REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
STERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE IBN KHALDOUN - TIARET

MEMOIRE

Présenté à :

FACULTÉ DES MATHEMATIQUES ET INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

MASTER

Spécialité :Génie Informatique

Par :

BAKHTI Chourouk

Sur le thème

Simulation l'équilibrage de charge pour l'ordonnancement des taches surplates-formes de calcul hétérogènes

---

Soutenu publiquement le ../07/2023 à Tiaret devant le jury composé de :

MrAID Lahcen	MCA	Université Ibn Khaldoun Tiaret	Président
Mr MEBAREK Bendaoud	Prof	Université Ibn Khaldoun Tiaret	Encadreur
Mr MEGHAZI Hadj Madani	MAA	Université Ibn Khaldoun Tiaret	Examinateur

2022-2023

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 sœur 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.

Résumé

Afin de répondre aux besoins de millions d'utilisateurs simultanés, il est essentiel d'assurer des performances élevées et une garantie de qualité de service. Pour cela, la mise en place d'algorithmes d'ordonnancement de tâches appropriés est nécessaire afin de traiter ces demandes de manière équitable et efficace. L'ordonnancement des tâches représente l'un des enjeux majeurs dans cet environnement, car il impacte directement les performances globales. Différents types d'algorithmes d'ordonnancement existent dans environnement de calculs hétérogénies avec plus de détail sur l'ordonnancement des taches dans les grilles calcul ,étudier et analyse quelque algorithmes l'ordonnancement, et proposer des modéliser le problème dans le cas d'équilibrage de de charge.

Dans cette étude, nous nous concentrons sur les performances des trois algorithmes d'ordonnancement de tâches les plus populaires : Min-Min, SLB et LBE.

Mot-clé:L'équilibrage de charge, tâches, Ordonnancement, SLB, LBE.

List des abréviations :

IaaS : Infrastructure as a Service

PaaS : Platform as a Service

SaaS: Software as a Service

CPU : Central Processing Unit

Makes pan : Le temps d'exécution

LBE : Loading Balancing Exchange

SLB : Simple Loading Balancing

ETC : Estimated Completion Time

UML : Unified Modeling Language

Liste des figures

Liste des tableaux

Table de matière

CHAPITRE I

L'ordonnancement des taches

Chapitre II

Les grilles calcul

Chapitre III

Parti 01 : L'équilibrage de charge

Parti 02 : Les Algorithme de l’équilibrage de charge

Chapitre IV

Implémentation, résultats & discussion

Introduction général

La planification des tâches est un aspect essentiel de l'optimisation des ressources et de l'efficacité du système. Il consiste à attribuer des tâches aux ressources disponibles, comme les processeurs ou les machines, pour assurer une exécution efficace. La planification des tâches maximise le débit du système, réduit le temps de réponse et équilibre la charge de travail entre les ressources.

Divers facteurs influent sur la planification des tâches, y compris les caractéristiques des tâches, les capacités en ressources et les contraintes du système. Les caractéristiques des tâches comprennent les exigences, les priorités et les dépendances en matière de calcul. Les ressources comprennent la puissance de traitement et la mémoire. Les contraintes du système peuvent comprendre des délais, des limites d'énergie et des exigences de qualité de service.

L'objectif de la planification des tâches est d'atteindre un rendement élevé du système et une utilisation des ressources. Les planificateurs utilisent des algorithmes et des politiques pour prendre des décisions intelligentes en matière d'affectation des tâches. Les algorithmes peuvent être statiques ou dynamiques, préventifs ou non, centralisés ou distribués.

L'ordonnancement statique attribue les tâches en fonction de connaissances préalables, tandis que l'ordonnancement dynamique prend des décisions en temps réel, tandis que les ordonnanceurs distribués distribuent les décisions.

L'objectif de notre travail est de modéliser et simuler l'ordonnancement des tâches dans un environnement de calcul hétérogène en utilisant les algorithmes d'équilibrage de charge. Il contribue à une gestion efficace de la charge de travail, améliorant le rendement et l'efficience du système.

Dans le premier chapitre : nous parlons des concepts liés à l'ordonnancement tâches et l'allocation des ressources

Dans le deuxième chapitre: Nous présentons d‘abord les definition de grille calculet les objectives, les avantages et les incovienents ensuite en a presenter quelque types .

Le troisième chapitre: L’équilibrage de charge dans l’environnement informatique en nuage a un impact important sur les performances du cloud.
Et dans le quetreiem chapitre: on a presentes les algorithem de L’équilibrage de charge (min-min ;slb ;lbe).

Chapitre I : L’ordonnancement des taches

1 -Introduction

L'ordonnancement est un champ d'investigation qui a connu un essor important ces dernières années, tant par les nombreux problèmes identifiés que par l'utilisations et développement de nombreuses techniques de résolutions.

Les problèmes d'ordonnancement se rencontrent souvent dans le milieu industriel, il s'agit de répartir un ensemble de travaux sur des machines ou ateliers de productions en respectant au mieux un ensemble de contraintes (technologiques, temporelles, etc.) et en cherchant à optimiser un ou plusieurs objectifs (cadence de productions, délais couts, etc.). En informatiques, on est également confronté aux problèmes d'ordonnancement pour allouer des processeurs à l'exécutions des programmes : génie civil (suivi de projets), administrations (gestion du personnel, emploi du temps) ou toute autre structure.

2-L'ordonnancement :

2-1définitions :

On peut considérer l'ordonnancement de taches comme la gestion et la prise en charge d'un ensemble de 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. [02] 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**[01]**.

3. Problème D'ORDONNANCEMENT AVEC RESSOURCES :

Dans cette leçon, nous retrouvons le problème d'ordonnancement déjà vu mais en ajoutant la prise en compte de contraintes portant sur les ressources. Après un exemple d'introduction, nous définirons le problème, et aborderons le cas où des tâches doivent être effectuées par un seul opérateur puis par deux opérateurs successifs. Après avoir souligné la difficulté de la grande majorité des problèmes d'ordonnancement, nous présenterons des méthodes de résolution approchées. Nous terminerons par la présentation du problème particulier du bin packing.

Les problèmes d'ordonnancement font partie de notre vie quotidienne ! Qui ne s'est jamais trouvé avec une liste de "choses à faire" et ne pas savoir par quel bout les prendre ?

Ces problèmes concernent bien évidemment aussi de nombreuses activités des entreprises. Le problème central de l'ordonnancement consiste à déterminer le calendrier d'exécution de tâches liées entre elles par des contraintes "temporelles", entre autres de succession, de manière à minimiser la durée totale.

Considérons par exemple la situation correspondant au graphe potentiel tâche suivant :

On peut sans difficulté déterminer le planning au plus tôt de ces tâches. Supposons maintenant que chaque tâche soit confiée à certaines personnes et que, par exemple, C et D soient réalisées par monsieur Dupont et que B et C fassent appel à l'équipe de Durand composée de 5 personnes, chacune de ces tâches mobilisant 3 de ces personnes.

On en déduit facilement que B et C ne peuvent plus être exécutées simultanément, pas davantage que C et D.

On a donc des couples de tâches en conflit. Pour chacun d'eux il faut déterminer l'ordre dans lequel on va faire les tâches correspondantes. Il est bien sûr hors de question d'examiner toutes les possibilités ; en effet avec n couples de tâches en conflit il y a 2n ordres possibles[22]4.

3.1-Définition d'un problème d'ordonnancement avec ressources :

Les problèmes d'ordonnancement sont très variés ; nous donnons ici les principaux éléments les définissant.

Un problème d'ordonnancement est caractérisé par un ensemble de tâches et un ensemble de ressources matérielles, humaines, financières … Les ressources utilisées par les tâches peuvent être de nature diverse. On peut distinguer entre elles **[05]**:

3.1.1-les ressources consommables : elles ne servent qu'une fois, par exemple les ressources financières…

3.1.2 -les ressources restituables : elles peuvent être réutilisées dès qu'elles sont libérées. C'est par exemple le cas des machines. On distinguera les ressources cumulatives (on dispose d'une certaine quantité de ressources) et les ressources disjonctives (on dispose d'une seule unité non fractionnable).

On se place dans le contexte où, pour chaque tâche, on connaît :
- sa durée - éventuellement la date à laquelle elle doit être achevée
- le cas échéant les contraintes entre les dates de début de ces tâches
- les ressources qu'elle utilise.

Le problème consiste à déterminer à quel moment, et avec quelles ressources, les tâches sont exécutées.

Différents critères peuvent être envisagés :
- le temps total d'exécution
-le respect de certains délais

3.2-Problème d'ordonnancement à un poste

On considère n tâches indépendantes, qui peuvent donc être exécutées dans un ordre quelconque. Dans le problème d'ordonnancement à un poste, chacune des tâches doit être réalisée par une seule machine ou poste de travail, qui ne peut faire qu'une tâche à la fois. On connaît le temps d'exécution pi de chaque tâche.

Par exemple, soient 5 taches dont les temps d'exécution sont donnés dans le tableau suivant :

Tâches i	1	2	3	4	5
Temps de traitement pi	4	3	7	2	2

***TABLEAU 01** : Il y a 5 ordres de passage possibles soit 120.*

Considérons les 2 ordres suivants : 1, 2, 3, 4, 5 et 4, 5, 2, 1, 3
Si les tâches sont exécutées sans interruption entre elles, la durée totale d'exécution est égale à 18 (4 + 3 + 7 + 2 + 2). Elle est évidemment indépendante de l'ordre Cependant, ces 2 ordonnancements ne sont pas équivalents.
Les graphiques suivant permettent de visualiser la pile des tâches en attente. Le deuxième ordonnancement conduit, a priori, à un temps d'attente total sur les 5 tâches plus petit que le premier.

4. **Les types d'ordonnancement :**

Différents types de planification sont basés sur des critères, tels que l'environnement statique ou dynamique, centralisé vs distribué, etc. bien qu'ils puissent se chevaucher et ne pas être clairement distincts**[09]** :

4.1-Planification statique : Pré-planification des tâches, toutes les informations sur les ressources et les tâches disponibles dans l'application doivent être connu et de plus une tâche est assignée une fois à une ressource, afin de faciliter l'adaptation en fonction du planificateur

4.2-Planification dynamique :Plus flexible que statique où les travaux sont disponibles dynamiquement pour planification au fil du temps par le planificateur sans problème, pour être en mesure de déterminer le temps d'exécution à l'avance. C'est hautement essentiel d'inclure l'équilibre de la charge comme facteur principal pour obtenir un algorithme de planification stable et efficace.

4.3-Planification centralisée : Comme mentionné dans la planification dynamique, un ordonnanceur centralisé / distribué est responsable de la prise de décision globale. En utilisant planification centralisée ; Facilité de mise en ouvre, efficacité et plus de contrôle et de surveillance des ressources sont des avantages acquis. D'autre part ; tel planificateur manque d'évolutivité, de tolérance aux pannes et d'efficacité performances, il n'est donc pas recommandé pour les applications à grande échelle.

4.4-Planification décentralisée/distribuée : Ce type de la planification est plus réaliste pour les réseaux réels malgré son faible efficacité par rapport à la planification centralisée, danslaquelle les planificateurs locaux demandent à gérer et maintenir l'état de la file d'attente des travailleurs car il n'y a plus d'entité centrale de contrôle.

4.5-Planification coopérative: Dans ce cas, le système a déjà beaucoup d'ordonnanceurs, chacun est responsable d'effectuer certaines activités dans le processus de planification vers système commun large échelle basée sur la coopération de procédures, règles, données et utilisateurs actuels du système .

4.6-Planification préventive : Ce genre de planification permet à chaque tâche à interrompre pendant l'exécution et une tâche peut être migrée vers une autre ressource en laissant son originale ressource allouée inutilisée pour être disponible pour d'autres tâche. Elle est plus utile s'il y a des contraintes comme priorité d'être considéré.

4.7-Planification non préventive : Dans laquelle les ressources ne sont pas autorisées à être réaffecté jusqu'à ce que le travail planifié termine son exécution.

4.8-Mode immédiat / en ligne : dans lequel le planificateur planifie toute tâche récemment arrivé dès qu'elle arrive sans en attente du prochain intervalle de temps sur les ressources disponibles à ce moment.

4.9-Mode batch / hors ligne : le planificateur conserve les travaux entrants comme groupe de problèmes à résoudre dans le temps intervalles, de sorte qu'il est préférable de catégoriser une tâche pour des ressources en fonction de ses caractéristiques.

5.Le role d'ordonnancement :

L'ordonnancement prévoit la mise en place d'un plan de travail détaillé, conçu pour coordonner l'activité industrielle d'une entreprise en fonction des ressources et conditions nécessaires à chacune des étapes de production d'un produit. Aussi appelé « scheduling », l'ordonnancement permet de visualiser avec précision les tâches à réaliser, leur durée d'exécution, les moyens nécessaires à chaque étape de fabrication et les difficultés auxquelles l'entreprise peut être confrontée dans sa production**[02]**.

De par son rôle, l'ordonnancement est utile aux différents services de l'entreprise. Par exemple, il permet aux responsables des achats de planifier la réception des produits, aux ressources humaines de mieux répartir les effectifs et aux commerciaux de définir les délais de livraison proposés aux clients. Sa planification et son exécution sont encadrées par un ordonnanceur dont la fonction consiste à garantir la bonne circulation des flux d'informations et le respect de la planification**[06]**.

6.Différences clés entre l'ordonnancement préemptif et non préemptif :

6.1-Dans l'**ordonnancement préemptif**, le CPU est allouée aux processus pour une durée limitée. Dans l'**ordonnancement non préemptif**, le CPU est allouée au processus jusqu'à ce qu'elle se termine ou passe en état d'attente. [09].

- Le processus d'exécution dans l'**ordonnancement préemptif** est interrompu au milieu de l'exécution, tandis que le processus d'exécution dans

l'**ordonnancement non préemptif** n'est pas interrompu au milieu de l'exécution

- L'**ordonnancement préemptif** a pour objectif de faire passer le processus de l'état prêt à l'état en cours et de maintenir la file d'attente prête. Tandis que, l'**ordonnancement non préemptif** n'a pas besoin de basculer le processus de l'état en cours à l'état prêt.

