IUT Colmar

# SAE 3.02

Rapport Final

Marius Keltz
02/01/2025

# Sommaire

# 1. Introduction

Ce rapport détaille le projet de mise en place d'un système de calculs distribués comprenant un client, un serveur maître et plusieurs serveurs esclaves. L'objectif était de concevoir une architecture répartie capable de traiter des tâches en parallèle, tout en gérant les pannes et les files d'attente pour optimiser les ressources.

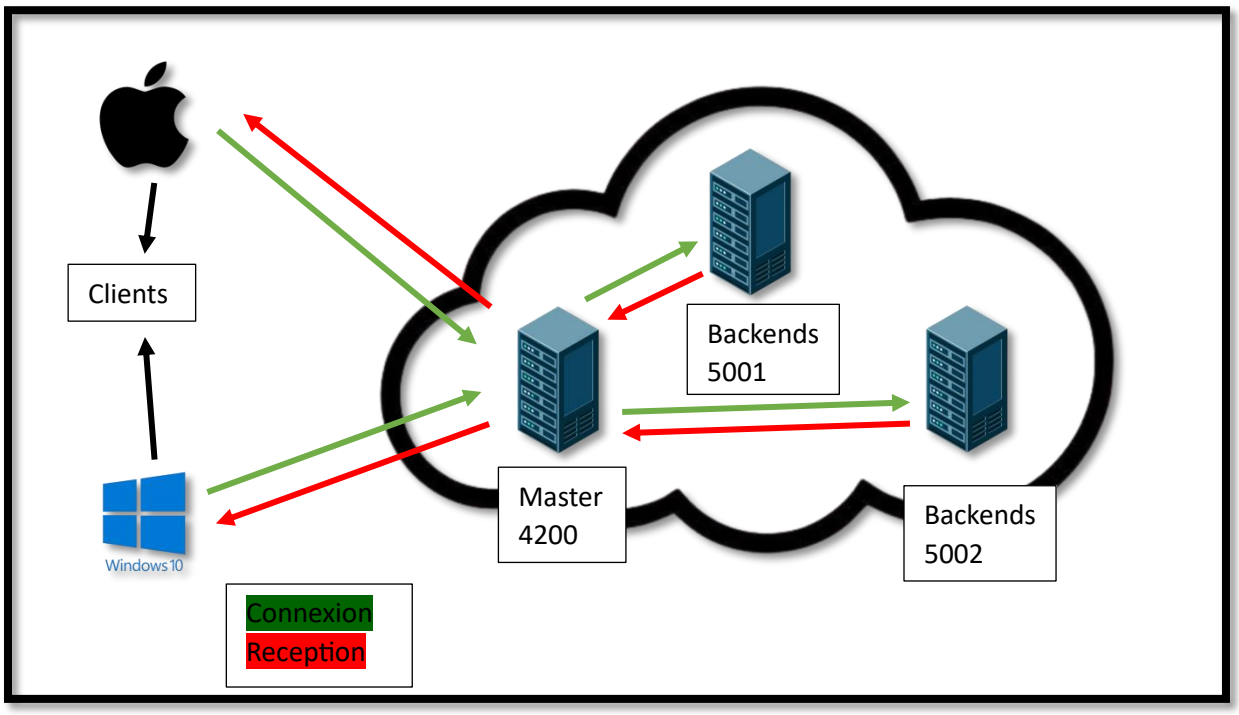
# 2. Architecture Globale

## 2.1 Description

L'architecture est divisée en trois composants principaux :

- Serveur maître
    - Reçoit les codes du client
    - Redistribue les codes du client aux serveurs esclaves
    - Monitoring du Cluster
    - Authentification
    - Robustesse
- Serveur Esclave
    - Compile et exécute
    - Reçoit du code à exécuter
    - Renvoie le résultat du code exécuté
    - Envoie la charge d'utilisation du CPU
- Client
    - Envoie des programmes
    - Reçoit les résultats des calculs

