|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | |  |
| CESM-2.1.1 | | |  |
| 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](#_Toc112405289)

[1 介绍 1](#_Toc112405290)

[2 环境要求 2](#_Toc112405291)

[3 移植规划数据 4](#_Toc112405292)

[4 配置编译环境 5](#_Toc112405293)

[4.1 搭建鲲鹏基座软件环境 6](#_Toc112405294)

[4.2 安装ZLIB 6](#_Toc112405295)

[4.3 安装HDF5 7](#_Toc112405296)

[4.4 安装PNETCDF 8](#_Toc112405297)

[4.5 安装NETCDF-C 10](#_Toc112405298)

[4.6 安装NETCDF-Fortran 11](#_Toc112405299)

[4.7 安装OpenBLAS 12](#_Toc112405300)

[5 获取源码 14](#_Toc112405301)

[6 编译和安装 15](#_Toc112405302)

[7 运行和验证 23](#_Toc112405303)

[8 修订记录 25](#_Toc112405304)

# 介绍

CESM（Community Earth System Model）是由美国UCAR于2010年07月推出的新一代地球系统模式（Earth System Model），是一个完全耦合的气候模式，在CESM中包括大气、陆地、海洋、海冰、陆冰等几大模块。其主要用于研究地球的过去、现在、和未来的气候状况。

关于CESM的更多信息请访问[CESM官网](https://www.cesm.ucar.edu/)。

语言：Python

一句话描述：地球系统模拟，地球气候模拟预报和建模框架。

开源协议：UCAR自定义开源协议

建议的版本

建议使用版本为“CESM-2.1.1”。

# 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 |
| --- | --- | --- |
| CESM | 2.1.1 | <https://github.com/ESCOMP/CESM/archive/refs/tags/cesm2.1.1.tar.gz> |
| 毕昇编译器 | 2.1.0 | https://www.hikunpeng.com/zh/developer/devkit/compiler/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> |
| OpenBLAS | 0.3.6 | <https://github.com/xianyi/OpenBLAS/archive/refs/tags/v0.3.6.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://github.com/Unidata/netcdf-c/archive/refs/tags/v4.8.1.tar.gz> |
| NETCDF-Fortran | 4.4.1 | <https://github.com/Unidata/netcdf-fortran/archive/refs/tags/v4.4.1.tar.gz> |
| 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> |
| 测试算例 | f19\_g16 | HPC应用程序创建 |

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CESM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 | Hyper MPI的安装规划路径。 |
| 3 | /usr/local/ZLIB | ZLIB的安装规划路径。 |
| 4 | /usr/local/NETCDF | NETCDF-C和NETCDF-Fortran的安装规划路径。 |
| 5 | /usr/local/HDF5 | HDF5的安装规划路径。 |
| 6 | /usr/local/PNETCDF | PNETCDF的安装规划路径。 |
| 7 | /usr/local/OPENBLAS | OpenBLAS的安装规划路径。 |
| 8 | /usr/local/CESM | CESM的构建规划路径。 |
| 9 | /usr/local/cesm | CESM的算例安装规划路径。 |
| 10 | /usr/local/cesm/inputdata | CESM的算例输入数据文件的存放路径。 |

# 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HDF5 | 参考4.3 [安装HDF5](#_安装HDF5) |
| 4 | 安装PNETCDF | 参考4.4 [安装PNETCDF](#_安装PNETCDF) |
| 5 | 安装NETCDF-C | 参考4.5 [安装NETCDF-C](#_安装NETCDF) |
| 6 | 安装NETCDF-Fortran | 参考4.6 [安装NETCDF-Fortran](#_安装NETCDF-Fortran) |
| 7 | 安装OpenBlas | 参考4.7 [安装OpenBLAS](#_安装OpenBLAS) |

## 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 -xzvf 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**

编译配置。

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

进行编译安装。

**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HDF5

操作步骤

使用PuTTY工具，以root用户登录服务器。

解压HDF5安装包。

**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 --with-zlib=/usr/local/ZLIB --enable-shared --enable-fortran --enable-static --enable-parallel**

修改“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 xf 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。

