|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | |  |
| NEMO-3.6 | | |  |
| 应用**移植指南** | | |
| **文档版本** | **01** | |
| **发布日期** | **2022-07-30** | |
| 华为网格系统---方案4-032.png | | | | |
|  | 华为技术有限公司 | | 附件1-16K |  |

|  |
| --- |
| 版权所有 © 华为技术有限公司2022。保留一切权利。  非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。  商标声明  附件3-版权声明页图和其他华为商标均为华为技术有限公司的商标。  本文档提及的其他所有商标或注册商标，由各自的所有人拥有。  注意  您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或默示的声明或保证。  由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。 |

|  |  |
| --- | --- |
| 华为技术有限公司 | |
| 地址： | 深圳市龙岗区坂田华为总部办公楼邮编：518129 |
| 网址： | <https://www.huawei.com> |
| 客户服务邮箱： | [support@huawei.com](mailto:support@huawei.com) |
| 客户服务电话： | 4008302118 |

目录

[目录 ii](#_Toc107667103)

[1 介绍 1](#_Toc107667104)

[2 环境要求 2](#_Toc107667105)

[3 移植规划数据 4](#_Toc107667106)

[4 配置编译环境 5](#_Toc107667107)

[4.1 搭建鲲鹏基座软件环境 5](#_Toc107667108)

[4.2 安装Zlib 6](#_Toc107667109)

[4.3 安装PNETCDF 7](#_Toc107667110)

[4.4 安装HDF5 8](#_Toc107667111)

[4.5 安装NETCDF-C 9](#_Toc107667112)

[4.6 安装NETCDF-Fortran 11](#_Toc107667113)

[4.7 安装Blitz 12](#_Toc107667114)

[4.8 安装XIOS 14](#_Toc107667115)

[5 获取源码 18](#_Toc107667116)

[6 编译和安装 19](#_Toc107667117)

[7 运行和验证 22](#_Toc107667118)

[8 修订记录 25](#_Toc107667119)

# 介绍

NEMO（Nucleus for European Modeling of the Ocean）是一个用于海洋学研究，业务海洋学季节预报和气候研究的最先进的建模框架。

关于NEMO的更多信息请访问[NEMO官网](http://www.nemo-ocean.eu/)。

语言：FORTRAN

一句话描述：海洋学研究，业务海洋学季节预报和气候研究的建模框架。

开源协议：CeCILL

建议的版本

建议使用版本为“NEMO-3.6”。

# 环境要求

硬件要求

硬件要求如表2-1所示。

硬件要求

| 项目 | 说明 |
| --- | --- |
| CPU | Kunpeng 920 |

软件要求

软件要求如表2-2所示。

注意

* 不同HPC应用的依赖软件不同，建议按照如下步骤判断其依赖软件：
* 1、查看其上游社区是否提供安装指导文档；
* 2、搜索网络上是否已经有社区提供相关安装文档或博客；
* 3、尝试安装该软件，根据报错情况，决定安装哪些依赖软件；
* 4、咨询华为工程师是否有相关经验。

软件要求

| 项目 | 版本 | 下载地址 |
| --- | --- | --- |
| 毕昇编译器 | 2.1.0 | <https://www.hikunpeng.com/developer/devkit/compiler?data=bisheng> |
| Hyper MPI | 1.1.1 | <https://github.com/kunpengcompute/hmpi/archive/refs/tags/v1.1.1-huawei.tar.gz> |
| HUCX | 1.1.1 | <https://github.com/kunpengcompute/hucx/archive/refs/tags/v1.1.1-huawei.tar.gz> |
| XUCG | 1.1.1 | <https://github.com/kunpengcompute/xucg/archive/refs/tags/v1.1.1-huawei.tar.gz> |
| Zlib | 1.2.11 | http://www.zlib.net/fossils/zlib-1.2.11.tar.gz |
| PNETCDF | 1.12.2 | https://parallel-netcdf.github.io/Release/pnetcdf-1.12.2.tar.gz |
| NETCDF-C | 4.8.1 | https://downloads.unidata.ucar.edu/netcdf-c/4.8.1/netcdf-c-4.8.1.tar.gz |
| NETCDF-Fortran | 4.4.1 | <https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.1> |
| HDF5 | 1.12.1 | https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.12/hdf5-1.12.1/src/hdf5-1.12.1.tar.gz |
| NEMO | 3.6 | <http://forge.ipsl.jussieu.fr/nemo/browser/NEMO/releases/release-3.6>/NEMOGCM |
| XIOS | 1.0 | 原链接失效，上传了压缩后的安装包。路径：./dependency/xios-1.0.tar.xz |
| 测试算例GYRE | 3.6 | 测试算例在NEMO-3.6源码包中的路径：NEMOGCM/CONFIG/GYRE |

