Overture Installation Instructions

W.D. Henshaw,
Department of Mathematical Sciences,
Rensselaer Polytechnic Institute, Troy NY, 12180, USA.
Overture : http://www.overtureFramework.org

Abstract

This document contains step-by-step instructions on how to install Overture and the supporting packages.

Contents

1 Machines and Compilers where Overture has been built

Platform (OS)	C++/C	Fortran
Intel/AMD (Redhat) (32/64 bit)	gcc 4.3.2	gfortran/pgf77/ifort
Mac OS X	gcc 4.3.2	gfortran/xlf

Table 1: Overture has been built on these machines with these compilers. Notes: pgf77 is the fortran compiler from PGI, ifort is the intel fortran compiler.

Our current development platform is Linux-Intel 64-bit with gcc and pgf77.

2 Installing Open Motif

Note: Mac OS/X users should skip ahead to Section 10.

You may need to install Motif if you do not already have a version. Most Linux systems now come with Motif in `/usr/lib/libXm.so` (or `/usr/lib64/libXm.so` for 64 bit machines) in which case you can skip this step.

The Motif library is called `libXm.a` or `libXm.so`. In older versions of Linux the default version of Motif is probably LessTif. I have had trouble with LessTif so you should install open Motif which you can get for free from open Motif. I have tried version 2.2.1. There is an rpm available with precompiled binaries. Here are the steps I took to install open Motif from the source files:

1. `gzip -d openMotif-2.2.1.tar.gz`
2. `tar -xf openMotif-2.2.1.tar`
3. `cd openMotif-2.2.1`
4. `setenv CC gcc` : set CC to your favourite C compiler, usually gcc on linux
5. `./configure --prefix="this directory"` : where "this directory" is the current directory (or the place you want to install Motif).
6. `make`
7. `make install`

3 Installing Mesa OpenGL

You may need to install Mesa (OpenGL), Most Linux systems should already have OpenGL. However, if you would like off-screen rendering support (for making high-resolution hard copies) then you should install Mesa. I normally run with Mesa rather than the native OpenGL.

I currently use version Mesa-7.2.

Assuming you have downloaded `MesaDemos-XXX.tar.gz` and `MesaLib-XXX.tar.gz` in directory MESA (e.g. /home/henshaw/Mesa)

1. `tar xzf MesaLib-XXX.tar`
2. `tar xzf MesaDemos-XXX.tar.gz` (optional)
3. `tar xzf MesaGLUT-XXX.tar.gz` (optional)
4. `cd Mesa-XXX`
5. `make` (to see a list of available machines)
6. `make <machine-name>` where `<machine-name>` is the appropriate choice, such as linux-x86 (linux-x86-64). I usually use dynamic libraries.
7. Goto to one of the example directories (such as demos) and run a test. If you have built dynamic libraries then you will first need to set your LD_LIBRARY_PATH as indicated by the message from Mesa when it finished building.
8. In previous versions of Overture one needed to build the libGLw library in the `cd widgets-sgi` directory. This is no longer necessary since these widgets will be built with Overture.
9. Mesa has off-screen rendering support (for creating hard-copy's) found in libOSMesa.a which should automatically be built with Mesa. Overture also needs access to some of the source files in Mesa (such as Mesa-XXX/src/mesa/main/context.h) so your OpenGL environmental variable should point to the Mesa source (as opposed to /usr/lib) if you want this off screen rendering. Overture can also do off-screen rendering using X pixmaps, but this may not work if you have a graphics card that does direct rendering.
10. You may increase the maximum possible resolution that an image can be saved as. Mesa is compiled with certain default maximum values and the highest resolution depends on the parameters that were used to compile Mesa. To increase the Mesa resolution you should change the definitions for `MAX_WIDTH` and `MAX_HEIGHT` in `Mesa/src/config.h`. For example you may set these to be 2048 or more. Mesa must be re-compiled after these changes are made.

4 Installing HDF

HDF (`http://www.hdfgroup.org/HDF5/`) is a data base package that you may need to install. It may already be installed on your system.