**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NETCDF-C

操作步骤

使用PuTTY工具，以root用户登录服务器。

解压NETCDF安装包。

**tar -zxvf 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**

编译，安装。

**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**

**prepend-path NETCDF\_PATH \$NETCDF**

**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 xf 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**

进行配置。

**./configure --prefix=/usr/local/NETCDF CPPFLAGS="-I/usr/local/HDF5/include -I/usr/local/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" 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：CESM-2.2.1目前只能使用NETCDF-Fortran 4.4.1，不能使用更高版本的NETCDF-Fortran。

----结束

## 安装OpenBLAS

操作步骤

使用PuTTY工具，以root用户登录服务器。

解压OpenBLAS安装包。

**tar zxvf OpenBLAS-0.3.6.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 OpenBLAS-0.3.6**

编译，安装。

**export CC=`which clang`**

**export CXX=`which clang++`**

**export FC=`which flang`**

**make -j$(nproc)**

**make PREFIX=/usr/local/OPENBLAS install**

配置环境变量。

**cat>"/usr/local/OPENBLAS/openblas\_modulefiles"<<EOF**

**#%Module1.0**

**conflict openblas**

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

**proc getModulefileDir {} {**

**variable modfile**

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

**return \$modfile\_path**

**}**

**set pwd [getModulefileDir]**

**set OPENBLAS \$pwd**

**setenv OPENBLAS \$OPENBLAS**

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

**EOF**

在当前shell中加载环境变量。

**module use /usr/local/OPENBLAS/**

**module load /usr/local/OPENBLAS/openblas\_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/OPENBLAS/**

**module load /usr/local/OPENBLAS/openblas\_modulefiles**

----结束

# 获取源码

操作步骤

下载CESM安装包“cesm-release-cesm2.1.1.tar.gz”。

下载地址：<https://github.com/ESCOMP/CESM/archive/refs/tags/cesm2.1.1.tar.gz>

**----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CMake。

**yum install -y cmake**

进入构建目录。

**cd /usr/local/****CESM**

通过下方链接下载源码。

[**https://github.com/ESCOMP/CESM/archive/refs/tags/release-cesm2.1.1.tar.gz**](https://github.com/ESCOMP/CESM/archive/refs/tags/release-cesm2.1.1.tar.gz)

解压CESM安装包，进入解压后的目录。

**tar xf release-cesm2.1.1.tar.gz && rm -rf release-cesm2.1.1.tar.gz**

**cd CESM-release-cesm2.1.1**

安装environment-modules，git和svn。

**yum install git svn environment-modules -y**

添加验证信息。

**svn ls https://svn-ccsm-models.cgd.ucar.edu/ww3/release\_tags**

下载CESM的编译脚本文件。

**wget -t 10 -c https://github.com/ESMCI/cime/archive/refs/tags/cime5.6.19.tar.gz**

**tar xf cime5.6.19.tar.gz && mv cime-cime5.6.19 cime && rm -f cime5.6.19.tar.gz**

下载CESM的组件源码。

**mkdir -p components**

**cd components**

**svn co https://svn-ccsm-models.cgd.ucar.edu/cam1/release\_tags/cam\_cesm2\_1\_rel\_29/components/cam/**

**wget -t 10 -c https://github.com/ESCOMP/CESM\_CICE5/archive/refs/tags/cice5\_20190321.tar.gz**

**tar xf cice5\_20190321.tar.gz && mv CESM\_CICE5-cice5\_20190321 cice && rm -f cice5\_20190321.tar.gz**

**wget -t 10 -c https://github.com/ESCOMP/CISM-wrapper/archive/refs/tags/release-cesm2.0.04.tar.gz**

**tar xf release-cesm2.0.04.tar.gz && mv CISM-wrapper-release-cesm2.0.04 cism && rm -f release-cesm2.0.04.tar.gz**

**cd cism**

**wget -t 10 -c https://github.com/ESCOMP/cism/archive/refs/tags/release-cism2.1.03.tar.gz**

**tar xf release-cism2.1.03.tar.gz && mv CISM-release-cism2.1.03 source\_cism && rm -f release-cism2.1.03.tar.gz**

**cd ..**

**wget -t 10 -c https://github.com/ESCOMP/CTSM/archive/refs/tags/release-clm5.0.25.tar.gz**

**tar xf release-clm5.0.25.tar.gz && mv CTSM-release-clm5.0.25 clm && rm -f release-clm5.0.25.tar.gz**

**cd clm**

**wget -t 10 -c https://github.com/ncar/fates-release/archive/refs/tags/fates\_s1.21.0\_a7.0.0\_br\_rev2.tar.gz**

**tar xf fates\_s1.21.0\_a7.0.0\_br\_rev2.tar.gz && mv fates-release-fates\_s1.21.0\_a7.0.0\_br\_rev2 src/fates && rm -f fates\_s1.21.0\_a7.0.0\_br\_rev2.tar.gz**

