HiFi/SEL Libraries

From PSI-Wiki

Table of contents

Howto: Installing Prerequisite Libraries for HiFi/SEL

MPI

This example shows how to configure and install the mpich2 version of the MPI libraries. Other versions are available, and they could potentially already be installed on the system. The source code for mpich2 at http://www.mcs.anl.gov/research/projects/mpich2/downloads/index.php?s=downloads. Unpack the archive into a directory where libraries are installed (e.g. `/opt`).

```
$ tar -xvzf mpich2-1.0.8p1.tar.gz
```

The compilers need to be specified. In bash they are set with:

```
export CC=icc
export FC=ifort
```

Now change into the unpacked directory and configure with

```
$ ./configure --prefix=/opt/mpich2
```

where the prefix is location where the libraries are to be installed. Once the configuration is complete mpich2 is compiled with:

```
make all
```

and then installed with:

```
make install
```

If these steps complete sucessfully then the library is installed!

Alternatively one could have PETSc automatically configure and install mpich2 by adding the configuration option

```
--download-mpich2=yes
```

to the configuration script that is described below in the PETSc section.

PETSc

First download the source code at: http://www.mcs.anl.gov/petsc/petsc-2/download/index.html and save the tar.gz file to the location where you want to install PETSc (e.g. `/opt`). Change into the directory where the file was saved and extract the archive by running the commands:

```
cd /opt
tar -xvzf petsc-lite-3.0.0-p6.tar.gz
```

This makes a folder called petsc-3.0.0-p6. Next change into that directory:

```
cd /opt/petsc-3.0.0-p6
```

From here PETSc will be configured and compiled. There are many possible options for configuring PETSc and including other libraries such as SuperLU, MPICH, ParMetis, etc. If these libraries are already installed they can be linked to PETSc, or alternatively PETSc can download, configure, and compile them automatically. Before configuring two environmental variables must be specified:

```
export PETSC_ARCH=linux-intel-opt
export PETSC_DIR=/opt/petsc-3.0.0-p6
```

The `PETSC_DIR` variable specifies where PETSc resides, and should be where it was extracted. The second variable `PETSC_ARCH` specifies a name for the configuration (this is free form and can be set to whatever is preferred). In this case it is called `linux-intel-opt` because it will be configured and compiled on a linux machine, with the intel compilers, and with debugging optimization flags. Multiple configurations can be made, but be careful to rename the `PETSC_ARCH` environmental variable between each configuration and compiling of PETSc.

In order to simplify the configuration of PETSc a script can be created. Alternatively one can run the command the script specifies straight from the command line, but having the script in a file keeps a record as well as makes it easier to re-run in the future.

A file called `petsc_configure` is created with these contents:

```
export PETSC_DIR=$PWD
./config/configure.py \
--with-gnu-compilers=0 \
--with-mpi-compilers=0 \
--with-vendor-compilers=intel \
--with-debugging=0 \
--with-shared=0 \
--with-blas-lapack-dir=/opt/intel/mkl/10.1.2.024/lib/em64t \
--with-mpi-dir=/opt/mpich2 \
--download-superlu_dist=1 \
--download-parmetis=1 \
--download-scalapack=1 \
--download-mumps=1 \
--download-blacs=1
```

This particular script uses the intel compilers, has optimized compiler flags (no debugging), specifies the installed intel MKL `blas-lapack` libraries for the system, and specifies the installed mpi libraries (/opt/mpich2). It also tells the configuration to download and install `superlu_dist`, `parmetis`, `scalapack`, `mumpsM`, and the `blacs` libraries. Any of these downloaded libraries are optional, but HiFi/SEL does has the ability to use them.

