



UNIVERSITE ABDELMALEK ESSAADI  
FACULTE DES SCIENCES ET TECHNIQUES  
DE TANGER



DEPARTEMENT GENIE INFORMATIQUE

**Cycle Master : SIM (Semestre III)**

**Module : Calcul Parallèle et L'applications Distribuées**

## Projet :

L'installation du middleware HTCondor



**Réalisé Par :**

EL MSAOURI Tarik

**Encadré Par :**

Pr. EL AMRANI Chaker

**Année Universitaire 2021/2022**

## Table de la matière

Table de la matière .....	2
Introduction .....	3
Technologies Utilisées .....	3
HTCondor.....	3
VirtualBox .....	3
MPICH .....	3
Installation et configuration HTCondor .....	4
1- Composants du cluster.....	4
2- Configuration des machines .....	4
• Configurer l'adresse IP statique.....	5
3- Installation de HTCONDOR .....	5
4- Installation de MPICH.....	7
• Teste de l'implémentation de MPI .....	7
5- Mettre en place une connexion SSH sans mot de passe .....	8
• Création des certificats et clés DSA .....	8
6- Mettre en place un partage NFS .....	10
7- Configuration HTCONDOR.....	13
• Test du middleware HTCondor sur un code séquentiel et parallèle.....	14
Conclusion.....	17
Références .....	17

## Introduction

Le parallélisme peut être défini par l'ensemble des modèles de calcul, architectures et techniques permettant de résoudre efficacement un problème calculatoire par la mise en œuvre simultanée de plusieurs unités de traitement. Le parallélisme s'oppose aux méthodes dites séquentielles, pour lesquelles la résolution d'un problème est assurée par une séquence d'instructions élémentaires sur un flux de données. Le temps total de résolution est alors la somme des temps d'exécution de chacune de ces instructions. Et parmi les middlewares qui utilisent la technique de parallélisme, il y a HTCondor qui gère les cluster computing.

Le but de ce projet est de pouvoir obtenir un cluster composé d'une machine master et deux machines slaves et l'utilisation de middleware HTCondor pour lancer un job dans le cluster.

## Technologies Utilisées

### HTCondor

HTCondor (High-throughput Computing) est une Framework open-source pour la parallélisation à distribution grossière de tâches de calcul lourdes. Il peut être utilisé pour gérer la charge de travail sur un cluster d'ordinateurs dédié ou pour confier le travail à des ordinateurs de bureau inactifs.

### VirtualBox

VirtualBox est Hyperviseur type2. C'est un outil Open Source de système d'exploitation qui vous permet de faire tourner des machines virtuelles comme Linux sous Windows.

### MPICH

MPICH est une implémentation hautes performances et largement portable de la norme MPI (Message Passing Interface). L'objectif de MPICH est de fournir une implémentation MPI qui prend en charge efficacement différentes plates-formes de calcul et de communication, y compris les clusters de produits (systèmes de bureau, systèmes à mémoire partagée, architectures multicœurs).

# Installation et configuration HTCondor

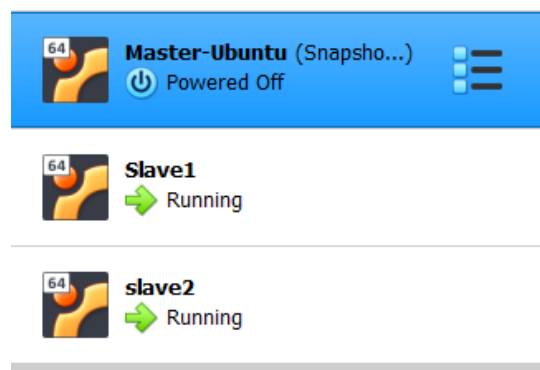
## 1- Composants du cluster

Oracle VM VirtualBox 6.1

Machine Virtuelle **Master** (MSR) : Ubuntu Server 20.04, Adresse IP : **192.168.1.12**

Machine Virtuelle **Slave1** (slave1-tk) : Ubuntu Server 20.04, Adresse IP : **192.168.1.13**

Machine Virtuelle **Slave2** (slave2-tk) : Ubuntu Server 20.04, Adresse IP : **192.168.1.14**



## 2- Configuration des machines

Il faut s'assurer que chacune des machines connaisse les adresses des autres machines donc les machines devraient donc être dans le même réseau. Et ensuite associer un nom de machine à chacune de ces adresses via /etc/hosts dans toutes les machines de cluster.

```
File Machine View Input Devices Help
GNU nano 4.8 /etc/hosts
127.0.0.1 localhost
127.0.1.1 slave2-tk

192.168.1.12 mastertk
192.168.1.13 slave1
192.168.1.14 slave2
```

Teste la connexion entre les machines

```
tarik-server@MSR:~$ ping -c 4 slave1
PING slave1 (192.168.100.51) 56(84) bytes of data.
64 bytes from slave1 (192.168.100.51): icmp_seq=1 ttl=64 time=0.390 ms
64 bytes from slave1 (192.168.100.51): icmp_seq=2 ttl=64 time=0.357 ms
64 bytes from slave1 (192.168.100.51): icmp_seq=3 ttl=64 time=0.371 ms
64 bytes from slave1 (192.168.100.51): icmp_seq=4 ttl=64 time=0.423 ms

--- slave1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3058ms
rtt min/avg/max/mdev = 0.357/0.385/0.423/0.024 ms
tarik-server@MSR:~$
```

