Computational Chemistry users may also consult the EGI.EU Application Database

Here is presented an overview of the Computational Chemistry applications currently ported in the GRID environment .

Installation and porting guides

The following best practices document provides some hints and examples on how to configure and compile some Computational Chemistry related applications on a grid based infrastructure.

DL_POLY

Application description

DL_POLY is a package of subroutines, programs and data files, designed to facilitate molecular dynamics simulations of macromolecules, polymers, ionic systems, solutions and other molecular systems on a distributed memory parallel computer. The package was written to support the UK project **CCP5** by Bill Smith and Tim Forester under grants from the Engineering and Physical Sciences Research Council and is the property of the Science and Technology Facilities Council (STFC). Two forms of DL_POLY exist. DL_POLY_2 is the earlier version and is based on a replicated data parallelism. It is suitable for simulations of up to 30,000 atoms on up to 100 processors. DL_POLY_3 is a domain decomposition version, written by I.T. Todorov and W. Smith, and is designed for systems beyond the range of DL_POLY_2 - up to 10,000,000 atoms (and beyond) and 1000 processors.

- Scientific Contact: W. Smith, CSE Department, STFC Daresbury Laboratory, UK
- Web Site
- VO using DL_POLY: COMPCHEM

DL_POLY 2.20

Sequential executable

To compile it, it's required :

1. a **FORTRAN90** compliant compiler (if the PATH to it is not passed to the DEFAULT ENVIRONMENT PATH, then it MUST be supplied in Makefile). Note: Compiler used: native gfortran; composerxe-2011.5.220.
2. a **MAKE** command (Makefile interpreter in the system SHELL). Note: native make (bash shell) used.

Contact your System Admin if the needed software is missing or not available.

A. Download or copy the tar file of DL_POLY_2.20 MD package in a machine with the gLite3.2 middleware installed, and untar the package in an appropriate sub-directory. Copy the file named *MakeSEQ* and stored in the *build* directory into the *srcmod* directory

```
# cp build/MakeSEQ srcmod/Makefile
```

. The file enable to compile the source code to obtain the sequential version of the executable.

B. Edit the *Makefile* as follow

- set ***EX*** variable to chose the appropriate name for your executable.

Using *gfortran* compiler - the architecture is already set in the *MakeSEQ* file

- add - static to the the ***LDFLAGS*** variable under the *gfortran* target architecture:

```
LDFLAGS="-static"
```

Using *ifort* compiler

- add the specific target architecture

```
#======== ifort (serial) ======================================
ifort:
        $(MAKE) LD="ifort -o " LDFLAGS="-static" FC=ifort \
        FFLAGS="-c -O2" \
        EX=$(EX) BINROOT=$(BINROOT) $(TYPE)
```

C. Compile the source code

```
 # make <target_architecture >
```

Note: for other architectures, please refer to the appropriate OS user guide or contact the System Admin.

After the compile procedure you should find into the *executable* directory the *DL_POLY* executable. To be sure that the executable is statically linked, run the following command

```
 # ldd <executable_name >
```

" *not a dynamic executable* " should be visualized.

You can now use the executable and submit it to the GRID environment.

Parallel executable

It's needed

1. a **FORTRAN90 compliant compiler** (if the PATH to it is not passed to the DEFAULT ENVIRONMENT PATH, then it MUST be supplied in Makefile). Note: Compiler used: native gfortran; composerxe-2011.5.220.
2. **MPI libraries COMPLIANT with the architecture and the compiler to be used** (if the PATH to them is not passed to the DEFAULT ENVIRONMENT PATH, then it MUST be supplied in Makefile). Note: MPI library used: mpich2-1.4.1.
3. a **MAKE command** (Makefile interpreter in the system SHELL). Note: native make (bash shell) used

Contact your Admin if the needed software is missing or not available.

A. Download or copy the tar file of DL_POLY_2.20 MD package in a machine with the gLite3.2 middleware installed, and untar it in an appropriate sub-directory. Copy the file named *MakePAR* and stored in the *build* directory into the *srcmod* directory

```
 # cp build/MakePAR srcmod/Makefile
```

The file enable to compile the source code to obtain the parallel version of the executable

B. Edit the Makefile as follow

- set ***EX*** variable to chose the appropriate name for your executable.

