Revue-IRS

Revue Internationale de la Recherche Scientifique
(Revue-IRS)
ISSN: 2958-8413
Vol. 3, No. 6, Décembre 2025

Problème d'affectation de tâches et équilibrage de charge dans un environnement multi-file d'attente

NGINDU MULUMBA Pascal[1], NTUMBA BADIBANGA Simon[2], KASENGEDI MUTOMBE Pierre[3], KANKU MUBENGA BANTU Eddy Michel[4], KALONJI SHIKAYI Sylvain[5].

1. *Chef de Travaux à Institut supérieur de Techniques Médicales de Kananga (ISTM)*
2. *Professeur Ordinaire à Université de Kinshasa (UNIKIN)*
3. *Professeur à Institut Supérieur des Techniques Appliquées (ISTA)*
4. *Chef de Travaux à l'Institut Supérieur de développement Rural (ISDR-Tshibashi)*
5. *Assistant à l'Académie militaire de Kananga (ACAMIL)*
En République Démocratique du Congo (RDC).

Abstract: The issue of task allocation and load management in a multi-queue framework aims to optimize the distribution of tasks among multiple servers or computing resources, each having its own queue. The main objective is to reduce waiting times and response times, to avoid overloading certain servers to prevent congestion and improve the overall use of resources, while considering the movement of nodes and communication constraints. This problem is particularly relevant in distributed contexts such as cloud computing, ad hoc networks, or parallel computing systems. It requires the development of clever decision-making mechanisms that can take into account the state of the queues, the capacities of the resources, and the specificities of the tasks. The suggested options often rely on queue management models, optimization algorithms, heuristics, or learning-based methods, with the aim of ensuring superior service quality, improved efficiency, and better system adaptability. This issue is of crucial importance for the efficiency and quality of services provided by contemporary distributed systems. Dynamic load balancing techniques, such as the improved Round-Robin algorithm, Least-Loaded First, or machine learning-based approaches, have all demonstrated their limitations regarding task allocation and the management of multiple queues. The development of a new method for task distribution and load balancing, as well as the incorporation of these three paradigms (Cloud computing, ad-hoc network, web service) into an unprecedented architectural structure, represents a significant contribution to this challenge.

Keywords: *Task allocation, Load balancing, Multi-queueing, Optimization, Cloud computing.*

Résumé: La question de l'attribution des tâches et de la gestion de la charge dans un cadre multi-file d'attente vise à optimiser la distribution des tâches entre plusieurs serveurs ou ressources informatiques, chacun ayant sa propre file d'attente. L'objectif principal est de réduire les délais d'attente et les temps de réponse, d'éviter la surcharge de certains serveurs pour prévenir l'engorgement et améliorer l'utilisation globale des ressources, tout en considérant le mouvement des nœuds et les contraintes en matière de communication. Ce problème est particulièrement pertinent dans les contextes distribués tels que le cloud computing, les réseaux ad hoc, ou encore les systèmes informatiques parallèles. Il requiert l'élaboration de dispositifs décisionnels astucieux qui peuvent prendre en compte l'état des files d'attente, les capacités des ressources et les spécificités des missions. Les options suggérées reposent souvent sur des modèles de gestion des files d'attente, des algorithmes d'optimisation, des heuristiques ou des méthodes fondées sur l'apprentissage, dans le but de garantir un service de qualité supérieure, une efficacité améliorée et une meilleure capacité d'adaptation du système. Cette problématique est d'une importance cruciale pour l'efficacité et la qualité des prestations des systèmes distribués contemporains. Des techniques d'équilibrage de charge dynamique, comme l'algorithme Round-Robin amélioré, le Least-Loaded First ou des approches basées sur l'apprentissage automatique, toutes ces techniques

ont démontré leurs limites en ce qui concerne l'attribution de tâches et la gestion des files d'attente multiples. Le développement d'une nouvelle méthode de répartition des tâches et d'équilibrage de la charge, ainsi que l'incorporation de ces trois paradigmes (Cloud computing, réseau ad-hoc, service web) dans une structure architecturale inédite constituent une contribution significative à ce challenge.

