

Amber 24 and AmberTools 24 Install on ML Linux (MC-23.3) - <https://ambermd.org/InstDebian.php>

Note that ML Linux is Debian (**Bookworm**) based.

Before you install the software:

Note that Synaptic Package Manager is available under Applications, System (install Gdebi from MX Package Installer - later).

As far as possible, install software from the MC Package Installer (always choose to also install the Recommended packages) or the Synaptic App. If not possible, download the required Deb (and dependencies) file from - <https://www.debian.org/distrib/>

Another option is to use the normal Ubuntu way:

```
$ sudo apt-get install appname
```

See python version:

```
$ python3 -V (mine gave version 3.11)
```

It seems that Amber 24 (page 25) requires the latest version (Python **3.11** was installed)

Note that **Amber 24 supports Cuda up to version 11.4** (Page 21 of the manual). (cuda is installed later)

However, the GNU version (according to “/home/gert/amber24_src/build/CmakeFiles/CmakeOutput.log” is version 11.3. (I tried Cuda version 11.4 first, but it gave an error: “Error: Incompatible CUDA and GNU versions 11.3.0”)

See table on which version of gcc and c++ works with this cuda version (**11.3**):
<https://stackoverflow.com/questions/6622454/cuda-incompatible-with-my-gcc-version>

According to the table, this requires gcc-10 (g++ and gfortran as well). Note that the default version that was installed is version 12.2. So you have to use “update-alternatives”.

Install gcc-10 and then g++-10 (and their dependency files) from https://debian.pkgs.org/11/debian-main-amd64/g++-10_10.2.1-6_amd64.deb.html using the deb files and Gdebi.

Install gfortran-10 from <https://packages.debian.org/bullseye/amd64/gfortran-10/download> and <https://packages.debian.org/bullseye/amd64/libgfortran-10-dev/download>

```
$ sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-10 30  
$ sudo update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-10 30  
$ sudo update-alternatives --install /usr/bin/gfortran gfortran /usr/bin/gfortran-10 30
```

```
$ gcc --version  
$ g++ --version  
$ gfortran --version
```

See if your PC has a GPU card:

```
$ sudo lshw -numeric -C display
```

Then make sure your GPU card is supported in Amber - <https://ambermd.org/gpus16/index.htm>

Use the MX Package installer and install the following Nvidia options (cuda is installed later):

```
sudo apt-get install nvidia-alternative nvidia-detect nvidia-driver nvidia-driver-bin nvidia-installer-  
cleanup nvidia-kernel-common nvidia-kernel-dkms nvidia-kernel-source nvidia-kernel-support gpustat
```

(This command will also install the correct Nvidia driver. Make sure you DON'T install Nvidia-toolkit at all! Version 11.8 will be installed and only version 11.4 is supported)

(To restore open source drivers later use (if required): \$ sudo ddm-mx -p nvidia)

Note that the “Nvidia X Server Settings” App is installed after successful installation of the Nvidia Driver.

The following should be installed from the MX Package Installer (click on “Also install recommended packages”):

```
sudo apt-get install tcsh cmake flex flexc++ bison bisonc++ wget pwget python3-wget wget2 openssh-  
client intel-mkl intel-mkl-cluster patch bc fftw2 python3-pyfftw sfftw2 gedit gedit-common gdebi  
gnome-system-monitor
```

gnome-system-monitor is used to see how many CPUs are used during a calculation – System Monitor)

```
sudo apt-get install python3 python3-libapparmor python3-libdnf python3-matplotlib python3-numpy  
python3-packaging python3-scipy python3-setuptools python3-setuptools-git python3-venv python-is-  
python3 python-dev-is-python3
```

Install with **Synaptic Package Manager:**

```
libbz2, libbz2-dev, libzip-dev, xorg-dev
```

Install VMD.

