MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE
LA RECHERCHE SCIENTIFIQUE
UNIVERSITE ABDELHAMID IBN BADIS - MOSTAGANEM

Faculté des Sciences Exactes et de l'Informatique

Département de Mathématiques et d'Informatique

Filière : Informatique

MEMOIRE DE FIN D'ETUDES

Pour l'Obtention du Diplôme de Master en Informatique

Option : Ingénierie Du Système Informatique

THEME : IMPLEMENTATION ET EVALUATION D'UN POLITIQUE DE SELECTION BASEE MICRO-SERVICES DANS UN ENVIRONNEMENT CLOUD

Etudiant(e) :

Mimoun Douaa

Zoud Abdelkader

Président : MECHAOUI M D

Examinateur : DJEBBARA R

Encadrant (e): FILLALI F Z

Année Universitaire 2018/2019

Résumé

Le cloud computing, ou l'informatique en nuage, est l'exploitation de la puissance de calcul ou de stockage de serveurs informatiques distants par l'intermédiaire d'un réseau, généralement l'internet.

Les ressources gérées dans le Cloud se basent sur deux notions principe de base qui sont les machines virtuelles et les conteneurs. Une machine virtuelle permet de simuler la présence de ressources matérielles et logicielles et d'exécuter des programmes dans les mêmes conditions que celles de la machine simulée. Un conteneur quant à lui, représente un concept qui permet d'isoler l'exécution d'ensembles de processus sur un même hôte, ou bien un ensemble de processus visant à offrir des outils de gestion de ressources,

L'objectif de ce travail est de proposer une approche pour gérer l'ordonnancement de ressources dans les conteneurs d'un environnement de Cloud computing pour le placement des micro-services, à travers l'utilisation des algorithmes heuristiques afin d'avoir une solution approchée de notre problème de façon optimale.

Mots clés : Cloud Computing, CloudSim, Conteneur, Micro-service, Machine Virtuelle, centre de données, Hôte, Algorithme génétique, Sélection, Cluster, Ordonnancement, Simulateur.

Abstract

Cloud computing, is the exploitation of computing power or storage of remote servers via a network, usually the Internet.

The resources managed in the Cloud are based on two basic notions : virtual machines and containers. A virtual machine simulates the presence of hardware and software resources and executes programs under the same conditions as those of the simulated machine. A container as for it, represents a concept that isolates the execution of process sets on a single host, or a set of processes to provide tools for managing resources.

The objective of this work is to propose an approach to manage the scheduling of resources in the containers of a cloud computing environment for the placement of micro-services, by So using heuristic algorithms and trying to converge towards the best solution, donations using an algorithm that the genetic algorithm that gives an approximate solution to our problem.

The aim of this work is to propose an approach to manage the scheduling of resources in the containers of a cloud computing environment for the placement of micro-services, through the use of heuristic algorithms in order to have an optimal approached solution to our problem.

Keywords: Cloud Computing, CloudSim, Container, Micro-Service, Virtual Machine, Data Center, Host, Genetic Algorithm, Selection, Cluster, Scheduling, Simulator.

Remerciement

Avant tout je remercie Dieu le tout puissant de m'avoir accordé la foi, le courage, la santé et les moyens de conception de ce modeste travail.

Mes remerciements vont aussi à mon encadreur, **Madame Fillali Fatima Zohra,** pour son aide, sa disponibilité et ses précieux conseils qui m'ont permis de surmonter toutes les difficultés que j'ai rencontrées.

Je tiens à remercier, Monsieur le Professeur **Djebarra Reda**, pour avoir accepté d'examiner de travail.

Je remercie également toute l'équipe pédagogie de faculté de Science Exacte et Informatique et les intervenants professionnels.

Mes sentiments de reconnaissance et mes remerciements vont également à l'encontre de toutes les personnes qui ont contribué de prés ou de loin à la réalisation de ce travail.

Je remercie enfin mes parents et mes petits frères qui m'ont été d'un grand soutien moral et qui m'ont encouragé à réaliser ce travail.

Tables des matières

Liste des figures

Introduction générale

Le cloud computing, ou l'informatique en nuage, est l'exploitation de la puissance de calcul ou de stockage de serveurs informatiques distants par l'intermédiaire d'un réseau, généralement l'internet. Selon la définition du National Institute of Standards and Technology *(NIST)*, le cloud computing est l'accès via un réseau de télécommunications, à la demande et en libre-service, à des ressources informatiques partagées configurables [*1*]. Il s'agit donc d'une délocalisation de l'infrastructure informatique. Le Cloud peut être décomposé en trois différentes couches -de la moins à la plus visible- afin d'offrir divers services pour les utilisateurs finaux : Applicative (SaaS, Software as a Service), Plateforme (PaaS, Platform as a Service) et Infrastructure (IaaS, Infrastructure as a Service).

Quand on parle de déploiement dans le cloud, on retrouve plusieurs tendances technologiques telles que les machines virtuelles, les conteneurs…etc.

Une machine virtuelle, ou VM représente une virtualisation d'un matériel informatique créée par un logiciel d'émulation qui permet de simuler la présence de ressources matérielles et logicielles telles que la mémoire, le processeur, le disque dur, voire le système d'exploitation et les pilotes, ce qui permet d'exécuter des programmes dans les mêmes conditions que celles de la machine simulée.

Les micro-services sont un style d'architecture logicielle à partir duquel un ensemble complexe d'applications est décomposé en plusieurs processus indépendants et faiblement couplés, souvent spécialisés dans une seule tâche.

Un conteneur quant à lui, représente un concept qui permet d'isoler l'exécution d'ensembles de processus sur un même hôte, ou bien un ensemble de processus visant à offrir des outils de gestion de ressources tout en assurant leur isolement au niveau du noyau et empêchant l'interférence entre eux. Pour son exécution, un conteneur contient notamment une application libérée du système d'exploitation. De ce fait les conteneurs représentent un cadre idéal pour le déploiement des micro-services applicatifs.

L'objectif du présent travail est de proposer, d'implémenter et d'évaluer une nouvelle stratégie de sélection de conteneurs dans un environnement de Cloud computing pour le placement de micro-services.

Dans ce rapport, nous allons étudier les différents concepts et terminologies se rapportant à notre thème dans le chapitre 1. Pour le chapitre 2, nous allons analyser quelques stratégies existantes et discuter les avantages et inconvénients de chacune.

Le troisième chapitre présentera notre solution : la sélection des conteneurs en utilisant des algorithmes heuristique. L'idée principale des heuristiques est d'explorer l'espace des solutions en essayant de converger vers la meilleure solution, pour cela nous avons choisit d'implémenter un algorithme génétique pour obtenir une solution approchée de notre problème.

Enfin, le dernier chapitre décrit l'environnement d'implémentation et la structuration de notre application, avec quelques imprimes d'écrans (captures d'écrans) pour l'illustration du travail réalisé.

Le rapport, qui récapitule le travail effectué, se compose de quatre chapitres qui sont :

- Chapitre 1 : nous présenterons dans ce chapitre le cadre général du projet, concept de base et définitions.
- Chapitre 2 : ce chapitre décrit quelques stratégies d'ordonnancement existantes et une discussion sur ces stratégies.

- Chapitre 3 : nous y présentons la solution proposée du problème qui concerne la sélection des conteneurs.
- Chapitre 4 : ce chapitre expose les résultats finaux de notre application avec des captures d'écrans.

Chapitre 1 : Concepts de base et définitions

.1.1. Introduction

Dans ce chapitre, nous allons présenter l'informatique en nuage, ses différents types de services et de déploiement, ainsi que ces avantages et inconvénients. De plus, nous allons définir les notions de machines virtuelles, micro-services ainsi que les conteneurs et ses différentes notions.

.1.2. Cloud Computing

1.2.1. Définition du Cloud

Les progrès technologiques en matière d'infrastructure matérielle (puissance de calcul, de stockage, etc...) et de connexions réseau et technologies Internet ont permis l'émergence d'une nouvelle branche appelée l'informatique en nuage (Cloud Computing en anglais).

Le cloud computing est un concept qui consiste à déporter sur des serveurs distants des stockages et des traitements informatiques traditionnellement localisés sur des serveurs locaux ou sur le poste de l'utilisateur.

Même si les experts ne sont pas d'accords sur sa définition exacte, la plupart s'accordent à dire qu'elle inclue la notion de services disponibles à la demande, extensibles à volonté. En contradiction avec les systèmes actuels, les services sont virtuels et illimités et les détails des infrastructures physiques sur lesquels les applications reposent ne sont plus du ressort de l'utilisateur.

Selon le National Institute of Standards and Technology (NIST), « *Le cloud computing est un modèle qui permet un accès réseau à la demande et pratique à un pool partagé de ressources informatiques configurables (telles que réseaux, serveurs, stockage, applications et services) qui peuvent être provisionnées rapidement et distribuées avec un minimum de gestion ou d'interaction avec le fournisseur de services.* » [1]

1.2.2. Les types de cloud computing (modèles de déploiement)

Il existe quatre modèle de déploiement ou types de Cloud Computing :

1.2.2.1. Cloud publique

Le Cloud publique est basé sur un modèle standard de Cloud Computing, dans lequel un prestataire de service met les ressources, tels que les applications, ou le stockage, à la disposition du grand public via internet. Le Cloud public peut être gratuit ou fonctionner selon le principe du paiement à la consommation.

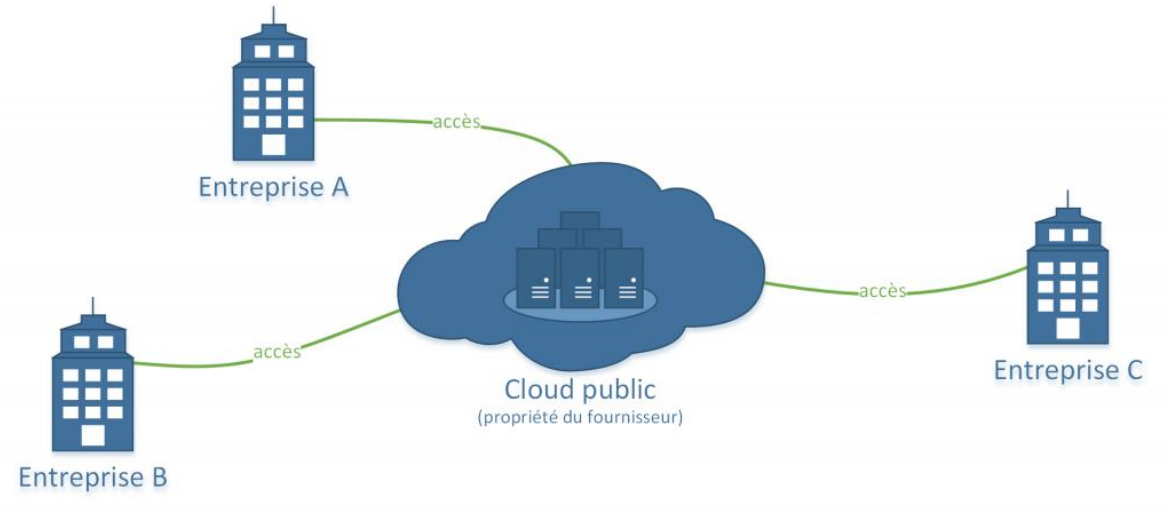


Figure 1 - Modèle de déploiement d'un cloud public [11]

1.2.2.2. Cloud privé

Le Cloud privé se base sur un environnement déployé au sein d'une entreprise. Ainsi, l'entreprise doit gérer toute seule son infrastructure. Dans ce cas, implémenter un Cloud privé signifie transformer l'infrastructure interne en utilisant des technologies telles que la virtualisation pour enfin délivrer, plus simplement et plus rapidement, des services à la demande.

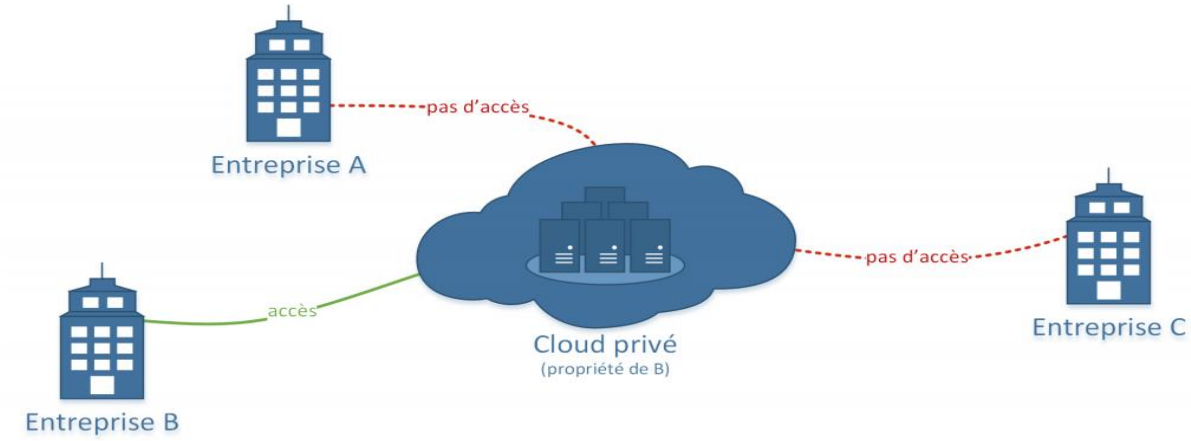


Figure 2 - Modèle de déploiement d'un cloud privé [11]

1.2.2.3. Cloud hybride

En général, on entend par Cloud hybride la cohabitation et la communication entre un Cloud privé et un Cloud publique dans une organisation partageant des données et des applications.

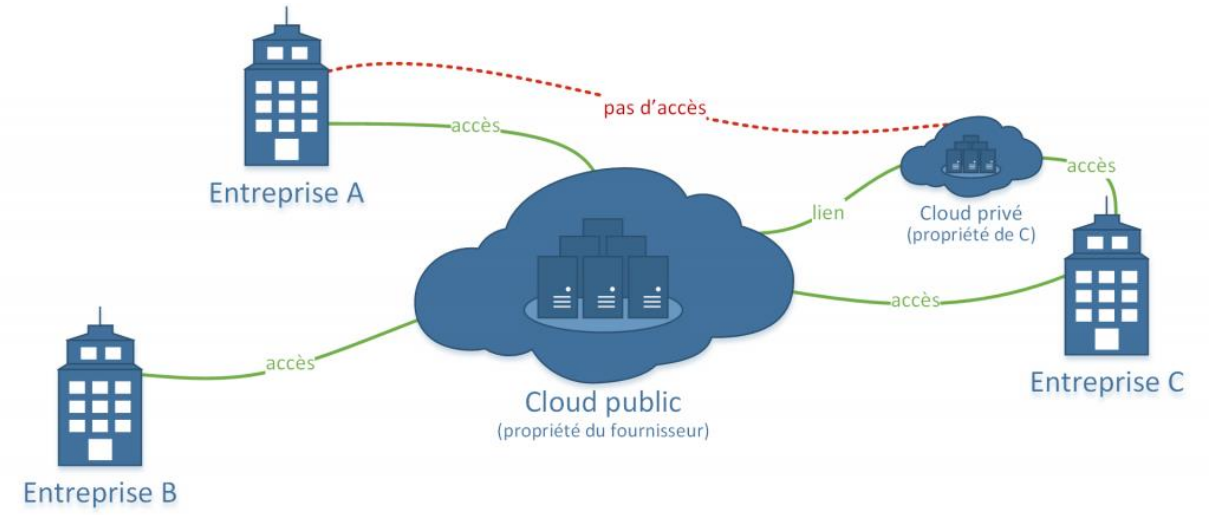


Figure 3 - Modèle de déploiement d'un cloud hybride [11]

1.2.2.4. Cloud communauté

L'infrastructure de Cloud est partagé par plusieurs organisations et soutient une communauté spécifique qui a des préoccupations communes, Il peut être géré par les organisations ou un tiers et peut exister sur site ou hors prémisse.

