Valérian Lambert
Julie Leroy
Arsene Jerome
Thomas Madrange-Maire
Clément Berard

SAE5 - Compte Rendu Global

sae 3e année 2025-2026

Sommaire

I. Introduction

A. Contexte du projet

Ce projet a été réalisé dans le cadre de la SAE 5 du semestre 5. Il a pour but de mettre en pratique les notions vues en cours.

Le sujet du projet repose sur la mise en place et l'exploitation d'une infrastructure de calcul distribué basée sur un cluster de Raspberry Pi. Ce type d'infrastructure permet d'expérimenter le calcul parallèle avec des ressources matérielles limitées, tout en restant proche des principes utilisés dans des environnements de calcul haute performance (HPC).

Ce projet combine donc plusieurs aspects : la mise en place d'un cluster, le développement d'algorithmes parallèles et la création d'une interface web permettant de faire le lien entre l'utilisateur et l'infrastructure de calcul.

B. Objectifs du document

Ce document a pour objectif de présenter de manière globale le travail réalisé durant la SAE 5. Il décrit l'organisation du projet, les choix techniques effectués ainsi que les résultats obtenus.

Plus précisément, ce rapport vise à :

- expliquer comment le projet a été organisé et découpé.
- expliquer la mise en place du cluster.
- présenter l'architecture de la plateforme web et du cluster de Raspberry Pi
- décrire les différents algorithmes de calcul parallèle développés.
- analyser les performances obtenues, notamment à travers l'étude de la scalabilité forte et faible.
- comparer les différentes approches de parallélisme utilisées.

Ce rapport sert également de synthèse des différents sous-rapports réalisés au cours du projet, qui sont disponibles en annexe pour plus de détails.

II. Organisation et gestion du projet

A. Présentation de l'équipe et du cadre de travail

Le projet a été réalisé par une équipe de cinq étudiants sur une durée d'environ trois à quatre mois. Le travail s'est déroulé dans le cadre de la SAE 5, avec des objectifs définis au cours des mois.

L'organisation du projet a reposé sur un travail collaboratif, avec une répartition des tâches en fonction des compétences de chacun, tout en gardant une communication régulière entre les membres du groupe. Cette organisation nous a permis d'avancer de manière progressive sur les différentes parties du projet, tout en assurant une cohérence globale entre les sous-projets.

B. Analyse des Besoins

1. Cahier des charges

La première étape du projet a été une analyse des besoins, formalisée par la rédaction d'un cahier des charges. Ce document avait pour objectif de définir clairement le périmètre du projet, ses objectifs principaux ainsi que les fonctionnalités attendues.

Le cahier des charges a permis de préciser les attentes générales du projet, notamment la mise en place d'un cluster de Raspberry Pi, le développement de plusieurs algorithmes de calcul parallèle et la création d'une plateforme web permettant de piloter ces calculs. Il a également servi de référence tout au long du projet afin de s'assurer que les développements réalisés correspondaient bien aux objectifs initiaux.
Annexe CdC:
https://github.com/TMMproject/SAE5_RaspBerryParallel/blob/main/Documentation/Requirements_Specification(CdC)/Cahier_des_charges.pdf

2. Recueil des besoins

Après la rédaction du cahier des charges, un recueil des besoins plus détaillé a été réalisé. Cette étape a permis d'identifier les besoins fonctionnels et non fonctionnels du projet.
Les besoins fonctionnels concernent principalement les actions que l'utilisateur doit pouvoir réaliser via la plateforme web, comme la création d'un compte, la connexion, le choix d'un module de calcul, le paramétrage des entrées et la consultation des résultats.

Les besoins non fonctionnels portent quant à eux sur les contraintes techniques, telles que les limites matérielles des Raspberry Pi, les performances attendues, la sécurité des données ou encore la fiabilité de l'infrastructure.
Cette phase a également permis de mieux définir le fonctionnement global du système à l'aide de cas d'utilisation, décrivant les interactions entre l'utilisateur, la plateforme web et le cluster de calcul.

