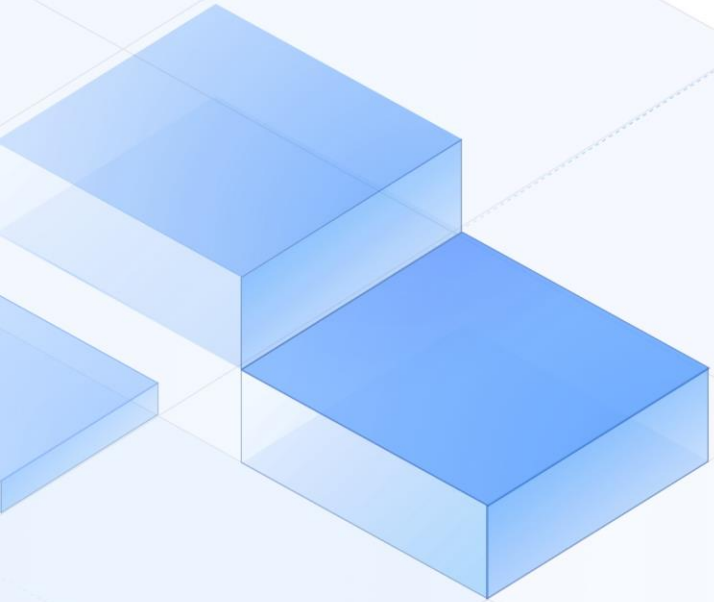
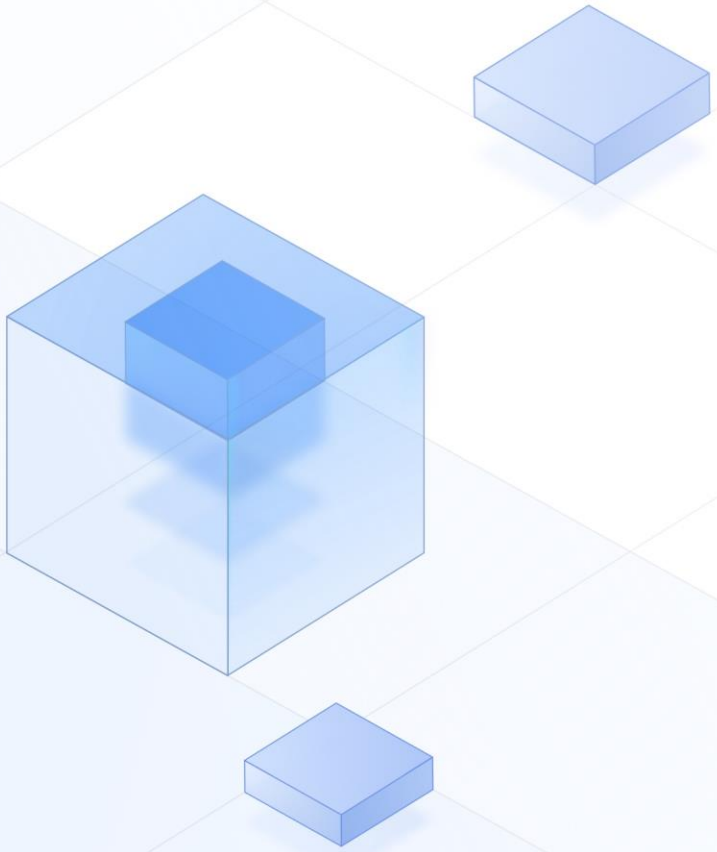
OCEANBASE