6.2-Dans l'**ordonnancement préemptif**, si un processus avec une haute priorité arrive fréquemment dans la file d'attente prête alors le processus avec une faible priorité doit attendre longtemps, et il peut mourir de faim. Alors, dans l'**ordonnancement non préemptif**, si le CPU est allouée au processus avec un temps de rafale plus important, alors les processus avec un petit temps de rafale peuvent avoir à mourir de faim.

- L'**ordonnancement préemptif** est assez flexible car les processus critiques sont autorisés à accéder au processeur dès qu'ils arrivent dans la file d'attente prête, quel que soit le processus en cours d'exécution. L'**ordonnancement non préemptif** est rigide car même si un processus critique entre dans la file d'attente prête, le processeur qui exécute le processus n'est pas perturbé**[10]**.
- L'**ordonnancement préemptif** est associer aux coûts car elle doit maintenir l'intégrité des données partagées, ce qui n'est pas le cas avec l'**or*donnancement non préemptif***

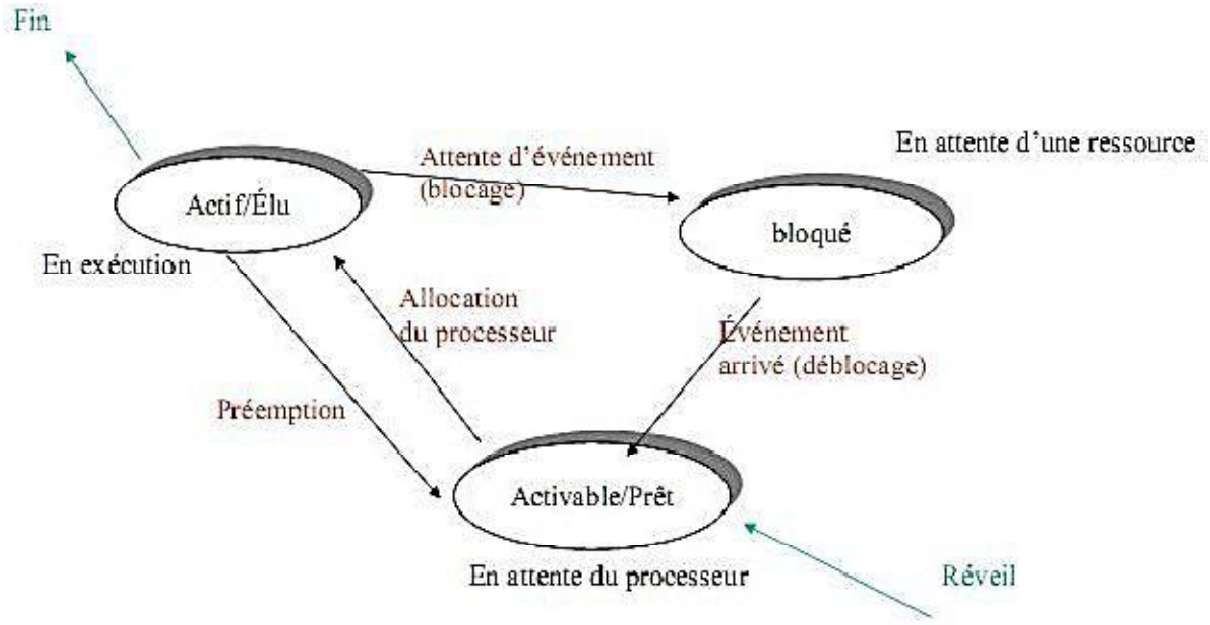

Figure 1 : Comportement d'une tâche

7.les différents algorithmes d'ordonnancement :

Algorithmes d'ordonnancement[4]

- Round-robin (ou méthode du tourniquet)
- Rate-monotonicscheduling (RMS)
- Earliest deadline first scheduling (EDF)
- FIFO.
- Shortest job first (SJF, ou SJN -Shortest Job Next-)
- CompletelyFairScheduler (CFS)
- LIFO**[04]**
-

8. Les éléments d'un problème d'ordonnancement :

Le problème d'ordonnancement est constitué principalement de quatre éléments suivants :

8.1 Les tâches:

Une tâche est une entité élémentaire localisée dans le temps, par une date de début et/ou de fin, et dont la réalisation nécessite une durée préalablement définie**[07]**.

Elle est constituée d'un ensemble d'opérations qui requiert, pour son exécution, certaines ressources et qu'il est nécessaire de programmer de façon à optimiser un certain objectif. On distingue deux types de tâches :

- **les tâches morcelables (préemptibles) :**qui peuvent être exécutées en plusieurs fois, facilitant ainsi la résolution de certains problèmes.
- **les tâches non morcelables (indivisibles) :**qui doivent être exécutées en une seule fois et ne sont interrompues qu'une fois terminées.

8.2 Les ressources :

Une ressource est un moyen technique ou humain utilisé pour réaliser une tâche. On trouve plusieurs types de ressources:

- **Les ressources renouvelables :** qui après avoir été allouées à une tâche, redeviennent disponibles (machines, personnel, etc.).
- **Les ressources consommables :** qui après avoir été allouées à une tâche, ne sont plus disponibles (argent, matières premières, etc.). Qu'elle soit renouvelable ou consommable, la disponibilité d'une ressource peut varier au cours du temps. Par ailleurs, dans le cas des ressources renouvelables, on distingue principalement, les ressources disjonctives qui ne peuvent exécuter

qu'une tâche à la fois et les ressources cumulatives qui peuvent être utilisées par plusieurs tâches simultanément mais en nombre limité.

8.3 Les contraintes :

Les contraintes expriment les restrictions que peuvent prendre conjointement les variables de décision. Donc les contraintes nous renseignent sur les limites imposées par l'environnement. On distingue plusieurs types de contraintes, qui sont:

- **Les contraintes temporelles**

: Pour la réussite d'un projet ou d'un plan de production, on doit se soumettre aux impératifs de production (réalisation) traduits par le respect d'un planning fixé avant. Au niveau global on parlera d'une date de lancement d'une tâche et d'une date de livraison. A un autre niveau de détail plus fin, pour une tâche ou une opération. Ce dernier niveau se traduit par la définition des dates suivante : (i) date de disponibilité : ri ; (ii) date d'échéance : di**[08]**.

- **Les contraintes de précédence :**

Une contrainte qui lie le début d'une activité à la fin d'un autre est appelée contrainte de succession ou de précédence. Les gammes opératoires sont un exemple de ces contraintes, d'autres contraintes sont imposées dans certain cas telles que les contraintes de synchronisation, de simultanéité, ou de recouvrements etc.

- **Les contraintes de ressources :**

Les contraintes de ressources représentent le fait que les activités requièrent tout au long de leur exécution une certaine quantité de ressources**[13]**.

Les contraintes de ressources concernent les deux points suivants: (i) l'utilisation de ces ressources, (leur nature, la quantité nécessaire, et les caractéristiques d'utilisation) ; (ii) la disponibilité et la quantité de ces ressources**[12]**.

8.4 Les critères :

Un critère correspond à des exigences qualitatives et quantitatives à satisfaire permettant d'évaluer la qualité de l'ordonnancement établi**[08]**.

Les critères dépendant d'une application donnée sont très nombreux; plusieurs critères peuvent être retenus pour une même application. Le choix de la solution la plus satisfaisante dépend du ou des critères préalablement définis, pouvant être classés suivant deux types, réguliers et irréguliers.

Différents critères ne sont pas indépendants; certains même sont équivalents. Deux critères sont équivalents si une solution optimale pour l'un est aussi optimale pour l'autre et inversement:

- **Les critères réguliers :**

Constituent des fonctions décroissantes des dates d'achèvement des opérations. Quelquesexemplessontcités ci-dessous:

-la minimisation des dates d'achèvement des actions.

-la minimisation du maximum des dates d'achèvement des actions.

-la minimisation de la moyenne des dates d'achèvement des actions. -la minimisation des retards sur les dates d'achèvement des actions.

-la minimisation du maximum des retards sur les dates d'achèvement des actions.

- **Les critères irréguliers :**

sont des critères non réguliers, c'est-à-dire qui ne sont pas des fonctions monotones des dates de fin d'exécution des opérations, tels que:

-la minimisation des encours.

-la minimisation du coût de stockage des matières premières.

-l'équilibrage des charges des machines.

-l'optimisation des changements d'outils**[11]**.

9. Conclusion

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

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

La difficulté de ces problèmes provient, à la fois du caractère hétérogène de l'environnement, dont l'objectif est de minimiser le makes-pan. Dans le chapitre suivant, nous nous intéressons à la description et l'implémentation de deux algorithmes Single LoadBalancing (SLB) et Min-min et la comparaison de ses résultats avec celles obtenus par quelques approches existantes dans la littérature.

Chapitre II :
Les grilles de calcul

1. Introduction :

Une grille de calcul est un outil qui permet de réaliser des opérations arithmétiques en utilisant des cases et des lignes

Par exemple, pour additionner 23 et 15, on peut utiliser une grille de calcul comme celle-ci :

On ajoute les chiffres de chaque colonne en commençant par la droite, et on écrit le résultat en dessous. Si le résultat dépasse 9, on reporte la retenue sur la colonne suivante.

La grille de calcul peut aussi servir à soustraire, multiplier ou diviser des nombres. Elle permet de visualiser les étapes du calcul et de vérifier son exactitude.[34]

Les grilles de calcul se sont développées au début des années 2000, dès que l'évolution technologique et des tarifs ont rendu abordables les réseaux informatiques académiques à très haute vitesse, supérieure à 1 gigaoctet par seconde. En effet, une telle vitesse permet d'envoyer des données distantes dans un processeur à un rythme voisin de celui de traitement interne des données. Cela permet donc de traiter des données stockées loin avec une efficacité comparable à un traitement local. Ce découplage entre localisation des ressources de calcul et de stockage est une caractéristique essentielle des grilles de calcul

Le terme anglais grille s'inspire de la grille d'électricité. Initialement, le concept de grille partait du principe d'un tel système : les ressources d'un ordinateur (processeur, espace disque, mémoire) ont été misse à la disposition d'un utilisateur aussi facilement que le branchement d'un appareil électrique sur une prise électrique [20].

2. **Définition :**

Ian Foster et Carl Kesselman ont été les premiers a proposé une définition de la grille de calcul :

« …une infrastructure matérielle et logicielle fournissant un accès fiable (dependable), cohérent (consistent), à taux de pénétration élevé (pervasive) et bon marché (in ex pensive) à des capacités de traitement et de calcul ».

Le terme infrastructure matérielle et logicielle

désigne un ensemble de ressources de calcul et de stockage, autonomes et hétérogènes interconnectées par un réseau de communication hétérogène et gérés au moyen d'une couche logicielle (middleware).

La fiabilité du système de grille est définie comme la probabilité pour tous les programmes de calcul en grille
Sont exécutés avec succès dans le système informatique en grille. Le besoin d'un accès fiable est fondamental. Les utilisateurs exigent des assurances qu'ils recevront des performances fiables et souvent de haut- niveaux à partir des composants constituant la grille.

Le besoin de cohérence (consistency) du service est un second intérêt fondamental. Une grille doit être construite avec des services standard, des protocoles, des interfaces, et opérants au sein de paramètres standards. Un important défi lors du développement des standards est d'encapsuler l'hétérogénéité sans compromettre la haute performance d'exécution.

L'accès à taux de pénétration élevé permettrait la disponibilité des ressources en s'adaptant à un environnement dynamique dans lequel la défaillance des ressources est courante ; cela n'implique pas que les ressources sont partout ou universellement accessibles.

Finalement l'infrastructure doit offrir un accès relativement bon marché au regard des bénéfices qu'elle peut apporter. L'aspect bon marché est très important d'un point de vue de la viabilité économique.

- **La grille coordonne des ressources qui ne sont pas sous contrôle centralisé** : une grille intègre et coordonne des ressources et utilisateurs qui se trouvent au sein de différents domaines de contrôle, et aborde les questions de sécurité, politique, paiement, appartenance (adhésion), etc., qui se posent dans ce cadre. Sinon, nous avons affaire à un système de gestion local.
- **La grille utilise des protocoles et interfaces standards, ouverts et universels** : la grille est conçue à partir de protocoles et interfaces universels qui adressent des questions fondamentales telle que l'authentification, l'autorisation, la découverte des ressources et l'accès aux ressources. Il est important que ces protocoles et interfaces soient standardisés et ouverts. Sinon, il s'agit d'un système à application spécifique.
- **La grille a pour but de délivrer des qualités de service non triviales:**

la grille autorise ses ressources constitutives à être utilisées d'une manière coordonnée afin de délivrer différentes qualités de service, concernant par exemple le temps de réponse, le débit, la disponibilité, et la sécurité, et/ou la co-allocation de multiples ressources pour satisfaire les demandes complexes de l'utilisateur, afin que l'utilité du système combiné est significativement plus grande que celle de la somme de ses parties.

3. Objective de grilles des calculs :

L'objective de la grille de calcul est de partager et d'utiliser des ressources informatiques (temps de calcul, espace de stockage, etc.) distribuées sur plusieurs sites, afin de réaliser des tâches complexes ou volumineuses qui nécessitent une grande puissance de calcul. La grille de calcul permet ainsi de créer un ordinateur virtuel très puissant à partir de plusieurs ordinateurs réels[34].

