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 : **Réseaux et Télécommunication**

Par :

BENSASSI Faiza

ARRARIA Saliha

Sur le thème

Ordonnancement temps réel des taches indépendantes sur environnements de cloud computing

Soutenu publiquement le... / 10/ 2020 à Tiaret devant le jury composé de :

Mr DJAAFRI Laouni	MCB	Université Ibn Khaldoun Tiaret	Président
Mr MEBAREK Bendaoud	MCA	Université Ibn Khaldoun Tiaret	Encadreur
Mr NASSANE Samir	MAA	Université Ibn Khaldoun Tiaret	Examinateur

2019-2020

Remerciement

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.

Ce travail n'a pu être mené à bien qu'avec le soutien de plusieurs personnes que je voudrais, à travers ces quelques lignes, remercier du fond du cœur.
nos remerciements les plus sincères vont à nos directeur de mémoire Docteur ***MEBAREK,Bendaoud*** *dont le sacrifice consentis a abouti à la réussite de notre master.*
nous adressons avec émotion, notre reconnaissance à nos parents ,nos sœurs et frères pour leurs soutiens sans faille.
Je tiens également à remercier toute personne À mes camarades de la promotion que j'ai servis avec humilité et avec lesquelles j'ai passé une scolarité exceptionnelle, riche d'enseignements, et d'expériences de rencontres, je veux ici dire ma sincère amitié.
Enfin, que toutes les personnes qui ont permis que ce travail voie le jour soient assurées de ma profonde reconnaissance.

Dédicace

Je dédie ce modeste travail :

A mes chers parents, pour tous leurs sacrifices, leur amour, leur tendresse, leur soutien et leurs prières tout au long de mes études,
A mon ami ***ghadir*** *qui m'a aidé dans ce travail*
Toute ma promotion de cette année et surtout ***bensahraoui fatima****.*

A mes enseignants de tous mon parcours.

Toute personne que j'aime.

ARRARIA Saliha

Dédicace

Je dédie ce modeste travail à :

Mes chers parents, quoi que je dise et quoi que je fasse je ne serais vous remercie comme il se doit. Votre soutien, et votre encouragement pour continuer mon parcours.

Mes sœurs et mon frère Abdelkarim.

Mes collègues de travail, qui m'ont soutenu durant ces années et j'apprécie leurs efforts.

Toute ma promotion de cette année et surtout ***bensahraoui fatima****.*

Mes enseignants de tous mon parcours.

Toute personne que j'aime.

BENSASSI Faiza

Résumé

Le Cloud computing sert des millions d'utilisateurs simultanément, il doit avoir la capacité de répondre à toutes les demandes des utilisateurs avec des performances élevées et une garantie de qualité de service. Par conséquent, nous devons mettre en œuvre des algorithmes d'ordonnancement de tâches approprié pour répondre équitablement et efficacement à ces demandes. Le problème d'ordonnancement des tâches est l'un des problèmes les plus critiques dans l'environnement de Cloud computing .

Il existe différents types d'algorithmes d'ordonnancement.

Dans ce travail, nous essayons de montrer les performances des six algorithmes d'ordonnancement de tâches et d'équilibrage de charge les plus populaires : premier arrivé, premier service (FCFS), SJF, Max-Min, Min-Min, SLB et LBE.

Mot-clé : L'équilibrage de charge, informatique en nuage, Ordonnancement, SLB, LBE.

Abstract

Cloud computing serves millions of users simultaneously, it must have the capacity to meet all user demands with high performance and guaranteed quality of service. Therefore, we need to implement an appropriate task scheduling algorithm to respond fairly and efficiently to these demands. The task scheduling issue is one of the most critical issues in the cloud computing environment, as the performance of the cloud mainly depends on it. There are different types of scheduling algorithms. In this work, we try to show the performance of the six most popular task scheduling algorithms: first come, first serve (FCFS), short first job scheduling (SJF), Max-Min, Min-Min, SLB and LBE.

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

ملخص

نظرًا لأن الحوسبة السحابية تخدم ملايين المستخدمين في وقت واحد، لهذا كان يجب أن تتمتع بالقدرة على تلبية جميع طلبات المستخدمين بأداء عالٍ وبجودة مضمونة. لتحقيق ذلك نحتاج إلى تنفيذ خوارزميات مناسبة لجدولة المهام للاستجابة بشكل عادل وفعال لهذه المطالب. تعد مشكلة جدولة المهام من أهم المشكلات في بيئة الحوسبة السحابية، حيث يعتمد أداء السحابة عليها بشكل أساسي. لتحقيق ذلك يوجد العديد من خوارزميات الجدولة.

في هذا العمل، نحاول إظهار أداء خوارزميات جدولة المهام الستة الأكثر شيوعًا: من يأتي أولاً، الخدمة الأولى (FCFS)، جدولة المهام الأولى القصيرة(SJF) ، Max-Min، Min-Min، SLBو.LBE

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

Liste des abbreviations

SOA: Service Oriented Architecture
EC2: Elastic Cloud computing
IBM: International Business Machines Corporation
NIST: National Institute of Standard and technology
PC: personal Computer
IT: Information Technology
CapEx: Capital Expenditures
OpEx: Operating Expenditures
GANTT: Generalized Activity Normalization Time Table
NWS: Network Weather Service
IaaS: Infrastructure as a Service
PaaS: Platform as a Service
S3: Simple Storage Service
SaaS: Software as a Service
CPU: Central Processing Unit
Makespan : Le temps d'exécution
OLB: Opportunistic Load Balancing
MET: Minimum Execution Time
MCT : Minimum Completion Time
AGs: Algorithmes génétiques
LS : Local Search
ANT: Ant colony algorithms
LBE: Loading Balancing Exchange
SLB: Simple Loading Balancing
ACO: The Ant Colony optimization
IA : Intelligence artificielle
ANN : Artificial Neural Network.
NP: non deterministic polynomial time
ETC: Estimated Completion Time
UML : Unified Modeling Language

Liste des figures

Liste des tableaux

Table de matière

Chapitre II :

L'ordonnancement des taches indépendantes dans l'environnement de cloud

Chapitre III :

Modélisation et algorithmes d'ordonnancement

Chapitre IV :

Implémentation, résultats & discussion

Introduction générale

L'**informatique dans le nuage** (en anglais, ***cloud computing***) est devenue un concept majeur faisant référence à l'utilisation de la mémoire et des capacités de calcul des ordinateurs et des serveurs répartis dans le monde entier et liés par un réseau, tel Internet.

Le Cloud computing permet de faire des allocations des ressources informatiques, mais les ressources ne sont pas toujours suffisantes. Elles ne peuvent pas souvent répondre aux besoins des utilisateurs. Alors les mécanismes d'ordonnancement des taches et d'allocation des ressources sont nécessaires pour améliorer des critères d'optimisation. Les nouvelles stratégies peuvent employer quelques concepts d'ordonnancement et d'allocation pour fournir un meilleur ordonnancement des taches et allocation des ressources.

L'ordonnancement de tâche dans un Cloud est un problème difficile. Ce problème est d'autant plus difficile lorsqu'il y a plusieurs facteurs à prendre en compte, donc c'est un problème d'optimisation combinatoire, ou il est possible de trouver la solution optimale en utilisant des métas heuristiques simples.

L'objectif principal de l'algorithme de l'ordonnancement des tâches est d'améliorer les performances et la qualité du service, ainsi que de maintenir l'efficacité des tâches et de réduire le coût. Dans l'ordonnancement des tâches, les ressources virtuelles disponibles sont utilisées de manière optimale. Grâce à l'ordonnancement efficace des ressources, la haute performance du cloud computing. Les divers paramètres pris en compte dans les algorithmes de l'ordonnancement sont le temps d'achèvement, le coût d'achèvement des tâches.

Nous allons intéresser dans ce mémoire au problème de l'ordonnancement dans le cloud, comparer les algorithmes de l'ordonnancement et choisir le plus optimal pour avoir une bonne qualité de service.

Dans cette section nous en représentons quelques algorithmes d'ordonnancement dans l'environnement de cloud :

Dans le premier chapitre : Nous présentons d'abord les concepts du Cloud computing et les notions fondamentales, les avantages et les limites, ensuite nous présentons les modèles de service de Cloud computing et nous terminons par les modèles de déploiement de Cloud computing.

Dans le deuxième chapitre : nous parlons des concepts liés à l'ordonnancement des tâches et l'allocation des ressources, ainsi que quelques algorithmes d'ordonnancement.

Le troisième chapitre sert à une modélisation des algorithmes avec des exemples de déroulements de ces heuristiques.

Le dernier chapitre : est consacré à la conception et l'implémentation de notre application

Chapitre I

Le cloud computing

1. Introduction

La naissance du Cloud Computing n'a pas une date précise, elle vient d'une évolution de certaines technologies telles que les web services, ou l'architecture SOA (Service Oriented Architecture). La notion de Cloud fait référence à un nuage tel qu'on a l'habitude d'utiliser dans les schémas techniques lorsqu'on veut représenter Internet, d'où la confusion entre Internet et les services du Cloud Computing.

L'informatique en nuage constitue l'une des avancées les plus importantes de l'informatique dans l'histoire. Même si certains principes fondamentaux sont présents depuis un certain temps, les progrès technologiques récemment réalisés facilitent la généralisation de l'informatique en nuage, en la rendant plus acceptable et surtout plus innovante, plus à même de relever les défis auxquels les professionnels des technologies de l'information et les chefs d'entreprise sont confrontés à l'heure actuelle.

Dans ce chapitre, nous donnons une présentation générale sur le Cloud Computing

2. Historique

Le mot "Cloud" est apparu au début des années 90 pour désigner des réseaux disposant d'un mode de transfert asynchrone. Le fournisseur "Salesforce.com" est le premier hébergeur de "Cloud" en 1999, suivi en 2002 par "Amazon" qui proposa un ensemble d'hébergements d'application, de stockage et d'offre d'emploi. "Amazon" développa ses services en 2005 (Amazon Web Services) et en 2006 EC2 (Elastic Cloud Computing), ce dernier est le premier service de "Cloud" réellement accessible.

C'est en 2009 que la réelle explosion du "Cloud" survint avec l'arrivée sur le marché de sociétés comme "Google" (Google App Engine), "Microsoft" (Microsoft Azure), "IBM" (IBM Smart Business Service) qui permet à ses utilisateurs de créer et de stocker des documents sur le Cloud. En 2010, "salesforce.com" lance sa base de données Cloud avec "Database.com" pour les développeurs, marquant ainsi le développement des services de Cloud Computing utilisables sur n'importe quel terminal, exécutables sur n'importe quelle plate-forme et écrits dans n'importe quel langage de programmation **[1]**.

3. Définition du cloud computing

Plusieurs définitions ont été proposées par différents auteurs. Mais la définition la plus couramment adoptée est celle du National Institute of Standards and Technology (NIST) formulée en ces termes: “cloud computing is a model for enabling ubiquitous, convinient, on-demand network access to a shared pool of configurable computing ressources that can be rapidly provisioned and released with minimal management effort or provider interaction” **[2]**.

Le *cloud computing* est une expression imagée désignant un ensemble de technologies matérielles et logicielles qui offrent à un utilisateur ou à une entreprise le moyen d'accéder en libre-service, n'importe quand et n'importe où, à des fichiers personnels, des applications logicielles opérationnelles ou toute autre ressource numérique au travers d'une infrastructure réseau fiable et sécurisée. Traduit en français par informatique dans les nuages, informatique dématérialisée ou encore informatique virtuelle, ce concept rend possible l'externalisation de la puissance de calcul et de stockage. Il permet de déporter, sur des serveurs distants, des traitements informatiques traditionnellement exécutés sur la machine locale de l'utilisateur.

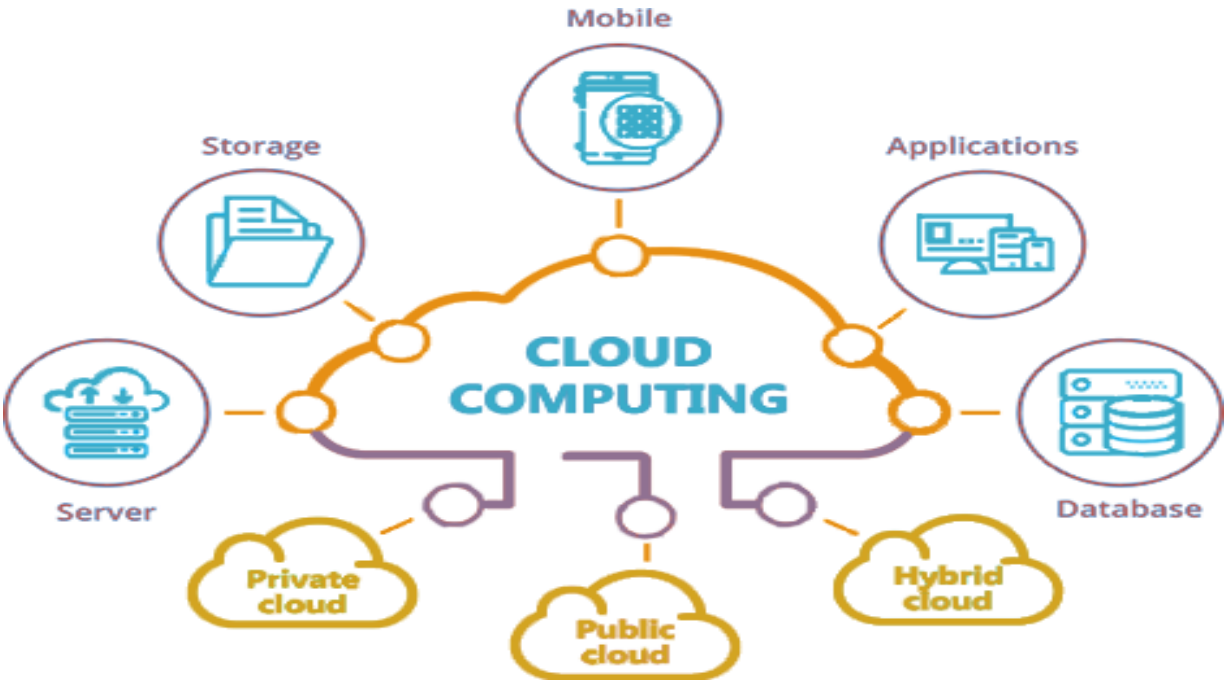

Figure 1: cloud computing

4. Les différents services du Cloud Computing

Les modèles de cloud sont de trois types : SaaS (Software as a Service), IaaS (Infrastructure as a Service) et PaaS (Platform as a Service). Chacun des modèles de cloud a son propre ensemble d'avantages qui pourraient répondre aux besoins de diverses entreprises. **[2]**

Pour choisir entre eux, il faut comprendre ces modèles de cloud, évaluer vos besoins et découvrir comment le modèle choisi peut fournir l'ensemble de flux de travail prévu. Voici une brève description des trois types de modèles cloud et de leurs avantages. Voire ***tableau 1***

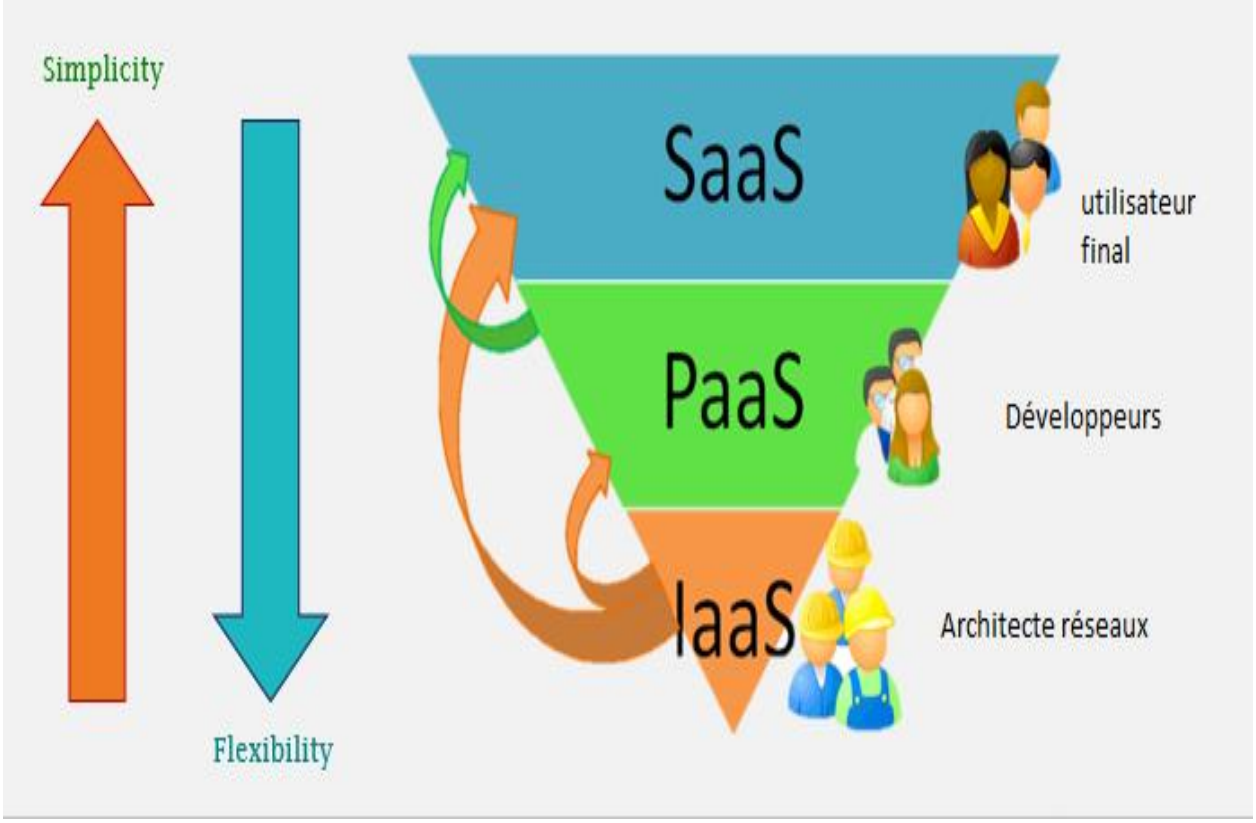

Figure 2: les services du cloud computing

4.1. SaaS (Software as a Service)

Ce type de service, des applications sont mises à la disposition des consommateurs. Les applications peuvent être manipulées à l'aide d'un navigateur web ou installées de façon locative sur un PC, et le consommateur n'a pas à se soucier d'effectuer des mises à jour, d'ajouter des patches de sécurité et d'assurer la disponibilité du service.

Gmail est un exemple de tel service. Il offre au consommateur un service de courrier électronique et le consommateur n'a pas à se soucier de la manière dont le service est fourni.

4.2. IaaS (Infrastructure as a Service)

Infrastructure as a Service est construit au-dessus de la couche de centre de données. IaaS permet la fourniture de stockage, le matériel, les serveurs et les composants réseau. Le client paie généralement sur une base par utilisation. Ainsi, les clients peuvent réduire les coûts que le paiement est seulement basé sur la quantité de ressources qu'ils utilisent vraiment. L'infrastructure peut être agrandi ou réduit de manière dynamique en fonction des besoins. Les exemples d'IaaS sont Amazon Elastic Cloud Computing et Simple Storage Service (S3).