- Configurer l'adresse IP statique

```
GNU nano 4.8 /etc/netplan/00-installer-config.yaml
# This is the network config written by 'subiquity'
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s3:
      dhcp4: false
      addresses:
        - 192.168.1.13/24
      gateway4: 192.168.1.1
      nameservers:
        addresses:
          - 8.8.8.8

tarik@slave1-tk:~/partage$ sudo netplan generate
tarik@slave1-tk:~/partage$ sudo netplan apply
tarik@slave1-tk:~/partage$
```

### 3- Installation de HTCONDOR

On utilise la commande : **\$ sudo apt-get install htcondor**

On installe HTCondor dans toutes les machines de cluster. Et on suit les étapes suivantes :

HTCondor configuration

The setup for HTCondor can be handled automatically, asking a few questions to create an initial configuration appropriate for a machine that is either a member of an existing pool or a fully functional "Personal HTCondor installation". This generated initial configuration can be further extended later on.

Otherwise, HTCondor will be installed with a default configuration that needs to be customized manually.

Manage initial HTCondor configuration automatically?

☒ <Yes> ☐ <No>

HTCondor configuration

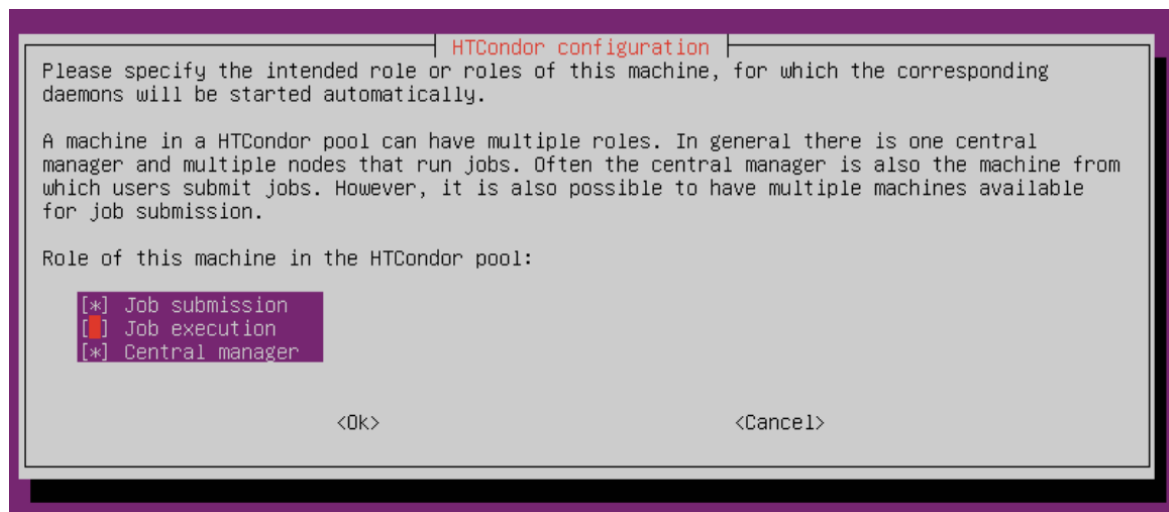
A Personal HTCondor installation is a fully functional HTCondor pool on a single machine. HTCondor will automatically configure and advertise as many slots as it detects CPU cores on this machine. HTCondor daemons will not be available through external network interfaces.

This configuration is not appropriate if this machine is intended to be a member of a pool.

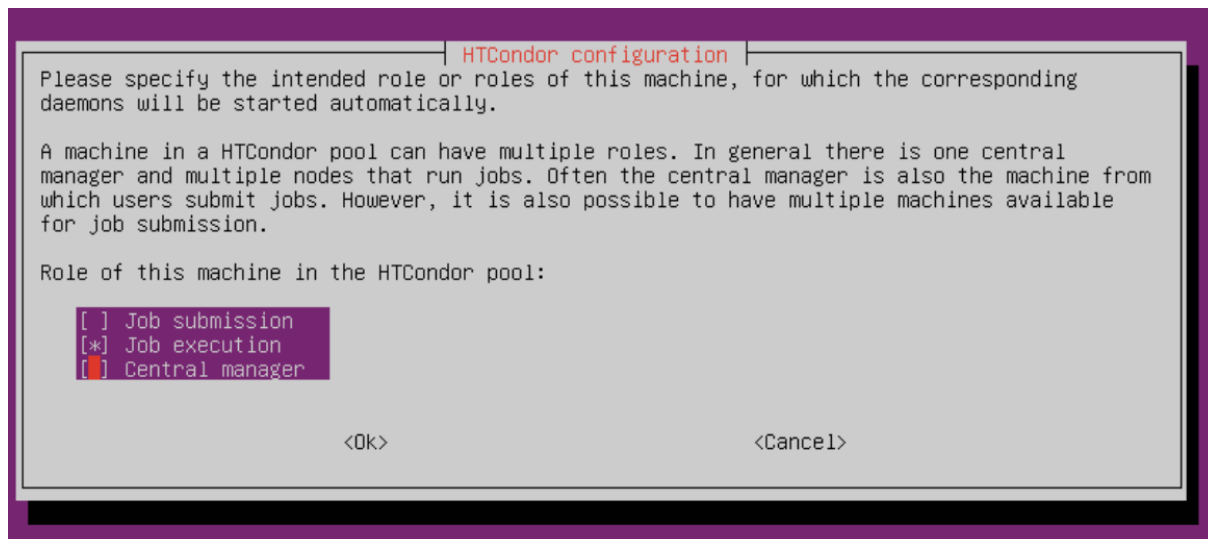
Perform a "Personal HTCondor installation"?

