鲲鹏 BoostKit 数据库使能套件

OceanBase 部署&调优指南

文档版本 02

发布日期 2024-09-30

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。

安全声明

漏洞处理流程

华为公司对产品漏洞管理的规定以“漏洞处理流程”为准，该流程的详细内容请参见如下网址：
https://www.huawei.com/cn/psirt/vul-response-process
如企业客户须获取漏洞信息，请参见如下网址：
https://securitybulletin.huawei.com/enterprise/cn/security-advisory

目 录

1 OceanBase 部署指南

1.1 介绍

本文主要介绍如何在使用openEuler 20.03操作系统的鲲鹏服务器上部署社区版OceanBase。

OceanBase数据库是一款完全自研的企业级开源分布式数据库，在普通硬件上实现金融级高可用，首创“三地五中心”城市级故障自动无损容灾新标准，刷新TPC-C标准测试，单集群规模超过1500节点，具有云原生、强一致性、企业版高度兼容Oracle/MySQL和社区版高度兼容MySQL等特性。OceanBase区分企业版和社区版，社区版是开源的。

OceanBase数据库包括以下常用组件：

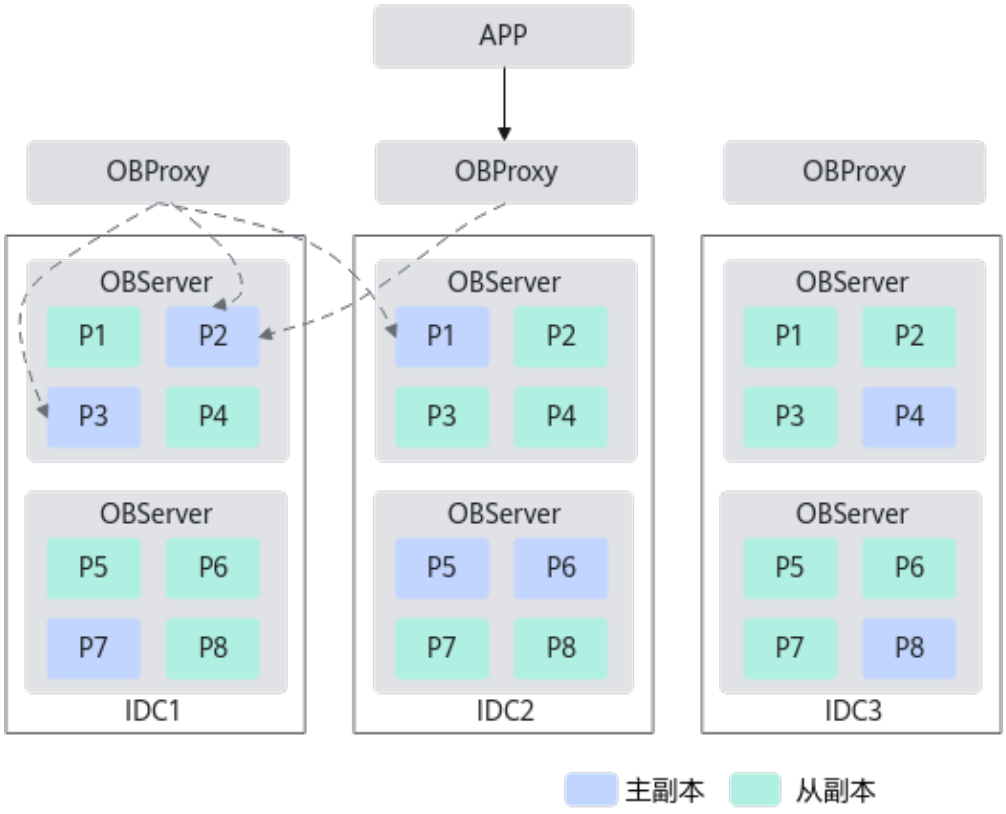
- OBD：OBD（OceanBase Deployer）是OceanBase开源软件的安装部署工具。OBD同时也是包管理器，可以用来管理OceanBase所有的开源软件。OBD的官方

文档请参见**https://www.oceanbase.com/docs/community-obd-cn-10000000001031920**。

- OBServer：OceanBase数据库进程名。
- OBClient：OceanBase自带客户端连接工具，社区版OceanBase兼容MySQL，因此也可以使用MySQL客户端工具连接OceanBase。
- ODP：ODP（OceanBase Database Proxy，简称为OBProxy），是OceanBase数据库专用的服务代理。OBProxy的官方文档请参见**https://www.oceanbase.com/docs/community-odp-cn-10000000001717354**。

  OBProxy中记录了OBServer的数据分布信息，用户的SQL会先发送给ODP节点，由ODP选择合适的Observer进行转发，并将结果返回给用户，如**图1-1**所示，这样的方式比直接转发到没有数据的节点执行效率更高。

图 1-1 OceanBase 数据转发方式



建议的版本

OceanBase 3.1.5或社区版4.2.1。

1.2 环境要求

本文基于鲲鹏服务器和openEuler操作系统提供指导，在正式操作前请确保软硬件均满足要求。

硬件要求

硬件要求如**表1-1**所示。

表 1-1 硬件要求

项目	说明
CPU	鲲鹏920系列处理器
硬盘	• 进行性能测试时，可以使用两块硬盘，即一块盘用于存储日志（clog）信息，另一块盘用于存储数据信息（数据盘可选用性能较好的SATA SSD盘、NVMe SSD盘等）。 • 非性能测试时，可以将日志信息与数据信息存储在同一块硬盘上。 • 具体硬盘数量根据实际需求配置。

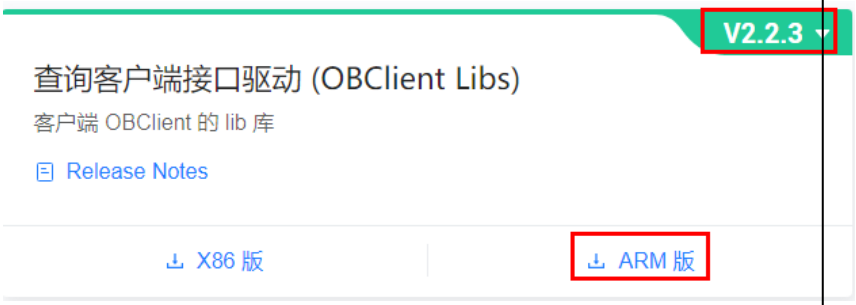
操作系统和软件要求

- 查看环境操作系统的信息：**cat /etc/*-release**
  查看环境处理器相关信息：**lscpu**
  查看环境内核版本：**uname -r**
  查看环境信息：**uname -a**
- 如果需要全新安装操作系统，可选择“Minimal Install”安装方式并勾选Development Tools套件，否则很多软件包需要手动安装。

操作系统要求如表1-2所示。

表 1-2 操作系统要求

项目和版本	下载地址
openEuler 20.03 LTS-SP1	**https://repo.huaweicloud.com/openeuler/openEuler-20.03-LTS-SP1/ISO/aarch64/openEuler-20.03-LTS-SP1-everything-aarch64-dvd.iso**

项目和版本	下载地址
OceanBase 3.1.5相关的RPM包	**https://www.oceanbase.com/softwarecenter** • libobclient-2.2.3-1.el7.aarch64.rpm  • obclient-2.2.3-1.el7.aarch64.rpm  • ob-deploy-2.3.1-2.el7.aarch64.rpm  • obproxy-ce-3.2.3.5-2.el7.aarch64.rpm  • oceanbase-ce-3.1.5-100020022023091114.el7.aarch64.rpm

项目和版本	下载地址
	 • oceanbase-ce-libs-3.1.5-100020022023091114.el7.aarch64.rpm
OceanBase 4.2.1社区版一键安装包	https://www.oceanbase.com/softwarecenter 

1.3 配置安装环境

1.3.1 关闭防火墙

测试环境下通常需要关闭防火墙以避免部分网络因素影响，请用户根据实际需求执行本章节操作。

步骤1 停止防火墙。

```
systemctl stop firewalld.service
```

步骤2 关闭防火墙。

须知

执行当前步骤的命令关闭防火墙的同时，也取消了开机自启动。

```
systemctl disable firewalld.service
```

步骤3 查看防火墙状态。

```
systemctl status firewalld.service
```

----结束

1.3.2 搭建数据盘

为了满足数据存储需求，需要搭建数据盘，以支持更大的数据存储容量。

步骤1 创建文件系统。以XFS为例，根据实际需求创建文件系统，其中“/dev/nvme0n1”可根据实际磁盘名称进行输入。

```
mkfs.xfs /dev/nvme0n1
```

若磁盘之前已创建文件系统，执行此命令会出现报错，可使用-f参数强制创建文件系统。

```
mkfs.xfs -f /dev/nvme0n1
```

步骤2 创建数据目录。

```
mkdir /data
```

步骤3 挂载磁盘。

```
mount /dev/nvme0n1 /data
```

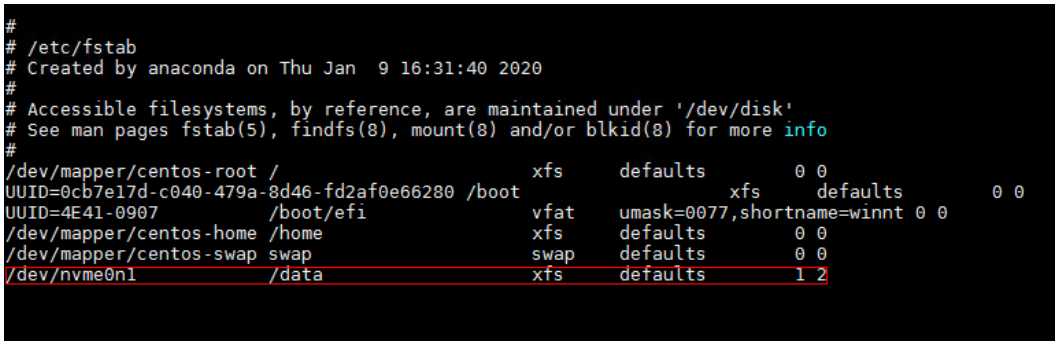
步骤4 打开“/etc/fstab”文件。

```
vi /etc/fstab
```

步骤5 按“i”进入编辑模式，在文件最后添加如下内容。

```
/dev/nvme0n1        /data                   xfs     defaults        1 2
```

添加内容完成后，如下图所示。



步骤6 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。

----结束

1.4 安装 OceanBase

1.4.1 安装 OceanBase 3.1.5

安装OceanBase 3.1.5版本，需要到官网下载RPM包，并进行安装。请根据实际需要，选择安装合适的OceanBase版本。

步骤1 前往OceanBase安装包下载中心下载OceanBase 3.1.5相关的RPM包，并将下载下来的安装包拷贝到服务器上。

下载链接请参见**1.2 环境要求**。

步骤2 创建文件夹oceanbasev3，并将所有RPM包上传到oceanbasev3文件夹中。

```
mkdir oceanbasev3
cd oceanbasev3
```

步骤3 安装OBD以及OBClient。

```
rpm -ivh ob-deploy-2.3.1-2.el7.aarch64.rpm
rpm -ivh obclient-2.2.3-1.el7.aarch64.rpm
```

