Gregory Runser

Univeristé de haute alsace, iut de colmar

Rapport final

Cluster de calcul – sae3.02 - BUT reseaux et telecomunication

# Préambule

Ce document a pour objectif de détailler l’architecture globale : ses fonctionnalités, le choix des technologies (Sécurité, Librairies), l’organisation du projet.   
J’expliquerai également les défis rencontrés, les solutions mises en œuvre, et une réflexion sur les améliorations possibles pour le système.  
Un second document détaillant la mise en place et le lancement des différentes composantes est disponible sur le répertoire GitHub.

# Sommaire

Table des matières

[Préambule 1](#_Toc185242305)

[Sommaire 1](#_Toc185242306)

[Fonctionnalités 1](#_Toc185242307)

[Elémentaire 1](#_Toc185242308)

[Avancé 1](#_Toc185242309)

[Choix des Technologies 2](#_Toc185242310)

[Socket et Thread 2](#_Toc185242311)

# Fonctionnalités

## Elémentaire

Ce projet permet, depuis un Client/Hôte, l’envoie d’un code Python, Java ou C, à un serveur qui l’exécute et renvoie le résultat.

## Avancé

Ce projet doit permettre un « Cluster de Calcul »[[1]](#footnote-1), depuis donc plusieurs clients envoyer des codes a un serveur dit « Maitre » qui exécute le code et renvoie une réponse au client.   
Cependant si le Serveur Maitre est « Surchargé », en termes de tache exécuter en simultané, il doit être capable de distribuer vers un Serveur dit « Esclave », qui prendra une charge de calcul, permettant de soulager l’intensité du serveur maitre.

# Choix des Technologies

Python permet d’exécuter une large gamme de codes, allant des plus simples aux plus complexes, en offrant une flexibilité pour différentes tâches. Dans ce projet, Python est utilisé non seulement pour gérer le lancement des interfaces clients, mais aussi pour exécuter des scripts sur les serveurs maîtres et esclaves.

Pour rendre cette solution encore plus robuste et fonctionnelle, plusieurs librairies sont nécessaires.

* Tout d'abord, Psutil est une bibliothèque essentielle pour surveiller la charge du processeur sur le serveur. Elle permet de collecter des informations détaillées sur l’utilisation du CPU, de la mémoire, ainsi que d’autres ressources systèmes, ce qui est crucial pour optimiser la performance du serveur.
* Ensuite, PyQt joue un rôle fondamental en permettant la création d’interfaces graphiques. Cette bibliothèque offre une approche structurée et modulaire pour concevoir des interfaces utilisateur, tout en permettant un design soigné et personnalisé.

L'utilisation de Socket et Thread dans un projet Python est souvent motivée par des besoins spécifiques liés à la gestion des connexions réseau et à l'exécution parallèle de tâches. Voici pourquoi ces deux concepts sont couramment utilisés :

## Socket et Thread

* L'utilisation de Socket et Thread dans ce projet Python est liés à la gestion des connexions réseau et à l'exécution parallèle de tâches.
  + Un socket permet la communication entre différents programmes, souvent situés sur des machines distinctes dans un réseau. En Python, la bibliothèque socket fournit une interface pour établir des connexions réseau en utilisant les protocoles TCP/IP, UDP, etc.

Communication entre clients et serveurs : Les sockets permettent d'établir une communication bidirectionnelle entre un serveur (qui écoute les connexions) et un client (qui initie la connexion). Cela peut être essentiel pour des applications distribuées, des services web, des systèmes client-serveur, etc.

* + Un thread permet d'exécuter plusieurs tâches en parallèle au sein du même programme. En Python, le module threading permet de créer et gérer des threads.

Les threads permettent d’effectuer plusieurs opérations simultanément sans bloquer l'exécution du programme principal. Par exemple, un programme peut gérer plusieurs connexions réseau en parallèle, où chaque thread répond à un client

* Pourquoi utiliser les deux ensembles ?

L'utilisation combinée des sockets et des threads est très courante dans les systèmes de type serveur-client. Par exemple :

Un serveur peut utiliser un socket pour écouter les connexions entrantes.

