République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université A/Mira de Béjaia
Faculté des Sciences Exactes
Département d'Informatique

MÉMOIRE DE MASTER RECHERCHE

En
Informatique

Option
Réseaux et Systèmes Distribués

Thème

Équilibrage de charge (données) dans le cloud computing

Présenté par : M. KHELOUFI Abderrahim

Soutenu le 2 Juillet 2018 devant le jury composé de :

Présidente	**Mme. GHANEM Souhila**	**MAA**	**U. A/Mira Béjaia.**
Examinateur	**M. BEDJOU Khaled**	**MAA**	**U. A/Mira Béjaia.**
Promotrice	**Mme. YAICI Malika**	**MCB**	**U. A/Mira Béjaia.**

Béjaia, Juillet 2018.

⁕ *Remerciements* ⁕

Je tiens tout d'abord à remercier Dieu le Tout Puissant et Miséricordieux qui m'a donné le courage, la force et l'endurance pour pouvoir achever ce travail et je le pris toujours qu'il soit à mes côtés.

J'adresse tous mes sincères et respectueux remerciements à toute personne ayant contribué à la réalisation de ce P.F.C, particulièrement à :
Mon encadreur Mme Yaici Malika pour m'avoir fourni une aide précieuse, m'avoir dirigé et accompagné pendant cette période de travail.

Je remercie tous les enseignants ayant contribué à ma formation.

Enfin, je remercie les membres du jury qui m'ont fait l'honneur de juger mon travail, espérons qu'ils trouveront l'expression de mes profonds respects et croire à mes sincère gratitude.

⁂ *Dédicaces* ⁂

Je dédie ce modeste travail à :
$\mathcal{A}$ mes trés chers parents, qui n'ont cessé de me soutenir tout au long de mon parcours d'étude, je les remercies pour leurs patience et leurs amour qui m'ont donné la force pour continuer mes études,
$\mathcal{A}$ mes chères sœurs, oncles, tantes, cousins et cousines,
$\mathcal{A}$ toute ma famille,
$\mathcal{A}$ mes amis et collègues, et tous ceux qui m'ont aidé,

$\mathcal{M}$. ***KHELOUFI Abderrahim***

Table des matières

Table des figures

Liste des tableaux

Notations et symboles

IT : Information Technology
S3 : Simple Storage Service
EC2 : Elastic Compute Cloud
QoS : Quality of Service
IaaS : Infrastructure as a Service
PaaS : Platform as a Service
SaaS : Software as a Service
VPN : Virtual Private Network
VM : Virtual Machine
SLA : Service Level Agreement

EC : Élément de Calcul
SOA : Service Oriented Architecture
MPI : Message Passing Interface
CPU : Central Processing Unit
DDBS : Distributed database system
HTTP : Hypertext Transfer Protocol
PGA : Proposed Genetic Algorithm

CH : Cluster Head
DL : Disponible Localement

Introduction générale

Le cloud computing (appelé en français l'informatique en nuage) apparaît comme une technologie à la croissance très rapide dans le monde des technologies de l'information (IT), qui se concentre sur la fourniture de services informatiques et de ressources informatiques 24 heures sur 24, et à travers le monde à ses utilisateurs sur Internet. L'infrastructure et les services de cloud computing sous-jacents sont généralement détenus et gérés par un tiers, connu sous le nom de fournisseur de services cloud. Le principal avantage du cloud computing par rapport aux technologies informatiques existantes est le libre-service à la demande, les services de réseau étendus, l'élasticité rapide, la mise en commun des ressources et le service mesuré. La croissance du service cloud peut entraîner un ralentissement du débit, l'utilisation des ressources informatiques et, en fin de compte, réduire l'efficacité du système cloud. L'un des inconvénients associés au cloud computing est qu'il fonctionne dans un environnement dynamique et que le bon mécanisme d'ordonnancement des tâches doit être déployé pour s'assurer qu'aucun noeud informatique n'est sous ou sur utilisé, sinon cela affectera le temps de réponse global ou le débit.

Le cloud computing est la technologie émergente dans un environnement distribué composé de plusieurs centres de données, serveurs, machines virtuelles, équilibreurs de charge, etc. qui sont connectés intelligemment. De plus, le nuage traite de nombreuses choses comme le stockage et la récupération de documents, le partage de contenu multimédia, l'octroi de ressources connexes sur le modèle de pay-as-you-go et bien plus encore. Même si l'ère des ordinateurs et de l'Internet des objets (IoT) progresse beaucoup en termes de réactivité, de fiabilité et de flexibilité,

il reste encore une marge d'amélioration dans l'ordonnancement, l'allocation optimale des ressources et les algorithmes de gestion, classés de complexité difficile et NP-complete. Par conséquent, il est nécessaire d'aborder ces problèmes complexes en utilisant différentes techniques. L'ordonnancement efficace des tâches et la gestion des ressources sont un problème complexe de l'informatique distribuée, mais elles n'en sont qu'à leurs débuts malgré des recherches exhaustives ces dernières années.

L'équilibrage de charge dans l'environnement informatique en nuage a un impact important sur les performances du cloud. Un bon équilibrage de la charge rend le cloud computing plus efficace et améliore la satisfaction des utilisateurs. Des algorithmes génétiques peuvent être utilisés pour améliorer les performances des approches d'équilibrage de charge. Des algorithmes d'ordonnancement conventionnels tels que le Round Robin, le Premier arrivé premier servi, l'optimisation de colonie de fourmi etc. ont été largement utilisés dans de nombreux systèmes de cloud computing. Le cloud reçoit les tâches des clients à un rythme rapide et l'allocation des ressources à ces tâches doit être gérée de manière intelligente. Outre les problèmes de cloud importants dans les domaines de l'ordonnancement, de l'allocation des ressources et de la sécurité, le cloud computing met en évidence d'autres problèmes urgents dont la tolérance aux pannes lors de l'exécution des tâches et des machines virtuelles. Ces types de problèmes dans un plan général sont appelés NP-difficile (temps polynomial non-déterministe), ce qui signifie qu'il n'y a pas de solution exacte et pas de solution rapide.

Dans le cadre de ce mémoire, nous allons nous intéresser au problème de l'équilibrage de charge dans le cloud, car dans la littérature le problème est toujours soumis a la recherche pour améliorer les performances enregistré, et donc nous allons tenter de proposer un algorithme qui va participer, ou qui va répondre a la problématique.

Pour cela, nous allons entreprendre notre étude selon les quatre chapitres suivants :

Dans le *premier chapitre* nous allons présenter le cloud, et donner ses caractéristiques, ses modèles de service, ses modèles de déploiement et ses limites.

Dans le *deuxième chapitre* nous allons détailler l'équilibrage de charge dans les systèmes distribuée, notamment les fermes de calcul et les grilles informatiques.

Dans le *troisième chapitre*, nous allons faire un état de l'art sur les solutions présentes dans la littérature pour l'équilibrage de charge dans le cloud.

A la fin, dans le *quatrième chapitre*, nous allons proposer un algorithme pour répondre a la problématique d'équilibrage de charge dans le cloud.

Nous terminerons ce mémoire par une conclusion et quelques perspective.

Chapitre 1

Cloud Computing

1.1 Introduction

Depuis quelques années, un nouveau paradigme nommé *cloud computing* révolutionne la façon dont les entreprises et les particuliers accèdent à des ressources informatiques telles que la puissance de calcul, la capacité de stockage...etc. Adoptant une approche élastique d'utilisation à la demande (pay-as-you-go), le cloud computing est une solution très intéressante pour l'externalisation et la simplification des services informatiques.

Dans ce premier chapitre, nous présenterons une définition, les modèles de services, les caractéristiques, les avantages et les inconvénients du cloud computing.

1.2 Historique

Il est difficile de dire avec précision quand a été inventé le *cloud computing*. Selon certains, il faut remonter en 1960, lorsque John McCarthy a écrit que «*le calcul pourrait un jour être organisé comme un service public.*» Ensuite, l'informatique en grille, un concept qui a vu le jour au début des années 1990, facile d'accès comme un réseau électrique a également, contribué à l'informatique en nuage.

Selon une autre source, c'est l'avènement des réseaux dans les années 1970 qui a rendu possible l'exécution déportée des tâches informatiques.

D'autres enfin mentionnent le fait qu'Amazon, site de commerce électronique de dimension mondiale, a trouvé dans le cloud computing une solution élégante à la sous-utilisation de son parc de serveurs informatiques en dehors des périodes de fête (qui représentent en termes de commandes un pic temporel ponctuel d'utilisation). En louant ses serveurs à la demande et en proposant à ses clients ses outils S3 (simple storage service) et EC2 (elastic compute cloud), qui offrent respectivement des services de stockage de données et de calcul, Amazon a pu rentabiliser ses propres investissements en matériel informatique.

L'expression « *cloud computing* » a, quant à elle, été citée pour la première fois en 1997 par un professeur en systèmes de l'information, Ramnath Chellappa, qui a défini les limites de l'informatique non en termes techniques mais en termes économiques. D'autres sociétés, comme Salesforces, Google 101, ou IBM ont commencé dès 1999, à développer une économie numérique fondée sur ces principes. Toutes ces entreprises de dimension mondiale participent de manière active à la création de centres (data-centers ou clusters) offrant une puissance de calcul et de stockage inégalée. Ces data-centers sont un des enjeux stratégiques majeurs de la décennie 2015-2025 [2].

1.3 Définitions

De nombreuses définitions existent, en voici quelques-unes : Le NIST (National Institute of Standards and Technology) qui définit le *Cloud Computing* comme suit : "*Le Cloud Computing est l'ensemble des disciplines, pratiques, technologies et modèles commerciaux utilisés pour délivrer, comme un service à la demande et par le réseau, des capacités informatiques (réseaux, serveurs, stockage, applications, et services)*" [34].

Une autre définition donnée lors de la 10e conférence international IEEE : " *Le Cloud computing est un ensemble de services accessible par réseau, évolutive, avec QoS garanti, personnalisable et peu coûteux, à la demande, qui pourraient être accessibles d'une manière simple et omniprésente* " [44].

Définition allégée de Wikipedia : "*Le Cloud computing est un concept de*

déportation sur des serveurs distants des traitements informatiques traditionnellement localisés sur le poste utilisateur" [16].

1.4 Caractéristique

Les principales caractéristique du Cloud Computing sont [34] :

— **À la demande, en libre-service :** Un consommateur peut disposé de manière unilatérale des capacités de calcul, stockage, selon les besoins, sans nécessiter d'interaction humaine avec chaque fournisseur de services.

— **Accès réseau large bande :** Les ressources ou les datacenters (centres de données) sont disponibles sur le réseau et accessibles via Internet pour bénéficier d'une excellente connectivité. Il sont accessibles via des mécanismes standard qui favorisent l'utilisation sur des plateformes client hétérogènes (par exemple, téléphones mobiles, tablettes, ordinateurs portables et stations de travail).
Les grands fournisseurs répartissent les datacenters sur la planète pour fournir un accès aux systèmes en moins de 50 ms de n'importe quel endroit.

— **Réservoir de ressources (non localisées) :** Les ressources et datacenters sont mis à disposition des utilisateurs en utilisant des dizaines de milliers de serveurs et de moyens de stockage pour permettre des montées en charge rapides, ces données peuvent se situer dans des emplacements indépendamment des utilisateurs. Mais il est souvent possible de choisir une zone géographique pour mettre les données "près" des utilisateurs.

— **Redimensionnement rapide (élasticité) :** Une nouvelle instance d'un serveur peut être mise en ligne en quelques minutes, l'arrêt et le redémarrage en quelques secondes. Toutes ces opérations peuvent s'effectuer automatiquement par des scripts. Ces mécanismes de gestion permettent l'approvisionnement des données de manière rapide et la facturation à l'usage

en adaptant la puissance de calcul au trafic instantané.

— **service mesurable :** les systèmes cloud contrôlent et optimisent automatiquement l'utilisation des ressources en exploitant une capacité de mesure (pay-as-you-go) à un niveau d'abstraction approprié au type de service (stockage, traitement, bande passante et comptes d'utilisateurs actifs, par exemple). L'utilisation des ressources peut être surveillée, contrôlée et signalée, ce qui garantit la transparence tant pour le fournisseur que pour le consommateur du service utilisé.

— **Facturation à l'usage :** La facturation est calculée en fonction de la durée et de la quantité de ressources utilisées. En générale, il n'y a pas de coût de mise en service (c'est l'utilisateur qui réalise les opérations). En cas d'interruption d'une unité de traitement, la factorisation est interrompue [39].

