

Attività di Tirocinio curriculare:

Integrazione di funzionalità su infrastruttura virtuale SLURM

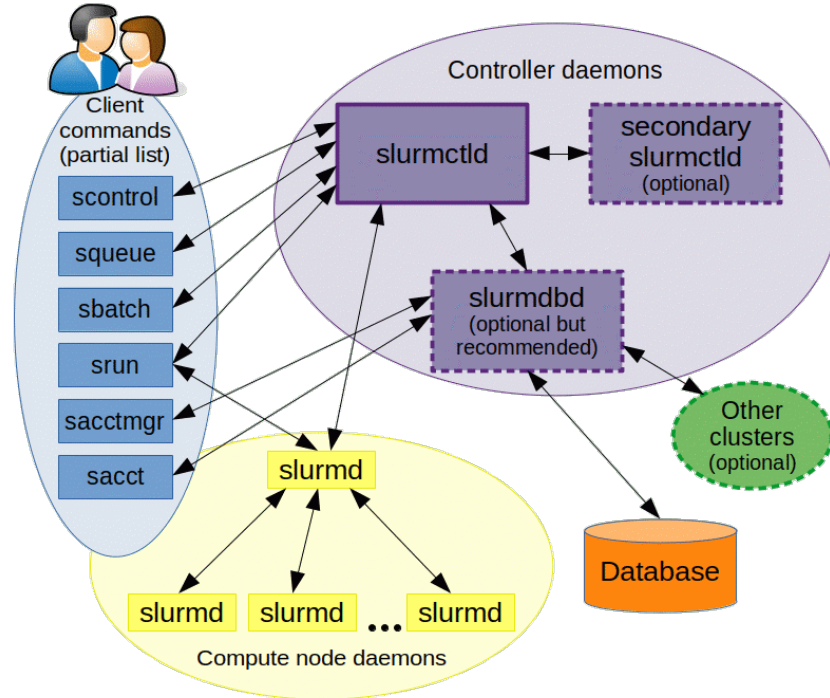
presso:
ULISSE – DISI, Unibo

svolto da:
Massimo Valerio Zerbini

Obiettivi

- *(Impostazione dell'infrastruttura virtuale)*
- **Federazione di 2 cluster SLURM:** coordinamento tra più cluster per l'esecuzione di job
- **Condivisione delle risorse di computazione:** assegnamento dinamico di risorse in base alle necessità dei job da eseguire
- **Priorità di scheduling:** configurazione di una partizione SLURM prioritaria (su una specifica risorsa), accessibile esclusivamente da un determinato utente

Architettura SLURM (Simple Linux Utility for Resource Management)



Ambiente di lavoro

- **Vagrant & Ansible:** virtual *Infrastructure as Code* (IaC), per la configurazione e il management di molteplici VM



- **Repository Git:** copia locale + copia remota sulla piattaforma GitLab del DISI



creazione di un branch di lavoro separato, denominato “*tirocinio*”



Risoluzione DNS

dnsmasq: local DNS resolver + DHCP server



I nodi della LAN eseguono una richiesta DHCP a dnsmasq, che concede un *lease* di un indirizzo IP all'interno di un range, registrando allo stesso tempo l'hostname corrispondente.

In questo modo, i nodi possono richiedere (a dnsmasq) la risoluzione di un hostname nella LAN, ricevendo l'indirizzo IP in risposta.

controller $\xrightarrow{\text{DNS resolution}}$ 192.168.10.23

DB per la registrazione delle attività



- **MariaDB:** *Database Management System* (DBMS) relazionale, basato su MySQL
- **sLurmdbd:** demone responsabile per l'interfacciamento con il DBMS e la registrazione delle attività SLURM


È necessario impostare correttamente i privilegi nel DB, affinché il demone possa svolgere la sua funzione.

sLurmdbd diventa **essenziale** sia per la configurazione della federazione di cluster, sia per l'impostazione della priorità di scheduling.

Cluster singolo

Attivazione del singolo cluster SLURM, costituito da **un nodo controllore** (che esegue `slurmctld` e `slurmdbd`) e **due nodi worker** (che eseguono `slurmd`).

Esempio di sottomissione di un job:

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The prompt is 'vagrant@controller:~\$'. The command 'srun -N2 hostname' is entered and executed. The output shows 'slurm1' on the first line and 'slurm2' on the second line, indicating the job was submitted to two worker nodes.

