جامعة سيدي محمد بن عبد الله بفاس
UNIVERSITE SIDI MOHAMED BEN ABDALLAH DE FES
كلية العلوم ظهر المهراز فاس
FACULTE DES SCIENCES DHAR EL MEHRAZ FES



N° d'Ordre :**03/2021**

Formation Doctorale : Sciences e t Technologies de l ' In formation e t de l a Communication (STIC)

Discipline : I nformatique

Spécialité : I nformatique

Laboratoire : Laboratoire d ' I nformatique, Signaux, Aut omatique e t Cognitivisme (LISAC)

THESE De DOCTORAT

Prés ent ée par

ELMOTAKISaloua

Prédiction de l'Efficacité Énergétique dans les Clouds Datacenters basée sur le Machine Learning et les Algorithmes Bio-inspirés

Soutenue le 16/ 01/ 2021 devant le jury composé de :

Pr. TAIRI Hamid	***Faculté des Sciences Dhar El Mahraz, Fès***	***Président***
Pr. BELLACH Benaissa	***ENSA, Université Mohammed Premier, Oujda***	***Rapporteur***
Pr. SABER Mohammed	***ENSA, Université Mohammed Premier, Oujda***	***Rapporteur***
Pr. SABRI My Abdelouahed	***Faculté des Sciences Dhar El Mahraz, Fès***	***Rapporteur***
Pr. GUALOUS Hamid	***Université de Caen Normandie, France***	**Examinateur**
Pr. EL BEQQALI Omar	***Faculté des Sciences Dhar El Mahraz, Fès***	**Examinateur**
Pr. ALJ Abdelkamel	***FSJES, Université Moulay Ismail, Meknès***	**Examinateur**
Pr. MAHRAZ Mohamed Adnane	***Faculté des Sciences Dhar El Mahraz, Fès***	**Examinateur**
Pr. YAHYAOUY Ali	***Faculté des Sciences Dhar El Mahraz, Fès***	**Directeur de thèse**

Année universitaire : 2020-2021

a

Remerciements

Je remercie tout d'abord Dieu tout puissant de m'avoir donné le courage, la force et la patience d'achever ce modeste travail.

Tout particulièrement, j'aimerais exprimer ma sincère gratitude à mon Directeur de thèse, le Pr. ***Ali YAHYAOUY****, pour ses conseils judicieux, ses avis fiables et son indéfectible soutien. Sa remarquable intelligence dans l'orientation du déroulement de la recherche a réduit la complexité de la description des défis auxquels je faisais face. Je tiens à remercier sincèrement mon codirecteur, Pr.* ***Hamid GUALOUS****, pour son suivi attentif, ses précieux conseils, ainsi que son accueil au sein du laboratoire LUSAC à l'université de Caen Normandie, France. Je remercie également toute l'équipe du laboratoire LISAC de la Faculté des Sciences Dhar El Mahraz. J'adresse mes remerciements au Pr.* ***Ahmet ARSLAN*** *de la Faculté d'Ingénierie de Eskişehir Technical University, pour toutes les discussions intéressantes que j'ai eues avec lui durant la période de mobilité dans le cadre du programme Erasmus+. Mes remerciements vont également au Pr.* ***Redouane MERROUCH*** *pour m'avoir permis de mener un stage de recherche dans le Cluster de calcul du Moroccan Academic and Research Wide Area Network (MARWAN) au sein du Centre National de la Recherche Scientifique et Technique (CNRST).*

Un grand merci pour mes collègues du laboratoire LISAC et du laboratoire LUSAC pour les discussions scientifiques qui m'étaient utiles pour mettre à jour mes connaissances scientifiques et techniques.

Je tiens à remercier Pr. ***Hamid TAIRI*** *pour l'honneur qu'il me fait d'être le président de jury. Je remercie également tous les membres de jury : Pr.* ***Benaissa BELLACH****, Pr.* ***Mohammed SABER*** *et Pr.* ***My Abdelouahed SABRI*** *comme rapporteurs, ainsi que Pr.* ***Hamid GUALOUS****, Pr.* ***Omar EL BEQQALI****, Pr.* ***Abdelkamel ALJ*** *et Pr.* ***Mohamed Adnane MAHRAZ*** *comme examinateurs. Je leur fait part de ma gratitude pour le temps consacré à l'évaluation de notre travail. Ils m'honorent par leur présence et je leur suis très reconnaissante pour l'intérêt qu'ils vont porter à nos travaux de recherches à travers leurs remarques.*

Enfin et surtout, je tiens à exprimer mes plus profonds remerciements à ma famille, qui m'a cru et soutenu pendant cette longue période d'étude. Je crois que l'éducation et les nombreuses sacrifices qu'ils m'ont données sont les choses les plus précieuses que je n'ai jamais reçues.

Résumé

Avec l'essor de la technologie infonuagique (*Cloud Computing*), les centres de données sont aujourd'hui censés d'occuper une place importante dans le scénario Internet. Malgré cette pertinence, ils sont probablement encore loin de leur véritable zénith en raison de la demande toujours croissante de contenus à stocker et à distribuer dans le Cloud, du besoin de puissance de calcul ou des quantités de plus en plus importantes de données analysées par les grandes entreprises telles que *Google*, *Microsoft* ou *Amazon*. Cependant, avoir un centre de données implique deux problèmes majeurs : ils sont terriblement coûteux à construire, et ils consomment une énorme quantité d'énergie, donc trop chers à entretenir. C'est pourquoi l'augmentation de l'efficacité énergétique (et donc la réduction de l'empreinte de carbone) des centres de données a été l'un des sujets de recherche les plus intéressants au cours des dernières années.

Pour augmenter convenablement l'efficacité énergétique d'un centre de données, nous devons comprendre comment l'énergie est consommée et établir des techniques pour bien gérer le fonctionnement du système informatique. Nous présentons, d'abord, une fonction de corrélation entre le service rendu par un serveur dans un centre de données et la consommation d'énergie de ce serveur en se basant sur le processeur, vu que ce dernier est considéré comme le composant le plus consommateur d'énergie. Ensuite, nous présentons deux différentes stratégies de consolidation des machines virtuelles et des machines physiques dans un centre de données. La première stratégie consiste en un processus de placement statique des machines virtuelles qui prend en compte la dimension énergétique ainsi que la qualité de service rendu aux utilisateurs. La deuxième stratégie de consolidation proposée est la migration en temps réel des machines virtuelles. En effet, afin de respecter les accords au niveau de service, on propose un modèle de migration dynamique permettant de prédire une variété de métriques, et par la suite aider à sélectionner automatiquement l'algorithme de migration le plus convenable pour la situation d'exécution donnée. Enfin, les modèles d'estimation et de gestion des ressources dans un centre de données sont efficaces dans le cas où le système fonctionne correctement ; cependant, vue la grande taille et la diversité des composants dans cette structure, une certaine probabilité d'occurrence d'anomalies peut se produire ; ainsi, on propose un algorithme d'apprentissage non supervisé pour identifier le comportement anormal des machines virtuelles lors de l'exécution de charges de travail dans un système en fonction des détails de l'utilisation des ressources.

Mots clés : ***Cloud Computing*** **; centre de données ; efficacité énergétique ; consolidation ; détection d'anomalies ; modèle stochastique ; Machine Learning ; métaheuristique.**

Abstract

With the rapid development of Cloud Computing technology, data centers are currently supposed to occupy an important place in the Internet scenario. Despite such relevance, they are probably still far from their true zenith due to the ever-increasing pressure for content to be stored and distributed in the Cloud, the need for computing capacity or the rising amounts of data being analyzed by large enterprises such as *Google*, *Microsoft* or *Amazon*. Nevertheless, operating a data center involves two major problems : they are terribly expensive to construct, and they consume an enormous amount of energy, therefore too expensive to maintain. As a result, increasing energy efficiency (thereby reducing the carbon footprint) of data centers has been one of the most interesting areas of research over recent years.

To ensure the energy efficiency of a data center, we need to understand how energy is consumed and establish techniques to effectively manage the functioning of the IT system. First, we present a correlation function between the service provided by a server in a data center and the energy consumption of that server based on the CPU, since it is considered to be the most energy-consuming component. Second, we present two different strategies for consolidating virtual machines and physical machines in a data center. The first strategy consists of an energy-based static placement process for virtual machines that takes into account the quality of service provided to users. The second strategy proposed for consolidation is the live migration of virtual machines. To meet service level agreements, a dynamic migration model is proposed to predict a variety of metrics, and then help to automatically select the most suitable migration algorithm for the given execution situation. Finally, resource estimation and management models in a data center are efficient when the system is running properly ; however, due to the large size and diversity of components in the data center, a certain probability of anomalies may occur ; thus, an unsupervised learning algorithm is proposed to identify the abnormal behaviour of virtual machines when running workloads in a system depending on the details of resource utilization.

Keywords : **Cloud Computing ; data center ; energy efficiency ; consolidation ; anomaly detection ; stochastic model ; Machine Learning ; metaheuristic.**

Table des matières

1 Introduction générale **1**
1.1 Problématique et objectifs de recherche . . . 3
1.2 Méthodologie de recherche . . . 7
1.3 Contributions et diffusion scientifique . . . 8
1.4 Organisation de la thèse . . . 9

Contexte **12**

2 Fondements de la thèse **12**
2.1 Introduction . . . 13
2.1.1 La virtualisation . . . 14
2.1.2 Le *Cloud Computing* . . . 17
2.1.3 Le centre de données . . . 19
2.2 Techniques de réduction de la consommation énergétique dans un centre de données . . . 20
2.2.1 Placement des VMs . . . 21
2.2.2 Migration en temps réel des VMs . . . 23
2.2.2.1 Algorithmes de migration en temps réel . . . 25
2.2.2.2 Métriques de migration en temps réel . . . 26
2.2.3 Détection des anomalies du système . . . 27
2.2.3.1 Types d'anomalies de performance . . . 28
2.2.3.2 Approches de détection d'anomalies . . . 29
2.3 Méthodes, approches et outils utilisés . . . 30
2.3.1 Méthodes utilisées pour le placement des VMs . . . 30
2.3.1.1 Méthodes exactes . . . 31
2.3.1.2 Méthodes approximatives . . . 33
2.3.2 Méthodes utilisées pour la migration en temps réel des VMs . . . 35
2.3.2.1 Méthodes de régression linéaire . . . 36
2.3.2.2 Méthodes de régression non linéaire . . . 38
2.3.3 Méthodes utilisées pour la détection des anomalies . . . 40
2.3.3.1 *Fuzzy C-means* pondérée . . . 41
2.3.3.2 Le comportement collectif des étourneaux . . . 42

2.3.4 Outil de simulation 43
2.4 Conclusion 45

3 Revue critique de la littérature 47
3.1 Introduction 48
3.2 Modèles d'estimation de la consommation énergétique 49
3.2.1 Modèles additifs 49
3.2.2 Modèles BA 50
3.2.3 Autres modèles 54
3.2.4 Comparaison des modèles de consommation électrique des serveurs 55
3.3 Modèles de placement des VMs 56
3.3.1 Placement statique des VMs 56
3.3.2 Placement dynamique des VMs 58
3.3.3 Algorithmes et méthodes de résolution du placement des VMs . . . 60
3.3.3.1 Placement basé sur l'optimisation mathématique 61
3.3.3.2 Placement basé sur des algorithmes métaheuristiques . . . 63
3.4 Modèles de migration en temps réel des VMs 65
3.4.1 L'origine de la migration en temps réel 65
3.4.2 *Machine Learning* pour la migration en temps réel des VMs 66
3.4.2.1 Modèles basés sur LR 67
3.4.2.2 Modèles basés sur SVM 69
3.4.2.3 Modèles basés sur KNN 70
3.4.2.4 Modèles basés sur ANN 71
3.4.2.5 Modèles basés sur RL 72
3.4.2.6 Modèles basés sur diverses approches statistiques 73
3.5 Modèles de détection des anomalies de performance 75
3.5.1 Modèles basés sur l'apprentissage supervisé 75
3.5.2 Modèles basés sur l'apprentissage non supervisé 76
3.6 Conclusion 78

Contributions 81

4 Corrélation entre la consommation énergétique et le service rendu par le centre de données 81
4.1 Introduction 82
4.2 Modélisation de la corrélation entre la consommation énergétique et la QoS 85
4.2.1 Contexte expérimental 87
4.2.2 Modèle stochastique 88

4.2.3 Estimation des fonctions de répartition 89
4.3 Résultats expérimentaux 91
4.3.1 Relation entre $\boldsymbol{w}$ et $\boldsymbol{p}$ 92
4.3.2 Estimation de la puissance $\boldsymbol{p}$ pour un serveur 95
4.3.3 Estimation de la puissance $\boldsymbol{p}$ pour plusieurs serveurs 96
4.4 Conclusion 96

5 Placement des VMs et efficacité énergétique dans le centre de données 101
5.1 Introduction 102
5.2 Modélisation du placement des VMs comme un problème de sac à dos . . . 105
5.2.1 Résolution par séparation et évaluation progressive 106
5.2.2 Résolution par la programmation dynamique 107
5.2.3 Résolution par algorithme génétique 109
5.2.4 Résolution par optimisation de colonie de fourmis 111
5.2.5 Analyse théorique de la complexité des algorithmes 113
5.3 Résultats expérimentaux et analyse de performance 115
5.3.1 Nombre de machines virtuelles 116
5.3.2 Taux de convergence 116
5.3.3 Utilisation de CPU 118
5.3.4 Temps d'exécution 118
5.4 Consommation énergétique 120
5.5 Conclusion 123

6 Migration en temps réel des VMs et SLAs dans un centre de données 126
6.1 Introduction 127
6.2 Modélisation de la migration en temps réel des VMs 130
6.2.1 Paramètres du modèle 131
6.2.2 Génération du modèle 133
6.3 Résultats expérimentaux et évaluation 135
6.3.1 Précision de la prédiction de la métrique cible 135
6.3.2 Exhaustivité des caractéristiques considérées 138
6.3.3 Aperçu sur la durée d'entraînement et de test 139
6.4 Scénario de migration en temps réel 140
6.5 Conclusion 143

7 Le *Clustering* pour détecter les anomalies dans le centre de données 144
7.1 Introduction 145
7.2 L'algorithme de *Clustering* proposé (sbWFCM) 147

7.3 Résultats expérimentaux et évaluation 149
7.3.1 Instructions d'implémentation 151
7.3.2 Résultats du *Clustering* et discussion 152
7.4 Conclusion 154

8 Conclusion Générale **156**
8.1 Apports de cette thèse 156
8.2 Limitations et futures recommandations 158
8.3 Remarque finale 159

Bibliographie **160**

A Outils d'évaluation **II**
A.1 MAE & MRE II
A.2 PDCVM & SVTAH II
A.3 Validation du *Clustering* III

Table des figures

1 Services fournis par le Cloud. 2
2 Consommation énergétique potentielle des centres de données entre 2010 et 2030 [187]. 4
3 Structure organisationnelle de la thèse. 10

4 Classification des plates-formes de virtualisation. 15
5 Vue d'ensemble des techniques adressées dans cette thèse. 21
6 Le mécanisme de placement des VMs. 22
7 Aperçu des processus de migration *Pre-Copy* et *Post-Copy*. 25
8 Exemples d'anomalies de performance [24]. 29
9 Différence entre la programmation dynamique et la méthode diviser pour régner. 33
10 Représentation (a) du croisement et (b) de la mutation dans l'AG 34
11 Un exemple simplifié du comportement collectif des étourneaux. 43
12 Représentation de l'architecture de *CloudSim* [43]. 45

13 Répartition de la puissance entre les composants des serveurs pour différentes configurations. 51
14 La répartition de la consommation de l'énergie d'un serveur [148]. 52
15 Le bilan statistique des différentes variables utilisées pour la modélisation de la consommation énergétique d'un serveur. 56

16 Principales sources de perte de puissance dans un centre de données traditionnel [139]. 82
17 Relation entre l'utilisation du CPU, la puissance consommée par le processeur et la puissance totale utilisée par le serveur. 86
18 La relation un-à-un entre $\boldsymbol{p}$ et $\boldsymbol{w}$ et la nature monotone des fonctions de répartition pour déterminer une relation quantitative entre les deux variables aléatoires. 89
19 Approximation de la fonction de distribution de $\boldsymbol{p}$. 93
20 La fonction de densité estimée théoriquement et la fonction de densité obtenue expérimentalement pour une puissance normalement distribuées. . 95

21 La consommation de puissance actuelle et estimée du serveur IBM X3550 à 2 processeurs XeonX5675 lors de l'exécution du benchmark *SpecPower*. . 97
22 La consommation de puissance actuelle et estimée du serveur IBM X3250 à 4 processeurs XeonX3480 lors de l'exécution du benchmark *SpecPower*. . 98
23 La consommation de puissance totale actuelle et estimée pour plusieurs serveurs. 99

24 Exemple d'un chromosome (sac à dos). 110
25 Évolution du taux d'utilisation du CPU indiqué par les quatre algorithmes dans les trois situations étudiées (20, 50, 100 VMs). 117
26 Le taux de convergence pour les quatre algorithmes afin de fournir une solution pour un nombre de VMs égal à 100. 117
27 Le taux de ressources CPU utilisé par les quatre algorithmes pour fournir une solution avec un nombre de VMs égal à (a) 20, (b) 50, et (c) 100. . . . 119
28 L'évolution du temps d'exécution des quatre algorithmes étudiés pour différents nombres de VMs. 119
29 Comparaison de la consommation totale de puissance de notre modèle (basé sur l'OCF) et du modèle standard (basé sur FFD). 123

30 Vue d'ensemble du modèle introduit. 131
31 Évolution de l'indice *R-squared* en fonction du paramètre Alpha (a)(b) et du paramètre K (c). La troisième dimension représente les métriques étudiées. 134
32 L'erreur moyenne absolue (MAE) de modèle de régression appliqué aux six métriques cibles avec les cinq algorithmes de migration en temps réel. . . . 136
33 L'erreur moyenne relative (MRE) du modèle de régression appliqué aux six métriques cibles avec les cinq algorithmes de migration en temps réel. . . . 137
34 Un aperçu du modèle incorporé dans le processus de migration en temps réel des VMs. 141
35 Taux de violation des SLAs 142
36 Nombre et nature des violations pour chaque politique, en considérant la SVR avec *Bagging* et la KNNR. 142

37 Une description visuelle des comportements élémentaires dans le modèle de simulation de *Raynolds*. 148
38 L'évolution du XBI et le nombre correspondant de clusters en fonction du nombre d'itérations pour une exécution typique de l'algorithme sbWFCM. 152
39 Projection *Sammom* de l'espace caractéristique résultant dans le plan pour le *Clustering* des charges de travail en utilisant l'algorithme sbWFCM. . . 153

40 *Boxplots* montrant les distributions de l'ARI sur 20 exécutions indépendantes de sbWFCM et de WFCM simples basées sur la densité. 154
41 Projection *Sammom* de l'espace caractéristique résultant dans le plan pour le *Clustering* des charges de travail en utilisant l'algorithme standard WFCM. 155

Liste des tableaux

2.1 Les phases usuelles du processus de migration en temps réel. 24

5.1 Les caractéristiques des PMs. 103
5.2 Nombre de CPU requis pour chaque VM. 103
5.3 Concepts de base de l'AG pour résoudre le problème de sac à dos. 110
5.4 Estimation de la complexité des algorithmes utilisés. 114
5.5 Les paramètres principaux de l'algorithme AG et OCF. 115
5.6 Nombre de VMs et taux de CPU correspondant utilisé, donné par l'exécution de chaque algorithme pour un nombre différent de VMs à placer dans un serveur avec différentes capacités. 116
5.7 synthèse de la comparaison entre les quatre algorithmes pour un nombre de VMs égal à 50. 121
5.8 Les tests menés 122
5.9 La consommation de puissance et le temps d'exécution pour le modèle proposé et un modèle standard basé sur FFD. 122
5.10 L'amélioration du pourcentage de notre modèle par rapport au modèle existant en termes de consommation totale d'énergie pour les 5 tests. . . . 123

6.1 Les caractéristiques d'entrée du modèle de prédiction. 132
6.2 *R-squared* agrégé des métriques étudiées. 139
6.3 Aperçu sur la durée d'entraînement et de test. 140
6.4 Politiques définissant les contraintes de SLAs 140

7.1 La description et la formulation mathématique des trois comportements élémentaires, la séparation, l'alignement et la cohésion. 148
7.2 Les statuts définis d'une VM. 151
7.3 Spécifications des paramètres avec les valeurs par défaut. 151

Liste des abbréviations

AG Algorithme génétique.

CPU Unité centrale de traitement (*Central Processing Unit*).

DLTC Compression Delta (*Delta Compression*).

DTC Compression de données (*Data Compression*).

IaaS *Infrastructure as a service* .

KNNR K plus proches voisins (*K-Nearest Neighbors*).

LASSO *Least Absolute Shrinkage and Selection Operation.*

OCF Optimisation de colonie de fourmis.

PaaS *Plateform as a service.*

PD Programmation dynamique.

PM Machine physique (*Physical Machine*).

QoS Qualité de service (*Quality of Service*).

SaaS *Service as a service* .

sbWFCM *Weighted Fuzzy C-Means with starling birds*).

SLA Accord au niveau de service (*Service Level Agreement*).

SVR Régression à vecteur support (*Support Vector Regression*).

VM Machine virtuelle (*Virtual Machine*).

WFCM Fuzzy c-means pondéré (*Weighted Fuzzy c-means*).

1 Introduction générale

L'évolution des technologies de l'information et de la communication (TIC) a donné naissance au paradigme du *Cloud Computing*, un modèle qui permet une allocation flexible, rentable et évolutive des ressources tout en offrant une facilité d'utilisation aux utilisateurs finaux. La croissance du besoin de services en Cloud a entraîné une augmentation rapide du nombre de ressources de calcul et du nombre de centres de données à différentes échelles.

Selon un rapport de *Research&Markets*[1], le marché mondial des services de *Cloud Computing* public est censé de passer de 371.4 milliards de dollars en 2020 à 832.1 milliards de dollars en 2025, avec un taux de croissance annuel composé, *Compound Annual Growth Rate* (CAGR), de 17.5%. La raison de succès du *Cloud Computing* est qu'il permet aux utilisateurs d'accéder à des ressources informatiques dans une boîte noire sans avoir les connaissances d'expert nécessaires pour mettre en place et maintenir toute l'infrastructure sous-jacente. Les utilisateurs peuvent donc se concentrer sur leur propre domaine de travail, ce qui permet à de nombreuses institutions de recherche, à des entreprises existantes et à de jeunes pousses d'innover et de résoudre des problèmes dans tous les domaines du travail humain.

Les avantages de l'utilisation des services dans le Cloud ont de plus en plus encouragé les clients à les adopter. La granularité du service offert varie de faible à élevée (voir la figure 1) et est souvent classée en trois modèles de prestation de services dans le Cloud :

***Infrastructure-as-a-Service* (IaaS)** : permet aux clients d'exécuter et de déployer des applications logicielles arbitraires sur un ensemble d'infrastructures, y compris des ressources informatiques fondamentales telles que le stockage et les machines virtuelles (VMs). Dans ce modèle, le client du Cloud a un contrôle total à la fois sur l'application logicielle et sur l'infrastructure qui héberge cette application. Un exemple marquant de la solution IaaS est *Amazon Elastic Compute Cloud (EC2)*[2].

***Plateform-as-a-Service* (PaaS)**. propose aux clients d'utiliser un environnement d'hébergement, comprenant les langages de programmation, les bibliothèques et les outils,

1. Research&Markets : https://www.globenewswire.com/
2. Amazon EC2 : http://aws.amazon.com/ec2

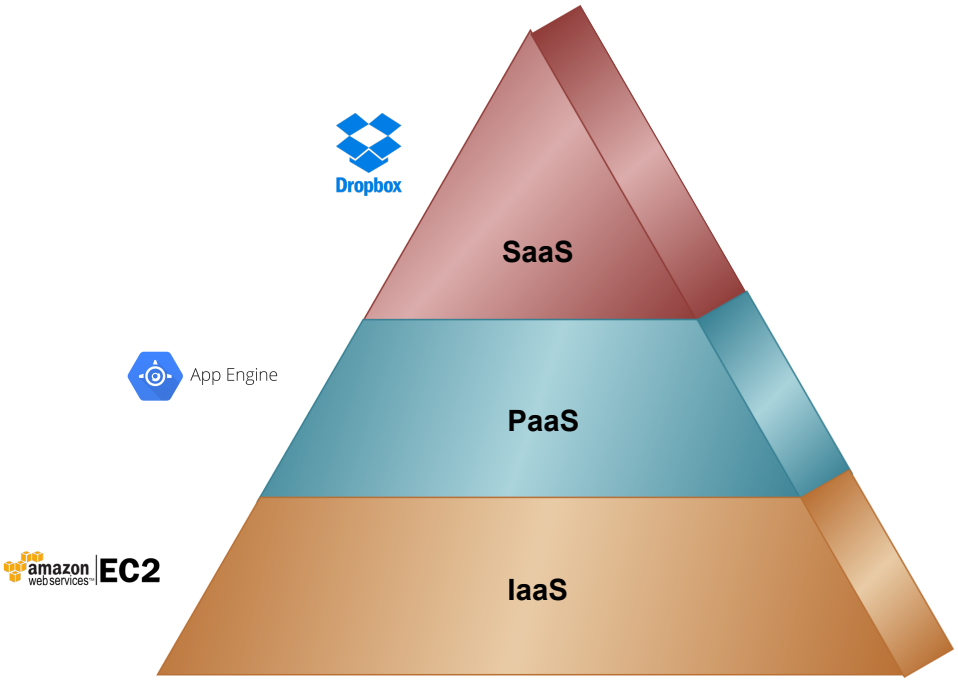


FIGURE 1 – Services fournis par le Cloud.

pour leurs applications. Bien que les clients aient la possibilité de contrôler leur application, ils ont un contrôle limité du système d'exploitation et de l'infrastructure. Un exemple intéressant d'offre PaaS est *Google App Engine*[3].

***Software-as-a-Service* (SaaS)**. propose des applications logicielles fonctionnant sur des infrastructures en Cloud. Les clients peuvent uniquement utiliser l'application et ils sont totalement limités dans le contrôle de l'infrastructure et de l'application. Des exemples notables de ce modèle sont les services de stockage tels que *Dropbox*[4], ou les services de streaming tels que *Netflix*[5].

Dans cette thèse, nous nous focalisons sur le IaaS. Comme le rapporte [211], le IaaS est le secteur du marché des services en Cloud qui connaît la plus forte croissance depuis plusieurs années. Pour décrire en détail le modèle de Cloud public IaaS, il est très utile d'examiner les différentes composantes nécessaires à un tel service. Le fournisseur de services, c'est-à-dire le fournisseur de Cloud, possède un ou plusieurs centres de données comprenant des milliers de machines physiques (PMs) virtualisées. En utilisant la technologie de virtualisation, chaque PM peut héberger plusieurs VMs qui sont isolées les unes des autres et qui ont accès à une certaine quantité de ressources physiques (CPU, mémoire, stockage, bande passante du réseau). Les VMs peuvent être transférées d'une PM à une autre via le réseau. En fait, la VM est un produit IaaS qui est proposé comme

3. Google App Engine : http://developers.google.com/appengine
4. Dropbox : http://www.dropbox.com
5. Netflix : http://www.netflix.com

ressource de calcul aux clients pour une utilisation sur Internet.

L'objectif principal de la recherche menée dans le cadre de cette thèse est d'étudier l'exploitation des méthodes mathématiques, du *Machine Learning* et des algorithmes inspirés de la nature pour relever les différents défis posés par les techniques énergiquement efficaces de gestion des ressources dans les centres de données ; notamment, les techniques de consolidation dynamique des VMs, où une entité centrale de gestion des VMs s'appuie sur une stratégie de placement ou de migration des VMs pour allouer ou réallouer efficacement les ressources afin de maintenir un équilibre entre la consommation d'énergie du centre de données (serveurs) et les performances des applications s'exécutant dans le système.

1.1 Problématique et objectifs de recherche

Les centres de données peuvent comprendre des milliers d'unités de serveurs qui occupent beaucoup d'espace et consomment une quantité considérable d'énergie. Un seul centre de données peut occuper $10.000.0m^2$ d'espace, et comprendre 12.000 armoires de serveurs. Selon une estimation du *Natural Resource Defense Council*, 12.000.000 de serveurs de centres de données permettraient de fournir la quasi-totalité des activités en ligne des États-Unis en 2014 [199] (avec une consommation qui passe de $338Twh$ à la fin de 2016 à $628Twh$ en 2021). De même, en France, environ 200 grands centres de données fonctionnent 24/24, avec une consommation de ressources permanente.

En conséquence, la consommation d'énergie liée aux centres de données pourrait également augmenter de manière significative. Selon une étude de [187], la proportion d'électricité consommée par les centres de données dans le monde passera de $1,3\%$ en 2010 à 8% en 2021, ainsi qu'ils sont susceptibles d'utiliser environ 3 à 13% de l'électricité mondiale en 2030. De même, les émissions annuelles de carbone sont prévues d'augmenter de 116 millions de tonnes en 2007 à 257 millions de tonnes, et pourraient atteindre ou dépasser celles du secteur aérien en 2020 [216]. Ainsi, les raisons pour lesquelles la mise en place des centres de données écologiques soit l'une des principales cibles de la communauté de la recherche pendant ces dernières années sont évidentes. En fait, l'écologisation des centres de données a deux objectifs : réduire les coûts d'exploitation et améliorer l'efficacité énergétique, c'est-à-dire diminuer la consommation d'énergie ainsi que les émissions de carbone qui leurs sont attribuées.

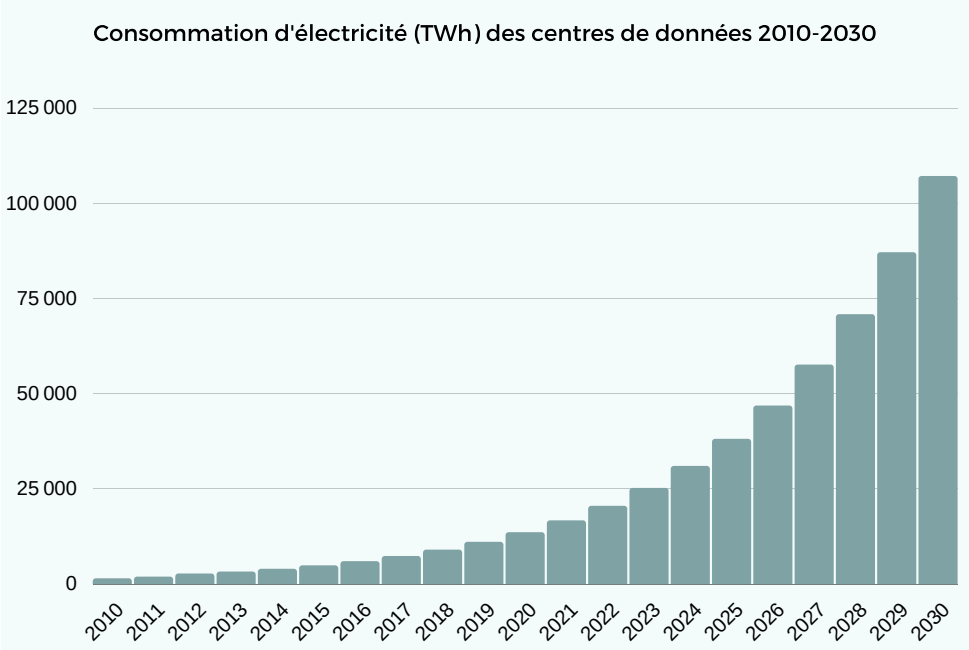


FIGURE 2 – Consommation énergétique potentielle des centres de données entre 2010 et 2030 [187].

Les approches existantes ou proposées pour atteindre l'efficacité énergétique englobent des solutions matérielles aussi bien que logicielles. L'examen des solutions matérielles dépasse le cadre de cette thèse. Les solutions logicielles, cependant, peuvent être classées en trois grandes catégories : l'ajustement dynamique (*Scaling*) [74], la planification (*Scheduling*) [145] et la consolidation (*Consolidation*) [192]. Les approches d'ajustement dynamique concernent essentiellement le processeur d'un serveur et visent à adapter sa tension et sa fréquence de calcul au taux d'arrivée des tâches au processeur. Cette approche réduit la consommation d'énergie du processeur en mode de veille et oblige le planificateur des tâches du système d'exploitation à éviter le surprovisionnement des ressources et à exécuter les tâches moins urgentes à une vitesse plus basse. Les approches de planification peuvent être de bas ou de haut niveau. Les approches de bas niveau se réfèrent à des planificateurs sensibles à l'énergie au sein du système d'exploitation, tandis que les approches de haut niveau se réfèrent à des approches d'équilibrage de charge de travail (*Workload Management*) au niveau de l'application. Les planificateurs orientés énergie visent à répartir les tâches (*Jobs*) de manière à utiliser efficacement les ressources matérielles et à éviter les conflits indésirables entre les tâches lors de cette utilisation (telles que la mémoire ou le CPU). Les solutions de répartition de la charge au niveau de la couche applicative sont pertinentes pour un environnement en cluster (tel qu'un centre de données) où plusieurs serveurs sont utilisés pour traiter simultanément un grand nombre

de requêtes émises par les utilisateurs. Ainsi, un outil de répartition de la charge énergétique vise à sélectionner un certain nombre de serveurs susceptibles de traiter la charge de travail du cluster entier sans violer les accords de niveau de service (SLAs). Notons que les SLAs désignent un contrat qui définit la qualité de service exigée entre un fournisseur de services unique ou multiple et un client ou un utilisateur de services unique ou multiple, qui peut être un individu, un département, une institution, etc. Les SLAs peuvent inclure la fiabilité, la sécurité, le débit et le temps d'exécution, etc. La consolidation concerne en particulier les environnements virtualisés. Comme pour les approches de planification, l'objectif est d'équilibrer la demande avec la disponibilité des ressources de calcul, mais cela est principalement réalisé par une migration des VMs (qui encapsulent les services ou les applications) vers des serveurs qui peuvent fonctionner à leur capacité maximale, au lieu de serveurs surchargés, sous-utilisés ou inactifs. Ensuite, les serveurs inactifs ou inutilisés sont éteints et ceux qui sont surchargés peuvent être libérés partiellement de leur charge.

Toutes les approches, indiquées ci-dessus, nécessitent une estimation de la charge de travail à court et moyen terme d'un serveur ou d'un centre de données, ainsi que de la consommation d'énergie correspondante qui peut être économisée grâce à une stratégie d'adaptation qu'elles visent à mettre en œuvre. Comme l'énergie peut être exprimée en fonction de la puissance consommée multipliée par le temps d'exécution, il est souvent suffisant d'estimer la puissance consommée par le système. Par conséquent, ces approches requièrent des mécanismes permettant de quantifier la consommation de l'énergie du système : Les planificateurs orientés énergie nécessitent des modèles pour évaluer la consommation électrique des serveurs ; les approches visant à la consolidation optimale des services exigent des modèles pour mesurer le coût de fonctionnement et de migration des VMs ; et les stratégies d'ajustement dynamique de la charge nécessitent des modèles pour mesurer la consommation électrique des différents serveurs.

En outre, les techniques de gestion des ressources dans le centre de données restent inefficaces dans le cas où il y a des variations inconnues ou inattendues dans le système. En effet, l'implication d'un environnement aussi complexe est que l'apparition de problèmes de performance tels que la dégradation (par exemple, la latence de réponse des pages) et les temps d'arrêt du service (par exemple, les points terminaux de service injoignables) est devenue une norme plutôt qu'une exception, avec de graves conséquences financières et opérationnelles.

L'objectif de cette thèse est de proposer des solutions pour améliorer l'efficacité énergétique des centres de données sans réduire les performances des applications et des services hébergés sur ces sites. En résumé, les problèmes de recherche suivants sont explorés :

- **Comment les différents aspects de la gestion des ressources affectent-ils la consommation électrique des serveurs ?**

  La compréhension de l'effet des différents paramètres sur la consommation de l'énergie est essentielle pour gérer les serveurs de manière efficace sur le plan énergétique. Ces paramètres peuvent comprendre la variation du nombre des VMs, la variation des ressources de calcul allouées à une VM (comme les cœurs des processeurs et la mémoire), la mise en service et l'arrêt des processeurs, la variation de la fréquence des processeurs et le réajustement des processeurs virtuels aux cœurs des processeurs physiques.

- **Comment établir une correspondance entre l'utilisation des ressources et la consommation énergétique ?**

  Il est important de convertir l'allocation des ressources du système informatique, telle que l'unité centrale de traitement, *Central Processing Unit* (CPU), à l'utilisation de la puissance propre au système afin de le diriger vers un état optimal. La complexité des systèmes dans le Cloud rend la modélisation des performances et de la puissance relativement compliquée.

- **Comment prévoir des ressources pour un rendement optimal et une consommation d'énergie minimale ?**

  La conception d'un système qui améliore l'efficacité énergétique par une allocation intelligente des ressources aux VMs tout en maintenant les exigences de qualité de service (QoS) pour les applications est un problème crucial. Un tel système doit prendre en considération de nombreux facteurs pour décider des ressources à allouer à une VM, comme la sensibilité de cette dernière aux changements de ressources, la dynamique de la charge de travail et les capacités de traitement des ressources physiques sous-jacentes. En outre, le système doit résoudre les éventuels conflits de ressources et les interférences de performance.

- **Comment pouvons-nous élaborer une stratégie de consolidation dynamique des VMs dans les centres de données afin de maintenir un équilibre optimal entre énergie et performance ?**

  La consolidation des VMs dynamiques est l'une des stratégies les plus proposées par les chercheurs, qui peut être utilisée pour rendre la consommation d'énergie efficace dans les centres de données en Cloud. Ces stratégies reposent sur un certain nombre d'algorithmes de gestion des ressources visant à trouver le bon équilibre entre la consommation d'énergie et les violations des SLAs en utilisant des techniques de migration en temps réel dans les centres de données virtualisés. De nombreuses approches de migration en temps réel ont été introduites ; chacune a souvent des politiques et des caractéristiques particulières, ainsi que des performances distinctes

et des effets secondaires tels que la consommation d'énergie et la dégradation des performances. Ainsi, l'une des questions clés auxquelles sont confrontés de nombreux fournisseurs de services est de savoir comment décider de l'algorithme de migration en temps réel le plus optimal dans certaines situations.

- **Comment pouvons-nous construire un modèle capable de prévenir autant que possible l'apparition de comportements anormaux du système ?**
  Afin d'atténuer efficacement le problème de la variation des performances, les administrateurs de système doivent détecter et identifier les anomalies responsables de cette variation et prendre des mesures préventives. Cependant, le diagnostic des anomalies est souvent une tâche difficile étant donné la variété et la grande quantité de données recueillies.
- **Proposer des méthodes de consolidation et de gestion des ressources efficaces sur le plan énergétique :**
  Pour aborder ces problématiques, nous avons défini les objectifs de recherche suivants :
  - ➔ Concevoir un modèle d'estimation de la puissance pour déterminer dynamiquement la relation entre l'utilisation des ressources, qui correspond à la QoS, et la consommation d'énergie au niveau des applications et des VMs.
  - ➔ Formuler le problème de placement des VMs comme un problème d'optimisation combinatoire en utilisant des informations provenant des VMs et des PMs créés à l'avance et développer une solution algorithmique pour résoudre le problème formulé.
  - ➔ Concevoir un modèle de migration en temps réel en utilisant de nouveaux concepts permettant d'automatiser le processus de migration tout en garantissant l'optimisation énergétique et la QoS.
  - ➔ Proposer un algorithme d'apprentissage non supervisé permettant de détecter et identifier les charges de travail et les VMs ayant un comportement anormal, et qui risque de produire des variations de performance, et par la suite, dans certain cas, de graves pannes dans le système.

1.2 Méthodologie de recherche

Notre travail utilise principalement la méthode de recherche constructive (CR) [204], une méthodologie de recherche très courante dans le domaine de la technologie informatique. L'utilisation de la méthode CR entraîne généralement un artefact ou une construction qui résout un problème pratique et spécifique à un domaine tout en produisant une

contribution théorique de valeur académique. La construction peut être une théorie, un algorithme, un modèle, un logiciel ou un framework. Conformément aux lignes directrices générales pour mener la recherche à l'aide de la méthode CR, nous avons effectué les étapes suivantes. Tout d'abord, nous avons identifié les problèmes de recherche présentant un intérêt pratique, comme décrit dans la section 1.1. Ensuite, nous avons effectué un bilan systématique de la littérature afin d'acquérir une compréhension suffisante du problème de recherche et du domaine. Nous avons ensuite conçu et développé des solutions au problème, les artefacts résultants étant des algorithmes, des modèles et des techniques pour améliorer l'efficacité énergétique dans les centres de données, dont les contributions scientifiques sont décrites dans la section 1.3. Pour démontrer la faisabilité de nos solutions et leurs contributions pratiques, et pour valider et relier les résultats au problème de recherche, nous avons évalué les contributions proposées de manière expérimentale et les avons comparées à quelques solutions existantes.

1.3 Contributions et diffusion scientifique

Pour atteindre les objectifs énumérés ci-dessus, cette thèse apporte quatre contributions principales. Un bref aperçu de ces contributions est présenté ci-après :

- **Un modèle stochastique pour estimer la consommation d'énergie par un serveur ;**

  La première contribution est une méthodologie pour caractériser la consommation énergétique d'un serveur. Nous fournissons de nouvelles observations sur le comportement de consommation énergétique des composants les plus pertinents du serveur, et nous proposons une technique précise pour estimer la consommation énergétique. Ainsi, nous modélisons la charge de travail et la consommation d'énergie d'un processeur comme des variables aléatoires et nous exploitons la propriété de monotonie de leurs fonctions de distribution pour établir une relation quantitative entre ces variables aléatoires. Nous démontrons la plausibilité de notre approche en estimant la consommation d'énergie des benchmarks fournis par le projet *CloudSim*. Ce modèle est utilisé pour évaluer le modèle de placement des VMs proposé.

- **Un modèle de placement statique des VMs suivant une nouvelle stratégie ;**

  La deuxième contribution de cette thèse consiste en une nouvelle approche de placement statique des VMs pour minimiser la consommation totale d'énergie d'un centre de données. Le problème de placement des VMs est formulé sous forme de problème de sac à dos qui est considéré comme un problème d'optimisation combinatoire sous contrainte. Comme ce problème est typiquement un problème NP-difficile, une approche métaheuristique est adoptée pour résoudre le problème formulé. La

particularité de cette étude réside dans l'adoption d'une nouvelle politique qui vise à maximiser l'utilisation de ressource CPU d'un serveur tout en respectant un seuil de capacité préalablement fixé. Les résultats de simulation montrent que le modèle proposé contribue, de manière significative, à la réduction de la consommation énergétique.

- **Un modèle basé sur la prédiction pour contrôler la migration en temps réel des VMs dans un centre de données ;**
  Comme troisième contribution, nous proposons un modèle, basé sur le *Machine Learning*, adaptatif qui permet de prédire avec une haute précision les différentes caractéristiques de la migration en temps réel en fonction de l'algorithme de migration et de la charge de travail qui s'exécute dans la VM. Nous abordons les paramètres d'entrée importants pour modéliser avec précision les métriques cibles, et nous décrivons comment les modéliser avec peu de dépenses. Comparativement aux travaux existants, nous pouvons modéliser tous les algorithmes de migration couramment utilisés, ainsi que prédire des paramètres importants qui n'ont pas été pris en compte jusqu'à présent, tels que la dégradation des performances de la VM. En comparaison avec l'état de l'art, nous montrons que le modèle proposé surpasse significativement les modèles statiques couramment utilisés.
- **Un nouvel algorithme d'apprentissage non supervisé permettant d'analyser et de détecter des anomalies de fonctionnement de système ;**
  Pour la quatrième contribution, nous introduisons un nouvel algorithme de *Clustering* pour analyser le comportement des VMs lors de l'exécution des charges de travail dans un système en fonction des détails de l'utilisation des ressources (comme l'utilisation du CPU, les événements d'arrêt, etc.). Les informations identifiées par le processus de *Clustering* sont indispensables pour le contrôle et la sécurité. L'algorithme proposé adopte un mécanisme intuitif de regroupement des oiseaux dans la nature pour former des groupes convenables. L'évaluation expérimentale, basée sur un ensemble de données réel, montre que le nouvel algorithme donne des résultats intéressants.

1.4 Organisation de la thèse

La structure globale de cette thèse est présentée dans la figure 3. Le reste de la thèse est organisé comme indiqué ci-dessous.

- **Chapitre 2** : examine les concepts du *Cloud Computing* et centre de données, et il donne les connaissances de base des différentes techniques étudiées ainsi que les principales théories, algorithmes et modèles utilisés dans le cadre de cette thèse.

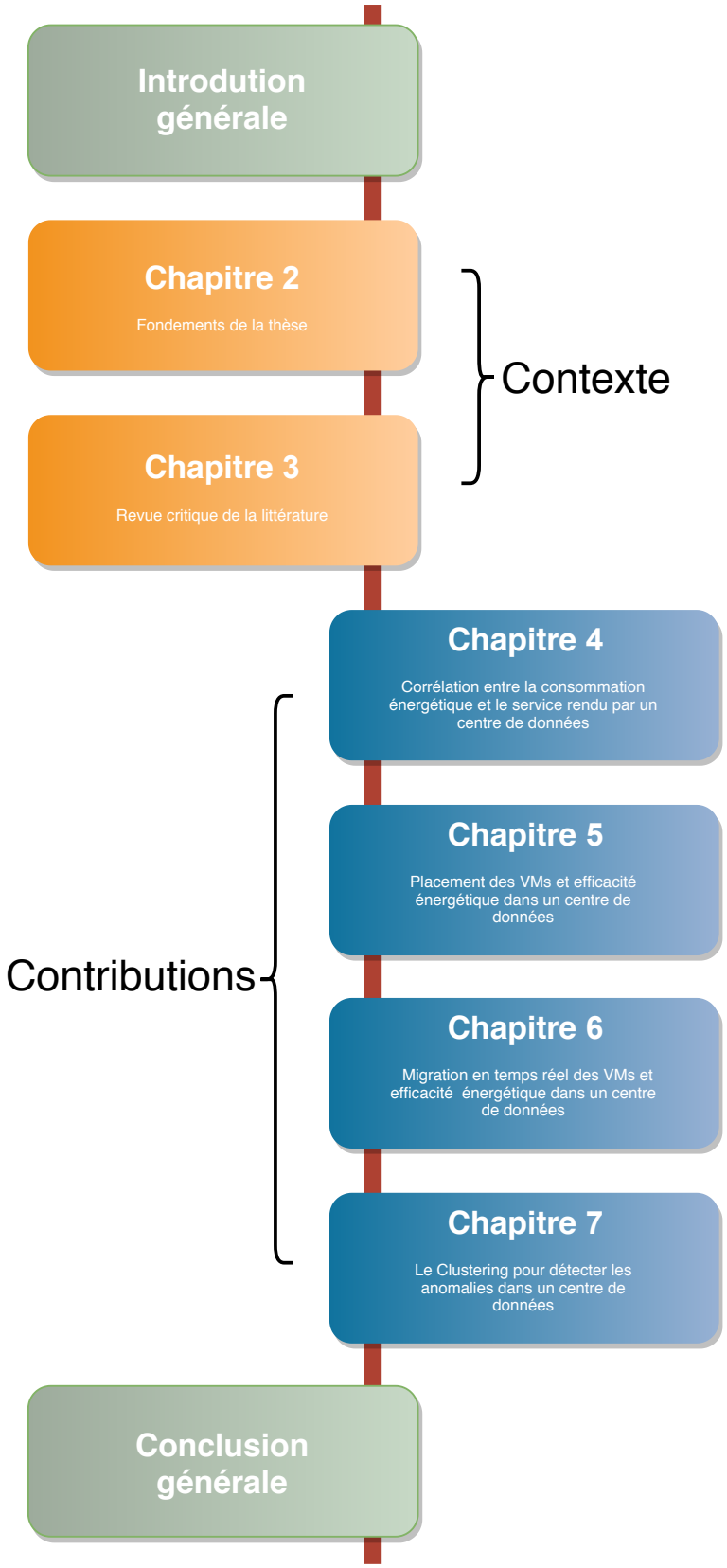


FIGURE 3 – Structure organisationnelle de la thèse.

- **Chapitre 3** : présente une synthèse compréhensive des études liées à nos thèmes de recherche.
- **Chapitre 4** : correspond à notre première contribution. Ce chapitre introduit le modèle stochastique proposé pour établir une relation de corrélation entre le service rendu par un centre de données et la consommation énergétique.
- **Chapitre 5** : représente notre deuxième contribution. Dans ce chapitre, nous proposons une approche de placement statique des VMs orientée-énergie, suivant un problème d'optimisation combinatoire. nous utilisons différentes techniques de résolution pour aborder le problème de placement des VMs défini.
- **Chapitre 6** : correspond à notre troisième contribution. Nous proposons un modèle basé sur le *Machine Learning* pour améliorer le processus de migration en temps réel des VMs au sein d'un centre de données. Le modèle se base sur des techniques de prédiction pour identifier l'algorithme de migration le plus convenable pour une situation donnée.
- **Chapitre 7** : correspond à notre quatrième contribution. Dans ce chapitre nous développons un nouvel algorithme de *Clustering* qui permet d'identifier les anomalies potentielles dans le système d'un centre de données. Cette détection permet de prévenir les causes de dysfonctionnement ou de pannes, ce qui permet de protéger le système et de garantir la continuité d'exécution des services.

2

Fondements de la thèse

Ce chapitre fournit des informations générales sur les concepts et les technologies établis qui constituent la base des travaux réalisés dans le cadre de cette thèse. Nous commençons par illustrer et définir clairement les concepts du Cloud Computing qui sont utilisés dans notre recherche, puis nous présentons les modèles, théories et concepts utilisés dans nos solutions provenant d'autres domaines scientifiques.



Facebook : https://www.journaldugeek.com/2016/09/29/zuckerberg-photos-data-center-facebook-suede/

2.1 Introduction

Dès la fin du dernier siècle, le recours aux outils informatiques a augmenté de façon spectaculaire, à la fois par le simple public et par les organisations. Les usages de cet outil se sont largement multipliés, à tel point qu'il est devenu incontournable dans la communauté contemporaine. Réseaux sociaux, messagerie instantanée, modélisation des phénomènes nucléaires, cartographie météorologique, management de l'entreprise, étude des évolutions des marchés, commerce électronique, diffusion de médias, etc, sont autant d'usages quotidiens de l'outil informatique. Le recours aux services à distance, en particulier, s'est considérablement accru, faisant de l'accès à Internet une nécessité pour de plus amples utilisations de l'outil informatique. Tous ces services à distance impliquent une gestion et une mise à jour de leurs capacités de calcul correspondantes.

Cette capacité de calcul est matérialisée sous la forme de serveurs informatiques. Ils exécutent, au sein d'un système d'exploitation, les programmes destinés à assurer les services sollicités. Comme le besoin de services informatiques ne cesse de croître, les entreprises fournissant ces services ont augmenté leur capacité de traitement en conséquence. Cette expansion a été provoquée par un élargissement de leur pool de serveurs, qui a toutefois entraîné des problèmes relatifs au redimensionnement. À titre d'exemple, les enjeux ayant rapport au stockage de données, à la fiabilité du dispositif et à la résilience des services requièrent une attention privilégiée.

Les entreprises sont ainsi poussées à investir fortement dans des espaces consacrés à l'exécution des tâches informatiques. Ces pièces sont désignées communément par le terme "centres de traitement de données" et sont en mesure de contenir un grand nombre de serveurs. Elles sont conçues de manière à pouvoir facilement installer et contrôler aussi bien les serveurs que les équipements utilitaires du centre (commutateurs, climatisation, etc.). Toutefois, l'augmentation du nombre de serveurs dans ces locaux entraîne également de nouveaux problèmes, tels que la consommation électrique, la dissipation de la chaleur ou le taux d'humidité de l'air.

Vu la nature diverse des applications informatiques industrielles, la mise en place et le maintien de ces centres restent donc des opérations délicates. Notamment, le fonctionnement de chaque centre comporte des caractéristiques spécifiques. Quelques centres sont par exemple dédiés à l'hébergement de services spécifiques comme le calcul à haute performance (HPC) et les bases de données, et sont donc adaptés à ces fins. D'autres centres, en revanche, visent à héberger une variété de services, avec des performances fluctuantes. Cette variabilité de services fait que les activités de chaque centre soient distinctes. En outre, au sein d'un centre de données, les activités des serveurs sont également variables, aussi bien entre les différents serveurs du centre qu'au fil du temps pour un même serveur.

On appelle cette activité variable du serveur une charge de travail (*workload*). Celle-ci correspond à la consommation des ressources d'un serveur relativement à sa capacité en ressources. L'administrateur tient à maintenir une charge normale du centre de données à un niveau bas, afin de permettre l'évolutivité en charge des applications hébergées. Même pour les centres à haute performance, dans lesquels les services exploitent les serveurs à leur capacité maximale, certains serveurs attendent que les tâches soient accomplies, et donc inactifs. Les centres de données sont alors surdimensionnés au niveau de la disponibilité des ressources informatiques et de leur infrastructure technique (stockage, réseau, climatisation dédiée).

Certaines entreprises sont amenées, pour des motifs de rentabilité, à louer certaines des machines installées dans leur centre à des organismes extérieurs (comme *Amazon*). Ce type d'externalisation permet de partager à la fois la salle, les équipements et l'administration du centre, et ainsi le coût du service. Au fur et à mesure que les besoins des centres de données se multiplient, leur mise en location est devenue une forme d'activité autonome. Ainsi, des entreprises particulières se sont consacrées à héberger des services informatiques. Quant à l'entreprise bénéficiaire, la délocalisation de ses services entraîne une réduction des coûts de maintenance et du besoin de personnel spécialisé. Elle améliore en outre la qualité de ses services, à travers, entre autres, des infrastructures de stockage ou de réseau sécurisées, et/ou une expertise de pointe.

Le *Cloud Computing* renforce cette notion de délocalisation de services en y introduisant deux notions : l'élasticité des ressources et la redevance en fonction de l'utilisation. Ces dernières peuvent être réalisées par le biais de la technologie de virtualisation des ressources.

2.1.1 La virtualisation

Compte tenu de la variété des systèmes d'exploitation (OS) et de leurs nombreuses versions, y compris les configurations souvent spécifiques requises pour accueillir le large éventail d'applications populaires, il est devenu de plus en plus difficile d'établir et de gérer des systèmes informatiques modernes. En raison de ces défis, et de plusieurs autres avantages importants qu'elle offre, la virtualisation est devenue une technologie essentielle pour la gestion des systèmes informatiques. La virtualisation permet à un seul ordinateur d'héberger plusieurs VMs (en utilisant éventuellement différentes piles de systèmes d'exploitation) en plus d'un système d'exploitation "natif". Les avantages de la virtualisation comprennent une meilleure utilisation des ressources, une allocation dynamique des ressources et une flexibilité accrue.

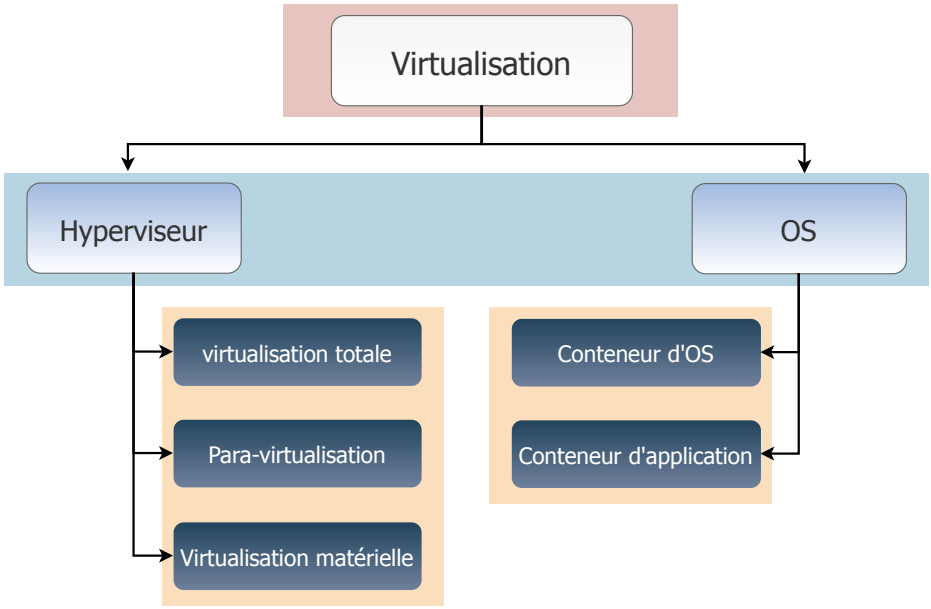


FIGURE 4 – Classification des plates-formes de virtualisation.

La virtualisation a une longue histoire, remontant au début des années 1960, avec des entreprises comme *General Electric*[1], *Bell Labs*[2] et *IBM*[3]. La technologie de la virtualisation a évolué depuis ces premiers systèmes de partage du temps et les ordinateurs centraux avec le support de la virtualisation, mais ce n'est qu'au début des années 2000 que la technologie s'est répandue, les systèmes basés sur $x86$ ayant commencé à prendre en charge les principales instructions de virtualisation dans le matériel [165].

Nous présentons un aperçu des plate-formes de virtualisation les plus courantes dans les environnements de production : méthodes de virtualisation basées sur l'hyperviseur et méthodes de virtualisation basées sur le système d'exploitation. Les technologies spécifiques sont récapitulées dans la figure 4.

Plate-formes basées sur un hyperviseur : Les systèmes basés sur l'hyperviseur sont les systèmes de virtualisation traditionnels, dans lesquels un hyperviseur est utilisé pour gérer le matériel physique sous-jacent. L'hyperviseur est chargé de créer et de gérer l'exécution des instances de systèmes d'exploitation virtualisés, VMs, et d'en allouer des ressources. L'hyperviseur isole les VMs du système hôte et permet d'héberger des VMs exécutant différents systèmes d'exploitations invités. Les systèmes basés sur l'hyperviseur sont généralement plus lents que les autres systèmes de virtualisation en raison des coûts supplémentaires engendrés par la présence d'un hyperviseur et de plusieurs noyaux de systèmes d'exploitation invités.

1. General Electric : https://www.ge.com/
2. Bell Labs : https://www.bell-labs.com/
3. IBM : https://www.ibm.com/

Les systèmes basés sur l'hyperviseur sont généralement classés en deux catégories : Les hyperviseurs de type 1 (virtualisation totale ou native) fonctionnent directement sur le matériel hôte (par exemple *Xen*[4]) ; les hyperviseurs de type 2 (para-virtualisation ou virtualisation matérielle) fonctionnent au-dessus du système d'exploitation de l'hôte. Cependant, la distinction entre les deux types d'hyperviseurs n'est pas toujours claire : le *KVM*[5], par exemple, présente des caractéristiques des deux types.

Dans les systèmes basés sur le protocole d'hyperviseur, la virtualisation complète a pour effet d'éloigner complètement le système d'exploitation invité du matériel sous-jacent. Dans de tels cas, les systèmes d'exploitation fonctionnant sur le matériel virtualisé ne sont en aucune façon modifiés. En fait, ils ignorent essentiellement qu'ils fonctionnent dans un environnement virtualisé. En revanche, la para-virtualisation présente une interface avec le système d'exploitation invité qui diffère du matériel sous-jacent. Ainsi, le noyau du système d'exploitation invité doit être modifié afin de fonctionner avec l'interface fournie par l'hyperviseur. La virtualisation matérielle exploite les fonctions de virtualisation intégrées dans le CPU de l'hôte.

Les dernières générations de processeurs d'*Intel*[6] et d'*AMD*[7] - respectivement les technologies *Intel VT* et *AMD-V* - prennent en charge la virtualisation du matériel. Ce type de virtualisation fournit les extensions matérielles nécessaires pour exécuter des VMs invitées non modifiées et permet une augmentation considérable des performances des VMs [21].

Plate-formes basées sur le système d'exploitation : également connus sous le nom de systèmes basés sur des conteneurs, n'utilisent pas d'hyperviseurs, mais virtualisent plutôt les ressources au niveau du système d'exploitation. De tels systèmes ne réalisent pas la virtualisation au même sens que les VMs, mais peuvent être utilisés pour de nombreuses raisons identiques à celles qui justifient l'utilisation des VMs [82]. Les instances virtuelles sont créées en encapsulant les processus standard du système d'exploitation et leurs dépendances ; ces instances sont gérées par le noyau du système d'exploitation sous-jacent. Plus précisément, le moteur de conteneur (*container engine*) remplit les mêmes fonctions que l'hyperviseur dans un système basé sur un hyperviseur, en gérant les conteneurs et les images tout en interagissant avec le noyau du système d'exploitation sous-jacent pour l'allocation et la gestion des ressources de base.

La virtualisation basée sur un système d'exploitation nécessite moins de ressources que la virtualisation basée sur un hyperviseur, car il n'y a pas de couche logicielle intermédiaire (sous la forme de l'hyperviseur) et un seul système d'exploitation est utilisé. Cependant, le

4. Xen : https://xenproject.org/
5. KVM : https://www.linux-kvm.org
6. Intel : https://www.intel.com/
7. AMD : https://www.amd.com/

partage du noyau hôte présente également des inconvénients, tels qu'une isolation réduite entre les instances et une incapacité à héberger différents systèmes d'exploitation invités.

Les plates-formes de virtualisation basées sur le système d'exploitation encapsulent des instances dans des conteneurs dont il existe deux types : les conteneurs de système et les conteneurs d'application. Les plates-formes utilisant des conteneurs de système peuvent exécuter plusieurs processus dans un seul conteneur et sont conçues pour fournir un environnement d'exécution complet, mais avec une conception plus légère que les plates-formes basées sur un hyperviseur. Les conteneurs de système (par exemple, les conteneurs *Linux*[8]) fournissent un environnement similaire à la virtualisation basée sur un hyperviseur, mais sans la contrainte d'avoir un noyau de système d'exploitation hôte séparé. En revanche, les conteneurs d'applications (par exemple, les conteneurs *Docker*[9]) sont conçus pour exécuter un seul processus et sont pratiques pour déployer des applications distribuées, comme dans une architecture à micro-services.

2.1.2 Le *Cloud Computing*

Le *Cloud Computing* est une suite évidente de l'évolution rapide de ces dernières années en matière de virtualisation, de connectivité, de stockage et de gestion des systèmes distribués [33]. Le *Cloud Computing* est un paradigme dans lequel une grande capacité de calcul intégrée est offerte à un ensemble de services qui partagent des ressources. Différentes définitions du *Cloud Computing* ont été proposées [36, 30]. Selon le'Institut *National Institute of Standards and Technology* [176] (NIST), le *Cloud Computing* est un modèle permettant un accès réseau omniprésent, pratique et à la demande à un ensemble partagé de ressources informatiques configurables (par exemple, réseaux, serveurs, stockage, applications et services) qui peuvent être rapidement approvisionnées et libérées avec un minimum d'efforts de gestion ou d'interaction avec les fournisseurs de services.