4.3. PaaS (Platform as a Service)

PaaS est essentiellement une base cloud où vous pouvez développer, tester et organiser les différentes applications pour votre entreprise. La mise en œuvre de PaaS simplifie le processus de développement de logiciels d'entreprise. L'environnement d'exécution virtuel fourni par PaaS offre un espace favorable pour le développement et le test d'applications.

L'ensemble des ressources proposées sous forme de serveurs, de stockage et de mise en réseau est gérable soit par l'entreprise, soit par un fournisseur de plateforme. Google App Engine et AWS Elastic Beanstalk sont deux exemples typiques de PaaS. PaaS est également basé sur un abonnement qui vous offre des options de tarification flexibles en fonction des besoins de votre entreprise.

5. Avantages et Inconvénients des services

TYPES	AVANTAGES	INCONVENIENTS
SaaS	• Pas d'installation • Plus de licence • Migration	• Logiciel limite • Besoin de sécurité • Dépendance des prestataires • Services dédiés
PaaS	• Ne nécessite pas d'infrastructure • Pas d'installation • Environnement hétérogène	• Limitation des langages • Pas de personnalisation dans la configuration des MV
IaaS	• Administration • Personnalisation • Flexibilité d'utilisation	• Besoin de sécurité • Besoin d'un administrateur système

Tableau 1 : Avantages et inconvénients des services de cloud computing.

6. Les caractéristiques essentielles du Cloud computing

Cloud computing se développe rapidement que diverses organisations à réaliser le potentiel d'avoir des ressources flexibles de technologie de l'information à leur disposition. Cloud computing permet à ces organisations de profiter sans avoir à payer les coûts d'infrastructure habituellement associés à ces ressources. Pour vraiment être considéré comme le Cloud Computing, le modèle de nuage d'un fournisseur doit avoir au moins cinq caractéristiques essentielles.**[3]**

Le modèle Cloud Computing se différencie par les cinq caractéristiques essentielles suivantes :

6.1. 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.

6.2. Accès réseau large bande

Ces centres de traitement sont généralement raccordés directement sur le backbone Internet pour bénéficier d'une excellente connectivité. 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.

6.3. 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.

6.4. Redimensionnement rapide (élasticité)

La mise en ligne d'une nouvelle instance d'un serveur est réalisée 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 de bénéficier pleinement de la facturation à l'usage en adaptant la puissance de calcul au trafic instantané.

6.5. 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.

7. Modèles de déploiement d'un nuage

Les modèles de déploiement d'un nuage représentent la manière dont l'informatique en nuage peut être organisée en fonction du contrôle et du partage des ressources physiques ou virtuelles. Les modèles de déploiement d'un nuage comprennent : **[4]**

7.1. Nuage public

Modèle de déploiement du nuage où les services en nuage peuvent être mis à la disposition de tout client de services en nuage et où les ressources sont contrôlées par le fournisseur de services en nuage. Un nuage public peut être possédé, géré et exploité par une entreprise, un organisme universitaire ou gouvernemental ou par plusieurs d'entre eux. Il est situé dans les locaux du fournisseur de nuage.

7.2. Nuage privé

Modèle de déploiement du nuage où les services en nuage sont utilisés exclusivement par un seul client de services en nuage et où les ressources sont contrôlées par le client de services en nuage. Il peut être possédé, géré et exploité par l'organisme, une tierce partie ou les deux et peut être situé dans les locaux ou en dehors de ceux-ci. Le client de services en nuage peut également autoriser l'accès à d'autres parties pour son compte.

7.3. Nuage communautaire

Modèle de déploiement d'un nuage dans lequel les services en nuage prennent en charge exclusivement un ensemble défini de clients de services en nuage et sont utilisés en partage par ce même ensemble de clients, qui ont des exigences communes et une relation entre eux, les ressources étant contrôlées par au moins un membre de cet ensemble. Un nuage communautaire peut être possédé, géré et exploité par un ou plusieurs organismes de la communauté, une tierce partie, ou les deux et peut être situé dans les locaux ou en dehors de ceux-ci.

7.4. Nuage hybride

Modèle de déploiement d'un nuage utilisant au moins deux modèles de déploiement de nuages différents. Les déploiements impliqués restent des entités uniques, mais sont reliés entre eux par une technologie appropriée qui permet l'interopérabilité, ainsi que la portabilité des données et des applications.

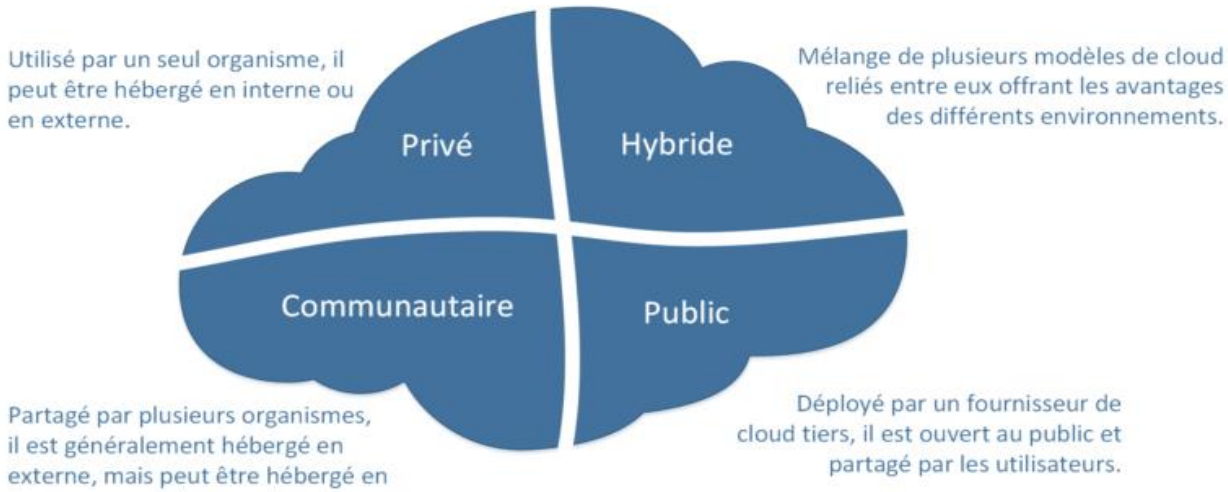

Figure 3: Les modèles de déploiement d'un nuage

8. Les principaux acteurs du Cloud Computing

Les fournisseurs de services de Cloud Computing sont des hébergeurs qui mettent à disposition des infrastructures physiques proposant une plate-forme de Cloud, nous citons quelques principaux acteurs **[5]**

Amazon Web Services : il s'agit d'une demi-douzaine de services, y compris l'Elastic Cloud Computing, pour la capacité de calcul et le Service de Stockage Simple (S3), pour la capacité de stockage à la demande, Amazon est un véritable innovateur en matière de calcul sur le web, offrant du paiement à l'usage sur des serveurs virtuels, et de l'espace de stockage.

Google : Google Apps est un ensemble d'outils de productivité de bureau, comprenant messagerie email, agenda, traitement de texte et un outil simple de création de sites web. App Engine, est une" plate-forme en tant que service" qui permet aux développeurs de bâtir et d'héberger des applications sur l'infrastructure de Google.

Le principal objectif de Google est l'exploration du web et de fournir de la publicité liée aux résultats de la recherche sur le web, l'incursion de Google dans le "logiciel en tant que service" pour les entreprises accélère le mouvement de l'industrie depuis les packages logiciels vers les services hébergés sur le web.

Microsoft : Windows Azure, une offre de "Windows en tant que plate-forme comme service" comprenant le système d'exploitation et les services pour les développeurs qui peuvent être utilisés pour construire et améliorer des applications Web hébergées.

9. La sécurité du Cloud Computing

La sécurité dans le cloud computing est un service en forte croissance qui fournit la plupart des fonctionnalités proposées par la sécurité informatique traditionnelle. Cela comprend notamment la protection des informations critiques contre le vol, la fuite de données et la suppression.

L'un des avantages des services de cloud est que vous pouvez opérer à échelle et conserver la sécurité. La sécurité du cloud computing s'apparente à la manière dont

vous gérez actuellement la sécurité, mais vous disposez désormais de nouvelles façons d'offrir des solutions de sécurité qui résout de nouveaux problèmes. La sécurité dans le cloud ne modifie pas l'approche en matière de gestion de la sécurité en passant de la prévention aux mesures de détection et de correction, mais vous donne la possibilité d'effectuer ces activités avec plus de souplesse

Vos données sont sécurisées dans des centres de données et, lorsque certains pays exigent que les données soient stockées dans leur pays, le choix d'un fournisseur possédant plusieurs centres de données peut permettre de répondre à cette exigence.

Le stockage de données est souvent soumis à des exigences en matière de conformité, notamment en ce qui concerne le stockage des numéros de carte de crédit ou des données de santé. De nombreux fournisseurs de services de cloud disposent de rapports d'audit réalisés par des tiers indépendants pour attester de l'existence de leur processus interne et gèrent efficacement la sécurité au sein des installations où sont stockées vos données.

10. Avantages du Cloud Computing

Le Cloud Computing simplifie les usages en permettant de s'affranchir des contraintes de l'outil informatique traditionnel (installation et mise à jour des logiciels, espace de stockage, portabilité des données…). Le Cloud Computing offre aussi plus d'élasticité et d'agilité car il permet d'accéder plus rapidement à des ressources IT (serveur, stockage ou bande passante) via un simple portail web et donc sans investir dans des équipements matériels supplémentaires. La mise à disposition est donc immédiate. De plus, l'utilisateur n'a pas d'infrastructure à gérer, c'est au fournisseur Cloud de maintenir le matériel serveur, le stockage, les réseaux.

Le Cloud Computing offre de multiples avantages aux entreprises et aux utilisateurs finaux. Voici les plus importants :

L'intérêt économique : plus besoin d'investir dans l'achat de logiciels ou plateformes coûteuses, le coût d'acquisition est transformé en loyer ce qui permet à l'entreprise de convertir ses dépenses d'investissement (Capex) en dépenses d'exploitation (Opex). Et surtout, ce loyer s'adapte au fur et à mesure des besoins de l'entreprise, à la hausse comme à la baisse.

La facilité d'administration : avec un service en cloud, la responsabilité de son fonctionnement repose sur l'hébergeur. L'entreprise n'a plus besoin de se soucier de l'hébergement du service, de sa maintenance logicielle, des mises à jour, de la sécurité du service, etc. L'entreprise peut ainsi bénéficier de conditions d'exploitation optimales (garanties de fonctionnement 24h/24, de sauvegardes, de support etc.) qu'elle n'aurait pas forcément les moyens d'assurer en interne.

La souplesse : grâce à la mutualisation du service entre de nombreux clients, le fournisseur du service cloud autorise une très grande souplesse. Ce qui rend simple et sans conséquence les évènements tels que la nécessité de monter/baisser en charge très rapidement, ou tout simplement ne plus utiliser le service.

Le gain de productivité : les entreprises peuvent davantage se concentrer sur l'utilisation du service plutôt que son administration. Cela permet de libérer des ressources humaines et de les concentrer sur le cœur de métier plutôt que sur des tâches périphériques.**[6]**

11. Les inconvénients du cloud

En dépit de ses nombreux avantages, le cloud computing présente aussi quelques inconvénients. Bien qu'il soit vrai que les informations sur le « nuage » peuvent être consultées à tout moment, ce système peut avoir parfois de graves dysfonctionnements. Vous devez être conscient du fait que cette technologie est toujours sujette à des pannes et à d'autres problèmes techniques. Même les meilleurs fournisseurs de services de cloud computing ont ce genre d'ennuis, malgré le maintien des normes élevées de maintenance. En outre, vous aurez besoin d'une très bonne connexion Internet pour accéder sur le serveur à tout moment. En cas de problèmes de réseau et de connectivité, il est impossible de travailler sur le cloud. L'achat de fichier client peut être la solution pour les limites du cloud et permet de disposer de ses données sans de contrainte particulière.

12. Conclusion

Dans ce chapitre nous avons données un aperçu général sur les concepts du Cloud, puis, nous avons défini ses services et ses types, ensuite, nous avons parlé des caractéristiques, avantages et inconvénients, enfin, nous avons cités quelques principaux acteurs du Cloud. Le chapitre suivant sera consacré à l'ordonnancements.

Chapitre II

L'ordonnancement des taches indépendantes dans le cloud

CHAPITRE II

1. Introduction

L'informatique dans le nuage ou le Cloud computing est un nouveau modèle de prestation de service informatique. Comme toute nouvelle technologie, elle a besoin cependant de nombreuses améliorations, des taches est souvent considéré comme un vrai challenge pour les gestionnaires dans ce type de technologies. L'ordonnancement des tâches et l'allocation des ressources dans le Cloud Computing sont considérés comme l'un des enjeux essentiels, plusieurs recherches ont été consacrées afin d'atteindre une gestion optimale des ressources au sein d'un Cloud Computing.

À travers cette partie nous allons présenter, quelques définitions, les principaux algorithmes d'ordonnancement des tâches dans le Cloud Computing.

2. Ordonnancement, définitions et concepts

On peut considérer l'ordonnancement des taches comme la gestion et la prise en charge d'un ensemble des taches de leur émission jusqu'à l'exécution. L'ordonnancement est un mécanisme de négociation entre deux objets, l'un représentant l'utilisateur (ou l'application) et l'autre les ressources. Les objets coté utilisateur sont les ordonnanceurs. **[07]**

Une autre définition, un ordonnancement constitue une solution au problème d'ordonnancement. 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. Un ordonnancement est très souvent représenté par un diagramme de GANTT

3. Problème d'ordonnancement

Le but de l'ordonnancement des tâches est de trouver un plan d'exécution optimal des tâches qui prend en considération leurs contraintes : les ressources, le budget, la date de fin, la performance, etc. En général, un problème contraint se compose de : tâches, ressources, conditions contraintes et une ou plusieurs fonctions objectives. Il

existe beaucoup d'algorithmes d'ordonnancement dans le Cloud computing. Les problèmes d'ordonnancement peuvent être classes en deux grandes catégories :

3.1. Les problèmes d'ordonnancement en ligne (online)

Pour lesquels la date d'arrivée (release date) des jobs n'est pas connue à l'avance ;

3.2. Les problèmes d'ordonnancement hors ligne (offline)

Pour lesquels les dates d'arrivées des jobs (généralement ils sont tous prêts à t=0 et toutes leurs caractéristiques sont connues avant l'ordonnancement. Ces problèmes ont été très largement étudiés pour les jobs séquentiels et pour les jobs parallèles. Les problèmes d'ordonnancement online sont généralement plus difficiles que les problèmes offlines, puisque nous ne connaissons qu'une partie des données du problème.

En effet, les décisions prises pour le placement ou l'exécution de tâches ne tiennent pas compte des données manquantes car on ne peut pas prévoir l'avenir.

Le processus d'ordonnancement se composent de tout ou partie des étapes suivantes: task prioritizing, ressource provisioning, allocation et enfin scheduling/mapping.

- **La phase task prioritizing :** établit l'ordre des tâches de départ leurs propriétés et leurs contraintes. Après cette phase, on a une liste ordonnée.
- **La phase resource provisioning/allocation :** réserve ou alloue un ensemble de ressources, c'est-à-dire qu'elle calcule le nombre de machines virtuelles pour l'ordonnancement des tâches.
- **La phase scheduling/mapping :** sélectionne les ressources parmi celles précédemment alloué qui permettent d'exécuter les tâches selon l'ordre prédéfinis. Quelle fait l'ordonnancement de chaque tâche à des ressources qui lui sont optimales.

4. Ordonnancement sur plates-formes hétérogènes de calcul

Sur une plate-forme hétérogène, le temps d'exécution d'une tâche dépend de la machine qui exécute cette dernière et potentiellement de l'instant de l'exécution. L'hétérogénéité entre deux machines peut provenir des caractéristiques des machines (processeurs de puissance différente, cartes mères différentes, . . .), mais aussi des applications qui sont exécutées par les machines. En fonction de la provenance de l'hétérogénéité, la modélisation est différente ; nous présentons ici différentes approches pour l'ordonnancement des tâches sur environnements hétérogènes.

Dans les environnements de ce type, chaque tâche a un temps d'exécution qui dépend de la machine. En fonction de la provenance de l'hétérogénéité et la modélisation du temps d'exécution change.

Dans le problème d'ordonnancement, l'objectif est d'obtenir l'exécution de l'ensemble des tâches le plus rapidement. Sur les environnements hétérogènes, la comparaison des dates de début d'une tâche ne donne pas directement d'information sur la date de fin de la tâche. Comme l'illustre ***la figure 4***, la tâche C commence plus tôt sur la machine m_1 mais se termine plus tard que si elle avait été ordonnancée sur la machine m_2.

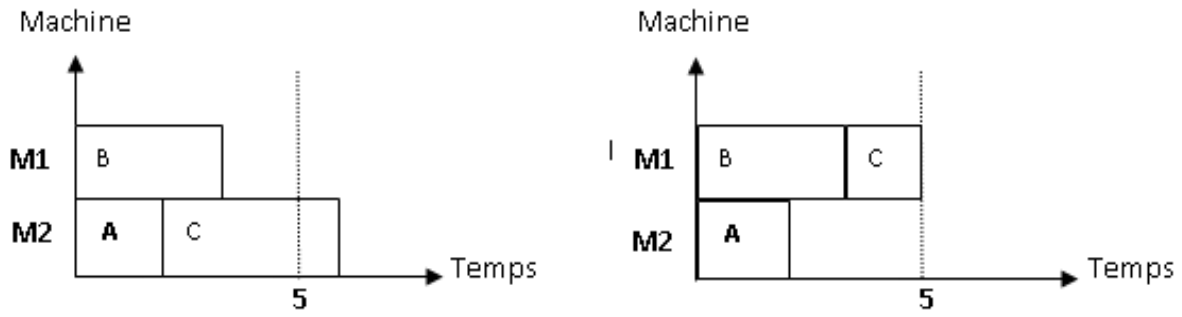

Figure 4: Illustration d'un ordonnancement sur plate-forme hétérogène

Dans un environnement hétérogène la propriété de dominance des ordonnancements sans attente est perdue : commencer les tâches le plus tôt possible, n'implique pas forcément d'obtenir le résultat le plus rapidement. Ainsi l'ordonnancement optimal inclut potentiellement des délais d'attentes. Certaines machines peuvent être inactives alors qu'il reste des tâches prêtes à s'exécuter.

5. Le rôle d'ordonnanceur

Un ordonnanceur (scheduler) devra trouver la machine la plus appropriée pour traiter les taches qui lui sont soumises. Les ordonnanceurs peuvent aller du plus simple (allocation de type round-robin) au plus compliqué (ordonnancement à base de priorités . . .). Les ordonnanceurs réagissent à la charge de la grille. Ils déterminent le taux de charge de chaque ressource afin de bien ordonnancer les prochaines taches. Ils peuvent être organisés en hiérarchie avec certains interagissant directement avec les ressources, et d'autres (méta ordonnanceurs) interagissant avec les ordonnanceurs intermédiaires. Ils peuvent aussi superviser le déroulement d'une tache jusqu'à sa terminaison, la soumettre à nouveau si elle est brusquement interrompue et la terminer prématurément si elle se trouve dans une boucle infinie d'exécution **[08]**.

