# *Supporto universale alle GPU in un cloud Openstack/Juju: configurazione e performance*

Sergio Rabellino
Università di Torino
Dip. di Informatica, Laboratorio HPC4AI

WORKSHOP GARR 2021

NET MAKERS

UNIVERSITÀ DEGLI STUDI DI TORINO

HPC4AI

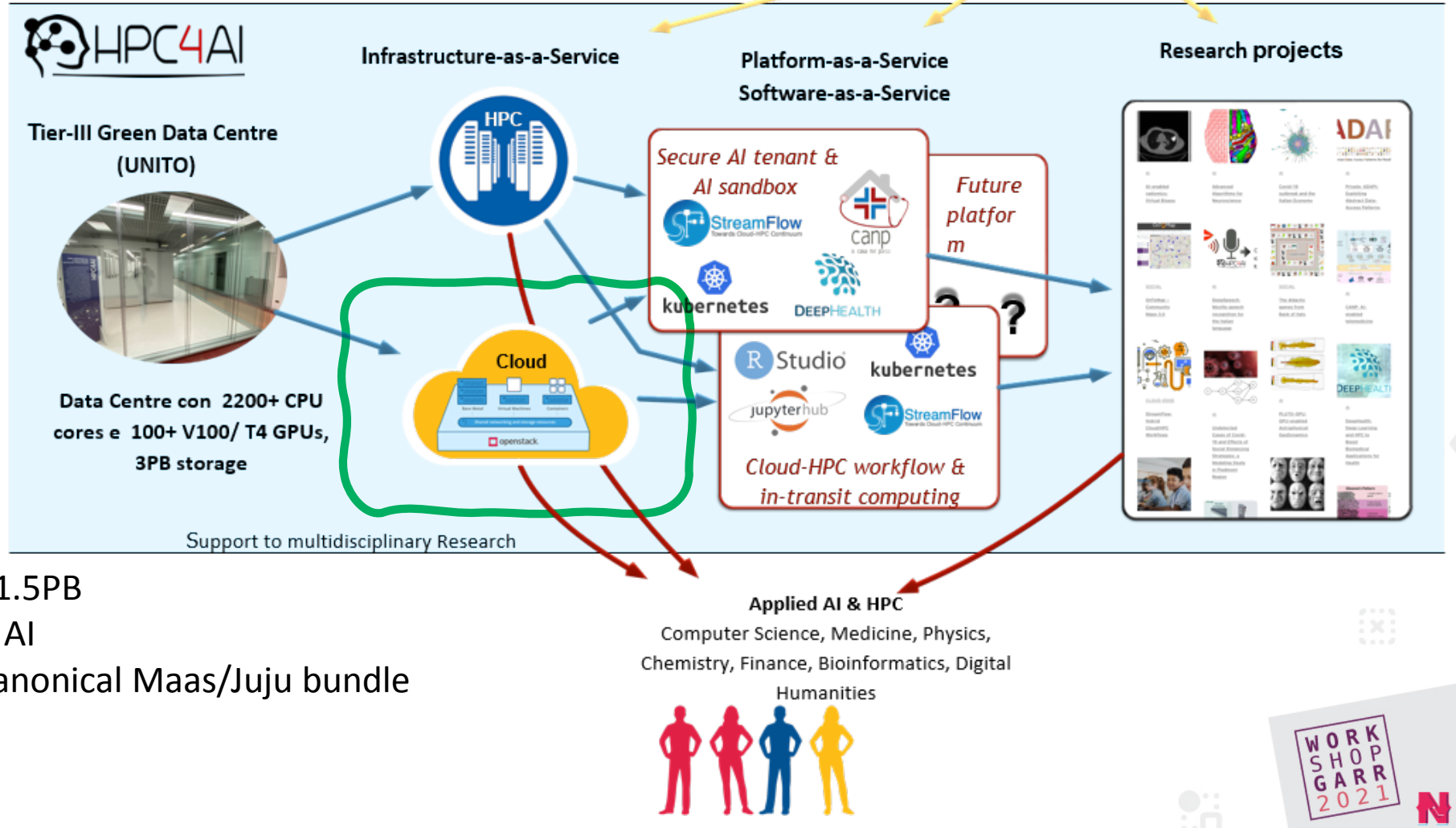
Mentimeter code: 9489 0567

## *Il contesto di HPC4AI*

**HPC4AI - High-Performance Computing for Artificial Intelligence – OpenLab**

Progetto Dec2017

Operativi da Ottobre 2019: continuous deploy strategy




Tech info for Cloud systems:

1500+cpu

190TB+ RAM

100+ GPUs

4 different storage systems ~1.5PB

60+ research projects mostly AI

Openstack Deploy through Canonical Maas/Juju bundle

## *Il contesto di HPC4AI*

# Menzione speciale CIOsummIT categoria Soluzioni infrastrutturali

**Progetto HPC4AI** – L'**Università di Torino** ha cofinanziato insieme alla **Regione Piemonte** HPC4AI, un centro di competenza sui temi di High-Performance Computing, AI e Big Data Analytics. A tal fine è stato necessario realizzare un nuovo Data Center che doveva essere progettato secondo i più innovativi sistemi sia per il raffreddamento sia per l'alimentazione in continuità e rispondere ai parametri più stringenti in fatto di consumo energetico e impatto ambientale. **Vertiv** ne ha quindi realizzato l'infrastruttura elettrica e per il Thermal Management.

***L'evoluzione delle GPU presenti sui nodi compute***

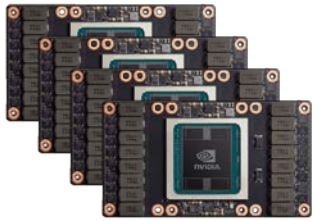
nVidia T4
pci-card
10de:1eb8

nVidia V100
SXM2
10de:1db5

nVidia A100
pci-card
10de:XXXX

2019 — 2020 — 2020 — 2021 — 2022

nVidia V100
pci-card
10de:1df6



nVidia A40
pci-card
10de:2235

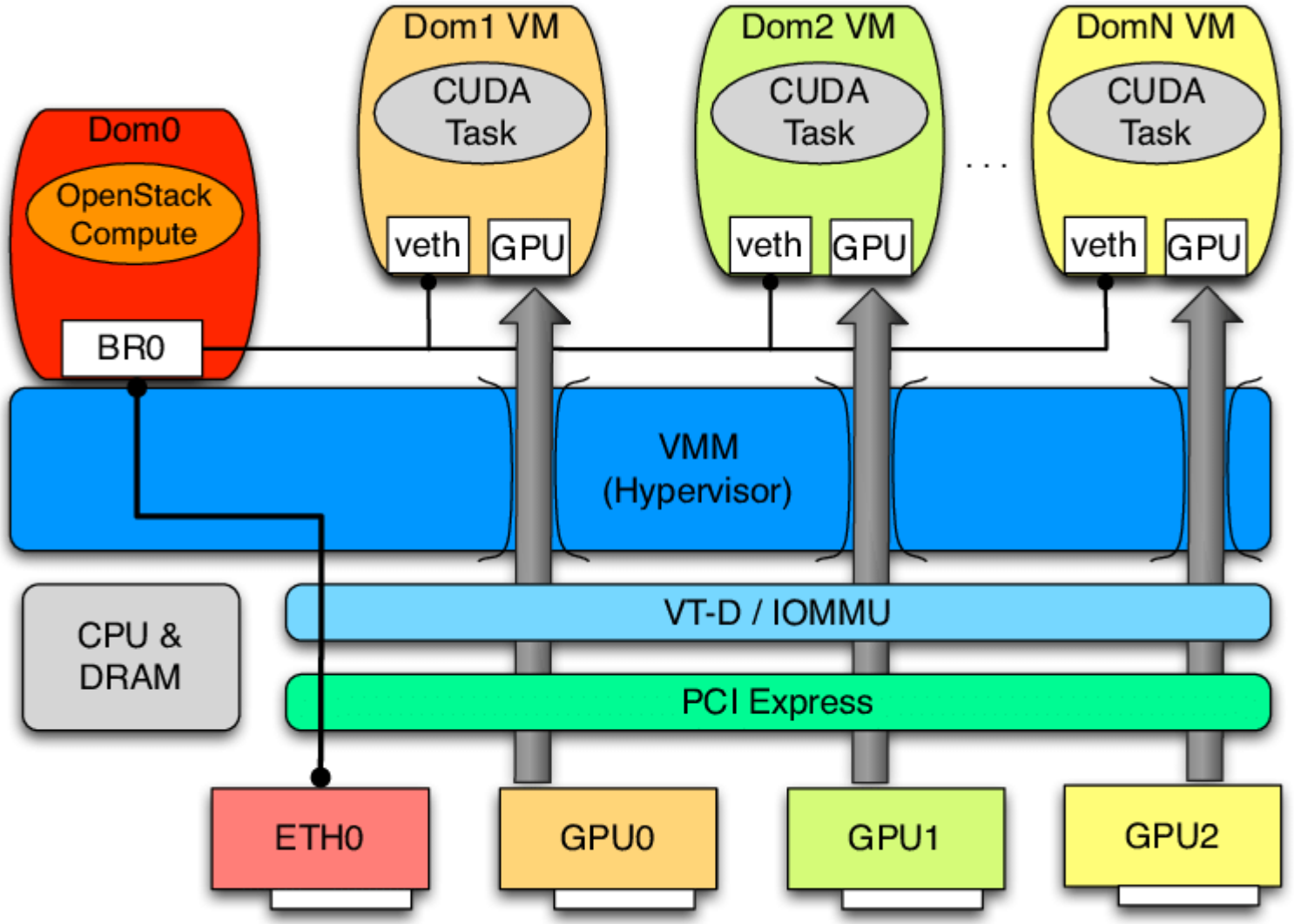
*A seconda delle acquisizioni di materiale hardware e l'evoluzione nel tempo, le gpu cambiano o, anche più semplicemente, si spostano tra un nodo e un altro …*