☐ <Yes> ☒ <No>

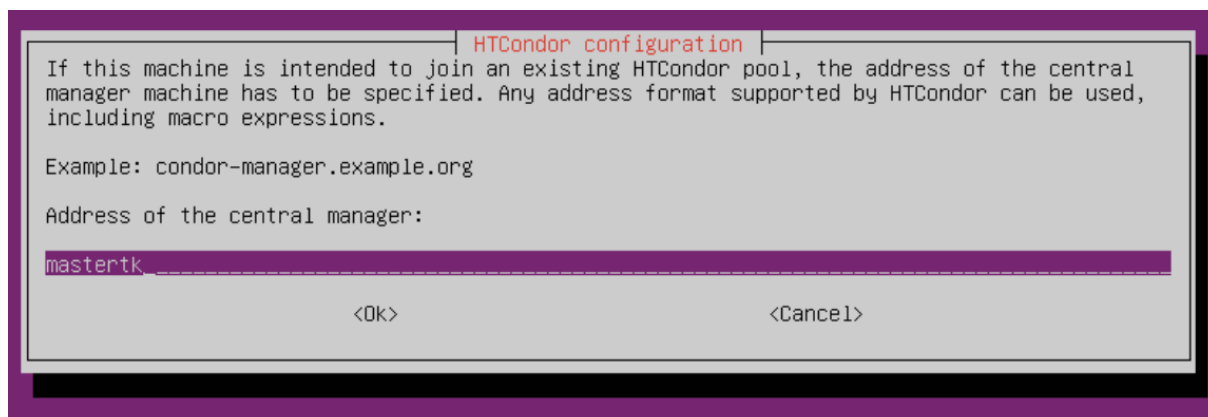
## Pour la Machine master



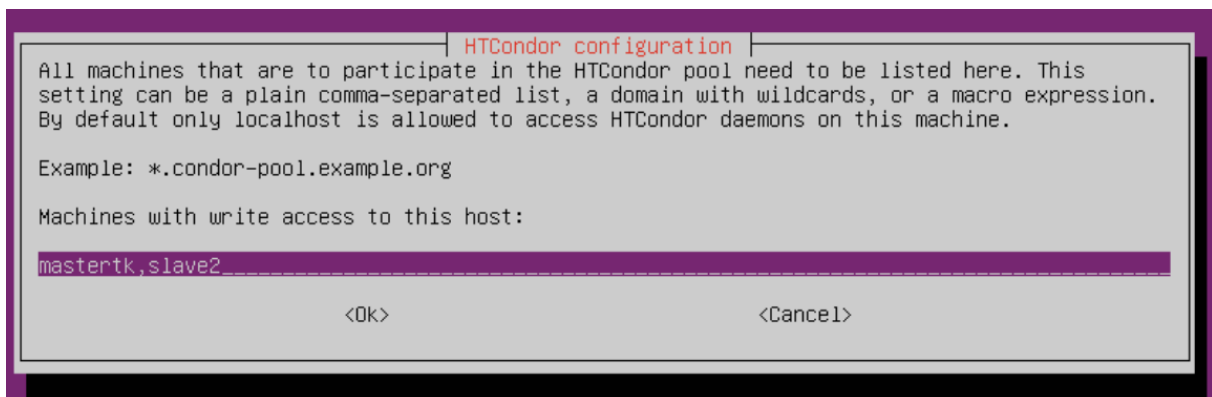
## Pour les deux machines slaves



Et pour les deux types des machines on ajoute l'adresse de la machine centrale **mastertk**.



Aussi les machines qui doivent avoir l'accès au cluster.



#### 4- Installation de MPICH

MPICH, anciennement connu sous le nom de MPICH2, est une implémentation portable librement disponible de MPI, une norme de transmission de messages pour les applications à mémoire distribuée utilisées dans le calcul parallèle.

##### \$ sudo apt-get install mpich

```
tarik@slave1-tk:~$ sudo apt-get install mpich
[sudo] password for tarik:
Sorry, try again.
[sudo] password for tarik:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
```

Cette installation se fait sur toutes les machines de cluster

- Teste de l'implémentation de MPI

On teste MPI avec un programme qui calcule la somme de 1 à 1000.

```
tarik@slave1-tk: ~
GNU nano 4.8 Sum_parallele.cpp Modified
parallel_sum.cpp
#include <mpi.h>
using namespace std;
int main(int argc, char ** argv)
{
    int mynode, totalnodes;
    int sum,startval,endval,accum;
    MPI_Status status;
    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD, &totalnodes);
    MPI_Comm_rank(MPI_COMM_WORLD, &mynode);
    sum = 0;
    startval = 1000*mynode/totalnodes+1;
    endval = 1000*(mynode+1)/totalnodes;
    for(int i=startval;i<=endval;i=i+1) sum = sum + i;
    if(mynode!=0) MPI_Send(&sum,1,MPI_INT,0,1,MPI_COMM_WORLD);
    else
    for(int j=1;j<totalnodes;j=j+1){ MPI_Recv(&accum,1,MPI_INT,j,1,MPI_COMM_WORLD, &status);
    sum = sum + accum; }
    if(mynode == 0){ cout << "The sum from 1 to 1000 is: " << sum;
    }
    MPI_Finalize();
}
```

