RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

MINISTÈRE DE L'ENSEIGNEMENTS SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ DR MOULAY TAHAR – SAIDA

FACULTÉ DE TECHNOLOGIE

DÉPARTEMENT DE L'INFORMATIQUE

Mémoire Présenté pour l'obtention du Diplôme De Master Académique

Option : Réseaux Informatique et Systèmes Répartis

Une stratégie d'ordonnancement et de réplication de données dans les environnements de Fog/Cloud Computing.

Présenté par :

AZZAZE NACIRA HANANE

BOUKHECHA IKRAM

Encadré par :

Dr. LIMAME SAIDE

Promotion : Septembre 2021

Résumé

Le Cloud Computing attire l'attention pour sa plus haute performance et sa disponibilité à faible coût. A la différence du Cloud Computing, le Fog Computing est au plus près des objets connectés. Il facilite le stockage et l'analyse de données. Une façon d'apporter la disponibilité et la performance est d'utiliser la réplication. Dans ce travail, nous proposons une stratégie de réplication et d'ordonnancement des données dans l'environnement du Fog et Cloud Computing qui permet de répondre aux exigences des utilisateurs surtout en termes de disponibilité et de performance, tout en tenant compte des bénéfices des fournisseurs.

La stratégie proposée utilise l'ordonnancement qui vise à localiser les Fogs les plus performants et qui hébergent les données requises pour le traitement des requêtes des utilisateurs. Ce qui permet de réduire le temps de réponse moyen. La stratégie de réplication de données vise à calculer le nombre minimum de copies nécessaire pour maintenir une haute performance, de diminuer le nombre des violations et de garantir un bénéfice pour les fournisseurs.

Nous avons évalué notre stratégie en réalisant plusieurs séries d'expériences en variant plusieurs paramètres. Les résultats montrent la supériorité de notre stratégie par rapport aux approches centralisées et décentralisées standard.

Mots clés : Fog Computing, Cloud Computing, Ordonnancement, Réplication, iFogSim.

Abstract

Cloud computing is attracting attention for its higher performance and low cost availability. Unlike Cloud Computing, Fog Computing is as close as possible to connected objects. It facilitates data storage and analysis. One way to bring availability and performance is to use replication.

In this work, we propose a data replication and scheduling strategy in the Fog and Cloud Computing environment that allows to meet user requirements especially in terms of availability and performance, while taking into account the benefits of suppliers. The proposed strategy uses scheduling which aims to locate the most efficient Fogs and which host the data required for processing user requests.

This makes it possible to reduce the average response time. The data replication strategy aims to calculate the minimum number of copies necessary to maintain high performance, decrease the number of violations and ensure a benefit for the suppliers. We evaluated our strategy by performing several series of experiments varying several parameters.The results show the superiority of our strategy over standard centralized and decentralized approaches.

Keywords : Fog Computing, cloud Computing, Scheduling, Replication, iFogSim.

ملخص

تجذب الحوسبة السحابية الانتباه لأدائها العالي وتوافرها بتكلفة منخفضة. على عكس الحوسبة السحابية ، فإن حوسبة الضباب قريبة قدر الإمكان من الكائنات المتصلة. يسهل تخزين البيانات وتحليلها. طريقة واحدة لتحقيق التوافر والأداء هي استخدام النسخ المتماثل. في هذا العمل ، نقترح استنساخ البيانات واستراتيجية الجدولة في بيئة الحوسبة السحابية والضبابية التي تسمح بتلبية متطلبات المستخدم خاصة من حيث التوافر والأداء ، مع مراعاة فوائد الموردين. تستخدم الإستراتيجية المقترحة الجدولة التي تهدف إلى تحديد أفضل أداء للضباب والذي يستضيف البيانات المطلوبة لمعالجة طلبات المستخدم. هذا يجعل من الممكن تقليل متوسط وقت الاستجابة. تهدف استراتيجية نسخ البيانات إلى حساب الحد الأدنى لعدد النسخ اللازمة للحفاظ على الأداء العالي وتقليل عدد الانتهاكات وضمان ربح للموردين. قمنا بتقييم استراتيجيتنا من خلال إجراء عدة سلاسل من التجارب متفاوتة في العديد من المعلمات. تظهر النتائج تفوق استراتيجيتنا على المقاربات المركزية واللامركزية القياسية.

الكلمات المفتاحية : حوسبة الضباب ، الحوسبة السحابية ، الجدولة ، النسخ المتماثل.

REMERCIEMENT

Je remercie, en premier lieu, ALLAH, le tout puissant, de m'avoir permis et accorder la volonté, la patience et le courage pour réaliser ce travail.

Nos parents à qui aucun remerciement ne saurai t exprimer toute l'affection et la reconnaissance que nous leur portons, Ce travail est le résultat de leur immense sacrifice et de leur tendresse qu'ils nous ont toujours apportés.

J'exprime ma profonde gratitude à mes frères et mes sœurs pour leur encouragement et tout leur apport qui n'était pas moindre

Nos formateurs sans exception pour leurs soutiens technique et psychologique et leurs efforts afin de nous assurer une meilleure formation et surtout notre encadreur Mr.LIMAM Said et leur précieux conseils.

Et tous ceux qui nous ont aidés de prés ou de loin.

TABLE DES MATIÈRES

abstract **4**

Table des matières **6**

Table des abréviations **9**

Table des figures **10**

Introduction Général **12**

1 cloud/fog computing **13**
1.1 Introduction : 13
1.2 Cloud computing : 13
1.2.1 Définition : 13
1.2.2 Types de services Cloud (SaaS, PaaS, et IaaS) : 14
1.2.3 Les modèles de déploiement : 15
1.2.4 Avantage du Cloud computing : 15
1.2.5 inconvénients du cloud computing : 16
1.3 fog computing : 16
1.3.1 definition : 16
1.3.2 Caractéristiques du Fog Computing : 17
1.3.3 Avantages d'une architecture fog computing : 18
1.3.4 Inconvénients d'une architecture fog computing : 19
1.3.5 Différenciation avec le Cloud-computing 20
1.4 Conclusion : 20

2 Etat de l'art sur les méthodes de réplication dans le Cloud computing **21**
2.1 Introduction : 21
2.2 Réplication des données dans le Cloud computing 21

2.3 Types de réplication 22
2.3.1 Réplication synchrone 22
2.3.2 Réplication asynchrone 22
2.4 . Les méthodes de réplication existantes dans la littérature : 22
2.4.1 Approche basée sur l'économie d'énergie : 22
2.4.2 Approche basée sur les réseaux de neurones : 23
2.4.3 Approche basée sur la réplication par bloc de fichiers (HDFS) . . . 24
2.4.4 Approche basée sur la réplication adaptative 25
2.4.5 Approche basée sur la réplication et le placement dynamique des répliques : 26
2.4.6 Approche basée sur les graphes : 28
2.5 Critères de comparaisons des méthodes de réplication étudiées : 29
2.5.1 Synthèse : 29
2.5.2 Conclusion : 30

3 Stratégie proposée 31
3.1 Introduction 31
3.2 Objectifs : 31
3.3 Stratégie proposée : 32
3.3.1 Architecture : 32
3.3.2 Traitement d'une requête : 34
3.4 La phase de Localisation d'un service : 36
3.4.1 La phase de Réplication : 38
3.4.2 La phase de Suppression : 40
3.4.3 Conclusion : 40

4 Simulation 42
4.1 Introduction : 42
4.2 Environnement de Développement : 42
4.3 Langage de programmation Java : 43
4.4 Extraire des fonctionnalités : 43
4.5 Métriques utilisées : 45
4.5.1 Temps de réponse : 45
4.5.2 Effet du nombre de requêtes sur le temps de réponse moyen : . . . 46
4.5.3 Effet de la taille des fichiers sur le temps de réponse moyen : . . . 46
4.6 Consommation d'énergie moyenne : 47

4.6.1 Effet du nombre de requêtes sur la consommation d'énergie moyenne : 48
4.6.2 Effet de la taille des fichiers sur la consommation d'énergie moyenne : 48
4.7 Utilisation de la bande passante : 49
4.7.1 Effet du nombre de requêtes sur l'utilisation de la bande passante : 50
4.7.2 Effet de la taille des fichiers sur l'utilisation de la bande passante : 51
4.8 Indice de performance : 52
4.8.1 Effet du nombre de requêtes sur l'indice de performance : 53
4.8.2 Effet de la taille des fichiers sur l'indice de performance : 53
4.9 Conclusion : 54

Conclusion Générale **56**

Bibliographie **57**

TABLE DES ABRÉVIATIONS

DN	Degré Négatif.
FP	Facteur Positif.
FN	Facteur Négatif.
FR	Facteur de Réplication.
HDFS	Fhadoop Distributed File System.
HAS	Heuristic Audit Strateg.
IAAS	Infrastructure As A Service.
PRM	Partial Réplication Model.
PAAS	Platform As A Service. .
SaaS	Software As A Service.
VPN	Réseau Virtuel Privé.
QoS	Quality of Service.
UOT	User Opération Table.
EDI	Environnement de Développement Intégré.
IOT	Interner Of Things.
VM	Machine Virtuelle.
DP	le Degré de Popularuté.

TABLE DES FIGURES

FIGURE **Page**

1.1 Utilisations de fog computing 17

3.1 Schéma de l'Architecture Proposée 33
3.2 Schéma de l'Architecture détailler de région 01 34
3.3 Diagramme d'activité de la stratégie proposée 35
3.4 Diagramme d'activité de localisation 37
3.5 Diagramme d'activité de la réplication 39
3.6 Diagramme d'activité de vérification et de suppression des données 40

4.1 Graphique linéaire pour l'effet du nombre de requêtes sur le temps de réponse moyen 46
4.2 Graphique à barres pour l'effet de la taille des fichiers sur le temps de réponse moyen 47
4.3 Graphique linéaire pour l'effet du nombre de requêtes sur la consommation d´énergie moyenne 48
4.4 Graphique à barres pour l'effet de la taille des fichiers sur la consommation d´énergie 49
4.5 Graphique linéaire pour l'effet du nombre de requêtes sur l'utilisation de la bande passante 50
4.6 Graphique à barres effect de la taille des fichiers sur l'utilisation de la bande passante 51
4.7 Graphique linéaire pour l'effet du nombre de requêtes sur l'indice de performance 53
4.8 Graphique à barres pour l'effet de la taille des fichiers sur l'indice de performance 54

Contexte et problématique :

Au cours des dix dernières années, l'évolution des technologies du réseau, les performances de calcul des microprocesseur et la croissance des capacités du support de stockage de données a largement contribué à l'émergence de l'informatique distribuée. Ce contexte inspire la communauté informatique à s'intéresser à l'architecture distribuée, en particulier au Cloud Computing pour profiter de sa puissance de calcul et de ses capacités de stockage des données. Ce type d'infrastructure constitue un environnement privilégié qui permet de partager des ressources et des services.

Dans le but de garantir une performance élevée et une haute disponibilité, les mêmes objets (fichiers, programmes, etc.) peuvent être répliquées plusieurs fois. Chaque réplique est détenue et accessible par un ou plusieurs utilisateurs, leur gestion peut poser des problèmes de la cohérence de données. La complexité de la gestion des stratégies de réplication est due à plusieurs facteurs : trouver le bon emplacement de la copie ; déterminer l'objet à répliquer, déterminer le nombre minimum (ou maximum) de copies et gérer les mises à jour et les problème de cohérence entre les répliques. Les méthodes existantes de gestion de la réplication offrent une Du point de vue de la performance et de la qualité de service, la garantie est limitée.

Le cloud computing ou informatique en nuage est une infrastructure dans laquelle la puissance de calcul et le stockage sont gérés par des serveurs distants auxquels les usagers se connectent via une liaison internet sécurisée. Les objets connectés deviennent des points d'accès pour exécuter des applications ou consulter des données qui sont hébergées sur les serveurs. Le cloud se caractérise également par sa souplesse qui permet aux fournisseurs d'adapter automatiquement la capacité de stockage et la puissance de calcul aux besoins des utilisateurs.

Le Fog Computing est un paradigme informatique assez récent qui vise à rapprocher les infrastructures et les services de Cloud Computing aux utilisateurs ce qui implique différents types d'applications dans un vaste réseau d'objets connectés et des villes

intelligentes avec une distribution dense et des ressources et une ´Energie limitée. Les services Fog Computing sont capables de collecter et de transférer en temps réel des gros volumes de données environnementales hétérogènes, qui s'exécutent 'à la fois dans le cloud et dans des appareils, tels que des passerelles et des routeurs intelligents dont des milliards d'appareils inter-connectés.[1]

L'objectif de notre travail est de proposer une stratégie d'ordonnancement et de réplication des données dynamique où les requêtes des utilisateurs seront capables d'être traitées depuis n'importe quel emplacement, avec un temps de réponses réduit, et un coût raisonnable, tout en optimisant l'utilisation du réseau.

Contributions :

Dans notre travail, nous proposons une stratégie d'ordonnancement et de réplication dans un environnement de cloud Computing et fog Computing , pour assurer la disponibilité des services et améliorer les performances tel que le temps de réponse et aussi de réduire la consommation d'´énergie.

Notre mémoire est structuré autour de quatre chapitres avec une introduction et une conclusion générale : Le premier chapitre est une initiation au terme de l'informatique en nuage plus connu sous le nom de Cloud computing.

Dans le deuxième chapitre nous avons présenté quelques méthodes développées dans la littérature traitant la réplication des données dans le Cloud.

Et le troisième chapitre nous avons proposé une stratégie d'ordonnancement et réplication qui permettent d'améliorer les performances, garantir la disponibilité et la cohérence des données avec un budget réduit.

