Computer configuration for Dummies - (from scratch)

I. Sticco* and F. Cornes†
Departamento de Física, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Pabellón I, Ciudad Universitaria, 1428 Buenos Aires, Argentina.

G.A. Frank‡
Universidad Tecnológica Nacional, Facultad Regional Buenos Aires, Av. Medrano 951, 1179 Buenos Aires, Argentina.
(Dated: February 1, 2018)

This article is intended as a starting point in the art of simulation. We will try to make it easy...

PACS numbers: DuMMy, 1.2.me

I. THE BASIC CONFIGURATION

This is the very first step in computing simulation. Here is a piece of advice: start with a simple, easy to configure, multi-core computer. The minimum requirement for making possible a computer simulation are

(1) A stable operating system.

(2) An internet connection.

(3) The specific software for your hardware.

A. Step 1: install the Linux distro

Since you will only want to drop simulation jobs into *the box* and get back the results, neither nice environment nor strange software will be needed. What you actually need is a stable operating system and a way for managing it through a terminal (*in situ* or not).

Download a small image of the operating system. For example, go to

`https://www.debian.org/CD/netinst/`

and download any suitable image for your computer. If you do not know which is one is the correct image for your computer, just try the `x86_64` compatible image. Most systems are compatible with this architecture. Thus, choose

`debian-9.2.0-amd64-netinst.iso`

In order to install this image in your (new) computer, you will have to make a bootable USB-stick (or *pendrive*). Get a pendrive with at least 300MB available and plug it into the USB port. Then, find out the device name for this port by typing into a terminal window

```
dmesg | grep scsi -A 3
```

This should inform you the device name for your USB. Alternatively, type `lsblk` for a shorter (and more friendly) device name. For example,you may get `sdf1` for this last instruction. This means that your device name is `/dev/sdf` (omit the number "1" in the device name).

You are now able to copy the linux image into your pendrive. Just type

```
sudo dd if=debian-9.2.0-amd64-netinst.iso
of=/dev/sdf bs=4M; sync
```

where the line has been broken for space reasons only (do no break the line instructions in your terminal). The `sudo` command allows you to copy the image avoiding any "permission denied" message. Of course, you will be asked for your personal password. The `dd` command is a low level instruction for copying data block by block. The `bs=4M` switch fixes the block size to 4MBytes. The `sync` command dispatches the data out to the `/dev/sdf` device.

You are now ready for the linux installation! But, before the installation, make sure that the network administrator has already provided you a working connection. Just ask him!

Unplug your pendrive and plug it into the computer where the linux distro will be installed. Turn on the computer and access the Bios at startup by pressing either the `F1`, `F2`, `F12`, `Del` or `Esc` key (it depends on the manufacturer settings). Change the BIOS boot order so the USB device option is listed first. Then, restart your computer.

1. *No graphical installation*

Do not choose the `Graphical Install`, just the simple `Install` option. Since this computer will be used for

* ignaciosticco@gmail.com
† fercornes@gmail.com
‡ guillermo.frank@gmail.com

computer simulations only, no graphical environments will be needed. This will save hard disk space and avoid more complicated configurations.

After you enter the installer, select the right options for the language and regional settings. The installer will download all the needed packages through the internet connection. Allow the installer to configure the network with DHCP automatically. Do not configure any proxy if not needed (just leave it as blank).

2. *Partition manager*

When you are asked for the partition configuration, select the guided partition procedure. You may choose one partition for the installation and another partition for swap. Usually swap size is twice the RAM size. Also, choose the `EXT4` file system.

3. *Software selection*

The `Software selection` menu is a very important step during the configuration. Unselect the `Graphical desktop environment` option since no graphical environment will be used on this machine (although you will be allowed to configure it at any time later). Select the `SSH` server and the `Standard` utilities. The menu should look something like this before you continue.

```
[ ] Graphical desktop environment
[ ] Web server
[ ] Print server
[ ] DNS server
[ ] File server
[ ] Mail server
[ ] SQL data base
[*] SSH server
[ ] Laptop
[*] Standard system utilities
```

If you get a dialogue box asking if services can be restarted automatically during upgrade, choose `yes`.

