REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEURE ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE IBN KHALDOUN - TIARET

MEMOIRE

Présenté à :

FACULTÉ MATHEMATIQUES ET INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

MASTER

Spécialité : [Réseau Informatique]

Par :

Slimani Hassina
Slimane yassmina

Sur le thème

Equilibrage de charge dans les environnement cloud computing

Soutenu publiquement le .. / .. / 2019 à Tiaret devant le jury composé de :

Mr MOSTEFAOUI Sid Ahmed	MCB	Université Ibn Khaldoun Tiaret	Président
Mr MEBAREK Bendaoud	MCA	Université Ibn Khaldoun Tiaret	Encadreur
Mr ALEM Abdelkader	MAA	Université Ibn Khaldoun Tiaret	Examinateur

2018-2019

Remerciements

Dieu merci pour la santé, la volonté, le courage et la détermination qui nous ont accompagnés tout au long de la préparation et l'élaboration de ce travail et qui nous ont permis d'achever ce modeste travail. Le présent travail est non seulement le résultat de notre courage, sacrifice, patience et endurance mais aussi une participation de plusieurs personnes qui nous sont chères.

En second lieu, nous tenons à remercier notre encadreur Mr : (Mebarak Bendaoud), son précieux conseil et son aide durant toute la période du travail.

Nos vifs remerciements vont également aux membres du jury pour l'intérêt qu'ils ont porté à notre recherche en acceptant d'examiner notre travail Et de l'enrichir par leurs propositions. Nous adressons nos plus sincères remerciements à tous nos proches et amis, qui nous ont toujours encouragées au cours de la réalisation de ce mémoire.

Enfin, nous tenons également à remercier toutes les personnes qui ont participé de près ou de loin à la réalisation de ce travail.

Merci à tous et à toutes.

Dédicace

« Je dédie ce projet marquant de ma vie à la mémoire »

De mon père disparu trop tôt, Mais je le n'ai jamais oublié.

A ma mère qui m'a soutenu et encouragé durant ces années des études.

A mon chère frère « Belaid » qui a toujours été à mes côtés.

A ma belle-sœur « Nassima »et son mari« Amine ».

Et n'oublier pas A mon chére oncle « Abdelkrim » qui ma toujours encouragé.

Et ma binôme « Yasmina ».

A tous mes chérs amies.

A tous ma famille.

A tous qui j'aime.

S.Hassina

Dédicace

« Je dédie ce modeste travail à »

A mes trés chers parents, qui n'ont cessé de me soutenir tout au long de mon parcours d'étude, je les remercies pour leurs patience et leurs amour qui m'ont donné la force pour continuer mes études,

A ma chére sœur « Amira », mes chères fréres « Zaki et Amine »

A ma tante « Zohera » ,

A ma binome « Hassina »

A toute ma famille,

A mes amis et collègues, et tous ceux qui m'ont aide.

S.Yassmina

Notion des symboles

NIST : National Institute of Standard and technologie

SaaS : Software as a Service

IaaS : Infrastructure as a Service

PaaS : Platform as a Service

VPN : Virtual Private Network

SLA : Service Level Agreement

EC : Elément de Calcul

DB : Data Base

IDC : Internet Data Center

HPC : High Performance Computing

HTC : High Throughput Computing

VM : machine virtuel

ETC : Estimated Completion Time

ETCs : Estimated Completion Time .Semi-Consisteny

Makespan : le temps d'exécution

SLB : Simple Loading Balancing

LBE : Loading Balancing Exchange

UML : Unified Modeling Language

Hi : High

Lo :Low

Listes De Figure

Listes de tableau

Table de matiére

Introduction Général

Le Cloud Computing (appelé en français l'informatique en nuage) apparait comme une technologie à la croissance très rapide dans le monde des technologies de l'information (IT), qui se concentre sur la fourniture de services informatiques et de ressources informatiques, et à travers le monde à ses utilisateurs sur Internet. L'infrastructure et les services de cloud computing sous-jacents sont généralement détenus et gérés par un tiers, connu sous le nom de fournisseur de services cloud.

Le principal avantage du Cloud computing par rapport aux technologies informatiques existantes est le libre-service à la demande, les services de réseau étendus, l'élasticité rapide, la mise en commun des ressources et le service mesuré. La croissance du service cloud peut entraîner un ralentissement du débit, l'utilisation des ressources informatiques et, en fin de compte, réduire l'efficacité du système cloud.

Le cloud computing est la technologie émergente dans un environnement distribué composé de plusieurs centres de données, serveurs, machines virtuelles, équilibreurs de charge, etc qui sont connectés intelligemment. De plus, le nuage traite de nombreuses choses comme le stockage et la récupération de documents, le partage de contenu multimédia et le calcule scientifique.

L'ordonnancement efficace des tâches et la gestion des ressources sont un problème complexe de l'informatique distribuée, mais elles n'en sont qu'à leurs débuts malgré des recherches exhaustives ces dernières années. L'équilibrage de charge dans l'environnement informatique en nuage a un impact important sur les performances du cloud. Un bon équilibrage de la charge rend le cloud computing plus efficace et améliore la satisfaction des utilisateurs.

Le cloud reçoit les tâches des clients à un rythme rapide et l'allocation des ressources à ces tâches doit être gérée de manière intelligente. Outre les problèmes de cloud importants dans les domaines de l'ordonnancement, de l'allocation des ressources et de la sécurité, le cloud computing met en évidence d'autres problèmes urgents dont la tolérance aux pannes lors de l'exécution des tâches et des machines virtuelles. Ces types de problèmes dans un plan général sont appelés NP-difficile, ce qui signifie qu'il n'y a pas de solution exacte et pas de solution rapide.

Dans le cadre de ce mémoire, nous allons nous intéresser au problème de l'équilibrage de charge dans le cloud, car dans la littérature le problème est toujours soumis à la recherche pour améliorer les performances enregistré, et donc nous allons tenter de proposer un algorithme qui va participer, ou qui va répondre à la problématique.

Les objectifs de notre travail sont d'abord l'étude de l'ordonnancement des tâches indépendantes sur un environnement et ensuite modéliser et simuler le problème d'ordonnancement pour estimer l'influence des différents paramètres participant dans ces problème,

En plus d'une introduction et d'une conclusion, ce mémoire est composé de quatre chapitres.

- Dans le premier chapitre nous allons présenter l'exploitation des ressources dans le cloud computing, et donner ses caractéristiques, ses modèles de service, ses modèles de déploiement.
- Le deuxième chapitre, traite la problématique d'ordonnancement des tâches indépendantes, il présente en premier la théorie de la complexité ainsi un état de l'art sur l'ordonnancement.
- Le troisième chapitre, présente en détail quelque algorithme d'ordonnancement (Min-Min, SLB et LBE) et quelque algorithme pour le placement des données dans un cloud, dans cette partie nous avons aussi présenté le fonctionnement de ces algorithmes avec des exemples concret.
- Dans le quatrième chapitre nous présentons la conception et l'implémentation de l'application ainsi que l'outil qui nous a servi pour la réalisation de notre application, et enfin la présentation des différentes interfaces de notre application, et nous avons présenté une série d'expérimentations et d'évaluations des algorithmes proposés, les résultats obtenus sont analysés et discutés afin de ressortir certaines conclusions importantes.

Chapitre I

Exploitation des ressources dans Cloud Computing

Chapitre I

Exploitation des ressources dans Cloud Computing

I.1 Introduction

Depuis quelques années, un nouveau paradigme nommé cloud computing révolutionne la façon dont les entreprises et les particuliers accèdent à des ressources informatiques telles que la puissance de calcul, la capacité de stockage...etc.

Parmi les domaines de recherche qui ont prêté une attention particulière au problème d'allocation de ressources, se trouve l'économie, la recherche opérationnelle et l'informatique. De plus le problème d'allocation de ressources est pertinent à une large gamme d'applications, tel que le commerce électronique, la chaîne d'approvisionnement, les réseaux de capteurs, la composition de service grid/web, le flux de travail, et l'intégration d'application d'entreprise.

Dans ce chapitre, nous donnons une présentation générale sur le Cloud Computing, ensuit nous parlons à l'exploitation des ressources dans le Cloud Computing.

I.2 Le Cloud Computing

Cette expression vient des premiers temps de l'Internet où l'habitude était prise de dessiner le réseau comme un nuage, la traduction en français du terme Cloud Computing c'est « Informatique en nuage » ou « Infonuagique ».

Le premier nuage était construit autour du réseau (abstraction TCP/IP). Le deuxième nuage était celui des documents (abstraction du World Wide Web) [01]. Le nuage actuel, Cloud computing, est une abstraction de l'infrastructure informatique qui masque la complexité des serveurs, des applications, des données et des plates-formes hétérogènes [01] néanmoins, la confusion demeure sur exactement ce que le Cloud est et quand il est utile.

En effet, il existe de nombreuses définitions et interprétations du Cloud Computing qui sont consultables à partir de différentes sources.

Le Cloud est un processus consistant à utiliser des serveurs informatique distants à travers des réseaux Internet [02], et il permet d'accéder à vos données sur n'importe quel ordinateur relié à internet.

Le Cloud Computing a permis de réaliser un rêve de longue date qui était de transformer l'usage des ressources computationnelles sous-forme d'utilité comme c'est le cas pour l'électricité.

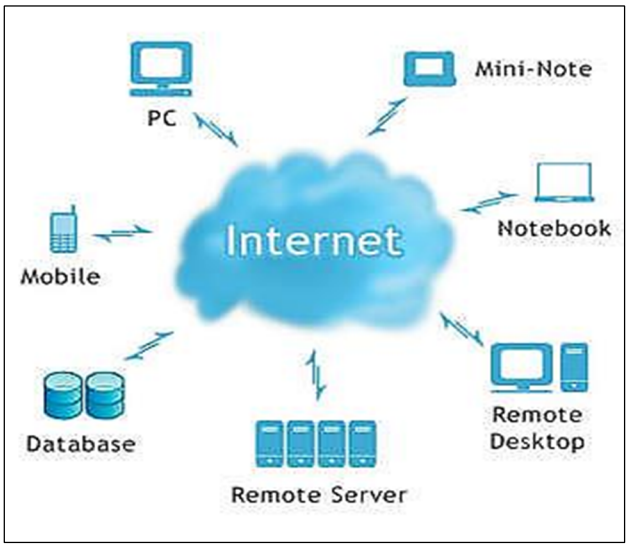


Figure I.1 Le cloud computing.

La définition opérationnelle retenue par le **NIST** (National Institute of Standard and technologie), le Cloud Computing est un modèle informatique qui permet un accès facile et à la demande par le réseau à un ensemble partagé de ressources informatiques configurables (serveurs, stockage, applications et services) qui peuvent être rapidement provisionnées et libérées par un minimum d'efforts de gestion ou d'interaction avec le fournisseur du service[01].

I.3 Caractéristiques

Le modèle cloud computing se différencie par les cinq caractéristiques essentielles suivantes **[01] :**

- **Accès aux services par l'utilisateur à la demande**

La mise en œuvre des systèmes est entièrement automatisée et c'est l'utilisateur, au moyen d'une console de commande, qui met en place et gère la configuration à distance.

- **Accès réseau large bande**

Les ressources ou les datacenters (centres de données) sont disponibles sur le réseau et accessibles via Internet pour bénéficier d'une excellente connectivité. Il sont accessibles via des mécanismes standard qui favorisent l'utilisation sur des plateformes client hétérogènes (par exemple, téléphones mobiles, tablettes, ordinateurs portables et stations de travail) [03]. Les grands fournisseurs répartissent les centres de traitement sur la planète pour fournir un accès aux systèmes en moins de 50 ms de n'importe quel endroit.

- **Réservoir de ressources (non localisées)**

La plupart de ces centres comportent des dizaines de milliers de serveurs et de moyens de stockage pour permettre des montées en charge rapides. Il est souvent possible de choisir une zone géographique pour mettre les données “près” des utilisateurs.

- **Redimensionnement rapide (élasticité)**

Une nouvelle instance d’un serveur peut être mise en ligne en quelques minutes, l’arrêt et le redémarrage en quelques secondes. Toutes ces opérations peuvent s’effectuer automatiquement par des scripts. Ces mécanismes de gestion permettent l’approvisionnement des données de manière rapide et la facturation à l’usage [03].

- **Facturation à l’usage**

Il n’y a généralement pas de coût de mise en service (c’est l’utilisateur qui réalise les opérations). La facturation est calculée en fonction de la durée et de la quantité de ressources utilisées. Une unité de traitement stoppée n’est pas facturée.

I.4 Modèles de services de cloud computing

Il existe trois modèles principaux de cloud computing, chaque modèle représente une partie différente de la pile du cloud computing

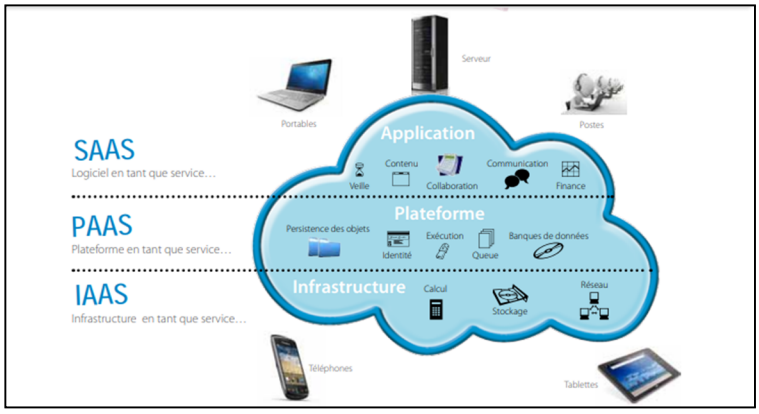


Figure I.2 Les modèles de services de Cloud.

I.4.1 Infrastructure as a Service(IaaS)

L’utilisateur loue des moyens de calcul et de stockage, des capacités réseau et d’autres ressources indispensables (partage de charge, pare-feu, cache), il a la possibilité de déployer n’importe quel type de logiciel incluant les systèmes d’exploitation.

L’utilisateur ne gère pas ou ne contrôle pas l’infrastructure Cloud sous-jacente mais il a le contrôle sur les systèmes d’exploitation, le stockage et les applications. Il peut aussi choisir les caractéristiques principales des équipements réseau comme le partage de charge, les pare-feu, etc.

L'exemple emblématique de ce type de service est Amazon Web Services qui fournit du calcul (EC2), du stockage (S3, EBS), des bases de données en ligne (SimpleDB) et quantité d'autres services de base. Il est maintenant imité par de très nombreux fournisseurs [01].

I.4.2 Platform as a Service (PaaS)

C'est une plateforme d'exécution, de déploiement et de développement des applications. La plateforme PaaS regroupe la partie développeur (client) et système (fournisseur) du Cloud Computing. Elle propose des fonctions qui privent le développeur de la gestion des utilisateurs ou des questions de disponibilité par exemple. Le développeur a ainsi uniquement besoin d'héberger son application pour qu'elle soit disponible en SaaS [04] .

I.4. 3 Softwar as a Service (SaaS)

Ce modèle de service est caractérisé par l'utilisation d'une application partagée qui fonctionne sur une infrastructure Cloud. L'utilisateur accède à l'application par le réseau au travers de divers types de terminaux (souvent via un navigateur web). L'administrateur de l'application ne gère pas et ne contrôle pas l'infrastructure sous-jacente (réseaux, serveurs, applications, stockage). Il ne contrôle pas les fonctions de l'application à l'exception d'un paramétrage de quelques fonctions utilisateurs limitées.

De bons exemples de SaaS sont les logiciels de messagerie au travers d'un navigateur comme Gmail ou Yahoo mail. Ces infrastructures fournissent le service de messagerie à des centaines de millions d'utilisateurs et à des dizaines de millions d'entreprises [01].

I.5 Modèles de déploiement du Cloud

Il existe quatre modèles de déploiement du Cloud communément utilisé : privé, public, et hybride, un modèle additionnel est le Cloud de communauté.

I.5.1 Cloud public

L'infrastructure Cloud est ouverte au public ou à de grands groupes industriels, cette infrastructure est possédée par une organisation qui vend des services Cloud, c'est le cas le plus courant, c'est celui de la plate-forme Amazon Web services déjà citée [01].

I.5.2 Cloud privé

Un Cloud privé est un ensemble des services et des ressources disponible pour un seul client par exemple une entreprise ou groupement d'entreprise (appelé organisation), il peut être géré par l'entreprise elle-même, ou ses branches, dans ce cas il s'appelle "Le Cloud privé Interne", en d'autre façons il peut être géré par un prestataire externe loué par l'entreprise, dans ce cas s'appelle "Le Cloud privé Externe", il est accessible via des réseaux sécurisés de

type VPN (Virtual Private Network). L'avantage de ce type de Cloud par rapport au Cloud public réside dans l'aspect de la sécurité et la protection des données [05].

I.5.3 Cloud hybride

L'infrastructure cloud hybride est une composition de deux ou plusieurs infrastructures cloud distinctes (privées, communautaires ou publiques) qui restent des entités uniques, mais sont liées par une technologie standardisée ou propriétaire qui permet la portabilité des données et des applications (par exemple, l'éclatement du cloud entre nuages) [06].

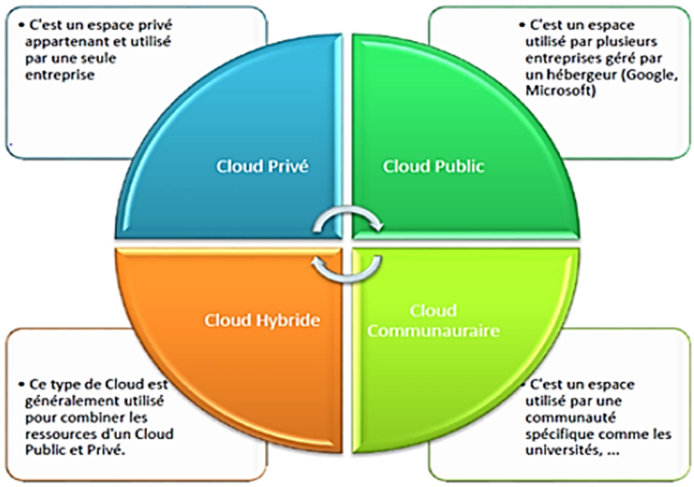


Figure I.3 Les modèles de services de Cloud de déploiement.

