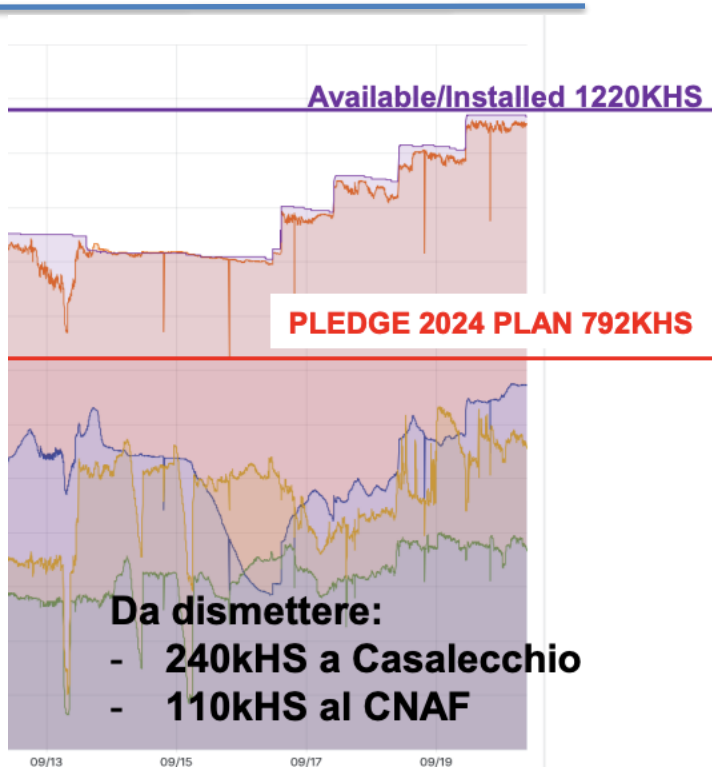
Novità dal reparto farming del T1

Diego Michelotto (diego.michelotto@cnaf.infn.it)
Andrea Chierici (andrea.chierici@cnaf.infn.it)
Daniele Lattanzio (daniele.lattanzio@cnaf.infn.it)
Alessandro Pascolini (alessandro.pascolini@cnaf.infn.it)
Giusy Sergi (giusy.sergi@cnaf.infn.it)

Esperienza con Leonardo

Evoluzione

- Dallo scorso workshop (vedi qui)
  - Immagine e procedura pronta
  - 1 job Leonardo = 1 VM whole node CNAF (HTCondor)
  - Infrastruttura Skyway da terminare
- Passi successivi:
  - Connettività completata **16x100Gb/s**
  - Entrati in **produzione** con 200 nodi **17-19/07/2024**
  - Erogati circa **570 kHEPScore**
  - Aggiornata immagine da CentOS 7 con HTCondor 23 ad Almalinux 9 con HTCondor 24 – aggiornamenti software vari (CVMFS, GPFS, ecc.)



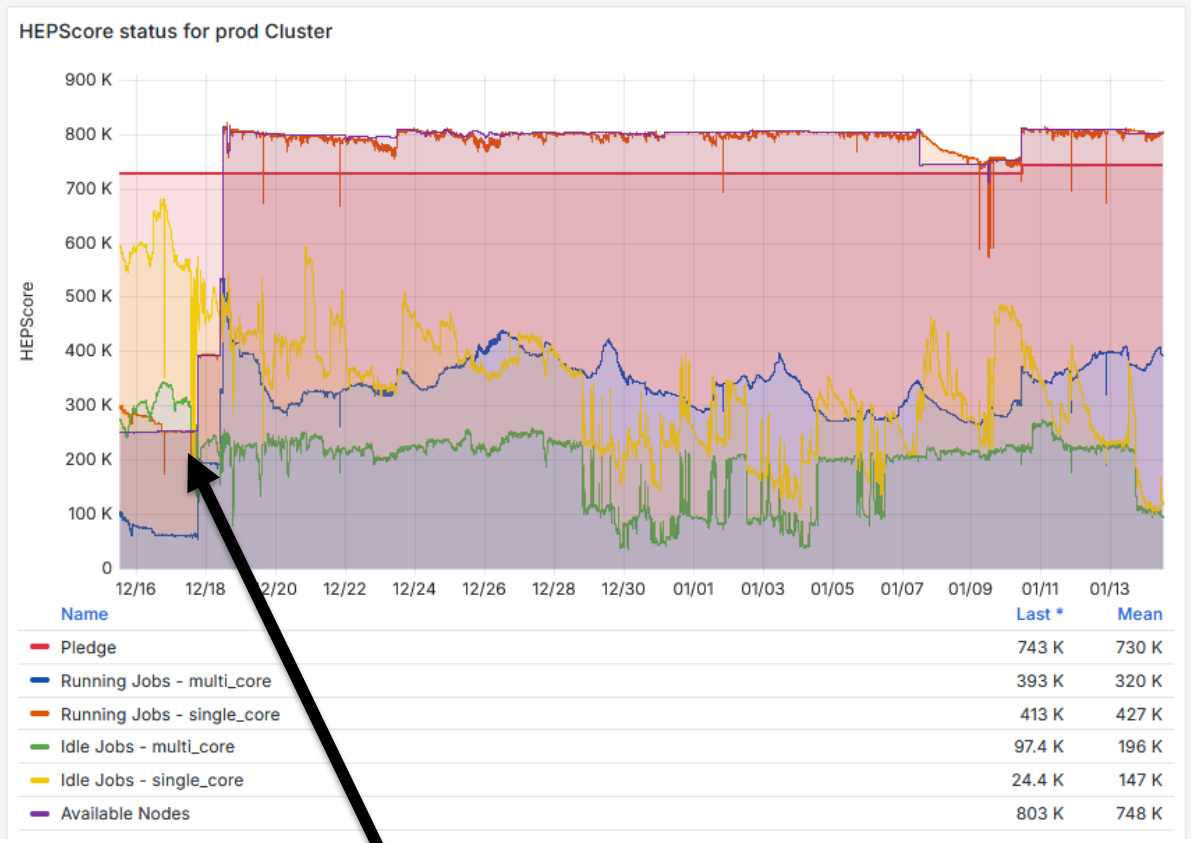
Problematiche varie

- **Instabilità di rete** (Skyway??? Infiniband Dragonfly????)
  - Perdita di PING
  - Job morti perché perdono connettività con Central Manager di HTCondor
  - FS GPFS che si smontano
- **Continui interventi di manutenzione** su Leonardo
  - CINECA esclude dal calcolo delle SLA le manutenzioni programmate
    - Ogni mese una manutenzione programmata