2008. A software tool for the simulation of a digital X-ray imaging system. pp.351-352. In INTERNATIONAL JOURNAL OF COMPUTER ASSISTED RADIOLOGY AND SURGERY - ISSN:1861-6410 vol. Volume 3, Supplement 1 O. Rampado; G. Scapola; N. Balossino; M. Lucenteforte; S. Rabellino; S.D. Bianchi; R. Ropolo

***La modalità pci-passthrough.***

***La scelta:***
***Con la modalità pci passthrough ogni GPU viene riservata per una VM in modo esclusivo.***

*Performance garantite all'utente finale che non sempre riesce a comprendere appieno l'infrastruttura tecnologica, ma ne percepisce solo gli effetti.*




https://www.researchgate.net/publication/263808350_Evaluating_GPU_Passthrough_in_Xen_for_High_Performance_Cloud_Computing/figures?lo=1

## *Configurare i nodi per il pci-passthrough durante il deploy /1*

Per impostare il sistema operativo abilitando il pci passthrough (tipicamente si devono modificare GRUB e impostare e abilitare uno script a startup in rc.local) abbiamo deciso di utilizzare cloudinit attraverso una modifica al model di Juju.

Il model si aggiorna costruendo un file yaml opportuno in cui si inseriscono gli script in formato encoded BASE64

cloudinit-userdata.yaml.template

```
cloudinit-userdata: |
  postruncmd:
    - "update-initramfs -u > /root/initramfs-update.log"
    - "update-grub > /root/grub-update.log"
  write_files:
    - path: /etc/initramfs-tools/modules
      content: pci_stub ids=10de:1ed8,10de:1df6,10de:1db5,10de:2235
    - path: /etc/modules
      content: |
        pci_stub
        vfio
        vfio_iommu_type1
        vfio_pci
    - path: /etc/rc.local
      encoding: b64
      permissions: '0755'
      content: $BASE64_ENCODED_RC_LOCAL
    - path: /etc/default/grub
      encoding: b64
      content: $BASE64_ENCODED_DEFAULT_GRUB
```

- `apply> juju model-config ./cloudinit-userdata.yaml`
- `check> juju model-config cloudinit-userdata |yq eval -P`

DEFAULT_GRUB

```
# /boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
#   info -f grub -n "Simple configuration"

GRUB_DEFAULT=0
GRUB_TIMEOUT_STYLE=hidden
GRUB_TIMEOUT=0
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="intel_iommu=on vfio_iommu_type1.allow_unsafe_interrupts=1 modprobe.blacklist=nvidiafb,nouveau"
GRUB_CMDLINE_LINUX=""
```

## *Configurare i nodi per il pci-passthrough durante il deploy /1*

cloudinit-userdata.yaml.template

```
cloudinit-userdata: |
  postruncmd:
    - "update-initramfs -u > /root/initramfs-update.log"
    - "update-grub > /root/grub-update.log"
```

Per impostare il sistema operativo abilitando il pci passthrough (tipicamente si devono modificare GRUB e impostare e abilitare uno script a startup in rc.local o cloudinit

Il model
inserisco

- apply> ju
- check> ju

**VFIO - “Virtual Function I/O”**

Many modern systems now provide DMA and interrupt remapping facilities to help ensure I/O devices behave within the boundaries they’ve been allotted. This includes x86 hardware with AMD-Vi and Intel VT-d, POWER systems with Partitionable Endpoints (PEs) and embedded PowerPC systems such as Freescale PAMU.
The VFIO driver is an IOMMU/device agnostic framework for exposing direct device access to userspace, in a secure, IOMMU protected environment. In other words, this allows safe, non-privileged, userspace drivers.