I.5.4 Cloud communautaire

L'infrastructure Cloud est partagée par plusieurs organisations pour les besoins d'une communauté qui souhaite mettre en commun des moyens (sécurité, conformité, etc..), elle peut être gérée par les organisations ou par une tierce partie et peut être placée dans les locaux ou à l'extérieur [01].

I.6 Eléments de Cloud Computing

Les éléments pouvant constitué le système Cloud sont les suivants :

I.6.1 La virtualisation

La virtualisation est la principale technologie dans le Cloud, elle permet une gestion optimisée des ressources matérielles en disposant de plusieurs machines virtuelles sur une machine physique, c'est une technologie qui permet une plus grand modularité dans la répartition des charges et la reconfiguration des serveurs en cas d'évolution ou de défaillance momentanée. Le principe de virtualisation permet d'intégrer les différents serveurs de façons plus flexibles pour faciliter l'utilisation, le but de la virtualisation est de faire la transparence

d'utilisation et l'efficacité d'exploitation des ressources, d'assurer le fonctionnement des différents services et la séparation entre de multiple locataire (utilisateurs) impliqués dans un matériel phtisique [07].

I.6.2 L'infrastructure

L'infrastructure informatique du Cloud est un assemblage de serveurs, d'espaces de stockage et de composants réseau organisés de manière à permettre une croissance incrémentale supérieur à celle que l'on obtient avec les infrastructures classique.

Ces composants doivent être sélectionnés pour leur capacité à répondre aux exigences d'extensibilité, d'efficacité, de robustesse et de sécurité, les serveurs d'entreprise classique ne disposent pas des capacités réseau, de la fiabilité ni des autres qualités nécessaire pour satisfaire efficacement et de manière sécurisé les accords de niveau de service SLA (Service Level Agreement) [08].

I.6.3 Le Datacenter

Un centre de traitement de données, en anglais 'datacenter ' est un site physique sur lequel sont regroupés des équipements constituants le système d'information de l'entreprise (mainframes, serveurs, baies de stockage, équipements réseaux et de la télécommunication, etc.). Il peut être interne ou externe à l'entreprise, exploité ou non avec le soutien de prestataires. Il comprend en général un contrôle sur l'environnement (climatisation, système de prévention contre l'incendie, etc), une alimentation d'urgence et redondante, ainsi qu'une sécurité physique élevée, des particuliers ou des entreprises peuvent venir y stocker leurs données suivant des modalités bien définies [09].

I.6.4 La plateforme collaborative

Une plate-forme de travail collaboratif est un espace de travail virtuel, c'est un outil, parfois sous la forme d'un site internet, qui centralise tous les outils liés à la conduite d'un projet et les met à disposition des acteurs (clients), l'objectif du travail collaboratif est de faciliter et d'optimiser la communication entre les individus dans le cadre du travail ou d'une tache [09].

I.7 Cloud Computing et sécurité

Il y a grandes préoccupations des utilisateurs sur le Cloud Computing est sa sécurité. Dans les Centres de Données Internet(IDC), les fournisseur de services offrent les grilles et les réseaux seulement, et les appareils restant doivent être préparés par les utilisateurs eux-mêmes, y compris les serveurs, le pare-feu, les logiciels, les périphériques de stockage, etc **,** la sécurité des utilisateurs peut être réfléchie dans les règles suivantes[10] :

- La confidentialité des données de stockage des utilisateurs : Le stockage des données d'utilisateur ne peuvent pas être lues ou modifiées par d'autres personnes (y compris l'opérateur).
- La confidentialité des données d'utilisateur lors de l'exécution : Les données d'utilisateur ne peuvent pas être lues ou modifiées par d'autres personnes lors de l'exécution (c.à.d. chargée dans le mémoire système).
- Le secret des données privées d'utilisateur lors du transfert à travers le réseau: Il comprend la sécurité de transfert des données Cloud Computing Internet. Ils ne peuvent pas être affichées ou modifiées par d'autres personnes.
- Authentification et autorisation nécessaire pour les utilisateurs d'accéder à leurs données : Les utilisateur peuvent accéder efficacement à leurs données et peuvent autoriser d'autres utilisateurs d'y accéder.

I.8 Avantages du Cloud Computing

Le Cloud Computing offre de nombreux avantages, d'après [11] et [12]. En voici les plus principaux :

- L'accès aux applications de n'importe où ;
- Augmentation fonctionnelle des capacités ;
- Augmentation de la puissance de calcul ;
- Réduction des couts d'infrastructure, de développement et des logiciels ;
- Accès aux ressources plus flexible ;
- Grande capacité de stockage (quasi illimitée) ;
- Gestion des mises à jour plus simple et rapide ;
- Pas de perte de données ;
- Infrastructure allouée et disponible juste à temps ;
- La possibilité offerte aux utilisateurs de réaliser des calculs parallèles.

I.9 Grid Computing vs Cloud Computing

De nombreux experts diront que le Cloud Computing dérive du grid computing. Cependant se sont deux implémentations qui ont certes des similitudes mais aussi des différences. Parmi ces déférences, il y a le fait que les grides sont déployées historiquement pour les grands travaux de calcul et ne sont pas originalement destinés pour être un modèle de service à usage utilitaire comme c'est le cas pour le Cloud Computing[13].

Le Cloud Computing et le Grid Computing sont deux termes qui portent souvent à confusion puisqu'ils se ressemblent en théorie, ils impliquent une infrastructure massive de

réseau informatique. Sur le front end, le Cloud Computing et le Grid Computing sont des concepts plus récents comparés à d'autres solutions informatiques de grande taille. Les deux concepts ont été développés à des fins de calcul distribué, c'est-à-dire de calcul d'un élément sur une grande surface, littéralement sur des ordinateurs séparés par d'autres moyens.

En fait, il existe plusieurs raisons qui motivent les spécialistes à choisir l'informatique distribuée (Computing distribué) sur l'informatique monoprocesseur [14]. Parmi ces raisons on trouve les suivants [14]:

- La raison d'opter pour l'informatique distribuée est d'offrir des ressources de calcul parallèles ou simultanées aux utilisateurs. Les requêtes ne doivent pas réellement attendre dans une file pour être traités les unes après les autres ;
- Les ordinateurs distribués utilisent tous les moments dans lesquelles votre processeur est en veille ;
- Les systèmes informatiques distribués sont constitués de nombreux systèmes, donc si l'un plante l'autre ne sera pas affecté ;
- L'évolution du modèle distribué se fait de façon rapide et efficace. Si vous avez besoin de plus de ressources informatiques, il vous suffit simplement de les brancher en installant le client sur des ordinateurs de bureau ou sur des serveurs supplémentaires.

I.9.1 La grappe d'ordinateur (Cluster)

C'est un ensemble d'ordinateurs connectés par un réseau LAN rapide et fiable pour assurer la disponibilité. Les ressources quasi-homogènes (même système d'exploitation, logiciels quasi-similaires) sont sous le contrôle d'un seul nœud appelé nœud maître [15].

I.9.2 Les grilles informatiques ou Grid

Le terme anglais Grid désigne un système distribué d'électricité. Initialement, le concept de grille partait du principe d'un tel système : les ressources d'un ordinateur (processeur, mémoire, espace disque) étaient mises à la disposition d'un utilisateur aussi facilement que l'on branche un appareil électrique à une prise électrique.

Une grille informatique est une infrastructure virtuelle constituée d'un ensemble de ressources informatiques potentiellement partagées, distribuées, hétérogènes, délocalisées et autonomes.

Une grille est en effet une infrastructure, c'est-à-dire des équipements techniques d'ordre matériel et logiciel. Cette infrastructure est qualifiée de virtuelle car les relations entre les entités qui la composant n'existent pas sur le plan matériel mais d'un point de vue logique.

D'un point de vue architectural, la grille peut être définie comme un système distribué constitué de l'agrégation de ressources réparties sur plusieurs sites et mises à disposition par

plusieurs organisations différentes [16]. Un site est un lieu géographiquement regroupant plusieurs ressources informatique administrées de maniéré autonome et uniforme. Il peut être composé d'un supercalculateur ou d'une grappe de machines (cluster).

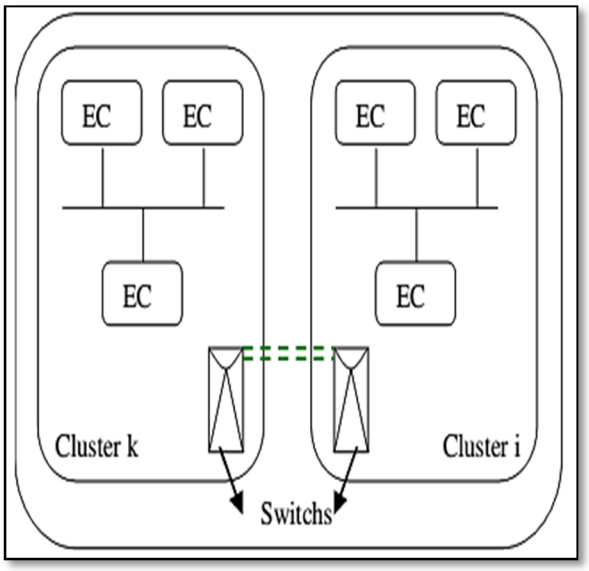


Figure I.4 Exemple de topologie d'une grille**.**

Les grilles informatiques sont caractérisées par une forte hétérogénéité et une grande dynamicité, l'infrastructure d'une grille est composée d'un nombre important de sites et de machines qui sont susceptibles de tomber en panne à tout moment.

A cela s'ajoute le fait que de nouveaux sites peuvent être ajoutés ou retirés de la grille sans trop impacter le fonctionnement de la grille. De par leurs hétérogénéités, leur gestion décentralisée et leur taille, les grilles sont des infrastructures très complexes à mettre en œuvre.

La transparence n'est assuré qu'à moitié parce qu'il faut avoir une information précise des ressources dans un site pour pouvoir distribuer les tâches n'est pas connue en général.

Les grilles de calcul paraissent comme une tendance prometteuse pour trois raisons principales [17] :

– La capacité d'utiliser de manière plus efficace un nombre donné de ressources de calcul.

– Une manière de résoudre des problèmes ne pouvant être solutionné sans une énorme puissance de calcul.

– La possibilité d'assembler et de faire coopérer les capacités de ressources, vers un objectif commun.

I.10 Partage efficace des ressources de calcul dans le nuage informatique

L'informatique en nuage ou l'infonuagique est apparue comme un nouveau paradigme capable de gérer une infrastructure informatique à grande échelle. Toutefois, la plupart des infrastructures infonuagiques existantes ne sont pas exploitées efficacement, et la sur-provision de ressources est un problème émergent [18].

Dans le domaine industriel, on utilise le plus souvent le Cloud pour des applications métiers (calcul numérique, PLM, CAO collaboration...). On parle dans ce cas de SaaS (Service as a Software) qui donne accès au bénéficiaire à un ou plusieurs logiciels préinstallés et à un espace de stockage pour ses données [18].

I.10.1 Cloud computing et calcul haute performance (High Performance Computing)

Le calcul intensif, ou calcul haute performance, est un domaine qui rassemble les outils matériels et logiciels nécessaires à l'exécution d'applications et de techniques complexes utilisées dans divers domaines académiques et industriels tels que les hydrocarbures, l'industrie pharmaceutique, l'imagerie médicale, la météorologie et les simulations physiques. Le terme HPC est aussi utilisé pour se référer aux différentes architectures parallèles existantes actuellement : les supercalculateurs, les clusters, les grilles de calcul et les processeurs graphiques (GPUs) [19].

- **Le HPC se caractérise par :**
  - Performances ;
  - Maitrise des workloads (tâches) ;
  - Gestionnaire de tâches et de ressources ;
  - Contrôle, Sécurité et confidentialité.

Dans la mesure où les besoins de débordement sur une infrastructure extérieure pour des besoins de puissance de calcul présentent un intérêt certain pour de nombreux clients HPC, les offres de Cloud ne pourraient-elles pas répondre à ce type de besoin ?

- **Le HPC présente malheureusement un certain nombre de contraintes [20] :**
  - Dans la mesure où l'adhérence fine entre les couches logicielles et le matériel est un enjeu dans le monde du calcul, l'utilisation d'une couche de virtualisation pose un problème à nombre d'experts, particulièrement en termes d'impact sur les performances ;
  - Le pilotage (scheduling de batchs de calculs) repose sur la visibilité fine de la disponibilité des ressources matérielles. Il a pour but de dédier ponctuellement les ressources à un calcul donné plutôt que de les partager (ce qui est quand même le fondement du Cloud) ;
  - Les temps de transferts de données potentiellement volumineuses entre différents clusters de calcul peuvent s'avérer prohibitifs.

- **Les avantages sont :**

- **Performance :** Tout comme dans le cas des usages Cloud, le recours à une puissance de calcul virtualisée sans limite repousse le champ des possibles. Sur des calculs hautement parallélisés, les performances peuvent être multipliées par un facteur 100.
- **Compétitivité :** Les gains obtenus vont au-delà des gains de productivité. Cela se traduit par un regain immédiat en termes de compétitivité avec des ressources qu'une entreprise n'aurait pas les moyens de s'offrir en interne.
- **Innovation :** Mieux encore, l'accès à des ressources sans limite permet d'accroître de façon exponentielle les capacités d'innovation d'une entreprise délivrée des freins de son infrastructure matérielle, logicielle et infrastructure, en donnant à ses concepteurs et développeurs les moyens de leur imagination et créativité.

I.10.2 Calcul à haut débit (high throughput computing ou HTC)

Ce type de calcul dans lequel les applications ont plusieurs tâches pouvant être traitées indépendamment les unes des autres sans que les nœuds de calcul individuels aient besoin de communiquer. Parfois, on qualifie ces charges de travail de parallélisme embarrassant *ou de* tâches par lots. Des exemples typiques incluent le rendu multimédia, le transcodage, la génomique, ainsi que la simulation et le traitement en physique des particules. Si vous devez traiter de nombreux fichiers individuels, il s'agit probablement d'une charge de travail HTC [21].

I.11 Conclusion

Le Cloud Computing est une nouvelle technologie d'utilisation des services informatique, nous pouvons être beaucoup plus flexibles et productif dans l'utilisation des ressources allouées dynamiquement.

Le Cloud Computing va continuer à évoluer comme le fondement de l'Internet du futur, ou nous seront interconnectés dans un réseau de contenus et des services.

Dans le prochain chapitre (Chapitre 02) on va traiter le problème de l'ordonnancement des taches dans les environnements cloud.

Chapitre II

L’état de l’art de l’ordonnancement et l’équilibrage de charge

Chapitre II

L'état de l'art de l'Ordonnancement et l'Equilibrage de charge

II.1 Introduction

L'ordonnancement est la programmation dans le temps de l'exécution d'une série de tâches (ou activités, opérations) sur un ensemble de ressources physiques (humaines et techniques), cherchant à optimiser certains critères, financiers ou technologiques, et en respectant les contraintes de fabrication et d'organisation. Cette définition de l'ordonnancement ne permet pas de voir le nombre important de problèmes que ce domaine comprend. En effet, l'ordonnancement est lié à plusieurs secteurs de recherches et d'activités très variés. En informatique, le choix des tâches à envoyer aux processeurs se modélise comme un problème d'ordonnancement. En production, l'ordonnancement consiste à déterminer les séquences d'opérations à réaliser sur les différentes machines de l'atelier. En gestion de projet, ordonnancer, c'est déterminer les dates d'exécution des activités constituant le projet. Chacun de ces contextes nécessite la détermination des caractéristiques propres des tâches et des ressources, le type des décisions à prendre, les modalités d'exécution des tâches par les ressources, qui déterminent des contraintes sur les décisions et aussi les différents critères à optimiser.

II.2 Ordonnancement

Un problème d'ordonnancement consiste à organiser dans le temps la réalisation de taches, compte tenu de contraintes temporelles (délai, contraintes portant sur la disponibilité des ressources requises.

En production (manufacturée, de biens, de service), on peut présenter comme un problème ou un ordonnancement constitue une solution au problème de planification, il est défini par le planning d'exécution des taches « ordre» et « calendrier » et d'allocation des ressources et vise à satisfaire un ou plusieurs objectifs [22].

II.3 Les taches

Une tache est une entité caractérisée par un nombre d'opérations à réaliser, chaque opération nécessite une durée de réalisation et des ressources. Selon les problèmes, les taches peuvent être réalisées sans interruption ou par morceaux. Plusieurs taches peuvent constituer une activité et plusieurs activités peuvent définir un processus [23].

II.3.1 Ordonnancements de taches indépendants

De nombreux études ont été effectués pour l'ordonnancement de taches indépendante [24,25-26] que ce soit en ciblent des plateformes homogènes ou hétérogènes, les taches sont le dénominateur commun des problèmes d'ordonnancement, leur définition n'est ni toujours immédiate, ni toujours triviale, une tache est caractérisée par : une durée, une date de disponibilité, une date de fin au plus tard et une quantité de ressources.

Est-ce qui concerne l'ordonnancement de taches parallèle indépendantes [27,28-29] les études ont été effectuées essentiellement pour les plates-formes homogènes.

Les algorithmes d'ordonnancement de taches indépendantes sur les plates-formes hétérogènes visent général à minimiser le temps de complétion de l'application ordonnancée ou à maximiser le débit de la plate-forme utilisés.

Le problème général à l'ordonnancement de **n** taches indépendantes sur **m** processeurs non corrélés visant à minimiser le makespan est un problème NP-difficile [30].

Dans le mode d'ordonnancement en ligne, une tache est ordonnancée sur un des processeurs de la plate-forme dès qu'elle arrive dans le système. En ce qui concerne l'ordonnancement par lot, les taches ne sont pas ordonnancées aussitôt qu'elles arrivent.

Elles sont d'abord collectées dans un ensemble de taches qui sera ordonnancé, à un instant prédéterminé [31].

