UNIVERSITÉ DU QUÉBEC À MONTRÉAL

ÉTUDE D'UNE PLATEFORME DE SURVEILLANCE MÉDICALE À DISTANCE RÉSISTANTE AUX PANNES

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

SERIGNE MBACKÉ GUEYE

DÉCEMBRE 2020

REMERCIEMENTS

Au terme de ce travail, je tiens à exprimer ma profonde gratitude et mes sincères remerciements à :

- ma Directrice de recherche, Pr Halima ELBIAZE pour tout le temps qu'elle m'a consacré, la qualité de son suivi, ses orientations et son soutien financier durant toute la période de ma formation ;
- M. Mouhamad DIEYE qui n'a ménagé aucun effort pour la réalisation de ce travail. Ces orientations et son apport ont été d'une plus-value importante ;
- tous les membres du jury par leurs remarques et propositions pour l'amélioration de notre travail ;
- tous mes collègues au sein du laboratoire TRIME avec qui j'ai passé de très bons moments d'échange et d'entraide ;
- tout le cadre professoral et administratif de l'UQAM ;
- toute personne qui a contribué de prés ou de loin à la réalisation de ce travail ;
- toute ma famille au sens africain du terme ;

Mes remerciements vont également à l'endroit du Pr Alassane DIOP, de par son soutien et ses encouragements.

Serigne Mbacke GUEYE

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ABRÉVIATIONS, DES SIGLES ET DES ACRONYMES

CCPM	Centre de Contrôle et de Prévention des Maladies
CGC	Centre de Gestion de la Crise
EA	Algorithmes évolutifs
ECG	électrocardiogramme
F-RHM	*Fog-based Remote Health Monitoring*
FC	*fog computing*
GAP	*Generalized Assignment Problem*
K-NN	k-Nearest Neighbors
MV	machine virtuelle
OMS	Organisation mondiale de la santé
QoS	*Quality of Service*
RAM	*Random Access Memory*
TP	taux de panne
TPA	Taux de Panne Annuel
TR	taux de réparation
TS	taux de service
UCT	*Unité Centrale de Traitement*

RÉSUMÉ

Ces dernières années, l'Internet des objets s'est largement incrusté dans le domaine de la santé, plus particulièrement dans la surveillance médicale à distance. Cette dernière, devenue primordiale dans le contexte de la COVID-19, produit une quantité massive de données à traiter avec comme objectif une prise de décision médicale optimale dans un délai restreint. Malheureusement, l'infonuagique se trouve incapable de répondre à ce nouveau type de besoin, car ses centres de données sont caractérisés par un délai de réponse très élevé. Dans ce contexte, l'informatique géodistribuée (fog computing) est récemment proposée dans la littérature pour une prise en charge des tâches sensibles au délai.

Dans le contexte de la pandémie de la COVID-19, la seule solution existante, pour le moment, pour limiter la transmission communautaire du virus est préventive (confinement et gestes barrières). Ainsi, l'aplatissement de la courbe est la politique épidémiologique de santé publique adoptée pour une meilleure prise en charge des malades. Cependant, l'application stricte de cette politique est difficile en pratique. En plus le virus peut être transmis par des personnes asymptomatiques ou présymptomatiques.

C'est dans ce cadre que nous préconisons F-RHM (*Fog-based Remote Health Monitoring*), une plateforme de surveillance médicale à distance, déployée dans le *fog computing* où les noeuds sont d'une fiabilité variable. F-RHM est capable d'identifier et d'estimer la disponibilité des nœuds afin de proposer un meilleur placement des conteneurs avec une prise en charge des exigences de disponibilité et de temps de réponse.

Ce problème de placement de conteneurs avec prise en charge des exigences de qualité de service est formulé selon un problème d'optimisation dont l'objectif est de maximiser le nombre d'applications à déployer qui satisfont aux exigences de délai et de disponibilité des requêtes des patients, sous plusieurs contraintes. Le problème est prouvé NP-difficile. Nous avons proposé une heuristique basée sur l'algorithme *Differential Evolution* (DE), nommée DE-Adapté pour la résolution du problème étudié.

Notre solution est comparée avec une stratégie gloutonne et une stratégie aléatoire. Sur plusieurs scénarios de simulation, les résultats ont montré que DE-adapté

donne toujours les meilleures performances en termes de violation des exigences de QoS.

Mots clés : Remote Health Monitoring, Fog Computing, Disponibilité, Evolution Différentielle, COVID-19

CHAPITRE I

INTRODUCTION GÉNÉRALE

1.1 Contexte

L'Internet des objets (IoT (Internet of Things), en anglais) s'est incrusté de façon omniprésente dans plusieurs domaines et activités de notre vie quotidienne (Guillemin *et al.*, 2009; Rahmani *et al.*, 2018). Le domaine de la santé n'est pas en reste, à travers surtout les applications de surveillance médicale à distance (en anglais *Remote Health Monitoring*). Ces applications sont devenues particulièrement prééminentes étant donné le contexte actuel de pandémie COVID-19.

Effectivement, tirant profit de capteurs dédiés à la santé, une plateforme de surveillance médicale à distance vise à optimiser les interactions entre différentes entités d'un système de santé (patients, médecins, capteurs, actionneurs, médicaments, etc.) afin d'améliorer la qualité des soins prodigués à travers l'automatisation de certaines tâches. En outre, ces plateformes visent à réduire le risque d'infections ainsi que les coûts liés à la rémunération des médecins (Kraemer *et al.*, 2017; Mandellos *et al.*, 2009; Yan *et al.*, 2015).

Bien évidemment, un déploiement massif de capteurs(ex. capturer le rythme cardiaque, l'enregistrement de la toux, la pression artérielle, le taux d'oxygène dans le sang, la température, etc.) aura pour conséquence immédiate d'engendrer un

volume exponentiel de données à traiter afin de prendre des décisions d'ordre médicales.

Dans un contexte où une décision doit être prise dans des délais très restreints, les centres de données infonuagiques qui sont typiquement prisés pour le traitement de données massif de par leur capacité de traitement et de stockage quasi infinie, ne sont pas adaptés étant donné le délai de réponse jugé trop élevé.

Dans ce cadre, l'informatique géodistribuée (en anglais le *fog computing*) a été récemment proposée dans la littérature pour les tâches sensibles au délai, notamment pour celles d'applications à but médicales (Kraemer *et al.*, 2017; Bonomi *et al.*, 2012a). C'est un paradigme récent qui vise à étendre l'infonuage à la périphérie du réseau pour relever les défis liés à la croissance exponentielle d'objets connectés à la périphérie du réseau. Effectivement, l'informatique géodistribuée s'appuie sur des ressources réseau disponibles à la périphérie du réseau plus connu sous le nom de noeuds périphériques (en anglais, fog nodes) pour fournir des services avec des temps de réponse réduits. Ces appareils, bien que caractérisés par une capacité de calcul et de stockage limitée, bénéficient d'une proximité aux utilisateurs finaux qui permet de réduire les délais de transmission pour des tâches d'analyse sensible aux délais (He *et al.*, 2017).

1.2 Motivation

Le monde est actuellement confronté à la pandémie de la COVID-19. En l'espace de quelques mois, nous sommes à plus de 22 millions de cas confirmés avec plus de 800 000 décès à travers le monde, selon (Remuzzi et Giuseppe, 2020). En l'absence d'un vaccin, des mesures préventives ont été préconisées pour réduire la transmission communautaire (Cascella *et al.*, 2020).

Il n'empêche que le taux d'infection continue de progresser rapidement dans cer-

tains pays, ce qui a engendré une saturation des systèmes de santé. Cela s'est traduit notamment par des restrictions d'accès aux salles d'urgence, une pénurie de kits de test et d'équipements médicaux de base comme les masques respiratoires, les équipements de protection individuelle, etc. D'autant plus que ces pénuries sont pointées du doigt comme étant des facteurs d'aggravation du taux de mortalité dans certains pays.

Les mesures de confinement et des gestes barrières (distanciation physique, lavage régulier des mains, etc.) sont devenus la norme pour faire face à cette pandémie. Cette pratique, plus connue en épidémiologie sous le nom d'aplatissement de la courbe, sert à ralentir la vitesse de contamination de la population (Jardine *et al.*, 2020), ce qui permet d'avoir suffisamment de ressources médicales (lits d'hôpital, médecins, infirmières, fournitures, etc.) afin de mieux prendre en charge les malades à traiter.

L'application stricte de telles mesures est cependant difficile à appliquer en pratique. D'une part, il est noté que les symptômes de la COVID-19 sont très similaires à ceux du rhume ou de la grippe courante. D'autre part, des travaux récents indiquent des cas de transmission à travers des personnes asymptomatiques ou présymptomatiques (Rothan et Byrareddy, 2020; Mizumoto *et al.*, 2020).

Dans ce cadre, en supplément des efforts de santé publique susmentionnés, nous préconisons une plateforme de surveillance médicale à distance, déployée dans l'informatique géodistribuée, capable de collecter, transférer et analyser en temps réel les données physiologiques provenant de capteurs portés par les personnes confinées. Nous nommons cette plateforme F-RHM (*Fog-Remote Health Monitoring*). Son objectif est :

- d'identifier et émettre des alarmes d'urgences, surtout au niveau des résidences des personnes âgées ;

- de détecter les transmissions communautaires ;
- de gérer adéquatement la planification et le déploiement des ressources rares (respirateurs, lits, trousses d'analyse, etc.) ;
- de réduire la charge du personnel soignant et les coûts des soins sanitaires liés aux hospitalisations précoces ;
- de traquer la non-observation des mesures de confinement ;
- d'éviter aux populations de faire des déplacements inutiles à l'hôpital pour un examen médical, aux risques d'exposition et/ou de transmission de la maladie.

1.3 Problématique

Afin de pouvoir servir comme support à la prise de décision médicale dans le contexte de la COVID-19 et par conséquent aider à réduire le taux d'infection, nous identifions deux exigences majeures pour notre plateforme de surveillance médicale à distance (F-RHM) : (1) minimiser le délai de réponse et (2) minimiser les pertes d'informations. Pour minimiser le temps de réponse, il convient de bien choisir le(s) nœud(s) chargé(s) de traiter les tâches d'analyses soumises à travers une requête. La minimisation des pertes d'informations passe par une évaluation adéquate de la fiabilité des nœuds avant de les sélectionner.

Ayant opté de déployer F-RHM sur l'informatique géodistribuée, il est à noter qu'elle offre plusieurs bénéfices :

- sa fonctionnalité de localisation peut être exploitée afin de surveiller les déplacements des personnes dans l'optique de détecter les transmissions communautaires ;

- une prise en compte de l'emplacement géographique des noeuds fog, lors du placement de services applicatifs ;

- en vertu du caractère confidentiel des données médicales, certaines données sensibles peuvent être traitées au niveau des noeuds fog au lieu des noeuds du cloud qui sont hors du contrôle des utilisateurs (Vaquero et Rodero-Merino, 2014; Kraemer *et al.*, 2017).

Malgré ces avantages, plusieurs défis sont à noter. Parmi les défis notables figurent ceux liés à l'hétérogénéité et à la fiabilité des noeuds périphériques (Madsen *et al.*, 2013). Contrairement à l'infonuagique où d'amples mécanismes de tolérance aux pannes existent, assurer la fiabilité et la disponibilité dans l'informatique géodistribuée reste toujours un problème ouvert (Hu *et al.*, 2017; Yi *et al.*, 2015; Madsen *et al.*, 2013; Mavrogiorgou *et al.*, 2018).

Ceci a de profondes implications sur la satisfaction des exigences de QoS (*Quality of service)*, particulièrement pour les systèmes de surveillance médicale à distance où le retard ou la perte d'informations peut engendrer des conséquences néfastes (Kraemer *et al.*, 2017; Dhanvijay et Patil, 2019).

A cet effet, la réplication est souvent utilisée comme mécanisme de tolérance aux pannes afin de garantir la continuité du service en cas d'indisponibilité d'un noeud. Elle est également utilisée pour répartir la charge de travail sur un ensemble de noeuds afin de réduire le temps de réponse global. Il est à noter cependant, du fait de l'hétérogénéité des noeuds périphériques en terme de pannes et de capacités de stockage, la sélection de l'emplacement des réplicas n'est pas évidente.

À travers les points énumérés ci-haut, il est clair que fournir des services de santé à distance en temps réel avec des exigences de QoS (temps de réponse et disponibilité) strictes reste toujours un défi pertinent.

1.4 Objectifs

Dans ce mémoire, nous étudions dans le contexte de la COVID-19, la mise en place d'une plateforme de surveillance médicale à distance déployée sur l'informatique géodistribuée où les noeuds sont d'une fiabilité variable. Dans ce cadre, nous proposons une plateforme capable d'identifier et d'estimer la disponibilité des noeuds du fog afin de proposer un meilleur placement des conteneurs avec une prise en charge de leurs exigences en disponibilité et en temps de réponse. C'est dans cette optique que nous proposons une heuristique basée sur l'algorithme d'évolution différentielle qui sera comparé aux stratégies gloutonne et aléatoire.

Ainsi, dans un contexte où la plus grande préoccupation des autorités médicales demeure la limitation de la transmission du virus et une bonne gestion des ressources de soins sanitaires rares, telles que les lits d'hôpital, les masques, les respirateurs et les équipements de protection individuelle pour le personnel médical, notre objectif principal vise à permettre aux responsables de la santé de tirer parti d'une plateforme de surveillance médicale à distance fiable qui permettrait de suivre à distance les patients présentant des symptômes légers et d'interner dans les hôpitaux que les patients en situation critique.

1.5 Contributions du mémoire

Nous identifions les contributions suivantes dans ce mémoire :

- la description d'un cas d'utilisation pertinent à la lutte contre la pandémie de la COVID-19 qui sévit présentement dans le monde entier et qui a globalement submergé les systèmes de santé ;
- une formulation mathématique du problème de fiabilité de la surveillance

médicale à distance basée sur le *Fog Computing* par un problème d'optimisation dont l'objectif est de maximiser le nombre d'applications à déployer qui satisfont aux exigences de disponibilité et de latence des requêtes des patients, sous plusieurs contraintes que nous allons détailler dans la partie correspondante. Un tel problème est prouvé étant NP-difficile ;

- la proposition d'une heuristique basée sur l'algorithme *Differential Evolution*(DE) que nous nommons DE-Adapté ;
- une évaluation des performances de nôtre solution DE-adapté.

1.6 Plan du mémoire

Nous organisons ce mémoire comme suit :

Dans le chapitre 2, nous allons présenter le concept du *fog computing*, son architecture, ses caractéristiques, ses avantages et son fonctionnement. Nous allons aussi, dans ce chapitre, parler de la virtualisation à base de conteneurs (la conteneurisation) avant de finir par une étude sur la disponibilité d'un système informatique. Dans le chapitre 3, nous allons présenté une revue de la littérature en faisant le point, d'abord, sur les travaux qui portent sur les systèmes de surveillance sanitaire à distance basés sur le *fog computing* et ensuite sur ceux qui se sont orientés sur la tolérance aux pannes au niveau des plates-formes *fog computing*. Le chapitre 4 sera consacré à l'étude de notre plateforme proposée, dans lequel nous commençons par la description de notre modèle de système, ensuite une formulation mathématique du problème, puis l'heuristique proposée. Nous allons évaluer les performances de notre solution dans le chapitre 5. Une conclusion générale mettra un terme à ce mémoire.

CHAPITRE II

GÉNÉRALITÉS

Le but de ce chapitre est de fournir une description détaillée des technologies et terminologies de recherche qui seront utilisées dans le contexte de la collecte et du traitement des données de l'IoT. Nous allons aussi donner l'expression de la disponibilité d'un système. Ainsi, nous allons présenter le concept du *fog computing* (informatique géodistribuée ou informatique en brouillard), son architecture, ses caractéristiques et ses avantages avant de faire le point sur son fonctionnement. Nous allons aussi, dans ce chapitre, parler de la virtualisation à base de conteneurs. Nous terminons ce chapitre par une étude sur la disponibilité d'un système informatique.

2.1 Fog Computing

Un grand nombre d'objets connectés liés à l'avènement de l'internet des objets, génère un gros volume de données qui doit être envoyé dans les centres de données de l'infonuagique (*cloud computing*) pour traitement. Cisco a estimé qu'en 2020, 50 milliards d'objets seraient connectés à internet générant un trafic IP mondial de 15,3 Zo de données (Cisco, 2016b; Kobusińska *et al.*, 2018).

Avec ces données massives, l'utilisation du paradigme de *cloud computing* entraînera un délai élevé et une congestion du réseau d'infrastructure. Sur cette base, le

fog computing a été proposé (Bonomi *et al*., 2012b) afin de faire face à ces limites du cloud. Ainsi, les noeuds fog sont dotés de puissances de calcul et de stockage pour une meilleure prise en charge des applications en temps réel (les applications sensibles au délai) (Bonomi *et al*., 2014).

Le *Fog Computing* (appelé aussi informatique géodistribuée ou informatique en brouillard) est une plateforme décentralisée de traitement de données hautement virtualisé qui fournit une couche intermédiaire de calcul, de stockage et de mise en réseau entre les appareils des utilisateurs finaux et les centres de données du Cloud Computing (Cisco, 2016a).