Here are the installation steps for HDF5. I currently use hdf5-1.8.8 (1.6.5 should also work). You will need to build a serial version and/or parallel version depending on which version of Overture you are building.

1. `tar xzf hdf5-1.8.8.tar.gz`
2. optionally rename the directory hdf5xxx
3. `cd hdf5xxx`
4. `unsetenv CC`
5. `unsetenv cc`
6. For the serial version : `setenv CC gcc` (if you want to use gcc).
7. For the parallel version : `setenv CC $MPI_ROOT/bin/mpicc`.
8. `./configure --prefix=‘pwd‘`
9. Sometimes in parallel I instead need to use `setenv CC gcc` and then `./configure --prefix=‘pwd‘ --enable-parallel`.
10. `make`
11. `make install`

5 Installing A++

A++/P++ is the serial/parallel array class. To get A++/P++ you will need to go to the Overture home page and register under software.

The A++ installation process requires the gnu autoconf package and gmake, available from the gnu site http://www.gnu.org/order/ftp.html.

If you have downloaded AP-nnn.tar.gz into directory XXX then the basic steps are as follows (if something goes wrong, consult the documentation that comes with A++).

1. `cd XXX`
2. `gzip -d AP-nnn.tar.gz`
3. `tar -xf AP-nnn.tar`
4. `cd A++P++nnn`
5. `configure --enable-SHARED_LIBS --prefix=‘pwd‘`
6. `make`
7. `make install`
8. Optionally type `make check` to run the A++ test codes.

The libraries will be installed in `XXX/A++P++nnn/A++/install/lib`. You should use `setenv APlusPlus XXX/A++P++nnn/A++/install` when building Overture.

6 Installing P++

P++ is the parallel version of the array class. NOTE: many but not all features of Overture run in parallel. **Note** that we build the version of P++ without the newer PADRE implementation.

To build the P++ library (only required if you are going to build the parallel version of Overture)

1. `cd A++P++nnn` (goto the A++P++ directory)
2. Set the enviroment variable `MPI_ROOT` to point to the mpi directory such as `setenv MPI_ROOT /usr/local/mpi`)
3. Here is the configure statement I used to compile with gcc version 4.3.2 on my dual processor work-station

```
configure --enable-PXX --prefix=‘pwd‘ --enable-SHARED_LIBS \
          --with-mpi-include=-I${MPI_ROOT}/include \
          --with-mpi-lib-dirs="-Wl,-rpath,/usr/apps/gcc/4.3.2/lib -L${MPI_ROOT}/lib" \
          --with-mpi-libs="-lpmpich -lmpich -lc -lm"  \
          --with-mpirun=${MPI_ROOT}/bin/mpirun --without-PADRE
```

4. Here is a configure statement I used on a Linux cluster that did not use mpirun to start parallel jobs. P++ expects to find mpirun by default but you can overide this and provide another script. In the example below I use `mpirun-wdh`, a script that I wrote which calls the appropriate routine to launch parallel jobs.

```
configure --enable-PXX --prefix=‘pwd‘ --enable-SHARED_LIBS \
          --with-mpi-include=-I${MPI_ROOT}/include \
          --with-mpi-lib-dirs="-Wl,-rpath,${MPI_ROOT}/lib -L${MPI_ROOT}/lib" \
           --with-mpi-libs="-lmpi++ -lmpi" \
           --with-mpirun=${HOME}/bin/mpirun-wdh --without-PADRE
```

5. Here is the configure statement I used when I couldn't run mpirun on a front end machine so I disabled the checks that parallel jobs can be executed:

```
configure --enable-PXX --prefix=‘pwd‘ --enable-SHARED_LIBS \
          --with-mpi-include=-I${MPI_ROOT}/include \
          --with-mpi-lib-dirs="-L${MPI_ROOT}/lib" \
          --with-mpi-libs="-lmpi"\
          --disable-mpirun-check \
          --without-PADRE
```