II.3.2 Ordonnancement des taches dépendantes

Les études sur l'ordonnancement d'applications composées de taches dépendantes visent essentiellement à minimiser le makspan de ces applications. Il existe quelques travaux sur l'ordonnancement de graphes de taches qui interagissent, notamment sur les plat-formes hétérogènes [30,32]. Mais la plupart des travaux concernent l'ordonnancement de graphes de taches avec contraintes de précédence, ce type d'applications étant plus répondu. Le problème d'ordonnancement de DAGs quelconques sur plusieurs processeurs avec l'objectif de minimiser la makspan est un problème NP-difficile, même pour le DAGs de tache séquentielles [33], la plupart des solutions d'ordonnancement proposées sont donc des heuristiques.

L'ordonnancement de DAGs composés de taches séquentielles a été largement étudié aussi dans le cas particulier des plat-formes homogènes que dans le cas général des ensembles de processeurs hétérogènes. Les algorithmes obtenus peuvent être classés en diverses catégories.

II.4 Les contraintes

Expriment des restrictions sur les valeurs qui peuvent prendre des variables de décision on distingue :

- **Des contrainte temporales** : de temps alloués issus généralement de gestion et relatives aux dates limite des taches ou a la durée totale.
- **Des contrainte de ressources** : les contraintes d'utilisation des ressources qui exprime la nature et la quantité ainsi que les caractéristique d'utilisation ces moyen.
- Des contraintes disponibilité des ressources [34].

II.5 Le Makespan

Dans les systèmes temps réels chaque sortie peut être fonction de plusieurs entrés. Le temps total d'exécution que l'on appellera dans la suite « makespan » car c'est le terme utilisé dans la littérature, reflète le temps qui s'écoule entre la date de début d'exécution de la première tâche exécutée et la date de fin de la dernière tache exécutée[35].

II.6 Typologie des algorithmes d'ordonnancement

II.6.1 Ordonnancement Hors-ligne

Un ordonnancement hors ligne et pré-calculé avant l'exécution puis est exécuté avec ou sans préemption, l'algorithme construit la séquence complète de planification des taches sur la base de tous les paramètres temporels des taches, séquence connue avant l'exécution. Très efficace mais cette approche statique est rigide ne s'adapte pas aux changements de l'environnement [36].

II.6.2 Ordonnancement En ligne

Un ordonnancement en ligne décide dynamiquement avec ou sans préemption de l'exécution des taches, capable à tous instant de l'exécution d'une application de choisir la prochaine tache à ordonnancer, en utilisant les informations des taches déclenchées à cet instant, ce choix peut être remis en cause par l'occurrence d'un nouvel événement, cette approche dynamique offre des solutions moins bonnes des surcouts de mise en œuvre, les avantages : permet l'arrivée imprévisible des taches, autorise la construction progressive de la séquence d'ordonnancement[36].

II.7 Modèle d'ordonnancement

Nous introduisons maintenant les différentes contraintes envisageables pour l'ordonnancement des taches.

- **Ordonnancement préemptif** : l'ordonnanceur peut interrompre une tache en cours d'exécution au profil d'une autre tâche.

- **Ordonnancement non préemptif** : l'ordonnanceur doit attendre la fin d'exécution de la tâche pour réordonnancer les taches [37].

II.8 Technique d'ordonnancement des taches

Les techniques ont pour objectif de répondre au mieux besoins exprimés par un client, au meilleur cout et dans les meilleurs délais, en tenant compte des différentes contraintes.

Il faut bien retenir qu'un seul algorithme d'ordonnancement universel, qui apporterait les meilleures performances pour tout type d'application ou matériel, n'existe pas. L'étude des différents algorithmes d'ordonnancement et leur classification montrent que ces dernier sont incapables de fournir des solutions optimales dans un temps polynomial si on prend en compte toutes les contraintes qu'on peut rencontrer dans les applications.

La plupart de ces approches opèrent sous la supposition classique que les données des problèmes d'ordonnancement sont tout à fait connus : l'ensemble des taches à ordonnancer et temps fini et détermine a priori les durées opératoires, les dates de début au plus tôt et de fin au plus tard des taches sont fixées et stables et les tendances des ressources sont connues à tout instant.

Cette hypothèse permet en effet de traiter un problème de façon déterministe, ce qui simplifie largement la résolution, celle-ci demeurant toutefois dure dans la plupart des cas.

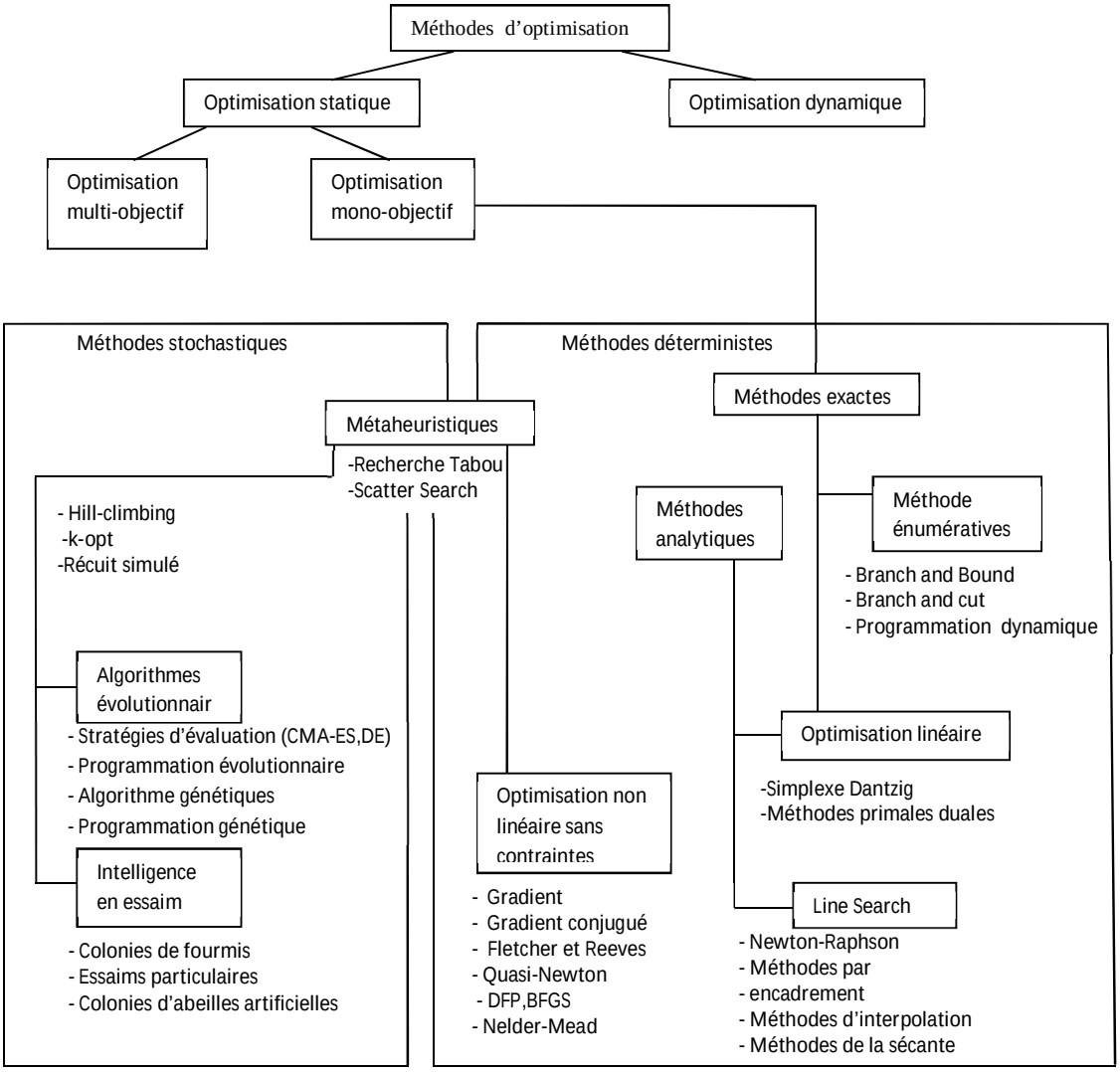


Figure II.1 Classification possible des différents types des méthodes d'optimisation.

II.9 Architecteurs d'ordonnancement

Trois architecteurs ont été avancés : centralisées, Hiérarchique et multi-agents.

II.9.1 Les Heuristiques

Une méthode heuristique est une technique utilisée pour trouver des solutions prêtes à être optimales pour un cout raisonnable. Les heuristiques ne garantissent pas la fiabilité ou l'optimalité des solutions qu'ils produisent. En exploitant la structure d'un problème, des solutions raisonnables peuvent être décelés. Une approche heuristique devrait être utilisée lorsque des méthodes de résolution exactes nécessitent des calculs énormes impraticables. Les problèmes NP-complet sont généralement abordés avec des techniques heuristiques.

Le théorème Woltper connu sous le nom de **no free lunch**, montre qu'il n'existe pas d'heuristique qui soit meilleure qu'une autre : pour toute heuristique, il existe au moins une

instance du problème pour laquelle une autre méthode lui une supérieure. L'importance et la portée pratique de ce théorème suscitent des contradictoire, néanmoins du point de vue dans les méthodes de résolution, une certaine connaissance ou information sur l'instance ou le problème à résoudre.

Les heuristiques sont également des règles ou des stratégies utilisées pour améliorer l'efficacité d'une autre méthode ou même d'autres approches heuristiques. Braun et al [38] fourniture comparaison de 11 heuristiques statiques d'ordonnancement dans des EHC, une gamme d'approches heuristiques de construction sont comparées, suivant la politique qu'elle suit chaque heuristique, on peut les regrouper dans des catégories suivantes [38]:

- **La construction** : les solutions finales sont générées par l'ajout d'une composante de la solution à la fois.
- **L'amélioration** : une séquence de transformations est appliquée à partir d'une solution de départ pour l'améliorer, cela comprend la classe importante des heuristiques de recherche.
- **Le partitionnement** : le problème est divisé en sous-problèmes qui sont résolu de façon indépendante, les solutions fragmentées sont combinées ensemble pour former une solution finale au problème originale.
- **La détente**: Certaines contraintes sont assouplies afin de pour former de rendre le problème plus facile à résoudre, des transformations doivent être réalisées sur des solutions irréalisables pour les rendre possibles.
- **La restriction** : L'espace de solution est limité pour rendre le problème plus facile à résoudre.

II.9.2 Les méta-heuristiques

Les méta-heuristiques constituent l'une des trois classes des méthodes d'ordonnancement (les autres étant les méthodes exactes et les heuristiques).

Elles sont adaptables à un grand nombre de problèmes sans changement majeur de l'algorithme, il existe un grand nombre de méta heuristique différent [32], leur permettant d'être adaptées à une large gamme des problèmes différents, une des façons de classer les méta-heuristiques est distinguer celles qui utilisent des méthodes de recherche locale, qui ne manipulent qu'une seul solution à la fois, de celles qui exploitent une population de solutions. Les méta-heuristiques à recherche locale évolué à chaque itération une solution sur l'espace de recherche, c'est pourquoi la notion de voisinage est primordiale.

Parmi les méta-heuristiques avec recherche locale, nous pouvons citer le recuit simulé et la recherche tabou.

Il existe principalement deux types de méta-heuristique à base de population : les colonies de fourmis et les algorithmes génétiques.

a . Les algorithmes génétiques

Les algorithmes génétiques (AG) sont des algorithmes d'optimisation stochastique fondés sur les mécanismes de la sélection naturelle et de la génétique, ils ont été adaptés à l'optimisation par John Holland [33], également les travaux de David Goldberg ont largement contribué à la enrichir [39,40], contrairement aux méthodes de recherche locale qui se font intervenir à chacun des étapes du processus de recherche.

Un algorithme génétique AG reproduit l'évolution naturelle d'organismes vivants, génération après génération, en respectant les phénomènes l'hérédité, dans une population, ce sont les individus les mieux adaptés au milieu qui survivront et pourront donner une descendance.

Le fonctionnement des AGs est extrêmement simple, on part d'une population de solutions potentielles initiales, arbitrairement choisis, on évalue leur performance relatives sur la base de ces performances on crée une nouvelle population de solution potentielles en utilisant des opérations évolutionnaires simples : la sélection, le croisement et la mutation.

Quelques individus se reproduisent, d'autres disparaissent et seuls les individus les mieux adaptés sont supposés survivre, on recommence ce cycle jusqu'à ce qu'on trouve une solution satisfaisante. En effet, l'héritage génétique à travers les générations permet à la population d'être adaptée et donc répondre au critère d'optimisation, la figure 2.2 illustre les principales étapes d'un algorithme génétique. Un algorithme génétique recherche le ou les extrema d'une fonction définie sur un espace de données.

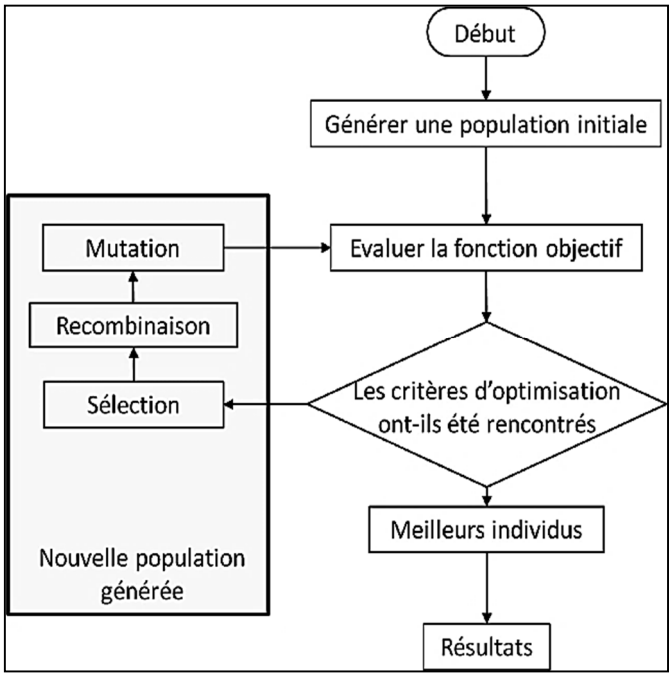


Figure. 2.2- Fonctionnement d'un algorithme génétique.

b. Les algorithmes de colonies de fourmis

Les algorithmes de colonies de fournis ont été proposés par C.Dorigo et Maniezzo en 1992 [41] et appliquées la première fois au problème du voyageur de commerce, ce sont des algorithmes itératifs à population ou tous les individus partagent un savoir commun qui leur permet d'orienter leurs futurs choix et d'indiquer aux autre individus des choix à suivre ou à éviter. Le principe de cette méta-heuristique repose sur le comportement particulier des fourmis, elles utilisent pour communiquer une substance chimique volatile particulière appelée phéromone grâce à une glande située dans leur abdomen. En quittant leur nid pour explorer leur environnement à la recherche de la nourriture, les fourmis arrivent à élaborer des chemins qui s'avèrent fréquemment être les plus courts pour aller du nid vers une source de nourriture, chaque fourmi dépose alors une quantité de phéromones sur ces pistes qui deviendront un moyen de communication avec leur congénères, les fourmis choisissent ainsi avec une probabilité élevée les chemins contenant les plus fortes concentrations de phéromones à l'aide des récepteurs situés dans leur antennes.

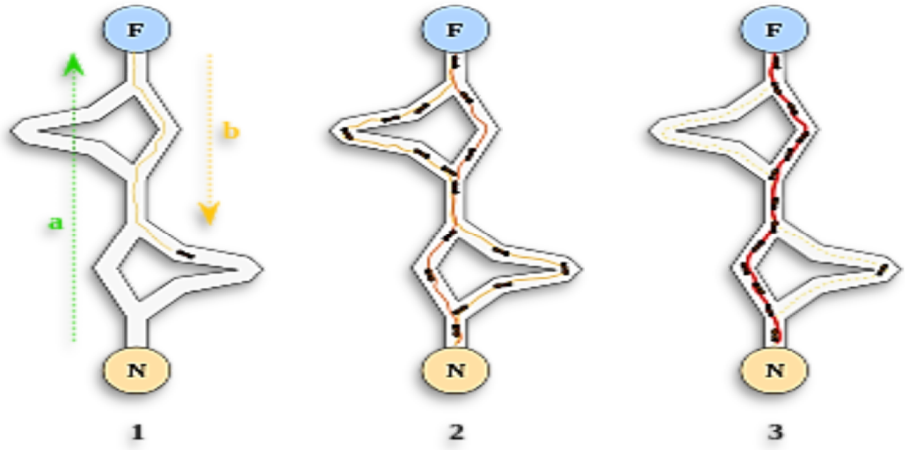


Figure II.3 sélection des branches les plus courts par une colonie de Fourmis.

II.10 Approches d'équilibrage de charge

Le problème de l'équilibrage étant un problème relativement ancien, beaucoup d'approches ont été proposées pour le résoudre. Casavant et Kuhl ont défini une taxonomie largement adoptée par la communauté scientifique dont les principales classes sont [42] :

II.10.1 Approche statique Vs. approche dynamique

Dans une approche statique, les tâches sont assignées aux machines avant l'exécution de l'application qui les contient. Les informations concernant le temps d'exécution des tâches et les caractéristiques dynamiques des machines sont supposées connues a priori. Cette approche est efficace et simple à mettre en œuvre lorsque la charge de travail est au préalable suffisamment bien caractérisée. Dans une approche dynamique, l'assignation des tâches aux machines se décide durant la phase d'exécution, en fonction des informations qui sont collectées sur l'état de charge du système. Ceci permet d'améliorer les performances d'exécution des tâches mais au prix d'une complexité dans la mise en œuvre de cette stratégie, notamment en ce qui concerne la définition de l'état de charge du système, qui doit se faire de manière continue.

II.10.2 Approche centralisée Vs. approche distribuée

Dans une approche centralisée, un site du système est choisi comme coordinateur. Il reçoit les informations de charge de tous les autres sites qu'il assemble pour obtenir l'état de charge global du système. Dans le cas d'une approche distribuée, chaque site du système est responsable de collecter les informations de charge sur les autres sites et de les rassembler pour obtenir l'état global du système. Les décisions de placement de tâches sont prises

localement, étant donné que tous les sites ont la même perception de la charge globale du système.