The last step is to choose a boot loader. Just accept the default `Grub` boot loader.

B. Step 2: boot into the new installation

Recall that at this stage you can access your computer through a `SSH` service. Thus, after booting into the new installation, you may enter the user login and password chosen previously) *in situ* or from a distant keyboard.

Warning: if you have an Nvidia card, or any other graphic card that needs a special driver, Debian will hang after (re)booting. You may surpass the hang up by pressing the "e" key at the Grub screen and typing at the end of the line beginning with "linux": `nomodeset`. That will solve the problem, but not persistently (see below).

Once you logged in, switch to super user mode (root) in order to edit the resources list, update the packages, and install sudo and grant sudo rights to the user. Type: `su` and enter the root password. Then type

```
nano /etc/apt/sources.list
```

and add at the end of each line `contrib non-free`. For example, for any line similar to

```
deb http://http.us.debian.org/debian stable main
```

replace it by

```
deb http://http.us.debian.org/debian stable main
contrib non-free
```

(the line breaking has been done for space reasons only).

This will allow the installation of non-GNU software. After you finish, save and exit (`ctrl+o` + `ctrl+x`). Then type

```
apt-get update
apt-get install sudo
usermod -a -G sudo <username>
```

where `<username>` means the username to which you want to grant sudo access. Logout and Log into your user (yes, logout and login, because otherwise the changes will not take effect!) and run the following commands

```
sudo apt-get update
sudo apt-get upgrade
```

After the upgrade is complete, restart the system by typing `sudo reboot`.

C. Step 3: special software (or firmware)

If you had problems with the graphics card, now is time to solve it definitely. We assume that your system includes an Nvidia graphic card and that you may want to use it in the future for parallel computing (*i.e.* CUDA compatible).

In order to install the Nvidia CUDA package (that is, the CUDA driver, CUDA toolkit and CUDA samples), it is necessary to include the following packages into your system

`firmware-linux`: binary firmware for various drivers in the Linux kernel

`llvm`: the low-level virtual machine package libraries and tools that make it easy to build compilers, optimizers, just-in-time code generators, and many other compiler-related programs.

`clang`: a front-end for the `llvm` compiler.

`build-essential`: necessary to create Debian packages.

`gcc-multilib`: the GNU C compiler with multilib support, that is, with support for the non-default multilib architecture(s).

`install linux-headers`: allows pieces of source code (say, the "headers") to be available for other packages.

The installation of these packages is straight forward. Type: `su` and enter the root password. Then type

```
apt install llvm
apt install clang
apt install firmware-linux
apt install build-essential
apt install gcc-multilib
apt install linux-headers-$(uname -r)
apt build-dep linux
```

where the `$(uname -r)` is a command substitution for the current kernel version. The last command (`build-dep`) re-builds the dependencies for `linux`. Thus, these seven commands sets a "friendly" environment for the installation for the CUDA package.

D. Step 3: configure the computer to restart automatically

Open your computer's BIOS and look for the Power Settings menu. Change the AC Power Recovery (or similar) setting to `On`. If this setting is not available, it means that your computer is not capable to restart automatically after a power off.

II. CUDA INSTALLATION

It is time to improve the computing capabilities through the GPU (Graphic Processing Unit)! Recall that no Nvidia software has been installed yet, and therefore, we assume that a "fresh" installation is required.

Warning: very important! If you suspect that previous Nvidia software was installed before (say, any previous installation that failed, or whatever), make sure that no pieces of Nvidia software are still present.

We further recommend to (strictly) follow the corresponding steps, but do not proceed to the next one if you did not succeed in the current one. Nvidia software may be somehow tricky on "not-supported" Linux distributions (see below).

