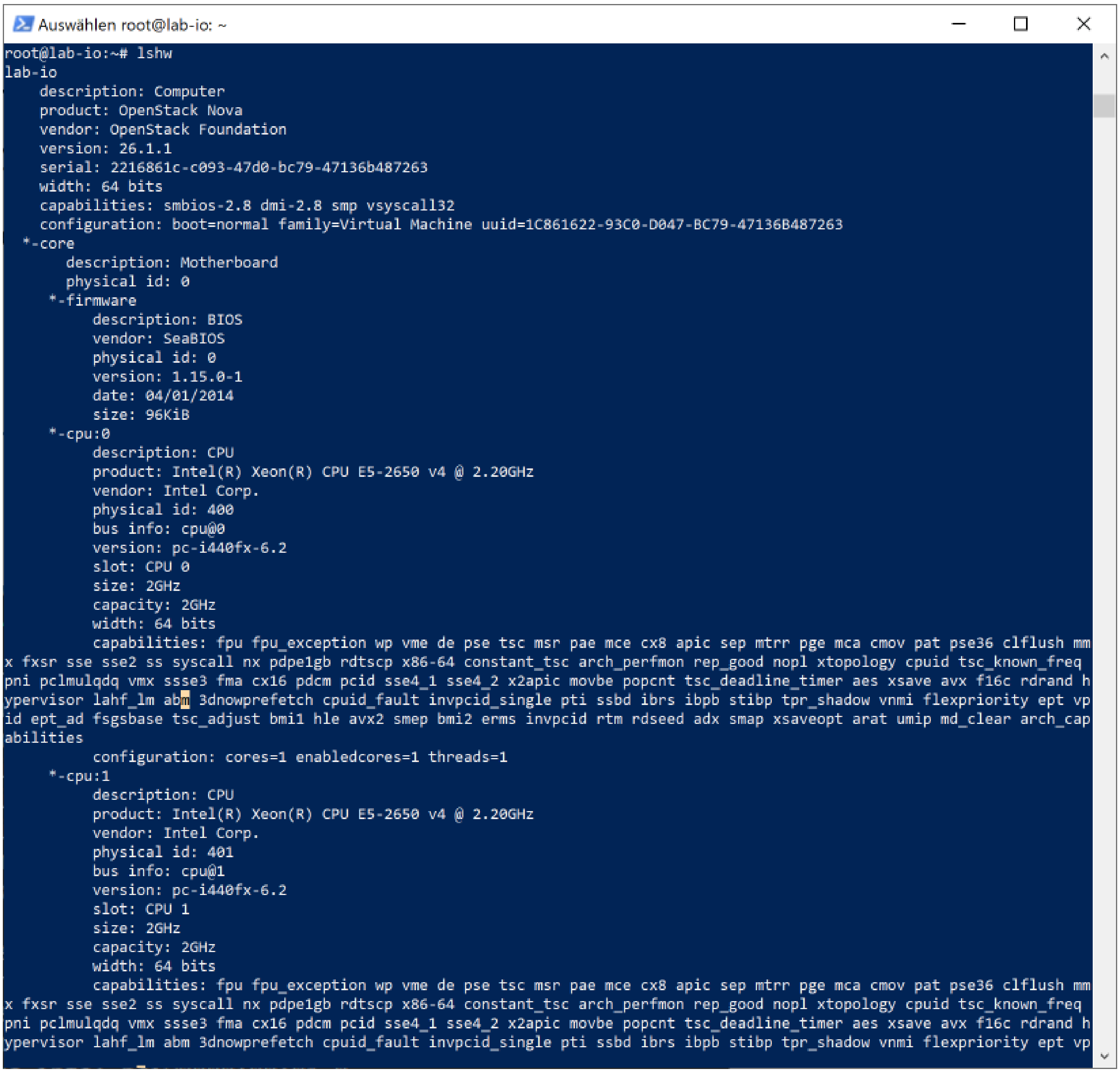
Lab 6 BSY IO 1

Task 1 – IO Hardware and Architecture

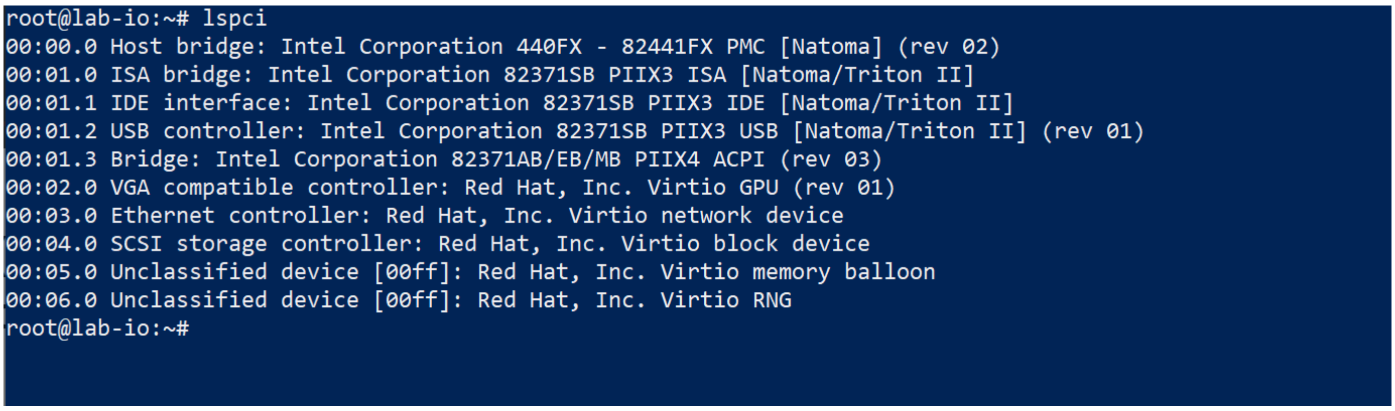
Overview about the available hardware in your system

```
lshw
```



