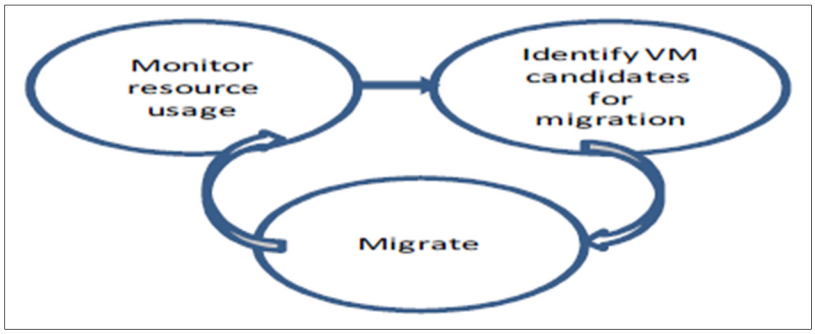

Figure 2.7 Partage des ressources physiques - modèle 3
Tirée de (Moses et al., 2011)

2.5.2 Solutions basées sur la prédiction

Le travail de (Elprince, 2013) a introduit un contrôleur de ressources autonomes qui permet d'allouer les ressources dans les conteneurs virtuels d'un centre de données. Le modèle est composé d'un ensemble de modules, nous en citons ceux de modélisation système, prédiction de ressources requises, allocation et reconfiguration de ressources (figure 2.8).

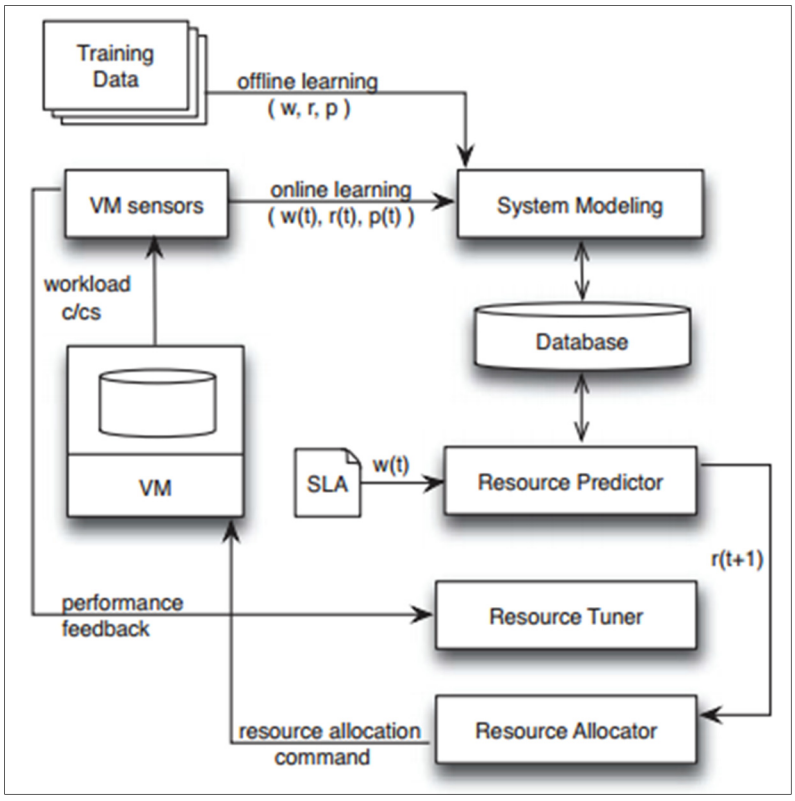

Figure 2.8 Partage des ressources virtuelles basé
sur la prédiction
Tirée de (Elprince, 2013)

La modélisation système s'appuie sur un historique de données ainsi que de nouvelles données entrées au fur et à mesure, soulignant ainsi un apprentissage hors ligne et en ligne. Ce module tente de modéliser les tâches réussies, selon l'auteur, en modélisant ces dernières, la quantité de ressources requise pour une certaine tâche future peut être prédite tout en garantissant sa réussite. La réussite est décidée suivant le respect des métriques de

performances telles que le temps de réponse et d'exécution. Des analyses statistiques sont menées afin d'explorer la corrélation entre les métriques de performance et l'utilisation de ressource, de cette manière, les ressources peuvent être assignées dépendamment de ces métriques demandées par le client. En outre, plus d'un type de technique d'apprentissage automatique est déployée, nous en citons la régression et la classification. Le prédicteur de ressources, de son côté, reçoit une requête pour héberger un type application accompagnée de métriques de performance bien définies, il effectue alors une estimation d'allocation initiale des ressources nécessaires permettant de répondre à la demande reçue. L'allocateur de ressource exécute toute commande d'allocation et le module de configuration reçoit des informations sur la performance (feedback) à travers lesquelles il décide d'attribuer plus ou moins de ressources à une application.

Les expérimentations ont montré que le système est capable de prédire convenablement les besoins en termes de ressources virtuelles tout en ayant la capacité de satisfaire les métriques de performances évaluées dans l'accord de niveau de service SLA (service level agreement), toutefois, cette recherche se focalise sur l'attribution des ressources virtuelles et leur partage par les applications sans considérer la plateforme physique.

Les auteurs (Wuhib, Stadler et Lindgren, 2012) ont implémenté une architecture de gestion de performance basée sur Openstack (figure 2.9). Dans un premier temps, le planificateur d'Openstack réalise un placement initial, ensuite, un contrôleur de placement dynamique exécute une relocalisation à travers la migration à chaud. D'autres composants tels qu'un profileur de demandes et un estimateur de ressources utilisées, permettent la prédiction des besoins de VMs en s'appuyant sur la caractérisation, les données sont par la suite envoyées aux contrôleurs. Les décisions prises concernant les migrations des VMs suivent des objectifs de gestion, soit l'équilibrage de charge et l'efficacité énergétique. L'architecture est validée par des expérimentations affirmant l'amélioration apportée au niveau de l'énergie et l'équilibrage de charge, néanmoins nous estimons que cette architecture n'est pas efficace dans les environnements assez dynamiques puisque la migration à chaud implique une utilisation intensive des ressources de réseau.

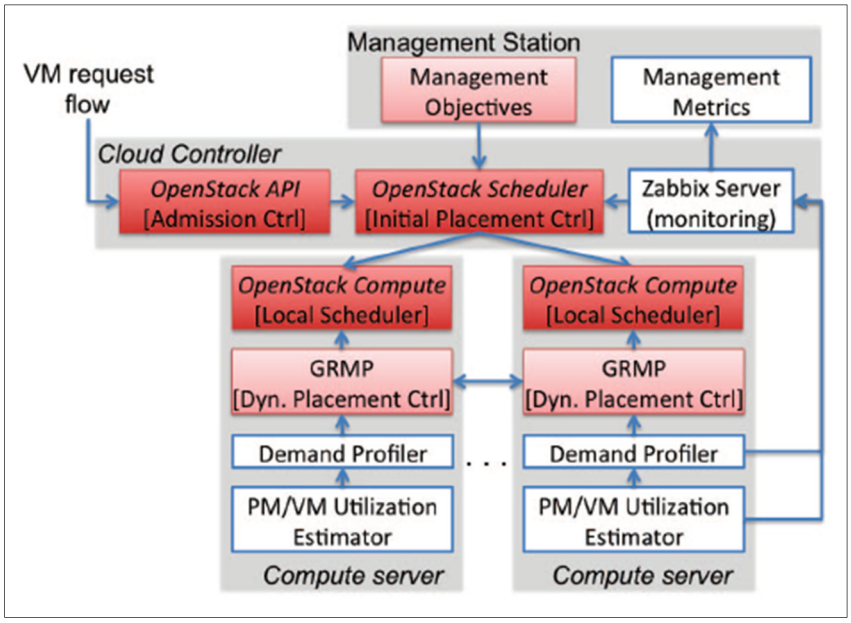

Figure 2.9 Partage des ressources physiques basé sur la prédiction
Tirée de (Wuhib, Stadler et Lindgren, 2012)

2.5.3 Limites

Les recherches que nous avions introduites suggèrent différents modèles de partage de ressources physiques et virtuelles tout en maintenant une qualité de service convenable. D'une manière générale, les limites exposées par ces travaux incluent deux points majeurs:

- Une seule technique de migration est déployée à la fois, soit la migration à chaud ou la migration à froid;
- Le coût lié à la relocalisation dynamique est souvent négligé.

Sous la lumière de ces limites et dans le cadre de notre recherche, nous proposons un modèle permettant de décider une relocalisation sur la base de l'utilisation courante et future des ressources physique. Le modèle parvient également à décider quelle technique de migration la relocalisation doit suivre, et ce, dépendamment du coût opérationnel dépendant. Ce modèle a pour objectif le partage optimal ayant un coût minimal.

2.6 Conclusion

La revue de littérature présentée dans ce chapitre met en évidence la richesse du domaine de l'infonuagique, c'est un domaine qui suscite l'attention des acteurs aussi bien industriels qu'académiques, ce qui le rend particulièrement dynamique et en évolution continue.

Au cours de ce chapitre, nous avons mis l'accent sur les architectures traditionnelles des centres de données et leurs inconvénients. À l'opposition, nous avons présenté les architectures virtualisées ainsi que leurs avantages en termes de l'exploitation des ressources matérielles. Nous avons également étudié le concept de l'infonuagique tout en spécifiant ses services et les applications qui y sont hébergées, de ce fait, le cas d'Openstack a été pris en charge. Étant donné que notre recherche se focalise sur le partage des ressources physiques en infonuagique, nous avons détaillé le partage de ressources, à savoir la relocalisation dynamique et la prédiction basée sur les modèles statistiques.

Nous avons achevé ce chapitre par une présentation des travaux connexes au nôtre, dont ceux s'appuyant principalement sur la relocalisation dynamique ainsi que la prédiction. Nous avons également analysé leurs limites vis-à-vis de et ce que nous proposons à travers notre projet de recherche.

CHAPITRE 3

MÉTHODOLOGIE

3.1 Introduction

Dans ce chapitre nous introduisons notre méthodologie qui permet de répondre à la problématique de recherche ainsi que les objectifs décrits dans le premier chapitre. À cet effet, nous présentons et détaillons les modèles proposés ainsi que l'architecture du système dans son intégralité.

3.2 Modèle de prédiction

Dans le cadre de notre méthodologie, nous proposons un modèle basé sur la probabilité, ceci permet de répondre à notre premier objectif défini dans la section 1.4. Cette dernière permet de vérifier que la future utilisation n'expose pas un risque important menant à la surcharge des ressources, d'où la dégradation des performances. D'une autre manière, nous calculons le risque de surcharger une ressource k d'un certain hôte qui est lié à une stratégie spécifique de relocalisation et nous le comparant à un certain seuil δ, s'il est dépassé, le système de décision cherche une autre stratégie. Le calcul de la probabilité s'appuie sur des données exprimant la future allocation des ressources à l'instant t_1 par les différentes VMs. Cette probabilité est exprimée à travers l'équation (3.1). Plus précisément, nous calculons la probabilité que les futurs besoins des VMs résidentes dans un hôte et celles qui doivent migrer vers lui surchargent ce dernier. Pour ce faire, nous nous basons sur une distribution normale de l'utilisation des ressources physiques. En effet, le grand nombre de données aléatoires explique notre choix de distribution normale comme étant la distribution la plus appropriée. La variable aléatoire dans notre cas est $AllocationFuture_k$.

$$\Pr_{t=t_1}\left(\frac{AllocationFuture_k}{Capacité\ totale_k} - UtilisationMax_k \geq 0\right) \leq \delta \tag{3.1}$$

Nous avons déployé ARIMA (p, d, q) en tant que modèle statistique nécessaire à la détermination de la future allocation des ressources physiques $TotalFutureAllocation_k$. En fait, à travers ce dernier, nous avons obtenu une meilleure prédiction lorsqu'il est appliqué à la charge de travail de nos VMs et ce, en comparaison avec d'autres modèles statistiques tels que l'auto-régression (AR) et la moyenne mobile (MA).

Le modèle ARIMA est appliqué sur l'historique d'utilisation des ressources physiques, ce dernier considère ainsi p dernières valeurs d'utilisation ainsi que q dernières erreurs de prédiction. Ce modèle peut être décrit en utilisant l'équation ci-dessous, où $x(t)$ est la valeur à estimer, $x(t-i)$ avec $i = 1..p$ sont les p dernières observations, $\varepsilon(t-i)$ avec $i = 1..q$ sont les q dernières erreurs d'estimation, $\propto_i$ expriment des coefficients d'autocorrélation et β_i réfèrent aux poids appliqués aux valeurs antérieures dans les séries temporelles.

$$x(t) = \sum_{i=1}^{p} \propto_i x(t-i) - \sum_{i=1}^{q} \beta_i \varepsilon(t-i) \tag{3.2}$$

Comme nous l'avons expliqué dans la revue de littérature au niveau du deuxième chapitre, le modèle est une combinaison de deux autres (auto régressif et moyenne mobile). L'historique est mis à jour périodiquement afin de garantir un modèle bien précis.

3.3 Modèle de partage des ressources physiques

Le problème du partage des ressources dans l'infonuagique se produit lorsque nous disposons d'un ensemble de composants virtuels ayant des exigences de ressources physiques variables dans le temps et partageant la même infrastructure sous-jacente.

Plus précisément, étant donné les besoins en ressources physiques des VMs variables dans le temps, nous désirons générer des plans de relocalisation qui prennent en compte l'utilisation courante et future des ressources. Les modèles statistiques doivent être déployés pour la

prédiction de la future utilisation des ressources, ainsi qu'un ensemble de techniques de migration.

Notre modèle diffère des modèles traditionnels de partage des ressources en considérant à la fois l'état courant et futur de l'utilisation des ressources physiques, ainsi qu'un ensemble de techniques de relocalisation dynamique. Ce modèle permet de répondre au deuxième objectif souligné dans la section 1.4.

La figure 3.1 montre un plan de relocalisation qui vise à minimiser le coût opérationnel total tout en tenant compte de l'utilisation courante et future des ressources physiques ainsi que les différentes techniques de migration, ces dernières sont décrites en détail au deuxième chapitre. Le plan indique la nouvelle position des VMs et à travers quelle technique elles doivent être migrées. En d'autres termes, ces VMs sont migrées à l'aide des techniques de migration spécifiques de telle manière à minimiser le coût opérationnel global et parvenir, conséquemment, à un partage optimal.

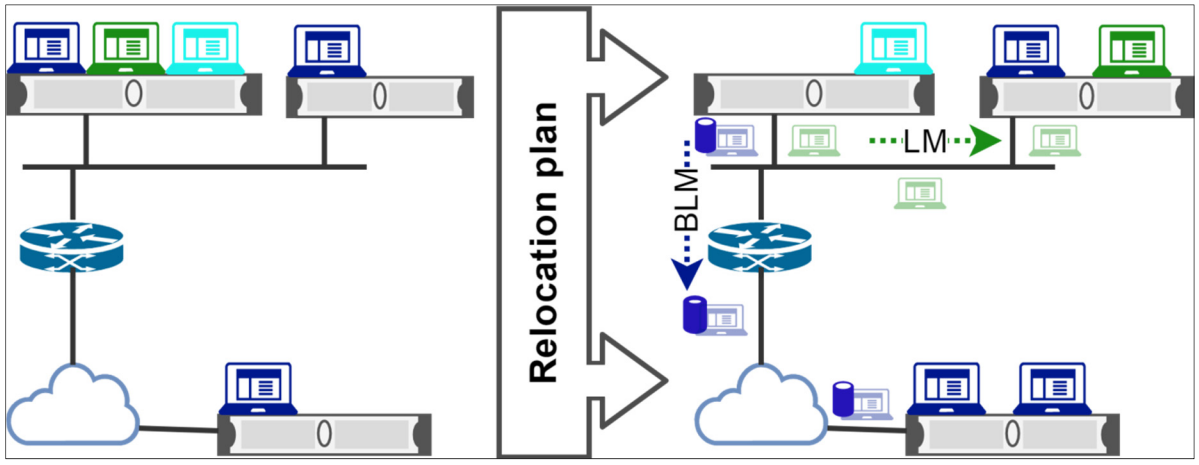

Figure 3.1 Partage des ressources basé sur la relocalisation dynamique

3.3.1 Modèles d'estimation des coûts de relocalisation

Les ressources physiques disponibles dans l'infonuagique peuvent être représentées par deux catégories, des ressources d'hébergements telles que la mémoire, la CPU et le disque, et

celles de réseautage comme la bande passante, les commutateurs et routeurs. L'utilisation de ces ressources implique un coût d'utilisation opérationnel, d'où nous associons à chacune d'entre elles un coût calculé sur la base du taux d'utilisation. Nous considérons parmi les ressources citées ci-haut, la mémoire, la CPU, le disque et la bande passante. Nous appuyons notre choix de ressources à considérer par leur importance dans la minimisation du coût total.

