



# OceanBase 0.4.1

## 安装指南

文档版本：01

发布日期：2013.10.30

支付宝（中国）网络技术有限公司 OceanBase 团队

# 前言

## 概述

本文档主要介绍OceanBase数据库的安装流程和安装方法，可以帮助安装工程师完成OceanBase数据库的安装。

## 读者对象

本文档主要适用于：

- 安装工程师。
- 数据库管理工程师。

## 通用约定

在本文档中可能出现下列各式，它们所代表的含义如下。

格式	说明
警告	表示可能导致设备损坏、数据丢失或不可预知的结果。
注意	表示可能导致设备性能降低、服务不可用。
小窍门	可以帮助您解决某个问题或节省您的时间。
说明	表示正文的附加信息，是对正文的强调和补充。
宋体	表示正文。
<b>粗体</b>	表示命令行中的关键字（命令中保持不变、必须照输的部分）或者正文中强调的内容。
斜体	用于变量输入。
{ a   b   ... }	表示从两个或多个选项中选取一个。
[ ]	表示用“[ ]”括起来的部分在命令配置时是可选的。

## 修订记录

修改记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本。

## 联系我们

如果您有任何疑问或是想了解 OceanBase 的最新开源动态消息，请联系我们：

支付宝（中国）网络技术有限公司·OceanBase 团队

地址：杭州市万塘路 18 号黄龙时代广场 B 座；邮编：310099

北京市朝阳区东三环中路 1 号环球金融中心西塔 14 层；邮编：100020

邮箱：[alipay-oceanbase-support@list.alibaba-inc.com](mailto:alipay-oceanbase-support@list.alibaba-inc.com)

新浪微博：<http://weibo.com/u/2356115944>

技术交流群（阿里旺旺）：853923637

# 目 录

---

1 安装前须知.....	- 1 -
1.1 产品简介 .....	- 1 -
1.2 部署模式 .....	- 2 -
1.3 软硬件要求 .....	- 2 -
1.4 安装规划 .....	- 3 -
1.4.1 服务器规划 .....	- 3 -
1.4.2 目录规划 .....	- 4 -
1.4.3 磁盘挂载点规划 .....	- 5 -
1.5 安装流程 .....	- 6 -
2 准备安装环境.....	- 7 -
2.1 修改操作系统配置 .....	- 7 -
2.1.1 设置网卡名称 .....	- 7 -
2.1.2 配置“sysctl.conf” .....	- 8 -
2.1.3 配置 NTP 时钟同步 .....	- 8 -
2.2 创建安装用户 .....	- 9 -
2.3 检查 gcc 版本 .....	- 10 -
2.4 配置环境变量 .....	- 10 -
2.5 配置免登录 .....	- 11 -
2.6 创建数据磁盘挂载点 .....	- 11 -
3 采用 RPM 包安装.....	- 13 -
3.1 下载安装包 .....	- 13 -
3.2 安装动态库 .....	- 14 -
3.3 安装 OceanBase 软件.....	- 14 -
4 采用源码安装.....	- 15 -
4.1 下载安装包 .....	- 15 -
4.2 安装动态库 .....	- 16 -
4.2.1 安装 liblz2.....	- 16 -
4.2.2 安装 Snappy .....	- 17 -
4.2.3 安装工具组 .....	- 17 -

4.2.4 安装 libnuma .....	- 19 -
4.2.5 安装 libaio .....	- 19 -
4.2.6 安装 gtest 和 gmock（可选） .....	- 19 -
4.2.7 安装其他库 .....	- 20 -
4.3 安装 tbsys 和 tbnet .....	- 21 -
4.4 安装 libeasy .....	- 21 -
4.5 安装 OceanBase 软件 .....	- 22 -
4.6 创建各 Server 所需目录 .....	- 22 -
5 启动 OceanBase .....	- 25 -
6 安装 MySQL 客户端 .....	- 32 -
7 配置 OceanBase .....	- 33 -
8 FAQ .....	- 39 -
8.1 启动 UpdateServer 时报错 .....	- 39 -
8.2 安装 gcc 时编译出错 .....	- 40 -
9 附录 .....	- 41 -
9.1 常用操作 .....	- 41 -
9.1.1 启动服务 .....	- 41 -
9.1.2 停止服务 .....	- 44 -
9.1.3 重新启动 .....	- 44 -
9.1.4 一键脚本操作 .....	- 45 -
9.1.5 卸载 .....	- 45 -
9.2 安装 gcc 4.1.2 .....	- 46 -
9.3 内部表参数说明 .....	- 47 -
9.3.1 __first_tablet_entry .....	- 47 -
9.3.2 __all_all_column .....	- 49 -
9.3.3 __all_join_info .....	- 50 -
9.3.4 __all_client .....	- 51 -
9.3.5 __all_cluster .....	- 52 -
9.3.6 __all_server .....	- 52 -
9.3.7 __all_server_stat .....	- 53 -
9.3.8 __all_sys_config .....	- 54 -
9.3.9 __all_sys_config_stat .....	- 55 -
9.3.10 __all_sys_param .....	- 56 -

9.3.11 __all_sys_stat .....	- 57 -
9.3.12 __all_table_privilege .....	- 58 -
9.3.13 __all_trigger_event.....	- 59 -
9.3.14 __all_user .....	- 60 -
9.4 配置参数说明 .....	- 61 -
9.4.1 RootServer 配置参数 .....	- 61 -
9.4.2 UpdateServer 配置参数.....	- 68 -
9.4.3 MergeServer 配置参数 .....	- 78 -
9.4.4 ChunkServer 配置参数 .....	- 81 -

# 1 安装前须知

介绍了安装 OceanBase 数据库前您需要了解的基本信息。

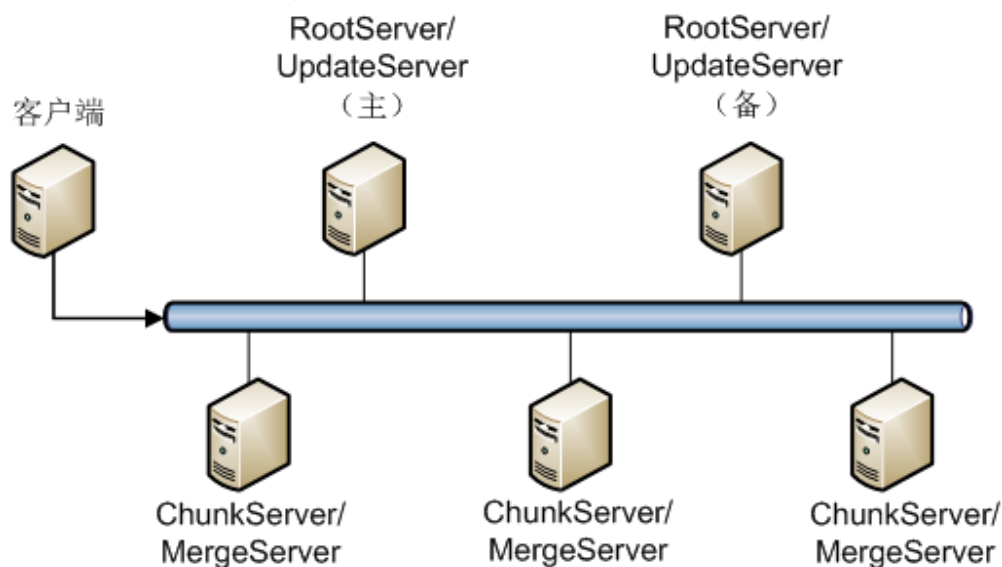
## 1.1 产品简介

OceanBase 数据库是阿里集团研发的可扩展的分布式关系数据库，实现了数千亿条记录、数百淘宝数据上的跨行跨表事务，主要支持收藏夹、直通车报表、天猫评价等 OLTP 和 OLAP 在线业务。

OceanBase 数据库的数据主要可以分为基准数据和增量数据。基准数据是只读数据，增量数据是需要修改更新的数据。OceanBase 数据库内部通过合并操作定期将增量数据融合到基准数据中。

OceanBase 数据库组网如[图 1-1](#)所示。

图 1-1 OceanBase 数据库组网



- **RootServer**  
主控服务器，主要进行集群管理、数据分布和副本管理。
- **UpdateServer**  
更新服务器，是集群中唯一能够接受写入的模块，存储每日更新的增量数据。
- **ChunkServer**  
基准数据服务器，存储 OceanBase 数据库中的基准数据，提供数据读取服务、执行定期合并以及数据分发。

- **MergeServer**  
合并服务器，主要提供协议解析、SQL 解析、请求转发、结果合并和多表操作等功能。  
OceanBase 集群内部还有一个特殊的 MergeServer 进程，即 Listener，一般与 RootServer 合设。负责从集群的内部表中查询主备集群的流量分布信息和所有的其他 MergeServer 的地址列表。
- **客户端**  
客户端中存放了多个集群的 RootServer 地址列表，并根据集群的流量分配比例将读写操作发往不同的集群，详细请参见《OceanBase 0.4.1 客户端 用户指南》。

## 1.2 部署模式

OceanBase 数据库部署模式灵活，可满足用户多种需求。

OceanBase 数据库推荐的部署模式说明如[表 1-1](#)所示。

**注意：**由于 RootServer 服务器中需要启动 Listener 服务，该进程为特殊的 MergeServer 进程。因此请勿将 RootServer 和 MergeServer 部署在同一台机器。

表 1-1 部署模式

部署模式	说明
RootServer 和 UpdateServer 合设	可采用主备双机模式。RootServer 主备由 HA 决定，UpdateServer 主备由 RootServer 选举决定。
ChunkServer 和 MergeServer 合设	<ul style="list-style-type: none"> <li>• ChunkServer 存储 OceanBase 数据库的基准数据。基准数据建议存储两份或者三份。可根据需求部署多台。</li> <li>• MergeServer 对 UpdateServer 上的动态数据和 ChunkServer 上的静态数据进行合并。可根据需求部署多台。</li> </ul>

OceanBase 数据库各 Server 的安装方式相同，启动方式不同。

例如，现需部署服务器 A 为 RootServer、UpdateServer；服务器 B 为 ChunkServer 和 MergeServer。只需在服务 A 和服务 B 中分别安装 OceanBase 数据库软件，然后在服务器 A 中启动 RootServer、UpdateServer；服务器 B 中启动 ChunkServer 和 MergeServer。

OceanBase 数据库还支持主备集群部署，即主备 OceanBase 数据库中分别含有 RootServer、UpdateServer、ChunkServer 和 MergeServer 服务。

## 1.3 软硬件要求



OceanBase 数据库服务器最低配置要求如[表 1-2](#)所示。

**表 1-2 最低配置**

模块	服务器数量	操作系统	CPU	内存	磁盘	其他
RootServer/ UpdateServer/ Listener	2 台（主备 RootServer）	Red Hat Enterprise Linux Server release 6.2 (Santiago) 64bit 内核 2.6.32 x86_64	每台服 务器 1 颗 4 核 CPU	48GB/ 台	8 块 *160GB/台	1 块 SAS 卡 或 1 块 RAID 卡，1 个千兆 口
ChunkServer/ MergeServer	3 台					

OceanBase 数据库服务器推荐配置要求如[表 1-3](#)所示。

**表 1-3 推荐配置**

模块	服务器数量	操作系统	CPU	内存	磁盘	其他
RootServer/ UpdateServer/ Listener	2 台（主备 RootServer）	Red Hat Enterprise Linux Server release 6.2 (Santiago) 64bit 内核 2.6.32 x86_64	每台服 务器 2 颗 6 核 CPU	192GB/ 台	300GB*12 块 (SSD) / 台	1 块 RAID 卡 1G 缓 存，2 个 万兆口
ChunkServer/ MergeServer	3 台		每台服 务器 1 颗 6 核 CPU	48GB/ 台	300GB*10 块 (SSD) / 台	1 块 SAS 卡，2 个 千兆口

当搭建主备集群时，主备 OceanBase 的各服务器均需满足以上配置要求。

## 1.4 安装规划

安装规划主要包括服务器规划、目录规划和磁盘挂载点规划。

本文档以单集群单 RootServer/UpdateServer 为例，简单介绍其安装方法，实际场景中请严格遵循“1.3 软硬件要求”。

### 1.4.1 服务器规划

OceanBase 数据库服务器规划如[表 1-4](#)所示。

表 1-4 服务器规划

规划项	规划
服务器 IP	<ul style="list-style-type: none"> <li>• RootServer/UpdateServer: 10.10.10.2</li> <li>• ChunkServer/MergeServer: 10.10.10.4, 10.10.10.5</li> </ul>
网卡名称	均为“eth0”。
端口	<ul style="list-style-type: none"> <li>• RootServer: 服务端口 2500。</li> <li>• UpdateServer: 服务端口 2700; 合并操作端口 2701。</li> <li>• ChunkServer: 服务端口 2600。</li> <li>• MergeServer: 服务端口 2800, MySQL 协议端口 2880。</li> <li>• Listener: 服务端口 2828, MySQL 协议端口 2828, 请勿修改。</li> </ul>
安装用户	admin <i>注意:</i> 采用 RPM 安装时, 安装用户必须为“admin”。
用户密码	Abc@123
安装目录	/home/admin/oceanbase
集群 ID	1
App 名称	obtest

## 1.4.2 目录规划

OceanBase 各 Server 的数据存放目录规划如[表 1-5](#)所示。

表 1-5 目录规划

规划项	规划
RootServer	<ul style="list-style-type: none"> <li>• 数据目录: /home/admin/oceanbase/data/log/rs</li> <li>• 日志目录: /home/admin/oceanbase/data/log/rs_commitlog</li> </ul>

规划项	规划
UpdateServer	<p>数据目录：</p> <ul style="list-style-type: none"> <li>• /home/admin/oceanbase/data/ups_data/raid0/store0</li> <li>• /home/admin/oceanbase/data/ups_data/raid0/store1</li> <li>• /home/admin/oceanbase/data/ups_data/raid1/store0</li> <li>• /home/admin/oceanbase/data/ups_data/raid1/store1</li> <li>• /home/admin/oceanbase/data/ups_data/raid2/store0</li> <li>• /home/admin/oceanbase/data/ups_data/raid2/store1</li> <li>• /home/admin/oceanbase/data/ups_data/raid3/store0</li> <li>• /home/admin/oceanbase/data/ups_data/raid3/store1</li> </ul> <p>日志目录： /home/admin/oceanbase/data/log/ups_commitlog</p>
ChunkServer	<p>数据目录：</p> <ul style="list-style-type: none"> <li>• /home/admin/oceanbase/data/1</li> <li>• /home/admin/oceanbase/data/2</li> <li>• .....</li> <li>• /home/admin/oceanbase/data/8</li> </ul>

### 1.4.3 磁盘挂载点规划

OceanBase 的 ChunkServer 和 UpdateServer 分别需要存储静态数据和动态数据，建议使用单独的磁盘进行数据存储。磁盘挂载点的规划如[表 1-6](#)所示。

**表 1-6 磁盘挂载点规划**

规划项	规划
ChunkServer	<p>数据存放磁盘的挂载点：</p> <ul style="list-style-type: none"> <li>• /data/1</li> <li>• /data/2</li> <li>• .....</li> <li>• /data/8</li> </ul>

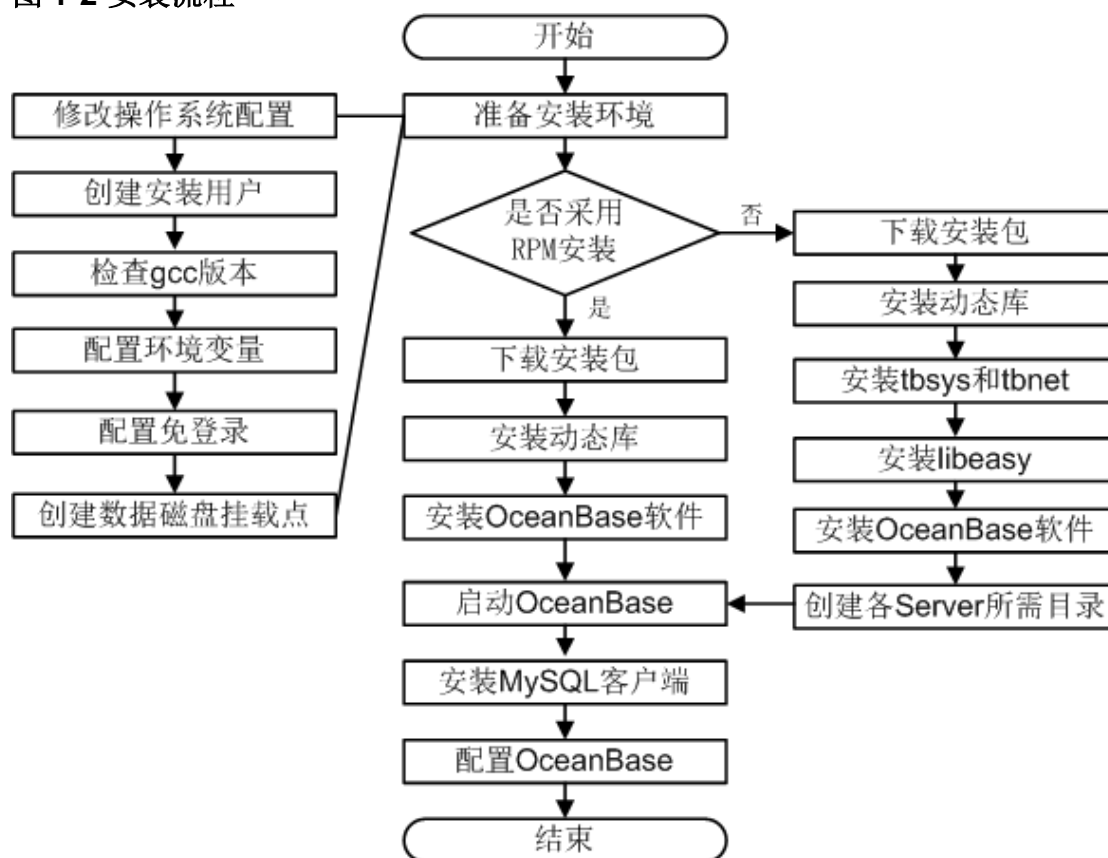
规划项	规划
UpdateServer	数据存放磁盘的挂载点： <ul style="list-style-type: none"> <li>• /data/1</li> <li>• /data/2</li> <li>• .....</li> <li>• /data/8</li> </ul>

## 1.5 安装流程

主要介绍 OceanBase 数据库的安装流程，有助于您更好地完成安装任务。

OceanBase 数据库安装流程如[图 1-2](#)所示。

图 1-2 安装流程



## 2 准备安装环境

---

在安装 OceanBase 前，请先根据磁盘规划和服务器规划，在各服务器中分别完成修改操作系统配置、创建安装用户、检查 gcc 版本、配置环境变量、配置免登录和创建数据磁盘挂载点，否则会造成安装失败。