Mots-clés : *Affectation de tâches, Équilibrage de charge, Multi-file d'attente, Optimisation, Cloud computing.*

Digital Object Identifier (DOI): https://doi.org/10.5281/zenodo.**18049291**

1. Introduction

1.1. Contexte de la recherche et problématique

Dans des contextes informatiques actuels comme le cloud computing, les réseaux mobiles ad hoc et les centres de traitement décentralisés, l'administration efficace des tâches représente un défi stratégique crucial. L'un des enjeux les plus importants est la question de l'attribution des tâches liée à l'équilibrage de charge dans un système avec plusieurs files d'attente. Ces dispositifs se composent de diverses files d'attente, chaque file étant liée à un ou plusieurs serveurs ou unités de traitement. L'efficacité globale est fortement influencée par la façon dont les tâches sont distribuées entre ces files.

Dans un contexte comme celui-ci, une mauvaise distribution des tâches peut provoquer l'engorgement de certaines files d'attente, pendant que d'autres sont à peine exploitées, ce qui amplifie le délai moyen de réponse, diminue l'efficience des ressources et met en péril la qualité du service (QoS). Pour répondre à ce défi, on utilise des méthodes d'attribution dynamique et des algorithmes de répartition de charge.

L'objectif est de développer un algorithme d'affectation et équilibrage de charge et paralléliser de tâches de multiple file d'attente sur n serveurs, pour réduire le temps d'attente moyen, d'optimiser le taux d'exploitation des ressources et de garantir une distribution équilibrée du travail.

1.2. Problématique et hypothèse

Les approches suggérées dans les écrits diffèrent en fonction des présupposés : centralisation ou décentralisation, caractère statique ou dynamique du système, uniformité ou diversité des serveurs, etc. Les méthodes traditionnelles comme Round-Robin, Least-Loaded First ou celles basées sur la théorie des files d'attente sont fréquemment associées à des techniques contemporaines telles que l'apprentissage automatique, la programmation linéaire ou encore les approches heuristiques et métaheuristiques on montrer leur limitent. Et à ce jour, l'intégration de plusieurs techniques pour de méthodes hybrides saveurs indispensable.

Par conséquent, l'examen de la question de l'attribution des tâches et de la répartition de la charge dans un cadre multi-file d'attente représente un champ d'études dynamique et crucial pour assurer la capacité d'évolution, la fiabilité et l'efficacité des infrastructures distribuées contemporaines.

La question fondamentale est celle de savoir : Comment gérer les conflits des requêtes dans ce système distribué et maintenir le système en équilibre ?

A cette problématique, vu le limite constaté dans l'utilisation des méthodes traditionnelles de gestion de conflits et répartition de charge, nous suggérons l'hypothèse selon laquelle la conception d'un algorithme utilisant une nouvelle approche d'affectation et ordonnancement de tâches (requêtes) sur plusieurs serveurs en parallèle pourrait être une solution pour éviter la congestion, les explosions et la surcharge pour maintenir le système en équilibre.

2. Système multi-file d'attente

Aujourd'hui, les files d'attente se présentent sous diverses formes et dans de nombreux domaines, devenant ainsi un phénomène courant que l'on rencontre régulièrement. Voici quelques illustrations parmi une multitude d'autres : l'informatique en nuage, les réseaux mobiles ad hoc, les centres de traitement décentralisés, l'engorgement à un point de service, la congestion du trafic routier ou d'un réseau de télécommunication, la gestion d'un inventaire de production, l'entretien d'un matériel informatique, le déplacement de populations, les prévisions météorologiques, et ainsi de suite. Il existe une multitude de modèles stochastiques, basés sur des hypothèses ajustées au contexte spécifique. Nous allons examiner le modèle M/M/c, considéré comme une spécificité du processus stochastique, et qui s'adapte au système que nous souhaitons représenter dans une file d'attente multiple.

