



UNIVERSITÀ DI PERUGIA
Dipartimento di Matematica e Informatica



LAUREA MAGISTRALE IN INFORMATICA
CORSO DI HIGH PERFORMANCE COMPUTING

Esercitazione n.2

CLUSTER AD ALTE PRESTAZIONI

Professore

Prof. Ovaldo Gervasi
Dott. Damiano Perri

Studente

Cristian Cosci
(Matricola: 350863)

Anno Accademico 2021-2022

Indice

1	Introduzione al problema	3
2	Descrizione specifiche Hardware e Software	4
2.1	Specifiche Software	4
2.2	Specifiche Hardware	6
3	Preparazione macchine virtuali	7
3.1	Installazione Sistema Operativo	7
3.2	Installazione Software Necessari	9
3.3	Clonazione Macchina Virtuale	11
3.4	Configurazione IP	13
4	Configurazione Sistema	18
4.1	HTCondor	18
4.1.1	master1	18
4.1.2	slave1 e slave2	19
4.2	Avvio di Condor	22
4.3	Creazione Job di prova	23
5	Conclusioni	27
5.1	Test funzionamento del cluster	27

Capitolo 1

Introduzione al problema

Il lavoro svolto ha l'obiettivo di realizzare un **cluster ad alte prestazioni**, ovvero un insieme di nodi interconnessi tra loro capaci di distribuirsi il carico di lavoro in maniera autonoma, in ambienti orientati all'esecuzione di applicazioni particolarmente onerose dal punto di vista dell'utilizzo del processore.

Un **cluster** di computer è costituito da un insieme di computer (nodi) interconnessi tra loro, capaci di lavorare in modo da essere visti come un unico sistema. Il carico di lavoro è quindi distribuito.

In numerosi casi è necessario effettuare delle simulazioni e dei calcoli particolarmente onerosi per essere sostenuti in una singola macchina, di conseguenza, l'idea alla base del progetto prevedere di suddividere il lavoro tra più computer. A scadenze di tempo regolari, dette checkpoint, i calcoli svolti della cpu vengono salvati su disco e successivamente integrati insieme per ottenere il risultato finale desiderato. Un calcolo che poteva richiedere anni, può essere così essere portato a compimento in mesi o addirittura settimane. In questa esercitazione è richiesto di implementare almeno 2 nodi e installarvi e configurarvi Condor (su tutte le macchine del cluster). Io ho deciso di implementare 2 nodi **slave** (worker) e un nodo **master**. Per testare il sistema sarà necessario creare dei lavori (jobs) da sottomettere a Condor e seguirne il processamento.

Capitolo 2

Descrizione specifiche Hardware e Software

A differenza del progetto precedente in cui erano richiesti ulteriori software come Corosync, HeartBeat ecc, in questo caso per la configurazione dei tre nodi del cluster è sufficiente installare **Condor**. Il cluster verrà realizzato tramite un ambiente virtualizzato (in questo caso Virtualbox è stato scelto come software di virtualizzazione). Il sistema operativo consigliato è Ubuntu Server (versione 20.04.3).

Per completare al meglio la configurazione del cluster e per il successivo test, ho ritenuto necessario installare anche i seguenti pacchetti software (non necessari al corretto funzionamento, ma utilizzati solo per effettuare verifiche e test migliori):

- **stress-ng**;
- **compilatore linguaggio c (gcc)**;
- **net-tools**.

I software sono descritti nella sezione successiva.

2.1 Specifiche Software

CONDOR

È un software che consente di svolgere molte attività di calcolo per un lungo periodo di tempo. Si occupa principalmente del numero di risorse di elaborazione disponibili

per le persone che desiderano utilizzare il sistema. Risulta essere molto utile per ricercatori e altri utenti che sono più interessati al numero di calcoli che possono fare su lunghi lassi di tempo, piuttosto che a calcoli brevi. Offre la possibilità di gestire sia CPU dedicate (cluster) che risorse non dedicate (desktop). Nessun file system condiviso è richiesto. È supportato per molti tipi di lavoro: seriale, parallelo, ecc. Può sopravvivere a arresti anomali, interruzioni di rete e qualsiasi singolo punto di errore. Con Condor è inoltre possibile:

- tenere d'occhio i lavori e rimanere aggiornato sui loro progressi;
- implementare una politica sull'ordine di esecuzione dei lavori;
- aggiungere tolleranza agli errori dei jobs.

Nel Capitolo 4 sono descritte le fasi e le procedure necessarie alla corretta configurazione di un cluster ad alte prestazioni con Condor.

STREES-NG

È un tool che permette di caricare e stressare le CPU di un computer. In particolare ho deciso di utilizzare questo programma in quanto, per verificare il corretto funzionamento del cluster, è necessario appurare che i jobs siano spostati dai nodi occupati e già a pieno carico in nodi con delle CPU libere. Ho quindi utilizzato questo tool per caricare di lavoro delle CPU (avendo piena libertà di scegliere quante CPU stressare).

Compilatore linguaggio c

Ho deciso di installare questo pacchetto software semplicemente per creare dei piccoli script in linguaggio c per farli fungere da jobs.

NET-TOOLS

Ho utilizzato questo pacchetto per semplificare il lavoro di configurazione della rete dei nodi, prima di implementare il cluster. In particolare per verificare mediante il comando **ifconfig** gli indirizzi delle varie schede di rete dei nodi.

2.2 Specifiche Hardware

Sono state utilizzate tre macchine virtuali con il sistema operativo Ubuntu Server con la seguente configurazione (la stessa per tutte e tre le macchine):

- **CORE ASSEGNATI:** 2;
- **RAM:** 2 GB;
- **DISCO:** 10 GB;
- **SCHEDA DI RETE:** impostata in modalità Bridge.

Per comodità è stata prima configurata una singola macchina virtuale e successivamente è stata clonata. È ovviamente necessario clonare la macchina virtuale scegliendo di creare nuovi indirizzi MAC per tutte le schede di rete. I processi e le fasi di installazione sono spiegate nel Capitolo 3.

Capitolo 3

Preparazione macchine virtuali

3.1 Installazione Sistema Operativo

Per prima cosa è stato necessario installare il sistema operativo Ubuntu su una macchina virtuale (vedi Figura 3.1). L'installazione è stata eseguita lasciando tutto di default e creando un solo utente "**cristian**".

In Figura 3.2 è possibile vedere la configurazione della *Macchina 1* ovvero il master con Ubuntu.

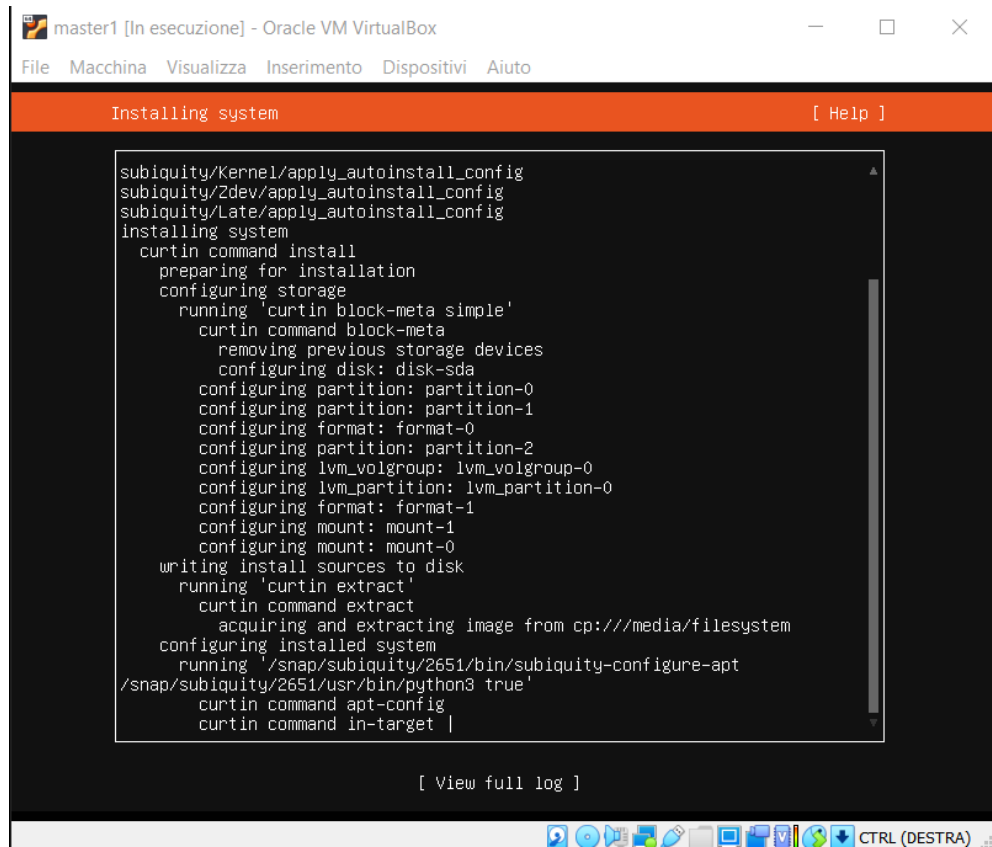


Figura 3.1: Installazione sistema operativo 1.