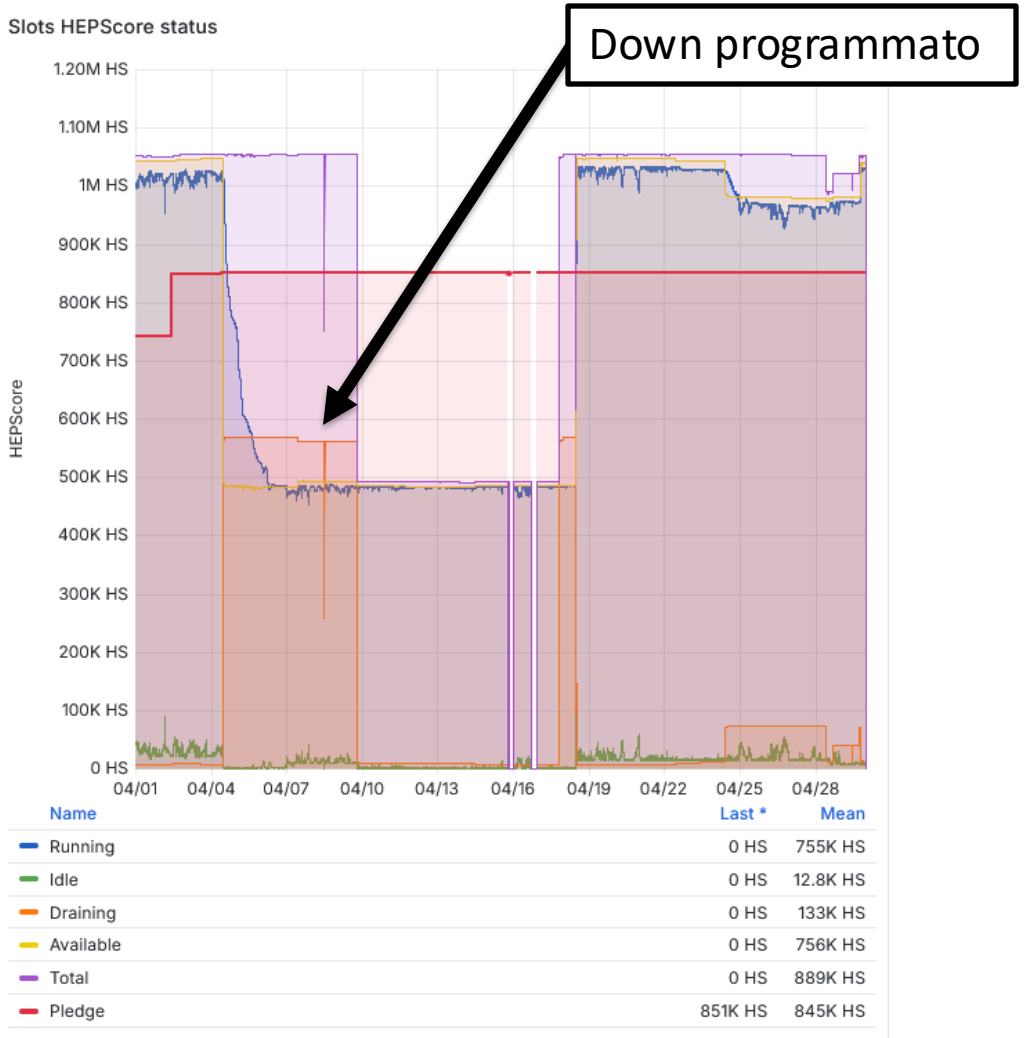
Manutenzioni

- 9 interventi **scheduled**
  - Spesso anche se non interessavano la partizione GP abbiamo **perso** da **qualche decina a tutti i nodi** (aggiornamento Lustre/DDN)
  - 1 intervento di down completo dal 7-16 aprile per aggiornamento sistema di gestione di Leonardo (software di Eviden) → avevamo messo in **drain** in maniera **precauzionale** i nodi → **scelta vincente**
- 7 interventi **unscheduled**
  - Surriscaldamento nodi x2
  - Problemi di alimentazione dei nodi x3

Manutenzioni



Intervento trasparente...

