PROJET SUR :

Qualité de Service dans le Cloud Computing

Effectué par :

William Staël KAYO

Elie Bou Saba

Sous la direction de : M. NADER MBAREK

TABLE DES MATIERES :

Listes des Figures :

Liste de Tableaux

Abréviations

QoS : Quality Of Service

REMERCIEMENT :

Nous tenons à remercier notre professeur, Mr Mbarek pour nous avoir permis de travailler sur ce projet très intéressant et pour le temps qui nous a consacré tout au long de ce projet**.**

INTRODUCTION GENERALE

Le Cloud Computing est un nouveau paradigme qui a pour objectif de permettre un accès à distance et d'une façon dynamique à un ensemble de ressources informatiques sous la forme d'un service. De nombreux opérateurs de cloud computing sont désormais actifs sur le marché et proposent une offre riche, comprenant des solutions d'infrastructure en tant que service (IaaS), de plate-forme en tant que service (PaaS) et de logiciel en tant que service (SaaS).

Bien que le cloud ait considérablement simplifié le processus d'approvisionnement en capacité, il pose plusieurs nouveaux défis dans le domaine de la gestion de la qualité de service (QoS). La QoS désigne les niveaux de performance, de fiabilité et de disponibilité offerts par une application et par la plateforme ou l'infrastructure qui l'héberge. La QoS est fondamentale pour les utilisateurs du cloud, qui attendent des fournisseurs qu'ils fournissent les caractéristiques de qualité annoncées, et pour les fournisseurs de cloud, qui doivent trouver les bons compromis entre les niveaux de QoS et les coûts opérationnels.

Pour apporter de la lumière sur la qualité de service dans le cloud, nous allons avoir 03 chapitres dans ce rapport. Le premier chapitre aborde l'étude des concepts relatifs au Cloud Computing. Le deuxième chapitre, met l'accent sur la QoS dans le Cloud Computing en mentionnant les différents travaux de standardisation, travaux de recherches et les projets menés sur la QoS dans le cloud computing. Le chapitre 3 quant à lui concerne la simulation d'un Cloud tout en assurant la QoS.

CHAPITRE 1 : Généralités Sur Le Cloud Computing

Introduction :

Nous allons dans ce premier chapitre, présenter le Cloud Computing (définition + caractéristiques), ensuite nous allons parler des différents modèles de service de Cloud et pour finir, nous allons voir les différents types de Cloud Computing.

I. Présentation du Cloud Computing

1. Définition

Il n'existe pas une définition exacte du cloud computing, mais plusieurs définitions ont été introduites, parmi lesquelles nous citons :

Selon **Wikipédia** :

« Le cloud computing est un concept qui consiste à déporter sur des serveurs distants des traitements informatiques traditionnellement localisés sur le poste client de l'utilisateur. » [1]

Selon **NIST (Institut national des normes et de la technologie des États-Unis** en français**)** :

« Le cloud computing est un modèle qui permet un accès réseau pratique et sur demande à un pool partagé de ressources informatiques configurables (par exemple, des réseaux, des serveurs, du stockage, des applications et des services) qui peut être rapidement approvisionné et disponible sans trop d'efforts de gestion ou d'interaction d'opérateurs. » [2]

D'une manière plus générale, le cloud computing est un concept qui propose des ressources informatiques sous forme de services à la demande, accessibles n'importe où et n'importe quand. Le cloud computing est devenu une partie importante de notre vie quotidienne. Avec cette technologie, la vision traditionnelle de l'informatique a été modifiée. Les exigences matérielles et logicielles de l'utilisateur et des entreprises diminuent.

2. Caractéristiques

Le Cloud Computing se distingue par les cinq caractéristiques essentielles suivantes :

Libre-service à la demande et à distance

L'utilisateur du cloud computing peut allouer les ressources et les services du Cloud lui-même lorsqu'il veut selon ses besoins sans interaction avec le fournisseur.

Les utilisateurs du cloud computing ne sont pas propriétaires des ressources informatiques qu'ils utilisent et ils ne connaissent pas l'emplacement physique de ces ressources. Ces dernières sont localisées dans des Datacenters chez les fournisseurs du Cloud Computing.

Élasticité

La capacité de stockage et la puissance de calcul des ressources peuvent être augmentées ou diminuées et elles sont adaptées automatiquement et rapidement à la demande des utilisateurs en fonction de leurs besoins.

Paiement et facturation à l'usage

Le service du cloud computing est mesuré et facturé par exemple en fonction de la durée, l'espace de stockage, la quantité de ressources utilisées, le nombre d'utilisateurs et la bande passante. Donc l'utilisateur ne paye que pour ce qu'il consomme et il est très facile pour lui d'arrêter un service du cloud computing s'il n'en a plus besoin et quand il veut.

Accès rapide via un réseau

Les services du cloud computing sont fournis via l'Internet ou un réseau privé VPN grâce à des mécanismes standards, généralement des protocoles Web et ils sont accessibles n'importe où, n'importe quand et depuis n'importe quel périphérique (PC, Mac, Tablette, SmartPhone...).

Mutualisation

Les ressources telles que la bande passante du réseau, machines virtuelles, mémoire, puissance de traitement, capacité de stockage sont mises en commun pour desservir plusieurs clients. En fait, les ressources virtuelles sont partagées par plusieurs utilisateurs et entreprises et elles sont affectées dynamiquement et réaffectées en fonction de leurs besoins. Autrement dit, une fois les ressources sont libérées par un utilisateur, d'autres clients peuvent les utiliser. Nous notons que c'est le fournisseur qui mutualise les ressources grâce à la virtualisation de ses serveurs, de son réseau (infrastructure et liaisons) et de ses capacités de stockage (SAN, NAS).

II. Modèle de services du Cloud Computing

Le cloud computing permet aux entreprises de consommer des services à la demande.

Ces services s'organisent en trois niveaux successifs : le niveau infrastructure (IaaS), le niveau plateforme (PaaS) et le niveau application (SaaS).

1. Infrastructure as a Service (IaaS)

L'IaaS ou l'infrastructure en tant que service permet aux entreprises de disposer à la demande d'une infrastructure matérielle virtuelle (serveurs virtuels, routeurs, commutateurs, espaces de stockage, machines virtuelles et systèmes d'exploitation) qui est localisée physiquement à distance dans des Datacenters chez le prestataire du service.

L'IaaS offre une grande flexibilité avec une administration à distance et permet aux entreprises de s'affranchir complètement de l'achat et de la gestion du matériel physique (coûts de gestion,

remplacement de matériel, climatisation, électricité…), ce qui les aide énormément à se concentrer en premier sur leurs processus métiers sans se préoccuper du matériel.

Le modèle IaaS offre plusieurs avantages à savoir : la réduction du coût des dépenses d'investissement, les utilisateurs paient pour le service qu'ils souhaitent, un accès à des ressources et à une infrastructure informatique de niveau entreprise et les utilisateurs peuvent augmenter ou réduire les ressources en fonction de leurs besoins à tout moment.

Parmi les principales entreprises proposant une infrastructure en tant que service figurent Rackspace Cloud Servers, Google, Amazon EC2, IBM et Verizon.

2. Platform as a Service (PaaS)

Le PaaS ou la plateforme en tant que service assure un environnement d'exécution en ligne qui fournit spécialement aux développeurs des plateformes (bases de données, serveurs d'application comme Apache Tomcat) sur lesquelles ils peuvent développer, tester, déployer et héberger leurs applications web via l'Internet.