操作系统要求

操作系统要求如表2-3所示。

操作系统要求

| 项目 | 版本 | 下载地址 |
| --- | --- | --- |
| openEuler | openEuler 20.03 SP3 | https://repo.openeuler.org/openEuler-20.03-LTS-SP3/ |
| Kernel | 4.19.90 | https://gitee.com/openeuler/kernel |

# 移植规划数据

本章节给出NEMO软件在移植过程中涉及到的相关软件安装规划路径的用途及详细说明。

移植规划数据

| 序号 | 软件安装规划路径 | 用途 | 说明 |
| --- | --- | --- | --- |
| 1 | /usr/local/bisheng | 毕昇编译器的安装规划路径。 | 这里的安装规划路径只是一个举例说明，建议部署在共享路径中。现网需要根据实际情况调整，**后续章节凡是遇到安装路径的命令，都以现网实际规划的安装路径为准进行替换，不再单独说明。** |
| 2 | /usr/local/hmpi | HyperMPI的安装规划路径。 |
| 3 | /usr/local/ZLIB | ZLIB的安装规划路径。 |
| 4 | /usr/local/PNETCDF | PNETCDF的安装规划路径。 |
| 5 | /usr/local/HDF5 | HDF5的安装规划路径。 |
| 6 | /usr/local/NETCDF | NETCDF-C的安装规划路径。 |
| NETCDF-Fortran的安装规划路径。 |
| 7 | /usr/local/BLITZ | Blitz的安装规划路径。 |
| 8 | /usr/local/XIOS | XIOS的安装规划路径。 |
| 9 | /usr/local/NEMO/nemo-3.6 | NEMO的安装规划路径。 |
| 10 | /usr/local/NEMO\_build | NEMO的构建规划路径。 |

# 配置编译环境

前提条件

使用SFTP工具将各安装包上传至服务器对应目录下。

配置流程

配置流程

| 序号 | 配置项 | 说明 |
| --- | --- | --- |
| 1 | 搭建鲲鹏基座软件环境 | 参考4.1 搭建鲲鹏基座软件环境 |
| 2 | 安装Zlib | 参考4.2 [安装Zlib](#_安装Zlib) |
| 3 | 安装PNETCDF | 参考4.3 [安装PNETCDF](#_安装PNETCDF) |
| 4 | 安装HDF5 | 参考4.4 [安装HDF5](#_安装HDF5) |
| 5 | 安装NETCDF-C | 参考4.5 [安装NETCDF-C](#_安装NETCDF-C) |
| 6 | 安装NETCDF-Fortran | 参考4.6 [安装NETCDF-Fortran](#_安装NETCDF-Fortran) |
| 7 | 安装Blitz | 参考4.7 [安装Blitz](#_安装Blitz) |
| 8 | 安装XIOS | 参考4.8 [安装XIOS](#_安装XIOS) |

