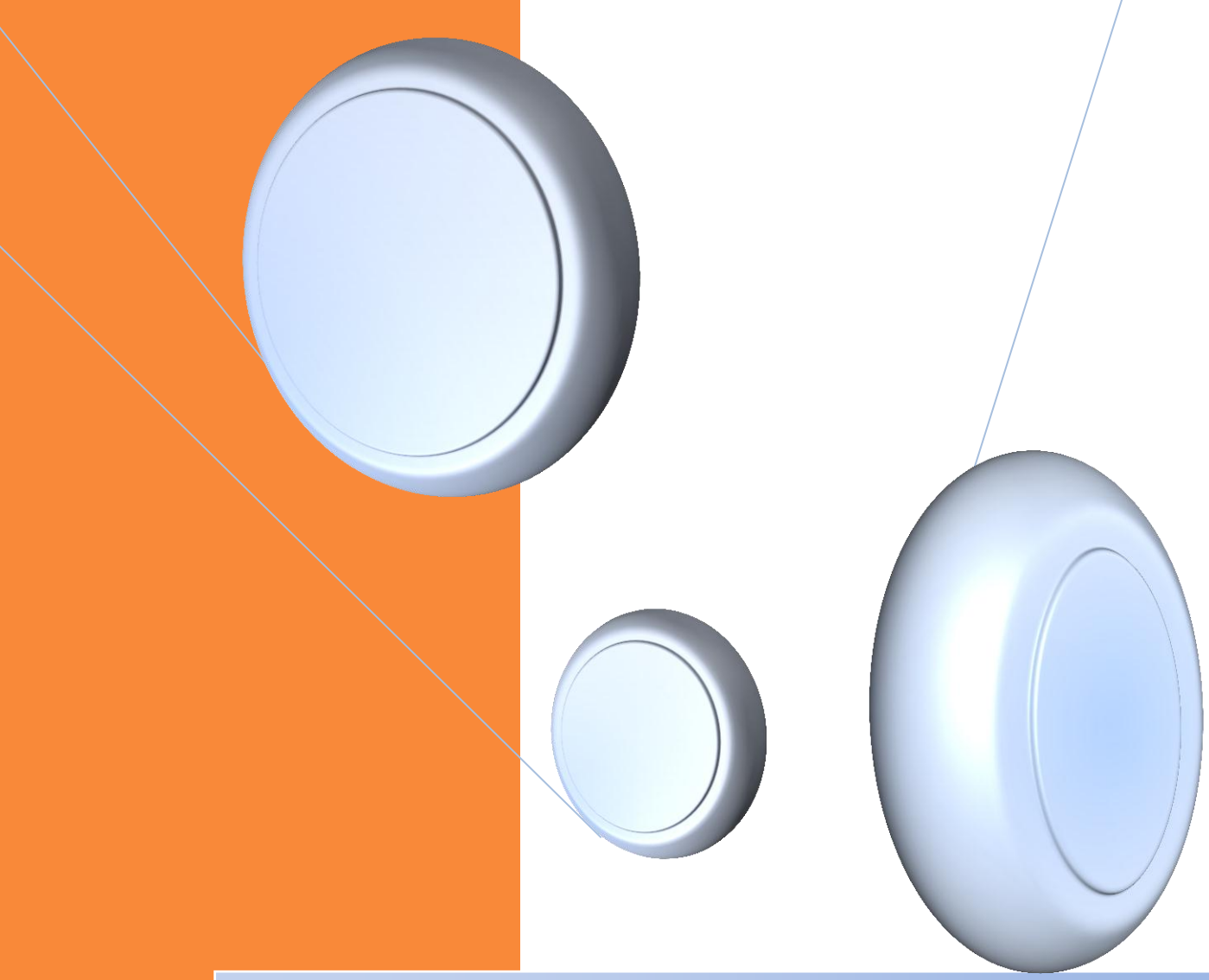

OceanBase 0.4.2
参考指南

文档版本：Beta 02

发布日期：2013.11.30

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 0.4.2的日志、系统结果码、内部表参考、配置项参考、监控项参考和术语等信息。

读者对象

本文档主要适用于：

- 安装工程师。
- 维护工程师。
- 开发工程师。
- 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。
粗体	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。

版本和发布日期	说明
Beta 02（2013-11-30）	第一次发布Beta版本，适用于OceanBase 0.4.2。
01（2013-10-30）	第一次正式发布，适用于OceanBase 0.4.1。

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

新浪微博：http://weibo.com/u/2356115944

技术交流群（阿里旺旺）：853923637

目 录

1 日志参考

OceanBase 在运行过程中会自动生成日志。维护工程师通过查看和分析日志，可以了解 OceanBase 的启动和运行状态。

1.1 日志概述

主要介绍日志清单、日志级别、查看日志的方法和收集日志的方法等。

1.1.1 日志清单

列出了 OceanBase 运行日志和操作日志的清单。

OceanBase 日志清单如表 1-1 所示。

表 1-1 日志清单

服务类型	日志名称	日志路径	说明
RootServer	rootserver.log	RootServer 所在 OceanBase 服务器的“~/oceanbase/log”目录下。	记录 RootServer 启动过程以及启动后的运行情况。
	1、2、3、…	RootServer 所在 OceanBase 服务器的“~/oceanbase/data/rs_commitlog”目录下。	RootServer 的 commit 日志，记录 RootServer 的 Tablet 信息和 schema 信息。
UpdateServer	updateserver.log	UpdateServer 所在 OceanBase 服务器的“~/oceanbase/log”目录下。	记录 UpdateServer 启动过程以及启动后的运行情况。

服务类型	日志名称	日志路径	说明
	1、2、3、…	UpdateServer 所在 OceanBase 服务器的“~/oceanbase/data/ups_commitlog”目录下。	UpdateServer 的 commit 日志，记录 UpdateServer 的 mutator 信息。
MergeServer	mergeserver.log	MergeServer 所在 OceanBase 服务器的“~/oceanbase/log”目录下。	记录 MergeServer 启动过程以及启动后的运行情况。
ChunkServer	chunkserver.log	所在 OceanBase 服务器的“~/oceanbase/log”目录下。	记录 ChunkServer 启动过程以及启动后的运行情况。
-	rs_admin.log	执行 **rs_admin** 命令的 OceanBase 服务器的“~/oceanbase”目录下。	记录 bin/rs_admin 的执行日志。

1.1.2 日志级别

介绍 OceanBase 日志的级别及每类级别的含义。当启动或运行异常时，可维护工程师可以通过日志级别和内容分析和定位异常原因。

OceanBase 的运行日志中的级别及含义如表 1-2 所示。日志级别从高到底排列。

表 1-2 日志级别及含义

日志级别	含义
ERROR	严重错误，用于记录系统的故障信息，且必须进行故障排除，否则系统不可用。
WARN	警告，用于记录可能会出现的潜在错误。
INFO	提示，用于记录系统运行的当前状态，该信息为正常信息。

日志级别	含义
DEBUG	调试信息，用于调试时更详细的了解系统运行状态，包括当前调用的函数名、参数、变量、函数调用返回值等。

1.1.3 查看日志

介绍了查看 OceanBase 日志的方法。

本文以查看 RootServer 为例，介绍查看 OceanBase 日志方法。

假设 RootServer 所在的 OceanBase 服务器 IP 为“10.10.10.2”，查看日志的步骤如下：

1. 登录 OceanBase 服务器（10.10.10.2）。
2. 执行以下命令，进入日志文件所在的目录。
   cd ~/oceanbase/log
3. 执行以下命令，查看已写入的日志记录。
   more rootserver.log
   或者执行以下命令，查看正在写入的日志记录。
   tail rootserver.log

1.1.4 收集日志

介绍了收集 OceanBase 日志的方法。建议日志收集命名格式为：“日志名称_收集日期.tar”。

本文以收集 RootServer 和 UpdateServer 日志为例，介绍收集 OceanBase 日志的方法。

假设 RootServer 和 UpdateServer 合设的 OceanBase 服务器 IP 为“10.10.10.2”，收集 OceanBase 日志的操作步骤如下：

1. 登录 OceanBase 服务器（10.10.10.2）。
2. 执行以下命令，进入日志文件所在的目录。
   cd ~/oceanbase/log
3. 执行以下命令，收集日志。
   tar cvf rs_and_ups_130809.tar rootserver.log updateserver.log
   说明：*如果需要收集当前目录下的所有日志，可执行* ***tar cvf 日志名称.tar ****命令。*

如果您需要求助支付宝 OceanBase 团队，请发送日志文件到邮箱：alipay-oceanbase-support@list.alibaba-inc.com。

1.2 启动和运行日志

记录 OceanBase 各 Server 启动过程以及启动后的运行情况。这是我们在日常工作中主要查看的日志。

*** 日志说明**

OceanBase 启动和运行日志的名称、路径以及说明如表 1-3 所示。

表 1-3 日志说明

服务类型	日志名称	日志路径	说明
RootServer	rootserver.log	RootServer 所在 OceanBase 服务器的“~/oceanbase/log”目录下。	记录 RootServer 启动过程以及启动后的运行情况。
UpdateServer	updateserver.log	UpdateServer 所在 OceanBase 服务器的“~/oceanbase/log”目录下。	记录 UpdateServer 启动过程以及启动后的运行情况。
MergeServer	mergeserver.log	MergeServer 所在 OceanBase 服务器的“~/oceanbase/log”目录下。	记录 MergeServer 启动过程以及启动后的运行情况。
ChunkServer	chunkserver.log	ChunkServer 所在 OceanBase 服务器的“~/oceanbase/log”目录下。	记录 ChunkServer 启动过程以及启动后的运行情况。

*** 日志格式**

日志记录主要由五部分组成：记录时间、日志级别、文件名:行号、线程 ID 和日志内容。

*** 日志样例**

```
[2013-08-09 10:02:18.339314] INFO  ob_root_worker.cpp:3159 [139702581581568] start to boot strap
```

解释：OceanBase 集群进行“boot srtap”初始化。

1.3 commit 日志

记录 RootServer 的 Tablet 信息和 schema 信息和 UpdateServer 的 mutator 信息。

UpdateServer 每次进行更新操作时，均会产生一条 commit log，并追加到已有的 commit log 序列之后。UpdateServer 通过同步 commit log 实现主备复制。

*** 日志说明**

OceanBase 的 commit 日志的名称、路径和说明如表 1-4 所示。

表 1-4 日志说明

服务类型	日志名称	日志路径	说明
RootServer	1、2、3、…	RootServer 所在 OceanBase 服务器的“~/oceanbase/data/rs_commitlog”目录下。	RootServer 的 commit 日志，记录 RootServer 的 Tablet 信息和 schema 信息。
UpdateServer	1、2、3、…	UpdateServer 所在 OceanBase 服务器的“~/oceanbase/data/ups_commitlog”目录下。	UpdateServer 的 commit 日志，记录 UpdateServer 的 mutator 信息。