La grille de calcul vise à atteindre les objectifs suivants :
Supercalculateurs distribués Grilles de calcul Grilles de données Grilles de services Type de grilles Accélérateurs de calcul Grille à la demande Grille de collaboration **[23]**

- **Le partage de ressources de calcul distribuées et hétérogènes appartenant à différentes organisations** : la grille de calcul est le partage, la sélection et l'agrégation d'un groupe de ressources telles que les superordinateurs, les ordinateurs centraux, les systèmes de stockage, les sources de données et les systèmes de gestion qui se comportent comme des réseaux de calcul. Elle favorise le partage des ressources distribuées pouvant être de nature hétérogènes.
- **L'exploitation des ressources sous-utilisées** : dans la plupart des organisations et des entreprises, les ressources de calcul sous-utilisées sont très nombreuses. La plupart de ces ressources ne sont occupées que moins de 5% du temps. Donc, ces ressources sont relativement inactives. La grille de calcul est conçue pour exploiter ces ressources sous-utilisées et augmenter l'efficacité de l'utilisation des ressources. Les utilisateurs peuvent également louer les ressources qui se trouvent sur la grille pour exécuter leurs applications de calcul intensif, au lieu d'acheter leurs propres ressources (coûteuses) dédiées.
- **L'activation et la simplification de la collaboration entre différentesorganisations** : une autre capacité de grille de calcul est la création d'un environnement propice à la collaboration entre organisations. permet aux systèmes très hétérogènes et distribués de fonctionner ensemble, ce qui simplifie ainsi la collaboration entre les différentes organisations en fournissant un accès direct aux ordinateurs, logiciel et données.
- La fourniture d'un service de connexion unique avec un accès sécurisé aux ressources du réseau tout en protégeant la sécurité des utilisateurs et des sites distants : les grilles fournissent un service de connexion unique à tous les

utilisateurs sur toutes les ressources distribuées en utilisant des mécanismes d'authentification de grille. Elles fournissent également un accès sécurisé à toute information n'importe où sur n'importe quel type de réseau. Ceci est réalisé en fournissant mécanismes de contrôle d'accès qui régissent ces ressources.

- **La fourniture de gestion des ressources, des services d'information, surveillance et transport sécurisé des données :** le partage des ressources et des réseaux impliqués dans la grille de calcul sont difficiles à gérer et surveiller, mais l grille de calcul est en mesure de relever ces défis en raison de son architecture et de ses protocoles.
- **Une solution aux problèmes à grande échelle** : les grilles sont conçues pour exploiter les ressources sous-utilisées, ce qui signifie qu'elles peuvent en employer un grand nombre pour résoudre un problème à grande échelle. C'est une solution prometteuse pour des problèmes tels que le stockage et le traitement de grandes quantités de données que même les ordinateurs centraux ne peuvent pas traiter, comme les prévisions météorologiques. Ces ressources peuvent être des dispositifs de grande capacité tels que le stockage de disque de grande capacité et le calcul de haute performance.
- **La fourniture de résultats rapides et une livraison plus efficace :** la grille de calcul permet le traitement en parallèle. Ce traitement peut également avoir des périphériques à haute capacité. Avec les grilles de calcul, les entreprises peuvent utiliser efficacement les ressources de calcul et de données et les combiner pour des charges de travail de grande capacité.

4. Les avantages et Les inconvénients :

Une grille de calcul présente des avantages et des inconvénients selon le contexte et les besoins des utilisateurs. Voici quelques exemples :

- **4.1 Les avantages d'une grille de calcul sont :**

o Elle permet de bénéficier d'une grande puissance de calcul à partir de ressources distribuées, ce qui peut être utile pour des tâches complexes ou volumineuses qui nécessitent beaucoup de temps ou de mémoire.

o Elle permet de partager et d'utiliser des ressources disponibles sur différents sites, ce qui peut favoriser la collaboration et l'innovation entre des équipes ou des organisations.

o Elle permet de réduire les coûts liés à l'achat ou à la maintenance de matériel informatique, en utilisant des ressources existantes ou inutilisées.

- **4.2 Les inconvénients d'une grille de calcul sont :**

- Elle nécessite un logiciel spécifique pour gérer les requêtes, la répartition et la synchronisation des travaux sur les différents nœuds, ce qui peut être complexe à concevoir, à installer et à maintenir.
- Elle dépend de la qualité et de la fiabilité du réseau informatique qui relie les nœuds, ce qui peut entraîner des problèmes de performance, de sécurité ou de disponibilité en cas de panne ou d'attaque.
- Elle pose des défis en termes d'interopérabilité, de standardisation et de réglementation entre les différentes grilles ou les différentes sources de données, ce qui peut limiter la compatibilité ou la confidentialité des informations.

5. Les types des grilles de calcul :

On distingue différents types de grilles suivant le type de matériel disponible à chaque nœud : les grilles de supercalculateurs, qui relient des supercalculateurs ; les grilles de production ou de recherche, ensembles administrés de centaines ou de milliers d'ordinateurs individuels ; et les grilles pair-à-pair (utilisées par exemple pour le téléchargement de fichiers musicaux sur Internet), qui relient des ordinateurs individuels de particuliers

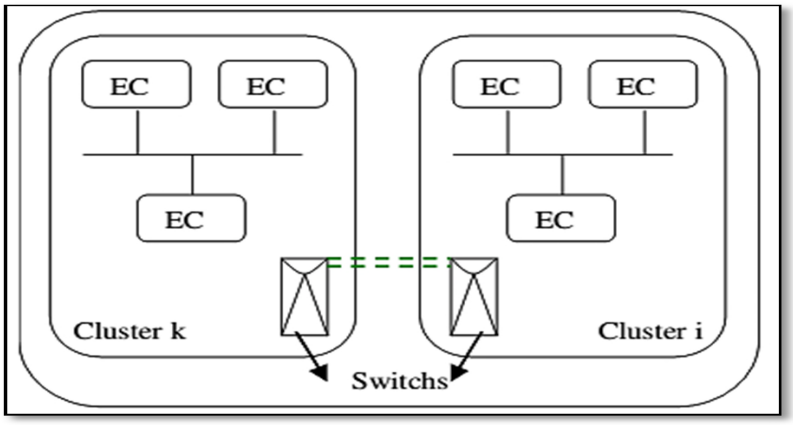

Figure 2. Types de Grilles

Une grille de calcul est une infrastructure virtuelle constituée d'un ensemble de ressources de calcul potentiellement partagées, hétérogènes, distribuées, autonomes et délocalisées. Une grille est en effet une infrastructure, c'est-à-dire, des équipements techniques d'ordre matériel et logiciel. Cette infrastructure est qualifiée de virtuelle car relations entre les entités qui la composent n'existent pas sur le plan matériel, mais d'un point de vue logique **[21]**.

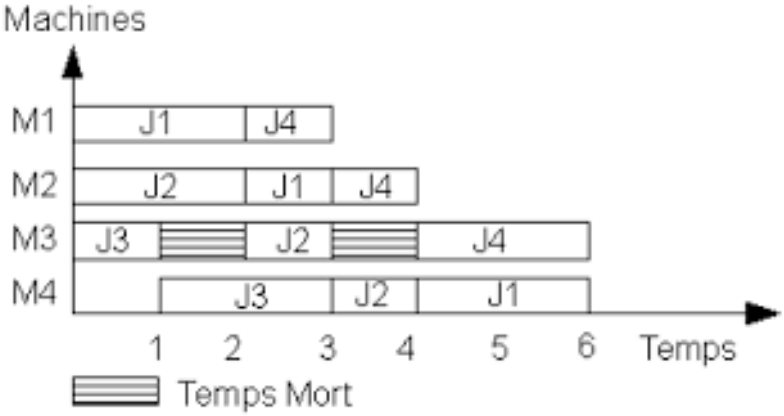

Figure 3.l'ordonnancement de Grilles calcul

6. Taxonomie des grilles

Il existe différents types de grilles qui correspondent aux différents types de problèmes à résoudre. Quelques grilles sont conçues pour tirer avantage de la puissance de calcul des ressources, alors que d'autres sont conçues pour exploiter la capacité de stockage de ces ressources. Le type de grille est donc, sélectionné suivant le type d'application à traiter. Nous pouvons citer les trois principaux types de grilles :

7. Grilles de calcul

Une grille de calcul se concentre sur la mise en réserve de ressources spécifiquement pour la puissance de calcul. Dans ce type de réseau, la plupart des machines sont des serveurs Hautes performances et sont dédiées aux calculs intensifs. Ce type de grille peut être à son tour subdivisée en deux catégories :

- **Supercalculateurs distribués** : exécutent une application parallèle sur plusieurs machines dans le but d'améliorer son temps de réponse ;

- **Accélérateurs de calcul :** exécutent une application sur une machine appropriée dans l'objectif d'améliorer le temps de réponse moyen d'un flot d'applications.

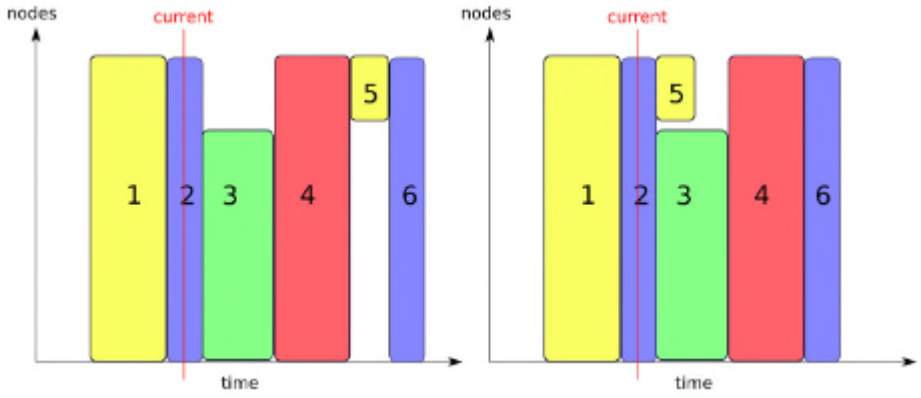

Figure 4. Ordonnancement statique sur la grille.

7.1. Grilles de données

Une grille de données est responsable de l'hébergement et de l'accès aux données de plusieurs organisations. Les utilisateurs ne sont pas concernés par l'emplacement de ces données tant qu'ils ont accès aux données. Une grille de données leur permettrait de partager leurs données, de gérer les données et de gérer des problèmes de sécurité tels que l'accès à quelles données. **[21]**

Ce type de Grille convient aux applications nécessitant une importante capacité de stockage. Elles assurent un accès sécurisé aux données distribuées. Les grilles de données incluent le concept de bases de données fédérées dans lequel un groupe disponible de ces bases fonctionne comme une seule.

7.2. Grilles de services

Les grilles de services sont utilisées pour les systèmes qui offrent des services ne peuvent pas être disponibles sur une simple machine. Elles peuvent être subdivisées en deux catégories :

- **Grille à la demande :** elle agrège dynamiquement différentes ressources pour fournir des nouveaux services. A titre d'exemple, nous pouvons citer le problème de simulation qui nécessite des ressources (en nombre et en type), qui dépendent des paramètres d4exécution. Autre exemple de grille à la demande est la grille Ces dernières années, les grilles ont suscité un intérêt particulier. Ce concept consiste à déporter sur des serveurs distants, des traitements

informatiques traditionnellement localisés sur le poste client de l'utilisateur ou les serveurs locaux.

- **Grille de collaboration :** elle regroupe plusieurs utilisateurs et applications en groupes de travail collaboratif. Ce type de grille permet une interaction entre les utilisateurs et les applications en temps réel au moyen d'un espace de travail virtuel.

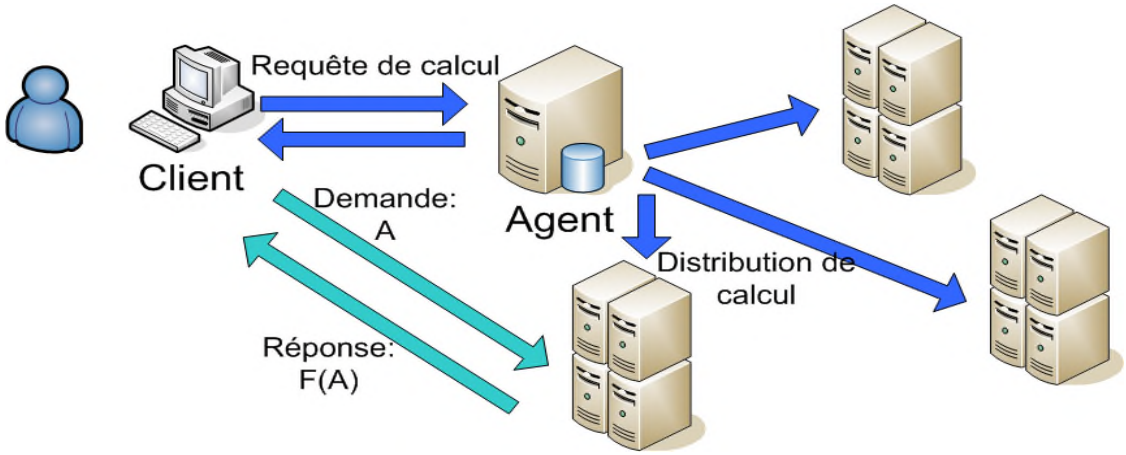

Figure 5. Types de Grilles

8. Différentes topologies de Grilles

Les auteurs dans ont répertorié les grilles en trois classes d'un point de vue topologique, par ordre croissant de l'étendue géographique et d'un point de vue de la complexité **[22]**.

Ainsi, nous distinguons les intra-grilles, les extra-grilles et les inter-grilles :

8.1. Intra-Grille

La plus simple des trois topologies est l'intra-grille, qui consiste simplement d'un ensemble de ressources et de services qui appartiennent à une organisation unique. Les principales caractéristiques d'une telle grille sont l'interconnexion à travers un réseau performant et haut débit, un domaine de sécurité unique et maıtrisé par les administrateurs de l'organisation et un ensemble relativement statique et homogène de ressources.

8.2. Extra-Grille

Une extra-grille étend le modèle en rassemblant plusieurs intra-grilles. Les principales caractéristiques d'une telle grille sont la présence d'un réseau d'interconnexion hétérogène haut et bas débit (LAN/WAN), de plusieurs

domaines de sécurité distincts, et d'un ensemble plus ou moins dynamique de ressources. Un exemple d'utilisation est lors d'alliances et d'échanges « Business-to-Business » (B2B) entre les entreprises partenaires.

