

# OceanBase 2.2 独立部署手册

作 者 ： OB 庆涛

最后更新：2020/5/26

# 目录

<b>目录</b>	<b>2</b>
<b>引语</b>	<b>4</b>
作者简介	4
文档受众和主旨	4
文档（更新）获取、反馈途径	4
相关文档	5
约定	5
<b>1. 前言</b>	<b>6</b>
1.1. OCEANBASE 产品简介	6
1.2. OCEANBASE 部署概述	7
<b>2. 环境准备</b>	<b>8</b>
2.1. 软件包说明	8
2.2. 标准化环境的说明	9
2.3. 机器初始化 I（手动做）	12
2.4. 机器初始化 II（自动做）	15
2.5. 机器初始化 II（手动做，可选）	21
<b>3. 部署 OCP(可选)</b>	<b>25</b>
3.1. OCP 安装配置文件	25
3.2. 开始安装 OCP	26
<b>4. 独立部署 OCEANBASE 集群</b>	<b>29</b>
4.1. 机器初始化	29
4.2. 安装目录和软件	29
4.3. 初始化 OB 集群	32
4.4. 初始化 OBPROXY	36
4.5. OCEANBASE 分配多租户	38
<b>5. 使用 OCP 部署 OCEANBASE 集群</b>	<b>42</b>
5.1. 机器资源管理	42
5.2. 软件管理	44
5.3. 集群管理	44
5.4. 实例管理	49
5.5. OBPROXY 运维	53
<b>6. OCEANBASE 常用运维和开发经验</b>	<b>54</b>
6.1. 参数和变量初始化	54

6.2.	性能监控经验 .....	57
6.3.	日志使用经验 .....	61
6.4.	常用 SQL .....	61

## 引语

### 作者简介

本人是 OceanBase 产品解决方案架构师，以前是 ORACLE/MySQL/SQL Server DBA，支持过阿里巴巴 B2B 业务、淘宝阿里旺旺业务、阿里云数据库业务、天猫数据库。有个人微信公众号：obpilot，专注分享阿里数据库技术、分布式数据库技术以及相关的互联网产品。有兴趣的朋友可以关注一下。



本人也是 OceanBase 售前，欢迎对 OceanBase 产品有兴趣的企业朋友联系我做产品咨询和交流。联系方式：个人公众号回复“加好友”。谢谢！

### 文档受众和主旨

本文档是为任何想了解 OceanBase 数据库的学生和学者、开发和运维朋友编写的，主要是介绍 OceanBase 在**非标准环境**下（即不满足产品官方文档列举的最低条件）的独立部署方法，该方法也适合标准环境。

本文档是个人编写，基于个人多年经验，难免失误，仅用于学习。**本文档不是官方文档**。OceanBase 客户生产环境部署请按 OceanBase 产品交付文档部署或由 OceanBase 产品交付人员部署。

### 文档（更新）获取、反馈途径

本文档内容会根据网友问题逐步完善更新。

本文档发布以及以后更新可以从以下途径获取和反馈：

- 通过个人公众号（obpilot），发消息“加好友”。
- 通过 github 网址：<https://github.com/obpilot/ob-docs>
- 通过 OceanBase 钉钉交流群（21949783），在群里 @OB 庆涛

OceanBase技术交流直播群

365人



扫一扫群二维码，立刻加入该群。

## 相关文档

- [OceanBase 2.2 ORACLE 实例和 MySQL 实例开发者指南](#)

## 约定

本文档使用的文本格式约定如下

约定	含义
<b>粗体</b>	粗体字表示与操作相关联的图形用户界面元素，或文本或术语表中定义的术语。
<i>斜体</i>	斜体类型表示您提供特定值的标题、强调或占位符变量。
等宽字体	等宽字体表示段落中的命令、URL、示例中的代码、屏幕上出现的文本或输入的文本。

# 1. 前言

## 1.1. OceanBase 产品简介

OceanBase 是阿里巴巴和蚂蚁金服完全自主研发的分布式关系型数据库，自 2010 年开始开发到现今也有近 10 年历史了。

OceanBase 历经三个大的版本：0.x、1.x 和 2.x。1.x 版本目前到了 1.4 比较成熟，2.x 版本目前到了 2.2。

OceanBase 当前的主要业务场景是在金融，但并不限于金融。除了支付宝和淘宝外，OceanBase 还在网商银行、南京银行、苏州银行、招生证券、人保财险、西安银行等客户开始使用。蚂蚁金融云和阿里云上也有上百家金融客户业务运行在 OceanBase 上。

阿里巴巴和蚂蚁金服的数据库发展路线是两条独立的。在早期都面临去 ORACLE 问题时，阿里巴巴走了分布式 MySQL 数据库这条路线，蚂蚁金服走了分布式 ORACLE 数据库这条路线，随后才逐步用 OceanBase 将 ORACLE 替换掉。所以蚂蚁金服的数据库架构完整的说是 SOFA 体系的分布式数据访问代理 DBP 加分布式数据库 OceanBase。蚂蚁金服之所以不用 MySQL 作为金融数据库是因为 MySQL 在高可用和数据强一致方面的天然缺陷。

OceanBase 的分布式数据库架构是原生的，跟分布式数据库中间件有根本的区别。OceanBase 有着很多独特的功能。这里就不深入介绍，有兴趣的朋友可以查看我的个人公众号。

### **OceanBase 的核心能力包括：**

- 一个集群跨机房跨地域部署，天然适合数据库异地多活或容灾建设。
- 极致的高可用能力。故障时自动恢复（RTO 约 30s），保证不丢数据（RPO 为 0）。
- 成熟的水平扩展能力。可以在线弹性伸缩、数据迁移和负载均衡，对业务影响低。现有集群最大规模高达 207 台。
- 兼容 MySQL 和 ORACLE 常用 SQL 用法。具体兼容 MySQL 5.6 和 ORACLE 11g 的常用 SQL 用法。
- 部署成本低。架构在普通的 x86 服务器上，不依赖共享存储、小型机和光纤网络。目前也支持 arm 架构的国产服务器。
- 支持多租户（即多实例），按需分配。在客户机房部署后，像云数据库，但跟云设施没有绑定关系。
- 高性能。2019 年 OceanBase 通过 TPC 委员会的 TPC-C 认证，并取得 6088 万 tpmC。

## 1.2.OceanBase 部署概述

OceanBase 集群可以部署在普通的物理服务器、虚拟机（VMWare、Docker 或云服务器 ECS），前提是资源满足 OceanBase 要求。

OceanBase 公有云业务在阿里云上，4 月份正式对全球售卖。其原理是客户购买相应规格的 OceanBase 实例，产品会自动购买匹配规格的 ECS 去部署 OceanBase 集群。然后客户在这个集群里可以自己申请 ORACLE 或 MySQL 实例。

OceanBase 公有云产品的详细介绍请参考“<https://www.aliyun.com/product/oceanbase>”。公有云 OceanBase 集群的部署和后台运维是由 OceanBase 产品和公有云支持团队完成，这里就不需要介绍。

OceanBase 集群还有部署在客户私有云环境的。这里只支持阿里云的私有云（也有叫专有云），专有云的本质也是在物理服务器上部署一个飞天云操作系统（Apsaras）。OceanBase 的运维产品 OCP 有跟飞天对接，所以部署也是自动化的，不需要客户手动操作，这里也不介绍。

OceanBase 集群更多的场景是部署在物理服务器上，官方提供的试用版下载里的安装文档要求 OceanBase 生产服务器在 392G 内存以上，开发环境至少 128 G 内存。这里只提内存是因为这是关键因素。此外还对 CPU、磁盘和网络有要求。如果机器满足条件，严格安装官方文档要求的步骤去执行，基本上都能部署成功。但如果机器不符合条件，则很难成功。

本文主要是总结在这种机器不符合条件的非标环境下如何部署 OceanBase 产品。

OceanBase 产品的部署通常有两大步。一是部署 OCP 产品，二才是部署 OceanBase 集群。OCP 是 OceanBase 自动化运维平台，也叫 OceanBase 云平台。注意只是像云，实际跟云没有关系。官网文档里，OCP 对机器内存要求是 128G 以上。实际测试 64G 内存也可以勉强部署 OCP，后期使用问题会比较多。96G 内存会好一些。

不过 OCP 并不是必需的。也可以绕开 OCP 直接手动部署 OceanBase 集群。手动部署的好处就是可以调整目录、定制参数以解决非标环境的问题。现实中的非标环境问题和非标操作问题还是非常多的。

目前手动部署，对 OB 机器内存要求是 1.4 版本的 OB 要求至少 16G 内存，2.2 版本的 OB 要求至少 64G 内存。

OceanBase 集群生产环境默认是三台机器起步，通常都是三副本，机器分为三个区域，每个区域的机器保持对等（建议）。所以独立部署的时候也建议部署三节点的 OceanBase 集群。但是 OceanBase 的单副本也是可以跑的，也可以扩容和缩容。

预计 OceanBase 后期还会发布一个版本能够在 8G 内存的笔记本上运行，部署会采用 Docker 技术，到时候只需要一个启动镜像即可，对于初学者会非常方便。本文暂时还不会讨论这个 Docker 版本的部署方法。

## 2. 环境准备

### 2.1. 软件包说明

OceanBase 集群架构在形态上是比较清爽的，每个节点只有一个 observer 进程，再加上一个分布式访问代理 obproxy 进程（可以在 observer 节点上，但不是必须在）。这两个进程只需要 2 个软件包。

OceanBase 运维平台的进程会比较多，在每个 OceanBase 产品节点上都有一个 obagentd 进程，服务端是一个 OCP 容器。OCP 还有自己的元数据库，是一个 1.4 版本的 OB 集群，每个节点以容器形式存在。OCP 会有自己的专用 obproxy，以容器形式存在。三个容器的镜像文件需要分别提供，镜像文件格式都是 \*.tar.gz。

部署 ocp、obproxy 和 ob 容器是通过一套自动化脚本运行的。官网试用版下载的文件解压缩后看到目录 oceanbase\_trial 就是这个自动化部署产品，客户生产环境这套脚本是通过安装软件 t-oceanbase-antman 部署的，默认会安装在 /root/t-oceanbase-antman/ 目录下。

OceanBase 实例兼容 ORACLE 或 MySQL，如果是 ORACLE 实例，Java 程序连接需要使用 OceanBase 提供的 Java 驱动文件（oceanbase-client-\*.jar）。如果要在命令行下访问 ORACLE 实例，还需要安装客户端 obclient。

#### 2.1.1. 汇总说明

所有软件包汇总说明如下：

文件名	版本	备注
t-oceanbase-antman-*.rpm	1.0.16	部署 OCP 容器用，默认安装目录 /root/t-oceanbase-antman/。
ocp-all-in-one-EI61488790_20190912.tar.gz	2.3.5	OCP 镜像文件，放在 /root/t-oceanbase-antman/ 下。 文件名不重要，关键是镜像的 repository 和 tag。 reg.docker.alibaba-inc.com/antman/ocp-all-in-one:20200321205957554-EI61488790_20190912.f23f15c8
OBP156_20191227_1705.tar.gz	1.5.6	OBProxy 镜像文件，只给 OCP 用，放在 /root/t-oceanbase-antman/ 下。 文件名不重要，关键是镜像的 repository 和 tag。 reg.docker.alibaba-inc.com/antman/obproxy:OBP156_20191227_1705
OB1478_20200113_1054.tar.gz	1.4.7	OB 数据库镜像文件，1.4 版本，只给 OCP 做元数据库用，放在 /root/t-oceanbase-antman/ 下。



		文件名不重要，关键是镜像的 repository 和 tag。 reg.docker.alibaba-inc.com/antman/ob-docker:OB1478_20200113_1054
t-oceanbase-obagent-*.rpm	3.11.52	OCP 在各个节点上的agent，用于获取节点状态和性能信息。
oceanbase-*.rpm	2.2.3	OceanBase 软件 RPM 文件，业务 OB 集群部署时用。
obproxy-*.rpm	1.7.1	OBProxy 软件 RPM 文件，业务 OB 集群连接时用。
obclient-*.rpm	1.1.6	命令行客户端 obclient 的 RPM 文件，连接 ORACLE 实例必需有。
oceanbase-client-*.jar guava-18.0.jar	1.1.0	OceanBase Java 连接驱动，支持 ORACLE 和 MySQL 实例。
<a href="#">oceanbase_developer_center</a>	2.0.1	OceanBase Developer Center，OceanBase 团队提供的一个面向开发的图形化客户端工具。

### 2.1.2. OCP 部署脚本说明

正式版部署脚本的 Home 目录是 /root/t-oceanbase-antman/。如果是官网下载的试用版，Home 目录是 /root/oceanbase\_trail。

脚本名	备注
clonescripts/clone.conf	机器环境初始化脚本的配置文件，配置机器角色、磁盘分区、文件系统等。
clonescripts/clone.sh	机器环境自动初始化脚本。
obcluster.conf	OCP 自动化部署脚本 install.sh 的配置文件。
install.sh	OCP 自动化部署脚本。

## 2.2. 标准化环境的说明

环境标准化对产品对运维都很重要。OceanBase 产品的特点对环境很多地方都有显式或隐式的要求，这些在文档里可能没有提示，是部署不顺利的主要原因。

### 2.2.1. 汇总说明

OceanBase 数据库主机生产环境（标准化环境）应当满足下面这些条件。

类别	要求
硬件	CPU 不少于 16 物理核。
	主机内存不少于 256G。
	网卡做 Bonding，速率不低于 10000MB。
	OS 磁盘、数据盘、日志盘分别使用独立的物理盘或者逻辑盘。

软件	节点之间网络延时小于 50ms。
	节点之间时间同步误差小于 3ms。
	节点有唯一的主机名，并能本地解析出 IP。
	内核参数有针对网络、并发、共享内存等模块做优化。
	关闭防火墙服务、关闭 SELinux 特性。
	有用户 admin, uid:500, gid:500。
	用户会话针对可用内存、文件句柄数等做提升。
	core dump 文件大小为 unlimited 。
	有 yum 源。
	安装软件开发的基础的包。包括但不限于 python、gcc、mysql等。
	有 NTPD 服务，所有节点从同一 NTP 源或者有关联的 NTP 源同步时间。
	关闭 NUMA 特性
	关闭 CPU 节能模式；关闭网卡自动软中断服务，手动绑定网卡软中断。
	IO 调度算法为 deadline 。
规范	目录 /data/1 /data/log1 使用独立的文件系统，目录 owner 为 admin 。
	/data/log1 可用空间大小为内存大小的 2 到 4 倍。
	/data/1 可用空间尽可能的大。
	/home/admin 可用空间建议不少于 100G 。
	observer 和 obproxy 必须安装在 admin 用户下，必须在软件自身 home 目录下启动。
	observer 监听端口是 2881 和 2882，obproxy 监听端口默认是 2883 ，可以自定义为 3306 或 1521 。

### 2.2.2. OCP 部署后形态说明

OCP 里各个组件都是以 Docker 容器形态运行，容器说明如下。

容器名	备注
dns	DNS 容器，主备架构部署在 2 个节点上。 生产环境 OCP 是三节点部署，需要 DNS 提供域名给使用者。
OB_ZONE[1-3]	OB 容器，OCP 的元数据库，三节点部署。也可以是单节点单副本部署。
obproxy[1-3]	OBProxy 容器，无状态，每个节点部署一个。
ocp[1-3]	OCP 容器，无状态，每个节点部署一个。

### 2.2.3. OBSERVER 进程相关目录

observer 进程运行在 admin 用户下，相关目录的所有者必须是 admin 用户。OCP 容器里的软件也运行在 admin 用户下，相关目录是从宿主机上目录映射过去的，都是 admin 用户，且 uid 和 gid 必须都是 500；否则，docker 容器映射目录后，会有权限不对问题。

下面是各个目录说明，注意目录中的软链接关系，以及运行日志 log 目录位置。

observer 进程 home 目录是 `/home/admin/oceanbase`，运行日志目录是 `/home/admin/oceanbase/log/`。

observer 进程的数据和事务日志总目录分别是 `/data/1` 和 `/data/log1`。

```
[admin@h1 ~]$ cd /home/admin/oceanbase/
$tree /home/admin/oceanbase/store
/home/admin/oceanbase/store
├── obdemo
│   ├── clog -> /data/log1/obdemo/clog
│   ├── etc2 -> /data/log1/obdemo/etc2
│   ├── etc3 -> /data/1/obdemo/etc3
│   ├── ilog -> /data/log1/obdemo/ilog
│   ├── oob_clog -> /data/log1/obdemo/oob_clog
│   ├── slog -> /data/log1/obdemo/slog
│   ├── sort_dir -> /data/1/obdemo/sort_dir
│   └── sstable -> /data/1/obdemo/sstable
└── 9 directories, 0 files

[admin@h1 ~]$ cd /home/admin/oceanbase/
$tree /data/
/data/
├── 1
│   ├── obdemo
│   │   ├── etc3
│   │   ├── sort_dir
│   │   └── sstable
│   └── log1
│       ├── obdemo
│       │   ├── clog
│       │   ├── etc2
│       │   ├── ilog
│       │   ├── oob_clog
│       │   └── slog
└── 12 directories, 0 files
```

## 2.2.4. OBPROXY 进程相关目录

obproxy 进程 home 目录是 `/opt/taobao/install/obproxy` 这是一个软链接，实际指向相应的 obproxy 版本软件目录。

```

[admin@obproxy ~]$ cd /home/admin/oceanbase/
$tree /opt/taobao/install/ -L 1
/opt/taobao/install/
├── ajdk-8.0.0-b60
├── obproxy -> obproxy-1.5.7
├── obproxy-1.4.1
└── obproxy-1.5.7

4 directories, 0 files

[admin@obproxy ~]$ cd /home/admin/oceanbase/
$tree /opt/taobao/install/obproxy
/opt/taobao/install/obproxy
├── bin
│   ├── obproxy
│   ├── obproxycd.sh
│   ├── obp_xflush.py
│   └── unzip.py
├── etc
│   ├── obproxy_config.bin
│   └── obproxy_config.bin.old
├── log -> /home/admin/logs/obproxy/log
├── minidump -> /home/admin/logs/obproxy/minidump
└── tools
    ├── dump_syms
    ├── minidump.sh
    ├── minidump_stackwalk
    └── obproxy.sym

5 directories, 10 files