Ce service évite donc aux développeurs et les entreprises d'acheter, d'installer et de gérer par exemple les logiciels, les middlewares et les bases de données. En contrepartie, il les aide à se concentrer sur la partie développement des applications.

L'un des inconvénients du PaaS est le manque d'interopérabilité et de portabilité entre les fournisseurs. Les consommateurs achètent l'accès aux plateformes, ce qui leur permet de déployer leurs propres logiciels et applications dans le cloud. Parmi les exemples de solutions PaaS, citons Rackspace Cloud Sites, Force.com de Salesforce.com, Google App Engine et Microsoft Azure.

Le modèle PaaS offre plusieurs avantages à savoir : Étant donné que la plupart du temps, de nombreuses personnes participent à la création d'applications dans des environnements PaaS, cela crée une forte communauté de soutien qui peut aider une équipe de développement tout au long de son processus. Le fournisseur PaaS se charge aussi de toutes les mises à niveau, des correctifs et de la maintenance logicielle de routine. Des coûts réduits car les entreprises courent moins de risques car elles n'ont pas à faire d'investissements initiaux en matériel et en logiciels. Un déploiement simplifié parce que l'équipe de développement peut se concentrer sur le développement de l'application sans avoir à se soucier de l'infrastructure de test et de déploiement.

3. Software as a Service (SaaS)

Le modèle SaaS apparaît au client comme une interface d'application basée sur le Web, où Internet est utilisé pour fournir des services accessibles à l'aide d'un navigateur Web.

Contrairement aux logiciels traditionnels, le SaaS présente l'avantage que le client n'a pas besoin d'acheter des licences, d'installer, de mettre à jour, de maintenir ou d'exécuter le logiciel sur son propre ordinateur. Les fournisseurs de service SaaS loue donc aux entreprises des applications en tant que service à la demande au lieu de leur facturer la licence des logiciels.
Les solutions SaaS offre plusieurs avantages à savoir : l'évolutivité rapide, l'accessibilité depuis n'importe quel endroit grâce à l'Internet, l'élimination des problèmes d'infrastructure, une maintenance et un support groupés.

Le Cloud Computing est donc la convergence de ses différents services SaaS, PaaS et IaaS. Ces services ont été conçus notamment pour aider les entreprises à réaliser des économies et à simplifier le système d'information en leur offrant la flexibilité et l'agilité dont elles ont besoin pour prospérer.
Par rapport au modèle classique client-serveur, où le client gère lui-même les différentes couches de sa solution logicielle, le Cloud Computing permet de confier certaines couches du modèle au prestataire Cloud, en fonction du niveau de service demandé par le client.
La figure 1 montre la répartition des responsabilités entre les fournisseurs et les clients en fonction du service du Cloud utilisé :

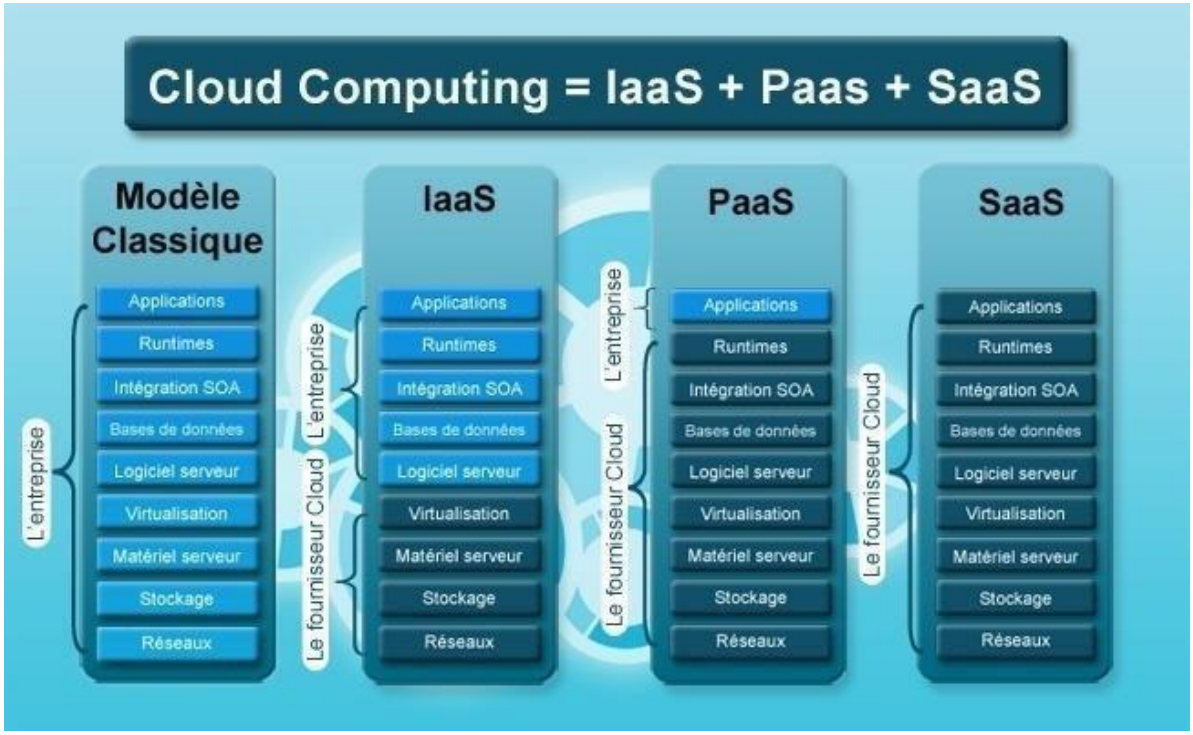

Figure 1 : Répartition de la responsabilité en fonction du service du cloud

III. Types de Cloud Computing

Il existe quatre types principaux du Cloud Computing selon les besoins des entreprises et des utilisateurs, à savoir : le Cloud public, le Cloud privé, le Cloud hybride et le Cloud communautaire.

1. Cloud Public

Le cloud public fournit une infrastructure et des services au public ou à toute organisation. Les ressources sont partagées par des centaines ou des milliers de personnes ce qui soulève beaucoup de questions de sécurité et de confidentialité. Il n'est donc pas conseillé de faire transiter des données sensibles sur un cloud public.

Ce type de cloud est géré par un fournisseur de services qui offre des services aux consommateurs sur la base d'un paiement à l'utilisation.

Amazon EC2, Google App Engine, etc. sont des exemples de cloud publics.

2. Cloud privé

Comme son nom l'indique, le cloud privé signifie qu'une seule entité utilise la plateforme. L'entreprise qui souscrit à un contrat avec un fournisseur de cloud privé garde le contrôle des ressources qu'elle possède ou qui lui sont dédiées. L'utilisation d'un cloud privé permet notamment d'assurer la confidentialité des données et d'assurer un mécanisme d'authentification quant à l'accès à ces dernières. On distingue les Cloud privés internes, utilisés par une entreprise pour satisfaire ses propres besoins. Ils sont administrés en interne par l'entreprise même. Il y a aussi les Cloud privés externes, destinés à satisfaire les besoins propres d'une entreprise cliente. Leurs gestions sont confiées à un prestataire extérieur de confiance.

