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

Université Abderrahmane Mira
Faculté de la Technologie

Département d'Automatique, Télécommunication et d'Électronique

---

Projet de Fin d'Études

Pour l'obtention du diplôme de Master

Filière : Télécommunications
Spécialité : Réseaux et télécommunications

Thème

Task Offloading dans un environnement de Fog Computing

Préparé par :
Arab Fariza
Ahannouh Djida

Examiné par :
M Diboune Abdelhani
Mme Gherbi Meriem

Dirigé par :
Mme Mammeri Karima
Mme Azizou Zahia

Année universitaire : 2024/2025

Remerciements

Allah nous a accordé la force, la santé, l'endurance et la patience tout au long de ce parcours. Nous exprimons notre sincère reconnaissance à notre encadrante **Mme Mammeri Karima**, ainsi qu'à **Mme Azizou Zahia**, pour leur disponibilité, leurs conseils avisés, leur suivi bienveillant et leur accompagnement précieux tout au long de ce travail

Nous tenons à remercier nos familles pour leur soutien inconditionnel, leur patience, leurs encouragements constants et les sacrifices qu'ils ont consentis pour que nous puissions mener à bien nos études.

Nous adressons également notre gratitude à l'ensemble de nos enseignants de l'Université **Abderrahmane Mira de Béjaïa**, pour la qualité de leur enseignement, leur rigueur académique, et l'ensemble des connaissances qu'ils nous ont transmises au fil des années.

Nos remerciements vont aussi aux responsables et encadrants qui nous ont permis d'effectuer nos stages, ainsi qu'aux professionnels rencontrés, pour leur accueil, leur disponibilité et les échanges enrichissants.

Nous remercions chaleureusement nos camarades de promotion, nos collègues, ainsi que nos amis pour leur entraide, leurs partages, leurs encouragements, et tous ces moments d'amitié qui ont rendu cette période plus humaine et motivante.

Enfin, nous remercions les membres du jury pour le temps qu'ils ont consacré à l'évaluation de notre travail, ainsi que pour leurs remarques et observations constructives.

Ce mémoire est le reflet d'un parcours commun, un effort partagé, et une étape importante franchie avec engagement et persévérance.

Dédicace

Je rends d'abord grâce à Allah, Le Tout-Puissant, Le Très Miséricordieux, sans qui rien de tout cela n'aurait été possible

À ma mère, pour son amour inconditionnel, sa patience, ses prières et son soutien constant, qui m'ont toujours donné la force d'avancer.

À la mémoire de mon père, dont les conseils, les valeurs et l'affection continuent de m'accompagner malgré son absence. Ce travail est aussi le fruit de son éducation.

À Fariza, ma binôme et amie, pour sa collaboration fidèle, sa persévérance et l'esprit d'équipe que nous avons partagé tout au long de ce projet.

À mes frères, Hmimi, Redouane et Tiziri, pour leur présence rassurante, leurs encouragements et leur affection fraternelle.

À mes voisins, Larbi et Fatima, ainsi qu'à leurs enfants ravah, Didou, Samia, Mohaned, Ahmed(paix a son âme), Kahina, Fatma et hakim, pour leur générosité, leur accueil chaleureux et leur présence réconfortante.

À mes amies proches, Daya, Rania et Saida, que je considère comme de véritables sœurs, pour leur écoute, leur bienveillance et leur soutien indéfectible.

À mes camarades de promotion : Tamazight, Ouardia, Loubna,Khoukha, Bouchra, Meriem, Yanis, Khawla et Sonia, pour l'ambiance de travail, la solidarité et les bons souvenirs partagés durant ces années.

À toutes les personnes qui m'ont soutenue sans rien attendre en retour, par un mot, un geste ou une présence bienveillante, je leur exprime ma profonde reconnaissance.

Djida Ahannouh

Dédicace

Je rends d'abord grâce à Allah, Le Tout-Puissant, Le Très Miséricordieux, sans qui rien de tout cela n'aurait été possible.

Je tiens à exprimer ma profonde gratitude à mes parents, pour leur amour inconditionnel, leurs prières constantes, leur patience et leurs sacrifices. Leur présence à mes côtés, a toujours été une source de force et de réconfort. Qu'Allah les récompense pour tout le bien qu'ils m'ont apporté.

À mes sœurs, Celia, Manissa et Neima, je dis merci pour leur affection, leurs encouragements et leur confiance en moi. Elles ont su, chacune à sa manière, m'apporter du soutien dans les moments de doute et de fatigue. Je remercie également ma binôme, avec qui j'ai partagé cette

aventure. Ta rigueur, ton esprit d'équipe et ta bienveillance ont grandement contribué à rendre cette expérience plus riche et plus humaine. Merci pour ta patience, ton écoute et ta bonne humeur.

Et enfin, je n'oublie pas jiji, pour son amitié sincère, ses encouragements et sa présence rassurante durant cette période exigeante.

Fariza Arab

Table des matières

Table des figures

Liste des tableaux

Liste des acronymes

ABC *Aartifial Bee colony*

AR/VR *Augmented Reality / Virtual Reality*

AWS *Amazon Web Services*

BLA *BeeLife Algorithm*

CPU *Central Processing Unit*

ETO *Enhanced Task Offloading*

FBSs *Fog Base Stations*

FUAVs *Fog Enmanned Aerial Vehicles*

GA *Genetic Algorithm*

GCP *Google Cloud Platform*

gBest *global Best*

IaaS *Infrastructure as a Service*

IoT *Internet of Things*

ML *Machine Learning*

MFA *Modified Firefly Algorithm*

NIST *National Institute of Standards and Technology*

NP *Non-deterministic Polynomial-time*

P *Polynomial-time*

PaaS *Platform as a Service*

pBest *personal Best*

PSO *Particle Swarm Optimization*

QoS *Quality of Service*

RFID ***R****adio* ***F****requency* ***I****dentification*

SaaS ***S****oftware* ***a****s* ***a*** ***S****ervice*

SACO ***S****ensitive* ***A****nt* ***C****olony* ***O****ptimization*

Introduction Générale

L'Internet des Objets (IoT) est un réseau permettant à une multitude d'objets physiques et virtuels d'être interconnectés afin de collecter, d'échanger et de traiter des données de manière intelligente.

Le Cloud Computing, quant à lui, offre une solution puissante pour le traitement et le stockage de ces données. Il permet l'accès à des ressources informatiques centralisées et évolutives, tout en assurant une haute disponibilité des services. L'intégration de l'IoT avec le Cloud permet d'optimiser la collecte et le traitement de données sur de grandes distances.

Cependant, cette architecture centralisée présente des limitations majeures lorsqu'il s'agit d'applications nécessitant une faible latence, une réactivité en temps réel ou une réduction de la consommation d'énergie. L'envoi systématique des données vers le Cloud peut engendrer des délais importants et une surcharge du réseau, ce qui rend cette approche inadaptée pour certains usages critiques.

C'est dans ce contexte que le Fog Computing a été introduit comme une solution intermédiaire entre le Cloud et les objets connecté. Cette technologie étend les capacités du Cloud vers la périphérie du réseau, en déployant des nœuds de traitement (Fog nodes) à proximité des sources de données. Le Fog optimise la prise de décision locale, réduit la charge réseau et améliore la gestion des ressources dans les systèmes distribués.

Cependant, une question centrale subsiste : comment et où exécuter les tâches générées par les objets IoT ? Faut-il les traiter localement, sur un nœud Fog, ou envoyer certaines d'entre elles au Cloud ? Ce processus, appelé Task Offloading, est un problème complexe d'optimisation impliquant plusieurs critères de performances, tels que la latence, la consommation d'énergie, la capacité des ressources disponibles et la qualité de service (QoS).

Dans ce travail, nous proposons une étude approfondie du Task Offloading dans un environnement de Fog Computing, en explorant les approches, les classifications, les critères de décision, ainsi que les algorithmes d'optimisation les plus adaptés. Notre étude s'intéresse particulièrement à l'adaptation de la metaheuristique ABC (Artificial Bee Colony) pour résoudre le problème de Task Offloading dans l'environnement Fog Computing. Cette algorithme est comparer avec une autre metaheuristique PSO (Particle Swarm Optimization), selon leur capacité à réduire le temps d'exécution et a minimiser la consommation d'énergie globale.

Notre mémoire est structuré en trois chapitres, détaillés comme suit :

Chapitre I « Généralités sur l'IoT, le Fog et le Cloud Computing » Chapitre introduit les concepts fondamentaux de l'Internet des objets (IoT), du Cloud et du Fog Computing. Il présente leurs architectures ainsi que leurs principales caractéristiques. Cette mise en contexte permet de mieux comprendre les enjeux liées à l'évolution des technologies distribuées.

Chapitre II « Task Offloading dans le Fog Computing » traite le Task Offloading dans le Fog

Computing, en présentant ses principes, les acteurs impliqués, les classifications des approches et les algorithmes existants.

Chapitre III « Proposition et évaluation de performances » présente notre contribution ainsi que l'évaluation en simulation de ces performances ; nous avons adapter la metaheuristique pour résoudre problème de Task Offloading et l'algorithme dans l'environnement de Fog Computing.

Chapitre I

Généralités sur l'IoT, le Fog et le Cloud Computing

I.1 Introduction

L'Internet des objets (IoT) représente l'une des avancées technologiques majeures de notre époque. Il connecte des milliards d'appareils à travers le monde. Selon Statista [1], le nombre d'appareils IoT devrait atteindre 32,1 milliards d'ici 2030. Cette croissance génère une quantité massive de données, qui nécessitent un traitement rapide et efficace. Cette expansion soulève des défis importants. Le traitement, le stockage et la gestion des données deviennent critiques, surtout pour les applications nécessitant une faible latence ou une bande passante optimisée.

Pour répondre à ces contraintes, deux modèles sont souvent utilisés de manière complémentaire, le Cloud Computing et le Fog Computing pour optimiser les architectures IoT [2].

Dans ce chapitre, nous présentons les concepts fondamentaux de l'IoT, du Cloud Computing et du Fog Computing. Nous mettons en avant leurs caractéristiques, leurs avantages et leurs interactions. Cette base est nécessaire pour comprendre les enjeux techniques et les usages possibles de ces technologies.

I.2 Internet des objets (IoT)

I.2.1 Définition

L'Internet des objets désigne un réseau d'objets physiques connectés. Ces objets sont équipés de capteurs, de processeurs et de modules de communication. Ils collectent, échangent et traitent des données via Internet. Ces objets peuvent interagir entre eux et avec des systèmes centraux. Ils permettent des services automatisés, en assurant la surveillance, le contrôle et l'optimisation à distance, en temps réel [3].

L'IoT repose sur des technologies comme l'identification par radiofréquence (RFID), les réseaux de capteurs sans fil et des protocoles de communication spécifiques. Ces technologies permettent une interaction fluide entre le monde physique et numérique, ouvrant la voie à des applications innovantes dans divers domaines [4].

I.2.2 Architecture de l'IoT

L'architecture de l'IoT est structurée en plusieurs couches, chacune jouant un rôle précis dans la collecte, la transmission et le traitement des données. Cette organisation modulaire et évolutive permet de gérer efficacement les systèmes IoT, en assurant une interaction fluide entre les dispositifs physiques, les réseaux de communication et les applications de traitement.

Bien que les architectures IoT puissent varier, elles reposent généralement sur trois couches principales : la couche perception, réseau et application.

Comme illustré dans la figure I.1, cette structure de base fournit un cadre essentiel pour comprendre les fonctionnalités clés et les considérations de conception d'un système IoT intégré [5].

Fig. I.1 : Architecture en couches de l'IoT
[6]

Couche perception (couche physique)

La couche perception, également appelée couche physique, constitue la base de l'architecture IoT. Elle collecte des données issues de l'environnement physique à l'aide de dispositifs comme les capteurs, les actionneurs et les étiquettes RFID (Radio Frequency Identification).

Les capteurs mesurent des paramètres physiques comme la température, l'humidité ou le mouvement. Les actionneurs exécutent des actions physiques en réponse à des commandes reçues.

Le RFID est l'une des technologies utilisées dans cette couche. Il utilise les ondes radio pour identifier et suivre des objets à distance. Un système RFID se compose d'une étiquette attachée à un objet et d'un lecteur qui interroge cette étiquette. Chaque étiquette contient un identifiant unique appelé EPC (Electronic Product Code).

Couche réseau (couche de communication)

La couche réseau assure la transmission des données collectées par la couche perception vers la couche application ou vers d'autres dispositifs. Elle utilise une variété de technologies de communication, notamment les réseaux filaires (comme Ethernet) et les réseaux sans fil (tels que Bluetooth, Zigbee, et les réseaux cellulaires 4G/5G). Les passerelles jouent un rôle clé dans

cette couche en reliant les capteurs aux réseaux de communication et en assurant la conversion des protocoles. Cette couche garantit également la sécurité des données lors de la transmission grâce à des protocoles de chiffrement [7].

Couche application

La couche application traite et analyse les données transmises par la couche réseau afin de les transformer en informations exploitables [8]. Elle organise ces données et les met à disposition des utilisateurs sous une forme compréhensible. Elle assure également la gestion des dispositifs connectés et la coordination de leurs échanges [9]. Cette couche joue un rôle clé dans l'optimisation du fonctionnement du système en assurant une interprétation efficace des données et en facilitant l'automatisation des actions en fonction des analyses effectuées [8]. Elle permet une utilisation efficace des informations recueillies et une meilleure exploitation des objets connectés [10].

I.2.3 Domaines d'application de l'IoT

L'Internet des objets est devenu une technologie essentielle dans plusieurs secteurs. Il propose des solutions pour améliorer l'efficacité, la productivité et l'automatisation [11]. La Figure I.2 montre ces différents domaines d'application.

Environnement intelligent

Les environnements intelligents incluent les villes intelligentes (Smart Cities) et les maisons connectées. L'IoT est utilisé pour améliorer la vie quotidienne, économiser l'énergie et limiter l'impact sur l'environnement. Dans les villes intelligentes, l'IoT aide à gérer la circulation avec des feux intelligents et des capteurs de trafic. L'éclairage public s'adapte à la lumière du jour et à la présence de piétons. Des capteurs dans les poubelles préviennent quand elles sont pleines. Dans les maisons intelligentes, l'IoT permet de contrôler la sécurité avec les caméras et les serrures connectées. On peut aussi gérer la consommation d'énergie avec des thermostats ou des ampoules intelligentes. Les prises connectées permettent d'éteindre ou d'allumer les appareils à distance. Les assistants vocaux comme Google Home ou Amazon Echo contrôlent la lumière, la musique, la météo ou une alarme à la voix [7].

Soins de santé

L'IoT a révolutionné la santé en permettant une surveillance continue des patients et une meilleure gestion des équipements médicaux. La télémédecine permet des consultations à distance et la surveillance des patients à domiciles, ce qui réduit les déplacements à l'hôpital. Les dispositifs portables, comme les montres connectées ou les capteurs de santé, mesurent en temps réel des données comme la fréquence cardiaque ou la tension artérielle. Dans les hôpitaux, l'IoT aide à suivre des équipements médicaux et à gérer les stocks de médicaments, ce qui rend le fonctionnement plus efficace [12].

Agriculture intelligente (Smart Agriculture)

L'agriculture intelligente utilise l'IoT pour mieux gérer les ressources et améliorer la productivité. Les capteurs d'humidité dans le sol régulent l'irrigation en fonction des besoins des plantes, ce qui réduit l'utilisation d'eau. Les drones et les capteurs surveillent la santé des plantes, détectent les maladies et optimisent l'utilisation des engrais. Dans l'élevage, l'IoT aide à suivre la santé des animaux et à gérer les troupeaux en temps réel [13].

Transport et logistique

L'IoT transforme le secteur du transport et de la logistique avec des solutions de suivi et d'optimisation en temps réel. Les véhicules connectés sont équipés de capteurs pour surveiller leur état et leur localisation, ce qui aide à gérer les flottes et à améliorer la sécurité routière. En logistique, l'IoT permet de suivre des colis, d'optimiser les chaînes d'approvisionnement et de gérer les entrepôts, ce qui réduit les coûts et améliorant l'efficacité [14].

Industrie et fabrication

L'IoT joue un rôle central dans la révolution industrielle. La maintenance prédictive utilise des capteurs pour surveiller l'état des machines et prévenir les pannes. Cela réduit les temps d'arrêt et les coûts de maintenance. Les usines intelligentes intègrent des robots et des systèmes connectés pour automatiser les processus de production. Cela améliore l'efficacité et la qualité. La gestion de la chaîne d'approvisionnement est aussi optimisée. L'IoT permet un suivi en temps réel des stocks et des livraisons [14].

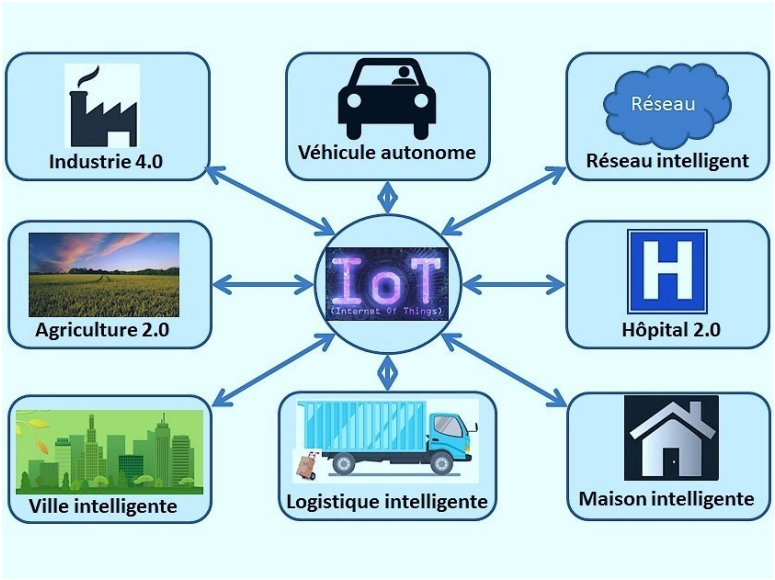

Fig. I.2 : Synthèse des domaines d'application de l'IoT [15]

I.3 Cloud computing

Le développement rapide du Cloud Computing ces dernières années a créé une situation typique pour de nombreuses innovations et technologies émergentes. Bien que de nombreuses entreprises aient entendu parler du Cloud, peu d'entre elles en comprennent véritablement les enjeux et savent comment l'adopter et en tirer profit. Le Cloud Computing a un impact considérable sur le domaine de l'informatique. Il offre une nouvelle plateforme pour naviguer dans un contexte économique complexe [16].