```

obproxy 进程的运行日志目录通过软链接指向 `/home/admin/logs/obproxy/log`。

## 2.3. 机器初始化 I (手动做)

有些初始化操作自动化脚本不能做，还需要手动做。这些要求并不是 OceanBase 特有的，一般客户生产机房服务器的标准模板都有要求。不过测试环境的服务器或者虚拟机可能就不遵守规范了，所以需要单独描述。

### 2.3.1. 主机名标准化

#### 1. 修改 `/etc/hosts`

```

# vi /etc/hosts
192.168.10.5 observer1

```

#### 2. 修改 `/etc/sysconfig/network`

```

# vi /etc/sysconfig/network
# 增加
HOSTNAME=observer1

```

#### 3. 修改 `/etc/hostname`

```

# vi /etc/hostname
observer1

```

#### 4. hostname 修改当前会话主机名

```
# hostname observer1
```

#### 5. 检查主机名，能反查出正确的 IP 即可。

```
# hostname -i
192.168.10.5
```

### 2.3.2. 配置时间同步服务

OceanBase 相关产品之间，在时间上都有一些隐含的联系，因此要求 OceanBase 产品的所有服务器的物理时间要保持同步，误差尽量控制在 10ms 以内，甚至更小。

通常使用 linux 自带的 NTP 服务可以同步时间。如果当前机器环境有稳定可靠的 NTP 服务器，则选它作为所有服务器的 NTP 源。如果没有，则选固定的一台服务器，把它作为 NTP 源。

#### 1. 安装 ntp 相关软件

```
# yum -y install ntp ntpdate
```

#### 2. 配置 ntp.conf

```
# vim /etc/ntp.conf
restrict default ignore
restrict 127.0.0.1
restrict 192.168.0.0 mask 255.255.0.0

driftfile /var/lib/ntp/drift
pidfile /var/run/ntpd.pid
logfile /var/log/ntp.log

# local clock
server 127.127.1.0
fudge 127.127.1.0 stratum 10

server 192.168.1.100 iburst minpoll 4 maxpoll 6
```

说明：

- restrict 用于指定 IP 的相关时间同步命令权限。

```
restrict [IP] mask [netmask_IP] [parameter]
```

parameter 包括 ignore、nomodify、noquery、notrap 和 notrust 等。ignore 默认指拒绝所有类型的 NTP 同步。

- server 用于指定上层 NTP 源服务器。

```
server [IP or hostname] [prefer]
```

如果没有上层 NTP 源服务器，可以设置为 127.127.1.0，即本机作为 NTP 源服务器。

### 3. 重启 NTP 同步服务

```
# systemctl restart ntpd
# systemctl status ntpd
• ntpd.service - Network Time Service
   Loaded: loaded (/usr/lib/systemd/system/ntpd.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2020-04-18 14:43:31 CST; 10s ago
   Process: 95939 ExecStart=/usr/sbin/ntpd -u ntp:ntp $OPTIONS (code=exited, status=0/SUCCESS)
   Main PID: 95944 (ntpd)
   Memory: 544.0K
   CGroup: /system.slice/ntpd.service
           └─95944 /usr/sbin/ntpd -u ntp:ntp -g
.....
```

### 4. 检查 NTP 同步状态

命令 `ntpstat`，如果 ntp 服务刚启动时，这个是没有结果，需要等一会。

```
# ntpstat
unsynchronised
time server re-starting
polling server every 64 s
```

此后再跑

```
# ntpstat
synchronised to NTP server (192.168.1.100) at stratum 3
time correct to within 8 ms
polling server every 64 s
```

命令 `ntpq`

remote	refid	st	t	when	poll	reach	delay	offset	jitter
LOCAL(0)	.LOCL.	10	l	589	64	0	0.000	0.000	0.000
*192.168.1.100	10.10.10.1	2	u	18	64	377	1.591	0.249	0.054

- remote：表示使用的 ntp 服务器，\* 表示目前选择的 NTP 服务器。LOCAL 表示本机。x 表示已不再使用，- 表示已不再使用，+ 良好的优先考虑的，# 良好的但未使用的。
- refid：远程 NTP 服务器使用的更高级 NTP 服务器。INIT 表示还在获取。
- st：远程 NTP 服务器的 stratum（级别）。
- when：最后一次同步到现在的时间（默认为秒，h 表示小时，d 表示天）。
- poll：同步的频率，单位秒。
- delay：从本机到远程 NTP 服务器的往返时间（单位毫秒）。
- offset：本机与远程 NTP 服务器的时间偏移量（单位毫秒）。
- jitter：本机与远程 NTP 服务器的时间偏移量平均偏差（单位毫秒）。

### 5. 最终检查方法

最终同步延时检查以下面为准：

```
# clockdiff 192.168.1.100
.....
host=192.168.1.100 rtt=1(0)ms/1ms delta=0ms/0ms Sat Apr 18 14:41:40 2020
```

## 6. 手动同步方法（可选）

如果时间总是有很大误差，还有个办法就是使用 `ntpdate` 命令手动订正时间。这个跟自动同步服务是冲突的，需要先停掉 NTP 服务。

```
# systemctl stop ntpd
# ntpdate -u 192.168.1.100
18 Apr 14:54:20 ntpdate[108001]: adjust time server 192.168.1.100 offset -0.000180 sec
```

如果这个方法有效，就配置到系统的 `crontab` 下。

```
# crontab -e
* * * * * /sbin/ntpdate -u 192.168.1.100 2>&1 1>>/tmp/ntpupdate.log
```

## 7. 防火墙规则（可选）

标准化的环境是要关闭防火墙的，如果客户的环境非要开启防火墙，那可以参照下面规则放行。

```
# iptables -A INPUT -p udp -i eth0 -s 192.168.1.0/16 -dport 123 -j ACCEPT
# systemctl restart ntpd
```

## 2.3.3. 配置好 Yum 源。

请参照 `linux` 或 `centos` 方法配置 `yum` 源。如果没有可用的外部 `yum` 源，就本地利用光盘文件做一个。具体方法请网上搜索。

## 2.4. 机器初始化 II（自动做）

注意：

如果有官网试用版文件 `ocean-base-trail.tar.gz` 或者 `t-oceanbase-antman.rpm` 包，则解压缩或安装后会有目录 `/root/oceanbase-trail` 或 `/root/t-oceanbase-antman` 目录等，则执行这一步骤。否则，不执行这一步。

### 2.4.1. 修改配置文件 clone.conf

进入到目录 `/root/oceanbase-trail/clonescripts` 或 `/root/t-oceanbase-antman/clonescripts` 下。这个目录主要是做机器自动化初始化的。

配置文件：`clone.conf`

#### 1. 指定机器角色

```
## machine role, value could be ob, ocp or obbackup
machineRole=ocp
```

- `ocp` : 这是OCP机器，会初始化`/docker`、`/data/1`、`/data/log1`、`/home`（可选）目录。
- `ob` : 这是OB机器，不会初始化`/docker`目录，会初始化`/data/1`、`/data/log1`、`/home`（可选）目录。
- `obbackup` : 这是备份机器，备份以后单独介绍。

#### 2. 磁盘分区配置(可选)

分区配置主要是为了针对不同机型创建上面相应的目录。客户机型千差万别，这里的配置和方法不一定适用所有的机型。客户可以针对本机磁盘特点自行分区创建相关目录。

```
## part disk mode, value either lvm or parted
diskPartMode=lvm
```

设置分区方式，目前有两种：

- `lvm` : 使用`lvm`相关命令，将多盘聚合到一个VG里，然后划分出多个LV，包含数据目录、日志目录、HOME目录（可选，由参数决定）、Docker目录（可选，由参数决定）。建议用`lvm`。
- `parted` : 使用`parted`命令，对一个大盘做划分，分出多个裸设备用于数据目录、日志目录、HOME目录（可选，由参数决定）、Docker目录（可选，由参数决定）。

关于目录部分，`lvm`和`parted`都分别有对应的配置文件部分定义各个目录的大小。

这里有几个注意点：

- 日志目录主要是存放事务日志，大小建议内存大小的3到4倍，尤其是要做性能测试的前提下。如果机器资源不好，只是看看OB的功能，这个日志目录大小可以是内存的2倍左右。
- 数据目录和日志目录强烈建议是不同的LV或者裸设备，即不能共用同一个文件系统。如果条件有限机器只有一块大盘并且还被操作系统用了，那就不能执行磁盘分区命令，而是需要手动创建相关目录。



通常建议使用lvm分区方便些。

```
##### lvm settings begin #####
## could be partition or device name
devArray=(sdb sdc sdd sde sdf sdg sdh sdi sdj sdk sdl)
```

这是设置多盘的正确方式。这些值取自命令pvs里列出来未使用的PV。至少要有一个独立的未使用的 PV 。

### 3. 操作系统类型

```
## os, option support centos, alios and redhat
os=alios
```

操作系统类型，外部客户机型一般选择centos或redhat。

这个主要是为了安装软件包时选择对应操作系统类型的软件包用。不过安装脚本并没有包含全部的软件包文件，还需要依赖客户机的 yum install命令。

所以建议客户机器所有机器都配置好YUM源。如果没有远程的可以做一个本地的 YUM 源指向光盘镜像文件。

### 4. Home目录

OCP、OB以及备份软件等都会安装在admin用户下，使用admin的默认目录 /home/admin。尤其是OB的运行日志(observer.log 和 rootservice.log 等)都会在 /home/admin 目录下。所以 /home 分区的空间建议在100G以上。测试环境可以小一些，但是如果很少的话，要时刻留意空间耗尽。OB的运行日志默认日志级别都是INFO级别，日志增长非常快。

```
## if make home disk, value could be yes or no. If yes selected, the script will make home disk, if no selected, the script
will not make home disk, in which case you already have home disk made and donot want re-make it again
makeHomeDisk=no
```

如果本机的/home分区空间够大，那就不需要初始化/home目录；否则这里设置为自动初始化/home目录。此前/home/目录的数据会备份到/tmp/下，但是并不会恢复回来。这个要注意！

## 2.4.2. 安装 RPM 包

先确保本机的yum源是可用的。

```
[root@xxx /root/oceanbase_trial/clonescripts]
#yum list
```

执行初始化脚本

```
[root@xxx /root/oceanbase_trial/clonescripts]
#./clone.sh -m
```

### 2.4.3. 修改相关参数

这一步必须做，主要包括：

- ▣ 修改内核参数 `/etc/sysctl.conf`。
- ▣ 修改会话参数 `/etc/security/limits.d/oceanbase_limits.conf`。
- ▣ 关闭 NUMA。
- ▣ 修改网卡软中断。
- ▣ 调整 CPU 主频为性能模式。
- ▣ 修改 IO 调度算法。
- ▣ 禁用 SELinux 特性。
- ▣ 关闭防火墙。
- ▣ 其他。

```
[root@xxx /root/oceanbase_trial/clonescripts]
#./clone.sh -c
```

### 2.4.4. 目录准备

本步骤的目的是为了创建 2 个必选目录 `/data/1` 和 `/data/log1` 和 `/home/admin`（可选）、`/docker`（可选）

**注意：**前面配置文件里要配置正确，否则可能破坏已有数据。

**注意：**如果机器没有给出独立的未使用的盘，那就不要执行这一步了。改为后面手动创建相关目录。

```
[root@xxx /root/oceanbase_trial/clonescripts]
#./clone.sh -p
```

这个步骤不支持重跑。如果运行出错后，可能是一个中间状态，这个时候需要手动删除相关裸设备或者 `Lv/VG/PV` 等。

下面是初始化后的一个示例。

```
[root@xxx /root/oceanbase_trial/clonescripts]
WARNING: Not using lvm2 with older version.
PV          VG      Fmt  Attr PSize  PFree
/dev/sdb    ob_vg  lvm2 a--  447.13g  0
/dev/sdc    ob_vg  lvm2 a--  447.13g  0
/dev/sdd    ob_vg  lvm2 a--  447.13g  0
/dev/sde    ob_vg  lvm2 a--  447.13g  0
/dev/sdf    ob_vg  lvm2 a--  447.13g  0
/dev/sdg    ob_vg  lvm2 a--  447.13g  0
/dev/sdh    ob_vg  lvm2 a--  447.13g  0
/dev/sdi    ob_vg  lvm2 a--  447.13g  0
/dev/sdj    ob_vg  lvm2 a--  447.13g  0
/dev/sdk    ob_vg  lvm2 a--  447.13g  0
/dev/sdl    ob_vg  lvm2 a--  447.13g  0

[root@xxx /root/oceanbase_trial/clonescripts]
#df -h
Filesystem                Size      Used Avail Use% Mounted on
/dev/sda3                  49G        21G   26G   45% /
devtmpfs                   63G         0    63G    0% /dev
tmpfs                      63G        66M    63G    1% /dev/shm
tmpfs                      63G       880K    63G    1% /run
tmpfs                      63G         0    63G    0% /sys/fs/cgroup
/dev/sda5                  387G       19G   349G    6% /home
/dev/sda2                  976M      130M   780M   15% /boot
/dev/mapper/ob_vg-docker_home 492G     2.7G   464G    1% /docker
/dev/mapper/ob_vg-ob_log    1007G     77M   956G    1% /data/log1
/dev/mapper/ob_vg-ob_data   3.3T      89M   3.1T    1% /OceanBase技术闲谈
```

- 手动创建相关目录方法（可选）：

```
# mkdir -p /data/1 /data/log1
# mkdir -p /docker
# chown -R admin.admin /data/
```

## 2.4.5. 新建用户

新建用户 `admin`，并且 `uid` 设置为500（这是阿里内部习惯）。后面 `docker` 内外目录映射的时候没有做 `uid` 映射转换，所以默认要求内外目录的 `owner` 都是 `admin` 并且 `uid` 是 500。否则，可能面临目录权限问题。

客户机如果已经存在 `admin` 用户，会被删除掉。如果你不想删除现有的 `admin` 用户，那就不要执行这一步。而是将 `admin` 用户的 `uid` 和 `gid` 都修改为 500。

```
[root@xxx /root/oceanbase_trial/clonescripts]
#./clone.sh -u
```

## 2.4.6. 安装 docker 软件(可选)

这一步安装 `docker` 相关软件包并启动 `docker` 服务，默认会使用目录 `/docker`。所以在这一步之前要确保磁盘已经分区并且 `/docker` 目录空间足够。

```
[root@xxx /root/oceanbase_trial/clonescripts]
```

```
#!/clone.sh -i
```

## 检查安装是否正确用命令

```
[root@xxx /root/oceanbase_trial/clonescripts]
# systemctl status docker
docker.service - Docker Application Container Engine
Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
Active: active (running) since Sat 2020-01-18 15:54:48 CST; 2 weeks 2 days ago
    Docs: http://docs.docker.com
    Main PID: 13771 (dockerd)
    Memory: 2.4G
    CGroup: /system.slice/docker.service
            └─13771 /usr/bin/dockerd -H tcp://0.0.0.0:4243 -H unix:///var/run/docker.sock --selinux-enabled=false --log-opt ma
x-size=1g --graph=/docker
            └─13791 docker-containerd --config /var/run/docker/containerd/containerd.toml

Jan 31 11:22:36 xxx dockerd[13771]: time="2020-01-31T11:22:36+08:00" level=info msg="shim docker-containerd-shim started"
address="/containerd-shim/moby/28379e1464373c7249a7...ks" pid=71070
Jan 31 11:26:19 xxx dockerd[13771]: time="2020-01-31T11:26:19+08:00" level=error msg="stat cgroup 28379e1464373c7249a76
c03162a7dd6e6d285618fd98346180e57dbfc271621" error=""/...ave 4 fields"
Jan 31 11:26:19 xxx dockerd[13771]: time="2020-01-31T11:26:19+08:00" level=info msg="shim reaped" id=28379e1464373c7249
a76c03162a7dd6e6d285618fd98346180e57dbfc271621 module=...ainerd/tasks"
Jan 31 11:26:19 xxx dockerd[13771]: time="2020-01-31T11:26:19.780741127+08:00" level=info msg="ignoring event" module=lib
containerd namespace=moby topic=/tasks/delete type="...s.TaskDelete"
Jan 31 11:26:20 xxx dockerd[13771]: time="2020-01-31T11:26:20+08:00" level=error msg="stat cgroup fb959a57982a6312804c0f
f765f345d8cf6f007b3d6d222d30eaa0573dceb44f" error=""/...ave 4 fields"
Jan 31 11:26:20 xxx dockerd[13771]: time="2020-01-31T11:26:20+08:00" level=info msg="shim reaped" id=fb959a57982a631280
4c0ff765f345d8cf6f007b3d6d222d30eaa0573dceb44f module=...ainerd/tasks"
Jan 31 11:26:20 xxx dockerd[13771]: time="2020-01-31T11:26:20.997790335+08:00" level=info msg="ignoring event" module=lib
containerd namespace=moby topic=/tasks/delete type="...s.TaskDelete"
Jan 31 11:26:25 xxx dockerd[13771]: time="2020-01-31T11:26:25+08:00" level=error msg="stat cgroup 175364fb736988d24c052
61497b9d4160188cd8d53e5b2f1b695e79cd4ac5d9c" error=""/...ave 4 fields"
Jan 31 11:26:25 xxx dockerd[13771]: time="2020-01-31T11:26:25+08:00" level=info msg="shim reaped" id=175364fb736988d24c
05261497b9d4160188cd8d53e5b2f1b695e79cd4ac5d9c module=...ainerd/tasks"
Jan 31 11:26:25 xxx dockerd[13771]: time="2020-01-31T11:26:25.837715440+08:00" level=info msg="ignoring event" module=lib
containerd namespace=moby topic=/tasks/delete type="...s.TaskDelete"
Hint: Some lines were ellipsized, use -l to show in full.

[root@xxx /root/oceanbase_trial/clonescripts]
# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
NAMES
```

## 2.4.7. 最终检查

在安装的最后一步，使用 `./clone.sh -t` 会检查前面的配置操作是否符合要求并给出建议。此外OB机器之间的网络延时检查、时间同步延时检查需要额外检查。

```
# ./clone.sh -t
```

相信测试环境的机器在这一步会有很多不满足的地方。这个时候需要评估其影响。OceanBase 对机器的初始化配置或检查标准有很多是所有数据库通用的，不难看懂。只有特别的几个项目是 OceanBase 有关的，必须满足。

