UNIVERSITÀ di VERONA

Centro PIATTAFORME TECNOLOGICHE

HPC e Cloud

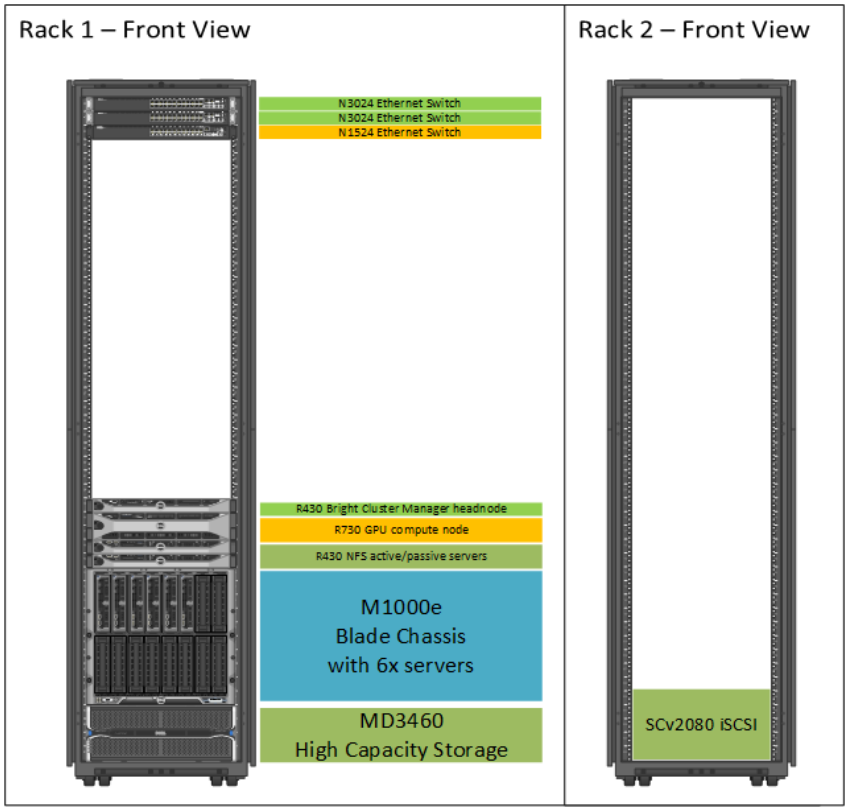
4 giugno 2018

Alberto Sabaini
Anna Perina
Piattaforma Computazionale

ARGOMENTI

- Struttura hw e sw della piattaforma
- HPC
- Cloud

STRUTTURA HW



- 1 headnode
- 6 nodi di calcolo
- 1 nodo gpu
- storage di massa con 2 server in failover
- apparati di rete

STRUTTURA HW

- Totale risorse:
  - 7 nodi
  - 1.7 TB ram
  - 14 cpu / 120 core / 240 thread
    - 8 cpu / 96 core / 192 thread a 2.2 GHz
    - 6 cpu / 24 core / 48 thread a 3.5 GHz
  - gpu 4992 cuda cores
  - 384 TB di spazio

STRUTTURA HW: PROSSIMO UPGRADE

- Il prossimo upgrade comprenderà:
  - 1 headnode
  - 6 nodi di calcolo:
    - 2x cpu 2,6GHz 12C/24T
    - 512GB ram
  - storage veloce: 75TB
  - apparati di rete

