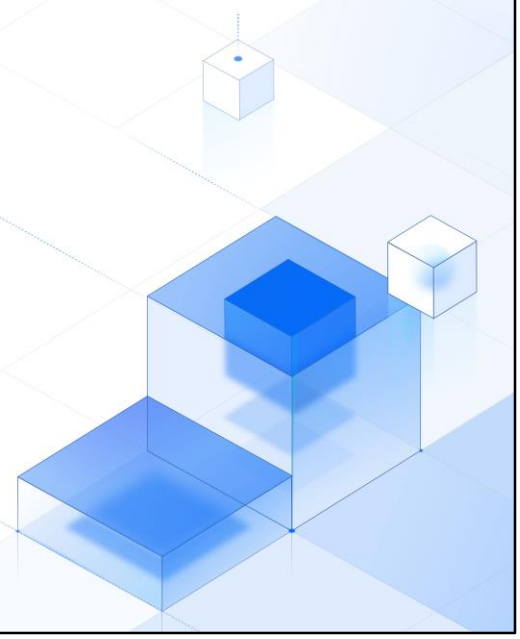
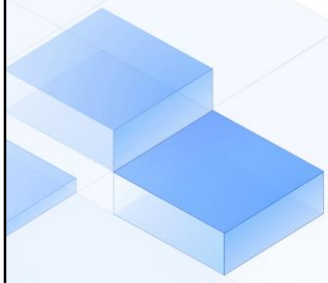
OCEANBASE

OBCP 认证培训



目录



第一章/ OB 分布式架构高级技术

第二章 / OB 存储引擎高级技术

第三章 / OB SQL 引擎高级技术

第四章/ OB SQL调优

第五章 / OB 分布式事务高级技术

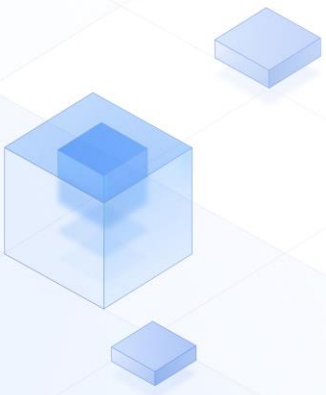
第六章/ OBProxy 路由与使用运维

第七章 / OB 备份与恢复

第八章 / OceanBase运维、 监控与异常处理

OCEANBASE

目录

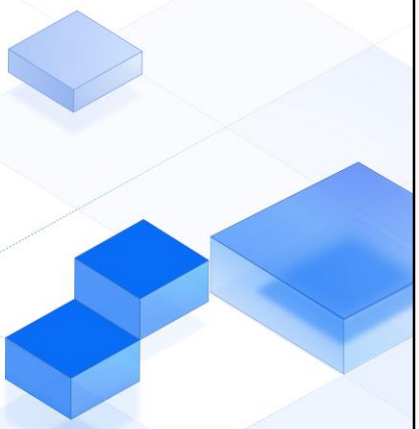


第八章 / OceanBase 运维、监控与异常处理

8.1 用户权限管理
8.2 日志查询
8.2 日常运维操作
8.3 数据库监控
8.4 常见异常处理
8.5 灾难恢复

OCEANBASE

8.1 用户权限管理



OCEANBASE

8.1 黑屏：用户管理

- 数据库用户管理操作包括新建用户、删除用户、修改密码、修改用户名、锁定用户、用户授权和撤销授权等
- 用户分为两类：系统租户下的用户，一般租户下的用户.

  创建用户时，如果当前会话的租户为系统租户，则新建的用户为系统租户用户，反之为一般租户下的用户。

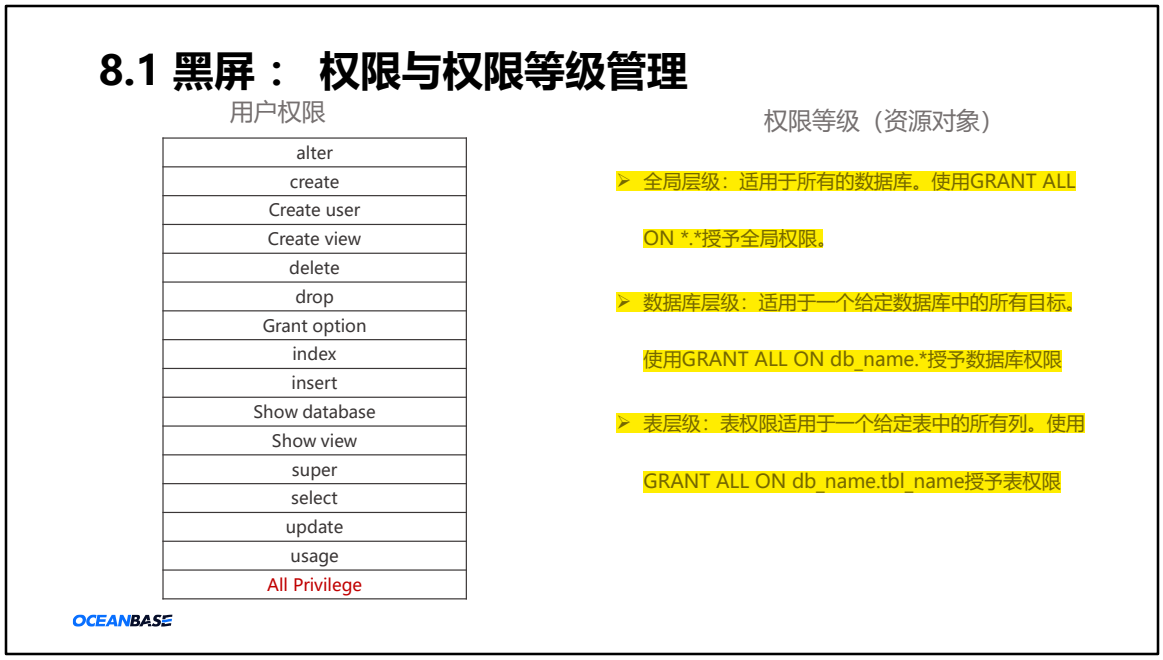
  - 创建用户：create user ‘my_user’ identified by ‘my_password’
  - 删除用户：drop user ‘my_user’
  - 给用户相应权限： grant [用户权限】on [资源对象】 to username
  - 收回权限：revoke [用户权限】on [资源对象】 to username
  - 查看用户： show grants for username

OCEANBASE

OceanBase 1.0中用户分为两类：系统租户下的用户，一般租户下的用户。创建用户时，如果Session当前租户为系统租户，则新建的用户为系统租户用户；反之为一般租户下的用户。

用户名称在租户内是唯一的，不同租户下的用户可以同名。用户名@租户名在系统全局唯一。为区别系统租户和一般租户下的用户，系统租户下的用户名称

使用特定前缀。
系统租户和普通租户都有一个内置用户root，系统租户的root为系统管理员和普通租户的root为租户管理员，购买了某个普通租户的客户得到普通租户root和密码，进行本租户范围的管理工作。
一般租户下的用户只能拥有该租户下对象的访问权限，权限和MySQL兼容；系统租户下的用户可以被授予跨租户的对象访问权限。当前系统租户下的用户不允许访问一般租户下的用户表数据。
用户在登录OceanBase系统时需指定唯一的租户名。对于系统租户下的用户，在登录后，可以使用CHANGE EFFECTIVE TENANT *tenantname*语句来切换当前访问的租户；对于一般租户下的用户，不能切换到其他租户。