如果目录空间、时间同步、基本的 RPM 包不满足，后面部署 oceanBase 或使用 OceanBase 的时候，难免会遇到各种奇怪的问题，那时候排查原因的成本会非常高。

## 2.5. 机器初始化 II（手动做，可选）

注意：

如果“[机器初始化II（自动做）](#)”不能执行，则手动执行本章节步骤。

### 2.5.1. 安装 RPM 包

根据操作系统的不同，安装相应的依赖包。前提是主机已经配置好软件包安装源（网络的或本机的）。

#### □ CentOS

```
# yum install -y expect mariadb mariadb-devel python-devel openssl-devel gcc gcc-gfortran gcc-c++ python-setuptools bc net-tools mtr ntp bind-utils libaio
```

#### □ Suse

```
zypper in -y expect mariadb mariadb-devel python-devel openssl-devel gcc python-setuptools bc net-tools mtr ntp bind-utils libaio
```

**注意：**安装 mysql 命令。

如果主机上有 mysql 8.0 及其以后版本的客户端，请安装 5.5/5.6/5.7 版本的 mysql 客户端，并通过修改 PATH 环境变量让新安装的 mysql 客户端生效。

### 2.5.2. 修改相关参数

参数包括内核参数、会话参数、防火墙设置、CPU、IO 和网卡中断相关设置。

#### □ 内核参数

```
#####
# for oceanbase
net.core.somaxconn=32768
net.core.netdev_max_backlog=10000
net.core.rmem_default=16777216
net.core.wmem_default=16777216
net.core.rmem_max=16777216
net.core.wmem_max=16777216

net.ipv4.ip_local_port_range=10000 65535
net.ipv4.ip_forward=1
net.ipv4.conf.default.rp_filter=1
net.ipv4.conf.default.accept_source_route=0
net.ipv4.tcp_rmem=4096 87380 16777216
net.ipv4.tcp_wmem=4096 65536 16777216
net.ipv4.tcp_max_syn_backlog=16384
net.ipv4.tcp_fin_timeout=15
net.ipv4.tcp_tw_reuse=1
net.ipv4.tcp_slow_start_after_idle=0
```

```
vm.swappiness=0
kernel.core_pattern=/data/1/core-%e-%p-%t
vm.min_free_kbytes=2097152
fs.aio-max-nr=1048576
vm.max_map_count=655360
```

#### □ 会话参数

```
vi /etc/security/limits.conf
* soft nofile 655360
* hard nofile 655360
* soft nproc 655360
* hard nproc 655360
* hard core unlimited
* soft core unlimited
* hard stack 10240
* soft stack 10240
* hard cpu unlimited
* soft cpu unlimited
```

会话参数修改后，需要重新登录才会生效。确认命令如下：

```
#ulimit -a
core file size          (blocks, -c) unlimited
data seg size           (kbytes, -d) unlimited
scheduling priority     (-e) 0
file size               (blocks, -f) unlimited
pending signals         (-i) 514881
max locked memory       (kbytes, -l) 64
max memory size         (kbytes, -m) unlimited
open files              (-n) 1048576
pipe size               (512 bytes, -p) 8
POSIX message queues    (bytes, -q) 819200
real-time priority      (-r) 0
stack size              (kbytes, -s) 10240
cpu time                (seconds, -t) unlimited
max user processes      (-u) 1048576
virtual memory          (kbytes, -v) unlimited
file locks              (-x) unlimited
```

#### □ 关闭防火墙

```
# systemctl disable firewalld
# systemctl stop firewalld
# systemctl disable iptables
# systemctl stop iptables
```

#### □ 调整 CPU 主频

```
# vim set_cpufreq.sh

#!/bin/bash

performance_cpu_num=`cat /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor | grep 'performance' | wc -l`
cpu_num=`cat /proc/cpuinfo | grep ^processor | wc -l`
if [[ performance_cpu_num -ne cpu_num ]]; then
    cpupower frequency-set -g performance
    if [[ $? -ne 0 ]]; then
        cpu_num=`cat /proc/cpuinfo | grep ^processor | wc -l`
        let cpu_num=cpu_num-1
        for i in $(seq 0 $cpu_num); do
            echo 'performance' > /sys/devices/system/cpu/cpu${i}/cpufreq/scaling_governor
```

```

done
fi
fi

```

执行上面 shell 脚本。

- 设置 IO 调度算法（可选，性能相关）

```

# vim set_deadline.sh
for DISK in `ls /sys/block | grep "sd\?"` ; do echo deadline > /sys/block/$DISK/queue/scheduler ; done

```

- 设置CPU 软中断（可选，性能相关）

```

# vim set_irq_affinity.sh

#!/bin/bash

irq_list=`cat /proc/interrupts | awk '{if ($0 ~ /eth/ && $2 > 1) {print $1}}' | awk -F ":" '{print $1}'|shuf`
core_num=4
#`cat /proc/cpuinfo | grep "processor" -c`

n=0
b=0
p=1
for i in $irq_list
do
    let "cpu_mask=1<<$b"
    let "cpu_mask=$cpu_mask*$p"
    let "n=$n+1"
    let "t=$n%4"
    if [ 0 -eq $t ]
    then
        let "b=0"
        let "p=$p*10"
    else
        let "b=$b+1"
    fi
    pv=`cat /proc/irq/$i/smp_affinity`
    echo -en "Set IRQ\033[31m[$i]\033[0m's mask from \033[32m[$pv]\033[0m to \033[33m[$cpu_mask]\033[0m"
    echo -e " by run \"echo $cpu_mask >/proc/irq/$i/smp_affinity\""
    echo $cpu_mask >/proc/irq/$i/smp_affinity
    if [ $n -ge $core_num ]
    then
        let "n=0"
        let "b=0"
        let "p=1"
    fi
done

```

- 设置网卡软中断（可选，性能相关）

网卡软中断的方法请在网上搜索一下。非必须步骤。

### 2.5.3. 目录准备

本步骤的目的是为了创建 2 个必选目录 `/data/1` 和 `/data/log1`。

如果主机有提供一块空间足够的独立的未使用的裸盘，则可以用 `fdisk` 或 `parted` 对该磁盘进行物理分区，得到 2 个裸设备。

#### ▣ 磁盘分区 (`fdisk`)

```
# fdisk /dev/sdb

p 查看当前物理分区
n 新建物理分区，选 primary。大小会提示输入起始位置（sector 位置）和结束位置（可以指定大小，如 +100G 等）
w 保存物理分区
q 退出
```

#### ▣ 建文件系统

```
# mkfs.ext4 /dev/sdb1
# mkfs.ext4 /dev/sdb2
# mkdir -p /data/1 /data/log1
# vi /etc/fstab
/dev/sdb1    /data/log1    ext4    defaults,noatime,nodiratime,nodelalloc,barrier=0    0    0
/dev/sdb2    /data/1       ext4    defaults,noatime,nodiratime,nodelalloc,barrier=0    0    0

# mount -t /data/1
# mount -t /data/log1
# df -h |grep data

# chown -R admin.admin /data/
```

如果主机没有提供独立的未使用裸盘，那只能复用当前已经分配的文件系统。如果当前可用空间是在根分区（/）下，则直接建相应目录。

```
# mkdir -p /data/1 /data/log1
# chown -R admin.admin /data/
```

如果当前可用空间是在另外一个文件系统，比如说 `/u01` 下。则使用软链接技术。

```
# mkdir -p /data /u01/ob/1 /u01/ob/log1
# ln -s /u01/ob/1 /data/1
# ln -s /u01/ob/log1 /data/log1
# chown -R admin.admin /data/1 /data/log1
```

## 2.5.4. 新建用户

如果 `admin` 用户不存在，就新建用户组和用户。

```
# groupadd -g 500 admin
# useradd -g 500 -u 500 admin -d /home/admin
```

如果 `/home` 可用空间很小，则需要指定其他空间大的盘作为 `admin` 用户的根目录。



### 3. 部署 OCP(可选)

**注意：**

如果有 128G 内存以上的机器，则可以部署 ocp；否则，就不要部署 ocp 了。

#### 3.1.OCP 安装配置文件

OCP安装配置文件是obcluster.conf在解压缩后的目录下。  
配置文件需要根据客户机器信息做少量修改。

##### 3.1.1. OCP 运行模式

```
##
SINGLE_OCP_MODE=TRUE
##### 根据环境必须修改 / MUST CHANGE ACCORDING ENVIRONMENT #####
##### 填写机器IP和root/admin密码 / Edit Machine IP and Password Of root/admin #####
ZONE1_RS_IP=11.xxx.xxx.5
OBSERVER01_ROOTPASS=root087005
OBSERVER01_ADMINPASS=admin087005
```

SINGLE\_OCP\_MODE 指定是否单节点运行。如果不是，那就默认是三节点集群形式运行。ZONE1\_RS\_IP 是单节点IP信息，后面 OBSERVER01 是这个节点上的 root 和 admin 用户的密码。安装完后可以再改密码。

如果是三节点运行，这里配置就要指定三组这样的信息，以1，2，3区分。官网下载的试用版只支持单节点运行模式，这个对学习和POC验证也足够了。

##### 3.1.2. 容器资源

```
#### 根据服务器CPU、内存设置容器资源编排 / Allocate Container Resources According To Server ####
OB_docker_cpus=18
OB_docker_memory=106G
OCP_docker_cpus=8
OCP_docker_memory=16G
OBProxy_docker_cpus=4
OBProxy_docker_memory=6G
```

OCP由三个容器组成：OB容器、OCP容器和OBProxy容器。这里分别指定每个容器需要的CPU和内存。如果主机资源充足，就多分一点资源。主机资源少的情况下，只能调小这里容器的资源。

OB容器内存至少要保证32G，OCP容器内存不要少于16G，OBProxy容器也可以省略直接安装RPM包。

上面示例的配置的主机是32C128G。

### 3.1.3. 容器镜像信息

这里是要指定三个容器对应镜像文件的信息：文件名、REPO和TAG。

```
##### 填写OCP各组件容器的版本信息 / Edit Docker Image, Repo And Tag of OCP Components #####
# OB docker
docker_image_package=observer1478.tar.gz
OB_image_REPO=reg.docker.alibaba-inc.com/antman/ob-docker
OB_image_TAG=OB1478_20200102_2113
# OCP docker
ocp_docker_image_package=ocp233.tar.gz
OCP_image_REPO=reg.docker.alibaba-inc.com/antman/ocp-all-in-one
OCP_image_TAG=OCP233_20200102_2122
# OBPROXY docker
obproxy_docker_image_package=obproxy156.tar.gz
obproxy_image_REPO=reg.docker.alibaba-inc.com/antman/obproxy
obproxy_image_TAG=OBP156_20200102_2110
```

如果是官网下载的安装文件，这部份配置信息是不用修改的。  
但是如果后续更新了镜像文件，这里信息可能就需要修改。而要查看镜像文件的REPO和TAG信息，可以先将镜像加载( `docker load` )，然后查看镜像信息( `docker images` )。

```
[root@xxx /root/oceanbase_trial]
# ls *.gz
obproxy156.tar.gz  observer1478.tar.gz  ocp233.tar.gz

[root@xxx /root/oceanbase_trial]
# for img in `ls *.gz`;do echo $img ;docker load < $img; doneobproxy156.tar.gz
Loaded image: reg.docker.alibaba-inc.com/antman/obproxy:OBP156_20200102_2110
observer1478.tar.gz
Loaded image: reg.docker.alibaba-inc.com/antman/ob-docker:OB1478_20200102_2113
ocp233.tar.gz
Loaded image: reg.docker.alibaba-inc.com/antman/ocp-all-in-one:OCP233_20200102_2122

[root@xxx /root/oceanbase_trial]
# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
reg.docker.alibaba-inc.com/antman/ocp-all-in-one	OCP233_20200102_2122	967f985769d2	4 weeks ago	1.8 GB
reg.docker.alibaba-inc.com/antman/ob-docker	OB1478_20200102_2113	5351e5990e0f	4 weeks ago	780MB
reg.docker.alibaba-inc.com/antman/obproxy	OBP156_20200102_2110	c4076538faa1	4 weeks ago	282MB

后面其他配置就不用修改了。

## 3.2. 开始安装 OCP

### 3.2.1. 安装命令

安装会分为好几步，查看帮助文档可以看到步骤。

```
[root@xxxs /root/oceanbase_trial]
#./install.sh -h
```

```
Usage: ./install.sh [OPTIONS]
```

#### Options:

```
-h, --help                Print help and exit
-d, --debug               Print debug information
-V, --version             Print version
-i, --install-steps string For example 1,3-5,7-
-c, --clear-steps string  For example 1,3-5,7-
-f, --config-file string  Read in a config file
-l, --load-balance        Load balance mode
```

#### Steps:

1. ssh authorization
2. install load balancer
3. install ob server
4. init ocp metadb
5. install temp OCP
6. install obproxy
7. install OCP
8. POSTCHECK

通常顺序是先按照负载均衡容器（可选），然后是 OB 容器，初始化 ocp 元数据库，然后按照临时 ocp 容器、OBProxy 容器和正式的 OCP 容器。

注意，安装参数是 `-i`，清理参数是 `-c`。都可以连续安装或清理，或者单步安装或清理。

每个命令后都会有相应的日志文件供查看进度。

### 3.2.2. 自动连续安装

```
[root@xxx /root/oceanbase_trial]
#./install.sh -i 1-
run install.sh with DEBUG=FALSE, INSTALL_STEPS=1 2 3 4 5 6 7 8 CLEAR_STEPS= CONFIG_FILE=/root/oceanbase_trial/obcluster.conf
[2020-02-04 17:25:26.261595] INFO [start antman API service]
LB_MODE=none
[2020-02-04 17:25:26.353919] INFO [step1: making ssh authorization, logfile: /root/oceanbase_trial/logs/ssh_auth.log]
[2020-02-04 17:25:26.832011] INFO [step1: ssh authorization done]
[2020-02-04 17:25:26.847573] INFO [step2: no action need when LB_MODE=none]
[2020-02-04 17:25:26.850503] INFO [step3: check whether OBSERVER port 2881,2882 are in use or not on 11.xxx.xxx.5]
[2020-02-04 17:25:27.197266] INFO [step3: OBSERVER port 2881,2882 are idle on 11.xxx.xxx.5]
[2020-02-04 17:25:27.199657] INFO [step3: installing ob cluster, logfile: /root/oceanbase_trial/logs/install_ob.log]
[2020-02-04 17:30:26.007264] INFO [step3: installation of ob cluster done]
[2020-02-04 17:30:26.009985] INFO [step4: initializing ocp metadb, logfile: /root/oceanbase_trial/logs/init_metadb.log]
[2020-02-04 17:30:26.941684] INFO [step4: initialization of ocp metadb done]
[2020-02-04 17:30:26.944131] INFO [step5: check whether OCP port 8080 is in use or not on 11.xxx.xxx.5]
[2020-02-04 17:30:27.308464] INFO [step5: OCP port 8080 is idle on 11.xxx.xxx.5]
[2020-02-04 17:30:27.311137] INFO [step5: installing temporary ocp, logfile: /root/oceanbase_trial/logs/install_tmp_ocp.log]
[2020-02-04 17:34:36.886177] INFO [step5: installation of temporary ocp done]
[2020-02-04 17:34:36.889325] INFO [step6: check whether OBPROXY port 2883 is in use or not on 11.xxx.xxx.5]
[2020-02-04 17:34:37.084248] INFO [step6: OBPROXY port 2883 is idle on 11.xxx.xxx.5]
[2020-02-04 17:34:37.087163] INFO [step6: installing obproxy, logfile: /root/oceanbase_trial/logs/install_obproxy.log]
[2020-02-04 17:35:09.314143] INFO [step6: installation of obproxy done]
[2020-02-04 17:35:09.316600] INFO [step7: installing ocp, logfile: /root/oceanbase_trial/logs/install_ocp.log]
[2020-02-04 17:36:19.347975] INFO [step7: installation of ocp done]
[2020-02-04 17:36:19.350974] INFO [step8: post-checking service, logfile: /root/oceanbase_trial/logs/post_check_service.log]
[2020-02-04 17:36:27.431920] INFO [step8: post check done]
```

自动化连续清理示例：

```
[root@xxx /root/oceanbase_trial]
#./install.sh -c 1-
run install.sh with DEBUG=FALSE, INSTALL_STEPS= CLEAR_STEPS=8 7 6 5 4 3 2 1 CONFIG_FILE=/root/oceanbase_trial/obcluster.conf
[2020-02-04 17:24:41.127969] INFO [start antman API service]
LB_MODE=none
[2020-02-04 17:24:41.183515] INFO [clear_step8: no need to clear for step network post-check]
[2020-02-04 17:24:41.186271] INFO [clear_step7: uninstalling ocp and remove docker, logfile: /root/oceanbase_trial/logs/uninstall_ocp.log]
[2020-02-04 17:24:42.034660] INFO [clear_step7: uninstallation of ocp done]
[2020-02-04 17:24:42.036854] INFO [clear_step6: uninstalling obproxy and remove docker, logfile: /root/oceanbase_trial/logs/uninstall_obproxy.log]
[2020-02-04 17:24:42.886604] INFO [clear_step6: uninstallation of obproxy done]
[2020-02-04 17:24:43.888851] INFO [clear_step5: uninstalling ocp and remove docker, logfile: /root/oceanbase_trial/logs/uninstall_tmp_ocp.log]
[2020-02-04 17:24:43.738130] INFO [clear_step5: uninstallation of temporary ocp done]
[2020-02-04 17:24:43.740634] INFO [clear_step4: drop ocp meta db/tenant/user/resource, logfile: /root/oceanbase_trial/logs/unit_metadb.log]
[2020-02-04 17:24:43.813946] INFO [clear_step4: uninit of metadb done]
[2020-02-04 17:24:43.816483] INFO [clear_step3: uninstalling OB server and remove docker, logfile: /root/oceanbase_trial/logs/uninstall_ob.log]
[2020-02-04 17:24:44.832929] INFO [clear_step3: uninstallation of ob done]
[2020-02-04 17:24:44.835425] INFO [clear_step1: no need to clear for step ssh authorization]
```

### 3.2.3. 单步安装或清理

比如说第3步重装一下。需要注意的是第3步如果是重装，建议先清理第3步及其以后所有步骤。