6. Différences entre l'ordonnancement préemptif et non préemptif

Un ordonnanceur préemptif peut interrompre une tâche au profit d'une tâche plus prioritaire

Un ordonnanceur non préemptif n'arrête pas l'exécution de la tâche courante. **[09]**

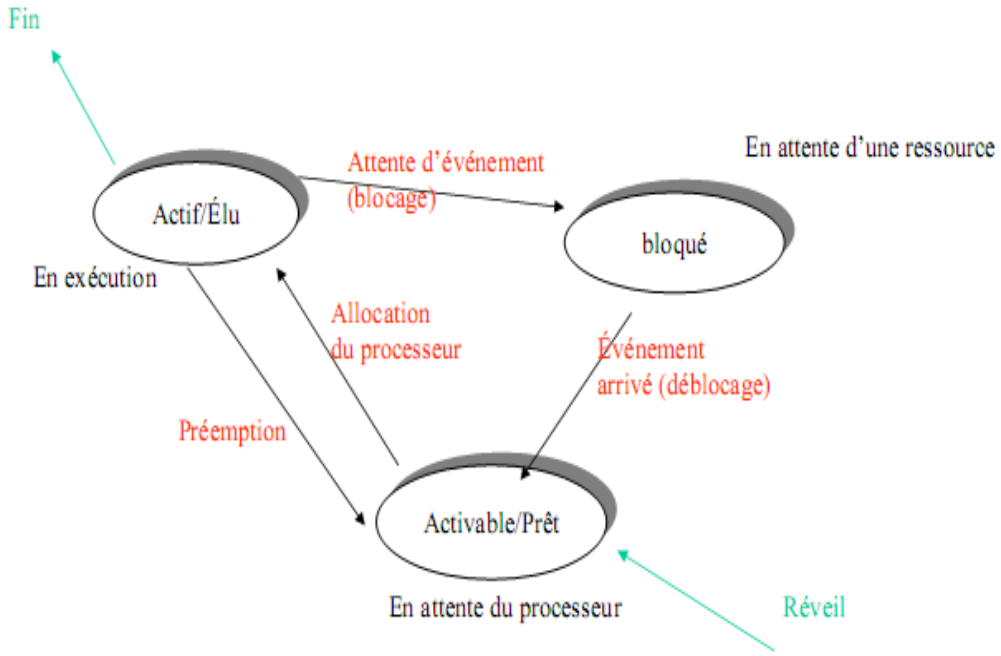

Figure 5: Comportement d'une tâche

7. Les stratégies d'ordonnancement

Lorsqu'on exécute plusieurs applications simultanément sur une plate-forme de calcul hétérogène, ils doivent se partager les ressources de calcul (processeurs) et de communication (bande-passante des liens réseau). Pour cela plusieurs stratégies d'ordonnancement ont été proposées.

7.1. La stratégie d'ordonnancement Chameleon

Un nouveau modèle d'ordonnancement Chameleon est proposé dans **[10]**, qui exécute une tâche sur un site tout en tenant compte de capacité de calcul et des coûts de communication. Cette stratégie considère à la fois la capacité des ressources de calcul et la disponibilité des données dans l'environnement de grille de données. Dans ce modèle, certains facteurs systèmes affectent la décision d'ordonnancement tel que la bande passante du réseau, le nombre de noeuds disponibles, etc. et des facteurs d'application spécifiques tel que : la taille des données d'entrée, la taille du code de l'application, la taille des données de sortie

produites. Dans cette approche, le coût est défini en fonction du temps de réponse. Le temps de réponse est séparé en deux parties, le temps de transmission de données et le temps d'exécution de tâche. Cette stratégie est constituée de cinq scénarios d'ordonnancement qui peuvent se produire.

Dans chaque scénario le coût est calculé en fonction du temps d'exécution. Cette approche tient compte à la fois de la capacité de calcul et de l'emplacement des réplicas de données,

Chameleon est construit au-dessus de Globus. NWS (Network Weather Service) est également utilisé pour mesurer et fournir une prévision d'une bande passante réseau. Chameleon utilise des services de Globus, tels que l'allocation des ressources et la prédiction de temps d'exécution d'une tâche sur chaque site et le temps de placement des données entre les sites pour estimer le temps de réponse global. Le temps d'exécution de tâche sur chaque site peut être calculé en comparant des informations systèmes de chaque site (CPU, mémoire, etc.). Le temps de transfert des données est prédit à partir de la bande passante du réseau entre les sites utilisant NWS.

7.2. Stratégies d'ordonnancement OLB, MET, MCT :

Le travail réalisé dans **[11]**. Consiste à comparer 11 stratégies d'ordonnancement, et certaines de ces derniers sont décrites ci-dessous :

7.2.1. OLB (Opportunistic Load Balancing (Algorithme d'équilibrage de charge opportuniste)

L'algorithme d'équilibrage de charge opportuniste permet de garder chaque nœud occupé. Il ne prend jamais en compte la charge de travail actuelle de chaque système. Quelle que soit la charge de travail actuelle sur chacun des nœuds, OLB distribue toutes les tâches inachevées à ces nœuds de manière aléatoire.

La tâche de traitement sera exécutée lentement en tant qu'OLB, elle ne calcule pas le calendrier d'implémentation du nœud en raison duquel certains goulots d'étranglement surviennent même lorsque certains nœuds sont libres.

7.2.2. MET (Minimum Execution Time) :

Il s'agit ici, pour chaque tâche adressée à l'ordonnanceur, de choisir la machine qui permettra de la réaliser le plus rapidement. MET ne prend pas en considération la charge des différents processeurs, il cause souvent le déséquilibre de charge entre les processeurs.

7.2.3. MCT (Minimum Completion Time) :

Il s'agit ici, pour chaque tâche adressée à l'ordonnanceur, de choisir la machine qui permettra de la réaliser en un minimum de temps possible **[12]** .

8. Les principaux algorithmes d'ordonnancement

L'objectif principal des algorithmes d'ordonnancement est de planifier des tâches sur des machines en fonction du temps d'adaptation, ce qui implique de trouver une séquence appropriée dans laquelle les tâches peuvent être effectuées sous les contraintes de la logique de transaction **[13]**

L'ordonnancement des tâches dans le cloud computing est un défi. Pour relever ce défi, nous passons en revue un nombre d'algorithmes d'ordonnancement des tâches.

L'ordonnancement des tâches de l'infonuagique consiste à répartir les tâches de calcul sur le regroupement des ressources entre différents utilisateurs de ressources en fonction de certaines règles d'utilisation des ressources dans des conditions de cloud données.

À l'heure actuelle, il n'existe pas de norme uniforme pour l'ordonnancement des tâches dans le cloud computing. La gestion des ressources et l'ordonnancement des tâches sont les techniques clés de l'informatique en nuage qui jouent un rôle essentiel dans une gestion efficace des ressources du cloud.

Les algorithmes d'ordonnancement de tâches et d'allocation des ressources suivants sont actuellement évoqués dans le contexte du cloud.

8.1. Algorithme Min-Min :

Min-Min commence par un ensemble de tâches qui sont toutes non assignées. Tout d'abord, il calcule le temps d'achèvement minimum pour toutes les tâches sur toutes les ressources. Puis, parmi ces temps minimums, la valeur minimale est sélectionnée, qui est le temps minimum entre toutes les tâches sur toutes les ressources. Ensuite, cette tâche est planifiée sur la ressource sur laquelle elle prend le moins de temps et le temps disponible de cette ressource est mise à jour pour toutes les autres tâches. Il est mis à jour de cette manière. Supposons qu'une tâche est affectée à une machine et que la tâche affectée dure 10 secondes, alors les temps d'exécution de toutes les autres tâches sur cette machine assignée seront augmentés de 10 secondes. **[14]**

Après cela, la tâche assignée n'est pas considérée et le même processus est répété jusqu'à ce que toutes les tâches aient des ressources affectées.

Un exemple d'application de l'algorithme pour 6 tâches et 4 machines virtuelles (***figure 06)***

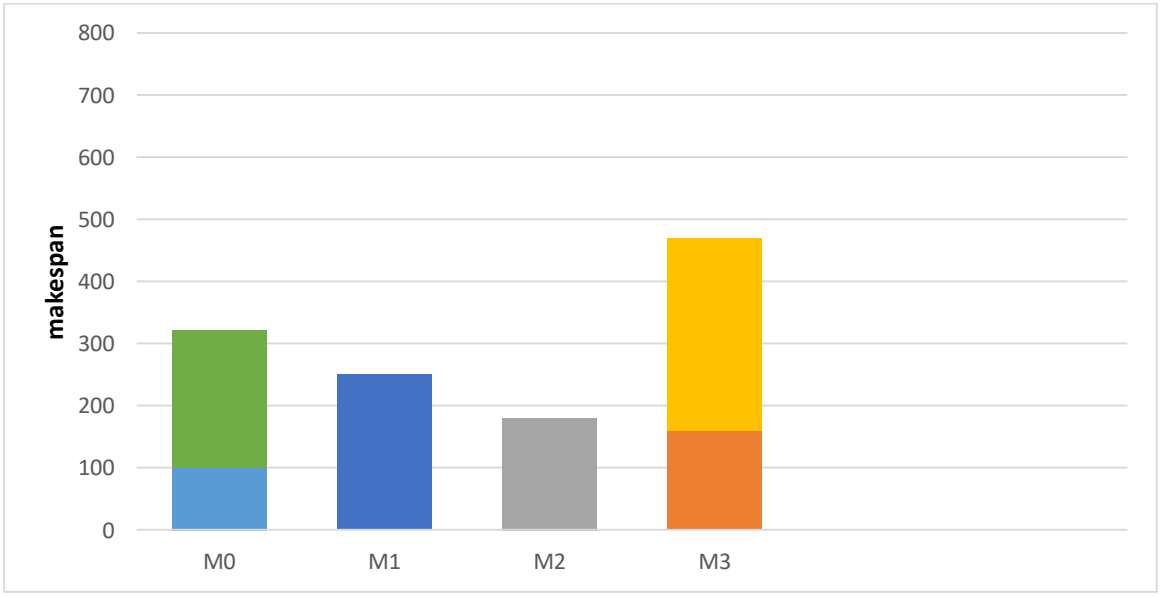

Figure 6: application de l'algorithme Min-Min

8.2. Algorithme Max-Min :

Max-Min suit le même principe que l'algorithme Min-Min excepté ce qui suit : dans ce cas, après avoir trouvé le temps de fin, les temps d'exécution minimum sont trouvés pour chaque tâche. Puis, parmi ces temps minimums, la valeur maximale est sélectionnée, qui est le temps maximum entre toutes les taches sur toutes les ressources. Ensuite, cette tâche est planifiée sur la ressource sur laquelle elle prend le moins de temps, puis le temps d'exécution de toutes les autres taches sont mise à jour sur cette machine. La mise à jour se fait de la même manière que pour le Min-Min. Toutes les tâches sont affectées aux ressources par cette procédure.**[14]**

Si nous appliquons la technique d'ordonnancement Max-Min, les tâches seront assignées aux machines comme indiqué dans ***la figure 7*** suivante :

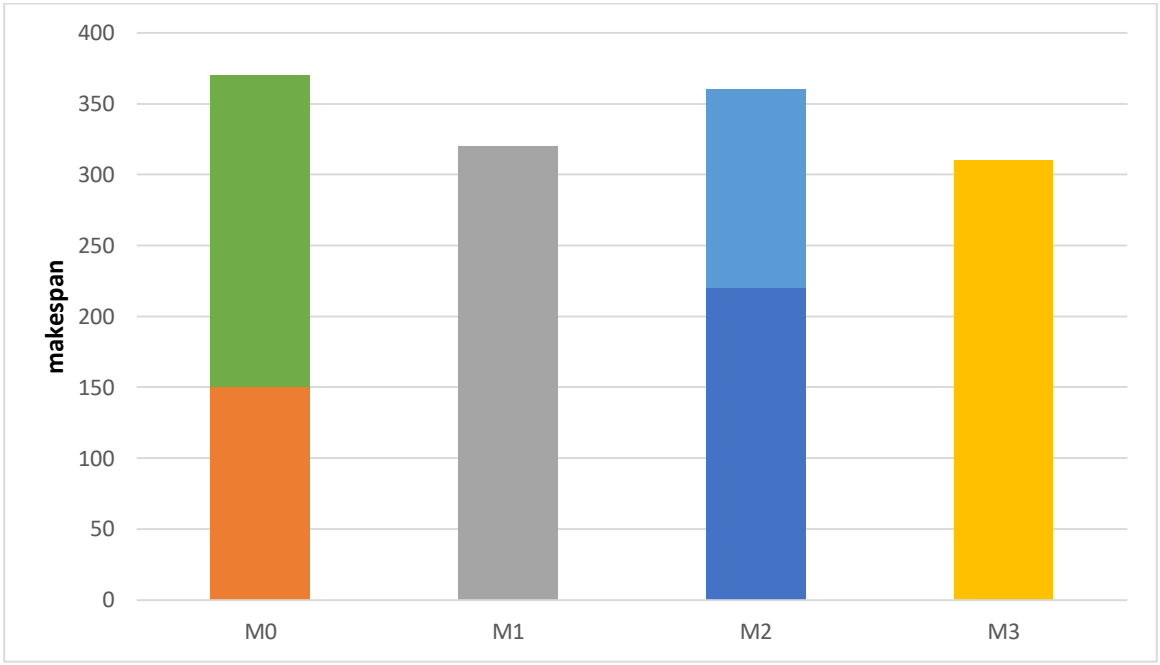

Figure 7: application de l'algorithme Max-Min

Le terme "makespan" dans les techniques d'ordonnancement Min-Min et Max-Min, comme expliquer auparavant est le temps d'exécution maximum sur n'importe quelle machine parmi les machines sur lesquelles les tâches sont planifiées. Par exemple, dans ***la Figure 6*** "470" est le rendu parce que c'est le temps d'exécution maximal entre les quatre machines.

Sur ***la figure 6*** et ***la figure 7***, l'axe des abscisses représente les différentes Machines et l'axe des y représente les temps d'exécution. Nous avons obtenu les différentes valeurs de makespan suivantes pour les deux techniques : Min-Min 470, Max-Min 370.

En fonction des différents temps d'exécution des tâches sur les ressources, une technique peut surpasser l'autre et l'affectation des ressources aux tâches peut changer, c'est-à-dire si une tâche est affectée à une machine si nous utilisons une technique, la même tâche peut être assignée à une autre machine si nous utilisons une autre technique.

8.3. Algorithme Round Robin :

Cet algorithme suit une stratégie simple qui consiste à distribuer de manière équitable les tâches sur les machines virtuelles disponibles, c'est –à-dire que le nombre de tâches pour chaque machine virtuelle est le même. Cet algorithme est implémenté dans le simulateur CloudSim **[15]**. L'algorithme Round Robin se concentre principalement sur la distribution de la charge de manière égale à toutes les ressources.

Le résultat de l'algorithme Round Robin montre un meilleur temps de réponse et un meilleur équilibrage de charge **[16]**.

8.4. Algorithme FCFS :

L'algorithme FCFS ‖First Come First Served " est tout simplement la politique FIFO (First in First out) c'est l'un des algorithmes les plus simples, la stratégie c'est d'ajouter chaque tâche et ressource dans une file et d'exécuter chaque tâche et ressource par ordre d'arrivée, cet algorithme est implémenté dans le simulateur CloudSim.

8.5. Algorithme SJF :

L'algorithme SJF « Short Job First » exécute les jobs selon l'ordre du temps d'exécution le plus court c'est-à-dire il choisit d'exécuter en premier la tâche qui sera la plus courte.

8.6. Earliest deadline First scheduling : ("échéance proche = préparation en premier")

Est un algorithme d'ordonnancement préemptif, à priorité dynamique, utilisé dans les systèmes temps réel. Il attribue une priorité à chaque requête en fonction de l'échéance de cette dernière selon la règle : Plus l'échéance d'une tâche est proche, plus sa priorité est grande. De cette manière, au plus vite le travail doit être réalisé, plus il a de chances d'être exécuté.

Cet algorithme est optimal pour tous types de système de tâches, cependant, il est assez difficile à mettre en œuvre et est donc peu utilisé. De plus, il ne prévoit aucun compromis "satisfaisant" en cas de surcharge du système.

9. Approches antérieures d'ordonnancement de tâches

9.1. Les algorithmes génétiques (AGs)

Parmi les métaheuristiques, les algorithmes évolutionnaires, et plus particulièrement les algorithmes génétiques, se sont distingués comme étant des techniques bien adaptées à la résolution de problèmes multi-objectifs notamment à cause de leur faculté à exploiter de vastes espaces de recherche et à générer des compromis multiples en une seule étape d'optimisation. Les algorithmes génétiques, initiés dans les années 1970 par John Holland, sont des algorithmes d'optimisation s'appuyant sur des techniques provenues de la génétique et des mécanismes d'évolution de la nature : croisement, mutation et sélection. Contrairement aux méthodes de recherche locale qui font intervenir une solution unique, les méthodes génétiques manipulent un groupe de solutions admissibles à chacune des étapes du processus de recherche. L'idée centrale consiste à utiliser régulièrement les propriétés collectives d'un ensemble de solutions distinguables, appelé population, dans le but de guider efficacement la recherche vers de bonnes solutions dans l'espace de recherche. La taille de la population reste constante tout au long du processus. Après avoir généré une population initiale de solutions, une méthode évolutive tente d'améliorer la qualité moyenne de la population courante en ayant recours à des principes d'évolution naturelle.

Les AGs travaillent sur un ensemble de points, appelés population d'individus. Chaque individu ou chromosome (chaîne binaire de longueur finie) représente une solution possible du problème donné. Il est constitué d'éléments, appelés gènes, dont les valeurs sont appelées allèles. On doit trouver une manière de coder chaque allèle différent de façon unique (établir une bijection entre l'allèle "réel" et sa représentation codée). Chaque paramètre d'une solution est assimilé à un gène, toutes les valeurs qu'il peut prendre sont les allèles de ce gène,

Un chromosome est une suite de gène **[15],** on peut par exemple choisir de regrouper les paramètres similaires dans un même chromosome (chromosome à un seul brin) et chaque gène sera repérable par sa position et chaque individu est représenté par un ensemble de chromosomes, et une population est un ensemble d'individus.