**wget -t 10 -c https://github.com/ESCOMP/ptclm/archive/refs/tags/PTCLM2\_180611.tar.gz**

**tar xf PTCLM2\_180611.tar.gz && mv PTCLM-PTCLM2\_180611 tools/PTCLM && rm -f PTCLM2\_180611.tar.gz**

**cd ..**

**wget -t 10 -c https://github.com/ESCOMP/mosart/archive/refs/tags/release-cesm2.0.03.tar.gz**

**tar xf release-cesm2.0.03.tar.gz && mv MOSART-release-cesm2.0.03 mosart && rm -f release-cesm2.0.03.tar.gz**

**wget -t 10 -c https://github.com/ESCOMP/POP2-CESM/archive/refs/tags/pop2\_cesm2\_1\_rel\_n06.tar.gz**

**tar xf pop2\_cesm2\_1\_rel\_n06.tar.gz && mv POP2-CESM-pop2\_cesm2\_1\_rel\_n06 pop && rm -f pop2\_cesm2\_1\_rel\_n06.tar.gz**

**cd pop**

**wget -t 10 -c https://github.com/CVMix/CVMix-src/archive/refs/tags/v0.93-beta.tar.gz**

**tar xf v0.93-beta.tar.gz && mv CVMix-src-0.93-beta externals/CVMix && rm -f v0.93-beta.tar.gz**

**wget -t 10 -c https://github.com/marbl-ecosys/MARBL/archive/refs/tags/cesm2.1-n00.tar.gz**

**tar xf cesm2.1-n00.tar.gz && mv MARBL-cesm2.1-n00 externals/MARBL && rm -f cesm2.1-n00.tar.gz**

**cd ..**

**wget -t 10 -c https://github.com/ESCOMP/rtm/archive/refs/tags/release-cesm2.0.02.tar.gz**

**tar xf release-cesm2.0.02.tar.gz && mv RTM-release-cesm2.0.02 rtm && rm -f release-cesm2.0.02.tar.gz**

**wget -t 10 -c https://github.com/ESCOMP/WW3-CESM/archive/refs/tags/ww3\_cesm2\_1\_rel\_01.tar.gz**

**tar xf ww3\_cesm2\_1\_rel\_01.tar.gz && mv WW3-CESM-ww3\_cesm2\_1\_rel\_01 ww3 && rm -f ww3\_cesm2\_1\_rel\_01.tar.gz**

**cd ..**

制作4个源码文件的补丁。

**cd /usr/local/CESM/CESM-release-cesm2.1.1/cime/src/externals/pio1/pio/**

**vi nf\_mod.F90.patch**

写入以下内容并保存：

1758,1764d1757

< #ifdef \_NETCDF4

< if(iotype==pio\_iotype\_netcdf4c) then

< if(vardesc%ndims>0 .and. ierr==PIO\_NOERR) then

< ierr = nf90\_def\_var\_deflate(File%fh,vardesc%varid,0,1,1)

< end if

< endif

< #endif

**vi ionf\_mod.F90.patch**

写入以下内容并保存：

65,107d64

< #ifdef \_NETCDF4

< case(PIO\_iotype\_netcdf4p)

< ! The 64 bit options are not compatable with hdf5 format files

<

< if(iand(PIO\_64BIT\_OFFSET,amode)==PIO\_64BIT\_OFFSET) then

< nmode = ieor(amode,PIO\_64BIT\_OFFSET)

< else if(iand(PIO\_64BIT\_DATA,amode)==PIO\_64BIT\_DATA) then

< nmode = ieor(amode,PIO\_64BIT\_DATA)

< else

< nmode=amode

< end if

<

< nmode = ior(nmode,NF90\_NETCDF4)

< #ifdef \_MPISERIAL

< ierr = nf90\_create(fname, nmode , File%fh)

< #else

< nmode = ior(nmode,NF90\_MPIIO)

< ierr = nf90\_create(fname, nmode, File%fh, &

< comm=File%iosystem%io\_comm, info=File%iosystem%info)

< #endif

< ! Set default to NOFILL for performance.

