Compte-rendu installation MPI

1. Objectif

L'objectif est de mettre en place un **cluster de Raspberry Pi** capable d'exécuter des programmes parallèles à l'aide de **MPI (Message Passing Interface)**.
Le cluster est composé de :

- 1 **master** (Raspberry Pi 4)
- Plusieurs **workers** (Raspberry Pi Zero – architecture armv6l)

MPI est utilisé pour répartir les calculs sur plusieurs nœuds via le réseau.

2. Architecture matérielle et logicielle

2.1 Architecture

- Réseau local Ethernet / USB
- Chaque Raspberry Pi possède :
    - Une adresse IP fixe
    - Un nom d'hôte unique (`rp01`, `rp02`, etc.)

2.2 Logiciels utilisés

- OS : Raspberry Pi OS (Lite)
- MPI : MPICH
- Python MPI : mpi4py

3. Création des utilisateurs

Un utilisateur commun est créé sur tous les Raspberry Pi :

```
sudo adduser mpiuser
sudo usermod -aG sudo mpiuser
```

Tous les calculs MPI seront faits à partir de cet utilisateur. Lorsque l'on effectue une commande avec MPI avec l'utilisateur mpiuser tel que `mpirun -np 5 --hostfile hosts python3 prime.py`, en réalité, MPI fait :

```
ssh mpiuser@rp01 python3 prime.py
ssh mpiuser@rp02 python3 prime.py
```

…

Il va essayer d'appeler les programmes des workers avec le même nom, il doit donc y avoir un utilisateur commun à tous les Raspberry Pi pour qu'ils puissent communiquer. De plus, cela permet également de pouvoir limiter les actions possibles de cet utilisateur car il va falloir l'autoriser à communiquer avec les autres Raspberry Pi en ssh sans mot de passe.

4. Configuration réseau

4.1 Noms d'hôtes

Chaque Raspberry Pi possède un nom d'hôte unique :

- rp4 pour le master
- rp01, rp02, rp03 et rp04 pour les workers

4.2 Fichier `/etc/hosts`

Le fichier `/etc/hosts` est **identique sur tous les nœuds** :

```
127.0.0.1    localhost
192.168.1.85 rp4
172.19.181.1 rp01
172.19.181.2 rp02
172.19.181.3 rp03
172.19.181.4 rp04
```

5. Mise en place de SSH sans mot de passe

MPI (MPICH) utilise SSH pour lancer les processus distants.

5.1 Génération de la clé SSH (sur le master)

```
su - mpiuser
ssh-keygen -t rsa -b 4096
```

(Accepter les valeurs par défaut, sans mot de passe)

5.2 Copie de la clé vers tous les nœuds

```
ssh-copy-id mpiuser@rp01
ssh-copy-id mpiuser@rp02
```

```
ssh-copy-id mpiuser@rp03
ssh-copy-id mpiuser@rp04
```

Attention : Copie de la clé vers le master lui-même :

```
ssh-copy-id mpiuser@localhost
```

Sans cela, MPICH échoue avec :

```
Permission denied (publickey,password)
```

5.3 Vérification

```
ssh rp01
ssh rp02
ssh localhost
```

Aucune demande de mot de passe ne doit apparaître.

6. Installation de MPI (MPICH)

6.1 Installation sur tous les nœuds

```
sudo apt update
sudo apt install -y mpich
```

Vérification :

```
mpirun --version
```

7. Fichier hostfile pour MPICH

Le fichier hosts est créé **uniquement sur le master** :

```
mpiuser@192.168.1.85
mpiuser@172.19.181.1
mpiuser@172.19.181.2
mpiuser@172.19.181.3
mpiuser@172.19.181.4
```

8. Test de fonctionnement MPI

8.1 Test simple

```
mpirun -np 5 --hostfile hosts hostname
```

Résultat attendu :

```
rp4
rp4
rp4
rp4
rp01
```

ou bien

```
mpirun -np 8 --hostfile hosts hostname
```

Résultat attendu :

```
rp4
rp4
rp4
rp4
rp01
rp02
rp03
rp04
```

Les résultats peuvent varier légèrement au niveau de l'ordre de renvoi des rpi0 mais les 4 cœurs du rp4 renverront leurs résultats en premier car ils sont plus puissants.

9. Installation de mpi4py (Python + MPICH)

9.1 Dépendances

```
sudo apt install -y python3-pip python3-dev
```

9.2 Installation de mpi4py

```
pip3 install mpi4py
```

Test :

```
python3 -c "from mpi4py import MPI;
print(MPI.COMM_WORLD.Get_size())"
```

Réponse : 1

10. Exécution d’un programme MPI

Exemple :

```
mpirun -np 5 -hostfile hosts python3 prime.py 5000
```

Réponse :

```
Find all primes up to: 5000
Nodes: 5
Time elasped: 1.29 seconds
Primes discovered: 669
```