步骤4 安装完成后，执行如下任意一个命令检测是否安装成功。

```
which obd
```

或者

```
which obclient
```

命令返回OBD的路径则表示安装成功。

```
[ ·]$ which obd
/usr/bin/obd
```

步骤5 禁用OBD的远程仓库。

```
obd mirror disable remote
```

步骤6 将下载的安装包添加至OBD管理的本地镜像中。

```
obd mirror clone *.rpm
```

步骤7 查看OBD管理的本地镜像库中的所有安装包列表。

```
obd mirror list local
```

```
[ ·]$ obd mirror list local
+-------------------------------------------------------------------------------------------------------------------+
|                                                local Package List                                                 |
+------------------+---------+----------------------+---------+------------------------------------------+
| name             | version | release              | arch    | md5                                      |
+------------------+---------+----------------------+---------+------------------------------------------+
| libobclient      | 2.2.3   | 1.el7                | aarch64 | b5dd50e451b691dba6b019e237acfbc904bde435 |
| obclient         | 2.2.3   | 1.el7                | aarch64 | 09d2d4c064ef27461a32360706b6de1f10607b93 |
| ob-deploy        | 2.3.1   | 2.el7                | aarch64 | 0b632b18fd87def0ed13cdbed98935b379ed6875 |
| obproxy-ce       | 3.2.3.5 | 2.el7                | aarch64 | dfeeb814e4061874bc8be6c4e88753b14aad5c6f |
| oceanbase-ce     | 3.1.5   | 100020022023091114.el7 | aarch64 | 80bd2db3a8cd78e2379f0af1d0c3a263109bd88a |
| oceanbase-ce-libs | 3.1.5   | 100020022023091114.el7 | aarch64 | 41b81f2ede47605fe0d63fcf0b524c28bb170211 |
+------------------+---------+----------------------+---------+------------------------------------------+
Trace ID: 127be4b4-4e44-11ef-8344-d4d51b2cba8f
If you want to view detailed obd logs, please run: obd display-trace 127be4b4-4e44-11ef-8344-d4d51b2cba8f
```

----结束

1.4.2 安装 OceanBase 4.2.1

OceanBase提供一键安装包all-in-one package，支持用户通过all-in-one package即可一键获得OceanBase数据库及其关联的组件服务，包括OceanBase数据库软件、OBD、OBProxy和OBClient。请根据实际需要，选择安装合适的OceanBase版本。

步骤1 在OceanBase一键安装包下载中心直接下载OceanBase 4.2.1社区版一键安装包，并将下载下来的安装包拷贝到服务器上。

下载链接请参见**1.2 环境要求**。

步骤2 在安装包所在目录执行如下命令解压安装包。

```
tar -xzf oceanbase-all-in-one-*.tar.gz
```

步骤3 进入bin目录，执行安装脚本。

```
cd oceanbase-all-in-one/bin/
./install.sh
```

步骤4 配置环境变量。

```
source ~/.oceanbase-all-in-one/bin/env.sh
```

步骤5 安装完成后，执行如下任意一个命令检测是否安装成功。

```
which obd
```

或者

```
which obclient
```

命令返回oceanbase-all-in-one下的OBD和OBClient路径则表示安装成功。

```
[ @localhost example]$ which obd
/home/ /.oceanbase-all-in-one/obd/usr/bin/obd
[ @localhost example]$ which obclient
/home/ /.oceanbase-all-in-one/obclient/u01/obclient/bin/obclient
```

----结束

1.5 部署和启动 OceanBase

通过一键安装包安装OceanBase软件后，请按照本节内容通过集群配置文件部署并启动OceanBase集群。启动OceanBase集群之前，需要确保已经安装并配置NTP。

步骤1 创建并修改yaml配置文件。

1. 创建yaml配置文件。distributed-example.yaml为集群配置文件名称，请根据实际情况修改。
   ```
   vi distributed-example.yaml
   ```
2. 按“i”进入编辑模式，参见“/oceanbase-all-in-one/obd/usr/obd/example”文件夹下的配置文件模版，根据需要修改yaml配置文件信息。关于yaml配置文件示例的详细信息，请参见**1.6 yaml配置文件示例**。
3. 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。

步骤2 使用yaml配置文件部署OceanBase。

```
cd /oceanbase-all-in-one/obd/usr/obd/example
obd cluster deploy obcluster -c distributed-example.yaml
```

说明

以下参数请根据实际情况修改：

- obcluster为集群名称。
- distributed-example.yaml为集群配置文件名称。

步骤3 安装并配置NTP服务。

1. 在所有集群和NTP客户端节点安装NTP。
   ```
   yum -y install ntp ntpdate
   ```

```
[root@ceph1 ~]# yum -y install ntp ntpdate
Loaded plugins: fastestmirror, langpacks
Determining fastest mirrors
 * base: centosu7.centos.org
 * epel: epel.dionipe.id
 * extras: centosg8.centos.org
 * updates: centosg9.centos.org
```

2. 在所有集群和NTP客户端节点备份旧配置。
```
cd /etc && mv ntp.conf ntp.conf.bak
```
3. 选择一台服务器为NTP服务端节点，在该服务器上新建并配置NTP文件。在本例中以192.168.0.216服务器为NTP服务端节点。
   a. 创建文件。
   ```
   vi /etc/ntp.conf
   ```
   b. 按“i”进入编辑模式，并在文件中新增如下内容作为NTP服务端：
   ```
   restrict 127.0.0.1
   restrict ::1
   restrict 192.168.0.216 mask 255.255.255.0
   server 127.127.1.0
   fudge 127.127.1.0 stratum 8
   ```
   说明

   其中，restrict 192.168.0.216 mask 255.255.255.0是NTP服务器的网段与掩码，请根据实际情况修改。
   c. 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。
4. 在其他所有NTP客户端节点新建NTP文件。
   a. 打开文件。
   ```
   vi /etc/ntp.conf
   ```
   b. 按“i”进入编辑模式，并在文件中新增如下内容作为客户端节点：
   ```
   server 192.168.0.216
   ```
   c. 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。

步骤4 启动和同步NTP服务。

1. 在192.168.0.216 NTP服务端节点启动NTP服务并查看NTP服务状态。
```
systemctl start ntpd
systemctl enable ntpd
systemctl status ntpd
```
```
[root@ceph1 ~]# systemctl start ntpd
[root@ceph1 ~]# systemctl enable ntpd
[root@ceph1 ~]# systemctl status ntpd
● ntpd.service - Network Time Service
   Loaded: loaded (/usr/lib/systemd/system/ntpd.service; enabled; vendor preset: disabled)
   Active: active (running) since Mon 2019-12-02 23:58:13 CST; 44min ago
 Main PID: 363629 (ntpd)
   CGroup: /system.slice/ntpd.service
           └─363629 /usr/sbin/ntpd -u ntp:ntp -g

Dec 02 23:58:13 ceph4 ntpd[363629]: Listen normally on 4 bond0 90.90.1.181 UDP 123
Dec 02 23:58:13 ceph4 ntpd[363629]: Listen normally on 5 bond1 90.90.2.181 UDP 123
Dec 02 23:58:13 ceph4 ntpd[363629]: Listen normally on 6 bond0 fe80::72c7:f2ff:fecb:4d04 UDP 123
Dec 02 23:58:13 ceph4 ntpd[363629]: Listen normally on 7 lo ::1 UDP 123
Dec 02 23:58:13 ceph4 ntpd[363629]: Listen normally on 8 bond1 fe80::72c7:f2ff:fecb:4d06 UDP 123
Dec 02 23:58:13 ceph4 ntpd[363629]: Listen normally on 9 enp189s0f0 fe80::3ee1:2c18:14ed:cb74 UDP 123
Dec 02 23:58:13 ceph4 ntpd[363629]: Listening on routing socket on fd #26 for interface updates
Dec 02 23:58:13 ceph4 ntpd[363629]: 0.0.0.0 c016 06 restart
Dec 02 23:58:13 ceph4 ntpd[363629]: 0.0.0.0 c012 02 freq_set kernel 0.000 PPM
Dec 02 23:58:13 ceph4 ntpd[363629]: 0.0.0.0 c011 01 freq_not_set
```
2. 在其他所有NTP客户端节点强制同步192.168.0.216 NTP服务端节点的时间。
```
ntpdate 192.168.0.216
```
3. 在除了192.168.0.216 NTP服务端节点的所有节点写入硬件时钟，以避免重启服务器后时间同步失效。
```
hwclock -w
```
4. 在除了192.168.0.216 NTP服务端节点的所有节点安装并启动crontab工具。
```
yum install -y crontabs
chkconfig crond on
systemctl start crond
crontab -e
```
在执行**crontab -e**命令后将打开当前用户的crontab文件，用于编辑定时任务。

5. 在打开的crontab文件中添加以下内容，设置每隔10min自动与192.168.0.216 NTP服务端节点同步时间。
```
*/10 * * * * /usr/sbin/ntpdate 192.168.0.216
```
6. 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。

步骤5 启动OceanBase集群。

```
obd cluster start obcluster
```

说明

- obcluster为集群名称，请根据实际情况修改。
- 首次部署集群时，当前步骤中的操作命令会进行集群初始化，包括创建系统表、系统租户等。非首次部署时，当前步骤中的操作命令表示启动OceanBase集群。

步骤6 查看observer进程，确认OceanBase集群已启动。

```
ps -aux | grep observer
```

预期结果：

```
[      @localhost ~]$ ps -aux | grep observer
yujiabin 3435475 1893  6.6 37449664 35670720 ?   Ssl  09:24 977:26 /data/server1/0124/bin/observer -r 192.168.0.216:2882:2881;192.168.0.216:2884:2883;192.168.0.214:2882:2881;192.168.0.214:2884:2883;192.168.0.212:2882:2881;192.168.0.212:2884:2883;192.168.0.56:2882:2881 -p 2881 -P 2882 -z zone1 -n obcluster -c 1706253599 -d /data/server1/0124/store -i enp5s0 -o __min_full_resource_pool_memory=2147483648,memory_limit=100G,system_memory=30G,enable_syslog_wf=False,enable_syslog_recycle=True,max_syslog_file_count=4,skip_proxy_sys_private_check=True,enable_strict_kernel_release=False,cpu_count=64,datafile_size=700G,log_disk_size=700G
```

步骤7 使用OBClient连接OceanBase。

```
obclient -h127.0.0.1 -P2883 -uroot -Doceanbase -A
```

说明

以下参数请根据实际情况修改：

- 127.0.0.1和2883为OceanBase服务器的IP地址和端口号。
- root为数据库用户，sysbench_tenant为数据库租户。
- oceanbase为OceanBase数据库的名称。