3. Cloud Hybride

Le Cloud hybride combine le Cloud public et privé. Il donne aux entreprises la possibilité d'adopter le Cloud privé pour des applications et des données critiques ou très spécifiques et le Cloud public pour les applications moins risquées. Une entreprise peut donc stocker les données sensibles de ses clients en interne sur une application de cloud privé, mais interconnecter cette application à une application de facturation fournie sur un cloud public en tant que service logiciel. Quant à la sécurité des données sensibles, elles peuvent restées au sein du cloud privé de l'entreprise ; les données moins sensibles sont stockées au sein du Cloud Public.

4. Cloud Communautaire

Le Cloud communautaire est utilisé par plusieurs organisations partageant des intérêts communs et ayant des préoccupations communes afin de mutualiser leur service cloud en une grosse structure fortement sécurisée ; c'est le cas du projet "Andromède" en France. Cela permet de réduire davantage les coûts par rapport à un cloud privé, car l'infrastructure est partagée par de grands groupes. Différents services gouvernementaux au niveau de l'État qui ont besoin d'accéder aux mêmes données utilisent le modèle communautaire pour recueillir des informations. La sécurité et la conformité sont pris en charge.

Conclusion :

Dans ce premier chapitre, nous nous sommes essentiellement consacrés à l'étude théorique du cloud computing. Nous avons tout d'abord présenté de manière générale le Cloud Computing ; ensuite nous avons parlé des différents de services offerts par le Cloud Computing et pour finir nous avons vu les différents types de Cloud et leurs spécificités.

Chapitre 2 : QoS dans le Cloud Computing

Introduction :

Nous allons dans ce deuxième chapitre, tout d'abord présenter la QoS dans le Cloud Computing (définition + caractéristiques), ensuite nous allons parler du SLA (Service Level Agreement) et pour finir, nous allons voir les différents travaux de standardisation, travaux de recherches et les projets menés sur la QoS dans le cloud computing.

I. Définition et caractéristiques :

La qualité de service (QoS) désigne le niveau de performance, de fiabilité et de disponibilité offert par une application et par la plateforme ou l'infrastructure qui l'héberge.
Les acteurs et services dans le cloud ne cessent de se démultiplier. Cependant, les fournisseurs de services cloud sont conscient des demandes des utilisateurs, de l'importance et l'effet de la QoS sur leurs services mais ils doivent trouver les bons compromis ce qui ne rend pas la tâche facile. Dans les architectures cloud, la QoS intervient, par exemple pour le modèle de service NaaS[1], la QoS amènera des améliorations de la bande passante, de la gigue, du taux de perte de paquets ; dans le modèle de service laaS elle améliorera le temps de réponse.
La QoS est fondamentale pour les utilisateurs du cloud, qui attendent des fournisseurs qu'ils fournissent les caractéristiques annoncées, et pour les fournisseurs de cloud, qui doivent trouver le bon compromis entre les niveaux de QoS et les coûts opérationnels. Trouver le compromis optimal n'est pas un problème de décision facile car il implique des accords de niveau de service (SLA) qui spécifient les objectifs de qualité de service et les pénalités économiques associées aux violations des SLA. Les fournisseurs de services doivent se conformer aux contrats SLA qui déterminent les revenus et les pénalités sur la base du niveau de performance atteint. Les accords de niveau de service (SLA) sont signés entre le fournisseur de services et le client, la violation des SLA constituant une contrainte majeure. La violation des SLA est réduite par des mécanismes impliquant la surveillance.

II. SLA (Service Level Agreement)

Un SLA (service-level agreement) est un accord entre un fournisseur de services cloud et un client qui assure le maintien d'un niveau de service minimum. Il garantit les niveaux de fiabilité, de disponibilité et de réactivité des systèmes et des applications, précise qui est responsable en cas d'interruption de service et décrit les pénalités en cas de non-respect des niveaux de service.

[1] Network As A Service

L'accord de niveau de service (SLA) du cloud computing doit définir les responsabilités de chaque partie, les paramètres de performance acceptables, une description des applications et des services couverts par l'accord, les procédures de contrôle des niveaux de service et un calendrier de réparation des pannes. Les SLA utilisent généralement des définitions techniques pour quantifier le niveau de service, comme le temps moyen entre les pannes (MTBF) ou le temps moyen de réparation (MTTR), qui spécifie une valeur cible ou minimale pour les performances du niveau de service.

Le niveau de service défini doit être spécifique et mesurable, de manière à pouvoir être étalonné et, si l'accord le stipule, déclencher des récompenses ou des pénalités en conséquence.

Les clients peuvent surveiller les paramètres de service tels que le temps de fonctionnement, les performances, la sécurité, etc., par le biais par exemple des outils natifs d'un fournisseur de cloud computing. Une autre option consiste à utiliser un outil tiers pour suivre les bases de performance des services de cloud computing, y compris la manière dont les ressources sont allouées (par exemple, la mémoire dans une machine virtuelle, ou VM) et la sécurité.

La plupart des accords de niveau de service sont négociés pour répondre aux besoins actuels du client, mais de nombreuses entreprises changent radicalement de taille au fil du temps. Un contrat de niveau de service de cloud computing solide prévoit des intervalles au cours desquels le contrat est revu et éventuellement ajusté pour répondre à l'évolution des besoins de l'entreprise.

Certains fournisseurs intègrent des flux de notification qui se déclenchent lorsqu'un contrat de niveau de service de cloud computing est sur le point d'être rompu, de sorte que de nouvelles négociations peuvent être engagées en fonction des changements d'échelle. Cela peut concerner les niveaux de disponibilité ou l'utilisation qui dépasse les critères et peut justifier une mise à niveau vers un nouveau niveau de service.

Dans tous les cas, si un fournisseur de services cloud ne parvient pas à atteindre les objectifs minimums fixés, il doit payer la pénalité au consommateur conformément à l'accord. Les accords de niveau de service sont donc comme des polices d'assurance dans lesquelles la société doit payer conformément aux accords en cas de sinistre.

Microsoft publie les accords de niveau de service liés aux composants de la plateforme Windows Azure, ce qui constitue une pratique exemplaire pour les fournisseurs de services cloud. Chaque plateforme a ses propres accords de niveau de service.

III. Travaux de standardisation sur la QoS dans le cloud :

Plusieurs organismes de standardisation ont traité la QoS dans le cloud. Les travaux menés par ces organismes sont encore en cours pour développer différentes propositions afin d'aboutir à des standards approuvés. On mentionne les différents organismes tels que : le DMTF (Distributed Management Task Force) qui propose des activités sur la QoS, le SLA, les politiques et la sécurité dans le cloud réalisées dans le groupe de travail « Cloud Management Working Group (CMWG) ». L'ONF (Open networking Foundation) est une organisation d'utilisateur consacrée à la promotion et l'adoption du Software-Defined Networking (SDN) et travaillant sur l'élaboration des mécanismes de QoS dans le protocole de communication OpenFlow. La CDMI (SNIA Cloud Data Management Interface) travaille sur la mise en œuvre de spécifications visant à améliorer les capacités de qualité de service des ressources de stockage distribuées. Le Cloud Standards Customer Council (CSCC) est un groupe de défense des utilisateurs finaux qui se consacre à l'accélération de l'adoption du cloud et à la résolution des problèmes de normes, de sécurité et d'interopérabilité liés à la transition vers le cloud. Ce groupe a publié un document sur le SLA : « Practical Guide to Cloud Service Level Agreements Version 3.0 » qui est un bon guide pour l'évaluation des contrats de services dans le cloud.