2.1. Le Modèle M/M/c

Erlang propose le modèle M/M/c qui est un modèle de file d'attente qui décrit un système avec c serveurs (ou canaux) où les appels arrivent selon un processus de Poisson (arrivées aléatoires avec un taux constant) et la

durée des appels suit une distribution exponentielle (temps de service aléatoire mais avec un taux constant). C'est un modèle très courant dans les centres d'appel et d'autres systèmes de service en file d'attente.

Le modèle M/M/1 et M/M/c ont la même logique de base sur les mêmes hypothèses fondamentales :

- ✓ M (premier M) : Arrivées selon un processus de Poisson (aléatoires) avec un taux λ,
- ✓ M (deuxième M) : Durée de service exponentielle, avec un taux μ /1 ou /c : nombre de serveurs (1 ou c)

Donc : M/M/1 = 1 seul serveur et M/M/c = c serveurs en parallèle

On choisit M/M/c à la place de M/M/1 quand le volume de tâches augmente, on veut réduire les temps d'attente et que le système dispose de plusieurs serveurs ; c'est le contexte de notre étude car nous analysons un système à multi file d'attente. Le modèle M/M/1 est un cas particulier de M/M/c avec c = 1.

2.2. Caractéristiques

Voici quelque caractéristique du M/M/C

1. Modèle réaliste pour plusieurs serveurs
   - Le modèle M/M/c est adapté aux situations avec plusieurs serveurs**.**
   - C'est plus réaliste que le M/M/1 quand plusieurs agents traitent les demandes en même temps (guichets, caisses, serveurs informatiques, centre téléphonique…).
2. Hypothèses simples et courantes
   - Arrivées aléatoires (processus de Poisson) → modélise bien les flux de clients non prévisibles.
   - Durées de service exponentielles (aléatoires aussi) → valable pour beaucoup de services rapides.
   - Les serveurs sont identiques et indépendants → simplifie l'analyse mathématique.(Donald, G. John, Shortle,F. James, M. Carl M, H. 2018).
3. Équilibre entre précision et simplicité

   C'est un bon compromis entre un modèle simple (M/M/1) et des modèles plus complexes (M/G/c, G/G/c).
4. Mesures de performances accessibles. Avec M/M/c, on peut calculer : Temps moyen d'attente, Nombre moyen de clients, Taux d'occupation des serveurs (ρ), Probabilité qu'un client attende. Ces résultats aident à dimensionner un service (combien de serveurs faut-il pour garantir un bon service ?)
5. Utilisé dans de nombreux domaines

2.3. Éléments d'un système à files multiples

Le système d'attente à plusieurs files comprend les éléments suivants.

- ➢ Les clients : entités à considérer
- ➢ Les serveurs : ressources de prestation de service.
- ➢ Les queues : zones d'attente
- ➢ Frequence d'arrivée (λ) et fréquence de service (μ)
- ➢ Politique de gestion des files d'attente (FIFO ou FCFS, priorité, etc.)

2.4. Divers scénarios.

2.4.1. La probabilité d'occupation Po (système inoccupé) :

Dans le modèle M/M/c, la formule ci-dessous est utilisée pour calculer la probabilité Po que tous les serveurs soient disponibles (aucune tâche en traitement) :

$$Po = \left(\sum_{n=0}^{C-1} \frac{(E^n)}{n!}\right) / \left(\sum_{n=0}^{c} \frac{(E^n)}{n!}\right)$$

Où :

Le numérateur illustre l'addition des éléments de la série de n = 0 jusqu'à n = c−1.

Le dénominateur correspond à l'addition de tous les termes de la série, allant de n = 0 jusqu'à n = c, cela inclut également le terme où n = c.

2.4.2. Probabilité qu'une tâche soit en attente :

La probabilité $P_{attente}$ qu'une tâche soit en attente (c'est-à-dire que tous les serveurs sont occupés et que la tâche doit patienter) dans un système M/M/c est exprimée de la manière suivante :

$$P_{attente} = \frac{\frac{(E^c)}{n!} x \frac{(1)}{(1-\rho)}}{\sum_{n=0}^{c} \frac{(E^n)}{n!}}$$

Où : $\rho = \frac{\lambda}{c\mu}$ est le taux de trafic par serveur (la charge par serveur).