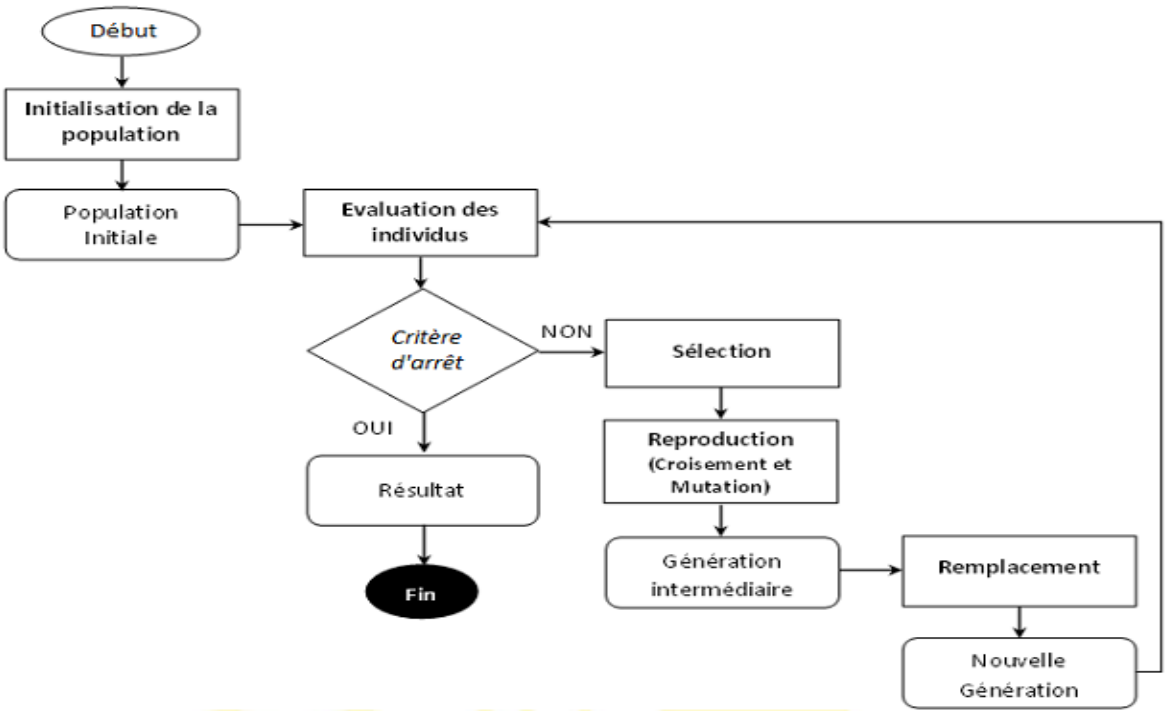

Figure 8: Étapes de l'algorithme génétique.

Yamada **[17]** a proposé une méthode directe consiste à utiliser un chromosome contenant les dates de fin de réalisation des opérations ainsi qu'un algorithme réalisant des ordonnancements actifs. Le chromosome utilisé peut-être vu comme une matrice à deux dimensions. Chaque ligne représente un job et chaque colonne les opérations. Chaque élément de cette matrice contient le numéro d'opération et la date de fin de réalisation de cette opération.

Graham Ritchie et John Levine **[18]** ont fait une recherche sur des algorithmes d'ordonnancement des tâches indépendante dans les environnements hétérogènes, et compris les algorithmes génétiques. Ils ont fait une comparaison entre l'AG et les deux heuristiques Min-min et Min-min+LS,

9.2. La descente stochastique avec la méthode KANGOUROU

La descente stochastique est une heuristique qui part d'un principe basé sur le voisinage d'une solution X. Le voisin Y obtenu est acceptée si et seulement si Y est meilleur que X. Le seul paramètre important d'un algorithme de ce type est la méthode de sélection ou de génération d'un voisin.

Comme cette heuristique n'autorise pas les voisins de moindre qualité que la solution en cours, la descente a en revanche le principal inconvénient connu sous le nom de "convergence prématurée". L'évolution se met à stagner et on n'atteindra alors jamais l'optimum, on restera bloqué sur un minimum local. Et pour trouver une solution à ce problème, les chercheurs ont proposé la méthode de kangourou.

La méthode de kangourou est publiée la première fois par Fleury en 1995 **[19]**. Elle étend et améliore la descente stochastique en autorisant la détérioration de la solution en cours pour ainsi éviter des blocages précoces dans des minimums locaux. Cet algorithme est basé sur l'affectation d'une descente stochastique, et l'acceptation d'une transition défavorable dans un voisinage de l'état actuel (pas nécessairement le même que celui utilisé pendant les descentes). Lorsque l'état minimal actuel est de même coût depuis trop longtemps ce qu'on appelle un saut et de cela l'algorithme prend son nom. Ensuite on débute une nouvelle descente stochastique.

L'avantage principal d'un algorithme du Kangourou est qu'il permet de minimiser des fonctions pouvant prendre des valeurs infinies. Lorsque l'algorithme se trouve dans un minimum local contenant un optimum local, il effectue après A itérations sans amélioration un saut. Il existe donc une probabilité non nulle que l'algorithme effectue un saut prématurément, c'est à dire qu'il peut exister un point (ou plusieurs) du minimum local (partiellement explorée) qui correspond à une valeur plus petite de la fonction objective.

des chercheurs ont fait des études sur l'application de cette méthode dans le domaine d'ordonnancement, précisément l'ordonnancement des activités de maintenance.

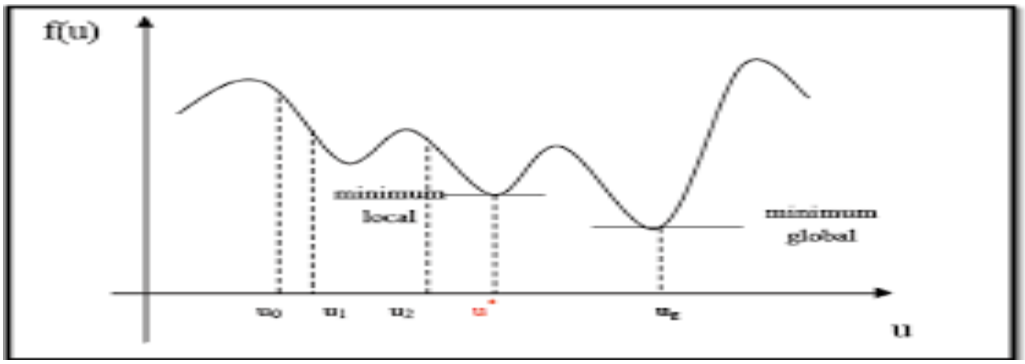

Figure 9: Comportement typique d'un algorithme de Kangourou

9.3. Les algorithmes TABOU

Des approches basées sur l'heuristique Tabou ont prouvé leur efficacité pour résoudre des problèmes d'ordonnancement. L'idée de la recherche Tabou est la suivante : à partir d'une donnée, on explore le voisinage et on choisit la position dans ce voisinage qui minimise la fonction objectif. Il est essentiel de noter que cette opération peut conduire à augmenter la valeur de la fonction : c'est le cas lorsque tous les points du voisinage ont une valeur plus élevée. C'est à partir de ce mécanisme que l'on échappe aux minima locaux. Le risque cependant est qu'à l'étape suivante, on retombe dans le minimum local auquel on vient d'échapper. C'est pourquoi, il faut que l'heuristique ait de la mémoire : le mécanisme consiste à

interdire, d'où le nom de Tabou, de revenir sur les dernières positions explorées. Les positions déjà explorées sont conservées dans une file (appelée souvent liste Tabou) d'une taille donnée, qui est un paramètre ajustable de l'heuristique. Cette file doit conserver des positions complètes, ce qui, dans certains types de problèmes, peut nécessiter l'archivage d'une grande quantité d'informations. Cette difficulté d'archivage peut être contournée en ne gardant en mémoire que les mouvements précédents.

A propos de l'utilisation des algorithmes Tabous pour l'ordonnancement dans Tracy D. Braun et al. **[20]** ont testé cette approche hybride avec l'heuristique Min-min pour plusieurs classes de problème.

9.4. Les algorithmes de colonie de fourmis (ANT)

Un algorithme de colonies de fourmis est un algorithme itératif à population où tous les individus partagent un savoir commun qui leur permet de guider leurs futurs choix et d'indiquer aux autres individus des directions à suivre ou au contraire à éviter.

Fortement inspirée du déplacement des groupes de fourmis, cette méthode a pour but de construire les meilleures solutions à partir des éléments qui ont été explorés par d'autres individus. Chaque fois qu'un individu découvre une solution au problème, bonne ou mauvaise, il enrichit la connaissance collective de la colonie. Ainsi, chaque fois qu'un nouvel individu aura à faire des choix, il pourra s'appuyer sur la connaissance collective pour pondérer ses choix.

Les algorithmes de colonies de fourmis sont nés à la suite de constatations faites sur le comportement des fourmis qui sont capables de trouver le chemin le plus court, du nid à une source de nourriture, et de s'adapter aux changements de l'environnement. Les biologistes ont, ainsi, étudié comment les fourmis arrivent à résoudre collectivement des problèmes trop complexes pour un seul individu, notamment les problèmes de choix lors de l'exploitation des sources de nourriture. Les fourmis sortent de leur nid pour chercher de la nourriture et déposent une substance chimique appelée phéromone tout au long de leur parcours. Les premières fourmis ayant trouvé la nourriture reviennent au nid et marquent deux fois leurs parcours.

Cette substance guide le choix des trajets que vont effectuer les autres fourmis par deux mécanismes qui sont le renforcement et l'évaporation de la phéromone, les fourmis sont capables de dépasser certains obstacles en faisant des ponts (constitué de plusieurs fourmis) et elles choisissent les branches les plus courtes, comme le montre ***la Figure 10***.

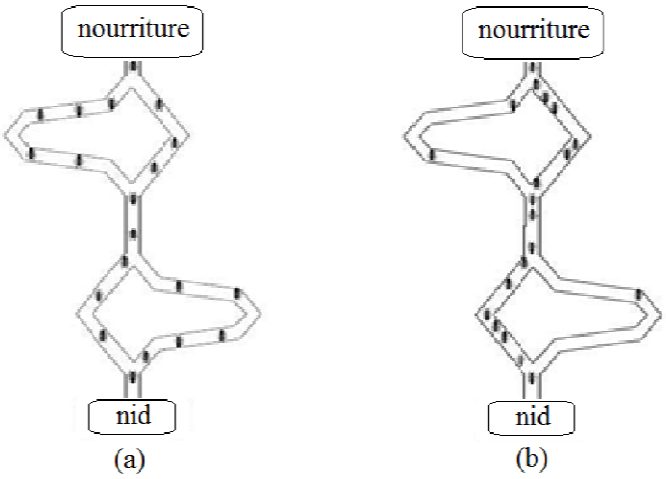

Figure 10: Sélection des branches les plus courtes par une colonie de Fourmis

Ce qui constitue une propriété très importante en optimisation qu'est celle de la convergence. Ces algorithmes ont donc pour but de reproduire le comportement naturel des fourmis pour retrouver le meilleur chemin possible vers un objectif donné. Cette approche est une méthode de diversification, elle consiste à attribuer à chaque fourmi une charge électrostatique. L'adaptation au domaine continu a été présentée par Krzysztof Socha dans **[21]**. L'algorithme itère sur une population de fourmis finie et constante de k individus, où chaque fourmi représente une solution ou une variable.

Les deux chercheurs Graham Ritchie and John Levine ont travaillé aussi sur l'utilisation des algorithmes de colonie de fourmis dans l'ordonnancement des tâches indépendantes **[18]**. Ils ont fait une comparaison entre l'ACO (*The antcolony optimisation*) et différentes autres approches appliquées.

10. Conclusion

Dans ce chapitre, nous avons abordé quelques concepts sur l'ordonnancement des tâches. Nous avons aussi mis l'accent sur quelques algorithmes de base de l'ordonnancement comme Min-Min, Round Robin, FIFO et SJF.

L'ordonnancement des tâches consiste à affecter des tâches aux ressources disponibles en fonction des propriétés et des conditions du travail.

C'est un aspect important du fonctionnement efficace du cloud, car il existe différents paramètres de tâche qui doivent être pris en compte pour l'ordonnancement approprié.

Chapitre III

Modélisation et algorithmes d'ordonnancement

1. Introduction

L‘ordonnancement est l'organisation de l'exécution des tâches dans le temps, les problèmes d'ordonnancement apparaissent dans tous les domaines de l'économie : l'informatique (tâches : jobs ; ressources : processeurs ou mémoire...), la construction (suivi de projet), l'industrie (problèmes d'ateliers, gestion de production), l'administration (emplois du temps). Dans l’environnement de cloud computing l’ordonnancement oppose plusieurs contraints, dans ce chapitre, on a une présentation générale des problèmes d'ordonnancement, de leur modélisation et des approches générales permettant de construire des solutions. On a encor des exemples des algorithmes et les déroulements étape par étape pour comprendre les résultats finals.

La méthode d’ordonnancement est une démarche et une succession d’étape pour mieux maitriser le déroulement d’un projet, la meilleure visibilité des utilisateurs sur certains résultats intermédiaires garantie à ce que le résultat soit attendu.

Lorsqu’on souhaite atteindre un objectif fixé dont la réalisation comporte un grand nombre des tâches successives.

Dans ce cas, on doit chercher à réaliser l’ensemble de tâches dans un ordre et délai. Tel que l’objectif fixé soit atteint dans le temps minimal. En faisant allusion au concept du projet, il faut une méthode à suivre **[22].**

Le problème d'ordonnancement est un problème de séquencement particulier dans lequel, en plus de choisir et d'ordonner un ensemble d'opérations ou tâches tout en satisfaisant un ensemble de contraintes, on doit allouer les opérations aux ressources et leur fixer des dates de début. En d'autres termes, le problème d'ordonnancement englobe les deux problèmes de séquencement et d'affectation. Du point de vue complexité, il a été démontré que ces deux problèmes sont fortement combinatoires et ils appartiennent à la classe des problèmes NP-difficiles.

2. Formulation du problème d’ordonnancement dans un environnement cloud

Un problème d’ordonnancement est caractérisé par deux ensembles :

- Un ensemble de n tâches T = {T1, T2……Tn}

– Un ensemble de m machines (ressources) M= {M1, M2…. Mm}.

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, 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 tj sur la machine mi, le temps d'exécution, ou bien la valeur de l'ETC (i, j) est règle à l'infini.

Nous allons présenter quelques algorithmes d'ordonnancement des tâches dans les Cloud Computing, ces algorithmes se sont focalisés essentiellement sur la rapidité d'exécution des tâches.

3. Les algorithmes proposés pour l'ordonnancement

Dans le cloud, de nombreux algorithmes de ordonnancement ont été proposés, nous en présentons quelques-uns ; avant en parlant des algorithmes de l'ordonnancement des tâches utilisés dans le cloud computing, nous ne pouvons pas négliger les algorithmes de base (voir chapitre II).

4. Les Heuristiques

Une heuristique est une stratégie de bon sens pour se déplacer intelligemment dans l'espace des solutions, afin d'obtenir une solution approchée, le meilleur possible, dans un délai de temps raisonnable.

En complément de la programmation mathématique, les heuristiques sont des méthodes de résolution purement algorithmiques qui permettent d'obtenir des solutions à n'importe quel problème décisionnel rapidement.

Une approche heuristique devrait être utilisée lorsque des méthodes de résolution exactes nécessitent des calculs énormes impraticables.

Les heuristiques sont également des règles ou des stratégies utilisées pour améliorer l'efficacité d'autre méthodes ou même d'autres approches heuristiques.

4.1. L'heuristique Max-Min :

Pseudo-Code for Max-Min Algorithm

1. **While** (Critère d'arrêt n'est pas satisfait) do
   {
2. Initialiser le temp_exe, à zéro ; //le temps d'exécution de chaque machine égale à la somme des temps d'exécution des taches qu'elles lui sont affectées.
3. **For** toutes les machines m_i
4. **For** toutes les tâches t_j
   {
5. Chercher max_{ij}, la valeur maximum de l'ETC ;
   }
6. Trouver le temps d'exécution de t_j qui a min_{ij}, dans la machine m_i ;
7. **For** toutes les machines m_i
   {
8. $temp\_exe_i = temp\_exe_i + ETC_{ij}$; // le temps d'exécution de le tâches qui a min_{ij}, dans la machine m_i.
   }

Figure 11: pseudo code de l'algorithme Max-Min

Soit l'exemple suivant pour mieux comprendre le déroulement de l'algorithme Max-Min avec une matrice simple de 08 taches et 04 machines :

Taches/Machines	M0	M1	M2	M3
T0	109	278	245	154
T1	163	230	262	141
T2	73	166	384	223
T3	252	359	388	246
T4	91	132	48	45
T5	216	176	148	154
T6	101	167	157	180
T7	130	83	33	42
Temps d'exécution	0	0	0	0

Tableau 2: Example de l'algorithme Max-Min

Iteration01 :

Chercher le temps d'exécution minimum dans chaque ligne :

Taches/Machines	M0	M1	M2	M3
T0	109	278	245	154
T1	163	230	262	141
T2	73	166	384	223
T3	252	359	388	246
T4	91	132	48	45
T5	216	176	148	154
T6	101	167	157	180
T7	130	83	33	42
Temps d'exécution	0	0	0	0

Tableau 3: déroulement de Max-Min (itération 01)

Chercher le Max entre les min => 246

Donc la taches T3 s'exécuté sur M3

Faire la mise à jour : pour toute cellule dans la colonne de M0 on ajoute 246

Alors on obtient notre nouvelle matrice comme suit :

Taches/Machines	M0	M1	M2	M3
T0	109	278	245	400
T1	163	230	262	387
T2	73	166	384	469
T3	252	359	388	0
T4	91	132	48	291
T5	216	176	148	400
T6	101	167	157	426
T7	130	83	33	288
Temps d'exécution	0	0	0	246

Tableau 4 : matrice ETC après la mise à jour (01)

Iteration2 :

Chercher le temps d'exécution minimum dans chaque ligne dans notre nouvelle matrice sans prendre en compte la ligne de la tache T3

Taches/Machines	M0	M1	M2	M3
T0	109	278	245	400
T1	163	230	262	387
T2	73	166	384	469
T3	252	359	388	0
T4	91	132	48	291
T5	216	176	148	400
T6	101	167	157	426
T7	130	83	33	288
Temps d'exécution	0	0	0	246

Tableau 5: déroulement de Max-Min (itération 02)

Chercher le Max entre les min => 163

Donc la tache T1 s'exécuté sur M0

Faire la mise à jour pour chaque cellule dans la colonne de M0 en ajoutant 163

Alors on obtient notre nouvelle matrice comme suit :

Taches/Machines	M0	M1	M2	M3
T0	272	278	245	400
T1	0	230	262	387
T2	236	166	384	469
T3	415	359	388	0
T4	254	132	48	291
T5	379	176	148	400
T6	264	167	157	426
T7	293	83	33	288
Temps d'exécution	163	0	0	246

Tableau 6: matrice ETC après la mise à jour (02)

Iteration3 :

Chercher le temps d'exécution minimum dans chaque ligne dans notre nouvelle matrice sans prendre en compte la ligne de la tache T1 et T3

Taches/Machines	M0	M1	M2	M3
T0	272	278	245	400
T1	0	230	262	387
T2	236	166	384	469
T3	415	359	388	0
T4	254	132	48	291
T5	379	176	148	400
T6	264	167	157	426
T7	293	83	33	288
Temps d'exécution	163	0	0	246

Tableau 7: déroulement de Max-Min (itération 03)

Chercher le Max entre les min => 245

Donc la tache T0 s'exécuté sur M2

Faire la mise à jour pour chaque cellule dans la colonne de M2 en ajoutant 245

Alors on obtient notre nouvelle matrice comme suit :

Taches/Machines	M0	M1	M2	M3
T0	272	278	0	400
T1	0	230	507	387
T2	236	166	629	469
T3	415	359	633	0
T4	254	132	293	291
T5	379	176	393	400
T6	264	167	402	426
T7	293	83	278	288
Temps d'exécution	163	0	245	246

Tableau 8: matrice ETC après la mise à jour (03)

On continue jusqu'à la fin des taches.

