Building GPU-enabled Kubernetes single node cluster for MLOps experiments with new+second hand components

1. **SUPERMICRO MBD-X11SAE-M-O Micro ATX Server Motherboard LGA 1151 Intel C236** from newegg.com - $200 (Link to Newegg)

2. **Intel Xeon E3-1275 V6 Kaby Lake 3.8 GHz (4.2 GHz Turbo) LGA 1151 Server Processor Intel HD Graphics P630** - $360 (Link to Newegg)

3. 64GB of RAM: four DIMMs of **16GB DDR4-2400 UDIMM 1.2V CL17 (SKU: CT16G4DFD824A)** from Crucial: $250 (Link to Crucial)

4. **Sabrent Rocket Q 1TB NVMe PCIe M.2 2280 Internal SSD High Performance Solid State Drive R/W 3200/2000MB/s (SB-RKTQ-1TB)**: $110 (Link to Amazon)

5. **Supermicro SNK-P0046A4 Heatsink 2U+ Active Heatsink LGA1156 & LGA1155**: $25 (Link to Supermicro store)

6. **Nvidia Tesla M40 12GB** (Second hand): $125(Link to Ebay)

7. Dual 8 to 8 Graphics Power Cable (SKU: 030-0571-000): $6 (Link to Ebay)

Really old components:

8. HDD and SSD: Western Digital HDD 250GB SATA3 6.0Gb/s 7.2K (Used): $0, SSD 120 GB SATA3 6.0 Gb/s 2.5" (Used): $0, SSD 120 GB SATA3 6.0 Gb/s 2.5" (Used): $0

9. SuperMicro Chassis 733I-500B with 500W PSU (Used): $0 (Link to SuperMicro)

Final configuration: 1 x Xeon E3-1275, 64GB RAM, 1.5TB storage with 1TB on NVMe, NVIDIA GPU (Maxwell architecture) with 12GB GDDR5 for **total: $1076.0 (Ouch!)**

Additional Notes:

The 500W power supply coming with the mid tower chassis seems adequate for the graphics card provided you do not plan to reach frequently the max 250W which the card may need in peak memory and GPU utilization. The CPU drains no more than 73W, and the motherboard with the chipset and 64GB of RAM. Note that the chassis has an extension space allowing mounting larger power supply in case the one with 500W is not sufficient.

Use Ubuntu 20.04 and upgrade to 20.10.

Change the runlevel to 3 which will disable the XWindows subsystem as the NVIDIA GPU will not be used by default and NVIDIA kernel module will not be loaded. With runlevel set to 3 install the latest NVIDIA drivers for Ubuntu as:

```
dimitar@xeon-ubuntu:~/Downloads$ sudo ./NVIDIA-Linux-x86_64-460.32.03.run
```

Connect all disks (NVMe, SSD1, SSD2, and HDD) in a single file system using the supplied with the Ubuntu 20.10 logical volume manager **LVM2**.

Do not configure swap partition in case you are going to run Kubernetes locally using microk8s or minikube.

Do install the Nvidia container toolkit on Ubuntu 20.10. Note: while Ubuntu 20.10 is not yet official supported the Nvidia Container Toolkit can be installed without issue on 20.10 using the following instructions:

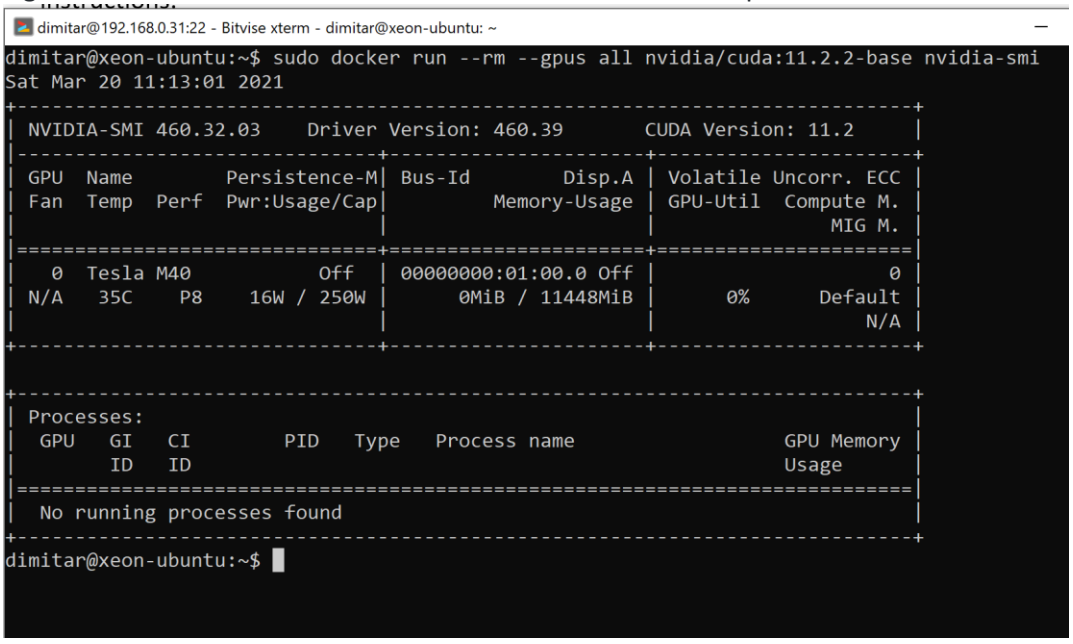
```
curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | \
  sudo apt-key add -
distribution=$(. /etc/os-release;echo ${ID}20.04)
curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list | \
  sudo tee /etc/apt/sources.list.d/nvidia-docker.list
sudo apt-get update
sudo apt-get install -y nvidia-docker2
```