E est la charge totale du système, E= $\frac{\lambda}{\mu'}$=, comme mentionné précédemment.

2.4.3. Probabilité qu'une tâche soit rejetée (c'est-à-dire, que tous les serveurs soient occupés et qu'il n'y ait plus de places disponibles) :

La probabilité qu'une tâche soit rejetée, c'est-à-dire lorsque tous les serveurs sont occupés et qu'une tâche entrant est rejetée, peut être approximée par P_{rejet}, qui est simplement la probabilité que le système soit à pleine capacité (c'est-à-dire que c serveurs sont occupés).

$$P_{rejet} = \frac{\frac{(E^c)}{n!}}{\sum_{n=0}^{c} \frac{E^n}{n!}}$$

2.5. Récapitulatif des formules :

1. Probabilité que le système soit vide P_0 :

$$P_0 = \frac{\sum_{n=0}^{c-1} \frac{(E^n)}{n!}}{\sum_{n=0}^{c} \frac{(E^n)}{n!}}$$

2. Probabilité qu'une tâche soit mis en attente $P_{attente}$:

$$P_{attente} = \frac{\frac{(E^c)}{c!} x \frac{(1)}{(1-\rho)}}{\sum_{n=0}^{c} \frac{E^n}{n!}}$$

3. Probabilité de rejet (système saturé) P_{rejet} :

$$P_{rejet} = \frac{\frac{(E^c)}{n!}}{\sum_{n=0}^{c} \frac{E^n}{n!}}$$

Le modèle M/M/c est utilisé pour représenter un système d'attente comportant plusieurs serveurs. L'estimation des chances que les tâches soient refusées, en attente ou que le système soit vide se fait grâce aux équations précédemment citées. Ce modèle est particulièrement pertinent pour évaluer les performances d'un système disposant de plusieurs canaux de service.

3. Problème d'affectation et équilibrage de charge

L'affectation et l'équilibrage de charge visent ainsi à améliorer les performances globales d'un système, à réduire les temps de réponse, à optimiser l'utilisation des ressources disponibles et à garantir une meilleure tolérance aux pannes. Ces techniques sont particulièrement cruciales dans les environnements distribués, les centres de données, les systèmes de cloud computing et les architectures orientées services (Nadjet, K., Louiza, B. & Djamil 2023).

3.1. Objectifs de l'affectation de tâches

L'objectif principal de ce problème est d'optimiser la répartition des ressources. Cela peut inclure :

- Minimiser les coûts ;
- Maximiser l'efficacité ;

Respecter les contraintes : telles que la capacité de Modélisation du problème d'affectation de tâches

3.2. Types de problèmes d'affectation

- Problème d'attribution typique :
- Problème d'attribution avec des coûts variables : chaque ressource dispose d'un coût distinct pour accomplir une tâche spécifique,
- Problème avec restrictions : une mission peut être confiée à un ouvrier, mais sous certaines conditions de coûts ou de capacités.

Hypothèses sous-jacentes

- Le volume de tâches et de ressources,
- Les ressources ont la capacité de gérer les tâches effectuées (il n'y a pas de dépendances complexes entre elles),
- Les tâches sont autonomes et peuvent être exécutées en parallèle.

Plusieurs approches existent pour traiter le problème de l'attribution de tâches sur un serveur :

- Affectation classique (optimisation d'un critère unique) ;
- Affectation avec contraintes (répartition sous contraintes spécifiques) ;
- Affectation dynamique (en temps réel, ou selon la charge du système) ;
- Affectation à plusieurs critères (multi-objectifs) ;
- Affectation par politique de priorité (affectation avec priorités) ;
- Affectation de type "équilibrage de charge.