8.3. Inter-Grille

Une inter-grille consiste à agréger les grilles de multiples organisations, en une seule grille. Les principales caractéristiques d'une telle grille sont la présence d'un réseau d'interconnexion très hétérogène haut et bas débit (LAN / WAN), de plusieurs domaines de sécurité différents et qui ont parfois des politiques de sécurité distinctes et mémé contradictoires, et d'un ensemble très dynamique de ressources **[22]**.

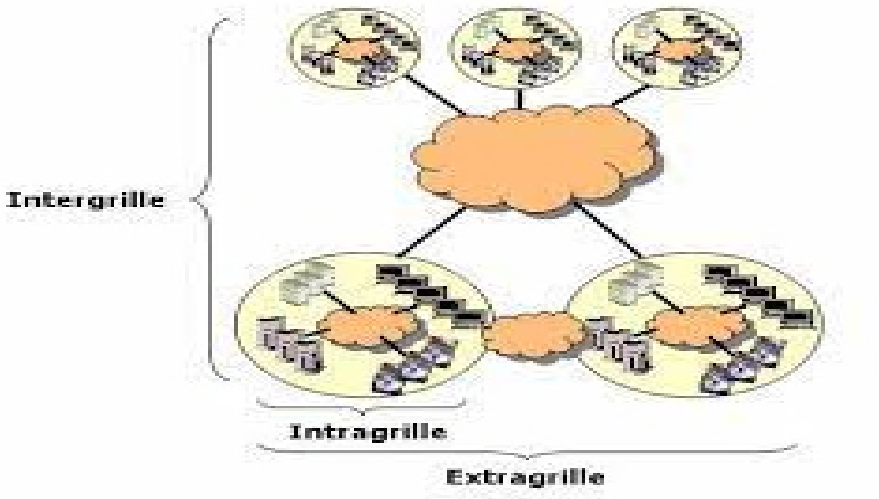

Figure 6. Topologique une Grille de calcul

9. Conclusion

Dans ce chapitre nous nous sommes intéressés à la grille de calcul, est une grande tâche de calcul est répartie entre des machines individuelles, qui exécutent des calculs en parallèle puis renvoient les résultats à l'ordinateur d'origine. Ces machines individuelles sont des nœuds d'un réseau, qui peuvent s'étendre sur plusieurs domaines administratifs et être géographiquement éloignés. Chacun des nœuds peut être considéré comme un système discret pouvant effectuer des travaux et avoir accès à un réseau. Les grilles de calcul sont souvent plus économiques que les superordinateurs de puissance de calcul égale

Chapitre III : L’équilibrage de charge

I) L'équilibrage de charge

1. Introduction

Les systèmes répartis ont connu ces dernières années un essor assez particulier grâce au potentiel de puissance de traitement qu'ils peuvent offrir, au développement des vitesses des réseaux et au faible coût de leur mise en œuvre. D'un autre côté, nous assistons à l'émergence d'applications de plus en plus exigeantes en ressources de calcul et de stockage et pour lesquelles les architectures actuelles s'avèrent insuffisantes. Parmi ces applications, nous pouvons citer celles liées à la météorologie, au calcul scientifique intensif, au datamining et à la bio informatique .

Pour répondre à de telles exigences, une solution consiste à agréger les ressources informatiques disséminées à l'échelle planétaire en les interconnectant grâce aux infrastructures réseaux existantes et en particulier Internet. C'est cette idée qui a permis l'émergence d'un nouveau concept de calcul appelé calcul de grille (Grille computing) .

Les architectures qui peuvent supporter ce type de calcul peuvent être classées selon différents points de vue. Si l'on se base sur le point de vue objectif, nous pouvons distinguer trois types de grilles : les grilles de calcul (agrégation de la puissance de calcul), les grilles de données (stockage à large échelle) et les grilles d'information (partage de la connaissance). Si par contre l'on se place du point de vue infrastructure, nous pouvons classer les grilles en grilles de calcul, grilles de PC (desktop grille) et systèmes P2P. Indépendamment de ces différentes classifications, les infrastructures de type grille se caractérisent par trois principaux éléments :

1. **Dynamité** : Dans une grille, le nombre de ressources évolue de manière dynamique (connexion et déconnexion de ressources), ce qui rend leur gestion plus complexe que dans des systèmes stables. D'un point de vue gestion, il faut donc s'adapter à ces fluctuations qui peuvent avoir un effet négatif sur les performances des applications.
2. **Hétérogénéité** : Plusieurs niveaux d'hétérogénéité existent dans une grille : matériel, systèmes d'exploitation, logiciels d'applications, réseaux, etc. Cette hétérogénéité doit évidemment être transparente par rapport aux utilisateurs et à leurs applications.
3. **Domaines administratifs multiples** : Les ressources d'une grille de calcul sont géographiquement distribuées et gérées à travers plusieurs domaines administratifs qui appartiennent à différentes organisations. Ces domaines

peuvent avoir des politiques de gestion et de sécurité qui peuvent être très différentes les unes par rapport aux autres.

La mise en œuvre d'infrastructures à large échelle peut se faire de manière relativement simple, dans la mesure où l'idée sous-jacente est de mettre en commun les infrastructures actuelles afin de construire un super calculateur virtuel. Ainsi, il est possible de construire une grille à partir de ressources existantes (PC, stations de travail, machines parallèles, etc.) en utilisant des réseaux d'interconnexion existants, et notamment l'Internet. Il est aussi possible d'utiliser une approche communautaire et volontariste dans laquelle chaque individu ou institution accepte de mettre ses ressources à la disposition de ceux qui en ont besoin et ce sans contraintes spécifiques.

La gestion de telles infrastructures pose naturellement des problèmes beaucoup plus complexes que dans le cas de systèmes répartis classiques. Parmi ces problèmes, l'optimisation de l'utilisation des ressources de calcul et de communication est un aspect très important qui mobilise beaucoup de chercheurs. En fonction des objectifs visés, cette optimisation peut être vue selon différents points de vue :

Du point de vue de l'utilisateur, il s'agit notamment de minimiser le temps d'exécution global (makespan) de son application. –

Du point de vue de l'administrateur du système, l'objectif visé consiste à optimiser les performances globales du système, en maximisant le taux d'exploitation des machines.

Nous pensons que, si le critère makespan est un critère de performance important pour les systèmes parallèles, l'exploitation optimale des grilles de calcul peut se mesurer avec deux autres métriques : un équilibrage de charge entre les ressources d'une grille et une réduction des coûts de communication. Dans cette perspective, nous proposons, dans ce papier, un modèle arborescent d'équilibrage de charge, sur lequel nous développons une stratégie d'équilibrage qui puisse répondre aux deux objectifs suivants : (i) la réduction, autant que possible, du temps de réponse moyen des tâches soumises à une grille de calcul; et, (ii) la réduction des coûts de communication en privilégiant une redistribution de la charge de calcul au sein d'une grappe plutôt qu'au niveau de la grille toute entière.

Le reste du papier est organisé comme suit : la Section 2 rappelle les principaux aspects du problème de l'équilibrage de charge. La Section 3 passe en revue les travaux existants relatifs à l'équilibrage de charge notamment pour les systèmes à large échelle. La Section 4 décrit le modèle arborescent que nous

proposons pour représenter une grille de calcul. La stratégie d'équilibrage de charge et les algorithmes associés sont décrits, respectivement, dans les Sections 5 et 6. La Section 7 présente quelques résultats relatifs à l'expérimentation de la stratégie d'équilibrage proposée. Finalement, la Section 8 conclut cet article et liste quelques perspectives de recherche.

2. Problème de l'équilibrage de charge

Le problème de l'équilibrage étant un problème relativement ancien, beaucoup d'approches ont été proposées pour le résoudre **[27]**. Casavant et Kuhl ont défini une taxonomie largement adoptée par la communauté scientifique dont les principales classes sont :

2.1-Approche statique Vs. approchedynamique : Dans une approche statique, les tâches sont assignées aux machines avant l'exécution de l'application qui les contient. Les informations concernant le temps d'exécution des tâches et les caractéristiques dynamiques des machines sont supposées connues a priori. Cette approche est efficace et simple à mettre en œuvre lorsque la charge de travail est au préalable suffisamment bien caractérisée. Dans une approche dynamique, l'assignation des tâches aux machines se décide durant la phase d'exécution, en fonction des informations qui sont collectées sur l'état de charge du système. Ceci permet d'améliorer les performances d'exécution des tâches mais au prix d'une complexité dans la mise en œuvre de cette stratégie, notamment en ce qui concerne la définition de l'état de charge du système, qui doit se faire de manière continue **[28]** .

2.**2) Approche centralisée Vs. approchedistribuée** : Dans une approche centralisée, un site du système est choisi comme coordinateur. Il reçoit les informations de charge de tous les autres sites qu'il assemble pour obtenir l'état de charge global du système. Dans le cas d'une approche distribuée, chaque site du système est responsable de collecter les informations de charge sur les autres sites et de les rassembler pour obtenir l'état global du système. Les décisions de placement de tâches sont prises localement, étant donné que tous les sites ont la même perception de la charge globale du système **[24]**.

2.3) Approche source-initiative Vs. receveur-initiative : L'approche source-initiative est appliquée lorsqu'un site, appelé source, détecte qu'il a une surcharge de travail et qu'il cherche à transférer le surplus vers un site

faiblement chargé. L'approche receveur initiative s'applique lorsqu'un site faiblement chargé, appelé receveur, demande à recevoir tout ou partie du surplus des sites surchargés **[27]**.

3. Politiques et mécanismes d'équilibrage de charge

Un système d'équilibrage de charge est composé de deux éléments essentiels : les politiques et les mécanismes. Les politiques considèrent l'ensemble des choix à effectuer pour distribuer une charge de travail alors que les mécanismes réalisent physiquement la répartition de la charge et fournissent les informations exigées par les politiques. La figure 1 illustre la décomposition arborescente d'un système d'équilibrage de charge **[24]**.

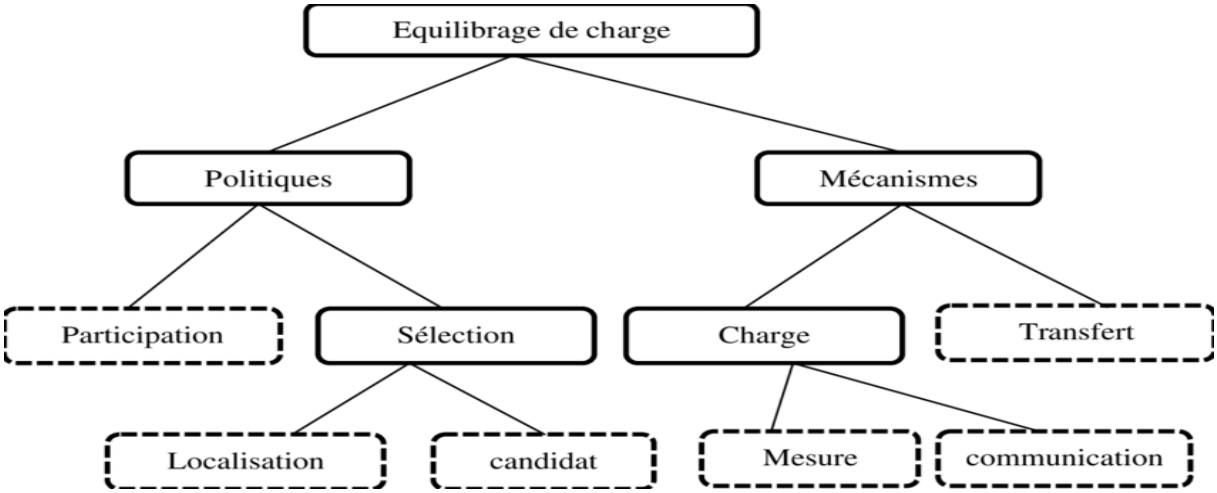

Figure 7. Composants d'un système d'équilibrage de charge

– Politique de participation : Le but de cette politique consiste à déterminer si un site est dans un état approprié pour participer à un transfert de tâches comme source (site surchargé) ou comme receveur (site sous-chargé).

–Politique de sélection de la localisation : Cette politique est responsable de trouver, pour un site donné, un partenaire (source ou receveur), une fois que la politique de participation a décidé que ce site était soit source, soit receveur.

– Politique de sélection des tâches à transférer : Une fois que les politiques de participation et de localisation ont décidé qu'un site Si est source et qu'un autre site Sj est receveur, cette politique est responsable du choix des tâches à transférer de Si vers Sj .

– Mécanisme de mesure de la charge : Dans toute approche d'équilibrage de charge, une des difficultés majeures est celle qui consiste à évaluer la mesure de la charge d'un site **[25]**. Dans la plupart des travaux existants, c'est la longueur

de la file d'attente qui détermine la charge d'un site. Certains auteurs préconisent comme indicateur de charge, une combinaison entre la longueur de la file d'attente CPU, celle des Entrées/Sorties et l'occupation mémoire. Dans le cas des grilles de calcul, il est nécessaire de tenir compte aussi de l'hétérogénéité des ressources et des réseaux de communication pour mesurer la charge d'un site.

– Mécanisme de définition de la charge : Ce mécanisme essaie de définir la charge globale d'un système en collectant les informations de charge (partielles) sur l'ensemble ou une partie des sites du système. Il faudra alors définir les méthodes selon lesquelles l'information de charge est collectée puis diffusée aux sites **[26]**.