CHAPITRE 1

CLOUD/FOG COMPUTING

1.1 Introduction :

Aujourd'hui, les systèmes d'information sont toujours à la recherche de nouvelles technologies pour rendre leurs systèmes informatiques performants, économiques et écologiques. Des études ont montré que la plupart des serveurs de nos salles informatiques n'utilisent que 10 % de leur capacité de processeur, cela signifie une énorme perte de ressources financières, humaines et de stockage. Afin de surmonter ce problème d'une grande entreprise comme Amazon, Microsoft... ont décidé de fournir leur infrastructure informatique qui a permet à la naissance du cloud computing. Cependant, de nombreuses entreprises à capital réduit peinent à entrer sur le terrain et utilisent des technologies obsolètes.

Dans ce chapitre, nous présenterons le cloud computing, ses caractéristiques et ses domaines d'application, ainsi que son architecture et les services qu'il fournit ensuite nous parlerons du Fog Computing.

1.2 Cloud computing :

1.2.1 Définition :

Le Cloud computing est un modèle d'accès, à travers le réseau Internet, à un ensemble de ressources numériques, pouvant être allouées et libérées à la demande et pour lesquelles le fournisseur du service assure l'ensemble des activités de maintenance, de support et d'exploitation. C'est une forme particulière de gestion de l'informatique,

dans laquelle l'emplacement et le fonctionnement du nuage ne sont pas portés à la connaissance des clients.

Le Cloud Computing fourni des services via l'internet. Il désigne le stockage et l'accès aux données par l'intermédiaire d'internet plutôt que via le disque dur d'un ordinateur. Il s'oppose ainsi à la notion de stockage local, consistant à entreposer des données ou à lancer des programmes depuis le disque dur.

De manière générale, on parle de Cloud Computing lorsqu'il est possible d'accéder à des données ou à des programmes depuis internet, ou tout du moins lorsque ces données sont synchronisées avec d'autres informations sur internet. Il suffit donc pour y accéder de bénéficier d'une connexion internet.

1.2.2 Types de services Cloud (SaaS, PaaS, et IaaS) :

1.2.2.1 Software as a service (SaaS) :

Le Cloud au niveau **SaaS** représente le plus souvent un catalogue d'application accessible en mode service aux utilisateurs ou clients finaux.

Pour une entreprise, l'utilisation d'un Cloud en mode **SaaS** peut-être particulièrement pertinente dans les phases de maquettage ou des prototypages car cela permet dans un délai très court et pour un cout réduit, d'évaluer une solution en œuvre des ressources propre, de l'internaliser, le cas échéant, si le résultat est concluant. [1]

1.2.2.2 Infrastructure as a service (IaaS)

Le **laas** concerne essentiellement les Clouds d'infrastructure. Ces derniers fournissent à la demande un ensemble de services de niveau « bas », c'est à dire des serveurs, réseaux etc. Cela permet ainsi à une entreprise cliente de pouvoir bénéficier de la puissance d'une infrastructure, ponctuellement, sans devoir investir beaucoup. Lorsqu'il y a une certitude de variations fortes de charge, ou une incertitude sur la capacité d'une infrastructure à délivrer un service, le Cloud Computing est alors une solution très adaptée : par exemple pour des besoins de type paie, messagerie ou éditique [1]

1.2.2.3 Platform as a service (PaaS) :

Si le **laas** concerne essentiellement la production et l'exploitation, Le Cloud de niveau Paas concerne les développeurs et les producteurs d'applications, soit deux niveaux de service : les plateformes de développement, et les applications qui fournissent le service du niveau supérieur **(SaaS)**. Le **PaaS** permet, par exemple, à disposition des développeurs un Framework développement adapté à leurs besoins. Il permet aussi

de donner aux applications un cadre d'exécution qui produira des services **SaaS** (par exemple Salesforce). [1]

1.2.3 Les modèles de déploiement :

Il existe 4 modèles de déploiement du Cloud computing :

1.2.3.1 Le Cloud privé :

qui peut se déployer sous deux formes distinctes :

Cloud privé interne : hébergé par l'entreprise elle-même, parfois partagé ou mutualisé en mode privatif avec les filiales.

Cloud privé externe : hébergé chez un tiers, il est entièrement dédié à l'entreprise et accessible via des réseaux sécurisés de type VPN (Réseau virtuel privé).

1.2.3.2 Le Cloud public :

est accessible par Internet et géré par un prestataire externe. Il est ouvert au public ou à de grands groupes industriels. Cette infrastructure est possédée par une organisation qui vend des services Cloud.

1.2.3.3 Le Cloud hybride :

Le Cloud hybride ou mixte associe l'utilisation, pour une même entreprise, d'un Cloud privé et d'un Cloud public. Ces infrastructures sont liées entre elles par la même technologie qui autorise la portabilité des applications et des données.

1.2.3.4 Le Cloud communautaire :

Le Cloud communautaire est dédié à une communauté professionnelle spécifique incluant partenaires, sous-traitants, etc. pour travailler de manière collaborative sur un même projet ou Cloud gouvernemental dédié aux institutions étatiques.

1.2.4 Avantage du Cloud computing :

Flexibilité : Les utilisateurs peuvent dimensionner les services en fonction de leurs besoins, personnaliser les applications et accéder aux services Cloud depuis n'importe où avec une connexion Internet.

Efficacité : L'entreprise peut mettre des applications sur le marché rapidement sans s'inquiéter des coûts d'infrastructure ou de la maintenance.

Valeur stratégique : Les services de Cloud donnent aux entreprises un avantage concurrentiel en leur apportant la technologie la plus novatrice qui existe.

1.2.5 inconvénients du cloud computing :

Le système du cloud computing révèle quelques inconvénients :

— Ses réseaux informatiques sont potentiellement attaquables et les services virtuels peuvent être mis hors fonction.

— Vous dépendez d'un prestataire (Google Cloud Platform, AWS, Azure, etc.). Il est fondamental de veiller à ce qu'il offre les garanties indispensables et connaisse les attentes de votre métier.

— Les services nécessaires à votre activité dépendent de la connexion internet. Une panne peut perturber votre organisation.

— Vous perdez une partie de la maîtrise de votre système informatique.

1.3 fog computing :

1.3.1 definition :

Le Fog ne remplace pas le Cloud, mais ces deux paradigmes de calcul sont complémentaires [2]. En effet, ils coopèrent pour fournir une plate-forme extensible qui supporte à la fois des applications nécessitant des latences courtes et prédictibles ainsi que des applications nécessitant des traitements complexes et du stockage de données permanent [3]. L'idée est de servir les requêtes nécessitant des latences courtes (ex. contrôle du trafic routier, parking intelligent et diffusion en direct) par des équipements localisés dans le Fog, tandis que les requêtes nécessitant des traitements complexes et des données d'historisation (ex. photos, flux vidéo, données des réseaux sociaux, etc) sont servies par le Cloud [3].

Dans le Fog computing, les équipements de traitement et de stockage de données sont appelés les "Nœuds du Fog". Ces équipements sont de nature hétérogène au regard de leur performance [3]. Ils peuvent être des équipements modestes en ressources comme des set-top-boxes, des points d'accès, des switchs et des routeurs, ou des équipements riches en ressources comme les points de présence (POP).

De plus, la distance géographique entre les nœuds du Fog et les utilisateurs finaux (et les objets connectés) est très variable ; de quelques mètres comme les points d'accès jusqu'à plusieurs centaines de kilomètres comme les POP. En effet, dans les infrastructures de type Fog, les équipements modestes en ressources sont localisés près des utilisateurs

finaux et ils fournissent des latences courtes (qui se comptent en millisecondes), tandis que les équipements riches en ressources sont localisés loin des utilisateurs finaux et fournissent des latences importantes et imprédictibles (qui se comptent en secondes, voir en minutes) [4].

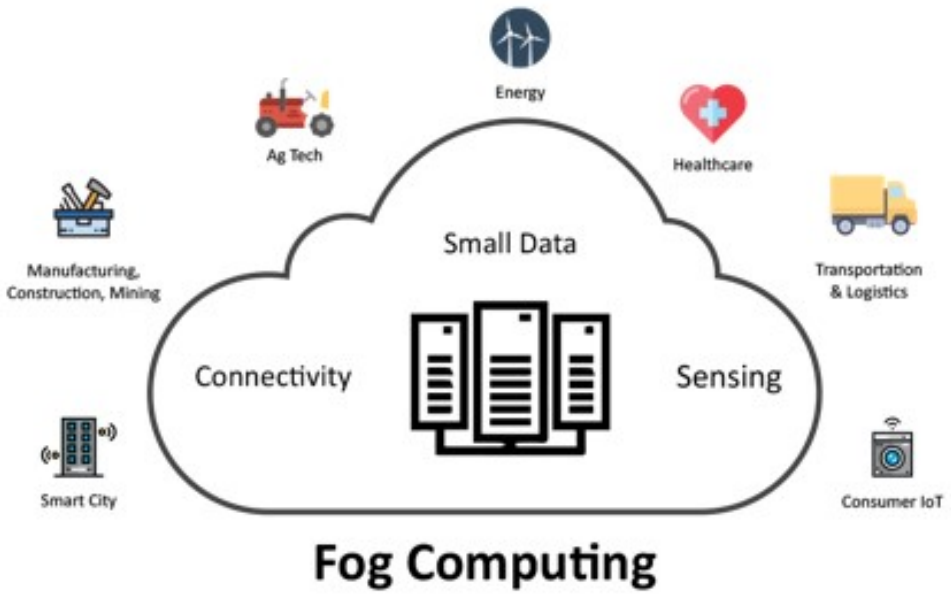

FIGURE 1.1: Utilisations de fog computing

1.3.2 Caractéristiques du Fog Computing :

Essentiellement, le Fog computing est une extension du cloud mais plus proche des éléments qui fonctionnent avec les données IoT. Comme le montre la Figure 2.1, il agit comme un intermédiaire entre le cloud et les objets connectées, ce qui rapproche les services de traitement à ces objets.

Les caractéristiques du Fog Computing peuvent être résumées comme suit [7] :

détection de l'emplacement et faible latence :

Le Fog computing prend en charge la reconnaissance de l'emplacement dans lequel les nœuds Fog peuvent être déployés à différents emplacements. De plus, comme le Fog est plus proche des objets connectés, il offre une latence plus faible lors du traitement des données.

Répartition géographique :

Contrairement au cloud centralisé, les services et applications fournis par le Fog sont distribués et peuvent être déployés n'importe où.

Scalabilité :

Il existe des réseaux de capteurs à grande échelle qui surveillent l'environnement. Le Fog fournit des ressources informatiques et de stockage distribué qui peuvent fonctionner avec de tels appareils d'extrémité grande échelle.

Prise en charge de la mobilité :

L'un des aspects importants des applications de Fog est la possibilité de se connecter directement aux appareils mobiles et donc d'activer les méthodes de mobilité, telles que le protocole de séparation des identifiants de localisation (LISP) qui nécessite un système d'annuaire distribu [7].

Interactions en temps réel :

Les applications de Fog computing fournissent des interactions en temps réel entre les nœuds Fog plutôt que le traitement par lots utilisé dans le cloud.

Hétérogénéité :

Les nœuds de Fog sont conçus par différents fabricants et se présentent donc sous différentes formes et doivent être déployées en fonction de leurs plates-formes. Le Fog a la capacité de fonctionner sur différentes plateformes [7].

Interopérabilité :

Les composants Fog peuvent inter-opérer et fonctionner avec différents domaines et entre différents fournisseurs de services.

1.3.3 Avantages d'une architecture fog computing :

Le Fog computing étend le modèle de cloud computing aux bords du réseau. Bien que le Fog et le cloud utilisent des ressources similaires (réseau, calcule et stockage) et partagent un bon nombre des mêmes mécanismes et attributs (virtualisation, multi-location).

Le Fog computing apporte de nombreux avantages pour les appareils d'internet des objets. Ces avantages peuvent être résumés comme suit [7] :

Une plus grande agilité commerciale :

Avec l'utilisation des bons outils, les applications du Fog computing peuvent être rapidement développées et déployées. De plus, ces applications peuvent programmer la machine pour qu'elle fonctionne selon les besoins du client [6].

Faible latence :

Fog computing a la capacité de prendre en charge des services en temps réel (par exemple, jeux, streaming vidéo).

Distribution géographique et à grande échelle :

Fog computing peut fournir des ressources informatiques et de stockage distribué à des applications de grande taille et largement distribuées [7].

Frais d'exploitation réduits :

Economie de bande passante réseau en traitant localement les données sélectionnées au lieu de les envoyer au cloud pour analyse [7].

Flexibilité et hétérogénéité :

Le Fog computing permet la collaboration de différents environnements physiques et infrastructures entre plusieurs services [6] .

Scalabilité :

La proximité du Fog avec les appareils finaux permet de faire évoluer le nombre d'appareils et de services connectés [6].

1.3.4 Inconvénients d'une architecture fog computing :

Coûts de matériel plus élevé :

les terminaux et capteurs IdO de l'architecture fog computing doivent être équipés d'une unité de traitement supplémentaire pour permettre le traitement des données local et la communication entre appareils.

Faible protection contre les pannes ou les abus :

les entreprises qui ont recours au fog computing doivent équiper les appareils et les capteurs IdO de contrôleurs situés à la périphérie du réseau, p.ex. Dans des installations de production qui sont difficiles à protéger contre les défaillances ou une mauvaise utilisation.

Augmentation des besoins de maintenance :

le traitement décentralisé des données implique des besoins accrus de maintenance. La raison en est que les contrôleurs et les éléments de stockage sont répartis sur l'ensemble du réseau et que, contrairement aux solutions Cloud, ils ne peuvent être ni maintenus ni administrés de manière centralisée.

Défis supplémentaires en sécurité réseau :

le fog computing est sensible aux attaques de type « homme du milieu » (ou intercepteur).