Résultat final :

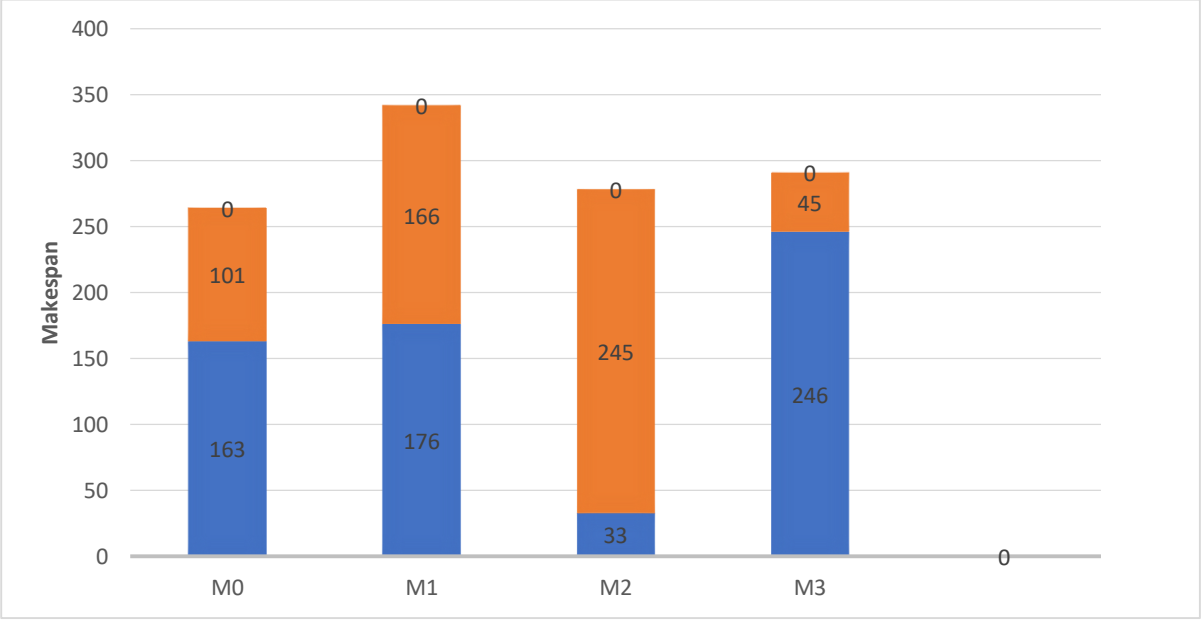

Figure 12: résultat final de l'algorithme Max-Min.

4.2. L'heuristique Min-Min :

Pseudo-Code for Min-Min Algorithm

1. **While** (Critère d'arrêt n'est pas satisfait) do
{
2. Initialiser le temp_exe, à zéro ; //le temps d'exécution de chaque machine égale à la somme des temps d'exécution des taches qu'elles lui sont affectées.
3. **For** toutes les machines m_i
4. **For** toutes les tâches t_j
{
5. Chercher min_{ij}, la valeur minimum de l'ETC ; // le minimum temps d'exécution de la tâche t_j dans la machine m_i.
}
6. **For** toutes les machines m_i
{
7. $temp\_exe_i = temp\_exe_i + ETC_{ij}$; // le temps d'exécution de le tâches qui a min_{ij}, dans la machine m_i.
8. }
9. For toutes les machines m
10. Trouver la machine qui a temp_exe minimum.

Figure 13: pseudo code de l'algorithme Min-Min

Soit l'exemple suivant pour mieux comprendre le déroulement de l'algorithme Min-Min avec une matrice simple de 06 taches et 04 machines :

Taches/Machines	M0	M1	M2	M3
T0	109	278	245	154
T1	163	230	262	141
T2	73	166	384	223
T3	252	359	388	246
T4	91	132	48	45
T5	216	176	148	154
T6	101	167	157	180
T7	130	83	33	42
Temps d'exécution	0	0	0	0

Tableau 9: Example de l'algorithme Min-Min

Itération 01 :

Chercher le temps d'exécution minimum dans la matrice => min = 33

Donc la tache T7 s'exécutée sur M2

Faire la mise à jour pour chaque cellule dans la colonne de M2 on ajoute 33

Alors on obtient notre nouvelle matrice comme suit :

Taches/Machines	M0	M1	M2	M3
T0	109	278	278	154
T1	163	230	295	141
T2	73	166	417	223
T3	252	359	421	246
T4	91	132	81	45
T5	216	176	181	154
T6	101	167	190	180
T7	130	83	66	42
Temps d'exécution	0	0	33	0

Tableau 10: déroulement de Min-Min (itération 01)

Itération 02 :

Chercher le temps d'exécution minimum dans notre nouvelle matrice sans prendre en compte la ligne de la tache T4 => min = 45

Donc la tache T4 s'exécutée sur M3

Faire la mise à jour pour chaque cellule dans la colonne de M3 on ajoute 45

Alors on obtient notre nouvelle matrice comme suit :

Taches/Machines	M0	M1	M2	M3
T0	109	278	278	199
T1	163	230	295	189
T2	73	166	417	268
T3	252	359	421	291
T4	91	132	81	90
T5	216	176	181	190
T6	101	167	190	225
T7	130	83	66	87
Temps d'exécution	0	0	33	45

Tableau 11: déroulement de Min-Min (itération 02)

Itération 03 :

Chercher le temps d'exécution minimum dans notre nouvelle matrice sans prendre en compte la ligne de la tache T4 et T7 => min = 73

Donc la tache T2 s'exécutée sur M0

Faire la mise à jour pour chaque cellule dans la colonne de M0 on ajoute 73

Alors on obtient notre nouvelle matrice comme suit :

Taches/Machines	M0	M1	M2	M3
T0	182	278	278	199
T1	236	230	295	189
T2	146	166	417	268
T3	325	359	421	291
T4	164	132	81	90
T5	289	176	181	190
T6	174	167	190	225
T7	203	83	66	87
Temps d'exécution	73	0	33	45

Tableau 12: déroulement de Min-Min (itération 03)

On continue jusqu'à la fin des taches.

Résultat final :

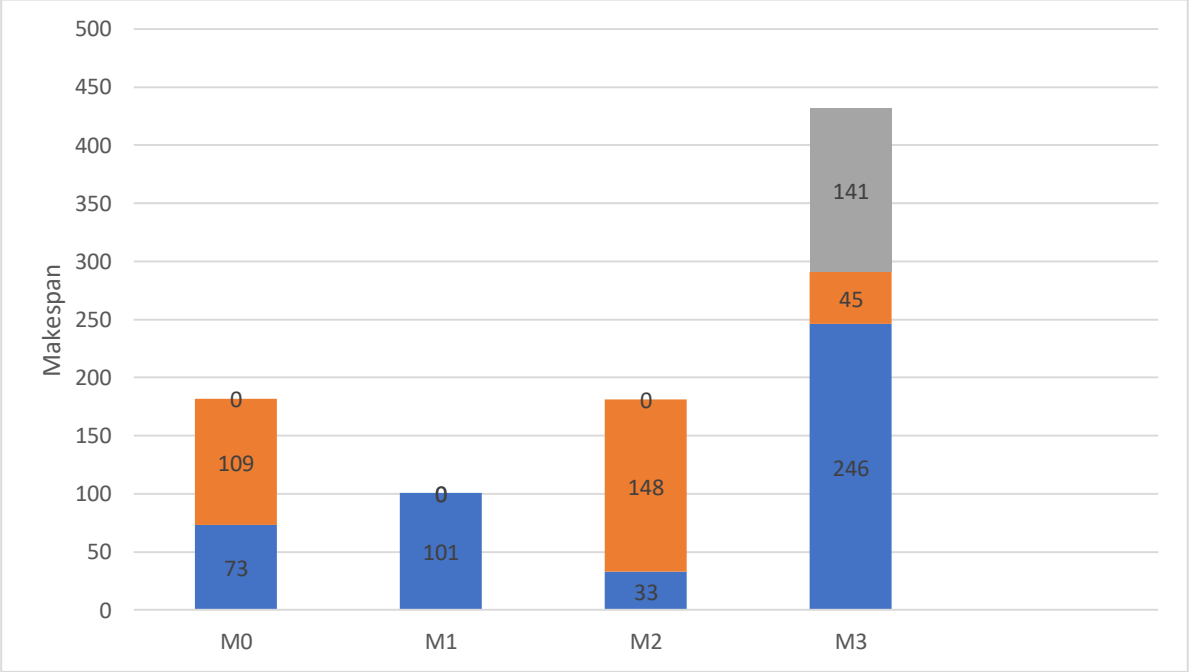

Figure 14: résultat final de Min-Min.

L'équilibrage de charge :

L'équilibrage de charge est le processus de distribution du trafic réseau sur plusieurs serveurs. Cela garantit qu'aucun serveur ne supporte trop de demande. En répartissant le travail uniformément, l'équilibrage de charge améliore la réactivité des applications. Il augmente également la disponibilité des applications et des sites Web pour les utilisateurs.

Figure 15: Rôle de l'équilibreur de charge

Toute machine participe à ordonnancement des tâches, elle a un temps maximal d'exécution. Toutefois la nécessité de minimiser ce temps permet de réduire le makespan tout en réaffectant les taches.

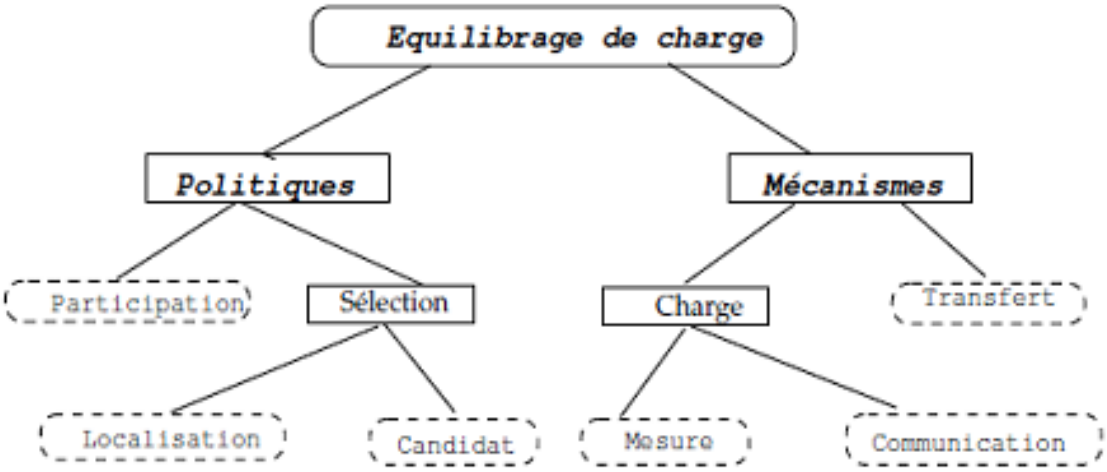

Figure 16: Composants d'un système d'équilibrage de charge

4.3. La méthode d'équilibre de charge SLB (Simple Loading Balancing)

Pseudo-Code for SLB Algorithm

1. **While** (critère d'arrêt n'est pas satisfait) do
   {
2. Initialiser un auxiliaire à 0 ; //pour faire le test ;
3. Trouver machine m_{max} avec le maximum temps d'exécution : ct_{max} ;
4. Trouver machine m_{min} avec le minimum temps d'exécution : ct_{min} ;
5. **For** toutes les tâches tj affectées sur mmax do
6. {Affecter tâche t_j sur m_{min};
7. $Cm_{minj} = Cm_{minj} + ETCm\_m_in_{j;}$ // Estimer temps d'exécution sur m_{min};
8. $Cm_{maxj} = Cm_{maxj} - ETCm\_m_in_j$; // Estimer temps d'exécution sur m max quand t_j est déplacée de m_{max};
9. **If** (($Cm_{minj} < ct_{max}$) and (Cm_{maxj} < auxiliaire) // pour avoir le minimum de Cm_{maxj}
   {
10. Affecter Cm_{minj} à l'auxiliaire ;
11. Sauvegarder j ; //sauvegarder la tâche qui donne le minimum temps d'exécution
    } }
12. Affecter la tâche tj qui donne le minimum temps d'exécution parmi toutes affectées sur la machine m_{max} // la tâche j que nous avons déjà sauvegardé ;
13. **Else**
14. Critère d'arrêt est satisfait ; //la valeur du makespan n'est pas changée.
    }

Figure 17: pseudo code de l'algorithme SLB

La méthode d'équilibre de charge SLB (Simple Loading Balancing) préconise la réduction du temps d'exécution en transférant les tâches 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 makespan.

4.4. L'équilibrage de charge avec échange LBE (Loading Balancing with Excange)

Pour améliorer les performances de l'heuristique SLB, une nouvelle technique d'équilibrage de charge est proposée, c'est l'équilibrage de charge avec échange LBE (Loading Balancing with Excange). Cette technique généralise l'ordonnancement SLB. Son principe est de minimiser le temps d'exécution maximal obtenue par l'ordonnancement, en réattribuant les tâches 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 tâches avec les autres machines, et par la réaffectation des tâches 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 tj et de la tâche t_k. Nous réaffecterons une seule paire d'échanges des tâches en même temps qui donne le makespan minimum parmi toutes les affectations de tâches entre toutes les machines.

Pseudo-Code for SLB Algorithm

1. **While** (critère d'arrêt n'est pas satisfait) do
2. {Trouver machine m_{max} avec le maximum temps d'exécution :ct_{max} ;
3. **For** toutes les tâches t_j affectées sur m_{max} do
{
4. **For** toutes les machines mi avec $m_i \neq m_{max}$ do
{
5. **For** toutes les tâches t_k affectées sur m_i do
{
6. Echange tâche tk avec t_j ;
7. C_{ij}= W_i+ ETC_{ij} - ETC_{ik} ; // temps d'exécution de mi quand t_j est affectée sur m_i.
8. Cm_{maxk}=Wm_{max}+ $ETCm_{maxk}$- $ETCm_{maxj}$; // temps d'exécution de m_{max} quand
9. t_k est affectée sur m_{max} .
10. **If** (C_{ij}<ct_{max}) and (cm_{maxk}<ct_{max})
{
11. ct_{max} = cm_{maxk},
12. 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}
}
}
}
13. Echange tâche t_k avec t_j de m_{max} qui donne minimum temps d'exécution parmi toutes les tâches affectées sur la machine m_{max} ;
}
14. **If** (makespan n'est pas minimisé)
15. Critère d'arrêt satisfait;
}

Figure 18: pseudo code de l'algorithme LBE

Nous allons ordonnancer les taches de l'exemple précédent par la matrice ETC initialisé par l'algorithme Min-Min ou la valeur du makespan sur **M3** est de **432** La première itération de l'algorithme a réaffecté la tache **T1** de la **M3** à la

M1 comme indiqué dans le graphe ci-dessus cette réaffectation permet de réduire le makespan de **432** à **308**

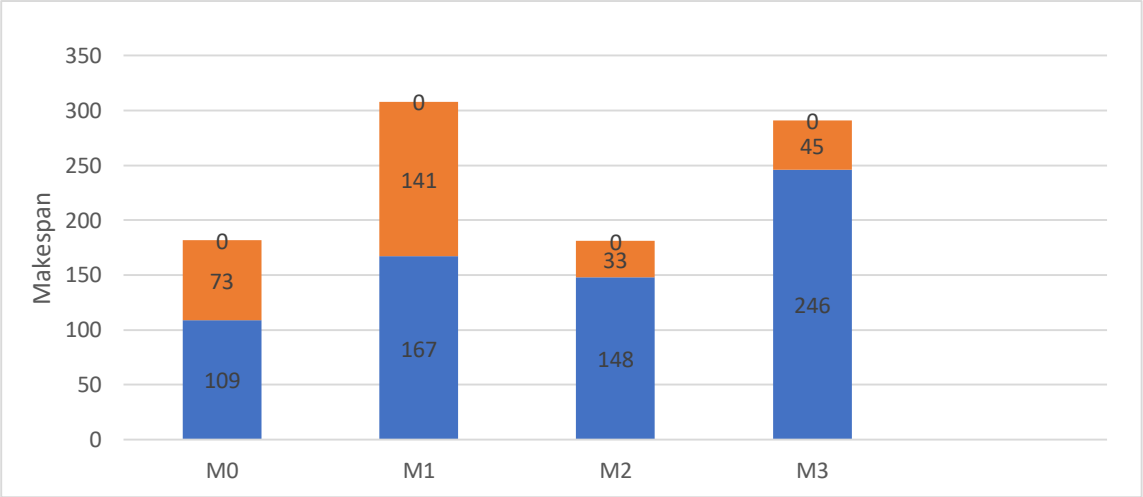

Figure 19: la première et la dernière itération de SLB.

5. Conclusions

Formaliser un problème d'ordonnancement est facile ; il est en général extrait d'une situation concrète, on envisage des tâches, des machines, des durées...

Les algorithmes de l'ordonnancement sont basés sur des heuristiques afin de réaliser un ou plusieurs objectifs tels que : la réduction du temps d'exécution moyen, l'économie d'énergie, la fiabilité d'exécution et la réduction de la date de fin de la dernière tâche (makespan). Mais chacune de ces heuristiques n'est pas sans défauts.

Des nombreuses recherches ont tenté de résoudre le problème de l'ordonnancement des tâches dans le cloud computing. La plupart d'entre eux utilisent des techniques d'intelligence artificielle telles que l'algorithme génétique et la colonie de fourmis pour résoudre le problème de l'ordonnancement des tâches et trouver la répartition optimale des ressources. En fin L'utilisation de réseaux neuronaux artificiels aura un fort potentiel pour résoudre et optimiser les problèmes de l'ordonnancement dans un environnement de cloud computing.

Chapitre IV

Implémentation, résultats et discutions

1. Introduction

Dans la première partie de ce chapitre, nous allons faire la conception de notre application en utilisant le langage UML. Cependant notre application n'est pas d'une grande complexité afin d'utiliser la totalité des diagrammes d'UML, nous n'utiliserons que les diagrammes que nous trouverons nécessaires.

La deuxième partie de chapitre consacré à la présentation de la mise en œuvre de notre application, nous commençons tout d'abord par une présentation du langage de programmation choisi. Ensuite nous montions les détails de notre application. On se termine ce chapitre par une synthèse de nos résultats obtenus.

2. La conception d'application

Nous avons choisi l'UML pour modéliser notre application grâce à ces caractéristiques. Simplifie et efficace, pour comprendre la démarche de notre application, elle est présentée par des diagrammes d'une manière bref et claire.

3. L'objectif de notre application

Notre objectif dans cette application est de minimiser la pénurie de ressources et d'assurer l'équité entre les parties utilisant les ressources. L'ordonnancement traite du problème de la décision sur laquelle des demandes en suspens doivent être allouées.

4. Présentation d'UML :

UML, c'est l'acronyme anglais pour « Unified Modeling Language ». On le traduit par « Langage de modélisation unifié ». La notation UML est un **langage visuel** constitué d'un ensemble de schémas, appelés des **diagrammes**, qui donnent chacun une vision différente du projet à traiter. UML nous fournit donc des diagrammes pour **représenter** le logiciel à développer : son fonctionnement, sa mise en route, les actions susceptibles d'être effectuées par le logiciel, etc.

1. Utilité de l'UML

Fournir aux concepteurs de systèmes, ingénieurs logiciels et développeurs de logiciels des outils pour l'analyse, la conception et la mise en œuvre de systèmes

logiciels, ainsi que pour la modélisation de processus métier et d'autres processus similaires.

