République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid Tlemcen
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études

Pour l'obtention du diplôme de Master en Informatique

Option : Réseaux et Systèmes Distribués (RSD)

Thème

Tolérance aux pannes dans les systèmes distribués : application au Cloud computing

Réalisé par :

- YOUSFI Amira

- BOUAYAD DEBBAGH Fatiha

Présenté le 23 Juin 2020 devant le jury composé de :

- Mr LEHSAINI Mohamed — Président
- Mr BENMOUNA Youcef — Examinateur
- Mr BENMAMMAR Badr — Encadrant
- Mr SEMMOUD Abderrazak — Co-Encadrant

Année universitaire 2019-2020

Remerciements

Nous tenons tout d'abord à remercier le grand Dieu et le tout puissant, qui nous a donné la force et la patience d'accomplir ce modeste travail;

On commence par nos très chers parents, c'est grâce à leurs prières et encouragement qu'on a pu dépasser tous les obstacles;

Nos remerciements vont en priorité à notre encadrant,

Mr BENMAMMAR BADR, pour sa patience et surtout sa disponibilité malgré ses préoccupations, ainsi que son suivi minutieux de notre travail avec un haut niveau de précision.

Mr SEMMOUD ABDEREZZAK, pour son énorme aide et son conseil, ainsi que ses encouragements.

Mr HAKEM MOURAD, on salue son amabilité, sa souplesse d'esprit et son savoir.

On remercie le chef de département **Mr MATALLAH HOUCINE**, pour ses efforts afin de nous fournir un environnement sain et confortable;

On tient à remercier le jury composé de **Mr LEHSAINI MOHAMED** et **Mr BENMOUNA Youcef** pour leur présence très honorée.

Enfin, on remercie énormément toutes les personnes qui ont participés dans ce travail, que dieu vous bénisses.

Dédicaces

Après le remercîment d'Allah, Je dédie ce travail à :

Mon très cher père,

Ma très chère mère,

Aucun mot et aucune expression ne peut exprimer mon grand amour, mon énorme respect et reconnaissance tant vers vous deux,

J'ai senti votre présence dans toutes les étapes de ma vie, avec des encouragements si chaleureux, vos prières et vos Doua m'ont toujours protégé et m'ont aidé à battre et à continuer de travail jusqu'à ce que j'attint mes objectifs, vous étiez et vous restez ma fierté et mon exemple jusqu'à la fin de ma vie.

Ce travail est dédié aussi,

Mes frères et ma sœur ainsi que mes amis,

Je vous remercie pour votre soutien et votre patience avec moi, vous êtes les meilleures personnes au monde.

Toute personne qui m'a aidé et m'a souhaité un bon courage, je vous remercie infiniment et que Dieu vous protège.

Je vous aime tous et je pris que vous restez en bonne santé et bonheur.

Yousfi Amira.

Dédicaces

Merci Allah de m'avoir donnée le courage et la patience pour que je puisse arriver à l'un de mes objectifs dans cette vie.

Du profond de mon cœur, je dédié ce travail à tous ceux qui mon chers,

A mes deux perles papa et maman,

Aucune dédicace ne saurait exprimer mon respect, mon amour infini, vous m'avais doté d'une éducation digne avec plein d'amour, vous êtes ceux qui mon jamais laisser seule, vous étiez mon école et mon exemple et vous serez pour tout le reste de ma vie, ce modeste travail soit l'exaucement de vos veux, et le fruit de vos sacrifices, que dieu vous gardez pour moi avec une bonne santé. Je vous aime.

A mes deux frères et mes amis,

Je vous remercie d'avoir supporté mon stresse et des fois ma faiblesse, et de m'avoir soutenu jusqu'au bout. Ma vie ne serait pas aussi belle son votre présence. Merci à toi Amira c'était un énorme plaisir de travail avec toi ma chérie.

A la fin, je dédié toutes personne m'a appris une leçon ou m'a donnée un conseil, que dieu vous protèges.

Bouayad DEBBAGH Fatiha.

Table des matières

Liste des figures

Liste des tableaux

Acronymes

CIS	Cloud Information Services
DC	Data Center
IAAS	Infrastructure As A Service
MDT	Mean Down Time
MIS	Maximal Independent Set
MTTF	Mean Time To Failure
MTTR	Mean Time To Restore
LAN	Local Area Network
PAAS	Platform As A Service
SAAS	Software As A Service
VM	Virtuel Machine
VPN	Virtual Private Network
WAN	Wireless Area Network

Résumé

La tolérance aux pannes dans un système distribué est l'aptitude de ce système à accomplir sa fonction malgré la présence de pannes. Le Cloud computing est un type de système distribué qui doit empêcher toute dégradation de son service en cas de panne en utilisant des techniques pour les tolérer. Dans ce travail, nous avons implémenté et évalué à l'aide du simulateur CloudSim une approche de tolérance aux pannes appliquée au Cloud computing. Notre approche est basée essentiellement sur la réplication hybride ainsi que sur le principe du Maximal Independent Set pour le choix des machines virtuelles afin d'effectuer la réplication sur la machine la moins chargée. Les résultats obtenus de ce travail sont très satisfaisants et montrent l'efficacité de notre solution en termes de taux de tâches exécutées.

Mots-clés : Cloud computing, tolérance aux pannes, systèmes distribués, réplication, MIS, CloudSim.

Abstract

Fault tolerance in a distributed system is the ability of that system to perform its function despite the presence of failures. Cloud computing is a type of distributed system that must prevent degradation of its service in the event of a breakdown by using techniques to tolerate it. In this work, we implemented and evaluated using the CloudSim simulator a fault tolerance approach applied to Cloud computing. Our approach is essentially based on hybrid replication as well as on the principle of the Maximal Independent Set for the choice of virtual machines in order to perform replication on the least loaded machine. The obtained results of our work are very satisfactory and demonstrate the effectiveness of our solution in terms of performed tasks rate.

Keywords: Cloud computing, fault tolerance, distributed systems, replication, MIS, CloudSim.

ملخص

التسامح مع الخطأ في النظام الموزع هو قدرة هذا النظام على أداء وظيفته على الرغم من وجود حالات الفشل. الحوسبة السحابية هي نوع من النظام الموزع الذي يجب أن يمنع تدهور خدمتها في حالة حدوث انهيار باستخدام تقنيات لتحملها. في هذا العمل ، قمنا بتطبيق وتقييم باستخدام محاكي CloudSim نهج تحمل الخطأ المطبّق على الحوسبة السحابية. يعتمد نهجنا بشكل أساسي على النسخ الهجين وكذلك على مبدأ المجموعة المستقلة القصوى لاختيار الأجهزة الافتراضية من أجل إجراء النسخ المتماثل على الجهاز الأقل تحميلًا. النتائج التي تم الحصول عليها من هذا العمل مرضية للغاية وتظهر فعالية حلنا من حيث معدل المهام المنجزة.

الكلمات المفتاحية : الحوسبة السحابية ، تحمل الأخطاء ، الأنظمة الموزعة ، النسخ المتماثل ,CloudSim, MIS.

Introduction générale

La demande croissante de nouveaux services informatiques a permis l'apparition d'une nouvelle architecture qui est l'informatique dans les nuages ou bien le Cloud computing.

En 2008, le monde a vu l'émergence du terme « Cloud computing » dans les revues spécialisées, et cette technologie a été présentée comme l'une des solutions potentielles de sauvegarde de données chez les grandes sociétés spécialisées dans le domaine de l'informatique telles que : Google, Microsoft…etc. Le Cloud computing est un modèle de distribution de ressources virtualisées, dynamiques et configurables. Il est considéré comme un ensemble de machines virtuelles qui utilisent un seul support ou infrastructures physique. Le Cloud computing offre à ses utilisateurs l'accès aux services via le réseau internet.

Comme tous services informatiques, les services du Cloud computing sont vulnérables aux défaillances, et donc nécessitent des mécanismes de tolérance aux pannes. Un système est tolérant aux pannes s'il arrive à grader son bon fonctionnement et sa production malgré l'existence de pannes.

La tolérance aux pannes a eu beaucoup de mécanismes et politiques afin de rendre plus fiable le système et d'éviter au maximum la perte de ses données. Les recherches et les études dans ce domaine sont en cours afin d'obtenir une tolérance satisfaisante et pourquoi pas parfaite. Les techniques de la tolérance aux fautes sont généralement basées sur deux méthodes : la réplication et la redondance.

Dans le cadre de ce projet de fin d'études, nous avons implémenté et évalué une approche de tolérance aux pannes appliquée au Cloud computing. Cette approche est basée sur la réplication hybride ainsi que sur le principe du Maximal Independent Set pour le choix des VMs pour effectuer la réplication.

Les simulations et évaluations effectuées dans le cadre de ce PFE ont été réalisées en utilisant le simulateur CloudSim. Ce dernier est un Framework pour la modélisation et la simulation des environnements Cloud.

Les résultats obtenus de notre travail sont satisfaisants car notre approche a permis d'améliorer le taux d'exécution des tâches soumises à l'exécution.

Organisation du manuscrit :

Le présent rapport est structuré autour de trois chapitres qui se résument comme suit :

- Le premier chapitre présente le principe des systèmes distribués et les caractéristiques du Cloud computing.

- Le deuxième chapitre présente les notions fondamentales ainsi que les différentes solutions liées à la tolérance aux pannes dans les systèmes distribués.

- Le troisième et dernier chapitre présente notre contribution dans le cadre de ce PFE.

Chapitre 1 : Cloud computing

Chapitre 1 : Cloud computing

1.1 Introduction

Suite à l'accroissement des besoins des entreprises et surtout celles multi-sites qui demandent de plus en plus de meilleures performances en termes de fiabilité des communications/calculs, de sécurité et de disponibilité, les systèmes distribués sont qualifiés pour répondre à tous ces besoins grâce à l'efficacité de leurs architectures informatiques qui permettent d'améliorer les performances et la disponibilité des systèmes informatiques en général.

L'Internet est le cas typique d'un système distribué qui a été utilisé dans ce contexte et qui a donné naissance au concept de « Cloud computing ». Ce dernier a d'autres appellations comme : l'informatique dans les nuages, l'informatique dématérialisé, ou encore l'infonuagique.

L'objectif du Cloud computing est de se débarrasser de la centralisation et des traitements traditionnels qui se situent sur un seul poste utilisateur et d'intégrer la notion de la distribution sur les sites distants ce qui forme un nuage ou « Cloud ».

L'objectif de ce chapitre est de donner un aperçu global sur les systèmes distribués en se focalisant sur la présentation du Cloud computing.

1.2 Les systèmes distribués

Un système distribué peut être considéré comme un réseau dont lequel il existe un ensemble de machines où chacune se situe dans un emplacement différent mais avec un objectif commun. Ces machines peuvent être soit homogènes ou hétérogènes, distribuées globalement ou localement. A titre d'exemple, l'internet est un exemple de système distribué ne possédant aucun nœud centralisé.

Il existe plusieurs catégories de système distribué.

La figure 1.1 donne un aperçu global sur ces catégories.

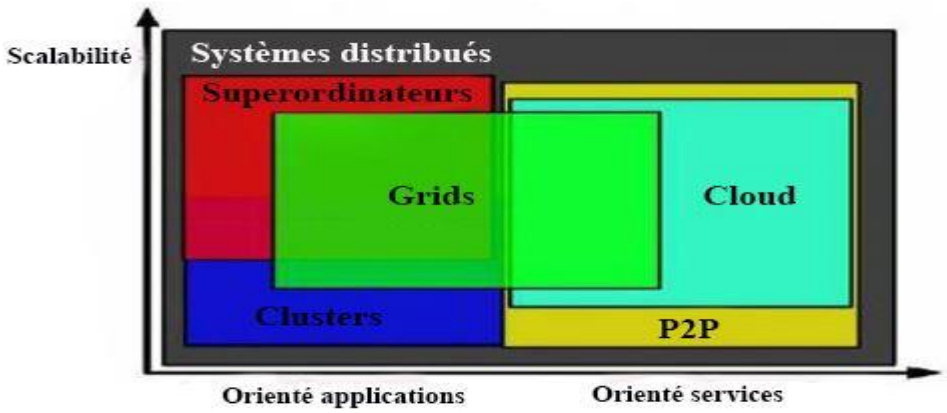

Figure 1.1 : Système distribué [15].

- **La grappe d'ordinateurs (Cluster) :** un ensemble d'ordinateurs indépendants connectés à un réseau local rapide et fiable afin de permettre une gestion globale et ainsi dépasser les limitations d'un seul ordinateur [1].
- **La grille informatique (Grid) :** un ensemble d'ordinateurs hétérogènes, situés sur différents sites et connectés par un réseau de type WAN où chacun a des ressources informatiques administrées de manière indépendante et uniforme.
- **Peer to Peer :** un modèle d'échange où chaque entité du réseau est à la fois client et serveur, contrairement au modèle client-serveur. Les termes « pair », « nœud », et « utilisateur » sont généralement utilisés pour désigner les entités composant un système pair-à-pair
- **Superordinateur** (supercalculateur) : un ordinateur avec de hautes performances surtout en termes de vitesse de calcul. Ce type de machines est utilisé par exemple pour l'étude de climat ou de simulations gourmandes en temps de calcul [2].
- **Cloud :** désigne le stockage et l'accès aux données par l'intermédiaire d'internet plutôt que via le disque dur d'un ordinateur.

Le reste de ce chapitre sera consacré à la présentation du Cloud computing, le type de système distribué sur lequel repose la réalisation de notre projet de fin d'études.

1.3 Cloud computing

1.3.1 Définition

Le Cloud computing est représenté sous forme d'un nuage de services et de données. Le terme « Cloud » est utilisé comme métaphore pour « Internet ». Il est basé sur la façon dont Internet est conventionnellement représenté (un nuage) afin de masquer la complexité de son architecture, ainsi que le terme « computing » pour représenter l'existence de l'informatique.

Selon NIST (l'institut des standards et technologie des États-Unis) le Cloud computing représente l'accès à un réservoir de partage de ressources informatiques configurables (serveurs, stockage, applications et services) situés sur un réseau à travers Internet [1]. Grâce à l'existence d'internet plusieurs machines distantes peuvent communiquer.

Cette technique offre la possibilité d'utiliser une puissance de calcul et une capacité de stockage offerte par des serveurs distants [3].

La figure 1.2 donne un aperçu global sur le Cloud computing.

Figure 1.2 : Cloud computing [16].

1.3.2 Historique du Cloud computing

L'histoire du Cloud computing a commencé autours des années 1960 où ses premières traces ont apparu quand John McCarthy affirmait que cette puissance de traitement informatique serait accessible au public dans le futur. Le terme en lui-même est apparu plus couramment aux alentours de la fin du XXe siècle.

La naissance officielle du Cloud computing a été en 1990 avec des options très limitées, parmi ses premières offres, l'existence d'un seul espace de stockage pour l'échange de fichiers où plusieurs utilisateurs peuvent travailler ensemble sur un document [12].

En 1991, le Cloud computing a connu une nouvelle étape très importante avec la naissance de l'Internet et aussi la mise en marche du logiciel développé par CERN (centre européen de recherche nucléaire) car ce dernier a été le premier logiciel accessible sur le Web. L'idée du Cloud computing a été progressée en 1993 lors du lancement du navigateur Mosaic puis le navigateur Netscape en 1994. Ensuite en 1995, le processus a été accéléré grâce à eBay et Amazon. En 1996, une nouvelle vie et un nouveau sens a été donné à ce processus lors du lancement de l'assistant Palm PDA [13].

En 2002, Il est communément admis que le concept de Cloud computing a été initié par le géant d'e-commerce Amazon. Ensuite en 2008, le Cloud computing est devenu l'un des technologies les plus « à la mode » et donc plusieurs entreprises comme IBM et Google ont commencé à s'y intéresser, car ils ont constaté l'importance de cette technologie qui offre plusieurs services et facilite beaucoup de tâches aux entreprises à travers l'Internet.

En 2009, 10% des entreprises interrogées utilisaient des services de Cloud computing dans le domaine de l'hébergement de leurs applications informatiques et infrastructures, ce qui prouve que cette technique a réussi à faire une évolution majeure en peu de temps.

Jusqu'à présent, le Cloud computing a pu garder son importance dans le domaine informatique, car il a continué à développer ses méthodes tout en respectant les besoins des utilisateurs et des grandes entreprises.

1.3.3 Modèles de services

Il existe trois modèles principaux de Cloud computing où chaque partie représente une entité importante dans le Cloud.

La figure 1.3 illustre ces modèles.