1.3.5 Différenciation avec le Cloud-computing

Le fog computing se distingue avant tout du Cloud computing par le lieu où les ressources sont mises à disposition et où les données sont traitées.

Le Cloud computing se fait généralement dans des centres de données centralisés.

Des ressources comme la puissance de traitement ou la capacité de stockage sont regroupées au niveau de serveurs principaux et utilisées par les clients via le réseau.

La communication entre deux périphériques ou plus s'effectue donc toujours via un serveur en arrière-plan. Une telle architecture atteint ses limites avec des concepts tels que « l'usine intelligente », où les données doivent être échangées en permanence entre d'innombrables terminaux.

Le fog computing recourt ici à un traitement intermédiaire proche de la source de données, afin de réduire le débit de données vers le centre de calcul.

1.4 Conclusion :

Dans le premier chapitre ,nous avons défini le Cloud computing et le fog computing , nous avons aussi présenté ses caractéristiques, ses avantages et ses inconvénients ainsi que ses domaines d'application. Le Cloud computing et le fog computing marquent un réel avancement vers l'infrastructure informatique dématérialisée.

Ils fournissent des ressources informatiques, logicielles ou matérielles, accessible à distance en tant que service. L'adoption de ces modèles soulèvent un certain nombre de défis, notamment au sujet de la disponibilité des ressources.

CHAPITRE 2

ETAT DE L'ART SUR LES MÉTHODES DE RÉPLICATION DANS LE CLOUD COMPUTING

2.1 Introduction :

Le besoin croissant des applications à accéder simultanément à des données réparties sur plusieurs sites, a conduit la communauté informatique à s'intéresser à l'étude de la réplication des données dans le Cloud computing et le fog computing . Ce chapitre comprend une présentation des méthodes développées dans la littérature traitant la réplication des données dans le Cloud.

2.2 Réplication des données dans le Cloud computing

La réplication crée plusieurs copies d'une entité existante [9]. Elle permet d'assurer la disponibilité des ressources, elle assure également la fiabilité en créant plusieurs copies des mêmes données sur différents sites. La réplication fournit un coût d'accès minimal et une utilisation partagée de la bande passante en répliquant les données. La valeur de la réplication est de fournir un accès transparent et sans faille aux ressources en cas de défaillance du système . [10]

2.3 Types de réplication

2.3.1 Réplication synchrone

Elle garantit que lorsqu'une transaction met à jour une réplique primaire, toutes ses répliques secondaires sont mises à jour dans la même transaction. L'avantage essentiel de la mise à jour synchrone est de garder toutes les données au même niveau de mise à jour. Le système peut alors garantir la fourniture de la dernière version des données quel que soit la réplique accédée. Les inconvénients sont cependant multiples, ce sont d'une part, la nécessité de gérer des transactions multiples coûteuses en ressources et d'autre part, la complexité des algorithmes de gestion de panne d'un site, etc. C'est pour cela que l'on préfère souvent le mode de mise à jour asynchrone [11].

2.3.2 Réplication asynchrone

C'est le mode de distribution dans lequel certaines sous-opérations locales effectuées suite à une mise à jour globale sont accomplies dans des transactions indépendantes en temps différé. Le temps de mise à jour des copies peut être plus au moins différé : les transactions de report peuvent être lancées dès que possible ou à des instants fixes, par exemple le soir ou en fin de semaine . L'avantage est la possibilité de mettre à jour en temps choisi des données, tout en autorisant l'accès aux versions anciennes avant la mise à niveau. L'inconvénient est que l'accès à la dernière version n'est pas garanti.La réplication asynchrone est basée sur deux approches [11] :

Approche basée sur l'état : La réplique source est mise à jour, ensuite le sous-système transmet l'état sur les répliques par la fusion de l'état livré à l'état local.

Approche basée sur les opérations : Le sous-système envoie l'opération de mise à jour et ses paramètres à toutes les répliques.

2.4 . Les méthodes de réplication existantes dans la littérature :

2.4.1 Approche basée sur l'économie d'énergie :

Les auteurs de l'article ont proposé un modèle basé sur la consommation électrique et les besoins en bande passante pour l'accès aux bases de données dans les centres de données de cloud computing. En outre, ils ont proposé une stratégie de réplication à faible consommation introduite à partir du modèle proposé, améliorant ainsi la qualité de service (QoS) et réduisant le temps de communication entre les centres de données

répartis géographiquement et au sein de chaque centre de données. Afin d'assurer l'efficacité énergétique et une bonne performance pour les applications Cloud ; Les auteurs proposent un algorithme de copie qui prend en compte la consommation électrique et la bande passante nécessaires à l'accès aux données. Chaque objet de données est disponible dans une base de données centrale et est basé sur des observations historiques de la fréquence d'accès aux données ; Un module appelé Réplica Manager (RM) situé dans la base de données centrale analyse les métadonnées pour déterminer quels sont les objets qui doivent être répliqués et où seront placés. RM calcule les taux d'accès et de mise à jour pendant les intervalles précédents afin d'estimer leurs valeurs futures.

2.4.2 Approche basée sur les réseaux de neurones :

Al Ridhawi et al. [12] proposent une méthode de prédiction de localisation pour identifier des emplacements potentiels de densité élevée d'utilisateurs. Un algorithme partiel de réplication de données est utilisé pour reproduire à l'avance les données demandées par l'utilisateur. La méthode proposée a pour objectif de minimiser le temps d' accès aux données.

La solution utilise une technique de prédiction de localisation qui s'appuie sur la Théorie de Dempster-Shafer [13], c'est une théorie probabiliste mathématique combine des preuves pour prendre des décisions dans des circonstances incertaines afin de fournir des résultats de prédiction rapides et précis. Il ont proposé une nouvelle stratégie de réplication partielle des données . La solution proposée garantira que les utilisateurs peuvent obtenir les données les plus accessibles à tout moment lorsqu'ils arrivent, en réduisant le temps d'accès aux données et de maintenir un certain niveau de QoS. Le modèle de sélection de copie proposé fournit une solution efficace pour accéder aux fichiers locaux et distants. La solution suppose qu'il existe plusieurs utilisateurs répartis dans différents sites distants , ces utilisateurs soumettront plusieurs travaux. Ces travaux nécessitent un ou plusieurs fichiers. Les fichiers sont situés dans des ressources locales et distantes (c'est-à-dire le site de stockage original ou des sites de stockage tiers).

Le modèle de réplication partielle (PRM) proposé permet de répliquer une partie d'un fichier au lieu du fichier entier. Un ensemble ou une partie de fichiers est sélectionné pour la réplication partielle en fonction des requêtes de soumission de jobs exécutées par les utilisateurs. L'approche proposée utilise un réseau de neurones artificiels (RNA) qui est basé sur l'historique passé des utilisateurs pour prédire l'emplacement d'un fichier soit dans le cache, les ressources locales ou dans des ressources à distance. Les RNA sont des programmes informatiques qui sont formés pour reconnaître les modèles d'entrée et

les associer à des sorties particulières .

2.4.3 Approche basée sur la réplication par bloc de fichiers (HDFS)

Dans l'article [14], l'auteur a proposé une stratégie de réplication de données pour le cloud computing avec des données bio-informatiques afin de réduire l'exécution d'applications bio-informatiques, en utilisant un système de classement distribué. Le Hadoop Distributed File System (HDFS) [15] permet aux ordinateurs ordinaires de faire partie du cluster, permettant l'utilisation de machines hétérogènes pour le calcul à faible coût. HDFS utilise la réplication de blocs, ou plutôt, divise le fichier en interne en un ou plusieurs blocs, chaque bloc dans HDFS est copié sur d'autres ordinateurs. La stratégie de réplication des données HDFS donne la priorité à l'équilibrage de charge . La distribution des Copies de blocs de données dans le système de fichiers est en fonction de la capacité de stockage de chaque nœud de données.

Conforme aux normes d'organisation des fichiers et peut être utilisé correctement Stratégie de copie, besoin de compléter certaines adaptations dans le système Dans un document qui peut expliquer ce besoin, ces adaptations sont divisées en quatre phases :

Phase 1 : Le bloc de données ne sera pas directement placé dans HDFS, car si le fichier est stocké directement, la fin du bloc de données ne peut pas être empêchée, ce qui peut entraîner l'Interruption de la séquence de filtrage. Au lieu de cela, les blocs sont traités via un processus avant d'être insérés dans la base de données. Ce processus détermine quelle partie du bloc pour chaque fichier doit être inséré

Phase 2 : Puisque chaque partie du fichier devient un bloc distinct, il est nécessaire de connaître l'emplacement de chaque bloc pour comprendre le choix de l'ordinateur impliqué dans le traitement du fichier. Pour cela, le mappage est appliqué à l'ordinateur propriétaire du fichier au stade de la création de la copie. Le mappage se compose uniquement d'un fichier contenant l'identifiant du bloc et chaque emplacement de copie. Une fois la reconnaissance terminée, l'exécution de la tâche est lancée en référençant le bloc de diffusion.

Phase 3 : Bien que Hadoop dispose d'un système de fichiers multiplateforme, il est toujours difficile pour les programmes d'allocation de ressources de comprendre l'état actuel du système. Afin de résoudre ce problème, un programme d'allocation de ressources est utilisé. Par exemple, un ordinateur à quatre cœurs peut être configuré pour exécuter jusqu'à huit tâches en même temps en puissance standard, mais en fonction des activités imprévues dans le cloud, différentes tâches peuvent être créées, le traitement des tâches

peut être affecté. De même, si on suppose que huit tâches peuvent être exécuté sans besoin de l'utilisation de la pleine puissance de CPU et donc, il peut y avoir une tâche qui peut utiliser les ressources disponibles sans affecter les autres processus. Phase 4 : La dernière étape est après l'allocation des fichiers nouvellement générés traitement. A ce stade, la première étape est terminée. Pour le fichier créé, générez Traiter le journal et compléter le mappage décrit dans la deuxième étape

2.4.4 Approche basée sur la réplication adaptative

Dans l'article [16], une stratégie de réplication adaptative des données est proposée, qui est principalement basée sur la sélection dynamique des fichiers de données copiés pour améliorer la fiabilité globale du système et répondre aux exigences de qualité de service. De plus, la stratégie proposée détermine dynamiquement le nombre de répliques et le nombre de nœuds à utiliser. La méthode de réplication analyse les fichiers de données populaires et les modèles de demande récents, puis repose sur l'utilisation d'une technologie de séries chronologiques légère (une série chronologique permet d'étudier l'évolution d'une variable dans le temps).

Les séries chronologiques sont présentées dans un tableau, où la première colonne représente le temps et les autres colonnes représentent les variables qui changent au fil du temps. Méthode recommandée pour identifier chaque fichier Le nombre d'exigences de réplication de données que nous avons cité : le nombre de copies Pour chaque donnée sélectionnée ; la position des nouvelles répliques sur les centres de données disponibles et enfin la surcharge de la stratégie de réplication sur l'infrastructure du Cloud. La réplication de données adaptative proposée comporte trois phases importantes :

(i) Quels fichiers de données doivent être sélectionnés pour la réplication ?

(ii) Définir le nombre de répliques appropriées pour répondre aux QoS spécifiées.

(iii) Déterminer le meilleur emplacement des répliques.

La première étape consiste à déterminer quelles données à répliquer et d'estimer le temps de réplication. La phase de sélection analyse l'historique des demandes d'accès aux données et utilise une technique de série chronologique légère pour prédire la future fréquence d'accès des données ; Si les demandes futures prévues de fragments de données dépassent un seuil adaptatif, les blocs de données seront sélectionnés pour la réplication. Soit pdk le degré de popularité d'un bloc bk.

pdk est défini comme la fréquence d'accès future basée sur le nombre de demande d'accès, ank(t) à l'instant t, le degré de popularité pdk d'un bloc bk peut être calculé à l'aide du Lissage Linéaire et exponentielle de Holt (HLES). Le lissage linéaire et

exponentiel de Holt (HLES) est une technique de prédiction de la série chronologique à faible coût. HLES est sélectionné pour sa capacité de lissage et fournit des prévisions à court terme pour les taux d'arrivée des demandes mesurées et les taux de demande de service.

Par conséquent, HLES permet au Framework proposé de surveiller les taux d'arrivée et les taux de service et de prévoir à court terme les taux d'arrivée futurs et les taux de service avec un faible temps de calcul. HLES lisse la série chronologique et fournit une prévision à court terme basée sur la tendance qui existe sur les séries chronologiques [10]. Supposons que ank(t) soit une valeur de série chronologique à l'instant t.

La prévision linéaire pour les m étapes à venir est la suivante :

$$pdk = ank(t+m) = Lt + btm \tag{2.1}$$

Où Lt et bt sont des estimations exponentiellement lissées du niveau et de la tendance linéaire de la série à l'instant t :

$$Lt = \alpha \cdot ank(t) + (1-\alpha)(Lt-1+bt) \tag{2.2}$$

Avec α est un paramètre de lissage, $0 < \alpha < 1$,

$$bt = \beta(Lt - Lt-1) + (1-\beta(bt-1)) \tag{2.3}$$

Où β est un coefficient de tendance, $0 < \beta < 1$.

Une grande valeur de α ajoute plus de poids aux valeurs récentes plutôt qu'aux valeurs précédentes afin de prédire la valeur suivante. Une grande valeur de β ajoute plus de poids aux variations du niveau que la tendance précédente. Le facteur de réplication est défini comme la moyenne du ratio du degré de popularité et de la disponibilité moyenne des répliques sur les différents nœuds de données de tous les blocs l du fichier de données fi. Il sert à déterminer si le fichier de données fi doit être répliqué.

2.4.5 Approche basée sur la réplication et le placement dynamique des répliques :

Karuppusamy et al. proposent un algorithme de réplication et de placement dynamique pour améliorer les performances du gestionnaire du Cloud. Ils calculent le degré de popularité et le facteur de réplication pour identifier le fichier approprié à répliquer.