1.5 Modèles de service du cloud computing

Le Cloud Computing se repose sur trois modèles fondamentaux : IaaS, PaaS et Saas. Le degré d'externalisation est variable. Les concepts de IaaS (Infrastructure as a Service), PaaS (Plateform as a Service) et Saas (Software as a Service) indiquent ce degré.

La figure 1.2 montre comment les responsabilités sont réparties entre le fournisseur de services et l'entreprise suivant les modèles internes, IaaS, PaaS, SaaS.

1.5.1 IaaS (Infrastructure as a Service)

C'est un modèle où l'entreprise dispose d'une infrastructure informatique (des capacités de calcul, de stockage et d'une bande passante suffisante) qui se trouve en fait chez le fournisseur. Cette infrastructure est mise à disposition de façon à gérer automatiquement la charge de travail requise par les applications. Cependant,

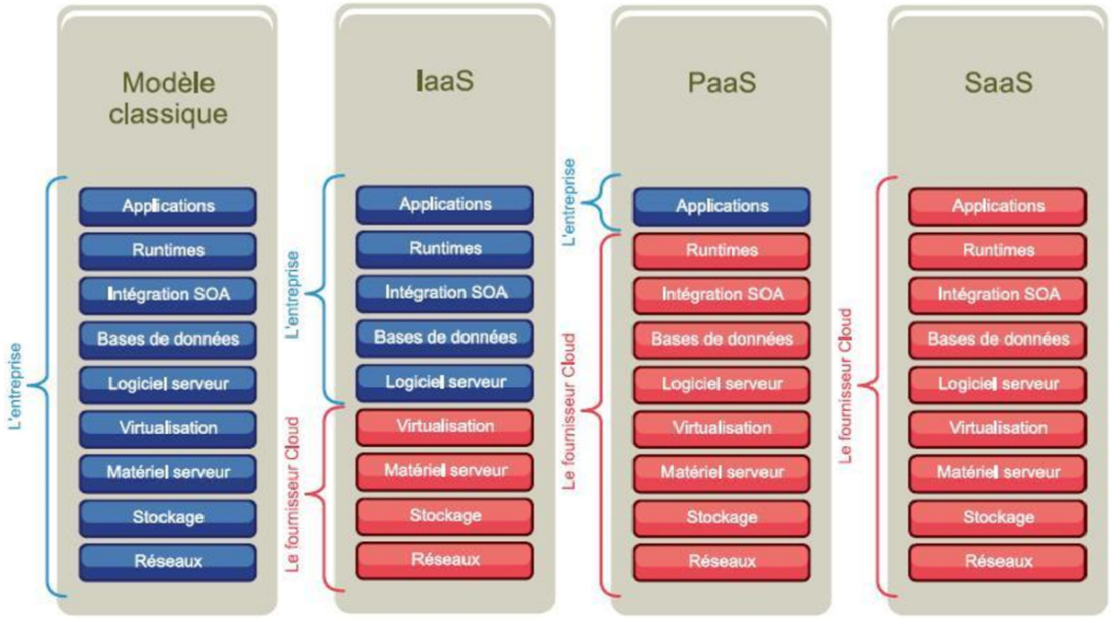


FIGURE 1.1 – Schématisation des concepts de IaaS, PaaS, SaaS [25]

l'entreprise y a accès sans restriction, comme si le matériel se trouvait dans ses locaux. Ceci lui permet de s'affranchir complètement de l'achat et de la gestion du matériel. L'entreprise exploite le matériel comme un service à distance. Cette couche permet à l'entreprise de se concentrer en premier sur ses processus métiers sans se préoccuper du matériel [13].

1.5.2 PaaS (Platform as a Service)

C'est une plateforme d'exécution, de déploiement et de développement des applications.
La plateforme PaaS regroupe la partie développeur (client) et système (fournisseur) du Cloud Computing. Elle propose des fonctions qui privent le développeur de la gestion des utilisateurs ou des questions de disponibilité par exemple. Le développeur a ainsi uniquement besoin d'héberger son application pour qu'elle soit disponible en SaaS [17].

1.5.3 SaaS (Software as a Service)

C'est la mise à disposition par Internet d'applications informatiques (logiciels) comme un service dans le cadre d'un abonnement, les données sont aussi stockées sur un serveur de l'opérateur SaaS. Il n'y a donc aucun pré requis sur le poste client si ce n'est d'avoir un accès réseau au Cloud (en général Internet). Le déploiement, la maintenance, la supervision du bon fonctionnement de l'application et la sauvegarde des données, sont alors de la responsabilité du fournisseur de services.
C'est en quelque sorte la partie visible du Cloud Computing pour l'utilisateur final, qui n'a plus besoin d'installer l'application sur son poste, et qui accède à son compte par le Web, sur un environnement sécurisé [12].

1.6 Modèles de déploiement du cloud computing

1.6.1 Le Cloud public

Le *Cloud public* ne veut pas dire que les données sont accessibles à n'importe qui, mais uniquement que vos données sont hébergées sur une multitude de serveurs eux-mêmes accessibles par un nombre déterminé d'utilisateurs. Les Cloud les plus connus sont d'ailleurs des Cloud public. Ces services peuvent être gratuits ou payants. Exemples de clouds publics : Amazon Elastic Compute Cloud (EC2), Sun Cloud, IBM's Blue Cloud, Google AppEngine and Windows Azure Services Platform.

1.6.2 Le Cloud privé

Un *Cloud privé* est un ensemble des services et des ressources disponible pour un seul client par exemple une entreprise ou groupement d'entreprise (appelé organisation), il peut être géré par l'entreprise elle-même, ou ses branches, dans ce cas il s'appelle "Le Cloud privé Interne", en d'autre façons il peut être géré par un prestataire externe loué par l'entreprise, dans ce cas s'appelle "Le Cloud privé Externe", il est accessible via des réseaux sécurisés de type VPN (Virtual Private Network).

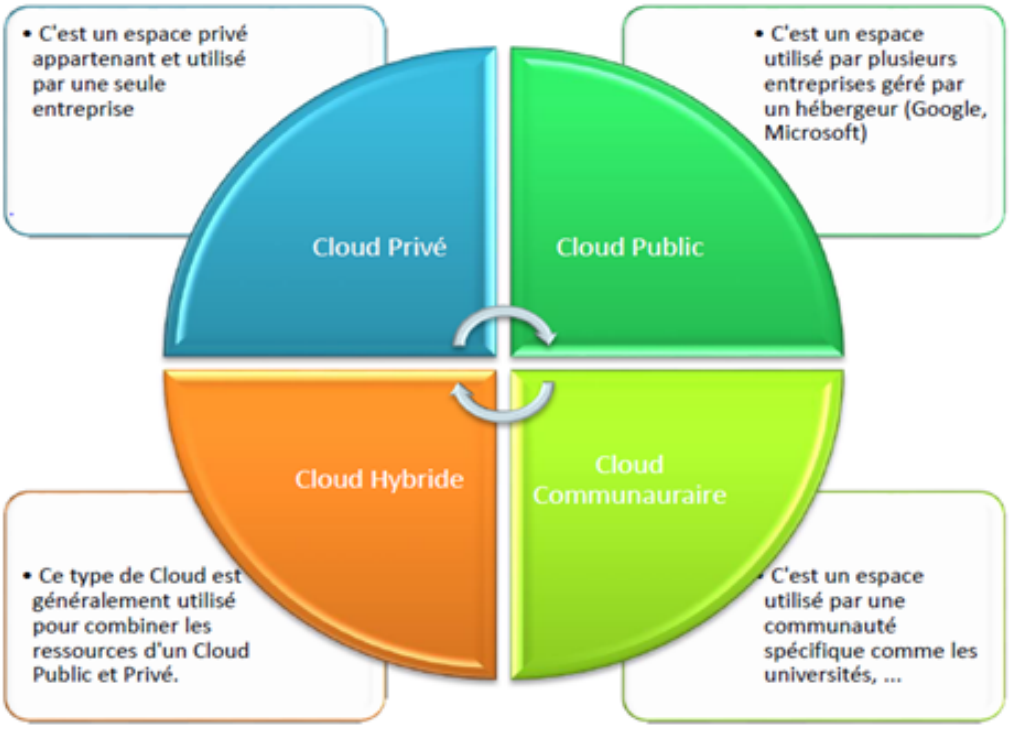


FIGURE 1.2 – Modèles de déploiements du Cloud Computing [9]

L'avantage de ce type de Cloud par rapport au Cloud public réside dans l'aspect de la sécurité et la protection des données, car l'infrastructures est dédié [14].

1.6.3 Le Cloud hybride

L'infrastructure *cloud hybride* est une composition de deux ou plusieurs infrastructures cloud distinctes (privées, communautaires ou publiques) qui restent des entités uniques, mais sont liées par une technologie standardisée ou propriétaire qui permet la portabilité des données et des applications (par exemple, l'éclatement du cloud entre nuages) [34].

1.6.4 Le Cloud Communautaire

Un *Cloud communautaire* est utilisé par plusieurs organisations qui ont les mêmes intérêts. Dans une telle architecture, l'administration du système peut être effectuée par l'une ou plusieurs des organisations partageant les ressources du Cloud. Ainsi cela peut porter sur l'hébergement d'une application métier très spécialisée, mais commune à de très nombreuses entreprises, qui décident de fédérer leurs efforts [15].

1.7 Eléments du Cloud Computing

Les éléments pouvant constitué le système Cloud sont les suivants [12] :

1.7.1 La virtualisation

La virtualisation est la principale technologie dans le Cloud, elle permet une gestion optimisée des ressources matérielles en disposant de plusieurs machines virtuelles sur une machine physique. C'est une technologie qui permet une plus grande modularité dans la répartition des charges et la reconfiguration des serveurs en cas d'évolution ou de défaillance momentanée. Le principe de virtualisation permet d'intégrer les différents serveurs de façons plus flexible pour faciliter l'utilisation. Le but de la virtualisation est de faire la transparence d'utilisation et l'efficacité d'exploitation des ressources, d'assurer le fonctionnement des différents services et la séparation entre de multiples locataires (utilisateurs) impliqués dans un matériel physique [31].

1.7.2 L'infrastructure

L'infrastructure informatique du Cloud est un assemblage de serveurs, d'espaces de stockage et de composants réseau organisés de manière à permettre une croissance incrémentale supérieure à celle que l'on obtient avec les infrastructures classiques. Ces composants doivent être sélectionnés pour leur capacité à répondre aux exigences

d'extensibilité, d'efficacité, de robustesse et de sécurité. Les serveurs d'entreprise classiques ne disposent pas des capacités réseau, de la fiabilité ni des autres qualités nécessaires pour satisfaire efficacement et de manière sécurisée les accords de niveau de service SLA (Service Level Agreement) [30].

1.7.3 Le Datacenter

Un centre de traitement de données, en anglais " *datacenter* " est un site physique sur lequel sont regroupés des équipements constituants le système d'information de l'entreprise (mainframes, serveurs, baies de stockage, équipements réseaux et de télécommunications, etc.). Il peut être interne ou externe à l'entreprise, exploité ou non avec le soutien de prestataires. Il comprend en général un contrôle sur l'environnement (climatisation, système de prévention contre l'incendie, etc.), une alimentation d'urgence et redondante, ainsi qu'une sécurité physique élevée. Des particuliers ou des entreprises peuvent venir y stocker leurs données suivant des modalités bien définies [37].

1.7.4 La plateforme collaborative

Une plate-forme de travail collaboratif est un espace de travail virtuel. C'est un outil, parfois sous la forme d'un site internet, qui centralise tous les outils liés à la conduite d'un projet et les met à disposition des acteurs (clients). L'objectif du travail collaboratif est de faciliter et d'optimiser la communication entre les individus dans le cadre du travail ou d'une tâche [37].

1.8 Avantages

Le cloud computing offre de nombreux avantages, d'après [24] et [19]. En voici les plus principaux :

— L'accès aux applications de n'importe où ;
— Augmentation fonctionnelle des capacités ;

— Augmentation de la puissance de calcul ;
— Réduction des coûts d'infrastructure, de développement et des logiciels ;
— Accès aux ressources plus flexible ;
— Grande capacité de stockage (quasi illimitée) ;
— Gestion des mises à jour plus simple et rapide ;
— Pas de perte de données ;
— Infrastructure allouée et disponible juste à temps ;
— La possibilité offerte aux utilisateurs de réaliser des calculs parallèles.

