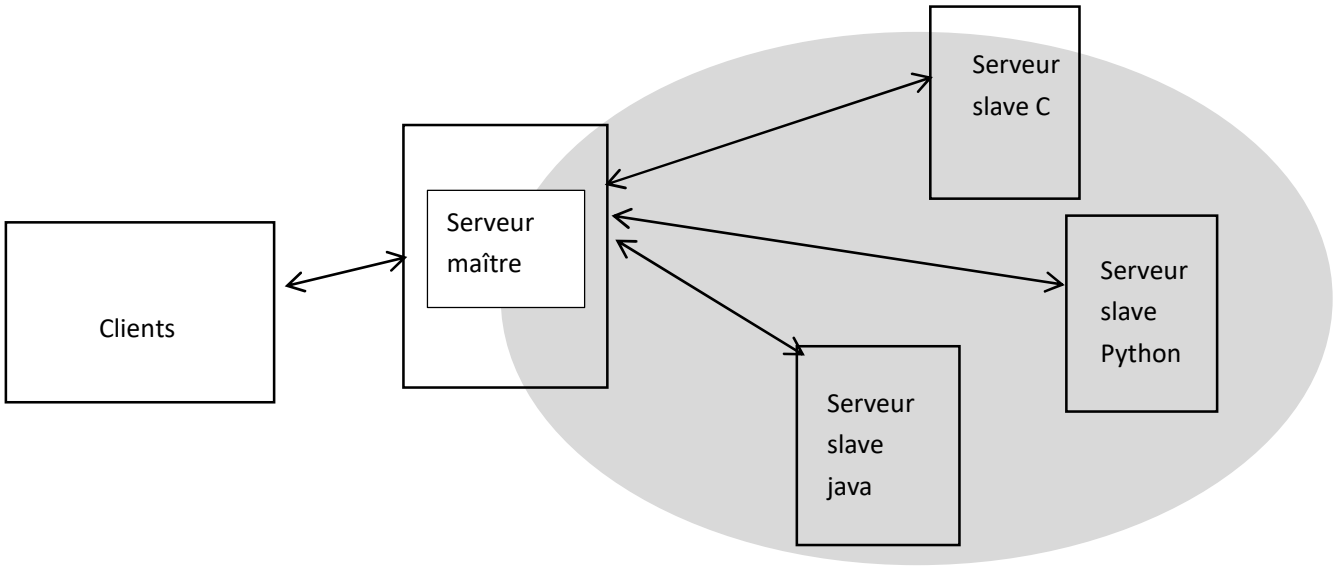
Document d'installation

1. Introduction :

Dans ce document d'installation nous allons voir comment, à partir du dépôt Github, installer et mettre en place l'ensemble de l'environnement de test du cluster de calcul effectué durant cette SAE, dans un premier temps nous allons voir l'installation du serveur maître , puis ensuite celle des différents serveurs esclaves, et pour finir les clients.
Une fois cette architecture mise en place, on pourra voir le fonctionnement et les fonctionnalités de chacun.

Pour cette installation nous allons aussi avoir besoin de plusieurs machines virtuelles inter-connectées entre elle (dans mon cas elles étaient toutes en bridge sur mon réseau personnel), les clients eux étaient sur ma machine physique mais elle peuvent aussi être implémentés dans une machine virtuelle si besoin.

On obtient donc finalement une architecture telle que :



2. Installation serveur maître

Désormais nous allons donc mettre en place notre serveur principal, celui-ci sera sur une machine virtuelle Linux (Debian 12).
Dans un premier temps, nous allons installer Python au cas où ce n'est pas encore fait, pour cela il nous faut donc ouvrir un terminal et entrer la liste des commandes suivantes :

sudo apt update
sudo apt install python3
sudo apt install python3-pip
sudo apt install python3-venv

Ensuite nous allons créer notre repertoire de test :

mkdir SAE
cd SAE
python3 -m venv venv

L'environnement virtuel est ici nécessaire afin de pouvoir faire tourner le programme car il nécessite des librairies extérieurs, ici il nous faut donc installer Pyqt6 à l'aide de l'outil «pip» :

source venv/bin/activate
python3 -m pip install PyQt6 *(Cette commande peut parfois prendre un peu de temps)*

Une fois ceci terminer il nous faut ajouter nos fichiers nécessaire dans notre repertoire, on récupère donc les fichiers master.py et liste_serveur.txt après avoir télécharger le fichier ZIP du répertoire Github, on doit avoir l'arborescence suivante :

toto@Debian12:~/SAE$ ls
liste_serveur.txt master.py venv

Afin de pouvoir lancer le serveur il faut donc entrer à nouveau (ou rester) dans l'environnement virtuel et lancer le programme comme suivant :

source venv/bin/activate
python3 master.py

L'installation du serveur maître est désormais fini.