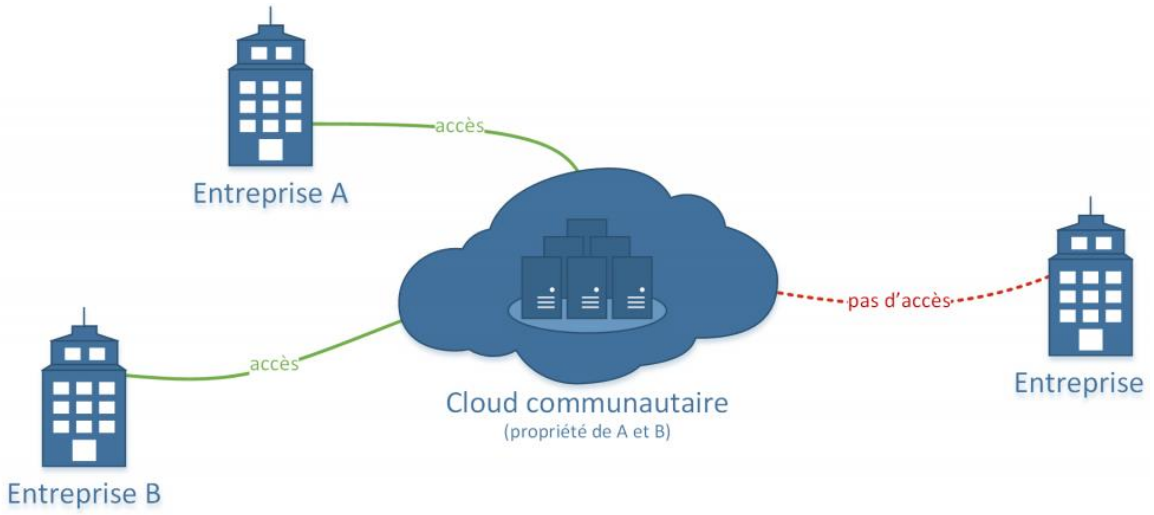


Figure 4 - Modèle de déploiement d'un cloud communautaire [11]

La figure suivante illustre les quatre modèles de déploiement, ainsi que leurs principales caractéristiques :

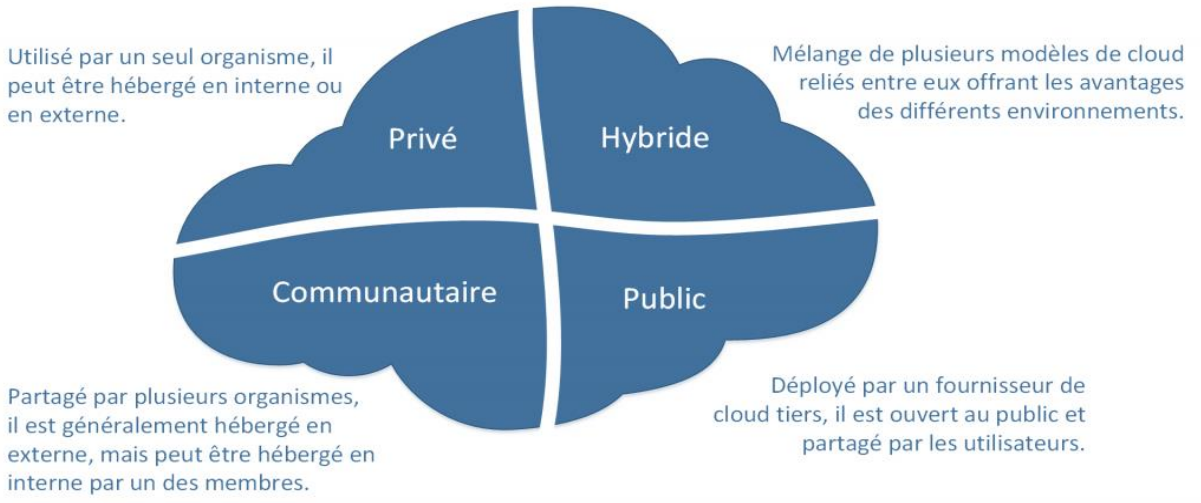


Figure 5 - Les modèles déploiement d'un cloud computing [11]

1.2.3. Architecture de Cloud Computing (modèles de service)

Généralement, on distingue trois types de service offerts par l'informatique en nuage qui sont :

1.2.3.1. Application en tant que service (SaaS -Software as a Service)

Le cloud de niveau SaaS représente le plus souvent un catalogue d'applications accessibles en mode service aux utilisateurs finaux ou clients finaux.

Dans le mode SaaS, les utilisateurs consomment les logiciels à la demande sans les commander, avec ou sans facturation à l'usage réel. Il n'est plus nécessaire pour l'utilisateur d'effectuer les installations, les mises à jour ou encore les migrations de données. Les solutions SaaS constituent la forme la plus répandue de Cloud Computing.

1.2.3.2. Plateforme en tant que service (PaaS - Platform as a Service)

Le cloud de niveau PaaS concerne les développeurs et les producteurs d'applications. Le PaaS permet de mettre à disposition des développeurs par exemple un Framework de développement adapté à leurs besoins. Il permet aussi de donner aux applications un cadre d'exécution qui produira des services SaaS.

Le modèle de livraison PaaS représente un environnement "prêt à l'emploi" prédéfini, généralement composé de ressources informatiques déjà déployées et configurées. Plus précisément, le PaaS s'appuie sur (et est principalement défini par) l'utilisation d'un environnement pré-établi qui établit un ensemble de produits et d'outils préemballés utilisés pour supporter l'intégralité du cycle de livraison des applications personnalisées.

1.2.3.3. Infrastructure en tant que service (IaaS - Infrastructure as a Service)

Le IaaS concerne essentiellement le cloud d'infrastructure. C'est une infrastructure virtuelle sur laquelle il est possible de bâtir une solution applicative.

Le modèle de livraison IaaS représente un environnement informatique autonome composé de ressources informatiques centrées sur l'infrastructure auxquelles on peut accéder et gérer via des interfaces et des outils basés sur le service cloud. Cet environnement peut inclure du matériel, du réseau, de la connectivité, des systèmes d'exploitation et d'autres ressources informatiques «brutes».

Contrairement aux environnements traditionnels d'hébergement ou de sous-traitance, avec IaaS, les ressources informatiques sont généralement virtualisées et regroupées en groupes qui simplifient la mise à l'échelle de l'exécution et la personnalisation de l'infrastructure.

1.2.4. Les avantages et inconvénients du Cloud Computing

1.2.4.1. Les avantages

Le Cloud Computing peut permettre d'effectuer des économies, notamment grâce à la mutualisation des services sur un grand nombre de clients. L'élasticité du nuage permet de fournir des services évolutifs et peut permettre de supporter des montées en charge.

L'abonnement à des services de Cloud Computing peut permettre à l'entreprise de ne plus avoir à acquérir des actifs informatiques et nécessitant une durée d'amortissement.

La maintenance, la sécurisation et les évolutions des services étant à la charge exclusive du prestataire, dont c'est généralement le cœur de métier, celles-ci ont tendance à être mieux réalisées et plus rapidement que sous la responsabilité du client.

1.2.4.2. Les inconvénients

L'utilisation des réseaux publics, dans le cas du Cloud public, entraîne des risques liés à la sécurité du Cloud.

Le client d'un service de Cloud Computing devient très dépendant de la qualité du réseau pour accéder à ce service. Aucun fournisseur de service Cloud ne peut garantir une disponibilité de 100 %.

Les entreprises n'ont plus de garanties (autres que contractuelles) de l'utilisation qui est faite de leurs données, puisqu'elles les confient à des tiers.

.1.3. Virtualisation

La technologie de virtualisation est l'une des composantes fondamentales du cloud computing, notamment en ce qui concerne les services basés sur l'infrastructure. La virtualisation permet de créer un environnement d'exécution sécurisé, personnalisable et isolé pour l'exécution d'applications, même si elles ne sont pas fiables, sans affecter les applications d'autres utilisateurs. La base de cette technologie est la capacité d'un programme informatique - ou d'une combinaison de logiciels et de matériel - à imiter un environnement d'exécution distinct de celui qui héberge ces programmes.

1.3.1. Définition de la virtualisation

La virtualisation est un mécanisme informatique qui consiste à faire fonctionner plusieurs systèmes, serveurs ou applications, sur un même serveur physique. La virtualisation est un composant technique clé dans le Cloud Computing.

La virtualisation permet d'ajouter une couche d'abstraction qui sépare le système d'exploitation du matériel afin de délivrer une meilleure utilisation et flexibilité des ressources de traitement.

1.3.2. Usage de la virtualisation

La virtualisation permet différents types d'applications :

- Installation de plusieurs systèmes d'exploitation sur un unique serveur.
- Test des applications sur plusieurs systèmes dans les phases de développement.
- Accélération de la montée en puissance du système d'information.

1.3.3. Machines virtuelles

La machine virtuelle est une imitation d'un appareil informatique créée par un logiciel d'émulation ou instanciée sur un hyperviseur. Elle représente un ordinateur complet contenant tout ce qui justifie cette imitation : un système d'exploitation(SE) complet, des pilotes, des systèmes de fichiers binaires ou bibliothèques ainsi que l'application elle-même.

Un des intérêts des machines virtuelles est de pouvoir s'abstraire des caractéristiques de la machine physique utilisée (matérielles et logicielles — notamment système d'exploitation), permettant une forte portabilité des logiciels et la gestion de systèmes hérités étant parfois conçus pour des machines ou des environnements logiciels anciens et plus disponibles.

Les machines virtuelles sont également utilisées pour isoler des applications pour des raisons de sécurité, pour augmenter la robustesse d'un serveur en limitant l'impact des erreurs système ou pour émuler plusieurs machines sur une seule machine physique (virtualisation).

Les inconvénients des machines virtuelles sont d'une part des performances brutes sensiblement inférieures à une exécution sur le matériel en natif, d'autre part de consommer une petite partie de la mémoire réelle pour leur propre fonctionnement. Leurs avantages sont de permettre des tests de variantes d'installation sur des machines simulées possédant de tailles de RAM, de mémoire graphique, et des nombres de processeurs très divers.

1.3.4. Hyperviseur

L'hyperviseur est le logiciel qui contrôle l'utilisation, par les différentes machines virtuelles, du matériel de l'appareil simulateur.

Il existe deux types d'hyperviseur :

- Hyperviseur de type 1 (natif) : il n'a pas besoin de système d'exploitation sur la matériel initial. Il s'exécute directement sur une plateforme matérielle ; cette plateforme est alors considérée comme outil de contrôle de système d'exploitation.
- Hyperviseur de type 2 (hosted) : il s'exécute à l'intérieur d'un autre système d'exploitation. Il utilise ainsi, le système d'exploitation de l'ordinateur simulateur.

.1.4. Conteneurs

1.4.1. Définition du conteneur

On peut définir un conteneur comme un processus ou un ensemble de processus isolés du reste du système. Tous les fichiers nécessaires à leur exécution sont fournis par une image distincte, ce qui signifie que le conteneur comprend une image plus les fichiers qui composent l'application qui va être exécutée à l'intérieur.

Son principe de base est l'empaquetage qui rend le déploiement des applications plus facile et portable. Ainsi, ils sont bien plus rapides. De plus, il consomme une quantité limitée de ressources afin de satisfaire les demandes des clients.

Parmi les exemples de conteneurs qu'on peut citer : le conteneur Linux LXC, Docker, …

1.4.2. Micro-services

Un micro-service est un modèle d'architecture logicielle à partir duquel un ensemble complexe d'applications est décomposé en plusieurs processus indépendants et faiblement couplés, souvent spécialisés dans une seule tâche. Les processus indépendants communiquent les uns avec les autres en utilisant des API.

Le principe des micro-services s'inspire en grande partie de la philosophie UNIX, qui consiste à « *ne faire qu'une seule chose, et la faire bien* ». Elle est décrite comme suit :

- Les services sont petits, et conçus pour remplir une seule fonction.
- L'organisation du projet doit prendre en compte l'automatisation, le déploiement et les tests.
- Chaque service est élastique, résilient, composable, minimal et complet.

Actuellement, une grande majorité des développeurs découvrent que les micro-services influencent positivement des aspects tels que le temps, la performance ou la stabilité du projet.

Les conteneurs, constituent un mécanisme de déploiement idéal pour les micro-services. Les conteneurs sont essentiellement des machines virtuelles légères pouvant être mis à disposition uniquement avec l'infrastructure et les outils de gestion nécessaires au service. Ils sont des plateformes idéales pour déployer des micro-services.

1.4.3. Différences entre une machine virtuelle et un conteneur

Les principales différences entre une machine virtuelle et un conteneur sont :

Machine Virtuelle

- Une VM s'exécute dans un environnement isolé de virtualisation matérielle grâce à un hyperviseur.
- Tous les éléments exécutés dans une VM sont masqués du système d'exploitation hôte.
- Une VM apparaît comme un ordinateur physique autonome.
- Très lourde lors de démarrage.

Conteneur

- Ne nécessite pas d'hyperviseur pour assurer son isolation.
- Il utilise les fonctionnalités d'isolation des processus et du système de fichiers du noyau Linux pour exposer sur le conteneur certaines fonctionnalités d'un noyau et son propre système de fichiers isolés.
- Le conteneur paraît comme une instance unique du système d'exploitation.
- L'espace disque nécessaire du conteneur ne contient pas un SE dans son intégralité ainsi le temps de démarrage du conteneur et la surcharge d'espace disque requis sont réduits.

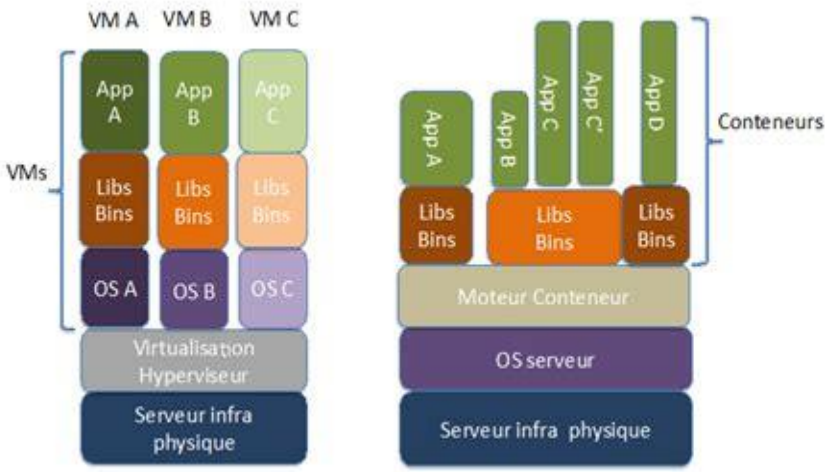


Figure 6 - Comparaison d'architecture entre ces VM et conteneur [8]

1.4.4. Technologies du conteneur

Selon les divers conteneurs qui existent telles que Docker, LXC, OpenVZ, Linux-Vserver, rkt…, on retrouve de nombreuses technologies qui réalisent le concept des conteneurs telles que :

- La fonction du noyau **cGroups** (groupes de contrôle) permet de contrôler et de limiter l'utilisation des ressources pour un processus ou un groupe de processus.
- Le système d'initialisation **systemd** permet de définir l'espace utilisateur et de gérer les processus associés.
- Les **espaces de noms utilisateur** « permettent de mettre en correspondance des identifiants d'utilisateurs et de groupes au sein d'un espace de noms. Dans le contexte des conteneurs, cela signifie que des utilisateurs et des groupes peuvent avoir des privilèges pour effectuer certaines opérations au sein du conteneur, mais que ces privilèges ne leur sont pas accordés en dehors du conteneur ».
- **LXC**, permet de démarrer des conteneurs via une simple interface en ligne de commande.

1.4.5. Architecture de référence pour système d'orchestration du conteneur [15]

Les systèmes d'orchestration de conteneur permettent le déploiement d'applications conteneurisées sur un système distribué tel que le Cloud Computing. Ils permettent leur exécution et leur surveillance en gérant de manière transparente les tâches et les données déployées dans un ensemble de ressources distribuées.

Quatre entités ou couches principales sont identifiées dans l'architecture présentée, à savoir un ou plusieurs travaux, un maître de cluster, un cluster de calcul et l'infrastructure physique.

- **Les emplois (travaux).** Les travaux appartiennent généralement à des utilisateurs différents et sont hétérogènes. Ils peuvent aller de services sensibles au temps de latence de longue durée à des travaux par lots gourmands en ressources.
- **Maître de cluster.** Le maître de cluster est le noyau du système. Son composant principal est le moniteur qui est responsable de la gestion des mesures de consommation de ressources en temps réel pour chaque nœud de travail du cluster.

- **Cluster de calcul.** Chaque ordinateur du cluster disponible pour le déploiement de tâches est un nœud de travail. Chacun de ces nœuds a un agent de travail avec différentes responsabilités.
- **Infrastructure.** L'un des principaux avantages des conteneurs est qu'ils peuvent être déployés sur une multitude de plates-formes. Les machines en cluster peuvent être des machines virtuelles sur des infrastructures de cloud public ou privé, des machines physiques sur un cluster ou même des périphériques mobiles ou périphériques, entre autres.

1.4.6. La sécurité des conteneurs

La sécurité des conteneurs est la protection de l'intégrité des conteneurs. Cela inclut, les applications qu'ils détiennent et l'infrastructure sur laquelle ils s'appuient. La sécurité des conteneurs doit être intégrée et continue.

En général, la sécurité des conteneurs concerne:

- Sécurisation du pipeline de conteneurs et de l'application.
- Sécurisation de l'environnement et des environnements de déploiement de conteneurs.
- Intégration aux outils de sécurité de l'entreprise et respect ou amélioration des stratégies de sécurité existantes.

.1.5. Conclusion

Ce chapitre a permis d'aborder et de définir les différents concepts et terminologies de base pour la compréhension du cadre de travail général. Nous avons fourni une base théorique sur le Cloud Computing, les conteneurs, les machines virtuelles et les micro-services. Dans le chapitre suivant, nous présenterons l'état de l'art et les différentes stratégies d'ordonnancement des conteneurs déjà existantes.

Chapitre 2 : Ordonnancement et leurs stratégies

.2.1. Introduction

L'utilisation de machines virtuelles et de conteneurs par plusieurs utilisateurs s'est très rapidement révélé nécessaire pour des raisons d'économie, de rentabilité et de convivialité. Sous cette hypothèse, le problème qui se pose alors, à chaque instant, pour chaque machine virtuelle (ou conteneur), est de décider s'il doit poursuivre ou interrompre l'exécution du (micro-) service courant, et, dans le second cas, de déterminer le prochain (micro-) service à activer.

La règle utilisée pour effectuer ce choix est contenue dans l'algorithme d'ordonnancement, plus couramment appelé ***ordonnanceur*** (***scheduler*** en anglais). Nous allons présenter dans ce chapitre les diverses stratégies d'ordonnancement des machines virtuelles et des conteneurs.

