《基于openEuler的openlb软件移植指南》

- 《基于openEuler的openlb软件移植指南》
  - 1. 介绍
  - 2. 配置编译与运行环境
    - 2.1 硬件平台、操作系统
    - 2.2 前置依赖、编译环境
    - 2.3 环境管理
    - 2.4 下载安装毕晟编译器
    - 2.5 下载编译Hyper MPI
    - 2.6 下载安装KML数学库
    - 2.7 下载编译OpenBlas线性代数库
  - 3. 编译优化OpenLB
    - 3.1 下载编译OpenLB
    - 3.2 运行OpenLB测试文件
    - 3.3 OpenLB优化思路
    - 3.4 数据可视化
  - 4. 使用hpcrunner一键安装OpenLB
    - 4.1 下载安装hpcrunner
    - 4.2 初始化hpcrunner
    - 4.3 安装必要软件包
    - 4.4 选择平台对应配置文件
    - 4.5 配置依赖环境
    - 4.6 进行编译
    - 4.7 运行测试

1. 介绍

- 简介：格子玻尔兹曼方法（LBM）是一类广泛应用的流体力学（CFD）算法，而OpenLB开源项目，由德国卡尔斯鲁厄理工学院（KIT）开发，提供了实现该算法的一个通用C++框架，可以用于解决很多领域比如流体力学的流体运动问题。OpenLB最新版本为1.5r0（发布日期2022/04/14），主要更新为GPU（CUDA）运算支持和对LBM算法格子碰撞步骤在AVX2/AVX512指令集上的SIMD向量化支持。由于CUDA和AVX指令集为部分平台的特性，可行的替代方案不多，因此本次移植选取仍具有时效性和实用性的OpenLB 1.4r0版本（发布日期2020/11/17）进行，且移植方法对于1.5r0版本仍具有可行性。

- 官网地址：https://www.openlb.net/

- OpenLB 1.4r0 用户手册：https://www.openlb.net/wp-content/uploads/2020/12/olb_ug-1.4r0.pdf

- 源码包下载地址：https://www.openlb.net/download/

- 版本特征：

- 1. 增强了用户体验，全面更新核心数据结构、提供改进的边界处理。
- 2. 更多的多物理场模型，例如二维和三维混合尺度扩散率湍流模型。
- 3. 对工作负载的性能改进，使用新的传播模式和通信方式，对案例程序平均提速28%。
- 4. 增加新的测试案例程序。

2. 配置编译与运行环境

2.1 硬件平台、操作系统

- 操作系统：openEuler
- 硬件平台：鲲鹏arm平台

本移植指南以鲲鹏920集群上的openEuler系统为例，软件编译部署需要根据安装平台进行合适的修改。

2.2 前置依赖、编译环境

- 编译器：毕晟编译器
- MPI实现：Hyper MPI
- MPI通信库相关：hucx，xucg
- 数学库：KML
- 开源线性代数库：OpenBlas

版本仅作参考，可使用高版本库进行替换。

名称	版本	软件下载地址
Bisheng Compiler	1.3.3	https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-compiler-1.3.3-aarch64-linux.tar.gz
Hyper MPI	1.1.1（二进制包）	https://mirrors.huaweicloud.com/kunpeng/archive/HyperMPI/1.1.1/Hyper-MPI_1.1.1_aarch64_OpenEuler20.03-LTS_Bisheng2.1.0_MLNX-OFED5.4.tar.gz
	1.1.1（源码）	https://github.com/kunpengcompute/hmpi/archive/refs/tags/v1.1.1-huawei.tar.gz
KML	1.6.0（毕昇编译器版本）	https://kunpeng-repo.obs.cn-north-4.myhuaweicloud.com/Kunpeng%20BoostKit/Kunpeng%20BoostKit%2022.0.RC3/BoostKit-kml_1.6.0_bisheng.zip
	1.6.0（GCC编译器版本）	https://kunpeng-repo.obs.cn-north-4.myhuaweicloud.com/Kunpeng%20BoostKit/Kunpeng%20BoostKit%2022.0.RC3/BoostKit-kml_1.6.0.zip
OpenBlas	0.3.21	https://github.com/xianyi/OpenBLAS/archive/refs/tags/v0.3.21.tar.gz
OpenLB	1.4r0	https://www.openlb.net/wp-content/uploads/2020/11/olb-1.4r0.tgz

