الجمـهورية الجزائرية الديمقراطية الشعبيـة
وزارة التعليم العالي والبحث العلمي
جـامعـة سعـيـدة د. مـولاي الطـاهـر
كـلـيـة التكنولوجيا
قسم: الإعلام الآلي

Mémoire de Master

Spécialité: Réseaux Informatiques ET Systémes Réparties

Thème

La réplication des données dans le fog computing pour l'internet des objets

Présenté par:

Laldji maroua chaimaa

Maata Ines

Dirigé par:

Dr. Said Limam

Promotion 2021 - 2022

Remerciements

Avant tout; nous remercions **Dieu** de nous avoir aidé à faire notre thèse.

Nous tenons à adresser nos chaleureux remerciements à notre encadreur ***LIMAM SAID*** qui nous a aidés à élaborer ce projet par ses conseils et soutient durant toute la période de notre projet.

Nos remerciements à nos enseignants qui ont contribué à notre formation.

Nos parents, pour leur soutien constant et leurs encouragements.

TABLE DES MATIÈRES

Table des abréviation

Directed acyclic graph
DAG

Infrastructure as a Service, des services virtuels disponibles à la demande
IaaS

Internet des objets
IOT

Les systèmes de gestion de bases de données
SGBD

Nœuds terminaux
TN

Platform as a Service, des plateformes de développement prêtes à l'emploi
PaaS

Propose Execution and Benefit Arranged Information Replication Methodology
PEPR,

Seuil de réplication
SR

Software as a Service, des services métiers à la demande
SaaS

Table des figures

Introduction général

L'Internet des objets changera non seulement notre vie quotidienne, mais aussi le monde de l'industrie et des entreprises. D'ici 2025, près de 75,44 milliard d'objets seront connectés [1] et d'énormes quantités des données seront générées (90% des données mondiales ont été générées au cours des deux dernières années [2]).En supposant que ces informations puissent être directement intégrées dans le centre de données ou le cloud, ce sera très simple. De plus, en termes de latence et de saturation de la bande passante, l'envoi des milliards de données sur Internet peut rapidement devenir un problème. C'est là que le cloud computing cède la place au fog computing [3].

Le fog computing, aussi appelé «informatique dans le brouillard», définit une infrastructure chargée de stocker et de traiter des données issues d'objets connectés. On peut le considérer comme un concurrent du cloud, car il peut fournir les mêmes services que le cloud, ou une solution complémentaire à ce dernier, car il réduit la latence pour les objets connectés et réduit les demandes de services de cloud par les appareils IoT.

Le fog computing étant un nouveau domaine de recherche, plusieurs problèmes ont été soulevés. Outre que les problèmes hérités du Cloud, il apporte également de nouveaux défis en matière de confidentialité et de sécurité. Les deux paradigmes ayant des architectures très différentes, implique que les solutions apportées dans le Cloud ne peuvent pas être directement appliquées dans le fog.

Dans notre travail l'objectif visé est de proposer une stratégie, basée sur la réplication des données dans le fog computing pour l'internet des objets.

Notre mémoire s'organise de la manière suivante : nous allons dans un premier temps faire un état de l'art sur les environnements cloud et fog computing, puis nous allons

Présenter aussi un état de l'art sur la technique de la réplication des données et les travaux connexes, ensuite nous allons exposer notre stratégie proposée de réplication de données dans le fog computing en détail.

Enfin, nous finirons avec une conclusion générale pour cloturer notre travail.

Chapitre 1

Cloud/Fog computing

1 Etat de l’art sur cloud/fog computing

1.1 Introduction:

L’internet des objets (IOT) désigne les échanges d’informations et de données numériques entre les objets présents dans le monde réel et le réseau Internet, elle est partiellement responsable de l’accroissement du volume de données générées sur le réseau. En conséquence, l’Internet des Objets apporte de nouvelles contraintes informatiques, tels que la latence et la saturation de la bande passante. Le Cloud computing n’est pas adapté pour répondre aux besoins d’Internet des objets. C’est pourquoi, le nouveau paradigme «Fog computing» a été introduit.

Dans ce chapitre, nous allons présenter le cloud computing, ses Caractéristiques et les services qu’il fournit, ainsi que Les avantages et les inconvénients.Par la suite, nous détaillons le fog computing, son architecture, et son fonctionnementainsi que ses avantages et ses inconvénients.

1.2 Cloud computing:

1.2.1 Définition:

Le terme "cloud" est analogue à "Internet". Le terme "Cloud Computing" est basé sur des dessins de nuages utilisés dans le passé pour représenter les réseaux téléphoniques et plus tard pour représenter Internet.

Le cloud computing est une informatique basée sur Internet où des serveurs partagés virtuels fournissent des logiciels, une infrastructure, une plate-forme, des appareils et d'autres ressources et un hébergement aux clients sur une base de paiement à l'utilisation.Toutes les informations qu'un système numérisé a à offrir sont fournies en tant que service dans le modèle de cloud computing. Les utilisateurs peuvent accéder à ces Minimum d’interaction avec les services disponibles sur le "cloud Internet" sans avoir de connaissances préalables sur la gestion des ressources impliquées. Ainsi, les utilisateurs peuvent se concentrer davantage sur leurs processus métier de base plutôt que de passer du temps et d'acquérir des connaissances sur les ressources nécessaires pour gérer leurs processus métier.Les clients du cloud computing ne sont pas propriétaires de l'infrastructure physique ; ils louent plutôt l'utilisation à un fournisseur tiers. Cela les aide à éviter d'énormes investissements en capital. Ils consomment des ressources en tant que service et ne paient que pour les ressources qu'ils utilisent. La plupart des infrastructures de cloud computing consistent en des services fournis via des ressources partagées. Cela augmente l'efficacité car les serveurs ne sont pas inutilement laissés inactifs, ce qui peut réduire considérablement les coûts tout en augmentant la vitesse de développement des applications [36].

1.2.2 Caractéristiques:

Le cloud computing se distingue des solutions traditionnelles par les caractéristiques suivantes [5]:

– **Large accessibilité via le réseau :** Les services sont accessibles en ligne et sur tout type de support (ordinateur de bureau, portable, smartphone, tablette). Tout se passe dans le navigateur Internet.

– **Mesurabilité du service :** L'utilisation du service par le client est supervisée et mesurée afin de pouvoir suivre le niveau de performance et facturer le client en fonction de sa consommation réelle.

– **Solution multi client :** Une même instance d'un logiciel est partagée par l'ensemble des clients de façon transparente et indépendante. Tous les clients utilisent la même version du logiciel et bénéficient instantanément des dernières mises à jour. Chaque client dispose d'un paramétrage utilisateur qui lui est propre.

– **Disponibilité à la demande :** Le service peut être souscrit rapidement et rendu opérationnel automatiquement avec un fournisseur.

– **Élasticité immédiate des ressources** : Des ressources supplémentaires peuvent être allouées au service pour assurer la continuité du service en cas de pic de charge, ou bien être réallouées à un autre service dans le cas inverse.

– **Mutualisation des ressources :** Des ressources utilisées pour exécuter le service sont mutualisées pour servir à de multiples clients. Les multiples serveurs sollicités, totalement interconnectés, ne forment plus qu'une seule ressource virtuelle puissante et performante.

1.2.3 Services de cloud computing:

1.2.3.1 *Modèles de services:*

Trois grands modèles d'usage du Cloud se dégagent actuellement, tous présentent des caractéristiques différentes comme illustré dans la figure 1.1 [4]:

IaaS (Infrastructure as a Service, des services virtuels disponibles à la demande)

Il s'agit de l'offre la plus basique dans le portefeuille Cloud computing correspondant à la location de capacités de calcul et de stockage.

Dans un service IaaS, le fournisseur met à disposition et administre les ressources matérielles virtualisées comprenant :

– La puissance de calcul.

– Les unités de stockage des données.

– Les réseaux.
– Les couches de virtualisation.
– Les systèmes d'exploitation.

Le client prend en charge la gestion et l'exploitation de toutes les couches supérieures, middlewares, bases de données et applications.

La souscription à une offre IaaS permet au client d'externaliser son parc matériel serveur et de s'affranchir des compétences de conception et d'exploitation des infrastructures techniques. Les prestataires des solutions IaaS les plus connus sont : Amazone avec Amazone Elastic Compute Cloud (EC2), ou Orange Business Service avec Flexible Computing.

PaaS (Platform as a Service, des plateformes de développement prêtes à l'emploi)

Il s'agit de l'offre intermédiaire dans le portefeuille Cloud computing.

Le fournisseur met à disposition une plateforme middleware opérationnelle, incluant des serveurs d'applications, des bases de donnée set les outils permettant au client de développer et de déployer ces propres applications.Cette configuration est très employée pour disposer de plateformes de développement ou de tests disposant de l'ensemble des outils et middleware nécessaires, en évitant ainsi les tâches de construction et de maintenance de ces plateformes non critiques. Elle se destine donc naturellement avant tout aux développeurs. Des services comme Google App Engine, Bungee Connect et Force.com sont des exemples PaaS. Les principaux fournisseurs de PaaS sont : Microsoft avec Azure, Google avec Google App Engine et Orange Business Services.

SaaS (Software as a Service, des services métiers à la demande)

Il s'agit d'une offre « tout compris ». Le prestataire met à disposition une application qu'il administre et configure en majeure partie.Le client externalise ainsi ses applications auxquelles il accède à la demande. Il paie à l'usage, selon lenombre d'utilisateurset/ou letemps d'utilisation dulogiciel. Les prestataires de solutions SaaS les plus connus sont:Google avec Gmail et Youtube ou encore les réseaux sociaux Facebook et Twitter.

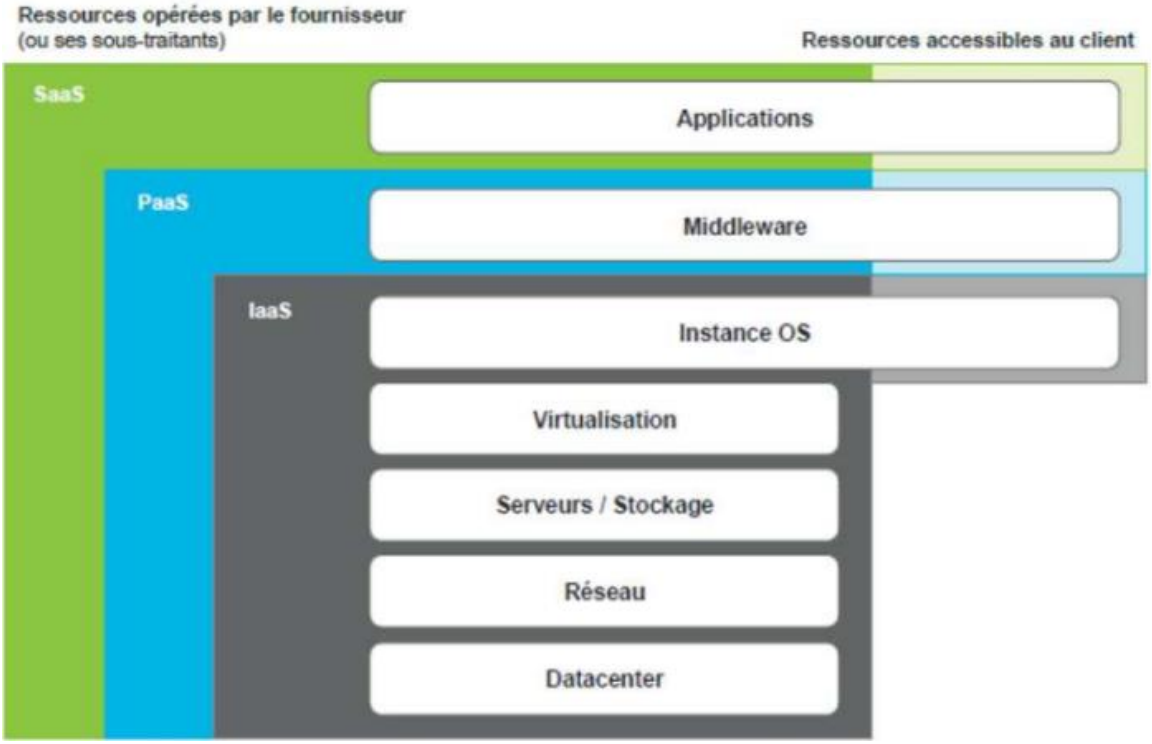

Figure 1.1: Catégories d'offres cloud computing et couches techniques

1.2.3.2 *Modèles de déploiement:*

La manière dont une organisation gère et sécurise les actifs et les besoins de l'entreprise peut se refléter dans la manière dont elle déploie son service cloud. Mais le déploiement du cloud est plus qu'un simple débat « cloud privé contre cloud public ». L'essor du déploiement du cloud hybride a ajouté une toute autre saveur [35].

1. Cloud public

Un cloud public est géré par un fournisseur de cloud IaaS tiers. Les serveurs, le stockage et d'autres ressources numériques sont fournis via Internet. Étant donné que le fournisseur absorbe tous les coûts d'infrastructure et de bande passante, un client n'a besoin que d'un navigateur Web pour accéder au service et gérer les comptes.

Avantages : Service fiable, rentable grâce aux économies d'échelle, pas de maintenance, évolutivité élastique

Inconvénients : souvent jugé dangereux pour le traitement de données hautement privées et sensibles ; doivent respecter des règles de sécurité strictes

2. Cloud privé

Dans un cloud privé, les services, l'infrastructure et la mise en réseau du cloud computing sont exploités uniquement par une organisation indépendante des autres entreprises ou plateformes publiques. Un cloud privé peut être maintenu de deux manières : le centre de données d'une entreprise est physiquement situé

en interne, ou un fournisseur tiers est payé pour tout héberger sur une instance privée.