From: https://www.kernel.org/doc/html/latest/driver-api/vfio.html

```
# /boot/gru
# For full
#   info -f

GRUB_DEFAULT=0
GRUB_TIMEOUT_STYLE=hidden
GRUB_TIMEOUT=0
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="intel_iommu=on vfio_iommu_type1.allow_unsafe_interrupts=1 modprobe.blacklist=nvidiafb,nouveau"
GRUB_CMDLINE_LINUX=""
```

## *Configurare i nodi per il pci-passthrough durante il deploy /2*

RC_LOCAL

```
#!/bin/bash
# /etc/rc.local
#
# NVIDIA Vendor is 0x10de for T4/V100/V100SXM2/A40 cards
#

vfiobind() {
    dev="$1"
    if [ -d /sys/bus/pci/devices/$dev/ ]; then
        vendor=$(cat /sys/bus/pci/devices/$dev/vendor 2>/dev/null)
        if [ "AA$vendor" == "AA0x10de" ]; then
                device=$(cat /sys/bus/pci/devices/$dev/device)
                echo "NVIDIA card detected $device"
                if [ -e /sys/bus/pci/devices/$dev/driver ]; then
                        echo $dev > /sys/bus/pci/devices/$dev/driver/unbind
                fi
                echo $vendor $device > /sys/bus/pci/drivers/vfio-pci/new_id
        fi
    fi
}

#
# Switch off the kernel driver and enable passthrough
#

# 4 CARD PCI SYSTEMS Addresses
vfiobind 0000:3b:00.0
vfiobind 0000:5e:00.0
vfiobind 0000:af:00.0
```

```
vfiobind 0000:5e:00.0
vfiobind 0000:af:00.0
vfiobind 0000:d8:00.0

# 4 SXM2 addresses
vfiobind 0000:61:00.0
vfiobind 0000:62:00.0
vfiobind 0000:89:00.0
vfiobind 0000:8a:00.0

# 8 CARD PCI SYSTEMS Addresses
vfiobind 0000:01:00.0
vfiobind 0000:22:00.0
vfiobind 0000:41:00.0
vfiobind 0000:61:00.0
vfiobind 0000:81:00.0
vfiobind 0000:a1:00.0
vfiobind 0000:c1:00.0
vfiobind 0000:e1:00.0
```

- Get Addresses> lspci –n |grep 10de

## *Configurare i nodi per il pci-passthrough durante il deploy /2*

RC_LOCAL

```
#!/bin/bash
# /etc/rc.local
#
# NVIDIA Vendor is 0x10de for T4/V100/V100SXM2/A40 cards
#

vfiobind() {
    dev="$1"
    if [ -d
        ven
        if [
        fi
    fi
}

#
# Switch of
#

# 4 CARD PCI SYSTEMS Addresses
vfiobind 0000:3b:00.0
vfiobind 0000:5e:00.0
vfiobind 0000:af:00.0
```

```
vfiobind 0000:5e:00.0
vfiobind 0000:af:00.0
vfiobind 0000:d8:00.0

# 4 SXM2 addresses
```

```
#!/bin/bash

export BASE64_ENCODED_RC_LOCAL=`cat RC_LOCAL |base64`
export BASE64_ENCODED_DEFAULT_GRUB=`cat DEFAULT_GRUB |base64`

envsubst < cloudinit-userdata.yaml.template > cloudinit-userdata.yaml
```

Ps. Il file yaml generato non è perfetto, bisogna correggere l'indentazione, ma vista la frequenza con cui si modifica, è ragionevole farlo a mano con un text-editor (vi…)

## *E ora juju … /1*

Se tutto è andato a buon fine, possiamo configurare i charm **nova-compute** e **nova-cloud-controller**

```
#!/bin/bash

PCI_ALIAS='[{"vendor_id":"10de", "product_id":"1eb8", "device_type":"type-PF",
"name":"gpu"},{"vendor_id":"10de", "product_id":"1df6", "device_type":"type-PCI",
"name":"V100"},{"vendor_id":"10de", "product_id":"1db5", "device_type":"type-PCI", "name":"V100SXM2"},
{"vendor_id":"10de", "product_id":"2235", "device_type":"type-PCI", "name":"A40"}]']'

ERRS=`echo $PCI_ALIAS |jq . 2>&1`

if [ $? -eq 0 ]; then
        echo "Actual config parse ok."

        echo "Applying CONFIG:"
        juju config nova-compute pci-alias="$PCI_ALIAS"
        juju config nova-compute pci-passthrough-whitelist="$PCI_ALIAS"
        juju config nova-cloud-controller pci-alias="$PCI_ALIAS"
else
        echo "Config parse errors, see below:"
        echo $ERRS
fi
exit
```

NB: per funzionare, il charm nova-compute deve già avere la «patch rabser» (>= train) per consentire la configurazione di pci-alias multipli nel modo corretto…

## *E ora juju … /2*

Sysadmin paranoia: ma siamo proprio sicuri che funzioni ??

