Compte-rendu SAE S5

I) Sujet

Ce projet a pour objectif de développer une plateforme web connectée à un cluster de Raspberry Pi composé d'un Raspberry Pi 4 et de 4 Raspberry Pi 0, permettant d'exécuter des programmes de calcul distribué.
Nous devions préparer les Raspberry Pi pour pouvoir être utilisés en installant l'OS et tous les outils qui pourraient nous être utiles ainsi que faire toutes les configurations nécessaires.

Les programmes de calcul distribué à installer étaient :

- Un programme de calcul permettant de calculer tous les nombres premiers jusqu'à un certain point donné en argument et ce grâce à MPI.
- Un programme de calcul permettant d'approximer la valeur de π grâce à la méthode de Monte Carlo
- Un programme de calcul permettant de calculer la valeur de l'intégrale de la fonction de Gauss-Laplace grâce à la méthode de Simpson

L'objectif de ces programmes était de mesurer leur performance en matière de temps d'exécution au sein du cluster avec un certain nombre de workers.

II) Groupe

Le groupe de projet était constitué de :

- Noah Lavocat :Setup du cluster, création de la BD et réalisation des tests de performances
- Luca Da Silva : Développement de l'application web
- Esteban Colombani : Développement du site, interactions avec le cluster
- Florian Bouchery : Configuration de MPi et développement des programmes de calcul
- Roméo Leblond : Ecriture des cas d'utilisation

III) Organisation

Pour ce projet, nous nous sommes organisés avec un cycle en cascade. En effet, il s'agit du cycle de vie de projet que nous avons le plus effectué et nous l'avons choisi naturellement.

Néanmoins, une utilisation des méthodes Agiles aurait été meilleure pour avoir plus de flexibilité. En effet, un cycle en cascade est très rigide et ne peut pas revenir en arrière sur quelque chose qui a été fait. Cependant, le sujet original ne donnait pas les modules de calcul à développer et de nouveaux modules pouvaient nous être demandés plus tard. Nous avons donc rencontré des problèmes lorsque le module de calcul intégral a été ajouté.

Ainsi, si un projet similaire nous est demandé à l'avenir nous choisirons une méthode Agile tel que le SCRUM.

IV) Setup du cluster

L'installation du cluster a débuté par la mise en place du système d'exploitation sur chaque Raspberry. Nous avons utilisé Raspberry Pi Imager pour flasher les cartes SD avec des images spécifiques fournies par ClusterHAT : une version "CNAT Desktop" pour le master (Raspberry Pi 4) et des versions "Lite" sans interface graphique pour les quatre workers (Raspberry Pi 0).

Une fois les cartes insérées et le cluster alimenté, la configuration réseau s'est faite via le contrôleur ClusterCTRL. Le Raspberry Pi 4 agit comme un pont : il reçoit la connexion Internet via Ethernet et distribue un sous-réseau local (via l'interface br0) aux Raspberry Pi 0. Nous avons ensuite activé le SSH sur toutes les machines pour pouvoir les contrôler à distance et installé les outils nécessaires au projet, notamment Apache2 et PHP pour l'hébergement web, ainsi que MariaDB pour la gestion de la base de données.

V) Site web

La partie site web du projet a été réalisée par Luca Da Silva et Esteban Colombani, avec une répartition précise des responsabilités.

Luca s'est occupé du développement de l'ensemble du site web, à l'exception des modules de calcul. Il a notamment développé :

- l'architecture globale du site,
- le système de connexion et d'inscription des utilisateurs,
- la gestion des sessions et des droits,
- le back-end et le front-end du site,
- l'interface générale permettant la navigation et l'accès aux différentes fonctionnalités pour les utilisateurs et administrateurs.

Esteban s'est chargé du développement des pages web dédiées aux modules de calcul distribués implémentés par Florian Bouchery. Il a travaillé sur les interfaces permettant de paramétrer (nombre de workers locaux et distants, paramètres numériques, bornes de calcul, etc.) et de lancer les calculs (MPI, Monte Carlo et Simpson), ainsi que sur l'affichage des résultats pour l'utilisateur.