Finally, to verify the successful installation run the following commands:

```
sudo systemctl restart docker
sudo docker run --rm --gpus all nvidia/cuda:11.2.2-base nvidia-smi
```

You should be seeing an output like the one on the Figure below:

Figure : execution of **`nvidia-smi`** from a docker container with preinstalled cuda-11.2 base image



Possible error which can be received while executing `nvidia-smi` in a container or directly on the OS is:

```
failed to initialize NVML: Driver/Library version mismatch
```

This error is discussed on StackOverflow here. The solution is to identify the duplicate versions of the installed nvidia drivers via `dpkg -l | grep -i nvidia` and update the nvidia driver which was installed on the OS with:

```
sudo apt install nvidia-driver-440
```

Figure: Shown are the top power consuming processes obtained by using the tool **PowerTOP**

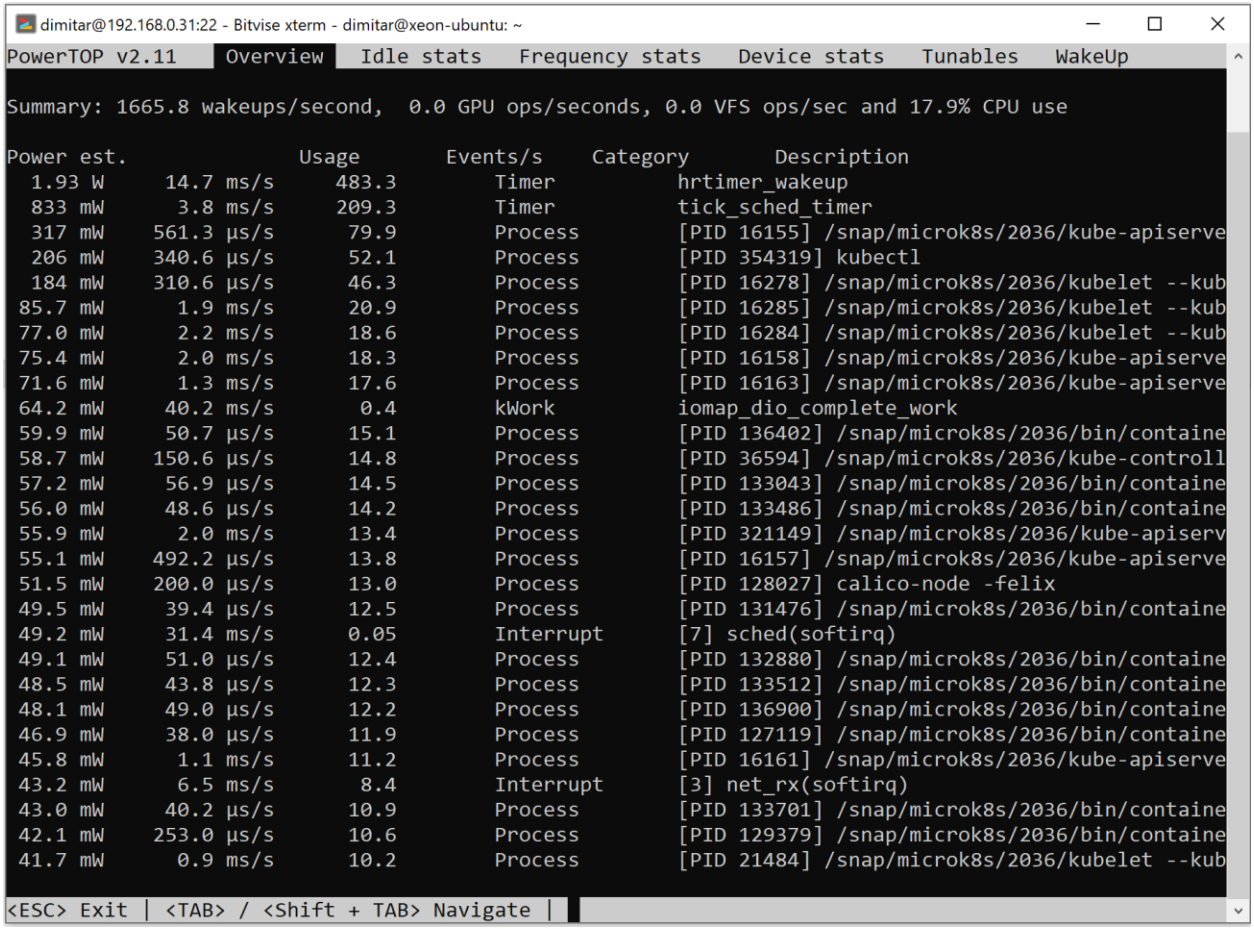


Figure: the total system memory and other memory related stats obtained by **`vmstat -s`**:

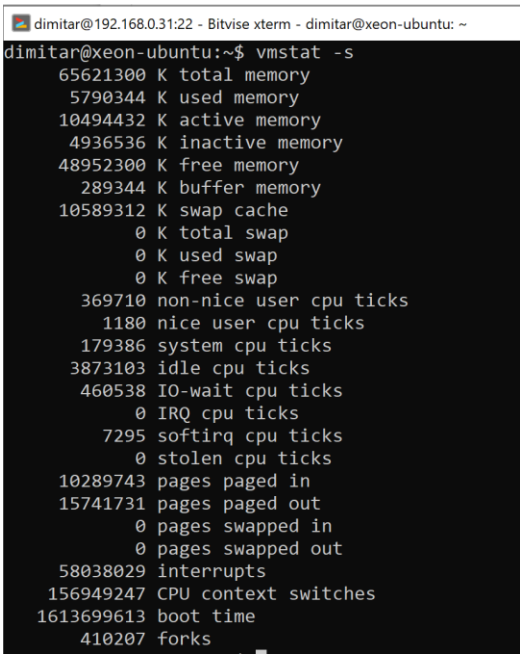


Figure: Querying the GPU and obtaining GPU configuration info by using Cuda 11.2 utility **`deviceQuery`**

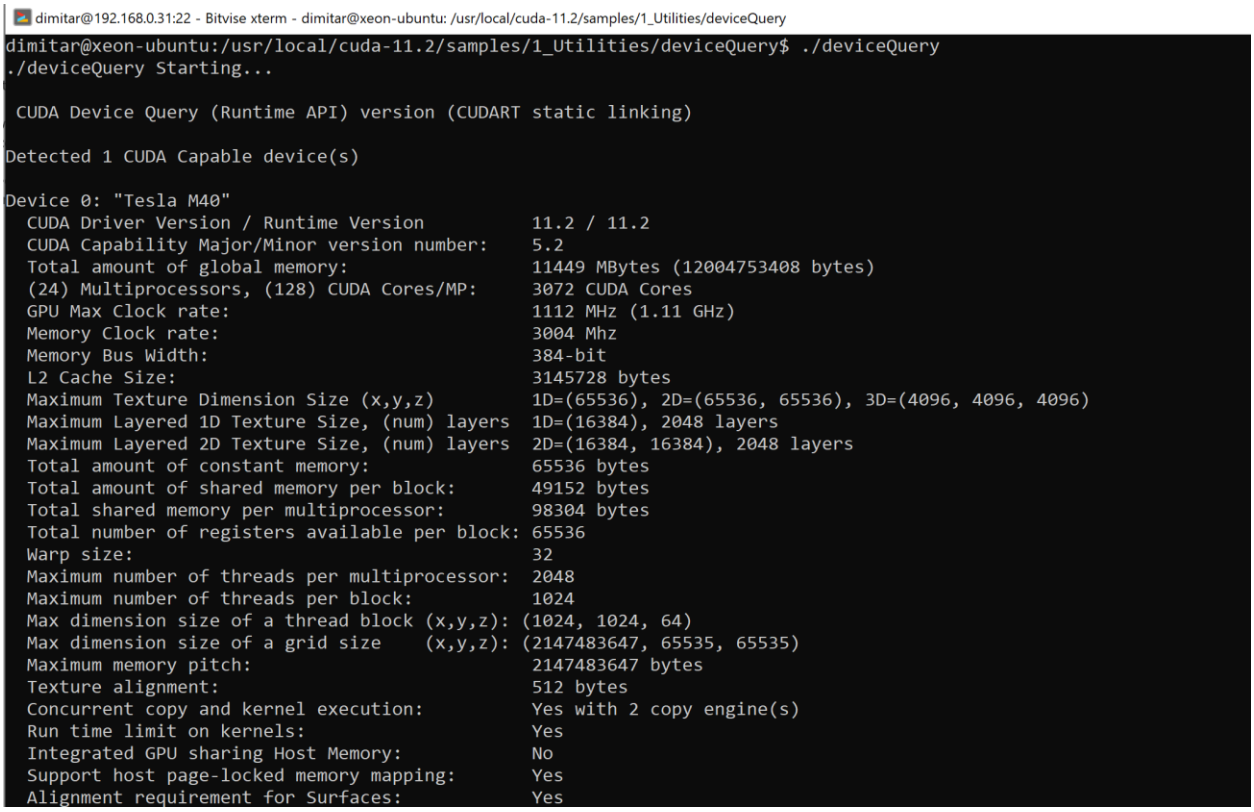