1.9 Limites

Le cloud computing possède quelques inconvénients qui sont abordés dans [19] et [35]. Parmi ces inconvénients, il y a :

— Nécessité d'une connexion Internet constante, la rupture de la connexion Internet implique la perte d'accès aux applications et aux données ;
— Ce n'est pas toutes les applications qui fonctionnent dans le cloud ;
— Les données stockées peuvent ne pas être sécurisé ;
— Mauvais fonctionnement avec les connexions à basse vitesse ;
— L'utilisation des réseaux publics, entraine des risques supplémentaires de cyberattaques liés à la sécurité du cloud ;
— La bande passante peut faire exploser le budget. Par exemple, pour les entreprises situées dans une zone non desservie par une excellente connexion Internet, l'investissement pour avoir une bonne connexion Internet peut ne pas être très raisonnable.

1.10 Conclusion

Dans ce premier chapitre, nous avons présenté les principaux concepts et points clés du Cloud Computing. L'apparition du Cloud Computing a donné un nouveau mode de consommation de l'informatique, une consommation a la demande, et a changé la manière d'investissement des entreprises dans les infrastructures informa-

tiques. Un coût relativement faible, stockage évolutif illimité et une grande puissance de calcul sont les promesses du Cloud Computing.

Chapitre 2

Équilibrage de charge dans les systèmes distribués

2.1 Introduction

L'*équilibrage de charge* dans les système distribués est un élément important lors de la mise en place de services amenés à croître. Il faut s'assurer que la capacité à monter en charge soit la plus optimale possible afin d'éviter toute dégradation que ce soit en terme de performances ou de fiabilité lors d'affluences importantes. Le principe de base de l'équilibrage de charge (Load Balancing) consiste à effectuer une distribution des tâches à des machines de façon intelligente.

2.2 Systèmes distribués existants

Nous présentons ici trois grands types : Les fermes de calcul ou clusters, les Grilles de calcul et les Clouds [29, 33].

2.2.1 Les clusters ou fermes de calcul

Définition du cluster : Un cluster , en français grappe, est une concentration de deux ou plusieurs machines, qui accèdent à des données communes.

La FIGURE 2.1 montre deux niveaux, le niveau 0 contient un gestionnaire de clusters qui coordonne les ECs (éléments de calcul) du niveau 1.

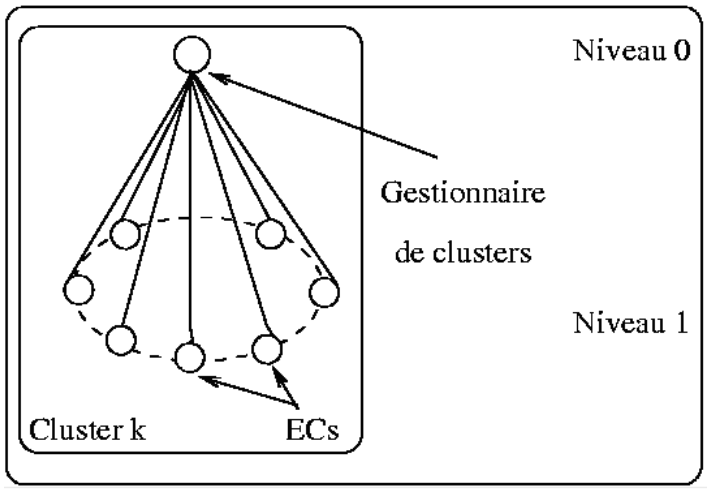


FIGURE 2.1 – Modèle générique de représentation d'un cluster [33]

Les scientifiques ont besoin d'effectuer des calculs de plus en plus massifs dans leur travail de recherche ; c'est ce qui a poussé le développement d'ordinateurs toujours plus puissants : les super-ordinateurs. Dans les deux dernières décennies, afin de satisfaire cette demande en super-ordinateurs, les industriels ont regroupé des ordinateurs afin d'accroître les capacités de calcul : ce sont les fermes de calcul (cluster). Plutôt que fabriquer un super-ordinateur avec un processeur très puissant, les industriels préfèrent concevoir des ordinateurs sur la base de processeurs existants afin de constituer une ferme de calcul. Avec ce procédé, de tels super-ordinateurs reviennent moins cher à fabriquer.

Une ferme de calcul, est constituée de différents serveurs composés d'un ou de plusieurs processeurs. Ces serveurs sont connectés au moyen de réseaux de communication très rapides (par exemple Infiniband [22]). L'utilisateur se connecte à la ferme de calcul sur un serveur frontal et exécute son application parallèle qui est conçue pour s'exécuter en parallèle sur un ensemble de processeurs. Généralement les serveurs qui composent une ferme de calcul sont tous similaires : ce sont des plates-

formes homogènes. De plus, les serveurs d'une ferme de calcul sont physiquement localisés au même endroit.

De nos jours les fermes de calcul sont très répandues dans le monde scientifique, dans les universités, les centres de recherche pour les calculs hautes performances (HPC). Elles sont utilisées pour effectuer des simulations de tout genre : simulations physiques (aérodynamique, nucléaire, et autres), prévisions météorologiques (super-ordinateur K de Fujitsu [36]), application à la médecine ou encore à la génétique (Blue Gene [20]).

2.2.2 Les grilles informatiques

Une grille informatique ou grille de calcul (grid computing) peut être considérée comme un super-ordinateur virtuel qui est physiquement constitué de plusieurs machines différentes et délocalisées. Cette infrastructure est dite hétérogène car les éléments qui la constituent ont des vitesses de calcul différentes et des capacités de stockage différentes ; les éléments peuvent aller d'une ferme de calcul à un ordinateur personnel. Ces machines sont interconnectées par des réseaux de communication beaucoup moins rapides que ceux d'une ferme de calcul. Elles sont par exemple interconnectées au moyen de câbles Ethernet ou du wifi. Les grilles sont soit privées ou soit publiques et dans ce cas elles sont accessibles à tous depuis internet.

Les grilles informatiques peuvent être classées dans trois grandes familles :

— les grilles pair-à-pair (peer-to-peer grids ou desktop grids) sont constituées d'une multitude de machines de petite taille, comme des ordinateurs personnels. Le problème à résoudre est découpé en sous problèmes résolus indépendamment sur les machines, avant qu'une machine maître ne récolte les différents résultats.
— les grilles de ressources sont constituées de machines de plus grande taille, telles que des fermes de calcul, comme Grid'5000 [3].
— les grilles de services proposent aux utilisateurs des applications (de calcul scientifique par exemple) déployées sur la grille. Ce sont des architectures orientées services (Service Oriented Architecture ou SOA). Par exemple l'uti-

lisateur fait la demande d'une application à la grille. En cas de disponibilité l'utilisateur fournit ses données à l'application puis récupère ses résultats de manière transparente sans se soucier, ni de la localisation du service, ni de la manière dont le service est réalisé.

2.2.3 Cloud Computing

L'informatique en nuage (cloud computing) est un paradigme qui propose à l'utilisateur de délocaliser ses ressources de calcul et ses ressources de stockage. Les ressources qui servent de support au cloud computing sont généralement des clusters car ceux-ci sont plus homogènes et disponibles que les grilles de calcul. À la manière d'un réseau électrique, les entreprises paient la consommation réelle des services proposés par le cloud du fournisseur qui leur garantit une qualité de calcul et de stockage. Avec ce concept, elles n'ont plus besoin de se préoccuper de l'achat de serveurs de calcul et de stockage ou de l'organisation des infrastructures réseaux.

2.3 Équilibrage de charge

Nous abordons à présent le problème d'équilibrage de charge dans les fermes de calcul et les grilles.

2.3.1 Fermes de calcul

Qui dit ressources de calcul dit aussi nécessité d'une bonne utilisation de celles-ci. En effet, les ressources sont néanmoins en quantité finie alors que les programmes qui s'exécutent sont de plus en plus gourmands en calcul. De plus les utilisateurs peuvent être nombreux à se connecter en même temps à un même super-ordinateur et demander pour leur besoin une partie des ressources de calcul disponibles. Il se dégage le problème de la bonne utilisation de la ferme de calcul. En effet les statistiques d'utilisation des centres de calcul montrent que les machines présentent des cycles inutilisés (idle-time). Des recherches dans ce domaine ont été menées afin d'optimiser à la fois le placement des programmes des différents utilisateurs sur

les ressources de calcul et leur utilisation tout en satisfaisant chaque utilisateur : l'*ordonnancement* est le terme qui englobe ces problèmes d'optimisation.

Prenons un exemple : supposons qu'une machine possède 100 processeurs, qu'un programme parallèle A arrive à la date t = 0 et demande 60 processeurs et dure 1 heure. Un autre programme parallèle B arrive à la date t = 30 minutes et demande 50 processeurs et dure 10 minutes. S'il fonctionne de manière très basique, le logiciel qui s'occupe du placement des tâches sur les processeurs, appelé ordonnanceur de tâches, choisit d'exécuter la tâche A et au bout de 60 minutes, comme 60 processeurs sont libérés, l'ordonnanceur exécute la tâche B. Le problème est que pendant 60 minutes (entre t = 0 et t = 60 minutes) la machine présente des cycles inutilisés dans 40 processeurs, puisqu'elle attend que la tâche A libère des processeurs pour que la tâche B puisse s'exécuter.

2.3.2 Grilles informatiques

Dans une ferme de calcul, les machines sont généralement homogènes (vitesses de calcul), leur nombre est connu à l'avance, et elles sont interconnectées par des réseaux de communication très rapides. Des outils comme MPI (Message Passing Interface) ou OpenMP (Open Multi-processing), permettent de tirer profit de l'homogénéité des machines et des vitesses de communication entre machines. Grâce à l'homogénéité des machines, les applications parallèles exécutées dans les fermes de calcul ne rencontrent pas le problème d'un goulot d'étranglement lié à une machine plus lente que les autres. De plus grâce au réseau très rapide des fermes de calcul, les données peuvent transiter rapidement d'une machine à l'autre.

Sur une grille, une application qui exploite la puissance de calcul des nombreuses ressources est dite distribuée. Pour maintenir de bonnes performances, exécuter une application distribuée sur une grille demande plus de travail en amont, qu'exécuter une application parallèle sur une ferme de calcul. En effet, les communications entre machines peuvent être plus lentes que dans une ferme de calcul ; les machines sont hétérogènes et ont des vitesses de calcul différentes ; les capacités de mémoire vive

sont différentes et il se peut qu'une bibliothèque nécessaire à l'exécution d'un programme ne soit pas installée sur toutes les ressources de calcul.

Les applications distribuées sont constituées de multitudes de tâches à exécuter dans un certain ordre avec des dépendances entre les tâches. Il est primordial d'utiliser au mieux les ressources. Placer quelle tâche sur quelle machine et déterminer à quelle date elle commence, afin d'obtenir la plus grande vitesse d'exécution pour une application distribuée est tout l'enjeu du problème d'ordonnancement dans ce contexte. Ainsi, si on ordonnance mal les tâches d'une application distribuée, une machine de faible vitesse peut globalement ralentir toute l'exécution.

2.4 Approches d'équilibrage de charge

Le problème de l'équilibrage étant un problème relativement ancien, beaucoup d'approches ont été proposées pour le résoudre. Casavant et Kuhl ont défini une taxonomie largement adoptée par la communauté scientifique dont les principales classes sont [11] :

1. Approche statique Vs. approche dynamique : Dans une approche statique, les tâches sont assignées aux machines avant l'exécution de l'application qui les contient. Les informations concernant le temps d'exécution des tâches et les caractéristiques dynamiques des machines sont supposées connues a priori. Cette approche est efficace et simple à mettre en oeuvre lorsque la charge de travail est au préalable suffisamment bien caractérisée. Dans une approche dynamique, l'assignation des tâches aux machines se décide durant la phase d'exécution, en fonction des informations qui sont collectées sur l'état de charge du système. Ceci permet d'améliorer les performances d'exécution des tâches mais au prix d'une complexité dans la mise en oeuvre de cette stratégie, notamment en ce qui concerne la définition de l'état de charge du système, qui doit se faire de manière continue.
2. Approche centralisée Vs. approche distribuée : Dans une approche centralisée, un site du système est choisi comme coordinateur. Il reçoit les informations de charge de tous les autres sites qu'il assemble pour obtenir l'état de charge

global du système. Dans le cas d'une approche distribuée, chaque site du système est responsable de collecter les informations de charge sur les autres sites et de les rassembler pour obtenir l'état global du système. Les décisions de placement de tâches sont prises localement, étant donné que tous les sites ont la même perception de la charge globale du système.

3. Approche source-initiative Vs. receveur-initiative : L'approche source-initiative est appliquée lorsqu'un site, appelé source, détecte qu'il a une surcharge de travail et qu'il cherche à transférer le surplus vers un site faiblement chargé. L'approche *receveur-initiative* s'applique lorsqu'un site faiblement chargé, appelé receveur, demande à recevoir tout ou partie du surplus des sites surchargés.

