Amber 22 and AmberTools 22 Install on Ubuntu 22 - https://ambermd.org/InstUbuntu.php

Before you install the software:

Install Synaptic Package Manager and Gdebi from the Ubuntu Software app.

$ sudo apt update && sudo apt install build-essential

Install Ubuntu server:

$ sudo apt-get install ubuntu-server

Read the Amber 22 manual to see what version of Cuda is required? Note that Amber 22 supports Cuda up to version 11.2 (Page 24 of the manual).

Before you can install Cuda, you should ensure the correct gcc and g++ software are installed. See table on which version of gcc and c++ works with you cuda version (required Cuda version is 11.2):
https://stackoverflow.com/questions/6622454/cuda-incompatible-with-my-gcc-version

According to the table, this requires gcc-10 and g++-10

See what version is installed (default is version 12):

$ gcc --version
$ g++ --version

How to install an older version of gcc (when you have never versions):
https://askubuntu.com/questions/950686/how-can-i-use-two-instances-of-gfortran

$ sudo apt-get install gcc-10 g++-10

$ sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-10 20
$ sudo update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-10 20

(higher number has higher priority)

To see if these versions are installed:

$ gcc --version
$ g++ --version

To change the order:
$ sudo update-alternatives --config gcc

(Both should be version 10 now)

Make sure you have the same gfortran version:
$ gfortran --version

Download the deb file (if required) -
https://www.ubuntuupdates.org/package/core/focal/universe/base/gfortran-8

$ sudo apt-get install gfortran-10
$ sudo update-alternatives --install /usr/bin/gfortran gfortran /usr/bin/gfortran-10 20
$ gfortran --version

To change the order:
$ sudo update-alternatives --config gfortran

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

If you have a Nvidia card, record the details and find the correct version of the recommended the driver from https://www.nvidia.com/Download/index.aspx?lang=en-us (don't download it). In my case (Nvidia Quatro RTXA6000) the X64 version 525 was recommended.

Install this driver from the "Software & Updates" utility (choose "Additional Drivers").

Alternative, use the instructions from https://docs.nvidia.com/datacenter/tesla/tesla-installation-notes/index.html

The "NVIDIA X Server Settings" should be installed after this.

See what driver is installed:

$ sudo lshw -c video | grep 'configuration'

$ cat /proc/driver/nvidia/version

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

Best way to **uninstall** Nvidia drivers (this is required if you cannot change the Drivers using "Software & Updates"): https://fedingo.com/how-to-uninstall-nvidia-drivers-in-ubuntu/

Install GPU test app :

$ sudo apt install mesa-utils

$ glxgears

And see if the GPU show variable activity?

Another option for Ubuntu is https://flathub.org/apps/details/com.basemark.BasemarkGPU

Install an older (or multiple) version(s) of 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 11.2) go to - https://developer.nvidia.com/cuda-toolkit-archive

Then install Cuda 11.2 using the instruction for the runfile option. It works well (don't reinstall the Nvidia Driver!).

Choose to continue, but don't override the Nvidia Driver that was previously installed.

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

Message after 11.2 was installed:

Please make sure that
- *PATH includes /usr/local/cuda-11.2/bin*
- *LD_LIBRARY_PATH includes /usr/local/cuda-11.2/lib64, or, add /usr/local/cuda-11.2/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):

$ sudo gedit ~/.bashrc

```
export PATH="/usr/local/cuda-11.2/bin:$PATH"
export LD_LIBRARY_PATH="/usr/local/cuda-11.2/lib64:$LD_LIBRARY_PATH"
```

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

$ nvcc -V (should give 11.2)

Install Yaksa:

Download the RMP file from https://rpmfind.net/linux/RPM/fedora/devel/rawhide/x86_64/y/yaksa-0.2-3.fc37.x86_64.html

$ wget https://rpmfind.net/linux/fedora/linux/development/rawhide/Everything/x86_64/os/Packages/y/yaksa-0.2-3.fc37.x86_64.rpm

Install Alien (software to convert a RPM file to a Deb file) - https://phoenixnap.com/kb/install-rpm-packages-on-ubuntu:

$ sudo apt-get install alien

$ sudo alien packagename.rpm In this case – yaksa-0.2-3.fc37.x86_64.rpm

$ sudo dpkg -i packagename.deb (jy kan ook direct die RPM file installeer).

See python version (was version 3.10):

$ python3 -V

If required, see what is the latest version and then install it:

$ sudo apt install python3.x -y

Install Pymol from the Ubuntu Software app.