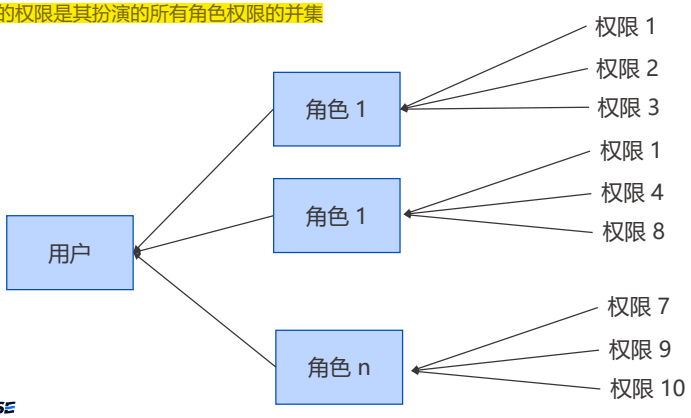
各个权限表示的信息描述如下：
ALTER权限：允许用户使用ALTER TABLE来改变表的结构。但ALTER TABLE执行RENMAE操作的时候，需要旧表上的ALTER、DROP权限，还有新表上的CREATE、INSERT权限（RENAME TABLE SQL也是如此）。
CREATE权限：允许创建database和table 。
DELETE权限：允许从database的table中删除行。
DROP权限：允许用户drop存在的databases、tables、views。DROP权限同时在TRUNCATE TABLE时也需要。如果在user层grant DROP权限给一个用户，那么这个用户可以drop掉系统除oceanbase,information_schema,mysql外的所有databases。
GRANT OPTION权限：用户可以给其他用户权限或者撤销其他用户拥有的权限。
INSERT权限：允许在database的tables中插入行。
UPDATE权限：允许database中tables的rows被更新。
SELECT权限：允许用户从database的table中select rows。SELECT权限也会在其他需要读取列值的语句中需要，例如UPDATE、DELETE、INSERT ON DUPLICATE KEY UPDATE。
IDNEX权限：允许用户创建或删除index。INDEX适用于已经存在的tables。如果用户拥有一张表的create权限，那么用户可以在create table语句中定义index。
CREATE VIEW权限：允许用户执行CREATE VIEW。（注：CREATE VIEW AS同时

需要select语句需要的权限，CREATE OR REPLACE VIEW同时需要VIEW的DROP权限）
SHOW VIEW权限：允许用户SHOW CREATE VIEW。SHOW CREATE VIEW SQL执行同时需要SHOW VIEW和SELECT权限。
SHOW DATABASES权限：允许用户查看database的名字通过SHOW DATABASE语句。对于没有这项权限的用户，只能看到他们就有权限的database。
CREATE USER权限：允许用户CREATE USER、DROP USER、RENAME USER、SET PASSWORD、LOCK USER。
SUPER权限：对于普通租户，可以kill他人登录的session，允许修改全局系统变量。对于系统租户，还允许通过ALTER SYSTEM执行各项系统控制语句。
PROCESS权限：展示登录租户下所有人登录session的权限。
ALL or ALL PRIVILEGES权限：是用于速记的标记符。它代表对于给定权限级别的所有权限（除了GRANT OPTION）。
USAGE权限。代表NO PRIVILEGES。

8.1 白屏：OCP 平台的用户管理

OCP 平台的用户，是OCP 平台的使用者、功能系统的操作者（非黑屏用户）。 OCP 用户主要有以下几个关键概念：

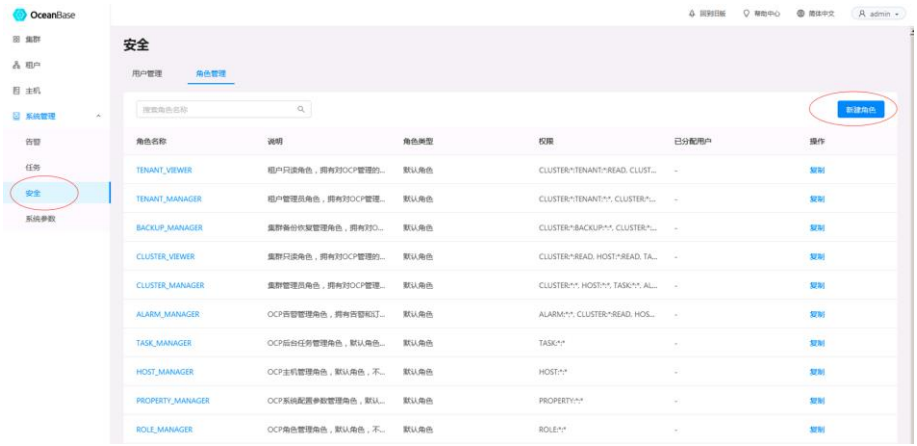
- 角色： 角色是一组权限的集合，是权限的载体。
- 用户： 每个用户都必须扮演一个或者多个角色，具备这些角色所包含的权限
- 用户的权限是其扮演的所有角色权限的并集



OCEANBASE

8.1 白屏：OCP 平台增加角色

OCP 管理界面—>安全—>角色管理： 查看系统已有的默认角色，或者创建新的角色并分配权限



8.1 白屏：OCP 平台增加新用户

OCP 管理界面—>安全—>用户管理：

- 默认 admin 系统管理员用户
- 增加新的用户，并给新用户关联角色

创建好新的用户后，可以使用新的用户登录 OCP

OCEANBASE


8.2 日志查询
OCEANBASE

8.2 日志概述

- Observer日志：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
  - /home/admin/oceanbase/log
- 事务/存储日志：OceanBase在事务执行过程中，会持久化事务/存储日志
  - /data/log1/集群名/

OCEANBASE

8.2 事务/存储日志清单

日志名称	日志路径	说明
clog	OBServer服务器的“~/datadir/clog”目录下。	Commit Log，所有Partition共用，日志可能是乱序的，记录事务、PartitionService提供的原始日志内容。此目录下的日志基于Paxos协议在多个副本之间同步。
ilog	OBServer服务器的“~/datadir/ilog”目录下。	Index Log，所有Partition共用，单Partition内部日志有序，记录Partition内部log_id>clog(file_id, offset)的索引信息。每个副本自行记录。
slog	OBServer服务器的“~/datadir/slog”目录下。	记录Storage log,指SSTable操作日志信息。

OCEANBASE

8.2 关于clog

事务的日志包括：redo log, prepare log, commit log, abort log, clear log等

- redo log记录了事务的具体操作，比如某一行数据的某个字段从A修改为B
- prepare log记录了事务的prepare状态
- commit log表示这个事务成功commit，并记录commit信息，比如事务的全局版本号
- clear log用于通知事务清理事务上下文
- abort log表示这个事务被回滚

所有的事务日志信息，不用于用户查看和定位系统问题

OCEANBASE

clog是指广义的 Commit Log，代表整个事务的所有日志信息。

8.2 Observer日志级别

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。
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。
TRACE	与DEBUG相比更细致化的记录事件消息。

8.2 Observer日志格式

- 日志记录主要组成：记录时间、日志级别、[模块名]、文件名：行号、线程ID和日志内容

[time] log_level [module_name] function_name (file_name:file_no) [thread_id][Ytrace_id0-trace_id1] [log=last_log_print_time]log_data

- 例子：

[2015-08-06 15:34:30.006962] INFO [SQL.RESV] ob_basic_session_info.cpp:220 [84753][Y0-0] use database success.(database_name=oceanbase)

