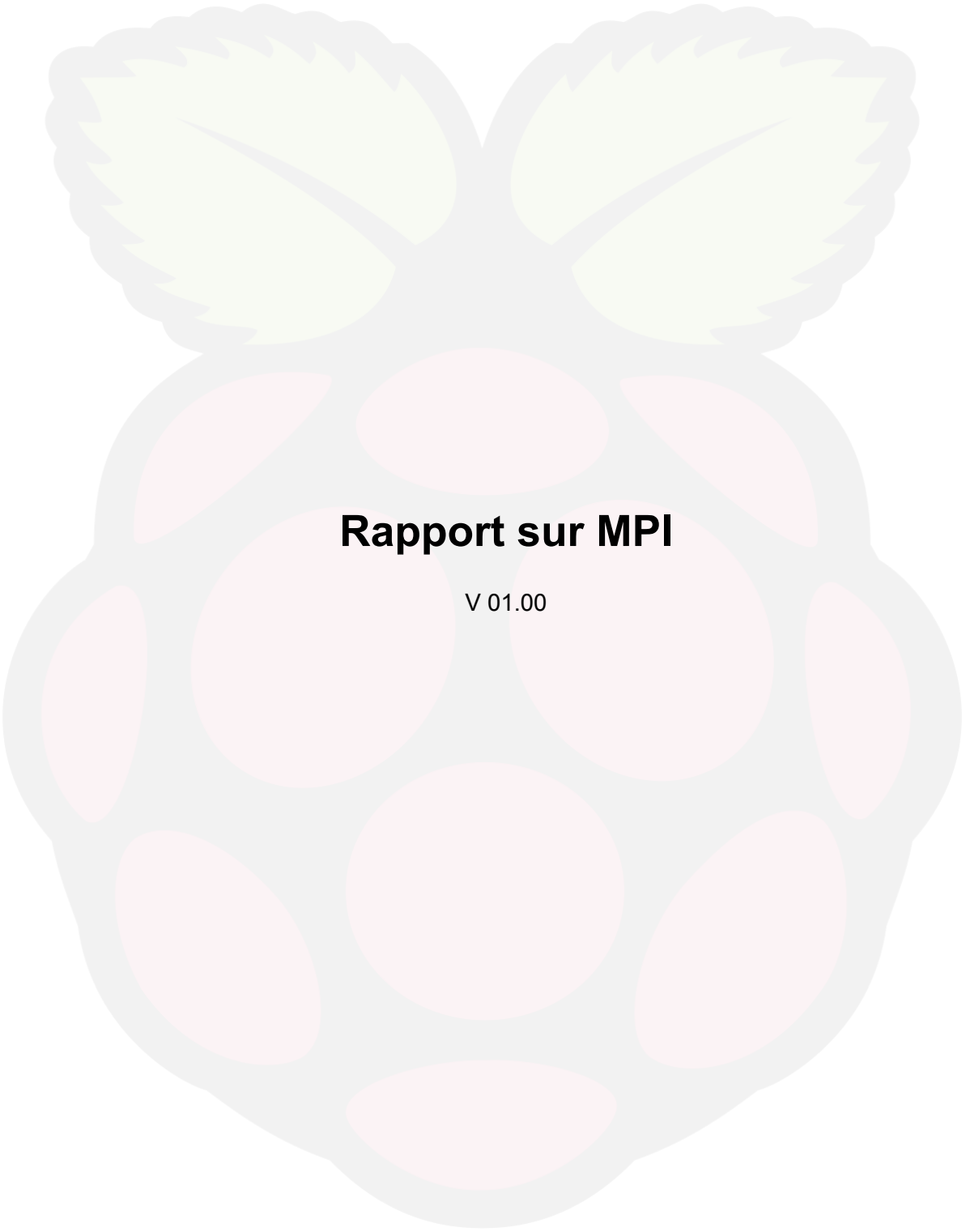
UVSQ université PARIS-SACLAY

IUT de Vélizy-Rambouillet
CAMPUS DE VÉLIZY-VILLACOUBLAY
CAMPUS DE RAMBOUILLET



Rapport sur MPI

V 01.00

Guenfici Rayane | Chedozeau Mathieu | Renouf Ugo | Belaidi Elyas | Bullock Patrick

1 - Introduction

L'objectif de ce document est de présenter le travail effectué en lien avec le protocole MPI afin de faire fonctionner les premiers corps de code en mode distribué. Dans un premier temps, nous présenterons le protocole en lui-même. Ensuite nous nous pencherons sur le cas particulier de notre projet. A l'heure de l'écriture de ce document, le seul exemple présenté sera celui du programme de calcul des nombres premiers de 1 à N. Ce document pourra évoluer au fur et à mesure du développement des modules si de nouvelles applications du protocole s'y retrouvent.