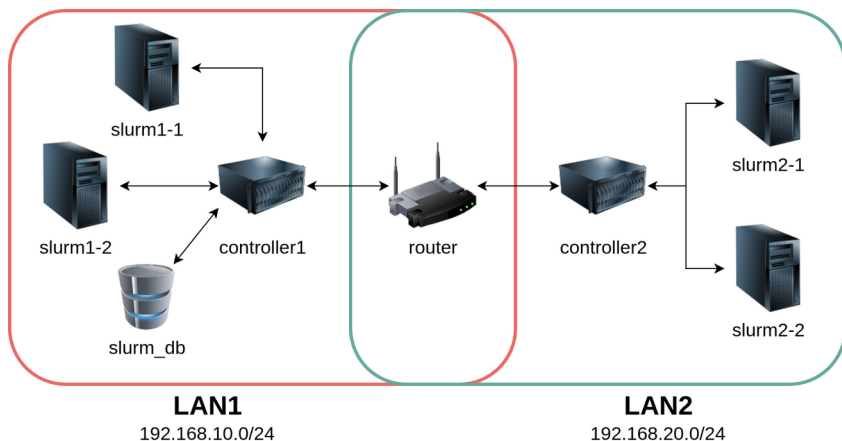
```
vagrant@controller:~$ srun -N2 hostname  
slurm1  
slurm2
```

(il programma `hostname` viene eseguito su entrambi i nodi worker, grazie all'opzione `-N`)

Topologia della federazione

- Introduzione di router e del secondo cluster
- Modifica alle configurazioni di dnsmasq e SLURM
- **Registrazione** della federazione nel DB

NB: i cluster appartenenti a una federazione devono condividere il medesimo DB



```
vagrant@controller1:~$ srun --clusters=cluster2 -N2 hostname  
slurm2-1  
slurm2-2  
vagrant@controller2:~$ srun --clusters=cluster1 -N2 hostname  
slurm1-1  
slurm1-2
```

Esempio di sottomissione di job in una federazione

Condivisione di risorse (CPU, RAM, GPU)

- Attivazione del plugin `select/cons_tres` (*consumable trackable resources*)
- Configurazione aggiuntiva per le **GRES** (*generic resources*), di cui la GPU fa parte

```
vagrant@controller:~$ srun -c1 --mem=256 sleep 60 &
[1] 18254
vagrant@controller:~$ srun -c1 --mem=256 sleep 60 &
[2] 18264
vagrant@controller:~$ srun -c1 --mem=256 sleep 60 &
[3] 18274
vagrant@controller:~$ srun -c1 --mem=256 sleep 60 &
[4] 18284
vagrant@controller:~$ squeue
JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)
1 debug sleep vagrant R 0:15 1 worker
2 debug sleep vagrant R 0:14 1 worker
3 debug sleep vagrant R 0:13 1 worker
4 debug sleep vagrant R 0:12 1 worker
```

Esecuzione parallela di 4 job (ciascuno richiedente 1 CPU e 256 MB di RAM) su un singolo nodo da 4 CPU e 1 GB di RAM

```
vagrant@controller:~$ srun --gpus=1 --mem=256 sleep 60 &
[1] 18294
vagrant@controller:~$ srun --gpus=1 --mem=256 sleep 60 &
[2] 18304
vagrant@controller:~$ squeue
JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)
5 debug sleep vagrant R 0:08 1 worker
6 debug sleep vagrant R 0:07 1 worker
```

Esecuzione parallela di 2 job (ciascuno richiedente 1 GPU) su un singolo nodo da 2 GPU

Priorità di scheduling

- Definizione di una partizione separata, accessibile solo da un determinato utente
- Attivazione del plugin `priority/multifactor` (per impostare la priorità di una partizione su una GPU)
- Registrazione** nel DB della nuova partizione e delle varie associazioni utente

```
vagrant@controller:~$ sudo -u user2 srun --gpus=1 --mem=256 sleep 60 &
[1] 18314
vagrant@controller:~$ sudo -u user3 srun --gpus=1 --mem=256 sleep 60 &
[2] 18324
vagrant@controller:~$ sudo -u user4 srun --gpus=1 --mem=256 sleep 60 &
[3] 18334
vagrant@controller:~$ sudo -u user1 srun --partition=gupart --gpus=1 --mem=256 sleep 60 &
[4] 18344
vagrant@controller:~$ squeue
JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)
9 debug sleep user4 PD 0:00 1 (Resources)
8 debug sleep user3 R 0:03 1 worker
7 debug sleep user2 R 0:17 1 worker
10 gupart sleep user1 PD 0:00 1 (QOSGrpGRES)
```

termine del job 7

```
vagrant@controller:~$ squeue
JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)
9 debug sleep user4 PD 0:00 1 (Resources)
8 debug sleep user3 R 0:50 1 worker
10 gupart sleep user1 R 0:04 1 worker
```

Grazie alla sottomissione nella partizione `gupart`,
il job dell'utente prioritario viene eseguito prima

Job 9 e 10 in contesa di risorse (in particolare, una GPU)