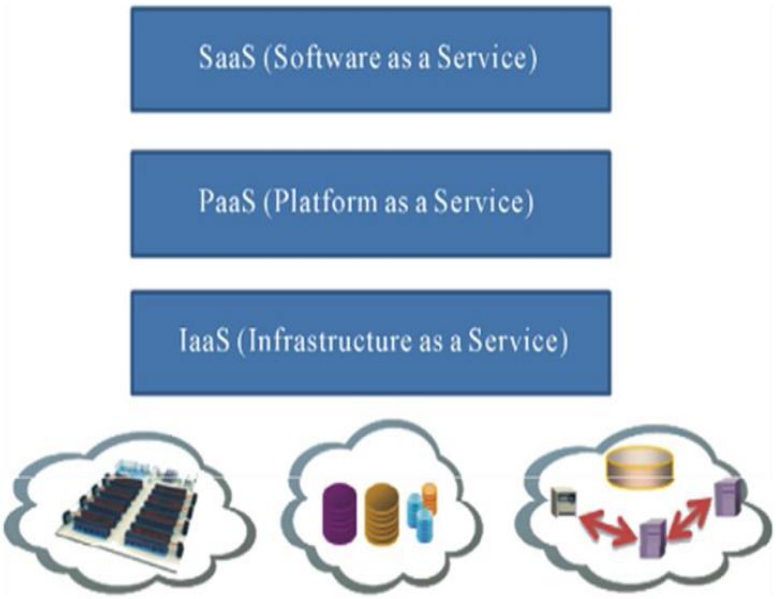

Figure 1.3 : Modèles de services du Cloud computing [14].

- **IaaS (Infrastructure as a Service) :** ou « l'infrastructure en tant que service » est la base du Cloud computing. C'est à ce niveau que le choix des systèmes d'exploitation et les serveurs ainsi que l'accès à des fonctionnalités de mise en réseau, à des ordinateurs (virtuels ou sur du matériel dédié) et à de l'espace de stockage de données a lieu. De plus, cette couche offre un haut niveau de flexibilité et de contrôle sur des ressources informatiques [4]. Amazon Elastic Compute Cloud (Amazon EC2) est un exemple de IaaS.
- **PaaS (Platform as a Service) :** ou « Plateforme en tant que Service » en Français. Ces plates-formes sont directement utilisables par des éditeurs qui proposeront leurs logiciels en mode Cloud. Elle offre des services Cloud au développeurs des applications afin de faciliter le déploiement de leurs applications grâce à un ensemble d'outils comme les langages de programmation, des services et des bibliographie…etc. [5], donc cette partie est conçue pour le développement et la programmation, exemple de PaaS : Google App engine ou App Fog...
- **SaaS (Software as a Service) :** ou « Logiciel en tant que Service » qui est un modèle de fourniture de logiciels hébergés à distance.
  Un client ou un utilisateur n'a aucune communication directe avec les couches PaaS et IaaS car il exécute les applications directement grâce à la couche SaaS par le biais d'internet. La phase d'utilisation des applications est faite à travers une interface client légère, comme un navigateur Web (exemple : le courrier électronique) ou une interface spéciale. Google Apps et OnLive sont des exemples de SaaS [1].

La figure 1.4 résume et représente les trois niveaux du Cloud computing.

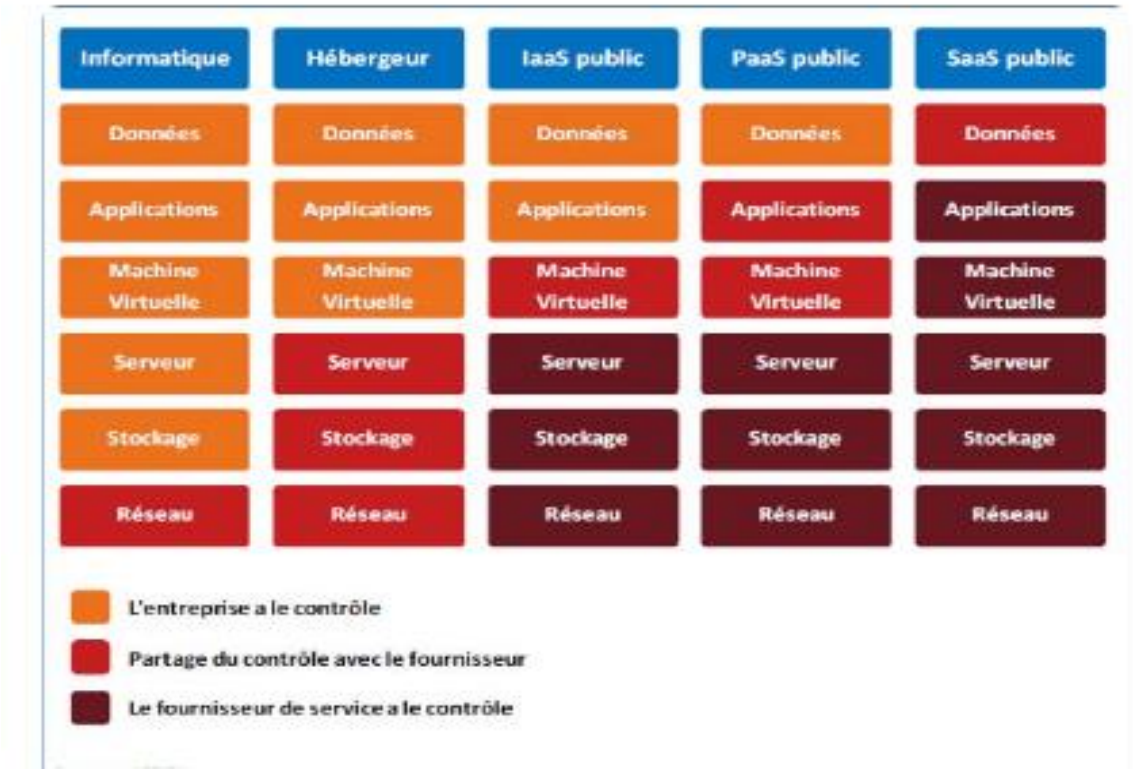

Figure 1.4 : Niveaux du Cloud computing [5].

Le tableau1.1 synthétise les avantages et les inconvénients de chaque niveau de service du Cloud computing.

	Avantages	Inconvénients
SaaS	▪ Pas d'installation ▪ Plus de licence ▪ Migration	▪ Logiciel limité ▪ Sécurité ▪ Dépendance des prestataires
PaaS	▪ Pas d'infrastructure nécessaire ▪ Pas d'installation ▪ Environnement hétérogène	▪ Limitation des langages ▪ Pas de personnalisation dans la configuration des machines
IaaS	▪ Administration ▪ Personnalisation ▪ Flexibilité d'utilisation	▪ Sécurité ▪ Besoin d'un administration système

Tableau 1.1 : Avantages et inconvénients des niveaux du Cloud computing [3].

1.3.4 Modèles de déploiement

Il existe différents types de Cloud computing. Le choix du type correspond au besoin des entreprises et selon leurs fonctionnements. Nous pouvons distinguer quatre types de déploiement pour le Cloud (public, privé, hybride et communautaire).

La figure 1.5 donne une vision générale sur ces types.

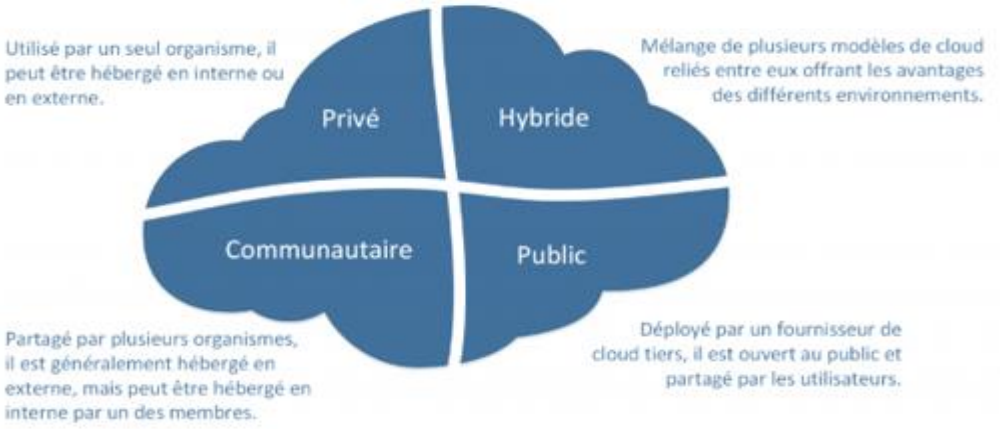

Figure 1.5 : Types de Cloud computing [6].

1.3.4.1 Cloud public

L'infrastructure d'un nuage public est accessible publiquement ; de nombreux utilisateurs et clients sont gérés par la société qui met à dispositions des services. A ce niveau, le client ou l'utilisateur n'a pas besoin d'avoir ni du matériel, ni des logiciels car tout est partagé.

Étant donné que l'infrastructure est partagée, le critère de sécurité est très important et il est géré par le fournisseur qui en assure la gestion dans le but de garder la confiance de l'utilisateur.

La figure 1.6 illustre ce type de Cloud.

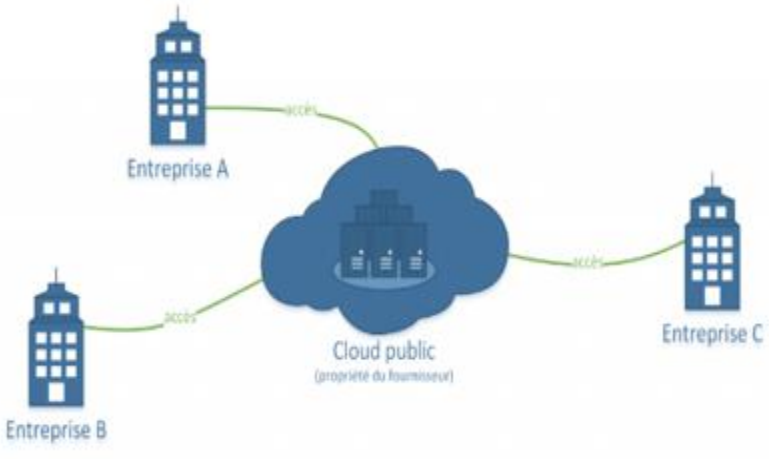

Figure 1.6 : Modèle de déploiement d'un Cloud public [6].

1.3.4.2 Cloud privé

L'infrastructure d'un nuage privé comme son nom l'indique est utilisée que par un seul client ou une seule entreprise dans le cas général. Les ressources appartiennent à un et un seul utilisateur et donc le domaine de sécurité est géré par cet utilisateur ce qui assure un haut niveau

de sécurité à travers des politiques et des normes de sécurité. Parmi les méthodes utilisées dans ce type, les réseaux sécurisés (VPN).

La figure 1.7 illustre ce type de Cloud.

Figure 1.7 : Modèle de déploiement d'un Cloud privé [6].

1.3.4.3 Cloud communautaire

L'infrastructure d'un nuage communautaire est partagée entre plusieurs utilisateurs ou plusieurs entreprises situées sur différents sites afin de produire ou de travailler sur un projet, ou faire des calculs communs par exemple. Parmi les exemples d'utilisation : les organismes gouvernementaux, les hôpitaux, etc.

La figure 1.8 illustre ce type de Cloud.

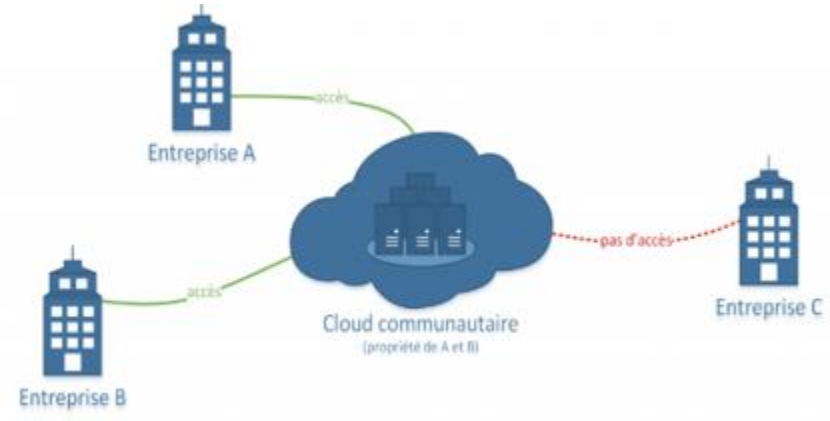

Figure 1.8 : Modèle de déploiement d'un Cloud communautaire [6].

1.3.4.4 Cloud hybride

L'infrastructure d'un nuage hybride et comme son nom l'indique est le mélange ou bien le fait de composer deux ou trois types de Cloud cités précédemment. Cette infrastructure prend que les avantages et les points forts des types composés. Par exemple, une entreprise qui utilise un

Cloud hybride peut bénéficier de la simplicité et le faible coût du Cloud public et de la sécurité du Cloud privé.

La figure 1.9 illustre ce type de Cloud.

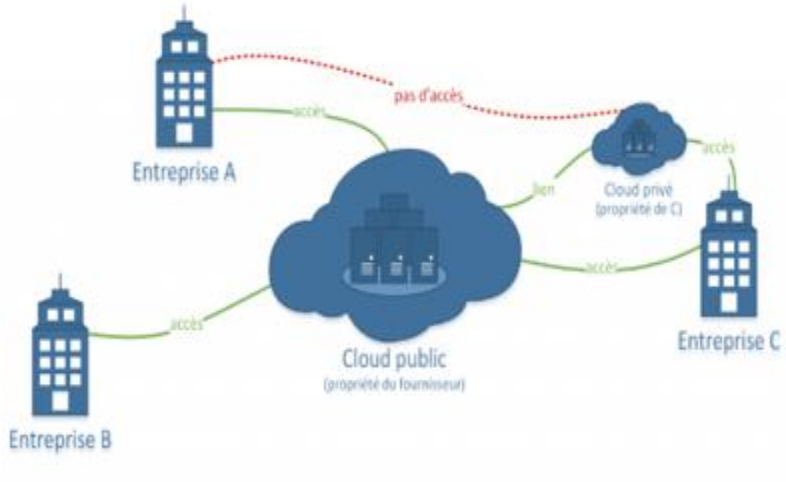

Figure 1.9 : Modèle de déploiement d'un Cloud hybride [6].

1.3.5 Caractéristiques du Cloud computing

L'environnement du Cloud computing a un ensemble de caractéristiques parmi lesquelles on peut citer [7] :

- **Le libre-service à la demande :** un des principales caractéristiques du Cloud computing, car il est lié à la consommation autonome du service, c.à.d. la liberté d'utilisation des ressources sans aucune intervention humaine.
- **L'accès aux ressources:** le fait de permettre à l'utilisateur d'accéder n'importe où et n'importe comment aux ressources.
- **Mise en commun des ressources :** les ressources sont partagées entre différents utilisateurs dans une entreprise, ce qui fait appel à un problème de localisation des données. Mais sur l'échelle économique, le cout est réduit et donc les dépenses des utilisateurs sont moindres.
- **Adaptabilité rapide (élasticité) :** le service qu'utilise le Cloud doit permettre à un utilisateur de déployer de nouvelles ressources. Le Cloud computing offre ainsi un moyen de fournir les ressources informatiques nécessaires à une évolution ou à un pic d'utilisation de ce service.
- **Paiement à l'usage :** l'utilisateur doit payer selon sa consommation du service et donc la facturation est liée à l'usage effectif du service [8].

1.3.6 Avantages du Cloud computing

Parmi les points forts du Cloud, nous pouvons citer [18]:

- **Utilisation transparente des ressources :** le Cloud computing élimine la nécessité de s'investir dans du matériel et du logiciel, de configurer et de gérer les centres de données. L'accès aux ressources se fait d'une manière transparente.
- **Fiabilité :** le Cloud computing offre un service fiable aux utilisateurs afin de manipuler leurs données.

- **Espace de stockage :** le Cloud computing offre un espace presque illimité pour stocker les données.

1.3.7 Inconvénients du Cloud computing

Malgré que le Cloud computing soit créé pour solutionner plusieurs problèmes surtout celles qui touchent les systèmes distribués, mais il a néanmoins des points faibles qu'on peut citer dans ce qui suit [19] :

- **Nécessite une bonne connexion internet :** vu que le réseau essentiel dans le Cloud computing est l'internet, alors en cas de panne de ce réseau ou d'une mauvaise connexion, l'entreprise peut être perturbée.
- **Coût :** en Cloud il n'y a pas que les frais de stockage, il faut prendre en considération les frais de transferts qui peuvent être importants.
- **La sécurité** : les réseaux informatiques sont potentiellement vulnérables et les services virtuels peuvent être mis hors service si les fournisseurs de l'entreprise ne les traitent pas convenablement.
- **Problèmes techniques :** malgré que le Cloud computing soit considéré comme une technologie fiable et qu'on peut garder nos données précieuses en toute sécurité, mais parfois le système peut être en dysfonctionnement et peut risquer de perdre quelques données.