IV. Projets sur la QoS dans le Cloud

Nous présentons quelques projets qui nous ont parus intéressants sur la QoS dans le cloud.

1. MyCloud

L'objectif du projet MyCloud est de définir et de mettre en œuvre un nouveau modèle de cloud : SLAaaS (SLA aware Service). Le modèle SLAaaS enrichit le paradigme général du cloud computing et permet une intégration systématique et transparente des niveaux de service et des SLA dans le cloud. SLAaaS peut s'appliquer à n'importe quel modèle de cloud que ce soit IaaS, PaaS ou SaaS. Le projet MyCloud tient compte à la fois du point de vue du fournisseur et du point de vue du client. Du point de vue du fournisseur cloud, MyCloud propose une gestion autonome des accords de niveau de service (SLA) pour gérer les problèmes de performance, de disponibilité, d'énergie et de coûts dans le cloud. Une approche innovante utilisant des algorithmes distribués. D'autre part, du point de vue du client, le projet MyCloud fournit une gouvernance SLA. Il permet aux clients de faire partie de la boucle et d'être automatiquement informés de l'état du cloud, comme la violation des SLA et la consommation d'énergie du cloud. Le premier cas offre une plus grande transparence sur les garanties SLA, et le second vise à sensibiliser les clients à l'empreinte énergétique du cloud. [3]

2. Qu4DS (Quality Assurance for Distributed Services)

QU4DS sert d'interface avec les clients des services cloud tout en coordonnant les activités de gestion des SLA. Son objectif est de combler cette lacune dans le cycle de vie de la QoS en fournissant des mécanismes qui garantissent les termes du contrat convenu. QU4DS fournit une solution qui est capable de : négocier les objectifs de qualité de service avec le client ; traduire les paramètres de qualité de service et les configurations de ressources de manière bidirectionnelle ; déployer automatiquement le service sur les ressources appropriées ; et garantir la qualité de service convenue en réagissant aux changements sous-jacents dans les infrastructures tout en restant conforme aux objectifs de qualité de service. [4]

3. EASI-CLOUDS

"Extendable Architecture and Service Infrastructure for Cloud-Aware Software" (EASI-CLOUDS) est un projet de recherche financé par le ministère allemand de l'éducation et de la recherche. L'objectif d'EASI-CLOUDS est de développer des normes ouvertes pour l'interopérabilité entre les cloud. Ces normes seront bénéfiques pour l'évolutivité et la disponibilité, ainsi que pour la collaboration et la concurrence entre les fournisseurs cloud tout en permettant une gestion avancée du SLA. [5]

V. Travaux de recherches sur la QoS dans le Cloud

Nous présentons quelques travaux de recherches qui nous ont parus intéressants sur la QoS dans le cloud. Linlin Wu et al [6] proposent des algorithmes d'allocation de ressources pour les fournisseurs de SaaS qui souhaitent minimiser le coût de l'infrastructure et les violations de SLA. Les algorithmes proposés sont conçus de manière à ce que les fournisseurs SaaS soient capables de gérer le changement dynamique des clients, en faisant correspondre les demandes des clients aux paramètres du niveau d'infrastructure et en gérant l'hétérogénéité des machines virtuelles. Les paramètres de qualité de service des clients, tels que le temps de réponse, et les paramètres de niveau d'infrastructure, tels que le temps de lancement du service sont pris en compte. Ils présentent également une étude d'évaluation approfondie afin d'analyser et de démontrer que les algorithmes poposés minimisent le coût du fournisseur de SaaS et le nombre de violations de SLA dans un environnement de partage dynamique des ressources dans le Cloud. Cicotti et al [7] proposent QoSMONaaS (Quality of Service MONitoring as a Service), une installation de surveillance de la qualité de service construite sur le SRT-15, une plate-forme orientée vers le cloud. Ye et al ont proposé un modèle de QoS extensible pour calculer

les valeurs de QoS des services dans le cloud computing. Une approche basée sur un algorithme génétique est proposée pour composer les services dans le cloud computing. Nathuji et al [8] ont développé Q-Clouds, un cadre de contrôle tenant compte de la QoS qui ajuste l'allocation des ressources pour atténuer les effets d'interférence sur les performances. Q-Clouds utilise le feedback en ligne pour construire un modèle MIMO (multi-input multi-output) qui capture les interactions d'interférence de performance, et l'utilise pour effectuer une gestion des ressources en boucle fermée. Vincent C. Emeakaroha dans sa thèse [9], propose une nouvelle infrastructure de gestion du Cloud, qui est basée sur des techniques et des mécanismes de surveillance. Il présente aussi la conception et la mise en œuvre de ces techniques. Dans une étude de cas, il montre l'intégration des techniques de gestion des connaissances dans les infrastructures de gestion du Cloud qui réalisent un comportement autonome et fournissent une action réactive pour prévenir / corriger les situations de violation des SLA.

Conclusion

La gestion flexible et fiable des ressources et les accords SLA sont d'une importance capitale tant pour les fournisseurs que pour les consommateurs du Cloud. D'une part, les fournisseurs doivent prévenir les violations des SLA pour éviter les pénalités et, d'autre part, ils doivent garantir une utilisation élevée des ressources pour éviter la maintenance coûteuse des ressources inutilisées. Bien que de nombreux travaux aient été consacrés au développement d'infrastructures d'informatique cloud flexibles et autogérables, il n'existe toujours pas d'infrastructures de surveillance adéquates capables de prédire les éventuelles violations des accords de niveau de service.

Chapitre 3 : Simulation avec Cloudsim

Introduction :

Le cloud est un système distribué. Gérer les différents paramètres dans un tel système pour aboutir à la solution la plus optimale au niveau de la performance et la stabilité n'est pas une tâche simple à réaliser. Il n'est pas possible d'effectuer des expériences d'analyse comparative de manière reproductible, fiable et évolutive en utilisant des environnements Cloud réels. Une alternative plus viable est l'utilisation d'outils de simulation.Il existe plusieurs outils de simulation qu'on peut utiliser à savoir : GridSim, CloudSim et Simgrid.

Dans la suite de notre projet, nous allons utiliser Cloudsim pour la simulation d'un Cloud.

I. Présentation CloudSim :

Dans le cadre de notre projet, le simulateur CloudSim a été utilisé. C'est une librairie écrite en Java développé par l'organisation CLOUDS Lab. de l'université de Melbourne qui nous permet de simuler l'environnement du cloud computing avec ses services. Plusieurs chercheurs d'organisations, telles que HP Labs aux États-Unis, utilisent CloudSim dans leur enquête sur l'approvisionnement des ressources Cloud et la gestion éco-énergétique des ressources du centre de données.

Les fonctionnalités de CloudSim :

CloudSim propose plusieurs fonctionnalités à savoir : la modélisation et la simulation d'un environnement informatique à grande échelle (cloud data centers, hôtes de serveurs virtualisés avec des politiques personnalisables, mise en file d'attente et traitement des événements); une flexibilité dans le choix de type d'allocation des services virtualisés (allocation en espace partagé ou allocation en temps partagé) et une plateforme autonome pour la modélisation des services brokers, des politiques d'approvisionnement et d'allocation du cloud.

Les bénéfices de CloudSim :

CloudSim nous permet de tester les différents services cloud dans un environnement reproductible et contrôlable; d'expérimenter plusieurs scénarios de performance des ressources avant l'implémentation réel; Facile à mettre en place une application basée sur l'environnement cloud de test d'approvisionnement.[10] [11]

Architecture du CloudSim :

Afin que nous puissions simuler notre environnement, il faut d'abord connaître l'architecture de ClouSim. La figure 2 illustre les différentes couches de la structure de CloudSim et ses éléments architecturaux.