Figure: Running bandwidth test with the CUDA utility **`bandwithTest`**

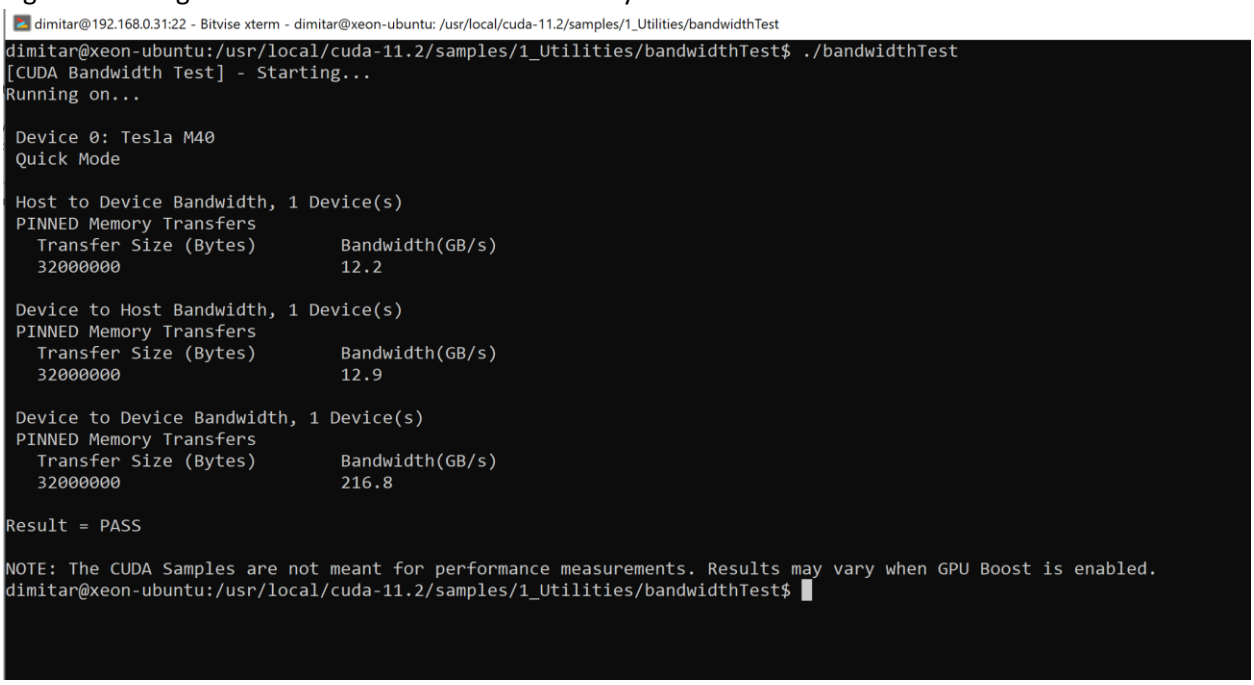


Figure: NVMe 1 TB mounted as **`/dev/nvme0n1`**

Figure: rotating disk drive