Avantages : Plus de contrôle, personnalisable, évolutif, flexible, sécurisé

Inconvénients : Plus cher et entretien (si conservé sur place)

3. Nuage hybride

Comme supposé, un déploiement de cloud hybride est un mélange de clouds privés et publics. Cette infrastructure permet aux données, aux informations et aux applications d'être partagées et transférées de manière interchangeable. Le côté privé peut être utilisé pour des processus sensibles tels que les finances et la récupération de données, tandis que le côté public peut exécuter des applications à haut volume

Avantages : Agilité, accessibilité, sécurité améliorées.

Inconvénients : Plus de maintenance, compatibilité complexe.

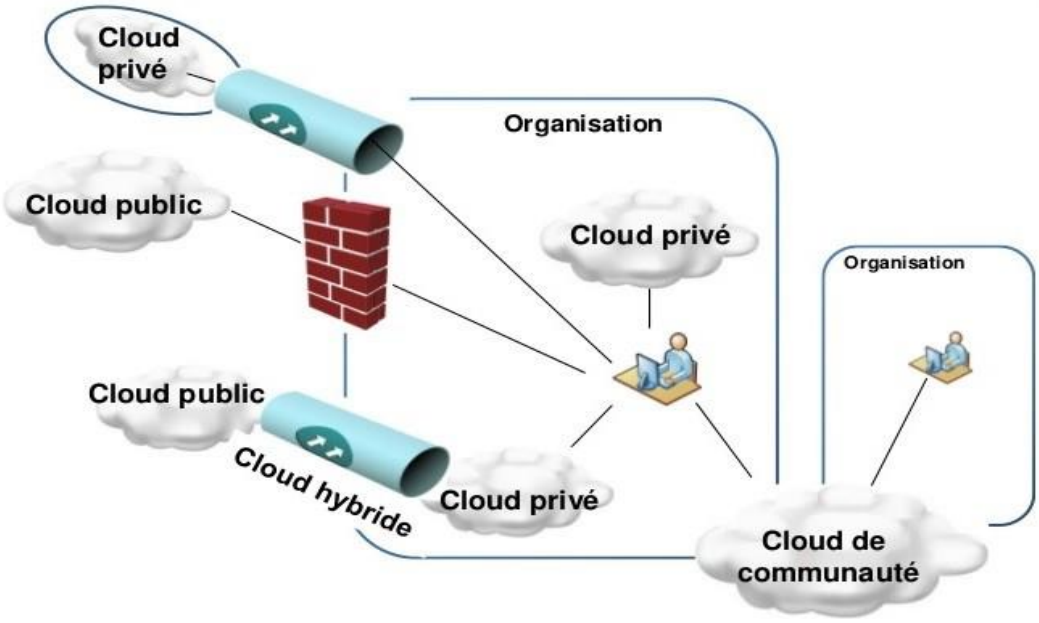

Figure 1.2: modéles de déploiment dans le Cloud computing

1.2.4 Les avantages du cloud:

1-Fiabilité:

Le cloud computing simplifie la sauvegarde des données, la récupération d'urgence et la continuité des activités. Il rend ces activités moins coûteuses, car les données peuvent être mises en miroir sur plusieurs sites redondants au sein du réseau du fournisseur.

2-Coût :

Le cloud computing élimine la nécessité d'investir dans du matériel et des logiciels, et de configurer et de gérer des centres de données sur site : racks de serveurs,

alimentation électrique permanente pour l'alimentation et le refroidissement, experts informatiques pour la gestion de l'infrastructure. La facture est vite salée.

3-Sécurité :

De nombreux fournisseurs de cloud offrent un vaste éventail de stratégies, technologies et contrôles qui renforcent globalement votre situation de sécurité, contribuant ainsi à protéger vos données, vos applications et votre infrastructure contre des menaces potentielles.

1.2.5 Inconvénients du cloud computing:

1. Temps d'arrêt

Alors que de plus en plus d'entreprises s'appuient sur des fournisseurs de services cloud tiers, ces fournisseurs peuvent être surchargés de demandes excessives de clients et peuvent faire face à des arrêts techniques. Comme pour toute panne liée au cloud ou perte de connexion Internet, une entreprise peut s'arrêter avec des applications, des données et des serveurs inaccessibles.

2. Sécurité

Même les marques les plus grandes et les plus connues avec les meilleures pratiques de sécurité ne sont pas complètement protégées contre la compromission de leurs données. Et le stockage d'informations importantes et sensibles sur des clouds de services externes n'est pas non plus une mesure infaillible. Il existe toujours des failles dans les systèmes sensibles, en particulier dans les clouds publics où l'accessibilité est largement ouverte aux pirates, aux utilisateurs négligents et à d'autres vulnérabilités.

3. Contrôle limité

Le cloud décharge une grande partie de la maintenance informatique traditionnelle sur le service cloud. Cependant, cela conduit également à moins de contrôle sur le processus informatique. Le responsable des applications d'une entreprise n'aura accès qu'aux outils de gestion frontale pour les applications, les services et les données, mais pas à l'infrastructure principale.

1.3 Fog computing:

1.3.1 Définition:

Les infrastructures informatiques«en nuage bas», «en brouillard»ou encore appelées infrastructures de Fog Computing ont été proposées par Cisco en 2012 [8] et son aujourd'hui supportées par de nombreux industriels.L'architecture proposée consiste à déployer, non pas des serveurs isolés près des utilisateurs mais de petits centres de données répartis en différents «sites» situés à la périphérie du réseau. Ces centres de

données comportent classiquement une dizaine de serveurs et fournissent des ressources de calcul et de stockage aux clients situés en bordure du réseau. De par leur taille plus importante par rapport à l'Edge Computing, les sites de Fog supportent une meilleure mobilité des utilisateurs tout en améliorant les temps de réponse par rapport à une infrastructure de type «Cloud». Afin de simplifier le déploiement, il a été proposé de placer les serveurs dans les points de présence de l'Internet [9]. Chaque équipement accède à un serveur situé dans son environnement proche. Ce serveur effectue certains calculs localement et délèguent ceux qui sont trop coûteux à un autre serveur situé un peu plus loin de l'utilisateur final. De cette façon, les calculs sont répartis dans l'infrastructure. Les serveurs proches des utilisateurs s'occupent des traitements peu coûteux mais qui nécessitent un faible temps de réponse, jusqu'au Cloud Computing qui effectue les calculs très gourmands en ressources mais pour lesquels les temps de réponses sont moins importants.

Les infrastructures de Fog peuvent également être hiérarchiques, avec une architecture de«Cloud Computing»en hau tdecette hiérarchie [10;11; 12]. L'idée générale est que plus un site de «Fog» est «haut» dans la hiérarchie, plus les ressources qu'il met à disposition ne sont importantes. En contrepartie, la latence pour atteindre ce site est élevée. Le «Cloud» fournit une capacité de stockage et de calcul quasi illimitée au prix d'une latence élevée tandis que le site de Fog le plus proche de l'utilisateur fournit avec une très faible latence, une faible capacité de calcul et de stockage. L'intérêt d'une telle architecture est que lorsqu'un site de Fog a besoin de données ou d'exécuter des calculs sur le «Cloud», l'infrastructure de «Cloud» n'est pas atteinte directement. De nombreux sites sont traversés, permettant de mettre en cache lesdonnées, de les agréger voire de réaliser certaines portions de calculs [13]. De ce fait, le « Cloud » reçoit beaucoup moins de données que lorsqu'il est utilisé directement, permettant un meilleur passage à l'échelle. Dans certains cas, tous les calculs sont effectués avant que l'infrastructure de «Cloud» ne soit atteinte, réduisant le temps de calcul. Illustre cette hiérarchie. L'idée générale du Fog est d'être capable d'effectuer des calculs de Big Data avec des contraintes de latence [14].

1.3.2 Fonctionnement du Fogcomputing:

En fonction de la sensibilité des données, le Fog computing dirigera les données vers le meilleur emplacement [17] :

—Les données les plus sensibles au temps sont analysées sur le nœud Fog le plus proche des éléments qui ont généré les données.

—Les données dont l'action peut prendre quelques secondes ou quelques minutes seront transmises au nœud d'agrégation pour l'analyse.

—Les données moins sensibles au temps sont envoyées au Cloud pour l'analyse ou le stockage à long terme.

1.3.3 Les Avantages du Fog Computing:

Le Fog Computing étend le modèle de cloud computing à la périphérie du réseau. Bien que le brouillard et le cloud utilisent des ressources similaires (réseau, calcul et stockage) et partagent bon nombre des mêmes mécanismes et attributs (virtualisation, multi-tenant), le fog computing apporte de nombreux avantages pour les appareils IoT. Ces avantages peuvent être résumés comme suit [27]:

• **Une plus grande agilité commerciale :** avec l'utilisation des bons outils, les applications de calcul de brouillard peuvent être rapidement développé et déployé. De plus, ces applications peuvent programmer la machine pour qu'elle fonctionne selon les besoins du client [26].

• **Faible latence** : le brouillard a la capacité de prendre en charge les services en temps réel (par exemple, les jeux, la vidéo diffusion en continu).

• **Distribution géographique et à grande échelle :** le Fog Computing peut fournir un calcul distribué et des ressources de stockage à des applications volumineuses et largement distribuées [27].

• **Réduction des dépenses d'exploitation** : économiser de la bande passante réseau en traitant les données sélectionnées localement à la place de les envoyer dans le cloud pour analyse [27].

• **Flexibilité et hétérogénéité** : le Fog computing permet la collaboration de différents environnements et infrastructures entre plusieurs services [26].

• **Évolutivité :** la proximité du Fog Computing avec les appareils finaux permet de faire évoluer le nombre d'appareils connectés dispositifs et services [26].

1.3.4 Inconvénients fog computing:

Le Fog semble être l'infrastructure idéale pour supporter les futures applications IoT. Cependant, il reste encore un certain nombre de verrous à lever pour qu'il puisse être efficacement utilisé. Voici une liste non exhaustive des défis apportés par l'utilisation du Fog en tant que paradigme de calcul.

1. Gérer efficacement les interactions entre les noeuds à la périphérie et les centres de données (offloading).

2. La découverte et l'estimation des ressources [Amir M. Rahmani 2017].

3. La problématique du placement et du déploiement des entités de gestion et de contrôle d'un environnement Fog.

4. La gestion du compromis entre efficacité d'utilisation des ressources de calculs et celle des ressources de communication.

5. Le problème du stockage distribué des données, leur persistance et leur intégrité.

6. L'absence d'un modèle économique pour les services offerts par les infrastructures Fog Computing.

7. Les contraintes spatiales pour le partage des ressources définies par la position des utilisateurs et la portée réseau des noeuds Fog.

1.3.5 Différence entre Fog etCloud :

Le Fog computing et le Cloud computing sont interdépendants dans la fourniture de ressources de stockage et de calcul. Cependant, ce sont des paradigmes informatiques différents. Le tableau 1résume les principales différences dont nous citions :

	CLOUDCOMPUTING	FOGCOMPUTING
Gestion	Centralisée	Distribuée
Taille	Très grands centres de données	Un grand nombre de petits nœuds
Latence	Élevée	Faible
Capacité de calcul	Très élevée	Faible
Évolutivité	Moyenne	Élevée
Nature de l'échec	Prédictible	Divers
Coût de déploiement	Élevé	Faible

TABLE1.1–Comparaison entre le Cloud et Fog computing

—La proximité du Fog avec les nœuds sous-jacents.

—La gestion : Le Cloud est géré de manière centralisée, en revanche, la gestion dans le Fog est distribuée.

—La latence : En raison de la position du Fog, son délai de traitement est réduit par rapport au cloud.

—En raison de la connectivité sans fil dominante et de la gestion décentralisée, le taux d'échec dans le Fog est élevé que celui du Cloud.

Il convient de noter que le Fog ne peut pas remplacer le Cloud. Les deux technologies ont des contributions différentes pour améliorer les performances.

1.4 Conclusion:

Dans ce chapitre nous avons défini le cloud et le fog computing en détaille, et nous avons aussi expliqué la différence entre ces deux paradigmes. Ils 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 soulève un certain nombre de défis, notamment au sujet de la disponibilité des ressources.

Chapitre 2

Réplication des données dans le Cloud/Fog computing

2 Etat de lart sur réplicatin des données

2.1 Introduction:

La réplication est une technique de gestion des données très connue qui a étécouramment adoptée par de nombreux systèmes traditionnels, notamment (i) les systèmes de gestion de bases de données (SGBD) [21]. (ii) les systèmes parallèles et distribués [22], (iii) les systèmes mobiles [23] et (iv) les autres systèmes à grande échelle incluant le P2P [24] et systèmes de grille de données [25].

Dans ce chapitre nous proposons un état de l'art sur la réplication des données et la gestion de répliques dans le cloud et le fog computing.

2.2 Réplication des données:

2.2.1 Définition:

La réplication de données améliore la disponibilité, les performances (via des requêtes de diffusion de différentes répliques) et la latence perçue par l'utilisateur (en affectant des requêtes à la réplique la plus proche) au détriment des coûts de coordination et de stockage.

2.2.2 Types de réplication:

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 [6].

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 [7]. 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 [6]:

-**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.

