Vikram Manikantan
vik@arizona.edu
Last Edited: 11/09/22

Setting Up Athena++

Athena++ already has great documentation - see here. However, setting up system dependencies are a little more complicated than they detail.

Downloading the Code

You can either create your own **fork** from the master repository or create a branch from the fork I've already created. I would go with the latter if we want to consolidate our group's development efforts into one branch. Here is a link to refresh your git commands, I find myself needing it very often :)

Installing Necessary Packages

You should start with this link for the basic system dependencies needed (a C/C++ compiler, python, etc.). Here are some pointers on the more annoying ones.

MPI

There are a couple of packages you can use (here is a comparison of the two main ones). You want to use **mpich**. You can install the package through your terminal (assuming linux) using:

```
sudo apt install mpich
```

HDF5

https://forum.hdfgroup.org/t/installing-hdf5-ready-version-of-open-mpi/4998

Follow the commands in the link above, BUT download your copy of HDF5 from this link.

You might have to also add some of these addresses (please change path to wherever you installed HDF5):

export PATH=/mpich/install/path/bin/:$PATH
export LD_LIBRARY_PATH=/mpich/install/path/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=/path/to/hdf5/lib:$LD_LIBRARY_PATH

For some reason, this command made everything work for me:

export CC=mpicc

Vikram Manikantan
vik@arizona.edu
Last Edited: 11/09/22

How to find your HDF5 package

>> find / -name "hdf5.h" -print 2>/dev/null

This will show you all the locations of hdf5.h on your computer. You can try each address one by one to see if it works if added to the hdf5_path link. Disclaimer: none of these actually worked for me and it was the one that did not show up on this list that worked (shown above).

FFTW

FFTW is a package for doing fast fourier transforms. According to the Athena++ wiki, this is needed to do self-gravity and turbulence driving in serial and parallel. We know the drill now:

```
sudo apt install libfftw3-3
```

```
Or from source: http://www.fftw.org/fftw2_doc/fftw_4.html#SEC55
```

Uninstalling Packages

I found myself installing and then uninstalling packages very often because I was confused. So, here is a quick summary of how to fully uninstall packages. **Note:** This only works with packages you installed using apt-get*.

Run: `sudo apt-get remove <package-name>`

However, you might find that certain package specific commands still work (namely h5cc -showconfig if you used apt-get to install hdf5). To get rid of all the associated packages that were downloaded when the package was installed, you can run:

```
sudo apt-get autoremove
```

Which will get rid of all files not associated with a currently installed package.

*I am now realizing that the -get portion of apt-get is not necessary for all of these commands to work. Just sudo apt will work fine.

Testing the Code Works

If you have installed everything above correctly, then the code should work! In your Athena folder, cd into tst/regression and run:

Vikram Manikantan
vik@arizona.edu
Last Edited: 11/09/22

```
python run_tests.py
```

This will take a while, so make sure you don't have anything else to do on the terminal. Or, open a new terminal window and let this run.

Test Log

11/11/2022:

- 61 out of 67 tests passed
- Failed tests:

Eos.eos_hdf5_table
Regression test for general EOS 1D Sod shock tube with HDF5 tables.

Grav.jeans_3d
Regression test based on Newtonian hydro linear wave convergence problem. Runs a linear wave convergence test in 3D including SMR and checks L1 errors (which are computed by the executable automatically and stored in the temporary file linearwave_errors.dat). Overall test.

Grav.unstable_jeans_3d_mg
Regression test based on Newtonian hydro linear wave convergence problem. Runs a linear wave convergence test in 3D including SMR and checks L1 errors (which are computed by the executable automatically and stored in the temporary file linearwave_errors.dat). Serial Multigrid execution.

Outputs.all_outputs
Regression test for all output types Runs Orszag Tang vortex test, restarting the job twice and making history (hst), formatted table (.tab), VTK, and HDF5 (if available) outputs. Then reads last version of each file to make sure output data is correct.

Pgen.hdf5_reader_parallel
Parallel test script for initializing problem with preexisting array

Pgen.hdf5_reader_serial
Serial test script for initializing problem with preexisting array

Shearingbox.mhd_shwave
No description

Turb.turb_3d
Regression test based on intercomparison between serial and MPI runs. Runs a driven turbulence test in 3D and checks L1 errors between serial and MPI runs using kinetic energy history.

Vikram Manikantan
vik@arizona.edu
Last Edited: 11/09/22

-

Running the Code

https://github.com/PrincetonUniversity/athena/wiki/Tutorial

Parallelization: Using MPI and OpenMP

There are two types of parallelization: MPI and OpenMP. MPI splits your process **across** CPU cores. You can check how many cores you have in settings. OpenMP multi-threads the process **within** each core.

For best performance, MPI should be set to the number of cores that you have available.

For OpenMP, you first have to check whether the cores have the ability to multi-thread (my intel CPU has something called hyper-threading). This should also tell you how many threads allows for best performance. In my case, it was 2.

So these are the optimal settings in my case:

No. MPI cores = 8
No. OpenMP = 2

To make these settings in the code, you will have to follow the instructions here about meshblocks etc.

tl;dr MeshBlocks divide the grid into sub blocks. You need to satisfy this condition:

(MeshBlock_x1/N_x1)*(MeshBlock_x2/N_x2)*(MeshBlock_x3/N_x3) > MPI_cores x OpenMP threads.

In words, the total number of meshblocks you have must exceed the product of the cores and threads you parallelize across. For example, if I had a grid 128x128x128 and defined meshblock size as 64x64x64. I would have 8 meshblocks. Therefore, these are my possible combinations for MPI and OpenMP:

8x1 ; 4x2 ; 2x4 ; 1x8;

But, since my cores can only do 2 threads per core, the best setting would be one of the first two.

Vikram Manikantan
vik@arizona.edu
Last Edited: 11/09/22

Resolution and Meshblocks

I don’t understand this yet. Okay now I do understand.

Your resolution defines the grid on which you will evolve your simulation. Your grid can be decomposed into sub-blocks, called MeshBlocks