I.3.1 Définition du Cloud Computing

Le Cloud Computing, ou informatique en nuage, désigne un modèle dans lequel la puissance de calcul et le stockage sont gérés par des serveurs distants, accessibles via une connexion Internet sécurisée. Les utilisateurs peuvent ainsi se connecter à ces ressources depuis un ordinateur, un téléphone portable, une tablette tactile ou de tout autre appareil connecte. Cela leur permet d'exécuter des applications et d'accéder à leurs données hébergées sur ces serveurs.

L'un des principaux avantages du Cloud réside dans sa flexibilité. Les fournisseurs peuvent ajuster dynamiquement les capacités de stockage et de calcul en fonction des besoins des utilisateurs. Pour le grand public, cela se traduit par des services comme Dropbox, OneDrive ou iCloud, qui permettent de stoker et partager des fichiers accessibles partout et à tout moment [17].

I.3.2 Caractéristiques du Cloud Computing

Généralement, un service, une solution ou un environnement d'exécution doit satisfaire à une série de caractéristiques pour être considéré comme du Cloud Computing. Parmi ces caractéristiques, certaines sont reconnues comme fondamentales. Par exemple, le National Institute of Standards and Technology (NIST) définit cinq caractéristiques essentielles qui sont [18] :

Ressources à la demande

Un utilisateur peut allouer unilatéralement des ressources informatiques (serveurs, réseau, stockage, environnement d'exécution, application) au besoin, de façon automatique et sans nécessité d'interaction humaine avec chaque fournisseur de services.

Large accès réseau

Les ressources Cloud Computing sont disponibles à travers le réseau et accessibles via des mécanismes standards qui favorisent leur utilisation à partir des appareils clients hétérogènes, voire légères comme ordinateurs portables, téléphones, tablettes.

Mutualisation des ressources

Les ressources informatiques du fournisseur Cloud Computing sont partagées pour servir plusieurs clients en utilisant un modèle multi-tenant. Ces ressources, physiques ou virtuelles,

sont allouées et libérées dynamiquement selon la demande du consommateur. L'utilisateur n'a généralement ni le contrôle ni la connaissance de l'emplacement exact des ressources allouées. Dans certains cas, il peut choisir une zone géographique large, comme un pays, un continent ou Data Center.

Élasticité rapide

Les ressources Cloud peuvent être allouées et libérées dynamiquement en fonction des variations de la demande. Cela permet une adaptation automatique et instantanée aux besoins des utilisateurs, donnant l'impression que les ressources sont illimitées.

Services mesurés

L'utilisation des ressources est surveillée et contrôlée pour mesurer leur consommation, comme le stockage, la puissance de calcul ou la bande passante. Cela permet une facturation précise et une gestion optimisée des services Cloud.

I.3.3 Types du Cloud Computing

Cloud Privé

Le Cloud privé se distingue par le fait qu'il s'agit d'une infrastructure Cloud exclusivement dédiée à une seule organisation. Ce modèle peut être hébergé en interne, au sein des locaux de l'entreprise, ou externe auprès d'un fournisseur de services. Le Cloud privé se caractérise par un niveau de contrôle élevé, une sécurité optimisée et une grande flexibilité en termes de personnalisation. Ces avantages en font une solution privilégiée pour les entreprises manipulant des données sensibles ou soumises à des exigences strictes en matière de conformité et de sécurité [19].

Cloud Communautaire (Community Cloud)

Le Cloud Communautaire fonctionne de la même manière qu'un Cloud privé, mais avec ce modèle, plusieurs clients partagent une instance matérielle dédiée. La combinaison d'utilisateurs n'est pas choisie au hasard, mais plusieurs clients, pour la plupart issus du même secteur d'activité ou ayant des intérêts similaires, se réunissent de manière ciblée. Le Cloud Communautaire peut aussi être géré au sein d'une entreprise ou à l'extérieur. L'objectif est de réaliser des économies par rapport à plusieurs Clouds privés [20].

Cloud Public

Le Cloud public est l'un des modèles de Cloud Computing les plus répandus. Il repose sur des services fournis via Internet par des prestataires externes tels qu'Amazon Web Services (AWS), Microsoft Azure ou Google Cloud Platform (GCP). Les utilisateurs partagent une même plateforme pour accéder aux ressources informatiques, qui sont entièrement gérées par le fournisseur. Ce modèle est particulièrement apprécié pour sa simplicité d'accès, sa flexibilité

et son système de paiement à l'usage, qui permet aux entreprises d'éviter des investissements lourds en infrastructures [19].

Cloud Hybride

Il s'agit d'un modèle qui combine le Cloud privé et le Cloud public. Il permet de stocker certaines données sensibles en privé tout en utilisant les ressources du Cloud public pour d'autres besoins. Ce modèle offre plus de flexibilité, une meilleure gestion des coûts et un bon équilibre entre sécurité et performance [20].

I.3.4 Modèles de service du Cloud

Le marché du Cloud Computing propose plusieurs types de services, comme l'IaaS, le PaaS et le SaaS, chacun étant adaptés à différents besoins informatiques. Comprendre ces modèles est essentiel pour les organisations qui souhaitent optimiser leurs opérations, améliorer leur efficacité et favoriser leur croissance en exploitant les avantages du Cloud [19].

Infrastructure as a Service (IaaS)

Dans le cadre de l'Infrastructure en tant que Service (IaaS), des solutions matérielles complètes sont proposées, incluant la performance des processeurs, l'espace de stockage et la technologie réseau. Les instances utilisées par l'utilisateur sont entièrement virtualisées et réparties dans un pool de ressources. Ce modèle peut servir de fondation pour les autres couches du Cloud, mais il est également disponible en tant que produit autonome [20].

Platform as a Service (PaaS)

Ce modèle se distingue en offrant non seulement une infrastructure matérielle, mais également un environnement de développement complet. La Plateforme en tant que Service (PaaS) cible principalement les développeurs de logiciels. Le fournisseur propose un environnement de développement préconfiguré dans le cloud, hébergé sur ses serveurs. Cela permet aux développeurs de gagner du temps en évitant la configuration et la maintenance de l'environnement [20].

Software as a Service (SaaS)

Avec le SaaS, les utilisateurs ont accès à un logiciel complet directement depuis le Cloud. Ce modèle s'adresse principalement aux consommateurs moyens, qui n'ont plus à se soucier de l'installation ni de la maintenance du logiciel. Ils peuvent également être assurés que les performances du matériel sont suffisantes pour faire fonctionner le logiciel correctement. L'accès au logiciel, se fait via un navigateur Web, soit par un programme léger qui charge l'essentiel des fonctionnalités depuis le Cloud [20].

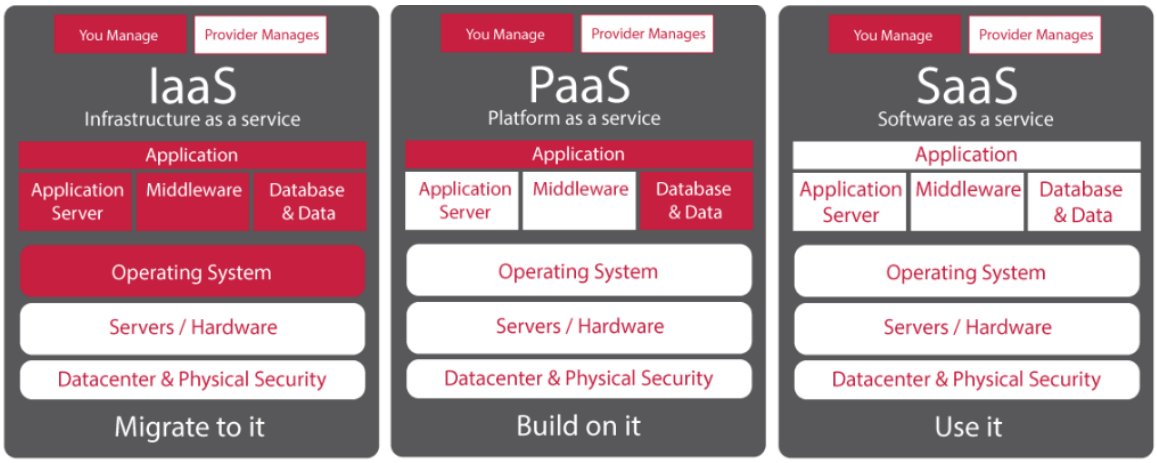

Fig. I.3 : Types de service Cloud Computing
[21]

I.3.5 Composants du Cloud Computing

L'architecture du cloud computing repose sur deux éléments clés : le frond-end (côté client) et le back-end (côté serveur). Le front-end, accessible aux utilisateurs, interagit avec le back-end via un réseau ou Internet. En revanche, le back-end, bien que caché, communique de manière sécurisée avec le client grâce à des protocoles établis, comme illustré dans la figure I.4.

Frond-end

Le front-end correspond au côté client dans l'architecture Cloud. Il regroupe tous les éléments avec lesquels l'utilisateur interagit directement. Il sert d'interface entre l'utilisateur et les ressources Cloud en permettant l'accès, la visualisation et la gestion des services à distance [21].

Back-end

Dans une infrastructure Cloud, le back-end joue un rôle essentiel en soutenant le front-end. Celui-ci s'appuie sur des serveurs distants, incluant des ressources matérielles et de stockage, entièrement supervisées par le fournisseur Cloud. Les composants essentiels d'une architecture Cloud back-end robuste sont [21] :

Application : L'interface utilisateur, essentielle dans l'architecture applicative, permet aux clients d'accéder aux données via des services back-end, tout en gérant leurs besoins et requêtes.

Service : Ce service permet la possibilité de réaliser l'ensemble des opérations disponibles dans le Cloud. Elle inclut notamment des fonctionnalités phares telles que la gestion et le développement d'applications, le stockage de données, ainsi que des services web, assurant une exécution optimale et performante des différentes tâches au sein d'un environnement Cloud.

Cloud Runtime : Le terme « Cloud Runtime » désigne le concept d'accessibilité immédiate des services, c'est un système d'exploitation pour le Cloud, qui utilise la virtualisation pour

donner accès à un réseau de serveurs. Chacun de ces serveurs fonctionne comme un espace de stockage indépendant.

Stockage : Le stockage Cloud désigne l'espace où sont conservées les données d'une application Cloud. Une partie dédiée du Cloud est généralement réservée à cette fonction.

Infrastructure et architecture : L'infrastructure Cloud regroupe tous les composants matériels virtualisés (CPU, cartes réseau, accélérateurs) qui sous-tendent les services logiciels. Son architecture dépend des besoins des utilisateurs et repose sur l'abstraction des ressources physiques pour assurer mobilité et élasticité.

Gestion : Il gène les ressources selon les besoins de chaque tâche. Si plusieurs activités concernent différents domaines de l'entreprise, il attribue à chacune des ressources précises et veille à leur bonne utilisation pour assurer le bon fonctionnement global. **Sécurité :** La structure permet un suivi régulier des processus de débogage. Les erreurs sont détectées et corrigées de façon continue, parfois plusieurs fois par jour.

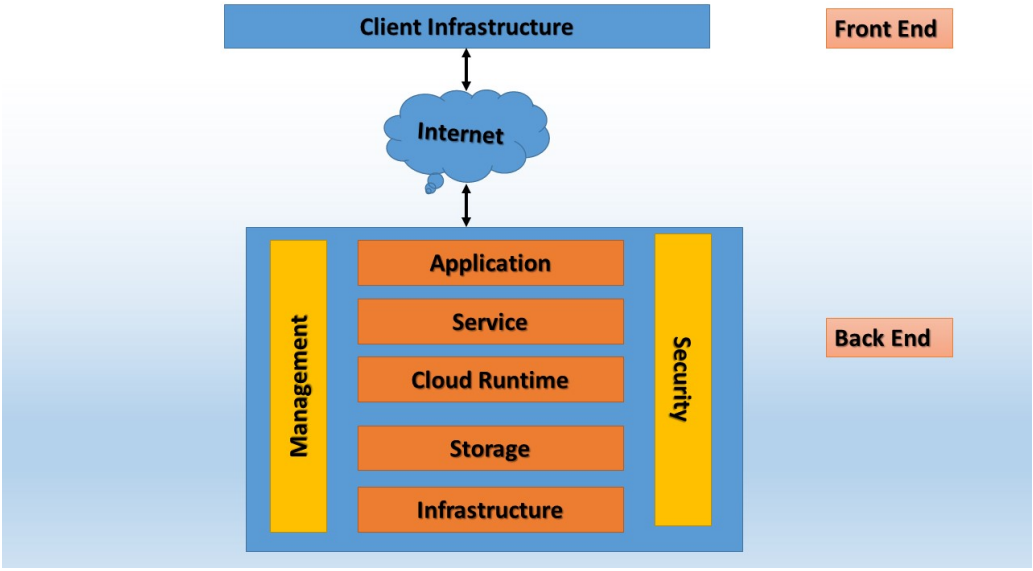

Fig. I.4 : Architecture du Cloud computing
[22]

I.4 Fog Computing

I.4.1 Définition du Fog Computing

Le Fog Computing est un modèle informatique distribué, proposé par Cisco en 2012, qui repose sur des nœuds intermédiaires situés entre les dispositifs connectés et le Cloud. Ces nœuds permettent un traitement local des données, ce qui réduit la latence, optimise l'utilisation du réseau et améliore la répartition des charges de calcul [23]. Contrairement à une infrastructure entièrement centralisée, le Fog Computing s'organise de manière hiérarchique. Ses nœuds peuvent être regroupés selon leur position géographique ou les besoins en latence des applications, ce qui garantit un traitement accéléré et une gestion optimisée des flux de données grâce à un équilibre entre ressources locales et serveurs distants [24]. Ce modèle renforce la sécurité

en limitant l'exposition des données sensibles vers le Cloud. Il améliore aussi l'interopérabilité entre des systèmes en simplifiant les échanges et en réduisant la dépendance exclusive aux infrastructures centralisées [23].

D'aprés Perrera et al. [25] « Le fog Computing est un scénario dans lequel un nombre considérable de dispositifs omniprésents et décentralisés hétérogènes (sans fil et parfois autonomes) communiquent et coopèrent potentiellement entre eux et avec le réseau pour effectuer des tâches de stockage et de traitement sans l'intervention de tiers. Ces tâches peuvent être destinées à prendre en charge des fonctions réseau de base ou de nouveaux services et applications s'exécutant dans un environnement en bac à sable. Les utilisateurs qui louent une partie de leurs appareils pour héberger ces services sont incités à le faire.»

I.4.2 Architecture du Fog Computing

Le Fog Computing repose sur une hiérarchie de traitement distribuée entre les dispositifs IoT, les nœuds intermédiaires et le Cloud. Cette structure permet de traiter une partie des données à proximité de leur source, ce qui améliore le temps de réponse et allège le trafic réseau [26]. La Figure I.5 présente cette architecture en trois couches complémentaires.

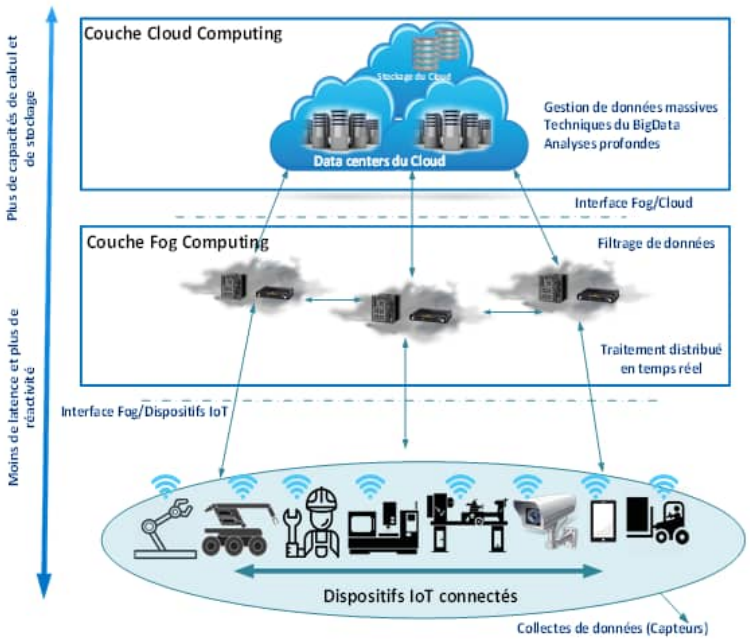

Fig. I.5 : Architecture du Fog Computing
[23]

Couche périphérique (Edge layer)

Elle est le premier niveau de l'architecture du Fog Computing. Elle regroupe les dispositifs connectés, tels que les capteurs, les actionneurs et les caméras intelligentes, responsables de la collecte des données en temps réel. Son rôle principal est d'effectuer un premier traitement des données en assurant la détection d'événements, le filtrage et la réduction du volume d'informations, ainsi que la prise de décisions locales rapides [27].

Couche Fog (Fog layer)

Elle représente l'intermédiaire entre la périphérie et le Cloud. Elle repose sur des équipements tels que les passerelles intelligentes, les routeurs avancés et les micro-datacenters. Son rôle est d'assurer le filtrage et l'agrégation des données issues de la couche périphérique, de permettre une analyse locale pour répondre aux besoins des applications sensibles à la latence et d'optimiser la bande passante en réduisant le volume d'informations envoyées vers le Cloud. Cette couche est particulièrement utile pour les systèmes nécessitant une prise de décision rapide, comme les applications de ville intelligente, les véhicules autonomes ou les dispositifs médicaux connectés [28].

Couche Cloud (Cloud Layer)

Elle est le niveau supérieur du Fog Computing. Elle repose sur des datacenters centralisés qui assurent le stockage massif et les traitements avancés des données. Son rôle est essentiel dans l'analyse approfondie des informations collectées, l'apprentissage automatique et le traitement intensif des données, ainsi que dans la gestion des historiques et la sauvegarde à long terme. En traitant uniquement les données essentielles transmises par les couches Edge et Fog, la couche Cloud réduit la charge de calcul et améliore l'efficacité globale du système [29].

I.4.3 Domaines d'applications du Fog Computing

Le Fog Computing s'impose comme une solution clé pour de nombreux domaines nécessitant rapidité, sécurité et traitement local des données. Ses principales applications sont :

Transports intelligents

Le Fog Computing permet d'analyser en temps réel les données des véhicules connectés, améliore la sécurité routière et optimise la fluidité du trafic. En faciliant la communication entre véhicules et infrastructures urbaines, il permet une gestion dynamique des routes et des feux de signalisation, réduisant les embouteillages et les accidents [30].

Villes intelligentes

Le Fog Computing est utilisé dans les infrastructures urbaines pour surveiller le trafic, réguler les feux de signalisation, optimiser l'éclairage public, suivre la pollution et détecter les incidents en temps réel. Il permet ainsi aux service municipaux d'intervenir plus rapidement [30].