< ! if(ierr==PIO\_NOERR) ierr = nf90\_set\_fill(File%fh, NF90\_NOFILL, nmode)

< case(PIO\_iotype\_netcdf4c)

< if(iand(PIO\_64BIT\_OFFSET,amode)==PIO\_64BIT\_OFFSET) then

< nmode = ieor(amode,PIO\_64BIT\_OFFSET)

< else if(iand(PIO\_64BIT\_DATA,amode)==PIO\_64BIT\_DATA) then

< nmode = ieor(amode,PIO\_64BIT\_DATA)

< else

< nmode=amode

< end if

<

< nmode = ior(nmode,NF90\_NETCDF4)

<

< ! Only io proc 0 will do writing

< if (File%iosystem%io\_rank == 0) then

< ! Stores the ncid in File%fh

< ierr = nf90\_create(fname, nmode, File%fh, &

< info=File%iosystem%info )

< ! Set default to NOFILL for performance.

< if(ierr==PIO\_NOERR) &

< ierr = nf90\_set\_fill(File%fh, NF90\_NOFILL, nmode)

< endif

< #endif

183,191d139

< #ifdef \_NETCDF

< #ifdef \_NETCDF4

< if(ierr /= PIO\_NOERR) then ! try hdf5 format

< if(Debug) print \*, 'try netcdf4 format'

< File%iotype = pio\_iotype\_netcdf4p

< iotype = pio\_iotype\_netcdf4p

< end if

< #endif

< #endif

196,213d143

< #ifdef \_NETCDF4

< if(iotype==PIO\_iotype\_netcdf4p) then

< ! we need to go through some contortions to make sure a file we are opening is okay for parallel access

< ierr = nf90\_open(fname,amode,File%fh)

< ierr = nf90\_inquire(File%fh,formatnum=format)

< #ifndef MPI\_SERIAL

< if(format == nf90\_format\_netcdf4) then

< ierr = nf90\_close(File%fh)

< ierr = nf90\_open(fname, ior(amode,ior(NF90\_NETCDF4,NF90\_MPIIO)), File%fh, &

< comm=File%iosystem%io\_comm, info=File%iosystem%info)

< if(ierr==nf90\_enotnc4 .or. ierr==nf90\_einval) then

< ierr = nf90\_open(fname, amode, File%fh,info=File%iosystem%info)

< end if

< end if

< #endif

< end if

< #endif

<

344,350d273

< #ifdef \_NETCDF4

< if(File%iotype /= PIO\_IOTYPE\_NETCDF4C .and. &

< File%iotype /= PIO\_IOTYPE\_NETCDF4P) then

< print \*,'Changing file type to netcdf4p'

< File%iotype=pio\_iotype\_netcdf4c

< end if

< #else

353c276

< #endif

---

>

**vi pionfput\_mod.F90.in.patch**

写入以下内容并保存：

143,147d142

< #ifdef \_NETCDF4

< case (pio\_iotype\_netcdf4p)

< ierr=nf90\_var\_par\_access(File%fh, varid, NF90\_COLLECTIVE)

< ierr = nf90\_put\_var(File%fh, varid, ival, start=index)

< #endif

240,244d234

< #ifdef \_NETCDF4

< case (pio\_iotype\_netcdf4p)

< ierr=nf90\_var\_par\_access(File%fh, varid, NF90\_COLLECTIVE)

< ierr = nf90\_put\_var(File%fh, varid, ival, start=index)

< #endif

699,703d688

< #ifdef \_NETCDF4

< case(pio\_iotype\_netcdf4p)

< ierr=nf90\_var\_par\_access(File%fh, varid, NF90\_COLLECTIVE)

< ierr = nf90\_put\_var(File%fh, varid, ival, start=int(pstart), count=int(pcount))

< #endif

820,824d804

< #ifdef \_NETCDF4

< case(pio\_iotype\_netcdf4p)

< ierr=nf90\_var\_par\_access(File%fh, varid, NF90\_COLLECTIVE)

< ierr = nf90\_put\_var(File%fh, varid, ival, start=start, count=count)

< #endif

**vi pionfwrite\_mod.F90.in.patch**

写入以下内容并保存：

43,45d42

< #ifdef \_NETCDF4

< use netcdf, only : nf90\_var\_par\_access, nf90\_collective

< #endif

116,120d112

< #ifdef \_NETCDF4

< case(PIO\_iotype\_netcdf4p)