Lorsqu'une connexion est établie, un nouveau thread est créé pour gérer cette connexion spécifique, permettant au serveur de continuer à écouter d'autres connexions sans interruption.

Cela permet de gérer plusieurs connexions simultanément tout en maintenant la réactivité et l'efficacité du serveur.

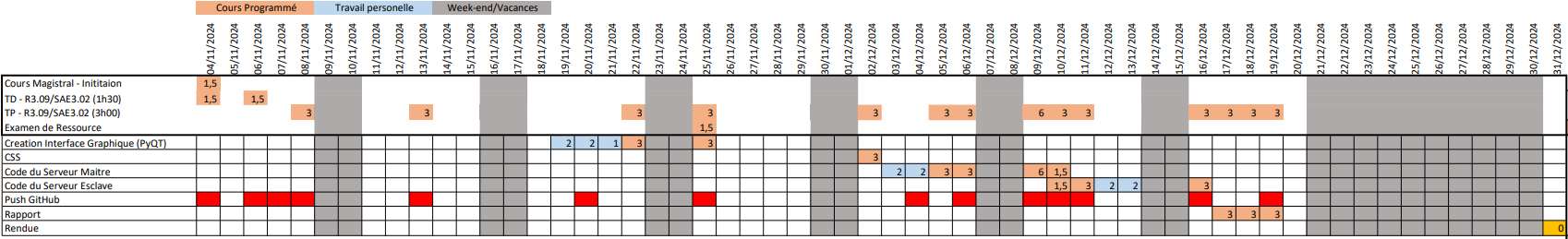
## Aspect lié à la sécurité

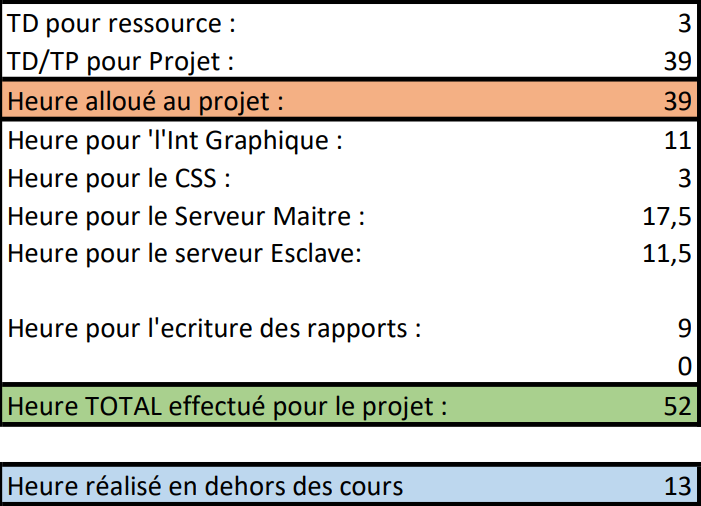
La sécurité est un aspect crucial dans la communication entre un serveur et un client.  
En effet, bien qu'il soit possible d'implémenter une authentification dans un projet open source, cela peut devenir complexe, notamment parce que les informations peuvent être visibles en clair. La bibliothèque hashlib permet de sécuriser un mot de passe en le transformant à l'aide d'un algorithme de hachage, mais ce procédé, est facilement réversible.

Cependant, pour renforcer la sécurité, il est essentiel de ne pas seulement protéger les mots de passe, mais aussi de sécuriser les données échangées entre le client et le serveur. Cela peut être fait en utilisant des mécanismes de chiffrement, tels que des certificats SSL/TLS, qui garantissent la confidentialité des informations transmises. Le chiffrement via un certificat permet de sécuriser les échanges et d’assurer que les données ne soient pas interceptées ou altérées pendant leur transit.

# Organisation

## Diagramme de GANTT





Le nombre d’heure alloué a ce projet est conséquent, mais il m’a beaucoup plus, notamment sur la création de l’interface graphique, j’ai effectué des heures en dehors des heures dédié a la SAE pour avancé et avoir un rendue fonctionnelle, esthétique

et développé.

1. Ensemble de plusieurs ordinateurs (ou nœuds), souvent interconnectés en réseau, qui travaillent ensemble pour exécuter des tâches de calcul intensif ou répartir des charges de travail. [↑](#footnote-ref-1)