Industrie

La maintenance prédictive constitue un des principaux bénéfices du Fog Computing. Les capteurs installés sur les machines industrielles analysent en continu les vibrations, température et d'autres indicateurs clé, détectant ainsi les anomalies avant qu'elles ne provoquent des pannes. Cette approche réduit considérablement les interruptions de production et optimise les coûts

de maintenance. De plus, elle renforce l'automatisation des processus grâce à une coordination intelligente entre les machines et les opérateurs [31].

Santé connectée

Les dispositifs médicaux intelligents exploitent le Fog Computing pour traiter localement les données des patients, permettre une surveillance continue et garantissant une réactivité immédiate en cas d'urgence. La télémédecine bénéficie également de cette approche en optimisant les échanges entre patients et professionnels de santé, tout en réduisant la charge sur les Datacenter [32].

Réalité augmentée et Réalité virtuelle (AR/VR)

Le Fog Computing améliore significativement les performances des applications AR/AV en réduisant la latence du rendu graphique. Cette avancée technologie est particulièrement cruciale pour les jeux vidéo, les formations immersives et les simulations industrielles complexes, ou une interaction en temps réel et une synchronisation fluide sont essentielles [33].

I.4.4 Avantages et défis du Fog Computing

Avantages du Fog Computing

1. **Réduction de la latence :** le Fog Computing traite les données au plus près de leur source, ce qui permet de réduire les délais de transmission par rapport au Cloud Computing. Cette caractéristique est essentielle pour les applications nécessitant une prise de décision rapide, comme les véhicules autonomes ou les dispositifs médicaux connectés [34].

2. **Optimisation de la bande passante :** en traitant et en filtrant les données localement avant de les envoyer vers le Cloud, le Fog Computing limite la surcharge du réseau et évite une consommation excessive de bande passante. Ce principe est particulièrement utile pour les environnements où une grande quantité de données est générée en permanence, comme dans les villes intelligentes ou les usines automatisées [34].

3. **Sécurité et confidentialité des données :** le traitement local des informations permet de réduire les risques liés aux cyberattaques et aux fuites de données lors des transmissions vers des serveurs distants. Cette approche est particulièrement importante dans des domaines sensibles, comme la santé, où la protection des informations personnelles est une priorité [34].

4. **Indépendance vis-à-vis de la connectivité Internet :** le Fog Computing peut fonctionner même en cas de défaillance de la connexion Internet, ce qui est essentiel pour les environnements isolés ou à connectivité intermittente, comme les mines, les plateformes pétrolières ou les zones rurales. Cela garantit une continuité des services même dans des conditions réseau difficiles [29].

5. **Scalabilité et flexibilité :** le Fog Computing permet une extension progressive des infrastructures en ajoutant des nœuds Fog selon les besoins, sans nécessiter de modifications majeures de l'architecture réseau. Cela le rend adapté aux environnements IoT où le nombre de dispositifs connectés augmente rapidement [35]

6. **Réduction des coûts opérationnels :** en traitant les données localement et en réduisant la quantité de données envoyées au Cloud, le Fog Computing diminue les coûts de stockage et de traitement, tout en optimisant l'utilisation des ressources locales [36].

Défis du Fog Computing

Le Fog Computing présente plusieurs défis liés à sa nature distribuée et à sa proximité avec des dispositifs en périphérie. Ces défis concernent à la fois les aspects techniques, organisationnels et économiques.

1. **Gestion des ressources distribuées :** la répartition géographique des nœuds Fog complique la gestion et l'allocation des ressources, nécessitant des mécanismes sophistiqués pour coordonner les tâches entre les nœuds périphériques et le Cloud [35].

2. **Sécurité des nœuds périphériques :** les nœuds Fog, souvent situés dans des environnements non contrôlés, sont vulnérables aux attaques physiques ou logicielles, ce qui exige des mécanismes de sécurité robustes pour protéger les données et les dispositifs [37].

3. **Complexité de la gestion des données :** Le Fog Computing nécessite des algorithmes sophistiqués pour décider quelles données traiter localement et lesquelles envoyer au Cloud, ce qui augmente la complexité de la gestion des données [36].

4. **Coûts de déploiement et de maintenance :** le déploiement et la maintenance des infrastructures Fog peuvent être coûteux, en particulier dans les environnements difficiles d'accès, comme les zones industrielles ou rurales, où les nœuds doivent être installés et maintenus régulièrement [38].

5. **Gestion de l'énergie :** les nœuds Fog alimentés par batterie nécessitent une gestion efficace de l'énergie pour prolonger leur durée de vie, ce qui est un défi dans les environnements où l'accès à l'énergie est limité, comme dans les zones reculées ou les installations industrielles éloignées [39].

6. **Latence déterministe pour les applications critiques** bien que le Fog Computing réduise la latence, garantir une latence déterministe (stable et prévisible) reste un défi pour certaines applications critiques, comme le contrôle de robots industriels, où des retards imprévisibles peuvent avoir des conséquences graves [40].

I.4.5 Comparaison entre le Cloud Computing et le Fog Computing

Pour mieux situer le Fog Computing par rapport au modèle classique du Cloud, le tableau I.1 présente une comparaison de leurs caractéristiques respective.

Critère	Cloud Computing	Fog Computing
Calcul et stockage	Analyses profondes, calcul et stockage centralisés	Calcul local, stockage et analyses décentralisés
Hétérogénéité	Interagit avec des sources de données et environnements hétérogènes	Supporte l'hétérogénéité des ressources, environnements et nœuds
Localisation	Indépendante de l'emplacement des utilisateurs	Proximité immédiate avec les utilisateurs et les appareils connectés
Architecture	Basée sur des serveurs centralisés	Distribuée,s'appuyant sur des nœuds intermédiaires
Mobilité	Peu adapté aux application mobiles	Supporte la mobilité
Latence	Élevée (quelques secondes à minutes) en raison de la distance des serveurs	Faible latence ($<$ 100 ms),optimisée par le traitement local
Bande passante	Limitée par la distance et le volume de données transférées	Optimisation de la bande passante
Disponibilité	Dépend de la connectivité Internet	Fonctionne même avec une connectivité Internet instable
Nombre de nœuds	Des dizaines/centaines de millions de serveurs dans le cloud	Des milliards de nœuds répartis géographiquement
Applications	Adapté aux applications tolérant une latence élevée	Idéal pour les applications en temps réel et sensibles aux délais
Accès aux services	Basé sur des centres de données distants via Internet	Déployé à la périphérie du réseau, proche des utilisateurs

Tab. I.1 : Comparaison entre le Cloud Computing et le Fog Computing [41]

I.5 Optimisation

I.5.1 Définition

D'après Chvatal [42], l'optimisation consiste en la recherche méthodique de la solution optimale au regard d'un critère prédéfini, au sein d'un ensemble de solutions réalisables. Cette démarche s'appuie sur des modèles mathématiques structurés visant soit à maximiser (comme un profit ou une performance), soit à minimiser (un coût ou un risque) une fonction objectif, tout en respectant un système de contraintes délimitant l'espace des solutions admissibles.

L'optimisation s'impose dans tous les domaines confrontés à des choix complexes, notam-

ment en ingénierie, en informatique ou en logistique. Elle suit une démarche structurée et logique qui se décompose en plusieurs étapes.

1. **Formuler le problème :** il s'agit d'identifier clairement le problème à résoudre. Cela implique de définir les objectifs, les variables impliquées et les contraintes à respecter.

2. **Modéliser le problème :** une fois le problème bien défini, il faut le représenter sous forme mathématique. On traduit la situation réelle en équations ou inégalités qui décrivent les relations entre les variables.

3. **Optimiser le problème :** Cette étape consiste à appliquer des méthodes algorithmiques ou numériques pour trouver la solution qui maximise ou minimise la fonction objectif selon les contraintes définies.

4. **Mettre en œuvre la solution :** La solution obtenue est ensuite testée dans la réalité. Si elle donne les résultats attendus, elle est adoptée. Sinon, le modèle est ajusté, et le processus recommence.

I.5.2 Problème d'optimisation

La résolution d'un problème d'optimisation combinatoire consiste à déterminer une solution optimale parmi l'ensemble des solutions réalisables. Pour y parvenir, on met en œuvre des algorithmes spécifiques visant soit à maximiser (dans le cas d'un problème de maximisation), soit à minimiser (pour un problème de minimisation) une ou plusieurs fonctions objectifs.

Ces algorithmes doivent tenir compte de contraintes qui limitent l'espace des solutions possibles en éliminant les configurations non valides.

Bien que la définition des problèmes d'optimisation soit généralement simple, leur résolution précise devient rapidement complexe, surtout pour les grands problèmes, nécessitant des ressources importantes en temps et mémoire [43].

On distingue principalement deux types de problèmes :

- Les problèmes dits **faciles** (classe P), qui peuvent être résolus en temps polynomial.

- Les problèmes dits **difficiles** (classe NP), pour lesquels aucune méthode connue ne permet une résolution efficace lorsque la taille augmente.

I.5.3 Notions de base en optimisation

Deux catégories fondamentales d'optimisation sont distingués : des problèmes de minimisation et des problèmes de maximisation. Un problème d'optimisation (de minimisation ou de maximisation) est défini par un ensemble de données et un ensemble de contraintes. Un ensemble de solutions S est associé au problème d'optimisation. Parmi les solutions S, un sous-ensemble $X \subseteq S$ représente des solutions réalisables respectant les contraintes C du problème, à chaque solution s est associée une valeur f(s) qui représente sa qualité. La résolution du problème d'optimisation consiste à trouver une solution $s^* \in X$ qui minimise ou maximise la valeur

f(s). Quelque soit le type du problème d'optimisation, ce dernier est défini par le 6-uplet (D, C,S, X, f, mode). Où D représente les données du problème, C les contraintes que doit satisfaire une solution afin d'être admissible, S l'ensemble des solutions possibles du problème traité, X un sous-ensemble de S représentant les solutions réalisables (admissibles), f une fonction du coût (aussi appelée fonction « objectif » ou fonction fitness) qui associe à chaque solution s une valeur numérique f(s) (nombre réel ou entier) représentant la qualité de s, mode indique le type du problème, il permet de savoir est ce qu'on doit minimiser ou maximiser les valeurs des solutions de X [43].

I.5.4 Méthodes de résolution

Pour résoudre des problèmes complexes, les chercheurs ont développé différentes méthodes d'optimisation. Ces méthodes évoluent constamment pour s'adapter à des défis toujours plus variés, améliorant à la fois la rapidité et la qualité des résultats, comme l'illustré la figure I.6.

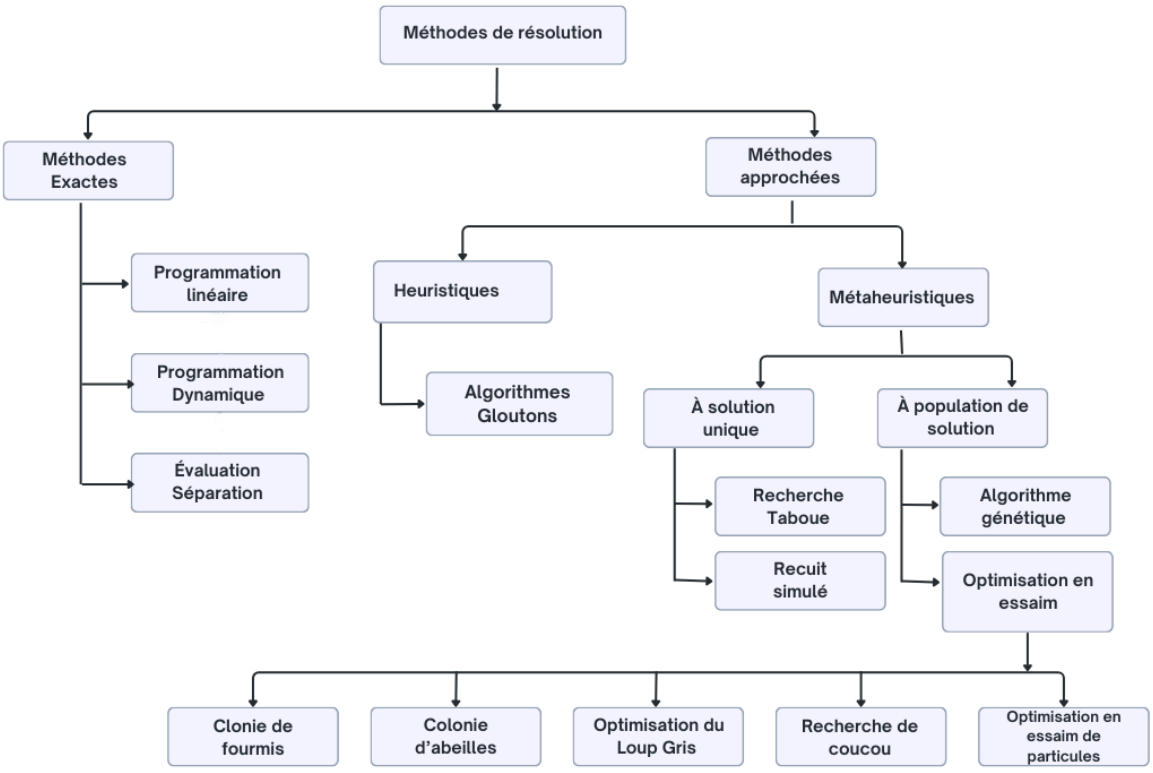

Fig. I.6 : Classification des méthodes de résolution [43]

Méthodes exactes

Les méthodes exactes garantissent la solution optimale en examinant toutes les possibilités, mais elle demandent beaucoup de temps de calcul et mémoire, surtout pour les problèmes complexes à nombreuses variables. Parmi les principales méthodes, on trouve l'algorithme du

simplexe (problèmes linéaires), la programmation dynamique (problèmes séquentiels), l'algorithme A* (recherche de chemins), les méthodes de séparation-évaluation (Branch and Bound, Branch and Cut). Leur coût en terme de temps de calcul qui peut être élevé limite souvent leur usage aux problèmes de taille modérée [43].

Méthodes approchées

les méthodes approchées représentent une famille d'algorithmes conçus pour résoudre des problèmes d'optimisation complexes en fournissant des solutions de qualité acceptable dans des délais raisonnables. Contrairement aux méthodes exactes qui garantissent mathématiquement la solution optimale mais deviennent rapidement impraticables pour des problèmes de grande échelle (en raison de leur complexité exponentielle), les méthodes approchées offrent un compromis délibéré entre la qualité de la solution et le temps de calcul. [44]. Les méthodes approchées sont classées en deux catégories : les heuristiques et les métaheuristiques :

Heuristiques

Une heuristique est une méthode approximative – un algorithme capable de fournir rapidement (en temps polynomial) une solution réalisable, mais pas nécessairement optimale, à un problème d'optimisation difficile. Contrairement aux approches générales, une heuristique est souvent spécifiquement conçue pour un problème donné, en exploitant ses particularités structurelles [43].

Selon [Feigenbaum et Feldman, 1963] [43] « Une méthode heuristique (ou simplement une heuristique) est une méthode qui aide à découvrir la solution d'un problème en faisant des conjectures plausibles mais faillible de ce qui est la meilleure chose à faire ».

Métaheuristique

Les métaheuristiques sont des techniques d'optimisation intelligentes, souvent inspirées par la nature. Plus évoluées que les méthodes heuristiques classiques, elles peuvent s'adapter à différents types de problèmes complexes. Leur objectif principal est de trouver la meilleure solution possible tout en évitant de rester bloquées sur des solutions "moins bonnes" mais faciles à trouver. Elles se divisent en deux catégories [43] :

- Les Méta-heuristiquement à base de solution unique :

  Ce sont des algorithmes d'optimisation qui débutent avec une solution initiale et l'améliorent progressivement en examinant systématiquement ses solutions voisines (appelées voisinage).

- Les méta-heuristiques à base de solutions multiples :

  Également appelées métaheuristiques populationnelles, ces algorithmes démarrent avec un ensemble de solutions initiales qu'ils font évoluer simultanément. Leur force réside dans une exploration plus large de l'espace de recherche, améliorant ainsi les chances de

trouver l'optimum global. Particulièrement efficaces pour les problèmes complexes, elles offrent une alternative puissante aux méthodes à solution unique.

I.6 Algorithmes d'optimisation

Dans cette section, nous présentons les deux algorithmes PSO (Particle Swarm Optimization) [45] [46] et ABC (Artifial Bee Colony) [47].

I.6.1 Algorithme PSO (Particle Swarm Optimization)

Principe général

L'Optimisation par Essaim de Particules est une méthode d'optimisation stochastique inspirée du comportement collectif observé dans la nature, notamment chez les oiseaux en vol ou les poissons en banc. Proposée en 1995 par Kennedy et Eberhart, elle simule les déplacements coordonnés d'un groupe de particules dans un espace de recherche multidimensionnel, où chaque particule représente une solution possible au problème traité .

Le PSO est particulièrement adapté aux problèmes complexes rencontrés en intelligence artificielle, ingénierie ou recherche opérationnelle, en raison de sa capacité à optimiser des fonctions non linéaires, non différentiables ou discontinues.

Le fonctionnement repose sur trois mécanismes fondamentaux :

Mémoire individuelle : chaque particule mémorise la meilleure solution qu'elle a atteinte, appelée pBest (personal best)

Interaction social : l'ensemble des particules partage cette information pour identifier la meilleure solution globale, appelée gBest (global best)

Mouvement dynamique : chaque particule ajuste sa trajectoire en combinant :

- Une composante d'inertie (conservation de la vitesse précédente)
- Une composante congntive (retour vers pBest)
- Une composante sociale (attraction vers gBest)

Ces trois forces influencent le déplacement de chaque particule dans l'espace de recherche, comme illustré dans la figure I.7. **Equation de mise à jour :** le comportement de chaque particule est défini par deux équations principales : [48]

Vitesse :

$$v_i(t+1) = \omega \cdot v_i(t) + c_1 \cdot r_1 \cdot (pBest_i - x_i(t)) + c_2 \cdot r_2 \cdot (gBest - x_i(t)) \tag{I.1}$$

Position :

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{I.2}$$

Où :

- $v_i(t)$: la vitesse de la particule i à l'instant t

- $x_i(t)$: la position de la particule i à l'instant t
- $pBest_i$: meilleure position personnelle de la particule i
- $gBest_i$: meilleure position globale atteinte par l'essaim
- ω : coefficient d'inertie (influence du mouvement précédent)
- c_1, c_2 : coefficients d'accélération (facteurs cognitifs et sociaux)
- r_1, r_2 : variables aléatoires $\in [0, 1]$ qui introduisent de la diversité

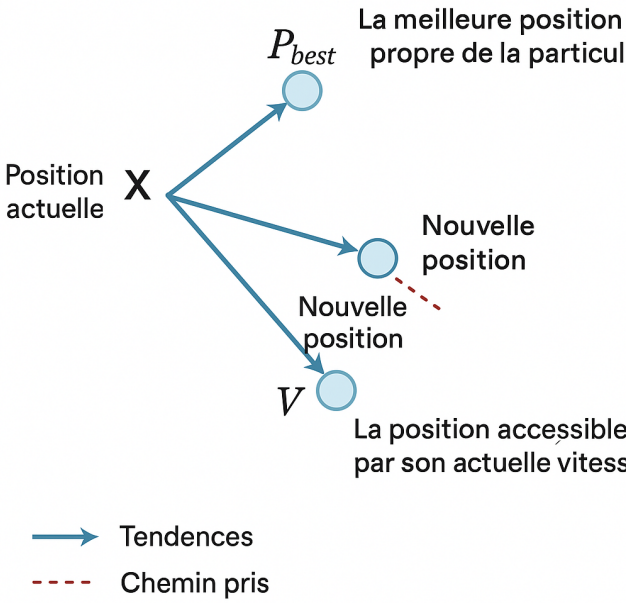

FIG. I.7 : Illustration du déplacement d'une particule dans l'algorithme PSO [48]

Ces paramètres ont des rôles bien définis. Le coefficient d'inertie ω contrôle l'exploration, tandis que c_1 et c_2 régulent l'équilibre entre recherche individuelle et coopération collective. Les variables aléatoires r_1 et r_2 assurent la variabilité du comportement des particules, évitant ainsi la stagnation dans des optima locaux.

