|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | |  |
| HPX-1.8.1 | | |  |
| 应用**移植指南** | | |
| **文档版本** | **01** | |
| **发布日期** | **2022-07-30** | |
| https://gimg2.baidu.com/image_search/src=http%3A%2F%2Fimg.gaosan.com%2Fupload%2F202108%2F20210803134739830.jpg&refer=http%3A%2F%2Fimg.gaosan.com&app=2002&size=f9999,10000&q=a80&n=0&g=0n&fmt=auto?sec=1667123127&t=8903f6576c856e02bb2b9708878762bb | | | | |
|  | 中国地质大学（武汉） | |  |  |

|  |
| --- |
| 版权所有 © 中国地质大学（武汉）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  为中国地质大学（武汉）的图标。  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。 |

|  |  |
| --- | --- |
| 中国地质大学（武汉） | |
| 地址： | 湖北省武汉市洪山区鲁磨路388号中国地质大学 邮编：430074 |
| 网址： | <https://www.cug.edu.cn> |
| 客户服务邮箱： | [support@cug.edu.com](mailto:support@cug.edu.com) |
| 客户服务电话： | 13971209102 |

目录

[目录 ii](#_Toc108207765)

[1 介绍 1](#_Toc108207766)

[2 环境要求 2](#_Toc108207767)

[3 移植规划数据 4](#_Toc108207768)

[4 配置编译环境 5](#_Toc108207769)

[4.1 搭建鲲鹏基座软件环境 5](#_Toc108207770)

[5 获取源码 6](#_Toc108207771)

[6 编译和安装 7](#_Toc108207772)

[7 运行和验证 10](#_Toc108207773)

[8 修订记录 11](#_Toc108207774)

# 介绍

HPX运行时系统，是由美国路易斯安那州立大学的STE||AR研究组开发的，用于提供实现并行化和并发操作的所有C++标准设施，拓展到分布式并行环境。HPX是对传统的并行化方法的再设计，基于重叠计算与通信的异步并行方法，增加HPC应用的扩展性。

关于HPX的更多信息请访问[HPX官网](https://hpx.stellar-group.org)。

语言：C++

一句话描述：异步并行计算运行时系统。

开源协议：Boost Software License

建议的版本

建议使用版本为“HPX-1.8.1”。

# 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 | AMD |

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 |
| --- | --- | --- |
|  |  |  |
|  |  |  |
| HPX | 1.8.1 | https://github.com/STEllAR-GROUP/hpx/archive/refs/tags/1.8.1.tar.gz |
| CMAKE | 3.23.3 | https://github.com/Kitware/CMake/releases/download/v3.23.3/cmake-3.23.3-linux-aarch64.tar.gz |
| asio | 1.21.0 | https://github.com/chriskohlhoff/asio/archive/refs/tags/asio-1-21-0.tar.gz |
| gperftools | 2.6.1 | https://github.com/gperftools/gperftools/archive/gperftools-2.6.1.tar.gz |
| hwloc | 2.7.1 | https://download.open-mpi.org/release/hwloc/v2.7/hwloc-2.7.1.tar.gz |
| boost | 1.75.0 | https://boostorg.jfrog.io/artifactory/main/release/1.75.0/source/boost\_1\_75\_0.tar.gz |
| HPXCL | 0.1-alpha | https://github.com/STEllAR-GROUP/hpxcl/archive/refs/tags/v0.1-alpha.tar.gz |
| APEX | 2.5.1 | https://github.com/UO-OACISS/apex/archive/refs/tags/v2.5.1.tar.gz |
| Active Harmony |  | https://dyninst.org/harmony |

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 |
| --- | --- | --- |
| Ubuntu | Ubuntu 20.04 |  |
| Kernel |  |  |

# 移植规划数据

本章节给出HPX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 |  | 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 |  |
| 3 | /usr/local/asio | ASIO的安装规划路径。 |
| 4 | /usr/local/gperftools | google-gperftools的安装规划路径。 |
| 5 | /usr/local/hwloc | HWLOC的安装规划路径。 |
| 6 | /usr/local/boost | BOOST的安装规划路径。 |
| 7 | /usr/local/hpx | HPX的安装规划路径。 |
| 8 | /usr/local/hpxcl | HPXCL的安装规划路径。 |

