Journal of Adrian Gusa for Spring Quarter 2007

Adrian G Gusa

April 4, 2007

1 Week of March 28, 2007

1.1 fsnode.py

I started off by running the fsnode.py program on the Itanium cluster at OSC. I launched three engines for directions 1, 2, and 3 respectively.

A problem related to running fsnode.py was the passing of different command line parameters to the engines. After trying different methods, I finally found a solution. This is based on doing IPython1 push calls for sending the parameters (different for each engine) to the engines, where they would be added in the global namespace. The program is then executed by an IPython1 executeAll call with "execfile('fsnode.py')". All the initialization is done in a separate python file which is run on the login node of the supercomputer after starting the engines.

1.2 HDF5

I read about the HDF5 file format, trying to understand the concepts behind it. I also read about parrallel HDF5 which uses MPI I/O for providing a unique file image to MPI processes. It is still unclear to me how we could have multiple processes or nodes of a supercomputer simultaneously write to an HDF5 file. In other words, how to solve the concurrency issue when dealing with HDF5 files.

After mailing to Fernando Perez and asking for advice on working with HDF5, he provided the code he had been working on for reading/writing HDF5 files using PyTables, a HDF5 implementation for Python.

I started reading the PyTables documentation, and looked at the code I downloaded from Fernando.

2 Week of April 4, 2007

I installed Pytables on my computer. I spent a lot of time on an installation problem. After installing all the prerequisites, and Pytables itself, when trying to use PyTables I had an error saying that the HDF5 library could not be found. However, the installation of PyTables reported to have found the HDF5 header and library files. I finally solved this problem by adding the path to the HDF5 library to the environment variable LD_LIBRARY_PATH.

I spent some time going through and understanding Fernando's code as well as learning how to use Pytables. I also looked briefly at Robert Harrison's C++ code for saving a mwadap function. Fernando raised the issue of being able to save scalar values to a node. Node values are saved using the saveArray(...) function which always saves an array dataset. However, if the value passed to the method is a scalar value, instead of an array with one element, the data is saved as a scalar value inside the dataset. When reading from the node, the returned value is a scalar of the right type. This solved the problem of dealing with scalar values.

I completed the methods for saving (mostly done by Fernando) and loading a mwadap function. I also implemented a small test suite for the save/load methods. This generates some functions from a script, saves a function to a file, opens the file again, makes sure the top-level nodes exist in the file, and loads the function. Then, based on the equality function implemented in the TreeSkeleton class, the two functions (original and loaded) are compared.

Finally, I worked on the methods for saving/loading seprep objects used for a wavefunction: SpinThing, TrueSeparable, SumSeparable.

3 Week of April 11, 2007

This week I spent most of my time implementing the methods for saving and loading seprep objects: SpinThing, TrueSeparable, SumSeparable.

I organized the code written so far into two libraries called h5lib.py. One of them is part of the mwadap library and contains the methods for working with mwadap Functions. The other one is in seprep and contains the methods for working with the seprep types.

I also spent quite a lot of time doing speed tests for the implemented methods. The big problem I encountered was that the HDF5 library I had written was a lot slower than the Python Pickle module. I ran different tests trying to figure out the causes for this, and I even came up with a new implementation for the library which brings a significant speed-up, but will definitely cause synchronization problems if used with multiple processes. So, this won't help very

much in practice, but it showed that HDF5 got slower with subsequent writes to the same file, and closing and opening the file for each write operation could bring an improvement.

I wrote a document describing the structure of the wavefunction in the HDF5 file and included the results of the speed tests.

4 Week of April 18, 2007

I started this week by finalizing my document on the speed tests, and working on a new simplified structure for the HDF5 wavefunction format. This simplified format brought a great speed-up to the library.

I implemented a series of new methods for mwadap and seprep objects serialization. This time, the functions used a directory structure similar to the structure of the HDF5 format. The actual information was pickled at the required positions in the directory structure: mwadap Functions at the lowest levels of the tree, and other pickles for the scalar and conditioned_flag of TrueSeparable objects in the TrueSeparable directories of the tree.