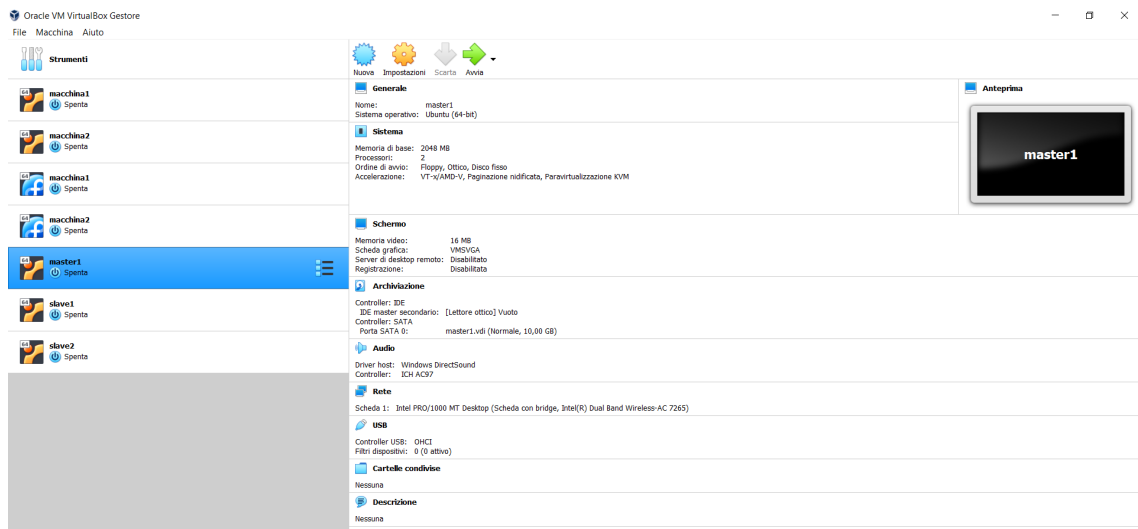
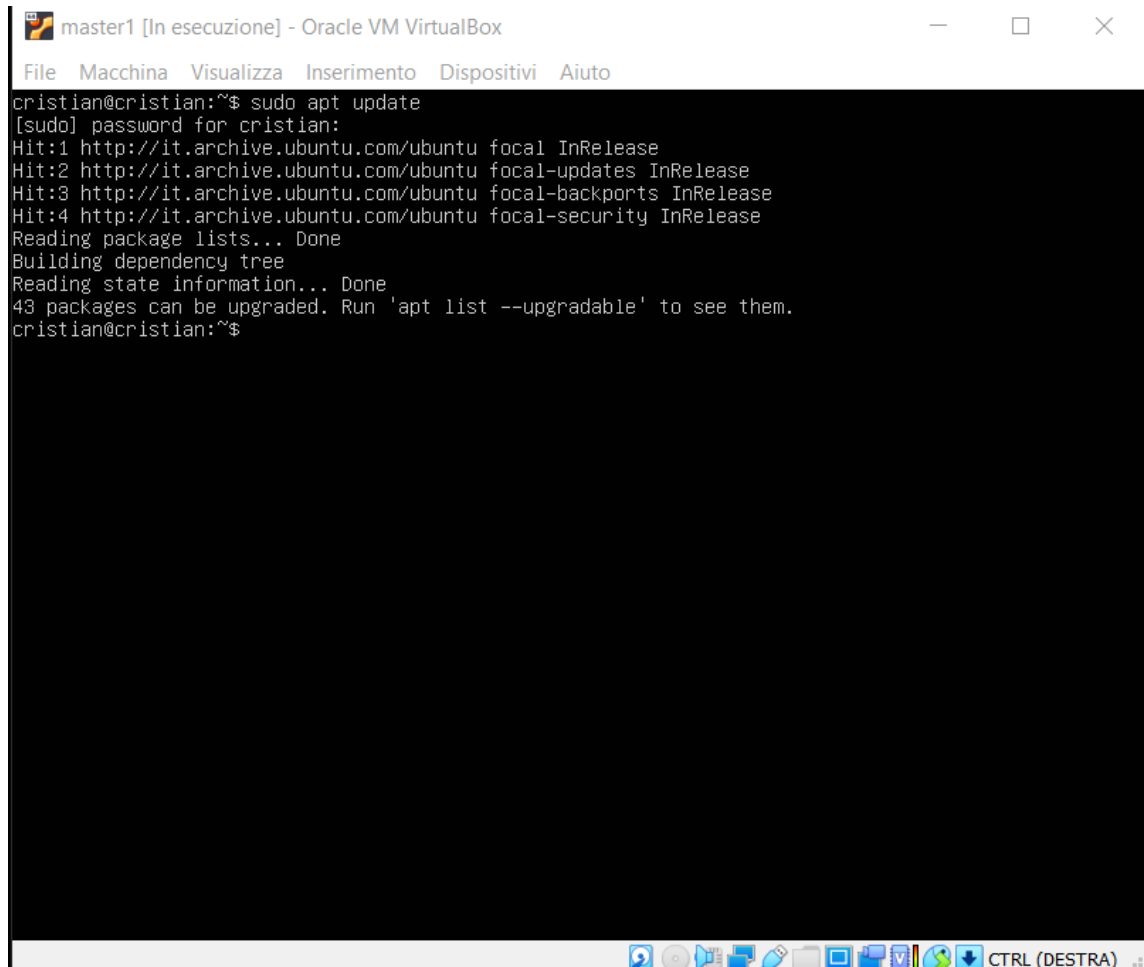


Figura 3.2: Configurazione macchina virtuale con Fedora.

3.2 Installazione Software Necessari

Una volta installato il sistema operativo è stato necessario effettuare un aggiornamento del sistema operativo mediante i comandi **update** e **upgrade**. In questo modo evitiamo tutti i possibili problemi che potrebbero verificarsi per l'incompatibilità di alcuni software.



```
master1 [In esecuzione] - Oracle VM VirtualBox
File  Macchina  Visualizza  Inserimento  Dispositivi  Aiuto
cristian@cristian:~$ sudo apt update
[sudo] password for cristian:
Hit:1 http://it.archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://it.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:3 http://it.archive.ubuntu.com/ubuntu focal-backports InRelease
Hit:4 http://it.archive.ubuntu.com/ubuntu focal-security InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
43 packages can be upgraded. Run 'apt list --upgradable' to see them.
cristian@cristian:~$
```

Figura 3.3: Update sistema operativo.

```
master1 [In esecuzione] - Oracle VM VirtualBox
File  Macchina  Visualizza  Inserimento  Dispositivi  Aiuto
Reading state information... Done
Calculating upgrade... Done
The following packages will be upgraded:
 alsa-ucm-conf cloud-init cloud-initramfs-copymods cloud-initramfs-dyn-netconf libasound2
libasound2-data libdrm-common libdrm2 libnetplan0 libnss-systemd libpam-modules
libpam-modules-bin libpam-runtime libpam-systemd libpam0g libprocps8 libssl1.1 libsystemd0
libudev1 libudisks2-0 linux-base netplan.io open-vm-tools openssl overlayroot procps
python-apt-common python3-apt python3-software-properties python3-update-manager rsync snapd
software-properties-common systemd systemd-sysv systemd-timesyncd ubuntu-advantage-tools udev
udisks2 ufw update-manager-core update-notifier-common wget
43 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 43.0 MB of archives.
After this operation, 1,149 kB of additional disk space will be used.
Do you want to continue? [Y/n]
Get:1 http://it.archive.ubuntu.com/ubuntu focal-updates/main amd64 libpam0g amd64 1.3.1-5ubuntu4.3 [55.4 kB]
Get:2 http://it.archive.ubuntu.com/ubuntu focal-updates/main amd64 libpam-modules-bin amd64 1.3.1-5ubuntu4.3 [41.2 kB]
Get:3 http://it.archive.ubuntu.com/ubuntu focal-updates/main amd64 libpam-modules amd64 1.3.1-5ubuntu4.3 [260 kB]
Get:4 http://it.archive.ubuntu.com/ubuntu focal-updates/main amd64 libnss-systemd amd64 245.4-4ubuntu3.13 [95.8 kB]
Get:5 http://it.archive.ubuntu.com/ubuntu focal-updates/main amd64 udev amd64 245.4-4ubuntu3.13 [1,365 kB]
Get:6 http://it.archive.ubuntu.com/ubuntu focal-updates/main amd64 libudev1 amd64 245.4-4ubuntu3.13 [77.6 kB]
Get:7 http://it.archive.ubuntu.com/ubuntu focal-updates/main amd64 systemd-sysv amd64 245.4-4ubuntu3.13 [10.3 kB]
Get:8 http://it.archive.ubuntu.com/ubuntu focal-updates/main amd64 systemd-timesyncd amd64 245.4-4ubuntu3.13 [28.1 kB]
Get:9 http://it.archive.ubuntu.com/ubuntu focal-updates/main amd64 libpam-runtime all 1.3.1-5ubuntu4.3 [37.3 kB]
Get:10 http://it.archive.ubuntu.com/ubuntu focal-updates/main amd64 libpam-systemd amd64 245.4-4ubuntu3.13 [186 kB]
Get:11 http://it.archive.ubuntu.com/ubuntu focal-updates/main amd64 systemd amd64 245.4-4ubuntu3.13 [3,809 kB]
15% [11 systemd 3,224 kB/3,809 kB 85%] 387 kB/s 1min 37s
```

Figura 3.4: Upgrade sistema operativo.