OBProxy 路由策略与使用运维

目录



目录

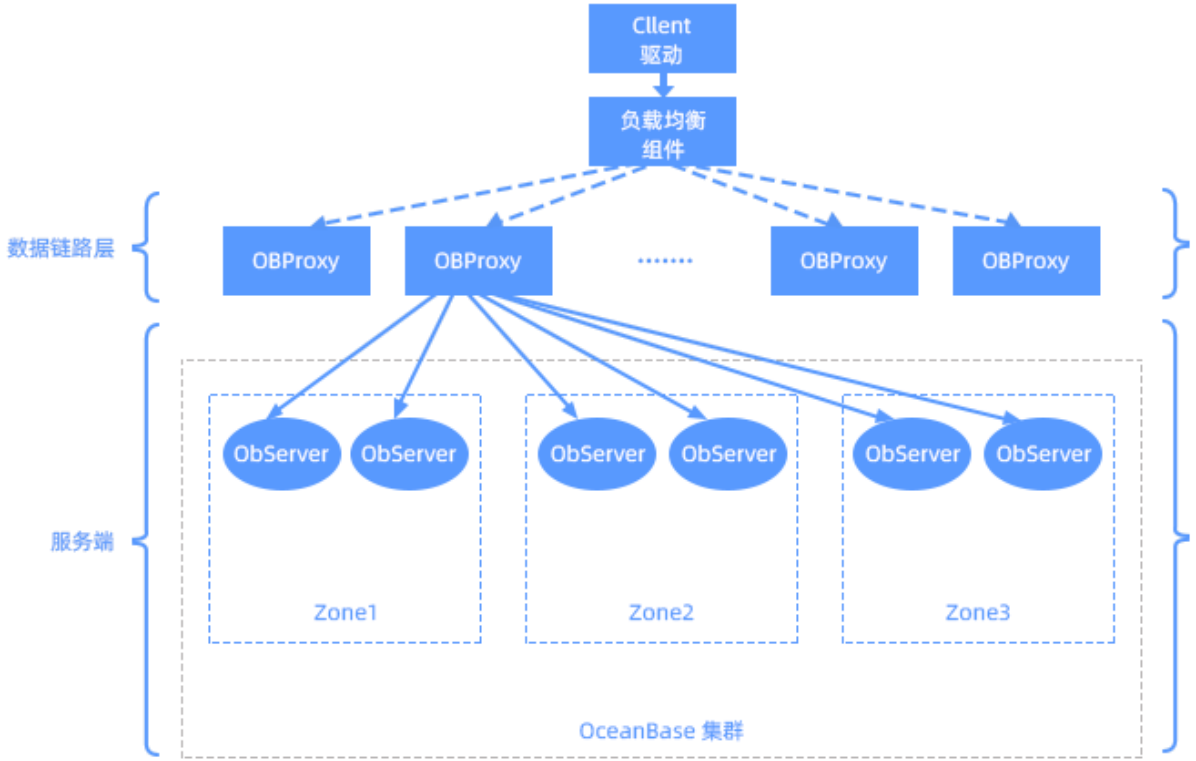


第六章 / OBProxy 路由策略与使用运维

6.1 OBProxy 简介

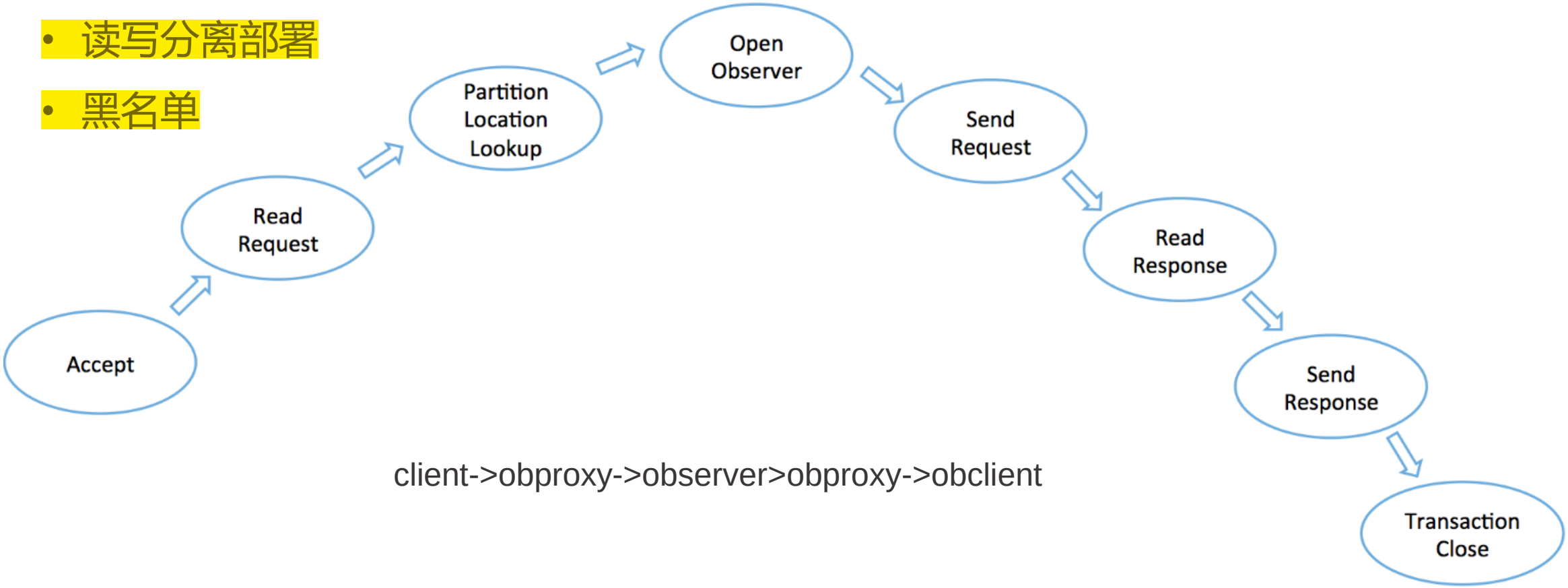
6.1 背景

1. OceanBase在部署完成后，应用可以采用OB提供的客户端、 MySQL客户端或者其他语言的客户端来访问OceanBase，OceanBase以服务的形式提供给应用访问。 OceanBase 集群一般包含多个 Zone ，每个 Zone 中又包含一台或多台 OBServer ，集群中的每个 OBServer 都可以接收用户的连接并处理请求。
2. 实现一个OBProxy来接受来自应用的请求，并转发给OBServer，然后OBServer将数据返回给OBProxy，OBProxy将数据转发给应用客户端



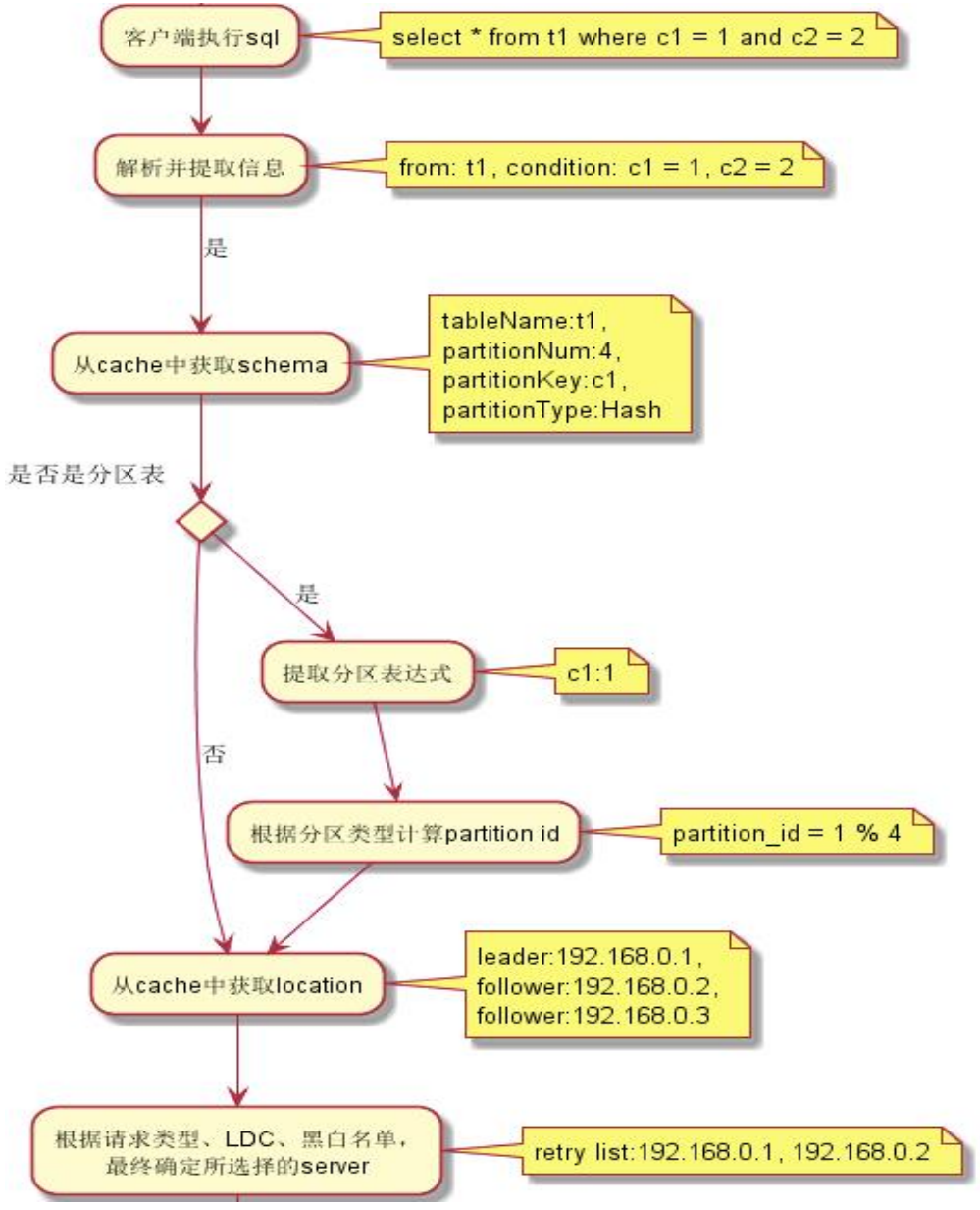
6.1 OBProxy核心功能（路由）

- 简单SqlParser
- LDC路由
- 读写分离部署
- 黑名单



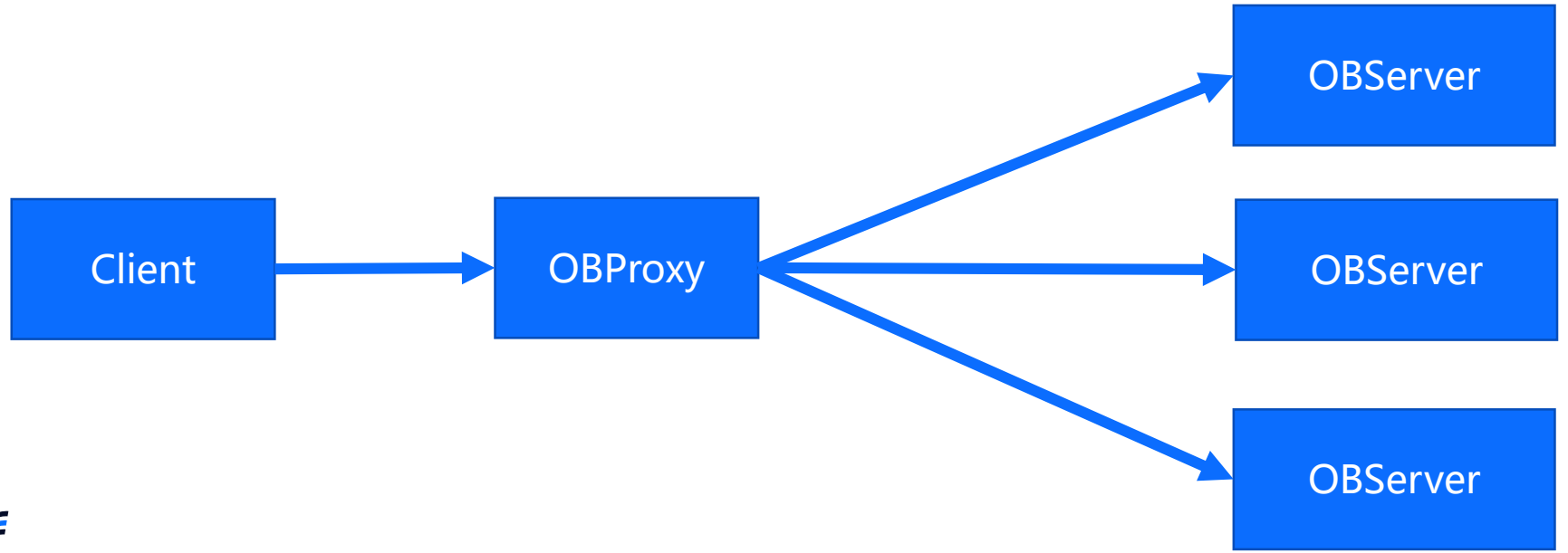
client->obproxy->observer>obproxy->obclient