Par la suite, il existe plusieurs couches principales qui assurent le fonctionnement du système simulé. La couche supérieure de ce simulateur est le code utilisateur qui est contrôlée par l'utilisateur et propose des entités de base pour les hôtes comme le nombre de machines et leurs caractéristiques, les applications avec les tâches à effectuer, etc... En fait, chaque entité est une instance d'objet définit par une classe. La couche au milieu qui est le CloudSim gère l'instanciation et l'exécution des entités de base comme les machines virtuelles, hôtes, Datacenters, applications. Au niveau le plus bas, on a la couche « CloudSim Core Simulation Engine » qui est le moteur de simulation des événements qui implémente les fonctionnalités de base requises pour les cadres de simulation au niveau supérieur.

Un environnement de développement comme Eclipse ou NetBeans doit être utilisé pour implémenter ce Framework et faire de la programmation en Java. [10] [11]

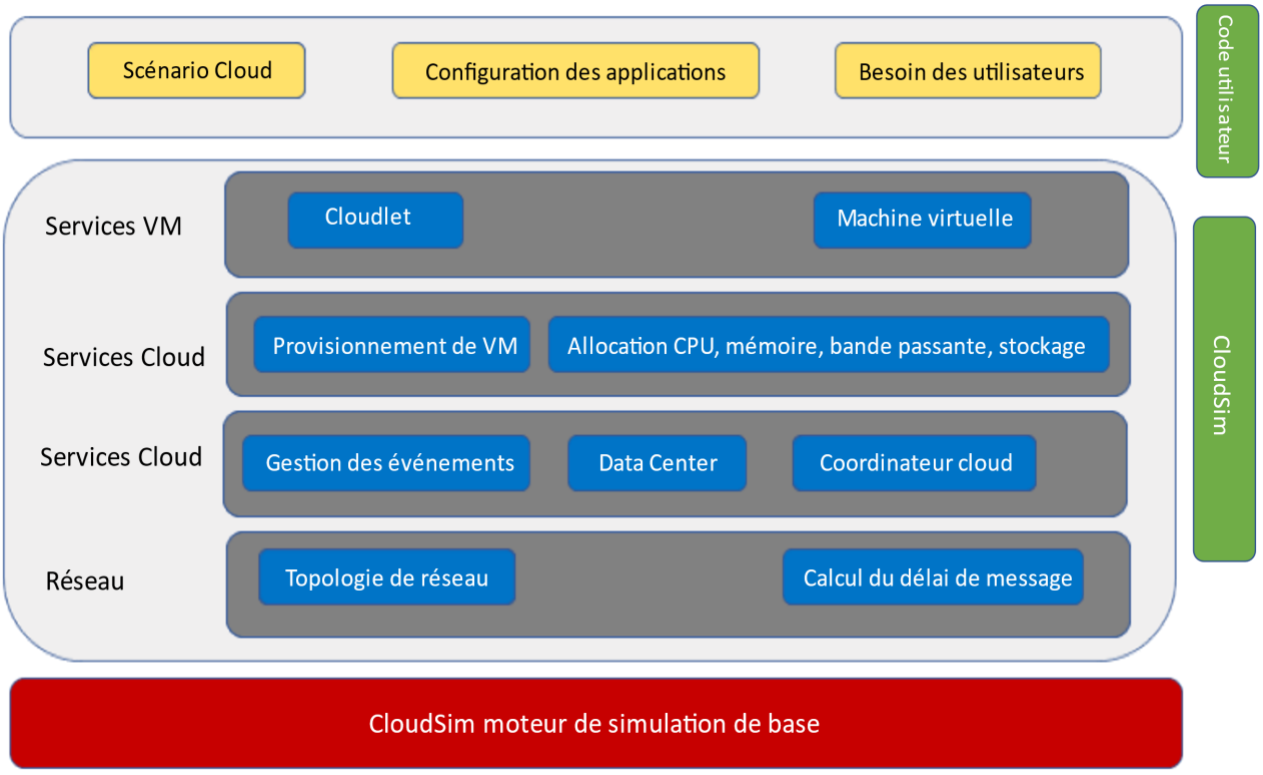

Figure 2 : Architecture de CloudSim

Les entités principales de la simulation :

- Data Center : Modélisation de matériel de base de l'environnement Cloud. Les Data Centers possèdent des hôtes qui eux hébergent les machines virtuelles.

- Hôte : Cette classe modélise une ressource physique telle qu'un serveur de stockage. Peut exécuter des actions liées à la gestion des machines virtuelles (Définir des politiques d'approvisionnement de la mémoire, de la bande passante ainsi que l'allocation des cœurs de CPU aux machines virtuelles.
- VM : Définit une machine virtuelle et des caractéristiques (RAM, bande passante, mips qui signifie millions d'instructions par seconde, taille…)
- Cloudlet : Tâche exécutée sur une machine virtuelle comme une opération d'accès à la mémoire, ou une mise à jour de fichiers, etc…
- CloudSim : Responsable de l'initialisation et du démarrage de l'environnement de simulation une fois que toutes les entités cloud nécessaires ont été définies et de l'arrêt ultérieur une fois que toutes les entités ont été détruites. [10] [11]

Comprendre le CloudSim Framework :

Avant d'entrer dans l'implémentation de cette librairie, il faut comprendre le processus de fonctionnement du CloudSim.

Sur la figure 3, on trouve le CIS (Common Information System) qui est un type d'enregistrement contenant les ressources (Data Center et hôtes) qui sont disponibles sur le cloud. Après la création du Data Center, le CIS va l'enregistrer. Chaque Data Center contient des hôtes et chaque hôte peut avoir une ou plusieurs machines virtuelles. Chaque hôte a des caractéristiques précises (Élément de traitement, Mémoire RAM) et grâce à la virtualisation, on aura plusieurs machines virtuelles ayant chacune sa propre caractéristique et partageant les ressources de l'hôte suite à plusieurs politiques : La Politique d'allocation de machine virtuelle (VmAllocationPolicy) : Utilisé par le Data Center dans le but d'allouer des machines virtuelles à l'hôte. Cette politique prend en compte plusieurs caractéristiques au niveau du matériel comme le nombre de cœurs de processeur, le partage de processeur et autres. La politique du planificateur de machine virtuelle (Utilisé par l'hôte chaque fois que le traitement doit être effectué sur la machine virtuelle) et la politique du planificateur du cloudet (Utilisé pour le traitement des cloudlets sur les machines virtuelles. Toutes ces politiques sont soit partagées sur le temps soit partagées sur l'espace.

Effectivement, en utilisant le simulateur CloudSim il n'y a pas d'entités réelles disponibles pour simuler des entités de réseau, telles que des routeurs ou commutateurs. Par contre, la latence du réseau qui s'applique sur les communications entre les différentes entités est mesurée en fonction des informations de latences stockés dans une matrice ce qui est une technique beaucoup plus simple et légère que de simuler des équipements de réseaux complexes comme les commutateurs et les routeurs.