WDC 250GB mounted as **`/dev/sda`**

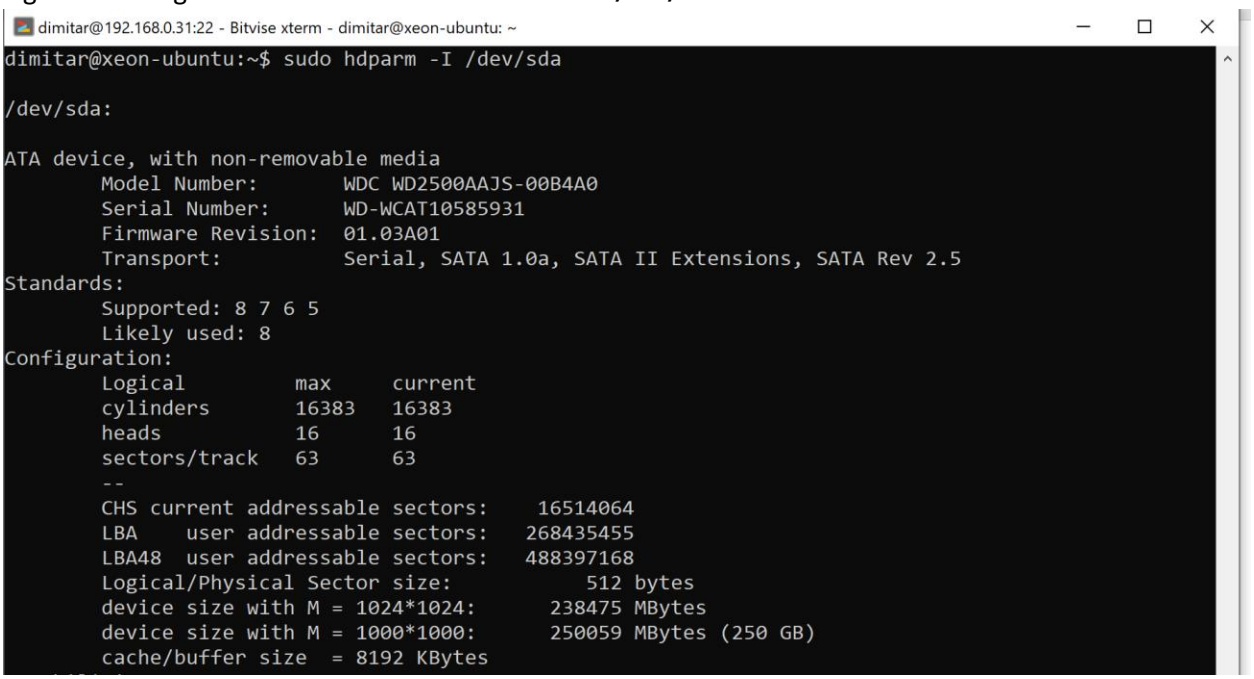


Figure: Solid State Disk MKN 120GB mounted as **`/dev/sdb`**

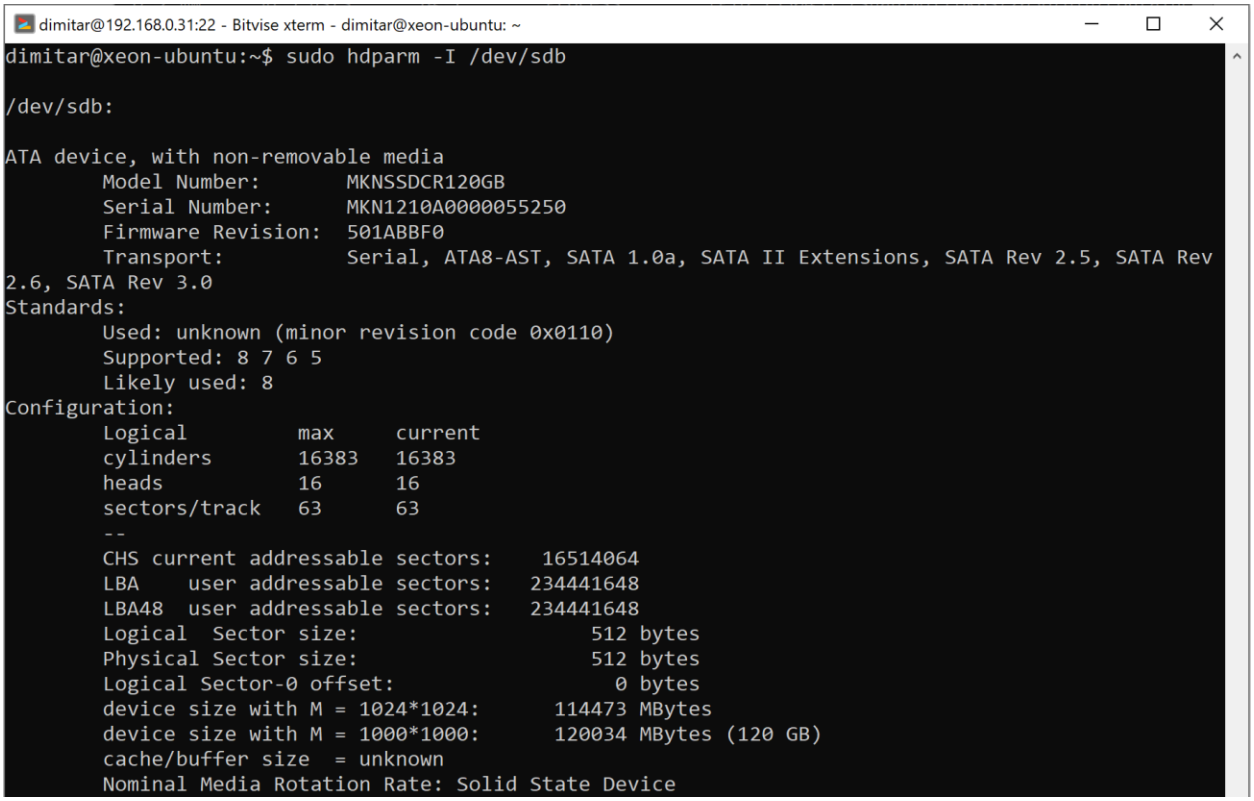