Une des caractéristiques du *Cloud Computing* est l'élasticité, c'est-à-dire la capacité d'acquérir et de libérer rapidement des ressources informatiques, en fonction de la demande des utilisateurs. L'élasticité peut être horizontale, lorsque le nombre de VMs (ou conteneurs) est modifié pendant le fonctionnement du service, ou verticale, lorsque les ressources disponibles pour une VM, généralement le CPU et la mémoire, sont modifiées. L'élasticité horizontale nécessite le maintien de l'application, par exemple, la capacité de cloner et de synchroniser les états entre les VMs connexes, mais ne nécessite pas de maintien supplémentaire de la part de l'hyperviseur. L'élasticité verticale, par contre, ne nécessite que peu de maintien de la part de l'application, à part le fait d'être *multi-threaded*. Cependant, l'élasticité verticale a besoin du support de l'hyperviseur et du noyau

8. Linux : https://www.linux.com/
9. Docker : https://www.docker.com/

du système d'exploitation invité, par exemple, la capacité de connecter, en temps réel, des ressources au moment de l'exécution [85].

En plus de l'élasticité, les technologies de virtualisation apportent de nombreux avantages au *Cloud Computing*, notamment la possibilité d'exécuter plusieurs VMs sur un seul hôte physique, ce qui améliore l'utilisation des serveurs. La virtualisation permet aux fournisseurs de *Cloud Computing* d'exécuter plusieurs VMs ou conteneurs sur un nombre réduit de serveurs physiques. Par conséquent, la consommation électrique des serveurs et des systèmes de refroidissement peut être réduite. La virtualisation offre également une gestion simplifiée grâce à un certain nombre d'outils de surveillance et de gestion des ressources, permettant de démarrer, d'arrêter, de migrer et de répliquer les VMs avec relativement peu d'efforts.

Plusieurs modèles de services existent pour le *Cloud Computing*, notamment le SaaS, le PaaS et le IaaS. Comme indiqué précédemment, le SaaS permet aux clients d'accéder à des logiciels d'application fonctionnant sur une infrastructure en Cloud. En PaaS, le fournisseur de *Cloud Computing* offre une plate-forme informatique prenant en charge certains langages de programmation, bibliothèques, services et outils, tandis que le client du Cloud gère ses propres applications et données. En IaaS, le fournisseur de *Cloud Computing* offre un ensemble de ressources informatiques, de stockage et de réseau, tandis que le client du *Cloud Computing* gère le système d'exploitation, les applications et les données. Dans cette thèse, nous nous concentrons principalement sur la gestion des ressources dans les centres de données fournissant des services IaaS.

Les services de *Cloud Computing* peuvent être déployés dans des environnements publics, privés ou communautaires, ainsi que dans des architectures hybrides [176]. Les fournisseurs de services en Cloud peuvent posséder et exploiter leurs propres centres de données ou louer des ressources à partir d'une installation partagée, également appelée centre de données à plusieurs exploitants. Les services qui fonctionnent dans l'un de ces modèles de déploiement et de service sont fournis par des centres de données comprenant de nombreux serveurs et équipements associés. Par ailleurs, et pour garantir le rendement convenable d'un service, les fournisseurs mettent en place un contrat de QoS avec le client.

La QoS peut être définie en termes des SLAs qui font partie des engagements d'un fournisseur de services envers un client. Un SLA comprend des objectifs de niveau de service, définis en termes de mesures et d'indicateurs clés de performance spécifiant les valeurs souhaitées de ces mesures. Le choix de ces indicateurs dépend du service considéré [66]. Par exemple, les indicateurs des services de réseau peuvent inclure le temps aller-retour, la perte de paquets et la largeur de bande passante, tandis que les indicateurs des services de stockage peuvent inclure la vitesse moyenne de lecture/écriture, l'entrée/sortie par seconde (IOPS) et l'espace disque disponible. Les exemples d'indicateurs communs à

de nombreux types d'applications comprennent le temps de réponse (le temps pris par une demande jusqu'à l'arrivée de la réponse à l'extrémité de réception) et le débit (la quantité de travail accompli par unité de temps). Les indicateurs de performance peuvent également faire référence à des mesures d'infrastructure telles que le nombre d'instances virtuelles utilisées, le nombre de CPU utilisées, l'utilisation du CPU, la quantité de mémoire fournie, l'utilisation de la mémoire et le nombre minimum de VMs.

2.1.3 Le centre de données

Un centre de données est une facilité qui regroupe un certain nombre de serveurs, d'unités de stockage et d'équipements de réseau, ainsi que tout le matériel non informatique tel que les installations de refroidissement et les alimentations électriques ininterrompues, en raison des exigences environnementales communes, des besoins de sécurité physique et de la facilité de maintenance [139]. La taille des centres de données varie entre les salles de serveurs qui accueillent les petites et moyennes organisations et les fermes de serveurs qui gèrent des services en Cloud à très grande échelle.

Parallèlement au déclin du nombre de centres de données traditionnels qui hébergent des applications sur un matériel dédié, le nombre de centres de données modernes dans le Cloud qui utilisent la virtualisation pour une gestion efficace des applications est en augmentation. L'indice *Cisco Global Cloud*[10] indique que 75% de toutes les charges de travail informatiques étaient exécutées dans des centres de données dans le Cloud en 2015, et prévoit que ce pourcentage passe à 95% en 2020. Dans cette thèse, nous nous focalisons sur les centres de données dans le Cloud et utiliserons ci-après les termes "centre de données" et "centre de données dans le Cloud" de manière alternée.

Les possibilités d'économies d'échelle - telles que la réduction du coût de l'électricité, des logiciels et du matériel - rendent la construction de grands centres de données financièrement importante. La croissance des grands centres de données est également stimulée par les capacités accrues des services en Cloud, la croissance rapide du *Cloud Computing* en tant que paradigme permettant de fournir des services en ligne, et la quantité extraordinaire de données générées par les applications modernes. Ces centres de données hébergent généralement des centaines et des milliers de serveurs qui consomment d'énormes quantités d'énergie. La consommation énergétique est donc devenue une préoccupation importante en raison de son impact sur les coûts d'investissement, les dépenses d'exploitation et la durabilité environnementale.

Dans ce qui suit, la section 2.2 présentent les différentes techniques adoptées dans notre thèse pour optimiser la consommation énergétique dans un centre de données, ainsi que la section 2.3 expose les algorithmes et les concepts utilisées pour adresser ces techniques.

10. Cisco Global Cloud : https://www.cisco.com/

2.2 Techniques de réduction de la consommation énergétique dans un centre de données

Afin de répondre à la demande croissante de services dans le Cloud, de nouveaux centres de données peuvent être construits, ou les installations existants peuvent être étendus. Cependant, l'extension des capacités des centres de données n'est pas toujours possible en raison de contraintes économiques et physiques telles que la capacité fixe des installations de production d'énergie et la densité limitée des serveurs. La construction de nouveaux centres de données peut coûter des millions de dollars, ce qui augmente considérablement le coût total de possession, à la fois en termes de dépenses d'investissement nécessaires pour construire une nouvelle installation et d'exploitation nécessaire pour faire fonctionner le centre de données. Une alternative à l'extension des centres de données existants ou à la construction de nouveaux centres de données consiste à mettre en place des techniques pour optimiser l'utilisation des ressources informatiques, et par la suite, la consommation énergétique.

Les approches actuelles de la gestion énergétique varient en termes de stratégies, d'hypothèses et de caractéristiques d'application. En général, l'efficacité énergétique comprend deux aspects principaux, la durabilité énergétique et la réduction des coûts énergétiques. Bien que la durabilité énergétique et les économies de coûts soient corrélées, les stratégies pour y parvenir sont différentes.

Les approches de la durabilité énergétique visent à réduire les émissions de CO_2 pour une minimisation des coûts et une durabilité écologiques grâce à l'utilisation de l'électricité produite par des sources d'énergie renouvelables, telles que l'énergie solaire, éolienne, houlomotrice, marémotrice, géothermique, hydroélectrique, la biomasse, les gaz de décharge, les gaz des stations d'épuration des eaux usées et les biogaz. La création de solutions écologiques a récemment fait l'objet d'une grande attention. Principalement motivés par les avantages financiers, environnementaux et concurrentiels, les opérateurs des centres de données actuels, dont *Facebook*, *Apple*, *Google*, *eBay* et *Microsoft*, ont évolué vers des opérations plus écologiques. Le lecteur intéressé est invité à consulter [86, 94, 217].

Les approches d'optimisation de la consommation énergétique visent à réduire le coût de l'énergie dans les centres de données. Les travaux dans ce domaine portent sur la gestion de l'énergie à différents niveaux, notamment au niveau des applications, au niveau des serveurs, au niveau des clusters/centres de données, et au niveau inter-centres de données. Nous décrivons ces approches plus en détail, tout en se concentrant davantage sur les approches de gestion de l'énergie au niveau des serveurs et des clusters, qui constituent l'objet principal de cette thèse.

L'efficacité énergétique au niveau du serveur se concentre sur le serveur lui-même et/ou

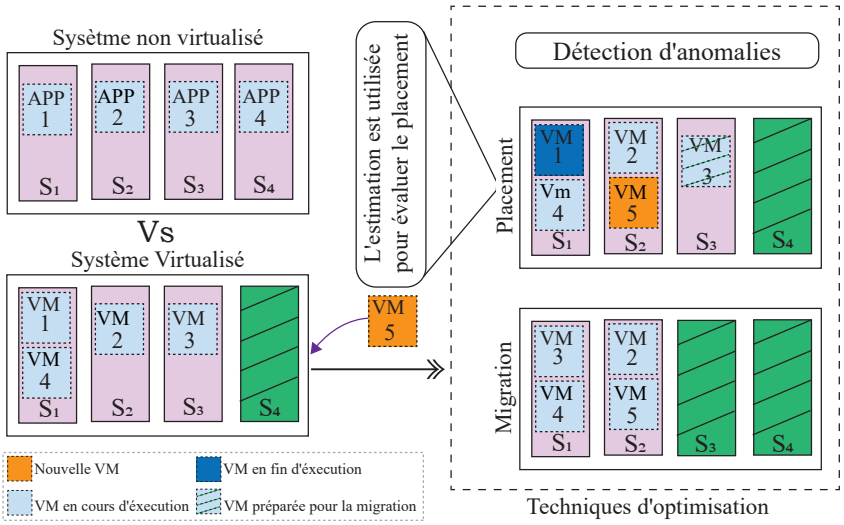


FIGURE 5 – Vue d'ensemble des techniques adressées dans cette thèse.

ses composants (cette dernière approche étant parfois dite au niveau des composants). Les travaux dans ce domaine ont pris en compte le rendement énergétique découlant des améliorations apportées au matériel, aux systèmes d'exploitation et aux systèmes virtualisés ; ils ont également pris en compte la gestion statique et dynamique de l'énergie. Une grande partie des travaux dans ce domaine s'est concentrée sur la réduction de la consommation d'énergie du CPU et/ou de la mémoire, les principaux consommateurs d'énergie dans un serveur (section 3.2).

D'autre part, la gestion de l'énergie dans un centre de données est rendue possible par la modélisation de la consommation des serveurs. Pour chaque nouvelle technologie de serveur, un nouveau travail de modélisation est nécessaire. Ainsi, il est possible, selon [201], de mesurer précisément l'activité (le service) d'un serveur à l'aide de compteurs de performance, et de modéliser une relation entre l'activité d'un serveur et sa consommation. Différents modèles de consommation sont comparés dans la sous-section 3.2.4.

Le basculement d'un centre physique vers un centre virtualisé nécessite la mise en corrélation de la consommation de serveur avec son activité, ainsi qu'avec l'activité de ses VMs. Par conséquent, la virtualisation apporte de nouvelles techniques de gestion des ressources et de la consommation énergétique ; notamment, l'estimation de la consommation énergétique des serveurs, la consolidation par placement et migration en temps réel des VMs, ainsi que la détection des comportement anormaux du système (figure 5).

2.2.1 Placement des VMs

Le placement des VMs est un processus qui consiste à sélectionner la PM la plus appropriée pour une VM donnée (figure 6). Ainsi, un algorithme de placement des VMs

vise à déterminer la correspondance optimale entre la VM et la PM, qu'il s'agisse d'un placement initial de la VM ou d'une migration de la VM pour une ré-optimisation du placement. La technique de placement pour la consolidation des VMs peut avoir deux objectifs principaux : l'un est la réduction de la consommation énergétique et l'autre est la garantie de la QoS. Le type d'approche de placement des VM varie d'un fournisseur de services Cloud à un autre. Il existe un conflit évident entre ces deux objectifs.

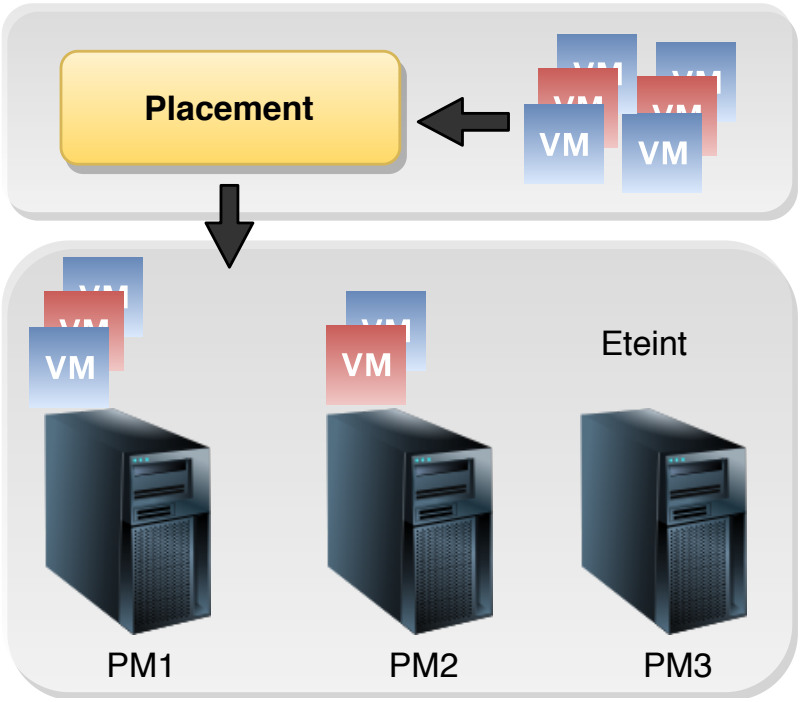


FIGURE 6 – Le mécanisme de placement des VMs.

Selon l'objectif du placement, nous distinguons trois types d'algorithmes de placement des VMs :

— **Approches basées sur la puissance** : visent à obtenir une modélisation VM-PM qui aboutit à un système énergétiquement efficace avec une utilisation maximale des ressources.

— **Approches basées sur la QoS** : visent à obtenir une modélisation VM-PM pour assurer une satisfaction maximale des exigences de QoS.

— **Approches hybrides** : visent à obtenir une modélisation VM-PM qui permet d'assurer une satisfaction maximale de QoS, tout en minimisant l'utilisation de l'énergie.

Selon le type d'approche utilisée pour obtenir la correspondance souhaitable entre les VMs et les PMs, les techniques de placement des VMs peuvent être classées en plusieurs catégories, nous pouvons citer les suivants :

1) Programmation sous contraintes : il s'agit d'une sorte de programmation logique visant à résoudre un problème combinatoire complexe de placement optimal des VMs. Elle utilise un ensemble de contraintes qui peuvent être généralement étendues à d'autres aspects [71].

2) *Bin-Packing* : le problème classique du conditionnement par bacs consiste en un ensemble d'articles ayant des tailles spécifiées dans l'intervalle $[0, 1]$ qui doivent être rangés dans le plus petit nombre possible de bacs de capacité égale à 1. Pour modéliser ce problème comme un algorithme d'allocation des ressources, nous considérons chaque élément comme une VM à placer dans un nombre minimum de bacs, chacun étant considéré comme une PM [208]. Sachant que le problème de sac à dos, utilisé dans cette thèse, est un cas particulier de *Bin-Packing* dans lequel le nombre de bacs est égal à 1.

3) Programmation stochastique en nombres entiers : il s'agit d'une technique d'optimisation mathématique dont les exigences futures sont incertaines. Elle fait appel à des modèles d'estimation utilisant des distributions de probabilité des données concernées. Ici, la demande future d'une VM ou d'une application est inconnue et, par conséquent, certaines techniques de placement des VMs utilisent cette approche pour prévoir la correspondance VM-PM appropriée [46].

Éventuellement, les algorithmes de placement des VMs vont du statique au dynamique. Ils sont en outre étendus aux algorithmes adaptatifs qui ont la capacité de se modeler en fonction de l'évolution de la charge de travail et de la demande. Nous pouvons facilement observer que pour obtenir une consolidation optimale des serveurs, différentes approches se concentrent sur différents aspects du scénario d'optimisation. L'approche basée sur les contraintes peut réduire le nombre de serveurs actifs, mais le temps de recherche prolongé reste un inconvénient majeur. Les algorithmes basés sur le *Bin-Packing* peuvent réduire considérablement le nombre de serveurs actifs requis et donc la consommation d'énergie nécessaire, mais ils aboutissent parfois à un regroupement extrêmement serré des VMs sur les PMs. Les techniques stochastiques devraient être une option à utiliser lorsqu'il y a plus d'incertitude dans les paramètres qui peuvent affecter les coûts. Dans le cas contraire, les techniques de prédiction nécessitent la recherche dans de grands espaces. La section 3.3 fournit un aperçu sur différentes techniques existant dans la littérature.

2.2.2 Migration en temps réel des VMs

La migration en temps réel désigne le déplacement d'une VM ou d'une application en cours d'exécution entre différentes MPs sans déconnecter le client ou l'application. L'état du processeur, le contenu de la mémoire, le stockage local et l'état du réseau de la VM sont transférés de la l'hôte source vers la destination. La migration d'une VM en cours d'exécution nécessite le déplacement de son état complet d'un hôte physique à un autre.

TABLE 2.1 – Les phases usuelles du processus de migration en temps réel.

Phase	*Description*
Profinling	Le moniteur de VM (VMM) établit le profil de la VM et l'état du système de l'hôte physique. Le profiling peut entraîner une légère réduction des performances de la VM.
Préparation	La migration en temps réel est lancée. Le VMM met la VM dans un mode géré et adapté à la migration en temps réel, ce qui entraîne généralement une légère réduction des performances de la VM. L'hôte source commence à envoyer des parties de l'état volatil à l'hôte de destination.
Suspension	La VM est arrêtée à la fois sur l'hôte source et sur l'hôte de destination et n'est donc pas disponible pour l'utilisateur.
Reprise	La VM est redémarrée sur l'hôte de destination. Les parties de l'état volatil qui manquent encore sont récupérées de l'hôte source. Les performances de la VM peuvent encore être réduites jusqu'à la fin de la phase de reprise.

Dans le cas d'une migration au sein d'un même centre de données, le stockage permanent est généralement assuré par le stockage en réseau (NAS) et n'a pas besoin d'être déplacé. L'état volatil d'une VM comprend sa mémoire et l'état des processeurs virtuels (VCPU) et des autres composants. La mémoire, qui occupe environ quelques gigaoctets, constitue la plus grande partie de l'état volatil d'une VM.

La façon la plus simple de déplacer une VM est d'arrêter complètement la machine sur l'hôte source, de transférer l'état entier vers la destination, puis de redémarrer la VM sur l'hôte de destination. Cependant, le long temps d'arrêt de cette technique de suspension et de copie n'est pas pratique pour la plupart des environnements ; il est donc courant de migrer les VM en cours d'exécution pour minimiser les effets perceptibles de la migration [221].

On distingue les phases suivantes du processus de migration en temps réel comme indiqué dans le tableau 2.1 (figure 7).

Sachant que certaines phases ne sont pas présentes dans tous les algorithmes de migration. Les techniques d'arrêt et de duplication, par exemple, comprennent une longue phase de suspension avec des phases de préparation et de reprise très courtes. La phase du profiling est spécifique aux approches de modélisation. Les algorithmes de migration en temps réel qui existent sont particulièrement dépendants de la charge de travail qui s'exécute dans la VM et de l'état de l'hôte physique. Pendant la phase du profiling, le VMM rassemble les paramètres nécessaires qui affectent les performances de la migration

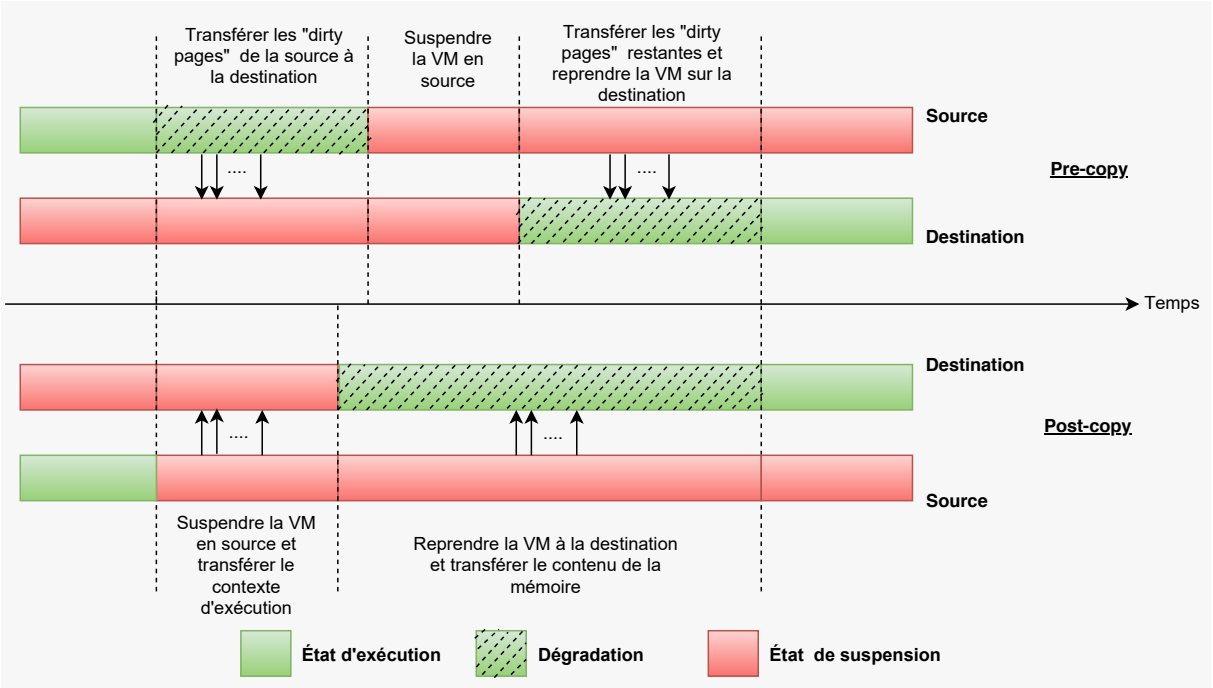


FIGURE 7 – Aperçu des processus de migration *Pre-Copy* et *Post-Copy*.

en temps réel et les intègre au modèle.

2.2.2.1 Algorithmes de migration en temps réel

Les algorithmes de migration en temps réel peuvent être classés en fonction du moment où l'état volatil de la VM est copié sur l'hôte de destination : pendant la phase de préparation (*Pre-Copy*), pendant la phase d'arrêt (suspension et duplication) ou pendant la phase de reprise (*Post-Copy*). D'autres algorithmes sont des hybrides de ces trois approches de base ou optimisent un certain aspect d'une méthode. Nous modélisons les cinq techniques qui sont prises en charge par les principaux environnements de virtualisation [121, 197].

1) *Pre-Copy (PRE)* [14] : transfère itérativement l'état volatil d'une VM pendant la phase de préparation. Dans chaque itération, les pages mémoire qui ont été modifiées par la VM en cours d'exécution depuis l'itération précédente sont envoyées à l'hôte de destination. Lorsque le nombre de ces *Dirty Pages* devient inférieur à un seuil donné, la VM est suspendue. Les *Dirty Pages* restantes et l'état du VCPU sont transférés, et la VM est redémarrée sur l'hôte de destination. Une des limites de cet algorithme est qu'il ne converge jamais si le taux d'infection de la mémoire est supérieur à la bande passante disponible du réseau. Ainsi, si le nombre de *Dirty Pages* par itération reste stable ou si un certain nombre maximum d'itérations a été atteint, une condition d'arrêt est activée pour empêcher ces migrations non terminantes. Néanmoins, *Pre-Copy* est l'algorithme de

migration par défaut pour la plupart des plateformes de virtualisation.

3) La compression des données (DTC) [25] : consiste à compresser les pages de mémoire avant leur transmission. Cette technique nécessite une quantité importante de ressources de calcul supplémentaires et peut donc ne pas être une option viable si l'utilisation du CPU sur l'hôte est élevée.

4) La compression delta (DLTC) : [35] est une autre optimisation pour la *Pre-Copy* qui applique la compression delta aux pages partiellement modifiées lors de la migration en temps réel. Cette technique peut nécessiter une quantité importante de mémoire supplémentaire pour stocker les pages en mémoire pour les calculs delta éventuels.

2) L'étranglement du CPU, *Throttling* (THR) [32] : est une technique qui impose la convergence du processus de *Pre-Copy* en diminuant explicitement le temps CPU alloué à une VM pour réduire le taux de *Dirty Pages*. La limitation du CPU peut, dans certains cas, dégrader de manière significative les performances de la charge de travail exécutée dans la VM.

5) *Post-Copy* (POST) [168] : est diamétralement opposé à *Pre-Copy* dans le sens où la VM est immédiatement redémarrée sur l'hôte de destination après avoir transféré uniquement le contexte d'exécution (registres VCPU et états des unités). Le contenu de la mémoire est transféré par un processus en arrière-plan. Les accès à des pages non encore copiées déclenchent un dysfonctionnement du VMM qui va alors chercher la page accédée à la demande. Ces pannes peuvent entraîner une dégradation des performances du VMM pendant la phase de reprise. *Post-Copy* est intéressant car il est caractérisé par un temps d'arrêt court et constant et ne transfère chaque page de mémoire qu'une seule fois. Toutefois, la dégradation potentiellement grave des performances pendant la phase de reprise limite son applicabilité dans les environnements avec des SLAs plus stricts.

Notons que THR et la compression de la mémoire DTC sont des optimisations indépendantes qui peuvent, en principe, être appliquées aux deux approches *Pre-Copy* et *Post-Copy*. Dans ce travail, nous utilisons les techniques fournies par le framework de virtualisation : l'algorithme sous-jacent pour THR, DLTC et DTC est *Pre-Copy*, alors qu'aucune optimisation n'est appliquée à la *Post-Copy*.

2.2.2.2 Métriques de migration en temps réel

Dans cette étude, nous comparons et prévoyons les mesures suivantes pour les différents algorithmes de migration en temps réel.

a) Temps total de migration (TT) : Il s'agit de la somme des temps de migration de toutes les VMs migrantes. Sa valeur peut varier en fonction de la quantité de données à déplacer pendant la migration et du débit de migration. Elle dépend de la quantité totale de mémoire transférée du serveur source au serveur de destination, ainsi que de la bande

passante allouée ou de la vitesse de transmission. Supposons que v_m désigne la quantité totale de mémoire, et b correspond à la largeur de la bande passante ; cette métrique est donnée par :

$$TT = \frac{v_m}{b}$$

b) Utilisation de la mémoire du hôte (MEM) : ressources mémoire consommées par la VM pendant la migration en temps réel sur l'hôte source.

c) Quantité totale de données transférées (TD) : quantité totale de données transférées à l'hôte de destination.

d) Temps d'arrêt (DT) : C'est le délai pendant lequel le service n'est pas exécuté ou disponible en raison de la migration des états du processeur. Le temps d'arrêt s'étend parce que les algorithmes actuels ne sont pas capables de garder une trace des *Dirty Pages* de la VM en cours de migration. Le temps d'arrêt dépend du taux de *Dirty Pages* d_r, de la taille de la page p_s, de la durée t_n du dernier tour de *Pre-Copy*, et de la largeur de la bande passante. [34] nous définissons cette métrique comme :

$$DT = \frac{d_r \times p_s \times t_n}{b}$$

e) Dégradation des performances (PERF) : performance réduite de la VM pendant la migration en temps réel, mesurée à partir du nombre d'instructions exécutées par seconde (IPS).

f) Utilisation du CPU de l'hôte (CPU) : désigne la quantité totale de ressources CPU consommées, sur l'hôte source, par la VM pendant la migration en temps réel.

Les deux premières mesures intéressent les opérateurs de centres de données pour estimer les ressources nécessaires à la migration en temps réel, tandis que le temps d'arrêt et la dégradation des performances sont susceptibles d'affecter les SLAs et la QoS dont bénéficient les utilisateurs.

Dans la section 6, nous présentons une synthèse des modèles existants de migration des VMs en temps réel. Le terme "modèle" est utilisé pour la représentation théorique des algorithmes impliqués dans la migration des VMs. Nous proposons ensuite un modèle basé sur la prédiction afin de sélectionner l'algorithme le plus approprié pour migrer une VM donnée, de manière à garantir les SLAs tout en optimisant l'utilisation des ressources.

2.2.3 Détection des anomalies du système

L'objectif de la détection des anomalies dans un centre de données est d'identifier les symptômes de problèmes (par exemple, la dégradation des performances) ou de prédire des anomalies pour prévenir de futurs incidents. La détection d'anomalies analyse les

données métriques pertinentes, en temps approprié, pour découvrir les signes indiquant un problème (par exemple, un temps de réponse dégradé), généralement en fonction des écarts par rapport aux lignes de base, aux modèles ou à la violation des exigences, par exemple, les SLAs. La détection des anomalies est un sujet bien étudié dans divers domaines de recherche et d'application. Le concept est largement utilisé dans des disciplines telles que la gestion de réseau, la détection des défauts, la cybersécurité et les capteurs industriels pour découvrir des modèles non conformes, la principale motivation étant que les modèles anormaux indiquent un événement significatif et exploitable dans le système [24].

Selon [24], les anomalies sont des phénomènes qui ne se conforment pas à une certaine notion de comportement normal. Les autres termes utilisés pour décrire les anomalies dans la littérature sur les systèmes comprennent les observations aberrantes, l'anomalie, l'exception, la nouveauté et les observations discordantes. Dans le domaine des tests de performance des logiciels, les anomalies peuvent également être appelées anti-modèles. En général, les anomalies de performance peuvent être définies comme des déviations dans le comportement d'un ou plusieurs indicateurs de performance clés par rapport à une règle, un modèle normal ou une exigence connue [84], ou comme des changements inutiles dans les performances clés des indicateurs d'intérêt, tels que l'augmentation anormale de la latence ou de l'utilisation des ressources. Les anomalies de performance signifient généralement une défaillance du système sous-jacent, des bugs d'application, un manque de capacité et des conflits, ou indiquent l'effet secondaire des opérations d'évaluation de la résilience.

2.2.3.1 Types d'anomalies de performance

Les anomalies de performance peuvent se manifester sous différentes formes [24]. Les types d'anomalies les plus courants sont des anomalies ponctuelles, c'est-à-dire des points qui s'écartent de la normalité, que ce soit localement dans un certain voisinage ou globalement dans l'ensemble du jeu de données. Un point de données anormal peut se présenter sous la forme d'un creux (voir la figure 8(a)) ou d'un pic dans la courbe métrique (voir la figure 8(e)). Certains points de données apparaissent également comme des anomalies contextuelles. Ces points de données peuvent être marqués comme anormaux dans un contexte (par exemple, sous un niveau de charge donné, ou à un moment particulier de la journée) et normaux dans un autre. Les figures 8(b) et 8(c) représentent toutes deux des anomalies contextuelles, en particulier des points de données anormaux se produisant dans les points bas des schémas cycliques.

En outre, un groupe homogène de points de données peut avoir des attributs qui s'écartent tellement du reste de l'ensemble de données qu'il peut être qualifié d'anormal. De tels points de données sont appelés anomalies collectives. Par exemple, en supposant

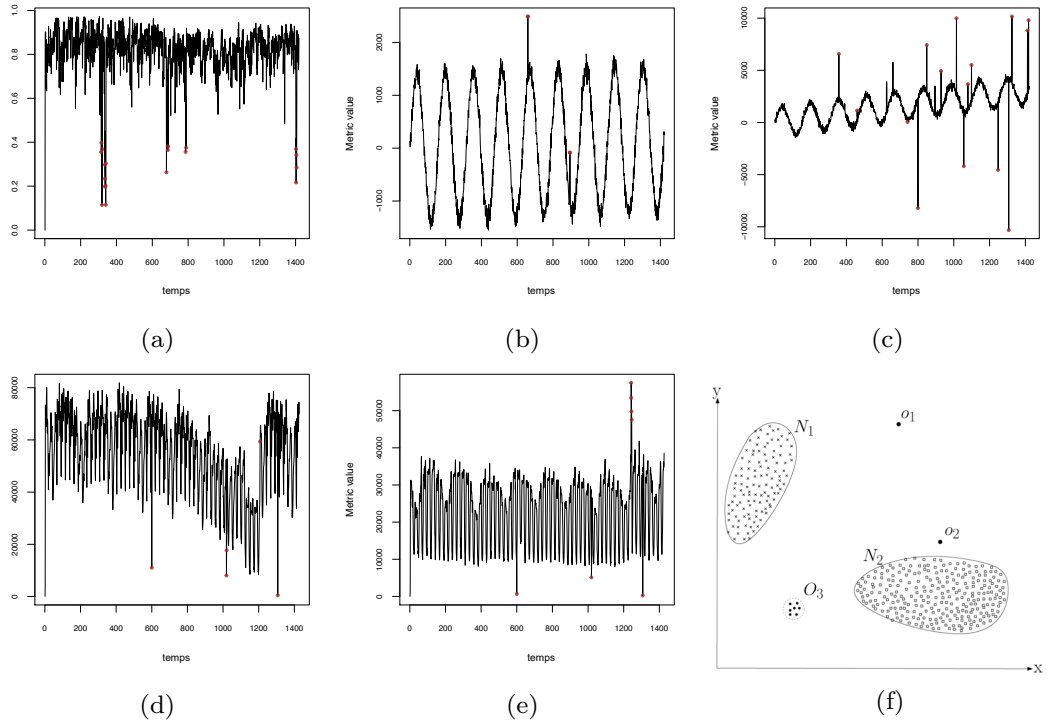


FIGURE 8 – Exemples d'anomalies de performance [24].

que la figure 8(f) est un graphique de l'utilisation moyenne de la mémoire (axe X) et de l'utilisation du CPU (axe y) sur un ensemble de serveurs. Les clusters N_1 et N_2 peuvent représenter les régions normales des données. Cela implique que la plupart des serveurs sont soit intensifs en mémoire, soit en calcul. Les points de données constituant le cluster O_3 couvrant les serveurs qui ne sont ni à forte consommation de mémoire ni à forte consommation de CPU peuvent être qualifiés comme des anomalies, le même pour les points éloignés o_1 et o_2. Des anomalies collectives peuvent également se produire dans les données de séries temporelles sous la forme d'une séquence de points de données connexes qui sont anormaux par rapport au reste des données ou à un voisinage spécifique.

2.2.3.2 Approches de détection d'anomalies

Les algorithmes de détection des anomalies sont généralement basés sur une variété de techniques couvrant des domaines tels que l'analyse statistique, le *Machine Learning*, les algorithmes évolutifs et l'analyse des séries temporelles [93] et peuvent être classés en deux grandes catégories. La détection d'anomalies supervisée, couramment étudiée dans le domaine de la sécurité des réseaux, suppose l'existence d'un ensemble de données étiquetées où les instances ou les enregistrements sont de nature multivariée et sont associés à une étiquette indiquant s'il s'agit d'enregistrements normaux ou d'anomalies. L'objectif est

alors de concevoir un modèle qui permet de classer suffisamment les nouvelles instances comme anormales ou non. Inversement, la détection non supervisée d'anomalies de performance porte principalement sur des données non étiquetées qui se présentent également sous différentes formes (par exemple, des séries temporelles) univariées et multivariées ou des données transversales. En raison de l'absence d'ensembles de données limités sur les performances des systèmes [93], la majorité des techniques de détection des anomalies de performance dans la littérature sont de nature non supervisée. Par conséquent, l'approche de détection proposée dans cette thèse est non supervisée.

2.3 Méthodes, approches et outils utilisés

Le centre de données se compose de plusieurs serveurs équipés d'une grande quantité de ressources en termes de CPU, de mémoire, de disque et de bande passante. L'utilisation efficace de ces ressources est la priorité du fournisseur de services en Cloud afin de maintenir la QoS et le coût de la consommation électrique. Ces deux tâches sont les principaux défis de la recherche dans le domaine du *Cloud Computing*. Pour y parvenir, divers travaux de recherche ont été menés avec succès dans le domaine de l'utilisation optimale des ressources dans le centre de données, tout en considérant le placement et la migration des VMs comme des tâches prioritaires. Les fournisseurs de services du *Cloud Computing* doivent fournir une consolidation optimisée des VMs dans le centre de données tout en maintenant les SLAs et le coût de la consommation d'énergie. De nombreuses méthodes ont été proposées en utilisant des solutions d'optimisation heuristiques, méta-heuristiques et basées sur le *Machine Learning*. Dans la plupart des méthodes de ce dernier, des données historiques ou des traces de charge de travail ont été utilisées comme données d'entrée nécessaires pour prédire l'utilisation future des ressources. La prédiction de l'utilisation des ressources nécessite un comportement du système par apprentissage, c'est pourquoi divers algorithmes ont été développés pour faire une prédiction précise pour la migration des VMs. Dans cette section, nous décrivons brièvement les modèles, théories et concepts utilisés dans les contributions proposées dans le cadre de cette thèse. Il s'agit de concepts du *Machine Learning*, tels que l'apprentissage supervisé, des méthaheuristiques tels que l'optimisation de colonie de fourmis, ainsi que des approches mathématiques.

2.3.1 Méthodes utilisées pour le placement des VMs

Le problème du placement des VMs est un problème de recherche largement étudié dans les centres de données, et de nombreux algorithmes ont été proposés et développés pour résoudre ce problème. La plupart de ces algorithmes visent à réduire le nombre des PMs actives afin de limiter la consommation totale d'énergie.

Les techniques de placement de plusieurs VMs se comparent étroitement à celles d'une seule VM. Elles sont généralement réparties en plusieurs sous-problèmes qui sont abordés individuellement. Or, il semble plus approprié de traiter ces sous-problèmes simultanément et de manière globale en les traitant par le même modèle mathématique ; ce type de traitement parallèle conduit à de meilleures solutions que l'agrégation des solutions des sous-problèmes ; ce qui est bien prouvé dans la littérature scientifique pour les problèmes de placement des VMs (section 3.3). Ainsi, nous pouvons considérer le problème de placement comme un problème NP-difficile, et la modélisation du problème de sac à dos peut en être réduit.

Le problème de sac à dos est l'un des problèmes les plus étudiés en recherche opérationnelle. Il consiste à sélectionner un ensemble d'éléments ayant un profit et un poids. L'objectif est de maximiser la somme des profits des éléments choisis sans dépasser la capacité du sac à dos. Plusieurs variantes du problème de sac à dos existe dans la littérature [12]. Par exemple, s'il y a plus d'une contrainte (une limite de volume et une limite de poids, où le volume et le poids de chaque article ne sont pas liés), le problème du sac à dos multidimensionnel est le plus approprié à utiliser. (Notez que le terme "dimension" ne se réfère pas ici à la forme des objets). De même, le problème du sac à dos $0-1$ est largement utilisé ; dans ce type de problème de sac à dos, il n'y a qu'un seul objet de chaque type (ou nous ne pouvons en choisir qu'un seul). Nous n'avons donc que deux options pour chaque objet, soit le choisir (1), soit le laisser (0) sans possibilité de le décomposer, ce qui est le cas de notre travail.

La résolution des problèmes combinatoires NP-difficiles (comme le problème de sac à dos) est relativement complexe, car le nombre limité de solutions possibles s'augmente généralement avec la taille du problème, ainsi que sa complexité. Cela a conduit les chercheurs à développer de nombreuses techniques qui peuvent être classées en deux grandes catégories : les méthodes exactes et les méthodes approximatives qui sont détaillées ci-dessous.

2.3.1.1 Méthodes exactes

Pour obtenir des solutions optimales aux problèmes NP-difficiles, il existe plusieurs techniques exactes, généralement basées sur une recherche complète de l'espace de toutes les combinaisons. Elles sont plus efficaces pour les problèmes de petits ensembles. Cependant, ces techniques exactes sont inefficaces dans le cas des problèmes plus larges, car leur temps de calcul varie de manière exponentielle en fonction de leur taille [42]. Certaines de ces méthodes ont été largement utilisées pour résoudre le problème de sac à dos [156, 154]. La séparation et évaluation progressive (*Branch&Bound*), et la programmation dynamique seront étudiées dans le cadre de cette recherche.

La méthode séparation et évaluation progressive, appartenant à la catégorie des méthodes exactes, repose sur l'énumération des solutions à partir de certaines propriétés du problème en question ; cette technique permet d'éliminer les solutions partielles qui ne conduisent pas à la solution désirée. De ce fait, cette solution est souvent atteinte dans un intervalle de temps raisonnable (problème simple). Dans le pire des cas, nous nous appuyons systématiquement sur l'élimination explicite de toutes les solutions du problème. Pour ce faire, cette méthode comporte une fonction qui permet de fixer une condition préalable pour certaines solutions afin de les exclure ou de les maintenir en tant que solutions potentielles. La performance d'une méthode *Branch&Bound* dépend, entre autres, de la qualité de cette fonction, c'est à dire, sa capacité à exclure les solutions partielles précoces.

Il existe deux implémentations majeures de l'algorithme *Branch&Bound*, *Breadth-First* et *Best-First*. Nous nous concentrons sur la deuxième puisqu'elle est la plus utilisée pour résoudre le problème de sac à dos. La mise en œuvre *Best-First* consiste en un triplet état-espace-arbre qui correspond, respectivement, à l'état initial, l'état cible et les états intermédiaires, et, en plus, des opérateurs de transformation avec des conditions initiales et des conditions finales [73]. La méthode débute par examiner le problème initial avec son ensemble de solutions, appelé la racine. Des procédures de limites inférieure et supérieure sont appliquées à cette racine. Si ces deux limites sont égales, alors une solution optimale est trouvée, et nous nous arrêtons là. Sinon, toutes les solutions sont divisées en deux ou plusieurs sous-problèmes, ce qui en fait des solutions dérivées de la racine. La méthode est ensuite appliquée de manière récursive à ces sous-problèmes, ce qui donne une structure arborescente. Si une solution optimale est trouvée pour un sous-problème, elle est réalisable, mais pas nécessairement optimale, pour le problème initial. Comme elle est réalisable, elle peut être utilisée pour éliminer toutes ses sources. Si la limite inférieure d'un nœud dépasse la valeur d'une solution déjà connue, nous pouvonst alors affirmer que la solution optimale globale ne peut pas être incluse dans le sous-ensemble des solutions représentées par ce nœud. Ainsi, la recherche se poursuit jusqu'à ce que tous les nœuds soient soit explorés, soit éliminés.

La programmation dynamique est un paradigme de conception qui peut être considéré comme une amélioration ou une adaptation de la méthode diviser pour régner. Ce concept a été introduit par Bellman dans les années 1950 comme une méthode exacte pour résoudre des problèmes d'optimisation typiques [113]. L'approche de la programmation dynamique est similaire à la méthode diviser pour régner dans le sens où la solution d'un problème dépend des solutions précédentes obtenues à travers des sous-problèmes. La différence significative entre ces deux méthodes est que la programmation dynamique permet aux sous-problèmes de se chevaucher. En d'autres termes, un sous-problème peut être

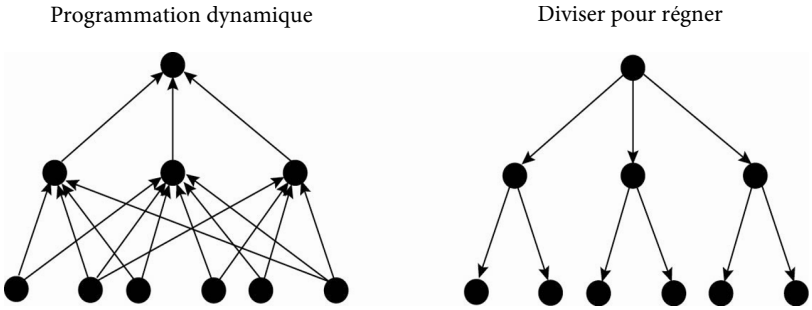


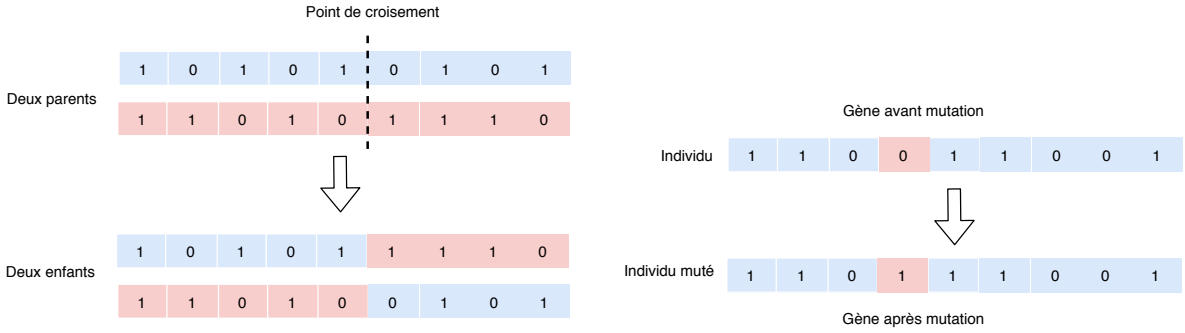
FIGURE 9 – Différence entre la programmation dynamique et la méthode diviser pour régner.

utilisé pour résoudre deux sous-problèmes différents. Alors que l'approche diviser pour régner crée des sous-problèmes qui sont complètement séparés et résolus indépendamment. En résumé, nous pouvons dire que la programmation dynamique suit une approche ascendante, tandis que la méthode diviser pour régner fonctionne en suivant une approche descendante. Le mécanisme de ces deux méthodes peut être illustré et comparé par les deux graphiques arborescents présentés dans la figure 9.

2.3.1.2 Méthodes approximatives

L'objectif de ces méthodes est de trouver une solution acceptable dans un délai raisonnable, mais sans garantir l'optimalité de cette solution. Le principal avantage de ces méthodes réside dans le fait qu'elles peuvent être appliquées à n'importe quelle catégorie de problèmes. En outre, elles ont démontré leur robustesse et leur efficacité face à plusieurs problèmes d'optimisation combinatoire. Elles comprennent deux classes : les heuristiques et les métaheuristiques. Les heuristiques sont des règles empiriques simples basées sur l'expérience, qui ne fournissent pas nécessairement une solution optimale [1, 175]. Une métaheuristique peut être définie comme une méthode algorithmique capable de guider et d'orienter le processus de recherche dans un espace de solutions (souvent très grand) vers un espace plus petit de solutions optimales afin de trouver une solution appropriée, peut-être pas toujours optimale. En tout cas très proche de l'optimum, dans un délai raisonnable [55]. Plusieurs algorithmes métaheuristiques ont été proposés pour résoudre les problèmes NP-difficiles. Les travaux présentés se concentrent sur l'algorithme génétique (AG) et l'optimisation de colonie de fourmis (OCF).

L'algorithme génétique est une méthode évolutive qui s'appuie sur des phénomènes biologiques. À partir d'une solution initiale, nous appliquons notamment certaines techniques génétiques telles que la sélection, le croisement et la mutation pour constituer de nouvelles générations constituant l'ensemble des solutions. Chacune des solutions potentielles dispose d'un certain nombre de propriétés sur lesquelles nous appliquons les opé-



(a) Croisement (b) Mutation

FIGURE 10 – Représentation (a) du croisement et (b) de la mutation dans l'AG

rations précitées. Ainsi, l'apparition de nouvelles générations implique que de nouvelles solutions ont été créées [215]

La sélection permet de conserver les solutions qui, selon les critères établis au préalable, seront les plus aptes à faire émerger de meilleures solutions. Le croisement de l'AG, illustré dans la figure 10a, est l'action de prélever deux nouveaux chromosomes à partir de deux chromosomes existants. Au cours de cette étape, un ou plusieurs points de croisement sont définis, qui permettront ensuite d'achever l'échange de gènes entre les deux chromosomes. La mutation permet d'éviter la convergence rapide de l'AG. La figure 10b décrit le principe de la modification d'un gène choisi au hasard sur un chromosome donné. Ainsi, pour exploiter les résultats obtenus lors des opérations de croisement et de sélection, la mutation ne doit pas être généralisée à tous les chromosomes de la génération obtenue. Les opérations de croisement ou de mutation entraînent une forte probabilité de perte des chromosomes qui ont fourni une solution satisfaisante. Afin de maintenir les meilleurs candidats, nous avons recours à l'élitisme lors du processus de sélection. Cela correspond au passage de ces meilleurs individus à la génération suivante sans subir aucune modification. Dans certains cas, les nouvelles solutions sont irréalisables. Il faut donc évaluer les individus et choisir, selon des critères de faisabilité, celui qui sera admis dans la nouvelle population. Si le nombre d'itérations précédemment fixé est atteint, l'algorithme est arrêté et la meilleure solution est choisie parmi celles qui auraient été générées.

L'optimisation de colonie de fourmis est inspiré du comportement de recherche de nourriture des colonies de fourmis. Ces dernières peuvent résoudre collectivement des problèmes complexes, comme trouver le chemin le plus court entre deux points dans un environnement rigoureux. Pour ce faire, elles communiquent entre elles localement et indirectement, en utilisant une hormone volatile, appelée phéromone : au fur et à mesure de leur progression, les fourmis déposent une trace de phéromone. Elles choisissent ensuite leur chemin de manière aléatoire, selon une probabilité qui dépend de la quantité de phéromone déposée au préalable. Ce mécanisme, qui permet aux fourmis de résoudre

collectivement des problèmes complexes, est à l'origine des algorithmes basés sur les fourmis artificielles. Ces algorithmes ont été initialement proposés dans [4], comme une approche multi-agents pour résoudre des problèmes d'optimisation combinatoire. L'idée est de représenter le problème à résoudre sous forme de recherche d'un meilleur chemin dans un graphe, puis d'utiliser des fourmis artificielles pour trouver de bons chemins dans ce graphe. Le comportement des fourmis artificielles est inspiré de celui des fourmis réelles : elles déposent des traces de phéromone sur les composants du graphe, et elles choisissent leurs chemins par rapport aux traces de phéromone précédemment déposées. Les traces s'évaporent au fil du temps. Intuitivement, cette communication indirecte fournit des informations sur la qualité des trajectoires utilisées pour attirer les fourmis, lors des futures itérations, vers les zones correspondantes de l'espace de recherche. Les caractéristiques du comportement des fourmis artificielles définissent le "métaheurisme de l'optimisation par une colonie de fourmis" ou algorithme OCF. Cette approche métaheuristique a permis de résoudre divers problèmes d'optimisation combinatoire, tels que le problème de sac à dos [72].

2.3.2 Méthodes utilisées pour la migration en temps réel des VMs

La migration en temps réel est l'une des technologies clés pour améliorer l'utilisation, l'efficacité énergétique et la maintenance des centres de données. Divers algorithmes de migration en temps réel ont été proposés, chacun présentant des caractéristiques distinctes en termes de temps de complétion, de quantité de données transférées, de temps d'arrêt et de dégradation des performances des VMs. De plus, non seulement l'algorithme de migration mais aussi les applications s'exécutant dans la VM migrée affectent les différentes mesures de performance. Avec les SLAs et les contraintes opérationnelles mises en place, le choix de la technique de migration optimale en temps réel est un problème compliqué et nécessite des techniques modernes pour le résoudre.

Une solution potentielle pour améliorer le processus de migration en temps réel est de concevoir un modèle autonome capable de prendre des décisions basées sur des informations prises précédemment. Les algorithmes de *Machine Learning* ont été largement utilisés dans plusieurs secteurs, et récemment, une tendance vers l'intégration des techniques du *Machine Learning* a été constatée afin d'améliorer l'efficacité globale des centre de données (section 6).

Les méthodes du *Machine Learning* visent généralement à apprendre des modèles à partir de données collectées précédemment et à appliquer ces modèles à des données nouvelles et peu visibles. Ainsi, les méthodes du *Machine Learning* comprennent généralement

deux phases principales : une phase d'entraînement où un modèle est construit à l'aide de données de formation et une phase d'inférence (test) où le modèle est appliqué [219, 203].

Les méthodes du *Machine Learning* peuvent être classées en méthodes supervisées et non supervisées. Les méthodes non supervisées n'utilisent généralement que les données d'entrée et peuvent être utilisées pour des applications telles que le regroupement, où le nombre ou la nature des classes n'est pas connu à l'avance. Cependant, les méthodes supervisées utilisent des ensembles de données labellisées où chaque élément de données d'entrée a une étiquette correspondante. Ces méthodes peuvent être utilisées pour des applications telles que la classification d'un objet dans l'une des nombreuses classes ou la régression. Pour les méthodes d'apprentissage supervisées, disposer d'une plus grande quantité de données de formation signifie que nous pouvons construire un meilleur modèle qui peut générer des prédictions avec une plus grande précision [219, 203].

Dans cette thèse, nous proposons un modèle de migration en temps réel des VMs basé sur plusieurs algorithmes de régression qui est une technique de prédiction dont le but est de déterminer une relation entre les entrées et les sorties en utilisant une fonction continue pour aider les systèmes à comprendre comment les sorties changent pour des entrées données.

2.3.2.1 Méthodes de régression linéaire

Le problème abordé dans cette thèse peut être considéré comme un problème de régression linéaire multiple. Considérons un ensemble d'observations $X = \{x_1, ..., x_i, ..., x_n\}$; X est une n×p-matrice contenant les valeurs de p variables prédictives pour chacun des n points de données. Nous définissons un vecteur de n-colonnes Y contenant les valeurs observées correspondantes des variables cibles y_i. Nous supposons que ces dernières sont modélisées en fonction des variables prédictives x_i. Le modèle standard de régression linéaire multiple est donné par :

$$y_i = \beta_0 + \beta_1 x_{i,1} + ... + \beta_p x_{i,p} + \epsilon_i, i = 1, ...n \tag{2.1}$$

Où β se réfère au (p+1)×1-vecteur des valeurs de la population liées aux paramètres ; et ϵ est un n×1-vecteur du biais expérimental. Nous supposons en outre que les valeurs de ϵ sont des termes d'erreur indépendants et identiquement distribués normalisées et non corrélées avec $E(\epsilon) = 0$ et $E(\epsilon\epsilon^T) = \sigma^2 I$. La forme matricielle de ce modèle est la suivante :

$$y = X\beta + \epsilon \tag{2.2}$$

$$\text{Avec } y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, X = \begin{pmatrix} 1 & x_{1,1} & \cdots & x_{1,p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n,1} & \cdots & x_{n,p} \end{pmatrix}, \beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix} \text{ et } \epsilon = \begin{pmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \end{pmatrix}$$

Régression *Ridge*

Dans un tel modèle de régression, l'estimateur des moindres carrés peut être le meilleur estimateur non biaisé de β [220]. Toutefois, dans les cas où des corrélations entre les variables prédictives se produisent, les valeurs propres de la matrice $X^T X$ tendent à se rapprocher de zéro et l'estimateur des moindres carrés n'est plus satisfaisant en raison de la variance élevée [153]. Pour contrôler la variance de l'estimateur, un moyen simple est de pénaliser ses valeurs élevées. C'est l'objectif fondamental de la régression *Ridge* [152], qui consiste à optimiser le problème suivant :

$$\arg\min_{\beta \in \mathbb{R}^{p+1}} \|y - X\beta\|^2 + l \sum_{j=0}^{p} |\beta_j|^2 \tag{2.3}$$

où $l > 0$ est un paramètre à déterminer.

où l'estimateur peut être désigné comme suit :

$$\tilde{\beta}_l^{ridge} = (X^T X + lI)^{-1} X^T y \tag{2.4}$$

Régression LASSO

Pareillement, la régression *Least Absolute Shrinkage and Selection Operator* (LASSO) repose sur le même concept de régression *Ridge*, mais elle applique la norme de *Manhattan* ($L1$) pour la régularisation de la pénalité au lieu de la norme euclidienne ($L2$) [147]. La méthode de LASSO est principalement utilisée pour construire un estimateur épars (certaines composantes sont nulles) et peut être définie comme suivant :

$$\tilde{\beta}_l^{\text{LASSO}} = \arg\min_{\beta \in \mathbb{R}^{p+1}} \|y - X\beta\|^2 + l \sum_{j=0}^{p} |\beta_j| \tag{2.5}$$

Les régressions *Ridge* et LASSO utilisent le mécanisme de rétrécissement pour les modèles de régression linéaire. Les deux techniques assignent une contrainte aux paramètres du modèle pour que les facteurs de régression de certaines variables soient abaissés à zéro ; l'objectif est de fournir le sous-ensemble de variables prédictives qui minimise le biais de prédiction pour une variable de réponse quantitative donnée. Après le processus de réduction, les variables ayant un coefficient de régression égal à zéro sont exclues du modèle, et les variables ayant des coefficients de régression différents de zéro sont plus étroitement

liées à la variable cible [227].

2.3.2.2 Méthodes de régression non linéaire

Lors d'une régression non linéaire, la combinaison des paramètres du modèle et la dépendance vis-à-vis les variables indépendantes ne sont pas linéaires. Contrairement à la régression linéaire, il n'existe pas d'expression générique de forme fermée pour trouver un ajustement optimal des paramètres pour une famille de fonctions donnée. On se place donc dans le cadre d'un modèle de régression :

$$y_i = f(x_i) + \epsilon_i \tag{2.6}$$

En l'absence de toute hypothèse sur la fonction de régression f, nous sommes dans un cadre non paramétrique. Ainsi, nous proposons trois méthodes d'apprentissage pour la fonction f.

Régression à vecteurs supports

Les machines à vecteurs supports (SVM) s'étendent aux problèmes de régression non linéaire - cette méthode est appelée régression à vecteurs supports, *Support Vector Regression* (SVR) [3]. Au lieu d'une valeur entière de y_i, les étiquettes prennent une valeur réelle.

Supposons que l'on nous donne les notations indiquées ci-dessus. Un modèle de régression est entraîné par ces entrées et utilisé pour prédire les valeurs cibles futures. La SVR est une méthode de régression non linéaire qui vise à trouver le meilleur hyperplan de régression avec le plus petit risque structurel dans un espace de caractéristiques à haute dimension [44].

L'un des types de SVR les plus populaires est le ε-SVR qui situe l'hyperplan par une fonction de perte indépendante du ε [228]. La fonction de SVR est définie comme ci-dessous :

$$f(x_i) = \prec w, x_i \succ + b \tag{2.7}$$

où w est un vecteur de coefficients de poids et b est une constante de biais. Les valeurs de w et b sont estimées en minimisant le problème d'optimisation suivant :

$$min \frac{1}{2} \|w\|^2 \tag{2.8}$$

s.c

$$\begin{cases} y_i - (\prec w, x_i \succ + b) \leq \varepsilon \\ (\prec w, x_i \succ + b) - y_i \leq \varepsilon \end{cases} \tag{2.9}$$

Pour résoudre le problème de la faisabilité et pour rendre la méthode plus fiable, les points de la marge ε restent présents. En revanche, ces points sont pénalisés par l'introduction de variables flottantes ξ_i et ξ_i^* [164].

$$min\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \tag{2.10}$$

s.c

$$\begin{cases} yi - (\prec w, x_i) \succ +b) \leq \varepsilon + \xi_i \\ (\prec w, x_i \succ +b) - yi \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases} \tag{2.11}$$

où la constante de coût $C > 0$ détermine le compromis entre la complexité du modèle et l'erreur de l'entraînement.

SVR avec *Bagging*

Pour améliorer l'apprentissage par SVR, la notion d'ensemble de régression est introduite, il s'agit de plusieurs décisions individuelles combinées d'une certaine manière (généralement par combinaison pondérée ou non pondérée) pour prédire de nouveaux exemples. L'un des traitements de combinaison les plus répandus pour la SVR est le *Bagging* qui consiste à manipuler l'ensemble d'entraînement. L'algorithme d'apprentissage est exécuté plusieurs fois, chacune avec un sous-ensemble (ou une distribution) différent des exemples d'entraînement. Cette technique fonctionne particulièrement pour les algorithmes d'apprentissage instables (des algorithmes dont la régression de sortie subit un changement majeur en réponse à de petits changements dans les données d'entraînement) [15].

À chaque cycle d'exécution, le *Bagging* fournit l'algorithme d'apprentissage avec un ensemble d'éléments d'entraînement qui consiste en un échantillon d'exemples tirés au hasard avec remplacement de l'ensemble d'éléments d'entraînement d'origine (soient T ensembles). Un tel ensemble est appelé une réplique *bootstrap* de l'ensemble d'entraînement original [202]. Ainsi, la solution adoptée pour la SVR avec *Bagging* est la suivante :

$$SVR_{Bagging}(x_i) = \frac{1}{T}\sum_{t=1}^{T} SVR_t(x_i) \tag{2.12}$$

K plus proches voisins

De même, l'algorithme du K plus proches voisins, K-Nearest Neighbors (KNN), est une approche d'apprentissage supervisé non-paramétrique ; il est, comme la SVR, utilisé à la fois pour la classification et la régression. La règle de base de KNN, introduite par [157], maintient l'ensemble d'entraînement grâce à un processus d'apprentissage et attribue à

chaque objet une classe caractérisée par les étiquettes les plus importantes de ses voisins les plus proches. Le mécanisme de l'approche KNN est le suivant : nous donnons à un objet x_i comme vecteur p-dimensionnel à classer. L'algorithme KNN sélectionne les K voisins x_k les plus proches de x_i, en fonction d'une métrique qui mesure la similarité entre l'objet de la requête et les modèles existants. Diverses mesures de distance ont été utilisées par convention pour évaluer la similarité entre l'élément x_i et x_k ; les distances largement adoptées sont les suivantes :

$$\begin{cases} d(x_i, x_k) = \left(\sum_{j=1}^{p} \left\| x_{i,j} - x_{k,j} \right\|^2\right)^{\frac{1}{2}} & \text{Distance Euclidienne} \\ d(x_i, x_k) = \sum_{j=1}^{p} \left\| x_{i,j} - x_{k,j} \right\| & \text{Distance de Manhattan} \\ d(x_i, x_k) = \sum_{j=1}^{p} \left\| x_{i,j} - x_{k,j} \right\|^2 & \text{Distance Euclidienne Carrée} \\ d(x_i, x_k) = max\left(\sum_{j=1}^{p} \left\| x_{i,j} - x_{k,j} \right\|\right) & \text{Distance de Chebychev} \end{cases}$$

Pour la régression, KNN est adapté en attribuant la valeur de la propriété à l'élément comme la moyenne des résultats de ses K plus proches voisins comme suivant :

$$y_i = \frac{1}{K} \sum_{k=1}^{K} y_k \tag{2.13}$$

Où y_i est la valeur prédite pour le motif x_i, et y_k correspond à la désignation de la valeur prédite associée à x_k, en notant que ce dernier est un élément des K voisins les plus proches sélectionnés.