2.3 环境管理

这里只提供一种通用的安装、环境配置手段，具体实现方式可根据编译安装的平台进行修改。

- 创建软件包下载、编译、安装文件夹并加入环境变量

```
# 软件包下载路径
mkdir -p ${HOME}/downloads
# 软件编译路径
mkdir -p ${HOME}/builds
# 软件安装路径
mkdir -p ${HOME}/softwares
# 加入环境变量
export OLB_PACKAGE_DIR=${HOME}/downloads
export OLB_BUILD_DIR=${HOME}/builds
export OLB_INSTALL_DIR=${HOME}/softwares
```

2.4 下载安装毕晟编译器

- 安装必要依赖和毕昇编译器

```
cd ${OLB_PACKAGE_DIR}
wget -c
https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/bisheng-
compiler-1.3.3-aarch64-linux.tar.gz
tar xzvf bisheng-compiler-1.3.3-aarch64-linux.tar.gz -C ${OLB_INSTALL_DIR}
```

- 配置毕昇编译器环境，可使用environment modules

```
echo "export PATH=$OLB_INSTALL_DIR/bisheng-compiler-1.3.3-aarch64-linux/bin:$PATH"
>> ~/.bashrc
echo "export LD_LIBRARY_PATH=$OLB_INSTALL_DIR/bisheng-compiler-1.3.3-aarch64-
linux/lib:$LD_LIBRARY_PATH" >> ~/.bashrc
source ~/.bashrc
export CC=`which clang`
export CXX=`which clang++`
```

2.5 下载编译Hyper MPI

- 安装必要前置环境

```
yum -y install libstdc++ libstdc++-devel
yum -y install unzip make autoconf automake git libtool
yum -y install flex
```

- 下载解压源码文件

```
wget https://github.com/kunpengcompute/hucx/archive/refs/tags/v1.1.1-huawei.zip -O
$OLB_PACKAGE_DIR/hucx-1.1.1-huawei.zip
wget https://github.com/kunpengcompute/xucg/archive/refs/tags/v1.1.1-huawei.zip -O
$OLB_PACKAGE_DIR/xucg-1.1.1-huawei.zip
wget https://github.com/kunpengcompute/hmpi/archive/refs/tags/v1.1.1-huawei.zip -O
$OLB_PACKAGE_DIR/hmpi-1.1.1-huawei.zip

cd $OLB_BUILD_DIR
unzip -q $OLB_PACKAGE_DIR/hucx-1.1.1-huawei.zip
unzip -q $OLB_PACKAGE_DIR/xucg-1.1.1-huawei.zip
unzip -q $OLB_PACKAGE_DIR/hmpi-1.1.1-huawei.zip
cp -rf xucg-1.1.1-huawei/* hucx-1.1.1-huawei/src/ucg/
```

- 编译hucx

```
cd $OLB_BUILD_DIR/hucx-1.1.1-huawei
./autogen.sh
# --disable-numa开启可能会影响性能，可安装libnuma-devel或编译安装numactl包并设置对应编译环境变量去除该参数的指定。
./contrib/configure-opt --prefix=$OLB_INSTALL_DIR/hmpi/hucx CFLAGS="-DHAVE___CLEAR_CACHE=1" --disable-numa --without-java
find . -name Makefile | xargs -I {} sed -i "s/-Werror//g" {}
find . -name Makefile | xargs -I {} sed -i "s/-implicit-function-declaration//g" {}
make -j
make install
```

- 编译Hyper MPI

```
cd $OLB_BUILD_DIR/hmpi-1.1.1-huawei
./autogen.pl
./configure --prefix=$OLB_INSTALL_DIR/hmpi --with-platform=contrib/platform/mellanox/optimized --enable-mpi1-compatibility --with-ucx=$OLB_INSTALL_DIR/hmpi/hucx
make -j
make install
```