2.1.1 Architecture du fog

L'architecture du *fog computing* est composée de trois couches (Cisco, 2016a; Bourhim, 2020), comme montré dans la figure 2.1 :

- la couche IoT/Utilisateur : Cette couche comprend tous les objets connectés tels que les appareils mobiles, les capteurs, les caméras de surveillance, les véhicules, etc. Ces dispositifs génèrent une variété importante de données qui sont traitées directement sur l'objet en question ou transmises à un noeud de la couche *fog* (Cisco, 2016a) ;
- la couche fog : Cette couche est composée d'un certain nombre de noeuds (*Fog Node* en anglais) dotés de puissance de calcul et de stockage et proches des objets connectés. Ces noeuds (routeurs, stations de base, ordinateurs, etc.) reçoivent les données de la couche précédente pour traitement et analyse, ou les acheminent au *cloud* si nécessaire ;
- la couche cloud : Elle est composée de l'ensemble des serveurs des centres de données du *cloud computing*.

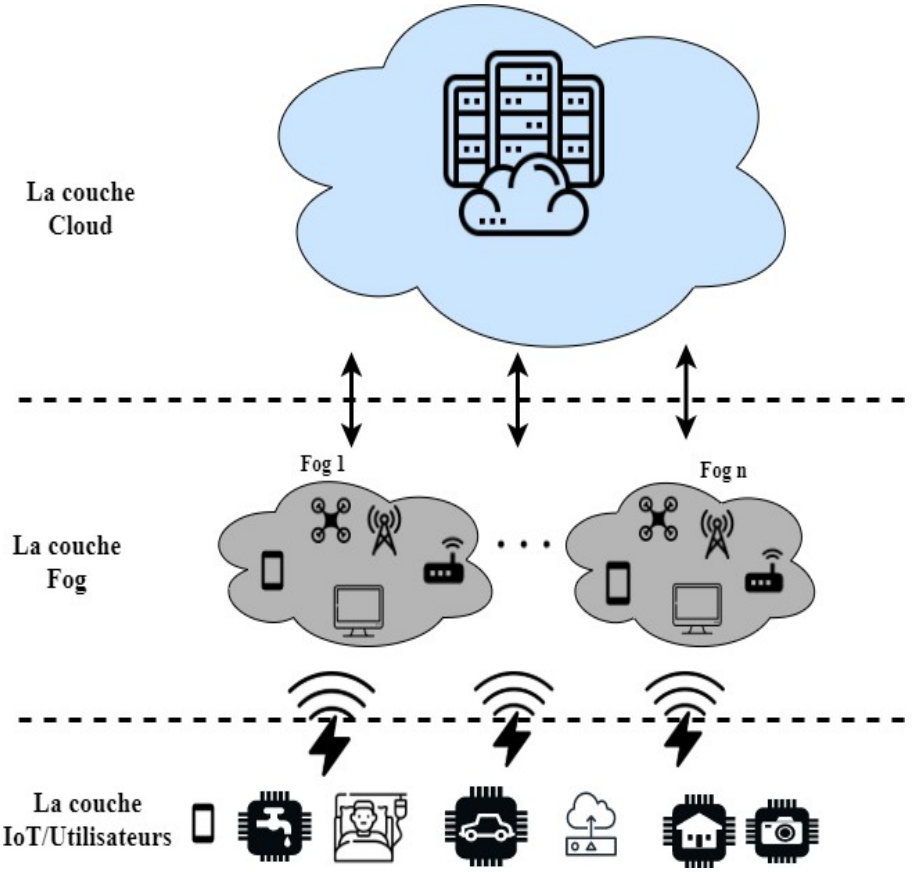

Figure 2.1 Architecture du Fog Computing

2.1.2 Caractéristiques et avantages

Selon les travaux de (Bonomi *et al.*, 2012a; Bourhim, 2020; Brogi *et al.*, 2020; Bonomi *et al.*, 2014; Zhu *et al.*, 2013), les caractéristiques du *fog computing* se déclinent ainsi :

- la mise en oeuvre d'un grand nombre de capteurs intelligents ou d'objets connectés. Le *fog computing* supporte des réseaux de capteurs à grande échelle qui surveillent leur environnement ;

- l'utilisation du Cloud Computing pour le traitement des tâches non sensibles au délai et qui nécessitent une très grande capacité de calcul et de stockage ;
- des volumes de données importantes et variées à traiter ;
- une réduction de la latence des communications dans le réseau : Les noeuds de la couche fog sont plus proches des utilisateurs finaux (en réseau local généralement) ;
- une large distribution géographique des objets connectés et la prise en compte de leur position : Contrairement au *cloud computing* où le calcul et le stockage sont centralisés au niveau des centres de données spécialisés, les applications et les services fournis par le *fog computing* sont distribués et peuvent être déployés n'importe où. Le *fog computing* prend aussi en charge la connaissance de l'emplacement des nœuds ;
- une hétérogénéité des équipements ;
- l'utilisation de réseaux sans-fils et d'équipements mobiles ;
- une prise en charge de la mobilité. Les applications qui tournent dans des environnements de fog computing ont la possibilité de se connecter directement sur des périphériques mobiles et d'activer des méthodes de mobilité, telles que le protocole LISP (*Locator ID Separation Protocol*) (Meyer, 2008) ;
- un meilleur contrôle des équipements, des flux d'informations et des services déployés.

Conformément à toutes ces caractéristiques précitées, le fog computing présente les avantages suivants (Cisco, 2015) :

- réduction de la congestion au niveau du réseau d'infrastructure (consommation réduite de la bande passante) et de la latence engendrant ainsi une amélioration des temps de réponse des requêtes des utilisateurs finaux ;
- élimination des goulots d'étranglement résultant de systèmes informatiques centralisés du Cloud ;
- amélioration de la sécurité des données plus proches de l'utilisateur final en réduisant leur exposition ;
- meilleure évolutivité découlant de systèmes virtualisés ;

2.1.3 Fonctionnement

Les applications IoT sont déployées sur les nœuds de la couche fog. Les données issues des objets connectés sont ensuite absorbées par les noeuds les plus proches avant d'être redirigées vers le lieu le plus optimal (le noeud en question, un autre autre noeud du même fog, un noeud d'un autre fog ou le cloud) pour analyse et traitement (Bonomi *et al.*, 2012a; Cisco, 2016a). En effet, les requêtes sensibles au délai sont traitées au niveau des nœuds les plus proches de l'objet qui les a générées. Les données qui peuvent attendre des secondes ou des minutes sont transmises à un nœud d'agrégation pour une meilleure analyse et traitement. Les données qui ne sont pas sensibles au délai sont envoyées au niveau des serveurs des centres de données du cloud pour une analyse massive, un stockage à long terme ou pour des besoins d'historiques.

Le *fog computing*, combinaison entre les technologies du *cloud computing* et de l'Internet des objets (IoT), ouvre de nouvelles opportunités aux fournisseurs de services cloud. En effet, cette variété importante d'objets connectés a engendré une nouvelle manière de conception de systèmes d'information et de communication.

En conséquence, les fournisseurs de service cloud, pour faire face à ce nouveau besoin, fournissent un nouveau type de service IoTaaS (*IoT as a Service*). Dans ce contexte, pour améliorer l'approvisionnement et le déploiement des services applicatifs IoT, la virtualisation par conteneurs est devenue une alternative à la virtualisation classique qui se base sur les machines virtuelles. Dans ce qui suit nous allons faire une étude détaillée de cette nouvelle forme de virtualisation (la conteneurisation) afin de montrer en quoi elle est plus avantageuse que les VMs. Nous allons terminer cette partie par une vue d'ensemble sur les conteneurs docker.

2.2 Virtualisation à base de conteneur

2.2.1 Aperçu général

Connu également sous le nom de conteneurisation, la virtualisation à base de conteneurs est une approche de virtualisation légère qui crée un environnement virtuel au niveau application à l'intérieur de la machine hôte. Elle est également appelée virtualisation au niveau du système d'exploitation (Soltesz *et al.*, 2007).

Elle supprime la charge supplémentaire liée à l'utilisation d'hyperviseur de machines virtuelles (MV) en créant des conteneurs. Contrairement aux MVs qui ont leur propre système d'exploitation, les conteneurs agissent comme des systèmes invités en partageant les ressources du système d'exploitation de la machine hôte (Celesti *et al.*, 2016).

La conteneurisation fournit différents niveaux d'abstraction dans lesquels les conteneurs se partagent un seul noyau. Chaque conteneur peut exécuter plus d'une application. Ainsi, le système peut être plus efficace en termes d'utilisation des ressource, car il n'y a plus de système d'exploitation, très gourmand en stockage, pour chaque conteneur. Elle fournit également une isolation entre les applications

qui s'exécutent au sein d'un conteneur (Morabito *et al*., 2015).

Cependant la conteneurisation présente quelques inconvénients. En effet, le système d'exploitation Windows ne peut pas s'exécuter en tant qu'application conteneurisée sur une machine hôte Linux. En outre, les conteneurs ne fournissent pas une parfaite isolation des ressources, comparés aux MVs, du simple fait du noyau partagé qui représente une faille de sécurité (Morabito *et al*., 2015). Pour y faire face et fournir une parfaite isolation entre les applications d'un conteneur donné et entre conteneurs et la machine hôte, les *cgroups* et les *namespaces*, deux fonctionnalités du noyau, sont utilisées.

- Les *cgroups* (*control groups*) sont l'une des principales caractéristiques du noyau qui permettent aux utilisateurs d'allouer ou de limiter les ressources (UCT, RAM, stockage disque et la bande passante réseau) entre les groupes de processus. Ils fournissent une allocation efficace de ces dernières entre les conteneurs et les applications qui y sont exécutées. Ainsi, ils permettent une parfaite isolation, car chaque groupe est unique et identifié par les processus qui s'y exécutent (Rosen, 2013).

- les *namespaces* (espaces de nom) fournissent une isolation entre processus d'un conteneur. Ils permettent d'assurer à chaque processus d'un conteneur de disposer de son propre espace de nom dans lequel il pourra s'exécuter sans interférer avec d'autres processus d'un autre *namespace* (Rosen, 2013).

2.2.2 Avantages de la conteneurisation

Les travaux de (Joy, 2015; Diouf, 2019; Javed *et al*., 2016) ont décris les caractéristiques de la conteneurisation qui permettent de mettre en exergue les avantages des conteneurs par rapport aux machines virtuelles.

- La portabilité :
  Les conteneurs peuvent s'exécuter sur n'importe quel environnement sans une modification des fonctionnalités du système d'exploitation. Les applications d'un conteneur peuvent être regroupées et déployées dans des environnements divers.

- La rapidité :
  La création d'un nouveau conteneur se fait en quelques secondes à cause de leur légèreté. Les développeurs et administrateurs système peuvent facilement développer et déployer, en temps réduit,les applications conteneurisées dans l'environnement de production.

- L'évolutivité :
  Un conteneur peut être déployé et s'exécuter dans n'importe quel environnement (système Linux, plateforme cloud, centre de données, etc.). En plus la taille des conteneurs peut être ajustée de manière dynamique en fonction des besoins. Ainsi, ils conviennent plus aux applications évolutives.

- Réduction de la consommation des ressources :
  L'utilisation des conteneurs permet d'exécuter un nombre important d'applications sur une seule machine hôte. Ils sont tous exécutés sur le système d'exploitation de cette dernière. Les ressources de la machine hôte sont ainsi consommées de manière efficace.

2.2.3 Comparaison entre conteneurs et MVs

Le tableau 2.1 fait une comparaison entre les conteneurs et les MVs en fonctions des paramètres de performance suivants : temps de démarrage, la consommation des ressources, la communication et l'utilisation du système d'exploitation. Notre comparaison s'est basée sur les travaux de (Felter *et al.*, 2015; Diouf, 2019).

Paramètres	Machines virtuelles	Conteneurs
Temps de démarrage	Le démarrage d'une machine virtuelle prend plusieurs minutes.	Le démarrage d'un conteneur se fait en quelques secondes.
Stockage, UCT et RAM	Les MVs occupent une importante capacité de stockage du fait de leurs système d'exploitation qui sont très gourmande sur la consommation des ressources.	Contrairement aux MVs,ils ne sont pas gourmands en consommation des ressources.
Communication	S'ils sont exécutées sur des serveurs différents, la communication se fait grâce aux périphéries réseau. Au cas contraire, La communication inter-processus est utilisée	Idem que pour les MVs
Système d'exploitation	Chaque MV s'exécute sur un hyperviseur, avec son propre système d'exploitation, qui a son tour s'exécute sur une machine physique.	Un conteneur s'exécute sur le système d'exploitation (S.E) de la machine hôte sous forme de S.E invité. ils partagent le même noyau.

Tableau 2.1 Comparaison conteneurs et MVs

2.2.4 Les conteneurs Docker

Docker est une plateforme open source, implémentée en Go, un langage de programmation orientée service. Il permet de créer, déployer et exécuter des applications, sous forme de conteneur, de manière rapide par rapport aux autres solutions traditionnelles (Diouf, 2019; Matthias et Kane, 2015). Docker permet de séparer les applications (les conteneurs) de l'infrastructure qui va être gérée comme une application. A cet effet, le cycle de vie de l'écriture d'un code est considérablement diminué. Docker se base sur la notion de *cgroups* et de *namespaces* pour déployer et exécuter les conteneurs en toute sécurité et de manière isolée. Ainsi, plusieurs conteneurs peuvent être exécutés simultanément sur une même machine hôte sans risque d'interférence avec une consommation efficace des ressources matérielles (disque, RAM, CPU) de la machine (Matthias et Kane, 2015). Docker peut être exécuté sur n'importe quel système hôte exécutant un noyau compatible à jour.

Le processus de conteneurisation des applications via la plateforme Docker se fait de la manière suivante : les applications sont d'abord encapsulées dans des conteneurs Docker qui sont ensuite soumis à d'autres équipes de développeur pour des modifications ultérieures avant d'être déployés dans des environnements de production (centre de données local, cloud ...) selon une architecture client-serveur illustrée dans la figure 2.2 (Turnbull, 2014). Les éléments de cette architecture sont décrites dans ce qui suit :

- *Docker daemon*

  Il est exécuté sur la machine hôte et permet de gérer les conteneurs de cette dernière. Pour communiquer avec ce *daemon*, les utilisateurs passe par le docker client qui s'exécute sur la même machine hôte avec le même fichier binaire.

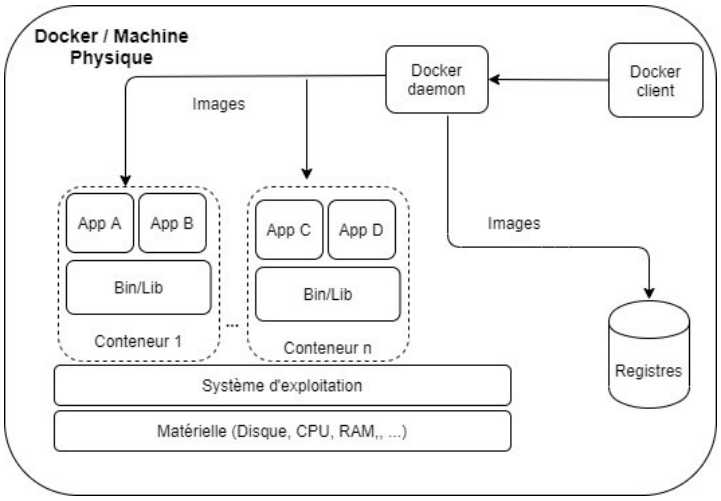

Figure 2.2 Architecture Docker

- *Docker client*

  C'est une interface en ligne de commande permettant aux utilisateurs de communiquer avec le *daemon*.

- *Docker images*

  Les images sont les principaux éléments constitutifs de Docker. Ce sont les codes sources utilisés pour la construction des conteneurs. Elles peuvent être créées suivant plusieurs instructions (commandes).

- Les registres

  Ils sont utilisés pour stocker les images créées. Pour donner un accès public à une image, cette dernière doit être sauvegardée dans Docker Hub qui est un registre public qui stocke un ensemble d'images déjà créées et téléchargeables. Toutefois, une image peut avoir un accès privé.

- Les conteneurs

  Ils sont créés à l'aide des images. Sur un conteneur peut s'exécuter un ou plusieurs services d'applications, mais ne contient qu'une seule image. Docker

peut créer, démarrer, arrêter, supprimer et déplacer des conteneurs.

2.3 La disponibilité d'un système

La disponibilité d'un système est définie comme étant la probabilité pour laquelle ce dernier soit opérationnel à chaque fois que son utilisation est requise (Elsayed, 1996). Elle est une importante métrique utilisée pour évaluer les performances des systèmes réparables en tenant compte de la fiabilité et de la maintenabilité de ses composants.