List_pci_devices.sh

```
#!/bin/bash

PERCONA_NODE=`juju status |grep percona-cluster |grep "*" | cut -f1 -d" " |rev|cut -b2-|rev`

QUERY="use nova;
SELECT JSON_OBJECT('host', compute_nodes.hypervisor_hostname,'pid', pci_devices.product_id,'vid',
pci_devices.vendor_id,'status', pci_devices.status) AS ''
            FROM pci_devices JOIN compute_nodes ON pci_devices.compute_node_id = compute_nodes.id
order by hypervisor_hostname;"

echo $QUERY | juju ssh ubuntu@$PERCONA_NODE sudo -H -u root mysql | jq -s .
```

Abilitando l'accesso a percona, possiamo eseguire una query SQL sul database di nova per sapere quali pci devices sono effettivamente disponibili…

output

```
[
  {
    "pid": "1eb8",
    "vid": "10de",
    "host": "gnode01.maas",
    "status": "allocated"
  },
  {
    "pid": "1eb8",
    "vid": "10de",
    "host": "gnode01.maas",
    "status": "allocated"
  }, ….
```

## *E ora juju … /3*

Ok, abbiamo i nodi installati, configurati, nova riconosce le schede e le pubblica al controller, ora le usiamo!

Flavor1_one_T4_card_vm

```
#!/bin/bash

openstack flavor create --ram 8192 --disk 100 --vcpu 8 --private hpc4ai.1xT4
openstack flavor set hpc4ai.1xT4 --property "pci_passthrough:alias"="gpu:1"
openstack flavor show hpc4ai.1xT4
```

Flavor2_two_V100_cards_vm

```
#!/bin/bash

openstack flavor create --ram 8192 --disk 100 --vcpu 8 --private hpc4ai.2xV100
openstack flavor set hpc4ai.2xV100 --property "pci_passthrough:alias"="V100:1,V100:1"
openstack flavor show hpc4ai.2xV100
```

Ogni PCI Device è agganciata ad un solo NUMA node !!

Flavor3_three_V100SXM2_cards_vm

```
#!/bin/bash

openstack flavor create --ram 8192 --disk 100 --vcpu 8 --private hpc4ai.3xV100
openstack flavor set hpc4ai.3xV100 --property "pci_passthrough:alias"="V100SXM2:2,V100SXM2:1"
openstack flavor show hpc4ai.3xV100
```

***Performance – risultati preliminari di benchmark con payload DeepLearning e differenti tipologie di storage***

Una virtual machine con 4 schede nVidia **V100/SXM2** in pci-passthrough e 4 volumi montati, uno per ciascun tipo di storage, utilizzando configurazioni variabili di memoria RAM per la virtual machine e tutti i core disponibili dell'host (80/HT).

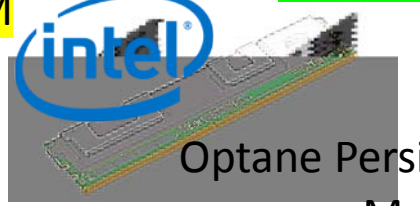
Unity-Cinder and Unity-Manila **charms sono self-made by UniTO**

**~2.2GB/s read**

Unity Appliance mixed disks

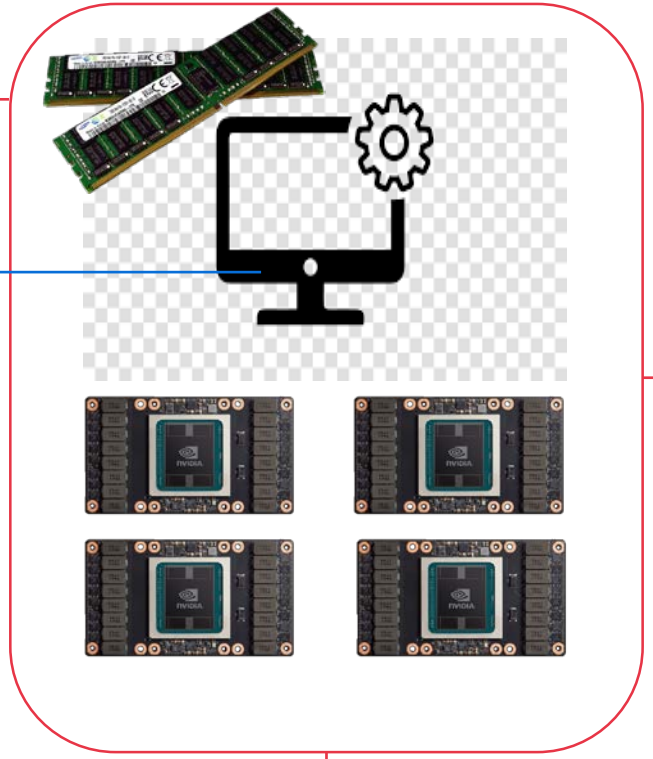
512GB DPCMEM **Donation of Intel Corporation**