## 搭建鲲鹏基座软件环境

操作步骤

请参考《鲲鹏基座软件搭建指南.docx》

## 安装Zlib

操作步骤

使用PuTTY工具，以root用户登录服务器。

解压zlib安装包。

**tar -zxf zlib-1.2.11.tar.gz**

注：解压缩时请以实际压缩包名称为准。

进入解压后的目录。

**cd zlib-1.2.11**

进行配置。

**CC=clang ./configure --prefix=/usr/local/ZLIB**

编译，安装。

**make -j$(nproc)**

**make install**

配置环境变量。

**cat>"/usr/local/ZLIB/zlib\_modulefiles"<<EOF**

**#%Module1.0**

**conflict zlib**

**variable modfile [file normalize [info script]]**

**proc getModulefileDir {} {**

**variable modfile**

**set modfile\_path [file dirname \$modfile]**

**return \$modfile\_path**

**}**

**set pwd [getModulefileDir]**

**set ZLIB \$pwd**

**setenv ZLIB \$ZLIB**

**prepend-path LD\_LIBRARY\_PATH \$ZLIB/lib**

**EOF**

在当前shell中加载环境变量

**module use /usr/local/ZLIB/**

**module load /usr/local/ZLIB/zlib\_modulefiles**

若要避免每打开一个shell，导入一次变量。可写入到系统配置文件中。

**vi /etc/profile**

新增如下内容：

**module use /usr/local/ZLIB/**

**module load /usr/local/ZLIB/zlib\_modulefiles**

----结束

## 安装PNETCDF

操作步骤

使用PuTTY工具，以root用户登录服务器。

解压PNETCDF安装包。

**tar -xvf pnetcdf-1.12.2.tar.gz**

注：解压缩时请以实际压缩包名称为准。

进入解压后的目录。

**cd pnetcdf-1.12.2**

进行配置安装。

**CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort CFLAGS="-fPIC -DPIC" FCFLAGS="-fPIC" FFLAGS="-fPIC" CXXFLAGS="-fPIC -DPIC" SEQ\_CC=mpicc \**

**./configure --prefix=/usr/local/PNETCDF --enable-shared**

编译，安装。

**make -j$(nproc)**

**make install**

配置环境变量。

**cat>"/usr/local/PNETCDF/pnetcdf\_modulefiles"<<EOF**

**#%Module1.0**

**conflict pnetcdf**

**variable modfile [file normalize [info script]]**

**proc getModulefileDir {} {**

**variable modfile**

**set modfile\_path [file dirname \$modfile]**

**return \$modfile\_path**

**}**

**set pwd [getModulefileDir]**

**set PNETCDF \$pwd**

**setenv PNETCDF \$PNETCDF**

**prepend-path LD\_LIBRARY\_PATH \$PNETCDF/lib**

**EOF**

在当前shell中加载环境变量

**module use /usr/local/PNETCDF/**

**module load /usr/local/PNETCDF/pnetcdf\_modulefiles**

若要避免每打开一个shell，导入一次变量。可写入到系统配置文件中。

**vi /etc/profile**

新增如下内容：

**module use /usr/local/PNETCDF/**

**module load /usr/local/PNETCDF/pnetcdf\_modulefiles**

----结束

## 安装HDF5

操作步骤

使用PuTTY工具，以root用户登录服务器。

解压文件。

**tar -xvf hdf5-1.12.1.tar.gz**

注：解压缩时请以实际压缩包名称为准。

进入解压后的目录。

**cd hdf5-1.12.1**

进行配置。

**CC=mpicc CXX=mpicxx FC=mpifort F77=mpi77 CFLAGS="-fPIC -DPIC" FCFLAGS="-fPIC -DPIC" FFLAGS="-fPIC" \**

**./configure --prefix=/usr/local/HDF5 --build=aarch64-unknown-linux-gnu --with-zlib=/usr/local/ZLIB --enable-shared --enable-fortran --enable-static --enable-parallel**

修改当前目录下的“libtool”文件。

**sed -i 's@\\\$wl-soname@-install\_name@g' libtool**

编译，安装。

**make -j$(nproc)**

**make install**

配置环境变量。

**cat>"/usr/local/HDF5/hdf5\_modulefiles"<<EOF**

**#%Module1.0**

**conflict hdf5**

**variable modfile [file normalize [info script]]**

**proc getModulefileDir {} {**

**variable modfile**

**set modfile\_path [file dirname \$modfile]**

**return \$modfile\_path**

**}**

**set pwd [getModulefileDir]**

**set HDF5 \$pwd**

**setenv HDF5 \$HDF5**

**prepend-path LD\_LIBRARY\_PATH \$HDF5/lib**

**EOF**

在当前shell中加载环境变量。

**module use /usr/local/HDF5/**

**module load /usr/local/HDF5/hdf5\_modulefiles**

若要避免每打开一个shell，导入一次变量。可写入到系统配置文件中。

**vi /etc/profile**

新增如下内容：

**module use /usr/local/HDF5/**

**module load /usr/local/HDF5/hdf5\_modulefiles**

----结束

## 安装NETCDF-C

操作步骤

使用PuTTY工具，以root用户登录服务器。

解压文件。

**tar -xvf netcdf-c-4.8.1.tar.gz**

注：解压缩时请以实际压缩包名称为准。

进入解压后的目录。

**cd netcdf-c-4.8.1**

进行配置。

**CC=mpicc CXX=mpicxx CPPFLAGS="-I/usr/local/HDF5/include -I/usr/local/ZLIB/include -I/usr/local/PNETCDF/include" \**

**CFLAGS="-I/usr/local/HDF5/include -I/usr/local/ZLIB/include -I/usr/local/PNETCDF/include -fPIC" \**

**LDFLAGS="-L/usr/local/HDF5/lib -L/usr/local/ZLIB/lib -L/usr/local/PNETCDF/lib" \**

**./configure --prefix=/usr/local/NETCDF --enable-shared --enable-netcdf-4 --enable-dap --with-pic --disable-doxygen --enable-pnetcdf --enable-static --enable-largefile**