- 将Hyper MPI环境加入环境变量

```
echo "export PATH=$OLB_INSTALL_DIR/hmpi/bin:$PATH" >> ~/.bashrc
echo "export LD_LIBRARY_PATH=$OLB_INSTALL_DIR/hmpi/lib:$LD_LIBRARY_PATH" >> ~/.bashrc
source ~/.bashrc
export CC=mpicc CXX=mpicxx
```

2.6 下载安装KML数学库

- 下载安装KML数学库

```
cd ${OLB_PACKAGE_DIR}
wget -c https://kunpeng-repo.obs.cn-north-4.myhuaweicloud.com/Kunpeng%20BoostKit/Kunpeng%20BoostKit%2022.0.RC3/BoostKit-kml_1.6.0_bisheng.zip
if [ -d /usr/local/kml ];then
  rpm -e boostkit-kml
fi
unzip -o BoostKit-kml_1.6.0_bisheng.zip
# 在将软件相关文件拷贝至在当前目录生成的usr/local文件夹
rpm --force --nodeps -ivh boostkit-kml-1.6.0-1.aarch64.rpm
```

- 若无权限使用rpm安装至/usr/local，使用如下命令对rpm包解压安装

```
rpm2cpio boostkit-kml-1.6.0-1.aarch64.rpm > boostkit-kml-1.6.0-1.aarch64.cpio
mv boostkit-kml-1.6.0-1.aarch64.cpio $OLB_INSTALL_DIR
```

```
cd $OLB_INSTALL_DIR
cpio -i --make-directories < boostkit-kml-1.6.0-1.aarch64.cpio
```

之后将KML的include、lib目录等在合适的环境变量中添加即可（LD_LIBRARY_PATH/LDFLAGS等）

2.7 下载编译OpenBlas线性代数库

- 编译安装OpenBlas库

```
cd ${OLB_PACKAGE_DIR}
wget -c https://github.com/xianyi/OpenBLAS/archive/refs/tags/v0.3.21.tar.gz -O
openblas-0.3.21.tar.gz
cd ${OLB_BUILD_DIR}
tar -xvf ${OLB_PACKAGE_DIR}/openblas-0.3.21.tar.gz -C .
cd openblas-0.3.21
# 针对鲲鹏920平台和毕昇编译器进行编译优化，需要根据具体平台进行修改。
make CC=clang CXX=clang++ FC=flang TARGET=TSV110 USE_OPENMP=1
make install PREFIX=${OLB_INSTALL_DIR}/openblas
```

- 将openblas环境加入环境变量

```
echo "export PATH=$OLB_INSTALL_DIR/openblas/bin:$PATH" >> ~/.bashrc
echo "export LD_LIBRARY_PATH=$OLB_INSTALL_DIR/openblas/lib:$LD_LIBRARY_PATH" >>
~/.bashrc
source ~/.bashrc
```

3. 编译优化OpenLB

3.1 下载编译OpenLB

- 获取软件包

```
cd ${OLB_PACKAGE_DIR}
wget -c https://www.openlb.net/wp-content/uploads/2020/11/olb-1.4r0.tgz
```

- 解压软件包并进行编译（最简方法，如何进行编译优化）

```
cd ${OLB_BUILD_DIR}
tar -xvf ${OLB_PACKAGE_DIR}/olb-1.4r0.tgz -C .
cd olb-1.4r0
make samples -j
```