Pour compiler et exécuter le programme, On lance les commandes suivantes :

```
tarik@slave1-tk: ~  
tarik@slave1-tk:~$ mpiCC -o parallel_Sum Sum_parallelele.cpp  
tarik@slave1-tk:~$  
tarik@slave1-tk:~$ mpirun -np 2 ./parallel_Sum  
The sum from 1 to 1000 is: 500500tarik@slave1-tk:~$
```

## 5- Mettre en place une connexion SSH sans mot de passe

Pour que les programmes puissent être lancés sur les machines slaves, il faut que celles-ci fassent confiance à la machine master

Pour cela, on a besoin de mettre en place une connexion SSH sans mot de passe depuis la machine master.

Mettre en place le serveur OpenSSH sur la machine master :

```
tarik-server@MSR:~$ sudo apt-get install openssh-server  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following additional packages will be installed:  
  openssh-client openssh-sftp-server  
Suggested packages:  
  keychain libpam-ssh monkeysphere ssh-askpass molly-guard  
The following packages will be upgraded:  
  openssh-client openssh-server openssh-sftp-server
```

- **Création des certificats et clés DSA**

Création de la clé publique au niveau du serveur Mastertk

```
tarik-server@MSR:~/.ssh$ ssh-keygen  
Generating public/private rsa key pair.  
Enter file in which to save the key (/home/tarik-server/.ssh/id_rsa):  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /home/tarik-server/.ssh/id_rsa  
Your public key has been saved in /home/tarik-server/.ssh/id_rsa.pub  
The key fingerprint is:  
SHA256:Z7ESrPzuK1yQ+BLLR9hXPArMckVy54ej7Wrh1+pX4lI tarik-server@MSR  
The key's randomart image is:  
+---[RSA 3072]-----+  
|  O.O+.. |  
| . =+ O+. |  
| * 000=.. |  
| +.=.O+ = |  
| . =00S = |  
| + O.O= E . |  
| + O...+ O |  
| O.O.O + |  
| +=+O+ |  
+---[SHA256]-----+  
tarik-server@MSR:~/.ssh$
```



Ensuite, on copie cette clé dans les machines slaves

```
tarik-server@MSR:~/.ssh$ ssh tarik@slave2 "mkdir ~/.ssh;chmod 700 ~/.ssh;touch ~/.ssh/authorized_keys;chmod 600 ~/.ssh/authorized_keys;cat >> ~/.ssh/authorized_keys" < ~/.ssh/id_rsa.pub
The authenticity of host 'slave2 (192.168.100.61)' can't be established.
ECDSA key fingerprint is SHA256:ayvvBuVykVnyfuV6lUfX6688Yg3IAPliDgb0TTCZU34.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'slave2,192.168.100.61' (ECDSA) to the list of known hosts.
tarik@slave2's password:
tarik-server@MSR:~/.ssh$
```

```
tarik-server@MSR:~/.ssh$ ssh tarik@slave1 "mkdir ~/.ssh;chmod 700 ~/.ssh;touch ~/.ssh/authorized_keys;chmod 600 ~/.ssh/authorized_keys;cat >> ~/.ssh/authorized_keys" < ~/.ssh/id_rsa.pub
tarik@slave1's password:
tarik-server@MSR:~/.ssh$
```

```
tarik@slave1-tk:~/.ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGDIr4louvepk3ifbp3+nHVyv6gYs+i9KFG02unxIOegJzNNW7C9yM+GM
PXYW9xxHfLtBXBPjDLitGrVtTwt09K1JH4FDPHlkgWTYpwIwW0n8W0W4nrgVBnA/8TPd/CTVoyXgUPpsmCKyFMtiYJ9oc
A6pCVarmsXS8F0h3Bvr8Rxb+KbHaQUk42B5e0LLQKEwncAYLaf8odvMpnCLVULEZ2S53NNIwr5t9uaQg19H3iOux8yGXm
ya1TRYEPKgE6/fIJ8lh8RV6qiRz7Nza1kvZ3NZLNIwbj1ZLDCuqtoMU0r7bSn5zED52r6hxnBA8V4SVbBe9JY+mfr/hWR
Fqz582H9804IwonSP6WAARBPDM+4GxtRMX5D8Vvh8BIyYndyPjhJpa00TR8H5RQfv4kFYltHF/ywd1DkXBpcxa4hSgB2
Rp/DFS6pVbbWpsQzs1XdAi5xbW7JT8GJ+txq8vFVFTx9yBRCX1FP2De1XnvUHCKfhHctG3kbeB+fYVvZzU6P0= tarik
-server@MSR
tarik@slave1-tk:~/.ssh$
```

La connexion via SSH devient directe et sans mot de passe depuis la machine master vers les autres machines.

```
tarik@slave1-tk: ~
tarik-server@MSR:~$ ssh tarik@slave1
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-91-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Fri 17 Dec 2021 09:12:55 PM UTC

System load:  0.0               Processes:    115
Usage of /:   34.8% of 13.71GB   Users logged in: 1
Memory usage: 15%              IPv4 address for enp0s3: 192.168.100.51
Swap usage:   0%

101 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Last login: Fri Dec 17 20:01:49 2021 from 192.168.100.107
tarik@slave1-tk:~$
```

## 6- Mettre en place un partage NFS

Network File System (ou NFS), c'est un système de fichiers en réseau, est à l'origine un protocole développé par Sun Microsystems en 1984 qui permet à un ordinateur d'accéder via un réseau à des fichiers distants. Il fait partie de la couche application du modèle OSI et utilise le protocole RPC. Ce système de fichiers en réseau permet de partager des données principalement entre systèmes UNIX.