Annexe Recueil des besoins:
https://github.com/TMMproject/SAE5_RaspBerryParallel/blob/main/Documentation/Need_Analysis(AdB)/Requirements_Gathering(RDB)/Requiremens_Gathering_(RDB)_SAE5_v2.pdf

Annexe besoins fonctionnel/non fonctionnel:
https://github.com/TMMproject/SAE5_RaspBerryParallel/blob/main/Documentation/Need_Analysis(AdB)/Functional-nonfunctional/Functional_Non_Functional_Requirements_SAE5_v1.pdf

C. Planification et gestion des risques

1. Planification

La planification du projet a été effectuée à l'aide d'un diagramme de Gantt, nous permettant de visualiser l'enchaînement des tâches et de suivre l'avancement du projet dans le temps. En parallèle, nous avons mené une analyse des risques, en identifiant les principaux obstacles potentiels (retards, problèmes matériels, difficultés techniques) et en proposant des solutions préventives afin de limiter leur impact sur le projet.
Gantt Annexe:
https://github.com/TMMproject/SAE5_RaspBerryParallel/tree/main/Documentation/Gantt_Diagram

2. Gestion des risques

En parallèle de la planification, une analyse des risques a été menée afin d'anticiper les éventuelles difficultés pouvant impacter le projet. Parmi les principaux risques identifiés figurent les problèmes matériels liés aux Raspberry Pi, les difficultés techniques liées au calcul parallèle, ainsi que les risques de retard dans l'avancement des tâches.
Pour chaque risque, des solutions préventives ont été proposées, comme la mise en place de sauvegardes, des tests réguliers ou encore une répartition flexible des tâches au sein de l'équipe. Cette approche nous a permis de limiter l'impact des imprévus sur le déroulement du projet.
Gestion des risques Annexe:
https://github.com/TMMproject/SAE5_RaspBerryParallel/blob/main/Documentation/Risk_Assessment(GdR)/Gestion%20des%20risques_SAE5.pdf

D. Découpage du projet et suivi des tâche

1. Sous-projets identifiés

Le projet a été découpé en plusieurs sous-projets afin de faciliter son organisation. Initialement, deux - trois sous-projets principaux avaient été définis :

- le développement de la plateforme web.
- le calcul des nombres premiers sur le cluster de Raspberry Pi.
- l'implémentation de l'algorithme de Monte Carlo. (Qui nous a été délivré au cours du semestre).

Un quatrième sous-projet, portant sur le calcul de l'intégrale de la loi normale par la méthode de Simpson, a été ajouté en fin de projet.

2. Organisation du travail

Chaque sous-projet a suivi une organisation proche d'un cycle en cascade, avec des étapes successives de conception, de développement, de tests et de validation. Cette approche nous a permis de structurer le travail et de progresser de manière claire sur chaque partie du projet.

La communication entre les membres du groupe a permis d'assurer la cohérence entre les sous-projets, notamment pour l'intégration des algorithmes de calcul au sein de la plateforme web.

3. Outils de suivi

Le suivi des tâches a été assuré à l'aide de l'outil Trello (ainsi que Discord), accessible à l'ensemble des membres du groupe. Les tâches étaient organisées par sous-projets, ce qui facilitait la visualisation du travail à réaliser.

Un système de codes couleur a été mis en place pour indiquer l'état des tâches : en attente, en cours, urgente, terminée ou abandonnée. Cet outil nous a permis de suivre l'avancement du projet en temps réel, de mieux gérer les priorités et d'améliorer la coordination au sein de l'équipe.

III. Mise en place et configuration du cluster de RPis

A. Matériel

- Un Raspberry Pi 4B (4 coeurs d'1.8GHz, 4Go de RAM)
- 4 Raspberry Pi Zéro (1 coeur d'10oGHz, 4Go de RAM)
- Un Clusterhat 4 places qui fait le lien entre RPi 4 et RPi 0
- Cartes microSD, câbles d'alimentation et de connexion

B. Choix de l'OS

Nous avons décidé d'utiliser un ensemble d'images système regroupés sous le nom Cluster CTRL et prenant pour base le Raspberry Pi OS, et nous avons sélectionné sur le site les images qui correspondent à notre hardware. Il s'agit d'images 32bit car les RPi 0 ne supportent pas le 64bit, avec pour le contrôleur une image utilisant le CNAT au lieu du CBRIDGE pour éviter les problèmes de configuration réseau (CNAT ne dépend pas de la configuration d'eth0 et marche en WiFi comme en Ethernet contrairement à CBRIGE). Nous avons pris une image avec interface graphique