```
[root@xxx /root/oceanbase_trial]
#./install.sh -c 3- -i 3
```

理论上每一步都可以重新安装。但是不同步骤之间可能会有一些依赖关系。这里要关注的是重新安装前后是否有配置信息变更。如果配置文件变更了，元数据库那一步肯定是要重新执行的。

### 3.2.4. 单步安装 OB 容器

<待补充>

### 3.2.5. 安装后检查

```
[root@xxx /root/oceanbase_trial]
# docker ps
CONTAINER ID        IMAGE                                     STATUS      PORTS          NAMES                COMMAND
04d585acbeed        reg.docker.alibaba-inc.com/antman/ocp-all-in-one:OCP233_20200102_2122  Up 12 minutes  ocp                  "/bin/sh -c '/usr/bi..." 12
17097103b74b        reg.docker.alibaba-inc.com/antman/obproxy:OBP156_20200102_2110         Up 13 minutes  obproxy              "sh start_obproxy.sh"
dca537d692ea        reg.docker.alibaba-inc.com/antman/ob-docker:OB1478_20200102_2113       Up 22 minutes  META_OB_ZONE_1      "/usr/bin/supervisor..." 2
```

打开浏览器，访问 `http://11.xxx.87.5:8080`，能出现登录界面基本安装就成功了。



OCP安装好后如果安装OB集群，可以参考官方下载的试用版安装文档，或者参考云栖社区的直播。

## 4. 独立部署 OceanBase 集群

通常说OceanBase 集群都是三副本，至少有三台主机。实际发现 OceanBase 集群也支持单副本。并且单副本 OceanBase 集群还可以扩容（增加节点），所以也称为集群。

### 4.1. 机器初始化

- 请参照“[机器初始化 I \(手动做\)](#)”。
- 请参照“[机器初始化 II \(手动做, 可选\)](#)”。

### 4.2. 安装目录和软件

#### 4.2.1. 安装 OceanBase 软件

OceanBase 是单进程软件，只需要安装 `oceanbase-2.2.*-*****.el7.x86_64.rpm` 包和 `obproxy-1.5.*-*****.el7.x86_64.rpm`。

首先确保admin用户已经创建，相应的文件系统目录{ /home/admin/ , /data/1 , /data/log1 }都存在，并且空间大小符合要求。

```
$sudo rpm -ivh oceanbase-2.2.30-1855102.el7.x86_64.rpm
warning: Unable to get systemd shutdown inhibition lock: Unit is masked.
Preparing... ##### [100%]
Updating / installing...
 1:oceanbase-2.2.30-1855102.el7 ##### [100%]
```

oceanbase 软件会安装在目录/home/admin/oceanbase下。

### 4.2.2. 清理 OB 目录（第一次不用）

当要清理之前的 OB 环境，或者 OB 安装部署过程中出现问题导致环境变乱或者产生文件影响下一次安装时，选择直接清理老的 OB 目录。

```
su - admin
kill -9 `pidof observer`
sleep 3
/bin/rm -rf /data/1/obdemo
/bin/rm -rf /data/log1/obdemo
/bin/rm -rf /home/admin/oceanbase/store/obdemo /home/admin/oceanbase/log/* /home/admin/oceanbase/etc/*config*
ps -ef|grep observer
df -h |egrep home\|data
```

### 4.2.3. 初始化 OB 目录

OB 的数据目录通常建议在独立的磁盘上，然后通过软链接方式链接到软件 Home 目录下面。

```
su - admin
mkdir -p /data/1/obdemo/{etc3,sort_dir,sstable}
mkdir -p /data/log1/obdemo/{clog,etc2,ilog,slog,oob_clog}
mkdir -p /home/admin/oceanbase/store/obdemo
for t in {etc3,sort_dir,sstable};do ln -s /data/1/obdemo/$t /home/admin/oceanbase/store/obdemo/$t; done
for t in {clog,etc2,ilog,slog,oob_clog};do ln -s /data/log1/obdemo/$t /home/admin/oceanbase/store/obdemo/$t; done
```

复制上面脚本直接执行，然后执行下面命令检查结果。

```
[admin@ /home/admin/oceanbase]
$tree store/
store/
├── obdemo
│   ├── clog -> /data/log1/obdemo/clog
│   ├── clog_shm
│   ├── etc2 -> /data/log1/obdemo/etc2
│   ├── etc3 -> /data/1/obdemo/etc3
│   ├── ilog -> /data/log1/obdemo/ilog
│   ├── ilog_shm
│   └── oob_clog -> /data/log1/obdemo/oob_clog
```

```
└─ slog -> /data/log1/obdemo/slog
└─ sort_dir -> /data/1/obdemo/sort_dir
└─ sstable -> /data/1/obdemo/sstable
```

#### 4.2.4. 测试 IO 能力 (可选)

测试数据盘IO能力，生成性能报告文件放在 `/home/admin/oceanbase/etc` 目录下，observer启动时会读取这个文件进而自动设置内部一些跟IO有关的参数。

每个observer节点都需要运行，当磁盘容量很大的时候，这个比较费时间。如果要节省时间，你也可以直接复制下面 `io_resource.conf` 内容放到 `/home/admin/oceanbase/etc/` 下。

```
$time /home/admin/oceanbase/bin/ob_admin io_bench -c /home/admin/oceanbase/etc -d /data/1/obdemo user:root
succ to open, filename=ob_admin.log, fd=3, wf_fd=2

real    6m10.313s
user    1m27.734s
sys     1m15.621s
cat /home/admin/oceanbase/etc/io_resource.conf
version 1
io_type  io_size_byte  io_ps      io_rt_us
0         4096          68154.25    234.46
0         8192          44714.00    160.50
0         16384         26674.75    190.22
0         32768         13843.00    260.55
0         65536         7490.75     397.28
0         131072        3529.75     583.39
0         262144        1788.75     919.48
0         524288        927.00      1674.65
1         2097152       147.50      7301.02
submit_thread_cnt 8
getevent_thread_cnt 8
```

#### 4.2.5. 安装 OB 客户端

在下载文件里找到 `obclient-*.rpm`，这个是OceanBase命令行客户端，可以访问OceanBase的MySQL和ORACLE租户。

```
$ sudo rpm -ivh obclient-1.1.6-20191211162923.el7.alios7.x86_64.rpm

$ which obclient
/usr/bin/obclient
```

#### 4.2.6. 安装 OBProxy

```
$ sudo rpm -ivh obproxy-*.rpm
```

obproxy安装目录在/opt/taobao/install 下，通常我们作一个 到obproxy的软链接比较好

```
[admin@xxx /opt/taobao/install]
$pwd
/opt/taobao/install

[admin@xxx /opt/taobao/install]
$sudo ln -s obproxy-1.5.8 obproxy
[admin@xxx /opt/taobao/install]
$ll
total 12
drwxr-xr-x 9 admin admin 4096 Dec 18 2018 ajdk-8.0.0-b60
lrwxrwxrwx 1 admin admin 13 Jan 9 15:12 obproxy -> obproxy-1.5.8
drwxr-xr-x 6 admin admin 4096 Jan 9 15:14 obproxy-1.5.8
```

### 4.3.初始化 OB 集群

初始化 OB 集群的时候如果选择三台主机，分为三个 Zone ，那就是搭建三副本集群；如果是选择一台主机，只有一个 Zone ，就是搭建单副本集群。

#### 4.3.1. 启动节点 OBServer 进程

注意：我对我的测试ip其中部分字段用\*号打码了，这不是安装需求。

到每个节点的admin用户下，启动observer进程。注意三副本下，每个节点启动参数并不完全相同。

参数里指定数据文件的大小、内存的大小，以方便个别环境资源不足想精确控制observer对资源的占用。具体说明如下：

- -i 后接网卡名称。如果有 2 个网卡，要指定业务 IP 所在的网卡名称。
- -z 后接 Zone 名称，可以自定义。Zone 是逻辑机房概念，三台机器分属于不同机房，不能重复。
- -d 后跟集群主目录，除集群名字 obdemo 外，其他不要变动。
- -r 后跟一组服务器信息，格式是 ip:2882:2881，分号分割，表示 rootservice 信息。三台机器参数一致。
- -c 后跟集群 ID，一组数字，可以自定义。不同集群不要重复即可。
- -n 后跟集群名称，可以自定义，不同集群名称不要重复即可。
- -o 后跟集群参数，需要根据实际情况设置。system\_memory 指定 OB 内部保留内存，默认是 30G，机器内存比较少的环境下把这个调小，影响就是可能在性能测试时有内存不足问题。datafile\_size 指定 OceanBase 数据文件 sstable 的大小(一次性初始化)，根据 /data/1/ 可用空间评估，建议不少于 100G，同时又保留一些剩余空间。如果/data/1跟



/data/log1 本身就是一个文件系统(共用一个盘)，那么务必保证留给/data/log1 的空间大小是内存的 2-4 倍。config\_additional\_dir 指定参数文件的冗余目录。

三副本启动进程示例如下：

zone1:

```
su - admin
cd /home/admin/oceanbase && /home/admin/oceanbase/bin/observer -i bond0 -P 2882 -p 2881 -z zone1 -d /home/admin/oceanbase/store/obdemo -r '11.*.84.78:2882:2881;11.*.84.79:2882:2881;11.*.84.84:2882:2881' -c 20200102 -n obdemo -o "system_memory=10G,datafile_size=100G,config_additional_dir=/data/1/obdemo/etc3;/data/log1/obdemo/etc2"
sleep 5
ps -ef|grep observer
```

zone2:

```
su - admin
cd /home/admin/oceanbase && /home/admin/oceanbase/bin/observer -i bond0 -P 2882 -p 2881 -z zone2 -d /home/admin/oceanbase/store/obdemo -r '11.*.84.78:2882:2881;11.*.84.79:2882:2881;11.*.84.84:2882:2881' -c 20200102 -n obdemo -o "system_memory=10G,memory_limit=61440M,datafile_size=100G,config_additional_dir=/data/1/obdemo/etc3;/data/log1/obdemo/etc2"
sleep 5
ps -ef|grep observer
```

zone3:

```
su - admin
cd /home/admin/oceanbase && /home/admin/oceanbase/bin/observer -i bond0 -P 2882 -p 2881 -z zone3 -d /home/admin/oceanbase/store/obdemo -r '11.*.84.78:2882:2881;11.*.84.79:2882:2881;11.*.84.84:2882:2881' -c 20200102 -n obdemo -o "system_memory=10G,datafile_size=100G,config_additional_dir=/data/1/obdemo/etc3;/data/log1/obdemo/etc2"
sleep 5
ps -ef|grep observer
```

单副本 OB 进程启动示例如下：

zone1:

```
su - admin
cd /home/admin/oceanbase && /home/admin/oceanbase/bin/observer -i bond0 -P 2882 -p 2881 -z zone1 -d /home/admin/oceanbase/store/obdemo -r '11.*.84.78:2882:2881' -c 20200102 -n obdemo -o "system_memory=10G,datafile_size=100G,config_additional_dir=/data/1/obdemo/etc3;/data/log1/obdemo/etc2"
sleep 5
ps -ef|grep observer
```

### 4.3.2. 集群 bootstrap 操作

随意用 MySQL 命令登录一台节点，密码为空。

如果是三副本，则如下：

```
mysql -h127.1 -uroot -P2881 -p
Enter password:
```

```
set session ob_query_timeout=1000000000;alter system bootstrap ZONE 'zone1' SERVER '11.*.84.78:2882', ZONE 'zone2' SERVER '11.*.84.79:2882', ZONE 'zone3' SERVER '11.*.84.84:2882';
```

如果是搭建单副本集群，则如下：

```
mysql -h127.1 -uroot -P2881 -p
Enter password:
set session ob_query_timeout=1000000000;alter system bootstrap ZONE 'zone1' SERVER '11.*.84.78:2882';
```

### 注意：

如果这一步失败报错了，其原因很可能就是三节点或单节点的 observer 进程启动参数有不对、observer 相关目录权限不对、日志目录空间不足一定比例（跟数据目录合用了大目录，空间被数据目录占用了）、三节点时间不同步、节点内存资源不足等等。请先排查这些问题点后，然后清理OB 目录从头开始（参见“[清理 OB 目录（第一次不用）](#)”）。

## 4.3.3. 验证集群初始化成功

```
$mysql -h127.1 -uroot@sys -P2881 -p -c -A
Enter password:
show databases;
obclient> show databases;
+-----+
| Database          |
+-----+
| oceanbase          |
| information_schema |
| mysql              |
| SYS                |
| LBACSYS            |
| ORAAUDITOR         |
| test               |
+-----+
7 rows in set (0.02 sec)
```

能看到数据库列表里有 oceanbase 即可。

sys 租户的 root 密码默认为空，初始化成功后请修改密码。

```
obclient> alter user root identified by "123456";
Query OK, 0 rows affected (0.01 sec)
```

## 4.3.4. 初始化集群参数

OceanBases 集群支持通过参数去调整集群的行为。参数修改通常是及时生效，并持久化到参数文件中（/home/admin/oceanbase/etc/observer.config.bin）。这个参数文件是 binary 文件，直接查看会有乱码，通常使用 strings 命令。

```
[admin@h07g12092.sqa.eu95 /home/admin/oceanbase]
$strings etc/observer.config.bin|grep rootservice
rootservice_list=11.*.84.78:2882:2881;11.*.84.79:2882:2881;11.*.84.84:2882:2881
```

也可以在集群里直接查看参数 `show parameters like` 命令。

```
MySQL [oceanbase]> show parameters like '%rootservice_list%'G
***** 1. row *****
zone: zone1
svr_type: observer
svr_ip: 11.166.87.5
svr_port: 2882
name: rootservice_list
data_type: NULL
value: 11.*.84.78:2882:2881;11.*.84.79:2882:2881;11.*.84.84:2882:2881
info: a list of servers against which election candidate is checked for validation
section: OBSERVER
scope: CLUSTER
source: DEFAULT
edit_level: DYNAMIC_EFFECTIVE
1 row in set (0.00 sec)
```

如果 `/home/admin` 目录空间很紧张，则设置运行日志滚动输出。

```
mysql -h127.1 -uroot@sys -P2881 -p
Enter password:
-- observer log自清理设置
alter system set enable_syslog_recycle=True;
alter system set max_syslog_file_count=10;

MySQL [OceanBase]> show parameters where name in ('enable_syslog_recycle', 'max_syslog_file_count')G
***** 1. row *****
zone: zone1
svr_type: observer
svr_ip: 11.166.87.5
svr_port: 2882
name: enable_syslog_recycle
data_type: NULL
value: False
info: specifies whether log file recycling is turned on. Value: True: turned on; False: turned off
section: OBSERVER
scope: CLUSTER
source: DEFAULT
edit_level: DYNAMIC_EFFECTIVE
***** 2. row *****
zone: zone1
svr_type: observer
svr_ip: 11.166.87.5
svr_port: 2882
name: max_syslog_file_count
data_type: NULL
value: 0
info: specifies the maximum number of the log files that can co-exist before the log file recycling kicks in. Each log f
ile can occupy at most 256MB disk space. When this value is set to 0, no log file will be removed. Range: [0, +∞) in inte
ger
section: OBSERVER
scope: CLUSTER
source: DEFAULT
edit_level: DYNAMIC_EFFECTIVE
2 rows in set (0.00 sec)
```

## 4.4. 初始化 OBProxy

OceanBase 是分布式数据库，三副本的 OceanBase 集群至少有三台机器，业务数据的读写可能在其中任意一台主机上，对业务方（客户端）来记录这么多主机地址并不合理，所以 OceanBase 产品还提供了一个分布式代理 OBProxy 进程，用于监听客户端的连接并转发请求到后端 OBCServer 机器。默认监听端口是 2883，可以自定义为 3306 或 1521、或者其他端口。

OBProxy 的作用类似 ORACLE 的监听进程，不同之处是 OBProxy 会一直承担连接中转角色。转发到后端 OBCServer 的请求返回数据时，依然要通过 OBProxy 回发给客户端。OBProxy 跟分布式数据库中间件的代理节点不同的区别在于 OBProxy 只做路由转发，不做 SQL 运算（合并、排序、过滤等逻辑）。所以 OBProxy 很轻量，是无状态的，可以部署多个进程。但是 OBProxy 进程之间没有集群能力。

### 4.4.1. 初始化 OBProxy 账户

OBProxy 需要跟后端 OBCServer 节点保持通信，使用的是内部账户密码。所以需要提前提前在 OceanBase 集群的 sys 实例下为 OBProxy 创建连接账户和密码。（请复制下面 SQL 不要改动）。

```
mysql -h127.1 -uroot@sys -P2881 -p -c -A
Enter password:

CREATE USER proxyro IDENTIFIED BY password '*e9c2bcd178a99b7b08dd25db58ded2ee5bff050' ;
GRANT SELECT ON *.* to proxyro;
show grants for proxyro;
```

### 4.4.2. 启动 OBProxy

obproxy 安装目录在 /opt/taobao/install 下，通常我们作一个 到 obproxy 的软链接比较好。

```
[admin@xxx /opt/taobao/install]
$pwd
/opt/taobao/install

[admin@xxx /opt/taobao/install]
$sudo ln -s obproxy-1.5.8 obproxy
[admin@xxx /opt/taobao/install]
$ll
total 12
drwxr-xr-x 9 admin admin 4096 Dec 18 2018 ajdk-8.0.0-b60
lrwxrwxrwx 1 admin admin 13 Jan 9 15:12 obproxy -> obproxy-1.5.8
drwxr-xr-x 4 admin admin 4096 Jan 6 14:06 obproxy-1.5.5
drwxr-xr-x 6 admin admin 4096 Jan 9 15:14 obproxy-1.5.8
```

**注意：**

启动 OBProxy 时，请在 admin 用户下并在 OBProxy 软件的 home 目录。其他用户或者其他目录下启动都可能带来问题。

OBProxy 启动时需要知道 OceanBase 集群在哪里，这是通过参数 rootservice\_list 指定。下面以三副本 OceanBase 集群为例。

```
$ cd /opt/taobao/install/obproxy && bin/obproxy -r "11.*.84.78:2881;11.*.84.79:2881;11.*.84.84:2881" -p 2883 -o "enable_strict_kernel_release=false,enable_cluster_checkout=false,enable_metadb_used=false" -c obdemo
```

如果前面模拟了 OCP Web 服务，也可以指定 API 地址。（可选）