II.10.3 Approche source-initiative Vs. receveur-initiative

L'approche source initiative est appliquée lorsqu'un site, appelé source, détecte qu'il a une surcharge de travail et qu'il cherche à transférer le surplus vers un site faiblement chargé. L'approche receveur-initiative s'applique lorsqu'un site faiblement chargé, appelé receveur, demande à recevoir tout ou partie du surplus des sites surchargés.

II.11 Le système d'équilibrage de charge

Un système d'équilibrage de charge est composé de deux éléments essentiels: les politiques et les mécanismes. Les politiques considèrent l'ensemble des choix à effectuer pour distribuer une charge de travail alors que les mécanismes réalisent physiquement la répartition de la charge et fournissent les informations exigées par les politiques. La figure 2.2 illustre la décomposition arborescente d'un système d'équilibrage de charge [42] :

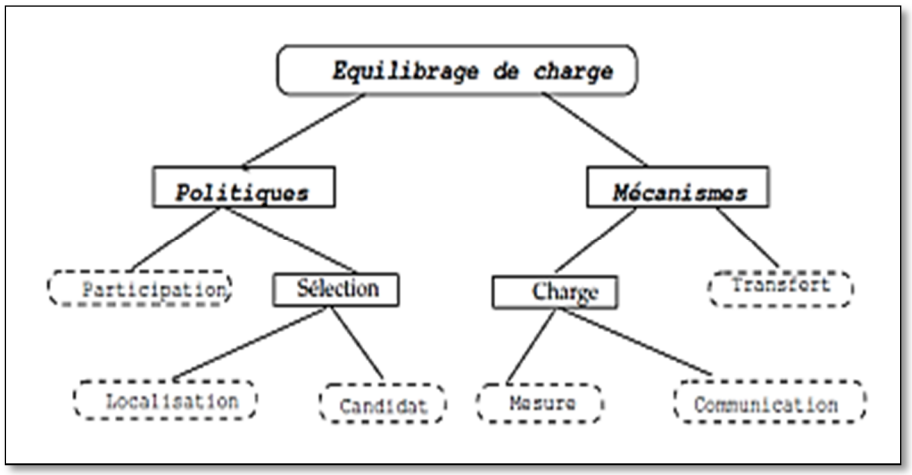


Figure II.4 Composants d'un système d'équilibrage de charge

- **Les Politiques et les mécanismes**

- **Politique de participation** : Le but de cette politique consiste à déterminer si un site est dans un état approprié pour participer à un transfert de tâches comme source (site surchargé) ou comme receveur (site sous-chargé).
- **Politique de sélection de la localisation** : Cette politique est responsable de trouver, pour un site donné, un partenaire (source ou receveur), une fois que la politique de participation a décidé que ce site était soit source, soit receveur.

- **Politique de sélection des tâches à transférer** : Une fois que les politiques de participation et de localisation ont décidé qu'un site S_i est source et qu'un autre site S_j est receveur, cette politique est responsable du choix des tâches à transférer de S_i vers S_j.
- **Mécanisme de mesure de la charge** : Dans toute approche d'équilibrage de charge, une des difficultés majeures est celle qui consiste à évaluer la mesure de la charge d'un site. Dans la plupart des travaux existants, c'est la longueur de la file d'attente qui détermine la charge d'un site. Certains auteurs préconisent comme indicateur de charge, une combinaison entre la longueur de la file d'attente CPU, celle des Entrées/Sorties et l'occupation mémoire. Dans le cas des grilles de calcul, il est nécessaire de tenir compte aussi de l'hétérogénéité des ressources et des réseaux de communication pour mesurer la charge d'un site.
- **Mécanisme de définition de la charge :** Ce mécanisme essaie de définir la charge globale d'un système en collectant les informations de charge (partielles) sur l'ensemble ou une partie des sites du système. Il faudra alors définir les méthodes selon lesquelles l'information de charge est collectée puis diffusée aux sites

II.12 Equilibrage de charge

L'équilibrage de charge (parfois appelé répartition de charge ou en anglais load balancing) consiste à distribuer une tâche à un pool de machines ou de périphériques afin :

- De lisser le trafic réseau, c'est-à-dire de répartir la charge globale vers différents équipements ;
- De s'assurer de la disponibilité des équipements, en n'envoyant des données qu'aux équipements en mesure de répondre, voire à ceux offrant le meilleur temps de réponse.

Ce type de mécanisme s'appuie sur un élément, appelé répartiteur de charge (en anglais load balancer) chargé de distribuer le travail entre différentes machines[43].

Il existe plusieurs façons de mettre en œuvre le load balancing :

- Grâce à un commutateur de niveau 4 ;
- Grâce à un serveur utilisant un algorithme de type Round-Robin.

II.13 Les algorithmes d'Equilibrage de charge

Les algorithmes d'équilibrage de charge servent à distribuer les requêtes des utilisateurs qu'arrivent à un centre de données, entre les machines virtuelles selon des techniques propres pour chaque algorithme, afin de garantir une égalité au niveau des nombres des taches allouées à chaque machines [44].

II.14 Conclusion

Dans ce chapitre, nous nous sommes intéressé au problème d'ordonnancement en essayant de donner des concepts généraux sur ce dernier, puis nous avons essayé de donner une description aux quelques approches antérieures d'ordonnancement de tache de calcul.

L'ordonnancement dans les environnements de calcul hétérogène est un problème NP-complet, par conséquent, pour faire face aux difficultés dans la pratique, plusieurs chercheurs dans le domaine d'informatique distribué ont proposé et développé des approches heuristiques et méta-heuristiques.

Dans le chapitre suivant, nous nous intéressons à la description et l'implémentation de trois algorithmes min-min et Single Load Balancing (SLB) et Load Balancing Exchange (LBE), et nous avons traité des solutions apportés au problème d'équilibrage de charge dans le Cloud.

Chapitre III

Ordonnancement et équilibrage de charge dans le cloud computing

Chapitre III

Ordonnancement et équilibrage de charge dans le cloud computing

III.1 Introduction

Le problème d'ordonnancement des tâches est un problème NP-complet, il consiste à allouer des ressources à ces taches et à établir un plan d'exécution sur ces ressources.

L'informatique en nuage (cloud computing) est un paradigme qui propose à l'utilisateur de délocaliser ses ressources de calcul et ses ressources de stockage. Les ressources qui servent de support au cloud computing sont généralement des clusters car ceux-ci sont plus homogènes et disponibles que les grilles de calcul.

En raison de la taille énorme des données, de la bande passante limitée du réseau, le placement des données et le temps d'exécution sur les centres de données est devenu un enjeu dans l'équilibrage de charge dans le Cloud.

Il s'agit d'un nouveau modèle d'informatique d'affaires qui répartira les tâches sur le pool de ressources constitué du grand nombre d'ordinateurs, de stockage dispositifs système parallèle et distribué consistant en une collection de ordinateurs virtuels interconnectés et virtuels provisionnés.

Les utilisateurs peuvent accéder à la puissance de calcul, à l'espace de stockage et à une variété de services logiciels. La planification multitâches est la NP-Complete problèmes par conséquent, la planification des tâches du Cloud Computing revêt une grande importance.

Dans ce chapitre, nous proposons quelques algorithmes d'ordonnancement des taches dans un environnement distribués et qui sont applique généralement dans le cas du Cloud Computing.

Avant de passer à la présentation des algorithmes d'ordonnancement des taches et des méthodes d'équilibrage de charge pour minimiser le temps d'exécution, cette partie contient une présentation sélective des méthodes développées dans la littérature, traitant l'équilibrage de charge dans le Cloud (pour les données).

L'équilibrage de charge peut être atteint soit par placement de données ou de fragments de données, soit par placement de taches ou de machine virtuel (VM).

III.2 Placement de donnée

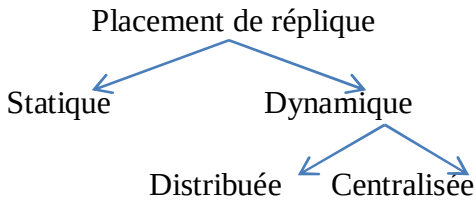
En raison de la taille énorme des données, de la bande passante limitée du réseau, le placement des données sur les centres de données est devenu un enjeu dans l'équilibrage de charge dans le Cloud. Les ensembles de données traités par un même calcul (une tache) doivent être placés sur le même data center.

Les algorithmes statiques de placement de données requirent une connaissance complète des statistiques sur les charges de travail tel que le temps d'exécution des services, et le taux d'accès. Les algorithmes dynamiques de placement de taches génèrent des schémas d'allocation en ligne pour s'adapter à différents modèles de charge de travail sans connaissance préalable des requêtes futurs. Les stratégies de placement de données dynamique mettent à jour le placement potentiellement à chaque requête.

III.2.1 Classification de placement et sélection de réplique

Pour mieux présenter notre état de l'art nous avons choisi de suivre la classification proposé par Grace et Manimegalai [45] qui propose dans leur article une étude sur les stratégies de placement et de sélection dynamique des répliques pour un environnement de grille de données.

- **Le classement est comme suite** :



- **Réplication de données statique** : Le nombre de réplique et leur emplacement sont prédéfinie. L'avantage est la simplicité de l'implémentation et le cout minimum, mais inadéquat a un environnement variable.
- **Réplication de données dynamique** : La décision du nombre de réplique et de leur emplacement est prise en ligne (temps réel) selon la demande des utilisateurs, la capacité de stockage et la bande passante.
- **Distribuée** : selon la topologie ou l'architecture de la grille de calcul, les réplications peuvent être crées par le nœud central et d'autres nœud sélectionnés.
- Centralisée : selon l'importance des données, la qualité de service, équilibrage de charge et minimisé le nombre de réplique. Les répliques sont criés uniquement dans le nœud central (head node).

III.2.2 Placement statique

Soit **D** ={d_1, .., d_n} : les ensembles de données ; **S** ={S_1, .., S_l} : les data centers .

La matrice de placement des ensembles de données **D** sur les data center **S** est donnée par $\beta = [\beta_{jk}]_{n\times 1}$ ou,

$$\beta_{jk} = \begin{cases} 1 \; si \; d_j \; est \; sur \; s_k \\ 0 \quad sinon \end{cases}$$

Le nombre de placement de données durant l'exécution de toutes les taches dans le système est donné par : Γ(B). Donc l'objectif est de trouver la matrice B* optimal qui minimise Γ(B).

Un algorithme de recherche exhaustif est un moyen direct pour rechercher la matrice de placement optimale. Il calcule toutes les matrices possibles de placement de données B, puis parcourt pour trouver le plus petit Γ(B), à ce point la matrice de placement est la solution optimale. Cependant, la complexité de calcul de l'algorithme de recherche exhaustive est très élevée, ce qui se rapproche de (l^n). Dans un système de cloud computing distribué, le nombre de jeux de données **n** est si important que la complexité du calcul est insupportable pour le système, qui plus est, certaines conditions de contrainte, telles que la limitation de la capacité de stockage, font que la résolution du problème de placement est un problème NP-difficile.

L'algorithme de recherche exhaustive n'est donc adéquat que lorsque le nombre de jeux de données est petit [46].

L'algorithme de Monte Carlo est basé sur la théorie des probabilités et les méthodes statistiques. Dans le cas du Big Data basé sur l'algorithme de Monte Carlo, un certain nombre de matrices B est généré aléatoirement, puis l'ordonnancement des données entre les data centers est calculé sur chaque matrice d'échantillon B, pour obtenir la matrice de placement avec l'ordonnancement minimum des données. Par rapport à l'algorithme de recherche exhaustive, la complexité de calcul de l'algorithme de Monte Carlo est améliorée, cependant, les matrices B sont peu réparties dans l'espace de solution, l'efficacité de recherche de l'algorithme de Monte Carlo n'est toujours pas élevée [46].

III.2.3 Placement dynamique

Xu et al. [46] proposent un algorithme génétique et test l'efficacité de ce dernier dans le placement de données, en le comparant à d'autres algorithmes. L'algorithme génétique est un algorithme adaptatif de recherche et d'optimisation basé sur la mécanique de la sélection naturelle et de la génétique naturelle.

Xu et al. [46] ont comparé leur algorithme a l'algorithme de recherche exhaustive et l'algorithme Monte carlo et ils ont remarqué qu'avec un nombre petit de jeux de données les

trois algorithmes ont le même résultat, avec l'augmentation du nombre de jeux de données, l'algorithme de recherche exhaustive est devenu irréalisable en raison de la complexité du calcul. Il est coûteux en temps pour l'algorithme de Monte Carlo de trouver une matrice de placement de données optimale approximative lorsque le nombre d'ensembles de données ou de centres de données augmente dans une certaine mesure.

Un système de base de données distribuée (DDBS) est l'outil le plus demandé pour traiter les données volumineuses, les performances DDBS dépendent essentiellement de la procédure avec laquelle DDBS est configuré (fragmenté et réparti sur tous les sites de son réseaux). Les données doivent être soigneusement divisées afin qu'une correspondance entre les données ciblées et les requêtes considérées soit satisfaite au maximum. La principale motivation pour l'allocation de données est de stocker des fragments de données sur plusieurs sites de maniéré à minimiser les coûts de transmission globaux au fur et à mesure qu'un ensemble de requêtes est en cours de traitement. La fragmentation horizontale est l'opération de fragmenter les données afin de minimiser une fonction objectif donnée par l'équation (3).

La première équation est définie pour être utilisée pour mesurer les coûts encourus lorsque des requêtes de récupération distribuées sont en cours de traitement. L'équation (2) va mesurer les coûts engendrés à la suite de l'exécution de requêtes de mise à jour distribuées sur DDBS. Les coûts de transmission au total seraient donc calculés avec précision à l'aide de l'équation (3).

$$TC_1 = \sum_{j=1}^{m}\sum_{i=1}^{m}\sum_{K=1}^{q}(1-X_{kj})*(QRM_{kj})*F_{size}*CMS_{ij} \quad (1)$$

$$TC_2 = \sum_{j=1}^{m}\sum_{i=1}^{m}\sum_{K=1}^{q}(1-X_{kj})*(QUM_{kj})*F_{size}*CMS_{ij} \quad (2)$$

$$TC_{total} = TC_1 + TC_2 \quad (3)$$

Ou CMS est la matrice de coûts entre sites (CSM) ou la matrice de coûts entre grappes (Cluster en anglais) de sites (CCM), F_{size} représente la taille du fragment considéré, et X_{ij} est une variable binaire destinée à indiquer l'allocation du fragment sur les sites.

QRM : fréquence de requête de retrait des données sur site, et

QUM : fréquence de requête de mise à jour (update) des sites.

III.3 Placement de tache (VMs)

Comme pour le placement de données, le placement de taches ou de VMs peut être classé en placement statique et dynamique.

III.3.1 Critère de comparaison

Xu et al. [47] présenté des classifications basées sur une étude complète sur les algorithmes d'équilibrage de charge VM existants qui sont analysés et classés dans le but de fournir une vue d'ensemble de la caractéristique des algorithmes associés. Plusieurs algorithmes d'équilibrage de charge VM sont étudié ; il montre pour chaque algorithme les approches utilisées pour évaluer les algorithmes, ainsi que la configuration dans la quel s'est déroulé l'expérience et les améliorations des performances pour l'équilibrage de la charge des machines virtuelles.

Une classification des algorithmes selon les critères de : Allocation VM (Dynamique, statique), Uniformité de VM (Homogène, Hétérogène), Type de ressource VM (CPU, mémoire…etc.) et Stratégie d'optimisation (Heuristique, Méta-heuristique, Hybride) est effectué.

Les métriques de comparaison des algorithmes d'´équilibrage de charge utilisé sont : Variance de charge et écart-type d'utilisation, Makespan (durées d'exécution, ou cycle de vie), Nombre d'hôtes surchargé, Pourcentage de toutes les machines virtuelles à placer dans l'hôte, Entropie d'équilibre quadratique, Débit, Ecart-type des connexions, Niveau de déséquilibre moyen, Capacité-makespan, Score de déséquilibre, Ecart-type de ressource restante, Nombre de migrations et Violations SLA.

III.3.2 Placement statique

Dans l'algorithme Round Robin, les processus sont partitionnés entre tous les processeurs de telle sorte que la charge de travail entre les processeurs est répartie également.

De plus, le processus distinctif n'a pas le même temps de traitement. Parfois, certains des nœuds peuvent être chargés vigoureusement et d'autres sont légèrement chargés dans les serveurs Web où la requête http est de nature comparative et transmise de la même façon qu'un algorithme Round Robin est utilisé [48].

III.3.3 Placement dynamique

Asha et al. [48] ont aussi étudié les algorithmes suivant :

Equally spread current execution algorithm (Algorithme d'exécution courant étalé de manière égale), est un calcul dynamique d'ajustement de charge. Il décide de la priorité en vérifiant la durée d'exécution du processus. Cet algorithme disperse la charge de manière aléatoire en vérifiant d'abord la taille du processus et en échangeant ensuite la charge avec une machine virtuelle qui est légèrement chargée. L'équilibreur de charge répartit la charge sur des nœuds distinctifs [48].

Throttled Load Balancing Algorithm (Algorithme d'équilibrage de charge étranglé) fonctionne comme suit : le gestionnaire de tache dispose de l'emploi du temps de chaque machine virtuelle unique, et donc alloue la tache souhaité donné par le client à la machine sur le principe de la taille et de l'accessibilité de la machine, si aucune machine virtuelle n'est accessible pour exécute les taches, le gestionnaire de tache va mettre la demande en file d'attente [48].

Ant Colony Optimization Algorithm (Algorithme d'optimisation des colonies de fourmi) travaille sur le comportement des vraies fourmis. L'objectif principal de l'optimisation est de trouver le chemin optimisé de la source à la destination. Les fourmis tout en cherchant la nourriture lâchent des composants spéciaux appelés des phéromones. Sur la base de ces phéromones, les prochaines fourmis suivront le même chemin. L'intensité des phéromones est constituée de divers facteurs tels que la qualité de la nourriture, la distance alimentaire, etc. Les chemins qui consistent en l'intensité de phéromone la plus élevée sont considérés comme étant de la distance la plus courte entre la source et la destination [48].