- *Cote machine Master*

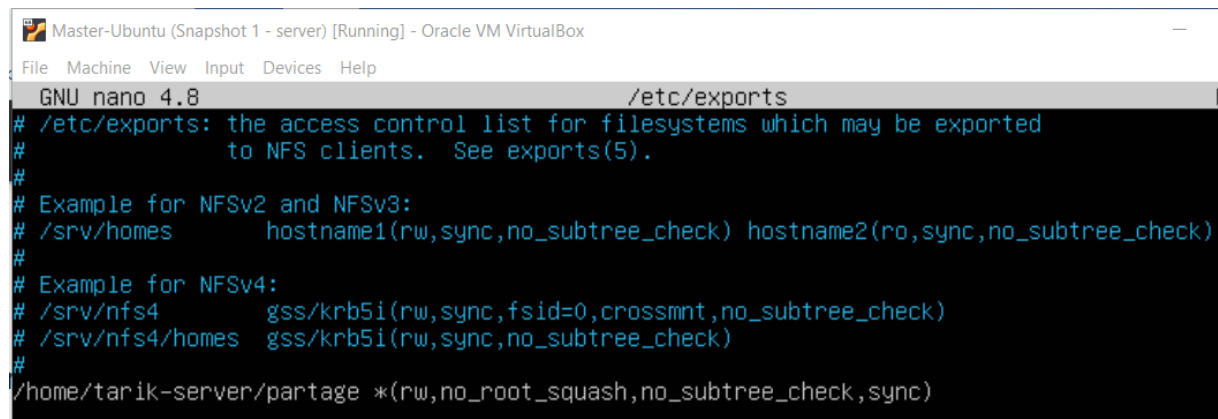
D'abord, on installe le serveur NFS dans la machine centrale Mastertk

```
tarik-server@MSR:~$ sudo apt-get install nfs-kernel-server
[sudo] password for tarik-server:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libnfsidmap2 libtirpc-common libtirpc3 nfs-common rpcbind
Suggested packages:
  watchdog
The following NEW packages will be installed:
  libnfsidmap2 libtirpc-common libtirpc3 nfs-common nfs-kernel-ser
0 upgraded, 6 newly installed, 0 to remove and 135 not upgraded.
Need to get 458 kB of archives.
After this operation, 1,782 kB of additional disk space will be us
Do you want to continue? [Y/n] _
```

Ensuite, On va créer le même dossier dans toutes les machines du cluster qui sera partagé entre eux.

```
tarik@slave1-tk:~$ mkdir partage
```

En plus, le dossier 'partage' doit être exporté. Pour cela, nous devons l'ajouter à /etc/exports.



```
Master-Ubuntu (Snapshot 1 - server) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 4.8 /etc/exports
# /etc/exports: the access control list for filesystems which may be exported
# to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4 gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes gss/krb5i(rw,sync,no_subtree_check)
#
/home/tarik-server/partage *(rw,no_root_squash,no_subtree_check,sync)
```

Enregistrement des modifications en utilisant la commande "exportfs -a".

```
tarik-server@MSR:~$ sudo exportfs -a
```

On redémarre le service.

```
sudo service nfs-kernel-server restart
```

- *Cote machines slaves 1 et 2*

Montage du dossier partage :

```

tarik@slave1-tk: ~
tarik@slave1-tk:~$ sudo apt-get install nfs-common
[sudo] password for tarik:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libnfsidmap2 libtirpc-common libtirpc3 rpcbind
tarik@slave2-tk:~$ sudo apt-get install nfs-common
[sudo] password for tarik:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libnfsidmap2 libtirpc-common libtirpc3 rpcbind
Suggested packages:
  watchdog
The following NEW packages will be installed:
  libnfsidmap2 libtirpc-common libtirpc3 nfs-common rpcbind
0 upgraded, 5 newly installed, 0 to remove and 86 not upgraded.
Need to get 360 kB of archives.
After this operation, 1,362 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
tarik@slave1-tk:~$ sudo mount -t nfs mastertk:/home/tarik-server/partage/ /home/tarik/partage/
tarik@slave1-tk:~$
tarik@slave2-tk:~$ sudo mount -t nfs mastertk:/home/tarik-server/partage /home/tarik/partage/
tarik@slave2-tk:~$

```

Il faut par contre savoir que la commande mount n'est qu'une solution temporaire, car le partage s'arrêtera au redémarrage de la machine dans ce cas. Pour une solution plus permanente, il faut éditer **/etc/fstab** dans les machines slaves.

```

Select tarik@slave2-tk: ~
GNU nano 4.8 /etc/fstab Modified
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options> <dump> <pass>
# / was on /dev/ubuntu-vg/ubuntu-lv during curtin installation
/dev/disk/by-id/dm-uuid-LVM-x3Yt6cMOgnYpuh7wTRELcySoNAGkjEd0lZamaBy5VeFujFyOUSi6k3b62adaoYbc
# /boot was on /dev/sda2 during curtin installation
/dev/disk/by-uuid/6faf0e87-d914-403a-846c-666c0c45786e /boot ext4 defaults 0 0
/swap.img none swap sw 0 0
mastertk:/home/tarik-server/partge /home/tarik/partage