Install an older (or multiple) version(s) of Cuda: (Cuda 11.3 required)

How to find the Cuda directory:

```
$ locate cuda
```

<https://stackoverflow.com/questions/40517083/multiple-cuda-versions-on-machine-nvcc-v-confusion>

Cuda is normally installed under the `/usr/local/` folder

To install other versions (such as Cuda 11.3) go to - <https://developer.nvidia.com/cuda-toolkit-archive>

Download the latest Ubuntu version (runfile).

Then install them using the instruction for the runfile option. It works well (don't reinstall the Nvidia Driver!).

Note that it will give an error if the gcc (and G++ versions) does not match.

Message after 11.3 was installed:

Please make sure that

- *PATH includes /usr/local/cuda-11.3/bin*
- *LD_LIBRARY_PATH includes /usr/local/cuda-11.3/lib64, or, add /usr/local/cuda-11.3/lib64 to /etc/ld.so.conf and run ldconfig as root*

Then add the following to the `~/.bashrc` file (change for the required Cuda version):

<https://stackoverflow.com/questions/19980412/how-to-let-cmake-find-cuda>

```
$ sudo gedit ~/.bashrc
```

```
export CUDA_HOME=/usr/local/cuda-11.3
export PATH="/usr/local/cuda-11.3/bin:$PATH"
export LD_LIBRARY_PATH="/usr/local/cuda-11.3/lib64:$LD_LIBRARY_PATH"
```

See <https://www.cs.colostate.edu/~info/cuda-faq.html>

Open and close the terminal so that the changes in the bashrc file is read.

```
$ nvcc -V          (should give 11.3)
```

How to find the Cuda directory:

```
$ locate cuda  (does not work??)
```

The "NVIDIA X Server Settings" should be installed after this (Reboot the computer if you cannot see the GPUs).

See what driver is installed:

```
$ sudo lshw -c video | grep 'configuration'
```

Then open Nvidia X Server Settings app to see if the GPU's are seen?

```
$ sudo apt install mesa-utils
```

```
$ glxgears
```

Another option - <https://www.chaos.com/vray/benchmark?AFFILIATE=120043>

And see if the GPU show variable activity?

Amber22 and AmberTools22:

Download the compressed Amber and AmberTools files and extract them to your home directory.

Follow the instructions in the Amber22 manual and <https://ambermd.org/InstUbuntu.php>

According to the latter, there are only 2 requirements: Cmake and the latest version of Python. Both were already installed (above).

Install the serial version of Amber24 first.

```
$ cd amber24_src/build
```

Make a back-up of the “run_cmake” file so that you can use it later if required:

```
$ cp run_cmake run_cmake_old (also make run_cmake_mpi and run_cmake_nccl )
```

I changed the following in the “run_cmake” script:

```
$ gedit run_cmake
```

```
# Assume this is Linux:
```

```
cmake $AMBER_PREFIX/amber24_src \  
-DCMAKE_INSTALL_PREFIX=$AMBER_PREFIX/amber24 \  
-DCOMPILER=GNU -DCUDA_TOOLKIT_ROOT_DIR=/usr/local/cuda-11.3 \  
-DMPI=FALSE -DCUDA=TRUE -DINSTALL_TESTS=TRUE \  
-DDOWNLOAD_MINICONDA=FALSE \  
2>&1 | tee cmake.log
```

(use `$./clean_build` is you want to restart a clean build).

The serial installation and tests worked well using the commands from the Amber22 Manual (0 tests experienced errors).

The “./run_cmake” command then gave 2 or so more python errors with missing python packages, that I could all install using Synaptic Package Manager.

The “make install” install command then also worked without errors.

\$ CC=gcc CXX=g++ | ./run_cmake (Duy used this command to specify the path for fortran
Using gcc and g++)

\$ gedit ~/.bashrc

Add “source /home/gert/amber22/amber.sh” right at the bottom and save the file.

MPI and Cuda install:

Read the information provided in the manual (Par 2.2.1 and 2.2.2).

For QM/MM: “Depending on the MPI implementation, this can, however, fail. In our experience, MPICH and MVAPICH work well *while OpenMPI does not work.*” (page 166 of the manual)

Mpich installation is described below.