4. Problématiques particulières liées aux grilles de calcul

Les spécificités des grilles de calcul font que le problème d'équilibrage de charge devient beaucoup plus complexe que dans le cas des systèmes répartis classiques ou dans

Le cas de grappes (clusters), qui peuvent offrir une certaine homogénéité et stabilité des ressources qui les composent. Par contre, dans des systèmes à large échelle comme les grilles de calcul, ces hypothèses ne sont pas tout à fait réalistes. Les grilles de calcul sont susceptibles d'être largement distribuées et fortement hétérogènes. Par conséquent, il est essentiel de considérer l'impact de ces caractéristiques dans la conception et l'analyse de techniques d'équilibrage de charge. La plus grande difficulté réside dans l'estimation de la charge globale de toute la grille. Ainsi, tout système d'équilibrage de charge d'une grille devra, en premier lieu, permettre d'estimer l'état de charge de chaque ressource de cette grille. Il s'agira notamment de préciser :

– Quels critères retenir pour la définition de la charge d'une ressource ?

– Comment mesurer cette charge ?

– Comment intégrer toutes les informations liées à l'hétérogénéité des ressources pour obtenir une moyenne représentative de la charge instantanée de tout le système ?

5. Modèle arborescent d'équilibrage de charge

5.1. Topologie générale d'une grille

D'un point de vue topologique (voir figure 2), Berstis et al répertorient les grilles en trois niveaux par ordre croissant d'étendue géographique et de complexité : Intra-grappe, Extra-grappe et Inter-grappes.

– Intra-grappe (par analogie à Intranet) : Cette grille est composée d'un ensemble relativement simple de ressources (Éléments de calcul et Éléments de stockage) appartenant à une organisation unique. Les principales caractéristiques d'une telle topologie sont la présence d'un réseau d'interconnexion performant et haut-débit et d'un ensemble relativement statique et homogène de ressources.

– Extra-grappe (par analogie à Extranet) : Ce type de grille étend la topologie précédente en agrégeant plusieurs grappes. Une extra-grappe est caractérisée par la présence d'un réseau d'interconnexion hétérogène haut et bas débit (LAN/WAN) et d'un ensemble plus ou moins dynamique de ressources.

– Inter-grappes (par analogie à Internet) : Cette topologie consiste à fédérer les grilles de multiples organisations en une seule grille générale. Ses principales caractéristiques sont la présence d'un réseau d'interconnexion très hétérogène haut et bas débit (LAN/WAN) et d'un ensemble dynamique et fortement hétérogène de ressources

5.2. Modèle de représentation proposé

Pour représenter une grille nous proposons de la transformer, de manière univoque, en un arbre d'interconnexion virtuel. Tel qu'illustré par la figure 3, cet arbre peut être instancié en trois configurations notées, respectivement : **G/C/M, 1/C/M** et **1/1/M**, où G représente le nombre d'extra-grappes, C le nombre d'intra-grappes et **M** le nombre total d'éléments de calcul (**EC's**) d'une grille.

5.2.1. Modèle 1/1/M

Cette instance du modèle représente la plus petite grille possible, à savoir une seule grappe composé de M éléments de calcul. Le schéma relatif à cette instance est le modèle 1/1/M de la figure 3 qui comprend deux niveaux définis comme suit :

1) Le niveau racine de cet arbre correspond à l'unique grappe de la grille. Appelée Gestionnaire des EC'S, cette racine a pour rôle de :

- Gérer l'information de charge relative aux EC'S de la grappe. - Maintenir l'état de charge de la grappe.
- Décider de déclencher un équilibrage local, que nous appellerons équilibrage intra-grappe.
- Informer les EC'S, pour mettre en œuvre l'équilibrage décidé par le gestionnaire.

2) Le niveau feuilles de l'arbre, où chaque feuille correspond à un EC ayant pour fonction de :
- Maintenir à jour l'information de l'état de charge de l'EC correspondant.
- Envoyer périodiquement cette information à son gestionnaire.
- Exécuter les opérations d'équilibrage décidées par son gestionnaire.

5.2.2. Modèle 1/C/M

Ce modèle représente une extra-grappe. C'est une extension du modèle précédent, dans le sens où l'on passe d'une seule grappe à C grappes. Nous obtenons ainsi le modèle 1/C/M représenté par un arbre à trois niveaux défini comme suit :

1) Le niveau racine, appelé Gestionnaire des grappes, a pour fonctions de :
- Gérer l'information de charge relative à chaque grappe sous son contrôle.
- Maintenir l'état de charge de l'extra-grappe.
- Décider d'un équilibrage local au niveau d'une grappe, que nous appellerons équilibrage extra-grappe.
- Envoyer les décisions d'équilibrage aux gestionnaires des EC's associés.

2) Le deuxième niveau est associé aux grappes de l'extra-grappe. Chaque nœud de ce niveau joue le même rôle que dans le modèle 1/1/M.

3) Le troisième niveau correspond aux éléments de calcul (feuilles de l'arbre) tel que défini dans le modèle précédent.

5.2.3. Modèle G/C/M

C'est le modèle générique qui correspond à une grille (inter-grappes). L'arbre correspondant a quatre niveaux représentant l'agrégation de G modèles de type 1/C/M, que l'on peut définir comme suit :

1) La racine de cet arbre, appelée Gestionnaire de grille, a pour fonctions de :
- Maintenir l'information de charge de l'ensemble de la grille.
- Décider d'un équilibrage global entre les extra-grappes de la grille, que nous appellerons équilibrage inter-grappes.
- Envoyer les décisions d'équilibrage aux gestionnaires de grappes pour exécution .

2) Les trois autres niveaux sont identiques à ceux du modèle 1/C/M.

5.2.4. Caractéristiques du modèle proposé

Le modèle de représentation défini ci-dessus peut être caractérisé comme suit :
1) La modélisation d'une grille en arbre s'effectue par une transformation univoque. A chaque grille correspond un et un seul arbre de représentation et ce quelque soit la complexité topologique de la grille. Nous pouvons considérer cet arbre comme un arbre de recouvrement d'une grille, qui nous permettra de définir une stratégie d'équilibrage.
2) Le modèle arborescent proposé supporte le passage à l'échelle en terme de ressources (ajout et retrait d'un élément de calcul, d'une grappe ou d'une extra-grappe).
3) La structure hiérarchique du modèle facilite les flux d'informations à travers les nœuds de l'arbre. En terme de flux d'informations, nous distinguons trois types :
- **Flux montant** : Ce flux concerne la circulation des informations de charge pour définir un état de charge à différents niveaux (intra/extra/inter-grappe(s)).
- **Flux horizontal** : Il concerne les informations nécessaires à l'exécution des opérations d'équilibrage de charge. –

Flux descendant : Ce flux permet de véhiculer les décisions d'équilibrage prises par les gestionnaires correspondants aux différents niveaux du modèle.

6. Stratégie d'équilibrage de charge

6.1. Principes

La structure arborescente du modèle proposé nous permet de développer une stratégie hiérarchique à trois niveaux d'équilibrage : Intra-grappe, Extra-grappe et Inter-grappes.

1) Équilibrage Intra-grappe : Dans ce premier niveau, chaque gestionnaire d'éléments de calcul décide de déclencher une opération d'équilibrage en fonction de la charge courante de la grappe qu'il gère. Cette charge est estimée à partir des différentes informations de charge envoyées périodiquement par les EC's qui composent la grappe. Le gestionnaire tente, en priorité, d'équilibrer la charge de la grappe localement en la répartissant entre les EC's qui lui appartiennent. Cette approche de localité a pour objectif de réduire les coûts de communication, en évitant les communications extra/inter grappes.

2) Équilibrage Extra-grappe : Dans ce deuxième niveau, l'équilibrage se fait à l'échelle des extra-grappes. Il intervient dans le cas où certains gestionnaires des

EC'S n'ont pas réussi à équilibrer localement leurs charges. Il y aura ainsi transfert de tâches entre grappes surchargées et grappes sous-chargées de la même extra-grappe. Dans un souci de réduire au maximum les coûts de communication, les grappes réceptrices seront sélectionnées en fonction des débits des réseaux.

3) Équilibrage Inter-grappes : Dans ce troisième niveau, l'équilibrage de charge n'est déclenché que si un ou plusieurs gestionnaires de grappes n'arrivent pas équilibrer leurs charges localement entre les grappes qu'ils gèrent. Dans ce cas extrême, il sera alors nécessaire au gestionnaire de grille de transférer un certain nombre de tâches à partir d'extra-grappes surchargées vers d'autres qui sont sous-chargées.

6.2. Description générique de la stratégie d'équilibrage

A n'importe quel niveau du modèle défini ci-dessus, nous proposons une stratégie d'équilibrage composée de trois étapes. Comme la description de cette stratégie se fera de manière générique, nous utiliserons les notions de groupe et d'élément. Un groupe peut désigner, selon les cas, soit une intra-grappe, une extra-grappe ou toute la grille. Un élément est un composant d'un groupe (un EC, une grappe ou une extra-grappe).**[28]**

1) Estimation de la charge du groupe : Cette étape définit les mécanismes de mesure et de communication de charge. Connaissant le nombre de ses éléments ainsi que leurs capacités respectives, chaque gestionnaire estime les capacités du groupe auquel il est associé en effectuant les actions suivantes :

- Estimation de la charge courante du groupe, sur la base des informations reçues périodiquement à partir de ses éléments.
- Calcul de l'écart type sur les charges de travail des éléments dans le but de mesurer l'étendue des variations de charge entre un groupe et ses éléments. Afin de prendre en considération l'hétérogénéité des EC's, nous proposons comme indice de charge, le temps d'exécution 1 noté T EX = LOD /SPD .
- Envoi de l'information de charge à son gestionnaire associé.

2) Prise de décision : Durant cette étape, le gestionnaire du groupe décide de l'opportunité de déclencher un équilibrage de charge local. Pour cela, il exécute les actions suivantes :

- **Définition de l'état de charge d'un groupe :** Sachant que l'écart type σ mesure la variation moyenne entre le temps d'exécution des éléments et celui de leur groupe associé, nous pouvons dire qu'un groupe est en état d'équilibre

lorsque cet écart est relativement faible. Cela signifie que le temps d'exécution de chaque élément converge vers le temps d'exécution de son groupe.

État d'équilibre : En pratique il s'agit de définir un seuil d'équilibre, noté ε, à partir duquel nous pouvons dire que l'écart type σ tend vers zéro et donc le groupe est en état d'équilibre. Ainsi nous pouvons écrire : **Si** ($\sigma \leq \varepsilon$) Alors le groupe est équilibré sinon le groupe est en état de déséquilibre.

État de saturation : Un groupe peut être déséquilibré tout en étant saturé. Dans ce cas précis, il n'est pas utile d'entamer un équilibrage local, puisque le groupe restera surchargé. Pour mesurer la saturation d'un groupe, nous définissons un autre seuil, noté δ, que nous appellerons seuil de saturation.

- Partitionnement du groupe : Lorsqu'un groupe non saturé est déséquilibré, nous pouvons envisager le déclenchement d'une opération d'équilibrage de charge. Pour déterminer si un élément d'un groupe est dans un état approprié pour participer à un transfert de tâches comme source ou comme receveur, nous partitionnons le groupe en trois classes d'éléments : les éléments surchargés (sources), les éléments équilibrés (neutres) et les éléments sous-chargés (receveurs). Cette classification dépend de l'écart entre l'indice de charge de chaque élément et celui de son groupe.

6.3. Transfert de tâches : Pour réaliser une opération d'équilibrage de charge, nous proposons l'heuristique suivante :

a) Calculer la disponibilité en terme de capacité de calcul, qui correspond à la charge totale offerte par les éléments receveurs.

b) Calculer la demande, i.e., la charge totale requise par l'ensemble des éléments sources.

c) Si l'offre n'est pas en mesure de satisfaire suffisamment la demande (écart trop grand), il n'est pas recommandé d'entamer un équilibrage local. Pour mesurer l'offre par rapport à la demande, nous définissons un seuil noté ρ, que nous appellerons seuil d'espérance. **Si** (OF F RE/ DEMANDE > ρ) Alors Équilibrage local Sinon Équilibrage au niveau supérieur.

d) Effectuer un transfert de charge en tenant compte des coûts de communication.

1. Le temps d'exécution d'une entité (groupe ou élément) est le rapport entre la charge (LOD exprimée en nombre d'unités de calcul) et la vitesse (SP D exprimée en nombre d'unités de calcul exécutées par unité de temps) de cette entité.

-Pour la sélection des tâches à transférer, nous pouvons utiliser l'un des critères suivants :

- **Plus petit temps d'exécution :** Donner la priorité de transfert à la tâche qui dispose du plus petit temps d'exécution.
- Plus grand temps d'exécution : Priorité à la tâche ayant le plus grand temps d'exécution.
- **FIFO** : Transférer la tâche la plus âgée.
- **LIFO** : Transférer la tâche la plus jeune.
- **Aléatoire** : Choix aléatoire.

7. Algorithmes d'équilibrage

Nous définissons 3 niveaux d'algorithmes : Intra-grappe, Extra-grappe et Inter-grappes **[29]**.

7.1 – **Algorithme d'équilibrage Intra-grappe** : Cet algorithme constitue le noyau de notre stratégie. L'approche de localité adoptée fait que c'est le niveau d'équilibrage qui sera le plus fréquemment sollicité. Il est déclenché lorsqu'un gestionnaire d'EC'S constate qu'il y a déséquilibre entre ses EC'S. Pour faire ce constat, le gestionnaire reçoit, de manière périodique, les informations de charge à partir de chaque élément de calcul. Sur la base de ces informations et du seuil d'équilibrage estimé ε, il analyse de manière régulière la charge du groupe. En fonction du résultat de cette analyse, soit il décide de déclencher un équilibrage local en cas de déséquilibre, soit il décide d'informer son gestionnaire du niveau supérieur sur sa charge actuelle.