Il a donc assuré la communication entre l'interface web et les programmes de calcul hébergés sur le cluster RaspberryPi, en mettant en place l'exécution distante des processus, la gestion des workers, ainsi que la récupération et l'affichage structuré des résultats. Il a aussi dû renforcer la validation des entrées utilisateur, la gestion des erreurs et la sécurisation minimale des appels système afin de garantir la stabilité et la fiabilité des calculs lancés depuis le site.

Cette organisation a permis de séparer clairement la gestion du site et l'implémentation des modules de calcul, facilitant leur intégration et la maintenance du projet.

VI) Installation MPI et programme prime.py

Le premier programme qu'il nous a fallu développer fut le code prime.py.
Ce programme Python avait pour objectif de calculer tous les nombres premiers jusqu'à un nombre passé en argument.

Le code nous était fourni et il nous a fallu le comprendre puisqu'il utilisait MPI, un outil que nous n'avions jamais utilisé auparavant. MPI (ou Message Passing Interface) est un outil permettant aux différentes machines de notre cluster de pouvoir communiquer par envoi de messages.

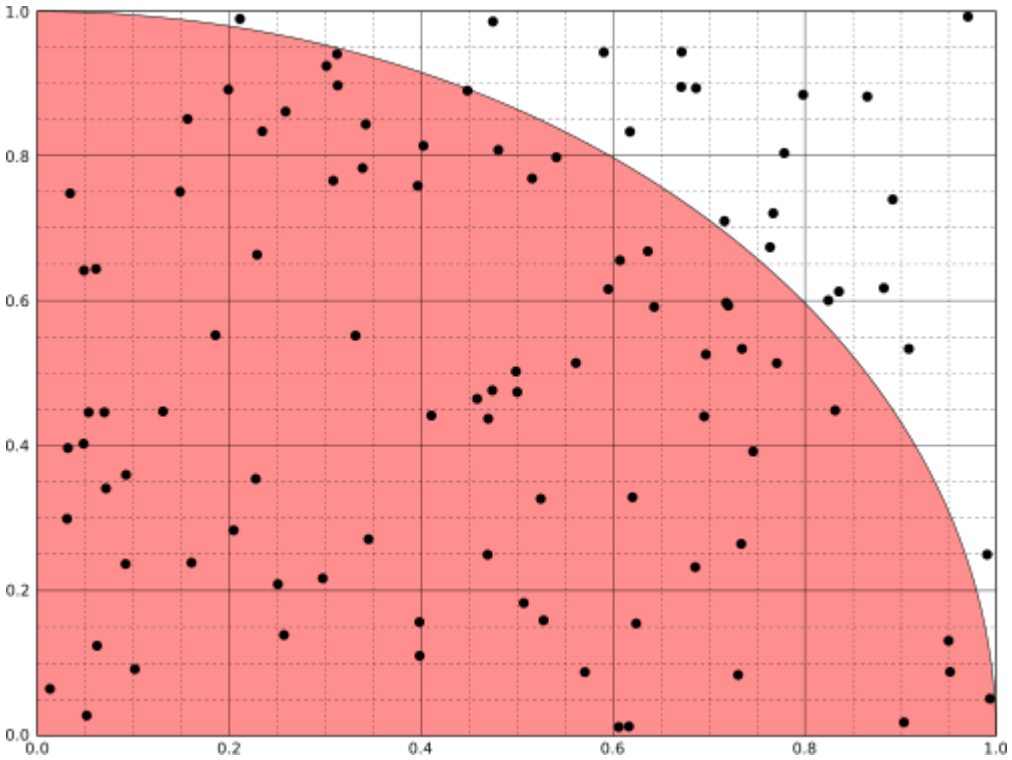
Le code s'organisait sous forme de paradigme Master/Worker cependant MPI inclut également une notion de rang. Ainsi, nous pouvons indiquer que seule un certain rang exécute certaines parties du programme et nous pouvons ainsi donner un même programme à exécuter pour le master et pour les workers puisqu'ils n'exécuteront pas la même partie du code. C'est ce que l'on appelle un paradigme SPMD (Single Program Multiple Data).
Ainsi le code prime.py utilisait 2 paradigmes : Master/Worker et SPMD.