**~12GB/s read**



Optane Persistent Memory

ephemeral volume



FULL NVMe bucket
Replication factor 3

**~3GB/s read**

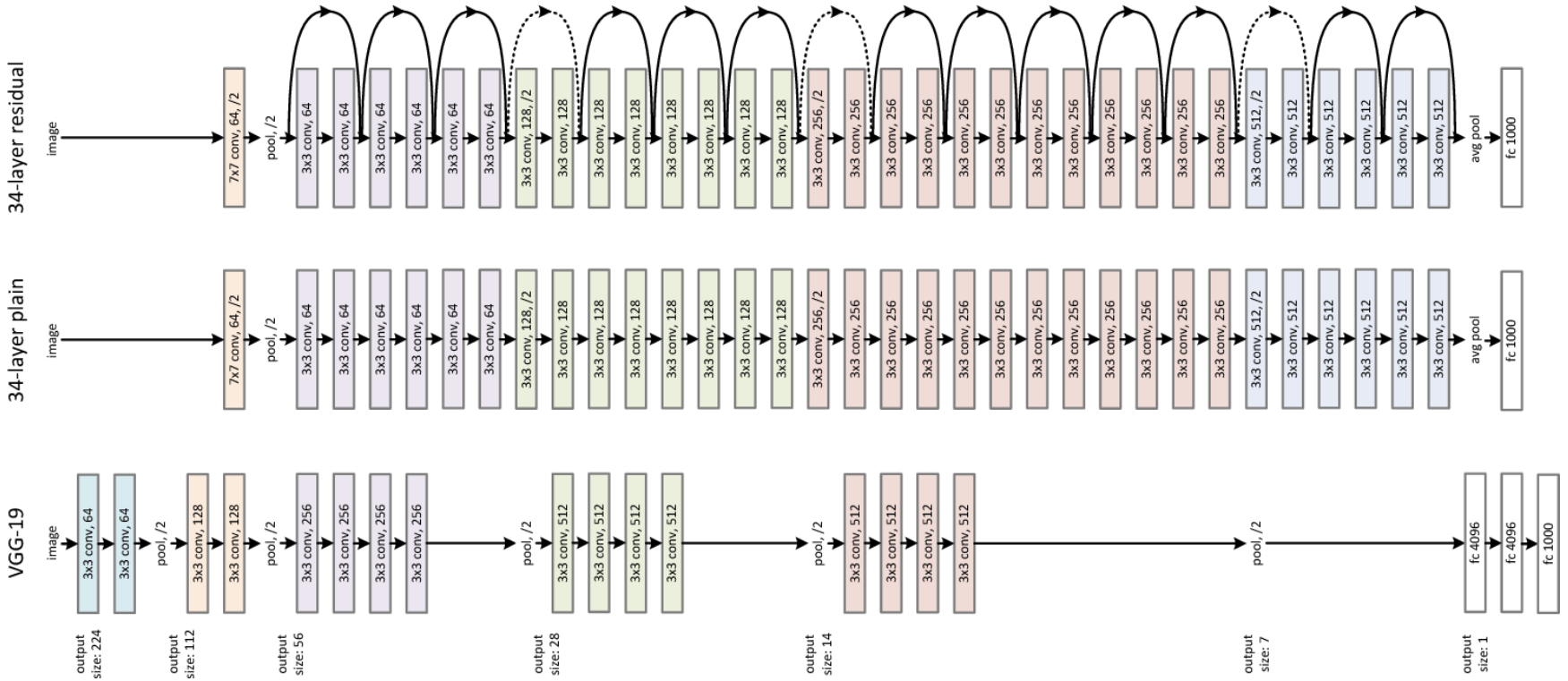
FULL Rotational
Erasure-coded

Lavoro di «fino» per ottenere prestazioni su erasure comparabili con replica3, grazie a (not well/un)-documented feature di ceph

Networking: Active Load Balancing - 2 links/25GBPs

## *Il benchmark: risultati preliminari /1*

Il modello è una resnet18 [1] addestrata con vanilla SGD con esecuzione in docker container [2]