Other systems might have an MPI version already installed and could potentially have the include and library files in different locations. For instance on the ICE cluster they are located in seperate locations and can be specified in the configuration options with:

```
--with-mpi-include=/usr/include
--with-mpi-lib=/usr/lib64/libmpi.so
```

Once the script is complete, it is invoked by:

```
$ ./petsc_configure
```

Some output is generated as during the configuration:

```
================================================================================
             Configuring PETSc to compile on your system
================================================================================
================================================================================
WARNING! Compiling PETSc with no debugging, this should
only be done for timing and production runs. All development should
be done when configured using --with-debugging=1
================================================================================
================================================================================
Compiling Blacs; this may take several minutes
================================================================================
================================================================================
Compiling Scalapack; this may take several minutes
================================================================================
================================================================================
Compiling & installing Parmetis; this may take several minutes
================================================================================
================================================================================
Compiling Mumps; this may take several minutes
================================================================================
================================================================================
Compiling superlu_dist; this may take several minutes
================================================================================
TESTING: alternateConfigureLibrary from PETSc.packages.petsc4py(config/PETSc/packages /petsc4py.py:69)
Compilers:
  C Compiler:         icc  -O
  Fortran Compiler:   ifort  -O
Linkers:
  Static linker:   /usr/bin/ar cr
PETSc:
  PETSC_ARCH: linux-intel-opt
  PETSC_DIR: /opt/petsc-3.0.0-p6
  **
  ** Now build the libraries with "make all"
  **
  Clanguage: C
  Scalar type:real
  PETSc shared libraries: disabled
  PETSc dynamic libraries: disabled
MPI:
  Includes: -I/opt/mpich2/include
  Library:  -Wl,-rpath,/opt/mpich2/lib -L/opt/mpich2/lib -lfmpich -lmpich -lpmpich -lnsl -laio -lrt
X11:
  Includes: [']
  Library: ['-lX11']
ParMetis:
  Includes: -I/opt/petsc-3.0.0-p6/linux-intel-opt/include
  Library:  -Wl,-rpath,/opt/petsc-3.0.0-p6/linux-intel-opt/lib -L/opt/petsc-3.0.0-p6/linux-intel-opt/lib
BLAS/LAPACK: -Wl,-rpath,/opt/intel/mkl/10.1.2.024/lib/em64t -L/opt/intel/mkl/10.1.2.024/lib/em64t -lmkl_l
  blacs:
  Library:  -Wl,-rpath,/opt/petsc-3.0.0-p6/linux-intel-opt/lib -L/opt/petsc-3.0.0-p6/linux-intel-opt/lib
SuperLU_DIST:
  Includes: -I/opt/petsc-3.0.0-p6/linux-intel-opt/include
  Library:  -Wl,-rpath,/opt/petsc-3.0.0-p6/linux-intel-opt/lib -L/opt/petsc-3.0.0-p6/linux-intel-opt/lib
SCALAPACK:
  Library:  -Wl,-rpath,/opt/petsc-3.0.0-p6/linux-intel-opt/lib -L/opt/petsc-3.0.0-p6/linux-intel-opt/lib
MUMPS:
  Includes: -I/opt/petsc-3.0.0-p6/linux-intel-opt/include
  Library:  -Wl,-rpath,/opt/petsc-3.0.0-p6/linux-intel-opt/lib -L/opt/petsc-3.0.0-p6/linux-intel-opt/lib
```

If the configuration is sucessful one should see a summary like shown above showing the different include and library paths. Now PETSc needs to be compiled with:

```
make all
```

This produces a bunch of output to the screen showing which file it's compiling. When it is done its displays:

```
Completed building libraries
=========================================
Now to check if the libraries are working do: make test
=========================================
```

The last step is to test the compiled PETSc library with:

```
make test
```

and one should see the following output:

```
Running test examples to verify correct installation
C/C++ example src/snes/examples/tutorials/ex19 run successfully with 1 MPI process
C/C++ example src/snes/examples/tutorials/ex19 run successfully with 2 MPI processes
Graphics example src/snes/examples/tutorials/ex19 run successfully with 1 MPI process
Fortran example src/snes/examples/tutorials/ex5f run successfully with 1 MPI process
Completed test examples
```

If it completes sucessfully you're done!

This above example was for the optimized libraries. It usually a good idea to also compile with debug flags. To do this change the `PETSC_ARCH` environmental variable to something that denotes debug options:

```
export PETSC_ARCH=linux-intel-debug
```

Then in the script turn on the debugging option with:

```
--with-debugging=1
```

and rerun the script, `make all`, and `make test` just like in the other case. Now in the HiFi/SEL makefile one specifies either `PETSC_ARCH` that was created.