2.3.3 Méthodes utilisées pour la détection des anomalies

Dans cette thèse, nous nous concentrons sur l'utilisation du *Fuzzy Clustering* pour la détection et la classification des anomalies. Les notions de cluster et de *Clustering* peuvent avoir des significations différentes. Dans cette étude, nous nous intéressons aux algorithmes de *Clustering* (basés sur les partitions) qui peuvent être considérés comme une fonction mettant en correspondance des formes (ou des observations) dans un ensemble de données fini, multi-varié et non étiqueté, X avec des partitions dans $\mathbb{P}$, l'ensemble de toutes les partitions compatibles avec la dimension X. Le problème est de partitionner l'espace $X \in \mathbb{R}^d$ en groupes (clusters) de sorte que les données d'un même groupe soient similaires les unes aux autres et aussi différentes que possible des données des autres groupes. Les dissemblances sont évaluées grâce à une fonction de distance appropriée qui satisfait aux trois propriétés : réflexivité, symétrie et inégalité triangulaire.

Essentiellement, le *Fuzzy Clusteirng)* diffère du *Clustering* conventionnel (dur) en sens qu'il permet à une observation d'appartenir, avec différents degrés d'appartenance, à plus

d'un cluster [166]. Chaque degré d'appartenance peut exprimer de manière ambiguë ou définitive l'appartenance d'une observation à un cluster donné et, sous des contraintes appropriées, peut être interprété comme la probabilité qu'une observation soit membre d'un cluster. Il existe actuellement une multitude d'algorithmes de *Clustering*. Ce qui suit se concentre uniquement sur l'algorithme utilisé dans cette thèse.

2.3.3.1 *Fuzzy C-means* pondérée

Le *Fuzzy C-means* pondérée, *Weighted Fuzzy C-means*(WFCM), est un algorithme de *Clustering* développé dérivant de l'algorithme *Fuzzy C-means* (FCM) (proposé par [2]), qui est est l'algorithme de *Clustering* le plus populaire et le plus largement utilisé dans la littérature.

WFCM consiste à associer, à chaque élément de données, une valeur de pondération qui définit la pertinence relative de l'élément tout en construisant la solution du *Clustering*. Pour un ensemble donné d'objets $O = \{o_1, o_2, ..., o_N\}$, nous associons un ensemble d'observations $X = \{x_1, x_2, ..., x_N\}$ en $\mathbb{R}^d$. x_i représente la valeur caractéristique décrivant l'objet o_i. WFCM vise à minimiser la fonction objective suivante :

$$Q_{WFCM} = \sum_{k=1}^{C} \sum_{i=1}^{N} w_i u_{ki}^m ||x_i - v_k||_A^2 \tag{2.14}$$

Sachant que les contraintes $\sum_{i=1}^{N} u_{ki} > 0$ et $\sum_{k=1}^{C} u_{ki} = 1$ doivent être respectées. w_i est le poids correspondant de chaque élément x_i, en notant que les objets ayant un poids élevé ont une plus grande influence sur le processus du *Clustering* que les objets à faible poids. $m > 1$ est le paramètre dit de *Fuzziness* qui régule l'influence des membres de l'équation. u_{ki} indique l'appartenance de l'observation x_i au $k^{ème}$ cluster , où $k = 1, ..., C$ et C est le nombre de clusters. u_{ki} prend des valeurs dans [0,1], 0 correspond à la non-appartenance tandis que 1 se réfère à l'adhésion totale. Les vecteurs u_i sont disposés en une C×N-matrice $U = [u_{ki}]_{C \times N}$. Ainsi que les u_{ki} sont calculés comme suit :

$$u_{ki} = \frac{1}{\sum_{j=1}^{C} (\frac{||x_i - v_k||}{||x_i - v_j||})^{\frac{2}{m-1}}} \tag{2.15}$$

De plus, v_k désigne le centroïde du $k^{ème}$ cluster et peut être mis à jour par l'équation suivante :

$$v_k = \frac{\sum_{i=1}^{N} w_i u_{ki}^m x_i}{\sum_{i=1}^{N} w_i u_{ki}^m} \tag{2.16}$$

On remarque que le prototype (centre) du cluster v_k est une somme pondérée des vecteurs de caractéristiques, puisque chaque objet x_i a un impact prédéterminé différent donné par un poids relatif w_i. La détermination du poids associé à chaque observation peut être effectuée selon différentes stratégies. Nous citons, entre autres, l'utilisation d'informations sur la densité de voisinage de chaque échantillon (le nombre d'objets qui sont proches de l'échantillon en utilisant un seuil de distance) ou des constantes définies par l'utilisateur [169]. Ainsi, l'optimisation de la fonction objectif 2.14 par un processus itératif construit les partitions floues. Dans cette thèse, nous proposons une nouvelle technique de détermination du poids. Cette technique consiste à imiter le comportement collectif des oiseaux étourneaux pour définir le poids associé à chaque élément de données.

2.3.3.2 Le comportement collectif des étourneaux

Le mouvement collectif de groupes d'animaux est l'un des phénomènes les plus observés dans la nature. Ces mouvements collectifs résultent d'interactions locales entre individus et sont soutenus par la formation de structures spatiales et temporelles à grande échelle. Pour comprendre les caractéristiques de ces mouvements collectifs, il est important de caractériser la dynamique des interactions entre les individus. Pour ce faire, de nombreux chercheurs ont étudié une espèce d'oiseau, les étourneaux, afin de développer un algorithme qui analyse les trajectoires individuelles et détermine pour chaque oiseau, et à chaque instant, combien de ses voisins influencent son mouvement [54, 87]. De même, en 1986, un informaticien, *Craig Reynolds*, a mis en place des règles qui simulent le comportement de nuages d'oiseaux, il a nommé ces oiseaux virtuels "boids" (pour "oiseaux") [223].

Selon l'étude de [222], chaque oiseau réagit à ce qui l'entoure, et seulement à cela. Cela signifie que son comportement peut être modélisé de manière à ce que chaque oiseau ne réagisse qu'à ses voisins. La compréhension des mécanismes de traitement collectif de l'information chez les oiseaux vivant en groupe ouvre des perspectives pour le développement d'algorithmes bio-inspirés pour le contrôle distribué de systèmes artificiels tels que les réseaux routiers automatisés [7], la reconnaissance robotique coopérative [159], le contrôle de la manipulation [9], le contrôle de vol basé sur le groupe [6], le déploiement de réseaux de capteurs distribués [10], etc.

Dans ce travail, nous suivons les simulations de Reynolds qui consistent en ce qui suit. Pendant le vol des étourneaux, la direction du mouvement de chaque étourneau dépend non seulement de l'auto-information mais aussi des informations fournies par d'autres étourneaux proches. Un étourneau réagit à ses voisins proches dans le troupeau et adapte progressivement sa direction de vol pour suivre la direction de ses congénères, comme le montre la figure 11. L'intégration du comportement des étourneaux dans le processus de

l'algorithme WFCM est détaillée dans le chapitre 7.

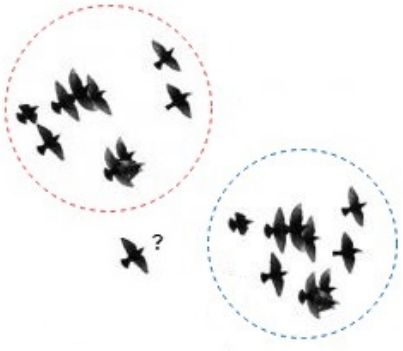


FIGURE 11 – Un exemple simplifié du comportement collectif des étourneaux.

2.3.4 Outil de simulation

La mesure du coût d'une infrastructure en Cloud à grande échelle est importante. Cependant, il est difficile de gérer des expériences de ce type sur une infrastructure réelle, surtout lorsqu'il est essentiel de reproduire les expériences dans les mêmes conditions (par exemple, en comparant différents scénarios). C'est pourquoi des simulations ont été choisies pour évaluer les modèles proposés dans cette thèse.

CloudSim[11] est l'un des frameworks de modélisation et de simulation des environnements de *Cloud Computing* les plus répandus. Il a été développé en 2010 par le laboratoire *CLOUDS* du département d'informatique et de génie logiciel de l'université de Melbourne (Australie), et il est utilisé dans la recherche par plusieurs universités et organisations, notamment l'université Duke de Caroline du Nord, *HP Labs* à *Palo Alto*[12] et le *National Research Center for Intelligent Computing System* (NCIC) en Chine. Selon [183], *CloudSim* est l'outil de simulation le plus populaire disponible pour les environnements de *Cloud Computing*. Sa mise en oeuvre en langage *Java* suivant la programmation orienté objet en fait un simulateur hautement évolutif. En outre, *CloudSim* est un logiciel à licence libre, de sorte que tout utilisateur peut utiliser l'intégralité de son code source et participer à son développement. Le simulateur permet de modéliser les composants du processeur, la mémoire vive, le stockage, la VM, l'hôte, le centre de données, ainsi que la politique d'allocation des VMs, le planificateur des VMs, la charge de travail dynamique, la simulation en fonction de la puissance et la modélisation du réseau, etc. Quant à l'architecture du

11. CloudSim : http://www.cloudbus.org/cloudsim/
12. Palo Alto : https://www.paloaltonetworks.com/

framework *CloudSim* (voir figure 12), trois couches principales ont été définies ci-dessous (description ascendante) :

- **Le moteur de simulation de base** : permet de modéliser et simuler un centre de données et l'environnement du *Cloud Computing*, et comprend des interfaces de gestion spécifiques aux VMs, la mémoire, la mise en œuvre des applications et la surveillance dynamique de l'état du système. Un prestataire de *Cloud Computing* doit mettre en œuvre ses politiques au niveau de cette couche afin de vérifier l'efficacité des différentes stratégies qui permettent l'attribution et le déplacement des VMs entre ses hôtes.
- **Les librairies de modélisation** : cette couche définit les entités de base pour les machines hôtes (telles que le nombre de machines et leurs spécifications), les applications (telles que le nombre des tâches et leurs exigences), les VMs, le nombre d'utilisateurs, le domaine des applications et les stratégies de planification des brokers. Un concepteur de *Cloud Computing* peut effectuer plusieurs activités, y compris des scénarios de mise en disponibilité de modèles de Cloud, mener des tests de validation en fonction de certaines configurations personnalisées prises en charge par *CloudSim* et mettre en œuvre des techniques de provisionnement d'applications adaptées dans le Cloud.
- **Le système de configuration des utilisateurs** : Cette couche inclut des scénarios, spécifications des infrastructures et des ressources, configurations des applications et de la charge de travail, déclarations des politiques d'allocation, de placement, et de migration.

L'outil *CloudSim* a été choisi comme plate-forme de simulation appropriée car elle permet la modélisation de l'environnement IaaS virtualisé et est à la base de nombreuses recherches dans le domaine des capacités de *Cloud Computing*, en particulier pour la conservation de l'énergie et l'allocation des ressources. *CloudSim* émule les caractéristiques du *Cloud Comuting* à travers ses packages et ses classes [43]. Il y a au total 12 packages dans *CloudSim* contenant plusieurs classes. De tous les packages, le plus pertinent, pour nous, est *org.cloudbus.cloudsim*. Ce package contient les codes pour la modélisation des différentes entités du Cloud comme un centre de données, un hôte, un Cloudlet et des VMs, différentes politiques de placement et de migration et différents modèles de consommation d'énergie. Il est possible d'étendre ou d'écraser ces classes pour définir de nouvelles entités de Cloud, ainsi que de modifier celles qui existent déjà et de créer de nouvelles politiques. En outre, en utilisant les mesures de performance de *CloudSim*, nous comparons les algorithmes de cette plate-forme (et d'autres techniques existantes) avec les approches qu'on propose. Ces mesures montreront l'efficacité de chaque algorithme, par exemple, en termes d'optimisation d'énergie et de violations des SLAs. Dans cette thèse, nous nous

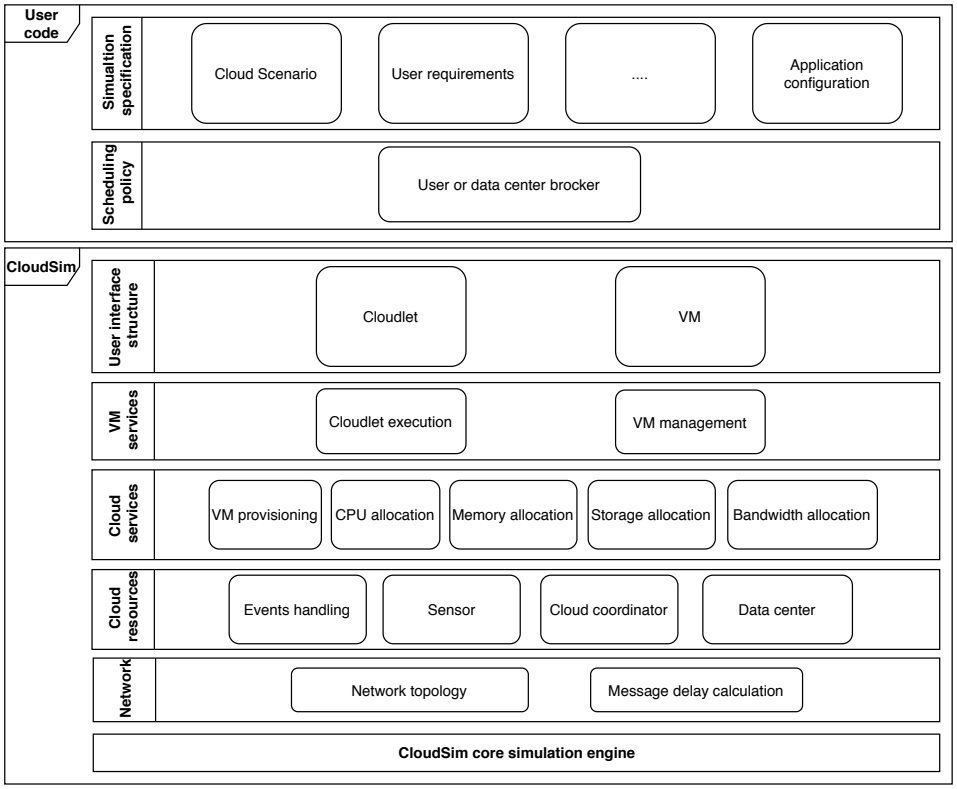


FIGURE 12 – Représentation de l'architecture de *CloudSim* [43].

intéressons principalement aux classes "*VMallocationPolicy.java*" et "*PowerModel.java*", ainsi, chaque outil *CloudSim* utilisé sera détaillé dans le chapitre correspondant.

2.4 Conclusion

Le *Cloud Computing* est une évolution majeure de l'informatique industrielle, associée à la délocalisation des services. Pour opérer, le *Cloud Computing* exige des infrastructures techniques composées de milliers de machines ; il s'agit des centres de données. Ces infrastructures consomment de l'énergie et posent une problématique à la fois technique et éthique intéressante.

De ce fait, de nombreux travaux se sont focalisés sur la problématique d'efficacité énergétique. Ces travaux sont très divers, de l'estimation de la puissance à l'optimisation des ressources utilisées par les MVs, en se basant sur la planification et la consolidation de ces dernières au sein des PMs. Dans le cadre de cette thèse, nous cherchons des solutions à cette problématique importante en nous focalisant sur le placement et la migration en temps réel des VMs, ainsi que le diagnostique de l'environnement virtualisé. Comme détaillé dans ce chapitre, il existe une variétés d'approches de gestion et de consolidation des VMs dans un centre de données. Pour les améliorer, plusieurs algorithmes et concepts sont

adoptés. Notamment, les algorithmes bio-inspirés et les techniques de *Machine Learning* que nous utilisons dans notre thèse.

3

Revue critique de la littérature

Ce chapitre présente une synthèse exhaustive des recherches effectuées précédemment dans le cadre des techniques abordées dans notre thèse, y compris les modèles d'estimation de la consommation énergétique, les modèles de placement et de migration en temps réel des VMs, ainsi que les techniques de détection des anomalies dans un centre de données

---



Microsoft : https://www.presse-citron.net/premier-succes-pour-microsoft-qui-songe-a-plonger-ses-datacenters-au-fond-des-oceans/

3.1 Introduction

L'évaluation de la consommation de l'énergie électrique est une étape principale pour améliorer l'efficacité énergétique. Les mesures de la consommation peuvent être effectuées de deux différentes manières. La première consiste à utiliser des dispositifs qui mesurent la puissance électrique fournie aux différentes PMs sur lesquelles une VM ou une application (dans le cas d'un HPC) est exécutée. La deuxième approche est basée sur l'estimation de la consommation de l'énergie à travers des modèles théoriques, généralement liés à la performance. Souvent, les mesures de la puissance électrique obtenues par les outils physiques sont utilisés pour vérifier l'exactitude des méthodes d'estimation. Dans ce travail, on se limite à ces dernières.

Plusieurs études visent à estimer la consommation de l'énergie en concevant des modèles. Selon des études récentes, les modèles de consommation énergétique sont établis pour les objectifs suivants :

- Pour estimer le potentiel de la gestion de ressources énergétiques ;
- Pour prédire la consommation électrique d'un serveur ou d'un centre de données ;
- Et pour répondre aux exigences du centre de données (c'est-à-dire réduire la consommation de l'énergie tout en fournissant une grande fiabilité) ;

Les modèles de la consommation énergétique comprennent le fonctionnement des PMs et des VMs, et peuvent être classés en modèles additifs, modèles BA (puissance de base + puissance active) et autres modèles en considérant la formule de calcul et certains autres facteurs comme discuté dans la section 3.2.

D'autre part, les problèmes de placement des VMs dans les centres de données font partie des problèmes d'allocation des ressources. Ce chapitre explore les aspects associés aux approches actuelles de placement des VMs conçues en termes d'efficacité énergétique. Comme nous l'avons déjà évoqué, il existe deux types d'approches de placement des VMs : le placement statique (ou initial) et le placement dynamique. Dans les deux cas, une optimisation doit être effectuée simultanément par le système. Différentes stratégies sont à utiliser pour ces deux types de placement en fonction de leurs différentes caractéristiques. Généralement, le problème de placement des VMs peut être posé comme un problème d'optimisation à contraintes. Il s'agit d'un problème de *Bin-Packing* ou un problème de sac à dos (*Knapsack*) [150, 142]. En tant que type spécial de problème d'optimisation, le problème de *Bin-Packing* vise à empaqueter des objets de volumes différents dans le plus petit nombre possible de bacs de taille identique ou différente. De même, le problème du sac à dos consiste à sélectionner un ensemble d'objets qui sont associés à un profit et à un poids ; l'objectif est de maximiser la somme des profits des objets choisis sans dépasser

la capacité du sac à dos. Des études pertinentes portant sur les problèmes de placement statique et dynamique sont examinées dans la section 3.3.

Par ailleurs, la migration en temps réel est une stratégie cruciale pour la gestion des ressources dans uns centre de données. En effet, les utilisateurs des services en Cloud demandent et utilisent des ressources auprès d'un fournisseur, et abandonnent après avoir terminé leurs tâches. En conséquence, les fournisseurs de services créent et détruisent constamment des VMs au sein du centre de données. Sans une gestion efficace, cette infrastructure fonctionnera ainsi : de nombreux serveurs fonctionnent avec une faible charge de travail, tandis que d'autres sont surchargés, ce qui mène à une consommation dramatique de l'énergie. La migration en temps réel consiste à faire en sorte qu'une VM ne soit plus fixée sur le serveur où elle est initialement placée. Une VM peut être déplacée d'un serveur à un autre, voire d'un centre de données à un autre, sans provoquer d'interruption des applications qui y tournent. De nombreux modèles de migration en temps réel des VMs ont été proposés dans la littérature et sont adressés dans la section 3.4.

Les techniques de gestion des ressources dans le centre de données (comme le placement et la migration en temps réel des VMs) restent cependant peu fonctionnelles en cas de variations inconnues ou inattendues du système. Les variations de performances (telles que l'étranglement des processeurs dû à la chaleur ou les processus orphelins) entraînent des fluctuations de performances, un gaspillage des ressources de calcul et une faible planification , sans nécessairement provoquer de défaillances [80]. Au cours de la dernière décennie, on a constaté un intérêt croissant pour la construction d'outils de détection automatique des anomalies de performance pour les systèmes informatiques à grande échelle, tel qu'un centre de données [93]. Les approches actuelles de détection des anomalies peuvent être divisées en trois catégories : les méthodes basées sur des normes, l'analyse statistique des séries temporelles et les solutions basées sur le *Machine Learning*. Vue que les deux premières catégories sont hors porté de cette thèse, on se limite à invoquer, dans la section 3.5, des modèles basées sur le *Machine Learning* pour détecter les anomalies sur les systèmes Cloud et HPC.

3.2 Modèles d'estimation de la consommation énergétique

3.2.1 Modèles additifs

La figure 14 illustre la répartition de la consommation d'énergiee d'un serveur selon [148]. On peut constater que le CPU représente la plus importante source de consommation de la puissance électrique, suivie par les connecteurs d'extension (y compris le réseau

et les dispositifs d'entrée/sortie), le bloc d'alimentation, la mémoire, la carte mère, les disques et le stockage et le ventilateur. Par conséquent, la méthode la plus simple pour modéliser la consommation totale de la puissance consiste à additionner les puissances de tous les composants. L'expression est la suivante :

$$P_{total} = P_{CPU} + P_{mémoire} + P_{disque} + ... + P_{E/S} \tag{3.1}$$

où P_{CPU}, $P_{mémoire}$, P_{disque}, $P_{E/S}$ représentent respectivement la consommation du CPU, de la mémoire, du disque et des dispositifs d'entrée/sortie, sachant que chacun de ces éléments a également son propre modèle de consommation énergétique. Par exemple, la consommation électrique du CPU est liée à la tension, à la fréquence et à un taux d'utilisation ; la consommation électrique de la mémoire est liée à sa taille, à sa fréquence et à son niveau d'utilisation ; la consommation électrique du disque dur est en fonction de la vitesse de lecture et d'écriture ; et la consommation électrique du ventilateur est en fonction de la largeur, de la profondeur et de la vitesse de rotation. Néanmoins, les principaux composants consommateurs de l'énergie électrique peuvent posséder des aspects légèrement différents selon les configurations des serveurs, ainsi que la proportion de la consommation électrique de chaque composant n'est pas toujours constante, comme le montre la figure 13 [48]. Certaines recherches [185, 38] ont considéré le CPU, la mémoire, le disque dur, la bande passante, la carte mère et le ventilateur comme étant les principaux composants du modèle de consommation de la puissance, tandis que d'autres études ont considéré le CPU et la mémoire les composants les plus consommateurs de l'énergie, les autres composants ont une portée de consommation faible ou présentant une forte corrélation avec l'activité du CPU.

3.2.2 Modèles BA

Dans un centre de données, les serveurs ne sont pas toujours actifs ; il s'agit de serveurs en état de veille (inactifs). Ainsi, la consommation électrique des contreparties peut être divisée en deux parties : (1) Une puissance de base (P_{base}), qui correspond à la consommation électrique lorsque la machine est au point de veille, comprend la consommation électrique des ventilateurs, du processeur, de la mémoire, des dispositifs d'entrée/sortie et des autres composants de la carte mère à l'état de veille ; P_{base} est souvent considérée en tant que valeur constante. (2) Une puissance active (P_{active}), qui désigne la consommation de la puissance causée par la charge de travail, et qui dépend de la tâche du serveur et de la façon dont le CPU, la mémoire et les composants d'entrée/sortie sont utilisés. Le modèle de puissance peut donc être exprimé par la somme de la puissance de base et de

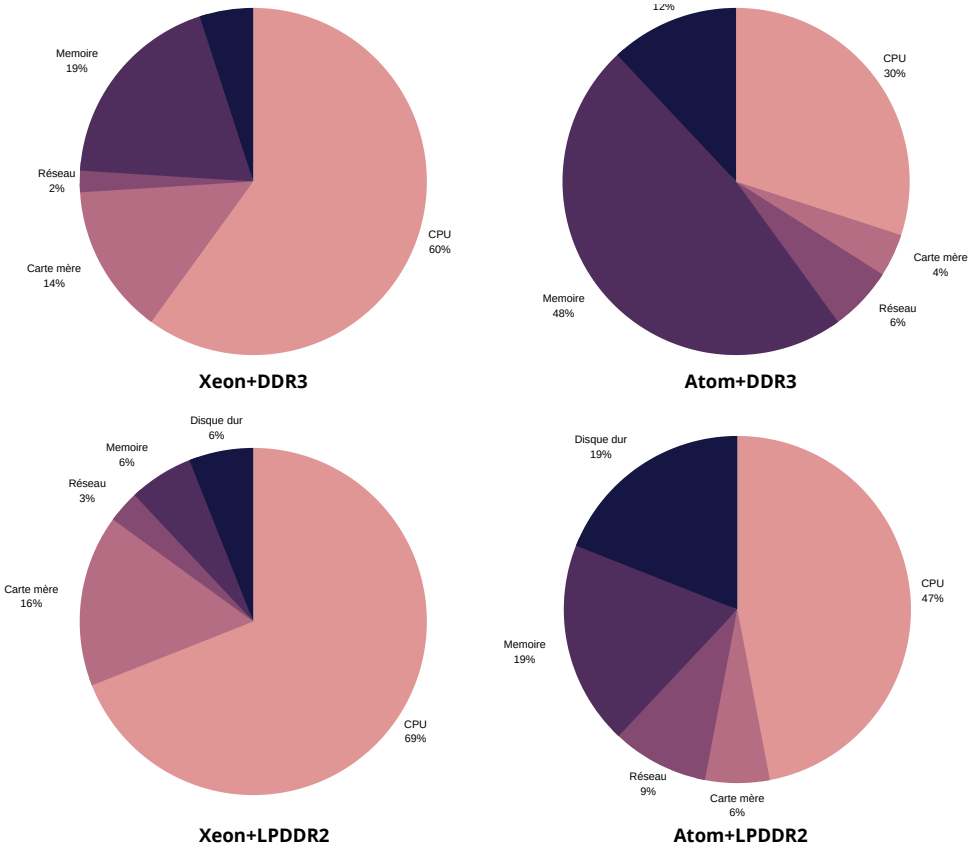


FIGURE 13 – Répartition de la puissance entre les composants des serveurs pour différentes configurations.

la puissance active comme suivant [69] :

$$P_{BA} = P_{base} + P_{active} + \Delta \tag{3.2}$$

où, Δ est le terme de rectification, qui peut être soit une valeur fixe, soit une expression. En outre, P_{base} peut être considéré comme un terme constant, et P_{actif} peut être exprimé sous plusieurs formes (par exemple, une fonction linéaire, une fonction puissance, un polynôme à haut degré, etc.). Ainsi, le modèle BA peut être défini comme étant un modèle de régression linéaire, un modèle de fonction puissance, un modèle non linéaire ou un modèle polynomial. En particulier, du point de vue de la génération et de la dissipation de chaleur, la consommation électrique du serveur comprend, non seulement les composants informatiques, mais aussi le système de refroidissement [101, 104].

1. **Modèle de régression linéaire** : les modèles de régression linéaire comprennent les modèles de régression simple et les modèles de régression multiple. [201] a été le précurseur à concevoir des modèles d'estimation basés sur la corrélation entre la consommation de la puissance et la performance des compteurs qui captent l'ac-

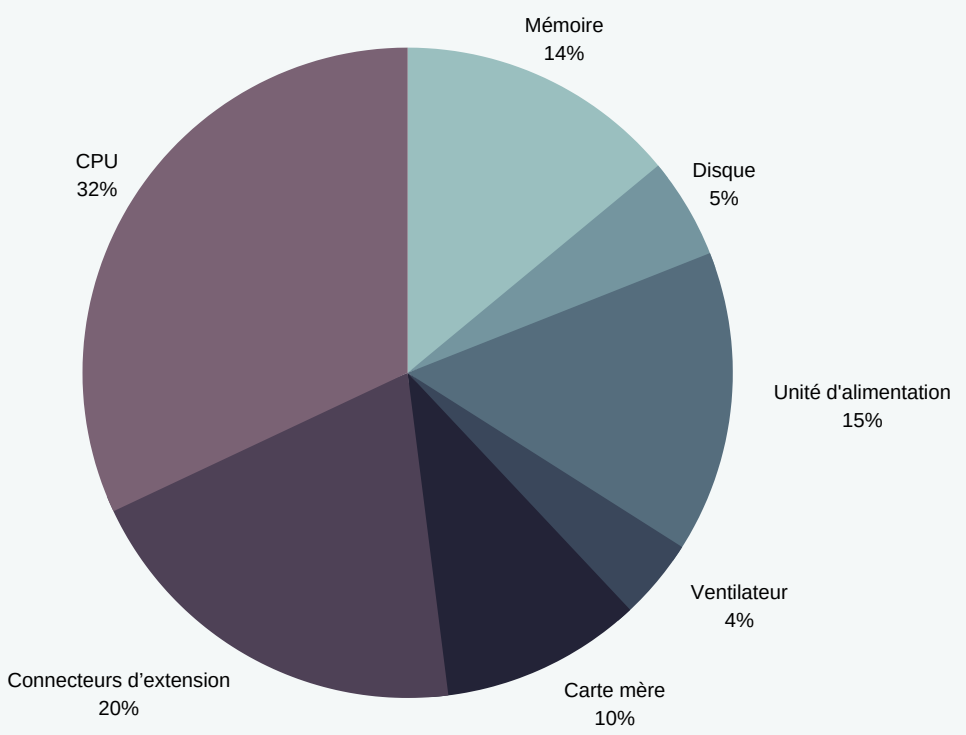


FIGURE 14 – La répartition de la consommation de l'énergie d'un serveur [148].

tivité des différentes unités fonctionnelles du CPU. [16] ont proposé et validé des modèles de régression linéaire simple basés uniquement sur l'utilisation du CPU pour déterminer la puissance du système dans son ensemble. Un modèle similaire pour les VMs a également été fourni par [140], et sa formule comporte différentes variables indépendantes pour des scénarios d'application différents. Par exemple, au niveau du réseau de diffusion de données, le taux d'utilisation du serveur pendant des intervalles de temps donnés constitue une variable explicative du modèle [179]. Dans le cas d'un serveur virtualisé, le rapport entre le bon débit et le débit maximal pris en compte est considéré comme une variable indépendante [57]. [37] ont assumé que la consommation de la puissance du serveur évolue linéairement en fonction du rapport entre le débit et la vitesse de traitement. Pour une boucle de commande à rétroaction, le modèle se base sur le niveau de performance des processeurs pendant la phase de contrôle [17]. De plus, si le serveur inactif consomme un pourcentage constant de la puissance utilisée en fonctionnement à la vitesse maximale du CPU, la puissance du serveur en veille peut être remplacée par la puissance maximale multipliée par ce pourcentage [51]. [143] ont mis en œuvre un modèle de régression linéaire en considérant la différence entre la température du CPU et la température de l'air en entrée pour étudier l'impact négatif de la température ambiante sur la consommation de la puissance. Les auteurs ont supposé que la température du pro-

cesseur est linéairement dépendante de l'utilisation du serveur, et la relation entre la température d'entrée et l'augmentation de la puissance, qui se reflète dans le terme Δ de la fonction 4.2, a été analysée [149].

Bien que les modèles de régression simple basés sur l'utilisation du CPU soient capables de fournir une précision de prédiction raisonnable pour les charges de travail intensives du CPU, ils tendent à être considérablement imprécis pour prédire la consommation de la puissance causée par les entrées/sorties et les applications intensives au niveau de la mémoire [29]. Dès lors, pour rendre le modèle plus représentatif, la régression linéaire multivariée est employée pour construire des modèles combinant la consommation de la puissance et les mesures de plusieurs composantes. [81] ont fourni un modèle de puissance basé sur l'utilisation du CPU, de la mémoire, du disque et de la bande passante, et [52] ont proposé un modèle de régression linéaire ternaire par pièces classifiées pour obtenir une prédiction de puissance plus précise. [218] a admis que les sous-systèmes tels que le processeur et le disque présentent une consommation de la puissance linéaire en fonction de leur propre utilisation. En revanche, [28] ont considéré l'utilisation du CPU, le nombre de *last level cache* (LLC) manquants et le nombre d'octets lus et écrits comme des paramètres du modèle. [172] ont décrit la consommation électrique des serveurs en se basant sur des indicateurs de performance ($perf_cycles$, $perf_cache.references$, $perf_cache.misses$). Les auteurs [68] ont en outre ajouté la température du processeur dans le modèle. *VMeter* est un outil de modélisation de la puissance basé sur la surveillance en temps réel des ressources du système et peut prédire la puissance instantanée avec une précision moyenne et médiane atteignant respectivement 93% et 94% [171]. Contrastant avec ces modèles, [20] ont combiné la température extérieure, la température du CPU, le système de transmission bus et les défauts dans le cache de processeur $L2$ pour régresser le modèle de puissance, le taux de précision étant de 4%. Habituellement, ces modèles tendent à bien prédire la consommation électrique dans la mesure où les caractéristiques de la charge de travail sont constantes.

2. **Modèle non linéaire et modèle polynomial** : parallèlement à la régression linéaire simple, [16] ont proposé un modèle non linéaire pour déterminer la consommation d'énergie du serveur. Ce modèle est plus précis que le modèle linéaire lorsqu'il s'agit d'estimer la consommation de l'énergie totale d'une machine. Cependant, comme mentionné précédemment, les modèles de régression linéaire simple ne sont pas adaptés à tous les serveurs, sauf ceux à forte intensité de CPU, et font appel à des entrées quadratiques multiples pour obtenir une erreur absolue moyenne de 9%. De ce fait, [70] ont employé des modèles polynomiaux à haut degré pour évaluer la consommation électrique du serveur et ont constaté que le polynôme cubique (équa-

tion polynomiale de degré 3) représente le meilleur modèle. De même, Lin et al [134] ont représenté la consommation de puissance au sein d'un serveur en se basant sur le polynôme du second ordre.

3. **Modèle de fonction puissance** : Pour certains modèles, le CPU est la composante principale du modèle de consommation de l'énergie. Elle fonctionne à une fréquence f et sa puissance dynamique est proportionnelle à $V^2 \times f$, où V représente la tension en cours de fonctionnement ; la fréquence diminue au fur et à mesure que celle-ci décroît. La consommation de la puissance électrique de tous les autres composants (sauf le processeur) est indépendante de la fréquence, alors, la consommation électrique d'un serveur peut être exprimée sous forme d'une fonction cubique de la fréquence de traitement [13]. En outre, dans le domaine de l'informatique en Cloud, l'utilisation des serveurs et le rendement des services ont été considérés comme les paramètres principaux du modèle en tenant compte du compromis entre le coût d'énergie consommée et la QoS rendue [77]. Cependant, certaines études font appel au taux d'utilisation des serveurs en tant que variable pour caractériser le modèle exponentielle de puissance de degré 3 [53]. De plus, [114] ont démontré qu'il existe une forte corrélation entre la consommation de l'énergie par les serveurs et l'utilisation des processeurs.

3.2.3 Autres modèles

En complément des modèles précédemment évoqués, plusieurs solutions ont été mises en œuvre. [167] ont analysé le taux d'utilisation du CPU et la fréquence pour déterminer la consommation de l'énergie de chaque dispositif actif. Pareillement, [138] ont employé le produit du taux d'utilisation du serveur par la température de la microplaquette pour représenter la consommation de l'énergie électrique. [180] ont conçu un modèle simplifié de la consommation de l'énergie par un serveur, en considérant que l'utilisation de l'énergie électrique est maximale si au moins un processus applicatif est exécuté, alors qu'elle reste minimale si aucun processus n'est en exécution.

En plus du CPU et des autres composants du serveur, les éléments de refroidissement tels que les ventilateurs du CPU consomment une certaine quantité de l'énergie électrique. [101] considèrent que la consommation de l'énergie d'un serveur est constituée de la consommation de la puissance liée aux activités de la technologie de l'information et à ceux associées au mécanisme de refroidissement. À la différence des approches décrites précédemment, [27] ont développé un modèle de consommation électrique qui repose sur la normalisation de la puissance du système. De manière spécifique, le procédé de *Power-Capping* [49], c'est-à-dire, la limitation de la consommation d'un système, se déclenche lorsque l'utilisation du processeur excède 80%. Ce mécanisme peut être ajusté par une

fonction de puissance en utilisant la puissance du CPU pendant une durée précise [39]. La corrélation entre l'énergie consommée lors du traitement des requêtes, la phase d'inactivité et la mise en veille a ainsi été établie. Par ailleurs, des modèles de consommation de puissance prenant en compte les processeurs multicœurs [102] et le nombre de VMs actives [190] ont été proposés, et la méthode des réseaux de neurones artificiels a été largement utilisée pour la modélisation de la consommation électrique dans les centres de données [146].

3.2.4 Comparaison des modèles de consommation électrique des serveurs

Pour conclure, la figure 15 illustre le bilan statistique des variables les plus utilisées pour la modélisation de la consommation énergétique d'un serveur. On constate que le taux d'utilisation du CPU est la variable la plus fréquemment utilisée dans les modèles de consommation de l'énergie, suivie par la fréquence du CPU, les indicateurs de la performance et de la température. Les différents modèles fournissent des estimations variées, et leur précision influence la prédiction des dépenses énergétiques ainsi que le choix de la meilleure stratégie de contrôle. Certaines études ont validé la précision des modèles. Selon ces études, les pourcentages d'erreur des modèles additifs et des modèles BA est respectivement inférieur à 9% et 11%. Pour les modèles additifs, le taux d'erreur de la majorité des modèles est inférieur à 5%, et le taux d'erreur le plus élevé atteint 10% dans le pire des cas. Quant aux modèles BA, le taux d'erreur des modèles de régression linéaire simple, des modèles de régression linéaire multivariée, des modèles de fonction de puissance et des modèles polynomiaux est généralement inférieur à 4.6%, 7%, 6% et 4%, respectivement. Plus spécifiquement, la fiabilité d'un modèle de régression linéaire est améliorée de 13.6% en tenant compte de la température de l'air d'entrée qui atteint 45°C [149]. De même, pour évaluer la précision des modèles sous des conditions similaires, [70] ont comparé les modèles linéaire, quadratique et cubique et ont conclu que le modèle cubique pouvait bien correspondre aux valeurs observées. [134] ont examiné les taux d'erreur de six modèles de puissance et ont conclu que le modèle polynomial avait le plus faible taux d'erreur (1.615%), suivi par le modèle de fonction de puissance (2.794%) et le modèle quadratique (2.974%). D'après ces études, il est possible de considérer que le modèle polynomial est le plus adapté pour modéliser la consommation de puissance des serveurs ; cependant, sa complexité en terme du temps de calcul reste un problème majeur.

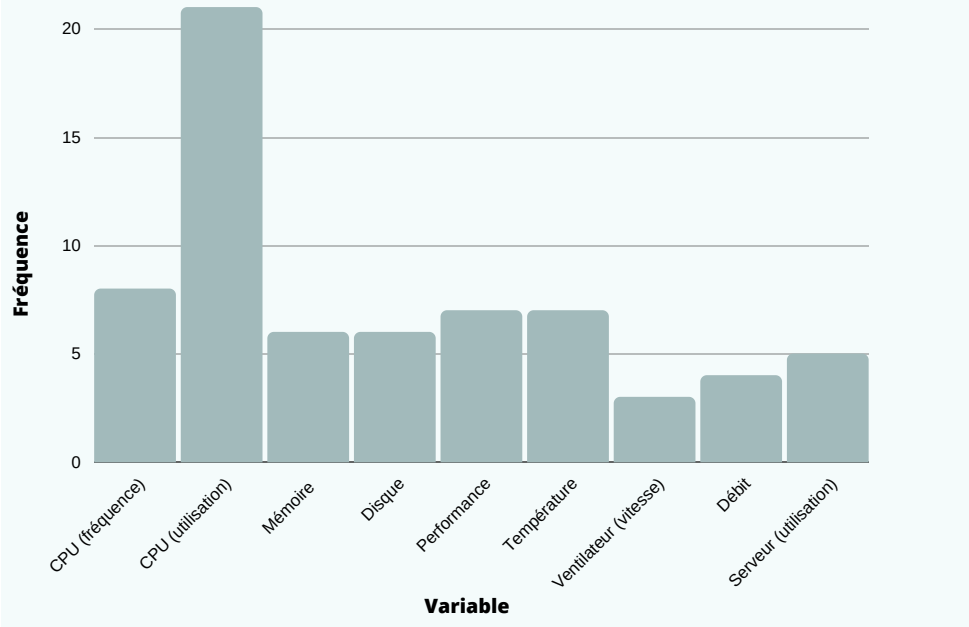


FIGURE 15 – Le bilan statistique des différentes variables utilisées pour la modélisation de la consommation énergétique d'un serveur.

3.3 Modèles de placement des VMs

3.3.1 Placement statique des VMs

Le placement statique des VMs a fait l'objet d'études sous différents angles : Conception, objectifs et techniques. Notamment, le placement statique des VMs a été étudié comme un problème de satisfaction des contraintes dans plusieurs contextes [51, 67, 90, 99]. Ces méthodes ont permis de placer les VMs en satisfaisant les contraintes de ressources des PMs tout en visant à minimiser la consommation énergétique du centre de données ; cependant, aucune de ces méthodes n'a pris en compte les différents profils énergétiques des PMs ou les fluctuations de la charge de travail des VMs.

L'une des premières études sur le problème du placement statique des VMs a été menée par [19]. Ils ont conçu une approche de placement statique de VMs, en particulier pour minimiser la déperdition de ressources. Ils ont formulé le problème de placement sous forme d'un problème de *Bin-Packing*. Pour résoudre ce problème, ils ont recouru à l'algorithme heuristique *First-Fit Decreasing* (FFD). Les serveurs ont été classés par ordre décroissant en fonction de leurs capacités, ainsi que les ressources de chaque serveur ont été allouées aux VMs en fonction de leurs demandes. Une fois que les VMs sont attribués au serveur, la capacité de ce dernier diminue. Néanmoins, le classement des serveurs par ordre décroissant en fonction de leur capacité n'est pas efficace du point de vue énergétique, car

un serveur ayant une plus grande capacité risque d'avoir une efficacité énergétique moins importante.

Afin de réduire les coûts d'exploitation et l'impact sur le milieu environnemental, une solution a été mise en œuvre pour gérer les différentes ressources des centres de données [51]. Deux stratégies ont été adoptées : l'une pour le placement basé sur le problème de *Bin-Packing* et l'autre pour le placement dynamique. Selon cette technique, les VMs ont été triées par ordre décroissant suivant l'utilisation du CPU. Puis, une variante de l'algorithme *Best-Fit Decreasing* (BFD) a été adopté pour attribuer la VM ayant la plus haute intensité d'utilisation du CPU à la PM ayant la plus basse consommation d'énergie. Ainsi, le processus de placement est répété jusqu'à ce que toutes les VMs soient placées sur les PMs. La qualité d'une solution utilisant l'algorithme BDF est cependant limitée, et des algorithmes heuristiques alternatifs plus puissants doivent, typiquement, être mis au point pour le placement des VMs. Les algorithmes métaheuristiques basés sur la notion de population permettent d'échanger des informations relatives aux résolutions entre les populations ; ce faisant, ils sont les plus adaptés pour résoudre un problème NP-difficile [92]. Notre travail dans cette thèse se base, entre autres, sur un algorithme métaheuristique puissant (OCF) pour le placement statique des VMs.

[97] ont proposé un modèle de placement des VMs en fonction de la disponibilité et de l'efficacité énergétique pour éviter les dysfonctionnements des applications. L'approche consiste à utiliser la redondance des VMs hébergeant les mêmes applications. Par la suite, les VMs sont placées dans des PMs différentes, de sorte que si une panne survient dans l'une des PMs, l'exécution des applications sera effectuée par d'autres VMs sur d'autres PMs. Il faut cependant noter que cette approche augmente la consommation d'énergie, car plus de PMs doivent être mises en marche. En plus, l'approche ne prend pas en compte la diversité des profils énergétiques de ces PMs, aussi bien que la charge de travail dynamique des VMs.

Compte tenu de multiples objectifs, un schéma de placement des VMs a été mis en place par [61] pour minimiser la consommation énergétique, augmenter l'utilisation des ressources et réduire les coûts du trafic sur le réseau. Les auteurs ont formulé le problème du placement des VMs en tant qu'un problème de *Bin-Packing*, et celui du trafic réseau en tant que problème d'affectation quadratique dans un centre de données. Les ressources CPU, mémoire, disque et bande passante ont été considérées comme des contraintes de formulation. Les ressources ont été également incorporées dans le modèle de consommation d'énergie. Pour résoudre le problème formulé, les auteurs ont utilisé la coupe minimum avec l'algorithme BFD afin d'obtenir un nombre réduit de PMs et d'éléments constitutifs du réseau. Pourtant, l'utilisation de la mémoire, qui a été prise en compte dans le modèle, n'a pas montré clairement un effet direct sur la consommation énergétique d'une PM. En

contraste, l'utilisation du CPU révèle une forte corrélation avec la consommation d'énergie. Notre travail consiste à utiliser les variations de la CPU pour modéliser le processus de placement, car ce sont elles qui ont généralement plus d'impact sur la consommation énergétique.

Dans leur étude, [63] ont formulé le placement des VMs comme un problème d'optimisation à deux objectifs : minimiser le nombre des PMs d'accueil tout en maximisant leur utilisation. Les ressources CPU et mémoire ont été considérées des contraintes d'optimisation, et ils ont conçu un algorithme qui utilise une transformation matricielle. L'algorithme consiste à regrouper les VMs pour générer une solution initiale, ce qui permet de réduire le temps de traitement nécessaire à la recherche d'une solution optimale. [186] ont affirmé que le travail de [63] n'était pas pratique car une recherche exhaustive était nécessaire pour obtenir une solution au problème NP-difficile du placement des VMs.

Dans le but de maximiser l'efficacité énergétique, un modèle a été proposé par [188] pour la consolidation des VMs dans un centre de données. Il porte sur la théorie d'AG pour le placement des VMs. En plus, le modèle prend en compte à la fois les PMs et le trafic du réseau. A cet effet, un algorithme a été mis au point qui utilise le concept de l'AG pour la recherche locale. De même, une procédure de correction est employée pour convertir les solutions infaisables en solutions réalisables. Les ressources considérées comprennent le CPU, la mémoire et la bande passante. Une étude ultérieure [105], a indiqué que la charge de travail est susceptible de varier avec le temps et qu'il convient de considérer ce changement.

Un autre modèle de placement statique des VMs a été développé par [193]. Les auteurs ont développé un algorithme de placement statique des VMs basé sur la compatibilité qui utilise l'interférence entre les VMs pour les regrouper dans les PMs. De plus, les données relatives aux contraintes ont été collectées pour combiner simultanément des VMs compatibles plutôt que de les regrouper uniquement en fonction de leur efficacité. Le problème du placement statique des VMs a été formulé comme un problème de *Bin-Packing*. Ensuite, les auteurs ont adopté l'algorithme FFD pour la résolution du problème ainsi formulé. Cependant, la combinaison d'un certain nombre de VMs compatibles à placer sur une PM n'améliore pas l'utilisation des ressources et donc, l'efficacité énergétique n'est pas garantie.

3.3.2 Placement dynamique des VMs

Les algorithmes de placement des VMs sont en constante évolution pour s'adapter aux nouveaux problèmes de gestion des ressources. L'objectif du placement dynamique des VMs est de parvenir à des solutions efficaces dérivées de la répartition actuelle de ces VMs et de réduire au minimum les éventuels coûts de l'utilisation de l'énergie. Vue

que, dans notre thèse, on se focalise sur le placement statique des VMs, on se limite à fournir une brève description des méthodes les plus récentes introduites pour modéliser le placement dynamique des VMs.

[105] ont proposé une approche pour le placement des VMs visant à minimiser la consommation d'énergie. Ils ont formulé le problème de placement dynamique des VMs comme étant une programmation linéaire en nombres entiers. L'approche a été traitée en deux phases. Dans la première phase, l'algorithme trouve des solutions possibles tout en respectant les contraintes. Dans la seconde phase, l'algorithme place les VMs dans les PMs en utilisant les solutions réalisables trouvées dans la première phase. En cas de violation des contraintes de ressources d'une PM, les VMs sont initialement migrées vers une PM provisoire. Après avoir vérifié qu'il n'y a pas de violation des contraintes de ressources d'une PM cible potentiel, les VMs sont migrées vers ce dernier. Cette approche permet de réduire efficacement la contrainte de temps, mais le fait de placer une VM à deux reprises dégrade les performances du système. En outre, [194] ont montré que les méthodes exactes, telles que la programmation linéaire en nombres entiers, ont une scalabilité inefficace et pourraient être non applicables aux centres de données à grande échelle.

[191] ont étudié les approches *Off-Line* et *On-Line* pour le placement des VMs afin d'aboutir à une consommation d'énergie minimale pour tout un centre de données. En termes de placement dynamique des VMs, ils ont mené une analyse comparative pour évaluer l'algorithme dynamique et trouver la relation existant entre les contraintes des PMs et celles des VMs. Ils ont défini des seuils minimaux et maximaux pour les PMs. Une PM peut augmenter sa charge de travail jusqu'au seuil maximal défini. Les auteurs ont suggéré la possibilité d'ajouter une autre PM au système lorsque toutes les PMs sont surchargées. Cependant, il ne semble pas rentable d'ajouter continuellement de nouvelles PMs dans les centres de données réels.

[115] ont également présenté un modèle de gestion dynamique des ressources pour minimiser la consommation d'énergie et éviter d'autres dégradations dans les SLAs pour assurer la QoS offerte aux utilisateurs finaux. Trois méthodes ont été définies dans ce schéma pour atteindre ces objectifs. La première méthode consiste à placer les VMs dans les PMs en utilisant une méthode d'allocation heuristique. La deuxième méthode est une planification dynamique des ressources basée sur des techniques de migration permettant de déplacer les VMs des PMs surchargés vers les PMs sous-chargés afin de garantir l'équilibre des ressources. Puis, la troisième méthode consiste en un processus d'optimisation qui identifie les PMs les plus sous-chargées et transfère leurs VMs vers d'autres PMs. Même si le modèle présenté englobent les contraintes les plus intéressantes pour un centre de données, le fait de garder les ressources en équilibre tout en allouant des VMs ne garantit pas une efficacité énergétique optimale, car il faut activer plus de PMs pour

équilibrer les ressources.

Plus récemment, [123] ont proposé un algorithme d'allocation des ressources pour les applications intensives (au niveau d'utilisation de ressources), en considérant le placement de la VMs et le placement des données simultanément. Cet algorithme adapte la métaheuristique populaire de l'optimisation OCF. Étant donné le nombre de VMs et de données en entrée, l'algorithme proposé sélectionne un ensemble PMs qui minimise le délai de transfert des données. Ensuite, les blocs de données à copier sur ce serveur particulier sont déterminés en fonction du nombre de VMs qu'une PM peut héberger. Les données sont copiées, ainsi, sur le stockage physique de la PM correspondante et le nombre donné de VMs est y exécuté. De façon similaire, [136] ont introduit une approche de placement et de consolidation multi-objectifs et orientée énergie et QoS, qui vise à atteindre un compromis entre l'efficacité énergétique, la performance des systèmes et la QoS rendu. Cette approche se compose de deux algorithmes complémentaires : un premier algorithme de placement des VMs et un second algorithme de consolidation (les deux algorithmes sont multi-objectifs basés sur l'OCF). L'algorithme de placement des VMs cherche à trouver des solutions optimales de placement visant à minimiser la consommation totale d'énergie, le gaspillage des ressources (en termes de CPU), et le coût de l'énergie utilisée pour les opérations de communication (résultant de la charge du trafic échangé entre les VMs). En se basant sur les résultats du premier, l'algorithme de consolidation des VMs optimise l'utilisation des ressources grâce à leur migration, en se focalisant sur la recherche de solutions optimales qui tendent vers non seulement la minimisation des violations des SLAs et la minimisation du nombre de migrations de VMs, mais aussi la minimisation du nombre de PMs actives, et donc la minimisation de la consommation totale d'énergie du centre de données.

Le placement des VMs dans les PMs a fait l'objet de nombreuses études. En général, le problème de placement des VMs est conçu en tant que problème d'optimisation à contraintes, soit un problème d'optimisation combinatoire. Pour les centres de données écoénergétiques, la fonction objective de l'optimisation consiste en la consommation totale d'énergie de l'ensemble du centre de données pendant une certaine période. Pour résoudre le problème d'optimisation, on adopte des méthodes mathématiques exactes, des heuristiques et des métaheuristiques [150, 141].

3.3.3 Algorithmes et méthodes de résolution du placement des VMs

Plusieurs algorithmes permettent de résoudre le problème du placement des VMs en terme d'efficacité énergétique. Habituellement, le placement des VMs est défini comme un problème de sac à dos qui est un type typique de problèmes NP-difficiles. Ce type de

problème est relativement compliqué et donc non résolvable en temps polynomial. Compte tenu de la littérature existante, nous constatons que de nombreux auteurs ont adopté des méthodes exactes et des techniques métaheuristiques pour résoudre le placement des VMs. Dans ce qui suit, nous abordons certains modèles qui sont pertinents pour ce type de problèmes.

3.3.3.1 Placement basé sur l'optimisation mathématique

[71] ont développé une approche pour l'allocation des ressources en utilisant la programmation dynamique à contraintes pour la formulation du problème de placement des VMs. La programmation à contraintes permet de répartir les ressources et de regrouper les VMs. Cette approche a été évaluée dans le cadre d'une simulation. Elle a été comparée à un autre modèle traditionnel et les résultats ont montré que ce modèle satisfait la QoS et réduit le coût d'allocation des ressources plus que le modèle traditionnel. Un autre modèle de programmation à contraintes a été élaboré par [173]. Les auteurs ont mis en place un mécanisme de gestion de l'approvisionnement de ressources et de placement dynamique des VMs afin de maximiser l'utilisation totale, d'atteindre les SLAs et de réduire les coûts énergétiques opérationnels.

Un modèle de programmation linéaire en nombres entiers a été adopté par [170] pour consolider les VMs dans un centre de données en Cloud. La méthode caractérise la charge de travail en combinant plusieurs éléments d'analyse de données afin de trouver une solution optimale pour l'affectation des VMs aux serveurs physiques. Dans cette approche, deux problèmes formulés ont été combinés, le traitement de données et l'optimisation pour présenter une planification globale de la capacité de l'infrastructure informatique. Ensuite, les auteurs ont utilisé un algorithme heuristique pour résoudre les problèmes formulés. Les résultats ont montré une importante amélioration des coûts atteint 31%. Le problème de cette approche est qu'un nombre limité de VMs est spécifié pour l'hébergement sur chaque PM. Cela entraîne un gaspillage de ressources au niveau des serveurs, ce qui augmente la consommation énergétique.

La programmation stochastique en nombres entiers peut manipuler des données inconnues et puisque les demandes de ressources nécessaires aux VMs peuvent être prédéfinies ou estimées à priori en utilisant cette approche, les problèmes de placement des VMs peuvent être définis par une programmation stochastique en nombres entiers afin de réduire la consommation énergétique conformément à la prédiction des besoins en ressources. Dans [46], un modèle de programmation stochastique a été appliqué pour allouer stochastiquement les VMs aux tâches qui arrivent. Ensuite, les VMs ont été placées sur les PMs en fonction de leurs exigences en matière de ressources de CPU, de mémoire et de stockage selon une charge de travail équilibrée. Les auteurs ont résolu le problème formulé

en appliquant l'algorithme *MaxWeight* qui a surperformé l'algorithme BFD en fournissant un taux de performance optimal. Cependant, l'application de la programmation stochastique en nombres entiers pour prévoir les demandes de ressources n'est pas optimale pour assurer l'efficacité énergétique en raison de certaines incertitudes des utilisateurs, ce qui entraîne une allocation inexacte.

La programmation dynamique a été appliquée dans quelques études liées au problème du placement des VMs. [83] ont mis au point un framework de placement dynamique des VMs pour réduire la consommation d'énergie et les violations des SLAs. Les auteurs ont conçu le problème du placement dynamique des VMs selon une programmation dynamique basée sur le problème de sac à dos. Les serveurs surchargés ont été prédits grâce à une méthode de régression locale. Ensuite, les auteurs ont proposé deux algorithmes heuristiques pour résoudre la migration des VMs et le problème de consolidation. Les performances des algorithmes ont été évaluées en termes de consommation d'énergie, de taux de migration des VMs, de nombre d'hôtes redémarrés et de violations des SLAs. Les résultats expérimentaux ont indiqué que l'algorithme a fonctionné plus efficacement par rapport aux algorithmes existants suivant différents critères. Cependant, dans le travail de [196], les auteurs ont souligné que les CPUs surchargées et les violations des des SLAs étaient élevés en PMs alors que les algorithmes associés ont atteint une efficacité énergétique et ont empêché le redémarrage des PMs.

Une autre étude adoptant la programmation dynamique a été présentée dans [178]. Les auteurs ont proposé une approche du placement des VMs visant à améliorer l'efficacité énergétique. Dans cette approche, plusieurs copies de VMs ont été créées. Ensuite, la programmation dynamique a été hybridé avec la recherche locale pour placer ces copies de VMs dans les PMs. Dans [128], les auteurs ont introduit une approche d'optimisation de la charge de travail pour l'attribution des tâches dans un centre de données en Cloud. L'algorithme proposé a été évalué par rapport à la méthode de programmation dynamique mentionnée ci-dessus, à l'algorithme glouton (*Greedy Algorithm*) [98] et au *First-In-First-Out* (FIFO).

La programmation mathématique, telle que les modèles d'optimisation mentionnés dans cette sous-section, est inefficace pour les centres de données à grande échelle. Selon [194] dans leur étude des algorithmes et des techniques de placement des VMs, certaines approches de placement des VMs ont adopté les méthodes exactes qui reposent sur la programmation mathématique et ces types de techniques algorithmiques sont inefficaces pour l'évolutivité de la mise en œuvre dans des environnements pratiques. Comme le placement des VMs est un type de problème d'optimisation, il est lié aux problèmes de sac à dos et de *Bin-Packing*. C'est pourquoi ces deux problèmes sont largement résolus par des algorithmes heuristiques tels que l'algorithme du BFD et l'algorithme du FFD. Ils

peuvent également être résolus par des métaheuristiques robustes telles que l'OCF. Tous ces algorithmes promettent de fournir des solutions très efficaces et pratiques, contrairement aux méthodes exactes mentionnées plus haut dans cette sous-section. L'heuristique et la métaheuristique diffèrent dans leur utilisation selon les exigences du problème, sa conception, ses objectifs et sa taille.

3.3.3.2 Placement basé sur des algorithmes métaheuristiques

L'algorithme métaheuristique est un type d'algorithme heuristique qui est efficacement conçu pour produire une solution suffisamment adéquate permettant de résoudre des problèmes d'optimisation NP-difficiles. Les algorithmes métaheuristiques surpassent remarquablement les algorithmes heuristiques classiques dans la résolution des problèmes d'optimisation complexes avec une conception incertaine et stochastique et des éléments dynamiques [23]. Le concept d'algorithme métaheuristique a été évoqué en 1986 par [212] comme un nouveau type d'algorithme approximatif. Le concept de ce dernier consiste en une combinaison de méthodes classiques visant à introduire un framework heuristique différent capable d'effectuer des recherches dans un grand espace de manière plus efficace et plus rentable que les méthodes classiques [163]. Dans ce travail, on étudie deux métaheuristiques inspirées de la nature, l'AG [214] et l'OCF.

L'une des premières tentatives d'utilisation de l'OCF pour le placement des VMs a été rapportée dans [40]. Il a étudié comment placer un ensemble de VMs dans un ensemble de PMs. Le problème a été formulé comme un problème de *Bin-Packing* statique multidimensionnel. L'objectif était de trouver le nombre minimum de PMs pour accueillir les VMs demandées. Un algorithme OCF a été conçu pour résoudre le problème de remplissage de bacs (PMs), basé sur un algorithme heuristique *max-min*.

Un autre aspect de l'adoption de l'algorithme OCF dans le placement des VMs dans un nombre limité de PMs a été l'objet du travail introduit par [79]. Ils ont utilisé une méthode différente pour traiter la phéromone déposée entre deux paires de VMs afin de mesurer leur utilité dans une PM. Dans la première génération, le nombre de PMs était supposé égal au nombre des VMs. Ensuite, le nombre de PMs a été réduit à plusieurs reprises pour obtenir moins de PMs actives. Ce travail sur l'OCF pour le placement des VMs a été amélioré ultérieurement par la même équipe [135]. Cette amélioration a été conçue en intégrant la recherche locale afin de perfectionner la solution optimale existante avant de mettre à jour la phéromone globale. Elle a été évaluée dans un centre de données à grande échelle avec des ressources homogènes et hétérogènes. La version originale de l'OCF [79] et sa version améliorée [135] ont été comparées avec la méthode FFD. Sans surprise, l'OCF a fourni une solution plus optimale ,au niveau du nombre de PMs, que la méthode FFD. Cependant, partir d'une solution initiale avec un nombre égal de VMs

et de PMs implique un espace de recherche inutilement énorme. Un centre de données à petite échelle accueille généralement des dizaines de milliers de VMs. Cela implique que l'obtention d'une solution à partir de ces algorithmes, pour un centre de données réel, devient trop difficile du point de vue computationnel.

Un travail similaire a été présenté par [56] sur l'application de l'OCF dans le placement des VMs. Dans cette approche, le problème du placement des VMs a été exprimé sous forme d'un problème d'optimisation combinatoire avec deux objectifs de minimisation, soient la consommation d'énergie et de gaspillage des ressources ; ainsi, il prend en compte deux types de ressources : CPU et mémoire. L'OCF a été adopté pour résoudre ce problème d'optimisation multi-objectifs. Il a été évalué par rapport à l'AG au niveau d'un centre de données à grande échelle. Les résultats de l'évaluation ont montré que l'OCF était plus performant que l'AG. Cependant, il a été indiqué par [198] que la solution finale de cette méthode devait être retenue à partir d'un ensemble de *Pareto*[1]. Cela signifie que la solution choisie pourrait ne pas être satisfaisante [198]. Pour résoudre ce problème, des améliorations ont été apportées par [130] en développant des colonies d'agents multiples pour des solutions diverses.