I spent most of my time on the problem of timestamping the loaded objects and synchronizing the object and the corresponding filesystem structure. The purpose of this is to make reloading objects (SumSeparable, TrueSeparable, SpinThing) faster by reloading outdated information only. However, after considering different models, I reached the conclusion that the easiest valid way to do this is to timestamp only on the bottom levels (mwadap Function levels). Any time a reload is needed on a SumSeparable for instance, the timestamps of the mwadap Function objects within are compared with the timestamps of the files on disk, and only outdated information is reloaded.

5 Week of April 25, 2007

This week I began a new assignment. This consists of profiling existing mwadap code which is very slow and needs to be improved in terms of speed.

I began by updating the libraries on my computer in order to have the latest versions. I also installed the python2.4-profiler package as well as KcacheGrind, a utility for visualizing profiling output. I continued by running small tests and getting familiar with KcacheGrind.

I did some profiling runs for different methods of a wavefunction which need speed ups. One of the methods that needed speed improvements was one computing values for Legendre polynomials. The method, called plegnv, is part of

the mwadap library.

I implemented two more versions for the plegnv method. One version is using the weave library for embedding C code into the python code. After running the profiling tests with the new function, I noticed a speed up of about 50% for plegnv as compared to the previous python only version. The second version used a C library and a Pyrex interface for wrapping the library. The speed up was about the same as for the first version, however the complexity of the code was greatly increased.

6 Week of May 2, 2007

I started the week by trying to implement an improved version of the weave library in which coefficients required by plegnv are pre-computed and available in a global static array. As it turned out, there was a small speed-up for this new version, but not significant. This new version failed to pass a precision test that the previous versions of the function passed. So, this version was abandoned for now, since the speed up didn't justify further investigation of the precision issue.

I continued by implementing a C extension module for Python which would perform the computation of plegnv. In this case, because of the speed of the C code, and the ability to import the C shared module as a Python module, there was a speed-up of about 3 times as compared to the weave version I implemented last week. This version passed the precision tests so I commited the code to the mwadap library.

Starting from the middle of the week I moved on to something different. I looked at some classes which Fernando Perez began to write and finished implementing them. I also completed a test suite for these new classes and the tests were passed.

I finished the week trying to integrate the new classes into the existing mwadap code. Apparently this is not a easy and fast thing to do, so this might take me some time.

7 Week of May 9, 2007

This week I continued to work on the integration of the new key classes into the mwadap code. This is something I started last week, but requires me to dig in the mwadap code quite a bit so the process is a little slow.

I finished fixing all the errors, and doing a small change that Fernando Perez

requested.

I then started looking at Robert Harrison's code. My next assignment is to convert his wavefunction to mwadap/seprep objects by going through a HDF5 file which could then serve as a means of exchanging information between the two teams.

I am having some problems with the format in which Robert stores the wavefunction which needs further investigation.

8 Week of May 16, 2007

I continued to look at Robert Harrison's code and wrote a function for dumping one of his wavefunctions into two mwadap functions.

I also spent some time installing VTK and mayavi on my laptop for visualizing mwadap functions. However, I had some errors when trying to plot a mwadap function imported from the other project. The plotting of a mwadap function generated from C code works just fine.

9 Week of May 23, 2007

I started the week by cleaning up the installation on the Itanium cluster at OSC, preparing everything for copying by others. I have also built a sample fsnode test that can be run to test that the installation works, as well as provide an example of how to use IPython to run parallel job.

I then continued to work on saving a madness wavefunction into HDF5 format and succeeded in doing that. However, as it turns out, because of the simetry in madness only a part of the wavefunction is saved and there should be a parameter for forcing the saving of the complete wavefunction.

Towards the end of the week I looked again at some installation issues on the Itanium cluster at OSC. As it turns out, there are some problems with the scipy installation. I don't know if that is because of the latest scipy version, or because of some configuration issue.

10 Week of May 30, 2007

In the beginning of the week I prepared for the presentation about IPython and the Itanium cluster which I had to make for the end of the quarter. I spent

most of my time building some new examples for using IPython with different computing models.

Throughout the week I kept writing the final report, more installation instructions as well as documenting the code that I had written so far.

11 Week of June 6

This week I worked on the code that I had previously wrote for dumping a madxx wavefunction from the running program. I corrected some parameters which were not properly saved for the mwadap functions. I also wrote some scripts for dumping a madxx wavefunction as two mwadap functions represented in HDF5 format, and for converting a mwadap function from HDF5 format to a pickle.