6.1 OBProxy核心功能（路由）



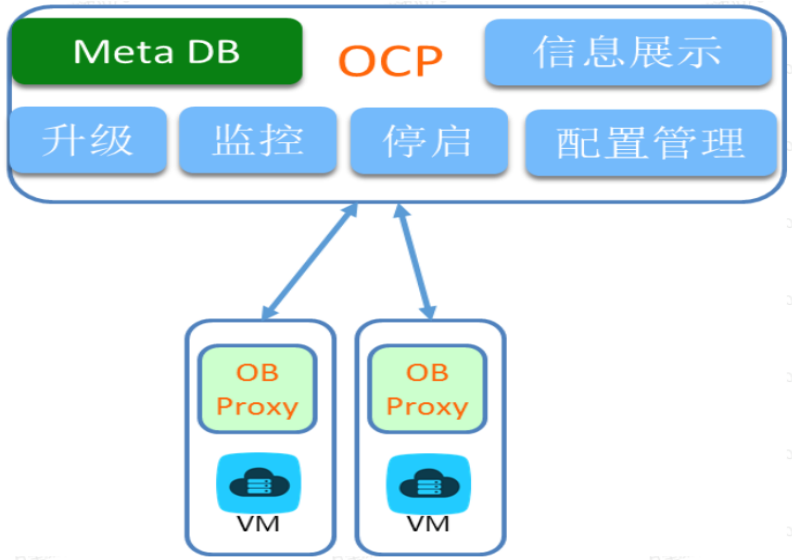
6.1 OBProxy核心功能（连接管理）

1. 在observer宕机/升级/重启时，客户端与OBProxy的连接不会断开，OBProxy可以迅速切换到正常的server上，对应用透明
2. OBProxy支持用户通过同一个OBProxy访问多个OceanBase集群
3. Server session对于每个client session独占
4. 同一个client session对应server session状态保持相同(session变量同步)

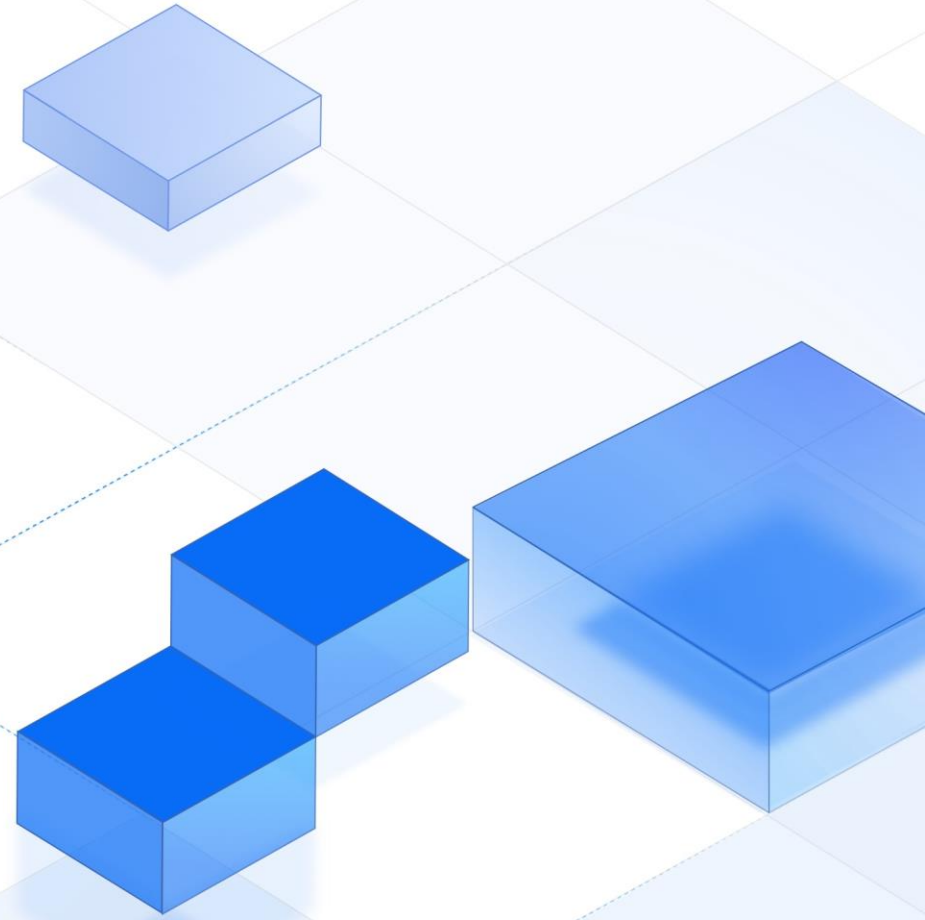


6.1 ObProxy核心功能(监控 && 运维)

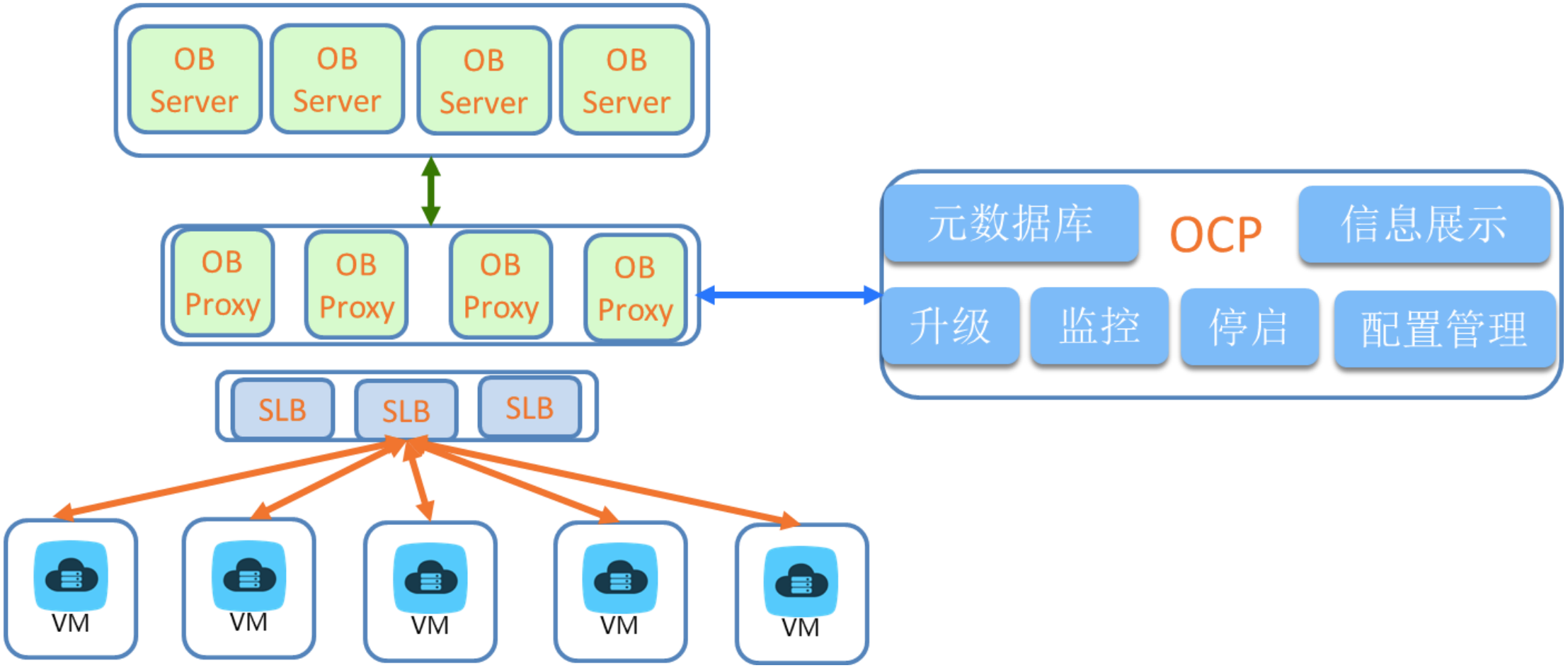
- 周期性汇报统计项到OCP，实现了语句级别，事务级别，session级别，Obproxy级别的各种统计。
- Xflush日志监控。
- Sql Audit功能。
- 实现了大量内部命令来实现远程监控，查询和运维。