C. Démarrage du cluster

On commence par flasher les images sur les contrôleurs, démarrer le RPi 4 et le configurer. On configure ensuite les utilisateurs et les mots de passe des 4 RPi 0 en cryptant les mots de passes sur le contrôleur, on les inclut dans les partitions de boot et on peut démarrer les RPi 0 (attention à respecter l'emplacement de chaque carte SD en fonction de l'OS installé comme précisé dans le rapport détaillé sur les RPis en annexe). Enfin, une fois les RPis démarrés, on commence la configuration du SSH en générant une clé SSH et en les stockant sur les RPis nodes. Ainsi, on assure une connexion sans mot de passe du contrôleur aux nodes.

Annexe Rapport RPI:
https://github.com/TMMproject/SAE5_RaspBerryParallel/blob/main/Documentation/Reports(CR)/SAE_RPi_Report.pdf

D. Installation de logiciels

Une fois le RPi fonctionnel, on peut commencer à installer les logiciels qui nous intéressent comme l'architecture LAMP pour faire tourner un serveur web en PHP avec une base de données, la bibliothèque Python MPI ou le langage Java. On installe aussi d'autres logiciels, qui servent plus à notre confort de développement qu'au fonctionnement réel du cluster. On pense notamment à Tailscale, qui permet de créer un tunnel SSH et de travailler à distance sur le RPi, et à RealVNC qui passe par ce tunnel SSH pour créer un bureau à distance du RPi et ainsi travailler avec une interface graphique.

Annexe NFS:
https://github.com/TMMproject/SAE5_RaspBerryParallel/blob/main/Documentation/Reports(CR)/rapport_NFS_setup.md

Annexe VNC & Tailscale :
https://github.com/TMMproject/SAE5_RaspBerryParallel/blob/main/Documentation/Reports(CR)/rapport_VNC_TailScale.md

IV. Site Web

La partie web du projet a pour objectif principal de rendre accessible l'infrastructure de calcul distribué basée sur un cluster de Raspberry Pi. L'idée est de permettre à un utilisateur connecté, quel que soit son niveau technique, de lancer et piloter des calculs parallèles sans avoir à interagir directement avec les machines du cluster ou les scripts de calcul.
Pour répondre à cet objectif, nous avons développé une interface web simple et intuitive, servant d'intermédiaire entre l'utilisateur et l'infrastructure de calcul. Cette interface centralise la gestion des utilisateurs, le paramétrage des calculs et l'affichage des résultats.

Annexe Website:
https://github.com/TMMproject/SAE5_RaspBerryParallel/blob/main/Documentation/Reports(CR)/SAE_Website_Report.pdf

Annexe Diagramme Infrastructure:
https://github.com/TMMproject/SAE5_RaspBerryParallel/tree/main/Documentation/Diagramme

A. Conception de l'interface

Le développement de la partie web a commencé par une phase de conception. Cette étape avait pour but de définir l'identité visuelle du site ainsi que son organisation générale, afin de proposer une interface claire et facile à prendre en main.

Une première maquette a été réalisée à l'aide de l'outil Figma. Cette maquette nous a permis de réfléchir à l'ergonomie du site, à la disposition des différents éléments (menus, formulaires, zones de résultats) ainsi qu'au choix des couleurs à partir d'une palette définie. La maquette a facilité les échanges au sein du groupe et a servi de base pour la suite du développement.

Cette phase de conception nous a permis d'avoir une vision claire du fonctionnement attendu de l'interface avant de passer à l'implémentation technique.

B. Conception de la base de données

En parallèle de la conception de l'interface, un travail a été mené sur la conception de la base de données. L'objectif était d'anticiper la gestion des données nécessaires au bon fonctionnement de l'application.

Un schéma de base de données a été réalisé afin de définir les différentes tables, notamment pour :

- la gestion des utilisateurs (inscription, connexion, sessions),
- le stockage des informations liées aux calculs et aux résultats.

