RT schedulers on Xen: test and Installation

Real-Time Industrial Systems

Marcello Cinque, Daniele Ottaviano

Roadmap

- Environment setup: Install Xen on Debian
- Network, GRUB and LVM configuration
- Guest mode and creation
- Xen RT schedulers:
  - RTDS
  - Null scheduler
- CPU pools
- RTDS simple tests

- References:
  - https://wiki.xenproject.org/wiki/Xen_Project_Software_Overview
  - https://wiki.xenproject.org/wiki/Xen_Project_Beginners_Guide
  - https://wiki.xenproject.org/wiki/Xen_Project_Schedulers#Use_cases_and_Support_Status
  - https://xenbits.xen.org/docs/unstable/man/xl.1.html
  - https://xenbits.xen.org/docs/4.10-testing/features/sched_rtds.html

- Full Guide on github: https://github.com/DanieleOttaviano/Xen-Installation-Guide/blob/main/README.md

Environment Setup

Environment setup: Debian Installation

- First, we need to download and Install Debian-11
  - Download: http://cdimage.debian.org/debian-cd/current/amd64/iso-cd/
  - Install Debian:
    - choose the default "**Install**" option.
    - Follow the prompts until you reach the **disk partitioning section** and choose **advanced/custom.** Create these partitions**:**
      - First create the "**/boot**" partition by choosing the disk and hitting enter, make the partition 300MB and format it as ext2, choose /boot as the mountpoint.
      - Repeat the process for "**/**" but of course changing the mountpoint to "/" and making it 15GB or so large. Format it as ext3.
      - Create another partition approximately 1.5x the amount of RAM you have in size and elect to have it used as a **swap** volume (6GB in my case).
      - Finally create a partition that consumes the rest of the diskspace and reserve it for **LVM**

```
sda1 - /boot 300MB
sda2 - / 15GB
sda3 – swap     6GB
sda4 - reserved for LVM
```

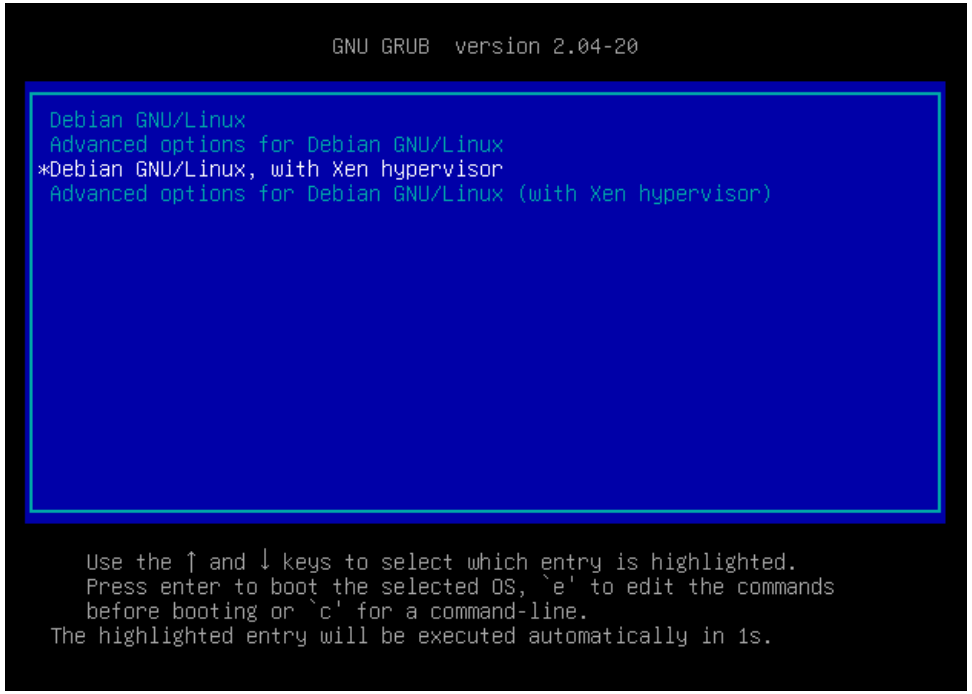
Install Xen on Debian

- We need to install the **Debian Xen Project** via an apt meta-package called **xen-linux-system**
- **The Debian Xen Project** packages consist primarily of:
  - Xen Project-enabled Linux kernel
  - The hypervisor itself
  - A modified version of QEMU that support the hypervisor's HVM mode
  - A set of userland tools

- Install the xen-linux-system meta-package:
  - ***apt-get install xen-system-amd64***