   Here is the configure statement Robert Read used to install P++ on his laptop running fedora and Open MPI

```
configure --enable-PXX --prefix=‘pwd‘ --enable-SHARED_LIBS \
          --with-mpi-include=-I/usr/include/openmpi-x86_64 \
          --with-mpi-lib-dirs="-Wl,-rpath,${MPI_ROOT}/lib -L${MPI_ROOT}/lib" \
          --with-mpi-libs="-lmpi -lmpi_cxx -lc -lm" \
          --with-mpirun=${MPI_ROOT}/bin/mpirun --without-PADRE
```

6. `make`
7. `make install`
8. Optionally type `make check` to run the P++ test codes.

The libraries will be installed in `XXX/A++P++nnn/P++/install/lib`. You should use `setenv PPlusPlus XXX/A++P++nnn/P++/install` and `setenv APlusPlus XXX/A++P++nnn/P++/install` when building the parallel version of Overture.

Note: To change the maximum number of parallel processes that can be used by P++, change the file A++P++nnn/P++/include/maxProcessors.h before building the P++ library.

Here are more example builds of P++:

- To build on a Linux cluster at LLNL I first ran `mpic++` on a C++ file with the "-v" option to see all the libraries that were being linked to:

```
mpic++ -v -o conftest -I/usr/local/tools/mvapich2-gnu/include conftest.C
```

I then configured P++ using these libraries:

```
configure --enable-PXX --prefix=‘pwd‘ --enable-SHARED_LIBS \
   --with-mpi-include=-I${MPI_ROOT}/include \
   --with-mpi-lib-dirs="-Wl,-rpath,${MPI_ROOT}/lib -L${MPI_ROOT}/lib" \
   --with-mpi-libs="-lmpichcxx -lmpich -lpmi -lopa -lmpl -lpthread -lrdmacm \
                    -libverbs -libumad -ldl -lrt -lnuma -lstdc++ -lm" \
   --with-mpirun=${HOME}/bin/mpirun-wdh --without-PADRE
```

7 Installing PETSc (optional)

Overture can optionally use PETSc to solve sparse matrix problems. You will probably want to get PETSc if you want to solve the 3D incompressible Navier-Stokes equations with cgins.

The version of PETSc that works with Overture.v25 or v22 is PETSc 2.3.2petsc-lite-2.3.2-p6.tar.gz. Here are the steps I took to install the (uni-processor version) on my Linux machine where I use the gcc and PGI fortran compilers:

1. `tar xzf petsc-lite-2.3.2-p6.tar.gz`
2. `mv petsc-2.3.2-p6 petsc-2.3.2`
3. `cd petsc-2.3.2`
4. `setenv PETSC_DIR ‘pwd‘`
5. `setenv PETSC_ARCH linux-gnu-opt`
6. `setenv PETSC_LIB $PETSC_DIR/lib/$PETSC_ARCH`
7. `setenv CC gcc`
8. `(for help type:  ./config/configure.py --help)`
9. `./config/configure.py --PETSC_ARCH=linux-gnu-opt --with-debugging=0 --with-fortran=0 --with-matlab=0 --with-mpi=0 --with-shared=1 --with-dynamic=1`
10. `make`

Notes for Mac: Follow the above instructions but use:

1. `setenv PETSC_ARCH macx`
2. `./config/configure.py --PETSC_ARCH=linux-gnu-opt --with-debugging=0 --with-fortran=0 --with-matlab=0 --with-mpi=0 --with-shared=0 --with-dynamic=0`

8 Installing Overture

Here are instructions to install the Overture library. See also the README file that comes with Overture. To get Overture you will need to go to the Overture home page and register under software.

Overture requires perl for configuration and running tests. If you don't have perl you can get it for free from http://www.perl.com

Overture needs a collection of environment variables to be set. These variables tell Overture where to look for other libraries. You can edit the file **Overture/defenv** to define these variables and then type "`source defenv`" to actually set the variables. You might want to add them to your `.cshrc` so they get assigned when you open a window (shell).