## 2.2 Schéma de l'Architecture



## 3. Architecture du GitHub :

```
Cluster-de-Calculs
├── R3.09/
│   ├── Exception/
│   │   ├── Ex1.py
│   │   ├── Ex2.py
│   │   ├── file.txt
│   │   └── lecture_seule.txt
│   ├── Les Interfaces Graphiques/
│   │   ├── Ex1_affiche_simple.py
│   │   ├── Ex1_affiche_simple_amelioré
│   │   └── Ex2_conversion.py
│   ├── Socket/
│   │   ├── Ex1_Socket_simple.py
│   │   ├── Ex2_un_chat_synchrone.py
│   │   └── Ex3_un_chat_asynchrone.py
│   ├── TP Test/
│   │   ├── Omar/
│   │   │    ├── Client.py
│   │   │    └── Server.py
│   │   ├── client.py
│   │   └── serveur_tchat.py
│   └── Threads/
│        ├── .venv/
│        ├── Ex1.py
│        ├── Ex2.py
│        └── Ex3.py
├── SAE2.02/
│   ├── Sujet_simple/
│   │   ├── client.py
│   │   └── serveur.py
│   ├── Sujet_Complex/
│   ├── client.py
│   ├── serveur_maitre.py
│   ├── serveur_esclave1.py
│   ├── serveur_esclave2.py
│   ├── documentation/
│   │   ├── Rapport Final.docx
│   │   ├── Rapport Final.pdf
│   │   ├── Documentation d'installation et d'utilisation.docx
│   │   └── Documentation d'installation et d'utilisation.pdf
│   ├── sphinx/
│   ├── Windows.mp4
│   ├── Linux.mp4
│   └── 10_serveurs_esclave.mp4
├── C.c
├── README.md
├── java.java
└── py.py
```

Le dossier Cluster-de-Calculs accueille le module R3.09, avec tous les TP effectués durant les cours de TD et TP. Ainsi que le module SAE 2.02 avec le code source du serveur maitre, du client, et les deux serveurs esclaves. Vous allez retrouver une vidéo de présentation du projet ainsi que différentes documentations. Dans ces différentes documentations, il y a un fichier Word présentant les différentes fonctionnalités, un schéma de l'architecture de la SAE, un planning de gantt. Pour la documentation du code, j'ai utilisé sphinx et enfin un fichier Word pour conclure ma SAE.