De même que les travaux examinés ci-dessus, l'OCF a également été adoptée pour résoudre un problème de consolidation des VMs à deux objectifs en vue de la réduction de la consommation énergétique et de la surutilisation des ressources [76]. L'OCF a été incorporé dans le calcul vectoriel de sorte que l'utilisation des ressources soit équilibrée. L'évaluation a également été menée grâce à des comparaisons entre l'OCF et d'autres méthodes. Il a été démontré que cette méthode surpasse les méthodes comparées en termes de performance de consommation d'énergie et de nombre de PMs actives. Cependant, une étude récente [126] a indiqué que le système introduit par [76] n'a pas pris en compte la fluctuation de la charge de travail dans les VMs.

Une autre étude récente de [125] portant sur le placement des VMs basé sur l'OCF vise à minimiser la consommation énergétique. Elle a été conçue avec un algorithme OCF normalisé, dans lequel la phéromone est mise à jour dynamiquement et non plus une constante issue d'une initialisation statique. Les informations heuristiques ont été calculées en fonction de l'utilisation du CPU et des ressources mémoire. L'inclusion de cette dernière parmi les indications heuristiques augmente la complexité du processus OCF tout en fournissant des précisions limitées pour l'amélioration de l'efficacité énergétique dans un centre de données.

1. Pareto : igi-global.com/dictionary/pareto-optimal-solution/21879

3.4 Modèles de migration en temps réel des VMs

3.4.1 L'origine de la migration en temps réel

[8] ont été les premiers à proposer la migration d'une VM (appelée capsule dans leur terminologie) en 2002. Il y a 20 ans de décalage entre cette proposition et les inventions de deux concepts dont elle peut directement découler, à savoir la migration des machines et des processus virtuels. La technologie des VMs avait été proposée par *IBM* dans les années 1960 [213], et la migration des processus avait déjà été mise en œuvre dans certains systèmes d'exploitation au début des années 1980 [5]. Une des raisons importantes de la naissance de la migration des VMs est l'évolution des demandes des utilisateurs. Un an auparavant, [162] avaient affirmé que la migration de processus n'était pas suffisante pour répondre à la demande de mobilité des utilisateurs. [8] ont affirmé également que le recours à la migration de leurs VMs inclut la mobilité des utilisateurs ainsi qu'une gestion aisée des PMs sous-jacentes, ces deux éléments étant considérés comme des caractéristiques propres au *Cloud Computing* actuel.

L'objectif principal de [8] était de savoir comment transférer un grand disque avec une liaison à ligne d'abonné numérique (*DSL*[2]) étroite. La VM migrée est suspendue pendant le transfert et la migration n'est donc pas en temps réel. Le paradigme du Cloud, devenant populaire, a évolué vers une migration en temps réel et les recherches ont également porté sur la manière de transférer efficacement les pages de mémoire pendant qu'elles sont mises à jour par la VM en cours de l'exécution. Ce changement d'orientation est dû à deux raisons. Du point de vue de la demande client, les utilisateurs du Cloud ont besoin d'une disponibilité supérieure à 99%, c'est pourquoi la migration en temps réel est obligatoire. D'un point de vue technique, les *Gigabits* ou même 10 *Gigabits* Ethernet sont devenus courants dans les centres de données. Ainsi, les images de disque de grande taille peuvent être simplement localisées sur un stockage partagé si la performance du stockage n'est pas la préoccupation principale.

L'amélioration des performances de la migration des VMs en temps réel (comme la réduction du temps total de migration et la diminution des interruptions de service pendant la migration) est un sujet d'actualité depuis que la migration des VMs en temps réel a été lancée, car ses performances sont étroitement liées au niveau de gestion du Cloud et à l'apparition de nouvelles applications du *Cloud Computing*. Après plus de dix ans de développement, de nombreuses réalisations ont été obtenues pour la migration des VMs en temps réel. Cependant, il reste encore quelques problèmes à résoudre ou à améliorer, à savoir l'imprévisibilité. Ainsi, la nature imprévisible de la consommation des ressources

2. DSL : http://gdt.oqlf.gouv.qc.ca/ficheOqlf.aspx?Id_Fiche=8873235

dans un centre de données a conduit les chercheurs à explorer des méthodes plus adaptatives et proactives qui ont la capacité d'analyser les données précédemment recueillies et d'en tirer des connaissances. Dans la sous-section suivante, nous abordons les algorithmes d'intelligence artificielle qui ont été appliqués pour améliorer le processus de la migration des VMs.

3.4.2 *Machine Learning* pour la migration en temps réel des VMs

Pour atténuer l'imprévisibilité de la migration en temps réel, une solution intéressante consiste à concevoir un modèle autonome capable de prendre des décisions en fonction des informations fournies en temps réel. Le *Machine Learning* est un outil puissant qui consiste à utiliser des données pour résoudre des problèmes complexes dans des systèmes réels. Plusieurs chercheurs ont appliqué les concepts du *Machine Learning* pour résoudre des problèmes critiques liés à la gestion des centres de données, tels que la modélisation des performances des VMs [45] ou un framework de gestion de cluster basé sur les interférences [181].

Dans la plupart des travaux effectués, plusieurs méthodes ont été mises en place pour prédire l'utilisation du CPU à l'aide des traces de la charge de travail. Différents modèles du *Machine Learning* peuvent être utilisés pour développer des stratégies de migration des VMs pour équilibrer la charge de travail par rapport aux ressources des PMs. Les modèles du *Machine Learning* fournissent une aptitude dans les systèmes en Cloud pour apprendre automatiquement et améliorer leurs performances. Le *Machine Learning* a été utilisé pour la gestion des VMs et leur répartition dans le centre de données. Il développe également le mécanisme qui permet d'apprendre et de prévoir l'utilisation future des ressources en utilisant des données historiques. Le processus de migration commence par l'observation de ces dernières, par exemple l'utilisation précédente du CPU, de la RAM, du disque et de la bande passante, afin de découvrir les modèles au sein du système et de faire de meilleures prévisions pour l'utilisation future. Dans l'environnement d'un système en Cloud, l'énorme quantité de données est traitée avec des exigences de calcul élevées. Par conséquent, la gestion des ressources en optimisant la consommation énergétique, le coût et le maintien du SLA est une question cruciale. Ainsi, pour faire face à des données volumineuses et à des besoins en ressources énormes, le *Machine Learning* peut être utilisé pour améliorer la migration des VMs dans le centre de données. Une meilleure prévision peut être effectuée pour contrôler le processus de migration en garantissant une gestion efficace des ressources, ainsi que la réduction de la consommation énergétique dans le centre de données.

Les données relatives à la charge de travail sont également utiles pour le modèle du *Machine Learning* afin d'apprendre le comportement du système et de prédire l'utilisation des ressources. La prédiction de la charge de travail est utile pour développer un système rentable. Au meilleur de notre connaissance, la prédiction de l'utilisation du CPU et de la mémoire est le domaine de recherche le plus tendance dans le *Cloud Computing* pour le développement du processus de migration des VMs tout en maintenant le SLA et l'efficacité énergétique.

Il a été observé lors de notre activité de recherche qu'à partir de 2009, les modèles du *Machine Learning* ont été appliqués sur des données historiquement établies afin de développer un mécanisme optimisé pour la migration des VMs dans les PMs et c'est toujours le cas en pratique. Par conséquent, l'apprentissage à l'aide de données qui portent sur l'utilisation passée des ressources est une approche largement acceptée par la communauté des chercheurs pour la la gestion (placement et migration) des VMs dans le centre de données. Des algorithmes basés sur le *Machine Learning* ont été utilisés, dans la littérature, pour améliorer l'allocation des ressources et la migration en temps réel des VMs dans un système Cloud et sont examinés plus en détail ci-dessous.

3.4.2.1 Modèles basés sur LR

La régression linéaire (LR) peut être appliquée pour la prédiction des caractéristiques de la charge de travail à venir dans un centre de données. Elle utilise les valeurs du taux d'utilisation à un moment donné dans certains intervalles de temps. Elle est appliquée pour établir la relation entre la charge de travail prévue et la charge de travail actuelle. On peut observer que la tendance de la charge de travail du système est linéaire sur un intervalle de temps relativement court [177]. Ainsi, un modèle de LR peut être utilisé pour résoudre un tel problème.

La plupart des approches basées sur la LR suivent la gestion des ressources d'une seule VM, par exemple l'utilisation du CPU, et fournissent une migration des VMs basée sur l'utilisation du CPU en considérant la relation entre la puissance totale consommée par la PM et son utilisation de cette métrique. La valeur de l'utilisation de cette dernière a été considérée comme le paramètre principal pour traiter les SLAs et les coûts d'énergie. Cependant, l'utilisation de la mémoire et de la bande passante est rarement prise en compte pour la migration des VMs, mais leur impact sur la consommation électrique et les SLAs ne peut être négligé. Toutefois, la régression multiple est une régression linéaire simple qui prédit la valeur de la variable dépendante en utilisant deux ou plusieurs autres variables indépendantes. Elle permet d'en savoir plus sur la relation entre plusieurs variables indépendantes et une variable dépendante. Ainsi, les valeurs d'utilisation du

CPU, de la mémoire, du disque, de la bande passante et d'autres éléments qui contribuent à la consommation énergétique peuvent être utilisées pour améliorer le mécanisme de migration des VMs.

Dans [34], les auteurs se sont concentrés sur la prédiction du coût de l'énergie et des performances de migration en utilisant deux modèles de prédiction. Le premier modèle de régression est utilisé pour la prédiction des coûts en se basant sur les données de consommation au niveau de l'hyperviseur. Le second est un modèle linéaire de haut niveau conçu selon le modèle de performance, pour estimer l'énergie nécessaire à la migration. Ils indiquent que la rapidité du réseau, la taille de la mémoire des VMs et le taux d'encrassement de la mémoire ont un impact direct sur la latence et le temps d'arrêt de la migration, ainsi que le trafic réseau. Ensuite, ils ont analysé les modèles en utilisant cinq niveaux de charge différents dans l'environnement *Xen* et ont indiqué que leurs modèles peuvent atténuer le coût de la migration.

[62] ont développé une méthode de régression linéaire (*LiRCUP*) pour prévoir l'utilisation éventuelle du CPU par les PMs en utilisant des données de *PlanetLab*[3]. Ils ont établi une relation linéaire entre l'utilisation prévue (la variable dépendante) et actuelle du CPU (la variable indépendante). Leur modèle a détecté les PMs surchargées en comparant la valeur d'utilisation prévue du CPU avec l'utilisation actuelle et a maintenu les SLAs et la consommation d'énergie. En se basant sur ce modèle, le processus de la migration de certaines VMs des PMs surchargées est effectué. Si la valeur d'utilisation prédite s'avère supérieure à la ressource CPU actuellement disponible, alors les PMs sont en état de surcharge. Une régression multiple a été également appliquée pour déterminer les PMs surchargées en utilisant la consommation du CPU, de la mémoire et de la bande passante. Ces valeurs ont été utilisées comme variables indépendantes dans l'équation de régression multiple pour prédire le taux d'utilisation à l'intervalle de temps ultérieur.

Pour optimiser le processus de migration, [118] ont proposé un modèle qui utilise la régression multiple où le CPU, la mémoire et la bande passante sont utilisées pour la détection de la surcharge en PMs et réduisent considérablement la consommation des ressources. Ils ont employé de multiples facteurs pour effectuer la gestion des VMs tout en maintenant la consommation d'énergie et les SLAs, y compris l'algorithme de migration à considéré.

[89] ont abordé le temps de migration comme conséquence de la consolidation dynamique des serveurs dans les centres de données et les environnements virtualisés. Ils ont démontré expérimentalement que le temps de migration peut être adéquatement exprimé comme une combinaison linéaire de quelques paramètres d'utilisation des ressources, notamment en termes de nombre total d'instructions CPU retirées et de nombre de *Dirty*

3. PlanetLab : https://www.planet-lab.org/

Pages observées dans le serveur source pendant la migration. En employant divers modèles de régression linéaire simple et multiple, ils ont examiné quantitativement l'importance de ces paramètres dans la réduction des erreurs résiduelles de prédiction. En outre, ils ont montré expérimentalement que la puissance expressive et la complexité des modèles dépendent du nombre de variables indépendantes qu'ils comprennent. Cependant, ils ont également montré que l'augmentation du nombre de variables indépendantes au-delà de cinq n'améliorent pas la fiabilité du modèle.

3.4.2.2 Modèles basés sur SVM

L'approche SVM a été utilisée pour développer des modèles de migration des VMs dynamiques, adaptatifs et basés sur les traces d'utilisation des ressources des VMs et des PMs. Les données ont été préparées régulièrement grâce à l'historique d'utilisation des ressources. [41] ont présenté une approche pour la migration en temps réel des VMs basée sur l'algorithme SVM. Ils ont développé un algorithme pour effectuer un contrôle de faisabilité, qui prend en compte l'application, la charge de travail, les SLAs et la configuration possible des ressources (allocation des ressources aux VMs) et classifie cette configuration en deux ensembles, l'un satisfaisant les SLAs et l'autre non. Les configurations possibles les plus proches de l'hyperplan ont été considérées comme des configurations retenues. Ils ont développé, ensuite, un système multi-agents auto-configurable et auto-optimisé pour apprendre le comportement du système afin d'en estimer le coût. Leur système apprend les modèles de performance de diverses applications et dérive un modèle de comportement à partir des données collectées auprès de tous les agents pendant leur exécution, puis la SVM est appliquée pour classer les données issues de la base de connaissances.

[111] ont développé un modèle basé sur la SVR pour prédire les *Dirty Pages* et ont configuré la migration en temps réel des VMs. Leur algorithme permet de paramétrer la migration en temps réel des VMs dans le centre de données en utilisant l'analyse et la prédiction des *Dirty Pages*, afin de rendre le processus de migration efficace. L'algorithme de prédiction basé sur le modèle SVR prend des données de séries temporelles relatives aux *Dirty Pages* et les a prédites comme séries de sortie. Ils ont mesuré la période de migration totale et les performances de la migration en temps réel. Ils ont également mis au point un modèle *Autoregressive Integrated Moving Average* (ARIMA) et les résultats expérimentaux ont indiqué que la SVR prédit les *Dirty Pages* avec plus de précision que le modèle ARIMA. Le temps de migration et le nombre total de pages transférées ont été utilisés comme les principaux indicateurs pour évaluer la performance de la migration en temps réel du système proposé.

De leur part, [129] ont introduit l'approche de consolidation des PMs qui utilise des données provenant de la consommation précédente de ressources dans le but de définir

les coefficients de pondération des PMs en fonction de la consommation de ressources des VMs hébergées sur chaque PM. Ils ont considéré l'ensemble de données d'utilisation des ressources (CPU, mémoire et pourcentage d'utilisation du disque) dans le plan et ont représenté ces données par des vecteurs, puis ils ont appliqué la SVM pour classer le statut des VMs. Des méthodes de filtrage et de pondération ont été appliquées pour la sélection de la PM accueillant la VM. Des PMs appropriés ont été sélectionnés pour héberger des VMs après le processus de filtrage et il est vérifié si une PM a suffisamment de ressources pour héberger des VMs. Le processus de pondération attribue un poids élevé aux meilleures PMs à partir des données collectées historiquement.

3.4.2.3 Modèles basés sur KNN

[65] ont développé une méthode de migration en temps réel des VMs afin de minimiser le nombre d'hôtes ou de PMs actifs en utilisant les historiques de consommation actuels et passés. La méthode KNN a été utilisée pour prédire l'utilisation du CPU de chaque PM. Le taux de consommation des ressources a été prédit en utilisant un échantillon de l'ensemble des données de base ; chaque échantillon contient une variable d'entrée et une variable de sortie pour modéliser la relation entre les paramètres d'entrée/sortie. Leur méthode de prédiction vise à détecter la surcharge et la sous-charge des PMs afin d'améliorer la consolidation dynamique des VMs. Les résultats expérimentaux ont montré que le système proposé minimise la consommation d'énergie et maintient le SLA.

[96] ont également étudié, dans un autre travail, l'efficacité de l'utilisation des prévisions au moyen de traces de charge de travail pour effectuer la migration en temps réel des VMs. La solution qu'ils ont proposée consolide ces dernières de manière dynamique pour réduire la consommation d'énergie et les violations des SLAs. Ils ont aussi associé les VMs aux PMs en se basant sur l'utilisation actuelle et future des ressources, ce qui a permis d'optimiser le processus migrations des VMs. Ils ont prédit l'utilisation du CPU en utilisant KNN et un jeu de données historiques. En effet, chaque échantillon de données contient des variables d'entrée et de sortie pour représenter les valeurs d'utilisation du CPU à l'heure actuelle et à l'heure suivante respectivement. Les valeurs d'utilisation du CPU prévues sont agrégées et délimitées en utilisant un seuil supérieur de la capacité totale. Si les utilisations prévues du CPU et de la mémoire sont supérieures à la valeur du seuils fixés, la PM est considérée comme destination ayant une capacité suffisante pour répondre aux besoins actuels et futurs en ressources. Le modèle consiste à créer une liste des PMs triées en fonction de leur charge CPU par ordre décroissant. Toutes les VMs ont été déplacées à partir de la PM la moins chargée de la liste. La sélection des VMs est basée sur leur charge CPU ; ensuite, la VM la moins chargée est transférée vers la PM la plus chargée de la liste et le processus continuera à effectuer l'allocation des ressources.

De cette façon, la migration de certaines VMs provenant de PMs surchargées est effectué pour maintenir le SLA et la performance. Leur analyse expérimentale, en fonction des traces de charge de travail réelle, a validé l'efficacité du système pour la migration au niveau de la consommation énergétique et les SLAs.

3.4.2.4 Modèles basés sur ANN

Le réseau de neurones artificiels, *Artificial Neural Network* (ANN), consiste en une couche d'entrée, des couches cachées et une couche de sortie. Chaque couche comporte un certain nombre de nœuds (neurones), intégrés par une fonction d'activation sigmoïde. Cette technique fournit des mécanismes d'auto-apprentissage basés sur le concept des neurones, des connexions et des fonctions de transfert. Plusieurs travaux de recherche ont indiqué que le recours à l'ANN pour la prédiction constitue un moyen efficace pour gérer l'utilisation des ressources dans un centre de données, et il peut être utilisé pour exploiter et analyser des caractéristiques réelles des charge de travail. Les données historiques ou les traces des charges de travail ont été réparties en deux ensembles d'entraînement et de test. Dans la plupart des recherches, les ressources CPU nécessaires à chaque intervalle de temps dans les données historiques ont été utilisées principalement comme données d'entrée pour la prédiction de la consommation des ressources. En outre, elles ont été utilisées comme données d'entrée pour que l'ANN se propage dans le réseau avec un certain poids.

[31] ont développé une méthode de prédiction des besoins futurs en ressources en utilisant un modèle basé sur l'ANN et ont implémenté une politique de migration pour réduire le nombre de PMs en les faisant s'arrêter ou démarrer lorsque cela est nécessaire. Leur travail vise à optimiser la consommation énergétique dans le centre de données. Leur système allume (respectivement éteint) certains PMs lorsque la charge du système diminue (respectivement augmente).

En [119], pour prédire l'utilisation du CPU des VMs pendant des périodes courtes dans des environnements où un changement occasionnel de l'utilisation du CPU a lieu, les auteurs ont utilisé un réseau neuronal récurrent (RNN) pour prédire l'utilisation du CPU, car ils sont capables de prédire des séries temporelles bruitées. Le RNN est capable de mieux gérer les données d'entraînement et de test que la moyenne mobile, la marche aléatoire (*Random Walk Algorithm*) et la rétropropagation (*Backpropagation*). Ils ont indiqué que le RNN a pu avoir une plus grande précision lors de la prévision de 15 minutes, mais sa précision diminue de façon linéaire lorsque la prédiction est faite pour des points temporels plus longs. Cependant, il est capable de prédire 3 phases successives au cours du temps, avec une précision acceptable. En outre, si l'on considère uniquement le facteur CPU pour la prédiction de l'état des PMs, il peut s'avérer inefficace et une surcharge

d'autres ressources peut se produire, ce que ce modèle n' pas pris pas en compte.

Dans un travail plus récent, [144] ont essayé de comparer les techniques de prédiction linéaire et non linéaire avec des techniques basé sur l'ANN pour prévoir l'utilisation de la bande passante et l'utilisation du CPU pendant la migration en temps réel. Ils ont permis de minimiser l'utilisation de la bande passante et d'améliorer le rendement global du centre de données. Ils ont évalué leur schéma par des simulations portant sur une charge de travail de 600 PMs tirées des données de *Google Cluestering*[4] et ont indiqué que ANN est en mesure de fournir des prédictions précises sur l'utilisation de la bande passante et du CPU. En disposant de ces informations, ce système peut réduire le trafic, les violations des SLAs, ainsi d'améliorer l'efficacité énergétique au sein du centre de données.

Dans [137], les auteurs ont essayé de prévoir l'utilisation du CPU des PMs en utilisant l'ANN évolutif pour de courtes périodes et ont utilisé des algorithmes tels que l'évolution différentielle, l'optimisation par essaims particulaires (*Particle Swarm Optimization*) et la stratégie évolutive d'adaptation de la matrice de covariance (CMA-ES) pour former l'ANN. Ils ont montré que la CMA-ES converge plus rapidement lors de la phase d'entraînement ; cependant, dans la phase de test, l'évolution différentielle se comporte de la même manière que la CMA-ES.

3.4.2.5 Modèles basés sur RL

Le framework apprentissage par renforcement, *Reinforcement Learning* (RL), contient un espace d'état, un espace d'action et un signal de récompense. Dans cette méthode, un agent d'apprentissage effectue des interactions répétées avec l'environnement et reçoit une récompense ou une pénalité selon l'action effectuée. Généralement, il permet à l'agent d'apprentissage d'obtenir la récompense maximale en utilisant des interactions répétées par essais et erreurs afin qu'il puisse apprendre une politique pour sélectionner la meilleure action [160]. Le RL a été utilisé pour permettre à un agent d'apprendre la solution optimisée pour la planification et la migration des VMs dans un centre de données. Il apprend par interaction avec l'environnement dynamique et effectue une optimisation basée sur des états donnés.

[122] ont présenté une stratégie de migration en temps réel, *Reinforcement Learning Network Aware Live Migration* (RLLM), qui permet de surveiller la demande de la bande passante et de prendre les mesures appropriées en fonction de l'expérience acquise en cas de congestion du réseau. Leur système a fonctionné comme un système d'aide à la décision dans lequel un agent a pu programmer la migration des VMs en apprenant un temps optimisé. Le processus de migration dépend de l'utilisation de la VM dans un centre de données. Les résultats de leurs expériences ont montré qu'un agent peut apprendre l'état

4. Google Clustering : https://github.com/google/cluster-data

de la bande passante disponible au moment de la saturation maximale du réseau et être capable de programmer des VMs pour la migration d'une PM sous-utilisée au moment approprié en utilisant la bande passante disponible dans le centre de données. Conformément à l'algorithme RLLM, l'agent a effectué une migration en temps réel à partir de la PM sous-utilisée en tenant compte du niveau de la bande passante et de la réduction de la congestion dans le centre de données. Ils ont également présenté une approche d'efficacité énergétique basée sur le RL pour sélectionner les VMs à migrer à partir des PMs surchargées. L'utilisation globale des PMs a été calculée en utilisant le pourcentage d'utilisation de chaque VM à partir de données historiques. La violation des SLAs et la consommation d'énergie ont été utilisées comme une fonction de récompense. En outre, cette approche a été utilisée pour déterminer le moment approprié pour programmer la migration des VMs, étant donné qu'une ou deux migrations à partir des PMs sous-utilisées n'ont eu aucun effet sur le processus de gestion des ressources du Cloud [110].

De même, [116] ont introduit un algorithme de RL pour la migration en temps réel des VMs qui réduit simultanément le coût de la consommation d'énergie et améliore les performances. Ce modèle apprend la dynamique incertaine des charges de travail au fur et à mesure, et il utilise un schéma de réduction de la dimensionnalité pour projeter l'espace combinatoire explosif d'état-action dans un espace dimensionnel polynomial. Ces schémas lui permettent d'être évolutif et de fonctionner en temps réel. Les auteurs ont validé leur modèle de manière expérimentale et ont montré qu'il s'est révélé plus rentable et plus rapide que certaines autres approches existantes.

3.4.2.6 Modèles basés sur diverses approches statistiques

Dans le travail suggéré par [120], un schéma basé sur le modèle ARMA est proposé pour prédire les besoins en ressources d'une VM donnée, tout en analysant le taux d'utilisation des ressources de la PM. Le travail introduit une matrice de coûts de migration, au lieu d'une migration individuelle, et recommande un ensemble de migrations avec les meilleures performances d'un point de vue global. De plus, il améliore la méthode des "trois étapes" dans le schéma de migration traditionnel pour atteindre l'efficacité énergétique. Les expériences de simulation montrent que le schéma proposé peut effectivement réduire la consommation énergétique et améliorer la QoS dans une certaine mesure.

Dans [127], les auteurs ont appliqué *Autoregressive Integrated Moving Average* (ARIMA), qui est un modèle statistique populaire pour la prévision et l'analyse des séries temporelles, pour prévoir les *Dirty Pages* dans la migration *Pre-Copy* des VMs afin de diminuer leurs retransmissions et d'empêcher la dégradation des performances. Ils ont appliqué une approche de compression delta basée sur la distance de la pile pour obtenir un modèle de compression permettant une migration efficace des VMs. L'avantage du modèle ARIMA

est qu'il peut prévoir un nombre élevé de *Dirty Pages* et qu'il est capable de minimiser le temps total de migration et le temps d'arrêt.

Une solution de migration de VMs basée sur la méthode *Exponentially Weighted Moving Average* (EWMA), présentée par [100], a fournit deux techniques pour équilibrer la charge de travail du système dans de multiples migrations en temps réel qui se complètent mutuellement, et chacune d'entre elles produit de meilleurs résultats sous certaines charges. La technique *Pull* redistribue la charge de travail du système lorsque celle-ci est faible à moyenne, et la technique *Push* est plus efficace lorsque la charge de travail est moyenne à élevée. L'évaluation a montré que lors de l'ajout ou de la suppression d'un grand nombre de VMs dans le système, leur stratégie peut rééquilibrer le système en 4 à 15 minutes.

Pareillement, un algorithme de prédiction de charge pour décider si la migration sera effectuée ou non a été proposé par [189]. Une fois la décision est prise, l'algorithme trouve un hôte de destination approprié où la VM est déplacée. Pour cela, ils ont proposé une nouvelle approche à base d'une méthode de lissage exponentiel pour décider si un hôte particulier est approprié comme destination en fonction de sa charge future probable. Ils ont simulé leurs algorithmes dans *CloudSim* en utilisant des traces de charge de travail réelles et les ont comparés avec des algorithmes standards existants.

Dans la même veine, plusieurs schémas de migration des VMs basés sur le processus de *Markov* sont fournis dans la littérature. [117] ont fourni une approche de prévision basée sur le modèle de *Markov* afin de prédire la charge de travail des PMs et d'éviter les migrations prématurées des VMs grâce à la prédiction de l'utilisation quasi future du CPU des PMs. Dans ce schéma, l'utilisation du CPU des PMs est comparée avec deux seuils supérieur et inférieur pour déterminer leur état, puis un modèle de prédiction basé sur la méthode de *Markov* permet de prédire l'état futur des PMs. Leur algorithme détermine le temps de migration des VMs pour réaliser l'équilibrage de charge et la consolidation des serveurs. Les évaluations ont été effectuées sur l'ensemble de données *PlanetLab* et sur des charges aléatoires qui indiquent une diminution de violation des SLAs et du nombre de migrations des VMs.

Les mêmes auteurs ont présenté récemment le *MadMCHD*, qui prédit l'état futur des PMs à l'aide d'un modèle markovien de prédiction. Cette approche prédit l'utilisation future du CPU et, grâce à un algorithme basé sur les chaînes de *Markov*, empêche la migration prématurée des VMs en fonction d'un seuil d'utilisation dynamique, ce qui permet de mieux consolider les PMs et d'équilibrer leur charge. Il calcule et compare l'utilisation du CPU avec les seuils inférieur et supérieur pour définir l'état des PMs. L'algorithme détermine le moment où il faut migrer les VMs pour obtenir une consolidation des serveurs et un équilibrage de charge pour tous les états des hôtes [131].

Éventuellement, la modélisation des performances de la migration en temps réel a été largement abordée au cours de la dernière décennie. À notre connaissance, les travaux les plus avancés sur la modélisation de la migration en temps réel ont été récemment présentés par [151]. Un autre travail intéressant, présenté par [103], révèle de sérieux problèmes de performance de douze modèles existant et propose un modèle analytique précis qui surpasse la plupart des modèles discutés pour *Xen* et *KVM*. Cependant, les mesures prédites sont limitées au temps total de migration, au temps d'arrêt et au trafic total d'une migration en temps réel. En comparaison avec le travail de [103], notre approche fondée sur le *Machine Leanring* est non seulement beaucoup plus polyvalente pour prendre en considération tous les algorithmes de migration en temps réel couramment disponibles et six mesures cibles, mais aussi beaucoup plus précise.

3.5 Modèles de détection des anomalies de performance

3.5.1 Modèles basés sur l'apprentissage supervisé

Un certain nombre d'approches basées sur l'apprentissage supervisé a été proposée pour détecter les anomalies sur les systèmes Cloud. Des études précédentes ont utilisé différentes approches telles que la détection des anomalies/erreurs, l'analyse des registres, et ont employé de nombreuses techniques, y compris la classification et la régression.

[64] ont proposé un framework adaptatif et flexible de diagnostic pour identifier automatiquement les principaux indicateurs de défaillance et détecter les schémas de dysfonctionnement. Ils ont représenté les relations hiérarchiques entre les mesures et les événements ainsi que la manière dont ils sont liés aux anomalies. Ils ont appliqué l'algorithme *EdgeRank* pour déterminer les événements clés qui contribuent à une panne. Les auteurs ont assumé que leur approche fonctionne dans différents environnements pour détecter les failles potentielles. Ils ont évalué leur modèle en utilisant un système d'entreprise basé sur le Cloud en utilisant une liste de fautes injectées qui varient de l'environnement (par exemple, une VM ou un réseau) à la dégradation des applications. Ils ont pris en considération les environnements du Cloud privés et publics.

En utilisant les erreurs de CPU, de disque, de mémoire et de perte de paquets, [133] ont conçu une méthode de surveillance du système appelée système de détection des anomalies (ADS) qui détecte les erreurs liées aux services dans le Cloud en collectant des données relatives aux paramètres du système, notamment les erreurs de CPU, de mémoire, de disque et de réseau. La performance de l'ADS a été validée à l'aide d'une plateforme de

Cloud Computing basée sur *VMware*[5] et ses performances ont été évaluées à l'aide de forêt d'arbres décisionnels (*Random Forrest*), ANN, KNN et naïve bayésienne. Les résultats ont montré que l'algorithme de forêt d'arbres décisionnels offre la meilleure performance de détection des erreurs et le temps d'exécution le plus faible.

[107] ont introduit une approche visant à détecter les anomalies dans un environnement Cloud. Leur proposition est basée sur la prédiction du trafic pour obtenir des caractéristiques qui représentent le comportement approprié attendu du trafic du réseau dans le Cloud. Ces informations ont été ensuite utilisées conjointement avec un modèle de SVM qui est fourni avec ces caractéristiques. Selon les auteurs, la combinaison de ces deux outils représente une approche nouvelle et efficace pour détecter les événements anormaux dans l'environnement du Cloud.

Le travail proposé par [132] concerne un modèle de détection en ligne basé sur une méthode de recherche asystématique de paramètres intitulée SVM-Grid, dont la construction est basée sur la classification SVM. SVM-Grid est utilisée pour optimiser les paramètres au sein de la SVM. Les attributs appropriés des données de fonctionnement d'un système de Cloud sont sélectionnés en utilisant la corrélation de *Pearson* et l'analyse en composantes principales. Des stratégies de prédiction des anomalies, dans le centre de données, et de mise à jour des bases de données d'échantillons d'anomalies sont proposées pour optimiser le modèle et améliorer ses performances.

[109] ont présenté une évaluation de 13 algorithmes de classification *Off-Line* utilisés pour détecter des anomalies sur un ensemble d'hôtes. L'évaluation est basée sur un banc d'essai dans le Cloud qui exécute un scénario de services de fonction de réseau virtualisée. Les données nécessaires à l'entraînement et à l'évaluation des algorithmes de *Machine Learning* sont obtenues par des expériences d'injection d'anomalies contrôlées et un suivi approfondi des données entre les différentes couches. Une validation a montré que certains algorithmes ont obtenu une mesure moyenne exceptionnellement élevée.

3.5.2 Modèles basés sur l'apprentissage non supervisé

Un nombre important et croissant de publications ont étudié les techniques de *Clustering* pour produire une représentation synthétique du comportement du système pendant l'exécution de différentes charges de travail. Une étude menée par [18] a examiné une approche de *Clustering* soustractif pour apprendre le comportement d'un conteneur virtuel en cas de fluctuations dynamiques des charges de travail. La méthode de *Clustering* a été utilisée pour identifier la relation entre la charge de travail et les besoins en ressources du conteneur virtuel. Dans [47], les auteurs ont étudié un algorithme de *co-Clustering* pour

5. VMware : https://www.vmware.com/

identifier les modèles de charge de travail qui sont exécutés sur un serveur pendant un certain laps de temps, ce qui permet de prédire la charge de travail de chaque VM.

De même, un modèle hybride combinant *Kernel* FCM et ANN optimisé par un AG (GA-WElman) est présenté dans [60] ; le modèle a vise prédire la charge du CPU pour l'intervalle de temps suivant dans la séquence de charge en cours. D'autres auteurs ont introduit un framework de *Clustering On-Line* décentralisé pour la coexistence d'applications hétérogènes basées sur le Cloud ; c'est le cas de [26]. Ce framework permet une caractérisation en ligne de divers types de ressources en regroupant les demandes de charge de travail dans les files d'attente du centre de données. Tout de même, le *Clustering K-Means* et FCM ont été utilisés pour analyser les VMs et le comportement des utilisateurs de manière à ce que chaque demande de VM soit mise en correspondance avec un seul cluster [106].

[91] ont proposé un framework général pour la construction d'un système de contrôle hiérarchique qui répond aux exigences clés d'une application de contrôle des services. Il est utilisé avec différents types de sources de données pour détecter et diagnostiquer les problèmes de service. Ensuite, ils ont fait la démonstration d'une approche d'apprentissage non supervisé tel que le *Clustering* avec propagation d'affinités basée sur des principes pour construire un système de surveillance en utilisant le framework défini. Le modèle est utilisé pour surveiller le service interne de stockage de données distribué et de traitement par lots de *Microsoft*, qui comprend des dizaines de milliers de machines. Ils ont en outre construit un pipeline de bout en bout pour faire fonctionner le système de surveillance, depuis la collecte de données sur les machines individuelles jusqu'à un outil de visualisation que les opérateurs de service utilisent pour examiner les sorties de détection de leur système afin de se concentrer rapidement sur les problèmes.

[124] a proposé d'utiliser la forêt d'isolement, un algorithme de détection des anomalies non supervisé, pour détecter les problèmes dans les centres de données à grande échelle sur le Cloud. En utilisant les données de surveillance (le taux de transfert du disque, le taux de transfert du réseau, l'utilisation du CPU et l'utilisation de la mémoire), il a été conclu que la forêt d'isolement a pu détecter des anomalies et s'adapter à la périodicité et aux évolutions des données, cependant, la précision des prévisions n'a pas été rapportée.

[108] ont présenté un schéma sous forme de boîte noire de détection des anomalies de performance et de localisation des problèmes, basé sur les modèles historiques de comportement d'utilisation des ressources. Plus important encore, ce système prenait en considération les aspects multitâches. Il a utilisé les mesures de performance au niveau des processus et des threads pour détecter la fonction anormale spécifique et localiser la tâche qui est à l'origine de l'anomalie de performance. Il a profilé les mesures d'utilisation des ressources au niveau du thread et déduit le type d'applications multitâches

(par exemple multiprocessus et multi-thread). Ensuite, ils ont pu mener la procédure de détection d'anomalie au niveau de la tâche d'une manière agnostique sur le plan logiciel ; ils ont appliqué la technique d'apprentissage non supervisé DBSCAN pour apprendre le comportement régulier avec les métriques précises au niveau du profiling de la tâche dans les données historiques non étiquetées. Ils ont ensuite utilisé le résultat du *Clustering* pour détecter l'anomalie de performance potentielle. Ils ont établi la relation entre la tâche et la connexion au réseau.

Le travail, proposé par [195], a été lancé comme un banc d'essai pour évaluer différents algorithmes d'apprentissage non supervisé pour la détection précoce des défaillances. Ils ont choisi un simple jeu de données de vibration recueilli par un ventilateur d'extraction et ont adapté différents algorithmes d'apprentissage non supervisé tels que l'analyse des composants principale, le *Clustering* hiérarchique, le *K-Means* et le *FCM*. Ensuite, ils ont proposé une méthodologie pour évaluer différents algorithmes et choisir le modèle final.

3.6 Conclusion

Dans ce chapitre, nous avons abordé les travaux relatifs à nos contributions. Tout d'abord, nous avons besoins d'estimer la consommation de puissance d'un serveur. Les travaux qui ont introduit des approches d'estimation ne répondent pas à notre besoin. D'une part, certains modèles prennent en compte plusieurs composants du serveur, ce qui rend l'estimation plus coûteuse. D'autre part, les approches qui adoptent seulement le processeur comme étant le composant le plus consommateur d'énergie n'aboutissent pas nécessairement à l'exactitude requis. Notre approche porte sur une seule quantité et utilise ses statistiques qui peuvent être obtenues sur n'importe quelle plate-forme. De plus, un modèle stochastique nous permet de traiter les variations de résolution des mesures ainsi que les évolutions imprévisibles et inobservables à la fois dans le processeur et dans la charge de travail. Notre approche est particulièrement utile pour les serveurs dont la charge de travail présente des fluctuations considérables. Dans ce cas, l'utilisation du CPU révèle suffisamment de statistiques sur la puissance consommée par le processeur.

Ensuite, nous avons étudié les différents modèles de placement des VMs visant à minimiser la consommation énergétique dans le centre de données. Il s'avère qu'il reste un manque de cohérence entre les études techniques et le véritable système du centre de données car les profils des VMs, c'est-à-dire les variations de la charge de travail des VMs au fil du temps, n'ont pas été pris en compte. Aussi, l'hétérogénéité des profils énergétiques des PMs, et le placement des VMs ne tient pas compte des intervalles de temps plus larges. De plus, la comparaison entre les différentes stratégies de placement des VMs a été faite indépendamment et non pas dans les mêmes situations et pour des

charges de travail identiques. Motivés par ces lacunes de recherche, dans cette thèse, nous proposons et développons un framework de placement des VMs dans un centre de données avec des PMs hétérogènes en termes de capacités du CPU pour une période de temps quelconque. Par ailleurs, nous présentons une étude comparative entre quatre méthodes différentes pour résoudre le problème du placement des VMs, l'algorithme de séparation et évaluation progressive (*Branch&Bound*) et la programmation dynamique (PD) étant des approches exactes, ainsi que deux métaheuristiques, l'AG et l'OCF. Deux raisons motivent l'utilisation de ces quatre approches ; d'une part, même si l'algorithme *Branch&Bound* et la PD sont largement utilisés pour résoudre le problème de sac à dos, leur mise en œuvre dans le contexte du problème de placement de VMs est limitée. D'autre part, même si les approches AG et OCF sont adoptés pour la résolution du problème de placement des VMs dans la littérature, fournissant un aperçu de leurs performances, aucune étude comparative n'a été réalisée entre les deux méthodes. Autrement dit, dans ce travail, on vise à répondre à la question "jusqu'à quel point le recours aux approches métaheuristiques devient-il une obligation pour résoudre le problème de placement des VMs en tant que problème de sac à dos ?".

En plus des solutions de placement des VMs, nous nous sommes intéressés à étudier les travaux portant sur la migration en temps réel des VMs. Bien qu'il y ait eu de nombreux travaux sur la modélisation de cette techniques, nous estimons que toutes les approches proposées jusqu'à présent sont loin d'être idéales. Premièrement, il n'existe pas de modèle général permettant de prédire la dégradation des performances dans le cadre d'une migration en temps réel. Selon [95], le niveau de dégradation des performances d'une charge de travail migrante dépend fortement de la charge de travail elle-même. Ils ont prédit la dégradation des performances d'une VMs en se basant sur un historique des migrations enregistrées, ce qui rend cette méthode inadaptée aux charges de travail inconnues. Deuxièmement, les modèles existants ne rendent pas compte adéquatement de l'effet des ressources limitées sur les performances de la migration en temps réel. Une tentative de [78] n'a considéré que la bande passante réservée du réseau comme paramètre d'entrée ; d'autres paramètres importants tels que l'utilisation du CPU et de la mémoire sont ignorés. Troisièmement, les modèles déjà existants ne permettent pas de prévoir les besoins en ressources supplémentaires attendus du processus de migration en temps réel. Il peut s'agir d'informations importantes, notamment en cas de manque de ressources sur les serveurs concernés par la migration. Enfin et surtout, il n'existe pas de modèle qui prend en charge tous les algorithmes usuels de migration en temps réel.

Étant donné que les migrations en temps réel *Pre-Copy* et *Post-Copy* ont été introduites, de nombreuses techniques d'optimisation telles que la DTC, la DLTC et *CPU-throttling* ont été proposées. Certains travaux sont menés sur la sélection de la technique

d'optimisation appropriée et des paramètres importants pour la performance des techniques d'optimisation, mais les modèles actuels, à notre connaissance, ne prennent en compte plus de deux techniques de migration simultanément. Dans ce travail, nous abordons les problèmes mentionnés ci-dessus. Nous exploitons la puissance d'un ensemble de données de migration diverses et des techniques de *Machine Learning* pour modéliser plusieurs algorithmes de migration en temps réel et prédire des mesures importantes, notamment la perte de performance de la VM et la consommation de ressources supplémentaires prévue pendant la migration. Notre modèle prend en compte les algorithmes de migration en temps réel les plus répandus et prédit six mesures de performance avec une grande précision, ce qui représente un grand pas en avant par rapport à ce qui existe actuellement.

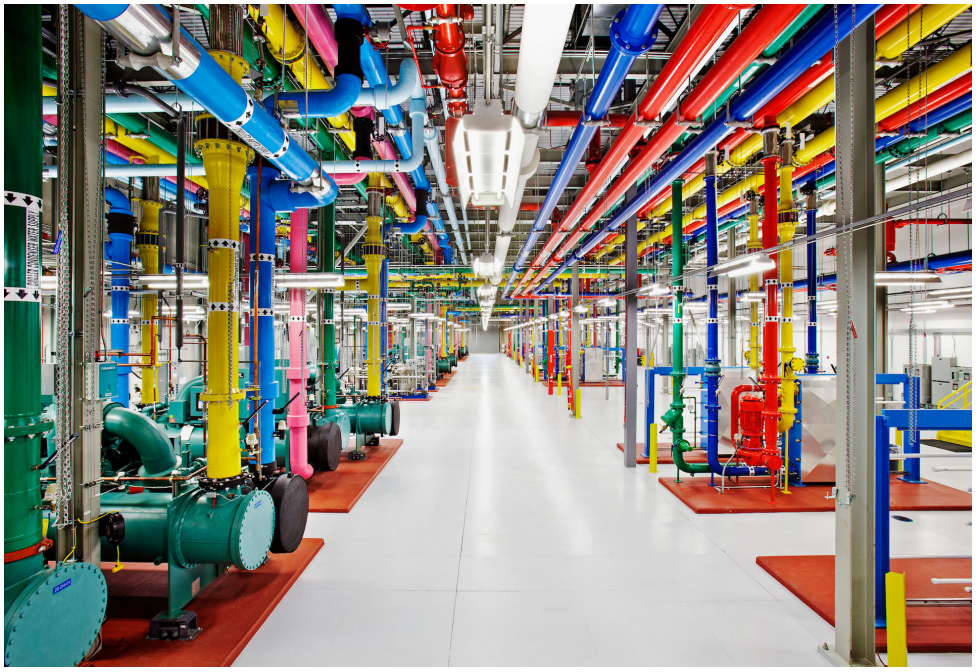
Enfin, notre contribution dans le domaine de détection et diagnostic de fonctionnement des VMs dans un centre de donnés consiste à introduire un nouveau algorithme de *Clustering* pour modéliser le processus de détection des comportements anormaux dans un centre de données. Le modèle permet d'identifier les charges de travail et les VMs qui se comportent de manière inhabituelle, et qui sont susceptible de provoquer des pannes, ou encore, de menacer la sécurité de tout le système.

4

Corrélation entre la consommation énergétique et le service rendu par le centre de données

---

Ce chapitre introduit le modèle stochastique proposé pour établir une relation de corrélation entre le service rendu par un centre de données et la consommation énergétique.



Google : https://www.numerama.com/tech/155071-visitez-un-datacenter-de-google-dans-une-video-360.html

4.1 Introduction

Plusieurs raisons expliquent les coûts associés à la consommation d'énergie d'un centre de données, qui peuvent être plus élevés que nécessaire. La raison principale est l'utilisation non optimale de l'énergie par les sous-composants des centres de données, ce qui entraîne une perte voire un gaspillage d'énergie [139]. La perte d'énergie fait référence à l'énergie fournie au système qui n'est pas directement consommée par les activités informatiques, comme la perte énergétique due au transfert et à la conversion d'énergie, au refroidissement, à l'éclairage, etc. Nous mesurons généralement la perte d'énergie par la métrique de l'efficacité de l'utilisation de l'énergie, *Power Usage Effectiveness* (PUE). La mesure PUE est le rapport entre la puissance totale consommée par une installation et la puissance consommée par les équipements informatiques (c'est-à-dire les serveurs, les équipements de réseau, et les dispositifs de stockage). La figure 16 montre la répartition générale de la consommation d'énergie dans un centre de données classique utilisant la PUE qui indique que le centre de données consomme deux fois plus d'énergie que celle nécessaire pour alimenter les serveurs, c'est-à-dire que 50% de la puissance totale de la structure est utilisée pour l'équipement informatique, et plus précisément les serveurs.

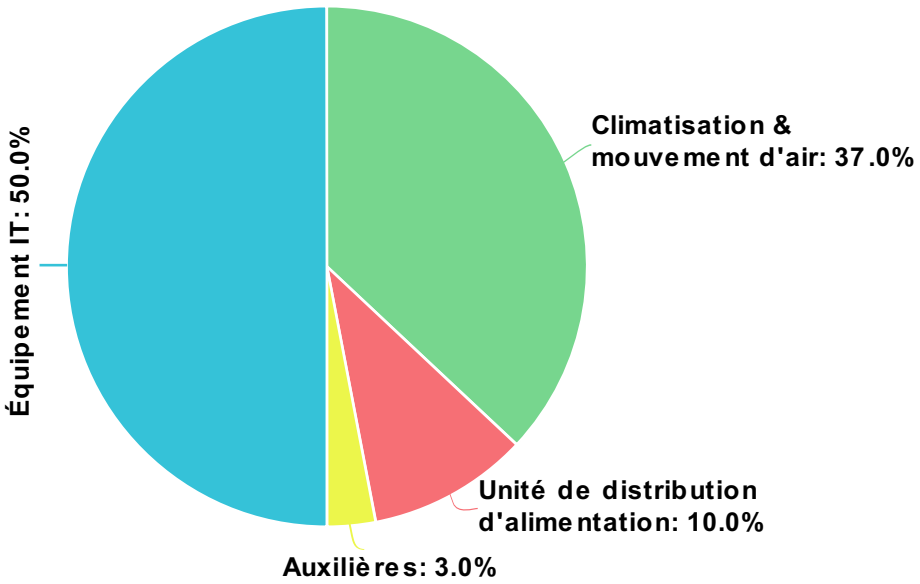


FIGURE 16 – Principales sources de perte de puissance dans un centre de données traditionnel [139].

Le serveur est un terme général décrivant un équipement informatique spécifique qui fournit une capacité de calcul et exécute des applications logicielles dans un environnement mis en réseau avec d'autres équipements informatiques, y compris d'autres serveurs.

La plupart des serveurs contiennent les principaux composants suivants : processeurs, mémoire, carte mère, dispositifs d'entrée/sortie, régulateurs de tension, unité d'alimentation et système de refroidissement. La consommation d'énergie totale de ces composants n'est pas constante, elle dépend plutôt du niveau de charge introduite dans chacun d'entre eux. En effet, la puissance consommée par le disque dur, la mémoire, le réseau, le CPU ou l'unité de refroidissement varie en fonction du nombre d'accès au disque et à la mémoire, de la masse de données envoyées ou reçues par le réseau et de la complexité du traitement effectué ou encore de la chaleur produite. Il y a également d'autres composants, mais nous supposons généralement que ce sont ceux qui contribuent majoritairement à la consommation électrique des serveurs dans un centre de données.

Par conséquent, la consommation d'énergie des serveurs est le principal facteur de flux d'énergie et de chaleur dans les centres de données, et de nombreux chercheurs ont développé ces dernières années des approches visant à accroître rapidement l'efficacité énergétique de ces composants, notamment les technologies d'allocation des ressources et de consolidation (chapitre 3). Ces technologies requièrent une compréhension des différents aspects des coûts. Un facteur important est l'estimation de la consommation d'énergie des VMs ou du serveur en fonction de la charge de travail exécutée. Étant donné que la consommation d'énergie est définie comme la consommation de puissance multipliée par le temps, il est donc utile d'estimer la consommation de la puissance. Les techniques orientées énergie utilisent généralement des modèles pour estimer la consommation d'énergie des processeurs multicœurs, ainsi que les politiques d'équilibrage de la charge des couches applicatives s'appuient également sur des modèles pour surveiller la consommation d'énergie de chaque serveur [88].

Les techniques de mesure directe de la puissance consistent soit en des compteurs matériels, soit des capteurs permettant de saisir la consommation d'énergie du serveur. Malgré leur précision, les mesures directes ne peuvent être applicables qu'aux centres de données homogènes à petite échelle. Dans le cas de centres de données hétérogènes à grande échelle, il est onéreux de mettre en place des dispositifs de contrôle ou un système d'acquisition de données, comme les *API IBM Active Energy Manager* [22], sur chaque serveur. Un intérêt important a récemment été consacré à la modélisation de l'énergie (voir section 3.2), et la plupart des contributions pertinentes ont établi des modèles d'estimation non seulement au niveau du serveur physique, mais aussi à une échelle plus raffinée. [206] a observé une forte corrélation entre la consommation d'énergie et l'utilisation du CPU dans un processeur multi-coeurs, en particulier lors de l'exécution d'applications qui nécessitent intensivement des ressources CPU. De même, certaines études signalent que le processeur est un composant important qui consomme l'énergie (figure 13).

Dans ce travail, nous nous concentrons principalement sur la modélisation de la consommation électrique du serveur en la résolvant en fonction d'un seul composant. Compte tenu de la nature des charges de travail, et comme le processeur représente généralement plus de 50% de la consommation totale du système, nous assumons une forte corrélation entre la charge du processeur et la consommation globale du système. Nous proposons donc un modèle d'estimation stochastique de la puissance d'un serveur afin d'évaluer une relation statistique entre l'utilisation du processeur et la consommation d'énergie. Ensuite, nous développons notre modèle pour considérer un cluster au lieu d'un seul serveur. Généralement, l'objectif principal est le suivant : en supposant que nous fournissons à un serveur une certaine quantité de charge de travail intensive en CPU pendant une période donnée et qu'il l'exécute. Nous démontrons qu'en modélisant le serveur comme un système non linéaire avec une entrée stochastique, nous pouvons définir une fonction de corrélation entre sa charge CPU et sa consommation électrique. Notre modèle s'inspire de la méthode proposée par [206], où un modèle stochastique d'estimation de la consommation d'énergie pour un processeur multi-coeur est introduit. Notre modèle diffère de la méthode proposée précédemment car il estime la consommation d'énergie du serveur pour n'importe quelle charge de travail. En outre, nous estimons la consommation d'énergie d'un cluster (groupe de serveurs) à partir du résultat de l'estimation de la puissance des serveurs. Nous résumons les principales contributions de notre travail comme suit :

- Nous introduisons d'abord un modèle de consommation électrique d'un serveur basé sur les caractéristiques statistiques de la charge de travail et de la puissance du processeur. Le modèle est adaptable à chaque nouvelle charge de travail.
- Les résultats du modèle constitueront la base d'une estimation à partir de laquelle nous déduirons la puissance consommée par l'ensemble du cluster lors de l'exécution des charges de travail intensives.
- Nous validons notre modèle d'estimation en utilisant l'outil de simulation *CloudSim*. Nous simulons le comportement de la puissance du serveur pendant l'exécution d'une charge de travail intensive, en terme du CPU ; ensuite, nous comparons la puissance estimée obtenue en appliquant le modèle proposé avec la puissance mesurée générée grâce au benchmark *SpecPower*.

Dans la suite du présent chapitre, nous allons aborder la problématique précédemment indiquée. Pour cela, nous nous appuyons essentiellement sur des formulations mathématiques pour élaborer le modèle théorique, ainsi que des simulations pour valider les résultats estimés en utilisant ce modèle.

La section 4.2 annonce les étapes suivies pour élaborer le modèle proposé. Le modèle consiste en un processus stochastique qui considère la consommation de puissance et la charge de travail comme des variables aléatoires suivant une distribution inconnue. Pour

estimer cette distribution, deux fonction non-paramétriques sont utilisées pour obtenir l'expression de la fonction de répartition de chaque variable.

Dans la section 4.3, nous présentons les résultats expérimentaux. Nous décrivons les différents processus de simulation, ainsi qu'on apporte une comparaison entre les résultats obtenus expérimentalement et les valeurs trouvés théoriquement soit des statistiques associés à chacune des variables aléatoires (fonction de densité par exemple) ou bien les variations de la puissance au cours du temps lors de l'exécution de la charge de travail. Pour conclure, nous synthétisons dans la section 4.4 le travail élaboré, et nous ouvrons de nouvelles perspective et voie de recherche dans le futur.

4.2 Modélisation de la corrélation entre la consommation énergétique et la QoS

Grâce à des expériences répétées, une forte corrélation entre la consommation de l'énergie et l'utilisation du CPU d'un processeur a été observée . La figure 17 illustre un exemple de trace de la charge de travail d'un serveur révélant les relations entre l'utilisation du CPU, la consommation électrique du CPU et la consommation électrique totale du serveur. L'expérience montre qu'à mesure que le taux de demande augmente, l'utilisation du processeur ainsi que la consommation de l'énergie augmentent. Également, nous remarquons que la contribution du CPU atteint environ 60% de la consommation électrique totale, ce qui confirme le rôle dominant connu du CPU dans le profil de la consommation électrique des serveurs. Cette observation inspire les efforts visant à concevoir des contrôleurs de l'utilisation des processeurs orientés énergie et adaptant la vitesse de traitement à la charge de travail observable à court terme.

Par ailleurs, un processeur peut consommer une quantité variable de puissance lorsqu'il exécute différents types de tâches, voire même en mode de veille, même s'il peut avoir les mêmes statistiques d'utilisation du CPU pour ces tâches. Les systèmes d'exploitation fournissent généralement un moyen d'obtenir un pourcentage d'utilisation du CPU. Par exemple, un taux d'utilisation du CPU de 40% peut être utilisé soit pour l'exécution de programmes utilisateur ou de programmes système, soit pendant l'état d'inactivité. Généralement, même si une CPU en veille ne consomme pas la même quantité d'énergie qu'une CPU qui exécute des programmes, elle consomme un pourcentage important de cette énergie. Ainsi, à chaque instant, le processeur peut être dans l'un des deux modes de consommation d'énergie ; soit il exécute des programmes et consomme 100% de sa puissance spécifiée, soit il est en mode veille et consomme une fraction importante de cette même puissance, par exemple 60%. Par conséquent, ces statistiques doivent être prises en compte dans la modélisation de la consommation d'énergie. Nous montrerons dans

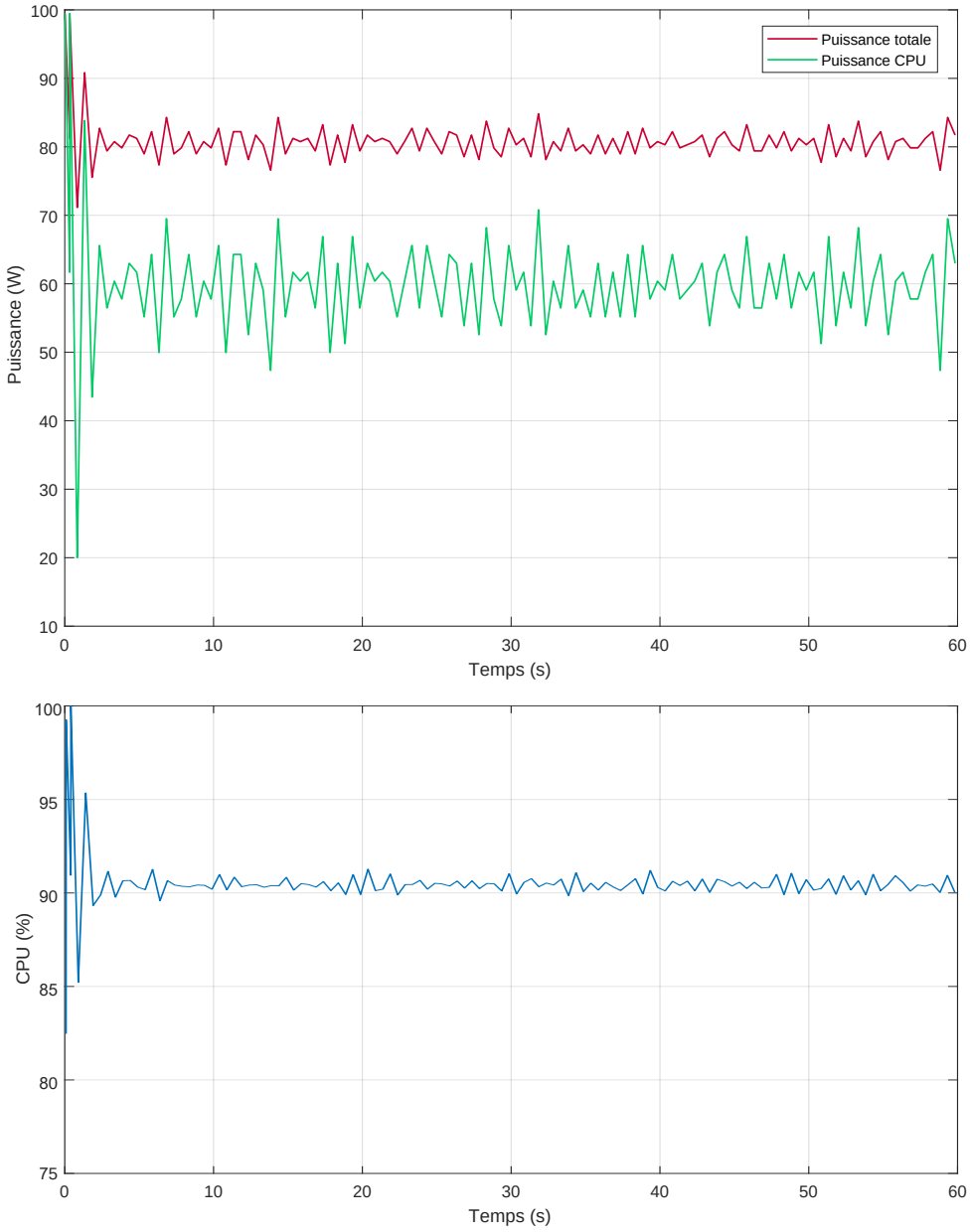


FIGURE 17 – Relation entre l'utilisation du CPU, la puissance consommée par le processeur et la puissance totale utilisée par le serveur.

cette thèse que notre modèle capture avec précision les statistiques de la consommation d'énergie d'un processeur à partir des statistiques de la charge de travail du CPU, étant donné que cette dernière subit des fluctuations importantes, y compris l'existence d'une consommation d'énergie en mode de veille. Nous testons notre modèle avec deux plates-formes de serveurs et différents benchmarks fournis par le projet *CloudSim* et les résultats observés sont intéressants.

4.2.1 Contexte expérimental

La consommation d'énergie d'un serveur comporte des aspects statique et dynamique. La consommation statique (indépendante de la charge de travail du processeur) est nécessaire pour alimenter le processeur et le préparer à effectuer une certaine tâche. Cette consommation n'est pas réellement statique mais elle peut être considérée ainsi car sa variation dans le temps est faible par rapport à la variation de la consommation dynamique. En revanche, l'aspect dynamique dépend de la charge de travail du serveur et représente en outre la somme totale des consommations dynamiques des différents composants architecturaux du serveur.

Une charge de travail consiste en une demande de haut niveau adressée au processeur et comprend plusieurs tâches de bas niveau telles que des opérations sur des nombres entiers, des opérations en virgule flottante, des opérations de lecture et d'écriture sur la mémoire ou le disque, etc. En général, la charge de travail du processeur d'un serveur Internet doit être modélisée comme un processus stochastique, car il est difficile de déterminer à l'avance le type et la taille de la charge de travail qui parviendra au processeur. Nous admettons qu'il existe une forte corrélation entre les statistiques de la charge de travail de ce dernier et les statistiques de son utilisation. Par conséquent, l'examen de ces dernières nous permet d'estimer la consommation d'énergie du serveur liée à la charge de travail. En raison de cette affirmation, nous utilisons alternativement les termes "charge de travail" et "utilisation du processeur".

La prise en charge d'un scénario de simulation orienté puissance dans *CloudSim* est l'une des principales innovations, et elle a été largement utilisée par la communauté des chercheurs pour publier les résultats optimisés de leurs propositions d'algorithmes concernant la migration des VMs, les SLAs, l'allocation des ressources, la réduction de la consommation d'énergie, etc. Comme notre objectif est d'améliorer la qualité d'un modèle d'estimation, nous devons exécuter nos méthodes en simulation. Il est avantageux pour nous que l'environnement *CloudSim* fournit la possibilité de présenter un centre de données composé d'un nombre spécifique de serveurs physiques, où chaque serveur a une architecture multicœur, une quantité adéquate de mémoire vive, de bande passante et de stockage. En général, la performance des CPUs est définie en millions d'instructions par

seconde (MIPS) et au lieu de disques locaux, le stockage est défini comme un stockage en réseau (NAS) ou un réseau de stockage (SAN). Nous devons réaliser un modèle de puissance pour estimer la consommation électrique en fonction de l'utilisation du CPU, en utilisant des données réelles sur la consommation électrique fournies par le résultat du benchmark *SpecPower*. Ainsi, nous avons adopté deux configurations de serveur : "*PowerModelSpecPowerIbmX3550XeonX5675.java*" [1] qui modélise la machine serveur IBM X3550 avec 2 processeurs XeonX5675 (6 cœurs chacun) ainsi que la capacité mémoire de 16 Go et "*PowerModelSpecPowerIbmX3250XeonX3480.java*" qui modélise la machine serveur *IBM X3250* avec 4 processeurs *XeonX3480* (6 cœurs chacun) et capacité mémoire de 16Go.