表示执行SQL指令use database oceanbase成功

OCEANBASE

主要用于协助排查OB问题

8.2 Observer日志注意事项

- Observer日志放在 /home/admin/oceanbase/log 目录，包含以下部分：
  - ✓ election.log , election.log.wf: 选举模块日志
  - ✓ rootservice.log , rootservice.log.wf: rootservice模块日志
  - ✓ observer.log , observer.log.wf: 除以上两个模块外其它所有日志
- 日志写满256MB时，会做日志文件切换，原日志文件名加上 .%Y%m%d%H%M%S 格式的时间，如 observer.log.20161230235020

OCEANBASE

8.2 Observer日志注意事项

- OceanBase默认不会自动清理日志。另外OceanBase可按日志个数回收日志，通过以下两个配置项控制：

  enable_syslog_recycle: 默认 false

  max_syslog_file_count: 默认0

- 日志限流控制参数 syslog_io_bandwidth_limit
- Warning及以上信息生成单独日志文件控制参数：enable_syslog_wf ： 默认 true

OCEANBASE

如果日志量大，syslog_io_bandwidth_limit参数配置较低，日志输出会不全（trim掉）
如果syslog_io_bandwidth_limit参数配置较高，日志量大，会抢占磁盘io资源，影响数据库性能

8.2 OB MySQL错误码

- 如果一个错误码的值大于4000，表明它是OB特有的错误码
  - 参考“OceanBase 1.0 参考指南”
- 如果在4000以内，表示它是MySQL兼容错误：
  - MySQL Server端错误码范围1000-2000，客户端错误码2000-3000, 1-1000预留

  参考：
  - http://dev.mysql.com/doc/refman/5.1/en/error-messages-server.html
  - http://dev.mysql.com/doc/refman/5.1/en/error-messages-client.html

OCEANBASE

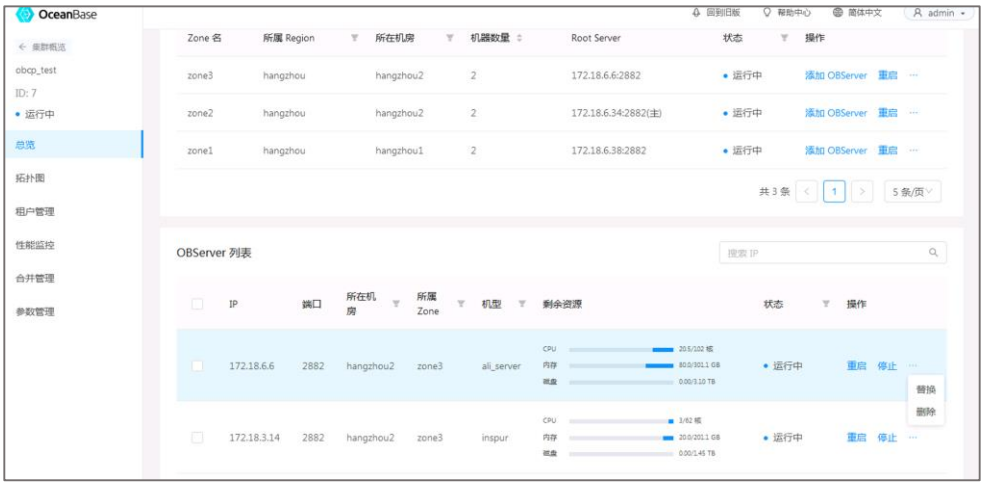
8.2 OB MySQL错误码取值范围说明

错误码范围	说明
[-1 , -4000)	和MySQL兼容的错误码。MySQL Server端错误码范围1000-2000，客户端错误码2000-3000, 1-1000预留。 MySQL服务端错误码,请参考http://dev.mysql.com/doc/refman/5.1/en/error-messages-server.html MySQL客户端错误码，请参考http://dev.mysql.com/doc/refman/5.1/en/error-messages-client.html
[-4000 , -4500)	通用错误码，含sstable等。
[-4500 , -5000)	RootService错误码。
[-5000 , -6000)	SQL错误码，包含各种schema相关错误。
[-6000, -7000)	事务引擎错误码，包含clog, memtable等。
[-7000 , -7100)	选举模块错误码
[-8000 , -9000)	致命错误，客户端收到8XXX错误，需要关闭SQL连接

OCEANBASE


8.3 日常运维操作
OCEANBASE

8.3 白屏： 集群、Zone、Observer 常用运维操作



OCEANBASE

断电、软件crash等场景

8.3 常用运维操作：时钟同步

- OceanBase从Partition的多个副本中选出主对外提供服务。为避免Paxos的活锁问题，OceanBase采用一种基于时钟的选举算法选主
- 检查ntp时间是否同步，OceanBase容忍的集群内时钟偏差为100ms
- ntpq –p，输出的offset应小于50ms
- clockdiff

OCEANBASE

8.3 黑屏：集群运维管理

在集群中启动或停止 Zone 的操作通常用于允许或禁止 Zone 内的所有物理服务器对外提供服务的需求场景。

1、 查看 Zone 的状态

Select * from __all_zone ;

2、 启动或停止 Zone

ALTER SYSTEM {START|STOP|FORCE STOP} ZONE [Zone_Name];

示例 1：ALTER SYSTEM START ZONE Zone1;

示例 2：ALTER SYSTEM STOP ZONE Zone1;

3、修改 Zone 信息

ALTER SYSTEM {ALTER|CHANGE|MODIFY} ZONE [Zone_Name] SET [Zone_Option_List];

OCEANBASE

Zone_option_list：region，IDC, Zone_type (READONLY, READWRITE)

断电、软件crash等场景

8.3 黑屏:Observer 运维管理

1、查看 observer 的信息

select * from __all_server ;

select * from __all_server_event_history;

2、管理 OBServer 状态： 进程启动后，在集群中 OBServer 作为节点单元的管理类似 Zone 的管理。

Start Server 操作：

ALTER SYSTEM START SERVER 'ip:port' [,'ip:port'...] [ZONE='zone ']

示例：alter system start server '192.168.100.1:2882'

Stop Server 操作：

ALTER SYSTEM STOP SERVER 'ip:port' [,'ip:port'...] [ZONE='zone ']

示例语：alter system stop server '192.168.100.1:2882' zone='z1'

stopped 并非等价于进程退出，进程可能仍然在运行，仅仅是集群认为该节点为 stopped 状态。

OCEANBASE

断电、软件crash等场景

8.3 黑屏:Observer 服务管理（进程）

1、查看 observer 进程： 登录 OceanBase Server 所在的宿主机

```
ps -ef |grep observer
```

2、启动 observer 进程： 登录 OceanBase Server 主机

```
cd /home/admin/oceanbase/
./bin/observer [启动参数]
```

可以运行 ./bin/observer --help 查看 observer 启动参数的详细信息。

3、停止 observer 进程:

```
kill -15 `pgrep observer`
kill -9 `pgrep observer`
```

OCEANBASE

断电、软件crash等场景

8.3 黑屏：observer服务启动恢复

- 注意：由于增删改数据在内存中，进程启动后
  - 需要与其他副本同步，将clog或者ssd基线数据进行同步（补齐）
  - 需要将上一次合并之后的内存数据恢复出来（clog回放），才能提供服务
- 这一过程可能需要数分钟
- 这一过程结束后，该OBServer才能对外提供服务
- 可以在停止 observer 服务前执行合并（alter system major freeze;），以加快 observer 服务恢复过程。

OCEANBASE

停机时间短（分钟或者小时级别），一般只追齐clog
停机时间长（天级别），clog落后太多，会直接追齐ssd基线数据，然后补齐合并版本后的clog
回放时间取决于内存中数据量
是否能对外提供服务，查询__all_server表，start service time

