A user's perspective on getting started with Carpet

Ulrich Sperhake, Erik Schnetter

Date: 2004/08/05 14:43:52

1 Introduction

These notes provide information on how to install and use the package Carpet as seen from a user's point of view. Carpet is a set of Thorns that provide fixed and to some extent adapted mesh refinement in the Cactus environment. As Cactus is a necessary requirement for using Carpet, these notes will inevitably contain some information about Cactus as well.

The reader should regard these notes as a first draft and the information represents the author's personal experiences rather than an exhaustive recipe on getting Carpet to work on an arbitrary given platform. In this sense I am hopeful that users as well as developers will continue to add to this document to make it more useful in the future.

Useful starting points for retrieving more detailed information on various issues are the project's web pages

```
http://www.cactuscode.org

http://www.carpetcode.org
```

2 Downloading the necessary packages

One first needs to download the Cactus version 4.0.13 (or alternatively for the more daring the development version). A more detailed description about how this is done can be found on the Cactus web page

`http://www.cactuscode.org`

Here we will summarize the required steps for downloading the complete Cactus-4.0.13 package. Change to a suitable directory on your system and log onto the Cactus cvs server via

```
cvs -d :pserver:cvs_anon@cvs.cactuscode.org:/cactus login
```

which will prompt you for a password which is `anon`. For the development version you will need to choose the directory `/cactusdevcvs` instead. Next check out the Cactus flesh which will create a directory `Cactus` under your current location

```
cvs -d :pserver:cvs_anon@cvs.cactuscode.org:/cactus checkout Cactus
```

The rest of the cactus checkout is best done with the scripts that are shipped as part of Cactus. Change to that directory

```
cd Cactus
```

and enter the command

```
make checkout
```

That will give you various options to choose those parts of cactus you want to ckeckout. The default option *arrangements* is quite satisfactory for this purpose, so just hit return. You will then be given a list of (at the time of writing) 13 Cactus arrangements. Getting them all is a good idea, so choose once more the default option by pressing return. Depending on your internet connection this may take a while. Once all is downloaded you want to quit the script. This is not the default option, so type `q` and hit return.

In order to run the WaveToy example that comes with CarpetExtra (see below) you will need to check out Erik Schnetter's package TAT. First switch the directory to

```
cd arrangements
```

then checkout

```
cvs -d :pserver:cvs_anon@cvs.cactuscode.org:/arrangements checkout TAT
```

Again this may take a little time. Finally you will have to check out the `Carpet` package. As of mid April 2004 Carpet consists of 4 arrangements. `Carpet` contains all the necessary thorns you will need to run Carpet in the first place. The latest cutting edge thorns currently under development are located in `CarpetDev`. Do not be too surprised, though, if you find some the tools in there not to be fully functional. Packages not required to run `Carpet`, but probably useful for various purposes, such as scalar wave examples, are located in `CarpetExtra`. Remain in the `arrangements` directory for this purpose and log into the carpet cvs-server

```
cvs -d :pserver:cvs_anon@cvs.carpetcode.org:/home/cvs/carpet login
```

the password being once more `anon`. Next checkout Carpet by typing

```
cvs -d :pserver:cvs_anon@cvs.carpetcode.org:/home/cvs/carpet checkout Carpet
cvs -d :pserver:cvs_anon@cvs.carpetcode.org:/home/cvs/carpet checkout CarpetExtra
cvs -d :pserver:cvs_anon@cvs.carpetcode.org:/home/cvs/carpet checkout CarpetDev
```

3 Documentation

Documentation about Cactus, Carpet and their separate thorns comes in different forms. Most importantly you generate the UsersGuide and ReferenceManual for Cactus by going into the `Cactus` directory and typing

```
make UsersGuide

make ReferenceManual

make ArrangementDoc

make ThornDoc
```

(four separate commands). They will be created in postscript format under the directory

```
doc
```

relative to your current position, i.e. the Cactus directory. In addition each thorn may contain a subdirectory `doc` where the author (or users) may store additional documentation, typically in the form of a file `documentation.tex`.