SLEPc

The SLEPc configuration is very simple once PETSc is installed, because it sits on top of any PETSc installation. Download the .tgz file from http://www.grycap.upv.es/slepc/download/download.htm and unpack in where other software is installed (e.g. `/opt`). After unpacking change into the SLEPc directory:

```
cd /opt/slepc-3.0.0-p4
```

The PETSc environmental variables need to be specified (`PETSC_DIR`, `PETSC_ARCH`, as well as one SLEPc variable, `SLEPC_DIR`.

```
export PETSC_DIR=/opt/petsc-3.0.0-p6
export PETSC_ARCH=linux-intel-opt
export SLEPC_DIR=/opt/slepc-3.0.0-p4
```

The type:

```
./configure
```

The output to screen looks like:

```
Checking environment...
Checking PETSc installation...
Checking LAPACK library...
===================================================================================
SLEPc Configuration
===================================================================================
```

```
SLEPc source directory:
 /opt/slepc-3.0.0-p4
SLEPc install directory:
 /opt/slepc-3.0.0-p4/linux-intel-opt
PETSc directory:
 /opt/petsc-3.0.0-p6
Architecture "linux-intel-opt" with double precision real numbers
```

Compile SLEPc with:

```
make all
```

it should generate some output to screen and end with:

```
Completed building SLEPc libraries
=========================================
Now to check if the libraries are working do: make test
=========================================
```

Run

```
make test
```

and if it's sucessful it says:

```
Running test examples to verify correct installation
C/C++ example src/examples/ex1 run successfully with 1 MPI process
Fortran example src/examples/ex1f run successfully with 1 MPI process
Completed test examples
```

Then it's done and ready to be used. The configure and compile should be done again after changing `PETSC_ARCH=linux-intel-debug`.

HDF5

The HDF5 libraries are used in both HiFi/SEL as well as the post processing code. HiFi/SEL uses a parallel

instalation, but can be run in either serial or parallel mode. The post processing for SEL only uses serial HDF5, while HiFi post processing has two versions, one serial, one parallel. Because of the varying types of uses, it is advised to install both a serial and parallel version of the HDF5 libraries (They need to be configured and compiled seperately).

Both are made from the same source code, which can be downloaded at http://www.hdfgroup.org/ftp/HDF5 /current/src/ for the latest 1.8.X branch, and at http://www.hdfgroup.org/ftp/HDF5/current16/src/ for the older 1.6.X. The 1.8.X branch should work with most machines. Unpack the source code in a spot where other software is installed (e.g. `/opt`)

Serial HDF5

Serial HDF5 is a fairly simple configuration procedure. The compilers (and any extra compiler flags the user might want) are set. The compilers should be the same ones used to compile other libraries and the HiFi/SEL code. In bash to set the C and Fortran compilers:

```
export CC=icc
export FC=ifort
```

Since HiFi/SEL is written in Fortran 90/95 support for fortran must be turned on during the configuration. The other configuration option is to set the installation location, using a `prefix`. This command runs the configuration when inside the hdf5 source directory:

```
./configure --enable-fortran --prefix=/opt/hdf5-serial
```

