WORKSHOP GARR 2022
NET MAKERS

Condivisione di una singola GPU verso multiple macchine virtuali attraverso la virtualizzazione delle GPU

C. Pisa[1], D. Passalacqua[1], A. Colla[1], A. Barchiesi[1], F. Galeazzi[1], M. Di Fazio[1], M. Lorini[1], F. Lombardo[1], E. Petagna[1], S. Rabellino[2], I. Colonnelli[2], M. Aldinucci[2], S. Donetti[2]

[1] Consortium GARR, Italy
[2] University of Turin, Italy

GARR Cloud Services

- Federated GARR Cloud Platform (IaaS)
- GARR Container Platform (CaaS)
- Deployment as a Service (DaaS)
- GARR Workplace (SaaS)

GARR cloud

7600 cores

13.5 PB 15% SSD

333 TFlop

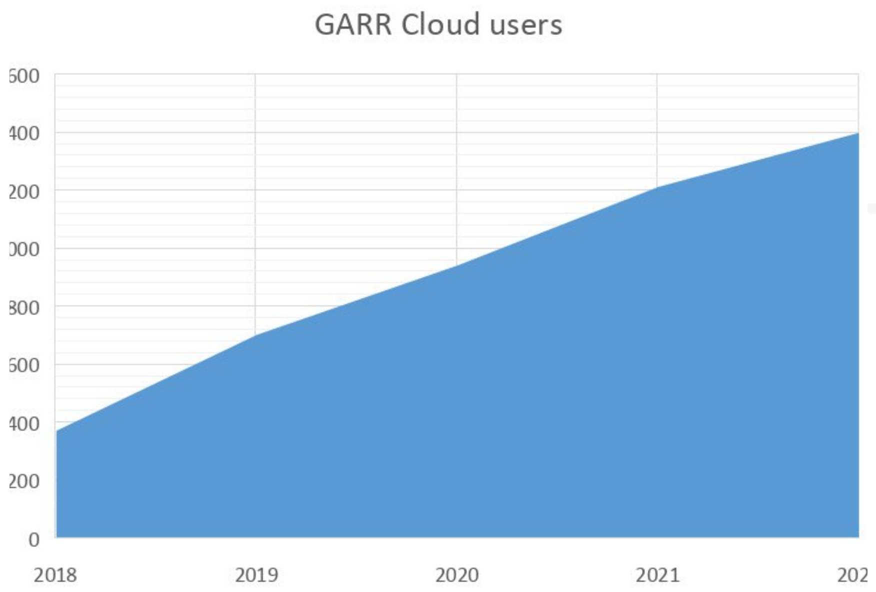
... 11 rack/CSD-modules

Team CSD in real life

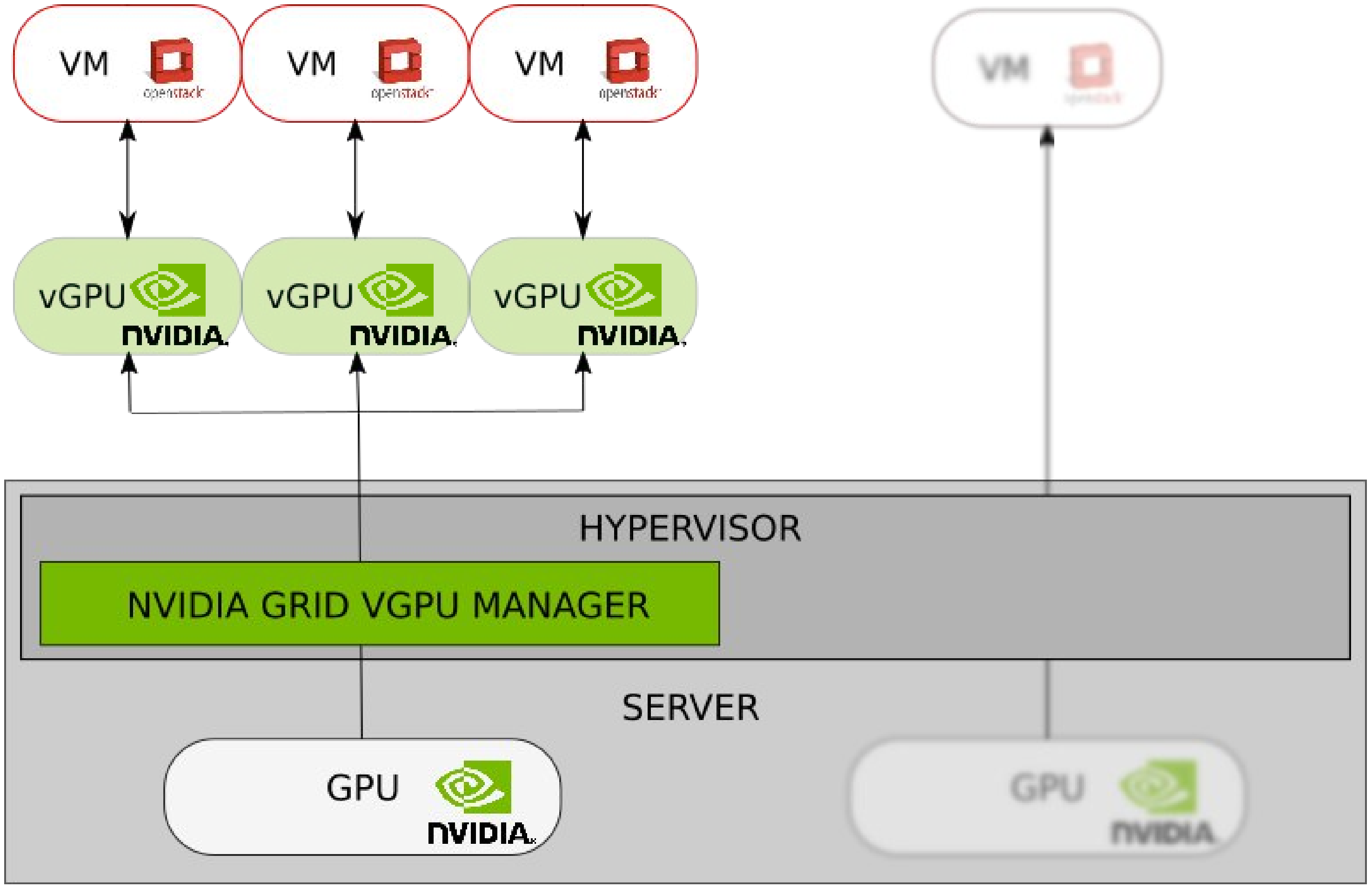
Users vs GPU workloads

- Increasing number of users
- Increasing popularity of GPU workloads
- GPU hardware scarcity

- GPU virtualization!