*** 日志格式**

由于 commit log 不用于用户查看和定位系统问题，因此日志格式和样例暂不做举例和说明。

如果您需要了解 commit log 的相关知识，请参考《日志同步详细设计文档》。

*** 日志样例**

由于 commit log 不用于用户查看和定位系统问题，因此日志格式和样例暂不做举例和说明。

如果您需要了解 commit log 的相关知识，请参考《日志同步详细设计文档》。

1.4 rs_admin 日志

记录在初始化 OceanBase 过程中运行 **rs_admin** 的日志。

*** 日志说明**

OceanBase 的 rs_admin 日志的名称、路径和说明如表 1-5 所示。

表 1-5 日志说明

日志名称	日志路径	说明
rs_admin.log	执行 **rs_admin** 命令的 OceanBase 服务器的“~/oceanbase”目录下。	记录 bin/rs_admin 的执行日志。

*** 日志格式**

日志记录主要由五部分组成：记录时间、日志级别、文件名:行号、线程 ID 和日志内容。

*** 日志样例**

```
[2013-08-09 10:02:18.338467] INFO ob_base_client.cpp:70 [140524464404096] start io thread
```

解释：启动 io 线程。

2 系统结果码

OceanBase 结果码可以分为用户结果码和系统结果码。用户结果码主要用于定位和解决 OceanBase Java 客户端用户或者 MySQL 命令行客户端返回的错误；系统结果码主要用于定位和解决 OceanBase 的系统内部错误。

注意：*在使用 OceanBase 时，如果 OceanBase 返回系统结果码或者您无法解决用户结果码的问题时，请联系“支付宝·OceanBase 团队”。*

OceanBase 用户结果码如**表 2-1** 所示。

表 2-1 用户结果码

结果码	含义	参数
-12	请求超时。	const int OB_RESPONSE_TIME_OUT
-13	分配内存空间失败。	const int OB_ALLOCATE_MEMORY_FAILED
-14	UpdateServer 内存溢出。	const int OB_MEM_OVERFLOW
-38	读写集群不是主集群。	const int OB_NOT_MASTER
-42	OceanBase 用户不存在。（OceanBase 0.3 使用）	const int OB_USER_NOT_EXIST
-43	连接 OceanBase 的密码错误。（OceanBase 0.3 使用）	const int OB_PASSWORD_WRONG
-46	没有操作权限。（OceanBase 0.3 使用）	const int OB_NO_PERMISSION
-49	UpdateServer 处理超时。	const int OB_PROCESS_TIMEOUT
-56	建立 TCP 连接失败。	const int OB_CONN_ERROR

结果码	含义	参数
-64	MergeServer 将 RPC 请求拆分成多个 SESSION 从 ChunkServer 上读取数据时，SESSION 被杀死。（OceanBase 0.3 使用）	const int OB_SESSION_KILLED
-119	冻结时事务被回滚。	const int OB_TRANS_ROLLBACKED
-4007	主集群在每日合并过程中不允许进行 DDL 操作。 备集群不允许进行 DD 操作。	const int OB_OP_NOT_ALLOW
-5001	SQL 语法错误。	const int OB_ERR_PARSE_SQL
-5009	列不存在。	const int OB_ERR_COLUMN_UNKNOWN
-5019	表格不存在。	const int OB_ERR_TABLE_UNKNOWN
-5035	OceanBase 用户不存在。	const int OB_ERR_USER_NOT_EXIST
-5036	没有操作权限。	const int OB_ERR_NO_PRIVILEGE
-5038	连接 OceanBase 的密码错误。	const int OB_ERR_WRONG_PASSWORD
-5040	更新 RowKey 列错误。	const int OB_ERR_UPDATE_ROWKEY_COLUMN
-5041	更新 JOIN 列错误。	const int OB_ERR_UPDATE_JOIN_COLUMN
-5048	执行 REPLACE 操作的行锁冲突。	const int OB_ERR_EXCLUSIVE_LOCK_CONFLICT

结果码	含义	参数
-5049	执行 INSERT/UPDATE/DELETE 操作的行锁冲突。	const int OB_ERR_SHARED_LOCK_CONFLICT
-5060	事务已开始，开启新事务失败。	const int OB_ERR_TRANS_ALREADY_STARTED
-5062	不在事务中。	const int OB_ERR_NOT_IN_TRANS

OceanBase 系统结果码说明如表 2-2 所示。

表 2-2 系统结果码

结果码	含义	参数
-1	Object 类型错误。	const int OB_OBJ_TYPE_ERROR
-2	传入的参数错误。	const int OB_INVALID_ARGUMENT
-3	数组越界。	const int OB_ARRAY_OUT_OF_RANGE
-4	服务器监听错误。	const int OB_SERVER_LISTEN_ERROR
-5	已经初始化。	const int OB_INIT_TWICE
-6	没有初始化。	const int OB_NOT_INIT
-7	不支持的功能。	const int OB_NOT_SUPPORTED
-8	迭代到最后一行。	const int OB_ITER_END
-9	IO 错误。	const int OB_IO_ERROR
-10	版本错误。	const int OB_ERROR_FUNC_VERSION
-11	包未发送。	const int OB_PACKET_NOT_SENT

结果码	含义	参数
-15	系统错误。	const int OB_ERR_SYS
-16	未知错误。	const int OB_ERR_UNEXPECTED
-17	项已存在。	const int OB_ENTRY_EXIST
-18	项不存在。	const int OB_ENTRY_NOT_EXIST
-19	大小越界。	const int OB_SIZE_OVERFLOW
-20	引用计数不为零。	const int OB_REF_NUM_NOT_ZERO
-21	值发生冲突。	const int OB_CONFLICT_VALUE
-22	项未设置。	const int OB_ITEM_NOT_SETTED
-23	需要重试。	const int OB_EAGAIN
-24	缓冲不足。	const int OB_BUF_NOT_ENOUGH
-25	同步 Slave 过程失败。	const int OB_PARTIAL_FAILED
-26	读取内容为空。	const int OB_READ_NOTHING
-27	文件不存在。	const int OB_FILE_NOT_EXIST
-28	日志不连续。	const int OB_DISCONTINUOUS_LOG
-29	Schema 错误。	const int OB_SCHEMA_ERROR
-30	不提供该数据服务。	const int OB_DATA_NOT_SERVE

结果码	含义	参数
-31	未知的 Object。	const int OB_UNKNOWN_OBJ
-32	没有监控数据。	const int OB_NO_MONITOR_DATA
-33	序列化失败。	const int OB_SERIALIZE_ERROR
-34	反序列化失败。	const int OB_DESERIALIZE_ERROR
-35	AIO 超时。	const int OB_AIO_TIMEOUT
-36	需要重试。	const int OB_NEED_RETRY
-37	SSTable 过多。	const int OB_TOO_MANY_SSTABLE
-39	Token 失效。	const int OB_TOKEN_EXPIRED
-40	加密失败。	const int OB_ENCRYPT_FAILED
-41	解密失败。	const int OB_DECRYPT_FAILED
-44	秘钥错误。	const int OB_SKEY_VERSION_WRONG
-45	不是 Token 类型。	const int OB_NOT_A_TOKEN
-47	条件检查失败。	const int OB_COND_CHECK_FAIL
-48	未注册。	const int OB_NOT_REGISTERED
-50	不是这个 Object。	const int OB_NOT_THE_OBJECT
-51	重复注册。	const int OB_ALREADY_REGISTERED

结果码	含义	参数
-52	最后一个日志损坏。	const int OB_LAST_LOG_RUINNED
-53	没有 ChunkServer 被选择。	const int OB_NO_CS_SELECTED
-54	没有 Tablets 被创建。	const int OB_NO_TABLETS_CREATED
-55	无效。	const int OB_INVALID_ERROR
-57	十进制数溢出。	const int OB_DECIMAL_OVERFLOW_WARN
-58	十进制数不合法。	const int OB_DECIMAL_ILLEGAL_ERROR
-59	除零非法。	const int OB_OBJ_DIVIDE_BY_ZERO
-60	除法错误。	const int OB_OBJ_DIVIDE_ERROR
-61	不是十进制数。	const int OB_NOT_A_DECIMAL
-62	十进制数精度不一致。	const int OB_DECIMAL_PRECISION_NOT_EQUAL
-63	获取空取值范围。	const int OB_EMPTY_RANGE
-65	日志不同步。	const int OB_LOG_NOT_SYNC
-66	目录不存在。	const int OB_DIR_NOT_EXIST
-67	结束流式接口调用。	const int OB_NET_SESSION_END
-68	日志无效。	const int OB_INVALID_LOG
-69	日志填充。	const int OB_FOR_PADDING

结果码	含义	参数
-70	数据无效。	const int OB_INVALID_DATA
-71	已经完成。	const int OB_ALREADY_DONE
-72	操作被取消。	const int OB_CANCELED
-73	数据源变更。	const int OB_LOG_SRC_CHANGED
-74	日志不对齐。	const int OB_LOG_NOT_ALIGN
-75	日志记录丢失。	const int OB_LOG_MISSING
-76	需要等待。	const int OB_NEED_WAIT
-77	不支持的操作。	const int OB_NOT_IMPLEMENT
-78	被零除。	const int OB_DIVISION_BY_ZERO
-79	值超出范围。	const int OB_VALUE_OUT_OF_RANGE
-80	超出内存限制。	const int OB_EXCEED_MEM_LIMIT
-81	未知的结果。	const int OB_RESULT_UNKNOWN
-82	数据格式错误。	const int OB_ERR_DATA_FORMAT
-83	可能循环。	const int OB_MAYBE_LOOP
-84	没有结果。	const int OB_NO_RESULT
-85	队列溢出。	const int OB_QUEUE_OVERFLOW
-101	死锁。	const int OB_DEAD_LOCK

结果码	含义	参数
-102	日志不完整。	const int OB_PARTIAL_LOG
-103	效验和错误。	const int OB_CHECKSUM_ERROR
-104	初始化失败。	const int OB_INIT_FAIL
-110	读出全零日志。	const int OB_READ_ZERO_LOG
-111	切换日志错误。	const int OB_SWITCH_LOG_NOT_MATCH
-112	写日志未开始。	const int OB_LOG_NOT_START
-113	致命错误状态。	const int OB_IN_FATAL_STATE
-114	停止状态。	const int OB_IN_STOP_STATE
-115	主 UpdateServer 已存在。	const int OB_UPS_MASTER_EXISTS
-116	日志未清除。	const int OB_LOG_NOT_CLEAR
-117	文件已经存在。	const int OB_FILE_ALREADY_EXIST
-118	未知的包。	const int OB_UNKNOWN_PACKET
-120	日志过大。	const int OB_LOG_TOO_LARGE
-121	同步发包失败。	const int OB_RPC_SEND_ERROR
-122	异步发包失败。	const int OB_RPC_POST_ERROR