Il faut considérer l'ordonnanceur sous trois aspects :

– Au niveau le plus élevé, le plus proche de l'utilisateur, sa fonction consiste à déterminer si un travail soumis doit être admis (tout travail admis devient un service ou micro-service) ou pas. Ainsi, l'évaluation des priorités, la gestion des ressources nécessaires (règlement de conflits), le maintien de la charge du système en dessous du seuil d'écroulement et de son intégrité, en particulier en cas d'incidents imprévisibles, sont autant de tâches dont l'ordonnanceur devra s'acquitter avec inévitablement des répercussions sur les travaux soumis.

– Le rôle du niveau moyen est de déterminer l'ensemble des services (ou micro-services) pouvant obtenir le contrôle. Autrement dit, il tient à jour les paramètres relatifs aux différents services qui permettront de dégager ceux étant susceptibles de devenir actifs.

– Le niveau le plus bas, le plus proche du matériel, choisit parmi les services prêts, en respectant les priorités, celui à qui le service va être alloué. C'est le répartiteur qui est toujours résident en mémoire.

.2.2. Description du problème

Un environnement de Cloud computing permet à plusieurs utilisateurs d'exécuter différentes machines virtuelles ou conteneurs simultanément. Chaque exécution nécessite une certaine quantité de ressources. Il est alors nécessaire de coordonner le partitionnement des ressources avec les besoins des différentes VMs et conteneurs. Un ordonnanceur a pour objectif de gérer de planifier l'exécution des VMs et conteneurs dans le temps. Il est donc chargé d'allouer à chaque VM (ou conteneur) une partition de ressources permettant de l'exécuter dans des conditions suffisantes et si possible en réduisant au maximum le temps entre sa soumission et son exécution. Le partitionnement des ressources doit tenir compte du niveau de flexibilité des VMs et des possibilités de partitionnement supportées par l'environnement.

Le problème d'ordonnancement des tâches étant un problème relativement ancien, c'est l'action d'assigner des ressources au traitement des tâches. Beaucoup d'approches ont été proposées pour le résoudre dans différentes plates-formes.

Avec l'apparition du Cloud computing, plusieurs solutions d'ordonnancement ont été adaptés pour ce type de plate-forme. La plupart des algorithmes d'ordonnancement dans les Clouds vise à remplir un ou plusieurs objectifs. Certains objectifs touchent à la rapidité de traitement des tâches et dans les délais.

D'autres sont liés à la répartition équitable des ressources entre les tâches. D'autres encore sont relatifs à l'utilisation optimale des ressources, par exemple l'équilibrage de la charge de travail entre les

ressources, ou au contraire sa consolidation sur un nombre restreint de ressources afin de maximiser l'utilisation de ces dernières. D'autres encore visent à respecter des contraintes de placement, qui peuvent par exemple découler des affinités ou des antagonismes entre les tâches. Enfin, certains objectifs peuvent consister à respecter d'autres types de contraintes, comme les contraintes d'antériorité entre les tâches.

2.3. Ordonnancement

2.3.1. Objectifs

Une politique d'ordonnancement doit :

1. être équitable : cette contrainte est satisfaite si tous les services sont considérés de la même manière et qu'aucun n'est retardé indéfiniment ;

2. rendre le débit maximum : elle doit faire en sorte de satisfaire le plus grand nombre de demandes par unité de temps ;

3. pouvoir prendre en charge un maximum d'utilisateurs interactifs tout en assurant des temps de réponse acceptables ;

4. être prédictible : un même service doit pouvoir s'exécuter dans un temps à peu près équivalent quelle que soit la charge du système ;

5. être la moins coûteuse possible afin de ne pas éprouver les performances générales du système en particulier dans les phases instables ;

6. avoir pour effet de rationaliser la gestion des ressources en :

- recherchant une utilisation optimum ;
- favorisant les tâches peu exigeantes en nombre et en qualité de ressources ;
- évitant la famine (par exemple en augmentant la priorité au fur et à mesure que l'attente s'accroît).

7. mettre en œuvre des priorités fondées sur des critères pertinents ;

8. avoir la possibilité de réajuster ces priorités, soit de manière globale (nécessité de modularité), soit de manière ponctuelle au cours du temps (priorités dynamiques) ;

9. favoriser les processus ayant un comportement souhaitable ;

10. veiller à ne pas accepter de nouveaux travaux lorsque le système est en surcharge.

2.3.2. Étapes d'un processus d'ordonnancement

Le processus d'ordonnancement se composent des étapes suivantes :

1) La phase de sélection : établit l'ordre des tâches de départ leurs propriétés et leurs contraintes. Après cette phase, on a une liste ordonnée.

2) La phase d'allocation : réserve ou alloue un ensemble de ressources, c'est-à-dire qu'elle calcule le nombre de machines virtuelles pour l'ordonnancement des tâches.

3) La phase de placement: sélectionne les ressources parmi celles précédemment alloué qui permettent d'exécuter les tâches selon l'ordre prédéfini. Elle fait le placement des tâches de chaque machine virtuelles avec les ressources qui lui sont optimales.

2.3.3. Stratégies d'ordonnancement

Un ordonnanceur a plusieurs objectifs comme mentionnés précédemment. Ces objectifs consiste en l'utilisation efficace des ressources, le respect des contraintes de positionnement fournies par l'utilisateur, la planification rapide des applications pour ne pas les laisser en attente, l'équité, la résistance aux erreurs et la disponibilité. Pour atteindre ces objectifs, trois principales architectures d'ordonnanceurs ont été proposées. ***[122]***

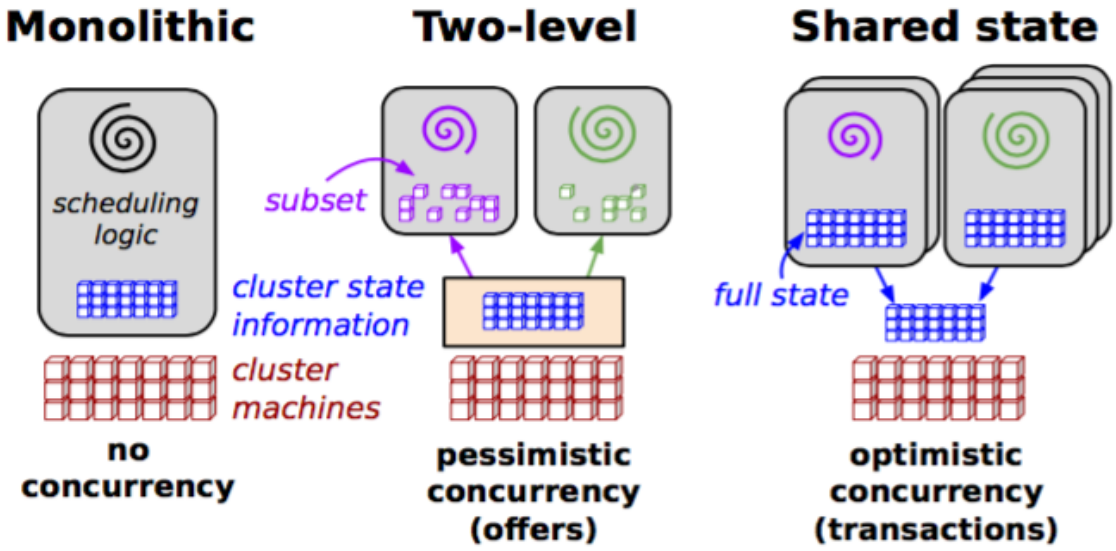


Figure 7 - Vue d'ensemble schématique des architectures ordonnancement [8]

Les architectures des ordonnanceurs sont généralement divisé en trois types :

- **L'ordonnancement monolithique (Monolithic scheduling) :** Les ordonnanceurs monolithiques sont composés d'un seul agent d'ordonnancement qui gère toutes les demandes. Ils sont couramment utilisés dans le calcul haute performance. Un ordonnanceur monolithique applique généralement une implémentation à un seul algorithme pour tous les travaux entrants. Il est donc difficile d'exécuter une logique d'ordonnancement différente en fonction du type de travail.
- **L'ordonnancement à deux niveaux (Two-level scheduling) :** Un ordonnanceur à deux niveaux ajustel'allocation des ressources à chaque ordonnanceur de manière dynamique en utilisant un coordinateur central pour décider du nombre de ressources que chaque sous-cluster peut avoir.
- **L'ordonnancement à état partagé (Shared-state scheduling) :** Un ordonnancement à état partagé accorde à chaque agent d'ordonnancement un accès complet à l'ensemble des ressources. Il n'y a pas d'allocateur de ressources central car toutes les décisions d'allocation de ressources ont lieu dans les ordonnanceurs. Il n'y a pas de moteur central d'application des règles, des ordonnanceurs individuels prennent des décisions.

.2.4. Ordonnancement des Machines virtuelles (VM Scheduling)

2.4.1. Description de l'ordonnancement de machines virtuelles

L'ordonnancement des machines virtuelles dans le Cloud Computing est utilisé pour planifier les requêtes de la VM au serveur physique, conformément aux exigences remplies avec les ressources demandées (c'est à dire mémoire, stockage, puissance du processeur, bande passante, etc.).

Actuellement, il y a tellement de fournisseurs de cloud sur le marché qui ont des capacités de centres de données et machines physiques différentes, tels que : SalesForce, Amazon, Microsoft Office 365, Windows Azure, Oracle Cloud, Google Apps, …

En général, l'algorithme d'ordonnancement de machines virtuelles fonctionne dans trois niveaux comme indiqué ci-dessous :

- Pour le groupe de machines virtuelles, rechercher le répertoire physique approprié.
- Déterminer le schéma d'approvisionnement approprié pour le machine virtuelle.
- Planifier les tâches sur l'ordinateur virtuel.

2.4.2. Les stratégies et les algorithmes d'ordonnancement

Le problème d'ordonnancement des taches étant un problème relativement ancien, c'est l'action d'assigner des ressources au traitement des taches. Beaucoup d'approches ont été proposées pour le résoudre dans différents plate-forme. Il existe de nombreux algorithmes d'ordonnancement, allant de schémas statiques [17] très simple aux algorithmes dynamiques [18] les plus complexes.

La plupart des algorithmes d'ordonnancement dans les Clouds vise à remplir un ou plusieurs objectifs. Certains objectif touchent à :

- La rapidité de traitement des taches et dans le délai [19,20].
- Répartition équitable et l'utilisation optimale des ressources [21].
- Respecter des contraintes de traitement.

En dernier, ces algorithmes sont focalisés sur :

- La rapidité d'exécution.
- L'optimisation de consommation d'énergie électrique dans divers facteurs tel que :
  - Le coût de l'énergie.
  - Le taux d'émission de CO2.
  - La charge de travail.
  - La puissance du processeur.

L'approche [23] à proposé deux algorithmes d'ordonnancement des taches, en tenant compte la complexité de calcul et la capacité des nœuds de calcul :

- Le premier algorithme est nommé Longest Cloudlet Fastest Processing (LCFP) ou les taches sont d'abord triées par coût décroissant en fonction du temps de traitement le plus rapide de la plus longue Cloudlet.
- Le second algorithme est nommé Shortest Cloudlet Processing Element (SCEPE) ou les taches sont d'abord triées par coût croissant en fonction du temps de traitement de la plus courte Cloudlet.

Les auteurs [25] ont proposés une stratégie d'ordonnancement sur les Clouds, basé sur la priorité. Deux algorithmes ont été proposés dans le cadre de ce travail :

- Le premier algorithme se base sur la date de fin des taches Earliest Deadline First (EDF) pour un système moins chargé.

- Le second est nommé V-Dover destiné pour un système surchargé.

.2.5. Ordonnancement des conteneurs (containers scheduling)

2.5.1. Docker Swarm

Docker Swarm est un ordonnanceur de conteneurs développé par Docker. Le développement de cette solution de cluster gérée par Docker offre des avantages tels que l'utilisation de l'API standard de Docker.

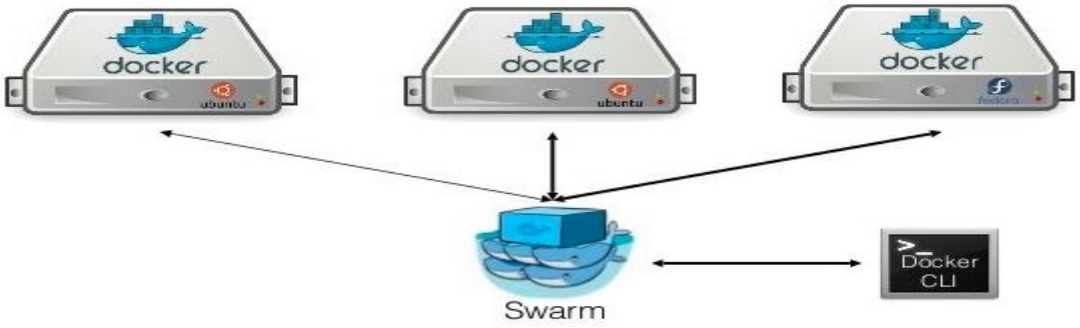


Figure 8 - Architecture de Docker Swarm [8]

Swarm utilise l'une de ces trois stratégies pour faire le classement des hôtes :

1) Spread :choisit le nœud qui a moins de conteneurs par rapport aux autres sans tenir compte de leurs états. C'est la stratégie utilisée par défaut dans écifiée au démarrage du Swarm, cette stratégie sera la stratégie adoptée par défaut.

2) Binpack : choisit le nœud le plus chargé. C'est à dire celui qui a le moins de ressources disponibles et qui contient le plus grand nombre de conteneurs.

3) Aléatoire : choisit aléatoirement un nœud.

2.5.2. Google Kubernetes

Kubernetes est un système d'orchestration pour les conteneurs Docker utilisant les concepts de «libellés» et de «pods» pour regrouper les conteneurs en unités logiques. Les pods sont des collections de conteneurs colocalisés formant un service déployé et planifié ensemble. Cette approche simplifie la gestion du cluster par rapport à un co-ordonnancement de conteneurs basé sur des affinités (comme Swarm et Mesos).

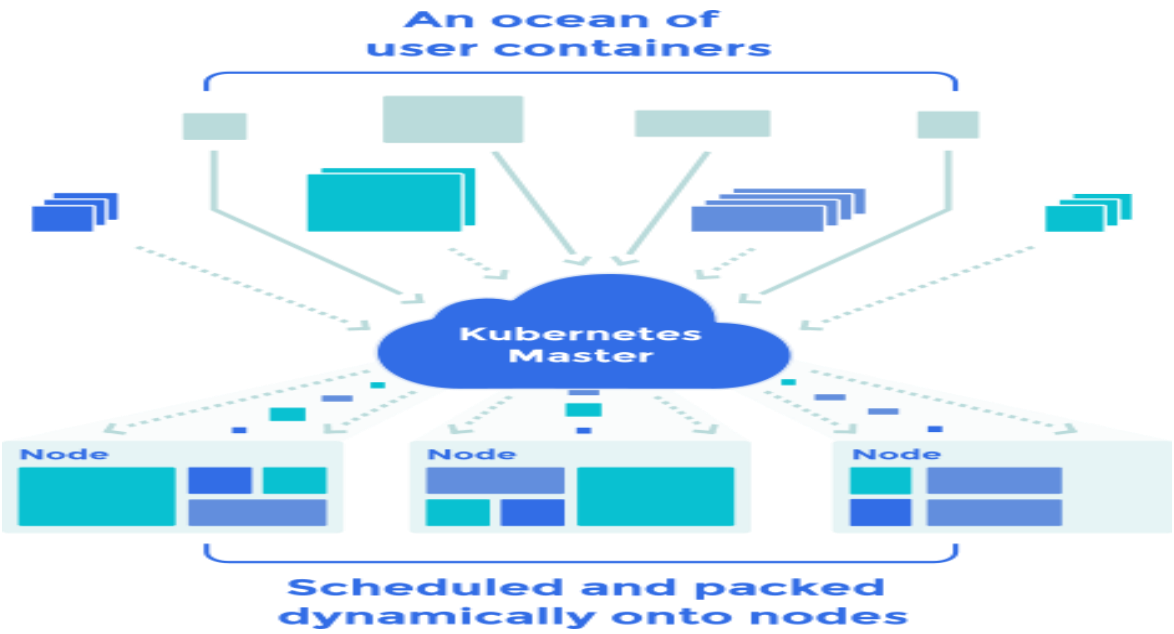


Figure 9 - Architecture de Kubernetes [8]

Parmi les stratégies de priorités utilisées par Kubernetes, on retrouve :

- **LeastRequestedPriority** : le nœud est prioritaire en fonction de la fraction du nœud qui serait libre si le nouveau Pod était planifié sur le nœud. Le CPU et la mémoire sont également pondérés. Le nœud avec la fraction libre la plus élevée est le plus préféré. Cette fonction de priorité a pour effet de répartir les pods entre les nœuds en ce qui concerne la consommation de ressources.
- **BalancedResourceAllocation** : cette fonction de priorité essaie de mettre le pod sur un nœud de telle sorte que le taux d'utilisation de la CPU et de la mémoire est équilibré après le déploiement du pod.
- **ServiceSepreadingPriority :** elle favorise les nœuds qui sont utilisés par différents pods.
- **EqualPriority** : cette fonction est utilisée uniquement pour les tests. Elle donne une priorité égale à tous les nœuds du cluster.