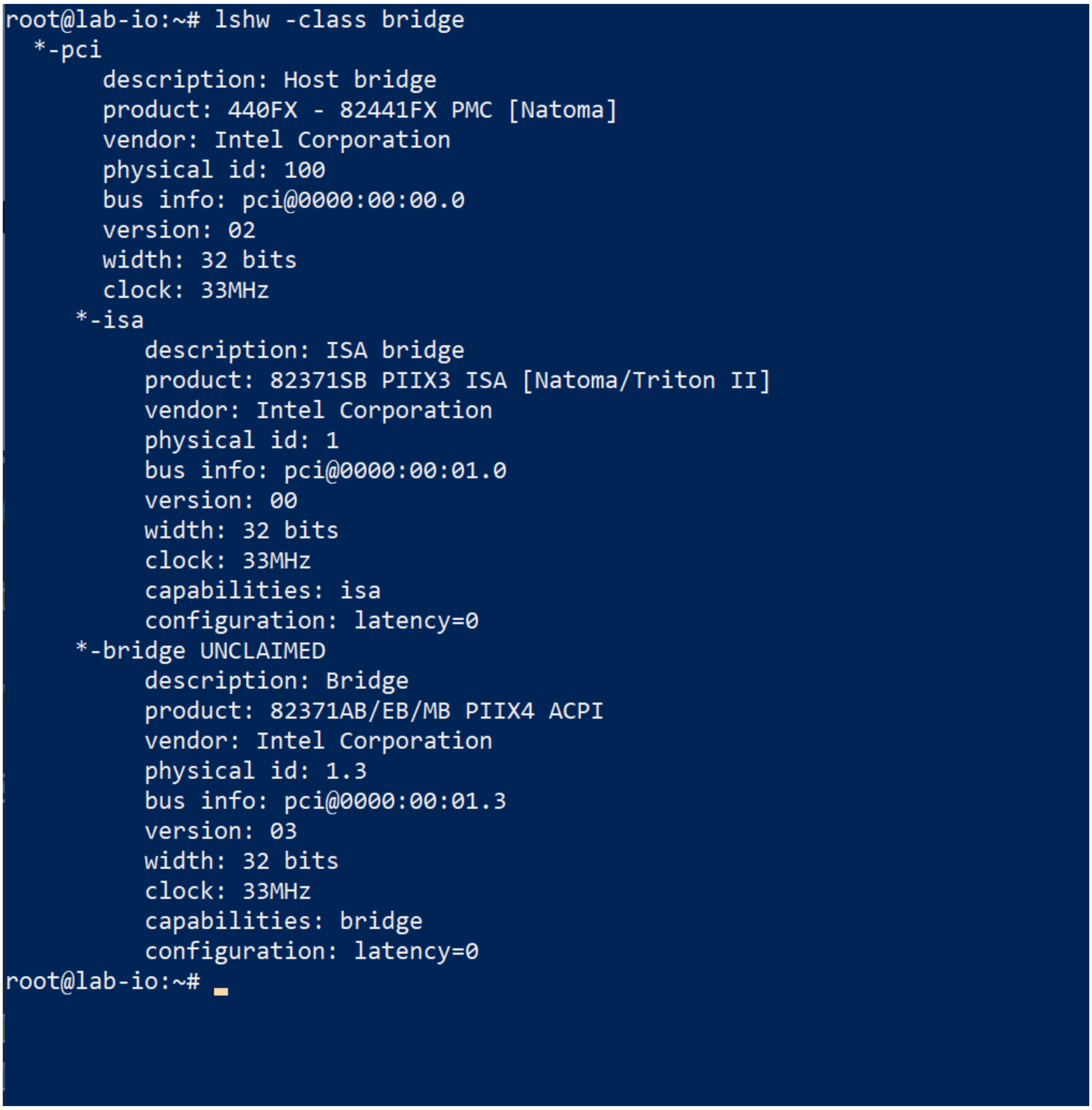
```
root@lab-io:~# lshw
lab-io
    description: Computer
    product: OpenStack Nova
    vendor: OpenStack Foundation
    version: 26.1.1
    serial: 2216861c-c093-47d0-bc79-47136b487263
    width: 64 bits
    capabilities: smbios-2.8 dmi-2.8 smp vsyscall32
    configuration: boot=normal family=Virtual Machine uuid=1C861622-93C0-D047-BC79-47136B487263
  *-core
       description: Motherboard
       physical id: 0
     *-firmware
          description: BIOS
          vendor: SeaBIOS
          physical id: 0
          version: 1.15.0-1
          date: 04/01/2014
          size: 96KiB
     *-cpu:0
          description: CPU
          product: Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz
          vendor: Intel Corp.
          physical id: 400
          bus info: cpu@0
          version: pc-i440fx-6.2
          slot: CPU 0
          size: 2GHz
          capacity: 2GHz
          width: 64 bits
          capabilities: fpu fpu_exception wp vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mm
x fxsr sse sse2 ss syscall nx pdpe1gb rdtscp x86-64 constant_tsc arch_perfmon rep_good nopl xtopology cpuid tsc_known_freq
pni pclmulqdq vmx ssse3 fma cx16 pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand h
ypervisor lahf_lm abm 3dnowprefetch cpuid_fault invpcid_single pti ssbd ibrs ibpb stibp tpr_shadow vnmi flexpriority ept vp
id ept_ad fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm rdseed adx smap xsaveopt arat umip md_clear arch_cap
abilities
          configuration: cores=1 enabledcores=1 threads=1
     *-cpu:1
          description: CPU
          product: Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz
          vendor: Intel Corp.
          physical id: 401
          bus info: cpu@1
          version: pc-i440fx-6.2
          slot: CPU 1
          size: 2GHz
          capacity: 2GHz
          width: 64 bits
          capabilities: fpu fpu_exception wp vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mm
x fxsr sse sse2 ss syscall nx pdpe1gb rdtscp x86-64 constant_tsc arch_perfmon rep_good nopl xtopology cpuid tsc_known_freq
pni pclmulqdq vmx ssse3 fma cx16 pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand h
ypervisor lahf_lm abm 3dnowprefetch cpuid_fault invpcid_single pti ssbd ibrs ibpb stibp tpr_shadow vnmi flexpriority ept vp
```

```
lspci
```



```
root@lab-io:~# lspci
00:00.0 Host bridge: Intel Corporation 440FX - 82441FX PMC [Natoma] (rev 02)
00:01.0 ISA bridge: Intel Corporation 82371SB PIIX3 ISA [Natoma/Triton II]
00:01.1 IDE interface: Intel Corporation 82371SB PIIX3 IDE [Natoma/Triton II]
00:01.2 USB controller: Intel Corporation 82371SB PIIX3 USB [Natoma/Triton II] (rev 01)
00:01.3 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 03)
00:02.0 VGA compatible controller: Red Hat, Inc. Virtio GPU (rev 01)
00:03.0 Ethernet controller: Red Hat, Inc. Virtio network device
00:04.0 SCSI storage controller: Red Hat, Inc. Virtio block device
00:05.0 Unclassified device [00ff]: Red Hat, Inc. Virtio memory balloon
00:06.0 Unclassified device [00ff]: Red Hat, Inc. Virtio RNG
root@lab-io:~#
```

```
lshw bridge
```



```
root@lab-io:~# lshw -class bridge
  *-pci
       description: Host bridge
       product: 440FX - 82441FX PMC [Natoma]
       vendor: Intel Corporation
       physical id: 100
       bus info: pci@0000:00:00.0
       version: 02
       width: 32 bits
       clock: 33MHz
     *-isa
          description: ISA bridge
          product: 82371SB PIIX3 ISA [Natoma/Triton II]
          vendor: Intel Corporation
          physical id: 1
          bus info: pci@0000:00:01.0
          version: 00
          width: 32 bits
          clock: 33MHz
          capabilities: isa
          configuration: latency=0
     *-bridge UNCLAIMED
          description: Bridge
          product: 82371AB/EB/MB PIIX4 ACPI
          vendor: Intel Corporation
          physical id: 1.3
          bus info: pci@0000:00:01.3
          version: 03
          width: 32 bits
          clock: 33MHz
          capabilities: bridge
          configuration: latency=0
root@lab-io:~#
```