Ensuite, l'algorithme proposé est utilisé pour placer les répliques à l'endroit approprié. L'approche proposée se compose de trois phases : sélection de la réplique en utilisant le degré de popularité (DP), Création de réplique en utilisant le facteur de la réplique (FR) et mise en place de la réplique. Dans la première phase ; le fichier à répliquer est trouvé en calculant le DP. Dans la deuxième phase ; les répliques sont créées à l'aide du DP qui dépend de la fréquence d'accès d'un utilisateur, de plusieurs utilisateurs et des fichiers demandés auxquels sont associés des poids suivant leur importance. Le degré négatif (DN) dépend de l'espace mémoire utilisé par un fichier, du nombre de réplique et du temps de réponse d'une requête pour accéder à ce fichier.

a. Sélection des répliques :

Pour sélectionner une réplique, un fichier de données populaire est identifié en analysant les historiques d'accès et en définissant des poids différents pour différentes données accédées. Fondamentalement, les données les plus récemment consultées sont plus pertinentes à l'analyse, et sont donc placées à une priorité plus élevée. Le nombre de répliques est calculé à l'instant t comme suit :

$$\begin{cases} (RFt+2)\cdot(RFt-1) & RFt > RFt-1 \\ 0 & RFt < RFt-1 \end{cases} \tag{2.4}$$

b. Création des répliques : Pour créer la réplique, un facteur de réplication (FR) est utilisé. Le FR est déterminé en calculant le facteur positif (FP) qui lui-même dépend des degrés de popularité courant, minimal et le facteur négatif (FN) qui est calculé en fonction des degrés courant, minimal et maximal. Le but de FP est d'identifier l'importance d'un fichier pour la réplication. FN d'un fichier spécifie si ce dernier ne doit pas être répliqué.

$$PF = \frac{DP_{current} - DP_{min}}{DP_{max} - DP_{min}} \tag{2.5}$$

$$PF = \frac{DP_{current} - DP_{min}}{DP_{max} - DP_{min}} \tag{2.6}$$

$$NF = \frac{ND_{current} - ND_{min}}{ND_{max} - ND_{min}} \tag{2.7}$$

c. Placement des répliques : L'objectif du placement des répliques (obtenues à l'étape précédente) est d'assurer la disponibilité requise avec le moins de répliques sans réduire les performances du système. Ceci peut être réalisé en utilisant une stratégie de

placement des répliques qui prend en compte : la disponibilité requise, La stabilité et la défaillance des nœuds du système. Pour cela, ils doivent d'abord Considérez la liste des nœuds dans le centre de données. Triez ensuite ces nœuds par ordre décroissant à l'aide de critères de classification.

Placer plusieurs copies des mêmes données sur le même nœud n'améliore pas la disponibilité ou la tolérance aux pannes. Par conséquent, il serait utile de stocker une seule copie des mêmes données dans un nœud. Dans le système à l'étude, les défaillances des nœuds peuvent être prédites. Par conséquent, en cas de défaillance suspecte, le nœud placera toutes ses données dans un autre nœud pour assurer la disponibilité requise.

Après avoir calculé les critères de classement et les avoir triés par ordre décroissant, l'algorithme vérifie si le fichier copié peut être placé dans le nœud ou pas. Si possible, la nouvelle réplique du fichier sera stocker sur le nœud correspondant. Si le placement de la nouvelle réplique n'est pas possible, choisir le nœud suivant et vérifiez si le nœud remplit les conditions. Ce processus continue Jusqu'à ce que toutes les répliques seront placées.

2.4.6 Approche basée sur les graphes :

L'auteur de l'article [17] propose un projet pour réaliser la diffusion Et une partie des données est copiée dans le cloud pour optimiser les performances Et la sécurité des fragments de données téléchargés par son propriétaire, Et implémenter une méthode graphique pour calculer la distance pour prédire l'emplacement des répliques. La méthode proposée est très utile aux propriétaires pour protéger ces données des attaquants. Ensuite, ils ont étendu la méthode de vérification de la cohérence du système cloud lors de la mise à jour des fichiers. L'auteur propose également une stratégie d'audit heuristique (HAS : Heuristic Audit Strategy) qui utilise la table des opérations de l'utilisateur pour ajouter des lectures appropriées afin de révéler autant de violations que possible. Chaque utilisateur dispose d'un UOT (User Operation Table) pour enregistrer les opérations locales.

La méthode T-Coloring [18] est utilisée pour placer des fragments chez divers fournisseurs. Ceci est utilisé pour l'allocation des canaux en allouant des canaux aux nœuds de sorte que les canaux soient séparés d'une certaine distance pour éviter les interférences. Cette méthode interdit de stocker des fragments à proximité du nœud Stockez un fragment, éliminant ainsi de nombreux nœuds pour le stockage.

Dans ce cas, seules les partitions restantes sont sélectionnées et les nœuds qui ne contiennent aucune partition sont sélectionnés pour le stockage aléatoire.

2.5 Critères de comparaisons des méthodes de réplication étudiées :

– Réplication partielle : Il s'agit de répliquer une partie des données au lieu de la donnée entière.

– Réplication totale : Réplication entière de la donnée.

– Réplication synchrone : Garantir que lorsqu'une transaction met à jour une réplique primaire, toutes les répliques secondaires sont mises à jour dans la même transaction.

– Réplication active : Lors de réplication active, les calculs effectués par la source (ou maître) sont répliqués, c'est à dire que le nœud principal propage le calcul vers les nœuds secondaires.

– Réplication dynamique : Les copies de données sont créées et supprimées automatiquement selon l'évolution des demandes des utilisateurs.

– Disponibilité : C'est la capacité d'assurer l'accessibilité aux données durant une période donnée.

2.5.1 Synthèse :

Dans les environnements Cloud, pour fournir et gérer un service extrêmement disponible, SANTHI et al. [17] Présentent une stratégie de réplication partielle basée sur les distances pour prédire les emplacements des répliques, la méthode proposée a un moindre coût et plus efficace pour cause de réplication partielle et l'ajout d'un système d'audit heuristique qui lance plusieurs lectures pour révéler des incohérences éventuelles des répliques. Da Silva et Araujo [14] ont développé une politique de réplication en utilisant Hadoop, ce dernier permet aux ordinateurs communs de calculer avec des machines hétérogènes ce qui permet de réduire le coût et d'assurer une efficacité élevée.

Hadoop utilise la réplication par blocs, et chaque bloc est copié sur d'autres ordinateurs et ces copies sont réparties aléatoirement dans le système de fichiers en fonction de la capacité de stockage de chaque nœud de données, ce qui permet d'accroitre la disponibilité de ces dernières. Dans ce contexte, Vaquero et MERINO [19] ont proposé un algorithme d'équilibrage de charge qui utilise une technique de réplication pour le maintien des données dans les réseaux Fog afin d'équilibrer la charge et rendre Internet moins dépendante du Cloud en ayant les données disponibles plus proche des utilisateurs.

L'approche proposée assure la disponibilité avec un moindre coût. Hussein et Moussa [16] ont adopté une stratégie adaptative de réplication basée sur une sélection dynamique

des fichiers de données répliquées, pour améliorer la fiabilité et la disponibilité globale du système, dans cette méthode la réplication analyse les Fichiers de données populaires et les demandes récentes, puis en s'appuyant sur les séries chronologiques légères, elle sélectionne les données à répliquer. Karuppusamy et MUTHAIYAN proposent un algorithme de réplication et de placement dynamique, cet algorithme sert à placer les répliques à l'endroit approprié, leur approche se compose de trois phases : une phase de sélection de la réplique, une phase de création, et une phase de placement de la réplique. BORU et al. [20]

Ont présenté une stratégie de réplication à faible consommation d'énergie pour améliorer la qualité de service et réduire les délais de communication entre les centres de données, ils ont également proposé un algorithme de réplication qui prend en considération la consommation d'énergie et la bande passante requises pour l'accès aux données.

2.5.2 Conclusion :

Ce chapitre représente un diaporama des méthodes de réplication existantes dans la littérature. Dans le chapitre qui suit, nous présentons notre contribution pour résoudre le problème de réplication dans les environnements Fog computing. Nous traitons le problème de la sélection des données à répliquer, nous détaillons la stratégie mise en œuvre pour la création de ces répliques, ainsi que leur placement.

CHAPITRE 3

STRATÉGIE PROPOSÉE

3.1 Introduction

Dans les chapitres précédents, nous avons présenté la réplication et l'ordonnancement dans les environnements Cloud/Fog Computing ainsi que ses différents concepts. De plus, nous avons exploré quelques travaux qui ont été proposés dans la littérature et qui ont des points communs avec notre projet en terme de contexte. Notre objectif principal est de proposer et d'implémenter une nouvelle architecture de réplication et d'ordonnancement des données dans un environnement de Fog et Cloud Computing. La solution proposée a pour but d'améliorer les performances, de réduire la consommation de la bande passante et de garantir une qualité de service acceptable. Notre solution est basée sur une combinaison du'une stratégie d'ordonnancement des requête et du mécanisme de réplication de données.

Le présent chapitre permet de détailler notre stratégie. Tous d'abord, nous allons présenter notre architecture ensuite nous décrivant ses différents concepts, différentes phases et tout ce qui est nécessaires pour son fonctionnement.

3.2 Objectifs :

L'objectif principal de notre travail est de proposer une stratégie basé sur l'ordonnancement et la réplication des données . Dans le but d'améliorer les performances, réduire le temps de réponse des requêtes, diminuer le nombre des violations et augmenter les profits de fournisseur .

Pour cela nous allons proposer une nouvelle architecteur qui à subdiviser horizontalement sur trois sous couches chaque sous-couche à son propre rôle et ses propres taches . et aussi subdiviser verticalement en plusieurs régions . Ensuite nous proposerons une approche pour la réplication et l'ordonnancement des données en se basant sur cette architecture .

Notre proposition prend en considération que les données nécessaires pour les traitements des services sont en lecture seule et que ces services sont atomiques, ce qui veut dire qu'ils n'auraient pas besoin d'autres services et on suppose que chaque requête venant des utilisateurs (IoT) peut avoir besoin d'une donnée ou non pour son exécution.

3.3 Stratégie proposée :

Notre stratégie est composée de plusieurs phases : la phase de localisation de service, la phase de réplication des données et de suppression des donnée. Dans la section suivante, nous présentons notre topologie et nous allons détailler les différentes phase de notre solution.

3.3.1 Architecture :

Dans notre travail, nous avons proposé une plateforme basée sur le Cloud et le Fog Computing. Le Fog présente une infrastructure dense et géo-distribuée à grande échelle [21] . Nous avons subdivisé notre architecteur horizontalement sur trois couches (Cloud, Fog, IOT) et sur plusieurs régions verticalement (région1 , région2 ...) les défirent nœuds du Fog sont interconnectées entre elles par un lien de latence (une région peut avoir un ou plusieurs liens) et nous avons ajouté un ordonnanceur sur la passerelle entre le Fog et IOT .

Et on spécifié les modules de Fog (localisation de service , réplication et suppression) De plus, par la nature des équipements de Fog qui sont hétérogènes en termes de performance [22], Dont chaque couche dispose des nœuds avec des fonctionnalités et ressources différentes. Les nœuds sont agencés hiérarchiquement avec un lien de latence. Donc, plus un nœud est en haut dans la hiérarchie, plus le lien de latence augmente.

Ci-dessous nous présentons notre architecture schématisée d'une façon simple dans la Figure 3.1 :

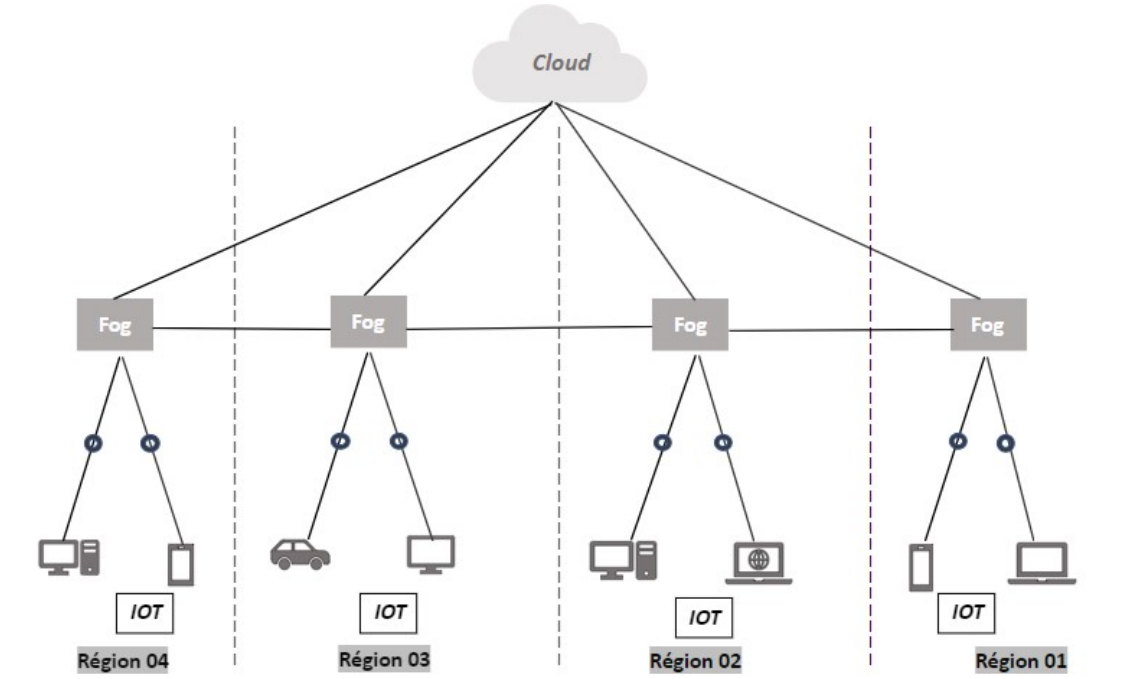

FIGURE 3.1: Schéma de l'Architecture Proposée.

