

openGauss的DB4AI特性实践

实验指导

1 ECS 弹性云服务器购买

1.1 登录华为云

步骤 1 进入华为云官网。

华为云官网：<https://www.huaweicloud.com/>，进入华为云官网，点击登录。



步骤 2 输入账号名和密码，点击登录。

步骤 3 点击右上角控制台，进入产品管理环境。



1.2 准备虚拟私有云 VPC 环境

步骤 1 在控制台页面下，直接点击“虚拟私有云 VPC”



或点击左侧目录找到“虚拟私有云 VPC”



步骤 2 在网络控制台界面中，点击“创建虚拟私有云”。



步骤3 填写如下配置信息，然后点击“立即创建”。

基本信息

- 区域：华北-北京四
- 名称：myvpc（可自定义）
- 网段：默认

子网配置

- 可用区：可用区1
- 名称：subnet-myvpc（可自定义）
- 子网网段：默认

创建虚拟私有云 [?](#) [返回虚拟私有云列表](#)

基本信息

*** 区域** 华北-北京四
不同区域的资源之间内网不互通。请选择靠近您客户的区域，可以降低网络时延、提高访问速度。

*** 名称** vpc-myvpc

*** 网段** 192 · 168 · 0 · 0 / 16
建议使用网段：10.0.0.0/8~24, 172.16.0.0/12~24, 192.168.0.0/16~24

标签
如果您需要使用同一标签标识多种云资源，即所有服务均可在标签输入框下拉选择同一标签，建议在TMS中创建预定义标签。 [查看预定义标签](#) [C](#)
标签键 标签值
您还可以添加10个标签。

子网配置

默认子网

*** 可用区** [?](#) 可用区1 可用区2 可用区3

*** 名称** subnet-myvpc

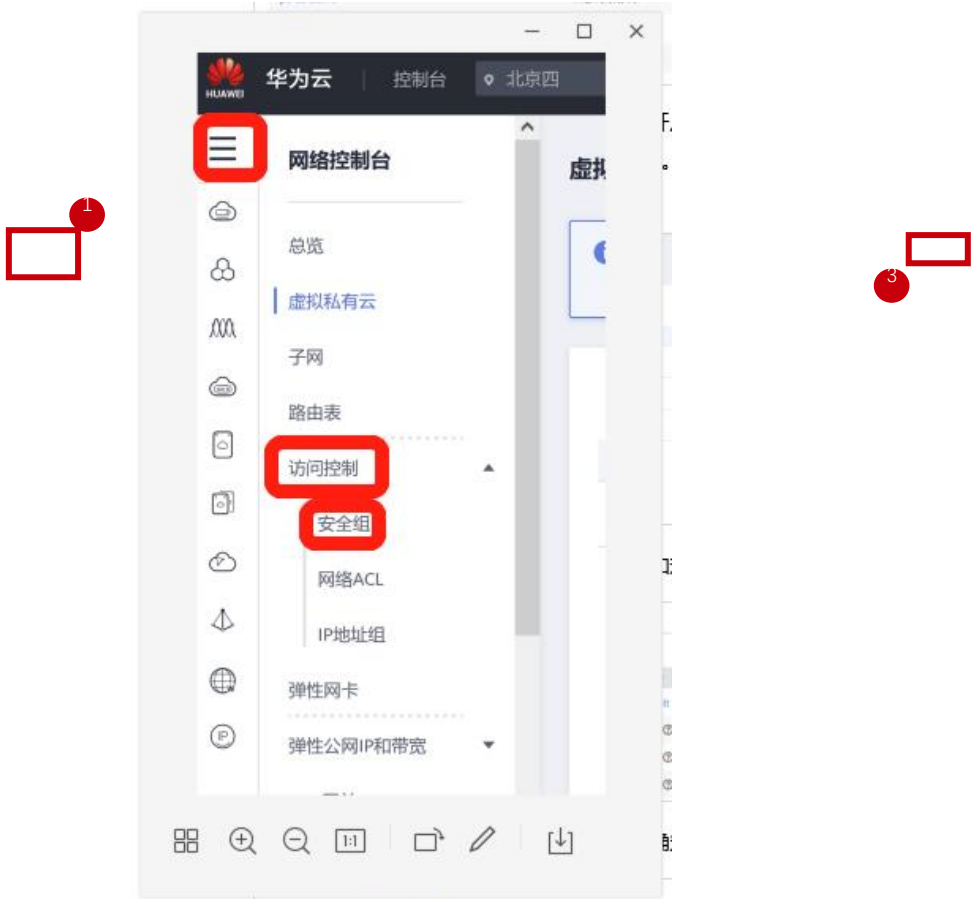
*** 子网网段** 192 · 168 · 0 · 0 / 24 [?](#)
可用IP数:251
子网创建完成后，子网网段无法修改