Successivamente sono stati installati tutti i vari software e pacchetti necessari alla creazione e test di un cluster ad alte prestazioni.

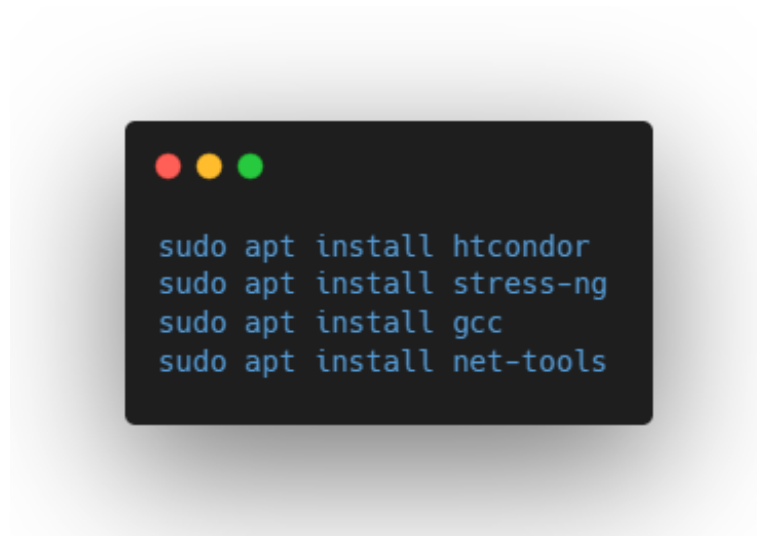


Figura 3.5: Installazione pacchetti fondamentali.

3.3 Clonazione Macchina Virtuale

Successivamente, avendo una macchina con tutti i software necessari, ho proceduto alla clonazione di quest'ultima mediante la procedura fornita direttamente da VirtualBox. Una particolare accortezza sta nello scegliere l'impostazione per generare nuovi indirizzi MAC per tutte le schede di rete (vedi Figura 3.6). A questo punto ho anche impostato le schede di rete delle tre macchine (*master1*, *slave1* e *slave2*) in modalità Bridge (vedi Figura 3.7).

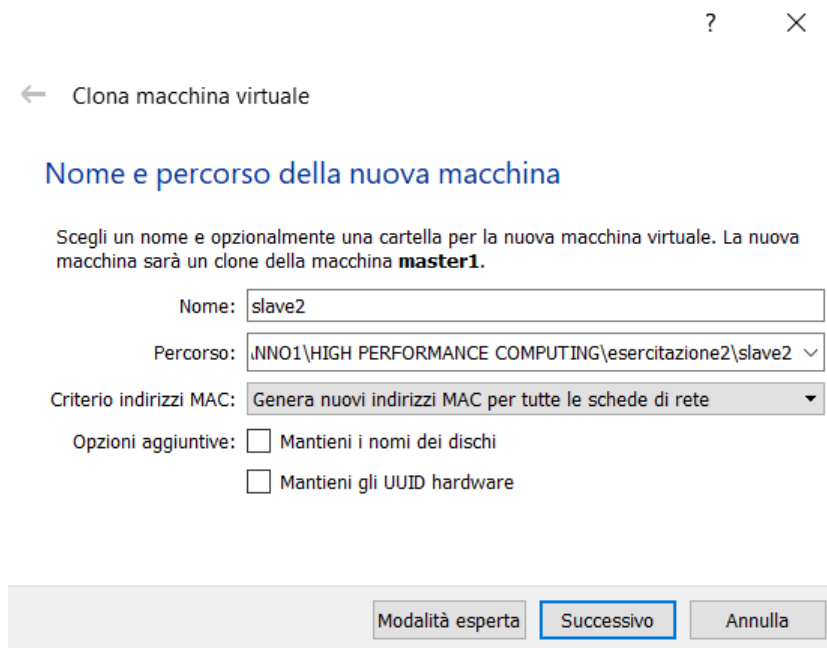


Figura 3.6: Clonazione macchina virtuale.

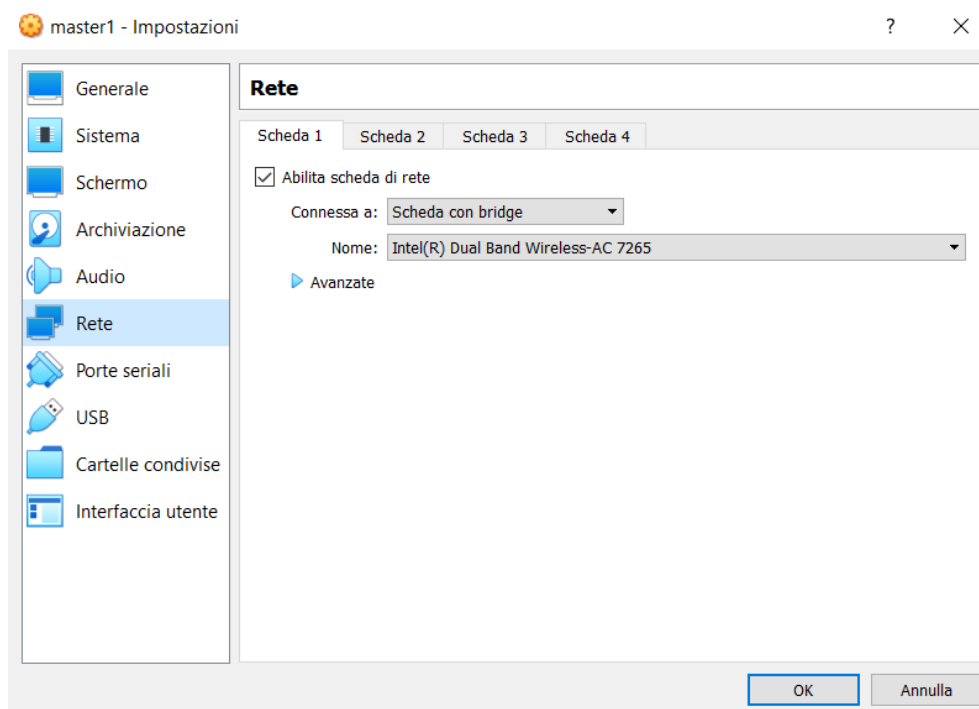


Figura 3.7: Scheda di rete in modalità bridge.

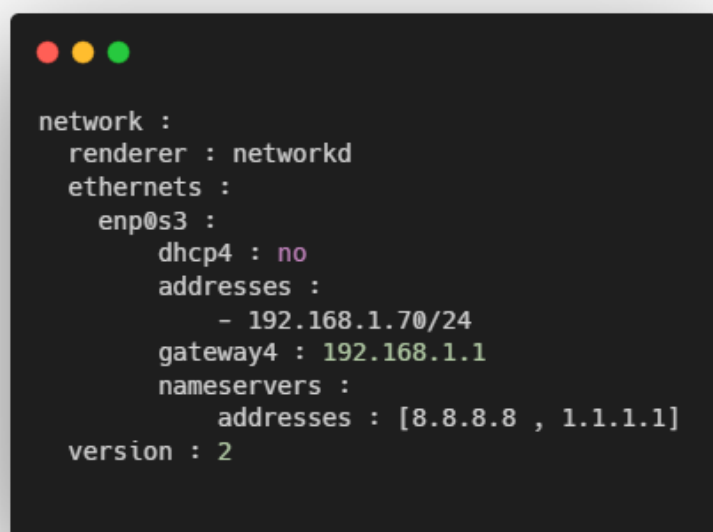
3.4 Configurazione IP

Per evitare problemi legati al DHCP del router di casa (riassegnamento di indirizzi diversi alle macchine virtuali nelle successive accensioni) ho deciso di settare degli indirizzi IP statici alle macchine.