### 2.1 修改操作系统配置

为保证 OceanBase 正常安装，我们需要对所有 OceanBase 服务器进行设置网卡名称、配置“sysctl.conf”和配置 NTP 时钟同步等操作。

#### 2.1.1 设置网卡名称

采用 RPM 包安装时，需要配置和使用一键脚本，要求各 Server 服务启动的网卡名称必须相同。

采用源码安装时，您可以使用 **ifconfig** 命令查看并记录网卡名称，并在启动 OceanBase 各 Server 时通过“-i”参数进行指定，但是为了便于管理和记忆，建议您修改成相同网卡名称。

所有 OceanBase 服务器的网卡名称设置为“eth0”的操作步骤如下：

1. 以 **root** 用户登录各 OceanBase 服务器。
2. 使用 **vi** 编辑器，修改“/etc/udev/rules.d/70-persistent-net.rules”文件。

```
# PCI device 0x1022:0x2000 (pcnet32)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*",
ATTR{address}=="08:00:27:9e:ee:33", ATTR{type}=="1", KERNEL=="eth*", NAME="eth0"
```

3. 使用 **vi** 编辑器，修改“/etc/sysconfig/network-scripts/ifcfg-eth0”文件。

```
DEVICE="eth0"
BOOTPROTO=static
NM_CONTROLLED="yes"
ONBOOT="yes"
TYPE=Ethernet
NETMASK=255.255.255.0
IPADDR=10.10.10.2
```

4. 执行 **reboot** 命令，重新启动服务器。
5. 执行 **ifconfig** 命令，查看网卡名称。
6. 参考“步骤 1”至“步骤 5”，将所有 OceanBase 服务器的网卡名称设置为“eth0”。

## 2.1.2 配置“sysctl.conf”

为保证 OceanBase 正常运行，请在安装 OceanBase 前修改“sysctl.conf”配置。  
所有 OceanBase 服务器的“sysctl.conf”配置过程如下：

1. 以 **root** 用户登录各 OceanBase 服务器。
2. 使用 **vi** 编辑器，在“/etc/sysctl.conf”文件末尾添加以下内容。

```
# for oceanbase
net.core.somaxconn = 2048
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 = 3500 65535
net.ipv4.ip_forward = 0
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.default.accept_source_route = 0
net.ipv4.tcp_syncookies = 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_max_syn_backlog = 16384
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_tw_recycle = 1

vm.swappiness = 0
```

3. 执行以下命令，使修改配置生效。  
**/sbin/sysctl -p /etc/sysctl.conf**

## 2.1.3 配置 NTP 时钟同步

为保证 OceanBase 正常运行，请在安装 OceanBase 前配置 NTP 时钟同步，使所有 OceanBase 服务器时间一致。

### \* 前提条件

在配置 NTP 时钟同步前，请关闭所有 OceanBase 服务器的防火墙。

关闭防火墙命令为：**service iptables stop**

### \* 开启 NTP Server 服务

如果已经存在 NTP Server 服务器，则可跳过本小节。否则，请在 OceanBase 服务器中任意选择一台，作为 NTP Server 服务器，并开启 NTP Server 服务。

开启 NTP Server 服务操作步骤如下：

1. 以 **root** 用户登录 NTP Server 服务器。

2. 使用 **vi** 编辑器，修改“/etc/ntp.conf”，修改结果如黑体部分所示。

```
.....  
# Permit time synchronization with our time source, but do not  
# permit the source to query or modify the service on this system.  
#restrict default kod nomodify notrap nopeer noquery  
restrict default kod nomodify  
restrict -6 default kod nomodify notrap nopeer noquery  
.....
```

3. 执行以下命令，重启 NTP Server 服务。  
**/etc/init.d/ntpd restart**
4. 执行以下命令，查看是否启动 NTP Server 服务。  
**ps -ef |grep -i ntpd**
5. 执行以下命令，设置 NTP 服务开机自启动。  
**chkconfig ntpd on**

#### \* 开启 NTP Client 服务

NTP Server 服务启动后，请等待 5 分钟再开启 NTP Client 服务，否则会提示“no server suitable for synchronization found”错误。

假设 NTP Server 服务器 IP 为 10.10.10.2。所有 OceanBase 服务器开启 NTP Client 服务操作步骤如下：

1. 以 **root** 用户登录 OceanBase 服务器。
2. 执行以下命令，停止 NTP 服务。  
**/etc/init.d/ntpd stop**
3. 执行以下命令，同步 NTP Server 时间。  
**ntpdate 10.10.10.2**
4. 执行 **crontab -e** 命令，并添加以下内容，设置时间同步间隔为每 10 分钟一次。

```
0-59/10 * * * * /usr/sbin/ntpdate 10.10.10.2 && /sbin/hwclock -w
```

#### \* 验证 NTP 时间同步

验证 NTP 时间同步操作步骤如下：

1. 在开启 NTP Client 服务的 OceanBase 服务器中，执行以下命令，修改系统时间，并同步到硬件时钟。  
**date --set "10/22/2010 00:00:00" && /sbin/hwclock -w**
2. 一分钟后，执行以下命令，查看客户端时间，并比较 NTP Server 服务器的时间。若一致则表明时间同步配置成功。  
**date && hwclock --show**

## 2.2 创建安装用户

在所有 OceanBase 服务器中，创建 OceanBase 安装用户的操作步骤如下：

1. 以 **root** 用户分别登录各 OceanBase 服务器。

2. 执行如下命令，创建 OceanBase 的安装用户。  
**useradd -d /home/admin -s /bin/bash -m admin**
3. 执行如下命令，为用户“admin”设置密码。  
**passwd admin**
4. 您需要根据系统的提示输入两次密码“Abc@123”。
5. 为“admin”赋予“sudo”权限。
  - a. 执行以下命令，添加“/etc/sudoers”文件的写权限。  
**chmod u+w /etc/sudoers**
  - b. 使用 **vi** 编辑器，在“/etc/sudoers”文件中“root ALL= (ALL) ALL”后添加语句，如黑体部分所示。

```
root ALL= (ALL) ALL
admin ALL=(ALL) ALL
```
  - c. 执行以下命令，删除“/etc/sudoers”文件的写权限。  
**chmod u-w /etc/sudoers**

## 2.3 检查 gcc 版本

如果您采用 RPM 包安装，则可以跳过本小节。在采用源码安装 OceanBase 前，确认 gcc 版本，否则会造成编译失败：

- Red Hat 5: gcc 4.1.2
- Red Hat 6: gcc 4.1.2 或者 gcc 4.4.6

**说明：**您可以执行 **cat /etc/issue** 命令查看 Linux 版本号。

如果您的 gcc 版本不符合要求，请参考本手册的“8.2 安装 gcc 4.1.2”重新安装。

检查 gcc 版本的操作步骤如下：

1. 以 **root** 用户分别登录各 OceanBase 服务器。
2. 执行 **gcc --version** 命令，检查 gcc 版本，系统显示如下。

```
gcc (GCC) 4.1.2 20080704 (Red Hat 4.1.2-51)
Copyright (C) 2006 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR
PURPOSE.
```

## 2.4 配置环境变量

OceanBase 在运行时需要使用到动态库，因此安装 OceanBase 前需要配置环境变量，操作步骤如下：

1. 以 **admin** 用户分别登录各 OceanBase 服务器。
2. 用 **vi** 编辑器在“/home/admin/.bashrc”文件中，添加如下语句：  
**说明：**拷贝添加以下语句时，请删除#后的注释语句。其中“/home/admin/oceanbase”为安装目录，“/opt/taobao/java”为 Java 安装目录。



```
export TBLIB_ROOT=~/.tb-common-utils
export
LD_LIBRARY_PATH=/home/admin/oceanbase/lib:/usr/local/lib/libsnapappy.so:/usr/:/usr/lib:/usr/local/lib:/lib:$TBLIB_ROOT/lib:/usr/local/lib64
export EASY_ROOT=/usr
export EASY_LIB_PATH=$EASY_ROOT/lib64
export JAVA_HOME=/opt/taobao/java
```

3. 执行 **source ~/.bashrc** 命令让环境变量配置生效。

## 2.5 配置免登录

在 OceanBase 的安装服务器中选择一台作为本机，配置该服务器到所有安装服务器的免登录（包括本机到本机）。配置免登录后，该服务器在连接其他服务器时，无需输入密码。

### \* 配置过程

假设本机的 IP 为“10.10.10.2”，配置免登录的操作步骤如下：

1. 以 **admin** 用户登录各 OceanBase 服务器。
2. 执行以下命令，修改“.ssh 目录”权限。  
**chmod 755 ~/.ssh**  
*说明：如果“.ssh”目录不存在，请先执行 **mkdir ~/.ssh** 命令创建。*
3. 以 **admin** 用户登录本机（10.10.10.2）。
4. 执行以下命令，进入“.ssh”目录。  
**cd ~/.ssh**
5. 执行以下命令，并按“Enter”键，直至生成公钥。  
**ssh-keygen -t rsa**
6. 执行以下命令，并根据提示输入登录密码，配置免登录。  
**ssh-copy-id admin@10.10.10.2**  
**ssh-copy-id admin@10.10.10.4**  
**ssh-copy-id admin@10.10.10.5**

### \* 验证

配置免登录完成后，在本机中输入“**ssh admin@10.10.10.X**”。

- 如果无需输入密码，则表示配置免登录成功。
- 如果仍需要输入密码，则请重新配置免登录。

## 2.6 创建数据磁盘挂载点

数据磁盘用于存放 UpdateServer 和 ChunkServer 的数据。如果您挂载磁盘，那么 UpdateServer 和 ChunkServer 的数据将存放到挂载的磁盘中，否则，将存放在挂载点中。

创建 UpdateServer 和 ChunkServer 数据磁盘挂载点的操作步骤如下：

1. 以 **admin** 用户分别登录 UpdateServer 和 ChunkServer 所在的 OceanBase 服务器。
2. 执行以下命令，创建磁盘挂载目录。  
**sudo mkdir /data**
3. 执行以下命令，将“/data”目录赋给“admin”用户。  
**sudo chown admin /data**
4. 根据磁盘规划和服务器规划创建挂载点。

*说明：*如果不能采用“for”语句创建，您可以根据规划在相应的服务器中直接使用 **mkdir** 命令逐个创建。

- UpdateServer  
**for disk in {1..8}; do mkdir -p /data/\$disk; done;**
- ChunkServer  
**for disk in {1..8}; do mkdir -p /data/\$disk; done;**

## 3 采用 RPM 包安装

在安装 OceanBase 前，请先完成“2 准备安装环境”，否则会造成安装失败。

安装 OceanBase 的主要方式有两种：通过 RPM 包安装和通过源码安装。如果您是普通用户，建议您采用 RPM 安装；如果您是开发人员，建议您采用源码安装。如果您采用源码安装，则可以跳过本小节。

### 3.1 下载安装包

下载 OceanBase 安装包的操作步骤如下：

1. 以 **admin** 用户分别登录各 OceanBase 服务器。
2. 执行以下命令，下载 OceanBase 安装包。  
**git clone https://github.com/alibaba/oceanbase\_rpm\_package ~/oceanbase\_install**

下载时间大概需要 20 分钟，请耐心等待。安装目录说明如[表 3-1](#) 所示。

表 3-1 安装包说明

目录	安装包	说明
Doc	-	OceanBase 文档存放目录。
Lib	<ul style="list-style-type: none"><li>• lzo-2.06-0.x86_64.rpm</li><li>• snappy-1.1.2-2.x86_64.rpm</li></ul>	采用 RPM 包安装 OceanBase 时需要的依赖包。
Package	<ul style="list-style-type: none"><li>• oceanbase-0.4.1.2-1209.el6.x86_64.rpm</li><li>• oceanbase-0.4.1.2-1209.el5.x86_64.rpm</li></ul>	<p>OceanBase 的 rpm 包。其中“el5”为 Linux 版本为 RedHat 5 的安装包；“el6”为 Linux 版本为 RedHat 6 的安装包。</p> <p>本文档中使用的安装包版本仅为举例，实际请采用最新安装包。</p> <p><b>说明：</b>您可以执行 <b>cat /etc/issue</b> 命令查看 Linux 版本号。</p>

目录	安装包	说明
Script	<ul style="list-style-type: none"> <li>oceanbase.conf.template</li> <li>oceanbase.pl</li> </ul>	一键安装脚本。

注：“-”表示无。

## 3.2 安装动态库

安装“LZO”和“Snappy”的操作步骤如下：

1. 以 **admin** 用户分别登录各 OceanBase 服务器。
2. 执行以下命令，进入依赖包目录。  
**cd ~/oceanbase\_install/Lib**
3. 依次执行以下命令，安装“LZO”和“Snappy”。  
**sudo rpm -ivh lzo-2.06-0.x86\_64.rpm**  
**sudo rpm -ivh snappy-1.1.2-2.x86\_64.rpm**

## 3.3 安装 OceanBase 软件

安装 OceanBase 软件操作步骤如下：

1. 以 **admin** 用户分别登录各 OceanBase 服务器。
2. 执行以下命令，进入安装包目录。  
**cd ~/oceanbase\_install/Package**
3. 执行以下命令，安装 OceanBase。  
**sudo rpm --nodeps -ivh oceanbase-0.4.1.2-1209.el6.x86\_64.rpm --prefix=/home/admin/oceanbase**
4. 参考“3.1 下载安装包”到“3.3 安装 OceanBase 软件”，分别在 5 台服务器中完成 OceanBase 的安装。

# 4 采用源码安装

在安装 OceanBase 前，请先完成“2 准备安装环境”，否则会造成安装失败。

安装 OceanBase 的主要方式有两种：通过 RPM 包安装和通过源码安装。如果您是普通用户建议您采用 RPM 安装；如果您是开发人员，建议您采用源码安装。如果您采用 RPM 安装，则可以跳过本小节。

## 4.1 下载安装包

下载 libeasys 和 OceanBase 安装包的操作步骤如下：

1. 以 **admin** 用户分别登录各 OceanBase 服务器。
2. 执行以下命令，下载 libeasys 和 OceanBase 安装包。

```
git clone https://github.com/alibaba/oceanbase
~/oceanbase_install
```

下载的安装包说名如[表 4-1](#) 所示。

表 4-1 安装包说明

分支	安装包	说明	存放位置
oceanbase_0.4	-	OceanBase 0.4 的安装源码。	存放在分支的起始目录。
	t_libeasys-1.0.13-183.el5.x86_64.rpm	Linux 版本为 RedHat 5 的 libeasys 安装包。采用源码安装时需要安装。	存放在分支的“libeasys_rpm”文件夹中。
	t_libeasys-devel-1.0.13-183.el5.x86_64.rpm	<i>说明：</i> 您可以执行 <b>cat /etc/issue</b> 命令查看 Linux 版本号。	
	t_libeasys-1.0.13-183.el6.x86_64.rpm	Linux 版本为 RedHat 6 的 libeasys 安装包。采用源码安装时需要安装。	
	t_libeasys-devel-1.0.13-183.el6.x86_64.rpm		

分支	安装包	说明	存放位置
oceanbase_0.3	-	OceanBase 0.3 的安装源码。	存放在分支的起始目录。

注：“-”表示无。

## 4.2 安装动态库

安装动态库主要包括 liblzo2、Snappy、安装工具组、libnuma、libaio、gtest、gmock 和其他动态库。如果您已经安装这些动态库，则可以跳过本章节。

### 4.2.1 安装 liblzo2

liblzo2 是一个压缩库，OceanBase 需要用它来压缩静态数据。

#### \* 安装

安装 liblzo2 的操作步骤如下：

1. 以 **admin** 用户分别登录各 OceanBase 服务器。
2. 执行以下命令，下载“liblzo2”的安装包。  
**wget -c http://www.oberhumer.com/opensource/lzo/download/lzo-2.06.tar.gz**
3. 执行以下命令，解压缩“lzo-2.06.tar.gz”。  
**tar xzf lzo-\***
4. 执行以下命令，进入“/home/admin/lzo-2.0.6”目录。  
**cd lzo-2.06**
5. 执行以下命令，编译并安装 liblzo2。  
**./configure --enable-shared --prefix=/usr/ && make && sudo make install**

#### \* 验证

安装完成后您可以编译一个 C 程序，验证 liblzo2 是否安装成功。

1. 在 OceanBase 服务器中输入以下代码：

```
echo "int main(){ return 0;}" > /tmp/a.c && gcc /tmp/a.c -llzo2 -o /tmp/a.out
```

2. 执行 **/tmp/a.out** 命令，看是否报错。
  - 没有报错，则说明安装成功。
  - 显示以下的消息，则说明环境变量配置不正确。  
请将“liblzo2.so.2”的目录加入到“/home/admin/.bashrc”文件的“LD\_LIBRARY\_PATH”参数中。

```
./a.out: error while loading shared libraries: liblzo2.so.2: cannot open shared
object file: No such file or directory
```

## 4.2.2 安装 Snappy

Snappy 是 Google 出品的压缩库。OceanBase 使用 Snappy 压缩静态数据。

*注意：Snappy 依赖于 liblzo2，因此，安装 Snappy 前请先安装 liblzo2。*

### \* 安装

安装 Snappy 的操作步骤如下：

1. 以 **admin** 用户分别登录各 OceanBase 服务器。
2. 执行以下命令，下载 Snappy 安装包。  
**wget http://snappy.googlecode.com/files/snappy-1.0.3.tar.gz**
3. 执行以下命令，解压缩“snappy-1.0.3.tar.gz”。  
**tar -xvf snappy-1.0.3.tar.gz**
4. 执行以下命令，进入 Snappy 的安装目录。  
**cd snappy-1.0.3**
5. 执行以下命令，安装 Snappy。  
**./configure && make -j 10 && sudo make install**

### \* 验证

安装完成后你可以编译一个 C 程序，验证 Snappy 是否安装成功。

1. 在 OceanBase 服务器中输入以下代码：

```
echo "int main(){ return 0;}" > /tmp/a.c && gcc /tmp/a.c -o /tmp/a.out -lsnappy
```

2. 执行 **/tmp/a.out** 命令，看是否报错。
  - 没有报错，则说明安装成功。
  - 显示以下的消息，则说明环境变量配置不正确。  
请将“libsnappy.so.1”的目录加入到“/home/admin/.bashrc”文件的“LD\_LIBRARY\_PATH”参数中。

```
./a.out: error while loading shared libraries: libsnappy.so.1: cannot open shared
object file: No such file or directory
```

## 4.2.3 安装工具组

编译 OceanBase 的脚本时，用到了 **aclocal**、**autoconf** 和 **automake** 等工具。因此我们需要安装 **libtoolize**（2.2.6 或以上版本），**autoconf**（2.66 或以上版本）和 **automake**（1.10.2 或以上版本）。

- 您可以执行 **libtoolize --version** 命令，查看 libtoolize 版本。
- 您可以执行 **autoconf --version** 命令，查看 autoconf 版本。