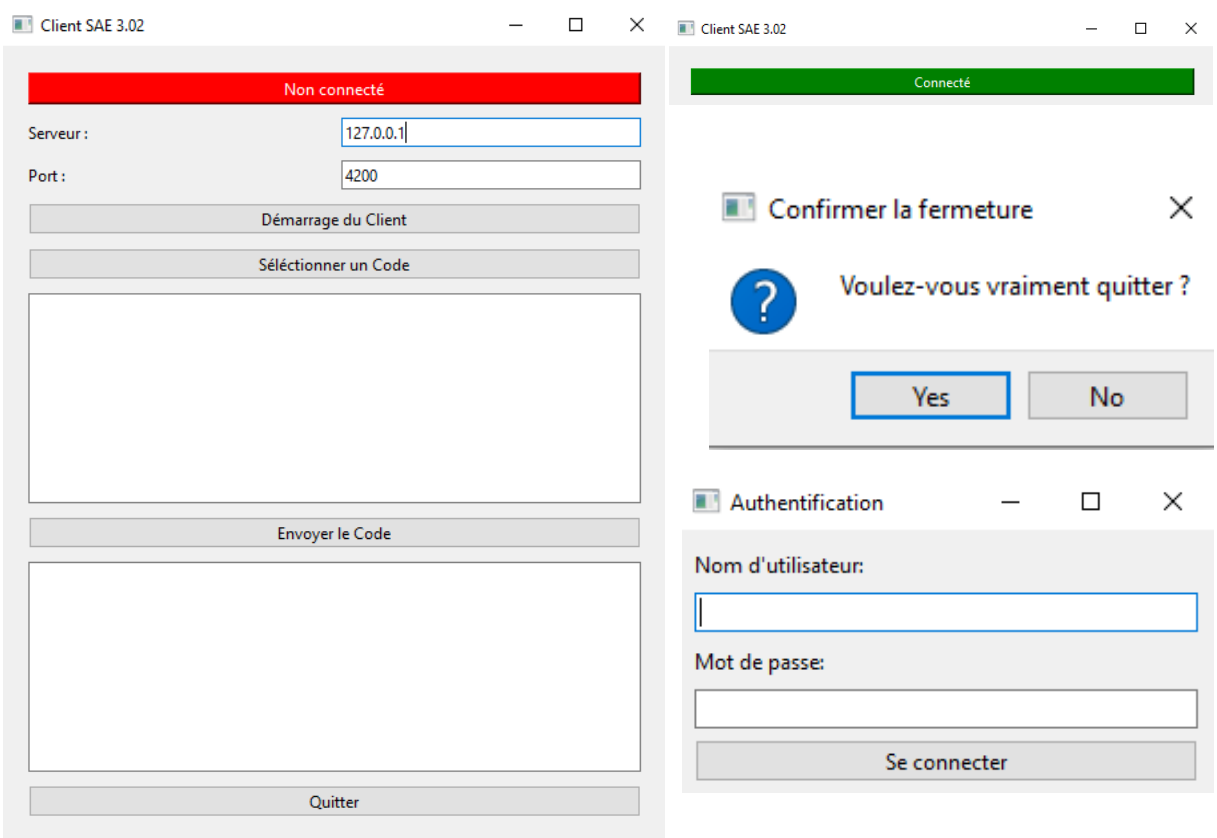
# 4. Choix

Durant ce projet, j'ai choisi d'utiliser un serveur maitre qui n'exécute aucun code. Mais qui le redistribue à d'autres serveur esclaves. Le serveur maitre fait office d'intermédiaire, il reçoit les programmes que le client lui envoie puis l'envoie par la suite au serveur esclave. Il y a deux serveurs esclaves, un exécute seulement du python et du C tandis que l'autre exécute du python et du java. Donc le serveur maitre redirige le message du client au serveur dédié.

# 5. Fonctionnel

## 5.1 Client

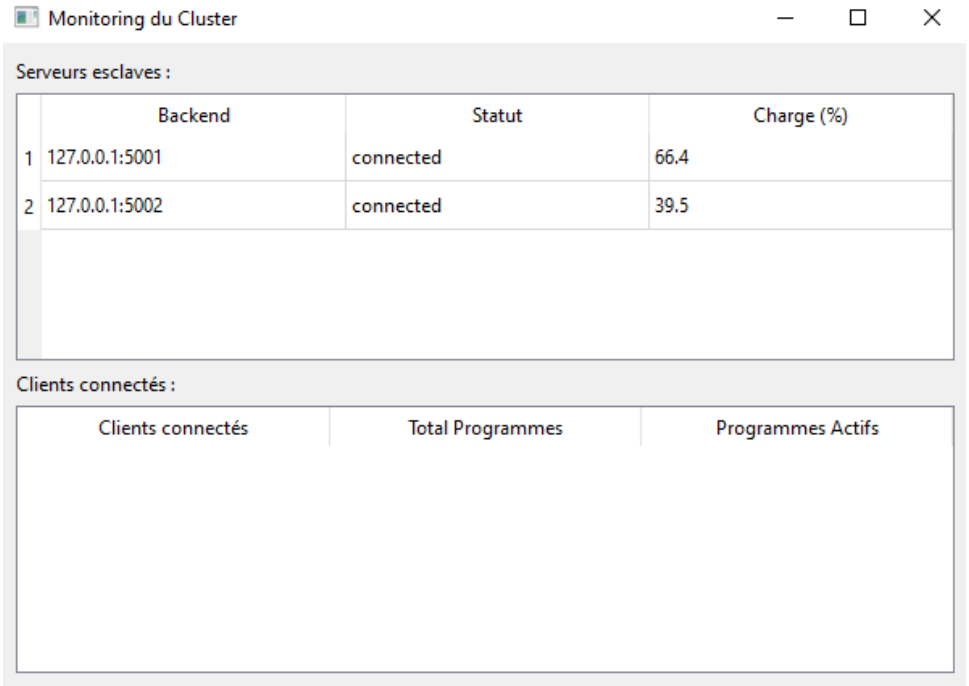
Dans mon projet, j'ai réussi à faire fonctionner le client avec une interface graphique qui me parait intuitive, on peut apercevoir en rouge que le client n'est pas connecté au serveur. Nous pouvons modifier l'adresse IP du serveur sur lequel nous souhaitons nous connecter ainsi que son port. Lorsque qu'on se connecte au serveur, ce qui était en rouge passe en vert. On peut sélectionner un programme avec le bouton « Sélectionner un Code », qui s'affiche dans le carré blanc, on peut aussi le modifier. Lorsqu'on envoie le code au serveur, il nous renvoie soit une erreur soit le résultat du code dans le carré blanc tout en bas. Enfin, un bouton quitter qui permet de fermer l'application proprement, avec une fenêtre de confirmation. Au niveau de la sécurité, le client à une fenêtre d'authentification avant d'accéder à l'interface du client, le login est client et le mot de passe client.