3.2 运行OpenLB测试文件

- 测试脚本

由于OpenLB本身并没有提供相对应的一键测试脚本，故这里提供一个参考脚本，可进行编译文件的备份和测试。需要上一步编译后在OpenLB根目录下运行。

```
#!/bin/bash

# 当出现报错时不退出，进行记录
set +e

# envs

name="default"
root=.
outdir="logs"
backdir="backs"
# 默认不进行全部的测试
isTest=false
# 默认开启案例程序的测试
isTestExamples=true
# 默认开启数据可视化输出
isVisual=true
# 测试文件位置存储，需要进行存储
testsFilelists=()
# 使用的测试例
benchmark="benchmarks/mpi-openmp-run.sh"
# 案例测试例
# multiComponent/contactAngle2d
# porousMedia/porousPoiseuille3d
# laminar/bstep3d
examplesBench=("multiComponent/contactAngle2d/contactAngle2d"
"porousMedia/porousPoiseuille3d/porousPoiseuille3d" "laminar/bstep3d/bstep3d")

# 函数声明

## 说明
help() {
    echo "--name: 指定编译测试名称，在指定目录下进行拷贝和记录保存"
    echo "--backdir: 指定备份目录"
    echo "--outdir: 指定测试输出目录"
    echo "--benchmark: 指定测试运行文件"
    echo "--test: 是否进行全部实验的测试（默认关闭）"
    echo "--visual: 是否开启可视化数据输出（需安装gnuplot，默认开启）"
    echo "--examples: 是否开启案例程序的测试（默认开启）"
}

## 遍历文件夹寻找可执行文件
## 执行过程中进行文件的备份和文件测试位置的存储（用于生成测试脚本）
find_files() {
        local path=$1
        local d=`ls ${path}`
        for file in ${d[@]}
        do
            if test -d ${path}/${file}
            then
                find_files "${path}/${file}"
            elif [[ ! ${file} =~ \.cpp$|\.o$|Makefile$|\.mk$|\.py$|\.sh$ ]];
            then
```

```
                echo "${path}/${file}"
                mkdir -p ${backdir}/${name}/${path}
                cp ${path}/${file} ${backdir}/${name}/${path}/${file}
                if test -x ${path}/${file}
                then

testsFilelists[${#testsFilelists[*]}]="${backdir}/${name}/${path}/${file}"
                fi
            fi
        done
}

## 生成测试脚本
generate_script() {
    local lists=$1
    local benchFile=$2
    echo "#!/bin/bash" >> ${benchFile}
    echo "testFilelists=(${lists[@]})" >> ${benchFile}
    echo "useBenchmark=${useBenchmark}" >> ${benchFile}
    echo "outdir=${outdir}" >> ${benchFile}
    echo "name=${name}" >> ${benchFile}
    echo 'for testfile in ${testFilelists[@]}' >> ${benchFile}
    echo 'do' >> ${benchFile}
    echo 'cd $(dirname ${testfile})' >> ${benchFile}
    echo 'genDate=`date "+%Y%m%d_%H%M"`' >> ${benchFile}
    echo 'logName=$(basename ${testfile})_${genDate}' >> ${benchFile}
    echo '${useBenchmark} ${testfile} 2>&1 | tee ${outdir}/${name}/${logName}.log' >>
${benchFile}
    echo 'mv ./tmp ${outdir}/${name}/tmp_${logName}' >> ${benchFile}
    echo 'done' >> ${benchFile}
}

## 读取参数
read_config() {
    until [ $# -eq 0 ]
    do
        case $1 in
            "--name")
                shift
                name=$1
                ;;
            "--backdir")
                shift
                backdir=$1
                ;;
            "--outdir")
                shift
                outdir=$1
                ;;
            "--benchmark")
                shift
                benchmark=$1
                ;;
            "--test")
                shift
                isTest=$1
                ;;
```

```
            "--help")
                help
                exit 0
                ;;
            "--examples")
                shift
                isTestExamples=$1
                ;;
            *)
                echo "$1 options not recognized!"
                exit -1
                ;;
        esac
        shift
    done
}

# 备份可执行文件

read_config $@

# 均使用绝对路径
root=$(pwd)
backdir=$(cd $(dirname ${backdir}) && pwd)/$(basename ${backdir})
outdir=$(cd $(dirname ${outdir}) && pwd)/$(basename ${outdir})
useBenchmark=$(cd $(dirname ${benchmark}) && pwd)/$(basename ${benchmark})

if ! test -e ${useBenchmark}
then
    echo "${useBenchmark} file not exists! please check ${benchmark} is located in your
machine or in the current folder!"
    exit -1
fi

echo "===info==="
echo "openlb root: ${root}"
echo "name: ${name}"
echo "backdir: ${backdir}"
echo "outdir: ${outdir}"
echo "benchmark: ${useBenchmark}"
echo "test all?: ${isTest}"
echo "test examples?: ${isTestExamples}"
echo "open visual data: ${isVisual}"
echo "===info==="

mkdir -p ${backdir}/${name}
mkdir -p ${outdir}/${name}

cd ./examples

if [[ $? == 1 ]]; then
    echo "There isn't any test file in your current folder, check if you're in the
openlb source code folder!, current folder is ${root}"
    exit -1
fi

find_files .
```

```
cd ..

for (( i=0;i<${#examplesBench[@]};i++ )) do
    # 补全案例测试程序路径
    examplesBench[${i}]="${backdir}/${name}/${examplesBench[${i}]}"
done

if [[ ${isTest} == true ]]; then
    for testfile in ${testsFilelists[@]}
    do
        cd $(dirname ${testfile})
        genDate=`date "+%Y%m%d_%H%M"`
        logName=$(basename ${testfile})_${genDate}
        ${useBenchmark} ${testfile} 2>&1 | tee ${outdir}/${name}/${logName}.log
        mv ./tmp ${outdir}/${name}/tmp_${logName}
    done
elif [[ ${isTestExamples} == true ]]; then
    for testfile in ${examplesBench[@]}
    do
        cd $(dirname ${testfile})
        genDate=`date "+%Y%m%d_%H%M"`
        logName=$(basename ${testfile})_${genDate}
        ${useBenchmark} ${testfile} 2>&1 | tee ${outdir}/${name}/${logName}.log
        mv ./tmp ${outdir}/${name}/tmp_${logName}
    done
else
    echo "generate all test benchmark script..."
    benchFile=${backdir}/${name}-benchmark.sh
    if test -f ${benchFile}
    then
        rm ${benchFile}
    fi
    touch ${benchFile}
    generate_script "${testsFilelists[*]}" "${benchFile}"
    echo "all test benchmark script ${benchFile} done!"
    echo "generate examples test benchmark script..."
    benchFile=${backdir}/${name}-examples-benchmark.sh
    if test -f ${benchFile}
    then
        rm ${benchFile}
    fi
    touch ${benchFile}
    generate_script "${examplesBench[*]}" "${benchFile}"
    echo "examples test benchmark script ${benchFile} done!"
fi
```