步骤8 **可选:** 停止集群。

```
obd cluster stop obcluster
```

说明

obcluster为集群名称，请根据实际情况修改。

步骤9 **可选:** 查看集群状态。

```
obd cluster display obcluster
```

说明

obcluster为集群名称，请根据实际情况修改。

步骤10 **可选:** 查看集群列表。

```
obd cluster list
```

----结束

1.6 yaml 配置文件示例

yaml配置文件为部署OceanBase时用以指定OceanBase配置项的文件。

本文档提供如下两个场景下部署OceanBase的yaml配置文件示例，仅供参考。

须知

请确保OBServer与OBProxy之间的延迟较低，以避免在启动集群时出现server之间或者server与Proxy之间连接失败的情况。

- 场景1：四台服务器，其中一台服务器上部署OBProxy，另外三台服务器上分别部署OBServer。部署OBServer的详细信息请参见《 OceanBase部署指南 》的**部署和启动OceanBase**章节。yaml配置文件如下：

```
user:
  username: XXX      # OBServer服务器的用户名，请根据实际情况自定义
  password: XXX      # OBServer服务器的用户密码，请根据实际情况自定义
oceanbase-ce: # OBServer配置信息
  servers:
    - name: server1     # 服务器名称
      ip: 192.168.0.216     # 服务器IP地址
    - name: server2
      ip: 192.168.0.214
    - name: server3
      ip: 192.168.0.212
  global:     # 全局参数
    memory_limit: 300G # OBServer可以占用的最大内存值
    system_memory: 30G # 分配给系统默认租户的内存值
    datafile_size: 300G # 数据文件最大容量
    log_disk_size: 300G # 日志文件最大容量
    enable_syslog_wf: false
    enable_syslog_recycle: true
    max_syslog_file_count: 4
    skip_proxy_sys_private_check: true
    enable_strict_kernel_release: false
    cpu_count: 128  # OBServer可以占用的最多CPU数
  server1: # 服务器列表
    devname: enp5s0 # 网卡名称
    mysql_port: 2881 # OceanBase外部端口
    rpc_port: 2882 # OceanBase内部端口
    home_path: /sata/1108     # OceanBase工作目录，请根据实际情况修改
    data_dir: /sata/1102/data # 数据文件目录，请根据实际情况修改
    redo_dir: /sata/1102/redo # 日志文件目录，请根据实际情况修改
    zone: zone1
  server2:
    devname: enp5s0
    mysql_port: 2881
    rpc_port: 2882
    home_path: /sata/1108
    data_dir: /sata/1102/data
    redo_dir: /sata/1102/redo
    zone: zone2
  server3:
    devname: enp3s0
    mysql_port: 2881
    rpc_port: 2882
    home_path: /sata/1108
    data_dir: /sata/1102/data
    redo_dir: /sata/1102/redo
    zone: zone3
obproxy-ce: # OBProxy配置
  depends:
    - oceanbase-ce
  servers:
    - 192.168.0.210  # Proxy服务IP地址
  global:
    listen_port: 2883 # Proxy对外端口
    prometheus_listen_port: 2884 # Proxy内部端口
    home_path: /home/1108obproxy # Proxy工作目录，请根据实际情况修改
    enable_cluster_checkout: false
    enable_compression_protocol: false
```

```
  skip_proxy_sys_private_check: true
  enable_strict_kernel_release: false
```

- 场景2：四台服务器，其中一台服务器仅部署HAProxy（部署HAProxy的详细操作步骤请参见本文档下文的**部署HAProxy**），另外三台服务器分别部署OBServer和OBProxy。yaml配置文件如下：

```
user:
  username: XXX      # OBServer服务器的用户名
  password: XXX      # OBServer服务器的用户密码
oceanbase-ce: # OBServer配置信息
  servers:
    - name: server1    # 服务器名称
      ip: 192.168.0.216    # 服务器IP地址
    - name: server2
      ip: 192.168.0.214
    - name: server3
      ip: 192.168.0.212
  global:    # 全局参数
    memory_limit: 300G # OBServer可以占用的最大内存值
    system_memory: 30G # 分配给系统默认租户的内存值
    datafile_size: 300G # 数据文件最大容量
    log_disk_size: 300G # 日志文件最大容量
    enable_syslog_wf: false
    enable_syslog_recycle: true
    max_syslog_file_count: 4
    skip_proxy_sys_private_check: true
    enable_strict_kernel_release: false
    cpu_count: 128  # OBServer可以占用的最多CPU数
  server1: # 服务器列表
    devname: enp5s0 # 网卡名称
    mysql_port: 2881 # OceanBase外部端口
    rpc_port: 2882 # OceanBase内部端口
    home_path: /sata/1108    # OceanBase工作目录，请根据实际情况修改
    data_dir: /sata/1102/data # 数据文件目录，请根据实际情况修改
    redo_dir: /sata/1102/redo # 日志文件目录，请根据实际情况修改
    zone: zone1
  server2:
    devname: enp5s0
    mysql_port: 2881
    rpc_port: 2882
    home_path: /sata/1108
    data_dir: /sata/1102/data
    redo_dir: /sata/1102/redo
    zone: zone2
  server3:
    devname: enp3s0
    mysql_port: 2881
    rpc_port: 2882
    home_path: /sata/1108
    data_dir: /sata/1102/data
    redo_dir: /sata/1102/redo
    zone: zone3
obproxy-ce: # OBProxy配置
  depends:
    - oceanbase-ce
  servers:
    - name: server1 # Proxy服务名称
      ip: 192.168.0.216 # Proxy服务IP地址
    - name: server2
      ip: 192.168.0.214
    - name: server3
      ip: 192.168.0.212
  global:
    listen_port: 2883 # Proxy对外端口
    prometheus_listen_port: 2884  # Proxy内部端口
    home_path: /home/1108obproxy # Proxy工作目录，请根据实际情况修改
    enable_cluster_checkout: false
    enable_compression_protocol: false
```



```
skip_proxy_sys_private_check: true
enable_strict_kernel_release: false
```

部署 HAProxy

场景2中涉及部署HAProxy，本节将提供部署HAProxy的详细操作步骤。在本例中，将以在IP地址为192.168.0.210的服务器上配置HAproxy为例进行说明。

1. 安装HAProxy。
   ```
   yum install haproxy
   ```
2. 修改HAProxy配置文件。
   a. 打开文件。
      ```
      vi /etc/haproxy/haproxy.cfg
      ```
   b. 按“i”进入编辑模式，将配置文件内容修改为如下信息：
      ```
      global
          log         127.0.0.1 local0
          chroot      /var/lib/haproxy
          pidfile     /var/run/haproxy.pid
          user        root
          group       root
          daemon
          maxconn     4000
          nbthread    48
      defaults
          log                     global
          retries                 3
          timeout connect         2s
          timeout client          30000s  # 客户端与HAProxy连接后，数据传输完毕，即非活动连接的超时时间。
          timeout server          30000s  # 服务器端非活动连接的超时时间。
      listen obcluster  # 配置database负载均衡
          bind 0.0.0.0:80  # 浮动IP地址和侦听端口。
          mode tcp  # HAProxy要使用第4层的传输层。
          balance leastconn # 连接数最少的服务器优先接收连接。`leastconn`建议用于长会话服务，例如LDAP、SQL、TSE等，而不是短会话协议，如HTTP。该算法是动态的，对于启动慢的服务器，服务器权重会在运行中作调整。
          server  obproxy-1 192.168.0.216:2883 check inter 2000 rise 2 fall 3 # 检测2883端口，检测频率为每2000ms一次。如果2次检测为成功，则认为服务器可用；如果3次检测为失败，则认为服务器不可用。
          server  obproxy-2 192.168.0.214:2883 check inter 2000 rise 2 fall 3
          server  obproxy-3 192.168.0.212:2883 check inter 2000 rise 2 fall 3
      ```
   c. 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。
3. 启动HAProxy服务。
   ```
   service haproxy start
   ```
4. 查看HAProxy服务状态。
   ```
   service haproxy status
   ```
   返回active (running)表示HAProxy服务启动成功。
   ```
   [root@localhost ~]# service haproxy status
   Redirecting to /bin/systemctl status haproxy.service
   • haproxy.service - HAProxy Load Balancer
      Loaded: loaded (/usr/lib/systemd/system/haproxy.service; disabled; vendor preset: disabled)
      Active: active (running) since Mon 2023-11-06 17:27:22 CST; 3 months 1 days ago
    Main PID: 3524627 (haproxy)
       Tasks: 49
      Memory: 37.0M
      CGroup: /system.slice/haproxy.service
              ├─3524627 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid
              └─3524628 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid
   ```


1.7 验证 OceanBase

部署OceanBase完成后，需要验证OceanBase是否已经可用。在验证OceanBase前，需要先创建租户。

OceanBase是支持多租户的数据库，OceanBase数据库租户是一个逻辑概念，是资源分配单位，OceanBase数据库租户间的数据是完全隔离的，每个租户都相当于传统数据库的一个数据库实例。

OceanBase数据库租户分为系统租户和普通租户。本节介绍通过创建普通租户来验证OceanBase。OceanBase数据库默认会自动创建系统租户，系统租户负责一部分OceanBase数据库管理工作，并且能够访问系统元数据表。系统租户自动预留了一定的资源。

以下为创建普通租户以及数据库的详细操作步骤：

步骤1 使用root用户通过OBClient连接OceanBase。

```
obclient -h127.0.0.1 -P2883 -uroot -Doceanbase -A
```

说明

以下参数请根据实际情况修改：

- 127.0.0.1和2883为OceanBase服务器的IP地址和端口号。
- root为数据库用户。
- **oceanbase**为OceanBase数据库的名称。

步骤2 创建资源单位。

- OceanBase 3.1.5：

  ```
  CREATE RESOURCE UNIT sysbench_unit max_cpu = 122, max_memory = '200G', min_memory = '200G', max_iops = 102400,  max_session_num = 30000, max_disk_size = '120G';
  ```

  说明

  以下参数请根据实际情况修改：

  - sysbench_unit为本例中需要创建的资源单位名称。
  - MAX_CPU和MAX_MEMORY表示使用该资源配置的资源单元能够提供的CPU和Memory的上限。
  - MAX_MEMORY的取值范围为[1073741824，+∞)，单位为字节，即最小值为1GB。
  - MAX_IOPS的取值范围为[128，+∞)。
  - MAX_SESSION_NUM的取值范围为[64，+∞)。

- OceanBase 4.2.1：

  ```
  CREATE RESOURCE UNIT sysbench_unit
  MEMORY_SIZE='200g',LOG_DISK_SIZE='120g',MIN_CPU=122,MAX_CPU=122,MAX_IOPS=102400,MIN_IOPS=102400;
  ```

说明

以下参数请根据实际情况修改：

- sysbench_unit为本例中需要创建的资源单位名称。
- MEMORY_SIZE字段用于指定内存容量。最小值为1GB。
- MIN_CPU和MIN_MEMORY表示使用该资源配置的资源单元能够提供的CPU和Memory的下限。
- LOG_DISK_SIZE字段用于指定租户的Unit日志盘的大小。默认值为内存规格值的3倍，最小值为2GB。
- MIN_CPU字段用于指定CPU的最少数量，为可选项。默认等于MAX_CPU，最小值为1C。
- MAX_CPU字段用于指定CPU的最多数量。最小值为1C。
- MAX_IOPS字段用于指定IOPS的最多数量，为可选项。最小值为1024。
- MIN_IOPS字段用于指定IOPS的最少数量，为可选项。最小值为1024。

步骤3 创建资源池。

```
CREATE RESOURCE POOL sysbench_pool unit = 'sysbench_unit', unit_num = 1,
zone_list=('zone1','zone2','zone3');
```

说明

以下参数请根据实际情况修改：

- sysbench_pool为本例中需要创建的资源池名称。
- sysbench_unit为**步骤2**中创建的资源单位名称。
- unit_num字段用于指定要创建的单个Zone下的Unit个数。每个单元会根据当前集群负载，自动在每个Zone中选择一个Server负载，但同一个资源池的多个Unit不能分配到同一个Server，即一个资源池包含的Unit个数不能超过单Zone内Server的个数。指定要创建的单个Zone下的Unit个数。每个单元会根据当前集群负载，自动在每个Zone中选择一个Server负载，但同一个资源池的多个Unit不能分配到同一个Server，即一个资源池包含的Unit个数不能超过单Zone内Server的个数。
- zone_list字段用于指定要创建的资源池所属的Zone。指定要创建的资源池所属的属于哪些Zone。可以通过查询oceanbase.DBA_OB_ZONES视图，获取集群中的Zone信息，操作步骤如下：
  1. 使用root用户登录到集群的系统租户。连接示例如下，连接数据库时请以实际环境为准。
     ```
     obclient -h127.0.0.1 -P2883 -uroot -Doceanbase -A
     ```
  2. 在数据库中查询oceanbase.DBA_OB_ZONES视图，获取集群中的Zone信息。
     ```
     SELECT * FROM oceanbase.DBA_OB_ZONES;
     ```