- `setenv XLIBS xxxx` : Directory holding the `include` and `lib` directories for the X window system. For example you might type "`setenv XLIBS /usr`". If you then type "`ls $XLIBS/lib`" you should see the X libraries such as `libX11.so` or `libX11.a`
- `setenv OpenGL xxxx` : If you type "`ls $OpenGL/lib`" you should see the OpenGL libraries such as `libGL.so` (`libGL.a`), or `libMesaGL.so` (`libMesaGL.a`) if you use Mesa.
- `setenv MOTIF xxxx` : If you type "`ls $MOTIF/lib`" you should see the Motif library `libXm.so` or `libXm.a`
- `setenv HDF xxx` : If you type "`ls $HDF`" you should see the HDF directories including `include` and `lib`. If the HDF variable contains the sub-string "hdf5" in it, then the Overture configure script will know to use HDF5 (otherwise you will need to use the configure option useHDF5).
- `setenv APlusPlus xxxx` : if you type "`ls $APlusPlus`" you should see the A++ directories `include` and `lib`; if you type "`ls $APlusPlus/lib`" you should see the A++ libraries and if you type "`ls $APlusPlus/include`" you should see the A++ include files. Here is an example: `setenv APlusPlus /home/dilbert/A++P++/A++/install`.
- `setenv Overture xxxx` : location of the Overture directory that we are just going to install. Here is an example : `Overture /home/henshaw/Overture.v23.d`.
- `setenv CG xxxx` : If you install the CG solvers this is the location of the CG source directory. Here is an example : `setenv CG /home/henshaw/cg.v23`.
- `setenv CGBUILDPREFIX xxxx` : This is the location of the CG executable and object files. You can make this the same as `CG` or choose a different directory location. This option is used to compile multiple versions of CG (e.g. serial and parallel) from the same source tree. Here is an example : `setenv CGBUILDPREFIX /home/henshaw/cg.v23.d`
- `setenv LAPACK xxx` : Overture and CG need the LAPACK and BLAS libraries. This env variable is the directory that holds the `liblapack.a` and `libblas.a` libraries. Example: `setenv LAPACK /home/henshaw/lapack`
- Set the `LD_LIBRARY_PATH` environment variable which is used by the loader when you run an Overture program in order to find dynamic libraries (such as libX11.so or libOverture.so):

  ```
  setenv LD_LIBRARY_PATH ${MOTIF}/lib:${XLIBS}/lib:${HDF}/lib:${OpenGL}/lib:$Overture/lib:${APlusPlus}/lib
  ```

  For machines with the "-rpath" like load flag, it may not be necessary to set the `LD_LIBRARY_PATH` since this information is stored with the executable.

Here now are the steps for installation:

1. `tar xzf Overture.xxx.tar`
2. `cd Overture.xxx`
3. `configure` : (or `configure [option1] [option2] ...`) (or type `configure --help`). If perl is in a funny place then type `perl configure`. The default is to build a debug version with double precision (recommended).
4. `make` : this will build Overture, takes about 1/2 hour for the default options, a couple of hours for an optimized version. You can also type "`make -j 2`" on machines with multiple processors in order to perform a parallel make.

If you would like to **test the distribution** type

```
check.p
```

from the main Overture directory to build a collection of grids and run a set of regression tests. You can also run these test separately (as well as other tests) by:

1. `cd sampleGrids` : goto the sample grids directory

2. `make` : will build a set of overlapping grids and check to see if they are correctly built. This is a good test that Overture is installed and working.

3. `cd ../primer`

4. `make` : to make the primer examples. Now run some of the primer examples, such as `mappedGridExample1` or `mappedGridExample2` etc.

5. `cd ../tests` : go to the tests directory which holds other tests

6. `make` : make the test routines. Type `checkop.p` to run a perl script to test the operators.