On prend la région01 comme exemple pour mieux détailler les défirent modules de Fog et l'emplacement de l'ordonnanceur. Ci-dessous architecture schématisée détailler de la région 01 :

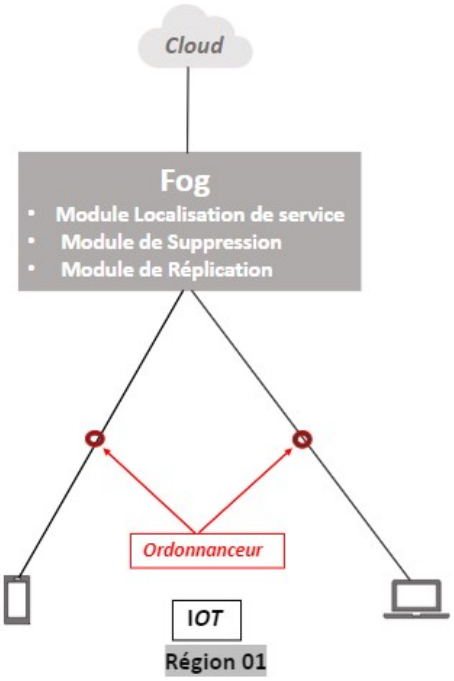

FIGURE 3.2: Schéma de l'Architecture détailler de région 01

3.3.2 Traitement d'une requête :

Dans cette section, nous d'écrivons notre première contribution à savoir une stratégie proposée à partir de l'arrivée de la requête jusqu'au l'exécution de la donnée.

La figure suivante représente un diagramme d'activité de la stratégie proposée.

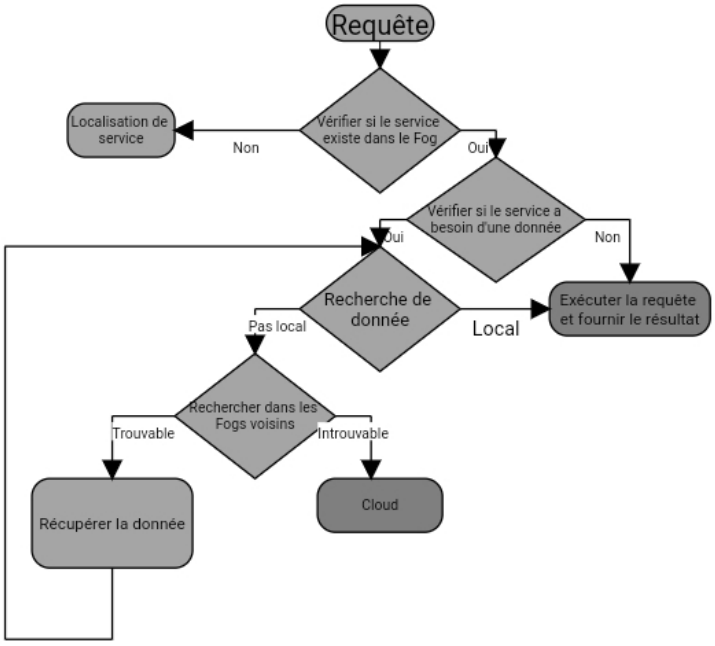

FIGURE 3.3: Diagramme d'activité de la stratégie proposée.

A l'arrivée d'une requête q, l'ordonnanceur vérifié si le service existe dans le meilleur Fog, si le service ne se trouve pas dans le Fog concerné, le gestionnaire de localisation de service est déclenché. Dans le cas où le service se trouve dans le Fog i, et la requête n'a pas besoin d'une donnée, la requête sera traitée dans le Fog i qui héberge le service et le résultat est retourné à l'utilisateur. Si la requête a besoin d'une donnée pour terminer son exécution, le Fog i recherche la donnée demandée. Si la donnée se trouve dans un nœud locale, le service peut accéder localement à la donnée demandée par la requête de l'utilisateur, cette requête sera exécutée dans le Fog i et le résultat sera fourni à l'utilisateur.

Si la donnée demandée ne se trouve pas localement, une recherche est lancé dans les Fogs voisins, si la donnée est trouvée, une estimation de temps de réponse est effectué afin de décider si le processus de réplication serait déclenché ou pas . Si la donnée est introuvable dans les Fog voisins, le processus de réplication est lancé afin de créer une réplique de la donnée dans le Fog à partir du Cloud.

Principe de traitement d'une requête :

Les clients se connectent aux Fogs plus proche de leurs régions. Dans chaque Fog nous mettons un ordonnanceur pour redériger les requêtes des clients selon leurs besoins vers le meilleur Fog.

Après l'arrivé d'une requête d'un client, le Fog vérifie si le service demandé par ce client se trouve localement. Si le service n'existe pas, alors le Fog va rediriger cette requête vers le gestionnaire de localisation de service. Dans le cas où le service est existant, le Fog vérifie si le service demandé a besoin d'une ou des données pour terminer son traitement. Si le service n'a pas besoin d'une donnée, la requête du client sera traitée dans le Fog actuel, sinon le processus de recherche de la donnée est lancé. Dans le cas où la donnée se trouve localement, le Fog va traiter la requête du client localement.

Si la donnée n'est pas trouvée localement, le processus de la recherche va vérifier l'existence de cette donnée dans les Fogs voisins et aussi dans le Cloud. Si la donnée est trouvée, une requête de récupération/réplication est envoyée au site qui héberge cette donnée. Après le traitement de la requête du client, le résultat est envoyé à ce client tout en suivant le chemin emprunté par l'exécution de cette requête.

3.4 La phase de Localisation d'un service :

La Figure 3.3 illustre un diagramme d'activité qui détaille le fonctionnement du gestionnaire de localisation de service.

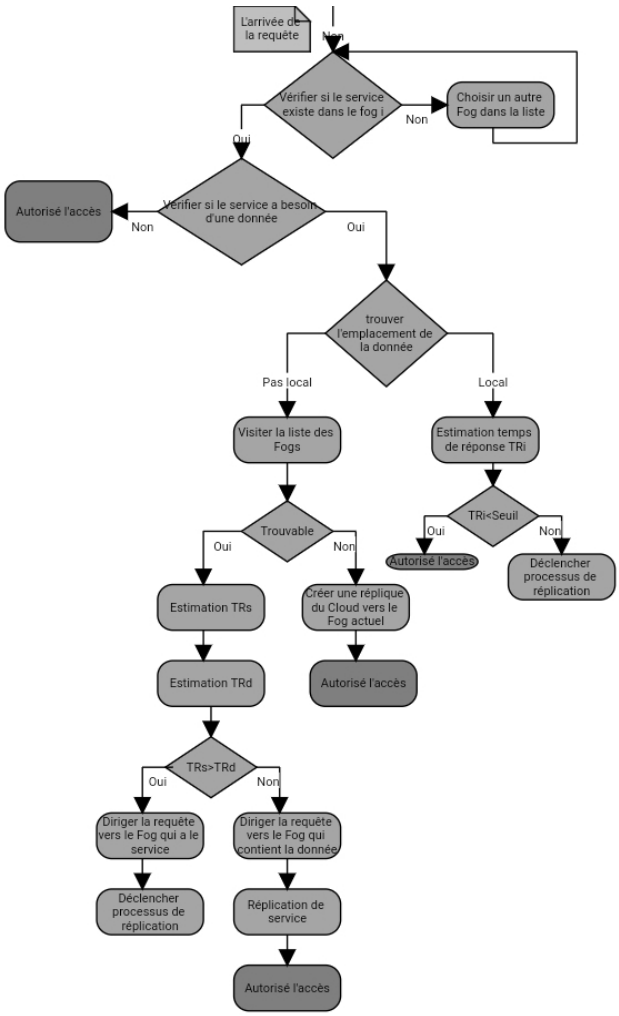

FIGURE 3.4: Diagramme d'activité de localisation.

Quand un service demandé par un client n'existe pas dans le Fog où ce client est connecté, le gestionnaire de localisation de service va être déclencher. Nous supposons que les services demandés par les clients se trouvent au moins dans un site (Cloud et Fogs).

Pour rechercher le service demandé par le client, le gestionnaire de localisation de service va rechercher au début dans la liste des Fogs voisin. Quand le service est trouvé dans le Fog i , le gestionnaire vérifie si une donnée est nécessaire pour l'exécution de ce service. Si aucune donnée est requise, le gestionnaire autorise l'accès pour traiter la requête du client. Si le service a besoin d'une donnée, le gestionnaire va lancer le processus de recherche de cette donnée. Si la donnée se trouve localement, le gestionnaire va estimer le temps de de réponse de la requête *i (TRi)*, si le temps de réponse *TRi* est

inférieur au seuil déjà défini dans le contrat SLA qui permet d'assurer la qualité de service, le gestionnaire va autoriser le traitement de la requête du client.

Dans le cas où l'estimation du temps de réponse est supérieure au seuil, le gestionnaire va déclencher le processus de réplication afin de réduire le temps de réponse. Si la donnée ne se trouve pas localement, le gestionnaire va rechercher dans la liste des Fogs. Si la donnée est trouvée, une estimation du temps de réplication de la donnée et une estimation du temps de réplication de service vont être calculés.

Quand le temps de réplication/migration du service *(TRs)* est supérieur au temps de réplication de la donnée *(TRd)*, le gestionnaire de réplication va diriger la requête vers le Fog qui héberge le service, ensuite le processus de réplication de la donnée est lancé. Si le temps de réplication de la donnée *(TRd)* est supérieur au temps de réplication/migration du service *(TRs)*, le gestionnaire va diriger la requête vers le Fog qui contient la donnée requis par le service demandé par le client, ensuite le processus de réplication/migration du service va être lancé. Une fois cette opération est terminée, le service peut traiter la requête du Client.

Si la donnée n'est pas trouvable dans la liste des Fogs voisin, le gestionnaire va créer une réplique à partir du Cloud.

Pour calculer le temps de réponse d'une requête TRi, nous allons utiliser l'équation suivante :

$$TR_i = T_{envoie_requête} + T_{traitement(CPU)} + T_{attente} + T_{acces_donnée} + T_{envoie_reponse} \quad (3.1)$$

3.4.1 La phase de Réplication :

Cette phase permet la réplication des données dans les autres régions. Dans chaque Fog, un gestionnaire de réplication a une liste des données à répliquer. Le but de ce gestionnaire de réplication est de minimiser les interactions avec le cloud et entre les régions, ce qui permet de réduire le temps de réponse ainsi que l'utilisation de la bande passante du réseau, il permet aussi de minimiser le nombre de violation et aussi permet d'assurer un profit pour les fournisseurs. Le gestionnaire de réplication permet de répliquer les données récupérées via les autres régions ou du cloud vers des régions voisines selon certaines conditions.

Le gestionnaire de réplication est lancé par le gestionnaire de localisation de service dans le cas où le temps de réponse d'une requête est supérieur au seuil définie dans le SLA. L'organigramme suivant détaille le fonctionnement du processus de réplication des données.

La Figures 3.5 illustrent le diagramme d'activités qui représente le processus de la réplication.

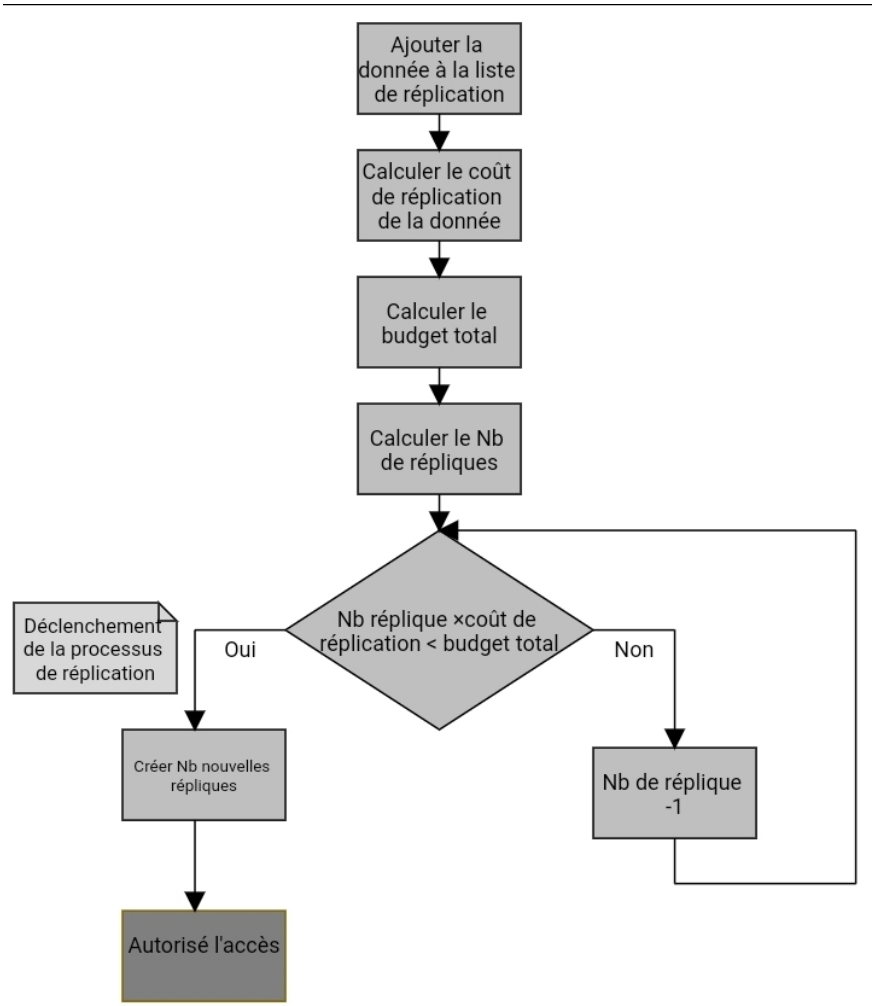

FIGURE 3.5: Diagramme d'activité de la réplication.