4.2.2 Modèle stochastique

Avant d'entamer la démarche de notre modèle, nous exposons la façon dont les variables sont représentées. Une lettre minuscule $\boldsymbol{w}$ en gras fait référence à la variable aléatoire associée à la charge de travail, ainsi que $\boldsymbol{p}$ désigne la variable aléatoire associée à la puissance consommée. Une lettre minuscule normale w_i (respectivement p_i) désigne un nombre réel associé à la variable aléatoire $\boldsymbol{w}$ (respectivement $\boldsymbol{p}$). Un $\boldsymbol{F}$ majuscule désigne la fonction de distribution probabilistique (PDF), ou simplement une fonction de répartition tandis qu'un f minuscule désigne la fonction de densité probabilistique (ou simplement une fonction de densité). Ainsi, la question principale à laquelle nous répondons peut être formulée comme suit : supposons que nous fournissions à un serveur une charge de travail et qu'il l'exécute dans un certain délai. Si nous étudions la distribution ou les fonctions de densité de l'utilisation du CPU et la consommation de puissance du serveur pendant cette période, est-il possible d'estimer une relation quantitative entre l'utilisation du CPU et la consommation de puissance ? Nous montrons qu'en modélisant le serveur comme un système non linéaire avec une entrée stochastique, nous pouvons répondre à cette question. De ce fait, l'utilisation du CPU et la consommation dynamique d'un processeur seront caractérisées comme des variables aléatoires $\boldsymbol{w}$ et $\boldsymbol{p}$, respectivement, et notre objectif sera d'exprimer les statistiques de $\boldsymbol{p}$ en fonction de $\boldsymbol{w}$ et vice versa.

Pour préciser notre propos, nous déterminons la relation entre les deux variables aléatoires étant donné que nous disposons de leurs fonctions de répartition et de densité, c'est à dire, on nous donne la distribution de $\boldsymbol{w}$ et nous souhaitons trouver une fonction $g(\boldsymbol{w})$ telle que la distribution de $\boldsymbol{p} = g(\boldsymbol{w})$ soit égale à une fonction donnée $\boldsymbol{F}(\boldsymbol{p})$. C'est généralement le cas lorsque nous essayons d'estimer la consommation de puissance de la charge de travail d'un processeur. Initialement, nous pouvons donner au serveur une charge de travail de statistiques connues et mesurer et analyser la consommation de puissance correspondante. Ensuite, nous établissons la relation entre les deux variables aléatoires. Une

1. https://github.com/Cloudslab/cloudsim

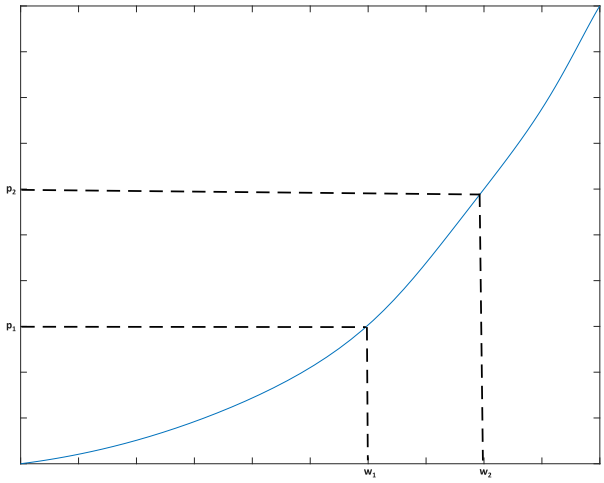


FIGURE 18 – La relation un-à-un entre $\boldsymbol{p}$ et $\boldsymbol{w}$ et la nature monotone des fonctions de répartition pour déterminer une relation quantitative entre les deux variables aléatoires.

fois la relation établie, la consommation énergétique du serveur peut être estimée pour n'importe quelle charge de travail. Si la relation entre $\boldsymbol{p}$ et $\boldsymbol{w}$ est assimilable à une fonction de type un-à-un, soit que chaque élément du domaine de $\boldsymbol{p}$ correspond exactement à un élément du domaine de $\boldsymbol{w}$, alors $P\{\boldsymbol{p} \leq p_i\}$ est égal à $P\{\boldsymbol{w} \leq w_i\}$ car $\boldsymbol{p} \leq p_i$ si et seulement si $\boldsymbol{w} \leq w_i$. Ceci peut être mieux visualisé dans la figure 18 qui illustre une fonction de type un-à-un. D'après la figure, il est apparent que la valeur p_2 correspond à w_2. Ainsi, $P\{\boldsymbol{p} \leq p_2\}$ correspond à $\boldsymbol{w} \leq w_2$. Parallèlement, $P\{\boldsymbol{p} \leq p_1\}$ correspond à $\boldsymbol{w} \leq w_1$ et nous pouvons en conclure que pour une fonction de type un-à-un, nous avons :

$$F(p_i) = F(w_i), \qquad \forall i \in \mathbb{R} \tag{4.1}$$

En utilisant alors l'équation 4.1, et puisque la fonction de répartition est bijective, nous pouvons exprimer $\boldsymbol{p}$ en termes de $F(p)$ et $F(w)$ comme suit :

$$\boldsymbol{p} = F_p^{-1}(F_w(\boldsymbol{w})) \tag{4.2}$$

Où F^{-1} représente la fonction réciproque de la fonction de répartition F.

4.2.3 Estimation des fonctions de répartition

Vue qu'un centre de données fourni des services différents, les distributions des deux variables aléatoires (charge de travail et puissance) peuvent être fluctuantes. Ainsi, il est difficile de décrire avec précision l'échantillon en utilisant une distribution paramétrique qui nécessite que le type de distribution des données soit connu, c'est à dire que dans le cas où un modèle paramétrique n'est pas disponible ou ne correspond pas aux données,

nous recourons à des méthodes non paramétriques pour apprendre un modèle probabilistique donné. Apprendre un modèle sans se baser sur une paramétrisation donnée est un défi, car nous devons estimer la distribution complète à partir des échantillons disponibles. Sans autres hypothèses, ce problème est mal posé ; de nombreuses distributions différentes auraient pu générer les données. Cependant, avec suffisamment d'échantillons, il est possible d'obtenir des modèles qui caractérisent assez précisément la distribution en question. Dans ce travail, nous proposons l'utilisation de deux méthodes pour estimer la fonction de répartition de chaque variable aléatoire. Il s'agit de la distribution à noyaux (*Kernel Distribution*) et la fonction de distribution cumulative empirique (ECDF) qui sont décries comme suivant :

- Une distribution à noyau : produit une estimation non paramétrique de la densité de probabilité qui s'adapte aux données, plutôt que de sélectionner une densité avec une forme paramétrique particulière et d'estimer les paramètres. Cette distribution est définie par un estimateur de densité du noyau, une fonction de lissage qui détermine la forme de la courbe utilisée pour générer la fonction de répartition, et une valeur de largeur de bande qui contrôle le lissage de la courbe de densité résultante. La distribution à noyau construit une fonction pour représenter la distribution de probabilité en utilisant les données de l'échantillon, et elle calcule la somme des fonctions de lissage des composantes pour chaque élément de données afin de produire une courbe de probabilité lisse et continue. L'estimation de la densité à noyaux avec une largeur de bande h pour une distribution de $x_1, x_2, ..., x_n \in \mathbb{R}$ est donnée comme suivant :

$$F_{h,n}(x) = \frac{1}{nh} \sum_{i=1}^{n} k(\frac{x - x_i}{h}) \tag{4.3}$$

où k est une fonction du noyau centrée à l'origine qui satisfait :

$$k(x) > 0 \qquad \forall x \in \mathbb{R} \tag{4.4}$$

$$\int_{\mathbb{R}} k(x)dx = 1 \tag{4.5}$$

Différents types de fonctions du noyau sont communément utilisés, le lecteur intéressé est renvoyé à [205]. Les tests d'approximation effectués ont montré que la fonction Guassienne est la plus adéquate pour notre jeu de données ; elle est définie

comme suivant :

$$k(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x^2\right) \tag{4.6}$$

- Une fonction de distribution cumulative empirique estime la fonction de répartition d'une variable aléatoire en attribuant une probabilité égale à chaque observation dans un échantillon. Grâce à cette approche, l'ECDF constitue une fonction de distribution cumulative discrète qui crée une correspondance exacte entre la distribution estimée et la distribution actuelle des données de l'échantillon. Pour une distribution de $x_1, x_2, ..., x_n \in \mathbb{R}$, l'ECDF est définie comme suivant :

$$F_n(x) = \frac{1}{n} \sum_{i=1}^{n} I_{\{x_i < x\}} \tag{4.7}$$

  Autrement dit, l'ECDF est une estimation de la fonction de distribution cumulative qui génère des points dans l'échantillon et elle converge avec une probabilité de 1 vers cette distribution sous-jacente.

4.3 Résultats expérimentaux

Pour illustrer la manière dont nous avons appliqué le concept développé précédemment pour établir la relation entre $\boldsymbol{p}$ et $\boldsymbol{w}$ d'un serveur, nous utilisons un ensemble des modèles de puissance de *CloudSim* pour concevoir un environnement de simulation composé d'un centre de données et d'un nombre arbitraire d'utilisateurs de Cloud. Le centre de données peut comporter un ou plusieurs serveurs, dont chacun est modélisé indépendamment avec des caractéristiques (par exemple, le serveur IBM X3550 avec 2 processeurs XeonX5675 (6 cœurs chacun) ainsi que la capacité mémoire de 16 Go). Les utilisateurs du Cloud sont modélisés en tant qu'un ensemble de VMs à allouer par l'infrastructure gérée par les hôtes et un profil d'utilisation. De plus, chaque VM peut être configurée à l'aide des caractéristiques telles que la consommation du CPU et de la mémoire, le type d'hyperviseur et un planificateur de Cloudlet.

Comme la modélisation de la charge de travail joue un rôle décisif sur les résultats des expériences de simulation, il est nécessaire d'utiliser un modèle aussi similaire que possible aux environnements réels des centres de données. Par conséquent, les expériences présentées dans cette étude ont fait appel à des données collectées à partir du framework *SpecPower* qui constitue la base de la génération de ces modèles de puissance. La charge de travail exercée dans les environnements simulés est modélisée dans *CloudSim* sous la forme de tâches représentées par la classe "*Cloudlet.java*" et à exécuter sur des VMs qui

sont allouées dans des hôtes. D'autre part, les traces extraites de *SpecPower* contiennent des informations concernant l'utilisation des ressources (par exemple le CPU) et sont présentées comme des *jobs* exécutés sur des machines réelles du cluster supervisé. Afin d'utiliser ces traces dans les expériences de simulation, les informations provenant des *jobs* ont été reproduites sous forme de Cloudlets. Ainsi, il a été possible de simuler des environnements comportant jusqu'à 100 hôtes avec une charge de travail similaire à l'utilisation d'un centre de données réel.

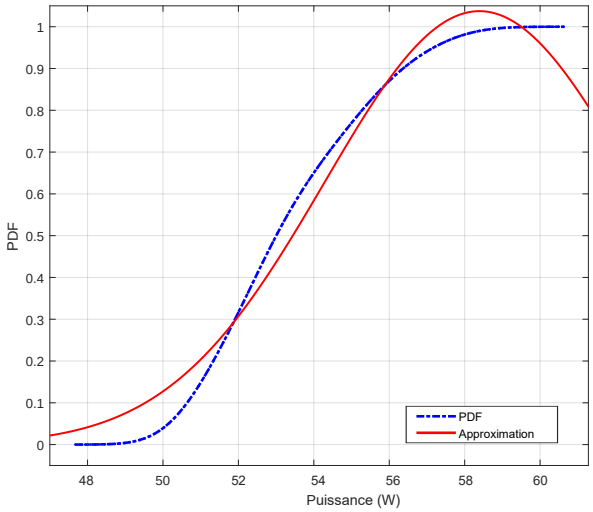
Principalement, les expériences ont exploité les traces de consommation de puissance collectées à partir d'un benchmark de machines réelles. Pour utiliser les informations de ce benchmark dans les expériences, nous avons dû développer une nouvelle classe qui étend la classe "*PowerModelSpecPower.java*" de *CloudSim* pour simuler un serveur *IBM X3550* avec 2 processeurs *XeonX5675*. Par conséquent, le centre de données représenté dans l'expérience de cette étude est composé d'un ensemble de machines de même type ; et vu que les données de référence fournissent des informations sur la consommation électrique en Watts en fonction de niveaux de charge discrets appliqués à une machine, la création de ce nouveau modèle de consommation de puissance sur *CloudSim* était directe car le framework traite déjà la consommation électrique en fonction des niveaux de charge appliqués aux hôtes simulés.

4.3.1 Relation entre w et p

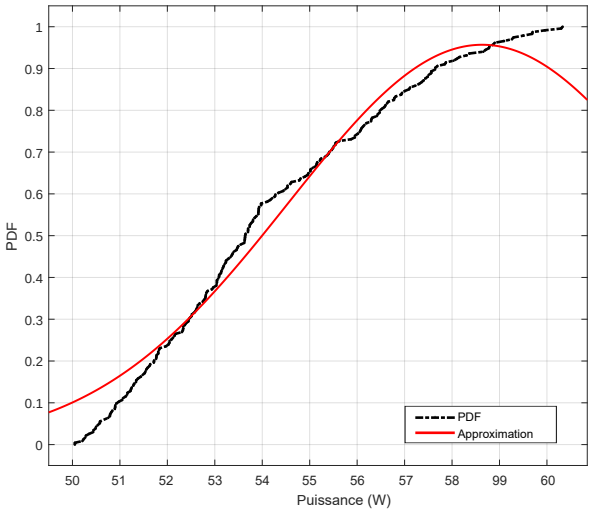
Après avoir obtenir les statistiques relatives à l'utilisation du CPU et la consommation énergétique du serveur, nous avons effectué une approximation de la fonction de répartition des variables aléatoires associées à la consommation de puissance et le taux d'utilisation du CPU en utilisant deux méthodes non-paramétriques décrites dans la sous-section précédent (notons qu'on a utilisé les fonctions introduites par les librairies de Matlab). La figure 19 présente l'approximation appliquée à la fonction de distribution de la consommation énergétique ($\boldsymbol{p}$), ainsi que l'approximation appliquée à la fonction de distribution du CPU ($\boldsymbol{w}$). À partir de ces graphes, nous pouvons conclure les expressions suivantes :

$$\tilde{F}(\boldsymbol{p}) = a_p \exp\left(-\left(\frac{b_p - \boldsymbol{p}}{c_p}\right)^2\right) \tag{4.8}$$

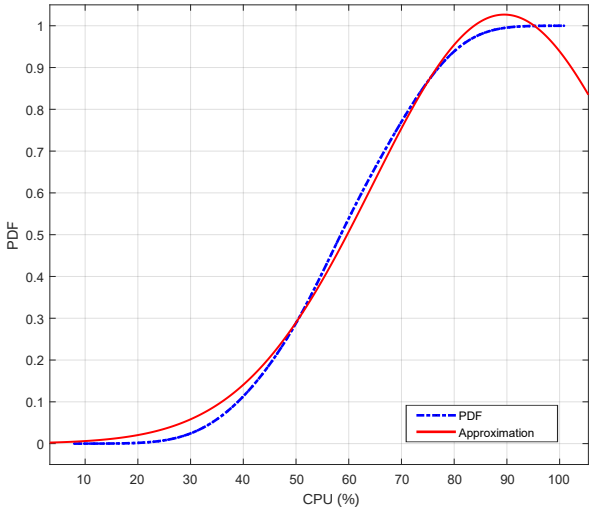
$$\tilde{F}(\boldsymbol{w}) = a_w \exp\left(-\left(\frac{b_w - \boldsymbol{w}}{c_w}\right)^2\right) \tag{4.9}$$



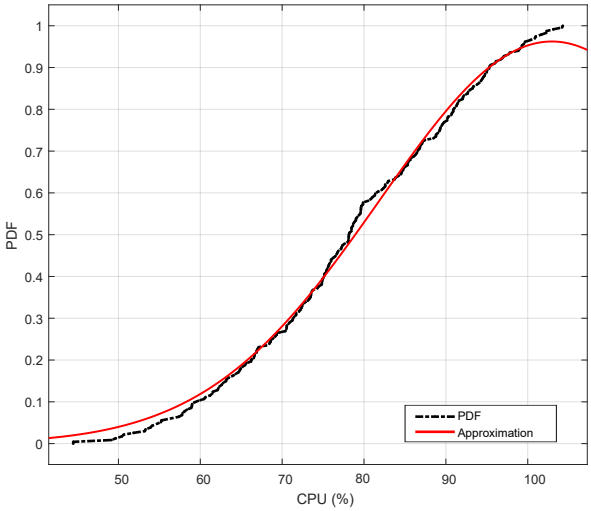
(a) Approximation de $\boldsymbol{p}$ par la fonction de distribution à noyaux.



(b) Approximation de $\boldsymbol{p}$ par l'ECDF.



(c) Approximation de $\boldsymbol{w}$ par la fonction de distribution à noyaux.



(d) Approximation de $\boldsymbol{w}$ par l'ECDF.

FIGURE 19 – Approximation de la fonction de distribution de $\boldsymbol{p}$.

Sachant que a_p, b_p, c_p, a_w, b_w, et c_w sont des constantes correspondantes au jeu de données traité.

Il est clair que F est une fonction strictement croissante sur l'intervalle $[0, 1]$, i.e, elle admet une fonction réciproque donnée par :

$$\tilde{F}_p^{-1}(\boldsymbol{p}) = c_p^2 \sqrt{\ln(a_p) - \ln(\boldsymbol{p})} + b_p \tag{4.10}$$

En appliquant l'équation 4.2 définie précédemment, nous obtenons la relation suivante :

$$\boldsymbol{p} = b_p - c_p \sqrt{\ln\left(\frac{a_p}{a_w}\right) + \left(\frac{\boldsymbol{w} - b_w}{c_w}\right)^2} \tag{4.11}$$

L'équation 4.11 est la relation souhaitée que nous avons voulu établir entre la charge de travail du CPU et la consommation de puissance. En se basant sur cette relation, il est maintenant possible de déterminer la distribution et la densité de la puissance $\boldsymbol{p}$ en fonction de la distribution et de la densité de n'importe quelle charge de travail $\boldsymbol{w}$. Nous avons montré ci-dessus que $F(\boldsymbol{p})$ peut être exprimé par $P\{\boldsymbol{p} \leq p\} = P\{g(\boldsymbol{w}) \leq p\}$, d'où :

$$\tilde{F}(\boldsymbol{p}) = P\left\{\left(b_p - c_p \sqrt{\ln\left(\frac{a_p}{a_w}\right) + \left(\frac{\boldsymbol{w} - b_w}{c_w}\right)^2}\right) \leq p\right\}, \qquad \boldsymbol{w} \leq b_w \tag{4.12}$$

L'expression de l'équation 4.12 en fonction de $F(\boldsymbol{w})$ résulte en :

$$\tilde{F}_w(p) = P\left\{\boldsymbol{w} \leq \left(b_w - c_w \sqrt{\ln\left(\frac{a_w}{a_p}\right) + \left(\frac{b_p - \boldsymbol{p}}{c_p}\right)^2}\right)\right\} \tag{4.13}$$

ce qui correspond à $\tilde{F}_w\big(b_w - c_w\big(\ln\left(\frac{a_w}{a_p}\right) + \left(\frac{b_p - p}{c_p}\right)^2\big)^{\frac{1}{2}}\big)$. De même, la densité de $\boldsymbol{p}$ peut être exprimée sous la forme :

$$\tilde{f}(\boldsymbol{p}) = \frac{2(\boldsymbol{p} - b_p)}{c_p^2} \tilde{f}_w\left(\left(\frac{b_p - p}{c_p}\right)^2\right) \tag{4.14}$$

En exploitant la relation exprimée dans l'équation 4.14 et les paramètres obtenus à partir des données, il est possible de calculer la fonction de répartition et la fonction de densité de $\boldsymbol{p}$ pour une charge de travail suivant une distribution probabilistique quelconque. Nous l'avons démontré en calculant la fonction de densité théorique de $\boldsymbol{p}$ pour une charge de travail normalement distribuée (en raison de simplification de calcul) et en comparant les résultats théoriques avec ceux que nous avons obtenus par des expériences. Pour cela, nous avons employé des charges de travail normalement distribuées faites de façon personnalisée

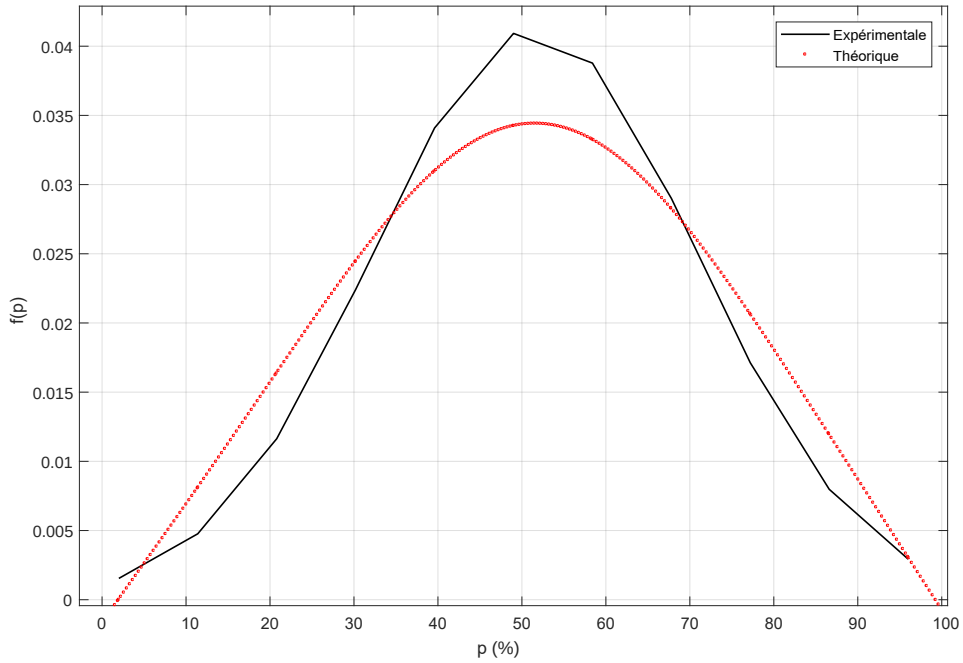


FIGURE 20 – La fonction de densité estimée théoriquement et la fonction de densité obtenue expérimentalement pour une puissance normalement distribuées.

du benchmark *SpecPower*.

En se basant sur la fonction de densité estimée théoriquement et la fonction de densité obtenue expérimentalement pour les charges de travail normalement distribuées, nous constatons que même si la charge de travail du test était dissimilaire à celle utilisée pour construire le modèle, sa fonction de densité associée à la consommation de puissance a pu être estimée avec une précision remarquable (avec une erreur moyenne de 6%), comme exposé dans la figure 20, quelque soit le type de serveur. L'erreur peut être due à l'approximation de $\boldsymbol{F(w)}$ et $\boldsymbol{F(p)}$ actuelles avec un outil d'ajustement de courbe. Il est possible de réduire cette erreur, mais cela peut rendre le modèle plus complexe en terme de calcul.

4.3.2 Estimation de la puissance $\boldsymbol{p}$ pour un serveur

Après avoir établi la relation entre $\boldsymbol{p}$ et $\boldsymbol{w}$, nous avons testé la validité de notre modèle en générant des charges de travail de distributions arbitraires et en les exécutant sur les deux serveurs *IBM X3550* à 2 processeurs *XeonX5675* et *IBM X3250* à 4 processeurs *XeonX3480*. Nous avons utilisé le benchmark *SpecPower* standard pour tester notre modèle qui stresse le processeur de différentes manières.

L'équation 4.11 peut être directement appliquée pour estimer la relation instantanée entre $\boldsymbol{w}$ et $\boldsymbol{p}$. Les figures 21 et 22 visualisent la consommation de puissance instantanée actuelle et estimée des serveurs *IBM X3550* à 2 processeurs *XeonX5675* et serveur *IBM*

X3250 à 4 processeurs *XeonX3480*, respectivement, lors de l'exécution du benchmark *SpecPower*. Selon ces deux figures, nous pouvons déduire les affirmations suivantes :

- Le taux d'erreur obtenu pour les deux méthodes ne dépasse pas 2% pour une charge de travail faible, et ne dépasse pas 1% pour une charge de travail plus élevée.
- Les résultats montrent que l'estimation est plus précise dans le cas où l'intensité de la charge de travail est élevée pour les deux méthodes.
- Les résultats obtenus par la considération de la fonction de distribution à noyaux est plus précise dans le cas où l'intensité de la charge de travail est élevée.

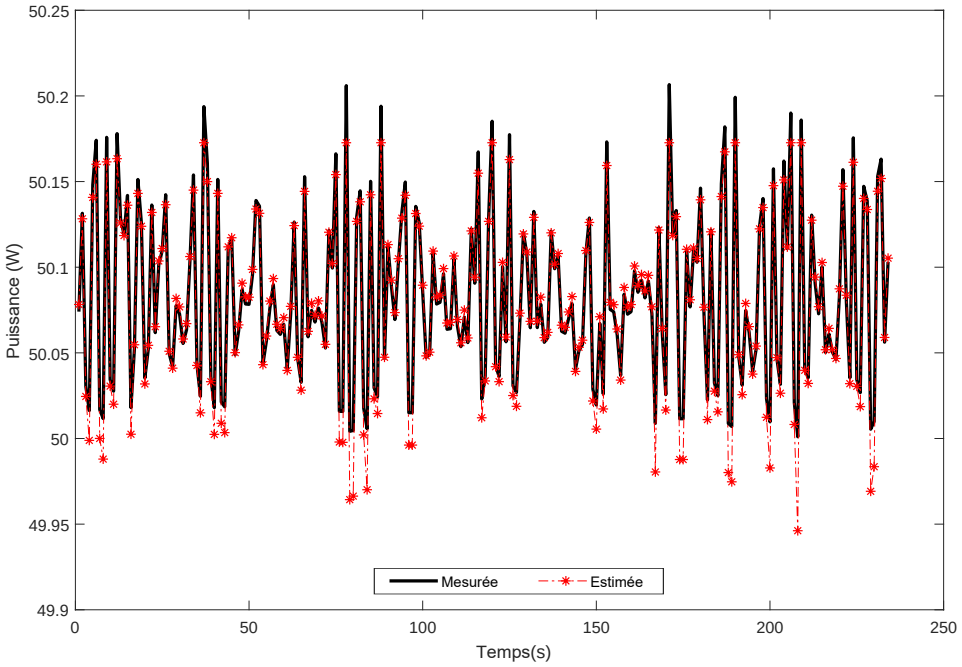
4.3.3 Estimation de la puissance p pour plusieurs serveurs

Fait plus intéressant encore, nous avons utilisé le modèle proposé pour estimer la puissance consommée par plusieurs serveur à la fois. Nous avons réalisé une simulation d'un centre de données comprenant 100 nœuds physiques de deux types (serveur *IBM X3550* à 2 processeurs XeonX5675 et serveur *IBM X3250* à 4 processeurs *XeonX3480*). Chaque nœud est modélisé de manière que chaque cœur de CPU a des performances équivalentes à 1000, 2000 ou 3000 MIPS, $8Go$ de RAM et $500Go$ de stockage. La consommation électrique des hôtes est définie selon un modèle par défaut disponible sur la plate-forme de *CloudSim*. Selon ce modèle, un hôte consomme de $75W$ avec 0% d'utilisation du CPU, jusqu'à $250W$ avec 100% d'utilisation du CPU. Chaque VM nécessite un cœur de CPU avec 250, 500, 750 ou 1000 MIPS, $128Mo$ de RAM et $1Go$ de stockage. Les utilisateurs soumettent des requêtes pour le provisionnement de 150 VMs hétérogènes qui remplissent la pleine capacité du centre de données simulé. Chaque VM exécute une application avec une charge de travail variable modélisée pour générer une utilisation intensive du CPU. Ensuite, nous avons calculé la somme de puissance consommée par tous les serveurs, et suivant l'approche additif, nous avons appliqué le modèle proposé pour chaque serveur individuellement.

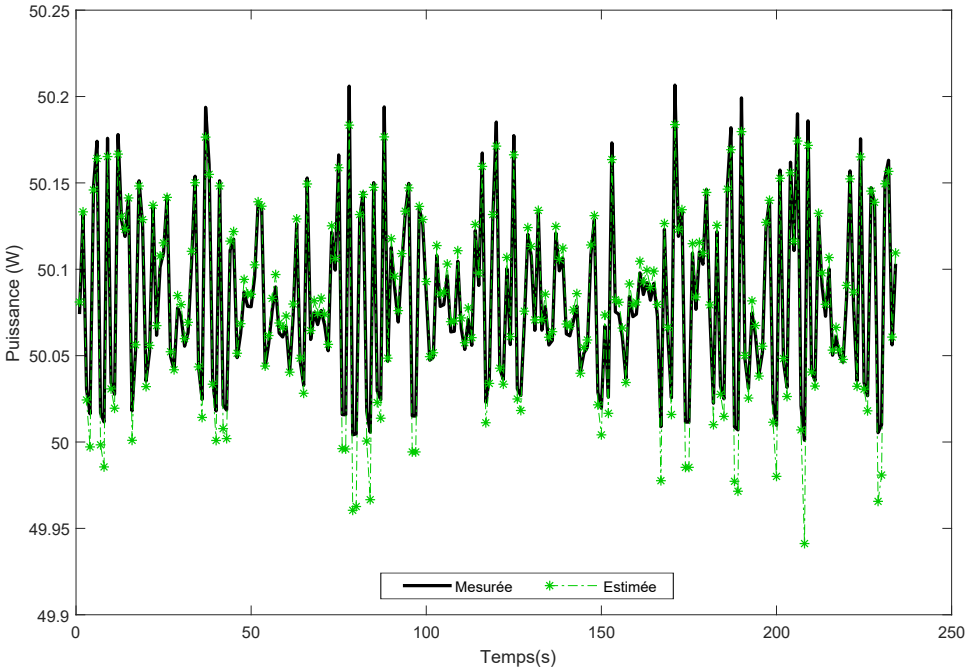
D'après l'analyse de la figure 23, il s'ensuit que le modèle a estimé la consommation de puissance de manière relativement précise et a également suivi le schéma de variation avec précision. Cependant, le taux d'erreur atteint 4%, ce qui peut être dû à la linéarité considéré pour calculer la somme des puissances considérées.

4.4 Conclusion

Les modèles de consommation électrique des serveurs jouent un rôle essentiel dans la gestion de l'énergie dans les centres de données. Dans ce travail, nous avons caractérisé la consommation électrique d'un serveur et sa charge de travail comme des variables

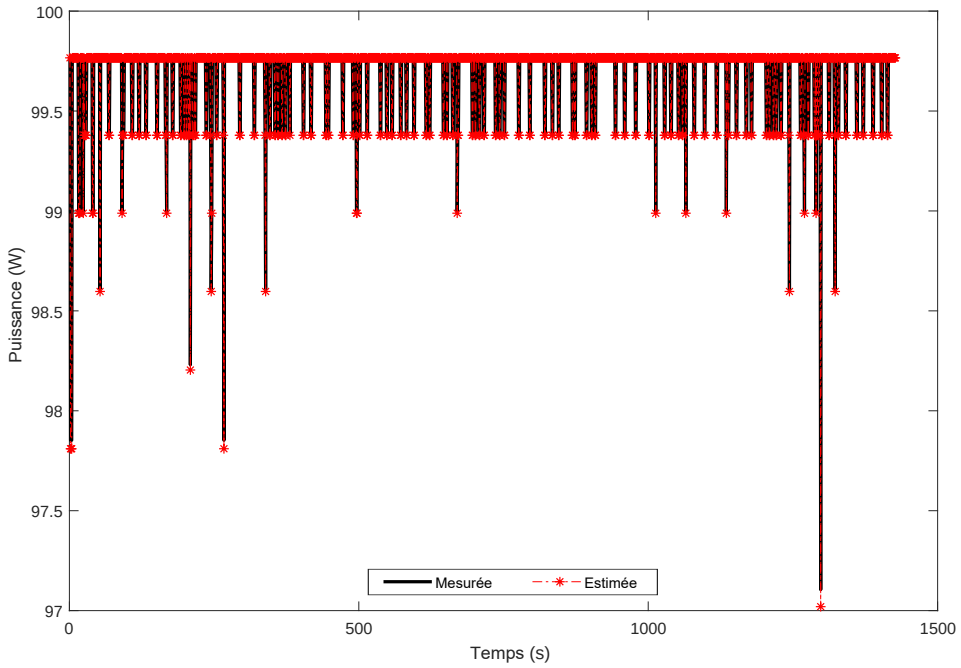


(a) Estimation basée sur la fonction de distribution ECDF.

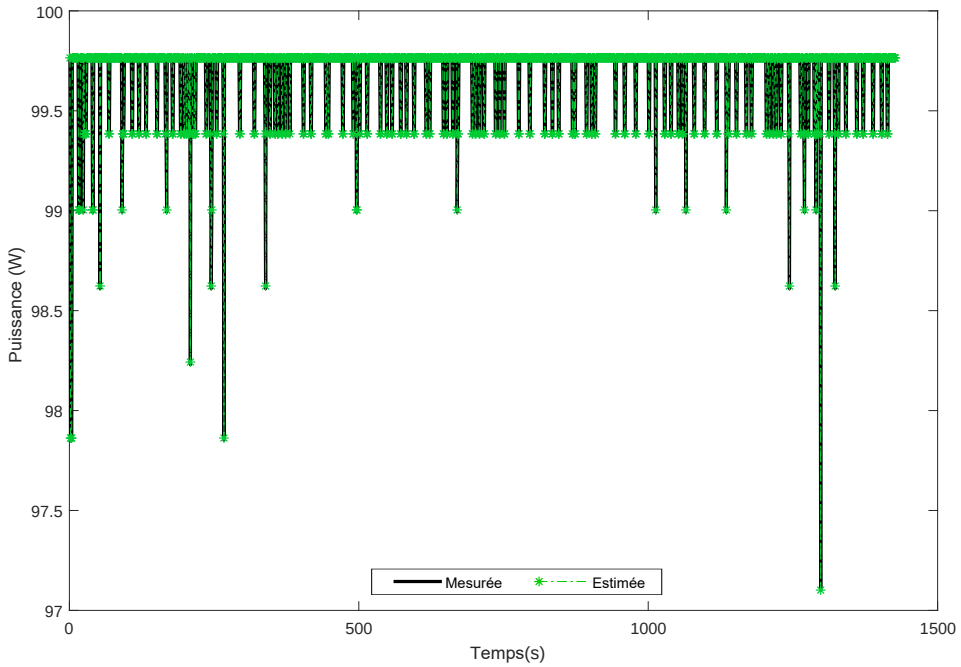


(b) Estimation basée sur la fonction de distribution à noyaux.

FIGURE 21 – La consommation de puissance actuelle et estimée du serveur IBM X3550 à 2 processeurs XeonX5675 lors de l'exécution du benchmark *SpecPower*.



(a) Estimation basée sur la fonction de distribution ECDF.



(b) Estimation basée sur la fonction de distribution à noyaux.

FIGURE 22 – La consommation de puissance actuelle et estimée du serveur IBM X3250 à 4 processeurs XeonX3480 lors de l'exécution du benchmark *SpecPower*.

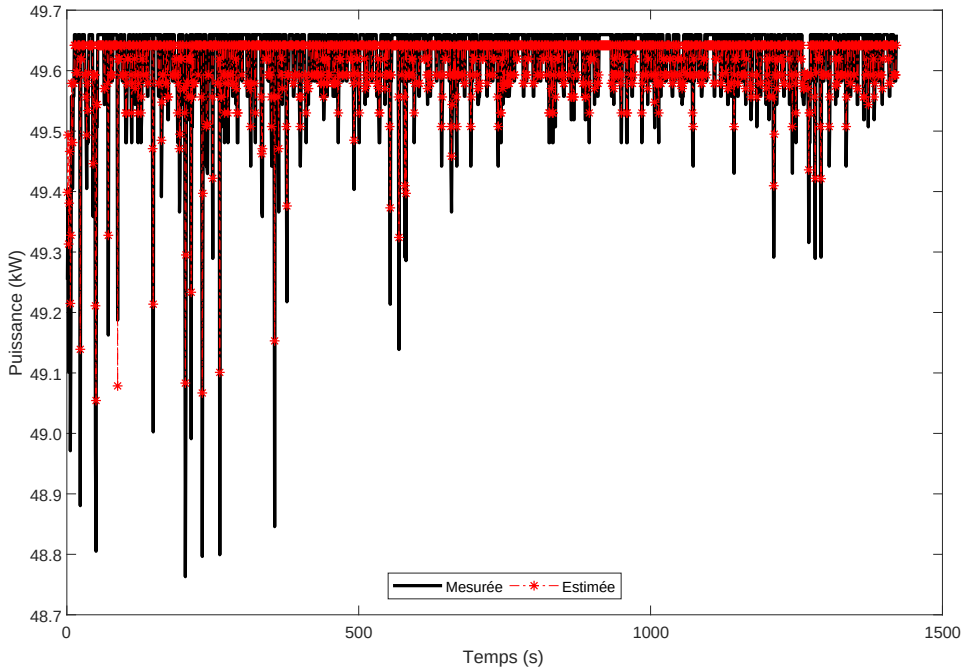


FIGURE 23 – La consommation de puissance totale actuelle et estimée pour plusieurs serveurs.

aléatoires et avons utilisé leurs fonctions de distribution probabilistique pour établir une relation quantitative entre elles : $\boldsymbol{p} = F_p^{-1}(F(\boldsymbol{w}))$. Notre modèle consiste à améliorer l'exactitude d'un modèle proposé précédemment qui a considéré seulement les distributions probabilistiques usuelles ; cependant, le modèle proposé dans ce travail permet d'établir une relation de corrélation entre la charge de travail et la puissance pour n'importe quelle distribution, ainsi que les paramètres associés à chaque fonction sont mis à jour au fur et à mesure avec le jeu de données considéré.

Nous avons testé notre modèle à l'aide de l'outils de simulation *CloudSim* qui permet d'intégrer différentes charges de travail, y compris celles du benchmark *SpecPower*. Contrairement aux familles de benchmarks *SPEC* pour les processeurs ou à des benchmarks similaires, le benchmark *SpecPower* met l'accent sur un serveur avec différentes magnitudes, générant un large spectre d'utilisation qui peut atteindre 100%. Ensuite, nous avons effectué une approximation de la fonction de répartition des variables aléatoires associées à la consommation de puissance et le taux d'utilisation du CPU en utilisant deux méthodes non-paramétriques. Les résultats ont montré que la fonction de densité à noyaux est plus adéquate pour des valeurs élevées, ce qui fait de cette méthode un choix convenable pour l'estimation de la consommation de puissance dans un centre de données.

On a appliqué également le modèle proposé au niveau d'un cluster (un ensemble de serveurs). Notre modèle a estimé la consommation énergétique avec une certaine précision tout en suivant le rythme des variations actuelles de puissance. Ainsi, le modèle proposé,

dans ce travail, peut être employé par un planificateur orienté énergie pour contrôler et organiser le mécanisme d'exécution des VMs (qui effectuent les services rendus par le centre de données). Cette opération sera le sujet du chapitre suivant.

5

Placement des VMs et efficacité énergétique dans le centre de données

---

Ce chapitre introduit une approche de placement statique, dont l'objectif est de minimiser la consommation énergétique totale d'un centre de données, suivant un problème d'optimisation combinatoire. Nous utilisons différentes techniques de résolution pour aborder le problème de placement des VMs défini.



Amazon : https://www.greenamerica.org/amazon-build-cleaner-and-fairer-cloud

5.1 Introduction

La consommation totale de l'énergie dans un centre de données dépend essentiellement de la consommation énergétique de ses PMs actives. Par conséquent, les chercheurs ont fait de nombreuses tentatives pour minimiser la consommation de l'énergie associée à ces unités. Un des procédés efficaces pour améliorer la disponibilité des systèmes et économiser l'énergie consiste à regrouper plusieurs VMs sur un nombre minimal d'appareils physiques puis d'éteindre les appareils inexploités. Comme les utilisateurs peuvent exécuter leurs applications sous forme de VMs avec des ressources partagées au sein des PMs, la manière d'attribuer ces VMs aux ces dernières a un impact significatif sur la consommation énergétique totale des centres de données. Le placement de la VM est une opération critique visant à déterminer l'hôte physique ou le serveur le plus approprié pour accueillir la VM. Le choix de cet hôte est très important pour améliorer l'efficacité énergétique, l'utilisation des ressources et le respect de la QoS dans un environnement de *Cloud Computing*.

L'étape d'assignation des VMs aux serveurs dans un centre de données nécessite la satisfaction de toutes les caractéristiques de ces VMs afin de respecter les SLAs. En même temps, la consommation d'énergie d'un serveur dépend de ses attributs énergétiques, de la charge du processeur et des VMs qui y sont déployées. Nous présentons une illustration motivante qui montre comment la consommation totale d'énergie d'un centre de données dépend des attributs énergétiques des PMs, leurs capacité et les VMs qui sont en cours de fonctionnement.

Un centre de données comprend des PMs hétérogènes, c'est-à-dire des PMs ayant des attributs différents (par exemple, la capacité du CPU et la consommation d'énergie), et héberge des types de VMs hétérogènes, à savoir des VMs ayant des exigences distinctes en matière de ressources. Pour simplifier notre exemple, nous considérons que toutes les VMs sont créées en même temps et exécutées pendant la même durée, et ne sont pas susceptibles de varier leurs demandes de ressources au cours de leur exécution. Les caractéristiques des PMs et des VMs sont présentées dans les tableaux 5.1 et 5.2, respectivement. Puisque la consommation d'énergie, comme expliqué dans le chapitre précédent, d'une PM dépend essentiellement de l'utilisation de son processeur, nous ne considérons que la ressource CPU. Les valeurs des caractéristiques sont choisies aléatoirement, ainsi que l'efficacité énergétique d'une PM est calculée à partir de la consommation d'énergie de chaque coeur de CPU, c'est à dire, $E_{énergétique} = \frac{Capacité_{CPU}}{Énergie_{maximale}}$. Ainsi, nous considèrons les scénarios de placement suivants :

➔ Placement_1 :

1. $(vm5, vm8) \in mp0$
2. $(vm1, vm9) \in mp1$
3. $(vm6, vm7) \in mp2$

TABLE 5.1 – Les caractéristiques des PMs.

PM	*pm0*	*pm1*	*pm2*	*pm3*	*pm4*
Capacité du CPU	10	11	11	11	11
Consommation d'énergie Maximale (Wh)	134	194	139	198	113
Consommation d'énergie Minimale (Wh)	85.8	116.4	83.4	118.8	67.8
Efficacité énergétique	0.07	0.06	0.08	0.06	0.1

TABLE 5.2 – Nombre de CPU requis pour chaque VM.

VM	*vm0*	*vm1*	*vm2*	*vm3*	*vm4*	*vm5*	*vm6*	*vm7*	*vm8*	*vm9*
CPU	8	3	3	1	1	1	1	8	7	1

4. $(vm2, vm3) \in mp3$
5. $(vm0, vm4) \in mp4$

➔ Placement_2 :

1. $(vm3, vm5, vm6) \in mp0$
2. $(vm0, vm4) \in mp1$
3. $(vm7) \in mp2$
4. $(vm2, vm8) \in mp3$
5. $(vm1, vm9) \in mp4$

Nous déterminons la consommation énergétique d'une PM en utilisant le modèle stochastique développé dans le chapitre précédent. Nous trouvons que Placement_1 produit une consommation d'énergie totale de $457.45Wh$ par les PMs et Placement_2 indique une consommation d'énergie totale de $481.72Wh$ pour ces mêmes PMs.

Cet exemple simple démontre que différents modes de placement des VMs entraînent un changement au niveau de la consommation énergétique totale. Cependant, il peut y avoir de nombreux autres scénarios de placement de VMs donnant des taux de consommation d'énergie totale différents. Dans cet exemple, nous ne considérons que les ressources CPU et les demandes de CPU constantes et spécifiques à chaque VM. Cependant, dans les scénarios réels, les exigences en matières de CPU d'une VM peuvent varier dans le temps, ce qui est pris en considération dans cette thèse, et d'autres contraintes de ressources doivent être respectées pour le processus de placement. Par conséquent, trouver le meilleur placement des VMs doit être effectué de manière plus stratégique.

Les méthodes de placement des VMs se répartissent en deux catégories : statiques, où la répartition des VMs est maintenue pendant toute la durée de vie de la VM et ne subira aucun nouveau traitement, et dynamiques où le placement initial est susceptible de varier en raison de certains aléas de la charge du dispositif. D'autre part, le placement d'une VM sur un hôte physique est désigné comme étant un problème en temps polynomial non-déterministe, plus précisément un problème NP-difficile (car, par exemple, si n est le nombre total de VMs et m le nombre total de serveurs, alors le nombre de correspondances

possibles est de m^n), de sorte que le problème du sac à dos peut y être réduit. En général, l'approche de sac à dos est un problème d'optimisation combinatoire qui consiste en une contrainte sur les éléments de la solution. Il constitue, toutefois, un défi pour les chercheurs en raison de sa complexité, puisqu'il appartient à la classe des problèmes NP-difficiles. Le contexte est celui d'une personne qui a un sac et une liste d'objets qu'elle est susceptible de transporter dans le sac. Chaque objet a une certaine valeur et occupe une certaine place dans le sac. Sachant que la capacité du sac est limitée, le problème pour la personne est de maximiser la valeur d'un ensemble d'objets mis dans le sac tout en respectant la limitation de sa capacité. Dans cette thèse, une mise en œuvre détaillée du processus de placement des VMs inspiré du problème de sac à dos est envisagée, dans la mesure où les objets représentent les VMs et le sac à dos correspond au serveur.

Ce chapitre met en relief l'utilisation du concept du problème de sac à dos pour la mise en place d'un mécanisme de placement des VMs dans un centre de données. Comme toutes les techniques d'optimisation, l'objectif principal est de parvenir à une solution optimale lors de l'affectation des VMs aux PMs, en suivant une politique de la minimisation de l'ensemble des besoins en ressources telles que le CPU, et par la suite, la minimisation de la consommation d'énergie. La présente approche suit une nouvelle politique qui consiste à maximiser l'utilisation des ressources dans un serveur (CPU par exemple). La politique adoptée reflète la philosophie générale du placement des VMs, qui consiste à concentrer le traitement sur un nombre minimum de serveurs, les autres pouvant être arrêtés.

Le reste de ce chapitre est organisé comme suivant : dans la section 5.2, nous introduisons la démarche adoptée pour formuler le problème de placement des VMs comme étant un problème de sac à dos. Nous présentons, également, les méthodes de résolution utilisées pour résoudre ce problème, en considérant les contrainte énergétique et de performance qu'on a défini. La section 5.3 discute les résultats expérimentaux, et nous menons une étude comparative entre les quatre méthodes de résolution qu'on a employés en terme de performance et d'optimisation des ressources CPU. Dans la section 5.4, nous adoptons un modèle de résolution basé sur l'approche OCF pour résoudre le problème de placement des VMs en tant que problème de sac à dos. Ensuite, nous développons un environnement de simulation qui a montré que notre modèle fourni une amélioration remarquable au niveau de l'efficacité énergétique par rapport à un modèle de minimisation d'utilisation de ressources d'un serveur basé sur l'algorithme FFD qui est largement utilisé dans la littérature. La section 5.5 présente la conclusion du chapitre.

5.2 Modélisation du placement des VMs comme un problème de sac à dos

En général, le processus du placement des VMs peut suivre différentes orientations stratégiques en fonction d'une ou plusieurs contraintes à respecter. Par exemple, si la contrainte concerne l'utilisation de la mémoire, le modèle est orienté pour réduire le taux d'utilisation de cette dernière. Les travaux cités dans la section 3.3 ont traité le problème d'affectation des VMs que ce soit en minimisant la consommation d'énergie d'un serveur donné, en minimisant un ensemble d'exigences (CPU, mémoire et bande passante), ou en maximisant le nombre de VMs dans un serveur tout en respectant la contrainte associée à la disponibilité du serveur et de sa capacité donnée. Dans ce travail, le problème est abordé de manière différente ; nous essayons de maximiser l'utilisation d'un serveur en maximisant l'utilisation de son CPU tout en respectant la contrainte de limitation de ces ressources. En effet, la principale priorité du placement des VMs dans un centre de données réside dans la minimisation du nombre de PMs en fonctionnement. C'est donc le nombre de serveurs utilisés qui détermine la consommation d'énergie d'un centre de données. Ainsi, maximiser l'utilisation d'un certain nombre d'hôtes permet de libérer les autres, et par la suite de les éteindre. Le choix du CPU revient au fait que ce composant est considéré comme étant le composant le plus important pour l'efficacité énergétique et les autres composants ayant une portée de consommation faible ou présentent une forte corrélation avec l'activité du CPU ne sont pas pris en compte en raison de simplification du modèle.

Pour formuler ce qui est exposé ci-dessus, considérons un ensemble de n VMs à attribuer, chacune d'entre elles requiert une certaine capacité de CPU p_i pour fonctionner pendant une durée spécifique, au cours de cette même durée, le serveur assure c_i de ressources CPU avec une capacité globale C. De plus, $x_i \in \{0, 1\}$ est une variable permettant d'indiquer si la VM est sélectionnée pour le placement ($x_i = 1$) ou non ($x_i = 0$). La formule est donc donnée par :

$$max \sum_{i=1}^{n} p_i x_i \tag{5.1}$$

s.c

$$\sum_{i=1}^{n} c_i x_i \leq C \tag{5.2}$$

La fonction objectif 5.1 consiste à sélectionner un ensemble de VMs parmi celles existantes de telle sorte que l'utilisation du CPU soit maximale dans un serveur, tandis que la contrainte 5.2 garantit que la quantité de CPU utilisée par l'ensemble sélectionné de ces VMs ne dépasse pas la valeur maximale C. En outre, nous supposons que cette capacité

maximale du CPU, C, est choisie en assumant qu'une VM peut nécessiter des ressources supplémentaires lorsqu'elle est exécutée sur le serveur.

Notre formulation du problème ci-dessus est exprimée en tant que problème de sac à dos $(0-1)$, qui est un problème d'optimisation combinatoire à contrainte ; vue que nous ne pouvons choisir une VM à placer qu'une seule fois et nous n'avons donc que deux options pour chaque VM, soit la choisir (1), soit la laisser (0) sans possibilité de la répartir. En outre, comme indiqué dans la sous-section 2.2.1, le volume et le poids de chaque VM sont liés, et par la suite le problème de sac à dos uni-dimensionnel est le plus convenable pour formuler le problème de placement défini. L'objectif est de minimiser l'utilisation du CPU sur des intervalles de temps variés. Cet objectif est atteint par le placement statique des VMs. Plusieurs algorithmes ont été développés pour résoudre le problème du placement des VMs en tant que le problème de sac à dos. Dans ce qui suit, nous exposons comment les algorithmes décrits dans la sous-section 2.3.1 sont appliquées pour résoudre le problème introduit.

5.2.1 Résolution par séparation et évaluation progressive

En appliquant la méthode *Branch&Bound* au problème de sac à dos pour le placement des VMs, un sac à dos vide correspond à l'état initial, tandis que l'état objectif correspond au sac à dos optimisé (solution optimale). Les états intermédiaires sont les différentes combinaisons d'éléments qui peuvent être la solution optimisée mais qui n'ont pas encore atteint la valeur optimale. Toute solution partielle dont le coût est supérieur à celui de la meilleure solution trouvée sera exclue de la recherche. Ce processus est poursuivi tant qu'une solution optimale n'a pas été trouvée. Si le coût de cette solution complète est inférieur à celui de la meilleure solution que nous avons déjà trouvée, nous remplaçons alors cette solution par la meilleure solution actuelle (dont le coût est inférieur). Dans chaque nœud de l'arbre, le poids total p et la capacité totale ς des éléments déjà sélectionnés sont enregistrés, ainsi que la valeur de la limite supérieure ub pouvant être calculée selon l'équation suivante :

$$ub = p + (p_{i+1}/c_{i+1}) * (C - \varsigma) \tag{5.3}$$

Où $C - \varsigma$ est la capacité disponible du sac à dos et la ration $\frac{p_{i+1}}{c_{i+1}}$ est le meilleur résultat par unité parmi les éléments restants. Le pseudo-code est présenté dans Algorithme 1.

Algorithme 1 *Branch&Bound*

Entrée:

— Type du noeud : courant, temporaire

— Initialiser la racine

— Q.enqueue(racine)

— $v_max \leftarrow valeur$

Sortie: Meilleure solution et valeur optimale

tant que $Q \neq \emptyset$ **faire**
  $courant \leftarrow Q.getMax()$
  si $courant \geq V_max$ **alors**
    Définir le nœud enfant gauche du nœud actuel pour inclure l'élément suivant
  fin si
  si $(V_{enfant.gauche} \geq V_max)$ **alors**
    $V_max \leftarrow V_{enfant.gauche}$
  fin si
  si $enfant.gauche_ub \geq meilleure_valeur$ **alors**
    Q.enqueue(enfant.gauche)
  fin si
  si $enfant.droit_ub \geq meilleure_valeur$ **alors**
    Q.enqueue(enfant.droit)
  fin si
fin tant que

5.2.2 Résolution par la programmation dynamique

La programmation dynamique est une approche générale de résolution de différents problèmes. Cette technique peut être appliquée lorsqu'une solution optimale consiste en une combinaison de solutions optimales des sous-problèmes, ce qui est le cas pour le problème de sac à dos. La valeur optimale d'une instance de ce problème est obtenue en considérant le maximum de deux situations sur le dernier élément, ici noté avec l'indice i :

— Soit l'objet fait partie de la solution (ajouté au sac à dos), dans ce cas la capacité disponible restante pour les autres objets est égale à $C - c_i$; la valeur optimale est alors égale à celle obtenue pour les autres objets ayant cette capacité, plus le profit p_i.

— Soit l'objet ne participe pas à la solution, dans ce cas la valeur optimale est égale à celle obtenue par les autres objets de même capacité.

Ce processus est appliqué de manière récursive jusqu'à ce qu'il n'y ait plus d'objet à traiter. Sachant que les calculs sont effectués du bas vers le haut. Mathématiquement, étant donné une paire d'entiers k ($1 \leq k \leq n$) et ς ($1 \leq \varsigma \leq C$), nous considérons la sous-instance du problème de sac à dos qui se compose des k premiers éléments et d'une capacité ς. Désignons sa valeur de solution optimale par :

$$DP_{\varsigma}^{k} = \left\{ \sum_{j=1}^{k} p_j x_j : \sum_{j=1}^{k} c_j x_j \leq \varsigma; x_j \in \{0,1\}, \forall i \in \{1,...,n\} \right\} \tag{5.4}$$

Initialement, nous avons :

$$DP_{\varsigma}^{k} = \begin{cases} 0 & pour \ \ 0 \leq \varsigma \leq c_1 \\ p_1 & pour \ \ c_1 \leq \varsigma \leq C \end{cases} \tag{5.5}$$

La programmation dynamique consiste à considérer n étapes (pour k augmentant de 1 à n) et à calculer à chaque étape $k \geq 1$ les valeurs DP_{ς}^{k} (pour ς augmentant de 0 à C) en utilisant la récursion classique de *Bellman* [200] :

$$DP_{\varsigma}^{k} = \begin{cases} DP_{\varsigma}^{k-1} & pour \ \ 0 \leq \varsigma \leq s_k \\ \max(DP_{\varsigma}^{k-1}, DP_{\varsigma-c_k}^{k-1} + p_k) & pour \ c_k \leq \varsigma \leq C \end{cases} \tag{5.6}$$

Algorithme 2 *Programmation dynamique*

Entrée:

— Tab_p : tableau trié des valeurs

— Tab_c : tableau trié des poids

— n : nombre des éléments

— C : capacité maximale

Sortie: Meilleure solution et valeur optimale

pour $\varsigma = 0$ *à* C **faire**
    $DP_{\varsigma}^{0} \leftarrow 0$
fin pour
pour $i = 0$ *à* n **faire**
    $DP_{0}^{i} \leftarrow 0$
fin pour
pour $i = 1$ *à* n **faire**
    pour $\varsigma = 0$ *à* C **faire**
        si $c_i \leq \varsigma$ **alors**
            si $p_i + DP_{\varsigma - c_i}^{i-1} \geq DP_{\varsigma}^{i-1}$ **alors**
                $DP_{\varsigma}^{i} \leftarrow p_i + DP_{\varsigma - c_i}^{i-1}$
            sinon
                $DP_{\varsigma}^{i} \leftarrow DP_{\varsigma}^{i-1}$
            fin si
        fin si
    fin pour
fin pour

5.2.3 Résolution par algorithme génétique

Nous commençons par développer un AG permettant de résoudre le problème de placement des VMs. Étant un algorithme basé sur la population, il est principalement constitué des individus composant chaque génération. Afin de fixer les opérations nécessaires à cet algorithme, nous devrons trouver la meilleure façon de représenter ces individus. Celles-ci doivent être conçues de manière à ce que les opérations de croisement et de mutation, soient aisément applicables.

L'objectif du placement des VMs dans les centres de données est de à trouver un emplacement convenable (en fonction de la contrainte définie) pour chaque VM dans les serveurs. Ainsi, il est possible de représenter l'individu sous la forme d'une liste de serveurs



FIGURE 24 – Exemple d'un chromosome (sac à dos).

TABLE 5.3 – Concepts de base de l'AG pour résoudre le problème de sac à dos.

Concept	*Description*
Population	Au début, un certain nombre de sacs à dos (serveurs) sont constitués de manière aléatoire (vides ou pleins). Ensuite, l'algorithme va progressivement modifier cette population (VMs), afin qu'elle contienne des éléments faisant intervenir une meilleure solution.
Chromosome	C'est l'un des sacs à dos qui composent la population. Autrement dit, la liste des objets qui se trouvent dans le sac à dos.
fonction de fitness	La somme des valeurs des objets mis dans le sac à dos. L'objectif est de maximiser cette fonction.

hébergeant les VMs (sac à dos). Par exemple, le premier élément de la liste correspond à la première VM, et le dernier élément de la liste correspond à l'emplacement de la dernière VM, comme décrit dans la figure 24.

Chaque composant constitutif du chromosome correspond à un gène. Ce dernier désigne par cet algorithme l'emplacement de la VM, autrement dit, l'identifiant du serveur qui l'héberge. Par conséquent, les opérations seront appliquées à tous les individus de la population.

Les paramètres de l'algorithme doivent être initialisés préalablement. Par exemple, le nombre des individus dans la population initiale et le nombre d'itérations à exécuter qui correspond au nombre de générations à créer. Ces paramètres sont choisis de manière arbitraire et seront ajustés par des tests initiaux.

Dans ce travail, nous exploitons l'AG pour la résolution du problème de sac à dos comme indiqué dans le tableau 5.3. Le pseudo-code est présenté dans Algorithme 3.

Algorithme 3 *Algorithme génétique*

Entrée:

— pX : taux global de croisement

— pU : taux uniforme de croisement

— pM : taux de mutation

— $max_générations \leftarrow 400$

— Générer la population initiale

Sortie: Meilleure solution et valeur optimale

1: Initialiser une population P
2: Évaluer P
3: **tant que** *Nombre de génération* $\leq$ *max_générations* **faire**
4: Sélectionner l'ensemble des parents $P^{'} \in P$
5: Appliquer les opérateurs de croisement (pX, pU) à $P^{'}$
6: Appliquer l'opérateur de mutation à $P^{'}$
7: Remplacer P par $P^{'}$
8: **fin tant que**

5.2.4 Résolution par optimisation de colonie de fourmis

OCF a été adopté et prouvé capable de résoudre des problèmes sophistiqués d'optimisation combinatoire statique, qui sont classés comme des problèmes NP-difficiles. OCF présente des caractéristiques très avantageuses qui le différencient des autres algorithmes. OCF peut construire plusieurs ensembles de solutions différentes qui permettent à un algorithme d'augmenter sa capacité à découvrir plus de solutions que les algorithmes heuristiques traditionnels [174]. En outre, les approches classiques peuvent trouver une solution optimale pour des problèmes d'optimisation compliqués dans des conditions d'incertitude et ce, uniquement pour des problèmes de petite taille, qui nécessitent d'énormes efforts de calcul, tandis que les approches métaheuristiques sont capables de trouver une solution réalisable qui pourrait être optimale pour des problèmes de taille raisonnable.

Une autre caractéristique qui distingue OCF des autres algorithmes métaheuristiques est que chaque fourmi possède une mémoire. Une fourmi utilise cette mémoire pour stocker l'information de son chemin (solution) afin de construire des solutions réalisables, de calculer des informations heuristiques, d'évaluer des solutions réalisables et de retracer son chemin à l'envers. De plus, l'OCF met à jour la phéromone en fonction de la qualité de la solution qui peut être augmentée lorsque la solution devient meilleure lors des itérations

suivantes. Par conséquent, nous adoptons cette technique pour résoudre le problème du placement statique des VMs formulé dans la section 5.2. Quelques informations générales sur l'OCF sont fournies dans la sous-section 2.3.1.

Selon la version originale de l'algorithme, la mise en œuvre de l'OCF se fait selon quatre procédures : Initialisation de la phéromone, définition des informations heuristiques, construction de la solution et mise à jour de la phéromone. L'algorithme peut se résumer comme suit : Pour construire une solution, les fourmis choisissent de manière aléatoire un élément initial, puis ajoutent itérativement des éléments qui sont choisis à partir d'un ensemble qui contient tous les éléments qui peuvent être sélectionnés (candidats) sans violer la contrainte de ressources . À chaque étape, l'élément à ajouter à la solution actuelle S_k est sélectionné dans l'ensemble des candidats avec une probabilité P_i. Cette probabilité est définie proportionnellement à un facteur phéromonal τ_i et à un facteur heuristique (également appelé facteur d'attractivité) η_i, tous les deux étant pondérés respectivement par deux paramètres α et β qui déterminent leur importance relative. Le pseudo-code est donné dans Algorithme 4, sachant que :

$$P_i = \begin{cases} \frac{\tau_i^\alpha \eta_i^\beta}{\sum_{i \in S_k} \tau_i^\alpha \eta_i^\beta} & pour\ i \in Candidats \\ 0 & pour\ i \notin Candidats \end{cases} \tag{5.7}$$

Le facteur phéromonal est mis à jour par :

$$\tau_i = \tau_i + \Delta\tau \tag{5.8}$$

Où

$$\Delta\tau = \frac{1}{1 + \frac{p_{best} - p_i}{p_{best}}} \tag{5.9}$$

Le facteur heuristique est exprimé par :

$$\eta_i = \frac{p_i}{w_i^2} \tag{5.10}$$

Appliqué au problème de sac à dos, les fourmis construisent des solutions selon une heuristique gloutonne : à partir d'un sac à dos vide, elles sélectionnent un nouvel objet à chaque itération ; cet objet est choisi selon un critère heuristique (valeur de l'objet candidat) et des traces d'une phéromone (caractérisant l'expérience passée de la colonie). Trois implémentations de l'algorithme OCF pour le problème de sac à dos ont été proposées. Ces trois algorithmes se différencient par leur façon de déposer et d'exploiter les traces des phéromones. [161] ont proposé de déposer des traces de phéromone sur les objets sélectionnés, tandis que [210] a proposé de déposer la phéromone sur le chemin formé plus

tard par les objets sélectionnés. [11] ont proposé de déposer des phéromones sur les paires d'objets sélectionnés dans la même solution. Pour la présente étude, nous avons adopté la première méthode.