8.3 黑屏：服务停止（停机运维）

机器需要运维操作时，需要停止OceanBase服务进程

1. 系统租户登陆，确定运维时长，如果大于1小时但小于1天，为了避免服务恢复后的补副本操作，需要设置永久下线时间 （alter system set server_permanent_offline_time = ‘86400s’）
2. 将服务从当前 observer 切走，保证停服务的时候，对于业务没有影响（alter system stop server 'ip地址:2882'；）
3. 检查主副本都切走（select count(*) from __all_virtual_table t, __all_virtual_meta_table m where t.table_id=m.table_id and role=1 and m.svr_ip='ip地址'；），返回值应为0
4. 停止进程 kill -15 <observer pid>

OCEANBASE

涉及停机操作的运维，比如停机运维机器、机器搬迁等

8.3 黑屏：服务恢复（停机运维结束）

机器需要运维操作结束后，需要恢复OceanBase服务进程

1. 机器上电
2. 检查该机器ntp同步状态和服务运行情况
3. admin用户启动observer进程
4. 系统租户登陆，启动server（alter system start server 'ip地址:2882'; ）
5. 检查__all_server表，查看status为 ‘ active ’ 且 ‘ start_service_time ‘的值不为NULL，则表示observer正常启动并开始提供服务
6. 将永久下线时间改回默认值3600s（ alter system set server_permanent_offline_time = ’3600s')

OCEANBASE

8.3 黑屏：故障节点替换

首先要确保集群中有足够的冗余资源（observer），可以代替故障节点进行工作

1. 系统租户登陆，stop server，确保主副本都切走。
2. 为目标zone添加新的server（alter system add server 'ip地址:2882' ZONE 'zone1'; ）
3. 将故障server下线（alter system delete server 'ip地址:2882' ZONE 'zone1’ ; ）

OB 会自动将被下线Observer的 Unit 迁移至新添加的 Observer 上。

4. 检查__all_server表检查server状态，旧 observer 的信息已经消失。

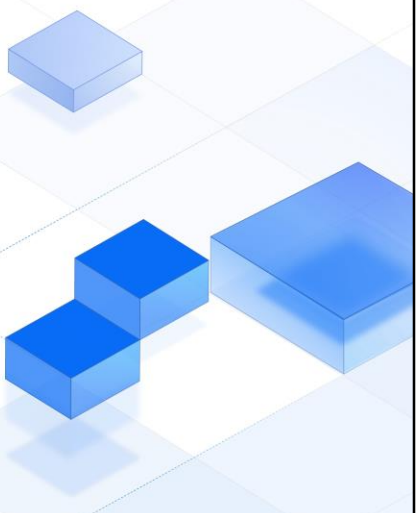
OCEANBASE

8.3 常用运维操作：容量不足

- 内存
  - OB是准内存数据库，任何写操作都需要消耗内存资源，只有合并和转储操作能够释放内存资源，所以当合并和转储速度长时间低于内存消耗速度时，内存最终将被耗尽，服务能力跌零
  - 调大租户内存
  - 转储 / 合并
- 外存
  - 运行日志盘满：可清空较老的日志
  - clog盘满：查询表__all_virtual_server_clog_stat，清除较老的日志，再合并
  - 数据文件满：扩容，或将较老的数据迁移到历史库，再合并

OCEANBASE

8.4 数据库监控



OCEANBASE

8.4.1 系统监控视图： 系统视图

- 每当一个observer启动之后，其对应的v$系列视图便可用于诊断查询
- gv$视图的不同之处在于，会从集群所有的observer查询结果并返回
- 对于等待事件和统计事件相关字段，时间类型单位如无特殊说明是微秒

OCEANBASE

视图跟oracle兼容的
gv$： 整个集群
v$：当前observer

8.4.2 实施监控： zone状态

- select * from __all_zone;
- 关注以下点:
  - is_merge_error 对应的value是否是0
  - status 是否全为ACTIVE

OCEANBASE

8.4.2 实时监控：server状态

- select zone, svr_ip, status from __all_server;
- 查看是否所有server状态都是active
- inactive状态的server说明对应机器宕机 / 断网，或上面的observer进程退出，此时应登录到该机器检查，并启动上面的observer服务
- 检查 start_service_time / last_offline_time

OCEANBASE

8.4.2 实时监控资源分配率

- select zone, svr_ip, cpu_assigned_percent, mem_assigned_percent, disk_assigned_percent from __all_virtual_server_stat;
- 如果某个zone中所有server的某项指标(cpu_assigned_percent, mem_assigned_percent)都比较高(>90)，后续加租户或扩租户资源可能会因资源不够失败，可考虑集群扩容

OCEANBASE

8.4.2 实时监控：查看机器剩余资源

select b.zone, a.svr_ip, a.cpu_total, a.cpu_assigned cpu_ass, a.cpu_assigned_percent cpu_ass_percent,round(a.mem_total/1024/1024/1024, 2) as mem_total, round(a.mem_assigned/1024/1024/1024, 2) mem_ass,round((a.mem_total-a.mem_assigned)/1024/1024/1024, 2) as mem_free,a.mem_assigned_percent mem_ass_percent from __all_virtual_server_stat a,__all_server b where a.svr_ip = b.svr_ip order by zone,cpu_assigned_percent desc;

zone	svr_ip	cpu_total	cpu_ass	cpu_ass_percent	mem_total	mem_ass	mem_free	mem_ass_percent
ZONE_1	[redacted]	30	15.5	51	161.08	43.00	118.08	26
ZONE_2	[redacted]	30	15.5	51	161.08	43.00	118.08	26
ZONE_3	[redacted]	30	15.5	51	161.08	43.00	118.08	26

OCEANBASE

8.4.1 系统监控视图：gv$memory

展示当前租户在所有ObServer上各个模块的内存使用情况，基于__all_virtual_memory_info创建

- CONTEXT 对应模块名，即mod_name
- COUNT 分配内存的次数，每次为该模块分配内存均加1
- USED 模块使用的内存大小

```
mysql> select * from gv$memory where USED>0;
+-----------+---------------+------+---------------------+-------+--------+-------------+------------+
| TENANT_ID | IP            | PORT | CONTEXT             | COUNT | USED   | ALLOC_COUNT | FREE_COUNT |
+-----------+---------------+------+---------------------+-------+--------+-------------+------------+
|      1152 | 100.81.140.78 | 2882 | OB_ELECTION         |     5 |  89160 |           5 |          0 |
|      1152 | 100.81.140.78 | 2882 | OB_MEMTABLE_OBJECT  |    10 |  99840 |         170 |        160 |
|      1152 | 100.81.140.78 | 2882 | OB_LOG_INDEX_MOD    |     5 |  40960 |           5 |          0 |
|      1152 | 100.81.140.80 | 2882 | OB_ELECTION         |     5 |  89160 |           5 |          0 |
|      1152 | 100.81.140.80 | 2882 | OB_MEMTABLE_OBJECT  |    10 |  99840 |         170 |        160 |
|      1152 | 100.81.140.80 | 2882 | OB_LOG_INDEX_MOD    |     5 |  40960 |           5 |          0 |
|      1152 | 100.81.140.82 | 2882 | OB_SQL_PHY_PLAN     |     5 | 327680 |         762 |        757 |
|      1152 | 100.81.140.82 | 2882 | OB_SQL_PLAN_CACHE   |    22 |   2334 |         730 |        708 |
|      1152 | 100.81.140.82 | 2882 | OB_ELECTION         |     5 |  89160 |           5 |          0 |
|      1152 | 100.81.140.82 | 2882 | OB_MEMTABLE_OBJECT  |    10 |  99840 |         170 |        160 |
|      1152 | 100.81.140.82 | 2882 | OB_LOG_INDEX_MOD    |     5 |  40960 |           5 |          0 |
```