Nous nous intéressons, par la suite, à deux coûts opérationnels, celui d'hébergement et celui de migration. En effet, le premier reflète le coût d'hébergement d'une VM dans le serveur en s'appuyant sur l'utilisation des ressources d'hébergements, et le deuxième reflète le coût de migration de cette VM en se basant sur l'utilisation de la bande passante.

Le coût additionnel est celui qui s'applique sur le coût total lors d'une tentative de relocalisation. Prenons l'exemple d'une VM i sur le point de migrer d'une source k à une destination k', le coût additionnel de sa relocalisation peut être calculé de la manière suivante :

$$Co\hat{u}tAdditionnel = Co\hat{u}tMig + (Co\hat{u}tH\acute{e}berg_{ik'} - Co\hat{u}tH\acute{e}berg_{ik}) \qquad (3.3)$$

En effet, nous calculons ce dernier par l'addition du coût de migration à la différence entre ceux d'hébergement de la VM dans la source et la destination. Dans le cas d'une relocalisation minimisant le coût total, le coût additionnel sera négatif, il sera donc positif dans le cas contraire et égal à zéro si aucune relocalisation à appliquer.

3.3.1.1 Coût de migration

La migration d'une VM est une technique de partage efficace, son efficacité se traduit par ses avantages dont nous avons souligné dans le chapitre précédent. Toutefois, avoir recours à cette technique implique une utilisation significative de la bande passante du réseau entre la source et la destination. En conséquence, et comme l'indique l'équation (3.4), nous estimons le coût de migration $Co\hat{u}tMig$ à travers la multiplication d'un coût prédéfini de l'utilisation

de la bande passante pendant une seconde $coûtBP$ par la durée totale de la migration $TempsMig$.

$$CoûtMig = TempsMig \times coûtBP \quad (3.4)$$

Le coût de l'utilisation de la bande passante $coûtBP$ est prédéfini, quant à la durée totale de la migration, elle dépend de plusieurs facteurs, incluant la bande passante disponible au moment de la migration, la taille des données à transférer et la technique de migration adoptée. En général, cette durée peut être exprimée par la somme du temps de la pré-copie $TempsPC$ et celui d'arrêt $TempsArr$ comme l'indique l'équation (3.5).

$$TempsMig = TempsPC + TempsArr \quad (3.5)$$

Le temps de pré-copie est essentiellement la durée pendant laquelle la mémoire ou le stockage sont migrés avant de suspendre la VM, toutefois le temps de l'indisponibilité, c'est le temps dans lequel les pages mémoires (dirty pages) qui ont été modifiées au cours du processus et qui n'ont pas pu être transmises pendant la première phase, seront retransmises à la destination. Étant donné que chaque technique diffère de par sa logique de migration, nous sommes dans la mesure d'étudier la durée de migration cas par cas. Dans le cadre de notre méthodologie, nous optons pour la migration à chaud, la migration à chaud avec stockage et la migration à froid. Nous avons eu recours au travail de (Pang, 2010) afin d'estimer le temps de migration.

Cas de la migration à chaud :

La migration à chaud consiste à migrer seulement la mémoire et l'état de la CPU, le stockage est partagé entre les différents hôtes et n'est pas transmis.

La phase de pré-copie consiste à migrer la mémoire jusqu'à ce qu'une taille minimale $MinTailleMem$ soit atteinte, la bande passante disponible $TauxBP$ sera utilisée pour cela,

mais aussi pour synchroniser le stockage $TauxSynch$ ainsi que pour envoyer inutilement des pages mémoires qui vont être modifiées $TauxPMM$ pendant le processus de transmission et qui doivent être retransmises. Ainsi, le temps de pré-copie est exprimé par l'équation (3.6).

$$TempsPC = \frac{TailleMem - MinTailleMem}{TauxBP - (TauxSynch + TauxPMM)} \tag{3.6}$$

Pendant la phase d'indisponibilité la source transmet le reste de la mémoire après avoir suspendu la VM, en conséquence, il n'y a plus de pages modifiées. Par ailleurs, le calcul de ce temps est obtenu à l'aide de l'équation (3.7).

$$TempsArr = \frac{MinTailleMem}{TauxBP - TauxSynch} \tag{3.7}$$

Dans notre modèle nous choisissons de négliger le temps d'indisponibilité lorsqu'il s'agit de la migration à chaud. En effet, cette dernière assure généralement un temps d'arrêt très réduit qui ne risque pas d'influencer le temps total (Voorsluys et al., 2009) et d'où le coût de migration. Pour la même raison, nous négligeons les données liées à l'état de la CPU ainsi que taille minimale de mémoire lors de l'estimation du temps de la pré-copie. Nous pouvons exprimer ainsi le temps de migration à chaud par l'équation ci-dessous :

$$TempsMig \approx \frac{TailleMem}{TauxBP - (TauxSynch + TauxPMM)} \tag{3.8}$$

Le cas de la migration à chaud supportée par les volumes est traité de la même manière que la migration à chaud.

Cas de la migration à chaud avec stockage :

La migration à chaud avec stockage consiste à migrer la mémoire, l'état de la CPU et le disque. L'équation (3.9) indique qu'au cours de la phase de pré-copie, le stockage $TailleDisque$ est migré en premier, jusqu'à ce qu'une certaine taille minimale $MinTailleDisque$ soit atteinte, alors qu'un sous-ensemble $TauxBDM$ de la bande passante est utilisé inutilement pour transmettre des blocs du disque modifiés. Puisqu'il ne s'agit pas d'un stockage partagé, aucun flux n'est destiné au processus de partage.

$$TempsPC = \frac{TailleDisque - MinTailleDisque}{TauxBP - TauxBDM} \tag{3.9}$$

La phase d'indisponibilité est caractérisée par sa durée qui peut être importante relativement à la taille de la mémoire de la VM. En effet, c'est pendant cette phase que la mémoire est envoyée ainsi qu'une partie minimale du disque.

$$TempsArr = \frac{MinTailleDisque + TailleMem}{TauxBP} \tag{3.10}$$

Nous négligeons, dans cette technique de migration, la taille minimale du disque $MinTailleDisque$ à cause de sa taille très réduite et qui n'impacte pas remarquablement le temps total de migration et conséquemment le coût, les données liées à la CPU sont aussi négligées en suivant la même logique. Nous supposons également que le stockage ne risque pas d'être modifié lors du processus de migration, de ce fait nous obtenons un taux de blocs modifiés égal à zéro. En conséquence, nous calculons le temps de migration à travers l'équation (3.11).

$$TempsMig \approx \frac{TailleDisque}{TauxBP} + \frac{TailleMem}{TauxBP} \tag{3.11}$$

Cas de la migration à froid:

La migration à froid suspend le client avant de commencer tout transfert de données. Le disque ainsi que la configuration de la VM sont envoyés depuis la source vers la destination. Il est à noter que dans le cas d'Openstack, même si nous parvenons à établir un stockage partagé, le disque est tout de même transmis par la source. Il n'existe pas une phase de pré-copie, mais seulement celle de l'indisponibilité de la machine dont nous rajoutons la durée nécessaire au démarrage de la machine $TempsDR$. Nous négligeons les données dédiées à transmettre la configuration de la machine en vue de leur non-influençabilité sur le coût total. Par ailleurs, nous exprimons le temps de migration par l'équation (3.12).

$$TempsMig = TempsArr \approx \frac{TailleDisque}{TauxBP} + TempsDR \tag{3.12}$$

3.3.1.2 Coût d'hébergement

Chaque VM hébergée dans un hôte alloue des ressources physiques telles que la mémoire, la CPU et le stockage. Cette allocation implique un coût opérationnel qui varie suivant la variation du besoin de la VM. Nous proposons de calculer ce dernier en maintenant des coûts opérationnels d'une durée fixe dans laquelle aucune VM n'est susceptible d'avoir plus ou moins de ressources physiques. Nous obtenons ainsi le coût opérationnel d'hébergement relatif à une instance (VM) i dans un certain serveur, par la somme de la multiplication du coût de l'allocation de la ressource k (exprimé à travers $Coût Alloc_k$), par la quantité allouée à cette instance (exprimée par $AllocRess_{ik}$), comme l'indique l'équation 3.13. À partir de cette section et jusqu'à la fin du chapitre, nous désignons une VM par « instance ».

$$HostingCost_i = \sum_{\substack{k \in \\ \{memory, \\ cpu, disk\}}} Coût Alloc_k \times AllocRess_{ik} \tag{3.13}$$

Il est important de mentionner que le coût d'allocation d'une certaine ressource dépend fortement de son type et sa puissance, autrement le coût d'hébergement peut varier d'un hôte à un autre.

3.3.2 Formulation du problème

Notre fonction objective vise à minimiser le coût opérationnel et ce, en ayant recours à l'utilisation courante et future des ressources de calculs et en considérant plus d'une technique de migration.

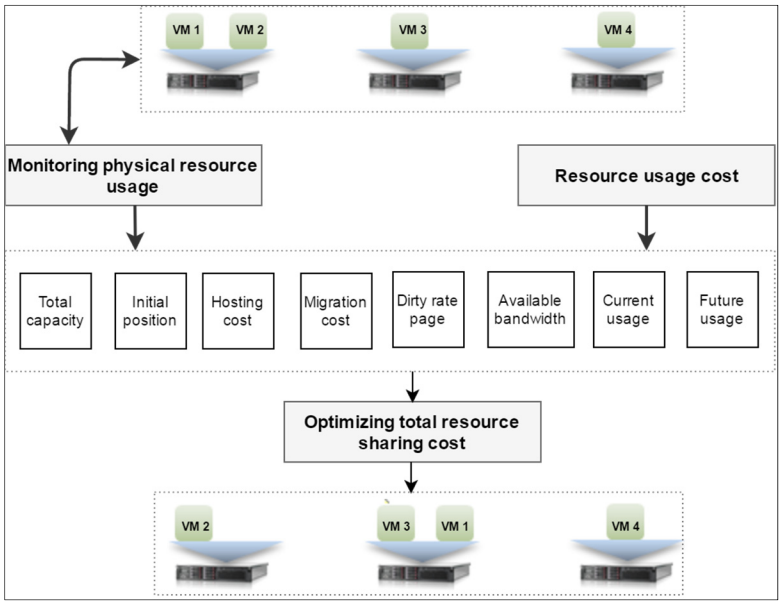

Figure 3.2 Structure du problème

La figure 3.2 définit la structure du problème. Comme entrées, le problème d'optimisation reçoit la capacité totale de chaque ressource physique, la distribution initiale des VMs, leurs coûts d'hébergement et de migration correspondants, la future allocation des ressources physiques, la bande passante disponible et les taux de pages modifiées de chaque VM. La sortie est une nouvelle relocalisation qui vise à minimiser le coût total de partage des ressources physiques. Ci-dessous, nous présentons à travers le tableau 3.1 les variables et constantes que nous utilisons pour la formulation du problème.

Tableau 3.1 Variables et constantes pour la formulation du problème

Symbole	Définition
i, j, k, K	i: instance, j: serveur, k: ressource physique, $k \in K$, $K = \{memory, cpu, disk\}$
z, Z	z: technique de migration, $z \in Z$, $Z =$ {live, block, cold}
n, m	n: nombre d'instances, m: nombre de serveurs
$x_{jj'}^{iz}$	Une matrice binaire qui indique la relocalisation optimale
l_j^i	Indique si l'instance i est hébergée au serveur j
$M_{jj'}^{iz}$	Le coût de migration de l'instance i du serveur j au serveur j' avec la technique de migration z
H_j^i	Le coût d'hébergement de l'instance i au serveur j
R_i^k	La quantité de ressource k allouée à l'instance i, $k \in K$
R_i^m	La quantité de la mémoire allouée à l'instance i
R_i^D	La quantité du disque allouée à l'instance i
F_i^k	La future allocation de la ressource k par l'instance i, $k \in K$
C_j^k	Coût d'allocation de la ressource k au serveur j, $k \in K$
$C_{jj'}^{bw}$	Coût de l'utilisation de la bande passante entre le serveur j et j'
$b_{jj'}$	Bande passante disponible entre serveur j et j'
D_i	Taux de pages modifiées d'une instance i
$TC_{j'}^k$	Capacité totale d'une ressource k dans le serveur j', $k \in K$
T_{max}	Utilisation maximale de toute ressource
T_{res}	Temps de démarrage d'une VM
δ	Un seuil de probabilité de surcharge toléré
p	Une pénalité constante associée au temps d'arrêt pendant la migration

3.3.2.1 Modèle 1 : sans prédiction

Dans un premier temps, nous focalisons sur le partage qui réfère seulement à l'utilisation courante des ressources physiques. Le problème consiste à déterminer la meilleure

relocalisation de manière à minimiser le coût de partage global et à satisfaire les exigences des VMs.

$$x_{jj'}^{iz} = \begin{cases} 1, quand\ l'instance\ i\ doit\ migrer\ du\ serveur\ j\ \text{à}\ j'avec\ la\ technique \\ \quad de\ migration\ z \\ \\ 0, sinon. \end{cases} \tag{3.14}$$

$$H_j^i = \sum_{k \in K} R_i^k C_j^k \,, K = \{memory, cpu, disk\}, i = 1, \dots, n, j = 1, \dots, m \tag{3.15}$$

$$M_{jj'}^{iz} = \begin{cases} \dfrac{R_i^M}{b_{jj'} - D_i} \times C_{jj'}^{bw}, & z = \{live\} \\ \dfrac{R_i^D}{b_{jj'}} C_{jj'}^{bw} + \dfrac{R_i^M}{b_{jj'}} \left(C_{jj'}^{bw} + p \right), & z = \{block\} \\ \left(\dfrac{R_i^D}{b_{jj'}} + T_{res} \right) \left(C_{jj'}^{bw} + p \right), & z = \{cold\} \end{cases} \tag{3.16}$$

$$L_j^i = \begin{cases} 1, quand\ l'instance\ i\ est\ hébergée\ au\ serveur\ j \\ \\ 0, sinon. \end{cases} \tag{3.17}$$

La fonction objective minimise le coût opérationnel de partage des ressources physiques dans l'infonuagique:

$$\text{Minimize} \quad \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{j'=1}^{m} \sum_{z \in Z} x_{jj'}^{iz} \left[M_{jj'}^{iz} + \left(H_{j'}^i - L_j^i H_j^i \right) \right] \tag{3.15}$$

S.T. :

$$\sum_{j,j'}^{m} \sum_{z \in Z} x_{jj'}^{iz} = 1 \,, i = 1, \dots, n \tag{3.16}$$

$$\sum_{i=1}^{n} x_{jj'}^{iz}\, D_i \;< b_{jj'}\; z = \{live\}, \qquad j, j' = 1, \dots, m, j \neq j' \tag{3.17}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{z\in Z} x_{jj'}^{iz},\; R_i^k \;\leq TC_{j'}^{k}\, T_{max},\; j' = 1, \dots, m, k \in K \tag{3.18}$$

La variable de décision est exprimée par (3.14). L'équation (3.15) calcule le coût d'hébergement d'une instance i dans le serveur j en multipliant les quantités des ressources allouées $k \in K$ par leurs coûts d'allocation correspondants. Le coût de migration dans (3.16) est donné suivant la technique de migration, dans l'ensemble, il s'agit de multiplier le rapport de la ressource à transférer (mémoire ou disque) à la bande passante disponible par le coût d'utilisation de la bande passante. Une pénalité constante est appliquée lorsque la technique de migration implique un temps d'arrêt remarquable, soit dans notre cas la migration à chaud avec stockage et celle à froid. Nous avons expliqué précédemment la méthode utilisée pour l'estimation des coûts. Les données qui renseignent quelle VM est hébergée par quel serveur sont exprimées par (3.17), il s'agit d'une donnée d'entrée dans notre modèle (connue auparavant). L'équation (3.18) représente la fonction objective qui minimise le coût opérationnel total du partage des ressources physiques dans le nuage informatique. Ce coût comprend les coûts des différentes migrations et ceux d'hébergements. La variable l_j^i, nous permet de considérer le coût d'hébergement courant d'une VM. La sortie est une matrice qui indique les relocalisations à effectuer pour obtenir un partage efficace. L'équation (3.19) indique qu'une seule relocalisation par instance est possible, s'une instance i dois demeurer dans le même serveur j qu'avant, dans ce cas x_{jj}^{iz} est égale à 1, à ce niveau la technique de migration z n'est plus importante. La contrainte dans (3.20) indique que le taux total des pages modifiées des instances candidates à la migration du serveur j au serveur j' ne doit pas dépasser la bande passante disponible entre ces serveurs, cette contrainte ne s'applique qu'à la migration à chaud, étant donné qu'elle assure la migration de la mémoire durant le stade de la pré-copie. Dans (3.21), la quantité totale allouée de la ressource $k \in K$ par les instances

existantes dans un serveur j et celles qui vont y migrer doit être inférieure ou égale à certain seuil spécifié par T_{max} par rapport à la capacité totale.