结果码	含义	参数
-123	Libeasy 错误。	const int OB_LIBEASY_ERROR
-124	连接错误。	const int OB_CONNECT_ERROR
-125	不处于释放状态。	const int OB_NOT_FREE
-126	初始化 SQL 语境失败。	const int OB_INIT_SQL_CONTEXT_ERROR
-127	跳过无效的 Row。	const int OB_SKIP_INVALID_ROW
-131	系统配置表错误。	const int OB_SYS_CONFIG_TABLE_ERROR
-132	读取配置失败。	const int OB_READ_CONFIG_ERROR
-140	事务不匹配。	const int OB_TRANS_NOT_MATCH
-141	只读事务。	const int OB_TRANS_IS_READONLY
-142	Row 被修改。	const int OB_ROW_MODIFIED
-143	版本不匹配。	const int OB_VERSION_NOT_MATCH
-144	无效的地址。	const int OB_BAD_ADDRESS
-145	重复的 COLUMN。	const int OB_DUPLICATE_COLUMN
-146	进队列失败。	const int OB_ENQUEUE_FAILED
-147	配置无效。	const int OB_INVALID_CONFIG

结果码	含义	参数
-1001	ChunkServer cache 未命中。	const int OB_CS_CACHE_NOT_HIT
-1002	ChunkServer 超时。	const int OB_CS_TIMEOUT
-1008	Tablet 不存在。	const int OB_CS_TABLET_NOT_EXIST
-1010	需要重试。	const int OB_CS_EAGAIN
-1011	扫描下一列。	const int OB_GET_NEXT_COLUMN
-1012	扫描下一行。	const int OB_GET_NEXT_ROW
-1013	反序列化失败。	const int OB_DESERIALIZE_ERROR
-1014	RowKey 无效。	const int OB_INVALID_ROW_KEY
-1015	不支持的查询模式。	const int OB_SEARCH_MODE_NOT_IMPLEMENT
-1016	错误的 Block 索引。	const int OB_INVALID_BLOCK_INDEX
-1017	错误的 Block 数据。	const int OB_INVALID_BLOCK_DATA
-1018	未找到。	const int OB_SEARCH_NOT_FOUND
-1020	查询的 Key 或者取值范围不在当前的 Tablet 中。	const int OB_BEYOND_THE_RANGE
-1021	完成移动块缓存。	const int OB_CS_COMPLETE_TRAVERSAL
-1022	Row 的结尾。	const int OB_END_OF_ROW

结果码	含义	参数
-1024	ChunkServer 合并失败。	const int OB_CS_MERGE_ERROR
-1025	ChunkServer 的 Schema 不兼容。	const int OB_CS_SCHEMA_INCOMPATIBLE
-1026	ChunkServer 服务未启动。	const int OB_CS_SERVICE_NOT_STARTED
-1027	Lease 过期。	const int OB_CS_LEASE_EXPIRED
-1028	合并超时。	const int OB_CS_MERGE_HAS_TIMEOUT
-1029	表已经被删除。	const int OB_CS_TABLE_HAS_DELETED
-1030	ChunkServer 合并取消。	const int OB_CS_MERGE_CANCELED
-1031	压缩依赖库出错。	const int OB_CS_COMPRESS_LIB_ERROR
-1032	超出磁盘空间。	const int OB_CS_OUTOF_DISK_SPACE
-1033	ChunkServer 移植已经存在的 Tablet。	const int OB_CS_MIGRATE_IN_EXIST
-1037	错误的 SSTable 数据。	const int OB_WRONG_SSTABLE_DATA
-1039	COLUMN GROUP 未找到。	const int OB_COLUMN_GROUP_NOT_FOUND
-1040	没有导入的 SSTable。	const int OB_NO_IMPORT_SSTABLE
-1041	导入 SSTable 不存在。	const int OB_IMPORT_SSTABLE_NOT_EXIST

结果码	含义	参数
-2001	UpdateServer 事务正在进行中。	const int OB_UPS_TRANS_RUNNING
-2002	MemTable 已经被冻结。	const int OB_FREEZE_MEMTABLE_TWICE
-2003	MemTable 已经被删除。	const int OB_DROP_MEMTABLE_TWICE
-2004	MemTable 启动的版本无效。	const int OB_INVALID_START_VERSION
-2005	UpdateServer 不存在。	const int OB_UPS_NOT_EXIST
-2006	UpdateServer 获取 memtable 或者 sstable 失败。	const int OB_UPS_ACQUIRE_TABLE_FAIL
-2007	主版本无效。	const int OB_UPS_INVALID_MAJOR_VERSION
-2008	UpdateServer 表未冻结。	const int OB_UPS_TABLE_NOT_FROZEN
-2009	UpdateServer 切主超时。	const int OB_UPS_CHANGE_MASTER_TIMEOUT
-2010	强制终止时间。	const int OB_FORCE_TIME_OUT
-3001	时间戳错误。	const int OB_ERROR_TIME_STAMP
-3002	交叉错误。	const int OB_ERROR_INTRESECT
-3003	超出取值范围。	const int OB_ERROR_OUT_OF_RANGE
-3004	RootServer 状态初始化。	const int OB_RS_STATUS_INIT
-3005	不提供旁路服务。	const int OB_IMPORT_NOT_IN_SERVER

结果码	含义	参数
-3006	发现取值范围。	const int OB_FIND_OUT_OF_RANGE
-3007	转换错误。	const int OB_CONVERT_ERROR
-3008	遍历 MergeServer 列表介绍。	const int OB_MS_ITER_END
-4001	内部状态错误。	const int OB_INNER_STAT_ERROR
-4002	Schema 版本太旧。	const int OB_OLD_SCHEMA_VERSION
-4003	输入参数错误。	const int OB_INPUT_PARAM_ERROR
-4004	找不到空项。	const int OB_NO_EMPTY_ENTRY
-4005	释放 Schema 失败。	const int OB_RELEASE_SCHEMA_ERROR
-4006	项目数量统计错误。	const int OB_ITEM_COUNT_ERROR
-4008	ChunkServer 缓存失败。	const int OB_CHUNK_SERVER_ERROR
-4009	没有新 Schema。	const int OB_NO_NEW_SCHEMA
-4010	子扫描请求过多。	const int OB_MS_SUB_REQ_TOO_MANY
-5000	启动 SQL 失败。	const int OB_ERR_SQL_START
-5000	语法解析初始化失败。	const int OB_ERR_PARSER_INIT

结果码	含义	参数
-5002	解析 SQL 失败。	const int OB_ERR_RESOLVE_SQL
-5003	产生物理执行计划错误。	const int OB_ERR_GEN_PLAN
-5004	未知的系统功能错误。	const int OB_ERR_UNKNOWN_SYS_FUNC
-5005	解析 SQL 语法时分配内存错误。	const int OB_ERR_PARSER_MALLOC_FAILED
-5006	解析 SQL 语法错误。	const int OB_ERR_PARSER_SYNTAX
-5007	COLUMN 大小错误。	const int OB_ERR_COLUMN_SIZE
-5008	重复列。	const int OB_ERR_COLUMN_DUPLICATE
-5010	未知的操作错误。	const int OB_ERR_OPERATOR_UNKNOWN
-5011	“*”使用错误。	const int OB_ERR_STAR_DUPLICATE
-5012	ID 不合法。	const int OB_ERR_ILLEGAL_ID
-5013	位置错误。	const int OB_ERR_WRONG_POS
-5014	值不合法。	const int OB_ERR_ILLEGAL_VALUE
-5015	列描述错误。	const int OB_ERR_COLUMN_AMBIGOUS
-5016	逻辑计划失败。	const int OB_ERR_LOGICAL_PLAN_FAILD
-5017	Schema 未设置。	const int OB_ERR_SCHEMA_UNSET

结果码	含义	参数
-5018	名称不合法。	const int OB_ERR_ILLEGAL_NAME
-5020	表格重复。	const int OB_ERR_TABLE_DUPLICATE
-5021	字符串被截断。	const int OB_ERR_NAME_TRUNCATE
-5022	表达式错误。	const int OB_ERR_EXPR_UNKNOWN
-5023	类型不合法。	const int OB_ERR_ILLEGAL_TYPE
-5024	主键已经存在。	const int OB_ERR_PRIMARY_KEY_DUPLICATE
-5025	已经存在。	const int OB_ERR_ALREADY_EXISTS
-5026	create_time 列已经存在。	const int OB_ERR_CREATETIME_DUPLICATE
-5027	modify_time 列已经存在。	const int OB_ERR_MODIFYTIME_DUPLICATE
-5028	索引非法。	const int OB_ERR_ILLEGAL_INDEX
-5029	Schema 无效。	const int OB_ERR_INVALID_SCHEMA
-5030	插入空 RowKey。	const int OB_ERR_INSERT_NULL_ROWKEY
-5031	COLUMN 未找到。	const int OB_ERR_COLUMN_NOT_FOUND
-5032	删除空 RowKey。	const int OB_ERR_DELETE_NULL_ROWKEY
-5033	插入 Join 列错误。	const int OB_ERR_INSERT_INNER_JOIN_COLUMN

结果码	含义	参数
-5034	用户为空。	const int OB_ERR_USER_EMPTY
-5037	没有进入权限。	const int OB_ERR_NO_AVAILABLE_PRIVILEGE_ENTRY
-5039	用户被锁。	const int OB_ERR_USER_IS_LOCKED
-5042	无效的列名。	const int OB_ERR_INVALID_COLUMN_NUM
-5043	未知的 PREPARE 语句。	const int OB_ERR_PREPARE_STMT_UNKNOWN
-5044	未知变量。	const int OB_ERR_VARIABLE_UNKNOWN
-5045	初始化 SESSION 失败。	const int OB_ERR_SESSION_INIT
-5046	旧权限版本。	const int OB_ERR_OLDER_PRIVILEGE_VERSION
-5047	缺少 RowKey 列。	const int OB_ERR_LACK_OF_ROWKEY_COL
-5050	用户已存在。	const int OB_ERR_USER_EXIST
-5051	密码为空。	const int OB_ERR_PASSWORD_EMPTY
-5052	授予 CREATE TABLE 权限错误。	const int OB_ERR_GRANT_PRIVILEGES_TO_CREATE_TABLE
-5053	错误的动态参数。	const int OB_ERR_WRONG_DYNAMIC_PARAM
-5054	参数大小错误。	const int OB_ERR_PARAM_SIZE

结果码	含义	参数
-5055	未知的功能错误。	const int OB_ERR_FUNCTION_UNKNOWN
-5056	建立 molidfy_time 列错误。	const int OB_ERR_CREAT_MODIFY_TIME_COLUMN
-5057	修改 Primary Key 错误。	const int OB_ERR_MODIFY_PRIMARY_KEY
-5058	重复的参数。	const int OB_ERR_PARAM_DUPLICATE
-5059	SESSION 过多。	const int OB_ERR_TOO_MANY_SESSIONS
-5061	PS 次数太多。	const int OB_ERR_TOO_MANY_PS
-5063	未知的 HINT 错误。	const int OB_ERR_HINT_UNKNOWN
-5999	SQL 结束错误。	const int OB_ERR_SQL_END