2.2.3 La gestion de répliques dans le Cloud et le fog :

La réplication de données vise à placer stratégiquement des copies de données afin d'augmenter la disponibilité, l'accès aux performances, la fiabilité et la tolérance aux pannes, ainsi que de réduire l'utilisation de la bande passante et les délais de réalisation des taches. De nombreuses stratégies de réplication ont été proposées pour atteindre ces objectifs. Chaque stratégie de réplication de données devrait être capable de résoudre plusieurs problèmes [18]. (i) Quelles données devraient être répliquées? Il n'est généralement pas possible de répliquer chaque ensemble de données. Il est donc important d'établir des critères significatifs pour choisir ce qu'il faut répliquer. (ii) Quand la réplication devrait-elle avoir lieu? L'établissement d'un bon compromis sur le moment de la réplication est crucial, car la réplication trop précoce risque d'entraîner un gaspillage de ressources, tandis que la réplication trop tardive peut ne pas donner tous les avantages de la réplication. (iii) Combien de répliques devraient être créées? Un nombre optimal, ou au moins un nombre quasi-optimal de répliques devrait idéalement être présent dans le système pour équilibrer les avantages et les coûts de la réplication. Tandis que trop de répliques peuvent être gaspilleuses, avoir trop peu peut être tout aussi indésirable car il peut ne pas être suffisant pour satisfaire une qualité de service souhaitée. (iv) Où les répliques doivent-elles être placées? Généralement, le fait de rapprocher les répliques des clients ayant le plus de demandes d'accès peut améliorer les performances globales lors des accès fréquents aux données. L'obtention d'une configuration de réplique optimale dans un système de gestion de données est un problème NP-difficile [19;20].

2.2.4 Avantages de réplication:

La technique de réplication procure un certain nombre d'avantages qui améliorent les performances. L'amélioration de performances grâce à la réplication est en terme de :

— Temps de réponse et accès rapide et écace aux données ;

— Une meilleure disponibilité de données et une meilleure localité des données ;

— Charge acceptable par le système car la réplication permet de répartir la charge entre les nœuds ;

— Permettre un parallélisme dans la consultation de la même donnée ;

— Une importante capacité de calcul parallèle ;

— Tolérer les pannes : la réplication permet les accès aux données même en cas de défaillance d'un support puisque la donnée se trouve sur plusieurs endroits [38].

2.2.5 Les inconvénients de la réplication :

La réplication soulève des problèmes que nous résumons comme suit :

— Choix de l'entité 'a répliquer : quoi répliquer ?

— Degré de la réplication : combien répliquer ?

— Moment de réplication : quand répliquer ? Statique ou dynamique ?

— Placement des répliques : ou' placer les répliques ?

— Localisation des meilleures répliques ;

— Gestion de la cohérence des répliques : il est nécessaire de dènir un protocole de cohérence qui puisse garantir que toutes les répliques d'une même donnée soient cohérentes ;

— Partitionnement du système : lorsque un lien entre deux nœuds tombe en panne, lesrépliques de la même donnée dans les deux systèmes doivent converger vers le mêmeétat ;

— Choix de la meilleure réplique d'un point de vue cohérence pour le traitement d'unerequête [37].

2.3 Travaux connexes:

Plusieurs types de recherche ont été consacrés au domaine de la réplication. Nous trouvons:

Fei Xie et al [28] définissez trois paramètres de seuil pour les conditions des ensembles de données parmi les ensembles de données, accédez aux fréquences des ensembles de données et à la capacité de stockage des centres d'information. La dépendance des ensembles de données entre les ensembles de données et la récurrence pour chaque ensemble de données sont calculées en tant que limites de l'ensemble de données. Ils utilisent l'estime limite de l'espace de capacité pour restreindre la réplication des informations afin de maintenir une distance stratégique par rapport aux problèmes d'inondation et de garantir l'achèvement complet des courses dans la zone correspondante. Ils classent en outre les types d'informations en trois catégories, les ensembles de données établis, les ensembles de données flexibles libres et les ensembles de données flexibles contraints, afin de développer une cartographie entre les ensembles de données et chaque centre d'information. En recevant leur méthodologie, ils s'efforcent d'aider à réduire les coûts de développement et d'échange d'informations.

Tadeusz et al. [29] proposent une stratégie de reproduction des informations NoSQL. Le calcul s'appelle Lorq. Les principaux points forts de Lorq sont (a) la réplication des informations est réalisée par des moyens de reproduction Desjournaux mettant de côté les opérations de mise à niveau, et ce qu'on appelle l'opération d'impulsion envoyée par le pionnier ; (b) les méthodologies de préparation et de réplication garantissent que toutes les opérations de chaque réplique sont inévitablement exécutées dans le même ordre et qu'aucune opération n'est déplacée. Une considération peu commune est accordée aux différents types de cohérence, qui peuvent être assurés par le cadre. Ils proposent une stratégie basée sur les données stockées par les administrations clientes pour assurer divers niveaux de cohérence, actualisant ainsi l'utilité des SLA. Mais la spécification, la vérification et l'optimalité des types de données répliquées [30] sont négligées.

Xiuguo et al. [31] sont allés vers la réalisation de la référence de dispersion des copies les moins prises d'une manière terre à terre, ils proposent une procédure d'arrangement des reproductions, en comptant la manière de distinguer le besoin de faire la reproduction, et prévoient un calcul pour les situations de copie qui peuvent efficacement réduire l'ensemble des prélèvements dans le cloud et proposer des ensembles d'informations récupérées par l'administration, y compris la capacité récupérée et les échanges prélevés ; montrant un nouvel ensemble d'informations mondiales copie la méthodologie des arrangements du rapport coût-efficacité, voir nommé MCRP, qui est un arrangement à coût minimum inexact. Ils ont proposé une méthodologie de réplication d'informations rentable avec une réflexion sur la récurrence et le temps de réponse moyen pour décider si l'ensemble de données doit être imité ou non dans un environnement cloud.

Sathiya et al. [32] examinent les changements sur une convention de cohérence appelée LibRe, qui agit comme une technique de cohérence intermédiaire entre la cohérence inévitable par défaut et les choix de cohérence solides déterminés à partir de la propriété de point de croisement. La convention LibRe initiale utilisait un registre, qui enregistre la liste des nœuds de réplique contenant la forme la plus récente des éléments d'information. Par conséquent, faire allusion au registre au cours du temps examiné fait une différence pour transmettre les demandes étudiées à un centre de reproduction détenant la forme la plus récente de l'élément d'information requis. De l'autre côté, le protocole rencontrerait de brèves incohérences.

Tos et al. [33] Propose Execution and Benefit Arranged Information Replication Methodology (PEPR) qui garantit que le SLA assure, par ex. l'accessibilité et l'exécution, à l'occupant tout en maximisant l'avantage financier du fournisseur de cloud. Pour le degré d'exécution, ils définissent le temps de réaction comme une partie fondamentale du SLA. Dans le PEPR, lors de l'évaluation d'une demande, dans le cas où une estimation du temps de réaction évaluée est plus importante que la limite de

temps de réaction SLO, cela implique qu'une préparation de réplication peut être activée. À ce moment-là, l'avantage économique, c'est-à-dire la rentabilité, le fournisseur de cloud s'affiche également. Le choix de la réplication est en quelque sorte fait lorsque le temps de réaction et l'avantage financier du fournisseur sont satisfaits. Le nombre de copies est puissamment équilibré en tenant compte du fait que les objectifs SLA sont remplis au fil du temps. De plus, le plus petit nombre de copies est conservé en permanence pour garantir la moindre disponibilité. L'estimation du temps de réponse pour les demandes des habitants est calculée lorsque les requêtes arrivent dans le cloud. Si l'estimation montre qu'une exécution intéressante ne peut pas être effectuée, la réplication des informations est effectuée, mais pour ainsi dire lorsqu'elle est financièrement réalisable pour le fournisseur.

Yaser et al. [34] penser est propulsé par ces réflexions pionnières car aucune d'entre elles ne peut en même temps répondre autour des agencements et des temps de relocalisation des objets. Pour répondre à ces questions, ils prennent des engagements clés : pour commencer, en abusant de la programmation énergétique, ils définissent la prise hors ligne d'un problème d'optimisation des péages dans lequel la récupération idéale de capacité, Get, Put et de mouvement est calculée là où le futur précis La charge de travail est supposée connue a priori. Pour le moment, ils proposent deux calculs en ligne pour découvrir des prises de péage quasi-optimales.

2.4 Conclusion:

Dans ce chapitre nous avons fait une brève présentation de la réplication des données et la gestion de répliques dans les environnements cloud et fog computing et quelque travaux connexes .Dans le chapitre suivant nous présentrons notre stratégie proposé.

Chapitre 3

Approche Proposée

3 Approche propose

3.1 Introduction:

Dans les chapitres précédents, nous avons présenté la réplication des données dans le fog computing pour l'internet des objets 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 termes de contexte. Notre objectif principal est de proposer et d'implémenter une nouvelle architecture de réplication des données dans un environnement de Fog computing, et cela dans le but d'améliorer les performances telles que l'utilisation du temps de réponse et de réduire la consommation de la bande passante. Notre solution est basée sur une combinaison d'une stratégie d'ordonnancement et de réplication de données dans les environnements de Fog/Cloud Computing et de de réplication dynamique de données dans les Fog Computing.

Dans ce chapitre, nous allons présenter notre stratégie proposée pour gérer la réplication de données dans l'environnement du Fog Computing, nous allons commencer par la déscription de l'architecture utilisé, ensuite nous expliquons notre stratégie en décrivant ses différentes phases et les algorithmes nécessaires pour son fonctionnement.

3.2 Architecture hiérarchique:

Dans notre strategie, nous avons utilisé une architecture à trois niveaux qui est l'une des structures hiérarchiques de base la plus largement utilisées dans le domaine du fog computing, comme le montre la figure3.1.

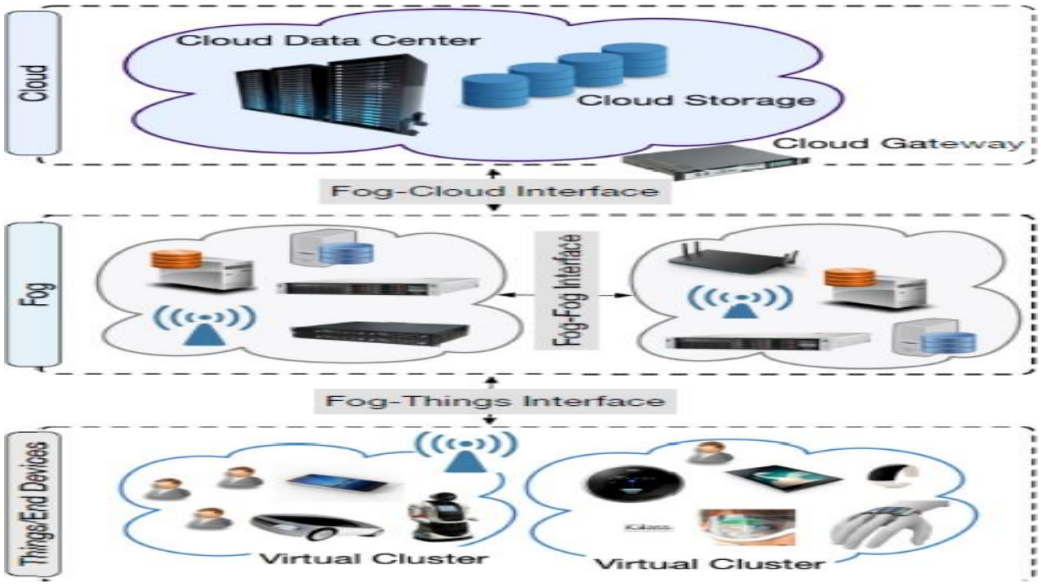

Figure 2.1:L'architecture à trois niveaux dans le Fog computing

—Le premier niveau comprend les dispositifs terminaux, les dispositifs compatibles d'Internet des Objets (IOT), y compris les nœuds capteurs, les dispositifs intelligents, etc. Ces dispositifs terminaux peuvent également être appelés "nœuds terminaux" (TN).

—Le deuxième niveau représente la couche de Fog computing. Il est composé des rou- teurs, des passerelles et de commutateurs... etc, qui peuvent partager des ressources de stockage et de calcule.

—Le troisième niveau est le Cloud, qui se compose des centres de données et fournit des ressources de calcul et de stockage suffisantes.

Le principal avantage de l'architecture à trois niveaux est une meilleure gestion du service. Les services au temps sont traités par des nœuds Fog les plus proches de l'utilisateur.

3.3 Stratégie de replication de données:

La stratégie proposée comprend quatre phases importantes: la première phase concerne l'ordonnancement des tâches et le placement des données, la deuxième phase permet le traitement des requêtes des utlisateurs, la troisième phase gère le processus de réplication des données et la dernière phase permet de supprimer les données inutiles.

Dans la section suivante, nous présentons notre stratégie et nous allons détailler les différentes phases de notre solution.

3.3.1 La phase d'ordonnancement des tâches et placement des données:

Cette phase est composée de quatre étapes illustrées dans la figure 3.2 suivante qui représente un organigramme détaillé pour le placement des données et l'ordonnancement des tâches :

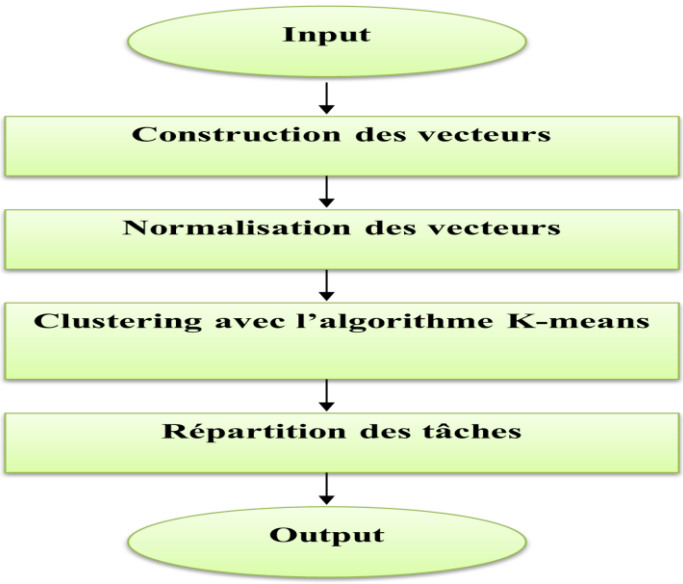

Figure3.2: organigramme d'ordonnancement des tâches et placement des données

Nous détaillons ces étapes comme suit :

3.3.1.1 ETAPE 1: Construction des vecteurs

Pour décrire les différentes étape de la phase d'ordonnacement des tâches et placement des données, nous allons utilliser les notations suivantes:

- Ensemble de données initial D = [d_1, d2, d3 ...] désigne tous les ensembles des Données d'entrée requis avant l'exécution de l'application de workflow.
- Ensemble des tuples (tâches) T = [t_1, t2, t3 ...] désigne toutes les tuples de l'application de workflow.
- Dépendance des données et modèle de taille des données. Selon la relation entre l'ensemble de données et la tâche dans le DAG [directed acyclic graph], un vecteur multidimensionnel est établi pour chaque donnée d_i= (f_{it0}, fit1, fit2...fitn).