- 脚本运行并进行备份和测试

假如上一步的脚本文件名为`BackupAndTest.sh`并存放于OpenLB项目根目录，则运行以下命令

```
cd ${OLB_BUILD_DIR}/olb-1.4r0
# 假设已经编译过，若仍未编译请回退至OpenLB编译步骤进行编译
# 具体参数请根据情况修改
```

```
chmod +x ./BackupAndTest.sh && ./BackupAndTest.sh --name default --backdir ./backs --
outdir ./logs --benchmark run.sh --examples false
```

完成后会生成${OLB_BUILD_DIR}/olb-1.4r0/backs/default目录存放examples下已经编译好的实验测试程序和相关数据文件，并提供前缀为default的自动测试文件（全部测试和三个典型程序测试），测试文件会将程序输出存放至${OLB_BUILD_DIR}/olb-1.4r0/logs/default文件夹。**run.sh**为测试文件，应为能接收第一个参数输入作为测试程序并能自动完成程序的运行、测试的可执行脚本，这里提供一个模板做参考：

```
#!/bin/bash

# using hyper mpi

testFile=$1

export OMP_NUM_THREADS=24
export OMP_PROC_BIND=true
export OMP_PLACES=cores

mpirun -machinefile nodes -np 12 -npernode 4 --bind-to numa --mca btl ^vader,tcp,openib
--map-by numa  --rank-by numa \
    -x UCX_TLS=sm,ud_x -x UCX_NET_DEVICES=mlx5_0:1 \
    -x UCX_BUILTIN_BCAST_ALGORITHM=3 \
    -x UCX_BUILTIN_ALLREDUCE_ALGORITHM=6 \
    -x UCX_BUILTIN_BARRIER_ALGORITHM=5 \
    -x UCX_BUILTIN_DEGREE_INTRA_FANOUT=3 \
    -x UCX_BUILTIN_DEGREE_INTRA_FANIN=2 \
    -x UCX_BUILTIN_DEGREE_INTER_FANOUT=7 \
    -x UCX_BUILTIN_DEGREE_INTER_FANIN=7 \
    --report-bindings ${testFile}
```