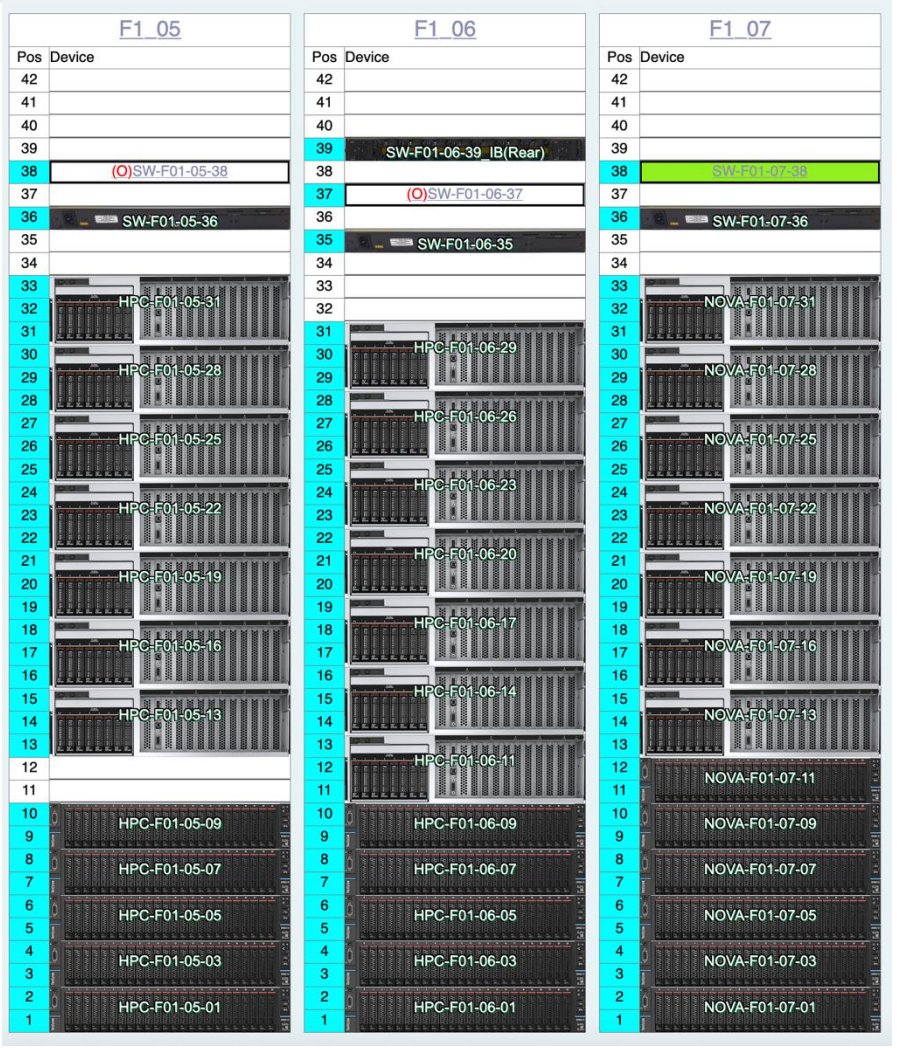


Esperienza con i nodi Terabit

Solo parte computing

Come fornire le risorse?

- A disposizione
  - 3 rack con **21** nodi **GPU** e **16** nodi **CPU**
- Esigenze:
  - Fornire risorse in maniera **efficiente**
  - Spostare di «ruolo» in maniera **dinamica**
- L'idea:
  - Batch system **SLURM** con tutte le risorse
    - Configurazioni ottimizzate
    - Prestazione elevate
  - 1 rack in **Openstack** con **VM «whole node»** che faranno parte del batch
    - Possibilità di assegnare queste risorse tramite Cloud@CNAF e/o INFN Cloud (Primo use case già in produzione)



I nuovi rack con “rear-door”

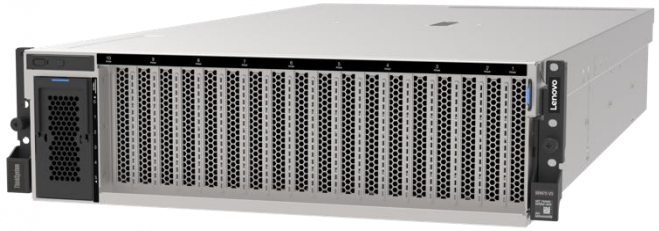
- Avremmo voluto dirvi di più...
  - Sono **ancora da tarare**, serve portare al massimo tutte le machine… ci siamo quasi
- Modello **Liebert DCD47**
- Solido e funzionale
  - Consente di **raffreddare nodi di vendor diversi**
  - Configurato per **dissipare 35kW** di potenza IT
- Stranamente (o forse no!) a SC24 non se ne sono visti tanti → molto DLC

I nuovi rack con “rear-door”

I nodi della gara

- **Nodo CPU**: Lenovo SR665 V3
  - 192 core, 1.5TB RAM, 2x25Gb/s net, 400Gb/s IB
  - 2x1TB nvme + 3x7TB nvme

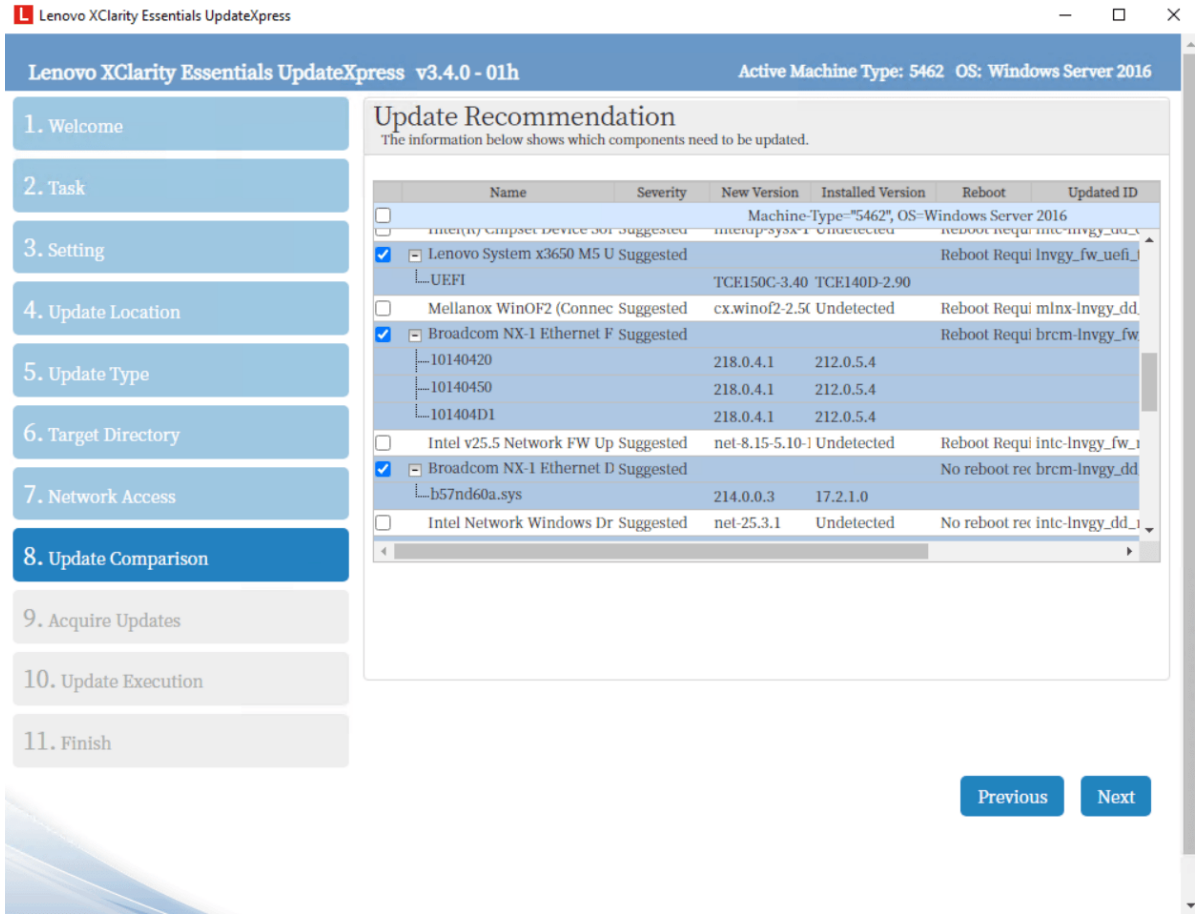
- **Nodo GPU**: Lenovo SR675 V3
  - Uguale al nodo CPU, 2x14TB nvme
  - Board HGX con 4 x H100 SXM5 80GB, nvlink v4