3 内部表参考

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。

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 3-1 所示。

表 3-1 __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(256)	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	• 1：普通表。 • 2：索引。 • 3：元数据表。 • 4：view。 • 5：临时表。
load_type	int	• 1：保存到磁盘。 • 2：保存到内存。
table_def_type	int	• 1：内部表。 • 2：用户定义表。

参数	类型	说明
rowkey_column_num	int	主键的列数，后续 endrowkeyobj1, endrowkeyobj2… 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 mem_version，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(256)	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。
rowkey_split	int	用于指定每日合并中 Tablet 的分裂点为 rowkey 的第几个 obj。
expire_condition	varchar(512)	使用表达式定义的此表的数据自动过期删除条件。
tablet_block_size	int	Tablet_block 的大小。
is_read_static	int	是否要读静态数据。
schema_version	int	schema 版本。

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 3-2 所示。

表 3-2 __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(128)	列名。
table_name	varchar(256)	表名。
column_id	int	列 ID。
column_group_id	int	列隶属的 column group id。
rowkey_id	int	• 0：非 rowkey。 • 正整数：rowkey 的序号，必须是从 1 开始的连续正整。 ***说明：*** *“__all_table_table”中的“rowkey_column_num”定义了该表的 rowkey 的列数量。*
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	• -1：没有 join。 • 正整数：连接表的表 ID。
join_column_id	int	• -1：没有 join。 • 正整数：连接表中的列 ID。

参数	类型	说明
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	• 1：不可以为空。 • 2：可以为空。

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 3-3 所示。

表 3-3 __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。
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(256)	左表的表名。
left_column_name	varchar(256)	左表的列名。

参数	类型	说明
right_table_name	varchar(256)	右表的表名。
right_column_name	varchar(256)	右表的列名。

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 3-4 所示。

表 3-4 __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(32)	• 0：与特定集群无关。 • 正整数：指定集群。
version	varchar(16)	客户端版本。
status	varchar(32)	客户端状态。
extra1	varchar(32)	预留。
extra2	int	预留。

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 3-5 所示。

表 3-5 __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(32)	Cluster 的 IP 地址。
cluster_port	int	Cluster 端口号。
cluster_role	int	• 1：Master • 2：Slave
cluster_name	varchar(128)	集群名称。
cluster_info	varchar(128)	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： • 0：随机轮转策略。 • 1：一致性哈希。
rootserver_port	int	RootServer 服务端口。

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 3-6 所示。

表 3-6 __all_server 参数

参数	类型	说明
gm_create	createtime	创建时间。

参数	类型	说明
gm_modify	modifytime	修改时间。
cluster_id	int	• 0：与特定集群无关。 • 正整数：指定的集群 ID。
svr_type	varchar(16)	• RootServer • ChunkServer • MergeServer • UpdateServer • Other
svr_ip	varchar(32)	Server IP 地址。
svr_port	int	Server 端口。
inner_port	int	内部交互端口。
svr_role	int	• 0：与服务器角色(RS 或 UPS 的主或备)无关。 • 1：slave • 2：master
svr_version	varchar(64)	程序版本信息。

3.7 __all_server_session

“__all_server_session”用于记录当前连接 OceanBase 的信息。

Rowkey：(id)

“__all_server_session”参数说明如表 3-7 所示。

表 3-7 __all_server_stession 参数

参数	类型	说明
id	int	连接 OceanBase 的 ID。
username	varchar(512)	连接 OceanBase 的用户名。

参数	类型	说明
host	varchar(128)	连接到 OceanBase 的客户端的 IP 和 Port。
db	varchar(128)	数据库名，目前只有 oceanbase。
command	varchar(1024)	执行的命令。
timeelapse	int	正在执行的 SQL 的耗时。单位：微秒。
state	varchar(128)	是否正在使用： • ACTIVE：正在使用。 • SLEEP：暂时未使用。
info	varchar(128)	连接说明信息。
mergeserver	varchar(128)	连接的 MergeServer。
index	int	连接 OceanBase 的 thread id。

3.8 __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 3-8 所示。

表 3-8 __all_server_stat 参数

参数	类型	说明
svr_type	varchar(16)	• RootServer • ChunkServer • MergeServer • UpdateServer • Other
svr_ip	varchar(32)	Server IP 地址。
svr_port	int	Server 端口。

参数	类型	说明
name	varchar(64)	监控项的名称。
value	int	监控项的值。

3.9 __all_statement

“__all_statement”用于记录全局缓存的 PREPARE 语句。

Rowkey：(svr_ip, svr_port, statement)

“__all_statement”参数说明如表 3-9 所示。

表 3-9 __all_statement 参数

参数	类型	说明
svr_ip	varchar(32)	创建语句的 IP。
svr_port	int	创建语句的端口号。
statement	varchar(1024)	语句文本。
id	int	内部标识的唯一 id。
prepare_count	int	执行 PREPARE 的次数。
execute_count	int	执行 EXECUTE 的次数。
avg_execute_usec	int	平均执行时间，单位：微秒。
slow_count	int	慢查询的次数。
create_time	timestamp	创建的时间。
last_active_time	timestamp	最后一次使用的时间。

3.10 _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 3-10 所示。

表 3-10 __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	• 0：与特定集群无关。 • 正整数：指定集群。
svr_type	varchar(16)	• RootServer • ChunkServer • MergeServer • UpdateServer • Other
svr_ip	varchar(32)	Server IP 地址。
svr_port	int	Server 端口。
name	varchar(256)	参数名称。
section	varchar(256)	参数所属的段。
data_type	varchar(256)	参数值的数据类型。
value	varchar(256)	参数值。
value_strict	varchar(256)	参数值的约束。
info	varchar(256)	对该项的说明。

3.11 __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 3-11 所示。

表 3-11 __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	• 0：与特定集群无关。 • 正整数：指定集群。
svr_type	varchar(16)	• RootServer • ChunkServer • MergeServer • UpdateServer • Other
svr_ip	varchar(32)	Server IP 地址。
svr_port	int	Server 端口。
name	varchar(256)	参数名称。
section	varchar(256)	参数所属的段。
data_type	varchar(256)	参数值的数据类型。
value	varchar(256)	参数值。
value_strict	varchar(256)	参数值的约束。
info	varchar(256)	对该项的说明。

3.12 __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 3-12 所示。

表 3-12 __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	• 0：与特定集群无关。 • 正整数：指定的集群 ID。
name	varchar(256)	参数名称。
data_type	int	参数值的数据类型。 • 0：null • 1：int • 2：float • 3：double • 4：datetime • 5：precisedate，与 datetime 相同，精确到毫秒。 • 6：vchar • 8：createtime • 9：modifytime • 11：bool • 12：decimal
value	varchar(256)	参数值。
info	varchar(256)	对该项的说明。

在“__all_sys_param”表中已定义的参数说明如表 3-13 所示。

表 3-13 已定义的参数

参数		说明
name	**data_type**	
auto_increment_increment	1	自增步长，仅用于 MySQL 客户端登录。
autocommit	1	是否自动提交。

参数		说明
character_set_results	6	字符集。
interactive_timeout	1	交互式的连接中，链接断开的超时时间。单位：秒。
max_allowed_packe	1	最大网络包大小。
ob_app_name	1	应用的名称。
ob_charset	6	返回给客户端的字符集。
ob_disable_create_sys_table	11	是否允许创建修改系统表。
ob_group_agg_push_down_param	11	聚合操作是否下移到 Chunkserver 的开关。
ob_query_timeout	1	每次 query 的超时时间。
ob_read_consistency	1	读一致性级别。
ob_tx_idle_timeout	1	事务开始后无任何操作时，事务的超时时间。
ob_tx_timeout	1	事务超时时间。
sql_mode	6	SQL 模式。
tx_isolation	6	事务隔离性。
version_comment	6	observer 的版本信息。
wait_timeout	1	非交互模式下，超时等待的时间。

3.13 __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 3-14 所示。

表 3-14 __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。
cluster_id	int	• 0：与特定集群无关。 • 正整数：指定集群。
name	varchar(256)	参数名称。
data_type	int	数据类型。 • 0：null • 1：int • 2：float • 3：double • 4：datetime • 5：precisedate，与 datetime 相同，精确到毫秒。 • 6：vchar • 8：createtime • 9：modifytime • 11：bool • 12：decimal
value	varchar(256)	参数值。
info	varchar(256)	对参数的说明。

在“__all_sys_stat”表中已经定义的参数说明如表 3-15 所示。

表 3-15 已定义的参数

参数			说明
name	**cluster_id**	**data_type**	
ob_cerrent_privilege_version	0	1	系统当前的权限版本。
max_used_table_id	0	1	已经使用的最大 table_id。

参数			说明
max_used_user_id	0	1	已经使用的最大 user_id。

3.14 __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 3-16 所示

表 3-16 __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。
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。

参数	类型	说明
priv_replace	int	是否有 replace 权限。

3.15 __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 3-17 所示。

表 3-17 __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。
event_type	int	事件类型。
event_param	int	消息参数。
extra	varchar	预留。

3.16 __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 3-18 所示。

表 3-18 __all_user 参数

参数	类型	说明
gm_create	createtime	创建时间。

参数	类型	说明
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。
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。

4 配置项参考

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 0.4.2 SQL 参考指南》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《OceanBase 0.4.2 参考指南》的“7 配置 OceanBase”。

4.1 RootServer 配置参数

RootServer 配置参数说明如表 4-1 所示。

表 4-1 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。 取值范围：[1,10]
balance_max_migrate_in_per_cs	20	一批迁移 SSTable 组成一个任务，该值表示单个 ChunkServer 上允许迁入的最大的迁移任务数。 取值范围：[1,100]
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。 取值范围：[1,100]
balance_max_timeout	5m	迁移任务的超时时间。 不建议修改。

参数	缺省值	说明
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。 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： [SSTable 数/ChunkServer 总数 - balance_tolerance_count, SSTable 数/ChunkServer 总数 + balance_tolerance_count] 取值范围：[1,1000] 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。 不建议修改。
build_root_table_time	5m	当 ChunkServer 和 RootServer 中的数据不一致时，需要 Clean RootTable 进行重新构建。该参数表示重新构建 RootTable 的时间。
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： • 0：用于 CommitLog 的内存缓冲区写满后再刷入 CommitLog 文件内。 • 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。

参数	缺省值	说明
cs_lease_duration_time	10s	ChunkServer 的租约有效期。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。 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。
ddl_system_table_switch	False	是否允许操作内部表的开关。 • True：开启。 • False：不开启。
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。 • True：开启。 • False：不开启。
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。 • True：是。 • False：否。
enable_load_data	False	是否开启允许外部数据进行旁路导入。 • True：开启。 • False：不开启。