Grâce à cette dynamique, le PSO explore efficacement l'espace des solutions tout en convergeant progressivement vers des solutions optimales. Sa simplicité de mise en œuvre, sa robustesse et sa capacité d'adaptation en font un outil puissant pour une large gamme d'applications, allant de l'optimisation de paramètres en apprentissage automatique à la conception de systèmes complexes, en passant par la planification et la finance.

Les étapes de l'Algorithme PSO

Initialisation de l'essaim :

On commence par générer une population de N particules. Pour chaque particule i, on définit aléatoirement :

- une position initiale $x_i(0)$ dans l'espace de recherche
- une vitesse initiale $v_i(0)$ fixée dans un intervalle autorisé

Chaque particule connaît également :

- sa meilleure position personnelle $pBest_i = x_i(0)$
- la meilleure position globale $gBest$, définie comme la meilleure position parmi toutes les particules initiales

Evaluation de la fonction d'aptitude (fitness) :

À chaque itération t, on évalue la fonction objective f à la position actuelle de chaque particule i :

$$f(x_i(t)) \Rightarrow \text{valeur de la fitness} \tag{I.3}$$

Cette évaluation permet de mesurer la qualité de la solution représentée par la particule. La fonction peut intégrer plusieurs critères selon le problème.

Mise à jour de la meilleure position personnelle (pBest) :

Après l'évaluation, chaque particule compare sa solution actuelle à la meilleure qu'elle a mémorisée :

$$f(x_i(t)) < f(pBest_i) \Rightarrow pBest_i = x_i(t) \tag{I.4}$$

Cette mise à jour permet à la particule de conserver les zones prometteuses déjà explorées. Sinon, la mémoire $pBest_i$ reste inchangée.

Mise à jour de la meilleure position globale (gBest) :

Parmi toutes les particules, on identifie celle qui a obtenu la meilleure valeur de fitness, et on met à jour la solution globale partagée :

$$gBest = \arg\min_i f(pBest_i) \tag{I.5}$$

Cette position sert de référence commune à tout l'essaim et oriente le déplacement des particules vers les zones les plus prometteuses de l'espace de recherche.

Mise à jour des vitesses et de la position :

Une fois la meilleure position personnelle ($pBest$) et la meilleure position globale ($gBest$) mises à jour, chaque particule ajuste sa vitesse en combinant trois composantes : l'inertie, l'attraction vers sa propre meilleure position et l'attraction vers la meilleure solution globale.

La vitesse ainsi obtenue oriente la direction du déplacement de la particule dans l'espace de recherche. À partir de cette vitesse mise à jour, la nouvelle position de la particule est calculée.

Ces ajustements permettent à chaque particule d'explorer de nouvelles zones tout en se rapprochant progressivement des régions optimales déjà identifiées par l'essaim.

Critère d'arrêt :

L'algorithme PSO s'exécute de manière itérative jusqu'à la satisfaction d'une condition d'arrêt. Les critères les plus couramment utilisés sont :

- nombre maximal d'itérations atteint, fixé au préalable

- stabilisation des solutions : les positions des particules ou la valeur de la fonction objectif ne varient plus de manière significative
- seuil de performance : la valeur de la fonction d'aptitude atteint un niveau jugé satisfaisant pour le problème étudié

Lorsque l'un de ces critères est rempli, l'algorithme s'interrompt et retourne la meilleure solution globale trouvée.

I.7 L'algorithme ABC (Artifial Bee Colony)

Une méthaheuristique d'optimisation inspirée du comportement collectif des abeilles mellifères lors de la recherche de nourriture. Proposé par D. Karaboğa en 2005. Elle appartient à la famille des algorithme à intelligence en essaim (Swarm Intelligence). L'ABC repose sur la coopération entre trois types d'agents artificiels, les abeilles employées, observatrices et éclaireuses, qui explorent l'espace de recherche pour identifier les solutions optimales ou quasi-optimales à un problème donné .

I.8 Fonctionnement de l'algorithme ABC

- Une abeille initiale (Beelnit) sélectionne un sommet de départ.
- à chaque sommet de cette zone, une abeille est assignée pour explorer localement ses voisins (phase d'intensification).
- Chaque abeille identifie la meilleure solution locale et la communique à l'essaim via une "danse" symbolique.
- Le sommet associé à la meilleure solution globale devient le nouveau sommet de départ, et une nouvelle zone de recherche est générée.
- Ce processus est répété pendant un nombre fixe d'itérations ou jusqu'à un critère d'arrêt.

I.8.1 Étapes de l'Algorithme ABC

L'algorithme Artificial Bee Colony (ABC) se décompose en trois phases clés reproduisant le comportement collectif des abeilles lors de la recherche de nourriture. Voici ses étapes fondamentales[49] :

1.Initialisation :

Générer aléatoirement des solutions initiales (sources de nourriture) :

$$x_{i,j} = x_{\min,j} + \text{rand}(0,1) \cdot (x_{\max,j} - x_{\min,j}) \tag{I.6}$$

où :

- $x_{i,j}$ est la j-ème composante de la i-ème solution,
- $x_{\min,j}$ et $x_{\max,j}$ sont les bornes inférieure et supérieure du problème,
- rand$(0,1)$ est un nombre aléatoire uniformément distribué entre 0 et 1.

2.Phase des Abeilles Employées

Pour chaque solution x_i, générer une nouvelle solution candidate v_i :

$$v_{i,j} = x_{i,j} + \phi_{i,j} \cdot (x_{i,j} - x_{k,j}) \tag{I.7}$$

où :

- $\phi_{i,j}$ est un nombre aléatoire uniformément distribué dans $[-1,1]$,
- x_k est une solution voisine choisie aléatoirement $(k \neq i)$,
- j représente la j-ème composante de la solution.

Étapes :

- Évaluer la qualité de v_i via la fonction objectif $f(v_i)$.
- Appliquer une sélection gourmande (méthode du meilleur) :

$$x_i \leftarrow \begin{cases} v_i & \text{si } f(v_i) \leq f(x_i) \quad \text{(minimisation)}, \\ x_i & \text{sinon.} \end{cases}$$

3.Phase des Abeilles Observatrices

Les abeilles observatrices sélectionnent une solution x_i avec une probabilité p_i proportionnelle à sa fitness :

$$p_i = \frac{\text{fit}_i}{\sum_{j=1}^{SN} \text{fit}_j} \tag{I.8}$$

où :

- fit_i représente la fitness de la solution x_i
- SN est le nombre total de sources de nourriture (solutions)

Calcul de la fitness

Pour un problème de minimisation, la fitness peut être définie par :

$$\text{fit}_i = \begin{cases} \frac{1}{1+f(x_i)} & \text{si } f(x_i) \geq 0 \\ 1 + |f(x_i)| & \text{si } f(x_i) < 0 \end{cases} \tag{I.9}$$

Mise à jour des solutions

Pour chaque solution sélectionnée :

- Générer une nouvelle solution candidate v_i comme dans la phase des abeilles employées :

$$v_{i,j} = x_{i,j} + \phi_{i,j} \cdot (x_{i,j} - x_{k,j}) \tag{I.10}$$

- Appliquer la sélection gourmande entre x_i et v_i

4.Phase des Abeilles Éclaireuses

Une solution x_i est abandonnée si elle n'est pas améliorée après un nombre maximal d'essais (`limit`) :

$$\text{Si compteur\_échecs}_i \geq \text{limit} \Rightarrow \text{Abandonner } x_i \tag{I.11}$$

Initialisation d'une nouvelle solution

La solution abandonnée est remplacée par une nouvelle solution aléatoire :

$$x_{i,j} = x_{\min,j} + \text{rand}(0,1) \cdot (x_{\max,j} - x_{\min,j}) \tag{I.12}$$

où :

- $x_{\min,j}$ et $x_{\max,j}$ sont les bornes du problème pour la dimension j
- $\text{rand}(0,1)$ est un nombre aléatoire uniforme dans $[0,1]$

Critères d'arrêt

L'algorithme s'arrête lorsque :

- Le nombre maximal d'itérations (`max_iter`) est atteint
- La convergence est satisfaisante ($|f(x_{\text{best}}) - f_{\text{target}}| < \epsilon$)

I.9 Conclusion

Ce chapitre a présenté les concepts clés de l'Internet des objets (IoT), du Cloud et du Fog Computing, soulignant les limites du Cloud (latence, bande passante) face aux besoins des applications IoT. Le Fog Computing émerge comme une solution optimale, combinant proximité, efficacité énergétique et réactivité pour les systèmes critiques. L'optimisation permet ainsi de trouver le meilleur équilibre entre performance et coût pour ces systèmes. Ces bases théoriques justifient l'étude du Task Offloading dans le chapitre suivant.

Chapitre II

Task Offloading dans le Fog Computing

II.1 Introduction

L'émergence d'objets connectés et de services intelligents dans des domaines variés a engendré une croissance exponentielle du volume de données générées à la périphérie du réseau. Face à cette explosion de données et à la nécessité de traitements rapides, le paradigme du Fog Computing s'impose comme une alternative pertinente au Cloud Computing traditionnel.

Dans ce contexte, le Task Offloading devient une opération clé pour optimiser l'utilisation des ressources disponibles, améliorer les performances des applications, et répondre aux contraintes strictes en matière de temps réel et d'énergie. Il s'agit de décider quelles tâches doivent être exécutées localement (sur l'appareil utilisateur), et lesquelles peuvent être transférées vers les nœuds fog ou vers le Cloud.

Ce chapitre est consacré à l'étude approfondie du Task Offloading dans un environnement de Fog Computing. Il présente les principes de base du Task Offloading, les critères influençant les décisions d'offloading, ainsi que les différentes approches, techniques et algorithmes proposés dans la littérature.

II.2 Concepts fondamentaux du Task Offloading

II.2.1 Définition et principes généraux

Le Task Offloading consiste à transférer le traitement de tâches depuis des dispositifs IoT à ressources limitées, vers des ressources de calcul plus puissantes, comme les nœuds Fog ou les server Cloud. Cette technique vise à optimiser les performances globales du système en exploitant les capacités de calcul là où elles sont les plus adaptées.

Le Fog Computing repose sur une architecture hiérarchique où les données sont d'abord traitées localement par les nœuds Fog. Si nécessaire, ils peuvent ensuite être transmis vers le Cloud pour un traitement supplémentaire. Cette organisation facilite le Task Offloading en répartissant intelligemment des charges de calcul selon leur urgence et leur complexité [**?**].

Dans ce contexte, le Task Offloading est perçu comme un problème d'optimisation multi-objectif, visant à équilibrer la latence, la consommation d'énergie et la charge réseau, tout en garantissant un niveau de performance conforme aux exigences des applications concernées.

II.2.2 Acteurs du Task Offloading

Le Task Offloading s'appuie sur la collaboration de trois catégories d'acteurs, intégrés dans une architecture distribuée typique IoT-Fog-Cloud. Comme l'illustre la figure II.1 [50].

Dispositifs IoT : Il s'agit d'objets connectés (capteur, actionneur, caméra, smartphone, etc.) qui génèrent les données et lancent les tâches à exécuter. Ces dispositifs ont des ressources limitées en puissance, mémoire et batterie. Ils ne peuvent pas toujours exécuter localement des traitements complexes. Leur enjeu est de déléguer intelligemment le traitement pour économiser l'énergie, réduire le temps de réponse et garantir la continuité du service.

Nœuds Fog : Ce sont des dispositifs intermédiaires situés à la périphérique du réseau (routeurs, stations de base, passerelles, etc.), proches des dispositifs IoT. Ils offrent une capacité

de traitement et de stockage local, au plus près des sources de données. Plus puissants que les capteurs, ils restent limités comparés aux serveurs Cloud. Leur rôle est de traiter localement les tâches déchargées, avec une fiable latence. En cas de surcharge, un nœud peut coopérer avec d'autres ou transférer la charge vers le Cloud. Selon Bonomi [27], cette flexibilité réduit les délais de réponse et diminue la charge du Cloud.

Serveurs Cloud : Ils constituent la couche de traitement centrale, situés dans des centres de données distants. Ils offrent une capacité élevée de calcul et de stockage. Ils prennent en charge les tâches complexes ou moins urgentes. Leur principal inconvénient est la latence causée par la distance réseau, Cela peut poser problème pour les applications temps réel. Malgré cela, le Cloud reste essentiel dans une stratégie globale de Task Offloading. Il permet un traitement massif, un stockage à long terme et compense les limites des couches IoT et Fog.

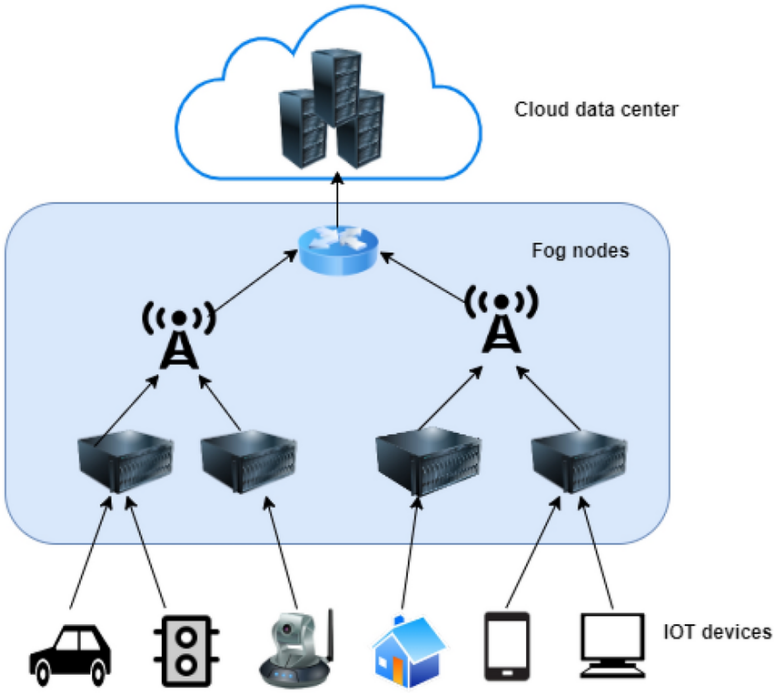

Fig. II.1 : Acteurs du Task Offloading
[51]

II.2.3 Métriques d'évaluation

Dans un environnement Fog Computing, le Task Offloading est souvent modélisé comme un problème d'optimisation multi-objectifs. Plusieurs critères sont pris en compte en même temps pour garantir une performance adaptée aux exigences spécifique de chaque application. Ces métriques servent à évaluer l'efficacité d'une stratégie d'Offloading selon le contexte. Les principales métriques sont [52] :

Latence totale : Elle désigne le temps nécessaire à l'exécution d'une tâche, depuis sa soumission jusqu'à la réception du résultat. Elle prend en compte le temps de traitement sur le nœud, le temps de transmission des données, et le temps d'attente si la tâche est mise en file.

Cette métrique est essentielle pour les applications sensibles au temps réel. Une latence trop élevée peut dégrader la performance perçue. Dans certains cas, comme la réalité augmentée

ou les véhicules autonomes, une latence <20ms est souvent indispensable pour garantir un fonctionnement fluide et sécurisé.

Consommation d'énergie : Elle représente l'énergie dépensée par le dispositif pour exécuter une tâche ou la transférer vers un autre nœud. Dans un environnement Fog, ce critère est crucial, surtout pour les dispositifs IoT alimentés par batterie. L'objectif du Task Offloading est souvent de réduire cette consommation pour prolonger l'autonomie des appareils sans compromettre la qualité du service.

Deux sources principales de consommation :

- **Exécution locale :** l'exécution d'une tâche directement sur le dispositif sollicite son processeur, et consomme de l'énergie en fonction de la charge de calcul et de la durée du traitement.
- **Transmission :** envoyer une tâche ou des données vers un nœud Fog ou Cloud implique une consommation d'énergie liée à l'utilisation du réseau, en particulier les interfaces radio pour les objets mobiles ou sans fil.

Puissance de calcul : Elle mesure la capacité d'un dispositif à traiter des tâches selon ses ressources disponibles, comme le processeur, la mémoire ou la bande passante. Dans un environnement avec des équipements de capacités différentes, comme le Fog Computing, tous les nœuds n'offrent pas les mêmes performances. Un capteur possède très peu de ressources, tandis qu'un serveur Cloud est beaucoup plus puissant. Le Task Offloading permet d'exploiter cette diversité en confiant les tâches complexes aux nœuds les plus performants. Une stratégie efficace répartit les traitements en fonction de la capacité de chaque nœud pour éviter les surcharges et améliorer la performance globale.

Qualité de service (QoS) : Elle représente la performance perçue par l'utilisateur. Elle dépend de plusieurs facteurs comme la disponibilité, de la stabilité, le débit, le temps de réponse et de la fiabilité du service. Dans un environnement Fog computing, la qualité de service est essentielle, surtout pour les applications critiques ou sensibles au délia. Une bonne stratégie d'offloading doit maintenir une qualité de service constante, même en cas de surcharge des nœuds ou de perturbation réseau. La continuité du service et la satisfaction de l'utilisateur reposent directement sur ce critère.

II.3 Classification du Task Offloading

Task Offloading constitue un mécanisme central dans les environnements Fog Computing. Il vise à améliorer les performances des systèmes IoT en transférant les charges de calcul vers des nœuds Fog ou Cloud, selon la nature des tâches et des contraintes du système.