Gli indirizzi IP utilizzati sono:

- **master1**: 192.168.1.70
- **slave1**: 192.168.1.71
- **slave2**: 192.168.1.72

Per assegnare questi indirizzi alle macchine è necessario disabilitare il DHCP e assegnare degli indirizzi IP fissi. La procedura necessaria è la seguente, per tutte e tre le macchine. L'unica accortezza sta nel cambiare l'indirizzo IP nel file `/00-installer-config.yaml`. Per prima cosa ho modificato il file di configurazione `/etc/netplan/00-installer-config.yaml`:



```
network :
  renderer : networkd
  ethernets :
    enp0s3 :
      dhcp4 : no
      addresses :
        - 192.168.1.70/24
      gateway4 : 192.168.1.1
      nameservers :
        addresses : [8.8.8.8 , 1.1.1.1]
  version : 2
```

Figura 3.8: Configurazione IP master1.

In questo caso la scheda di rete da configurare è **enp0s3**. Le istruzioni in successione servono ad effettuare i seguenti passaggi:

- assegno indirizzo IP (con subnet mask) alla macchina;
- assegno il gateway alla macchina;
- assegno DNS;
- l'istruzione **dhcp4: no** permette di evitare incongruenze con i due indirizzi (uno statico e uno assegnato dal DHCP). Con questa istruzione non si ha più l'indirizzo assegnato dal DHCP.

Successivamente ho creato il file **99-disable-network-config.cfg** e vi ho inserito la seguente configurazione:

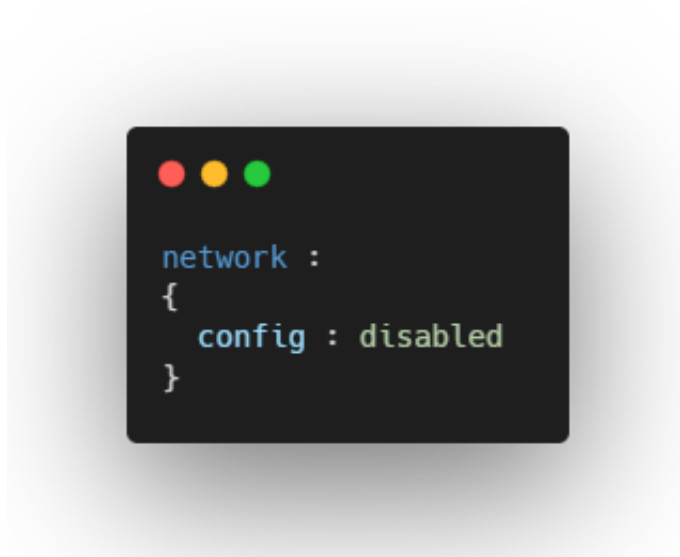
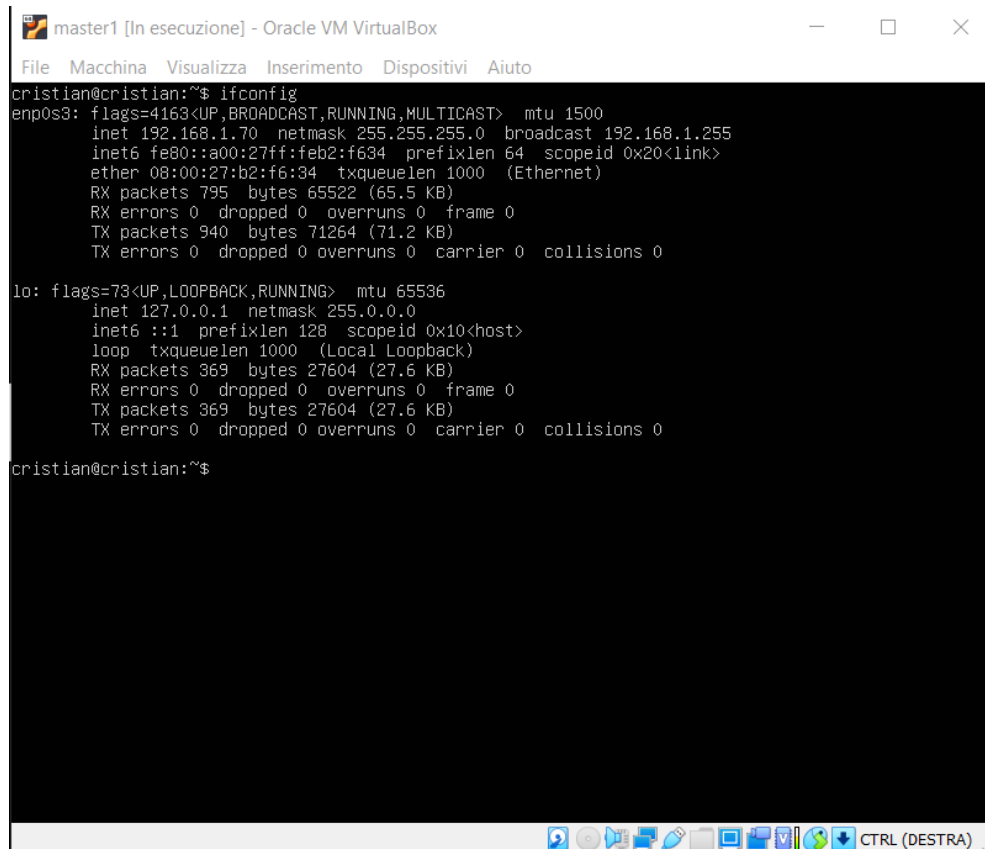


Figura 3.9: Configurazione IP master1.

È possibile vedere (dopo il riavvio del sistema) il corretto funzionamento delle modifiche, leggendo la configurazione delle schede di rete (vedi Figura 3.10).



```
master1 [In esecuzione] - Oracle VM VirtualBox
File  Macchina  Visualizza  Inserimento  Dispositivi  Aiuto
cristian@cristian:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.1.70  netmask 255.255.255.0  broadcast 192.168.1.255
    inet6 fe80::a00:27ff:feb2:f634  prefixlen 64  scopeid 0x20<link>
    ether 08:00:27:b2:f6:34  txqueuelen 1000  (Ethernet)
    RX packets 795  bytes 65522 (65.5 KB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 940  bytes 71264 (71.2 KB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1  netmask 255.0.0.0
    inet6 ::1  prefixlen 128  scopeid 0x10<host>
    loop txqueuelen 1000  (Local Loopback)
    RX packets 369  bytes 27604 (27.6 KB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 369  bytes 27604 (27.6 KB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

cristian@cristian:~$
```

Figura 3.10: Comando ifconfig.

Ho successivamente assegnato un nome ad ogni macchina nel relativo file **/etc/hosts**. Per fare ciò ho semplicemente aggiunto le seguenti righe ai rispettivi file delle macchine.

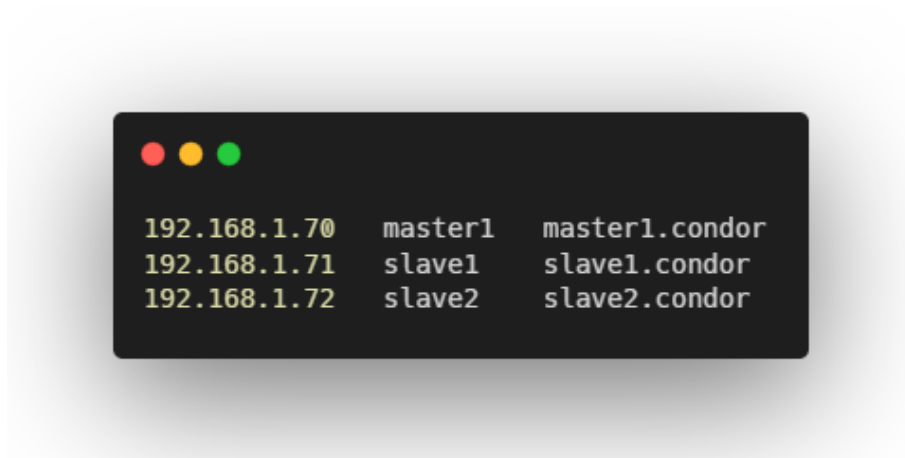


Figura 3.11: Configurazione file hosts.

A questo punto il file `/etc/hosts` compare nel seguente modo.

