openGauss 2.1.0 编译指导书

文档版本 01

发布日期 2021-09-30

华为技术有限公司

商标声明


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暗示的声明或保证。

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

客户服务电话： 4008302118

目 录

1 简介

目的

本文档帮助读者快速了解编译openGauss所需的软硬件要求、环境配置，以及如何从源码编译出软件或者安装包。

概述

本文档介绍了openGauss对于操作系统的要求、编译环境的要求、软件依赖、编译方法以及编译结果的存放位置等。

2 搭建编译环境

硬件要求

编译openGauss的硬件要求：

- 机器数量：1台
- 机器硬件规格：
  - CPU: 4U
  - Memory: 8G
  - Free Disk：100G (Linux 64位)

软件要求

操作系统要求

openGauss支持的操作系统：

- CentOS 7.6 (x86 架构)
- openEuler-20.03-LTS (aarch64 架构)
- openEuler-20.03-LTS (X86 架构)
- Kylin-V10 (aarch64 架构)

软件依赖要求

编译openGauss的软件依赖要求如**表 软件依赖要求**所示。

建议使用上述操作系统安装光盘或者源中，下列依赖软件的默认安装包，若不存在下列软件，可参看软件对应的建议版本。

表 2-1 软件依赖要求

所需软件	建议版本
libaio-devel	建议版本：0.3.109-13
flex	要求版本：2.5.31 以上
bison	建议版本：2.7-4

所需软件	建议版本
ncurses-devel	建议版本：5.9-13.20130511
glibc-devel	建议版本：2.17-111
patch	建议版本：2.7.1-10
redhat-lsb-core	建议版本：4.1
readline-devel	建议版本：7.0-13

环境变量配置

编译openGauss的环境变量配置已经统一写入一键式编译和打包脚本，因此无需自行配置。

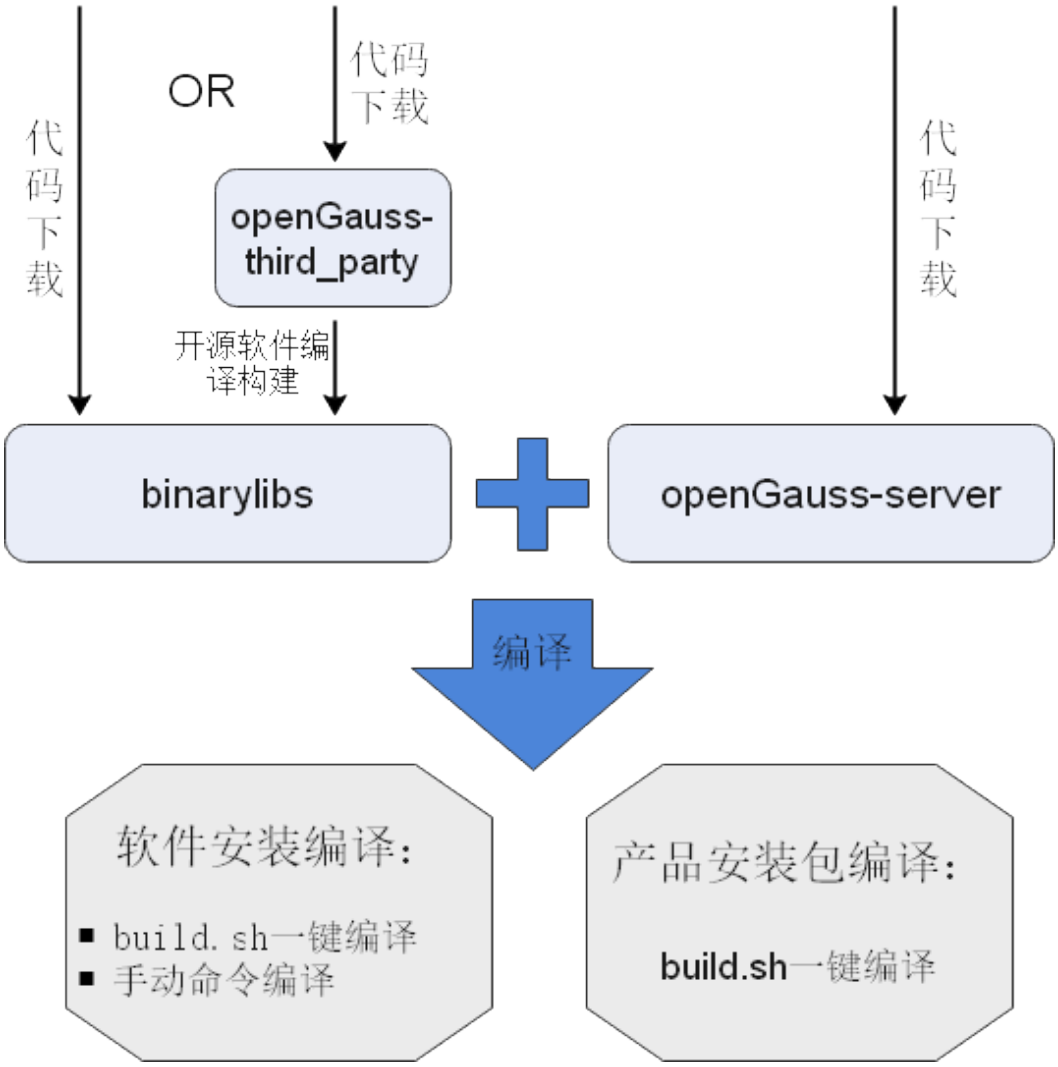
若想不使用一键式编译脚本，则需要手动配置环境变量，将在**3.2 软件安装编译**介绍。