Où d_i désigne le i-ème ensemble de données d'entrée dans le DAG, N est le nombre total de tâches. f_{itn} indique la taille d_i est utilisé par le tache t_n, si d_i est utiliser par t_n, la valeur de f_{itn} c'est la taille de d_i, sinon, sa valeur est zéro. Il peut être exprimé par la formule suivante :

$$f_{itn} = |d_i \in t_n \text{ (input) } | * \text{size } (d_i)$$

t_n (input) C'est l'ensemble des ensembles de données d'entrée requis par t_n. Dans le modèle vectoriel multidimensionnel, la dépendance des données est reflétée par le nombre de valeurs non nulles dans le vecteur, Si plusieurs données sont traitées par la même tâche, les attributs correspondants aux valeurs de ses vecteurs multidimensionnels égaux à la taille de l'ensemble de données. Parexemple:

d_0=1GB d_1=1GB d_2=3GB d_3=1GB d_4=5GB d_5=2GB d_6=1GB

d_7=3GB

d_0= (1, 0, 0, 0, 0, 0)
d_1= (1, 1, 0, 0, 0, 0)
d_2= (0, 3, 3, 3, 0, 0)
d_3= (1, 1, 0, 0, 0, 0)
d_4= (0, 0, 5, 5, 0, 0)
d_5= (0, 0, 0, 2, 2, 0)
d_6= (0, 0, 0, 1, 1, 0)
d_7= (0, 0, 0, 0, 3, 3)

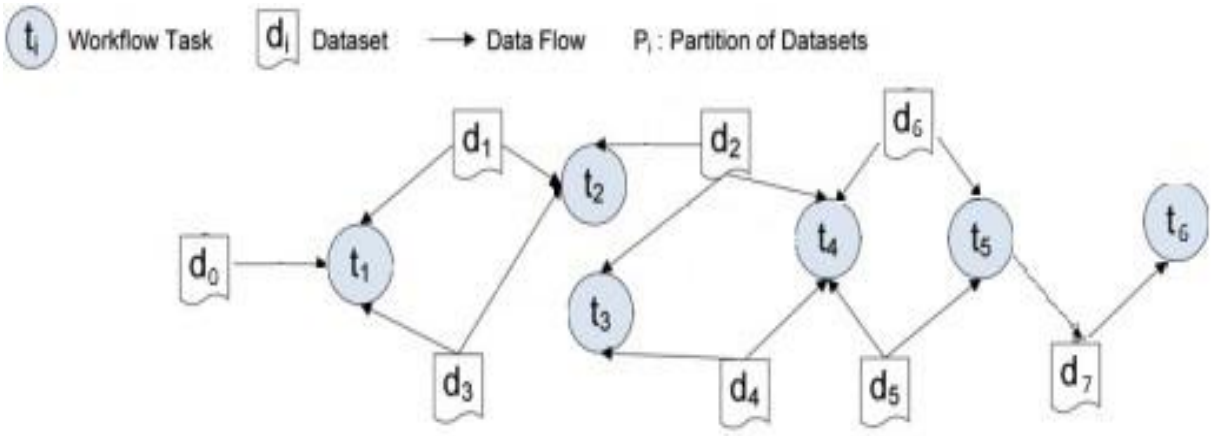

Figure3.3: exemple de placement de données.

3.3.1.2 ETAPE2:la normalisation des vecteurs

La normalisation est un processus utilisé pour normaliser tous les attributs de l'ensemble de données et leur donner un poids égal pour améliorer laprécision du résultat. Il existedifférentes méthodes de normalisation des données telles que Min-Max, Z-Score et Decimal Scaling. La meilleure méthode de normalisation dépend des données ànormaliser. Ici, nous avons utilisé la technique de normalisation Min-Max dans notre algorithme car notre ensemble de données est limité et n'a pas beaucoup de variabilité entre le minimum et le maximum. La technique de normalisation Min-Max effectue une transformation linéaire sur les données. Dans cette méthode le vecteur multidimensionnel normalisé [39]:

$$d'_i = (f'_{it0}, f'_{it1}, f'_{it2}..f'_{itn})$$

Avec :

$$f'_{itn} = \frac{fitn - min(fitn)}{max(fitn) - min(fitn)}$$

Ça donne les résultats suivants:

d'_0= (1, 0, 0, 0, 0,0)

d'_1= (1, 1, 0, 0, 0,0)

d'_2= (0, 1, 1, 1, 0,0)

d'_3 = (0, 1, 1, 1, 0,0)

d'_4= (0, 0, 1, 1, 0, 0)

d'_5= (0, 0, 0, 1, 1, 0)

d'_6= (0, 0, 0, 1, 1, 0)

d'_7= (0, 0, 0, 0, 1, 1)

3.3.1.3 ETAPE 3: SERVICE DE CLUSTERING AVEC K-MEANS

L'algorithme k-means mis au point par McQueen en 1967, un des plus simples algorithmes d'apprentissage non supervisé, appelée algorithme des centres mobiles, il attribue chaque point dans un cluster dont le centre (centroïde) est le plus proche. Le centre est la moyenne de tous les points dans le cluster, ses coordonnées sont la moyenne arithmétique pour chaque dimension séparément de tous les Points dans le cluster c'est à dire chaque cluster est représentée par son centre de gravité. Retournons à notre proposition étant donné que la relation entre les ensembles dedonnées est maintenant reflétée dans le vecteur, la question de trouver la dépendance entre ces ensembles de données est transformée en regroupement vectoriel.

L'algorithme K-means est adopté pour regrouper des ensembles de données hautement dépendants. Ici, nous utilisons la similarité vectorielle pour mesurer le degréde dépendance des données, qui est représenté par la distance entre deux vecteurs d_i et d_j. Plus la distance des deux vecteurs est petite, plus ils sont similaires. La formule de distance euclidienne est utilisée dans notre modèle où

$$d(d'_i, d'_j) = \sqrt{\sum_{k=1}^{n} (f'itk - f'jtk)2}$$

Ensuite, la fonction de critère d'erreur au carré est utilisée pour déterminer si le regroupement des ensembles de données est stable [39].

$$E = \sum_{i=0}^{k} \sum_{p \in Di} |p - mi|2$$

Où m_i représente le centre géométrique du cluster i, et D_i désigne une liste de sous-ensembles de données comprenant des ensembles de données à forte dépendance. Les resultats aprés le regroupment

P1 (d0, d1, d3)

P2 (d2, d4)

P3 (d5, d6)

3.3.1.4 ETAPE 4: Répartition des tâches:

Pour améliorer la phase d'ordonnancement des tâches et placement des données, nous avons utilisé un algorithme de répartition des tâches qui est détaillé dans l'organigramme suivant:

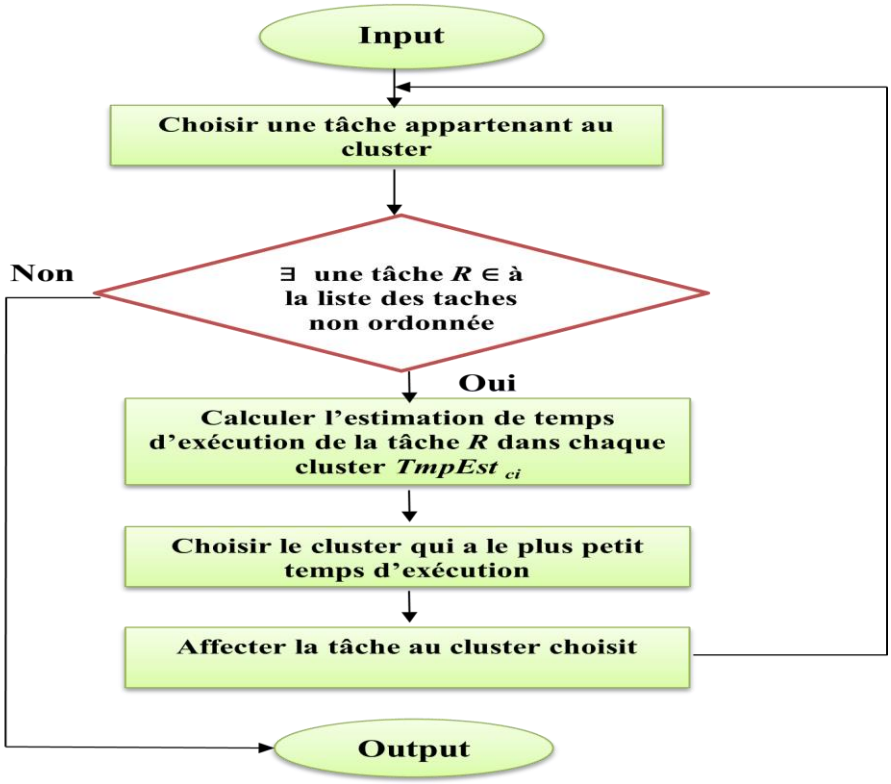

Figure3.4: organigramme de répartition des tâches

Après la construction des clusters en utlisant les étapes précedantes, nous allons améliorer la répartition des tâches selon la charge réel des Fogs.Un Cluster correspond au Fog qui va exécuter les tâches des différents utilisateurs et héberger les données nécessaires à l'exécution des différentes tâches.Tout d'abord, nous Allons choisir une tâche *R* non ordonnée appartenant au Fog. Ensuite, nous allons calculer l'estimation du temps d'exécution $TmpEst_{ci}$ de la tâche *R* dans les Fog voisins.Nous allons choisir le Fog qui a le plus petit temps d'exécution, enfin on affecte cette tâche au Fog choisit.On effectue cette operation jusqu'à on se termine avec tous les tâches existantes.Pour calculer l'estimation du temps d'exécution d'une tâche R, nous allons utiliser l'équation suivante :

$$TmpEst_{ci} = Temp\ d'attente + Temp\ d'accès\ aux\ donnés\ locaux + Temp\ d'accès\ aux\ données\ distante$$

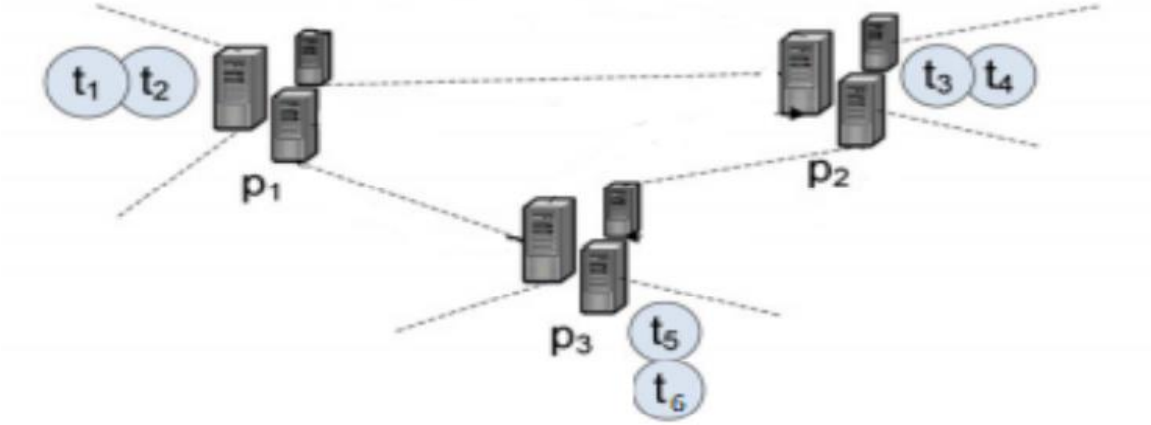

Figure 3.5: Exemple après le clustering et répartition des tâches

3.3.2 La phase de traitement d'une requête :

Dans cette phase nous présentons le principe du traitement d'une requête à partir de l'arrivée de la requête *R* jusqu'au l'exécution de cette requête et l'accès aux données necessaire.L'organigramme suivant détaille le principe de traitement :