La commande pour lancer ce code est la suivante (avec l'utilisateur mpiuser) :
`mpirun -np <nombre-de-workers> -hostfile hosts python3 prime.py <nombre-max>`

VII) Programme Monte Carlo

Le deuxième programme qu'il nous a fallu développer fut le code du calcul d'une valeur approchée de π grâce à la méthode de Monte Carlo.

La méthode de Monte Carlo est la suivante :
Soit un point M de coordonnées (x, y), où 0 < x < 1 et 0 < y < 1. On tire aléatoirement les valeurs de x et y entre 0 et 1 suivant une loi uniforme. Le point M appartient au disque de centre (0,0) de rayon R = 1 si et seulement si x^2 + y^2 ≤ 1.



Soit d = √(y^2 + x^2) et R le rayon du cercle, on a donc P(d < 1) = ((π * R^2)/4)/R^2 = π/4
Or P(d < 1) ≃ n_cible/n_tot où n_cible est le nombre de points dans le quart de cercle et n_tot est le nombre de points total.
Ainsi, π/4 ≃ n_cible/n_tot et donc π ≃ 4 * n_cible/n_tot.

Nous avons développé plusieurs versions de ce code lors des cours de Programmation Avancée mais nous avons choisi de développer sur notre cluster la version utilisant les Sockets car il fallait faire du calcul distribué entre plusieurs machines distinctes et les méthodes précédentes de calcul ne permettant que de faire du calcul partagé sur une seule machine.

La commande pour lancer un worker est la suivante :
`java WorkerSocket <port>`

La commande pour lancer le master est la suivante :
`java MasterSocket <totalCount/n_tot>`

VIII) Programme calcul intégral de la fonction de Gauss-Laplace avec la méthode de Simpson

Le dernier programme que nous avions à développer fut un code permettant d'effectuer le calcul de l'intégrale de la fonction de Gauss-Laplace (loi normale) grâce à la méthode de Simpson.
La méthode de SImpson est la suivante :

1. On découpe l'intervalle [a,b][a,b][a,b] en n sous-intervalles de largeur égale (n pair).

2. On regroupe les sous-intervalles par deux.
3. Sur chaque paire, on approxime la fonction par une parabole passant par 3 points.

Soit :

$$h = \frac{b-a}{n} \quad \text{et} \quad x_i = a + ih$$

Alors :

$$\int_a^b f(x)\,dx \approx \frac{h}{3}\left[f(x_0) + f(x_n) + 4\sum f(x_{2k-1}) + 2\sum f(x_{2k})\right]$$

Ce code était à développer intégralement soit en Python avec MPI (comme pour prime.py) soit en Java avec les Socket comme pour Monte Carlo. Nous avons choisi les Sockets Java car il s'agissait de la méthode que nous maîtrisons le mieux.

La commande pour lancer un worker est la suivante :
`java WorkerSimpsonSocket <port>`

La commande pour lancer le master est la suivante :
`java MasterSimpsonSocket <a> <b> <n> <nombre-de-workers>`

IX) Conclusion

Ce projet nous a permis de mettre en pratique tout ce que nous avons vu lors de notre BUT en matière de site web, de développement Python et Java, de gestion de base de données, de configuration Linux et également de calcul distribué.
Nous avons également pu voir ce à quoi ressemble un projet lorsque nous ne sommes pas guidés comme lors des SAE précédentes.

Nous aurions cependant pu mieux nous organiser ce qui aurait rendu certaines parties du projet plus simples et efficaces.