- **Switch Infiniband**: qm9700
  - 32 port NDR 800Gb/s splittate in 2 da NDR 400Gb/s

Per i firmware: Lenovo BoMC

- Consigliamo di utilizzare Lenovo XClarity Essentials **Bootable Media Creator** (BoMC) per creare supporti avviabili adatti per aggiornamenti dei firmware
- Utilizzabili da console remota (**BMC**)
- Processo di creazione su PC Windows



Elementi da configurare

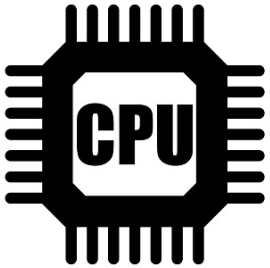
- Ottimizzazione BIOS/UEFI
- Connessione infiniband
- Driver NVIDIA per GPU
- Sottosistema MPI HPCX

Ottimizzazione BIOS

- Per evitare di intervenire manualmente sui singoli nodi, utilizzare l'utility **OneCLI di Lenovo**.
  - rpm -Uvh https://download.lenovo.com/servers/mig/2025/05/06/62436/lnvgy_utl_lxcer_onecli01x-5.2.0_linux_indiv.rpm
  - Tramite comandi **da shell** è possibile intervenire direttamente
  - Necessario comunque un **reboot** al termine delle operazioni
  - Le ottimizzazioni intervengono su diversi aspetti, e potrebbero causare aumento del regime rotazione delle ventole

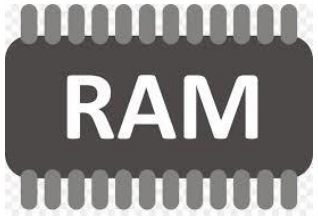
Ottimizzazioni suggerite

 # Prerequisito
OperatingModes.ChooseOperatingMode "Custom Mode"

 # CPU e NUMA


Processors.DeterminismSlider Performance
Processors.CorePerformanceBoost Enabled
Processors.GlobalC-stateControl Disabled
Processors.SMTMode Disabled
Processors.ACPISRATL3CacheasNUMADomain Enabled
Memory.NUMANodesperSocket NPS1

 # Prefetcher, migliorano throughput CPU / GPU in molti workload
Processors.L1StreamHWPrefetcher Enabled
Processors.L2StreamHWPrefetcher Enabled
Processors.L1StridePrefetcher Enabled
Processors.L1RegionPrefetcher Enabled
Processors.L2UpDownPrefetcher Enabled

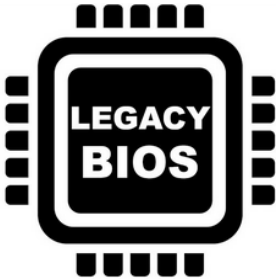
 # Memory tuning


Memory.MemorySpeed Maximum
Memory.DRAMScrubTime Disabled

 # Alimentazione e Power States
Power.PCIePowerBrake Disabled
Power.EfficiencyMode Disabled

 # PCIe e IOMMU


DevicesandIOPorts.IOMMU Enabled
DevicesandIOPorts.SRIOV Enabled
DevicesandIOPorts.PCI64BitResourceAllocation Enabled
DevicesandIOPorts.PCIeTenBitTagSupport Enabled

 # Opzioni legacy inutili


- LegacyBIOS.LegacyBIOS Disabled

/opt/lenovo/lnvgy-utl-lxce-onecli/onecli config set <comando>

Rete Infiniband

- Switch NVIDIA qm9700
- attivare subnet manager
- Selezionare «updn» come routing engine
- Impostare tra 30 e 60 il parametro
  - Advanced SM: Number of seconds between subnet sweeps (0 to disable)
- Attenzione agli update di firmware, usare solo quelli “Lenovo”
  - Anche per le schede infiniband dei nodi!
  - https://network.nvidia.com/support/firmware/lenovo-intelligent-cluster/

Home » Support » OEM Firmware Downloads

OEM Firmware Downloads

OEM Partnerships

Mellanox products that are sold by systems OEMs may have differer products. As a service to our OEM partners and their customers, we releases for these products.

DELL EMC

Dell EMC

Lenovo

Lenovo

IBM

IBM

Driver Infiniband

- DOCA-Host vs MLNX_OFED
  - MLNX_OFED has transitioned into DOCA-Host, and now is available as DOCA-OFED profile
  - MLNX_OFED last standalone release is October 2024 Long Term Support (3 years). Starting January 2025 all new features will only be included in DOCA-OFED.

- Sarebbero consigliati quindi i driver DOCA rispetto agli OFED per nuove installazioni come le nostre…
  - …se riuscite a farli andare, fateci sapere!

MLNX_OFED

- Creare repo per rpm

  [mlnx-ofed]

  name=OFED Online Repo

  baseurl=https://linux.mellanox.com/public/repo/mlnx_ofed/24.10-2.1.8.0/rhel9.5/x86_64/

  enabled=1

- dnf remove –y openmpi
- dnf install –y mlnx-ofed-hpc
  - vengono creati i moduli da dkms,

    normale ci voglia molto tempo

Controlli su installazione

- /etc/init.d/openibd viene creato durante installazione
  - status → fornisce dettagli su device e moduli caricati
- ibstat → State: Active
- ibhosts → mostra l'elenco dei nodi nella rete IB
  - gli host con driver "aggiornati" vengono mostrati con HCA-3, quelli non aggiornati, con mlnx5_2
- ibv_devinfo → link_layer deve risultare "Infiniband", MTU 4096, state "PORT_ACTIVE"
- ofed_info –s → restituisce info su versione driver

Test prestazioni

- Si fa tra un server e un client
  - server: ib_write_bw -d mlx5_2
  - client: ib_write_bw -d mlx5_2 <hostname_server>
- Output fornisce **throughput massimo e medio**
- Per avere prestazioni vicine ai valori di targa, occorre parallelizzare
  - apporre –q 8 ai due comandi precedenti
- Se, come probabile, ancora non basta, occorre intervenire su binding numa

Binding numa: raggiungere i 400Gbit

- Se non già presente installare il pacchetto numactl
  - potrebbe essere necessario creare un link simbolico ad una libreria
  - cd /usr/lib64/; ln -s libnuma.so.1 libnuma.so
- cat /sys/class/infiniband/mlx5_2/device/numa_node
  - mostra il numa node a cui siamo collegati (probabile sia “0”)
- lscpu | grep -i numa
  - mostra quali core sono collegati a quale nodo numa (sono “0” o “1”)
- Ai comandi precedenti ora possiamo aggiungere la configurazione specifica per il nostro caso (per nodi CPU), apporre:
  - numactl --physcpubind=0-95 --membind=0 ib_write_bw ...