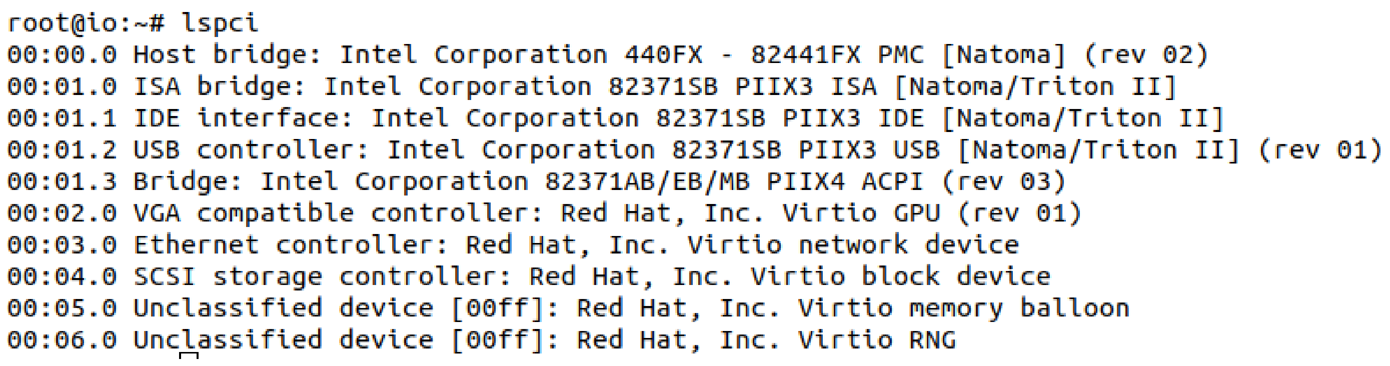
Task 2 – Principles of IO

Physical vs Logical (Virtual) Devices *Double-check your hardware configuration, can you identify logical (virtual) IO devices? If, identify and analyse some and discuss their origin and features.*

To identfy logical IO devices use the following command

```
lspci
```

The "Virtio" states, that as an example the "Ethernet controller" is a virtual IO device.



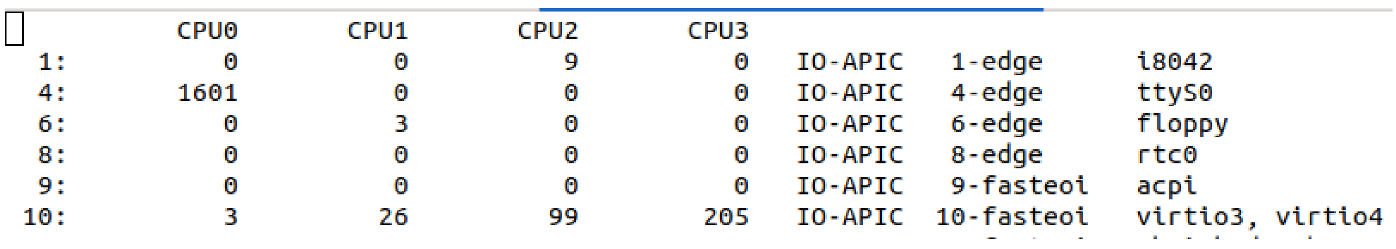
```
root@io:~# lspci
00:00.0 Host bridge: Intel Corporation 440FX - 82441FX PMC [Natoma] (rev 02)
00:01.0 ISA bridge: Intel Corporation 82371SB PIIX3 ISA [Natoma/Triton II]
00:01.1 IDE interface: Intel Corporation 82371SB PIIX3 IDE [Natoma/Triton II]
00:01.2 USB controller: Intel Corporation 82371SB PIIX3 USB [Natoma/Triton II] (rev 01)
00:01.3 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 03)
00:02.0 VGA compatible controller: Red Hat, Inc. Virtio GPU (rev 01)
00:03.0 Ethernet controller: Red Hat, Inc. Virtio network device
00:04.0 SCSI storage controller: Red Hat, Inc. Virtio block device
00:05.0 Unclassified device [00ff]: Red Hat, Inc. Virtio memory balloon
00:06.0 Unclassified device [00ff]: Red Hat, Inc. Virtio RNG
```

1. Type -> Ethernet Controller
2. vendor -> Red Hat
3. description

Synchronous vs Asynchronous Access *Understand the output of /proc/interrupts and discuss the columns in detail.*

Use the following to open /proc/interrupts:

```
vim /proc/interrupts
```



```
           CPU0       CPU1       CPU2       CPU3
  1:          0          0          9          0   IO-APIC   1-edge      i8042
  4:       1601          0          0          0   IO-APIC   4-edge      ttyS0
  6:          0          3          0          0   IO-APIC   6-edge      floppy
  8:          0          0          0          0   IO-APIC   8-edge      rtc0
  9:          0          0          0          0   IO-APIC   9-fasteoi   acpi
 10:          3         26         99        205   IO-APIC  10-fasteoi   virtio3, virtio4
```

1. CPU: Shows the the number of interupts per CPU

2. IRQ: unction block over which the interrupts run.
3. Device: Name of the device

Print the evolution of this file in real-time onto your screen. Hint, you may find the "watch" tool useful.

```
watch -n 1 cat /proc/interrupts
```

Verify your understanding using "lspci -vvv", check for instance, the USB configuration. What can you tell about the interrupt (IRQ) configuration?