7.2 – **Algorithme d'équilibrage Extra-grappe** : Cet algorithme, qui utilise une approche source-initiative, est exécuté uniquement lorsque certains gestionnaires d'EC'S n'ont pas réussi à équilibrer localement leur charge pour cause de saturation ou d'offre insuffisante. Dans ce cas, le gestionnaire de grappes tente d'équilibrer la charge globale de l'extra-grappe à travers les éléments qu'il gère. Contrairement à l'algorithme intra-grappe, l'équilibrage extra-grappe devra tenir compte des coûts de communication. Durant le transfert de tâches, nous choisirons comme grappe réceptrice, celle qui nécessite le plus petit coût de transfert. Ainsi, le critère de sélection sera pondéré par le coût de communication (coût de transfert de tâches) pour assurer qu'une tâche ne peut être transférée que lorsque son temps de réponse estimé dans la grappe réceptrice, auquel nous rajoutons le temps de transfert à partir de la grappe source, est meilleur que son temps de réponse dans la grappe source**. [30]**

7.3 – **Algorithme d'équilibrage Inter-grappes** : Ce troisième niveau d'algorithme procède à un équilibrage global à travers toutes les extra-grappes de la grille. Il est exécuté dans le cas extrême où la majorité des gestionnaires de

grappes n'arrivent pas à équilibrer localement leur surplus de charge. Le recours à ce type d'équilibrage ne sera effectif que si les coûts de communication, associés aux différentes tâches à transférer, seront réellement intéressants. En d'autres termes, il serait plus judicieux de laisser un ensemble de tâches attendre la libération de ressources que de procéder à leur transfert vers d'autres sites sans un gain de temps substantiel. Dans ce qui suit, nous allons décrire l'algorithme générique associé à notre stratégie**. [30]**

8. Conclusion :

Dans ce chapitre, nous nous sommes intéressé au problème L'équilibrage de charge qui vous permet de faire évoluer votre système et de prendre en charge un plus grand nombre d'utilisateurs, tout en les faisant bénéficier de prestations performantes et fiables, sans point de panne unique.

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

II) Les Algorithme de l'équilibrage de charge

1-Introduction :

L'ordonnancement dans les grilles informatique peut être vu comme une famille de problèmes. Ceci est dû au nombre de paramètres qui interviennent aussi bien dans l'ordonnancement que dans les différents besoins des applications.

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 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.

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.

2. Formulation du problème d'ordonnancement dans un Environnement cloud :

Un problème d'ordonnancement est caractérisé par deux ensembles :

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 ces algorithmes se sont focalisés essentiellement sur la rapidité d'exécution des tâches.

3. 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.

3. **1- L'heuristique Min-Min :**

L'algorithme Min-Min commence par un arrangement de toutes les taches non assignées. Premièrement, le temps d'exécution minimum pour toutes les tâches est trouvé. Les tâches ayant un temps d'exécution minimum sont d'abord choisies. Deuxièmement, le temps d'exécution pour toutes les autres tâches est repensé pour cette machine Dans la suit ont un petite exemple simple de trois machines et treize taches et nous appliquons le fonctionnement de l'algorithme d'initialisation Min-Min **[04].** La matrice ETC est donnée par le tableau 2

Algorithmes Min-Min

```
While (Critère d'arrêt n'est pas satisfait) do
                                        {
//le temps d'exécution de chaque machine égale à la somme des temps
d'exécution des taches
                                qu'elles lui sont affectées.
                                  For toutes les Temps T
                                        {
                                For toutes les machines mi {
                                    For toutes les tâches tj
                                        {
Chercher minij la valeur minimum de l'ETC ; // le minimum temps d'exécution
de la
                                tâche tj dans la machine mi.
                                        }
                                For toutes les machines mj
                                        {
Tempexei = temp_exei +ETCij. // le temps d'exécution de la tâche qui a
minij dans la machine mi.
```

}
For toutes les machines mi
{
Trouver la machine qui a temp_exei minimum.
}
Affecter cette tâche tj à la machine qui a le minimum de qui temp_exei .
If (toutes les taches sont traitées)
Critère d'arrêt est satisfait ;
}

Figure 8 : Algorithmes Min-Min

Machines	**Taches**												
	T1	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**T10**	**T11**	**T12**	**T13**
M1	30	46	66	99	6	7	65	88	140	56	8	700	46
M2	50	26	71	86	62	20	101	75	45	14	600	83	17
M3	5	33	49	70	11	87	60	15	44	38	150	13	179

Tableau 2. Un exemple de la matrice ETC.

Dans cette étape nous choisissons la tâche qui termine la première, ensuit le temps d'exécution minimum pour T1sera réalisé sur la M3, comme le montre le tableau 3

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

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

Donc, dans chaque itération nous comparons les dates de fin d'exécution de toutes les tâches non encore ordonnancées et nous prenons le minimum de ces dates. Ensuite, l'algorithme d'initialisation affectera la tâche **T5** à la **M1**

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

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

Puis la tache T1 à M1, et on va continuera avec le même principe avec les restes taches :

L'algorithme d'initialisation affecter la tache **T6**à la machine **M1.**

Machines	Taches													Temps d'exécution
	T1	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**T10**	**T11**	**T12**	**T13**	
M1	0	_	_	_	6	7	_	_	_	_	_	_	_	13
M2	0	_	_	_	0	0	_	_	_	_	_	_	_	0
M3	5	_	_	_	0	0	_	_	_	_	_	_	_	5

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

L'algorithme d'initialisation affecter la tache **T11**à la machine **M1.**

Machines	Taches													Temps d'exécution
	T1	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**T10**	**T11**	**T12**	**T13**	
M1	0	_	_	_	6	7	_	_	_	_	8	_	_	21
M2	0	_	_	_	0	0	_	_	_	_	0	_	_	0
M3	5	_	_	_	0	0	_	_	_	_	0	_	_	5

Tableau 6. La quatrième itération de l'algorithme Min-Min sur ETC

L'algorithme d'initialisation affecter la tache **T12**à la machine **M3.**

Machines	Taches													Temps d'exécution
	T1	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**T10**	**T11**	**T12**	**T13**	
M1	0	_	_	_	6	7	_	_	_	_	8	0	_	21
M2	0	_	_	_	0	0	_	_	_	_	0	0	_	0
M3	5	_	_	_	0	0	_	_	_	_	0	13	_	13

Tableau 7. La cinquième itération de l'algorithme Min-Min sur ETC

L'algorithme d'initialisation affecter la tache **T10**à la machine **M2**

Machines	Taches													Temps d'exécution
	T1	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**T10**	**T11**	**T12**	**T13**	
M1	0	_	_	_	6	7	_	_	_	0	8	0	_	21
M2	0	_	_	_	0	0	_	_	_	14	0	0	_	14
M3	5	_	_	_	0	0	_	_	_	0	0	13	_	13

Tableau 8.6 itération de l'algorithme Min-Min sur ETC

L'algorithme d'initialisation affecter la tache **T8**à la machine **M3**

Machines	Taches													Temps d'exécution
	T1	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**T10**	**T11**	**T12**	**T13**	
M1	0	_	_	_	6	7	_	0	_	0	8	0	_	21
M2	0	_	_	_	0	0	_	0	_	14	0	0	_	14
M3	5	_	_	_	0	0	_	15	_	0	0	13	_	28

Tableau 9. 7 Itération de l'algorithme Min-Min sur ETC

L'algorithme d'initialisation affecter la tache **T2**à la machine **M2**

Machines	Taches													Temps d'exécution
	T1	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**T10**	**T11**	**T12**	**T13**	
M1	0	0	_	_	6	7	_	0	_	0	8	0	_	21
M2	0	26	_	_	0	0	_	0	_	14	0	0	_	40
M3	5	0	_	_	0	0	_	15	_	0	0	13	_	28

Tableau 10. 8 Itération de l'algorithme Min-Min sur ETC

A la fin nous aurons la matrice ETC finale donnée par le tableau 11, avec un temps d'exécution globale égale à 266 second.

Machines	Taches													Temps d'exécution
	T1	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**T10**	**T11**	**T12**	**T13**	
M1	0	0	0	0	6	7	0	0	0	0	8	0	0	21
M2	0	26	0	0	0	0	0	0	0	14	0	0	17	57
M3	5	0	49	70	0	0	60	15	44	0	0	13	0	266

Tableau 11.ETC finale de l'algorithme.

3.2-Equilibrage de charge SLB :

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

Notre contribution consiste à implémenter l'heuristique **SLB** (Simple LoadingBalancing). Elle 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 Machines Taches Temps possédant le temps d'exécution minimum afin de minimiser le makespan montre (figure 9).

Machine avec maximum temps d'exécution	T1	T2			Tj	Tn

Machine avec maximum temps d'exécution	T1	T2			Tj	Tn

Figure.9 Stratégie de SLB.

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

Algorithme SLBE

Figure 10: Algorithme d'équilibrage de charge

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

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

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

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

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

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

{Affecter tâche t_j sur m_{min} ;

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

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

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

{

Affecter $C_{m_{maxj}}$à l'auxiliaire ;

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

} }

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

Else

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

}

A titre exemple, nous allons ordonnancer les taches de l'exemple précédent par la matrice ETC initialisé par l'algorithme Min-Min**,** voir le tableau3.7 ou la valeur du makespan sur **M3** est de **266** . La première itération de l'algorithme a réaffecté la tache**T1** de la **M3** à la **M1** comme indiqué dans le tableau 12 cette réaffectation permet de réduire le makespan de **266** à**161** .

Machines	**Taches**													**Temps d'exécution**
	T1	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**T10**	**T11**	**T12**	**T13**	
M1	5	0	0	0	6	7	0	0	0	0	8	0	0	26
M2	0	26	0	0	0	0	0	0	0	14	0	0	17	57
M3	0	0	49	70	10	0	60	15	44	0	0	13	0	261

Tableau.12 Processus d'exécution de l'algorithme SLB- itération 1.

Dans la deuxième itération, l'algorithme SLB considère **M3** pour la réaffectation. La tâche **T5** est réaffectée sur **M1** et la makespan est raffiné à **42**, comme le montre le tableau.13

Machines	**Taches**													**Temps d'exécution**
	T1	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**T10**	**T11**	**T12**	**T13**	
M1	5	0	0	0	16	7	0	0	0	0	8	0	0	42
M2	0	26	0	0	0	0	0	0	0	14	0	0	17	57
M3	0	0	49	70	0	0	60	15	44	0	0	13	0	251

Tableau.13 Processus d'exécution de l'algorithme SLB- itération 2.

Dans troisième itération, l'algorithme SLB considère **M3** pour la réaffectation. La tâche **T4** est réaffectée sur **M1** et la makespan est raffiné à **112**, comme le montre le tableau.14.

Machines	**Taches**													**Temps d'exécution**
	T1	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**T10**	**T11**	**T12**	**T13**	
M1	**5**	**0**	**0**	**70**	**16**	**7**	**0**	**0**	**0**	**0**	**8**	**0**	**0**	**112**
M2	**0**	**26**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**14**	**0**	**0**	**17**	**57**
M3	**0**	**0**	**49**	**0**	**0**	**0**	**60**	**15**	**44**	**0**	**0**	**13**	**0**	**181**

Tableau.14 Processus d'exécution de l'algorithme **SLB- itération 3.**

Dans le quatrième itération, l'algorithme SLB considère **M3** pour la réaffectation. La tâche **T7** est réaffectée sur **M2** et la makespan est raffiné à **117**, comme le montre le tableau.15

Machines	Taches													Temps d'exécution
	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	
M1	5	0	0	70	16	7	0	0	0	0	8	0	0	112
M2	0	26	0	0	0	0	60	0	0	14	0	0	17	117
M3	0	0	49	0	0	0	0	15	44	0	0	13	0	121

Tableau.15 Processus d'exécution de l'algorithme **SLB- itération 4.**

Dans le **5** itération, l'algorithme SLB considère **M3** pour la réaffectation. La tâche **T3** est réaffectée sur M1 et la makespan est raffiné à **161**, comme le montre le tableau.16

Machines	Taches													Temps d'exécution
	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	
M1	5	0	49	70	16	7	0	0	0	0	8	0	0	161
M2	0	26	0	0	0	0	60	0	0	14	0	0	17	117
M3	0	0	0	0	0	0	0	15	44	0	0	13	0	72

Tableau.16 Processus d'exécution de l'algorithme **SLB- itération 5.**

De même notre algorithme continue de raffiner le makespan, en déplaçant les taches qui réduisent la valeur du makespan de la machine MMaxà la machineMMin, jusqu'à que la valeur de makespan ne peut pas être raffinée. D'apré les résultats de l'algorithme d'ordonnancement Min-Min, l'algorithme d'équilibrage de charge simple donne le meilleur résultat du makespan, ce dernier est réduit de 266 à 161,

3.3- Equilibrage de charge avec Echange LBE :

Pour améliorer les performances de l'heuristique SLB, une nouvelle technique d'équilibrage de charge est proposé, c'est l'équilibrage de charge avec échange LBE (LoadingBalancing Exchange). Cette technique généralise l'ordonnancement SLB. Son principe est de minimiser le temps d'exécution

maximal obtenu par l'ordonnancement, en réattribuant les taches dans le but d'améliorer le temps nécessaire de la solution d'ordonnancement Min-Min, tout en minimisant le temps d'exécution maximum.

LBE considère la machine ayant le temps d'exécution maximum pour le réduire, par l'échange des taches avec les autres machines et par la réaffectation des taches entre elles. Pour chaque tache tj de la machine ayant le temps maximal d'exécution, et pour chaque tâche tk des autres machines, on effectue un échange de la tâche tk. Nous réaffecterons Une seul paire d'échange des taches en même temps qui donne le makespan minimum parmi toutes les affectations de tâche entre les machines.