```
$ cd /opt/taobao/install/obproxy && bin/obproxy -p2883 -cobdemo -o "obproxy_config_server_url=http://11.*.84.83:8088/services?Action=GetObProxyConfig&User_ID=alibaba-inc&uid=ocpmaster,enable_cluster_checkout=false,enable_strict_kernel_release=false,enable_metadb_used=false"
```

启动之后，可以检查进程是否存在

```
$ ps -ef|grep obproxy
```

停止 OBProxy 进程的方法是直接 kill

```
$ kill -9 `pidof obproxy`
```

再次启动 OBProxy 进程时就不需要指定那么参数。因为参数已经写到参数文件里。

```
$ cd /opt/taobao/install/obproxy && bin/obproxy
```

OBProxy 的运行日志在 /opt/taobao/install/obproxy/log 下，也就是 /home/admin/logs/obproxy/log 下。

```
[admin@h07g12092.sqa.eu95 /opt/taobao/install/obproxy]
$ll log
lrwxrwxrwx 1 admin admin 28 Feb  6 11:46 log -> /home/admin/logs/obproxy/log
```

### 4.4.3. OB 连接示例

通过 OBProxy 连接时，用户名的格式需要包含集群名、实例名和用户名。格式为：用户名@实例名#集群名，或 集群名:实例名:用户名。

```
$mysql -h11.*.84.83 -uroot@sys#obdemo -P2883 -p -c -A oceanbase
Enter password:
```

或

```
$mysql -h11.*.84.83 -uobdemo:sys:root -P2883 -p -c -A oceanbase
Enter password:
```

#### 4.4.4. 调整 OBProxy 参数

下面是obproxy的一些参数配置用于减少运行日志量或降低 CPU 消耗，请根据实际情况修改。

```
alter proxyconfig set slow_proxy_process_time_threshold='1000ms';
alter proxyconfig set xflush_log_level=ERROR;
alter proxyconfig set syslog_level=WARN;
alter proxyconfig set enable_compression_protocol=false;
MySQL [OceanBase]> show proxyconfig like '%compress%'\G
***** 1. row *****
      name: enable_compression_protocol
      value: False
      info: if enabled, proxy will use compression protocol with server
      need_reboot: false
      visible_level: USER
1 row in set (0.00 sec)
```

### 4.5. OceanBase 分配多租户

OceanBase 集群将多个机器的资源能力聚合为一个大的资源池，然后再通过分配多租户的方式提供给不同业务使用。租户就是实例。OceanBase 集群的这种使用方式跟云数据库按需分配的思路是一致的。后期还可以在线扩容或缩容。

#### 4.5.1. 集群资源规划

集群资源指集群所有节点（OBServer进程）所收集的资源总和，扣除集群内部实例（sys）占用的少量资源外，剩余的都是集群的可用资源。

- 确认 OceanBase 集群可用资源。

```
select a.zone,concat(a.svr_ip,':',a.svr_port) observer, cpu_total, (cpu_total-cpu_assigned) cpu_free, round(mem_total/1024/1024/1024) mem_total_gb, round((mem_total-mem_assigned)/1024/1024/1024) mem_free_gb, usec_to_time(b.last_offline_time) last_offline_time, usec_to_time(b.start_service_time) start_service_time
from __all_virtual_server_stat a join __all_server b on (a.svr_ip=b.svr_ip and a.svr_port=b.svr_port)
order by a.zone, a.svr_ip
;
```

zone	observer	cpu_total	cpu_free	mem_total_gb	mem_free_gb	last_offline_time	start_service_time
1	zone1 11 .84.78-2882	30	27.5	51	38	1970-01-01 08:00:00.0	2020-01-06 16:10:31.50229
2	zone2 11 .84.79-2882	30	27.5	50	37	1970-01-01 08:00:00.0	2020-01-06 16:10:31.32146
3	zone3 11 .84.84-2882	30	27.5	51	38	1970-01-01 08:00:00.0	2020-01-06 16:10:32.254266

集群资源里划分一个小的资源池，分配给实例使用，就变成实例资源。每个实例的资源池通常是由至少三个同样大小的资源单元（UNIT）组成。

#### □ 确认 OceanBase 实例资源池分配细节

```
select t1.name resource_pool_name, t2.`name` unit_config_name, t2.max_cpu, t2.min_cpu, round(t2.max_memory/1024/1024/1024) max_mem_gb, round(t2.min_memory/1024/1024/1024) min_mem_gb, t3.unit_id, t3.zone, concat(t3.svr_ip,':',t3.`svr_port`) observer, t4.tenant_id, t4.tenant_name
from __all_resource_pool t1 join __all_unit_config t2 on (t1.unit_config_id=t2.unit_config_id)
join __all_unit t3 on (t1.`resource_pool_id` = t3.`resource_pool_id`)
left join __all_tenant t4 on (t1.tenant_id=t4.tenant_id)
order by t1.`resource_pool_id`, t2.`unit_config_id`, t3.unit_id
;
```



	name	unit_config_name	max_cpu	min_cpu	max_mem_gb	min_mem_gb	unit_id	zone	observer	tenant_id	tenant_name
1	sys_pool	sys_unit_config	5	2.5	15	13	1	zone1	11.34.78:2882	1	sys
2	sys_pool	sys_unit_config	5	2.5	15	13	2	zone2	11.34.79:2882	1	sys
3	sys_pool	sys_unit_config	5	2.5	15	13	3	zone3	11.34.84:2882	1	sys

每个实例的资源单元大小可以不相同，也可以相同，这由使用的资源规格模板决定。

#### □ 创建资源规格模板

```
CREATE resource unit my_unit_config max_cpu=20, min_cpu=15, max_memory='25G', min_memory='20G', max_iops=10000, min_iops=1000, max_session_num=1000000, max_disk_size='1024G';

// 如果要修改上面规格用下面sql
ALTER resource unit my_unit_config max_cpu=20, min_cpu=15, max_memory='25G', min_memory='20G';
;
```

## 4.5.2. 创建 ORACLE 租户（实例）

#### □ 分配资源池

```
CREATE resource pool bmsql_pool unit = 'my_unit_config', unit_num = 1;
```

#### □ 创建租户（关联资源池）

```
create tenant obbmsql resource_pool_list=('bmsql_pool'), primary_zone='RANDOM',comment 'oracle tenant/instance', charset='utf8' set ob_tcp_invited_nodes='%', ob_compatibility_mode='oracle';
```

创建租户的时候指定了租户使用的资源池、数据分布策略（`primary\_zone` 为RANDOM）、租户字符集（默认`utf8`，也可以改为`gbk`）、租户访问白名单（`ob\_tcp\_invited\_nodes`）、租户兼容级别（`ob\_compatibility\_mode`）。

#### □ 此时再次检查租户资源余量和分配细节

```
select a.zone,concat(a.svr_ip,',',a.svr_port) observer, cpu_total, (cpu_total-cpu_assigned) cpu_free, round(mem_total/1024/1024/1024) mem_total_gb, round((mem_total-mem_assigned)/1024/1024/1024) mem_free_gb, usec_to_time(b.last_offline_time) last_offline_time, usec_to_time(b.start_service_time) start_service_time
from __all_virtual_server_stat a join __all_server b on (a.svr_ip=b.svr_ip and a.svr_port=b.svr_port)
order by a.zone, a.svr_ip
;
```

SQL 表达式未通过结果 (使用 Ctrl+Space)

zone	observer	cpu_total	cpu_free	mem_total_gb	mem_free_gb	last_offline_time	start_service_time
1 zone1	1 84.78:2882	30	12.5	51	18	1970-01-01 08:00:00.0	2020-01-06 16:10:31.50229
2 zone2	1 84.79:2882	30	12.5	50	17	1970-01-01 08:00:00.0	2020-01-06 16:10:31.32146
3 zone3	1 84.84:2882	30	12.5	51	18	1970-01-01 08:00:00.0	2020-01-06 16:10:32.254266

```
select t1.name resource_pool_name, t2.`name` unit_config_name, t2.max_cpu, t2.min_cpu, round(t2.max_memory/1024/1024/1024) max_mem_gb, round(t2.min_memory/1024/1024/1024) min_mem_gb, t3.unit_id, t3.zone, concat(t3.svr_ip,',',t3.`svr_port`) observer, t4.tenant_id, t4.tenant_name
from __all_resource_pool t1 join __all_unit_config t2 on (t1.unit_config_id=t2.unit_config_id)
join __all_unit t3 on (t1.`resource_pool_id` = t3.`resource_pool_id`)
left join __all_tenant t4 on (t1.tenant_id=t4.tenant_id)
order by t1.`resource_pool_id`, t2.`unit_config_id`, t3.unit_id
;
```

SQL 表达式未通过结果 (使用 Ctrl+Space)

name	unit_config_name	max_cpu	min_cpu	max_mem_gb	min_mem_gb	unit_id	zone	observer	tenant_id	tenant_name
1 sys_pool	sys_unit_config	5	2.5	15	13	1	zone1	1 84.78:2882	1	sys
2 sys_pool	sys_unit_config	5	2.5	15	13	2	zone2	1 84.79:2882	1	sys
3 sys_pool	sys_unit_config	5	2.5	15	13	3	zone3	1 84.84:2882	1	sys
4 bmsql_pool	my_unit_config	20	15	25	20	1,001	zone1	1 84.78:2882	1,001	obdemo
5 bmsql_pool	my_unit_config	20	15	25	20	1,002	zone2	1 84.79:2882	1,001	obdemo
6 bmsql_pool	my_unit_config	20	15	25	20	1,003	zone3	1 84.84:2882	1,001	obdemo

## 连接 ORACLE 实例

ORACLE 实例的连接用户名是 sys，使用 obclient 客户端连接，不支持用 mysql 客户端连接。初始密码是空，可以修改。

```
$ obclient -h127.1 -usys@obdemo -P2883 -p -c -A
Enter password:

obclient> alter user sys identified by 123456;
Query OK, 0 rows affected (0.08 sec)

obclient>
obclient> select username from dba_users;
+-----+
| USERNAME |
+-----+
| SYS      |
| LBACSYS  |
| ORAAUDITOR |
+-----+
3 rows in set (0.10 sec)
obclient> show grants for sys;
+-----+
| Grants for SYS@% |
+-----+
| GRANT ALL PRIVILEGES ON *.* TO 'SYS' WITH GRANT OPTION |
| GRANT ALL PRIVILEGES ON "LBACSYS".* TO 'SYS' |
| GRANT ALL PRIVILEGES ON "ORAAUDITOR".* TO 'SYS' |
| GRANT ALL PRIVILEGES ON "SYS".* TO 'SYS' |
| GRANT ALL PRIVILEGES ON "oceanbase".* TO 'SYS' |
| GRANT ALL PRIVILEGES ON "___public".* TO 'SYS' |
| GRANT ALL PRIVILEGES ON "___recyclebin".* TO 'SYS' |
+-----+
7 rows in set (0.01 sec)
```



### 4.5.3. 创建 MySQL 实例

- 资源分配方法跟 ORACLE 实例类似。

```
CREATE resource pool mysql_pool unit = 'my_unit_config', unit_num = 1;

create tenant obmysql resource_pool_list=('mysql_pool'), primary_zone='RANDOM',comment 'mysql tenant/instance', charset='utf8' set ob_tcp_invited_nodes='%', ob_compatibility_mode='mysql';
```

- 连接 MySQL 实例

MySQL 实例可以使用 mysql 客户端或者 obclient 客户端连接。

```
$mysql -h127.1 -uroot@obmysql#obdemo -P2883 -p -c -A test
Enter password:

alter user root identified by '123456';

MySQL [test]> show databases;
+-----+
| Database          |
+-----+
| oceanbase          |
| information_schema |
| mysql              |
| test               |
+-----+
4 rows in set (0.01 sec)

MySQL [test]>
```

### 4.5.4. 调整实例参数

OceanBase 的租户（实例）也支持调整参数，影响范围是实例级别的，不同实例的参数互不影响。不过在实例里通常把参数叫做变量（variables），修改方法跟 MySQL 实例变量一致。

初次使用 OceanBase 的开发，会遇到各种超时问题。我在文章《[从ORACLE/MySQL到OceanBase：数据库超时机制](#)》里曾经介绍过OceanBase超时变量设置知识，这里修改一下ORACLE租户的默认设置，尽可能的跟ORACLE保持一致。

```
$obclient -h127.1 -usys@obmysql#obdemo -P2883 -p -c -A
Enter password:

SET GLOBAL ob_query_timeout = 10000000000;
SET GLOBAL ob_trx_idle_timeout = 12000000000;
SET GLOBAL ob_trx_timeout = 10000000000;

SHOW GLOBAL variables WHERE variable_name IN ('ob_query_timeout', 'ob_trx_idle_timeout', 'ob_trx_timeout');
+-----+-----+
| VARIABLE_NAME | VALUE |
+-----+-----+
```

```

+-----+-----+
| ob_query_timeout      | 10000000000 |
| ob_trx_idle_timeout   | 12000000000 |
| ob_trx_timeout        | 10000000000 |
+-----+-----+
3 rows in set (0.01 sec)

```

#### 4.5.5. OceanBase 数据库图形化客户端 DBeaver

DBeaver 是一款开源的通用的数据库客户端，通过添加 OceanBase 的 JDBC 驱动，DBeaver 也支持 OceanBase 数据库的使用。具体请参考《[用DBeaver工具连接OceanBase数据库](#)》。

更多 OceanBase ORACLE/MySQL实例的使用帮助请参考[相关文档](#)中的开发者指南。

## 5. 使用 OCP 部署 OceanBase 集群

注意：下面是以 OCP 2.3.5 版本为例，不同版本的 OCP 功能会有些细节差异。

### 5.1. 机器资源管理

OCP 里可以维护三类机器资源：

- ▣ OBServer 机器
- ▣ OBProxy 机器
- ▣ 备份和恢复机器

#### 5.1.1. 机器录入前提

在 OCP 里维护这三类机器资源之前，先要确保机器先满足一些条件。

- ▣ OBServer 机器的初始化跟独立部署 OceanBase 集群中机器要求一样，参考“[机器初始化](#)”。
- ▣ OBProxy 机器的初始化要求少一些，只需要做参数初始化和用户初始化，参考“[机器初始化 I \(手动做\)](#)”，“[修改相关参数](#)”和“[新建用户](#)”。
- ▣ 备份和恢复机器的初始化要求也只需要做参数初始化和用户初始化。参考“[机器初始化 I \(手动做\)](#)”，“[修改相关参数](#)”和“[新建用户](#)”。

### 5.1.2. 自定义机型信息（可选）

OCF 管理 OBCServer 机器资源时，为区别不同机器的能力，设计了机型信息。默认的机型值是 DOCKER，是 OCF 的元数据库 OB 使用，是个 Docker 容器。

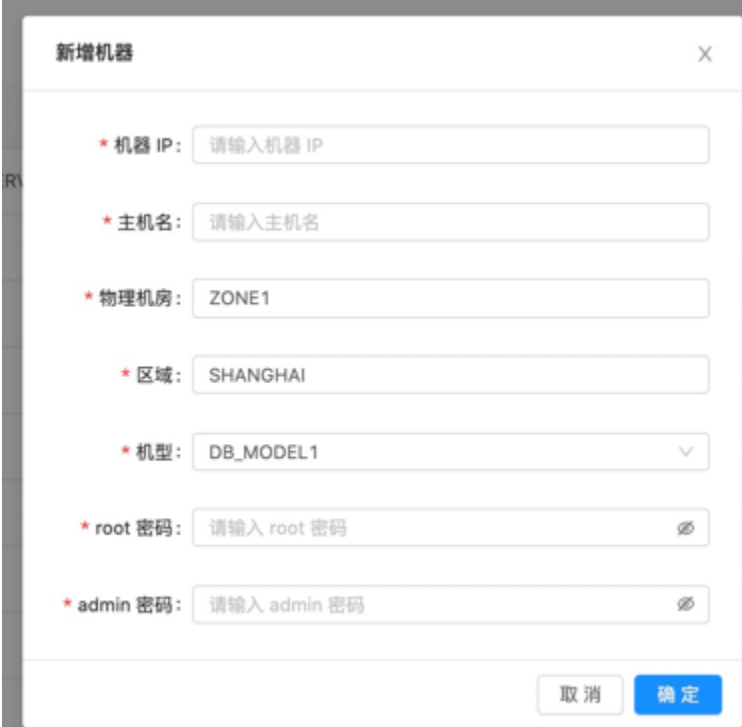
当前版本的 OCF 可能没有这个机型信息的维护界面，需要手动到元数据库里插入一笔记录。