1.4 Conclusion

Dans ce chapitre, nous avons fourni un aperçu sur les notions de base des systèmes distribués en se focalisant sur le Cloud computing qui est le contexte de travail de notre PFE. Nous avons présenté la définition du Cloud, son historique, ses caractéristiques, ses modèles de déploiement, ses avantages ainsi que ses inconvénients.

Le chapitre suivant sera consacré à la présentation de la tolérance aux pannes dans les systèmes distribués en particulier dans le Cloud computing.

Chapitre 2 : Tolérance aux pannes dans les systèmes distribués

Chapitre 2 : Tolérance aux pannes dans les systèmes distribués

2.1 Introduction

Les systèmes distribués ont approuvé leurs progrès remarquables dans le domaine de l'informatique grâce à leurs nombreuses techniques de distribution. Le développement de ces systèmes risque d'être attaqué ou interrompu à cause des pannes qui perturbent le comportement normal du système.

La panne dans un système distribué est le fait de paralyser ou perturber une des fonctionnalités du système. La panne peut être une interruption de communication, des défaillances de machines, de périphériques de stockage ou autres. Le système qui tolère les pannes doit pouvoir fonctionner sous une forme dégradée même si une panne apparaît dans le système.

Les types de pannes et leurs effets ainsi que leurs origines se diffèrent d'une panne à une autre, d'où l'intérêt de ce chapitre qui consiste à présenter les notions de la tolérance aux pannes ainsi que les différentes solutions relatives à ce domaine.

2.2 Attributs de la sureté de fonctionnement

Un système distribué doit respecter certaines propriétés pour assurer son bon fonctionnement [9] :

- **Fiabilité (reliability) :** c'est la capacité du système à délivrer un service correct lorsqu'il est disponible.
- **Maintenabilité (maintainability) :** c'est la capacité du système à faire évoluer sa structure ou son état pour faciliter le retour à un état disponible/fiable, ou autrement dit lorsque l'entité est caractérisée par sa maintenabilité.
- **Disponibilité (availability) :** c'est la capacité du système à offrir tout ou une partie du service, dans des conditions données, à un instant donné.

Selon [17], nous pouvons donner les définitions suivantes **:**

MTTF (Mean Time To Failure) est le temps moyen avant la première défaillance.
MTTR (Mean Time To Restore/Restauration) est le temps moyen entre la dernière défaillance de l'entité et sa remise en service.
MDT (Mean Down Time) est le temps moyen où l'entité n'est pas en état d'accomplir une fonction requise.
MUT (Mean Up Time) est la durée moyenne de fonctionnement après réparation.
MTBF (Mean Time Between Failures) est la durée moyenne des temps de bon fonctionnement entre deux défaillances consécutives.

La figure 2.1 résume les différentes phases lors de l'occurrence d'une panne.

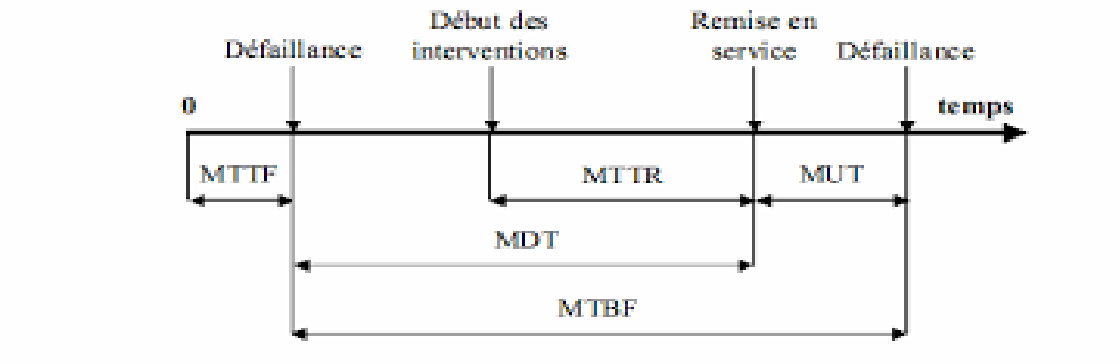

Figure 2.1: Les différentes phases lors de l'occurrence d'une panne [10].

- ***La sécurité (safety) :*** est l'aptitude d'une entité à éviter de faire apparaître, dans des conditions données, des événements critiques ou catastrophiques. Parmi les critères de sécurité utilisés on peut citer *:*

  - ***Sécurité-innocuité (safety-innocuity) :*** est la maitrise des conséquences d'un comportement du système (attendu ou non).
  - ***Intégrité (safety-integrity) :*** est la capacité du système à détecter ou empêcher toute altération de sa structure ou de son état en contradiction avec le comportement attendu.

La figure 2.2 illustre les liens entre fiabilité, maintenabilité, disponibilité et sécurité.

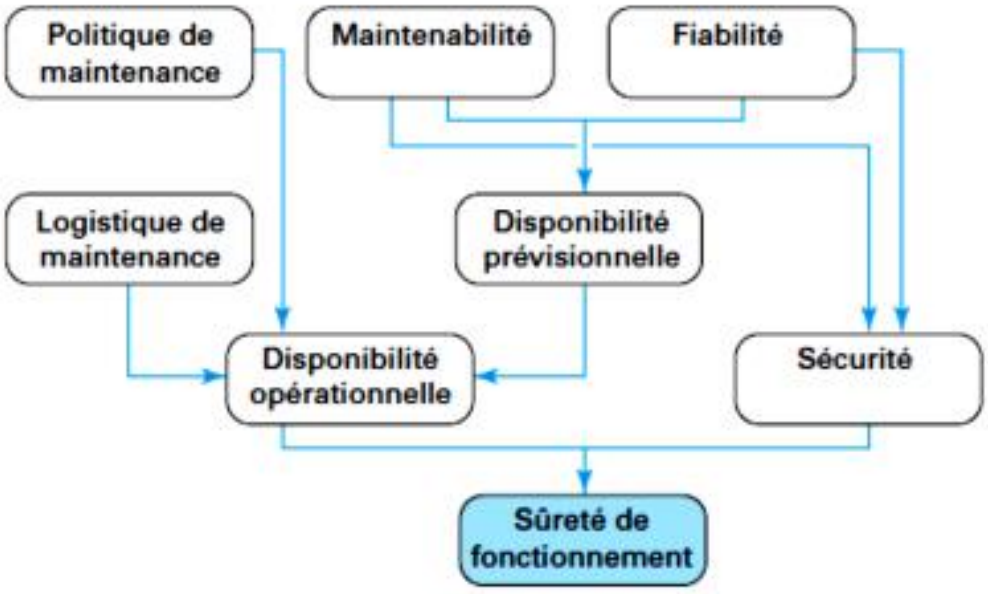

Figure 2.2*:* Les liens entres les attributs [11].

2.3 Limites de la sureté de fonctionnement

Les limites ou les entraves de la sureté de fonctionnement sont classifiées en trois catégories : les fautes, les erreurs et les défaillances reliées par une relation de causalité qui forme une chaine.

La figure 2.3 représente la chaine de liaison entre les entraves de la sureté de fonctionnement.

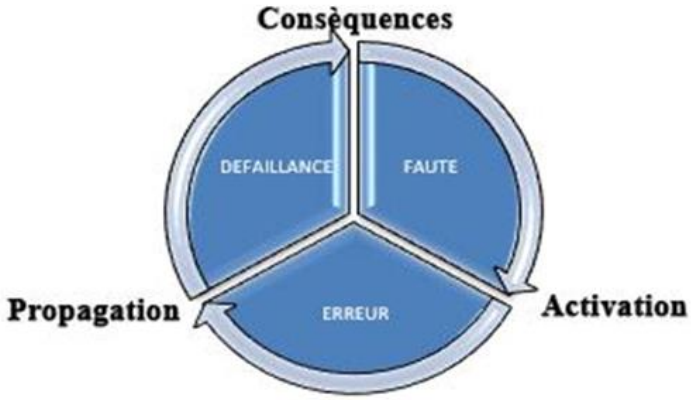

Figure 2.3 : Chaine des entraves de la sureté de fonctionnement [20]

- **Faute :** la faute est définie par un état anormal d'une unité fonctionnelle la mettant dans l'impossibilité d'accomplir une fonction requise [21]. Une faute ou panne est caractérisée par sa nature, son origine et son étendue temporelle. La nature d'une faute peut prendre deux états : intentionnelle (exprès dans le dessein de nuire) ou accidentelle (apparaisse d'une façon inattendue). De plus, la faute est caractérisée par une durée soit temporaire ou bien permanente dans une condition ponctuelle et une origine qui représente la source d'une faute. Elle peut être causée par un phénomène physique ou humain. Elle est provoquée soit par une partie de l'état du système soit par l'environnement. Elle appartient soit à la phase de conception soit à la phase d'exploitation du système [3].

- **Erreur :** comme indiquée sur la figure 2.3, l'erreur est une conséquence d'une faute et par la suite elle se transforme en une défaillance. Dans [27], les auteurs ont proposé une classification des erreurs selon deux critères : la valeur et le temps. Dans le premier cas (selon la valeur), l'erreur est dite non-codée si le service ne correspond pas en valeur à celui spécifié et si la valeur rendue n'est pas interprétable, sinon l'erreur est dite arbitraire. Dans le deuxième cas (selon le temps), le système ne peut pas faire une délivrance dans l'intervalle de temps spécifié. C'est-à-dire, il est soit en avance ou en retard. Le fait que le système omette de rendre le service attendu est considéré également comme une erreur.

- **Défaillance :** une défaillance représente l'incapacité d'un élément du système à assurer le service spécifié par l'utilisateur. La défaillance est caractérisée par son domaine, sa perception par les utilisateurs (cohérente ou pas) et ses conséquences sur l'environnement (mineures ou catastrophiques) [22].

2.4 Types de pannes

Quatre types de pannes peuvent survenir durant une exécution répartie. Elles sont présentées comme suit [23]:

- **Panne franche (fail-stop) :** est le type de panne le plus simple et non catastrophique car il peut se traiter facilement et sans une grande perte de données. En cas de panne, un processus va prendre l'un des deux cas suivants : soit il a planté avant l'envoi de ses résultats ou après l'envoi. Dans les deux cas, le processus est considéré comme défaillant et il cesse définitivement de fonctionner ou de générer de nouvelles requêtes jusqu'à sa réparation. Parmi les exemples de pannes franches : une coupure de voie physique (changement de topologie du système) ou système d'exploitation inter bloqué.

- **Pannes par omission ou transitoire (transient, omission failures) :** dans ce cas, le processus ne délivre jamais une réponse à un évènement en entré et il ne peut fournir son service habituel pendant une certaine période ce qui cause une perte de données. Ce type de pannes se divise en deux catégories : des pannes transitoires qui apparaissent une fois ou plus puis disparaisses, et des pannes intermittentes qui apparaissent de façon sporadique. Quelques exemples de pannes de ce type : la perte de messages à cause du bruit.

- **Pannes temporaires (timing, performance failures) :** ce sont des pannes qui ont un comportement anormal par rapport au temps qui est soit trop tôt soit trop tard. Quelques exemples de pannes de ce type : dépassement de délai pour répondre à des événements temps réel.

- **Pannes arbitraires ou byzantines (byzantine failures) *:*** ce type de panne est classé dans la catégorie des pannes catastrophiques, car c'est là où le processus commence à faire n'importe quoi et donne des résultats non conformes. On distingue deux types de pannes byzantines : les fautes byzantines naturelles comme dans le cas des erreurs physiques non détectées ou les erreurs logicielles, et les fautes byzantines malicieuses qui ont un intérêt de faire échouer le système comme le cas des virus.

La Figure 2.4 représente la hiérarchie des classes de pannes.

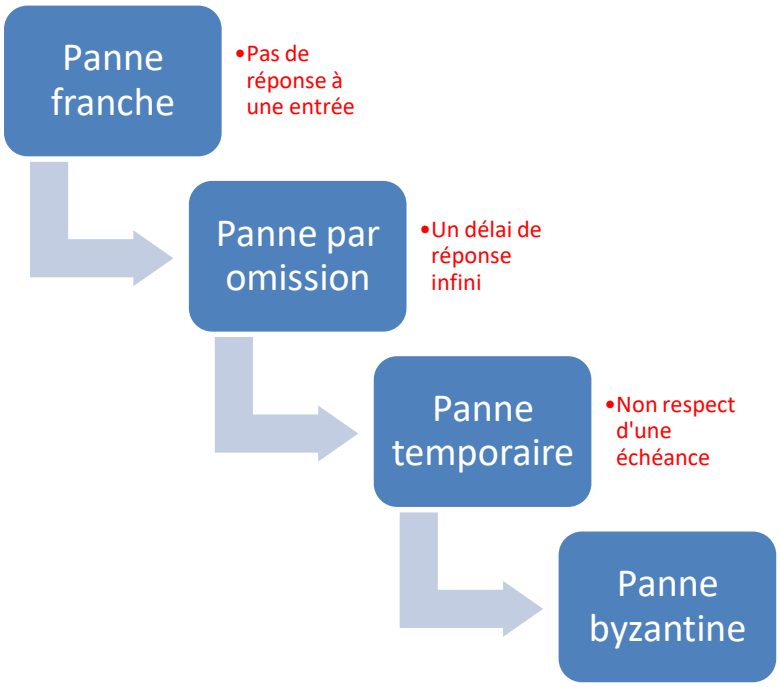

Figure 2.4 : Hiérarchie des classes de pannes.

2.5 Moyens de garantir la sureté de fonctionnement

Suite aux résultats catastrophiques et non conformes qui se produisent avec l'existence de pannes, un ensemble de solutions et méthodes ont été proposées [24]:

- **La construction :**
  a) **La prévention des fautes :** elle s'attache à des moyens pour éviter l'occurrence de faute dans le système. Autrement dit, comment l'empêcher, par construction l'occurrence ou l'introduction de fautes. Ce sont généralement les approches de vérification des modèles conceptuels.
  b) **La tolérance aux fautes :** elle permet d'assurer la continuité du fonctionnement malgré l'existence de la panne dans le système. Le degré de la tolérance varie selon la capacité du système à continuer à délivrer un service conforme en dépit de fautes.
- **La validation :**
  a) **L'élimination de fautes :** elle se focalise sur les techniques permettant de réduire la présence de fautes ou leurs impacts. Cela est réalisé par un ensemble de méthodes statiques de preuve de la validité du système comme : la simulation, les tests, etc.
  b) **La prévision des fautes :** elle estime, après une évaluation, la présence et l'apparition des pannes (temps, nombre, impact) et leurs conséquences. L'injection des fautes, afin de valider le système relativement à ces fautes, est l'une des méthodes utilisées pour réaliser la prévision.

2.6 Tolérance aux pannes

Le principe de la tolérance aux pannes est que le système fonctionne malgré la présence de la faute et il continu de présenter ses résultats, c'est-à-dire que le système essaye d'éviter les défaillances selon la chaine décrite sur la figure 2.3. En effet, la faute se transformera en une défaillance qui sera plus dangereuse pour le système. La tolérance aux fautes est la mise en œuvre des approches de détection de pannes ainsi que le rétablissement du système.

2.6.1 Procédure générale de la tolérance aux pannes

La procédure de tolérance aux pannes permet de sauver le système et le tolérer aux pannes ou aux fautes. Pour cela, une suite d'étapes est exécutée dans la plupart des systèmes réparties, comme indiquée sur la figure 2.5.

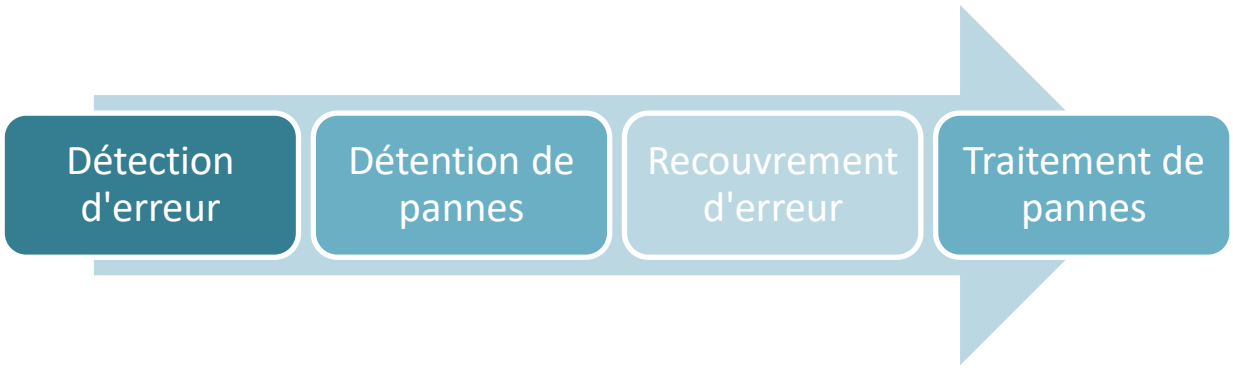

Figure 2.5 : Procédure générale de tolérance aux pannes [25].