2.3.1 La fiabilité

La fiabilité d'un système est sa probabilité de fonctionner adéquatement sans tomber en panne durant une période de temps donnée (Elsayed, 1996). Le MTTF (*Mean Time To Fault*), la durée moyenne de fonctionnement d'une entité avant sa première panne, est l'indice de fiabilité d'un équipement. La fiabilité est généralement notée R(t) et se calcule de deux manières selon le type de système à considérer :

- **système en série**

  C'est un système constitué de plusieurs éléments pour lesquels les défaillances sont indépendantes et que la défaillance d'un élément entraîne celle du système. La fiabilité de ce genre de systèmes est exprimée comme suit :

$$R(t) = \prod_{i=1}^{n} R_i(t) \tag{2.1}$$

  avec $R_i(t)$ étant la fiabilité du i^{iem} élément du système à l'instant t.

- **système en parallèle**

C'est le contraire d'un système en série. En effet, la défaillance d'un élément n'entraîne pas celle du système. Dans un système parallèle, la fiabilité est exprimée selon l'expression suivante :

$$R(t) = 1 \quad \prod_{i=1}^{n} (1 \quad R_i(t)) \tag{2.2}$$

2.3.2 La maintenanbilité

La maintenanbilité d'un système représente sa capacité à réparer un équipement en panne en accord avec les conditions de fonctionnement dans un intervalle de temps donné (Elsayed, 1996). Elle caractérise la facilité à remettre ou de maintenir un équipement réparable dans un état de fonctionnement. Elle est souvent représenté par MTTR (*Mean Time To Repair*) qui représente la durée moyenne de réparation d'un équipement. Elle est exprimée comme suit :

$$MTTR = \sum_{i=1}^{n} \frac{T_i}{n} \tag{2.3}$$

avec T, le temps d'un arrêt et n le nombre d'arrêt.

Dans ce qui suit, nous allons présenter les différentes classifications de disponibilité afin de justifier celle que nous avons utilisée dans le cadre de notre travail en se basant sur les travaux de (Elsayed, 1996; Dieye, 2016).

1. La disponibilité instantanée

   La disponibilité instantanée est la probabilité pour qu'un système (ou un composant) soit opérationnel à un instant t. Elle est généralement représentée par $A(t)$. Elle est très similaire à la fiabilité dans la mesure où elle donne la probabilité pour laquelle le système soit fonctionnel à un instant t. Les informations de maintenabilité sont aussi intégrées dans le calcul de

$A(t)$. Ainsi, pour que le système soit fonctionnel à un instant t, il y'a deux possibilités :

(a) Soit le système est fiable durant toute la période $[0, t]$. Cette fiabilité est représentée par la probabilité R(t).

(b) Soit le système fonctionne correctement depuis la dernière réparation à l'instant t', tel que $0 < t' < t$. Cette condition est exprimée par la probabilité suivante :

$$\int_0^t R(t - t')m(t')\,dt'$$

$m(t')$ représente une fonction de densité du processus de renouvellement du système. Un processus de renouvellement est une approche habituellement utilisée pour modéliser la disponibilité d'un système (Dieye, 2016; Elsayed, 1996). Il a pour but de décrire la fréquence des pannes du système.

Ainsi, La disponibilité instantanée A(t) sera donnée par la somme de ces deux précédentes probabilités.

$$A(t) = R(t) + \int_0^t R(t - t')m(t')\,dt' \qquad (2.4$$

2. La disponibilité moyenne ($\overline{A(t)}$)

La disponibilité moyenne d'un système est la période de temps durant laquelle ce dernier reste disponible pour utilisation. Elle représente la valeur moyenne de la disponibilité instantanée A(t) sur un intervalle de temps. Par exemple, sur une période $[0, t]$,

$$\overline{A(t)} = \frac{1}{t}\int_0^t A(t')\,dt' \qquad (2.5)$$

3. La disponibilité à l'état stable ($A(1)$)

Connue également sous le nom de disponibilité à long terme ou disponibilité asymptotique, la disponibilité à l'état stable d'un système est la limite de la disponibilité instantanée A(t) lorsque t tend vers l'infini. Elle est généralement représentée par l'équation suivante :

$$A(\infty) = \lim_{t \to +\infty} A(t) \tag{2.6}$$

4. La disponibilité inhérente (A_I)

La disponibilité inhérente (A_I) représente la disponibilité à l'état stable ($A(\infty)$) qui ne considère que le temps d'arrêt de maintenance curative des équipements du système. Elle exclut toute autre cause de retard. Elle est exprimée selon l'équation suivante :

$$A_I = \frac{MTTF}{\text{MTTF} + \text{MTTR}} \tag{2.7}$$

Avec $MTTF = \frac{1}{\lambda}$ (λ représente le taux de panne) et $MTTR = \frac{1}{\mu}$ (μ représente le taux de réparation)

Cette formulation de la disponibilité inhérente considère le taux de panne d'un équipement comme étant une constante. Elle a fait l'objet de plusieurs critiques (Schroeder *et al.*, 2010; Gibson, 2007). En réalité, d'après (Gibson, 2007) ce taux de panne d'un équipement varie dans le temps durant le cycle de vie du système.

5. La disponibilité atteinte (A_A)

Elle est un peu identique à la disponibilité inhérente. En effet, la disponibilité atteinte (A_A), en plus du temps de maintenance curative, prend seulement en compte le temps de maintenance préventive. Les autres retards ne sont pas inclus. A_A est calculée en fonction du temps moyen entre les actions de maintenance MTBM (*Mean Time Between Maintenence*) et le temps d'arrêt

moyen de maintenance $\overline{M}$. Elle est exprimée comme suit :

$$A_A = \frac{MTBM}{MTBM + \overline{M}} \tag{2.8}$$

6. La disponibilité opérationnelle (Ao)

   Elle est la mesure de la disponibilité moyenne réelle du système sur une période de temps en incluant toutes les sources de retard. Elle est essentiellement basée sur les évènements réels qui se sont passés au système. Ainsi, elle ne peut pas être contrôlée par le fabricant.La disponibilité opérationnelle (Ao) est le rapport entre la disponibilité du système ($A(t)$) et la durée du cycle de vie du système. Elle est représentée par l'expression suivante :

$$Ao = \frac{\text{disponibilité du système}}{\text{durée du cycle de vie du système}} \tag{2.9}$$

La conteneurisation et le *fog computing* sont deux nouveaux paradigmes qui ont grandement amélioré les performances des systèmes informatiques en terme de consommation de ressources matérielles et de temps de réponse liée aux traitements des requêtes des utilisateurs (Skarlat *et al.*, 2016). Grâce à Docker, le déploiement d'applications conteneurisées sur les noeuds de la couche fog est devenu possible et avantageux aux applications IoT (L'e-santé [1] par exemple). Cependant, plusieurs défis liés aux exigences de QoS sont à relever par les fournisseurs de services de *fog computing*, surtout dans le domaine du e santé où les exigences de disponibilité et de délai doivent être satisfaites (Yi *et al.*, 2015). Dans ce qui suit, nous allons faire la revue de la littérature des précédentes contributions liées au e santé basée sur le *fog computing* et des travaux qui se sont sont focalisés sur sur la tolérance aux pannes dans le fog.

---

1. wikipédia : L'e-santé ou santé numérique ou information numérique sur la santé recouvre les domaines de la santé qui font intervenir les technologies de l'information et de la communication.

CHAPITRE III

REVUE DE LA LITTÉRATURE

Dans ce chapitre, nous faisons une revue de la littérature présentant l'ensemble des travaux qui sont liés à ce travail. Nous commencerons par récapituler les contributions qui ont été faites sur la surveillance médicale à distance déployée sur l'informatique géodistribuée, avant de finir par celles qui se sont focalisées sur la tolérance aux pannes liée au délai et à la disponibilité dans les plateformes de *fog computing*.

3.1 Surveillance médicale à distance déployée sur *fog computing*

L'avènement de l'IoT a conduit à une nouvelle forme de soins sanitaire (en anglais, *Healthcare-the Internet of Thing* (H-IoT)). L'émergence du H-IoT, plus particulièrement la surveillance de patients à distance, est une continuité du modèle de soins centré sur la donnée dont le but est de permettre une accessibilité efficace et continue à des soins de qualité à faible coût (Kraemer *et al.*, 2017; Mandellos *et al.*, 2009).

Les auteurs des articles (Dhanvijay et Patil, 2019; Kraemer *et al.*, 2017) mettent en évidence l'impact positif et immédiat des systèmes efficaces de surveillance médicale à distance. Selon eux, de tels systèmes permettent de réduire la charge du personnel soignant, ce qui permettra d'avoir de meilleurs soins en automatisant

certaines tâches. Ils permettent aussi de garder les patients qui ne sont pas en situation critique, chez eux aussi longtemps que possible.

(Bertini *et al.*, 2016), quant à eux, ont détaillé les avantages de surveiller les patients à distance au lieu de les interner dans les hôpitaux, notant un impact positif sur la survie des patients.

De la même manière, (Topol, 2012) a souligné que l'utilisation de capteurs portés par des patients permet aux autorités sanitaires de capturer un flux de données biométriques précieuses qui faciliteront, grâce à des techniques analytiques, les détections précoces, les diagnostics et l'exploration de nouvelles voies de traitement.

La plupart des travaux existants dans la littérature proposent une analyse des données des patients afin de déclencher des alarmes lorsque des situations critiques se présentent. Par exemple, (Cao *et al.*, 2015) proposent un mécanisme de détection de chutes de personnes en utilisant une analyse locale de l'accélération des valeurs d'amplitude. Une combinaison de techniques de filtrage et d'analyse de séries chronologiques non linéaires est également utilisée pour détecter le taux de faux positifs (probabilité de rejeter faussement l'hypothèse nulle). Leur système repose sur le *fog computing* pour distribuer les tâches analytiques sur le réseau en répartissant la tâche de détection entre les nœuds fog et les serveurs cloud.

(Craciunescu *et al.*, 2015) proposent un mécanisme de surveillance à distance qui inclut la détection des chutes grâce à une analyse en temps réel des données issues des capteurs environnementaux et ceux attachés sur les patients.

(Aazam et Huh, 2015) ont proposés un service d'alerte d'urgence nommée E-HAMC qui est déployé à des fins de déchargement et de prétraitement de tâches sur le réseau fog. Il envoie, en cas d'urgence, le lieu de l'incident et établit automatiquement le contact avec le service d'urgence approprié.

(Gia *et al.*, 2015; Chen et Liu, 2016; Granados *et al.*, 2014) ont étudié l'extraction de fonctionnalités électrocardiogramme (ECG) visant à faciliter le diagnostic des maladies cardiaques, en s'appuyant sur un mécanisme de transformation en ondelettes légères.

Dans un contexte plus large, (López *et al.*, 2010; Huang *et al.*, 2009) ont proposé un système de surveillance à distance des signes vitaux des patients en capturant et en traitant leurs données d'activité et de localisation.

Une autre facette explorée consiste à contrôler à distance et avec précision les capteurs d'action connectés aux patients. Par exemple, (Masip-Bruin *et al.*, 2016) ont présenté un scénario dans lequel, en tenant compte des données environnementales, le niveau d'oxygène des patients est collecté et analysé. Sur la base de cette analyse, la dose d'oxygène du patient est ajustée de manière adéquate en temps réel.

Pour plus d'efficacité, tous les systèmes proposés ci-dessus doivent être tolérants aux pannes afin d'assurer une continuité du service pour de meilleures performances. Dans ce qui suit, nous allons essayer de faire le tour des propositions qui ont été faites dans ce sens.

3.2 La tolérance aux pannes dans le Fog

Caractérisés par l'hétérogénéité de leurs *fog nodes* en terme de pannes, les environnements de *fog computing* doivent obligatoirement être tolérants aux pannes afin de pouvoir fournir des services fiables. Sur ce, plusieurs méthodes y remédiant ont été proposées dans la littérature. Ces méthodes sont classées en deux groupes : celles dites réactives et celles dites proactives (Alarifi *et al.*, 2019). Les méthodes réactives permettent de prendre en charge la panne d'un noeud par une quelconque solution une fois que cette défaillance s'est produite. La réplication

et les points de contrôle (*checkpoints)* [1] en sont les plus utilisées (Zhang *et al.*, 2018). Les méthodes proactives quant à elles sont des mesures préventives prises lors du placement du service afin d'éviter de le déployer sur un noeud qui a une probabilité élevée de tomber en panne.

Beaucoup de solutions basées sur les méthodes réactives ont été proposées dans la littérature afin de garantir la fiabilité au niveau du *cloud computing* : (Goiri *et al.*, 2010) ont proposé une méthode basée sur les *checkpoints*. Leur proposition minimise le temps de stockage nécessaire des *checkpoints*. Pour ce faire, ils enregistrent uniquement les valeurs des paramètres modifiés. (Abd Latiff *et al.*, 2017) ont proposé un algorithme de déploiement qui prend en compte le *checkpoint* et la migration pour faire face aux pannes. (Das et Khilar, 2013) ont proposé un schéma de réplication pour améliorer la fiabilité du système en évitant d'allouer des tâches aux noeuds qui ont une probabilité d'échec élevé. Une méthode de tolérance aux pannes qui, en fonction de la priorité des tâches, sélectionne entre la resoumission et la réplication est proposée par (Saranya *et al.*, 2015).

(Souza *et al.*, 2017) ont proposé une solution basée sur la programmation linéaire afin d'évaluer l'utilisation des méthodes réactives et proactives dans le fog. Ils se fixent comme objectif d'observer l'impact de chacune sur le temps d'allocation de services, le délai de réparation et l'utilisation des ressources. Ils ont modélisé le problème comme un problème de sac à dos multidimensionnel. (Ozeer *et al.*, 2018) ont proposé un protocole en quatre étapes afin de gérer les pannes dans les environnements de *fog computing*. Le protocole se base sur la technique de *checkpoint* afin d'enregistrer l'état des services puis observe le système pour détecter et signaler les éventuelles pannes. Le protocole est capable de prendre une décision

1. Un point de contrôle (*checkpoint)* : C'est un arrêt au cours duquel sont copiées les informations nécessaires à la reprise ultérieure du système

afin d'éviter une panne. Par contre, en cas de panne, le protocole la notifie aux différentes entités dépendantes. (Oma *et al.*, 2018) ont proposé une approche tolérante aux pannes dans les environnements de *fog computing* arborescent. Leur technique se base à la fois sur la réplication et la non-réplication.

Cependant, toutes ses solutions, basées sur des méthodes réactives ne sont pas très adaptées dans les environnements de *fog computing* dans la mesure où elles peuvent entraîner des retards et une consommation abusive des ressources (Alarifi *et al.*, 2019).

Dans le contexte des soins sanitaires basés sur l'internet des objets (H-IoT), la disponibilité et la latence sont des exigences de (*QoS*) essentielles à satisfaire lors du placement de services dans les noeuds du réseau. Une défaillance de ces systèmes et de leurs services peut avoir des impacts négatifs pouvant aller jusqu'à engendrer des décès (Kraemer *et al.*, 2017). Pourtant, tous les travaux présentés ci-dessus ne tiennent pas compte de cet aspect malgré le fait que les nœuds fog ont un taux de défaillance accru et une faible redondance contrairement aux nœuds cloud (Madsen *et al.*, 2013; Mavrogiorgou *et al.*, 2018). En effet, (Madsen *et al.*, 2013; Mavrogiorgou *et al.*, 2018) sont parmi les rares travaux existants dans la littérature qui ont considéré ces aspects dans le *fog computing*.

Nous allons, dans le chapitre suivant, proposer une plateforme de surveillance médicale à distance déployée sur le *fog computing* et tolérante aux pannes. Notre approche sera basée sur les méthodes proactives qui sont plus adaptées dans les environnements de *fog computing*. Pour de meilleures performances en termes d'équilibrage de charge et de délai de traitement des requêtes des patients, le déploiement d'un service sur les noeuds périphériques se fera avec un nombre limité de replicas.

CHAPITRE IV

F-RHM : *FOG-BASED REMOTE HEALTH MONITORING*

À travers ce chapitre, nous proposons une plateforme de surveillance médicale à distance déployée sur le *fog computing* et tolérante aux pannes. Nous commençons par une description du modèle du système avant de s'atteler sur le fonctionnement de ladite plateforme en décrivant ses différents composants. Par la suite, nous formulons mathématiquement le problème sous la forme d'un problème d'optimisation dont l'objectif est de maximiser le nombre de services déployé sur les *fog nodes* tout en satisfaisant les exigences des requêtes en termes de disponibilité et de délai. Nous terminons le chapitre par une proposition d'une solution heuristique qui se base sur l'algorithme d'évolution différentielle.

4.1 Présentation du modèle du système F-RHM

Nous présentons ci-dessous notre système de surveillance médicale à distance (F-RHM) basé sur l'informatique géodistribuée. Il est illustré par la figure 4.1 ci-après. Nous commençons par décrire le cas d'utilisation considéré avant de finir la partie par détailler le fonctionnement de F-RHM par une description de ses différents composants.