3 版本编译

openGauss的编译过程和生成安装包的过程已经写成了一个一键式的脚本build.sh，可以方便的通过脚本进行编译操作。也可以自己配置环境变量，通过命令进行编译。

本章节主要介绍openGauss版本编译，编译过程如**图3-1**所示，详细内容参见下文。

图 3-1 openGauss 编译流程



3.1 编译前准备

代码下载

前提条件

已在本地安装并配置git和git-lfs。

操作步骤

步骤1 执行如下命令下载代码和开源第三方软件仓库等，其中*[git ssh address]*表示实际代码下载地址，可在openGauss社区获取这些地址。


```
[user@linux sda]$ git clone [git ssh address] openGauss-server
[user@linux sda]$ git clone [git ssh address] openGauss-third_party
[user@linux sda]$ # mkdir binarylibs       关于此注释步骤，请阅读说明
```

说明

- openGauss-server：openGauss的代码仓库。
- openGauss-third_party：openGauss依赖的开源第三方软件仓库。
- binarylibs：存放编译构建好的开源第三方软件的文件夹，用户可通过**开源软件编译构建**获取。由于开源软件编译构建耗时长，我们特地使用openGauss-third_party编译构建出了一份binarylibs并压缩上传到了网上，用户可以直接下载获取。

  下载地址：https://opengauss.obs.cn-south-1.myhuaweicloud.com/2.0.0/openGauss-third_party_binarylibs.tar.gz

  下载完毕后请解压，重命名文件夹为 binarylibs。

步骤2 下载项进度均显示为100%时表示下载成功。

----结束

开源软件编译构建

开源软件编译构建

openGauss的编译，需要提前把所依赖的开源第三方软件进行编译和构建。这些开源第三方软件存放在代码openGauss-third_party代码仓中，用户下载完毕之后应用git lfs pull获取代码仓中的大文件，并且用户通常只需要构建一次。若存在开源软件版本更新，则需要重新构建。

由于此步骤耗时较长，我们使用openGauss-third_party编译构建出了一份binarylibs，用户可以参考**代码下载**直接下载获取。

表 3-1 openGauss 开源第三方软件编译前置软件要求

所需软件	建议版本
python3	建议版本：3.6
python3-devel	建议版本：3
setuptools	建议版本：36.6.1
libaio-devel	建议版本：0.3.109-13
flex	要求版本：2.5.31 以上
ncurses-devel	建议版本：5.9-13.20130511
lsb_release	建议版本：4.1
pam-devel	建议版本：1.1.8-1.3.1
libffi-devel	建议版本：3.1
patch	建议版本：2.7.1-10
golang	建议版本：1.13.3及以上
autoconf	建议版本：2.69