步骤4 创建普通租户。

```
create tenant sysbench_tenant resource_pool_list=('sysbench_pool'), charset=utf8mb4, replica_num=3,
zone_list('zone1', 'zone2', 'zone3'), primary_zone=RANDOM, locality='F@zone1,F@zone2,F@zone3' set
variables ob_compatibility_mode='mysql', ob_tcp_invited_nodes='%';
```


说明

以下参数请根据实际情况修改：

- sysbench_tenant为本例中需要创建的租户名称。
- sysbench_pool为资源池名称。
- charset字段用于指定字符集。
- replica_num字段用于指定副本数。一般情况下设置为与Zone数相同，且不允许超过Zone数。
- zone_list字段用于指定要创建的资源池所属的Zone。
- primary_zone字段用于指定租户所在的PrimaryZone名称。
- locality字段用于描述副本在Zone间的分布情况。
- ob_compatibility_mode字段用于指定租户的兼容模式（可选MySQL或Oracle模式），而且只能在创建时指定；如果不指定ob_compatibility_mode，默认兼容模式为MySQL。

步骤5 使用root用户登录创建的租户下的系统默认数据库。

```
obclient -h127.0.0.1 -P2883 -uroot@sysbench_tenant -Doceanbase -A
```

说明

以下参数请根据实际情况修改：

- 127.0.0.1和2883为OceanBase服务器的IP地址和端口号。
- root为数据库用户，sysbench_tenant为数据库租户。
- oceanbase为OceanBase数据库的名称。

步骤6 创建sysbench_tenant租户下的数据库。

```
CREATE DATABASE sysbench_test;
```

说明

sysbench_test为本例中需要创建的sysbench_tenant租户下的数据库，请根据实际情况修改。

完成**步骤1**~**步骤6**后，即可使用创建的普通租户以及数据库。

----结束

1.8 故障排除

1.8.1 启动 OceanBase 集群时提示 open files must be not be less than 20000 的解决方法

问题现象描述

启动OceanBase集群时提示：

```
open files must be not be less than 20000(Current value: 1024)
```

```
Check before start observer x
[ERROR] OBD-1007: (127.0.0.1) open files must not be less than 20000 (Current value: 1024)
[WARN] (127.0.0.1) clog and data use the same disk (/)
```

关键过程、根本原因分析

配置文件里面的最大文件数设置过小，导致打开文件数超过最大文件数时被阻止。

结论、解决方案及效果

步骤1 在服务器的命令行界面执行如下命令。

```
cat >> /etc/security/limits.conf << EOF
*       soft    nofile  655350
*       hard    nofile  655350
EOF
```

步骤2 重新登录服务器，使**步骤1**的配置生效。

步骤3 查看open files当前值。

```
ulimit -n
```

命令返回655350，表示已配置成功。

----结束

1.8.2 启动 OceanBase 集群时提示 not enough disk space 的解决方法

问题现象描述

启动OceanBase集群时提示：

```
not enough disk space.
```

```
Load cluster param plugin ok
Check before start observer x
[WARN] (127.0.0.1) clog and data use the same disk (/)
[ERROR] (127.0.0.1) / not enough disk space. (Avail: 37.8G, Need: 54.0G)
```

关键过程、根本原因分析

部署OceanBase集群过程中，在配置yaml文件时，需要确保以下给目录预留足够的空间，如果预留空间不足，就会出现这个报错。

```
datafile_size: 300G # 数据文件最大容量
log_disk_size: 300G # 日志文件最大容量
```

- data_dir配置项所在磁盘空间需要大于datafile_size配置项的值。
- redo_dir配置项所在磁盘空间需要大于log_disk_size配置项的值。

结论、解决方案及效果

修改部署OceanBase时用到的yaml配置文件，将配置文件的data_dir、redo_dir三个配置项指定到空间足够大的磁盘。

步骤1 打开yaml配置文件，例如example.yaml。

```
vi example.yaml
```

步骤2 按“i”进入编辑模式，将配置文件的data_dir和redo_dir三个配置项修改为如下内容：

```
data_dir: /sata/data
redo_dir: /sata/redo
```

说明

以下参数请根据实际情况修改：

- /sata/data为数据文件的存放路径。
- /sata/redo为日志文件的存放路径。

步骤3 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。

----结束

1.8.3 启动 OceanBase 集群时提示 Cluster NTP is out of sync 的解决方法

问题现象描述

启动OceanBase集群时提示：

```
[ERROR] Cluster NTP is out of sync.
```

关键过程、根本原因分析

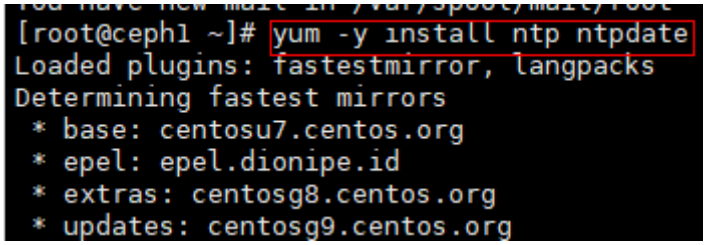
由于OceanBase集群中主机时差超过100ms导致产生该报错，需要通过NTP服务来进行时间同步。

结论、解决方案及效果

步骤1 关闭防火墙。

```
systemctl stop firewalld
systemctl disable firewalld
systemctl status firewalld
```

步骤2 安装并配置NTP服务。

1. 在所有集群和NTP客户端节点安装NTP。
   ```
   yum -y install ntp ntpdate
   ```
   

2. 在所有集群和NTP客户端节点备份旧配置。
   ```
   cd /etc && mv ntp.conf ntp.conf.bak
   ```
3. 选择一台服务器为NTP服务端节点，在该服务器上新建并配置NTP文件。在本例中以192.168.0.216服务器为NTP服务端节点。
   a. 创建文件。
      ```
      vi /etc/ntp.conf
      ```
   b. 按“i”进入编辑模式，并在文件中新增如下内容作为NTP服务端：
      ```
      restrict 127.0.0.1
      restrict ::1
      restrict 192.168.0.216 mask 255.255.255.0
      server 127.127.1.0
      fudge 127.127.1.0 stratum 8
      ```


说明

其中，restrict 192.168.0.216 mask 255.255.255.0是NTP服务器的网段与掩码。

c. 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。

4. 在其他所有NTP客户端节点新建NTP文件。

a. 打开文件。

```
vi /etc/ntp.conf
```

b. 按“i”进入编辑模式，并在文件中新增如下内容作为客户端节点：

```
server 192.168.0.216
```

c. 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。

步骤3 启动和同步NTP服务。

1. 在192.168.0.216 NTP服务端节点启动NTP服务并查看NTP服务状态。

```
systemctl start ntpd
systemctl enable ntpd
systemctl status ntpd
```

```
[root@ceph1 ~]# systemctl start ntpd
[root@ceph1 ~]# systemctl enable ntpd
[root@ceph1 ~]# systemctl status ntpd
● ntpd.service - Network Time Service
   Loaded: loaded (/usr/lib/systemd/system/ntpd.service; enabled; vendor preset: disabled)
   Active: active (running) since Mon 2019-12-02 23:58:13 CST; 44min ago
 Main PID: 363629 (ntpd)
   CGroup: /system.slice/ntpd.service
           └─363629 /usr/sbin/ntpd -u ntp:ntp -g

Dec 02 23:58:13 ceph4 ntpd[363629]: Listen normally on 4 bond0 90.90.1.181 UDP 123
Dec 02 23:58:13 ceph4 ntpd[363629]: Listen normally on 5 bond1 90.90.2.181 UDP 123
Dec 02 23:58:13 ceph4 ntpd[363629]: Listen normally on 6 bond0 fe80::72c7:f2ff:fecb:4d04 UDP 123
Dec 02 23:58:13 ceph4 ntpd[363629]: Listen normally on 7 lo ::1 UDP 123
Dec 02 23:58:13 ceph4 ntpd[363629]: Listen normally on 8 bond1 fe80::72c7:f2ff:fecb:4d06 UDP 123
Dec 02 23:58:13 ceph4 ntpd[363629]: Listen normally on 9 enp189s0f0 fe80::3ee1:2c18:14ed:cb74 UDP 123
Dec 02 23:58:13 ceph4 ntpd[363629]: Listening on routing socket on fd #26 for interface updates
Dec 02 23:58:13 ceph4 ntpd[363629]: 0.0.0.0 c016 06 restart
Dec 02 23:58:13 ceph4 ntpd[363629]: 0.0.0.0 c012 02 freq_set kernel 0.000 PPM
Dec 02 23:58:13 ceph4 ntpd[363629]: 0.0.0.0 c011 01 freq_not_set
```

2. 在其他所有NTP客户端节点强制同步192.168.0.216 NTP服务端节点的时间。

```
ntpdate 192.168.0.216
```

3. 在除了192.168.0.216 NTP服务端节点的所有节点写入硬件时钟，以避免重启服务器后时间同步失效。

```
hwclock -w
```

4. 在除了192.168.0.216 NTP服务端节点的所有节点安装并启动crontab工具。

```
yum install -y crontabs
chkconfig crond on
systemctl start crond
crontab -e
```

在执行**crontab -e**命令后将打开当前用户的crontab文件，用于编辑定时任务。

5. 在打开的crontab文件中添加以下内容，设置每隔10min自动与192.168.0.216 NTP服务端节点同步时间。

```
*/10 * * * * /usr/sbin/ntpdate 192.168.0.216
```

6. 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。

----结束

1.8.4 启动 OceanBase 集群时提示 General socket error (Permission denied)的解决方法

问题现象描述

部署HAProxy服务过程中，无法连接HAProxy服务。查看HAProxy状态时提示：


```
General socket error (Permission denied)
```

关键过程、根本原因分析

HAProxy不允许被连接到其他端口。

结论、解决方案及效果

执行如下命令，允许HAProxy进程建立到任何地址和端口的连接。

```
getsebool haproxy_connect_any
setsebool -P haproxy_connect_any 1
```


2 OceanBase 调优指南

使用毕昇编译器对OceanBase数据库进行PGO（Profile-guided Optimization）优化，提升OceanBase的执行效率和性能，大致过程包括OceanBase源码准备、编译器配置、优化选项设置、PGO数据采集和基于PGO数据的优化。