To compile the parallel version of Amber:

\$ cd amber24_src/AmberTools/src/

[note the command in the manual (cd \$AMBERHOME/AmberTools/src) takes you to the wrong directory!]

Make the following changes before you run “./configure_mpich gnu”:

Otherwise it will give some errors:

“You must download MPICH and extract it here (in *amber24_src/AmberTools/src/*)
(for example, type 'tar xvzf mpich-3.3.2.tar.gz')
See <http://www.mpich.org> for more info;
Then, re-run this script.”

(Download the First mpich-4.1 (stable release) at the top)

“configure: error: The Fortran compiler gfortran does not accept programs that call the same routine with arguments of different types without the option -fallow-argument-mismatch. Rerun configure with FFLAGS=-fallow-argument-mismatch and FCFLAGS=-fallow-argument-mismatch
MPICHls configure failed, returning 1”

Open the “configure_mpich” script with a text editor and change the “gnu” part as follows:

\$ gedit configure_mpich

```
##### gcc #####  
gnu)  
cc=gcc  
cflags=-fPIC  
cplusplus=g++  
ocflags=-O3  
fc=gfortran
```

```
fcflags=-fallow-argument-mismatch
fflags=-fallow-argument-mismatch
```

```
;;
```

(Note use small letters and not “*FFLAGS*” and “*FCFLAGS*” as the system suggested. Also, the order should be reversed as above.)

Then run the command:

```
$ ./configure_mpich gnu
```

(This takes a very long time. Perhaps 3 or more hours. 2 hour and 20 minutes on the new Power PC). (Twice I got a permission error – used “`sudo ./configure_mpich gnu`” and it then worked)

Open a new Terminal tab:

Go into the “`amber22_src/build/`” directory.

```
$ cd ~
```

```
$ cd amber24_src/build
```

```
$ gedit run_cmake_mpi
```

This one worked:

```
# Assume this is Linux:
```

```
cmake $AMBER_PREFIX/amber24_src \  
-DCMAKE_INSTALL_PREFIX=$AMBER_PREFIX/amber24 \  
-DCOMPILER=GNU -DCUDA_TOOLKIT_ROOT_DIR=/usr/local/cuda-11.3 \  
-DMPI=TRUE -DCUDA=TRUE -DINSTALL_TESTS=TRUE \  
-DDOWNLOAD_MINICONDA=FALSE \  
2>&1 | tee cmake.log
```

Then follow the instructions from the manual. Last, do the test.

```
$ ./run_cmake_mpi
```

```
$ make install
```

```
$ cd $AMBERHOME
```

```
$ export DO_PARALLEL="mpirun -np 30"
```

```
$ make test.parallel
```

This test gave an error: “Error: Could not import Amber Python modules! Probably your Amber Python environment was not set up correctly.”

This solution worked - <http://archive.ambermd.org/202202/0036.html> :

```
$ gedit ~/.bashrc
```

Add the following lines:

```
export PYTHONPATH=$AMBERHOME/lib/python3.11/site-packages
```

```
export PYTHONPATH=$AMBERHOME/lib/python3.11/site-packages:$PYTHONPATH
```

Summary of AmberTools parallel tests:

961 file comparisons passed

16 file comparisons failed (2 of which can be ignored)

17 tests experienced errors

21.6.5. Installation and Testing - GPU Accelerated PMEMD (page 439)

Building pmemd.cuda.MPI with NCCL support (page 440) and with Quick.

Download the local installer for Ubuntu 20.04 (deb file) via the website

(<https://developer.nvidia.com/nccl/nccl-download> then choose older versions (11.3) at

<https://developer.nvidia.com/nccl/nccl-legacy-downloads>) and install the DEB file (latest local

Ubuntu) with Gdebi (from my home directory). (you should use the Google Authenticator on your Phone when logging into the above web site).