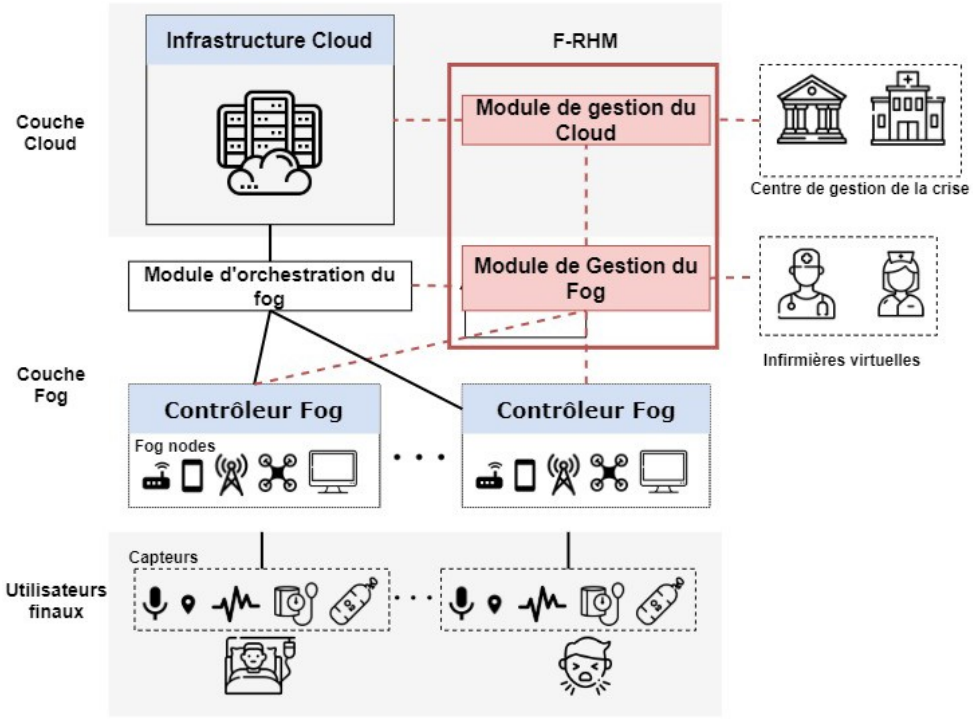

Figure 4.1 modèle du système F-RHM

4.1.1 Description du cas d'utilisation

L'accroissement du taux d'infection de la COVID-19 a globalement engendré le submergement des systèmes de santé. À cet effet, la plus grande préoccupation des autorités sanitaires demeure la limitation de la transmission du virus et une bonne gestion des ressources de soins sanitaires rares, telles que les lits d'hôpital, les masques, les respirateurs et les équipements de protection individuelle pour le personnel médical. C'est dans ce contexte que nous proposons cette plateforme de surveillance sanitaire à distance fiable basée sur le *fog computing* : F-RHM (*Fog-based Remote Health Monitoring*), afin de permettre une meilleure gestion de la crise à l'endroit des responsables de la santé.

Considérons un scénario où F-RHM travaille en collaboration avec les autorités de

santé via un Centre de Gestion de la Crise (CGC) selon la Fig.4.1, en déployant une plateforme d'assistance en ligne, fonctionnant 24h/24, dans laquelle des infirmiers autorisés seront chargés de dépister et de suivre les patients symptomatiques de COVID-19. Ainsi, une consultation via téléphone ou par vidéo-conférence, en fonction des besoins, est organisée pour évaluer l'état de santé du patient.

Les patients présentant de symptômes graves seront ensuite redirigés vers le niveau de soin approprié pour subir des tests à l'issue desquels, les confirmés de COVID-19 seront hospitalisés. Ceci permettra au centre de gestion de la crise (CGC) de pouvoir planifier et informer, à l'avance, le personnel hospitalier de l'arrivée de patients atteints de la COVID-19 pour une meilleure prise en charge. Les patients présentant des symptômes légers seront suivis à distance jusqu'à la disparition de ceux-ci ou jusqu'à leur aggravement occasionnant ainsi l'hospitalisation du patient en question.

F-RHM cible en priorité les groupes de population à haut risque et vulnérables (les personnes en quarantaine préventive, les personnes qui ont plus de 60 ans, les personnes immunodéprimées, ou souffrantes de maladies chroniques). Une fois admis sur la plateforme de surveillance à distance, un kit contenant divers capteurs permettant de mesurer la température corporelle, la pression artérielle, la fréquence respiratoire, la saturation en oxygène et d'enregistrer la toux est livré au domicile des patients. Les mesures précises fournies par ces capteurs permettent une détection précoce et un suivi efficace de la progression de la maladie, grâce aux différents composants de F-RHM qui seront décrits dans la partie qui suit. Par exemple, l'Organisation mondiale de la santé (OMS) indique qu'un taux d'oxygène dans le sang inférieur à 93 % stipule le début d'une pneumonie sévère qui est un symptôme significatif de la COVID-19.

L'objectif est d'empêcher les personnes présentant des symptômes légers de se

présenter dans les cliniques d'urgences où le risque d'exposition au virus est grand. Les agents de santé pourront ainsi se concentrer sur les patients qui ont des besoins de soins immédiats. Les autres patients seront suivis à distance depuis chez eux. En effet, conformément aux directives du Centre de Contrôle et de Prévention des Maladies (CCPM), les maisons sont des environnements moins stressants et contribuent positivement à la guérison des patients. Cela permettrait également de réduire les pénuries de lits d'hôpitaux et le risque de transmission du virus à d'autres patients, aux membres de la communauté et au personnel soignant.

F-RHM fonctionne en déclenchant des alarmes à la suite d'une détection de situations critiques. En effet, à la suite d'une détection de la détérioration des symptômes des patients abonnés à la plateforme, F-RHM intervient en émettant des alarmes auprès des infirmiers virtuels autorisés qui, à leur tour, surveillent de près les patients. Si les symptômes s'aggravent de plus en plus, le système avertit les infirmiers virtuels de prendre des mesures supplémentaires en contactant les patients concernés via téléphone ou par vidéo-conférence afin d'évaluer ou de fournir des soins à distance dans la mesure du possible. Au cas échéant, une hospitalisation du patient sera nécessaire pour un meilleur suivi.

F-RHM s'appuie également sur un module de gestion basé sur le cloud afin de permettre au CGC d'obtenir, grâce à une analyse avancée dans le cloud, de nouvelles informations sur la maladie (découverte de nouveaux symptômes) et d'être vigilant sur tous les événements qui nécessitent son attention (nouveau cas, cas guéris, détection d'une transmission communautaire, détection d'un patient en suivi qui se rend dans des zones surpeuplées, etc.).

4.1.2 Composants et fonctionnement de F-RHM

F-RHM est basé sur l'informatique géodistribuée (*fog computing*) et fonctionne selon la collaboration de trois couches hiérarchiques : la couche cloud, la couche fog et la couche utilisateur selon l'architecture illustrée dans la figure 2.1

4.1.2.1 Fonctionnement de F-RHM

La couche utilisateur est composée d'une panoplie de dispositifs IoT caractérisés par une distribution géographique étendue. Ils sont utilisés pour capturer les données médicales issues des patients. Dans notre scénario, nous considérons des capteurs médicaux portables (ou implantables) qui sont intégrés dans un module sans fil qui envoie les données collectées (la température, la localisation, la saturation en oxygène, l'ECG), de manière cryptée, aux couches supérieures (en particulier la couche fog) à des fins de traitement et de stockage.

La couche fog est composée de *fog nodes* qui sont des appareils intelligents (routeurs, commutateurs, passerelles, points d'accès, etc) qui peuvent être mobiles ou statiques (fixes) et capables de traiter, acheminer ou stocker des données issues des capteurs. Les *fog nodes* sont généralement regroupés dans des domaines qui sont placés sous l'autorité d'un contrôleur fog chargé d'effectuer les requêtes des utilisateurs finaux aux *fog nodes* les plus appropriés à l'aide des informations issues des capteurs de la couche inférieure. Cependant, les ressources de calcul et de stockage des noeuds de la couche fog sont très limitées. À cet effet, la couche fog n'est pas adaptée aux tâches très gourmandes en ressources.

Ainsi, ces requêtes sont déchargées vers la couche cloud pour bénéficier d'importantes capacités de calcul et de stockage de ses centres de données moyennant un délai plus élevé. La gestion des services entre les contrôleurs fog et les centres de

données de la couche cloud est coordonnée par le module d'orchestration.

4.1.2.2 Les modules composant le F-RHM

Un aperçu détaillé des différents modules constituant le F-RHM est fourni dans la figure 4.2.

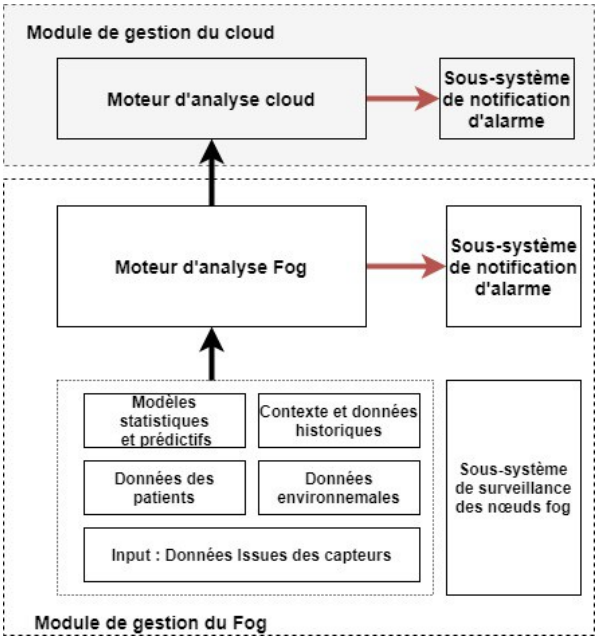

Figure 4.2 Composants des modules de F-RHM

F-RHM est conçu afin de permettre l'intégration des patients, des professionnels de santé et des CGSs (Centres de Gestion de la Crise) dans une plateforme où les données collectées à partir des capteurs attachés aux patients peuvent être analysées en profondeur afin d'obtenir des diagnostics et des alertes précis. Ainsi, ces capteurs sont équipés d'actionneurs afin de pouvoir exécuter les commandes des professionnels de santé (Mohsin *et al.*, 2018), ce qui nécessite de strictes mesures d'authentification et de confidentialité.

Le module de gestion du fog fonctionne au niveau de la couche fog. Nous supposons que des mesures de sécurité adéquates sont déployées sur les *fog nodes*. La mission principale de ce module est de combiner les données issues des capteurs et des données contextuelles telles que les informations des patients, les données environnementales, les bases de données médicales, etc.

Toutes ces données combinées seront envoyées au moteur d'analyse du fog pour des analyses plus précises. Ainsi, les données environnementales peuvent être mises en cache près des utilisateurs finaux afin de réduire la latence. L'accès aux informations sur les patients est restreint et soumis à des politiques de sécurité et de confidentialité. Le moteur d'analyse du fog détermine aussi les tâches analytiques à appliquer sur les données reçues (l'extraction d'ECG par exemple).

Elles sont ensuite communiquées au module d'orchestration du fog qui échange avec les contrôleurs fog afin de trouver l'emplacement (le(s) *fog node(s)* le(s) plus approprié(s)) pour l'exécution de ces tâches analytiques. Les résultats sont ensuite renvoyés au moteur d'analyse du fog pour déterminer si une alarme doit être déclenchée ou pas, sur la base des variations soutenues au fil du temps et sur plusieurs capteurs connectés à un patient donné.

Dans notre contexte, les alarmes déclenchées à ce niveau sont considérées comme urgentes et sont donc redirigées vers les autorités sanitaires locales, y compris les infirmières autorisées qui déterminent les prochaines actions à entreprendre.

Les résultats des tâches analytiques, qui ont une forte demande en ressources et qui requièrent un stockage à long terme, sont transférés au module de gestion du cloud pour une analyse plus adaptée aux exigences de ces dernières par le moteur d'analyse du cloud qui va extraire des informations globales et déclenche des alarmes auprès du CGC si nécessaire.

Le moteur d'analyse du cloud peut être utilisé par le CGC pour améliorer l'efficacité opérationnelle du système en effectuant un suivi de l'efficacité des interventions à la suite des alarmes. Ce qui peut éviter le retour au moteur d'analyse du fog afin de réduire les alarmes. Il peut également être utilisé pour planifier des tests de groupe.

Comme indiqué précédemment, il est essentiel de garantir la fiabilité et la disponibilité des *fog nodes* afin de satisfaire aux exigences de QoS, surtout pour la surveillance médicale à distance où le retard et la perte d'informations (non-déclenchement d'une alarme vitale critique par exemple) peut entraîner des conséquences néfastes. Étant donné la forte corrélation entre la disponibilité et la fiabilité (voir section 2.3), nous allons dans ce qui suit nous concentrés sur la disponibilité des *fog nodes*. Ainsi, pour F-RHM, un sous-système de surveillance de *fog nodes*, inspiré du travail de (Mavrogiorgou *et al*., 2018), est proposé afin de pouvoir calculer ou d'estimer la disponibilité des *fog nodes*. Le résultat est ensuite envoyé au module d'orchestration du fog afin d'être pris en compte par ce dernier dans l'optique de placer les services sur les *fog nodes* les plus adéquats.

Le fonctionnement de ce sous-système de surveillance de *fog node* repose sur un processus à trois étapes.

- 1^{ere} étape :

  Nous commençons par identifier les spécifications techniques liées à la disponibilité de base de chaque nouveau *fog node* en se basant sur les données fournies par le fabricant du modèle de *fog node*. Le tableau 4.1 fourni un extrait de ces données.

  En général, la crédibilité de ces données est partiellement mise en doute en raison du manque de normalisation pour le calcul des métriques liées aux pannes (O'Neill et Tolley, 2020; Krasich, 2009).

#	Nom *fog node*	Type *fog node*	TPA *fog node* (%)
1	Cisco C812G-CIFI	Routeur	1,11
2	Meraki MX67W	Appareil de sécurité	1,08
3	Xiaomi Redmi	Mobile	9
4	iPhone 6	Mobile	20
5	Cisco WS-C3850-12S	Commutateur	1,07
6	Cisco C1000-48T-4X-L	Commutateur	1.01
7	Aruba AP-504	Point d'accès	1,01
8	Allied Telesis TQ4400e	Point d'accès	1,08

Tableau 4.1 Extrait des données du fabricant de *fog nodes*

Source : Fiche technique du fabricant.

$TPA = 1 \quad \exp 8766↵ = 1 \quad \exp 8766/MT\ TF$ avec $↵ = 1/MT\ TF$, le taux de défaillance par heure

- 2^{eme} étape :

  À cause de ce manque de normalisation, ces données sont ensuite exploitées pour obtenir des données de base qui sont ajustées à leurs vraies valeurs au fil du temps. En effet, si un nouveau *fog node* n'a pas une entrée correspondante dans la base de données (données du fabricant), nous appliquons un algorithme de *clustering* des k plus proches voisins (*k-nearest neighbors* (K-NN)) , une méthode d'apprentissage supervisé, afin d'identifier un modèle de périphérique avec le comportement de défaillance la plus proche après une période d'observation prédéterminée au cours de laquelle nous collectons des données de défaillance pour le nouveau *fog node*.

  Pour illustrer le fonctionnement de notre *clustering* qui utilise l'algorithme K-NN, on considère trois classe de *fog nodes* (routeur, switch et point d'accès) et un jeu de données d'entraînement constitué de N couples (taux de

panne annuel et classe de *fog node*). Pour estimer le comportement de défaillance (déterminer la classe de *fog node*) associée à une nouvelle entrée de *fog node* dont on a pas les données du fabricant, la méthode des k plus proches voisins consiste à prendre en compte (de façon identique) les k échantillons des données d'apprentissage dont le taux de panne annuel est plus proche de celui du nouveau *fog node*, obtenu après une période d'observation. Ainsi, la classe la plus fréquente sur les classes les plus proches va représenter le modèle de périphérique de ce nouveau *fog node*.

- 3^{eme} étape :

  La dernière étape est l'échange de *feedbacks* entre le sous-système de surveillance de *fog node* et le module d'orchestration du fog afin d'améliorer la précision de notre modèle. En effet, le module d'orchestration du fog informe le sous-système de surveillance d'une panne d'un *fog node* afin qu'il la prenne en compte pour réévaluer la disponibilité de ce dernier.

4.2 Formulation mathématique de la problématique

Dans cette section, nous formulons mathématiquement notre plateforme de surveillance médicale à distance basée sur le *fog computing* avec une prise en compte des exigences en disponibilité et en délai.

4.2.1 Description du problème

Supposons le processus suivant où des capteurs attachés sur des patients envoient des tâches à traiter, suivant un certain nombre d'exigences de disponibilité et de délai, au niveau de la couche fog.

Un contrôleur fog sera chargé de déployer une stratégie d'allocation de ressources

afin de maximiser le nombre de requêtes des patients qui vont satisfaire aux exigences de disponibilité et de délai en prenant en compte la localisation des patients, le taux de pannes des *fog nodes*, la capacité des ressources disponibles (stockage disque, CPU, RAM) et la localisation des noeuds.

Après avoir mis sur pied cette stratégie d'allocation de ressources, les applications conteneurisées sont déployées sur les *fog nodes* avec une prise en charge des exigences de disponibilité et de délai des tâches qui seront redirigées vers ces applications . Afin de garantir une satisfaction continue des exigences des tâches des patients, le contrôleur créé, pour chaque application, un certain nombre de replica qui doit être minimisé, autant que faire se peut, car un nombre important de replica peut augmenter le temps de réponse à cause du délai de synchronisation de ces derniers (Farris *et al.*, 2017). Pour des raisons d'optimisation des ressources, un conteneur donné est capable d'exécuter des tâches compatibles de plusieurs patients.