Le cloudlet est une tâche à effectuer sur une machine virtuelle qui sera envoyé au Data Center Broker qui lui, est le lien qui achemine cette tâche vers la machine virtuelle dans le Data Center. En premier temps, le Data Center broker communique avec le CIS pour demander les ressources enregistrées par le CIS et cela est vue sur la figure 3 avec la flèche numérotée 1. Ensuite il y aura la possibilité de communiquer directement avec le Data Center sans passer par le CIS et cela est marqué par la flèche numéro 2. [10] [11]

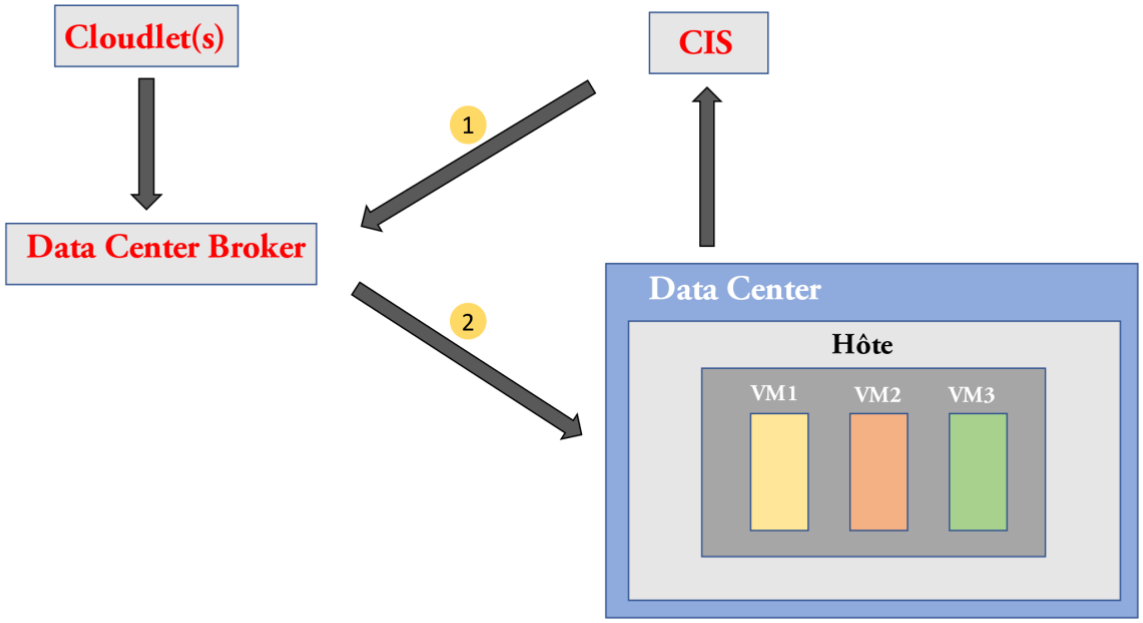

Figure 3 : Fonctionnement du CloudSim

II. Exemple d'implémentation du CloudSim Framework :

On a simulé un exemple d'environnement Cloud avec l'outil CloudSim. Pour le faire, on a pris l'exemple de deux Data Centers avec un Data Center Broker qui fera le lien entre les cloudlets et les Data Centers et un CIS sera aussi créer. Après, on a créé deux hôtes ayant des caractéristiques différentes. Le premier hôte contient 4 cœurs de traitement et le deuxième hôte contient 2 cœurs. Concernant les cloudlet, on a créé 10 cloudlets avec chacun une taille différente de l'autre ce qui est le cas normalement dans la vie réelle. La longueur définie pour un cloudlet est un ensemble estimé d'instructions qui vont être exécutées dans une simulation de test. Parallèlement à cela, nous spécifions deux autres attributs : pour la taille du fichier

d'entrée (cet attribut définit la taille totale des données d'entrée en octets) et pour la taille du fichier de sortie (définit les calculs liés à la bande passante).
On a aussi créé 10 machines virtuelles.
Sur la figure 4, on retrouve les éléments suivants : le Cloudlet ID, l'identifiant du Data Center où le cloudlet a été exécuté (Data center ID), l'identifiant de la machine virtuelle sur laquelle on a exécuté le cloudlet (VM ID), le temps émis pour l'exécution du cloudlet (Time), la date début et date fin de l'exécution de cloudlet (Start Time et Finish Time).

```
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID        Time    Start Time    Finish Time
    6          SUCCESS         2              1         2.04        0.1           2.14
    3          SUCCESS         2              3         2.15        0.1           2.25
    0          SUCCESS         2              0         2.26        0.1           2.36
    1          SUCCESS         2              1         2.26        0.1           2.36
    4          SUCCESS         2              4         2.26        0.1           2.36
    8          SUCCESS         2              3         2.61        0.1           2.71
    5          SUCCESS         2              0         2.72        0.1           2.82
    9          SUCCESS         2              4         2.72        0.1           2.82
    7          SUCCESS         2              2         3.66        0.1           3.76
    2          SUCCESS         2              2         3.77        0.1           3.87
CloudSimExample finished!
```

Figure 4 : Résultat de la simulation

Sur la figure 5, on peut voir les différentes étapes qui ont été exécuté lors du début de lancement de la simulation jusqu'à la fin de la simulation. Le premier champ à la ligne 8 indique le temps auquel une opération a été faite. On remarque la création des 5 machines virtuelles et leur remplacement dans le Datacenter_0. Ensuite ces machines sont attribuées à un des deux hosts existants. Toutes ces informations sont renseignées par le Data Center Broker. Après, le broker à envoyer au temps 0.1 les cloudlets vers les différentes machines virtuelles et on peut voir aussi à quel temps les cloudets ont été reçus vers les machines et quand ils ont été exécutés. Après l'exécution de tous les cloudlets, les machines virtuelles sont détruites et on arrête la simulation. Le code lié à la simulation avec CloudSim est présent dans l'annexe du rapport.

```
Starting CloudSimExample...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Datacenter_1 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 2 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
0.0: Broker: Trying to Create VM #2 in Datacenter_0
0.0: Broker: Trying to Create VM #3 in Datacenter_0
0.0: Broker: Trying to Create VM #4 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: VM #1 has been created in Datacenter #2, Host #0
0.1: Broker: VM #2 has been created in Datacenter #2, Host #0
0.1: Broker: VM #3 has been created in Datacenter #2, Host #1
0.1: Broker: VM #4 has been created in Datacenter #2, Host #0
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Sending cloudlet 1 to VM #1
0.1: Broker: Sending cloudlet 2 to VM #2
0.1: Broker: Sending cloudlet 3 to VM #3
0.1: Broker: Sending cloudlet 4 to VM #4
0.1: Broker: Sending cloudlet 5 to VM #0
0.1: Broker: Sending cloudlet 6 to VM #1
0.1: Broker: Sending cloudlet 7 to VM #2
0.1: Broker: Sending cloudlet 8 to VM #3
0.1: Broker: Sending cloudlet 9 to VM #4
2.142: Broker: Cloudlet 6 received
2.252: Broker: Cloudlet 3 received
2.3619999999999997: Broker: Cloudlet 0 received
2.3619999999999997: Broker: Cloudlet 1 received
2.3619999999999997: Broker: Cloudlet 4 received
2.7139999999999995: Broker: Cloudlet 8 received
2.8239999999999994: Broker: Cloudlet 5 received
2.8239999999999994: Broker: Cloudlet 9 received
3.7579999999999996: Broker: Cloudlet 7 received
3.8679999999999994: Broker: Cloudlet 2 received
3.8679999999999994: Broker: All Cloudlets executed. Finishing...
3.8679999999999994: Broker: Destroying VM #0
3.8679999999999994: Broker: Destroying VM #1
3.8679999999999994: Broker: Destroying VM #2
3.8679999999999994: Broker: Destroying VM #3
3.8679999999999994: Broker: Destroying VM #4
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Datacenter_1 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.
```

Figure 5 : Étapes détaillées des différentes étapes de l'exécution de la simulation

III. QoS dans le CloudSim Framework

Le simulateur CloudSim par défaut n'offre pas la qualité de service. Par exemple pour le traitement des tâches par priorité, CloudSim ne supporte que FIFO (premier arrivé, premier servit) et ne supporte pas la mise en file d'attente par priorité donc par exemple si la politique de partage du temps est sélectionnée et que tous les cloudlets ont le même temps de traitement, ils commencent et se terminent en même temps. Effectivement, CloudSim calcule le temps d'exécution du cloudlet en utilisant les performances données dédiés aux machines virtuelles. En fait, le traitement prioritaire s'applique différemment entre le partage sur l'espace et le partage sur le temps.