- 您可以执行 **automake --version** 命令，查看 automake 版本。

#### \* 安装 libtoolize

安装 libtoolize 2.2.6 的操作步骤如下：

1. 以 **admin** 用户分别登录各 OceanBase 服务器。
2. 执行以下命令，下载“libtoolize”。  
**wget http://mirrors.kernel.org/gnu/libtool/libtool-2.2.6b.tar.gz**
3. 执行以下命令，解压缩安装包。  
**tar zxf libtool-2.2.6b.tar.gz**
4. 执行以下命令，进入安装目录。  
**cd libtool-2.2.6b**
5. 执行以下命令，安装 libtoolize。  
**./configure && make && sudo make install**

#### \* 安装 autoconf

安装 autoconf 2.66 的操作步骤如下：

1. 以 **admin** 用户分别登录各 OceanBase 服务器。
2. 执行以下命令，下载“autoconf”。  
**wget http://ftp.gnu.org/gnu/autoconf/autoconf-2.66.tar.gz**
3. 执行以下命令，解压缩安装包。  
**tar zxf autoconf-2.66.tar.gz**
4. 执行以下命令，进入安装目录。  
**cd autoconf-2.66**
5. 执行以下命令，安装 autoconf。  
**./configure && make && sudo make install**
6. 执行以下命令，将“~/autoconf-2.66/bin/autoconf”文件拷贝到“/usr/bin”目录下。  
**sudo \cp ~/autoconf-2.66/bin/autoconf /usr/bin**

#### \* 安装 automake

安装 automake 1.11.1 操作步骤如下：

1. 以 **admin** 用户分别登录各 OceanBase 服务器。
2. 执行以下命令，下载“automake”。  
**wget http://ftp.gnu.org/gnu/automake/automake-1.11.1.tar.gz**
3. 执行以下命令，解压缩安装包。  
**tar zxf automake-1.11.1.tar.gz**



4. 执行以下命令，进入安装目录。  
**cd automake-1.11.1**
5. 执行以下命令，安装 automake。  
**./configure && make && sudo make install**
6. 执行以下命令，将“~/automake-1.11.1/automake”文件拷贝到“/usr/bin”目录下。  
**sudo cp ~/automake-1.11.1/automake /usr/bin**

#### 4.2.4 安装 libnuma

Oceanbase 数据库中使用了 NUMA，因此需要 libnuma 支持。

安装 libnuma 的操作步骤如下：

1. 以 **admin** 用户分别登录各 OceanBase 服务器。
2. 执行以下命令，安装“libnuma”。  
**sudo yum install numactl-devel.x86\_64**

#### 4.2.5 安装 libaio

Oceanbase 中用到了 AIO，需要 libaio 的支持。下面通过安装 libaio 来添加 numa 相关的头文件和库。

安装 libaio 的操作步骤如下：

1. 以 **admin** 用户分别登录各 OceanBase 服务器。
2. 执行以下命令，下载 libaio 安装包。  
**wget -c http://libaio.sourceforge.net/downloads/0.3.107-7/libaio\_0.3.107.orig.tar.gz**  
*说明：如果该地址失效，请到“<http://libaio.sourceforge.net>”手工下载。*
3. 执行以下命令，解压缩 libaio 安装包。  
**tar xzf libaio\***
4. 执行以下命令，进入 libaio 安装目录。  
**cd libaio-0.3.107**
5. 执行以下命令，编译安装 libaio。  
**make && sudo make install**

#### 4.2.6 安装 gtest 和 gmock（可选）

如果您执行 `./configure --without-test-case` 不编译 OceanBase 的 test，则不需要安装 gtest 和 gmock。

##### \* 安装 gtest

安装 gtest 的操作步骤如下：

1. 以 **admin** 用户分别登录各 OceanBase 服务器。
2. 执行以下命令，下载 gtest 安装包。  
**wget http://googletest.googlecode.com/files/gtest-1.6.0.zip**
3. 执行以下命令，解压缩“gtest-1.6.0.zip”。  
**unzip gtest-1.6.0.zip**
4. 执行以下命令，进入 gtest 的安装目录。  
**cd gtest-1.6.0**
5. 依次执行以下命令，安装 gtest。  
**./configure && make**  
**sudo cp -r include/gtest /usr/local/include**  
**sudo cp -r lib/.libs/\* /usr/local/lib/**

#### **\* 安装 gmock**

安装 gmock 的操作步骤如下：

1. 以 **admin** 用户分别登录各 OceanBase 服务器。
2. 执行以下命令，下载 gmock 安装包。  
**wget http://googlemock.googlecode.com/files/gmock-1.6.0.zip**
3. 执行以下命令，解压缩 gmock 安装包。  
**unzip gmock-1.6.0.zip**
4. 执行以下命令，进入 gmock 的安装目录。  
**cd gmock-1.6.0**
5. 依次执行以下命令，安装 gmock。  
**./configure && make**  
**sudo cp -r include/gmock /usr/local/include**  
**sudo cp -r lib/.libs/\* /usr/local/lib/**

#### **4.2.7 安装其他库**

在编译 OceanBase 时，还需要使用“libcurl-devel”、“openssl-devel”、“readline-devel”、“ncurses-devel”和“mysql-devel”。

安装这些库的操作步骤如下：

1. 以 **admin** 用户分别登录各 OceanBase 服务器。
2. 依次执行以下命令，安装“libcurl-devel”、“openssl-devel”、“readline-devel”、“ncurses-devel”和“mysql-devel”。  
**sudo yum install libcurl-devel**  
**sudo yum install openssl-devel**  
**sudo yum install readline-devel**  
**sudo yum install ncurses-devel**  
**sudo yum install mysql-devel**

## 4.3 安装 tbsys 和 tbnet

tbsys 主要对操作系统服务进行封装，tbnet 主要提供网络框架。OceanBase 依赖于这两个库。

### \* 安装

安装 tbsys 和 tbnet 的操作步骤如下：

1. 以 **admin** 用户分别登录各 OceanBase 服务器。
2. 执行以下命令，下载 tbsys 和 tbnet 的安装包。  
**svn checkout http://code.taobao.org/svn/tb-common-utils/trunk/~tb-common-utils**
3. 执行以下命令，进入 tbsys 和 tbnet 的安装目录。  
**cd ~/tb-common-utils**
4. 执行以下命令，编译安装 tbsys 和 tbnet。  
环境变量文件“/home/admin/.bashrc”中的“TBLIB\_ROOT”参数所指示的目录下会生成“include”和“lib”两个子目录。  
**sh build.sh**
5. 参考“步骤 1”至“步骤 4”，分别在各服务器中完成 tbsys 和 tbnet 的安装。

### \* 验证

安装成功后，可以采用如下方法验证编译器能否找到库：

1. 在 OceanBase 服务器中输入以下代码。  

```
echo "int main(){ return 0;}" > /tmp/a.c && gcc /tmp/a.c -o /tmp/a.out -L$TBLIB_ROOT/lib -ltbnet -ltbsys
```
2. 执行 **/tmp/a.out** 命令，运行“a.out”。
  - 如果没报错，则说明安装成功。
  - 如果报错，请检查“/home/admin/.bashrc”文件中的“TBLIB\_ROOT”参数是否配置正确。

## 4.4 安装 libeasy

libeasy 是 Oceanbase 中新的网络通讯框架。

安装 libeasy 的操作步骤如下：

1. 执行以下命令，进入安装目录。  
**cd ~/oceanbase\_install**
2. 执行以下命令，切换到 libeasy 分支。  
**git checkout oceanbase\_0.4**

3. 执行以下命令，进入“libeasy\_rpm”目录。  
**cd ~/oceanbase\_install/libeasy\_rpm**
4. 依次执行以下命令，安装 libeasy。  
**sudo rpm -ivh t\_libeasy-1.0.13-183.el6.x86\_64.rpm**  
**sudo rpm -ivh t\_libeasy-devel-1.0.13-183.el6.x86\_64.rpm**

## 4.5 安装 OceanBase 软件

安装 OceanBase 软件操作步骤如下：

1. 执行以下命令，进入安装目录。  
**cd ~/oceanbase\_install**
2. 执行以下命令，切换到 OceanBase 0.4 的分支。  
**git checkout oceanbase\_0.4**
3. 执行以下命令，初始化安装。  
**sh build.sh init**
4. 执行以下命令，指定安装目录“/home/admin/oceanbase”。  
**./configure --prefix=/home/admin/oceanbase --with-release=yes --with-test-case=no;**
5. 依次执行以下命令，编译安装程序。  
**make -j 10 -C src/**  
**make -j 10 -C tools/**
6. 执行以下命令，安装 OceanBase。  
**make install**
7. 执行以下命令，进入“io\_fault”目录。  
**cd ~/oceanbase\_install/tools/io\_fault/**
8. 执行以下命令，编译 tool 工具。  
**make**

## 4.6 创建各 Server 所需目录

启动 RootServer、UpdateServer 和 ChunkServer 需要创建文件存放目录。

### \* 创建

创建 RootServer、UpdateServer 和 ChunkServer 所需目录操作步骤如下：

1. 以 **admin** 用户登录 RootServer 和 UpdateServer 所在的 OceanBase 服务器。
2. 执行以下命令，创建数据存放目录。  
**mkdir -p /home/admin/oceanbase/data**

3. 执行以下命令，创建 RootServer 所需目录。  
**mkdir -p /home/admin/oceanbase/data/rs**  
**mkdir -p /home/admin/oceanbase/data/rs\_commitlog**
  4. 执行以下命令，创建 UpdateServer 所需目录。  
**mkdir -p /home/admin/oceanbase/data/ups\_commitlog**  
**mkdir -p /home/admin/oceanbase/data/ups\_data/raid0**  
**mkdir -p /home/admin/oceanbase/data/ups\_data/raid1**  
**mkdir -p /home/admin/oceanbase/data/ups\_data/raid2**  
**mkdir -p /home/admin/oceanbase/data/ups\_data/raid3**
  5. 执行以下命令，建立 UpdateServer 与数据存放磁盘的软连接。  
**ln -s /data/1 /home/admin/oceanbase/data/ups\_data/raid0/store0**  
**ln -s /data/2 /home/admin/oceanbase/data/ups\_data/raid0/store1**  
**ln -s /data/3 /home/admin/oceanbase/data/ups\_data/raid1/store0**  
**ln -s /data/4 /home/admin/oceanbase/data/ups\_data/raid1/store1**  
**ln -s /data/5 /home/admin/oceanbase/data/ups\_data/raid2/store0**  
**ln -s /data/6 /home/admin/oceanbase/data/ups\_data/raid2/store1**  
**ln -s /data/7 /home/admin/oceanbase/data/ups\_data/raid3/store0**  
**ln -s /data/8 /home/admin/oceanbase/data/ups\_data/raid3/store1**
  6. 以 **admin** 用户登录 ChunkServer 所在的 OceanBase 服务器。
  7. 执行以下命令，创建数据存放目录。  
**mkdir -p /home/admin/oceanbase/data**
  8. 在 ChunkServer 挂载的磁盘创建 sstable 存放的目录  
“obtest/sstable”。
- 注意：*“obtest”与 APP 名称相同。
- ```
for disk in {1..8}; do mkdir -p /data/$disk/obtest/sstable; done;
```
9. 执行以下命令，建立 ChunkServer 与数据存放磁盘的软连接。  
**for disk in {1..8}; do ln -s /data/\$disk**  
**/home/admin/oceanbase/data/\$disk; done;**
  10. 参考“4.1 下载安装包”至“4.6 创建各 Server 所需目录”，分别完成各服务器 OceanBase 软件的安装。

#### \* 验证

在 RootServer/UpdateServer 所在的 OceanBase 服务器中：

- “/home/admin/oceanbase/data”目录如下所示：

```
[admin@localhost data]$ ll
total 16
drwxrwxr-x 2 admin admin 4096 Aug  5 04:57 rs
drwxrwxr-x 2 admin admin 4096 Aug  5 04:57 rs_commitlog
drwxrwxr-x 2 admin admin 4096 Aug  5 04:57 ups_commitlog
drwxrwxr-x 6 admin admin 4096 Aug  5 04:57 ups_data
```

- “/home/admin/oceanbase/data/ups\_data/raid0”目录如下所示，  
“raid0”到“raid3”下目录均相同：

```
[admin@localhost raid0]$ ll
total 0
lrwxrwxrwx 1 admin admin 7 Aug  5 04:58 store0 -> /data/1
lrwxrwxrwx 1 admin admin 7 Aug  5 04:58 store1 -> /data/2
```

在 ChunkServer 所在的 OceanBase 服务器中：

- “/home/admin/oceanbase/data” 目录如下所示：

```
[admin@localhost data]$ ll
total 0
lrwxrwxrwx 1 admin admin 7 Aug  5 05:14 1 -> /data/1
lrwxrwxrwx 1 admin admin 7 Aug  5 05:14 2 -> /data/2
lrwxrwxrwx 1 admin admin 7 Aug  5 05:14 3 -> /data/3
lrwxrwxrwx 1 admin admin 7 Aug  5 05:14 4 -> /data/4
lrwxrwxrwx 1 admin admin 7 Aug  5 05:14 5 -> /data/5
lrwxrwxrwx 1 admin admin 7 Aug  5 05:14 6 -> /data/6
lrwxrwxrwx 1 admin admin 7 Aug  5 05:14 7 -> /data/7
lrwxrwxrwx 1 admin admin 7 Aug  5 05:14 8 -> /data/8
```

- “/data” 目录如下所示：

```
[admin@localhost data]$ ls /data/*/obtest/sstable
/data/1/obtest/sstable:
/data/2/obtest/sstable:
/data/3/obtest/sstable:
/data/4/obtest/sstable:
/data/5/obtest/sstable:
/data/6/obtest/sstable:
/data/7/obtest/sstable:
/data/8/obtest/sstable:
```

# 5 启动 OceanBase

如果您需要部署 OceanBase 集群或者主备 RootServer，请跳过本小节，并参考《OceanBase 0.4.1 配置指南》进行配置和启动。

## \* 前提条件

已经在各服务器上正确安装 OceanBase。

## \* 采用 RPM 安装的启动方式

配置一键启动及初始化的方法如下：

1. 以 **admin** 用户登录 OceanBase 服务器（10.10.10.2）。
2. 执行以下命令，进入一键脚本目录。  
**cd ~/oceanbase\_install/Script**
3. 执行以下命令，复制配置文件。  
**cp oceanbase.conf.template deploy.conf**
4. 使用 **vi** 编辑器，修改配置文件“deploy.conf”，如黑体部分所示。参数说明见注释部分。

**注意：**所有以“#@”开始的行有特殊含义，不允许当注释删除。

```
# @begin_global [settings]
# rs_admin 工具的位置，请勿修改。
rs_admin=./bin/rs_admin
# OceanBase 的安装目录。
ob_home=/home/admin/oceanbase
[public]
# APP 名称。
appname=obtest
# 主集群 ID，与集群名称对应，即以#@begin_cluster_x 和#@end_cluster_x 开头的行。
# OceanBase 内部使用纯数字 ID，即该配置中的数字部分为 ob 内部使用的集群 ID 号。
# 如果不指定集群 ID，则默认使用数字最小的集群为主集群。
master_cluster_id=cluster_1
# 网络接口名称，默认是 bond0。放到不同的 section 下可以单独为那个 section 中的 server 进行配置。
devname=eth0
[rootserver]
# RootServer 的服务端口。
port=2500
# RootServer 存放 commitlog 的目录。
# 执行脚本后，会在“/home/admin/oceanbase/data”下创建“rs_commitlog”目录，并软连接到“/data/log/rs_commitlog”。
commitlog_dir=/data/log/rs_commitlog
```

```

[chunkserver]
# ChunkServer 的端口。
port=2600
# ChunkServer 使用的磁盘数。
# 需要已经建立/data/{1..max_disk_num}的目录。
max_disk_num=8
[mergeserver]
# MergeServer 的服务端口。
port=2800
# MergeServer 的 MySQL 端口。
sql_port=2880
# 部署在 RootServer 上的 Listener 端口。
# 请勿修改！
lms_port=2828
[updateserver]
# UpdateServer 的服务端口。
port=2700
# UpdateServer 用于每日合并的端口。
inner_port=2701
# UpdateServer 转储用的磁盘的数目。
# 需要已经建立/data/{1..max_disk_num}的目录。
max_disk_num=8
# UpdateServer 存放 commitlog 的目录。
# 执行脚本后，会在“/home/admin/oceanbase/data”下创建“ups_commitlog”目录，
并软连接到“/data/log/ups_commitlog”。
commitlog_dir=/data/log/ups_commitlog
#@end_global

#@begin_init_config
# 各 Server 启动时使用的配置项。
[rootserver]

[chunkserver]

[mergeserver]

[updateserver]
log_sync_type=1
#@end_init_config

#@begin_cluster_1
[public]
[rootserver]
# vip 为主备 RootServer 的虚拟 IP。
# 当 RootServer 单机时，请输入 RootServer 的实际 IP。
vip=10.10.10.2
# 主备 RootServer 的 IP 地址。
10.10.10.2
[updateserver]
# 主备 UpdateServer 的 IP 地址。
10.10.10.2
[chunkserver]
# 所有 ChunkServer 的 IP 地址。

```



```

10.10.10.4
10.10.10.5
[mergeserver]
# 所有 MergeServer 的 IP 地址。
10.10.10.4
10.10.10.5
#@end_cluster_1

##@begin_cluster_2
## 多集群时需要配置，详细请参考 cluster_1。
##@end_cluster_2

```

5. 执行以下命令，一键启动及初始化。参数说明如[表 5-1](#)所示，其他脚本命令请参见“8.1.4 一键脚本操作”。

**./oceanbase.pl init --force -c 1 deploy.conf**

**表 5-1 参数说明**

| 参数           | 说明                                            |
|--------------|-----------------------------------------------|
| oceanbase.pl | 运行脚本名称。                                       |
| init         | 操作类型。初始化环境、启动并初始化集群。在首次安装时使用。                 |
| --force      | 强制执行，可省略。                                     |
| -c 1         | 只对 cluster_1 进行初始化。如果不指定集群 ID，则初始化配置文件中的所有集群。 |
| deploy.conf  | 配置文件名称。                                       |

#### \* 采用源码安装的启动方式

启动 RootServer、UpdateServer、ChunkServer 和 MergeServer 的方法如下：

1. 以 **admin** 用户分别登录各 OceanBase 服务器。
2. 执行以下命令，进入 OceanBase 安装目录。  
**cd /home/admin/oceanbase**
3. 启动 RootServer、UpdateServer、ChunkServer 和 MergeServer。参数说明如[表 5-2](#)所示。

**注意：**启动 ChunkServer 前请先启动 RootServer，否则 ChunkServer 在一段时间后会自动结束进程。

- “10.10.10.2”中，启动主 RootServer/UpdateServer/Listener  
**bin/rootserver -r 10.10.10.2:2500 -R 10.10.10.2:2500 -i eth0 -C 1**  
**bin/updateserver -r 10.10.10.2:2500 -p 2700 -m 2701 -i eth0**  
**bin/mergeserver -r 10.10.10.2:2500 -p 2800 -z 2828 -i eth0**