< ierr=nf90\_var\_par\_access(File%fh, vardesc%varid, NF90\_COLLECTIVE)

< ierr=nf90\_put\_var(File%fh, vardesc%varid, iobuf,start=int(start),count=int(count))

< #endif

为4个源码文件打上补丁。

**patch -p0 nf\_mod.F90 nf\_mod.F90.patch**

**patch -p0 ionf\_mod.F90 ionf\_mod.F90.patch**

**patch -p0 pionfput\_mod.F90.in pionfput\_mod.F90.in.patch**

**patch -p0 pionfwrite\_mod.F90.in pionfwrite\_mod.F90.in.patch**

制作shr\_sys\_mod.F90文件的补丁文件。

**cd /usr/local/CESM/CESM-release-cesm2.1.1/cime/src/share/util/**

**vi shr\_sys\_mod.F90.patch**

写入以下内容并保存：

91c91

< rcode = system(str)

---

> call system(str)

135c135

< rcode=chdir(path(1:lenpath))

---

> call chdir(path(1:lenpath))

为shr\_sys\_mod.F90文件打上补丁。

**patch -p0 shr\_sys\_mod.F90 shr\_sys\_mod.F90.patch**

检查组件安装情况。

**cd** **/usr/local/CESM/CESM-release-cesm2.1.1**

**./manage\_externals/checkout\_externals -S**

显示如下信息表示正常。

Processing externals description file : Externals.cfg

Processing externals description file : Externals\_CLM.cfg

Processing externals description file : Externals\_POP.cfg

Processing externals description file : Externals\_CISM.cfg

Checking status of externals: clm, fates, ptclm, mosart, ww3, cime, cice, pop, cvmix, marbl, cism, source\_cism, rtm, cam,

./cime

./components/cam

./components/cice

./components/cism

./components/cism/source\_cism

./components/clm

./components/clm/src/fates

./components/clm/tools/PTCLM

./components/mosart

./components/pop

./components/pop/externals/CVMix

./components/pop/externals/MARBL

./components/rtm

./components/ww3

进入CESM编译环境设置目录。

**cd /usr/local/CESM/CESM-release-cesm2.1.1/cime/config/cesm/machines**

制作config\_machines.xml文件的补丁。

**vi config\_machines.xml.patch**

写入以下内容并保存：

153c153

< <NODENAME\_REGEX>regex.expression.matching.your.machine</NODENAME\_REGEX>

---

> <NODENAME\_REGEX>openEuler</NODENAME\_REGEX>

170,171c170,171

< <MAX\_TASKS\_PER\_NODE>8</MAX\_TASKS\_PER\_NODE>

< <MAX\_MPITASKS\_PER\_NODE>8</MAX\_MPITASKS\_PER\_NODE>

---

> <MAX\_TASKS\_PER\_NODE>16</MAX\_TASKS\_PER\_NODE>

> <MAX\_MPITASKS\_PER\_NODE>16</MAX\_MPITASKS\_PER\_NODE>

176c176

< <arg name="ntasks"> -np {{ total\_tasks }} </arg>

---

> <arg name="ntasks">--allow-run-as-root --mca btl ^openib -np {{ total\_tasks }} </arg>

192,194c192,194

< <command name="load">compiler/gnu/8.2.0</command>

< <command name="load">mpi/3.3/gcc-8.2.0</command>

< <command name="load">tool/netcdf/4.6.1/gcc-8.1.0</command>

---

> <command name="load">/usr/local/bisheng/bisheng\_modulefiles</command>

> <command name="load">/usr/local/hmpi/hmpi\_modulefiles</command>

> <command name="load">/usr/local/NETCDF/netcdf\_modulefiles</command>

为config\_machines.xml文件打上补丁。

**patch -p0 config\_machines.xml config\_machines.xml.patch**

制作config\_compilers.xml文件的补丁。

**vi config\_compilers.xml.patch**

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）：

109c109

< <base> -std=gnu99 </base>

---

> <base> -std=gnu99 **-march=armv8.2-a -fsigned-char -ffp-contract=off** </base>

125c125

< <base> -fconvert=big-endian -ffree-line-length-none -ffixed-line-length-none </base>

---

> <base> -fconvert=big-endian **-ffree-form -ffree-line-length-none** **-march=armv8.2-a -mtune=tsv110** **–std=legacy** </base>