Cette étape a permis de structurer les données dès le départ et d'assurer une cohérence entre la partie backend et la partie frontend de l'application.
Doc BDD Annexe:
https://github.com/TMMproject/SAE5_RaspBerryParallel/tree/main/Documentation/BDD_documentation

C. Architecture du site web

1. Technologies utilisées

À partir de la maquette réalisée sur Figma, une première version statique du site a été développée en HTML et CSS. Cette version a ensuite servi de base pour la mise en place de l'application web dynamique.

L'architecture du site repose sur les technologies suivantes :

- PHP pour la logique serveur et le traitement des requêtes.

- HTML, CSS et JavaScript pour la construction de l'interface utilisateur et les interactions côté client.
- MySQL pour la gestion de la base de données.

Ce choix de technologies correspond à des outils couramment utilisés, adaptés à un projet web de cette ampleur et compatibles avec l'environnement du cluster.

2. Structure du site

L'application web est composée de plusieurs pages fonctionnelles, parmi lesquelles :

- une page d'accueil.
- des pages d'inscription et de connexion.
- un tableau de bord.
- une page de profil utilisateur.
- des pages de déconnexion et de redirection en cas d'erreur.

Afin d'améliorer la lisibilité et la maintenabilité du code, les éléments communs du site, tels que le header, le footer et la barre de navigation, ont été mutualisés à l'aide de fichiers inclus dans chaque page. Cette organisation permet d'éviter les duplications de code et de faciliter les évolutions futures du site.

D. Fonctionnalités

1. Gestion des utilisateurs

La gestion des utilisateurs est l'une des premières fonctionnalités mises en place, car elle est essentielle au bon fonctionnement de la plateforme. Un système d'authentification a été développé, comprenant des formulaires d'inscription et de connexion utilisant la méthode HTTP POST, afin de sécuriser les données transmises.

Lors de l'inscription, les informations demandées sont l'adresse email et le mot de passe. Ces données sont stockées dans la base de données de manière sécurisée. Le mot de passe est haché à l'aide de la fonction password_hash() de PHP, qui offre un niveau de sécurité bien supérieur aux méthodes de hachage plus anciennes comme MD5.

La gestion des sessions PHP permet de restreindre l'accès aux fonctionnalités sensibles, notamment le tableau de bord, aux seuls utilisateurs authentifiés. Un captcha a également été intégré lors de l'inscription afin de limiter la création automatisée de comptes.

2. Tableau de bord

Le tableau de bord constitue le coeur de la plateforme web. Il sert de passerelle entre l'utilisateur et l'infrastructure de calcul distribué.

Il a été conçu de manière à faciliter l'utilisation des différents modules de calcul, avec une séparation claire entre :

- la sélection du module de calcul.
- le paramétrage des entrées.
- l'affichage des résultats.

Des scripts JavaScript ont été utilisés pour rendre l'interface plus dynamique, notamment afin d'éviter des rechargements complets de page et de conserver les choix de l'utilisateur après l'exécution d'un calcul.

3. Maintenabilité et Formulaire dynamique

Afin d'améliorer la qualité du code et de faciliter les évolutions futures de l'application, nous avons mis en place un système de formulaires dynamiques pour le paramétrage des calculs.

Les caractéristiques des différents modules de calcul (paramètres requis, types d'entrées, scripts associés) sont centralisées dans un fichier de configuration. Le code PHP parcourt ce fichier pour générer automatiquement les formulaires correspondants.

Cette approche permet d'ajouter de nouveaux modules de calcul sans modifier directement la structure HTML du site, ce qui améliore la maintenabilité et l'évolutivité de l'application.

4. Intégration des modules de calcul

Plusieurs modules de calcul ont été intégrés à la plateforme web :

un module de calcul des nombres premiers en parallèle.

- un module d'approximation de Pi par la méthode de Monte Carlo.
- un module de calcul de l'intégrale de la loi normale par la méthode de Simpson.

Pour chaque module, l'utilisateur renseigne les paramètres nécessaires ainsi que le nombre de workers à utiliser. Ces informations sont ensuite transmises au backend, qui déclenche l'exécution des scripts correspondants sur le cluster de Raspberry Pi.