3.3. Méthodes de résolution du problème d'affectation de tâches (Brahim Mebrek, 2021).

3.3.1. Approche brute-force (exhaustive)

L'approche brute-force consiste à tester toutes les permutations possibles d'affectation entre les tâches et les ressources, et à choisir celle qui minimise la fonction objective. Cette méthode est extrêmement coûteuse en termes de temps de calcul, particulièrement lorsque le nombre de tâches et de ressources est élevé, car elle nécessite de tester même n solutions possibles :

3.3.2. Méthodes exactes

1. Algorithme de Kuhn-Munkres (Hongrois) : Cet algorithme est utilisé pour résoudre le problème d'affectation dans les cas de $O(n^3)$ Sur 3 pour un problème de n tâches et ressources. Il est basé sur un processus de réduction de matrice de cout ou de gains.
2. Programmes linéaires : Formulation du problème comme un programme linéaire (Yves Tabourier, 1972).

3.3.3. Méthodes heuristiques et approximatives

1. Algorithmes gloutons : Ces algorithmes prennent des décisions locales basées sur des critères immédiats
2. Algorithmes génétiques : Ce type d'algorithme de recherche évolutionnaire explore l'espace des solutions en générant des populations de solutions et en exécutant des mécanismes de sélection, croisement et mutation.
3. Algorithmes de recherche locale : Des méthodes comme la recherche tabou ou le recuit simulé permettent d'améliorer une ainsi la solution en ajustant localement les affectations. (Irène Charon & Olivier Hudry, 2019).

3.3.4. Méthodes d'optimisation combinatoire

Pour les grands problèmes, les approches d'optimisation combinatoire comme les méthodes de branchement et liées ou les algorithmes de dynamique de programmation peuvent être utilisés pour trouver des solutions optimales.

3.3.5. Approches par intelligence artificielle

1. Apprentissage par renforcement : Utilisation d'un agent intelligent pour apprendre à résoudre le problème d'affectation au fur et à mesure de ses interactions avec l'environnement.
2. Optimisation par essais particulaires (PSO) : Utilisation de groupes de "particules" qui explorent l'espace de solution et ajustent leur position pour trouver de solution.

3.3.6. Approche hybride

Cette approche permet de tirer parti des forces de plusieurs stratégies de planification et affectation tout en compensant leurs faiblesses.

À travers une analyse critique des méthodes classiques, notamment l'algorithme hongrois et les techniques d'ordonnancement statiques, nous avons mis en évidence leurs limites face aux exigences des systèmes modernes, caractérisés par leur complexité, leur dynamique et leur hétérogénéité.

4. Nouvel méthode d'affectation et équilibrage de tâche

Face à certaines contraintes des méthodes traditionnelles et aux exigences changeantes des systèmes contemporains, les algorithmes tels que hongrois ou les files d'attente fondées sur le round-robin ne tiennent pas systématiquement compte de la dynamique du système (utilisation du processeur, mémoire, réseau, etc.).

Ils sont efficaces en théorie, mais manquent de flexibilité dans des environnements variés ou en changement rapide.

Voici les motifs principaux de l'adoption de cette nouvelle méthode :

Nécessité d'une meilleure répartition de la charge (équilibrage de charge) : Dans un contexte multi-serveurs, une affectation incorrecte peut entraîner :

- ✓ Une surcharge de certains serveurs,
- ✓ Une exploitation insuffisante d'autres serveurs,
- ✓ Un déclin général de la performance.

Un bon équilibre de charge optimise :

- ✓ La latence,
- ✓ Les délais de réponse,
- ✓ L'espérance de vie des ressources (réduction de la surchauffe ou de l'usure)

1. **Scalabilité et complexité croissante**

Avec la montée en puissance de l'informatique en nuage, de l'informatique périphérique et des centres de données à grande échelle, il y a une explosion du nombre de serveurs et de tâches. Les approches classiques ne s'adaptent pas aisément à l'échelle, d'où la nécessité de solutions plus performantes et évolutives.

2. **Adaptation à la diversité**

Les serveurs peuvent présenter les caractéristiques suivantes :

- Des spécifications variées (Serveur, CPU, RAM, GPU, etc.),
- Des priorités distinctes de traitement,
- Des politiques de sécurité ou des régions géographiques spécifiques.