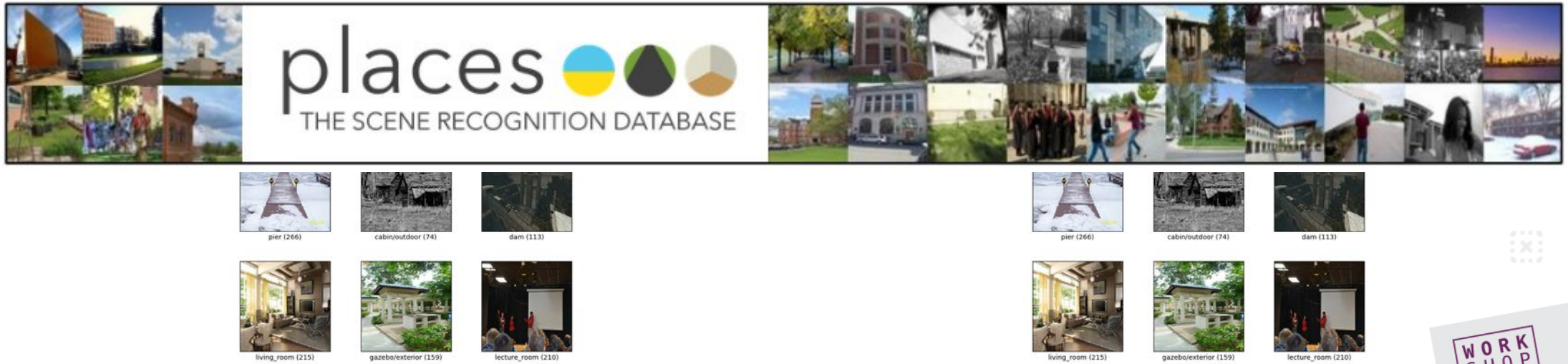

[1] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
[2] Carlo Alberto Barbano – Eidoslab Research Group - http://www.di.unito.it/do/gruppi.pl/Show?_id=qkzz

## *Il benchmark: risultati preliminari /1*

4 diversi dataset di dimensione crescente con adattamento del batchsize alla RAM della vm (480GB/180GB/32GB)