Les résultats produits par les programmes sont récupérés et affichés directement sur la plateforme web, permettant à l'utilisateur de visualiser simplement les résultats des calculs.

5. Graphique dans le SiteWeb

Un module de scalabilité est disponible à côté des autres modules et permet de générer un graphique de scalabilité forte ou faible. Il est généré grâce à un code Python utilisant matplotlib et est affiché sur le site web. On peut sélectionner tous les programmes proposés à l'utilisateur dans les différents modules, on peut régler la scalabilité forte ou faible, et les graphiques sont basés sur des données préenregistrées pour des raisons de performance.

V. Calcul des nombres premiers

A. Présentation du problème

L'objectif est de trouver tous les nombres premiers dans un intervalle donné. Plus l'intervalle est grand, plus le calcul devient long.
Chaque nombre peut être testé indépendamment, ce qui permet de répartir le travail sur plusieurs machines ou processus.
Sur notre cluster de Raspberry Pi, il est important de bien organiser la répartition pour éviter que certaines machines restent inactives tandis que d'autres travaillent encore.

B. Implémentations réalisées

Plusieurs versions du programme de calcul des nombres premiers ont été développées au cours du projet. Ces différentes implémentations avaient pour objectif d'améliorer progressivement les performances et d'explorer différentes stratégies de parallélisation et de répartition de charge.

1. Version de base : prime.py

La première implémentation, nommée prime.py (programme de base), correspond à une version simple du calcul des nombres premiers en parallèle. Le principe repose sur un découpage statique de l'intervalle de calcul entre les différents processus.

Chaque processus reçoit une portion fixe de l'intervalle et vérifie si les nombres de cette plage sont premiers. Une fois le calcul terminé, les résultats sont rassemblés afin d'obtenir le résultat global.

Cette version permet de mettre en place une première parallélisation, mais elle présente rapidement des limites. En effet, le temps de calcul peut varier d'un processus à l'autre en fonction des valeurs à traiter, ce qui peut entraîner un déséquilibre de charge et une sous-utilisation de certaines ressources.

2. Version améliorée : prime_improve.py

La version prime_improve.py apporte des optimisations par rapport à la version de base. Ces améliorations concernent principalement l'algorithme de test de primalité, afin de réduire le nombre d'opérations inutiles.

Par exemple, certaines vérifications redondantes ont été supprimées et des optimisations mathématiques ont été mises en place, comme la limitation des tests jusqu'à la racine carrée du

nombre étudié. Ces changements permettent de réduire le temps de calcul global sans modifier fondamentalement la stratégie de parallélisation.

Cette version montre que l'optimisation de l'algorithme lui-même est aussi importante que la parallélisation pour améliorer les performances.

Sous Rapport dans l'Annexe:
https://github.com/TMMproject/SAE5_RaspBerryParallel/blob/main/Documentation/Reports(CR)/rapport_prime_improve.md

3. Répartition dynamique de la charge : prime_dynamic_load_balancer.py

L'implémentation prime_dynamic_load_balancer.py a été développée dans l'optique de travailler sur un cluster hétérogène. Contrairement à un cluster homogène, où toutes les machines ont des performances similaires, notre infrastructure peut inclure des Raspberry Pi de générations différentes, comme le Raspberry Pi 4, plus puissant, utilisés en tant que worker.

Dans cette version, la répartition du travail ne se fait plus de manière statique. Un processus maître est chargé de distribuer dynamiquement des segments de calcul aux workers. Lorsqu'un worker termine le traitement d'un segment, il en demande un nouveau au maître. Cela permet aux machines les plus performantes de traiter davantage de travail que les machines plus lentes.

Cette approche permet de tester différentes configurations du cluster, par exemple en intégrant ou non certains noeuds plus puissants, et d'observer leur impact sur les performances globales. Le load balancing dynamique améliore ainsi l'utilisation des ressources et rend l'algorithme plus adapté à des environnements réalistes où les performances des machines ne sont pas uniformes.

Dans le contexte de notre cluster nous pouvons avoir de potentielles pertes au vu du délais que la transmission des informations implique.