Afin de garantir un profit pour le fournisseur, nous allons calculer le coût engendré par le processus de réplication de la donnée, ensuite nous allons calculer le budget total payé par les utilisateurs, après nous allons déduire le nombre de réplique à créer, le nombre total de réplique à créer ne doit pas dépasser le seuil de réplication S qui est égale à la racine carrée du nombre des requêtes. Si le budget permet de créer autant de répliques que le seuil de réplication S, nous allons nous contenter à créer S répliques et c'est le cas le plus optimal.

Si le budget ne permet pas de créer aucune réplique, nous allons créer quand même une seule réplique afin de minimiser le coût de violation qui est dû à l'échec du traitement

des requêtes des utilisateurs, cette situation représente le pire des cas où le fournisseur va perdre un peu de ses bénéfices.

Dans le cas où le budget ne permet pas de créer S répliques, nous allons réduire le nombre de répliques jusqu'à ce que ce nombre garanti un bénéfice pour le fournisseur.

3.4.2 La phase de Suppression :

Les données expirées vont être supprimées dans le but d'optimiser l'utilisation de l'espace disque des nœuds et pour préserver la cohérence de notre système La suppression déclenche la diffusion de ces changements au nœuds Fog concernées. La Figure 3.6 illustrent les diagrammes d'activités qui détaillent le principe de fonctionnement de la suppression et de la mise à jour des données.

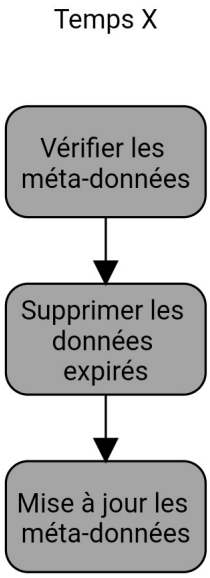

FIGURE 3.6: Diagramme d'activité de vérification et de suppression des données.

3.4.3 Conclusion :

Au cours de ce chapitre, nous avons décrit le fonctionnement de notre stratégie. La stratégie proposée gère le problème de la réplication des données, cette stratégie est proposée dans un enivrement du Cloud et Fog comuting, l'utilisation de l'environnement du Fog Comuting mène à une diminution importante du temps de latence puisque les interactions avec les Clouds vont être réduites. Nous avons aussi utilisé l'ordonnancement

des requêtes afin de proposer une stratégie dynamique et performante. L'ordonnancement permet d'affecter les requêtes vers les meilleurs Fogs afin de minimiser le temps d'attente et de transfert ce qui a un effet important sur le temps de réponse. La stratégie de réplication proposée est basée sur un modèle économique qui permet d'améliorer la performance en terme de temps de réponse et de garantir un gain pour les fournisseurs.

Dans le prochain chapitre nous allons implémenter notre stratégie en utilisant le simulateur iFogSim afin d'évaluer l'efficacité de notre stratégie en la comparant avec d'autres solutions existantes..

CHAPITRE 4

SIMULATION

4.1 Introduction :

Dans le chapitre précèdent, nous avons présenté notre stratégie d'ordonnancement et de réplication de données dans les environnement Fog et Cloud computing et nous avons détailler son fonctionnement. Ce chapitre est consacré à la phase de mise en œuvre de cette stratégie dans le but d'évaluer les performances et de valider la stratégie proposée. Pour cela, nous avons étendu le simulateur IFogSim qui est un simulateur développée en java. Nous avons réalisé plusieurs séries d'expérimentations dont les résultats et les interprétations font l'objet de ce chapitre.

4.2 Environnement de Développement :

Pour implémenter et tester notre stratégie d'ordonnancement et de réplication de données dans les environnement du Fog et Cloud Computing, nous avons développé le simulateur IFogSim afin de supporter notre proposition. Nous avons utilisé l'environnement du travail suivant pour cette implémentation :

— **Caractéristiques matérielles et logicielles de l'ordinateur utilisés :** Nous avons développe notre application sur une machine avec un processeur Intel(R) Core(TM) i3-4005U CPU @ 1.70GHz et d'une capacité mémoire de 4Go. Le simulateur est sous Windows10 64 bits.

Simulateur utilisé : iFogSim.

Langage utilisée : Java.

Version Java Développement Kit utilisée : JDK 11.

IDE utilisé : Eclipse.

4.3 Langage de programmation Java :

Java est un langage de programmation informatique orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien de Bill Joy (cofondateur de Sun Microsystems en 1982), présenté officiellement le 23 mai 1995 au SunWorld. La société Sun a été ensuite rachetée en 2009 par la société Oracle qui déteint et maintient désormais Java [27].

Le langage JAVA a la particularité principale que les logiciels écrits dans ce langage peuvent être très facilement portables sur plusieurs systèmes d'exploitation tels que UNIX, Windows, Mac OS ou GNU/Linux, avec ou sans modifications. C'est la plate-forme qui garantit la portabilité des applications développées en Java [27].

Java est devenu aujourd'hui une direction incontournable dans le monde de la programmation [27].

Eclipse est un environnement de développement intégré (EDI), placé en Open Source par Sun en Juin 2000. En plus de Java, Eclipse permet également de supporter différents autres langages, comme Python, C, C++, JavaScript, XML, Ruby, PHP et HTML téléchargeable du site : https :https ://www.eclipse.org/downloads. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web).

Conçu en Java, Eclipse est disponible sous Windows, Linux, Solaris, MacOS ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java).

De plus, Eclipse est écrit en Open Source, téléchargeable directement du site http ://java.sun.com. Il est puissant et compatible avec toutes les nouvelles technologies Java (les technologies Java EE, les bases de données, UML, XML, ...).

4.4 Extraire des fonctionnalités :

Le simulateur iFogSim est développée sur le cadre fondamental de CloudSim [24]. CloudSim est l'un des simulateurs les plus largement adoptés pour modéliser les environnements de cloud computing.

En étendant l'abstraction des classes CloudSim de base, iFogSim propose des étendues pour simuler un environnement informatique Fog personnalisé avec un grand nombre de nœuds Fog et de dispositifs IoT (par exemple capteurs, actionneurs) .

Cependant, dans iFogSim, les classes sont annotées de telle manière que les utilisateurs, n'ayant aucune connaissance préalable de CloudSim, peuvent facilement définir l'infrastructure, le placement de service et les politiques d'allocation des ressources pour le calcul du Fog.

iFogSim applique le modèle de flux de données Capter-Traiter-Actionner et distribué tout en simulant n'importe quel scénario d'application dans un environnement informatique Fog.

Il facilite l'évaluation de la latence de bout en bout, de l'encombrement du réseau, de la consommation d'énergie, des dépenses opérationnelles et de la satisfaction QoS (quality of service).

Dans un nombre important de travaux de recherche, iFogSim a déjà été utilisé pour simuler les ressources , la mobilité , la latence, la qualité de l'expérience (QoE) [23], l'énergie [25], la sécurité et Gestion QoS [24] de l'environnement informatique Fog.

iFogSim est composé de 3 composants de base :

— **Composants physiques :** ce sont les modèles des équipements physiques trouvés dans une infrastructure de Fog (par exemple serveurs, capteurs, actionneurs, ...) :

  (i) **FogDevice :** cette entité modélise les nœuds de Fog (par exemple les switchs, les routeurs, les stations de base, les passerelles, ...). Elle spécifie les caractéristiques matérielles d'un nœud de Fog comme le modèle du processeur, la taille de la RAM, la capacité du stockage et la bande passante du réseau.

  (ii) **Capteur :** cette entité modélise les différents capteurs déployés dans l'infrastructure simulée. Elle spécifie les attributs d'un capteur allant de sa connectivité réseau jusqu'aux données capturées.

  (iii) **Capteur :** cette entité modélise les actionneurs et les effets de leur action sur l'environnement simulé. Elle spécifie les attributs d'un actionneur comme sa connectivité, sa latence et la passerelle associée.

— **Capteur :** Les modules d'application (AppModules) et les bords d'application (AppEdges) sont les composants logiques d'iFogSim. Dans iFogSim, les applications sont considérées comme une collection d'AppModules interdépendants qui promeut par conséquent le concept d'application distribuée [26].

  La dépendance entre deux modules est définie par les caractéristiques de AppEdges. Dans le domaine du cloud computing, les AppModules peuvent être mappés avec des machines virtuelles (VM) et les AppEdges sont le flux de données logique entre deux machines virtuelles. Dans iFogSim, chaque AppModule (VM) traite un type

particulier de tuples (tâches) provenant du prédécesseur AppModule (VM) du flux de données .

La transmission de tuple entre deux AppModules peut être périodique et à la réception d'un tuple d'un type particulier, le fait qu'un module déclenche un autre tuple (type diffèrent) vers le module suivant est déterminé par le modèle de sélectivité fractionnaire [26].

— **Composants de gestion :** Le composant de gestion d'iFogSim est constitué d'objets de mappage de contrôleur et de module [26]. L'objet Module Mapping selon les exigences des AppModules, identifie les ressources disponibles dans les appareils Fog et les place à l'intérieur . Par défaut, iFogSim prend en charge le placement hiérarchique des modules. Si un appareil Fog n'est pas en mesure de répondre aux exigences d'un module, le module est envoyé à l'appareil Fog de niveau supérieur. L'objet Controller lance les AppModules sur leurs appareils Fog affectés en suivant les informations de placement fournies par l'objet Module Mapping et gère périodiquement les ressources des appareils Fog [26]. Une fois la simulation terminée, l'objet Controller collecte les résultats des coûts, de l'utilisation du réseau et de la consommation d'énergie pendant la période de simulation à partir des appareils Fog .

4.5 Métriques utilisées :

Pour évaluer le comportement de la stratégie proposée, nous avons utilisé les métriques suivantes :

4.5.1 Temps de réponse :

Cette métrique représente la moyenne de temps de réponse pour toutes les requêtes. Elle est calculée comme suit :

$$TR_m = \frac{TR_i}{Nb_r} \tag{4.1}$$

Avec :

TRm : Temps de réponse moyen.

Nbr : Nombres des requêtes.

TRi : Temps de réponse représente le temps écoulé entre l'envoi d'une requête et la réception d'une réponse, il est calculé comme suit :

$$TR = T_{réponse} - T_{envoie} \tag{4.2}$$

T réponse : Temps de réception d'une réponse à la requête.

T envoie : Temps d'envoi de la requête.

4.5.2 Effet du nombre de requêtes sur le temps de réponse moyen :

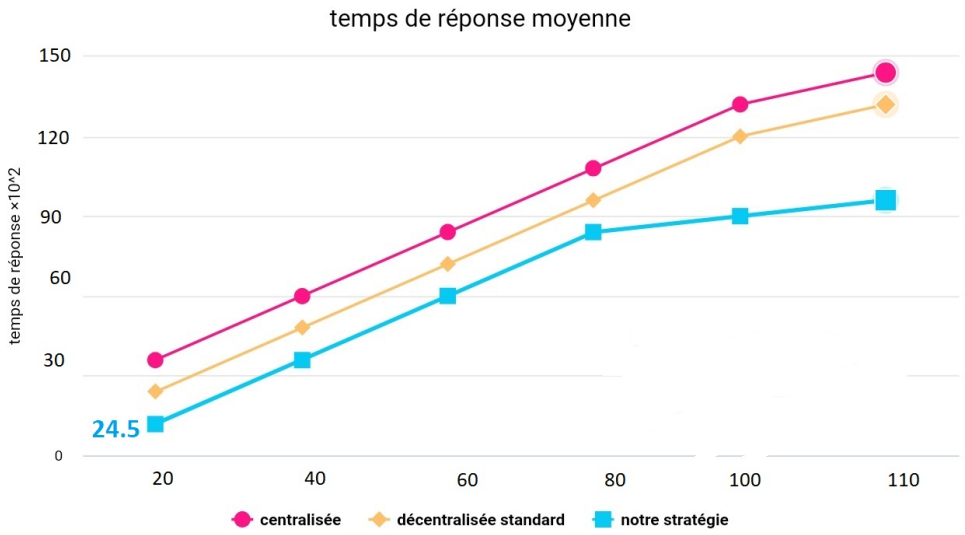

FIGURE 4.1: Graphique linéaire pour l'effet du nombre de requêtes sur le temps de réponse moyen.

Dans cette série de simulation, nous avons étudié l'effet de la variation du nombre de requêtes sur le temps de réponse moyen. Nous avons constaté que le temps de réponse pour l'approche centralisée est très élevé quelle que soit le nombre de requêtes, car la latence entre le Cloud et les IoTs est élevée. Par contre pour les deux autres approches décentralisées nous avons remarqué une baisse du temps de réponse avec l'augmentation du nombre de requêtes ce qui peut être justifié par le rapprochement des données aux utilisateurs. Concernant notre approche, nous avons remarqué la même baisse du temps de réponse que les approches décentralisées mais cette baisse est plus importante.

4.5.3 Effet de la taille des fichiers sur le temps de réponse moyen :

Le but de cette série de simulations est d'étudier l'impact de la taille des fichiers sur le temps de réponse moyen, pour cela , nous avons générer plusieurs fichier de taille différente, la taille des fichiers varie entre 100 et 4096 par un pas de 1024.