After the configuration completes sucessfully a summary is given. Here's an example:

```
           SUMMARY OF THE HDF5 CONFIGURATION
           =================================
General Information:
-------------------
                   HDF5 Version: 1.8.3
                  Configured on: Wed Jun 17 17:35:14 PDT 2009
                  Configured by: wlowrie@feynman
                 Configure mode: production
                    Host system: x86_64-unknown-linux-gnu
              Uname information: Linux feynman 2.6.20-gentoo-r8 #2 SMP PREEMPT Fri Jul 20 19:12:30 PDT 2
                                  Intel(R) Xeon(R) CPU 5130 @ 2.00GHz GenuineIntel GNU/Linux
                        Byte sex: little-endian
                      Libraries:
             Installation point: /opt/hdf5-serial
Compiling Options:
------------------
               Compilation Mode: production
                     C Compiler: /opt/intel/cce/10.1.022/bin/icc (icc-10.1)
                CFLAGS/H5_CFLAGS: /-std=c99  -O2 -Wl,-s
          CPPFLAGS/H5_CPPFLAGS: -D_LARGEFILE_SOURCE -D_LARGEFILE64_SOURCE -D_FILE_OFFSET_BITS=64 -D_POSI
                                -DNDEBUG -UH5_DEBUG_API
              Shared Libraries: yes
              Static Libraries: yes
  Statically Linked Executables: no
                 Extra libraries:   -lz -lm
                       Archiver: ar
                         Ranlib: ranlib
             Debugged Packages:
                   API Tracing: no
Languages:
----------
                        Fortran: yes
                Fortran Compiler: ifort
                   Fortran Flags: -I.   -fPIC
                             C++: no
Features:
---------
                  Parallel HDF5: no
             High Level library: yes
                   Threadsafety: no
            Default API Mapping: v18
 With Deprecated Public Symbols: yes
         I/O filters (external): deflate(zlib)
         I/O filters (internal): shuffle,fletcher32,nbit,scaleoffset
                            MPE: no
                     Direct VFD: no
                        dmalloc: no
Clear file buffers before write: yes
           Using memory checker: no
         Function Stack Tracing: no
                           GPFS: no
     Strict File Format Checks: no
   Optimization Instrumentation: no
 Linux Large File Support (LFS): yes
```

After completion HDF5 is compiled with:

```
make all
```

when the code has finished compiling

```
make install
```

compies the binary files, include files, and library files to the directory specified by the `--prefix=` configuration option. Now in the makefile this location can be specified for the HDF5 libraries. The libraries can optionally be tested by running the command:

```
make test
```

This can take some time to complete.

Parallel HDF5

Parallel HDF5 is slightly more complicated and there is more than one way to configure the software. It makes use of the MPI commands, so it must be installed before configuration. Depending on the MPI type, parallel compilers may or may not be available. In the case of MPICH2 usually they are created. If one is using the fortran compilers, `mpicc` and `mpif90` would reside in a path like `/opt/mpich2/bin/`. If this is the case one can specify these compilers with:

```
export CC=/opt/mpich2/bin/mpicc
export FC=/opt/mpich2/bin/mpif90
```

Keep in mind these compilers are basically wrappers of the original compilers (e.g. icc, ifort) and they should be consistent with the compilers that will be used to compile the main HiFi/SEL code.

If these parallel compilers are not available for your version of MPI, then the serial compilers can be specified with some additional compiler flags. For the case of intel compilers, these are set with:

```
export CC=icc
export FC=ifort
```

and the compiler flags need to reference the path to the include and lib directories of the MPI installation:

```
export CPPFLAGS=-I/opt/mpich2/include
export LDFLAGS=-I/opt/mpich2/lib
```

Now HDF5 is ready to be configured with parallel option:

```
./configure --enable-fortran --prefix=/opt/hdf5-parallel --enable-parallel
```

The configuration summary should be similar to the serial summary except is should say:

```
Parallel HDF5: yes
```

and the compilers should be listed accordingly:

```
     C Compiler: /opt/mpich2/bin/mpicc (icc-10.1)
Fortran Compiler: /opt/mpich2/bin/mpif90
```

After sucessful configuration the code is compiled with (if a previous configuration was compiled make sure to run '`make clean`' before recompiling):

```
make all
```

and then installed to the prefix directory with:

```
make install
```

netCDF

The current netCDF source code can be downloaded at http://www.unidata.ucar.edu/downloads/netcdf/index.jsp . The libraries needed are 'The netCDF C, C++, and Fortan libraries, ncgen/ncdump'. Either version 4.0.X or 3.6.3 is known to work with HiFi/SEL. Version 4.0.X should work with most systems. Unpack the downloaded file to a location where libraries are installed (e.g. `/opt`).

In the standard configuration there are no dependencies to netCDF so configuration involves setting the compilers with:

```
export CC=icc
export FC=ifort
```

and then running the configuration script within the netcdf source code directory:

```
./configure --prefix=/opt/netcdf
```

The prefix is optional since only one configuration is necessary. Once the configuration is complete compile the code with:

```
make all
```

and then if a prefix was specified:

```
make install
```

Thats all for this library!