2.6.1.1 Détection des pannes

La détection des pannes ou d'erreur est le fait de déceler la présence d'un événement inattendu dans le système. Dans ce cas, l'erreur est signalée à l'utilisateur du système par un mécanisme de détection qui permet de vérifier l'exactitude du résultat délivré [23].

Parmi les méthodes les plus utilisées pour la détection des pannes, on peut citer :

- **L'utilisation des programmes de tests :** dans cette méthode, il existe plusieurs formes comme la détection hors ligne (par des diagnostics), la détection en ligne (ou la détection continue), mais la forme la plus connue est celle du « watchdogs (les chiens de garde) ». Cette dernière a le principe de la surveillance mutuelle et elle permet de contrôler le temps d'exécution d'une tâche, de le comparer avec le « timeout » (une valeur maximale défini au préalable). Si le temps de la tâche dépasse le « timeout », une erreur est détectée.

- **La détection des erreurs de données :** pour vérifier cette étape, il suffit d'avoir des codes détecteurs d'erreurs et des tests. Elle consiste à coder les données en ajoutant de la redondance, et cela aide à détecter directement une erreur lors de sa réception (décodage). Plus le code est complexe plus la capacité de détection d'erreur est importante.

- **La détection des erreurs d'enchaînement :** cette technique se focalise sur la comparaison avec un référentiel ou des séquences valides, comme vérifier si les données sont conformes à des règles générales ou définies en fonction de l'application. Ce type de contrôle peut s'effectuer au niveau matériel (accès à une zone mémoire interdite par exemple) ou au niveau logiciel où on les intègre dans le système d'exploitation ou même au niveau de l'application.

- **Autres méthodes :** comme la programmation défensive qui consiste à ajouter des tests sur les entrées et les sorties et en cas d'erreur, un traitement d'exception sera déclenché. Ce dernier permet de protéger les parties non erronées d'être touchées par la défaillance existante.

2.6.1.2 Détention de la panne

Cette phase permet de viser et de cerner la panne au cœur du système puis de limiter ses effets sur une zone particulière, c'est-à-dire d'isoler cette zone pour ne pas affecter le reste du système.

En cas de détection d'intrusion, par exemple, l'isolation des composants compromis minimise le risque d'attaque des composants encore fonctionnels [26].

2.6.1.3 Recouvrement d'erreur

Le recouvrement d'erreur ou le rétablissement du système vise à changer l'état erroné en un état correct et sans erreur. Pour ce faire, un ensemble de techniques sont proposées [25] :

- La **reprise** est l'une des techniques les plus utilisées. La sauvegarde régulière de l'état du système est indispensable et donc lors d'une détection d'erreur, le système est ramené à un état survenu avant l'occurrence d'erreur. Cet état sauvegardé est appelé point de reprise.
- La **poursuite** consiste à trouver un état où le système peut fonctionner afin de sortir de l'état erroné. Cette phase peut être vérifiée grâce à un traitement d'exception lors de la détection d'erreur par exemple.
- La **compensation** nécessite que l'état du système comporte suffisamment de redondance pour permettre sa transformation en un état cohérent. Elle est différente par rapport aux méthodes précédentes, car elle peut être réalisée, par exemple, par l'utilisation des codes correcteurs d'erreurs.

2.6.1.4 Traitement des pannes

Dans cette phase, des diagnostiques sont faits sur la panne pour parvenir aux causes des erreurs puis établir une solution afin de réparer la panne comme, par exemple, l'isolation des parties erronées des parties seines dans le but d'empêcher la propagation de la panne dans tout le système. La procédure de réparation est liée au type de la panne.

2.6.2 Techniques de la tolérance aux pannes

Les techniques le plus connues pour tolérer les pannes dans les systèmes distribuées sont : la prévision des pannes, le masquage des pannes et le recouvrement après pannes.

2.6.2.1 Prévision des pannes

La prévision des fautes est une méthode intégrée dans la phase du développement afin d'empêcher l'introduction des pannes dans cette étape. La méthode la plus utilisée est la

migration. Son principe est que le système migre vers un autre composant (serveur ou service) plus robuste (avec plus de ressources disponibles). Ce composant prend le relai dans le cas d'un problème pour éviter la perte de données.

2.6.2.2 Masquage des pannes

Le masquage des pannes est une technique qui consiste à dupliquer les services et les serveurs sur différents dispositifs afin de réduire la probabilité de panne du système. Dans ce cas, grâce à la duplication, la panne sera masquée. Pour assurer cette technique, il faut bien gérer le temps et la cohérence entre les copies.

2.6.2.3 Recouvrement après pannes

Le principe du recouvrement, après l'occurrence des pannes, est la sauvegarde périodique sur un support de stockage stable. Dans le cas où le système s'arrête, il sera automatiquement redémarré à partir du point de sauvegarde.

2.7 Tolérance aux pannes dans le Cloud computing

Dans le Cloud computing, les techniques de la tolérance aux pannes sont basées sur deux types de détection ou politiques standards qui sont la détection de pannes proactive et la détection de pannes réactive [25].

2.7.1 Politique de la tolérance aux pannes proactive

Le système vise à prédire les pannes avant leurs apparitions afin de remplacer les composants affectés par des composants corrects en se basant sur des méthodes de prédiction des pannes.

Parmi les techniques basées sur cette politique, on peut citer :

a) **La migration des tâches :** le principe de cette politique se base sur le fait de faire migrer la tâche d'une machine vers une autre machine plus fiable.

b) **La régénération du logiciel (Software Rejuvenation) :** un ensemble de redémarrages périodiques sont programmés pour le système, et à chaque fois que ce système démarre il prend un nouvel état.

c) **L'auto-guérison (Self-Healing) :** le système gère automatiquement l'échec des instances d'applications qui sont en cours d'exécution sur plusieurs machines virtuelles.

2.7.2 Politique de la tolérance aux pannes réactive

Contrairement à la politique précédente, le système détecte une panne après son apparition, et il essaie de réduire les impacts de cette défaillance avec certaines techniques :

a) **Resoumissions des tâches :** si la machine tombe en panne, la tâche va être resoumise pour exécution soit sur la même machine en cas de dépannage ou sur une autre machine.

b) **Gestion des exceptions :** un ensemble de traitements particuliers doivent être exécutés dans le cas de détection de pannes où l'utilisateur les spécifie.

c) **Re-essai (Retry) :** est la méthode la plus simple. Son principe est de réessayer d'exécuter la tâche échouée sur la même ressource.

d) **Redondance ou réplication :** chaque tâche est répliquée sur une ou plusieurs machines d'une façon active ou passive dans le but d'assurer son exécution.

La partie suivante sera consacrée à la réplication dans les systèmes distribués.

2.8 Réplication de données

La réplication est utilisée comme une solution pour la tolérance aux pannes dans les systèmes distribués. Elle est passive, active ou hybride (semi-active).

2.8.1 Réplication passive

Dans un protocole de réplication passive, il existe deux états d'un composant répliqué. Une copie primaire pour effectuer tous les traitements, et une autre copie secondaire afin de surveiller la copie primaire et dès qu'elle tombe en panne, elle prend sa place. La copie primaire diffuse régulièrement son nouvel état aux copies secondaires afin de garantir la cohérence. Dans la réplication passive, un seul serveur (primaire) a le rôle de traiter les demandes des clients, et après le traitement, il diffuse le résultat pour mettre à jour l'état des autres serveurs (de sauvegarde) puis il renvoie la réponse au client.

La figure 2.6 résume le principe de la réplication passive.

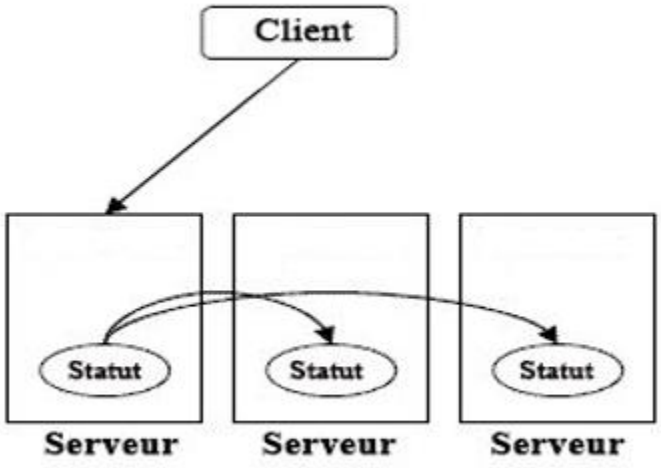

Figure 2.6 : Réplication passive [29].

2.8.2 Réplication active

Dans un protocole de réplication active, toutes les copies ont un rôle identique. Le principe de ce type de réplication est défini par trois points essentiels [28] :

- **La réception des requêtes** : toutes les copies reçoivent les mêmes requêtes dans le même ordre.

- **Le traitement des requêtes** : toutes les copies traitent les requêtes d'une manière déterministe, c.à.d. elles produisent le même résultat pour une requête donnée.
- **L'émission de réponse** : toutes les copies émettent la même séquence de réponses.

Dans le cas de la réplication active, chaque demande client est traitée par tous les serveurs, elle a été introduite pour la première fois par Leslie Lamport sous le nom de réplication de machine d'état. Cela nécessite que le processus hébergé par les serveurs soit déterministe.

Pour que tous les serveurs reçoivent la même séquence d'opérations, un protocole de diffusion atomique doit être utilisé. Ce protocole garantit que tous les serveurs reçoivent un message ou aucun, plus qu'ils reçoivent tous des messages dans le même ordre [29].

La figure 2.7 résume le principe de la réplication active.

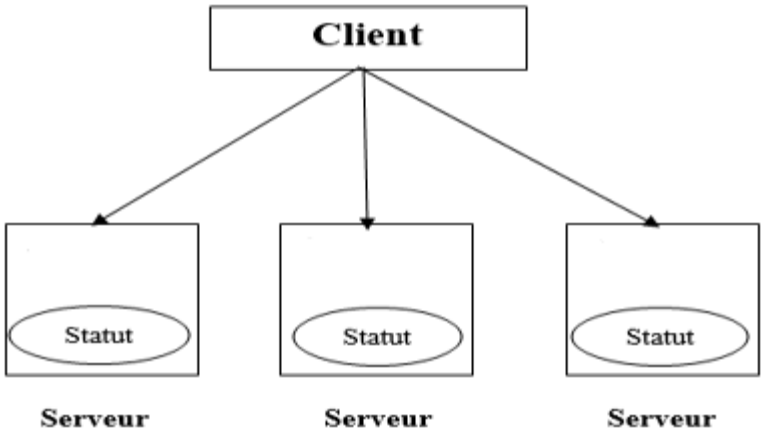

Figure 2.7 : Réplication active [29].

2.8.3 Réplication hybride (semi-active)

Comme son nom l'indique "hybride" veut dire un mélange entre les deux types de réplications citées précédemment, où toutes les copies traitent les requêtes du client mais une seule copie entre elles émet la réponse ; cette dernière s'appelle "leader". Le reste des copies (suiveurs) mettent à jours leurs états internes et suit leur leader d'une façon synchrone.

La figure 2.8 résume le principe de la réplication semi-active.

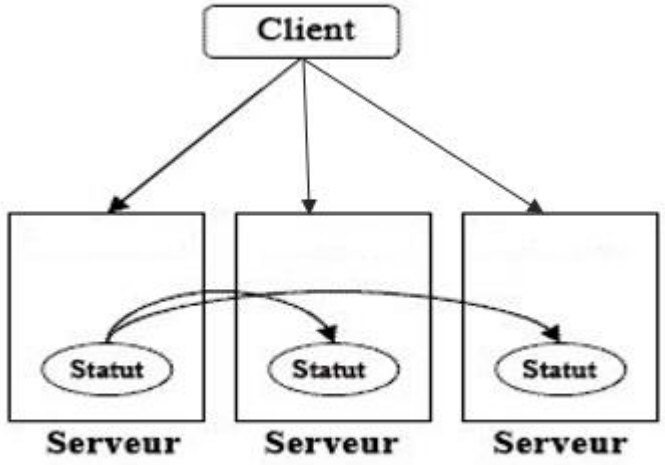

Figure 2.8 : Réplication semi-active.

Dans le cadre de ce PFE, et dans le domaine du Cloud computing, nous avons utilisé la réplication hybride pour la tolérance aux pannes. La tâche originale est affectée à une VM 1 et sa copie est affectée à l'une des voisines de VM1 (réplication active utilisant uniquement deux VMs pour minimiser la consommation de ressources). Pour ce qui concerne la réplication passive, on prend on considération la VM qui est tombée en panne (s'il y en a), on récupère la tâche perdue et on la soumit à une autre VM et ainsi de suite. Contrairement à la réplication active, la passive a besoin d'un mécanisme de détection de pannes mais pas trop gourmande en termes de ressources. L'objectif de la réplication hybride est de trouver un compromis entre la fiabilité et la consommation de ressources. La réplication active seule est fiable (si beaucoup de VMs exécutent en parallèle la tâche et ses copies) mais trop gourmande en termes de ressources.

2.9 Conclusion

Dans ce chapitre, nous avons présenté un aperçu sur la tolérance aux pannes dans les systèmes distribués. Nous avons discuté les types de pannes, les moyens d'assurer la sureté dans les systèmes distribués, les politiques de détection des pannes ainsi que d'autres notions très importantes dans ce domaine. La tolérance aux pannes n'est qu'une approche parmi d'autres pour assurer la sureté de fonctionnement mais elle est la plus demandée et la plus utilisée grâce à ses techniques efficaces.

Le prochain chapitre sera consacré à la présentation de notre contribution dans le cadre de ce PFE qui consiste à implémenter un algorithme de tolérance aux pannes appliqué au Cloud computing. Une application a été développée ainsi que des simulations utilisant CloudSim seront présentés avec une évaluation des résultats obtenus.

Chapitre 3 : Implémentation de l'application et évaluations des résultats obtenus

Chapitre 3 : Implémentation de l'application et évaluations des résultats obtenus

3.1 Introduction

L'objectif de ce chapitre consiste à présenter notre contribution dans le cadre de ce PFE. Il s'agit d'implémenter et d'évaluer un algorithme de tolérance aux pannes appliqué au Cloud computing. Nous allons donc commencer par présenter l'environnement de développement et de simulation, en particulier CloudSim, le simulateur Cloud que nous avons utilisé dans le cadre de ce travail. Nous présentons également l'approche utilisée pour la tolérance aux pannes qui est basée sur la réplication hybride ainsi que sur le principe du Maximal Independent Set (MIS).

Le chapitre expose les différentes IHMs réalisées dans le cadre de ce PFE ainsi que l'évaluation de l'algorithme de tolérance aux pannes à travers des graphes de comparaison.

3.2 Environnement de développement et de simulation

3.2.1 Java

Java représente à la fois un langage de programmation orienté objet et un environnement d'exécution portable, crée par James Gosling et Patrick Naughon employés de l'entreprise Sun en 1995. En 2009 la société a été rachetée par Oracle. Il s'agit d'un langage de programmation très performant qui a été adopté par la majorité des fournisseurs.

La sécurité est l'une des caractéristiques les plus importantes du langage car elle offre aux utilisateurs le sentiment de confiance ainsi que la sureté de leurs données. Java est dit portable car il peut s'exécuter sur différents systèmes d'exploitation tels que : Unix, Microsoft Windows, Linux…etc. Il a beaucoup de point commun avec le C++ mais avec quelques améliorations comme par exemple l'utilisation des interfaces pour l'héritage multiple ainsi que la gestion efficace de la mémoire à travers un Garbage Collector (GC) implémenté par la JVM.

De plus Java incorpore des fonctionnalités afin de facilité la simulation de quelques tâches de programmation, comme pour la connectivité avec les bases de données ainsi que le développement des applications multitâches, etc. [30].

3.2.2 NetBeans

NetBeans est un environnement de développement intégré (EDI), placé en open source et gratuitement par Sun Microsystems depuis juin 2000. C'est un outil très puissant pour la programmation, la compilation, et le déploiement des programmes.

NetBeans est écrit en Java par l'ajout de JDK (Java Développement Kit) mais il peut supporter n'importe quel langage de programmation (Python, C, C++…) [31].