ImageNet: ~146GB dataset size

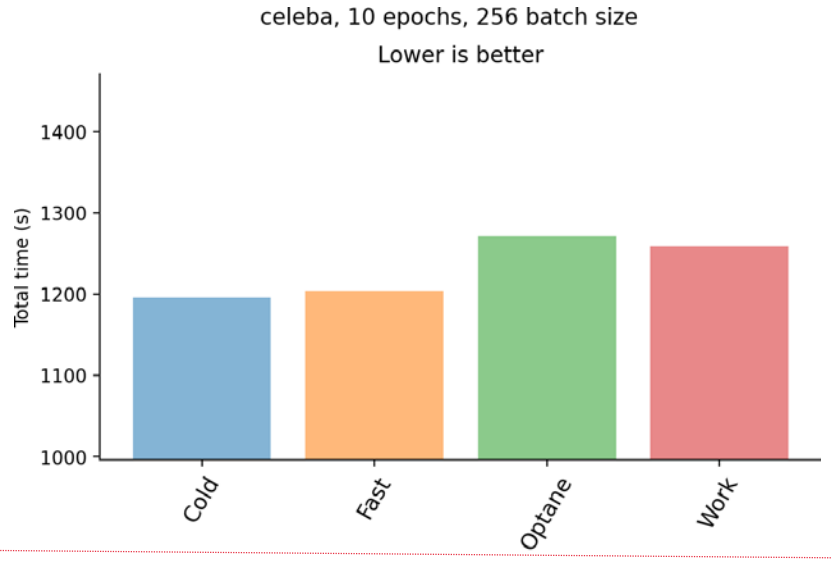

CelebA: ~2GB dataset size

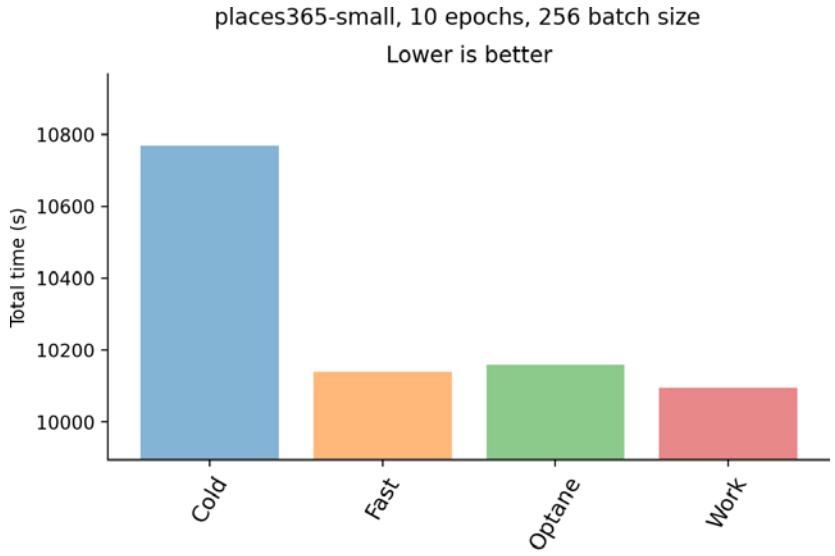


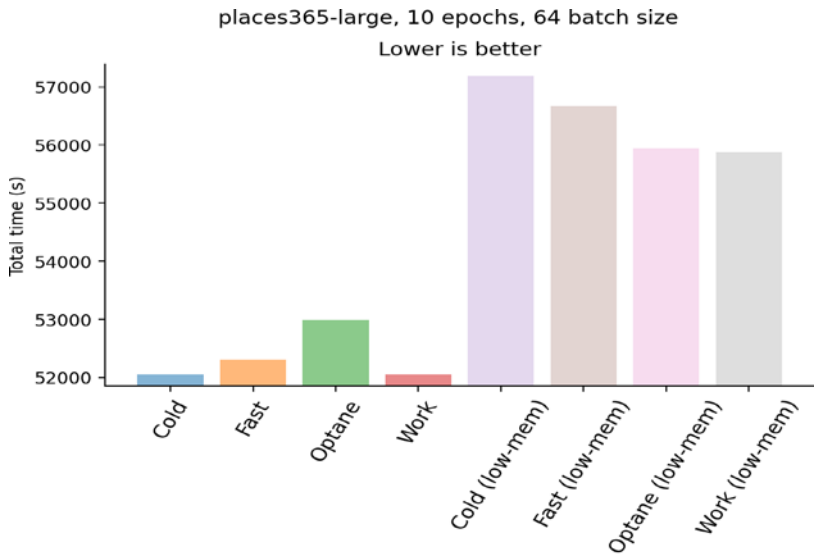

Places365 small: ~52GB dataset size

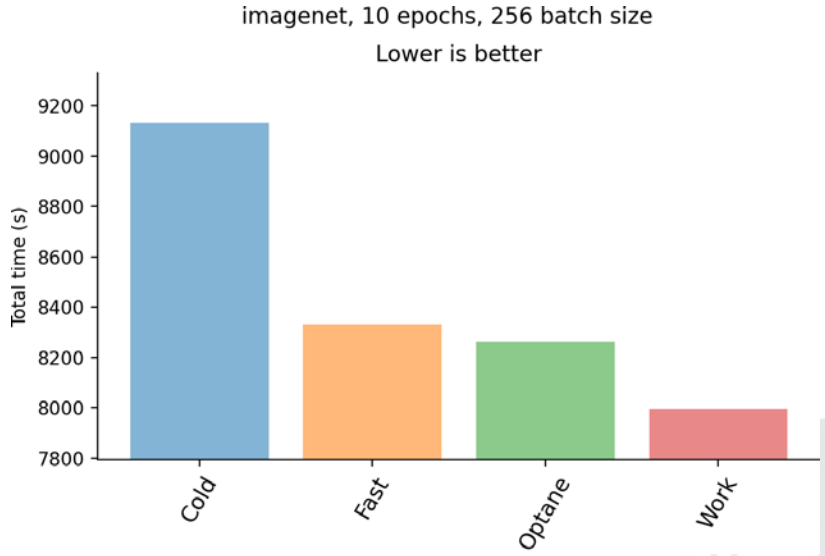
Places365 small: ~110GB dataset size

## *Il benchmark: risultati preliminari /2*




To be continued …

***The missing config in nova-cloud-controller – do it once…***

```
scheduler-default-filters:
RetryFilter,AvailabilityZoneFilter,CoreFilter,RamFilter,ComputeFilter,ComputeCapabilitiesFilter,ImagePropertie
sFilter,ServerGroupAntiAffinityFilter,ServerGroupAffinityFilter,DifferentHostFilter,SameHostFilter,AggregateIn
stanceExtraSpecsFilter,PciPassthroughFilter
```

```
> juju config scheduler-default-filters="RetryFilter, AvailabilityZoneFilter, CoreFilter,RamFilter,
ComputeFilter, ComputeCapabilitiesFilter, ImagePropertiesFilter, ServerGroupAntiAffinityFilter,
ServerGroupAffinityFilter, DifferentHostFilter, SameHostFilter, AggregateInstanceExtraSpecsFilter,
PciPassthroughFilter"
```