高级配置 默认配置 自定义配置

步骤4 返回到网络控制台，可看见VPC与子网已创建。

名称	IPv4网段	状态	子网个数	操作
vpc-headquater	192.168.0.0/16	正常	1个	修改 删除
vpc-db-active	172.16.0.0/16	正常	2个	修改 删除
my	192.168.0.0/16	正常	1个	修改 删除
myvpc	192.168.0.0/16	正常	1个	修改 删除

步骤 5 在左侧菜单栏选择“访问控制”，展开后找到“安全组”，点击进入



步骤 6 在“安全组”页面中找到 default 安全组。点击“配置规则”。

名称	安全组规则	关联实例	描述	操作
default	7	0	Default security group	配置规则 管理实例 克隆

步骤 7 选择“入方向规则”页签，点击“添加规则”。

基本信息		入方向规则	出方向规则	关联实例
添加规则 快速添加规则 删除 一键放通		入方向规则：4 教我设置		
<input type="checkbox"/> 协议端口	类型	源地址	描述	操作
<input type="checkbox"/> 全部	IPv4	Sys-default	--	修改 复制 删除
<input type="checkbox"/> TCP : 22	IPv4	0.0.0.0/0	Permit default Linux SSH port.	修改 复制 删除
<input type="checkbox"/> TCP : 3389	IPv4	0.0.0.0/0	Permit default Windows remote desktop port.	修改 复制 删除
<input type="checkbox"/> TCP : 1-65535	IPv4	0.0.0.0/0	--	修改 复制 删除

步骤 8 协议端口选择“全部放通”，点击“确定”，完成安全组规则的添加。

添加入方向规则 教我设置

安全组入方向规则为白名单（允许），放通入方向网络流量。

安全组 default

如您要添加多条规则，建议单击导入规则以进行批量导入。

优先级 ?	策略	协议端口 ?	类型	源地址 ?	描述	操作
1-100	允许	全部放通	IPv4	IP地址 0.0.0.0/0		复制 删除

+ 增加1条规则

确定 取消

1.3 购买云服务器 ECS 并登录

步骤 1 点击左侧目录栏，选择弹性云服务器 ECS



步骤 2 选择购买弹性云服务器



步骤 3 填写如下基础配置信息，然后点击“下一步”。

- 计费模式：按需计费

- 区域：华北-北京四
- 可用区：随机分配
- CPU 架构：鲲鹏计算
- 规格：鲲鹏通用计算增强型 | kc1.2xlarge.4 | 8vCPUs | 32GB
- 镜像：公共镜像 openEuler 20.03 64bit with ARM(40GB)
- 系统盘：通用型 SSD | 40G

计费模式: 包年/包月 | **按需计费** | 竞价计费

区域: **华北-北京四** | 推荐区域: 西南-贵阳一 (0) | 华北-北京四 (4) | 华南-广州 (0) | 华东-上海一 (0) | 亚太-香港 (0)

可用区: **随机分配** | 可用区1 | 可用区2 | 可用区3 | 可用区7

CPU架构: x86计算 | **鲲鹏计算**

规格: 最新系列 | vCPUs: **8vCPUs** | 内存: **32GiB** | 规格名称:

规格名称	vCPUs 内存	CPU	基准 / 最大带宽	内网收发包	规格参考价
kc1.2xlarge.4	8vCPUs 32GiB	Huawei Kunpeng 920 2.6GHz	3 / 7 Gbit/s	800,000	¥1.63/小时

镜像: **公共镜像** | 私有镜像 | 共享镜像 | 市场镜像

openEuler | **openEuler 20.03 64bit with ARM(40GB)**

系统盘: **通用型SSD** | GB | IOPS上限1,820, IOPS突发上限8,000

+ 增加一块数据盘 您还可以挂载 23 块磁盘 (云硬盘)

Linux实例添加的数据盘可使用脚本向导式初始化。如何操作?

步骤 4 填写如下网络配置信息，然后点击“下一步”。

- 网络：选择已创建的网络和子网，如 myvpc 和 subnet-myvpc
- 安全组： default
- 弹性公网 IP：现在购买
- 规格：全动态 BGP
- 计费方式：按流量计费
- 带宽：100 Mbit/s

1 基础配置 — 2 网络配置 — 3 高级配置 — 4 确认配置

网络 自动分配IP地址 可用私有IP数量250个

如需创建新的虚拟私有云，您可前往控制台创建。

扩展网卡 [增加一块网卡](#) 您还可以增加 2 块网卡

安全组 [新建安全组](#)

安全组类似防火墙功能，是一个逻辑上的分组，用于设置网络访问控制。
请确保所选安全组已放通22端口（Linux SSH登录），3389端口（Windows远程登录）和ICMP协议（Ping）。 [配置安全组规则](#)