安装，编译。

**make -j$(nproc)**

**make install**

配置环境变量。

**cat>"/usr/local/NETCDF/netcdf\_modulefiles"<<EOF**

**#%Module1.0**

**conflict netcdf**

**variable modfile [file normalize [info script]]**

**proc getModulefileDir {} {**

**variable modfile**

**set modfile\_path [file dirname \$modfile]**

**return \$modfile\_path**

**}**

**set pwd [getModulefileDir]**

**set NETCDF \$pwd**

**setenv NETCDF \$NETCDF**

**prepend-path LD\_LIBRARY\_PATH \$NETCDF/lib**

**EOF**

在当前shell中加载环境变量。

**module use /usr/local/NETCDF/**

**module load /usr/local/NETCDF/netcdf\_modulefiles**

若要避免每打开一个shell，导入一次变量。可写入到系统配置文件中。

**vi /etc/profile**

新增如下内容：

**module use /usr/local/NETCDF/**

**module load /usr/local/NETCDF/netcdf\_modulefiles**

----结束

## 安装NETCDF-Fortran

操作步骤

使用PuTTY工具，以root用户登录服务器。

解压NETCDF-Fortran安装包。

**tar -xvf netcdf-fortran-4.4.1.tar.gz**

注：解压缩时请以实际压缩包名称为准。

进入解压后的目录。

**cd netcdf-fortran-4.4.1**

编译配置。

**./configure --prefix=/usr/local/NETCDF CPPFLAGS="-I/usr/local/HDF5/include -I$NETCDF/include" LDFLAGS="-L/usr/local/HDF5/lib -L/usr/local/NETCDF/lib" --build=aarch64-unknown-linux-gnu --enable-static=yes --enable-shared CFLAGS="-O3 -fPIC -Wno-incompatible-pointer-types-discards-qualifiers -Wno-non-literal-conversion" FCFLAGS="-O3 -fPIC" LDFLAGS="-Wl,--build-id" CC=mpicc FC=mpif90 CXX=mpicxx**

制作libtool文件的补丁。

**vi libtool.patch**

写入以下内容并保存：

--- libtool 2022-06-02 11:40:47.571912526 +0800

+++ libtool\_new 2022-06-02 11:41:27.383581653 +0800

@@ -10115,7 +10115,7 @@

pic\_flag=""

# How to pass a linker flag through the compiler.

-wl=""

+wl="-Wl,"

# Compiler flag to prevent dynamic linking.

link\_static\_flag=""

@@ -10267,7 +10267,7 @@

pic\_flag=""

# How to pass a linker flag through the compiler.

-wl=""

+wl="-Wl,"

# Compiler flag to prevent dynamic linking.

link\_static\_flag=""