4.2.2 Définition des variables

On considère T , un système de temps discrets. Soit K, un ensemble de domaines Fog. Chaque domaine est géré par un contrôleur Fog. F_k représente un ensemble hétérogène de noeuds Fog dans un domaine donné géré par un contrôleur k. La conteneurisation (une techniques de virtualisation légère) est utilisée dans le cadre de ce travail comme moyen de déploiement pour une amélioration de l'isolation, de la portabilité et de la sécurité des applications. Ainsi, on suppose que chaque noeud Fog $f \in F_k$ intègre une plateforme de conteneurisation pour une meilleure gestion des applications conteneurisées qui lui sont assignées. Pour la suite, nous allons considérer un seul domaine Fog pour une simplification de l'objectif. Pour le reste du document, nous allons omettre l'écriture des indices $k \in K$ et $t \in T$

pour une meilleure lecture.

Soit F un ensemble de *fog nodes*. Considérons les vecteurs suivants :

Le vecteur $\mathbf{l} = [l^f]$ exprime les emplacements des *fog nodes* et $\mathbf{c} = [c^f]$ désigne leur capacité de charge restante. Il est à noter que les ressources des *fog nodes* ne sont pas entièrement disponibles pour la plateforme *fog computing* (FC) . Elles sont allouées en priorité aux fonctions dédiées des *fog nodes*. Par exemple, un routeur doit acheminer les paquets réseau en priorité du traitement des tâches. Étant donné que la charge supportée par les fonctions dédiées varie dans le temps, la quantité de ressources disponibles sur la plateforme FC fluctue en conséquence Mseddi *et al.* (2019). Ainsi, nous exprimons par le vecteur $\mathbf{z} = [z^f]$, l'utilisation des *fog nodes*, proportionnelle à la quantité de ressources dédiées aux fonctions primaires. Le vecteur $\mathbf{h} = [h^f]$ représente le taux de panne des *fog nodes* qui peut être estimé ou extrait des données du fabricant. $\mathbf{b} = [b^f]$ est un vecteur binaire représentant si un *fog node* est disponible ou pas pour héberger des applications conteneurisées.

Nous définissons par U , un ensemble de patients dans le système avec $\mathbf{p} = [p^u]_{|U|}$, le vecteur représentant la localisation des patients. Nous considérons Y comme étant un ensemble de types d'application (exemple : électrocardiogramme, température corporelle, saturation oxygène, etc.) qui sont disponibles dans le système.

Soient $\mathbf{Q} = [Q^{u,y}]_{|U|\to|Y|}$ la matrice définissant la charge des tâches des applications, $\mathbf{T} = [T^{u,y}]_{|U|\to|Y|}$ la matrice définissant le délai seuil des applications et $\mathbf{A} = [A^{u,y}]_{|U|\to|Y|}$ la matrice définissant la disponibilité minimum requise par une application.

Comme noté précédemment, nous utilisons la réplication pour assurer la disponibilité requise par une application. Elle est également utilisée pour le partage

de charge. Chaque application doit être déployée sur R replicas, tout en chassant qu'un nombre important de R peut engendrer un délai de synchronisation important entre les différents replicas d'une application.

Soit la matrice binaire $\mathbf{W} = [w^{r,u,y,f}]_{R \to |U| \to |Y| \to |F|}$ indiquant si un replica r d'une application de charge u de type y est placé ou non sur le *fog node* f.

En exploitant la capacité des replicas pour le partage de charge, nous définissons par $\tilde{\mathbf{Q}} = [\tilde{Q}^{r,u,y,f}]_{R \to |U| \to |Y| \to |F|}$ la matrice représentant le partage de la charge des tâches applicatives dans les *fog nodes*. $\tilde{Q}^{r,u,y,f}$ représente la charge partielle de $Q^{u,y}$ induit par le replica r qui est déployé sur le noeud f.

Le tableau 4.2.2 donne un résumé plus clair de toutes ces variables que nous venons de définir.

Dans la partie qui suit, nous allons déterminer les expressions de disponibilité et de délai des applications que nous allons utiliser dans la formulation du problème d'optimisation.

4.2.3 Exigences en QoS

Les applications IoT doivent être conçues avec une certaine prise en charge des exigences en qualité de services (QoS) requis par leurs différents services. Sur ce, dans la mise en place de notre plateforme de surveillance médicale à distance basée sur le fog, nous allons considérer la disponibilité et le délai comme paramètres de QoS. Dans ce qui suit, nous allons donner leurs expressions que nous avons considérées dans la réalisation de ce travail.

T	Système de temps.
K	Ensemble de domaine fog.
F	Ensemble de *fog nodes*.
F_k	Ensemble de *fog nodes* dans le fog domaine $k \in K$.
U	Ensemble de patients suivis.
Y	Ensemble des types d'application (i.e., ECG, temperature, etc.).
l	$\|F\|$ vecteur de localisation des *fog nodes*.
c	$\|F\|$ vecteur représentant la capacité de charge restante des *fog nodes*.
z	$\|F\|$ Vecteur représentant l'utilisation des *fog nodes*.
h	$\|F\|$ Vecteur représentant le taux de panne des *fog nodes*.
b	$\|F\|$ Vecteur où l'élément b^f est un vecteur binaire représentant si $f \in F$ est libre pour héberger les conteneurs de l'application ou pas.
p	$\|U\|$ Vecteur de localisation des patients suivis.
Q	$\|U\| \times \|Y\|$ matrice représentant la charge des applications.
T	$\|U\| \times \|Y\|$ matrice indiquant les exigences de latence seuil pour les applications.
A	$\|U\| \times \|Y\|$ matrice représentant la disponibilité minimale requise par les applications.
W	$R \times \|U\| \times \|Y\| \times \|F\|$ matrice binaire représentant si un replica r d'une application de charge u de type y est déployé sur le *fog node* f ou pas.
$\tilde{\mathbf{Q}}$	$R \times \|U\| \times \|Y\| \times \|F\|$ matrice représentant la distribution des charges de tâches des applications, où $\tilde{Q}^{r,u,y,f}$ désigne une charge partielle de $Q^{u,y}$ d'un replica r placé dans le *fog node* f.
R	Nombre maximum de replica permis pour une application.

Tableau 4.2 Tableau de notations

4.2.3.1 Expression de la disponibilité

Dans le cadre de ce mémoire, nous avons considéré une disponibilité inhérente d'un noeud fog du système exprimée par l'équation 2.7. Le calcul d'une telle disponibilité se base sur un taux de panne constant. En réalité, le taux de panne d'un noeud varie dans le temps durant la durée de vie du système. À cet effet, on s'est inspiré des travaux de (Sun et Han, 2001; Dieye, 2016) qui fournissent une approximation de cette disponibilité. Les auteurs partent du principe selon lequel une variation du taux de panne n'est observable que sur une longue période. Par conséquent, nous supposons que le taux de panne varie peu et peut être considéré comme étant constant.

Nous simplifions notre modèle en supposant que la disponibilité d'une application est celle du nœud fog sur lequel elle est hébergée. Nous pensons que cette hypothèse est raisonnable, car les défaillances matérielles sont généralement la cause première de la perturbation du service à la périphérie du réseau en plus d'être la plus longue à atténuer (Lloyd's, 2018; Gill *et al.*, 2011).

Nous supposons μ, le taux constant de réparation d'un noeud en panne. La disponibilité $>^f$ à l'état stationnaire (on ne considère que le temps d'arrêt lié à la maintenance curative des *fog nodes* du système : Donc équivaut à la disponibilité inhérente exprimée dans l'équation 2.7) d'un *fog node* donné f est représentée par l'équation suivante :

$$>^f = \frac{\mu}{\mu + h^f} \quad 8\, f\, 2\, F, t\, 2\, T \tag{4.1}$$

où h^f représente le taux de panne d'un noeud fog f.

Étant donné qu'une application est disponible si, au moins, un de ses replicas est accessible. Ainsi, la disponibilité $A^{u,y}$ d'une application y d'un patient u est

représentée par l'équation suivante :

$$A^{u,y} = 1 \quad \prod_{f2F} (1 \quad \sum_{r=} >^{f} \cdot w^{r,u,y,f}), \ 8\, u\, 2\, U, y\, 2\, Y. \tag{4.2}$$

4.2.3.2 Expression du délai

La latence $d^{u,y}$ des tâches d'une application y pour un patient u est composé de trois délais :(1) le délai de communication (aller-retour) entre le noeud fog et le patient, (2) le délai de traitement des tâches et (3) le délai d'attente, qui est inclus dans le délai de traitement, dans le cadre de ce travail. Ainsi, la latence d'une application peut être écrite comme suit :

$$d^{\mu,y} = \ {}^{u,y}_{cmp} + d^{u,y}_{com} \quad 8\, u\, 2\, U, y\, 2\, Y \tag{4.3}$$

où ${}^{u,y}_{cmp}$ et $d^{u,y}_{com}$ représentent respectivement le délai de traitement et le délai de communication.

Étant donné que la charge des tâches d'une application est partagée sur plusieurs replicas, le délai de traitement $d^{u,y}_{cmp}$ des tâches d'un patient u pour une application y est égale au délai maximal des délais de traitement des replicas $d^{r,f,u,y}_{cmp}$. En plus, si un noeud n'est pas disponible, on suppose que son délai de traitement va prendre une très grande valeur telle que :

$$d^{u,y}_{cmp} = \max_{\substack{r=1,\ldots,R \\ f2F}} (d^{r,f,u,y}_{cmp} \cdot w^{r,u,y,f}), \quad 8u\, 2\, U, y\, 2\, Y. \tag{4.4}$$

Pour des raisons de simplicité, et pour que notre plateforme soit compatible avec d'autres modèle de fil d'attente, nous avons dérivé $d^{r,f,u,y}_{cmp}$ en utilisant un modèle de fille d'attente M/M/1. Ainsi, $d^{r,f,u,y}_{cmp}$ est représenté comme suit :

$$d^{r,f,u,y}_{cmp} = \frac{1}{\ldots f \quad \bar{Q}^{r,u,y,f}} \tag{4.5}$$

où μ^f est le taux de service d'un noeud f

De la même manière pour le délai de traitement, le délai de communication associé à un patient u pour une application y est bornée supérieurement par le plus grand délai de communication $d_{cmp}^{r,f,u,y}$ de ses replicas. Le délai de communication est donnée par l'expression suivante :

$$d_{com}^{u,y} = \max(d_{co}^{r,f,u,y} \cdot w^{r,u,y,f}) \quad \forall f \in F, u \in U, y \in Y. \tag{4.6}$$

Le délai de communication $d_{co}^{r,f,u,y}$ d'un replica r est décomposé comme suit :

$$d_{com}^{r,f,u,y} = d_{com,u,f}^{r,f,u,y} + d_{com,f,u}^{r,f,u,y} \tag{4.7}$$

où $d_{com,u,f}^{r,f,u,y}$ représente le délai de transmission de la communication sans fil depuis le patient u vers le noeud f et $d_{com,f,u}^{r,f,u,y}$ vice versa. Il est exprimé comme suit :

$$\frac{\tilde{Q}^{r,u,y,f}}{B \cdot \log_2(1 + SNR)} \tag{4.8}$$

où B représente la bande passante du support de transmission, SNR le rapport signal sur bruit et $\tilde{Q}^{r,u,y,f}$ représente la charge du replica r.

Suivant le modèle, notre objectif est de déterminer le placement optimal des replicas des applications conteneurisées. Nous cherchons à maximiser le nombre de tâches patients satisfaites en respectant le délai seuil et la disponibilité requis par application sous contraintes de disponibilité et des capacités des *fog nodes*.

4.2.4 Formulation du problème

Nous allons dans cette sous-section donner la formulation de problème de placement de service conteneurisé de e santé dans une infrastructure de *fog compting*. Nous allons commencer par décrire les contraintes auxquelles le système doit satisfaire avant de terminer par la fonction objective de notre problème qui sera la

maximisation du nombre d'applications à déployer qui satisfait à la disponibilité et à la latence requises.

4.2.4.1 Expression des contraintes

On commence par définir la contrainte pour satisfaire un délai seuil pour les tâches des patients.

$$T^{u,y} \quad d^{u,y} \; 2' \; G \cdot (S^{u,y} \quad 1), \quad 8u \; 2 \; U, y \; 2 \; Y \tag{4.9}$$

où G est un grand nombre positif choisi arbitrairement, $\mathbf{S}^{u,y}$ est une variable binaire indiquant si le délai $d^{u,y}$ est inférieur ou non au délai seuil $T^{u,y}$ de l'application.

La contrainte suivante assure que la satisfaction des requêtes d'un patient u pour une application y est limitée par la satisfaction de la charge requise des tâches (tous les replicas) de l'application. Elle est écrite comme suit :

$$S^{u,y} \leftarrow Q^{u,y} \leq \sum_{r=1}^{R} \sum_{f2F} \tilde{Q}^{r,u,y,f}, \quad 8u \; 2 \; U, y \; 2 \; Y. \tag{4.10}$$

La contrainte suivante concerne la satisfaction de la disponibilité de l'application utilisateur pour laquelle elle est inférieure au seuil requis.

$$A^{u,y} \quad a^{u,y} \; 2' \; G \cdot (S_A^{u,y} \quad 1), \quad 8u \; 2 \; U, y \; 2 \; Y \tag{4.11}$$

où $\mathbf{S}_A^{u,y}$ est une variable binaire indiquant si la disponibilité $A^{u,y}$ de l'application est supérieur ou non à la disponibilité requise $a^{u,y}$.

La satisfaction de la disponibilité d'une application y pour un patient u est limitée par la satisfaction de la charge requise des tâches (tous les replicas) de l'application. Elle est écrite comme suit :

$$S_A^{u,y} \leftarrow Q^{u,y} \leq \sum_{r=1}^{R} \sum_{f2F} \tilde{Q}^{r,u,y,f}, \quad 8u \; 2 \; U, y \; 2 \; Y. \tag{4.12}$$

La contrainte suivante va garantir que la charge totale allouée à chaque noeud fog ne dépasse passe la capacité de ce dernier (C^f). Elle est représentée comme suit :

$$C^f \geq \sum_{u \in U} \sum_{y \in Y} \sum_{r} \tilde{Q}^{r,u,y,f}, \quad \forall f \in F. \tag{4.13}$$

La contrainte suivante s'assure que deux replicas d'une application conteneurisée d'une application d'un patient ne peuvent pas être hébergés sur un même noeud fog.

$$\sum_{r=1}^{R} w^{r,u,y,f} \leq 1, \quad \forall u \in U, y \in Y, f \in F. \tag{4.14}$$

La contrainte suivante s'assure qu'un replica d'une application conteneurisée ne peut être héberge que sur un seul noeud fog.

$$\sum_{f \in F} w^{r,u,y,f} \leq 1, \quad \forall u \in U, y \in Y, 1 \leq r \leq R. \tag{4.15}$$

La contrainte suivante s'assure qu'un conteneur ne peut être déployé sur un noeud fog si et seulement si une portion de charge du conteneur est traitée par le noeud fog.

$$\tilde{Q}^{r,u,y,f} \leq G \cdot w^{r,u,y,f}, \quad \forall u \in U, y \in Y, f \in F, 1 \leq r \leq R. \tag{4.16}$$

4.2.4.2 Expression de la fonction objectif

Comme la latence des tâches des patients peut varier en fonction de la disponibilité sporadique et de l'emplacement des nœuds du fog, l'objectif de notre problème est de maximiser conjointement le nombre d'applications déployées répondant aux exigences de disponibilité et de délai seuil spécifiés.

La fonction objectif peut s'écrire comme suit :

$$\max_{W,\tilde{Q}} \sum_{u2U} \sum_{y2Y} (S^{u,y} + S_A^{u,y}) \quad \text{(P1)}$$
$$\text{s.t.} \quad (4.9) \ (4.16).$$

Le problème (P1) présenté ci-dessus est NP-Difficile. Pour le démontrer, considérons une forme simplifiée de notre problème de réplication visant à placer des applications conteneurisées sur des noeuds fog dans l'objectif de maximiser le nombre d'applications de patients capables de satisfaire le délai et la disponibilité requis, sans aucun mécanisme de réplication. Simplifions encore le problème en considérant le scénario où l'exigence en disponibilité de l'application est satisfaite. Cet exemple particulier du problème peut être trivialement modélisé comme par le GAP (Generalized Assignment Problem) qui est un problème bien connu de la littérature (Ross et Soland, 1975).

En effet, l'objectif du problème GAP est de trouver un mappage entre m agents et n tâches hétérogènes afin de maximiser le profit total. Chaque tâche est assignée, exactement, à un agent, étant donné qu'elle a une capacité suffisante. En plus, selon l'agent assigné, chaque tâche peut avoir un profit différent. Par analogie à notre problème, les agents peuvent être assimilés à des nœuds fog et à des applications conteneurisées en tant que tâches, avec un profit de tâche comme étant le nombre de requêtes répondues avec satisfaction de la latence. Étant donné qu'il est bien connu que GAP est NP-Difficile, notre problème simplifié est également NP-Difficile.