A. Before the installation

Suppose for a while that you can not remember the graphic card model in our system. Just type `lspci` into the terminal, and you will get the following line (or similar) among others

```
01:00.0 VGA compatible controller:
NVIDIA Corporation GM204
[GeForce GTX 980] (rev a1)
```

(the line breaking corresponds to space reasons only). This means that a Nvidia `GeForce GTX 980` is available, but it says nothing if it is currently in use. You may further type

```
lspci -k | grep -EA3 'VGA|3D|Display'
```

and the more verbose report will appear as follows

```
01:00.0
VGA compatible controller:
NVIDIA Corporation GM204 [GeForce GTX 980] (rev a1)
Subsystem: eVga.com. Corp. GM204 [GeForce GTX 980]
Kernel driver in use: nouveau
Kernel modules: nouveau, nvidia_drm, nvidia
```

The report confirms that `GeForce GTX 980` is available, although `Nouveau` is currently the driver in use. This is right, since `Nouveau` is the default driver shipping with Debian. The Nvidia people, however, may not feel really comfortable with `Nouveau` playing around. Thus, switch the computer mode to `runlevel 3` (you are probably running the system to `runlevel 5`) in order to turn off the current "display manager". In other words, you will somehow switch to "text mode" only. Type into the terminal: `sudo /sbin/init 3`

You are right in the way to install the CUDA package. But before proceeding, it's time to make a few "just in case" actions. These are "recommended" actions from Nvidia and other people. ¡They seem to be the result of

many try-and-failure experiences!

First, install some extra packages. Do not worry if some of them were previously installed because in that case they will be passed over.

```
sudo apt-get install freeglut3-dev build-essential
libx11-dev libxmu-dev libxi-dev libgl1-mesa-glx
libglu1-mesa libglu1-mesa-dev
```

Second, check if this card is CUDA capable at `http://developer.nvidia.com/cuda-gpus`. The current card should be listed as CUDA capable (you may click on the listed name for more information).

Third, verify (once more!) the current Linux version, the `gcc` version, and the current headers and development packages. Although this is quite boring, take a few minutes to ensure that you are in the right way. Type the following commands

```
uname -m && cat /etc/*release
gcc --version
uname -r
```

The first command will return the several lines, but the important ones are

```
x86_64
PRETTY_NAME="Debian GNU/Linux 9 (stretch)"
```

meaning that we are running a 64-bits system (Debian 9, in this case). The second command will return

```
gcc (Debian 6.3.0-18) 6.3.0 20170516
```

that is, the version 6.3.0 for `gcc`. The third command will return something like `4.9.0-4-amd64` (the version of the kernel headers).

Well, the pre-installation is actually complete! Good job till now!

B. The installation

You have now enough information for the installation. Download the corresponding CUDA package from

```
http://developer.nvidia.com/cuda-downloads
```

and choose the `Linux` option, the `x86_64` architecture (since you have already verified in Section II A that this is the right one), the `Ubuntu` distribution (since it is based on Debian), the 16.04 version (since it is currently an LTS version, but most important, you will find at `/etc/debian_version` that the Ubuntu 16.04 (xenial) is based on Debian (stretch)), and the `runfile` installer type.

Warning: Perhaps, at this stage, you might be operating the system from a remote terminal (via `ssh`). If this the case, you can download the runfile as follows

```
wget https://developer.nvidia.com/compute/
cuda/9.1/Prod/local_installers/
cuda_9.1.85_387.26_linux
```

(the line breaking corresponds to space reasons only). The above url location can be obtained by right-clicking on the download button and selecting the "save link" option.