所需软件	建议版本
automake	建议版本：1.13.4
byacc	建议版本：1.9
cmake	建议版本：3.19.2
diffutils	建议版本：3.7
openssl-devel	建议版本：1.1.1
libtool	建议版本：2.4.2及以上
libtool-devel	建议版本：2.4.2及以上

在开始编译第三方库之前，请自行准备好gcc7.3。建议用已发布的编译好的第三方库中gcc，并配置好环境变量。

在安装完**表 openGauss开源三方件编译前置软件要求**中的软件后，请将python默认版本指向python3.x并执行如下操作：

步骤1 执行如下命令进入内核依赖的开源第三方软件目录，进行开源第三方软件的编译和构建，产生相应的二进制程序或库文件。/sda/openGauss-third_party为开源第三方软件下载目录。

```
[user@linux sda]$ cd /sda/openGauss-third_party/build
[user@linux build]$ sh build_all.sh
```

步骤2 用户执行以上命令之后，可以自动生成数据库编译所需的开源第三方软件，如果想单独的生成某个开源三方软件，可以进入对应的目录，执行build.sh脚本，如：

```
[user@linux sda]$ cd /sda/openGauss-third_party/dependency/openssl
[user@linux openssl]$ sh build.sh
```

即可编译生成openssl。

说明

相关的报错日志可以查看对应的build目录下对应名字的log以及对应模块下的log，如dependency模块下的openssl的相关编译安装日志可以查看：

- /sda/openGauss-third_party/build/dependency_build.log。
- /sda/openGauss-third_party/dependency/build/openssl_build.log。
- /sda/openGauss-third_party/dependency/openssl/build_openssl.log。

----结束

编译构建结果

执行上述脚本，最终编译构建出的结果会存放在openGauss-third_party目录下的output目录。这些文件会在后面编译openGauss-server时使用到。

build.sh 介绍

openGauss-server/build.sh是编译过程中的重要脚本工具。其集成了软件安装编译、产品安装包编译两种功能，可快速进行代码编译和打包。

详细参数选项如下表所示：

表 3-2 build.sh 参数功能选项介绍。

功能选项	缺省值	参数	功能
-h	不使用此选项	-	帮助菜单。
-m	release	[debug | release | memcheck]	选择编译目标版本。
-3rd	${代码路径}/binarylibs	[binarylibs path]	指定binarylibs的路径，需绝对路径。
-pkg	不使用此功能	-	将代码编译结果压缩封装成安装包。
-nopt	不使用此功能	-	如果使用此功能，则对鲲鹏平台的相关CPU不进行优化。

说明

1. -m [debug | release | memcheck] 表示可选择三种目标版本：
   - release: 代表生成release版本的二进制程序，该版本编译时，配置GCC高级别优化选项，去除内核调试代码，通常用于生产环境或性能测试环境。
   - debug：代表生成debug版本的二进制程序，该版本编译时，增加内核代码调试功能，通常用于开发自测环境。
   - memcheck：代表生成memcheck版本的二进制程序，该版本编译时，在debug版本基础上新增ASAN功能，通常用于定位内存问题。
2. -3rd [binarylibs path] 为binarylibs的路径。缺省情况下，会认为当前代码文件夹下存在binarylibs。因此如果将binarylibs移动到openGauss-server下，或在openGauss-server下创建了指向binarylibs的软链接，可不指定此选项。但需要注意其容易被git clean等操作删除。
3. 此脚本每个参数选项都设置了缺省值，且数量并不多，依赖关系简单，因此使用时非常方便。如果用户需求值与缺省值不同，请根据实际情况进行设置。

3.2 软件安装编译

软件安装编译即将代码编译生成软件，并将软件安装到机器上。提供一键式编译脚本build.sh进行操作，也可以自己配置环境变量手动操作。两种方式将在本章节的一键式脚本编译操作步骤、手动编译操作步骤中进行讲解

前提条件

- 已按照**搭建编译环境**的要求准备好相关软硬件，并且已参考**代码下载**下载了代码。
- 已完成开源软件编译构建，具体请参见**开源软件编译构建**。并将gcc7.3按已发布的编译好的第三方库目录结构放置在output目录中。
- 了解 **build.sh**脚本的参数选项和功能。
- 代码环境干净，没有以前编译生成的文件。具体请参见**FAQ 4.1**。