```

- *Test du cluster*

Nous allons le tester en utilisant un petit script en C.

```

Master-Ubuntu (Snapshot 1 - server) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 4.8                                     exemple2.c
#include<stdio.h>
#include<mpi.h>
int main(int argc, char **argv)
{
    int size, node, name;
    //int serial_counter = 0;
    double start, end;
    char nom[MPI_MAX_PROCESSOR_NAME];

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &node);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Barrier(MPI_COMM_WORLD);

    start = MPI_Wtime();
    MPI_Barrier(MPI_COMM_WORLD);

    MPI_Get_processor_name(nom, &name);
    printf("Hello From %. Node = %d. Time = %lf \n", nom, node, (MPI_Wtime() - start));

    MPI_Finalize(); }
  
```

Et nous avons créé le fichier dans la machine master et le fichier a été partagé avec les autres machines

```

tarik@slave2-tk:~/partage$ ll
total 16
drwxrwxr-x 2 tarik tarik 4096 Dec 18 02:29 ./
drwxr-xr-x 5 tarik tarik 4096 Dec 18 01:42 ../
-rwxrwxr-x 1 tarik tarik  516 Dec 18 02:29 exemple2.c*
-rw-rw-r-- 1 tarik tarik 1024 Dec 18 02:29 .exemple2.c.swp
tarik@slave2-tk:~/partage$
  
```

On va créer une hostfile, qui va contenir la liste des machines que on va l'utilisez dans notre cluster.

```

tarik@slave1-tk: ~/partage
GNU nano 4.8                                     hostList
192.168.1.12:1
192.168.1.13:2
192.168.1.14.3
  
```

- Exécution et compilation

```

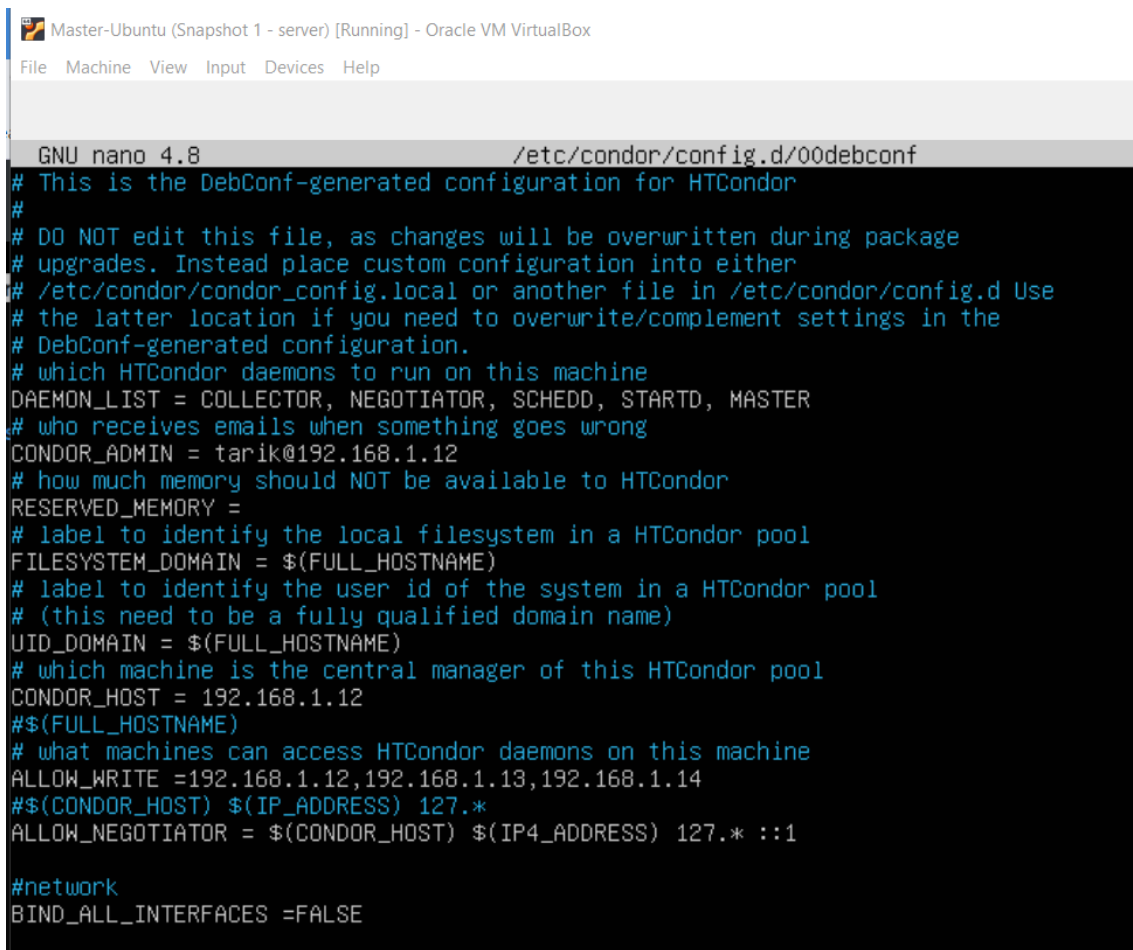
tarik-server@MSR:~/partage$ mpicc -o test2 exemple2.c
tarik@MSR:~/partage$ mpirun ./test2
Hello From MSR. Node = 0. Time = 0.000004
tarik@MSR:~/partage$ _

tarik@MSR:~/partage$ mpirun -np 4 --hostfile hostList ./test2
Hello From MSR. Node = 0. Time = 0.007356
Hello From slave2-tk. Node = 3. Time = 0.007710
Hello From slave1-tk. Node = 1. Time = 0.008265
Hello From slave1-tk. Node = 2. Time = 0.006243
tarik@MSR:~/partage$ _