2.5 Le système d'équilibrage de charge

Un système d'équilibrage de charge est composé de deux éléments essentiels : les politiques et les mécanismes. Les politiques considèrent l'ensemble des choix à effectuer pour distribuer une charge de travail alors que les mécanismes réalisent physiquement la répartition de la charge et fournissent les informations exigées par les politiques. La figure 2.2 illustre la décomposition arborescente d'un système d'équilibrage de charge.

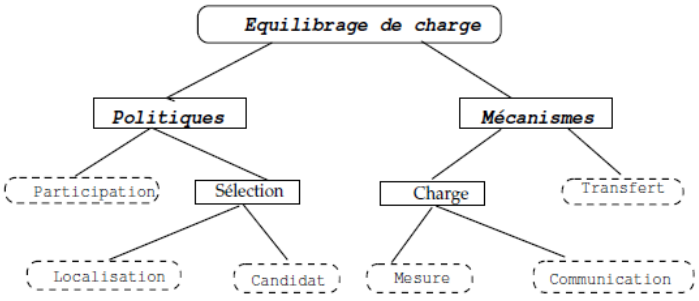


FIGURE 2.2 – Composants d'un système d'équilibrage de charge [11]

Les Politiques :

1. *Politique de participation* : Le but de cette politique consiste à déterminer si un site est dans un état approprié pour participer à un transfert de tâches comme source (site surchargé) ou comme receveur (site sous-chargé).
2. *Politique de sélection de la localisation* : Cette politique est responsable de trouver, pour un site donné, un partenaire (source ou receveur), une fois que la politique de participation a décidé que ce site était soit source, soit receveur.
3. *Politique de sélection des tâches à transférer* : Une fois que les politiques de participation et de localisation ont décidé qu'un site Si est source et qu'un autre site Sj est receveur, cette politique est responsable du choix des tâches à transférer de Si vers Sj.
4. *Mécanisme de mesure de la charge* : Dans toute approche d'équilibrage de charge, une des difficultés majeures est celle qui consiste à évaluer la mesure de la charge d'un site. Dans la plupart des travaux existants, c'est la longueur de la file d'attente qui détermine la charge d'un site. Certains auteurs préconisent comme indicateur de charge, une combinaison entre la longueur de la file d'attente CPU, celle des Entrées/Sorties et l'occupation mémoire. Dans le cas des grilles de calcul, il est nécessaire de tenir compte aussi de l'hétérogénéité des ressources et des réseaux de communication pour mesurer la charge d'un site.
5. *Mécanisme de définition de la charge* : Ce mécanisme essaie de définir la charge globale d'un système en collectant les informations de charge (partielles) sur l'ensemble ou une partie des sites du système. Il faudra alors définir les méthodes selon lesquelles l'information de charge est collectée puis diffusée aux sites.

2.6 Stratégie d'équilibrage de charge

La structure arborescente du modèle proposé nous permet de développer une stratégie hiérarchique à trois niveaux d'équilibrage : Intra-grappe, Extra-grappe et Inter-grappes.

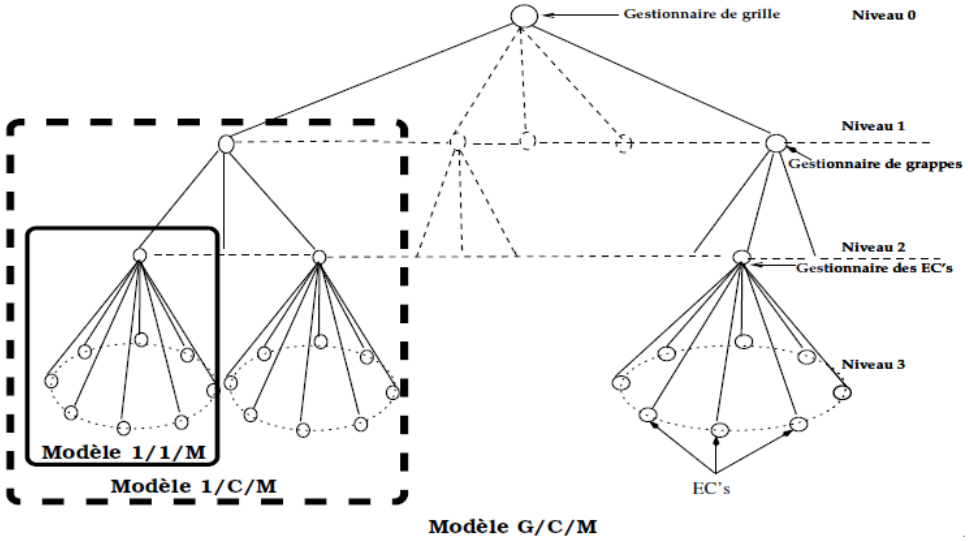


FIGURE 2.3 – Modèle générique de représentation d'une grille [11]

La FIGURE 2.3 représente le modèle générique de représentation d'une grille, en partant du gestionnaire de grille, jusqu'aux éléments de calcul (EC's). Le modèle 1/1/M, représente la plus petite grille possible, à savoir une seule grappe composé de M éléments de calcul. Le modèle 1/C/M, représente une extra-grappe. C'est une extension du modèle précédent, dans le sens où l'on passe d'une seule grappe à C grappes. Le modèle générique G/C/M, correspond à une grille (inter-grappes). L'arbre correspondant a quatre niveaux représentant l'agrégation de G modèles de type 1/C/M.

1. *Équilibrage Intra-grappe* : Dans ce premier niveau, chaque gestionnaire d'éléments de calcul décide de déclencher une opération d'équilibrage en fonction de la charge courante de la grappe qu'il gère. Cette charge est estimée à partir des différentes informations de charge envoyées périodiquement par les EC's qui composent la grappe. Le gestionnaire tente, en priorité, d'équilibrer la charge de la grappe localement en la répartissant entre les EC's qui lui appartiennent. Cette approche de localité a pour objectif de réduire les coûts de communication, en évitant les communications extra/inter grappes.

2. *Équilibrage Extra-grappe* : Dans ce deuxième niveau, l'équilibrage se fait à l'échelle des extra-grappes. Il intervient dans le cas où certains gestionnaires des EC's n'ont pas réussi à équilibrer localement leurs charges. Il y aura ainsi transfert de tâches entre grappes surchargées et grappes sous-chargées de la même extra-grappe. Dans un souci de réduire au maximum les coûts de communication, les grappes réceptrices seront sélectionnées en fonction des débits des réseaux.
3. *Équilibrage Inter-grappes* : Dans ce troisième niveau, l'équilibrage de charge n'est déclenché que si un ou plusieurs gestionnaires de grappes n'arrivent pas équilibrer leurs charges localement entre les grappes qu'ils gèrent. Dans ce cas extrême, il sera alors nécessaire au gestionnaire de grille de transférer un certain nombre de tâches à partir d'extra-grappes surchargées vers d'autres qui sont sous-chargées.

2.7 Conclusion

Nous avons présenté dans ce chapitre trois systèmes distribués, le problème d'équilibrage de charge et les différentes stratégie d'équilibrage de charge. Au chapitre suivant, nous présenterons l'état de l'art pour le problème d'équilibrage de charge dans le cloud computing.

Chapitre 3

État de l'art

3.1 Introduction

Ce chapitre contient une présentation sélective des méthodes développées dans la littérature, traitant l'équilibrage de charge dans le Cloud.

L'équilibrage de charge peut être atteint soit par placement de données ou de fragments de données, soit par placement de taches ou de machine virtuel (VM).

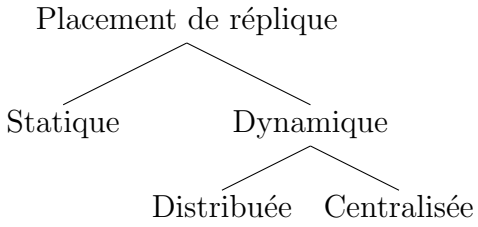
3.2 Placement de donnée

En raison de la taille énorme des données, de la bande passante limitée du réseau, le placement des données sur les centres de données est devenu un enjeu dans l'équilibrage de charge dans le Cloud.
Les ensembles de données traités par un même calcul (une tache) doivent être placés sur le même data center.
Les algorithmes statique de placement de données requirent une connaissance complète des statistiques sur les charges de travail tels que le temps d'exécution des services, et le taux d'accès.
Les algorithmes dynamique de placement de taches génèrent des schémas d'allocation en ligne pour s'adapter a différents modèles de charge de travail sans connaissance préalable des requêtes futurs. Les stratégies de placement de données dynamique mettent a jour le placement potentiellement a chaque requête.

3.2.1 Classification de placement et sélection de réplique

Pour mieux présenter notre état de l'art nous avons choisi de suivre la classification proposé par Grace et Manimegalai [21] qui propose dans leur article une étude sur les stratégies de placement et de sélection dynamique des répliques pour un environnement de grille de données.

Le classement est comme suite :



— Réplication de données statique : Le nombre de réplique et leur emplacement sont prédéfinie. L'avantage est la simplicité de l'implémentation et le cout minimum, mais inadéquat a un environnement variable.

— Réplication de données dynamique : La décision du nombre de réplique et de leur emplacement est prise en ligne (temps réel) selon la demande des utilisateurs, la capacité de stockage et la bande passante.

  — Distribuée : selon la topologie ou l'architecture de la grille de calcul, les réplications peuvent être crées par le noeud central et d'autres noeud sélectionnés.

  — Centralisée : selon l'importance des données, la qualité de service, équilibrage de charge et minimisé le nombre de réplique. les répliques sont crées uniquement dans le noeud central (head node).

3.2.2 Placement statique

Soit $D = \{d_1, .., d_n\}$, les ensembles de données, $S = \{S_1, .., S_l\}$: les data centers. La matrice de placement des ensembles de données D sur les data centers S est donnée par : $B = [\beta_{jk}]_{n \times l}$ ou,

$$\beta_{jk} = \begin{cases} 1 & \text{si } \mathrm{d}_j \text{ est sur } \mathrm{s}_k \\ 0 & \text{sinon.} \end{cases}$$

Le nombre de placement de données durant l'exécution de toutes les taches dans le système est donné par : $\Gamma(B)$. Donc l'objectif est de trouver la matrice B* optimal qui minimise $\Gamma(B)$.

Un algorithme de recherche exhaustif est un moyen direct pour rechercher la matrice de placement optimale. Il calcule toutes les matrices possibles de placement de données B, puis parcourt pour trouver le plus petit $\Gamma(B)$, à ce point la matrice de placement est la solution optimale. Cependant, la complexité de calcul de l'algorithme de recherche exhaustive est très élevée, ce qui se rapproche de (l^n). Dans un système de cloud computing distribué, le nombre de jeux de données n est si important que la complexité du calcul est insupportable pour le système, qui plus est, certaines conditions de contrainte, telles que la limitation de la capacité de stockage, font que la résolution du problème de placement est un problème NP-difficile. L'algorithme de recherche exhaustive n'est donc adéquat que lorsque le nombre de jeux de données est petit [47].

L'algorithme de Monte Carlo est basé sur la théorie des probabilités et les méthodes statistiques. Dans le cas du Big Data basé sur l'algorithme de Monte Carlo, un certain nombre de matrices B est généré aléatoirement, puis l'ordonnancement des données entre les datacenters est calculé sur chaque matrice d'échantillon B, pour obtenir la matrice de placement avec l'ordonnancement minimum des données. Par rapport à l'algorithme de recherche exhaustive, la complexité de calcul de l'algorithme de Monte Carlo est améliorée, cependant, les matrices B sont peu réparties dans l'espace de solution, l'efficacité de recherche de l'algorithme de Monte Carlo n'est toujours pas élevée [47].

3.2.3 Placement dynamique

Xu et al. [47] proposent un algorithme génétique et test l'efficacité de ce dernier dans le placement de données, en le comparant a d'autres algorithmes.

L'algorithme génétique est un algorithme adaptatif de recherche et d'optimisation basé sur la mécanique de la sélection naturelle et de la génétique naturelle. Une population de solutions candidates (appelées individus) à un problème d'optimisation évolue vers de meilleures solutions. Dans l'algorithme génétique, le degré d'adaptation de chaque individu à l'environnement est représenté par sa forme physique. Un individu avec une bonne forme physique a plus de chance de survivre. Dans chaque génération, la valeur de fitness de chaque individu dans la population est évaluée, les individus avec une valeur de fitness plus élevée sont sélectionnés stochastiquement à partir de la population actuelle, puis l'opérateur de croisement et de mutation sont manipulés pour former une nouvelle génération. La nouvelle génération de solutions candidates est ensuite utilisée dans la prochaine itération de l'algorithme.