7. There are other demos and tests in the `sampleGrids`, `sampleMappings` and `examples` directories.

8.1 More comments on installing Overture

Overture uses the perl script `configure` to build configuration files. Type

```
configure --help
```

to see the configure options. If perl is not installed where I think it should be then you may have to type

```
perl configure --help
```

The default options are "precision=double" for double-precision and "debug" for compiling with debug information so that you can diagnose problems with a debugger. In most cases performance is not seriously degraded by the "debug" option although some programs such as the grid generator "ogen" will run quite a bit faster if you configure with the "opt" option for optimization:

```
configure opt
```

Generally the configure script will automatically determine the machine type. If not you may have to type

```
configure <machine-name> [options]
```

The configure script will read the file config/Makedefs.<machine-name> and build Makefile's in each directory. If you need to change the Makefile's for your setup then you should usually change the config/Makedefs.xxx file and re-configure, rather than changing all the Makefile's.

9 General comments

Here are some steps that I would take to track down bugs.

1. Do the A++ test codes run?
2. If you are having trouble with the graphics try building the `paperplane` example in the `Overture/tests` directory. This example tests whether OpenGL and Motif are working (and does not even link to any Overture libraries).
3. The perl script `Overture/check.p` will run a series of regression tests.
4. Some Overture test codes are in the directory Overture/tests. The perl script `Overture/tests/checkop.p` will run regression tests on the operators (these tests are also run by Overture/check.p).
5. Use gdb or your favourite debugger to see what the problem is. If you don't know how to use a debugger you've come to the wrong party.

10 Installing Overture on Mac OS/X

10.1 OS X 10.9 - Mavericks

Stage I: The default compiler for Mavericks is *clang* which replaces gcc and g++. DO NOT install gcc.

1. Install Xcode from the apple web site.
2. Install the command line tools in Xcode:

```
xcode-select --install
```

   This command will ask you whether you want to install the command lines tools. You will know if you have the command lines tools if you can issue the command `gcc --version`.

3. Install gfortran from `http://gcc.gnu.org/wiki/GFortranBinaries#MacOS`
4. Install XQuartz (if you do not already have it). This should install X11 libraries in `/opt/X11/lib`. This is needed so you can pop up X windows but since there is no Motif here we will need to install another version using Macports.
5. Install Macports
6. Use Macports to install Open Motif and OpenGL:

```
sudo port install openmotif
sudo port install mesa
sudo port install libGLU
```

   This will install another version of X11 in `/opt/local/lib` in addition to the Open Motif and Mesa libraries.

Stage II:

1. Install HDF following the normal instructions.
2. Install A++ following the normal instructions but, you may need to make a symbolic link for gmake (somewhere in your path)

```
ln -s /usr/bin/make /home/dilbert/bin/gmake
```

   and configure with

```
./configure  --disable-SHARED_LIBS  --prefix=‘pwd‘
```

3. Install Overture and cg following the regular instructions and using

```
setenv XLIBS /opt/local
setenv MOTIF /opt/local
setenv OpenGL /opt/local
```

   as the locations for X11, Motif and OpenGL (do NOT accidently link to /opt/X11).

4. You should add the Overture/lib directory to your dynamic library path: `setenv DYLD_LIBRARY_PATH $DYLD_LIBRARY_PATH:$Overture/lib`

10.2 Installing Overture on Mac OS/X – *OLDER VERSIONS*

Overture and cg have been build on Mac OS/X. Thanks to Kyle for working this out. The installation process is very similar to the process for Linux after some initial steps to install the appropriate compilers etc..

Stage I:

1. Install and update the Apple developer tools (an extra DVD that came with the mac).
2. Install macports, see, http://www.macports.org/
3. Using macports, install hdf, mesa, glw, petsc, etc using:
   (a) port install gcc4.3 hdf5 petsc mesa glw motif

   Here we have installed version 4.3 of gcc.
4. It is convenient to make soft links for the 4.3 compilers that macports installs. Go to the MacPorts bin directory and make links like
   (a) `ln -s gcc-mp-4.3 gcc`

   Do this for gcc, g++ and gfortran.

