The Center for Astrophysical Thermonuclear Flashes

Makefiles and Libraries

Flash Tutorial

June 23, 2009

Paul Rich and Shawn Needham

An Advanced Simulation & Computing (ASC)
Academic Strategic Alliances Program (ASAP) Center
at The University of Chicago

Needed Libraries and Software

- Fortran Compiler
- MPI
- IO - HDF5 - PnetCDF
- Python
- GNU-compatible make
  - Makefile concatenation is used extensively by FLASH's setup script.

Example Makefile.h

Located in FLASH3/sites/<machine name>

```
MPI_PATH   = /usr/local/mpich-intel
HDF5_PATH  = /usr/local/hdf5-icc
NCMPI_PATH = /usr/local/pnetcdf-icc

FCOMP   = ${MPI_PATH}/bin/mpif90
CCOMP   = ${MPI_PATH}/bin/mpicc
CPPCOMP = ${MPI_PATH}/bin/mpiCC
LINK    = ${MPI_PATH}/bin/mpif90 FFLAGS_OPT   = -c -r8 -i4 -O3 -real_size 64# -unroll -align -prefetch -pad -ip

FFLAGS_DEBUG = -c -g -r8 -i4 -check bounds -check format -check output_conversion -warn all -real_size 64
FFLAGS_TEST  = -c -r8 -i4 -O2 -real_size 64

CFLAGS_OPT   = -c -O3 -D_LARGEFILE64_SOURCE
CFLAGS_DEBUG = -c -g -debug extended -D_LARGEFILE64_SOURCE
CFLAGS_TEST  = -c -O2 -D_LARGEFILE64_SOURCE

CFLAGS_HDF5  = -I $(HDF5_PATH)/include
CFLAGS_NCMPI = -I $(NCMPI_PATH)/include
CFLAGS_MPI   = -I$(MPI_PATH)/include

LFLAGS_OPT   = -r8 -i4 -Vaxlib -lsvml -Ur -o
LFLAGS_DEBUG = -r8 -i4 -Vaxlib -g -o
LFLAGS_TEST  = -r8 -i4 -Vaxlib -o

LIB_HDF5 = -L $(HDF5_PATH)/lib -lhdf5 -lz
LIB_MPI     = -L$(MPI_PATH)/lib -lfmpich -lmpich
LIB_NCMPI = -L$(NCMPI_PATH)/lib -lpnetcdf
```

Makefiles

- Useful examples can be found in sites/Prototypes
  - The compiler options set in these files are a good starting point
- Can be specified using the -site flag and the -makefile flag in the setup script
- When setting flags, make sure to consult the compiler documentation
- Make sure to verify that more aggressive optimizations do not impact code accuracy too greatly
- If possible, use the MPI commands (mpicc, mpif90, etc) as your compiler settings
  - Helps prevent library linking issues

MPI

- FLASH has traditionally been built using MPI 1 features.
- MPI 2 support is required, however, for most parallel I/O support, as the libraries typically utilize MPI2's collective IO operations
- Usually can go with a default build, so long as Fortran support is compiled in.

HDF5

- FLASH uses the HDF5 version 1.6 bindings.
- Version 1.8 can be used if built with support for the 1.6 bindings
- When building the code, you can access the 1.6 bindings by passing the H5_USE_16_API preprocessor macro through the compiler
- We have had problems with using IDL's HDF5 reader with the 1.8 libraries
- We usually pass the following flags:

If 1.6.x

./configure --prefix=/usr/local/hdf5-loc --enable-production --enable-parallel

if 1.8.x:

./configure --prefix=/usr/local/hdf5-loc --enable-production
--enable-parallel --with-default-api-version=v16

PnetCDF

- Version 1.0.3 corrects an issue we've seen with x86-64 platforms.
- Version 1.1.0.pre1 has a fix for large file support that is presently undergoing further testing.

Library general tips

- Make sure your MPI & IO library software stack is built with the same compiler!
  - This is particularly true of Intel and Portland Group binaries
  - It is easiest to build MPI first, and add your freshly built MPI bin to your PATH and set you compiler variables to use the MPI commands

ie: Building HDF5 against a freshly built Intel based MPI

- setenv PATH /usr/local/mpich-1.2.7p1/intel/bin:${PATH}
- which mpicc

/usr/local/mpich-1.2.7p1/intel/bin/mpicc

> setenv CC mpicc

> ./configure –prefix=/usr/local/hdf5-1.6.5/intel ...

Other Software

- Must use at least version 2.3 of Python for setup script.
- IDL version 6 or later required for fidlr3.0 and xflash3.
- Visit 1.10 or comes with a reader for FLASH 3.0 files, when invoking, use -assume_format FLASH