NetBeans est disponible sur plusieurs systèmes d'exploitation comme Windows, Linux...etc., il est connu mondialement et il a une large base d'utilisateurs [32].

3.2.3 JFreeChart

JFreeChart est une API Java, elle permet de créer et dessiner des graphiques et des diagrammes en plusieurs formes (2D ou 3D) [33]. Cette API est open source mais sa documentation est payante.

3.2.4 CloudSim

CloudSim est un Framework qui modélise et simule les scénarios du Cloud computing et les services d'application associés, il a été écrit avec Java [34].

La figure 3.1 illustre l'architecture de CloudSim.

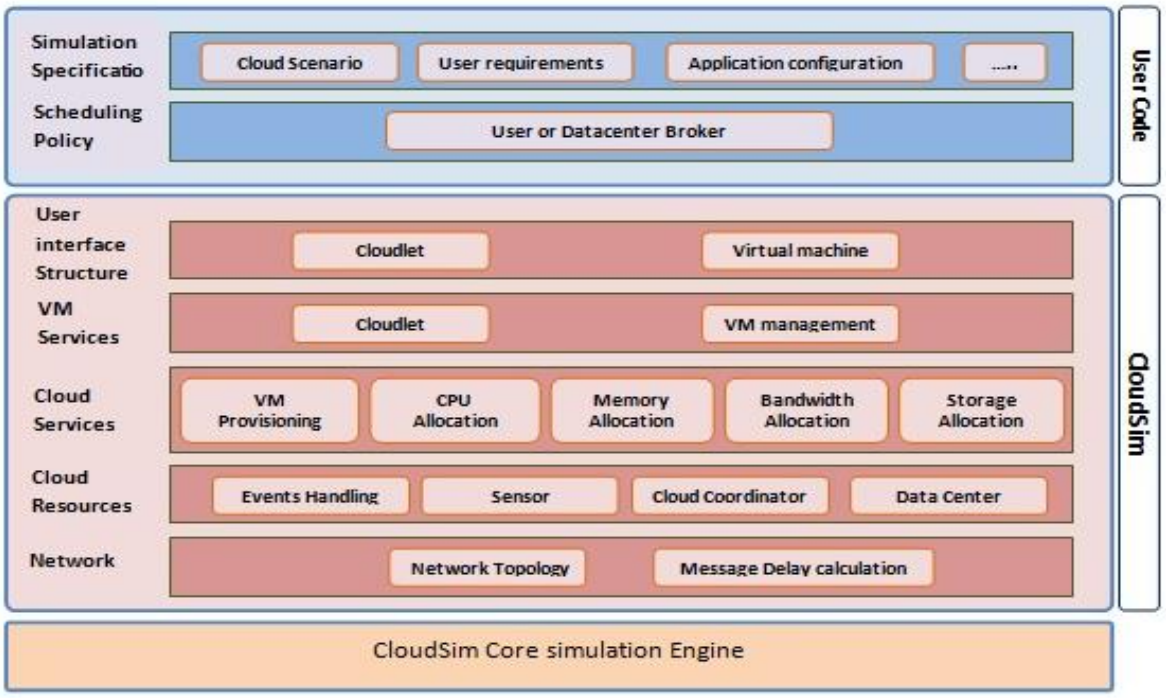

Figure 3.1 : Architecture de CloudSim [35].

Parmi les classes fondamentales qui forment les blocs constitutifs du simulateur CloudSim, on peut citer [36]:

- **Datacenter :** cette classe modélise les services au niveau des infrastructures de base (matériel). Elle contient un ensemble de machine physique qui prennent l'appellation de « Host », cette dernière peut être homogène ou hétérogène selon la configuration matérielle (mémoire, CPU, capacité et stockage). Chaque Data center implémente un ensemble d'algorithmes pour faire l'allocation de la bande passante, l'espace de stockage et la mémoire aux différents machines physiques (Hosts) et virtuelles (VMs) du Cloud.

- **Cloudlet :** cette classe modélise les services d'application du Cloud. Elle a un nombre de données et d'instructions connu à exécuter et à transférer. Cette classe peut être étendue afin de supporter la modélisation de la performance ainsi que d'autres paramètres de composition des applications comme les applications orientés base de données.

- **Datacenter Broker :** cette classe modélise le Broker, son rôle est de faire la médiation de négociation entre les utilisateurs et les prestataires du service selon les demandes des utilisateurs en termes de QoS (qualité de service). Le Broker négocie avec les prestataires du service approprier du Cloud afin d'allouer les ressources qui répondent aux besoins des utilisateurs.
- **Host :** cette classe représente une ressource matérielle ou physique comme un serveur de stockage ou de calcul. Chaque Host appartient à un Datacenter et il partage les caractéristiques proposées par ce dernier. La classe contient des informations très importante telles que la quantité de mémoire et de stockage, le type du cœur de traitement (mon ou multi-cœurs), un plan d'allocation de partage de puissance du traitement, ainsi que des politiques d'attribution de mémoire et de bande passante entre ses VMs [37].

- **Machine virtuelle (Vm) :** cette classe représente une instance de machine virtuelle qui est liée à une machine physique (Host). Toutes les Vms qui appartiennent au même Host partagent un ensemble de caractéristiques telles que la mémoire et l'espace de stockage…etc.

Il existe d'autre classes qui sont responsables de la configuration des caractéristiques de quelques composantes comme : la classe ***DatacenterCharacteristics***. Cette dernière est spécifiée pour le Datacenter et elle contient des informations comme : le système d'exploitation utilisé, la liste des Hosts, le cout par seconde d'utilisation des ressources…etc.

A l'aide de ces composants, il est facile de développer de nouvelles stratégies pour l'utilisation de Cloud, tout en tenant compte des politiques, des algorithmes de planification, des plans d'équilibrage de charge, etc.

CloudSim est suffisamment flexible pour être utilisé comme une bibliothèque et de pouvoir intégrer des scénarios en utilisant Java [33].

La figure 3.2 présente l'architecture simplifiée de CloudSim (Cloud Information Services (CIS) contient les informations sur les ressources).

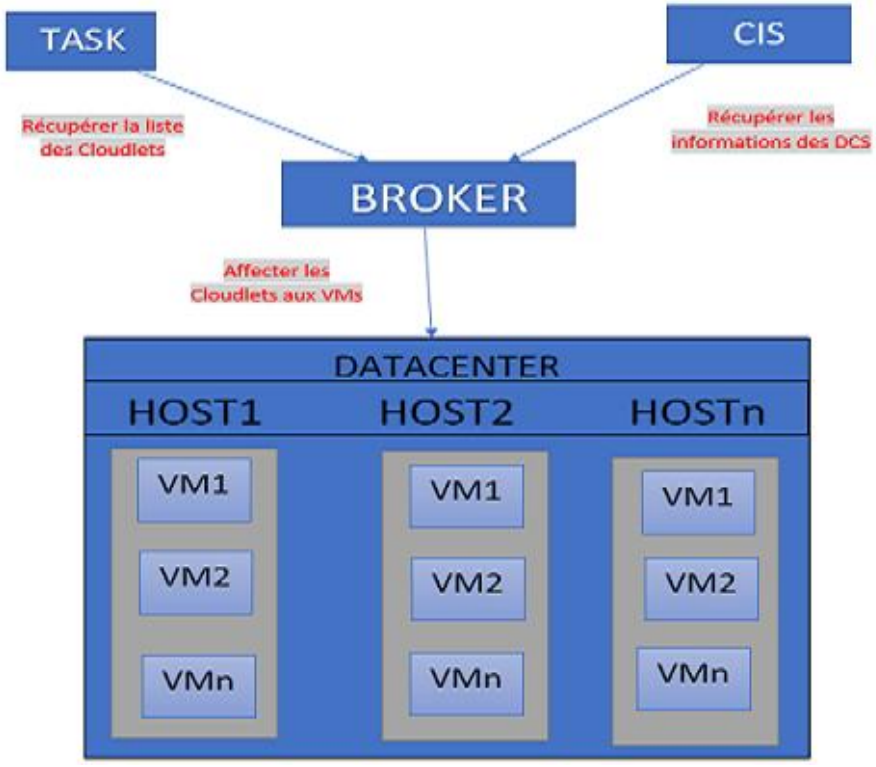

Figure 3.2 : Architecture simplifié de CloudSim.

3.3 Approche proposée pour la tolérance aux pannes

Dans le cadre de ce PFE et pour tolérer les pannes dans un environnement Cloud, nous avons opté pour la réplication hybride. Une tâche sera répliquée d'une première VM à une deuxième VM (une tâche originale et une copie et donc réplication active). En cas de panne de la première VM, la tâche sera répliquée à nouveau sur une autre VM et ainsi de suite (réplication passive).

Pour l'algorithme de tolérance aux pannes, nous avons choisi de répliquer sur la VM la moins chargée mais en intégrant un moyen formel qui est l'ensemble maximal indépendant ou le Maximal Independent Set (MIS) qui sera détaillé dans ce qui suit.

3.3.1 Maximal Independent Set

Soit G = (VM, E) un graphe représentant les liens entre les VMs, où VM est l'ensemble des machines virtuelles et E est les liens entre les VMs. L'arc **$(VM_i, VM_j) \in E$** est le lien entre la VM_i et la VM_j.

Soit le sous-ensemble S de VM dans G = (VM, E), on dit que S est un **Indépendant Set (IS)** ou ensemble indépendant si : **$\forall VM_i, VM_j \in S, (VM_i, VM_j) \notin E$.** IS est un ensemble de sommets deux à deux non adjacents.

On dit que S est un **Maximal Independent Set (MIS)** ou ensemble maximal indépendant si : **$\forall$ VM $\notin$ S, elle a une voisine dans S.** La définition du MIS implique que pour tout graphe G, si

un nœud n'est pas dans le MIS, alors il doit être adjacent à au moins un nœud du MIS. Un MIS est un IS qui ne contient aucun autre IS avec une cardinalité plus grande.

La figure 3.3 illustre un graphe qui peut avoir trois cas différents de Maximal Independent Set.

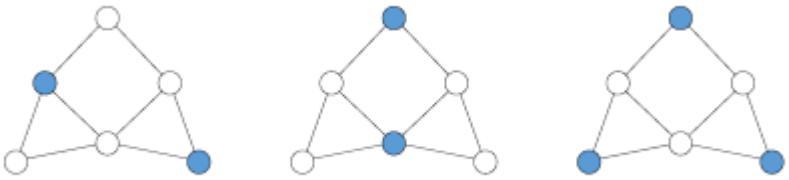

Figure 3.3 : Graphe de MIS.

Les cercles en bleu représentent le Maximal Independent Set. Les cercles en blanc ont tous des voisins dans le MIS.

Les nœuds du MIS sont appelés « dominants », les autres nœuds sont appelés « dominés ».

Avant de présenter le pseudo code de l'algorithme MIS qui a été implémenté dans le cadre de ce PFE, nous donnons les variables/définitions suivantes :

- $\mathbf{S_i}$: état d'une VM (wait, dominé ou dominant). Initialement $\mathbf{S_i}$ = wait.
- Le Maximal Independent Set est défini comme suit : S = {$\mathbf{VM_i} \in \mathbf{VM}$: $\mathbf{S_i}$ = dominant}.
- $\mathbf{N_i}$: les voisins directs de la VM_i.
- $\mathbf{VM_i}$ est dominée si : ($\mathbf{S_i}$ = dominé ⋁ wait) ⋀ ($\exists\ \mathbf{VM_j} \in \mathbf{N_i}$: $\mathbf{S_j}$ = dominant). $\mathbf{VM_i}$ n'a pas l'état dominant et elle a au moins une VM voisine dominante.
- $\mathbf{VM_i}$ est dominante si : ($\mathbf{S_i}$ = dominant) ⋀ ($\forall\ \mathbf{VM_j} \in \mathbf{N_i}$: **Sj** = dominé ⋁ wait). $\mathbf{VM_i}$ a l'état dominant et tous ces voisins directs ont l'état dominé ou wait.
- S'il existe deux voisins qui ont le même état dominant, la VM qui a le plus petit Id garde l'état dominant et l'autre aura l'état dominé (chaque VM est identifié par un Id, dans un MIS, on n'a pas deux voisins dominants. Le critère « plus petit » est utilisé pour les départager).
- $\mathbf{W_i}$: l'ensemble des voisins de $\mathbf{VM_i}$ qui ont l'état wait et qui ont un Id inférieur à celui de VM_i : ($\mathbf{VM_j} \in \mathbf{N_i}$ **:** $\mathbf{S_j}$ = wait ⋀ i > j).
- $\mathbf{A_i}$: l'ensemble des voisins dominants de $\mathbf{VM_i}$. : ($\mathbf{VM_j} \in \mathbf{N_i}$ **: Sj** = dominant).
- $\mathbf{A_i}^*$: l'ensemble des voisins dominants de $\mathbf{VM_i}$ qui ont un Id inférieur à celui de VM_i : ($\mathbf{VM_j} \in \mathbf{N_i}$ **: Sj** = dominant ⋀ i > j).

  L'algorithme MIS démarre sur une VM quelconque. Ensuite, un appel récursif pour tous les nœuds est effectué jusqu'à ce que tous les états des nœuds deviennent « dominants » ou « dominés » à la fin de l'algorithme. Ce type d'algorithme est appelé auto-stabilisant.

Le pseudo code de l'algorithme MIS est comme suit [38] :

Algorithme 1 : MIS

Begin
1: **LastState$_i$** = **S$_i$**
2: if (**S$_i$** = wait ∧ **A$_i$**) ∨ (**S$_i$** = dominant ∧ **A$_i$***) then
3: **S$_i$** = dominé
4: else
5: if (**S$_i$** = wait ∧ non **A$_i$** ∧ non **W$_i$***) then
6: **S$_i$** = dominant
7: end if
8: end if
9: if **LastState$_i$** ≠ **S$_i$** then
10: for $VM_j \in N_i$ do
11: VM_j.MIS
12: end for
13: end if
End

3.3.2 Algorithme de tolérance aux pannes utilisant le MIS

Dans ce qui suit, nous présentons le pseudo code de l'algorithme Fault tolerance (FT) utilisant le MIS.

À la réception d'une nouvelle tâche **T$_k$** par **VM$_i$**, cette tâche sera répliquée sur une autre machine **VM$_j$**. **VM$_j$** est identifiée à travers l'algorithme MIS, le nœud dominant réplique sur un nœud dominé moins chargé et le nœud dominé réplique sur un nœud dominant moins chargé. En cas de détection de pannes, l'algorithme MIS est lancé à nouveau afin de mettre à jour la liste de voisinage. La tâche qui a été considérée comme réplica devient originale et le traitement est lancé à partir de la ligne 2 de l'algorithme.

Le pseudo code de l'algorithme de tolérance aux pannes est comme suit :

Algorithme 2: FT

1: **while** true **do**
2: **if** (Recevoir une nouvelle tâche T_k) **then**
3: **if** (S_i = dominant) **then**
4: Sélectionner la VM dominée la moins chargée VM_j
5: **else**
6: Sélectionner la VM dominante la moins chargée VM_j
7: **end if**
8: Répliquer T_k sur VM_j
9: **end if**
10: **if** (Recevoir une tâche répliquée T_k de VM_j) **then**
11: $RpVMTask_i = RpVMTask_i \cup \{(VM_j, T_k)\}$ // Ajouter T_k à la liste des tâches répliquées de VM_i
12: **end if**
13: **if** (le Host de VM_f est en panne) **then**
14: **if** $((VM_f, T_f) \in RpVMTask_i)$ **then**
15: $Q_i = Q_i \cup \{T_f\}$ // Ajouter T_f à la liste des tâches originales de VM_i
16: $RpVMTask_i = RpVMTask_i - \{(VM_f, T_f)\}$ // Retirer (VM_f, T_f) de la liste des tâches répliquées de VM_i
17: this.MIS // Appel de MIS avec la nouvelle liste des VMs
18: Répéter à partir de 2 pour T_f // Refaire les mêmes étapes que la tâche T_k
19: **end if**
20: **end if**
21: **end while**

3.4 Présentation de l'application

Afin de bien présenter notre algorithme et vu que CloudSim n'est pas fourni avec une interface graphique, un ensemble d'IHMs ont été développé dans le cadre de ce PFE.

3.4.1 Interface principale

Au moment du lancement de l'application, une IHM s'affiche (figure 3.4), cette dernière contient trois boutons :

- **Accueil :** pour afficher la page principale.
- **Mon Cloud :** pour passer à l'interface suivante afin de configurer le Cloud.
- **Quitter :** pour fermer l'application.

Figure 3.4 : Interface principale.

Pour configurer le Cloud, deux interfaces ont été développées.