- 依次在“10.10.10.4”和“10.10.10.5”中，启动  
ChunkServer/MergeServer  
**bin/chunkserver -r 10.10.10.2:2500 -p 2600 -n obtest -i eth0**  
**bin/mergeserver -r 10.10.10.2:2500 -p 2800 -z 2880 -i eth0**  
说明：obtest 为“4.6 创建各 Server 所需目录”中创建 sstable 所在的目录。

表 5-2 参数解释

| 服务器          | 参数 | 说明                                                                                                  |
|--------------|----|-----------------------------------------------------------------------------------------------------|
| RootServer   | -r | 需要启动的 RootServer 的 IP 地址和服务端口。主备 RootServer 时，为需要启动的 RootServer 的 VIP 地址和服务端口。<br>格式：-r [IP]:[Port] |
|              | -R | 主集群的 RootServer 的 IP 地址和端口。主备 RootServer 时，为主集群的 RootServer 的 VIP 地址和服务端口。<br>格式：-R [IP]:[Port]     |
|              | -i | 设置绑定的网卡。<br>格式：-i [NIC Name]                                                                        |
|              | -C | 设置集群 ID，必须为数字。<br>格式：-C [Cluster ID]                                                                |
| UpdateServer | -r | 所在集群的 RootServer 的 IP 地址和端口。主备 RootServer 时，为所在集群的 RootServer 的 VIP 地址和服务端口。<br>格式：-r [IP]:[Port]   |
|              | -p | 设置当前 UpdateServer 的服务端口。<br>格式：-p [Port]                                                            |
|              | -m | 每日合并操作时，ChunkServer 请求合并数据所用的端口。<br>格式：-m [Port]                                                    |

| 服务器         | 参数 | 说明                                                                                                  |
|-------------|----|-----------------------------------------------------------------------------------------------------|
|             | -i | 设置绑定的网卡。<br>格式: -i [NIC Name]                                                                       |
| ChunkServer | -r | 所在集群的 RootServer 的 IP 地址和端口。主备 RootServer 时, 为所在集群的 RootServer 的 VIP 地址和服务端口。<br>格式: -r [IP]:[Port] |
|             | -p | 设置当前 ChunkServer 的服务端口。<br>格式: -p [Port]                                                            |
|             | -n | APP 名称。与“4.6 创建各 Server 所需目录”中 sstable 的父目录名称保持一致。<br>格式: -n [APP Name]                             |
|             | -i | 设置绑定的网卡。<br>格式: -i [NIC Name]                                                                       |
| MergeServer | -r | 所在集群的 RootServer 的 IP 地址和端口。主备 RootServer 时, 为所在集群的 RootServer 的 VIP 地址和服务端口。<br>格式: -r [IP]:[Port] |
|             | -p | 设置当前 MergeServer 的服务端口。<br>格式: -p [Port]                                                            |
|             | -z | 设置 MergeServer 的 MySQL 的协议端口。<br>格式: -z [Port]                                                      |
|             | -i | 设置绑定的网卡。<br>格式: -i [NIC Name]                                                                       |

| 服务器      | 参数 | 说明                                                                                                |
|----------|----|---------------------------------------------------------------------------------------------------|
| Listener | -r | 所在集群的 RootServer 的 IP 地址和端口。主备 RootServer 时，为所在集群的 RootServer 的 VIP 地址和服务端口。<br>格式：-r [IP]:[Port] |
|          | -p | 设置 Listener 的服务端口。<br>格式：-p [Port]                                                                |
|          | -z | 设置 Listener 的 MySQL 的协议端口。<br>格式：-z [Port]                                                        |
|          | -i | 设置绑定的网卡。<br>格式：-i [NIC Name]                                                                      |

- 以 **admin** 用户登录主 RootServer 所在的 OceanBase 服务器（10.10.10.2）。  
*说明：* 以下操作仅第一次启动 OceanBase 各 Server 后需要执行。
- 执行以下命令，进入 OceanBase 安装目录。  
**cd /home/admin/oceanbase**
- 依次执行以下命令，初始化 OceanBase，参数说明如[表 5-3](#)所示。  
**bin/rs\_admin -r 10.10.10.2 -p 2500 set\_obi\_role -o OBI\_MASTER**  
**bin/rs\_admin -r 10.10.10.2 -p 2500 -t 60000000 boot\_strap**

表 5-3 参数说明

| 参数 | 说明                                                                      |
|----|-------------------------------------------------------------------------|
| -r | RootServer 的 IP 地址。主备 RootServer 时，为 RootServer 的 VIP 地址。<br>格式：-r [IP] |
| -p | RootServer 的端口号。<br>格式：-p [Port]                                        |
| -o | 指定主集群 RootServer。<br>格式：set_obi_role -o OBI_MASTER                      |

| 参数 | 说明                                                           |
|----|--------------------------------------------------------------|
| -t | <p>命令的超时时长。</p> <p>单位：微秒。</p> <p>格式：-t [Time] boot_strap</p> |

**小窍门：**在“/home/admin/oceanbase/bin”目录下，执行**`./rs_admin`**命令，可以查看**`help`**信息。

配置成功后，系统显示如下：

```
[admin@obtest-1-2 ztt.alipay.net /home/admin/oceanbase/bin]
$ ./rs_admin -r 10.10.10.2 -p 2500 set_obi_role -o OBI_MASTER
timeout=10000000
set_obi_role...role=0
Okay

[admin@obtest-1-2 ztt.alipay.net /home/admin/oceanbase/bin]
$ ./rs_admin -r 10.10.10.2 -p 2500 -t 60000000 boot_strap
timeout=60000000
do_rs_admin, cmd=16...
Okay
```

# 6 安装 MySQL 客户端

---

您需要在本地计算机中安装 MySQL 客户端链接 OceanBase。

## \* 安装

假设本地计算机的用户为 **sqluser**。安装客户端的操作步骤如下：

1. 以 **sqluser** 用户登录本地计算机。
2. 执行以下命令，安装 MySQL 客户端。  
**sudo yum install mysql**

## \* 后续操作

- 执行 **mysql -h 10.10.10.4 -P2880 -uadmin -padmin** 命令，链接 OceanBase。
  - IP 为 MergeServer 的 IP 地址。
  - 端口号为 MySQL 协议端口。
  - OceanBase 的初始“用户名/密码”为“admin/admin”。
- 执行 **exit** 命令，退出 OceanBase。
- 如果您想要详细了解 OceanBase 的使用，请参考《OceanBase 0.4.1 SQL 参考指南》。

# 7 配置 OceanBase

安装完成后，需要对 OceanBase 的各个 Server 进行配置，优化 OceanBase 服务。

OceanBase 各个 Server 的配置项存放在 OceanBase 的“\_\_all\_sys\_config”表中，修改 Server 的配置项的方法举例如下：

```
ALTER SYSTEM SET balance_tolerance_count=30
                COMMENT 'Modify by Bruce'
                SCOPE = BOTH
                SERVER_TYPE = ROOTSERVER
                SERVER_IP= '10.10.10.2'
                SERVER_PORT= 2500;
```

**说明：**您可以执行 **SHOW PARAMETERS LIKE 'param\_name';** 命令，查看配置项的值。

如果您需要详细了解修改命令，请参见《OceanBase 0.4.1 SQL 参考指南》的“5.4 修改系统配置项”。

[表 7-1](#)、[表 7-2](#)、[表 7-3](#) 和 [表 7-4](#) 分别列出了 DBA 需要配置和关注的各 Server 参数，如果您需要了解其他配置参数，请参见“9.4 配置参数说明”。

## \* RootServer

RootServer 配置参数如[表 7-1](#)所示。

表 7-1 RootServer 配置参数

| 参数                      | 缺省值 | 推荐值 | 含义                                                                                                                                                                           |
|-------------------------|-----|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| balance_tolerance_count | 10  | 10  | 对单个表来说，可能保存有一个或多个 SSTable，这些 SSTable 分布在各个 ChunkServer 数量在以下区间内：<br>[SSTable 数/ChunkServer 总数 - balance_tolerance_count, SSTable 数/ChunkServer 总数 + balance_tolerance_count] |

| 参数                             | 缺省值 | 推荐值   | 含义                                                                                                         |
|--------------------------------|-----|-------|------------------------------------------------------------------------------------------------------------|
| read_master_master_ups_percent | 100 | 40    | OceanBase 主集群中主 Update Server 的读服务百分比，而备 UpdateServer 的读服务百分比为“(1-主 UpdateServer 的百分比)/备 UpdateServer 数量”。 |
| read_queue_size                | 500 | 10000 | RootServer 处理读请求任务的队列大小。                                                                                   |
| read_slave_master_ups_percent  | 100 | 50    | OceanBase 备集群中主 Update Server 的读服务百分比，而备 UpdateServer 的读服务百分比为“(1-主 UpdateServer 的百分比)/备 UpdateServer 数量”。 |

### \* UpdateServer

UpdateServer 配置参数如[表 7-2](#)所示。

表 7-2 UpdateServer 配置参数

| 参数               | 缺省值    | 推荐值                                                       | 含义                                                                                                                                                                                                                                                                                                                                        |
|------------------|--------|-----------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| active_mem_limit | 系统自动生成 | 系统自动生成<br>• app_mod=import 时：69G<br>• app_mod=oltp 时：286G | <p>用户的更新操作写入 Active MemTable。当 Active MemTable 大小到达该值后，则冻结 Active MemTable，同时开启新的 Active MemTable 接受更新操作。</p> <p>计算方法如下：</p> <ul style="list-style-type: none"> <li>app_mod=import 时：table_memory_limit/minor_num_limit*0.7</li> <li>app_mod=oltp 时：“table_mem_limit_gb”减去为冻结表预留内存的大小。预留大小为 table_mem_limit_gb 的 10%，但最大为 10G。</li> </ul> |



| 参数                    | 缺省值    | 推荐值                  | 含义                                                                                                                                                                                                                                                                                                                                                                                       |
|-----------------------|--------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| blockcache_size       | 系统自动生成 | 系统自动生成（21218420522B） | <p>Block 缓存大小。该参数不可以动态改小，但是可以动态改大。</p> <p>需要重新启动 UpdateServer 服务才能生效。</p> <p>计算方法：table_memory_limit/15</p>                                                                                                                                                                                                                                                                              |
| blockindex_cache_size | 系统自动生成 | 系统自动生成（15913815391B） | <p>Block 索引缓存大小。该参数不可以动态改小，但是可以动态改大。</p> <p>需要重新启动 UpdateServer 服务才能生效。</p> <p>计算方法：table_memory_limit/20</p>                                                                                                                                                                                                                                                                            |
| consistency_type      | 2      | 3                    | <p>一致性 SQL 请求只能读取主 UpdateServer 的 CommitLog；弱一致性 SQL 请求读取 UpdateServer 中 CommitLog 时，需要根据该参数配置的一致性类型进行读取 CommitLog：</p> <ul style="list-style-type: none"> <li>• 1: Strong，只能读取主 UpdateServer 的 CommitLog。</li> <li>• 2: Normal，只有当主备 UpdateServer 同步时，才允许读取备 UpdateServer 的 CommitLog；否则只能读取主 UpdateServer 的 CommitLog。</li> <li>• 3: Weak，可以读取主或备 UpdateServer 的 CommitLog。</li> </ul> |

| 参数                         | 缺省值    | 推荐值                                                    | 含义                                                                                                                                                                                                                                        |
|----------------------------|--------|--------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| minor_num_limit            | 系统自动生成 | 系统自动生成<br>• app_mod=import 时: 3<br>• app_mod=oltp 时: 1 | 小版本的个数大于或等于该值后，再次执行冻结时，则执行主版本冻结。                                                                                                                                                                                                          |
| table_available_error_size | 系统自动生成 | 系统自动生成（10G）                                            | MemTable 可用内存小于该值时，则打印 Error 日志。                                                                                                                                                                                                          |
| table_available_warn_size  | 系统自动生成 | 系统自动生成（8GB）                                            | MemTable 可用内存小于该值时，则打印 Warn 日志。                                                                                                                                                                                                           |
| table_memory_limit         | 系统自动生成 | 系统自动生成（318276307830 B）                                 | MemTable 可用内存。<br>计算方法: $(total\_memory\_limit - total\_reserve\_gb) / (1/20 + 1/15 + 1)$ ，一般情况下 $total\_reserve\_gb=10G$<br><i>说明: UpdateServer 全局内存包括 MemTable、SSTable Cache、事务 Session 和其他。“total_reserve_gb”为事务 Session 与其他预留的内存。</i> |
| total_memory_limit         | 系统自动生成 | 系统自动生成（361850994688 B）                                 | UpdateServer 可用内存。                                                                                                                                                                                                                        |

## \* MergeServer

MergeServer 配置参数如[表 7-3](#)所示。

表 7-3 MergeServer 配置参数

| 参数                           | 缺省值 | 推荐值 | 含义                                                 |
|------------------------------|-----|-----|----------------------------------------------------|
| memory_size_limit_percentage | 40  | 40  | 在 MergeServer 所在服务器的物理内存中，可用于 MergeServer 的最大百分比数。 |

| 参数              | 缺省值 | 推荐值 | 含义                                  |
|-----------------|-----|-----|-------------------------------------|
| network_timeout | 2s  | 3s  | MergeServer 与其他 Server 进行网络互交的超时时间。 |

### \* ChunkServer

ChunkServer 配置参数如[表 7-4](#)所示。

表 7-4 ChunkServer 配置参数

| 参数                        | 缺省值  | 推荐值  | 含义                                                                                                                                                                                                                                                               |
|---------------------------|------|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| block_cache_size          | 1G   | 1G   | Block 缓存大小。类似于 Oracle 的 db cache。配置值越大越好，但是不可超过 MergeServer 和 ChunkServer 总内存大小。                                                                                                                                                                                 |
| block_index_cache_size    | 512M | 4G   | Block 索引缓存大小，主要保存每个 Block 的索引数据。<br>计算方法：(Disk Size / Block Size) * Block Entry Size<br>Block 的大小一般为 4KB~64KB，每个 Block 的管理开销是：20~30Byte+一个 Rowkey 长度，假设 Rowkey 为 50 个 Byte，则一个 Block 的管理成本 70-80byte，如果 ChunkServer 存储 1T 的数据，那么索引的管理成本是“(1T/64k)*80Byte=1.28G”。 |
| merge_delay_interval      | 600s | 600s | 当收到新版本数据后，需要等待该时间后才开始合并。                                                                                                                                                                                                                                         |
| merge_threshold_load_high | 16   | 10   | 每日合并时，当 ChunkServer 负载线程超过该值，且每秒 get 或 scan 请求的次数超过“merge_threshold_request_high”时，则暂停部分合并线程。                                                                                                                                                                    |
| merge_timeout             | 10s  | 30s  | 在数据合并时，读取 UpdateServer 数据的超时时间。                                                                                                                                                                                                                                  |

| 参数                     | 缺省值 | 推荐值 | 含义                                                |
|------------------------|-----|-----|---------------------------------------------------|
| sstable_row_cache_size | 1G  | 20G | SSTable 的行缓存大小。                                   |
| task_thread_count      | 20  | 40  | 单个 ChunkServer 中允许的处理线程总数。OLAP 应用中一般配置为核心的 2 倍左右。 |

# 8 FAQ

---

## 8.1 启动 UpdateServer 时报错

### \* 现象描述

执行 `bin/updateserver -r 10.10.10.2:2500 -p 2700 -m 2701` 命令启动 UpdateServer 时，出现如下报错：

```
[2013-04-12 09:07:01.513130] INFO ob_server_config.cpp:139 [140607873587520]
===== *stop server config report* =====
libnone.so: cannot open shared object file: No such file or directory
[2013-04-12 09:07:01.513823] ERROR check_all (ob_update_server_config.cpp:94)
[140607873587520] cannot load compressor library name=[none]
[2013-04-12 09:07:01.513849] WARN do_work (ob_update_server_main.cpp:98)
[140607873587520] failed to load from conf, ret: [1]
[2013-04-12 09:07:01.513939] ERROR do_work (ob_update_server_main.cpp:103)
[140607873587520] Start Update server failed, ret: [1]
[2013-04-12 09:07:01.513959] INFO base_main.cpp:405 [140607873587520] exit program.
```

### \* 可能原因

- OceanBase 自带的压缩库 `libnone` 环境变量配置错误。
- 当前用户对 `log` 和 `run` 文件没有写权限。

### \* 处理方法

- 将 `libnone` 路径添加到环境变量中，详细操作步骤如下：
  1. 以 **admin** 用户登录 OceanBase 服务器。
  2. 用 **vi** 编辑器在“`/home/admin/.bashrc`”文件中，添加 `libnone` 的安装路径，如黑体部分所示：

```
export TBLIB_ROOT=~/.tb-common-utils
export
LD_LIBRARY_PATH=/home/admin/oceanbase/lib:/usr/local/lib/libsnappy.so:/usr:/usr/lib:/usr/local/lib/lib:$TBLIB_ROOT/lib
export EASY_ROOT=$TBLIB_ROOT
export EASY_LIB_PATH=$EASY_ROOT/lib/lib64
export JAVA_HOME=/opt/taobao/java
```

3. 执行 **`source ~/.bashrc`** 命令让环境变量配置生效。
- 将“`log`”和“`run`”目录的拥有者修改为 **admin**，操作步骤如下：

1. 以 **admin** 用户登录 OceanBase 服务器。
2. 执行以下命令，查看并记录当前用户所在组。  
**groups**
3. 执行以下命令，并输入密码，切换到 root 用户。  
**su - root**
4. 执行以下命令，进入“/home/obuser/oceanbase”目录。  
**cd /home/obuser/oceanbase**
5. 依次执行以下命令，修改“log”和“run”目录的拥有者为 admin。  
**chown -R admin:admin log**  
**chown -R admin:admin run**

## 8.2 安装 gcc 时编译出错

### \* 现象描述

安装 gcc 4.1.2 时，编译报错：

```
/usr/include/gnu/stubs.h:7:27: 错误：gnu/stubs-32.h：没有那个文件或目录
```

### \* 可能原因

glibc-devel 没有安装。

### \* 处理方法

安装 glibc-devel，详细操作步骤如下：

1. 以 root 用户登录 OceanBase 服务器。
2. 执行以下命令，安装 glibc-devel。
  - Ubuntu 操作系统  
**sudo apt-get install libc6-dev-i386**
  - Red Hat 操作系统  
**yum install glibc-devel.i686**
  - CentOS 5.8 操作系统  
**yum install glibc-devel.i386**
  - CentOS 6.3 操作系统  
**yum install glibc-devel.i686**
  - SLES 操作系统  
**zypper in glibc-devel-32bit**