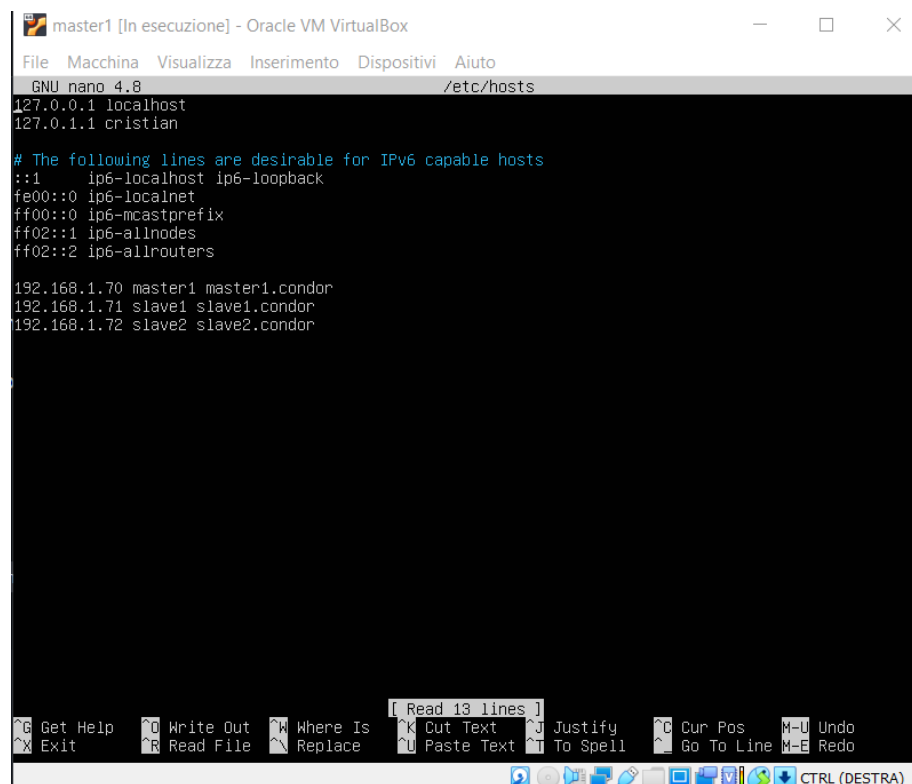


Figura 3.12: File `/etc/hosts` macchina1.

È consigliabile anche modificare in modo appropriato il nome nel file `/etc/hostna-`

me.

Ho deciso anche disabilitare il firewall, in quanto potrebbe limitare delle comunicazioni e causare errori nella realizzazione del cluster se non appositamente configurato. In questo caso si tratta solo di una configurazione di test per verificare l'effettivo funzionamento del cluster, quindi si può procedere semplicemente disabilitandolo con i seguenti comandi (in entrambe le macchine ovviamente):



Figura 3.13: Disabilitato firewall.

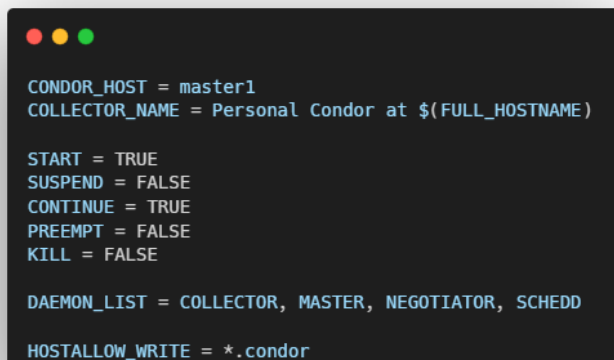
Capitolo 4

Configurazione Sistema

4.1 HTCondor

Per procedere alla configurazione di Condor è necessario modificare il file di configurazione di Condor `/etc/condor/config.d/00personal_condor.config` su ogni nodo, in modo opportuno, distinguendo tra il master e gli slave.

4.1.1 master1

A screenshot of a terminal window with a dark background and light-colored text. The window has three colored window control buttons (red, yellow, green) in the top-left corner. The text inside the terminal is a configuration file for HTCondor, showing settings for a master node named 'master1'.

```
CONDOR_HOST = master1
COLLECTOR_NAME = Personal Condor at $(FULL_HOSTNAME)

START = TRUE
SUSPEND = FALSE
CONTINUE = TRUE
PREEMPT = FALSE
KILL = FALSE

DAEMON_LIST = COLLECTOR, MASTER, NEGOTIATOR, SCHEDD

HOSTALLOW_WRITE = *.condor
```

Figura 4.1: Configurazione condor master1.

- La variabile **CONDOR_HOST** indica qual è il nodo master. Questo sarà uguale anche per gli Slave in quanto il master è solo un nodo.
- Il parametro **COLLECTOR_NAME** viene utilizzato per specificare una breve descrizione del pool. Dovrebbe essere lungo circa 20 caratteri.
- **START** è la policy principale. Un'espressione booleana che, quando True, indica che la macchina è disposta ad avviare l'esecuzione di un lavoro HTCondor. START viene preso in considerazione quando il demone condor_negotiator sta valutando di eliminare il lavoro per sostituirlo con uno che genererà un rango migliore per il demone condor_startd o un utente con una priorità più alta.
- Quando **SUSPEND** diventa true, il lavoro viene sospeso.
- Quando **CONTINUE** diventa true, viene rilasciato un lavoro sospeso.
- **PREEMPT**: Un'espressione booleana che, se True, fa sì che HTCondor interrompa un lavoro attualmente in esecuzione una volta scaduto *MAXJOBRETI-REMENTTIME*. Questa espressione non viene valutata se WANT_SUSPEND è True. Il valore predefinito è False, in modo che la prelazione sia disabilitata.
- **KILL**: Un'espressione booleana che, quando è True, fa sì che HTCondor interrompa immediatamente l'esecuzione di un lavoro vuoto, senza ritardi. Il lavoro è duramente ucciso, quindi qualsiasi tentativo del lavoro di effettuare un checkpoint o di ripulire verrà interrotto. Questa espressione dovrebbe normalmente essere False.
- **HOSTALLOW_WRITE** permette di autorizzare tutti gli host all'interno del dominio condor definito in Sezione 3.4.

4.1.2 slave1 e slave2

La configurazione per i nodi slave è la stessa ed è qui di seguito riportata. Come da richiesta ho deciso di far interrompere l'esecuzione dei job nel momento in cui viene rilevata attività da tastiera. Allo stesso modo i job verranno interrotti anche nel momento in cui il carico della CPU viene utilizzata oltre il 60%.

```

# Macro ( n )
NonCondorLoadAvg = ( LoadAvg - CondorLoadAvg )
HighLoad = 0.6
BgndLoad = 0.3
CPU_Busy = $( NonCondorLoadAvg ) >= $( HighLoad )
CPU_Idle = $( NonCondorLoadAvg ) <= $( BgndLoad )
KeyboardBusy = ( KeyboardIdle < 10 )
MachineBusy = $( CPU_Busy ) || $( KeyboardBusy )
ActivityTimer = ( CurrentTime - EnteredCurrentActivity )

CONDOR_HOST = master1
COLLECTOR_NAME = Personal Condor at $( FULL_HOSTNAME )
WANT_SUSPEND = True
WANT_VACATE = True

START = $( CPU_Idle ) && !$( KeyboardBusy )
SUSPEND = $( MachineBusy )
CONTINUE = $( CPU_Idle ) && KeyboardIdle > 40
PREEMPT = $( ActivityTimer ) > 60 && ( Activity == " Suspended " )
KILL = $( ActivityTimer ) > 180

DAEMON_LIST = MASTER , STARTD
HOSTALLOW_WRITE = *. condor

```

Figura 4.2: Configurazione condor slave.

- **HighLoad**: definisco la soglia di lavoro massima del 60%.
- **CPU_Busy**: nel momento in cui il carico di lavoro esterno a condor supera la soglia è True.
- **CPU_Idle**: è True quando il carico di lavoro della CPU è inferiore al 30%.
- **KeyboardBusy**: utilizzata per monitorare il lavoro da tastiera.
- **WANT_SUSPEND**: in base al valore (True o False) dice al sistema di valutare la variabile **SUSPEND**.

- **WANT_VACATE**: in base al valore (True o False) dice al sistema di valutare la variabile **preempt**.
- **START**: in base al valore assunto (True o False) indica che il nodo è pronto o meno ad accettare ed eseguire il job (è True se non viene rilevata attività da tastiera per 10 secondi e la CPU è sotto la soglia del 30% di utilizzo).
- **SUSPEND**: se True sospende (mette in pausa) l'esecuzione del job in esecuzione.
- **CONTINUE**: in base al valore (True o False) permette di riprendere l'esecuzione del job sospeso (è True se la CPU è sotto la soglia di utilizzo e non sono rilevate attività da tastiera da 40 secondi)
- **PREEMPT**: quando è True crea un checkpoint del lavoro svolto e riporta il job nella coda dei processi, per riassegnarlo ad un altro nodo (è True quando è in stato di sospensione da più di un minuto)
- **KILL**: indica quando killare immediatamente un job (True dopo 180 secondi).