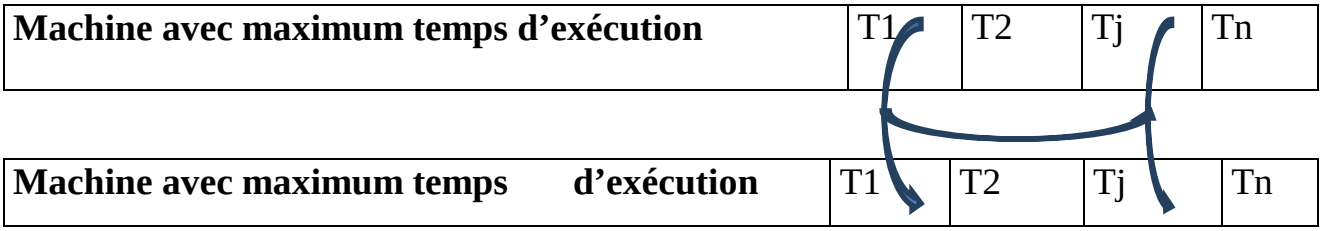

Figure11. Stratégie d'échange LBE**.**

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

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

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

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

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

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

{Affecter tâche t_j sur m_{min} ;

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

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

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

{

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

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

} }

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

Else

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

}

Figure 12- pseudo code de l'algorithme SLB

Nous allons ordonnancer les taches précédentes données par la matrice ETC initialisée par l'algorithme Min-Min, voir le tableau ou la valeur du makespan sur**M3** est de.La première itération de l'algorithme a réaffecte la tache **T12** de la **M3** à la **M1**, la tâche T **4** de la **M1** à la **M3** comme indiqué dans le tableau .16. Cette réaffectation permet de réduire le makespan de**266**à 121 .

Machines	**Taches**													**Temps d'exécution**
	T1	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**T10**	**T11**	**T12**	**T13**	
M1	**5**	**0**	**49**	**0**	**16**	**7**	**0**	**0**	**0**	**0**	**8**	**13**	**0**	**104**
M2	**0**	**26**	**0**	**0**	**0**	**0**	**60**	**0**	**0**	**14**	**0**	**0**	**17**	**117**
M3	**0**	**0**	**0**	**70**	**0**	**0**	**0**	**15**	**44**	**0**	**0**	**0**	**0**	**121**

Tableau.17- Processus d'exécution de l'algorithme LBE itération 1.

Dans la deuxième itération, la **M3 est** la machine mmax ou la valeur de makespan est **121** . La tâche **T8**de la **M3**est réaffectée sur la **M1,** et la tâche**T1**de la **M1** est réaffectée sur la **M3**. Le makespan est raffiné **121** à **111**, comme il est indiqué dans le tableau .17.

Machines	**Taches**													**Temps d'exécution**
	T1	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**T10**	**T11**	**T12**	**T13**	
M1	**0**	**0**	**49**	**0**	**16**	**7**	**0**	**15**	**0**	**0**	**8**	**13**	**0**	**114**
M2	**0**	**26**	**0**	**0**	**0**	**0**	**60**	**0**	**0**	**14**	**0**	**0**	**17**	**117**
M3	**5**	**0**	**0**	**70**	**0**	**0**	**0**	**0**	**44**	**0**	**0**	**0**	**0**	**111**

Tableau.18- Processus d'exécution de l'algorithme LBE itération 2.

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

Comme le montre le tableau .15(.16) l'algorithme d'équilibrage de charge avec échange LBE donne le meilleur résultat du makespan, ce dernier est réduit de à, voir letableau.17.

4- Conclusion :

Dans ce chapitre, il a été question de proposer deux approches pour notre problème. Ces deux approches sont SLB et LBE, ceux-ci visent à minimiser le (makespan) pour s'approcher le plus possible de la solution optimale.

Le problème d'ordonnancement consiste à organiser dans le temps l'exécution de tâches, compte tenu de contraintes temporelles et de contraintes portant sur l'utilisation et la disponibilité des ressources requises.

Les problèmes d'ordonnancement étaient déjà difficiles en environnement homogène, il n'y a aucune raison qu'ils se simplifient en environnement hétérogène, nous avons traité dans ce chapitre les solutions apportées dans la littérature au problème d'équilibrage de charge.

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

Chapitre IV : Implémentation, résultats & discussion

1. Introduction

A travers ce chapitre, nous présenterons notre contribution et l'apport de notre travail en faisant une synthèse comparative entre notre méthode et les méthodes existantes, avec des discussions des différents résultats obtenus.

On présentera, également, la conception de notre application et sa modélisation ainsi que ses différentes interfaces.

2. Conception de l'application

Nous avons choisi le langage UML (UnifiedModelingLanguage) en raison de son efficacité et sa simplicité de modélisation et de représentation des systèmes entiers. Nous présenterons brièvement le fonctionnement de l'application développée, les différents diagrammes de sa mise en oeuvre.

2.1 Expression des besoins

Notre projet consiste à proposer et implémenter des algorithmes d'ordonnancement des taches indépendantes dans un environnement hétérogène. Vu les difficultés rencontrées avec les résultats fournis, nous avons opté pour faire une seconde implémentation qui facilite la gestion des données, résultats et méthodes.

2.2 Exigences fonctionnelles

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

➢**Formulations des demandes et exigences par l'utilisateur :**cette option permettra à l'utilisateur de formuler ses demandes et ses exigences surtout en termes de minimisation du temps de calcul.

➢**Exécution de projet :**lors de l'exécution de notre application, les demandes de l'utilisateur pourront être satisfaites avec les meilleurs résultats possibles dans les meilleurs délais.

➢**Traitement des problèmes :**si un problème survient lors l'exécution, cela veut dire que les données fournis par l'utilisateur sont erronées ou bien mal introduites, et ce dernier devra modifier ses entrées.

2.3 Exigences non fonctionnelles

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

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

➢ L'application doit être facile à utiliser car les interfaces trop riches et complexes demandent un effort supplémentaire de compréhension.

➢ Le système doit pouvoir gérer plusieurs opérations ; tel que la vue des résultats au fur et à mesure des calculs et la consultation de l'aide.
➢ Le système doit pouvoir gérer plusieurs opérations, tel que les résultats.

3 Présentation du langage de modélisation

Ce travail a été réalisé avec le langage de la modélisation unifié UML, cette méthode représente un moyen de spécifier, représenter et construire les composantes d'un système informatique ; UML (UnifiedModelingLanguage) est un langage unifié pour la modélisation objet.
UML est conçu pour modéliser divers types de systèmes, de taille quelconque et pour tous les domaines d'application (gestion, scientifique, temps réel, système embarqué).
UML se compose d'une part des éléments de modélisation qui représentent toutes les propriétés du langage et d'autre part des diagrammes (de cas d'utilisation, de classes, d'objets, d'états-transitions, d'activités, de séquence, de collaboration, de composants et de déploiement) qui en constituent l'expression visuelle et graphique.

3.1 Diagramme de cas d'utilisation

➢**Identification des acteurs** : Dans notre application on a un seul acteur humain est un seul acteur de system.
➢**L'utilisateur :**C'est lui qui prend en charge la gestion des données d'entrées, le traitement des méthodes et la gestion des résultats.
➢**Identification des cas d'utilisation :**Après avoir défini l'acteur principal de l'application, nous passerons à la détermination de ces cas d'utilisation qui illustrent le comportement du système en réponse à une interaction avec l'utilisateur.

On peut associer à l'utilisateur plusieurs cas d'utilisation du système, Notre diagramme de cas est illustré dans la figure 13.

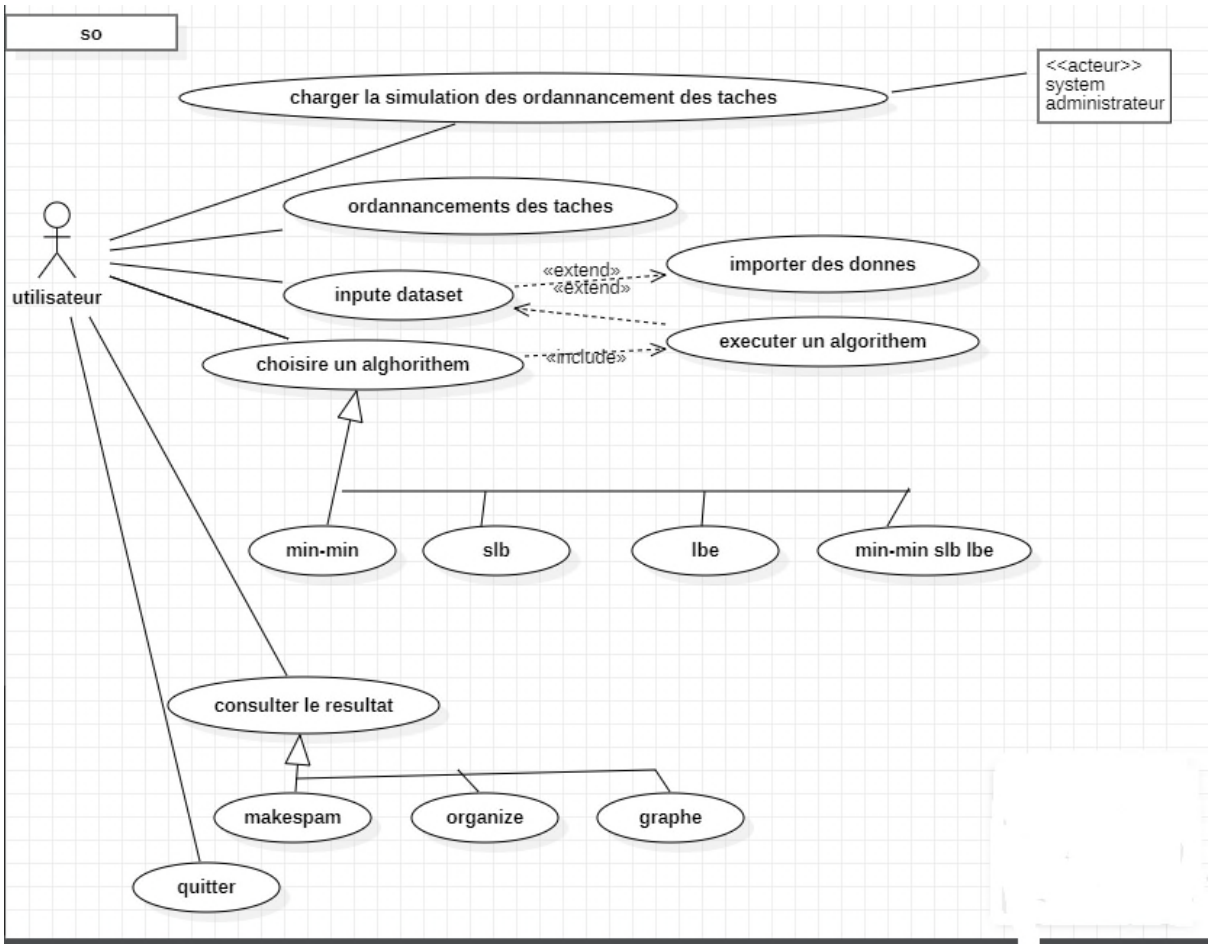

Figure 13 : Diagramme de cas d'utilisation.

- **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 (datas set).

4. Implémentation de l'application

4.1. Présentation du langage de programmation

Java est un langage de programmation inspiré du langage C++, avec un modèle de programmation orienté objet, développé par Sun Microsystems. Il permet de créer des logiciels compatibles avec de nombreux systèmes d'exploitation (Windows, Linux

Macintosh, Solaris). Java donne aussi la possibilité de développer des programmes pour téléphones portables et assistants personnels. Enfin, ce langage peut être utilisé sur internet pour des petites applications intégrées à la page web (applet) ou encore comme langage serveur.

5. Présentation de l'application

Pour que notre application réussisse, il est important de satisfaire le maxi mum des besoins des utilisateurs, tel que :

➢ Formulation des demandes et des exigences par l'utilisateur : cette option permettra à l'utilisateur de formuler ses demandes et ses exigences surtout en termes de minimisation du temps de calcul.

➢ Exécution du projet : lors de l'exécution de notre application, les demandes de l'utilisateur pourront être satisfaites avec les meilleurs résultats possibles dans les meilleurs délais.

➢ Traitement des problèmes : si un problème survient lors de l'exécution, cela veut dire que les données fournies par l'utilisateur sont erronées ou bien mal introduites, et ce dernier devra modifier ses entrées**.**

a. Interface d'accueil

Comme son nom l'indique, elle constitue le point d'entrée de notre application (voir la figure 14). Elle permet d'avoir un accès à l'application. Elle est composée d'un menu qui contient les fonctionnalités de notre système. De cette fenêtre on est capable d'exécuter l'ordonnancement des taches indépendantes, aussi à partir de cette fenêtre en peut lancer un algorithme pour l'ordonnancement des données (le placement de données).

Figure 14 . Fenêtre d'accueil.

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

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

L'utilisateur peut importer ses données à partir d'un fichier existant (parcourir un fichier). Dans cette interface on a un menu qui permet l'accès aux différents algorithmes à exécuter à savoir l'algorithme Min-Min, LBE, SLB.