# 9 附录

---

## 9.1 常用操作

### 9.1.1 启动服务

RootServer、UpdateServer、ChunkServer 和 MergeServer 的服务端口将在启动时设置。

启动 RootServer、UpdateServer、ChunkServer 和 MergeServer 服务方法如下：

1. 以 admin 用户登录 OceanBase 服务器。
2. 执行以下命令，进入 OceanBase 的安装目录。  
**cd /home/admin/oceanbase**
3. 执行以下命令，启动 RootServer、UpdateServer、ChunkServer 和 MergeServer。参数说明如[表 9-1](#)所示。  
*注意：启动 ChunkServer 前请先启动 RootServer，否则 ChunkServer 在一段时间后会自动结束进程。*
  - 启动 RootServer  
**bin/rootserver -r 10.10.10.2:2500 -R 10.10.10.2:2500 -i eth0 -C 1**
  - 启动 UpdateServer  
**bin/updateserver -r 10.10.10.2:2500 -p 2700 -m 2701 -i eth0**
  - 启动 ChunkServer  
**bin/chunkserver -r 10.10.10.2:2500 -p 2600 -n obtest -i eth0**
  - 启动 MergeServer  
**bin/mergeserver -r 10.10.10.2:2500 -p 2800 -z 2880 -i eth0**
  - 启动 Listener  
**bin/mergeserver -r 10.10.10.2:2500 -p 2800 -z 2828 -i eth0**

表 9-1 参数解释

| 服务器          | 参数 | 说明                                                                                                  |
|--------------|----|-----------------------------------------------------------------------------------------------------|
| RootServer   | -r | 需要启动的 RootServer 的 IP 地址和服务端口。主备 RootServer 时，为需要启动的 RootServer 的 VIP 地址和服务端口。<br>格式：-r [IP]:[Port] |
|              | -R | 主集群的 RootServer 的 IP 地址和端口。主备 RootServer 时，为主集群的 RootServer 的 VIP 地址和服务端口。<br>格式：-R [IP]:[Port]     |
|              | -i | 设置绑定的网卡。<br>格式：-i [NIC Name]                                                                        |
|              | -C | 设置集群 ID，必须为数字。<br>格式：-C [Cluster ID]                                                                |
| UpdateServer | -r | 所在集群的 RootServer 的 IP 地址和端口。主备 RootServer 时，为所在集群的 RootServer 的 VIP 地址和服务端口。<br>格式：-r [IP]:[Port]   |
|              | -p | 设置当前 UpdateServer 的服务端口。<br>格式：-p [Port]                                                            |
|              | -m | 每日合并操作时，ChunkServer 请求合并数据所用的端口。<br>格式：-m [Port]                                                    |
|              | -i | 设置绑定的网卡。<br>格式：-i [NIC Name]                                                                        |



| 服务器         | 参数 | 说明                                                                                                |
|-------------|----|---------------------------------------------------------------------------------------------------|
| ChunkServer | -r | 所在集群的 RootServer 的 IP 地址和端口。主备 RootServer 时，为所在集群的 RootServer 的 VIP 地址和服务端口。<br>格式：-r [IP]:[Port] |
|             | -p | 设置当前 ChunkServer 的服务端口。<br>格式：-p [Port]                                                           |
|             | -n | APP 名称。与“4.6 创建各 Server 所需目录”中 sstable 的父目录名称保持一致。<br>格式：-n [APP Name]                            |
|             | -i | 设置绑定的网卡。<br>格式：-i [NIC Name]                                                                      |
| MergeServer | -r | 所在集群的 RootServer 的 IP 地址和端口。主备 RootServer 时，为所在集群的 RootServer 的 VIP 地址和服务端口。<br>格式：-r [IP]:[Port] |
|             | -p | 设置当前 MergeServer 的服务端口。<br>格式：-p [Port]                                                           |
|             | -z | 设置 MergeServer 的 MySQL 的协议端口。<br>格式：-z [Port]                                                     |
|             | -i | 设置绑定的网卡。<br>格式：-i [NIC Name]                                                                      |

| 服务器      | 参数 | 说明                                                                                                |
|----------|----|---------------------------------------------------------------------------------------------------|
| Listener | -r | 所在集群的 RootServer 的 IP 地址和端口。主备 RootServer 时，为所在集群的 RootServer 的 VIP 地址和服务端口。<br>格式：-r [IP]:[Port] |
|          | -p | 设置 Listener 的服务端口。<br>格式：-p [Port]                                                                |
|          | -z | 设置 Listener 的 MySQL 的协议端口。<br>格式：-z [Port]                                                        |
|          | -i | 设置绑定的网卡。<br>格式：-i [NIC Name]                                                                      |

### 9.1.2 停止服务

在 OceanBase 服务器中，停止 RootServer、UpdateServer、ChunkServer、MergeServer 和 Listener 服务方法如下：

1. 以 obuser 用户登录 OceanBase 服务器。
2. 执行以下命令，停止 RootServer、UpdateServer、ChunkServer 和 MergeServer。

**注意：**停止各个服务时，不建议使用“kill -9”。

- 停止 RootServer  
**killall rootserver**
- 停止 UpdateServer  
**killall updateserver**
- 停止 ChunkServer  
**killall chunkserver**
- 停止 MergeServer 或 Listener  
**killall mergeserver**

### 9.1.3 重新启动

如果您进行 OceanBase 数据库各 Server 的重新启动操作，请您遵守以下规则：

- 重新启动前，确保各个 Server 的进程已退出。

- 重新启动 RootServer 时，Cluster ID 与之前保持一致。同时必须重新指定主备。
- 重新启动 ChunkServer 时，App Name 与之前保持一致。
- 如果 OceanBase 为单机部署，启动不同 Server 的进程时，建议间隔 10 秒。

### 9.1.4 一键脚本操作

一键脚本的命令以及功能如下所示，参数说明如[表 9-2](#)所示。

- 初始化环境、启动并初始化集群。在首次安装时使用。  
**`./oceanbase.pl init [--force] [-c 1] deploy.conf`**
- 启动服务，不初始化环境。  
**`./oceanbase.pl start [--force] deploy.conf`**
- 停止服务。  
**`./oceanbase.pl stop [--force] deploy.conf`**
- 清除服务。系统将被还原到“init”前，请谨慎使用。  
**`./oceanbase.pl clean [--force] deploy.conf`**

表 9-2 参数说明

| 参数                    | 说明                                            |
|-----------------------|-----------------------------------------------|
| oceanbase.pl          | 运行脚本名称。                                       |
| init/start/stop/clean | 操作类型。                                         |
| --force               | 强制执行，可省略。                                     |
| -c                    | 只对 cluster_1 进行初始化。如果不指定集群 ID，则初始化配置文件中的所有集群。 |
| deploy.conf           | 配置文件名称。                                       |

### 9.1.5 卸载

卸载 OceanBase 数据库只需要删除 OceanBase 的安装用户及目录即可，删除安装用户的操作步骤如下：

1. 以 root 用户登录 OceanBase 服务器。
2. 执行以下命令，停止 admin 下的所有进程。  
**`ps -ef |grep admin|awk '{print $2}' | xargs kill`**

3. 执行如下命令，删除 `admin` 用户及用户目录。  
**`userdel -r admin`**
4. 执行如下命令，删除数据文件。  
**`rm -rf /data`**
5. 执行如下命令，删除临时文件。  
**`rm -rf /tmp/*`**

## 9.2 安装 gcc 4.1.2

安装 gcc 4.1.2 的操作步骤如下：

1. 以 `root` 用户登录 OceanBase 服务器。
2. 执行以下命令，查看是否安装“`makeinfo`”。  
**`makeinfo --version`**
  - 已安装，则记录版本号，然后执行“步骤 3”。
  - 未安装，则执行 **`yum install texinfo`** 命令，安装“`makeinfo`”。
3. 执行以下命令，下载“`gcc-4.1.2.tar.bz2`”。  
**`wget ftp://ftp.gnu.org/gnu/gcc/gcc-4.1.2/gcc-4.1.2.tar.bz2`**
4. 执行以下命令，解压缩“`gcc-4.1.2.tar.bz2`”。  
**`tar -xvf gcc-4.1.2.tar.bz2`**
5. 执行以下命令，进入“`gcc-4.1.2`”目录。  
**`cd gcc-4.1.2`**
6. 使用 `vi` 编辑器，修改“`configure`”文件。如果您的“`makeinfo`”的版本在“4.2-4.9”之间，则跳过此步骤。

```
# For an installed makeinfo, we require it to be from texinfo 4.2 or
# higher, else we use the "missing" dummy.
if ${MAKEINFO} --version \
| egrep 'texinfo[^\0-9]*([1-3][0-9]|4\.[2-9]|[5-9])' >/dev/null 2>&1;
```

- “`makeinfo`”的版本为“4.13”，则将粗体部分修改为以下内容：

```
'texinfo[^\0-9]*([1-3][0-9]|4\.[4-9]|4\.[1-9][0-9]*|[5-9])'
```

- “`makeinfo`”为其他版本，则将粗体部分修改为以下内容：

```
'texinfo[^\0-9]*([1-3][0-9]|4\.[2-9]|4\.[1-9][0-9]*|[5-9])'
```

*小窍门：* 您可以在 `vi` 里使用 `/texinfo[^\0-9]` 快速定位上面两行。

7. 执行以下命令编译 gcc 4.1.2。  
**`./configure --prefix=/usr/local/gcc-4.1.2&& make`**
8. 执行以下命令安装 gcc 4.1.2。  
**`make install`**
9. 执行以下命令，进入“`/usr/bin`”目录。  
**`cd /usr/bin`**

10. 依次执行以下命令，删除原有的 gcc 链接文件。  

```
rm -rf gcc
rm -rf g++
```
11. 依次执行以下命令，建立 gcc 4.1.2 的链接。  

```
ln -s /usr/local/gcc-4.1.2/bin/gcc /usr/bin/gcc
ln -s /usr/local/gcc-4.1.2/bin/g++ /usr/bin/g++
```
12. 执行以下命令，查看 gcc 版本。  

```
gcc -v
```

## 9.3 内部表参数说明

为了区别用户定义的表，OceanBase 的内部表的名称都以下划线“\_\_”开头。

### 9.3.1 \_\_first\_tablet\_entry

“\_\_first\_tablet\_entry”记录了集群中所有 table 的基本属性信息。

Rowkey: (table\_name)

“\_\_first\_tablet\_entry”参数说明如[表 9-3](#)所示。

表 9-3 \_\_first\_tablet\_entry 参数

| 参数                    | 类型         | 说明                                                                                                                             |
|-----------------------|------------|--------------------------------------------------------------------------------------------------------------------------------|
| gm_create             | createtime | 创建时间。                                                                                                                          |
| gm_modify             | modifytime | 修改时间。                                                                                                                          |
| table_name            | varchar    | 表名。                                                                                                                            |
| creat_time_column_id  | int        | create_time 列的列 id。                                                                                                            |
| modify_time_column_id | int        | modify_time 列的列 id。                                                                                                            |
| table_id              | int        | 表 ID。                                                                                                                          |
| table_type            | int        | <ul style="list-style-type: none"> <li>1: 普通表。</li> <li>2: 索引。</li> <li>3: 元数据表。</li> <li>4: view。</li> <li>5: 临时表。</li> </ul> |

| 参数                          | 类型      | 说明                                                                              |
|-----------------------------|---------|---------------------------------------------------------------------------------|
| load_type                   | int     | <ul style="list-style-type: none"> <li>1: 保存到磁盘。</li> <li>2: 保存到内存。</li> </ul>  |
| table_def_type              | int     | <ul style="list-style-type: none"> <li>1: 内部表。</li> <li>2: 用户定义表。</li> </ul>    |
| rowkey_column_num           | int     | 主键的列数，后续 <code>endrowkeyobj1</code> , <code>endrowkeyobj2</code> ...等来依次表示主键的列。 |
| column_num                  | int     | 全部的列数(包括主键)。                                                                    |
| max_used_column_id          | int     | 该表使用过的最大列 ID(列 ID 不重用)。                                                         |
| replica_num                 | int     | 单个集群的 Tablet 的 replica 的个数(1~6)。                                                |
| create_mem_version          | int     | 新建该表时候系统的 <code>mem_version</code> ，暂时保留。                                       |
| tablet_max_size             | int     | 该表每个 Tablet 的 SSTable 文件最大允许大小。                                                 |
| max_rowkey_length           | int     | Rowkey 的最大长度限制。                                                                 |
| compress_func_name          | varchar | 存储 SSTable 所使用的压缩方法名称。                                                          |
| is_use_bloomfilter          | int     | 指定是否使用 bloomfilter。                                                             |
| merge_write_sstable_version | int     | 合并的时候写哪个版本的 SSTable                                                             |
| is_pure_update_table        | int     | 指定是否属于内存更新表。                                                                    |
| expire_condition            | varchar | 使用表达式定义的此表的数据自动过期删除条件。                                                          |

| 参数                | 类型  | 说明                                      |
|-------------------|-----|-----------------------------------------|
| rowkey_split      | int | 用于指定每日合并中 Tablet 的分裂点为 rowkey 的第几个 obj。 |
| tablet_block_size | int | Tablet_block 的大小。                       |
| is_read_static    | int | 是否要读静态数据。                               |

### 9.3.2 \_\_all\_all\_column

“\_\_all\_all\_column”存储了每个表的所有列、column\_id、列类型等，包括内部表(不包括核心表)和用户定义表。与内部表“\_\_all\_join\_info”共同定义了各个表的 schema 信息。

Rowkey: (table\_id, column\_name)

“\_\_all\_all\_column”参数说明如[表 9-4](#)所示。

**表 9-4 \_\_all\_all\_column 参数**

| 参数              | 类型         | 说明                                                                                                                                                                                       |
|-----------------|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| gm_create       | createtime | 创建时间。                                                                                                                                                                                    |
| gm_modify       | modifytime | 修改时间。                                                                                                                                                                                    |
| table_id        | int        | 表 ID。                                                                                                                                                                                    |
| column_name     | varchar    | 列名。                                                                                                                                                                                      |
| table_name      | varchar    | 表名。                                                                                                                                                                                      |
| column_id       | int        | 列 ID。                                                                                                                                                                                    |
| column_group_id | int        | 列隶属的 column group id。                                                                                                                                                                    |
| rowkey_id       | int        | <ul style="list-style-type: none"> <li>0: 非 rowkey。</li> <li>正整数: rowkey 的序号，必须是从 1 开始的连续正整。</li> </ul> <p><b>说明:</b> “__all_table_table” 中的 “rowkey_column_num” 定义了该表的 rowkey 的列数量。</p> |

| 参数               | 类型  | 说明                                                                                      |
|------------------|-----|-----------------------------------------------------------------------------------------|
| length_in_rowkey | int | 如果是 rowkey 列，表示在二进制 rowkey 串中占用的字节数。                                                    |
| order_in_rowkey  | int | 表示该列的升降序。                                                                               |
| join_table_id    | int | <ul style="list-style-type: none"> <li>-1: 没有 join。</li> <li>正整数: 连接表的表 ID。</li> </ul>  |
| join_column_id   | int | <ul style="list-style-type: none"> <li>-1: 没有 join。</li> <li>正整数: 连接表中的列 ID。</li> </ul> |
| data_type        | int | 数据类型。                                                                                   |
| data_length      | int | 整数的字节数或字符串的最大长度。                                                                        |
| data_precision   | int | 整数的十进制位数或 decimal 的有效位数(小数点前和小数点后)。                                                     |
| data_scale       | int | decimal 小数点后的位数。                                                                        |
| nullable         | int | <ul style="list-style-type: none"> <li>1: 不可以为空。</li> <li>2: 可以为空。</li> </ul>           |

### 9.3.3 \_\_all\_join\_info

“\_\_all\_join\_info”存储了表之间的内部 join 关系，即左表通过其某些列对应到右表的 rowkey。

**说明：**左表及右表的对应列的类型必须一致。

Rowkey: (left\_table\_id, left\_column\_id, right\_table\_id, right\_column\_id)

“\_\_all\_join\_info”参数说明如[表 9-5](#)所示。

**表 9-5 \_\_all\_join\_info 参数**

| 参数        | 类型         | 说明    |
|-----------|------------|-------|
| gm_create | createtime | 创建时间。 |
| gm_modify | modifytime | 修改时间。 |



| 参数                | 类型      | 说明       |
|-------------------|---------|----------|
| left_table_id     | int     | 左表的表 ID。 |
| left_column_id    | int     | 左表的列 ID。 |
| right_table_id    | int     | 右表的表 ID。 |
| right_column_id   | int     | 右表的列 ID。 |
| left_table_name   | varchar | 左表的表名。   |
| left_column_name  | varchar | 左表的列名。   |
| right_table_name  | varchar | 右表的表名。   |
| right_column_name | varchar | 右表的列名。   |

### 9.3.4 \_\_all\_client

“\_\_all\_client”用来保存 JAVA 客户端的版本信息。

Rowkey: (client\_ip, version)

“\_\_all\_client”参数说明如[表 9-6](#)所示。

**表 9-6 \_\_all\_client 参数**

| 参数        | 类型         | 说明                                                                                |
|-----------|------------|-----------------------------------------------------------------------------------|
| gm_create | createtime | 创建时间。                                                                             |
| gm_modify | modifytime | 修改时间。                                                                             |
| client_ip | varchar    | <ul style="list-style-type: none"> <li>0: 与特定集群无关。</li> <li>正整数: 指定集群。</li> </ul> |
| version   | varchar    | 客户端版本。                                                                            |
| status    | varchar    | 客户端状态。                                                                            |
| extra1    | varchar    | 预留。                                                                               |
| extra2    | int        | 预留。                                                                               |

### 9.3.5 \_\_all\_cluster

“\_\_all\_cluster”记录了系统中所有的集群，这个表由每个集群的主 RootServer 更新。

Rowkey: (cluster\_id)

“\_\_all\_cluster”参数说明如[表 9-7](#)所示。

**表 9-7 \_\_all\_cluster 参数**

| 参数                   | 类型         | 说明                                                                                         |
|----------------------|------------|--------------------------------------------------------------------------------------------|
| gm_create            | createtime | 创建时间。                                                                                      |
| gm_modify            | modifytime | 修改时间。                                                                                      |
| cluster_id           | int        | 集群 ID，正整数。                                                                                 |
| cluster_vip          | varchar    | Cluster 的 IP 地址。                                                                           |
| cluster_port         | int        | Cluster 端口号。                                                                               |
| cluster_role         | int        | <ul style="list-style-type: none"><li>1: Master</li><li>2: Slave</li></ul>                 |
| cluster_name         | varchar    | 集群名称。                                                                                      |
| cluster_info         | varchar    | 集群说明信息。                                                                                    |
| cluster_flow_percent | int        | 流量配比。                                                                                      |
| read_strategy        | int        | 客户端使用的负载均衡策略： <ul style="list-style-type: none"><li>0: 随机轮转策略。</li><li>1: 一致性哈希。</li></ul> |