3.3.2.2 Modèle 2 : avec prédiction

Le premier modèle que nous avons introduit s'intéresse à l'utilisation courante des ressources physiques afin de décider le meilleur plan de relocalisation qui permet un partage optimal de ces dernières. Cependant, comme les VMs ont des besoins, en termes de ressources physiques, qui sont variables au cours du temps et que la disponibilité des serveurs change fréquemment, en particulier lors de la création de nouvelles VMs, l'infrastructure physique peut être surchargée. Cela aura un impact négatif sur la performance de l'infrastructure physique et la QoS (quality of service). En conséquence, nous ajoutons une nouvelle contrainte basée sur le modèle de prédiction précédemment introduit.

$$\Pr_{t_1}\left[\left(\frac{\sum_{j=1,j\neq j'}^{m}\sum_{i=1}^{n}\sum_{z\in Z} x_{jj'}^{iz}\ F_i^k\ +\ \sum_{i=1}^{n}\sum_{z\in Z} x_{j'j'}^{iz} F_i^k}{TC_{j'}^k} - T_{max}\right)\geq 0\right]\leq \delta \tag{3.19}$$

$j' = 1, \ldots, m, k \in K$

Dans (3.22), lorsque le rapport de la future quantité totale allouée de ressource $k \in K$ à sa capacité totale dans le serveur j' est égal ou supérieur à un seuil fixe T_{max}, la ressource est considérée comme trop utilisée. La quantité totale allouée considère l'allocation par les instances existantes dans le serveur j' et celles qui vont migrer vers lui. La contrainte indique que la probabilité que la ressource $k \in k$ soit surchargée doit demeurer inférieure ou égale à une constante δ. Grâce à cette contrainte, nous pouvons éviter une surexploitation probable d'une certaine ressource dans le serveur j'.

Étant la non-linéarité de cette contrainte, nous devons la reformuler autrement. Nous nous sommes basé sur les travaux de (Wang, Meng et Zhang, 2011) et (Vakilinia, Qiu et Ali, 2014) pour linéariser le modèle proposé. L'équation (3.22) peut être alors exprimée de la manière suivante :

$$\frac{\sum_{j=1,j\neq j'}^{m}\sum_{i=1}^{n}\sum_{z\in Z}x_{jj'}^{iz}\ F_i^k\ +\ \sum_{i=1}^{n}\sum_{z\in Z}x_{j'j'}^{iz}F_i^k}{TC_{j'}^k} \tag{3.20}$$

$$+\ \varphi^{-1}(1-\delta)\ \frac{\sum_{j=1,j\neq j'}^{m}\sum_{i=1}^{n}\sum_{z\in Z}x_{jj'}^{iz}\ \sigma_i^k\ +\ \sum_{i=1}^{n}\sum_{z\in Z}x_{j'j'}^{iz}\sigma_i^k}{TC_{j'}^k}$$

$$<\ T_{max}, j'=1,\dots,m, k\in K$$

Dans cette équation $\varphi^{-1}(1-\delta)$ est la fonction inverse de la CDF de la distribution normale et σ_i^k est la déviation standard obtenue à partir du modèle ARIMA.

3.3.3 Algorithme

L'objectif de l'algorithme proposé est d'effectuer le meilleur plan de relocalisation qui minimise autant que possible le coût opérationnel total du partage de l'infrastructure physique. Nous considérons les utilisations courantes et futures des ressources physiques parmi les VMs. En outre, les décisions concernant les techniques de migration doivent être effectuées en fonction de leur contribution à la réduction des coûts.

L'algorithme illustre les différentes instructions pour mettre en œuvre notre problème d'optimisation. Il consiste à initialiser le problème en tant que problème linéaire que nous appelons «$SharingProb$» et X comme matrice binaire, puis définir la fonction objective ainsi que les contraintes à travers la fonction $lpsum$ de la bibliothèque $Pulp$. À la fin, l'algorithme appelle le solveur qui est basé sur la méthode Simplex. Une fois que le problème est résolu, le migrateur est appelé pour effectuer les déplacements nécessaires selon le plan de relocalisation renseigné dans X.

Comme entrée, l'algorithme se voit attribuer le nombre d'instances, de serveurs, de ressources et de techniques de migration qui correspondent à nb_{inst}, nb_s, nb_r et nb_m. Les coûts de migration et d'hébergement désignés par $Mcost$ et $Hcost$ sont précédemment calculés en fonction de la distribution des VMs puis transmises à notre algorithme. Le reste des variables

$pos, B, TC, Usage, FUsage$, D et $Migrator$ correspondent à la position initiale des VMs, la bande passante disponible, la capacité totale, l'allocation courante des ressources physiques, leur allocation future, au taux de pages modifiées de toutes les VMs et à l'exécuteur des migrations spécifiées. L'algorithme accepte également en entrée SE, $Tmax$, δ, μ et σ comme l'erreur standard, le seuil au-dessus duquel la ressource est considérée surchargée, la probabilité maximale tolérée d'une éventuelle surcharge de ressource, moyenne et déviation standard de la distribution normale.

Algorithme 3.1 : Minimisation du coût opérationnel

Input : $nb_{inst}, nb_s, nb_r, nb_m, Mcost, Hcost, pos, B, TC, Usage, FUsage, D, Migrator, Tmax, \delta, SE, \mu, \sigma$

Output : X

$prob = LpProblem("SharingProb", LpMinimize)$

$relocation = getList(nb_{inst}, nb_s, nb_r, nb_m)$

$X = LpVariable.dicts("Unknown", relocation, 0, 1, LpBinary)$

$prob \mathrel{+}= lpsum(X(i,j,z,k)\left[Mcost(i,j,z,k) + \left(Hcost(i,z) - pos(i,j)Hcost(i,j)\right)\right]$ For i in nb_{inst} For j in nb_s For z in nb_s For k in nb_m)

For i in nb_{inst}:

  $prob \mathrel{+}= lpsum(X(i,j,z,k)$ For j in nb_s For z in nb_s For k in $nb_m) == 1$

  For j in nb_s:

    For z in nb_s:

      If $j != z$:

        $prob \mathrel{+}= lpsum(X(i,j,z,1)\ D(i)$ For i in $nb_{inst}) \leq B(j,z)$

For z in nb_s:

  For r in nb_r:

    $prob \mathrel{+}= lpsum(X(i,j,z,k)\ Usage(i,r)$ For i in nb_{inst} For j in nb_s For k in $nb_m) \leq TC(z,r) * Tmax$

For z in nb_s:

  For r in nb_r:

    $prob \mathrel{+}= lpsum(X(i,j,z,k)\ \frac{FUsage(i,r)}{TC(z,r)}$ For i in nb_{inst} For j in nb_s For k in $nb_m)$

    $+ppf(1-\delta,\mu,\sigma)lpsum\left(X(i,j,z,k)\ \frac{SE(i,r)}{TC(z,r)}\right.$ For i in nb_{inst} For j in nb_s For k in $nb_m\left.\right) < Tmax$

$prob.solve()$

$Migrator.migrate\ (X)$

Les lignes 6, 10, 13 et 16 introduisent d'une manière cumulative les contraintes décrites par les équations (3.19), (1.20), (3.21) et (3.22). La ligne 17 fait appel au solveur qui est basé sur Simplex, ce dernier permet de calculer le plan de relocalisation le plus optimal.

La sortie est la variable de décision X qui présente le nouveau plan de relocalisation optimisant le partage des ressources physiques. Nous soulignons également que la complexité de l'algorithme proposé est O ($I \times S^2$) où I est le nombre d'instances existantes et S est le nombre de serveurs.

3.4 Système proposé

3.4.1 Hypothèses du système

Dans le cadre de notre méthodologie, il est important de spécifier que le système proposé s'appuie sur les hypothèses suivantes:

- Les connexions entre les différentes VMs ne sont pas considérées, c'est-à-dire que les VMs communicantes ne sont pas forcément placées dans des serveurs proches;
- Les types d'applications qui s'exécutent sur les VMs ne sont pas pris en compte;
- La migration n'implique pas l'utilisation de plus d'un chemin réseau;
- L'utilisation du cache mémoire n'est pas considérée.

3.4.2 Architecture du système

Au niveau de cette sous-section nous présentons l'architecture du système dans son intégralité, ce qui nous permet de répondre à notre troisième objectif (section 1.4). Ce dernier est composé d'un ensemble de modules proposés tels qu'illustrés par la figure 3.3, chacun d'eux a ses propres fonctionnalités, toutes orchestrées par le noyau.

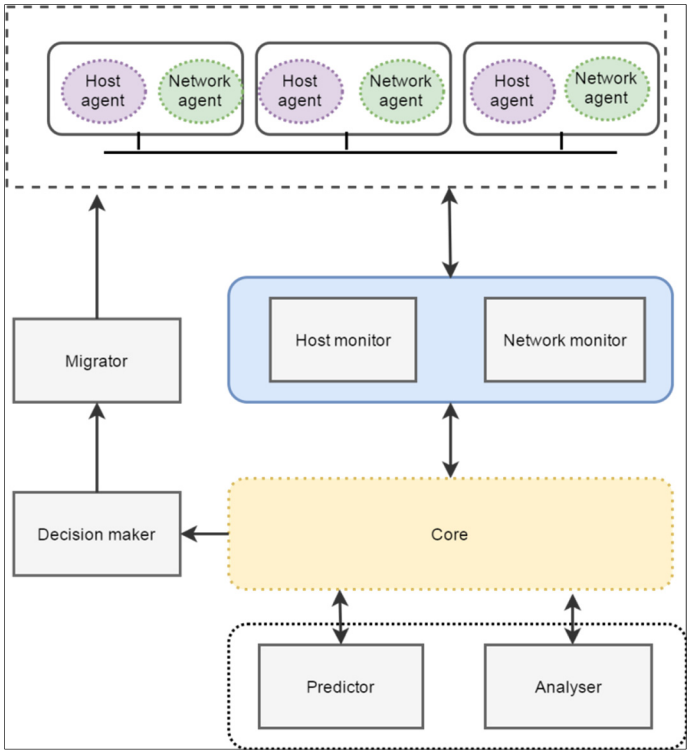

Figure 3.3 Architecture du système

Dans ce qui suit, nous détaillons le fonctionnement de chacun:

- Les agents hôte et réseau sont des agents qui s'exécutent sur chaque nœud physique afin de reporter des informations relatives à la consommation des ressources physiques par les VMs;
- Les moniteurs hôte et réseau permettent de construire une vue globale de l'utilisation des ressources physiques par les VMs, et ce en communiquant avec les différents agents locaux;
- L'analyseur assure l'analyse des données capturées afin de fournir des modèles de prédiction;
- Le module de prédiction s'appuie sur les modèles de prédiction fournis dans le but de prédire la future utilisation des ressources physiques et leur disponibilité;
- Le module de prise de décision est le plus important, ce dernier implémente notre modèle de partage des ressources et assure en conséquence un partage avec un coût

opérationnel optimal, pour ce faire, il s'appuie sur les informations produites par les modules précédents;

- Le migrateur est un module qui performe essentiellement la relocalisation dynamique selon la sortie du module de prise de décision;
- Le noyau permet d'orchestrer l'ensemble des tâches effectuées par les différents modules et rediriger le flux entre elles.

Ci-dessous nous présentons un scénario qui montre la manière avec laquelle le système procède pour l'obtention d'une relocalisation optimale.

1- Les modules de suivi de l'hôte et du réseau collectent les données depuis les agents réseaux et hôtes se trouvant dans chaque nœud physique, les données incluent l'utilisation des composants physiques mais également la position courante de chaque VM;
2- Le module noyau appelle les modules moniteurs afin de capturer les données qui concerne l'infrastructure sous forme de fichiers et les passe à l'analyseur;
3- L'analyseur reçoit ces données, les analyse et crée les modèles de prédiction, ces derniers sont renvoyés au noyau (le nombre de modèles dépend des ressources à considérer ainsi que le nombre des VMs);
4- Le noyau passe les modèles au prédicteur, qu'à son tour assure la prédiction de la future utilisation des ressources par chaque VM et les lui renvoie;
5- Le module noyau appelle celui de prise de décision, en lui associant les données relatives à l'état courant et futur d'utilisation des ressources physiques par les VMs ainsi que d'autres informations, telles que les coûts, la position courante de chaque machine, la disponibilité de la bande passante...etc;
6- Le module de prise de décision exécute le modèle d'optimisation et produit une stratégie de relocalisation qu'il juge optimale;
7- Le résultat est ensuite passé au migrateur qui est responsable de la configuration et l'exécution de la migration suivant la stratégie renseignée;
8- Le migrateur exécute le plan de relocalisation.

3.5 Conclusion

En guise de conclusion à ce chapitre, nous avons présenté notre méthodologie qui répond à notre problématique de recherche d'une part et nos objectifs d'une autre part. Nous avons présenté en particulier le modèle à travers lequel nous aboutissons à un partage optimal de point de vue coût opérationnel. À ce niveau, nous avons présenté notre méthode d'estimation des coûts opérationnels suivant l'utilisation des ressources, nous avons également formulé le problème de partage de ressource.

CHAPITRE 4

RÉSULTATS EXPÉRIMENTAUX

4.1 Introduction

Ce chapitre portera sur les expérimentations et résultats nécessaires pour valider l'hypothèse de notre recherche. De ce fait, nous implémentons le système que nous avons proposé au niveau de la méthodologie, en particulier les modules nécessaires à la prédiction et celui de prise de décision, ensuite nous acheminons nos expérimentations.

Au cours de ce chapitre, nous commençons par l'implémentation du système, ensuite nous présentons notre protocole expérimental. Nous exposons, par la suite, l'environnement du travail qui est divisé en plateforme physique et l'outil infonuagique Openstack, par ailleurs, nous introduisons un scénario, présentons et interprétons les résultats obtenus. Nous menons par la fin une discussion pour souligner ce que nous avons achevé.

4.2 Implémentation du système

Au niveau de la méthodologie, nous avons présenté l'architecture du système. Dans cette section ne détaillons son implémentation.

4.2.1 Collecte des données

La collecte des données est un mécanisme fondamental dans le nuage informatique, sa plus courante utilisation vise à fournir suffisamment d'informations pour mettre en œuvre un système de facturation précis et juste. Ce mécanisme peut être notamment déployé pour répondre aux besoins des VMs en termes de ressources physiques et respecter la disponibilité des hôtes à la fois. En effet, les données peuvent être enregistrées dans une base de données et analysées à la suite pour des fins de prédiction. Notre objectif derrière la collecte des

données est la construction des modèles de prédictions permettant de prédire le futur besoin des VMs, cela enrichit certainement le partage des ressources dans l'infonuagique.

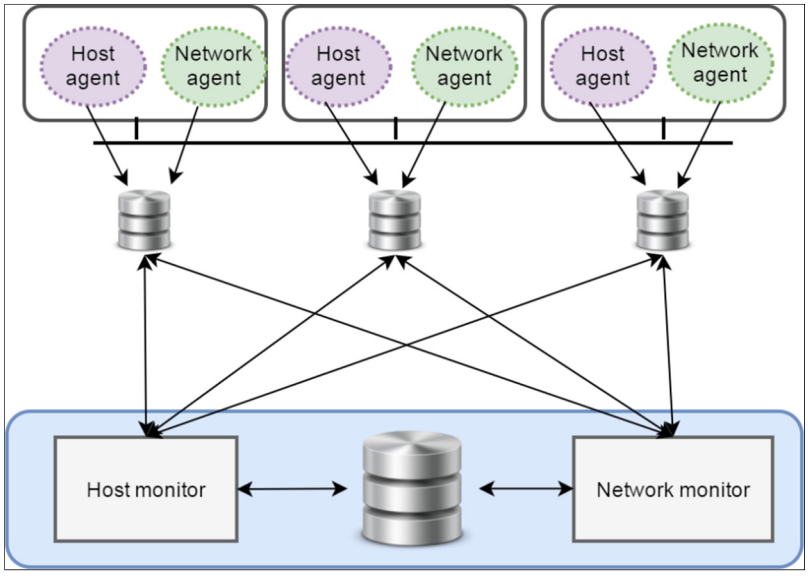

Figure 4.1 Stratégie de collecte des données

Comme l'indique la figure 4.1, il existe des modules moniteurs appelés « Host monitor » et « Network monitor », qui communiquent périodiquement avec des agents situés dans chaque nœud physique afin d'obtenir les données nécessaires. La communication est fondée sur des bases de données caractérisées par des fichiers. Toutes les informations qui concerne l'infrastructure globale sont enregistrées dans une base de données centralisée, et sont sollicitées, ensuite, par les moniteurs sur demande.

