OpenFOAM 向神威平台的移植

（以 OpenFOAM-3.0.0 为例）

任虎 2018-09-10

1. 下载压缩包
   a) OpenFOAM-3.0.0.tgz
   b) ThirdParty-3.0.0.tgz
2. 解压压缩包至同一目录，这里用${FOAM_PRJ_DIR}表示
3. OpenFOAM 编译配置
   a) 进入 OpenFOAM 根目录${FOAM_PRJ_DIR}/OpenFOAM-3.0.0，以下用 ${FOAM_ROOT}表示
   b) 打开${FOAM_ROOT}/etc/bashrc，各行配置如下
      44：foamInstall=${ FOAM_PRJ_DIR } //设置根目录
      66：export WM_COMPILER=SWGCC453 //设置编译工具名称，本配置使用神威 gcc453 交叉编译器
      81：export WM_LABEL_SIZE=64 //使用 64 位整型
      90：export WM_MPLIB=SWMPI //设置 MPI 库名称，用于后续 MPI 设置
      其它设置如常，各选项意义参看该脚本注释
   c) 打开文件${FOAM_ROOT}/etc/config/settings.sh，配置如下
      90：export WM_CC='gcc' //用于编译 wmake 工具的 c 编译器，不能使用交叉编译
      91：export WM_CXX='g++' //用于编译 wmake 工具的 c++编译器，不能使用交叉编译
      484 行后添加 MPI 路径和库名如下
      ```
      SWMPI)
          export FOAM_MPI=mpi
          export MPI_ARCH_PATH=/usr/sw-mpp/mpi2
          ;;
      ```
      其它设置如常
   d) 打开文件${FOAM_ROOT}/etc/config/ CGAL.sh，配置如下
      35: #export BOOST_ARCH_PATH=$WM_THIRD_PARTY_DIR/platforms/WM_ARCHWM_COMPILER/$boost_version
      36: #export CGAL_ARCH_PATH=$WM_THIRD_PARTY_DIR/platforms/WM_ARCHWM_COMPILER/$cgal_version
   e) 打开文件${FOAM_ROOT}/etc/config/ metis.sh，配置如下
      36: #export METIS_VERSION=metis-5.1.0
      37: #export METIS_ARCH_PATH=$WM_THIRD_PARTY_DIR/platforms/WM_ARCHWM_COMPILER/$METIS_VERSION
   f) 进入${FOAM_ROOT}/wmake/rules，创建规则目录 linux64SWGCC453

g) 由于配置与 GCC 最为接近，拷贝${FOAM_ROOT}/rules/linuxGcc45/*至刚创建的规则目录

h) 进入新的规则目录

i) 对文件 c++配置如下

8：CC = swg++453

14：C++FLAGS = $(GFLAGS) $(c++WARN) $(c++OPT) $(c++DBUG) $(ptFLAGS) $(LIB_HEADER_DIRS) –fPIC –I${COMPILER_INCLUDE} （其中添加编译器的引用目录${COMPILER_INCLUDE}，因为 GCC453 编译器目前缺少自动的引用路径，目前为-I/usr/sw-mpp/swcc/swgcc453-binary/cross-tools/include -I${HOME}/include，其中${HOME}/include 中主要增加了 FlexLexer.h 头文件）

23：LINKLIBSO = $(CC) $(c++FLAGS) -static -Xlinker --add-needed -Xlinker --no-as-needed //神威平台只（比较好地）支持静态库。

24：LINKEXE = OFswld //自动连接工具，较好地支持静态库链接，排除运行时缺少文件问题

j) 对文件 c 配置如下：

5：cc = swgcc453

9：cFLAGS = $(GFLAGS) $(cWARN) $(cOPT) $(cDBUG) $(LIB_HEADER_DIRS) –fPIC –I{ COMPILER_INCLUDE }

15：LINKLIBSO = $(cc) –static

16：LINKEXE = OFswld

k) 对 general 文件配置如下：

3：PROJECT_LIBS = -l$(WM_PROJECT) // 链接器不支持-ldl，但会自动连接 libdl.a 库

l) 创建文件 mplibSWMPI，设置如下

1：PFLAGS = -DMPICH_SKIP_MPICXX

2：PINC = -I$(MPI_ARCH_PATH)/include

3：PLIBS = -L$(MPI_ARCH_PATH)/lib –lmpi

m) 进入${FOAM_ROOT}/wmake/rules/General

n) 对 general 文件配置如下

3：AR = OFswar //自动归档工具，较好地支持静态库链接，排除运行时缺少文件问题

5：RANLIB = swranlib

7：LD = OFswld

9： GFLAGS = -fPIC -D$(WM_ARCH) -DWM_ARCH_OPTION=$(WM_ARCH_OPTION) \

10： -DWM_$(WM_PRECISION_OPTION) -DWM_LABEL_SIZE=$(WM_LABEL_SIZE)