2.5.3. Autres stratégies

L'ordonnancement des conteneurs, équilibrage de la charge, optimisation énergétique et les migrations sont les principaux défis auxquels sont confrontés les prestataires de service dans le cadre des conteneurs. Actuellement les défis sont abordés comme suit :

- Zhang[26] a proposé une stratégie d'ordonnancement des conteneurs en utilisant la programmation linéaire pour minimiser le coût total. Cet algorithme de planification est exploité à l'aide de trois paramètres à savoir :
  - ✓ Coût de transmission du réseau entre client et hôtes.
  - ✓ Utilisation de l'hôte en termes de coût de consommation.
  - ✓ Coût de téléchargement des images de conteneur.
- Kaewkasi [27] a utilisé une technique d'optimisation de méta-heuristique basée sur les colonies de fourmis «ACO» pour implémenter un algorithme d'ordonnancement des conteneurs qui amplifie les performances des applications.

- Les auteurs de [28] ont discuté du défi de l'équilibrage de la charge défi. Ils ont proposé une technique de d'ordonnancement proactive. Ils ont utilisé des traces de cluster Google pour simuler différentes comportements des conteneurs. Les données du conteneur sont regroupées dans ceux de courte et de longue durée. L'équilibrage de la charge se fait en sélectionnant les hôtes les plus et les moins utilisés pour les échanges de conteneurs. Cet échange conduit à une répartition uniforme de la charge sur le centre de données.
- Piraghaj [29] ont abordé le défi de l'optimisation énergétique. Ils ont décrit l'utilisation de l'hôte comme étant soit normal, sous-utilisé ou sur-utilisé selon l'utilisation du processeur. Dans ce travail, un conteneur dans un hôte surexploité est migré pour éviter les violations de SLA. D'autre part, un conteneur dans un hôte sous-utilisé est migré pour optimiser énergie.

2.5.4. Analyse et discussions

Swarm crée une image comme un seul hôte ; c'est à dire qu'elle ne vivra que sur un seul nœud et elle ne sera pas distribuée sur les autres nœuds du cluster. De plus, il ne conserve pas une trace du nombre d'instances d'un service. Si un conteneur échoue, il ne crée pas un nouveau.

Kubernetes est le meilleur ordonnanceur pour les micro-services, puisqu'il possède une architecture plus complexe que Swarm. Il offre la possibilité de faire des mises à jour aux conteneurs.

.2.6. Conclusion

Comme nous l'avons vu, il existe quelques systèmes d'ordonnancement de conteneurs à l'heure actuelle, mais ils ne font que fournir des stratégies d'ordonnancement simples.

Plus le le nombre de conteneurs augmente, plus les dépendances sur ces conteneurs deviennent de plus en plus complexes, les restrictions de ces politiques d'ordonnancement sont plus évidentes.

De plus, la charge de travail du système change avec le temps. Ainsi, on devrait veiller à ce que la charge de travail ne dépasse pas la limite de la machine physique, nous ne pouvons réserver qu'une partie du ressources à l'avance pour faire face à la montée des charges de travail du système avec l'avenir, ce qui réduira considérablement l'utilisation des ressources.

Chapitre 3 : Conception et développement

.3.1. Introduction

Dans ce chapitre, nous allons aborder les étapes de conception et développement de la solution proposée. Nous présentons notre contribution relative à l'optimisation de la sélection des conteneurs en utilisant un algorithme approchées qui est l'algorithme génétique.

.3.2. La motivation et problématique

L'objectif de notre travail est de proposer une nouvelle stratégie de sélection de conteneurs dans un environnement de Cloud Computing.

Le sélection de conteneurs consiste à choisir le conteneur le plus approprié selon certaines mesures. Ce type d'algorithmes représente un problème un peu délicat en raison de la nature de l'environnement du Cloud Computing qui est hétérogène. De plus, avoir une bonne organisation du Data Center des conteneurs demande en effort particulier.

Afin de pouvoir efficacement cette Data Center, il est nécessaire de connaitre l'emplacement de chaque conteneur.

Nous nous intéressons dans ce qui suit à l'exploitation des algorithmes de la sélection optimale. En ce sens, nous allons proposés un algorithme de sélection des conteneurs de telle sorte qu'on trouve un emplacement optimal pour chaque service de coneneur.

.3.3. Solution proposée

Pour résoudre ce problème, une grande variété des techniques ont été proposés (voir chapitre2). En totalité, dans les lectures que nous avons effectuées il y avait quatre grands axes de solutions qui sont : algorithme déterministe, algorithme d'optimisation, heuristique et méta heuristique.

Pour résoudre notre problème, nous allons utiliser des algorithmes heuristiques afin de trouver une solution optimale. L'idée principale des heuristiques est d'explorer l'espace des solutions en essayant de converger vers la meilleure solution.

Ainsi, nous avons choisie d'implémenter l'algorithme génétique comme une solution de notre problème.

3.3.1. Pourquoi choisir les algorithmes génétiques ?

Nous avons choisie d'implémenter l'algorithme génétique pour les raisons suivants :

- Obtention d'une solution approchée, en un temps correct (raisonnable).
- Offre une grande liberté dans le paramétrage et dans l'implémentation du différent traitement.
- Permet de simuler l'environnement, sa population et son évolution.

3.3.2. Les critères de développement de l'algorithme génétique

Pour développer notre solution basée sur l'algorithme génétique, on va supposer que les critères suivants sont satisfaits :

- Le type de conteneur : chaque conteneur à un type, les conteneurs de même type sont placés dans le même micro-service.
- Un conteneur occupe un et un seul emplacement.

3.3.3. Présentation de l'algorithme génétique

L'algorithme génétique est une méta-heuristique qui manipule une population de solution à la fois. C'est un algorithme de recherche basé sur le mécanisme de la sélection naturelle et de la génétique. Il combine une stratégie de « survie des plus forts » avec un échange d'information aléatoire mais structuré.

Le mode de fonctionnement d'un algorithme génétique est claqué sur les principes biologiques de la sélection naturelle et de la survie des individus les mieux adaptés à l'environnement.

La sélection naturelle est basée sur l'idée que les modifications des générations successives sont orientées par les pressions extérieures aux quelles sont soumises les espaces, exemple : la limitation des ressources, les modifications de l'environnement, les prédateurs et parasites.

Il en résulte que les individus les mieux adaptés à l'environnement tendent à survivre plus longtemps et à se reproduire plus fréquemment. S'inspirant ainsi de ce mécanisme, Holland a posé les bases des techniques d'optimisation appelée **« Algorithme Génétique ».**

Mais Goldberg qui c'est investi dans l'étude d'AG a développé la forme actuelle que nous connaissons. Vocabulaire employé par les AG est directement calqué sur celui de la théorie de l'évolution et de la génétique. Ainsi les termes comme : individu, population, gène, chromosome, croisement et mutation sont utilisés.

Dans le vocabulaire des AG l'environnement se rapporte à l'espace de recherche qui définit l'ensemble de la configuration possible des paramètres de la fonction à optimiser.

- Un individu dans cet environnement est une configuration possible des paramètres. Un ensemble d'individu forme une population. Chaque individu peut être représenté par un chromosome qui est composé d'une chaine de gènes contenant les caractéristiques génétiques de cet individu.
- Le gène étant la partie élémentaire d'un chromosome représente un trait de caractère ou une fonction particulière. La capacité d'adaptation d'un individu à l'environnement est matérialisée par la mesure de la performance de l'individu à travers la fonction à optimiser**.**

3.3.4. Le fonctionnement de la solution :

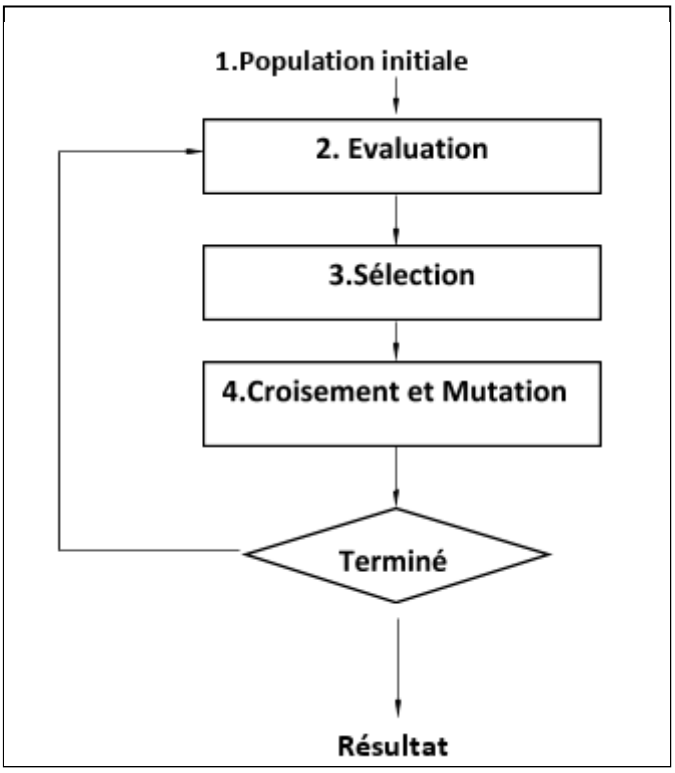


Figure 10 - Les étapes de l'algorithme génétique.

3.3.4.1. Algorithme général

1. ***Création des Data Center.***
2. ***Importation des hôtes.***
3. ***Importation des machines virtuelles.***
4. ***Création des conteneurs.***
5. ***Création des micro-services.***
6. ***Affectation des micro-services aux conteneurs aléatoirement.***
7. ***Estimation de meilleures sélections avec algorithme génétique.***
8. ***Affichage de résultat.***

3.3.4.2. Algorithme génétique

Les algorithmes génétiques ont la particularité de s'inspirer de l'évolution des espèces dans leur cadre naturel. Les espèces s'adaptent à leur cadre de vie qui peut évoluer, les individus de chaque espèce se reproduisent créant aussi de nouveaux individus, certains subissent des modifications, certains disparaissent …… etc.

Un AG va reproduire ce modèle dévolution dans le but de trouver des solutions pour un problème donnée.

Nous identifions dans ce qui suit, les correspondances entre les paramètres des AG et le cadre d'étude relatif à la sélection des conteneurs.

- Une population sera un ensemble d'individu.

- Un individu sera une solution à un problème donnée : des sélections des conteneurs dans les micro-services.
- Un gène sera une partie d'une solution, donc un individu : une sélection d'un conteneur dans un micro-service.
- Une génération est une itération de l'algorithme génétique.

```
Générer Population
{
    pour i de 0 a nb_essai_max {
        pour tous les individus {
        Evaluer (individu)}
        Si   meilleur_individu == resultat_attend{
        Arrêt
        }
        Sélectionner(Population)
        Croiser(Population)
        Muter(Population)
    }
}
```

- Solution = meilleur_individu
- Il est possible de ne pas exiger l'égalité et de se contenter d'une valeur approchée

3.3.4.3. Adaptation de l'algorithme génétique

L'adaptation de l'algorithme génétique pour notre problématique est comme suit :

- **Chromosome :** une sélection d'un conteneur pour un micro-service.
- **Individu :** représente l'ensemble des sélections de tous les conteneurs.
- **Population :** l'ensemble des solutions de sélection potentielles.

3.3.5. Description de l'Algorithme Génétique

L'algorithme génétique est composé de six étapes :

- **Étape 1 : Génération de la population initiale :**

Cette étape consiste à générer la population initiale de l'algorithme génétique. La population initiale doit contenir des chromosomes qui sont repartis dans l'espace des solutions par fournir à l'algorithme génétique un matériel génétique varié. La façon la plus simple est de générer aléatoirement les chromosomes, alors affecté aléatoirement des micro-services au conteneur.

- **Étape 2 :Évaluation des individus :**

Une fois que la population initiale a été crée, nous allons identifier les individus les plus prometteurs, ceux qui vont participer à l'amélioration de notre population. La méthode d'évaluation des individus est laissée au programmeur en fonction du problème qu'il a à optimiser ou à résoudre. On a utilisé la fonction de fitness qui illustrée dans la figure suivante :

```
//-----------------------------------------------------------------------
    public static float GetFitness(int ind, int nbMS)
    {
        float fit = 0.f;
        for(int j=0;j<nbMS;j++)
        fit = fit + genetique.TI[ind].TpsMort[j];
        return fit;
    }
//-----------------------------------------------------------------------
```

Figure 11 - Algorithme de calcul de la fonction de fitness

- **Étape 3 : La sélection des individus :**

La sélection tend à augmenter l'importance des bonnes solutions par rapport aux mauvais. C'est une heuristique utilisé par l'algorithme génétique : les bonnes solutions sont supposées être les plus prometteurs. Pour notre cas, nous allons sélectionné les individus ayant le temps de réponse le plus rapide.

- **Étape 4 : Le croisement des individus :**

Le croisement consiste à copier et recombiner les gènes de deux individus parent de façon à former deux nouveaux individus fils possèdent des caractéristiques issues des parent.

Il existe type plusieurs types de croisement, nous présentons brièvement les trois principaux :

- ✓ Le croisement « un point » détermine aléatoirement un point de coupure et échange la deuxième partie des deux parents.
- ✓ Le croisement « deux points » possède deux points de coupures qui sont déterminés aléatoirement. Les deux points de croisement sont également choisit au hasard, puis, les séquences des chromosomes situés entre les deux points sont échangées
- ✓ Enfin, le croisement uniforme échange chaque bit avec une probabilité fixée.

Le croisement utilisé dans notre étude est le croisement avec « un point ».

- **Étape 5 : La mutation :**

Traditionnellement, la mutation est appliquée aux enfants avec une probabilité faible pour empêcher une convergence locale de l'algorithme génétique. Dans notre cas, l'algorithme converge vers le minimum local à 5%.

- **Étape 6 :Réitération du processus :**

Le programmeur a l'opportunité d'évaluer les individus de sa population avant ou/et après les phases de création d'individus. En effet, il peut s'avérer pertinent de les évaluer avant de les insérer dans la future population, de même qu'il peut être utile de le réévaluer au début de la génération suivante. Ainsi on peut être amené à évaluer deux fois par génération (itération) chacun des individus.

Une fois le nombre maximum d'itération atteint, on obtient une population de solution. Mais rien ne nous dit que la solution théorique optimal aura été trouvé. Les solutions se rapprochent des bonnes solutions, mais sans plus. Ce n'est pas une méthode exacte.

.3.4. La conception de la solution

3.4.1. L'architecture fonctionnelle du système associée aux l'algorithme génétique

L'organigramme de la figure suivante résume de façon générale l'implémentation de l'algorithme génétique dans notre système. Lors de l'arrivée des conteneurs, on aborde le problème de sélection des conteneurs. Le résultat final sera l'affectation des conteneurs dans un micro-service de façons optimale.

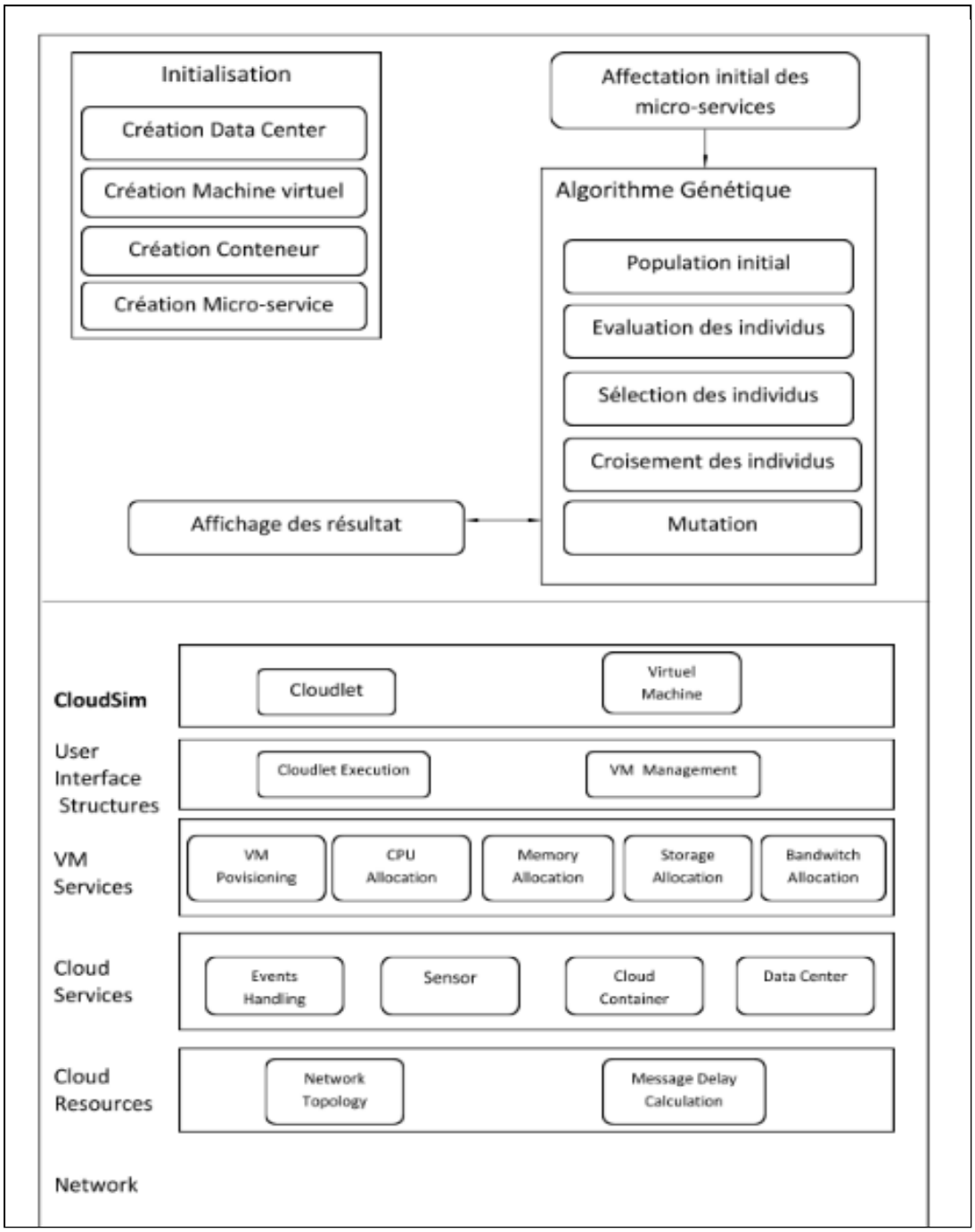


Figure 12 - Architecture fonctionnelle du système de sélection des conteneur dans les Micro-Services

3.4.2. Modélisation UML du modèle proposé

3.4.2.1. Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation permet de définir le système, les acteurs, les cas d'utilisation et les liens entre eux. Dans notre système de simulation les cas d'utilisation commencent lorsque l'utilisateur crée des data center qui va consister en l'allocation des hôtes et des VMs. Ensuite, l'utilisateur va entamer les étapes de création des micro-services, puis des conteneurs et faire l'affectation de ces micro-services aux conteneurs. Enfin il va lancer le processus d'estimation de la meilleure sélection en utilisant l'algorithme génétique.