II.3.1 Classification selon le niveau de transfert

Offloading total (Full Offloading)

Consiste à transférer l'intégrité d'une tâche depuis un dispositif IoT vers un nœud plus puissant. Ce modèle convient aux appareils ayant du capacités de calcul limitées, comme ceux

utilisés dans des applications intensives telles que l'analyse vidéo ou les transports intelligents. Il permet de libérer complètement le dispositif local de la charge de traitement, améliorant ainsi son autonomie énergétique. Cependant, ce modèle exige une gestion efficace des ressources du système Fog Computing, car un trop grand nombre de tâches peut saturer les nœuds intermédiaires si l'architecture n'est pas bien optimisée. Des études récentes ont montré que ce modèle permet d'améliorer à la fois la consommation d'énergie et la qualité du service, surtout dans des environnements denses en dispositifs [53].

Offloading partiel (Partial Offloading)

Consiste à répartir l'exécution d'une tâche entre le dispositif local et un ou plusieurs nœuds distants. Une partie est traitée localement, tandis que le reste est transféré vers le Fog ou le Cloud. Ce modèle optimise l'usage des ressources, limite les transferts de données et réduit la consommation d'énergie. Il améliore aussi la réactivité du système, surtout lorsque certaines sous-tâches exigent beaucoup de ressources, et que d'autres peuvent être exécutées localement avec peu de latence. Cette flexibilité permet d'adapter la répartition selon les contraintes réseau, les besoins du traitements ou la politique de sécurité. Des études récentes montrent qu'il réduit la charge réseau et équilibre la consommation d'énergie, tout en maintenant de bonnes performances dans les environnements Fog-Cloud [53].

II.3.2 Classification selon le mode de prise de décision

Offloading statique

Applique des règles fixes définies à l'avance. La décision de transfert ne tient pas compte de l'état actuel du réseau ou des ressources disponibles. Par exemple, une tâche complexe est toujours envoyée vers le Cloud, quelle que soit la charge du système. Ce modèle est simple à mettre en œuvre et peu coûteux en calcul. Il est adapté aux environnements prévisibles où les charges sont constantes. En revanche, dans des contextes dynamiques comme les systèmes IoT urbain, il peut entraîner une mauvaise allocation des ressources et une latence plus élevée. Ce modèle a surtout été utilisé dans les premières recherches sur le Fog Computing [54].

Offloading dynamique

Il repose sur une prise de décision en temps réel. Il tient compte de l'état des ressources locales et distantes, de la charge réseau, de la latence, de la taille des tâches ou des délais exigés. L'objectif est d'adapter la stratégie d'offloading pour optimiser la qualité de service. Par exemple, une tâche peut être traitée localement si le réseau est saturé, puis déléguée vers un nœud Fog dès que les conditions s'améliorent. Ce modèle améliore la réactivité et la gestion de la charge. En contrepartie, il demande plus de calculs pour évaluer en continu plusieurs paramètres. Des études récentes montrent qu'il réduit efficacement la latence dans les architectures Fog-Cloud, même en cas de forte variabilité des ressources [55].

Offloading stochastique

Il introduit l'incertitude dans la prise de décision. Il s'appuie sur des modèles probabilistes pour anticiper les variations de la charge du réseau, la disponibilité des nœuds ou la latence. Au lieu de suivre des règles fixes ou un état instantané, il évalue la probabilité qu'une décision améliore les performances. Ce modèle convient aux environnements dynamiques, comme les réseaux Fog distribués. Il est utile lorsque les informations sont incomplètes ou incertaines. Il permet des décisions plus robustes, mais exige une modélisation précise des incertitudes et in suivi continu. Des études récentes montrent qu'il améliore la stabilité et l'efficacité du l'offloading dans des contextes à forte variabilité de ressources [56].

Offloading hybrid

La méthode hybride a été proposée par Canepa, Lee et al [57]. Une partie de la décision est prise à partir de spécifications définies par le programmeur et d'outils d'analyse statique, tandis que l'autre partie est déterminée en temps réel. L'objectif de cette approches dynamique et statique était de minimiser les effets secondaires du profilage et les temps d'attente (la surcharge). Cependant, cette solution ne garantit pas toujours de bonnes performances, car le temps d'exécution sur un appareil mobile est souvent court.

II.4 Architectures décisionnelles dans le Fog Computing

Dans le Fog Computing, la stratégie et l'architecture décisionnelle jouent un rôle essentiel dans le processus de Task Offloading. Il ne s'agit pas seulement de décider quoi offloader, mais aussi de déterminer qui prend la décision et comment les ressources sont coordonnées. L'efficacité du système dépend directement de cette organisation. On distingue principalement trois approches [58], comme le montre la Figure II.2, l'approche centralisée du Cloud Computing se différencie nettement de l'approche distribuée du Fog Computing, notamment en termes de localisation des décisions et de gestion des ressources.

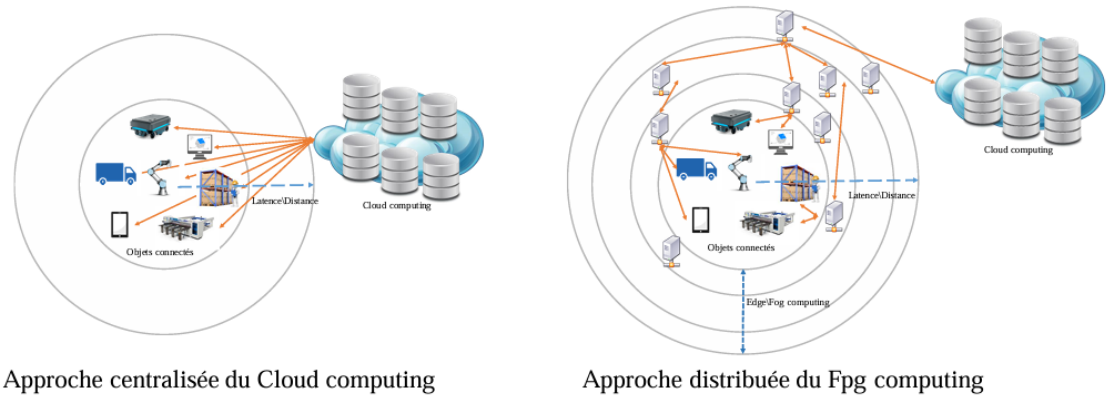

Fig. II.2 : Comparaison entre l'approche centralisée du Cloud Computing et l'approche distribuée du Fog Computing
[23]

II.4.1 Approche centralisée

Dans l'approche centralisée, un seul nœud, souvent appelé serveur Fog maître ou contrôleur, prend en charge toutes les décisions d'Offloading. Il collecte des informations sur l'état du réseau, la charge des ressources et les tâches en attente. À partir de ces données, il décide où et comment répartir les traitements. Ce modèle permet une coordination optimale et une vue d'ensemble du système, ce qui réduit les conflits entre nœuds et assure une répartition rationnelle des tâches. Cependant, ce type d'organisation présente un inconvénient majeur : un point unique de défaillance. Si le nœud central est surchargé ou tombe en panne, tout le système qui peut en être affecté. De plus, la centralisation des décisions peut introduire de la latence, surtout dans les environnements distribués, à grand échelle. Cette approche a été utilisé dans plusieurs travaux de recherches, comme celui de Bonnie et al [27].

II.4.2 Approche décentralisée et distribuée

Dans l'approche décentralisée, chaque nœud Fog ou terminal IoT prend ses décisions localement, sans coordination centrale. Le dispositif analyse son état, la latence, la charge et les ressources disponibles à proximité, puis décide d'exécuter ou d'offloader une tâche.

Ce modèle réduit la latence, renforce la résilience du système et s'adapte bien aux environnements Fog dynamiques, notamment en présence de ressources mobiles, instables ou intermittentes. Cependant, cette autonomie peut provoquer des conflits. Plusieurs nœuds peuvent tenter d'utiliser simultanément la même ressource, causant une surcharge. Pour éviter cela, des mécanismes de régulation, de négociation ou de coopération sont introduits. C'est à ce moment qu'on parle d'approche distribuée.

- La décentralisation concerne la prise de décisions locales par chaque nœud.
- La distribution concerne la répartition physique des ressources et des traitements dans le système.

Une approche décentralisée et distribuée combine les deux : Les nœuds prennent des décisions locales tout en échangeant des informations entre eux pour équilibrer la charge. Ce modèle horizontal supprime les points de défaillance uniques, augmente la tolérance aux pannes et améliore la fluidité du système. Il est particulièrement adapté aux architectures Fog à grande échelle, dans des contextes industriels ou urbains où les objets connectés sont nombreux, mobiles et fortement répartis [23].

II.4.3 Approche hiérarchique

elle combine les avantages des deux modèles précédents. Elle répartit les décisions d'Offloading sur plusieurs niveaux. Les décisions simples et rapides sont prises localement par les terminaux, tandis que les tâches plus complexes ou sensibles sont remontées vers un niveau supérieur, comme un nœud Fog ou le Cloud. Cette organisation permet d'optimiser l'allocation des ressources tout en assurant une bonne réactivité. Elle convient particulièrement aux architectures IoT–Fog–Cloud à plusieurs couches. Les traitements urgents sont gérés à proximité de

la source, tandis que et les traitements lourds sont transférés plus haut dans la hiérarchie. Ce modèle améliore la flexibilité, la performance et la résilience du système.

II.5 Critères de décision pour l'Offloading

Le mécanisme de Task Offloading repose sur des décisions dynamiques, influencées par le type de tâche, l'état du réseau, la disponibilité des ressources et les exigences de sécurité. Une stratégie de décision bien conçue permet une répartition efficace de la charge entre les dispositifs IoT, les nœuds Fog et le Cloud [52].

Priorité des tâches Les tâches sont classées selon leur niveau d'urgence et leur importance . Les applications sensibles, comme les alertes médicales ou la sécurité industrielle, exigent une latence très faible. Elles doivent être exécutées localement ou transférées vers un nœud Fog. En revanche, les tâches moins urgentes, telles que les mises à jour de données ou les analyses, peuvent être confiées au Cloud. Cela souligne l'importance de la priorité des tâches pour optimiser les système IoT.

Ressources disponibles La décision d'Offloading dépend de l'état actuel des ressources locales et distantes. Lorsque le processeur, la mémoire ou la bande passante d'un dispositif IoT est limité ou saturé, il devient nécessaire de transférer la tâche vers un nœud Fog . Une surveillance continue des nœuds Fog permet d'adapter le comportement du système aux variations de charge et d'optimiser la qualité de service. Des recherches ont montré que l'évaluation dynamique des ressources est crucial pour une allocation efficace des tâches.

Consommation d'énergie L'énergie disponible, particulièrement pour les dispositifs IoT alimentés par batterie. Si le traitement local consomme trop d'énergie, il peut être préférable de transférer la tâche afin de préserver l'autonomie du dispositif. Dans ce cas, l'Offloading vers des nœuds Fog ou Cloud devient une solution efficace pour prolonger la durée de fonctionnement sans recharge.

Contraintes de sécurité et de confidentialité Le traitement des données sensibles, comme les informations médicales ou les données personnelles, doit respecter les réglementations en vigueur. Le Règlement Général sur la Protection des Données (RGPD) impose des obligations strictes concernant la collecte, le traitement et le transfert de ce type d'information. De plus la norme ISO/IEC 27001, est un standard international, définit les exigences pour mettre en place un système de gestion de la sécurité de l'information. Lorsque le transfert des données présente un risque, l'exécution doit être locale ou limitée à des serveurs certifiés.

Choix de la destination

- Le Fog est recommandé pour les tâches temps réel ou sensibles à la latence en raison de sa proximité avec les dispositifs. Il permet une réactivité plus élevée.
- Le Cloud est plus adapté aux traitements complexes à forte intensité de calcul ou non sensibles aux délais, grâce à sa puissance de calcul et sa capacité de stockage élevée.

II.6 Algorithmes de Task Offloading dans le Fog Computing

Le Fog Computing repose sur des mécanismes intelligents de répartition des tâches pour optimiser les performances, réduire la latence et économiser l'énergie. Nous classons ces algorithmes en quatre grandes catégories, chacune adaptée à des besoins spécifiques comme illustré dans la figure II.3 :

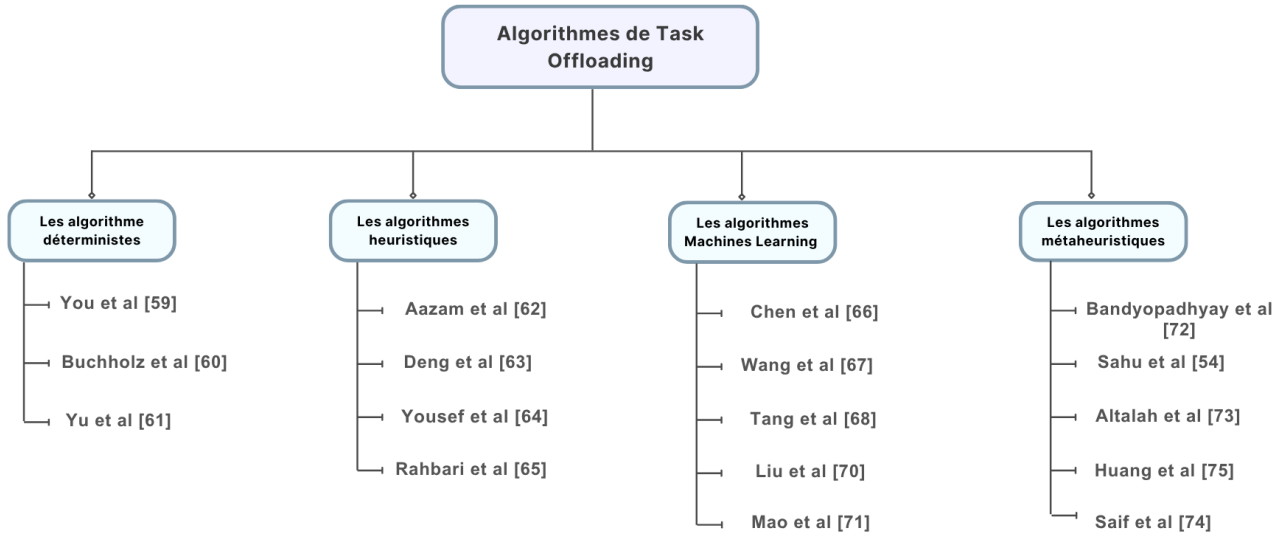

Fig. II.3 : Classification des algorithmes de Task Offloading

II.6.1 Algorithmes déterministes

Les algorithmes déterministes sont couramment utilisés dans le Fog Computing pour automatiser le transfert des tâches. Leur comportement est fixe et prévisible, ce qui permet de suivre des règles prédéfinies selon des critères comme la taille de la tâche, la consommation d'énergie, la qualité de la connexion ou la distance au nœud cible. Dans des conditions identiques, ils donnent toujours le même résultat.

You et al. [59] ont formalisé l'un des premiers modèles d'algorithmes déterministes dans le cadre du Cloud mobile. Leur approche repose sur l'utilisation de seuils. Ils ont conçu un algorithme qui transfère les tâches uniquement lorsque des limites, telles que la charge CPU ou la latence, sont dépassées. Les résultats obtenus ont montré une réduction significative de la consommation d'énergie tout en maintenant une qualité de service acceptable.

Buchholz et al. [60], ont développé un algorithme déterministe nommé Priority and Energy-based Resource allocation. Leur méthode associe les tâches à des nœuds de calcul selon trois

critères fixes : la priorité de la tâche, la capacité de traitement du nœud et sa consommation d'énergie. Les tâches urgentes sont envoyées vers les nœuds les plus puissants et les plus stables, tandis que les tâches secondaires sont orientées vers des ressources disponibles. Cette stratégie a permis d'améliorer l'efficacité d'énergie tout en réduisant les délais de traitement.

Yu et al. [61] ont proposé un algorithme déterministe qui coordonne l'allocation des ressources radio (sous-porteuses) et des ressources de calcul (temps CPU) dans un système de Mobile Edge Computing. Leur approche vise à optimiser simultanément l'utilisation des ressources de communication et de calcul, ce qui permet de réduire la consommation d'énergie et d'améliorer l'efficacité du système.

Les algorithmes déterministes sont simples et peu exigeants en ressources. Ils conviennent à des environnements stables et bien définis. Cependant, leur rigidité et leur manque de réactivité aux changements en temps réel les rendent moins adaptés aux contextes mobiles ou imprévisibles. Ces limites ont conduit au développement de méthodes plus flexibles, comme les algorithmes heuristiques ou intelligents.

II.6.2 Algorithmes heuristiques

Dans le contexte du Fog Computing, la gestion efficace des tâches est un enjeu central. Contrairement aux algorithmes déterministes, les algorithmes heuristiques offrent une prise de décision plus flexible. Leur but n'est pas de trouver la meilleure solution possible, mais une solution satisfaisante en un temps réduit. Cela les rend adaptés aux environnements distribués, dynamiques et incertains.

Aazam et al. [62] ont proposé un modèle d'Offloading basé sur la prévision des ressources. Chaque tâche reçoit une note selon la disponibilité anticipée des ressources et l'historique d'exécution. Le modèle prend en compte les changements dynamiques du réseau et ajuste l'allocation des tâches en temps réel, selon la situation coûtante. Elle est testée dans un environnement IoT intelligent et elle a donné de bons résultats sur la latence et le taux de réussite.

Deng et al. [63] ont conçu un algorithme qui répartit les tâches selon leur urgence et la capacité des nœuds Fog. Le système accorde une priorité élevée aux tâches urgentes et pondère plusieurs critères pour sélectionner le nœud cible. L'algorithme s'adapte automatiquement aux variations de charges. Il a amélioré les délais de traitement dans les réseaux surchargés comme ceux des véhicules connectés.

Yousef et al. [64] ont utilisé une approche First-Fit-Decreasing (FFD) combinée à une stratégie adaptative. Dans FFD, les tâches sont triées par leur taille, puis affectées aux premiers nœuds Fog capable de les exécuter. La stratégie est dite adaptative car le système surveille en continu l'état du réseau et réévalue les affectations à chaque nouvelle tâche. Cette méthode a été appliquée dans un scénario de ville intelligente avec vidéo surveillance. Les résultats montrent une baisse importante du temps de réponse global.