OCEANBASE

主要用于研发排查问题，看资源都被哪些模块占用了
gv$：整个集群
v$：当前observer

8.4.1 系统监控视图：gv$memstore

展示当前租户在所有ObServer上memstore的信息，基于__all_virtual_tenant_memstore_info 创建

- ACTIVE 当前租户活跃的内存大小
- TOTAL 当前租户memstore总和
- FREEZE_TRIGGER 触发冻结的内存大小

```
Oceanbase>desc gv$memstore;
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| TENANT_ID      | bigint(20)  | NO   |     | NULL    |       |
| IP             | varchar(32) | NO   |     | NULL    |       |
| PORT           | bigint(20)  | NO   |     | NULL    |       |
| ACTIVE         | bigint(20)  | NO   |     | NULL    |       |
| TOTAL          | bigint(20)  | NO   |     | NULL    |       |
| FREEZE_TRIGGER | bigint(20)  | NO   |     | NULL    |       |
| MEM_LIMIT      | bigint(20)  | NO   |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
```

```
mysql> select * from gv$memstore;
+-----------+---------------+------+--------+-------+----------------+------------+
| TENANT_ID | IP            | PORT | ACTIVE | TOTAL | FREEZE_TRIGGER | MEM_LIMIT  |
+-----------+---------------+------+--------+-------+----------------+------------+
|      1152 | 100.81.140.78 | 2882 |      0 | 40960 |     5862630340 | 8375186220 |
|      1152 | 100.81.140.80 | 2882 |      0 | 40960 |     5862630340 | 8375186220 |
|      1152 | 100.81.140.82 | 2882 |      0 | 40960 |     5862630340 | 8375186220 |
+-----------+---------------+------+--------+-------+----------------+------------+
3 rows in set (0.59 sec)
```

8.4.2 实时监控：磁盘空间

- select svr_ip, total_size/1024/1024/1024 total_G, free_size/1024/1024/1024 free_G, (total_size - free_size) /1024/1024/1024 used_G from __all_virtual_disk_stat;
- free_G 一般应 > 800（根据实际机器配置会有区别），如果所有server都小于此值，说明集群存储空间不够，应考虑集群扩容。其它情况应检查租户空间

OCEANBASE

8.4.2 实时监控：检查租户分区表情况

- __all_virtual_meta_table 记录了副本信息，可按租户，表统计磁盘空间使用
- select tenant_id, svr_ip, unit_id, table_id, sum(data_size) /1024/1024/1024 size_G from __all_virtual_meta_table group by 1, 2, 3, 4;
- 如果租户某unit磁盘空间占用过大(比如>4TB)应考虑增加租户unit
- 如果单表磁盘空间占用过大 (比如>200GB)，应考虑对表进行分区
- 只包含 SSTable磁盘空间，不含memTable内存中数据。

OCEANBASE

8.4.1 系统监控视图： gv$sql_audit (sys租户权限)

展示了observer所有执行的sql

```
+-----------------------+---------------------+------+-----+---------+-------+
| Field                 | Type                | Null | Key | Default | Extra |
+-----------------------+---------------------+------+-----+---------+-------+
| SVR_IP                | varchar(32)         | NO   |     | NULL    |       |
| SVR_PORT              | bigint(20)          | NO   |     | NULL    |       |
| REQUEST_ID            | bigint(20)          | NO   |     | NULL    |       |
| TRACE_ID              | varchar(128)        | NO   |     | NULL    |       |
| CLIENT_IP             | varchar(32)         | NO   |     | NULL    |       |
| CLIENT_PORT           | bigint(20)          | NO   |     | NULL    |       |
| TENANT_ID             | bigint(20)          | NO   |     | NULL    |       |
| TENANT_NAME           | varchar(64)         | NO   |     | NULL    |       |
| USER_ID               | bigint(20)          | NO   |     | NULL    |       |
| USER_NAME             | varchar(64)         | NO   |     | NULL    |       |
| SQL_ID                | varchar(32)         | NO   |     | NULL    |       |
| QUERY_SQL             | varchar(32768)      | NO   |     | NULL    |       |
| AFFECTED_ROWS         | bigint(20)          | NO   |     | NULL    |       |
| RETURN_ROWS           | bigint(20)          | NO   |     | NULL    |       |
| RET_CODE              | bigint(20)          | NO   |     | NULL    |       |
| EVENT                 | varchar(64)         | NO   |     | NULL    |       |
| P1TEXT                | varchar(64)         | NO   |     | NULL    |       |
| P1                    | bigint(20) unsigned | NO   |     | NULL    |       |
| P2TEXT                | varchar(64)         | NO   |     | NULL    |       |
| P2                    | bigint(20) unsigned | NO   |     | NULL    |       |
| P3TEXT                | varchar(64)         | NO   |     | NULL    |       |
| P3                    | bigint(20) unsigned | NO   |     | NULL    |       |
| LEVEL                 | bigint(20)          | NO   |     | NULL    |       |
| WAIT_CLASS_ID         | bigint(20)          | NO   |     | NULL    |       |
| WAIT_CLASS#           | bigint(20)          | NO   |     | NULL    |       |
| WAIT_CLASS            | varchar(64)         | NO   |     | NULL    |       |
| STATE                 | varchar(19)         | NO   |     | NULL    |       |
| WAIT_TIME_MICRO       | decimal(38,3)       | NO   |     | NULL    |       |
| TOTAL_WAIT_TIME_MICRO | decimal(38,3)       | NO   |     | NULL    |       |
| TOTAL_WAITS           | bigint(20)          | NO   |     | NULL    |       |
| RPC_COUNT             | bigint(20)          | NO   |     | NULL    |       |
| PLAN_TYPE             | bigint(20)          | NO   |     | NULL    |       |
| IS_INNER_SQL          | tinyint(4)          | NO   |     | NULL    |       |
| IS_EXECUTOR_RPC       | tinyint(4)          | NO   |     | NULL    |       |
| IS_HIT_PLAN           | tinyint(4)          | NO   |     | NULL    |       |
| REQUEST_TIME          | bigint(20)          | NO   |     | NULL    |       |
| ELAPSED_TIME          | bigint(20)          | NO   |     | NULL    |       |
| NET_TIME              | bigint(20)          | NO   |     | NULL    |       |
| NET_WAIT_TIME         | bigint(20)          | NO   |     | NULL    |       |
| QUEUE_TIME            | bigint(20)          | NO   |     | NULL    |       |
| DECODE_TIME           | bigint(20)          | NO   |     | NULL    |       |
| GET_PLAN_TIME         | bigint(20)          | NO   |     | NULL    |       |
| EXECUTE_TIME          | bigint(20)          | NO   |     | NULL    |       |
| APPLICATION_WAIT_TIME | bigint(20) unsigned | NO   |     | NULL    |       |
| CONCURRENCY_WAIT_TIME | bigint(20) unsigned | NO   |     | NULL    |       |
| USER_IO_WAIT_TIME     | bigint(20) unsigned | NO   |     | NULL    |       |
| SCHEDULE_TIME         | bigint(20) unsigned | NO   |     | NULL    |       |
| ROW_CACHE_HIT         | bigint(20)          | NO   |     | NULL    |       |
| BLOOM_FILTER_CACHE_HIT | bigint(20)         | NO   |     | NULL    |       |
| BLOCK_CACHE_HIT       | bigint(20)          | NO   |     | NULL    |       |
| BLOCK_INDEX_CACHE_HIT | bigint(20)          | NO   |     | NULL    |       |
| DISK_READS            | bigint(20)          | NO   |     | NULL    |       |
| RETRY_CNT             | bigint(20)          | NO   |     | NULL    |       |
| TABLE_SCAN            | tinyint(4)          | NO   |     | NULL    |       |
| CONSISTENCY_LEVEL     | bigint(20)          | NO   |     | NULL    |       |
| MEMSTORE_READ_ROW_COUNT | bigint(20)        | NO   |     | NULL    |       |
| SSSTORE_READ_ROW_COUNT | bigint(20)         | NO   |     | NULL    |       |
+-----------------------+---------------------+------+-----+---------+-------+
```