Figure : Solid State Disk OSZ-VERTEX3 120GB mounted as **`/dev/sdc`**

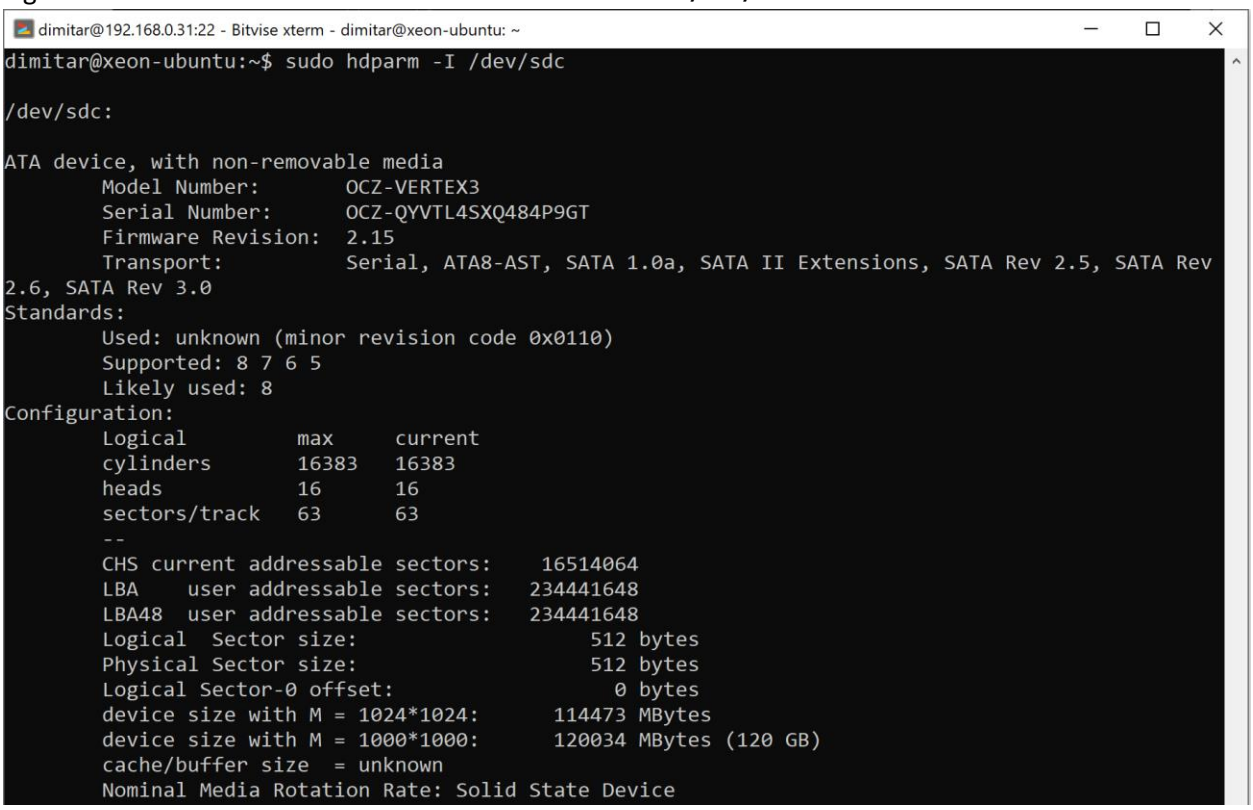


Figure: Mounted partitions obtained via `lsblk`

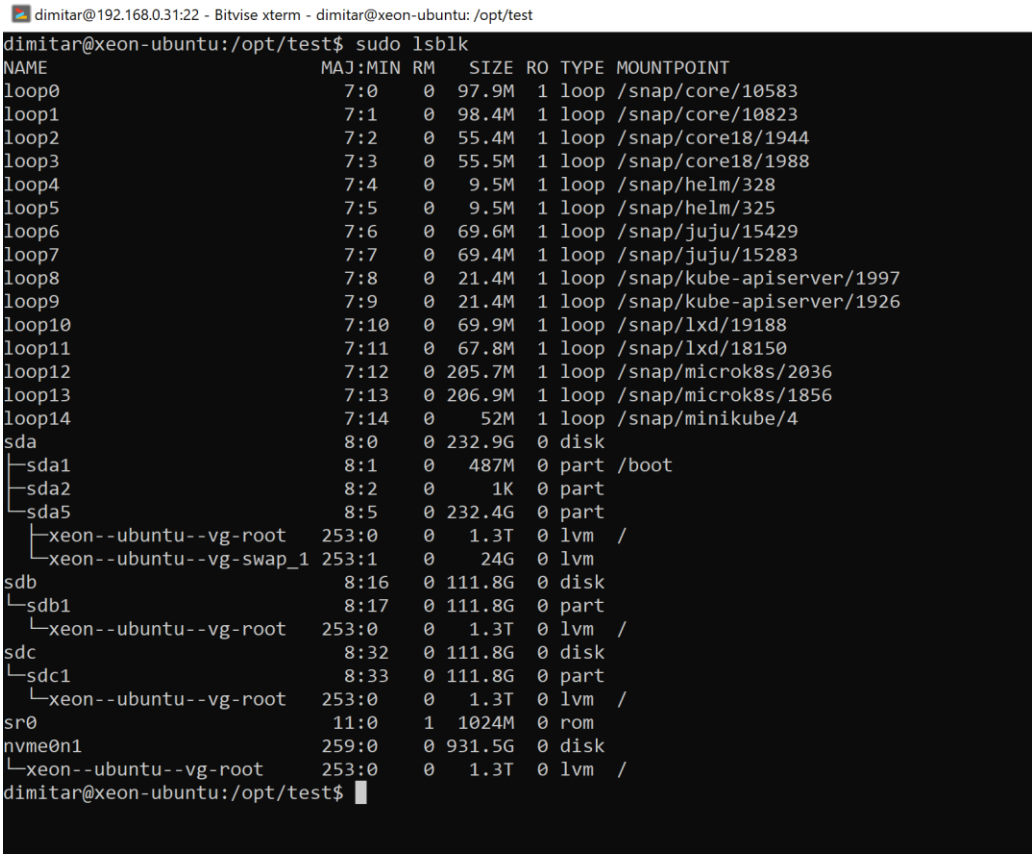