Rahbari et al. [65] ont proposé un système hybride vise à résoudre le problème d'Offloading

dans des environnements aux ressources instables (comme les réseaux IoT) en combinant une approche heuristique et une gestion dynamique des ressources. Pour éviter la surcharge des nœuds, le modèle utilise des règles heuristiques pour évaluer, avant tout transfert, si l'exécution d'une tâche sur un nœud cible risquerait de saturer ses ressources (CPU, mémoire, bande passante). Si le risque est trop élevé, la tâche est exécutée localement ou redirigée vers un autre nœud. Ensuite, le système intègre une vérification périodique après l'Offloading pour s'assurer que les conditions initiales (ex. : disponibilité des ressources) restent valides. Si un changement critique est détecté (ex. : pic de demande imprévu), la tâche peut être migrée dynamiquement vers un nœud plus stable. Cette double couche de décision (prédictive et réactive) permet de minimiser les échecs liés à la congestion ou aux fluctuations soudaines, tout en optimisant l'utilisation des ressources. Les simulations ont été réalisé dans des scénarios de forte variabilité, les résultats montrent une meilleure résilience face aux perturbations, réduisant les interruptions de service et améliorant la qualité de l'exécution des tâches critiques.

Les algorithmes heuristiques permettent de prendre des décisions rapides et adaptatives pour éviter la surcharge des nœuds et exécuter les tâches critiques, malgré l'instabilité des environnements Fog/IoT.

II.6.3 Algorithme de Machine Learning (ML)

Dans le cadre du Fog Computing, le Machine Learning joue un rôle clé dans la gestion des tâches distribuées. Ces algorithmes utilisent des données historiques et des informations en temps réel pour optimiser la répartition des tâches selon l'état du réseau, la charge des nœuds ou de la latence observée. Contrairement aux méthodes traditionnelles qui suivent des règles fixes, les algorithmes de ML apprennent et adaptent leurs comportement aux changements de l'environnement.

Les auteur Chen et al. [66], ont proposé un système qui utilise un algorithme de Machine Learning pour décider automatiquement où exécuter les tâches. Le traitement peut se faire sur l'appareil IoT, sur un serveur Fog proche, ou le Cloud. Leur aproche vise à réduire les temps d'attente et à économiser de l'énergie. Elle est adaptée aux réseaux de capteurs urbains, comme ceux utilisés pour l'éclairage public ou la gestion de trafic.

Wang et al. [67], ont proposé des méthodes pour protéger les données sensibles pendant le Task Offloading dans un environnement IoT/Fog. Au lieu d'envoyer les données personnelles, comme les signaux médicaux, vers un serveur central, chaque appareilles les traître localement. Ensuite, il partage uniquement les résultats d'apprentissage (mises à jour du modèle d'IA), sans transmettre les données brutes. Cette approche permet de préserver la confidentialité des utilisateurs et de réduire la latence de 20 %, car les données ne transitent pas par le Cloud.

Tang et al. [68] ont proposé une méthode basée sur le Machine Learning. Leur modèle est entraîné pour prédire la latence et la consommation d'énergie en fonction des caractéristiques des tâches et des nœuds. Ensuite, il décide automatiquement du meilleur lieu d'exécution. Cette méthode réduit la latence moyenne et améliore l'efficacité énergétique dans les systèmes Fog denses.

Liu et al. [69] ont développé une solution de Task Offloading basée sur le Machine Learning, une forme d'apprentissage par renforcement. L'algorithme observe l'état du système (charge, bande passante, délai) et apprend à choisir les actions qui optimisent les performances au fil du temps. Ce modèle évolue avec l'environnement et s'adapte aux changements. Il est efficace dans les scénarios mobiles ou imprévisibles.

Mao et al. [70] ont proposé un modèle d'optimisation de l'Offloading basé sur l'apprentissage profond, appliqué au mobile edge computing avec des appareils à énergie récoltée. Ils utilisent un réseau de neurones profond pour prendre des décisions en temps réel sur l'exécution locale ou déportée des tâches. Le système s'appuie sur une fonction de coût intégrant plusieurs critères, comme la latence, la consommation d'énergie et la qualité du canal. Il apprend à s'adapter automatiquement aux variations de l'environnement (batterie, réseau, charge). Cette approche est conçue pour des applications sensibles comme les soins de santé, les véhicules autonomes et l'IoT intelligent.

Les algorithmes de Machine Learning apprennent à partir des données et s'adaptent aux changements de l'environnement. Ils offrent une meilleure capacité à gérer la complexité et l'imprévisibilité des systèmes Fog IoT. Contrairement aux méthodes fixes, ils peuvent optimiser la répartition des tâches en temps réel, améliorer la performance et réduire la consommation d'énergie. Toutefois, ils demandent plus de ressources, de calculs et de données pour être efficaces, ce qui peut limiter leur déploiement sur certains appareils.

II.6.4 Algorithmes métaheuristiques

En tant que techniques d'optimisation inspirées de phénomènes naturels, biologiques ou sociaux, capables d'explorer efficacement un espace de solutions vastes et complexes. Leur principal avantage est de trouver des solutions proches de l'optimum en un temps raisonnable, même dans des environnements dynamiques et contraints, comme le Fog Computing. Ces méthodes sont particulièrement adaptées au Task Offloading, car elles permettent d'optimiser plusieurs critères à la fois (la latence, la consommation d'énergie, la bande passante, etc) dans des systèmes distribués et hétérogènes [71].

Bandyopadhyay et al. [72] ont proposé une approche centrée sur l'optimisation de Task Offloading des tâches dans un système Fog–Cloud hétérogène, en s'appuyant sur l'algorithme Particle Swarm Optimization (PSO). L'environnement modélisé repose sur un ensemble de tâches générées dynamiquement par des objets IoT, devant être affectées à des nœuds de traitement, soit dans le Fog, soit dans le Cloud. Chaque solution candidate est représentée sous forme de particule, illustrant une configuration d'assignation des tâches aux ressources disponibles. La qualité de chaque solution est évaluée à l'aide d'une fonction objectif qui intègre plusieurs paramètres clés : la latence globale, la consommation de ressources et les contraintes de capacité des nœuds de traitement. L'algorithme permet aux particules d'évoluer dans l'espace de recherche en s'appuyant à la fois sur leur expérience individuelle (meilleure solution personnelle) et sur celle de l'ensemble de l'essaim, favorisant ainsi une exploration collaborative vers des solutions efficaces. Cette dynamique facilite une convergence rapide vers des résultats performants. Les

résultats expérimentaux montrent que l'approche basée sur PSO permet une réduction significative du temps de réponse par rapport aux méthodes classiques, tout en assurant une meilleure répartition de la charge entre les nœuds Fog et Cloud. L'étude met en évidence la capacité du PSO à réduire les risques de saturation. Elle confirme ainsi la pertinence de cet algorithme en tant que solution robuste et adaptable pour la gestion dynamique des ressources dans les environnements distribués soumis à de fortes contraintes.

Sahu et al. [54] ont présenté une version améliorée de l'algorithme ACO, appelée SACO (Smart Ant Colony Optimization), dans le but de distribuer efficacement les tâches dans une architecture Fog multi-niveaux. Ici, la construction de la solution s'effectue à travers des agents intelligents, les "fourmis", qui sélectionnent les nœuds de traitement en fonction d'un mécanisme probabiliste lié au niveau de phéromone accumulé et à une heuristique associée au temps de traitement. L'algorithme prend en compte les capacités de traitement des nœuds, leur charge actuelle et la distance réseau. Il permet une prise de décision progressive, améliorée à chaque itération par une mise à jour des phéromones basée sur la qualité des solutions trouvées. L'approche SACO est comparée à plusieurs autres méthodes, dont PSO et BLA, et se révèle particulièrement performante. Elle permet notamment de réduire la latence moyenne de manière significative, tout en assurant une meilleure stabilité et un meilleur équilibrage des tâches. Ces résultats mettent en avant l'efficacité d'une stratégie inspirée du comportement collaboratif des fourmis pour optimiser le traitement décentralisé dans des systèmes Fog–Cloud complexes.

Dans Attalah et al. [73], une stratégie d'optimisation nommée GA Hybrid-Fog est proposée pour résoudre les défis du Task Offloading dans un environnement Fog hybride dédié à l'Internet des drones (IoD). Leur approche combine des stations Fog fixes (FBSs) et des drones Fog mobiles (FUAVs) afin de minimiser les délais de transmission et de traitement, tout en tenant compte des contraintes de ressources. Chaque solution est modélisée sous forme de chromosome, où chaque gène représente l'affectation d'une tâche à un nœud Fog ou à un traitement local. La fonction de fitness évalue le délai total, incluant la latence de transmission et le temps de traitement. Grâce aux opérateurs génétiques (sélection, croisement, mutation), l'algorithme explore dynamiquement les solutions possibles en s'adaptant à la mobilité des drones et à la disponibilité des ressources. Les résultats expérimentaux montrent que cette approche améliore l'efficacité de l'allocation des tâches dans des environnements Fog–Cloud dynamiques et distribués.

Saif et al. [74] proposent une approche d'optimisation du Task Offloading dans un environnement Edge–Fog–Cloud basée sur un algorithme Firefly multi-objectifs (MFA). Leur méthode repose sur une stratégie composée de deux étapes : d'abord, un module nommé Enhanced Task Offloading (ETO) sélectionne dynamiquement la couche de calcul (Edge, Fog ou Cloud) la plus adaptée à chaque tâche, selon la disponibilité des ressources et l'état de la charge réseau. Ensuite, le MFA est utilisé pour affecter la tâche à un dispositif de traitement optimal à l'intérieur de la couche choisie. L'algorithme Firefly s'inspire du comportement naturel des lucioles, où l'intensité lumineuse (fitness) guide le déplacement des agents vers de meilleures solutions. Dans cette version améliorée, plusieurs critères sont optimisés simultanément, notamment la

latence, la consommation d'énergie, et le taux d'échec. Les simulations montrent que l'approche proposée surpasse d'autres méthodes de référence, en améliorant l'utilisation des ressources et en réduisant les délais de traitement. Ce travail met en évidence la capacité des algorithmes bio-inspirés, comme le Firefly Algorithm, à résoudre efficacement des problèmes complexes de Task Offloading dans des environnements distribués et contraints.

Huang et al. [75] ont utilisé une approche par algorithmes génétiques. Leur méthode encode les différentes possibilités d'allocation de tâches dans des chromosomes. À chaque itération, les combinaisons les plus efficaces sont conservées et croisées pour générer de nouvelles solutions. Le processus continue jusqu'à atteindre une performance satisfaisante. Ce système est utilisé dans des réseaux Fog industriels avec des contraintes strictes.

Les algorithmes métaheuristique exploitent des mécanismes d'inspiration naturel pour trouver des solutions proches de l'optimum sans explorer tout l'espace possible. Ils sont efficaces dans les environnements Fog Cloud hétérogènes, dynamiques et contraints. Leurs forces réside dans leur capacité à gérer plusieurs critères simultanément comme la latence, la consommation d'énergie, la bande passante ou les ressources. Contrairement aux approches fixes ou aux heuristiques simples, ces méthodes s'adaptent aux changements du système et offrent une bonne robustesse face à l'incertitude.

Une synthèse comparative des différents algorithmes étudiés de Task Offloading est présentée dans le tableau II.1.

Catégorie	Travaux	Techniques	Latence	Énergie	Ressources	Coût
Déterministes	You et al. [59]	Algorithme à Seuils	×	×		
	Buchholz et al. [60]	Algorithme PIER	×	×		
	Yu et al. [61]	Optimisation Conjointe	×	×	×	
Heuristiques	Aazam et al. [62]	Prévision dynamique	×		×	
	Deng et al. [63]	Priorisation intelligente	×		×	
	Yousef et al. [64]	First-Fit-Decreasing	×		×	
	Rahbari et al. [65]	Anti-saturation		×	×	
Machine Learning	Chen et al. [66]	IA (local/Fog/Cloud)	×	×	×	×
	Wang et al. [67]	Apprentissage fédéré	×			×
	Tang et al. [68]	Réseaux de neurones	×	×		×
	Liu et al. [69]	Q-Learning	×		×	×
	Mao et al. [70]	ML	×	×		×
Méta-heuristiques	Bandyopadhyay et al. [72]	PSO	×	×	×	×
	Sahu et al. [54]	SACO (fourmis)	×		×	×
	Attalah et al. [73]	GA (drones)	×	×		×
	Huang et al. [75]	GA	×	×	×	×
	Saif et al. [74]	Firefly Algorithm (MFA)	×	×		×

Tab. II.1 : Comparaison des algorithmes de Task Offloading selon les critères de performance.

II.7 Contraintes défis et limitations du Task offloading

Le offloading dans le Fog Computing transfère le traitement des données des appareils IoT vers des nœuds Fog plus puissants, améliorant la réactivité et réduisant la charge, mais générant des contraintes importantes.

II.7.1 Gestion de la latence et de la fiabilité

Le Task Offloading vise à réduire la latence. Pourtant, cet objectif est compromis si la tâche est dirigée vers un nœud distant, indisponible ou surchargé. Dans le Fog Computing, la qualité de service dépend de la proximité, de la capacité de calcul et de la stabilité du nœud. Un mauvais routage augmente les délais de traitement et détériore les performances. La fiabilité reste une limite majeure. Les nœuds Fog sont souvent mobiles, instables et exposés à des interruptions. Cette variabilité rend difficile la continuité de service. Dans un système temps réel, une défaillance ponctuelle peut causer l'échec total d'une application [76].

II.7.2 Consommation d'énergie

L'offloading peut réduire la consommation locale d'énergie, mais cela dépend des décisions prises par le système. Chaque transfert implique une dépense énergétique pour la transmission et le traitement. Si l'économie espérée est mal évaluée, la consommation totale peut augmenter. Ce problème touche directement les dispositifs IoT alimentés par batterie. Une stratégie d'offloading mal calibrée peut entraîner une décharge rapide, sans l'autonomie des capteurs, drones ou objets intelligents. Dans certains cas, une mauvaise allocation augmente la consommation de plus de 40%. À l'inverse, les algorithmes adaptatifs permettent de réduire cette consommation jusqu'à 28% , en tenant compte de l'état de la batterie et de la priorité des tâches [77].

II.7.3 Sécurité et confidentialité

Le Task Offloading expose les données à des risques (interception, accès non autorisé), notamment en raison de la diversité des équipements et des failles de sécurité. Pour les protéger, des mécanismes comme le chiffrement, l'authentification et le contrôle d'accès sont essentiels. [78].

II.7.4 Scalabilité et interopérabilité

Le Fog Computing doit gérer un grand nombre d'appareils connectés. Quand ce nombre augmente vite, le système peut devenir lent ou moins efficace. C'est le problème de la scalabilité. Le système doit rester rapide, même avec plus de données et plus de connexions. L'interopérabilité est un autre problème. Les appareils viennent de marques différentes. Ils n'utilisent pas les mêmes systèmes ou les mêmes façons de communiquer. Pour qu'ils puissent bien fonctionner ensemble, il faut des règles communes. On utilise pour cela des standards, des interfaces de programmation souples, et une structure qui permet de s'adapter facilement [79].

II.7.5 Tolérance au pannes

Dans le Fog Computing, les coupures sont fréquentes. Un nœud peut tomber en panne, perdre la connexion ou être déplacé. Ces problèmes peuvent bloquer le traitement des tâches. Le système doit donc détecter ces incidents, rediriger les tâches et relancer les opérations sans délai. Pour cela, il faut des mécanismes de secours. Par exemple, avoir une copie des données, surveiller les nœuds en temps réel et redémarrer automatiquement en cas d'erreur. Si ces protections ne sont pas bien mises en place, le système peut devenir instable, surtout dans les domaines sensibles, comme la santé ou l'industrie.

II.8 Conclusion

Dans ce chapitre, nous avons présenté les fondements du Task Offloading dans le Fog Computing, en détaillant ses principales approches, les classes d'algorithmes existants et les métriques utilisées pour en évaluer les performances.

Cette étude nous permis de mettre en évidence les avantages et limites de chaque méthode, notamment en termes de latence, de consommation énergétique et d'adaptabilité. Ces éléments constituent une base essentielle pour orienter la conception de notre propre proposition, qui sera présentée dans le chapitre suivant, accompagnée d'une évaluation comparative.

Chapitre III

Proposition et évaluation de performances

III.1 Introduction

Dans le domaine du Fog Computing, l'offloading des tâches représente un défi majeur. L'objectif est d'assigner les tâches aux ressources de calcul de manière optimale, afin de réduire le temps de réponse tout en maintenant une qualité de service (QoS) élevée pour les utilisateurs.

Dans cette optique, nous proposons une approche fondée sur la métaheuristique ABC (Artifial Bee Colony) pour résoudre le problème de Task Offloading dans un environnement Fog. Afin d'évaluer les performances d'ABC, nous le comparerons à une autre métaheuristique bien connue : PSO (Particle Swarm Optimization).

III.2 Modélisation du problème de Task Offloading dans le Fog Computing

Cette section présente la modélisation formelle de notre contribution. L'objectif principal est d'optimiser deux métriques clé de qualité de services : Le temps d'exécution et la consommation d'énergie. Nous modélisons efficacement le processus de Task Offloading dans un environnement Fog. On suppose les taches ne peuvent pas être exécutés localement et sont directement offloadées vers une machine virtuelle (VM) du Fog. Le système détermine la VM optimale pour chaque tâche afin d'améliorer la performance globale du réseau, tout en respectant les contraintes matérielles.

III.2.1 Les métriques QoS

→ **Temps d'exécution**

Le temps d'exécution représente la durée nécessaire à une machine virtuelle pour traiter une tâche T_i. Il dépend de deux facteurs principaux : la taille de la tâche et la capacité de traitement de la VM choisie. L'équation est donner comme suite :

$$T_i^{\text{exec}} = \frac{L_i}{f_r} \tag{III.1}$$

Ou :

- T_i^{exec} : temps d'exécution de la tâche t_i (en secondes).
- L_i : taille de la tâche t_i (en millions d'instructions, MI)
- f_r : capacité de traitement de la VM r (en MIPS - Millions d'Instructions Par Seconde)

Cette équation permet de calculer, le temp d'execution de la tache i dans la machine virtuelle (MVr).

→ **Consommation d'énergie :**

L'énergie consommée par un nœud Fog dépend de la charge totale de traitement qui lui est assignée. Cette consommation est modélisée par la fonction suivante[74] :

$$P_j^{fog} = a \cdot (Y_j^{fog})^2 + b \cdot Y_j^{fog} + c \tag{III.2}$$

Où :

- P_j^{fog} : énergie consommée par le nœud Fog j pour le traitement de toutes les tâches qui lui sont affectées.
- Y_j^{fog} : charge totale affectée au nœud Fog j, calculée comme suite :

  $$Y_j^{fog} = \sum_{i \in \mathcal{T}_j} L_i$$

  où L_i est la taille de la tâche t_i (en millions d'instructions) et $\mathcal{T}_j$ l'ensemble des tâches affectées au nœud j.
- $a > 0$, $b \geq 0$, $c \geq 0$: coefficients représentant respectivement la consommation quadratique, linéaire et statique du nœud Fog.