# 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asio | 参考4.2安装ASIO |
| 3 | 安装gperftools | 参考4.3安装gperftools |
| 4 | 安装 | 参考4.4安装 |
| 5 | 安装 | 参考4.5安装 |
| 6 | 安装 | 参考4.6安装 |
| 7 | 安装 | 参考4.7安装 |
| 8 | 安装 | 参考4.8安装 |

## 搭建基座软件环境

安装GNU编译器

安装不同版本的编译器。

**apt install gcc-9 g++-9 gfortran-9**

版本切换设置。

安装OpenMPI

## 安装ASIO

操作步骤

使用PuTTY工具，以root用户登录服务器。

解压asio安装包。

**tar -xvf asio-1-21-0.tar.gz**

注：解压缩时请以实际压缩包名称为准。

设置和编译asio。

**cd asio-asio-1-21-0**

**./autogen.sh**

**CC=`which clang`**

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

**./configure --prefix=/usr/local/asio-1.21.0**

**make -j$(nproc)**

**make install**

配置环境变量。

**cat>"/usr/local/asio-1.21.0/asio\_modulefiles"<<EOF**

**#%Module1.0**

**conflict asio**

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

**proc getModulefileDir {} {**

**variable modfile**

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

**return \$modfile\_path**

**}**

**set pwd [getModulefileDir]**

**set ASIO \$pwd**

**setenv ASIO \$ASIO**

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

**prepend-path INCLUDE \$ASIO/include**

**EOF**

## 安装google-gperftools

介绍

TCMalloc(Thread-Caching Malloc)与标准glibc库的malloc实现一样的功能，但是TCMalloc在效率和速度效率都比标准malloc高很多。TCMalloc是google-perftools工具中的一个（gperftools四个工具分别是：TCMalloc、heap-checker、heap-profiler和cpu-profiler），这个工具是开源的，以源码形式发布。如果觉得自己维护一个内存分配器麻烦的话，可以考虑将TCMalloc静态库连接到你的程序中。使用的时候和glibc中的malloc调用方式一模一样。你需要做的只是把TCMalloc的动态库或者静态库连接进你的程序中，你就可以获得一个高效，快速，安全的内存分配器。