Dans la politique de partage d'espace, les cloudlets soumises sont mises dans le tableau contenant l'ensemble des cloudlets. On pourra trier la liste des cloudlets par priorité avant que

l'exécution des cloudlets ne commence, le simulateur exécutera le cloudlet par sa priorité et donc les cloudlets ayant une priorité plus élevée termineront l'exécution plut tôt ce qui va permettre d'assurer une qualité de service avec la mise en file d'attente en se basant sur la priorité sur CloudSim.

Pour le faire, on a besoin de deux nouvelles fonctions. La première fonction va trier les différents cloudlets selon leurs priorités. Par la suite, la deuxième fonction permet l'exécution du cloudlet ayant la priorité la plus élevé en premier.

```
function priorityCloudlet(cloudlets) {
        cloudletlist = new DoubleArray ();//2-dimension array
        foreach (cloudlets as entity) {
                int priority = entity.getPriority();
                cloudletlist[priority].addlist(entity);
        }
return cloudletlist;
}

function processCloudletReturn(cloudlet) {
CloudletReceivedList.add(cloudlet);
if(cloudletlist.size() > 0){
        submitNextPriorityCloudlets();
}
else if(cloudletlist.size() == 0){
        FinishExecution();
}
```

Figure 6 : Fonctions à ajouter

		Cloudlet 1	Cloudlet 2	Cloudlet 3	Cloudlet 4
Time cost		20	10	20	10
FIFO	Start	0	0	0	0
	Finish	80	50	80	50
Priority service	Priority	2	3	1	5
	Start	40	20	60	0
	Finish	60	40	80	20

Tableau 1 : Affichage des différents résultats de la simulation

On remarque dans le tableau 1 qu'avec la mise en file d'attente par priorité, les cloudlets ayant la plus haute priorité vont être exécutés en premier temps ce qui va prioriser les tâches qu'on souhaite exécuter rapidement et offrir une sorte de qualité de service.

Conclusion :

La simulation d'un cloud est une phase primordiale qui doit être faite avant la vrai implémentation de n'importe quel service de cloud dans le but de satisfaire pleinement les besoins et attentes des utilisateurs. Comme vue précédemment, CloudSim peut calculer le temps de traitement de tous les cloudlets, mais il peut utiliser uniquement le service FIFO et donc n'assure pas une qualité de service proprement dit. L'un des inconvénients de CloudSim est qu'il n'offre pas une interface graphique aux utilisateurs.

Conclusion Générale

En définitive, nous avons tout d'abord fait un état de l'art du Cloud Computing en présentant le concept et en donnant les modèles et types de services Cloud. Ensuite, nous sommes entrés dans l'étude de la qualité de service dans le cloud en parlant du SLA (Service Level Agreement), des différents travaux de standardisation, travaux de recherches et les projets menés sur la QoS dans le cloud computing. Pour finir, nous avons simulé un cloud assurant la QoS.

Bien que les propriétés de la QoS aient fait l'objet d'une attention constante bien avant l'avènement du cloud computing, l'hétérogénéité des performances et les mécanismes d'isolation des ressources des plateformes de cloud computing ont considérablement compliqué l'analyse, la prédiction et l'assurance de la QoS. Cela a incité plusieurs chercheurs à étudier des méthodes de gestion automatisée de la QoS. Nous avons pu nous rendre compte de l'engouement énorme autour de la QoS dans le cloud, par le nombre de travaux de recherches et projets existants.

Pour conclure, ce projet a été une réelle source d'enrichissement, tant au niveau de la théorie que de la technique. Nous connaissons maintenant quelques détails sur la qualité de service dans le cloud et nous sommes aussi en mesure de simuler un cloud sur l'outil CloudSim.