4 Compilers

Before we indulge in using Cactus/Carpet, we have to address issues concerning the system you are working on. We begin with the compilers although we will not be able to deal with the subject in an exhaustive fashion. Basically these notes list our experiences with local machines (i.e. at Penn State) and may or may not be valid for your environment. Users are encouraged to add their experiences to this list.

At Penn State we largely work with the Intel compilers and the success of compilations has been found to depend sensitively on which version of the Intel compilers we are using. We will discuss some error messages encountered in the process of compiler testing below.

Free download (at least for Linux) of the Intel compiler (Fortran and C++) for non-commercial private or academic use is available from the web page

`http://downloadfinder.intel.com/scripts-df/support_intel.asp`

(click on Software Development, check for the compilers on your system and follow their instructions).

In case you haven't got root access, you may need to install the compiler locally or you will have to ask your sys-admin. Additional difficulties may arise in case you have no root access, i.e. install locally, while your sys-admin keeps some older version installed. In order to make sure that no conflict arises thereof (e.g. by linking against old versions of the library) the environment variable `LD_LIBRARY_PATH` must point to your local new version and not to the old version in `/usr/local` or wherever. You will probably end up with error messages such as `undefined symbols ...` otherwise. We decided to use the Intel compiler for both Fortran and C++ code. This was mainly a result of the current version of g++ not having the complete stl libraries that are made use of extensively in Carpet.
An important aspect of the Intel compilers is that they come in various different versions. Even the

same version number (say 7.1) comes in many different releases. You can check this by typing

```
ifc -V
```

and likewise for `icc`. Note in particular the date of build given in the form of *20030307Z*. This corresponds to the March 2003 build of version 7.1 and caused difficulties for me. I encountered an error message like

```
/home/terminator/sperhake/src/2004_02_16_cactus-FMR/configs/test01/build/CarpetLib/
data.cc(173): error: no instance of overloaded function "dist::datatype"
matches the argument list
```

This can be rectified by switching to a newer release, at least the September 2003 build of version 7.1 (I'd recommend doing that for both the Fortran and the C++ compiler).

Some Cactus-Carpet users have reported problems, such as segmentation faults, by using the most recent versions of the Intel compilers, namely the March 2004 release of version 7.1 and the latest version 8.0. So far we have been using the former of these without encountering any difficulties, but you should probably stick to the December or September 2003 version of 7.1 if you can.

On my Gentoo Linux laptop, on the other hand, I experienced trouble with the September 2003 version of 7.1. I received error messages like

```
struct stat stat_bbox ...
Incomplete components in structure not allowed
```

at compilation (I have forgotten the exact wording, but you'll recognize it). I managed to work around this by using the Intel Fortran and C++ compilers version 8.0 (build October 2003). As I have not done extensive code development on this laptop, though, I cannot really comment on the potential issues concerning the 8.0 version mentioned above.

Trouble may also arise from preprocessing in case you are using RedHat 7.3 (possibly also with other versions). This is essentially related to the treatment of white space in Fortran files. Should you encounter rather stupid error messages which clearly indicate that proper lines of Fortran have been corrupted by introducing white space (e.g. line breaks) at preprocessing, you should check your cpp and possibly download another (probably older) version. Details about this can be found on

`http://www.cactuscode.org/Documentation/Architectures/Linux.html`

which also gives a link to the preprocessor of the older RedHat 6.2 distribution. I downloaded that older version and it solved the preprocessing problems I encountered prior to that.

5 Libraries

As much as the compiler issue is strongly dependent on your platform, the extent to which you will have to install new libraries will depend on what your system administrator has already done for you. Again these notes cannot be exhaustive and rather focus on our experience. Feel free, as before, to add to our list.

5.1 HDF library

The HDF5 library is required for handling in/output in a particular binary data format. The use of these libraries in Cactus/Carpet is entirely optional, but in the end I found it easier to install the libraries than to convince my system that I do not want to use them. They should be useful in the long run anyway, so I recommend their installation unless they are already part of your system.