Xen first commands

- When you next boot the system, the boot menu should include entries for starting Debian with the Xen hypervisor
- Logging in as *root* and launch the following command to see the Xen section of **dmesg** created during the boot process:
  - ***xl dmesg***
- **xl** is the Xen Project management tool, based on LibXenlight. You can launch xl to see the existing commands:
  - ***xl***



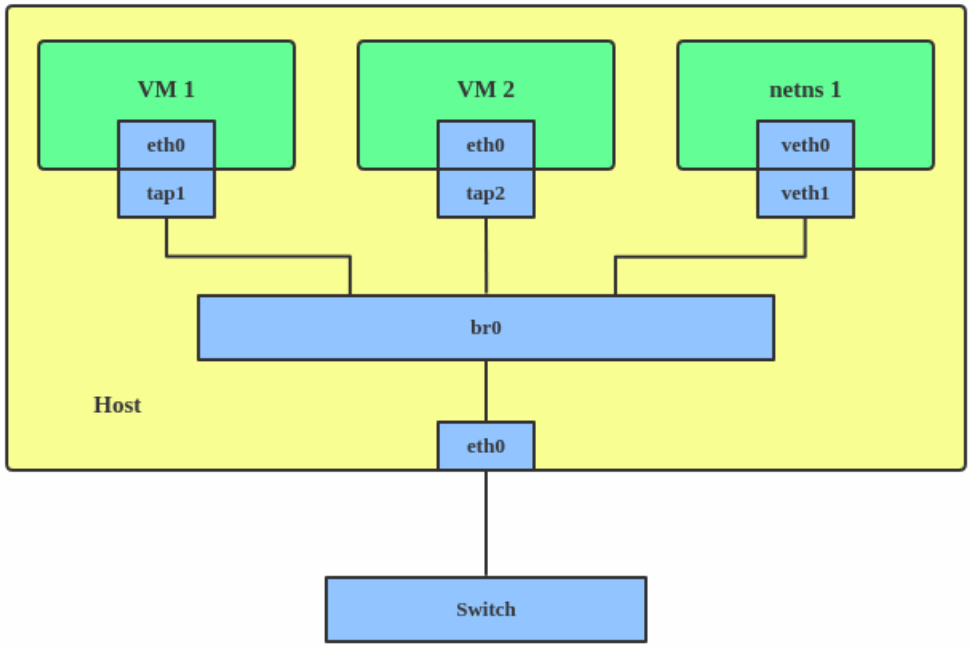
*OSS: if something is going wrong check if the virtualization is enabled in the bios searching for the strings **vmx** or **svm** or **hypervisor** in /proc/cpuinfo:*

- ***egrep '(vmx|svm|hypervisor)' /proc/cpuinfo***

Network, GRUB and LVM configuration

Network Setup

- We need to set up our system so that we can attach virtual machines to the external network. This is done by creating a **virtual switch** within dom0:
  - The switch will take packets from the virtual machines and forward them on to the physical network so they can see the internet and other machines on your network.
- To do it we are going to use **Linux bridge** that is a kernel module already present in Debian kernel. We just need to install an utility to use it:
  - ***apt-get install bridge-utils***



- Advanced configurations: https://wiki.xenproject.org/wiki/Xen_Networking

Network Setup

- To configure the network, we need to modify the interface file
  - ***nano /etc/network/interfaces***

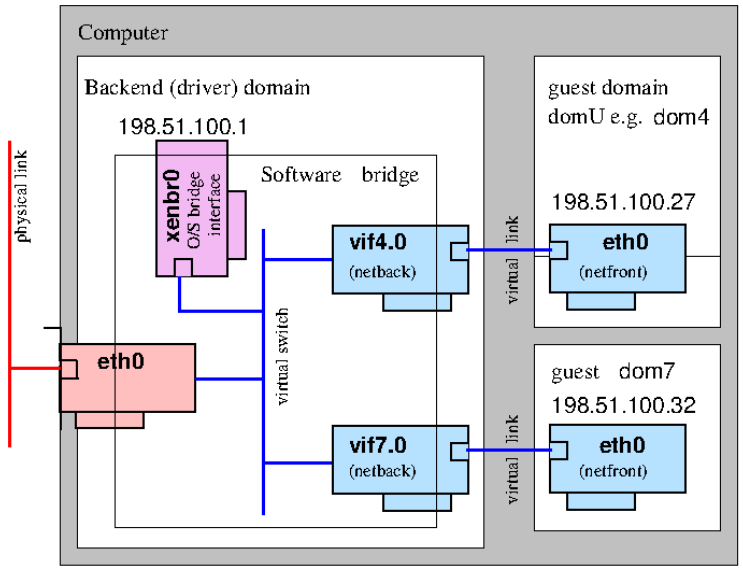
```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug ens33
iface ens33 inet dhcp
```