[展开安全组规则](#)

弹性公网IP ☒ 现在购买 ☐ 使用已有 ☐ 暂不购买

线路 ☒ 全动态BGP ☐ 静态BGP

不低于99.95%可用性保障

公网带宽 ☒ 按带宽计费 ☐ 按流量计费 ☐ 加入共享带宽

流量较大或较稳定的场景 流量小或流量波动较大场景 多业务流量错峰分布场景

指定带宽上限，按实际使用的出公网流量计费，与使用时间无关。

带宽大小 自定义 带宽范围：1-300 Mbit/s

☒ 免费开启DDoS基础防护

步骤5 填写如下高级配置信息，然后点击“下一步”。

- 云服务器名称：opengauss01（服务器名称建议不要使用下划线，可自定义）
- 登录凭证：密码
- 设置密码（自行设置登录密码）
- 云备份：暂不购买

云服务器名称 ☐ 允许重名

购买多台云服务器时，支持自动增加数字后缀命名或者自定义规则命名。

登录凭证 ☒ 密码 ☐ 密钥对 ☐ 创建后设置

用户名 root

密码 请牢记密码，如忘记密码可登录ECS控制台重置密码。

确认密码

云备份 使用云备份服务，需购买备份存储库，存储库是存放服务器产生的备份副本的容器。

☐ 现在购买 ☐ 使用已有 ☒ 暂不购买

步骤6 在确认配置界面，勾选“我已经阅读……”后，点击“立即购买”。

弹性云服务器 自定义购买 快速购买 放心购 灵活调整

① 基础配置 ② 网络配置 ③ 高级配置 ④ 确认配置

配置

基础配置

计费模式	按需计费	区域	北京四	可用区	可用区2
规格	鲲鹏通用计算增强型 kc1.2xlarge...	镜像	openEuler 20.03 64bit with ARM	主机安全	基础版
系统盘	通用型SSD, 40 GiB				

网络配置

虚拟私有云	default_vpc(192.168.0.0/16)	安全组	default	主网卡	default_subnet(192.168.0.0/24)
弹性公网IP	全动态BGP 计费方式: 按流量计...				

高级配置

云服务器名称	opengauss01	登录凭证	密码	云服务器组	--
--------	-------------	------	----	-------	----

购买数量: 1 您最多可以创建200台云服务器。申请更多云服务器配额请单击[申请扩大配额](#)。

协议: ☒ 我已经阅读并同意 《[镜像免责声明](#)》

配置费用: /小时 + 弹性公网IP流量费用: /GB

参考价格, 具体扣费请以账单为准。 [了解计费详情](#)

上一步 立即购买

步骤7 购买完成后，在云服务器控制台可以看到正常运行中的 ECS，记录弹性 IP 地址。

名称/ID	监控	可用区	状态	规格/镜像	IP地址	计费模式	标签	操作
opengauss01 c80a694a-7ed9-459f-f...		可用区2	运行中	4vCPUs 8GB kc1.xlarge.2 openEuler 20.03 64bit with ARM	121.36... (弹性公网) 1... 192.16... (私有)	按需计费		远程登录 更多

步骤8 打开 putty，输入弹性 IP 地址后（可在弹性云服务器页面查看并复制，请选择弹性公网 IP 地址并进行复制），点击 open。

Putty Configuration

Category:

- Session
- Logging
- Terminal
 - Keyboard
 - Bell
 - Features
- Window
 - Appearance
 - Behaviour
 - Translation
 - Selection
 - Colours
- Connection
 - Data
 - Proxy
 - Telnet
 - Rlogin
 - SSH
 - Serial

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address)	Port
121.36...	22

Connection type:

☐ Raw ☐ Telnet ☐ Rlogin ☒ SSH ☐ Serial

Load, save or delete a stored session

Saved Sessions

Default Settings

Load Save Delete

Close window on exit:

☐ Always ☐ Never ☒ Only on clean exit

About Help Open Cancel


步骤9 用 root 用户名，使用之前设置的密码登录 ECS。

2 openGauss 数据库安装及基本操作

2.1 实验任务及步骤

2.1.1 编译前准备

步骤 1 用 root 用户名，使用之前设置的密码登录 ECS。



```
121.36.11.84 - PuTTY
Authorized users only. All activities may be monitored and reported.
root@121.36.11.84's password:

Welcome to Huawei Cloud Service

Last login: [redacted]

Welcome to 4.19.90-2003.4.0.0036.oel.aarch64

System information as of time: Fri [redacted]

System load:      0.03
Processes:        129
Memory used:      4.9%
Swap used:        0.0%
Usage On:         13%
IP address:       [redacted]
Users online:     1

[root@opengauss01 ~]#
```