4. Version optimisée avec équilibrage dynamique et crible d'Ératosthène : prime_DLB_improve.py

La version prime_DLB_improve.py va encore plus loin en combinant la répartition dynamique de charge avec une amélioration significative de l'algorithme de calcul des nombres premiers.

Dans cette implémentation, le calcul de primalité est optimisé grâce à l'utilisation du crible d'Ératosthène, qui permet de réduire fortement le nombre de calculs nécessaires par rapport à un test de primalité classique. Cette méthode est particulièrement efficace pour le calcul de grands intervalles de nombres premiers.

En associant le crible d'Ératosthène à un équilibrage dynamique de charge, cette version permet à la fois :

- une meilleure répartition du travail sur un cluster potentiellement hétérogène.
- une réduction du temps de calcul grâce à un algorithme plus performant.

Cette implémentation représente la version la plus avancée du calcul des nombres premiers dans le cadre du projet.

Sous rapport dans l'Annexe:
https://github.com/TMMproject/SAE5_RaspBerryParallel/blob/main/Documentation/Reports(CR)/rapport_Load_Balancing_et_Crible_d'Ératoshène.md

5. Implémentation alternative en TCP Client/Server : prime_tcp_server.py et prime_tcp_client.py

En complément des implémentations basées sur MPI, une version du calcul des nombres premiers utilisant un modèle TCP Client/Server a également été développée.

Dans cette approche, un serveur (prime_tcp_server.py) est responsable de la gestion du calcul et de la distribution du travail, tandis que plusieurs clients (prime_tcp_client.py) se connectent au serveur pour exécuter les calculs qui leur sont attribués. Cette architecture se rapproche d'un modèle compute-on-server, où la communication repose sur des sockets TCP plutôt que sur MPI.

Cette implémentation a pour objectif de comparer le paradigme TCP Client/Server avec les approches basées sur MPI, notamment en termes de simplicité de mise en oeuvre, de performances et de gestion des communications. Une analyse plus détaillée de cette version est présentée dans le rapport dédié à la comparaison des paradigmes de parallélisme et au modèle TCP Client/Server, disponible en annexe.

sous rapport dans l'Annexe:
https://github.com/TMMproject/SAE5_RaspBerryParallel/blob/main/Documentation/Reports(CR)/Rapport%20_Comparaison_des_paradigmes_de_parallélisme_(Analyse_TCP).md

C. Analyse des performances

L'analyse des performances a été réalisée en étudiant la scalabilité forte et la scalabilité faible des différentes implémentations sur le cluster de Raspberry Pi.

L'intégralité de nos mesures sont présentes dans le sous rapport en annexe.
Rapport Annexe:
https://github.com/TMMproject/SAE5_RaspBerryParallel/blob/main/Documentation/Reports(CR)/SAE_Rapport_Scalabilité.pdf

1. Scalabilité forte

La scalabilité forte consiste à mesurer l'évolution du temps de calcul lorsque la taille du problème reste constante et que le nombre de processus augmente.

2. Scalabilité faible

La scalabilité faible mesure la capacité du système à maintenir un temps de calcul stable lorsque la taille du problème augmente proportionnellement au nombre de processus

VI. Méthode de Monte Carlo pour Pi

A. Principe de la méthode

Le principe consiste à générer un grand nombre de points aléatoires dans un carré de côté 1 et à compter combien de ces points se trouvent à l'intérieur d'un quart de cercle de rayon 1. Le rapport entre le nombre de points dans le cercle et le nombre total de points permet d'estimer la valeur de Pi.

Cette méthode est particulièrement adaptée au calcul parallèle, car chaque point peut être généré et traité indépendamment des autres. Il est donc possible de répartir le travail entre plusieurs processus ou machines sans dépendance forte entre eux.

B. Implémentations

Plusieurs implémentations de la méthode de Monte Carlo ont été développées afin d'explorer différents paradigmes de parallélisme et de communication. Ces implémentations ont été réalisées tout d'abord en Java puis en Python.

1. Version Master/Worker en Java

Une première implémentation de la méthode de Monte Carlo a été réalisée en Java, en utilisant un modèle Master/Worker basé sur des sockets TCP. En utilisant les API Concurrent et Java Socket.