2.1 调优概述

2.1.1 介绍

本文主要介绍如何在openEuler操作系统的鲲鹏服务器上调优OceanBase。

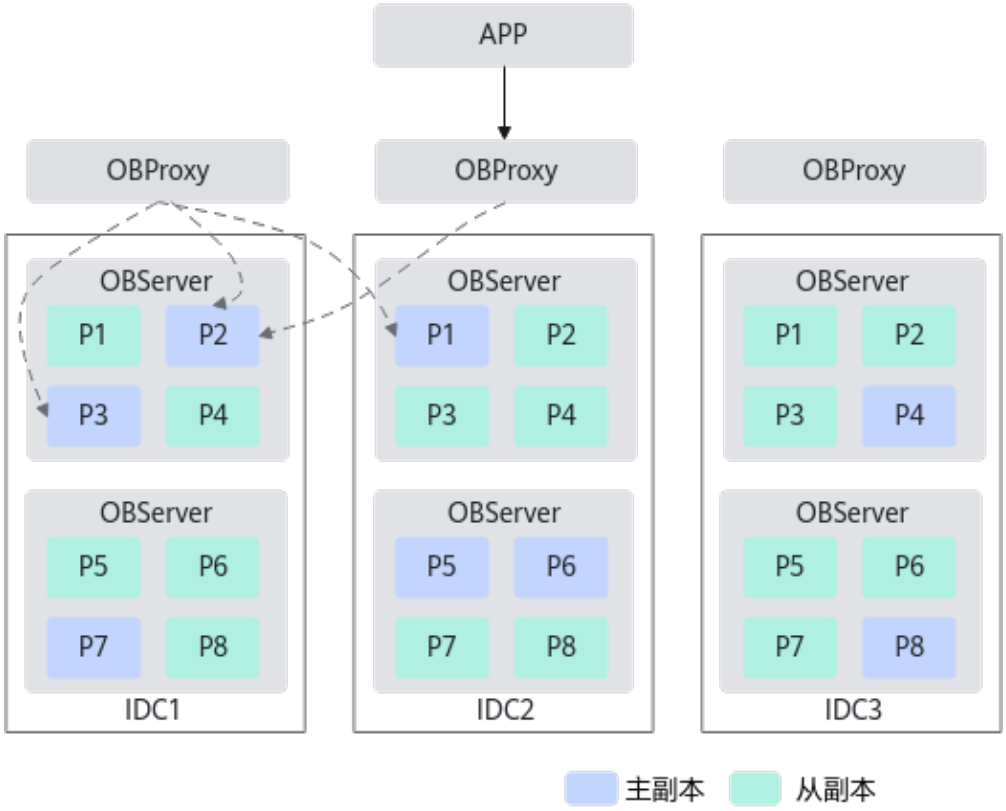
OceanBase数据库是一款完全自研的企业级开源分布式数据库，在普通硬件上实现金融级高可用，首创“三地五中心”城市级故障自动无损容灾新标准，刷新TPC-C标准测试，单集群规模超过1500节点，具有云原生、强一致性、企业版高度兼容Oracle/MySQL和社区版高度兼容MySQL等特性。

OceanBase数据库包括以下常用组件：

- OBD：OBD（OceanBase Deployer）是OceanBase开源软件的安装部署工具。OBD同时也是包管理器，可以用来管理OceanBase所有的开源软件。OBD的官方文档请参见**https://www.oceanbase.com/docs/community-obd-cn-10000000001031920**。
- OBServer：OceanBase数据库进程名。
- OBClient：OceanBase自带客户端连接工具，社区版OceanBase兼容MySQL，因此也可以使用MySQL客户端工具连接OceanBase。
- ODP：ODP（OceanBase Database Proxy，简称为OBProxy），是OceanBase数据库专用的服务代理。OBProxy的官方文档请参见**https://www.oceanbase.com/docs/community-odp-cn-10000000001717354**。

OBProxy中记录了OBServer的数据分布信息，用户的SQL会先发送给ODP节点，由ODP选择合适的Observer进行转发，并将结果返回给用户，如图2-1所示，这样的方式比直接转发到没有数据的节点执行效率更高。

图 2-1 OceanBase 数据转发方式



建议的版本

OceanBase 3.1.5或社区版4.2.1。

2.1.2 环境要求

本文基于鲲鹏服务器和openEuler操作系统提供指导，在正式操作前请确保软硬件均满足要求。

硬件要求

硬件要求如表2-1所示。

表 2-1 硬件要求

项目	说明
CPU	鲲鹏920系列处理器

项目	说明
硬盘	• 进行性能测试时，可以使用两块硬盘，即一块盘用于存储日志（clog）信息，另一块盘用于存储数据信息（数据盘可选用性能较好的SATA SSD盘、NVMe SSD盘等）。 • 非性能测试时，可以将日志信息与数据信息存储在同一块硬盘上。 • 具体硬盘数量根据实际需求配置。

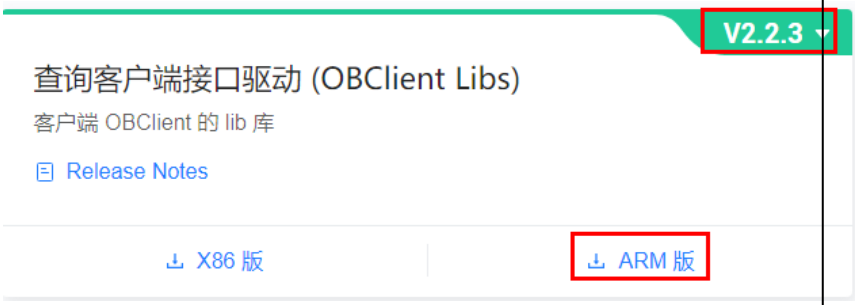
操作系统和软件要求

- 查看环境操作系统的信息：**cat /etc/*-release**
  查看环境处理器相关信息：**lscpu**
  查看环境内核版本：**uname -r**
  查看环境信息：**uname -a**
- 如果需要全新安装操作系统，可选择“Minimal Install”安装方式并勾选Development Tools套件，否则很多软件包需要手动安装。

操作系统要求如表2-2所示。

表 2-2 操作系统要求

项目和版本	下载地址
openEuler 20.03 LTS-SP1	https://repo.huaweicloud.com/openeuler/openEuler-20.03-LTS-SP1/ISO/aarch64/openEuler-20.03-LTS-SP1-everything-aarch64-dvd.iso

项目和版本	下载地址
OceanBase 3.1.5相关的RPM包	https://www.oceanbase.com/softwarecenter • libobclient-2.2.3-1.el7.aarch64.rpm  • obclient-2.2.3-1.el7.aarch64.rpm  • ob-deploy-2.3.1-2.el7.aarch64.rpm  • obproxy-ce-3.2.3.5-2.el7.aarch64.rpm  • oceanbase-ce-3.1.5-100020022023091114.el7.aarch64.rpm

项目和版本	下载地址
	 • oceanbase-ce-libs-3.1.5-100020022023091114.el7.aarch64.rpm
OceanBase 4.2.1社区版一键安装包	https://www.oceanbase.com/softwarecenter 

2.1.3 调优原则

性能调优是一个涉及多个层面的复杂过程，从硬件和操作系统的选择到子系统的设计和算法选择，都需要仔细考虑。在调优过程中，必须遵循一定的原则以确保得到正确的结果。

性能调优从大的方面来说，在系统设计之初，需要考虑硬件的选择、操作系统的选择以及基础软件的选择；从小的方面来说，包括每个子系统的设计、算法选择、如何使用编译器的选项，以及如何发挥硬件最大的性能等。

性能优化原则主要有以下几个方面：

- 对性能进行分析时，需要从多方面分析系统的资源瓶颈所在。因为系统如果在某一方面性能低，也许不是系统本身的原因，而是受到其他因素的影响。例如，CPU利用率100%可能是内存容量过小、CPU忙于处理内存调度的原因所导致的。
- 一次只对影响性能的某方面的一个参数进行调整，如果对多个参数同时进行调整，将难以界定影响性能的真正原因。
- 进行系统性能分析时，性能分析工具本身会占用一定的系统资源，如CPU资源、内存资源等，因此分析工具本身运行可能会导致系统某方面的资源瓶颈情况更加严重。

- 必须确保调优后的程序运行正确。
- 调优过程是持续的过程，每一次调优的结果都需要反馈到后续的版本开发中去。
- 性能调优不能以牺牲代码的可读性和可维护性为代价。

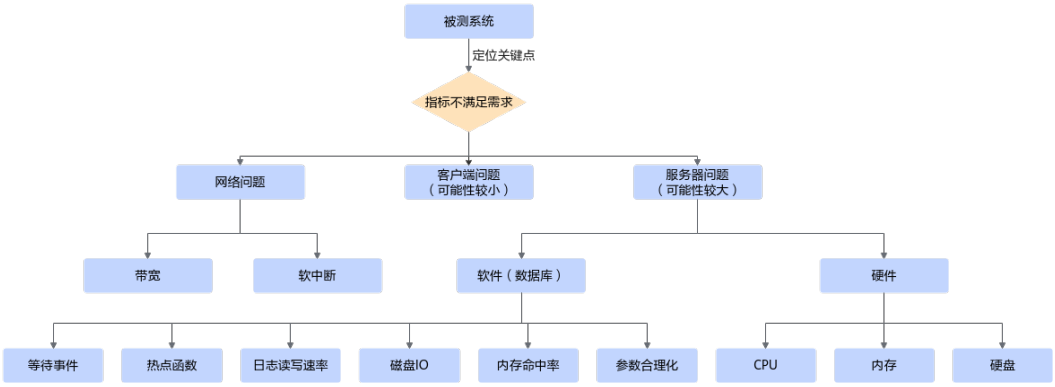
2.1.4 调优思路

调优思路主要围绕精准定位问题、分析性能瓶颈以及选择优化方法展开。

性能优化首先需要较为精准地定位问题，分析系统性能瓶颈，然后根据其性能指标以及所处层级选择优化的方式。

下面介绍OceanBase数据库具体的调优思路和分析过程，如图2-2所示。

图 2-2 调优思路



调优分析思路如下：

1. 很多情况下压测流量并没有完全进入到服务端，在网络上可能就会出现由于各种规格（带宽、最大连接数、新建连接数等）限制，导致压测结果达不到预期的情况。
2. 接着看关键指标是否满足要求，如果不满足，需要确定是哪个地方有问题，一般情况下，服务器端问题可能性比较大，也有可能是客户端问题（这种情况比较少）。
3. 对于服务器端问题，需要定位的是硬件相关指标，例如CPU、Memory、Disk IO、Network IO，如果是某个硬件指标有问题，需要深入地进行分析。
4. 如果硬件指标都没有问题，需要查看数据库相关指标，例如：等待事件、内存命中率等。
5. 如果以上指标都正常，应用程序的算法、缓冲、缓存、同步或异步可能有问题，需要具体深入的分析。
6. 对于OceanBase数据库，需要给对应租户分配足够的资源（也即CPU、内存等），否则性能可能达到不了预期。

可能的瓶颈点如表2-3所示。

表 2-3 可能的瓶颈点

瓶颈点	说明
硬件/规格	一般指的是CPU、内存、磁盘IO方面的问题，分为服务器硬件瓶颈、网络瓶颈（对局域网可以不考虑）。
操作系统	一般指的是Windows、UNIX、Linux等操作系统。例如，在进行性能测试，出现物理内存不足时，虚拟内存设置也不合理，虚拟内存的交换效率就会大大降低，从而导致行为的响应时间大大增加，这时认为操作系统上出现性能瓶颈。
数据库	一般指的是数据库配置等方面的问题。例如，由于参数配置不合理，创建租户时，分配的资源过少等问题。