为libtool文件打上补丁。

**patch -p0 libtool libtool.patch**

编译，安装。

**make -j$(nproc)**

**make install**

注：NETCDF-FORTRAN与NETCDF-C的环境变量一样，因此，此处不再设置。

注意：nemo-3.6可以使用netcdf-fortran-4.4.1，不能使用更高版本的netcdf-fortran。

----结束

## 安装Blitz

操作步骤

使用PuTTY工具，以root用户登录服务器。

解压Blitz安装包。

**tar -xvf blitz-1.0.1.tar.gz**

注：解压缩时请以实际压缩包名称为准。

进入解压后的目录。

**cd blitz-1.0.1/**

编译配置。

**autoreconf -fiv**

**./configure --prefix=/usr/local/BLITZ --enable-fortran --enable-64bit**

编译，安装。

**make lib**

**make install**

配置环境变量。

**cat>"/usr/local/BLITZ/blitz\_modulefiles"<<EOF**

**#%Module1.0**

**conflict blitz**

**variable modfile [file normalize [info script]]**

**proc getModulefileDir {} {**

**variable modfile**

**set modfile\_path [file dirname \$modfile]**

**return \$modfile\_path**

**}**

**set pwd [getModulefileDir]**

**set BLITZ \$pwd**

**setenv BLITZ \$BLITZ**

**prepend-path LD\_LIBRARY\_PATH \$BLITZ/lib**

**EOF**

在当前shell中加载环境变量。

**module use /usr/local/BLITZ/**

**module load /usr/local/BLITZ/blitz\_modulefiles**

若要避免每打开一个shell，导入一次变量。可写入到系统配置文件中。

**vi /etc/profile**

新增如下内容：

**module use /usr/local/BLITZ/**

**module load /usr/local/BLITZ/blitz\_modulefiles**

---结束

## 安装XIOS

操作步骤

使用PuTTY工具，以root用户登录服务器。

安装编译的依赖库。

**yum install -y perl-lib\* boost-\***

拷贝dependency目录下的xios-1.0.tar.xz到XIOS构建目录。

**cp ./dependency/xios-1.0.tar.xz /usr/local/XIOS**

注：由于xios-1.0的svn源码仓库链接失效。因此，需要拷贝xios-1.0源码压缩包到XIOS构建目录。

进入XIOS构建目录。

**cd /usr/local/XIOS**

解压缩xios-1.0安装包。

**tar xf xios-1.0.tar.xz**

进入源码目录。

**cd xios-1.0**

制作bld.cfg文件的补丁。

**vi bld.cfg.patch**

写入以下内容并保存：

--- bld.cfg 2022-05-22 15:53:29.386630000 +0800

+++ bld.cfg\_new 2022-05-22 16:20:20.582128663 +0800

@@ -24,13 +24,13 @@

search\_src true

src::zzz .

-src::date $PWD/extern/boost/src/date\_time

+#src::date $PWD/extern/boost/src/date\_time

src::blitz $PWD/extern/blitz/src

src::netcdf $PWD/extern/netcdf4

bld::lib xios

bld::target libxios.a

#bld::target generate\_fortran\_interface.exe

-bld::target xios\_server.exe test\_client.exe parse\_xml.exe test\_complete.exe test\_xios\_interface.exe

+#bld::target xios\_server.exe test\_client.exe parse\_xml.exe test\_complete.exe test\_xios\_interface.exe

bld::exe\_dep

bld::tool::cc %CCOMPILER

为bld.cfg文件打上补丁。

**patch -p0 bld.cfg bld.cfg.patch**

创建“arch-AARCH64\_GNU\_LINUX.env”配置文件。

**vi arch/arch-AARCH64\_GNU\_LINUX.env**

写入以下内容并保存：

export OMPI\_CC=clang

export OMPI\_CXX=clang++

export OMPI\_FC=flang

export HDF5\_INC\_DIR=/usr/local/HDF5/include

export HDF5\_LIB\_DIR=/usr/local/HDF5/lib

export NETCDF\_INC\_DIR=/usr/local/NETCDF/include

export NETCDF\_LIB\_DIR=/usr/local/NETCDF/lib

export BOOST\_INC\_DIR=/usr/include/boost/

export BOOST\_LIB\_DIR=/usr/lib64/

export BLITZ\_INC\_DIR=/usr/include/BLITZ/include

export BLITZ\_LIB\_DIR=/usr/include/BLITZ/lib

创建“arch-AARCH64\_GNU\_LINUX.fcm”配置文件。

**vi arch/arch-AARCH64\_GNU\_LINUX.fcm**

写入以下内容并保存：

%CCOMPILER mpicc

%FCOMPILER mpif90

%LINKER mpif90

%BASE\_CFLAGS -ansi -w -D\_GLIBCXX\_USE\_CXX11\_ABI=0 -D\_\_XIOS\_EXCEPTION

%PROD\_CFLAGS -O3 -DBOOST\_DISABLE\_ASSERTS

%DEV\_CFLAGS -g -O2

%DEBUG\_CFLAGS -g

%BASE\_FFLAGS -D\_\_NONE\_\_

%PROD\_FFLAGS -O3

%DEV\_FFLAGS -g -O2

%DEBUG\_FFLAGS -g

%BASE\_INC -D\_\_NONE\_\_

%BASE\_LD -lstdc++

%CPP cpp

%FPP cpp -P

%MAKE make

创建“arch-AARCH64\_GNU\_LINUX.path”配置文件。

**vi arch/arch-AARCH64\_GNU\_LINUX.path**

写入以下内容并保存：

NETCDF\_INCDIR="-I /usr/local/NETCDF/include"

NETCDF\_LIBDIR="-L /usr/local/NETCDF/lib"

NETCDF\_LIB="-lnetcdff -lnetcdf"