Les procédures conventionnelles présument fréquemment que tous les serveurs sont identiques (une hypothèse peu réaliste). Ces contraintes hétérogènes peuvent être incorporées dans une nouvelle technique.

4.1. **Environnements en temps réel et de nature dynamique**

Les missions peuvent se présenter de manière constante, possédant diverses caractéristiques (durée, priorité, dépendances...).

Nous avons besoin d'algorithmes :

- Rapides (capacité de prendre des décisions rapidement),
- Résistants à des modifications inattendues (pannes, pics de fréquentation…),
- Mieux encore, parfois autonomes grâce à l'apprentissage machine.

4.2. Optimisation à plusieurs objectifs

Dans les systèmes contemporains, il ne suffit pas simplement de réduire le temps de traitement, mais également de :

- Diminuons la consommation d'énergie,
- Réduire les frais de migration ou de communication,
- Respecter les exigences en matière de sécurité ou de priorité.

Les techniques traditionnelles ont fréquemment un seul objectif.

4.3. Modélisation de l'affectation et équilibrage de tâches

Cette nouvelle méthode présuppose que : Nous avons à notre disposition un certain nombre de serveurs de traitement, chargés d'exécuter une fonction d'attribution sur diverses files d'attente. Il suffit d'attribuer à chaque appareil une tâche à accomplir en fonction de sa capacité de traitement.

4.3.1. Modèle de distribution contenu :

Quelle est le nombre de tâches *ni* à affecter à chaque processeur pour que le traitement se termine au même moment ?

a. Description de variables

- ✓ n_i = nombre des tâches à confier aux machine de catégorie *i* ;
- ✓ N_i = nombre de machines de la catégorie *i* ;
- ✓ N = nombre total de tâches ;
- ✓ v_i = vitesse des machines de catégorie *i* ;
- ✓ n = nombre des catégories de machines.

On cherche le nombre n_i de taches à confier à chaque machine de catégorie *i ;*

On doit alors satisfaire :

$$n_i * N_i + n_{i+1} * N_{i+1} = N$$

$$n_i = \frac{v_i \,.\, N}{v_i * N_i + v_{i+1} * N_{i+1} \ldots} =$$

Ces relation se généralisent facilement à

$$\boldsymbol{n_i = \frac{v_i \,.N}{\sum_{i=1}^{n} v_i \,.N_i}}$$

Chaque machine de la catégorie i, exécutera chacune le nombre de tâche n_i

Ainsi toutes les machines termineront leurs tâches en même temps si on leur confie à chacune ce nombre n_i de tâche. Comme ces nombres sont fractionnaires certains des machines de la catégorie *i* recevront plus de tâches à réalités, que les autres et cela selon leur catégorie.

On voit que ce modèle donne une approximation du nombre de calculs à confier à chaque processeur, mais ne donne pas le nombre optimal.

4.3.2. Modèle de distribution discret :

Partant de la distribution contenu, le nombre de tâche est affecter en fonction de la vitesse du serveur de catégorie i, cela fait que certaine machine reçoivent plus de tâche que les autres. La nouvelle idée est de confier une tâche à un processeur de catégorie i en fonction de sa vitesse, comme l'illustre la figure ci-après :

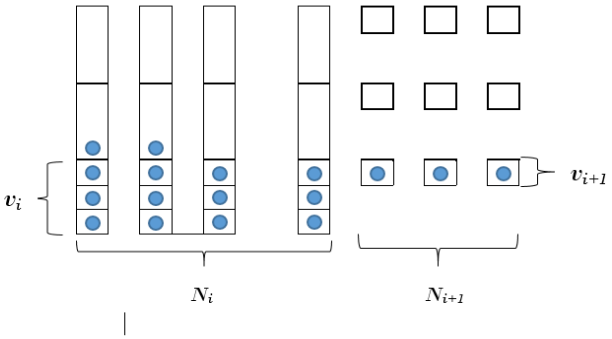

Figure 1 : Modèle de distribution discret

L'algorithme sera donc exécuter sous ces étapes :

1. Le processeur doivent être classés par ordre décroissant de rapidité (vitesse) ;
2. Répartir le tâches une par une :
    1. En commençant par la gauche
    2. En se décalent d'une case vers la droit si $n_i / v_i \geq (n_{i+1} + 1) / v_{i+1}$
    3. On revient à la première case si la condition ci-haut n'est pas vérifiée ou si on arrive tout à droite.