步骤 2 创建 openGauss 数据库的安装用户 omm 及其属组 dbgrp。

```
[root@opengauss01 ~]# groupadd -g 1000 dbgrp
[root@opengauss01 ~]# useradd -g dbgrp -u 1000 -d /home/omm omm
```

步骤 3 修改 omm 用户密码。

```
[root@opengauss01 ~]# passwd omm
```

输入修改的 omm 用户密码，建议设置成复杂密码。

步骤 4 创建 openGauss 源码存放及 openGauss 安装路径。

```
[root@opengauss01 ~]# mkdir -p /opt/software/openGauss/data
```

步骤 5 下载第三方编译库。

社区针对 centos_7.6_x86_64、openEuler20.03 LTS_arm、openEuler20.03 LTS_x86_64 三种架构及操作系统已经提供了编译好的二进制，对于这三种系统架构，可以直接使用社区提供的编译好的文件 openGauss-third_party_binarylibs.tar.gz。

```
[root@opengauss01 ~]# cd /opt/software
[root@opengauss01 software]# wget https://opengauss.obs.cn-south-1.myhuaweicloud.com/1.1.0/openGauss-third_party_binarylibs.tar.gz
```

下载时间比较长大概 10 分钟，请耐心等待。

步骤 6 解压下载好的第三方编译库，并重命名为 binarylibs。

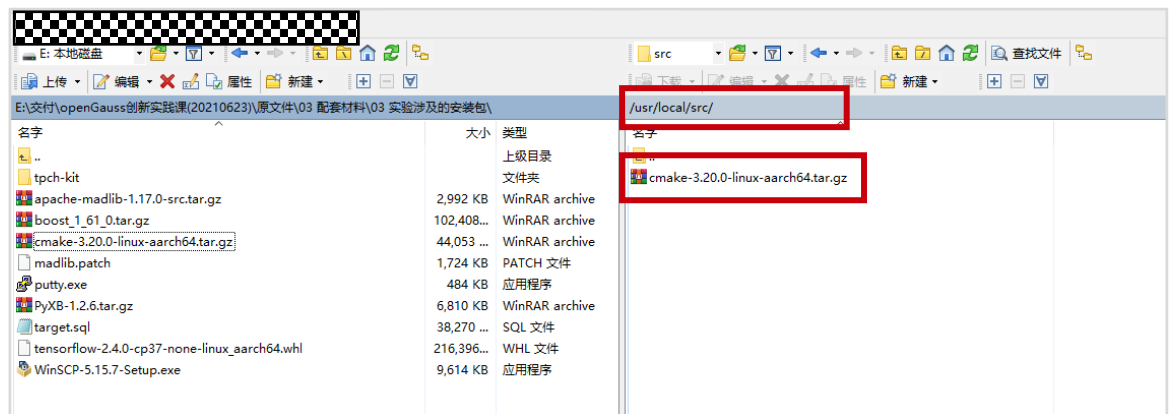
```
[root@opengauss01 software]# tar -zxvf openGauss-third_party_binarylibs.tar.gz
[root@opengauss01 software]# mv openGauss-third_party_binarylibs/ binarylibs/
```

步骤 7 下载 openGauss 源码。

```
[root@opengauss01 software]# git clone -b master https://gitee.com/opengauss/openGauss-server.git
[root@opengauss01 software]# cd openGauss-server
[root@opengauss01 openGauss-server]# git checkout f40ff5a0e9dcde6178ac116b391c662f3a516bbb
```

步骤 8 上传 cmake 包。

将 cmake-3.20.0-linux-aarch64.tar.gz 包，上传至服务器/usr/local/src/下（可使用 winscp 工具），如下图：



然后进行解压，具体如下：

```
[root@opengauss01 software]# cd /usr/local/src/
[root@opengauss01 src]# tar -zxvf cmake-3.20.0-linux-aarch64.tar.gz
[root@opengauss01 src]# chmod 755 /usr/local/src/cmake-3.20.0-linux-aarch64/
```

步骤 9 使用 yum 安装依赖包。

```
[root@opengauss01 src]# yum install -y libaio-devel ncurses-devel pam-devel libffi-devel
libtool libtool-devel libtool-ltdl openssl-devel bison golang flex dkms-2.6.1-5.oe1.noarch
python3-devel patch
```

步骤 10 替换 python 版本。

将 python3 的链接设置为 python 的。

```
[root@opengausso1 src]# cd /usr/bin
[root@opengausso1 bin]# mv python python.bak
[root@opengausso1 bin]# ln -s python3 /usr/bin/python
```

步骤 11 检查 python 的版本。

```
[root@opengausso1 bin]# python -V
```

返回值为: Python 3.7.4

步骤 12 修改/opt/software 路径的用户所属组及权限。

```
[root@opengausso1 bin]# chown omm:dbgrp -R /opt/software
[root@opengausso1 bin]# chmod 755 -R /opt/software
```