MPI\_INCDIR="-I /usr/local/hmpi/include"

MPI\_LIBDIR="-L /usr/local/hmpi/lib"

MPI\_LIB="-lmpi"

HDF5\_INCDIR="-I /usr/local/HDF5/include"

HDF5\_LIBDIR="-L /usr/local/HDF5/lib"

HDF5\_LIB="-lhdf5\_hl -lhdf5 -lz"

BOOST\_INCDIR="-I /usr/include/boost/"

BOOST\_LIBDIR="-L /usr/lib64/"

BOOST\_LIB=""

BLITZ\_INCDIR="-I /usr/local/BLITZ/include"

BLITZ\_LIBDIR="-L /usr/local/BLITZ/lib"

BLITZ\_LIB="-lblitz"

建立编译依赖库的符号链接。

**chmod +x ./make\_xios**

**cd extern**

**ln -s /usr/local/BLITZ ./blitz**

**ln -s /usr/local/NETCDF ./netcdf4**

**ln -s /usr/include/boost ./boost**

编译，安装。

**cd ..**

**./make\_xios --dev --job $(nproc) --full --arch AARCH64\_GNU\_LINUX**

配置环境变量。

**cat>"/usr/local/XIOS/xios-1.0/xios\_modulefiles"<<EOF**

**#%Module1.0**

**conflict xios**

**variable modfile [file normalize [info script]]**

**proc getModulefileDir {} {**

**variable modfile**

**set modfile\_path [file dirname \$modfile]**

**return \$modfile\_path**

**}**

**set pwd [getModulefileDir]**

**set XIOS \$pwd**

**setenv XIOS \$XIOS**

**prepend-path LD\_LIBRARY\_PATH \$XIOS/lib**

**EOF**

在当前shell中加载环境变量。

**module use /usr/local/XIOS/xios-1.0/**

**module load /usr/local/XIOS/xios-1.0/xios\_modulefiles**

若要避免每打开一个shell，导入一次变量。可写入到系统配置文件中。

**vi /etc/profile**

新增如下内容：

**module use /usr/local/XIOS/xios-1.0/**

**module load /usr/local/XIOS/xios-1.0/xios\_modulefiles**

---结束

# 获取源码

操作步骤

下载NEMO源码包“NEMO-3.6”。

下载地址：<http://forge.ipsl.jussieu.fr/nemo/browser/NEMO/releases/release-3.6>/NEMOGCM

----结束

# 编译和安装

操作步骤

使用PuTTY工具，以root用户登录服务器。

安装环境依赖包

**yum install -y environment-modules patch systemd-devel wget svn**

**set -e**

进入NEMO构建目录。

**cd /usr/local/NEMO\_build**

创建并设置编译NEMO的环境变量。

**echo "export NETCDF\_DIR=/usr/local/NETCDF" >> nemo-env.sh**

**echo "export HDF\_DIR=/usr/local/HDF5" >> nemo-env.sh**

**echo "export XIOS\_DIR=/usr/local/XIOS/xios-1.0" >> nemo-env.sh**

**source nemo-env.sh**

使用svn命令下载NEMO源码。

**svn co** [**http://forge.ipsl.jussieu.fr/nemo/svn/NEMO/releases/release-3.6/NEMOGCM**](http://forge.ipsl.jussieu.fr/nemo/svn/NEMO/releases/release-3.6/NEMOGCM)