Algorithme 4 *Algorithme de colonie de fourmis*

Entrée:

- Tab_p : tableau des valeurs
- Tab_c : tableau des poids
- K : nombre de fourmis
- C : capacité
- ρ : facteur d'évaporisation
- τ : facteur phéromonal initial

Sortie: Meilleure solution et valeur optimale

1: **pour** *pour chacune des* K *fourmis* **faire**
2: **tant que** $C \geq 0$ **faire**
3: Sélectionnez l'élément suivant i parmi les candidats avec une probabilité de P_i suivant la formule 5.9
4: Ajouter l'élément sélectionné à une solution partielle $S \leftarrow S + i$
5: Mettre à jour la capacité de portance actuelle du sac à dos $C \leftarrow C - c_i$
6: Mettre à jour le profit $p \leftarrow p + p_i$
7: Éliminer les articles dont le poids dépasse la nouvelle capacité C
8: **fin tant que**
9: **fin pour**

5.2.5 Analyse théorique de la complexité des algorithmes

En général, le choix d'une méthode pour résoudre un problème d'optimisation dépend non seulement de sa fiabilité, mais aussi de sa complexité. En effet, certains algorithmes sont plus efficaces que d'autres, mais leur complexité rend le processus de résolution plus lent. Il convient donc de trouver le meilleur compromis entre la qualité de la solution fournie et la complexité en termes de temps observée. Le tableau 5.4 montre la complexité des quatre algorithmes étudiés. Il s'ensuit que la méthode *Branch&Bound* a une complexité exponentielle qui est donnée par $O(2^n)$. L'algorithme de programmation dynamique est de complexité pseudo-polynomiale, c'est-à-dire que la complexité temporelle $O(n * C)$ dépend à la fois du nombre d'éléments n, et de la capacité du sac à dos C qui est de nature

TABLE 5.4 – Estimation de la complexité des algorithmes utilisés.

Algorithme	*Complexité*
Branch&Bound	Au pire, tous les nœuds de l'arbre sont générés, soit 2^n possibilités de vérification. Si l'on considère, en plus, les nœuds intermédiaires, il y aura un total de $2^{n+1}-1$ nœuds à traiter. Par conséquent, la complexité la plus élevée possible est donnée par $O(2^n)$.
PD	Le calcul de la valeur DP_S^i est l'opération de base qui peut être exécutée $n * C$ fois, car elle nécessite que DP_S^{i-1} et $DP_{S-c_i}^i$ soient déjà calculés (équation 2), ce qui sera assuré par un balayage sur toutes les valeurs. Par conséquent, dans le pire des cas, la complexité requise est de $O(n * C)$.
AG	Selon les procédures de croisement et de sélection, la complexité d'un AG est définie par la taille de la population N, car le nombre de chromosomes dans chaque génération et le nombre de générations sont fixes. Ainsi, la complexité globale du programme est de $O(N)$.
OCF	En utilisant une heuristique statique, seule la phéromone suit les changements de valeur à chaque étape de construction d'une solution partielle. Ainsi, la valeur de la probabilité de sélection de chaque élément est calculée une seule fois pour un cycle, et elle est utilisée par toutes les fourmis pour construire une solution. Sachant que le modèle de phéromone utilisé associe une piste de phéromone à chaque élément, la complexité est de $O(n)$ (car le nombre de fourmis est constant).

exponentielle telle qu'elle est représentée dans le système de calcul binaire . De plus, nous pouvons établir que les algorithmes AG et OCF fournissent tous les deux une manière possible de traiter le problème de sac à dos selon une complexité linéaire. En conséquence, nous supposons que théoriquement, sauf dans des cas limités, il est difficile de résoudre le problème de placement des VMs comme un problème de sac à dos en temps raisonnable par un algorithme exact. Cependant, les approches métaheuristiques peuvent être de bons candidats pour résoudre un tel problème, ce qui sera vérifié de manière pratique dans ce qui suit.

TABLE 5.5 – Les paramètres principaux de l'algorithme AG et OCF.

	Paramètre	Valeur
AG	Taux global de croisement	0.15
	Taux uniforme de croisement	0.55
	Taux de mutation	1.2/(nombre de VMs)
OCF	Nombre de fourmis	500
	Facteur phéromonal initial	0.2
	Facteur d'évaporisation	0.4

5.3 Résultats expérimentaux et analyse de performance

Il est important, lorsque des algorithmes sont proposés pour résoudre un problème, de comparer ces algorithmes en fonction du problème dont la solution a été prouvée. Une façon de comparer différents algorithmes est de considérer deux facteurs cruciaux, la qualité de la solution, en répondant à des questions telles que, le problème exige-t-il une solution exacte ou approximative ? Comment les approches proposées se comparent-elles les unes aux autres, c'est-à-dire en termes de précision pour le problème de placement des VMs ? et l'efficacité, qui consiste à examiner, entre autres, le temps et les ressources nécessaires pour chaque approche.

Dans cette section, un jeu de données réelles est employé pour tester les quatre méthodes proposées pour résoudre le problème de sac à dos dans le cadre de placement des VMs dans un centre de donnée. Pour de plus amples renseignements concernant cet ensemble de données, le lecteur est invité à se reporter au chapitre suivant. À partir de cet ensemble de données, nous sélectionnons deux caractéristiques qui correspondent à la formulation du problème dans la section précédente. Il s'agit de la quantité de CPU nécessaire à une VM pour fonctionner pendant une période donnée, ainsi que la quantité de CPU disponible pendant la même période. En outre, le tableau 5.5 présente le paramétrage de l'algorithme AG et OCF, sachant que les valeurs données sont choisies en fonction des résultats obtenus par plusieurs exécutions indépendantes. La version des algorithmes a été implémentée en *Python 3.5* (qui fournit diverses bibliothèques *open source* pour l'exploration des données et des calculs scientifiques robustes) et évaluée avec la même taille de capacité sur un processeur *i5-4210U 1.70GHz*, et 4GB de mémoire. Cette section présente et examine une comparaison entre les résultats de l'application des méthodes adoptées à l'ensemble de données sélectionné. Cette analyse se focalise dans un premier temps sur les combinaisons d'éléments (VMs) produites par chaque méthode, ainsi que sur l'examen du taux de convergence. Ensuite, elle analyse le temps d'exécution et la quantité de CPU utilisés pour les quatre approches.

TABLE 5.6 – Nombre de VMs et taux de CPU correspondant utilisé, donné par l'exécution de chaque algorithme pour un nombre différent de VMs à placer dans un serveur avec différentes capacités.

	20 VMs C=400MB		50 VMs C=800MB		100 VMs C=1600MB	
	Nombre de VMs	CPU utilisée par les VMs	Nombre de VMs	CPU utilisée par les VMs	Nombre de VMs	CPU utilisée par les VMs
Branch&Bound	5	392.36	6	796.68	13	1585.41
PD	3	385.38	6	765.13	10	1484.6
AG	4	294.2	5	734.2	13	1251.48
OCF	4	353.84	8	758.8	15	1393.88

5.3.1 Nombre de machines virtuelles

Cette mesure représente le nombre total de VMs à intégrer dans un serveur de manière à maximiser l'utilisation de la ressource CPU du serveur tout en respectant ses capacités limites. Les résultats obtenus sont résumés dans le tableau 5.6. Dans un premier temps, nous essayons de placer 20 VMs dans un serveur qui offre une capacité maximale de 400 Mo. Les résultats obtenus montrent que *Branch&Bound* donne la solution la plus adaptée au problème étudié ; il s'agit de cinq VMs qui utilisent presque 100% de la capacité autorisée. Cependant, les valeurs obtenues par l'AG sont les plus faibles. En passant à la deuxième série (50VMs, C=800 Mo), nous remarquons que l'utilisation de l'algorithme OCF permet de placer le plus grand nombre de VMs (8 VMs), mais les ressources CPU, utilisées par les VMs sélectionnées, données par les quatre algorithmes sont de toute façon presque égales. En augmentant le nombre de VMs à placer (100 VMs), nous constatons que la différence entre les résultats obtenus, en termes de taux d'utilisation du CPU, s'augmente relativement. En effet, les deux méthodes exactes donnent une solution qui garantit l'utilisation maximale des ressources CPU du serveur. Toutefois, les valeurs données par l'OCF sont également pertinentes. Pour mieux illustrer ce point, la figure 25 présente graphiquement les résultats discutés.

5.3.2 Taux de convergence

Le taux de convergence est une mesure pertinente sur laquelle nous pouvons baser notre comparaison. Il s'agit principalement du nombre d'itérations nécessaires pour obtenir une solution optimale. La convergence est assurée lorsque le nombre d'itérations est prédéterminé, comme dans le cas de la programmation dynamique, la solution optimale est donnée par 100 itérations pour garantir qu'il ne reste aucun élément à traiter. L'algorithme *Branch&Bound* converge nécessairement, puisqu'il élimine toujours les solutions partielles à chaque étape, qui sont en nombre fini. Dans le cas de l'exécution présentée dans la figure 26, la solution optimale est atteinte après environ 130 itérations. De son côté, l'AG converge si la valeur de fitness reste presque constante lors du passage d'une génération à une autre. Nous pouvons remarquer que l'AG nécessite un nombre important

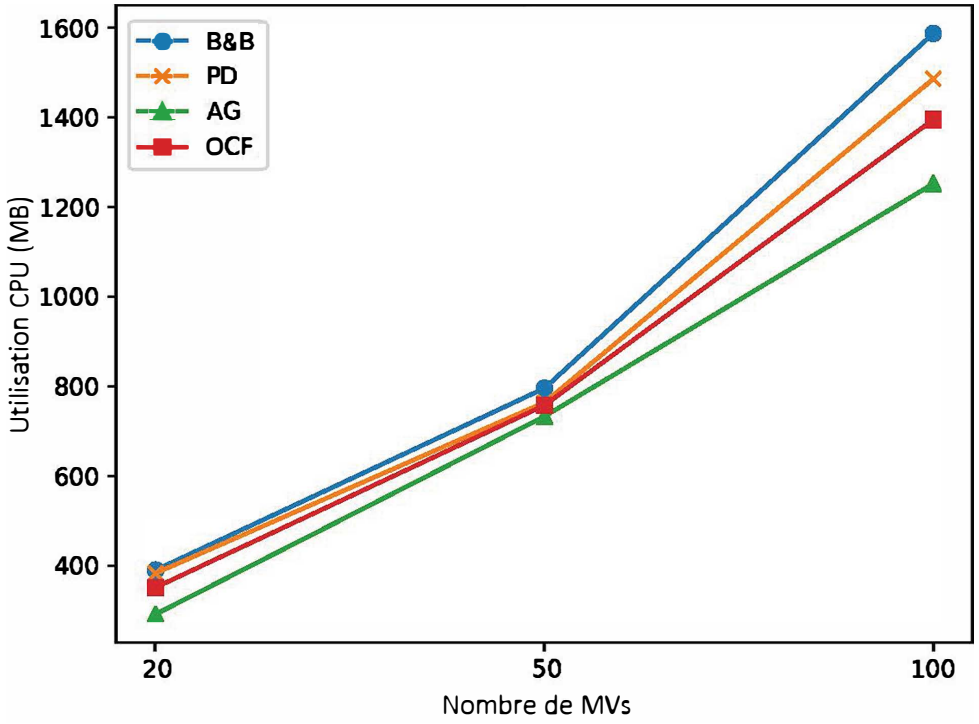


FIGURE 25 – Évolution du taux d'utilisation du CPU indiqué par les quatre algorithmes dans les trois situations étudiées (20, 50, 100 VMs).

d'étapes itératives pour converger (près de 400 itérations Figure 26). La convergence de l'approche OCF est finalement assurée lorsque toutes les fourmis fournissent pratiquement la même solution optimale. La figure 26 montre que l'OCF converge après 250 itérations. Notons que l'exécution a été effectuée avec un nombre de fourmis égal à 500, ce qui signifie que l'on peut réduire le nombre de fourmis supplémentaires sans affecter la solution optimale.

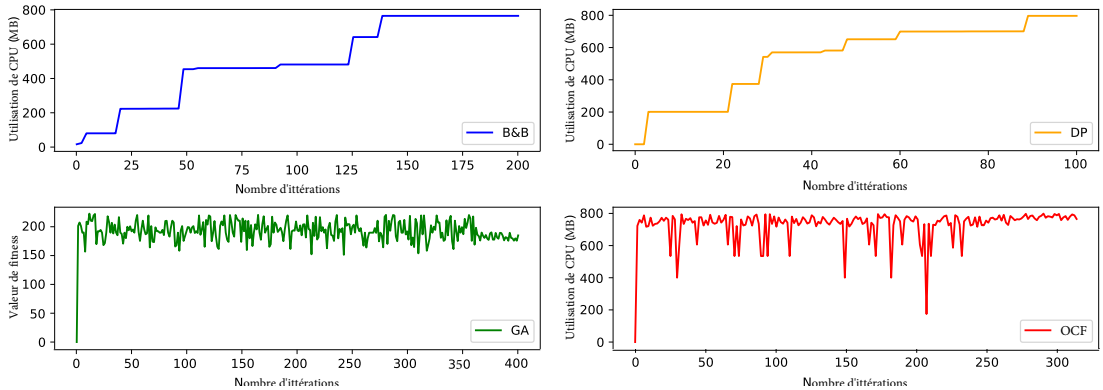


FIGURE 26 – Le taux de convergence pour les quatre algorithmes afin de fournir une solution pour un nombre de VMs égal à 100.

5.3.3 Utilisation de CPU

L'analyse et la comparaison entre les quatre méthodes au niveau de l'utilisation du CPU sont également importantes. Dans cette étude, nous utilisons la bibliothèque *Psutil*[1] *(Python System and Process Utilities)* de *Python 3*, les résultats obtenus sont résumés dans la figure 27 où l'on peut observer une différence de taux d'utilisation du CPU sur 50 exécutions indépendantes de chaque algorithme pour les trois cas mentionnés ci-dessus (20, 50, et 100 VMs respectivement). Il s'ensuit que pour les 20 VMs , il n'y a pas de différence significative entre les méthodes testées, sauf pour l'AG. Néanmoins, au fur et à mesure que le nombre de VMs s'augmente, une variation importante est identifiée. Il est évident que le taux de CPU utilisé par les deux approches exactes *Branch&Bound* et la programmation dynamique dépasse celui des deux métaheuristiques AG et OCF, en particulier la différence entre la méthode OCF et les autres méthodes est clairement visible.

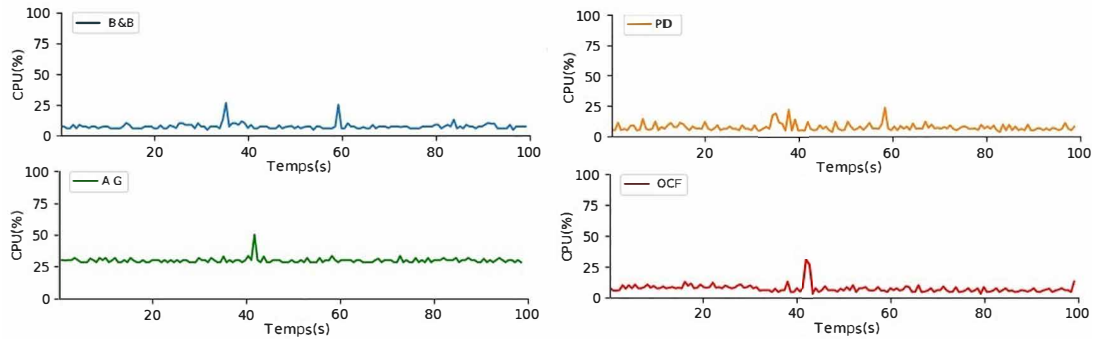
5.3.4 Temps d'exécution

Pour explorer davantage la notion d'efficacité, il est nécessaire d'invoquer la mesure du temps d'exécution. Nous utilisons le *Python Profiler* pour évaluer la performance du temps d'exécution (en fonction du nombre d'appels par fonction) des algorithmes étudiés. De l'analyse de la figure 28a, il en résulte que pour un nombre réduit de VMs à adapter, le temps maximal est donné par la programmation dynamique, sans différence significative entre les autres méthodes. Suivant le même processus et à partir de 50 VMs, la différence entre les méthodes exactes et les métaheuristiques commence à se faire remarquer. Nous constatons, dans la figure 28b, que l'algorithme *Branch&Bound* et la programmation dynamique prennent beaucoup plus de temps pour être exécutés. Plus intéressant encore, la figure 28c montre les temps d'exécution des quatre approches lorsque le nombre de VMs à placer est égal à 100. Il a été noté que le temps d'exécution de *Branch&Bound* et la programmation dynamique dépasse $8.10^{-3}s$. En revanche, le temps d'exécution de l'algorithme OCF varie entre $6.10^{-3}s$ et $2.10^{-3}s$.

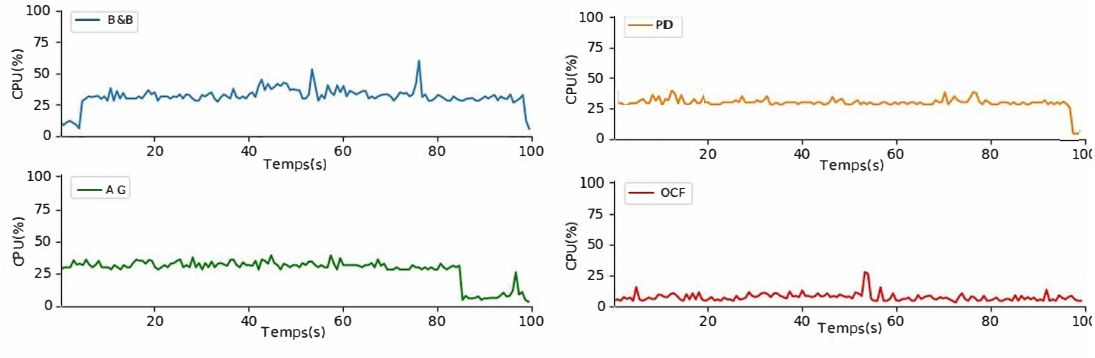
Cette analyse des résultats obtenus permet de tirer les conclusions suivantes :

- Les deux méthodes *Branch&Bound* et la programmation dynamique, représentant la démarche exacte pour résoudre le problème de sac à dos appliqué au placement des VMs dans un centre de données, garantissent des solutions optimales qui résident dans l'utilisation maximale des ressources CPU du serveur pour les machines qui lui sont affectées. Cela conduit à l'objectif principal du problème qui consiste à assigner

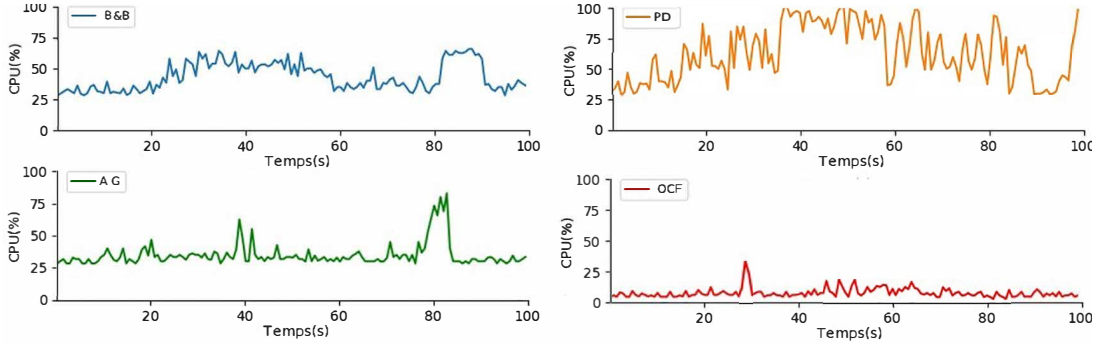
1. Psutil : https://psutil.readthedocs.io/en/latest/



(a) 20 VMs

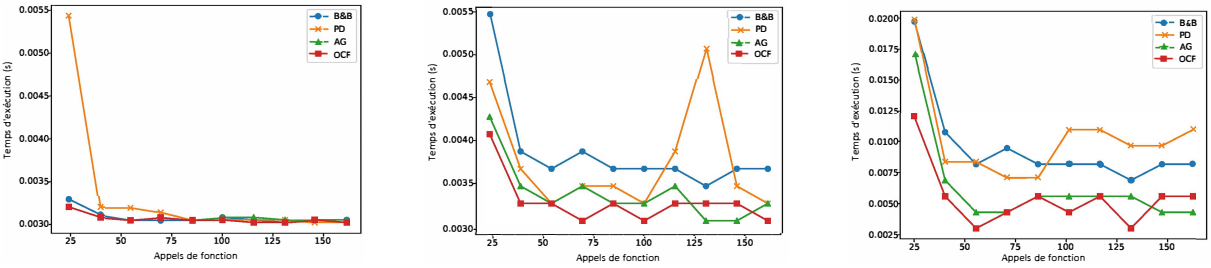


(b) 50 VMs



(c) 100 VMs.

FIGURE 27 – Le taux de ressources CPU utilisé par les quatre algorithmes pour fournir une solution avec un nombre de VMs égal à (a) 20, (b) 50, et (c) 100.



(a) 20 VMs (b) 50 VMs (c) 100 VMs.

FIGURE 28 – L'évolution du temps d'exécution des quatre algorithmes étudiés pour différents nombres de VMs.

un nombre maximum de VMs à un nombre minimum d'hôtes et à laisser les autres hôtes hors tension.

- L'étude du taux de convergence montre que, pour les méthodes exactes, la convergence est obtenue après un certain nombre d'itérations qui peut être soit prédéfini, soit dépendant de la taille des données en entrée. Cependant, pour les métaheuristiques considérées, le taux de convergence est lié en plus au processus de chaque méthode. Pour cette analyse, nous pouvons supposer que le taux de convergence des approches métaheuristiques est faiblement supérieur à celui des méthodes exactes.
- En termes d'utilisation des ressources, et comme était prouvé théoriquement, il a été établi que les méthodes métaheuristiques OCF et AG nécessitent moins de ressources CPU pour fournir une solution optimale (solution approximative) dans un temps minimum.

Pour résumer ce qui précède, le recours à des méthodes métaheuristiques pour résoudre le problème de placement des VMs dans un centre de données devient une obligation au fur et à mesure que le nombre de VMs s'augmente. En effet, selon les résultats obtenus, les méthodes exactes étudiées sont plus adaptées à un petit nombre de VMs, car elles apportent la solution optimale en utilisant les mêmes ressources. Au contraire, l'utilisation de la métaheuristique (principalement l'OCF) est recommandée pour un grand nombre de VMs, ce qui est confirmé par le tableau 5.7 qui résume la comparaison faite dans cette étude.

5.4 Consommation énergétique

Nous avons mis en œuvre l'algorithme de placement des VMs proposé. Selon les résultats discutés précédemment, l'OCF est la méthode la plus efficace, parmi les techniques considérées ; par conséquent, nous adoptons cette approche dans cette implémentation. Nous considérons un nombre d'hôtes physiques, et le modèle proposé est appliqué à chaque PM indépendamment. Les expériences sont effectuées au moyen du simulateur *CloudSim* en étendant la classe *"vmAllocationPolicy.java"* existante. Xen est désigné comme moniteur (hyperviseur) des VMs. Les simulations sont réalisées sur un ensemble de plate-formes hétérogènes dans le centre de données, c'est-à-dire des hôtes et des VMs, ainsi que des demandes de services d'entrée hétérogènes. Les besoins en ressources et la durée des demandes de service sont également générés de manière aléatoire. Au début de la simulation, un nombre constant de VMs (de différents types spécifiés) est créé, et les tâches (cloudlets) sont soumises au broker du centre de données qui fait correspondre une tâche à une VM. Lorsqu'il n'y a plus de tâches en attente, toutes les VMs sont terminées et la simulation est achevée. Comme nous nous sommes concentrés sur le placement des VMs

TABLE 5.7 – synthèse de la comparaison entre les quatre algorithmes pour un nombre de VMs égal à 50.

	Branch&Bound	PD	AG	OCF
Optimalité	Fournir une solution optimale	Fournir une solution optimale	N'apporte pas nécessairement une solution optimale	N'apporte pas nécessairement une solution optimale
Taux de convergence	Relativement faible	Prédéfini	Relativement élevé	Relativement faible
Utilisation de CPU	Relativement élevée	Relativement élevée	Relativement modérée	Relativement faible
Temps d'exécution	Relativement élevé (plus lent)	Relativement élevé	Relativement faible	Relativement faible
Efficacité	Relativement faible	Relativement faible	Relativement modérée	Relativement élevée

TABLE 5.8 – Les tests menés

Test	*test1*	*test2*	*test3*	*test4*	*test5*
VMs	50	100	150	200	250
PMs	20	40	60	80	100

sur des hôtes physiques, nous avons créé un placement *One-to-One* entre les cloudlets et les VMs. De plus, nous avons mis en place la terminaison des VMs pendant la simulation afin de garantir une simulation complète. Sans la création et la terminaison dynamiques des VMs, l'ensemble initial des VMs fonctionne jusqu'à la fin de la simulation, ce qui entraîne le blocage pendant une durée indéterminée des tâches dans le système. Les ressources CPU pour les VMs sont générées de manière aléatoire et les ressources de toutes les VMs déployées sur un hôte sont inférieures à la capacité de cet hôte. Pour évaluer l'efficacité énergétique de notre modèle, nous l'avons comparé avec un modèle standard de minimisation de l'utilisation de ressources par plusieurs serveurs, qui se base sur le problème de *Bin-Packing* et est résolu par le modèle basé sur l'algorithme FFD en termes de consommation d'énergie du système.

Nous avons comparé les valeurs de consommation énergétique totales de toutes les VMs avec les valeurs de consommation énergétique estimées (utilisant le modèle d'estimation présenté dans le chapitre 4) pour l'ensemble du serveur, un par un. Nous utilisons des tests représentant différentes échelles de centres de données en fonction du nombre de VMs et de PMs. Nous menons des expériences pour cinq tests. Pour chaque test, nous considérons des nombres de VMs et PMs différents. En effet, le nombre de VMs varie de 50 à 250 tandis que le nombre de PMs varie de 20 à 100, comme le montre le tableau 5.8.

TABLE 5.9 – La consommation de puissance et le temps d'exécution pour le modèle proposé et un modèle standard basé sur FFD.

	Modèle proposé		Modèle standard	
	Puissance (W)	Temps d'exécution(ms)	Puissance (W)	Temps d'exécution(ms)
Test1	115.2	190	217.8	118
Test2	210.2	311	664.3	107
Test3	384.1	396	922.3	135
Test4	572.8	502	1389.5	243
Test5	646.3	1044	1634.3	668

D'après le tableau 5.9, l'utilisation de notre modèle résolu par l'OCF permet de consommer moins d'énergie que le modèle existant pour tous les différents tests. Nous montrons les pourcentages d'amélioration de l'efficacité énergétique du modèle proposé

TABLE 5.10 – L'amélioration du pourcentage de notre modèle par rapport au modèle existant en termes de consommation totale d'énergie pour les 5 tests.

Test	*test1*	*test2*	*test3*	*test4*	*test5*
Optimization (%)	14.8	8.1	11.9	14.8	39.8

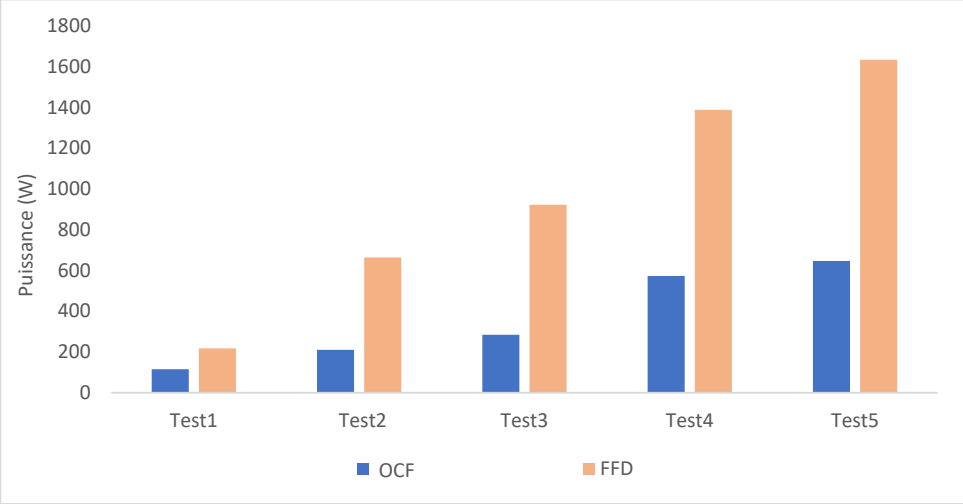


FIGURE 29 – Comparaison de la consommation totale de puissance de notre modèle (basé sur l'OCF) et du modèle standard (basé sur FFD).

par rapport à la minimisation des ressources par l'algorithme FFD pour les 5 tests considérés. Notre modèle atteint une amélioration de 14% de l'efficacité énergétique par rapport à l'approche existant dans *test1*, comme le montre le tableau 5.10. Dans les autres tests, le modèle introduit maintient l'amélioration de l'efficacité énergétique d'au moins 8% par rapport au modèle standard, avec une amélioration significative aboutit à plus de 39% pour *test5*. Afin de mieux visualiser les résultats significatifs de notre modèle en terme d'efficacité énergétique par rapport à FFD, nous avons établi un diagramme dans la figure 29. Cette dernière montre la supériorité du modèle proposé par rapport au modèle existant en économisant l'énergie pour tous les test, ce qui est une réalisation considérable pour l'optimisation de la consommation énergétique dans un centre de données.

5.5 Conclusion

L'utilisation des technologies de l'informatique dans le Cloud a récemment accru de façon drastique, entraînant une augmentation de la taille des centres de données. Un grand centre de données consomme des quantités massives d'énergie puisqu'il est constitué de nombreuses unités de calcul. La partie majeure de l'énergie consommée dans ces centres de données provient de ses PMs qui disposent de ressources hétérogènes (par exemple CPU et mémoire). Le processus de placement des VMs dans les PMs a un impact direct

sur l'efficacité énergétique dans ces infrastructures. Toutefois, allouer un nombre statique de VMs à des serveurs hétérogènes dans un délai précis constitue un problème complexe dans tout centre de données. C'est pourquoi, dans ce chapitre, nous avons développé une approche pour le placement statique des VMs afin de minimiser la consommation totale d'énergie d'un centre de données. Le problème a été formulé selon un modèle mathématique sous forme d'un problème de sac à dos. Sachant que les profils de consommation du CPU des VMs et des PMs étaient connus à l'avance et utilisés pour prendre la décision de placement.

L'étude présentée, dans ce chapitre, suit une politique particulière qui consiste à maximiser l'utilisation des ressources dans un serveur (CPU par exemple). La politique adoptée illustre la stratégie globale du placement des VMs qui consiste à concentrer le traitement dans un nombre minimum de serveurs tandis que les autres serveurs peuvent être éteints. Le problème de sac à dos a été traité et résolu par différents algorithmes. D'abord, une étude comparative de quatre méthodes a été menée : *Branch&Bound* et la programmation dynamique qui appartiennent à la catégorie des méthodes exactes, l'AG et l'OCF, qui donnent des résultats approximatifs et appartiennent à la catégorie des méthodes métaheuristiques. Ensuite, et en se fondant sur cette comparaison, nous avons adopté l'algorithme OCF pour élaborer les simulations du processus de placement des VMs pour analyser l'efficacité énergétique de la politique proposée. Ces expériences de simulation ont permis de démontrer la pertinence de notre approche d'optimisation pour le placement statique des VMs. Notre modèle a été comparée, en termes de réduction de la consommation d'énergie, à au modèle standard existant qui vise à minimiser la consommation des ressources en se basant sur le problème de *Bin-Packing* résolu par l'approche FFD. Les résultats expérimentaux ont montré que le modèle proposé était plus performant que celui existant dans la littérature. Notre modèle a amélioré l'efficacité énergétique jusqu'à 39% par rapport à l'approche existant tout en prenant quelques secondes de temps d'exécution supplémentaires, et il a présenté également une bonne évolutivité avec l'augmentation du nombre des VMs.

Nous avons vu dans ce chapitre comment répartir équitablement les VMs sur les PMs afin de réduire la consommation d'énergie et d'optimiser les performances. Nous avons également soulevé que le problème que cette répartition est statique, ce qui signifie que nous connaissons à l'avance le nombre de toutes les VMs et PMs, et donc adaptée à des charges qui ne varient que peu au cours du temps. Dans le monde du *Cloud Computing*, il est souvent intéressant de pouvoir réagir à la demande à un moment donné. Pour ce faire, il sera indispensable d'utiliser des mécanismes de redéploiement dynamique des charges de travail en utilisant des migrations en temps réel des VMs (c'est-à-dire sans interruption de service) fournie par l'infrastructure de la virtualisation. De ce fait, le chapitre suivant

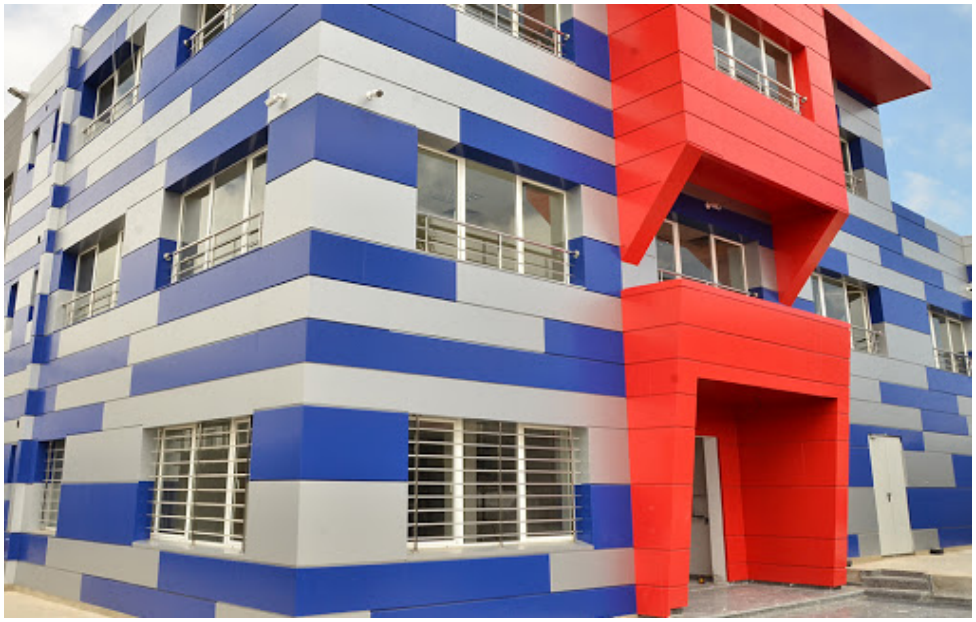
étudie un modèle basé sur le *Machine Learning* pour surveiller le processus de migration en temps réel des VMs ; le modèle permet d'identifier l'algorithme de migration en temps réel le plus approprié pour des mesures de performance données, à la base d'un diagnostic système effectué au préalable.

6

Migration en temps réel des VMs et SLAs dans un centre de données

---

Ce chapitre présente le modèle proposé pour contrôler et gérer le processus de migration en temps réel des VMs dans un centre de données. Nous adressons le problème de sélection de la technique optimale de migration en temps réel, en utilisant les approches du Machine Learning et en se basant sur un jeux de données réel issu de différentes opérations d'exécution des VMs et des charges de travails sur un cluster.



Maroc Datacenter : http://www.marocdatacenter.ma/about-mdt.php

6.1 Introduction

Au cours de la dernière décennie, la demande de techniques de gestion dynamique des ressources dans les centres de données a considérablement accru. L'objectif est de minimiser la consommation énergétique tout en maximisant l'utilisation des ressources matérielles afin de réduire le coût opérationnel et l'impact environnemental [225]. La virtualisation est l'une des technologies clés pour l'exploitation efficace des centres de données [112], car elle permet une meilleure utilisation des ressources en faisant exécuter plusieurs VMs sur un hôte physique. Pour s'adapter à des charges de travail fluctuantes et optimiser dynamiquement l'utilisation des ressources, les VMs sont migrées en temps réel [14], c'est-à-dire déplacées d'un hôte physique à un autre pendant que la VM poursuit son fonctionnement. La migration en temps réel permet d'équilibrer la charge, de consolider, de tolérer les pannes et de faciliter la maintenance [50].

La migration en temps réel est une technique qui permet aux VMs de se déplacer d'une PM à une autre. Contrairement à une méthode de base qui fait d'abord passer la VM cible en veille sur la PM source, puis la fait reprendre sur la PM de destination (également appelée migration à froid), le plus grand avantage de la migration en temps réel est que la VM cible subit une courte période d'interruption de service. La migration en temps réel est une technique essentielle pour déplacer de façon offensive les VMs d'un centre de données en cours d'exécution, du fait que les utilisateurs des VMs ne veulent jamais que leurs services soient interrompus par des changements d'attribution des VMs. Supposons que la migration à coût réduit prenne une minute pour transférer tous les états des VMs et qu'une migration à froid se produise une fois par semaine, alors le ratio du temps d'arrêt des VMs par rapport à l'ensemble du fonctionnement du centre de données est de :

$$\frac{1}{(60 \times 24 \times 7)} \approx 0.01\%$$

Il s'agit de la plus longue interruption de service pour ne pas violer le SLA de 99,99% de temps de fonctionnement, ce qui est courant à présent dans le contexte de *Cloud Computing*. Par conséquent, la migration à froid ne peut pas être utilisée pour une relocalisation des VMs, et une migration en temps réel est essentielle. La migration en temps réel d'une VM est une opération relativement dispendieuse car elle implique l'envoi de plusieurs gigaoctets d'état volatil de la VM en cours d'exécution de l'hôte source à l'hôte de destination. Au cours des années, plusieurs algorithmes de migration en temps réel ont été proposés [14, 25, 168, 58]. Comme nous l'avons souligné dans la sous-section 2.2.2, chacun de ces algorithmes présente des caractéristiques de performance différentes qui dépendent non seulement de l'algorithme lui-même, mais aussi de l'état du système hôte, du réseau d'interconnexion et, dans une plus large mesure, de la charge de travail exécutée à l'intérieur

de la VM elle-même.

Les principaux fournisseurs de plateformes de *Cloud Computing*, *Amazon*, *Google* et *Microsoft*, recourent tous à des techniques de virtualisation dans leurs centres de données. Un problème important consiste à sélectionner la meilleure technique de migration en fonction des politiques d'exploitation, des caractéristiques de la charge de travail dans la VM, de l'état des hôtes concernés et des SLAs existants. Il y a eu un nombre important de tentatives de modélisation des performances de la migration en temps réel (voir section 3.4), néanmoins, étant donné les différents algorithmes de migration et le vaste espace de paramètres, les modèles analytiques ou probabilistes simples n'atteignent pas une précision de prédiction satisfaisante. Une autre lacune des travaux existants est que tous les algorithmes de migration en temps réel couramment utilisés ne sont pas modélisés et que des mesures de performance importantes manquent.

Étant donné que les performances des différentes mesures de la migration en temps réel dépendent de la charge de travail exécutée à l'intérieur de la VM, ainsi que de l'état des ressources de l'hôte (cela inclut les VMs colocalisées), la question qui se pose est de savoir quel algorithme est le moins susceptible de violer les SLAs tout en satisfaisant les autres contraintes opérationnelles d'un centre de données. Les ressources du système supplémentaires consommées par un algorithme de migration en temps réel ne doivent ni provoquer un ralentissement du système ni affecter les autres VMs ; cela est particulièrement vrai lorsque la migration en temps réel est utilisée pour éliminer les points chauds : si l'hôte source est déjà surchargé, toute consommation supplémentaire de ressources doit être minimisée. Le problème devient plus complexe si des SLAs garantissent, par exemple, une dégradation des performances inférieure à 5%.

D'autre part, les centres de données modernes gèrent des centaines de milliers de serveurs pour accueillir des millions d'utilisateurs dans le monde entier. Les serveurs d'un centre de données génèrent de nombreuses informations, des journaux de performances des VMs aux capteurs matériels. Cela ouvre de nombreuses possibilités pour les systèmes pour la gestion des centres de données. [182] ont été parmi les premiers à utiliser le *Machine Learning* pour prédire l'efficacité de l'utilisation de l'énergie des centres de données en utilisant $200,000$ échantillons de données collectées dans les centres de données de *Google* sur une période de deux ans. Le modèle prend en compte 43 paramètres d'entrée distincts ; la construction d'un modèle analytique prenant en compte un si grand nombre de paramètres serait pour autant peu pratique.

Prédire les performances d'une migration en temps réel en fonction de l'état de la VM et de l'hôte physique sous-jacent est encore moins adapté à une analyse analytique. En effet, il existe non seulement différents algorithmes de migration en temps réel, mais aussi plusieurs mesures intéressantes pour chaque technique. Par exemple, Pour n algorithmes

et m métriques, il faut construire $n \times m$ modèles analytiques qui dépend chacun d'un ensemble différent de paramètres. Les techniques du *Machine Learning* sont des solutions idéales pour générer automatiquement des modèles pour les différentes métriques et les algorithmes de migration en temps réel disponibles en utilisant les données de profils recueillies dans les centres de données. Une telle configuration permet également d'ajouter facilement de nouveaux algorithmes ou mesures.

Dans notre thèse, nous utilisons des techniques du *Machine Learning* pour obtenir un modèle polyvalent capable de prédire avec précision les mesures clés de différents algorithmes de migration en temps réel. Compte tenu de l'utilisation des ressources des hôtes physiques et des caractéristiques de la charge de travail de la VM, le modèle présenté prévoit six mesures clés de la migration en temps réel (temps total de migration de la VM, quantité totale de données transférées, temps d'arrêt de la VM, dégradation des performances de la VM, et utilisation du CPU et de la mémoire sur les hôtes physiques) avec une grande précision. Le modèle peut être intégré dans des frameworks de migration existants afin de sélectionner l'algorithme de migration en temps réel le plus approprié pour le déplacement d'une VM.

Nous montrons quelles caractéristiques d'entrée sont pertinentes pour modéliser la migration en temps réel et décrivons une mise en œuvre efficace pour profiler ces caractéristiques. À partir des profils de plus de $40,000$ migrations en temps réel de VMs exécutant une grande variété de charges de travail, nous étudions différentes techniques d'apprentissage supervisé pour générer le modèle de prédiction. Grâce à l'approche du *Machine Learning*, de nouveaux algorithmes de migration et de nouvelles caractéristiques de profil peuvent être facilement intégrés, ce qui rend la procédure proposée plus extensible et flexible. De manière récapitulative, les principales contributions de ce travail sont les suivantes :

- Nous montrons que parmi les algorithmes de migration en temps réel actuels, il n'existe pas une meilleure technique pour tous les cas. Le choix du "bon" algorithme pour une situation donnée peut améliorer de manière significative l'efficacité des ressources et réduire les violations des SLAs.
- Nous présentons une approche de modélisation basée sur le *Machine Learning* pour prédire les performances importantes des algorithmes de migration en temps réel pour une VM donnée. Le modèle présenté peut prédire plusieurs métriques cibles pour plusieurs algorithmes de migration en temps réel couramment utilisés, de manière flexible et automatisée.
- Grâce à une simulation sur *CloudSim*, nous démontrons comment l'utilisation du modèle dans un framework de migration en temps réel existant pour sélectionner

automatiquement l'algorithme de migration approprié peut réduire de manière significative le nombre total de violations des SLAs et améliorer l'utilisation des ressources.

Le reste de ce chapitre est organisé comme suivant : dans la section 6.2, nous décrivons les détails et les étapes de la modélisation du processus de migration en temps réel suivant la théorie du *Machine Learning*. Nous décrivons les caractéristiques d'entrée sélectionnées pour la phase d'entraînant ; ensuite, nous étudions les techniques d'apprentissage supervisé utilisées pour l'élaboration du modèle. La section 6.3 présente les résultats expérimentaux, avec une comparaison détaillée et basée sur plusieurs critères, y compris l'erreur moyenne absolue et l'erreur moyenne relative données par chaque méthode de régression employée (voir l'annexe A). Ensuite, en se basant sur une plate-forme de simulation *CloudSim*, nous introduisons un exemple d'utilisation du modèle proposé, en l'incorporant au sein d'un processus de migration dans un environnement qui reproduit un centre de données (section 6.4). La section 6.5 présente la conclusion du chapitre.

6.2 Modélisation de la migration en temps réel des VMs

L'une des principales raisons de succès du *Machine Learning* est la conception d'un bon ensemble de caractéristiques d'entrée, en d'autres termes, une bonne représentation des données d'entrée du modèle. La conception des caractéristiques requiert des connaissances spécifiques du domaine et une analyse minutieuse des caractéristiques d'entrée susceptibles d'être corrélées avec les paramètres de sortie modélisés. Comme la migration en temps réel nécessite de copier la mémoire entière d'une VM d'un hôte à un autre, la taille de la mémoire de la VM et la largeur de la bande passante sont fortement corrélées au temps total de migration et à la quantité de données transférées. Pour les algorithmes itératifs basés sur le processus *Pre-Copy* qui copient de manière répétée les pages modifiées vers l'hôte de destination, la vitesse à laquelle la VM infecte les pages et l'ensemble de travail de la VM sont importants. Les algorithmes qui permettent de compresser les données profitent de la connaissance de l'entropie pour estimer la compressibilité des données. La sous-section 6.2.1 explique plus en détails le processus de sélection des caractéristiques.

Notre modèle prédit six mesures de performance (sous-section 2.2.2.2) pour chacun des cinq algorithmes de migration en temps réel (sous-section 2.2.2.1). Nous construisons un sous-modèle dédié pour chaque combinaison de $\prec$ *algorithme, métrique* $\succ$, ce qui donne un total de 30 sous-modèles. Chaque modèle est formé séparément. Le même vecteur d'entrée comprenant des propriétés qui représentent l'état de la VM, de la source et de l'hôte de

destination est fourni à chaque sous-modèle. La sortie d'un modèle est une prédiction pour un algorithme donné de la métrique pour laquelle il a été entraîné.

Pour des raisons de simplification, les sous-modèles sont logiquement agrégés en un modèle global, comme le montre la figure 30. Le modèle agrégé prend en entrée l'algorithme de migration en temps réel, la métrique souhaitée et les paramètres d'entrée profilés ≺*algorithme, métrique, vecteur des caractéristiques*≻. Au niveau intérieur, l'algorithme et la métrique sont utilisés pour sélectionner le sous-modèle approprié, qui est ensuite alimenté par le vecteur de caractéristiques en entrée. Ainsi, la sortie du sous-modèle sélectionné est égale à la sortie du modèle agrégé.

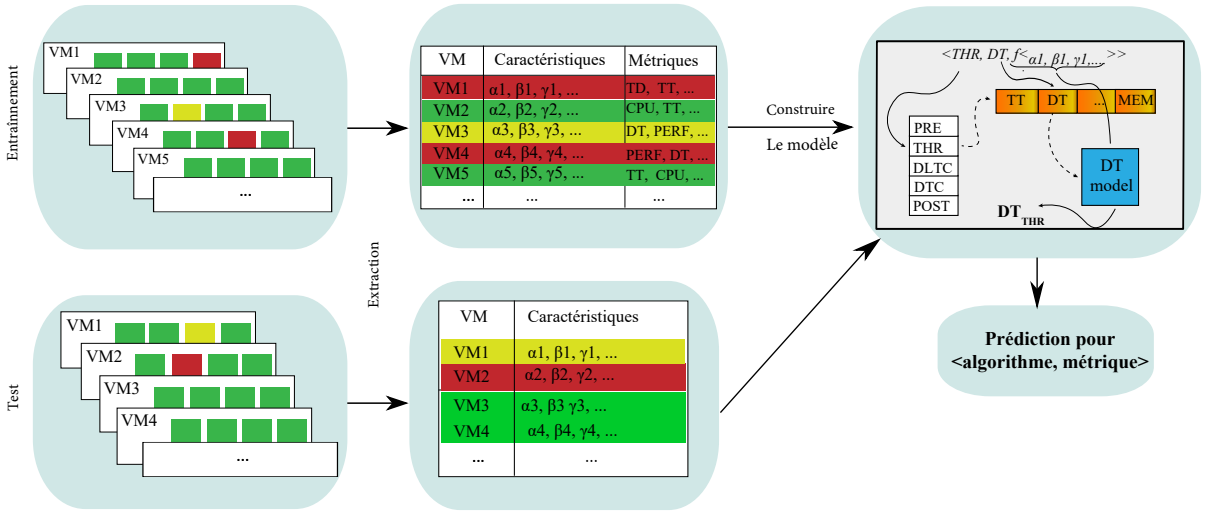


FIGURE 30 – Vue d'ensemble du modèle introduit.

6.2.1 Paramètres du modèle

Les caractéristiques d'entrée sélectionnées sont énumérées dans le tableau 6.1. Les six dernières caractéristiques du tableau sont calculées à l'aide d'autres caractéristiques. Ces paramètres sont choisis de manière à ce que, dans leur ensemble, ils couvrent tous les aspects des systèmes concernés qui peuvent avoir un impact sur les métriques estimées. Étant donné que la migration en temps réel nécessite un transfert de la mémoire de la VM vers un hôte de destination via un réseau, la taille de la mémoire allouée, $Taille_{VM}$, et la bande passante réseau disponible pour le transfert, *Page Transfer Rate* (PTR), sont des caractéristiques importantes pour tous les modèles. Notons que $Taille_{VM}$ indique la taille de la mémoire réellement allouée à la VM, et non la taille maximale de mémoire assignée. De même, l'utilisation $CPU_{utilisée}$, le nombre d'instructions retirées par seconde (MIPS) et l'utilisation réseau $Réseau_{utilisé}$ d'une VM sont nécessaires pour saisir et estimer la

TABLE 6.1 – Les caractéristiques d'entrée du modèle de prédiction.

Source	Caractéristique	Description
VMM	Taille de VM	Nombre de pages allouées à la VM
	Taux des *Dirty Pages*	Nombre moyen de pages modifiées par seconde
	Taille de l'ensemble fonctionnel	Nombre de pages modifiées lors de profilage
	Entropie de l'ensemble fonctionnel	Entropie de la mémoire de l'ensemble fonctionnel
	Entropie de l'ensemble non fonctionnel	Entropie de la mémoire de l'ensemble non fonctionnel
	Mots modifiés par page	Nombre de mots modifiés dans les pages modifiées
Hôte source	Taux de transfert des pages	Largeur de bande réservée pour la migration en temps réel
	Instructions par seconde	Le nombre d'instructions retirées par seconde
	Utilisation du CPU par la VM	Utilisation du CPU pour le traitement de la VM
	Utilisation du réseau par la VM	Utilisation du réseau pour le traitement VM
Hôtes source & destination	CPU consommée sur l'hôte	Utilisation du CPU sur les hôtes concernés
	Mémoire consommée sur l'hôte	Utilisation de la mémoire sur les hôtes concernés
Compressé	Taux pondéré relatif de transfert des pages	Taux relatif pondéré de transfert de pages par rapport au taux de *Dirty Pages*
	Taille de l'ensemble non fonctionnel	Nombre de pages non modifiées pendant la période de profilage
	Profit de la compression delta	Profit prévu de la technique de compression delta
	Profit de CPU throttling	Profit prévu de la technique de CPU-*throttling*
	Taille compressée de l'ensemble fonctionnel	Taille prévue de l'ensemble fonctionnel après compression
	Taille compressée de l'ensemble non fonctionnel	Taille prévue de l'ensemble non fonctionnel après compression

,

dégradation des performances lors de la migration en temps réel. De plus, les algorithmes basés sur la méthode *Pre-Copy*, qui copient itérativement la mémoire infectée vers l'hôte de destination, nécessitent le taux d'infection des pages, *Page Dirty Rate* (PDR), et la taille de l'ensemble de fonctionnement, *Working Set Size* (WSS). La relation entre le PDR et le WSS est que le premier représente le nombre de *Dirty Pages* au cours d'une période de profilage, tandis que le deuxième représente toutes les *Dirty Pages* pendant toute la phase de profilage. Quant aux techniques impliquant la compression des données, l'entropie de l'ensemble de fonctionnement, *Working Set Entropy* (WSE), et de l'ensemble (non)-fonctionnel, *Non-Working Set Entropy* (NWSE), ainsi que l'utilisation du CPU et de la mémoire de l'hôte source et de destination, $CPU_{SRC|DST}$ et $MEM_{SRC|DST}$, sont incluses. Pour estimer l'efficacité de la compression delta, le nombre de mots modifiés par page, *Modified Words Per Page* (MWPP), est établi.

Les caractéristiques composées représentent des combinaisons spéciales à partir des caractéristiques régulières qui permettent une meilleure prédiction. Pour les algorithmes basés sur la méthode *Pre-Copy*, le taux de transfert relatif pondéré des pages est un paramètre combiné calculé comme suivant :

$$Taux_{PTR} = max\left(WSS, WSS \times \left({}^{PDR}/_{PTR}\right)^2\right)$$

pour indiquer que si le taux de *Dirty Pages* est supérieur au taux de transfert de pages, l'algorithme de *Pre-Copy* ne converge pas et les données restantes seront transférées après l'activation de la condition d'arrêt. Ainsi, l'avantage de l'étranglement (*Throtteling Benefit*) donné par :

$$BF_{THR} = PDR \times CPU_{utilisée}$$

Ce paramètre permet aux modèles de mieux prendre en compte le fait que la technique d'étranglement du CPU est inefficace pour les charges de travail qui infectent plusieurs pages différentes avec une faible utilisation du CPU. L'algorithme de compression *Run-*

Length utilisé par la technique de compression delta est plus efficace lorsque le nombre de mots modifiés est faible. Voici une approximation de cet avantage de la compression delta en tenant compte de la taille de l'ensemble de fonctionnement, du taux d'infection de la page et du nombre de mots modifiés par page :

$$DLTC.BF = WSS \times \sharp mots_par_page/MWPP$$

En ce qui concerne la technique de compression de données, l'entropie des données non compressées détermine son efficacité. La taille estimée de l'ensemble fonctionnel (respectivement non fonctionnel) compressé, $WSS_{taille.estimée}$ (respectivement $WSS_{taille.estimée}$), est calculée, en multipliant la taille de cet ensemble par son entropie E_{WSS} (respectivement E_{NWSS}) comme suivant :

$$WSS_{taille.estimée} = WSS \times E_{WSS}$$

$$NWSS_{taille.estimée} = NWSS \times E_{NWSS}$$

La sous-section 6.3.2 analyse si les caractéristiques sélectionnées sont à la fois pertinentes et suffisantes pour une modélisation précise.

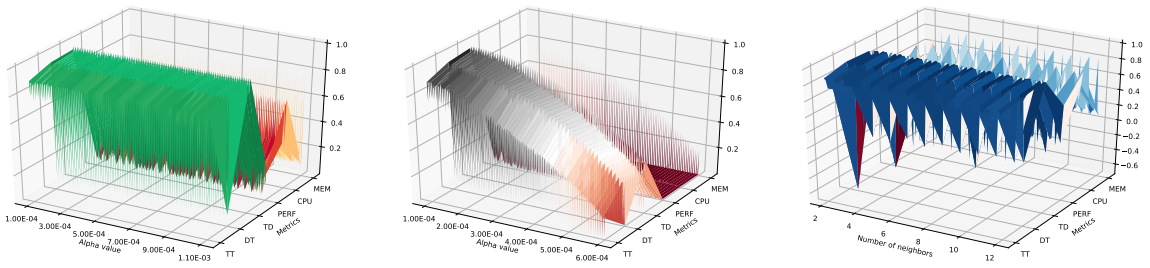
6.2.2 Génération du modèle

La production manuelle de modèles distincts pour toutes les combinaisons de techniques et de mesures cibles est difficile en raison des effets importants sur le système visé. Dans ce travail, les différents sous-modèles sont générés automatiquement à l'aide de techniques du *Machine Learning* supervisées. Nous utilisons différentes approches de régressions : LR, régression LASSO, régression *Ridge*, SVR, SVR avec *Bagging* et KNNR.

La LR est une technique de régression courante et simple qui s'adapte aux échantillons donnés en utilisant une ligne droite avec une erreur minimale. Cette technique semble un choix raisonnable s'il existe une relation linéaire claire entre l'ensemble de données et la valeur visée ; cependant, en utilisant la LR pour traiter plusieurs caractéristiques, le problème de sur-ajustement est plus susceptible de se produire, ce qui peut diminuer la précision du modèle en raison de la forte corrélation entre les caractéristiques. Ainsi et pour résoudre ce problème, la régression *Ridge* et LASSO sont introduites, comme une forme de régularisation de la LR. D'autre part, la SVR est une technique de régression destinée à des ensembles de données plus complexes qui révèlent une relation non linéaire entre les caractéristiques et la valeur cible. Nous utilisons une fonction de base radiale pour la fonction de perte avec un paramètre de pénalité égale à 10, et les caractéristiques d'entrée sont normalisées. Une autre technique de modélisation utilise l'agrégation du modèle

SVR pour améliorer la précision des prédictions, également connue sous le nom de *Bagging*, construit plusieurs sous-modèles avec un sous-ensemble de l'ensemble des données et adapte le modèle à l'ensemble des données. En outre, en raison d'être conceptuellement simple et d'avoir l'avantage d'être non paramétrique, ainsi que de donner d'assez bons résultats pour les grands ensembles d'entraînement, la technique KNNR est également utilisée (voir en détails dans la sous-section 2.3.2).

La plupart des algorithmes du *Machine Learning* ont des paramètres qui décrivent la configuration et ajustent la précision de l'algorithme. Dans ce travail, nous choisissons les paramètres les plus importants pour trois des modèles de régression énoncés ci-dessus, et nous entraînons chaque modèle avec différentes valeurs du paramètre sélectionné ; ensuite, nous sélectionnons la meilleure valeur en utilisant les données d'entraînement. Pour KNNR, le choix de K (le nombre de voisins) est crucial. Une faible valeur de K peut augmenter l'impact du bruit sur les résultats de la prédiction ; tout de même, une valeur élevée de K peut rendre la prédiction plus coûteuse. Par conséquent, nous avons utilisé l'ensemble de données de validation pour évaluer différentes valeurs éventuelles de K. La figure 31c montre qu'avec $K \geq 6$, KNNR fournit une bonne précision. Ainsi, nous considérons $K = 10$ comme le paramètre définitif pour l'approche KNNR. De même, le paramètre Alpha (α) a un impact majeur sur les deux modèles *Ridge* et LASSO. Par conséquent, la génération de résultats de prédiction pour différentes valeurs de α est fortement requise. D'après la figure 31b et la figure 31c, nous pouvons supposer que $\alpha = 10^{-4}$ correspond aux meilleurs scores obtenus par les deux modèles.



(a) Ridge (b) LASSO (c) KNNR

FIGURE 31 – Évolution de l'indice *R-squared* en fonction du paramètre Alpha (a)(b) et du paramètre K (c). La troisième dimension représente les métriques étudiées.

6.3 Résultats expérimentaux et évaluation

Pour chaque modèle de régression, nous considérons le couple (algorithme de migration en temps réel, métrique), ce qui donne 30 sous-modèles. Ainsi, nous évaluons la précision et l'efficacité de chaque méthode de régression appliquée à chaque couple considéré. Pour l'entraînement et le test des sous-modèles de mesure de prédiction, nous utilisons la boîte à outils *Scikit-learn*[1]. Les données d'entraînement et de test sont générées par une validation croisée et décuplée dans laquelle l'ensemble de données est d'abord divisé en 10 sous-ensembles de taille égale. Chaque sous-ensemble sert une fois d'ensemble de test, tandis que l'union des neuf autres constitue l'ensemble de données d'entraînement. Les valeurs rapportées représentent la moyenne des 10 évaluations.

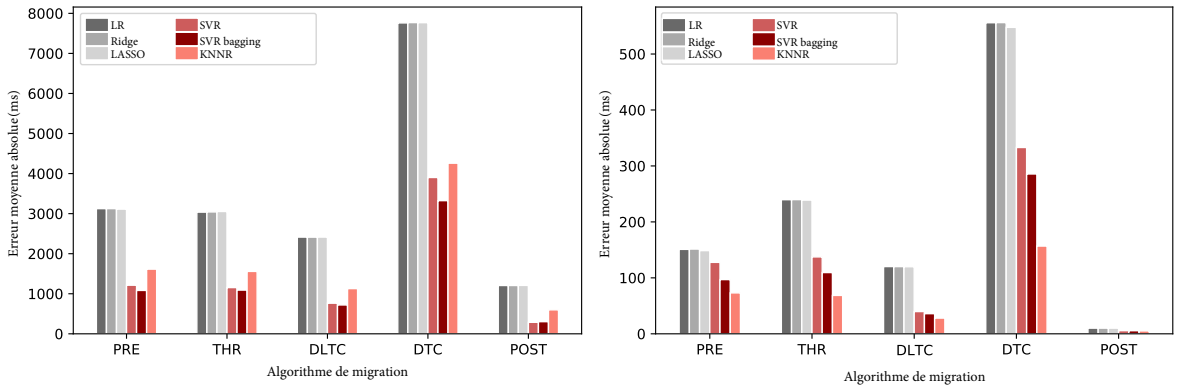
On note que le jeu de données utilisé dans le cadre de cette thèse est préparé par une équipe de recherche au sein du *Computer Systems and Platforms Laboratory*[2]. L'ensemble d'entraînement pour les modèles du *Machine Learning* ont été générées sur une période de plusieurs mois par la migration en temps réel d'un total de plus de $40,000$ VMs avec les différents algorithmes de migration en temps réel. Pour toutes les migrations, les paramètres des modèles définis et les valeurs des six mesures cibles ont été enregistrés dans une base de données. Sur chaque hôte physique, jusqu'à quatre VMs sont co-localisées et se partagent les ressources. Chaque VM exécute soit l'un des benchmarks de l'application, soit le générateur synthétique de charge de travail. Une VM aléatoire d'un hôte source aléatoire est sélectionnée et migrée vers l'un des hôtes de destination. La largeur de la bande passante disponible du réseau de migration est fixée à une valeur aléatoire comprise entre 50 et $125 Mo/s$ pour émuler un environnement à ressources limitées.