    --physcpubind usa solo i core vicini alla scheda

    --membind alloca la memoria solo sul numa node dove ha accesso veloce la scheda

Se le prestazioni non decollano

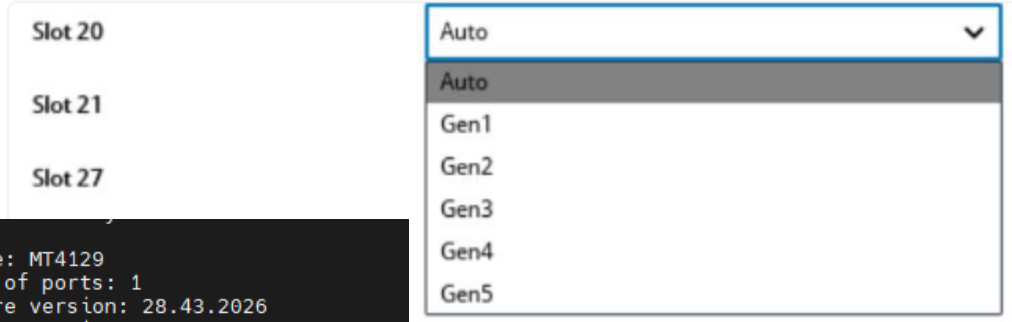
- Probabile collo di bottiglia sullo slot PCIe della scheda (per i nodi GPU)
- lspci|grep Conne → mostra la connessione della scheda
- Nodo CPU: lspci -s 41:00.0 -vv|grep -i lnksta

  LnkSta: Speed 32GT/s (ok), Width x16 (ok) → 512Gb/s
- Nodo GPU (scheda su slot 21): lspci -s a1:00.0 -vv|grep -i lnksta

  LnkSta: Speed 16GT/s (downgraded), Width x16 (ok) → 256Gb/s

Il setup di Terabit

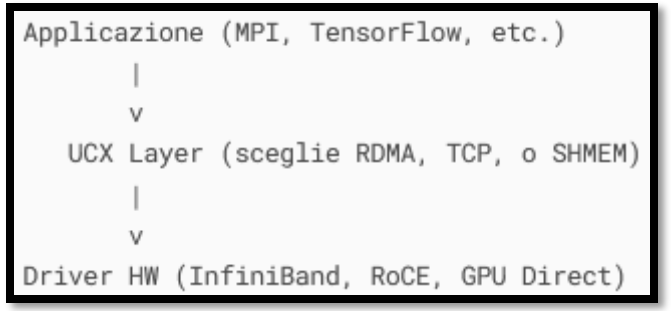
- I server con GPU hanno **solo uno slot PCI gen5**
  - è individuato dal **numero 20** nella documentazione ufficiale Lenovo
- Le schede **inizialmente erano su slot frontale**
  - al CNAF, **reinstallate su slot libero sbagliato**
  - Prestazioni **non ottimali**
- Tutte le schede di rete sono Mellanox!
  - Questo crea confusione se non si specifica il device ad alcuni comandi
  - Nel caso, usare mlx5_2




```
CA 'mlx5_2'
        CA type: MT4129
        Number of ports: 1
        Firmware version: 28.43.2026
        Hardware version: 0
        Node GUID: 0xa088c203005ec2b0
        System image GUID: 0xa088c203005ec2b0
        Port 1:
                State: Active
                Physical state: LinkUp
                Rate: 400
                Base lid: 26
                LMC: 0
                SM lid: 12
                Capability mask: 0xa751e848
                Port GUID: 0xa088c203005ec2b0
                Link layer: InfiniBand
```

Subnet manager e altri dettagli

- Subnet manager: componente critico in una rete InfiniBand
  - Inizializzazione, QoS, monitoraggio
  - sullo switch semplifichiamo la gestione e riduciamo la latenza
- Per i nostri casi d'uso non è probabilmente necessario configurare IPoIB
- RDMA: Remote Direct Memory Access
  - OpenMPI (e HPCX) sono ottimizzate per RDMA
  - Se si usa UCX (altamente probabile) non dobbiamo fare altro
    - UCX: Unified Communication X (UCX) is an **optimized point-to-point communication framework**.



Driver NVIDIA per GPU

- Dal sito NVIDIA: two “flavors” of kernel modules are provided
  - **Proprietary** - This is the flavor that NVIDIA has historically shipped. For older GPUs from the Maxwell, Pascal, or Volta architectures.
  - **Open-source** - Published kernel modules that are dual licensed MIT/GPLv2. These are only for Turing and newer architectures, and this is what you should use if you have one of those architectures.
  - Starting in the **560** driver release series, the open kernel module flavor is the default installation
  - Open-source non significa nouveau!

Installare i driver NVIDIA

- Quanto esposto in precedenza non funziona!
- Per esperienza diretta, funzionano solo i driver “proprietary”
- Creare il repo

```
[nvidia]
name=NVIDIA Online Repo
baseurl=https://developer.download.nvidia.com/compute/cuda/repos/rhel9/x86_64
enabled=1
```

```
dnf install kernel-devel-matched kernel-headers
dnf module install -y nvidia-driver:latest-dkms
dnf install –y cuda-toolkit
```

Grazie Massimo e Sergio!