3.4.2 Interface de configuration de Cloud

Pour configurer le Cloud, il suffit d'entrer un ensemble d'informations comme : le nombre de Data Centers, le nombre de Hosts…etc. dans les champs présentés dans la figure 3.5. Le bouton « Valider » a la fonctionnalité d'enregistrer les données et de passer à la prochaine interface afin de continuer la configuration du Cloud.

Figure 3.5 : Interface de configuration.

3.4.3 Interface de configuration des Data Centers et des Hosts

Afin de compléter la partie précédente de la configuration de Cloud, cette interface est divisée en deux parties :

- L'interface représentée par la figure 3.6 permet de configurer les Data Centers. Le bouton « Par défaut » permet de garder le nom qui a été donné au DC lors de sa création. Le bouton « Créer » permet d'enregistrer le DC (son nom et son numéro). Le bouton « Annuler » permet de supprimer les données saisies avant la validation.

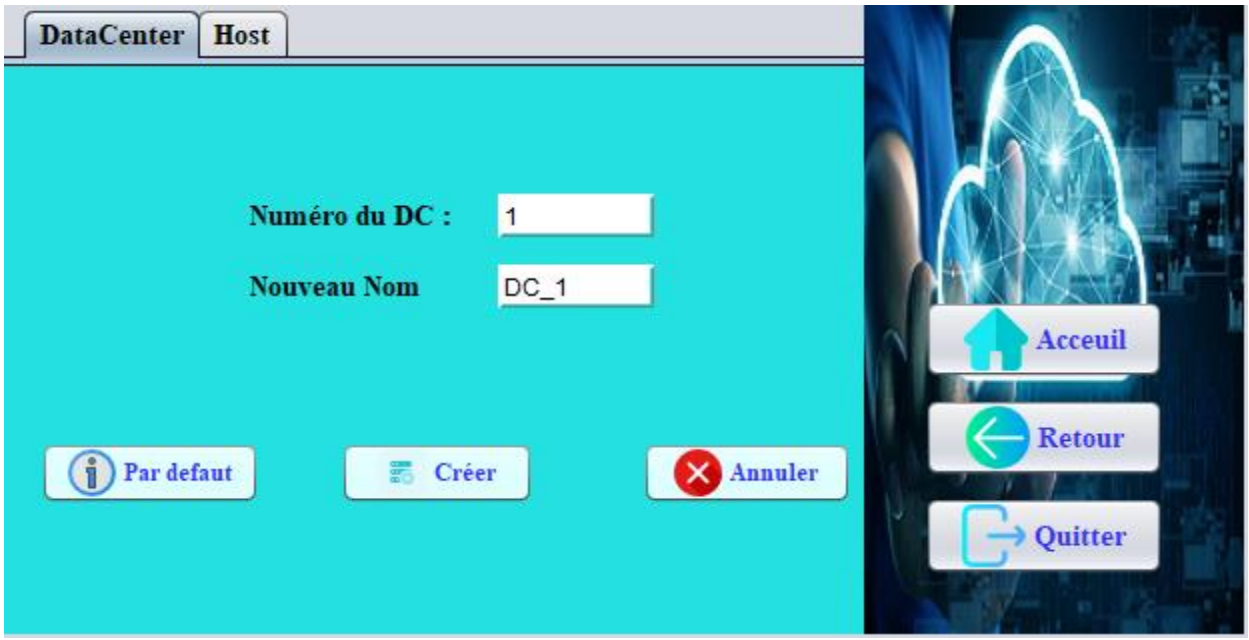

Figure 3.6 : Interface de configuration des Data Centers.

- L'interface représentée par la figure 3.7 permet de configurer les machines physiques (Host), en particulier les capacités du stockage de la machine en saisissant le numéro du Host ainsi que le numéro de son Data Center. Il est aussi possible à travers cette IHM de choisir le type d'architecture utilisée, homogène ou hétérogène.

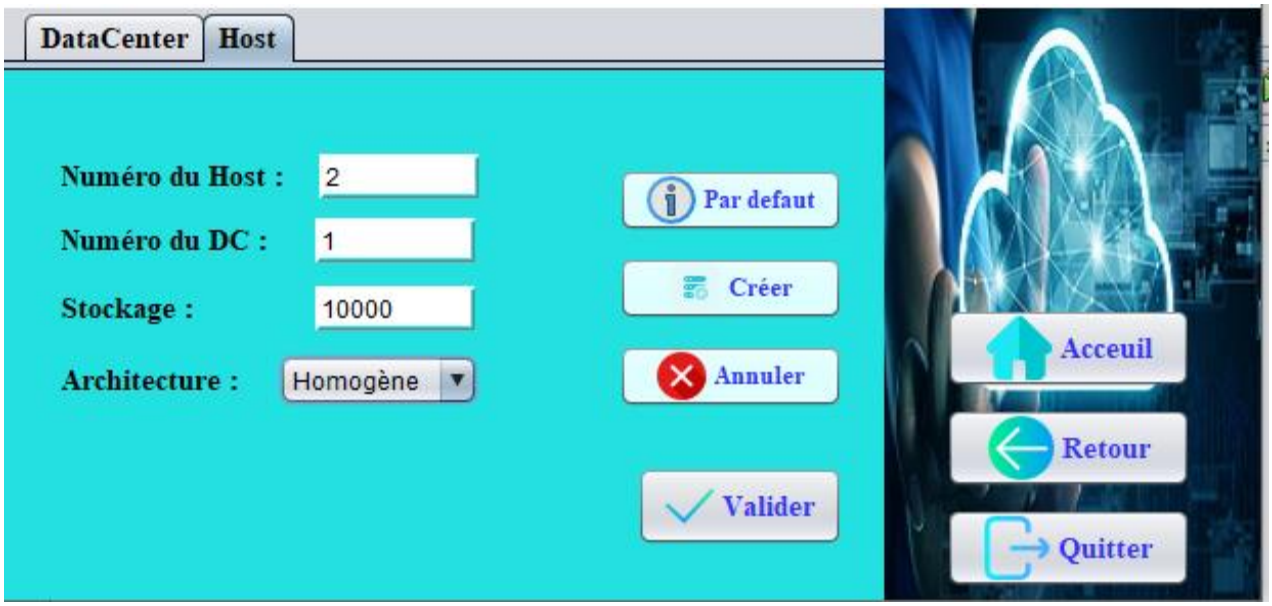

Figure 3.7 : Interface de Configuration des Hosts.

3.4.4 Interface pré-simulation

Au niveau de cette interface (figure 3.8), la phase de configuration est déjà validée, par conséquent, nous pouvons afficher la topologie du système (les DCs avec leurs Hosts et leurs capacités de stockage). L'IHM offre la possibilité de lancer un traitement avec ou sans tolérance aux pannes. La simulation sera lancée en cliquant sur le bouton « Simulation ».

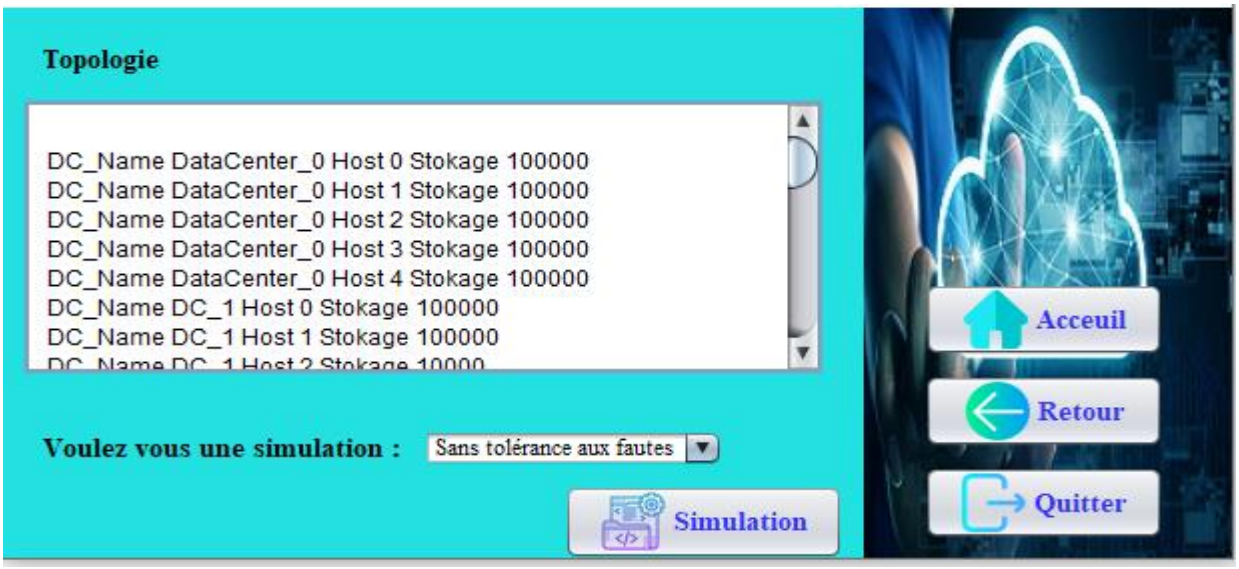

Figure 3.8 : Interface pré-simulation.

3.4.5 Interface post-simulation

L'interface représentée par la figure 3.9 montre les résultats obtenus à savoir : le nombre de tâches exécutées, le temps de réponse et le coût total. Le bouton « Statistiques » visualise les résultats obtenus sous forme de graphes.

Figure 3.9 : Interface post-simulation.

3.4.6 Interface de choix de graphe

L'IHM illustrée dans la figure 3.10 permet de générer un certain nombre de graphes afin d'évaluer les performances de notre algorithme.

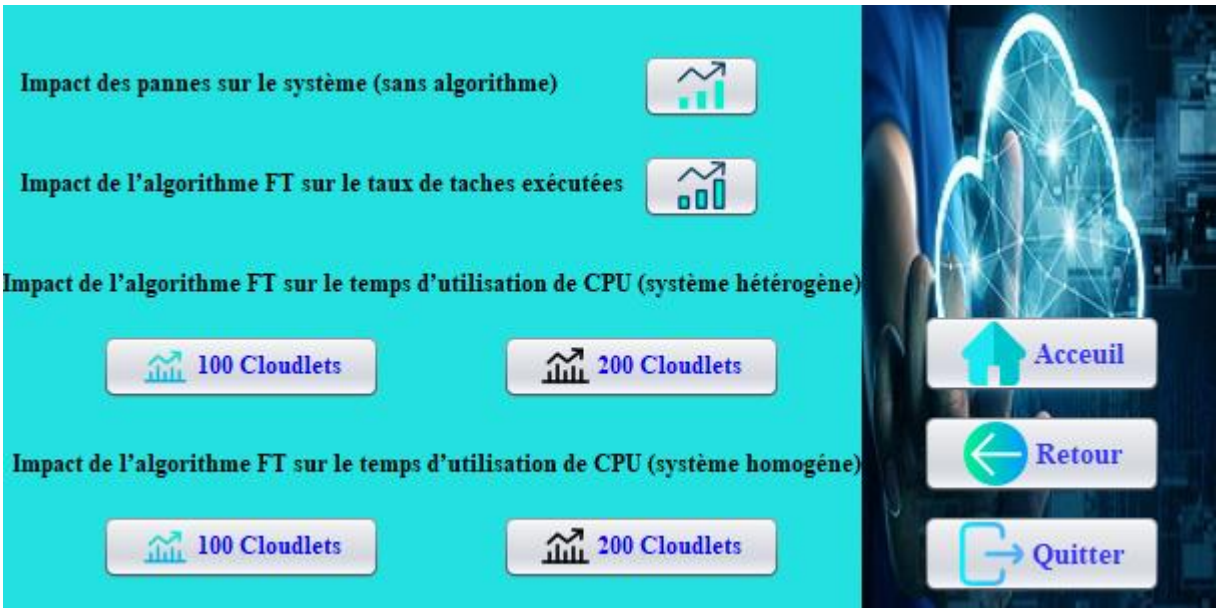

Figure 3.10 : Interface de choix de graphe.

3.4.7 Interface de résultats

Selon le choix confirmé au niveau de l'interface précédente, l'IHM suivante (figure 3.11) permet d'afficher le graphe correspondant. A titre d'exemple :

- L'IHM suivante montre l'impact des pannes sur le système.

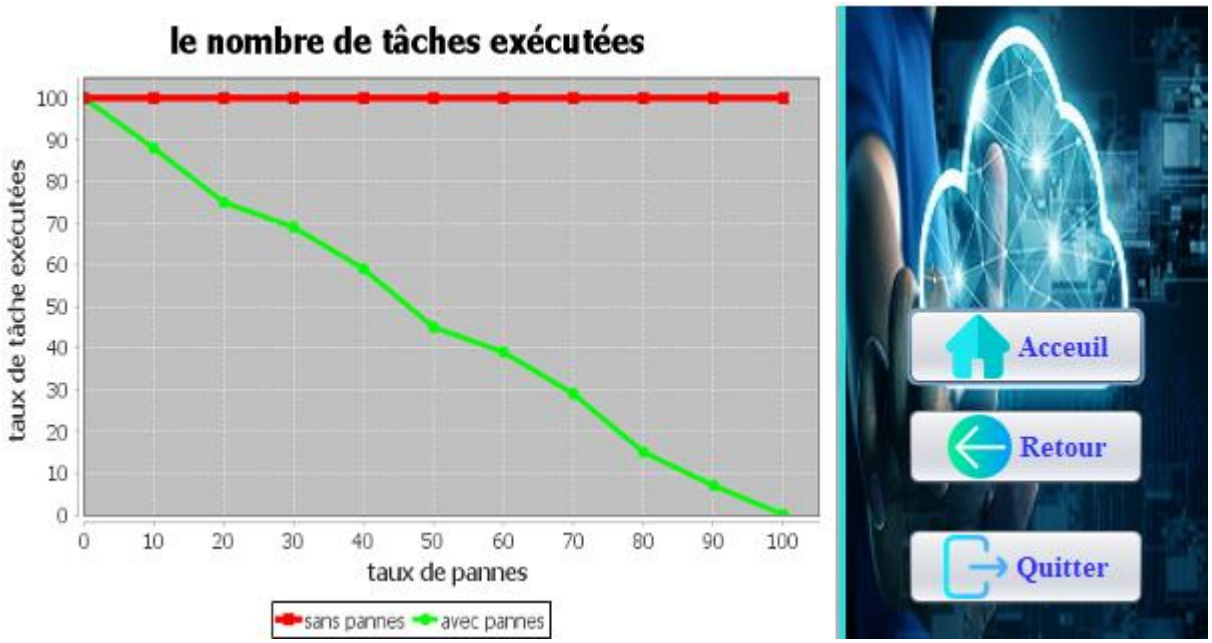

Figure 3.11 : Interface du premier résultat.

- L'IHM suivante montre l'impact de l'algorithme sur le taux de tâches exécutées.

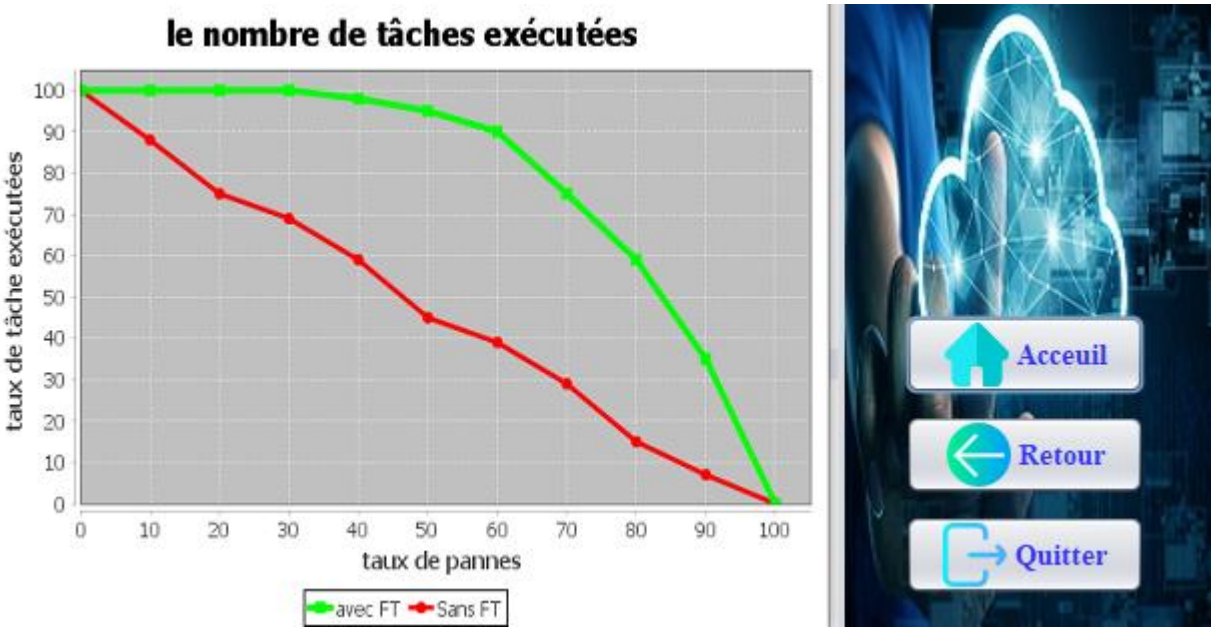

Figure 3.12 : Interface du deuxième choix.