6.3.1 Précision de la prédiction de la métrique cible

La précision des prédictions pour les six différentes techniques de régression (RL, régression LASSO, régression *Ridge*, SVR, SVR avec *Bagging* et KNNR) avec les cinq algorithmes de migration en temps réel PRE, THR, DLTC, DTC et POST, et les six mesures cibles, est rapportée. Les résultats sont illustrés dans la figure 32 et la figure 33. L'erreur absolue moyenne (MAE) représente la divergence moyenne de la valeur prévue par rapport à la valeur réelle en unités absolues de la mesure (ms, MB ou %), tandis que l'erreur relative moyenne (MRE) affiche l'erreur relative moyenne de la prédiction (Annexe A).

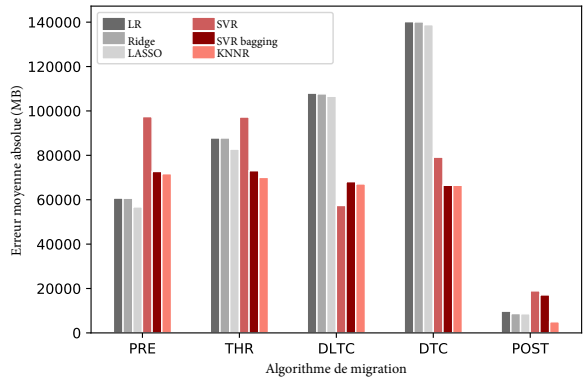
La LR, ne permet pas d'obtenir une précision satisfaisante : l'erreur moyenne de prédiction dépasse 8% pour tous les algorithmes. La raison principale de cette erreur élevée

1. Scikit-learn : cikit-learn.org/s
2. CSAP : https://csap.snu.ac.kr/

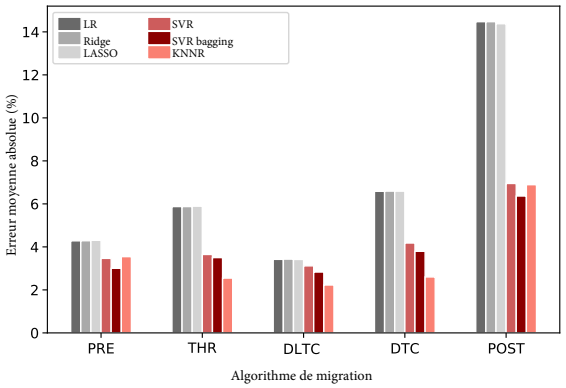


(a) TT

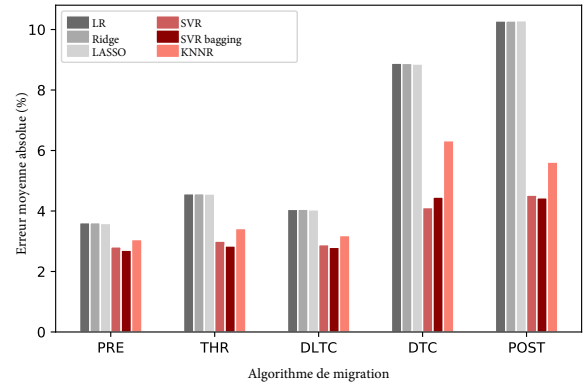
(b) DT



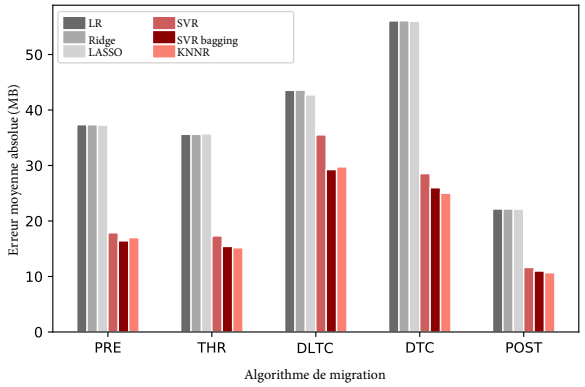
(c) TD



(d) PERF



(e) CPU

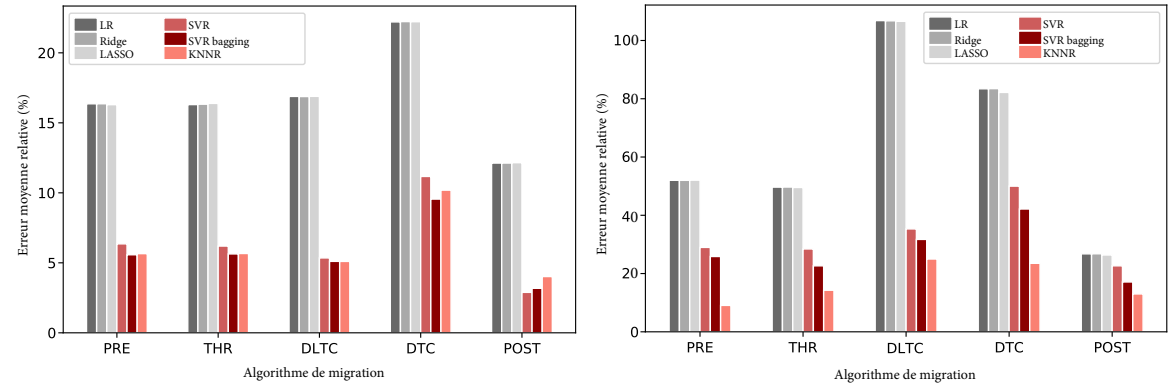


(f) MEM

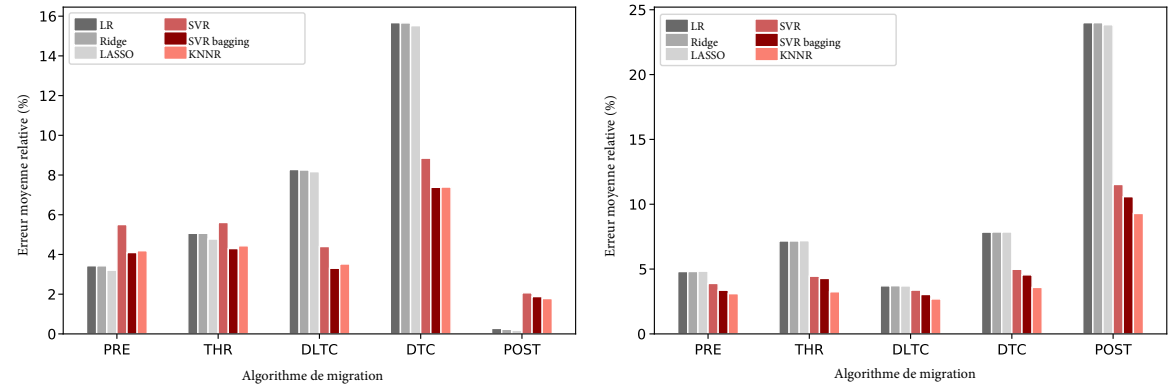
FIGURE 32 – L'erreur moyenne absolue (MAE) de modèle de régression appliqué aux six métriques cibles avec les cinq algorithmes de migration en temps réel.

provient de la corrélation complexe des caractéristiques. Une approche naïve ne peut pas saisir les complexités et ne parvient pas à entraîner le modèle avec succès. Nous remarquons, de plus, qu'il y a une légère différence entre les techniques linéaires LR, *Ridge* et LASSO qui peut être prise en considération pour prédire la métrique de temps d'arrêt (TD).

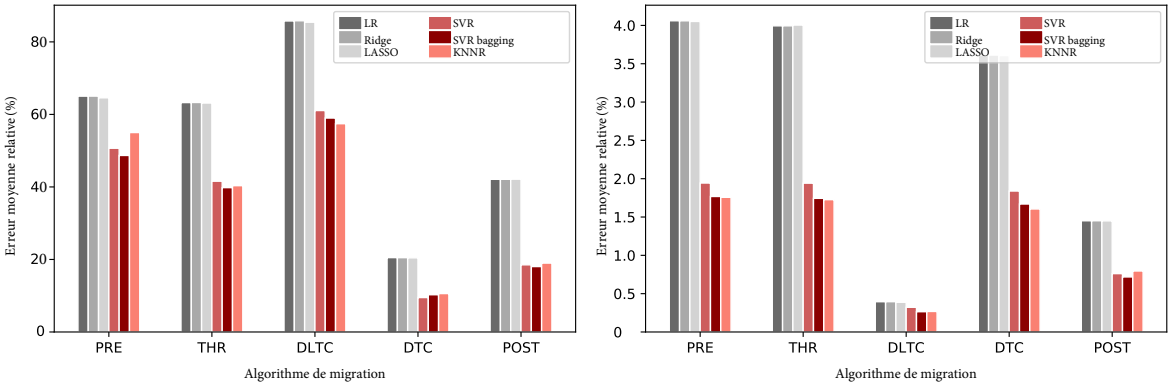
Par rapport à la LR, la précision globale des méthodes SVR est significativement



(a) TT (b) DT



(c) TD (d) PERF



(e) CPU (f) MEM

FIGURE 33 – L'erreur moyenne relative (MRE) du modèle de régression appliqué aux six métriques cibles avec les cinq algorithmes de migration en temps réel.

élevée. En particulier, les mesures cibles qui montrent une dépendance non linéaire basée sur plusieurs caractéristiques d'entrée montrent de grandes améliorations. L'erreur relative de prédiction du temps d'arrêt (DT) de la technique de compression (DLTC), par exemple, est réduite de 107% avec la LR à 35% avec la SVR. Dans certains cas, la précision relative diminue de quelques pourcents, par exemple, la prédiction du temps total de migration (TT) avec une erreur de seulement 3% pour la LR et 6% pour SVR dans le cas de

l'algorithme de migration *Pre-Copy*. Ces cas montrent une relation linéaire relativement simple entre les caractéristiques d'entrée et la valeur cible. Les noyaux complexes de la SVR n'atteignent pas les mêmes résultats ; cependant, la baisse de précision n'est pas importante. Les résultats montrent en outre que la SVR avec *Bagging* atteint une précision significative pour la plupart des mesures et surpasse visiblement la technique SVR.

De même, si l'on considère que la métrique à prédire est l'utilisation du CPU (figure 32e), la MAE passe de 80% pour les approches linéaires à 37% pour l'approche SVR. Aussi, tout en prédisant le temps d'arrêt (DT) (figure 32b), les approches non linéaires fournissent une erreur moyenne relative inférieure à 45%. Cependant, le total des données transférées (TD) (figure 32c) représente une exception avec les algorithmes de migration en temps réel *Pre-Copy* et *Post-Copy* ; la prédiction, dans ce cas, est effectuée avec une erreur plus petite pour les algorithmes linéaires (LR, Ridge et LASSO).

Plus intéressant encore, en comparant les approches non linéaires entre elles, nous pouvons remarquer que la SVR avec *Bagging* et la KNNR sont généralement plus précises que les autres techniques considérés. Les figures 32 et 33 montrent que pour prédire les mesures d'utilisation du CPU et le temps total de migration (TT), la SVR avec *Bagging* correspond à la valeur la plus basse d'erreur. Quant à la prédiction des métriques de dégradation de performance (PERF) et temps d'arrêt (DT), la KNNR correspond à la valeur minimale des deux erreurs. En outre, pour les mesures du total des données transférées (TD) et d'utilisation de la mémoire (MEM), la SVR avec *Bagging* et la KNNR ont approximativement la même précision.

Quant aux algorithmes de migration en temps réel, il convient de noter que l'algorithme DTC indique des taux d'erreur moyenne absolue et d'erreur moyenne relative remarquablement élevés (jusqu'à 90%). Cela peut être justifié par le fait que l'algorithme DTC nécessite une quantité considérable de ressources de calcul supplémentaires. L'algorithme *Post-Copy* fournit des erreurs raisonnablement réduites avec les mesures du temps total de migration (TT), temps d'arrêt (DT), total des données transférées (TD) et utilisation de la mémoire (MEM), principalement parce que l'algorithme assure un temps d'arrêt minimum et ne transfère le contenu de la mémoire qu'une seule fois. Cependant, l'algorithme *Post-Copy* peut dégrader considérablement les performances pendant la phase de reprise ; ainsi que le montrent les figures 32 et 33, la métrique de dégradation de performance (PERF) indique que les valeurs des erreurs (MAE et MRE) sont élevées pour cet algorithme.

6.3.2 Exhaustivité des caractéristiques considérées

Une question importante consiste à savoir si les caractéristiques sélectionnées reflètent tous les paramètres pertinents. Le tableau 6.2 indique le coefficient de détermination R-

TABLE 6.2 – *R-squared* agrégé des métriques étudiées.

Algorithme	TT	DT	TD	PERF	CPU	MEM
Pre-Copy	0.89	0.91	0.65	0.84	0.82	0.83
Post-Copy	0.94	0.99	0.97	0.74	0.61	0.92
THR	0.89	0.93	0.63	0.91	0.80	0.84
DTC	0.65	0.79	0.57	0.93	0.59	0.75
DLTC	0.91	0.96	0.61	0.93	0.82	0.79

squared [158] des 20 caractéristiques pour chaque algorithme et métrique avec le modèle KNNR. Le coefficient de détermination mesure l'adéquation de la prédiction à la valeur mesurée ; un *R-squared* de 1 indique une adéquation parfaite de la prédiction à la valeur cible mesurée. Nous observons une corrélation élevée pour la plupart des mesures, ce qui indique que les caractéristiques sélectionnées sont suffisantes pour faire une prédiction précise. Néanmoins, des valeurs de *R-squared* relativement faibles, comme 0.57 pour le total de données transférées (TD) avec l'algorithme de compression de données *DTC*, sont causées par une valeur cible (presque) constante ; alors, même une petite divergence dans les valeurs prédites provoque une valeur faible de *R-squared*. Nous constatons, en revanche, que l'indice *R-squared* atteint une valeur maximale, vaut 0.99, pour l'algorithme *Post-Copy* couplé avec la métrique du temps d'arrêt (DT), ce qui est confirmé par sa valeur minimale d'erreur relative qui ne dépasse pas 20%. Notons que l'erreur relative du temps d'arrêt pour l'algorithme *Post-Copy* de 20% correspond presque à seulement 1ms. Indifféremment, le *R-squared* global pour la mesure des performances est élevé car la dégradation des performances est généralement faible, sauf pour l'algorithme de migration *Post-Copy*, c'est-à-dire qu'il y a peu d'impact sur la valeur cible par les valeurs des caractéristiques en entrée.

6.3.3 Aperçu sur la durée d'entraînement et de test

Le tableau 6.3 indique le temps nécessaire pour entraîner et prédire les valeurs avec le modèle proposé. Le temps d'apprentissage (*Learning*) comprend le temps total de CPU nécessaire pour entraîner les 30 sous-modèles pour la technique du *Machine Learning* respective sur un hôte. Le temps de prédiction est le temps CPU total nécessaire pour prédire toutes les mesures cibles pour tous les algorithmes de migration en temps réel, c'est-à-dire le temps total pour évaluer une fois chacun des 30 sous-modèles. Nous observons que l'entraînement peut prendre un certain temps, en particulier dans le cas de SVR avec *Bagging*. Néanmoins, le temps de prédiction fait l'objet d'une durée suffisamment courte pour ne pas avoir d'impact sur la sélection de l'algorithme, surtout en comparaison avec la durée totale de la migration en temps réel (plusieurs dizaines de secondes). Dans le cadre d'un centre de données, il peut être souhaitable d'effectuer une nouvelle formation

TABLE 6.3 – Aperçu sur la durée d'entraînement et de test.

Algorithme	LR	*Ridge*	LASSO	SVR	SVR avec *Bagging*	KNNR
Durée d'entraînement (s)	5.0	4.1	8.8	671.0	14118.1	20.2
Durée de test (ms)	0.85	0.81	1.06	13.03	884.38	3.57

des modèles afin que ceux-ci reflètent l'ensemble des charges de travail actuelles de ses MVs. Une surcharge de calcul de quelques heures de CPU tous les quelques jours semble acceptable.

6.4 Scénario de migration en temps réel

Nous illustrons une application potentielle du modèle décrit plus haut. Le modèle est intégré dans un schéma de migration des VMs en temps réel permettant de migrer automatiquement les VMs dans le cadre de l'équilibrage de charge ou de la consolidation. Pour une charge de travail donnée, le modèle est utilisé pour identifier la meilleure technique de migration en temps réel en fonction des différentes contraintes des SLAs fournies par l'opérateur et/ou l'utilisateur. Les contraintes des SLAs sont définis de manière affinée pour répondre aux besoins d'un centre de données. L'objectif est de maximiser les profits par rapport aux objectifs définis conformément aux contraintes des SLAs présentées dans le tableau 6.4. Par exemple, la politique *P1* consiste à maintenir le taux d'interruption de service à moins de 2s, la performance relative à plus de 70%, et l'utilisation de la mémoire à moins de 300MB.

TABLE 6.4 – Politiques définissant les contraintes de SLAs .

Politique					
		P1	*P2*	*P3*	*P4*
Contraintes SLAs	Temps d'interruption maximal ($2s$)	•		•	•
	Performance minimale (70%)	•	•		•
	Utilisation de mémoire maximale (300MB)	•	•	•	
	Utilisation de CPU maximale (70%)		•	•	•

En éliminant d'abord tous les algorithmes susceptibles de violer les contraintes des SLAs données, grâce aux prédictions de notre modèle, nous sommes en mesure de déterminer un algorithme de migration en temps réel adapté à une politique donnée. Nous utilisons l'erreur relative agrégée de violation des SLAs. Cette mesure est calculée en additionnant les erreurs relatives des valeurs cibles attendues et des contraintes données. Par exemple, si le modèle prévoit un temps d'arrêt de 2.1s et que la contrainte donnée est de

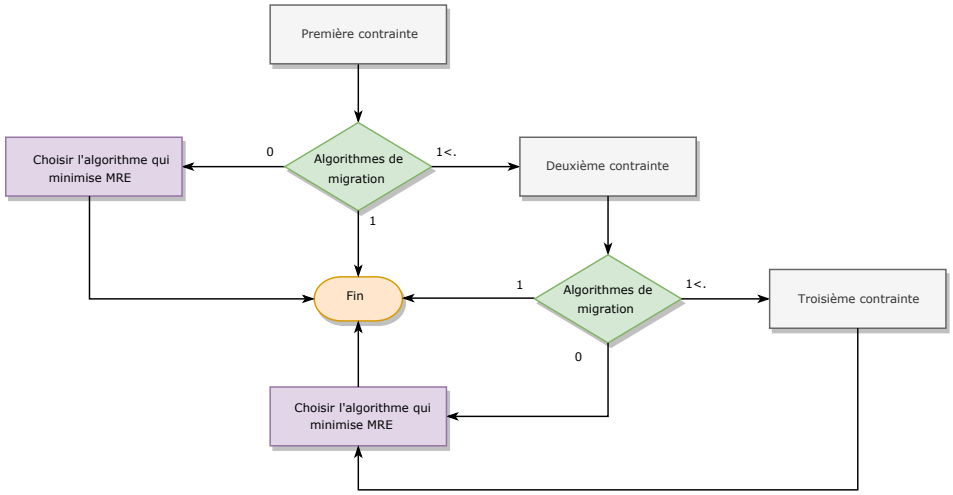


FIGURE 34 – Un aperçu du modèle incorporé dans le processus de migration en temps réel des VMs.

2s, l'erreur relative est $|(1-(2.1/2))| = 0.05$. Suite à ce processus, nous pouvons confronter deux scénarios : soit il reste plus d'un algorithme, par conséquent, celui qui cherche à atteindre la métrique cible est choisi dans une étape ultérieure, soit tous les algorithmes sont éliminés lors de la première étape, et dans ce cas, dans une deuxième étape (en considérant une deuxième contrainte) nous sélectionnons celui qui minimise l'erreur relative agrégée liée à la violation des SLAs, et ainsi de suite jusqu'à ce que toutes les contraintes considérées soient examinées (pour simplifier, nous considérons trois contraintes). Le processus est présenté dans la figure 34.

Étant donné la récurrence expérimentale et les nombreux atouts du framework *CloudSim*, tels que la capacité à modéliser des environnements virtualisés et à prendre en charge l'approvisionnement en ressources à la demande, nous avons adopté la boîte à outils *CloudSim* pour intégrer le modèle proposé dans le processus de migration en temps réel. Nous simulons un centre de données qui comprend 100 hôtes physiques, *HP Proliant G4* et *HP Proliant G5* avec un processeur *Intel Xeon3040* et *Intel Xeon3075* respectivement, tous deux avec 2 cœurs et 4 Go de mémoire. Pour comparer la migration guidée par le modèle de prédiction avec un algorithme de migration statique, 100 VMs sont migrées selon un planning définissant la VM, le temps de migration et l'état de l'hôte de destination. Le modèle permet de sélectionner l'algorithme comme décrit ci-dessus ; cependant, la migration statique utilise un seul algorithme de migration pour transférer les 100 VMs. En plus des charges de travail fournies par la boîte à outils *CloudSim*, nous utilisons des charges de travail aléatoires contenant des demandes pour fournir 50 VMs sélectionnées au hasard qui occupent toute la capacité du centre de données simulé et soumises par les utilisateurs. Chaque VM exécute une application avec une charge de travail fluctuante, qui est modélisée pour générer une utilisation du CPU et de la mémoire basée sur des

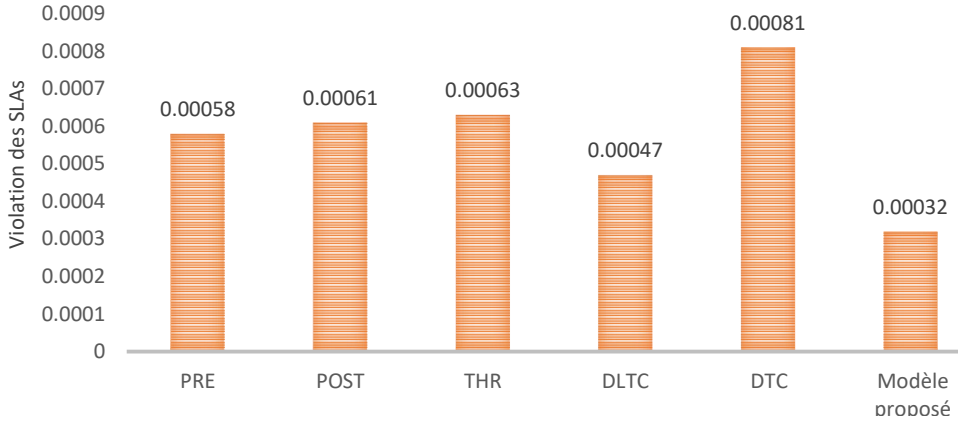


FIGURE 35 – Taux de violation des SLAs

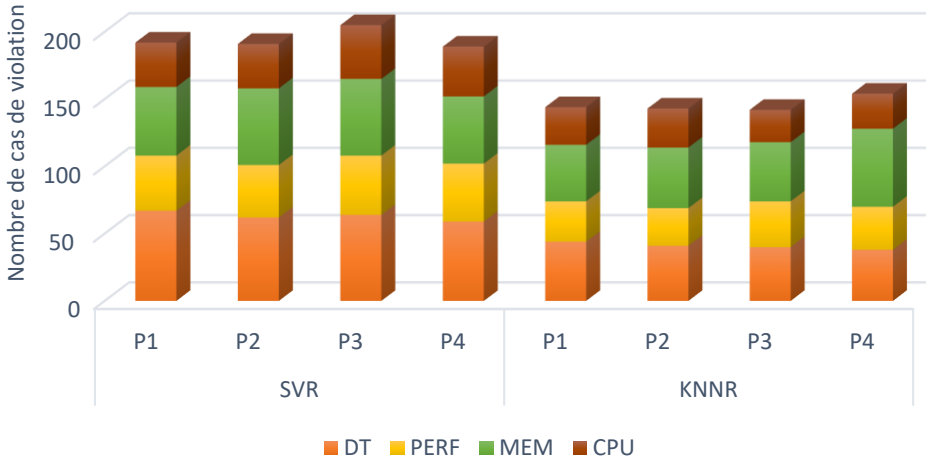


FIGURE 36 – Nombre et nature des violations pour chaque politique, en considérant la SVR avec *Bagging* et la KNNR.

variables aléatoires.

Nous évaluons le taux de violation des SLAs (décrit en annexe A) de la sélection guidée par le modèle introduit par rapport aux approches statiques. En utilisant les taux de violation des SLAs présentés dans la figure 35, il s'ensuit que la migration basée sur le modèle de prédiction surpasse chaque algorithme de migration en temps réel statique et fournit le plus faible taux de violation des SLAs. De plus, la figure 36 montre le nombre et le type de violations des SLAs pour les deux méthodes de prédiction les plus fiables (comme discuté ci-dessus) qui sont utilisées pour accomplir les contraintes des SLAs définies par les politiques données à satisfaire pendant la migration de la VM. Nous observons, également, que la prédiction basée sur la SVR avec la technique de *Bagging* conduit à plus de violations des SLAs que la prédiction utilisant l'algorithme KNNR. Pourtant, ce résultat ne peut pas être généralisé, vue que l'exemple se limite à un cas simple, et que les approches linéaires ou les techniques de SVR pourraient être plus exactes dans certains cas.

6.5 Conclusion

Au cours de la dernière décennie, la consommation énergétique des centres de données a connu une augmentation alarmante. En conséquence, le besoin de techniques de gestion dynamique des ressources dans ces structures a augmenté de manière significative. L'objectif est de réduire la consommation d'énergie tout en maximisant l'utilisation des ressources physiques afin de réduire à la fois les frais d'exploitation et l'impact sur l'environnement. La migration en temps réel est l'une des solutions les plus puissantes pour améliorer les performances des centres de données, l'efficacité énergétique et la facilité de maintenance. De nombreux schémas de migration en temps réel des VMs ont été conçus dans l'environnement de *Cloud Computing* ; chaque schéma tient compte des caractéristiques des VMs telles que l'utilisation des ressources et la dégradation des performances. Le choix de la technique optimale de migration en temps réel a jusqu'à présent été une question ouverte avec les SLAs et les contraintes opérationnelles données. Dans cette thèse, nous avons fourni une analyse d'un modèle basés sur le *Machine Learning* pour le suivi du processus de migration en temps réel des VMs ; le modèle proposé permet d'identifier l'algorithme de migration en temps réel le plus convenable pour certaines mesures de performance, sur la base d'un diagnostic préalable du système.

La technique proposée a permis d'obtenir une grande précision de prédiction pour toutes les mesures cibles et les algorithmes de migration en temps réel pour un ensemble hétérogène de machines physiques et virtuelles. Par rapport aux algorithmes statiques, le modèle présenté consiste en une amélioration remarquable de la précision de prédiction. Nous sommes convaincus que l'approche basée sur le *Machine Learning* présentée est la bonne façon d'aller vers le progrès de la migration en temps réel des VMs grâce à son adaptation facile et automatique à des nouvelles métriques cibles, des algorithmes de migration et à des environnements matériels. Utilisé dans un cadre de migration en temps réel, le modèle proposé a permis de réduire considérablement le nombre de violations des SLAs et de contraintes spécifiées par l'opérateur, et par la suite, a permis de réduire la consommation énergétique associée.

7

Le *Clustering* pour détecter les anomalies dans le centre de données

---

Ce chapitre présente un nouvel algorithme de Clustering qui permet d'identifier les anomalies potentielles dans le système d'un centre de données. Cette détection permet de prévenir les causes de dysfonctionnement ou de pannes, ce qui permet de protéger le système et de garantir la continuité d'exécution des services.



IBM : https://www.lemondeinformatique.fr/actualites/lire-zr1-version-allegee-du-mainframe-z14-d-ibm-71443.html

7.1 Introduction

Les problèmes de performance sont une norme, plutôt une exception dans les systèmes à grande échelle [184]. Les anomalies de performance et la variabilité inattendue sont courantes, même dans les infrastructures telles que les clusters *Google* et *Amazon Web Services* [59]. Outre le fait qu'ils affectent négativement la satisfaction de l'utilisateur final et la réputation du service, les problèmes de performances entraînent parfois des interruptions imprévues. Les conséquences profondes des problèmes de performances entraînent la nécessité de techniques et de systèmes pour détecter automatiquement les problèmes et guider les opérateurs vers les causes potentielles et les solutions.

En général, l'objectif du diagnostic de performance dans un centre de données est de détecter les événements de performance attendus et inattendus, d'analyser les mesures opérationnelles afin de recueillir des preuves pour localiser la ou les causes responsables du comportement observé, et de recommander des stratégies susceptibles de résoudre le problème ou de déclencher automatiquement la correction.

Par exemple, la fraude, le gaspillage et la violation des ressources ont été des préoccupations majeures dans le domaine du Cloud et de HPC. Le gaspillage de ressources dans le domaine du calcul intensif provient de diverses sources, telles que les blocages d'applications et des VMs dus à des défaillances logicielles et matérielles, les conflits dans les ressources partagées (comme les réseaux à haut débit et les systèmes de fichiers parallèles partagés ou la mémoire) et l'utilisation frauduleuse (par exemple, l'extraction de *bitcoins* et le craquage de mots de passe). Savoir quelles applications fonctionnent sur le système est une aide précieuse pour faire face aux problèmes de fraude, de gaspillage et de mauvais traitements.

La connaissance des programmes s'exécutant sur le système peut également être utilisée pour diverses optimisations au niveau du système. [75] ont présenté une méthode de planification qui utilise des modèles de consommation d'énergie spécifiques à l'application pour réduire la consommation d'énergie globale. Connaître les caractéristiques des applications les plus courantes est également utile aux concepteurs des systèmes qui prennent des décisions de conception, ou aux acheteurs des services qui peuvent prendre de meilleures décisions de financement et d'achat en se basant sur la connaissance des exigences des programmes typiques.

En général, les opérateurs de centres de données et les gestionnaires de services qui fonctionnent sur ces grands systèmes ne savent pas quelles VMs sont exécutées dans les serveurs à un moment donné. Les ressources physiques sont partagées par de nombreux utilisateurs et exécutent simultanément des centaines, voire des milliers d'applications et de VMs par jour. Ces charges de travail sont compilées par les utilisateurs selon des

paramètres différents, ce qui donne des modèles d'exécution différents même s'ils sont compilés à partir de la même source. De plus, les utilisateurs ont tendance à utiliser des noms non descriptifs pour les programmes et les scripts utilisés dans leur soumission de travail. Par conséquent, les méthodes naïves de détection des charges de travail, telles que l'examen des désignations des VMs et des scripts, ne sont pas utiles.

Un exemple plus intéressant réside dans la situation mondiale actuelle. En effet, pour contrôler la propagation de COVID-19, et suivant les conseils de l'organisation mondiale de la santé, les gouvernements ont adopté le système de confinement comme mesure de protection. L'utilisation des services liés à Internet augmente de manière significative, ce qui entraîne une utilisation croissante des centres de données et de l'industrie du Cloud. Ainsi, la plupart des centres de données fonctionnent 24/24, avec une consommation électrique permanente ; leur principal défi est de fournir à leurs utilisateurs des garanties de confidentialité, de sécurité et de performance. De leur côté, les utilisateurs finaux, qui exigent une expérience de navigation performante, exercent une forte pression sur les centres de données. Par exemple, *Microsoft*[1] a déclaré qu'elle a adopté un plan d'action global pour aider ses clients à maintenir leurs activités pendant cette période sensible. La société a déclaré qu'elle menait une enquête intensive sur les performances et les niveaux d'utilisation afin d'optimiser les services pour les clients du monde entier et de répondre aux nouveaux besoins.

Pour relever les défis mentionnés ci-dessus, nous nous concentrons, dans ce travail, sur l'emploi d'une approche de *Clustering* pour la caractérisation des charges de travail. Pour une meilleure compréhension des caractéristiques et du comportement des charges de travail, les données de surveillance sont analysées, y compris des informations détaillées sur l'utilisation des ressources telles que l'utilisation du CPU, la mémoire, les activité relatives au réseaux, etc. qui sont collectées périodiquement au moment de l'exécution à partir d'un cluster. Nous proposons un nouvel algorithme (sbWFCM) qui intègre le comportement collectif des étourneaux (*Starling Birds*). Un étourneau diffuse naturellement certaines informations à ses proches voisins pour maintenir sa position, et par la suite, ces informations sont diffusées à l'ensemble de la colonie. Selon cette analogie, nous développons une technique pour calculer le poids de chaque élément de données de manière itérative en fonction de sa position actuelle dans l'espace de caractéristiques (par rapport à ses voisins les plus proches). C'est une sorte de communication informationnelle entre l'individu (objet) et ses voisins les plus proches. Plus précisément, les apports de ce travail sont les suivants :

- Nous présentons un algorithme de *Clustering* WFCM basé sur le comportement

1. Microsoft :https://data-economy.com/microsoft-gives-cloud-priority-to-emergency-services-home-workers-and-critical-infrastructure/

des étourneaux (sbWFCM) : une nouvelle approche basée sur la densité de l'ensemble qui consiste à explorer la contiguïté topologique des voisins et à appliquer une recherche de modèle améliorant les performances de l'algorithme WFCM. La méthode proposée produit un algorithme d'interaction qui fournit différentes alternatives de *Clustering* en fonction du nombre d'itérations des instructions relatives au mouvement des oiseaux.

- Pour valider l'exactitude de l'algorithme proposé, nous utilisons un ensemble de données réel (données de surveillance collectées périodiquement au moment de l'exécution par un cluster de calcul). Les résultats du *Clustering* sont comparés au *Ground Truth* (la véritable classification des charges de travail).

Le reste de ce chapitre est organisé comme suivant : La section 7.2 montre comment le nouvel algorithme sbWFCM est formulé comme étant une variante de l'algorithme WCFM qui adopte un comportement inspiré de la nature pour formuler le paramètre du poids. La section 7.3 discute les résultats expérimentaux en menant une étude comparative, basée sur différents indices de validation du *Clustering*, entre le nouvel algorithme et l'algorithme de base WFCM. La section 7.4 présente la conclusion du chapitre.

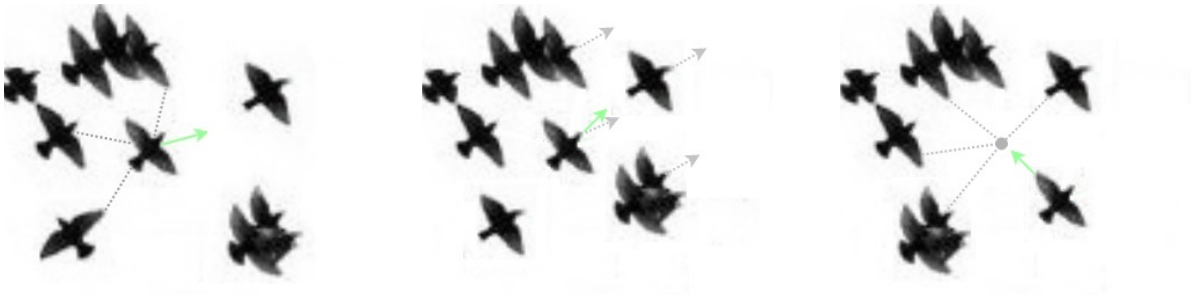
7.2 L'algorithme de *Clustering* proposé (sbWFCM)

À la base, le modèle de flockage repose sur l'hypothèse qu'il existe trois modèles de pilotage comportemental, qui décrivent la manière dont un individu utilise la position relative de ses proches voisins pour décider de sa prochaine position et direction. La séparation signifie le maintien d'une distance minimale par rapport aux oiseaux environnants, l'alignement pour adapter la direction à celle des oiseaux voisins et la cohésion qui fait référence au déplacement vers le centre de la densité perçue des oiseaux du voisinage. Les trois comportements sont modélisés comme indiqué dans le tableau 7.1, sachant que N_i^* est l'ensemble des individus dans l'aire d'interaction, w_s, w_a, w_c sont respectivement les facteurs de pondération de la séparation, l'alignement et la cohésion, r_s est la distance d'évitement requise entre les individus, e_x est la direction vers l'avant de chaque individu, g est le taux d'évitement (donné en fonction de la proximité du voisin), et σ est l'écart-type Gaussien. L'étourneau est une façon pour un individu i de répondre à ses pairs dans une petite zone qui est définie par une distance d_{il} et une trajectoire de vol des étourneaux e_{x_i}. Il en résulte une force, appelée force sociale, qui consiste en une somme des trois forces f_{s_i}, f_{a_i} et f_{c_i}, pour déterminer la position suivante de l'individu i.

$$F_{sociale_i} = f_{s_i} + f_{a_i} + f_{c_i} \tag{7.1}$$

TABLE 7.1 – La description et la formulation mathématique des trois comportements élémentaires, la séparation, l'alignement et la cohésion.

Comportement	Description	Formulation
Séparation 37a	L'individu i est orienté par une force f_{s_i} qui s'écoule dans la direction opposée à la moyenne des autres positions de l'ensemble N_i^* de sa proximité.	$f_{s_i} = -\frac{w_s}{\lvert N_i^* \rvert} \sum_{l \in N_i^*} g(d_{il}) d_{il}$ $g(d_{il}) = \begin{cases} 1 & d_{il} \leq r_s \\ \exp(-\frac{(d_{il}-r_s)^2}{\sigma^2}) & d_{il} > r_s \end{cases}$
Alignment 37b	L'individu i est attiré par une force f_{a_i} à aligner avec la direction moyenne l'ensemble de ses voisins N_i^*.	$f_{a_i} = w_a \frac{\sum_{l \in N_i^*} (e_{x_l} - e_{x_i})}{\lVert \sum_{l \in N_i^*} (e_{x_l} - e_{x_i}) \rVert}$
Cohésion 37c	L'individu i est soumis à une force f_{c_i} pour se rapprocher de la position moyenne de son voisinage N_i^*.	$f_{c_i} = \frac{w_c}{\lvert N_i^* \rvert} \sum_{l \in N_i^*} \chi_{il} d_{il}$ $\chi_{il} = \begin{cases} 0 & d_{il} \leq r_s \\ 1 & d_{il} > r_s \end{cases}$



(a) Séparation (b) Alignement (c) Cohésion

FIGURE 37 – Une description visuelle des comportements élémentaires dans le modèle de simulation de *Raynolds*.

La communication entre les individus du troupeau implique la diffusion d'informations à tous les étourneaux du troupeau après un certain temps, sans interaction directe. De cette façon, la cohérence de la volée d'étourneaux peut être obtenue en regroupant les oiseaux dans la même direction, tout en maintenant une distance par rapport aux autres groupes.

Pour développer l'algorithme proposé, cette métaphore est fondée comme suit. Chaque objet o_i est représenté comme un individu dans le troupeau d'étourneaux (cluster), et x_i correspond à sa position dans l'espace des caractéristiques. On suppose que la direction vers l'avant e_{x_i} varie en fonction de la distance à chaque prototype du cluster ; elle est donnée par 7.2. La force sociale (donnée par 7.1) est utilisée pour mettre à jour le poids attribué à chaque objet, compte tenu des expressions de séparation, d'alignement et de cohésion définies dans le tableau 7.1. On note que les valeurs des forces f_{s_i}, f_{a_i} et f_{c_i} sont

normalisées de façon que la valeur du coefficient de pondération $w_i = f_{s_i} + f_{a_i} + f_{c_i}$ soit inférieure ou égale à 1. Principalement, l'algorithme sbWFCM fonctionne comme décrit dans Algorithme 5. Notant que d_N est une distance donnée pour identifier l'ensemble des voisins N_i^*.

7.3 Résultats expérimentaux et évaluation

Pour évaluer l'algorithme proposé, nous utilisons un ensemble de données réel décrit dans la section 6.3. Plus d'une trentaine de suites et d'applications de référence reflétant des charges de travail réelles ont été utilisées. Pour générer l'ensemble de données, les benchmarks ont été réalisés sur un cluster comprenant quatre serveurs identiques avec un processeur *Skylake i5-6600* à 4 cœurs, 16 Go de mémoire, une carte réseau de 1 Gigabit conçue pour le trafic de migration. De nombreuses caractéristiques couvrant diverses fonctionnalités de la mémoire et des schémas d'utilisation du CPU d'une VM ont été prises en compte. Dans ce travail, nous nous concentrons sur les modèles d'utilisation du CPU, des entrées/sorties et de la mémoire générés par l'exécution des benchmarks *OLTPbenchmark*[2] , *SPECWeb2009*[3], et *MPlayer*[4].

2. OLTPbenchmark : http://oltpbenchmark.com
3. SPECWeb2009 : https://www.spec.org/web2009/
4. Mplayer : http://www.mplayerhq.hu/

Algorithme 5 *sbWFCM*

Entrée: X, m, C initial, w_s, w_a, w_c, r_s, d_N et σ

1: Choisir les centroïdes primaires v_k

2: **pour** *un nombre donné d'itérations* **faire**

3: Calculer le degré d'appartenance de tous les éléments dans tous les clusters en utilisant :

$$u_{ki} = \frac{1}{\sum_{j=1}^{C} (\frac{||x_i - v_k||}{||x_i - v_j||})^{\frac{2}{m-1}}}$$

4: Calculer la direction à parcourir e_{x_i} :

$$e_{x_i} = \sum_{k=1}^{C} \frac{(v_k - x_i)}{\|(v_k - x_i)\|} \tag{7.2}$$

5: **pour** *chaque* $l \neq i$ **faire**

6: Calculer la distance $d_{il} = \|(x_i - x_l)\|^2$

7: **si** $d_{il} \leq d_N$ **alors**

8: Calculer la direction à parcourir : $e_{x_l} = \sum_{k=1}^{C} \frac{(v_k - x_l)}{\|(v_k - x_l)\|}$

9: **fin si**

10: **fin pour**

11: Calculer les forces f_s, f_a, et f_c (tableau 37) et mettre à jour le poids w_i

$$w_i = f_{s_i} + f_{a_i} + f_{c_i} \tag{7.3}$$

12: Mettre à jour les centroïdes v_k en utilisant l'équation :

$$v_k = \frac{\sum_{i=1}^{N} w_i u_{ki}^m x_i}{\sum_{i=1}^{N} w_i u_{ki}^m}$$

13: **fin pour**

Pour adopter les données à notre analyse, nous considérons trois mesures de performance, le temps d'arrêt, la consommation de mémoire et la surcharge de CPU. Pour chaque mesure, nous fixons un niveau : faible ou élevé. Par exemple, pour la mesure du temps d'arrêt, si une VM dépasse un certain seuil (donné en millisecondes) lors de l'exécution d'une charge de travail, ou lors de la migration d'une PM à une autre, la possibilité

TABLE 7.2 – Les statuts définis d'une VM.

Id	Temps d'arrêt	surcharge de CPU	Utilisation de mémoire
S1	Faible	Faible	Faible
S2	Élevé	Faible	Faible
S3	Élevé	Faible	Élevé
S4	Élevé	Élevé	Faible
S5	Élevé	Élevé	Élevé

que la VM provoque un temps d'arrêt est élevée. De cette façon, nous définissons cinq statuts qui peuvent être attribués à une VM et qui sont décrits dans le tableau 7.2. De plus, nous avons vérifié la distribution des valeurs des caractéristiques, et nous avons éliminé les valeurs aberrantes en utilisant la méthode de [207].

7.3.1 Instructions d'implémentation

Nous évaluons les résultats d'une exécution typique de l'algorithme où le paramètre de *Fuzziness* a été fixé à $m = 2$, les données ont été normalisées à la moyenne zéro, et d'autres paramètres sont donnés dans le tableau 7.3. Le nombre d'itérations est également un paramètre important qui nécessite un réglage approprié. L'expérience consiste en 50 essais pour assurer une évaluation statistique de l'algorithme. Pour valider les différentes alternatives proposées par l'algorithme, le *Xie-Beni Index* (XBI) et le *Adjusted Rand Index* (ARI) sont examinés et brièvement repris dans l'annexe A.3. L'ARI est l'un des indices de validation externe les plus recommandables utilisés pour mesurer la similarité de deux partitions. Dans ce travail, ces partitions sont la *Ground Truth*, c'est-à-dire la partition réelle, et la partition hypothétique produite par l'application de la méthode de *Clustering*. Pour la visualisation, l'approche classique de réduction de la dimensionnalité des données de *Sammon* est utilisée [224]. Le modèle *Sammon* crée une projection non linéaire d'une dimension élevée dans un espace de dimension plus faible, visant à maintenir la structure des distances entre points dans les deux espaces.

TABLE 7.3 – Spécifications des paramètres avec les valeurs par défaut.

Paramètre	valeur par défaut
d_N	0.8
w_s	0.5
w_a	0.5
w_c	0.8
r_s	0.05
σ	0.8

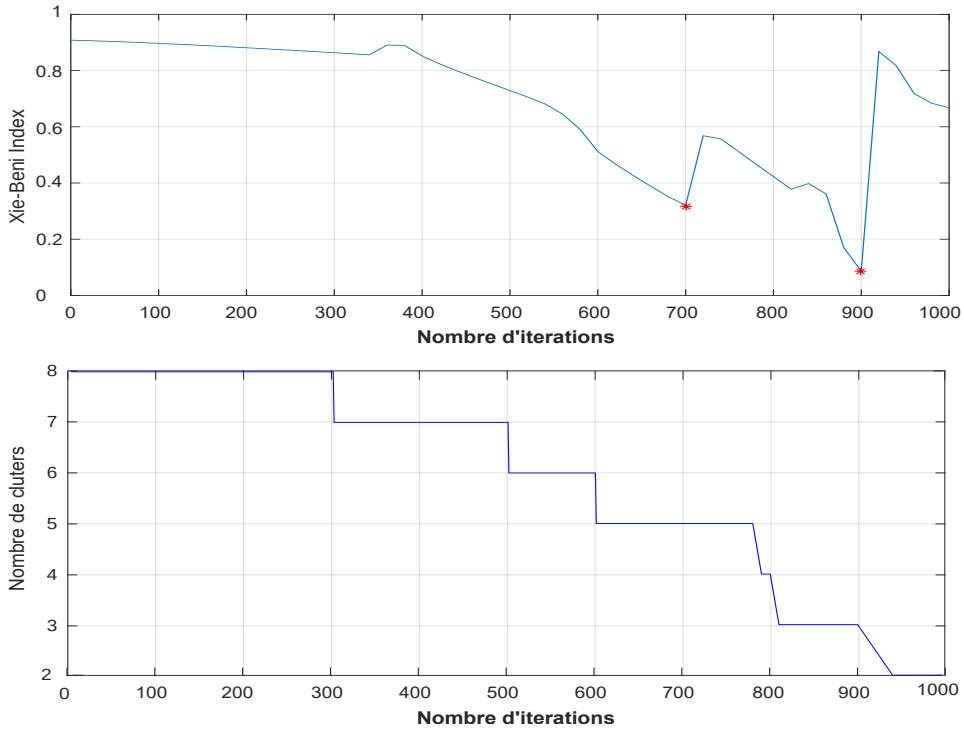
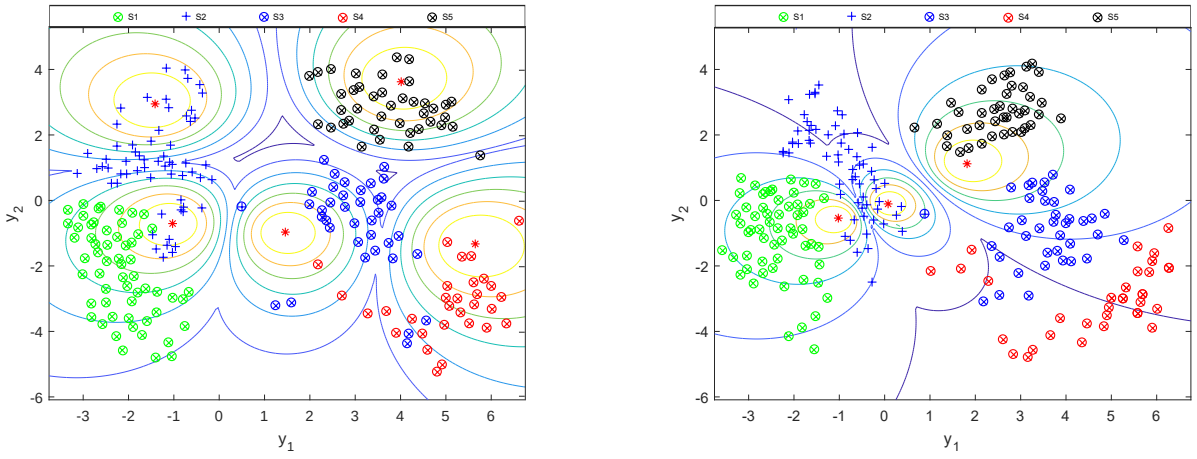


FIGURE 38 – L'évolution du XBI et le nombre correspondant de clusters en fonction du nombre d'itérations pour une exécution typique de l'algorithme sbWFCM.

7.3.2 Résultats du *Clustering* et discussion

L'algorithme sbWFCM proposé est appliqué à l'ensemble de données résultant. Certaines partitions raisonnables ont été révélées. La figure 38 montre la qualité des différentes alternatives structurelles fournies par l'algorithme basé sur le XBI. Dans cette figure, les résultats correspondent à l'évolution du XBI et au nombre correspondant de clusters en fonction du nombre d'itérations pour une exécution typique de l'algorithme sbWFCM. Chaque minimum local de l'indice correspond à un nombre raisonnable de clusters. L'analyse de la figure 38 montre que les partitions avec un nombre de clusters égal à 3 et 5 ($\boldsymbol{C} \in \{3, 5\}$) constituent des alternatives structurelles raisonnables. La figure 39 présente des illustrations des deux différentes perspectives en utilisant la projection *Sammom*, où le rouge $*$ indique le centre de chaque cluster ; dix courbes en trait plein de différentes couleurs allant du rouge au bleu foncé sont tracées autour de celui-ci ; chaque courbe représente un contour de rang d'appartenance similaire, c'est-à-dire que la valeur d'appartenance devient plus faible au fur et à mesure que la courbe s'éloigne du centre (plus le bleu est foncé, plus l'appartenance est faible).

En commençant par la figure 39a, on observe qu'un cluster contenant les symboles verts $\otimes$ $(S1)$, correspondant aux charges de travail qui ont une faible utilisation du CPU et de la mémoire et un faible temps d'arrêt, est formé. De même, le cluster constitué par les symboles noirs $\otimes$ $(S5)$ est clairement formé. On peut dire que le cluster formé repré-



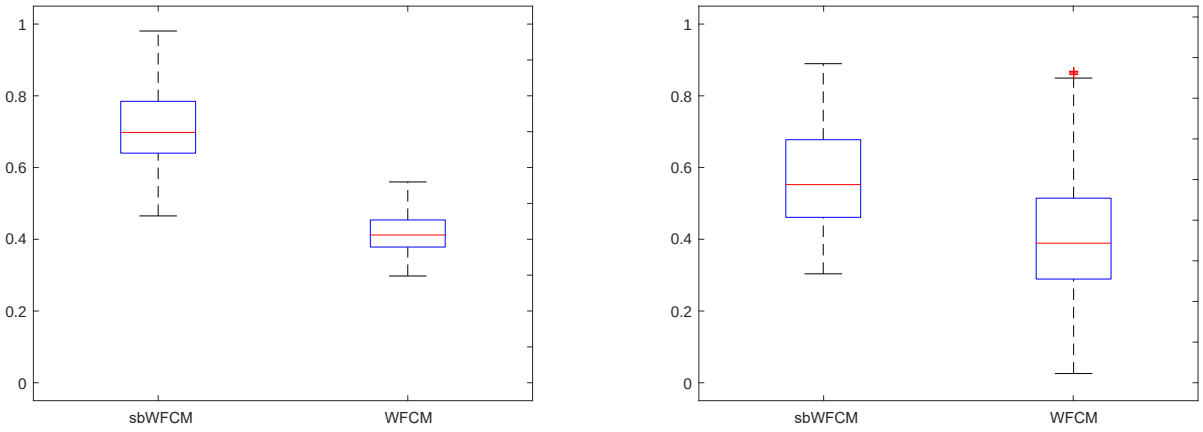
(a) clusters atteints après 700 itérations. (b) clusters atteints après 900 itérations.

FIGURE 39 – Projection *Sammom* de l'espace caractéristique résultant dans le plan pour le *Clustering* des charges de travail en utilisant l'algorithme sbWFCM.

sente des charges de travail susceptibles de provoquer une dégradation des performances ; en revanche, le cluster contenant les symboles verts n'influence pas les performances du système. Par conséquent, leur répartition en deux groupes largement séparés est raisonnable. Par ailleurs, même si les clusters formés par les symboles bleu ⊗ et rouge ⊗ sont en gros deux clusters séparés, on remarque que certains éléments des deux clusters sont mal classés. Cela peut s'expliquer par le fait que les deux types de charges de travail ont un impact élevé sur le temps d'arrêt.

Plus intéressant encore, en augmentant le nombre d'itérations, le nombre de clusters correspondant à un partitionnement raisonnable diminue (de 5 à 3). Passant à la figure 39b, les charges de travail avec une faible utilisation du CPU et de la mémoire mais un temps d'arrêt élevé (+ bleu) sont toujours considérées par l'algorithme comme étant similaires aux charges de travail ayant l'état *S1*. De la même manière, les trois types de charges de travail qui ont un temps d'arrêt élevé (*S3*, *S4* et *S5*) sont fusionnés en un seul cluster avec quelques éléments mal classés. On peut noter que le nombre d'itérations pour la mise à jour de la valeur de pondération attribuée à chaque point de données est important, et peut être ajusté par un expert de domaine.

De plus, nous comparons l'algorithme de *Clustering* proposé sbWFCM avec un simple algorithme WFCM basé sur la densité. Nous procédons comme si nous sommes intéressés par la qualité des résultats finaux. Nous évaluons statistiquement la qualité des partitions obtenues, telles qu'enregistrées par l'ARI externe, pour 20 exécutions des deux algorithmes pour des clusters $C = 3$ et $C = 5$ avec les mêmes paramètres initiaux. Comme le cas présent ne requiert aucune normalité ou homogénéité des échantillons analysés, nous utilisons le test statistique non paramétrique de *Wilcoxon* pour analyser les résultats [226]. En d'autres termes, le test sert à répondre à la question dans quelle mesure la qualité des



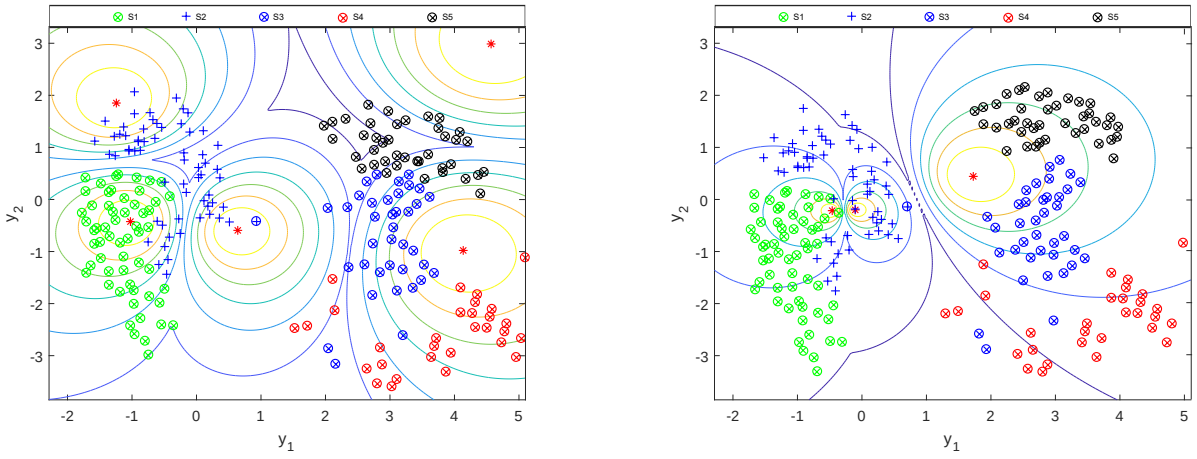
(a) 3 clusters ($C = 3$). (b) 5 clusters ($C = 5$).

FIGURE 40 – *Boxplots* montrant les distributions de l'ARI sur 20 exécutions indépendantes de sbWFCM et de WFCM simples basées sur la densité.

partitions peut refléter deux populations différentes. La distribution observée de l'ARI sur 20 exécutions indépendantes des algorithmes pour un nombre variable de clusters est résumée dans la figure 40 avec les résultats statistiques correspondants atteints. Le niveau de signification considéré est donné par $\alpha = 0,05$ se référant à un intervalle de confiance de 95%. Par exemple, $p - value < 0,05$ indique l'existence d'une différence statistiquement significative entre les échantillons de données analysés. À partir de ces expériences, nous pouvons conclure que le sbWFCM surpasse le WFCM standard basé sur la densité tant pour $C = 3$ que pour $C = 5$. Une telle amélioration est un objectif de ce type d'algorithmes ; c'est un effet pratique de la technique comportementale élémentaire utilisée pour concevoir le processus du *Clustering*. Les améliorations sont constatées pour les deux nombres de clusters considérés. En pratique, cela signifie essentiellement que, dans ce cas, le WFCM simple basé sur la densité n'est pas capable de distinguer clairement les points de données pour les différents états de charge de travail, comme le montre la figure 41 ; nous remarquons que les clusters identifiés par l'algorithme ne sont pas adéquats avec la répartition réelle des données ; par exemple, dans le cluster supérieur droit présenté dans la figure 41a, les points de données désignés par le symbole noir ⊗ sont éloignés du centre du cluster.

7.4 Conclusion

Pour un fonctionnement robuste et efficace, la détection et le diagnostic automatiques des anomalies de performance des centre de données sont devenus de plus en plus importants dans les systèmes actuels en pleine expansion. Dans ce document, nous proposons une nouvelle technique de *Clustering* (sbWFCM) pour la surveillance de la charge de travail et des performances des VMs au sein d'un centre de données. Cet algorithme permet



(a) 3 clusters ($C = 3$). (b) 5 clusters ($C = 5$).

FIGURE 41 – Projection *Sammom* de l'espace caractéristique résultant dans le plan pour le *Clustering* des charges de travail en utilisant l'algorithme standard WFCM.

de déterminer l'appartenance d'un objet à un cluster, en fonction de sa position par rapport à son voisinage, afin de construire des clusters raisonnables, c'est-à-dire des clusters qui correspondent à la structure réelle des données.

L'algorithme proposé sbWFCM s'inspire du mouvement collectif de groupes d'étourneaux. Selon les simulations de *Reynolds*, la direction du mouvement de chaque étourneau dépend des informations fournies par lui-même et par d'autres étourneaux voisins, pendant le vol. Un étourneau réagit à ses voisins et adapte progressivement sa direction de vol pour suivre ses compagnons. La cohérence de la volée d'étourneaux peut donc être assurée en regroupant les oiseaux dans la même direction, tout en maintenant une distance par rapport aux autres oiseaux de la volée ; l'algorithme sbWFCM suit ce mécanisme intuitif pour construire des clusters.

Les résultats expérimentaux inclus ont montré que le sbWFCM est capable de fournir un certain nombre de partitions raisonnables pour des nombres d'itérations prédéfinis. Les partitions raisonnables sont déterminées par un indice de validation interne, dans ce cas le XBI. En outre, un indice de validité externe ARI a été utilisé pour évaluer et comparer l'algorithme sbWFCM avec l'gorithme standard WFCM existant ; les résultats ont montré que, pour les deux cas examinés dans ce travail, le sbWFCM proposé surpasse le WFCM en produisant des partitions avec de meilleurs scores ARI.

8 Conclusion Générale

Cette thèse est située dans le contexte des centres de données, soit des salles dédiées à la mise en œuvre des services informatiques. Plus précisément, nous nous intéressons aux centres de données virtualisés, où les services sont encapsulés au sein d'environnements virtuels (VMs) exécutés à leur tour sur des serveurs.

Ces centres de données sont au cœur du *Cloud Computing*, une tendance du calcul industriel et scientifique, où les infrastructures physiques et l'exécution des services client sont disponibles sous forme de plate-formes, en mettant en œuvre des ressources adaptables qui peuvent être facturées à la demande. Ce modèle est actuellement utilisé par la majorité des grandes entreprises du monde entier, aussi bien pour les plates-formes *Web-server* satisfaisant les requêtes fréquentes et dont l'activité varie en fonction du nombre de clients qui accèdent aux services, que pour les plates-formes de calcul à haute performance (HPC), qui sont destinées à résoudre des problèmes complexes et requérant des machines de grande capacité et des infrastructures hautement performantes.

8.1 Apports de cette thèse

Compte tenu de la grande importance accordée à ce concept de *Cloud Computing*, le nombre de centres de données virtualisés et le nombre de serveurs ne cessent d'augmenter depuis quelques années à travers le monde. La consommation électrique de ces centres a également accru de manière significative. Si le rendement électrique des serveurs a progressé grâce aux évolutions technologiques, l'augmentation du nombre de circuits par serveur, voire la capacité de calcul des serveurs - surtout pour les serveurs blades - a augmenté leurs besoins en énergie.

La majeure partie de la consommation d'énergie des centres de données peut être attribuée aux problèmes de gestion et de consolidation des VMs et des PMs. Il est donc nécessaire de développer des stratégies efficaces pour quantifier l'utilisation des ressources et gérer les problèmes d'affectation et de réaffectation des VMs afin de maintenir avec succès l'efficacité énergétique, l'évolutivité et la durabilité des centres de données virtualisés modernes.

