Exercise Introduction

The first exercise will guide you to install VirtualBox on your personal computer and run the Linux OS Ubuntu Desktop via VirtualBox as a VM.
The second task will require you to utilize the HPC system to run a simple MPI program.
This sheet also contains an optional task with installation instructions for setting up X2Go in order to run graphical applications via SSH.

Contents

Task 1: Setting up a Linux Virtual Machine (30 min)

Here we present instructions for running a Linux environment on your personal computer via VirtualBox. This environment can then be used for experimentation during the student projects as you have superuser permissions on these VMs while we can only provide you with user permissions on the HPC-cluster.

In order to run a VM, your host system needs to have enough resources available to effectively simulate a second operating system. These resources are then reserved by the VM such that they are no longer available to the host. Your system should have at least 2 GB RAM (recommended 4 GB or more), 2 CPU cores (recommended 4 cores or more) and at least 8 GB of disk space (recommended 20 GB) to spare.

If you are already using Linux as a daily driver or have access to dedicated Linux machines you can forgo setting up a Linux VM. Nevertheless, in order to provide a save environment for experimenting with installing and modifying software, it can still be sensible to set up such an environment.

Linux VM via VirtualBox

Check if your system supports hardware virtualization:

1. On Windows press WIN+r
2. Type in `cmd` and press enter
3. Type in *systeminfo* and press enter
4. After a moment it should show a message about the status of **Hyper-V** at the bottom
5. If its enabled you are done, if its not you need to enable it, if its not supported, you cannot use VirtualBox

Enabling virtualization:

1. Reboot your system into the BIOS by pressing the BIOS key for your system during boot, commonly its one of these: Esc, Del, F1, F2, F4, F12

2. In the BIOS navigate to CPU settings and look for a virtualization setting and enable it
3. Save the changes and reboot
4. On Windows, search for a menu called **Turn Windows features on or off** and open it
5. Search for a feature called **Hyper-V**, enable it, press **okay** and wait for it to install it
6. Reboot again

Installing VirtualBox and VM setup:

1. Download and install VirtualBox on your system `https://www.virtualbox.org/wiki/Downloads`
2. Download a Linux VM image, to skip the installation process and get started right away, we recommend a pre-installed image from OSBoxes.org.[1] Download the newest 64-bit Ubuntu Desktop image `https://www.osboxes.org/ubuntu/` and check the default username and password under info. Its commonly *osboxes* as username and *osboxes.org* as password. Also note that the default keyboard layout is US Qwerty. This can be adjusted in the settings of the running VM.
3. Open VirtualBox and press **New**
4. Set a name for it and a folder where the files can live
5. Set type as Linux and version as Ubuntu 64-bit and press next
6. Set memory size to 2048 MB (or more) or at least 1024 MB depending on your hardware and press **next**
7. For hard disk selection press **Use an existing hard disk file** and open the selector
8. In the selector, add the **.vdi** file you downloaded from osboxes.org, confirm and press **create**
9. Now you can start the VM via VirtualBox and you should shortly see a login prompt
10. Use the default username and password to login
11. Now you have Ubuntu as a Linux OS running in virtualization on your machine. Search for terminal or press CTRL+ALT+t to open a shell
12. Run `sudo apt update && sudo apt upgrade -y` to get all updates

Hints

- By default, the VM will run with a resolution of 800x600. When selecting the Window in monitor icon at the bottom and navigating to **Virtual Screen 1**, there should be more options but greyed out. To enable these, shutdown the VM and navigate in VirtualBox to **Settings** and **Display**. Here set **Graphics Controller** to **VBoxSVGA** and set the **Video Memory** slide to the maximum. Confirm the settings with **OK**. After restarting the VM, the other resolutions should now be available.

- If the VM is running slow, you might need to increase the number of CPU cores dedicated to the VM in VirtualBox.