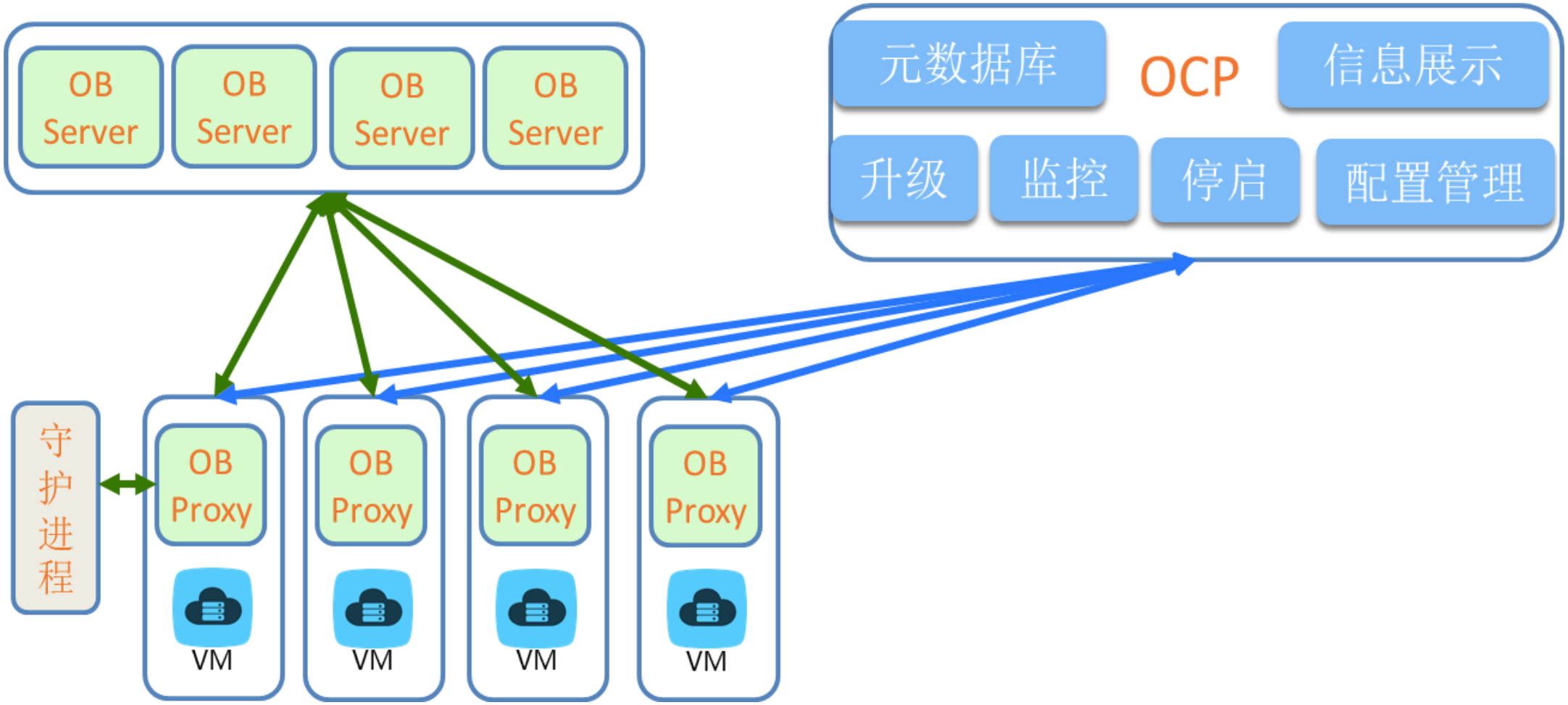
6.2 OBProxy 部署



6.2 部署方式 – 集中部署


OB Server
OB Server
OB Server
OB Server
OB Proxy
OB Proxy
OB Proxy
OB Proxy
SLB
SLB
SLB
VM
VM
VM
VM
VM
元数据库
OCP
信息展示
升级
监控
停启
配置管理

6.2 部署方式 – 客户端部署



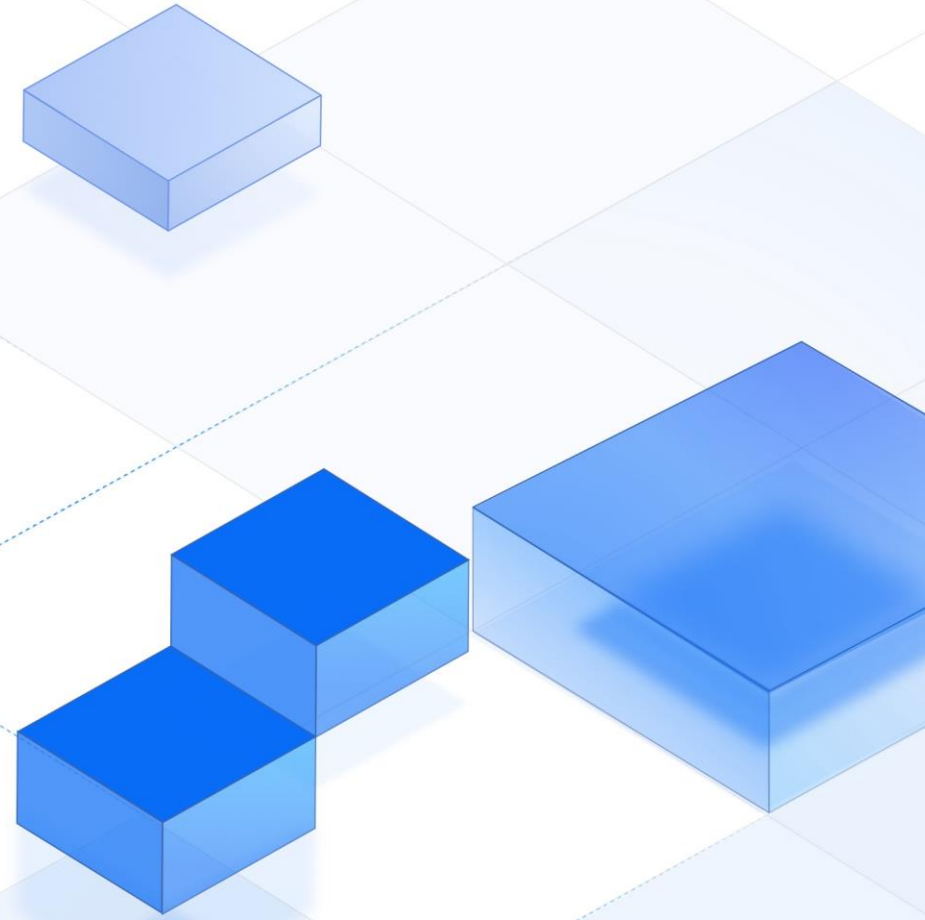
6.2 两种启动模式

- 测试模式 主要用于现阶段开发调试，无需依赖 ConfigServer
  - ConfigServer是OCP平台提供的OB集群物理入口管理服务，是一个web api的服务
  - 测试模式通过指定集群的RSList（ip列表）来启动
- 生产模式
  - ObProxy可以通过指定 config server 提供的 config_url 来启动，config server服务可以协助获取该集群的配置信息。同一个config server可以保存多个OB集群的RSList信息，使obproxy能为多个OB集群同时提供服务。
  - 在连接ObProxy时，其用户名类似 root@sys#cluster，其中 root 为用户名，sys 为租户名，cluster 为集群名

6.2 OBProxy运行环境

- OS: Linux Redhat 7u x86-64 及以上
- OS内核：3.10 及以上版本
- CPU: 2 核及以上
- 内存：1G 及以上
- 磁盘空间：对磁盘大小没有特别需求，推荐 10G 及以上，主要用于存放OBProxy的应用日志