2 - Présentation du protocole MPI

MPI, ou Message Passing Interface, est un protocole servant principalement au calcul scientifique de haute performance (HPC) et dont l'utilisation permet la communication entre plusieurs machines. De cette façon, il est possible d'utiliser plusieurs machines ou noyaux de processeurs pour effectuer un calcul. Conçue en 1993, cette norme est devenue un standard de communication entre les nœuds d'un système à mémoire distribuée exécutant des programmes parallèles. Grâce à MPI, il serait théoriquement possible de diviser par deux le temps d'exécution d'un programme.

3 - Dans notre projet

3.1 Installation et travail préparatoire

Dans un premier temps, nous avons dû faire en sorte que les dépendances de MPI soient installées. Puisque MPI utilise ssh pour effectuer les communications entre les machines et/ou coeurs du système, il faut que notre installation de ssh soit parfaitement fonctionnelle. Dans un premier temps, nous avons fait en sorte de permettre la connexion sans mots de passe entre les nœuds. Plus précisément, nous avons permis une telle connexion du nœud principal vers les nœuds ouvriers, et de ces nœuds vers le nœud principal. Pour ce faire, nous avons créé des clés avec `ssh-keygen -t rsa`. Cette commande doit être utilisée sur chaque machine devant se connecter aux autres via une clé. Une fois la clé créée, `ssh-copy-id @IP-noeud` permet d'envoyer la clé vers la machine sur laquelle on souhaite se connecter. La même chose doit être faite dans l'autre sens puisque l'on souhaite permettre cette connexion dans les deux sens.

La seconde étape consiste à renommer les nœuds de manière à faciliter la connexion. Pour ce faire, il faut aller dans ~/.ssh/config et ajouter des lignes comme suit :

```
Host nouveauNom
Hostname vraiNomOuIP
User NomdUtilisateurPourLaConnexion
```

Il y a donc quatre blocs de ce type sur le nœud maître, et un seul sur le nœud ouvrier.

Une fois cela fait, le travail sur MPI peut commencer ! En partant du principe qu'un environnement python est déjà installé, il faut installer les packages mpich et python3-mpi4py. En cas d'erreur, il faut être sûr d'avoir effectué la commande `apt-get update.`

```
apt install mpich python3-mpi4py
```

3.2 Programmation avec MPI

Avec cette librairie installée, on peut tester le bon fonctionnement de l'installation en utilisant la commande suivante :

```
mpiexec -n 1 hostname
```

En principe, cela devrait renvoyer l'hostname de la machine locale. On peut donc maintenant tester le cluster entier. Pour ce test, il faut entrer la commande suivante en remplaçant les @ip1 et autres ip par les noms donnés aux nœuds dans la configuration de ssh.

```
mpiexec -n 4 --hosts @ip1,@ip2,@ip3,@ip4 hostname
```

Si ce test fonctionne, les hostname de chaque nœud ouvrier devraient apparaître. On peut donc passer à la dernière étape de notre test sur MPI, qui est de tester le protocole avec un code python. Si on utilise le code donné dans le mail du client, M. Hoguin, le tout devrait fonctionner si les tests précédents sont passés. Voici la commande à utiliser :

```
mpiexec -n1 python3 prime.py 1000
```

Celle-ci devrait calculer les nombres premiers de 1 à 1000 sur 1 nœud (ici 1 cœur) sur la machine locale. Si c'est un succès, on peut alors propager la commande.

```
mpiexec -n4 --host @ip1,@ip2,@ip3,@ip4 python3 prime.py 1000
```

Si cette commande renvoie le même résultat, alors tout fonctionne et MPI a bien répartit la charge de l'application entre les nœuds. Je vous invite à regarder le code de notre module Nombres Premiers, qui est légèrement différent de celui procuré par le client. En

effet, il s’agit d’un prototype codé d’abord pour fonctionner seul, puis modifié pour fonctionner avec MPI de manière plus performante que le code servant d’exemple.