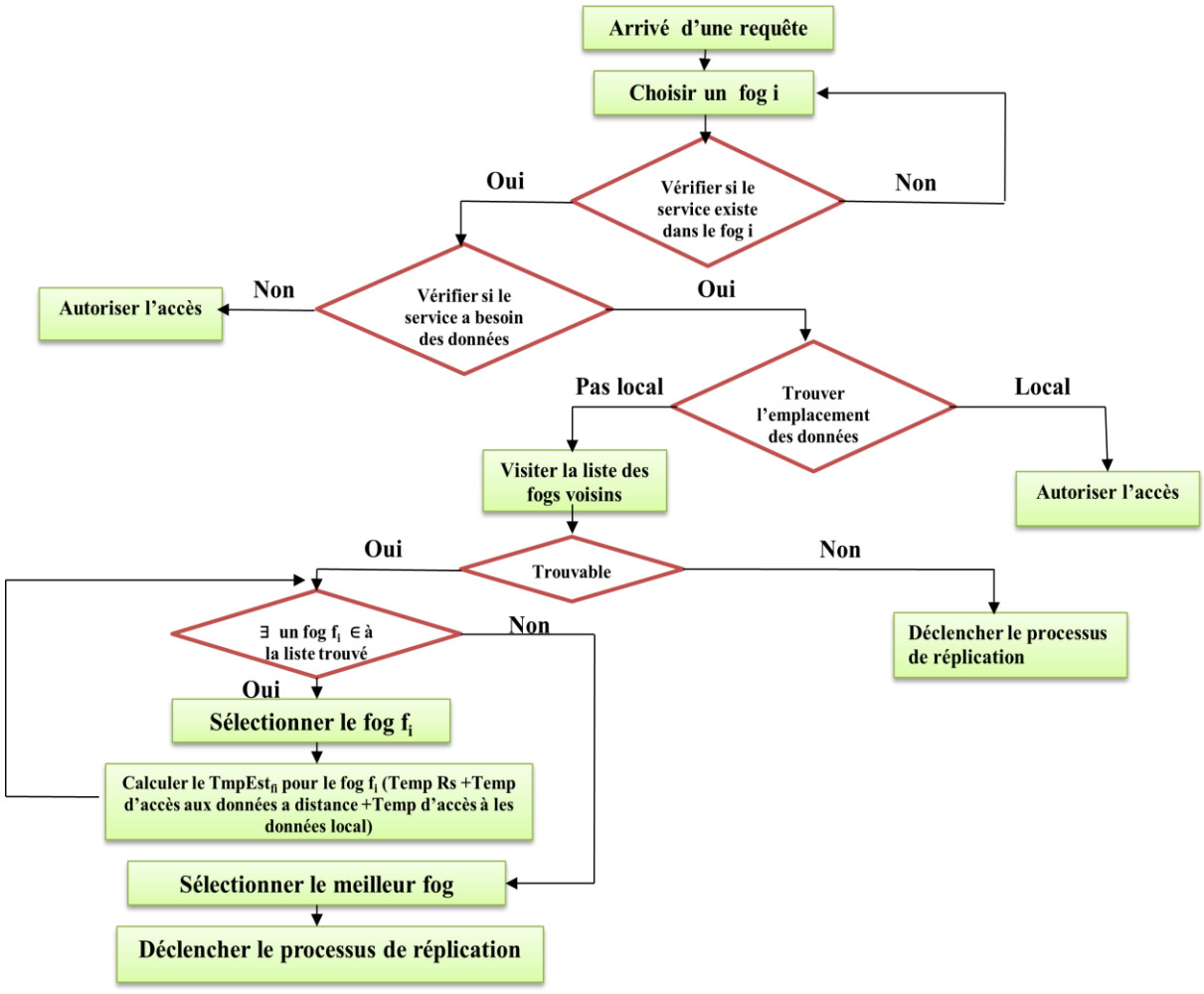

Figure 3.6: organigramme de traitement d'une requêt

Après l'arrivée des requêtes des clients, la phase d'ordonnacneent et de placement va être executer afin d'affecter les tâches au meilleur fog, Après le choix du Fog, le gestionnaire de traitement va vérifier si le service existe dans ce fog, s'il ne se trouve pas dans le Fog i le gestionnaire va choisir un autre fog i. Dans le cas où le service se trouve dans le Fog i, le gestionnaire vérifie si les données sont nécessaires pour l'exécution de ce service. Si aucune donnée n'est requise, le gestionnaire autorise l'accès pour traiter la requête du client. Si le service à besoin des données, le gestionnaire va lancer le processus de recherche de ces données. Si les données se trouvent localement le gestionnaire autorise l'accès pour traiter la requête du client. Si les données ne se trouvent pas localement, le gestionnaire va rechercher dans la liste des Fogs voisins.

Si les données ne sont pas trouvables dans la liste des Fogs voisin, le gestionnaire va déclencher le processus de réplication (créer une réplique à partir du Cloud).Dans le cas où les données sont trouvables dans les fog voisin, nous allons calculer une estimation du temps d'execution de cette requête dans tous les fogs voisin. Nous avons utilisé la formule suivante pour calculer le temps d'estimation d'une requête R qui a besoin d'un esemble de données:

TmpEst$_{fi}$ = (Temp Rs +Temp d'accès aux données à distance +Temp d'accès aux données locaux)

Temps Rs c'est le temps de migration du service, si le service est présent dans le fog ce temps est null ; sinon ce service est migré à partir du cloud, il est calculé par la formule suivante :

$$TempsRs = \frac{taille\ du\ Service}{bande\ passante} + Latence(Fog, Cloud)$$

Le gestionnaire selectionne le meilleur fog qui a le plus petit temps d'estimation pour traiter la requête du client.

3.3.3 La phase de réplication :

Cette phase permet la réplication des données les plus fréquemment dans chaque Fog par le gestionnaire de réplication. Le but de la réplication est de minimiser le transfert de données, ce qui permet de réduire le temps de réponse ainsi que l'utilisation de la bande passante.

Le fonctionnement de processus de réplication est décrit dans l'organigramme suivant :

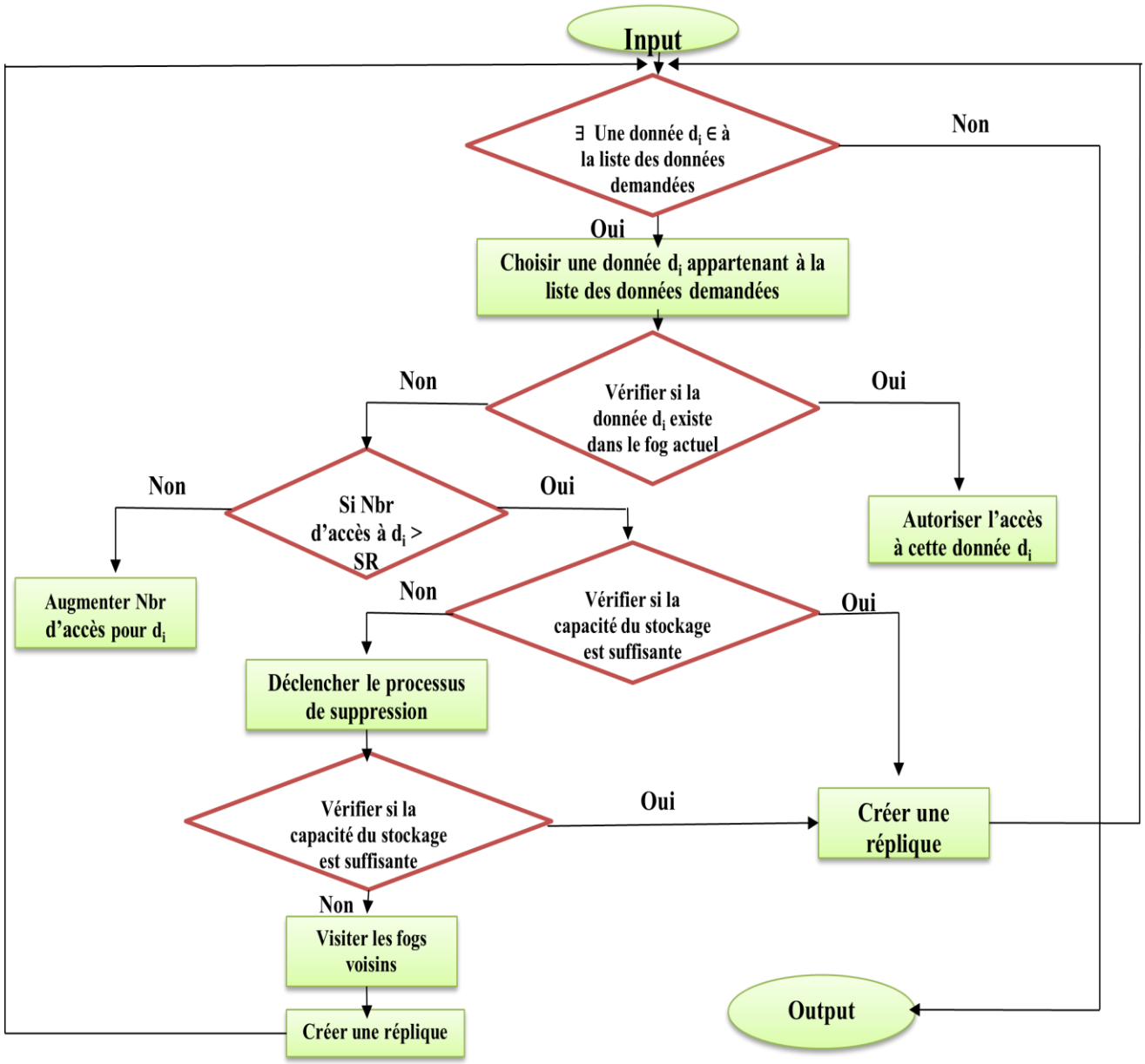

Figure 3.7: organigramme de réplication

Chaque fog contient une liste des données demandées par les différentes requêtes. L'algorithme de réplication va parcourir cette liste de données, si la donnée d_i existe dans le fog actuel, le gestionnaire de réplication va autoriser l'accès pour cette donnée d_i. Si la donnée d_i n'existe pas localement, le gestionnaire de réplication doit vérifier le nombre d'accès à cette donnée d_i, si le nombre d'accès dépasse le seuil de Réplication(SR) un réplique de cette donnée va être crée dans le fog actuel. Le gestionnaire de réplication va vérifier si la capacité du stockage est suffisante pour crée une réplique de la donnée d_i. S'il y a suffisamment de stockage, la réplique sera créer et le processus se poursuit jusqu'à ce que le gestionnaire sélectionne toutes les données demandées.Si la capacité de stockage est insuffisante pour créer la nouvelle réplique, le processus de suppression est déclenché.Si la capacité de stockage reste

insuffisante, le gestionnaire de réplication va créer la réplique dans un fogs voisins. Le fog sélectionné a une capacité de stockage suffisante, une petite charge et une grande bande passante.

3.3.4 La phase de suppression :

Le gestionnaire de suppression est lancé par le gestionnaire de réplication dans le cas où la capacité du stockage n'est pas suffisante pour optimiser l'utilisation de l'espace disque des nœuds.

La figure suivante illustre l'organigramme qui détaille le principe de fonctionnement de la suppression :

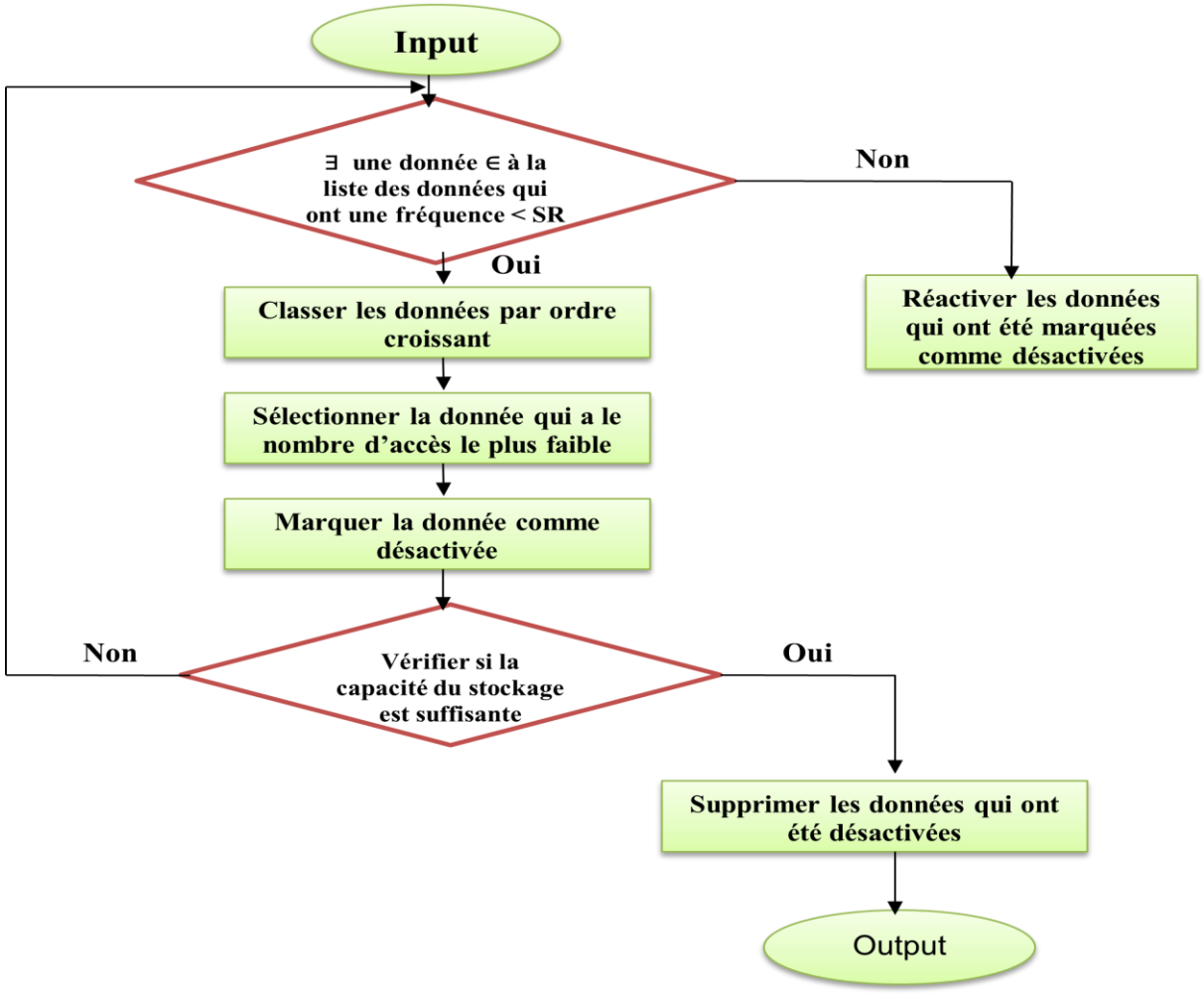

Figure 3.8: organigramme de suppression

Tout d'abord, le gestionnaire de suppression classe par ordre croissant les données qui se trouvent dans le fog concerné et qui ont une fréquence d'accès inférieure au SR. le gestionnaire selectionne la donnée qui a le nombre d'accès le plus faible, en suite va marquer cette donnée comme désactivée (pas encore supprimé).Après il va vérifier si la capacité du stockage est suffisante, si c'est le cas, la donnée qui a été marqué comme désactivé va être supprimé.Dans le cas ou la capacité n'est pas suffisante, le

gestionnaire va sélectionner une autre donnée qui a le deuxième nombre d'accès le plus faible parmis la liste précédente, cette donnée va être marquer comme désactivée. Le gestionnaire repète ce procesus jusqu'à ce que la capacité de stockage sera suffisante pour créer la nouvelle réplique. Si le gestionnaire désactive toute les données de la liste qui contient les données qui ont la fréquence d'accès inférieur au SR et la capacité de stockage reste insuffisante, le gestionnaire réactive tous les données de cette liste et le processus de suppression sera annulé. Le gestionnaire de suppression informe le gestionnaire de réplication de cet échec. Dans ce cas, le gestionnaire de réplication choisi un autre fog voisin pour placer la nouvelle réplique.