Selon Garey et Johnson, un problème NP-Difficile est synonyme d'impossibilité de trouver une solution optimale en temps raisonnable (Hochba, 1997). C'est dans ce sens que nous proposons dans ce qui suit une heuristique qui se base sur l'algorithme différentiel afin d'avoir une solution proche de l'optimale en temps po-

lynomial.

4.3 Heuristique proposée : DE-Adapté

Nous présentons dans cette section notre approche de résolution du problème formulé ci-dessus. Elle est notée DE-Adapté et s'est basée sur l'algorithme d'évolution différentielle (DE).

4.3.1 L'évolution différentielle (DE)

DE est un algorithme de recherche efficace à base de population crée par Stor et Price en 1995. Comparé à d'autres algorithmes évolutionnaires (EA), il est beaucoup plus simple à mettre en œuvre, nécessite moins de paramètres de contrôle tout en étant très performant dans une variété de domaines (Das et Suganthan, 2010). Dans la terminologie de l'algorithme d'évolution différentielle, un parent de la génération actuelle est appelé cible (*target*), un mutant obtenu par mutation est appelé donneur (*donor*) et enfin une progéniture, formée par croisement du donneur et de la cible, est appelée essai (*trial*).

Comme les autres algorithmes évolutionnaires traditionnels, DE comporte quatre étapes principales : initialisation, mutation, croisement et sélection.

DE commence par initialiser une génération de NP individus (solutions). Contrairement aux autres EAs, DE va générer ensuite une nouvelle génération en perturbant, grâce à un facteur d'échelle, la différence de deux solutions parentes (*targets*) différentes et choisies aléatoirement sur la population actuelle. C'est la phase de mutation. Ils existent plusieurs stratégies de mutation et chacune d'elles a son propre compromis entre l'exploitation et l'exploration (Sharma *et al.*, 2019; Das et Suganthan, 2010).

Soient v_j la j^{ieme} solution parente (*target*) et o_j la j^{ieme} solution enfant (*trial*) d'une génération courante. Considérons ainsi les stratégies de mutation suivantes :

- **DE/rand/1** : $o_j = v_{b_1} + M \cdot (v_{b_2} \; v_{b_3})$
- **DE/rand/2** : $o_j = v_{b_1} + M \cdot (v_{b_2} \quad v_{b_3}) + M \cdot (v_{b_4} \quad v_{b_5})$
- **DE/best/1** : $o_j = v_{best} + M \cdot (v_{b_1} \; v_{b_2})$
- **DE/best/2** : $o_j = v_{best} + M \cdot (v_{b_1} \quad v_{b_2}) + M \cdot (v_{b_3} \quad v_{b_4})$
- **DE/target-to-best/1** : $o_j = v_j + M(v_{best} \quad v_j) + M(v_{b_1} \quad vb_2)$
- **DE/rand-to-best/2** : $o_j = v_{b_1} + M \cdot (v_{best} \quad v_{b_1}) + M \cdot (v_{b_2} \; v_{b_3}) + M \cdot (v_{b_4} \quad v_{b_5})$

M est un facteur d'échelle, v_{best} est le meilleur parent de la population et b_i ($i \; 2 \; \{1, 2, 3, 4, 5\}$) est un indexe choisi aléatoirement dans $[1, NP]$.

Pour améliorer la diversité des solutions, un croisement est fait entre un *target* (une solution parent) $\checkmark_j = \{\checkmark_{j,1}, \ldots, \checkmark_{j,J'}\}$ et un *donor* (un mutant) $_j = \{ _{j,1}, \ldots, _{j,J'}\}$ pour créer une solution dite *trial* (une solution enfant) $\lessdot_j = \{\lessdot_{j,1}, \ldots, \lessdot_{j,J'}\}$ tel que, $8j' = 1, \ldots, J'$,

$$\lessdot_{j,j'} = \begin{cases} \checkmark_{j,j'} & \text{si } rand_{j,j'}[0, 1] \leq CR \text{ ou bien } j' = j_{rand} \\ \checkmark_{j,j'} & \text{si non} \end{cases} \tag{4.18}$$

$CR \; 2 \; [0, 1]$ représente le taux de croisement qui permet de savoir si la stratégie de mutation est appliquée ou non. La fonction $rand_{j,j'}[0, 1]$ génère un nombre aléatoire uniformément distribué entre $[0, 1]$ et j_{rand} est un indice choisi au hasard qui garantit qu'au moins un composant de l'individu donor $_j$ est hérité dans la solution enfant $\lessdot_j$.

Afin de maintenir une population constante au cours des prochaines générations, une phase de sélection est généralement appliquée, après la phase de croisement.

L'opération de sélection détermine entre un parent x_j et un enfant u_j, laquelle fera partie de la prochaine génération suivant l'équation suivante :

$$x_j^{+} = \begin{cases} u_j & \text{si } f(u_j) \leq f(x_j) \\ x_j & \text{si } f(u_j) > f(x_j), \end{cases} \tag{4.19}$$

x_j^{+} représente la solution cible de la prochaine génération et $f(\cdot)$ est une fonction d'évaluation (notre fonction de *fitness*), qui doit être maximisée. [1] Elle permet de garantir la non-détérioration de la qualité de la population dans le temps.

4.3.2 Adaptation de D.E à notre problème

Puisque DE est conçu pour résoudre des problèmes d'optimisation avec des paramètres de nombres réels, il ne peut pas être utilisé directement pour résoudre notre problème formulé. Dans cette sous-section, nous nous concentrons sur les procédures d'adaptation pour ce problème spécifique.

Commençons par définir au sein de la population, le nombre maximum de réplicas $|R|$ possible pour une demande d'application donnée. Nous déterminons $|R|$ en triant les nœuds fog en fonction de leurs taux de défaillance. Nous ajoutons progressivement les nœuds qui ont le plus faible taux de défaillance dans l'ensemble F' de telle sorte que :

$$1 - \prod_{f \in F'} \left(1 - \sum_{r=1}^{R} \lambda_f \right) \geq A^{u,y}, \quad \forall u \in U, y \in Y, \tag{4.20}$$

avec $F' \neq \emptyset$

Le choix du nombre de replica $|R|$ doit se faire judicieusement, car un grand nombre de replica va engendrer un délai important de synchronisation. Ce qui

1. les détails de $f(\cdot)$ seront décrits dans la section suivante

peut impacter les performances du système.

Pour éviter le scénario de point de défaillance unique, c'est-à-dire lorsque $A^{u,y} ! 0$ et donc $T^{u,y} ! 1$, nous nous assurons que le nombre maximum de replica R pour une application soit supérieur ou égale à 2 ($R 2' 2$).

Soit $\checkmark_j$ une descendante cible de dimension R représentant une solution potentielle à notre problème d'optimisation. Soit $\{\checkmark_{j,1}, \checkmark_{j,2}, \ldots, \checkmark_{j,j^0}, \ldots, \checkmark_{j,R}\}$ les *fog nodes* la composant. Bien qu'une progéniture cible ait une dimension fixe, l'utilisation de progénitures de taille dynamique reste possible dans notre modèle grâce à l'ajout de *fog node* de remplissage virtuels qui n'ont aucun impact sur le système. De plus, à des fins de décodage, la correspondance entre un index j^0 et son *fog node* associé f est maintenue par une routine interne. Pour s'assurer que les contraintes (4.14) - (4.15) sont respectées, nous validons que deux composantes de $\checkmark_j$ ne pointent pas vers le même *fog node* f.

Ainsi, notre fonction d'évaluation (*fitness function*) $\leftarrow(\cdot)$ pour une progéniture $\checkmark_j$ donnée est exprimée comme suit :

$$\leftarrow(\checkmark_j) = \frac{(A^{u,y}_{\checkmark} \quad A^{u,y})^2}{\{^A} \quad \frac{(T^{u,y}_{\checkmark} \quad T^{u,y})^2}{\{^T}, \tag{4.21}$$

$A^{u,y}_{\checkmark_j}$ et $T^{u,y}_{\checkmark}$ représentent respectivement la disponibilité et le délai courent fournis par les composants de $\checkmark_j$. Les variables $\{^A$ et $\{^T$ sont des facteurs d'échelle de pénalité visant à décourager à la fois le sous-approvisionnement et le sur-approvisionnement. Elles sont définies comme suit :

$$\{^A = \begin{cases} \{^A_1, & \text{si } (A^{u,y}_{\checkmark} \quad A^{u,y}) > 0 \\ \{^A_2, & \text{si non,} \end{cases} \tag{4.22}$$

$$\{^T = \begin{cases} \{^T_1, & \text{si } (T^{u,y}_{\checkmark} \quad T^{u,y}) > 0 \\ \{^T_2, & \text{si non,} \end{cases} \tag{4.23}$$

où ($\{^A_1, \{^A_2, \{^T_1, \{^T_2 \in \mathbb{R}^*$), avec $\{^A_2 < \{^A_1$ et $\{^T_2 < \{^A_1$.

Le sous-approvisionnement induit la non-satisfaction des exigences de QoS, tandis que le sur-approvisionnement peut mettre en danger le placement des demandes futures (surconsommation des ressources).

Afin d'adapter le processus de mutation, nous commençons par caractériser, pour toute descendance χ_j, la contribution inhérente $\Phi(\chi_{j,j'})$ d'un *fog node* $\chi_{j,j'}$ composant de χ_j pour la satisfaction des exigences de QoS comme suit :

$$\Phi(\chi_{j,j'}) = \omega^A \cdot (\gamma_{j'} - A^{u,y}) + \left(\frac{\omega^T}{(\hat{o}^{r,u,y,j'}_{\text{em}} + \hat{o}^{r,u,y,j'}_{\text{co}}) \cdot T^{u,y}} \right), \tag{4.24}$$

où $(\omega^A, \omega^T \in \mathbb{R} \mid \omega^A + \omega^T = 1)$ sont respectivement les poids indiquant la priorité des exigences de disponibilité et de latence. Nous priorisons la disponibilité sur la latence, d'où $\omega^A > \omega^T$. Ensuite, nous définissons la contribution relative $\tilde{\Phi}(\chi_{j,j'})$ d'un *fog node* composant $\chi_{j,j'}$ par rapport à ses pairs par l'équation suivante :

$$\tilde{\Phi}(\chi_{j,j'}) = (NP - 1) \cdot \frac{\Phi(\chi_{j,j'})}{\sum_{\substack{j''=1 \\ j'' \neq j'}} \Phi(\chi_{j,j''})}, \tag{4.25}$$

où NP est la taille de la population sur laquelle, les *fog nodes* sont triés en conséquence. Nous ajustons ensuite le processus de mutation de la manière suivante :

Soient χ_1 et χ_2 deux descendants impliqués dans le processus de mutation et o le mutant résultant. Ensuite, les descendants sont redéfinis selon les relations suivantes :

$$_1\chi \; \chi_2 \begin{bmatrix} \\ (\chi_{1,1}) & (\chi_{2,1}) \\ (\chi_{1,2}) & (\chi_{2,2}) \\ \vdots & \vdots \\ (\chi_{1,R}) & (\chi_{2,R}) \end{bmatrix}$$

et

$$\checkmark_1 + \checkmark_2 = \begin{bmatrix} (\checkmark_{1,1}) + (\checkmark_{2,1}) \\ (\checkmark_{1,2}) + (\checkmark_{2,2}) \\ \cdots \\ (\checkmark_{1,R}) + (\checkmark_{2,R}) \end{bmatrix}.$$

Nous conservons une utilisation similaire au facteur d'échelle de mutation classique M comme paramètre d'ajustement d'exploration/exploitation de recherche.

Dans notre contexte, nous supposons une règle appliquée pour les composants M telle que, lorsque $0 = (\checkmark_{1,j_0}) \quad (\checkmark_{2,j_0}) > 0$, on retient dans la progéniture mutante $(\checkmark_{1,j_0})$ au détriment de $(\checkmark_{2,j_0})$. Le but est de créer une progéniture mutante avec des composants caractérisés par une forte contribution inhérente aux exigences de QoS.

Pour $0 = (\checkmark_{1,j_0}) + (\checkmark_{2,j_0})$, nous utilisons l'algorithme glouton afin de déterminer les composants de $(\checkmark_{1,j_0})$ qui doivent être transférés dans $(\checkmark_{2,j_0})$ dans le but de maximiser $\leftarrow(0)$, ce qui implique que les composants les plus contributifs (respectivement les moins contributifs) de $(\checkmark_{1,j_0})$ peuvent être conservés pour éviter le sous-approvisionnement (respectivement le sur-approvisionnement).

Dans ce qui suit, nous allons évaluer les performances de notre proposition.

CHAPITRE V

ÉVALUATION DES PERFORMANCES DE DE-ADAPTÉ

Nous allons, à travers ce chapitre, évaluer les performances de notre heuristique proposée (DE-Adapté). Pour ce faire, nous allons la comparer avec des approches largement utilisées dans la littérature (une stratégie gloutonne et une stratégie aléatoire) suivant un certain nombre de métriques de performance. Les détails seront décrites dans ce qui suit.

5.1 Stratégies de résolution (*Baselines*)

Nous allons dans cette partie, présenter les différentes stratégies et leur algorithme respectifs que nous avons utilisé comme *baseline* par rapport à notre problème d'optimisation (P1).

5.1.1 Stratégie gloutonne *(Greedy First Fit)*

C'est une stratégie algorithmique qui, à chaque étape de résolution d'un problème, considère le premier choix qui satisfait aux contraintes comme étant le meilleur choix optimal dans le but de mener finalement à une solution globale optimale. Il suit le principe de faire étape par étape. Il choisit une meilleure solution à chaque étape sans pour autant tenir compte des conséquences sur la solution globale du

Algorithme 1 : Stratégie Gloutonne

Input : F, $\mathbf{L}_{|F|}$, $\mathbf{C}_{|F|}$, $\mathbf{Z}_{|F|}$, $\mathbf{H}_{|F|}$, $\mathbf{D}_{|F|}$, $\mathbf{P}_{|U|}$, Y, $\mathbf{Q}_{|U|\to|Y|}$, $\mathbf{T}_{|U|\to|Y|}$, $\mathbf{A}_{|U|\to|Y|}$

Output : Schémas de déploiement : $\mathbf{W}_{R\to|U|\to|Y|\to|F|}$

Initialisation de la matrice $\boldsymbol{W}_{R\to|U|\to|Y|\to|F|}$;
for *chaque application à placer :* **do**
    Définir un nombre constant de réplicas R ;
    Choisir aléatoirement R fog node avec respect des contraintes : F_R ;
    Mise à jour de la matrice $\mathbf{W}_{R\to|U|\to|Y|\to|F|}$;
    for *chaque* $f_i \; 2 \; F_R$ **do**
        for *chaque* $f_j \; 2 \; F \setminus F_R$ **do**
            if f_j *est meilleur que* f_i **then**
                Remplacer f_i par f_j dans $\mathbf{W}_{R\to|U|\to|Y|\to|F|}$;
                break ;
            end
        end
        Mise à jour de $\mathbf{C}_{|F|}$;
    end
end
return $\mathbf{W}_{R\to|U|\to|Y|\to|F|}$

Algorithme 2 : Stratégie Aléatoire

Input : F, $\mathbf{L}_{|F|}$, $\mathbf{C}_{|F|}$, $\mathbf{Z}_{|F|}$, $\mathbf{H}_{|F|}$, $\mathbf{D}_{|F|}$, $\mathbf{P}_{|U|}$, Y, $\mathbf{Q}_{|U|\to|Y|}$, $\mathbf{T}_{|U|\to|Y|}$, $\mathbf{A}_{|U|\to|Y|}$

Output : Schémas de déploiement : $\mathbf{W}_{R\to|U|\to|Y|\to|F|}$

Initialisation de la matrice $\mathbf{W}_{R\to|U|\to|Y|\to|F|}$ à 0

for *chaque application à placer :* **do**

  Définir le nombre de réplicas R

  Choisir aléatoirement R fog node avec respect des contraintes : F_R

  Mise à jour de la matrice $\mathbf{W}_{R\to|U|\to|Y|\to|F|}$

  Mise à jour de $\mathbf{C}_{|F|}$

end

return $\mathbf{W}_{R\to|U|\to|Y|\to|F|}$

problème, d'où son nom d'algorithme gourmand. (Cormen *et al.*, 2009).

Notre stratégie gloutonne pour la résolution de (P1) consiste à déployer, autant que faire se peut, les conteneurs sur des *fog nodes* qui satisfont aux exigences de disponibilité et de délai des requêtes des patients, en tenant compte des contraintes 4.9 - 4.16. Pour se faire, pour chaque conteneur, on choisit aléatoirement R *fog nodes* (*fog nodes* replica) avec satisfaction des contraintes 4.9 - 4.16. Pour chaque *fog nodes* replica, une recherche exhaustive est faite sur les noeuds restant afin de mettre à jour les *fog nodes* replica avec les noeuds qui offrent les meilleures performances en terme de délai et de disponibilité. Les détails sont décrits dans l'algorithme 1.

5.1.2 Stratégie aléatoire

C'est une approche statique consistant à choisir de manière aléatoire, pour chaque conteneur à déployer, un nombre R de replica (R *fog nodes* différents) sur lesquels sera placé ce dernier. Le pseudo-code est décrit dans l'algorithme 2.

