Computer Structure

Computadors – Grau en Ciència i Enginyeria de Dades – 2020-2021 Q2

Facultat d'Informàtica de Barcelona

The Computer Structure topic of this course introduces multiple hardware related concepts. Even though GCED is not hardware oriented, it is important you get experience on collecting information from computer devices, as well as understand the impact of them on the performance of apps.

During this Lab Session you will play with several commands that present characteristics of the host. Although we introduce them, we suggest you query information to the "man" about the commands. Afterwards, we present you a collection of codes that you have to compile and run to analyze the impact of different execution settings on the system performance.

The Processor and Components Layout

Linux-like Operating Systems comprise a special folder, "/proc" (process information pseudo-filesystem), that contains many data about environment characteristics as well as the current status of what is going on in the system. It would be good that you read the "man 5 proc" information to dive into information of particular files we will access.

As first step, we will collect information about the processor, which can be found in "/proc/cpuinfo". Among the specs, you will see "siblings" that refers to the total number of hardware threads (also known as processing units) of the processor. There is actually a particular command, called "lscpu", that presents complementary information, mostly collected from this file.

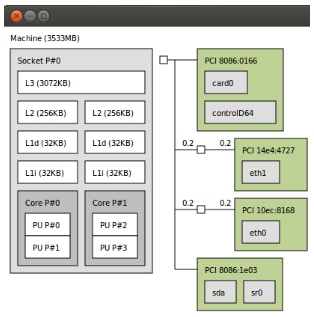
Exercise 1

Edit an "answers.txt" file and indicate the following information about the processor: Model of the processor, number of cores, number of threads per core, frequency of the processor, and byte-order. These specs have been explained in Theory lectures.

There is another command called "lstopo" that trigger a graphical representation of the computer elements arrangement. This tool (included in the "hwloc" package) is not installed by default in Linux distributions. Although it is installed in the images used in some Labs, if your image does not include it, there is no chance to install the package by the student in the Lab (it has to be done by the IT staff). On the other hand, the OpenSUSE VM image does not include it. If you are using such VM, you may check the link from OpenSUSE (OpenSUSE Leap 42.3 version). Finally, if you are using the Ubuntu VM image, you may execute the following command lines:

```
#>sudo dpkg --configure –a
#>sudo apt install hwloc
```

Once we are sure we have the "lstopo" command available, launch it from the terminal. You may see a graphical layout like this:



Thus, you are able to see the arrangement of the key components of the system in a similar way than explained in theory lectures. If you are curious to get further details, you may execute “#>lstopo –v”, which prints more specs about displayed components. Nevertheless, we suggest you to execute an alternative version of this command, namely: “#>lstopo-no-graphics”.

Exercise 2

Edit the “answers.txt” file and, from the output of the above command lines, indicate what is the hardware component pluged-in every PCI slot of your host.

The Memory Hierarchy

From all previously mentioned command lines you may find information about memory hierarchy.

Exercise 3

Edit the “answers.txt” file and, from the output of the above command lines, indicate: what cache levels you have found; their goal (i.e. instructions, data, or both); what is the size of everyone; whether the cache unit is per core or shared among cores.

There is another tool called “numactl” that controls NUMA policy for shared memory, among other goals. For the following exercise, we suggest you invoque it as “#>numactl –H”.

Exercise 4

Edit the “answers.txt” file how many memory nodes your system has, as well as the size and how much is still free. In other words, this is the status summary of your memory.

USB based external components

There is another command called “lsusb”. We suggest you invoque it as “#>lsusb –v”, which shows information about plugged USB devices.

Exercise 5

Edit the “answers.txt” file and list the USB devices you have found in your system.

Hard Disk

There is another command called “df”. We suggest you invoque it as “#>df –H”, which shows information about different logical partitions of the hard disk/s. That is, every row is managed like an independent disk (we will study more details about them in other lessons of this course).

Exercise 6

Edit the “answers.txt” file and list the entries you have found in your system.

Unfortunately, in the Lab you do not have permissions to run another command line that would give you further details about the hard disk. We strongly suggest you to run the command line: “#>sudo hdparm –I XXX”. This command line means the following: “sudo” asks for temporary higher permissions; “hdparm” shows hard disk parameters; and “XXX” is the filesystem to be inspected (e.g. “/dev/sda1”) out of the filesystems shown by the previous command “df”. Actually, “df” shows several logical disks that cannot be analyzed by “hdparm”, such as “udev” or “tmpfs”. From the output of "hdparm", just try to identify parameters discussed during the theory lectures of Computer Structure lesson.