4.2.1.1 Agents de collecte des données

Les agents de collecte que nous proposons ont comme mission de collecter les données spécifiques à l'utilisation des ressources dans chaque hôte, de ce fait, ils sont répartis sur tous les hôtes de l'infrastructure physique. Ces derniers reportent les données collectées à des modules moniteurs à travers des bases de données locales sous forme de fichiers. Des intervalles de temps sont fixés afin de collecter les données sous des périodes bien uniformes.

Lorsqu'un nouvel hôte est rajouté à l'infrastructure, l'agent doit obligatoirement y être déployé, sans quoi nous ne serons pas en mesure de réaliser la collection.

Dans le cadre de notre projet nous avons le choix d'utiliser soit le service Ceilometer soit un algorithme développé en python présent dans chaque hôte et qui permet de collecter seulement les données dont nous avons besoin. Nous avons choisi d'utiliser notre propre algorithme, en effet, l'utilisation de Ceilometer requiert une architecture complexe généralement nécessaire à un système de facturation, en plus, comme nous l'avons expliqué dans le chapitre précédant, ce service se base sur un ensemble de démons qui opèrent en collaboration pour fournir plusieurs métriques, cela fait preuve d'une grande performance, mais étant données que nous avons besoin de quelques métriques, il devient plus intéressant d'adopter une architecture simple basée sur un ensemble d'agents de collecte des données.

Agent hôte :

L'agent hôte se trouve dans chaque hôte, et permet de mesurer un ensemble de métriques. Plus précisément il s'agit d'un algorithme développé en python ayant comme fonction de rassembler des données et les enregistrer dans une base de données locale, et ce d'une manière périodique. Les métriques dont nous considérons sont :

- La mémoire : la mesure du taux d'utilisation de la mémoire physique par chaque VM est essentielle pour assurer la bonne exécution des applications qui y sont hébergées, surtout lorsque celles-ci sont instables au cours du temps;
- La CPU : c'est un composant important pour garantir une meilleure performance de la VM et en conséquence une meilleure qualité de service, nous mesurons ainsi son utilisation par chaque VM afin de répondre à leur futur besoin;
- Le disque : il s'agit d'un composant qui assure le stockage en bloc, d'une manière générale, chaque VM peut en avoir, que ce soit sous la forme d'un stockage éphémère ou volumes, dans les deux cas le bon partage du disque mène à une meilleure performance, d'où notre choix de mesurer l'utilisation du stockage par ces VMs.

4.2.1.2 Agent réseau

Ayant la même logique que l'agent hôte, il s'agit d'un algorithme développé en python et présent dans chaque hôte, ce dernier permet de collecter des données indiquant le taux d'envoie et réception de flux à partir d'une ou plusieurs interfaces connectées aux différents réseaux (de management, de données et externe), en s'appuyant sur ça, nous obtenons le taux d'utilisation de la bande passante. La mesure du taux d'utilisation de la bande passante est très important dans notre projet, et ce afin d'assurer une meilleure qualité de relocalisation dynamique (migration des VMs) et calculer son coût opérationnel.

4.2.1.3 Modules moniteurs

Nous proposons deux modules responsables du suivi de l'utilisation globale des ressources physiques au niveau de l'architecture sous-jacente. Ces modules communiquent périodiquement avec les différents agents (agents hôtes et réseaux) situés dans chaque nœud à travers des bases de données locales (fichiers), le but est de rassembler toutes les informations requises pour les modules d'analyse, de prédiction et de décision. Les deux modules forment une base de données globale, cette dernière maintient en conséquence les métriques précédemment définies de tous les nœuds.

Ci-dessous nous expliquons les deux modules moniteurs :

- Le moniteur de l'hôte permet de construire une base de données complète ayant comme informations, le taux d'utilisation de mémoires, unités de calculs et disques par toutes les VMs dans toute l'infrastructure physique. Il est également utile pour reporter la position initiale de ces VMs. Cette base est ensuite accessible pour prédire la future utilisation des ressources et la disponibilité des nœuds physiques, ainsi que pour décider la relocalisation dynamique.
- Le module de suivi de réseau suit le même concept, sauf que les informations reportées concernent l'utilisation des différentes bandes passantes. Similairement, ces informations serviront ultérieurement au niveau du module de décision.

4.2.2 Modules de prédiction de l'utilisation des ressources physiques

Nous considérons la prédiction de l'utilisation des ressources physiques dans l'infonuagique comme élément primordial permettant d'une part de répondre aux besoins des VMs et d'autre part d'éviter la mauvaise utilisation des ressources physique. En général une prédiction ayant une haute précision mène fréquemment à un meilleur partage des ressources, d'où notre intérêt.

Nous suggérons dans notre système deux modules appelés « Analyseur » et « Prédicteur », ensembles, ils ont la mission de fournir des données exprimant la future utilisation des ressources physiques.

4.2.2.1 Analyseur

Dans notre cas l'analyseur reçoit les données relatives à l'utilisation des ressources physiques par chaque VM dans chaque serveur des centres de données. L'analyseur extrait seulement les données valides à partir desquelles il peut construire un modèle de prédiction. La sélection des données consiste sur :

- L'élimination des données non-significatives afin d'assurer une meilleure qualité de prédiction. Une donnée non-significative ne s'agit pas d'une valeur numérique, mais plutôt d'un symbole qui exprime l'incapacité du moniteur à fournir une information à un moment précis;
- La sélection des données selon un intervalle de temps fixe, à condition que cet intervalle ne soit inférieur à celui qui correspond à la fréquence de capture des données par le moniteur. En conséquence nous obtenons des séries temporelles à partir desquelles nous obtenons notre modèle statistique de prédiction.

L'analyseur a également comme mission de produire des modèles statistiques nécessaires à la prédiction de la future utilisation des ressources physiques par chaque VM.

Les modèles sont construits à partir du modèle statistique autorégressif à moyenne mobile intégrée (ARIMA) avec des paramètres fixés auparavant. Le modèle est obtenu sur la base de l'historique d'utilisation des ressources physiques, ce dernier considère ainsi p dernières valeurs d'utilisation ainsi que q dernières erreurs de prédiction.

4.2.2.2 Prédicteur

Le prédicteur reçoit les modèles et assure la prédiction des prochaines valeurs qui correspondent à la future utilisation des ressources en sollicitant la fonction de prédiction. Ce module est appelé autant de fois que le nombre de VMs existantes et les ressources concernées, et ce pour déterminer leurs futurs besoins.

4.2.3 Modules de relocalisation

Dans l'architecture du système décrite au niveau de la méthodologie, nous proposons deux modules, l'un assure la prise de décision et l'autre effectue ces dernières.

4.2.3.1 Module de prise de décision

Ce module est celui qui prend les décisions concernant une architecture infonuagique. En effet, il implémente notre modèle de relocalisation, d'où l'algorithme présenté au niveau de la section 3.3.3. De ce fait, il permet de générer des plans de relocalisation assurant un partage optimal et ce en ayant recours à la prédiction de l'utilisation courante et future des ressources physiques. Les plans de relocalisation renseignent sur les nouvelles positions des VMs ainsi que la technique de migration à adopter.

4.2.3.2 Module de migration des VMs

Nous proposons un module que nous appelons « migrateur ». Ce dernier effectue les migrations désignées par les plans de relocalisation, il prend en entrée une matrice binaire qui lui indique quelle VM doit être migrée à partir de quelle source vers quelle destination et par

quelle technique de migration. Ce module doit s'appuyer sur l'API client nova afin de configurer et exécuter la migration. Trois techniques sont prises en considération, la migration à chaud, celle à chaud avec stockage ainsi que la migration à froid. Seule la migration à chaud nécessite une configuration d'un stockage partagé, ce qui peut être réalisé avec la technologie NFS (network file system).

4.3 Protocole expérimental

Nous avons présenté notre méthodologie dans le chapitre 3, cette dernière utilise un modèle de prédiction et propose un autre de relocalisation qui permet la génération et l'exécution des plans de relocalisation incluant différentes techniques de migration et se basant sur l'état courant et futur de l'utilisation des ressources physiques.

Notre protocole expérimental suit cette méthodologie et valide notre hypothèse à travers des étapes indispensables que nous présentons ci-dessous :

- Étape 1 : nous commençons tout d'abord par la préparation de notre environnement expérimental (testbed). Cet environnement s'appuie sur un ensemble de serveurs physiques distribués entre Montréal et Moncton sur lesquelles nous déployons notre plateforme infonuagique (Openstack). Chaque serveur se voit attribuer les services qui sont adéquats avec son rôle, c'est-à-dire, pour contrôler l'ensemble de l'infonuagique, gérer la partie réseau, héberger les VMs ou le stockage. En général, l'environnement expérimental nous accorde l'opportunité de tester les différents techniques de migration, de gérer les VMs et de mettre en œuvre des plans de relocalisation assurant un meilleur partage de ressources;
- Étape 2 : dans une seconde étape, nous définissons un scénario bien spécifique à travers lequel nous pouvons obtenir des résultats et les analyser en conséquence. D'une manière générale, ce scénario consiste à cinq VMs ayant différentes tailles et des besoins qui varient au cours du temps, ces dernières sont distribuées sur trois nœuds de calculs. Chaque période de 15 minutes, notre modèle produit un nouveau

plan de relocalisation en s'appuyant sur l'utilisation courante et future des ressources physique. Ces ressources sont réassignées aux différentes VMs en se basant sur les valeurs prédites. La réassignation suit le même intervalle. Nous considérons, au niveau de ce scénario, une période de 3 heures dans laquelle, nous devons obtenir 12 plans de relocalisation;

- Étape 3 : cette étape consiste, à tester le modèle de prédiction souligné dans notre méthodologie. Cette prédiction concerne différentes ressources physiques telles que la mémoire, la CPU et le disque. Pour ça, nous utilisons des données précédemment supervisées qui reflètent divers comportements d'utilisation des ressources. Nous utilisons également, dans ce cadre, le modèle ARIMA dont nous faisons le choix des paramètres. De plus, dans notre modèle, les modules d'analyse et de prédiction sont sollicités périodiquement (toutes les 15 minutes);
- Étape 4 : au niveau de cette étape, nous veillons à tester le modèle de relocalisation suggéré au niveau de la méthodologie. Nous élaborons quatre stratégies, dans la première nous acheminons des comparaisons entre les différentes techniques de migration. La deuxième consiste à une technique hétérogène (MIXT) dans laquelle plusieurs techniques de migration sont utilisées et ce, selon leur contribution dans la minimisation du coût opérationnel. Au niveau de la troisième stratégie, nous comparons le modèle 1 avec le modèle 2 (tels que définis dans la section 3.3.2), c'est-à-dire notre solution avec/sans prédiction. La quatrième assure une comparaison avec une heuristique proposée dans un travail antérieur.

4.4 Environnement expérimental

L'environnement expérimental est le cadre dans lequel nous acheminons nos expérimentations dans le but de valider notre travail. De ce fait, nous utilisons un ensemble de serveurs distribués dans deux centres de données différents et ayant des caractéristiques spécifiques. Le nuage informatique est par la suite déployé sur l'ensemble des serveurs, à partir de là nous pouvons créer, modifier, migrer et supprimer des VMs, au-delà de ces

tâches, nous pouvons manipuler des réseaux, routeurs et commutateurs virtuels ainsi que l'ensemble des services permettant la mise en œuvre de l'infonuagique.

4.4.1 Configuration physique

La plateforme physique sur laquelle nous menons nos expérimentations est composée d'un ensemble de serveurs majoritairement propriétés d'Ericsson. Ces derniers sont caractérisés de leur robustesse et tolérance aux pannes, de plus, ils peuvent être facilement configurables afin de mettre en place un environnement infonuagique.

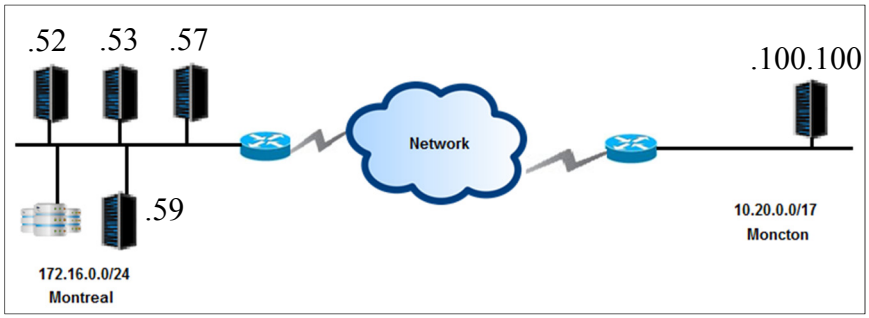

Figure 4.2 Configuration physique de l'environnement expérimental

Au total, nous utilisons cinq serveurs (figure 4.2), se trouvant à Montréal et Moncton, et faisant partie de centres de données différents. Leurs capacités sont décrites par le tableau 4.1.

Les quatre premiers serveurs sont connectés au réseau 172.16.0.0/16 alors que le cinquième est connecté au réseau 10.20.0.0/17. Une route statique est configurée entre lui et l'ensemble des serveurs distants pour qu'ils puissent communiquer. Chacun possède une interface publique ayant 1G de capacité, sauf le serveur 5 qui a une interface de 10G. Nous procédons avec une connexion SSH (secure shell) pour accéder à chaque serveur, de plus, nous les configurons de telle sorte que chacun d'entre eux puisse accéder à l'autre via une connexion SSH sans l'exigence d'un mot de passe, ceci peut s'avérer important pour faciliter le fonctionnement des services infonuagiques par la suite.

Tableau 4.1 Caractéristiques de l'infrastructure physique

	Système	CPU	Mémoire	Disque	Lieu
Serveur 1	Ubuntu Server 14.04	16	24G	250G	Montréal
Serveur 2	Ubuntu Server 14.04	16	24G	250G	Montréal
Serveur 3	Ubuntu Server 14.04	16	24G	250G	Montréal
Serveur 4	Ubuntu Server 14.04	16	24G	250G	Montréal
Serveur 5	Ubuntu Server 14.04	24	64G	1T	Moncton

4.4.2 Déploiement d'Openstack

Dans le cadre de notre projet, nous avons opté vers la technologie Openstack, cette dernière est déployée dans chaque serveur comme étant un ensemble de services collaboratifs. Ces services sont installés en s'appuyant sur le futur rôle du serveur, c'est-à-dire, soit un nœud de contrôle, de réseau, de calcul, de stockage ou bien multifonctionnel où tous les services y résident. Notre approche consiste à une architecture composée de plusieurs nœuds afin d'atteindre la performance exigée, et pouvoir également migrer des VMs d'un serveur source à un serveur destination. Nous utilisons KVM comme hyperviseur pour virtualiser notre structure, de ce fait, nous nous sommes référé au travail de (Habib, 2008) pour son installation et sa mise en œuvre.

4.4.2.1 Approche multi-nœuds

Comme nous l'avons mentionné, nous déployons Openstack suivant une approche multi-nœuds, plus précisément, nous avons les nœuds ci-dessous :

- Nœud de contrôle : il s'agit dans notre cas d'une VM, sur le serveur 1, ayant 12G de mémoire, 8 Unités de calculs et 125G de stockage. Ce nœud est celui qui héberge la plupart des services fondamentaux pour la gestion de l'infonuagique au complet. Il

n'héberge pas de VMs, mais il assure leur gestion en communiquant et ordonnant les autres nœuds. Le stockage est requiert au niveau de ce dernier afin de permettre de journaliser toute activité exercée sur la plateforme;

- Nœud de réseau : c'est également une VM hébergée au serveur 1 et ayant les mêmes caractéristiques que le nœud de contrôle. Cette machine héberge des services qui assurent la gestion des réseaux, routeurs et commutateurs virtuels. Elle est aussi le seul moyen à travers lequel les nœuds de calculs fournissent la connectivité externe aux VMs qu'ils hébergent;
- Nœuds de calculs : chaque nœud de calculs est un serveur physique ayant les caractéristiques que nous avons mentionnées dans le tableau 4.1. Les serveurs 3, 4 et 5 sont, dans notre cas, les nœuds de calculs en question. Ces entités incluent des services de calculs comme « Nova compute » et de réseau comme « q-agt » qui communiquent continuellement avec le nœud de contrôle et de réseau. C'est dans ceux-ci que les VMs sont hébergées, d'où l'importance d'utiliser des serveurs physiques et non virtuels;
- Nœud de stockage : il s'agit d'un nœud facultatif, mais utile lorsque nous avons recours à la création de plusieurs volumes. Ainsi, nous l'utilisons pour mettre en place des volumes, ensuite, les associer aux VMs. Dans la description du tableau 4.1, il s'agit du serveur 2.