用来监测不同sql语句的执行效率

另一个视图，称为SQL审计，它直接展现每条SQL每次执行的明细，SQL审计同时满足安全合规的要求。在SQL审计的明细中，除了SQL文本、SQL执行信息、客户端地址、租户名、用户名等审计需要的信息之外，还包括前面我们在SQL统计中提到的一些资源消耗，包括RT、逻辑读、物理读、各类等待事件（Lock、Latch、IO）的等待时间之和，特别地，还包括了本次SQL执行中最长一次等待的等待明细（Max Wait）。

SQL审计主要用于慢查询分析，并构造系统中所有SQL按照RT展现的直方图（比如当前集群中，每分钟RT在100~500ms之间的SQL有多少个，长什么样），和SQL统计不同，后者主要反映一条SQL的平均执行情况，而前者主要用于尖刺分析。前面在介绍等待事件的时候，我们已经介绍过这种场景，某条SQL的平均执行rt小于1ms，但是偶尔会出现rt > 100ms的情况，通过记录这种尖刺的执行明细，结合Max Wait及其它资源消耗信息，能够很清楚地展示出慢查询的卡点，并找出系统潜在的问题。

尽量了解sql audit中记录的每一个字段的含义，比如

retry_cnt: 太多基本上就是锁冲突

queue time: 很高表明CPU资源不够用

8.4.1 系统监控视图：gv$sql_audit(sys租户权限)

- 检查特定租户下Top 10的sql执行时间
  - select sql_id, query_sql,count(*), avg(elapsed_time), avg(execute_time), avg(queue_time), avg(user_io_wait_time) from gv$sql_audit where tenant_id=1002 group by sql_id having count(*)>1 order by 5 desc limit 10\G;
- 检查特定租户下消耗cpu最多的top sql
  - select sql_id, avg(execute_time) avg_exec_time, count(*) cnt, avg(execute_time-TOTAL_WAIT_TIME_MICRO) cpu_time from gv$sql_audit where tenant_id=1002 group by 1 order by avg_exec_time * cnt desc limit 5;

OCEANBASE

8.4.1 系统监控视图：gv$sql

记录所有租户在所有 ObServer 上的 SQL 相关统计信息，每个 plan 都会在表中有一行数据

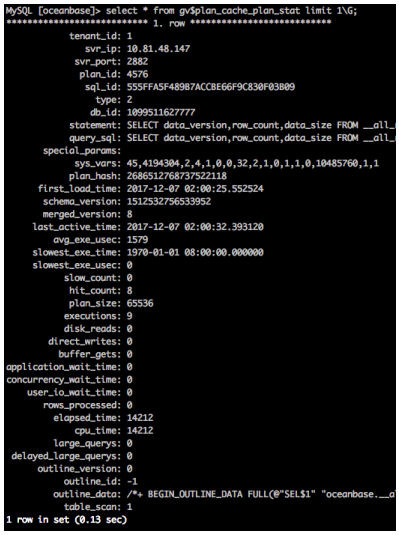
CON_ID 租户 ID
PLAN_ID 执行计划 ID
SQL_ID SQL 的标识符
TYPE 计划的类型，local，remote 或 distribute
SQL_TEXT SQL 语句内容
PLAN_HASH_VALUE 执行计划的 hash 值
FIRST_LOAD_TIME 第一次执行的开始时间
LAST_ACTIVE_TIME 最近一次执行的开始时间
AVG_EXE_USEC 平均执行耗时
SLOWEST_EXE_TIME 最慢一次执行的开始时间
SLOWEST_EXE_USEC 最慢一次执行的耗时
SLOW_COUNT 慢查询次数统计
HIT_COUNT 命中 plan cache 的次数
MEM_USED 内存空间使用大小
EXECUTIONS 执行次数
DISK_READS 读盘次数
DIRECT_WRITES 写盘次数
BUFFER_GETS 逻辑读次数
APPLICATION_WAIT_TIME application 类事件等待时间
CONCURRENCY_WAIT_TIME concurrentcy 类事件等待时间
USER_IO_WAIT_TIME 所有 IO 类事件等待时间
ROWS_PROCESSED 返回的行数
ELAPSED_TIME 接收至处理完成总消耗时间
CPU_TIME 消耗的 cpu 时间

OCEANBASE

8.4.1 系统监控视图：gv$plan_cache_plan_stat

- 普通租户权限登陆
- 记录所有 ObServer 上当前租户缓存计划的统计信息，可以通过该表找到Top-SQL
- 每日合并（Merge）会触发计划淘汰，这个表会被更新

OCEANBASE



常用sql：select * from oceanbase.gv$plan_cache_plan_stat where executions > 10 order by avg_exe_usec desc limit 1\G;

相比于**gv$plan_cache_stat**，有详细的sql语句信息，便于dba排查问题，**gv$plan_cache_stat**这个表更多是统计信息，用于排查整体资源问题（比如内存不足）

Oceanbase会在ObServer中为每个租户维护一个执行计划缓存(plan cache)，__all_virtual_plan_cache_stat记录所有ObServer上租户的plan cache的统计信息。

在启用plan cache的情况下，Oceanbase会将sql进行参数化，即每一类sql对应一个plan cache项，比如 select * from ta where a = 1 和 select * from ta where a = 5虽然是不同的sql，但参数化以后的形式是一样的，均为select * from ta where a = ?，类似形式的sql都会命中该计划。但参数化结果相同的sql可能对应不同的执行计划（计划类型可能不同）。

first_load_time 第一次加载时间
schema_version schema 版本
merged_version 当前缓存的 plan 对应的合并版本号

last_active_time 上一次被执行的时间
avg_exe_usec 平均执行时间
slowest_exe_time 最慢一次执行的时间戳
slowest_exe_usec 最慢一次执行耗时
slow_count 慢查询的次数
hit_count 命中次数
mem_used 内存使用量
executions 执行次数
disk_reads 物理读次数
direct_writes 物理写次数
buffer_gets 逻辑读次数
application_wait_time 所有 application 类事件的总时间
concurrency_wait_time 所有 concurrency 类事件的总时间
user_io_wait_time 所有 user_io 类事件的总时间
rows_processed 返回的行数
elapsed_time 接收到请求到执行结束消耗时间
cpu_time cpu耗时
large_querys 大查询的数量
delayed_large_querys 被延迟调度的大查询的数量
outline_version outline的版本
outline_id outline ID