一键式脚本编译

步骤1 执行如下命令进入到软件代码编译脚本目录。

```
[user@linux sda]$ cd /sda/openGauss-server
```

步骤2 执行如下命令，编译安装openGauss。

```
[user@linux openGauss-server]$ sh build.sh -m [debug | release | memcheck] -3rd [binarylibs path]
```

例如：

```
sh build.sh       # 编译安装release版本的openGauss。需代码目录下有binarylibs或者其软链接，否则将会失败。
sh build.sh -m debug -3rd /sdc/binarylibs          # 编译安装debug版本的openGauss
```

步骤3 显示如下内容，表示编译成功。

```
make compile sucessfully!
```

- 编译后软件安装路径为：/sda/openGauss-server/mppdb_temp_install。
- 编译后的二进制放置路径为：/sda/openGauss-server/mppdb_temp_install/bin。
- 编译日志为：./build/script/makemppdb_pkg.log。

步骤4 导入环境变量，即可进行初始化和启动数据库。

```
export CODE_BASE=_____     # openGauss-server的路径
export GAUSSHOME=$CODE_BASE/mppdb_temp_install/
export LD_LIBRARY_PATH=$GAUSSHOME/lib::$LD_LIBRARY_PATH
export PATH=$GAUSSHOME/bin:$PATH
```

----结束

手动编译

步骤1 执行如下命令进入到软件代码目录。

```
[user@linux sda]$ cd /sda/openGauss-server
```

步骤2 执行脚本获取自己系统的版本

```
[user@linux openGauss-server]$ sh src/get_PlatForm_str.sh
```

说明

- 显示的结果表示openGauss当前支持的操作系统，openGauss支持的操作系统为centos7.6_x86_64、openeuler_aarch64。
- 如果结果显示为 Failed 或者其他版本，表示openGauss不支持当前操作系统。

步骤3 配置环境变量，根据自己的代码下载位置补充两处"____"，并将**步骤2**获取到的结果替换下面的***。

```
export CODE_BASE=_____     # openGauss-server的路径
export BINARYLIBS=_____    # binarylibs的路径
export GAUSSHOME=$CODE_BASE/dest/
export GCC_PATH=$BINARYLIBS/buildtools/***/gcc7.3/
export CC=$GCC_PATH/gcc/bin/gcc
export CXX=$GCC_PATH/gcc/bin/g++
export LD_LIBRARY_PATH=$GAUSSHOME/lib:$GCC_PATH/gcc/lib64:$GCC_PATH/isl/lib:$GCC_PATH/mpc/lib/:$GCC_PATH/mpfr/lib/:$GCC_PATH/gmp/lib/:$LD_LIBRARY_PATH
export PATH=$GAUSSHOME/bin:$GCC_PATH/gcc/bin:$PATH
```

步骤4 选择版本进行configure。

debug版：

```
./configure --gcc-version=7.3.0 CC=g++ CFLAGS='-O0' --prefix=$GAUSSHOME --3rd=$BINARYLIBS --enable-debug --enable-cassert --enable-thread-safety --with-readline --without-zlib
```


release版：

```
./configure --gcc-version=7.3.0 CC=g++ CFLAGS="-O2 -g3" --prefix=$GAUSSHOME --3rd=$BINARYLIBS --enable-thread-safety --with-readline --without-zlib
```

memcheck版：

```
./configure --gcc-version=7.3.0 CC=g++ CFLAGS='-O0' --prefix=$GAUSSHOME --3rd=$BINARYLIBS --enable-debug --enable-cassert --enable-thread-safety --with-readline --without-zlib --enable-memory-check
```