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug ens33
iface ens33 inet manual

auto xenbr0
iface xenbr0 inet dhcp
        bridge_ports ens33
```



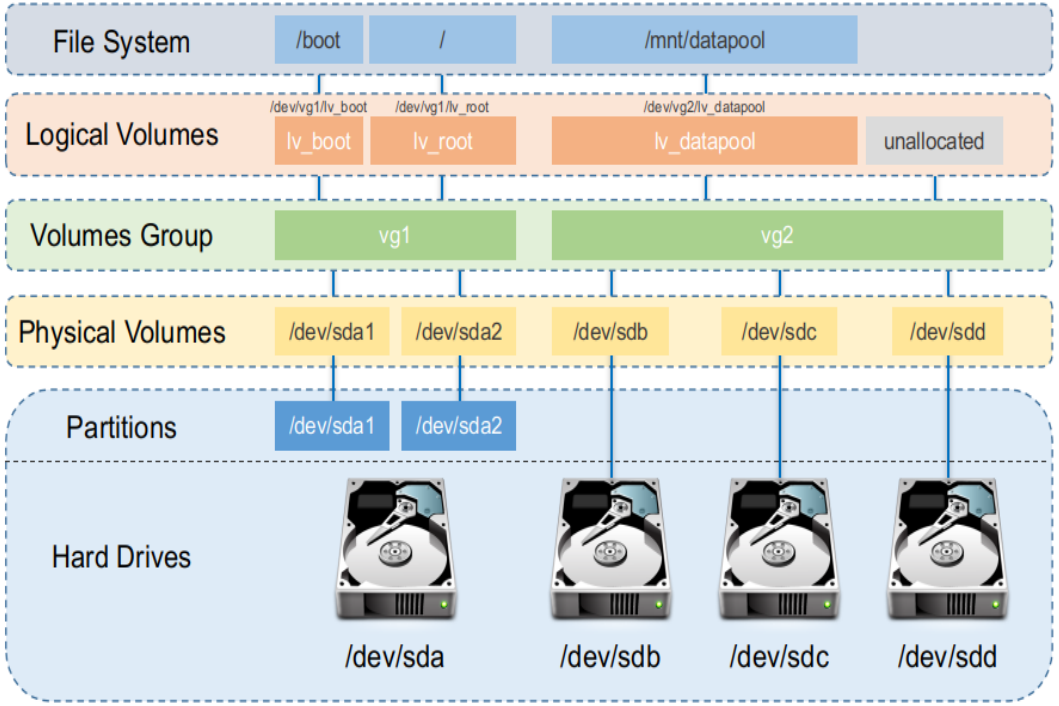
- Depending on your hardware you might notice some differences.
- Each stanza represents a single interface, let's analyze the second:
  - "**allow-hotplug ens33**" means that ens33 will be configured when ifup -a is run, which happens at boot time. This means that the interface will automatically be started/stopped for you.
  - "**iface ens33**" describes the interface itself. In this case, it specifies that it should be configured by DHCP
- In this way, we assign the IP address to the bridged interface. Now restart networking:
  - ***service networking restart***

GRUB Setup

- GRUB (Grand Unified Bootloader) is the bootloader installed during the installation of Debian. It tells the computer which OS to start and how.
- To make sure that GRUB starts Xen before the operating system, we have to modify the file grub:
  - ***nano /etc/default/grub***
- If XEN is the third choice in the GRUB menu, you will change the string *GRUB_DEFAULT=0* into *GRUB_DEFAULT=2* to have Xen load by default.
- Then regenerate the /boot/grub/grub.cfg file by running:
  - ***update-grub***

LVM Setup for guests

- LVM is the Linux ***Logical Volume Manager***. It is a technology that allows Linux to manage *block devices* in a more abstract manner.
- Each "***logical volume***" (lv) is a virtualized block composed of blocks written to one or more physical devices. Unlike the classical disk partition, these blocks don't need to be continuous.
- Logical volumes are created inside the "**volume group**" (vg), which is a set of logical volumes associated to the same physical storage, known as physical volumes.
- The idea is to create a volume group (vg) on top of some physical volumes, and then create a series of logical volumes (lv) each associated with a specific VM.

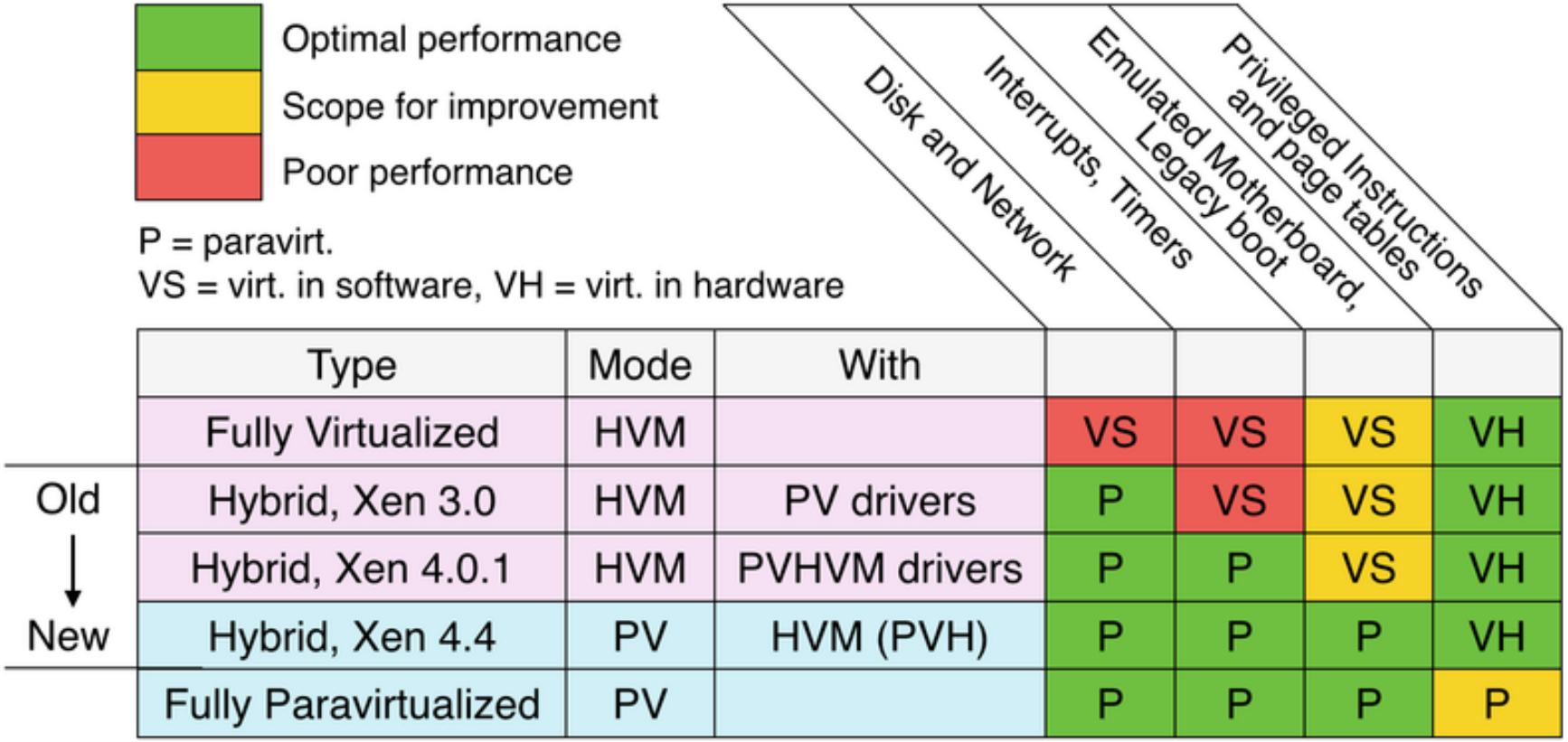


LVM Setup for guests

- First, we need to install LVM:
  - ***apt-get install lvm2***
- Then configure a physical device to maintain a volume group:
  - ***pvcreate /dev/<physical device>***
- Now we can create a volume group using this physical volume:
  - ***vgcreate <name of the group> /dev/<physical device>***
- To create a logical volume for a VM we will use this command:
  - ***lvcreate -n <name of the volume> -L <size, you can use G/M here> <volume group>***
- It is possible to remove the created volume:
  - ***lvremove /dev/<name of the group>/<name of the volume>***

Guest mode and creation

VM guest Modes



	Type	Mode	With				
	Fully Virtualized	HVM		VS	VS	VS	VH
Old	Hybrid, Xen 3.0	HVM	PV drivers	P	VS	VS	VH
↓	Hybrid, Xen 4.0.1	HVM	PVHVM drivers	P	P	VS	VH
New	Hybrid, Xen 4.4	PV	HVM (PVH)	P	P	P	VH
	Fully Paravirtualized	PV		P	P	P	P

- See: https://wiki.xenproject.org/wiki/Understanding_the_Virtualization_Spectrum

PV guest creation (Ubuntu)

- Configure a physical device to maintain a volume group:
  - ***pvcreate /dev/sda4***
- Create a volume group using this physical volume:
  - ***vgcreate vg0 /dev/sda4***
- Create a local volume partition for the new VM:
  - ***lvcreate -L 5G -n lv_vm_ubuntu /dev/vg0***
- Download the netboot image for Ubuntu 18.04:
  - wget http://archive.ubuntu.com/ubuntu/dists/bionic-updates/main/installer-amd64/current/images/netboot/xen/vmlinuz
  - wget http://archive.ubuntu.com/ubuntu/dists/bionic-updates/main/installer-amd64/current/images/netboot/xen/initrd.gz
- To create the VM, we will use the command "***xl create***". This command needs to parse a configuration file that we can write manually. Go into the path */etc/xen/* and create a file *".cfg"*

PV guest configuration file

- Once the configuration is created and the netboot image is downloaded, we can finally create the VM:
  - ***xl create -c /etc/xen/ubuntu_vm_example.cfg***
- You can leave and enter in the guest virtual console :
  - leave : "***ctrl+]***"
  - re-enter :"***xl console <VM name>***"
- After the first installation of ubuntu is completed, you must modify the configuration file:

```
# kernel = "/root/vmlinuz"
# ramdisk = "/root/initrd.gz"
bootloader = "/usr/lib/xen-4.11/bin/pygrub"
```

```
# Guest name
name = "ubuntu_vm"
# Kernel image to boot
kernel = "/root/vmlinuz"
# Ramdisk (optional)
ramdisk = "/root/initrd.gz"
# Initial memory allocation (MB)
memory = 1024
# Number of VCPUS
vcpus = 2
# physic cpu
cpus="2,3"
# Network devices
# A list of 'vifspec' entries as described in
# docs/misc/xl-network-configuration.markdown
vif = ['bridge=xenbr0']
# Disk Devices
# A list of `diskspec' entries as described in
# docs/misc/xl-disk-configuration.txt
disk = [ '/dev/vg0/lv_vm_ubuntu,raw,xvda,rw']
# it depends on the xen version installed, 4.11 in this
example
# bootloader = "/usr/lib/xen-4.11/bin/pygrub"
```

Xen RT schedulers

Setup of RTDS on Xen

- Once Xen is installed, the default scheduler is **credit2**, that is a general purpose, weighted fair share scheduler. (run "***xl info***" to check the current scheduler)
- This kind of scheduler is **not suitable for Real-Time**. Hence, we can change the scheduler in favour of **RTDS** from the configuration file*:*
  - ***nano /etc/default/grub.d/xen.cfg***

```
# GRUB_CMDLINE_XEN_DEFAULT=""
GRUB_CMDLINE_XEN="sched=rtds"
```

- Update Grub to make the changes effective:
  - ***update-grub***
- If you reboot the system and run the command "***xl info***", you should see RTDS as scheduler.
- ***OSS: you can do the same with other XEN schedulers like "null-scheduler" or "ARINC653"***

RTDS scheduler configuration

- It is possible to set the parameters of the scheduler running "**xl sched-rtds**" with the following parameters:
  - -d DOMAIN, --domain=DOMAIN
  - -v VCPUID/all, --vcpuid=VCPUID/all
  - -p PERIOD, --period=PERIOD
  - -b BUDGET, --budget=BUDGET
  - -e Extratime, --extratime=Extratime
  - -c CPUPOOL, --cpupool=CPUPOOL
- As an example, we can assign to our ubuntu VM a period of 1000us and budget 500us:
  - ***xl sched-rtds -d ubuntu_vm -v all -p 1000 -b 500 -e 0***

Xen Null Scheduler

- In cases where one is absolutely sure that there will be less vCPUs than pCPUs, having to pay the cost, mostly in terms of **overhead**, of an advanced scheduler may be not desirable. In this case it is possible to use the **null_scheduler**
  - ***nano /etc/default/grub.d/xen.cfg***