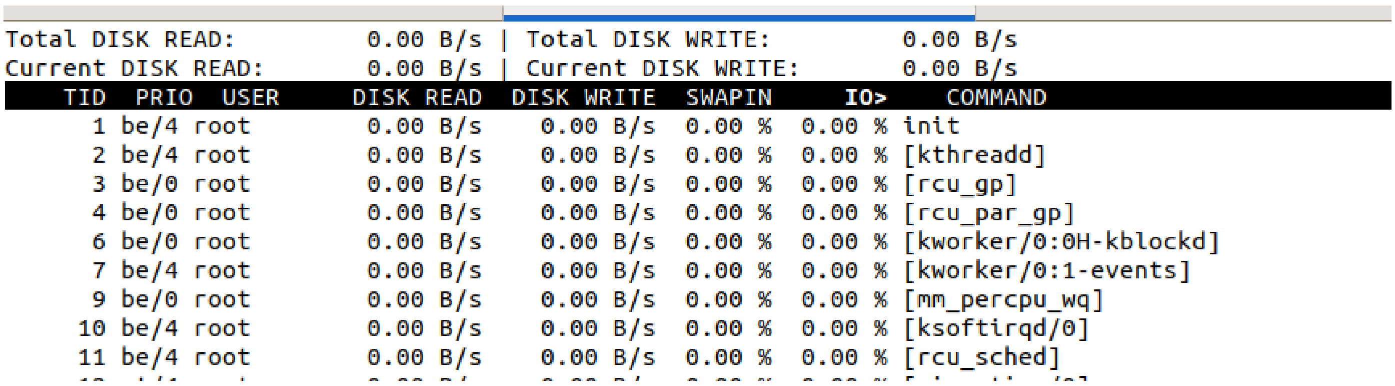
```
lspci -vvv
```

THE IRQ is 11, it's shared between multiple devices like USB, ethernet controller or SCSI storage controller

Shared IO Access

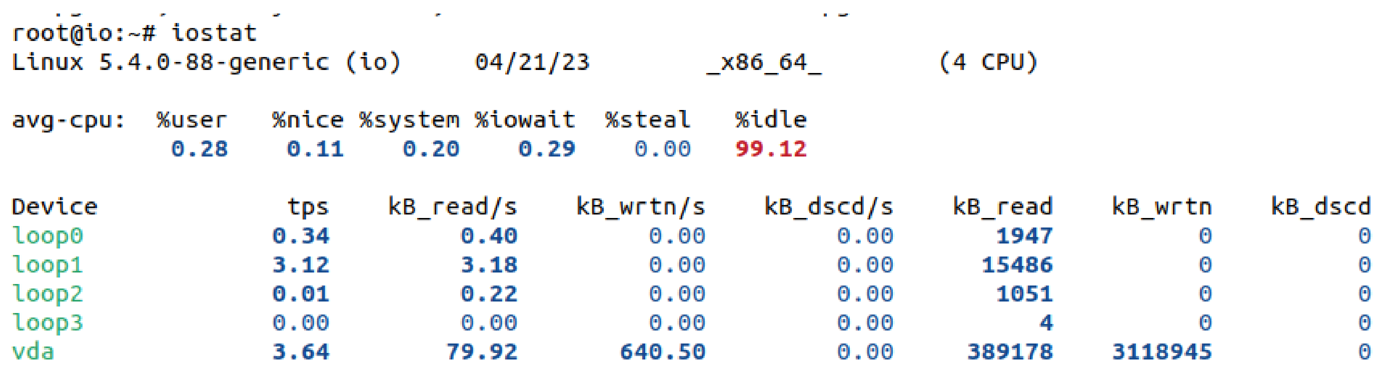
As soon as access to IO is shared, performance becomes a real issue. There are several tools to analyze IO performance, like iostat and iotop. Familiarize yourself with these two by running them and understanding their outputs.

```
iotop
```



```
Total DISK READ:         0.00 B/s | Total DISK WRITE:         0.00 B/s
Current DISK READ:       0.00 B/s | Current DISK WRITE:       0.00 B/s
    TID  PRIO  USER     DISK READ  DISK WRITE  SWAPIN     IO>    COMMAND
      1 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % init
      2 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [kthreadd]
      3 be/0 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [rcu_gp]
      4 be/0 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [rcu_par_gp]
      6 be/0 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [kworker/0:0H-kblockd]
      7 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [kworker/0:1-events]
      9 be/0 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [mm_percpu_wq]
     10 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [ksoftirqd/0]
     11 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [rcu_sched]
```

```
iostat
```



```
root@io:~# iostat
Linux 5.4.0-88-generic (io)     04/21/23        _x86_64_        (4 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.28    0.11    0.20    0.29    0.00   99.12

Device             tps    kB_read/s    kB_wrtn/s    kB_dscd/s    kB_read    kB_wrtn    kB_dscd
loop0             0.34         0.40         0.00         0.00       1947          0          0
loop1             3.12         3.18         0.00         0.00      15486          0          0
loop2             0.01         0.22         0.00         0.00       1051          0          0
loop3             0.00         0.00         0.00         0.00          4          0          0
vda               3.64        79.92       640.50         0.00     389178    3118945          0
```

Now create a scenario in which you create IO load. Try to understand and use the following command: “wc /dev/zero &” . Then analyze the impact via iostat and iotop for several scenarios. Record and discuss what you notice

use:

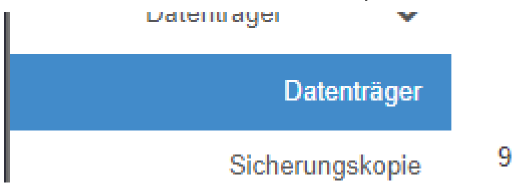
```
wc /dev/zero &
```

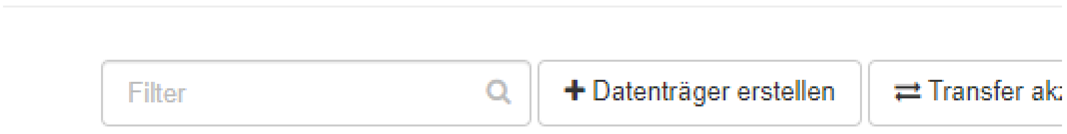
& to run it in the background

Using iotop and iostat shows, that there is no load. The reason behind that is, it's a virtual device, there is no physical workload on an io device.