8.4.1 系统监控视图：gv$plan_cache_plan_explain

- 记录当前租户在所有ObServer上缓存的计划的 explain 信息（具体的计划信息）
- 需要提供四元组(tenant_id, ip, port, plan_id)

```
mysql> select * from oceanbase.gv$plan_cache_plan_explain where tenant_id = 1061 and ip = "11.232.2.27" and port = 2882 and plan_id = 121005;
+-----------+-------------+------+---------+------------------+------------------------------------------------------------+------+------+---------------------------------------------------------------------------------------------------------------
+
| TENANT_ID | IP          | PORT | PLAN_ID | OPERATOR         | NAME                                                       | ROWS | COST | PROPERTY
                                                                                                                            |
+-----------+-------------+------+---------+------------------+------------------------------------------------------------+------+------+---------------------------------------------------------------------------------------------------------------
+
|      1061 | 11.232.2.27 | 2882 |  121005 | PHY_ROOT_TRANSMIT | NULL                                                      |    0 |    0 | NULL
                                                                                                                            |
|      1061 | 11.232.2.27 | 2882 |  121005 |   PHY_TABLE_SCAN  | business_activity_10(business_activity_gmt_ind) |    5 |   83 | table_rows:5, access_rows:5, est_method:storage, opt_metho
=cost_based, avl_name:business_activity_gmt_ind, pruned_name:business_activity_10,idx_oracle_flag |
+-----------+-------------+------+---------+------------------+------------------------------------------------------------+------+------+---------------------------------------------------------------------------------------------------------------
+
2 rows in set (0.01 sec)
```

OCEANBASE

8.4.2 实时监控：集群级事件 - 查看root service操作记录

- __all_rootservice_event_history：记录集群级事件，如major freeze, 合并，server 上下线，负载均衡任务执行等，保留最近7天历史
  - ✓ select * from __all_rootservice_event_history order by gmt_create desc limit 10;
- 如发现一台server宕机，要确认OceanBase探测到此server宕机时间点，可执以下SQL:
  - ✓ select * from __all_rootservice_event_history where module = 'server';

OCEANBASE

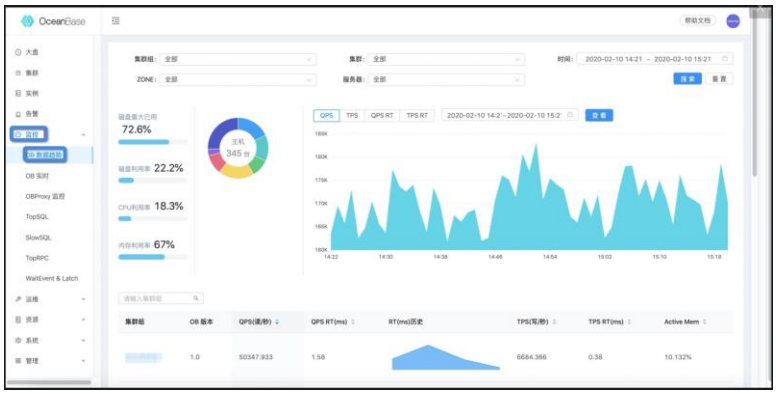
8.4.2 实时监控：server级事件

- __all_server_event_history：记录server级事件，如转储，合并完成，切主，执行用户命令等，保留最近3天历史
- 查看转储次数：select * from __all_server_event_history where module like '%minor%' order by gmt_create desc limit 10;

OCEANBASE

8.4.3 性能监控：常规监测

性能问题应优先通过OCP 管理员入口 ==> 集群入口 ==>性能 监控==> 数据趋势中查看QPS_RT, TPS_RT，大致定位出问题时间点



OCEANBASE

8.4.3 性能监控：捞取慢SQL

定位方法 1：

- OceanBase中执行时间超过 trace_log_slow_query_watermark (系统参数)的sql，在 observer 日志中都会打slow query消息。
- 在 observer 日志中查找慢 SQL 消息：

cd /home/admin/oceanbase/log

fgrep ‘[slow query]’ observer.log*

OCEANBASE

8.4.3 性能监控：捞取慢SQL

慢 SQL 实例：

total_timeu表示sql执行总时间(单位：微秒)

stmt记录了用户执行的sql语句

```
[2017-08-11 10:02:07.379969] TRACE [TRACE]:0 [37703][YB420A376759-28D639DF0] [lt=16] [slow query](TRACE=begin_ts=1502416926006425 2017-08-11 10:02:06.006425|[process_begin] u=0 in_queue_time:8, receive_ts:1502416926006416, enqueue_ts:1502416926006417|[start_sql] u=0 addr:{ip:"11.233.88.94", port:48077}|[query_begin] u=1 |[before_processor_run] u=1 |[session] u=3 sid:2148411417, tenant_id:1031|[parse_begin] u=1 stmt:"select ... from ... where ...", stmt_len:626|[cache_get_plan_begin] u=7 |[pc_choose_plan] u=42 |[check_priv] u=2 |[plan_id] u=1 plan_id:80027|[exec_begin] u=1 arg1:false, end_trans_cb:false|[start_trans] u=4 trans_id:{hash:7199905523483370054, inc:921851582, addr:{ip:"10.55.103.89", port:2882}, t:1502416926006356}, timeout:1502417026006416, start_time:1502416926006416|[do_open_plan_begin] u=1 |[sql_start_stmt_begin] u=0 |[sql_start_stmt_end] u=6 |[exec_plan_begin] u=1 |[exec_plan_end] u=0 |[sql_start_participant_begin] u=0 |[start_part] u=24 trans_id:{hash:7199905523483370054, inc:921851582, addr:{ip:"10.55.103.89", port:2882}, t:1502416926006356}|[sql_start_participant_end] u=0 |[storage_table_scan_begin] u=14 |[storage_table_scan_end] u=10 |[do_open_plan_end] u=7 |[found_rows] u=1373169 total_count:0, input_count:0|[close_plan_begin] u=21 |[revert_scan_iter] u=35 |[end_participant_begin] u=6 |[end_participant_end] u=5 |[start_end_stmt] u=0 |[end_stmt] u=1 |[close_plan_end] u=0 |[affected_rows] u=0 affected_rows:-1|[store_found_rows] u=1 found_rows:0, return_rows:0|[auto_end_plan_begin] u=1 |[auto_end_plan_end] u=74 |[result_set_close] u=0 ret:0, arg1:0, arg2:0, arg3:-4008, async:false|[exec_end] u=4 |[query_end] u=33 |[process_end] u=7 run_ts:1502416926006428|total_timeu=1373493)
```

OCEANBASE

8.4.3 性能监控：捞取慢SQL

- OceanBase提供两张虚拟表 vsql_audit , gvsql_audit 记录最近一段时间sql执行历史
- v$sql_audit 存储本机的sql执行历史， gv$sql_audit 存储整个集群的sql执行历史
- 查询 v$sql_audit 表，如查询某租户执行时间大于1s (1000000微秒)的SQL：
  - select * from v$sql_audit where tenant_id = <tenant id> and elapsed_time > 1000000 limit 10
- 查询SQL执行时间按秒分布的直方图
  - select round(elapsed_time/1000000), count(*) from v$sql_audit where tenant_id = <tenant_id> group by 1;

OCEANBASE

8.4.3 修复慢SQL

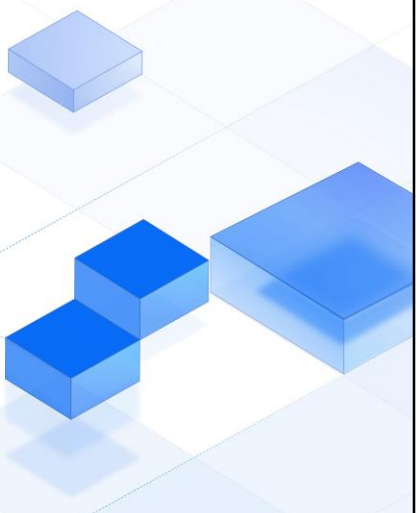
- 创建索引：当慢SQL因无合适索引可用时导致时，可创建索引
- outline绑定：如慢SQL由OceanBase优化器选择了不够优的执行计划导致，可通过outline绑定执行计划(具体hint语法和outline绑定语法请参考OceanBase官网文档) 。
- Outline绑定实例：将 select * from t1 where v1 = 3 这条SELECT绑定走主键索引