3.3 OpenLB优化思路

- 初始提供的config.mk

OpenLB根目录内提供的config.mk文件为主要编译配置文件，可修改该文件的参数进行编译优化，默认参数如下：

```
#  This file is part of the OpenLB library
#
#  Copyright (C) 2017 Markus Mohrhard, Mathias Krause
#  E-mail contact: info@openlb.net
#  The most recent release of OpenLB can be downloaded at
#  <http://www.openlb.net/>
#
#  This program is free software; you can redistribute it and/or
#  modify it under the terms of the GNU General Public License
#  as published by the Free Software Foundation; either version 2
#  of the License, or (at your option) any later version.
#
#  This program is distributed in the hope that it will be useful,
#  but WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#  GNU General Public License for more details.
```

```
#
#  You should have received a copy of the GNU General Public
#  License along with this program; if not, write to the Free
#  Software Foundation, Inc., 51 Franklin Street, Fifth Floor,
#  Boston, MA  02110-1301, USA.

#################################################################################
#################################################################################

CXX             := g++
#CXX             := icpc -D__aligned__=ignored
#CXX             := mpiCC
#CXX             := mpic++

CC              := gcc                                          # necessary for zlib,
for Intel use icc

OPTIM           := -O3 -Wall -march=native -mtune=native        # for gcc
#OPTIM           := -O3 -Wall -xHost                            # for Intel compiler
#OPTIM           := -O3 -Wall -xHost -ipo                       # optional for Intel
compiler
DEBUG           := -g -Wall -DOLB_DEBUG

CXXFLAGS        := $(OPTIM)
#CXXFLAGS        := $(DEBUG)

# compilation requires support for C++14
# works in:
#  * gcc 5 or later      (https://gcc.gnu.org/projects/cxx-status.html#cxx14)
#  * icc 17.0 or later   (https://software.intel.com/en-us/articles/c14-features-
supported-by-intel-c-compiler)
#  * clang 3.4 or later  (https://clang.llvm.org/cxx_status.html#cxx14)
CXXFLAGS        += -std=c++14

ARPRG           := ar
#ARPRG           := xiar                 # mandatory for intel compiler

LDFLAGS         :=

PARALLEL_MODE   := OFF
#PARALLEL_MODE   := MPI
#PARALLEL_MODE   := OMP
#PARALLEL_MODE   := HYBRID

MPIFLAGS        :=
OMPFLAGS        := -fopenmp

#BUILDTYPE       := precompiled
BUILDTYPE       := generic

FEATURES        :=
```

- 参数说明和优化思路

config.mk文件内的参数意义如下，大部分参数和常规编译手段中用到的变量意义一致：

1. CXX/CC：指定C++、C编译器。
2. OPTIM：链接前编译优化参数，会加在CXXFLAGS后。
3. DEBUG：开启DEBUG模式，此时会生成OpenLB的debug代码。
4. ARPRG：指定静态文件链接器。
5. LDFLAGS：指定链接时参数。
6. PARALLEL_MODE：指定编译模式：1. 串行模式。2. MPI多进程模式。3. OMP多线程模式。4. MPI+OpenMP混合模式。一般指定第四种可更好地提升性能。
7. FEATURE：开启特性（实质上是定义宏来开启额外的代码编译），OpenLB 1.4版本有OPENBLAS选项，可在部分代码中使用OpenBlas线性代数库提升运算性能。可参考用户手册或源码等开启其他特性

