**Table des Matiere**

1. Introduction

2. Modélisation du système

2.1 - Description générale

2.2 - Hypothèses et distributions

2.3 - Paramètres et contraintes

3. Méthodologie

3.1 - Outils utilisés

3.2 - Fonctionnement de la simulation

3.3 - Expériences réalisées

4. Résultats expérimentaux

4.1 - Temps de réponse moyen

4.2 - Taux de perte

5. Conclusion

6. Annexes

**Introduction :**

Dans notre projet de simulation de cette année universitaire 2024-2025, nous observerons le fonctionnement d’une ferme de serveurs capable de travailler sur les requêtes informatiques. L’objectif principal est de développer et d’améliorer l’idée de l’organisation spatiale des serveurs de manière à minimiser le temps que le système passe à traiter les requêtes formulées tout en gardant un minimum de pertes à cause de la surcharge. Le système consiste en 12 serveurs indiscernables. Ces serveurs peuvent être regroupés en C groupes spécialisés, avec C ∈ {1, 2, 3, 6}. Chaque groupe peut traiter une catégorie spécifique de requêtes plus rapidement, mais cette spécialisation introduit un coût de routage et nécessite une coordination stricte.

Afin de modéliser ce système, nous avons recours à une simulation événementielle en Python. Cette approche permet de représenter dynamiquement l’arrivée des requêtes, leur traitement par un routeur central, et leur affectation aux serveurs disponibles. Le comportement du système dépend de deux paramètres principaux :

* λ, le taux d’arrivée des requêtes, qui suit une loi exponentielle.
* C, le nombre de groupes spécialisés dans lesquels les serveurs sont répartis.

Nous avons pour objectif d’évaluer :

* Le **temps moyen** de réponse des requêtes.
* Le **taux de perte**, qui doit rester inférieur à 5%.
* Le choix de C en fonction de différentes valeurs de λ.

Pour ce faire, nous commencerons par la modélisation du système, suivie de la méthodologie que nous avons utilisée pour cette simulation. Ensuite, nous comparerons les graphiques des résultats pour le système dans chaque configuration, et enfin, nous conclurons le rapport par l’option optimale en fonction de la charge.

**2) Modélisation du système**

**2.1. Description générale**

Le système que nous modélisons est une **ferme de serveurs composée de 12 machines** identiques, qui ont pour but de traiter des requêtes entrantes de manière rapide et efficace. Pour optimiser les performances du système, il est possible de **répartir les serveurs en groupes spécialisés** selon les types de requêtes à traiter.

Le nombre de groupes, noté **C**, peut prendre les valeurs 1, 2, 3 ou 6. Chaque groupe contient alors **K = 12 / C** serveurs. Lorsque **C = 1**, tous les serveurs sont généralistes. À l’inverse, plus C est élevé, plus les groupes sont spécialisés dans une catégorie précise.

Un **routeur central** reçoit l'ensemble des requêtes et les achemine vers le groupe spécialisé approprié. Il agit comme un goulot d’étranglement : il a une **file d’attente limitée**, traite les requêtes en **ordre FIFO**, et peut être **bloqué** si aucun serveur du groupe de destination n’est disponible.

**2.2 - Hypothèses et distributions**

Pour simuler ce système, nous avons adopté les hypothèses probabilistes suivantes :

* **Arrivée des requêtes**
  + Les **inter-arrivées** suivent une **loi exponentielle de paramètre λ**.
  + Chaque requête est assignée à une **catégorie choisie uniformément** parmi les C catégories disponibles.
* **Comportement du routeur :**
  + La **file d’attente du routeur** a une capacité maximale de **100 requêtes**, incluant celle en cours de traitement.
  + Les requêtes sont traitées en **FIFO**.
  + Le **temps de traitement du routeur** dépend du nombre de groupes C :
  + Si la file est pleine à l’arrivée d’une requête, **celle-ci est perdue**.
  + Après traitement, la requête est dirigée vers un **serveur libre** du groupe correspondant.
  + Si aucun serveur du bon groupe n’est disponible, le **routeur est bloqué** jusqu’à libération d’un serveur.
* **Serveurs**
  + Un serveur ne peut traiter qu’une seule requête à la fois et n’a pas de file d’attente.
  + Le temps de service d’un serveur suit une loi exponentielle de paramètre dépendant de C :
  + Les serveurs spécialisés ont un taux de service plus rapide (le traitement est accéléré par un facteur configuré).

**2.3 - Paramètres et contraintes**

La simulation dépend des **paramètres suivants** :

* + **λ (lambda)** : taux d’arrivée des requêtes (ex : de 0.2 à 1.8).
  + **C** : nombre de groupes spécialisés (1, 2, 3, 6).
  + **Durée maximale de simulation** : ex. 1000 unités de temps.
  + **Facteur d'accélération** des serveurs spécialisés (ex. 1.5).

**Contraintes du système :**

* Le **taux de perte** des requêtes **ne doit pas dépasser 5%**.
* Le **temps de réponse** doit être minimisé.
* Une **seule file d’attente (dans le routeur)** est autorisée.
* Les serveurs ne doivent pas introduire de file d’attente individuelle.

**3. Méthodologie**

**3.1 - Outils et langage utilisés**

Les modules principaux sont :

* *random :* génération aléatoire (loi exponentielle).
* *collections.deque :* gestion de la file FIFO du routeur.
* *enum :* définition des types d’événements.
* *math :* calculs numériques (exponentielles, logarithmes).

Le code est modulaire, avec une classe dédiée pour chaque composant : événements, requêtes, routeur, serveurs, groupes de serveurs…  
Une **approche événementielle** a été adoptée afin de simuler le système de manière efficace et contrôlée dans le temps.

**3.2 - Architecture de la simulation**

Le cœur de la simulation repose sur un **échéancier**, qui est une liste d’événements ordonnés dans le temps. Chaque événement correspond à l’un des états suivants :

* Arrivée d’une requête,
* Fin de traitement par le routeur (routeur devient disponible),
* Tentative d’envoi vers un serveur,
* Blocage du routeur (aucun serveur libre).

À chaque étape :

1. L’événement le plus ancien est extrait.
2. Le système est mis à jour en conséquence.
3. De nouveaux événements sont potentiellement générés.

**3.3. Déroulement d’une simulation**

Le déroulement global d'une simulation suit les étapes suivantes :

**I - Initialisation**

* Définition des paramètres : C, λ, durée de simulation, facteur d’accélération des serveurs spécialisés.
* Création du **routeur** avec sa file limitée à 100 requêtes.
* Création des **groupes de serveurs** : chaque groupe contient 12/C serveurs.
* Définition des **catégories traitées** par chaque groupe.
* Génération de la **première requête** et insertion dans l’échéancier.

**II - Boucle de simulation**

Tant que le temps simulé est inférieur à la durée limite, et que le **taux de perte reste inférieur à 5%** :

1. On extrait le **prochain événement** dans l’échéancier.
2. Le **routeur traite** la requête si libre, sinon il attend.
3. Si la requête est traitée :
   * Elle est envoyée à un serveur **du groupe approprié**.
   * Si aucun serveur n’est disponible, le routeur **bloque** jusqu’à libération.
4. Un nouvel événement d'**arrivée de requête** est généré.

**III - Mise à jour**

* Tous les serveurs sont mis à jour à chaque événement : s’ils ont terminé un traitement, ils deviennent disponibles.

**3.4. Critères d’évaluation**

À la fin de la simulation, nous calculons :

* Le **temps de réponse moyen** :
* Le **taux de perte** :

Si le taux de perte dépasse 5%, la configuration est considérée comme **non acceptable**.

Expérimentation :