3.5 Résultats obtenus

L'objectif de cette section est de montrer l'efficacité de l'algorithme implémenté à travers des graphes de comparaisons. A noter qu'une machine avec un processeur Intel(R) Core (TM) i3-6006U CPU et une vitesse de 2.00 GHz et capacité mémoire de 4GB, sous le système d'exécution Windows 10 de 64 bits a été utilisée pour la réalisation de l'ensemble des simulations.

Afin de configurer les pannes et les affectées au programme, nous avons utilisé un fichier de configuration « failurs.txt ». Un exemple de contenu de ce fichier est illustré dans la figure 3.13.

```
// DCID HOSTID StartTime EndTime
2 1 10 40
2 2 20 30
3 2 10 40
3 3 10 40
3 1 10 40
5 1 10 40
6 2 10 40
6 3 10 40
6 4 10 40
7 1 20 50
8 2 10 30
8 1 10 40
9 2 10 40
9 3 10 40
10 4 10 40
11 1 10 40
12 2 10 40
13 3 10 40
15 4 10 40
15 2 10 30
```

Figure 3.13 : Fichier failurs.txt.

- **DCID** : l'ID du Data Center.
- **HOSTID** : l'ID du host.
- **StartTime** : le temps de départ de la panne en seconde.
- **EndTime** : le temps de la fin de la panne en seconde.

Dans la première ligne (2 1 10 40) : le host numéro 1 qui se trouve dans le Data Center numéro 2 tombe en panne entre la dixième seconde et la quarantième seconde.

Si on considère que le nombre de hosts est égal à 100, un taux de panne égal à 20% veut dire qu'on doit insérer 20 lignes dans le fichier « failurs.txt » comme indiquer dans la figure 3.13. Le fichier de configuration sera utilisé par le simulateur afin d'insérer les pannes dans le système.

Dans ce qui suit, nous allons considérer un Cloud contenant : 16 DCs, 100 Hosts et 160 VMs. L'évaluation sera effectuée avec 100 tâches.

3.5.1 Impact des pannes sur le système (sans algorithme)

La figure 3.14 représente l'impact des pannes sur le nombre de tâches exécutées dans le cas où nous n'avons pas encore déployé l'algorithme de tolérance aux pannes.

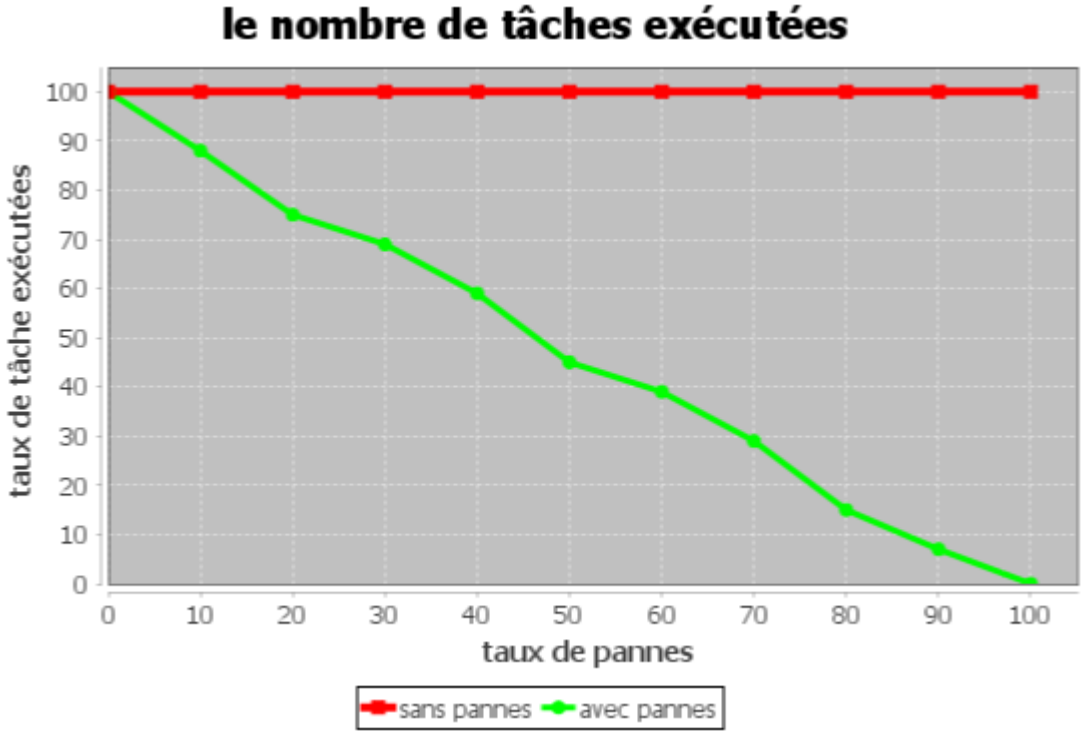

Figure 3.14 : Impact des pannes sur le système (sans algorithme).

Le résultat est logique, le taux de pannes a un impact significatif sur le taux de tâches exécutées. A titre d'exemple, avec un taux de pannes qui avoisine les 60%, nous avons moins de 40 % de tâches qui sont exécutées.

3.5.2 Impact de l'algorithme FT sur le taux de tâches exécutées

La figure 3.15 montre l'impact de notre algorithme de tolérance aux pannes sur le taux de tâches exécutées.

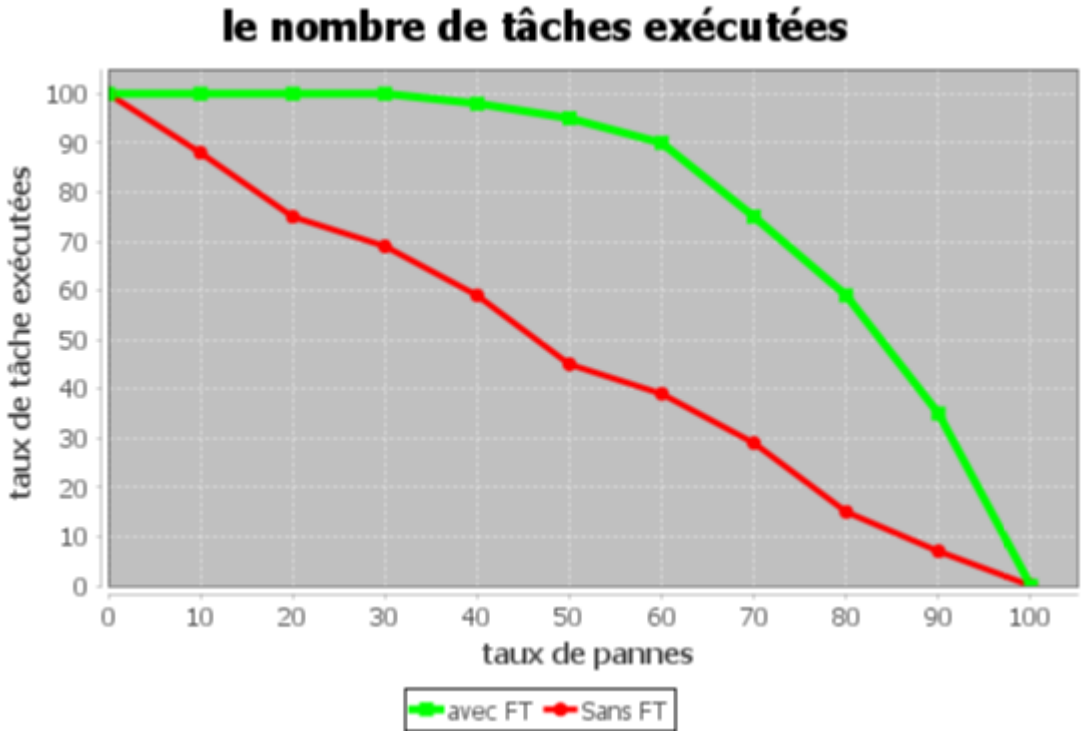

Figure 3.15 : Impact de l'algorithme sur le taux de tâches exécutées.

Le taux de tâches exécutées se dégrade rapidement sans utilisation de l'algorithme. Par contre, l'algorithme FT a des bonnes performances jusqu'à un taux de panne égal à 70%. Nous remarquons aussi que le taux de tâches exécutées s'est stabilisé à 100% jusqu'à 30% de taux de pannes et donc l'algorithme FT a été très performant dans ce cas. Si le taux de pannes est très élevé (plus de 80%), l'algorithme échoue à trouver des VMs pour exécuter les tâches et donc plusieurs tâches ne seront pas récupérées. Mais même dans ce cas, l'algorithme FT a atteint un taux d'exécution de tâches qui dépasse les 40% pour un taux de pannes qui est égal à 80 % contrairement à un peu moins de 15% de taux d'exécution de tâches pour le même taux de pannes sans algorithme.

Nous pouvons conclure donc que l'algorithme FT améliore significativement le taux de tâches exécutées. Dans ce qui suit, nous allons voir son impact sur l'utilisation de ressources, en particulier pour l'utilisation de CPU.

3.5.3 Impact de l'algorithme FT sur le temps d'utilisation de CPU

3.5.3.1 Système homogène

Dans le cas d'un système homogène, nous avons considéré les caractéristiques suivantes pour toutes les Vms : MIPS = 1000 instructions/sec, Nbr_CPU = 1, RAM = 2048 MB, BW = 500 MB/seconde.

Deux cas ont été étudiés :

- **Avec 100 tâches :** la figure 3.16 représente le temps d'utilisation de CPU selon différents taux de pannes dans un système homogène.

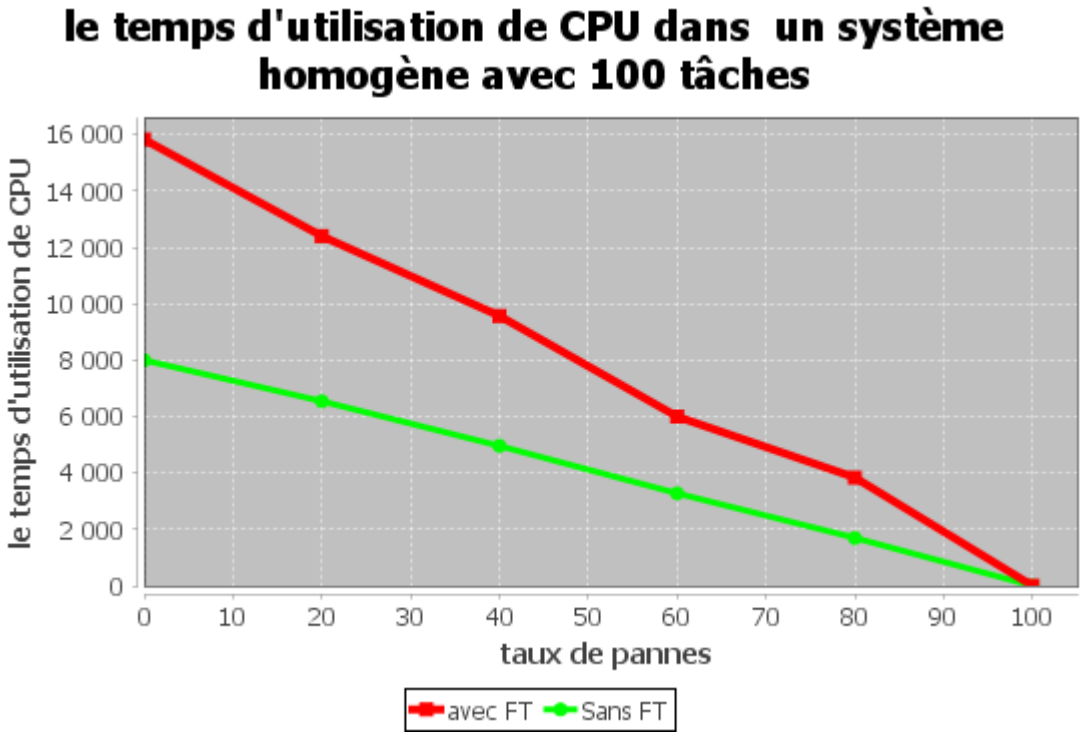

Figure 3.16 : Impact de l'algorithme sur le temps de CPU (système homogène avec 100 tâches).

- **Avec 200 tâches :** la figure 3.17 représente le temps d'utilisation de CPU selon différents taux de pannes dans un système homogène.

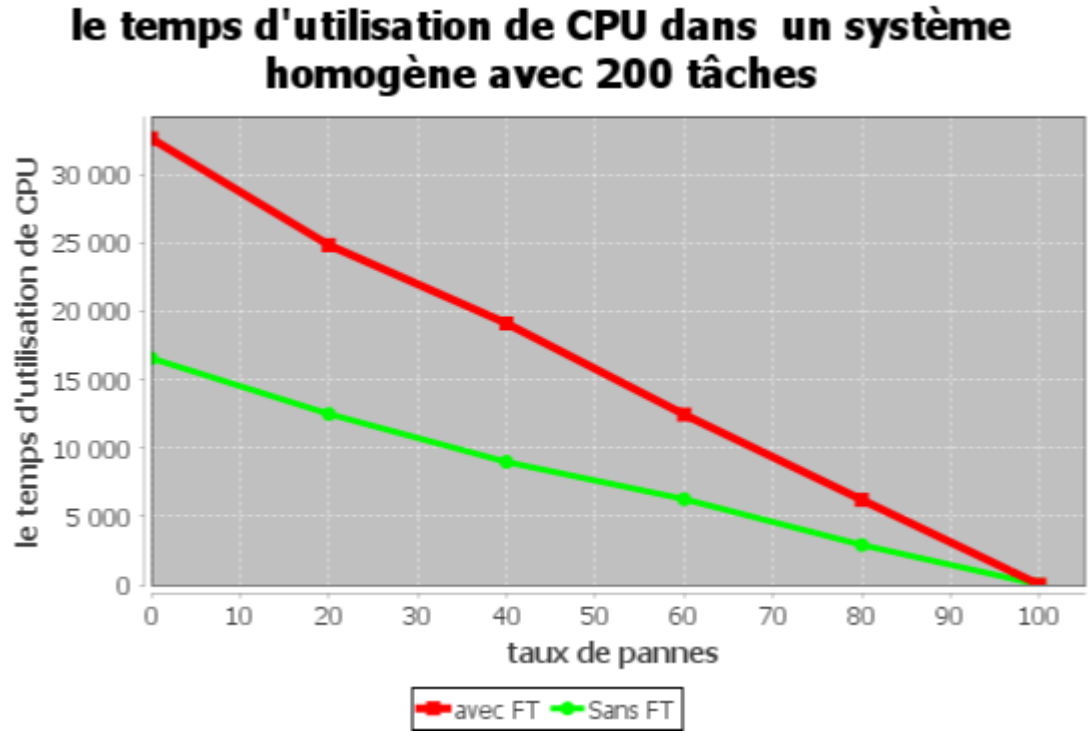

Figure 3.17 : Impact de l'algorithme sur le temps de CPU (système homogène avec 200 tâches).

L'algorithme FT utilise plus de ressources en raison des différentes réplications effectuées dans le système. Avec l'augmentation du taux de pannes, quelques tâches ne seront pas récupérables et par conséquent l'utilisation de ressources sera moindre.

3.5.3.2 Système hétérogène

Dans le cas d'un système hétérogène, nous avons gardé les mêmes valeurs pour CPU, RAM et BW de la configuration homogène et générer le MIPS d'une manière aléatoire entre 500 et 3000 instructions/seconde.

Pareil que pour le cas homogène, nous avons considéré deux cas :

- **Avec 100 tâches :** la figure 3.18 représente le temps d'utilisation de CPU selon différents taux de pannes dans un système hétérogène.