参数	缺省值	说明
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。 • True：开启。 • False：不开启。
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。
first_meta_filename	first_tablet_meta	用来标示是否已经执行“boot_strap”命令。
inner_table_network_timeout	50s	访问内部表的超时时间。
io_thread_count	4	用于 Libeay 的 I/O 线程数。 需要重新启动 RootServer 服务才能生效。 取值范围：[1,100]
is_ups_flow_control	False	主备 UpdateServer 是否按照流量分配进行读服务。 • True：主备 UpdateServer 的读服务按流量分配。 • False：主备 UpdateServer 的读服务按主备各 50%分配。
lease_interval_time	15s	主 RootServer 给备 RootServer 发送一个租约的有效期。 不建议修改。

参数	缺省值	说明
lease_on	True	主备 RootServer 间是否需要开启租约模式。 • True：开启。 • False：不开启。
lease_reserved_time	10s	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。 不建议修改。
load_data_max_timeout_per_range	3m	RootServer 发起 Tablet 迁移时，Tablet 按 range 从 ChunkServer1 迁移到 ChunkServer2。该参数表示 ChunkServer2 向 RootServer 汇报的超时时间。
load_sstable_time	20m	ChunkServer 加载 SSTable 的超时时间。
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 。 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。 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。

参数	缺省值	说明
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，否侧日志同步失败。 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。 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。 不建议修改。
monitor_drop_table_interval	600s	用户在 DROP TABLE 时，OceanBase 内部不会立即删除该表，而是在每日合并时删除。该参数表示删除表的时间间隔。
monitor_row_checksum	True	备集群开启行一致性校验的开关。
monitor_row_checksum_interval	1800s	备集群进行行一致性校验的周期。
monitor_row_checksum_timeout	3s	备集群向主集群取校验码的超时时间。
network_timeout	50s	RootServer 与其他 Server 进行网络互交的超时时间。

参数	缺省值	说明
port	2500	RootServer 的服务端口。 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”。 取值范围：[0,100] 推荐值：40
read_queue_size	500	RootServer 处理读请求任务的队列大小。 取值范围：[10,100000] 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”。 取值范围：[0,100] 推荐值：50
read_thread_count	20	RootServer 处理读任务的线程数。 取值范围：[10,100]
report_tablet_time	5m	Clean RootTable 时，ChunkServer 向 RootServer 汇报的超时时间。
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。 不建议修改。
safe_wait_init_time	60s	Clean RootTable 时，有一个线程检查新的 RootTable 是否构建完成的时间间隔。
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。 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。 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。 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。 不建议修改。

参数	缺省值	说明
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。
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。 不建议修改。
write_queue_size	100	RootServer 处理写请求任务的队列大小。 取值范围：[10,100000]

4.2 UpdateServer 配置参数

UpdateServer 配置参数说明如表 4-2 所示。

表 4-2 UpdateServer 配置参数

参数	缺省值	说明
active_mem_limit	系统自动生成	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。 推荐值和计算方法如下： • app_mod=import 时计算方法：table_memory_limit/minor_num_limit*0.7 • app_mod=oltp 时计算方法：“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。

参数	缺省值	说明
blockcache_size	系统自动生成	Block 缓存大小。该参数不可以动态改小，但是可以动态改大。 需要重新启动 UpdateServer 服务才能生效。 推荐值：系统自动生成
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。 需要重新启动 UpdateServer 服务才能生效。 推荐值：系统自动生成
commit_bind_core_id	-1	并发事务提交线程所绑定的 CPU id。
commit_end_thread_num	4	CommitEndHandlePool 线程池中线程数量，用于给客户端回包以及销毁一次请求所用的数据结构。
commit_log_dir	data/ups_commitlog	OceanBase 安装目录下 UpdateServer 的 CommitLog 目录。 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。 需要重新启动 UpdateServer 服务才能生效。 不建议修改。

参数	缺省值	说明
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： • 1：Strong，只能读取主 UpdateServer 的 CommitLog。 • 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。 • 3：Weak，可以读取主或备 UpdateServer 的 CommitLog。 推荐值：3
devname	eth0	启动 UpdateServer 服务的网卡名称。 需要重启 UpdateServer 服务才能生效。
dir_regex	^store[0-9]+$	raid 目录下指向磁盘实际目录的软链接的名称匹配式。 需要重新启动 UpdateServer 服务才能生效。 不建议修改。
disk_delay_warn_param	40ms;8000us;10;100000	写 CommitLog 的超过该时间。如果超时，则产生一条 Warn 日志。
fetch_log_wait_time	500ms	备 UpdateServer 从主 UpdateServer 两次读取 CommitLog 的时间间隔。 不建议修改。

参数	缺省值	说明
fetch_schema_timeout	3s	UpdateServer 从 RootServer 获取 schema 的超时时间。 需要重新启动 UpdateServer 服务才能生效。 不建议修改。
fetch_schema_times	10	UpdateServer 从 RootServer 获取 schema 失败时的重试次数。 不建议修改。
high_prio_quota_percent	50	对于工作线程中高优先级的 CPU 计算资源配给比例。
inner_port	2701	用于每日合并的端口号。 需要重新启动 UpdateServer 服务才能生效。 不建议修改。 取值范围：(1024,65536)
io_thread_count	3	用于 Libeay 的 I/O 线程数。 需要重新启动 UpdateServer 服务才能生效。
io_thread_end_cpu	-1	与 io_thread_start_cpu 一起，表示读写线程绑定的 CPU 范围。例如 io_thread_start_cpu = 1，io_thread_end_cpu = 10，表示多核 CPU 中，从核 1 到核 10 用于处理读写请求。
io_thread_start_cpu	-1	与 io_thread_end_cpu 一起，表示读写线程绑定的 CPU 的范围。例如 io_thread_start_cpu = 1，io_thread_end_cpu = 10，表示多核 CPU 中，从核 1 到核 10 用于处理读写请求。

参数	缺省值	说明
keep_alive_interval	500ms	控制 UpdateServer 向 RootServer 发送心跳信息的最小间隔时间。
keep_alive_reregister_timeout	800ms	备 UpdateServer 超过此时间都未收到主 UpdateServer 日志的情况下，则需要重新向主 UpdateServer 注册。
keep_alive_timeout	5s	备 UpdateServer 未收到主 UpdateServer 的消息超过该值时，则重新向主 UpdateServer 注册。 需要重新启动 UpdateServer 服务才能生效。 不建议修改。
lease_queue_size	100	UpdateServer 从 RootServer 获取租约的任务队列长度。 需要重新启动 UpdateServer 服务才能生效。
lease_timeout_in_advance	500ms	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。 需要重新启动 UpdateServer 服务才能生效。 不建议修改。
log_cache_block_size	32MB	用于备 UpdateServer 接收主 UpdateServer 的 CommitLog 的缓存块大小。 需要重新启动 UpdateServer 服务才能生效。

参数	缺省值	说明
log_cache_n_block	4	用于备 UpdateServer 接收主 UpdateServer 的 CommitLog 的缓存块的数量。 需要重新启动 UpdateServer 服务才能生效。
log_queue_size	100	CommitLog 同步任务队列长度。 需要重新启动 UpdateServer 服务才能生效。
log_sync_delay_tolerable_time_threshold	5s	在同步 CommitLog 过程中，如果延迟超过该值，则会设置 UpdateServer 状态为“NOT_SYNC”。
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。 不建议修改。
log_sync_timeout	500ms	主 UpdateServer 向备 UpdateServer 发送 CommitLog 的超时时间。

参数	缺省值	说明
log_sync_type	1	主机发备机 CommitLog 时，写入磁盘的方式： • 0：用于 CommitLog 的内存缓冲区写满后再刷入 CommitLog 文件内。 • 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。 需要重新启动 UpdateServer 服务才能生效。
low_prio_quota_percent	10	对于工作线程中低优先级的 CPU 计算资源配给比例。
low_priv_adjust_flag	True	是否调整请求处理优先级概率的标志。 • True：是。 • False：否。
low_priv_cur_percent	10	低优先级请求处理概率的初始值。
low_priv_network_lower_limit	30MB	低优先级网络请求的最小带宽, 单位 M/s, 若 low_priv_adjust_flag 设置为 1，并且最近一段时间低优先级的网络请求使用网络带宽比下面的值小, 则提高低优先级请求的处理概率, 每次调整概率加 1%。
low_priv_network_upper_limit	30MB	低优先级网络请求的最大带宽, 单位 M/s, 若 low_priv_adjust_flag 设置为 1，并且最近一段时间低优先级的网络请求使用网络带宽比下面的值大, 则降低低优先级请求的处理概率, 每次调整概率减 1%。

参数	缺省值	说明
lsync_fetch_timeout	5s	备 UpdateServer 从 LsyncServer 或主 UpdateServer 读取 CommitLog 的超时时间。 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。 需要重新启动 UpdateServer 服务才能生效。
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。 取值范围：(1024,65536)
major_freeze_duty_time	Disable, 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	3968678 76B	Hash 索引大小。 需要重新启动 UpdateServer 服务才能生效。 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。

参数	缺省值	说明
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。 推荐值如下： • app_mod=import 时：3 • app_mod=oltp 时：1
net_delay_warn_param	50ms;5ms;5;10000	备 UpdateServer 向主 UpdateServer 同步 CommitLog 的网络超时超过该值时，产生告警。
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。
packet_max_wait_time	10s	通用网络请求超时。 需要重新启动 UpdateServer 服务才能生效。 不建议修改。
port	2700	UpdateServer 的服务端口。 需要重启 UpdateServer 服务才能生效。
raid_regex	^raid[0-9]+$	raid 目录名的匹配式。 需要重新启动 UpdateServer 服务才能生效。 不建议修改。
read_queue_size	1000	处理管理命令线程任务队列大小。 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。 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。

参数	缺省值	说明
register_timeout	3s	UpdateServer 向 RootServer 注册的超时时间。 需要重新启动 UpdateServer 服务才能生效。 不建议修改。
register_times	10	UpdateServer 向 RootServer 注册失败时的重试次数。 需要重新启动 UpdateServer 服务才能生效。 不建议修改。
replay_checksum_flag	True	日志回放时，是否对 MemTable 进行校验和检查。 • True：是。 • False：否。
replay_log_buf_size	10GB	用于 CommitLog 重放的缓冲区大小。 不建议修改。
replay_queue_len	500	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。 不建议修改。
replay_wait_time	100ms	两次 CommLog 重放的时间间隔。 不建议修改。
replay_worker_num	20	CommLog 重放线程数。
resp_root_timeout	1s	UpdateServer 向 RootServer 汇报冻结数据版本的超时时间。 需要重新启动 UpdateServer 服务才能生效。 不建议修改。

参数	缺省值	说明
resp_root_times	20	UpdateServer 向 RootServer 汇报冻结数据版本失败时的重试次数。 需要重新启动 UpdateServer 服务才能生效。 不建议修改。
retry_times	3	UpdateServer 与其他 Server 进行网络互交失败时的重试次数。
root_server_ip	-	UpdateServer 所在集群的主 RootServer 的 IP。
root_server_port	2500	UpdateServer 所在集群的主 RootServer 的服务端口。 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“*$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。 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。