### 9.3.6 \_\_all\_server

“\_\_all\_server”记录了系统中所有的服务器，这个表仅仅由主集群的主 RootServer 更新。

Rowkey: (cluster\_id, svr\_type, svr\_ip, svr\_port)

“\_\_all\_server”参数说明如[表 9-8](#)所示。

表 9-8 \_\_all\_server 参数

| 参数          | 类型         | 说明                                                                                                                                          |
|-------------|------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| gm_create   | createtime | 创建时间。                                                                                                                                       |
| gm_modify   | modifytime | 修改时间。                                                                                                                                       |
| cluster_id  | int        | <ul style="list-style-type: none"> <li>0: 与特定集群无关。</li> <li>正整数: 指定的集群 ID。</li> </ul>                                                       |
| svr_type    | varchar    | <ul style="list-style-type: none"> <li>RootServer</li> <li>ChunkServer</li> <li>MergeServer</li> <li>UpdateServer</li> <li>Other</li> </ul> |
| svr_ip      | varchar    | Server IP 地址。                                                                                                                               |
| svr_port    | int        | Server 端口。                                                                                                                                  |
| inner_port  | int        | 内部交互端口。                                                                                                                                     |
| svr_role    | int        | <ul style="list-style-type: none"> <li>0: 与服务器角色(RS 或 UPS 的主或备)无关。</li> <li>1: slave</li> <li>2: master</li> </ul>                          |
| svr_version | varchar    | 程序版本信息。                                                                                                                                     |

### 9.3.7 \_\_all\_server\_stat

“\_\_all\_server\_stat”用于记录本集群内所有服务器的监控信息，属于虚拟的内存表，不对监控数据进行存储。

Rowkey: (svr\_type, svt\_ip, svr\_port, name)

“\_\_all\_server\_stat”参数说明如[表 9-9](#)所示。

表 9-9 \_\_all\_server\_stat 参数

| 参数 | 类型 | 说明 |
|----|----|----|
|----|----|----|

| 参数       | 类型      | 说明                                                                                                                                                    |
|----------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| svr_type | varchar | <ul style="list-style-type: none"> <li>• RootServer</li> <li>• ChunkServer</li> <li>• MergeServer</li> <li>• UpdateServer</li> <li>• Other</li> </ul> |
| svr_ip   | varchar | Server IP 地址。                                                                                                                                         |
| svr_port | int     | Server 端口。                                                                                                                                            |
| name     | varchar | 监控项的名称。                                                                                                                                               |
| value    | int     | 监控项的值。                                                                                                                                                |

### 9.3.8 \_\_all\_sys\_config

“\_\_all\_sys\_config”存储了 Server 所需的配置项参数。

Rowkey: (cluster\_id, svr\_type, svr\_ip, svr\_port, name)

“\_\_all\_sys\_config”参数说明如[表 9-10](#)所示。

**表 9-10 \_\_all\_sys\_config 参数**

| 参数         | 类型         | 说明                                                                                                                                                    |
|------------|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| gm_create  | createtime | 创建时间。                                                                                                                                                 |
| gm_modify  | modifytime | 修改时间。                                                                                                                                                 |
| cluster_id | int        | <ul style="list-style-type: none"> <li>• 0: 与特定集群无关。</li> <li>• 正整数: 指定集群。</li> </ul>                                                                 |
| svr_type   | varchar    | <ul style="list-style-type: none"> <li>• RootServer</li> <li>• ChunkServer</li> <li>• MergeServer</li> <li>• UpdateServer</li> <li>• Other</li> </ul> |
| svr_ip     | varchar    | Server IP 地址。                                                                                                                                         |

| 参数           | 类型      | 说明         |
|--------------|---------|------------|
| svr_port     | int     | Server 端口。 |
| name         | varchar | 参数名称。      |
| section      | varchar | 参数所属的段。    |
| data_type    | varchar | 参数值的数据类型。  |
| value        | varchar | 参数值。       |
| value_strict | varchar | 参数值的约束。    |
| info         | varchar | 对该项的说明。    |

### 9.3.9 \_\_all\_sys\_config\_stat

“\_\_all\_sys\_config\_stat”用于显示当前各个 Server 已经生效的配置项参数值，它的 Schema 与“\_\_all\_sys\_config”的相同。

Rowkey: (cluster\_id, svr\_type, svr\_ip, svr\_port, name)

“\_\_all\_sys\_config\_stat”参数说明如[表 9-11](#)所示。

**表 9-11 \_\_all\_sys\_config\_stat 参数**

| 参数         | 类型         | 说明                                                                                                                                          |
|------------|------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| gm_create  | createtime | 创建时间。                                                                                                                                       |
| gm_modify  | modifytime | 修改时间。                                                                                                                                       |
| cluster_id | int        | <ul style="list-style-type: none"> <li>0: 与特定集群无关。</li> <li>正整数: 指定集群。</li> </ul>                                                           |
| svr_type   | varchar    | <ul style="list-style-type: none"> <li>RootServer</li> <li>ChunkServer</li> <li>MergeServer</li> <li>UpdateServer</li> <li>Other</li> </ul> |
| svr_ip     | varchar    | Server IP 地址。                                                                                                                               |

| 参数           | 类型      | 说明         |
|--------------|---------|------------|
| svr_port     | int     | Server 端口。 |
| name         | varchar | 参数名称。      |
| section      | varchar | 参数所属的段。    |
| data_type    | varchar | 参数值的数据类型。  |
| value        | varchar | 参数值。       |
| value_strict | varchar | 参数值的约束。    |
| info         | varchar | 对该项的说明。    |

### 9.3.10 \_\_all\_sys\_param

“\_\_all\_sys\_param”存储了系统所需的诸多参数，如环境变量等，不同的参数保存在不同行。

Rowkey: (cluster\_id,name)

“\_\_all\_sys\_param”参数说明如[表 9-12](#)所示。

**表 9-12 \_\_all\_sys\_param 参数**

| 参数         | 类型         | 说明                                                                                    |
|------------|------------|---------------------------------------------------------------------------------------|
| gm_create  | createtime | 创建时间。                                                                                 |
| gm_modify  | modifytime | 修改时间。                                                                                 |
| cluster_id | int        | <ul style="list-style-type: none"> <li>0: 与特定集群无关。</li> <li>正整数: 指定的集群 ID。</li> </ul> |
| name       | varchar    | 参数名称。                                                                                 |
| data_type  | int        | 参数值的数据类型。                                                                             |
| value      | varchar    | 参数值。                                                                                  |
| info       | varchar    | 对该项的说明。                                                                               |

在“\_\_all\_sys\_param”表中已定义参数说明如[表 9-13](#)所示。

**表 9-13 已定义的参数**

| 参数                           |           | 说明                            |
|------------------------------|-----------|-------------------------------|
| name                         | data_type |                               |
| autocommit                   | int       | 是否自动提交。                       |
| character_set_results        | vchar     | 字符集。                          |
| max_allowed_packe            | int       | 最大网络包大小。                      |
| ob_app_name                  | vchar     | 应用的名称。                        |
| ob_group_agg_push_down_param | bool      | 聚合操作是否下移到<br>Chunkserver 的开关。 |
| ob_tx_idle_timeout           | int       | 事务开始后无任何操作时，事<br>务的超时时间。      |
| ob_read_consistency          | int       | 读一致性级别。                       |
| ob_tx_timeout                | int       | 事务超时时间。                       |
| tx_isolation                 | vchar     | 事务隔离性。                        |
| sql_mode                     | vchar     | SQL 模式。                       |

### 9.3.11 \_\_all\_sys\_stat

“\_\_all\_sys\_stat”存储了系统各种状态值，不同的项保存在不同行。

Rowkey: (cluster\_id, name)

“\_\_all\_sys\_stat”参数说明如[表 9-14](#)所示。

**表 9-14 \_\_all\_sys\_stat 参数**

| 参数        | 类型         | 说明    |
|-----------|------------|-------|
| gm_create | createtime | 创建时间。 |
| gm_modify | modifytime | 修改时间。 |

| 参数         | 类型      | 说明                                                                                |
|------------|---------|-----------------------------------------------------------------------------------|
| cluster_id | int     | <ul style="list-style-type: none"> <li>0: 与特定集群无关。</li> <li>正整数: 指定集群。</li> </ul> |
| name       | varchar | 参数名称。                                                                             |
| data_type  | int     | 值的类型。                                                                             |
| value      | varchar | 参数值。                                                                              |
| info       | varchar | 对参数的说明。                                                                           |

在“\_\_all\_sys\_stat”表中已经定义的参数说明如[表 9-15](#)所示。

**表 9-15** 已定义的参数

| 参数                |            |           | 说明                |
|-------------------|------------|-----------|-------------------|
| name              | cluster_id | data_type |                   |
| max_used_table_id | 0          | int       | 已经使用的最大 table_id。 |
| max_used_user_id  | 0          | int       | 已经使用的最大 user_id。  |

### 9.3.12 \_\_all\_table\_privilege

“\_\_all\_table\_privilege”记录了系统中用户在每个表的读写等权限。

Rowkey: (user\_id, table\_id)

“\_\_all\_table\_privilege”参数说明如[表 9-16](#)所示

**表 9-16** \_\_all\_table\_privilege 参数

| 参数        | 类型         | 说明                                  |
|-----------|------------|-------------------------------------|
| gm_create | createtime | 创建时间。                               |
| gm_modify | modifytime | 修改时间。                               |
| user_id   | int        | 用户内部 ID。                            |
| table_id  | int        | 表 ID, table_id = 0 时, 表示 all_table。 |



| 参数                | 类型  | 说明                   |
|-------------------|-----|----------------------|
| priv_all          | int | 是否有所有权限。             |
| priv_alter        | int | 是否有 alter table 权限。  |
| priv_create       | int | 是否有 create table 权限。 |
| priv_create_user  | int | 是否有 create user 权限。  |
| priv_delete       | int | 是否有 delete table 权限。 |
| priv_drop         | int | 是否有 drop table 权限。   |
| priv_grant_option | int | 是否有 grant 授权权限。      |
| priv_insert       | int | 是否有 insert 权限。       |
| priv_update       | int | 是否有 update 权限。       |
| priv_select       | int | 是否有 select 权限。       |
| priv_replace      | int | 是否有 replace 权限。      |

### 9.3.13 \_\_all\_trigger\_event

“\_\_all\_trigger\_event”用于记录内部通知事件。

Rowkey: (event\_ts)

“\_\_all\_trigger\_event”参数说明如[表 9-17](#)所示。

**表 9-17 \_\_all\_trigger\_event 参数**

| 参数        | 类型             | 说明          |
|-----------|----------------|-------------|
| gm_create | createtime     | 创建时间。       |
| gm_modify | modifytime     | 修改时间。       |
| event_ts  | PrecisDateTime | 事件发生时间戳。    |
| src_ip    | varchar        | 事件发生源机器 ip。 |

| 参数          | 类型      | 说明    |
|-------------|---------|-------|
| event_type  | int     | 事件类型。 |
| event_param | int     | 消息参数。 |
| extra       | varchar | 预留。   |

### 9.3.14 \_\_all\_user

“\_\_all\_user”记录了系统中所有的可以登录 OceanBase 的用户，每个用户一行记录。

Rowkey: (user\_name)

“\_\_all\_user”参数说如[表 9-18](#) 所示。

**表 9-18 \_\_all\_user 参数**

| 参数               | 类型         | 说明                   |
|------------------|------------|----------------------|
| gm_create        | createtime | 创建时间。                |
| gm_modify        | modifytime | 修改时间。                |
| user_name        | varchar    | 用户名。                 |
| user_id          | int        | 用户内部 ID。             |
| pass_word        | varchar    | 用户密码（密文存储）。          |
| info             | varchar    | 注释。                  |
| priv_all         | int        | 是否拥有所有的权限。           |
| priv_alter       | int        | 是否有 alter 权限。        |
| priv_create      | int        | 是否有 create table 权限。 |
| priv_create_user | int        | 是否有 create user 权限。  |
| priv_delete      | int        | 是否有 delete table 权限。 |
| priv_drop        | int        | 是否有 drop table 权限。   |

| 参数                | 类型  | 说明              |
|-------------------|-----|-----------------|
| priv_grant_option | int | 是否有 grant 授权权限。 |
| priv_insert       | int | 是否有 insert 权限。  |
| priv_update       | int | 是否有 update 权限。  |
| priv_select       | int | 是否有 select 权限。  |
| priv_replace      | int | 是否有 replace 权限。 |
| is_locked         | int | 是否被锁。           |

## 9.4 配置参数说明

本章节主要介绍 OceanBase 各 Server 的所有配置参数、缺省值和参数说明（参数含义、取值范围和推荐值等）。

- 查看、修改系统参数的方法请参见《OceanBase 0.4.1 SQL 参考手册》的“5.4 修改系统配置项”章节。
- DBA 需要配置和关注的各 Server 参数以及这些参数的配置方法，请参见本手册的“7 配置 OceanBase”。

### 9.4.1 RootServer 配置参数

RootServer 配置参数说明如[表 9-19](#)所示。

**表 9-19 RootServer 配置参数**

| 参数                                 | 缺省值 | 说明                                                                   |
|------------------------------------|-----|----------------------------------------------------------------------|
| balance_max_concurrent_migrate_num | 2   | 单个 ChunkServer 上迁移 SSTable 时，允许的最大的迁移数。<br>取值范围：[1,10]               |
| balance_max_migrate_out_per_cs     | 20  | 一批迁移 SSTable 组成一个任务，该值表示单个 ChunkServer 上允许的最大的迁移任务数。<br>取值范围：[1,100] |

| 参数                      | 缺省值               | 说明                                                                                                                                                                                                      |
|-------------------------|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| balance_max_timeout     | 5m                | 迁移任务的超时时间。<br>不建议修改。                                                                                                                                                                                    |
| balance_timeout_delta   | 10s               | 迁移任务超时时间允许的容错时间，即超时时间达到“balance_max_timeout + balance_timeout_delta”后，才判定任务失败。<br>不建议修改。                                                                                                                |
| balance_tolerance_count | 10                | 对单个表来说，可能保存有一个或多个 SSTable，这些 SSTable 分布在各个 ChunkServer 数量在以下区间内：<br>[SSTable 数/ChunkServer 总数 - balance_tolerance_count, SSTable 数/ChunkServer 总数 + balance_tolerance_count]<br>取值范围：[1,1000]<br>推荐值：10 |
| balance_worker_idl_time | 30s               | 检查所有表的 SSTable 是否均衡分布在 ChunkServer 上的时间间隔。<br>不建议修改。                                                                                                                                                    |
| cluster_id              | -                 | OceanBase 集群 ID。                                                                                                                                                                                        |
| commit_log_dir          | data/rs_commitlog | OceanBase 安装目录下 RootServer 的 CommitLog 目录。                                                                                                                                                              |
| commit_log_sync_type    | 1                 | RootServer 的 CommitLog 同步类型： <ul style="list-style-type: none"> <li>0：用于 CommitLog 的内存缓冲区写满后再刷入 CommitLog 文件内。</li> <li>1：在用于 CommitLog 的内存缓冲区内，每写一条日志就刷入 CommitLog 文件里。</li> </ul>                     |

| 参数                            | 缺省值  | 说明                                                                                                                                                                        |
|-------------------------------|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| cs_lease_duration_time        | 10s  | ChunkServer 的租约有效期。<br>RootServer 给 ChunkServer 发送一个租约，如果在该有效期时间内 ChunkServer 无应答，则判定 ChunkServer 不在线。<br>不建议修改。                                                          |
| cs_probation_period           | 5s   | 新 ChunkServer 上线后再该时间段内不允许 tablet 迁入，用于防止 ChunkServer 上线后马上下线的情况。                                                                                                         |
| devname                       | eth0 | 启动 RootServer 服务的网卡名称。                                                                                                                                                    |
| enable_balance                | True | 是否开启 ChunkServer 上的迁移 SSTable 任务均衡。 <ul style="list-style-type: none"> <li>• True: 开启。</li> <li>• False: 不开启。</li> </ul>                                                  |
| enable_cache_schema           | True | RootServer 会从 UpdateServer 和 ChunkServer 中读取 Schema，该参数表示在 RootServer 内是否保存 Schema 的缓存。 <ul style="list-style-type: none"> <li>• True: 是。</li> <li>• False: 否。</li> </ul> |
| enable_rereplication          | True | 是否开启 Tablet 的副本复制。如果不开启，则即使 Tablet 的副本数小于设置的“tablet_replicas_num”，也不会进行复制。 <ul style="list-style-type: none"> <li>• True: 开启。</li> <li>• False: 不开启。</li> </ul>           |
| expected_request_process_time | 10ms | 在 RootServer 和其他 Server 进行网络交互时，其他 Server 会给 RootServer 发送包，如果“RootServer 收到包到处理包的时间间隔 + 该时间 > 其他 Server 要求响应的时间”，则 RootServer 放弃该包。                                      |

| 参数                  | 缺省值               | 说明                                                                                                                                                                           |
|---------------------|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| first_meta_filename | first_tablet_meta | 用来标示是否已经执行“boot_strap”命令。                                                                                                                                                    |
| io_thread_count     | 4                 | <p>用于 Libeay 的 I/O 线程数。</p> <p>需要重新启动 RootServer 服务才能生效。</p> <p>取值范围：[1,100]</p>                                                                                             |
| is_import           | False             | <p>是否开启允许外部数据进行旁路导入。</p> <ul style="list-style-type: none"> <li>• True: 开启。</li> <li>• False: 不开启。</li> </ul>                                                                |
| is_ups_flow_control | False             | <p>主备 UpdateServer 是否按照流量分配进行读服务。</p> <ul style="list-style-type: none"> <li>• True: 主备 UpdateServer 的读服务按流量分配。</li> <li>• False: 主备 UpdateServer 的读服务按主备各 50%分配。</li> </ul> |
| lease_interval_time | 15s               | <p>主 RootServer 给备 RootServer 发送一个租约的有效期。</p> <p>不建议修改。</p>                                                                                                                  |
| lease_on            | True              | <p>主备 RootServer 间是否需要开启租约模式。</p> <ul style="list-style-type: none"> <li>• True: 开启。</li> <li>• False: 不开启。</li> </ul>                                                       |
| lease_reserved_time | 10s               | <p>主 RootServer 给备 RootServer 发送一个租约后，如果在该时间内，备 RootServer 对主 RootServer 无应答，则判定备 RootServer 不在线。</p> <p>不建议修改。</p>                                                          |