3.4 Conclusion :

Au cours de ce chapitre, nous avons décrit le fonctionnement de notre stratégie dans un environnement du Fog comuting. Cette stratégie proposée gère le problème de la réplication des données Nous avons utilisé l'ordonnancement des données et le traitement des requêtes qui permet d'affecter les requêtes vers les Fogs les plus performants .La stratégie de réplication proposée permet de minimiser le temp réponse et réduire la bande passante.

Dans le chapitre suivant, nous allons implémenter notre stratégie et présenter les résultats obtenus par la simulation.

Chapitre 4

SIMULATION

4 SIMULATION

4.1 Introduction :

Dans le chapitre précèdent, nous avons présenté notre stratégie de réplication des données dans le Fog computing.Dans ce chapitre, nous allons présenter le langage Java, l'environnement de développement Eclipse et Le simulateur IfogSim et sesclassesfondamentalesainsiquelesclasses que nous avons intégrées.

Par la suite, nous citerons les différentes étapes et les configurations nécessaires avant le lancement de la simulation de nos approches. Enfin nous discuterons les résultats obtenus.

4.2 Environnement de Développement :

Pour implémenter et tester notre stratégie de réplication des données dans l'environnement du Fog 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) i3-3217U CPU @ 1.80GHz 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 signifie café en Slang (argot américain). Le langage 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. Le but était d'avoir un langage de développement simple et portable sur plusieurs systèmes d'exploitation tels que UNIX, Windows, Mac OS ou GNU/Linux, avec ou sans modifications, afin de programmer tous les processeurs (ordinateurs ou appareils électroménagers, …) tout en veillant à la robustesse, la compatibilité, la petite taille du runtime ou des codes générés et aussi facilité de programmation. Java reprend la syntaxe de C++ tout en le simplifiant et offre aussi un ensemble de classes pour développer des applications de types très variés (réseau, interface graphique, multi-

tâches, etc.). Java comprend bien d'autres aspects (programmation graphique, applets, programmation réseau, multi-tâches)

Le langage Java a connu plusieurs évolutions depuis le JDK (Java Development Kit) 1.0 A partir de la version Java 1.2 on parle de Java 2 avec 2 grandes inovations : l'API graphique Swing est intégrée ; et la machine virtuelle Java de Sun inclut un compilateur "Juste à temps" (Just in Time) [41].

4.4 Environnement de Développement :

4.4.1 Eclipse :

Eclipse IDE est un environnement de développement intégré libre (le terme Eclipse désigne également le projet correspondant, lancé par IBM) extensible, universel et polyvalent, permettant potentiellement de créer des projets de développement mettant en œuvre n'importe quel langage de programmation. Eclipse IDE est principalement écrit en Java (à l'aide de la bibliothèque graphique SWT, d'IBM), et ce langage, grâce à des bibliothèques spécifiques, est également utilisé pour écrire des extensions. La spécificité d'Eclipse IDE vient du fait de son architecture totalement développée autour de la notion de plug-in (en conformité avec la norme OSGi) : toutes les fonctionnalités de cet atelier logiciel sont développées en tant que plug-in [41].

4.4.2 IFogSim :

La boîte à outils de simulation iFogSim est développée sur le cadre fondamental de CloudSim. CloudSim est l'un des simulateurs largement adoptés pour modéliser les environnements de cloud computing. 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 d'appareils IoT (par exemple, capteurs, actionneurs). Cependant, dans iFogSim, les classes sont annotées de manière à ce que les utilisateurs, n'ayant aucune connaissance préalable de CloudSim, puissent facilement définir l'infrastructure, le placement des services et les politiques d'allocation des ressources pour le Fog computing. IFogSim s'applique Sense-Process-Actuate et modèle de flux de données distribué tout en simulant n'importe quel scénario d'application dans l'environnement informatique Fog. Il facilite l'évaluation de la latence de bout en bout, de la congestion du réseau, de la consommation d'énergie, des dépenses opérationnelles et de la qualité de service. Dans un nombre important de travaux de recherche, iFogSim a déjà été utilisé pour simuler les ressources, la mobilité, la latence, la qualité d'expérience (QoE), l'énergie, la sécurité et la QoS- conscient de la gestion de l'environnement fog computing.

Les principales classes d'iFogSim sont représentées dans les figures 4.1 et 4.2. Dans cette section, nous présentons les détails de ces classes et leurs interactions. L'implémentation d'iFogSim est constituée d'entités et de services simulés. Tout d'abord, nous décrivons comment les éléments d'architecture sont modélisés sous forme de classes iFogsim [40].

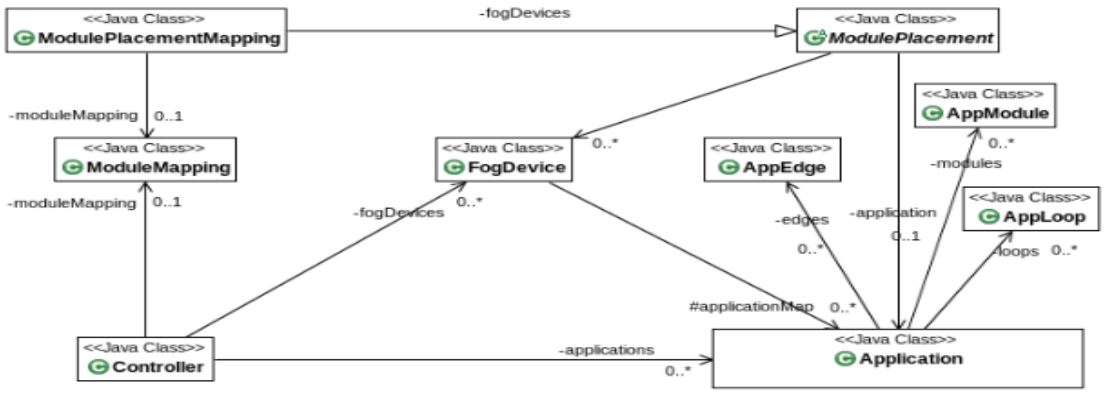

Figure4.1: Principales classesd'iFogSim

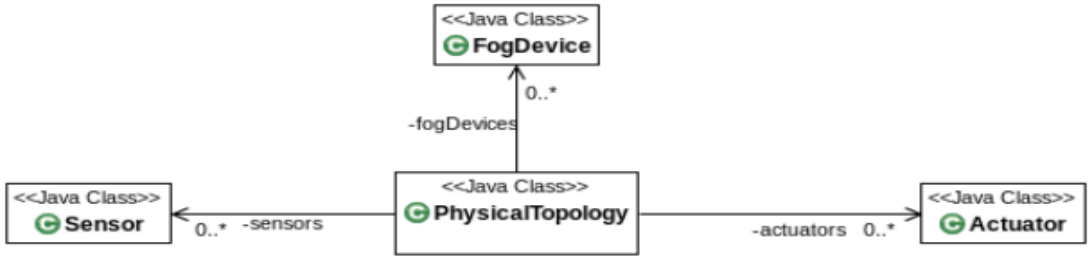

Figure 3.2: Classes de topologie physique d'iFogSim

- **FogDevice :** cette classe spécifie les caractéristiques matérielles des appareils Fog et leurs connexions à d'autres appareils, capteurs et actionneurs Fog. Réalisés par extension de la classe PowerDatacenter dans CloudSim, les attributs majeurs de la classe FogDevice sont la mémoire accessible, le processeur, la taille de stockage, les bandes passantes montantes et descendantes (définissant la capacité de communication des appareils Fog). Les méthodes de cette classe définissent la manière dont les ressources d'un périphérique Fog sont planifiées entre les modules d'application exécutés sur celui-ci et la manière dont les modules sont déployés et désactivés sur ceux-ci. Le remplacement de ces méthodes permet aux développeurs de plug-in des politiques personnalisées pour les fonctions mentionnées ci-dessus.
- **Sensor :** les instances de la classe Sensor sont des entités qui agissent comme des capteurs IoT décrits dans l'architecture. La classe contient des attributs représentant les caractéristiques d'un capteur, allant de sa connectivité aux attributs de sortie. La classe contient un attribut de référence à l'appareil passerelle Fog auquel le capteur est connecté et la latence de connexion entre eux. Plus important encore, il définit les caractéristiques de sortie du capteur et la distribution du temps inter-transmission ou inter-arrivée de tuple - qui identifie le taux d'arrivée de tuple à la passerelle.
- **Tuple :** Les tuples forment l'unité fondamentale de communication entre les entités dans le brouillard. Les tuples sont représentés comme des instances de la classe Tuple dans iFogSim, qui est héritée de la classe Cloudlet de CloudSim. Un tuple est caractérisé par son type et les modules d'application source et destination. Les attributs de la classe spécifient les exigences de traitement (définies en millions d'instructions (MI)) et la longueur des données encapsulées dans le tuple.
- **Actionneur :** cette classe modélise un actionneur en définissant ses propriétés de connexion réseau. Un attribut de la classe fait référence à la passerelle à laquelle l'actionneur est connecté et à la latence de cette connexion. La classe

définit une méthode pour effectuer une action à l'arrivée d'un tuple d'un module d'application.

- **Application :** une application est modélisée sous la forme d'un graphe orienté, les sommets du DAG représentant les modules qui effectuent le traitement des données entrantes et les arêtes indiquant les dépendances de données entre les modules. Ces entités sont réalisées à l'aide des classes suivantes :
    - AppModule : les instances de la classe AppModule représentent les éléments de traitement des applications Fog. AppModule est implémenté en étendant la classe PowerVm dans CloudSim. Pour chaque tuple entrant, une instance AppModule le traite et génère des tuples de sortie qui sont envoyés aux modules suivants dans le DAG. Le nombre de tuples de sortie par tuple d'entrée est décidé à l'aide d'un modèle de sélectivité - qui peut être basé sur une sélectivité fractionnaire ou un modèle en rafales.
    - AppEdge : une instance AppEdge indique la dépendance des données entre une paire de modules d'application et représente un bord dirigé dans le modèle d'application. Chaque arête est caractérisée par le type de tuple qu'elle transporte, qui est capturé par l'attribut tupleType de la classe AppEdge ainsi que les exigences de traitement et la longueur des données encapsulées dans ces tuples. iFogSim prend en charge deux types de bords d'application - périodiques et basés sur des événements. Les tuples sur un AppEdge périodique sont émis à intervalles réguliers. Un tuple sur un bord basé sur des événements e=(u,v)est envoyé lorsque le module sourcetu reçoit un tuple et le modèle de sélectivité den tu permet l'émission de tuples portés par e.
    - AppLoop : AppLoop est une classe supplémentaire, utilisée pour spécifier les boucles de contrôle de processus qui intéressent l'utilisateur. Dans iFogSim, le développeur peut spécifier les boucles de contrôle pour mesurer la latence de bout en bout. Une instance AppLoop est fondamentalement une liste de modules commençant à l'origine de la boucle jusqu'au module où la boucle se termine [40].

4.5 Implémentation :

Dans cette partie, nous avons intéressées à la démonstration de notre application à travers un exemple en faisant référence à quelques interfaces graphiques.

4.5.1 Interface Principale :

Nous avons créé une interface qui facilite l’accès au simulateur. L’interface doit faire appel à iFogSim ainsi qu’aux différentes approches qui se trouvent dans différents packages.

4.5.2 Configuration des paramétres de simulation:

Les figures 4.3, 4.4, 4.5 montrent les fenêtres de configuration des paramètres de simulation faite par l'utilisateur, elle contient 3 partiesprincipales:

4.5.2.1 Fog:

Pour configurer un fog nous avons besoin de définir les caractéristiques de ces fogstel que identificateur de fog (ID), la vitesse d'exécutions(MIPS) et la bande passante (BW) et l'existence de service oui ou non, aussi les fogs voisins (fog Neig) et les données qui se trouvent dans le fog (Figure 4.3).