参数	缺省值	说明
store_root	data/ups_data	OceanBase 安装目录下 UpdateServer 的数据目录。 需要重新启动 UpdateServer 服务才能生效。 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。 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。 推荐值：系统自动生成
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。 推荐值：系统自动生成
table_memory_limit	系统自动生成	MemTable 可用内存。 计算方法：(tatol_memory_limit - total_reserve) / (1/20 + 1/15 + 1)，一般情况下 total_reserve =10G 推荐值：系统自动生成 ***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。*
total_memory_limit	系统自动生成	UpdateServer 可用内存。 推荐值：系统自动生成
trans_proc_time_warn	1s	UpdateServer 处理某一事务的时间超过该值时，打印一条告警日志。 不建议修改。

参数	缺省值	说明
trans_thread_end_cpu	-1	与 trans_thread_start_cpu 一起，表示 UpdateServer 处理事务的 CPU 范围。例如 trans_thread_start_cpu = 1，trans_thread_end_cpu = 10，表示多核 CPU 中，从核 1 到核 10 用于处理事务。
trans_thread_num	30	处理 SQL 读写请求的工作线程数。
trans_thread_start_cpu	-1	与 trans_thread_end_cpu 一起，表示 UpdateServer 处理事务的 CPU 范围。例如 trans_thread_start_cpu = 1，trans_thread_end_cpu = 10，表示多核 CPU 中，从核 1 到核 10 用于处理事务。
using_hash_index	True	是否使用 Hash 索引。 • True：是。 • False：否。
using_memtable_bloomfilter	False	是否使用 Bloomfilter。 • True：是。 • False：否
using_static_cm_column_id	False	在 UpdateServer 中，有一个 Cache，缓存了 table_id 和其 create_time 和 mtime 的映射。UpdateServer 可以根据 table_id 查询其 create_time 和 modify_time 的列 ID。该参数表示是否通过查询缓存获取 create_time 和 modify_time 的列 ID。 • True：直接使用 2 和 3。 • False：在缓存中查询获取。
wait_slave_sync_time	100ms	备 UPS 接收日志的等待时间，需要在此时间之内回包给主 UpdateServer。

参数	缺省值	说明
wait_slave_sync_type	0	备 UpdateServer 回放 CommitLog 时，应答主 UpdateServer 的时机： • 0：在 CommitLog 回放前，应答主 UpdateServer。 • 1：在 CommitLog 回放后，且写入磁盘前，应答主 UpdateServer。 • 2：在 CommitLog 写入磁盘后，应答主 UpdateServer。
warm_up_time	10m	SSTable 的预热缓存时间。 取值范围：[10s,1800s]
write_queue_size	1000	写线程所对应的任务队列的容量。
write_sstable_use_dio	True	是否使用 DIO(Direct IO)方式写 SSTable。 • True：直接写入磁盘。 • False：先写入缓存，再写入磁盘。

4.3 MergeServer 配置参数

MergeServer 配置参数说明如表 4-3 所示。

表 4-3 MergeServer 配置参数

参数	缺省值	说明
bloom_filter_cache_size	256MB	INSERT 语句用 bloomfilter 大小。
change_obi_timeout	30s	主备集群切换超时时间。
check_ups_log_interval	1	主备集群切换过程中检查 UpdateServer 日志是否同步的间隔。
devname	eth0	启动 MergeServer 服务的网卡名称。

参数	缺省值	说明
frozen_data_cache_size	256M	静态数据缓存大小。
io_thread_count	12	用于内部任务端口的 I/O 线程数。 取值范围：[1,∞)
io_thread_end_cpu	-1	与 io_thread_start_cpu 一起，表示 MergeServer 用于 IO 请求的 CPU 范围。例如 io_thread_start_cpu = 1，io_thread_end_cpu = 10，表示多核 CPU 中，从核 1 到核 10 用于 IO 请求。
io_thread_start_cpu	-1	与 io_thread_end_cpu 一起，表示 MergeServer 用于 IO 请求的 CPU 范围。例如 io_thread_start_cpu = 1，io_thread_end_cpu = 10，表示多核 CPU 中，从核 1 到核 10 用于 IO 请求。
lease_check_interval	6s	检查与 RootServer 租约是否失效的时间间隔。 不建议修改。
lms	False	是否为 Listener MergeServer。
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。
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。 不建议修改。

参数	缺省值	说明
max_get_rows_per_subreq	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。 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。 取值范围：[0,∞)
max_parellel_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。 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。 取值范围：(0,100] 推荐值：40
monitor_interval	600s	MergeServer 执行两次内部定时任务（如打印日志等）的的时间间隔。
network_timeout	2s	MergeServer 与其他 Server 进行网络互交的超时时间。 推荐值：3s
obmysql_io_thread_count	8	用于 MySQL 端口的 I/O 最大线程数。 取值范围：[1,∞)

参数	缺省值	说明
obmysql_io_thread_end_cpu	-1（表示不生效）	与 obmysql_io_thread_start_cpu 一起，表示 MergeServer 中 MySQL 协议端口的 IO 线程绑定的 CPU 范围。例如 obmysql_io_thread_start_cpu = 1，obmysql_io_thread_end_cpu = 10，表示多核 CPU 中，从核 1 到核 10 用于 MySQL 协议端口的 IO 请求。
obmysql_io_thread_start_cpu	-1	与 obmysql_io_thread_end_cpu 一起，表示 MergeServer 中 MySQL 协议端口的 IO 线程绑定的 CPU 范围。例如 obmysql_io_thread_start_cpu = 1，obmysql_io_thread_end_cpu = 10，表示多核 CPU 中，从核 1 到核 10 用于 MySQL 协议端口的 IO 请求。
obmysql_port	2880	MySQL 服务端口。 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。 取值范围：[1,∞)
obmysql_work_thread_count	120	用于执行 SQL 任务的最大线程数。 取值范围：[1,∞)
port	2800	MergeServer 内部任务的服务端口。
query_cache_consistent_expiration_time	10ms	对于一致性读查询，query cache 的失效时间。
query_cache_expiration_time	10s	对于非一致性读查询，query cache 的失效时间。
query_cache_max_param_num	512	绑定变量数小于这个值的 PREPARED STATEMENT 才会使用查询结果缓存。
query_cache_max_result	4KB	只有查询结果大小小于该值的 PREPARED STATEMENT 才会被缓存。

参数	缺省值	说明
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。 取值范围：[1,∞)
query_cache_type	OFF	查询结果缓存功能开关。 • ON：开。 • OFF：关。
read_only	False	如果开启只读模式，除了系统表以外不允许执行任何修改操作。 • True：开。 • False：关
recorder_fifo_path	run/mergeserver.fifo	记录查询流量的命名管道路径。
retry_times	3	MergeServer 向其他 Server 进行网络互交失败时的重试次数。
root_server_ip	-	MergeServer 所在集群的主 RootServer 的 IP。
root_server_port	3500	RootServer 的服务端口。 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。 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。 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。 不建议修改。 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。 取值范围：[10,80]
vtable_location_cache_timeout	8s	虚拟表位置缓存超时时间。

4.4 ChunkServer 配置参数

ChunkServer 配置参数说明如表 4-4 所示。

表 4-4 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。 取值范围：(0,∞) 推荐值：1G

参数	缺省值	说明
block_index_cache_size	512MB	Block 索引缓存大小，主要保存每个 Block 的索引数据。 计算方法：(Disk Size / Block Size）* Block Entry Size 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”。 取值范围：(0,∞) 推荐值：4G
bypass_sstable_loader_thread_num	0	旁路导入的线程数。当值为“0”时，表示不启用旁路导入。 取值范围：[0,10]
check_compress_lib	snappy_1.0:none:lzo_1.0	ChunkServer 启动时检查使用的压缩库。
choose_disk_by_space_threshold	60	数据写磁盘时，如果线程数没有超过该值，则选则并发写的线程数最少的磁盘进行写。如果超过该值，则选磁盘空间最小的写。
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	每合并一个 Tablet 是否都将索引文件写入磁盘。 • True：是。 • False：合并所有 Tablet 后或当 ChunkServer 退出时写入磁盘。

参数	缺省值	说明
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	文件句柄缓存个数。 取值范围：(0,∞)
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。 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。 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 SSTable。 • True：是。 • False：读取数据是才装载 SSTable。
lease_check_interval	5s	检查与 RootServer 租约是否失效的时间间隔。 不建议修改。 取值范围：[5s,5s]
max_merge_thread_num	10	每日合并的最大线程数 取值范围：[1,32]
max_migrate_task_count	2	Tablet 迁移的最大线程数。 取值范围：[1,∞)

参数	缺省值	说明
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。 取值范围：[1,∞)
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。 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。 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。 取值范围：(0,∞) 推荐值：600s
merge_highload_sleep_time	2s	ChunkServer 负载线程超过“merge_threashold_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。 • True：允许。 • False：不允许。

参数	缺省值	说明
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。 单位：微秒
merge_scan_use_preread	True	当进行每日合并时，是否采用异步 I/O 机制。 • True：是。 • False：否。
merge_thread_per_disk	2	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。 不建议配置这个选项。 取值范围：[1,∞)
merge_threshold_load_high	16	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。 取值范围：[1,∞) 推荐值：10
merge_threshold_request_high	3000	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。 取值范围：[1,∞)
merge_timeout	10s	在数据合并时，读取 UpdateServer 数据的超时时间。 取值范围：(0,∞) 推荐值：30s

参数	缺省值	说明
merge_write_sstable_version	2	数据合并后，新 SSTable 的版本。 取值范围：[1,∞)
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。
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“max_sstable_size *（1 + 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。 该值为百分数。 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。 取值范围：(0,∞) 推荐值：20G

参数	缺省值	说明
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。 • True：是。 • False：否。
task_left_time	300ms	请求预留给 MergeServer 的处理时间。 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。
task_queue_size	10000	ChunServer 中，读任务队列大小。 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。 取值范围：[1,∞) 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。 • True：不需要合并。 • False：需要合并。
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。 取值范围：[1,∞)

参数	缺省值	说明
write_sstable_use_dio	True	是否使用 DIO 进行写 SSTable。 • True：是。 • False：不是。

5 监控项参考

OceanBase 监控项存放在表“__all_server_stat”。各监控项的值的类型可以分为以下三种：

- **累计值**：递增变化的值类型，整个生命期内，一直保持增长趋势，除非值溢出。其计算方式有以下两种：
  - 自增：每次计算都原子加 1，相当于计数器。例如“location_cache_hit”指 location cache 命中次数，location cache 每命中一次，该值就加 1。
  - 累加：每次计算都将当前值累加到历史值之和上。例如“sql_select_time”指 select 语句执行时间，其值等于历史所有 select 语句执行时间之和。
- **状态值**：表明当前时刻状态，反应状态参数，无固定变化规律，一般采用定时更新或查询时主动更新的方式改变值，因此本文命名其计算方式为更新。例如“serving_version”指当前正在服务的数据版本号，它采用定时更新的方式改变值。
- **表达式**：依赖于其他监控项的值类型，该类型监控项通过对一个或多个其他监控项进行表达式计算而获取值。例如“get_count”的计算公式是“get_count = hl_get_count + nl_get_count + ll_get_count”。