2.1.2 数据库安装编译

步骤 1 切换 omm 用户。

```
[root@opengausso1 bin]# su - omm
```

步骤 2 设定 omm 用户的环境变量。

```
[omm@opengausso1 ~]$ vi ~/.bash_profile
```

步骤 3 添加环境变量。

将以下环境变量内容，红色加粗部分根据实际安装环境进行设置，并写入（点击 i 变为 INSERT 模式）.bash_profile 文件中。完成写入后点击 ESC 退出 INSERT 模式，并输入:wq 保存退出

```
# GaussDB Code-
export GAUSSHOME=/opt/software/openGauss #the path for openGauss
export PGDATA=$GAUSSHOME/data #the path for data
export BINARY_PATH=/opt/software/binarylibs #the path for binarylibs
# GCC 7.3 compiler path
export CMAKEROOT=/usr/local/src/cmake-3.20.0-linux-aarch64/ #the path for cmake
export GCC_PATH=$BINARY_PATH/buildtools/openeuler_aarch64/gcc7.3
export CC=$GCC_PATH/gcc/bin/gcc
export CXX=$GCC_PATH/gcc/bin/g++
export
LD_LIBRARY_PATH=$GCC_PATH/gcc/lib64:$GCC_PATH/lib:$GCC_PATH/mpc/lib:$GCC_PATH/
H/mpfr/lib:$GCC_PATH/gmp/lib:$LD_LIBRARY_PATH
export PATH=$GCC_PATH/bin:$CMAKEROOT/bin:$PATH
# Gauss DB install path
export PATH=$GAUSSHOME/bin:$PATH
export LD_LIBRARY_PATH=$GAUSSHOME/lib:$LD_LIBRARY_PATH
export GAUSS_WARNING_TYPE=2
```

```
export GS_CLUSTER_NAME=single
```

步骤 4 使环境变量生效。

```
[omm@opengausso1 ~]$ source ~/.bash_profile
```

步骤 5 进入 openGauss 源码下，生成配置文件。

```
[omm@opengausso1 ~]$ cd /opt/software/openGauss-server/  
[omm@opengausso1 openGauss-server]$ ./configure --gcc-version=7.3.0 --  
prefix=$GAUSSHOME CFLAGS="-Oo" --enable-debug --without-readline --without-zlib --  
enable-cassert --enable-thread-safety CC=g++ --3rd=$BINARY_PATH --with-python
```

步骤 6 执行 make 命令进行编译。

```
[omm@opengausso1 openGauss-server]$ make -j8
```

当返回 All of openGauss successfully made. Ready to install.时，表明编译成功。

```
make[2]: Leaving directory '/opt/software/openGauss-server/src/test/regress'  
make[1]: Leaving directory '/opt/software/openGauss-server/src'  
All of openGauss successfully made. Ready to install.
```

步骤 7 执行 make install 安装。

```
[omm@opengausso1 openGauss-server]$ make install
```

当返回 openGauss installation complete.时，表明安装完成。

```
make[1]: Leaving directory '/opt/software/openGauss-server/contrib/gspreadistribute'  
openGauss installation complete.
```

步骤 8 初始化数据库。

在初始化数据库时，需要设置数据库密码，并且要使用复杂密码，如下命令。

```
gs_initdb -D $PGDATA --nodename=hostname --locale="en_US.UTF-8" -Atrust -w  
{password}
```

实际使用中，将{password}部分进行替换。

例如（此处只是作为举例，建议设置为复杂密码，注意密码要用英文的单引号括起来）：

```
[omm@opengausso1 openGauss-server]$ gs_initdb -D $PGDATA --nodename=opengausso1 --  
locale="en_US.UTF-8" -Atrust -w 'Huawei#!13'
```

显示结果如下，表示初始化成功：

```
freezing database postgres ... The core dump path is an invalid directory  
ok  
  
Success. You can now start the database server of single node using:  
  
    gaussdb -D /opt/software/openGauss/data --single_node  
or  
    gs_ctl start -D /opt/software/openGauss/data -Z single_node -l logfile
```

步骤 9 启动数据库。

```
[omm@opengauss01 openGauss-server]$ gs_ctl start -D /opt/software/openGauss/data -Z single_node -l logfile
```

查看数据库是否启动成功。

```
[omm@opengausso1 openGauss-server]$ ps -ef|grep omm
```

显示结果如下，表示启动成功：

```
[omm@opengauss01 openGauss-server]$ ps -ef|grep omm
root      6458      2608    0 15:57 pts/0    00:00:00 su - omm
omm       6459      6458    0 15:57 pts/0    00:00:00 bash
omm       260516    1      4 16:14 pts/0    00:00:01 /opt/software/openGauss/bin/gaussdb -D /opt/software/openGauss/data
omm       260562    6459    0 16:14 pts/0    00:00:00 ps -ef
omm       260563    6459    0 16:14 pts/0    00:00:00 grep --color=auto omm
```