Now understand what the “dd” tool (man dd) is for. Now run a scenario in which you run several parallel instances of dd, each creating a file of 10G each. Before that make sure that you have enough disk space by creating a new volume via the Openstack user interface, attaching it to your Lab Instance, and mounting it into the filesystem, using /mnt as mountpoint. Hint, you can use “dmesg”, “fdisk” to get info about your new virtual disk, and “mkfs.ext4” and “mount” to format and mount it.

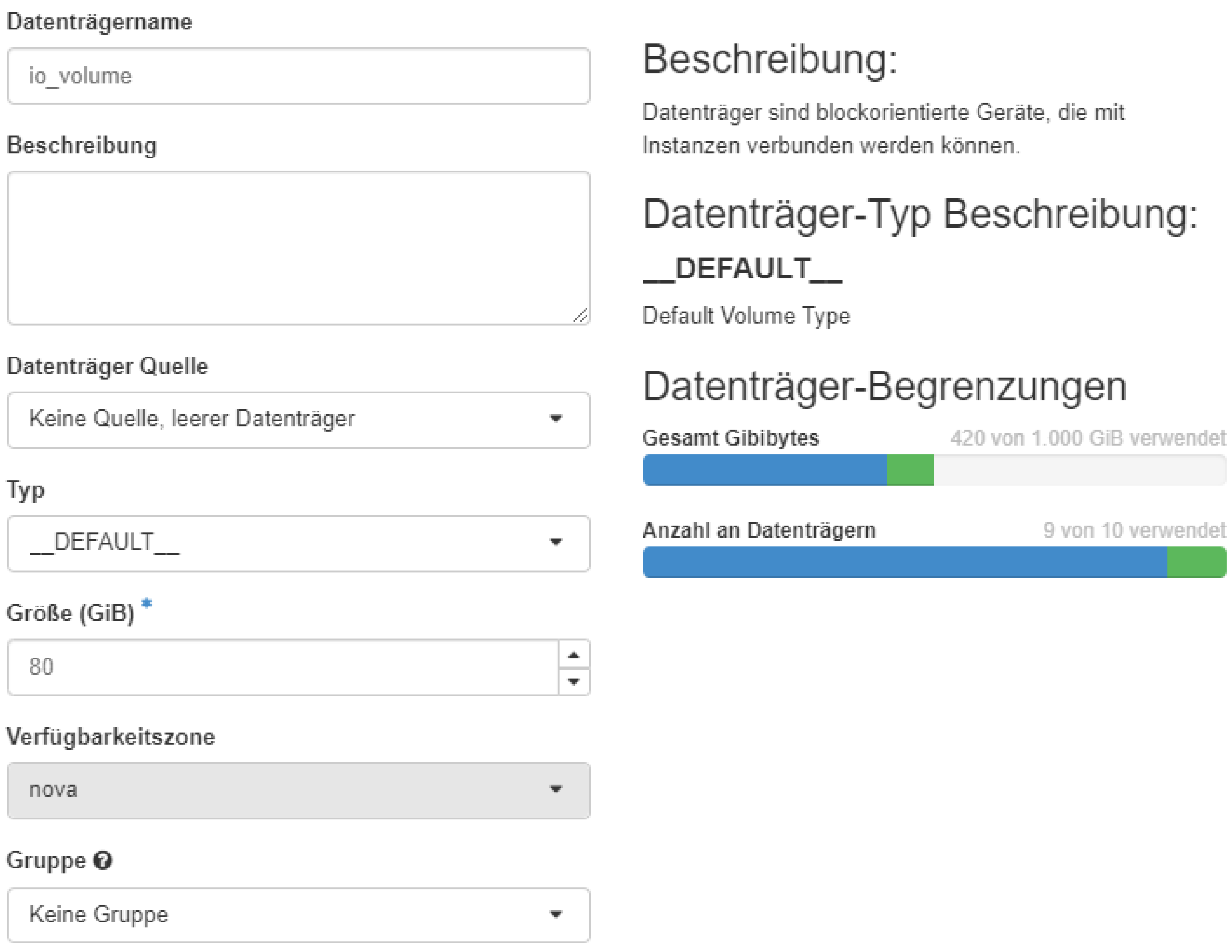
The dd tool is a command-line utility for Unix-like operating systems used to convert and copy files. It can also be used for low-level operations such as copying data between devices, creating disk images, and benchmarking disk performance.

1. Create a new Volume in openstack:
   a. 



b.

c.


Datenträgername: io_volume

Beschreibung:

Datenträger Quelle: Keine Quelle, leerer Datenträger

Typ: __DEFAULT__

Größe (GiB) *: 80

Verfügbarkeitszone: nova

Gruppe: Keine Gruppe

Beschreibung:

Datenträger sind blockorientierte Geräte, die mit Instanzen verbunden werden können.

Datenträger-Typ Beschreibung:

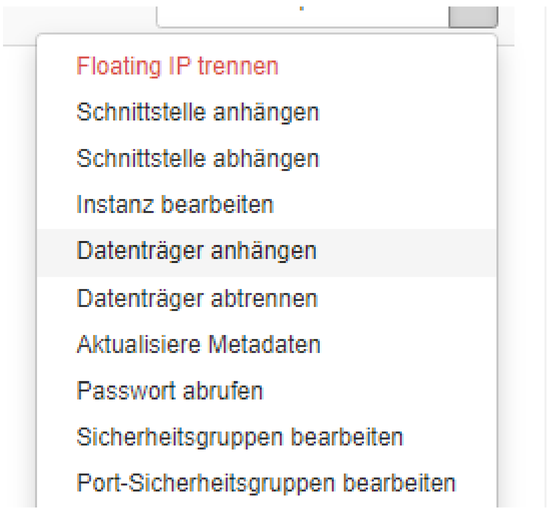
__DEFAULT__

Default Volume Type

Datenträger-Begrenzungen

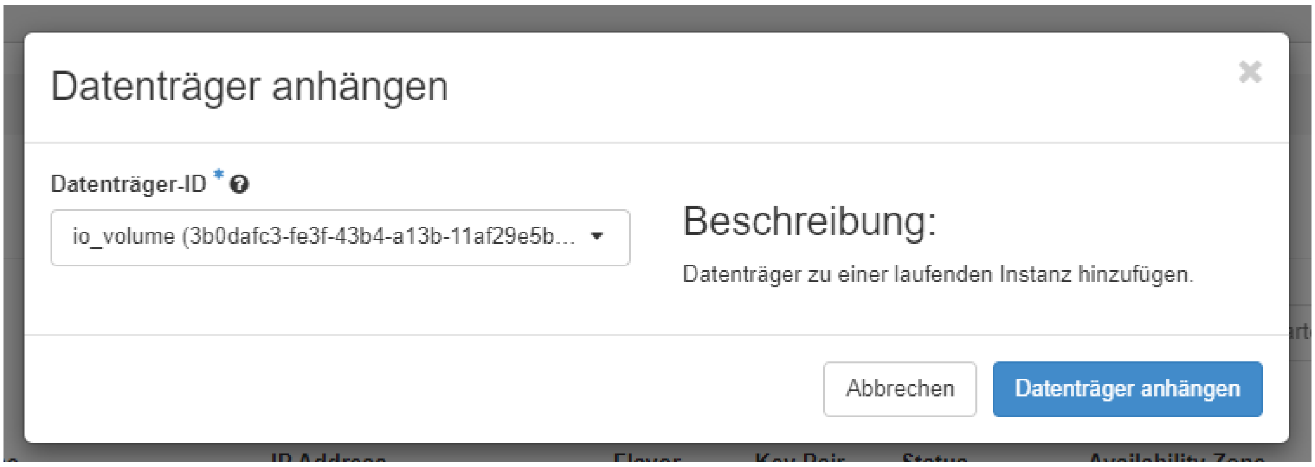
Gesamt Gibibytes: 420 von 1.000 GiB verwendet

Anzahl an Datenträgern: 9 von 10 verwendet



d.

e.


Datenträger anhängen

Datenträger-ID *

io_volume (3b0dafc3-fe3f-43b4-a13b-11af29e5b…

Beschreibung:

Datenträger zu einer laufenden Instanz hinzufügen.