Faire progresser l'industrie en permettant l'interopérabilité des outils de modélisation visuelle orientés objet. Toutefois, pour permettre un échange significatif d'informations de modèles entre outils, il est nécessaire de trouver un accord sur la sémantique et la notation.

5. Implémentation de l'application

5.1. Présentation du langage de programmation

C# est un langage orienté objet élégant et de type sécurisé qui permet aux développeurs de créer une variété d'applications sécurisées et fiables qui s'exécutent dans l'écosystème .NET. L'écosystème .NET est constitué de toutes les implémentations de .NET, y compris mais non limitées à .net Core et .NET Framework.

Le C# sa syntaxe ressemble beaucoup au langage Java de Sun Microsystems et au C++, vous pouvez l'utiliser pour créer des application clientes Windows, services Web XML, composants distribué, applications client-serveur, application de base de données et bien plus encore.

5.2. Environnement de développement

L'environnement de développement intégré Visual Studio est une zone de lancement de création que vous pouvez utiliser pour modifier, déboguer et créer du code, puis publier une application. Un environnement de développement intégré (IDE) est un programme riche en fonctionnalités qui peut être utilisé pour de nombreux aspects du développement logiciel. Au-delà de l'éditeur et du débogueur standard fournis par la plupart des IDE, Visual Studio comprend des compilateurs, des outils de complétion de code, des concepteurs graphiques et de nombreuses autres fonctionnalités pour faciliter le processus de développement logiciel.

5.3. Diagramme de cas d'utilisation

Identification des acteurs : Dans notre application on a un seul acteur humain.

Identification des cas d'utilisation : Après avoir défini l'acteur principal de l'application, nous passerons à la détermination de ces cas d'utilisation qui illustrent

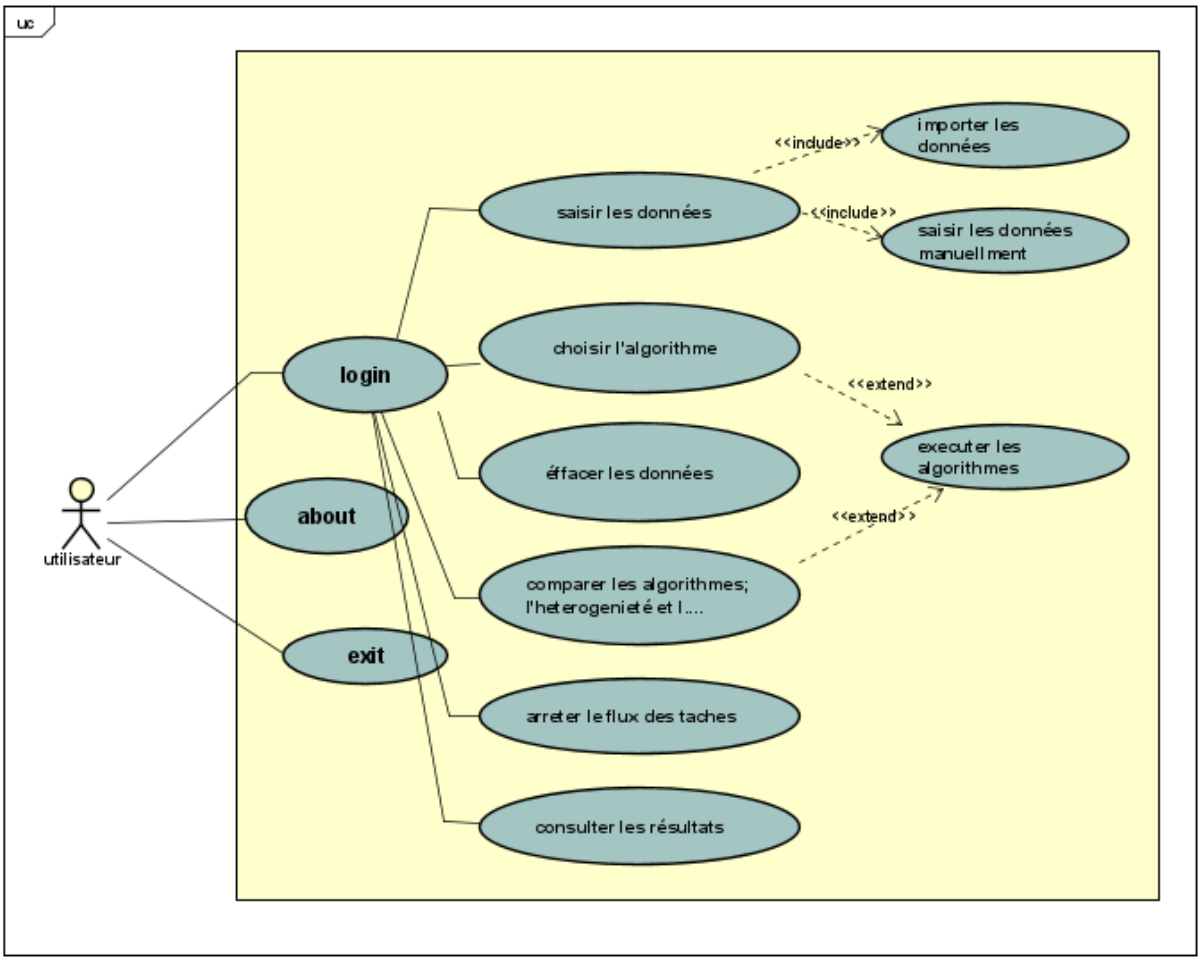

Figure 20:Diagramme de cas d'utilisation.

le comportement du système en réponse à une interaction avec l'utilisateur.

5.4. Diagramme de séquence

- **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 «**login** »

L'assistant affiche la fenêtre d'exécution

L'utilisateur clique le bouton « import data file »

L'assistant affiche la matrice de données.

L'utilisateur choisit l'algorithme.

L'assistant affiche les résultats.

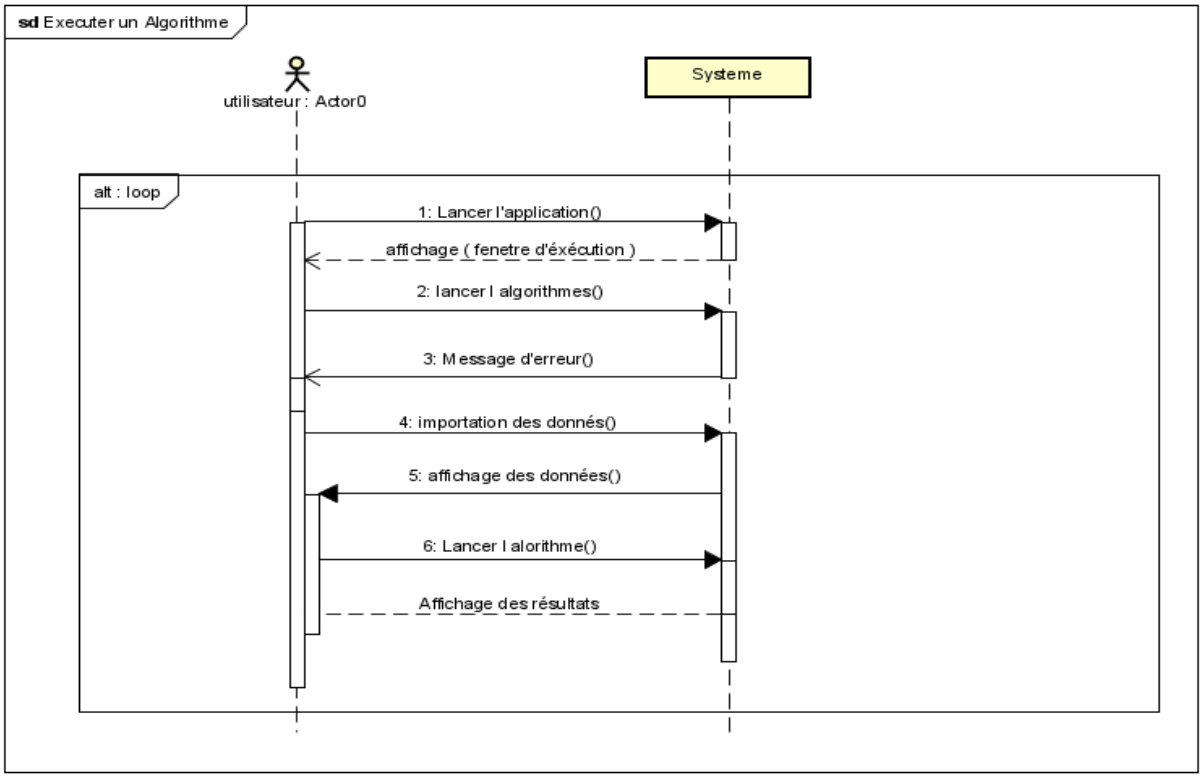

Figure 21::DDS exécution de l'algorithme

Alternatives :

L'utilisateur peut à tout moment quitter.

L'utilisateur peut modifier ses données

L'utilisateur peut choisir un autre algorithme

L'utilisateur peut importer les données en cliquant sur le bouton (import data file).

- **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 « import data file ».

L'assistant affiche la matrice des données.

2 -ème scenario

L'utilisateur introduit les données

Alternatives :

L'utilisateur peut à tout moment quitter.

L'utilisateur peut modifier ses données. L'utilisateur peut importer les données en

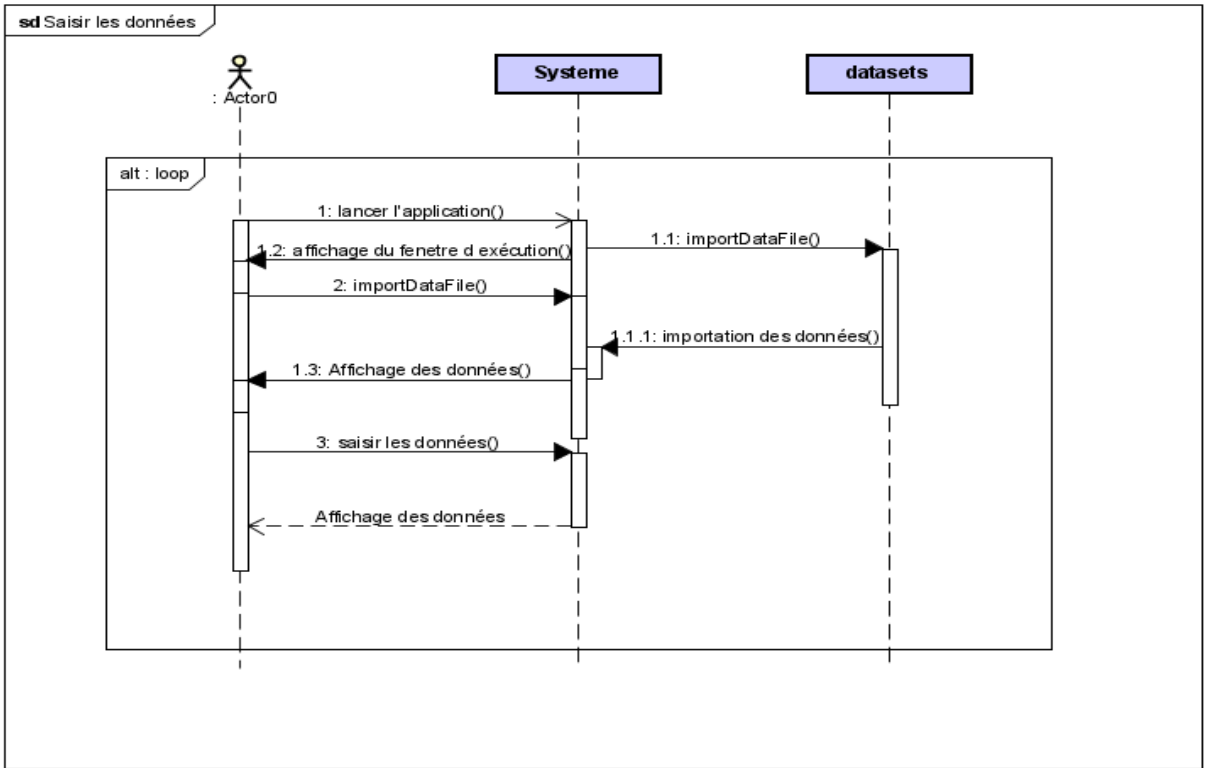

Figure 22::DDS saisir les données

cliquant sur le bouton (import data file).

- **Cas d'utilisation :** Consulter l'aide**.**

Acteur : Utilisateur.

Pré conditions : L'utilisateur lance l'application et atteint la fenêtre « Aide ».

Post conditions : Consulter l'aide

Scénario nominal :

L'utilisateur clique sur l'onglet « about ».

L'assistant affiche la fenêtre de l'aide.

L'utilisateur consulte les différentes sous catégories de l'aide.

Alternatives :

L'utilisateur peut à tout moment quitter.

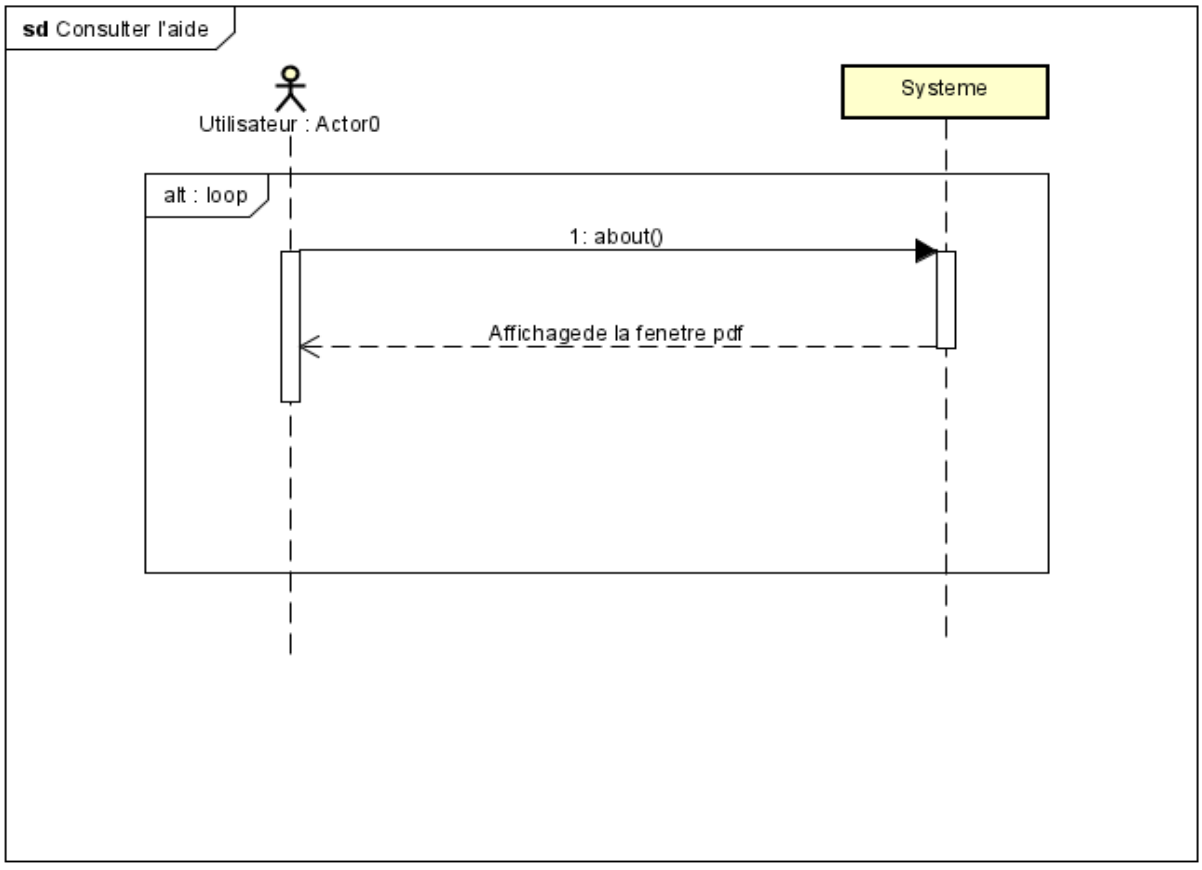

Figure 23: DDS consulter l'aide

6. Présentation de l'application :

L'interface Login : représente la fenêtre d'accueil de notre application.

Figure 24: fenêtre d'accueil

Cette fenêtre contient 03 bottons :

Botton About : ouvrir notre mémoire sous forme d'un PDF pour mieux comprendre l'application.

Botton Exit : pour quitter l'application.

Botton Login : ouvrir la fenêtre d'ordonnancement suivante :

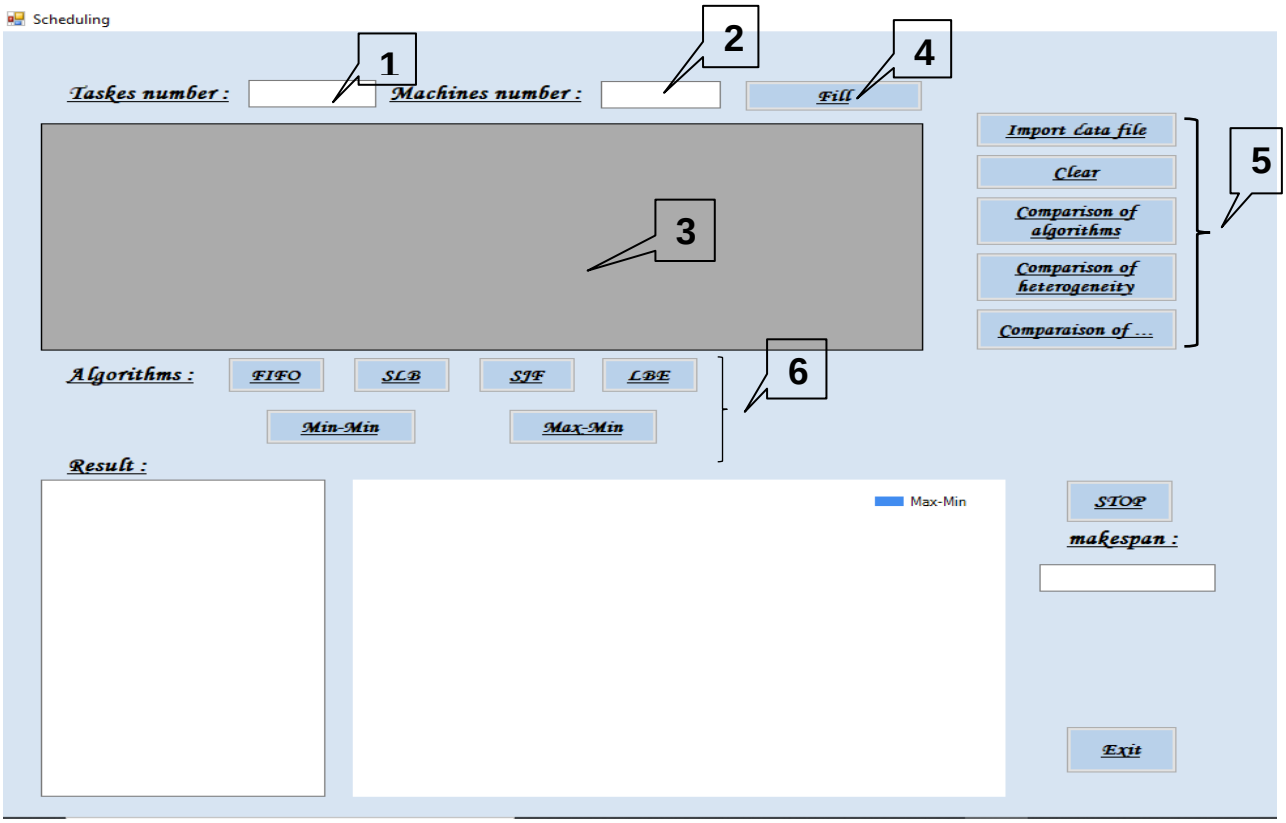

Figure 25: fenêtre d'ordonnancement.