Using *gfortran* compiler - the architecture is already set in the *MakePAR* file

- be sure that the *mpif90* compiler is set and uses *gfortran*. The following command may help you on that.

```
#which mpif90
```

  If not, replace the *mpif90* compiler in the ***LD*** and ***FC*** variables with the full path or contact the System Admin.

Using *ifort* compiler

- The version of the *ifort* compiler installed (composerxe-2011.5.220) comes with integrated mpi libraies. In this porting procedure we made use of the standard mpich2-1.4.1 library compiled with *ifort*. Be sure that the *mpif90* compiler is set and uses *ifort*. The following command may help you on that.

```
# which mpif90
```

  If not, replace the *mpif90* compiler in the ***LD*** and ***FC*** variables with the full path or contact the System Admin.

- add the specific target architecture

```
#======== ifort (parallel) ====================================
ifort:
        $(MAKE) LD=" mpif90 -o " LDFLAGS=" " FC=mpif90 \
        FFLAGS="-c -O2" \
        EX=$(EX) BINROOT=$(BINROOT) $(TYPE)
```

C. Compile the source code

```
 # make <target_architecture >
```

Note: for other architectures, please refer to the appropriate OS user guide or contact the System Admin.

After the compile procedure you should find into the *executable* directory the *DL_POLY* executable. In this case the executable is dynamically linked. To obtain an executable statically linked, add - static to the ***LDFLAGS*** variable under the gfortran target architecture:

```
 LDFLAGS="-static"
```

You can use the executable and submit it to the GRID environment.

DL_POLY 4.02

Sequential executable

Needed for compilation are:

1. a **FORTRAN90 compliant compiler** (if the PATH to it is not passed to the DEFAULT ENVIRONMENT PATH, then it MUST be supplied in Makefile). Note: Compiler used: native gfortran; composerxe-2011.5.220.
2. a **MAKE command** (Makefile interpreter in the system SHELL). Note: native make (bash shell) used.

Contact your System Admin if the needed software is missing or not available.

A. Download or copy the tar file of DL_POLY_4.02 MD package in a machine with the EMI middleware installed, and untar it in an appropriate sub-directory. Copy the file named *Makefile_SRL1* and stored in the *build* directory into the *source* directory

```
# cp build/Makefile_SRL1 srcmod/Makefile
```

The file enable to compile the source code to obtain the sequential version of the executable.

B. Edit the *Makefile* as follow

- set ***EX*** variable to chose the appropriate name for your executable.

Fill in the

```
Generic target template
```

as follow

- add - static to the the ***LDFLAGS*** variable:

```
LDFLAGS="-static"

ifort:
        $(MAKE) LD="ifort -o " LDFLAGS="-static" FC=ifort \
        FFLAGS="-c -O2" \
        EX=$(EX) BINROOT=$(BINROOT) $(TYPE)
```

C. Compile the source code

```
# make <target_architecture >
```

Note: for other architectures, please refer to the appropriate OS user guide or contact the System Admin.

After the compile procedure you should find into the *executable* directory the *DL_POLY* executable. To be sure that the executable is statically linked, run the following command

```
# ldd <executable_name >
```

" *not a dynamic executable* " should be visualized.

You can use the executable and submit it to the GRID environment.

Parallel executable

Needed for compilation are:

1. a **FORTRAN90 compliant compiler** (if the PATH to it is not passed to the DEFAULT ENVIRONMENT PATH, then it MUST be supplied in Makefile). Note: Compiler used: native gfortran; composerxe-2011.5.220.
2. **MPI libraries COMPLIANT with the architecture and the compiler to be used** (if the PATH to them is not passed to the DEFAULT ENVIRONMENT PATH, then it MUST be supplied in Makefile). Note: MPI library used: mpich2-1.4.1.
3. a **MAKE command** (Makefile interpreter in the system SHELL). Note: native make (bash shell) used

Contact your Admin if the needed software is missing or not available.