2.2 硬件调优

2.2.1 BIOS 调优

对于不同的硬件设备，通过在BIOS中设置一些高级选项，可以有效提升服务器性能。

调优方法

SMMU（System Memory Management Unit）是IO设备与总线桥之间的一个地址转换桥，可以实现虚拟地址到物理地址的转换，同时还可以对内存访问进行权限控制和缓存管理，确保系统内存的安全和高效使用。因为数据库通常会使用大量的内存和IO资源，而SMMU会增加额外的开销和延迟，从而降低系统的性能。因此在数据库场景，开启SMMU并不能获得更好的性能。此外，关闭SMMU还可以减少系统的复杂性和维护成本。因此在鲲鹏平台，建议在BIOS中关闭SMMU。

硬件预取是通过跟踪指令和数据地址的变化，将指令和地址提前读到Cache里，硬件预取对数据库场景的性能有影响，建议在BIOS中关闭预取功能。

在BIOS设置中，内存刷新速率选项提供了“自动”选项，该选项可根据工作温度自动调节内存刷新速率，相比默认的32ms选项配置，可以提升内存性能。同时，该选项还可以确保工作温度在85℃到95℃时内存数据仍具有可靠性。

如果服务器场景为高性能场景，建议将电源策略设置为性能模式，确保CPU始终以最高频率运行。

- 关闭SMMU。

  说明

  此优化项只在非虚拟化场景使用，在虚拟化场景，则需要开启SMMU。

  a. 重启服务器，进入BIOS设置界面。

     具体操作请参见《**TaiShan 服务器 BIOS 参数参考（鲲鹏920处理器）**》中“进入BIOS界面”的相关内容。

  b. 依次选择“Advanced > MISC Config”，按“Enter”进入。

  c. 将“Support Smmu”设置为“Disabled”。

- 关闭硬件预取。

a. 在BIOS中，选择“Advanced>MISC Config”，按“Enter”进入。
b. 将“CPU Prefetching Configuration”设置为“Disabled”，按“F10”保存并退出。

- 设置内存刷新为Auto。
  a. 重启服务器，进入BIOS设置界面。
     具体操作请参见《**TaiShan 服务器 BIOS 参数参考（鲲鹏920处理器）**》中“进入BIOS界面”的相关内容。
  b. 选择“Advanced > Memory Config > Custom Refresh Rate”，按“Enter”进入。
  c. 设置“Custom Refresh Rate”选项为“Auto”，按“F10”保存保存并退出。
- 设置电源策略为性能模式。
  a. 重启服务器，进入BIOS设置界面。
     具体操作请参见《**TaiShan 服务器 BIOS 参数参考（鲲鹏920处理器）**》中“进入BIOS界面”的相关内容。
  b. 依次选择“BIOS > Advanced > Performance Config > Power Policy”，按“Enter”进入。
  c. 设置“Power Policy”选项为“ “Performance”，按“F10”保存保存并退出。

2.3 操作系统调优

2.3.1 文件系统调优

对于不同的IO设备，通过调整文件系统相关参数配置，可以有效提升服务器性能。

调优方法

本章节以xfs文件系统为例，介绍文件系统的调优步骤。

建议在文件系统的mount参数上加上noatime、nobarrier两个选项。操作命令如下，其中数据盘以及数据目录以实际为准。

```
mount -o /dev/sdb /data
```

- 一般来说，Linux会给文件记录了三个时间：
  - access time指文件最后一次被读取的时间。
  - modify time指的是文件的文本内容最后发生变化的时间。
  - change time指的是文件的inode（比如位置、用户属性、组属性等）最后发生变化的时间。

  一般情况下，文件都是读多写少，而且用户很少关注某一个文件最近什么时间被访问了。因此，建议采用noatime选项，文件系统在程序访问对应的文件或文件夹时，不会更新对应的access time。文件系统不记录access time，从而避免浪费资源。
- 很多文件系统会在数据提交时强制底层设备刷新Cache，以避免数据丢失，称为write barriers。但是，其实数据库服务器底层存储设备要么采用RAID控制卡，通过RAID控制卡本身的电池实现掉电保护；要么采用Flash卡，Flash卡也有自我保护机制，以保证数据不会丢失。因此可以安全地使用nobarrier挂载文件系统。

- 对于ext3、ext4和reiserfs文件系统，可以在挂载时指定barrier=0。
- 对于xfs，可以指定nobarrier选项。

2.3.2 网卡中断绑核

相比使用内核的irqbalance使网卡中断在所有核上进行调度的方式，使用手动绑核将中断固定住的方式更能有效提高业务网络收发包的能力。

须知

进行网卡中断绑核之前，需要先关闭irqbalance。

调优方法

步骤1 停止irqbalance服务。

```
systemctl stop irqbalance.service
```

步骤2 关闭irqbalance服务。

```
systemctl disable irqbalance.service
```

步骤3 查看irqbalance服务状态是否已关闭。

```
systemctl status irqbalance.service
```

状态为inactive即为关闭。

```
[root@localhost home]# systemctl stop irqbalance.service
[root@localhost home]# systemctl disable irqbalance.service
[root@localhost home]# systemctl status irqbalance.service
● irqbalance.service - irqbalance daemon
   Loaded: loaded (/usr/lib/systemd/system/irqbalance.service; disabled; vendor preset: enabled)
   Active: inactive (dead)
[root@localhost home]#
```

步骤4 查看网卡PCI设备号，假设当前网卡名为enp131s0。

```
ethtool -i enp131s0
```

```
[root@hadoop1 opt]# ethtool -i enp131s0
driver: hinic
version: 2.3.2.1
firmware-version: 2.3.2.1
expansion-rom-version:
bus-info: 0000:83:00.0
supports-statistics: yes
supports-test: yes
supports-eeprom-access: no
supports-register-dump: no
supports-priv-flags: no
```

步骤5 查看PCIe网卡所属的NUMA Node。请将以下命令中的bus-info替换为**步骤4**中查询到的具体值。

```
lspci -vvvs <bus-info>
```

```
[root@hadoop1 opt]# lspci -vvvs 0000:83:00.0
83:00.0 Ethernet controller: Huawei Technologies Co., Ltd. Hi1822 Family (4*25GE) (rev 45)
        Subsystem: Huawei Technologies Co., Ltd. Device d129
        Control: I/O- Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr+ Stepping- SERR+ FastB2B- DisINTx+
        Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR- INTx+
        Latency: 0, Cache Line Size: 32 bytes
        NUMA node: 2
        Region 0: Memory at 280003d00000 (64-bit, prefetchable) [size=128K]
        Region 2: Memory at 2800044a0000 (64-bit, prefetchable) [size=32K]
        Region 4: Memory at 280000000000 (64-bit, prefetchable) [size=1M]
        Expansion ROM at b0000000 [disabled] [size=1M]
        Capabilities: [40] Express (v2) Endpoint, MSI 00
```

步骤6 查看NUMA Node对应的Core的区间。例如此处就可以绑到NUMA Node 48～63。

```
lscpu
```

```
[root@hadoop1 opt]# lscpu
Architecture:          aarch64
Byte Order:            Little Endian
CPU(s):                96
On-line CPU(s) list:   0-95
Thread(s) per core:    1
Core(s) per socket:    48
Socket(s):             2
NUMA node(s):          4
Model:                 0
CPU max MHz:           2600.0000
CPU min MHz:           200.0000
BogoMIPS:              200.00
L1d cache:             64K
L1i cache:             64K
L2 cache:              512K
L3 cache:              49152K
NUMA node0 CPU(s):     0-23
NUMA node1 CPU(s):     24-47
NUMA node2 CPU(s):     48-71
NUMA node3 CPU(s):     72-95
```

步骤7 创建smartIrq.sh脚本。

1. 创建smartIrq.sh。
   ```
   vi smartIrq.sh
   ```
2. 按“i”进入编辑模式，在文件中添加以下内容：
   ```
   #!/bin/bash
   irq_list=(`cat /proc/interrupts | grep enp131s0 | awk -F: '{print $1}'`)
   cpunum=48  # 修改为所在Node的第一个Core
   for irq in ${irq_list[@]}
   do
   echo $cpunum > /proc/irq/$irq/smp_affinity_list
   echo `cat /proc/irq/$irq/smp_affinity_list`
   (( cpunum+=1 ))
   done
   ```

   说明

   enp131s0为用于绑定队列的网卡名称，请根据实际情况修改。
3. 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。

步骤8 进行中断绑核。1822网卡共有16个队列，将这些中断逐个绑定至所在NUMA Node的16个Core上。例如本例中绑核到NUMA Node2对应的48～63上面。

```
bash smartIrq.sh
```

步骤9 创建irqCheck.sh脚本。

1. 创建irqCheck.sh脚本。
   ```
   vi irqCheck.sh
   ```
2. 按“i”进入编辑模式，在文件中添加以下内容：
   ```
   #!/bin/bash
   # 网卡名
   intf=$1
   log=irqSet-`date "+%Y%m%d-%H%M%S"`.log
   # 可用的CPU数
   cpuNum=$(cat /proc/cpuinfo |grep processor -c)
   # RX TX中断列表
   ```



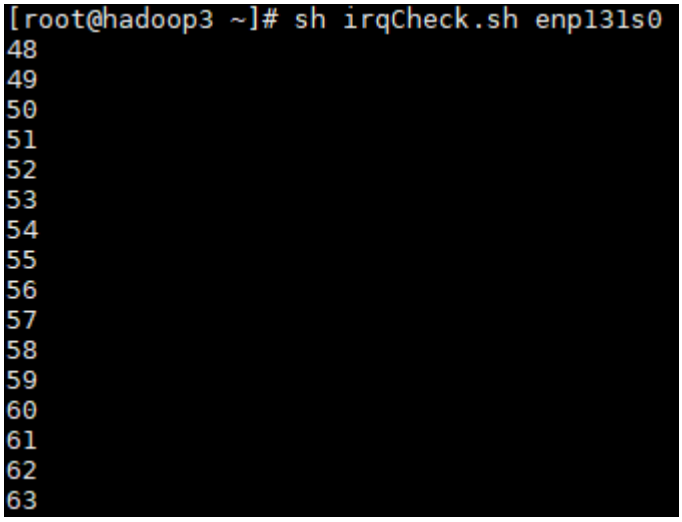
```
irqListRx=$(cat /proc/interrupts | grep ${intf} | awk -F':' '{print $1}')
irqListTx=$(cat /proc/interrupts | grep ${intf} | awk -F':' '{print $1}')
# 绑定接收中断rx irq
for irqRX in ${irqListRx[@]}
do
cat /proc/irq/${irqRX}/smp_affinity_list
done
# 绑定发送中断tx irq
for irqTX in ${irqListTx[@]}
do
cat /proc/irq/${irqTX}/smp_affinity_list
done
```

3. 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。

步骤10 利用脚本查看是否绑核成功，假设当前网卡名为enp131s0。

```
sh irqCheck.sh enp131s0
```

绑核成功结果示例：



----结束

2.3.3 操作系统参数调优

对于不同的操作系统，通过在操作系统层面调整一些参数配置，可以有效提升服务器性能。

调优方法

通过设置如表2-4所示的系统参数进行调优。

表 2-4 操作系统参数优化建议

参数类型	参数名称	参数含义	建议
网络参数	net.core.somaxconn	Socket侦听队列的最大长度，频繁建立连接需要调大该值。	默认为128，建议配置为2048。