5.1 RootServer 监控项

RootServer 的监控项说明如表 5-1 所示。

表 5-1 RootServer 监控项

参数	含义	类型
succ_get_count	get 成功次数。	累计值（自增）
succ_scan_count	scan 成功次数。	累计值（自增）
fail_get_count	get 失败次数。	累计值（自增）
fail_scan_count	scan 失败次数。	累计值（自增）

参数	含义	类型
get_obi_role_count	获取 OBI ROLE 的次数统计。	累计值（自增）
migrate_count	SSTable 的迁移任务数。	累计值（自增）
copy_count	SSTable 的复制任务数。	累计值（自增）
get_schema_count	获取 schema 的次数统计。	累计值（自增）
report_version_count	UpdateServer 汇报冻结版本的次数。	累计值（自增）
all_table_count	集群中 Table 总数。	状态值（更新）
all_tablet_count	集群中 TABLET 总数。	状态值（更新）
all_row_count	集群数据总行数。	状态值（更新）
all_data_size	集群数据总量。	状态值（更新）
location_cache_hit	本地缓存命中次数。（当前版本的 OceanBase 暂未使用。）	累计值（自增）
location_cache_miss	本地缓存失效次数。（当前版本的 OceanBase 暂未使用。）	累计值（自增）
rpc_bytes_in	RPC 接收的网络包字节数。	累计值（累加）
rpc_bytes_out	RPC 发送的网络包字节数。	累计值（累加）

5.2 UpdateServer 监控项

UpdateServer 的监控项说明如表 5-2 所示。

表 5-2 UpdateServer 监控项

参数	含义	类型
get_count	随机读取次数。	表达式（hl_get_count + nl_get_count + ll_get_count）
scan_count	顺序扫描次数。	表达式（hl_scan_count + nl_scan_count + ll_scan_count）
trans_count	事务次数。	表达式（hl_trans_count + nl_trans_count + ll_trans_count）
apply_count	写操作次数包括：insert、update、delete、replace、以及索引修改操作。	表达式（hl_apply_count + nl_apply_count + ll_apply_count）
batch_count	事务批处理次数。	累计值（自增）
merge_count	MemTable 行内合并次数。	累计值（自增）
get_qtime	get 操作排队时间。	累计值（累加）
scan_qtime	scan 操作排队时间。	累计值（累加）
apply_qtime	apply 操作排队时间。	表达式（hl_apply_qtime + nl_apply_qtime + ll_apply_qtime）
get_time	get 操作处理时间。	表达式（hl_get_time + nl_get_time + ll_get_time）

参数	含义	类型
scan_time	scan 操作处理时间。	表达式（hl_scan_time + nl_scan_time + ll_scan_time）
trans_time	事务开始处理到响应完成时间。	表达式（hl_trans_time + nl_trans_time + ll_trans_time）
trans_wtime	事务等待时间。	累计值（累加）
trans_htime	事务处理时间。	累计值（累加）
trans_ctime	事务提交等待时间。	累计值（累加）
trans_ftime	事务日志生成、刷盘及同步时间。	累计值（累加）
trans_rtime	事务响应时间。	累计值（累加）
apply_time	写操作处理时间。	表达式（hl_apply_time + nl_apply_time + ll_apply_time）
batch_time	批处理时间。	累计值（累加）
merge_time	MemTable 行内合并时间。	累计值（累加）
memory_total	内存使用总量。	状态值（更新）
memory_limit	可用内存总量。	状态值（更新）
memtable_total	MemTable 内存分配总量。	状态值（更新）
memtable_used	MemTable 内存使用总量。	状态值（更新）
total_rows	MemTable 总行数。	状态值（更新）

参数	含义	类型
active_memtable_limit	活跃 MemTable 可用内存总量。	状态值（更新）
active_memtable_total	活跃 MemTable 内存分配总量。	状态值（更新）
active_memtable_used	活跃 MemTable 内存使用总量。	状态值（更新）
active_total_rows	活跃 MemTable 总行数。	状态值（更新）
frozen_memtable_limit	冻结 MemTable 可用内存总量。	状态值（更新）
frozen_memtable_total	冻结 MemTable 内存分配总量。	状态值（更新）
frozen_memtable_used	冻结 MemTable 内存使用总量。	状态值（更新）
frozen_total_rows	冻结 MemTable 总行数。	状态值（更新）
apply_fail_count	失败的 apply 操作次数。	累计值（自增）
pakcet_long_wait_count	长时间等待的包数。	累计值（自增）
commit_log_size	CommitLog 大小。	累计值（累加）
commit_log_id	CommitLog ID。	状态值（更新）
clog_sync_count	同步日志数。	累计值（自增）
clog_sync_delay	同步日志延时。	累计值（累加）
slow_clog_sync_count	慢速同步日志数。	累计值（自增）

参数	含义	类型
slow_clog_sync_delay	慢速同步日志延时。	累计值（累加）
last_replay_clog_time	上次回放日志时间。	状态值（更新）
frozen_version	冻结版本号。	状态值（更新）
apply_row_count	apply 行数。	累计值（自增）
apply_row_unmerged_cell_count	apply 行内未合并的单元数。	累计值（自增）
ll_get_count	低优先级随机读取次数。	累计值（自增）
ll_scan_count	低优先级顺序扫描次数。	累计值（自增）
ll_apply_count	低优先级写操作次数。	累计值（自增）
ll_trans_count	低优先级事务次数。	累计值（自增）
ll_apply_qtime	低优先级写操作排队时间。	累计值（累加）
nl_get_count	中优先级随机读取次数。	累计值（自增）
nl_scan_count	中优先级顺序扫描次数。	累计值（自增）
nl_apply_count	中优先级写操作次数。	累计值（自增）
nl_trans_count	中优先级事务次数。	累计值（自增）

参数	含义	类型
nl_apply_qtime	中优先级写操作排队时间。	累计值（累加）
hl_get_count	高优先级随机读取次数。	累计值（自增）
hl_scan_count	高优先级顺序扫描次数。	累计值（自增）
hl_apply_count	高优先级写操作次数。	累计值（自增）
hl_trans_count	高优先级事务次数。	累计值（自增）
hl_apply_qtime	高优先级写操作排队时间。	累计值（累加）
ll_get_time	低优先级随机读取处理时间。	累计值（累加）
ll_scan_time	低优先级顺序扫描处理时间。	累计值（累加）
ll_apply_time	低优先级写操作处理时间。	累计值（累加）
ll_trans_time	低优先级事务处理到响应完成时间。	累计值（累加）
nl_get_time	中优先级随机读取处理时间。	累计值（累加）
nl_scan_time	中优先级顺序扫描处理时间。	累计值（累加）
nl_apply_time	中优先级写操作处理时间。	累计值（累加）

参数	含义	类型
nl_trans_time	中优先级事务处理到响应完成时间。	累计值（累加）
hl_get_time	高优先级随机读取处理时间。	累计值（累加）
hl_scan_time	高优先级顺序扫描处理时间。	累计值（累加）
hl_apply_time	高优先级写操作处理时间。	累计值（累加）
hl_trans_time	高优先级事务处理到响应完成时间。	累计值（累加）
lock_wait_time	行锁等待时间。	累计值（累加）
dml_replace_count	REPLACE 语句数。	累计值（自增）
dml_insert_count	INSERT 语句数。	累计值（自增）
dml_update_count	UPDATE 语句数。	累计值（自增）
dml_delete_count	DELETE 语句数。	累计值（自增）
sql_grant_privilege_count	执行用户授权语句次数。	累计值（自增）
sql_revoke_privilege_count	执行撤销权限语句次数。	累计值（自增）
sql_show_grants_count	执行查看权限语句次数。	累计值（自增）
sql_create_user_count	执行新建用户语句次数。	累计值（自增）
sql_drop_user_count	执行删除用户语句次数。	累计值（自增）

参数	含义	类型
sql_lock_user_count	执行锁定用户语句次数。	累计值（自增）
sql_set_password_count	执行修改密码语句次数。	累计值（自增）
sql_rename_user_count	执行修改用户名语句次数。	累计值（自增）
sql_create_table_count	执行创建表语句次数。	累计值（自增）
sql_create_index_count	执行创建索引语句次数。	累计值（自增）
sql_drop_table_count	执行删除表语句次数。	累计值（自增）
sql_ps_allocator_count	Prepare Statement Allocator 的个数，它是一个动态值，新创建一个 Prepare Statement 语句，就增加一个 Prepare Statement Allocator，关闭 Prepare Statement 语句就减少一个 PS Allocator。	状态值（更新）
sql_insert_cache_hit	INSERT 操作缓存命中次数。	累计值（自增）
sql_insert_cache_miss	INSER 操作缓存失效次数。	累计值（自增）

参数	含义	类型
sql_update_cache_hit	UPDATE 操作缓存命中次数。	累计值（自增）
sql_update_cache_miss	UPDATE 操作缓存失效次数。	累计值（自增）
sql_query_cache_hit	query cache 命中次数。	累计值（自增）
sql_query_cache_miss	query cache 失效次数。	累计值（自增）
distinct_stmt_count	执行不同 SQL 语句的种类。	累计值（自增）
ps_count	活跃的 PREPARE Statement 语句数。	状态值（更新）
location_cache_hit	本地缓存命中次数。（当前版本的 OceanBase 暂未使用。）	累计值（自增）
location_cache_miss	本地缓存失效次数。（当前版本的 OceanBase 暂未使用。）	累计值（自增）
rpc_bytes_in	RPC 接收的网络包字节数。	累计值（累加）
rpc_bytes_out	RPC 发送的网络包字节数。	累计值（累加）
block_index_cache_hit	Block 索引缓存命中次数。	累计值（自增）
block_index_cache_miss	Block 索引缓存失效次数。	累计值（自增）

参数	含义	类型
block_cache_hit	Block 缓存命中次数。	累计值（自增）
block_cache_miss	Block 缓存失效次数。	累计值（自增）
sstable_disk_io_num	SSTable 读操作次数。	累计值（自增）
sstable_disk_io_bytes	SSTable 读操作数据量。	累计值（累加）
sstable_disk_io_write_num	SSTable 写操作次数。	累计值（自增）
sstable_disk_io_write_bytes	SSTable 写操作数据量。	累计值（累加）
sstable_row_cache_hit	SSTable row cache 命中次数。	累计值（自增）
sstable_row_cache_miss	SSTable row cache 失效次数	累计值（自增）
sstable_get_rows	SSTable “get_row”操作次数。	累计值（自增）
sstable_scan_rows	SSTable “scan_row”操作次数。	累计值（自增）