Honey bee foraging algorithm (algorithme de recherche des abeilles a miel) est basé sur le comportement des abeilles domestiques, lorsqu'une VM sous-chargée est affecté a une tâche, la liste des taches et la charge est mises a jour et informe les autres VM. La prise en compte de la priorité des taches améliore le débit [49, 50].

Pareto based fruit fly optimization algorithm (Algorithme d'optimisation pareto basé sur la mouche des fruits) fonctionne comme suit : D'abord, une heuristique initialise la population. Deuxièmement, un opérateur de réaffectation de ressources est utilisé pour générer des solutions. Troisièmement, un opérateur de recherche basé sur le chemin critique est utilisé pour améliorer la capacité d'exploitation [49, 51].

Multi objective Scheduling cuckoo algorithm (algorithme d'ordonnancement multi-objectif du coucou) reproduit le comportement de reproduction des coucous, ou chaque individu recherche le nid le plus approprié pour pondre un œuf (solution de compromis) afin de maximiser le taux de survie de l'œuf. La théorie des ensembles flous est utilisée pour créer le domaine de recherche des appartenances floues qui est constitué de toutes les solutions de compromis possibles. L'algorithme recherche la meilleure solution de compromis dans le domaine de recherche floue en réglant simultanément les variables de la frontière de conception floue. L'ajustement des variables de conception floue élimine l'exigence d'expertise nécessaire pour définir ces variables [49].

III.4 Présentation des algorithmes Min-Min, SLB et LBE

III.4.1 Formalisation du problème d'ordonnancement

Beaucoup d'approches ont été proposées pour résoudre le problème d'ordonnancement dans différentes plates-formes. Avec l'apparition de l'informatique en nuage (ou Cloud Computing), plusieurs algorithmes d'ordonnancement ont été adaptés pour ce type de plate-forme étudient le problème d'ordonnancement des tâches dans les Clouds et proposent différentes techniques pratiques mises en œuvre pour résoudre ce problème. 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. Dans cette section, nous allons présenter quelques algorithmes d'ordonnancement des tâches dans les Cloud. Ces algorithmes se sont focalisés essentiellement sur la rapidité d'exécution des tâches.

Un problème d'ordonnancement est caractérisé par deux ensembles :

- Un ensemble de n tâches $\mathbf{T} = \mathbf{T}_1, \mathbf{T}_2 \ldots\ldots \mathbf{T}_n\}$ et
- Un ensemble de m machines (resources) $\mathbf{M} = \{\mathbf{M}_1, \mathbf{M}_2 \ldots. \mathbf{M}_m\}$.

Dans ce cas, une machine est considérée comme une ressource. Ordonnancer c'est assigner les taches de l'ensemble T aux machines de l'ensemble M, dans l'ordre de compléter toutes les tâches sous des contraintes imposées.

Le problème d'ordonnancement est défini par les caractéristique des machines, celles des taches et par le critère d'optimalité.

Pour aborder ce problème, nous utilisons une matrice ETC dont le nombre de colonnes(**j**) est égale au nombre des taches, et le nombre des lignes(**i**) est égale au nombre de machines. Une rangée de la matrice ETC contient les temps d'exécution estimés pour une tache tj sur chaque machine mi de l'environnement. De même, une colonne de la matrice se compose du temps d'exécution estimé d'une machine mi donnée pour chaque tâche **tj**. Pour le cas où il est possible de s'exécuter une tâche $\mathbf{t_j}$ sur la machine m_i, le temps d'exécution, ou bien la valeur de l'ETC (**i,j**) est règle à l'infini.

III.5 Approches utilisés

Nous allons proposer trois approches d'ordonnancement des taches indépendantes, pour résoudre un problème d'ordonnancement des taches dans un environnement distribué, plusieurs méthodes de distributions existent. Cette dernière permet de distribuer plusieurs taches sur plusieurs machines en même temps, mais cette méthode comporte plusieurs

problèmes à savoir, le nombre des tâches qu'exécute chaque machine afin d'optimiser le temps d'exécution **"makespan".**

Né au moins, dans une stratégie d'ordonnancement il est nécessaire de bien choisir un meilleur ordonnancement initial pour atteindre le makespan minimum. Pour cela, l'algorithme d'affectation initiale des tâches utilise l'heuristique Min-Min.

III.5.1- L'heuristique Min-Min

L'algorithme Min-Min commence par un arrangement de toutes les taches non assignées. Premièrement, le temps d'exécution minimum pour toutes les tâches est trouvé. Les tâches ayant un temps d'exécution minimum sont d'abord choisies. Deuxièmement, le temps d'exécution pour toutes les autres tâches est repensé pour cette machine [52].

Algorithme Min-min

While (Critère d'arrêt n'est pas satisfait) do

{

Initialiser le temp_exe$_i$ à zero ; //le temps d'exécution de chaque machine égale à la somme des temps d'exécution des taches qu'elles lui sont affectées.

For toutes les machines m_i

For toutes les tâches t_j

{

Chercher min_{ij} la valeur minimum de l'ETC ; // le minimumtemps d'exécution de la tâche $t_j$dans la machine m_i.

}

For toutes les machines m_i

{

temp_exe$_i$ = temp_exe$_i$ +ETC$_{ij}$. // le temps d'exécution de le tâches qui a min_{ij} dans la machine m_i.

}

For toutes les machines m_i

{

Trouver la machine qui a temp_exe$_i$ minimum.

Figure III.1- Min-Min heuristique.

Dans la suit ont un petite exemple simple de trois machines et treize taches et nous appliquons le fonctionnement de l'algorithme d'initialisation Min-Min.

La matrice ETC est donnée par le tableau 3.1

Machines	Tâches												
	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13
M_1	12	45	67	99	6	34	65	88	145	16	8	766	45
M_2	54	26	71	86	62	22	100	75	48	14	677	93	17
M_3	5	33	49	70	11	87	63	15	44	38	140	10	189

Tableau III.1 Un exemple de la matrice ETC.

Dans cette étape nous choisissons la tâche qui termine la première, ensuit le temps d'exécution minimum pour **T_1** sera réalisé sur la **M_3,** comme le montre le tableau 3.2.

Machines	Tâches													Temps d'exécution
	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	
M1	0	-	-	-	-	-	-	-	-	-	-	-	-	0
M2	0	-	-	-	-	-	-	-	-	-	-	-	-	0
M3	5	-	-	-	-	-	-	-	-	-	-	-	-	5

Tableau III.2 La première itération de l'algorithme Min-Min sur ETC.

Donc, dans chaque itération nous comparons les dates de fin d'exécution de toutes les tâches non encore ordonnancées et nous prenons le minimum de ces dates.

Ensuite, l'algorithme d'initialisation affectera la tâche **T_5** à la **M_1**

Machines	Tâches													Temps d'exécution
	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	
M1	0	-	-	-	6	-	-	-	-	-	-	-	-	6
M2	0	-	-	-	0	-	-	-	-	-	-	-	-	0
M3	5	-	-	-	0	-	-	-	-	-	-	-	-	5

Tableau III.3 La deuxième itération de l'algorithme Min-Min sur ETC.

Puis la tache **T_1** à **M_1**, et on va continuera avec le même principe avec les restes taches

Machines	Tâche													Temps d'exécution
	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	
M1	0	-	-	-	6	-	-	-	-	-	8	-	-	14
M2	0	-	-	-	0	-	-	-	-	-	0	-	-	0
M3	5	-	-	-	0	-	-	-	-	-	0	-	-	5

Tableau III.4 La troisième itération de l'algorithme Min-Min sur ETC .

L'algorithme d'initialisation affecter la tache **T_{10}** à la machine **M2.**

Machines	Tâches													Temps d'exécution
	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	
M1	**0**	**-**	**-**	**-**	**6**	**-**	**-**	**-**	**-**	**0**	**8**	**-**	**-**	**14**
M2	**0**	**-**	**-**	**-**	**0**	**-**	**-**	**-**	**-**	**14**	**0**	**-**	**-**	**14**
M3	**5**	**-**	**-**	**-**	**0**	**-**	**-**	**-**	**-**	**0**	**0**	**-**	**-**	**5**

Tableau III.5 La quatrième itération de l'algorithme Min-Min sur ETC.

L'algorithme d'initialisation affecter la tache **T_8** à la machine **M3.**

Machines	Tâches													Temps d'exécution
	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	
M1	**0**	**-**	**-**	**-**	**6**	**-**	**-**	**0**	**-**	**0**	**8**	**-**	**-**	**14**
M2	**0**	**-**	**-**	**-**	**0**	**-**	**-**	**0**	**-**	**14**	**0**	**-**	**-**	**14**
M3	**5**	**-**	**-**	**-**	**0**	**-**	**-**	**15**	**-**	**0**	**0**	**-**	**-**	**20**

Tableau III.6 La cinquième itération de l'algorithme Min-Min sur ETC**.**

A la fin nous aurons la matrice ETC finale donnée par le tableau 3.7, avec un temps d'exécution globale égale à **143** second.

Machines	Tâches													Temps d'exécution
	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	
M1	0	0	67	0	6	34	0	0	0	0	8	0	0	**115**
M2	0	26	0	86	0	0	0	0	0	14	0	0	17	**143**
M3	5	0	0	0	0	0	63	15	44	0	0	10	0	**137**

Tableau III.7 ETC finale de l'algorithme.

III.5.2- Equilibrage de charge

Dans cette étude notre contribution consiste à implémenter l'heuristique SLB (Simple Loading Balancing). Elle préconise la réduction du temps d'exécution en transférant les taches affectées sur la machine qui possède le temps d'exécution maximum à une autre machine possédant le temps d'exécution minimum afin de minimiser le makspan, comme le montre (la figure III.2).

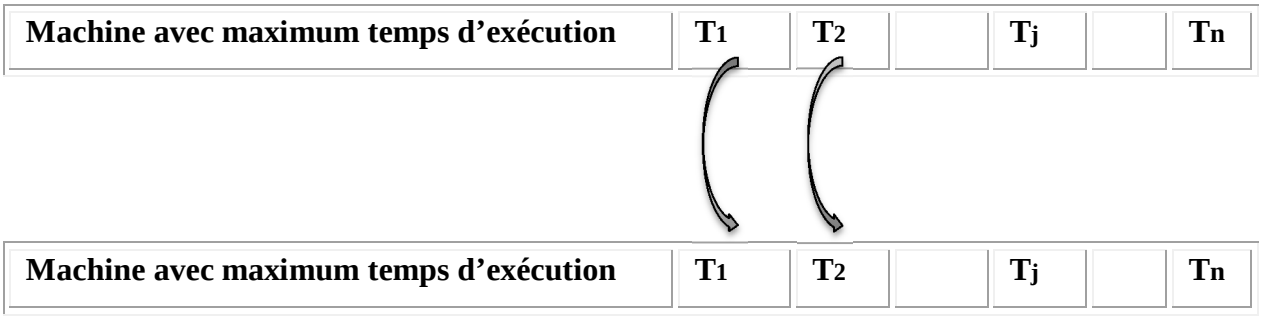


Figure III.2 Stratégie de SLB.

L'algorithme SLB effectue le raffinement du temps d'exécution maximum en réaffectant les taches. Ces taches ont nécessairement un temps inférieur à celui affecté à la machine ayant le plus grand temps d'exécution. Ensuit nous passerons au raffinement du makspane avec cette nouvelle affectation.

Algorithme 1 : ***Simple Algorithme d'équilibrage de charge***

While (critère d'arrêt n'est pas satisfait) do

{ Initialiser un auxiliaire à 0 ;//pour faire le test ;

Trouver machine m_{max}avec le maximum temps d'exécution :ct_{max} ;

Trouver machine m_{min} avec le minimum temps d'exécution :ct_{min} ;

For toutes les tâches t_j affectées sur m_{max} do

{Affecter tâche t_j sur m_{min} ;

$C_{m_{min}j} = w_{m_{min}j} + ETC_{m_{min}j}$; // Estimer temps d'exécution surm_{min};

$C_{m_{max}j} = w_{m_{max}\,j} - ETC_{m_{min}j}$; // Estimer temps d'exécution surm_{max} quand t_j est déplacée dem_{max};

If (($C_{m_{min}j}$<ct_{max}) and ($C_{m_{max}j}$< auxiliaire) // pour avoirle minimum de $C_{m_{max}j}$

{

Affecter $C_{m_{max}j}$à l'auxiliaire ;

Sauvegarder j ;//sauvegarder la tâche qui donne le minimum temps d'exécution

} }

Affecter la tâche t_j qui donne le minimum temps d'exécution parmi toutes affectées sur la machine m_{max} // la tâche j que nous avons déjà sauvegardé ;

Else

Critère d'arrêt est satisfait ; //la valeur du makespan n'est pas changée.

}

Figure III.3 Simple Algorithme d'équilibrage de charge.

A titre exemple, nous allons ordonnancer les taches de l'exemple précédent par la matrice ETC initialisé par l'algorithme Min-Min**,** voir le tableau 3.7 ou la valeur du makespan sur **M2** est de **143**. La première itération de l'algorithme a réaffecté la tache **T10** de la **M2** à la **M1** comme indiqué dans le tableau 3.8, cette réaffectation permet de réduire le makespan de **143** à **132**.

Machines	Tâches													Temps d'exécution
	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	
M1	0	0	67	0	6	34	0	0	0	**14**	8	0	0	**129**
M2	0	26	0	86	0	0	0	0	0	**0**	0	0	17	**129**
M3	5	0	0	0	0	0	63	15	44	**0**	0	10	0	**137**

Tableau III.8 Processus d'exécution de l'algorithme SLB-itération 1.

Dans la deuxième itération, l'algorithme SLB considère **M3** pour la réaffectation. La tâche **T1** est réaffectée sur **M1** et la makespan est raffiné à **134**, comme le montre le tableau.3.9.

Machines	Tâches													Temps d'exécution
	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	
M1	**5**	0	67	0	6	34	0	0	0	**14**	8	0	0	134
M2	**0**	26	0	86	0	0	0	0	0	**0**	0	0	17	129
M3	**0**	0	0	0	0	0	63	15	44	**0**	0	10	0	132

Tableau III.9 Processus d'exécution de l'algorithme SLB_itération 2-final.

De même notre algorithme continue de raffiner le makespan, en déplaçant les taches qui réduisent la valeur du makespan de la machine M_{Max} à la machine M_{Min}, jusqu'à que la valeur de makespan ne peut pas être raffinée. D'apré les résultats de l'algorithme d'ordonnancement Min-Min, l'algorithme d'équilibrage de charge simple donne le meilleur résultat du makespan, ce dernier est réduit de **143** à **134**, (voir le tableau III.9).

III..3- Equilibrage de charge avec Echange

Pour améliorer les performances de l'heuristique SLB, une nouvelle technique d'équilibrage de charge est proposé, c'est l'équilibrage de charge avec échange LBE (Loading Balancing Exchange). Cette technique généralise l'ordonnancement SLB.

Son principe est de minimiser le temps d'exécution maximal obtenu par l'ordonnancement, en réattribuant les taches dans le but d'améliorer le temps nécessaire de la solution d'ordonnancement Min-Min, tout en minimisant le temps d'exécution maximum.
LBE considère la machine ayant le temps d'exécution maximum pour le réduire, par l'échange des taches avec les autres machines et par la réaffectation des taches entre elles.

Pour chaque tache tj de la machine ayant le temps maximal d'exécution, et pour chaque tâche t_k des autres machines, on effectue un échange de la tâche t_k. Nous réaffecterons

une seul paire d'échange des taches en même temps qui donne le makespan minimum parmi toutes les affectations de tâche entre les machines.

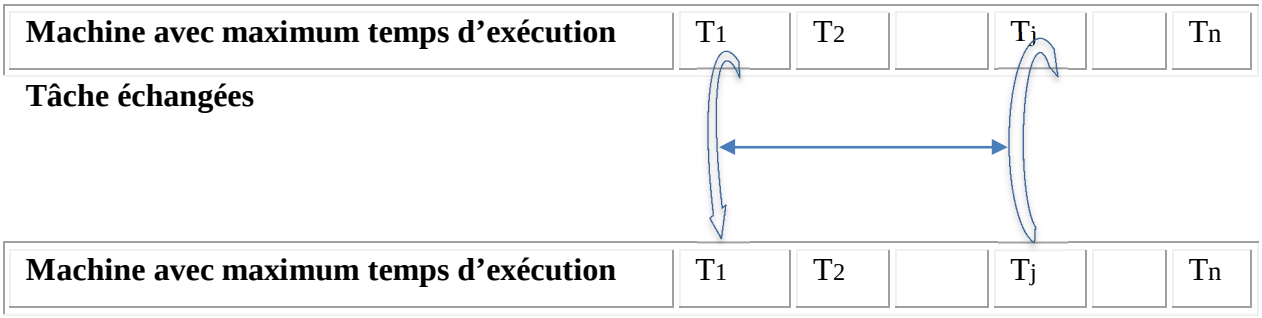


Figure III.4 Stratégie d'échange LBE**.**

La complexité de l'algorithme LBE est O(k,m,n^2), ou k le nombre d'itération tant que la valeur de l'affinement du makespan n'a pas changé, m est le nombre de machine et *n* est le nombre de tâches.

Algorithme 2 : Algorithme d'équilibrage de charge avec Echange

While (critère d'arrêt n'est pas satisfait) do

{ Trouver machine m_{max} avec le maximum temps d'exécution : ct_{max} ;

For toutes les tâches t_j affectées sur m_{max} do

{

For toutes les machines m_i avec $m_i \neq m_{max}$ do

{

For toutes les tâches t_k affectées sur m_i do

{

Echange tâche t_k avec t_j ;

$C_{ij} = W_i + ETC_{ij} - ETC_{ik}$; // temps d'exécution de m_i quand t_j est affectée sur m_i .

$C_{m_{max}k} = W_{m_{max}} + ETC_{m_{max}k} - ETC_{m_{max}j}$; // temps d'exécution de m_{max} quand t_k est affectée sur m_{max} .