6.3 路由策略



6.3 OBProxy执行流程

收到用户请求后，OBProxy的执行流程：

1. parse sql，通过简易parse，解析出sql涉及的table name、database name
2. fetch route entry，根据用户的tenant name、database name、table name、partition id 向observer拉取该partition的路由表
3. sort route entry，根据各种相关属性对路由表中ip进行排序
    - read_consisitency：强一致性读or弱一致性读
    - 目标server状态： 正在合并or常态
    - 路由精准度： PS or TS
    - LDC 匹配： 本地、同城、异地
    - Zone 类型
    - 读写分离的ob_route_policy取值
4. filter by congestion，从路由表中依次尝试目标ip，通过黑名单进行过滤
5. forward request，将用户请求转发给目标server

6.3 和路由相关的一些基础概念

这些基础概念和OBproxy路由密切相关，根据不同的配置，Obproxy 进行综合的路由排序：

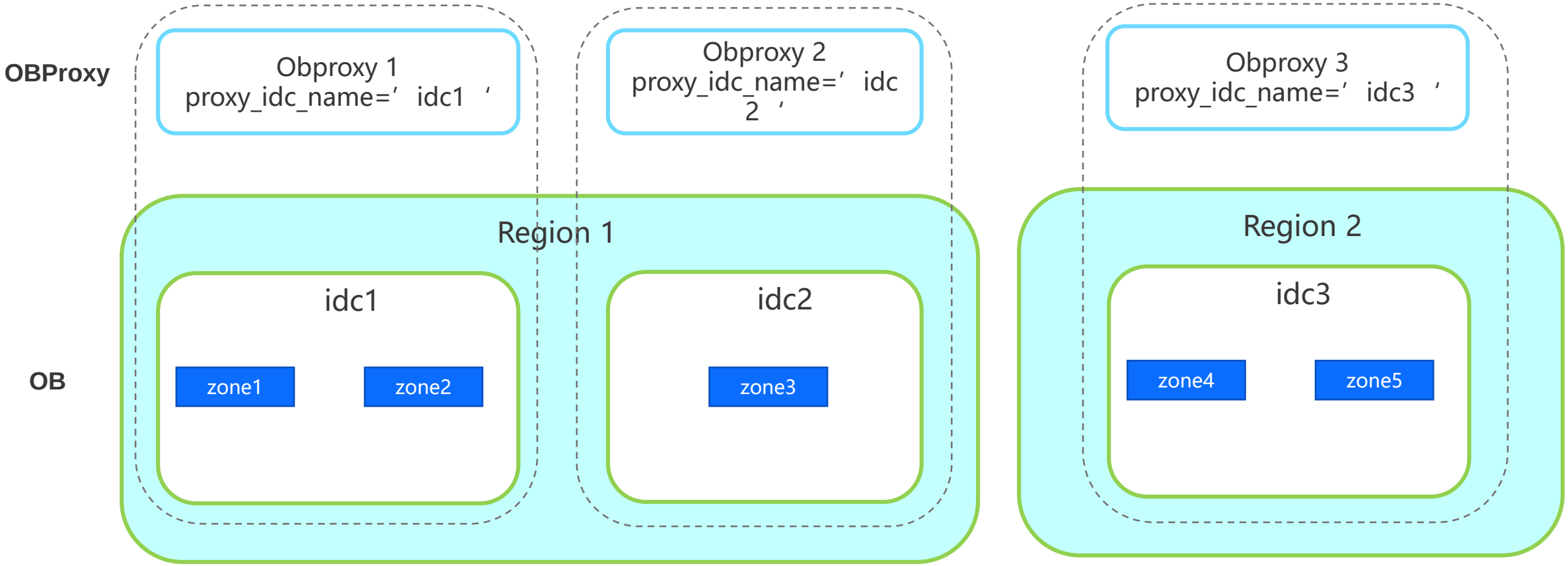
1. LDC配置：
    1. 本地： 同城同机房（IDC相同）
    2. 同城： 同城不同机房（IDC不同，Region 相同）
    3. 异地： 不同的地域（Region不同）
2. Observer 状态： 常态 vs 正在合并
3. 租户的Zone 类型： 读写型 vs 只读型
4. 路由精准度： 优先精准度高的

    OBproxy 中有目标 partition 的路由信息（PS）

    OBproxy 中没有Partition的路由信息，只有租户的路由信息（TS）

6.3 LDC

- IDC：互联网数据中心，可以简单看成一个物理机房
- LDC： logical data center，是对 IDC 的一种逻辑划分
- Region：地域信息，通常代表一个城市，包含一个或多个IDC，每个IDC部署一个或多个zone。一个OB集群可以包含若干个Region，每个Region包含若干个IDC，每个IDC部署若干个zone



6.3 LDC配置：集群的配置

设置sql：Observer 2881

1. alter system modify zone “z1” set region = “SHANGHAI";
2. alter system modify zone “z1” set idc = “zue”；