Install VMD. (See notes)

Amber22 and AmberTools22:

Install the following from the synaptic Package manager:
Flex, Flexc++, Bison, Bison++, Bisonc++, libboost-all-dev, libbz2, zlib1g, zlib1g-dev, intel-mkl, intel-mkl-cluster (dev, computational-dev etc), libopenblas-dev, libtool, OpenSSL, libssl-dev, python3-scipy, python3-matplotlib, OpenSSH-server, python3-setuptools

Also: $ sudo apt-get install flex bison

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

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

According to the latter, there are only 3 requirements: Cuda 11.2 (and the required gcc, g++ and gfrotran versionns) Cmake and the latest version of Python. Python 3.10 (and 3.7 for OpenPBS) was already installed (above and before).

Cmake:

Install "cmake" from the prescribed website (https://cmake.org/download/). Choose the Unix/Linux source code.

Just download the CMake zip file in your Downloads directory and extract it there. Then in your terminal, go into this directory and use:

$ ./bootstrap && make && sudo make install

To see if it is correctly installed:

$ make --version

Follow the Amber22 Manual to install the software:

Go to the /home/ directory in your terminal.

$ ls (you should see the 2 Amber source files)

The serial installation and tests worked well using the commands from the Amber22 Manual.

I use the default “run_cmake” file (cd amber22_src/build).

Install the serial version of Amber22 first. During the “./run_cmake” step there were no error messages:

The “make install” step gave an error at 26%.

The “make install” step then worked.

Add the following to the “bashrc” file (at the end): source /opt/amber22/amber.sh

$ sudo gedit ~/.bashrc

Make a copy of the cmake file (cd amber22_src/build):

$ cp run_cmake run_cmake_old

Change the following in the “run_cmake” file (in the amber22_src/build directory) under the **Linux** option:
-DDOWNLOAD_MINICONDA=FALSE (since we have the correct Cuda version installed)

Install the serial version of Amber22 first. During the “./run_cmake” step there were no error messages:

Add the following to the “bashrc” file (at the end): source /home/gert/amber22/amber.sh

$ sudo gedit ~/.bashrc

Serial tests gave no errors.

How to enable other users on the server to execute the amber.sh script:

Go to you /home/kruger/amber22 directory, and do the following:

$ sudo chmod 755 amber.sh

Parallel or MPI 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 is installed during the "./configure_mpich gnu" step (below).

$ cd amber22_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 Amber22_src/AmberTools/src/)*
(for example, type 'tar xvfz mpich-3.3.2.tar.gz')
See http://www.mpich.org for more info;
Then, re-run this script."

(I use the top MPICH stable release download – not the Ubuntu one further down)

The "./configure_mpich gnu" step give more errors:

"*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:

```
##################### 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:

$ sudo ./configure_mpich gnu

(This takes a very long time. Perhaps 35 minutes on the new Power PC).

Then go to the "amber22_src/build" directory:
$ cd amber22_src/build

Copy the "run_cmake" file to create "run_cmake_mpi". Then change "-DMPI=TRUE" as the manual describes (under the "# Assume this is Linux:" section). Keep "-DDOWNLOAD_MINICONDA=FALSE".

The "./run_cmake_mpi" step and "make install" steps then worked.

I used a parallel test with 24 CPUs and all CPUs ran at about 100% - use "System Monitor". Few errors:

$ cd $AMBERHOME
$ export DO_PARALLEL="mpirun -np 30"
$ make test.parallel

Summary of AmberTools parallel tests:

471 file comparisons passed
8 file comparisons failed (1 of which can be ignored)
8 tests experienced errors

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

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

I downloaded the Network Installer for Ubuntu20.04 from the website (https://developer.nvidia.com/nccl/nccl-download) and click on the Cuda 11 version! Choose the "Installation instructions" at the bottom - https://docs.nvidia.com/deeplearning/nccl/install-guide/index.html

You will have to register for an account and then download the Google Authenticator App for when yu log in.

The following command worked to download the Cuda Keys Deb file: (from my Downloads directory).

$ wget https://developer.download.nvidia.com/compute/cuda/repos/debian11/x86_64/cuda-keyring_1.0-1_all.deb

$ sudo dpkg -i cuda-keyring_1.0-1_all.deb

Note the instruction at the bottom of the instructions!:

$ sudo apt install libnccl2 libnccl-dev

This gave an error - Packages are not found. Download the DEB files from (cuda 11.2 version):

https://pkgs.org/download/libnccl2 and https://pkgs.org/download/libnccl-dev

Then install these with Gdebi.

Then change the “run_cmake” file again (add :”-DNCCL=TRUE”):
$ cd amber22_src/build
This one worked:

```
#  Assume this is Linux:

  cmake $AMBER_PREFIX/amber22_src \
    -DCMAKE_INSTALL_PREFIX=$AMBER_PREFIX/amber22  -DNCCL=TRUE \
    -DCOMPILER=GNU   -DBUILD_QUICK=TRUE \
    -DMPI=TRUE  -DCUDA=TRUE  -DINSTALL_TESTS=TRUE \
    -DDOWNLOAD_MINICONDA=FALSE \
    2>&1 | tee  cmake.log
```

Safe the file and close the tab.

Then reinstall from the “./run_cmake_nccl” step:

$ cd amber22_src/build/

$ ./run_cmake_nccl

$ make install

Download the test benchmark files:

GPU Benchmark tests: - https://www.exxactcorp.com/blog/Molecular-Dynamics/RTX3090-Benchmarks-for-HPC-AMBER22-A100-vs-RTX3080-vs-RTX3070-vs-RTX6000 (link there to Amber Benchmark Suit - https://ambermd.org/Amber20_Benchmark_Suite.tar.gz)

Go to the Diownloads folder and use “wget” to transfer the file.

$ wget https://ambermd.org/Amber20_Benchmark_Suite.tar.gz

(Open the Nvidia X-server Settings GUI to see how many GPUs your Nvidia card has (10752 in my case)

Extract the folder into your Downloads directory. Command to test GPUs only:

$ ./runBenchmarks.sh

Use the Nvida X Server settings GUI to see the GPU usage.

$ ./runBenchmarks.sh

Usage: ./runBenchmarks.sh [-NGPU <number>, -RUNID <string>, -NCORE <number>,
-SPEC_GPU <gpu ids>]
[-SKIP_CPU, -SKIP_SERIAL_CPU, -SKIP_PARALLEL_CPU
-SKIP_GPU, -HELP, --help, -CLEAN]

Options:
-NGPU :: the number of GPUs in the system (default behavior is to use nvidia-smi to count the number of GPUs). This can also be used to request benchmarks on the first NGPU cards, as enumerated by ${CUDA_VISIBLE_DEVICES}.
-RUNID :: identification tag to leave on all run output files
-NCORE :: the number of CPU cores in the system (relevant for running serial CPU tests in parallel and the degree of parallelism in pmemd.MPI tests
-SPEC_GPU :: followed by a string of integers delimited by commas (no spaces) such as '0,3,4,6' to indicate the GPU IDs to benchmark. This can be fed the value of ${CUDA_VISIBLE_DEVICES}, for example. The default behavior is to benchmark all GPUs, one by one.

-SKIP_CPU :: skip all CPU benchmarks
-SKIP_SERIAL_CPU :: skip serial CPU benchmarks
-SKIP_PARALLEL_CPU :: skip parallel CPU benchmarks
-SKIP_GPU :: skip GPU benchmarks

-HELP, --help :: print the help message (exit if this is the only argument)
-CLEAN :: clear previous results corresponding to ${RUNID}. This script takes ${RUNID} as a means of keeping multiple benchmarks simultaneously. Furthermore, results bearing a given ${RUNID} will be accepted by the script when found, rather than repeating the test.

NOTE: for those really interested in serial CPU tests, in order to accelerate them this script will run as many independent tests simultaneously as the CPU cores allow. However, this may cause serial CPU tests to compete for chip cache, slowing each of them down somewhat. For the best possible single CPU speed, run these benchmarks with:

./runBenchmarks.sh -SKIP_GPU -SKIP_PARALLEL_CPU -NCORE 1

Error :: A number of GPUs has been requested that exceeds system contents.

Test if **Quick** is working: https://quick-docs.readthedocs.io/en/22.3.0/hands-on-tutorials.html

source /amber22_src/AmbertTools/scr/quick/quick-cmake/quick.rc

Old errors:

The “make install” step gave an error at 26%.

[26%] [BISON][CifYacc] Building parser with bison 1.21.9-1
"cifparse.y", line 1: unknown character `-' in declaration section
"cifparse.y", line 1: no input grammar
free(): unaligned chunk detected in tcache 2
Aborted (core dumped)
make[2]: *** [AmberTools/src/cifparse/CMakeFiles/cifparse.dir/build.make:74:
AmberTools/src/cifparse/cifp.tab.c] Error 134
make[1]: *** [CMakeFiles/Makefile2:4209: AmberTools/src/cifparse/CMakeFiles/cifparse.dir/all]
Error 2
make: *** [Makefile:156: a sudo nautilus ll] Error 2

Install Flex and Bison:
$ sudo apt-get install flex bison

The “make install” step then worked.