**cd NEMOGCM**

创建编译NEMO的编译器配置文件。

**vi ARCH/arch-aarch64\_gnu.fcm**

写入以下内容并保存（黑体部分为针对鲲鹏平台的优化编译参数）：

%NCDF\_HOME $NETCDF\_DIR

%XIOS\_HOME $XIOS\_DIR

%NCDF\_INC -I%NCDF\_HOME/include

%NCDF\_LIB -L%NCDF\_HOME/lib -lnetcdff -lnetcdf -lstdc++

%XIOS\_INC -I%XIOS\_HOME/inc

%XIOS\_LIB -L%XIOS\_HOME/lib -lxios

%CPP cpp

%CPPFLAGS -P –traditional **-march=armv8.2-a -ffp-contract=off**

%FC mpif90 -c –cpp

%FCFLAGS -mcpu=native -fdefault-real-8 -O3 -funroll-loops -g

%FFLAGS %FCFLAGS

%LD mpif90

%LDFLAGS -lstdc++

%FPPFLAGS -P -C -traditional

%AR ar

%ARFLAGS rs

%MK make

%USER\_INC %XIOS\_INC %NCDF\_INC

%USER\_LIB %XIOS\_LIB %NCDF\_LIB

%CC cc

%CFLAGS -O2

编译，安装。

**cd ./CONFIG**

**./makenemo -m aarch64\_gnu -j $(nproc) -r GYRE -n 'MY\_GYRE' add\_key "key\_nosignedzero"**

注：执行上述命令后，在CONFIG路径下将生成MY\_GYRE算例。

创建安装目录并转移至安装目录。

**mkdir -p** **/usr/local/NEMO/nemo-3.6 && cp -r /usr/local/NEMO\_build/NEMOGCM/\* /usr/local/NEMO/nemo-3.6**

配置环境变量。

**cat>"/usr/local/NEMO/nemo-3.6/nemo\_modulefiles"<<EOF**

**#%Module1.0**

**conflict nemo**

**variable modfile [file normalize [info script]]**

**proc getModulefileDir {} {**

**variable modfile**

**set modfile\_path [file dirname \$modfile]**

**return \$modfile\_path**

**}**

**set pwd [getModulefileDir]**

**set NEMO \$pwd**

**setenv NEMO \$NEMO**

**prepend-path PATH \$NEMO/CONFIG/MY\_GYRE/BLD/bin**

**EOF**

在当前shell中加载环境变量

**module use /usr/local/NEMO/nemo-3.6**

**module load /usr/local/NEMO/nemo-3.6/nemo\_modulefiles**

若要避免每打开一个shell，导入一次变量。可写入到系统配置文件中。

**vi /etc/profile**

新增如下内容：

**module use /usr/local/NEMO/nemo-3.6**

**module load /usr/local/NEMO/nemo-3.6/nemo\_modulefiles**

删除构建目录。

**rm -rf /usr/local/NEMO\_build**

----结束

# 运行和验证

操作步骤

使用PuTTY工具，以root用户登录服务器。

加载环境变量。

**module use /usr/local/NEMO/nemo-3.6**

**module load /usr/local/NEMO/nemo-3.6/nemo\_modulefiles**

进入算例路径。

**cd /usr/local/NEMO/nemo-3.6/CONFIG/MY\_GYRE/EXP00**

制作namelist\_cfg文件的补丁。

**vi namelist\_cfg.patch**

写入以下内容并保存：

--- namelist\_cfg 2022-06-09 08:36:23.262617941 +0800

+++ namelist\_cfg\_new 2022-06-09 08:28:04.907940421 +0800

@@ -148,6 +148,7 @@

!-----------------------------------------------------------------------

&namberg ! iceberg parameters

!-----------------------------------------------------------------------

+ln\_icebergs = .false.

/

!-----------------------------------------------------------------------

&namlbc ! lateral momentum boundary condition

@@ -378,3 +379,20 @@

!-----------------------------------------------------------------------

ln\_neptramp = .false. ! ramp down Neptune velocity in shallow water

/

+

+!-----------------------------------------------------------------------

+&namsto ! Stochastic parametrization of EOS

+!-----------------------------------------------------------------------

+ ln\_rststo = .false. ! start from mean parameter (F) or from restart file (T)

+ ln\_rstseed = .true. ! read seed of RNG from restart file

+ cn\_storst\_in = "restart\_sto" ! suffix of stochastic parameter restart file (input)

+ cn\_storst\_out = "restart\_sto" ! suffix of stochastic parameter restart file (output)

+ ln\_sto\_eos = .false. ! stochastic equation of state

+ nn\_sto\_eos = 1 ! number of independent random walks

+ rn\_eos\_stdxy = 1.4 ! random walk horz. standard deviation (in grid points)

+ rn\_eos\_stdz = 0.7 ! random walk vert. standard deviation (in grid points)

+ rn\_eos\_tcor = 1440.0 ! random walk time correlation (in timesteps)

+ nn\_eos\_ord = 1 ! order of autoregressive processes

+ nn\_eos\_flt = 0 ! passes of Laplacian filter

+ rn\_eos\_lim = 2.0 ! limitation factor (default = 3.0)