步骤 10 数据库登录。

```
[omm@opengausso1 openGauss-server]$ gsql -d postgres -p 5432 -r
```

步骤 11 修改 omm 账号密码（可选步骤）。

```
postgres=# ALTER USER omm identified by 'Huawei@13' replace 'Huawei#!13';
```

说明: ALTER USER omm identified by '新密码' replace '原密码';

步骤 12 查询数据库版本。

```
postgres=# select version();
version
-----
PostgreSQL 9.2.4 (GaussDB Kernel V500R001C20 build 4c77cobb) compiled at 2021-03-30
17:12:43 commit o last mr   debug on aarch64-unknown-linux-gnu, compiled by g++ (GCC) 7.3.0,
64-bit
(1 row)
```

步骤 13 版本截图说明完成 openGauss 数据库编译安装完成 (截图 1)

步骤 14 退出数据库。

```
postgres=# \q
```

步骤 15 对数据库状态进行验证，截图说明此步骤完成（截图 2）

```
[omm@opengauss01 openGauss-server]$ gs_ctl status
```

```
lomm@opengauss01 ~|$ gs_ctl status
[7428][[gs_ctl]: gs_ctl status,datadir is /opt/software/openGauss/data
gs_ctl: server is running (PID: 7228)
/opt/software/openGauss/bin/gaussdb "-D" "/opt/software/openGauss/data"
```

步骤 16 其次，对数据库进程进行截图验证，需包含数据库服务器的主机名。（截图 3）

```
[omm@opengausso1 openGauss-server]$ ps -ef|grep omm
```

```
[omm@opengauss01 ~]$ ps -ef|grep omm
root      5655      5543      0 pts/0    00:00:00 su - omm
omm       5656      5655      0 pts/0    00:00:00 -bash
root      6676      6236      0 pts/2    00:00:00 su - omm
omm       6677      6676      0 pts/2    00:00:00 -bash
omm       7228          1      49 pts/2    00:04:13 /opt/software/openGauss/bin/gaussdb -D /opt/software/openGauss/data
omm       7411      5656      0 pts/0    00:00:00 ps -ef
omm       7412      5656      0 pts/0    00:00:00 grep --color=auto omm
```

3 openGauss 的原生 DB4AI 引擎

3.1 实验介绍

3.1.1 关于本实验

本实验在安装完成的 openGauss 数据库上，使用 DB4AI 的功能，利用 openGauss 的 DB4AI 性能进行房价的预测。

3.1.2 实验目的

- 掌握 openGauss 数据库的 DB4AI 的功能。

3.2 实验任务及步骤

3.2.1 利用 DB4AI 原生 AI 引擎训练并预测模型

步骤 1 在数据库中创建一张表，用于数据的训练与预测。

在 OMM 用户环境下，登录数据库。

```
[omm@opengauss01 data]$ gsql -d postgres -p 5432 -r
```

然后用以下语句来创建表。

```
postgres=# DROP TABLE IF EXISTS houses;
postgres=# CREATE TABLE houses (id INT, tax INT, bedroom INT, bath FLOAT, price INT,
size INT, lot INT);
INSERT INTO houses VALUES
(1, 590, 2, 1, 50000, 770, 22100),
(2, 1050, 3, 2, 85000, 1410, 12000),
(3, 20, 3, 1, 22500, 1060, 3500),
(4, 870, 2, 2, 90000, 1300, 17500),
(5, 1320, 3, 2, 133000, 1500, 30000),
(6, 1350, 2, 1, 90500, 820, 25700),
(7, 2790, 3, 2.5, 260000, 2130, 25000),
(8, 680, 2, 1, 142500, 1170, 22000),
(9, 1840, 3, 2, 160000, 1500, 19000),
(10, 3680, 4, 2, 240000, 2790, 20000),
(11, 1660, 3, 1, 87000, 1030, 17500),
```

```
(12,1620,      3,      2,118600,1250,20000),
(13,3100,      3,      2,140000,1760,38000),
(14,2070,      2,      3,148000,1550,14000),
(15, 650,      3, 1.5, 65000,1450,12000);
```

步骤2 通过 \d 命令观察新创建的表 house 结构。

```
postgres=# \d houses
```

可以得到表结构信息：

Table "public.houses"		
Column	Type	Modifiers
id	integer	
tax	integer	
bedroom	integer	
bath	double precision	
price	integer	
size	integer	
lot	integer	