5. Algorithme d'affectation contenu et équilibrage de taches dans un système distribué.

```
Début de l'algorithme
  Ecrire "Entrer le nombre d'éléments n :"
  Lire n
  Dimensionner v[n], N_i [n], n_i [n]
   Ecrire "Entrer la valeur de N (constante) :"
  Lire N
  somme ← 0
  i ← 1
  TANTQUE i ≤ n
    Ecrire "Entrer v[",i, "] :"
    Lire v[i]
    Ecrire "Entrer N_ i[",i, "] :"
    Lire N_i[i]
    i ← i + 1
  FINTANTQUE
  i ← 1
  TANTQUE i ≤ n
    somme ← somme + v[i] * N_i[i]
    i= 1 + 1
  FINTANTQUE
   i ← 1
  TANTQUE i ≤ n
    n_i [i] = (v[i] * N) / somme
    i = 1 + 1
  FINTANTQUE
Fin de l'algorithme
```

Figure 2 : algorithme d'affectation et équilibrage de charge (notre contribution)

Cet algorithme permet de calculer et déterminer le nombre de tâche à confier à chaque serveur placé en parallèle tout en équilibrant et optimisant la gestion des capacités de traitement des serveurs disponibles. En utilisant une approche basée sur la gestion de multiple file et l'optimisation dynamique, il est possible de minimiser efficacement le temps d'attente total des tâches dans un environnement de traitement parallèle, réduire la surcharge et éviter de explosion de système.

5.1. Complexités de l'Algorithme

1. Complexité temporelle
   - Étape 1 (somme des vitesses) : O(n)
   - Étape 2 (calcul des tâches) : O(n)

Il s'agit de deux boucles linéaires indépendantes, donc la complexité globale est : O(n)

2. Complexité spatiale

Le tableau tâches [1..n] nécessite un espace proportionnel au nombre de serveurs : O(n).

3. Complexité totale est :
   - Complexité temporelle : O(n) — linéaire avec le nombre de serveurs.
   - Complexité spatiale : O(n) — on stocke une tâche par serveur.

C'est un algorithme très efficace, adapté à la répartition rapide même sur un grand nombre de serveurs

5. Application en python

```python
# Début de l'algorithme en Python

# Entrée du nombre d'éléments
n = int(input("Entrer le nombre d'éléments n : "))

# Déclaration des tableaux
v = [0] * n
N_i = [0] * n
n_i = [0] * n

# Lecture de la constante N
N = float(input("Entrer la valeur de N (constante) : "))

# Lecture des valeurs v[i] et N_i[i]
for i in range(n):
    v[i] = float(input(f"Entrer v[{i+1}] : "))
    N_i[i] = float(input(f"Entrer N_i[{i+1}] : "))

# Calcul de la somme
somme = 0
for i in range(n):
    somme += v[i] * N_i[i]
```

```python
# Calcul des n_i[i]
for i in range(n):
    n_i[i] = (v[i] * N) / somme

# Affichage des résultats
print("\n===== Résultats =====")
print("Somme :", somme)
for i in range(n):
    print(f"n_i[{i+1}] = {n_i[i]}")
```

Figure 3 : algorithme de répartition de charge en Python

5.1. Test de l'algorithme avec les vraies données

```
Entrer le nombre d'éléments n : 3
Entrer la valeur de N (constante) : 100
Entrer v[1] : 2
Entrer N_i[1] : 5
Entrer v[2] : 3
Entrer N_i[2] : 4
Entrer v[3] : 1
Entrer N_i[3] : 6

===== Résultats =====
Somme : 28.0
n_i[1] = 7.142857142857142
n_i[2] = 10.714285714285714
n_i[3] = 3.571428571428571
```

Figure 4 : Test de l'algorithme et résultat.