- Modern Apple laptops now commonly ship with Apple's own ARM-based chipset called M1 (or M2) instead of any widespread x86_64-based CPUs. For such a system the utm VM manager (`https://mac.getutm.app/`) should be used instead.

- Furthermore, M1 devices require software to be built specifically for ARM devices and as ARM devices are less common, this will limit the pool of available prebuilt software for such a system. The above described osboxes, for example, do not provide prebuilt images for ARM. To install an OS such as Ubuntu Desktop, one has to install it from the ARM 64 iso file (for Ubuntu ARM iso files see here: `https://cdimage.ubuntu.com/focal/daily-live/current/`)

- Alternative VM-managers to VirtualBox exist, such as QEMU and virt-manager but their cross-platform support is often limited. If your system supports an alternative to VirtualBox that you prefer, feel free to use it.

[1]You are free to install the system from an installer image, however, this will take extra time.

- A video guide that was created for a previous course can be found here: `https://www.youtube.com/playlist?list=PLhu3GTWaNSho7v8Zbmkn06wWYHQbdBrfI`

Task 2: Performance of Parallel Primes (30 min)

When operating with HPC systems, performance becomes a key factor. For this task you will compile an MPI program and run it via Slurm. Furthermore, you will inspect the performance of the program.

Complete the following steps:

1. Download the source code for the program `primes.c` from the webpage `https://hps.vi4io.org/teaching/autumn_term_2023/hpcsa`[2]
2. Connect to the HPC cluster if you aren't already connected
3. Save the program to a new file on the cluster and compile it to `primes`
4. Test out that it works `mpirun -np 2 primes`
   The program computes the number of primes up to a given number.
5. Run the job via Slurm `srun --nodes=2 --tasks-per-node=4 primes`
6. When Slurm takes your job, it gives you a *jobid*
   Using this id, run the following command
   `sacct -j JOBID --format="JobID,Elapsed,CPUTime"`
7. Observe the `Elapsed` time as the wall clock time the program ran and `CPUTime` as the time it spent by the CPUs
8. Experiment with different numbers of nodes and tasks per node. (Please do not use too many nodes.)

Optional Task 3: Graphical Interface via X2Go and xterm (0 min)

This is a difficult **additional** task that will support your understanding in the topic.

When working with the HPC system via SSH it might be necessary to work with a graphical application running on the cluster, e.g., vampir), however the cluster does not provide a desktop. To forward the graphical interface via SSH we recommend X2Go. The following instructions will explain how to set up X2Go on your machine:

Setup:

1. Download and install the X2Go client for your operating system: `https://wiki.x2go.org/doku.php`
2. Start X2Go and create a new session if it does not prompt you to create one
3. Set **Session name** to a name such as `HPCSA 2023`
4. Set **Host** to `login-mdc.hpc.gwdg.de`
5. Set **Login** to `hpctrainingNN` , where **NN** is the number in the key file name
6. Set **Use RSA/DSA key for ssh connection** to the ssh key file you received from us
7. Tick **Use Proxy server for SSH connection**, which should open another menu

[2]The program was adapted from here `https://people.sc.fsu.edu/~jburkardt/c_src/prime_mpi/prime_mpi.c`

8. Set **Type** to **SSH**
9. Set **Host** to `login.gwdg.de`
10. Set **Login** to `hpctrainingNN`
11. Set **RSA/DSA key** to the ssh key
12. Set **Session type** to **Single application**
13. In the new drop-down menu on the bottom right select **Terminal**
14. Switch to the third tab **Input/Output** and set **Display** to **Custom** and a resolution you are comfortable with
15. Confirm and close the settings page

Connecting:

1. Open X2Go if it is not already open
2. Double click on the session you created for this course or type in its name and press ENTER
3. Enter the passphrase of your ssh key **twice**
4. After a delay you will be connected to the SCC in a xterm session

You can confirm that it works by running `module load vampir` and `vampir`, which should open vampir in another window.

MacOS users might need to install XQuartz as prompted by a dialog.