3. Installation des serveurs esclaves

Désormais il faut mettre en place les serveurs esclaves pour assurer la compilation des 4 languages supportés par notre cluster.
Pour cela, dans un premier temps il faut aussi installer python sur chacune des machines si cela n'est pas encore fait.
On refait donc les mêmes commandes que pour le serveur maitre :

sudo apt update
sudo apt install python3
sudo apt install python3-pip

sudo apt install python3-venv

Maintenant que cela est fait, il faut de nouveau mettre en place notre repertoire de test et récuperer les fichiers nécessaires sur Github, en l'occurence ici seulement le fichier «slave.py», on veut donc obtenir en fin de compte l'arborescence suivante :

toto@Debian12:~/SAE$ ls
slave.py venv

Cette fois-ci il faudra de nouveau installer PyQt6 mais aussi en plus la bibliothèque «psutils» :

source venv/bin/activate
python3 -m pip install PyQt6
python3 -m pip install psutil

Et pour le serveur slave qui devra compiler Java, il faut aussi lui installer le kit de compilation Java avec la commande suivante (cette fois ci hors du venv) :

sudo apt install default-jdk

Pour ce qui est du C et C++ il faudra effectuer la commande suivante si le compilateur n'est pas encore installé :

sudo apt install build-essential

Il ne reste maintenant plus qu'a lancer les serveurs et c'est bon :

source venv/bin/activate
python3 slave.py

4. Installation des clients

Pour ce qui est des clients, il nous faut juste installer Python comme pour les serveurs, je remet ici les commandes pour une installation sous debian :

sudo apt update
sudo apt install python3
sudo apt install python3-pip
sudo apt install python3-venv

On fait ensuite en sorte d'avoir l'arborescence suivante :
toto@Debian12:~/SAE$ ls
client.py code_test style.qss venv

Avec le dossier code_test contenant l'ensemble des codes de test des 4 langages supportés.
On installe aussi PyQt6 :

source venv/bin/activate
python3 -m pip install PyQt6

Et on lance le programme :

source venv/bin/activate
python3 client.py

Mais les clients peuvent aussi être lancer sous d'autres systèmes d'exploitation (dans mon cas ils étaient lancés sous Windows) il suffit juste d'installer Python et de ne pas oublier de mettre interpréteur Python dans le Path des variables d'environnement afin que celui-ci soit opérationnel.

Avant de pouvoir commencer les tests, il faut encore faire une petite chose très importante, sur chaque serveur slave, il faut effectuer un «ip a» afin d'y récupérer l'adresse IP de chaque serveur, on indique ensuite chacune de ces IP dans le fichier liste_serveur.txt après le langage correspondant à chaque serveur.

L'installation et la mise en place de l'architecture de test est désormais finie !

5. Démarrage et tests

Maintenant que tout est prêt, il ne reste plus qu'a démarrer les serveurs et les clients, en 1er temps il faut lancer le serveur maître, il est le seul à pouvoir accepter les requêtes de connexions des autres, pour cela, il faudra juste cliquer sur le bouton «Démarrer le serveur», une fois fait, ce serveur ne doit plus être toucher ou éteint, ensuite il faudra connecter les serveurs esclaves (ne pas oublier de bien préciser les adresses IP dans le fichier «liste_serveur» ou il seront considérer comme des clients et vont crash. Et en dernier temps les clients peuvent être lancés à n'importe quel moment après les serveurs.

(Malheureusement il y a un petit problème que je n'ai pas su régler, si un clients se connecte avant un serveur, il ne pourra pas envoyer de codes de langage de ce serveur et cela peut causer un crash du client en question, afin de ne pas a avoir a fermé et rouvrir la fenêtre, il suffit juste de le déconnecter et le reconnecter.)
(Il y a aussi un autre problème, si un serveur esclave crash, la déconnexion de celui-ci ne s'affiche pas et il sera toujours considérer comme connecter, il faut donc le reconnecter ou redémarrer le serveur maître si cela arrive).