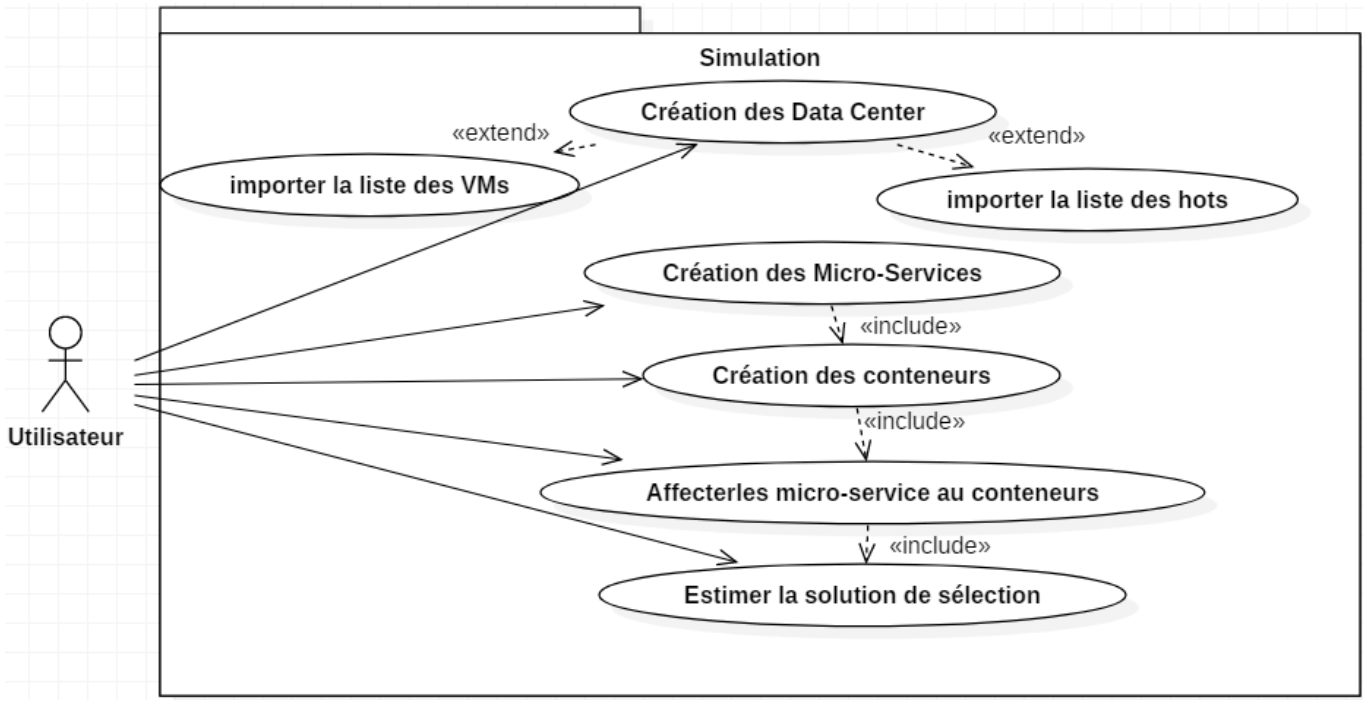


Figure 13 –Diagramme de Cas d'utlisation

3.4.2.2. Diagramme de classe

Le diagramme de classe est considéré comme le plus important de modélisation orienté objet. La figure 14 illustre le diagramme de classe de notre travail qui est constitué huit classes : CloudSim, Simulateur, Algorithme génétique, Conteneur, Micro-service, Data center, Machine virtuelle et Hôte qui sont reliés par des liens de composition par exemple le data center contient des hôtes et des machines virtuelles. Lorsque l''utilisateur crée les conteneurs, les micro-services, les hôtes, les data center et les machines virtuelles, le simulateur lancera le processus de sélection à travers l'algorithme génétique.

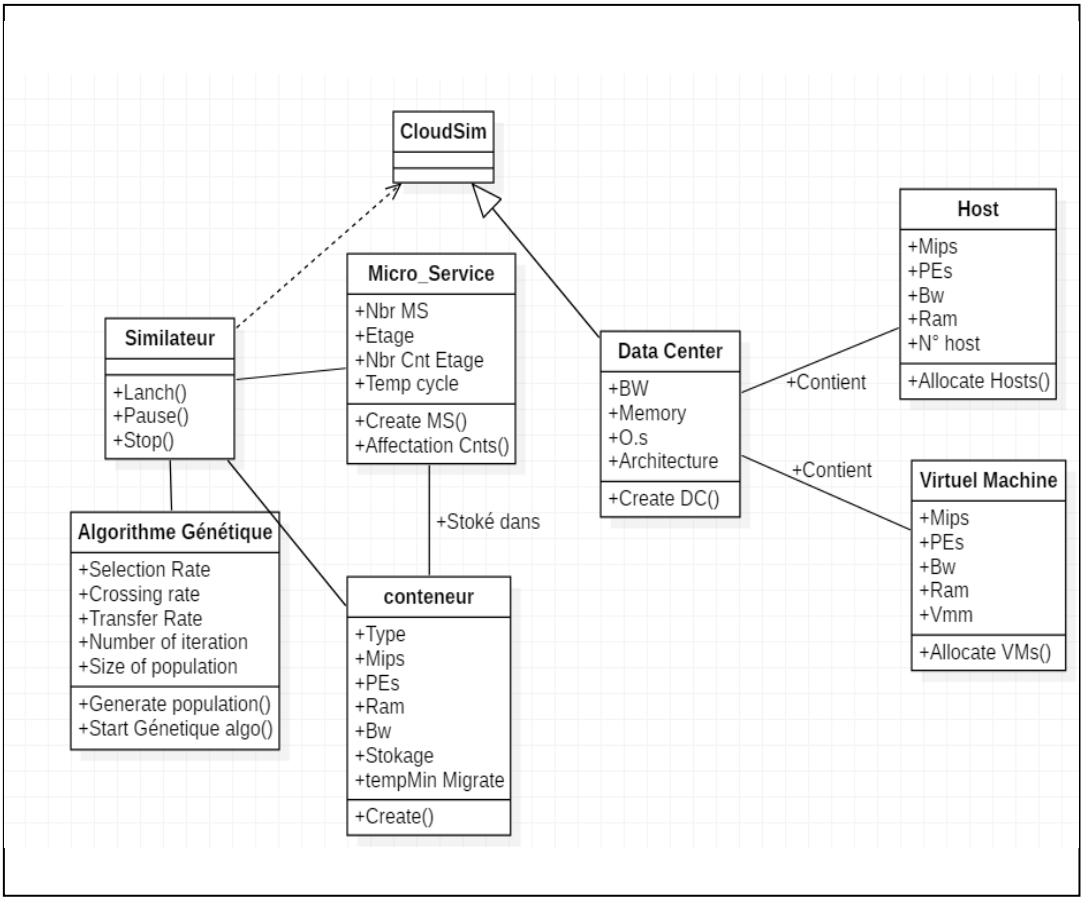


Figure 13 - Diagramme de classe

3.4.2.3. Diagramme d'activité

Ce diagramme est une formalisation graphique des actions qui sont réalisées dans un cas d'utilisation. La figure 15 illustre le diagramme d'activité de notre travail et les différentes représentations des actions. Après avoir listé les micro-services, le simulateur procède au traitement du problème de sélection, en lançant l'algorithme génétique. Ce traitement va déclencher la création des individus, l'évaluation des individus, la sélection des individus, leur croisement et mutation. Une fois l'algorithme générique terminé, ce programme renvoie les solutions de sélection optimales.

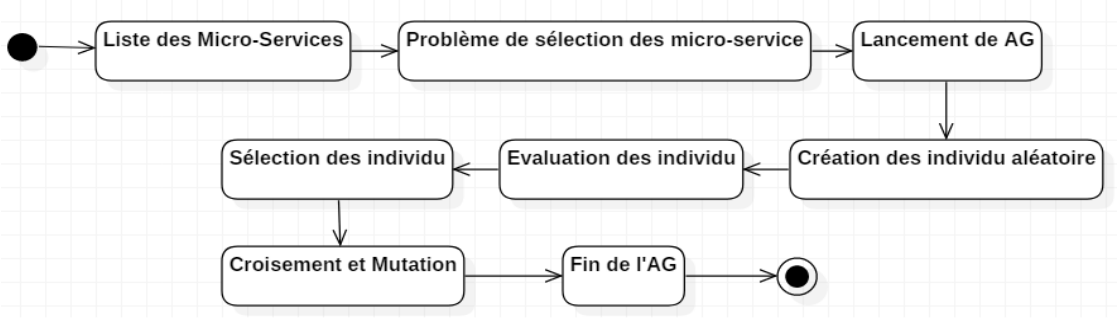


Figure 14 - Diagramme d'activité

3.4.2.4. Diagramme de séquence

Le diagramme de séquence est la représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique dans la formulation Unified Modeling Language.

Le diagramme de séquence de notre travail est très simple : l'utilisateur crée des conteneurs et des micro-services, puis lance le processus d'affectation qui va placer initialement les micro-services dans les conteneurs. Cette liste sera renvoyée ç l'utilisateur qui va pouvoir lancer le processus de recherche de la meilleure solution de sélection. Une fois, la sélection faîte, la solution est renvoyée à l'utilisateur.

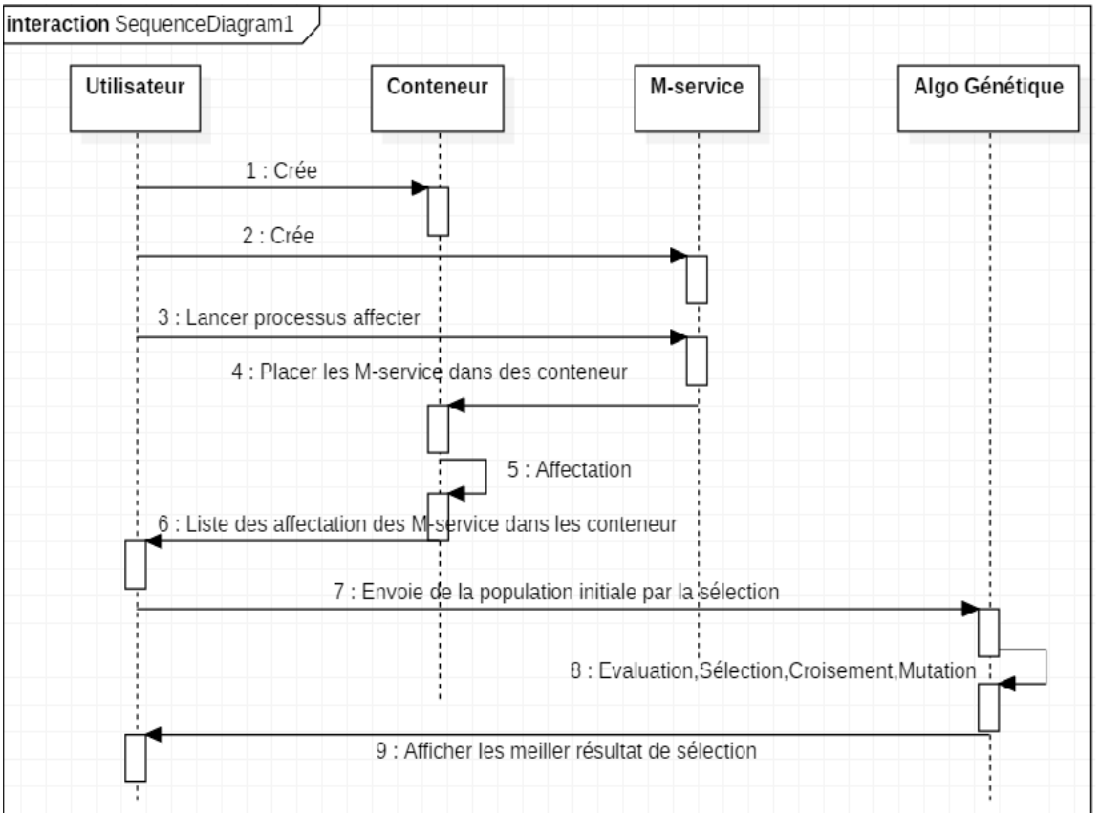


Figure 16 – Diagramme de séquence

3.5. Conclusion

Pour conclure, nous avons pu à travers ce chapitre, présenter les étapes de la conception et modélisation de l'approche proposée en utilisant l'algorithme génétique afin de trouver une solution performante au problème de sélection de conteneurs dans un environnement de Cloud Computing.

Dans le chapitre suivant, nous allons procéder à la réalisation et la mise en œuvre de ce travail et présenter le résultat final de notre application avec des captures d'écrans et des exemples d'exécutions.

Chapitre 4 : Implémentation et expérimentations

.4.1. Introduction

Dans cette partie, nous présentons la mise œuvre de nos contributions, avec des captures d'écran, et des exemples d'exécution.

L'objectif de la phase d'implémentation est d'aboutir à un produit final, exploitable par les utilisateurs. En premier lieu, nous présenterons les outils de développement d'où nous spécifierons l'environnement de travail et le langage de programmation utilisé. Quant à la deuxième partie nous décrirons notre application par la présentation des scénarios les plus généraux de notre application illustrés par des captures d'écrans.

.4.2. Outils de développement

Dans cette section, nous présentons les outils de développement utilisés pour implémenter notre application.

.4.2.1. L'environnement de travail

Nous avons choisi e » développer notre logiciel avec l'environnement Net-Beans version 8.2, Figure(17) qui est un environnement de développement intégré(EDI), placé en open source par Sun en juin 2000 sous License CDDL(Common Dévelopment and Distribution License). En plus de java, NetBeans permet également de supporter différents autre langage, comme C, C++, JavaScript, XML, Groovy, PHP et HTML de façon native ainsi que bien d'autres(comme Phyton ou Ruby) par l'ajout de greffons. Il comprend toutes les caractéristiques d'un IDE moderne(éditeur en couleur, projets multi-langage, réfactoring, éditeur graphique d'interfaces et de pages web).+++-

Concu en Java, NetBeans est disponible sous Windows, Linux, Solarus(sur x86 et SPARC),Mac OS X ou sous une version indépendante des systèmes d'exploitation(requérant une machine virtuelle Java).

NetBeans consititue par ailleurs une plate forme qui permet le développement d'applications spécifiques(bibiliothéque Swing(Java)). L'IDE NetBeans s'appuie sur cette plate forme.

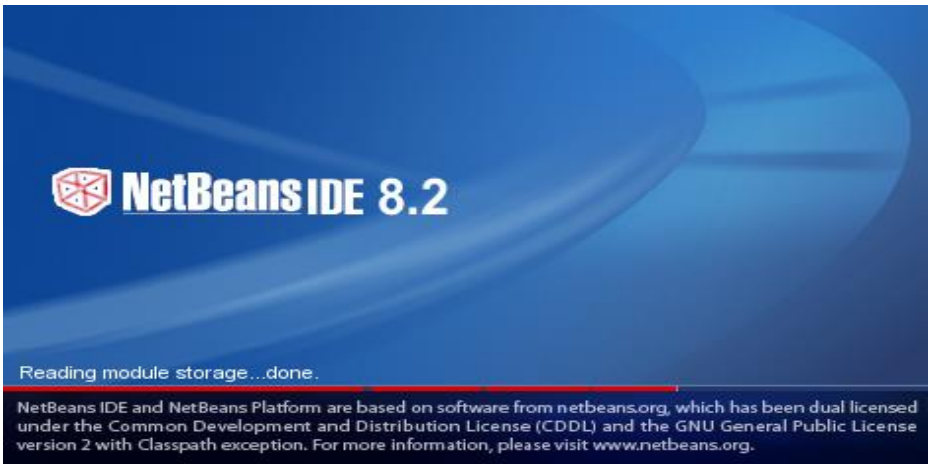


Figure17 – NetBeansIDE 8.2

4.2.2. Le langage de programmation

Nous avons choisi de développer notre logiciel avec le langage Java qui est un langage de programmation informatique orienté objet créé par James Gosling et Ptarick naughton, employés de Sun Microsystems, avec le solution de Bill Joy (cofondateur de Sun Microsystems 1982), présenté officiellement le 23 mai1995 au SunWorld. La société Sun a été ensuite rachetée en 2009 par la société Oracle qui détient et maintient désormais Java.