优化思路需要根据具体平台进行，这里根据鲲鹏920平台提供一种优化配置文件（头文件和库路径根据实际情况修改）：

```
CXX             := mpicxx
CC              := mpicc                    # necessary for zlib, for Intel use icc

OPTIM           := -Ofast -ffast-math -finline-functions -ffp-contract=fast -Wall -
mtune=MTUNE_PARAM -march=MARCH_PARAM+CPU_FEATURE_PARAM -I BISHENG_INCLUDE -I KML_INCLUDE
-I OPENBLAS_INCLUDE
DEBUG           := -g -Wall -DOLB_DEBUG
DEBUGNoWall     := -g -DOLB_DEBUG

CXXFLAGS        := $(OPTIM)
# for debug mode
# CXXFLAGS      += $(DEBUGNoWall)
# CXXFLAGS      := $(DEBUG)

# open pgo optimize
PGOCollect      := -fprofile-instr-generate
PGOOptim          := -fprofile-instr-use=code.profdata

CXXFLAGS        += -std=c++14

ARPRG           := ar
#ARPRG           := xiar                 # mandatory for intel compiler

LDFLAGS         := -fuse-ld=lld -flto -L OPENBLAS_LIB -lopenblas -L KML_LIB -lkm -lm -
lkfft -L BISHENG_LIB -ljemalloc -Wl,-z,muldefs
# for IPM analysis (static used)
# LDFLAGS  += -LIPM_LIB -lipm
# for pgo optimize
#LDFLAGS  += $(PGOCollect)
#LDFLAGS  += $(PGOOptim)

#PARALLEL_MODE   := OFF
#PARALLEL_MODE   := MPI
#PARALLEL_MODE   := OMP
PARALLEL_MODE   := HYBRID
MPIFLAGS        :=
OMPFLAGS        := -fopenmp
#BUILDTYPE       := precompiled
BUILDTYPE       := generic
FEATURES        := OPENBLAS
```

同时各个案例程序的文件夹内也提供了Makefile文件，可以修改该文件对特定程序做更加细化的优化。

3.4 数据可视化

OpenLB支持使用gnuplot读取OpenLB生成的gnuplot数据并输出对应仿真实验的图像。OpenLB各个仿真测试文件在运行后会在当前目录生成`tmp`文件夹并存放对应的gnuplot脚本和生成的图像（如果程序运行时gnuplot应用能找到并正常运行）。

若想手动生成图像，安装好支持PNG格式的gnuplot软件环境后在对应目录下执行`gnuplot -c .\tmp\gnuplotData\data\plotPNG.p`即可生成对应图像。

此外OpenLB同样生成了PDF输出类型的gnuplot脚本，需要gnuplot支持PDF格式即可依照上述步骤生成对应文件。

OpenLB同样会生成供Paraview应用进行解析的数据，可使用Paraview软件进行更加细化的科学分析。

4. 使用hpcrunner一键安装OpenLB

使用hpcrunner编译、部署OpenLB

4.1 下载安装hpcrunner

通过克隆git库安装hpcrunner

```
git clone https://gitee.com/openeuler/hpcrunner.git
```

4.2 初始化hpcrunner

初始化项目助手

```
cd hpcrunner
source init.sh
```

4.3 安装必要软件包

arm / x86 需要的软件包不同，根据实际环境和编译需求进行选择

```
# arm
yum install -y environment-odules git wget unzip make flex tar
# x86
yum install -y environment-modules git wget unzip make flex
yum install -y gcc gcc-c++ glibc-devel
yum install -y tcsh tcl lsof tk bc
```

4.4 选择平台对应配置文件

- arm平台的配置文件为`templates/openlb/1.4/data.openlb.arm.cpu.config`或其MPI编译优化版本`templates/openlb/1.4/data.openlb.arm.cpu.opt.config`（使用Hyper MPI编译，package内包含其他编译参数配置的patch补丁文件，可自行选择或修改）

```
# default config
./jarvis -use templates/openlb/1.4/data.openlb.arm.cpu.config
# optimization config
./jarvis -use templates/openlb/1.4/data.openlb.arm.cpu.opt.config
```

- x86 平台的配置文件为`templates/openlb/1.4/data.openlb.amd.cpu.config`或其MPI编译优化版本`templates/openlb/1.4/data.openlb.amd.cpu.oneapi.config`（使用Intel MPI编译，package内包含其他编译参数配置的patch补丁文件，可自行选择或修改）

```
# default config
./jarvis -use templates/openlb/1.4/data.openlb.amd.cpu.config
# optimization config
./jarvis -use templates/openlb/1.4/data.openlb.amd.cpu.oneapi.config
```

4.5 配置依赖环境

```
./jarvis -dp
```

4.6 进行编译

```
./jarvis -b
```

4.7 运行测试

```
./jarvis -r
```