If ($C_{ij} < ct_{max}$) and ($C_{m_{max}k} < ct_{max}$)

{

$ct_{max} = C_{m_{max}k}$,

t_k de m_i et t_j de m_{max} ; // qui donne minimum temps d'exécution parmi toutes les tâches affectées sur la machine m_{max}

}

}

}

Echange tâche t_k avec t_j de m_{max} qui donne minimum teps d'execution parmi toutes les tâches affectées sur la machine m_{max} ;

}

If (makespan n'est pas minimiser)

critère d'arrêt satisfait ;

}

Figure III.5 Algorithme d'équilibrage de charge avec Echange.

Nous allons ordonnancer les taches précédentes données par la matrice ETC initialisée par l'algorithme Min-Min, voir le tableau 3.7 ou la valeur du makespan sur **M2** est de **143** .La première itération de l'algorithme a réaffecte la tache $\mathbf{T_{13}}$ de la **M2** à la **M1**, la tâche $\mathbf{T_5}$ de la **M1** à la **M2** comme indiqué dans le tableau 3.10. Cette réaffectation permet de réduire le makespan de**143** à **140**.

Machines	Tâches													Temps d'exécution
	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	
M1	0	0	67	0	**0**	34	0	0	0	14	8	0	**17**	**140**
M2	0	26	0	86	**6**	0	0	0	0	0	0	0	**0**	**118**
M3	5	0	0	0	**0**	0	63	15	44	0	0	10	**0**	**137**

Tableau III.10- Processus d'exécution de l'algorithme LBE_itération 1.

Dans la deuxième itération, la **M1 est** la machine m_{max} ou la valeur de makespan est **140**. La tâche $\mathbf{T_8}$ de la **M3** est réaffectée sur la **M2,** et la tâche $\mathbf{T_{11}}$ de la **M1** est réaffectée sur la **M3**. Le makespan est raffiné **140** à **133**, comme il est indiqué dans le tableau III.11.

Machines	Tâches													Temps d'exécution
	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	
M1	0	0	67	0	0	34	0	**0**	0	14	**0**	0	17	**132**
M2	0	26	0	86	6	0	0	**15**	0	0	**0**	0	0	**133**
M3	5	0	0	0	0	0	63	**0**	44	0	**8**	10	0	**130**

Tableau III.11- Processus d'exécution de l'algorithme LBE_itération 2-final.

De même l'algorithme LBE continue de raffiner le makespan, en réaffecterons une seul paire d'échange des tâches en même temps qui donne un makespan minimum parmi toutes les affectations jusqu'à que la valeur de makespan ne peut pas être raffinée.

D'après les résultats de l'algorithme d'ordonnancement Min-Min, et de l'algorithme SLB,

Comme le montre le tableau III.7 (et III .10) l'algorithme d'équilibrage de charge avec échange LBE donne le meilleur résultat du makespan, ce dernier est réduit de à , voir le tableau III.11.

III.6 Conclusion

Le problème d'ordonnancement consiste à organiser dans le temps l'exécution de tâches, compte tenu de contraintes temporelles et de contraintes portant sur l'utilisation et la disponibilité des ressources requises. Les problèmes d'ordonnancement étaient déjà difficiles en environnement homogène, il n'y a aucune raison qu'ils se simplifient en environnement hétérogène, nous avons traité dans ce chapitre les solutions apportées dans la littérature au problème d'équilibrage de charge dans le cloud.

L'objectif de l'étude est de consolider les méthodologies existantes pour l'équilibrage de charge dans le cloud.

Chapitre IV

Implémentation, résultat et discussion

Chapitre IV

Implémentation, résultat et discussion

IV .1 Introduction

A travers ce chapitre, nous présenterons les apports de notre travail en faisant une synthèse comparative entre les méthodes existantes, avec des discussions des différents résultats obtenus.

On devise ce chapitre la en deux parties, premier partie constituent la conception de notre application et le modèle de modélisation ensuit on présente les différents interfaces de notre application.

Partie 1

IV.2 Conception de l'application

Nous avons choisi le langage UML (Unified Modeling Language) en raison de son efficacité et sa simplicité de modélisation et de représentation des systèmes entiers. Nous présentons brièvement le fonctionnement de l'application développée et certains diagrammes de la mise en œuvre de notre application.

IV.2.1 Expression des besoins

Notre projet se résume à proposer et implémenter les d'algorithmes d'ordonnancement des taches dans un environnement cloud. Vu les difficultés rencontrées avec les résultats fournis, no avons opté pour faire une seconde implémentation qui facilite la gestion des données, résultats et méthodes. Les avantages et les fonctionnalités de cette application sont présentés ci-dessus :

IV.2.2 Exigences fonctionnelles

Pour que note application et réussie, il est important de satisfaire le maximum de besoins utilisateurs, tel que :

- **Formulations des demandes et exigences par l'utilisateur:** cette option permettra à l'utilisateur de formuler ses demandes et ses exigences surtout en termes de minimisation de temps d'exécution ou de stockage ;
- **Exécution de projet :** lors de l'exécution de notre application, les demandes de l'utilisateur pourront être satisfaites avec les meilleurs résultats possibles dans les meilleurs délais ;
- **Traitement des problèmes:** si un problème survient lors l'exécution, cela veut dire que les données fournis par l'utilisateur sont erronées ou bien mal entrées, et ce dernier devra modifier ses entrées.

IV.2.3 Exigences non fonctionnelles

Pour que notre application prenne place dans une organisation, elle répondre aux exigences de qualité et de performance suivantes :

Il faut que les réponses aux demandes de l'utilisateur surviennent dans les plus brefs délais ;

- L'application doit être facile à utiliser car les interfaces trop riches et complexe demandent un effort supplémentaires de compréhension ;
- Le système doit pouvoir gérer plusieurs opérations, tel que les résultats.

IV.3 Présentation du langage de modélisation

Ce travail a été fait à travers le langage de la modélisation unifié UML, cette méthode représente un moyen de spécifier, représenter et construire les composantes d'un système informatique ; UML (Unified Modeling Language) est un langage unifié pour la modélisation objet. UML est conçu pour modéliser divers types de systèmes, de taille quelconque et pour tous les domaines d'application (gestion, scientifique, temps réel, système embarqué).

UML se compose d'une part des éléments de modélisation qui représentent toutes les propriétés du langage et d'autre part des diagrammes (de cas d'utilisation, de classes, d'objets, d'états-transitions, d'activités, de séquence, de collaboration, de composants et de déploiement) qui en constituent l'expression visuelle et graphique.

IV.3.1 Diagramme de cas d'utilisation

- **Identification des acteurs** : Dans notre application on a un seul acteur humain
- **L'utilisateur** : C'est lui qui prend en charge la gestion des données d'entrées, le traitement des méthodes et la gestion des résultats.
- **Identification des cas d'utilisation**: Après avoir défini l'acteur principal de l'application. Nous passons à la détermination de ses cas d'utilisation qui illustrent le comportement du système en réponse à une interaction avec l'utilisateur. On peut associer à l'utilisateur plusieurs cas d'utilisation du système, Notre diagramme de cas d'utilisation est illustré dans la figure IV.1.

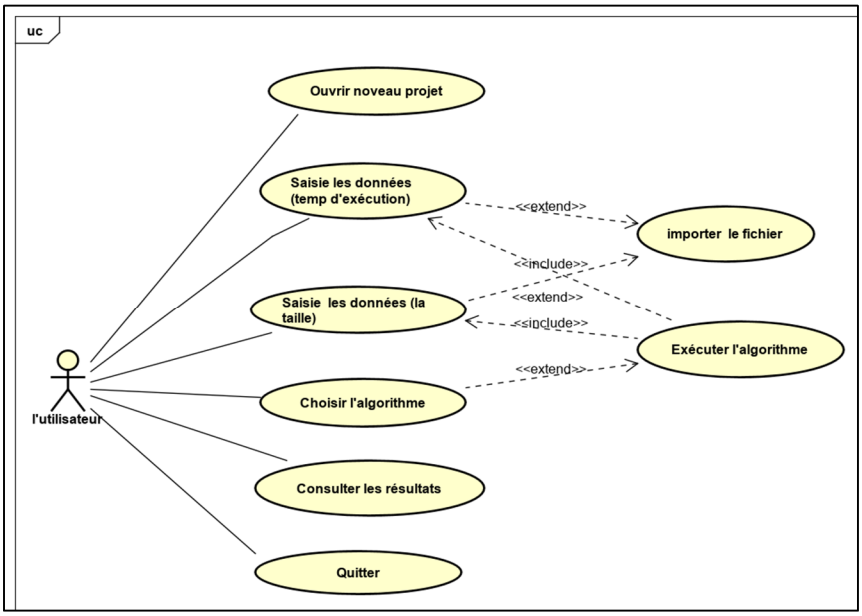


Figure IV.1 Diagramme de cas d'utilisation.

IV.3.2 Diagramme de séquence

Nous décrivant à travers cette section les principaux cas d'utilisation mis en évidence par l'expression des besoins préliminaires et énumérés dans la figure IV.1

Pour se faire, nous allons détailler les fonctionnalités de quelque cas d'utilisation dont nous jugeons importantes via une description textuelle et un digramme de séquences. Pour ce qui suit nous considérons dans les descriptions textuelles que l'acteur est identifié et ont eu accès à tous les espaces avec succès dans tous les scénarios.

IV.3.3 Cas d'utilisation : Saisir les données.

- **Acteur** : Utilisateur.
- **Pré conditions**: L'utilisateur lance l'application et atteint la fenêtre « Données».
- **Post conditions** : Affichage de la fenêtre des données.
- **Scénario nominal** :
  - L'utilisateur clique sur bouton « Ordonnancer».
  - L'assistant affiche la matrice à remplir manuellement.
  - L'utilisateur introduit les données d'entrées en clique sur le bouton (Ajouter).
  - L'utilisateur clique sur le bouton (Confirmer).

- **Alternatives** :
  - L'utilisateur peut à tout moment quitter.
  - L'utilisateur peut modifier ses données.
  - L'utilisateur peut importer les données en cliquant sur le bouton (parcourir).

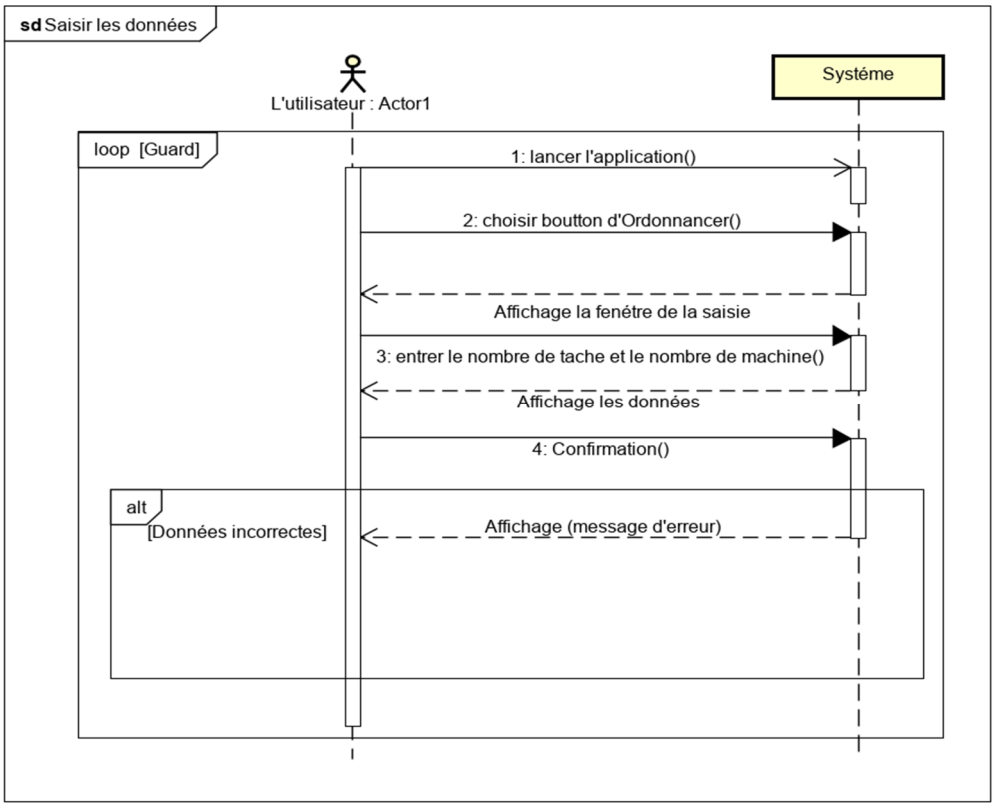


Figure IV.2 DSS Saisir les données.

IV.3.4 Cas d'utilisation : Choisir un algorithme.

- **Acteur** : Utilisateur.
- **Pré conditions** : L'utilisateur lance l'application et atteint la fenêtre « saisie les données ».
- **Post conditions** : Affichage des algorithmes à exécuter.
- **Scénario nominal :**
  - L'utilisateur clique sur bouton d'ordonnancer.
  - L'assistant affiche la fenêtre de la saisie.
  - L'utilisateur entrer le nombre de tache et le nombre de machine.
  - L'utilisateur choisit un algorithme en cliquant sur le bouton indiquant cet algorithme.
  - L'assistant affiche la fenêtre de l'algorithme choisi pour l'exécution.

- **Alternatives** :
  - L'utilisateur peut à tout moment quitter.
  - L'utilisateur peut choisir un autre algorithme L'utilisateur peut à tout moment quitter.
  - L'utilisateur peut exécuter l'algorithme.

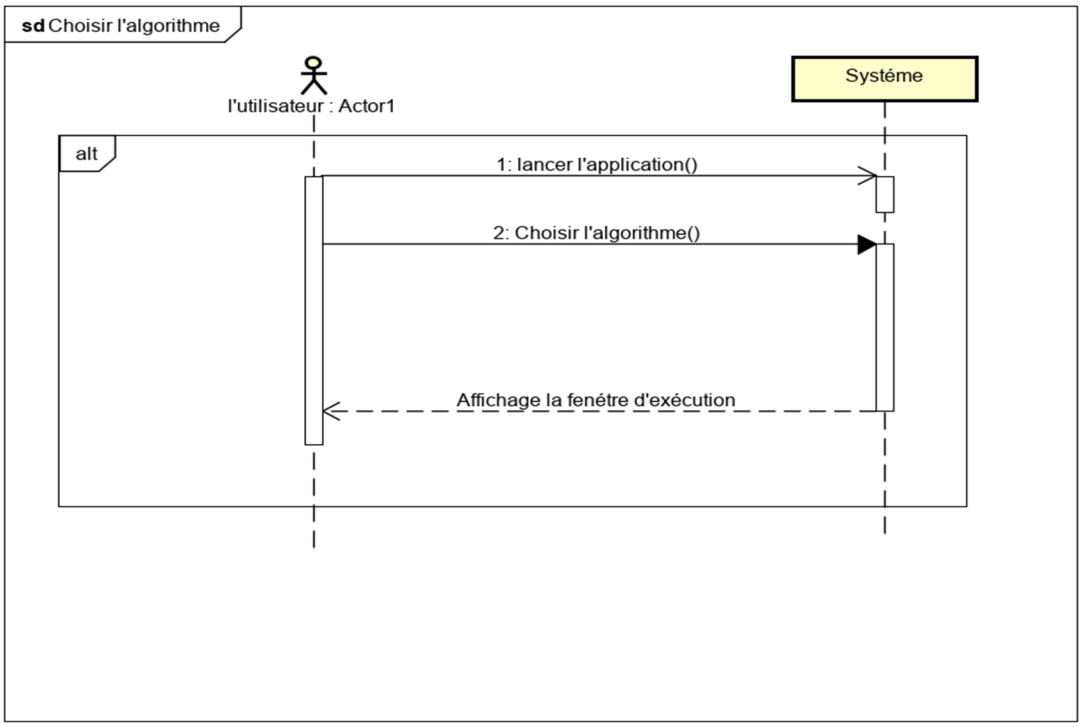


Figure IV.3 DSS Choisir un algorithme.

IV.3.5 Cas d'utilisation : Exécuter l'algorithme.

- **Acteur :** Utilisateur.
- **Pré conditions :** Saisir les données.
- **Post conditions :** Affichage les résultats d'exécution.
- **Scénario nominal :**
  - L'utilisateur clique sur le bouton « Ordonnancer ».
  - L'utilisateur introduit les données d'entrées et clique sur le bouton Confirmer.
  - L'utilisateur choisit l'algorithme.
  - L'assistant exécute l'algorithme et il affiche les résultats.

- **Alternatives :**
  - L'utilisateur peut modifier ses données.
  - L'assistant affiche un message d'erreur.
  - L'utilisateur peut choisir un autre algorithme.
  - L'utilisateur peut à tout moment quitter.