步骤3 通过 CREATE MODEL 语句，基于 SVM 算法创建一个二分类模型。

```
postgres=# CREATE MODEL house_binary_classifier USING svm_classification FEATURES
tax, bath, size TARGET price < 100000 FROM houses;
```

执行 CREATE MODEL 语句后，会开启模型的训练，并输出训练过程中使用的超参数信息：

```
postgres=# CREATE MODEL house_binary_classifier USING svm_classification FEATURES tax, ba
h, size TARGET price < 100000 FROM houses;
NOTICE: Hyperparameter batch_size takes value DEFAULT (1000)
NOTICE: Hyperparameter decay takes value DEFAULT (0.950000)
NOTICE: Hyperparameter lambda takes value DEFAULT (0.010000)
NOTICE: Hyperparameter learning_rate takes value DEFAULT (0.800000)
NOTICE: Hyperparameter max_iterations takes value DEFAULT (100)
NOTICE: Hyperparameter max_seconds takes value DEFAULT (0)
NOTICE: Hyperparameter optimizer takes value DEFAULT (gd)
NOTICE: Hyperparameter tolerance takes value DEFAULT (0.000500)
NOTICE: Hyperparameter seed takes value DEFAULT (0)
NOTICE: Hyperparameter verbose takes value DEFAULT (FALSE)
NOTICE: GD shuffle cache size 4143
MODEL CREATED. PROCESSED 1
```

步骤4 在 gs_model_warehouse 系统表中查看训练后的模型信息，**将执行结果截图**。（截图4）

```
postgres=# SELECT * FROM gs_model_warehouse WHERE modelname =
'house_binary_classifier';
```

```
openGauss=# select * from gs_model_warehouse where modelname = 'house_binary_classifier';
 modelname | modelowner | createtime | processedtuples | discardedtuples | pre_process_time | exe
sname      | query | hyperparametersvalues | modeldata | hyperparametersoi
e | trainingscoresvalue | modeldescribe |
-----+-----+-----+-----+-----+-----+-----+
house_binary_classifier | 10 | 2021-07-29 10:07:50.133507 | 15 | 0 | 0 | .
y_classifier using svm_classification features tax, bath, size target price < 100000 from houses; | | -.0623524,.000
,max_seconds optimizer tolerance seed verbose | {1000,.95,.01,.8,100,0,gd,.0005,1627524470,false} | {23,701,701,701,23,23,1043
call,loss} | {-.666667,.666667,.625,.714286,7.10655}
(1 row)
```

通过系统表的返回信息可以看到训练过程的预测准确率为 0.666667。

这对于二分类来说准确率并不高，我们后面将尝试通过超参数来提升模型表现。

步骤 5 使用 DROP MODEL 语句删除模型。

```
postgres=# DROP MODEL house_binary_classifier;
```

步骤 6 使用更优的超参数提高 SVM 算法的训练表现。

```
postgres=# CREATE MODEL house_binary_classifier USING svm_classification FEATURES tax, bath, size TARGET price < 100000 FROM houses WITH batch_size=1, learning_rate=0.001;
```

在该步骤中，修改训练时的学习率（learning_rate）为 0.001，批大小（batch_size）设置为 1。

```
postgres=# DROP MODEL house_binary_classifier;DROP MODEL house_binary_classifier;DROP MODEL house_binary_classifier;
ERROR: model "house_binary_classifier" does not exist
ERROR: model "house_binary_classifier" does not exist
postgres=# CREATE MODEL house_binary_classifier USING svm_classification FEATURES tax, bath, size TARGET price < 100000 FROM houses WITH batch_size=1, learning_rate=0.001;
NOTICE: Hyperparameter batch_size takes value 1
NOTICE: Hyperparameter decay takes value DEFAULT (0.950000)
NOTICE: Hyperparameter lambda takes value DEFAULT (0.010000)
NOTICE: Hyperparameter learning_rate takes value 0.001000
NOTICE: Hyperparameter max_iterations takes value DEFAULT (100)
NOTICE: Hyperparameter max_seconds takes value DEFAULT (0)
NOTICE: Hyperparameter optimizer takes value DEFAULT (gd)
NOTICE: Hyperparameter tolerance takes value DEFAULT (0.000500)
NOTICE: Hyperparameter seed takes value DEFAULT (0)
NOTICE: Hyperparameter verbose takes value DEFAULT (FALSE)
NOTICE: GD shuffle cache size 342391
MODEL CREATED. PROCESSED 1
```

步骤 7 观察新模型的信息，**将执行结果截图。**（截图 5）

```
postgres=# SELECT * FROM gs_model_warehouse WHERE modelname = 'house_binary_classifier';
```