Cette formulation permet de capturer l'effet de surcharge sur un nœud Fog. En effet, une affectation déséquilibrée des tâches peut entraîner une augmentation rapide de la consommation d'énergie, ce qui motive la recherche d'un offloading optimal.

$$E^{\text{total}} = \sum_{j=1}^{M} P_j^{fog} \tag{III.3}$$

avec M le nombre total de nœuds Fog dans le système.

III.2.2 Fonction objectif (fitness)

$$F_{(P)} = \cdot \sum_{i=1}^{N} w_1 . T_i^{\text{exec}} + w_2 \cdot E^{\text{total}} \tag{III.4}$$

Où :

- $F_{(P)}$: valeur de la solution P
- w_1, w_2 : poids associés au temps d'exécution et à la consommation d'énergie qui sont entre 0 et 1.
- $\sum_{i=1}^{N} T_i^{\text{exec}}$: somme des temps d'exécution de toutes les tâches
- E^{total} : énergie totale consommée par les nœuds Fog

III.2.3 Problème de Task Offloading

Chaque solution est représentée par un vecteur de décision ou chaque position correspond à une tâche, et chaque valeur indique la ressource assignée à cette tâche. Il s'agit d'affecter un ensemble de tâches $t = \{t_1, t_2, \ldots, t_N\}$ à un ensemble de ressources disponibles

$M = \{m_1, m_2, \ldots, m_k\}$, comprenant des dispositifs Fog. L'objectif est de déterminer la ressource optimale pour chaque tâche, afin de minimiser à la fois le temps d'exécution totale et la consommation d'énergie globale du système.

Le système Fog est modélisé comme un ensemble de machines physiques noté :

$$dFog = \{MP_1, MP_2, \ldots, MP_{N_{pm}}\} \tag{III.5}$$

où chaque machine physique MP_i (avec i = 1,2 …,N_{pm}) représente un dispositif Fog, et peut être décrite comme suit :

$$MP_i = \{VM_1, VM_2, \ldots, VM_{Nvm}\} \tag{III.6}$$

Chaque machine physique contient plusieurs machines virtuelles (VM), et chaque machine virtuelle est caractérisée par :

$$VM_k = [ID_{VM_k}, MIPS_k] \tag{III.7}$$

ou :

- ID_{VM_k} : désigne l'identifiant unique de la VM
- $MIPS_k$: représente sa capacité de traitement en millions d'instructions par seconde (MIPS)

L'ensemble des tâches à offloader est représenté par :

$$T = \{T_1, T_2, \ldots, T_{N_{\text{tsk}}}\} \tag{III.8}$$

Chaque tâche est définie comme suit :

$$T_i = [ID_{T_i}, Tlength_i, ECT_i] \tag{III.9}$$

ou :

- ID_{T_i} : l'identifiant de la tâche T_i
- $Tlength_i$: la taille de la tâche (en millions d'instructions)
- ECT_i : le temps d'exécution estimé

Chaque solution possible est représentée sous forme d'un vecteur de décision, dans lequel chaque position du vecteur correspond à une tâche, et la valeur indique la machine Fog (VM) qui lui est assignée. L'algorithme ABC est utilisé pour explorer l'espace de recherche et trouver une répartition optimale des tâches, tout en respectant les contraintes matérielles.

III.2.4 Motivation

Le Task Offloading est devenu cruciale avec l'essor des objets connectés et la croissance massive des données. Les méthodes exactes, efficaces dans des cas simples, échouent face à la complexité des environnements Fog.

Les métaheuristiques proposent des solutions adaptée et explorent efficacement l'espace de recherche pour affecter les tâches aux bonnes ressources, en tenant compte du temps d'exécution, de l'énergie consommée.

Elles produisent de bons résultats en un temps raisonnable et s'adaptent à divers objectifs comme la réduction du temps d'exécution ou de la consommation d'énergie. Elles peuvent aussi être combinées à l'apprentissage automatique pour améliorer leur efficacité.

III.3 Algorithme d'optimisation ABC pour le Task Offloading

Algorithme1: ABC (Artifial Bee Colony) pour le Task Offloading
Input : Entrées

- n_tasks : Nombre de tâches
- $n\_machines$: Nombre de machines disponibles
- $machines\_speed$: Vitesse des machines
- $task\_sizes$: Tailles des tâches
- $data\_sizes$: Données à transférer
- a, b, c : Paramètres énergétiques
- max_iter : Nombre maximal d'itérations
- $limit$: Limite d'échecs sans amélioration
- SN : Nombre de solutions (sources de nourriture)

Output : Sorties

- Meilleure solution
- Temps d'exécution total
- Énergie consommée totale

Générer SN solutions aléatoires
Évaluer chaque solution
Sauvegarder la meilleure solution trouvée

Algorithm 1 : Algorithme ABC (Artifial Bee Colony) pour le Task Offloading

```
for t ← 1 to max_iter do
    Phase des abeilles employées for chaque solution x_i do
        Générer une solution candidate v_i (Formule I.6)
        Évaluer v_i
        if fitness(v_i) > fitness(x_i) then
            x_i ← v_i (Formule I.7)
            Réinitialiser le compteur d'échecs de x_i
            Mettre à jour la meilleure solution si nécessaire
        else
            Incrémenter le compteur d'échecs de x_i
        end if
    end for
    Phase des abeilles observatrices Calculer les probabilités p_i (Formule I.8)
    for chaque abeille observatrice do
        Sélectionner une solution x_i selon p_i
        Générer une solution candidate v_i (Formule I.9)
        Évaluer v_i
        if fitness(v_i) > fitness(x_i) then
            x_i ← v_i (Formule I.10)
            Réinitialiser le compteur d'échecs
            Mettre à jour la meilleure solution si nécessaire
        else
            Incrémenter le compteur d'échecs
        end if
    end for
    Phase des abeilles éclaireuses for chaque solution x_i do
        if compteur d'échecs x_i ≥ limit then
            Remplacer x_i par une nouvelle solution aléatoire (Formule I.11)
            Réinitialiser le compteur d'échecs
            Évaluer la nouvelle solution
            Mettre à jour la meilleure solution si nécessaire
        end if
    end for
end for
```

III.4 Réalisation et évaluation de performance

L'implémentation des algorithmes a été réalisée à l'aide du logiciel MATLAB (MathWorks, 2023), en appliquant l'algorithme ABC au problème de Task Offloading. Afin d'évaluer ses performances, nous le comparons à un autre algorithme d'optimisation PSO. L'objectif est de déterminer lequel des deux fournit de meilleurs résultats en termes de temps d'exécution, consommation d'énergie et la valeur de la fonction objectif.

Les paramètres expérimentaux utilisés sont présentés dans tableau III.1

Nous définissons deux expériences. Dans la première expérience, le nombre de machines virtuelles est maintenu constant à 30, tandis que le nombre de tâches augmente progressivement de 100 à 1000 (pas de 200).

Dans la deuxième expérience, le nombre de tâches est fixe (500), alors que le nombre de machines virtuelles varie de 10 à 60, avec un pas de 10.

Paramètre	**Expérience 1**	**Expérience 2**
CPU vitesse du Processeur	[500, 2000] MIPS	
Taille des tâches	[2000, 50000] MI	
Taille de la population	40	
Nombre d'itération	150	
Nombre de tâches	[100, 300, 500, 700, 900, 1000]	500
Nombre de machines virtuelles	30	[10, 20, 30, 40, 50, 60]

Tab. III.1 : Ensemble de paramètres d'expérimentation

III.5 Résultats et discussions

Les résultats obtenus sont regroupés dans les tableaux III.2 III.3, ils sont analysés et comparés dans la section suivante :

III.5.1 Résultats de la première expérience

La première expérience						
Nb. tâches	**PSO**			TO-ABC		
	Temps d'execution (s)	**Énergie**	**Fitness**	**Temps d'execution (s)**	**Énergie**	**Fitness**
100	1995.05	6.91	1000.98	1733.48	6.58	870.03
300	6849.07	1.63	3432.71	6588.19	1.55	3301.85
500	11802.71	2.57	5914.20	11208.19	2.64	5617.28
700	16640.74	4.01	8340.43	16452.06	3.85	8245.31
900	22670.25	5.65	11363.38	22516.12	5.35	11284.81
1000	25126.11	6.02	12593.16	24041.26	6.02	12050.71

Tab. III.2 : Ensemble des résultats obtenus dans la première expérience

Dans cette expérience, les résultats montrent que l'algorithme TO-ABC est meilleure que PSO sur tous les critères évalués. En de temps d'exécution et d'énergie. ABC offre de meilleures

performances, notamment lorsque le nombre de tâches augmente, ce qui démontre sa robustesse dans des scénarios complexes. Concernant le temps d'exécution, ABC est significativement plus rapide que PSO, ce qui est crucial pour les applications en temps réel nécessitant une réactivité élevée. Enfin, aux attentes initiales, ABC surpasse également PSO en terme de consommation d'énergie, offrant ainsi une solution globale plus efficace et économique.

Comme attesté par les figures III.1 III.2 III.3

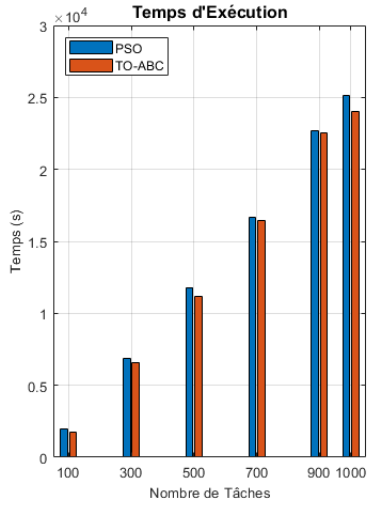

Fig. III.1 : Temps d'exécution : TO-ABC vs PSO

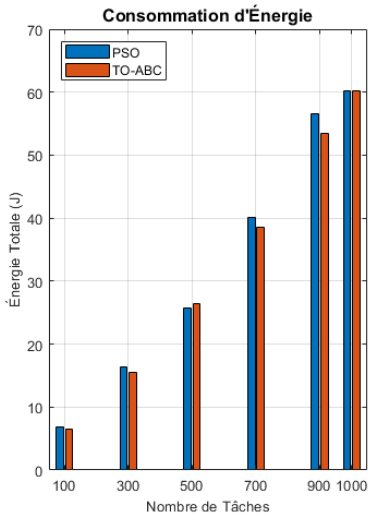

Fig. III.2 : Consommation d'énergie : TO-ABC vs PSO

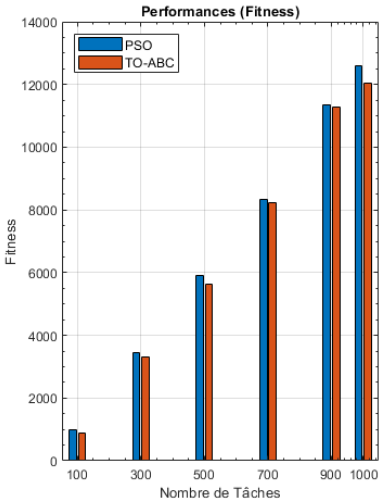

Fig. III.3 : Fitness : TO-ABC vs PSO

III.5.2 Résultats de la deuxième expérience

La deuxième expérience				
Nb. machines	**PSO**		**ABC**	
	Temps (s)	**Énergie**	**Temps (s)**	**Énergie**
10	10125.55	3.49	9920.07	3.43
20	12420.32	2.51	12442.26	2.56
30	12617.74	2.31	12358.90	2.27
40	11813.79	2.39	11822.29	2.38
50	12012.90	2.51	12152.88	2.57
60	14162.62	2.58	13921.11	2.55

Tab. III.3 : Ensemble des résultats obtenus dans la deuxième expérience

La deuxième expérience confirme les résultats de la première expérience. L'algorithme TO-ABC donne de meilleurs résultats que PSO, notamment en termes de temps d'exécution le grand nombres de machines important.

De plus, contrairement aux observations initiales, ABC montre une meilleurs efficacité d'énergie que PSO dans ces nouveaux tests. Cette amélioration, ses avantages combinés à la rapidité et la qualité des solutions, font d'ABC le choix incontournable pour les applications exigeantes où la performance globale et l'optimisation des ressources sont cruciales.

Les figures III.4 III.5 illustrent les résultat de tableau

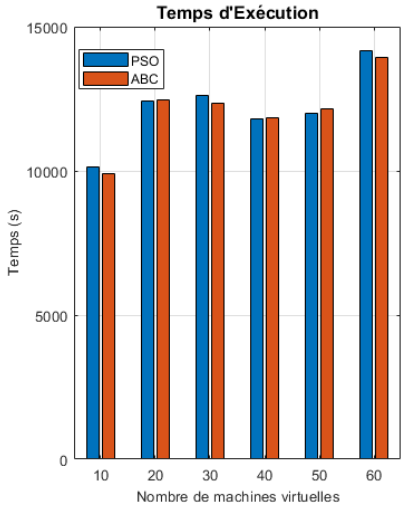

Fig. III.4 : Temps d'exécution : TO-ABC vs PSO

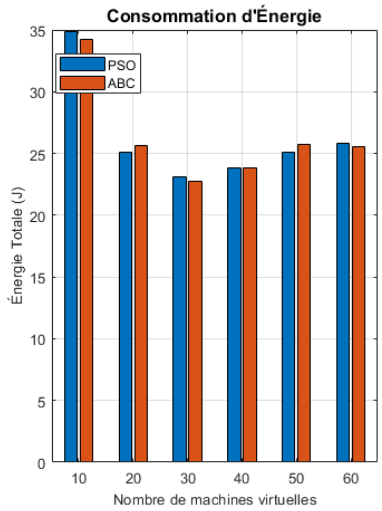

Fig. III.5 : Consommation d'énergie : TO-ABC vs PSO

III.6 Synthèse

Les résultats des deux expériences montrent que l'algorithme Artificial Bee Colony (ABC) surpasse Particle Swarm Optimization (PSO) pour le Task Offloading dans un environnement Fog Computing. L'amélioration globale moyenne est de 15.83 %.

L'analyse des résultats confirme la supériorité de l'algorithme ABC.

- **Temps d'exécution** : To-ABC est plus rapide. Pour 1000 tâches, le gain est de 4.31 %. Cette tendance se retrouve dans les autres cas.
- **Consommation d'énergie** : TO-ABC consomme moins d'énergie. Pour 100 tâches, la réduction atteint 4.77 %.
- **Fitness** : TO-ABC fournit des solutions plus optimales. Pour 100 tâches, la valeur de fitness est 13.08 % inférieure à celle obtenue par PSO.

ABC est donc plus adapté. Il combine rapidité, économie d'énergie et qualité de solution, même avec un grand nombre de tâches.

III.7 Conclusion

L'analyse comparative entre les algorithme ABC et PSO pour le Task Offloading le Fog montre que ABC est globalement plus performant. Il offre un temps d'exécution plus court, une meilleure efficacité d'énergie et une meilleure qualité de solution. Ces résultats font de l'algorithme ABC un choix préférable pour les systèmes nécessitant rapidité et optimisation des ressources.

Conclusion Générale

À travers ce mémoire, nous avons exploré l'une des problématiques clés liées à l'évolution des système distribués modernes : le Task Offloading dans un environnement Fog Computing. Notre travail s'est concentré sur l'optimisation du processus d'offloading des tâches générées par les objets IoT, dans le but de réduire à la fois le temps d'exécution et la consommation d'énergie.

Pour cela, nous avons adapté la metaheuristique ABC pour résoudre le problème de Task Offloading dans le Fog Computing en suit nous a vous réalisé une étude comparative avec la metaheuristique PSO, connu et largement utilisé dans la littérature, les deux méthodes ont été implémentées sous MATLAB et évaluée à travers deux séries d'expériences.

Les résultats obtenus ont montré que l'algorithme TO-ABC est plus performant que PSO. Il se distingue par une meilleure consommation d'énergie, un temps d'exécution plus court et une meilleure qualité de solution (fitness), en particulier lorsque le nombre de tâches augmente. Cette comparaison monte que ABC est plus adapté dans les scénarios exigeant efficacité d'énergie, temps d'exécution et performance globale.

Ce travail nous a permis de mieux comprendre les enjeux techniques du Fog Computing et d'apporter une contribution concrète à la résolution d'un problème complexe lié au Task Offloading.

Comme perspectives futures, il serait pertinent d'approfondir cette étude en testant d'autres méthaheuristiques, en explorant des approche hybrides ou en intégrant des techniques d'apprentissage automatique afin d'adapter de manière dynamiquement la stratégie d'Offloading en fonction de l'évolution des paramètres réseau et système.

Bibliographie

[1] Statista. Iot technology trends and predictions for 2024. `https://mobidev.biz/blog/iot-technology-trends`, 2020. Consulté en mai 2025.

[2] A Shaji George. The role of fog computing in enabling real-time iot applications. *Partners Universal International Innovation Journal*, 2(2) :39–54, 2024.

[3] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things : A survey. *Computer Networks*, 54(15) :2787–2805, 2010. doi : 10.1016/j.comnet.2010.05.010.

[4] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot) : A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7) :1645–1660, 2013.

[5] Hamed HaddadPajouh, Ali Dehghantanha, Reza M Parizi, Mohammed Aledhari, and Hadis Karimipour. A survey on internet of things security : Requirements, challenges, and solutions. *Internet of Things*, 14:100129, 2021.

[6] NomoSense. Comment se constitue une architecture réseau iot, 2022. URL `https://www.nomosense.com/comment-se-constitue-une-architecture-reseau-iot/`. Consulté en mai 2025.

[7] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of things : A survey on enabling technologies, protocols, and applications. *IEEE communications surveys &amp ; tutorials*, 17(4) :2347–2376, 2015.

[8] Rafael Marques. *Système de produits et services bas'es sur l39;Internet des objets : conception et implantation pilote dans une station-service*. Ecole Polytechnique, Montreal (Canada), 2018.

[9] Clovis Anicet Ouedraogo, Samir Medjiah, Christophe Chassot, and Khalil Drira. Enhancing middleware-based iot applications through run-time pluggable qos management mechanisms. application to a onem2m compliant iot middleware. *Procedia computer science*, 130:619–627, 2018.

[10] Partha Pratim Ray. A survey on internet of things architectures. *Journal of King Saud University-Computer and Information Sciences*, 30(3) :291–319, 2018.

[11] Rodrigo Roman, Jianying Zhou, and Javier Lopez. On the features and challenges of security and privacy in distributed internet of things. *Computer networks*, 57(10) :2266–2279, 2013.