与标准的glibc库的malloc相比，TCMalloc在内存的分配效率和速度要高，可以在高并发的情况下很好的控制内存的使用，提高服务器的性能，降低负载。

HPX使用TCMalloc分配内存空间。

操作步骤

解压安装包。

**tar -xvf gperftools-2.6.1.tar.gz**

注：解压缩时请以实际压缩包名称为准。

配置和编译源码。

**cd gperftools-gperftools-2.6.1**

**./autogen.sh**

**CC=`which clang`**

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

**./configure --prefix=/usr/local/gperftools-2.6.1**

**make -j$(nproc)**

**make install**

配置环境变量。

**cat>"/usr/local/gperftools-2.6.1/gperftools\_modulefiles"<<EOF**

**#%Module1.0**

**conflict gperftools**

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

**proc getModulefileDir {} {**

**variable modfile**

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

**return \$modfile\_path**

**}**

**set pwd [getModulefileDir]**

**set GPERFTOOLS \$pwd**

**setenv GPERFTOOLS \$GPERFTOOLS**

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

**prepend-path INCLUDE \$GPERFTOOLS/include**

**EOF**

## 安装hwloc

操作步骤

解压安装包。

**tar -xvf hwloc-2.7.1.tar.gz**

注：解压缩时请以实际压缩包名称为准。

配置和编译源码。

**cd hwloc-2.7.1**

**./autogen.sh**

**CC=`which clang`**

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

**./configure --prefix=/usr/local/hwloc-2.7.1**

**make -j$(nproc)**

**make install**

配置环境变量。

**cat>"/usr/local/hwloc-2.7.1/hwloc\_modulefiles"<<EOF**

**#%Module1.0**

**conflict hwloc**

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

**proc getModulefileDir {} {**

**variable modfile**

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

**return \$modfile\_path**

**}**

**set pwd [getModulefileDir]**

**set HWLOC \$pwd**

**setenv HWLOC \$HWLOC**

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

**prepend-path INCLUDE \$HWLOC/include**

**EOF**

在当前shell中加载环境变量。

**module use /usr/local/hwloc-2.7.1/**

**module load /usr/local/hwloc-2.7.1/hwloc\_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/hwloc-2.7.1/**

**module load /usr/local/hwloc-2.7.1/hwloc\_modulefiles**

----结束

## 安装boost

操作步骤

使用PuTTY工具，以root用户登录服务器。

安装依赖库。

**yum -y install gcc gcc-c++ python python-devel libicu libicu-devel zlib zlib-devel bzip2 bzip2-devel**

解压Boost安装包。

**tar -xvf boost\_1\_75\_0.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 boost\_1\_75\_0**

进行编译配置。

**./bootstrap.sh --with-toolset=clang**

编译，安装。

**./b2 install --prefix=/usr/local/BOOST**

配置环境变量。

**cat>"/usr/local/BOOST/boost\_modulefiles"<<EOF**

**#%Module1.0**

**conflict boost**

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

**proc getModulefileDir {} {**

**variable modfile**

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

**return \$modfile\_path**

**}**

**set pwd [getModulefileDir]**

**set BOOST \$pwd**

**setenv BOOST \$BOOST**

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

**EOF**

在当前shell中加载环境变量

**module use /usr/local/BOOST/**

**module load /usr/local/BOOST/boost\_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/BOOST/**

**module load /usr/local/BOOST/boost\_modulefiles**

----结束

## 安装APEX

操作步骤

# 获取源码

操作步骤

下载HPX-1.8.1安装包

下载地址：https://github.com/STEllAR-GROUP/hpx/archive/refs/tags/1.8.1.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。

安装环境依赖包

**yum install -y environment-modules patch cmake wget cmake**

**set -e**

获取源码并解压

**wget -c https://github.com/STEllAR-GROUP/hpx/archive/refs/tags/1.8.1.tar.gz**

进入解压后的目录。

**cd hpx-1.8.1**

编译，安装。

**mkdir build**

**cd build**

**cmake \**

**-DCMAKE\_CXX\_COMPILER=clang++ \**

**-DCMAKE\_CXX\_COMPILER\_AR=llvm-ar \**

**-DCMAKE\_BUILD\_TYPE=Release \**

**-DCMAKE\_CXX\_FLAGS\_RELEASE=-O2 \**

**-DCMAKE\_INSTALL\_PREFIX=/usr/local/hpx1.8.1 \**

**-DBoost\_INCLUDE\_DIR=/usr/local/boost-1.75.0/include \**

**-DASIO\_INCLUDE\_DIR=/usr/local/asio-1.21.0/include \**

**-DHPX\_WITH\_MALLOC=tcmalloc \**

**-DHWLOC\_INCLUDE\_DIR=/usr/local/hwloc-2.7.1/include \**

**-DHWLOC\_LIBRARY=/usr/local/hwloc-2.7.1/lib/libhwloc.so \**

**-DHPX\_WITH\_GENERIC\_CONTEXT\_COROUTINES=ON \**

**-DHPX\_WITH\_ASYNC\_MPI=ON \**

**-DHPX\_WITH\_CUDA=ON \**

**-DTCMALLOC\_LIBRARY=/usr/local/gperftools-2.6.1/lib/libtcmalloc.so \**

**-DTCMALLOC\_INCLUDE\_DIR=/usr/local/gperftools-2.6.1/include \**

**-DHPX\_WITH\_MAX\_CPU\_COUNT=256 \**

**-DHPX\_WITH\_DYNAMIC\_HPX\_MAIN=OFF \**

**-DHPX\_WITH\_EXAMPLES=OFF \**

**..**

**make -j$(nproc) && make install**

配置环境变量。

**cat>"/usr/local/hpx-1.8.1/hpx\_modulefiles"<<EOF**

**#%Module1.0**

**conflict hpx**

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

**proc getModulefileDir {} {**

**variable modfile**

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

**return \$modfile\_path**

**}**

**set pwd [getModulefileDir]**

**set HPX \$pwd**

**setenv HPX \$HPX**

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

**prepend-path INCLUDE \$HPX/include**

**EOF**

在当前shell中加载环境变量。

**module use /usr/local/hpx-1.8.1**

**module load /usr/local/hpx-1.8.1/hpx\_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/hpx-1.8.1**

**module load /usr/local/hpx-1.8.1/hpx\_modulefiles**

----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。

在当前shell中加载环境变量。

**module use /usr/local/hpx-1.8.1**

**module load /usr/local/hpx-1.8.1/hpx\_modulefiles**

执行计算。

运行成功后将显示下图：

----结束

# 修订记录

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