Nous avons codé cette algorithme avec python, et faisant le jeu d'essai avec les données réelle prisent au hasard pour voir son exécution ainsi que les résultats. A partir de ce résultat, les affectations s'effectuent selon le modèle contenu ou discret ci-haut, selon chaque catégorie.

Conclusion

L'attribution efficace des tâches et l'équilibrage de la charge dans un cadre de files d'attente multiples constituent toujours un défi majeur pour les systèmes informatiques contemporains, surtout dans les situations de cloud computing, de réseaux ad hoc ou de services en temps réel. La méthode innovante suggérée dans ce document se démarque par sa flexibilité à s'ajuster en temps réel aux changements de charge, à la diversité des serveurs et aux variations des requêtes des utilisateurs.

Cette méthode, qui fusionne des techniques basées sur la théorie des files d'attente avec des algorithmes sophistiqués (comme l'apprentissage machine, l'optimisation heuristique ou la prise de décision décentralisée), facilite une distribution plus juste des tâches, un abaissement notable du temps d'attente moyen et une exploitation plus homogène des ressources à disposition. Elle dépasse les techniques traditionnelles en considérant la condition générale du système et en prévoyant l'élément fragile qui entrave le flux et diminue la performance du système.

Par conséquent, cette technique innovante représente un progrès majeur dans la gestion des soucis de congestion des serveurs et de détérioration de l'efficacité. Elle pave aussi la voie pour des évolutions futures en incorporant des mécanismes de prévision, d'apprentissage autonome ou d'autonomie dans des contextes dynamiques tels que les réseaux mobiles, les infrastructures cloud hybrides ou les systèmes critiques en temps réel.

REFERENCES

[1]. BARDET, S., et Blanc, L., *Les Services Web*, Ed. Eyrolle, Paris, 2003.
[2]. BAYNAT, B., *Théorie de la file d'attente* ; Ed. Eyrolles, Paris, 1970,
[3]. Brahim Mebrek. (2021). *Modèles d'affectation avec capacités de ressources*, Université d'Annaba, 27.
[4]. Coordonnateurs divers : Théorie des files d'attente 2, ISTE Editions, Paris, 2021.
[5]. Donald, G. John, Shortle,F. James, M. Carl M, H. (2018). *Fundamentals of Queueing Theory (5e édition)*. Éd. John Wiley & Sons, USA. 90-93.
[6]. Harchol-Balter, M. (2013). *Performance Modeling and Design of Computer Systems: Queueing Theory in Action.* Cambridge University Press.
[7]. Charon, I. & Hudry, O. (2019). Introduction à l'optimisation continue et discrète, Édition Lavoisier, Cachan, 339-374.
[8]. Nadjet, K., Louiza, B. & Djamil. *Équilibrage de charge pour l'amélioration des performances : dans les réseaux de capteurs sans fil,* Éditions Universitaires, Bruxelles, 2023.
[9]. Ndungu, M. N. *Système de gestion des files d'attente virtuelles,* Editions Notre Savoir, Paris, 2023
[10]. Philippe, R. (2000). *Réseaux et files d'attente : méthodes probabilistes*, Springer, 2e éd., Paris, 101-104.
[11]. Philippe, R. (2003). *Réseaux stochastiques et files d'attente* (édition française). Springer, Berlin/Heidelberg.
[12]. STALLINGS, W. « Queuing Analysis (A Practical Guide for Computer Scientists) », 2000.
[13]. Toky, B. R. & Falimanana, R. *Des files d'attente au télétrafic*, ISTE Editions, Paris 2023
[14]. Yves Tabourier, (1972), *Un algorithme pour le problème d'affectation*, Recherche Opérationnelle, Vol. 6, No. V3 pp. 3-15.
[15]. Oliveira, A. L. (2025). *Évaluation des stratégies d'équilibrage de charge : stratégies maître-esclave pour les clusters et les grilles informatiques*, Éditions Notre Savoir. Paris.