What if…

In this section we will play with different environment settings. To do this, you have to download a file attached to the following link:
https://docencia.ac.upc.edu/FIB/GCED/COM/documents/Lab/FilesS4.tar.gz
Once you uncomprise the file, as indicated in the slot, you will see three C++ codes and another file, named "launch.sh". In the following exercises, if you are using a VM, try different Hw configurations (e.g. changing the number of cores or hardware threads).

Exercise 7

Compile the source codes included in the session file with the "make" command and using the attached Makefile. As a result you should get the executables named "floats", "integers", and "mems". Write in the "answers.txt" file the command line you have executed to compile all of them at once.

The "launch.sh" file is a special group of command lines (also known as shell script) that can be interpreted by the Shell Bash in your terminal. This file needs pairs of input parameters. The first parameter indicates the n-th processing unit that will run the program indicated by the second parameter. For example, if you type:
"#>./launch.sh 0 ./integers 1 ./floats 1 ./mems"

It will execute the program "./integers" in the processing unit "0", whilst "./floats" and "./mems" will run in the processing unit "1".

We strongly suggest you open another terminal and run a command called "top". This command shows the current status of the system and all processes running on it. The output is automatically updated every 2 seconds, by default (although it can be configured to other frequencies). To quit from it, press "Q" button. You should focus on the column "%CPU" that means percentage of the CPU that is consuming every process. Once you write the command line to call "launch.sh" you should automatically see in the output of "top" the name of the programs you indicated.

Finally, after the end of every program execution, there is another output similar to:

```
3.20user  0.00system  0:04.12elapsed  99%CPU …
```

You ONLY have to take note of the "elapsed" time (that is, "4,12" in this example) and "%CPU". The former value indicates how much time (in minutes, seconds and decimals) the program took to finish the execution. The latter value is the average consumption of the CPU by this particular program execution. PLEASE, wait till these data appear in the terminal (one measurement output per program). Otherwise you may perform incoherent analises.

Exercise 8

Edit the "answers.txt" file and write down the following:

a) The command line to run a single instance of "./integers" program in any processing unit
b) The elapsed and %CPU will be the baseline performance of this program
c) The command line to run a single instance of "./floats" program in any processing unit
d) The elapsed and %CPU will be the baseline performance of this program
e) The command line to run a single instance of "./mems" program in any processing unit

f) The elapsed and %CPU will be the baseline performance of this program

If you are using a high performance computer and you see the elapsed time is too short (e.g. less than 10 seconds), modify the correspondent C++ code to increase the number of iterations of the main loop of that code in order to increase the execution time.

Contention in hardware resources

Let's analyze what happens when you run multiple independent instances of the same program in different processing units, in a different core each. But in this experiment no instances can be bound to the same core or processing unit. To do this, use the information collected during the first part of the Lab session to know what are the correct processing unit IDs (e.g. "lstopo" command).

Exercise 9

Edit the "answers.txt" file and write down the following pair of information per every type of program (i.e. "./integers", "./floats", "./mems") to bind the programs as indicated in the previous paragraph:

a) The command line to run the experiment for a given program type
b) Average elapsed and %CPU

Finally, analyze and briefly explain whether there is any impact on average time or average %CPU compared to baseline measurements.

As next step we will bind instances of the same program in the same core, but not in the same processing unit.

Exercise 10

Edit the "answers.txt" file and write down the following pair of points per every experiment:

a) The command line to run the experiment for a given program type
b) Average elapsed and %CPU

Finally, analyze and briefly explain whether there is any impact on average time or average %CPU compared to baseline measurements. Also indicate if you find any behavioral pattern in the "%CPU" value provided by the "top" command.

Finally, execute experiments mixing different types of programs bound to the same core, but not in the same processing unit.

Exercise 11

Edit the "answers.txt" file and write down the following pair of points per every experiment:

a) The command line to run the experiment for a given program type
b) Average elapsed and %CPU

Finally, analyze and briefly explain whether there is any impact on average time or average %CPU compared to baseline measurements. Also indicate if you find any behavioral pattern in the "%CPU" value provided by the "top" command.

Exercise 12

Last but not least, find the required commands/tools to find out the hardware specs of your own computer if you are using a different Operating System, or the Windows image in the computers of the

Lab. Thus, add to the "answers.txt" the commands/tools of the Operating System you have used to get similar hardware data than the previous exercises of this Lab Session.

Upload the Deliverable

To save the changes you can use the tar command as follows:

```
#tar czvf session4.tar.gz answers.txt Makefile *.cc
```

Now go to RACO and upload this recently created file to the corresponding session slot.