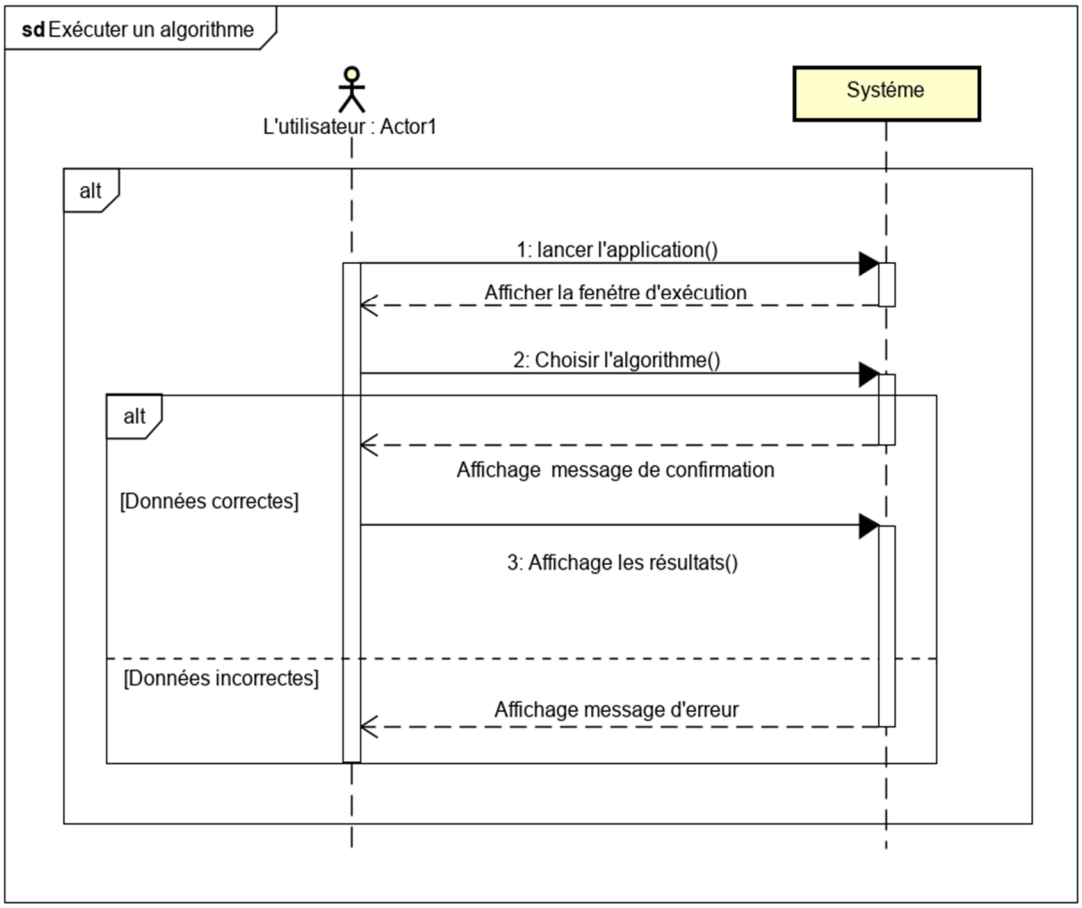


Figure IV.4 DSS Exécuter l'algorithme.

IV.4 Implémentation de l'application

IV.4.1 Présentation du langage de programmation :

Borland C++Builder est un environnement de programmation visuelle orienté objet permettant le développement d'applications 32 bits en vue de leur déploiement sous Windows et sous Linux. En utilisant C++Builder, vous pouvez créer de puissantes applications avec un minimum de programmation. C++Builder propose un ensemble d'outils de conception pour le développement rapide d'applications (RAD), dont des experts programmateur et des modèles d'applications ou de fiches, et gère la programmation orientée objet avec deux bibliothèques étendues de classes :

La bibliothèque de classes VCL comprend des objets qui encapsulent l'API Windows ainsi que d'autres techniques de programmation utiles (Windows). La bibliothèque de composants Borland multi-plateforme (CLX), qui contient des objets encapsulant la bibliothèque Qt (Windows ou Linux).

IV.4.2 Présentation de l'application

Pour que notre application réussie, il est important de satisfaire le maximum des besoins des utilisateurs, tel que :

- Formulation des demandes et des exigences par l'utilisateur: cette option permettra à l'utilisateur de formuler ses demandes et ses exigences surtout en termes de minimisation du temps d'exécution ;
- Exécution du projet: lors de l'exécution de notre application, les demandes de l'utilisateur pourront être satisfaites avec les meilleurs résultats possibles dans les meilleurs délais.
- Traitement des problèmes : si un problème survient lors de l'exécution, cela veut dire que les données fournis par l'utilisateur sont erronées ou bien mal entrées, et ce dernier devra modifier ses entrées.

a.Interface d'accueil

Comme son nom l'indique, elle constitue le point d'entrée de notre application (voir la figure IV.5). Elle permet d'avoir un accès à l'application. Elle est composée d'un menu qui contient les fonctionnalités de notre système.

De cette fenêtre on est capable d'exécuter l'ordonnancement des taches indépendantes, aussi à partir de cette fenêtre en peut lancer un algorithme pour l'ordonnancement des données (le placement de données).

Figure IV.5 Fenêtre d'accueil.

b. Interface saisir les données et exécuter un algorithme d'ordonnancement

Cette page permet à l'utilisateur de saisir ses données, à savoir, le nombre de tâches, le nombre de machines et le temps d'exécution de chaque tâche dans chaque machine (manuelle).

L'utilisateur peut importer ses données à partir d'un fichier existant (parcourir un fichier).

Dans cette interface on a un menu qui permet l'accès aux différents algorithmes à exécuter à savoir l'algorithme Min-Min, LBE, SLB.

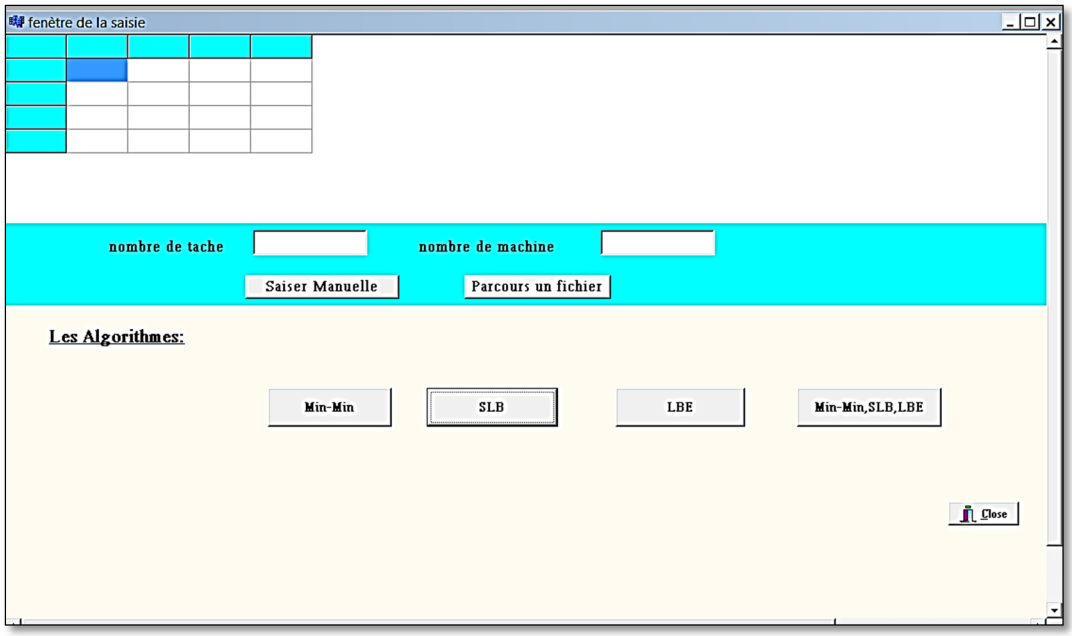


Figure IV.6 Fenêtre de saisie.

c. La page d'exécution,

Premièrement, on va exécuter l'algorithme d'initialisation Min-Min.

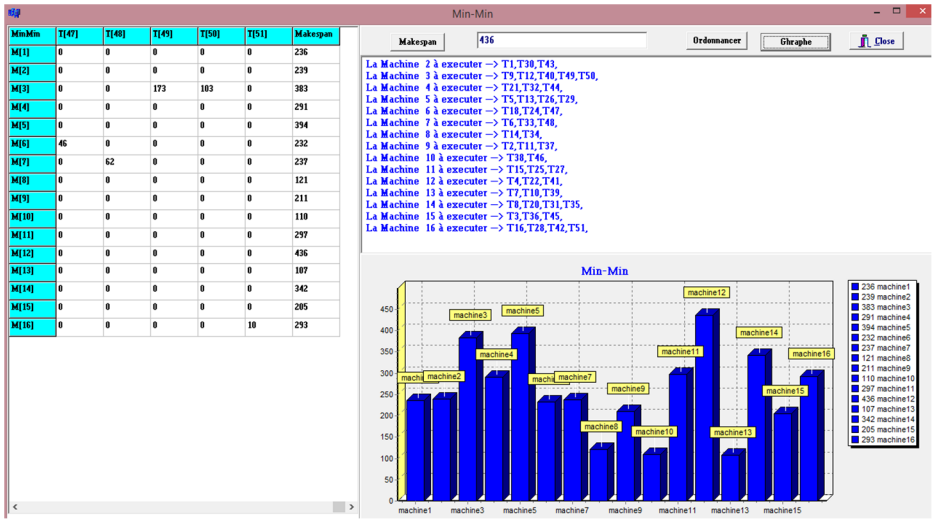


Figure IV.7 Fenêtre d'exécution de l'algorithme Min-Min.

Apres l'exécution de l'algorithme Min-Min, on va exécuter les deux algorithmes SLB et LBE (ce sont des algorithmes d'équilibrage de charge).

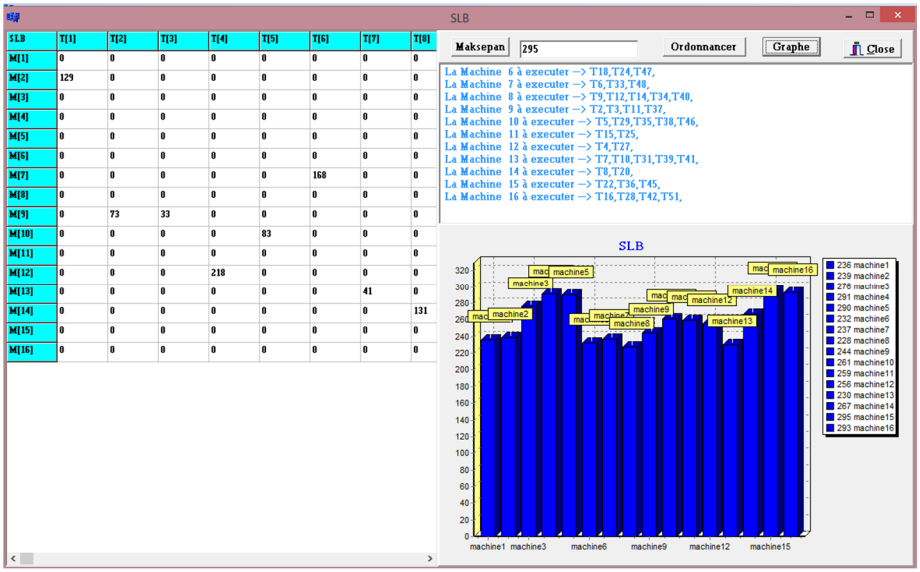


Figure IV.8 Fenêtre d'exécution de l'algorithme SLB.

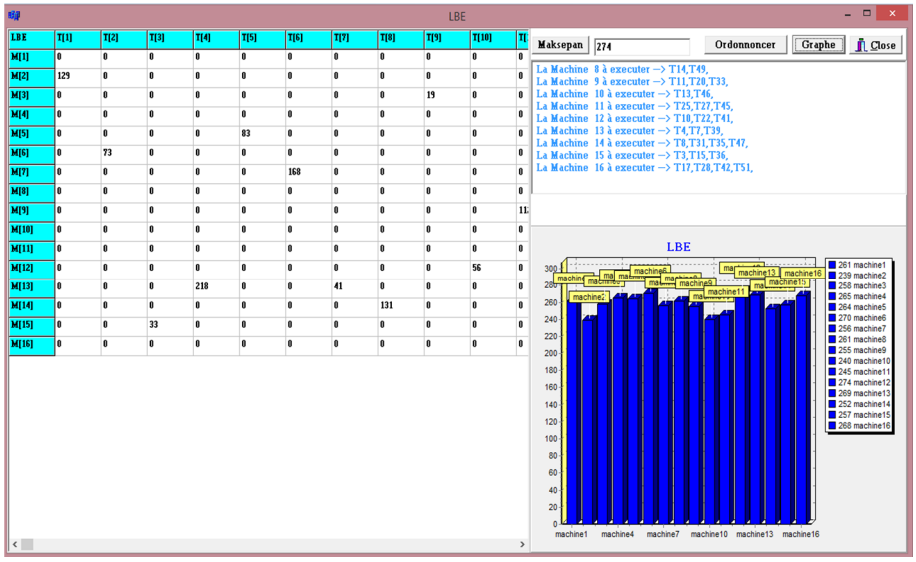


Figure IV.9 Fenêtre d'exécution de l'algorithme LBE.

La figure IV.10 représente la comparaison entre les différents algorithmes dans le cas de l'ordonnancement des taches.

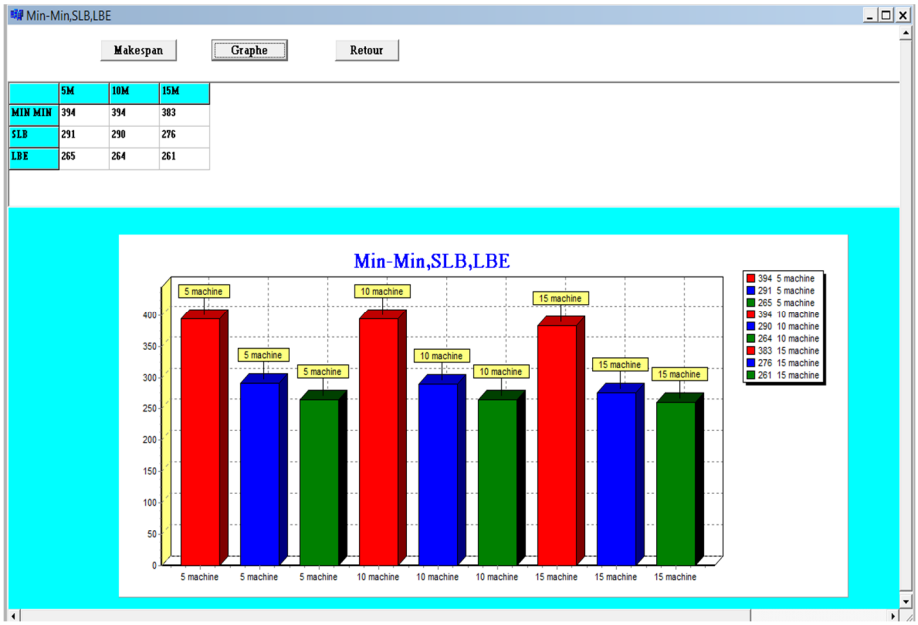


Figure IV.10 Fenêtre d'exécution de Min-Min, SLB, LBE.

On cliquer sur bouton retour pour revenir à la page d'accueil, ensuit on va exécuter le deuxième point pour afficher les résultats de placement de données.

La figure IV.11 représente l'exécution de l'ordonnancement et le placement des données.

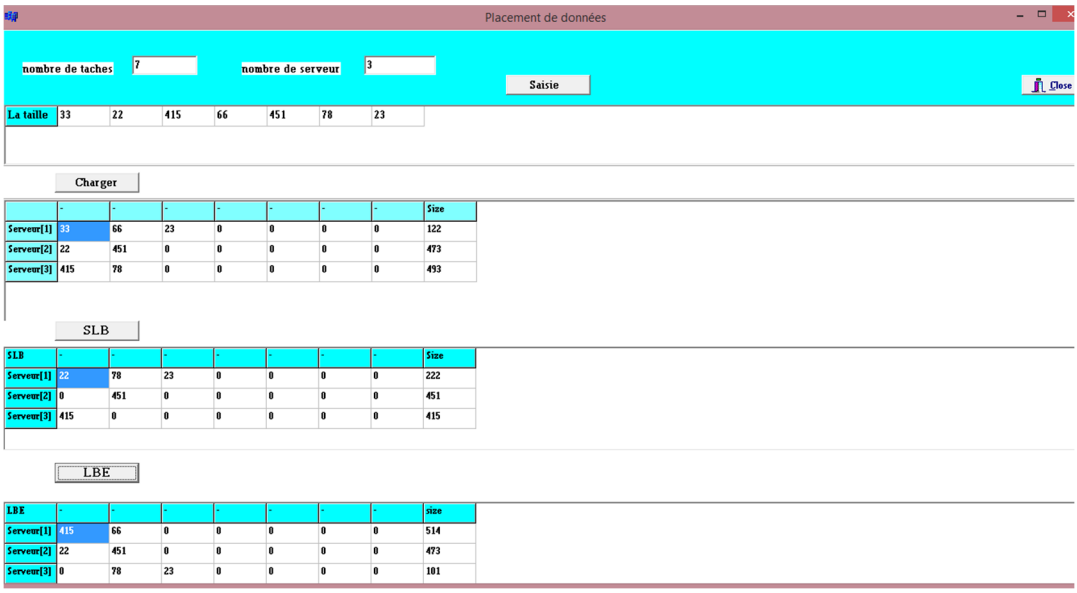


Figure IV.11 Fenêtre de placement de données.

Partie 2

IV.1 Résultats expérimentaux

Au travers de cette partie, nous présenterons les apports de notre travail en faisant une synthèse comparative avec les méthodes existantes à travers une discussion des différents résultats obtenus. L'évaluation des méthodes SLB et LBE exploite des problèmes ordonnancement des tâches indépendantes dans un environnement hétérogène.

IV.2 L'évaluation des approches proposées

Dans cette phase, afin de valider notre stratégie d'ordonnancement de tâches sur environnements hétérogènes, nous avons opté pour une évaluation et une comparaison avec les approches proposées. Pour se faire, des séries d'expériences ont été réalisées. Les algorithmes proposées ont été appliqués à des matrices ETCs avec 12 différents types, jusqu'à 16 machines hétérogènes et jusqu'à 512 tâches hétérogènes[53]. Ces matrices ont une des propriétés suivantes :

- Hétérogénéité des tâches : représente la somme de la variance entre les temps d'exécution des tâches pour une machines donnée. L'hétérogénéité des tâches est définie comme : **lo : low** et **hi :high**.
- Hétérogénéité des machines : représente la variation à entre le temps d'exécution pour une tâche donnée dans l'ensemble des machines. L'hétérogénéité machine est définie comme : **lo et hi**.
- L'unité de temps pour toutes ces matrices est seconde.

IV.2.1 Le perfectionnement de l'heuristique Min-min

Dans le cadre d'évaluer nos algorithmes SLB et LBE, nous monterons dans cette section la contribution de ces derniers par rapport à l'heuristique Min-min. Nous avons fait des tests en utilisant 16 matrices ETC (une pour chaque cas algorithme et une pour chaque cas d'hétérogénéité). Les résultats obtenus sont reportés dans les figures.