```
# GRUB_CMDLINE_XEN_DEFAULT=""
GRUB_CMDLINE_XEN="sched=null"
```

- Update Grub to make the changes effective:
  - ***update-grub***

```
xen_version            : 4.14.3
xen_caps               : xen-3.0-x86_64
xen_scheduler          : null
xen_pagesize           : 4096
platform_params        : virt_start=0xffff800
```

Dom0 vCPU limitation

- If null-scheduler is used as default scheduler, at Xen boot, **it is recommended to limit the number of Dom0 vCPUs**.
  - Otherwise, all the pCPUs will have one Dom0's vCPU assigned, and there won't be room for running efficiently (if at all) any guest.
- To do it we must change the same configuration file: */etc/default/grub.d/xen.cfg*.
- For example I decided to assign 2 vCPU to dom0 (dom0_max_vcpus=2) and to pin these vCPUs to two fixed pCPU(dom0_vcpus_pin):

```
# GRUB_CMDLINE_XEN_DEFAULT=""
GRUB_CMDLINE_XEN="dom0_max_vcpus=2 dom0_vcpus_pin sched=null"
```

Fixed amount of memory for Dom0

- This is **not mandatory** but dedicating a fixed amount of memory for dom0 is good for two reasons:
  - First of all (dom0) Linux kernel calculates various network related parameters based on the boot time amount of memory.
  - The second reason is Linux needs memory to store the memory **metadata** (per page info structures), and this allocation is also based on the boot time amount of memory.
- If you boot up the system with dom0 having all the memory visible to it, and then balloon down dom0 memory every time you start up a new guest, you end up having only a small amount of the original memory.
- ***You end up wasting a lot of memory for the metadata for a memory you don't have anymore.***

Fixed amount of memory for Dom0

- So, first we have to configure the toolstack to make sure dom0 memory is never ballooned down.
- It is enough to modify the xl configuration file in the following path */etc/xen/xl.conf*:

```
# Control whether dom0 is ballooned down when xen doesn't have enough
# free memory to create a domain. 'auto' means only balloon if dom0
# starts with all the host's memory
Autoballoon=0
```

- Then we can fix the quantity of RAM of Dom0 in the grub configuration file */etc/default/grub* (then *upgrade-grub* and *reboot*):

```
# GRUB_CMDLINE_XEN_DEFAULT=""
GRUB_CMDLINE_XEN="dom0_mem=1024M,max:1024M dom0_max_vcpus=2 dom0_vcpus_pin
sched=null"
```

CPU Pools

CPU pools

- CPUpools is a introduced in Xen 4.2 which allows you to divide your physical CPUs into distinct groups called "**cpupools**".
- Each pool can have its entirely separate scheduler. Domains are assigned to pools on creation and can be moved from one pool to another.
- On boot, a "default pool" named **Pool-0** will be created.
  - Run "***xl cpupool-list***" to see the created pools

```
root@debian:~# xl cpupool-list
Name               CPUs   Sched     Active   Domain count
Pool-0               4   credit2       y          1
```

  - Run "***xl cpupool-list -c***" to see which CPUs are associated to a pool

```
root@debian:~# xl cpupool-list -c
Name               CPU list
Pool-0             0,1,2,3
```

CPU pools

- We can create a new pool with a specific scheduler:
  - ***xl cpupool-create name=\"testing\" sched=\"rtds\"***
- To use it we need to remove some CPUs from Pool-0 and add them to the newly created pool:
  - ***xl cpupool-cpu-remove Pool-0 3***
  - ***xl cpupool-cpu-add testing 3***

- By default, once you create a new VM it is assigned to the default pool (Pool-0). Hence, if you want to use another pool there are three ways:
  1. We can assign a VM to run on a cpupool at creation time:
     - ***xl create /etc/xen/ubuntu_vm_example.cfg pool="testing“***
  2. We can modify the configuration file (*/etc/default/grub.d/xen.cfg*) adding the string:
     - ***pool="testing“***
  3. We can migrate a particular VM to run on a particular cpupool:
     - ***xl cpupool-migrate ubuntu_vm Pool-0***

RTDS simple tests

RTDS tests

- To test if the scheduler is working fine you can:
  1- Configure RTDS as scheduler.
  2- Create two VMs with 1 vCPU and associated to the same pCPU.
  3- Set the scheduling parameters such as it has a 50% reservation.
  4- Run a CPU-burning process inside the VMs (e.g., yes or stress-ng).
  5- Check with "***xl top***" that the VM is getting no more than 50% pCPU time.