Let us start with the hdf5 libraries. The binary version can be obtained from

`ftp://ftp.ncsa.uiuc.edu/HDF/HDF5/hdf5-1.6.1/bin`

As before I prefer compiling the source which you can get from

`ftp://ftp.ncsa.uiuc.edu/HDF/HDF5/hdf5-1.6.1/src`

Again the instructions in the `INSTALL` file are straightforward. I included the C++ interface by setting the options

```
./configure --enable-cxx
```

and used the variables `CPPFLAGS` and `LDFLAGS` to ensure that the szip libraries were found (see `INSTALL` file). The Fortran interface did not work for me, so I did not enable that. In future versions of this document this issued may be readdressed. Finally you may need to point the environment variable `LD_LIBRARY_PATH` in your `.bashrc` or `.cshrc` to the directory containing the hdf5 library.

5.2 Parallelization

This subsection is relevant only if you plan to do multi processor runs (which you are rather likely to do, though, since it is a key feature of Cactus/Carpet). There are various packages that take care of parallelization, such as `MPICH` or `lam` and your machine will probably come equipped with one of these.

I have only had the need to install a message passing interface `(MPI)` on my laptop. It's a single processor laptop but you can emulate multi-processor runs none the less. Furthermore it appears to me that Carpet expects `MPI` at least in the form of a header file `mpi.h`, so you'd better install it. I chose the `lam` package for this purpose, so that is the only experience I have to report.

Installation of this package was straightforward on my Gentoo Linux laptop by typing

```
emerge lam-mpi
```

Depending on your Linux flavor installation may be done differently, for example using `rpm`. `lam` is started by typing

```
lamboot
```

and then executables can be started via

```
mpirun -np <n> <executable>
```

where <n> is the number of processors and <executable> the binary file (with full path) you want to run.

6 Creating a configuration

6.1 The configuration file

Eventually we can start writing a configuration file for a Cactus-Carpet project. In this configuration file the paths to various files, such as libraries and compilers need to be specified. Naturally these paths will differ from machine to machine. In this subsection I will assume the installation path

```
/usr/local/<name>
```

for most libraries, where <name> is the name of the library, e.g. `hdf4` or `szip`. I further assume that each of these directories contains subdirectories `lib` and `include` which contain the libraries and header files. Similarly I presume that all compilers/preprocessors are installed in the directory

```
/usr/local/for_carpet/bin
```

This is, of course, not where they reside on your machine (nor on mine), but it'll be sufficient for this document and you will merely have to replace each of these paths with the correct one on your system.

We are now in the position to create the configuration file, say `mycode_carpet.cfg` (you can store that file wherever you think convenient). We will focus on the most important entries in this file only. Please refer to the Cactus documentation for a more detailed description. First we specify information about the compilers

```
F90     /usr/local/for_carpet/bin/ifc
F77     /usr/local/for_carpet/bin/ifc
CC      /usr/local/for_carpet/bin/icc
CXX     /usr/local/for_carpet/bin/icc
CPP     /usr/local/for_carpet/bin/cpp
FPP     /usr/local/for_carpet/bin/cpp
```

(the exact amount of white space between the variables `F90, F77,...` and their entries should not matter and you may even put in an = sign). Note that you do not need to specify the full path if your environment variable `PATH` points to the correct versions of the compilers/preprocessors already. Next we need to specify information about the message passing interface. In my case that was `lam`, so the next entries in my file `mycode_carpet.cfg` are

```
MPI             LAM
LAM_INC_DIR     /usr/include
LAM_LIB_DIR     /usr/lib
```

In case you are using a different `MPI` package refer to the Cactus users guide to find the correct entry for `MPI`. Make sure that you specify the correct paths for the corresponding header files and libraries (ask your sys-admin if necessary).