2. Use the following command to get informations about the volumes

```
dmesg
```

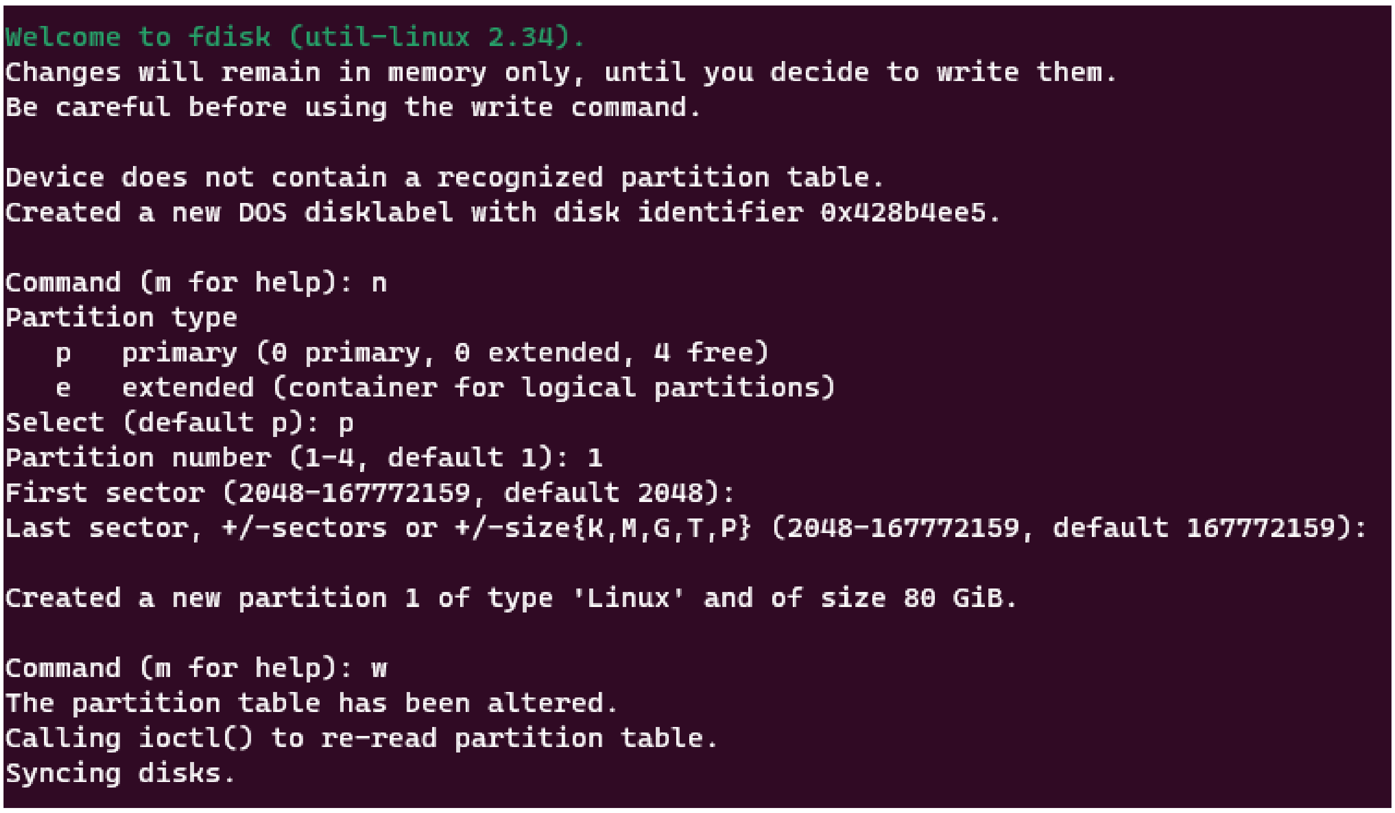
or

```
lsblk
```

to show all volumes

3. Next is creating a new partition with:

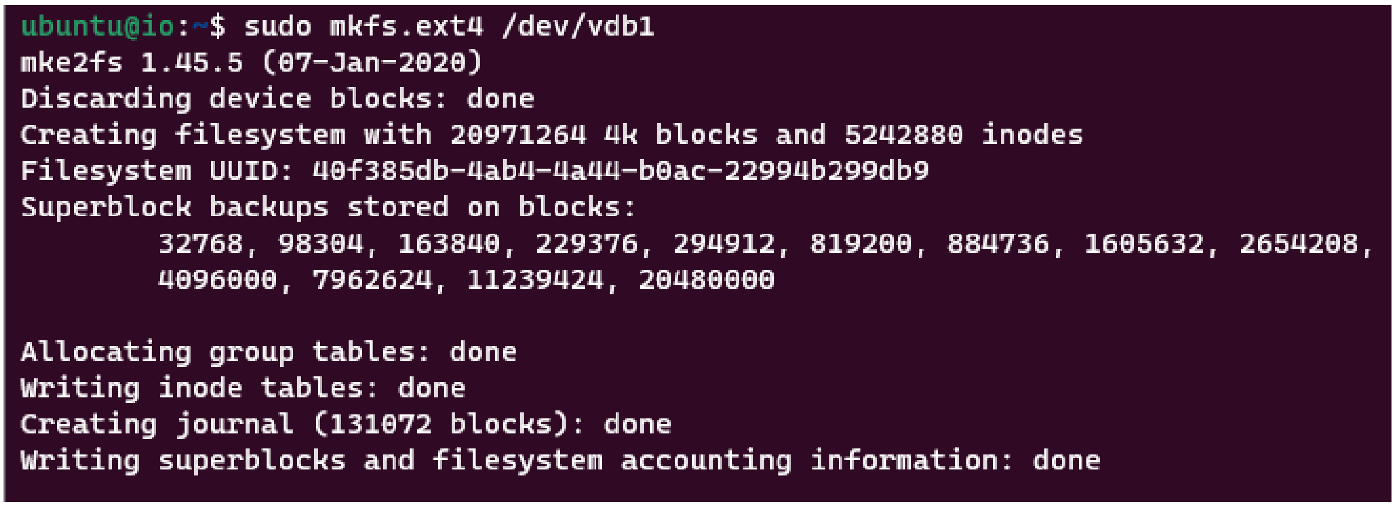
```
sudo fdisk /dev/vdb
```



```
Welcome to fdisk (util-linux 2.34).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x428b4ee5.

Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-167772159, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-167772159, default 167772159):

Created a new partition 1 of type 'Linux' and of size 80 GiB.

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

4. Now the new partition should be formatted:

```
sudo mkfs.ext4 /dev/vdb1
```



```
ubuntu@io:~$ sudo mkfs.ext4 /dev/vdb1
mke2fs 1.45.5 (07-Jan-2020)
Discarding device blocks: done
Creating filesystem with 20971264 4k blocks and 5242880 inodes
Filesystem UUID: 40f385db-4ab4-4a44-b0ac-22994b299db9
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000, 7962624, 11239424, 20480000

Allocating group tables: done
Writing inode tables: done
Creating journal (131072 blocks): done
Writing superblocks and filesystem accounting information: done
```

5. Next it should be mounted:

```
sudo mkdir /mnt
sudo mount /dev/vdb1 /mnt
```

6. In the new partition the 10gb files can be created:

```
sudo dd if=/dev/zero of=/mnt/file1 bs=1G count=10
sudo dd if=/dev/zero of=/mnt/file2 bs=1G count=10
```

On your new disk, create in parallel a number of files, using the following command. “dd if=/dev/zero of=/mnt/iotest/testfile_n bs=1024 count=1000000 &” What exactly is this command doing? What do you have to modify in order to use this command in parallel? Perhaps you want to run this as a shell script, simply put this command line-by-line into a text file and execute the text file (aka bash script file) on the shell. Once load is created, check IO performance. Discuss the results and verify your understanding

This Command creates a file named "testfile_n". The if option sets the input file, in this case all zeros. The of sets the output file. bs = blocksize and count = number of blocks. To Use this command first you need to create the folder and five the ubuntu user ownership. To make it parallel you can use a loop:

```
sudo mkdir /mnt/iotest
sudo chown ubuntu:ubuntu /mnt/iotest
for n in {1..5}; do dd if=/dev/zero of=/mnt/iotest/testfile_$n bs=1024
count=1000000 & done
```

Now experiment with the command “ionice”. Add different schedulers with different priorities to your operations and verify the result. Can you actually run one process faster than another? If yes, explain, if not explain too. Hint, “less /sys/block/vXXX/queue/scheduler” and https://wiki.ubuntu.com/Kernel/Reference/IOSchedulers

ionice is a command that allows to set the priority of a process.

```
ionice -c <class> -n <priority> <command>
```

-c class The scheduling class. 0 for none, 1 for real time, 2 for best-effort, 3 for idle.
-n classdata The scheduling class data. This defines the class data, if the class accepts an argument. For real time and best-effort, 0-7 is valid data.
p pid Pass in process PID(s) to view or change already running processes. If this argument is not given, ionice will run the listed program with the given parameters.
-t Ignore failure to set requested priority. If COMMAND or PID(s) is specified, run it even in

case it was not possible to set desired scheduling priority, what can happen due to insufficient privilegies or old kernel version.

(from https://linux.die.net/man/1/ionice)

Schedulers IO scheduler are used to schedule operations between storage devices and operating system. They are set per device. To view th currently used scheduler, you can use this command:

```
cat /sys/block/<device>/queue/scheduler
```

The ist replaced with the desired block device, ex. "sda".

IO Scheduler can support priorities. When using ionice the the priority for best-effort is 4 by default. The Priorites are depending on the class. While best effort has priority 0-7, the realtime class has 8 priority levels. The best priority is 0 and it gets worse the higher the number is.

Here an example to set a command with the highest priority:

```
  ionice -c 4 -n 0 <command>
```

Class best effort an priority 0 is set for .

Blocking and Non-Blocking Devices Interpret and understand the code below. What is the use case of the function select()?

```
#include <stdio.h>, #include <stdlib.h>, #include <sys/time.h>, #include
<sys/types.h>, #include <unistd.h>
int main(void)
{
fd_set rfds;
struct timeval tv;
int retval;
/* Set rfds to STDIN (fd 0) */
FD_ZERO(&rfds); FD_SET(0, &rfds);
/* Wait up to five seconds. */
tv.tv_sec = 5;
tv.tv_usec = 0;
retval = select(1, &rfds, NULL, NULL, &tv);
if (retval == -1)
perror("select()");
else if (retval)
printf("OK");
/* FD_ISSET(0, &rfds) will be true. */
else
printf("Not OK within five seconds.\n");
exit(EXIT_SUCCESS);
}
```

The code perform a non-blocking I/O operation. It waits for 5 seconds to get userinput, without blocking the device.

Task 3 – Linux Device Model and UDEV

Navigate to /sys and familiarize yourself with the structure. Look for, and understand the content of the “uevent” file for the block device (volume) created for the IO performance task.

```
cd /sys/block/vdb
```

```
cat uevent
```

MAJOR=252
MINOR=16
DEVNAME=vdb
DEVTYPE=disk

MAJOR: identifies the driver
MINOR: if a driver controls several devices the minor number provides a way for the driver to differentiate among them
DEVNAME: device name
DEVTYPE: device type

Enter the /dev directory and look for the device file that represents the volume. What can you see from the file attributes?

