Dossier de conception Monte Carlo

1. Objectif du module

L'objectif de ce module est d'approximer la valeur de π à l'aide de la **méthode de Monte Carlo**, en exploitant un **cluster de Raspberry Pi** organisé selon une architecture **Master / Workers**, reposant sur des **communications réseau via sockets TCP**.

Le calcul est parallélisé afin de :

- réduire le temps d'exécution,
- exploiter les ressources matérielles disponibles,
- étudier l'impact du nombre de workers sur les performances.

2. Architecture générale

L'architecture adoptée est une **architecture distribuée** :

- **Un Master** :
    - coordonne les calculs,
    - répartit le travail,
    - collecte les résultats,
    - calcule la valeur finale de π,
    - mesure les performances.
- **Plusieurs Workers** :
    - exécutent localement une simulation Monte Carlo,
    - renvoient au Master le nombre de points dans le quart de disque.

Type d'architecture

- **Mémoire distribuée**
- **Communication explicite** par sockets TCP
- **Modèle Client / Serveur**

3. Choix techniques

- **Langage**
  - Java (portabilité, sockets natives, gestion simple des threads et I/O)
- **Communication**
  - Sockets TCP (`java.net.Socket`, `ServerSocket`)
  - Protocole texte simple (messages `String`)
- **Parallélisation**
  - Répartition statique :
    - chaque worker reçoit le même nombre d'itérations
  - Agrégation côté Master
- **Déploiement**
  - Plusieurs workers sur le **RPi4 (4 cœurs)**
  - Un worker par **RPi Zero (1 cœur)**

4. Description détaillée des classes

4.1. Classe `MasterSocket`

Rôle

Le Master est responsable de :

- initialiser les connexions réseau,
- distribuer les paramètres de calcul,
- collecter les résultats,
- calculer π,
- mesurer les performances,
- sauvegarder les résultats.

Attributs principaux

Attribut	Type	Rôle
maxServer	int	Nombre maximal de workers
tab_port[]	int[]	Ports des workers
ips[]	String[]	Adresses IP des workers
reader[]	BufferedReader[]	Réception des résultats
writer[]	PrintWriter[]	Envoi des paramètres
sockets[]	Socket[]	Connexion TCP

Méthodes

- `main(String[] args)`
    - Lecture de `totalCount` en ligne de commande
    - Connexion aux workers
    - Distribution des tâches
    - Agrégation des résultats
    - Calcul de π
    - Sauvegarde dans un fichier CSV

4.2. Classe `WorkerSocket`

Rôle

Le Worker est un serveur TCP chargé de :

- recevoir un nombre d'itérations,
- exécuter une simulation Monte Carlo locale,
- renvoyer le nombre de points dans le quart de disque.

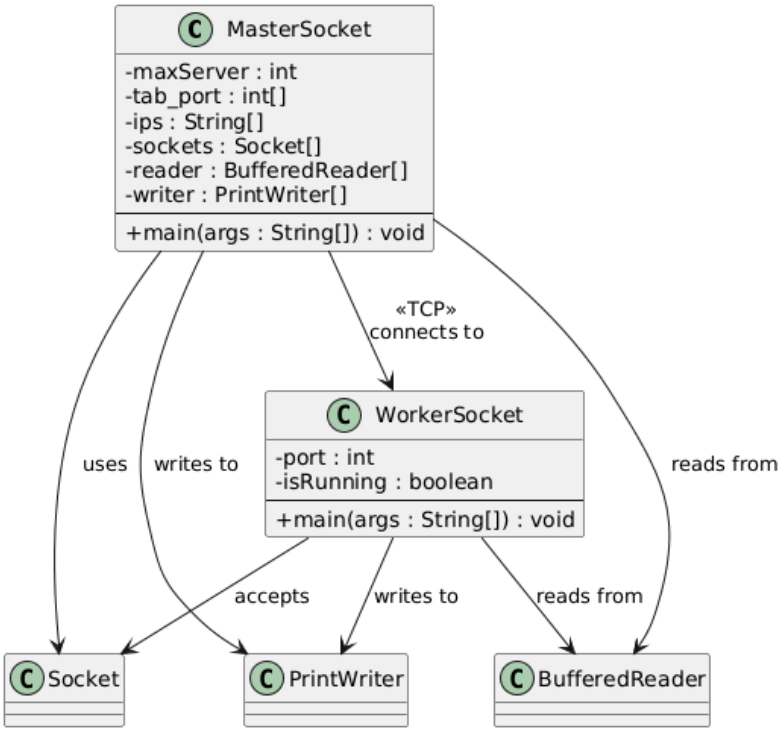
Attributs principaux

Attribut	Type	Rôle
port	int	Port d’écoute
isRunning	boolean	Boucle d’exécution

Méthodes

- `main(String[] args)`
    - Démarrage du serveur
    - Attente de connexions
    - Calcul Monte Carlo
    - Envoi du résultat au Master

5. Diagramme de classes



6. Gestion du parallélisme

- Chaque Worker exécute :
    - `totalCount` itérations indépendantes
- Le calcul est **embarrassingly parallel**
- Aucun partage d'état entre workers
- Synchronisation implicite lors de la réception des résultats

9. Perspective d'amélioration

- Passage à MPI pour réduire la latence