La page (fenêtre d'ordonnancement) permet à l'utilisateur de saisir ses données, à savoir, le nombre de tâches (1), le nombre de machines (2) et le temps d'exécution de chaque tâche dans chaque machine (3). Cette interface donne le choix à l'utilisateur de saisir les données manuellement par le Botton Fill (4) ou par l'importation des données à partir d'un fichier en utilisant le Botton Import data file (5). Dans ***la figure 26*** un exemple de remplissage de l'ETC manuellement, pour 5 machines et 10 taches. (L'utilisateur doit saisir le nombre de taches et de machines dans les champs Tasks number et Machines number).

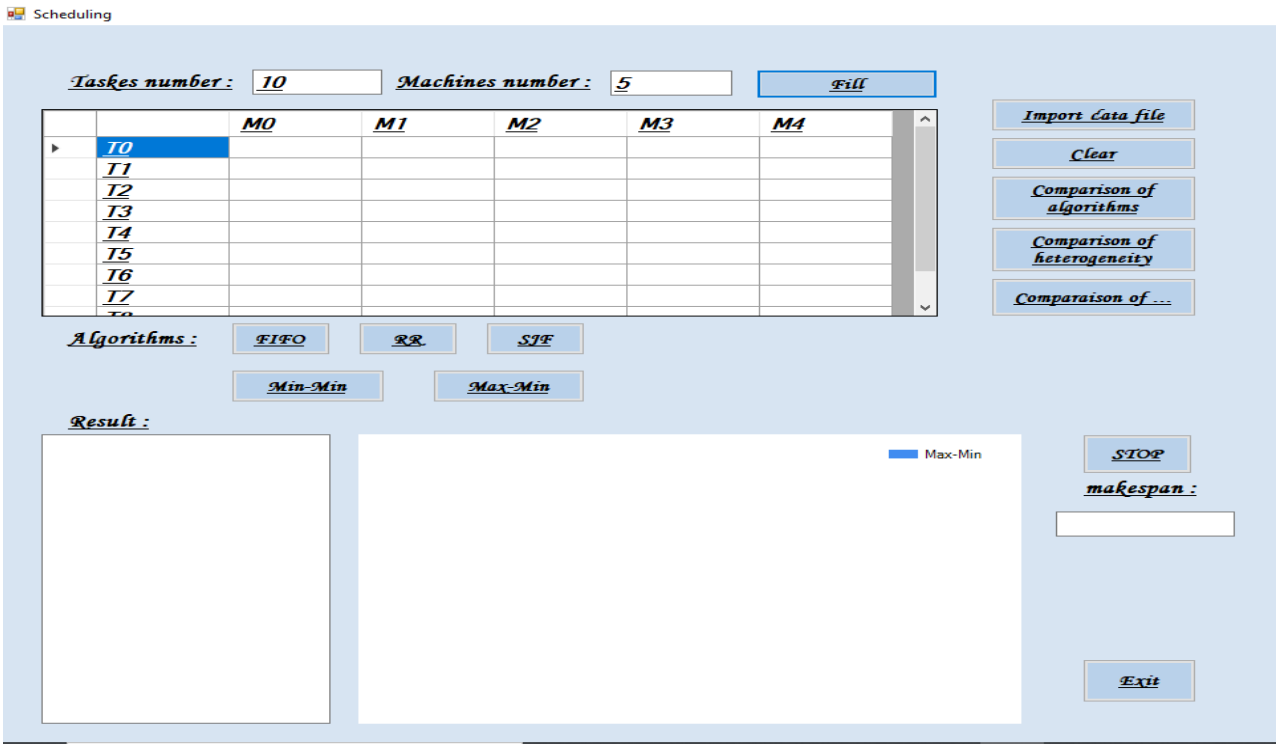

Figure 26: remplissage de DGV

L'interface ordonnancement des taches est composée d'un menu qui permet l'accès aux différents algorithmes à exécuter (6) qui peut être appliqué sur les données de l'ETC.

Apres l'exécution d'un algorithme, les résultats seront affiches. Nous avons choisi d'afficher la solution finale. A titre d'exemple, **la *figure 27* *et* *28*** montrent l'exécution pour l'algorithme FIFO et Max-Min

- **L'exécution de l'algorithme FIFO (First in First out):**

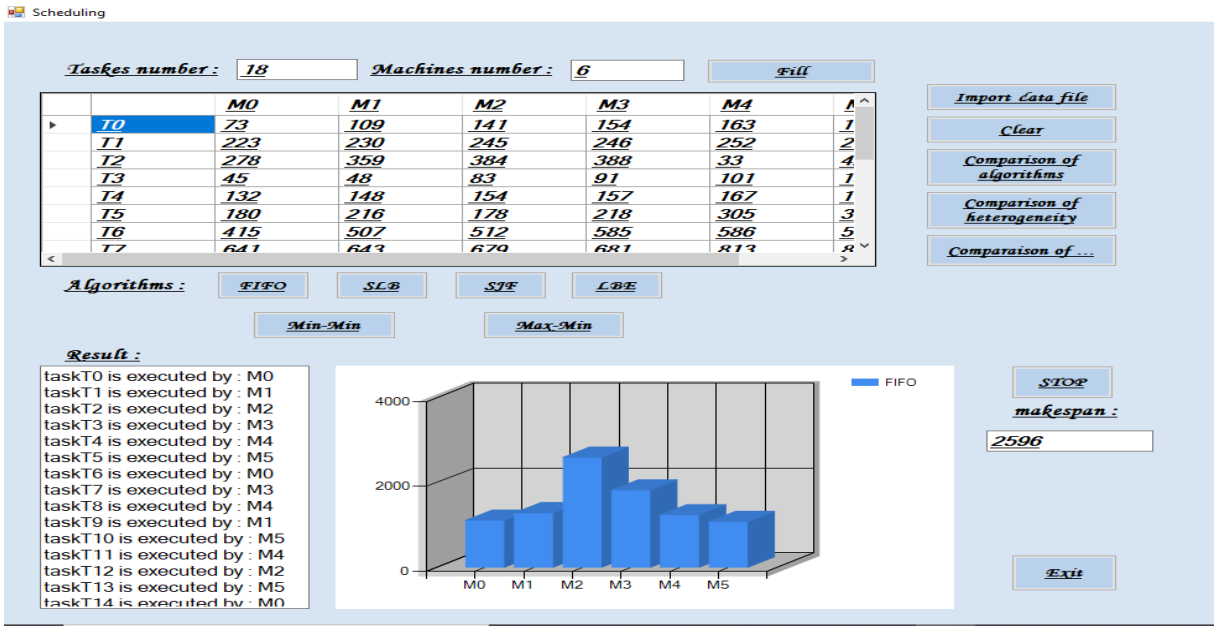

Figure 27: Exécution de FIFO.

- **L'exécution de l'algorithme Max-Min :**

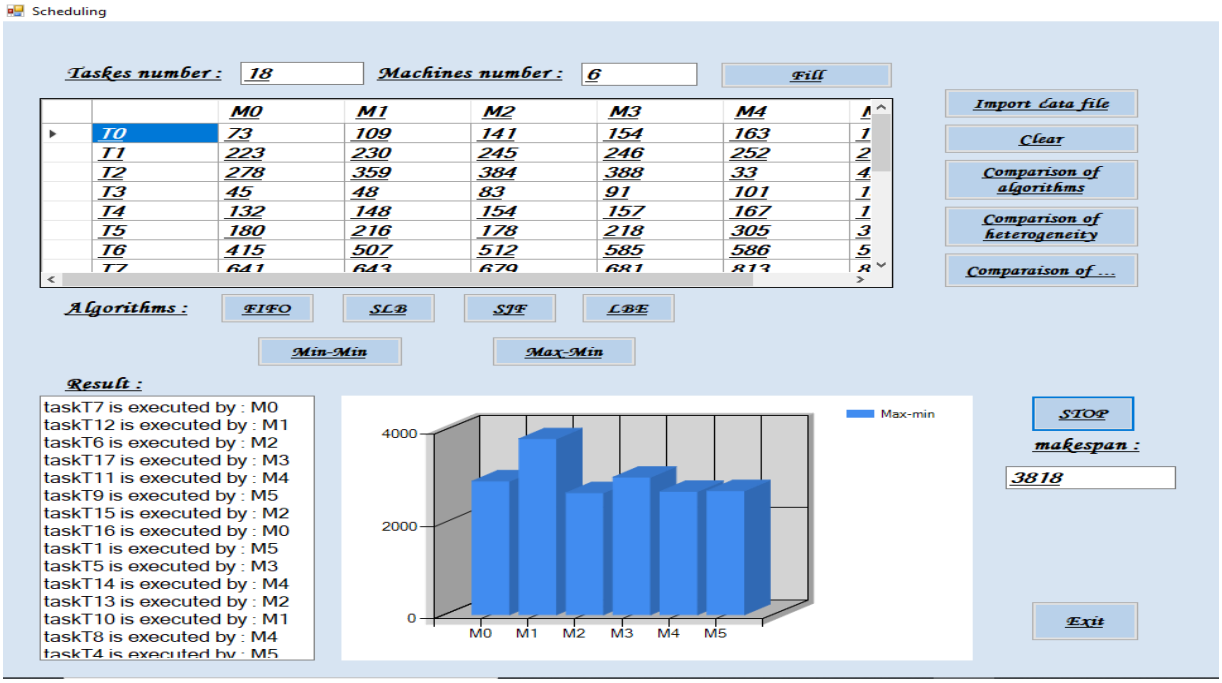

Figure 28: Exécution de Max-Min.

- **L'exécution de l'algorithme Min-Min :**

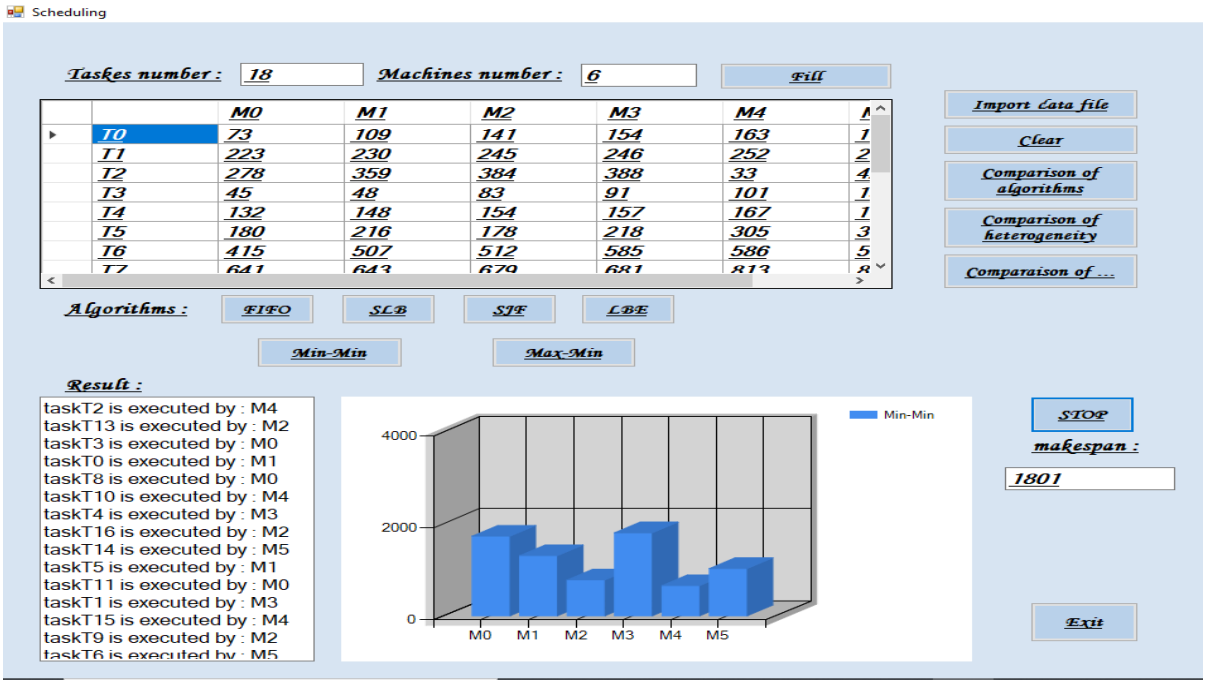

Figure 29: Exécution de Min-Min.

- **L'exécution de l'algorithme SLB et LBE :**

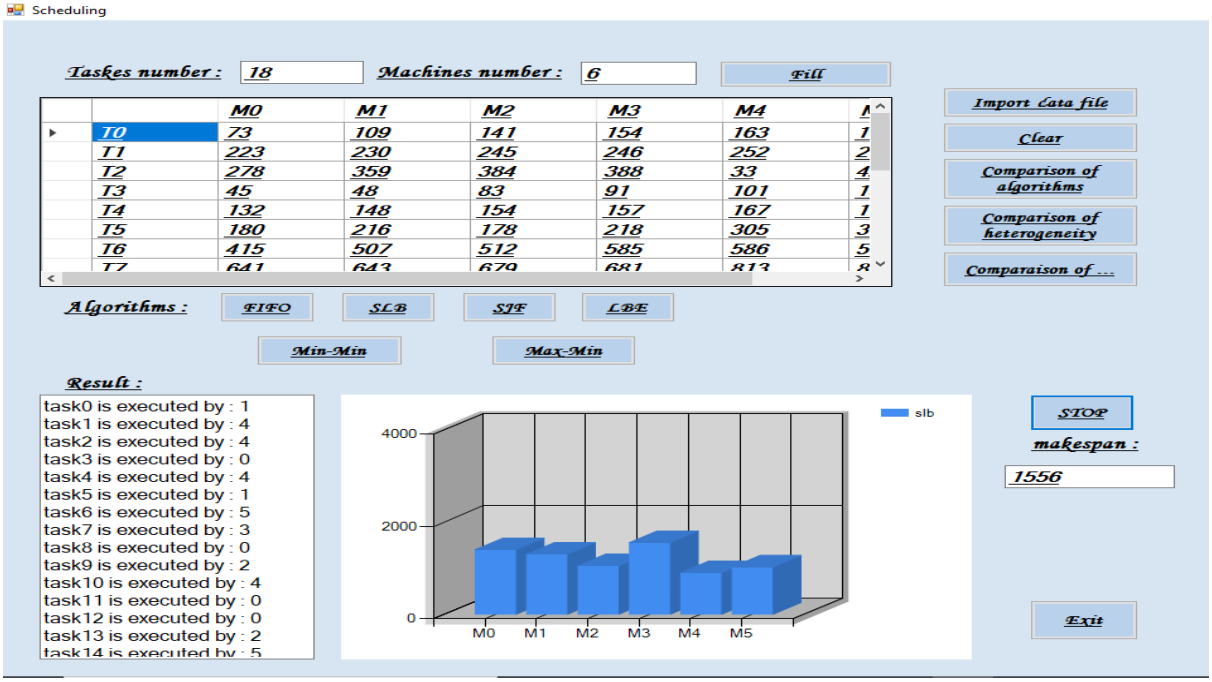

Figure 30: Exécution de SLB

Figure 31: Exécution de LBE.

- **Fenêtre des résultats (Comparisons of algorithms)**

Cette fenêtre permet d'afficher les résultats des algorithmes en affichant le makespan obtenu pour chaque algorithme et la réaffectation finale des tâches.

Cette fenêtre permet de comparer les algorithmes développés et estimer le meilleur algorithme qui en calculant le temps d'exécution optimal (makespan).

A titre d'exemple, ***la figure 32*** montre la comparaison entre les deux algorithmes pour Max-Min et Min-Min sachant que ce dernier est utilisé généralement pour initialiser l'algorithme SLB et LBE.

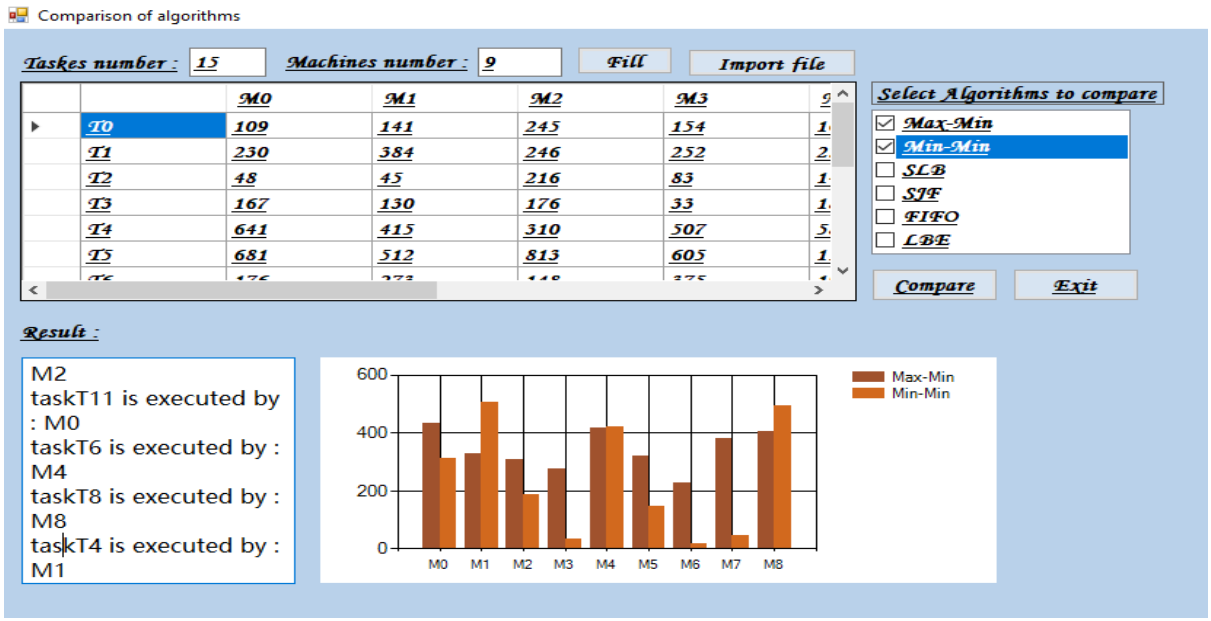

Figure 32: Comparaison entre Max-Min et Min-Min :

- **Comparaison entre Min-Min, SLB et LBE:**

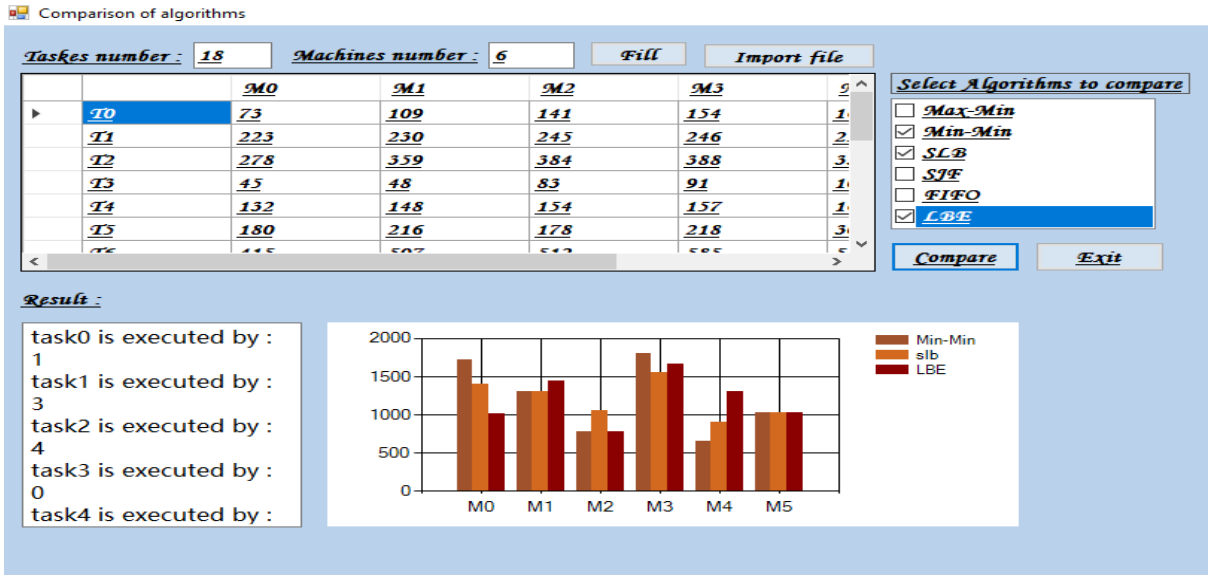

Figure 33: Comparaison entre Min-Min, SLB et LBE

7. Evaluation des algorithmes proposés

Afin de valider notre stratégie d'ordonnancement de tâches sur environnements de cloud computing, 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 proposés ont été appliqués à des matrices ETCs avec 12 différents types, jusqu'à 16 machines hétérogènes et jusqu'à 512 tâches hétérogènes **[23]**. Ces matrices ont une des propriétés suivantes :

Hétérogénéité des tâches : L'hétérogénéité des tâches est définie comme : lo : low et hi : high.