在返回信息中，我们可以看到新模型的表现：

```
postgres=# SELECT * FROM gs_model_warehouse WHERE modelname = 'house_binary_classifier';
 modelname | modelowner | createtime | processedtuples | discardedtuples | preprocesstime | exectime | iterations | outputtype | modeltype | hyperparametersnames | modeldata | weight | hyperparametersval |
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
 house_binary_classifier | 10 | 2021-08-19 15:43:57.828218 | 15 | 0 | 0 | .011857 | 100 | 16 | svm_classification | CREATE MODEL house_binary_classifier USING svm_classification FEATURES tax, bath, size TARGET price < 100000 FROM houses WITH batch_size=1, learning_rate=0.001; | {(-.00251012,5.05816e-05,.00205143,.000486893)} | {batch_size,decay,lambda,learning_rate,max_iterations,max_seconds,optimizer,tolerance,seed,verbose} | {1,.95,.01,.001,100,0,gd,.0005,162966733,0.001,0.001,0.001,0.001,23,16} | {categories} | {false,true} | {accuracy,f1,precision,recall,loss} | {(.733333,.714286,.714286,.714286,9.48355)} |
```

整体准确率要相对默认超参数有所提升。

步骤 8 使用 PREDICT BY 语句预测样本数据。

```
postgres=# SELECT tax, bath, size, price, price < 100000 AS price_actual, PREDICT BY house_binary_classifier (FEATURES tax, bath, size) AS price_pred FROM houses;
```

返回结果为：

tax	bath	size	price	price_actual	price_pred
590	1	770	50000	t	t
1050	2	1410	85000	t	t
20	1	1060	22500	t	t
870	2	1300	90000	t	t
1320	2	1500	133000	f	f
1350	1	820	90500	t	f
2790	2.5	2130	260000	f	f
680	1	1170	142500	f	t
1840	2	1500	160000	f	f
3680	2	2790	240000	f	f
1660	1	1030	87000	t	f
1620	2	1250	118600	f	f
3100	2	1760	140000	f	f
2070	3	1550	148000	f	f
650	1.5	1450	65000	t	t

(15 rows)

Price_pred 列的输出结果即为预测结果，price_actual 列的结果为真实值。

步骤9 通过 CREATE MODEL 语句创建一个逻辑回归（logistic regression）模型。

```
postgres=# CREATE MODEL house_logistic_classifier USING logistic_regression FEATURES
tax, bath, size target price < 100000 FROM houses WITH learning_rate=0.001;
```

步骤10 利用训练好的逻辑回归模型预测数据，并与 SVM 算法进行比较，**将执行结果截图。**（截图6）

```
postgres=# SELECT tax, bath, size, price, price < 100000 AS price_actual, PREDICT BY
house_binary_classifier (FEATURES tax, bath, size) AS price_svm_pred, PREDICT BY
house_logistic_classifier (FEATURES tax, bath, size) AS price_logistic_pred FROM houses;
```

输出结果如下：

```
postgres=# SELECT tax, bath, size, price, price < 100000 AS price_actual, PREDICT BY house_binary_classifier (FEATURES tax, bath, size)
AS price_svm_pred, PREDICT BY house_logistic_classifier (FEATURES tax, bath, size) AS price_logistic_pred FROM houses;
```

tax	bath	size	price	price_actual	price_svm_pred	price_logistic_pred
590	1	770	50000	t	t	t
1050	2	1410	85000	t	t	t
20	1	1060	22500	t	t	t
870	2	1300	90000	t	t	t
1320	2	1500	133000	f	f	t
1350	1	820	90500	t	f	f
2790	2.5	2130	260000	f	f	f
680	1	1170	142500	f	t	t
1840	2	1500	160000	f	f	f
3680	2	2790	240000	f	f	f
1660	1	1030	87000	t	f	f
1620	2	1250	118600	f	f	f
3100	2	1760	140000	f	f	f
2070	3	1550	148000	f	f	f
650	1.5	1450	65000	t	t	t

(15 rows)

4 资源释放

4.1 删除弹性云服务器及相关资源

完成实验后请务必删除华为云上的收费资源，以免造成不必要的收费。找到创建的弹性云服务器 ECS，按照如下步骤进行删除。

步骤 1 打开云服务器控制台，在需要删除的云服务器后面选择“更多>删除”。



步骤 2 在弹出对话框中勾选“释放云服务器绑定的弹性公网 IP 地址”和“删除云服务器挂载的数据盘”，然后点击“是”。



步骤 3 查看到列表中已没有资源时，表示弹性云服务器已删除。

弹性云服务器 ②

最新动态

帮助引导

购买弹性云服务器

开机

关机

重置密码

更多

所有运行状态

名称

标签搜索

<input type="checkbox"/> 名称/ID	可用区	状态	规格/镜像	IP地址	计费模式	操作
<div><div></div><div>暂无表格数据</div></div>						