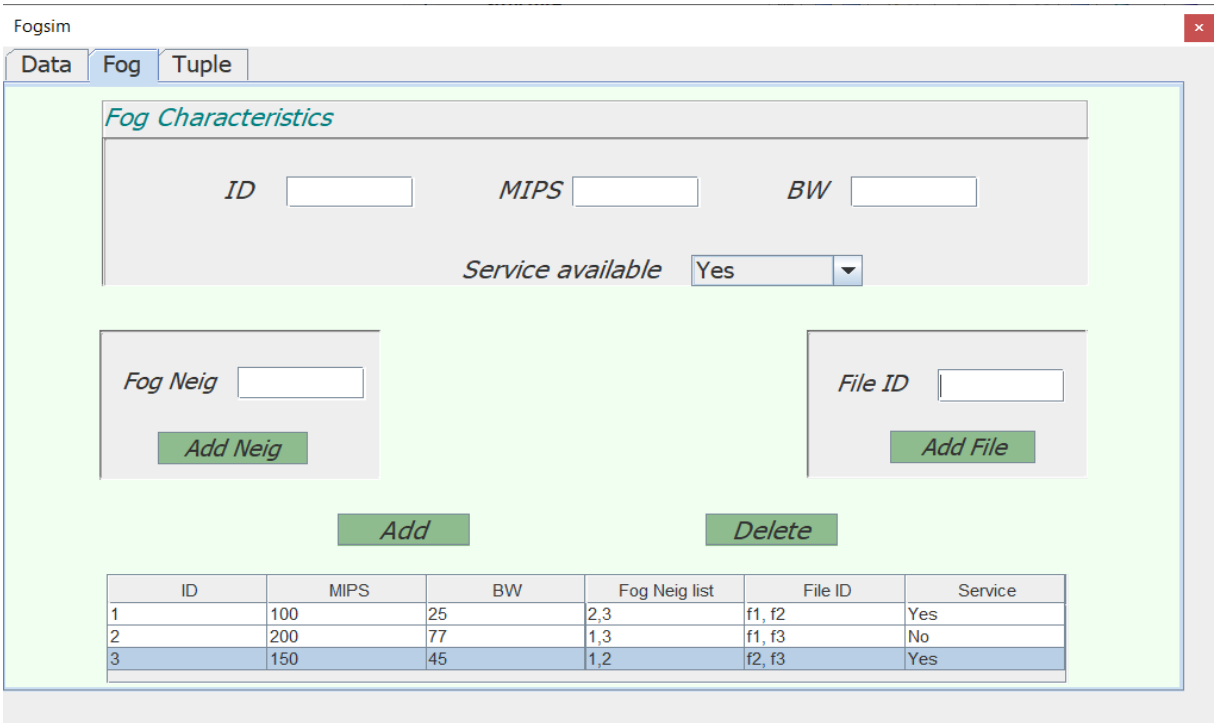

Figure4.3: configuration des fogs

4.5.2.2 Data :

Les requêtes des utilisateurs sont besoins des données pour terminer leurs exécutions.Cette interface nous permet de configurer les données.Nous avons besoin de définir les caractéristiques de ces données telles que le nom de la donnée et sa taille (Figure 4.4).

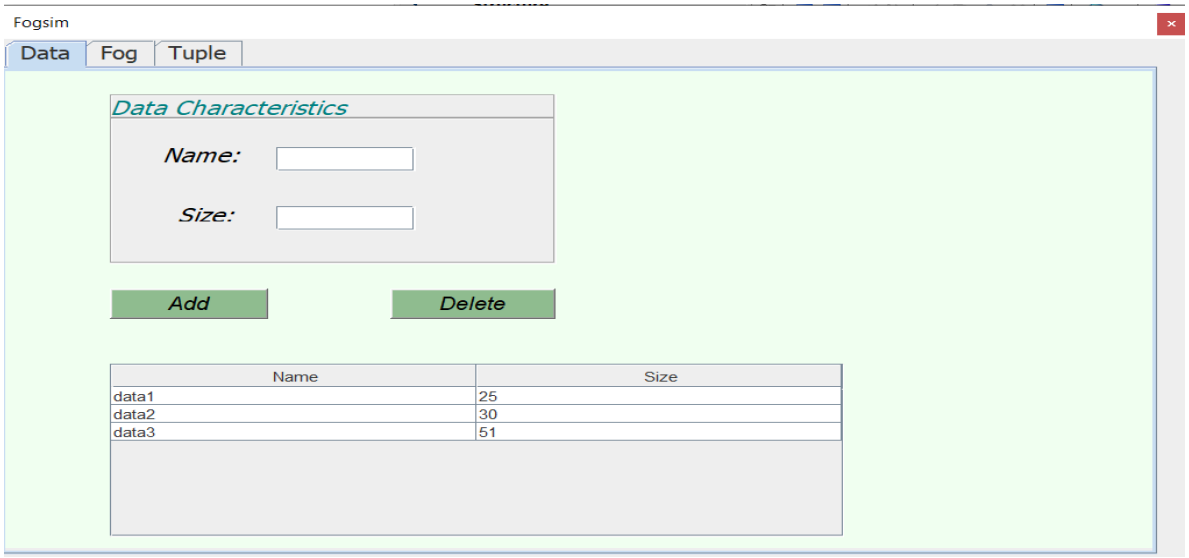

Figure4.4 : configuration des données

4.5.2.3 Tuple:

Cette interface nous permet de configurer les différentes requêtes (Tuple) des utilisateurs, tel que l'identificateur de la requête (ID) à simuler et les données demandées.

Après avoir personnaliser les paramètres de simulation, l'utilisateur peut lancer la simulation en cliquant sur le bouton Start simulation (Figure 4.5).

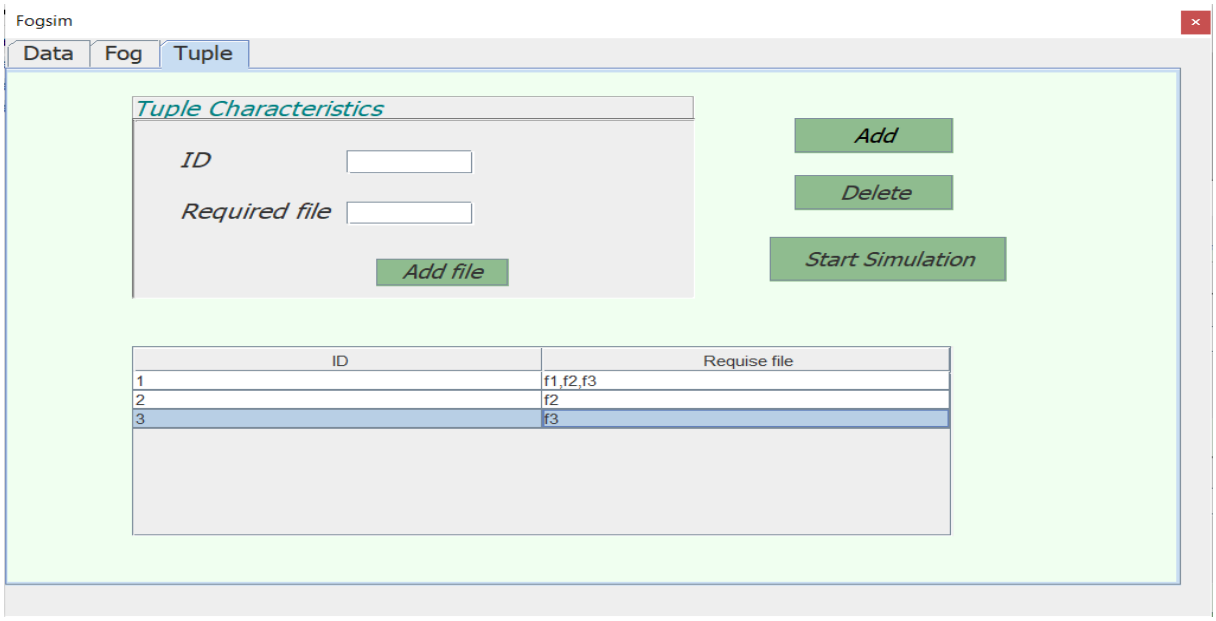

Figure4.5 : configuration des tuples

4.6 Métriques utilisées :

Pour évaluer notre stratégie proposée, nous avons utilisé les métriques suivantes :

4.6.1 Temps de réponse :

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

$$TRm = \sum \frac{TRi}{Nbr}$$

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 :

$$TR = T_{réponse} - T_{envoie}$$

$T_{réponse}$: Temps de réception d'une réponse à la requête.

T_{envoie} : Temps d'envoi de la requête.

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

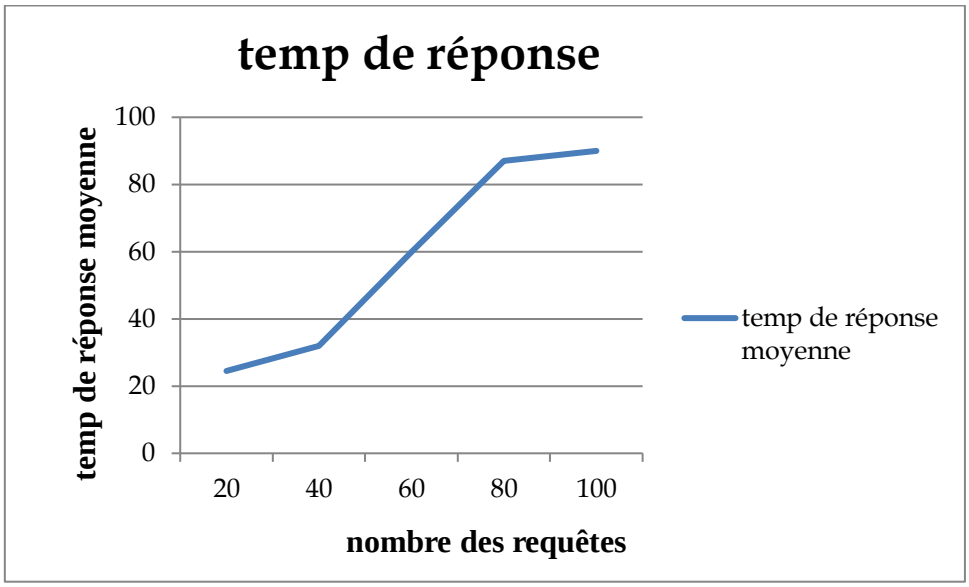

Figure4.6: Graphique linéaire pour l'effet du nombre de requêtes sur le temps de réponse moyen

4.6.3 Consommation d'énergie moyenne :

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

$$CE_m = \sum \frac{CEi}{N}$$

Avec :

CEm : Consommation d'énergie moyenne.

N : Nombre des nœuds.

CEi : Consommation d'énergie du nœud $_i$.

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

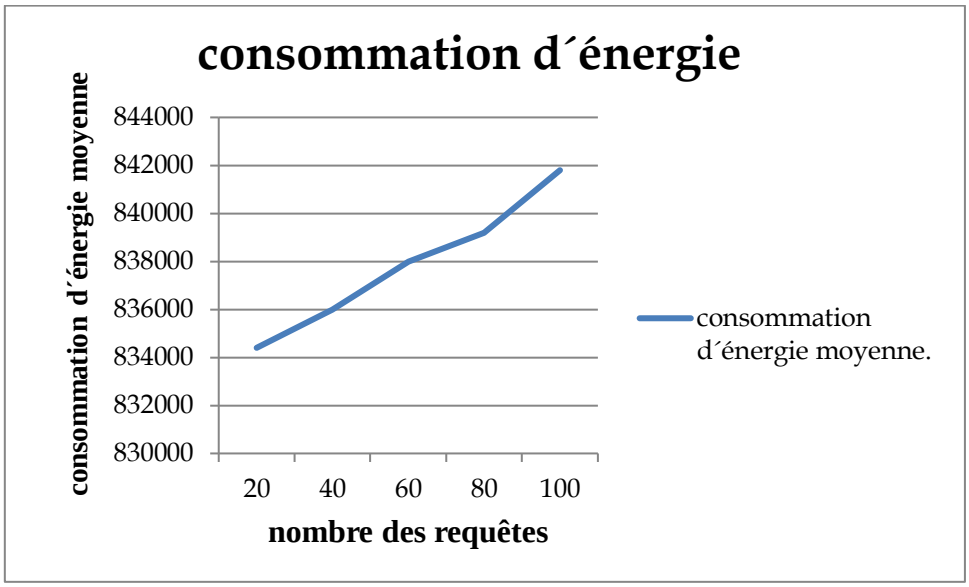

Figure4.7: Graphique linéaire pour l'effet du nombre de requêtes sur la consommation d´énergie moyenne.

4.6.5 Utilisation de la bande passante :

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

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

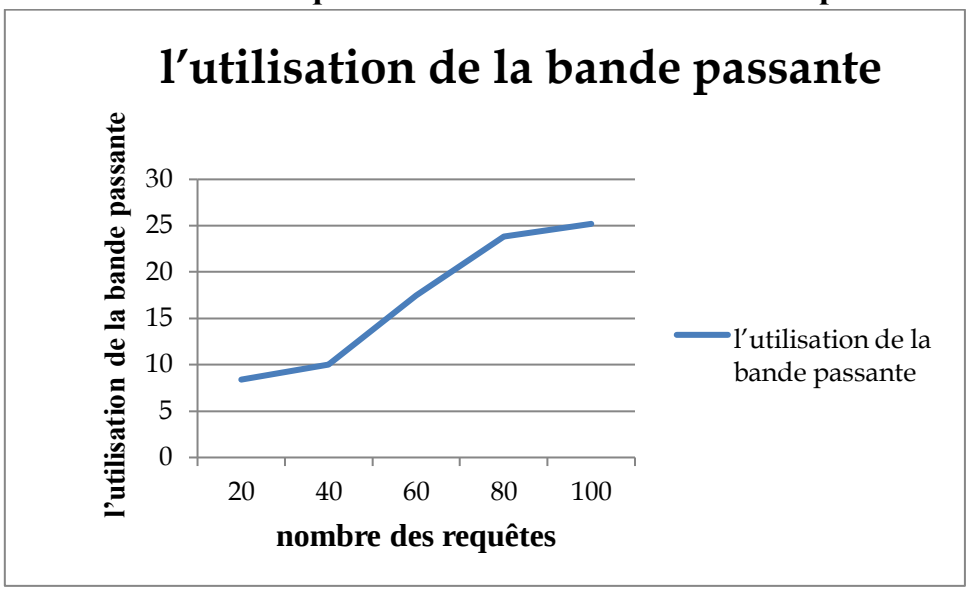

Figure4.8: Graphique linéaire pour l'effet du nombre de requêtes sur l'utilisation de la bande passante

4.7 Conclusion :

Nous avons détaillés dans ce chapitre la phase d'implémentation qui contient le Langage et l'environnement de développement que nous avons utilisés.