GDRCopy

- L'acronimo sta per GPU Direct RDMA Copy
  - Enable faster memory transfers between CPU and GPU with GDRCopy
  - low-latency GPU memory copy library based on GPUDirect RDMA technology that allows the CPU to directly map and access GPU memory.
  - GDRCopy also provides optimized copy APIs and is widely used in high-performance communication runtimes like UCX, OpenMPI, MVAPICH, and NVSHMEM
- Si tratta di un modulo kernel e di un rpm
  - Si scarica il sorgente da cui si ottengono gli rpm, dopo compilazione: https://github.com/NVIDIA/gdrcopy

```
[root@hpc-f01-05-31 ~]# ucx_info -d | grep gdr
# Memory domain: gdr_copy
#     Component: gdr_copy
#      Transport: gdr_copy
```

```
[root@hpc-f01-05-28 ~]# lsmod|grep gdr
gdrdrv                 45056  0
nvidia              104693760  3 nvidia_uvm,gdrdrv,nvidia_modeset
```

Sottosistema MPI HPC-X

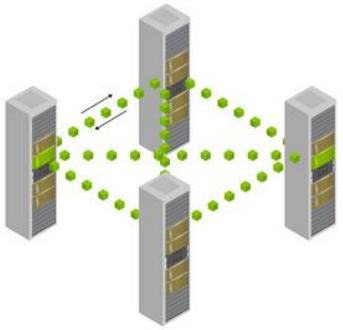
- Di cosa si tratta:

HPC-X Download Center

CURRENT VERSIONS | ARCHIVE VERSIONS | START OVER

Version Current	DOCA-OFED/MLNX_OFED /OFED	DOCA-OFED/MLNX_OFED /OFED Ver.	OS Distro	OS Distrp Ver.	Arch.	Download/ Documentation
2.23-CUDA12.x	inbox	mlnx_ofed	Debian	7.x	aarch64	tbz2: hpcx-v2.18.1-gcc-mlnx_ofed-redhat9-cuda12-x86_64.tbz
2.21.2-CUDA12.x-LTS	mlnx_ofed		RHEL/CentOS/Rocky	8.x	x86_64	Size: 318M
2.18.1-CUDA12.x-LTS			Ubuntu	9.x		SHA256: cbdf61cb4e43d443ffbf6a091800889b7af92ee0358c00ea536a193708002a54
2.14.0-CUDA11			kylin			
2.13.1-LTS			openeuler			

  - NVIDIA HPC-X is a comprehensive software package that includes Message Passing Interface
  - HPC-X takes advantage of InfiniBand hardware-based networking acceleration engines to maximize application performance
  - Versione specifica per accoppiata mlnx_ofed e CUDA12.x
  - https://developer.nvidia.com/networking/hpc-x/



Configurazione HPC-X

- scaricare il pacchetto e scompattarlo sotto /opt
- Mantenere la dir con il nome originale, creare link simbolico a “hpcx”
- Creare il file /etc/profile.d/hpcx.sh →

File probabilmente ancora non ottimale, collaboriamo!

```
# HPC-X Environment
export HPCX_HOME=/opt/hpcx
export LD_PRELOAD="/usr/lib64/libgdrapi.so"
export
LD_LIBRARY_PATH=/opt/hpcx/ucx/lib:$LD_LIBRARY_PATH
source $HPCX_HOME/hpcx-init.sh
hpcx_load

export UCX_MEMTYPE_CACHE=y
export UCX_RNDV_THRESH=8192
export UCX_CUDA_IPC_CACHE=y

# infiniband
export UCX_NET_DEVICES=mlx5_2:1
export
UCX_TLS=cuda_copy,cuda_ipc,gdr_copy,rc,ud,self,sm

# MPI config
export OMPI_MCA_pml=ucx
export OMPI_MCA_btl=^openib
export OMPI_MCA_osc=ucx
```

Configurazione ambiente MPI

- **Consentire login passwordless** tra gli host
- creare hostfile con hostname e slot disponibili

  hpc-f01-05-31 slots=4

  hpc-f01-05-28 slots=4

  hpc-f01-05-25 slots=4

  ...

- semplice test su più nodi (da utente non privilegiato)

  mpirun -np 12 -hostfile hostfile /opt/hpcx/ompi/tests/examples/hello_c

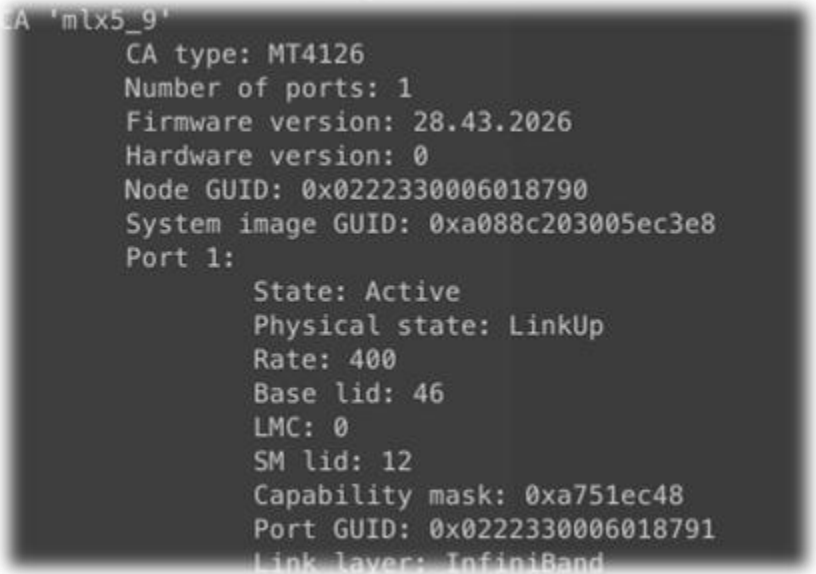
Abilitare nodi cloud

- Le risorse Terabit in cloud ospiteranno **VM** grandi quanto il nodo (**whole node**) con assegnate tutte le risorse (**IB e GPU**)
  - La VM sarà configurata con tutto il software che abbiamo già visto
  - Le VM faranno parte del batch system SLURM

- Possibilità di **cambiare scopo in maniera veloce** alle risorse
  - Se arriva richiesta di risorse Terabit da un utente cloud, si rimuove il nodo da SLURM, si cancella la VM whole node e si assegnano le risorse liberate all'utente.