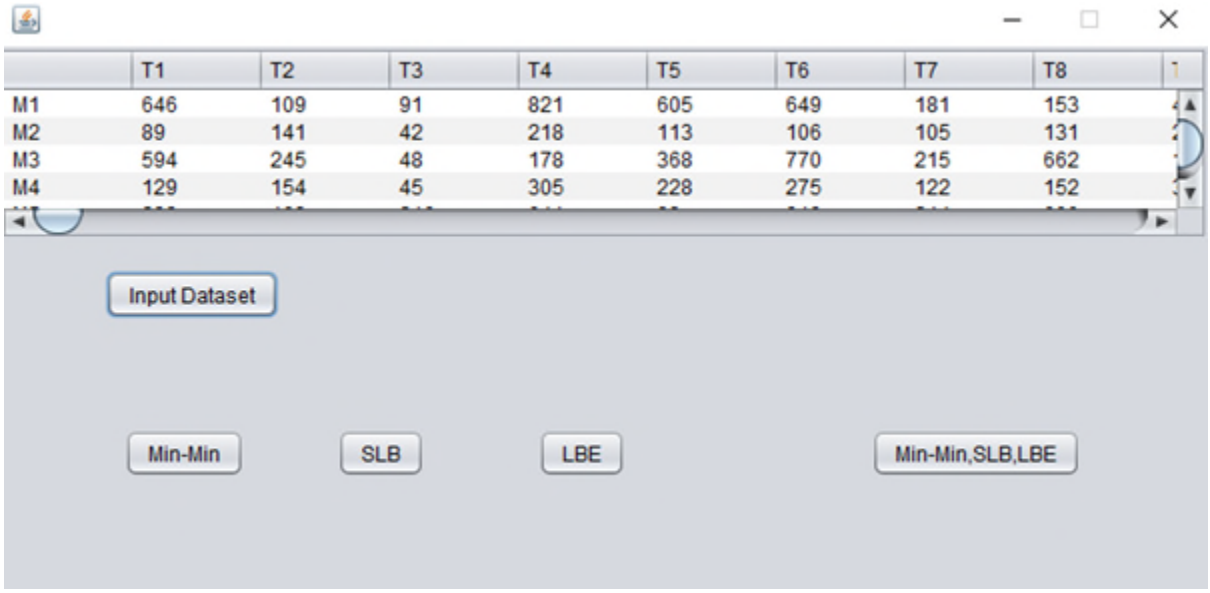

Figure 15 . Fenêtre de saisie.

c. La page d'exécution,

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

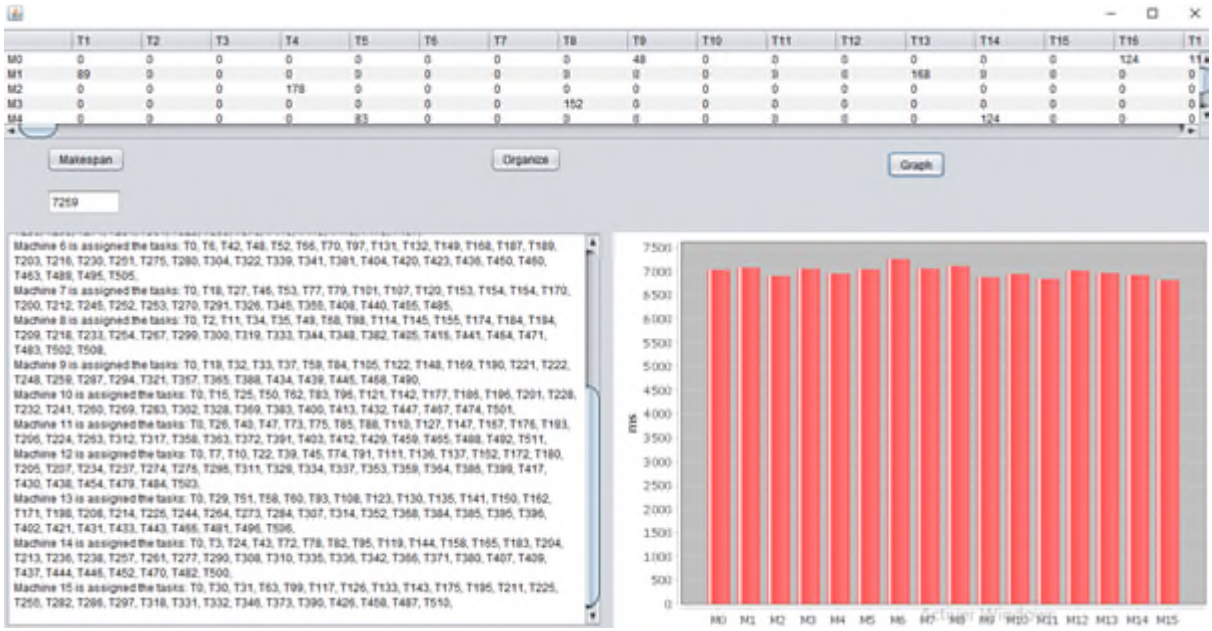

Figure 16 . Fenêtre d'exécution de l'algorithme Min-Min.

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

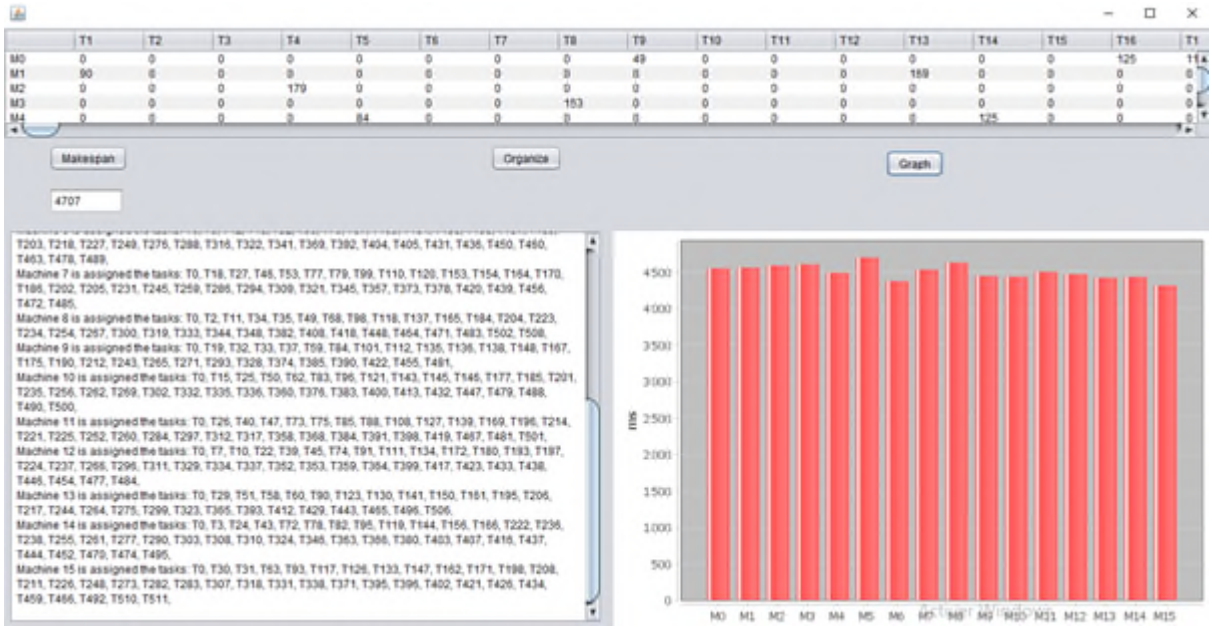

Figure 17. Fenêtre d'exécution de l'algorithme SLB.

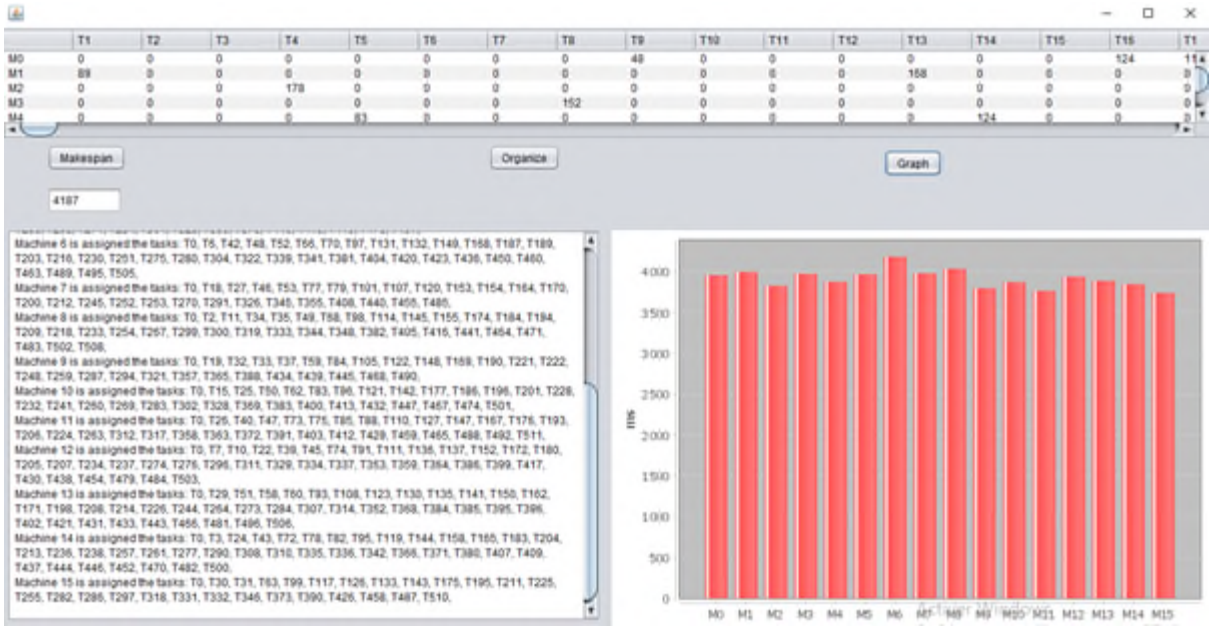

Figure 18 . Fenêtre d'exécution de l'algorithme LBE.

La figure 18 représente la comparaison entre les différents algorithmes dans le cas de l'ordonnancement des taches.

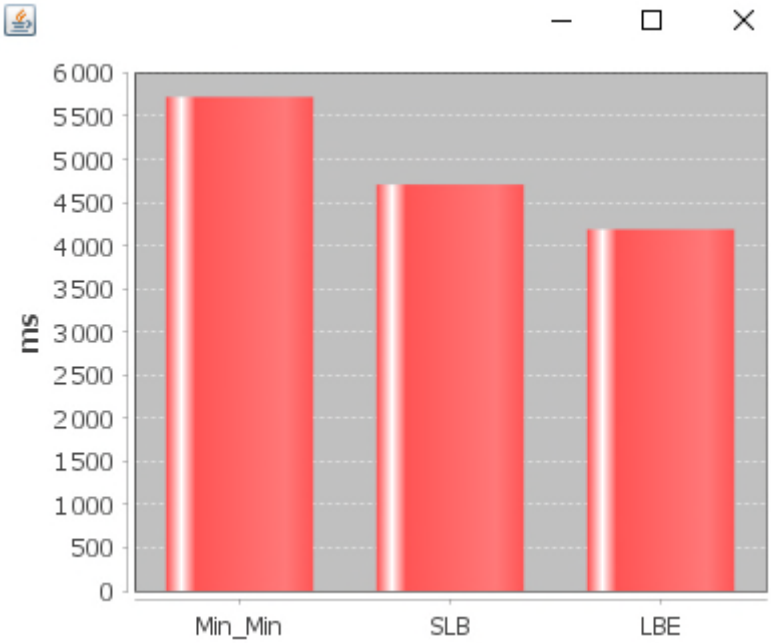

Figure 19. Fenêtre d'exécution de Min-Min, SLB, LBE.

6. évaluation des algorithmes proposés

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

7. Conclusion

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

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

Conclusions générales et perspective

Dans notre projet nous avons choisi trois algorithmes parmi les algorithmes autres Min-Min ,SLB ,LBE pour simulé et modéliser le problème dans le cas l'équilibrages de charge D'après les tests effectués, l'algorithme SLB minimise le temps total d'exécution, alors que l'algorithme LBE permet d'approcher au mieux la solution optimale. Ensuite, nous avons abordés les différentes fonctionnalités que propose notre application en illustrant ceci avec des explications textuelles et les captures d'écrans de l'application.
Les résultats obtenus montrent que les méthodes SLB et LBE donnent des bons résultats en termes de temps d'exécution global (makespan).

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

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

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

Bibliographie

Bibliographie

[01]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).

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

[03]https://waytolearnx.com/2018/07/difference-entre-ordonnancement-preemptif-etnon-preemptif.htmlconsulte le : 15 05-2020

[04] 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).

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

[06] A.A.Y. Elwessabi. Une approche basée agent mobile pour le cloudcomputing. Mémoire de magister, Université HADJ LAKHDAR - BATNA, 2014.

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

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

[09] Ahmed Gara-Ali. Ordonnancement de tâches et de périodes d'indisponibilité de durée variable. Autre. Université Grenoble Alpes, 2016. Français. ffNNT : 2016GREAI027ff. fftel-01492812f.

[10]https://waytolearnx.com/2018/07/difference-entre-ordonnancement-preemptif-etnon-preemptif.html consulte le : 15-05-202

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

[12] J. Carlier, A. Moukrim ; Problèmes d'ordonnancement à contraintes de ressources et applications ; HeuDiaSyC UMR CNRS 6599 UTC, Compiègne.

[13] A. BEN HMIDA SAKLY ; Méthodes arborescentes pour la résolution de problèmes d'ordonnancement flexible ; thèsedoctorat ; 2009 ; Université De Toulouse.

[14] Y. Choon Lee; Problem-centric scheduling for heterogeneous computing systems; thèsedoctorat; 2007; université de sydney

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

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

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

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

[19] 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.

[20] I. Foster, "What is the Grid?" GRIDSTART Technical Newsletter, Mars 2003

[21] Jesus Israel Hernandez Reactive Scheduling of DAG Applications on Heterogeneous and Dynamic Distributed Computing Systems Institute of Computing Systems Architecture School of InformaticsUniversity of Edinburgh 2008.

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

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

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

[25] Jean. François Pillou ,LoadBalancing (équilibrage de charge) , Mai 2008.

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

[27] H. Renard ; Équilibrage de charge et redistribution de données sur plates-formes hétérogènes ; THÈSE doctorat ; 2005 ; ÉCOLE NORMALE SUPÉRIEURE DE LYON.

[28] G. Cybenko; Dynamic load balancing for distributed memory multiprocessors;J. Parallel Distrib.Comput. 7, numéro2 ; 1989.

[29] S. Amédée, R. Francois-Gérard ; Algorithmes Genetiques ; TE de fin d'année ; 2004

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

[31] Y. Choon Lee; Problem-centric scheduling for heterogeneous computing systems; thèsedoctorat; 2007; université de sydney

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

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

[34]https://www.larousse.fr/encyclopedie/divers/grille_de_calcul/56663