Xu et al. [47] ont comparé leur algorithme a l'algorithme de recherche exhaustive et l'algorithme Monte carlo et ils ont remarqué qu'avec un nombre petit de *jeux de données* les trois algorithmes ont le même résultat et les matrices de placement de données optimales.

Avec l'augmentation du nombre de jeux de données, l'algorithme de recherche exhaustive est devenue irréalisable en raison de la complexité du calcul.

Il est coûteux en temps pour l'algorithme de Monte Carlo de trouver une matrice de placement de données optimale approximative lorsque le nombre d'ensembles de données ou de centres de données augmente dans une certaine mesure. Dans le cas d'un grand nombre d'ensembles de données, les caractéristiques de parallélisme inhérent et de convergence de l'algorithme génétique ont permis de trouver une meilleure solution dans un délai acceptable.

Amer et al. [4] proposent dans leur article des stratégies précédemment développé et combinées en une seule approche efficace influente pour un meilleur rendement du système de base de données distribuée (DDBS). DDBS est l'outil le plus demandé pour traiter les données volumineuse. les performances DDBS dépendent essentiel-

lement de la procédure avec laquelle DDBS est configuré (fragmenté et réparti sur tous les sites de son réseau).

Les données doivent être soigneusement divisées afin qu'une correspondance entre les données ciblées et les requêtes considérées soit satisfaite au maximum. la principale motivation pour l'allocation de données est de stocker des fragments de données sur plusieurs sites de manière à minimiser les coûts de transmission globaux au fur et à mesure qu'un ensemble de requêtes est en cours de traitement.

La fragmentation horizontale est l'opération de fragmenter les données afin de minimiser une fonction objectif donnée par l'équation (3).

La première équation est définie pour être utilisée pour mesurer les coûts encourus lorsque des requêtes de récupération distribuées sont en cours de traitement. L'équation (2) va mesurer les coûts engendrés à la suite de l'exécution de requêtes de mise à jour distribuées sur DDBS. Les coûts de transmission au total seraient donc calculés avec précision à l'aide de l'équation (3).

$$TC_1 = \sum_{j=1}^{m} \sum_{i=1}^{m} \sum_{k=1}^{q} (1 - X_{kj}) * (QRM_{kj}) * F_{size} * CMS_{ij} \quad (1)$$

$$TC_2 = \sum_{j=1}^{m} \sum_{i=1}^{m} \sum_{k=1}^{q} (1 - X_{kj}) * (QUM_{kj}) * F_{size} * CMS_{ij} \quad (2)$$

$$TC_{total} = TC_1 + TC_2 \quad (3)$$

ou CMS est la matrice de coûts entre sites (CSM) ou la matrice de coûts entre grappes (Cluster en anglais) de sites (CCM), F_{size} représente la taille du fragment considéré, et X_{ij} est une variable binaire destinée a indiquer l'allocation du fragment sur les sites.

QRM : fréquence de requête de retrait des données sur site, et

QUM : fréquence de requête de mise a jour (update) des sites.

3.3 Placement de tache (VMs)

Comme pour le placement de données, le placement de taches ou de VMs peut être classé en placement statique et dynamique.

3.3.1 Critère de comparaison

Xu et al. [46] présente des classifications basées sur une étude complète sur les algorithmes d'équilibrage de charge VM existants qui sont analysés et classés dans le but de fournir une vue d'ensemble de la caractéristique des algorithmes associés.

Treize algorithmes d'équilibrage de charge VM sont étudié. Il montre pour chaque algorithme les approches utilisées pour évaluer les algorithmes, ainsi que la configuration dans la quel s'est déroulé l'expérience et les améliorations des performances pour l'équilibrage de la charge des machines virtuelles.

Une classification des algorithmes selon les critère de : Allocation VM (Dynamique, statique), Uniformité de VM (Homogèn, Hétérogène), Type de ressource VM (CPU, mémoire .. etc.) et Stratégie d'optimisation (Heuristique, Méta-heuristique, Hybride) est effectué.

Les métriques de comparaison des algorithmes d'équilibrage de charge utilisé sont : Variance de charge et écart-type d'utilisation, Makespan (durée d'exécution, ou cycle de vie), Nombre d'hôtes surchargé, Pourcentage de toutes les machines virtuelles à placer dans l'hôte, Entropie d'équilibre quadratique, Débit, Écart-type des connexions, Niveau de déséquilibre moyen, Capacité-makespan, Score de déséquilibre, Écart-type de ressource restante, Nombre de migrations et Violations SLA.

Asha et al. [8] fournissent une étude sur les algorithmes d'équilibrage de charge statique et dynamique. L'objectif de l'étude est de consolider les méthodologies existantes pour l'équilibrage de charge dans le cloud.

3.3.2 Placement statique

Dans l'algorithme Round Robin, les processus sont partitionnés entre tous les processeurs de telle sorte que la charge de travail entre les processeurs est répartie également. De plus, le processus distinctif n'a pas le même temps de traitement. Parfois, certains des noeuds peuvent être chargés vigoureusement et d'autres sont légèrement chargés dans les serveurs Web où la requête http est de nature com-

parative et transmise de la même façon qu'un algorithme Round Robin est utilisé [8].

L'algorithme Min-Min commence par un arrangement de toutes les tâches non assignées. Premièrement, le temps d'exécution minimum pour toutes les tâches est trouvé. Les tâches ayant un temps d'exécution minimum sont d'abord choisies. Deuxièmement, le temps d'exécution pour toutes les autres tâches est repensé pour cette machine [8].

3.3.3 Placement dynamique

Asha et al. [8] ont aussi étudié les algorithmes suivant :

Equally spread current execution algorithm (Algorithme d'exécution courant étalé de manière égale), est un calcul dynamique d'ajustement de charge. Il décide de la priorité en vérifiant la durée d'exécution du processus. Cet algorithme disperse la charge de manière aléatoire en vérifiant d'abord la taille du processus et en échangeant ensuite la charge avec une machine virtuelle qui est légèrement chargée. L'équilibreur de charge répartit la charge sur des noeuds distinctifs [8].

Throttled Load Balancing Algorithm (Algorithme d'équilibrage de charge étranglé) fonctionne comme suit : le gestionnaire de tache dispose de l'emploi du temps de chaque machine virtuelle unique, et donc alloue la tache souhaité donné par le client à la machine sur le principe de la taille et de l'accessibilité de la machine, si aucune machine virtuelle n'est accessible pour exécute les taches, le gestionnaire de tache va mettre la demande en file d'attente [8].

Ant Colony Optimization Algorithm (Algorithme d'optimisation des colonies de fourmi) travaille sur le comportement des vraies fourmis. L'objectif principal de l'optimisation est de trouver le chemin optimisé de la source à la destination. Les fourmis tout en cherchant la nourriture lâchent des composants spéciaux appelés des phéromones. Sur la base de ces phéromones, les prochaines fourmis suivront le même chemin. L'intensité des phéromones est constituée de divers facteurs tels que la qualité de la nourriture, la distance alimentaire, etc. Les chemins qui consistent en

l'intensité de phéromone la plus élevée sont considérés comme étant de la distance la plus courte entre la source et la destination [8].

Honey bee foraging algorithm (algorithme de recherche des abeilles a miel) est basé sur le comportement des abeilles domestiques, lorsqu'une VM sous-chargée est affecté a une tâche, la liste des taches et la charge est mises à jour et informe les autres VM. La prise en compte de la priorité des taches améliore le débit [26, 27].

Pareto based fruit fly optimization algorithm (Algorithme d'optimisation pareto basé sur la mouche des fruits) fonctionne comme suit : D'abord, une heuristique initialise la population. Deuxièmement, un opérateur de réaffectation de ressources est utilisé pour générer des solutions. Troisièmement, un opérateur de recherche basé sur le chemin critique est utilisé pour améliorer la capacité d'exploitation [26, 28].

S. Jain [23] propose un algorithme génétique (PGA) de recherche heuristique qui vise à trouver les résultats optimisés basés sur le théorème "survie de l'individu le plus apte" théorie individuelle. L'objectif est de maintenir l'équilibre de charge sur la machine virtuelle et également de réduire le temps d'exécution. Une fois les taches planifiées en fonction de leur arrivés sur le cloud, elles sont allouées aux neouds pour une exécution ultérieurs en utilisant l'algorithme de Djikstra. L'algorithme génétique proposé est utilisé pour l'ordonnancement des VMs. La 1ère génération de l'algorithme est obtenu du résultat de l'algorithme de djikstra.

Multi objective Scheduling cuckoo algorithm (algorithme d'ordonnancement multi-objectif du coucou) reproduit le comportement de reproduction des coucous, où chaque individu recherche le nid le plus approprié pour pondre un oeuf (solution de compromis) afin de maximiser le taux de survie de l'oeuf. La théorie des ensembles flous est utilisée pour créer le domaine de recherche des appartenances floues qui est constitué de toutes les solutions de compromis possibles. L'algorithme recherche la meilleure solution de compromis dans le domaine de recherche floue en réglant simultanément les variables de la frontière de conception floue. L'ajustement des variables de conception floue élimine l'exigence d'expertise nécessaire pour définir ces variables [26].

3.4 Analyse des articles étudié

Nous avons présenté dans la partie précédente quelques travaux qui proposent de résoudre les problèmes liés à l'équilibrage de charge dans le cloud computing. Quelques approches semblent être pertinentes et assurent un niveau de sécurité acceptable mais qui reste insuffisant.
Par exemple l'algorithme de recherche exhaustive et l'algorithme de Monte Carlo sont très faible dans le placement de donnée a partir d'une certaine taille de jeu de donnée. Quand a l'algorithme génétique [47], il présente de meilleur résultat mais insuffisante, la matrice optimale de placement de données trouver ne rend pas le nombre de données planifiées entre les data centers aussi petit que possible. L'auteur peut améliorer sont algorithme en prenant en compte l'impact de l'historique d'accès aux données dans sont algorithme.
Les points faible relevé pour les autres algorithmes sont, pour l'algorithme Round Robin l'utilisation de trop de ressource, algorithme Min Min provoque la famine, pour Equally Spread les machines virtuel sont affecté aléatoirement, par exemple on peut les affectés selon le niveau de charge des vm et le poids estimé des taches.

3.5 Conclusion

Nous avons traité dans ce chapitre les solutions apporté dans la littérature au problème d'équilibrage de charge dans le cloud.

Chapitre 4

Proposition

4.1 Problématique

Avant de détailler notre proposition, nous allons dans cette section présenter la problématique de notre thème.

Les données dans le cloud sont stocké sur des data centers qui sont localisé sur différents lieux (Cela peut aller de la ville au continent). Il est évident que le fournisseur cloud a intérêt a rapprocher les données des clients qui les sollicites, tout en garantissant un bon niveau de service, ce qui implique un bon équilibrage de charge, c'est a dire qu'il faut que les clusters ne soit pas surchargé en données trop demandé, donc il faut répartir les données sur les clusters des data centers. Cela va réduire les temps des communications et donc de réponse.
Les questions a se poser sont répliquer ou pas ? quoi répliquer ? quand répliquer ? ou placer ?

4.2 Notre proposition

Dans ce chapitre nous allons continuer le travail fait par Frissou et Houari [18] pour prendre en charge l'état de saturation du cluster dans la prise de décision de réplication. Pour y parvenir nous allons nous basé sur le protocole X-ON/X-OFF [1] qu'on trouve dans les réseaux.

4.3 Protocole X-ON/X-OFF

Xon/Xoff est un protocole logiciel qui permet une transmission série sur 3 fils (le plus souvent en RS232 [1]) bi-directionnelle, c'est une sorte de protocole logiciel.

Les signaux X-on et X-off sont basés sur les caractères spéciaux ascii. Pendant le transfert de données, le périphérique qui reçoit (ordinateur, terminal, imprimante, modem, ...) récupère simplement les données transmises dans une mémoire tampon avant de les utiliser. Lorsque ce tampon est rempli à 80%, il demande d'interrompre la transmission en renvoyant le code ascii 19 (x-off). Quand le tampon repasse à 50% de remplissage, il renvoit le code asci 17 (X-on) pour reprendre la transmission. Ces codes sont renvoyés sur la broche transmit data du connecteur [1].

4.4 Architecture du Cloud

Nous avons repris l'architecture du Cloud proposé par Frissou et Houari [18]. L'architecture du Cloud est composée de clusters disjoints de datacenters (Figure 4.1, 4.2). Chaque cluster est supervisé par un clusterhead (CH), et est composé des quorums [2] de lecture et des quorums d'écriture.