Abilitare nodi cloud: SR-IOV

- **Necessario virtualizzare le schede Infiniband** tramite SR-IOV (Single Root I/O Virtualization)
  - **Configurazione BIOS** fatta in precedenza, vedi le prime slide
  - Installare **driver MLNX-OFED** sul nodo host
  - **Abilitare la scheda**
    - mst start
    - mst status → si recupera il device
    - mlxconfig -d /dev/mst/mt4129_pciconf0 set SRIOV_EN=1 NUM_OF_VFS=8
  - **Abilitare 8 VF** (Virtual Function) per ogni nodo
    - 1 VF associata ad 1 VM tramite pci-passthrough
    - **Abilitazione al boot e setting port and node GUID**
      - Crezione di unit systemd che esegue questo script al boot https://gist.github.com/koallen/32709a244d77a2c0f8e17ed79a4092ed
    - Dopo il reboot ibstat deve dare «**State Active**» anche su tutte le VF



Abilitare nodi cloud: Switch IB

- Per permettere la **corretta connessione delle interfacce virtuali di IB** è necessario abilitare il Subnet Manager sullo switch
  - switch > enable
  - switch # config terminal
  - ib sm virt enable
  - configuration write
  - reload

Abilitare nodi cloud: Openstack

- **Abilitare scheduler PciPassthroughFilter**
  - Creare **alias** per IB e H100
    - alias = {"name":"H100","product_id":"2330","vendor_id":"10de","device_type":"type-PF"}
    - alias = {"name":"IB","product_id":"101e","vendor_id":"15b3","device_type":"type-VF"}
- Creare **flavor**
  - **1 whole node con Infiniband**
    - openstack flavor create --vcpus 192 --ram 1572864 --disk 100 --property "pci_passthrough:alias"="IB:1" hpc.ib
  - **1 whole node con Infiniband e 4 GPU**
    - openstack flavor create --vcpus 192 --ram 1572864 --disk 100 --property "pci_passthrough:alias"="IB:1,H100:4" hpc.ib.gpu
- Creare **immagine**
  - **Type q35** (per numero cpu elevato, per bus PCIe)
    - openstack image create --public --disk-format raw --container-format bare --file a9.raw almalinux-9-CNAF-x86_64-pcie --property hw_machine_type=q35
  - L'immagine deve contenere i driver nvidia per GPU e i driver mlnx_ofed per IB

Verso la

Compiliamo Linpack

- Scaricare il benchmark qui: https://www.netlib.org/benchmark/hpl/
- Per compilare installare librerie openblas da repo CRB:

  dnf install openblas-devel
- creare il file make.terabit secondo quanto indicato →
- compilare con:

  make arch=terabit

make.terabit

```
SHELL        = /bin/sh
CD           = cd
CP           = cp
LN_S         = ln -fs
MKDIR        = mkdir -p
RM           = /bin/rm -f
TOUCH        = touch
ARCH         = $(arch)
TOPdir       = $(HOME)/hpl-2.3
INCdir       = $(TOPdir)/include
BINdir       = $(TOPdir)/bin/$(ARCH)
LIBdir       = $(TOPdir)/lib/$(ARCH)
HPLlib       = $(LIBdir)/libhpl.a
LAdir       = /usr
LAlib       = -L$(LAdir)/lib64 -lopenblas -lpthread -lm
LAinc       = $(LAdir)/include/openblas

F2CDEFS      = -DAdd__ -DF77_INTEGER=int -DStringSunStyle
HPL_INCLUDES = -I$(INCdir) -I$(INCdir)/$(ARCH) -I$(LAinc) $(MPinc)
HPL_LIBS     = $(LIBdir)/libhpl.a $(LAlib) $(MPlib) -lm -lgfortran
HPL_OPTS     = -DHPL_DETAILED_TIMING -DHPL_PROGRESS_REPORT
HPL_DEFS     = $(F2CDEFS) $(HPL_OPTS) $(HPL_INCLUDES)
CC       = mpicc
CCNOOPT  = $(HPL_INCLUDES) $(F2CDEFS)
OMP_DEFS = -qopenmp
CCFLAGS = -O3 -funroll-loops -fomit-frame-pointer -Wno-unused-result
$(HPL_INCLUDES) $(F2CDEFS)

LINKER       = $(CC)
LINKFLAGS    = $(CCFLAGS)
ARCHIVER     = ar
ARFLAGS      = r
RANLIB       = echo
```

Linpack: primo test

- Attenzione al file HPL.dat che deve essere ben formato!
- deve essere nella stessa dir del benchmark
- Lanciare semplicemente il commando:

  ./xhpl

```
HPLinpack benchmark input file
Innovative Computing Laboratory, University of Tennessee
HPL.out      output file name (if any)
6            device out (6=stdout,7=stderr,file)
1            # of problems sizes (N)
1024         Ns
1            # of NBs
128          NBs
0            PMAP process mapping (0=Row-,1=Column-major)
1            # of process grids (P x Q)
1            Ps
1            Qs
16.0         threshold
1            # of panel fact
2            PFACTs (0=left, 1=Crout, 2=Right)
1            # of recursive stopping criterium
4            NBMINs (>= 1)
1            # of panels in recursion
2            NDIVs
1            # of recursive panel fact.
1            RFACTs (0=left, 1=Crout, 2=Right)
1            # of broadcast
1            BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
1            # of lookahead depth
0            DEPTHs (>=0)
2            SWAP (0=bin-exch,1=long,2=mix)
64           swapping threshold
0            L1 in (0=transposed,1=no-transposed) form
0            U  in (0=transposed,1=no-transposed) form
1            Equilibration (0=no,1=yes)
8            memory alignment in double (> 0)
```

HPL.dat

```
================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR01C2R4        1024   128     1     1               0.02             3.9028e+01
HPL_pdgesv() start time Thu May 22 17:08:46 2025
```

HPL.dat

- E’ il **cuore del nostro benchmark**
- Deve essere il medesimo su tutti i nodi!
- Alcuni parametri da considerare attentamente
  - Ns è la **dimensione della matrice** quadrata N x N da risolvere
    - Ns ≈ sqrt((GBram * 1e9) / 8), per i nostri nodi in cui ram 1.5TB → Ns ≈ 433000
- P x Q = griglia logica di processi MPI → **numero di processi**
  - 16 x 12 ci permette di saturare le CPU dei nodi terabit
- NB dimensione dei blocchi usati nella decomposizione: si usano valori tra 128 e 512

Iniziamo a parallelizzare

- linpack.sh:

```
numactl --interleave=all \
mpirun -np 192 \
--hostfile ./hostfile \
--mca btl ^tcp \
--mca pml ucx \
--mca osc ucx \
./xhpl
```

disabilita il layer TCP (inutile con InfiniBand).