检查observer LDC设置内容是否生效：

select * from __all_zone;

```
mysql> select * from __all_zone;
+----------------------------+----------------------------+------+-------------------------+-----------------+----------------+
| gmt_create                 | gmt_modified               | zone | name                    | value           | info           |
+----------------------------+----------------------------+------+-------------------------+-----------------+----------------+
| 2017-12-07 10:13:43.556956 | 2017-12-07 10:13:43.556956 |      | cluster                 |               0 | ob1.jianhua.sjh |
| 2017-12-07 10:13:43.559956 | 2017-12-07 10:36:10.759389 |      | config_version          | 1512614170754402 |                |
| 2017-12-07 10:13:43.559199 | 2017-12-07 10:19:41.824562 |      | frozen_time             | 1512613181815870 |                |
| 2017-12-07 10:13:43.558963 | 2017-12-07 10:19:41.816285 |      | frozen_version          |               2 |                |
| 2017-12-07 10:13:43.559389 | 2017-12-07 10:19:41.829305 |      | global_broadcast_version |               2 |                |
| 2017-12-07 10:13:43.560805 | 2017-12-07 10:13:43.560805 |      | is_merge_error          |               0 |                |
| 2017-12-07 10:13:43.559587 | 2017-12-07 10:13:43.559587 |      | last_merged_version     |               1 |                |
| 2017-12-07 10:13:43.560404 | 2017-12-07 10:36:10.759650 |      | lease_info_version      | 1512614170759080 |                |
| 2017-12-07 10:13:43.561091 | 2017-12-07 10:19:41.829640 |      | merge_status            |               1 | MERGING        |
| 2017-12-07 10:13:43.559752 | 2017-12-07 10:13:43.559752 |      | privilege_version       |               0 |                |
| 2017-12-07 10:13:43.561590 | 2017-12-07 10:19:41.662268 |      | proposal_frozen_version |               2 |                |
| 2017-12-07 10:13:43.561758 | 2017-12-07 10:13:51.206103 |      | time_zone_info_version  | 1512612831205719 |                |
| 2017-12-07 10:13:43.558772 | 2017-12-07 10:19:41.665850 |      | try_frozen_version      |               2 |                |
| 2017-12-07 10:13:43.561340 | 2017-12-07 10:13:43.561340 |      | warm_up_start_time      |               0 |                |
| 2017-12-07 10:13:43.563078 | 2017-12-07 10:20:26.294279 | z1   | all_merged_version      |               2 |                |
| 2017-12-07 10:13:43.562442 | 2017-12-07 10:20:01.838148 | z1   | broadcast_version       |               2 |                |
| 2017-12-07 10:13:43.564065 | 2017-12-07 10:27:00.932976 | z1   | idc                     |               0 | zue            |
| 2017-12-07 10:13:43.563423 | 2017-12-07 10:20:26.294059 | z1   | is_merge_timeout        |               0 |                |
| 2017-12-07 10:13:43.562230 | 2017-12-07 10:20:26.293140 | z1   | is_merging              |               0 |                |
| 2017-12-07 10:13:43.562909 | 2017-12-07 10:20:26.293850 | z1   | last_merged_time        | 1512613226292603 |                |
| 2017-12-07 10:13:43.562703 | 2017-12-07 10:20:26.293625 | z1   | last_merged_version     |               2 |                |
| 2017-12-07 10:13:43.563277 | 2017-12-07 10:20:01.838395 | z1   | merge_start_time        | 1512613201837491 |                |
| 2017-12-07 10:13:43.563718 | 2017-12-07 10:20:26.294685 | z1   | merge_status            |               0 | IDLE           |
| 2017-12-07 10:13:43.563874 | 2017-12-07 10:35:15.459295 | z1   | region                  |               0 | SHANGHAI       |
| 2017-12-07 10:13:43.562012 | 2017-12-07 10:13:43.562012 | z1   | status                  |               2 | ACTIVE         |
| 2017-12-07 10:13:43.563568 | 2017-12-07 10:20:15.248527 | z1   | suspend_merging         |               0 |                |
| 2017-12-07 10:13:43.564211 | 2017-12-07 10:13:43.564211 | z1   | zone_type               |               0 | ReadWrite      |
| 2017-12-07 10:13:43.565129 | 2017-12-07 10:13:43.565129 | z2   | all_merged_version      |               1 |                |
| 2017-12-07 10:13:43.564662 | 2017-12-07 10:13:43.564662 | z2   | broadcast_version       |               1 |                |
| 2017-12-07 10:13:43.566052 | 2017-12-07 10:27:05.088372 | z2   | idc                     |               0 | zue            |
| 2017-12-07 10:13:43.565463 | 2017-12-07 10:13:43.565463 | z2   | is_merge_timeout        |               0 |                |
| 2017-12-07 10:13:43.564520 | 2017-12-07 10:13:43.564520 | z2   | is_merging              |               0 |                |
| 2017-12-07 10:13:43.564967 | 2017-12-07 10:13:43.564967 | z2   | last_merged_time        | 1512612823564340 |                |
| 2017-12-07 10:13:43.564803 | 2017-12-07 10:13:43.564803 | z2   | last_merged_version     |               1 |                |
| 2017-12-07 10:13:43.565302 | 2017-12-07 10:13:43.565302 | z2   | merge_start_time        | 1512612823564340 |                |
| 2017-12-07 10:13:43.565749 | 2017-12-07 10:13:43.565749 | z2   | merge_status            |               0 | IDLE           |
| 2017-12-07 10:13:43.565899 | 2017-12-07 10:35:19.539224 | z2   | region                  |               0 | SHANGHAI       |
| 2017-12-07 10:13:43.564365 | 2017-12-07 10:13:43.564365 | z2   | status                  |               2 | ACTIVE         |
| 2017-12-07 10:13:43.565604 | 2017-12-07 10:13:43.565604 | z2   | suspend_merging         |               0 |                |
| 2017-12-07 10:13:43.566199 | 2017-12-07 10:13:43.566199 | z2   | zone_type               |               0 | ReadWrite      |
| 2017-12-07 10:13:43.567102 | 2017-12-07 10:13:43.567102 | z3   | all_merged_version      |               1 |                |
| 2017-12-07 10:13:43.566650 | 2017-12-07 10:13:43.566650 | z3   | broadcast_version       |               1 |                |
| 2017-12-07 10:13:43.567961 | 2017-12-07 10:27:11.937454 | z3   | idc                     |               0 | ztg            |
| 2017-12-07 10:13:43.567393 | 2017-12-07 10:13:43.567393 | z3   | is_merge_timeout        |               0 |                |
| 2017-12-07 10:13:43.566497 | 2017-12-07 10:13:43.566497 | z3   | is_merging              |               0 |                |
| 2017-12-07 10:13:43.566946 | 2017-12-07 10:13:43.566946 | z3   | last_merged_time        | 1512612823566318 |                |
| 2017-12-07 10:13:43.566791 | 2017-12-07 10:13:43.566791 | z3   | last_merged_version     |               1 |                |
| 2017-12-07 10:13:43.567252 | 2017-12-07 10:13:43.567252 | z3   | merge_start_time        | 1512612823566318 |                |
| 2017-12-07 10:13:43.567677 | 2017-12-07 10:13:43.567677 | z3   | merge_status            |               0 | IDLE           |
| 2017-12-07 10:13:43.567822 | 2017-12-07 10:35:30.421431 | z3   | region                  |               0 | HANGZHOU       |
| 2017-12-07 10:13:43.566342 | 2017-12-07 10:13:43.566342 | z3   | status                  |               2 | ACTIVE         |
| 2017-12-07 10:13:43.567537 | 2017-12-07 10:13:43.567537 | z3   | suspend_merging         |               0 |                |
| 2017-12-07 10:13:43.568108 | 2017-12-07 10:36:10.195361 | z3   | zone_type               |               1 | ReadOnly       |
+----------------------------+----------------------------+------+-------------------------+-----------------+----------------+
53 rows in set (0.00 sec)
```

6.3 LDC配置： Obproxy的配置

proxy的LDC的支持全局级别 和 session级别（Obproxy 2883 ）

- **全局级别**, 配置项proxy_idc_name用来控制全局级别的当前IDC机房信息, 默认为空。配置项的设置可以通过启动参数/登陆修改/ocp配置项更新进行, 在proxy的启动脚本中使用-i 机房名启动传入，或者proxy运行后通过alter proxyconfig set proxy_idc_name='机房名';设置
- **session级别**, 设置用户变量set @proxy_idc_name='xx'控制session级别的当前机房信息, 默认不指定，用户可以通过进行设置

6.3 LDC配置： Obproxy 的配置

ObProxy配置机房名方式：

1.启动时通过启动参数(推荐)：

```
/opt/taobao/install/obproxy/bin/obproxy -n cif -o proxy_idc_name=gtj
```

2.修改proxy配置项：

```
mysql> alter proxyconfig set proxy_idc_name = 'gtj';
Query OK, 0 rows affected (0.01 sec)
```

6.3 LDC配置： 检查LDC的匹配情况

通过执行内部命令show proxyinfo idc;可以检查proxy内部识别的LDC部署情况

```
mysql> show proxyinfo idc;
+-----------------+----------------+----------------+----------------+-----------------------------------------------------------------+-------------+--------------------+
| global_idc_name | cluster_name   | match_type     | regions_name   | same_idc                                                        | same_region | other_region       |
+-----------------+----------------+----------------+----------------+-----------------------------------------------------------------+-------------+--------------------+
| zue             | ob2.jianhua.sjh | MATCHED_BY_NONE | □              | [[0]"z1", [1]"z1", [2]"z2", [3]"z2", [4]"z3", [5]"z3"]          | □           | □                  |
| zue             | ob1.jianhua.sjh | MATCHED_BY_IDC  | [[0]"SHANGHAI"] | [[0]"z1", [1]"z1", [2]"z2", [3]"z2"]                            | □           | [[0]"z3", [1]"z3"] |
| zue             | MetaDataBase   | MATCHED_BY_IDC  | [[0]"SHANGHAI"] | [[0]"z1", [1]"z1", [2]"z2", [3]"z2"]                            | □           | [[0]"z3", [1]"z3"] |
+-----------------+----------------+----------------+----------------+-----------------------------------------------------------------+-------------+--------------------+
3 rows in set (0.00 sec)
```

6.3 Obproxy 主要路由策略

写请求：

- 写请求路由到ReadWrite zone的主副本

读请求：

- 强一致性读
- 弱一致性读
  - 主备均衡路由策略（默认）
  - 备优先读策略
  - 读写分离策略

6.3 强一致性读路由策略

- 默认
- 需要读取partition主的数据
- 即SQL必须转发到涉及partition的leader server上执行, 以此保证获取到实时最新的数据
- 强一致性读适用于对读写一致性要求高的场景

6.3 弱一致性读：配置弱一致性读

1. 用户设置**当前租户全局系统**session变量set global ob_read_consistency='weak'，对当前租户所有会话都生效（当前会话不生效）

2. **session级别**， set ob_read_consistency='weak'，只对当前正在连接的session（会话）生效

3. **sql hint**，用户在select中加/*+ read_consistency(weak)*/的Hint，仅本语句生效

6.3 弱一致性读：主备均衡路由策略（默认）

- 可以读主也可以读备，流量按照副本均匀分配
- 弱一致性读时, 或者建立连接时, 或者强一致性读找不到partition时, 或者强一致性读partition主不可用时, 按照:本地常态 -> 同城常态 -> 本地合并 -> 同城合并 -> 异地常态