| 参数                      | 缺省值        | 说明                                                                                                                                                     |
|-------------------------|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| log_queue_size          | 100        | 主 RootServer 向备 RootServer 同步 CommitLog 时，同步任务会先进入备 RootServer 的日志任务队列，然后在备 RootServer 空闲时进行处理。该参数表示备 RootServer 中的日志任务队列存放的对大任务数。<br>取值范围：[10,100000] |
| log_replay_wait_time    | 100ms      | 备 RootServer 中，日志回放线程读取 CommitLog 的时间间隔。<br>不建议修改。                                                                                                     |
| log_sync_limit          | 40MB       | 备 RootServer 启动时，先取主 RootServer 中的日志。该参数表示取 CommitLog 的带宽上限。                                                                                           |
| log_sync_timeout        | 500ms      | 主往备同步日志的超时时间。在该时间内备 RootServer 需对主 RootServer 进行应答，否则日志同步失败。<br>不建议修改。                                                                                 |
| master_root_server_ip   | 10.10.10.2 | OceanBase 主集群 RootServer 的 VIP。                                                                                                                        |
| master_root_server_port | -          | OceanBase 主集群的 RootServer 端口。                                                                                                                          |
| max_commit_log_size     | 64MB       | RootServer 中每个 CommitLog 文件大小的最大值。当文件大小达到该值之后，则生成下一个 CommitLog 文件。文件名按“1、2、3、4...”依次生成。<br>不建议修改。                                                      |
| max_merge_duration_time | 2h         | 数据合并开始到数据合并结束的最大允许时间，超过该时间则合并失败。                                                                                                                       |
| migrate_wait_time       | 60s        | 负载均衡线程启动在且等待该时间后才开始工作。<br>不建议修改。                                                                                                                       |

| 参数                             | 缺省值     | 说明                                                                                                                                        |
|--------------------------------|---------|-------------------------------------------------------------------------------------------------------------------------------------------|
| network_timeout                | 50s     | RootServer 与其他 Server 进行网络互交的超时时间。                                                                                                        |
| port                           | 2500    | RootServer 的服务端口。<br>取值范围: (1024, 65535)                                                                                                  |
| read_master_master_ups_percent | 100     | OceanBase 主集群中主 UpdateServer 的读服务百分比，而主集群中备 UpdateServer 的读服务百分比为“(1-主 UpdateServer 的百分比)/备 UpdateServer 数量”。<br>取值范围: [0,100]<br>推荐值: 40 |
| read_queue_size                | 500     | RootServer 处理读请求任务的队列大小。<br>取值范围: [10,100000]<br>推荐值: 10000                                                                               |
| read_slave_master_ups_percent  | 100     | OceanBase 备集群中主 UpdateServer 的读服务百分比，而备集群中备 UpdateServer 的读服务百分比为“(1-主 UpdateServer 的百分比)/备 UpdateServer 数量”。<br>取值范围: [0,100]<br>推荐值: 50 |
| read_thread_count              | 20      | RootServer 处理读任务的线程数。<br>取值范围: [10,100]                                                                                                   |
| retry_times                    | 3       | RootServer 与其他 Server 网络交互失败时的重试次数。                                                                                                       |
| root_server_ip                 | -       | RootServer 的 IP。                                                                                                                          |
| rs_data_dir                    | data/rs | OceanBase 安装目录下 RootServer 的数据目录。                                                                                                         |



| 参数                              | 缺省值            | 说明                                                                                                                                    |
|---------------------------------|----------------|---------------------------------------------------------------------------------------------------------------------------------------|
| safe_lost_one_time              | 3600s          | 当 ChunkServer 下线，造成缺少一个 Tablet 副本时，进行重新复制的等待时间。<br>不建议修改。                                                                             |
| schema_filename                 | etc/schema.ini | OceanBase 安装目录下 Schame 文件的路径和名称。                                                                                                      |
| slave_register_timeout          | 3s             | 备 RootServer 启动后，向主 RootServer 进行注册的超时时间。<br>不建议修改。                                                                                   |
| tablet_migrate_disabling_period | 60s            | 刚经过迁移的 Tablet 或者刚上线的 ChunkServer 中的 Tablet 时，需要经过这段时间才可以被迁移。<br>不建议修改。                                                                |
| tablet_replicas_num             | 3              | Tablet 副本数。                                                                                                                           |
| ups_lease_reserved_time         | 8500ms         | 更新租约的保留时间。在“ups_lease_time - ups_lease_reserved_time”时间内，RootServer 向主 UpdateServer 重新发送租约。<br>不建议修改。                                 |
| ups_lease_time                  | 9s             | RootServer 给主 UpdateServer 发送的租约的有效时长。                                                                                                |
| ups_renew_reserved_time         | 7770ms         | “ups_lease_time - ups_renew_reserved_time”时间内，主 UpdateServer 未收到 RootServer 发送的租约时，主 UpdateServer 会主动向 RootServer 发送更新租约请求。<br>不建议修改。 |
| ups_waiting_register_time       | 15s            | RootServer 需要在第一个 UpdateServer 进行注册，并再等待该参数设置的时间后，才进行选主 UpdateServer。                                                                 |

| 参数               | 缺省值   | 说明                                               |
|------------------|-------|--------------------------------------------------|
| vip_check_period | 500ms | 主备 RootServer 检查 VIP 是否在自己所在服务器上的时间间隔。<br>不建议修改。 |
| write_queue_size | 100   | RootServer 处理写请求任务的队列大小。<br>取值范围：[10,100000]     |

### 9.4.2 UpdateServer 配置参数

UpdateServer 配置参数说明如[表 9-20](#)所示。

**表 9-20 UpdateServer 配置参数**

| 参数               | 缺省值    | 说明                                                                                                                                                                                                                                                                                                                                                                                                                           |
|------------------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| active_mem_limit | 系统自动生成 | <p>用户的更新操作写入 Active MemTable。当 Active MemTable 大小到达该值后，则冻结 Active MemTable，同时开启新的 Active MemTable 接受更新操作。</p> <p>推荐值和计算方法如下：</p> <ul style="list-style-type: none"> <li>app_mod=import 时<br/>推荐值：69G；<br/>计算方法：<br/><math>\text{table\_memory\_limit} / \text{minor\_num\_limit} * 0.7</math></li> <li>app_mod=oltp 时<br/>推荐值：286G；<br/>计算方法：“table_mem_limit”减去为冻结表预留内存的大小。预留大小为 table_mem_limit 的 10%，但最大为 10G。</li> </ul> |
| blockcache_size  | 系统自动生成 | <p>Block 缓存大小。该参数不可以动态改小，但是可以动态改大。</p> <p>需要重新启动 UpdateServer 服务才能生效。</p> <p>推荐值：21218420522B</p>                                                                                                                                                                                                                                                                                                                            |

| 参数                    | 缺省值                | 说明                                                                                                                                                                                                                                                                                                                                                                                |
|-----------------------|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| blockindex_cache_size | 系统自动生成             | Block 索引缓存大小。该参数不可以动态改小，但是可以动态改大。<br>需要重新启动 UpdateServer 服务才能生效。<br>推荐值：15913815391B                                                                                                                                                                                                                                                                                              |
| commit_log_dir        | data/ups_commitlog | OceanBase 安装目录下 UpdateServer 的 CommitLog 目录。<br>需要重新启动 UpdateServer 服务才能生效。                                                                                                                                                                                                                                                                                                       |
| commit_log_size       | 64MB               | UpdateServer 中每个 CommitLog 文件大小的最大值。当文件大小达到该值之后，则生成下一个 CommitLog 文件。文件名按“1、2、3、4...”依次生成。<br>需要重新启动 UpdateServer 服务才能生效。<br>不建议修改。                                                                                                                                                                                                                                                |
| consistency_type      | 2                  | 一致性 SQL 请求只能读取主 UpdateServer 的 CommitLog；弱一致性 SQL 请求读取 UpdateServer 中 CommitLog 时，需要根据该参数配置的一致性类型进行读取 CommitLog： <ul style="list-style-type: none"> <li>1: Strong，只能读取主 UpdateServer 的 CommitLog。</li> <li>2: Normal，只有当主备 UpdateServer 同步时，才允许读取备 UpdateServer 的 CommitLog；否则只能读取主 UpdateServer 的 CommitLog。</li> <li>3: Weak，可以读取主或备 UpdateServer 的 CommitLog。</li> </ul> 推荐值：3 |
| devname               | eth0               | 启动 UpdateServer 服务的网卡名称。<br>需要重启 UpdateServer 服务才能生效。                                                                                                                                                                                                                                                                                                                             |

| 参数                   | 缺省值            | 说明                                                                                                 |
|----------------------|----------------|----------------------------------------------------------------------------------------------------|
| dir_regex            | ^store[0-9]+\$ | <p>raid 目录下指向磁盘实际目录的软链接的名称匹配式。</p> <p>需要重新启动 UpdateServer 服务才能生效。</p> <p>不建议修改。</p>                |
| disk_warn_threshold  | 5ms            | <p>写 CommitLog 的超过该时间。如果超时，则产生一条 Warn 日志。</p>                                                      |
| fetch_log_wait_time  | 500ms          | <p>备 UpdateServer 从主 UpdateServer 两次读取 CommitLog 的时间间隔。</p> <p>不建议修改。</p>                          |
| fetch_schema_timeout | 3s             | <p>UpdateServer 从 RootServer 获取 schema 的超时时间。</p> <p>需要重新启动 UpdateServer 服务才能生效。</p> <p>不建议修改。</p> |
| fetch_schema_times   | 10             | <p>UpdateServer 从 RootServer 获取 schema 失败时的重试次数。</p> <p>不建议修改。</p>                                 |
| inner_port           | 2701           | <p>用于每日合并的端口号。</p> <p>需要重新启动 UpdateServer 服务才能生效。</p> <p>不建议修改。</p> <p>取值范围：(1024,65536)</p>       |
| io_thread_count      | 3              | <p>用于 Libeay 的 I/O 线程数。</p> <p>需要重新启动 UpdateServer 服务才能生效。</p>                                     |

| 参数                       | 缺省值   | 说明                                                                                                                                                                                       |
|--------------------------|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| keep_alive_timeout       | 5s    | <p>备 UpdateServer 未收到主 UpdateServer 的消息超过该值时，则重新向主 UpdateServer 注册。</p> <p>需要重新启动 UpdateServer 服务才能生效。</p> <p>不建议修改。</p>                                                                 |
| lease_queue_size         | 100   | <p>UpdateServer 从 RootServer 获取租约的任务队列长度。</p> <p>需要重新启动 UpdateServer 服务才能生效。</p>                                                                                                         |
| lease_timeout_in_advance | 500ms | <p>在租约有效期内，且该租约经过“ups_lease_time - lease_timeout_in_advance”的时间后，如果主 UpdateServer 还没有收到 RootServer 发送的新租约，则主 UpdateServer 将不再接受写事务。</p> <p>需要重新启动 UpdateServer 服务才能生效。</p> <p>不建议修改。</p> |
| log_cache_block_size     | 32MB  | <p>用于备 UpdateServer 接收主 UpdateServer 的 CommitLog 的缓存块大小。</p> <p>需要重新启动 UpdateServer 服务才能生效。</p>                                                                                          |
| log_cache_n_block        | 4     | <p>用于备 UpdateServer 接收主 UpdateServer 的 CommitLog 的缓存块的数量。</p> <p>需要重新启动 UpdateServer 服务才能生效。</p>                                                                                         |
| log_queue_size           | 100   | <p>CommitLog 同步任务队列长度。</p> <p>需要重新启动 UpdateServer 服务才能生效。</p>                                                                                                                            |

| 参数                                  | 缺省值   | 说明                                                                                                                                                                                                                              |
|-------------------------------------|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| log_sync_delay_warn_report_interval | 10s   | 备 UpdateServer 接收主 UpdateServer 的 CommitLog 延迟时，两次产生报警的最小时间间隔。                                                                                                                                                                  |
| log_sync_delay_warn_time_threshold  | 500ms | 备 UpdateServer 两次接收主 UpdateServer 的 CommitLog 的间隔时间超过了该值，则产生报警。                                                                                                                                                                 |
| log_sync_retry_times                | 2     | 主 UpdateServer 向备 UpdateServer 发送 CommitLog 失败时的重试次数。<br>不建议修改。                                                                                                                                                                 |
| log_sync_timeout                    | 500ms | 主 UpdateServer 向备 UpdateServer 发送 CommitLog 的超时时间。                                                                                                                                                                              |
| log_sync_type                       | 1     | <p>主机发备机 CommitLog 时，写入磁盘的方式：</p> <ul style="list-style-type: none"> <li>• 0：用于 CommitLog 的内存缓冲区写满后再刷入 CommitLog 文件内。</li> <li>• 1：在用于 CommitLog 的内存缓冲区内，每写一条日志就刷入 CommitLog 文件里。</li> </ul> <p>需要重新启动 UpdateServer 服务才能生效。</p> |
| lsync_fetch_timeout                 | 5s    | <p>备 UpdateServer 从 LsyncServer 或主 UpdateServer 读取 CommitLog 的超时时间。</p> <p>LsyncServer 是一个假主机（实际上是一个服务进程），主要提供 CommitLog，用于备 UpdateServer 读取。</p> <p>需要重新启动 UpdateServer 服务才能生效。</p>                                            |

| 参数                         | 缺省值                           | 说明                                                                                                                                         |
|----------------------------|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| lsync_ip                   | 0.0.0.0                       | LsyncServer 的 IP 地址。如果配了这个地址，则备 UpdateServer 从 LsyncServer 读取 CommitLog；如果未配置，则备 UpdateServer 从主 UpdateServer 那读取 CommitLog。               |
| lsync_port                 | 3000                          | 从 LsyncServer 读取 CommitLog 的同步监听端口。<br>取值范围：(1024,65536)                                                                                   |
| major_freeze_duty_time     | Disable,<br>OB_CONFIG_DYNAMIC | 每天定时升级主版本的冻结操作的时间。                                                                                                                         |
| max_n_lagged_log_allowed   | 10000                         | 备 UpdateServer 与主 UpdateServer 的日志延迟的条数超过了该值时，则产生报警。                                                                                       |
| max_row_cell_num           | 256                           | 在 Memtable 中，当一行中的 Cell 数量超过该值时，则执行一次合并。                                                                                                   |
| memtable_hash_buckets_size | 396867876B                    | Hash 索引大小。<br>需要重新启动 UpdateServer 服务才能生效。<br>不建议修改。                                                                                        |
| min_major_freeze_interval  | 1s                            | 两次升级主版本的最小时间间隔。如果小于该值，则执行主版本冻结失败。                                                                                                          |
| minor_num_limit            | 系统自动生成                        | 小版本的个数大于或等于该值后，如果再次执行冻结，则执行主版本冻结。<br>推荐值如下： <ul style="list-style-type: none"> <li>app_mod=import 时：3</li> <li>app_mod=oltp 时：1</li> </ul> |
| net_warn_threshold         | 5ms                           | 备 UpdateServer 向主 UpdateServer 同步 CommitLog 的网络超时超过该值时，产生告警。                                                                               |

| 参数                          | 缺省值           | 说明                                                                             |
|-----------------------------|---------------|--------------------------------------------------------------------------------|
| packet_max_wait_time        | 10s           | 通用网络请求超时。<br>需要重新启动 UpdateServer 服务才能生效。<br>不建议修改。                             |
| port                        | 2700          | UpdateServer 的服务端口。<br>需要重启 UpdateServer 服务才能生效。                               |
| raid_regex                  | ^raid[0-9]+\$ | raid 目录名的匹配式。<br>需要重新启动 UpdateServer 服务才能生效。<br>不建议修改。                         |
| read_queue_size             | 1000          | 处理管理命令线程任务队列大小。<br>需要重新启动 UpdateServer 服务才能生效。                                 |
| read_thread_count           | 4             | 用于管理命令任务的最大线程数。<br>需要重新启动 UpdateServer 服务才能生效。                                 |
| refresh_lsync_addr_interval | 60s           | 两次更新 LsyncServer 地址的时间间隔。                                                      |
| register_timeout            | 3s            | UpdateServer 向 RootServer 注册的超时时间。<br>需要重新启动 UpdateServer 服务才能生效。<br>不建议修改。    |
| register_times              | 10            | UpdateServer 向 RootServer 注册失败时的重试次数。<br>需要重新启动 UpdateServer 服务才能生效。<br>不建议修改。 |



| 参数                   | 缺省值   | 说明                                                                                                                   |
|----------------------|-------|----------------------------------------------------------------------------------------------------------------------|
| replay_checksum_flag | True  | <p>日志回放时，是否对 MemTable 进行校验和检查。</p> <ul style="list-style-type: none"> <li>• True: 是。</li> <li>• False: 否。</li> </ul> |
| replay_log_buf_size  | 10GB  | <p>用于 CommitLog 重放的缓冲区大小。</p> <p>不建议修改。</p>                                                                          |
| replay_queue_len     | 500   | <p>CommitLog 重放时，会先将任务放入一个队列。该参数表示用于 CommitLog 重放任务的最大队列长度。</p> <p>不建议修改。</p>                                        |
| replay_wait_time     | 100ms | <p>两次 CommLog 重放的时间间隔。</p> <p>不建议修改。</p>                                                                             |
| replay_worker_num    | 20    | CommLog 重放线程数。                                                                                                       |
| resp_root_timeout    | 1s    | <p>UpdateServer 向 RootServer 汇报冻结数据版本的超时时间。</p> <p>需要重新启动 UpdateServer 服务才能生效。</p> <p>不建议修改。</p>                     |
| resp_root_times      | 20    | <p>UpdateServer 向 RootServer 汇报冻结数据版本失败时的重试次数。</p> <p>需要重新启动 UpdateServer 服务才能生效。</p> <p>不建议修改。</p>                  |
| retry_times          | 3     | UpdateServer 与其他 Server 进行网络交互失败时的重试次数。                                                                              |
| root_server_ip       | -     | UpdateServer 所在集群的主 RootServer 的 IP。                                                                                 |

| 参数                         | 缺省值           | 说明                                                                            |
|----------------------------|---------------|-------------------------------------------------------------------------------|
| root_server_port           | 2500          | UpdateServer 所在集群的主 RootServer 的服务端口。<br>取值范围：(1024,65535)                    |
| sstable_block_size         | 4K            | 从内存转储到磁盘的 SSTable 的 Block 大小。                                                 |
| sstable_compressor_name    | none          | SSTable 从内存转储到磁盘使用的压缩库。                                                       |
| sstable_time_limit         | 7d            | 当 SSTable 加载的时间达到该值时，将被转入<br>“\$OB_INSTALL/data/ups_data/raid2/store0/trash”。 |
| state_check_period         | 500ms         | 检查 UpdateServer 主备、是否可服务等内部状态的周期。<br>不建议修改。                                   |
| store_queue_size           | 100           | 用于 SSTable 从内存转储到磁盘的线程队列长度。                                                   |
| store_root                 | data/ups_data | OceanBase 安装目录下 UpdateServer 的数据目录。<br>需要重新启动 UpdateServer 服务才能生效。<br>不建议修改。  |
| store_thread_count         | 3             | 用于 SSTable 从内存转储到磁盘的线程个数。<br>需要重新启动 UpdateServer 服务才能生效。                      |
| table_available_error_size | 系统自动生成        | MemTable 可用内存小于该值时，则打印 Error 日志。<br>推荐值：10G                                   |