```

## 7- Configuration HTCONDOR

On change CONDOR\_ADMIN, CONDOR\_HOST, ALLOW\_WRITE



```

GNU nano 4.8 /etc/condor/config.d/00debconf
# This is the DebConf-generated configuration for HTCondor
#
# DO NOT edit this file, as changes will be overwritten during package
# upgrades. Instead place custom configuration into either
# /etc/condor/condor_config.local or another file in /etc/condor/config.d Use
# the latter location if you need to overwrite/complement settings in the
# DebConf-generated configuration.
# which HTCondor daemons to run on this machine
DAEMON_LIST = COLLECTOR, NEGOTIATOR, SCHEDD, STARTD, MASTER
# who receives emails when something goes wrong
CONDOR_ADMIN = tarik@192.168.1.12
# how much memory should NOT be available to HTCondor
RESERVED_MEMORY =
# label to identify the local filesystem in a HTCondor pool
FILESYSTEM_DOMAIN = $(FULL_HOSTNAME)
# label to identify the user id of the system in a HTCondor pool
# (this need to be a fully qualified domain name)
UID_DOMAIN = $(FULL_HOSTNAME)
# which machine is the central manager of this HTCondor pool
CONDOR_HOST = 192.168.1.12
#$(FULL_HOSTNAME)
# what machines can access HTCondor daemons on this machine
ALLOW_WRITE =192.168.1.12,192.168.1.13,192.168.1.14
#$(CONDOR_HOST) $(IP_ADDRESS) 127.*
ALLOW_NEGOTIATOR = $(CONDOR_HOST) $(IP4_ADDRESS) 127.* ::1

#network
BIND_ALL_INTERFACES =FALSE

```

- Test du middleware HTCondor sur un code séquentiel et parallèle

On utilise la commande : **\$ condor\_submit Nom\_fichier\_submit**

Avant de soumettre un travail à Condor, nous avons besoin d'ajouter un fichier de configuration d'un job.

- 1- Soumettre le travail avec universe vanilla

```
GNU nano 4.8 submit.jdl
Universe      = vanilla
Executable    = exemple2
log            = exemple2.log
output        = exemple2.$(process).txt
error         = exemple2.$(process).txt
queue
```

**Universe** : signifie le mode de soumettre un (parallel, vanilla, MPI...).

**Executable** : le nom du programme.

**Log** : C'est le nom d'un fichier où Condor va enregistrer des informations sur l'exécution de notre job.

**Output** : Ou Condor devrait mettre la sortie standard du job.

**Erreur** : Ou Condor devrait mettre les sorties d'erreur de notre job.

- **Exemple 1** : Serial Sommation avec vanilla universe

```
GNU nano 4.8 serial_sum.cpp
#include<iostream>
using namespace std;
int main(int argc, char **argv)
{ int sum; sum = 0;
for(int i=1;i<=1000;i=i+1) sum = sum + i;
cout << "The sum from 1 to 1000 is: " << sum; }
```

```
tarik@MSR: ~/partage/somme
GNU nano 4.8 somme_submit
should_transfer_files = YES
Universe      = vanilla
Executable    = Serial_sum
Log           = Serial_sum.log
Output        = Serial_sum.$(process).out
Error         = Serial_sum.$(process).err
Queue
```

```

tarik@MSR:~/partage/somme$ nano somme_submit
tarik@MSR:~/partage/somme$ condor_submit somme_submit
Submitting job(s).
1 job(s) submitted to cluster 20.
tarik@MSR:~/partage/somme$
tarik@MSR:~/partage/somme$ ls
Serial_sum Serial_sum.0.err Serial_sum.0.out serial_sum.cpp Serial_sum.log somme_submit
tarik@MSR:~/partage/somme$ cat Serial_sum*out
The sum from 1 to 1000 is: 500500tarik@MSR:~/partage/somme$
tarik@MSR:~/partage/somme$
tarik@MSR:~/partage/somme$ cat Serial_sum*log
000 (019.000.000) 12/18 17:26:29 Job submitted from host: <192.168.1.12:9618?addrs=192.168.1.12-9618+[--1]-9618&noUDP&sock=5318_48bd_4>
...
001 (019.000.000) 12/18 17:26:49 Job executing on host: <192.168.1.12:9618?addrs=192.168.1.12-9618+[--1]-9618&noUDP&sock=5318_48bd_5>
...
006 (019.000.000) 12/18 17:26:49 Image size of job updated: 17
    0 - MemoryUsage of job (MB)
    0 - ResidentSetSize of job (KB)
...
005 (019.000.000) 12/18 17:26:50 Job terminated.
    (1) Normal termination (return value 0)
        Usr 0 00:00:00, Sys 0 00:00:00 - Run Remote Usage
        Usr 0 00:00:00, Sys 0 00:00:00 - Run Local Usage
        Usr 0 00:00:00, Sys 0 00:00:00 - Total Remote Usage
        Usr 0 00:00:00, Sys 0 00:00:00 - Total Local Usage
    33 - Run Bytes Sent By Job
    17224 - Run Bytes Received By Job
    33 - Total Bytes Sent By Job
    17224 - Total Bytes Received By Job
Partitionable Resources :   Usage   Request Allocated
    Cpus                :           1           1
    Disk (KB)           :          26          17       7697600
    Memory (MB)         :           0           1       1963