```
#obclient -h127.1 -uroot@ocf_meta#obcluster -P2883 -p -c -A ocf
Enter password:

use ocf;
insert into ocf_sm_config (type, os, cpu, memsize, bondinfo, eth0, eth1)
VALUES ('DB_MODEL1', 'CentOS', 56, 384, 'bonded', '10000', '10000');
COMMIT;
```

### 5.1.3. 录入 OBCServer 机器

OBCServer 机器录入路径：OCF-【资源】-【添加机器】-【添加机器】



The image shows a web form titled "新增机器" (Add Machine) with a close button (X) in the top right corner. The form contains several input fields, each with a red asterisk indicating it is required:

- \* 机器 IP: 请输入机器 IP (Machine IP: Please enter machine IP)
- \* 主机名: 请输入主机名 (Hostname: Please enter hostname)
- \* 物理机房: ZONE1 (Physical Data Center: ZONE1)
- \* 区域: SHANGHAI (Region: SHANGHAI)
- \* 机型: DB\_MODEL1 (Machine Type: DB\_MODEL1, with a dropdown arrow)
- \* root 密码: 请输入 root 密码 (root password: Please enter root password, with a password icon)
- \* admin 密码: 请输入 admin 密码 (admin password: Please enter admin password, with a password icon)

At the bottom right of the form, there are two buttons: "取消" (Cancel) and "确定" (Confirm).

机器录入时信息说明：

- 机器 IP：如果机器是双 IP，这里一定要录入网卡速率高的那个网卡对应的 IP（即万兆网络的 IP），以免影响后期性能测试。
- 主机名：跟实际主机名保持一致，以免后期运维失误。

- 物理机房：通常填实际机房信息即可。如果要模拟多机房，可以人为填不同的机房。如 ZONE1、ZONE2、ZONE3。
- 区域：通常填机房的区域信息。开发测试环境简单起见所有机器都填一个区域，后面部署 OceanBase 集群少一些问题。
- 机型：默认机型数据是 Docker，在 OCP 元数据库里定义。目前 OCP 版本没有定义功能，直接改元数据库表（参考“[自定义机型信息（可选）](#)”）。开发测试环境就选择默认机型数据 Docker，尽管实际是物理机。

## 5.2. 软件管理

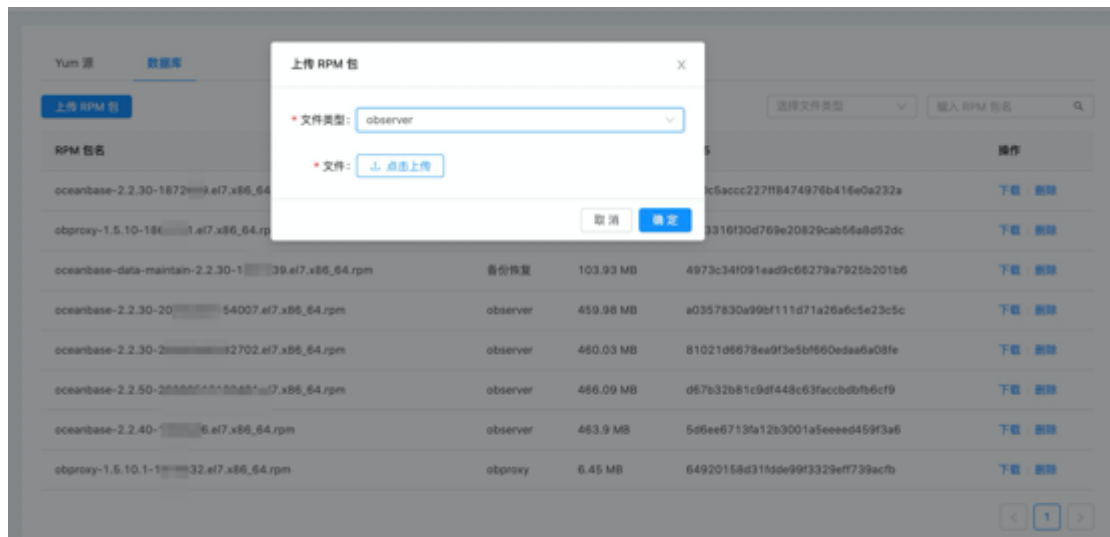
### 5.2.1. 录入相关软件

OCP 要将各类软件版本管理起来，方便软件升级。

软件包括：

- observer 软件：OB 内核进程，部署在 OBCServer 机器上。
- obproxy 软件：OB 访问用的反向代理，部署在 OBCServer 机器、虚拟机或应用服务器上。
- obagent 软件：OBCServer 机上的 OCP 客户端进程。
- 备份恢复软件：用于 OB 的备份和恢复。

路径：【运维】-【RPM 包管理】-【数据库】。



## 5.3. 集群管理

### 5.3.1. 新建集群

当 OBCServer 软件和机器资源都维护到 OCP 里以后，就可以新建 OceanBase 集群。

路径：【集群】-【新建集群】

集群名称: obdemo  
集群名称只支持英文、数字和下划线，且长度不能超过 48

OB 版本: current: oceanbase-2.2.50-2020 | 17.x86\_64.rpm

集群分组: obgroup  
集群分组名称只支持英文、数字和下划线，且长度不能超过 48

集群模型: 1-1-1-0-0 机房&机型分配

表示每个 Zone 的 OBCServer 数量，完成集群模型配置后，添加相应的机房和机型信息

OBServer 数量	1	1	1	0	0
机房及机型	BEIJING DB_MODEL1	BEIJING DB_MODEL1	am171 DB_MODEL1	机房 机型	机房 机型

机房及机器详情

创建 取消

说明：

- 集群名称：尽可能简单明了，太长了后面数据库访问账户也会很长，不是很方便。
- OB 版本：下来选择合适的版本。
- 集群分组：给 OB 集群一个分组属性，当集群很多时，按集群对应的业务区分不同 OB 集群。
- 集群模型：如果是三副本架构，就是 1-1-1-0-0；如果是五副本架构，就是 1-1-1-1-1。
- 机房和机型：为集群每个副本挑选对应的机房和机型。**注意：**至少要有 2 个机房是来自于同一个城市。
- 点击“机房以及机器详情”可以查看该机房该机型的剩余机器数。要确保剩余机器满足集群搭建要求才点击“创建”按钮。
- 点击“创建”后，会生成一个 任务，名字叫“INSTALL”。

### 5.3.2. 集群安装任务说明

在标准环境里，新建集群成功的概率很高，时间通常在 30 分钟左右。如果 OBCServer 机器数量多于 3 台，或者机器的磁盘很大时，这个时间会更长一些。当在非标环境里，集群安装任务可能会因为环境问题而报错。

路径：**【运维】-【任务】**，描述里输入：INSTALL，点击“搜索”

\* 状态:

全部

创建者:

请输入任务创建者

描述:

INSTALL

ID:

请输入任务ID

搜索

重置

任务 ID	任务名称	任务描述	用户	状态	创建时间	结束时间	操作	
-	1000035	INSTALL	INSTALL ob_new	admin	成功	2020-05-12 18:41:46	2020-05-12 19:20:16	
子任务 ID	创建时间	调度时间	子任务执行机器	子任务执行状态	操作			
1000089	2020-05-12 18:41:45	2020-05-12 18:41:46	ocp1-44979	成功(共 32 步)	<a href="#">操作信息</a> <a href="#">详情</a>			
1000090	2020-05-12 18:41:45	2020-05-12 18:41:46	ocp1-85698	成功(共 3 步)	<a href="#">操作信息</a> <a href="#">详情</a>			

INSTALL 任务会分为 2 个子任务，主要是第一个子任务。点进去，会看到更多原子任务。

任务 / 任务详情						
任务详情   1000089						
ID	名称	描述	期望耗时	状态	操作	
10	at_install_rpm	<a href="#">详情</a>	200	成功		
11	at_bootstrap_observer	<a href="#">详情</a>	800	成功		
12	at_add_observer	<a href="#">详情</a>	800	成功		
13	at_stop_observer	<a href="#">详情</a>	800	成功		
14	at_start_observer_without_param	<a href="#">详情</a>	800	成功		
15	at_init_obs_list_with_rs	<a href="#">详情</a>	30	成功		
16	at_major_freeze	<a href="#">详情</a>	10	成功		
17	at_set_cluster_id	<a href="#">详情</a>	10	成功		
18	at_create_unit_config	<a href="#">详情</a>	300	成功		
19	at_password_sys_root	<a href="#">详情</a>	30	成功		
						<a href="#">&lt;</a> <a href="#">1</a> <a href="#">2</a> <a href="#">3</a> <a href="#">4</a> <a href="#">5</a>

大部分原子任务如果失败后，点开任务编号前的 **+**，可以看到相应的日志。在任务后面的状态里，如果任务失败了，后面操作里可以“重试任务、跳过任务或者放弃任务”。

- 重试任务：根据任务报错日志找到失败原因，修复问题后，就点击“重试任务”。大部分子任务都支持“重试”操作。如果重试失败，子任务会回滚部分完成的动作。
- 跳过任务：当手动执行了这个子任务要做的事情时，在 OCP 里可以跳过这个任务，继续执行下一步任务。注意，在不熟悉这个子任务的情况下，跳过子任务可能会为将来埋下风险。
- 放弃任务：该子任务改为结束状态，直接终止所在父任务。可能有部分已完成的子任务步骤无法回滚。

at\_bootstrap\_observer 任务做了 OBServer 部署初始化的最关键的逻辑，跟手动部署 OB 的逻辑相同。请先查看“[安装目录和软件](#)”和“[初始化OB 集群](#)”。下面节选这个任务的部分日志可以看出它的主要逻辑。

```

2020-05-12 18:45:51 - info - 44979 - ob_ssh:106 - begin to execute on 11.1.6.27, cmd : ps -ef | grep -w observer | grep -v grep | awk '{print $2}' | xargs --no-run-if-empty kill -9

2020-05-12 18:45:58 - info - 44979 - ob_ssh:106 - begin to execute on 11.1.6.27, cmd : for file in `find /data/1 -type f`;do /bin/rm -f $file; done

2020-05-12 18:46:05 - info - 44979 - ob_ssh:106 - begin to execute on 11.1.6.27, cmd : rm -fr /data/log1/*

2020-05-12 18:46:12 - info - 44979 - ob_ssh:106 - begin to execute on 11.1.6.27, cmd : mkdir -p /data/1/ob_new/{sstable,etc3,sort_dir} /data/log1/ob_new/{slog,clog,ilog,oob_clog,etc2} /home/admin/oceanbase/store/ob_new && chown -R admin:admin /data/1/ob_new && chown -R admin:admin /home/admin/oceanbase && chown -R admin:admin /data/log1

2020-05-12 18:46:15 - info - 44979 - ob_ssh:106 - begin to execute on 11.1.6.27, cmd : ln -sf /data/1/ob_new/sstable /home/admin/oceanbase/store/ob_new/sstable && ln -sf /data/1/ob_new/sort_dir /home/admin/oceanbase/store/ob_new/sort_dir && ln -sf /data/log1/ob_new/slog /home/admin/oceanbase/store/ob_new/slog && ln -sf /data/log1/ob_new/clog /home/admin/oceanbase/store/ob_new/clog && ln -sf /data/log1/ob_new/ilog /home/admin/oceanbase/store/ob_new/ilog && ln -sf /data/log1/ob_new/oob_clog /home/admin/oceanbase/store/ob_new/oob_clog

2020-05-12 18:46:17 - info - 44979 - ob_ssh:55 - begin to execute on 11.1.6.27, cmd: /home/admin/oceanbase/bin/ob_admin io_bench -c /home/admin/oceanbase/etc -d /data/1/ob_new

2020-05-12 19:09:17 - info - 44979 - ob_ssh:55 - begin to execute on 11.1.6.27, cmd: cd /home/admin/oceanbase; ulimit -s 10240; ulimit -c unlimited; LD_LIBRARY_PATH=/home/admin/oceanbase/lib:/usr/local/lib:/usr/lib:/usr/lib64:/usr/local/lib64:$LD_LIBRARY_PATH LD_PRELOAD="" /home/admin/oceanbase/bin/observer -i bond0 -p 2881 -P 2882 -n ob_new -z BEIJING_1 -d /home/admin/oceanbase/store/ob_new -l info -o "obconfig_url=http://11.1.4.64:8080/services?Action=ObRootServiceInfo&User_ID=alibaba&UID=admin&ObRegion=ob_new,rootSERVICE_list=11.1.6.27:2882:2881;11.1.6.28:2882:2881;11.1.6.29:2882:2881,datafile_disk_percentage=95,config_additional_dir=/data/log1/ob_new/etc2;/data/1/ob_new/etc3,cluster_id=2100001,__min_full_resource_pool_memory=1073741824,_ob_enable_prepared_statement=false"

2020-05-12 19:09:52 - info - 44979 - ob_operate_plus:558 - svr(11.1.6.28) /data/1 disk size less than 2048G, no need modify datafile_disk_percentage param

2020-05-12 19:12:02 - info - 44979 - ob_operate_plus:412 - execute cmd:mysql -uroot -h11.1.6.27 -P2881 -e "set ob_query_timeout=1000000000; alter system bootstrap REGION 'BEIJING' ZONE 'BEIJING_1' SERVER '11.1.6.27:2882', REGION 'BEIJING' ZONE 'BEIJING_2' SERVER '11.1.6.28:2882', REGION 'BEIJING' ZONE 'BEIJING_3' SERVER '11.1.6.29:2882'"

```

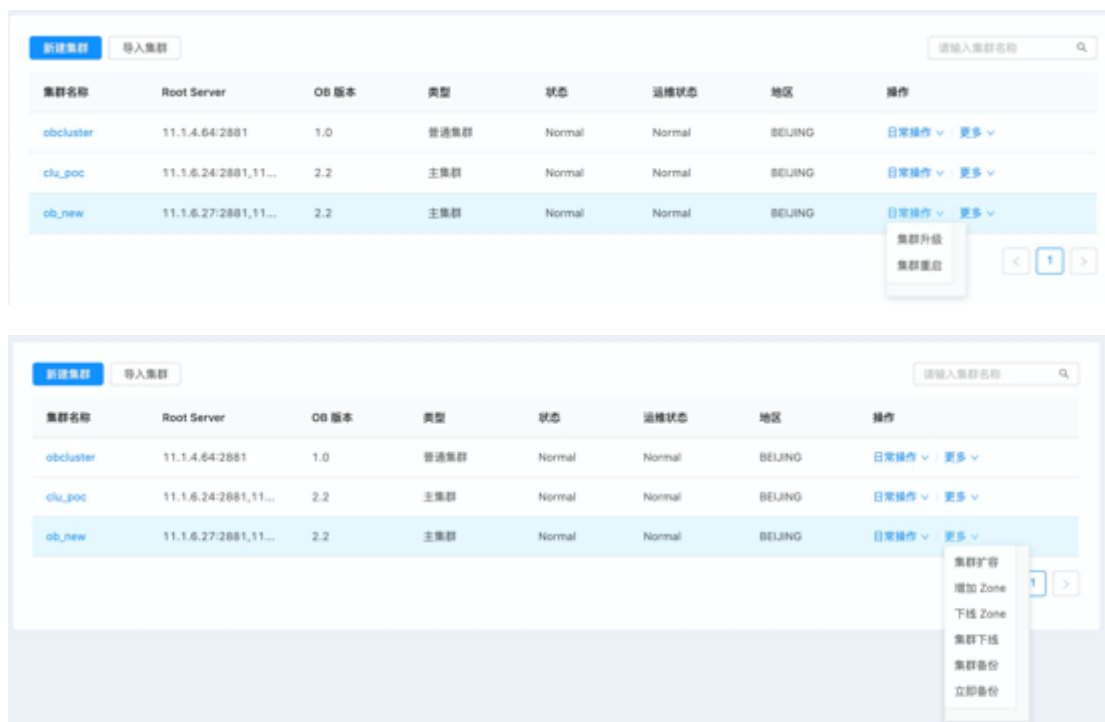
说明：

- 第一次安装集群时，ob\_admin io\_bench 操作会测试 IO 性能，这个跟数据盘 /data/1 的大小和性能有关。
- 相关目录 /data/1 /data/log1 要提前准备好，权限要对。
- 最后一步 alter system bootstrap 语句如果报错了，说明机器环境有些条件不满足要求。请参考[“机器初始化”](#)。这一步失败了，一定是机器环境问题，不需要问 OB 怎么分析或解决这个问题。最快的方法只有排查前提条件，然后重试一法。

### 5.3.3. 集群运维

集群的常规运维操作都在集群信息列表右边的“日常操作”和“更多”里。

路径：**【集群】**



在部署的时候，如果 OB 集群初始规模是 1-1-1，后期可以集群扩容到 2-2-2。扩容时建议每个 ZONE 增加相同的机器数。



### 5.3.4. 集群资源查看

OceanBase 集群并不能直接提供给业务使用，还需要从中分配出实例。分配实例时要指定相应的资源大小。在做之前需要先对集群的可用资源进行摸底。



目前 OCP 没有资源展示页面，所以主要是到 OB 集群的默认实例 sys 里查看。默认实例访问用户名是：root@sys#ob\_new，ob\_new 是集群名。密码在 INSTALL 任务的子任务 at\_password\_sys\_root 的任务日志里。

```
#mysql -h11.1.6.24 -uroot@sys#ob_new -P2883 -p -c -A oceanbase
Enter password:
```

```
select a.zone,concat(a.svr_ip,',',a.svr_port) observer,cpu_total,(cpu_total-cpu_assigned) cpu_free,
round(mem_total/1024/1024/1024) mem_total_gb,round((mem_total-mem_assigned)/1024/1024/1024) mem_free_gb,
usec_to_time(b.last_offline_time) last_offline_time,usec_to_time(b.start_service_time) start_service_time,b.status
from __all_virtual_server_stat a join __all_server b on (a.svr_ip=b.svr_ip and a.svr_port=b.svr_port)
order by a.zone, a.svr_ip
```

zone	observer	cpu_total	cpu_free	mem_total_gb	mem_free_gb	last_offline_time	start_service_time	status
BEIJING_1	11.1.6.27:2882	54	34.5	110	15	2020-05-12 23:56:39.093484	2020-05-12 22:26:27.638844	active
BEIJING_2	11.1.6.28:2882	54	34.5	110	15	2020-05-12 22:31:21.886247	2020-05-12 22:32:32.002938	active
BEIJING_3	11.1.6.29:2882	54	34.5	110	15	2020-05-12 22:37:21.611136	2020-05-12 22:39:06.562892	active

3 rows in set (0.02 sec)

查看集群当前资源分配细节：