A. Download or copy the tar file of DL_POLY_4.02 MD package in a machine with the EMI1 middleware installed, and untar it in an appropriate sub-directory. Copy the file named *Makefile_MPI* and stored in the *build* directory into the *source* directory

```
# cp build/Makefile_MPI srcmod/Makefile
```

The file enable to compile the source code to obtain the parallel version of the executable

B. Edit the Makefile as follow

- set ***EX*** variable to chose the appropriate name for your executable.

Using *ifort* compiler

- The version of the *ifort* compiler installed (composerxe-2011.5.220) comes with integrated mpi libraies. In this porting procedure we made use of the standard mpich2-1.4.1 library compiled with *ifort*. Be sure that the *mpif90* compiler is set and uses *ifort*. The following command may help you on that.

  ```
  # which mpif90
  ```

  If not, replace the *mpif90* compiler in the ***LD*** and ***FC*** variables with the full path or contact the System Admin.

- add the specific target architecture

```
#======== ifort (parallel) ======================================
ifort:
        $(MAKE) LD=" mpif90 -o " LDFLAGS=" " FC=mpif90 \
        FFLAGS="-c -O2" \
        EX=$(EX) BINROOT=$(BINROOT) $(TYPE)
```

C. Compile the source code

```
# make <target_architecture >
```

Note: for other architectures, please refer to the appropriate OS user guide or contact the System Admin.

After the compile procedure you should find into the *executable* directory the *DL_POLY* executable. In this case the executable is dynamically linked. To obtain an executable statically linked, add - static to the ***LDFLAGS*** variable under the gfortran target architecture:

```
LDFLAGS="-static"
```

You can use the executable and submit it to the GRID environment.

GROMACS

Application description

GROMACS is a versatile package to perform molecular dynamics, i.e. simulate the Newtonian equations of motion for systems with hundreds to millions of particles. It is primarily designed for biochemical molecules like proteins, lipids and nucleic acids that have a lot of complicated bonded interactions, but since

GROMACS is extremely fast at calculating the nonbonded interactions (that usually dominate simulations) many groups are also using it for research on non-biological systems, e.g. polymers. GROMACS supports all the usual algorithms you expect from a modern molecular dynamics implementation, (check the online reference or manual for details).

- Web Site :
- VO using it: COMPCHEM

GROMACS 4.5.5

Sequential executable

To compile it, It is needed for compilation:

1. a **FORTRAN90 compliant compiler** (the PATH has to be passed setting the variable F77). Note: Compiler used: composerxe-2011.5.220.
2. a **MAKE command** (Makefile interpreter in the system SHELL). Note: native make (bash shell) used.
3. FFT libraries (see http://www.fftw.org/)

Contact your System Admin if the needed software is missing or not available.

A. Download or copy the tar file of gromacs-4.5.5.tar.gz MD package in a machine with the gLite3.2 middleware installed, untar it in an appropriate sub-directory. B. Set the following variables

```
# export CPPFLAGS=-I$FFTPATH/include

# export LDFLAGS=-L$FFTPATH/lib
```

C. Compile the source code in a X86_64 architecture

```
# ./configure --prefix=$GROMACSPATH/gromacs --disable-x86-64-sse --with-fft={fftw3,fftw2,mkl} --

# make

# make install
```

Note: for other architectures, please refer to the appropriate OS user guide or contact the System Admin.

After the compile procedure you should find into the *$GROMACSPATH/gromacs/bin* directory the *mdrun* executable. To be sure that the executable is statically linked, run the following command

```
# ldd mdrun
```

" *not a dynamic executable* " should be visualized.

You can use the executable and submit it to the GRID environment.

Parellel executable

Needed for compilation are:

1. a FORTRAN90 compliant compiler (the PATH has to be passed setting the variable F77). Note: Compiler used: composerxe-2011.5.220.
2. a MAKE command (Makefile interpreter in the system SHELL). Note: native make (bash shell) used.
3. FFT libraries (see http://www.fftw.org/)

4. MPI libraries

Contact your System Admin if the needed software is missing or not available.

