UVSQ université PARIS-SACLAY

IUT de Vélizy-Rambouillet
CAMPUS DE VÉLIZY-VILLACOUBLAY
CAMPUS DE RAMBOUILLET

Dossier de spécifications

V 01.00

Guenfici Rayane | Chedozeau Mathieu | Renouf Ugo | Belaidi Elyas | Bullock Patrick

1. Introduction

1.1 Contexte du projet

Ce projet se constitue d'un groupe de cinq personnes en autonomie dont l'objectif est de créer et de développer un site internet, permettant d'exécuter des modules déployés sur un cluster de 5 Raspberry non configurés initialement.

1.2 Objectifs du dossier de spécifications

Le dossier de spécifications a pour objectifs de rassembler et d'expliquer les exigences fonctionnelles et non fonctionnelles auxquelles nous devrons répondre au travers de ce projet.

1.3 Périmètre du projet

Le produit final du projet doit être un serveur web hébergé sur un cluster de Raspberry pi capable d'effectuer du calcul distribué, ainsi que des prédictions à l'aide d'un modèle de machine learning. Le projet prend fin à la mi-mars et se conclut par une soutenance.

2. Description générale du projet

2.1 Présentation du projet

Ce projet vise à développer une application web et à effectuer son déploiement sur un microordinateur en autonomie. De plus, les fonctions de l'application doivent intégrer du calcul distribué entre les différents composants du microordinateur qui sera à configurer au préalable.

2.2 Besoins et attentes des utilisateurs

Les utilisateurs doivent pouvoir accéder au serveur web et utiliser les modules développés dans l'application grâce à un compte utilisateur, pour effectuer des calculs ou des prédictions à l'aide d'un modèle de machine learning. Les calculs doivent être effectués de manière distribuée.

2.3 Contraintes et limites du projet

Ce projet se résume à plusieurs contraintes fonctionnelles et non fonctionnelles (voir grand 3). Ces contraintes visent à respecter un objectif précis, certaines contraintes sont inchangeables et les limites ne peuvent pas être modifiables.

3. Spécifications fonctionnelles

3.1. Exigences fonctionnelles

3.1.1 Liste des fonctionnalités principales

- Réalisation de calculs parallèles sur un cluster de Raspberry Pi.
- Mise en place d'une interface web avec laquelle les utilisateurs interagirons.
- Affichage des résultats sous forme de graphiques et de données.
- Génération de modèles optimisés pour la prédiction de données et affichage des prédictions.

3.1.2 Cas d'utilisation

- Simulation de calculs parallèles : Permet à l'utilisateur de lancer des simulations de calculs parallèles depuis la page dédiée a cet effet.
- Visionner les statistiques des simulations : Autorise le gestionnaire à visualiser les statistiques d'utilisation des modules.

3.1.3 Scénarios d'utilisation

- **Simulation de calculs parallèles :**
    1. L'utilisateur se rend sur la page d'accueil.
    2. Navigue vers la section de simulations de calculs parallèles.
    3. Sélectionne la simulation souhaitée.
    4. L'utilisateur se connecte.
    5. Entre les paramètres souhaités.
    6. Lance la simulation en cliquant sur le bouton "lancer."
    7. Visualise les résultats.

- **Visionner les statistiques des simulations :**
    1. L'utilisateur se rend sur la page d'accueil.
    2. L'utilisateur se connecte.
    3. Le gestionnaire accède à la page dédiée aux statistiques.
    4. Visualise les statistiques d'utilisation des modules.

3.2. Exigences non fonctionnelles

- Le système doit fonctionner sur le "kit cluster hat" avec 4 Pi Zero w et 1 Pi 4B.
- Utilisation de GitLab pour le suivi et le versionnage du projet.
- Conception graphique à l'aide de Figma et de la suite Adobe.
- Développement avec la suite de logiciels JetBrains, VisualCode, SublimeText.
- Exécution de code Python et PHP sur le cluster.
- Utilisation de MPI pour le calcul distribué.
- Affichage des résultats des calculs Python sur les pages web.