参数类型	参数名称	参数含义	建议
	net.core.netdev_max_backlog	协议栈处理的缓冲队列长度，设置的过小有可能造成丢包。	建议配置为10000。
	net.core.rmem_default	接收缓冲区队列的默认长度。	建议配置为16777216。
	net.core.wmem_default	发送缓冲区队列的默认长度。	建议配置为16777216。
	net.core.rmem_max	接收缓冲区队列的最大长度。	建议配置为16777216。
	net.core.wmem_max	发送缓冲区队列的最大长度。	建议配置为16777216。
	net.ipv4.ip_local_port_range	本地TCP/UDP的端口范围，本地使用该范围内的端口与远端发起连接。	建议的端口范围为3500～65535。
	net.ipv4.tcp_rmem	Socket接收缓冲区的大小，分别为最小值、默认值、最大值。	建议最小值、默认值、最大值分别配置为4096、87380、16777216。
	net.ipv4.tcp_wmem	Socket发送缓冲区的大小，分别为最小值、默认值、最大值。	建议最小值、默认值、最大值分别配置为4096、65536、16777216。
	net.ipv4.tcp_max_syn_backlog	处于SYN_RECVD状态的连接数。	建议配置为16384。
	net.ipv4.tcp_fin_timeout	Socket主动断开之后FIN-WAIT-2状态的持续时间。	建议配置为15。
	net.ipv4.tcp_tw_reuse	允许重用处于TIMEWAIT状态的Socket。	建议配置为1。
	net.ipv4.tcp_slow_start_after_idle	禁止TCP连接从Idle状态的慢启动，降低某些情况的网络延迟。	建议配置为0。
虚拟内存配置	vm.swappiness	优先使用物理内存。	建议配置为0。
	vm.max_map_count	进程可以拥有的虚拟内存区域数量。	建议配置为655360。
AIO配置	fs.aio-max-nr	异步IO的请求数目。	建议配置为1048576。

查看及修改操作系统相关参数

须知

请修改操作系统内核参数，因为这些参数对系统性能和稳定性具有重要影响。建议您在经验丰富的系统管理员的指导下进行此操作。

表2-4所示的系统参数均可以通过如下两种方式查看和修改。

- 以下命令只会临时修改参数值，重启操作系统后参数将会恢复到默认值。

  例如查看当前系统中net.core.somaxconn的值：
  ```
  sysctl net.core.somaxconn
  ```
  如果需要修改net.core.somaxconn的值，可以通过如下命令设置：
  ```
  sysctl -w net.core.somaxconn=NEW_VALUE
  ```
- 如果想要永久修改net.core.somaxconn的值，需要修改“/etc/sysctl.conf”配置文件并使配置文件生效，操作步骤如下：

  a. 打开“/etc/sysctl.conf”文件。
  ```
  vi /etc/sysctl.conf
  ```
  b. 按“i”进入编辑模式，在“/etc/sysctl.conf”配置文件中添加以下语句：
  ```
  net.core.somaxconn=NEW_VALUE
  ```
  c. 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。

  d. 使用以下命令使修改的参数值生效。
  ```
  sysctl -p
  ```

2.3.4 GRUB 参数调优

通过修改GRUB（GRand Unified Bootloader）配置来优化操作系统启动和运行时的性能，包括修改**nohz**、**cgroup_disable**、**idle**和**numa**参数。

步骤1 修改GRUB配置文件。

1. 打开文件。
   ```
   vim /etc/default/grub
   ```
2. 按“i”进入编辑模式，修改启动参数。

   在文件中找到“GRUB_CMDLINE_LINUX”这一行，添加如下内容。
   ```
   nohz=off cgroup_disable=files idle=poll
   ```
   删除如下内容：
   ```
   numa=off
   ```
3. 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。

步骤2 生成新的GRUB配置文件。
```
grub2-mkconfig -o /boot/efi/EFI/bclinux/grub.cfg
```

步骤3 重启操作系统，使GRUB配置的更改生效。
```
reboot
```

步骤4 验证修改是否生效。

重启操作系统完成后，查看“/proc/cmdline”文件，如果已经包含**nohz=off cgroup_disable=files idle=poll**和已经删除**numa=off**，表示GRUB配置的更改已生效。

```
cat /proc/cmdline
```

----结束

2.3.5 启用 steal task 功能

启用steal task功能以进行系统调度的优化。

当一个进程正在执行，而另一个进程需要执行但没有CPU时间片可用时，steal task功能会从正在运行的进程中抢占一部分CPU时间片，分配给需要执行的进程。

步骤1 确认当前内核是否支持steal task功能。

```
cat /sys/kernel/debug/sched_features |tr " " "\n" |grep STEAL
```

如果输出中包含**STEAL**或**NO_STEAL**（NO_STEAL意味着STEAL功能存在但可能被禁用），则表明当前内核支持steal task功能。

步骤2 配置steal task功能。

1. 打开文件。
   ```
   vim /etc/default/grub
   ```
2. 按“i”进入编辑模式，修改启动参数。

   在文件中找到GRUB_CMDLINE_LINUX这一行，添加如下内容：
   ```
   sched_steal_node_limit=8
   ```
   其中，8为NUMA的个数，请根据实际情况修改。
3. 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。

步骤3 生成新的GRUB配置文件。

```
grub2-mkconfig -o /boot/efi/EFI/bclinux/grub.cfg
```

步骤4 重启操作系统，使GRUB配置的更改生效。

```
reboot
```

步骤5 通过写入“/sys/kernel/debug/sched_features”文件来启用steal task功能。

```
echo STEAL > /sys/kernel/debug/sched_features
```

说明

如果需要禁用steal task功能，请使用如下命令。

```
echo NO_STEAL > /sys/kernel/debug/sched_features
```

----结束

2.3.6 开启并设置大页内存

为了优化内存使用并提升TLB（Translation Lookaside Buffer）的命中率，可以通过配置大页内存来实现。

设置内存大页后，大页内存仅大页内存程序使用，会一直占用内存，因此，需要手动管理大页内存，考虑其他应用占用内存的情况再进行分配。

步骤1 在数据库中配置。

1. 启用大页内存。

   在OceanBase的OBServer配置中，需要设置“use_large_pages”参数为“true”，以指示数据库使用大页内存。
   ```
   alter system set use_large_pages = "true";
   ```


2. 重启OBServer服务，使配置生效。
3. 查看大页内存是否启用成功。
```
SHOW PARAMETERS LIKE 'use_large_pages';
```
命令返回“use_large_pages”的值为“true”，表示OBServer已经启用大页内存。

步骤2 在操作系统中配置。

1. 查看操作系统支持的大页大小。
```
cat /proc/meminfo |grep Huge
```
在本例中，当前操作系统支持的大页大小为2048kB。
```
[root@localhost bin]# cat /proc/meminfo |grep Huge
AnonHugePages:  276897792 kB
ShmemHugePages:        0 kB
FileHugePages:         0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
Hugetlb:               0 kB
```
2. 设置大页内存数量。在本例中，需要分配的大页内存数量是256000。
```
echo 256000 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages
```
上述命令会请求系统为应用程序分配256000个2048kB大小的大页。

----结束

2.4 数据库参数调优

2.4.1 OBServer 调优

通过调整OBServer参数，可以提升OceanBase的性能。

调优方法

本章节针对OBServer的CPU、IO、内存和网络传输相关参数提供了调优建议。

表 2-5 CPU 相关参数调优建议

配置项	描述	建议
workers_per_cpu_quota	表示租户在每个CPU上能分配的最大工作线程数。	该参数是指系统分配给程序可使用的最大线程数量，而不是程序同时运行的最大线程数。一般情况下，这个参数不需要进行调整。

配置项	描述	建议
net_thread_count	用于设置网络IO线程数。	通过**top -H**命令可以查看MySQL IO线程的利用率。如果发现某个线程的利用率超过90%，则该线程可能成为瓶颈，建议增加该参数以提高性能。如果所有线程的利用率都低于50%，则建议减小该参数，以减少线程切换的开销。 **须知** 修改该参数后，需要重启OceanBase集群才能生效。
autoinc_cache_refresh_interval	用于设置自动刷新后台线程的工作间隔时间。单位为s。	后台线程进行自动刷新会引起性能波动，建议将该参数值调大，以减少后台线程自动刷新的频率。
enable_early_lock_release	用于控制在事务中锁定的行是否可以在事务提交之前被释放。 该参数为bool类型，值为true时表示开启，值为false时表示关闭。	建议在热点行场景下开启该功能，即将该参数设置为true。
enable_monotonic_weak_read	用于控制是否启用弱一致性读取。 该参数为bool类型，值为true时表示开启，值为false时表示关闭。	建议在性能场景下关闭该功能，即将该参数设置为false。
weak_read_version_refresh_interval	用于设置弱一致性读版本号的刷新周期，该参数影响弱一致性读数据的延时。单位为ms。	当该值为0时，将停止刷新弱一致性读版本号，不再提供单调读功能。建议在性能场景下关闭该功能以提高性能，即将该参数设置为0。
enable_sql_audit	用于启用SQL审计功能。 该参数为bool类型，值为true时表示开启，值为false时表示关闭。	在生产环境中必须启用该功能，即将该参数设置为true；但在性能测试场景中可以根据需要进行关闭，即将该参数设置为false。
enable_perf_event	用于设置是否开启性能事件的信息收集功能。 该参数为bool类型，值为true时表示开启，值为false时表示关闭。	在生产环境中必须启用该功能，即将该参数设置为true；但在性能测试场景中可以根据需要进行关闭，即将该参数设置为false。
enable_record_trace_log	用于控制是否启用记录跟踪日志。 该参数为bool类型，值为true时表示开启，值为false时表示关闭。	在启用Perf Event和SQL Audit功能的生产环境中，可以关闭记录跟踪日志功能，即将该参数设置为false。

表 2-6 IO 相关参数

配置项	描述	建议
syslog_io_bandwidth_limit	用于限制syslog-ng在写入日志文件时的带宽。默认值为30MB。	建议设置为10MB。
clog_sync_time_warn_threshold	当Clog同步的时间超过该阈值时，会记录一条警告信息。	建议将该参数调大。但是需要注意，调大该参数可能会对解决Clog同步慢问题的调查造成影响。
trace_log_slow_query_watermark	在执行查询时，如果查询的执行时间超过该参数设置的阈值，则会被记录在慢查询日志中。	建议将该参数调大，以避免打印慢查询的跟踪日志。
max_syslog_file_count	observer.log的最大数量，超过这个数量的日志文件将被自动删除。	建议根据磁盘容量调整该参数值。
enable_sql_operator_dump	用于设置是否允许SQL处理过程的中间结果写入磁盘以释放内存。 该参数为bool类型，值为true时表示开启，值为false时表示关闭。	在OLAP场景下测试时，建议设置为true，以避免由于内存过大而产生报错。
builtin_db_data_verify_cycle	用于设置数据坏块自检周期。单位为天。 当设置为0时表示关闭自检。	性能场景下，建议设置为0。
disk_io_thread_count	用于设置磁盘IO线程数。 **须知** 该参数值必须是偶数。	根据IO线程数的压力适当调整该参数值的大小。
enable_async_syslog	用于开启或关闭异步syslog日志记录。 该参数为bool类型，值为true时表示开启，值为false时表示关闭。	性能场景下，建议开启异步syslog日志记录，即将该参数设置为true。