Quorum d'écriture : Lors d'une requête d'écriture, l'opération est effectué si la données est réellement écrite chez un sous-ensemble de nœuds appelé quorum d'écriture.

Quorum de lecture : Lors d'une requête de lecture, l'opération est effectué si la donnée est récupérée cohérente d'un sous-ensemble de nœuds appelé quorum de lecture.

1. RS232 : est une norme standardisant une voie de communication de type série
2. Le quorum est le nombre de membres présents exigé dans une assemblée délibérante pour que le vote soit valable (Larousse)

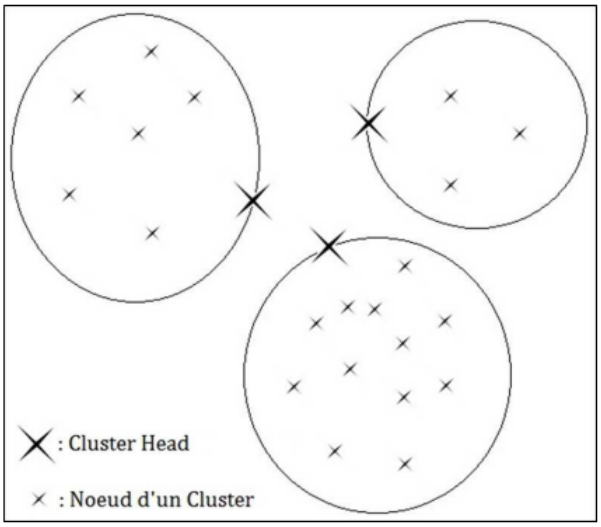


FIGURE 4.1 – Architecture du Cloud proposée. [6]

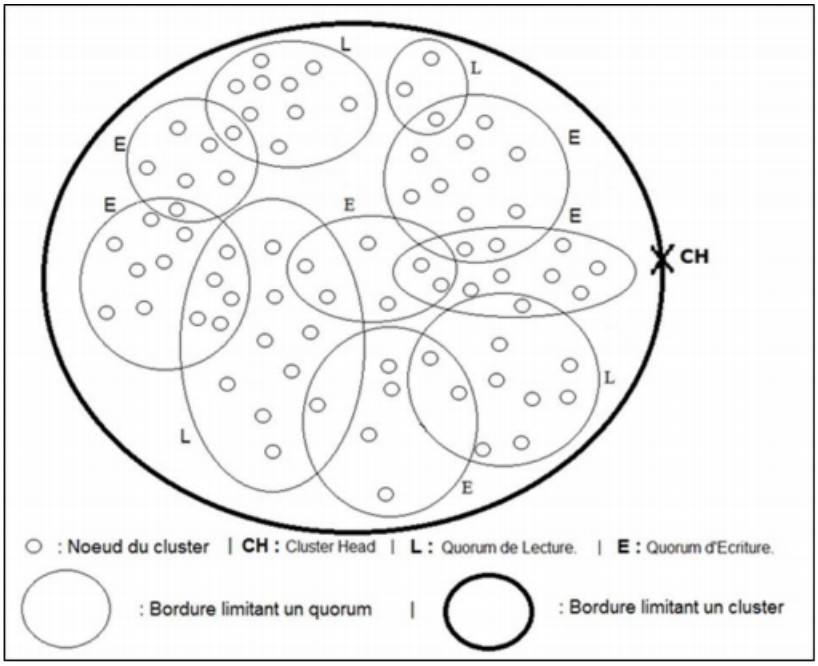


FIGURE 4.2 – Zoom sur un cluster [6]

4.5 Structuration des données

Les données sont stocké dans un tableau dynamique de deux dimension l*c, tel que l représente le nombre de lignes qui augmente suivant les requêtes de stockage (ReqS) ou de réplication (ReqR), et c le nombre de colonnes qui reste fixe (mais dont la taille est dynamique). La colonne du tableau comporte 3 éléments : le premier concerne l'identifiant de la donnée, le deuxième est sa valeur, le troisième son estampille. Id-Donnée est l'identifiant de la donnée, Val-Donnée, la valeur de cette donnée, estampille, son estampille, voir Table 4.1.

/	Colonne 1		
/	id-donnée	val-donnée	estampille
Ligne 1	data 1	true	654931

Table 4.1 – Exemple de tableau représentant une donné

Chaque CH conserve l'intégralité des requêtes qui concernent les données. L'historique est sauvegardé comme un tableau de deux dimension i*j, tel que le nombre de colonne est fixe, et le nombre de ligne augmente avec chaque requête de lecteur (ReqL) ou d'écriture (ReqE). La colonne est composé de 5 éléments, où ID-Requête, l'identifiant de la requête, Date-Requête, la date et l'heure ou cette requête a été généré la première fois, Type-requête, le type de cette requête (requête de stockage, de lecture, ou d'écriture), ID-Donnée, l'identifiant de la donnée sollicité et Loc-noeud la localisation du noeud le plus proche sollicité par le client qui introduit la nouvelle donné, voir Table 4.2.

Les dates sont enregistrer suivant le format : jj/mm/aaaa hh :mn :ss. Tel que : jj représente le jour, mm le mois, aaaa l'année, hh l'heure, mn les minutes, ss les secondes.

/	Colonne 1				
/	ID-Requête	Date-Requête	Type-Reqiête	ID-Donnée	Loc-Noeud
Ligne1	Req1	09/06/2018 15 :20 :46	ReqL	data1	noeud001

TABLE 4.2 – Exemple de structure pour la sauvegarde de l'historique.

4.6 Construction des quorums

Les quorums sont construit de façon statique, tel que (FIGURE 4.3) :

— L'intersection entre un quorum de lecture et un quorum d'écriture n'est jamais vide.
— Dans chaque intersection, il y a un noeud spécial appelé coordonateur de quorum.
— Chaque quorum de lecture ou d'écriture est muni d'un coordonateur.
— Chaque quorum de lecture ou d'écriture contient toutes les données au moins une fois.

Remarques

— Chaque noeud d'un quorum connait son coordonateur.
— Chaque coordonateur connait les nouds de son quorum ainsi que leurs emplacements géographiques.
— Chaque CH connait tous les coordonateurs de son cluster.

L'objectif étant de garantir la cohérence des données, en d'autres termes si une donnée est lue, on garantit que c'est la plus récente.

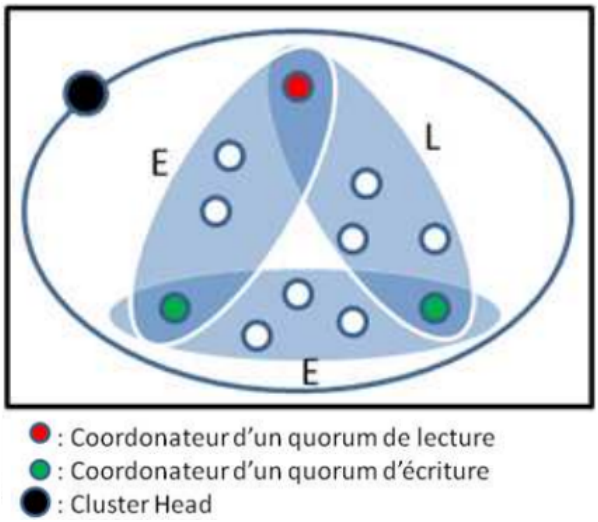


FIGURE 4.3 – Zoom sur l'intersection de quorums dans un Cluster [18]

4.7 Protocole de stockage

1. Quand un client introduit une nouvelle donnée x, il sollicite le noeud le plus proche en envoyant ReqS (Id-Donnée,Val-Donnée), où ReqS représente une requête de stockage.
2. Ce noeud redirige cette requête ReqS (ID-Donnée,Val-Donnée,Date, LocClient,Loc-noeud) vers CH.
3. CH envoi ReqS (ID-Donnée,Val-Donnée,Date,Loc-noeud) vers les coordonnateurs des quorums et vers les autres CH.
4. En recevant ce message, les CHs diffusent ce message vers les coordonnateurs de quorums de leurs clusters. Chaque coordonnateur désigne le noeud le plus proche géographiquement de son quorum pour effectuer la sauvegarde.
5. Le noeud désigné reçoit la requête de stockage, stocke la donnée et renvoi un ACK à son coordonnateur
6. Le coordonateur attend un ACK du noeud désigné sinon désigne un autre noeud géographiquement proche du client et exécute de nouveau le point (6).

7. Le coordonnateur reçoit un ACK du noeud désigné le transmet au CH.
8. CH, après avoir reçu au moins un ACK, il le renvoi au client.
9. CH exécute une sauvegarde dans l'historique (X, 09/06/2018, ReqS, data1, neoudRax).

4.8 Protocole de lecture

1. Le client envoi une requête de lecture ReqL (Lec, ID-Donnée, estampille, ID-client) au noeud le plus proche, tel que : Lec : désigne une requête de lecture.
2. A la réception de la ReqL :
   2.1. Si le noeud récepteur n'est pas un coordonnateur, alors il redirige la ReqL vers son coordonnateur et passe au point (2.2).
   2.2. Si le noeud récepteur est un coordonnateur, alors il exécute l'un des points 2.3, 2.4.
   2.3. S'il est coordonnateur d'un quorum de lecture, il redirige la ReqL vers le CH et attend une confirmation.
   2.4. S'il coordonne un quorum d'écriture et n'appartient à aucun quorum de lecture, il redirige la ReqL vers le CH.
3. Lorsque CH reçoit la ReqL :
   3.1. (Dans le cas du point 2.3) : s'il n'y a pas de requête d'écriture en cours sur la donnée sollicité, CH envoie un ReqL + ACK au coordonnateur sinon il attend la fin de l'écriture pour l'envoyer.
4. (Dans le cas du point 2.4) : CH désigne un quorum de lecture proche, puis envoi la ReqL accompagnée d'un ACK vers sont coordonnateur. A la réception de l'ACK + ReqL le coordonnateur diffuse la ReqL vers les noeuds de son quorum.
5. Chaque noeud, y compris le coordonnateur, vérifie s'il dispose de la donnée sollicité.
   5.1. Si le noeud dispose de la donnée, il l'envoie vers le coordonnateur + un ACK, la Date de la donnée (Date-Donnée).
   5.2. Sinon, il envoi un NACK.

6. Le coordonnateur, lorsqu'il reçoit la totalité des réponses des noeuds de son quorum (ACK et NACK), il renvoi au CH les informations concernant la donnée avec la date la plus récente.
7. CH répond au coordonnateur avec une requête d'affichage, pour indiquer au noeud ayant la date la plus récente et la localisation la plus proche du client de transmettre la donnée au client.
8. CH exécute le Protocole de Sauvegarde de l'historique.

9. Le Protocole de Réplication.

4.9 Protocole d'écriture

1. Le client envoi une requête d'écriture ReqE (Ecr, ID-Donnée, Val-Donnée, ID-client) au noeud le plus proche, tel que, Ecr : est une requête d'écriture.
2. Lors de la réception de la ReqE :
   2.1. Si le noeud récepteur n'est pas un coordonateur, alors on passe aux points 2.1.1 et 2.1.2.
   2.1.1. Si le noeud récepteur appartient à un quorum de lecture, alors il redirige la ReqE vers son coordonateur et passe au point 2.2.1.
   2.1.2. Si le noeud récepteur appartient à un quorum d'écriture, alors il exécute l'écriture s'il dispose de la donnée, puis redirige la ReqE suivie du résultat (ACK ou NACK) vers son coordonateur.
   2.2. Si le noeud récepteur est un coordonateur, alors il exécute l'un des points 2.2.1 ou 2.2.2.
   2.2.1. Si le noeud récepteur coordonne un quorum de lecture, il redirige la ReqE vers le coordonateur du quorum d'écriture et passe au point 2.2.2.
   2.2.2. Si le noeud récepteur coordonne un quorum d'écriture :

   (a) Il redirige la ReqE vers CH et vers les noeuds de son quorum, sauf le noeud récepteur.

(b) Chaque noeud, y compris le coordonnateur, exécutent l'écriture (la mise à jour) s'il dispose de la donnée sollicité et envoi un ACK. Sinon il envoi un NACK.

(c) En parallèle, CH diffuse un message de supervision vers les coordonnateurs de quorums de son cluster (sauf celui qui a envoyé la ReqE) et vers les autres CHs.

(d) A la réception de ce message, les CHs le diffusent vers tous les coordonnateurs et quorums de leur cluster.

(e) A la réception de ce message, chaque coordonnateur le diffuse aux noeuds de son quorum et tous les noeuds exécutent l'écriture.