3.2.1 Performances attendues

Le système devra garantir des performances optimales pour assurer une exécution rapide des calculs parallèles. Les temps de réponse de l'interface web et la génération de modèles prédictifs doivent être suffisamment rapides pour assurer une expérience utilisateur fluide. De plus, le calcul distribué sur le cluster de Raspberry Pi doit être efficace, en minimisant les temps d'attente et en assurant une utilisation optimale des ressources.

3.2.2 Sécurité

La sécurité est une priorité pour garantir l'intégrité des données et la confidentialité des utilisateurs. Les mesures de sécurité incluent la gestion des clés SSH entre les nœuds du cluster, l'application de bonnes pratiques de sécurité pour le stockage et le transfert des données, ainsi que la protection contre les vulnérabilités potentielles. De plus, l'accès aux statistiques d'utilisation des modules doit être contrôlé et limité au gestionnaire.

3.2.3 Fiabilité

Le système doit être fiable pour garantir une exécution stable des calculs parallèles et la disponibilité constante de l'interface web. Les programmes de calcul distribué doivent être robustes, capables de gérer d'éventuelles erreurs et de récupérer rapidement en cas de dysfonctionnement.

3.2.4 Ergonomie

L'application doit être conviviale, avec une navigation intuitive permettant aux utilisateurs de trouver facilement les fonctionnalités. Les résultats des calculs, présentés sous forme de graphiques et de données,

doivent être clairs et compréhensibles. De plus, le processus de lancement de simulations de calculs parallèles doit être simple et bien guidé pour les utilisateurs.

4. Architecture du système

4.1. Architecture globale

Le projet sera déployé sur un cluster de Raspberry pi composé de 4 Raspberry pi 0 w ainsi que d'un Raspberry pi 4 b. Le cluster hébergera un serveur apache qui servira d'interface pour le lancement des modules logiciels.

4.1.1 Présentation des composants majeurs

Voir le *rapport d'installation du cluster*.

4.1.2 Interactions entre les composants

Le Raspberry pi 4 est doté de la technologie NAT qui lui permet d'attribuer aux autres Raspberry une adresse IP, créant un sous-réseau dans lequel les 5 machines peuvent communiquer. Lors de la configuration initiale du cluster, le protocole SSH a été activé, de sorte que les Raspberry puissent accéder de l'un à l'autre aisément. Cela a aussi servi de travail préliminaire dans la mise en place du protocole MPI qui utilise SSH pour permettre le calcul distribué. Certains programmes utilisés dans le projet seront amenés à utiliser ce protocole pour répartir les tâches entre les machines.

4.2. Architecture logicielle

4.2.1 Description des modules/logiciels

En plus de reprendre les modules de l'année dernière, qui faisaient du calcul de probabilités suivant la loi normale et du web scraping, nous ajouterons un module capable de donner tout les nombres premiers entre 1 et le paramètre n donné, ainsi qu'un module faisant des calculs suivant la méthode de monte carlo. Les deux derniers seront effectués en dérivé, de sorte à ce que la charge soit répartie sur les différents processeurs disponibles. Le module de web scraping quant à lui recevra une mise à jour contenant une nouvelle fonctionnalité.

4.2.2 Interfaces entre les modules

Un site web servira d'interface entre les modules, permettant à l'utilisateur d'accéder aux différentes fonctionnalités du projet.

5. Spécifications techniques

5.1. Langages et technologies

5.1.1 Choix des langages de programmation

Ce projet nécessite l'utilisation de différents langages pour assurer le développement back-end et front-end.
Pour le développement front-end, les langages utilisés sont :

- Le **HTML** pour créer la structure des différentes pages .html et .php.
- Le **CSS** pour ajouter le style basé sur la charte graphique
- Le **JavaScript** pour gérer les différentes interactions utilisateur et créer les différentes animations.

Le développement back-end lui sera assuré par :

- Le **PHP** pour gérer les sessions utilisateurs et admin et pour permettre la compilation et la lecture de structures .html sur le RPI.
- Le **python** pour créer les différents scripts et algorithmes des différents modules.
- Le **JSON** pour stocker de manière efficace et personnalisée des données.