1. 选取本机房不在合并的副本;
2. 选取同地域机房不在合并的副本;
3. 选取本机房在合并的副本;
4. 选取同地域机房在合并的副本;
5. 随机选取非本地域机房不在合并的副本;
6. 随机选取非本地域机房在合并的副本;

6.3 弱一致性读：备优先读策略

- OBProxy 支持备优先读路由策略，通过用户级别系统变量 proxy_route_policy 控制备优先读路由。备优先读仅在弱一致性读时生效，且优先读 follower 而非主备均衡选择
- 使用 root 用户登录集群的 sys 租户后，运行下述语句对用户系统变量 proxy_route_policy 进行设置：

设置命令 SET @proxy_route_policy='[policy]' ;

验证命令 select @proxy_route_policy; | show proxysession variables;

取值为 follower_first 时
路由逻辑是优先发备（即使集群在合并状态）

```
同机房不合并的备 --> 同城不同机房不合并的备 -->
同机房在合并的备 --> 同城不同机房合并的备 -->
同机房不合并的主 --> 同城不同机房不合并的主 -->
不同城不合并的备 --> 不同城合并的备 -->
不同城不合并的主 --> 不同城合并的主
```

取值为 unmerge_follower_first 时
路由逻辑是优先发不在集群合并状态的备机（Follower 节点）

```
同机房不合并的备 --> 同城不同机房不合并的备 -->
同机房不合并的主 --> 同城不同机房不合并的主 -->
同机房在合并的备 --> 同城不同机房合并的备 -->
不同城不合并的备 --> 不同城不合并的主 -->
不同城合并的备 --> 不同城合并的主
```

注：当取其他值时，退化至普通弱一致性读主备均衡的路由逻辑

6.3 弱一致性读：读写分离策略

- 读写分离部署，要客户端将写请求路由到 ReadWrite Zone主副本，将弱一致性读请求路由到ReadOnly Zone
- 使用 root 用户登录到集群的业务租户，运行下述语句设置 Global 级别系统变量 ob_route_policy 为对应取值即可，默认取值为 readonly_zone_first：

```
set @@global.ob_route_policy=readonly_zone_first;
```

值	路由策略	说明
readonly_zone_first	1.本地常态读库PS -> 2.同城常态读库PS -> 3.本地合并读库PS -> 4.同城合并读库PS ->	默认值, 读库优先访问 优先级： zone类型 > 合并状态 > idc状态
only_readonly_zone	1.本地常态读库PS -> 2.同城常态读库PS -> 3.本地合并读库PS -> 4.同城合并读库PS -> 5.本地常态读库TS ->	只访问读库 优先级: 合并状态 > idc状态
unmerge_zone_first	1.本地常态读库PS -> 2.同城常态读库PS -> 3.本地常态写库PS -> 4.同城常态写库PS->	优先访问不在合并的zone 优先级：合并状态 > zone类型 > idc状态

6.3 OBProxy使用限制

1. proxy parser在根据SQL选择server时，有以下几点特殊的逻辑：

   ① proxy parser只解析Begin/START TRANSACTON/SET/和其他DML，如果遇到其他单词开头的语句，proxy的parser会直接跳过，认为该语句不包含表名

   ② proxy parser会按照第一条包含实体表名的stmtement进行路由，如果整个stmtement都不包含表名，则将请求发送至上一条SQL所发送的server

2. observer会根据执行计划的类型，来告诉proxy是否将请求路由至正确的server，如果路由失败，proxy会更新location，当前的反馈机制如下：

   - server返回第一条DML的命中情况

6.3 示例 - 比较推荐的用法

1. 以下几种情况（select可以等价替换成update/delete/replace/insert，下同），proxy能够将请求发送至正确的server，并且server能够按照proxy的命中情况进行反馈：

    ① begin; select * from t1; commit;

    ② set @@autocommit = 1; insert into t1 values(); set @@autocommit = 0;

    ③ select * from t1; insert into t2 values;

    ④ set @@ob_trx_timeout = 10000000; begin; select * from t1; commit;

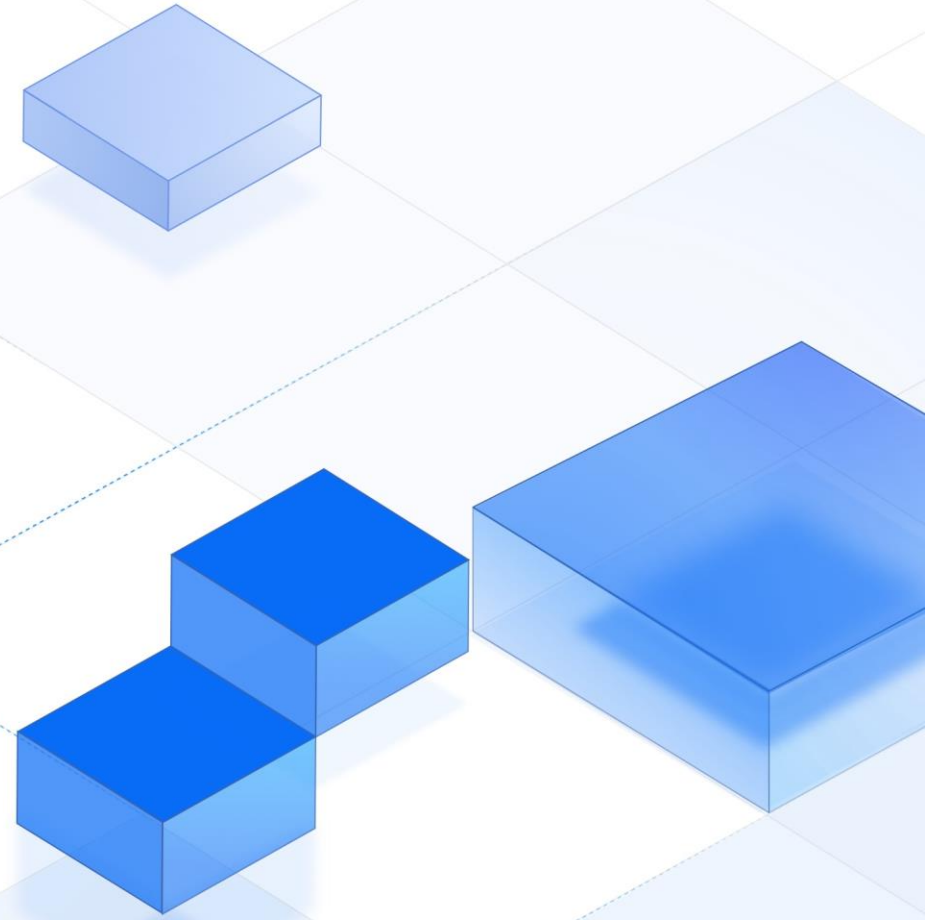
2. 以下几种情况，proxy会将请求发送至上一个SQL所使用的server

    create table t1 (id int primary key); create table t2 (id int primary key);

6.3 示例 – 不建议使用的语句

1. 以下几种情况（第一个DML是非实体表），proxy能够将请求发送至正确的server，但是server反馈的信息可能不准，不建议使用：
   ① select '1'; select * from t1;
   ② select '1' from dual; select * from t1;
2. 以下几种情况，proxy可能能够将请求路由至正确的server，但是server端反馈信息可能不准，不建议使用：
   create table t1 (id int primary key); insert into t1 values();
3. 以下几种情况，proxy会强制将请求路由至上一次使用的server，server端反馈信息可能不准，不建议使用：
   ① show warnings; select * from t1;
   ② show count(*) errors; select * from t1;

6.4 OBProxy 的使用与运维



6.4 生产环境运行OBProxy - 守护进程

- ObProxy无状态，即使宕机重启也不会导致数据一致性，所以ObProxy在部署时都带有一个守护进程，周期性检查Obproxy的健康程度，一旦发现宕机就立即重启ObProxy