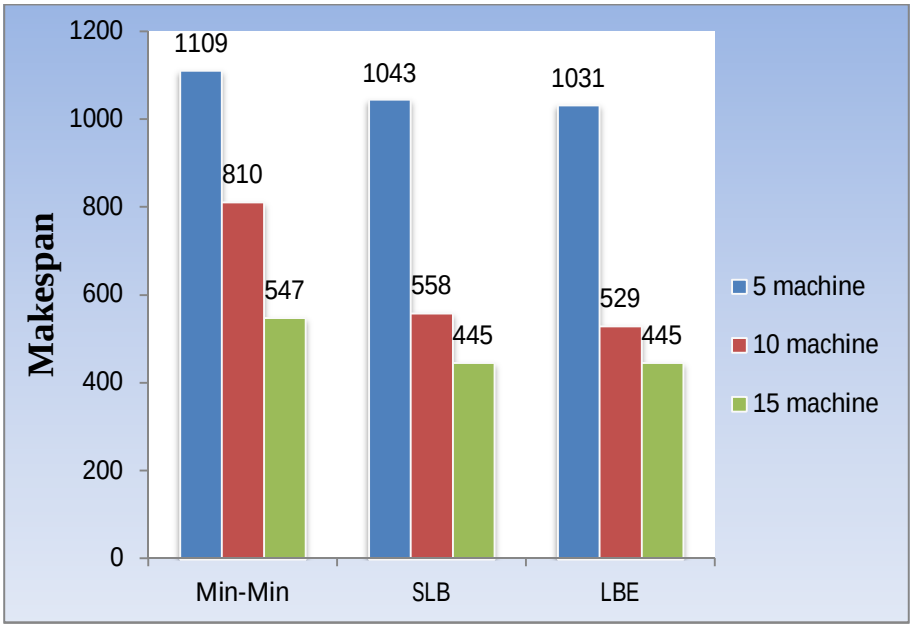


Figure IV.17 Le cas hétérogénéité faible de la tache et de la machine faible (**lolo**).

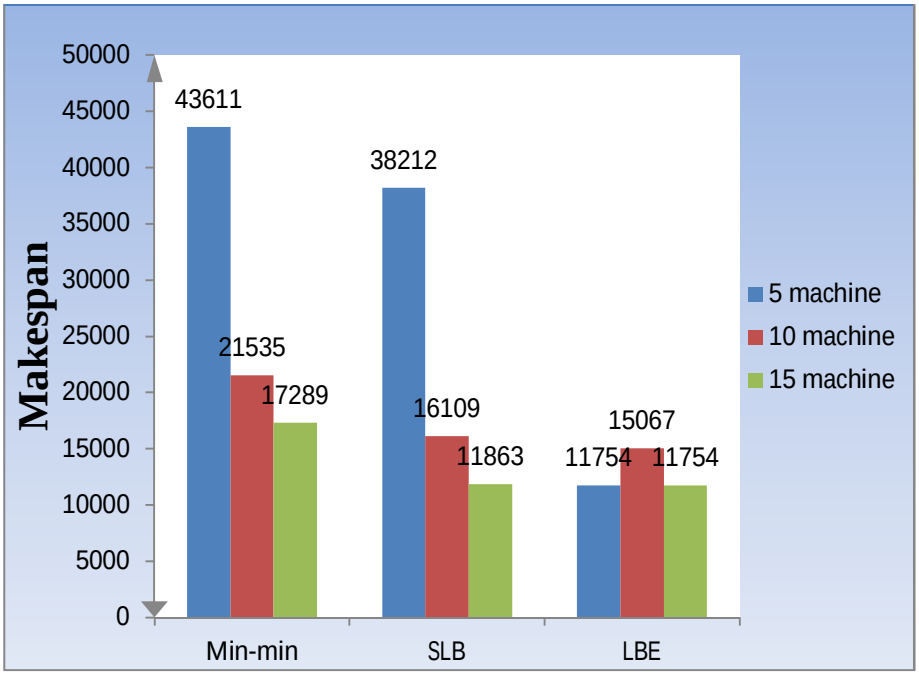


Figure IV.18 Le cas hétérogénéité élevée de la tache et de la machine élevée (**hihi**).

La figure IV .17 et figure IV.18 montrent le perfectionnement apporté par les approches SLB et LBE à l'heuristique Min-min. Les améliorations sont considérables pour tous les cas d'hétérogénéité.

IV.3 Conclusion

Dans ce chapitre nous avons d'abord présenté l'environnement matériel et l'environnement logiciel de notre implémentation. Puis il a été question d'évaluer nos approches proposées, à savoir celui du SLB et LBE.

D'après les tests effectués, l'algorithme SLB minimise le temps total d'exécution, alors que l'algorithme LBE permet d'approcher au mieux la solution optimale. Ensuite, nous avons abordés les différentes fonctionnalités que propose notre application en illustrant ceci avec des explications textuelles et les captures d'écrans de l'application.

Conclusion Général

D'après la recherche et le travail qu'on a mené à propos du cloud computing on peut sans hésitation conclure que ce dernier est une révolution dans la manière d'organiser, de gérer et de distribuer des ressources informatiques. Sa définition opérationnelle annonce un modèle informatique qui permet un accès facile et à la demande, par le réseau, à un ensemble partagé de ressources informatiques configurables et depuis n'importe quel appareil disposant d'un navigateur et d'une connexion internet. Il peut s'agir de serveurs, de stockage, d'applications ou de services, rapidement provisionnés et libérés par un minimum d'efforts de gestion.

Dans ce mémoire, nous avons essayé de proposé des algorithmes qui réalise le problème d'ordonnancement de tache et répondre à la problématique d'équilibrage de charge dans le cloud, qui va éviter la surcharge des Clusters, et qui va répliquer les données au plus prés des clients afin d'améliorer le temps de réponse.

Ce travail contient une présentation sélective des méthodes développées dans la littérature, traitant l'équilibrage de charge dans le Cloud. L'équilibrage de charge peut être atteint soit par placement de données ou de fragments de données, soit par placement de taches ou de machine virtuel (VM), dans cette étude nous avons développé les approches SLB et LBE pour résoudre le problème d'ordonnancement des tâches sur environnements Cloud dans le cas de calcul distribué.

L'approche d'équilibrage de charge simple SLB (Simple Loading Balancing) destinée à la réaffectation des tâches entre deux ressources sans échange. La seconde est l'approche d'équilibrage de charges avec échange LBE (Loading Balancing Exchange) qui utilise le même principe que SLB mais avec échange en réaffectant un pair de tâches entre deux ressources, avec un algorithme d'initialisation Min-Min. Ces deux heuristiques ont été présentées en détaille et des tests de validation ont été effectués pour évaluer leurs bons fonctionnements.

Dans une deuxième partie de ce mémoire nous avons essayé d'appliquer les précédents algorithmiques dans le cas de placement des données sur un cloud, et dans l'objectif l'optimisation de stockage de données.

Les résultats obtenus montrent que les méthodes SLB et LBE donnent des bons résultats en terme de temps d'exécution global (makespan) et plus particulièrement la méthode LBE.

De plus, et comme perspective de notre travail, il serait intéressant de simuler les différentes approches afin de les comparer entre elles.

Bibliographie

[01] Jean-Paul Figer, L'nformatique en nuage[Cloud Computing],H6 020 **:** Cloud Computing, Informatique en nuage dans la rubrique Architecture des systèmes des Techniques de l'ingénieur, 29/12/2017.

[02] BENMAACHOU Zahira, Simulation D'ordonnoncement Des Taches Independantes Sur Environnement De Calcul Heterogenes , Mémoire de Master,2015 .

[03] P. Mell and T. Grance. The NIST Definition of Cloud Computing. 2011.

[04] A.A.Y. Elwessabi. Une approche bas´ee agent mobile pour le cloud computing. M´emoire de magister, Universit´e HADJ LAKHDAR - BATNA, 2014.

[05] R. Buyya, S.k. Garg, and R.N. Calheiros. SLA-oriented resource provisioning for cloud computing : Challenges, architecture, and solutions. Proceedings of the International Conference on Cloud and Service Computing (CSC),Hong Kong, China, pp. 1-10, 2011.

[06] P. Mell and T. Grance. The NIST Definition of Cloud Computing. 2011

[07] I .Laribi, La mise en place de solution Open Stack,Mémoire master, Université Abou Bekr Belkaid-Telemcen,2014.

[08] S,Lanani, Une approche BPN(Business Process Managment)par composition d'application dans le cloud computing,Mémoire de magister, Université Mouhamed khider.

[09] L.F.Noumsi, Etude et mise en place d'une solution « Cloud Computing » privé dans une entreprise moderne : cas de CAMTEL.

[10] J.Zhu,Cloud Computing Technologie and Application Springer Science+Business Media,LLC 2010.

[11] M.J algaonkar and A.Kanojia , Adoption of Cloud Computing in Distance Learning, International Journal of Advanced Trends in Computer Science and Enginering,vol,2no1,pp,17-20,2013 .

[12] K.Gai and S .Li.Towards Cloud Computing : A Literature Review un Clou Computing and Its Development Trends. In 2012 Fourth International Conferenc on Multimedia Information Networking and Secuity (MINES), pp,142-46.IEEE,2012.

[13] M.R.HABES , Allocation de Roussources dans le Cloud Computing , These de Doctorat, Université BADJI MOKHTAR ANNABA,2015.

[14] https://www.tophebergeur.com/tutoriels/cloud/cloud_vs_grid_computing/

[15] I.S.ARR, Routage des Transactions dans les Bases de Donnéea large Echelle, Thèse du gard Doctorat .Université Pierre Marrie Curie(ParisVI),2011 .

[16] M.Quinson, Découvert automatiquedes Ccractiristique et capacité d'une Plate-forme de Calcul distribué, Thèses du gard Docteur de l'Ecol de Normale supérieure de lyon spécialité :Informatique, 2003 .

[17] Mathieu jani.JuxMen : un service de partage transporent de donnée pour les grilles de calculs fondé sur une approche pair-à-pair.These de doctorat, Université de Rennes 1 , IRISA, Rennes France, Novombre 2006

[18] Achraf LABIDI, partage efficace des ressources de calcul dans le nuage informatique, mémoire présenté à l'école de technologie supérieure comme exigence partielle à l'obtention de la maîtrise avec mémoire en génie concentration : génie logiciel m. sc. a. , Montréal, le 11 juillet 2017

[19] Imène BERGAOUI,presentation , cloud computing et calcul haute performance,2017, https://www.slideshare.net/MounaMaazoun/cloud-computing-et-calcul-haute-performance?from_action=save

[20] https://www.journaldunet.com/solutions/expert/68446/cloud-et-calcul-a-haute-performance---opportunite-des-start-up-aux-grands-groupes.shtml

[21] Françoise Roch, Calcul Haute Performance architectures et modèles de programmation programmation, Observatoire des Sciences de l'Univers de Grenoble.

[22] J. Carlier et Ph. Chrétienne, Problèmes d'ordonnancement : modélisation, complexité, algorithmes, Masson, Paris, 1988.

[23] Philippe Baptiste, Emmanuel Néron, Francis Sourd, Modèles et algorithmes en ordonnancement, Ellipses, Paris, 2004.

[24] S.Bansal, P.Arpaci-Dusseau,D.E., D.E.Culler, and A.M.Maintwarinh.Scheduling with Implicite Information in Distributed Systems SIGMETRICS performance Evaluation Review, 26(1), 1998.

[25] C.H.Papadimotriou.Computational complexity. John Wiley and Sons, 2003.

[26] D.S .Johnson .The NP-completeness column: an ongoing guide .Journal of Algorithms, vol.6, no.3, 1985.

[27] D.G.Feitelson and M,A. Jette .Improved Utilization and Responsiveness with Gang Scheduling. In proceedings of the Job Scheduling Strategies for parallel Processing (JSSPP 1997),Geneva,1997.

[28] D.E .Knuth .The art of computer programming. Addison-Wesley, Reading, MA.Ist edition,1973.

[29] V.Ceny.A thermodymical approach to the traveling salesman problem: an efficient simulation algorithm Journal of Optimization theory and Application,45(1),1985.

[30] Kirkpatrick, C.D.Gelatt, et K. P.Vecchi.Optimization by simulated annealing .science.220,1983.

[31] M.Mahewswaran,S .Ali,H.J.Siegl ,D.Hensgen,and R.F .Freund.Dynamic Matching and Scheduling of a Class of Independent Tacks onto Heterogeneous Computing Sustems.In Proceedings of the Eighth Heterogeneous Computing Workshop (HCW*99),Puerto Rico ,April 1999.IEEE Computer Society.

[32] M.Dorigo,G.Caro et L.M.Gambardella. Ant algorithms for discret optimization Artificiel Life, 5(2),1999.

[33] M.Dorigo,G.Caro et L.M.Gambardella.. Ant colony system:Acooprative learning approach to the traveling problem IEEE Transactions on Evolutionary Computation 1(1),1997.

[34] P. Esquirol et P. Lopez, L'ordonnancement [archive], Economica, Paris, 1999.

[**35**] C-H.Lin and C-J.Liao.Makespan minization for multiple uniform machines.Comut.Ind.Eng, 54(4):983.992,2008.

[36] B.Sadeg, Systèmes temps réel et Ordonnancement.

[37] N.MOUHOUB.Algorithmes de construction de graphe dans les problèmes .

[38] A.Sprecher & C.l.Hwang.Resource-constrained project scheduling :(exact methods for the multi-mode case).Springer,1994.

[39] S.H.Bokhari.On the Mapping Problem.IEEE Transactions on Comuters,30(3),1981.

[40] R.Buyya ,M.Murshed,D.Abramson,and S.Venugopal.Scheduling Parameter Sweep Applications on Global Grids : a deadline and Budget Constrained Cost-Time Optimization Algoritm .Software –Practice and Experience, 35(5),2005.

[41] M.Meddeber et Yagoubi Equilibrage de charge pour les grilles de calcule :classe des tàaches dépendant et indépendantes.

[42] Y.Bellbbas.Modèle d' »quilibrage de charge pour les grilles de calcul.Revue Africaine de la Recheche en Informatique et Mathémathique Appliqués .

[43] Jean. François Pillou , Load Balancing (équilibrage de charge) , Mai 2008.

[44] P.Warstein, H.Situ and Z.Huang, « Loadbalcanig in a cluster computer », In processing of the seventh International Conférence on parallel and Distributed Copmuting, Application and Technologies, IEEE.2010.

[45] R.K.Grace and R. Dynamic replica placement and selection strategies in data grids-a comprehensive survey. Journal of Parallel ans Distrubuted Computing, Vol. 74, no 2, pp. 2099-2018, 2014.

[46] Q. XU, Z. Xu, and T. Wang. A Data-Placement Strategy Based on Genetic Algorithm in Cloud Computing. Intenational Journal of Intelligence Scince, vol. 5, no 03 , PP. 145-157, 2015.

[47] M. Xu, W. Tian, and R. Buyya. A survey on load balancing algorithme for virtual machines placement in cloud computing. Concurrency and Compution : Practice and Experience journal, vol. 29, no 12, 2017.

[48] V. Asha, Bharath Kumar, and V. Girish. Load Balancing in Cloud Computing.International Journal of Recent Trends in Engineering and Research, vol. 4,no 3, 2018.

[49] Pooja R . Kathalkar1 and A. V . Deorankar. A Review on different load balancing Algorithm in cloud computing. International Research Journal of Engineering and Technologie (IRJET), vol. 5, no 2,2018.

[50] P. V. Krishna. Honey bee behavior inspired load balancing of tasks in cloud computing environment. Applied Soft Computing, vol. 13, no 5, pp. 2292-2303,2013.

[51] X. I. Zheng and L. A pareto based fruit fly optimization algorithm for task scheduling and ressource allocation in cloud computing environment. Evolutionary Computation (CEC), pp. 3393-3400, 2016.

[52] P.Mell and T. Grance.The NIST Définition of Cloud Computing .2011.

[53] K.Beghdad-Bey.Identification distribuée des empreintes digitales sur environnement de calcul hétérogènes via la prédiction de la chaerge CPU ; thése doctorat ;2010.USTHB .

Résumé

Le problème traité dans ce mémoire concerne l'équilibrage de charge dans les environnements cloud computing, la gestion d'une telle architecture reste assez difficile et l'un des problèmes majeur en informatique en nuage étant l'ordonnancement des tâches indépendantes dans ces environnements, et il est reconnu comme un problème NP-complet.

Dans ce travail, nous proposons des heuristiques d'ordonnancement de tâches en utilisant le principe d'équilibrage de charge, afin de minimiser le temps global d'exécution ou d'optimiser l'espace de stockage. Dans le cas du placement de donnée, on assure l'équilibrage de charge en distribuant les données sur l'ensemble des clusters et en répliquant le plus prés de l'utilisateur, afin de garantir un bon niveau de service.

Mot-clé : L'équilibrage de charge, Cloud Computing, Ordonnancement, SLB, LBE

Abstract

The problem addressed in this brief is about load balancing in cloud computing environments, the management of such an architecture remains quite difficult and one of the major problems in cloud computing is the scheduling of independent tasks in these environments, and it is recognized as an NP-complete problem.

In this work, we propose task scheduling heuristics using the load balancing principle, to minimize overall execution time or optimize storage space. In the case of data placement, load balancing is provided by distributing data across all clusters and replicating closer to the user to ensure a good level of service.

Keyword: Load balancing, Cloud Computing, Scheduling, SLB, LBE.

ملخص

المشكلة التي تمت معالجتها في هذا الموجز هي حول موازنة التح3.3ميل في بيئات الحوسبة السحابية ، وإدارة هذه البنية لا تزال صعبة للغاية وإحدى المهام الرئيسية في الحوسبة السحابية هي جدولة المهام المستقلة في هذه البيئات ، ومن -NP كاملة المسلم به كمشكلة

في هذا العمل، نقترح استدلال جدولة المهام باستخدام مبدأ موازنة التحميل، لتقليل وقت التنفيذ الكلي أو تحسين مساحة التخزين. في حالة وضع البيانات ، يتم توفير موازنة التحميل عن طريق توزيع البيانات عبر جميع المجموعات والنسخ المتماثل أقرب إلى المستخدم لضمان مستوى جيد من الخدمة.

الكلمة الرئيسية: موازنة التحميل ، الحوسبة السحابية ، الجدولة ، ، SLB ، LBE