4.4.2.2 Réseau virtuel d'Openstack

L'architecture est composée de trois réseaux, celui de données, de gestion et le réseau externe. Comme le montre la figure 4.3, les nœuds de calculs et de réseau sont membres du réseau de données, ce dernier est caractérisé par des tunnels VXLAN, permettant la communication intra-réseau et la redirection des flux vers le nœud de réseau en cas de communication inter-réseau virtuel. Celui de gestion est accessible par tous les hôtes pour assurer un bon fonctionnement d'Openstack, c'est également celui à travers lequel tous les nœuds communiquent les uns avec les autres. Le réseau externe n'est cependant accessible que par le nœud réseau puisqu'il s'agit du seul nœud offrant un accès internet aux VMs. Dans

notre cas, tous ces réseaux sont agrégés au même, autrement, la séparation est en réalité logique et non physique.

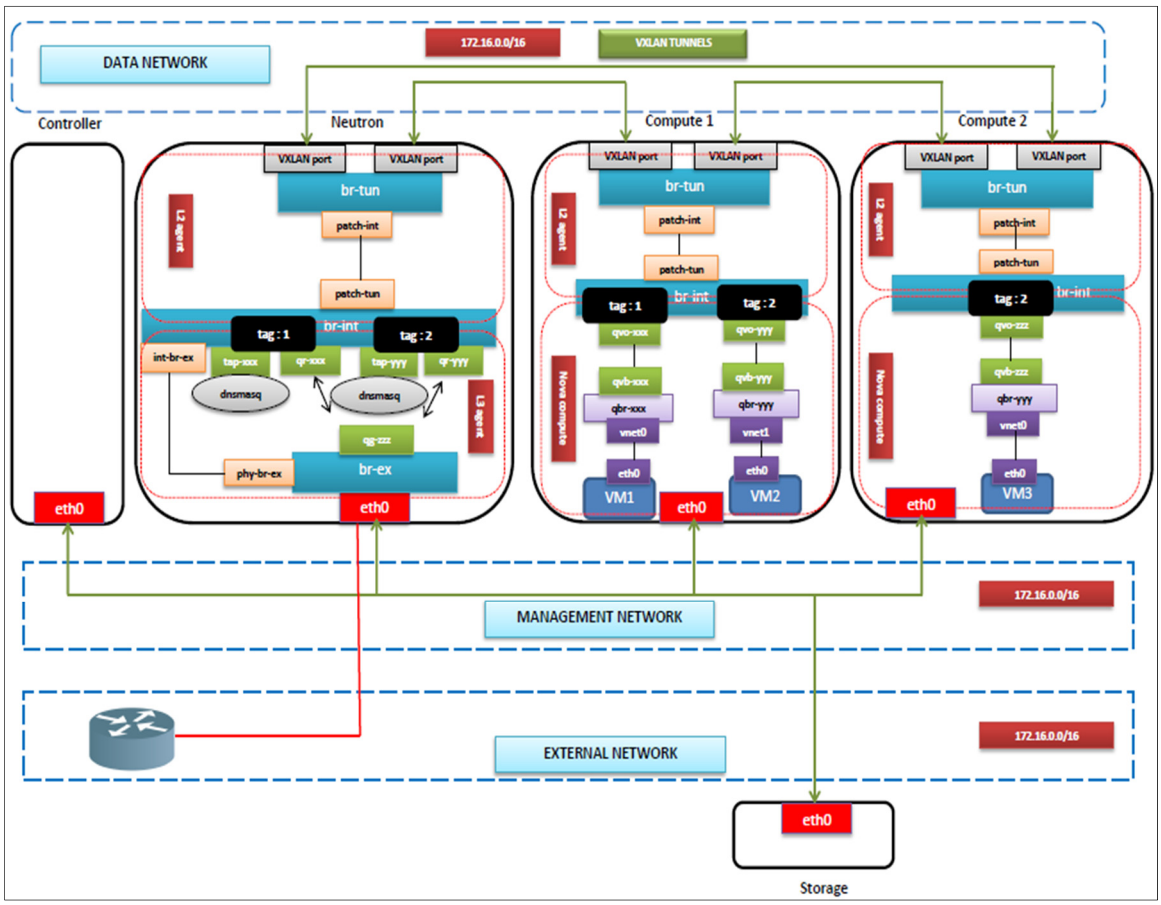

Figure 4.3 Architecture réseau de la plateforme infonuagique

L'architecture du réseau (figure 4.3) est essentiellement configurée par trois agents, l'agent de calcul "nova-compute", celui de deuxième couche réseau L2 et de troisième couche L3.

L'agent de calcul, dans ce contexte, a pour tâche de connecter la VM à un pont Linux « qbr-yyy » à travers des interfaces virtuelles, par exemple « Vnet1 », il crée également une paire d'interfaces Ethernet virtuelle « veth » telle que « qvb-yyy » et « qvo-yyy » afin d'attacher le pont Linux au pont interne « br-int ». Il est important de mentionner qu'une paire « veth » (spécifiquement « qvo ») a une étiquette, cette dernière permet à Openstack de distinguer

deux réseaux virtuels différents (balises VXLAN). De l'autre côté, l'agent de couche deux utilise des interfaces de patch pour relier le pont interne et le tunnel « br-tun » qui est responsable de la communication intra-réseau privé. L'agent de couche trois, assure sa gestion et configure les routeurs virtuels, il s'agit d'un moyen pour se connecter au réseau public. Un routeur a un minimum de deux interfaces, l'une attachée au réseau externe par exemple "qg-zzz" et une autre attachée au pont interne sous une étiquette VXLAN spécifique "qr-yyy", l'interface « tap » est liée à Serveur DNS local et présent dans chaque réseau VLAN.

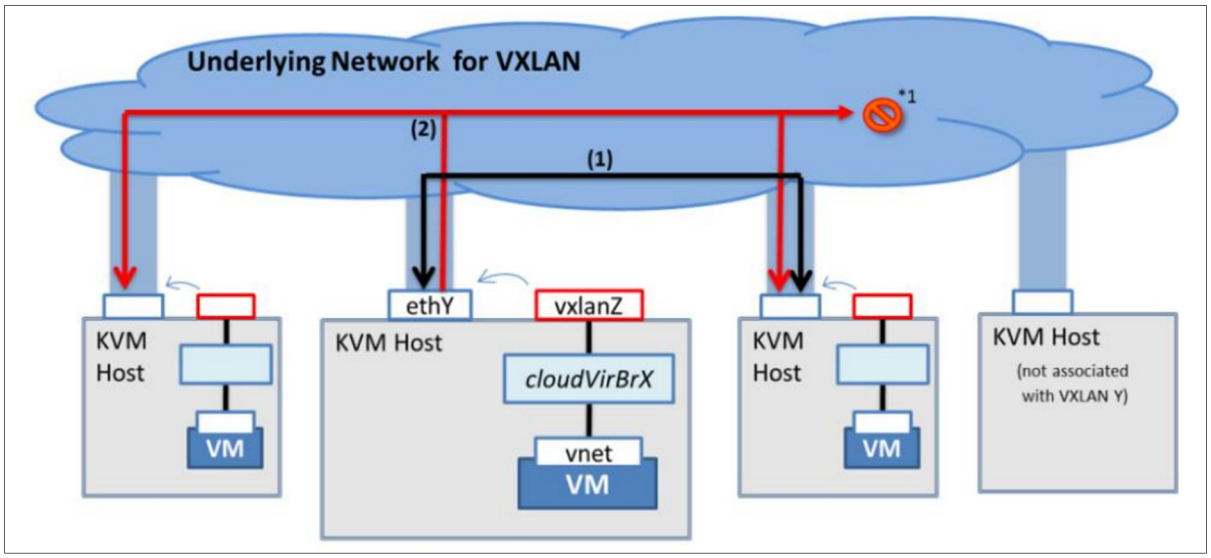

Figure 4.4 Tunnels VXLAN
Tirée de (Hatano, 2013)

La communication se base sur des tunnels VXLAN, ces derniers sont mis en place virtuellement. En effet, les ports de l'hôte jouent le rôle d'un commutateur virtuel en redirigeant le flux comme le montre la figure 4.4.

4.5 Description du scénario

Nous démontrons l'efficacité et la performance de notre solution à travers un scénario expérimental. L'infrastructure physique est décrite dans le tableau 4.2 ainsi que les tailles et les emplacements initiaux des VMs sont décrits dans le tableau 4.3.

Tableau 4.2 Caractéristiques des nœuds de calculs

	Mémoire	**CPU**	**Stockage**
Serveur 1	24 GB	16 (lent)	250 GB
Serveur 2	24 GB	16 (moyen)	250 GB
Serveur 3	64 GB	24 (rapide)	1 TB

Tableau 4.3 Distribution initiale des VMs

	Mémoire	**CPU**	**Stockage**	**Location**
VM 1	8GB	2	40GB	Server 2
VM 2	4GB	2	30GB	Server 3
VM 3	10GB	2	90GB	Server 1
VM 4	14GB	2	110GB	Server 2
VM 5	2GB	2	20GB	Server 1

Ce scénario consiste à effectuer un ensemble de relocalisations dynamiques qui minimisent le coût total du partage des ressources. La relocalisation peut être basée sur la migration à chaud, à froid, ou avec stockage. Dans chaque intervalle de 15 minutes, une nouvelle tentative de relocalisation est effectuée. Nous choisissons cet intervalle puisque, lors de nos expérimentations, les VMs ont montré des besoins presque stables pendant une durée inférieure à 15 minutes.

Le coût total d'hébergement d'une VM comprend les coûts de la mémoire allouée, de la CPU et du disque tels que définis dans le tableau 4.4. En outre, le coût de la CPU dépend de sa vitesse. Nous représentons donc trois classes différentes, des CPUs lents, moyens et rapides. Le coût de migration est calculé suivant le temps de migration et le coût du transfert de 1 Gigabyte de données (tableau 4.4). De plus, à travers le tableau 4.5, nous indiquons la taille de la bande passante existante entre les différents serveurs. Nous nous sommes basés sur le travail de (Grant et Eluwole, 2013) pour estimer ces coûts.

Tableau 4.4 Coûts opérationnels

	Coût opérationnel		
Mémoire **Stockage** **Transfert**	0.05 USD/1MB par heure 0.10 USD/1GB par stockage 0.10 USD/1GB par transfert		
CPU	Lent ≤ 100 MIPS	100 MIPS < Moyen < 80,000 MIPS	Rapide ≥ 80,000 MIPS
	0.025 USD par heure	0.8 USD par heure	1.2 USD par heure

Tableau 4.5 Taille de la bande passante entre les serveurs

	Serveur 1	**Serveur 2**	**Serveur 3**
Serveur 1	-	1G	10G
Serveur 2	1G	-	10G
Serveur 3	10G	10G	-

Comme le montre la figure 4.5, un ensemble de migrations est effectué dans chaque tentative selon un plan de migration spécifique. Plus d'une technique de migration peut être considérée en fonction de leur coût opérationnel. Sachant que les exigences de VM en termes de ressources physiques varient en fonction du temps, une prédiction est nécessaire afin d'éviter la surcharge des serveurs à l'avenir. Nous supposons qu'avant chaque nouvelle relocalisation, les VMs sont redimensionnées sur la base de leurs besoins prédits précédemment. En outre, lors de ce scénario, nous ne considérons pas la création de nouvelles VMs. Dans la figure 4.5, les VMs sont désignées par des cadres. Nous changeons leurs formes lorsqu'elles subissent un processus de redimensionnement.

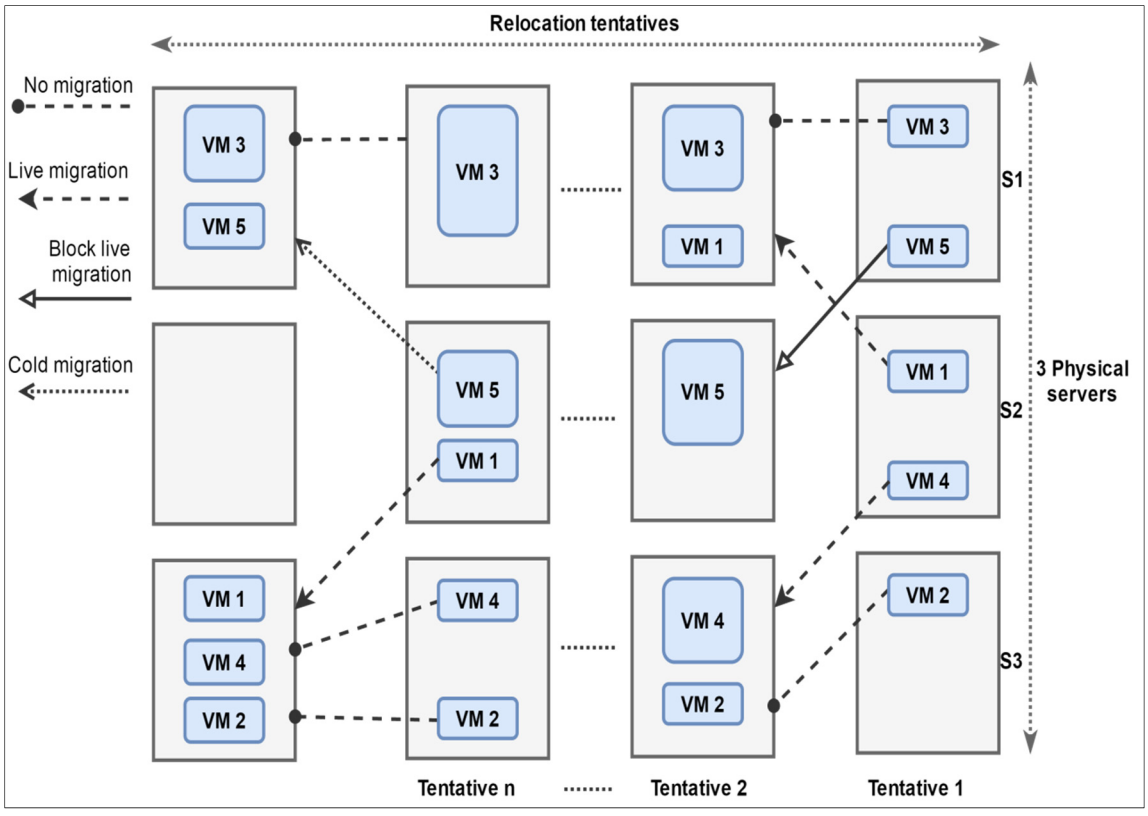

Figure 4.5 Scénario de partage des ressources physiques

4.6 Résultats et interprétation

Les résultats obtenus dans le cadre des expérimentations sont d'une grande importance à la validation de notre hypothèse de recherche. Pour ça, nous avons élaboré un scénario suite auquel nous obtenons des résultats. Dans cette section, nous présentons et interprétons les résultats de la prédiction, et ceux de la minimisation du coût opérationnel.

4.6.1 Prédiction

Comme le système considère les exigences courantes et futures des VMs pour effectuer la relocalisation la plus appropriée, nous présentons d'abord le choix des paramètres, ensuite, nous introduisons quelques résultats de prédiction de l'utilisation de la mémoire, de la CPU et du disque au long d'une période de trois heures.

4.6.1.1 Choix des paramètres

Les paramètres du modèle statistique reflètent le nombre d'observations et d'erreurs de prédiction à considérer lors de la construction du modèle (p, q). Le nombre de différentiations est également inclus dans ces paramètres, puisqu'il est primordial d'éliminer toute saisonnalité et tendance (trend), nous avons expliqué ceci dans la revue de littérature.

Afin de mieux expliquer le processus de sélection des paramètres en question, nous prenons l'exemple de l'utilisation de la mémoire par une VM quelconque. Chaque point de la figure 4.6 représente le taux d'utilisation de la mémoire, cette donnée est affichée d'une manière périodique (série temporelle).