[12] SM Riazul Islam, Daehan Kwak, MD Humaun Kabir, Mahmud Hossain, and Kyung-Sup Kwak. The internet of things for health care : a comprehensive survey. *IEEE access*, 3: 678–708, 2015.

[13] Antonis Tzounis, Nikolaos Katsoulas, Thomas Bartzanas, and Constantinos Kittas. Internet of things in agriculture, recent advances and future challenges. *Biosystems engineering*, 164:31–48, 2017.

[14] Li Da Xu, Eric L Xu, and Ling Li. Industry 4.0: state of the art and future trends. *International journal of production research*, 56(8) :2941–2962, 2018.

[15] Chaimae Jourra. 0 iot généralités. `https://fr.scribd.com/document/581483220/0-Iot-Generalies`, 2022. Consulté en mai 2025.

[16] Suman Madan and Kanika Behl. Cloud computing : Leveraging shared services for it transformation. *IITM JOURNAL OF MANAGEMENT AND IT*, 2(2) :52–61, 2011.

[17] Futura Sciences. Cloud computing : Définition, traduction et acteurs, 2025. URL `https://www.futura-sciences.com/tech/definitions/informatique-cloud-computing-11573/`.

[18] Houssem Medhioub. *Architectures et m&39;ecanismes de f&39;ed&39;eration dans les environnements cloud computing et cloud networking*. PhD thesis, Institut National des T&39;el&39;ecommunications, 2015.

[19] EDC Paris Business School. Cloud computing : Définition, types et services, 2023. URL `https://www.edcparis.edu/fr/blog/cloud-computing-definition-types-et-services`. Consulté le [date de consultation].

[20] IONOS Digital Guide. Cloud computing : Définition, fonctionnement et avantages, 2023. URL `https://www.ionos.fr/digitalguide/serveur/know-how/cloud-computing/`.

[21] Advancia-IT System. Virtualisation : vers une nouvelle ère pour les infrastructures informatiques, 2012. URL `https://www.advancia-itsystem.com/site/fr/news_details.php?id_article=19&id_news=139`.

[22] Rahul Gupta. Architecture of cloud computing with explanation, 2023. URL `https://pywix.blogspot.com/2023/03/cloud-architecture.html`. Consulté en mai 2025.

[23] Imen Bouzarkouna. *Impl&39;ementation industrielle duquot ; Fog Manufacturingquot ;: d&39;efis et applications à la logistique interne dans le contexte de l'industrie du futur*. PhD thesis, INSA Rouen ; LINEACT Rouen, 2021.

[24] Bennabi Narimane and Ait Ibrahim Nassima. *Conception et r&39;ealisation d39;une base de donn&39;ees NoSQL sous Hbase et Firebase Cas d39;un tremblement de terre.* PhD thesis, Universit&39;e Mouloud Mammeri, 2019.

[25] Charith Perera, Yongrui Qin, Julio C. Estrella, Stephan Reiff-Marganiec, and Athanasios V. Vasilakos. Fog computing for sustainable smart cities : A survey. *ACM Computing Surveys (CSUR)*, 50(3) :32, 2017. doi : 10.1145/3057266.

[26] Redowan Mahmud, Kotagiri Ramamohanarao, and Rajkumar Buyya. Latency-aware application module management for fog computing environments. *ACM Transactions on Internet Technology (TOIT)*, 19(1) :1–21, 2018.

[27] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16, 2012.

[28] Mung Chiang and Tao Zhang. Fog and iot : An overview of research opportunities. *IEEE Internet of things journal*, 3(6) :854–864, 2016.

[29] Luis M Vaquero and Luis Rodero-Merino. Finding your way in the fog : Towards a comprehensive definition of fog computing. *ACM SIGCOMM computer communication Review*, 44(5) :27–32, 2014.

[30] Korupalli V Rajesh Kumar, K Dinesh Kumar, Ravi Kumar Poluru, Syed Muzamil Basha, and M Praveen Kumar Reddy. Internet of things and fog computing applications in intelligent transportation systems. In *Architecture and security issues in fog computing applications*, pages 131–150. IGI Global, 2020.

[31] Rabeea Basir, Saad Qaisar, Mudassar Ali, Monther Aldwairi, Muhammad Ikram Ashraf, Aamir Mahmood, and Mikael Gidlund. Fog computing enabling industrial internet of things : State-of-the-art and research challenges. *Sensors*, 19(21) :4807, 2019.

[32] Bastien Confais. *Conception d39;un système de partage de donn&39;ees adapt&39;e à un environnement de Fog Computing.* PhD thesis, Universit&39;e de Nantes, 2018.

[33] Dongho You, Tung V Doan, Roberto Torre, Mahshid Mehrabi, Alexander Kropp, Vu Nguyen, Hani Salah, Giang T Nguyen, and Frank HP Fitzek. Fog computing as an enabler for immersive media : Service scenarios and research opportunities. *IEEE Access*, 7:65797–65810, 2019.

[34] Abdul Majid Farooqi, M Afshar Alam, Syed Imtiyaz Hassan, and Sheikh Mohammad Idrees. A fog computing model for vanet to reduce latency and delay using 5g network in smart city transportation. *Applied Sciences*, 12(4) :2083, 2022.

[35] Michaela Iorga, Larry Feldman, Robert Barton, Michael J Martin, Nedim S Goren, and Charif Mahmoudi. Fog computing conceptual model. 2018.

[36] Amir Vahid Dastjerdi and Rajkumar Buyya. Fog computing : Helping the internet of things realize its potential. *Computer*, 49(8) :112–116, 2016.

[37] Ivan Stojmenovic and Sheng Wen. The fog computing paradigm : Scenarios and security issues. In *2014 federated conference on computer science and information systems*, pages 1–8. IEEE, 2014.

[38] Prateeksha Varshney and Yogesh Simmhan. Demystifying fog computing : Characterizing architectures, applications and abstractions. In *2017 IEEE 1st international conference on fog and edge computing (ICFEC)*, pages 115–124. IEEE, 2017.

[39] Faisal Karim Shaikh, Sherali Zeadally, and Ernesto Exposito. Enabling technologies for green internet of things. *IEEE Systems Journal*, 11(2) :983–994, 2016. doi : 10.1109/JSYST.2015.2415194.

[40] Matthew Weiner, Milos Jorgovanovic, Anant Sahai, and Borivoje Nikoli&39;e. Design of a low-latency, high-reliability wireless communication system for control applications. In *2014 IEEE International conference on communications (ICC)*, pages 3829–3835. IEEE, 2014.

[41] Innen Bouarkouma. *Implémentation industrielle du quot ;Fog Manufacturingquot ;: défis et applications à la logistique interne dans le contexte de l39;industrie du futur*. PhD thesis, INSA Rouen ; LINEACT Rouen, 2023. URL `https://hal.science/tel-04108010v1`. HAL Id : tel-04108010.

[42] Vaek Chv&39;atal. *Linear Programming*. W. H. Freeman, 1983.

[43] Amira Gherboudj. *Méthodes de résolution de problèmes difficiles académiques*. PhD thesis, Université de Constantine, Faculté des Technologies de l39;Information et de la Communication, Constantine, Algérie, 2013. Thèse de doctorat en Informatique, soutenue devant le jury composé de : Pr. Allaoua Chaoui (Président), Pr. Salim Chikhi (Directeur de thèse), Dr. Randane Mamrei, Dr. Baghdad Atmani, Dr. Fojil Cherf (Examinateurs). Année universitaire 2012-2013.

[44] Carla Mouradian, Diala Naboulsi, Sami Yangui, Roch H. Glitho, Monique J. Morrow, and Paul A. Polakos. A comprehensive survey on fog computing : State-of-the-art and research challenges. *IEEE Communications Surveys & Tutorials*, 20(1) :416–464, 2018. ISSN 1553-877X. doi : 10.1109/COMST.2017.2771153.

[45] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. ieee, 1995.

[46] James Kennedy and Russell C Eberhart. A discrete binary version of the particle swarm algorithm. In *1997 IEEE International conference on systems, man, and cybernetics. Computational cybernetics and simulation*, volume 5, pages 4104–4108. ieee, 1997.

[47] Baris Yuce, Michael S. Packianather, Ernesto Mastrocinque, Duc Truong Pham, and Alfredo Lambiase. Honey bees inspired optimization method : The bees algorithm. *Insects*, 4(4) : 646–662, 2013. doi : 10.3390/insects4040646. URL `https://www.mdpi.com/2075-4450/4/4/646`.

[48] BENCHENNA Ahmed Khaireddine ET BOUBEKEUR Ahmed and BOUBEKEUR Ahmed Bouzid Bouzid. Optimisation des paramètres du régulateur de vitesse d'un véhicule électrique par la technique pso.

[49] Dervis Karaboga. Artificial bee colony algorithm. *Scholarpedia*, 5(3) :6915, 2010. URL `http://www.scholarpedia.org/article/Artificial_bee_colony_algorithm`. Version révisée le 2 juillet 2011.

[50] Jaber Almutairi and Mohammad Aldossary. Exploring and modelling iot offloading policies in edge cloud environments. *Computer Systems Science & Engineering*, 41(2), 2022.

[51] ResearchGate. Illustration of fog computing architecture, 2023. URL `https://www.researchgate.net/figure/Illustration-of-fog-computing-architecture_fig2_380665643`. Consulté le [date].

[52] M Vedaraj, A Satwika, B Monisha, and I Namratha. Energy optimized and dynamic design of task offloading in mobile fog computing. *Grenze International Journal of Engineering and Technology*, 10(2), 2024.

[53] Mohammad Reza Rezaee, Nor Asilah Wati Abdul Hamid, Masnida Hussin, and Zuriati Ahmad Zukarnain. Fog offloading and task management in iot-fog-cloud environment : Review of algorithms, networks and sdn application. *IEEE Access*, 2024.

[54] Amit Kishor and Chinmay Chakarbarty. Task offloading in fog computing for using smart ant colony optimization. *Wireless personal communications*, 127(2) :1683–1704, 2022.

[55] Yasaman Seraj, Soheil Fadaei, Bardia Safaei, Ali Javadi, Amir Mahdi Hosseini Monazzah, and Ali Mohammad Afshin Hemmatyar. Limo : Load-balanced offloading with mape and particle swarm optimization in mobile fog networks. *arXiv preprint arXiv :2408.14218*, 2024.

[56] Faten A Saif, Rohaya Latip, Zurina Mohd Hanapi, Kamarudin Shafinah, AV Senthil Kumar, and Awadh S Bajaher. Multi-objectives firefly algorithm for task offloading in the edge-fog-cloud computing. *IEEE Access*, 2024.

[57] Alice Martin. *Optimisation des stratégies de délégation hybride dans les réseaux edge*. PhD thesis, Sorbonne Université, Décembre 2020. URL `https://theses.hal.science/tel-02268196v1`. Thèse de doctorat. Consulté le 10 novembre 2023.

[58] Faten Alenizi and O Rana. Fog computing : Towards dynamically controlling the offloading threshold and managing fog resources in online dynamic systems. *Preprints. org*, 2021.

[59] Changsheng You, Kaibin Huang, Hyukjin Chae, and Byoung-Hoon Kim. Energy-efficient resource allocation for mobile-edge computation offloading (extended version). *arXiv preprint arXiv :1605.08518*, 2016.

[60] Florian Bock, Sebastian Siegl, Peter Bazan, Peter Buchholz, and Reinhard German. Corrigendum to "reliability and test effort analysis of multi-sensor driver assistance systems" [journal of systems architecture 85-86 (2018) 1–13]. *Journal of Systems Architecture*, 117: 102117, 2021. ISSN 1383-7621. doi : https://doi.org/10.1016/j.sysarc.2021.102117. URL `https://www.sciencedirect.com/science/article/pii/S1383762121000898`.

[61] Yinghao Yu, Jun Zhang, and Khaled B Letaief. Joint subcarrier and cpu time allocation for mobile edge computing. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2016.

[62] Mohammad Aazam and Eui-Nam Huh. Fog computing and smart gateway based communication for cloud of things. In *2014 International conference on future internet of things and cloud*, pages 464–470. IEEE, 2014.

[63] Ruilong Deng, Rongxing Lu, Chengzhe Lai, Tom H Luan, and Hao Liang. Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption. *IEEE internet of things journal*, 3(6) :1171–1181, 2016.

[64] Ashkan Yousefpour, Caleb Fung, Tam Nguyen, Krishna Kadiyala, Fatemeh Jalali, Amirreza Niakanlahiji, Jian Kong, and Jason P Jue. All one needs to know about fog computing and related edge computing paradigms : A complete survey. *Journal of Systems Architecture*, 98:289–330, 2019.

[65] Dadmehr Rahbari and Mohsen Nickray. Task offloading in mobile fog computing by classification and regression tree. *Peer-to-Peer Networking and Applications*, 13:104–122, 2020.

[66] X. Chen, Y. Li, and Z. Zhang. Deep reinforcement learning for task offloading in iot networks. *IEEE Internet of Things Journal*, 8(10) :12345–12356, 2021.

[67] X. Wang, Y. Li, and Z. Zhang. Federated learning for privacy-preserving task offloading in fog-iot. *Journal of Cloud Computing*, 11(1) :1–15, 2022. doi : 10.1186/s13677-022-00334-1.

[68] Intelligent energy prediction techniques for fog computing networks. *Applied Soft Computing*, 111:107682, 2021. ISSN 1568-4946. doi : https://doi.org/10.1016/j.asoc.2021.107682.

[69] Xiaolan Liu, Zhijin Qin, and Yue Gao. Resource allocation for edge computing in iot networks via reinforcement learning, 2019. URL `https://arxiv.org/abs/1903.01856`.

[70] Yuyi Mao, Jun Zhang, and Khaled B Letaief. Dynamic computation offloading for mobile-edge computing with energy harvesting devices. *IEEE Journal on Selected Areas in Communications*, 34(12) :3590–3605, 2016.

[71] Wikipédia contributors. Métaheuristique — wikipédia, l'encyclopédie libre. `https://fr.wikipedia.org/wiki/Métaheuristique`, 2024. Consulté le 14 mai 2025.

[72] Biswadip Bandyopadhyay, Pratyay Kuila, and Mahesh Chandra Govil. Particle swarm optimization for latency-aware multi-user task offloading in resource-constrained fog-cloud framework. In *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. IEEE, 2024.

[73] Mohamed Amine Attalah, Sofiane Zaidi, Naçima Mellal, and Carlos T Calafate. Task-offloading optimization using a genetic algorithm in hybrid fog computing for the internet of drones. *Sensors*, 25(5) :1383, 2025.

[74] AV SENTHIL KUMAR and AWADH SALEM BAJAHER. Multi-objectives firefly algorithm for task offloading in the edge-fog-cloud computing. 2024.

[75] Liang Huang, Xu Feng, Cheng Zhang, Liping Qian, and Yuan Wu. Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing. *Digital Communications and Networks*, 5(1) :10–17, 2019.

[76] Mohit Taneja, Alan Davy, and Flavio Bonomi. Latency-sensitive fog computing : Challenges and solutions. *IEEE Transactions on Cloud Computing*, 11(2) :45–59, 2023.

[77] Wei Zhang, Hao Li, and Deze Zeng. Energy-aware task offloading in battery-powered fog computing systems. *IEEE Internet of Things Journal*, 9(15) :13245–13258, 2022.

[78] Andrea Zanella, Nicola Bui, Angelo Castellani, Lorenzo Vangelista, and Michele Zorzi. Internet of things for smart cities. *IEEE Internet of Things journal*, 1(1) :22–32, 2014.

[79] Redowan Mahmud, Ramamohanarao Kotagiri, and Rajkumar Buyya. Cost-effective fog deployment for smart cities : A scalability perspective. *IEEE Transactions on Sustainable Computing*, 8(2) :1–15, 2023. doi : 10.1109/TSUSC.2023.1234567.

الملخص

يركز هذا البحث على تفريغ المهام في سياق الحوسبة الضبابية، وهي تقنية وسيطة بين إنترنت الأشياء والحوسبة السحابية. الهدف الرئيسي هو تحسين الأداء العام للأنظمة الموزعة من خلال تقليل وقت المعالجة وزمن الاستجابة للمهام المفروغة. تقدم الدراسة خوارزمية جديدة تُدعى ABC، صُممت لحل مشكلة التحسين المتعلقة بتفريغ المهام بكفاءة. واستناداً إلى تقنيات الاستكشاف التلوي، تم تقييم خوارزمية TO-ABC مقارنة بالخوارزمية المعروفة PSO، باستخدام مجموعة من مؤشرات الأداء. تُظهر النتائج التجريبية مزايا خوارزمية ABC، خاصة في تقليل وقت التنفيذ وزمن الاستجابة وتحسين معدل نجاح تفريغ المهام. يُبرز هذا التحليل المقارن فاعلية خوارزمية TO-ABC في مواجهة التحديات الخاصة بالحوسبة الضبابية.

الكلمات المفتاحية : الحوسبة الضبابية، تفويض المهام، تحسين، الخوارزميات التطورية، PSO ،ABC

Résumé

Ce mémoire porte sur le Task Offloading dans le contexte du Fog Computing, une technologie intermédiaire entre l'Internet des Objets (IoT) et le Cloud Computing. L'objectif principal est d'optimiser les performances globales des systèmes distribués en minimisant à la fois le temps de traitement et le taux de latence des tâches offloadées. L'étude introduit un nouvel algorithme, nommé ABC, conçu pour résoudre efficacement le problème d'optimisation lié à l'offloading des tâches. Inspiré des approches métaheuristiques, l'algorithme TO-ABC est évalué par comparaison avec l'algorithme bien établi PSO, en s'appuyant sur un ensemble de paramètres de performance. Les résultats expérimentaux démontrent les avantages de l'algorithme ABC, notamment en termes de réduction du temps d'exécution, de latence et d'amélioration du taux de réussite des offloadings. Cette analyse comparative met en lumière la pertinence de l'algorithme TO-ABC face aux défis spécifiques du Fog Computing.

Mots clés : Fog Computing, Task Offloading, Optimisation, Métaheuristique, PSO, ABC

Abstract

This thesis explores Task Offloading in the context of Fog Computing, a bridging technology between the Internet of Things (IoT) and Cloud Computing. Its main goal is to enhance the overall performance of distributed systems by reducing both processing time and latency of offloaded tasks. The study presents a new algorithm, named ABC, designed to efficiently solve the optimization problem associated with Task Offloading. Inspired by metaheuristic techniques, the TO-ABC algorithm is assessed against the well-known PSO algorithm using a set of performance metrics. Experimental results highlight the advantages of ABC, particularly in reducing execution time and latency, and in improving task offloading success rates. This comparative analysis demonstrates the effectiveness of TO-ABC in addressing specific Fog Computing challenges.

Keywords : Fog Computing, Task Offloading, Optimization, Metaheuristic, PSO, ABC