5.2 L'environnement des simulations

Nous avons considéré un réseau fog constitué de n fog nodes appartenant à un seul domaine sous la supervision d'un seul contrôleur de domaine. Dans l'exécution de notre DE-adapté, nous avons considéré une population de taille 100 et 500 itérations. Nous avons considéré un taux de croisement égal à 0.7 et 2 comme facteur de mutation.

Nous avons implémenté notre solution DE-Adapté, ainsi que nos deux *baselines* en Python. Nous avons lancé les simulations sur une machine HP Windows 10 avec les caractéristiques suivantes : intel core(TM) i7-5500U CPU @2.4 GHz 2.4 GHz, 8Go de RAM et un système d'exploitation à 64 bits.

5.3 Scénarios de simulation

Soient des patients positifs de COVID-19 à surveiller à distance via notre plateforme F-RHM. Nous considérons quatre types de capteurs, attachés sur les patients afin de mesurer leur rythme cardiaque, leur taux d'oxygène dans le sang, la température, l'enregistrement de la toux. Ces données capturées sont ensuite envoyées dans un domaine fog constitué de dix *fog nodes* statiques (la mobilité n'est pas prise en compte) qui sont sous la supervision d'un contrôleur fog. On considère aussi quatre types d'application (rythme cardiaque, taux d'oxygène, température et enregistrement de la toux). Chaque application est constituée de n services conteneurisés qui doivent être déployés au niveau des noeuds du réseau fog sous contrainte d'une meilleure satisfaction des exigences de QoS des requêtes des patients. Ces requêtes sont traitées (placement de conteneurs) suivant leur ordre d'arrivée selon une file d'attente M/M/1.

Pour évaluer les performances de notre solution DE-Adapté, considérons les scé-

narios suivants :

5.3.1 Scénario 1

Considérons deux types d'application permettant respectivement de mesurer le rythme cardiaque et la température de |n| patients qui sont surveillés à distance via notre plateforme suivant un réseau fog constitué de cinq *fog nodes*. Le tableau 5.1 suivant donne la charge et les exigences des applications (disponibilité requise, délai requis et sa charge). On fait varier le nombre d'applications à déployer (allant de 10 à 50) qui seront générées suivant les exigences de chaque type d'application renseignées dans le tableau 5.1.

Un patient est caractérisé par sa position et une puissance de transmission de sa passerelle. Ainsi, nos n patients sont générés aléatoirement suivant les valeurs mentionnées dans le tableau 5.2.

Type d'application	Dispo Requise (%)	Delai Requis (ms)	Charge (Mbits)
ECG	0,99	10	0,3
Température	0,99	5	0,7

Tableau 5.1 Exigences des applications

Patients	Coord. X (m)	Coord. Y (m)	Puiss. Trans. (Watt)
Patient 1	[0-10]	[0-10]	[5-15]
...	[0-10]	[0-10]	[5-15]
Patient 5	[0-10]	[0-10]	[5-15]

Tableau 5.2 Informations relatives aux patients

Un *fog node* est caractérisé par sa position (ses coordonnées géographique (m)), sa

capacité de stockage (Go), son taux de panne (TP), un taux de réparation (TR) et un taux de service (TS). Il sont aussi générés suivant les valeurs mentionnées dans le tableau 5.3.

Fog Nodes	Capacité (Go)	Coord. X	Coord. Y	TP	TR	TS
fog node 1	[15-70]	[0-10]	[0-10]	[0,1-0,3]	0,9	[0,2-0,9]
...	[15-70]	[0-10]	[0-10]	[0,1-0,3]	0,9	[0,2-0,9]
fog node n	[15-70]	[0-10]	[0-10]	[0,1-0,3]	0,9	[0,2-0,9]

Tableau 5.3 Caractéristiques des *fog nodes*

À l'issue de ce scénario, on cherche à évaluer la performance de notre solution (DE-Adapté) par rapport à une stratégie gloutonne et une stratégie aléatoire. En effet, on cherche à déterminer, pour chaque déploiement, le nombre d'applications qui ont réussi à être déployées par chaque stratégie (DE-Adapté, Greedy et le Random). Nous allons par la suite déterminer, parmi les applications réellement déployées, le nombre qui a satisfait aux exigences de délai et de disponibilité. Pour se faire, 100 *fog nodes* seront générés aléatoirement suivant les valeurs du tableau 5.3. Les résultats obtenus seront présentés et discutés dans la section 5.4.

5.3.2 Scénario 2

Dans ce scénario, nous faisons varier la fiabilité des noeuds. Ainsi, nous considérons trois cas d'utilisation. Dans le premier cas (meilleur cas), tous les noeuds ont une fiabilité faible (taux de panne compris entre 0,1 et 0,3). Dans le deuxième cas (cas moyen), les noeuds ont une fiabilité moyenne (taux de panne compris entre 0,4 et 0,6). Dans le troisième cas (pire cas), les noeuds ont une fiabilité élevée (taux de panne compris entre 0,7 et 0,9). Nous considérons 50 applications de 4 types dont les valeurs de leurs exigences et charge respectives sont renseignées dans le

tableau 5.4 . Nous cherchons, à l'issue de ce scénario, à déterminer le nombre de requêtes pour lesquelles une violation de la disponibilité requise est observée, de même que pour le délai, selon les trois stratégies pour chaque cas d'utilisation.

Type d'application	Dispo Requise (%)	Delai Requis (ms)	Charge (Mbits)
ECG	0,999	10	0,3
Température	0,998	5	0,7
Enregistrement toux	0,997	12	0,9
Pression artérielle	0,996	8	0,5

Tableau 5.4 Exigences des applications

Les résultats obtenus sont présentés et discutés dans la partie qui suit.

5.4 Résultats et discussions

5.4.1 Scénario 1

Les résultats de nos simulations du scénario 1 sont représentés dans les figures suivantes :

Dans la figure 5.1, nous montrons les performances de notre solution (DE-Adapté), comparées à celles des *baselines* (stratégie gloutonne et aléatoire), par rapport au nombre d'applications réellement déployées. Comme le montre la figure, nous constatons nettement que, pour chaque nombre d'applications à déployer (variant de 10 à 50), DE-Adapté donne les meilleures performances.

Par la suite, dans ce même scénario, nous nous sommes intéressés à la satisfaction des exigences de qualité de services pour l'ensemble des requêtes des patients. Dans la figure 5.2, nous montrons, en faisant varier le nombre de requêtes de 10 à 50, le nombre de requêtes pour lesquelles les exigences de délai ont étaient

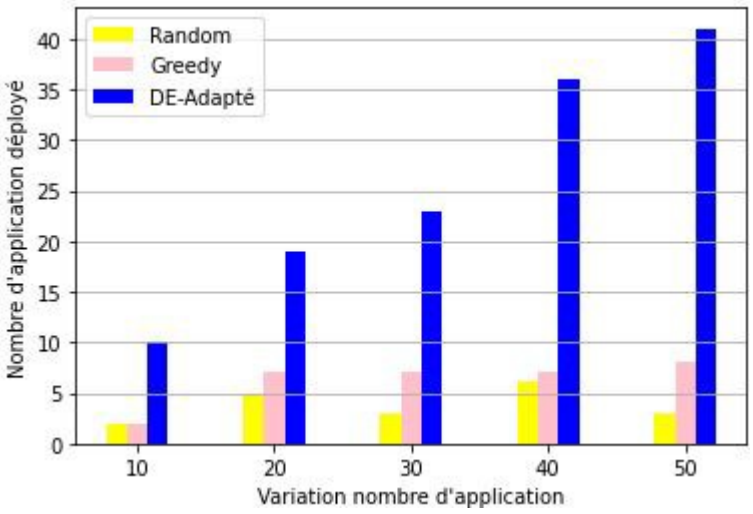

Figure 5.1 Nombre d'application déployée par stratégie

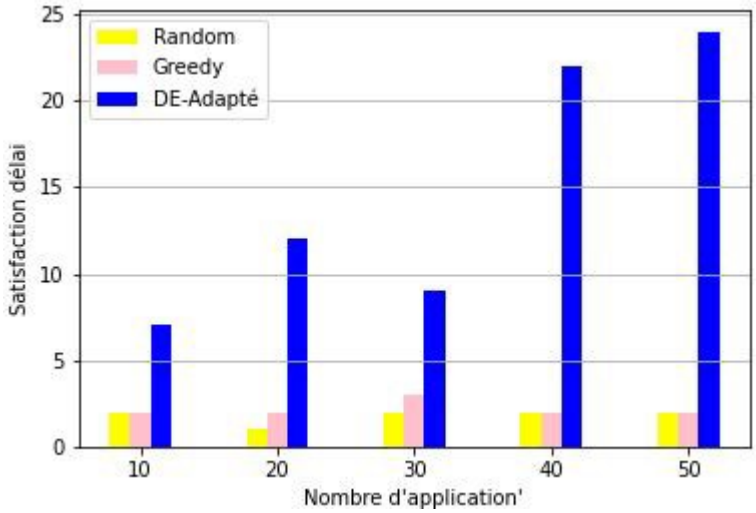

Figure 5.2 Nombre d'application placée avec satisfaction du délai

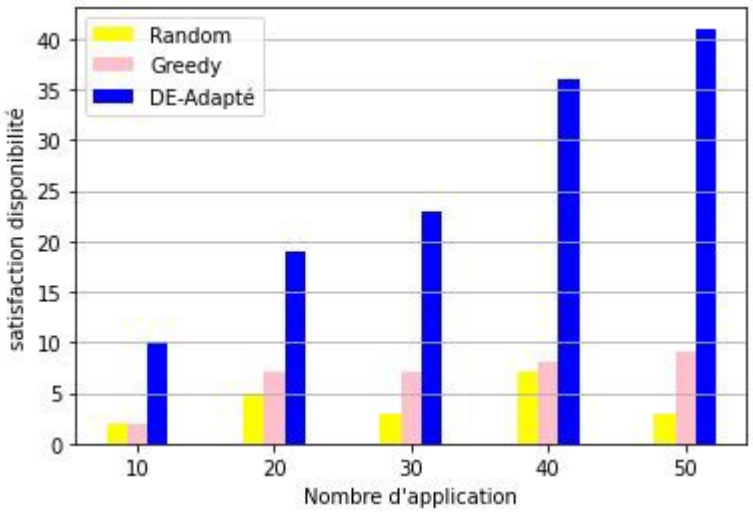

Figure 5.3 Nombre d'application placée avec satisfaction de la disponibilité

satisfaites. Dans la figure 5.3, le nombre de requêtes pour lesquelles les exigences de disponibilité ont étaient satisfaites est mis en exergue. Ainsi, nous constatons que DE-adapté donne les meilleures performances aussi bien sur la satisfaction des exigences de disponibilité que sur la satisfaction des exigences de délai.

5.4.2 Scénario 2

Dans ce scénario 2, nous cherchons à évaluer les performances de notre solution suivant la violation des exigences de disponibilité (figure 5.4) et de délai (figure 5.5) des requêtes des patients. Ainsi, nous avons considéré trois cas d'utilisation (pire cas, cas moyen et meilleur cas). Pour rappel, au pire, nous avons considéré que les *fog nodes* ont un taux de panne élevé, au cas moyen, un taux de panne moyen et au meilleur cas un taux de panne faible.

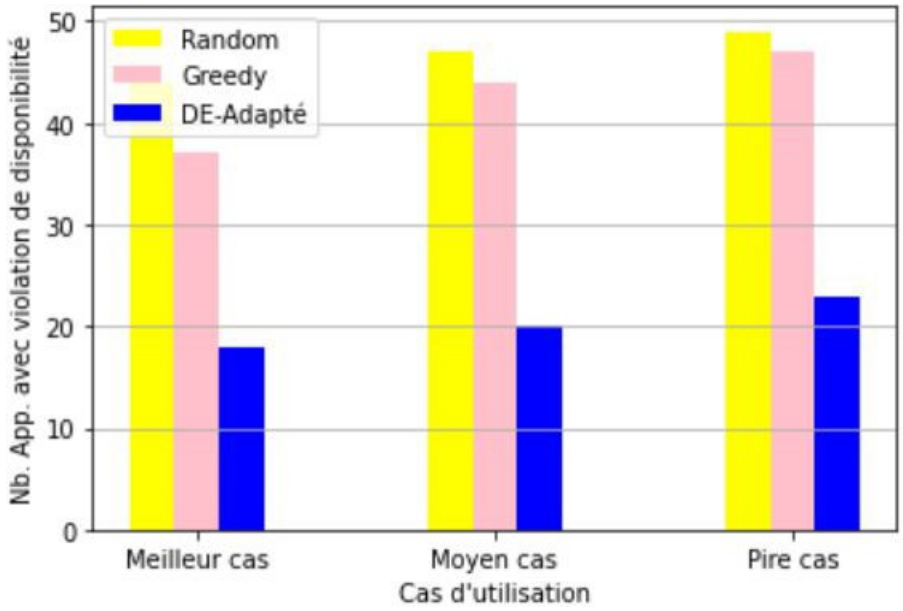

Figure 5.4 Violation disponibilité en fonction de la fiabilité des noeuds.

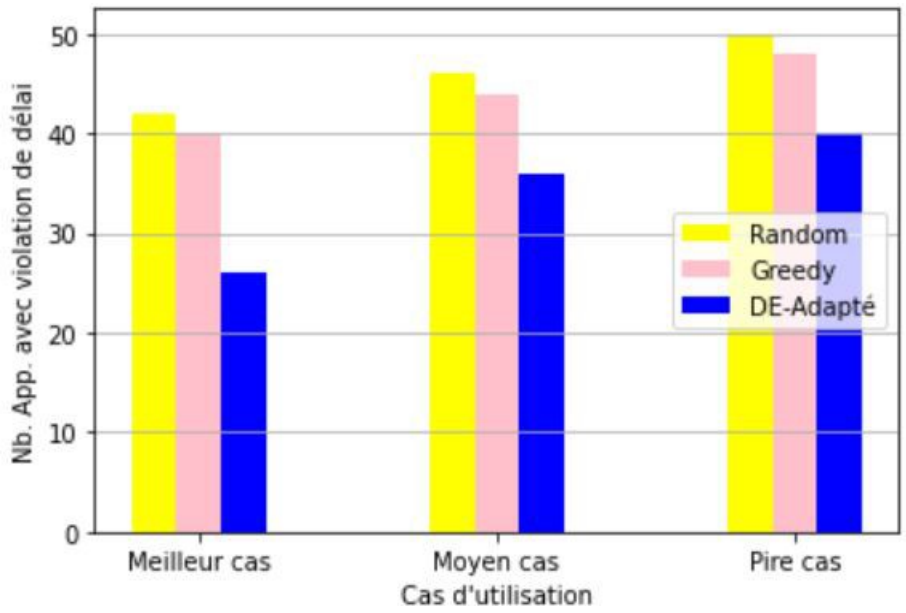

Figure 5.5 Violation du délai en fonction de la variation de la fiabilité des noeuds.

Les figures nous montrent que DE-adapté offre moins de violation des exigences au pire, moyen et meilleur cas. Par contre il est à noter aussi que ces performances diminuent en fonction de l'importance du taux de panne des noeuds. En d'autres termes, les violations sont plus importantes au pire cas qu'au meilleur cas.

Ci-dessous la conclusion a tiré des résultats obtenus :

En partant de notre objectif de départ qui était de maximiser le nombre d'applications à déployer avec satisfaction des exigences de délai et de disponibilité, on constate que notre solution est relativement solide dans la mesure où, malgré ses bonnes performances par rapport aux deux autres stratégies, nous constatons une violation des exigences de sa part. Cela peut s'expliquer par l'hypothèse suivante :

Le choix du nombre de replicas joue un rôle déterminant dans l'atteinte de ces objectifs. Plus ce nombre est grand, moins sont les violations de QoS. Par contre cela occasionne un important délai de synchronisation. À cet effet, notre solution a fait un compromis entre le délai de synchronisation des replicas et la violation de disponibilité et de délai. Ainsi, elle est parvenue à minimiser les violations et les délais de synchronisation par le choix petit nombre de replicas pour chaque application.

CHAPITRE VI

CONCLUSION

Ces dernières années l'Internet des objets est largement utilisé dans le domaine de la santé, plus particulièrement dans la surveillance médicale à distance. Ses applications produisent un gros volume de données à traiter pour une meilleure prise de décision médicale dans un délai court. Ainsi, l'infonuagique n'est pas adaptée à ce nouveau type de besoin, car ses centres de données sont caractérisés par un délai de réponse très élevé. A cet effet, l'informatique géodistribuée (fog computing) est récemment proposée dans la littérature pour une prise en charge des tâches sensibles au délai.

Dans le contexte actuel de la pandémie de la COVID-19 où la seule solution existante, pour le moment, pour limiter la transmission communautaire du virus est préventive (confinement et gestes barrières). Ainsi, l'aplatissement de la courbe est la politique épidémiologique de santé publique adoptée pour une meilleure prise en charge des malades. Cependant, l'application stricte de cette politique est difficile en pratique. En plus le virus peut être transmis par des personnes asymptomatiques ou présymptomatiques.