Hétérogénéité des machines : L'hétérogénéité machine est définie comme : lo et hi.

L'unité de temps pour tous ces matrices est second.

Les algorithmes sont exécutés sur 03 base de données (THMH / THML / TLML) ; en cliquant sur bouton **Exécute**

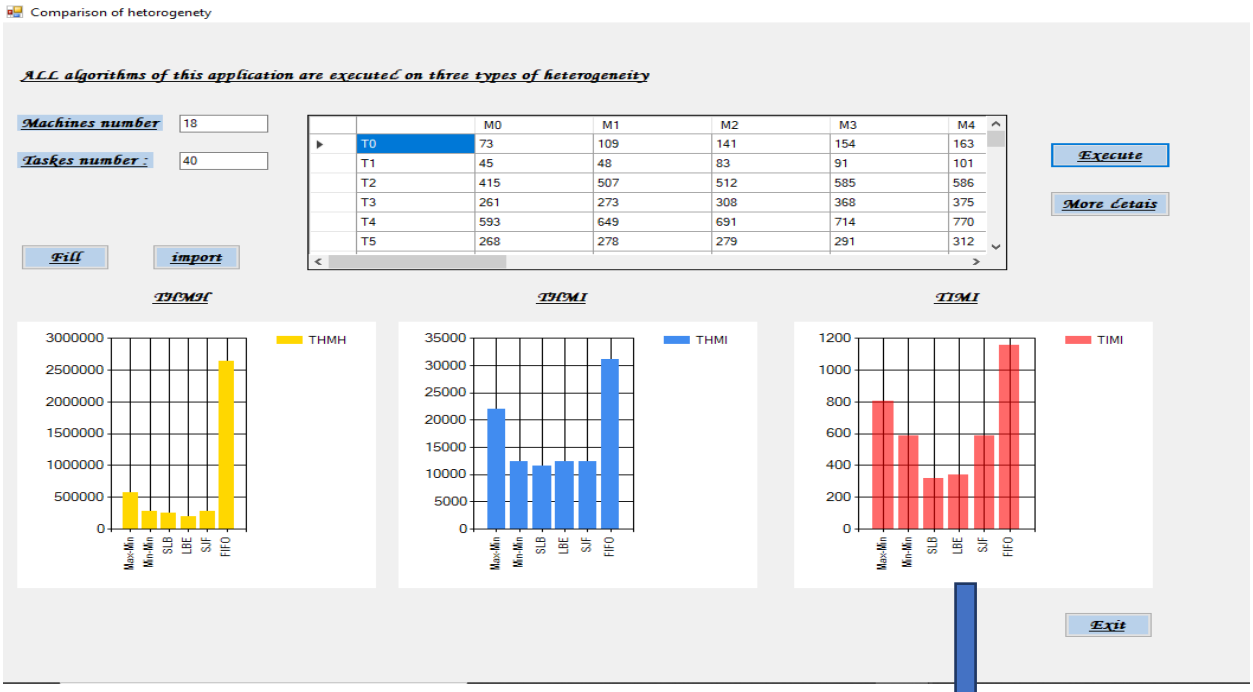

Figure 34: comparaison de l'hétérogénéité.

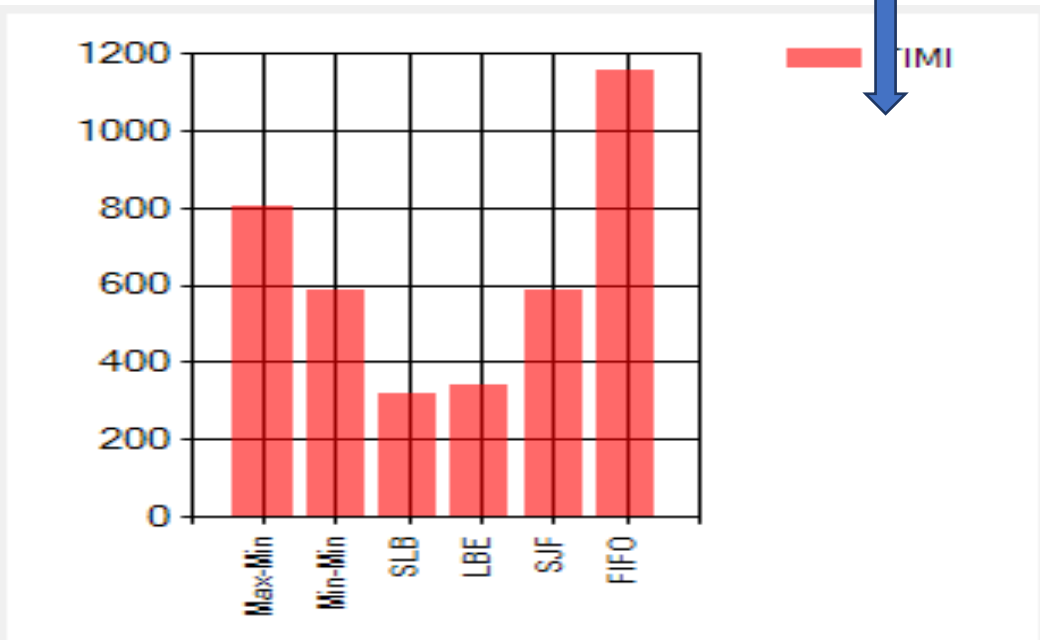

*Figure 35 : Le cas hétérogénéité faible de la tâche et de la machine faible (**lolo**).*

Botton **More details** pour afficher l'exécution et l'affectation des taches de chaque algorithme sur les 03 bases (THML, THMH, TLML), ***la figure 36*** montre un exemple d''exécution pour l'algorithme Max-Min.

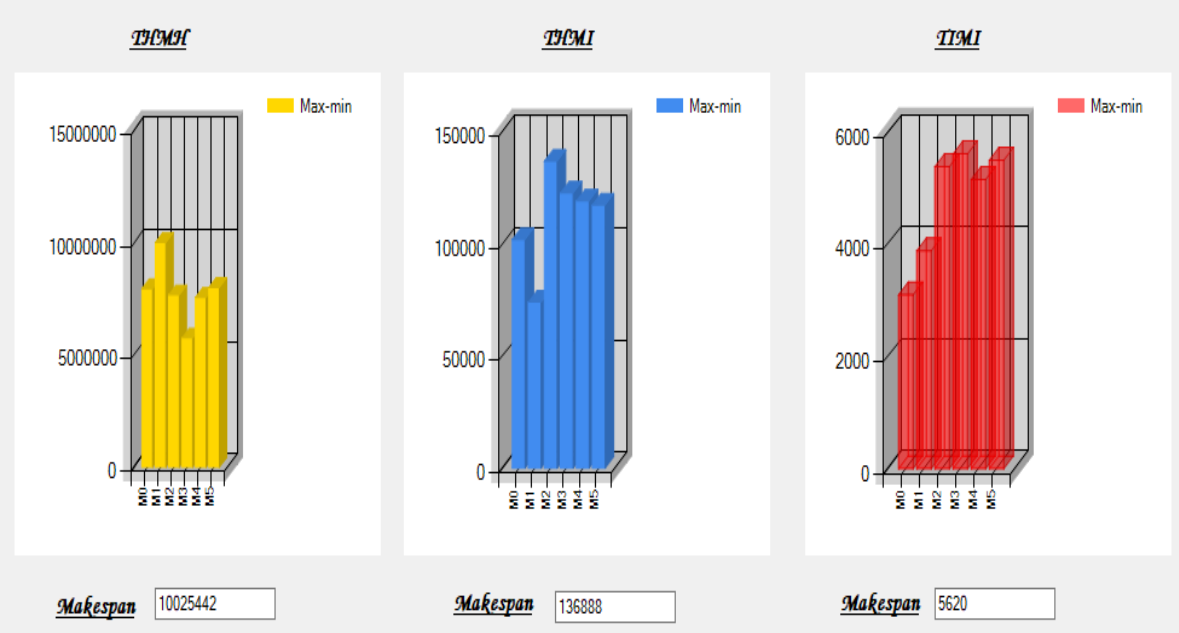

Figure 36: détaille de l'exécution de l'algorithme Max-Min

Les graphes suivants représentent des comparaisons expérimentales de deux cas :

- En cas de nombre des machines fixe (08 machines) : on remarque une élévation du makespan avec l'augmentations du nombre de taches.

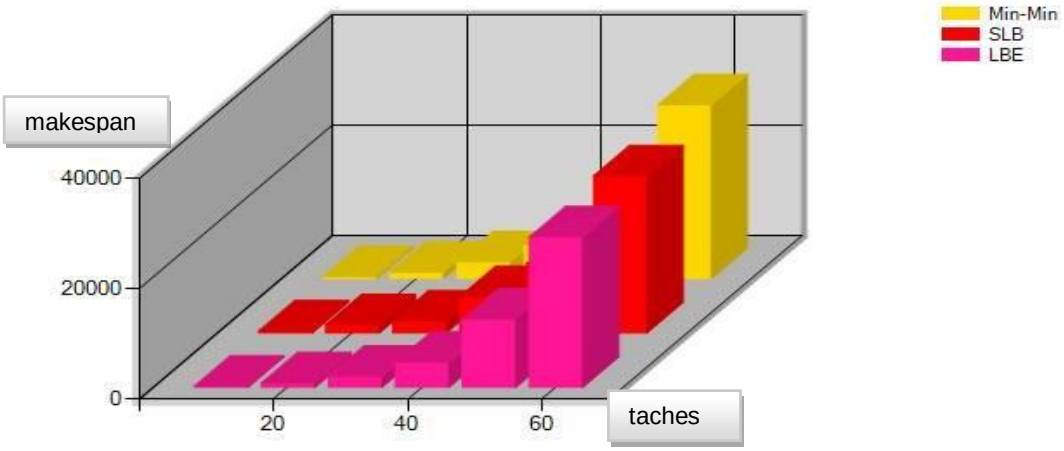

Figure 37: comparaison avec nombre des machines fixe.

- En cas de nombre des taches fixe (06 taches) : dans ce cas le makespan diminue avec l'augmentation du nombre de machines.

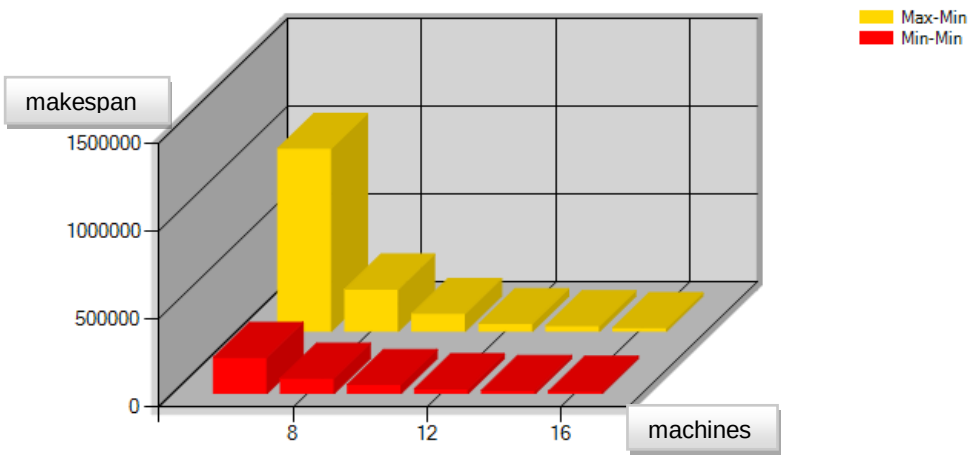

Figure 38: comparaison avec nombre des taches fixe

8. Conclusion

Au terme de notre étude, nous avons pu établir que l'algorithme Max-min et Min-min sont deux algorithme d'ordonnancement des taches et d'allocation des ressources pour améliorer le critère Makespan et que Min-Min comme algorithme d'ordonnancement des taches donne de meilleur résultat pour ce critère.

Après nos études en constate que SLB et LBE sont plus efficace que Min-min.

Conclusion générale et perspective

Comme nous l'avons noté tout au long de ce travail, le cloud computing a le potentiel d'être une force perturbatrice en affectant le déploiement et l'utilisation de la technologie. Le cloud pourrait être la prochaine évolution de l'histoire de l'informatique, en suivant les traces des mainframes, des mini-ordinateurs, des PC, des serveurs, des téléphones intelligents, etc., et en changeant radicalement la façon dont les entreprises gèrent l'informatique.

La théorie de l'ordonnancement est une branche de la recherche opérationnelle qui s'intéresse au calcul de dates d'exécution optimales de tâches. Pour cela, il est très souvent nécessaire d'affecter en même temps les ressources nécessaires à l'exécution de ces tâches. Un problème d'ordonnancement peut être considéré comme un sous-problème d'ordonnancement dans lequel il s'agit de décider de l'exécution opérationnelle des tâches planifiées.

Dans ce mémoire, nous avons essayé de proposer des algorithmes qui réalise le problème d'ordonnancement de tache et répondre à la problématique d'équilibrage de charge dans le cloud, afin de réduire le temps total d'exécution (makespan) des applications. Les tâches ont des durées d'exécution différentes, alors toute la difficulté est de réussir à attribuer chaque tâche à sa machine favorable, tout en conservant un bon équilibrage de la charge de différentes ressources.

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

De plus, et comme perspective de notre travail, il serait intéressant d'améliorer des novelles algorithmes basées sur les algorithmes SLB et LBE pour obtenir des résultats plus optimaux

Bibliographie

[01] M. Audin, Etat de l'art du Cloud Computing et adaptation au Logiciel Libre, livre blanc, 2009.

[02]" The NIST Definition of Cloud Computing" Recommendations of the National Institute of Standards and Technology, Peter Mell and Timothy Grance, September 2011.

[03] Jean-Paul Figer, L'informatique 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.

[04] R. Laurent, « Guide du Cloud », édition 2011.

[05] Le cloud computing une nouvelle filière fortement structurante, publication, directe ile de France, 2012.

[06] P. Warrior, "le Cloud computing" il tire sa puissance du réseau, livre blanc, 2010

[07] L. Baccouche. « Un Mécanisme d'Ordonnancement Distribué de Tâches Temps Réel. Thèse de doctorat, l'institut national polytechnique de Grenoble ». In: (1995) (cf. p. 14, 17).

[08] Richard WOLSKI Francine BERMAN. « Adaptive Computing on the Grid Using AppLeS. IEEE Transactions on Parallel and Distributed System, » in : (2002) (cf. p. 15).

[09] https://waytolearnx.com/2018/07/difference-entre-ordonnancement-preemptif-et-non-preemptif.html consulte le : 15-05-2020

[10] Sang-Min Park Jai-Hoon Kim Chameleon: « A Resource Scheduler in A Data Grid Environment* Graduate School of Information and Communication Ajou University, South Korea ». In: (2009) (cf. p. 16).

[11] Siegel H. J. Beck N. Boloni L. L. Maheswaran M. Reuther A. I. Robertson J.P. Theys M.D. Yao B. Hensgen D. Braun T. D. et R.F. Freund. « A comparison of eleven static heuristics for mapping a class of independent tasks on to heterogeneous distributed computing systems. Journal of Parallel and Distributed Computing, 61(6): 810 – 837. » In: (2001) (cf. p. 16).

[12] Gael Le Mahec. « Gestion des bases de données biologiques sur grilles de calcul ". Décembre ». In: (2008) (cf. p. 17).

[13] Huang Q.Y., Huang T.L., —An Optimistic Job Scheduling Strat egy based on QoS for Cloud Comput ing‖, IEEE Int ernat ional Conference on Intelligent Comput ing and Integrated Systems (ICISS).

[14] K. Etminani, and M. Naghibzadeh, "A Min-min Max-min Selective Algorithm for Grid Task Scheduling,"The Third IEEE/IFIP International Conference on Internet, Uzbekistan,

[15]. The cloud computing and distributed systems (cloud laboratory), university of merbourne (cloud sim) http:// www. Loud bus org /cloud sim /

[16]. Pooja Samal and Pranati Mishra, (2013), « Analysis ofvariants in Round Robin Algorithms for load balancing inCloud Computing", International Journal of Computer Science and Information Technologies, pp. 416-419, Vol. 4(3)

[17] T. Yamada, R. Nakano; *A genetic algorithm applicable to large-scale jobshob problems; BioSystems*; 1992.

[18].J. Levine, G. Ritchie; *A fast, effective local search for scheduling independent jobs in heterogeneous computing environments* ; Technical report ; University of Edinburgh.

[19] G.Fleury ; *Application de method stochastiques inspires du recuit simulé à des problems d'ordonnancement*; RAIRO A.P.I.I ; 1995 ;

[20] R. Braun, .A Comparison of Eleven Heuristics for Mapping a Class of IndependentTasks onto Heterogeneous Distributed Computing Systems. Internatio-nal Journal of Parallel and Distributed Computing, vol. 61, no. 6, pages810–837, December 2001

[21] K. Socha. ACO for Continuous and Mixed-Variable Optimization. In M. Dorigo and al.,editors, ANTS 2004, Springer LNCS 3172, pages 25–36, 2004

[22] MVIBUDULU, Initiation aux modèles, méthodes et pratique de la recherche opérationnelle ,2é Ed. Corrigé C.R.S.A.T., Kinshasa, 2007, p.169

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