Next we specify the libraries to be included in the compilation. For the 7.1 version of the Intel compilers in combination with `lam` we found the following to work fine

```
LIBS            crypt lapack blas g2c z BINDF90 CEPCF90 F90 IEPCF90 PEPCF90
                POSF90 cprts cxa guide imf intrins irc ircmt ompstub svml
                unwind X11 ieeeio df m mpi lam pmpi
```

(all in one line). It goes without saying that all these libraries must be installed on your machine. Most of them probably are and the installation of some that may not is described in more detail above in Sec. 5.
The paths to some of these libraries may not be known automatically by the linker and needs to be specified separately. This is done with the variable `LIBDIRS` which I had to set to

```
LIBDIRS         /usr/local/intel/compiler70/ia32/lib
                /usr/X11R6/lib /usr/local/IEEEIO/lib /usr/local/hdf4/lib
                /usr/lib/gcc-lib/i386-redhat-linux/egcs-2.91.66
```

(again on all in one line). As before you will have to adjust this line to your demands.
Finally I set

```
PTHREADS        yes
```

though I am not sure what this is exactly doing.

6.2 make-config

In order to create a configuration change into the `Cactus` directory and type

`make` <name>`-config options=`<config-file>

where you can choose an arbitrary <name> for your configuration and <config-file> is the file (with full path) created in the previous subsection.

6.3 Creating a thornlist

Next you will need to generate a thornlist, i.e. a list of all those thorns you want to compile. This is done in the `Cactus` directory by typing

`make` <name>`-thornlist`

where <name> must be the same as in setting up the configuration. This command will search all arrangements for all thorns and eventually prompt you whether you want to modify the list. As all thorns are activated by default you do want to modify the list and type *yes* and hit return. This will open an editor session where you can unselect thorns by putting a hash '#' at the beginning of the line. Unselect all thorns in this way except for the following

```
CactusBase/Boundary
```

```
CactusBase/CartGrid3D
CactusBase/CoordBase
CactusBase/IOBasic
CactusBase/IOUtil
CactusBase/LocalInterp
CactusBase/SymBase
CactusBase/Time
Carpet/Carpet
Carpet/CarpetIOASCII
Carpet/CarpetIOHDF5
Carpet/CarpetInterp
Carpet/CarpetLib
Carpet/CarpetReduce
Carpet/CarpetRegrid
Carpet/CarpetSlab
CarpetExtra/IDScalarWave
CarpetExtra/WaveToyF77
```

Before you compile, you need to apply one modification to the file

`arrangements/CarpetExtra/WaveToyF77/configuration.ccl`

namely remove the entry `Cart3d` from the list of `REQUIRED` thorns. This thorn is actually required but, for some reason unknown to me, must not be mentioned here. It gave an error message complaining that there is no thorn `Cart3d`. Having applied this modification you can start compiling by typing

`make` <name>

There is no guarantee, but at least you have a chance of compiling through without error messages (do not be intimidated by the odd warning, though). In case you still cannot compile, please add your wisdom to this document to help future users.

6.4 Running the first application: WaveToyF77

If you've gotten this far, you should be able to run your first simulation with mesh refinement. Change to some convenient directory for this purpose and copy over from relative to the main `Cactus` directory the parameter file

`arrangements/CarpetExtra/WaveToyF77/par/wavetoyf77_rad_full_rl2.par`

You will need to adjust this parameter file a little to get it running (I am not aware of a WaveToy-parameter file that does not require such minor modification). First add to the first line beginning with `ActiveThorns` the thorns `Slab CoordBase SymBase` (that is within the quotes). Finally you should be able to run this example by typing something like

```
mpirun -np 1 ~/Cactus/exe/cactus-<name> wavetoyf77_rad_full_rl2.par
```

where <name> is again the name of the configuration above. In case you do not have your main `Cactus` directory under your home directory you will need to adjust that part in the command. By running this command you should obtain a directory `wavetoyf77_rad_full_rl2` with the resulting data in ascii format. You can check for example the file

```
wavetoyf77_rad_full_rl2/phi.x.asc
```

(relative to the directory where you ran the code) which lists the data on the separate refinement levels.