Per creare una buona configurazione è consigliato seguire e leggere le varie guide fornite dal progetto HTCondor stesso, qui di seguito alcuni link utili che sono stati seguiti personalmente per la realizzazione di questo progetto e la relativa risoluzione di errori:

- http://www0.mi.infn.it/condor/manual/3_4Configuring_Condor.html
- <https://htcondor.readthedocs.io/en/latest/admin-manual/introduction-to-configuration.html> (manuale ufficiale)
- <https://htcondor.readthedocs.io/en/latest/admin-manual/configuration-macros.html> (manuale ufficiale)
- https://indico.cern.ch/event/611296/contributions/2604377/attachments/1471361/2276881/TannenbaumT_AdminIntro.pdf

- <https://indico.cern.ch/event/611296/contributions/2604409/attachments/1473514/2281097/EUCW17-Debugging.pdf>
- https://research.cs.wisc.edu/htcondor/tutorials/fermi-2005/submit_first.html

4.2 Avvio di Condor

Risulta ora possibile avviare Condor e verificare che i vari nodi vadano online e siano a disposizione per eseguire i job. Per fare ciò vanno utilizzati i seguenti comandi:

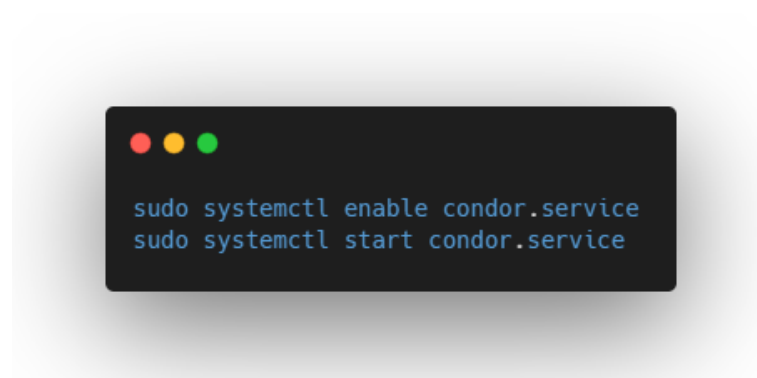


Figura 4.3: Abilito e avvio Condor.

Per verificare lo stato e tutti i parametri è possibile utilizzare il seguente comando:

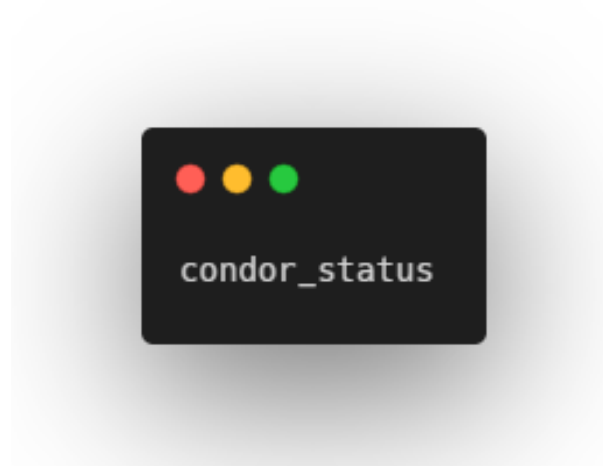


Figura 4.4: Stato Condor.

Si può vedere come tutti i nodi siano online e vi sia uno slot per ogni CPU di quest'ultimo.

```

Wed 05 Jan 2022 02:48:16 PM UTC
Name      OpSys      Arch      State      Activity LoadAv Mem      ActvtyTime
slot1@master1 LINUX      X86_64    Unclaimed Idle       0.020  993  0+00:14:33
slot2@master1 LINUX      X86_64    Unclaimed Idle       0.000  993  0+00:15:03
slot1@slave1  LINUX      X86_64    Unclaimed Idle       0.000  993  0+00:04:34
slot2@slave1  LINUX      X86_64    Unclaimed Idle       0.000  993  0+00:05:03
slot1@slave2  LINUX      X86_64    Unclaimed Idle       0.050  993  0+00:04:33
slot2@slave2  LINUX      X86_64    Unclaimed Idle       0.000  993  0+00:05:03

              Total Owner Claimed Unclaimed Matched Preempting Backfill Drain
              X86_64/LINUX      6      0      0          6          0          0      0      0
              Total      6      0      0          6          0          0      0      0

-- Schedd: master1 : <192.168.1.70:9618?... @ 01/05/22 14:48:16
ID      OWNER      SUBMITTED      RUN_TIME ST PRI SIZE CMD
0 jobs; 0 completed, 0 removed, 0 idle, 0 running, 0 held, 0 suspended

```

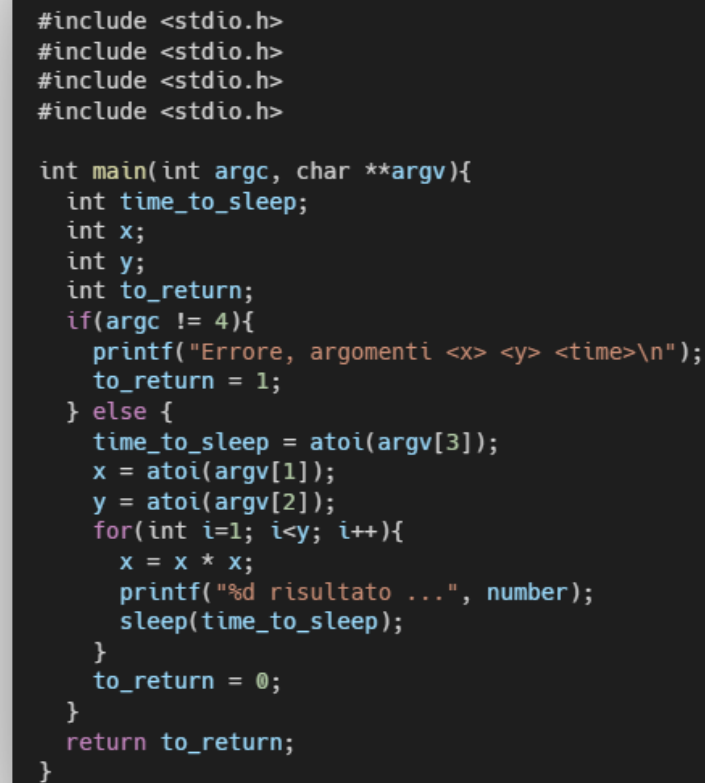
Figura 4.5: Condor Status.

Come si può vedere in Figura 4.5 tutti gli slot sono in stato **Unclaimed/Idle**, la configurazione è quindi corretta ed è pronta ad essere testata.

4.3 Creazione Job di prova

Per verificare l'effettivo funzionamento del cluster ad alte prestazioni è necessario assegnare dei job da eseguire e seguirne il processamento. Ci sono vari modi per fare ciò, personalmente ho deciso di creare un piccolo script nel linguaggio C che va ad

effettuare delle moltiplicazioni in un ciclo for e attende dei secondi ad ogni iterazione. Questo permette di occupare la CPU di un nodo per un tempo necessario a verificarne il processamento. Ho scelto di scrivere uno script in C perchè è piuttosto veloce e semplice anche facendolo direttamente con l'editor di testo da terminale **nano**. Qui di seguito è riportato lo script, il quale è stato successivamente compilato con il comando **gcc -o eseguibile eseguibile.c**. L'eseguibile sarà poi preso dal **submit_file** per testare il cluster.



```
#include <stdio.h>
#include <stdio.h>
#include <stdio.h>
#include <stdio.h>

int main(int argc, char **argv){
    int time_to_sleep;
    int x;
    int y;
    int to_return;
    if(argc != 4){
        printf("Errore, argomenti <x> <y> <time>\n");
        to_return = 1;
    } else {
        time_to_sleep = atoi(argv[3]);
        x = atoi(argv[1]);
        y = atoi(argv[2]);
        for(int i=1; i<y; i++){
            x = x * x;
            printf("%d risultato ...", x);
            sleep(time_to_sleep);
        }
        to_return = 0;
    }
    return to_return;
}
```

Figura 4.6: Script in C che fungerà da eseguibile.

Successivamente è stato necessario scrivere un **submit_file** per sottomettere il job

al master. Il file nel mio caso chiamato **clustertester** è qui di seguito riportato:

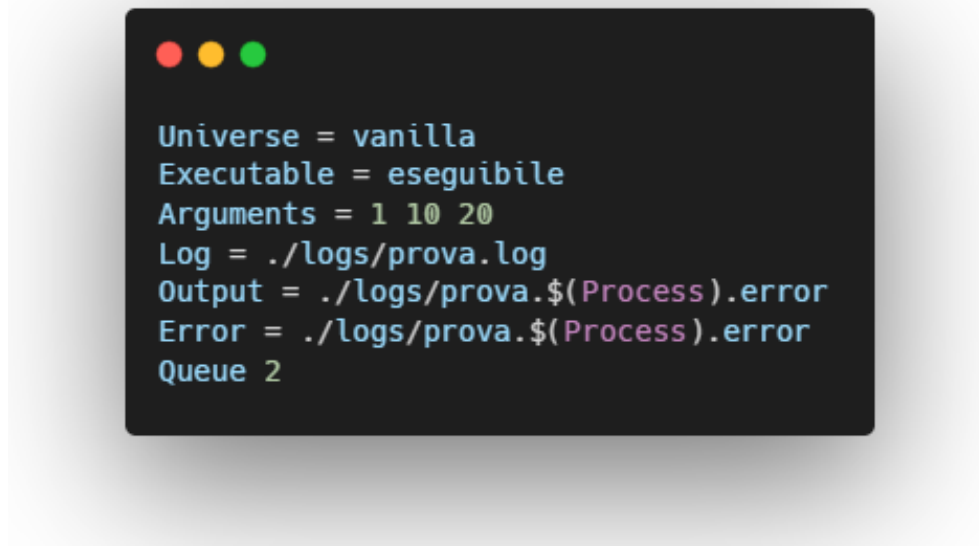


Figura 4.7: **clustertester** in C che fungerà da eseguibile.