Figure: Physical volumes under LVM2 obtained by **pvdisplay**

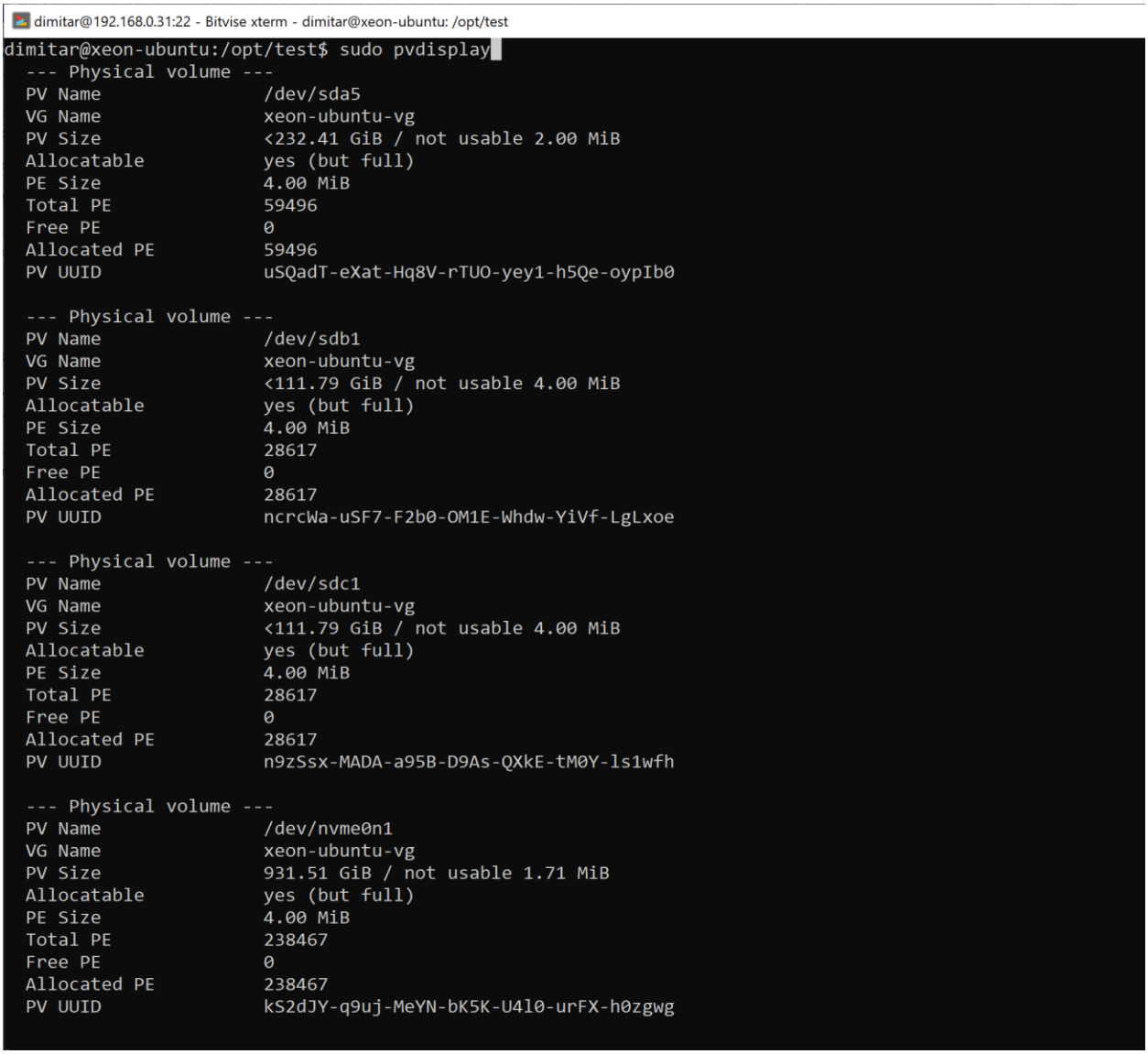
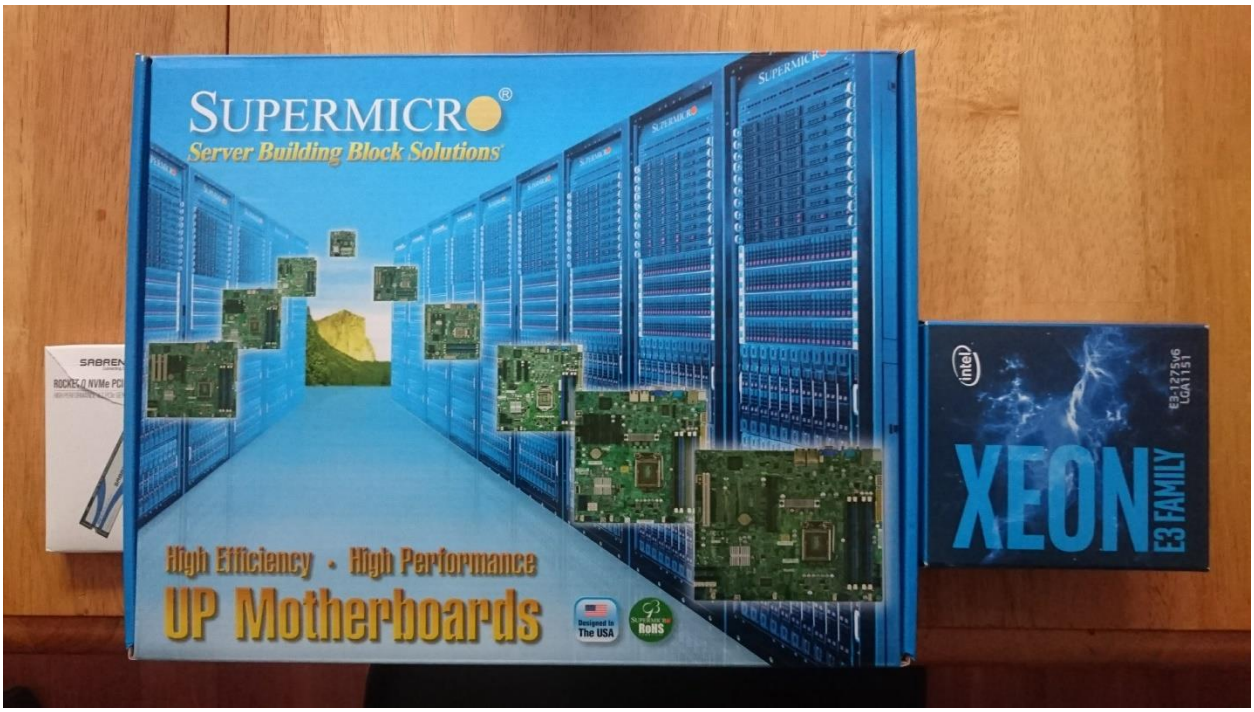
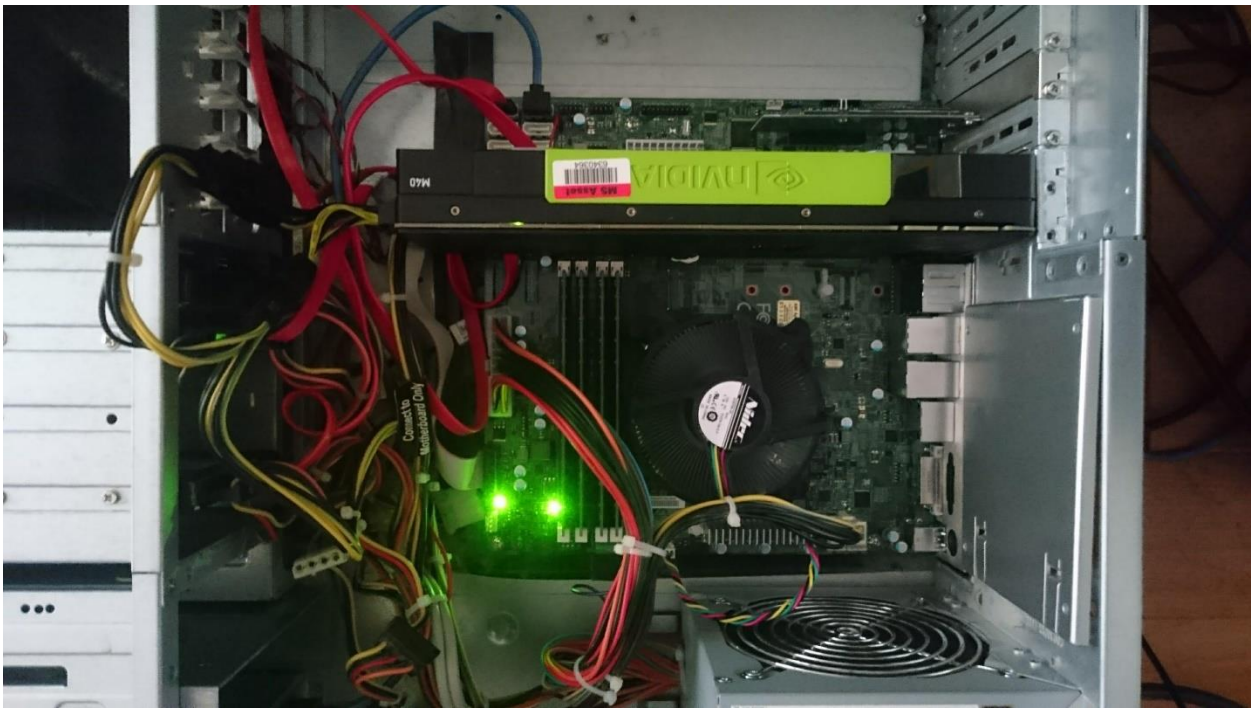


Figure: Snapshots from the assembly of the system including the new mother board, The NVIDIA Tesla card, memory DIMMs, the Xeon CPU and the PCIe NVMe module




NVIDIA
M40
Connect to
Motherboard Only