```
ls -l | grep vdb
```

```
brw-rw---- 1 root disk 252, 16 Apr 25 15:14 vdb
brw-rw---- 1 root disk 252, 17 Apr 25 15:15 vdb1
```

b = block device rw = read and write, 252: MAJOR (driver id), 16: MINOR

UDEV uses /sys to create devices files in /dev upon the appearance (e.g. hotplugged) of a device. List the rule files of UDEV that define this on-demand behavior. What happens if you attach a mouse as IO input device?

```
cd /lib/udev/rules.d/
```

```
ls
```

```
01-md-raid-creating.rules 60-persistent-storage-tape.rules 70-uaccess.rules 10-cloud-init-hook-hotplug.rules 60-persistent-storage.rules 71-power-switch-proliant.rules 40-vm-hotadd.rules 60-persistent-v4l.rules 71-seat.rules 50-apport.rules 60-sensor.rules 73-seat-late.rules 50-firmware.rules 60-serial.rules 73-special-net-names.rules 50-udev-default.rules 60-tpm-udev.rules 75-net-description.rules 55-dm.rules 61-autosuspend-manual.rules 75-probe_mtd.rules 55-scsi-sg3_id.rules 61-persistent-storage-android.rules 78-graphics-card.rules 56-dm-mpath.rules 63-md-raid-arrays.rules 78-sound-card.rules 56-dm-parts.rules 64-btrfs-dm.rules 80-debian-compat.rules 56-lvm.rules 64-btrfs.rules 80-drivers.rules 58-scsi-sg3_symlink.rules 64-md-raid-assembly.rules 80-net-setup-link.rules 60-autosuspend-chromiumos.rules 66-azure-ephemeral.rules 80-udisks2.rules 60-block.rules 66-snapd-autoimport.rules 81-net-dhcp.rules 60-cdrom_id.rules 68-del-part-nodes.rules 85-hdparm.rules 60-drm.rules 69-bcache.rules 90-bolt.rules 60-evdev.rules 69-lvm-metad.rules 90-console-setup.rules 60-fido-id.rules 69-md-clustered-confirm-device.rules 90-fwupd-devices.rules 60-input-id.rules 70-iscsi-disk.rules 95-dm-notify.rules 60-multipath.rules 70-iscsi-network-interface.rules 95-kpartx.rules 60-open-vm-tools.rules 70-joystick.rules 96-e2scrub.rules 60-persistent-alsa.rules 70-mouse.rules 99-systemd.rules 60-persistent-input.rules 70-power-switch.rules 99-vmware-scsi-udev.rules 60-persistent-storage-dm.rules 70-touchpad.rules
```

For the mouse: 70-mouse.rules

When udev is notified by the kernel of the appearance of a new device, it collects various information on the given device by consulting the corresponding entries in /sys/, especially those that uniquely identify it (MAC address for a network card, serial number for some USB devices, etc.). Armed with all of this information, udev then consults all of the rules contained in /etc/udev/rules.d/ and /lib/udev/rules.d/. In this process it decides

how to name the device, what symbolic links to create (to give it alternative names), and what commands to execute. All of these files are consulted, and the rules are all evaluated sequentially
Source: https://debian-handbook.info/browse/stable/sect.hotplug.html

In short: creates new mouse device in /dev/input

Remove the volume (detach it via OpenStack User Interface) created for the IO performance testing before and look for changes in the /sys and /dev directory.

vdb and vdb1 are no longer visible in /dev

Task 4 – Example IO Device: Hard Drive

iostat (see man iostat) is a useful tool to provide low-level disk statistics.

• Pick one hard drive of your system and display only the device utilization report in a human readable format and using megabytes per second.

```
iostat -d -h -m [device]
```

• What does the output tell you?
tps: Transfers (I/O requests) per second issued to the device
MB_read/s: Amount of data read from the device in megabytes per second.
MB_wrtn/s: Amount of data written to the device in megabytes per second.
MB_read: The total number of megabytes read since system was booted.
MB_wrtn: The total number of megabytes written since system was booted.
MB_dscd/s: The amount of discarded requests in megabyte per second.
MB_dscd: The amount of discarded requests in megabyte since system was booted.

• How can you only show one specific device?

```
iostat -d -h -m [device]
```

• How can you show further extended details of that specific device i.e. sub-devices (hint: /proc)?

```
iostat -d -h -m -p -x [device]
```

- p for partition -x for extended

- Investigate in the /proc file system where disk statistics are taken by iostat for display.

```
root@lab-io:/proc# cat diskstats
   7       0 loop0 1654 0 3894 9260 0 0 0 0 0 1084 7328 0 0 0 0
   7       1 loop1 15157 0 30906 13717 0 0 0 0 0 2400 8804 0 0 0 0
   7       2 loop2 69 0 2102 659 0 0 0 0 0 412 576 0 0 0 0
   7       3 loop3 2 0 10 0 0 0 0 0 0 8 0 0 0 0 0
   7       4 loop4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   7       5 loop5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   7       6 loop6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   7       7 loop7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 252       0 vda 8592 1887 685628 54594 2230 4024 2679330 20797 0 48012
62752 0 0 0 0
 252       1 vda1 7385 1887 661142 52729 2203 4016 2679248 19950 0 47232
61088 0 0 0 0
 252      14 vda14 214 0 1960 134 0 0 0 0 0 200 20 0 0 0 0
 252      15 vda15 886 0 17966 1647 2 0 2 0 0 672 776 0 0 0 0
```

- What is the await field and why is it important?

The average time (in milliseconds) for I/O requests issued to the device to be served. This includes the time spent by the requests in queue and the time spent servicing them. A high await time would show, that the disk can't keep up with the requests.

- If rqm/s is > 0 what does this indicate? What does it hint about the workload?

The number of I/O requests merged per second that were queued to the device. Indicates a higher workload

Now, measure your disk write output with dd (c.f. man dd):
dd if=/dev/zero of=speedtest bs=10M count=100
rm speedtest
dd if=/dev/zero of=speedtest bs=10M count=100 conv=fdatasync

- How can you explain the difference in output?

conv=fdatasync ensures, that all the data from cache has been written to the drive before the process is shown as completed. Important when device is removable and data could be lost when it is removed too early.

- What is a fsync() operation?

In addition to the function of fdatasync, it also transfers all the metadata information associated with the transfered files.