```
select t1.name resource_pool_name, t2.`name` unit_config_name, t2.max_cpu, t2.min_cpu,
round(t2.max_memory/1024/1024/1024) max_mem_gb, round(t2.min_memory/1024/1024/1024) min_mem_gb, t3.unit_id,
t3.zone, concat(t3.svr_ip,',',t3.svr_port) observer, t4.tenant_id, t4.tenant_name
from __all_resource_pool t1 join __all_unit_config t2 on (t1.unit_config_id=t2.unit_config_id)
join __all_unit t3 on (t1.`resource_pool_id` = t3.`resource_pool_id`)
left join __all_tenant t4 on (t1.tenant_id=t4.tenant_id)
order by t1.`resource_pool_id`, t2.`unit_config_id`, t3.unit_id
```

resource_pool_name	unit_config_name	max_cpu	min_cpu	max_mem_gb	min_mem_gb	unit_id	zone	observer	tenant_id	tenant_name
sys_pool	sys_unit_config	5	2.5	30	25	1	BEIJING_1	11.1.6.27:2882	1	sys
sys_pool	sys_unit_config	5	2.5	30	25	2	BEIJING_2	11.1.6.28:2882	1	sys
sys_pool	sys_unit_config	5	2.5	30	25	3	BEIJING_3	11.1.6.29:2882	1	sys
pool_oracle	ora_tenant_unit	17	17	70	70	1004	BEIJING_1	11.1.6.27:2882	1003	ora_tenant
pool_oracle	ora_tenant_unit	17	17	70	70	1005	BEIJING_2	11.1.6.28:2882	1003	ora_tenant
pool_oracle	ora_tenant_unit	17	17	70	70	1006	BEIJING_3	11.1.6.29:2882	1003	ora_tenant

6 rows in set (0.01 sec)

OceanBase 资源池的概念请参考“[集群资源规划](#)”。

## 5.4. 实例管理

OceanBase 集群初始化好后，就准备新建实例。新建实例之前先要为业务制定适合实际情况的资源单元规格（模板）。

### 5.4.1. 新建资源单元规格

路径：**【实例】-【实例规格】**

这里列举了一些默认的实例规格模板，跟你的集群可用资源大小不一定匹配，所以需要定义自己的实例规格。此外，在 OB 2.2 版本集群里，分配实例时

内存最小不要少于 5G 。所以有些小规格不要使用。

实例 / 实例规格

← 实例规格

新建实例规格

全部 请输入规格名称

规格名称	规格类型	CPU	磁盘空间	内存	每秒 IO 次数	最大连接数
B	基本	0.25	120 GB	1 GB	250	75
LogOnlyNormal	logonly	1	500 GB	2 GB	1000	300
LogOnlySystem	logonly	5	500 GB	35 GB	5000	1500
S0	标准	0.5	500 GB	2 GB	500	150
S1	标准	1.5	500 GB	6 GB	1250	375
S2	标准	3	500 GB	12 GB	2500	750
S3	标准	6	500 GB	20 GB	5000	1500
S4	标准	12	500 GB	40 GB	5000	3000
test_unit	用户自定义	20	200 GB	100 GB	10000	500

## 5.4.2. 新建实例

路径：【实例】-【新建实例】

实例 / 新建实例

← 新建实例

OB 版本: 1.0 2.0 2.1 2.2

部署模式: Oracle MySQL

集群分组: ob\_new

实例名称: oboracle

实例规格: 用户自定义 test\_unit 详情

可用区: 默认可用区

租户白名单: %

租户总数: 1

ob\_new 的租户列表: oboracle0

在该物理集群上创建的租户前缀列表, 如无单物理集群多租户功能, 请忽略

BEIJING\_1 ☒ F ☐ R ☐ L ☐ 无

BEIJING\_2 ☒ F ☐ R ☐ L ☐ 无

BEIJING\_3 ☒ F ☐ R ☐ L ☐ 无

确定 取消

说明:

- OB 版本: 选择 2.2。实例的版本就是所在集群的版本。

- 部署模式：ORACLE 或 MySQL 二选一，就是兼容模式。实例创建后就不能修改。
- 集群分组：从哪个 OB 集群分组中分配实例。
- 实例名称：指定实例名称前缀，OCP 会在后面增加 3 位。实例名称不要太长，否则后面实例用户名会很长。
- 可用区：目前外部输出没有用途。
- 租户白名单：指定实例连接的客户端白名单。默认是 % 表示允许所有来源。可以指定 IP、网段等，以分号隔开。如 192.168.0.0/16;10.100.0.0/16
- 租户总数：默认是 1。如果大于 1，则一次性创建多个相同规格和相同实例名前缀的实例。这是给 SOFA 的 DBP 调用 OCP 创建一组实例用的。
- 最后一组是勾选三个 ZONE 的副本类型。F：表示全功能副本，是默认副本类型；R：是只读副本，L：是日志副本。属于高级玩法。
- 点击“确定”后，会生成后台任务，任务名称“add user instance”。

### 5.4.3. 实例新建任务说明

路径：【运维】-【任务】，输入描述：create user instance，点击“搜索”

\* 状态:

全部

创建者:

请输入任务创建者

描述:

create user instance

ID:

请输入任务ID

搜索

重置

任务 ID	任务名称	任务描述	用户	状态	创建时间	结束时间	操作
<div>[-]</div> 1000036	add user instance	create user insta...	admin	<div>●</div> 成功	2020-05-12 19:26:36	2020-05-12 19:33:46	
子任务 ID	创建时间	调度时间	子任务执行机器	子任务执行状态	操作		
1000091	2020-05-12 19:26:35	2020-05-12 19:26:36	ocp1-98550	成功(共 19 步) <div>●</div>	<a href="#">操作信息</a> <a href="#">详情</a>		
1000092	2020-05-12 19:26:35	2020-05-12 19:26:36	ocp1-99886	成功(共 3 步) <div>●</div>	<a href="#">操作信息</a> <a href="#">详情</a>		
1000093	2020-05-12 19:26:35	2020-05-12 19:26:36	ocp1-100414	成功(共 3 步) <div>●</div>	<a href="#">操作信息</a> <a href="#">详情</a>		
<div>+</div> 1000021	add user instance	create user insta...	admin	<div>●</div> 成功	2020-04-26 17:19:21	2020-04-26 17:21:46	
<div>+</div> 1000017	add user instance	create user insta...	admin	<div>●</div> 成功	2020-04-25 21:09:47	2020-04-25 21:12:16	

实例新建任务分为 3 个子任务，主要是第一个子任务，点击子任务 ID。

子任务里又分为一组原子任务，这些原子任务就是创建实例的几个过程，关键的步骤也就 3 个。可以跟手动创建实例时步骤对比观看

任务 / 任务详情

← 任务详情 | 1000091

ID	名称	描述	期望耗时	状态	操作
0	at_init	<a href="#">详情</a>	10	成功	
1	at_set_operating	<a href="#">详情</a>	10	成功	
2	at_init_variable	<a href="#">详情</a>	10	成功	
3	at_check_and_create_unit_config	<a href="#">详情</a>	10	成功	
4	at_create_log_only_resource_pool	<a href="#">详情</a>	10	成功	
5	at_create_read_only_resource_pool	<a href="#">详情</a>	10	成功	
6	at_create_full_resource_pool	<a href="#">详情</a>	30	成功	
7	at_create_tenant	<a href="#">详情</a>	30	成功	
8	at_create_idb_ddl_4_tenant	<a href="#">详情</a>	30	成功	
9	at_create_drc_user	<a href="#">详情</a>	30	成功	

#### 5.4.4. 访问实例

路径：**【实例】**，显示当前实例列表。单击其中一个实例名称，进入**【实例详情】**。

实例 / 实例详情

← 实例详情 | ten\_orc

基本信息

性能指标

连接串信息

内网 IP

用户名

租户

所属集群

公网 IP

OB Proxy 信息

实例 ID: 21

实例名称: ten\_orc

实例规格: S2

OceanBase 版本: 2.2

创建时间: 2020-04-26 17:19:21

过期时间: 内部实例为永久实例

Primary Zone:

可维护时间段

当前可维护时间: 18:00 - 22:00 修改

操作

重置密码

修改实例规格

删除

这里会显示实例的多个信息。其中根用户、租户和所属集群三样信息，组合在一起就是访问实例的完整的用户名。如：sys@ten\_oracle021#clu\_poc，另外一种格式是：clu\_poc:ten\_oracle021:sys。

实例新建好时，左下方会要求设置密码，设置后会显示为【重置密码】，以后还可以再改。这里设置的是 sys 用户的密码。

实例支持其他操作就是【修改实例规格】，这个就是在线弹性扩容或缩容。

还有一个操作是【实例删除】。

这里没有展示实例连接的 IP，因为这个 IP 跟实例没有直接关系。OB 里的实例都是逻辑实例，不跟具体的 OBPProxy 机器绑定。实例连接时通过连接 OBPProxy 实现的。

实例的访问方式请参考《OceanBase ORACLE 实例开发指南》。主要是 obclient 和 DBeaver 两种访问方式。

## 5.5.OBProxy 运维

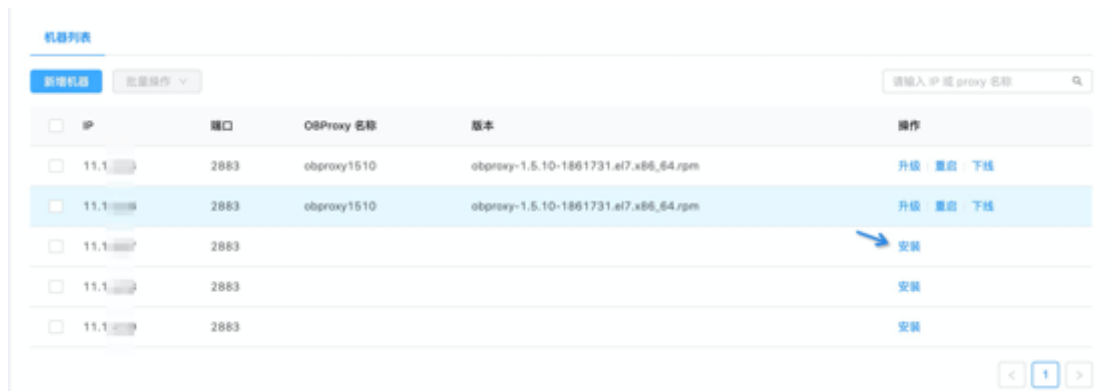
### 5.5.1. 新增 OBProxy 机器

路径：【运维】-【OBProxy】-【新增机器】。

输入 ip 和 root、admin 的密码即可。

然后会生成任务“添加OBProxy”机器，做一些基本的权限检查和设置。

然后在 OBProxy 机器列表的机器后面点击“安装”



OBProxy 默认安装在用户 admin 下的目录 /opt/taobao/install/obproxy 下。

## 6. OceanBase 常用运维和开发经验

### 6.1. 参数和变量初始化

OceanBase 里参数和变量的含义基本相同。参数（parameters）来自于 Oracle 的术语，变量（variables）来自于 MySQL 的术语。通常参数是在集群级别设置，作用范围是全集群所有节点所有实例；变量是在实例级别设置，作用范围是具体的实例。当然，在实例里也可以设置部分参数，作用范围也限于该实例。

#### 6.1.1. 集群参数读取和修改

集群参数的读取和修改都是在 sys 实例下，用户名是 root。  
查看集群所有参数命令：

```
SHOW parameters;
```

查看具体的参数命令是：

```
SHOW parameters LIKE '%sql_audit%';
```

参数说明里描述了该参数的作用和作用范围。

修改参数的命令是：

```
ALTER SYSTEM SET enable_sql_audit=TRUE;
```

修改参数的时候也可以指定租户名。

```
ALTER SYSTEM SET enable_sql_audit=TRUE tenant='obmysql';
```

#### 6.1.2. 实例变量读取和修改

实例的变量也可以改变实例的行为，需要单独设置，在业务实例里通过管理员账户修改。

查看变量：

```
show global | session variables ;
```

查看具体的变量

```
show global variables like '%xxx%';
```

示例：

```
obclient> show global variables like '%timeout%';
+-----+-----+
| VARIABLE_NAME | VALUE |
+-----+-----+
| connect_timeout | 10 |
| interactive_timeout | 28800 |
| net_read_timeout | 30 |
| net_write_timeout | 60 |
| ob_query_timeout | 999999999999 |
| ob_trx_idle_timeout | 999999999999 |
| ob_trx_lock_timeout | -1 |
| ob_trx_timeout | 999999999999 |
| wait_timeout | 28800 |
+-----+-----+
9 rows in set (0.01 sec)
```

修改变量方法：

```
set global | session xxx = xxx;
```

global 指修改实例全局设置，影响所有后续新的会话，但不影响当前所有会话。当前会话需要断开重连才会生效。

session 指修改会话变量设置，只影响当前会话的设置，不影响其他会话或者后续新增会话。

### 6.1.3. 集群参数建议

参数名	默认值	备注
minor_freeze_times	5	设置转储次数，性能测试时设置为 100。 /data/log1 盘很小时不宜太大。
data_disk_usage_limit_percentage	90	设置 block_file 默认占 /data/1 大小，仅影响新增的 OBCServer 节点。 /data /1 空间很大时或者 /data/1 和 /data/log1 共用同一块设备时调小这个值为 50，以为 /data/1 腾出更多的可用空间。
clog_transport_compress_all	False	设置 clog 在节点之间传输时是否启用压缩协议，会增加主机 cpu 压力，但节省带宽。跨机房测试性能时带宽不宽裕的时候开启这个参数。
clog_disk_usage_limit_percentage	95	设置 clog 盘空间满的标准，默认 /data/log1 使用率到 95 的时候空间判断为满，此后节点掉线。可以临时调大这个值让节点恢复正常，然后解决空间问题后再调整回去。
enable_merge_by_turn	True	设置是否开启轮转合并。默认开启时，在合并之前会将要合并的 ZONE 中的分区主副本都切换到其他 ZONE。建议设置为 False。这样合并之前不会触发有主切换（switchover）。
major_freeze_duty_time	02:00	设置大合并的默认时间。通常是凌晨低峰期。建议跟业务批处理、备份时间尽可能的错开。
data_copy_concurrency	20	数据负载均衡时集群层面同时副本复制的并发数。节点 CPU 很多时或者追求数据负载均衡速度时可以调高这个参数。
server_data_copy_out_concurrency	2	数据负载均衡时该节点内部同时副本复制出去的并发数。节点 CPU 很多时或者追求数据负载均衡速度时可以调高这个参数。不能超过集群层面副本复制的并发数。
server_data_copy_in_concurrency	2	数据负载均衡时该节点内部同时副本复制进来的并发数。节点 CPU 很多时或者追求数据负载均衡速度时可以调高这个参数。不能超过集群层面副本复制的并发数。
enable_rebalance	True	设置是否开启自动负载均衡。默认是开启。当数据分布符合期望时，如果数据均衡动作频繁发生影响业务使用时就关闭这个负载均衡机制。
resource_hard_limit	100	设置资源分配百分比上限。生产环境是 100%，实打实的分配。测试环境可

		以将这个值超过 100, 设置为 150 或 200. 即资源超卖. 不过资源超卖时还是有可能分配不出资源。
resource_soft_limit	50	节点资源分配比例是否满的标志. 即当一个节点资源利用率高于 50% 的时候, OB 在后期分配资源的时候会优先考虑其他节点资源分配比例低于 50% 的节点。如果所有节点都高于这个值, 那就取分配比例最低的节点。
enable_auto_leader_switch	True	是否开启自动切主, 跟开启负载均衡搭配使用。
freeze_trigger_percentage	70	增量内存使用比例出发 minor freeze 的阈值。性能测试时, 增量内存资源紧张时可以调低这个比例, 尽可能的出发转储, 释放更多的增量内存。
memstore_limit_percentage	50	增量内存占总内存的比例。总内存不大时, 可以稍微调大这个内存比例。
large_query_threshold	100ms	SQL 大查询的阈值。根据业务 SQL 平均执行时间而定。大查询会使用单独的 CPU 队列, 所以大查询比例应该很小。
large_query_worker_percentage	30	SQL 大查询的 CPU 队列占总的 CPU 资源比例。如果没有或者很少有大查询, 这个比例可以调低。
memory_chunk_cache_size	0M	内部内存分配单位。默认为 0 表示自适应。如果确认是批量结算业务, 有大量大事务, 则调大这个值大于实际内存大小。
memory_limit_percentage	80	OB 内存占用主机节点内存比例。主机内存不高时, 可以调高这个比例到 90; 如果主机内存很高, 超过 512G 时, 可以调低这个内存比例为 50~70 等。
enable_syslog_recycle	False	设置 OBServer 运行日志是否循环利用。/home/admin 空间不大时建议为 True。
max_syslog_file_count	0	设置 OBServer 运行日志最多保留多少个就循环利用。默认为 0 就是没有限制。/home/admin 空间不大时建议为 10~100。
syslog_level	INFO	设置 OBServer 运行日志的日志级别 (DEBUG/INFO/WARN/USER_ERROR/ERROR 等)。/home/admin/ 空间很小的时候可以改为 WARN。但是不利于通过日志诊断问题。
obconfig_url		这是 OCP API 地址。如果有 OCP 部署的 OB 集群, OCP 会自动设置这个。如果是独立部署的 OB 集群, 可以模拟一个 OCP API, 然后在这里设置。可选功能。
enable_perf_event	True	是否开启集群的性能指标信息收集。影响是否展示 QPS/TPS/QPS RT/TPS RT 等多项指标。
enable_sql_audit	True	是否开启 SQL 全量日志功能, 影响是否查询全量 SQL、TOP SQL 和满 SQL 功能。开启带来的额外成本很低, 建议性能分析的时候开启。极端性能测试的时候关闭这个参数。
sql_audit_queue_size	10000000	设置 SQL 全量日志存储队列的长度, 队列是 FIFO, 长度影响容量。
sql_audit_memory_limit	3G	设置 SQL 全量日志存储队列的大小, 队列是 FIFO, 大小就是容量。
system_memory	50G	设置 OB 内部保留的内存大小。如果主机内存很小时, 可以把这个调小至 10G。

### 6.1.4. 业务实例参数建议

业务实例的参数也可以单独设置, 但是只能在 sys 实例里通过命令修改, 修改时指定租户名 (实例名)。