146c146

< <append compile\_threaded="true"> -fopenmp </append>

---

> <append compile\_threaded="true"> -fopenmp **-mabi=lp64** -fconvert=big-endian </append>

151,153c151,153

< <SCC> gcc </SCC>

< <SCXX> g++ </SCXX>

< <SFC> gfortran </SFC>

---

> <SCC> clang </SCC>

> <SCXX> clang++ </SCXX>

> <SFC> flang </SFC>

为config\_compilers.xml文件打上补丁。

**patch -p0 config\_compilers.xml config\_compilers.xml.patch**

----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。

修改主机名。

**hostname openEuler**

进入构建目录。

**cd /usr/local/CESM/CESM-release-cesm2.1.1/cime/scripts**

创建算例。

**./create\_newcase --case /usr/local/cesm/mycase --compset X --res f19\_g16 --input-dir=/usr/local/cesm/inputdata --output-root=/usr/local/cesm --handle-preexisting-dirs=r**

**cd /usr/local/cesm/mycase**

注意：

（1）如果执行./create\_newcase时，不指定--input-dir参数，当运行./check\_input\_data下载数据时，数据会默认下载保存到/root/cesm/inputdata；如果指定--input-dir=/usr/local/cesm/inputdata参数，下载数据会保存到指定路径：/usr/local/cesm/inputdata。执行命令./xmlquery DIN\_LOC\_ROOT，可以查看当前的数据保存路径。

（2）指定参数--output-root=/usr/local/cesm，编译CESM后可执行程序会保存在指定路径/usr/local/cesm/mycase。

（3）指定参数--handle-preexisting-dirs=r，避免创建算例中的交互。

下载输入数据文件。

**./check\_case**

**./check\_input\_data --download**

注意：

（1）由于CESM会根据模拟情景下载对应的输入数据文件，由于数据量较大且从美国UCAR的服务器上下载，可以预先准备好输入数据文件。

（2）从美国UCAR服务器上下载数据，不能使用国外代理，否则会拒绝访问。

（3）将准备好的输入数据文件存放于/usr/local/cesm/inputdata目录下，并检验数据的完整性。

重新配置编译环境。

**./case.setup --clean**

**./case.setup**

编译CESM。

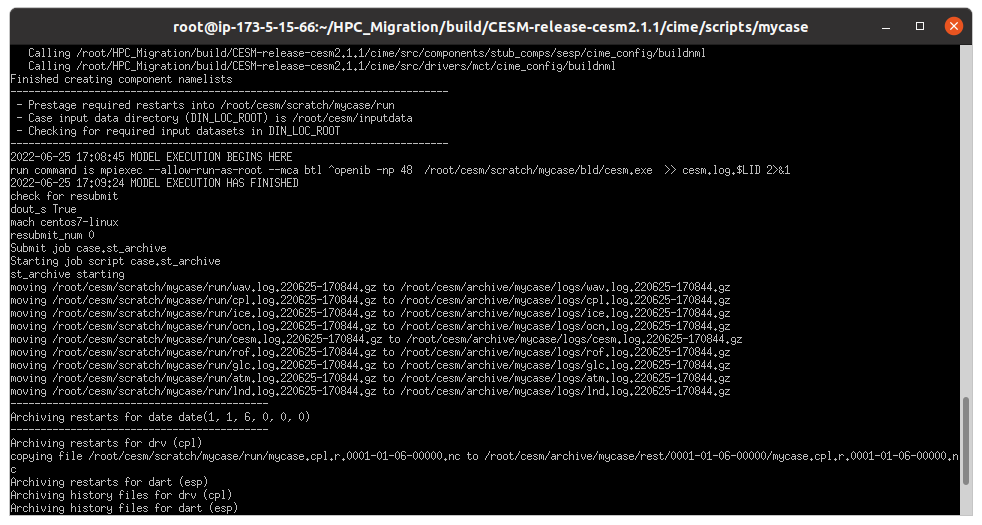
**./case.build**

运行CESM。

**./case.submit**

当CESM运行的结尾处出现如图所示内容时，表示CESM程序运行正常结束。

需要查看输出到屏幕中的日志，手动计算出时间差，数值越少性能越优。如图，算例开始时间为“2022-06-25 17:08:45”，结束时间为“2022-06-25 17:09:24”，那么此算例运行时间为39s。



----结束

# 修订记录

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