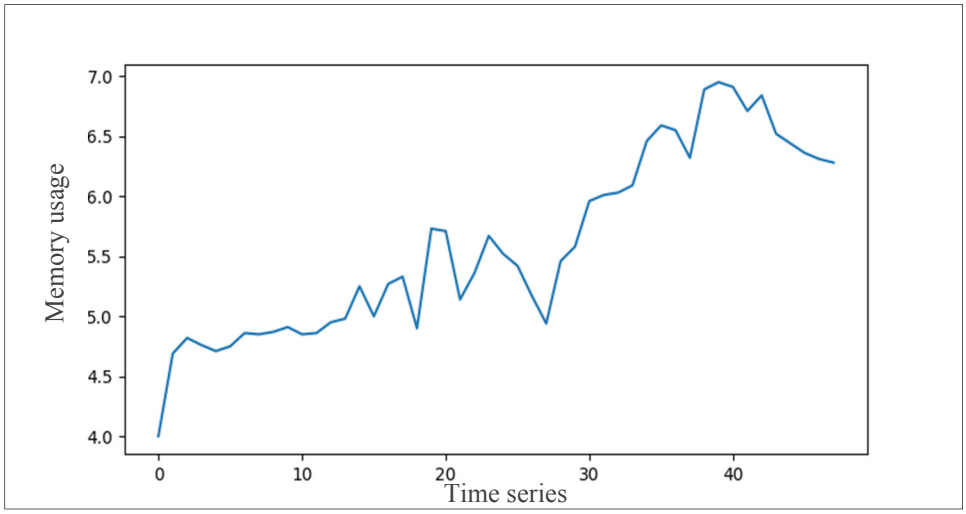

Figure 4.6 Visualisation des données relatives à l'utilisation de la mémoire

Il est clair d'après la figure ci-dessus que les données incluent de la tendance, d'où la nécessité d'appliquer le processus de différenciation. De ce fait, nous utilisons le test appelé « Dickey-Fuller ». Ce dernier permet de vérifier si les séries temporelles sont stationnaires ou non en posant une hypothèse nulle qui suppose que ces séries sont non-stationnaires. Dans notre cas, le résultat appuyait l'utilisation d'un seul cycle de différenciation ($d = 1$). La figure 4.7 montre l'allure de la courbe après l'application de ce processus, nous pouvons ainsi remarquer que la tendance a été presque éliminée.

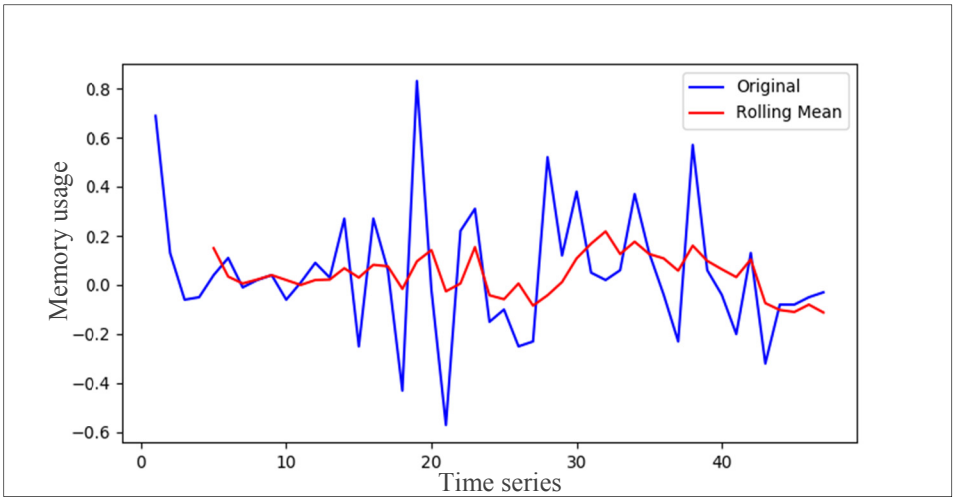

Figure 4.7 Données après différenciation

Afin de déterminer le couple (p, q), nous avons utilisé l'autocorrélation, ce qui nous a permis de mettre en évidence la corrélation entre les données. La figure 4.8, montre une corrélation positive entre les 16 premiers points (lags), qui est plutôt significative entre les 5 premiers. Nous choisissons de mener la prédiction en nous basant sur les deux premières données seulement ($p = 2, q = 2$) pour éviter un temps de traitement significatif.

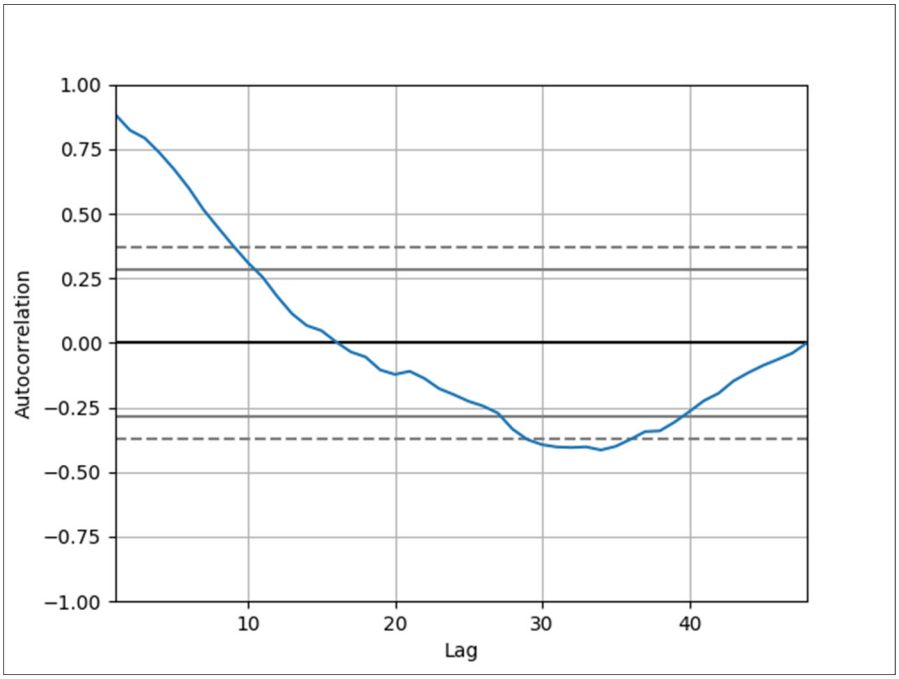

Figure 4.8 Autocorrélation entre les données

4.6.1.2 Future utilisation des ressources physiques

Dans le but de déterminer la future utilisation des ressources physiques, nous appliquons le modèle ARIMA, avec les paramètres déterminés au préalable, sur les données collectées (données mémoires, CPU et stockage). Une nouvelle prédiction est faite chaque intervalle de 15 minutes.

La figure 4.9 montre la prédiction de l'utilisation de la mémoire. La ligne continue représente les valeurs observées, tandis que la ligne pointillée représente les valeurs prédites. Dans le

cas de la mémoire, l'erreur de la racine-moyenne-carré (RMSE pour root-mean-square deviation) est égale à 0.322. Nous considérons cette valeur raisonnable vu que les valeurs prédites sont bien proches de celles observées. De même, la figure 4.10 illustre la prédiction de l'utilisation de la CPU avec une RMSE égale à 21.537. Étant donné que la charge traitée par un processeur varie substantiellement au cours du temps, il s'avère délicat de prédire la future utilisation de ce dernier et ainsi le futur besoin. Ceci explique l'erreur élevée en comparaison avec les autres prédictions. En ce qui concerne le stockage, il est plus facile de prédire sa future utilisation, en vue de sa stabilité d'un intervalle à un autre. En effet selon la figure 4.11, la prédiction s'affiche efficace avec une RMSE égale à 0.336. À travers les données collectées, nous pouvons continuer à fournir une prédiction précise, et ce en comparant les résultats obtenus avec ceux observés.

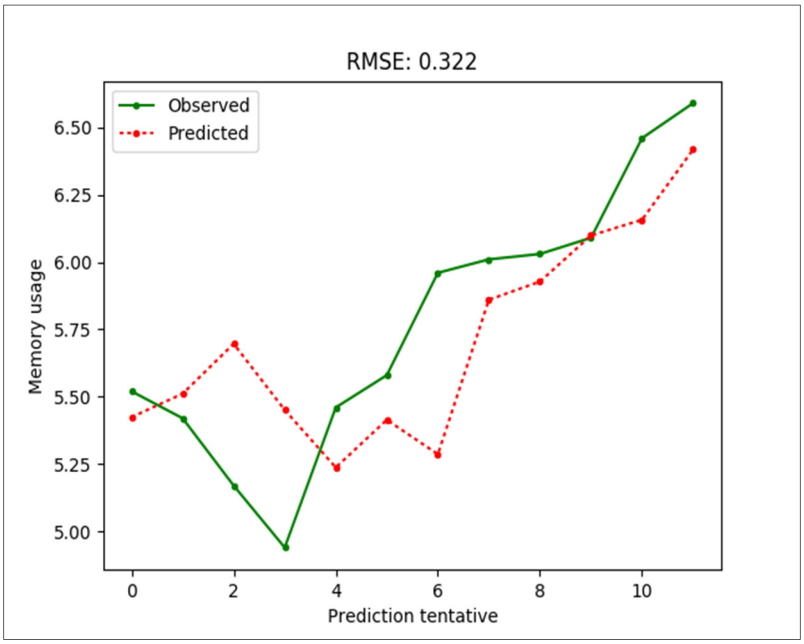

Figure 4.9 Prédiction de l'utilisation de la mémoire

Il est à noter que chaque tentative de prédiction correspond à une tentative de relocalisation (figure 4.12). En fait, avant qu'un nouveau plan de relocalisation ne soit fourni, les futurs besoins des ressources physiques, parmi toutes les VMs existantes, sont pris en considération.

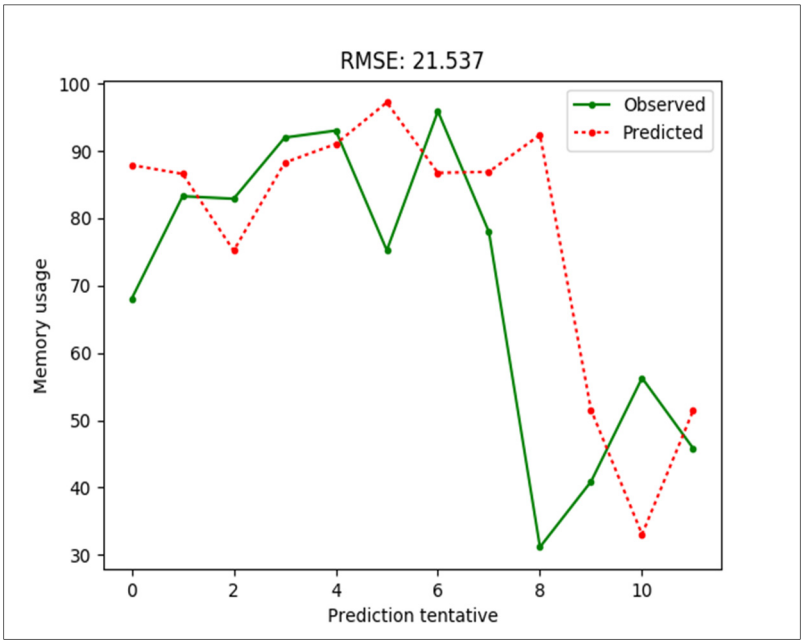

Figure 4.10 Prédiction de l'utilisation de CPU

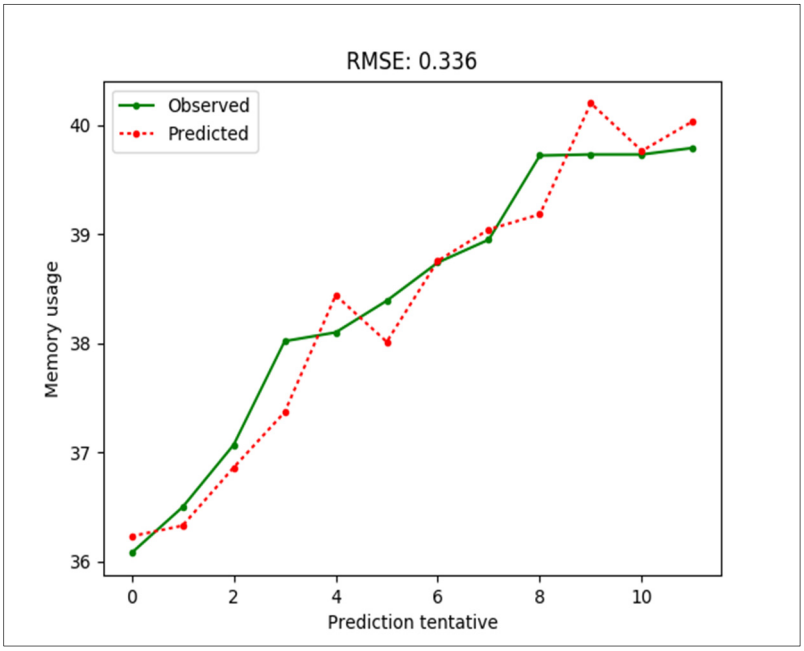

Figure 4.11 Prédiction de l'utilisation du stockage

4.6.2 Minimisation du coût opérationnel

Dans cette sous-section nous présentons les expérimentations menées suite à l'implémentation du module de prise de décision. Après le redimensionnement des VMs (ceci est fait manuellement à travers l'API nova), la relocalisation serait cruciale afin de trouver de meilleurs emplacements qui réduisent le coût courant. Nous finissons par une comparaison entre notre modèle d'optimisation et un autre suggéré dans le cadre d'un travail antérieur.

Chaque point des figures 4.12, 4.13 et 4.15 représente un coût additionnel qui s'applique lors de l'exécution d'un plan de relocalisation. Lorsque la relocalisation entraîne un coût égal à zéro, cela signifie que la relocalisation précédente reste la même et aucun coût supplémentaire ne sera appliqué au coût opérationnel du prochain intervalle. Cependant, lorsqu'il est inférieur à zéro, une nouvelle relocalisation qui minimise le coût par la valeur indiquée est suggérée. Par opposition à cela, lorsque le coût supplémentaire est supérieur à zéro, la nouvelle relocalisation augmente le coût associé au prochain intervalle. Cette situation se produit habituellement lorsque la relocalisation est utilisée pour éviter une future surcharge de serveurs au détriment du coût de partage, dans ce cas, certaines VMs doivent être migrées de manière obligatoire malgré les coûts conséquents.

4.6.2.1 Comparaison entre les différents types de migration

À travers cette comparaison, nous pouvons visualiser et analyser la différence entre les techniques de migration en termes de minimisation du coût opérationnel. Ceci est d'une grande importance puisque ça permet d'étudier le comportement du module décisionnel face aux différentes techniques, mais aussi de souligner l'avantage et l'inconvénient de chacune.

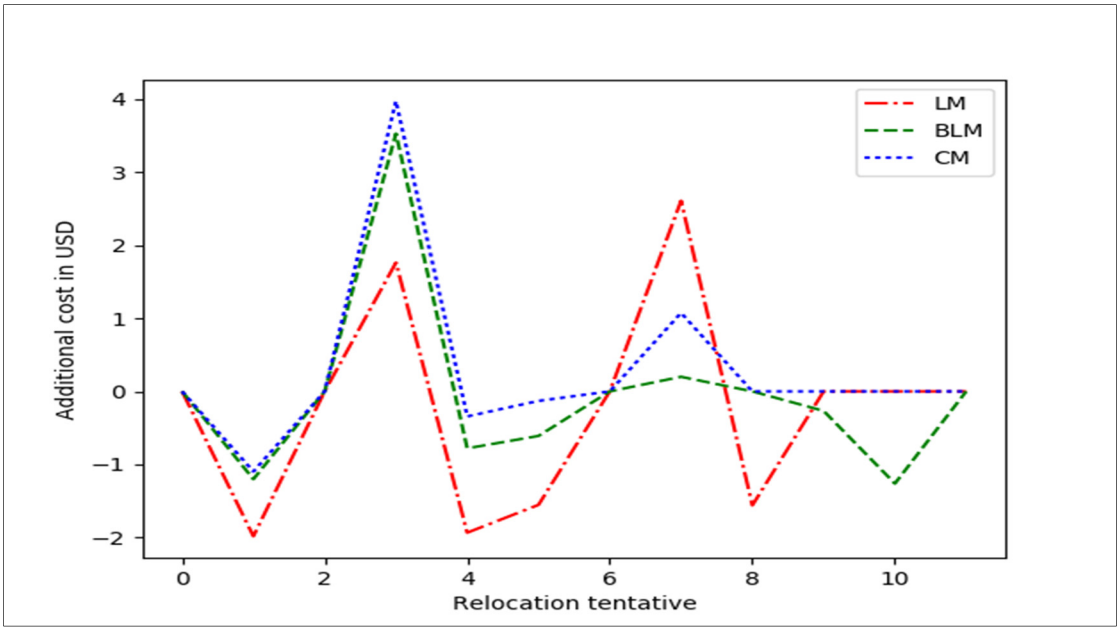

Figure 4.12 Minimisation du coût selon différentes techniques de migration

Les résultats de différentes techniques de migration sont présentés dans la figure 4.12. La migration à chaud LM (live migration) montre un meilleur coût opérationnel par rapport à la migration à froid CM (cold migration) et celle avec stockage BLM (block live migration). En effet, les plans de relocalisation proposés utilisant LM, aux tentatives 1, 3, 4, 5 et 8, offrent moins de coûts opérationnels. Cependant, son coût qui correspond à la tentative 7 dépasse celui offert par BLM et CM. Pour expliquer cela, concrètement, deux VMs ont été migrées à la place d'une autre VM (contrairement aux cas de BLM et CM) puisque son taux de pages modifiées (dirty pages) est supérieur au taux de transfert, donc afin de ne pas violer cette contrainte, le module de prise de décision choisit de ne pas la migrer (voir Annexe II). Ce dernier, se trouve, dans certain cas, sous l'obligation de migrer des VMs pour éviter la surcharge d'une ressource physique lorsqu'une ou plusieurs seront redimensionnées. BLM améliore le coût opérationnel par rapport au CM, car les temps d'arrêt dans le cas de CM sont généralement plus élevés, d'où le coût opérationnel. De plus, BLM offre le meilleur coût au niveau de la tentative 10 puisque la distribution des VMs n'est plus la même que dans CM et LM, la situation s'avère, en conséquence, différente. D'autre part, CM fait preuve de la minimisation la plus défavorable des coûts opérationnels en raison de son temps d'arrêt élevé

lorsqu'il s'agit d'un stockage volumineux. Néanmoins, cette technique peut être pertinente dans le cas d'un stockage à volume faible ou lorsque BLM n'est pas disponible.