La particularité et l'objectif central de java est que les logiciels écrits dans un ce langage doivent être très facilement portables sur plusieurs systèmes d'exploitation tels que UNIX, Windows, Mac OS ou GNU/Linux, avec peu ou pas de modification , Figure (A.2). pour cela ,divers plateformes et Framework associés visent à guider, sinon garantir, cette portabilité des applications développées en Java.

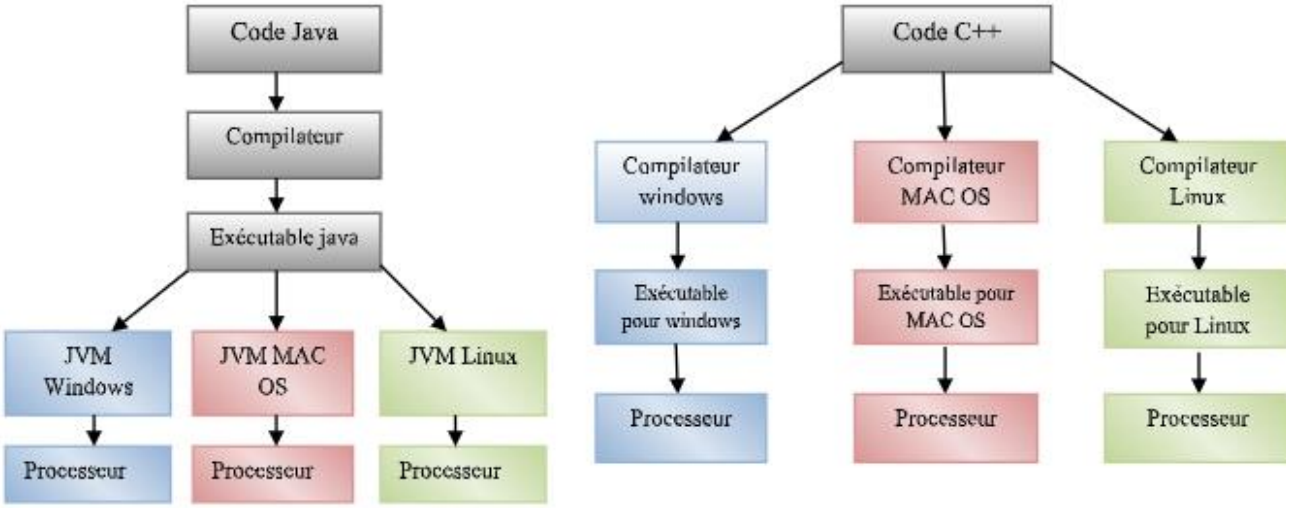


Figure 15 - Comparaison entre le fonctionnement d'un programme java et un programme écrit en C++

4.2.3. Simulateur du cloud computing : CloudSim

CloudSim est un Framework pour la modélisation et la simulation d' infrastructures et de services de cloudcomputing . Construit à l'origine principalement pour le laboratoire Cloud Computing and Distributed Systems (CLOUDS) à l'Université de Melbourne en Australie, CloudSim est devenu l'un des simulateurs de cloud open source les plus populaires dans la recherche et le monde universitaire. CloudSim est complètement écrit en Java.

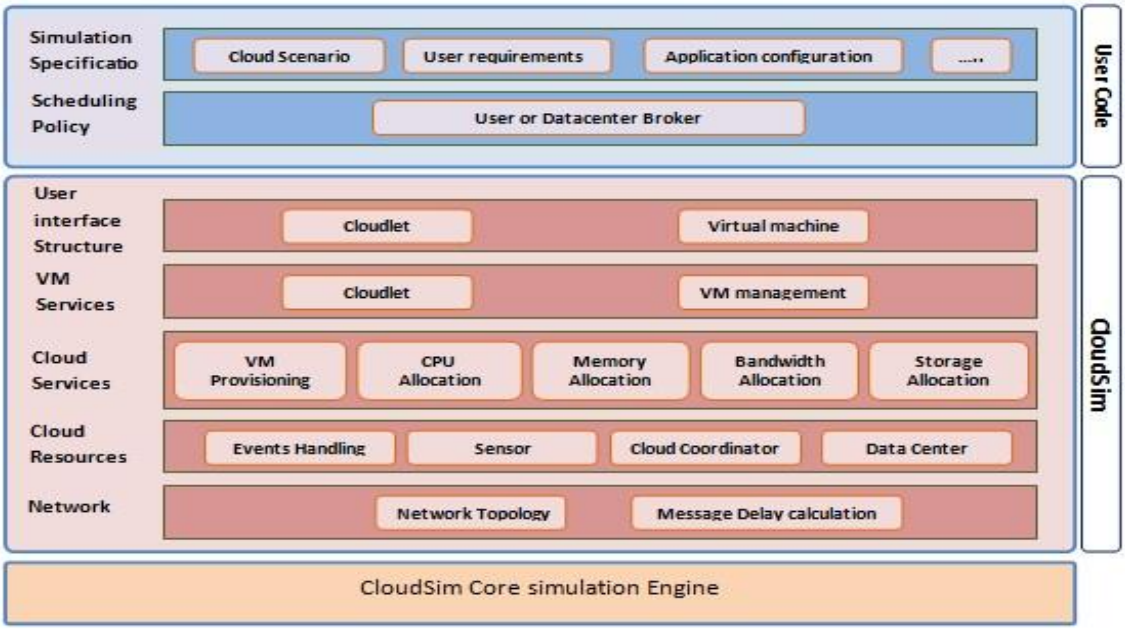


Figure 18- Architecture de CloudSim

La figure illustre les différentes couches de la structure du CloudSim et ses éléments architecturaux. Au niveau le plus bas est le moteur de simulation aux événements discrets SimJava, qui implémente les fonctionnalités de base requises pour les cadres de simulation au niveau supérieur, telles que les files d'attente, le traitement des événements, création de composants du système (services, hôte, Datacenter, Broker, les machines virtuelles), la communication entre les composants et la gestion de l'horloge de simulation.

.4.3. Description de l'application

Dans cette section nous décrivons le fonctionnement de notre application à travers un exemple de déroulement détaillé.

4.3.1. Lancement de l'application

A L'aide d'une interface d'accueil, l'utilisateur peut accéder à la fenêtre de l'application via le bouton« « Entre » » qui permet d'afficher l'interface principale. De même le bouton « Quitter » permet de quitter l'application.

4.3.2. Présentation de l'interface Principale

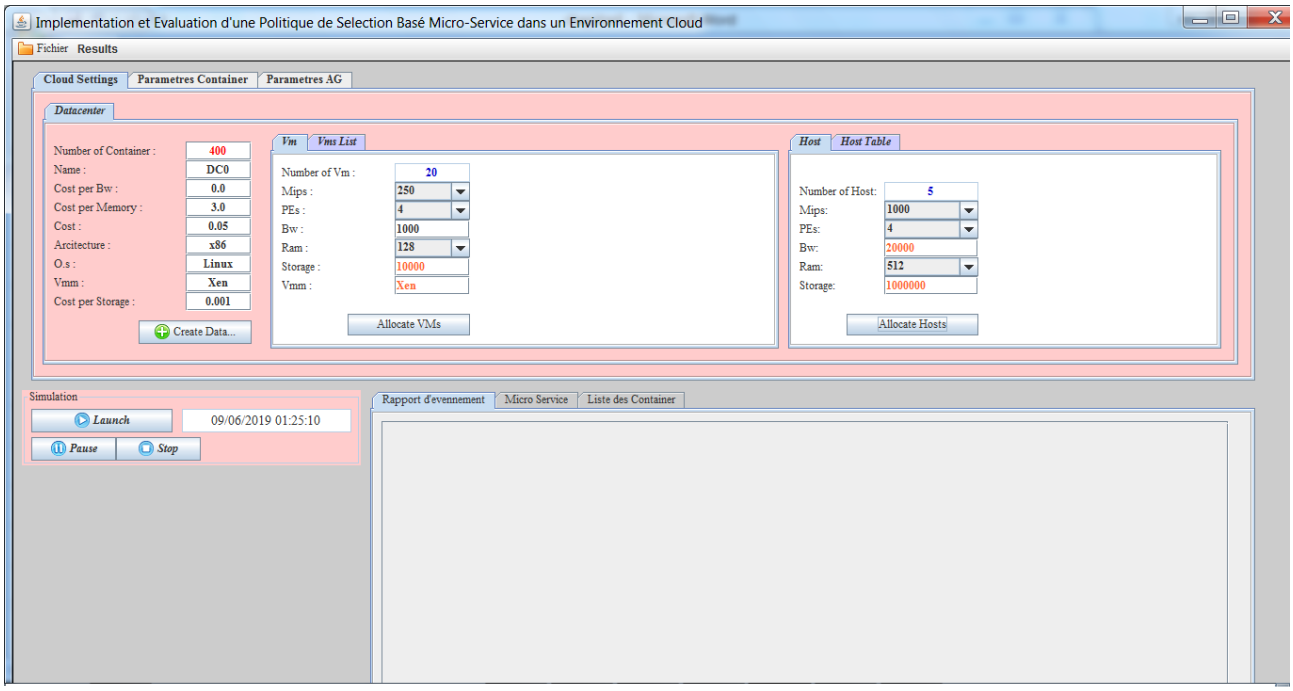


Figure 19 – Interface de Configuration

Après avoir accéder à l'interface principale, on clique sur me menu fichier pour commencer la configuration du cloud.

La fonction de configuration du Cloud permet de faire la création des Data Center, allocation et importation de la liste des VMs et de la liste des Hôtes. Pour cela, on configure le tes tailles des hôtes, des machines virtuelles machine et leur paramètres pour faire la création des conteneurs sélectionné.

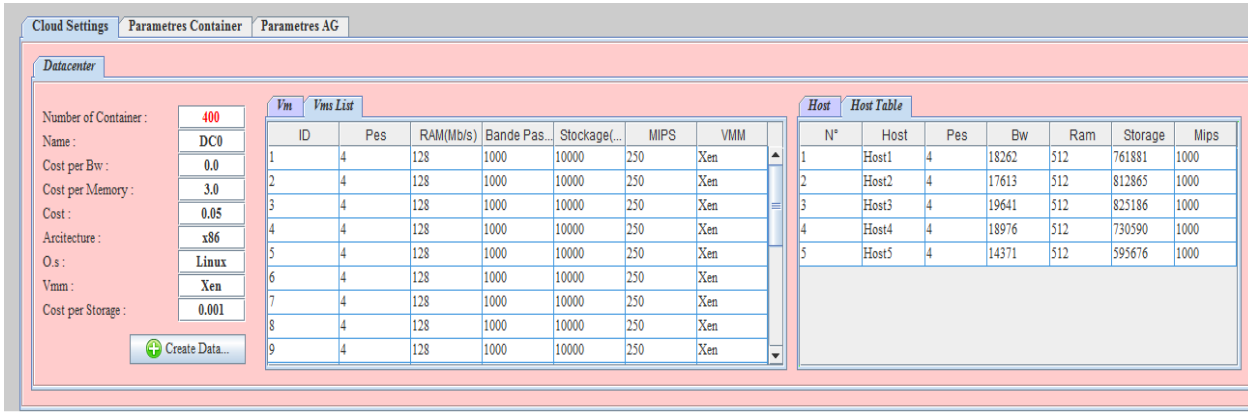


Figure 16 - Création Datacenter, des VMs et Hôtes

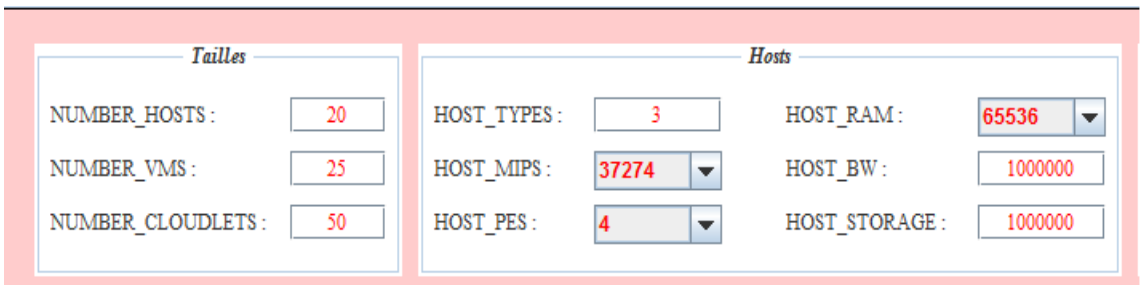


Figure 22 – Configuration des tailles et hots

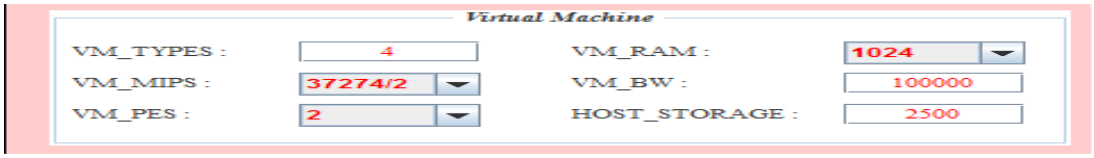


Figure 17 - Configuration des VMs

L'onglet suivant représente la configuration des paramètres des micro-services par la définition des types de conteneur et des ressources nécessaires au conteneur pour son exécution.

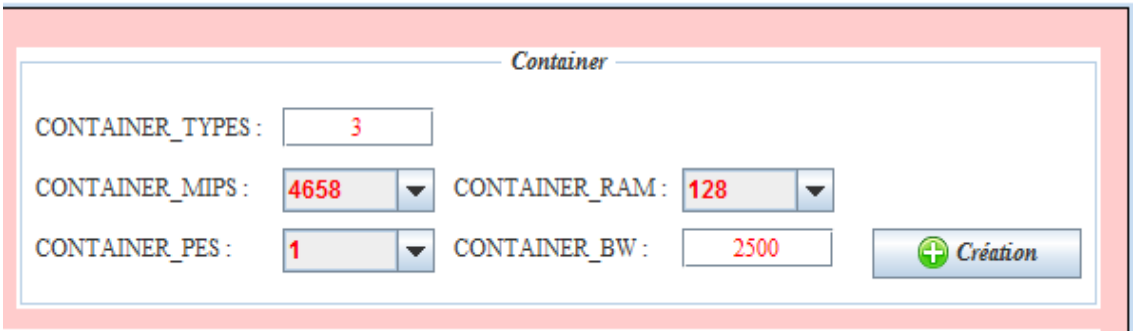


Figure 18 - Création des conteneurs

Lorsqu'on a terminé l'étape de création des micro-services, automatiquement la liste des micro-services sera affiché en bas en indiquant chaque conteneur avec ces critères.

Rapport d'evennement | Containers | Liste des Micro-service

N° Cont	Type	Mips	Pes	Ram	Bw	Stockag	Temps Min Mig.
1	2	4658	3	256	2500	30000	16.67
2	3	4658	3	384	2500	30000	1.95
3	2	9316	1	256	2500	30000	15.7
4	3	9316	3	128	2500	20000	91.84
5	3	9316	1	384	2500	10000	79.33
6	2	4658	2	128	2500	10000	17.44
7	1	9316	2	256	2500	20000	52.55
8	3	9316	1	256	2500	30000	95.74
9	2	13974	3	384	2500	30000	94.4
10	1	9316	3	256	2500	20000	64.77
11	2	9316	1	128	2500	30000	19.97
12	3	4658	3	384	2500	30000	5.74
13	3	4658	3	128	2500	10000	34.33
14	1	9316	2	128	2500	30000	97.08
15	3	13974	2	384	2500	10000	16.99

Figure 19 - Détails des micro-services

Ensuite, on commence l'étape de création des micro services et leur affectation aux conteneurs.

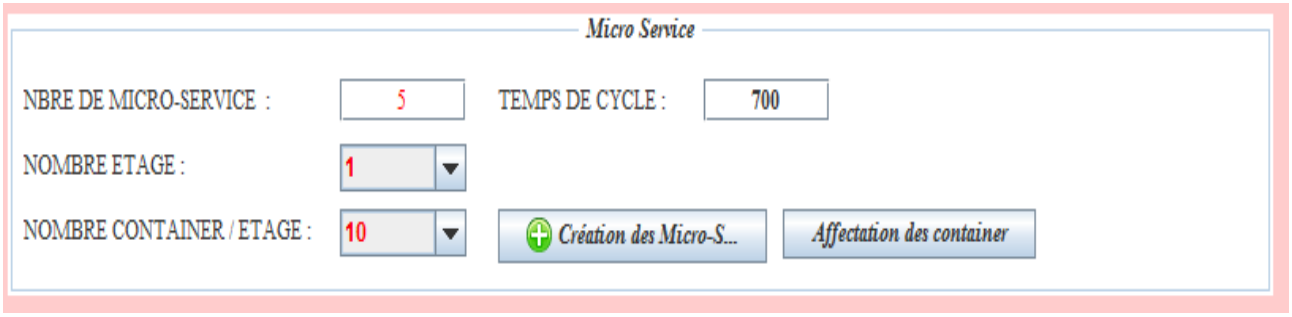


Figure 20 - Création et affectation des micro-services

La figure suivante décrit le contenu de chaque conteneur (c'est à dire les micro-services affectés déjà aux conteneurs) avec l'affichage de temps total et du temps mort.