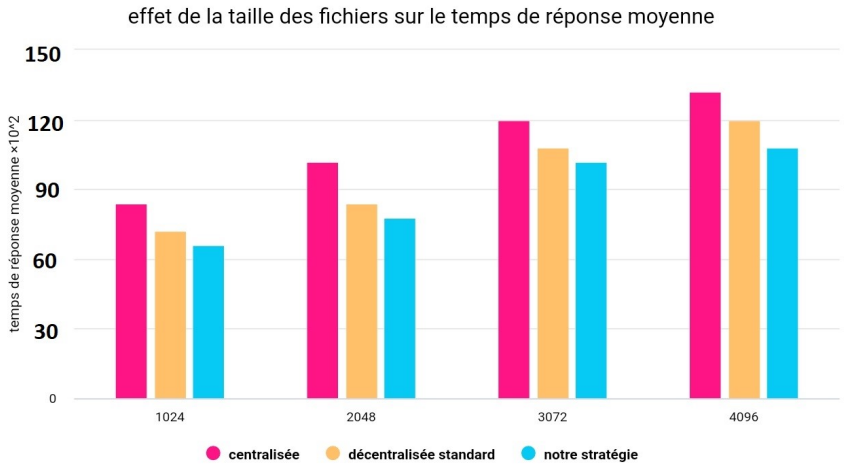

FIGURE 4.2: Graphique à barres pour l'effet de la taille des fichiers sur le temps de réponse moyen.

Nous remarquons que le changement de la taille des fichiers a un effet sur le temps de réponse moyenne qui augmente avec l'augmentation de la taille des fichier dans le cas de toutes les stratégies. Nous constatons une baisse du temps de réponse pour les autres approches par rapport à l'approche centralisée, cette baisse est plus remarquable dans le cas de notre approche.

4.6 Consommation d'énergie moyenne :

Cette métrique représente la moyenne de la consommation d'énergie. Elle est calculée comme suit :

$$CE_m = \frac{CE_i}{N} \tag{4.3}$$

Avec :

CEm : Consommation d'énergie moyenne.

N : Nombre des nœuds.

CEi : Consommation d'énergie du nœuds i.

4.6.1 Effet du nombre de requêtes sur la consommation d'énergie moyenne :

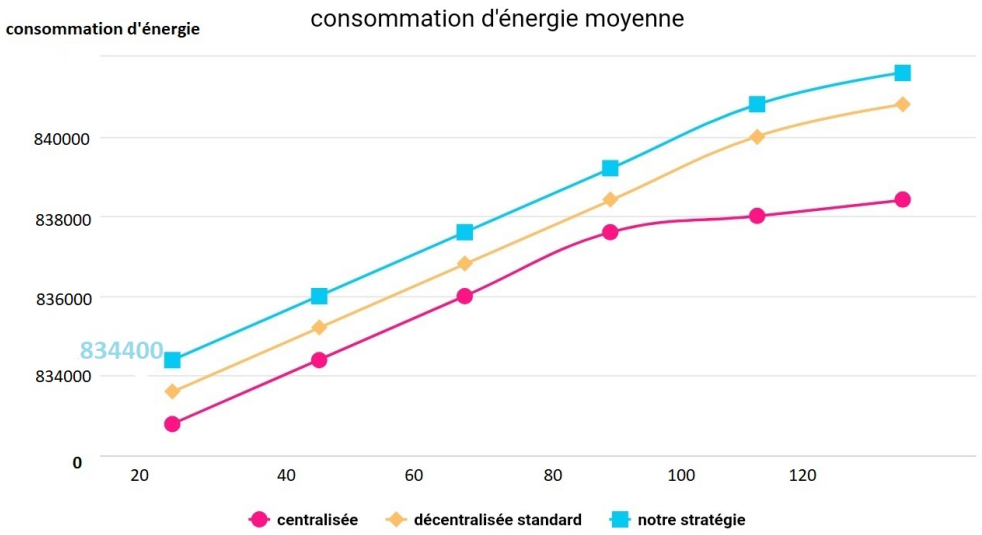

FIGURE 4.3: Graphique linéaire pour l'effet du nombre de requêtes sur la consommation d´énergie moyenne.

Afin d'étudier l'effet de la variation du nombre de requêtes sur la consommation d'énergie moyenne. Nous avons lancer plusieurs simulations. Les résultats de ces séries de simulations nous a permet de constater que la consommation d'énergie moyenne de l'approche centralisée est basse et presque constante quel que soit le nombre de requêtes. Par contre, dans les approches restantes nous notons une consommation moyenne élevée par rapport l'approche centralisée.

Ceci peut être expliqué par l'augmentation du nombre des nœuds qui ont été ajouté dans la couche Fog. Cependant nous remarquons que notre approche reste efficace par rapport à l'approche décentralisée . La consommation dénergie moyenne est plus proche de celle de l'approche centralisée malgré le nombre de noeuds très élevé dans notre topologie.

4.6.2 Effet de la taille des fichiers sur la consommation d'énergie moyenne :

Dans cette série de simulations, nous avons étudié l'impact de la taille des fichiers sur la consommation d'énergie moyenne, nous avons varié la taille des fichiers entre 100

et 4096 par un pas de 1024.

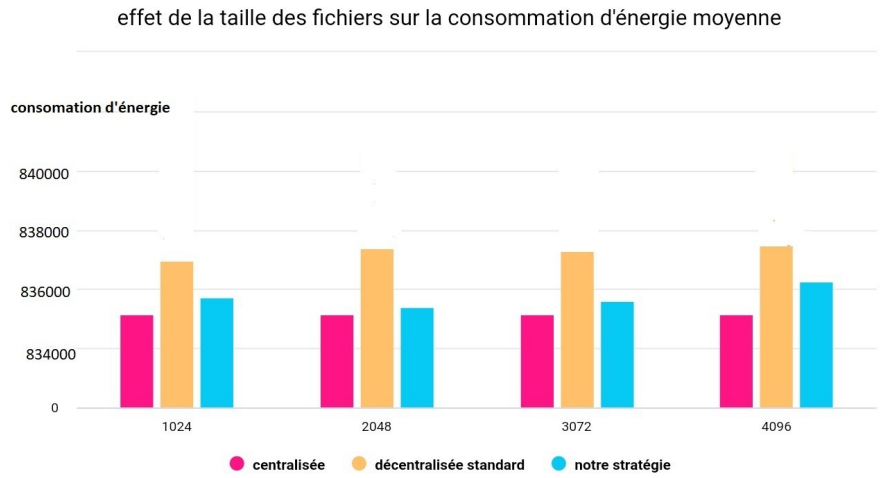

FIGURE 4.4: Graphique à barres pour l'effet de la taille des fichiers sur la consommation d´énergie.

montrent que la variation de la taille des fichiers a un effet très léger sur la consommation d'énergie moyenne dans l'approche centralisée. Cette consommation reste toujours basse par rapport aux autres approches. Cependant nous remarquons que les déplacements des utilisateurs et la variation de la taille des fichiers n'ont pas une grande influence sur la consommation de l'énergie moyenne pour notre approche qui reste très faible et elle se rapproche de la consommation d'énergie moyenne de l'approche centralisée.

4.7 Utilisation de la bande passante :

Cette métrique représente la quantité de données transmises sur le réseau durant la simulation en mo.

4.7.1 Effet du nombre de requêtes sur l'utilisation de la bande passante :

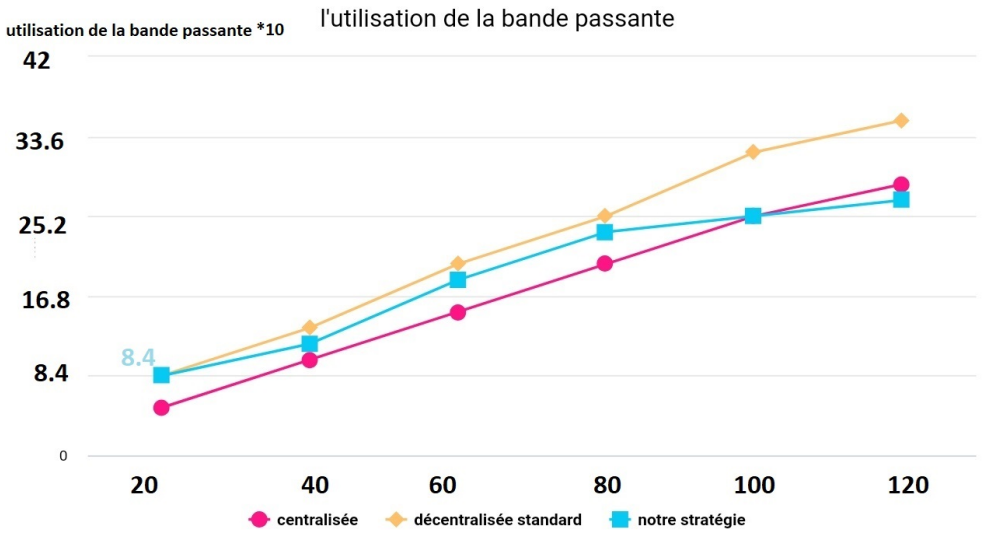

FIGURE 4.5: Graphique linéaire pour l'effet du nombre de requêtes sur l'utilisation de la bande passante.

La figure 4.7 et 4.8 représente les résultats de simulation de la variation du nombre de requêtes et son effet sur l'utilisation de la bande passante. L'utilisation de la bande passante de l'approche centralisée augmente d'une façon linéaire avec l'augmentation du nombre des requêtes. Par contre l'utilisation de la bande passante des approches décentralisées est légèrement supérieure à l'utilisation de l'approche centralisé quand le nombre de requêtes est faible.

Nous remarquons que la différence d'utilisation de la bande passante diminue quand le nombre de requêtes augmente dû au fait qu'initialement les données se trouve dans le Cloud. Nous notons que l'utilisation de la bande passante de notre approche est supérieure à celle de l'approche centralisée, cette utilisation devient inférieur quand le nombre de requêtes dépasse 100 requêtes.L'efficacité de notre approche est claire quand le nombre de requêtes est important.

Notre approche est meilleurs que l'approche décentralisée quel que soit le nombre de requêtes.

Au début, la réplication de données génère un trafic réseau important, mais l'efficacité de la réplication de données augmente avec l'augmentation du nombre de requêtes, donc

les clients ont plus de chance d'accéder localement ou dans la même région aux données demandées ce qui permet de réduire l'utilisation de la bande passante.

4.7.2 Effet de la taille des fichiers sur l'utilisation de la bande passante :

Dans cette série de simulation, nous avons étudié l'impact de la taille des fichiers sur l'utilisation de la bande passante, nous avons varié la taille des fichiers entre 100 et 4096 par un pas de 1024.

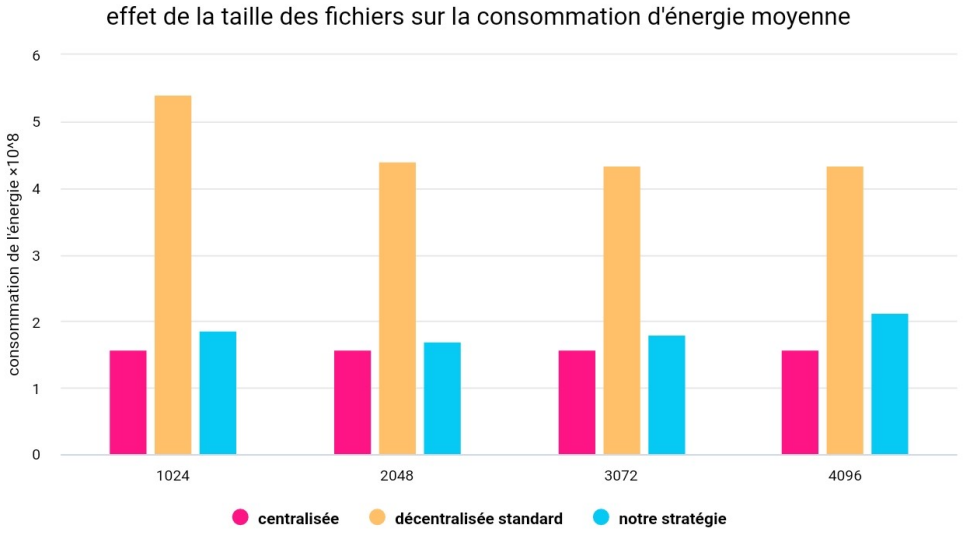

FIGURE 4.6:

Graphique à barres effect de la taille des fichiers sur l'utilisation de la bande passante.

Les résultats de simulation présenté dans la figure 4.9 montrent que la variation de la taille des fichiers a un impact faible sur l'utilisation de la bande passante pour l'approche centralisée parce que les données et les services se trouvent dans le Cloud.

Les deux approches décentralisées se montrent efficace en terme d'utilisation de la bande passante quand la tailles des fichiers est inférieure a 2048mo, par contre quand la taille des fichiers est supérieure à 2048 mo nous notons que l'utilisation de la bande passante devient largement supérieure à celle de l'approche centralisée, ces deux approches n'utilisent pas le mécanisme de réplication et donc l'accès aux données devient trop coûteux quand la taille des fichiers demandées est grande.

Nous constatons que l'utilisation de la bande passante pour notre approche est inférieure aux autres approches grâce à la réplication de donnée, mais quand la taille

des fichiers dépasse 3072 mo, nous remarquons que l'utilisation de la bande passante devient légèrement supérieure à l'approche centralisée mais qui reste toujours inférieur par rapport aux approches décentralisées.

4.8 Indice de performance :

Cette métrique sert à déterminer l'efficacité de chaque approche selon la consommation d'énergie moyenne, la charge du réseau et le temps de réponse au même temps. Plus l'indice de performance est bas plus l'approche est bonne. Il est calculé comme suit :

$$IP_j = \frac{V_{nij} * P_i}{P_i} \tag{4.4}$$

Avec :

IPj : Indice de performance.

Nbm : Nombres des métriques.