说明

- [debug | release | memcheck]表示可选择三种目标版本，三种目标版本如下所示：
  - release: 代表生成release版本的二进制程序，该版本编译时，配置GCC高级别优化选项，去除内核调试代码，通常用于生产环境或性能测试环境。
  - debug：代表生成debug版本的二进制程序，该版本编译时，增加内核代码调试功能，通常用于开发自测环境；
  - memcheck：代表生成memcheck版本的二进制程序，该版本编译时，在debug版本基础上新增ASAN功能，通常用于定位内存问题。
- 在ARM平台上，CFLAGS需要添加 -D__USE_NUMA 。
- 在ARMv8.1或者更高的平台上(例如鲲鹏920)，CFLAGS需要添加 -D__ARM_LSE 。
- 若将binarylibs移动到openGauss-server下，或在openGauss-server下创建了指向binarylibs的软链接，可不指定--3rd参数。但这样做的话需要注意其容易被git clean等操作删除。
- 如需使用MOT，需要在命令中添加 --enable-mot。

步骤5 执行如下命令，编译安装。

```
[user@linux openGauss-server]$ make -sj
[user@linux openGauss-server]$ make install -sj
```

步骤6 显示如下内容，表示编译安装成功。

```
openGauss installation complete.
```

- 编译后软件安装路径为：$GAUSSHOME
- 编译后的二进制放置路径为：$GAUSSHOME/bin

----结束

3.3 产品安装包编译

安装包编译即将代码编译生成软件安装包，安装包的编译打包过程也集成在build.sh之中。

前提条件

- 已按照**搭建编译环境**的要求准备好相关软硬件，并且已参考**代码下载**下载了代码。
- 已完成开源软件编译构建，具体请参见**开源软件编译构建**。
- 了解 **build.sn**脚本的参数选项和功能。
- 代码环境干净，没有以前编译生成的文件。具体请参见**FAQ 4.1**。

操作步骤

步骤1 执行如下命令进入到代码目录。

```
[user@linux sda]$ cd /sda/openGauss-server
```


步骤2 执行如下命令编译出openGauss产品安装包。

```
[user@linux openGauss-server]$ sh build.sh -m [debug | release | memcheck] -3rd [binarylibs path] -pkg
```

例如：

```
sh build.sh -pkg       # 生成release版本的openGauss安装包。需代码目录下有binarylibs或者其软链接，否则将会失败。
sh build.sh -m debug -3rd /sdc/binarylibs -pkg          # 生成debug版本的openGauss安装包
```

本操作和**3.2 软件安装编译**相比，同样会经历的一键式编译最终生成软件的过程、与将软件封装成安装包的过程。对比**3.2软件安装编译**中build.sh的使用命令可发现，此处仅增加了一个‘-pkg’功能选项。

步骤3 显示如下内容，表示安装包编译成功。

```
success!
```

- 生成的安装包会存放在./output目录下。
- 编译日志为：./build/script/makemppdb_pkg.log。
- 安装包打包日志为：./build/script/make_package.log。

----结束

4 FAQ

4.1 如何清除编译过程中生成的临时文件

问题

如何清除编译过程中生成的临时文件。

回答

进入/sda/openGauss-server目录，选择执行如下命令清除编译过程中生成的临时文件。

- 删除由configure和make生成的文件。
  ```
  make distclean -sj
  ```
- 删除make生成的文件。
  ```
  make clean -sj
  ```

4.2 如何解决“Configure error: C compiler cannot create executables”报错

问题

如何解决版本编译时出现的“Configure error: C compiler cannot create executables”报错。

回答

报错原因：binarylibs文件不完整或者被损坏。

解决办法：若binarylibs是通过开源软件构建而来，请重新构建开源第三方软件；若binarylibs是代码下载而来，请重新下载。最后重新执行当前脚本或命令。

4.3 如何解决 "g++: fatal error: Killed signal terminated program cclplus" 报错

问题

如何解决编译过程中出现的 "g++: fatal error: Killed signal terminated program cclplus" 报错。

回答

报错原因：脚本中的编译过程都添加了-sj参数，并发数太大导致错误。

解决办法：编译过程中降低make 并发数，或者直接使用make命令。使用一键式脚本的话需要修改脚本。

4.4 如何解决"out of memory allocating xxx bytes after a total of xxx bytes"报错

问题

如何解决编译过程中出现的“out of memory allocating xxx bytes after a total of xxx bytes”报错。

回答

报错原因：脚本中的编译过程都添加了-sj参数，同时机器配置较低，内存不足，并发数太大导致错误。

解决办法：编译过程中降低make 并发数，或者直接使用make命令。使用一键式脚本的话需要修改脚本。