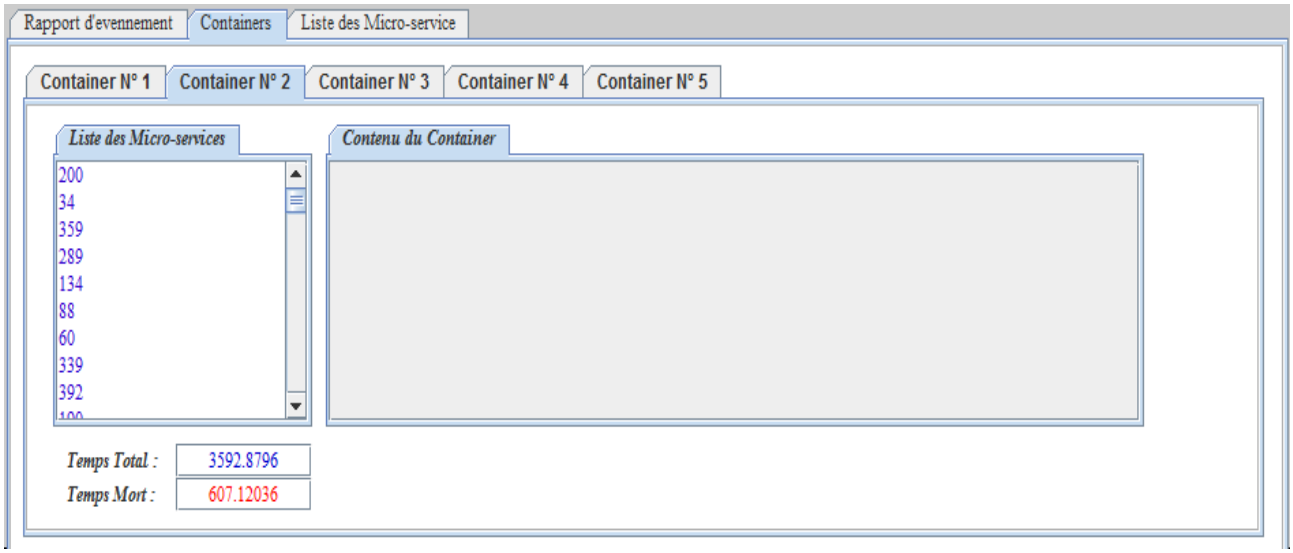


Figure 21 - Détails des conteneurs

Après avoir fait aléatoirement l'affectation initiale des micro-services dans les conteneurs, on passe à l'étape de sélection avec l'algorithme génétique.

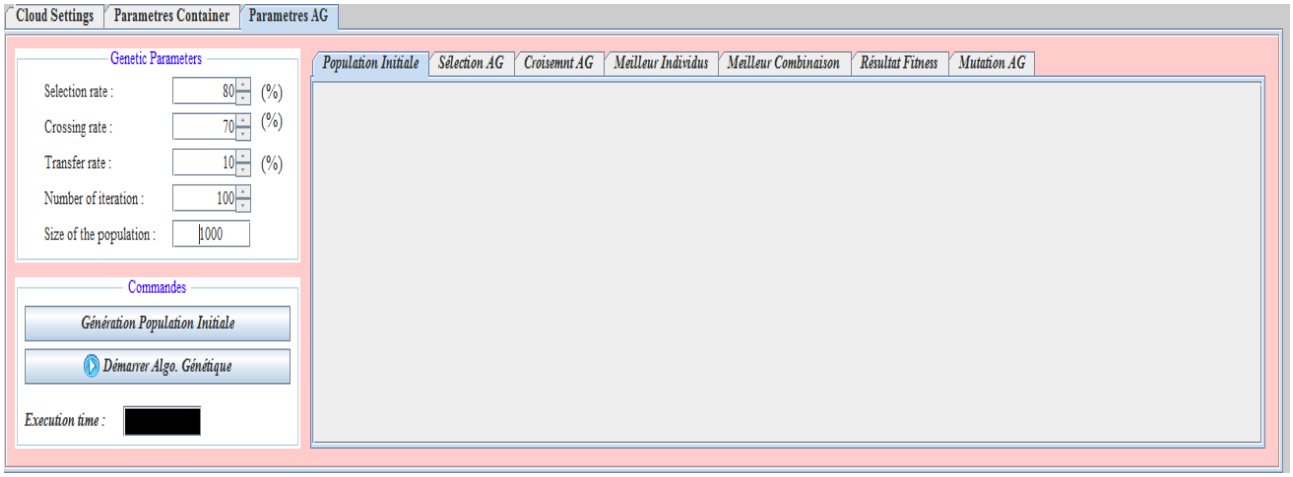


Figure 22 - Lancement de l'algorithme génétique

Dans notre exemple on fixe le nombre itération de l'algorithme génétique à 100 :

Taux de Sélection	**80 Pourcent**
Taux de Croussing	70 Pourcent
Taux de Transfer	10 Pourcent
Nombre Itération	100
Taille de Population	1000

Table 1 : Paramètre d'algorithme génétique

Puis, on clique sur le bouton générer la population initiale afin de créer la population initiale :

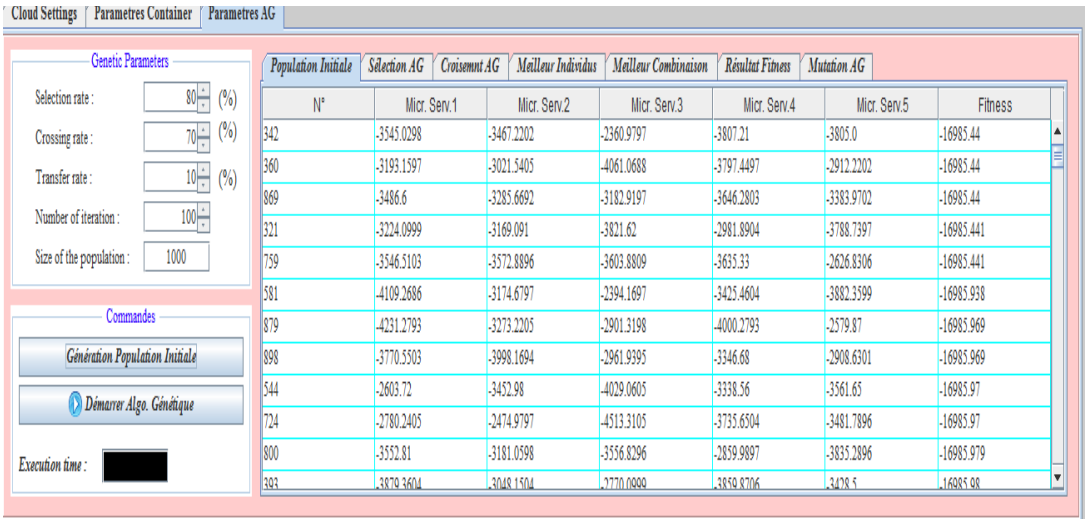


Figure 23 - Génération de la population initiale

On obtient par la suite la création de notre population initiale qui est la sélection des conteneurs dans les micro-services.

Après le lancement de l'algorithme génétique on obtient :

1) le temps de calcul qui est le temps exact pour trouver la solution optimale de notre problème :

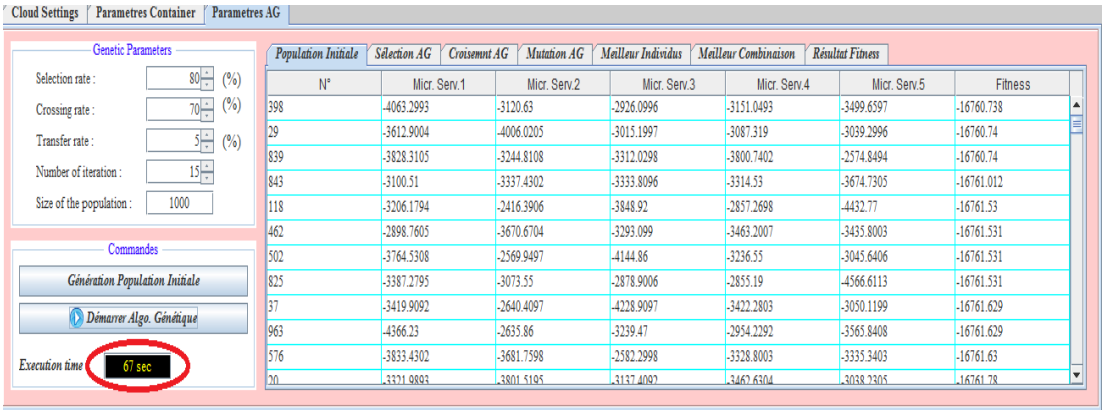


Figure 24 - Affichage du temps d'exécution

2) Génération de la population sélectionné, croisée et mutée et l'obtention de résultat finale.

- Population sélectionnée : qui représente 80 pour cent de la population initiale(dans notre exemple 80 individus triés en ordre décroissant selon la fonction de fitness).

Population Initiale | Sélection AG | Croisemnt AG | Mutation AG | Meilleur Individus | Meilleur Combinaison | Résultat Fitness

N°	Micr. Serv.1	Micr. Serv.2	Micr. Serv.3	Micr. Serv.4	Micr. Serv.5	Fitness
435	-3347.1702	-3912.8896	-3337.2798	-2775.01	-3425.9106	-16798.26
572	-3129.789	-3338.6504	-3464.3208	-3267.4297	-3624.4092	-16824.598
319	-4056.8901	-4116.5293	-3119.9905	-3003.74	-2474.2498	-16771.4
744	-3208.9402	-3895.2998	-3507.4395	-3590.6108	-2593.0999	-16795.39
22	-2921.9297	-3238.0593	-3620.2305	-3049.89	-3968.6099	-16798.719
226	-2966.7705	-2496.2703	-3596.7598	-3714.25	-4064.059	-16838.11
123	-3638.271	-3521.0298	-3792.5205	-2881.6597	-2980.0298	-16813.51
482	-4306.681	-3071.7007	-3517.629	-2805.1895	-3129.7	-16830.9
29	-3612.9004	-4006.0205	-3015.1997	-3087.319	-3039.2996	-16760.74
862	-3296.9592	-3643.1294	-3027.6897	-3700.0708	-3154.9497	-16822.799
265	-3294.8396	-3494.5698	-3907.6094	-3285.4692	-2813.8103	-16796.299

Figure 25 - Population sélectionnée

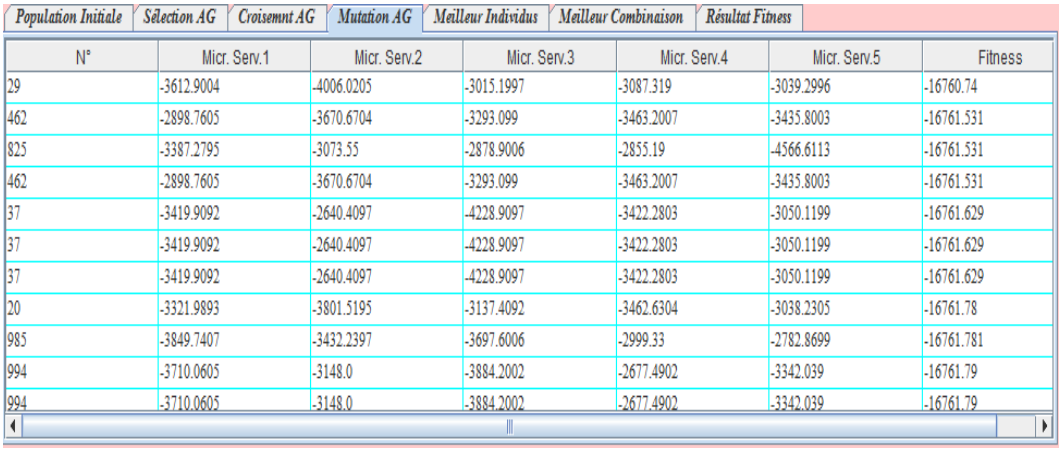
- Population croisée : qui représente 70 pour cent de la population sélectionnée (dans notre exemple 56 individu triés en ordre décroissant en fonction de la fitness).

Population Initiale | Sélection AG | Croisemnt AG | Mutation AG | Meilleur Individus | Meilleur Combinaison | Résultat Fitness

N°	Micr. Serv.1	Micr. Serv.2	Micr. Serv.3	Micr. Serv.4	Micr. Serv.5	Fitness
29	-3612.9004	-4006.0205	-3015.1997	-3087.319	-3039.2996	-16760.74
462	-2898.7605	-3670.6704	-3293.099	-3463.2007	-3435.8003	-16761.531
825	-3387.2795	-3073.55	-2878.9006	-2855.19	-4566.6113	-16761.531
462	-2898.7605	-3670.6704	-3293.099	-3463.2007	-3435.8003	-16761.531
37	-3419.9092	-2640.4097	-4228.9097	-3422.2803	-3050.1199	-16761.629
37	-3419.9092	-2640.4097	-4228.9097	-3422.2803	-3050.1199	-16761.629
37	-3419.9092	-2640.4097	-4228.9097	-3422.2803	-3050.1199	-16761.629
20	-3321.9893	-3801.5195	-3137.4092	-3462.6304	-3038.2305	-16761.78
985	-3849.7407	-3432.2397	-3697.6006	-2999.33	-2782.8699	-16761.781
994	-3710.0605	-3148.0	-3884.2002	-2677.4902	-3342.039	-16761.79
994	-3710.0605	-3148.0	-3884.2002	-2677.4902	-3342.039	-16761.79

Figure 26 - Population croisée

Population mutée : dans notre exemple le taux de mutation est de 5 pour cent.

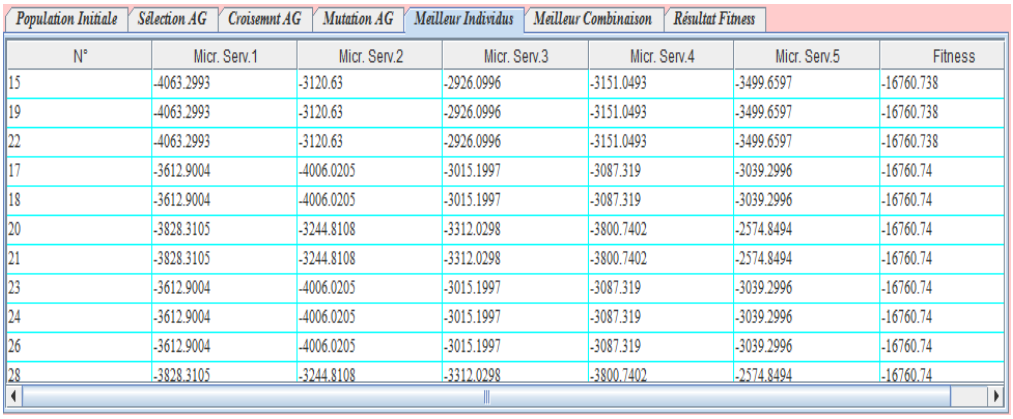


Population Initiale | Sélection AG | Croisemnt AG | Mutation AG | Meilleur Individus | Meilleur Combinaison | Résultat Fitness

N°	Micr. Serv.1	Micr. Serv.2	Micr. Serv.3	Micr. Serv.4	Micr. Serv.5	Fitness
29	-3612.9004	-4006.0205	-3015.1997	-3087.319	-3039.2996	-16760.74
462	-2898.7605	-3670.6704	-3293.099	-3463.2007	-3435.8003	-16761.531
825	-3387.2795	-3073.55	-2878.9006	-2855.19	-4566.6113	-16761.531
462	-2898.7605	-3670.6704	-3293.099	-3463.2007	-3435.8003	-16761.531
37	-3419.9092	-2640.4097	-4228.9097	-3422.2803	-3050.1199	-16761.629
37	-3419.9092	-2640.4097	-4228.9097	-3422.2803	-3050.1199	-16761.629
37	-3419.9092	-2640.4097	-4228.9097	-3422.2803	-3050.1199	-16761.629
20	-3321.9893	-3801.5195	-3137.4092	-3462.6304	-3038.2305	-16761.78
985	-3849.7407	-3432.2397	-3697.6006	-2999.33	-2782.8699	-16761.781
994	-3710.0605	-3148.0	-3884.2002	-2677.4902	-3342.039	-16761.79
994	-3710.0605	-3148.0	-3884.2002	-2677.4902	-3342.039	-16761.79

Figure 27 - Population mutée

- Meilleur individus : représente l'ensemble des meilleurs individus qui ont presque les mêmes taux de sélections.