Annexe :

```java
package org.cloudbus.cloudsim.examples;

import java.text.DecimalFormat;

public class CloudSimExample6 {

    // Liste contenant les cloudlets
    private static List<Cloudlet> cloudletList;

    // Liste des machines virtuelles
    private static List<Vm> vmlist;

    private static List<Vm> createVM(int userId, int vms) {

        // Creation d'un container pour sauvegarder les VM
        LinkedList<Vm> list = new LinkedList<Vm>();

        // Paramètres de la machine virtuelle
        long size = 10000; // Taille d'image (MB)
        int ram = 512; // Mémoire de la machine (MB)
        int mips = 1000; // Million d'instructions par seconde
        long bw = 1000; // Bande passante
        int pesNumber = 1; // Nombre d'élémenet de traitement
        String vmm = "Xen"; // Nom de la machine virtuelle

        // Creation de la machine virtuelle
        Vm[] vm = new Vm[vms];

        for(int i=0;i<vms;i++){
            // Creation d'une machine virtuelle avec politique de planification partagée sur le temps pour les cloudlets
            vm[i] = new Vm(i, userId, mips, pesNumber, ram, bw, size, vmm, new CloudletSchedulerTimeShared());
            // Creation d'une machine virtuelle avec politique de planification partagée sur l'éspace pour les cloudlets
            //vm[i] = Vm(i, userId, mips, pesNumber, ram, bw, size, priority, vmm, new CloudletSchedulerSpaceShared());

            list.add(vm[i]);
        }

        return list;
    }


    private static List<Cloudlet> createCloudlet(int userId, int cloudlets){
        // Creation d'un container pour sauvegarder les cloudlets
        LinkedList<Cloudlet> list = new LinkedList<Cloudlet>();

        // Paramètres du cloudlet
        long length = 1000;
        long fileSize = 300;
        long outputSize = 300;
        int pesNumber = 1;
        UtilizationModel utilizationModel = new UtilizationModelFull();

        Cloudlet[] cloudlet = new Cloudlet[cloudlets];

        // Creation des nombre aléatoires pour avoir des tailles de cloudlet différent
        Random seed = new Random();
```

```
    for(int i=0;i<cloudlets;i++){

        long addonLen = seed.nextInt(1000);
        Log.printLine("La taille du cloudlet est :" + (length+addonLen));
        cloudlet[i] = new Cloudlet(i, (length+addonLen), pesNumber, fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);
        // définir le propriétaire des cloudlets
        cloudlet[i].setUserId(userId);
        list.add(cloudlet[i]);
    }

    return list;
}



public static void main(String[] args) {
    Log.printLine("Starting CloudSimExample...");

    try {
        // Initialiser la répértoire de CloudSim
        int num_user = 1;   // Nombre d'utilisateurs
        Calendar calendar = Calendar.getInstance();
        boolean trace_flag = false;  // Pour le suivi des événements

        // Initialiser la librairie CloudSim
        CloudSim.init(num_user, calendar, trace_flag);

        // Creation de deux Data Center
        @SuppressWarnings("unused")
        Datacenter datacenter0 = createDatacenter("Datacenter_0");
        @SuppressWarnings("unused")
        Datacenter datacenter1 = createDatacenter("Datacenter_1");

        // Creation du Data Center broker
        DatacenterBroker broker = createBroker();
        int brokerId = broker.getId();

        // Creation de machines virtuelles
        vmlist = createVM(brokerId,20); // 20 machines virtuelles
        //Creation de cloudlets
        cloudletList = createCloudlet(brokerId,40); // 40 cloudlets

        // Envoi vers le Data Center Broker
        broker.submitVmList(vmlist);
        broker.submitCloudletList(cloudletList);

        // Lancer la simulation
        CloudSim.startSimulation();

        // Affichage des resultats quand la simulation est terminée
        List<Cloudlet> newList = broker.getCloudletReceivedList();

        CloudSim.stopSimulation();

        printCloudletList(newList);

        Log.printLine("CloudSimExample finished!");
    }
```

```
    catch (Exception e)
    {
        e.printStackTrace();
        Log.printLine("The simulation has been terminated due to an unexpected error");
    }
}

private static Datacenter createDatacenter(String name){

    // List contenant les hôtes.
    List<Host> hostList = new ArrayList<Host>();

    // List1 contenant les éléments de traitement du premier hôte
    List<Pe> peList1 = new ArrayList<Pe>();

    int mips = 1000;

    // Creation des éléments de traitement et ajout à la liste1.
    peList1.add(new Pe(0, new PeProvisionerSimple(mips))); // need to store Pe id and MIPS Rating
    peList1.add(new Pe(1, new PeProvisionerSimple(mips)));
    peList1.add(new Pe(2, new PeProvisionerSimple(mips)));
    peList1.add(new Pe(3, new PeProvisionerSimple(mips)));

    // List2 contenant les éléments de traitement du deuxième hôte
    List<Pe> peList2 = new ArrayList<Pe>();

    // Creation des éléments de traitement et ajout à la liste2.
    peList2.add(new Pe(0, new PeProvisionerSimple(mips)));
    peList2.add(new Pe(1, new PeProvisionerSimple(mips)));

    // Creation des 2 hôtes (premier hôte est un quad-core et deuxième hôte dual-core)
    int hostId=0;
    int ram = 2048; //host memory (MB)
    long storage = 1000000; //host storage
    int bw = 10000;

    //Premier hôte
    hostList.add(
            new Host(
                hostId,
                new RamProvisionerSimple(ram),
                new BwProvisionerSimple(bw),
                storage,
                peList1,
                new VmSchedulerTimeShared(peList1)
            )
        );

    hostId++;
```

```
    //Deuxième hôte
    hostList.add(
            new Host(
                hostId,
                new RamProvisionerSimple(ram),
                new BwProvisionerSimple(bw),
                storage,
                peList2,
                new VmSchedulerTimeShared(peList2)
            )
        );


    // Pour créer un hôte avec une stratégie d'allocation d'espace partagé pour les éléments de traitement
    // vers les machines virtuelles
    // hostList.add(
    //       new Host(
    //           hostId,
    //           new CpuProvisionerSimple(peList1),
    //           new RamProvisionerSimple(ram),
    //           new BwProvisionerSimple(bw),
    //           storage,
    //           new VmSchedulerSpaceShared(peList1)
    //       )
    //   );




    // Creation des caractéristics d'un objet Data Center contenant des propriétés d'un Data Center
    // Architecture, Système d'exploitation, liste des machines, politique d'allocation (Partagé sur
    // le temps ou sur l'éspace et le coût de chaque ressource utilisé

    String arch = "x86";              // Architecture du système
    String os = "Linux";              // Système d'exploitation
    String vmm = "Xen";
    double time_zone = 10.0;          // Time zone
    double cost = 3.0;                // Le coût d'utilisation du traitement dans cette ressource
    double costPerMem = 0.05;         // Le coût d'utilisation de la mémoire dans cette ressource
    double costPerStorage = 0.1;      // Le coût d'utilisation de stockage dans cette ressource
    double costPerBw = 0.1;           // Le coût d'utilisation de la bande passante dans cette ressource
    LinkedList<Storage> storageList = new LinkedList<Storage>();

    DatacenterCharacteristics characteristics = new DatacenterCharacteristics(
            arch, os, vmm, hostList, time_zone, cost, costPerMem, costPerStorage, costPerBw);


    Datacenter datacenter = null;
    try {
        datacenter = new Datacenter(name, characteristics, new VmAllocationPolicySimple(hostList), storageList, 0);
    } catch (Exception e) {
        e.printStackTrace();
    }

    return datacenter;
}
```

```
    private static DatacenterBroker createBroker(){

        DatacenterBroker broker = null;
        try {
            broker = new DatacenterBroker("Broker");
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
        return broker;
    }

    // Affichage des cloudlets
    private static void printCloudletList(List<Cloudlet> list) {
        int size = list.size();
        Cloudlet cloudlet;

        String indent = "    ";
        Log.printLine();
        Log.printLine("========== OUTPUT ==========");
        Log.printLine("Cloudlet ID" + indent + "STATUS" + indent +
                "Data center ID" + indent + "VM ID" + indent + indent + "Time" + indent + "Start Time" + indent + "Finish Time");

        DecimalFormat dft = new DecimalFormat("###.##");
        for (int i = 0; i < size; i++) {
            cloudlet = list.get(i);
            Log.print(indent + cloudlet.getCloudletId() + indent + indent);

            if (cloudlet.getCloudletStatus() == Cloudlet.SUCCESS){
                Log.print("SUCCESS");

                Log.printLine( indent + indent + cloudlet.getResourceId() + indent + indent + indent + cloudlet.getVmId() +
                        indent + indent + indent + dft.format(cloudlet.getActualCPUTime()) +
                        indent + indent + dft.format(cloudlet.getExecStartTime())+ indent + indent + indent + dft.format(cloudlet.getFinishTime()));
            }
        }

    }
}
```

Bibliographie :

[1] https://fr.wikipedia.org/wiki/Cloud_computing

[2] http ://www.nist.gov/itl/cloud/upload/cloud-def-v15.pdf

[3] http ://mycloud.inrialpes.fr/

[4] http ://www.irisa.fr/myriads/software/quads/

[5] http ://easi-clouds.eu/2012/02/03/project-description/

[6] L. Wu, S. Garg et R. Buyya, «SLA-based resource allocation for a software as a service provider in cloud computing environments.,»

[7] G. Cicotti, L. Coppolino, R. Cristaldi, S. D'Antonio et L. Romano, «QoS Monitoring in a Cloud Services Environment : the SRT-15 Approach,»

[8] R. Nathuji, A. Kansal et A. Ghaffarkhah, «Q-Clouds : Managing Performance Inter- ference Effects for QoS-Aware Clouds,»

[9] V. C. Emeakaroha, «Managing Cloud Service Provisioning and SLA Enforcement via Holistic Monitoring Techniques,»

[10] https://www.sciencedirect.com/science/article/pii/S1877705812023259

[11] https://www.cloudsimtutorials.online/cloudsim/