Table des matières

[Présentation de la SAE : 1](#_Toc187597382)

[Architecture Globale 1](#_Toc187597383)

[Déploiement et Tests : 1](#_Toc187597384)

[Problèmes rencontrés 2](#_Toc187597385)

[Améliorations possibles 2](#_Toc187597386)

[Conclusion : 3](#_Toc187597387)

## Présentation de la SAE :

Dans cette SAE nous avons dû répliquer l’architecture d’un cluster de server de type Maitre/Esclave qui a pour but d’exécuter du code venant d’un client et de lui renvoyer l’output.

La communication entre client/server/slaves se fera avec des socket et le serveur doit être également équippé d’un load balancing.

## Architecture Globale

L’architecture de cette sae se veut ressembler a celle d’un cluster de serveurs comme expliqué précédemment, elle comporte donc 3 codes python servant a répliquer : Le client, Le serveur maitre et son serveur Esclave.

Pour tout les codes, l’utilisation de library s’est uniquement fait avec des library déjà inclus avec python, afin de premièrement gagner en compatibilité mais également rendre le travail plus complexe et personnel.

Pour toute aide lié au déploiement de la sae, veuillez lire le README inclus dans le github.

### Déploiement et Tests :

**Déploiement,**

1. **Serveur** : Une fois le serveur configuré, vous pouvez démarrer le serveur maître qui se chargera d'écouter les connexions des clients et de gérer les serveurs esclaves.
2. **Clients** : Les clients se connectent au serveur maître via l'interface graphique et peuvent envoyer des messages ou du code.
3. **Serveurs esclaves** : Les serveurs esclaves seront exécuté par le serveur maitre a la demande du client, ils géreront : l’interprétation/compilation et l’exécution 5 fois, une fois ce nombre atteint, un nouvel esclave sera lancé qui fera la même chose.

**Tests,**

* **Tests de connexion** : Le système a été testé avec plusieurs clients connectés au serveur pour vérifier la gestion des connexions simultanées. J’ai connecté jusqu’à 6 clients (ce nombre n’est en revanche pas une limite) en même temps afin de vérifier le load balancing et la réception.
* **Tests d'envoi/execution de fichiers** : L'envoi de fichiers a été testé avec succès, mais des problèmes peuvent survenir si l'envoi est trop rapide (ex, spam envoyer sur le client).
* **Tests de déconnexion** : La fonctionnalité de déconnexion a été vérifiée pour s'assurer qu'elle fonctionne correctement.
* **Autres**: Toute autres fonctionnalité cruciale a cette SAE a été testé et est normalement fonctionnel.

### Problèmes rencontrés

1. **Problèmes liés au client/s** : Probablement le problème qui est arrivé le plus souvent. La mise en place de la connexion simple était relativement facile mais la connexion multiple a été probablement ce qui m’as pris le plus de temps, non pas dans sa création qui a été relativement si3mple mais plutôt dans sa robustesse qui était désastreuse  ! En effet après n’importe quel changement (pas non plus tout mais une grande partie) il fallait vérifier que l’envoi, la transmission et la réception se faisait correctement et dans la majeure partie des cas ceci n’était plus la cas après un changement. A part ça j’ai eu énormément de problèmes liés a la stabilité du client, ce dernier subissais des crash assez fréquemment.

**Explication**, il n’y en a pas vraiment, c’est surtout qu’il a fallut adapter des fonctions qui n’était pas adapté a de futur changements trop complexe. Pour ce qui est des crash je n’en ai pas non plus actuellement, précédemment j’ai réussis a régler ces problèmes qui était généralement lié au sockets ou au threading mais actuellement un crash lié au parsing peut arriver de manière aléatoire, je n’ai ni la cause ni la solution à ce problème.

1. **Load balancing** : Normalement je n’ai pas a me plaindre vis-à-vis du load balancing car j’ai choisi la manière la plus simple de faire, timeout du serveur esclave après 5 exécutions. Cependant ça a été bien plus difficile que je ne l’aurais pensé !

**Explication,** Concrètement une fois que le client/server maitre et esclave était au minimum fonctionnel sur l’envoi, réception et exécution je me suis mis a travailler sur le load balancing. Cependant pendant mes changements sur le serveur maitre je me rendais peu a peu compte que le code du serveur esclave ne ferait juste pas du tout l’affaire, j’ai du donc presque recommencer ce code du début afin de pouvoir tout implémenter de la manière que je voulais et ne pas me forcer a adapter mes changements sur le serveur maitre.

1. **Problèmes d’exécutions :** Premier problème, les fichiers avec espaces. Problème tout simple mais qui m’as quand même demandé de modifier pas mal de choses. Problèmes d’accès refusé, probablement le plus étrange car je n’ai jamais eu ça dans d’autre projet de programmation mais tout simplement certains fichier après un certain temps me mettait une erreur «accès refusé » après l’exécution, je n’ai absolument aucune idée de la cause (pas un problème de permission bizarrement) mais le fix a été le même dans tout les cas : supprimer et recréer un fichier a exécuter.

### Améliorations possibles

1. **Optimisation de la gestion des threads** : Actuellement, chaque action comme l'envoi de message ou de fichier crée un nouveau thread. Cela pourrait être amélioré en optimisant la gestion des ressources et des threads.
2. **Sécurisation des communications** : Ajouter un chiffrement pour sécuriser les communications entre le client et le serveur, afin d'éviter toute interception ou altération des données.
3. **Amélioration de l'interface utilisateur** : L'interface graphique pourrait être améliorée pour offrir plus de fonctionnalités et de convivialité, comme des indicateurs de progression lors de l'envoi de fichiers.
4. **Ou d’autres tel que : Interface web, Nouveaux languages, monitoring du cluster ou généralement une plus grande scalabilité/robustesse.**

## Conclusion :

J’ai beaucoup apprécié travailler sur cette sae bien que je pense que ça aura été une des plus difficile pour moi jusqu’à présent. Le fait de ne pas avoir accès a des libraires extérieures a ce qui est proposé de base avec python nous a forcer a développer des solutions nous même et je pense que c’était une très bonne idée pour faire en sorte de structurer notre travail. Également j’ai pu remarquer qu’axer son code sur la scalabilité dès le départ devrait être primordial car c’est probablement ce qui m’as causé le plus de problèmes.

Table des matières

[Présentation de la SAE : 1](#_Toc187597382)

[Architecture Globale 1](#_Toc187597383)

[Déploiement et Tests : 1](#_Toc187597384)

[Problèmes rencontrés 2](#_Toc187597385)

[Améliorations possibles 2](#_Toc187597386)

[Conclusion : 3](#_Toc187597387)