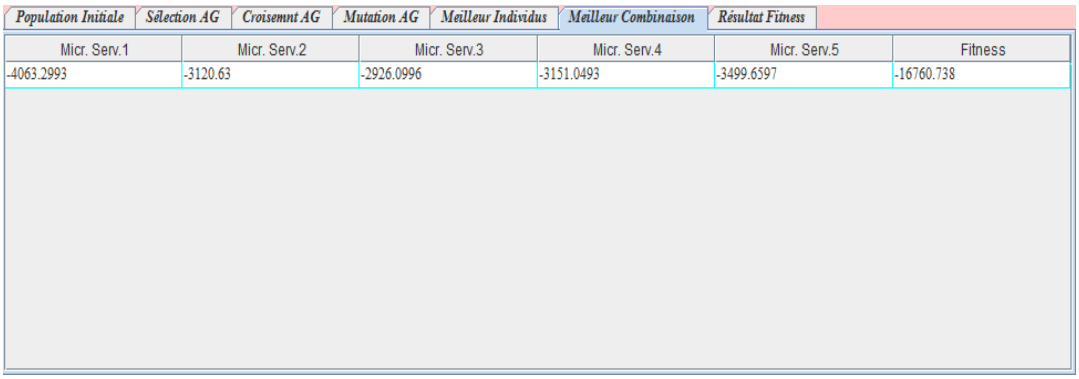


Population Initiale | Sélection AG | Croisemnt AG | Mutation AG | Meilleur Individus | Meilleur Combinaison | Résultat Fitness

N°	Micr. Serv.1	Micr. Serv.2	Micr. Serv.3	Micr. Serv.4	Micr. Serv.5	Fitness
15	-4063.2993	-3120.63	-2926.0996	-3151.0493	-3499.6597	-16760.738
19	-4063.2993	-3120.63	-2926.0996	-3151.0493	-3499.6597	-16760.738
22	-4063.2993	-3120.63	-2926.0996	-3151.0493	-3499.6597	-16760.738
17	-3612.9004	-4006.0205	-3015.1997	-3087.319	-3039.2996	-16760.74
18	-3612.9004	-4006.0205	-3015.1997	-3087.319	-3039.2996	-16760.74
20	-3828.3105	-3244.8108	-3312.0298	-3800.7402	-2574.8494	-16760.74
21	-3828.3105	-3244.8108	-3312.0298	-3800.7402	-2574.8494	-16760.74
23	-3612.9004	-4006.0205	-3015.1997	-3087.319	-3039.2996	-16760.74
24	-3612.9004	-4006.0205	-3015.1997	-3087.319	-3039.2996	-16760.74
26	-3612.9004	-4006.0205	-3015.1997	-3087.319	-3039.2996	-16760.74
28	-3828.3105	-3244.8108	-3312.0298	-3800.7402	-2574.8494	-16760.74

Figure 28 - Meilleurs individus

- Meilleur combinaison : représente notre solution de sélection optimale.



Population Initiale | Sélection AG | Croisemnt AG | Mutation AG | Meilleur Individus | Meilleur Combinaison | Résultat Fitness

Micr. Serv.1	Micr. Serv.2	Micr. Serv.3	Micr. Serv.4	Micr. Serv.5	Fitness
-4063.2993	-3120.63	-2926.0996	-3151.0493	-3499.6597	-16760.738

Figure 29 - Meilleure combinaison

- Résultat fitness : dans cette interface, on affiche les taux de fitness pour les sélections optimales des micro-services dans les conteneurs.

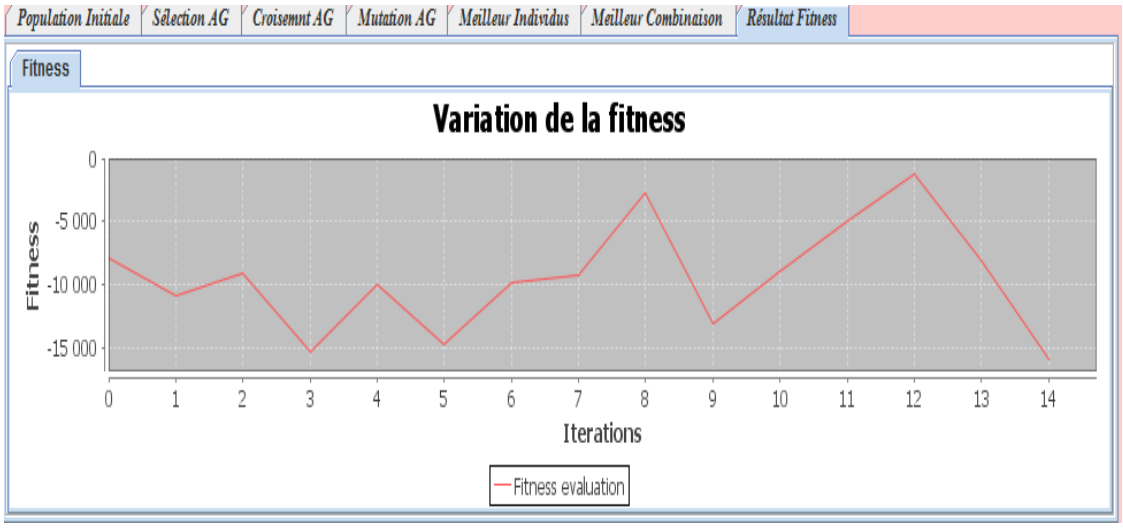


Figure 30 - Résultat fitness

A l'aide de l'algorithme génétique on obtient des sélections optimales des micro-services dans les conteneurs, on les représente dans une zone de simulation qui nous permet d'établir des calcules.

Notre rapport d'évènement affiche le traitement des micro-services à partir le temps (temps début, temps traitement, temps d'exécution).

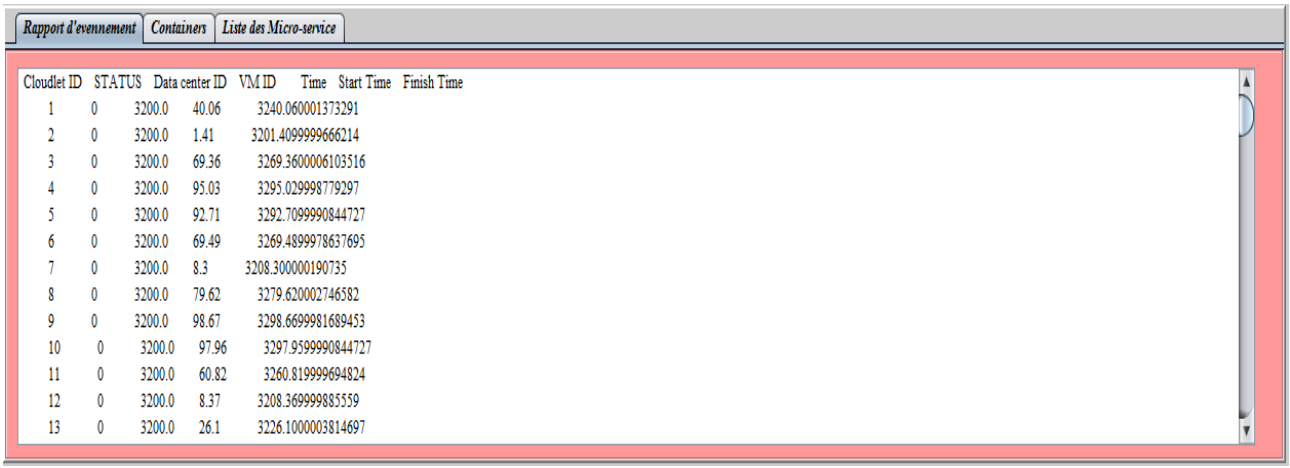


Figure 37 : Rapport d'évènement

La figure(38) représente la sélection finale des micro-services dans des conteneurs avec un graphe qui représente le temps finant par rapport chaque micro-service

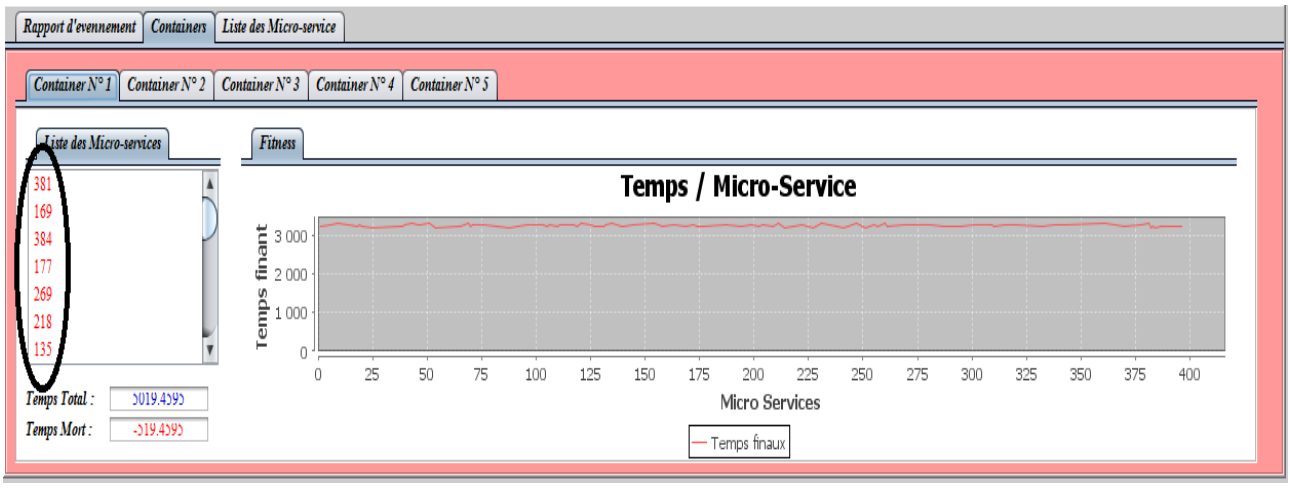


Figure 39 : La sélection finale des micro-service

Conclusion Générale et futures perspectives

La problématique de ce travail était de proposer une solution pour la sélection et l'ordonnancement des conteneurs dans un environnement de Cloud Computing pour le placement des micro-services.

Dans ce contexte, nous avons proposés une solution de sélection des mirco-services basé sur l'optimisation de l'ordonnancement des conteneurs. Nous avons choisis d'utiliser les algorithmes heuristiques pour traiter notre problème afin de trouver une solution optimale. Nous avons choisit d'implémenter l'algorithme génétique dont le rôle principal est d'essayer de converger vers la meilleure solution, en un temps raisonnable.

Nous avons tenu en compte les critères suivant pour le proposition de la solution : le type de conteneur et le fait que le service ne peut avoir qu'un et un seulement emplacement de sélection.

Nous avons aussi impélemnter et tester notre approche en utilisant le simulateur CloudSim à travers l'IDE NetBeens.

Enfin, les points suivants peuvent être considérés pour de futures continuations de notre travail :

- Comparer la solution proposée avec d'autres solutions.
- Tester une autre technique d'optimisation outre l'algorithme génétique.
- Tester la solution sur un environnement Cloud réel.

Références bibliographiques

1. Containerizing monolithic applications.[en ligne].consulté le :03/02/2015.Disponible sur address : https://github.com/dotnet/docs/blob/master/docs/standard/microservices-architecture/architect-microservice-container-applications/containerize-monolithic-applications.md.
2. C'est quoi le cloud ?.[en ligne]. Consulté le :13/02/2014.Disponible sur address : https://www.culture-informatique.net/cest-quoi-le-cloud/.
3. Les conteneurs au sein de google.[en ligne].Consulté le :28/07/2016. Disponible sur address : https://cloud.google.com/containers/?hl=fr
4. Bonnes pratiques en matière de création de conteneurs.[en ligne].Consulté le :27/03/2019.Disponible sur address : https://cloud.google.com/solutions/best-practices-for-building-containers?hl=fr
5. Conteneur.[en ligne]. Consulté le :05/01/2018.Disponible sur address : https://fr.wikipedia.org/wiki/Conteneur.
6. Cloud computing et entreprise : historique et définitions du cloud computing.[en ligne].Consulté le :01/08/2017.Disponible sur address : https://fr.wikiversity.org/wiki/Cloud_computing_et_entreprise/Historique_et_d%C3%A9finition_du_Cloud_Computing
7. Les conteneurs sont-ils l'avenir de votre production informatique ?.[en ligne].Consulté le :08/11/2016.Disponible sur address : https://blog.d2si.io/2016/11/08/conteneurs-avenir-production/.
8. Comparison of container schedulers.[en ligne]. Consulté le :24/02/2016.Disponible sur address : https://medium.com/@ArmandGrillet/comparison-of-container-schedulers-c427f4f7421.
9. Disponible sur address :https://lipn.univ-paris13.fr/~menouer/Rapport_PFE.pdf
10. Cloud : les modèles de déploiement.[en ligne]. Consulté le : 01/03/2016.Disponidble sur address : https://blog.3li.com/cloud-les-modeles-de-deploiement/
11. P.F.Dutot, L.Eyraud, G.Mounié, D.Trystram, "Bi-criteria algorithm for scheduling jobs on cluster platforms". In Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures. pp. 125_132. SPAA '04, Barcelona, Spain, ACM, New York, NY, USA 15, 23. 2004.
12. D.Ye, G.Zhang, "On-line scheduling of parallel jobs in a List". J. of Scheduling 10(6), 407_413, Kluwer Academic Publisher, Hingham, MA, USA 15, Déc. 2007.
13. J.Li, M.Qiu, J.Niu, W.Gao, Z.Zong, "Feedback dynamic algorithms for preemptable job scheduling in cloud systems". Proceedings of the International Conférence on Web Intelligence and Intelligent Agent Technology, Aug. 31-Sep. 3, IEEE Xplore Press, Toronto, ON, pp: 561-564. DOI: 10.1109/WI-IAT.2010.30, 2010.
14. J.Li, J.Peng, W.Zhang, "A scheduling algorithm for private clouds". J. Convergence Inform. Technol., 6: 1-9. 2011.

15. P.K.Gupta, N.Rakesh, “Different job scheduling methodologies for web application and web server in a cloud computing environment”. Proceedings of the 3rd International Conference on Emerging Trends in Engineering and Technology, Nov. 1921, IEEE Xplore Press, Goa, pp: 569-572. DOI: 10.1109/ICETET.2010.24, 2010.

16. J.Li, J.Huai, T.Wo, Q.Li, “EnaCloud : An energy-saving application live placement approach for cloud computing environments”. Proceedings of the International Conference on Cloud Computing, Sept. 21-25, IEEE Xplore Press, Bangalore, pp: 17-24. DOI: 10.1109/CLOUD.2009.72, 2009.

17. S.Sindhuand S.Mukherjee, “Efficient task scheduling algorithms for cloud computing environment”. Commun. Comput. Inform. Sci., 169:79-83. DOI: 10.1007/978-3-642-22577-2_11, 2011.

18. T.He, S.Chen, H.Kim, L.Tong, K.W.Lee, “Scheduling parallel tasks onto opportunistically available cloud resources”. In IEEE CLOUD, pages 180-187, 2012.

19. S.Chen, T.He, H.Y.Starsky-Wong, K.W.Lee, and L.Tong, “Secondary job scheduling in the cloud with deadlines”. In IPDPS Workshops, pages 1009-1016, 2011.

20. D. Zhang, B.-H. Yan, Z. Feng, C. Zhang, and Y.-X. Wang, “Container oriented job scheduling using linear programming model,” in 2017 3rd International Conference on Information Management (ICIM), 2017, pp.174–180.

21. J.Li, M.Qiu, Z.Ming, G.Quan, X.Qin, Z.Gu, “Online optimization for scheduling preemptable tasks on IaaS cloud systems”. J. Parallel Distributed Computing. Elsevier, 72(5):666-677, May 2012.

22. C.C.Lin, P.Liu, J.J.Wu, “Energy-aware virtual machine dynamic provision and scheduling for cloud computing”. Proceedings of the 4th International Conference on Cloud Computing, Jul. 4-9, IEEE Xplore Press, Washington, DC., pp: 736-737. DOI: 10.1109/CLOUD.2011.94, 2011.

23. D.A.Heger, “Optimized resource allocation & task scheduling challenges in cloud computing environments”.

24. I.Foster, Y.Zhao, I.Raicu, S.Lu, “Cloud Computing and Grid Computing 360-Degree Compared”. In GCE ’08: Proceedings of Grid Computing Environments Workshop, pages 1–10, Washington, DC, USA. IEEE Computer Society.26, 42, 44, November 2008.

25. P.K.Gupta, N.Rakesh, “Different job scheduling methodologies for web application and web server in a cloud computing environment”. Proceedings of the 3rd International Conference on Emerging Trends in Engineering and Technology, Nov. 19.

26. J.Li, M.Qiu, J.Niu, W.Gao, Z.Zong, “Feedback dynamic algorithms for preemptable job scheduling in cloud systems”. Proceedings of the International Conference on Web Intelligence and Intelligent Agent Technology, Aug. 31-Sep. 3, IEEE Xplore Press, Toronto, ON, pp: 561-564. DOI: 10.1109/WI-IAT.2010.30, 2010.

27. J.Li, J.Peng, W.Zhang, “A scheduling algorithm for private clouds”. J. Convergence Inform. Technol., 6: 1-9. 2011.

28. J.Li, J.Huai, T.Wo, Q.Li, “EnaCloud : An energy-saving application live placement approach for cloud computing environments”. Proceedings of the International Conference on Cloud Computing, Sept. 21-25, IEEE Xplore Press, Bangalore, pp: 17-24. DOI: 10.1109/CLOUD.2009.72, 2009.

29. Rodrigo N Calheiros, Rajiv Ranjan, César AF De Rose, and Rajkumar Buyya. Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services. *arXiv preprint arXiv:0903.2525*, 2009.
30. Yachba Khadija, Vers une contribution dans le transport maritime de marchandises :optimisation de placement des conteneurs dans un port maritme.juin 2017.
31. Zouheyr Bouafia, Badr Benmammar, Algorithme d'ordonnancement des taches dans un environemet cloud.Vol. 5 N° 2,june2015.
32. Walid.A. Hanafy, Amr.E.Mohamed Sameh .A Salem. Novel sélection policies for container-based cloud deployement models