5.3 MergeServer 监控项

MergeServer 的监控项说明如表 5-3 所示。

表 5-3 MergeServer 监控项

参数	含义	类型
nb_get_count	服务 get 请求的次数。	累计值（自增）

参数	含义	类型
nb_get_time	服务 get 请求的总时间。	累计值（累加）
nb_scan_count	服务 scan 请求的次数。	累计值（自增）
nb_scan_time	服务 scan 请求的总时间。	累计值（累加）
get_event_count	get 事件次数。	累计值（自增）
get_event_time	get 事件处理总时间。	累计值（累加）
scan_event_count	can 事件次数。	累计值（自增）
scan_event_time	scan 事件处理总时间。	累计值（累加）
ms_memory_limit	最大内存限制。	状态值（更新）
ms_memory_total	内存使用总量。	状态值（更新）
ms_memory_parser	parser 内存使用量。	状态值（更新）
ms_memory_transformer	transformer 内存使用量。	状态值（更新）
ms_memory_ps_plan	ps plan 内存使用量。	状态值（更新）
ms_memory_rpc_request	rpc request 内存使用量。	状态值（更新）
ms_memory_sql_array	sql array 内存使用量。	状态值（更新）
ms_memory_expression	expression 内存使用量。	状态值（更新）

参数	含义	类型
ms_memory_row_store	row store 内存使用量。	状态值（更新）
ms_memory_session	session 内存使用量。	状态值（更新）
sql_grant_privilege_count	执行用户授权语句次数。	累计值（自增）
sql_revoke_privilege_count	执行撤销权限语句次数。	累计值（自增）
sql_show_grants_count	执行查看权限语句次数。	累计值（自增）
sql_create_user_count	执行新建用户语句次数。	累计值（自增）
sql_drop_user_count	执行删除用户语句次数。	累计值（自增）
sql_lock_user_count	执行锁定用户语句次数。	累计值（自增）
sql_set_password_count	执行修改密码语句次数。	累计值（自增）
sql_rename_user_count	执行修改用户名语句次数。	累计值（自增）
sql_create_table_count	执行创建表语句次数。	累计值（自增）
sql_drop_table_count	执行删除表语句次数。	累计值（自增）

参数	含义	类型
sql_ps_allocator_count	Prepare Statement Allocator 的个数，它是一个动态值，新创建一个 Prepare Statement 语句，就增加一个 Prepare Statement Allocator，关闭 Prepare Statement 语句就减少一个 PS Allocator。	状态值（更新）
sql_insert_cache_hit	INSERT 操作缓存命中次数。	累计值（自增）
sql_insert_cache_miss	INSERT 操作缓存未命中次数。	累计值（自增）
sql_update_cache_hit	UPDATE 操作缓存命中次数。	累计值（自增）
sql_update_cache_miss	UPDATE 操作缓存未命中次数。	累计值（自增）
sql_query_cache_hit	SQL 请求缓存命中次数。	累计值（自增）
sql_query_cache_miss	SQL 请求缓存失效次数。	累计值（自增）
distinct_stmt_count	执行不同 SQL 语句的种类。	累计值（自增）
ps_count	活跃的 PREPARE Statement 语句数。	状态值（更新）
sql_succ_query_count	执行 SQL 请求成功次数。	累计值（自增）
sql_fail_query_count	执行 SQL 请求失败次数。	累计值（自增）

参数	含义	类型
sql_succ_prepare_count	执行 PREPARE 成功次数。	累计值（自增）
sql_fail_prepare_count	执行 PREPARE 失败次数。	累计值（自增）
sql_succ_exec_count	执行 EXECUTE 成功次数。	累计值（自增）
sql_fail_exec_count	执行 EXECUTE 失败次数。	累计值（自增）
sql_succ_close_count	关闭 SQL 请求成功的次数。	累计值（自增）
sql_fail_close_count	关闭 SQL 请求失败的次数。	累计值（自增）
sql_succ_login_count	登录 OceanBase 成功次数。	累计值（自增）
sql_fail_login_count	登录 OceanBase 失败次数。	累计值（自增）
sql_logout_count	退出 OceanBase 次数。	累计值（自增）
sql_compound_count	compound 语句个数。	累计值（自增）
sql_compound_time	compound 语句执行时间。	累计值（累加）
sql_select_count	执行 SELECT 次数。	累计值（自增）
sql_select_time	SELECT 语句执行时间。	累计值（累加）
sql_insert_count	执行 INSERT 次数。	累计值（自增）

参数	含义	类型
sql_insert_time	INSERT 语句执行时间。	累计值（累加）
sql_replace_count	执行 REPLACE 次数。	累计值（自增）
sql_replace_time	REPLACE 语句执行时间。	累计值（累加）
sql_update_count	执行 UPDATE 次数。	累计值（自增）
sql_update_time	UPDATE 语句执行时间。	累计值（累加）
sql_delete_count	执行 DELETE 次数。	累计值（自增）
sql_delete_time	DELETE 语句执行时间。	累计值（累加）
sql_query_bytes	SQL 请求数据总量。	累计值（累加）
sql_commit_count	执行 COMMIT 次数。	累计值（自增）
sql_rollback_count	执行 ROLLBACK 次数。	累计值（自增）
sql_autocommit_on_count	设置 autocommit on 语句个数。	累计值（自增）
sql_autocommit_off_count	设置 autocommit off 语句个数。	累计值（自增）

参数	含义	类型
location_cache_hit	SSTable 的本地缓存命中次数。MergeServer 每次向 ChunkServer 取数据时，优先从 location cache 中获取 location 信息，如果命中，则 location_cache_hit 加一。	累计值（自增）
location_cache_miss	SSTable 的本地缓存失效次数。MergeServer 每次向 ChunkServer 取数据时，优先从 location cache 中获取 location 信息，如果未命中，则 location_cache_miss 加一。	累计值（自增）
rpc_bytes_in	RPC 接收的网络包字节数。	累计值（累加）
rpc_bytes_out	RPC 发送的网络包字节数。	累计值（累加）

5.4 ChunkServer 监控项

ChunkServer 的监控项说明如表 5-4 所示。

表 5-4 ChunkServer 监控项

参数	含义	类型
serving_version	当前服务的数据版本号。	状态值（更新）
old_ver_tablets_num	旧版本的 TABLET 个数。	状态值（更新）

参数	含义	类型
old_ver_merged_tablets_num	旧版本完成合并的 tablet 个数。	状态值（更新）
new_ver_tablets_num	新版本的 TABLET 个数。	状态值（更新）
memory_used_network	network 内存使用量。	状态值（更新）
memory_used_thread_buffer	thread buffer 内存使用量。	状态值（更新）
memory_used_tablet	tablet 内存使用量。	状态值（更新）
memory_used_bi_cache	block index cache 内存使用量。	状态值（更新）
memory_used_block_cache	block cache 内存使用量。	状态值（更新）
memory_used_join_cache	join cache 内存使用量。	状态值（更新）
memory_used_sstable_row_cache	sstable row cache 内存使用量。	状态值（更新）
memory_used_merge_buffer	merge buffer 内存使用量。	状态值（更新）
request_count	请求次数。	累计值（自增）
request_count_per_second	每秒的请求次数。	表达式（计算 request_count 变化速率）
queue_wait_time	请求队列等待时间。	累计值（累加）

参数	含义	类型
get_count	get 请求次数。	累计值（自增）
scan_count	scan 请求次数。	累计值（自增）
get_time	get 请求处理时间。	累计值（累加）
scan_time	scan 请求处理时间。	累计值（累加）
get_bytes	get 请求数据量。	累计值（累加）
scan_bytes	scan 请求数据量。	累计值（累加）
location_cache_hit	本地缓存命中次数。（当前版本的 OceanBase 暂未使用。）	累计值（自增）
location_cache_miss	本地缓存失效次数。（当前版本的 OceanBase 暂未使用。）	累计值（自增）
rpc_bytes_in	RPC 接收的网络包字节数。	累计值（累加）
rpc_bytes_out	RPC 发送的网络包字节数。	累计值（累加）
block_index_cache_hit	Block 索引缓存命中次数。	累计值（自增）
block_index_cache_miss	Block 索引缓存失效次数。	累计值（自增）
block_cache_hit	Block 缓存命中次数。	累计值（自增）

参数	含义	类型
block_cache_miss	Block 缓存失效次数。	累计值（自增）
sstable_disk_io_num	SSTable 读操作次数。	累计值（自增）
sstable_disk_io_bytes	SSTable 读操作数据量。	累计值（累加）
sstable_disk_io_write_num	SSTable 写操作次数。	累计值（自增）
sstable_disk_io_write_bytes	SSTable 写操作数据量。	累计值（累加）
sstable_row_cache_hit	SSTable row cache 命中次数。	累计值（自增）
sstable_row_cache_miss	SSTable row cache 失效次数	累计值（自增）
sstable_get_rows	SSTable “get_row”操作次数。	累计值（自增）
sstable_scan_rows	SSTable “scan_row”操作次数。	累计值（自增）

6 术语

B

表

一个表由若干列（在 schema 中定义）和任意行组成，有些表的每一行都存储了 schema 中定义的每一列，这样的表是稠密的，如基准数据的表一般是稠密的；另外一些表行只存储了部分的列，例如 UpdateServer 中的表的修改增量，这样的表是稀疏的。

C

Commit Log

操作日志。UPS 每次更新操作会产生一条 commit log，并追加到已有的 commit log 之后，并通过同步 commit log 实现主备复制。完整的 commit log 序列代表了 OceanBase 的更新历史。

CS

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。

D

DIO

Direct Input-Output，直接输入输出。

H

行

一个行由若干列组成，有些时候其中的部分列构成主键（row key）并且整个表按主键顺序存储。有些表（如 select 指令的结果）可以不包含主键。

J

基准数据

OceanBase 存储的某个时间点以前的数据快照，作为静态数据以 SSTable 的格式存放在 ChunkServer 上。

L

LMS

Listener MergeServer。只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。

列

一个列由列 ID(column_id)及其值(column_value)组成。

M

MS

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。

MVCC

Multi-Version Concurrency Control，多版本并发控制。

慢查询

超过指定时间的 SQL 语句查询。

O

OB

OceanBase，核心系统研发部的海量存储系统。

OLAP

On-Line Analytical Processing，联机分析处理。

OLTP

On-Line Transaction Processing，联机事务处理。

R

RPC

Remote Process Call，远程方法调用。

RS

RootServer，主控服务器。主要进行集群管理、数据分布和副本管理，并提供 Listener 服务。

S

schema

表的列的类型、值范围等以及该表与其他表的 join 等关系称为表的 schema。

U

UPS

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。

UPS Master

UpdateServer Master，主 UpdateServer。主要处理实际读写请求。

UPS Slave

UpdateServer Slave，备 UpdateServer。主要用于实时备份。

V

VIP

Virtual IP，虚拟 IP 地址。RootServer 主机由 VIP 决定。

Z

增量数据

OceanBase 存储的某个时间点以后的更新数据，存放在 UpdateServer 上，在内存中以 btree 和 hash table 的形式组织。