3. CH exécute le Protocole de Sauvegarde de l'historique.

4. Exécute le protocole de réplication

Remarques

— Si un noeud dispose de la donnée sollicité vérifie si la date de la nouvelle requête est supérieure à celle qui est stockée avant d'exécuteur l'écriture.

— Le coordonnateur, s'il réussit à écrire la donnée envoi directement un ACK au CH, sinon il l'envoi au premier ACK reçu.

— Si un client ne reçoit pas de confirmation à sa requête après un certain délai, il sollicite un autre noeud qui lui est proche ainsi le processus est recommencé.

4.10 Notre Contribution : Protocole de réplication

Nous nous somme basé sur le protocole X-ON/X-OFF qu'on trouve dans les réseaux et nous l'avons adapté pour répondre a notre problématique, de sorte que la réplication n'arrive que lorsque la mémoire est disponible et dans le cas contraire attendre jusqu'à la libération de la mémoire.

La proposition repose sur le calcul de deux paramètres :
La popularité de la donnée et la charge du cluster.

— Popularité de la donnée : Une donnée est dite populaire si le nombre de requêtes de lecteur/écriture/stockage des 30 derniers jours est supérieur ou égale a 30.

— Charge du cluster : Taux d'occupation de la mémoire des membres du cluster. Si la taille de la mémoire des membres du cluster réservé aux données représente 100%, alors la charge au niveau d'un membre représente le taux d'occupation de sa mémoire.
Le CH récupère les taux d'occupation de tous ses membres et calcul une moyenne. Ce taux est calculé à la demande, en d'autres termes pour décider s'il faut répliquer ou pas.

Ce protocole est exécuté au niveau du CH lors de la réception d'une ReqL ou ReqE.

1. A la réception d'une requête de lecteur ou d'écriture, elle est enregistré dans l'historique au niveau du CH.

2. Le CH vérifie si la donnée existe au niveau du cluster grâce a sa table d'historique. Si elle existe, il Exécute 2.1, Sinon 2.2.

2.1. Il calcul la popularité de la donnée. Si donnée populaire, il exécute 2.1.1, sinon 2.1.2.

2.1.1. Envoi/Écriture de la donnée. FIN.

2.1.2. Envoi/Écriture de la donnée, puis suppression de la donnée du cluster après un délai dû à la propagation de la donnée. FIN.

2.2. La popularité de la donnée est aussi calculé. Si donnée populaire, il exécute 2.2.1, sinon 2.2.2.

2.2.1. Il calcul la charge du Cluster. Si la charge est supérieur a 80% (surcharger), il exécute 2.2.1.1, sinon 2.2.1.2.

2.2.1.1. Envoi/Écriture de la donnée. Quand la charge du cluster diminue (Qu'elle passe en dessous de 50%), la donnée est répliquer en exécutant le protocole de stockage. FIN.

2.2.1.2. Il réplique en exécutant le protocole de stockage, et Envoi/Écriture de la donnée. FIN.

2.2.2. Envoi/Écriture de la donnée. FIN.

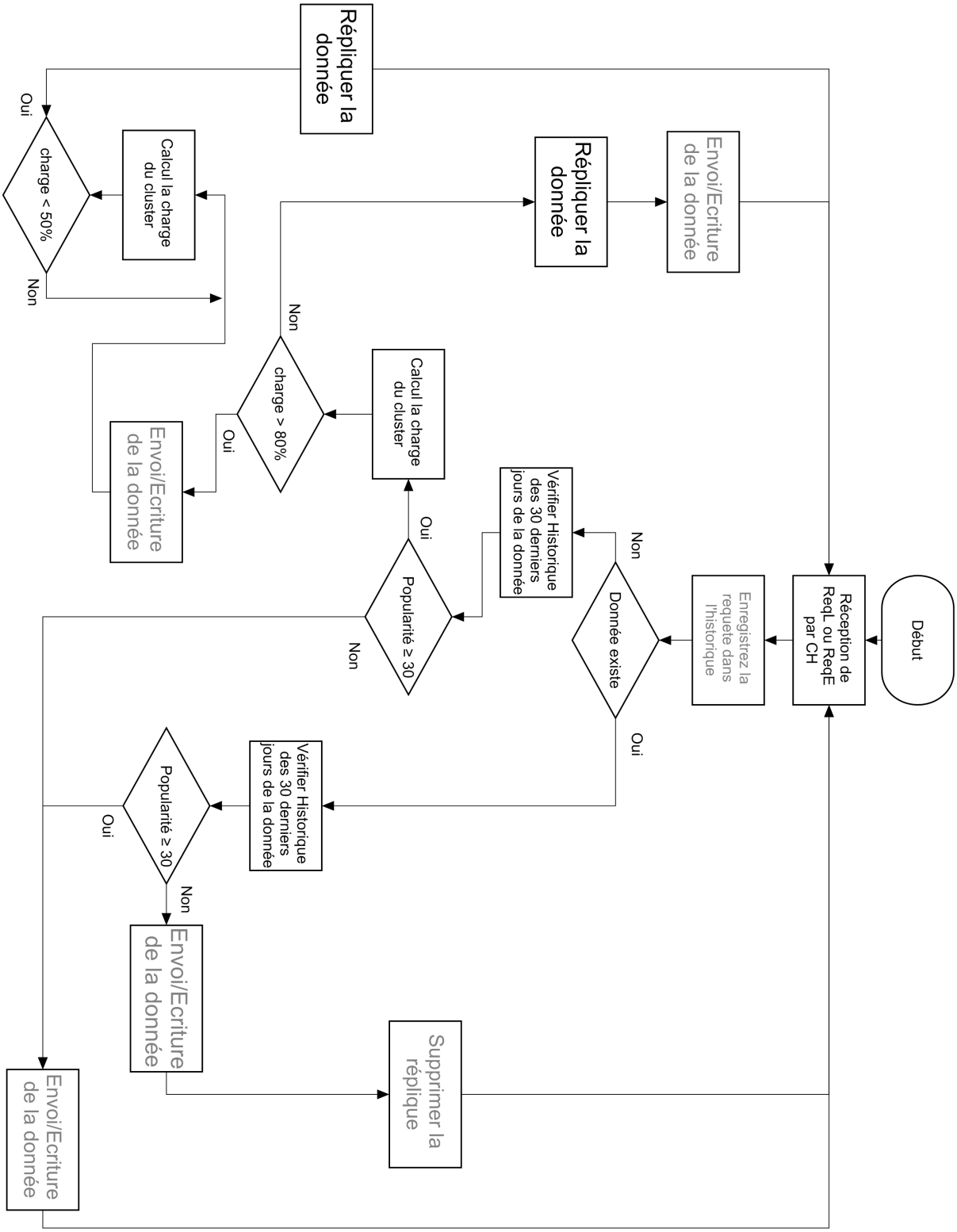


FIGURE 4.4 – Diagramme du protocole proposé

4.11 Exemple concret

Nous allons dérouler dans cette section notre algorithme pour montrer son efficacité et son fonctionnement avec un exemple concret de donnée.

Nous avons pour chaque donnée indiquer si elle existe au niveau du cluster et la popularité après l'arrivé de la donnée sur le CH.

Les donnée d2 et d3 n'existe pas au niveau du cluster et leur popularité est égale a 30, donc elle sont répliquer.

La donnée d4 existe au niveau du cluster, mais sa popularité est inférieur a 30, donc elle est supprimer.

On remarque que la donnée d3 n'est pas directement répliquer, elle est répliquer après que la charge mémoire soit passé en dessous de 50%, donc ultérieurement.

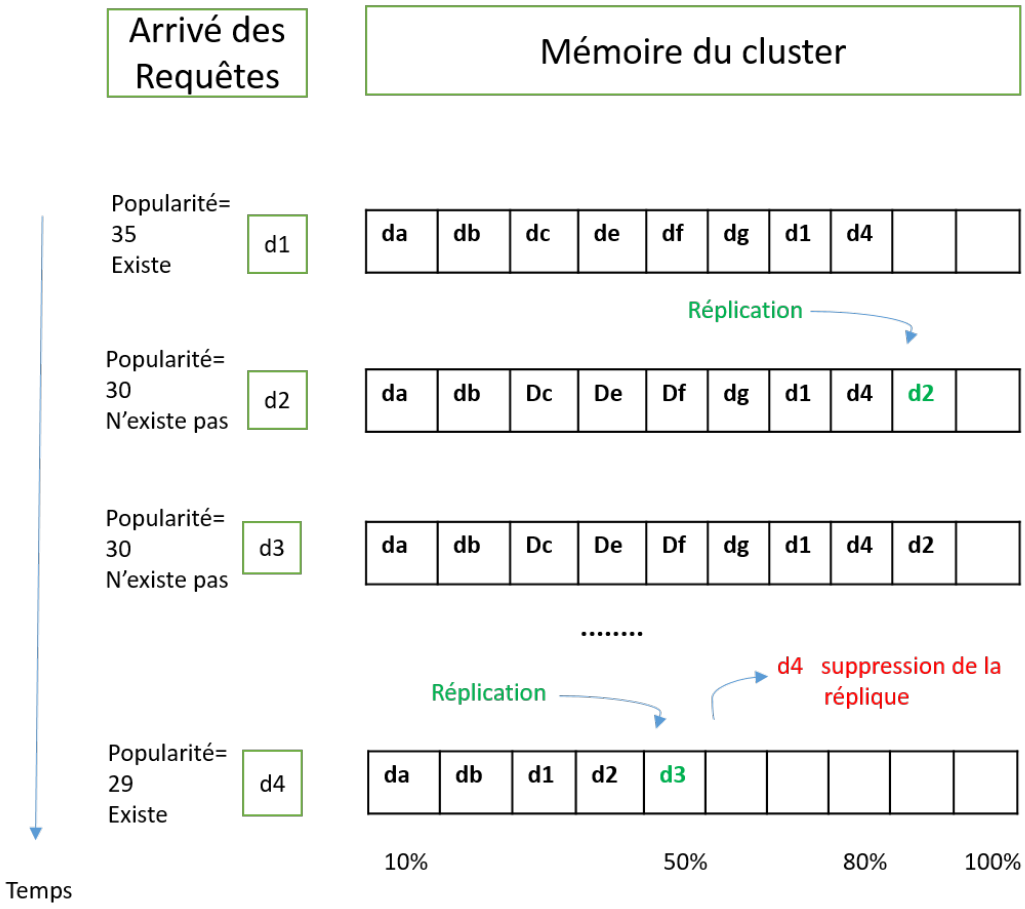


FIGURE 4.5 – Exemple de fonctionnement de notre protocole

4.12 Analyse

La solution proposé est basé sur un modèle simple et cohérent, ce qui la rend facile a implémenter par rapport aux autres algorithmes vus dans l'état de l'art. De plus, la réplication, comme modèle d'assurer l'équilibrage de charge, permet d'éviter les algorithmes qui assure l'équilibrage de charge par l'affectation initiale des données. Notre algorithme fini toujours par répliquer la donnée après un délai, contrairement a l'algorithme Min-Min qui peut causer la famine.

Notre solution :

— Prend en considération la popularité des données et l'état de charge du cluster dans sa prise de décision pour la réplication.
— Ne garde dans sa mémoire que les répliques des données trop sollicité.
— Supprime les répliques des données les moins sollicité.

On peut citer comme inconvénient :

— La boucle présente dans l'algorithme en cas de cluster surchargé et la vérification a chaque fois de la charge du cluster peut augmenter l'utilisation du CPU du CH.

4.13 Conclusion

Dans cette section nous avons présenté l'architecture du cloud sur la quel on s'est basé, notre contribution, et un exemple de déroulement de notre algorithme. Notre solution par rapport aux approches cité à la section 2.4 est une approche dynamique, centralisé et receveur-initiative.

Conclusion et perspectives

Le Cloud Computing représente une révolution dans la manière d'organiser, de gérer et de distribuer des ressources informatiques. Sa définition opérationnelle annonce un modèle informatique qui permet un accès facile et à la demande, par le réseau, à un ensemble partagé de ressources informatiques configurables et depuis n'importe quel appareil disposant d'un navigateur et d'une connexion internet. Il peut s'agir de serveurs, de stockage, d'applications ou de services, rapidement provisionnés et libérés par un minimum d'efforts de gestion. Le Cloud Computing se différencie des autres solutions d'utilisation de ressources à distance par six caractéristiques essentielles. Une infrastructure ainsi équipée offre alors trois modèles de services et quatre modèles de déploiements possibles, privé, communautaire, public ou hybride.