## 5.2 Serveur maitre :

Malheureusement je ne suis pas parvenu à faire du Load balencing, mais cependant je pense quand même qu'il y ait un cluster, car les programmes sont redistribués aux serveurs esclaves, le serveur maitre n'exécute aucun programme. Les serveurs esclaves sont capables d'exécuter du code en python, en java, en C++ et en C. Le serveur parvient à gérer correctement les erreurs lorsqu'il n'y a aucun esclave donc l'absence d'interpréteur et compilateur, ou si le code n'est pas bon. Il va y avoir un message d'erreur l'indiquant. Je ne suis pas parvenu à faire fonctionner la communication avec plusieurs clients. Depuis que j'ai fait l'interface du serveur maître, la gestion de plusieurs clients ne marche plus. Le serveur maitre est équipé d'un monitoring du cluster, il a une interface graphique qui permet de voir les clients connectés avec le nombre total de programmes exécutés et le nombre de programmes en cours, ainsi que les serveurs esclaves connectés avec le statut du serveur et la charge du CPU.

Je ne suis donc pas parvenu à faire le Load balencing, et la gestion de plusieurs clients. J'ai essayé de faire les fonctionnalités supplémentaires tel que la sécurité avec le cryptage et la persistance de données, qui est le fait qu'en cas de défaillance d'un serveur, les tâches non accomplies doivent pouvoir être récupérées et relancées sur un autre serveur. Je n'y suis pas parvenu, le cryptage était trop compliqué, lorsque je regardais sur wireshark, je voyais quand même les messages transités malgré de nombreuses tentatives. Concernant la Persistance des données, je pense que j'essayais de faire quelque chose de trop compliqué, je voulais faire en sorte que le client envoie un programme, puis que le serveur maître le garde dans une liste. Seulement j'ai eu un problème avec l'interface graphique, qui ne faisait que crasher lorsqu'un code était en attente, et le serveur maitre ne renvoyait pas le code gardé en mémoire au serveur esclave pour qu'il l'exécute. Je n'ai pas essayé le démarrage automatique de serveurs, je trouvais ça beaucoup trop compliqué à faire. Au niveau de la robustesse du projet, vous pouvez très bien lancer 10 serveurs esclaves en même temps, il faut que les serveurs soient sur les ports 5001 à 5012, vous pouvez le voir sur la vidéo.



## 5.3 Serveur Esclave :

Le serveur esclave attend la connexion du serveur maître, puis une fois que le serveur maitre est connecté, il reçoit des messages STATUS. Lorsqu'il reçoit les messages statuts, il renvoie qu'il est connecté ainsi que la charge du CPU utilisé. Le client envoie un programme à exécuter au serveur maître qui le renvoie au serveur esclave qui peut exécuter le langage de programmation du programme envoyé par le client. Enfin le serveur esclave exécute et compile le programme puis le résultat au serveur maître, qui le renvoie au client et l'affiche. Il existe deux sortes de serveur esclaves, un serveur esclave exécute du python, du C et du C++ tandis que l'autre exécute du python et du java.

# Conclusion :

Je trouve que le projet est très intéressant, j'ai adoré travailler dessus, j'ai appris beaucoup de chose concernant la programmation et la compilation de programme. Les Fonctionnalités supplémentaires proposées sont super, ce qui nous permet d'avoir un projet fonctionnel et très fourni.

Personnellement, je ne me trouve pas très compétent en programmation et pour effectuer ce projet, j'ai travaillé environ 50h. J'ai pris énormément de temps à faire le projet simple pour avoir un client et un serveur solide pour le début. Je trouve que mon travail est bien en général mais j'aurais aimé qu'il soit plus abouti. Je tenais également à m'excuser pour le retard de mon rendu sachant qu'on avait un mois pour faire ce travail, mais des problèmes de santé sont venus perturber mon avancée. De plus, je pense que je me suis mal organisé en termes de travail, j'ai passé beaucoup trop de temps sur des fonctionnalités supplémentaires tandis que je n'avais pas fini les fonctionnalités de base.