Dans cette version, un processus maître est chargé de répartir le nombre de points à générer entre plusieurs workers. Chaque worker effectue localement ses tirages aléatoires et renvoie au

maître le nombre de points tombant à l'intérieur du quart de cercle. Le maître agrège ensuite les résultats afin de calculer l'approximation finale de Pi.

Cette approche permet de bien illustrer le fonctionnement du modèle Master/Worker et la communication réseau via TCP. En revanche, cette version introduit un surcoût lié aux communications réseau et à la gestion des sockets, ce qui peut impacter les performances lorsque le nombre de workers augmente.

2. Version Python Socket

Cette version reproduit fidèlement l'architecture Java :

- un Master Python qui distribue les tâches
- plusieurs Workers Python exécutant le calcul Monte Carlo
- communication par sockets TCP (API python socket)

3. Version Python Monte Carlo distribué avec mpi4py

Une autre implémentation utilise le modèle SPMD (Single Program, Multiple Data) avec MPI. Dans ce cas, tous les processus exécutent le même programme, mais travaillent sur des portions différentes du calcul.

Chaque processus génère un nombre équivalent de points aléatoires, puis une réduction MPI est utilisée pour agréger les résultats. Cette approche permet de simplifier l'architecture du programme et de réduire le rôle central du maître, ce qui peut améliorer la scalabilité.

Annexe:
https://github.com/TMMproject/SAE5_RaspBerryParallel/blob/main/Documentation/Reports(CR)/rapport_Monte_Carlo.md

VII. Calcul de l'intégrale de la loi normale

A. Problématique et contexte

Ce module a pour objectif de réaliser un calcul intégral distribué sur le cluster de Raspberry Pi, en s'appuyant sur les modèles de parallélisation déjà utilisés dans le projet.

Le calcul porte sur l'intégrale de la fonction de Gauss-Laplace (loi normale), étudiée en deuxième année. Comme cette intégrale ne peut pas être calculée simplement de manière analytique, une méthode d'approximation numérique est nécessaire.

B. Méthode de Simpson

La méthode de Simpson permet d'approximer une intégrale en découpant l'intervalle [a, b] en un nombre pair n de sous-intervalles.

Le principe consiste à calculer une somme pondérée des valeurs de la fonction aux différents points de l'intervalle. Cette méthode offre une bonne précision lorsque "n" est suffisamment grand et se prête bien à une répartition du calcul sur plusieurs machines.

Dans notre cas, la fonction intégrée est la loi normale, définie par les paramètres mu (moyenne) et sigma (écart-type).

C. Implémentation parallèle

L'implémentation repose sur un modèle Master/Worker en Java utilisant des sockets TCP.

Le programme maître :

- récupère les paramètres d'entrée (a, b, n, mu, sigma).
- découpe l'intervalle global en sous-intervalles.
- distribue le travail aux workers.
- récupère les résultats partiels et calcule le résultat final.

Chaque worker calcule l'intégrale sur son sous-intervalle à l'aide de la méthode de Simpson, puis renvoie sa contribution au maître. Le calcul est entièrement parallèle et les workers fonctionnent de manière indépendante.

D. Résultats et limites

Les résultats obtenus sont cohérents avec les valeurs théoriques attendues. Par exemple, pour une intégration entre 0 et 1 avec mu = 0 et sigma = 1, le résultat est proche de 0,3413

Les principales limites de cette approche sont le coût des communications réseau et la répartition statique du travail, qui ne tient pas compte d'éventuelles différences de performance entre les machines du cluster.

Malgré cela, ce module valide la faisabilité d'un calcul intégral distribué et s'intègre correctement dans l'architecture globale du projet.

VIII. Conclusion et perspectives

Ce projet mené dans le cadre de la SAE 5 nous a permis de remplir les objectifs fixés au départ. Le cluster de Raspberry Pi a été mis en place et configuré correctement, les différents

algorithmes de calcul parallèle ont été développés et intégrés dans une interface web fonctionnelle, permettant de lancer et visualiser les résultats des calculs.