```
ALTER SYSTEM SET enable_sql_audit=TRUE tenant='ob_bmsql';
```

参数名	默认值	备注
writing_throttling_trigger_percentage	100	设置租户增量内存写入限速触发条件 (内存使用比例), 调整为 90。

### 6.1.5. 业务实例变量建议

参数名	默认值	备注
ob_query_timeout	10000000	语句执行默认超时时间, 单位毫秒。为避免 SQL 执行报超时错误, 建议改为非常大, 如 9999999999999999。
ob_trx_idle_timeout	120000000	事务默认空闲时间, 单位毫秒。空闲时间超过这个值, 会话被 kill。



		，锁释放，事务回滚。建议改为非常大，如9999999999999999
ob_trx_timeout	100000000	事务默认最长时间，单位毫秒。事务未提交时间超过这个值，事务会报错处于“超时状态”，锁释放。需要会话发起 rollback 命令才可以继续使用。
recyclebin	off	是否开启回收站，默认开启。如果有频繁的 DDL（如 truncate table），建议关闭这个变量。

## 6.2. 性能监控经验

### 6.2.1. 命令行下性能监控工具 dooba

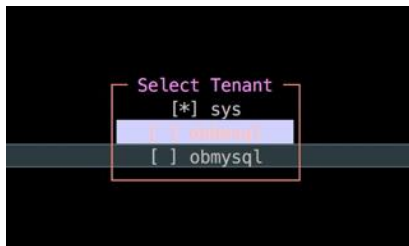
```
python dooba.py -h 127.1 -u root@sys#obdemo -P 2883 -p123456
```

帮助文档：



常用功能按键是：c 2 3 tab d

使用 c 选择要监控的实例



选 2 看实例层面的性能指标：

Gallery

Shown: 6 / Valid: 6 / Total: 6

TIME	TENANT	SQL COUNT							SQL RT							RPC			MEMORY(T)			IOPS				
	TIDWAT	SEL.	INS.	UPD.	DEL.	REP.	CMT.	ROL.	SEL.	INS.	UPD.	DEL.	REP.	CMT.	FAIL	NET	FRAME	ACTIVE	TOTAL	PCT.	SES.	IOR	IOR-SZ	IOW	IOW-SZ	
13:59:37	1001	0	0	0	0	0	15	0	0.00	0.00	0.00	0.00	0.00	0.39	0	0.00	0.00	0	536.00M	10.5%	0	0	0	0	0	
13:59:36	1001	0	0	0	0	0	16	0	0.00	0.00	0.00	0.00	0.00	0.31	0	0.00	0.00	0	536.00M	10.5%	0	0	0	0	0	
13:59:35	1001	0	0	0	0	0	16	0	0.00	0.00	0.00	0.00	0.00	0.31	0	0.00	0.00	0	536.00M	10.5%	0	0	0	0	0	
13:59:34	1001	0	0	0	0	0	16	0	0.00	0.00	0.00	0.00	0.00	0.43	0	0.00	0.00	0	536.00M	10.5%	0	0	0	0	0	
13:59:33	1001	0	0	0	0	0	17	0	0.00	0.00	0.00	0.00	0.00	0.35	0	0.00	0.00	0	536.00M	10.5%	0	0	0	0	0	
13:59:32	1001	0	0	0	0	0	15	0	0.00	0.00	0.00	0.00	0.00	0.13	0	0.00	0.00	0	536.00M	10.5%	0	0	0	0	0	
13:59:31	1001	0	0	0	0	0	16	0	0.00	0.00	0.00	0.00	0.00	0.25	0	0.00	0.00	0	536.00M	10.5%	0	0	0	0	0	
13:59:30	1001	0	0	0	0	0	15	0	0.00	0.00	0.00	0.00	0.00	0.33	0	0.00	0.00	0	536.00M	10.5%	0	0	0	0	0	
13:59:29	1001	0	0	0	0	0	16	0	0.00	0.00	0.00	0.00	0.00	0.43	0	0.00	0.00	0	536.00M	10.5%	0	0	0	0	0	
13:59:28	1001	0	0	0	0	0	16	0	0.00	0.00	0.00	0.00	0.00	0.37	0	0.00	0.00	0	536.00M	10.5%	0	0	0	0	0	
13:59:27	1001	0	0	0	0	0	16	0	0.00	0.00	0.00	0.00	0.00	0.19	0	0.00	0.00	0	536.00M	10.5%	0	0	0	0	0	
13:59:26	1001	0	0	0	0	0	16	0	0.00	0.00	0.00	0.00	0.00	0.31	0	0.00	0.00	0	536.00M	10.5%	0	0	0	0	0	
13:59:25	1001	0	0	0	0	0	16	0	0.00	0.00	0.00	0.00	0.00	0.43	0	0.00	0.00	0	536.00M	10.5%	0	0	0	0	0	
13:59:24	1001	0	0	0	0	0	16	0	0.00	0.00	0.00	0.00	0.00	0.19	0	0.00	0.00	0	536.00M	10.5%	0	0	0	0	0	
13:59:23	1001	0	0	0	0	0	16	0	0.00	0.00	0.00	0.00	0.00	0.31	0	0.00	0.00	0	536.00M	10.5%	0	0	0	0	0	

分类	指标名	备注
	TIME	当前时间。第一行是60 秒内平均值。
	TENANAT	租户 ID
SQL COUNT SQL QPS 信息	SEL.	平均每秒 SELECT 次数。
	INS.	平均每秒 INSERT 次数。
	UPD.	平均每秒 UPDATE 次数。
	DEL.	平均每秒 DELETE 次数。
	REP.	平均每秒 REPLACE 次数。
	CMT.	平均每秒 COMMIT 次数。
	ROL.	平均每秒 ROLLBACK 次数。
SQL RT SQL 执行耗时	SEL.	平均每次 SELECT 耗时（单位 ms，下同）。
	INS.	平均每次 INSERT 耗时。
	UPD.	平均每次 UPDATE 耗时。
	DEL.	平均每次 DELETE 耗时。
	REP.	平均每次 REPLACE 耗时。
	CMT.	平均每次 COMMIT 耗时。
MEMORY	ΔACTIVE	平均每秒增量内存的消耗量（为负数表示释放的增量内存大于消耗的增量内存）
	TOTAL	增量内存当前使用的量
	PCT.	增量内存当前使用的量占最大可用增量内存的比例。
IOPS	SES.	实例所有节点的会话数。
	IOR	实例平均每秒读 IO 次数。
	IOR-SZ	实例平均每秒读 IO 的大小。
	IOW	实例平均每秒写 IO 次数。
	IOW-SZ	实例平均每秒写 IO 的大小。

选 3 ，看实例各个节点的性能信息。

SQLPage

zone1:11.166.87.5:28822020-05-26 16:33:27

Active Sess: 0CPU: 0.1%Cache-BI Hit: 0.0%Cache-Blk Hit: 0.0%Cache-Loc Hit: 0.0%Cache-Row Hit: 0.0%

IO-R Cnt: 0IO-R Size: 0IO-W Cnt: 0IO-W Size: 0

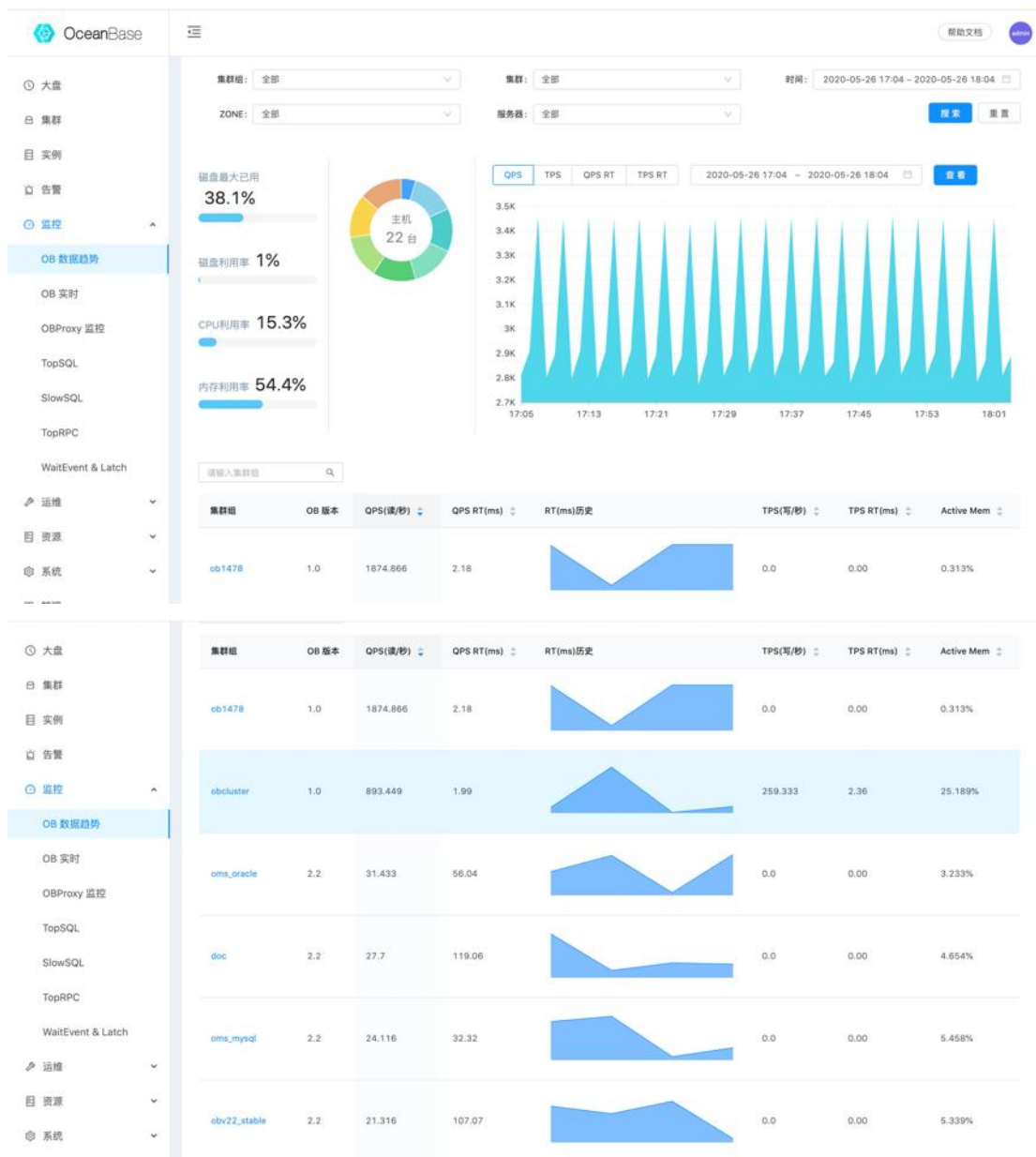
SSC	SSRT	SIC	SIRT	SUC	SURT	SDC	SDRT	SRC	SRRT	TCC	TCRT	SLC	SRC
0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	16	0.22	0	0
0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	15	0.44	0	0
0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	16	0.24	0	0
0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	17	0.33	0	0
0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	16	0.29	0	0
0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	17	0.50	0	0
0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	16	0.18	0	0
0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	17	0.28	0	0
0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	16	0.18	0	0
0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	16	0.47	0	0
0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	16	0.22	0	0
0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	16	0.24	0	0
0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	16	0.28	0	0
0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	17	0.28	0	0
0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	16	0.12	0	0
0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	16	0.28	0	0
0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	16	0.35	0	0
0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	16	0.33	0	0
0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	16	0.24	0	0

指标名	备注
SSC	该节点平均每秒 SELECT 次数。
SIC	该节点平均每秒 INSERT 次数。
SUC	该节点平均每秒 UPDATE 次数。
SDC	该节点平均每秒 DELETE 次数。
SRC	该节点平均每秒 REPLACE 次数。
TCC	该节点平均每秒 COMMIT 次数。
SLC	该节点平均每秒本地 SQL 执行次数。
SRC	该节点平均每秒远程 SQL 执行次数。
SSRT	该节点平均每次 SELECT 耗时（单位 ms，下同）。
SIRT	该节点平均每次 INSERT 耗时。
SURT	该节点平均每次 UPDATE 耗时。
SDRT	该节点平均每次 DELETE 耗时。
SRRT	该节点平均每次 REPLACE 耗时。
TCRT	该节点平均每次 COMMIT 耗时。

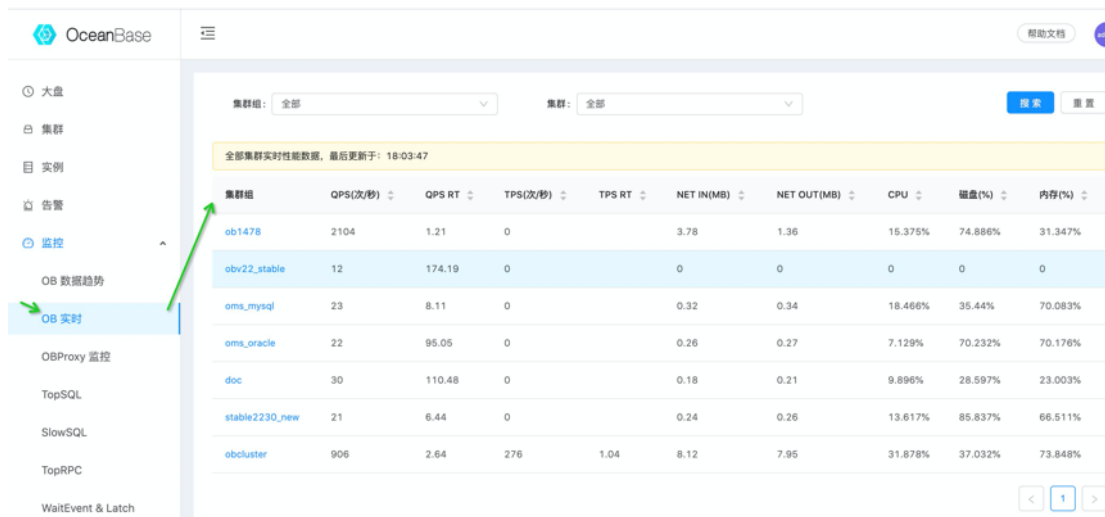
## 6.2.2. OCP 里性能监控

主要看【OB 数据趋势】和【OB 实时】监控。操作都比较简单就不详细解释了。可以看集群、实例、主机三个维度的各项监控指标。

OB 趋势监控：



## OB 实时监控:



## 6.3. 日志使用经验

### 6.3.1. 根据 OCP 日志排查问题

### 6.3.2. 根据 OBServer 日志排查问题

OBServer 进程日志在 admin 用户下的 `/home/admin/oceanbase/log` 目录下。

日志文件有两类：`observer.log` 和 `observer.log.wf`。前者包含所有级别的日志（默认是 INFO 级别及其以上），后者包含 WARN 级别及其以上的日志（如 ERROR 级别）。

`observer` 的日志内容是给内核开发用于分析问题的，不怎么适合用户看。所以通常用户不用看这个日志。如果 `observer` 进程异常退出，可以去这个日志里搜索一些 ERROR 级别的日志，并查看相邻的日志的提示信息，从里面找寻非程序语言的那种有意义的日志。

类似下面这个 ERROR 日志可以忽略，没有影响。

```
[2020-05-26 11:03:52.027994] ERROR [LIB] pidfile_test (utility.cpp:1168) [6310][0][Y0-0000000000000000] [lt=0] fid file doesn't exist(pidfile="run/observer.pid") BACKTRACE:0x726996a 0x7216ea6 0x453877 0x3e762f6 0x726d54c 0x3fc3a46 0x2b527c84f505 0x40240c5
```

## 6.4. 常用 SQL

### 6.4.1. 集群事件

▣ 集群事件查看

```
SELECT DATE_FORMAT(gmt_create, '%b%d %H:%i:%s') gmt_create_, module, event, name1,
value1, name2, value2, rs_svr_ip
FROM __all_rootservice_event_history
WHERE 1 = 1
ORDER BY gmt_create DESC
LIMIT 20;
```

## 6.4.2. 资源管理

- 查看集群总的资源使用情况

```
select a.zone,concat(a.svr_ip,':',a.svr_port) observer, cpu_total, (cpu_total-cpu_assigned)
cpu_free, round(mem_total/1024/1024/1024) mem_total_gb, round((mem_total-
mem_assigned)/1024/1024/1024) mem_free_gb, usec_to_time(b.last_offline_time)
last_offline_time, usec_to_time(b.start_service_time) start_service_time, b.status
from __all_virtual_server_stat a join __all_server b on (a.svr_ip=b.svr_ip and
a.svr_port=b.svr_port)
order by a.zone, a.svr_ip
;
```

- 查看租户资源分配细节

```
select t1.name resource_pool_name, t2.`name` unit_config_name, t2.max_cpu, t2.min_cpu,
round(t2.max_memory/1024/1024/1024) max_mem_gb,
round(t2.min_memory/1024/1024/1024) min_mem_gb, t3.unit_id, t3.zone,
concat(t3.svr_ip,':',t3.svr_port) observer,t4.tenant_id, t4.tenant_name
from __all_resource_pool t1 join __all_unit_config t2 on (t1.unit_config_id=t2.unit_config_id)
join __all_unit t3 on (t1.`resource_pool_id` = t3.`resource_pool_id`)
left join __all_tenant t4 on (t1.tenant_id=t4.tenant_id)
order by t1.`resource_pool_id`, t2.`unit_config_id`, t3.unit_id
;
```

## 6.4.3. 增量内存管理

- 查看增量内存消耗速度

```
SELECT tenant_id, ip, round(active/1024/1024) active_mb, round(total/1024/1024) total_mb,
round(freeze_trigger/1024/1024) freeze_trg_mb, round(mem_limit/1024/1024)
mem_limit_mb
, freeze_cnt , round((active/freeze_trigger),2) freeze_pct, round(total/mem_limit, 2)
mem_usage
FROM `gv$memstore`
WHERE tenant_id IN (1001)
ORDER BY tenant_id, ip;
```

- 查看冻结合并进度

```
SELECT
ZONE,svr_ip,major_version,min_version,ss_store_count,merged_ss_store_count,modified_ss_store_count,merge_start_time,merge_finish_time,merge_process
FROM __all_virtual_partition_sstable_image_info;
```

## 6.4.4. 分区管理

- 查看租户的所有分区的主副本位置分布

```
SELECT
t1.tenant_id,t1.tenant_name,t2.database_name,t3.table_id,t3.table_name,t3.tablegroup_id,t3.partition_num,t4.partition_id,t4.zone,t4.svr_ip,t4.role, round(t4.data_size/1024/1024) data_size_mb
from `gv$tenant` t1
  join `gv$database` t2 on (t1.tenant_id = t2.tenant_id)
  join gv$table t3 on (t2.tenant_id = t3.tenant_id and t2.database_id = t3.database_id
and t3.index_type = 0)
  left join `__all_virtual_meta_table` t4 on (t2.tenant_id = t4.tenant_id and ( t3.table_id =
t4.table_id or t3.tablegroup_id = t4.table_id ) and t4.role in (1))
where t1.tenant_id = 1001
order by t3.tablegroup_id, t4.partition_id, t3.table_name ;
```

## 6.4.5. SQL 审计

- 查看所有 SQL 运行日志

```
SELECT /*+ read_consistency(weak) ob_querytimeout(100000000) */
substr(usec_to_time(request_time),1,19) request_time_, s.svr_ip, s.client_ip, s.sid,s.tenant_id,
s.tenant_name, s.user_name, s.db_name, s.query_sql, s.affected_rows, s.return_rows, s.ret_code,
s.event, s.elapsed_time, s.queue_time, s.execute_time,
round(s.request_memory_used/1024/1024,2) req_mem_mb, plan_type, is_executor_rpc,
is_inner_sql
FROM gv$sql_audit s
WHERE s.tenant_id=1001 and user_name='testuser' and svr_ip in ('11.166.84.78')
ORDER BY request_time DESC
LIMIT 100;
```