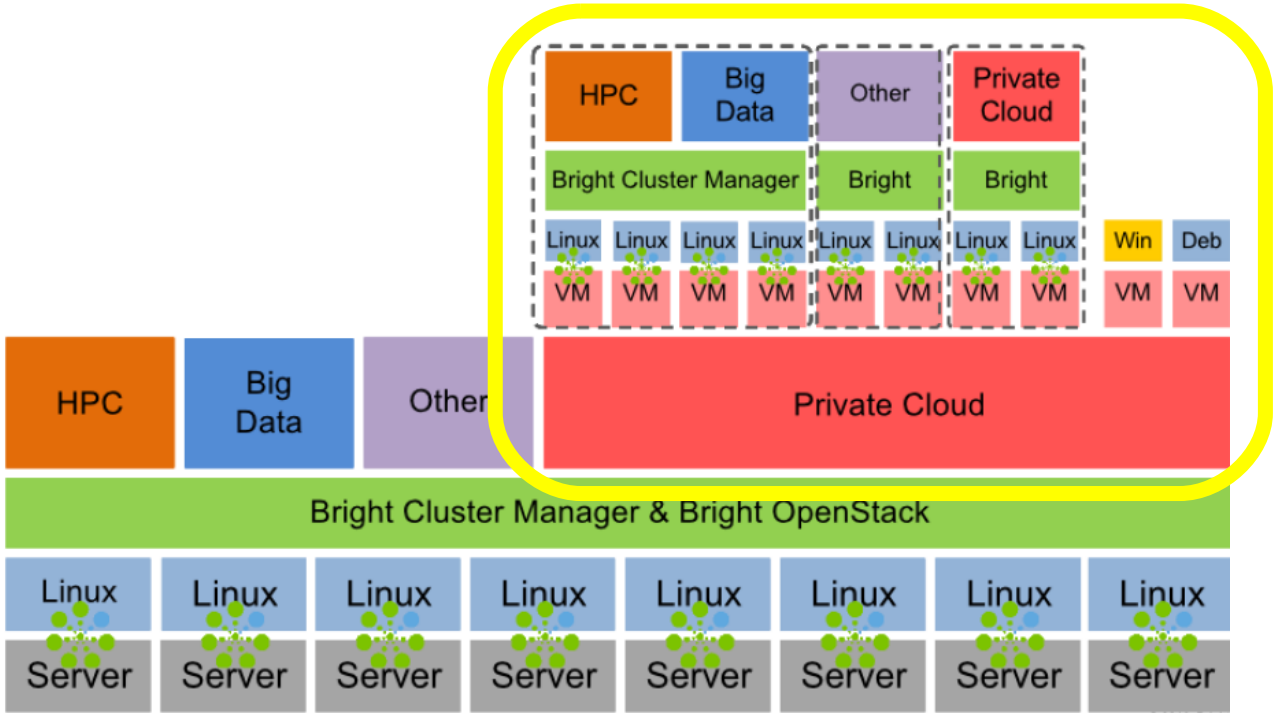
STRUTTURA HW: PROSSIMO UPGRADE

- Totale risorse post upgrade:
  - 13 nodi di calcolo:
  - 4.8 TB ram
  - 26 cpu / 264 core / 528 thread di cui
    - 8 cpu / 96 core / 192 thread a 2.2 GHz
    - 12 cpu / 144 core / 288 thread a 2.6 GHz
    - 6 cpu / 24 core / 48 thread a 3.5 GHz
  - gpu 4992 cuda cores
  - 384 TB di spazio su disco
  - 75 TB di spazio su ssd

ARCHITETTURA SW

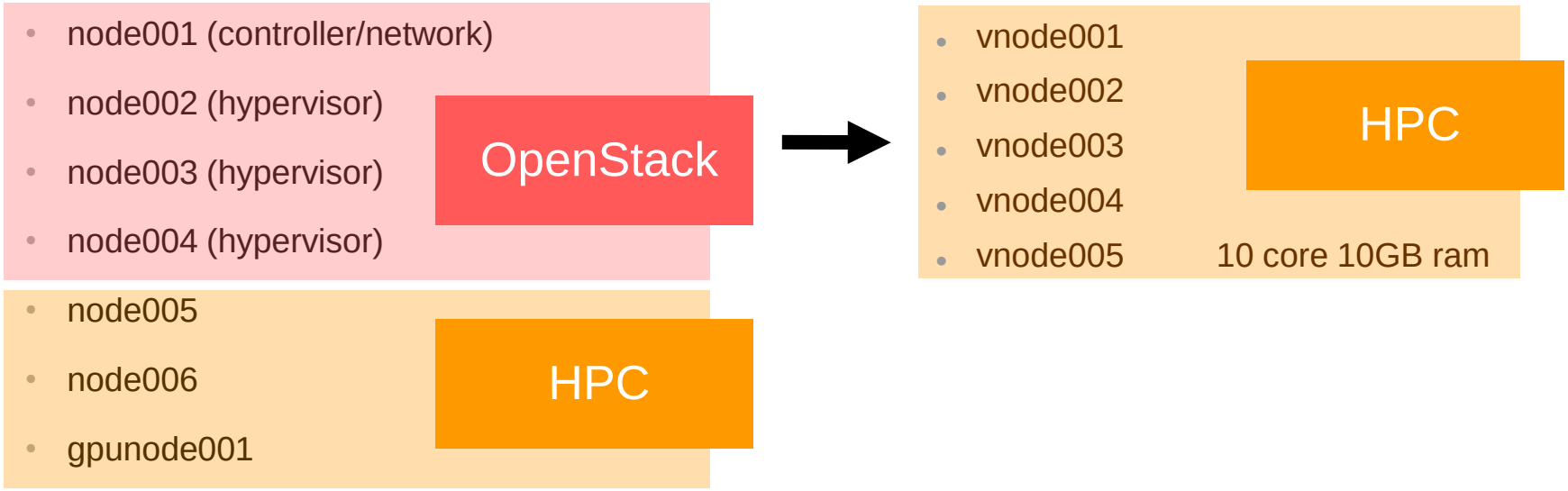
Bright cluster manager e Openstack:

- Permette di gestire l'infrastruttura cluster
- Modulo virtual machine e cloud privata
- Permette creazione di macchine e reti virtuali



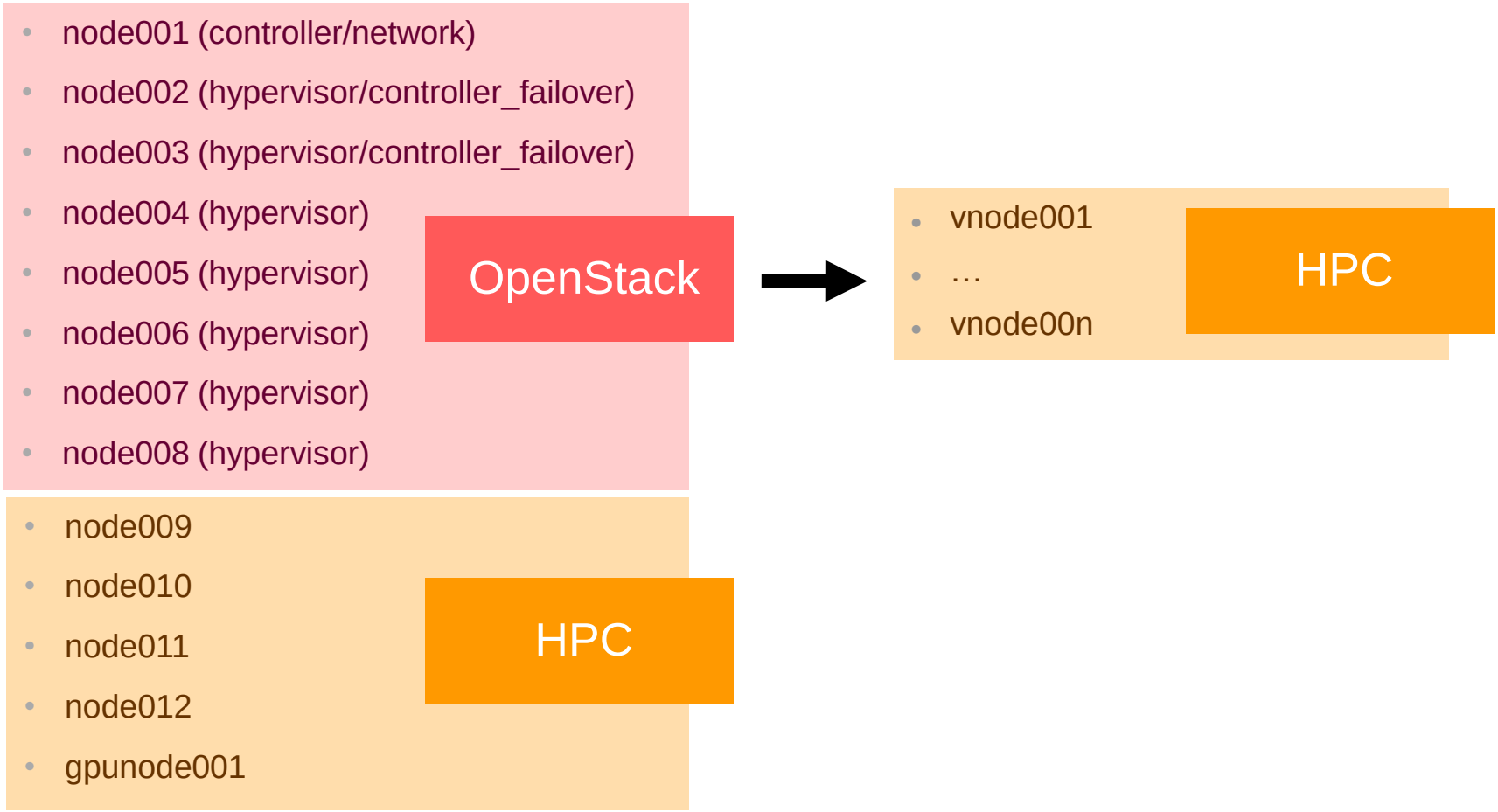
Suddivisione utilizzo nodi

- 4 server blade dedicati ad Openstack (in condivisione al bisogno con HPC): nodi da 1 a 4
- 2 server blade + nodo GPU riservati all'HPC
  - nodo 5: parallelismo (2 x 2.2 GHz 12C/24T)
  - nodo 6: velocità di calcolo (2 x 3.5GHz 4C/8T)