Stage II: Install A++P++:

1. Download A++P++, Overture and cg from the Overture web site.
   (a) Unpack A++: `tar xzf AP-nnn.tar.gz`
   (b) Rename A++ directory if desired: `mv A++P++-nnn A++P++-nnn-gcc4.3`
   (c) `setenv CC gcc; setenv CXX g++; setenv FC gfortran`
   (d) `./configure --with-CC="$CC -I/usr/include/malloc -mlongcall" --with-CXX="$CXX -I/usr/include/malloc -mlongcall" --disable-SHARED_LIBS --prefix=‘pwd‘`
   (e) `make MAKE=make install`
   (f) `setenv APlusPlus ‘pwd‘/A++/install; setenv PPlusPlus $APlusPlus`

Stage III: Install Overture and CG: follow the instructions in Sections 8 and 14 for installing Overture and CG.
Notes:

1. You should add the Overture/lib directory to your dynamic library path: `setenv DYLD_LIBRARY_PATH $DYLD_LIBRARY_PATH:$Overture/lib`

11 Installing Overture on Ubuntu

Here are instructions for Ubuntu version 18.04.

(I) Sudo install the following to get Ubuntu setup.

```
apt install build-essential
apt install build-essential
apt install gfortran
apt install autoconf
apt install mpich
apt install emacs    (optional)
apt install libperl-dev
```

(IIa) If you do not build your own version of OpenGL Mesa (instructions below), then install OpenGL: (libjpeg62-dev may be needed even if you do install Mesa)

```
apt install libmotif-dev
apt install libgl1-mesa-dev
apt install libglu1-mesa libxi-dev libxmu-dev libglu1-mesa-dev
apt install libjpeg62-dev
```

(IIb) Optionally install Mesa OpenGL for better hardcopies. Get MesaLib-7.2.tar.gz from OvertureFramework install page

```
tar xzf MesaLib-7.2.tar.gz
tar xzf MesaDemos-7.2.tar.gz
cd Mesa-7.2/
make linux-x86-64   (note: Error building demos, but lib's built OK)

--- REPLACE THE GL Header files in /usr/include with the MESA versions
su -c "cp /home/henshw/software/Mesa-7.2/include/GLView.h /usr/include/GLView.h"
su -c "cp -r /home/henshw/software/Mesa-7.2/include/GL /usr/include/GL"
```

(III) Install HDF (serial version) from hdf5-1.8.15-patch1.tar.gz – follow normal instructions:

```
tar xzf hdf5-1.8.15-patch1.tar.gz
mv hdf5-1.8.15-patch1 hdf5-1.8.15-gcc7.4.0

cd hdf5-1.8.15-gcc7.4.0
unsetenv cc
setenv CC gcc
./configure --prefix=‘pwd‘
make
make install
```

(IV) Install A++ following normal instructions (get AP-0.8.2.tar.gz from overtureFramework.org)

```
su -c "ln -s /usr/bin/make /usr/bin/gmake"

tar -xzf AP-0.8.2.tar.gz
mv A++P++-0.8.2 AppPpp-0.8.2-gcc7.4.0
cd AppPpp-0.8.2-gcc7.4.0
configure --enable-SHARED_LIBS --prefix=‘pwd‘
make -j24
make install
```

(V) Optionally install PETSc 3.4.5 (linear solvers for cgins, for example)

```
tar xzf petsc-3.4.5.tar.gz
mv petsc-3.4.5 petsc-3.4.5-serial
setenv PETSC_DIR /home/henshw/software/petsc-3.4.5-serial
cd petsc-3.4.5-serial
configure --with-cc=gcc --with-fc=gfortran --PETSC_ARCH=linux-gnu-opt --with-debugging=0 \
          --with-fortran=0 --with-matlab=0 --with-mpi=0 --with-shared-libraries=1
make
```

(VI) Install Overture from sourceforge: – if you have an account on sourceforge you can download the latest Overture using

```
git clone ssh://USERNAME@git.code.sf.net/p/overtureframework/code /home/USERNAME/overtureFramework
```