forza l'uso del transport UCX

per sincronizzazioni RDMA-based

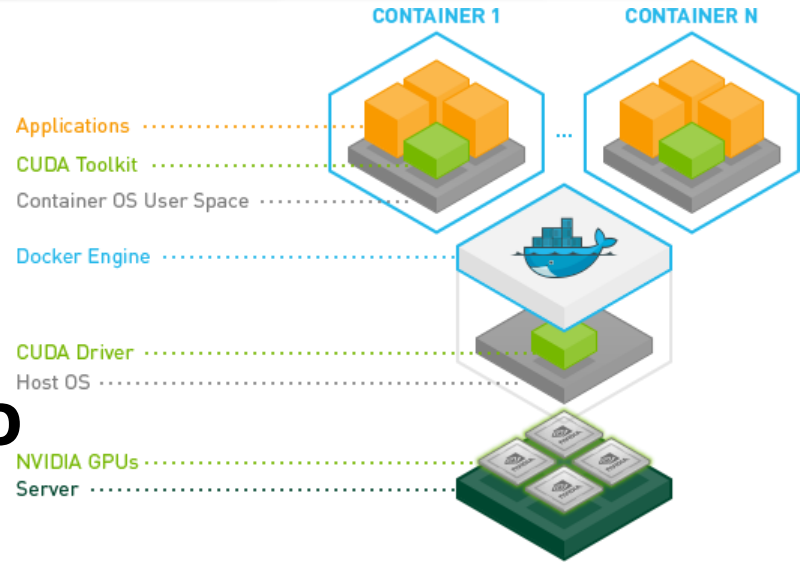
```
HPLinpack benchmark input file
Innovative Computing Laboratory, University of Tennessee
HPL.out      output file name (if any)
6            device out (6=stdout,7=stderr,file)
1            # of problems sizes (N)
400000       Ns
1            # of NBs
256          NBs
0            PMAP process mapping (0=Row-,1=Column-major)
1            # of process grids (P x Q)
12           Ps
16           Qs
16.0         threshold
1            # of panel fact
2            PFACTs (0=left, 1=Crout, 2=Right)
1            # of recursive stopping criterium
4            NBMINs (>= 1)
1            # of panels in recursion
2            NDIVs
1            # of recursive panel fact.
2            RFACTs (0=left, 1=Crout, 2=Right)
1            # of broadcast
1            BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Ln,5=LnM)
1            # of lookahead depth
1            DEPTHs (>=0)
2            SWAP (0=bin-exch,1=long,2=mix)
64           swapping threshold
0            L1 in (0=transposed,1=no-transposed) form
0            U  in (0=transposed,1=no-transposed) form
1            Equilibration (0=no,1=yes)
8            memory alignment in double (> 0)
```

E le GPU??

- hpl è **CPU-only**, e come abbiamo visto, basato su BLAS
- Abbiamo bisogno di qualcosa d'altro per i nodi con GPU
- **I benchmark si devono usare separatamente**, ciascuno sul nodo giusto
- Il risultato **totale** si ottiene **sommando i punteggi** ottenuti nei due sottoinsiemi

NVIDIA container

- **NVIDIA** ha fatto il lavoro per noi
- Da rhel9x **rende disponibile container** specifici sul proprio registry
- Occorre creare un account gratuito su **nvcr.io**
- Qui la pagina di **riferimento** per il container
  - https://catalog.ngc.nvidia.com/orgs/nvidia/containers/hpc-benchmarks



- Tralasciamo tutti i dettagli, indichiamo solo alcuni esempi
  - Creeremo una pagina confluence con i passaggi dettagliati

hpc-benchmarks:25.04

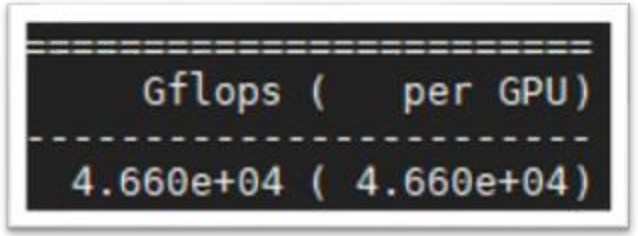
- Esecuzione del container in interattivo:

  docker run --gpus all --ipc=host --ulimit memlock=-1 --ulimit stack=67108864 -it nvcr.io/nvidia/hpc-benchmarks:25.04

- Una volta dentro:

  cd /workspace

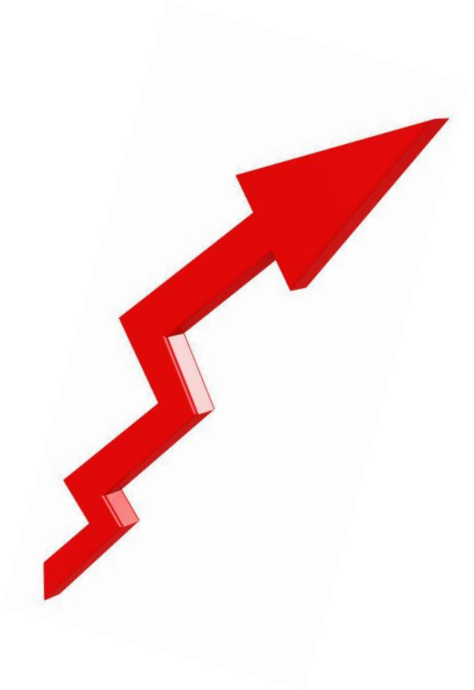
  ./hpl.sh hpl-linux-x86_64/sample-dat/HPL-1GPU.dat



- **Risultato da ottimizzare, sotto le aspettative**.

Top500 quindi?

- Ci abbiamo provato...
- Non siamo arrivati in tempo
  - Problemi di varia natura
  - inesperienza



- Ora abbiamo quasi tutto pronto

- Ad una prossima riunione vi aggiorneremo, se interessa

Concludendo

- Nell'ultimo periodo sono cambiate molte cose nel nostro reparto, siamo attivi su più fronti e questo è stimolante
- L'uso di Leonardo ci «costringe» a dipendere per manutenzione da esterni
  - Non abbiamo le stesse SLA, al momento però nessuno si è lamentato in modo particolare
- Riuscire a «spremere» il massimo dai nodi terabit è complesso
  - In HTC siamo abituati al nodo «plug&play»