```

- **Exemple 2** : Sommation Parallèle avec Parallel universe

Programme de teste qui calcule la somme en parallèle de 1 à 1000 :

```

GNU nano 4.8                                     parallel_sum.cpp
#include <iostream>
#include <mpi.h>
using namespace std;
int main(int argc, char ** argv)
{
    int mynode, totalnodes;
    int sum, startval, endval, accum;
    MPI_Status status;
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &totalnodes);
    MPI_Comm_rank(MPI_COMM_WORLD, &mynode);
    sum = 0;
    startval = 1000*mynode/totalnodes+1;
    endval = 1000*(mynode+1)/totalnodes;
    for(int i=startval; i<=endval; i=i+1) sum = sum + i;
    if(mynode!=0) MPI_Send(&sum, 1, MPI_INT, 0, 1, MPI_COMM_WORLD);
    else
    for(int j=1; j<totalnodes; j=j+1){ MPI_Recv(&accum, 1, MPI_INT, j, 1, MPI_COMM_WORLD, &status);
    sum = sum + accum; }
    if(mynode == 0){ cout << "The sum from 1 to 1000 is: " << sum;
    }
    MPI_Finalize();
}

```



Fichier de configuration de job

```
GNU nano 4.8                               sommep_submit

Universe   = parallel
Executable = parallel_sum
Log         = parallel_sum.log
Output     = parallel_sum.%(process).out
Error      = parallel_sum.%(process).err
machine_count = 1
requirements = (machine == "tarik@mastertk")
queue

machine_count = 2
requirements = (machine != "tarik@mastertk")
queue

should_transfer_files = YES
arguments= -np 3

when_to_transfer_output = ON_EXIT

Queue
```

Soumission du travail

```
tarik@MSR:~/partage/somme_parallel$ condor_submit sommep_submit
Submitting job(s).
1 job(s) submitted to cluster 14.
tarik@MSR:~/partage/somme_parallel$
tarik@MSR:~/partage/somme_parallel$ ls
hostlist  parallel_sum  parallel_sum.0  parallel_sum.cpp  parallel_sum.log  sommep_submit
tarik@MSR:~/partage/somme_parallel$

tarik@MSR:~/partage/somme_parallel$ cat parallel_sum.log
000 (014.000.000) 12/18 16:59:04 Job submitted from host: <192.168.1.12:9618?addrs=192.168.1.12-9618+[--1]-9618&noUDP&sock=5318_48bd_4>
...
14.000 (014.000.000) 12/18 16:59:04 Job submitted from host: <192.168.1.12:9618?addrs=192.168.1.12-9618+[--1]-9618&noUDP&sock=5318_48bd_4>
```



## Conclusion

A travers ce projet, nous avons pu mettre en pratique nos connaissances en termes de la programmation parallèle et l'optimisation du temps d'exécution. Nous avons aussi acquis une compréhension générale du fonctionnement du middleware HTCondor et la programmation parallèle en utilisant l'utile MPI. De plus, ce projet nous a permis de développer une bonne maîtrise des outils d'administration (SSH, DNS, Terminal server...). Ainsi notre niveau de programmation en C, Cpp. Nous avons pu nous mettre d'accord sur la répartition des tâches et nous organiser pour une gestion optimale du temps.

## Références

### Installation et configuration HTCondor

[https://indico.cern.ch/event/936993/contributions/4022094/attachments/2106388/3542498/Installing\\_HTCondor.pdf](https://indico.cern.ch/event/936993/contributions/4022094/attachments/2106388/3542498/Installing_HTCondor.pdf)

<https://medium.com/mpi-cluster-setup/mpi-clusters-within-a-lan-77168e0191b1>

### Soumission des jobs à HTCondor

[https://indico.cern.ch/event/533066/contributions/2210976/attachments/1293983/1928948/HTCondor\\_Part\\_1.pdf](https://indico.cern.ch/event/533066/contributions/2210976/attachments/1293983/1928948/HTCondor_Part_1.pdf)

[https://research.cs.wisc.edu/htcondor/tutorials/intl-grid-school-3/submit\\_first.html](https://research.cs.wisc.edu/htcondor/tutorials/intl-grid-school-3/submit_first.html)

[https://research.cs.wisc.edu/htcondor/manual/v8.5/2\\_9Parallel\\_Applications.html](https://research.cs.wisc.edu/htcondor/manual/v8.5/2_9Parallel_Applications.html)

### Résoudre le problème de trouver l'adresse du schedd local

<https://stackoverflow.com/questions/30722791/condor-cant-find-address-of-local-schedd>

### Parallélisme et aux architectures parallèles

<https://www.techniques-ingenieur.fr/base-documentaire/technologies-de-l-information-th9/architectures-materielles-42308210/introduction-au-parallelisme-et-aux-architectures-paralleles-h1088/>

### Générer des certificats ssh et des clés publiques et privées

<https://www.slothparadise.com/how-to-create-ssh-keys-for-secure-ssh-access/>