Dans ce mémoire, nous avons essayé de proposé un algorithme de réplication basé sur le principe du protocole X-ON/X-OFF pour répondre a la problématique d'équilibrage de charge dans le cloud, qui va éviter la surcharge des Clusters, et qui va répliquer les données au plus près des clients afin d'améliorer le temps de réponse.

Notre solution prend en considération la popularité des données et l'état de charge du cluster dans sa prise de décision pour la réplication, et ne garde dans sa mémoire que les répliques des données trop sollicité, et supprime les répliques des données les moins sollicité.

Sur la base du travail réalisé nous dressons les perspectives suivantes :

— Il serait intéressant d'implémenter l'algorithme et de simuler la proposition sur un environnement Cloud Computing réel.

— Améliorer l'algorithme pour déléguer aux autres cluster la réplication des données quand elles sont surchargé.

Bibliographie

[1] X-on/x-off. http://bp.perso.eisti.fr/doc/reseaux/cours6.html, (Consulté le 25 Juin 2018).

[2] Historiques du cloud computing. https://www.universalis.fr/encyclopedie/cloud-computing-informatique-dans-les-nuages/2-quelques-jalons-historiques-du-cloud-computing/, (Consulté le 28 Janvier 2018).

[3] Grid 5000. http://www.grid5000.fr, (Consulté le 2 mai 2018).

[4] A. A.Amer, A. A. Sewisy, and T. M. Elgendy. *An optimized approach for simultaneous horizontal data fragmentation and allocation in Distributed Database Systems.* heliyon journal, vol. 3, no 12, DOI : 10.1016/j.heliyon.2017.e00487, 2017.

[5] H. I. Abdallaha, A. A. Amer, and H. Mathkour. *Performance optimality enhancement algorithm in DDBS (POEA).* computers in human behavior Journal, vol. 30, pp. 419-426, 2014.

[6] S. Ait Ali and M. Lahdir. *Gestion de la Cohérence des données Répliquées dans le Cloud.* Université de Béjaia, Mémoire de master, 2016.

[7] S. V. Ambade and P. R. Deshpande. *Heterogeneity-based files placement in Hadoop cluster.* International Conference on Computational Intelligence and Communication Networks (CICN), Jabalpur, India, DOI : 10.1109/CICN.2015.325, 2015.

[8] V. Asha, Bharath Kumar, and V. Girish. *Load Balancing in Cloud Computing*. International Journal of Recent Trends in Engineering and Research, vol. 4, no 3, 2018.

[9] B. Azria and J. Cherki. *L'impact du Cloud Computing dans les PME*. Mémoire ingénieur, Ecole supérieure de génie informatique ESGI - Paris, 2014.

[10] B. G. Babu, T. P. Shabeera, and M. K. SD. *Dynamic Colocation Algorithm For Hadoop*. International Conference on Advances in Computing, Communications and Informatics (ICACCI), New Delhi, India, DOI : 10.1109/ICACCI.2014.6968384, 2014.

[11] Y. Belabbas. *Modèle d'équilibrage de charge pour les grilles de calcul*. Revue Africaine de la Recherche en Informatique et Mathématiques Appliquées, 2007.

[12] N. Berkani and S. Moussaoui. *La sécurité des données dans le Cloud Computing*. Université de Béjaia, Mémoire de master, 2016.

[13] R. B. Bohn, J. Messina, F. Liu, J. Tong, and J. Mao. *NIST cloud computing reference architecture*. In Services (SERVICES), IEEE World Congress on pp. 594-596, 2011.

[14] R. Buyya, S.k. Garg, and R.N. Calheiros. *SLA-oriented resource provisioning for cloud computing : Challenges, architecture, and solutions*. Proceedings of the International Conference on Cloud and Service Computing (CSC),Hong Kong, China, pp. 1-10, 2011.

[15] P.P. Codo. *Conception d'Une Solution de Cloud Computing Privé Basée sur un Algorithme de Supervision Distribué : Application aux Services IAAS*. Mémoire, Ecole Polytechnique d'Abomey-Calavi (EPAC) - Bénin, 2012.

[16] Définition du Cloud Computing. `http://fr.wikipedia.org/wiki/Cloud_computing`, (Consulté le 17 Mars 2018).

[17] A.A.Y. Elwessabi. *Une approche basée agent mobile pour le cloud computing*. Mémoire de magister, Université HADJ LAKHDAR - BATNA, 2014.

[18] Y. Frissou and N. Houari. *Réplication des données dans le Cloud computing*. Université de Béjaia, Mémoire de master, 2016.

[19] K. Gai and S. Li. *Towards Cloud Computing : A Literature Review on Cloud Computing and Its Development Trends*. In 2012 Fourth International Conference on Multimedia Information Networking and Security (MINES), pp. 142-146. IEEE, 2012.

[20] A. Gara, M. A. Blumrich, D. Chen, G. T. Chiu, P. Coteus, M. E. Giampapa, R. A. Haring, P. Heidelberger, D. Hoenicke, G. V. Kopcsay, T. A. Liebsch, M. Ohmacht, B. D. Steinmacher-Burow, T. Takken, and P. Vranas. *Overview of the Blue Gene/L system architecture*. IBM Journal of research and development, vol. 49, no 2.3, pp. 195-212, 2005.

[21] R. K. Grace and R. Manimegalai. *Dynamic replica placement and selection strategies in data grids-a comprehensive survey*. Journal of Parallel and Distributed Computing, vol. 74, no 2, pp. 2099-2108, 2014.

[22] An introduction to the infiniband architecture. `http://gridbus.csse.unimelb.edu.au/~raj/superstorage/chap42.pdf`, (Consulté le 25 avril 2018).

[23] S. Jain. *Task Scheduling in Cloud Computing using Genetic Algorithm*. International Journal of Computer Science Engineering and Information Technology Research, vol. 6, no 4, 2016.

[24] M. Jalgaonkar and A. Kanojia. *Adoption of Cloud Computing in Distance Learning*. International journal of Advanced Trends in Computer Science and Engineering, vol. 2, no 1, pp. 17-20, 2013.

[25] M. Juganaru-Mathieu. Cloud computing. `http://www.emse.fr/~mathieu/pub/CGC/cours_CC.pdf`, (Consulté le 26 Janvier 2018).

[26] Pooja R. Kathalkar1 and A. V. Deorankar. *A Review on different load balancing Algorithm in cloud computing*. International Research Journal of Engineering and Technology (IRJET), vol. 5, no 2, 2018.

[27] P. V. Krishna. *Honey bee behavior inspired load balancing of tasks in cloud computing environments*. Applied Soft Computing, vol. 13, no 5, pp. 2292-2303, 2013.

[28] X. l. Zheng and L. Wang. *A pareto based fruit fly optimization algorithm for task scheduling and resource allocation in cloud computing environment*. Evolutionary Computation (CEC), pp. 3393-3400, 2016.

[29] T. Lamiel. *Contributions aux techniques d'ordonnancement sur plates-formes parallèles ou distribuées*. Thèse doctorat, Université de Franche-Comté - Besançon, France, 2012.

[30] S. Lanani. *Une approche BPM (Business Process Managment) par composition d'applications dans le cloud computing*. Mémoire de magister, Université Mohamed Khider de Biskra, 2015.

[31] I. Laribi. *La mise en place de la solution Openstack*. Mémoire master, Université Abou Bekr Belkaid - Tlemcen, 2014.

[32] C. W. Lee, K. Y. Hsieh, S. Y. Hsieh, and H. C. Hsiao. *A Dynamic Data Placement Strategy for Hadoop in Heterogeneous Environments*. Big Data Research Journal, vol. 1, pp. 14-22, DOI : 10.1016/j.bdr.2014.07.002, 2014.

[33] M. Meddeber and Y. Belabbes. *Équilibrage de charge pour les grilles de calcul : classe des tâches dépendantes et indépendantes*. Proceedings of the 2nd Conférence Internationale sur l'Informatique et ses Applications (CIIA'09) - Saida, Algeria, 2009.

[34] P. Mell and T. Grance. *The NIST Definition of Cloud Computing*. 2011.

[35] S.P. Mirashe and N.V. Kalyankar. *Cloud Computing*. Journal of computing, vol. 2, no 3, pp. 78-82, 2010.

[36] H. Miyazaki, Y. Kusano, N. Shinjou, F. Shoji, M. Yokokawa, and T. Watanabe. *Overview of the K computer System*. FUJITSU SCIENTIFIC AND TECHNICAL JOURNAL, vol. 48, no 3, pp. 302-309, 2012.

[37] L.F. Noumsi. *Etude et mise en place d'une solution "cloud computing" privée dans une entreprise moderne : cas de CAMTEL*. Mémoire ingénieur, Ecole nationale supérieure des postes et télécommunications - Paris, 2012.

[38] S. Sangavi, A. Vanmathi, R. Gayathri, R. Rajua, P. Victer Paula, and P. Dhavachelvan. *An Enhanced DACHE model for the MapReduce Environment*. Procedia Computer Science journal, vol. 50, pp. 579-584, DOI : 10.1016/j.procs.2015.04.087, 2015.

[39] A. Sellami. *Une nouvelle Méthode multi-objective pour l'ordonnancement des workflows dans le cloud computing*. Université de Béjaia, Mémoire de master, 2016.

[40] A. A. Sewisy, A. A. Amer, and H. I. Abdalla. *A Novel Query-Driven Clustering-Based Technique for Vertical Fragmentation and Allocation in Distributed Database Systems*. International Journal on Semantic Web and Information Systems (IJSWIS), vol. 13, no 2, pp. 27-54, 2017.

[41] V. Ubarhande, A. M. Popescu, and H. Gonzalez-Velez. *Novel Data-Distribution Technique for Hadoop in Heterogeneous Cloud Environments*. Ninth International Conference, IEEE, Blumenau, Brazil, DOI : 10.1109/CISIS.2015.37, 2015.

[42] S. D. Vipulkumar and H. Somani. *A Survey on Data Placement in Heterogeneous Cloud Environment for Big Data*. International Journal of Engineering Development and Research, vol. 4, no 4, 2016.

[43] Wang, Tingting, and al. *Load Balancing Task Scheduling Based on Genetic Algorithm in Cloud Computing*. Secure Computing (DASC), pp. 146-152, 2014.

[44] L. Wang, J. Tao, and M. Kunze. *Scientific Cloud Computing : Early Definition and Experience*. 10th IEEE International Conference on High Performance Computing and Communications, 2008.

[45] J. X. Wu, C. S. Zhang, B. Zhang, and P. Wang. *A New Data-Grouping Aware Dynamic Data Placement Method that Take into Account Jobs Execute Frequency for Hadoop*. Microprocessors and Microsystems journal, vol. 47, pp. 161-169, DOI :10.1016/j.micpro.2016.07.011, 2016.

[46] M. Xu, W. Tian, and R. Buyya. *A survey on load balancing algorithms for virtual machines placement in cloud computing*. Concurrency and Computation : Practice and Experience journal, vol. 29, no 12, 2017.

[47] Q. Xu, Z. Xu, and T. Wang. *A Data-Placement Strategy Based on Genetic Algorithm in Cloud Computing*. International Journal of Intelligence Science, vol. 5, no 03, pp. 145-157, 2015.

RÉSUMÉ

L'équilibrage de charge dans le cloud consiste à distribuer le travail à faire sur un ensemble de nœuds de manière à ce qu'aucun nœud ne soit surchargé ou sous-chargé, c'est à dire que le travail est affecté aux nœuds selon leur capacité, et de telle sorte que tout le monde participe à sa réalisation. Dans le cas du placement de donnée, on assure l'équilibrage de charge en distribuant les données sur l'ensembles des clusters et en répliquant le plus près de l'utilisateur, afin de garantir un bon niveau de service.

Dans ce mémoire, une introduction au cloud, et les techniques d'équilibrage de charge dans les systèmes distribués est donnée. Un état de l'art des solutions disponible dans la littérature est effectué. Enfin, la contribution consiste en un algorithme d'équilibrage de charge par la réplication basé sur le principe du protocole X-ON/X-OFF.

Mots clés : Cloud Computing, Équilibrage de charge, réplication, grappe

ABSTRACT

Load balancing in the cloud consists of distributing the work to be done on a set of nodes so that no nodes are overloaded or underloaded, ie the job is assigned to the nodes according to their capacity, and so that everyone participates in its realization. In the case of data placement, load balancing is provided by distributing data across the set of clusters and replicating closer to the user to ensure a good level of service.

In this document, an introduction to the cloud, and load balancing techniques in distributed systems is given. A state of the art of the solutions available in the literature is performed. Finally, the contribution consists of a replication-based load balancing algorithm based on the X-ON / X-OFF protocol.

Key words : Cloud Computing, Load Balancing, replication, cluster