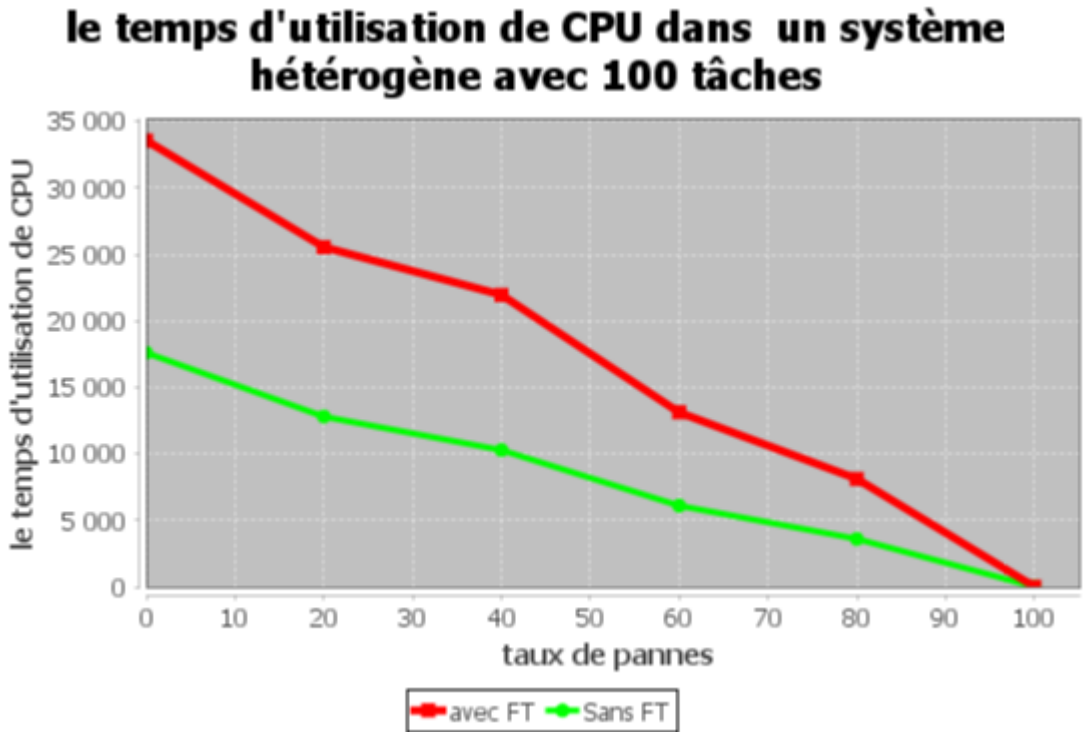

Figure 3.18 : Impact de l'algorithme sur le temps de CPU (système hétérogène avec 100 tâches).

- **Avec 200 tâches :** la figure 3.19 représente le temps d'utilisation de CPU selon différents taux de pannes dans un système hétérogène.

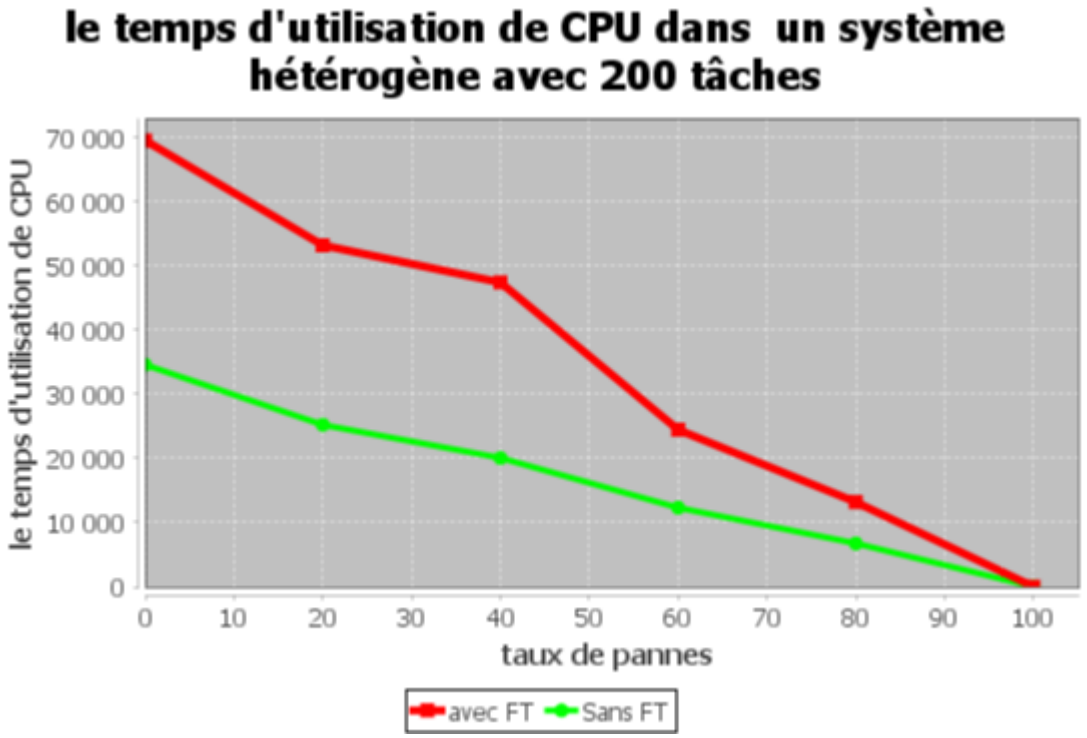

Figure 3.19 : Impact de l'algorithme sur le temps de CPU (système hétérogène avec 200 tâches).

Pareil que le cas homogène, l'algorithme FT utilise plus de ressources en raison des différentes réplications effectuées dans le système.

3.6 Conclusion

Dans ce chapitre, nous avons implémenté et évalué une approche de tolérance aux pannes appliquée au Cloud computing. Cette approche est basée sur la réplication hybride ainsi que sur le principe du Maximal Independent Set pour le choix des VMs pour effectuer la réplication.

L'approche adoptée dans ce PFE améliore le taux d'exécution des tâches en consommant bien évidement plus de ressources. Les résultats obtenus de ce travail ont été discutés dans ce chapitre.

Conclusion générale et perspectives

L'utilisation du Cloud computing a vécu une augmentation majeure durant ces dernières années, grâce aux multiples services offerts aux utilisateurs selon leurs besoins et leurs contrats. Aujourd'hui, nous pouvons affirmer que le Cloud computing a réussi à résoudre le problème de stockage des données, mais il est toujours face à des risques sous formes d'attaques provoquant des pannes ou des défaillances. Dans ce contexte la sureté de fonctionnement est un point essentiel. Un mécanisme de tolérance aux fautes est obligatoire. En effet, une panne peut endommager un nombre important de composants du système et par la suite être à l'origine de la perte de ses données ou de ses traitements.

Au cours de ce présent mémoire, nous avons simulé un Cloud contenant un ensemble de centres de données, de machines physiques et de machines virtuelles. Nous avons injecté des pannes dans ce système et puis nous avons simulé et évalué une approche afin de tolérer ces pannes. L'approche proposée est basée sur la réplication hybride ainsi que sur l'utilisation du MIS en répliquant toujours sur la machine la moins chargée (dominante ou dominée).

Pour évaluer notre approche, nous avons varié le taux de pannes des hosts et mesurer le taux de tâches exécutées. Nous avons aussi évalué l'impact de notre approche sur la consommation de ressources. Les résultats obtenus sont satisfaisants car notre approche a permis un taux d'exécutions de tâches élevé même en présence de beaucoup de pannes dans le système.

Comme perspective à ce travail, il serait intéressant de choisir le bon nœud de réplication en utilisant une technique par niveau afin de répliquer sur des nœuds sur un autre host ou un autre Datacenter. Un facteur de fiabilité qui dépend d'un indice de confiance adaptative serait intéressant aussi à considérer afin d'améliorer les performances de l'approches proposées.

Bibliographie

[1] B. Meroufel. « Coopération dynamique des agents pour la tolérance aux pannes dans les systèmes à large échelle ». Thèse de Doctorat en informatique de l'université d'Oran 1. Mai 2016.

[2] Définition | Supercalculateurs-Superordinateur | Futura Tech. https://www.futura-sciences.com/tech/definitions/informatique-supercalculateur-15261/. Consulté le 30 Mars 2020.

[3] Y. Meslem, SI. Debbas. « Ordonnancement et réplication de données dans le Cloud Computing ». Mémoire de Master en informatique de l'université de Saida. 2016.

[4] Modèles et types de Cloud computing. https://aws.amazon.com/fr/types-of-cloud-computing/. Consulté le 03 mars 2020.

[5] Les différents modèles du Cloud computing. https://www.funinformatique.com/les-differents-modeles-du-cloud-computing/. Consulté le 03 mars 2020.

[6] Cloud : les modèles de déploiement. https://blog.3li.com/cloud-les-modeles-de-deploiement/.Consulté le 03 mars 2020.

[7] Les caractéristiques essentielles du Cloud. https://blog.3li.com/les-caracteristiques-essentielles-du-cloud/.Consulté le 04 mars 2020.

[8] Cloud Computing : définition simple, caractéristiques essentielles. https://hebergeurs.top/cloud-computing-definition.Consulté le 04 mars 2020.

[9] Sureté de fonctionnement. http://www.suretedefonctionnement.fr/. Consulté le 05 mars 2020.

[10] Rachid, B., Hafaifa, A., Mouloud, G. Evaluation et exploitation de modèle de fiabilité et de maintenabilité d'un système industriel à base d'un système expert. In : Proc. International Symposium on Operational Research and Applications ISORAP2013. 2013.

[11] Fiabilité des systèmes. https://www.doyoubuzz.com/var/f/qk/ro/qkroqn1-pf5ti6bmHUScaBE94ly7YW2r8QZwNseROgV_master.pdf. Consulté le 05 mars 20

[12] Cloud computing : définition, explication, histoire-IONOS. https://www.ionos.fr/digitalguide/serveur/know-how/cloud-computing/?.Consulté le 30 mars 2020.

[13] L'historique du Cloud computing-Infographie.http://www.cloud-entreprise.info/historique-cloud-computing/.Consulté le 27 mars 2020.

[14] Les-différantes-couches-des-services-du-Cloud-computing.png. https://www.institut-numerique.org/wp-content/uploads/2013/06/Les-diff%C3%A9rentes-couches-des-services-du-cloud-computing.png.Consulté le 1 mars 2020.

[15] I. Foster, Y. Zhao, I, Raicu, and S. Lu. Cloud computing and grid computing 360-degree compared. To appear at IEEE Grid Computing Environments (GCE08), co-located with IEEE/ACM Supercomputing, 2008.

[16] Les meilleurs grossistes de Cloud computing - guide 2 midi pyrenees. http://www.guide2midipyrenees.com/cloud-computing/ . Consulté le 1 mars 2020.

[17] Introduction au concept FMD : Fiabilité - Maintenabilité - Disponibilité. http://jackadit.com/index.php?page=indus3. Consulté le 1 avril 2020.

[18] Quels sont les avantages du Cloud Computing? - Cloudwatt. https://support.cloudwatt.com/kb/faq/lecloud/quels-sont-les-avantages-du-cloud-computing.html. Consulté le 1 avril 2020.

[19] Les inconvénients | Cloud computing. https://missarte.wordpress.com/les-inconvenients-du-cloud-computing/. Consulté le 1 avril 2020.

[20] Zwingelstein, Gilles. *Diagnostic des défaillances: théorie et pratique pour les systèmes industriels*. Hermès, 1995.

[21] Panne dictionnaire informatique : DicoFR. http://www.dicofr.com/cgi-bin/n.pl/dicofr/definition/20010101004213. Consulté le 7 avril 2020.

[22] Randell, Brian. "System structure for software fault tolerance." *Ieee transactions on software engineering* 2 (1975): 220-232.

[23] La tolérance aux pannes dans les systèmes répartis. http://deptinfo.cnam.fr/Enseignement/CycleSpecialisation/SAR/cours_tolerance.pdf. Consulté le 8 avril 2020.

[24] Systèmes et algorithme répartis - Tolérance aux fautes. http://queinnec.perso.enseeiht.fr/Ens/SAR/fautes.pdf. Consulté le 10 avril 2020.

[25] Mezouer, Leila, and NadjetMekkioui. *Tolérances aux Pannes dans les infrastructures Cloud Computing*. Rapport de PFE Master en Informatique. 2018. Université de Tlemcen.

[26] BENAHMED DAHO, Amel. *Détection préventive de pannes guidée par les données dans les réseaux de capteurs sans fil*. Rapport de PFE Master en Informatique. 2014. Université de Tlemcen.

[27] Laprie, Jean-Claude, et al. *Sûreté de fonctionnement des systèmes informatiques*. Dunod-Bordas, 1989.

[28] Sonna Momo, L. *Replication et Durabilité dans les systèmes répartis*. No. REP_WORK. 2001.

[29] Active and passive replication in distributed Systems. https://jaksa.wordpress.com/2009/05/01/active-and-passive-replication-in-distributed-systems/ . Consulté le 22 avril 2020.

[30] A. Mansouri, A. Tounsi. « Un mécanisme dynamique tolérant aux fautes Des données dans le Cloud Computing ». Mémoire de Master en informatique à l'université de SAIDA. 2017.

[31] NetBeans – définition. https://www.techno-science.net/definition/5346.html. Consulté le 13 mai 2020.

[32] Bienvenue à NetBeans. https://netbeans.org/index_fr.html. Consulté le 13 mai 2020.

[33] JFreeChart. https://elhedhly.wordpress.com/java/jfreechart/. Consulté le 13 mai 2020.

[34] The CloudSim Framework: Modeling and Simulating the cloud. https://opensourceforu.com/2014/03/cloudsim-framework-modelling-simulating-cloud-environment/. Consulté le 13 mai 2020.

[35] CloudSim, simulateur du cloud. https://graal.ens-lyon.fr/~ecaron/m2rts/2015/blogbazm/. Consulté le 13 mai 2020.

[36] Dad Djouhra. « Optimisation des performances des data centers des clouds sous contraintes d'énergie consommer ». Thèse de Doctorat en informatique à l'université d'Oran. 2016.

[37] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., Buyya, R. "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms." *Software: Practice and experience* 41.1 (2011): 23-50.

[38] Semmoud, A., Hakem, M., Benmammar, B., Charr, J. C. Load balancing in cloud computing environments based on adaptive starvation threshold. Concurrency and Computation: Practice and Experience, 32(11), e5652. 2020.

Résumé

La tolérance aux pannes dans un système distribué est l'aptitude de ce système à accomplir sa fonction malgré la présence de pannes. Le Cloud computing est un type de système distribué qui doit empêcher toute dégradation de son service en cas de panne en utilisant des techniques pour les tolérer. Dans ce travail, nous avons implémenté et évalué à l'aide du simulateur CloudSim une approche de tolérance aux pannes appliquée au Cloud computing. Notre approche est basée essentiellement sur la réplication hybride ainsi que sur le principe du Maximal Independent Set pour le choix des machines virtuelles afin d'effectuer la réplication sur la machine la moins chargée. Les résultats obtenus de ce travail sont très satisfaisants et montrent l'efficacité de notre solution en termes de taux de tâches exécutées.

Mots-clés : Cloud computing, tolérance aux pannes, systèmes distribués, réplication, MIS, CloudSim.

Abstract

Fault tolerance in a distributed system is the ability of that system to perform its function despite the presence of failures. Cloud computing is a type of distributed system that must prevent degradation of its service in the event of a breakdown by using techniques to tolerate it. In this work, we implemented and evaluated using the CloudSim simulator a fault tolerance approach applied to Cloud computing. Our approach is essentially based on hybrid replication as well as on the principle of the Maximal Independent Set for the choice of virtual machines in order to perform replication on the least loaded machine. The obtained results of our work are very satisfactory and demonstrate the effectiveness of our solution in terms of performed tasks rate.

Keywords: Cloud computing, fault tolerance, distributed systems, replication, MIS, CloudSim.

ملخص

التسامح مع الخطأ في النظام الموزع هو قدرة هذا النظام على أداء وظيفته على الرغم من وجود حالات الفشل. الحوسبة السحابية هي نوع من النظام الموزع الذي يجب أن يمنع تدهور خدمتها في حالة حدوث انهيار باستخدام تقنيات لتحملها. في هذا العمل ، قمنا بتطبيق وتقييم باستخدام محاكي CloudSim نهج تحمل الخطأ المطبّق على الحوسبة السحابية. يعتمد نهجنا بشكل أساسي على النسخ الهجين وكذلك على مبدأ المجموعة المستقلة القصوى لاختيار الأجهزة الافتراضية من أجل إجراء النسخ المتماثل على الجهاز الأقل تحميلًا. النتائج التي تم الحصول عليها من هذا العمل مرضية للغاية وتظهر فعالية حلنا من حيث معدل المهام المنجزة.

الكلمات المفتاحية: الحوسبة السحابية ، تحمل الأخطاء ، الأنظمة الموزعة ، النسخ المتماثل ,CloudSim, MIS.