Suddivisione utilizzo nodi post upgrade

- 8 server blade dedicati ad Openstack (in condivisione al bisogno con HPC): nodi da 1 a 4
- 4 server blade + nodo GPU riservati all'HPC



HIGH PERFORMANCE COMPUTING

HPC - DI COSA SI TRATTA

- Calcolo ad elevate prestazioni
- Si realizza unendo le risorse di più macchine e sfruttando il parallelismo per ottenere una maggiore potenza di calcolo
- Permette di eseguire calcoli complessi in meno tempo e con un sistema più robusto

HPC – NODI DI CALCOLO

- 3 nodi fisici:
  - **node005**: 2x 2.2GHz 12C/24T 512 GB RAM → parallelismo
  - **node006**: 2x 3.5GHz 4C/8T 128 GB RAM → velocità
  - **gpunode001:** 2x 2.2GHz 12C/24T 192 GB RAM
    NVIDIA Tesla K80 GPU (4992 cuda cores)
    utilizzabile per il calcolo su CPU o GPU

- 5 nodi virtuali:
  - **vnode001, vnode002,… vnode005**: 10 core, 10 GB RAM, 40 GB disco

HPC – MODALITA' DI ACCESSO

- Richiesta **account**: scrivere da un indirizzo @univr.it
- Richiesta di **software** da installare (nome, versione, link, priorità)
- **Prenotare** risorse (es. nodo)
- **Accesso** alla piattaforma per calcolo HPC tramite:
  ```
  ssh nome.cognome@computing.cpt.univr.it
  ssh -X nome.cognome@computing.cpt.univr.it
  ```
- **Cartella home** a disposizione dell'utente: /home/nome.cognome

computing.cpt@ateneo.univr.it

HPC - MODULI

- I software richiesti dagli utenti vengono installati come **moduli**
- Per visualizzare i sw disponibili:
  `module avail`
- Prima di usare un software e' necessario caricarlo con:
  `module load modulo/versione`
- Vengono visualizzate eventuali istruzioni di utilizzo (se diverse da quelle standard)
- Per visualizzare i moduli caricati:
  `module list`

HPC – GESTORE DEL WORKLOAD: SLURM

- Slurm è il sw che si occupa di gestire i lavori degli utenti: accodare, allocare le risorse, eseguire, sospendere,…
- E' fondamentale **eseguire i sw tramite slurm**
- **srun *[opzioni] comando***:
  - per eseguire un singolo comando
  - alloca le risorse e crea il job
  - esecuzione immediata e output a video
- **sbatch *[opzioni] script***:
  - per eseguire uno o più comandi
  - crea il job e lo passa a slurm
  - esecuzione in background, output su file slurm-<jobid>.out

HPC – STRUTTURA DELLO SCRIPT

- Lo script è composto da:
  - Shebang: `#!/bin/sh`
  - **Dichiarazioni**: `#SBATCH`
  - **Comandi shell**: es. `module load …`
  - **Comandi** per eseguire il sw sul cluster

```
#!/bin/sh
#SBATCH -o my.stdout
#SBATCH --time=30
#SBATCH -N2
#SBATCH --ntasks=8
#SBATCH --ntasks-per-node=4
module load shared java
java -j mylib.jar
srun –p medium blasr file1 file2
```

HPC – OPZIONI DI SRUN / SBATCH

Opzioni principali per richiedere le **risorse** da allocare al job:

- -N : numero di nodi
  - es. -N2 : 2 nodi esatti
  - es. -N2-3 : minimo 2 massimo 3 nodi
- -w o --nodelist : specificare una lista di nodi
- -x o --exclude: escludere una lista di nodi
- -n : numero di task
- --cpu-per-task : numero di cpu per task
- --mem : memoria richiesta per ciascun nodo (in MB)
- --mem-per-cpu : memoria richiesta per cpu (in MB)
- --extra-node-info=S:C:T : specificare il numero di socket (cpu), core e thread
- --exclusive : i nodi non possono essere condivisi con altri job (già in modalità esclusiva)