Goto overtureFramework/Overture and read the README file there.

```
cd /home/USERNAME/overtureFramework/Overture
# Location for compiled version of Overture g=serial debug version :
setenv OvertureBuild /home/USERNAME/Overture.g
buildOverture
cd /home/USERNAME/Overture.g
-- set Overture enviromental variables found in Overture.g/defenv  ---
configure
make -j8
```

To build cg:

```
setenv CG /home/USERNAME/overtureFramework/cg  (location of cg source files)
setenv CGBUILDPREFIX /home/USERNAME/cg.g       (location for compiled version of cg)
cd /home/USERNAME/overtureFramework/cg
make
```

For the **parallel version**, install P++

```
tar xzf AP-0.8.2.tar.gz
mv A++P++-0.8.2 AppPpp-0.8.2-gcc7.4.0-parallel

cd AppPpp-0.8.2-gcc7.4.0-parallel
NOTE: mpi include directory:
configure --enable-PXX --prefix=‘pwd‘ --enable-SHARED_LIBS \
          --with-mpi-include=-I${MPI_ROOT}/include/mpich \
          --with-mpi-lib-dirs="-L${MPI_ROOT}/lib" --with-mpi-libs="-lmpich -lmpl -lc -lm" \
          --with-mpirun=${MPI_ROOT}/bin/mpiexec --without-PADRE --disable-mpirun-check
make -j24
make install
```

Parallel hdf5:

```
tar xzf hdf5-1.8.15-patch1.tar.gz
mv hdf5-1.8.15-patch1 hdf5-1.8.15-gcc7.4.0-parallel

cd hdf5-1.8.15-gcc7.4.0-parallel
unsetenv CC
unsetenv cc
setenv CC $MPI_ROOT/bin/mpicc
 ./configure --prefix=‘pwd‘ --enable-parallel
make
make install
```

To build parallel Overture

```
cd /home/USERNAME/overtureFramework/Overture
# Location for compiled version of Overture p=parallel debug version :
setenv OvertureBuild /home/USERNAME/Overture.p
buildOverture
cd /home/USERNAME/Overture.p
-- set Overture enviromental variables found in Overture.g/defenv  ---
configure parallel
make -j8
```

To build parallel cg:

```
setenv CG /home/USERNAME/overtureFramework/cg  (location of cg source files)
setenv CGBUILDPREFIX /home/USERNAME/cg.p       (location for compiled version of cg)
cd /home/USERNAME/overtureFramework/cg
make
```

12 Installing Overture on the Blue Gene

Here are notes on installing Overture on a Blue Gene/Q at RPI.

I used the gnu compilers:

```
module load gnu
```

12.1 Installing parallel HDF

Set `RUNSERIAL` to be the command to run a program on the Blue Gene.

```
tar -xzf hdf5-1.8.8.tar.gz
mv hdf5-1.8.8 hdf5-1.8.8-parallel
cd hdf5-1.8.8-parallel

unsetenv CC
unsetenv cc

./bin/yodconfigure configure

setenv RUNSERIAL "srun -N1 -n1 --time 1 --partition debug"
setenv LDFLAGS "-dynamic"

configure --prefix=‘pwd‘ --enable-parallel


make
make install
```

12.2 Installing P++

```
tar xzf AP-0.8.2.tar.gz
mv A++P++-0.8.2 A++P++-0.8.2-gcc.4.4.6-parallel

./configure --enable-PXX --enable-SHARED_LIBS --prefix=‘pwd‘ \
             --with-CC=mpicc --with-CXX=mpicxx --disable-mpirun-check --without-PADRE
```

12.3 Installing Overture

Overture was built without graphics and without support for Perl.

```
configure bg  parallel CC=mpicxx bCC=g++ cc=mpicc bcc=gcc FC=mpif90 bFC=gfortran\
          --disable-X11 --disable-perl --disable-gl
```

13 Installing LAPACK

CG needs the LAPACK and BLAS libraries. If you need to build these libraries you should find a version `lapack.tgz` from the web.
Here are the steps for installation:

1. `tar xzf lapack.tgz` : Untar the lapack library.
2. `cd LAPACK`
3. Copy the appropriate `make.inc.XXX` file from the INSTALL directory into `make.inc` in the main directory. Example: `cp INSTALL/make.inc.LINUX make.inc`
4. Edit the `make.inc` file and define the compiler and compiler options.
5. Type `make blaslib` to make BLAS.
6. Type `make lapacklib` to make LAPACK.
7. CG expects the libraries to be named `liblapack.a` and `libblas.a` so you may need to rename the libraries that are built in the main directory.

14 Installing CG

Here are instructions to install the CG solvers. CG uses the same enviromental variables as Overture. There is no need to configure CG since it finds all the information it needs from Overture. See the cg `Readme` file for more information.
Here are the steps for installation:

1. `tar xzf cg.xxx.tar` : untar the cg files.
2. Define the CG and CGBUILDPREFIX environmental variables as described in Section 8.
3. `cd cg.xxx`
4. `make` : this will build the default CG solvers. You can also type "`make -j 2`" on machines with multiple processors in order to perform a parallel make. Sometimes the parallel make gets confused so you may have to type `make` again after the first one finishes.
5. Instead of making all the default CG solvers as above, you can also just build individual ones. If you only want to build cgins, for example, go to the `$CG/ins` directory and type `make`.
6. By default the Maxwell solver is not built. Go to the `$CG/mx` directory and type `make` if you want to use it.

Here are some tests you can run

1. From the main cg directory type `make check` to run regression tests for each solver.
2. To test just one solver, say cgins, go to the `$CG/ins/check` directory and type `make`.

If you want to change some CG files and rebuild a solver you can use the `$CG/user` directory to do this. See the `$CG/user/Readme` file for more information.

15 Notes for running Overture in parallel

The parallel version of Overture is in beta release to friendly users. Not everything works in parallel but many things do.

To configure and build the parallel version (you need to install P++ first):

1. `setenv PPlusPlus xxxx` where xxxx is the location of the P++ installation.
2. `setenv APlusPlus xxxx` where xxxx is ALSO the location of the P++ installation.
3. `configure parallel`
4. `make`

General notes:

- You will need to have both a serial and parallel version of Overture built.
- It is safer to use the serial version of plotStuff to plot grids and plot results in show files, although some of the graphics, such as plotting grids and 2d/3d contours, should work in parallel.
- Build grids with the serial version of ogen unless you have to build a very large grid. (The parallel version of ogen works in many cases but is still under development).
- Parallel moving grids work (but not yet parallel deforming grids).
- Parallel AMR works (but not yet parallel AMR and parallel moving grids).
- Some Mapping's don't work in parallel since I haven't gotten around to writing the new mapS and inverseMapS functions that take serial arrays as arguments – these are usually easy to write.
- Generally it is best to convert some Mapping's (e.g. DataPointMapping, HyperbolicMapping) into NurbsMappings for use with parallel (see, for example, `Overture/sampleGrids/loftedBox.cmd`).

PDE solvers that should run in parallel:

- The primer example pwave is a good example of a small parallel code. Many of the other primer examples may run in parallel but are generally not good examples of how to write efficient parallel code.
- The cg solvers cgcns, cgins, cgad, cgmx, cgsm, and cgmp run in parallel but not all options will work. cgasf does not run in parallel yet.

Developer notes: If you are writing Overture code to run in parallel here are some notes:

- Avoid scalar indexing of a parallel array, $u(i1, i2, i3)$ since this will use communication by default. Instead, access the local serial array (see examples in pwave.C for example).
- Avoid P++ array operations except for simple copies. It is more efficient and robust to grab the serial arrays, operate on the serial arrays and then call updateGhostBoundaries.
- There are a variety of parallel utility routines for dealing with parallel array operations, grid functions and parallel copies etc. See the files `App.{h,C}` and `ParallelUtility.{h,C}`.