- **Universe:** L'universo **vanilla** è una buona impostazione predefinita, poiché ha il minor numero di restrizioni sui job.
- **Executable:** indica il file eseguibile che verrà avviato.
- **Arguments:** è la lista degli argomenti da passare allo script C.
- **Log, Output, Error:** indicano dove verranno salvati i file di log, output del programma e i possibili errori.
- **Queue:** il valore assegnato indica quanti job dello stesso eseguibile verranno sottomessi al cluster.

È necessario anche creare la cartella **logs** per tenere il tutto più ordinato possibile. Risulta ora possibile eseguire il submit per il test e controllare e verificare la corretta configurazione del cluster.

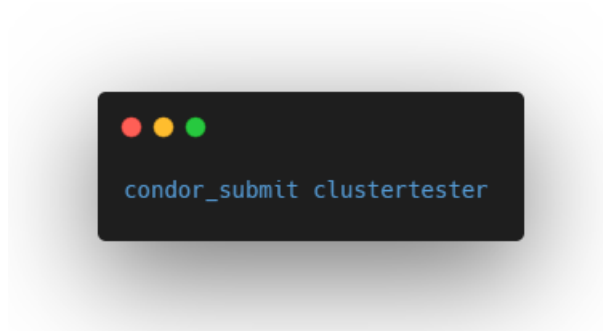


Figura 4.8: Comando per il submit del job.

Si consiglia inoltre di assegnare tutti i permessi alla cartella **/logs**, all'eseguibile e al tester da submittare. Nel mio caso, non garantendo i permessi a questi file e cartelle ha causato degli errori durante il processamento dei jobs, non permettendo la corretta esecuzione. In particolare risultava un errore nella scrittura dei log di Output dell'eseguibile nella cartella opportuna. Tracciando e analizzando i job non eseguiti l'errore incontrato è **errno13** dovuto appunto ai permessi mancanti. Per ovviare al problema si consiglia il seguente comando:

sudo chmod -R 777 .

(se ci si trova nella directory con tutti i file di lavoro).

Nel Capitolo 5 è descritto l'effettivo test da svolgere per verificare se la configurazione è corretta.

Capitolo 5

Conclusioni

5.1 Test funzionamento del cluster

Per verificare che il cluster sia correttamente configurato e funzioni a dovere è necessario che sia in grado di distribuire i job alle CPU nella maniera più efficiente possibile per il loro processamento. In questa particolare configurazione il cluster deve essere anche in grado di controllare l'attività dell'utente e, nel caso in cui sia registrata attività da tastiera o la CPU sia carica oltre la soglia del 60% il job deve essere interrotto. Dove possibile, il job interrotto deve essere spostato in un'altra CPU libera o messo in coda per essere eseguito. Per verificare il tutto sono state svolte le seguenti fasi:

- assegno dei job al cluster e verifico che siano messi in esecuzione correttamente (ho deciso di processare un solo job per comodità nel seguirne il processing, è comunque possibile eseguire il submit di più job cambiando il parametro **Queue**);
- riassegno dei job osservando a quale macchina vengono assegnati;
- mediante il tool **stress-ng** carico di lavoro le CPU su cui è svolto il job (causando il superamento della soglia del 60% e anche attività da tastiera);
- verifico che i job siano sospesi e spostati su altre CPU libere (e completati) o rimessi in coda.

Qui di seguito sono riportati alcuni comandi che ho utilizzato in questo test:



Figura 5.1: Comandi utili..

- il primo comando serve per assegnare i job al cluster (il test è effettuato assegnando un job);
- **condor_status** è utile per osservare lo stato di ogni CPU del cluster;
- **condor_q --nobatch** è utile per osservare la coda dei job e il loro stato;
- il quarto comando serve a caricare di lavoro le due CPU sulla macchina di riferimento (per stoppare il lavoro si utilizzando i comandi **fg** e **CONTROL C**);
- **htop** serve a mostrare il carico delle CPU della macchina.

Per prima cosa vado quindi ad eseguire il submit del job (è consigliabile dei valori per lo sleep molto alti, in modo che il sistema abbia tempo di verificare i cambiamenti ed agire di conseguenza) con il primo comando in Figura 5.1. Nel mio caso il job viene assegnato a **slave2**. A questo punto vado ad avviare anche il tool per caricare le CPU del nodo (vado a caricare entrambe le CPU anche se utilizzo un solo job in quanto voglio verificare che il job possa essere preso in carico da un altro nodo). In Figura 5.2 e in Figura 5.3 si mostra lo stato del cluster e dello **slave2**. Si può vedere come lo **slot1** dello **slave2** sia in stato di **Claimed** e il job è stato sospeso.

```

Thu 06 Jan 2022 03:22:57 PM UTC
Name      OpSys    Arch    State    Activity    LoadAv    Mem    ActvtyTime
slot1@master1  LINUX    X86_64  Unclaimed Idle        0.040    993    0+00:39:36
slot2@master1  LINUX    X86_64  Unclaimed Idle        0.000    993    0+00:40:05
slot1@slave1   LINUX    X86_64  Unclaimed Idle        0.010    993    0+00:34:33
slot2@slave1   LINUX    X86_64  Unclaimed Idle        0.000    993    0+00:35:03
slot1@slave2   LINUX    X86_64  Claimed   Suspended   0.000    993    0+00:00:03
slot2@slave2   LINUX    X86_64  Owner     Idle        0.160    993    0+00:00:03

Total Owner Claimed Unclaimed Matched Preempting Backfill Drain
X86_64/LINUX    6      1      1      4      0      0      0      0
Total          6      1      1      4      0      0      0      0

-- Schedd: master1 : <192.168.1.70:9618?... @ 01/06/22 15:22:57
ID    OWNER    SUBMITTED    RUN_TIME ST PRI SIZE CMD
35.0  cristian    1/6  15:21    0+00:01:02 S  0   0.0 eseguibile 1 20 40

1 jobs; 0 completed, 0 removed, 0 idle, 0 running, 0 held, 1 suspended

```

Figura 5.2: Controllo lo stato del cluster dopo aver caricato la CPU dello slave2 oltre la soglia del 60%.

```

1  [|||||||||||||||||||||||||||||||||||||100.0%]  Tasks: 39, 36 thr; 2 running
2  [|||||||||||||||||||||||||||||||||||||100.0%]  Load average: 1.64 0.59 0.27
Mem[|||||196M/1.94G]  Uptime: 04:21:22
Swp[0K/1.69G]

  PID USER      PRI  NI  VIRT   RES   SHR  S  CPU% MEM%   TIME+  Command
 2219 cristian   20    0 44056   7124  3628  R  99.7  0.4   1:27.33 stress-ng-cpu
 2220 cristian   20    0 44056   7124  3628  R  99.7  0.4   1:27.10 stress-ng-cpu
 2221 cristian   20    0  8060   4116  3380  R   0.0  0.2   0:00.20 htop

```

Figura 5.3: CPU del cluster con il 100% di lavoro.

Ora è necessario che **slave2** passi in stato **Owner** indicando che non è in grado di eseguire job.

Il job assegnatogli in precedenza viene quindi liberato e passa in **idle**. A questo punto il master sarà in grado di riassegnare il job, al checkpoint di lavoro precedente, ad un nodo libero (nel mio caso **slave1**). In Figura 5.4 viene mostrato lo stato del cluster a questo punto.

```

master1 [In esecuzione] - Oracle VM VirtualBox
File  Macchina  Visualizza  Inserimento  Dispositivi  Aiuto
Thu 06 Jan 2022 03:25:18 PM UTC
Name      OpSys      Arch      State      Activity LoadAv Mem  ActvtyTime
slot1@master1 LINUX      X86_64 Unclaimed Idle      0.040 993 0+00:39:36
slot2@master1 LINUX      X86_64 Unclaimed Idle      0.000 993 0+00:40:05
slot1@slave1  LINUX      X86_64 Unclaimed Idle      0.010 993 0+00:34:33
slot2@slave1  LINUX      X86_64 Unclaimed Idle      0.000 993 0+00:35:03
slot1@slave2  LINUX      X86_64 Owner   Idle      0.860 993 0+00:00:03
slot2@slave2  LINUX      X86_64 Owner   Idle      0.160 993 0+00:00:03

Total Owner Claimed Unclaimed Matched Preempting Backfill Drain
X86_64/LINUX      6      2      0      4      0      0      0      0
Total      6      2      0      4      0      0      0      0

-- Schedd: master1 : <192.168.1.70:9618?... @ 01/06/22 15:25:18
ID      OWNER      SUBMITTED      RUN_TIME ST PRI SIZE CMD
35.0    cristian      1/6 15:21      0+00:03:16 I 0      0.0 eseguibile 1 20 40

1 jobs; 0 completed, 0 removed, 1 idle, 0 running, 0 held, 0 suspended

```

Figura 5.4: Stato del cluster con **slave2** in stato **Owner**.