Au cours du projet, nous avons pu travailler sur des aspects variés tels que le calcul parallèle, la communication réseau avec MPI et TCP, la gestion d'un cluster, ainsi que le développement d'une application web complète. L'étude des performances à travers la scalabilité forte et faible nous a permis de mieux comprendre le comportement des algorithmes sur une infrastructure à ressources limitées.

Certaines difficultés ont été rencontrées, notamment liées aux contraintes matérielles des Raspberry Pi, aux temps de communication entre les nœuds et à l'organisation du travail entre les différents sous-projets ou bien à des bugs qui sont parfois hors de notre compréhension. Ces problèmes ont toutefois été pour la plupart surmontés grâce à des ajustements techniques et une bonne coordination au sein du groupe.

Des améliorations restent possibles, comme l'automatisation de l'installation et de la configuration du cluster, l'ajout de nouveaux modules de calcul, une meilleure supervision des ressources ou encore des optimisations réseau. Ce projet constitue ainsi une base solide pouvant être améliorée et étendue dans le futur.

IX. Annexes : Sous-rapports

I/Doc

Gantt

https://github.com/TMMproject/SAE5_RaspBerryParallel/tree/main/Documentation/Gantt_Diagram

BDD Doc

https://github.com/TMMproject/SAE5_RaspBerryParallel/tree/main/Documentation/BDD_documentation

Diagramme d'infrastructure

https://github.com/TMMproject/SAE5_RaspBerryParallel/tree/main/Documentation/Diagramme

Cahier des charges

https://github.com/TMMproject/SAE5_RaspBerryParallel/blob/main/Documentation/Requirements_Specification(CdC)/Cahier_des_charges.pdf

Analyse des besoins/ Recueil des besoins

https://github.com/TMMproject/SAE5_RaspBerryParallel/blob/main/Documentation/Need_Analysis(AdB)/Requirements_Gathering(RDB)/Requiremens_Gathering_(RDB)_SAE5_v2.pdf

Analyse des besoins/ Exigences fonctionnelles et non fonctionnelles

https://github.com/TMMproject/SAE5_RaspBerryParallel/blob/main/Documentation/Need_Analysis(AdB)/Functional-nonfunctional/Functional_Non_Functional_Requirements_SAE5_v1.pdf

Gestion des risques

https://github.com/TMMproject/SAE5_RaspBerryParallel/blob/main/Documentation/Risk_Assessment(GdR)/Gestion%20des%20risques_SAE5.pdf

Rapport NFS

https://github.com/TMMproject/SAE5_RaspBerryParallel/blob/main/Documentation/Reports(CR)/rapport_NFS_setup.md

Rapport VNC & Tailscale

https://github.com/TMMproject/SAE5_RaspBerryParallel/blob/main/Documentation/Reports(CR)/rapport_VNC_TailScale.md

Rapport RPi

https://github.com/TMMproject/SAE5_RaspBerryParallel/blob/main/Documentation/Reports(CR)website/SAE_RPi_Report.pdf

II/Scripts

Rapport Prime Improve

https://github.com/TMMproject/SAE5_RaspBerryParallel/blob/main/Documentation/Reports(CR)/rapport_prime_improve.md

Rapport Prime Load Balancing & Crible d’Ératosthène

https://github.com/TMMproject/SAE5_RaspBerryParallel/blob/main/Documentation/Reports(CR)/rapport_Load_Balancing_et_Crible_d'Ératoshène.md

Rapport Monte Carlo

https://github.com/TMMproject/SAE5_RaspBerryParallel/blob/main/Documentation/Reports(CR)/rapport_Monte_Carlo.md

Rapport Paradigme & TCP

https://github.com/TMMproject/SAE5_RaspBerryParallel/blob/main/Documentation/Reports(CR)/Rapport%20_Comparaison_des_paradigmes_de_parallélisme_(Analyse_TCP).md

Rapport Website

https://github.com/TMMproject/SAE5_RaspBerryParallel/blob/main/Documentation/Reports(CR)/SAE_Website_Report.pdf

Rapport Scalabilité

https://github.com/TMMproject/SAE5_RaspBerryParallel/blob/main/Documentation/Reports(CR)/SAE_Rapport_Scalabilité.pdf