4.6.2.2 Comparaison entre MIXT et les migrations individuelles

Nous comparons dans cette sous-section les résultats de MIXT (utilisation de différentes techniques de migration selon leur contribution à la minimisation du coût opérationnel) avec ceux de LM, BLM et CM. La figure 4.13 démontre que MIXT et LM ont un comportement étroit. Cependant, MIXT surpasse LM dans les tentatives 7, 9 et 10. En effet, cette dernière a utilisé la technique LM pour migrer les VMs jusque la tentative 6, ensuite, a adopté BLM pour le reste des migrations ce qui a entraîné un meilleur coût opérationnel. L'utilisation de BLM est justifiée par son coût réduit par rapport à LM durant ces tentatives. Comme nous l'avons précédemment expliqué, ceci revient à l'incapacité d'appliquer une migration à chaud à une VM ayant un taux de pages modifiées supérieur au taux de transfert, d'où, d'autres VMs sont migrées, ce qui peut entrainer une hausse de coût. En outre, étant donné que la méthode MIXT combine, dans ce scénario, à la fois LM et BLM, elle a nettement surpassé BLM et CM en termes de meilleur coût.

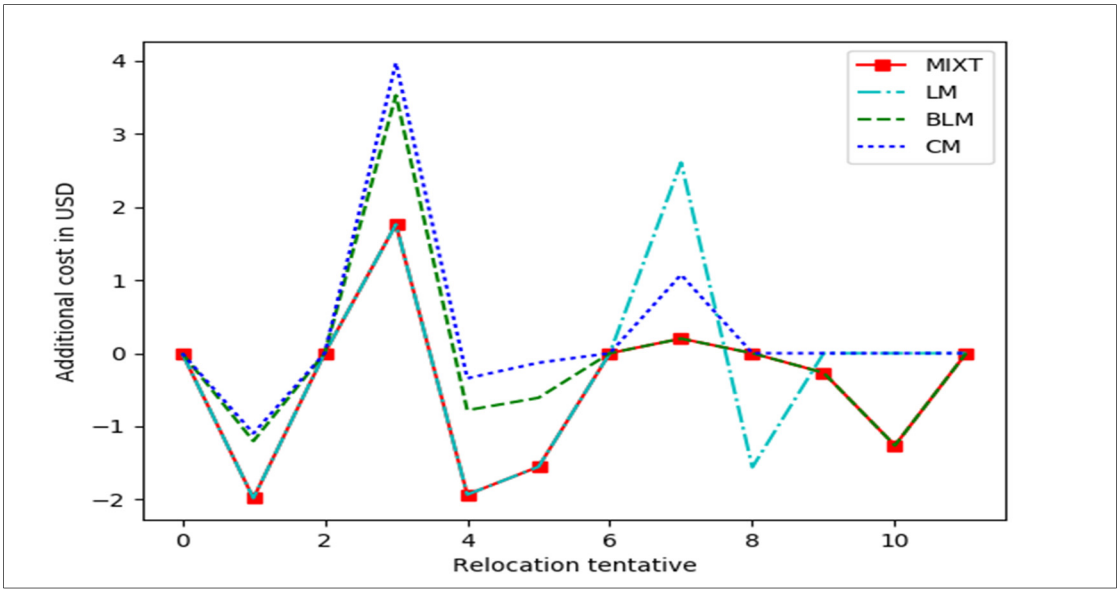

Figure 4.13 Minimisation du coût: MIXT vs LM

4.6.2.3 Modèle 1 vs Modèle 2

Nous comparons dans la figure 4.14 notre modèle d'optimisation lors de l'utilisation de la prédiction et sans en avoir recours. Nous avons enregistré le taux d'utilisation de la mémoire au niveau du serveur 1 après chaque redimensionnement des VMs (comme indiqué dans notre scénario) et lors de chaque tentative de relocalisation. En utilisant le modèle 1 (sans prédiction), nous obtenons des plans de relocalisation totalement différent de ceux fournis par le modèle 2 (avec prédiction), c'est ce qu'implique la différence au niveau de l'utilisation de la mémoire. En outre, les résultats démontrent qu'en omettant les prédictions de l'utilisation future des ressources physiques, nous risquons la surcharge des ressources et ce, en atteignant le seuil spécifié (dans notre travail, ce dernier est fixé à 90% d'utilisation) lors de la tentative 4 et 8. En utilisant la prédiction, le seuil n'est pas atteint, ceci s'explique par le fait que le modèle de prédiction permet d'anticiper toute relocalisation qui pose une menace de surcharge des serveurs et en conséquence l'éviter. À titre de clarification, à travers notre modèle 2, nous pouvons éviter d'atteindre le seuil, toutefois, nous ne parvenons pas forcément à des ressources moins chargées. Nous avons utilisé pour cette expérimentation la technique MIXT.

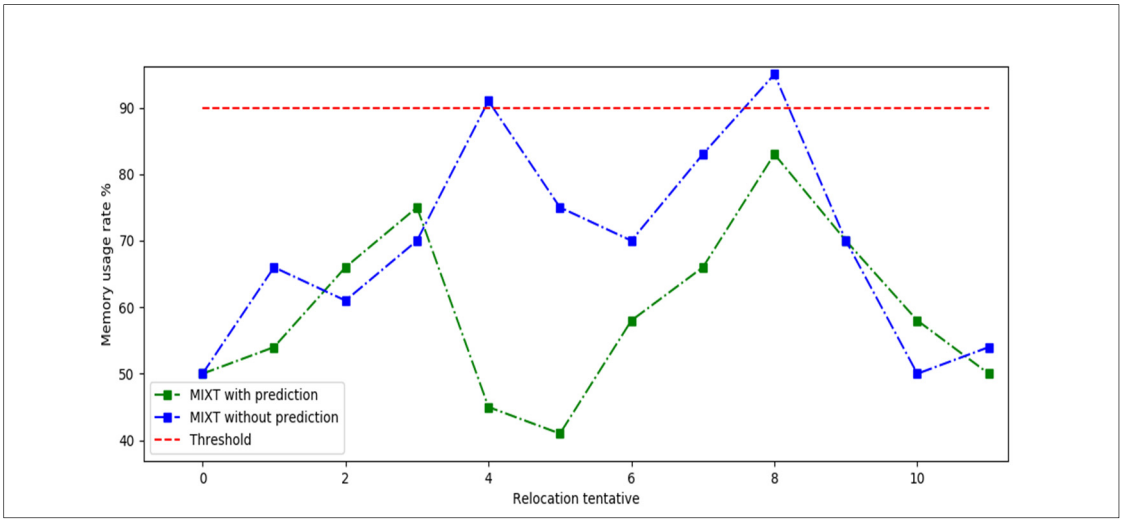

Figure 4.14 Utilisation de la mémoire avec/sans prédiction

4.6.2.4 MIXT vs MM

Nous avons présenté au niveau de la revue de littérature un travail qui a introduit une heuristique appelée MM (Beloglazov, Abawajy et Buyya, 2012). Cette dernière consiste à migrer le minimum de VMs à chaque fois qu'un seuil maximal d'utilisation de CPU est atteint. Un seuil minimal est fixé dans le but d'éviter la sous-utilisation des serveurs. Dans ce cas, toutes les VMs résidentes dans un serveur sous-utilisé sont migrées vers d'autres destinations. Comme le choix de la destination se fait sur la base d'une estimation de la puissance (destination qui contribue le moins à la hausse de la puissance), et afin d'adapter le travail proposé au nôtre, nous estimons plutôt le coût opérationnel et sélectionnons la destination impliquant le moindre coût. Dans le but de calculer le coût additionnel, nous calculons la différence entre le coût de la relocalisation précédente et celui de la relocalisation que nous venons d'obtenir, nous en rajoutons le coût de migration à chaud.

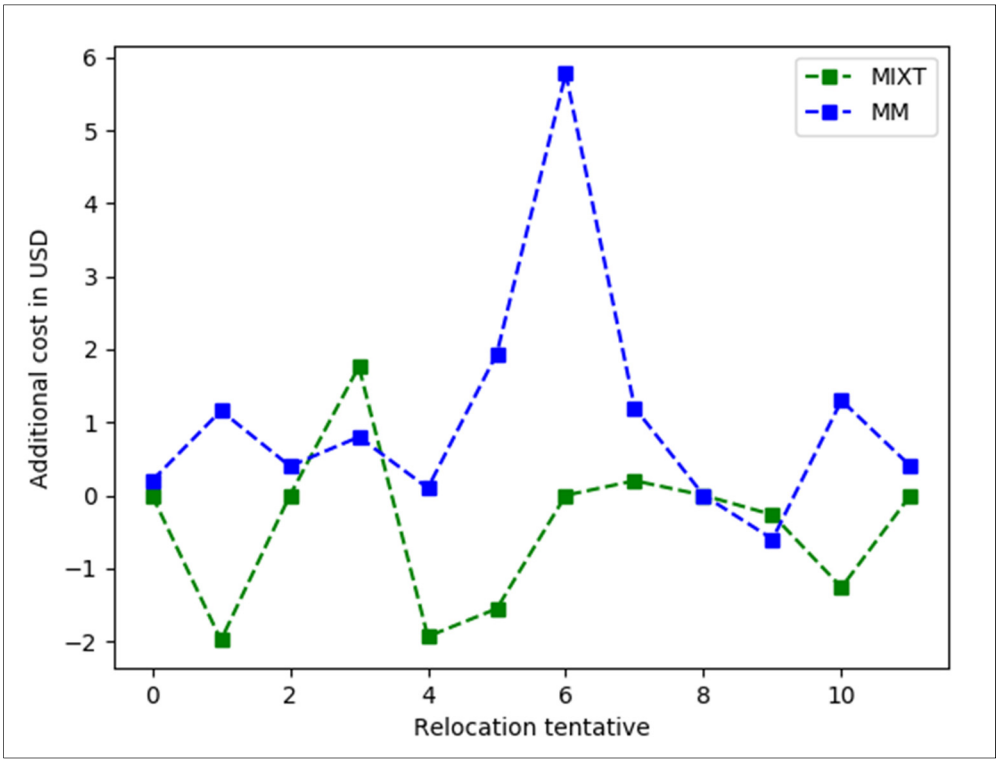

Figure 4.15 Minimization du coût opérationnel: MIXT vs MM

Les résultats présentés au niveau de la figure 4.15 montrent que notre modèle d'optimisation, dans la plupart des cas, offre un meilleur coût opérationnel. Nous expliquons ceci premièrement par le fait que notre modèle MIXT prend en considération le coût de migration pendant la prise de décision. Autrement si cette dernière est trop couteuse, le modèle peut maintenir la VM en question dans son ancien hôte. Ce n'est pas le cas pour MM, qui a recours à la migration indépendamment de son coût conséquent. De plus, considérer plus d'une technique de migration joue un rôle important dans la minimisation du coût, puisque, la migration à chaud peut s'avérer, dans certains cas, encore plus coûteuse que les autres (taux important des pages mémoire modifiées). Le modèle MM prend en compte seulement la migration à chaud. La prédiction implémentée dans MIXT offre également la possibilité d'éviter une éventuelle occasion de surcharger les ressources et dans certains cas de minimiser les migrations. Ceci impacte forcément le coût opérationnel en le réduisant.

4.7 Discussion

Dans cette section, nous discutons les résultats des expérimentations et ce, dans le but de valider notre hypothèse que nous avons présentée dans la section 1.5. Précédemment, nous avons introduit notre plan expérimental qui inclut des expérimentations effectuées sur la prédiction de l'utilisation des ressources physiques et sur la minimisation du coût opérationnel par le biais de différentes techniques de migration et les résultats de prédictions.

Au niveau de notre méthodologie, nous avons souligné un modèle de prédiction pour déterminer la future utilisation des ressources physiques. Dans ce cadre, nous avons effectué des tests pour prédire l'utilisation future de la mémoire, la CPU et le stockage. Pour ça, nous nous sommes appuyés sur le modèle statistique ARIMA dont nous l'avons adapté à nos données en fixant les paramètres convenables. Ce modèle nous a permis de prédire les valeurs futures que nous avions comparées avec celles observées (erreur). L'erreur RMSE est un élément essentiel à travers lequel nous pouvons valider la prédiction. Cette erreur est très réduite dans le cas de la mémoire et le stockage mais plus remarquable dans le cas de la CPU puisqu'il varie fréquemment au cours du temps, ainsi la prédiction devient encore plus

défiante. En général, les résultats montrent une prédiction que nous pouvons qualifier d'efficace.

Au niveau de la minimisation du coût opérationnel, nous avons enchainé une série d'expérimentations visant à montrer que notre modèle d'optimisation, proposé dans le cadre de la méthodologie, est capable de générer des plans de relocalisation qui parviennent à réduire d'une manière optimale le coût opérationnel et ce, en s'appuyant sur le modèle de prédiction ainsi qu'un ensemble de techniques de migration. D'abord, nous avons effectué des expérimentations visant à mettre en évidence la différence entre les diverses techniques de migration en termes de minimisation du coût opérationnel. Les résultats que nous avons obtenus montrent que la migration à chaud est la plus efficace mais présente une vulnérabilité majeure s'agissant de son incapacité à migrer la VM lorsque celle-ci a un taux de pages mémoires modifiées supérieur au taux de transfert, dans ce cas, les autres techniques de migration s'avèrent plus utiles. Ces résultats nous ont menés vers la conclusion que l'utilisation d'une seule technique de migration peut être une bonne option, cependant, la combinaison de ces dernières peut renforcer notre solution d'optimisation. En conséquence, nous avons mené une autre expérimentation qui vérifie l'effet de combiner les différentes techniques de migration (méthode appelée MIXT) sur la minimisation du coût opérationnel et ainsi sur le partage des ressources physiques. Les résultats ont montré clairement que MIXT fournit le coût le plus réduit, et en conséquence le partage le plus optimal des ressources physiques. D'une manière similaire, et afin de démontrer l'utilité de la prédiction dans notre système, nous avons implémenté et tester deux modèles, l'un assure la relocalisation en ayant recours aux résultats de prédiction (modèle 2) et l'autre, en les omettant (modèle 1). Nous avons remarqué que dans le cas du modèle 2, nous parvenons à éviter la surcharge des ressources, et en conséquence la dégradation des performances contrairement au cas du modèle 1. Nous avons comparé également MIXT avec une autre solution proposée dans la littérature appelée MM. À travers cette comparaison nous avons démontré l'utilité de la prise en compte du coût de migration ainsi que l'utilisation de différentes techniques de migration.

Ces résultats confirment notre hypothèse qui concerne l'optimisation du partage des ressources en termes de coût opérationnel lors de la mise en place d'un modèle qui génère des plans de relocalisation basés sur la prédiction et un ensemble de techniques de migration.

4.8 Conclusion

À travers le scénario élaboré, nous avons obtenu les résultats de prédiction et de minimisation du coût opérationnel. Au niveau de la prédiction, l'application du modèle ARIMA avec les paramètres ($p = 2, d = 1, q = 2$) nous a fourni clairement une prédiction précise qui est nettement proche des résultats observés. Son utilisation dans l'élaboration d'un plan de relocalisation a permis la distribution des ressources physiques aux VMs d'une manière à considérer leurs futurs besoins. Le module de prise de décision permet de générer un plan de relocalisation qui optimise le partage des ressources physiques à travers la minimisation du coût opérationnel. Les résultats ont montré que les plans générés sont efficaces et assurent la minimisation du coût dépendamment de la technique de relocalisation sans pour autant surcharger les ressources physiques. À la fin, nous avons discuté les résultats afin de valider notre hypothèse.

Dans ce qui suit, une conclusion générale pour conclure notre travail au complet et présenter les perspectives qui seront acheminées dans un temps futur.