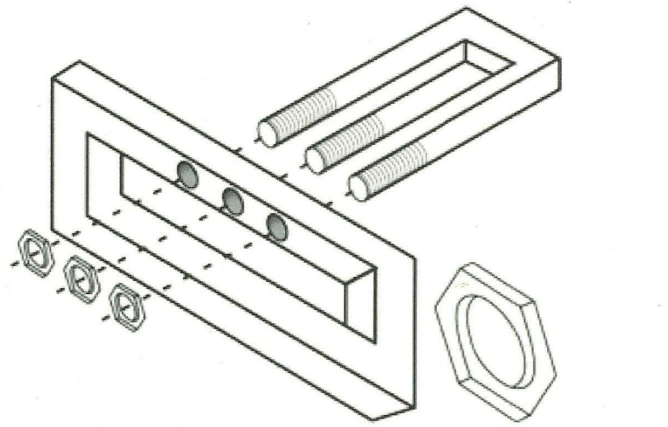


Making GPUs available to users



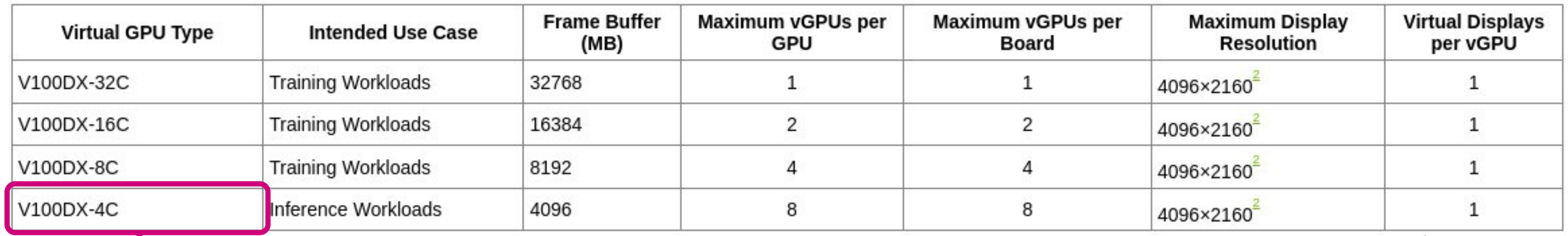
GPU virtualization can be challenging..

- Slicing GPUs can be non-trivial
- e.g. For NVIDIA V100 GPUs, several steps need to be taken and several technical issues need to be tackled



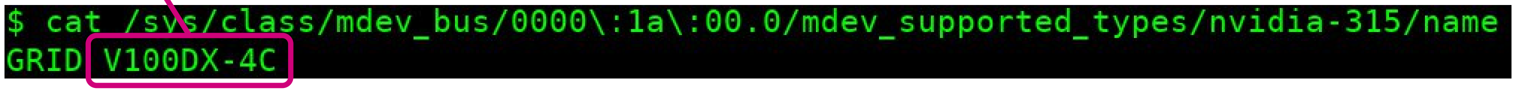
GPU virtualization in OpenStack

- Get Nvidia GRID license and install license manager
- Download and install drivers on the hypervisor servers
- Configure OpenStack nova
  - Choose one of the supported vGPU types, depending on your use case



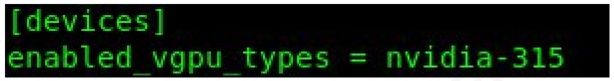
Virtual GPU Type	Intended Use Case	Frame Buffer (MB)	Maximum vGPUs per GPU	Maximum vGPUs per Board	Maximum Display Resolution	Virtual Displays per vGPU
V100DX-32C	Training Workloads	32768	1	1	4096×2160[2]	1
V100DX-16C	Training Workloads	16384	2	2	4096×2160[2]	1
V100DX-8C	Training Workloads	8192	4	4	4096×2160[2]	1
V100DX-4C	Inference Workloads	4096	8	8	4096×2160[2]	1

https://docs.nvidia.com/grid/13.0/grid-vgpu-user-guide/index.html#installing-configuring-grid-vgpu



```
$ cat /sys/class/mdev_bus/0000\:1a\:00.0/mdev_supported_types/nvidia-315/name
GRID V100DX-4C
```

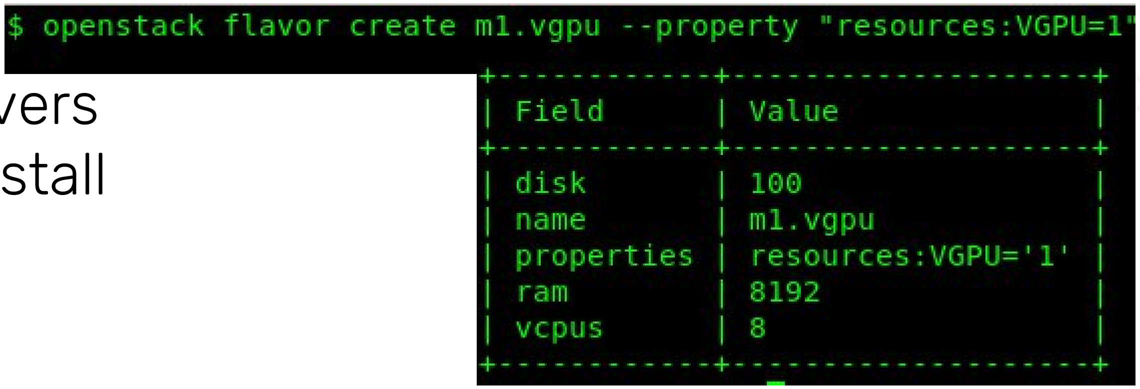
- Add configuration to `nova-compute.conf`



```
[devices]
enabled_vgpu_types = nvidia-315
```