HPC – OPZIONI DI SRUN / SBATCH

Altre opzioni utili:

- --partition : scegliere la coda
- --begin=<hh:mm:ss> : non iniziare l'esecuzione prima dell'orario indicato
- --time : tempo limite in minuti
- --dependency=<state:jobid> : trattiene il job finché il job indicato non raggiunge lo stato
  - es. --dependency=afterok:1234
- --output=<file> : standard output su file
- --error=<file> : standard error su file
- --overcommit :
  - Se applicato al job (#SBATCH) alloca una sola cpu per nodo per il job (ignora altre indicazioni)
  - Se applicato ad srun esegue più di un processo per cpu

HPC – MONITORAGGIO

- Per visualizzare i lavori in coda o in esecuzione: `squeue`
  - `--users=<utenti>` : filtra sulla lista di utenti
  - `--start` : informazioni sul tempo di inizio previsto e le risorse richieste
  - `--priority` : ordina per priorità
  - `--partition` : filtro sulla partizione (coda)
  - `--state` : filtra sullo stato (PD, R, S)

- Per informazioni sulle partizioni e i nodi: `sinfo`
  - `--Node` : focus sui nodi
  - `--reservation` : mostra le reservation se esistenti
  - `-R` : mostra informazioni sui nodi non allocabili (stati down, drain, fail,...)

HPC – MONITORAGGIO

- Per cancellare un lavoro: `scancel <jobID>`
  - è consentito solo sui propri job
  - `--nodelist=…` : tutti i job su un nodo
  - `--partition=…` : tutti i job in una partizione
  - `--reservation=…` : tutti i job che usano una reservation
  - `--state=…` : tutti i job nello stato specificato
- Per modificare le caratteristiche di un lavoro: `scontrol …`
  - `show job <jobID>` : mostra le caratteristiche del job
  - `update …` : spostare da una coda a un'altra, usare una reservation, cambiare le risorse richieste (numero di nodi, numero di cpu,…)

HPC – MANUALI

- Opzione `--help` per ciascun comando
  - es. `srun --help`
- Pagina di manuale per ciascun comando
  - es. `man srun`

HPC – PRIORITA' E SCHEDULING

- Un lavoro può trovarsi nei seguenti stati:
  - PENDING (PD): è stato lanciato dall'utente ma non è ancora in esecuzione
  - RUNNING (R): in esecuzione
  - SUSPENDED (S): temporaneamente sospeso (es. prelazionato)

HPC – PRIORITA' E SCHEDULING

- Quando un lavoro viene lanciato dall'utente, riceve una **priorità** calcolata in base a questi fattori:
  - **Età**: il tempo di permanenza nella coda dei processi pending
  - **Dimensione**: numero di nodi o cpu richieste per il job
  - **Partizione**: i nodi possono essere suddivisi in partizioni, ciascuna con diversa priorità

- A ciascuno di questi fattori è assegnato un **peso**, configurabile dall'amministratore

HPC – PRIORITA' E SCHEDULING

- La **priorità** è calcolata con la formula seguente:

  Job_priority =

  (PriorityWeightAge) * (age_factor) +

  (PriorityWeightJobSize) * (job_size_factor) +

  (PriorityWeightPartition) * (partition_factor) +

  (PriorityWeightFairshare) * (fair-share_factor) +

  (PriorityWeightQOS) * (QOS_factor) +

  SUM(TRES_weight_cpu * TRES_factor_cpu,

  TRES_weight_<type> * TRES_factor_<type>,

  …)

- La priorità **può variare** finché il job è in stato PENDING.

HPC – CODE E PRELAZIONE

- Abbiamo definito **4 code (partitions)** con diverse caratteristiche:
  - **low** (default): ha priorità minima e nessun limite di durata del job; include anche il nodo gpu
  - **medium**: ha priorità maggiore di low e limite di durata del job di 12 ore
  - **high**: ha la priorità massima e limite di durata del job di 2 ore
  - **gpuq**: esegue i lavori solo sul nodo gpu e ha priorità superiore alle altre tre; nessun limite di tempo
- Allo scadere del tempo limite il job viene **cancellato**
- Per scegliere la coda utilizzare il parametro: `-p nomecoda`

HPC – CODE E PRELAZIONE

- La **priorità delle code** ha un peso molto elevato nel calcolo della priorità di un job
- Un job può essere **prelazionato** da un altro job lanciato su una coda a priorità più alta, salvo priorita' precedentemente acquisita (nello stato PENDING)

Domande?