配置项	描述	建议
fuse_row_cache_priority	用于设置融合行缓存在缓存系统中的优先级。 融合行缓存即在OceanBase数据库多级存储架构下，如果一行的多个列存在于memstore、mini sstable、minor sstable和sstable，那么在查询该行时，需要将多个列做融合，此时可以用fuse row cache来缓存该行，避免在下次查询时继续做融合操作。	建议调整为大于1的值，以避免行缓存过早被淘汰替换。
syslog_level	用于设置系统日志的级别。	性能场景下，建议将syslog_level设置为PERF。
merge_thread_count	用于设置合并调度线程的数量。 默认值为10。	根据系统的具体负载和日志处理需求，建议酌情调大merge_thread_count的值。

表 2-7 内存相关参数调优建议

配置项	描述	建议
memory_limit_percentage	指允许OceanBase数据库可用的内存量占系统总内存的比例。	建议提高该比例，以增加OceanBase数据库可用的内存量。
memstore_limit_percentage	用于设置租户使用MemStore的内存占其总可用内存的百分比。	建议尽量增大MemStore的空间。如果该值设置过大可能会存在由于转储速度跟不上写入速度导致内存超限的风险。
freeze_trigger_percentage	用于设置触发全局冻结的租户使用内存阈值。	对于写入压力比较大的系统，建议可以调整该参数值为30～50之间，以实现尽快触发转储，以防止内存不足。触发转储会带来额外的CPU和IO开销，且频繁触发转储后，mini sstable的个数会增加，增加了查询路径，从而对性能产生一定影响。该参数从OceanBase数据库V4.0版本开始改为租户级配置项。
use_large_pages	用于启用或禁用大页内存。 **须知** 需要在OS端开启内存大页功能，此功能才会生效。	建议设置为true，开启大页内存，以提高内存页表的查询效率。

表 2-8 网络传输相关参数调优建议

配置项	描述	建议
__easy_memory_limit	用于控制发往单个OBServer的rpc packet的最大内存量。默认值为4GB。	如果查询的数据量较大，建议将该配置项调大。
ob_proxy_readonly_transaction_routing_policy	用于控制Proxy对于事务的路由是否受只读语句的影响。 该参数为bool类型，值为true时表示开启，值为false时表示关闭。	建议将该参数设置为false，表示Proxy对于事务的路由以第一条实际开启事务的语句为准。

查看及修改 OBServer 的相关参数

步骤1 使用OBClient连接OceanBase。

```
obclient -h127.0.0.1 -P2883 -uroot@sysbench_tenant -Doceanbase -A
```

说明

以下参数请根据实际情况修改：

- 127.0.0.1和2883为OceanBase服务器的IP地址和端口号。
- root为数据库用户，sysbench_tenant为数据库租户。
- oceanbase为OceanBase数据库的名称。

步骤2 查看OBServer参数。例如查看workers_per_cpu_quota：

```
SHOW PARAMETERS LIKE 'workers_per_cpu_quota';
```

步骤3 设置OBServer参数。例如将workers_per_cpu_quota的值设置为20：

```
alter system set workers_per_cpu_quota = 20;
```

----结束

2.4.2 OBProxy 调优

通过调整OBProxy参数，可以提升OceanBase的性能。

调优方法

本章节针对OBProxy的CPU相关参数提供了调优建议。如所示中的参数修改后，需要重启OBProxy才能生效。

表 2-9 CPU 相关参数调优建议

配置项	描述	建议
work_thread_num	OBProxy的工作线程数。 对CPU占用影响比较大。默认值为8。	可根据环境动态调整，OBProxy的CPU使用上限为work_thread_num的值。
automatic_match_work_thread	判断是否根据CPU核数自动创建工作线程。 默认值为true，值为true时，工作线程数的上限为work_thread_num的值。	• 如果服务器上同时部署OBProxy和OBServer，会抢占CPU，建议关闭。 • 如果OBProxy单独部署，建议开启。
enable_compression_protocol	用于指定是否启用压缩协议。 该参数为bool类型，值为true时表示开启，值为false时表示关闭。	建议配置为true，开启压缩协议，以减少OBProxy对CPU的占用。

查看及修改 OBProxy 的相关参数

步骤1 使用OBClient连接OceanBase。

```
obclient -h127.0.0.1 -P2883 -uroot@sysbench_tenant -Doceanbase -A
```

说明

- 127.0.0.1和2883为OceanBase服务器的IP地址和端口号。
- root为数据库用户，sysbench_tenant为数据库租户。
- oceanbase为OceanBase数据库的名称。

步骤2 查看Proxy参数。例如查看enable_compression_protocol参数：

```
SHOW PROXYCONFIG LIKE 'enable_compression_protocol';
```

步骤3 设置Proxy参数。

```
ALTER PROXYCONFIG SET enable_compression_protocol = ture;
```

步骤4 重启OceanBase集群。

须知

部分参数修改后需重启OceanBase集群后才能生效。

```
obd cluster restart obcluster
```

----结束

2.5 毕昇编译器

使用毕昇编译器对OceanBase数据库进行PGO（Profile-guided Optimization）优化，提升OceanBase的执行效率和性能，大致过程包括OceanBase源码准备、编译器配置、优化选项设置、PGO数据采集和基于PGO数据的优化。

步骤1 以3.1.5版本为例，下载OceanBase源码并进入源码目录，初始化并编译源码。

```
git clone https://github.com/oceanbase/oceanbase oceanbase-3.1.5_CE
cd oceanbase-3.1.5_CE
bash build.sh release --init --make
```

说明

您也可以通过**https://github.com/oceanbase/oceanbase/archive/refs/tags/v3.1.5_CE.zip**下载OceanBase 3.1.5源码并存放至目标路径。

等待编译完成。

步骤2 下载并解压毕昇编译器到指定目录，例如“/home”。

编译器版本需要选择4.0.0。毕昇编译器源码下载链接：**https://www.hikunpeng.com/developer/devkit/download/bishengcompiler**

步骤3 修改CMake配置文件以适配毕昇编译器。

1. 修改“cmake/Env.cmake”文件。
   a. 打开“cmake/Env.cmake”文件。
   ```
   cd oceanbase-3.1.5_CE
   vim cmake/Env.cmake
   ```
   b. 按“i”进入编辑模式，将“DEVTOOLS_DIR”设置为毕昇编译器的路径；并添加Clang的头文件路径到“BUILD_OPT”，其中，x.x.x为Clang对应的版本号，请根据实际安装的版本修改。
   ```
   ob_define(DEVTOOLS_DIR "/home/BiShengCompiler4.0")
   set(BUILD_OPT "${BUILD_OPT} -I${DEVTOOLS_DIR}lib/clang/x.x.x/include/")
   ```
   c. 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。
2. 修改“./deps/easy/CMakeLists.txt”和“/deps/oblib/src/CMakeLists.txt”文件。
   a. 打开“cmake/Env.cmake”文件。
   b. 按“i”进入编辑模式，修改“./deps/easy/CMakeLists.txt”文件的第46行和“/deps/oblib/src/CMakeLists.txt”文件的第27行，去除“-Werror”选项，避免编译警告被当作错误处理。
   c. 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。

步骤4 配置编译选项以支持PGO。

1. 打开“cmake/Env.cmake”文件。
   ```
   vim cmake/Env.cmake
   ```
2. 按“i”进入编辑模式，为“CMAKE_CXX_FLAGS”和“CMAKE_C_FLAGS”添加优化选项。

   找到“CMAKE_CXX_FLAGS”和“CMAKE_C_FLAGS”并将这两个选项设置为与以下内容一致。
   ```
   set(CMAKE_CXX_FLAGS "${BUILD_OPT} -mcpu=tsv110 -fuse-ld=lld -Wno-register -Wl,-q -no-pie -flto=thin -mllvm -enable-aggressive-inline -Wl,-mllvm,-enable-aggressive-inline -mllvm -aggressive-inline-level=3 -Wl,-mllvm,-aggressive-inline-level=3 ")
   set(CMAKE_C_FLAGS "${BUILD_OPT} -mcpu=tsv110 -fuse-ld=lld -Wno-register -Wl,-q -no-pie -flto=thin -mllvm -enable-aggressive-inline -Wl,-mllvm,-enable-aggressive-inline -mllvm -aggressive-inline-level=3 -Wl,-mllvm,-aggressive-inline-level=3 ")
   ```


3. 启用PGO的插桩编译。

   找到“CMAKE_CXX_FLAGS”和“CMAKE_C_FLAGS”变量，为这两个选项都添加“-fprofile-generate”选项和“-Wno-backend-plugin”选项，并指定采样数据存放路径（例如“/opt/profile”，请根据实际情况自定义）。修改示例参考如下：

   ```
   set(CMAKE_CXX_FLAGS "${BUILD_OPT} -fprofile-generate=/opt/profile -Wno-backend-plugin")
   set(CMAKE_C_FLAGS "${BUILD_OPT} -fprofile-generate=/opt/profile -Wno-backend-plugin")
   ```

4. 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。

步骤5 重新编译OceanBase源码以生成插桩版本的OceanBase。

```
cd oceanbase-3.1.5_CE
bash build.sh release  --make
```

编译完成将生成插桩版本的OBServer二进制文件。

步骤6 采集性能数据。

1. 将插桩版本的OBServer替换到OceanBase集群中，并启动OceanBase集群。
2. 运行测试模型，如Sysbench或TPC-C等，对OceanBase集群进行测试，生成性能数据。
3. 停止OBServer进程，收集生成的采样数据文件。

   ```
   kill `pidof observer`
   ```

   上述命令执行完成后，将生成采样数据文件（例如xxx.profraw）。
4. 使用llvm-profdata工具将xxx.profraw文件合并成.profdata文件。

   ```
   llvm-profdata merge xxx.profraw -output=xxx.profdata
   ```

步骤7 使用性能数据进行优化编译。

需要修改OceanBase源码中的“cmake/Env.cmake”文件。

1. 打开“cmake/Env.cmake”文件。

   ```
   cd oceanbase-3.1.5_CE
   vim cmake/Env.cmake
   ```

2. 按“i”进入编辑模式，将“CMAKE_CXX_FLAGS”和“CMAKE_C_FLAGS”的编译选项从“-fprofile-generate”更改为“-fprofile-use”，并指定xxx.profdata文件的路径。修改示例参考如下：

   ```
   set(CMAKE_CXX_FLAGS "${BUILD_OPT} -fprofile-use=/opt/profile/xxx.profdata -Wno-backend-plugin")
   set(CMAKE_C_FLAGS "${BUILD_OPT} -fprofile-use=/opt/profile/xxx.profdata -Wno-backend-plugin ")
   ```

3. 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。

步骤8 重新编译OceanBase。

```
cd oceanbase-3.1.5_CE
bash build.sh release  --make
```

编译完成将生成优化后的OBServer二进制文件，将优化后的OBServer替换到OceanBase集群中，并启动OceanBase集群。

----结束

A 修订记录

发布日期	修订记录
2024-09-30	第二次正式发布。 • 新增OceanBase 3.1.5的部署操作指导，新增**1.4.1 安装OceanBase 3.1.5**，刷新**1.1 介绍**、**1.2 环境要求**和**1.7 验证OceanBase**。 • 修改**2.2.1 BIOS调优**，新增设置内存刷新为Auto、开启NUMA和设置电源策略为性能模式操作步骤。 • 修改**表2-6**，新增syslog_level、merge_thread_count参数的参数说明和调优建议。 • 新增**2.3.4 GRUB参数调优**、**2.3.5 启用steal task功能**、**2.3.6 开启并设置大页内存**、**2.5 毕昇编译器**。
2024-02-19	第一次正式发布。