GPU Virtualization in OpenStack

- Create a new flavor
- Download the vGPU drivers
- Create a new VM and install vGPU drivers



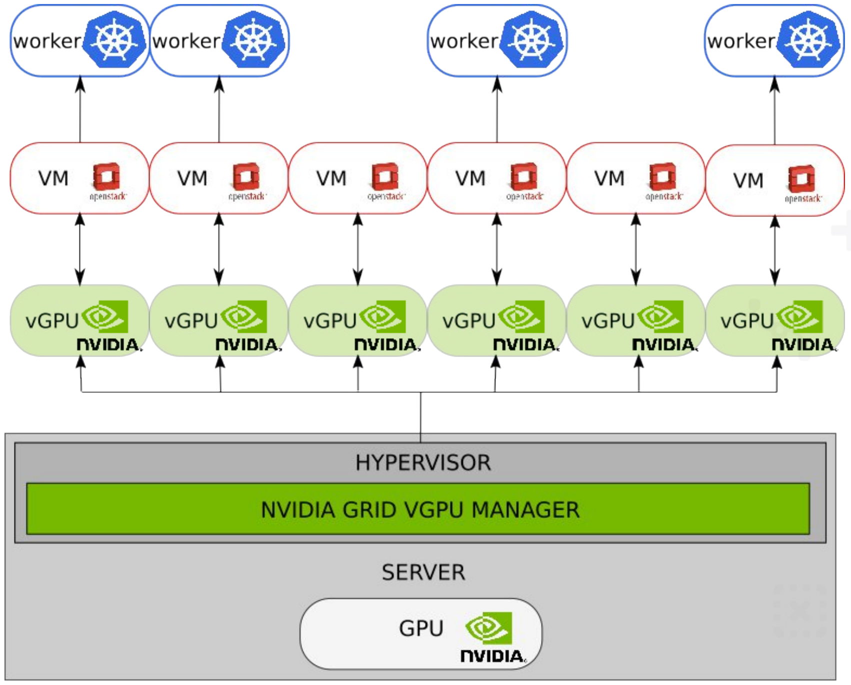
```
$ openstack flavor create m1.vgpu --property "resources:VGPU=1"
+------------+---------------------+
| Field      | Value               |
+------------+---------------------+
| disk       | 100                 |
| name       | m1.vgpu             |
| properties | resources:VGPU='1'  |
| ram        | 8192                |
| vcpus      | 8                   |
+------------+---------------------+
```

Nice but… there is an increasing demand from our users for GPU powered **Kubernetes workloads**

Enable GPU powered Kubernetes workloads

- Create GPU powered Kubernetes workers using the vGPU OpenStack flavor created previously*
- Enable Nvidia device plugin on Kubernetes Workers*
- Enable Kubernetes API `PodTolerationRestriction` admission controller and “flag” GPU enabled Kubernetes worker nodes*

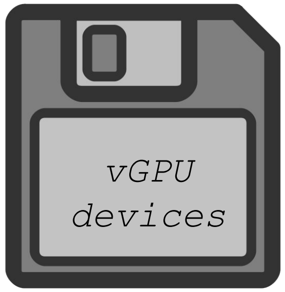
- **Now users can create vGPU powered Kubernetes workloads!**



* these operations can be performed automatically through Juju

But...

- *Issue: Linux devices associated with virtual GPUs are not persistent across reboots*
  - *Solution:*
    - *save the list of devices during operation*
    - *restore them at boot time*
  - *Our scripts available at https://github.com/ConsortiumGARR/gpu_mdev_scripts*



Wrap-Up

We have increased the availability of GPU resources for our users by leveraging virtualization

- *GPUs are scarce compared to the demand from users*
- *Several use cases don't need all the resources of a physical GPU*
- *=> More Happy Users!*

Future work:

- *Benchmark vGPU vs GPU Passthrough*
- *Improve automatic deployment of new GPU servers*
- *Improve GPU reservation and accounting*
- *Integrate A30/A100 GPU hardware*
- *Mix different vGPU types (/2, /4, /8)*

End

Thank you for your attention!