Dopo un paio di minuti il cluster rieffettuerà un controllo dei nodi liberi e dei lavori da eseguire e assegnerà il job ad un nodo disponibile. In Figura 5.5 si può vedere come stia funzionando tutto come previsto.

```

master1 [In esecuzione] - Oracle VM VirtualBox
File  Macchina  Visualizza  Inserimento  Dispositivi  Aiuto
Thu 06 Jan 2022 03:29:12 PM UTC
Name      OpSys      Arch      State      Activity LoadAv Mem  ActvtyTime
slot1@master1 LINUX      X86_64 Unclaimed Idle      0.010 993 0+00:44:35
slot2@master1 LINUX      X86_64 Unclaimed Idle      0.000 993 0+00:45:04
slot1@slave1  LINUX      X86_64 Claimed  Busy      0.000 993 0+00:00:03
slot2@slave1  LINUX      X86_64 Unclaimed Idle      0.000 993 0+00:40:03
slot1@slave2  LINUX      X86_64 Owner   Idle      0.970 993 0+00:01:38
slot2@slave2  LINUX      X86_64 Owner   Idle      1.000 993 0+00:04:09

Total Owner Claimed Unclaimed Matched Preempting Backfill Drain
X86_64/LINUX      6      2      1      3      0      0      0      0
Total      6      2      1      3      0      0      0      0

-- Schedd: master1 : <192.168.1.70:9618?... @ 01/06/22 15:29:12
ID      OWNER      SUBMITTED      RUN_TIME ST PRI SIZE CMD
35.0    cristian      1/6 15:21      0+00:03:33 R 0      0.0 eseguibile 1 20 40

1 jobs; 0 completed, 0 removed, 0 idle, 1 running, 0 held, 0 suspended

```

Figura 5.5: Stato del cluster con **slave1** che ha preso in carico il job precedente.

Come richiesto sono andato a verificare che anche nel caso in cui sia registrata attività da tastiera dal nodo il job deve essere sospeso e venga liberata la CPU. Sono andato quindi a creare attività da tastiera e ho verificato che il job venga sospeso (vedi Figura 5.6). Dopo 40 secondi dal momento in cui non si registra più attività da tastiera il nodo deve essere in grado di riprendere l'esecuzione del lavoro. Come si vede in Figura 5.7 ciò accade.

```

Thu 06 Jan 2022 03:30:49 PM UTC
Name      OpSys      Arch      State      Activity  LoadAv  Mem  ActvtyTime
slot1@master1  LINUX      X86_64    Unclaimed  Idle      0.010   993  0+00:44:35
slot2@master1  LINUX      X86_64    Unclaimed  Idle      0.000   993  0+00:45:04
slot1@slave1   LINUX      X86_64    Claimed    Suspended 0.000   993  0+00:00:03
slot2@slave1   LINUX      X86_64    Owner      Idle      0.000   993  0+00:00:03
slot1@slave2   LINUX      X86_64    Owner      Idle      0.970   993  0+00:01:38
slot2@slave2   LINUX      X86_64    Owner      Idle      1.000   993  0+00:04:09

      Total Owner Claimed Unclaimed Matched Preempting Backfill Drain
      X86_64/LINUX      6      3      1          2          0          0          0          0
      Total      6      3      1          2          0          0          0          0

-- Schedd: master1 : <192.168.1.70:9618?... @ 01/06/22 15:30:49
ID   OWNER      SUBMITTED  RUN_TIME ST PRI SIZE CMD
35.0 cristian      1/6  15:21   0+00:05:10 S  0    0.0 eseguibile 1 20 40

1 jobs; 0 completed, 0 removed, 0 idle, 0 running, 0 held, 1 suspended

```

Figura 5.6: Stato del cluster con **slave1** che sospende il job nel momento in cui rileva attività da tastiera.

```

master1 [In esecuzione] - Oracle VM VirtualBox
File  Macchina  Visualizza  Inserimento  Dispositivi  Aiuto
Thu 06 Jan 2022 03:32:31 PM UTC
Name      OpSys      Arch      State      Activity LoadAv Mem  ActvtyTime
slot1@master1 LINUX      X86_64    Unclaimed Idle       0.000 993 0+00:49:35
slot2@master1 LINUX      X86_64    Unclaimed Idle       0.000 993 0+00:50:04
slot1@slave1  LINUX      X86_64    Claimed   Busy       0.000 993 0+00:00:03
slot2@slave1  LINUX      X86_64    Unclaimed Idle       0.000 993 0+00:00:38
slot1@slave2  LINUX      X86_64    Owner     Idle       1.000 993 0+00:06:38
slot2@slave2  LINUX      X86_64    Owner     Idle       1.000 993 0+00:09:09

Total Owner Claimed Unclaimed Matched Preempting Backfill Drain
X86_64/LINUX      6      2      1      3      0      0      0      0
Total            6      2      1      3      0      0      0      0

-- Schedd: master1 : <192.168.1.70:9618?... @ 01/06/22 15:32:31
ID      OWNER      SUBMITTED  RUN_TIME ST PRI SIZE CMD
35.0    cristian      1/6 15:21 0+00:06:52 R 0 0.0 eseguibile 1 20 40

1 jobs; 0 completed, 0 removed, 0 idle, 1 running, 0 held, 0 suspended

```

Figura 5.7: Stato del cluster con **slave1** che ha ripreso a processare il job precedente.

Infine possiamo notare come il job sia portato a termine e lo **slave1** torni in stato **Unclaimed** mettendosi a disposizione di nuovi job (vedi Figrua 5.8).

```

master1 [In esecuzione] - Oracle VM VirtualBox
File  Macchina  Visualizza  Inserimento  Dispositivi  Aiuto
Thu 06 Jan 2022 03:46:17 PM UTC
Name      OpSys      Arch      State      Activity LoadAv Mem  ActvtyTime
slot1@master1 LINUX      X86_64    Unclaimed Idle       0.030 993 0+00:59:35
slot2@master1 LINUX      X86_64    Unclaimed Idle       0.000 993 0+01:00:04
slot1@slave1  LINUX      X86_64    Unclaimed Idle       0.000 993 0+00:00:03
slot2@slave1  LINUX      X86_64    Unclaimed Idle       0.000 993 0+00:10:38
slot1@slave2  LINUX      X86_64    Owner     Idle       1.000 993 0+00:16:38
slot2@slave2  LINUX      X86_64    Owner     Idle       1.000 993 0+00:19:09

Total Owner Claimed Unclaimed Matched Preempting Backfill Drain
X86_64/LINUX      6      2      0      4      0      0      0      0
Total            6      2      0      4      0      0      0      0

-- Schedd: master1 : <192.168.1.70:9618?... @ 01/06/22 15:46:17
ID      OWNER      SUBMITTED  RUN_TIME ST PRI SIZE CMD
0 jobs; 0 completed, 0 removed, 0 idle, 0 running, 0 held, 0 suspended

```

Figura 5.8: Stato del cluster con **slave1** che ha terminato il job .

La configurazione è quindi corretta e operativa.

Il vantaggio che si ottiene realizzando un'architettura di questo tipo è sia economico che prestazionale: invece che avere un'unica macchina ad altissime prestazioni e dal costo elevato, si distribuiscono i calcoli su più unità detti nodi (anche con caratteristiche hardware eterogenee tra loro) e con un costo di messa in opera contenuto. I problemi di configurazione possono essere difficili da gestire: i job non partono, partono ma non vengono gestiti dalla cpu, partono ma rimangono idle, ecc. Personalmente, nel caso in cui ci siano dei problemi, consiglio di controllare lo stato dei vari job utilizzando il seguente comando:

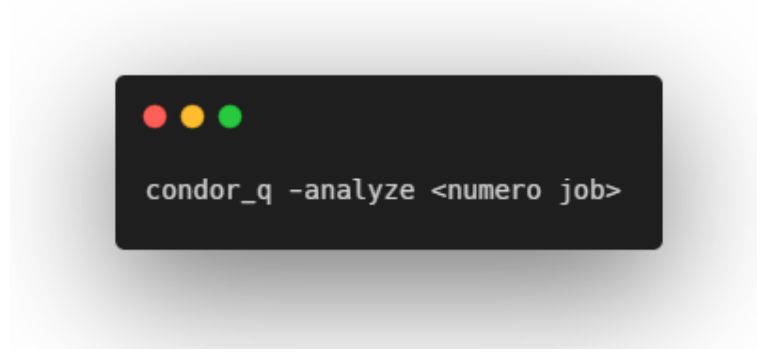


Figura 5.9: Comandi utili..

in questo modo è possibile individuare eventuali errori e correggerli in quanto il feedback fornito è molto utile ed esaustivo.