以业务租户身份登录到OceanBase （注意 mysql 命令中要加上 -c 参数)后执行以下命令：

```
create outline bind_to_primary_key on select/*+ index(t1 primary)*/ * f rom t1 where v1 = 3;
```

OCEANBASE

8.5 常见异常处理



OCEANBASE

8.5.1 集群问题排查解决：集群故障

- 通过OCP或者内部表（__all_server）查看是否有节点处于inactive状态
- 应急操作：
  - 重启OBServer进程
  - 如果是硬件因素导致，先修复硬件故障（包括网络）

OCEANBASE

8.5.1 集群问题排查解决：机器硬件故障

- 建议“alter system stop zone” 将observer 上的 leader 服务切走，减少对业务的影响。
- 如果故障集群正在合并，可以快速暂停掉，确定影响后再决策是否打开
- 修改primary zone，如果stop zone不能解决，可以尝试用这个办法

OCEANBASE

暂停合并的命令：alter system suspend merge

8.5.1 集群问题排查解决：合并较慢

合并和转储较慢：

- 有些压缩算法会比其它压缩算法慢，如zstd要比lz4慢。
- 轮转合并被打开
- 合并线程数太少

OCEANBASE

有时还会有一些极端情况，比如某些zone的某些机器出现问题，阻塞了合并（不过这个在线上极少出现，因为机器在出现问题的时候已经会有报警了），比如线下，尤其是poc这种过程中，因为资源有限，会出现一些问题，可以通过系统表和observer的log进行排查

8.5.1 集群问题排查解决：内部表 / 系统参数不可用

- 检查NTP服务
- 检查时钟是否同步 (比如 ntpq –p, clockdiff)
- 时钟同步的安全范围在100ms以内，超过100ms则会给集群带来各种问题：选主失败、partition无主等

OCEANBASE

时钟跳变会导致OB无主，可以通过内核log文件/var/log/messages中的关键字 ntpd[2425]: 0.0.0.0 0613 03 spike_detect -0.368888 s 安全值是150ms以内，超过这个值就有风险。

8.5.1 Unit/副本迁移失败

– 检查动态负载均衡是否开启：show parameters like 'enable_rebalance';

如果需要副本自动迁移（比如delete server），该配置项必须为true。

– 检查resource_soft_limit参数的值是否小于100： show parameters like 'resource_soft_limit’ ;

如果值>=100，默认不会往新添加的机器上做Unit迁移和均衡。

OCEANBASE

8.5.1 集群问题排查解决：负载均衡 - 迁移复制

内部表:

- __all_virtual_rebalance_task_stat：迁移复制任务
- __all_virtual_partition_migration_status：observer端正在执行的对象迁移任务

配置项:

enable_rereplication 复制开关

enable_rebalance 迁移开关

resource_soft_limit Unit自动均衡至空闲资源的开关

data_copy_concurrency 任务并发

server_data_copy_in_concurrency 单机拷入并发

server_data_copy_out_concurrency 单机拷出并发

sys_bkgd_net_percentage 网络带宽限制

sys_bkgd_io_low_percentage io限制

sys_bkgd_io_high_percentage io限制

OCEANBASE

8.5.2 业务问题排查解决：远程/异地执行

- gv$sql_audit中的plan_type字段：1 – 本地执行；2 – 远端执行；3 – 分布式执行
- obproxy是轻量级的sql parser；过于复杂的WHERE条件，可能会导致proxy路由不准。
- JDBC在执行sql前可能会发一些 set auto_commit/select @tx_read_only 之类的请求，obproxy会随机选择事务的协调者，可能造成异地执行。
- 在OCP 数据趋势中选中server后看SQL_COUNT中SQL_REMOTE_COUNT指标

OCEANBASE


8.6 灾难恢复
OCEANBASE

8.6 灾难恢复

灾难恢复是指当数据库中的数据在被有意或无意破坏后复原数据库所需要执行的活动。

- **回收站：** 用来存储被删掉的对象。被放入回收站的对象数据都被保存，仍然占据着物理空间，除非手动进行清除（ PURGE RECYCLEBIN）。
- **闪回查询：**允许用户可以获取某个历史版本的数据。

OCEANBASE

8.6 回收站

➢ **回收站支持的对象**

如下表所示可以进入回收站的对象有索引、表和库。被删除的租户是无法进入回收站的.

模式	索引（Index）	表（Table）	数据库（Database）	租户（Tenant）
MySQL	✓	✓	✓	✗
Oracle	✗	✓	✗	✗

➢ **查看回收站**

show recyclebin;

➢ **开关回收站**

租户创建之后，默认是开启回收站的，此时对数据库对象进行 Truncate / Drop 操作后，对象会进入到回收站

- 租户级别的开启关闭语句：set global `recyclebin` = on /off;
- Session 级别的开启关闭语句：set @@recyclebin = on/off

OCEANBASE

8.6 回收站

➢ 恢复回收站数据

使用 FLASHBACK 命令可恢复回收站中的数据库和表对象，只有租户的管理员用户才可以使用该命令：

- 恢复对象数据库

```
FLASHBACK DATABASE <object_name> TO BEFORE DROP [RENAME TO database_name];
```

- 恢复对象表

```
FLASHBACK TABLE <object_name> TO BEFORE DROP [RENAME to table_name];
```

➢ 回收站清理

回收站中的数据可以通过 PURGE 命令清理；当一个对象的上层对象被 PURGE，那么当前回收站中关联的下一层对象也会被 PURGE

- 指定库物理删除： PURGE DATABASE <object_name>;
- 指定表物理删除： PURGE TABLE <object_name>;
- 指定索引表物理删除： PURGE INDEX <object_name>;
- 清空整个回收站： PURGE RECYCLEBIN;

OCEANBASE

8.6 闪回查询

闪回查询（Flashback Query）是 Oracle 中记录级别的闪回功能。该功能允许用户获取某个历史版本的数据。OB同时支持 MySQL 和 Oracle 两种模式下的查询，具体细节可以参考官网文档。

闪回查询支持 SCN（time_to_usec()）和 TIMESTAMP 两种维度的查询（usec_to_time()）。

示例 1：通过 TIMESTAMP 指定的历史时间并闪回查询一张单表在该历史时间中的状态的数据：

```
select * from tbl1 as of timestamp TO_TIMESTAMP('2020-08-13 16:20:00','yyyy-mm-dd hh24:mi:ss');
```

示例 2：通过 TIMESTAMP 指定的历史时间并闪回查询多表在该历史时间中的状态的数据：

```
select * from tbl1 as of timestamp TO_TIMESTAMP('2020-08-13 16:20:00','yyyy-mm-dd hh24:mi:ss'),tbl2 as
of timestamp TO_TIMESTAMP('2020-08-13 16:20:00','yyyy-mm-dd hh24:mi:ss');
```

示例 3：通过 SCN 指定历史时间并闪回查询单表在该历史时间点的状态的数据：

```
select * from tbl1 as of scn 1582807800000000;
```

OCEANBASE


感谢学习
OCEANBASE