Fort de ce constat, nous avons proposé, dans le cadre de ce mémoire, F-RHM (*Fog-based Remote Health Monitoring*), une plateforme de surveillance médicale à distance, déployée dans le *fog computing* où les noeuds sont d'une fiabilité va-

riable. F-RHM est capable d'identifier et d'estimer la disponibilité des noeuds afin de proposer un meilleur placement des conteneurs avec une prise en charge des exigences de disponibilité et de temps de réponse.

Nous avons formulé ce problème de placement de conteneurs avec prise en charge des exigences de qualité de service sous forme d'un problème d'optimisation dont l'objectif est de maximiser le nombre d'applications à déployer qui satisfont aux exigences de délai et de disponibilité des requêtes des patients, sous plusieurs contraintes. Nous avons par la suite prouvé que ce problème est NP-difficile avant de proposer une heuristique basée sur l'algorithme *Differential Evolution* (DE), nommée DE-Adapté pour la résolution de ce dernier.

Nous avons comparé notre solution avec une stratégie gloutonne et une stratégie aléatoire. Sur plusieurs scénarios de simulation, les résultats ont montré que DE-adapté donne toujours les meilleures performances en termes de violation des exigences de QoS.

Nous comptons, dans le futur, appliquer à notre heuristique une méthode de sélection d'opérateurs adaptative (en anglais, *Adaptative Operator Selection (AOS))* pour un meilleur choix de la stratégie de mutation qui, selon plusieurs analyses expérimentales, est très déterminant sur les performances des algorithmes évolutifs.

RÉFÉRENCES

Aazam, M. et Huh, E.-N. (2015). E-hamc : Leveraging fog computing for emergency alert service. Dans *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, 518–523. IEEE.

Abd Latiff, M. S. *et al.* (2017). A checkpointed league championship algorithm-based cloud scheduling scheme with secure fault tolerance responsiveness. *Applied Soft Computing*, *61*, 670–680.

Alarifi, A., Abdelsamie, F. et Amoon, M. (2019). A fault-tolerant aware scheduling method for fog-cloud environments. *PloS one*, *14*(10), e0223902.

Bertini, M., Marcantoni, L., Toselli, T. et Ferrari, R. (2016). Remote monitoring of implantable devices : should we continue to ignore it ? *International journal of cardiology*, *202*, 368–377.

Bonomi, F., Milito, R., Natarajan, P. et Zhu, J. (2014). Fog computing : A platform for internet of things and analytics. In *Big data and internet of things : A roadmap for smart environments* 169–186. Springer.

Bonomi, F., Milito, R., Zhu, J. et Addepalli, S. (2012a). Fog computing and its role in the internet of things. Dans *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, 13–16.

Bonomi, F., Milito, R., Zhu, J. et Addepalli, S. (2012b). Fog computing and its role in the internet of things. Dans *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, 13–16.

Bourhim, E. (2020). *La communication et le placement de conteneurs docker dans le fog computing*. (Mémoire de maîtrise). Université du Québec à Montréal.

Brogi, A., Forti, S., Guerrero, C. et Lera, I. (2020). How to place your apps in the fog : State of the art and open challenges. *Software : Practice and Experience*, *50*(5), 719–740.

Cao, Y., Chen, S., Hou, P. et Brown, D. (2015). Fast : A fog computing assisted distributed analytics system to monitor fall for stroke mitigation. Dans *2015 IEEE International Conference on Networking, Architecture and Storage (NAS)*, 2–11. IEEE.

Cascella, M., Rajnik, M., Cuomo, A., Dulebohn, S. C. et Di Napoli, R. (2020). Features, evaluation and treatment coronavirus (covid-19). In *Statpearls [internet]*. StatPearls Publishing.

Celesti, A., Mulfari, D., Fazio, M., Villari, M. et Puliafito, A. (2016). Exploring container virtualization in iot clouds. Dans *2016 IEEE International Conference on Smart Computing (SMARTCOMP)*, 1–6. IEEE.

Chen, H. et Liu, H. (2016). A remote electrocardiogram monitoring system with good swiftness and high reliablility. *Computers & Electrical Engineering*, *53*, 191–202.

Cisco (2016a). Fog computing and the internet of things : Extend the cloud to where the things are. Récupéré le 2020-06-13 de `https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf`

Cisco, C. V. (2016b). Cisco global cloud index : Forecast and methodology, 2015-2020. white paper. *Cisco Public, San Jose*.

Cisco, M. (2015). Iot, from cloud to fog computing. Récupéré le 2020-06-18 de `https://blogs.cisco.com/perspectives/iot-from-cloud-to-fog-computing`

Cormen, T. H., Leiserson, C. E., Rivest, R. L. et Stein, C. (2009). *Introduction to algorithms*. MIT press.

Craciunescu, R., Mihovska, A., Mihaylov, M., Kyriazakos, S., Prasad, R. et Halunga, S. (2015). Implementation of fog computing for reliable e-health applications. Dans *2015 49th Asilomar Conference on Signals, Systems and Computers*, 459–463. IEEE.

Das, P. et Khilar, P. M. (2013). Vft : A virtualization and fault tolerance approach for cloud computing. Dans *2013 IEEE Conference on Information & Communication Technologies*, 473–478. IEEE.

Das, S. et Suganthan, P. N. (2010). Differential evolution : A survey of the state-of-the-art. *IEEE transactions on evolutionary computation*, *15*(1), 4–31.

Dhanvijay, M. M. et Patil, S. C. (2019). Internet of things : A survey of enabling technologies in healthcare and its applications. *Computer Networks*.

Dieye, M. (2016). Disponibilité des données dans les centres de données à caractéristiques hétérogènes.

Diouf, G. M. (2019). Une plateforme d'orchestration de conteneurs docker tolérante aux fautes byzantines.

Elsayed, E. (1996). Reliability engineering (addisonwesley longman, inc., reading, ma).

Farris, I., Taleb, T., Bagaa, M. et Flick, H. (2017). Optimizing service replication for mobile delay-sensitive applications in 5g edge network. Dans *2017 IEEE International Conference on Communications (ICC)*, 1–6. IEEE.

Felter, W., Ferreira, A., Rajamony, R. et Rubio, J. (2015). An updated performance comparison of virtual machines and linux containers. Dans *2015 IEEE international symposium on performance analysis of systems and software (ISPASS)*, 171–172. IEEE.

Gia, T. N., Jiang, M., Rahmani, A.-M., Westerlund, T., Liljeberg, P. et Tenhunen, H. (2015). Fog computing in healthcare internet of things : A case study on ecg feature extraction. Dans *2015 IEEE international conference on computer and information technology ; ubiquitous computing and communications ; dependable, autonomic and secure computing ; pervasive intelligence and computing*, 356–363. IEEE.

Gibson, B. S. G. A. (2007). Disk failures in the real world : What does an mttf of 1,000,000 hours mean to you ? FAST.

Gill, P., Jain, N. et Nagappan, N. (2011). Understanding network failures in data centers : measurement, analysis, and implications. Dans *Proceedings of the ACM SIGCOMM 2011 conference*, 350–361.

Goiri, Í., Julia, F., Guitart, J. et Torres, J. (2010). Checkpoint-based fault-tolerant infrastructure for virtualized service providers. Dans *2010 IEEE Network Operations and Management Symposium-NOMS 2010*, 455–462. IEEE.

Granados, J., Rahmani, A.-M., Nikander, P., Liljeberg, P. et Tenhunen, H. (2014). Towards energy-efficient healthcare : An internet-of-things architecture using intelligent gateways. Dans *2014 4th International Conference on Wireless Mobile Communication and Healthcare-Transforming Healthcare Through Innovations in Mobile and Wireless Technologies (MOBIHEALTH)*, 279–282. IEEE.

Guillemin, P., Friess, P. *et al.* (2009). Internet of things strategic research roadmap. *The Cluster of European Research Projects, Tech. Rep*.

He, J., Wei, J., Chen, K., Tang, Z., Zhou, Y. et Zhang, Y. (2017). Multitier fog computing with large-scale iot data analytics for smart cities. *IEEE Internet of Things Journal*, *5*(2), 677–686.

Hochba, D. S. (1997). Approximation algorithms for np-hard problems. *ACM Sigact News*, *28*(2), 40–52.

Hu, P., Dhelim, S., Ning, H. et Qiu, T. (2017). Survey on fog computing : architecture, key technologies, applications and open issues. *Journal of network and computer applications*, *98*, 27–42.

Huang, Y.-M., Hsieh, M.-Y., Chao, H.-C., Hung, S.-H. et Park, J. H. (2009). Pervasive, secure access to a hierarchical sensor-based healthcare monitoring architecture in wireless heterogeneous networks. *IEEE journal on selected areas in communications*, *27*(4), 400–411.

Jardine, R., Wright, J., Samad, Z. et Bhutta, Z. A. (2020). Analysis of covid-19 burden, epidemiology and mitigation strategies in muslim majority countries. *Eastern Mediterranean Health Journal*, *26*(10).

Javed, A. *et al.* (2016). Container-based iot sensor node on raspberry pi and the kubernetes cluster framework.

Joy, A. M. (2015). Performance comparison between linux containers and virtual machines. Dans *2015 International Conference on Advances in Computer Engineering and Applications*, 342–346. IEEE.

Kobusińska, A., Leung, C., Hsu, C.-H., Raghavendra, S. et Chang, V. (2018). Emerging trends, issues and challenges in internet of things, big data and cloud computing.

Kraemer, F. A., Braten, A. E., Tamkittikhun, N. et Palma, D. (2017). Fog computing in healthcare-a review and discussion. *IEEE Access*, *5*, 9206–9222.

Krasich, M. (2009). How to estimate and use mttf/mtbf would the real mtbf please stand up ? Dans *2009 Annual Reliability and Maintainability Symposium*, 353–359. IEEE.

Lloyd's (2018). Lloyd's emerging risk report. *URL :https ://www.lloyds.com/ /media/files/news-and-insight/risk-insight/2018/cloud-down/aircyberlloydspublic2018final.pdf*.

López, G., Custodio, V. et Moreno, J. I. (2010). Lobin : E-textile and wireless-sensor-network-based platform for healthcare monitoring in future hospital environments. *IEEE Transactions on Information Technology in Biomedicine*, *14*(6), 1446–1458.

Madsen, H., Burtschy, B., Albeanu, G. et Popentiu-Vladicescu, F. (2013). Reliability in the utility computing era : Towards reliable fog computing. Dans *2013 20th International Conference on Systems, Signals and Image Processing (IWSSIP)*, 43–46. IEEE.

Mandellos, G. J., Lymberopoulos, D. K., Koutelakis, G. V., Koukias, M. N. et Panagiotakopoulos, T. C. (2009). *Requirements and solutions for advanced Telemedicine applications*. INTECH Open Access Publisher.

Masip-Bruin, X., Marín-Tordera, E., Alonso, A. et Garcia, J. (2016). Fog-to-cloud computing (f2c) : The key technology enabler for dependable e-health services deployment. Dans *2016 Mediterranean ad hoc networking workshop (Med-Hoc-Net)*, 1–5. IEEE.

Matthias, K. et Kane, S. P. (2015). *Docker : Up & Running : Shipping Reliable Containers in Production*. " O'Reilly Media, Inc.".

Mavrogiorgou, A., Kiourtis, A., Symvoulidis, C. et Kyriazis, D. (2018). Capturing the reliability of unknown devices in the iot world. Dans *2018 Fifth International Conference on Internet of Things : Systems, Management and Security*, 62–69. IEEE.

Meyer, D. (2008). The locator identifier separation protocol (lisp). *The Internet Protocol Journal*, *11*, 23–36.

Mizumoto, K., Kagaya, K., Zarebski, A. et Chowell, G. (2020). Estimating the asymptomatic proportion of coronavirus disease 2019 (covid-19) cases on board the diamond princess cruise ship, yokohama, japan, 2020. *Eurosurveillance*, *25*(10), 2000180.

Mohsin, A., Zaidan, A., Zaidan, B., Albahri, A., Albahri, O., Alsalem, M. et Mohammed, K. (2018). Real-time remote health monitoring systems using body sensor information and finger vein biometric verification : A multi-layer systematic review. *Journal of medical systems*, *42*(12), 238.

Morabito, R., Kjällman, J. et Komu, M. (2015). Hypervisors vs. lightweight virtualization : a performance comparison. Dans *2015 IEEE International Conference on Cloud Engineering*, 386–393. IEEE.

Mseddi, A., Jaafar, W., Elbiaze, H. et Ajib, W. (2019). Joint container placement and task provisioning in dynamic fog computing. *IEEE Internet of Things Journal*, 1–1. http://dx.doi.org/10.1109/JIOT.2019.2935056

Oma, R., Nakamura, S., Duolikun, D., Enokido, T. et Takizawa, M. (2018). Fault-tolerant fog computing models in the iot. Dans *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, 14–25. Springer.

Ozeer, U., Etchevers, X., Letondeur, L., Ottogalli, F.-G., Salaün, G. et Vincent, J.-M. (2018). Resilience of stateful iot applications in a dynamic fog environment. Dans *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems : Computing, Networking and Services*, 332–341.

O'Neill, G. et Tolley, N. S. (2020). Cochlear implant reliability : Reporting of device failures. *Indian Journal of Otolaryngology and Head & Neck Surgery*,1–3.

Rahmani, A. M., Gia, T. N., Negash, B., Anzanpour, A., Azimi, I., Jiang, M. et Liljeberg, P. (2018). Exploiting smart e-health gateways at the edge of healthcare internet-of-things : A fog computing approach. *Future Generation Computer Systems*, *78*, 641–658.

Remuzzi, A. et Giuseppe, R. (2020). Covid-19 and italy : what next ? *The Lancet*.

Rosen, R. (2013). Resource management : Linux kernel namespaces and cgroups. *Haifux, May*, *186*, 70.

Ross, G. T. et Soland, R. M. (1975). A branch and bound algorithm for the generalized assignment problem. *Mathematical programming*, *8*(1), 91–103.

Rothan, H. A. et Byrareddy, S. N. (2020). The epidemiology and pathogenesis of coronavirus disease (covid-19) outbreak. *Journal of autoimmunity*, p. 102433.

Saranya, S., Srimathi, T., Ramanathan, C. et Venkadesan, T. (2015). Enhanced fault tolerance and cost reduction using task replication using spot instances in cloud. *International Journal of Innovative Research in Science, Engineering and Technology*, *4*(6), 12–16.

Schroeder, B., Damouras, S. et Gill, P. (2010). Understanding latent sector errors and how to protect against them. *ACM Transactions on storage (TOS)*, *6*(3), 1–23.

Sharma, M., Komninos, A., López-Ibáñez, M. et Kazakov, D. (2019). Deep reinforcement learning based parameter control in differential evolution. Dans *Proceedings of the Genetic and Evolutionary Computation Conference*, 709–717.

Skarlat, O., Schulte, S., Borkowski, M. et Leitner, P. (2016). Resource provisioning for iot services in the fog. Dans *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*, 32–39.

Soltesz, S., Pötzl, H., Fiuczynski, M. E., Bavier, A. et Peterson, L. (2007). Container-based operating system virtualization : a scalable, high-performance alternative to hypervisors. Dans *Proceedings of the 2Nd ACM SIGOPS/EuroSys european conference on computer systems 2007*, 275–287.

Souza, V. B., Masip-Bruin, X., Marín-Tordera, E., Ramírez, W. et Sánchez-López, S. (2017). Proactive vs reactive failure recovery assessment in combined fog-to-cloud (f2c) systems. Dans *2017 IEEE 22nd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, 1–5. IEEE.

Sun, H. et Han, J. J. (2001). Instantaneous availability and interval availability for systems with time-varying failure rate : stair-step approximation. Dans *Proceedings 2001 Pacific Rim International Symposium on Dependable Computing*, 371–374. IEEE.

Topol, E. (2012). *The creative destruction of medicine : How the digital revolution will create better health care*. Basic Books.

Turnbull, J. (2014). *The Docker Book : Containerization is the new virtualization*. James Turnbull.

Vaquero, L. M. et Rodero-Merino, L. (2014). Finding your way in the fog : Towards a comprehensive definition of fog computing. *ACM SIGCOMM Computer Communication Review*, *44*(5), 27–32.

Yan, H., Xu, L. D., Bi, Z., Pang, Z., Zhang, J. et Chen, Y. (2015). An emerging technology–wearable wireless sensor networks with applications in human health condition monitoring. *Journal of Management Analytics*, *2*(2), 121–137.

Yi, S., Hao, Z., Qin, Z. et Li, Q. (2015). Fog computing : Platform and applications. Dans *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, 73–78.

Yi, S., Li, C. et Li, Q. (2015). A survey of fog computing : concepts, applications and issues. Dans *Proceedings of the 2015 workshop on mobile big data*, 37–42.

Zhang, J., Zhou, A., Sun, Q., Wang, S. et Yang, F. (2018). Overview on fault tolerance strategies of composite service in service computing. *Wireless Communications and Mobile Computing*, *2018*.

Zhu, J., Chan, D. S., Prabhu, M. S., Natarajan, P., Hu, H. et Bonomi, F. (2013). Improving web sites performance using edge servers in fog computing architecture. Dans *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*, 320–323.