| 参数                         | 缺省值    | 说明                                                                                                                                                                                                                                                 |
|----------------------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| table_available_warn_size  | 系统自动生成 | MemTable 可用内存小于该值时，则打印 Warn 日志。<br>推荐值：8G                                                                                                                                                                                                          |
| table_memory_limit         | 系统自动生成 | MemTable 可用内存。<br>计算方法：(total_memory_limit - total_reserve) / (1/20 + 1/15 + 1)，一般情况下 total_reserve = 10G<br>推荐值：318276307830B<br><i>说明：UpdateServer 全局内存包括 MemTable、SSTable Cache、事务 Session 和其他。<br/>“total_reserve_gb”为事务 Session 与其他预留的内存。</i> |
| total_memory_limit         | 系统自动生成 | UpdateServer 可用内存。<br>推荐值：361850994688B                                                                                                                                                                                                            |
| trans_proc_time_warn       | 1s     | 处理 SQL 读写请求的超时时间。（暂时不起作用）<br>不建议修改。                                                                                                                                                                                                                |
| trans_thread_num           | 30     | 处理 SQL 读写请求的工作线程数。                                                                                                                                                                                                                                 |
| using_hash_index           | True   | 是否使用 Hash 索引。<br><ul style="list-style-type: none"> <li>True：是。</li> <li>False：否。</li> </ul>                                                                                                                                                       |
| using_memtable_bloomfilter | False  | 是否使用 Bloomfilter。<br><ul style="list-style-type: none"> <li>True：是。</li> <li>False：否</li> </ul>                                                                                                                                                    |
| using_static_cm_column_id  | False  | 是否使用内置 ID 作为“create_time”和“modify_time”列的 ID：<br><ul style="list-style-type: none"> <li>True：是。</li> <li>False：否。</li> </ul>                                                                                                                       |

| 参数                    | 缺省值  | 说明                                                                                                                                                                                                                                                        |
|-----------------------|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| wait_slave_sync_type  | 0    | <p>备 UpdateServer 回放 CommitLog 时，应答主 UpdateServer 的时机：</p> <ul style="list-style-type: none"> <li>• 0：在 CommitLog 回放前，应答主 UpdateServer。</li> <li>• 1：在 CommitLog 回放后，且写入磁盘前，应答主 UpdateServer。</li> <li>• 2：在 CommitLog 写入磁盘后，应答主 UpdateServer。</li> </ul> |
| warm_up_time          | 10m  | <p>SSTable 的预热缓存时间。</p> <p>取值范围：[10s,1800s]</p>                                                                                                                                                                                                           |
| write_sstable_use_dio | True | <p>是否使用 DIO(Direct IO)方式写 SSTable。</p> <ul style="list-style-type: none"> <li>• True：直接写入磁盘。</li> <li>• False：先写入缓存，再写入磁盘。</li> </ul>                                                                                                                     |

### 9.4.3 MergeServer 配置参数

MergeServer 配置参数说明如[表 9-21](#)所示。

**表 9-21 MergeServer 配置参数**

| 参数                   | 缺省值  | 说明                                                           |
|----------------------|------|--------------------------------------------------------------|
| devname              | eth0 | 启动 MergeServer 服务的网卡名称。                                      |
| io_thread_count      | 12   | <p>用于内部任务端口的 I/O 线程数。</p> <p>取值范围：[1,∞)</p>                  |
| lease_check_interval | 6s   | <p>检查与 RootServer 租约是否失效的时间间隔。</p> <p>不建议修改。</p>             |
| location_cache_size  | 32MB | MergeServer 从 RootServer 中获取 Tablet 的位置信息，并缓存到本地。该参数表示最大缓存值。 |

| 参数                           | 缺省值  | 说明                                                                                                                                                                                                                                                                                                        |
|------------------------------|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| location_cache_timeout       | 600s | MergeServer 从 RootServer 中获取 Tablet 的位置信息，并缓存到本地的超时时间。<br>不建议修改。                                                                                                                                                                                                                                          |
| max_get_rows_per_subrequest  | 20   | MergeServer 向 ChunkServer 发送的每个 get 请求最多包含的行数。如果设置为“0”，MergeServer 向 ChunkServer 发送的每个 get 请求最多包含的行数不受限制。<br>该参数设置较小时，可以获得更小的响应时间，但会增加 MergeServer 向 ChunkServer 发起 RPC 次数，从而减少整个 OB 系统的 QPS；该参数设置较大时，MergeServer 处理 get 请求的响应时间决定于 MergeServer 发送给 ChunkServer 的最大的一个 get 请求的耗时，响应时间可能比较长。<br>取值范围：[0,∞) |
| max_parallel_count           | 16   | 每个请求同一时刻最多并发执行的子请求数量，即一个请求同时最多可有多少个 ChunkServer 线程并发执行。<br>取值范围：[1,∞)                                                                                                                                                                                                                                     |
| memory_size_limit_percentage | 40   | 在 MergeServer 所在服务器的物理内存中，可用于 MergeServer 的最大百分比数。<br>取值范围：(0,100]<br>推荐值：40                                                                                                                                                                                                                              |
| monitor_interval             | 600s | MergeServer 执行两次内部定时任务（如打印日志等）的时间间隔。                                                                                                                                                                                                                                                                      |
| network_timeout              | 2s   | MergeServer 与其他 Server 进行网络互交的超时时间。<br>推荐值：3s                                                                                                                                                                                                                                                             |

| 参数                        | 缺省值   | 说明                                                                                                     |
|---------------------------|-------|--------------------------------------------------------------------------------------------------------|
| obmysql_io_thread_count   | 8     | 用于 MySQL 端口的 I/O 最大线程数。<br>取值范围: [1,∞)                                                                 |
| obmysql_port              | 2880  | MySQL 服务端口。<br>取值范围: (1024,65536)                                                                      |
| obmysql_task_queue_size   | 10000 | 用于 SQL 操作的任务队列的最大值。<br>取值范围: [1,∞)                                                                     |
| obmysql_work_thread_count | 120   | 用于执行 SQL 任务的最大线程数。<br>取值范围: [1,∞)                                                                      |
| port                      | 2800  | MergeServer 内部任务的服务端口。                                                                                 |
| query_cache_size          | 0     | 查询缓存允许使用的内存大小。该值为“0”时，表述不启用查询缓存；大于“0”时，启用。<br>取值范围: [1,∞)                                              |
| retry_times               | 3     | MergeServer 向其他 Server 进行网络交互失败时的重试次数。                                                                 |
| root_server_ip            | -     | MergeServer 所在集群的主 RootServer 的 IP。                                                                    |
| root_server_port          | 3500  | RootServer 的服务端口。<br>取值范围: (1024,65535)                                                                |
| slow_query_threshold      | 100ms | SQL 语句查询时间超过该值时，则标示此次查询为慢查询。                                                                           |
| task_left_time            | 100ms | 请求预留给 MergeServer 的处理时间。<br>如果“一个请求的超时时间 - 在 packet quene 中的等待时间 < task_left_time”，则放弃处理该请求。<br>不建议修改。 |

| 参数                | 缺省值   | 说明                                                                                                                                                                                          |
|-------------------|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| task_queue_size   | 10000 | 用于 MergeServer 内部任务的任务队列最大值。<br>不建议修改。<br>取值范围：[1,∞)                                                                                                                                        |
| task_thread_count | 10    | 用于处理内部任务的最大线程数。                                                                                                                                                                             |
| timeout_percent   | 70    | MergeServer 给 ChunkServer 发 SQL 请求时，如果该 SQL 请求的内部超时时间为 100ms，那么 MergeServer 发送给 ChunkServer 的超时时间为“SQL 请求的内部超时时间 * timeout_percent”，即“100ms*70%”；剩余时间预留，用于重试其他 ChunkServer。<br>取值范围：[10,80] |

#### 9.4.4 ChunkServer 配置参数

ChunkServer 配置参数说明如[表 9-22](#)所示。

**表 9-22 ChunkServer 配置参数**

| 参数               | 缺省值 | 说明                                                                                                       |
|------------------|-----|----------------------------------------------------------------------------------------------------------|
| appname          | -   | OceanBase 启动时指定的 App 名称。                                                                                 |
| block_cache_size | 1GB | Block 缓存大小。类似于 Oracle 的 db cache。配置值越大越好，但是不可超过 MergeServer 和 ChunkServer 总内存大小。<br>取值范围：(0,∞)<br>推荐值：1G |

| 参数                               | 缺省值                     | 说明                                                                                                                                                                                                                                                                                                              |
|----------------------------------|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| block_index_cache_size           | 512MB                   | <p>Block 索引缓存大小，主要保存每个 Block 的索引数据。</p> <p>计算方法：(Disk Size / Block Size) * Block Entry Size</p> <p>Block 的大小一般为 4KB~64KB，每个 Block 的管理开销是：20~30Byte+一个 Rowkey 长度，假设 Rowkey 为 50 个 Byte，则一个 Block 的管理成本 70-80byte，如果 ChunkServer 存储 1T 的数据，那么索引的管理成本是“(1T/64k)*80Byte=1.28G”。</p> <p>取值范围：(0,∞)</p> <p>推荐值：4G</p> |
| bypass_sstable_loader_thread_num | 0                       | <p>旁路导入的线程数。当值为“0”时，表示不启用旁路导入。</p> <p>取值范围：[0,10]</p>                                                                                                                                                                                                                                                           |
| check_compress_lib               | snappy_1.0:none:lzo_1.0 | ChunkServer 启动时检查使用的压缩库。                                                                                                                                                                                                                                                                                        |
| datadir                          | /data                   | SStable 存放路径。配置值为绝对路径，不支持相对路径。                                                                                                                                                                                                                                                                                  |
| devname                          | bond0                   | 启动 ChunkServer 使用的网卡名称。                                                                                                                                                                                                                                                                                         |
| each_tablet_sync_meta            | True                    | <p>每合并一个 Tablet 是否都将索引文件写入磁盘。</p> <ul style="list-style-type: none"> <li>• True: 是。</li> <li>• False: 合并所有 Tablet 后或当 ChunkServer 退出时写入磁盘。</li> </ul>                                                                                                                                                           |
| fetch_ups_interval               | 5s                      | ChunkServer 从 RootServer 中读取 UpdateServer 地址列表的时间间隔                                                                                                                                                                                                                                                             |
| file_info_cache_num              | 4096                    | <p>文件句柄缓存个数。</p> <p>取值范围：(0,∞)</p>                                                                                                                                                                                                                                                                              |



| 参数                     | 缺省值   | 说明                                                                                                                                   |
|------------------------|-------|--------------------------------------------------------------------------------------------------------------------------------------|
| groupby_mem_size       | 8MB   | 使用“Group By”操作符时，每次允许的最大内存。                                                                                                          |
| io_thread_count        | 4     | 用于 libeasy 的 I/O 线程数。<br>取值范围：[1,∞)                                                                                                  |
| join_batch_count       | 3000  | 使用“JION”操作符时，允许的最大的数据条数。<br>取值范围：(0,∞)                                                                                               |
| join_cache_size        | 512MB | 使用“JION”操作符缓存大小。                                                                                                                     |
| lazy_load_sstable      | True  | ChunkServer 启动时，是否立即装载 SS Table。<br><ul style="list-style-type: none"> <li>• True: 是。</li> <li>• False: 读取数据是才装载 SSTable。</li> </ul> |
| lease_check_interval   | 5s    | 检查与 RootServer 租约是否失效的时间间隔。<br>不建议修改。<br>取值范围：[5s,5s]                                                                                |
| max_merge_thread_num   | 10    | 每日合并的最大线程数<br>取值范围：[1,32]                                                                                                            |
| max_migrate_task_count | 2     | Tablet 迁移的最大线程数。<br>取值范围：[1,∞)                                                                                                       |
| max_version_gap        | 3     | 如果 ChunkServer 本地版本与 RootServer 的最后一次冻结版本相差超过该值，则放弃合并本地数据，等待 RootServer 复制。<br>取值范围：[1,∞)                                            |

| 参数                        | 缺省值   | 说明                                                                                                                              |
|---------------------------|-------|---------------------------------------------------------------------------------------------------------------------------------|
| merge_adjust_ratio        | 80    | 如果“ChunkServer 负载 > merge_load_high * (1 + merge_adjust_ratio)”时，则挂起当前合并线程。<br>该值为百分数。                                          |
| merge_delay_for_lsync     | 5s    | 每日合并开始时，如果需要读取备 UpdateServer 数据，则需要备 UpdateServer 均冻结 SSTable。该值表示主 UpdateServer 等待备 UpdateServer 冻结 SSTable 的时间。<br>取值范围：(0,∞) |
| merge_delay_interval      | 600s  | 当收到新版本数据后，需要等待该时间后才开始合并。<br>取值范围：(0,∞)<br>推荐值：600s                                                                              |
| merge_highload_sleep_time | 2s    | ChunkServer 负载线程超过“merge_threshold_load_high”时的 sleep 时间。                                                                       |
| merge_mem_limit           | 64MB  | 每个合并线程使用的内存大小。                                                                                                                  |
| merge_mem_size            | 8MB   | 每个查询可能经过多轮 merge 操作。每轮 merge 中，如果存储数据的 cell array 的内存大于该值，本轮 merge 操作完成。                                                        |
| merge_migrate_concurrency | False | 是否允许同时进行数据合并和数据迁移。<br><ul style="list-style-type: none"> <li>• True: 允许。</li> <li>• False: 不允许。</li> </ul>                      |
| merge_pause_row_count     | 2000  | 合并减速选项。当 merge 的数据达到该值的行数后，进行一次 merge 检查。                                                                                       |
| merge_pause_sleep_time    | 0     | 每日合并数据达到“merge_pause_row_count”行后，进行 sleep 的微秒数。<br>单位：微秒                                                                       |

| 参数                            | 缺省值  | 说明                                                                                                                                                           |
|-------------------------------|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| merge_scan_use_preread        | True | <p>当进行每日合并时，是否采用异步 I/O 机制。</p> <ul style="list-style-type: none"> <li>• True: 是。</li> <li>• False: 否。</li> </ul>                                             |
| merge_thread_per_disk         | 2    | <p>每个 disk 的合并线程。线程数越多，每日合并速度越快，但查询响应越慢。</p> <p>不建议配置这个选项。</p> <p>取值范围: <math>[1, \infty)</math></p>                                                         |
| merge_threshold_load_high     | 16   | <p>每日合并时，当 ChunkServer 负载线程超过该值，且每秒 get 或 scan 请求的次数超过“merge_threshold_request_high”时，则暂停部分合并线程。</p> <p>取值范围: <math>[1, \infty)</math></p> <p>推荐值: 10</p>    |
| merge_threshold_request_high  | 3000 | <p>每日合并时，每秒 get 或 scan 请求的最大值次数。如果每秒 get 或 scan 请求数超过该值，且 ChunkServer 负载线程超过“merge_threshold_load_high”，则暂停部分合并线程。</p> <p>取值范围: <math>[1, \infty)</math></p> |
| merge_timeout                 | 10s  | <p>在数据合并时，读取 UpdateServer 数据的超时时间。</p> <p>取值范围: <math>(0, \infty)</math></p> <p>推荐值: 30s</p>                                                                 |
| merge_write_sstable_version   | 2    | <p>数据合并后，新 SSTable 的版本。</p> <p>取值范围: <math>[1, \infty)</math></p>                                                                                            |
| migrate_band_limit_per_second | 50MB | SSTable 迁移的最大带宽。                                                                                                                                             |
| min_drop_cache_wait_time      | 300s | 数据合并完成后，原版本数据的保留时间。                                                                                                                                          |

| 参数                         | 缺省值   | 说明                                                                                                                                                              |
|----------------------------|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| min_merge_interval         | 10s   | 两次合并最小时间间隔。单位：秒。                                                                                                                                                |
| network_timeout            | 3s    | ChunkServer 与其他 Server 进行网络互交的超时时间。                                                                                                                             |
| over_size_percent_to_split | 50    | 当前合并的 SSTable 的大小达到“ $\text{max\_stable\_size} * (1 + \text{over\_size\_percent\_to\_split})$ ”时，才允许分裂该 SSTable。参数可以防止分裂出太多很小的 Tablet。<br>该值为百分数。<br>取值范围：(0,∞) |
| port                       | 2600  | ChunkServer 的服务端口。                                                                                                                                              |
| retry_times                | 3     | ChunkServer 向其他 Server 进行网络互交失败时的重试次数。                                                                                                                          |
| root_server_ip             | -     | ChunkServer 所在集群的主 RootServer 的 IP。                                                                                                                             |
| root_server_port           | 2500  | ChunkServer 所在集群的主 RootServer 的服务端口。                                                                                                                            |
| slow_query_warn_time       | 500ms | 鉴定为慢查询的超时时间。如果查询时间超过本选项设置的阈值，ChunkServer 打印一条慢查询日志。                                                                                                             |
| sstable_row_cache_size     | 2GB   | SSTable 的行缓存大小。<br>取值范围：(0,∞)<br>推荐值：20G                                                                                                                        |
| switch_cache_after_merge   | False | 每日合并完成后，旧版本 Block 的缓存是否迁移成新版本 Block 的缓存。<br><ul style="list-style-type: none"> <li>True: 是。</li> <li>False: 否。</li> </ul>                                       |

| 参数                    | 缺省值   | 说明                                                                                                                                                                |
|-----------------------|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| task_left_time        | 300ms | 请求预留给 MergeServer 的处理时间。<br>如果“一个请求的超时时间 - 在 packet queue 中的等待时间 < task_left_time”，则放弃处理该请求。                                                                      |
| task_queue_size       | 10000 | ChunkServer 中，读任务队列大小。<br>取值范围：[1000,∞)                                                                                                                           |
| task_thread_count     | 20    | 单个 ChunkServer 中允许的处理线程总数，OLAP 应用中建议配置为核心的 2 倍左右。<br>取值范围：[1,∞)<br>推荐值：40                                                                                         |
| unmerge_if_unchanged  | True  | 未修改的 SSTable 是否需要参与每日合并。选项需要所有 ChunkServer 在非合并期间统一修改，严禁合并过程中修改并重新加载。<br><ul style="list-style-type: none"> <li>• True: 不需要合并。</li> <li>• False: 需要合并。</li> </ul> |
| ups_blacklist_timeout | 5s    | 如果该 Update Server 在黑名单中的时间超过该值时，则该 UpdateServer 标示为可用状态，并从黑名单中移除。                                                                                                 |
| ups_fail_count        | 100   | 如果连接 UpdateServer 失败次数超过该值时，将该 UpdateServer 加入到黑名单。<br>取值范围：[1,∞)                                                                                                 |
| write_sstable_use_dio | True  | 是否使用 DIO 进行写 SSTable。<br><ul style="list-style-type: none"> <li>• True: 是。</li> <li>• False: 不是。</li> </ul>                                                       |