Type the following commands to change the name of the file and the corresponding executable permissions

```
mv cuda_9.1.85_387.26_linux
   cuda_9.1.85_387.26_linux.run
chmod +x cuda_9.1.85_387.26_linux.run
```

(the line breaking corresponds to space reasons only). You may now execute the file

```
sudo ./cuda_9.1.85_387.26_linux.run
```

The text-based installer will first show a looooong disclaimer (press the `space` key to continue). Then you will have to answer some questions, as follows

```
---------------------------------------
Do you accept the previously read EULA?
accept/decline/quit: accept

You are attempting to install on an unsupported
configuration. Do you wish to continue?
(y)es/(n)o [ default is no ]: y

Install NVIDIA Accelerated Graphics Driver
for Linux-x86_64 387.26?
(y)es/(n)o/(q)uit: y

Do you want to install the OpenGL libraries?
(y)es/(n)o/(q)uit [ default is yes ]: y

Do you want to run nvidia-xconfig?
This will update the system X configuration file
so that the NVIDIA X driver is used....
(y)es/(n)o/(q)uit [ default is no ]: y

Install the CUDA 9.1 Toolkit?
(y)es/(n)o/(q)uit: y

Enter Toolkit Location
 [ default is /usr/local/cuda-9.1 ]:

Do you want to install a symbolic link
at /usr/local/cuda?
(y)es/(n)o/(q)uit: y

Install the CUDA 9.1 Samples?
(y)es/(n)o/(q)uit: y

Enter CUDA Samples Location
 [ default is /home/me ]: /home/me/mydir
---------------------------------------
```

Notice that we accepted all the options. This seems to be safe enough! After proceeding, the installer reports the following

```
Installing the NVIDIA display driver...
A system reboot is required to continue installation.
Please reboot then run the installer again.
An attmept has been made to disable Nouveau.
If this message persists after reboot, please see
the display driver log file at
/var/log/nvidia-installer.log for more information.

===========
= Summary =
===========

Driver:   Reboot required to continue
Toolkit:  Installation skipped
Samples:  Installation skipped

To uninstall the NVIDIA Driver,run nvidia-uninstall

Logfile is /tmp/cuda_install_1134.log
```

This report looks surprisingly neat `:)` . It says that our attempt to disable `Nouveau` was not completely effective, and thus, the installer had to further proceed with additional actions. These actions require a system reboot before installing the CUDA toolkit and the samples.

Please, do not get annoyed because of this report. We experienced more annoying messages in our first try! Thus, it is very important to receive the above report. If your report includes additional messages like "missing recommended library" or "driver not selected" or the annoying "***WARNING: Incomplete installation!", it means that something is going (really) wrong, and you should step back to find the failure.

Type `sudo reboot` and after login type

```
sudo /sbin/init 3
sudo ./cuda_9.1.85_387.26_linux.run
```

and repeat the previous steps. The installation should now be successful. The report, however, still remind you to include the corresponding paths. The paths are necessary for locating files along the system directory structure. Just open the `.bashrc` file located in you home directory (*i.e.* `/home/me`) and add the following lines at the end of the file

```
export CUDA_HOME=/usr/local/cuda-9.1
export LD_LIBRARY_PATH=/usr/local/cuda-9.1/lib64
export PATH=/usr/local/cuda-9.1/bin
```

save and exit. You may also type into the terminal the command `source /home/me/.bashrc` for the changes to take place. You are done!

C. Testing CUDA

We now proceed with a few tests (suggested by the Nvidia people). First, reboot your system (that is, `sudo reboot`). After login (as user), type into the terminal `nvcc -V`. This is the CUDA compiler and you are requesting some kind of proof that it is alive `:)`. You should receive

```
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2017 NVIDIA Corporation
Built on Fri_Nov__3_21:07:56_CDT_2017
Cuda compilation tools, release 9.1, V9.1.85
```

meaning that the compiler is in good health.