CONCLUSION

Notre travail traite le problème de partage de l'infrastructure physique de point de vue coût opérationnel. Dans ce cadre, nous avons étudié les différentes techniques de migration ainsi que la mesure du coût opérationnel dans un environnement partagé. Nous nous sommes également focalisé sur les modèles statistiques utilisés généralement dans l'anticipation d'événements futurs, dans notre contexte, la future utilisation des ressources physiques par les VMs.

Notre travail vise à offrir un partage optimal et ce, en s'appuyant sur l'utilisation courante et future des ressources physiques et la mise en œuvre d'un ensemble de techniques de migration. À cet effet et dans le cadre de notre méthodologie, nous avons proposé un modèle composé d'un ensemble de modules appartenant à des catégories différentes suivant leur fonction. Nous avons conçu des modules pour la collecte des données, d'autres pour l'analyse et la prédiction de la future utilisation des ressources physiques en ayant recours au modèle statistique ARIMA ainsi qu'un module de prise de décision qui implémente notre fonction objective. Un module appelé noyau permet d'orchestrer les divers composants du système. Notre modèle permet de générer des plans de relocalisation qui minimisent le coût, ces plans, qui seront plus tard envoyés au migrateur, renseignent sur le nouvel emplacement de la VM et la technique de migration avec laquelle elle sera déplacée. Nous avons suggéré, à cet égard, deux modèles, l'un agit seulement sur la base de l'utilisation courante des ressources, et l'autre considère davantage la future utilisation. De plus, un algorithme qui concerne la prise de décision a été proposé.

Afin de valider notre modèle et tout en suivant un scénario bien précis, nous avons acheminé des expérimentations dans un environnement infonuagique déployé à l'aide d'Openstack. Ces expérimentations ont montré que le modèle proposé permet de minimiser le coût opérationnel d'une manière optimale tout en respectant les futurs besoins des VMs. Nous avons aussi démontré que l'utilisation de différentes techniques de migration est très efficace dans notre chemin vers un partage optimal.

Notre contribution se manifeste dans la construction et la mise en œuvre d'un système qui assure l'optimisation du partage de ressources en termes de coût opérationnel. Ce système s'appuie sur plus d'une technique de migration ainsi que l'utilisation courante et future des ressources physiques. Sur le plan scientifique et technologique, notre travail offre un nouveau terrain qui permet à la prédiction et aux différentes techniques de migration de résoudre les problèmes de partage de ressources dans le nuage. Sur le plan économique, ce dernier ouvrira les portes vers une meilleure utilisation du nuage informatique et par conséquent remplacer des services qui peuvent s'avérer couteux. Au niveau social, le travail améliore l'expérience utilisateur à travers la satisfaction des VMs (les services conséquemment), en termes de ressources physiques.

Étant donné que la recherche scientifique est en perpétuelle évolution, notre modèle peut lui-même être amélioré dans le futur. De ce fait, au niveau de la prédiction il serait plus intéressant de prédire bien plus de ressources, telles que la bande passante, en l'intégrant dans notre modèle sous forme de contrainte permettant d'étudier la future faisabilité d'une certaine migration de VM. Il serait avantageux de prendre en considération les types d'applications, de ce fait, nous évitons d'exécuter une migration inadéquate par rapport à une application. Ceci, peut être mis en œuvre à travers l'addition d'une contrainte au niveau de notre modèle. De plus, nous pouvons prendre en compte plusieurs routes pour acheminer le trafic de la migration. Nous envisageons également mettre à l'échelle notre solution. Pour cette fin, nous considérons l'application d'une heuristique puisque notre algorithme impliquera une grande complexité dans une telle situation.

Nous avons rédigé un article de journal intitulé « Optimized virtual machine migration schemes in cloud based on data prediction » sujet de publication à « IEEE transactions on computers ».

ANNEXE I

PLANS DE RELOCALISATION

Nous présentons dans cette annexe les différents plans de relocalisation générés suivant les techniques de migration:

Tableau A.I 1 Plans de relocalisation - migration à chaud

	Live migration				
	VM 1	**VM 2**	**VM 3**	**VM 4**	**VM 5**
0	Server 2	Server 3	Server 1	Server 2	Server 1
1	Server 2	**Server 1**	Server 1	Server 2	Server 1
2	Server 2	Server 1	Server 1	Server 2	Server 1
3	Server 2	Server 1	Server 1	**Server 3**	Server 1
4	**Server 3**	Server 1	Server 1	Server 3	**Server 2**
5	**Server 2**	Server 1	Server 1	Server 3	**Server 1**
6	Server 2	Server 1	Server 1	Server 3	Server 1
7	Server 2	**Server 3**	Server 1	Server 3	**Server 2**
8	Server 2	Server 3	Server 1	**Server 2**	Server 2
9	Server 2	Server 3	Server 1	Server 2	Server 2
10	Server 2	Server 3	Server 1	Server 2	Server 2
11	Server 2	Server 3	Server 1	Server 2	Server 2

Tableau A.I 2 Plans de relocalisation - migration à froid

	Cold migration				
	VM 1	**VM 2**	**VM 3**	**VM 4**	**VM 5**
0	Server 2	Server 3	Server 1	Server 2	Server 1
1	Server 2	**Server 1**	Server 1	Server 2	Server 1

Tableau A.I 2 Plans de relocalisation - migration à froid (suite)

	Cold migration				
	VM 1	**VM 2**	**VM 3**	**VM 4**	**VM 5**
2	Server 2	Server 1	Server 1	Server 2	Server 1
3	Server 2	Server 1	Server 1	**Server 3**	Server 1
4	**Server 3**	Server 1	Server 1	Server 3	**Server 2**
5	**Server 2**	Server 1	Server 1	Server 3	**Server 1**
6	Server 2	Server 1	Server 1	Server 3	Server 1
7	Server 2	Server 1	**Server 3**	Server 3	Server 1
8	Server 2	Server 1	Server 3	Server 3	Server 1
9	Server 2	Server 1	Server 3	Server 3	Server 1
10	Server 2	Server 1	Server 3	Server 3	Server 1
11	Server 2	Server 1	Server 3	Server 3	Server 1

Tableau A.I 3 Plans de relocalisation - migration à chaud avec stockage/MIXT

	Block live migration/MIXT				
	VM 1	**VM 2**	**VM 3**	**VM 4**	**VM 5**
0	Server 2	Server 3	Server 1	Server 2	Server 1
1	Server 2	**Server 1**	Server 1	Server 2	Server 1
2	Server 2	Server 1	Server 1	Server 2	Server 1
3	Server 2	Server 1	Server 1	**Server 3**	Server 1
4	**Server 3**	Server 1	Server 1	Server 3	**Server 2**
5	**Server 2**	Server 1	Server 1	Server 3	**Server 1**
6	Server 2	Server 1	Server 1	Server 3	Server 1
7	Server 2	Server 1	**Server 3**	Server 3	Server 1
8	Server 2	Server 1	Server 3	Server 3	Server 1
9	Server 2	Server 1	Server 3	**Server 1**	Server 1
10	Server 2	Server 1	Server 3	Server 1	**Server 2**

Tableau A.I 3 Plans de relocalisation - migration à chaud avec stockage/MIXT (suite)

	Block live migration/MIXT				
	VM 1	**VM 2**	**VM 3**	**VM 4**	**VM 5**
11	Server 2	Server 1	Server 3	Server 1	Server 2

BIBLIOGRAPHIE

Abbasov, B. 2014. « Cloud computing: State of the art reseach issues ». In *2014 IEEE 8th International Conference on Application of Information and Communication Technologies (AICT)*. (15-17 Oct. 2014), p. 1-4.

Abu, Sean. 2016. « Seasonal ARIMA with Python ». <http://www.seanabu.com/2016/03/22/time-series-seasonal-ARIMA-model-in-python/>. Consulté le 09-05-2016.

Alam, M. R., M. B. I. Reaz et M. A. M. Ali. 2012. « A Review of Smart Homes—Past, Present, and Future ». *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews),* vol. 42, nº 6, p. 1190-1203.

Armbrust, Michael, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica et Matei Zaharia. 2010. « A view of cloud computing ». *Commun. ACM,* vol. 53, nº 4, p. 50-58.

Arrivé-Cornec, Alexandre. 2014. « Comprendre la virtualisation des réseaux ». <https://www.virtual-sddc.ovh/comprendre-la-virtualisation-des-reseaux/>. Consulté le 25-10-2015.

Beloglazov, Anton, Jemal Abawajy et Rajkumar Buyya. 2012. « Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing ». *Future generation computer systems,* vol. 28, nº 5, p. 755-768.

Beloglazov, Anton, Sareh Fotuhi Piraghaj, Mohammed Alrokayan et Rajkumar Buyya. 2012. « Deploying OpenStack on CentOS using the KVM Hypervisor and GlusterFS distributed file system ». *Cloud Computing and Distributed Systems (CLOUDS) Laboratory Department of Computing and Information Systems, The University of Melbourne, Australia*.

Besson, Maxime. 2012. *Virtualisation et Cloud open source*.

Botta, A., W. de Donato, V. Persico et A. Pescapé. 2014. « On the Integration of Cloud Computing and Internet of Things ». In *2014 International Conference on Future Internet of Things and Cloud*. (27-29 Aug. 2014), p. 23-30.

Boutaba, Raouf, Lu Cheng et Qi Zhang. 2012. « On Cloud computational models and the heterogeneity challenge ». *Journal of Internet Services and Applications,* vol. 3, nº 1, p. 77-86.

Chatfield, Chris. 2016. *The analysis of time series: an introduction*. CRC press.

Daniels, Jeff. 2009. « Server virtualization architecture and implementation ». *Crossroads,* vol. 16, nº 1, p. 8-12.

Dinh, Hoang T, Chonho Lee, Dusit Niyato et Ping Wang. 2013. « A survey of mobile cloud computing: architecture, applications, and approaches ». *Wireless communications and mobile computing,* vol. 13, nº 18, p. 1587-1611.

Elprince, N. 2013. « Autonomous resource provision in virtual data centers ». In *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*. (27-31 May 2013), p. 1365-1371.

Grant, Aaron Blojay, et Opeoluwa Tosin Eluwole. 2013. « Cloud resource management—Virtual machines competing for limited resources ». In *ELMAR, 2013 55th International Symposium*. p. 269-274. IEEE.

Guérineau, Hugo. 2008. « La virtualisation de serveurs ». <http://www-igm.univ-mlv.fr/~dr/XPOSE2008/virtualisation/definitions.html>.

Habib, Irfan. 2008. « Virtualization with KVM ». *Linux J.,* vol. 2008, nº 166, p. 8.

Hatano, Toshiaki. 2013. « Linux native VXLAN support on KVM hypervisor ». <https://cwiki.apache.org/confluence/display/CLOUDSTACK/Linux+native+VXLAN+support+on+KVM+hypervisor>. Consulté le 25-01-2016.

He, S., L. Guo et Y. Guo. 2011. « Real Time Elastic Cloud Management for Limited Resources ». In *2011 IEEE 4th International Conference on Cloud Computing*. (4-9 July 2011), p. 622-629.

Jain, Aarshay. 2016. « A comprehensive beginner's guide to create a Time Series Forecast ». <https://www.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python/>. Consulté le 12-05-2016.

Jang, M., K. Schwan, K. Bhardwaj, A. Gavrilovska et A. Avasthi. 2014. « Personal clouds: Sharing and integrating networked resources to enhance end user experiences ». In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*. (April 27 2014-May 2 2014), p. 2220-2228.

Kashyap, R., S. Chaudhary et P. M. Jat. 2014. « Virtual machine migration for back-end mashup application deployed on OpenStack environment ». In *2014 International Conference on Parallel, Distributed and Grid Computing*. (11-13 Dec. 2014), p. 214-218.

Kumar, Rakesh, Neha Gupta, Shilpi Charu, Kanishk Jain et Sunil Kumar Jangir. 2014. « Open source solution for cloud computing platform using OpenStack ». *International Journal of Computer Science and Mobile Computing,* vol. 3, nº 5, p. 89-98.

Linusnova. 2013. « OpenStack : Le compute avec Nova » . <http://www.linusnova.com/2013/02/openstack-le-compute-avec-nova-2/ >. Consulté le 23-12-2015.

Lorido-Botran, Tania, Jose Miguel-Alonso et Jose A. Lozano. 2014. « A Review of Auto-scaling Techniques for Elastic Applications in Cloud Environments ». *Journal of Grid Computing,* vol. 12, nº 4, p. 559-592.

Manvi, Sunilkumar S., et Gopal Krishna Shyam. 2014. « Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey ». *Journal of Network and Computer Applications,* vol. 41, p. 424-440.

Mell, Peter, et Tim Grance. 2011. « The NIST definition of cloud computing ».

Moghaddam, F. F., et M. Cheriet. 2010. « Decreasing live virtual machine migration down-time using a memory page selection based on memory change PDF ». In *2010 International Conference on Networking, Sensing and Control (ICNSC)*. (10-12 April 2010), p. 355-359.

Moreno-Vozmediano, Rafael, Rubén S Montero et Ignacio M Llorente. 2012. « Iaas cloud architecture: From virtualized datacenters to federated cloud infrastructures ». *Computer,* vol. 45, nº 12, p. 65-72.

Moses, Jaideep, Ravi Iyer, Ramesh Illikkal, Sadagopan Srinivasan et Konstantinos Aisopos. 2011. « Shared resource monitoring and throughput optimization in cloud-computing datacenters ». In *Parallel & Distributed Processing Symposium (IPDPS), 2011 IEEE International*. p. 1024-1033. IEEE.

Pang, Z. 2010. *High Performance Live Migration Over Low-bandwidth, High-delay Network with Loss Prevention*. University of Alberta.

Pepple, Ken. 2011. *Deploying openstack*. " O'Reilly Media, Inc.".

Rosado, Tiago, et Jorge Bernardino. 2014. « An overview of openstack architecture ». In *Proceedings of the 18th International Database Engineering & Applications Symposium*. p. 366-367. ACM.

Sadiku, M. N. O., S. M. Musa et O. D. Momoh. 2014. « Cloud Computing: Opportunities and Challenges ». *IEEE Potentials,* vol. 33, nº 1, p. 34-36.

Sefraoui, Omar, Mohammed Aissaoui et Mohsine Eleuldj. 2012. « OpenStack: toward an open-source solution for cloud computing ». *International Journal of Computer Applications,* vol. 55, nº 3.

Semnanian, A. A., J. Pham, B. Englert et X. Wu. 2011. « Virtualization Technology and its Impact on Computer Hardware Architecture ». In *2011 Eighth International Conference on Information Technology: New Generations*. (11-13 April 2011), p. 719-724.

Sv, Petter, #228, rd, Benoit Hudzia, Johan Tordsson et Erik Elmroth. 2011. « Evaluation of delta compression techniques for efficient live migration of large virtual machines ». *SIGPLAN Not.,* vol. 46, nº 7, p. 111-120.

Svärd, Petter, Benoit Hudzia, Johan Tordsson et Erik Elmroth. 2011. « Evaluation of delta compression techniques for efficient live migration of large virtual machines ». *ACM Sigplan Notices,* vol. 46, nº 7, p. 111-120.

Trezeciak, Ralf. 2016. « OpenStack Liberty Neutron Deployment (Part 1 Overview) ». <http://www.opencloudblog.com/?p=557>. Consulté le 27-01-2016.

Vakilinia, Shahin, Dongyu Qiu et Mustafa Mehmet Ali. 2014. « Optimal multi-dimensional dynamic resource allocation in mobile cloud computing ». *EURASIP Journal on Wireless Communications and Networking,* vol. 2014, nº 1, p. 1-14.

Voorsluys, William, James Broberg, Srikumar Venugopal et Rajkumar Buyya. 2009. « Cost of virtual machine live migration in clouds: A performance evaluation ». In *IEEE International Conference on Cloud Computing*. p. 254-265. Springer.

Wang, M., X. Meng et L. Zhang. 2011. « Consolidating virtual machines with dynamic bandwidth demand in data centers ». In *2011 Proceedings IEEE INFOCOM*. (10-15 April 2011), p. 71-75.

Wuhib, Fetahi, Rolf Stadler et Hans Lindgren. 2012. « Dynamic resource allocation with management objectives—Implementation for an OpenStack cloud ». In *Network and service management (cnsm), 2012 8th international conference and 2012 workshop on systems virtualiztion management (svm)*. p. 309-315. IEEE.

Xiao, Z., W. Song et Q. Chen. 2013. « Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment ». *IEEE Transactions on Parallel and Distributed Systems,* vol. 24, nº 6, p. 1107-1117.

Zhang, Q., Q. Zhu, M. F. Zhani, R. Boutaba et J. L. Hellerstein. 2013. « Dynamic Service Placement in Geographically Distributed Clouds ». *IEEE Journal on Selected Areas in Communications,* vol. 31, nº 12, p. 762-772.