+/

为namelist\_cfg文件打上补丁。

**patch -p0 namelist\_cfg namelist\_cfg.patch**

注：毕昇编译器编译nemo-3.6时，会生成错误的namelist\_cfg文件，因此需修正后，才能在openEuler系统上正确执行。

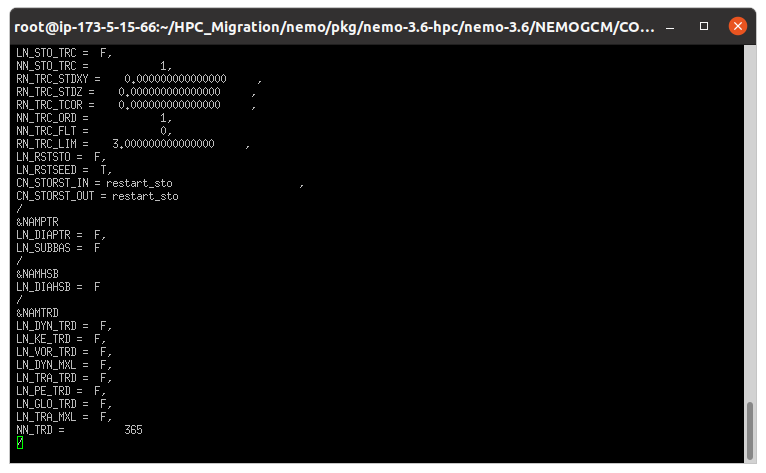
运行NEMO的算例。

**mpirun --allow-run-as-root -np 16 -N 16 -mca btl ^openib nemo.exe**

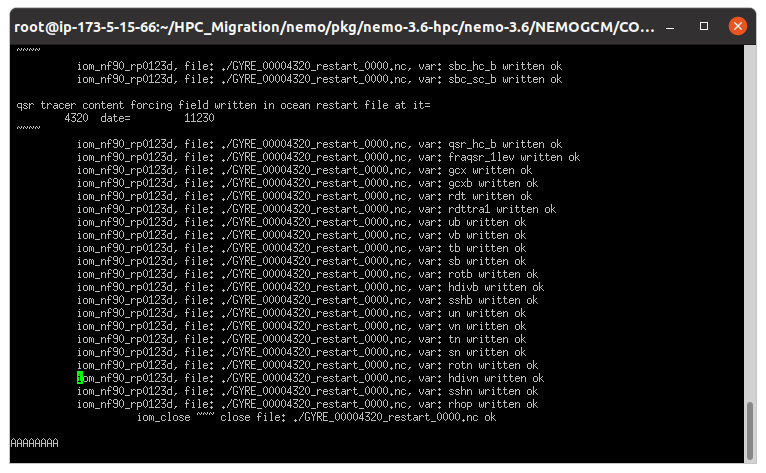
表1：参数说明

|  |  |
| --- | --- |
| **参数** | **说明** |
| **-np** | 测试使用的总进程数 |
| **-N** | 每个服务器上运行的进程数 |

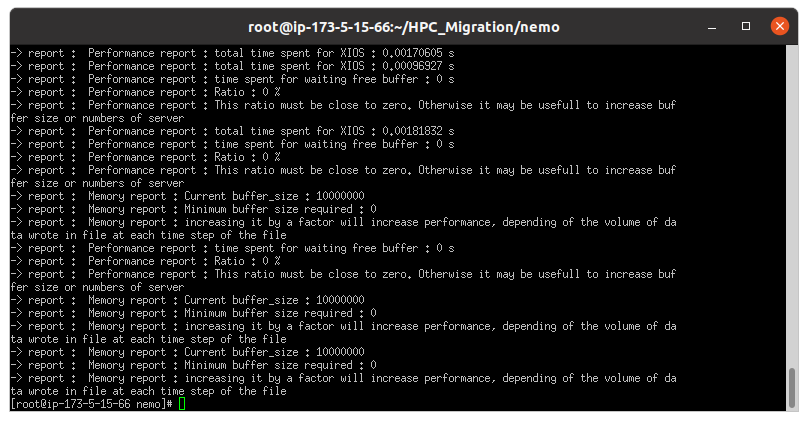
注意：使用修正后的namelist\_cfg文件，运行NEMO程序后，才能出现下图正确的output\_namelist.dyn，注意文件的结尾部分，必须要有绿色框中的/符号。



当NEMO运行完毕后，生成的日志“ocean.output”的结尾处出现如下图所示内容时，表示NEMO程序运行正常结束。



另外，运行完毕后，屏幕会输出调整XIOS输出参数提高保存效率的建议和计时，如下图。



----结束

# 修订记录

| 发布日期 | 修订记录 |
| --- | --- |
| 2022-07-30 | 第一次正式发布。 |