5.1.2 Utilisation de frameworks, bibliothèques, etc.

Pour ce qui est des librairies, plusieurs sont utilisées dans nos scripts python.
Dans celles-ci, on retrouve :

- **Numpy** : librairie très utilisée pour les traitements et le calcul scientifique. Ici, nous allons l'utiliser pour générer des données aléatoires, créer des listes adaptées et autres.

- **matplotlib** : librairie qui permet la représentation graphique.

- **random** : sélectionner de nombres aléatoirement

- **spicy.stats** (from **norm**) : librairie pour effectuer des tests statistiques

- **MPI4py** : librairie permettant d'utiliser le protocole MPI sur python. Ici, elle servira à mettre en place le calcul distribué.

- **Request** : Librairie qui permet d'effectuer le protocole HTTP à travers un code python

- **BeautifulSoup** : librairie qui aide la lecture syntaxique de documents HTML et XML.

- **webdriver** : classe de la librairie **selenium** qui autorise interaction avec différents navigateurs web (Chrome, Firefox, Safari, etc). La librairie **selenium** permet d'automatiser des navigateurs web à l'aide d'une interface.

- **time :** librairie qui donne accès à différentes fonctions pour manipuler du temps.

5.2 Base de données

5.2.1 Structure de la base de données

5.2.2 Schéma relationnel

6. Gestion de projet

6.1 Planning prévisionnel

6.2 Ressources nécessaires

6.3 Risques identifiés et stratégies d'atténuation

7. Tests

7.1 Plan de tests

Les tests sont organisés en deux catégories principales, chacune correspondant à un module spécifique du projet. La procédure effectuée répond au système de test unitaire. Le nombre de modules peut augmenter.

Module 1 : Calcul de Nombres Premiers

Test n°1 : Calcul de l'effectif des nombres premiers de 1 à N

Module 2 : Probabilité de Loi Normale

Test n°2 : Calcul utilisant la méthode de Monte-Carlo pour probabilité de loi normale

Chaque test comporte plusieurs cas de test avec des variables différentes pour évaluer la fiabilité et la précision du module

7.2 Scénarios de tests

Module 1 : Calcul de Nombres Premiers

- Scénario 1 : Calcul de l'effectif des nombres premiers avec N = -1
- Scénario 2 : Calcul de l'effectif des nombres premiers avec N = 0
- Scénario 3 : Calcul de l'effectif des nombres premiers avec N = 1
- Scénario 4 : Calcul de l'effectif des nombres premiers avec N = 2
- Scénario 5 : Calcul de l'effectif des nombres premiers avec N = 35
- Scénario 6 : Calcul de l'effectif des nombres premiers avec N = 472

Module 2 : Probabilité de Loi Normale

- Scénario 1 : Calcul de probabilité de loi normale avec moyenne (Moy) = 0, écart type (σ) = 1, et seuil (T) = 0, le résultat vient du calcul : R = P(X<T)
- Scénario 2 : Calcul de probabilité de loi normale avec Moy = 0, σ = 1, et T = 1.2
- Scénario 3 : Calcul de probabilité de loi normale avec Moy = 2, σ = 4, et T = 3.5
- Scénario 4 : Calcul de probabilité de loi normale avec Moy = -1, σ = 0.5, et T = -1.4
- Scénario 5 : Calcul de probabilité de loi normale avec Moy = 6, σ = 20, et T = 33
- Scénario 6 : Calcul de probabilité de loi normale avec Moy = 0, σ = 50, et T = 75

7.3 Critères de validation

Chaque test est considéré comme réussi si les résultats obtenus par le module correspondent de manière précise aux résultats attendus qui sont obtenues via un logiciel exterieur fiable. La comparaison se fera à l'échelle nécessaire pour garantir une fiabilité et une précision suffisantes.
Le dossier de tests est disponible pour référence, détaillant les scénarios, les résultats attendus, et les résultats obtenus pour chaque test.

8. Documentation

8.1 Manuel utilisateur