A. Download or copy the tar file of gromacs-4.5.5.tar.gz MD package in a machine with the gLite3.2 middleware installed, untar it in an appropriate sub-directory. B. Set the following variables

```
# export CPPFLAGS=-I$FFTPATH/include $MPIPATH/include

# export LDFLAGS=-L$FFTPATH/lib $MPIPATH/lib
```

C. Compile the source code in a X86_64 architecture

```
# ./configure --prefix=$GROMACSPATH/gromacs --program-suffix=-mpi --disable-x86-64-sse --with-ff

# make

# make install
```

Note: for other architectures, please refer to the appropriate OS user guide or contact the System Admin.

After the compile procedure you should find into the *$GROMACSPATH/gromacs/bin* directory the *mdrun-mpi* executable. To be sure that the executable is statically linked, run the following command

```
# ldd mdrun-mpi
```

" *not a dynamic executable* " should be visualized.

You can use the executable and submit it to the GRID environment.

NAMD (2.9)

Application description

NAMD is a parallel, object-oriented molecular dynamics code designed for high-performance simulation of large biomolecular systems. NAMD is distributed free of charge and includes source code.

Parellel executable

Building a complete NAMD binary from source code requires working C and C++ compilers, Charm++/Converse, TCL, and FFTW. NAMD will compile without TCL or FFTW but certain features will be disabled.

A. Unpack NAMD and matching Charm++ source code and enter directory:

```
tar xzf NAMD_2.9_Source.tar.gz
cd NAMD_2.9_Source
tar xf charm-6.4.0.tar
cd charm-6.4.0
```

B. Build and test the Charm++/Converse library (multicore version):

```
./build charm++ mpi-linux-x86_64 --with-production
cd mpi-linux-x86_64/tests/charm++/megatest
make pgm
./pgm +p4   (multicore implementation does not support multiple nodes)
cd ../../../../..
```

C. Build and test the Charm++/Converse library (MPI version):

```
 env MPICXX=mpicxx ./build charm++ mpi-linux-x86_64 --with-production
  cd mpi-linux-x86_64/tests/charm++/megatest
  make pgm
  mpirun -n 4 ./pgm    (run as any other MPI program on your cluster)
  cd ../../../../..
```

D. Download and install TCL and FFTW libraries: (cd to NAMD_2.9_Source if you're not already there)

```
  wget http://www.ks.uiuc.edu/Research/namd/libraries/fftw-linux-x86_64.tar.gz
  tar xzf fftw-linux-x86_64.tar.gz
  mv linux-x86_64 fftw
  wget http://www.ks.uiuc.edu/Research/namd/libraries/tcl8.5.9-linux-x86_64.tar.gz
  wget http://www.ks.uiuc.edu/Research/namd/libraries/tcl8.5.9-linux-x86_64-threaded.tar.gz
  tar xzf tcl8.5.9-linux-x86_64.tar.gz
  tar xzf tcl8.5.9-linux-x86_64-threaded.tar.gz
  mv tcl8.5.9-linux-x86_64 tcl
  mv tcl8.5.9-linux-x86_64-threaded tcl-threaded
```

E. Edit configuration files as follow fill in the path of the needed libraries:

```
$ cat arch/Linux-x86_64-grid.arch
NAMD_ARCH = Linux-x86_64
CHARMARCH = mpi-linux-x86_64
CXX = /opt/openmpi-1.4.3-gfortran44/bin/mpic++  -m64 -O3
CXXOPTS = -fexpensive-optimizations -ffast-math
CC = /opt/openmpi-1.4.3-gfortran44/bin/mpicc -m64 -O3
COPTS = -fexpensive-optimizations -ffast-math
```

F. Set up build directory and compile: MPI version:

```
  ./config Linux-x86_64-grid --charm-arch mpi-linux-x86_64
  cd Linux-x86_64-grid
  make   (or gmake -j4, which should run faster)
```

G. Quick tests using one and two processes: (this is a 66-atom simulation so don't expect any speedup)

```
  ./namd2 src/alanin
```

(for MPI version, run namd2 binary as any other MPI executable)

Gaussian

To be completed by Daniele

CRYSTAL

To be completed by Alessandro

Tools

Links to GRIF and GCRES

-- DanieleCesini - 2012-11-16

This topic: UserSupport > CCPorting
Topic revision: r4 - 2013-11-05 - EmidioGiorgio