12：GLIBS = #-lm //此处禁止是由于所用链接器不能识别此选项，libm.a 会在链接器中自动连接

o) 对${FOAM_ROOT}/wmake/src/Makefile 进行配置，此脚本用于编译工具编译，必须使用本地编译配置

i. 首先删除或禁止对于编译规则的引用

49：#GENERAL_RULES = $(WM_DIR)/rules/General

50：#RULES = $(WM_DIR)/rules/$(WM_ARCH)$(WM_COMPILER)
53：#include $(RULES)/general
54：#include $(RULES)/$(WM_LINK_LANGUAGE)

ii. 其次添加本地编译工具如下

cc = $(WM_CC)
CC = $(WM_CXX)
cFLAGS = $(WM_CFLAGS) -O3
c++FLAGS = $(WM_CXXFLAGS) -O3

4. ThirdParty 编译配置，这里最重要的是 scotch 网格分解工具的编译，其它非必要第三方库暂时不予讨论，这里约定用${3rdParty}表示第三方库根目录——${FOAM_PRJ_DIR}/ThirdParty-3.0.0

a) 首先对${3rdParty}/Allwmake 进行配置，这个脚本整体采用的是本地编译工具，不适合对第三方库进行交叉编译

i. 增加 MPI 库分支

```
66：SWMPI)
67：        if [ -r $MPI_ARCH_PATH/lib/libmpi.so ]
68：        then
69：            echo "      have $WM_MPLIB shared library ($FOAM_MPI)"
70：            echo
71：         elif [ -r $MPI_ARCH_PATH/lib/libmpi.a ]
72：         then
73：            echo "      have $WM_MPLIB static library ($FOAM_MPI)"
74：            echo
75：         else
76：            echo " Error the SWMPI has no libs!! "
77：         fi
78：         ;;
79：
```

ii. 更改 scotch 编译配置脚本

206：
scotchMakefile=../../etc/wmakeFiles/scotch/Makefile.inc.sw26010_hpc_linux2.shlib-OpenFOAM-WM_ARCH_OPTIONWM_LABEL_OPTION

iii. 将 scotch 编译检测标准改为静态库（默认会编译层静态库），否则会反复编译

```
209：    -a -r $FOAM_EXT_LIBBIN/libscotch.a \
210：    -a -r $FOAM_EXT_LIBBIN/libscotcherrexit.a
270：        -a -r $FOAM_EXT_LIBBIN/$FOAM_MPI/libptscotch.a \
271：        -a -r $FOAM_EXT_LIBBIN/$FOAM_MPI/libptscotcherrexit.a ]
```

iv. 将 scotch 外部编译工具设置删除或禁止，以便让 scotchMakefile 指向文件生效

241： #["${WM_CC:-gcc}" != gcc] && configEnv="CC=$WM_CC CCS=$WM_CC"
303： #["${WM_CC:-gcc}" != gcc] && configEnv="CC=$WM_CC CCS=$WM_CC"

b) 配置 scotch 编译脚本

i. 进入${3rdParty}/etc/wmakeFiles/scotch

ii. 拷贝 Makefile.inc.i686_pc_linux2.shlib-OpenFOAM-64Int64 为 Makefile.inc.sw26010_hpc_linux2.shlib-OpenFOAM-64Int64

iii. 配置该脚本如下

2：LIB = .a

6：AR = swar

7：ARFLAGS = -cru –o

9：CCS = sw5cc –host

11：CCD = $(WM_CC) -I/usr/sw-mpp/mpi2/include/

12：CFLAGS = -O3 -fPIC -DINTSIZE64 -DCOMMON_FILE_COMPRESS_GZ -DCOMMON_RANDOM_FIXED_SEED -DSCOTCH_RENAME -Drestrict=__restrict

13：CLIBFLAGS = -shared –fPIC

14：LDFLAGS = -lz -lm –lrt

c) 去掉${3rdParty}/scotch_6.0.3/src/libscotch/Makefile 中编译器强制替换，即改为

55： $(MAKE) CC="$(CCS)" \

5. 将自动归档和链接脚本 OFswar 和 OFswld 拷贝到${FOAM_ROOT}/wmake 下并进行设置

a) OFswar 脚本

3：# parameters to be set by user

4：AR=swar // 静态库归档工具

5：ARFLAG=cr // 选项与${FOAM_ROOT}/wmake/rules/General/general 一致

6：OFLIB_PATH=${FOAM_LIBBIN} // OF 库文件安装目录

7：EXTLIB_PATH=${FOAM_EXT_LIBBIN} // OF 第三方库文件安装目录

8：DEBUG_OFAR=false # true | false // debug 开关，一般选 false

b) OFswld 配置

3 # parameters to be set by user

4 AR=swar

5 LD="swld453" // 链接器

6 LINK_FLAG_FLAG= # empty or "-Wl," // 根据链接器是否为编译器选择

7 OFLIB_PATH=${FOAM_LIBBIN} // OF 库文件安装目录

8 EXTLIB_PATH=${FOAM_EXT_LIBBIN} // OF 第三方库文件安装目录

9 DEBUG_OFLD=true # true | false // debug 开关，一般选 false

6. 进入${FOAM_ROOT}/src，强制修改静态库编译：find ./ -name Allwmake | xargs sed -i 's/libso/lib/g'

7. 在${FOAM_ROOT}/Allwmake 中修改编译类型：6：targetType=lib

8. 进入${FOAM_ROOT}/applications，强制修改静态库编译：find ./ -name Allwmake | xargs sed -i 's/libso/lib/g'

9. 打开静态库链接规则，${FOAM_ROOT}/wmake/Makefile 设置如下

172： $(AR) $(ARFLAGS) $(LIB).a $(OBJECTS)\

173： -L$(LIB_PLATFORMS) $(LIB_LIBS) $(GLIB_LIBS)

10. 编译过程

a) source ${FOAM_ROOT}/etc/bashrc

b) 运行./Allwmake（如需并行编译，设置环境变量 WM_NCOMPPROCS 为并行度）

致谢：

感谢顾寒锋博士的对本移植说明的测试和订正，感谢江南所何香博士前期对移植工作的支持。