As a second check, type `nvidia-smi` to make sure that the CUDA driver is communicating correctly with the system. If a similar report as the one below appears, it means that you are in the right way

```
Wed Jan 31 23:07:09 2018
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 387.26                 Driver Version: 387.26                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce GTX 980     Off  | 00000000:01:00.0 Off |                  N/A |
|  0%   40C    P0    37W / 185W |      0MiB /  4036MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

A last check on the health of the CUDA driver can be done by writing

```
cat /proc/driver/nvidia/version
```

and you should get the following nice report

```
NVRM version: NVIDIA UNIX x86_64
Kernel Module  387.26
Thu Nov  2 21:20:16 PDT 2017
GCC version:  gcc version 6.3.0 20170516
(Debian 6.3.0-18)
```

So, the CUDA driver seems to be correctly installed. However, a couple of examples needs to be run, in order to relax after this (somehow) stressing installation. Type the lowing commands (from `/home/me/mydir`) in order to compile the examples bundled in the CUDA package.

```
cd NVIDIA_CUDA-9.1_Samples
make
```

and wait.... until *all* the examples are compiled. Notice that we did not install the graphical desktop environment (see Section I A 3), and consequently, it will not be possible to execute those examples requiring a graphical environment. Furthermore, some warning messages might appear during the examples compilation due to missing graphical libraries.

After the compilation procedure is finished, proceed by changing directory

```
cd /home/me/software/
NVIDIA_CUDA-9.1_Samples/bin/
x86_64/linux/release
```

run the text mode example `./deviceQuery`. You should receive something like

```
----------------------------------------
./deviceQuery Starting...

 CUDA Device Query (Runtime API) version
 (CUDART static linking)

Detected 1 CUDA Capable device(s)

Device 0: "GeForce GTX 980"
  CUDA Driver Version /
  Runtime Version          9.1 / 9.1
  ...
  ...


deviceQuery, CUDA Driver = CUDART,
CUDA Driver Version = 9.1, CUDA Runtime
Version = 9.1, NumDevs = 1
Result = PASS
----------------------------------------
```

These are the relevant pieces of the report. The final result is `PASS`, so the example, ran successfully. Also, try the example `./bandwidthTest`. The report should read

```
----------------------------------------
[CUDA Bandwidth Test] - Starting...
Running on...

 Device 0: GeForce GTX 980
 Quick Mode

 Host to Device Bandwidth, 1 Device(s)
 PINNED Memory Transfers
   Transfer Size (Bytes) Bandwidth(MB/s)
   33554432 12873.2

 Device to Host Bandwidth, 1 Device(s)
 PINNED Memory Transfers
   Transfer Size (Bytes) Bandwidth(MB/s)
   33554432 12469.6

 Device to Device Bandwidth, 1 Device(s)
 PINNED Memory Transfers
   Transfer Size (Bytes) Bandwidth(MB/s)
   33554432 163707.9

Result = PASS

NOTE: The CUDA Samples are not meant
for performance measurements. Results may
vary when GPU Boost is enabled.
----------------------------------------
```

You get another `PASS` result. So, congratulations, you are done! Enjoy!.

III. FINE TUNING OUR SYSTEM

A. e-mail notifications

`msmtp` is a small program that communicates via the SMTP protocol to the Gmail server. Install this program in order to receive e-mail notifications from your system. Follow the instructions outlined in *Trick for email notifications*.

1. *Step 1: install msmtp*

You may download the latest version `msmtp` from

```
https://sourceforge.net/projects/msmtp/files/msmtp/
```

You will get something like `msmtp-x.x.x.tar.xz`. Don't worry because it is a small file.

Drop the file into your home directory in the cluster. You will probably need a `ssh` connection. Use the command `scp` to copy the file into the cluster. From now on we will assume that your home directory is `/home/you` and that the expression `$HOME` is an alias for your home directory (technically, `$HOME` is an environment variable).

To copy the file into the cluster, type from the terminal

```
scp ./msmtp-x.x.x.tar.xz you@cluster:/home/you
```

where `you` is your username and `cluster` is the name of the cluster (obvious). After that, log into your user directory in the cluster (say, `ssh you@cluster`) and uncompress the source file. For example,

```
tar -xf msmtp-x.x.x.tar.xz
```

You will notice that a new directory has been create called `msmtp-x.x.x`. It contains all the source files for building the binaries. Now follow the steps