Note I had to do the command at the bottom as well (otherwise it gave NCCL errors):

```
$ sudo apt install libnccl2=2.8.4-1+cuda11.2 libnccl-dev=2.8.4-1+cuda11.2
```

Gave an error:

```
E: Unable to locate package libnccl2
```

```
E: Unable to locate package libnccl-dev
```

These deb files were in the NCCL folder (`/var/nccl-local-repo-ubuntu2004-2.9.9-cuda11.3/`) Use Gdebi to install them.

Edit the cmake file:

```
$ cd amber24_src/build
```

Add the path in the the bashrc file:

```
$ gedit ~/.bashrc
```

Add the following line at the end:

```
export NCCL_HOME=/var/nccl-local-repo-ubuntu2004-2.9.9-cuda11.3/
```

Then change the “run_cmake” file again (add :”-DNCCL=TRUE”):

Save the file as: `run_cmake_nccl`

```
$ source ~/.bashrc
```

This one worked:

```
# Assume this is Linux:
```

```
cmake $AMBER_PREFIX/amber22_src \  
-DCMAKE_INSTALL_PREFIX=$AMBER_PREFIX/amber24 -DNCCL=TRUE \  
-DCOMPILER=GNU -DBUILD_QUICK=TRUE \  
-DMPI=TRUE -DCUDA=TRUE -DINSTALL_TESTS=TRUE \  
-DDOWNLOAD_MINICONDA=FALSE \  
2>&1 | tee cmake.log
```

Then execute the file:

```
$ ./run_cmake_nccl
```

Error: Could NOT find NCCL (missing: NCCL_INCLUDE_DIR NCCL_LIBRARY)

```
$ make install
```

GPU Benchmark tests: - <https://www.exxactcorp.com/blog/Molecular-Dynamics/RTX3090-Benchmarks-for-HPC-AMBER22-A100-vs-RTX3080-vs-RTX3070-vs-RTX6000>

Make Amber available to other users:

Log into the account of the other user(s):

Go to your `/home/user/` account, and do the following (change username as required):

```
$ sudo chmod 755 /home/gert/amber24/amber.sh
```



```
$ sudo chmod 755 /home
$ sudo chmod 755 /home/gert
$ sudo chmod 775 /home/gert/amber24
```

That will allow other users to execute the amber.sh script.

Also ensure that you add the Amber line (<source /home/gert/amber24/amber.sh>) in the `~/.bashrc` file for the other users as well.

Close and open the Terminal, so that the changes in the “bashrc” file kick in.

Test if amber is working:

```
$ tleap
```

(Should give you some positive reaction).

(this worked for Amber22 on MX Linux 23.1. Did not work for Amber24 on MX Linux 23.3?)

Install later version of Python (3.11):

<https://techviewleo.com/how-to-install-python-3-on-debian/>

(Also see addition document!)

If MX Package Installer is used to install Nvidia-Toolkit:

Next make sure were Cuda Toolkit was installed:

```
$ dpkg-query -L nvidia-cuda-toolkit      (/usr/lib/cuda, /usr/lib/cuda/bin, /usr/lib/nvidia-cuda-toolkit/,
/usr/lib/nvidia-cuda-toolkit/bin, last one has the nvcc executable)
```

```
$ sudo gedit ~/.bashrc
```

```
export CUDA_HOME=/usr/lib/nvidia-cuda-toolkit/bin/
```

Install Nvidia HPC SDK - <https://developer.nvidia.com/hpc-sdk> The earlier versions is at - <https://developer.nvidia.com/nvidia-hpc-sdk-releases> and 11,2 at <https://developer.nvidia.com/nvidia-hpc-sdk-213-downloads>

I used the Linux x86_64 Tarball option. I used sudo for the last command.

Errors with solutions:

amber22_src/src/pmemd/src/xray/cuda/src/xray/BulkMaskGPU.cu". CMake Error at
pmemd_xray_cuda_generated_BulkMaskGPU.cu.o.RELEASE.cmake:278 (message): Error generating
file.

<http://archive.ambermd.org/202208/0112.html> (solution works).