Compte tenu de cette situation, le but de cette thèse est d'aborder le problème de l'utilisation des ressources dans un centre de données, plus précisément en *Infrastructure as a Servic*e (IaaS), qui peut avoir un effet sur la consommation énergétique globale du centre et sur la qualité de service (QoS) rendu à l'utilisateur. Afin de progresser vers cet objectif, cette thèse comporte quatre contributions :

- **Première contribution (chapitre 4)** : nous avons proposé une nouvelle approche pour déterminer la consommation électrique d'un serveur. Nous avons apporté des observations sur le comportement des composants les plus influents du serveur en matière de consommation électrique et nous proposons une technique précise pour estimer la consommation électrique d'un serveur. On a proposé ainsi un modèle stochastique pour estimer la consommation d'énergie d'un serveur afin d'établir une relation statistique entre l'utilisation de la CPU et la consommation d'énergie. Puis, nous avons conçu notre modèle pour considérer un ensemble de serveur (cluster) au lieu d'un seul serveur. Ainsi, nous avons utilisé ce modèle pour tester le modèle de placement des VMs proposé dans cette thèse.
- **Deuxième contribution (chapitre 5)** : une nouvelle approche de placement statique des VMs pour minimiser la consommation totale d'énergie d'un centre de données a été proposée. Le problème de placement des VMs est formulé sous forme de problème de sac à dos qui est considéré comme un problème d'optimisation combinatoire sous contrainte. Après avoir étudier différentes méthodes de résolution, une approche métaheuristique est adoptée pour résoudre le problème formulé. Une nouvelle politique, consistant à maximiser l'utilisation de ressource CPU d'un serveur tout en respectant un seuil de capacité préalablement fixé, a été adoptée pour formuler le problème d'optimisation de placement des VMs. Éventuellement, les résultats de simulation ont prouvé la contribution de notre modèle au niveau de l'optimisation de la consommation énergétique.
- **Troisième contribution (chapitre 6)** : nous avons conçu un modèle de migration adaptatif, basé sur le *Machine Learning*, permettant de prédire avec une grande précision des métriques de performance de la migration en temps réel selon l'algorithme de migration et la charge de travail fonctionnant dans les VMs. Pour cela, nous avons examiné les paramètres en entrée nécessaires à la modélisation précise des métriques cibles, et nous avons expliqué la façon de les modéliser à moindre coût. Par rapport aux travaux existants, nous avons modélisé les principaux algorithmes de migration utilisés et prédit des paramètres importants qui n'ont pas été pris en compte jusqu'à présent, comme la dégradation des performances des VMs.
- **Quatrième contribution (chapitre 7)** : un nouvel algorithme de *Clustering* a été introduit pour analyser le comportement des VMs lors de l'exécution des charges de

travail dans un système en fonction des détails de l'utilisation des ressources (comme l'utilisation de la CPU, des événements d'arrêt, etc). Toutes les informations identifiées par le processus de *Clustering* sont essentielles pour le contrôle et la sécurité du système. Pour concevoir l'algorithme proposé, on a adopté un mécanisme intuitif de regroupement des oiseaux dans la nature pour former des groupes convenables. L'évaluation expérimentale, basée sur un ensemble de données réel, a montré que le nouvel algorithme donne des résultats intéressants.

8.2 Limitations et futures recommandations

Bien que nous ayons démontré des résultats significatifs des solutions que nous avons proposées, il est important de souligner les limites de cette thèse. Nous indiquons un certain nombre de limites notables, qui mettent en évidence des observations qui sont hors de portée dans nos considérations et qui restent pour les recherches futures.

Le modèle d'estimation stochastique fournit dans le chapitre 4 s'est concentré essentiellement sur les variations de la CPU et sa relation de corrélation avec la consommation énergétique. Notre modèle a montré une grande précision, selon les tests de simulation effectués ; cependant, vue la diversité des serveurs au niveau de l'architecture physique nécessite la prise en considération de plusieurs ressources, telles que la mémoire et le disque, pour améliorer la précision du modèle.

Pareillement, dans le problème du placement statique des VMs (chapitre 5), nous ne prenons en compte que la CPU dans la définition des contraintes de QoS. Elle repose sur le fait que la CPU est la principale ressource qui est généralement sursouscrite par les fournisseurs de *Cloud Computing*. Cette hypothèse ne signifie pas que les autres ressources telles que la mémoire, le stockage et la bande passante n'ont aucun impact sur la QoS. Nous pensons que la prise en compte d'autres ressources peut se révéler plus performante dans le processus de placement statique des VMs mais augmente la complexité de la conception du modèle.

Les résultats présentés dans le chapitre 6 de cette thèse montrent que la prédiction de différentes métriques cibles pour une variété d'algorithmes de migration en temps réel en utilisant le *Machine Learning* est possible et mène à de bons résultats dans le cadre de notre simulation expérimentale. Les caractéristiques présentées et l'ensemble de données utilisé peuvent toutefois ne pas être suffisants pour des environnements de centres de données plus complexes avec des nœuds hétérogènes, des réseaux hiérarchiques ou des technologies de virtualisation spéciales. Néanmoins, ce travail démontre que les techniques de *Machine Learning* peuvent être utilisées pour générer des modèles de prédiction semi-automatiques et précis pour un large éventail de caractéristiques. Il se peut que

les caractéristiques d'entrée alimentant les modèles doivent être étendues et que de nouvelles données de migration en temps réel doivent être collectées pour des cas spécifiques, mais nos résultats montrent également que quelques milliers de migrations suffisent pour obtenir des prédictions relativement exactes des métriques cibles.

8.3 Remarque finale

Cette thèse a abordé les quatre problèmes de recherche considérés avec succès et a atteint les objectifs de recherche spécifiés dans la section 1.1. Les frameworks proposés pour la gestion énergiquement efficace des ressources et des VMs dans les centres de données peuvent offrir une solution aux fournisseurs de services en Cloud pour réduire la consommation d'énergie dans les centres de données. Les futures orientations de recherche résultant de cette thèse ouvriront la voie à d'autres développements dans l'innovation future pour la technologie du *Cloud Computing*.

Bibliographie

[1] E. A. Silver, R. Victor, V. Vidal, et D. de Werra. A tutorial on heuristic methods. *European Journal of Operational Research*, 5(3) :153 – 162, 1980. ISSN 0377-2217. doi : https://doi.org/10.1016/0377-2217(80)90084-3.

[2] J. C. Bezdek, R. Ehrlich, et W. Full. Fcm : The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2-3) :191–203, 1984.

[3] H. Drucker, C. J. C. Burges, L. Kaufman, A. Smola, et V. Vapnik. Support vector regression machines. In *Proceedings of the 9th International Conference on Neural Information Processing Systems*, NIPS'96, page 155–161. MIT Press, 1996.

[4] M. Dorigo, V. Maniezzo, et A. Colorni. Ant system : optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1) :29–41, Feb 1996. ISSN 1083-4419. doi : 10.1109/3477.484436.

[5] D. S. Milojičić, F. Douglis, Y. Paindaveine, R. Wheeler, et S. Zhou. Process migration. *ACM Comput. Surv.*, 32(3) :241–299, Sept. 2000. ISSN 0360-0300. doi : 10.1145/367701.367728.

[6] F. Giulietti, L. Pollini, et M. Innocenti. Autonomous formation flight. *IEEE Control Systems Magazine*, 20(6) :34–44, Dec 2000. ISSN 1941-000X. doi : 10.1109/37.887447.

[7] A. Pant, P. Seiler, et K. Hedrick. Mesh stability of look-ahead interconnected systems. *IEEE Transactions on Automatic Control*, 47(2) :403–407, Feb 2002. ISSN 2334-3303. doi : 10.1109/9.983389.

[8] C. P. Sapuntzakis, R. Chandra, B. Pfaff, J. Chow, M. S. Lam, et M. Rosenblum. Optimizing the migration of virtual computers. *SIGOPS Oper. Syst. Rev.*, 36(SI) :377–390, Dec. 2003. ISSN 0163-5980. doi : 10.1145/844128.844163.

[9] H. G. Tanner, S. G. Loizou, et K. J. Kyriakopoulos. Nonholonomic navigation and control of cooperating mobile manipulators. *IEEE Transactions on Robotics and Automation*, 19(1) :53–64, Feb 2003. ISSN 2374-958X. doi : 10.1109/TRA.2002.807549.

[10] P. Ogren, E. Fiorelli, et N. E. Leonard. Cooperative control of mobile sensor networks :adaptive gradient climbing in a distributed environment. *IEEE Transactions on Automatic Control*, 49(8) : 1292–1302, Aug 2004. ISSN 2334-3303. doi : 10.1109/TAC.2004.832203.

[11] A. Ines, S. Christine, et G. Khaled. Ant algorithm for the multidimensional knapsack problem. In *Proceedings of international conference on Bioinspired Methods and their Applications (BIOMA 2004)*, pages 63–72, Slovenia, Slovenia, May 2004.

[12] H. Kellerer, U. Pferschy, et D. Pisinger. *Knapsack Problems*. 2004.

[13] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, et N. Gautam. Managing server energy and operational costs in hosting centers. In *Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '05, page 303–314, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595930221. doi : 10.1145/1064212.1064253.

[14] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, et A. Warfield. Live migration of virtual machines. NSDI'05, page 273–286, USA, 2005. USENIX Association.

[15] Yu-Feng Deng, Xing Jin, et Yi-Xin Zhong. Ensemble svr for prediction of time series. In *2005 International Conference on Machine Learning and Cybernetics*, volume 6, pages 3528–3534 Vol. 6, 2005. doi : 10.1109/ICMLC.2005.1527553.

[16] X. Fan, W.-D. Weber, et L. A. Barroso. Power provisioning for a warehouse-sized computer. *SIGARCH Comput. Archit. News*, 35(2) :13–23, June 2007. ISSN 0163-5964. doi : 10.1145/1273440.1250665.

[17] C. Lefurgy, X. Wang, et M. Ware. Server-level power control. In *Fourth International Conference on Autonomic Computing (ICAC'07)*, pages 4–4, 2007.

[18] J. Xu, M. Zhao, J. Fortes, R. Carpenter, et M. Yousif. On the use of fuzzy modeling in virtualized data center management. In *Fourth International Conference on Autonomic Computing (ICAC'07)*, pages 25–25, 2007.

[19] M. Bichler, T. Setzer, et B. Speitkamp. Capacity planning for virtualized servers. 11 2007.

[20] A. Lewis, S. Ghosh, et N.-F. Tzeng. Run-time energy consumption estimation based on workload in server systems. In *Proceedings of the 2008 Conference on Power Aware Computing and Systems*, HotPower'08, page 4, USA, 2008. USENIX Association.

[21] W. Chen, H. Lu, L. Shen, Z. Wang, N. Xiao, et D. Chen. A novel hardware assisted full virtualization technique. In *2008 The 9th International Conference for Young Computer Scientists*, pages 1292–1297, 2008.

[22] A. Verma, P. Ahuja, et A. Neogi. pmapper : Power and migration cost aware application placement in virtualized systems. In V. Issarny et R. Schantz, editors, *Middleware 2008*, pages 243–264, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[23] L. Bianchi, M. Dorigo, L. M. Gambardella, et W. J. Gutjahr. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing : An International Journal*, 8(2) :239–287, June 2009. ISSN 1567-7818. doi : 10.1007/s11047-008-9098-4.

[24] V. Chandola, A. Banerjee, et V. Kumar. Anomaly detection : A survey. *ACM Comput. Surv.*, 41 (3), 2009. ISSN 0360-0300. doi : 10.1145/1541880.1541882.

[25] H. Jin, L. Deng, S. Wu, X. Shi, et X. Pan. Live virtual machine migration with adaptive, memory compression. In *2009 IEEE International Conference on Cluster Computing and Workshops*, pages 1–10, 2009. doi : 10.1109/CLUSTR.2009.5289170.

[26] A. Quiroz, H. Kim, M. Parashar, N. Gnanasambandam, et N. Sharma. Towards autonomic workload provisioning for enterprise grids and clouds. In *2009 10th IEEE/ACM International Conference on Grid Computing*, pages 50–57, 2009.

[27] H. Li, G. Casale, et T. Ellahi. Sla-driven planning and optimization of enterprise applications. In *Proceedings of the First Joint WOSP/SIPEW International Conference on Performance Engineering*, WOSP/SIPEW '10, page 117–128, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781605585635. doi : 10.1145/1712605.1712625.

[28] A. Kansal, F. Zhao, J. Liu, N. Kothari, et A. A. Bhattacharya. Virtual machine power metering and provisioning. In *Proceedings of the 1st ACM Symposium on Cloud Computing*, SoCC '10, page 39–50, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450300360. doi : 10.1145/1807128.1807136.

[29] G. Dhiman, K. Mihic, et T. Rosing. A system for online power prediction in virtualized environments using gaussian mixture models. In *Proceedings of the 47th Design Automation Conference*, DAC '10, page 807–812, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450300025. doi : 10.1145/1837274.1837478.

[30] S. Zhang, S. Zhang, X. Chen, et X. Huo. Cloud computing research and development trend. In *2010 Second International Conference on Future Networks*, pages 93–97, 2010.

[31] T. V. T. Duy, Y. Sato, et Y. Inoguchi. Performance evaluation of a green scheduling algorithm for energy savings in cloud computing. In *2010 IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW)*, pages 1–8, 2010.

[32] Z. Liu, W. Qu, W. Liu, et K. Li. Xen live migration with slowdown scheduling algorithm. In *2010 International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 215–221, 2010. doi : 10.1109/PDCAT.2010.88.

[33] R. Buyya, J. Broberg, et A. M. Goscinski. *Cloud Computing : Principles and Paradigms*. 2010.

[34] H. Liu, C.-Z. Xu, H. Jin, J. Gong, et X. Liao. Performance and energy modeling for live migration of virtual machines. In *Proceedings of the 20th International Symposium on High Performance Distributed Computing*, page 171–182, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450305525. doi : 10.1145/1996130.1996154.

[35] P. Svärd, B. Hudzia, J. Tordsson, et E. Elmroth. Evaluation of delta compression techniques for efficient live migration of large virtual machines. *SIGPLAN Not.*, 46(7) :111–120, Mar. 2011. ISSN 0362-1340. doi : 10.1145/2007477.1952698.

[36] Z. Shen, S. Subbiah, X. Gu, et J. Wilkes. Cloudscale : Elastic resource scaling for multi-tenant cloud systems. In *Proceedings of the 2nd ACM Symposium on Cloud Computing*, SOCC '11. Association for Computing Machinery, 2011. ISBN 9781450309769. doi : 10.1145/2038916.2038921.

[37] V. Gupta, R. Nathuji, et K. Schwan. An analysis of power reduction in datacenters using heterogeneous chip multiprocessors. *SIGMETRICS Perform. Eval. Rev.*, 39(3) :87–91, 2011. ISSN 0163-5999. doi : 10.1145/2160803.2160867.

[38] R. Basmadjian, N. Ali, F. Niedermeier, H. de Meer, et G. Giuliani. A methodology to predict the power consumption of servers in data centres. In *Proceedings of the 2nd International Conference on Energy-Efficient Computing and Networking*, e-Energy '11, page 1–10, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450313131. doi : 10.1145/2318716.2318718.

[39] S. Lim, B. Sharma, B. C. Tak, et C. R. Das. A dynamic energy management scheme for multi-tier data centers. In *(IEEE ISPASS) IEEE International Symposium on Performance Analysis of Systems and Software*, pages 257–266, 2011.

[40] E. Feller, L. Rilling, et C. Morin. Energy-aware ant colony based workload placement in clouds. In *2011 IEEE/ACM 12th International Conference on Grid Computing*, pages 26–33, 2011.

[41] O. Niehorster, A. Krieger, J. Simon, et A. Brinkmann. Autonomic resource management with support vector machines. In *2011 IEEE/ACM 12th International Conference on Grid Computing*, pages 157–164, 2011.

[42] C. Blum, J. Puchinger, G. R. Raidl, et A. Roli. Hybrid metaheuristics in combinatorial optimization : A survey. *Applied Soft Computing*, 11 :4135 – 4151, 2011. ISSN 1568-4946. doi : https://doi.org/10.1016/j.asoc.2011.02.032.

[43] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, et R. Buyya. Cloudsim : a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software : Practice and Experience*, 41(1) :23–50, 2011. doi : 10.1002/spe.995.

[44] C.-Y. Yeh, C.-W. Huang, et S.-J. Lee. A multiple-kernel support vector regression approach for stock market price forecasting. *Expert Systems with Applications*, 38(3) :2177–2186, 2011.

[45] S. Kundu, R. Rangaswami, A. Gulati, M. Zhao, et K. Dutta. Modeling virtualized applications using machine learning techniques. *SIGPLAN Not.*, 47(7) :3–14, 2012. ISSN 0362-1340. doi : 10.1145/2365864.2151028.

[46] S. T. Maguluri, R. Srikant, et L. Ying. Stochastic models of load balancing and scheduling in cloud computing clusters. In *2012 Proceedings IEEE INFOCOM*, pages 702–710, 2012.

[47] A. Khan, X. Yan, S. Tao, et N. Anerousis. Workload characterization and prediction in the cloud : A multiple time series approach. In *2012 IEEE Network Operations and Management Symposium*, pages 1287–1294, 2012.

[48] K. T. Malladi, F. A. Nothaft, K. Periyathambi, B. C. Lee, C. Kozyrakis, et M. Horowitz. Towards energy-proportional datacenter memory with mobile dram. In *2012 39th Annual International Symposium on Computer Architecture (ISCA)*, pages 37–48, 2012.

[49] S. Reda, R. Cochran, et A. K. Coskun. Adaptive power capping for servers with multithreaded workloads. *IEEE Micro*, 32(5) :64–75, 2012.

[50] M. Mishra, A. Das, P. Kulkarni, et A. Sahoo. Dynamic resource management using virtual machine migrations. *IEEE Communications Magazine*, 50(9) :34–40, 2012.

[51] A. Beloglazov, J. Abawajy, et R. Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, 28(5) :755 – 768, 2012. ISSN 0167-739X. doi : https://doi.org/10.1016/j.future.2011.04.017. Special Section : Energy efficiency in large-scale distributed systems.

[52] Y. Li, D. Zhang, X. Feng, Y. Xu, et B.-F. Liu. A microsecond microfluidic mixer for characterizing fast biochemical reactions. *Talanta*, 88 :175 – 180, 2012. ISSN 0039-9140. doi : https://doi.org/10.1016/j.talanta.2011.10.028.

[53] Y. Yao, L. Huang, A. B. Sharma, L. Golubchik, et M. J. Neely. Data centers power reduction : A two time scale approach for delay tolerant workloads. *2012 Proceedings IEEE INFOCOM*, pages 1431–1439, 2012.

[54] W. Bialek, A. Cavagna, I. Giardina, T. Mora, E. Silvestri, M. Viale, et A. M. Walczak. Statistical mechanics for natural flocks of birds. *Proceedings of the National Academy of Sciences*, 109(13) : 4786–4791, 2012. doi : 10.1073/pnas.1118633109.

[55] M. Baghel, S. Agrawal, et S. Silakari. Survey of metaheuristic algorithms for combinatorial optimization. *International Journal of Computer Applications*, 58(19) :21–31, November 2012.

[56] Y. Gao, H. Guan, Z. Qi, Y. Hou, et L. Liu. A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. *J. Comput. Syst. Sci.*, 79(8) :1230–1242, 2013. ISSN 0022-0000. doi : 10.1016/j.jcss.2013.02.004.

[57] Y. Jin, Y. Wen, Q. Chen, et Z. Zhu. An Empirical Investigation of the Impact of Server Virtualization on Energy Efficiency for Green Data Center. *The Computer Journal*, 56(8) :977–990, 02 2013. ISSN 0010-4620. doi : 10.1093/comjnl/bxt017.

[58] C. Jo, E. Gustafsson, J. Son, et B. Egger. Efficient live migration of virtual machines using shared storage. In *Proceedings of the 9th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, VEE '13, page 41–50. Association for Computing Machinery, 2013. ISBN 9781450312660. doi : 10.1145/2451512.2451524.

[59] X. Zhang, E. Tune, R. Hagmann, R. Jnagal, V. Gokhale, et J. Wilkes. Cpi2 : Cpu performance isolation for shared compute clusters. EuroSys '13, page 379–391. Association for Computing Machinery, 2013. ISBN 9781450319942. doi : 10.1145/2465351.2465388.

[60] D. Xu, S. Yang, et H. Luo. A fusion model for cpu load prediction in cloud computing. *Journal of Networks*, 8, 10 2013. doi : 10.4304/jnw.8.11.2506-2511.

[61] J. Dong, X. Jin, H. Wang, Y. Li, P. Zhang, et S. Cheng. Energy-saving virtual machine placement in cloud data centers. In *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, pages 618–624, 2013.

[62] F. Farahnakian, P. Liljeberg, et J. Plosila. Lircup : Linear regression based cpu usage prediction algorithm for live migration of virtual machines in data centers. In *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*, pages 357–364, 2013.

[63] M. Sun, W. Gu, X. Zhang, H. Shi, et W. Zhang. A matrix transformation algorithm for virtual machine placement in cloud. In *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pages 1778–1783, 2013.

[64] Q. Zhu, T. Tung, et Q. Xie. Automatic fault diagnosis in cloud infrastructure. In *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, volume 1, pages 467–474, 2013. doi : 10.1109/CloudCom.2013.68.

[65] F. Farahnakian, T. Pahikkala, P. Liljeberg, et J. Plosila. Energy aware consolidation algorithm based on k-nearest neighbor regression for cloud data centers. In *2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*, pages 256–259, 2013.

[66] S. Frey, C. Reich, et C. Lüthje. Key performance indicators for cloud computing slas. 2013.

[67] X. Li, Z. Qian, S. Lu, et J. Wu. Energy efficient virtual machine placement algorithm with balanced and improved resource utilization in a data center. *Mathematical and Computer Modelling*, 58 (5) :1222 – 1235, 2013. ISSN 0895-7177. doi : https://doi.org/10.1016/j.mcm.2013.02.003. The Measurement of Undesirable Outputs : Models Development and Empirical Analyses and Advances in mobile, ubiquitous and cognitive computing.

[68] M. Witkowski, A. Oleksiak, T. Piontek, et J. Węglarz. Practical power consumption estimation for real life hpc applications. *Future Generation Computer Systems*, 29(1) :208 – 217, 2013. ISSN 0167-739X. doi : https://doi.org/10.1016/j.future.2012.06.003. Including Special section : AIRCC-NetCoM 2009 and Special section : Clouds and Service-Oriented Architectures.

[69] P. Xiao, Z. Hu, D. Liu, G. Yan, et X. Qu. Virtual machine power measuring technique with bounded error in cloud environments. *Journal of Network and Computer Applications*, 36(2) :818 – 828, 2013. ISSN 1084-8045. doi : https://doi.org/10.1016/j.jnca.2012.12.002.

[70] X. Zhang, J.-J. Lu, X. Qin, et X.-N. Zhao. A high-level energy consumption model for heterogeneous data centers. *Simulation Modelling Practice and Theory*, 39 :41 – 55, 2013. ISSN 1569-190X. doi : https://doi.org/10.1016/j.simpat.2013.05.006. S.I.Energy efficiency in grids and clouds.

[71] L. Zhang, Y. Zhuang, et W. Zhu. Constraint programming based virtual cloud resources allocation model. *International Journal of Hybrid Information Technology*, 6 :333–344, 2013.

[72] C. Changdar, G. Mahapatra, et R. Pal. An ant colony optimization approach for binary knapsack problem under fuzziness. *Applied Mathematics and Computation*, 223 :243 – 253, 2013. ISSN 0096-3003. doi : https://doi.org/10.1016/j.amc.2013.07.077.

[73] B. Anne, R.Yves, et V. Frédéric. *A Guide to Algorithm Design : Paradigms, Methods, and Complexity Analysis*. Applied Algorithms and Data Structures series. Chapman & Hall/CRC, Aug. 2013. URL https://hal.inria.fr/hal-00908448.

[74] C. Li, Y. Hu, R. Zhou, M. Liu, L. Liu, J. Yuan, et T. Li. Enabling datacenter servers to scale out economically and sustainably. In *2013 46th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 322–333, 2013.

[75] A. Auweter, A. Bode, M. Brehm, L. Brochard, N. Hammer, H. Huber, R. Panda, F. Thomas, et T. Wilde. A case study of energy aware scheduling on supermuc. In J. M. Kunkel, T. Ludwig, et H. W. Meuer, editors, *Supercomputing*, pages 394–409, Cham, 2014. Springer International Publishing. ISBN 978-3-319-07518-1.

[76] M. H. Ferdaus, M. Murshed, R. N. Calheiros, et R. Buyya. Virtual machine consolidation in cloud data centers using aco metaheuristic. In F. Silva, I. Dutra, et V. Santos Costa, editors, *Euro-Par 2014 Parallel Processing*, pages 306–317, Cham, 2014. Springer International Publishing. ISBN 978-3-319-09873-9.

[77] Y. Tian, C. Lin, et K. Li. Managing performance and power consumption tradeoff for multiple heterogeneous servers in cloud computing. *Cluster Computing*, 17(3) :943–955, 2014. ISSN 1386-7857. doi : 10.1007/s10586-013-0326-z.

[78] S. Nathan, P. Kulkarni, et U. Bellur. Resource availability based performance benchmarking of virtual machine migrations. In *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*, page 387–398, 2014. ISBN 9781450316361. doi : 10.1145/2479871.2479932.

[79] X.-F. Liu, Z.-H. Zhan, K.-J. Du, et W.-N. Chen. Energy aware virtual machine placement scheduling in cloud computing based on ant colony optimization approach. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, GECCO '14, page 41–48, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450326629. doi : 10.1145/2576768.2598265.

[80] M. Snir, R. W. Wisniewski, J. A. Abraham, S. V. Adve, S. Bagchi, P. Balaji, J. Belak, P. Bose, F. Cappello, B. Carlson, A. A. Chien, P. Coteus, N. A. Debardeleben, P. C. Diniz, C. Engelmann, M. Erez, S. Fazzari, A. Geist, R. Gupta, F. Johnson, S. Krishnamoorthy, S. Leyffer, D. Liberty, S. Mitra, T. Munson, R. Schreiber, J. Stearley, et E. V. Hensbergen. Addressing failures in exascale computing. 28(2), 2014. ISSN 1094-3420. doi : 10.1177/1094342014522573.

[81] I. Alan, E. Arslan, et T. Kosar. Energy-aware data transfer tuning. In *2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 626–634, 2014.

[82] Z. J. Estrada, Z. Stephens, C. Pham, Z. Kalbarczyk, et R. K. Iyer. A performance evaluation of sequence alignment software in virtualized environments. In *2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 730–737, 2014.

[83] J. Huang, K. Wu, et M. Moh. Dynamic virtual machine migration algorithms using enhanced energy consumption model for green cloud data centers. In *2014 International Conference on High Performance Computing Simulation (HPCS)*, pages 902–910, 2014.

[84] M. Peiris, J. H. Hill, J. Thelin, S. Bykov, G. Kliot, et C. Konig. Pad : Performance anomaly detection in multi-server distributed systems. In *2014 IEEE 7th International Conference on Cloud Computing*, pages 769–776, 2014. doi : 10.1109/CLOUD.2014.107.

[85] E. B. Lakew, C. Klein, F. Hernandez-Rodriguez, et E. Elmroth. Towards faster response time models for vertical elasticity. In *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, pages 560–565, 2014.

[86] K. Bilal, S. U. R. Malik, O. Khalid, A. Hameed, E. Alvarez, V. Wijaysekara, R. Irfan, S. Shrestha, D. Dwivedy, M. Ali, U. Shahid Khan, A. Abbas, N. Jalil, et S. U. Khan. A taxonomy and survey on green data center networks. *Future Generation Computer Systems*, 36 :189 – 208, 2014. ISSN 0167-739X. doi : https://doi.org/10.1016/j.future.2013.07.006.

[87] W. Bialek, A. Cavagna, I. Giardina, T. Mora, O. Pohl, E. Silvestri, M. Viale, et A. M. Walczak. Social interactions dominate speed control in poising natural flocks near criticality. *Proceedings of the National Academy of Sciences*, 111(20) :7212–7217, 2014. doi : 10.1073/pnas.1324045111.

[88] C. Möbius, W. Dargie, et A. Schill. Power consumption estimation models for processors, virtual machines, and servers. *IEEE Transactions on Parallel and Distributed Systems*, 25(6) :1600–1614, 2014.

[89] K. Rybina, W. Dargie, S. Umashankar, et A. Schill. Modelling the live migration time of virtual machines. In C. Debruyne, H. Panetto, R. Meersman, T. Dillon, G. Weichhart, Y. An, et C. A. Ardagna, editors, *On the Move to Meaningful Internet Systems : OTM 2015 Conferences*, pages 575–593, Cham, 2015. Springer International Publishing. ISBN 978-3-319-26148-5.

[90] X. Lin, Z. Liu, et W. Guo. Energy-efficient vm placement algorithms for cloud data center. In *Revised Selected Papers of the Second International Conference on Cloud Computing and Big Data - Volume 9106*, CloudCom-Asia 2015, page 42–54, Berlin, Heidelberg, 2015. Springer-Verlag. ISBN 9783319284293. doi : 10.1007/978-3-319-28430-9_4.

[91] V. Nair, A. Raul, S. Khanduja, V. Bahirwani, Q. Shao, S. Sellamanickam, S. Keerthi, S. Herbert, et S. Dhulipalla. Learning a hierarchical monitoring system for detecting and diagnosing service issues. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, page 2029–2038, 2015. doi : 10.1145/2783258.2788624.

[92] Z.-H. Zhan, X.-F. Liu, Y.-J. Gong, J. Zhang, H. S.-H. Chung, et Y. Li. Cloud computing resource scheduling and a survey of its evolutionary approaches. *ACM Comput. Surv.*, 47(4), 2015. ISSN 0360-0300.

[93] O. Ibidunmoye, F. Hernández-Rodriguez, et E. Elmroth. Performance anomaly detection and bottleneck identification. *ACM Comput. Surv.*, 48(1), July 2015. ISSN 0360-0300. doi : 10.1145/2791120.

[94] Z. Liu, M. Lin, A. Wierman, S. Low, et L. L. H. Andrew. Greening geographical load balancing. *IEEE/ACM Transactions on Networking*, 23(2) :657–671, 2015. doi : 10.1109/TNET.2014.2308295.

[95] K. Ye, Z. Wu, C. Wang, B. B. Zhou, W. Si, X. Jiang, et A. Y. Zomaya. Profiling-based workload consolidation and migration in virtualized data centers. *IEEE Transactions on Parallel and Distributed Systems*, 26(3) :878–890, 2015.

[96] F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, et H. Tenhunen. Utilization prediction aware vm consolidation approach for green cloud computing. In *2015 IEEE 8th International Conference on Cloud Computing*, pages 381–388, 2015.

[97] Z. Yang, L. Liu, C. Qiao, S. Das, R. Ramesh, et A. Y. Du. Availability-aware energy-efficient virtual machine placement. In *2015 IEEE International Conference on Communications (ICC)*, pages 5853–5858, 2015.

[98] K. Gai, Z. Du, M. Qiu, et H. Zhao. Efficiency-aware workload optimizations of heterogeneous cloud computing for capacity planning in financial industry. In *2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing*, pages 1–6, 2015.

[99] Hui Zhao, Q. Zheng, Weizhan Zhang, Y. Chen, et Yunhui Huang. Virtual machine placement based on the vm performance models in cloud. In *2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC)*, pages 1–8, 2015.

[100] M. Forsman, A. Glad, L. Lundberg, et D. Ilie. Algorithms for automated live migration of virtual machines. *Journal of Systems and Software*, 101 :110 – 126, 2015. ISSN 0164-1212. doi : https://doi.org/10.1016/j.jss.2014.11.044.

[101] S.-W. Ham, M.-H. Kim, B.-N. Choi, et J.-W. Jeong. Simplified server model to simulate data center cooling energy consumption. *Energy and Buildings*, 86 :328 – 339, 2015. ISSN 0378-7788. doi : https://doi.org/10.1016/j.enbuild.2014.10.058.

[102] H. Kataoka, D. Duolikun, T. Enokido, et M. Takizawa. Power consumption and computation models of a server with a multi-core cpu and experiments. In *2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops*, pages 217–222. IEEE, 2015.

[103] S. Nathan, U. Bellur, et P. Kulkarni. Towards a comprehensive performance model of virtual machine live migration. In *Proceedings of the Sixth ACM Symposium on Cloud Computing*, page 288–301. Association for Computing Machinery, 2016. ISBN 9781450336512. doi : 10.1145/2806777.2806838.

[104] G. Peter, A.-A. Yaser, S. Jon, H. Thompson, N. Kapur, et K. Djemame. A unified model for holistic power usage in cloud datacenter servers. In *Proceedings of the 9th International Conference on Utility and Cloud Computing*, UCC '16, page 11–19, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450346160. doi : 10.1145/2996890.2996896.

[105] Z. Zhang, C. Hsu, et M. Chang. Cool cloud : A practical dynamic virtual machine placement framework for energy aware data centers. In *2015 IEEE 8th International Conference on Cloud Computing*, pages 758–765, 2016.

[106] S. Ismaeel, A. Miri, et A. Al-Khazraji. Energy-consumption clustering in cloud data centre. In *2016 3rd MEC International Conference on Big Data and Smart City (ICBDSC)*, pages 1–6, 2016.

[107] B. L. Dalmazo, J. P. Vilela, P. Simoes, et M. Curado. Expedite feature extraction for enhanced cloud anomaly detection. In *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, pages 1215–1220, 2016. doi : 10.1109/NOMS.2016.7502990.

[108] X. Zhang, F. Meng, P. Chen, et J. Xu. Taskinsight : A fine-grained performance anomaly detection and problem locating system. In *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, pages 917–920, 2016. doi : 10.1109/CLOUD.2016.0136.

[109] A. Gulenko, M. Wallschläger, F. Schmidt, O. Kao, et F. Liu. Evaluating machine learning algorithms for anomaly detection in clouds. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 2716–2721, 2016. doi : 10.1109/BigData.2016.7840917.

[110] M. Duggan, K. Flesk, J. Duggan, E. Howley, et E. Barrett. A reinforcement learning approach for dynamic selection of virtual machines in cloud data centres. In *2016 Sixth International Conference on Innovative Computing Technology (INTECH)*, pages 92–97, 2016.

[111] M. Patel, S. Chaudhary, et S. Garg. Machine learning based statistical prediction model for improving performance of live virtual machine migration. *The Journal of Engineering*, 2016 :1–9, 2016.

[112] A. Shehabi, S. J. Smith, D. Sartor, R. E. Brown, M. Herrlin, J. G. Koomey, E. R. Masanet, H. Nathaniel, I. L. Azevedo, et L. William. United states data center energy usage report. Technical report, 06/2016 2016.

[113] I. Shachar, S. Rohit, S. Armando, Y. Kuat, L. Yongquan, L. Charles, et C. Rezaul. Deriving divide-and-conquer dynamic programming algorithms using solver-aided transformations. *SIGPLAN Not.*, 51 :145–164, Oct. 2016. doi : 10.1145/3022671.2983993.

[114] W. Wu, W. Lin, et Z. Peng. An intelligent power consumption model for virtual machines under cpu-intensive workload in cloud environment. *Soft Comput.*, 21(19) :5755–5764, 2017. ISSN 1432-7643. doi : 10.1007/s00500-016-2154-6.

[115] W. Zhu, Y. Zhuang, et L. Zhang. A three-dimensional virtual resource scheduling method for energy saving in cloud computing. 69(C) :66–74, 2017. ISSN 0167-739X. doi : 10.1016/j.future.2016.10.034.

[116] D. Basu, X. Wang, Y. Hong, H. Chen, et S. Bressan. Learn-as-you-go with megh : Efficient live migration of virtual machines. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 2608–2609, 2017.

[117] S. B. Melhem, A. Agarwal, N. Goel, et M. Zaman. A markov-based prediction model for host load detection in live vm migration. In *2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud)*, pages 32–38, 2017.

[118] R. Shaw, E. Howley, et E. Barrett. An advanced reinforcement learning approach for energy-aware virtual machine consolidation in cloud data centers. In *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 61–66, 2017.

[119] M. Duggan, K. Mason, J. Duggan, E. Howley, et E. Barrett. Predicting host cpu utilization in cloud computing using recurrent neural networks. In *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 67–72, 2017.

[120] L. Yin, J. Luo, S. Zhang, et Z. Yang. Virtual machine migration scheme based on score matrix in data centers. In *2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)*, pages 737–743, 2017.

[121] A. Choudhary, M. C. Govil, G. Singh, L. K. Awasthi, E. Pilli, et D. Kapil. A critical survey of live virtual machine migration techniques. *Journal of Cloud Computing*, 6 :1–41, 2017.

[122] M. Duggan, J. Duggan, E. Howley, et E. Barrett. A reinforcement learning approach for the scheduling of live migration from under utilised hosts. *Memetic Computing*, 9 :283–293, 2017.

[123] T. Shabeera, S. Madhu Kumar, S. M. Salam, et K. Murali Krishnan. Optimizing vm allocation and data placement for data-intensive applications in cloud using aco metaheuristic algorithm. *Engineering Science and Technology, an International Journal*, 20(2) :616 – 628, 2017. ISSN 2215-0986. doi : https://doi.org/10.1016/j.jestch.2016.11.006.

[124] R. N. Calheiros, K. Ramamohanarao, R. Buyya, C. Leckie, et S. Versteeg. On the effectiveness of isolation-based anomaly detection in cloud data centers. *Concurrency and Computation : Practice and Experience*, 29(18) :e4169, 2017.

[125] X.-F. Liu, Z.-H. Zhan, et J. Zhang. An energy aware unified ant colony system for dynamic virtual machine placement in cloud computing. *Energies*, 10(5), 2017. ISSN 1996-1073. doi : 10.3390/en10050609.

[126] Z. Li, C. Yan, L. Yu, et X. Yu. Energy-aware and multi-resource overload probability constraint-based virtual machine dynamic consolidation method. *Future Gener. Comput. Syst.*, 80 :139–156, 2018. ISSN 0167-739X. doi : 10.1016/j.future.2017.09.075.

[127] M. Patel, S. Chaudhary, et S. Garg. Improved pre-copy algorithm using statistical prediction and compression model for efficient live memory migration. *Int. J. High Perform. Comput. Netw.*, 11 (1) :55–65, 2018. ISSN 1740-0562. doi : 10.1504/IJHPCN.2018.088879.

[128] K. Gai, M. Qiu, H. Zhao, et X. Sun. Resource management in sustainable cyber-physical systems using heterogeneous cloud computing. *IEEE Transactions on Sustainable Computing*, 3(2) :60–72, 2018.

[129] S. Sotiriadis, N. Bessis, et R. Buyya. Self managed virtual machine scheduling in cloud systems. *Information Sciences*, 433-434 :381–400, 2018. ISSN 0020-0255. doi : 10.1016/j.ins.2017.07.006.

[130] X. Sun, K. Zhang, M. Ma, et H. Su. Multi-population ant colony algorithm for virtual machine deployment. *IEEE Access*, 5 :27014–27022, 2018.

[131] S. B. Melhem, A. Agarwal, N. Goel, et M. Zaman. Markov prediction model for host load detection and vm placement in live migration. *IEEE Access*, 6 :7190–7205, 2018.

[132] P. Zhang, S. Shu, et M. Zhou. An online fault detection model and strategies based on svm-grid in clouds. *IEEE/CAA Journal of Automatica Sinica*, 5(2) :445–456, 2018. doi : 10.1109/JAS.2017.7510817.

[133] M. Farshchi, J.-G. Schneider, I. Weber, et J. Grundy. Metric selection and anomaly detection for cloud operations using log and metric correlation analysis. *Journal of Systems and Software*, 137 : 531 – 549, 2018. ISSN 0164-1212. doi : https://doi.org/10.1016/j.jss.2017.03.012.

[134] W. Lin, W. Wang, W. Wu, X. Pang, B. Liu, et Y. Zhang. A heuristic task scheduling algorithm based on server power efficiency model in cloud environments. *Sustainable Computing : Informatics and Systems*, 20 :56 – 65, 2018. ISSN 2210-5379. doi : https://doi.org/10.1016/j.suscom.2017.10.007.

[135] X. Liu, Z. hui Zhan, J. Deng, Y. Li, T. Gu, et J. Zhang. An energy efficient ant colony system for virtual machine placement in cloud computing. *IEEE Transactions on Evolutionary Computation*, 22 :113–128, 2018.

[136] M.-H. Malekloo, N. Kara, et M. El Barachi. An energy efficient and sla compliant approach for resource allocation and consolidation in cloud computing environments. *Sustainable Computing : Informatics and Systems*, 17 :9 – 24, 2018. ISSN 2210-5379. doi : https://doi.org/10.1016/j.suscom.2018.02.001.

[137] K. Mason, M. Duggan, E. Barrett, J. Duggan, et E. Howley. Predicting host cpu utilization in the cloud using evolutionary neural networks. *Future Generation Computer Systems*, 86 :162 – 173, 2018. ISSN 0167-739X. doi : https://doi.org/10.1016/j.future.2018.03.040.

[138] Y. Bai, L. Gu, et X. Qi. Comparative study of energy performance between chip and inlet temperature-aware workload allocation in air-cooled data center. *Energies*, 11(3) :669, 2018.

[139] B. Luiz André, H. Urs, et R. CParthasarathy. *Designing Warehouse-Scale Machines, Third Edition*. Morgan & ClayPool, 2018. ISBN 9781118981122. doi : https://doi.org/10.2200/S00874ED3V01Y201809CAC046.

[140] W. J. Kavanagh, A. Miller, G. Norman, et O. Andrei. Balancing turn-based games with chained strategy generation. *IEEE Transactions on Games*, pages 1–1, 2019.

[141] F. Alharbi, Y.-C. Tian, M. Tang, W.-Z. Zhang, C. Peng, et M. Fei. An ant colony system for energy-efficient dynamic virtual machine placement in data centers. *Expert Systems with Applications*, 120 : 228 – 238, 2019. ISSN 0957-4174. doi : https://doi.org/10.1016/j.eswa.2018.11.029.

[142] E. M. Saloua, Y. Ali, G. Hamid, et S. Jalal. Comparative study between exact and metaheuristic approaches for virtual machine placement process as knapsack problem. *The Journal of Supercomputing*, 75 :6239 – 6259, 2019. ISSN 1573-0484. doi : 10.1007/s11227-019-02847-0.

[143] M. Rezaei-Mayahi, M. Rezazad, et H. Sarbazi-Azad. Temperature-aware power consumption modeling in hyperscale cloud data centers. *Future Generation Computer Systems*, 94 :130 – 139, 2019. ISSN 0167-739X. doi : https://doi.org/10.1016/j.future.2018.11.029.

[144] M. Duggan, R. Shaw, J. Duggan, E. Howley, et E. Barrett. A multitime-steps-ahead prediction approach for scheduling live migration in cloud data centers. *Software : Practice and Experience*, 49(4) :617–639, 2019. doi : 10.1002/spe.2635.

[145] S. El Motaki, A. Yahyaouy, H. Gualous, et J. Sabor. Comparative study between exact and metaheuristic approaches for virtual machine placement process as knapsack problem. *Journal of Supercomputing*, Apr. 2019. doi : 10.1007/s11227-019-02847-0.

[146] W. Lin, G. Wu, X. Wang, et K. Li. An artificial neural network approach to power consumption model construction for servers in cloud data centers. *IEEE Transactions on Sustainable Computing*, 2019.

[147] T. Mayooran, R. Azizur, et G. Mathew. Recent developments in data science : Comparing linear, ridge and lasso regressions techniques using wine data. 2019.

[148] T. L. Vasques, P. Moura, et A. de Almeida. A review on energy efficiency and demand response with focus on small and medium data centers. *Energy Efficiency*, 12(5) :1399–1428, June 2019. ISSN 1570-6478. doi : 10.1007/s12053-018-9753-2.

[149] Y. Wang, D. Nörtershäuser, S. Le Masson, et J.-M. Menaud. An Empirical Study of Power Characterization Approaches for Servers. In *ENERGY 2019 - The Ninth International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies*, pages 1–6, Athens, Greece, June 2019. URL https://hal.inria.fr/hal-02120589.

[150] Z. Ding, Y. Tian, M. Tang, Y. Li, Y. Wang, et C. Zhou. Profile-guided three-phase virtual resource management for energy efficiency of data centers. *IEEE Transactions on Industrial Electronics*, 67 (3) :2460–2468, 2020.

[151] V. Barthwal, M. Rauthan, et R. Varma. A survey on application of machine learning to manage the virtual machines in cloud computing. *International Review of Applied Sciences and Engineering IRASE*, 2020. doi : https://doi.org/10.1556/1848.2020.00065.

[152] A. E. Hoerl et C. Hoerl. Application of ridge analysis to regression problems. 1962.

[153] D. W. Marquardt et R. D. Snee. Ridge regression in practice. *The American Statistician*, 29(1) : 3–20, 1975. doi : 10.1080/00031305.1975.10479105.

[154] M. Silvano et T. Paolo. A mixture of dynamic programming and branch-and-bound for the subset-sum problem. *Manage. Sci.*, 30 :765–771, June 1984. doi : 10.1287/mnsc.30.6.765.

[155] L. Hubert et P. Arabie. Comparing partitions. *Journal of Classification*, 2(1) :193–218, Dec 1985. ISSN 1432-1343. doi : 10.1007/BF01908075.

[156] K. Dudzinski et S. Walukiewicz. Exact methods for the knapsack problem and its generalizations. *European Journal of Operational Research*, 28 :3–21, 1987. ISSN 0377-2217. doi : https://doi.org/10.1016/0377-2217(87)90165-2.

[157] E. Fix et J. L. Hodges. Discriminatory analysis. nonparametric discrimination : Consistency properties. *International Statistical Review / Revue Internationale de Statistique*, 57(3) :238–247, 1989.

[158] A. Colin Cameron et F. A. Windmeijer. An r-squared measure of goodness of fit for some common nonlinear regression models. *Journal of Econometrics*, 77(2) :329 – 342, 1997. ISSN 0304-4076. doi : https://doi.org/10.1016/S0304-4076(96)01818-0.

[159] T. Balch et R. C. Arkin. Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation*, 14(6) :926–939, Dec 1998. ISSN 2374-958X. doi : 10.1109/70.736776.

[160] R. S. Sutton et A. G. Barto. *Reinforcement Learning*. 1998.

[161] G. Leguizamon et Z. Michalewicz. A new version of ant system for subset problems. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 2, pages 1459–1464, July 1999. doi : 10.1109/CEC.1999.782655.

[162] P. M. Chen et B. D. Noble. When virtual is better than real. HOTOS '01, page 133, USA, 2001. IEEE Computer Society.

[163] C. Blum et A. Roli. Metaheuristics in combinatorial optimization : Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3) :268–308, Sept. 2003. ISSN 0360-0300. doi : 10.1145/937503.937505.

[164] A. Smola et B. Scholkopf. A tutorial on support vector regression, neuro-colt technical report. Technical report, TR-98-030, 2003.

[165] K. Adams et O. Agesen. A comparison of software and hardware techniques for x86 virtualization. page 2–13. Association for Computing Machinery, 2006. ISBN 1595934510. doi : 10.1145/1168857.1168860.

[166] J. V. De Oliveira et W. Pedrycz. *Advances in fuzzy clustering and its applications*. John Wiley & Sons, 2007.

[167] T. Horvath et K. Skadron. Multi-mode energy management for multi-tier server clusters. In *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*, PACT '08, page 270–279, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605582825. doi : 10.1145/1454115.1454153.

[168] M. R. Hines et K. Gopalan. Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning. In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, page 51–60, 2009. ISBN 9781605583754. doi : 10.1145/1508293.1508301.

[169] R. J. Hathaway et Y. Hu. Density-weighted fuzzy c-means clustering. *IEEE Transactions on Fuzzy Systems*, 17(1) :243–252, Feb 2009. ISSN 1941-0034. doi : 10.1109/TFUZZ.2008.2009458.

[170] B. Speitkamp et M. Bichler. A mathematical programming approach for server consolidation problems in virtualized data centers. *IEEE Transactions on Services Computing*, 3(4) :266–278, 2010.

[171] A. E. Husain Bohra et V. Chaudhary. Vmeter : Power modelling for virtualized clouds. In *2010 IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW)*, pages 1–8, 2010.

[172] G. Da Costa et H. Hlavacs. Methodology of measurement for energy consumption of applications. In *2010 11th IEEE/ACM International Conference on Grid Computing*, pages 290–297, 2010.

[173] F. Teng et F. Magoulès. A new game theoretical resource allocation algorithm for cloud computing. In *Proceedings of the 5th International Conference on Advances in Grid and Pervasive Computing*, GPC'10, page 321–330, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3642130666. doi : 10.1007/978-3-642-13067-0_35.

[174] M. Dorigo et T. Stützle. *Ant Colony Optimization : Overview and Recent Advances*, pages 227–263. Springer US, 2010. ISBN 978-1-4419-1665-5. doi : 10.1007/978-1-4419-1665-5_8.

[175] A. Uwe et C. Alistair. Heuristic optimisation. *Journal of the Operational Research Society*, 62 : 251–252, Feb 2011. doi : 10.1057/jors.2010.160.

[176] P. M. Mell et T. Grance. Sp 800-145. the nist definition of cloud computing. Technical report, Gaithersburg, MD, USA, 2011.

[177] A. Beloglazov et R. Buyya. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurrency and Computation : Practice and Experience*, 24(13) :1397–1420, 2012. doi : 10.1002/cpe.1867.

[178] H. Goudarzi et M. Pedram. Energy-efficient virtual machine replication and placement in a cloud computing system. In *2012 IEEE Fifth International Conference on Cloud Computing*, pages 750–757, 2012.

[179] S. ul Islam et J.-M. Pierson. Evaluating energy consumption in cdn servers. In A. Auweter, D. Kranzlmüller, A. Tahamtan, et A. M. Tjoa, editors, *ICT as Key Technology against Global Warming*, pages 64–78, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-32606-6.

[180] T. Enokido et M. Takizawa. An extended power consumption model for distributed applications. In *2012 IEEE 26th International Conference on Advanced Information Networking and Applications*, pages 912–919, 2012.

[181] C. Delimitrou et C. Kozyrakis. Qos-aware scheduling in heterogeneous datacenters with paragon. *ACM Trans. Comput. Syst.*, 31(4), Dec. 2013. ISSN 0734-2071. doi : 10.1145/2556583.

[182] J. Gao et R. Jamidar. Machine learning applications for data center optimization. 2014.

[183] A. Ahmed et A. S. Sabyasachi. Cloud computing simulators : A detailed survey and future direction. In *2014 IEEE International Advance Computing Conference (IACC)*, pages 866–872, 2014.

[184] H. Malik et E. M. Shakshuki. Classification of post-deployment performance diagnostic techniques for large-scale software systems. *Procedia Computer Science*, 37 :244 – 251, 2014. ISSN 1877-0509. doi : https://doi.org/10.1016/j.procs.2014.08.036.

[185] V. Perumal et S. Subbiah. Power-conservative server consolidation based resource management in cloud. *International Journal of Network Management*, 24(6) :415–432, 2014. doi : 10.1002/nem.1873.

[186] F. L. Pires et B. Barán. A virtual machine placement taxonomy. In *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 159–168, 2015.

[187] A. S. G. Andrae et T. Edler. On global electricity usage of communication technology : Trends to 2030. *Challenges*, 6 :117–157, 2015. doi : https://doi.org/10.3390/challe6010117.

[188] M. Tang et S. Pan. A hybrid genetic algorithm for the energy-efficient virtual machine placement problem in data centers. *Neural Process. Lett.*, 41(2) :211–221, 2015. ISSN 1370-4621. doi : 10.1007/s11063-014-9339-8.

[189] S. B. Shaw et A. K. Singh. Use of proactive and reactive hotspot detection technique to reduce the number of virtual machine migration and energy consumption in cloud data center. *Computers Electrical Engineering*, 47 :241 – 254, 2015. ISSN 0045-7906. doi : https://doi.org/10.1016/j.compeleceng.2015.07.020.

[190] T. Enokido et M. Takizawa. Power consumption model of a server to perform communication type application processes on virtual machines. In *2015 10th International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA)*, pages 275–282. IEEE, 2015.

[191] A. Nasrin et O. Mohamed. Energy aware resource allocation of cloud data center : Review and open issues. 19(3) :1163–1182, 2016. ISSN 1386-7857. doi : 10.1007/s10586-016-0579-4.

[192] A. Varasteh et M. Goudarzi. Server consolidation techniques in virtualized data centers : A survey. *IEEE Systems Journal*, 11(2) :772–783, 2017.

[193] M. Rahman et P. Graham. Compatibility-based static vm placement minimizing interference. *J. Netw. Comput. Appl.*, 84 :68–81, 2017. ISSN 1084-8045. doi : 10.1016/j.jnca.2017.02.004.

[194] Z. Mann et M. Szabó. Which is the best algorithm for virtual machine placement optimization ? *Concurrency and Computation : Practice and Experience*, 29(10) :e4083, 2017. doi : 10.1002/cpe.4083. e4083 cpe.4083.

[195] N. Amruthnath et T. Gupta. A research study on unsupervised machine learning algorithms for early fault detection in predictive maintenance. In *2018 5th International Conference on Industrial Engineering and Applications (ICIEA)*, pages 355–361, 2018. doi : 10.1109/IEA.2018.8387124.

[196] T. Chaabouni et M. Khemakhem. Energy management strategy in cloud computing : A perspective study. *J. Supercomput.*, 74(12) :6569–6597, 2018. ISSN 0920-8542. doi : 10.1007/s11227-017-2154-z.

[197] Bhagyalakshmi et D. Malhotra. A critical survey of virtual machine migration techniques in cloud computing. In *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*, pages 328–332, 2018. doi : 10.1109/ICSCCC.2018.8703322.

[198] A. Ashraf et I. Porres. Multi-objective dynamic virtual machine consolidation in the cloud using ant colony system. *International Journal of Parallel, Emergent and Distributed Systems*, 33(1) : 103–120, 2018. doi : 10.1080/17445760.2017.1278601.

[199] J. Whitney et P. Delforge. *Data Center Efficiency Assessment*. 2019.

[200] R. Bellman. Dynamic programming. *Science*, 153(3731) :34–37, 1966.

[201] F. Bellosa. The benefits of event : Driven energy accounting in power-sensitive systems. EW 9, page 37–42, New York, NY, USA, 2000. Association for Computing Machinery. ISBN 9781450373562. doi : 10.1145/566726.566736.

[202] L. Breiman. Bagging predictors. Technical report, Journal of Time Series Analysis, 1994.

[203] A. Burkov. *The Hundred-Page Machine Learning Book*. 2019.

[204] G. D. Crnkovic. Constructive research and info-computational knowledge generation. 2010.

[205] H. P. D. Kernel estimation of a distribution function. *Communications in Statistics - Theory and Methods*, 14(3) :605–620, 2007. doi : 10.1080/03610928508828937.

[206] W. Dargie. A stochastic model for estimating the power consumption of a processor. *IEEE Transactions on Computers*, 64(5) :1311–1322, 2015.

[207] R. H. Dieck. *Measurement uncertainty : methods and applications*. ISA, 2007.

[208] A. Dixit. Bin packing algorithm for virtual machine placement in cloud data centers for resource management. *International Journal for Research in Applied Science and Engineering Technology*, 6 :437–439, 2018.

[209] J. C. Dunn. Well-separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, 4(1) : 95–104, 1974. doi : 10.1080/01969727408546059.

[210] S. Fidanova. Evolutionary algorithm for multiple knapsack problem. 2004.

[211] Gartner. Gartner says worldwide iaas public cloud services market grew 37.3in 2019, 2020. URL https://www.gartner.com/en/newsroom/press-releases/2020-08-10-gartner-says-worldwide-iaas-public-cloud-services-market-grew-37-point-3-percent-in-2019.

[212] F. W. Glover. Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.*, 13 :533–549, 1986.

[213] R. P. Goldberg. Survey of virtual machine research. *Computer*, 7(9) :34–45, Sept. 1974. ISSN 0018-9162. doi : 10.1109/MC.1974.6323581.

[214] J. H. Holland. *Adaptation in Natural and Artificial Systems : An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992. ISBN 0262082136.

[215] J. H. Holland. *Adaptation in Natural and Artificial Systems : An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992. ISBN 0262082136.

[216] N. Jones. How to stop data centres from gobbling up the world's electricity. *Nature*, 561 :163–166, 2018.

[217] J. Kaivosoja. Green data centers : Evaluating efficiency. 2016.

[218] R. Lent. A model for network server performance and power consumption. *Sustainable Computing : Informatics and Systems*, 3(2) :80 – 93, 2013. ISSN 2210-5379. doi : https://doi.org/10.1016/j.suscom.2012.03.004.

[219] T. M. Mitchell. *Machine Learning*. 1990.

[220] R. H. Myers. *Classical and modern regression with applications*, volume 2. Duxbury press Belmont, CA, 1990.

[221] G. C. Plateform. Google compute engine uses live migration technology to service infrastructure without application downtime, 2015. URL https://goo.gl/Ui3HFd.

[222] W. K. Potts. The chorus-line hypothesis of manoeuvre coordination in avian flocks. *Nature*, 309 (1) :344–345, 1984. ISSN 5966. doi : 10.1038/309344a0.

[223] C. W. Reynolds. Flocks, herds and schools : A distributed behavioral model. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, page 25–34, New York, NY, USA, 1987. Association for Computing Machinery. ISBN 0897912276. doi : 10.1145/37401.37406.

[224] J. W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on computers*, 100(5) :401–409, 1969.

[225] G. Schulz. *The Green and Virtual Data Center*. Auerbach Publications, USA, 1st edition, 2009. ISBN 1420086669.

[226] D. J. Sheskin. *Handbook of parametric and nonparametric statistical procedures*. crc Press, 2003.

[227] R. Tibshirani. Regression shrinkage and selection via the lasso. 58(1) :267–288, 1996.

[228] V. Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

A

Outils d'évaluation

A.1 MAE & MRE

Pour évaluer la précision du modèle étudié dans le chapitre 6, nous utilisons les critères suivants :

- **L'erreur moyenne absolue**, ou *Mean Absolute Error* (MAE), est mesurée en se basant sur la moyenne de la différence absolue entre la prédiction et les résultats potentiels. Elle donnée par :

$$MAE = \frac{1}{n}\sum_{i=1}^{n} \|\tilde{y}_i - y\| \tag{A.1}$$

- **L'erreur moyenne relative**, ou *Mean Relative Error* (MRE). Elle implique l'erreur relative moyenne de la prédiction et est donnée comme suit :

$$MRE = \frac{1}{n}\sum_{i=1}^{n} \left\| 1 - \frac{\tilde{y}_i}{y} \right\| \tag{A.2}$$

  Où $\tilde{y}_i$ représente la valeur prédite du modèle.

A.2 PDCVM & SVTAH

Dans le chapitre 6, nous considérons également les violations des SLAs, car c'est l'un des critères les plus importants pour tout processus de migration d'une VM. Il existe deux approches standard pour représenter les violations de SLA :

- **Le temps de violation des SLA par hôte actif**, ou *SLA Violation Time per Active Host* (SVTAH), qui correspond au temps de violation des SLA sur la durée totale de fonctionnement de l'hôte. Le temps de violation du SLA se réfère à 100% de l'utilisation du CPU pendant ce temps sur l'hôte actif. C'est-à-dire qu'une fois que l'utilisation du CPU d'un hôte est à 100%, les VM appartenant à l'hôte ne

peuvent pas obtenir la capacité CPU requise, ce qui entraîne des violations de SLA. En général, l'équation SVTAH est donnée comme suit :

$$SVTAH = \frac{1}{M}\sum_{i=1}^{M}\frac{T_{v_i}}{T_{a_i}} \tag{A.3}$$

où M est le nombre d'hôtes dans un centre de données, T_{v_i} est le temps total pendant lequel l'utilisation de la CPU atteint 100%, et T_{a_i} indique le temps total d'état actif de l'hôte i.

- Dégradation des performances causée par la migration des VM, *Performance Degradation Caused by VM Migration* (PDCVM), qui suppose que la migration d'une VM peut avoir une influence négative sur les performances des applications s'exécutant sur une VM dans un délai limité. La dégradation des performances due à une VM peut être exprimée comme suit :

$$PDCVM = \frac{1}{N}\sum_{j=1}^{N}\frac{P_{d_i}}{P_{r_j}} \tag{A.4}$$

Où les paramètres N, P_{d_i}, et P_{r_j} correspondent respectivement au nombre de VMs, à la dégradation de performance estimée causée par la $j^{ème}$ VM , et à la capacité totale du CPU requise par la $j^{ème}$ VM pendant sa durée de vie. Par conséquent, la mesure des violations des SLAs peut être donnée comme le produit $SVTAH \times PDCVM$.

A.3 Validation du *Clustering*

Le but principal de l'illustration, dans le chapitre 7, était de déterminer la pertinence de l'algorithme proposé. Pour l'algorithme standard FCM et ses variantes, le nombre de clusters C est le paramètre le plus significatif. Si C correspond au nombre non spécifié de sous-ensembles existant dans les données, il est plus probable que le processus de *Clustering* donne un aperçu efficace de la structure sous-jacente des données. Ainsi, deux indices de validité des clusters sont utilisés pour vérifier l'efficacité du choix de C.

- **L'indice Xie-Beni (XBI)** : est un indice de *Clustering* flou interne, défini comme un rapport entre la compacité et la séparation de l'algorithme de *Clustering* flou. Le XBI a été mathématiquement prouvé par sa dépendance à l'indice de Dunn, une fonction robuste de validité des clusters [209]. L'objectif est d'obtenir une structure de *Clustering* dans laquelle un élément de l'ensemble de données appartient fortement à un certain cluster tout en maintenant une séparation aussi grande que

possible des centres des autres clusters. Le XBI peut être mathématiquement exprimé comme suivant :

$$XBI = \frac{\sum_{k=1}^{C} \sum_{i=1}^{N} u_{ki} ||x_i - v_k||^2}{\min_{i \neq k} \|v_k - v_i\|^2} \tag{A.5}$$

Notez qu'une petite valeur de XBI signifie que les clusters sont compacts et clairement séparés.

- **Adjusted Indice Rand (ARI)** : est un indice de validation externe. Il s'agit de la formulation corrigée du biais de *Rand Index* (RI) [155]. Ce dernier mesure la similarité entre une partition G, représentant les informations de *Groung Truth* des éléments de l'ensemble de données, et une partition H générée par le processus de *Clustering*. Il est donné par ce qui suit :

$$RI(G, H) = \frac{p_{00} + p_{11}}{p_{00} + p_{01} + p_{10} + p_{11}} = \frac{p_{00} + p_{11}}{p_t} \tag{A.6}$$

Où p_{00} est la somme des paires séparées dans les partitions G et H ; p_{11} est le nombre total de paires qui sont réunies dans les deux partitions ; p_{01} (p_{10}, respectivement) est le nombre de paires qui sont regroupées dans G (H, respectivement) mais pas dans H (G, respectivement) ; $p_t = \binom{N}{2} = \frac{N(N-1)}{2}$ est le nombre total de paires. Comme la valeur de RI n'est pas stable, l'ARI est introduit pour pallier cet inconvénient en considérant ce qui suit :

$$ARI(G, H) = \frac{\sum_{g,h} \binom{p_{gh}}{2} - \frac{\sum_g \binom{p_g}{2} \sum_h \binom{p_h}{2}}{\binom{N}{2}}}{\frac{1}{2}[\sum_g \binom{p_g}{2} + \sum_h \binom{p_h}{2}] - \frac{\sum_g \binom{p_g}{2} \sum_h \binom{p_h}{2}}{\binom{N}{2}}} \tag{A.7}$$

Où p_{gh} est le nombre d'éléments appartenant à G et H, p_g et p_h sont le nombres d'éléments appartenant à G et H. On note que ARI prend des valeurs dans l'intervalle $[0, 1]$; ainsi, des valeurs autour de 1 correspondent à une grande ressemblance entre les partitions considérées.