Aussi nous avons définir le simulateur IfogSim. Enfin on constate un résultat d'exécution est ce le courbe graphique.

Conclusion général

Notre mémoire avait pour objectif de résoudre un probléme de replication des données dans le fog computing.

La réplication de données est une technique qui permet de régler en général certains Problèmes dans le fog tels que les problèmes de la disponibilité des données à chaque instant.

Cette technique consiste à répliquer les données dans les nœuds selon une disponibilité Exigée par l'utilisateur dont le but d'améliorer la disponibilité et réduire le temps de réponse selon certains critères.

Dans ce travail, nous avons proposé une approche qui consiste à étudier le Traitement d'une requête et la réplication des données, le but est d'optimiser l'exécution des requêtes des utilisateurs.L'approche proposée permet aussi d'équilibrer la charge entre les différents noeuds. Nous avons étendu le simulateur Ifogsim pour implémenter l'approche proposé.

Bibliographie

[1]Accessed on 12 May, 2020. [Online]. Available : https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/.

[2]AAse Dragland. Big data, for better or worse: 90% of world's data generatedover last two years – sciencedaily. https://www.sciencedaily.com/releases/2013/05/130522085217.htm, 05 2013. (Accessed on 5 septembre2020).

[3]Accessedon5spetembre,2020.[Online].Available: https://www.vinci-energies. com/le-fog-computing-la-solution-pour-gerer-des-milliards-dobjets-connectes/

[4] Michael Behrendt, Bernard Glasner, Petra Kopp, Robert Dieckmann, GerdBreiter, Stefan Pappe, Heather Kreger, Ali Arsanjani, Introduction and Architecture Overview IBM Cloud Computing Reference Architecture 2.0, CCRA.IBMSubmission.02282011.doc, V1.0, Draft,(Février 2011).

[5]Pascal Faure, Gabrielle Gauthey, Marie-Caroline Bonnet-Galzy, Guide sur le Cloud Computing et les Datacentersà l'attention des collectivités locales, (Juillet 2015).

[6]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.

[7]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).

[8]Bonomi, Flavio, Rodolfo Milito, Jiang Zhu et Sateesh Addepalli (2012). « Fog Computing and Its Role in the Internet of Things». Dans: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing. MCC '12. Helsinki, Finland: ACM, p. 13-16. isbn: 978-1-4503-1519-7. doi: 10.1145/2342509.2342513.

[9]Lebre,Adrien,JonathanPastor,MarinBertier,FrédéricDesprez,JonathanRouzaudCornabas,CédricTedeschi,PaoloAnedda,GianluigiZanetti,RamonNou,Toni Cortes, Etienne Rivière et Thomas Ropars (juil. 2013). Beyond The Cloud, How Should Next Generation Utility Computing Infrastructures Be Designed? Research Report RR-8348. INRIA.

[10]Bonomi, Flavio, Rodolfo Milito, Preethi Natarajan et Jiang Zhu (2014). « Fog Computing : A Platform for Internet of Things and Analytics». Dans: Big Data and Internet of Things: A Roadmap for Smart Environments. Sous la dir. de Nik Bessis et Ciprian Dobre. Cham : Springer International Publishing, p. 169-186. isbn : 978-3-319-05029-4. doi : 10.1007/978-3-319-05029-4_7.

[11]Firdhous, Mohamed, Osman Ghazali et Suhaidi Hassan (2014). «Fog Computing: Will it be the Future of Cloud Computing?» Dans: Third International Conference on Informatics & Applications, Kuala Terengganu, Malaysia, p. 8-15. isbn : 978-1941968-00-0.

[12]Byers, Charles C. et Patrick Wetterwald (nov. 2015). «Fog Computing Distributing Data and Intelligence for Resiliency and Scale Necessary for IoT: The Internet of Things (Ubiquity Symposium) ». Dans: Ubiquity 2015.November, 4:1-4:12. issn: 1530-2180. doi : 10.1145/2822875.

[13]Aazam, Mohammad et Eui-Nam Huh (2014). «Fog Computing and Smart Gateway Based Communication for Cloud of Things ». Dans: Proceedings of the 2014 International Conference on Future Internet of Things and Cloud. FICLOUD '14. Washington, DC, USA: IEEEComputerSociety, p.464-470. isbn: 978-1-4799-4357-9.

doi:10.1109/FiCloud.2014.83.

[14]Bittencourt, L. F., J. Diaz-Montes, R. Buyya, O. F. Rana et M. Parashar (mar. 2017). «Mobility-Aware Application Scheduling in Fog Computing». Dans : IEEE Cloud Computing 4.2, p. 26-35. issn : 2325-6095. doi : 10.1109/MCC.2017.27.

[15]Bittencourt, L. F., J. Diaz-Montes, R. Buyya, O. F. Rana et M. Parashar (mar. 2017). «Mobility-Aware Application Scheduling in Fog Computing». Dans: IEEE Cloud Computing 4.2, p. 26-35. issn: 2325-6095. doi: 10.1109/MCC.2017.27.

[16]Bittencourt, L. F., J. Diaz-Montes, R. Buyya, O. F. Rana et M. Parashar (mar. 2017). «Mobility-Aware Application Scheduling in Fog Computing». Dans: IEEE Cloud Computing 4.2, p. 26-35. issn : 2325-6095. doi : 10.1109/MCC.2017.27.

[17]Bittencourt, L. F., J. Diaz-Montes, R. Buyya, O. F. Rana et M. Parashar (mar. 2017). «Mobility-Aware Application Scheduling in Fog Computing». Dans: IEEE Cloud Computing 4.2, p. 26-35. issn: 2325-6095. doi: 10.1109/MCC.2017.27.

[18]Ranganathan. K and Ian Foster. Identifying dynamic replication strategies for a high-performance data grid. In In Proc. of the International Grid Computing Workshop, volume 2242, pages 75–86. Springer, 2001.

[19]Tang. X and Jianliang Xu. QoS-aware replica placement for content distribution. In IEEE Transactions on Parallel and Distributed Systems, volume 16, pages 921–932, oct 2005. doi: 10.1109/TPDS.2005.126.

[20]Du. Z, Jingkun Hu, Yinong Chen, Zhili Cheng, and Xiaoying Wang. Optimized qosaware replica placement heuristics and applications in astronomy data grid. Journal of Systems and Software, 84(7):1224–1232, jul 2011. ISSN 01641212. doi: 10.1016/j.jss.2011.02.038.

[21]Kemme B, Ricardo Jimenez-Peris, and Marta Patino-Martinez. Database Replication. Morgan and Claypool Publishers, 2010.

[22]Özsu, T M. and Patrick Valduriez. Principles of Distributed Database Systems. Springer New York, New York, NY, 3rd edition, 2011. ISBN 978-1-4419-8833-1. doi: 10.1007/978-1-4419-8834-8.

[23]Guerrero-Contreras. G, Carlos Rodriguez-Dominguez, Sara Balderas-Diaz, and JoseLuis Garrido. Dynamic replication and deployment of services in mobile environments. In New Contributions in Information Systems and Technologies, pages 855–864. Springer International Publishing, 2015.

[24]Spaho. E, Admir Barolli, Fatos Xhafa, and Leonard Barolli. P2P data replication: Techniques and applications. In Modeling and Processing for NextGeneration BigData Technologies, pages 145–166. Springer International Publishing, 2015.

[25]Tos. U, Riad Mokadem, Abdelkader Hameurlain, Tolga Ayav, and Sebnem Bora. Dynamic replication strategies in data grid systems: a survey. The Journal of Supercomputing, 71(11):4116–4140, 2015. ISSN 0920-8542. doi: 10.1007/s11227015-1508-7.

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

[27] 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.

[28]Fei Xie; Jun Yan; Jun Shen, "Towards Cost Reduction in Cloud-Based Workflow Management," Fifth International Conference on Advanced Cloud and Big Data, (2017).

[29]T. Pankowski, "Lorq: A System for Replicated NoSQL Data," Evaluation of Novel Approaches to Software Engineering. ENASE 2015, pp. 62-79, (2016).

[30] Sebastian Burckhardt, "Replicated data types: specification, verification, optimality," ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, (2014).

[31]Xiuguo Wu, "Data Sets Replicas Placements Strategy from Cost-Effective View in the Cloud," Hindawi Publishing Corporation, pp. 1-14, (2016).

[32]Sathiya Prabhu Kumar, Sylvain Lefebvre,"CaLibRe: A better Consistency-Latency Tradeoff for Quorum based Replication systems," Database and Expert Systems Applications. Globe (2015), DEXA 2015. Lecture Notes in Computer Science, pp. 2-14,(2015).

[33]Uras Tos ; Riad Mokadem ; Abdelkader Hameurlain ; Tolga Ayav ; Sebnem Bora , "A Performance and Profit Oriented Data Replication Strategy for Cloud Systems," IEEE Conferences on Ubiquitous Intelligence & Computing, (2016).

[34]Yaser Mansouri; Adel Nadjaran Toosi; Rajkumar Buyya, "Cost Optimization for Dynamic Replication and Migration of Data in Cloud Data Centers," IEEE Transactions on Cloud Computing, pp. 1-16, (2017).

[35]https://www.cleo.com/blog/knowledge-base-what-is-cloud-computing visiter le 7 mai 2022.

[36] https://www.luitinfotech.com/kc/what-is-cloud-computing.pdf visiter le 8 mai 2022.

[37]S. Hwang5. N. N. Danget S. B.Lim. « Improvement of data grid's perfor-mance by combining job scheduling with dynamic replication strategy. In GCC'07: Proceedings of the Sixth International Conference on Grid and Coopera-tive Computing, pp. 513–520 (Washington, DC, USA, » in: (2007) (cf. p. 22, 30, 31)

[38]L.MOINE. « La gestion et la sécurite dans une architecture de ressourcesde calcul distribuées sur l'Internet.

Mémoire d'ingénieur c.n.a.m. en informa-tique, UREC (Unite Réseaux du CNRS), Centre d'enseignement de Grenoble, Grenoble Cedex 9,
France ». In : (2002) (cf. p. 30).

[39] Mémoire Une stratégie de réplication dynamique de données dans les Fog Computing Ayadi yazid, Promotion 2021, UNIVERSITÉ DR MOULAY TAHAR – SAIDA.

[40] 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.

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

ملخص

في السنوات القادمة ، سيكون إنترنت الأشياء (IoT) أحد أكثر تطبيقات توليد البيانات شيوعًا. بحلول عام 2025 ، سيتم توصيل ما يقرب من 75.44 مليار كائن.لا يمكن أن تتطور معالجة بيانات إنترنت الأشياء في البيئة السحابية بسبب حوسبة الضباب ، وهي أقرب إلى الكائنات المتصلة. بالإضافة إلى ذلك، فإنه يسهل تخزين البيانات وتحليلها.في هذا العمل ، نقترح استراتيجية تكرار البيانات في بيئة الحوسبة الضبابية مما يجعل من الممكن معالجة طلبات المستخدم في الضباب الأكثر كفاءة وإنشاء نسخ متماثلة للبيانات لتحسين الأداء مثل استغلال وقت الاستجابة وتقليل استهلاك النطاق الترددي.

قمنا بتقييم استراتيجيتنا باستخدام عدة معايير ، وأظهرت النتائج فعالية اقتراحنا.

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

Abstract

In the coming years, the Internet of Things (IoT) will be one of the most popular data generation applications. By 2025, nearly 75.44 billion objects will be connected. The processing of IoT data in the cloud environment can hardly evolves due to fog computing, which is closer to connected objects. In addition, it facilitates data storage and analysis. In this work, we propose a data replication strategy in the Fog computing environment which makes it possible to process user requests in the most efficient fogs and create replicas data to improve performance such as response time utilization and reduce bandwidth consumption.

We evaluated our strategy using several parameters. The results show the effectiveness of our proposal.

Keywords: Fog computing, Cloud computing, Replication.

Résumé

Dans les années à venir, l'Internet des objets (IoT) sera l'une des applications de génération de données les plus populaires. D'ici 2025, près de 75,44 milliard d'objets seront connectés.le traitement des données IoT dans l'environnement cloud peut difficilement évoluer par apport le fog computing qu'il est au plus près des objets connectés. De plus, il facilite le stockage et l'analyse de données.Dans ce travail, nous proposons une stratégie de réplication des données dans l'environnement du Fog computing qui permet de traiter les requêtes des utilisateurs dans les fogs les plus performants et créer des répliques de donées pour améliorer les performances telles que l'utilisation du temps de réponse et réduire la consommation de la bande passante .

Nous avons évalué notre stratégie en utilisant plusieurs paramètres .Les résultas montrent l'efficacité de notre proposition.

Mots clés : Fog computing, Cloud computing, Réplication.