Pi : Le poids du métrique i.

Vnij : Valeur j de la métrique i normalisée, elle est calculée comme suit :

$$V_{nij} = \frac{V_{ij} - Min_{Evi}}{Max_{Evi} - Min_{Evi}} \tag{4.5}$$

Dans nos expérimentations, nous avons donné à chaque métrique un poids égale à 1, ce qui veut dire que toutes les métriques précédentes ont la même importance.

4.8.1 Effet du nombre de requêtes sur l'indice de performance :

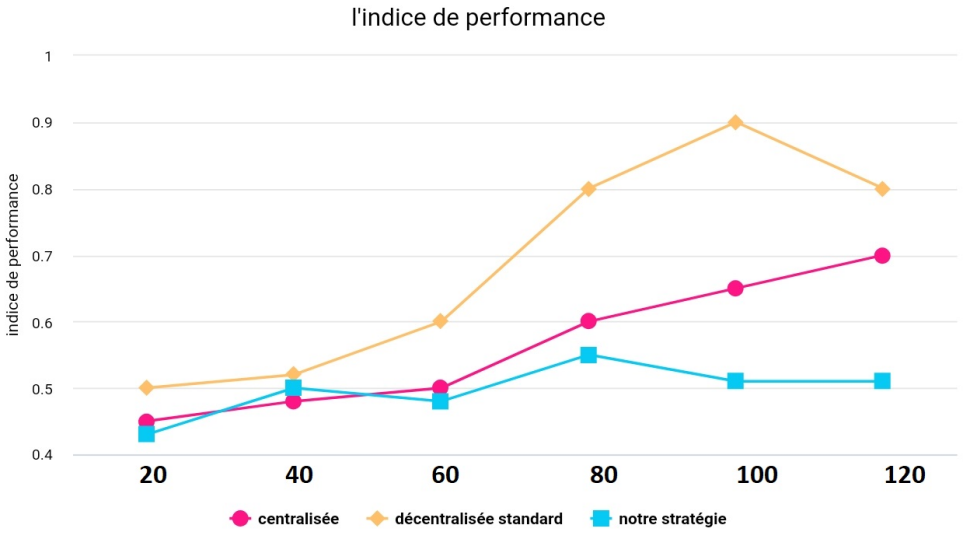

FIGURE 4.7: Graphique linéaire pour l'effet du nombre de requêtes sur l'indice de performance.

La figuré 4.10 et 4.11 représentent les résultat de simulation de l'effet de la variation du nombre de requêtes sur l'indice de performance pour les différentes approches. Nous constatons que l'indice de performance de notre approche est le plus bas par rapport aux autres approches. le nombre de requêtes est important, notre approche devient plus efficace par rapport aux autres.

4.8.2 Effet de la taille des fichiers sur l'indice de performance :

Dans cette série de simulations, nous avons étudié l'impact de la taille des fichiers sur l'indice de performance, nous avons varié la taille des fichiers entre 100 et 4096 par un pas de 1024.

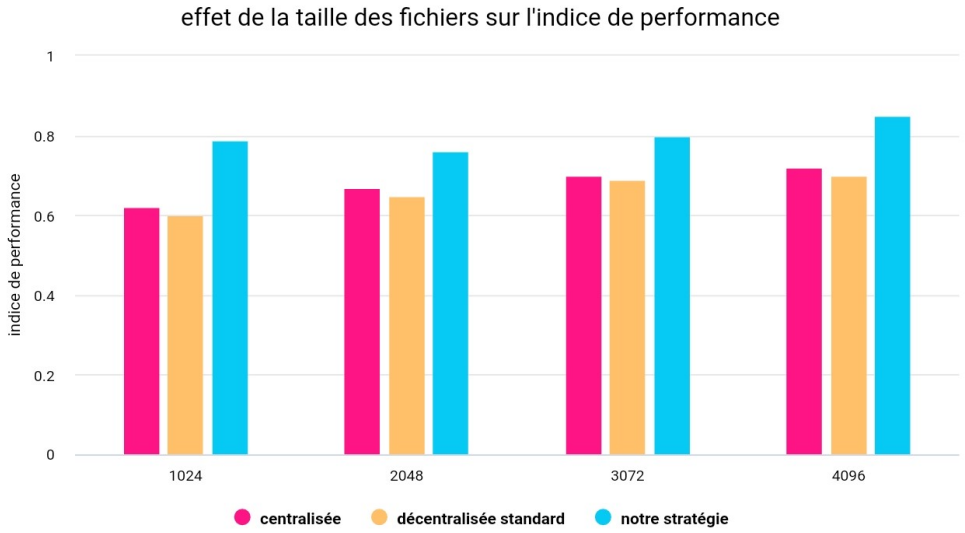

FIGURE 4.8: Graphique à barres pour l'effet de la taille des fichiers sur l'indice de performance.

Nous constatons que notre approche possède toujours l'indices de performance le plus bas par rapport aux autres approches quelle que soit la taille des données mais nous remarquons une légère augmentation quand la taille des fichiers devient plus grande, qui peut s'expliquer avec l'augmentation de la consommation d'énergie ou de l'utilisation de la bande passante.

4.9 Conclusion :

Dans ce chapitre, nous avons présenté notre environnent de travail et nous avons détaillé l'outil de simulation iFogSim que nous avions étendu afin de simuler notre approche de gestion de donnée dans l'environnement du Cloud et Fog Computing ainsi que les différentes approches.

Nous avons simulé plusieurs scénarios afin d'évaluer notre proposition en utilisant plusieurs paramètres de configuration. L'analyse des résultats de simulations obtenus montre que l'approche proposée a permis de diminuer le temps de réponse moyen quand le nombre de requêtes est élève, elle a permis aussi d'optimiser la consommation de l'énergie et de réduire l'utilisation de la bande passante malgré l'augmentation de la taille des données demandées .

Notre approche offre aussi un meilleur indice de performance qui représente un compromis entre les trois métriques citées auparavant.

Conclusion Générale

Les applications de traitement de données intensives traitent des volumes extrêmement importants. La gestion traditionnelle de ces données s'appuie sur des solutions basées sur le Cloud pour fournir un ensemble centralisé et riche. Les avantages des frameworks basés sur le cloud sont leur omniprésence, leur capacité de ressources illimitée, leur rapport coût / efficacité, ainsi que leur élasticité. Cependant, accéder aux données depuis le cloud implique un trafic réseau important, des latences de données élevées et des risques de sécurité plus élevés. A la différence du Cloud Computing, le Fog Computing est au plus près des objets connectés. Il facilite le stockage et l'analyse de données.

Dans le travail présent, nous avons étudié le problème de réplication des données Dans le Fog Computig et dans le cloud Computing. Dans ce contexte, la réplication est un moyen essentiel dans le Cloud/Fog Computing. Elle fournit les moyens nécessaires pour garantir la disponibilité des données sur plusieurs copies, l'accès rapide via des copies locales de données, la tolérance aux pannes en répliquant les données sur plusieurs sites. Nous avons utilisé l'ordonnancement dans notre stratégie afin d'optimiser les performances. L'ordonnancement permet de réduire le temps d'accès aux données et de minimiser aussi le temps d'exécution des tâches en affectant les requêtes vers les meilleurs Fogs.

La stratégie proposée utilise l'ordonnancement qui vise à localiser les Fogs les plus performants et qui hébergent les données requises pour le traitement des requêtes des utilisateurs. Ce qui permet de réduire le temps de réponse moyen. La stratégie de réplication de données vise à calculer le nombre minimum de copies nécessaire pour maintenir une haute performance, de diminuer le nombre des violations et de garantir un bénéfice pour les fournisseurs. Nous avons évalué notre stratégie en réalisant plusieurs séries d'expériences en variant plusieurs paramètres. Les résultats montrent la supériorité de notre stratégie par rapport aux approches centralisées et décentralisées

standard.

Pour mettre en évidence notre approche, nous avons étendu le simulateur iFogSim, nous avons lancé plusieurs séries de simulations en créant plusieurs scénarios. Nous avons utilisé un ensemble de métriques pour évaluer les performances, telles que le temps de réponse, la consommation d'énergie et l'utilisation de la bande passante.

D'autre part, le choix de l'architecture cloud est un facteur clé dont dépend la performance attendue du système. Dans ce contexte, nous avons proposé une solution pour assurer la cohérence des données entre l le cloud computing et le fog computing , nous avons défini Ecrivez, lisez, stockez et copiez des protocoles, ce qui peut également garantir la disponibilité et réduire le temps d'accès aux données. Ce travail ouvre des perspectives intéressantes, telles que la vérification de la proposition et sa simulation et l'évaluation de ses performances.

BIBLIOGRAPHIE

[1] Zahraoui.N fondamentaux du Cloud computing : le point de vue des grandes entreprises.

[2] T.V. Rao, A. Khan, M. Maschendra, and M.K. Kumar. A paradigm shift from cloud to fog computing. International Journal of Science, Engineering and Computer Technology, 5(11) :385, 2015.

[3] S. Sarkar, S. Chatterjee, and S. Misra. Assessment of the suitability of fog computing in the context of internet of things. IEEE Transactions on Cloud Computing, PP(99) :46–59, 2015. ISSN 2168-7161. doi : 10.1109/TCC.2015. 2485206.

[4] Fog Computing. the internet of things : Extend the cloud to where the things are. Cisco White Paper, 2015.

[5] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the internet of things. In workshop on Mobile cloud computing. ACM, 2012

[6] Sunyaev, A., Fog and Edge Computing, Springer International Publishing, Cham, 2020, pp. 237–264.

[7] Atlam, H., Walters, R., and Wills, G., “Fog Computing and the Internet of Things : A Review,”Big Data and Cognitive Computing, Vol. 2, No. 2, Avril 2018, pp. 10.

[8] Arindam Das and Ajanta De Sarkar,?On Fault Tolerance of Resources in Com putational Grids, International Journal of Grid Computing and Applications, Vol.3, No.3, DOI : 10.5121/ijgca.2012.3301. Sept. 2012,

[9] Ciobanu (Iacob) Nicoleta ? Magdalena, Ciobano (Defta) costenelaLuminita, Synchronous partial replication case study : implementing elearning platform in an academic environment, Procedia - Social and Behavioral Sciences 46 (2012) 1522-1526 Elsevier.

[10] Alouache Lilia, Ait Mesbah Sofia, Solution de disponibilité dans les environnements Cloud Computing. Mémoire de Master en Informatique. Option : Réseaux et systèmes distribués : Université A/Mira de Bejaia, 2015.

[11] Nagamani H Shahapure, P Jayarekha, Replication, A Technique for Scalability in Cloud Computing, International Journal of Computer Applications Volume 122 ? No.5, (0975 ? 8887),(Juillet 2015).

[12] I. Al Ridhawi, N. Mostafa, W. Masri, Location-Aware Data Replication in Cloud Computing SystemsWireless and Mobile Computing, Networking and Communication, Abu Dhabi, UAE, pp20-27, 19-21 Octobre 2015.

[13] G. Shafer, A mathematical theory of evidence. Princeton Univ. Press, 1975.

[14] G. da Silva, M. Holanda, A. Araujo, Data Replication Policy in a Cloud Computing Environment, 2016, URL : http ://ieeexplore.ieee.org/document/7521383/.

[15] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, ?The Hadoop distributed file system, ? in Proceedings of the IEEE 26th Symposium on Mass Storage Systems and Technologies, pp. 1 ?10, USA, 2010.

[16] Mohamed-K HUSSEIN, Mohamed-H MOUSA, A Light-weight Data Replication for Cloud Data centers environment, International Journal of Innovative Research in Computer, and Communication Engineering, Vol. 2, Issue 1, pp. 2392-2400, 2014.

[17] N. Santhi, R. Selva Kumar, Consistency and Privacy Based Replication System in Cloud Sharing Framework.International Journal of Current Research and Modern Education (IJCRME) 2455 ? 5428 Volume I, Issue I, 2016.

[18] G. J. Chang, D. D-Liu, X. Zhu, Distange graphs and T-Coloring, International journal of Combinatorial Theory, Series B 75, 259-269, 1999.

[19] L. M. Vaquero, L. Rodero-Merino, Finding your Way in the Fog : Towards a Comprehensive Definition of Fog Computing, 2014.

[20] D. Boru, D. Kliazovich, F. Granelli, P. Bouvry, A.Y. Zomaya, Energy-efficient data replication in Cloud Computing datacenters, Springer Cluster Computing, vol 18, n1, pp 385-402, 2015.

[21] Bonomi, F. and Milito, R., “Fog Computing and its Role in the Internet of Things,” Proceedings of the MCC workshop on Mobile Cloud Computing , Août 2012.

[22] CISCO, “Cisco Fog Computing Solutions : Unleash the Power of the Internet of Things, White Paper,”Tech. rep., 2015.

[23] Mahmud, M., Srirama, S., Ramamohanarao, K., and Buyya, R., “Quality of Experience (QoE)-aware placement of applications in Fog computing environments,” Journal of Parallèle and Distributed Computing, Mars 2018.

[24] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A. F., and Buyya, R., “CloudSim : a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms,”Software : Practice and Experience, Vol. 41, No. 1, 2011, pp. 23–50.

[25] Mahmoud, M., Rodrigues, J., Saleem, K., Al-Muhtadi, J., Kumar, N., and Korotaevf, V., “Towards energy-aware fog-enabled cloud of things for healthcare,” Computers Electrical Engineering, Vol. 67, Mars 2018, pp. 58–69.

[26] Gupta, H., Dastjerdi, A. V., Ghosh, S. K., and Buyya, R.,“iFogSim : A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments,”Softw. Pract. Exp., Vol. 47, 2017, pp. 1275–1296.

[27] Friedel, D. H. and Potts, A., Java Programming Language Handbook, Coriolis Group Books, USA, 1996.