```
cd msmtp-x.x.x
./configure --prefix=$HOME
make install
```

The `--prefix` option ensures that everything will be inside the limits of the `/home/you` directory.

The installation is almost done! You only need to create a configuration file. Do this by with any text editor (such a `gedit`). Your configuration file should have the name `.msmtprc` (the dot is mandatory to "hide" the file) and it should be located in `/home/you` (or `$HOME` as you wish). For security reasons, you better assign only user read/write permission to the file. Type

```
chmod 600 $HOME/.msmtprc
```

The configuration file is empty till now. Open the file (with a text editor like `gedit`) and include the following information

```
account default
host smtp.gmail.com
port 587
from mycluster@gmail.com
tls  on
tls_starttls on
tls_certcheck off
auth on
user mycluster@gmail.com
password xxxxxx
```

Save and exit. The `xxxxxx` is our secret password. The account `mycluster@gmail.com` is your recently created mail account for the cluster.

Configuration is finished, congratulations!

2. *Step 2: test it*

The installation process is over. You can make a simple test to ensure that everything is working fine. Type the following and see if a new mail appears in the inbox of webmail

```
echo "Subject: hello" | msmtp mymail@gmail.com
```

where `mymail@gmail.com` is assumed to be your personal account (not the recently opened account `mycluster@gmail`). Thus, if you now check your personal account, you will find an incoming mail from `mycluster@gmail` with the subject `hello`. The body of the mail should be empty.

The above instruction is a pipe, that is, `echo` passes the `"Subject: hello"` to `msmtp`. The `|` character chains both instructions. Thus, `msmtp` knows that the string `hello` is the subject needed to complete the mail to `mymail@gmail.com`.

3. *Step 3: automate it*

This last step makes the use of `msmtp` more comfortable, although it is not really necessary. You can create a bash file with the following content

```
#!/bin/bash

args=("$@")

if (($# > 1)); then
  i=1
  while [ $i -lt $# ]
  do
     message+=${args[i]}
     message+=" "
     i=$[i+1]
  done

  if [[ $1 != @* ]]; then
  subject="Subject: "
  subject+="${message}"
  echo -e "${subject}" |
  /home/you/msmtp-1.6.3/src/msmtp "${args[0]}"
  fi

fi
```

(caution: the line starting with `echo` has been broken into two lines for space reasons only)

The `args` variable contains the all the arguments passed to the bash file. The first argument is `args[0]` and corresponds to the mail address where the message should be delivered. The rest of the arguments are collected into a single string (called `message`) by the `while` loop. Finally, `subject` joins the word `Subject` to the message.

This bash file is really useful, since we only need to type the address and the message. For example, if the bash file name is `myfile`, then the instruction

```
myfile  mymail@gmail.com My first message
```

will deliver the mail with the message `My first message`.

Enjoy!

B. Re-booting notifications

We now want to allow the computer to notify whenever a re-booting occurred (due to power off-on, an administrator re-booting, etc.). We assume that a bash file `notify` already exists, similar to the notification file shown in Section III A 3. We further assume that the file is located in

```
/home/me/notify
```

(make sure that the chmod -x permissions are set). Then type

```
crontab -e
```

A text file will appear on screen. Just add at the end of the file the following

```
@reboot sleep 60 && /home/me/
notify me@mymail.com rebooted
```

(caution: the line has been broken into two lines for space reasons only)

Save and exit (if the text editor is `nano`, just press `ctrl-o` and `ctrl-x`)

This line tells de program `cron` (the piece of software in Debian 9 that controls the tasks schedule) to make the following tasks after reboot (or restart, or power on, as you like)

1. sleep 60 (in order to wait enough time until the computer is completely ready)
2. use the script `notify` to send an email to `me@mymail.com` with the word "rebooted"

That is all!!! You can test it by typing `sudo reboot` or power off/power on the computer. Enjoy!