- 使用方法：使用 obproxyd.sh -c COMMAND -e ENVIRONMENT [-n APP_NAME] [-i IDC_NAME] 启动OBProxy

```
e.g.
1. ./bin/obproxyd.sh -c start -e alipay -n obpay -i zue
2. ./bin/obproxyd.sh -c start -e offline -n obpay

e.g.
1. obproxyd.sh -c start -e alipay
   在蚂蚁金服机房启动obproxy
2. obproxyd.sh -c start -e alipay -n obtrade -i zue
   在蚂蚁金服zue机房启动obproxy，并制定该obproxy的业务名为obtrade
3. obproxyd.sh -c start -e inc -n obpay -i su18
   在集团su18机房启动obproxy，并制定该obproxy的业务名为obpay
```

6.4 OBProxy配置项

1. 系统租户，通过OBProxy连接OceanBase集群
2. 涉及配置项的内部命令有两种，如下：2883 obproxy； 2881 observer
    1. show proxyconfig，展示proxy内部各配置项属性以及config server的配置信息
    2. alter proxyconfig set key=value，更新指定config配置项值
3. 注：更新命令只对除config server配置信息之外的其他配置项有效，config server配置信息只能通过config server来更新
4. 有些配置需要重启Proxy才生效（参考 need_reboot 这列的值）

6.4 OBProxy配置项

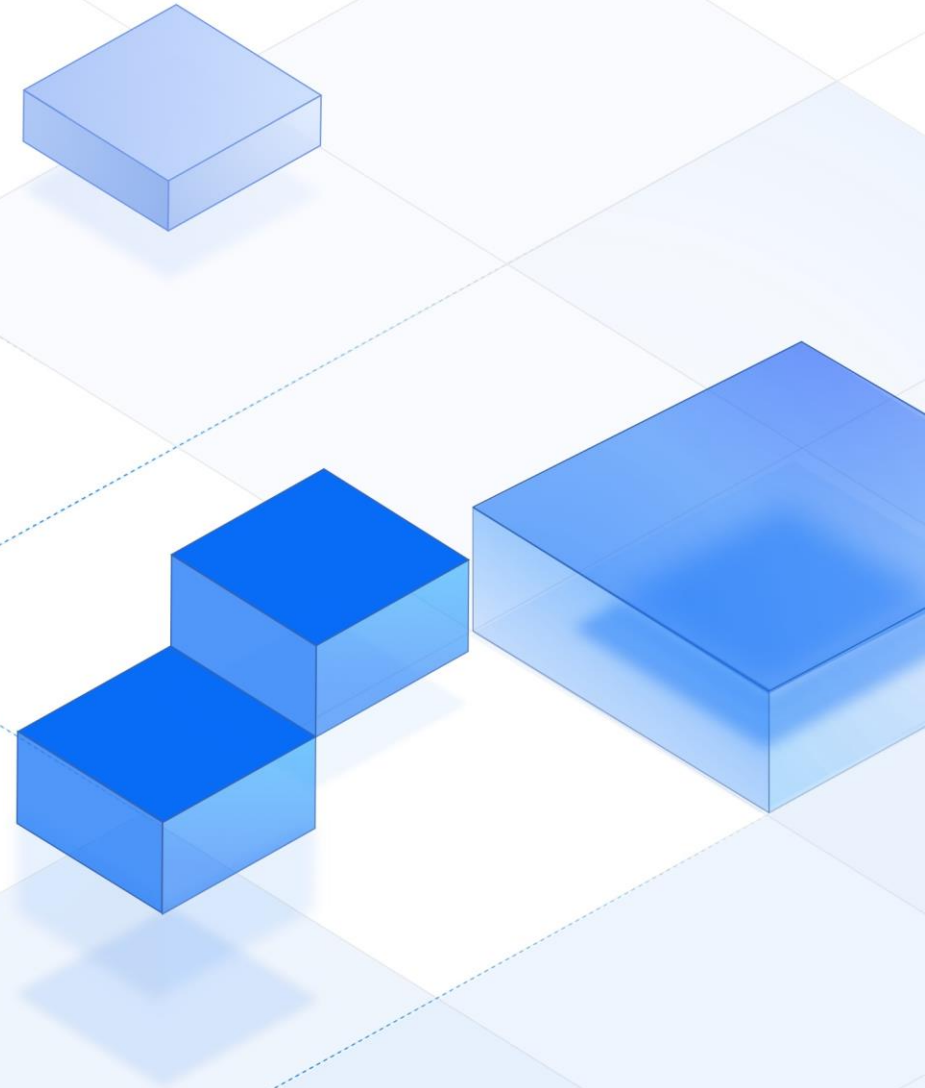
配置项可以分为3种类型况来说明：

1. 第一种是proxy写到本地etc文件夹中配置文件的配置项，这些配置项可供用户根据使用场景进行配置和更新
2. 第二种是proxy内部自己使用，对一般用户不可见的配置项，不会注册到内部表中
3. 第三种是proxy从config server中获取到的配置信息(包括版本号、meta table信息、cluster信息、bin url和更新时间)，这些信息只用来展示config server的配置，不会注册到内部表或者dump到本地配置文件中，并且它们全部以字符串“json_config”开头，查询时可以使用like进行过滤

6.4 常用OBProxy配置项

1. xflush_log_level：监控用的xflush的日志级别
2. syslog_level： OBProxy自己的应用日志的日志级别
3. observer_query_timeout_delta：关系到网络断开连接，到认为Observer不可用的delta时间
4. log_cleanup_interval:：清理OBProxy自身应用日志的间隔时间
5. log_dir_size_threshold
6. internal_cmd_mem_limited：会话较多，导致buffer内存不足时需要调大

6.5 OBProxy 常见问题



6.5 OBProxy的启动

ObProxy只要能启动成功、建立连接，就解决90%的问题

```
obproxy [OPTIONS]
  -h,--help                          print this help
  -p,--listen_port          LPORT    obproxy listen port
  -o,--optstr               OPTSTR   extra options string
  -n,--appname              APPNAME  application name
  -r,--rs_list              RS_LIST  root server list(format ip:sql_prot)
  -c,--cluster_name         CLUSTER_NAME  root server cluster name
  -d,--dump_config_sql      DSQL     dump config sql to file
  -e,--execute_config_sql   ESQL     exectue config sql(create tables, insert initial data)
  -N,--nodaemon                      don't run in daemon
  -V,--version              VERSION  current obproxy version
  -R,--releaseid            RELEASEID  current obproxy kernel release id
  -t,--regression_test      TEST_NAME  regression test
```

6.5 启动失败

1. 机器是否存在 hostname ： 输入hostname -i, 确认host ip是否存在
2. 目录是否存在，权限是否正确：确保当前目录下有读、写、执行的权限
3. 端口是否被占用：使用obproxyd.sh启动OBProxy, 使用的端口为2883
4. 启动环境是否指定正确：如果通过obproxyd.sh启动， 需要使用-e参数指定OBProxy运行环境

6.5 无法建立连接

OBProxy启动成功后, 当使用客户端进行连接OBProxy的时候出错

类型	ERROR
IP PORT 错误	ERROR 2003 (HY000): Can't connect to MySQL server on '127.1' (111)
权限错误	ERROR 1045 (42000): Access denied for user 'XXXXXXXXX'
租户名错误	ERROR 5160 (HY000): invalid tenant name specified in connection string
认证错误	ERROR 2013 (HY000): Lost connection to MySQL server at 'reading authorization packet', system error: 0

6.5 无法建立连接 - IP PORT错误

ERROR 2003 (HY000): Can't connect to MySQL server on '127.1' (111)

检查所需建立连接的obproxy, obproxy是否存在

6.5 无法建立连接 -权限错误 / 租户名错误 / 认证错误

- 权限错误： ERROR 1045 (42000): Access denied for user ‘XXXXXXXXX’
- 租户名错误： ERROR 5160 (HY000): invalid tenant name specified in connection string
- 认证错误： ERROR 2013 (HY000): Lost connection to MySQL server at ‘reading authorization packet’, system error: 0

- 用户名/租户名/集群名/密码 是否正确
  - 可以直接连接observer确认该信息是否正确, 注意连接observer的时候不能带集群名
- 本机mysql版本是否过低
  - MySQL 5.7.8之前版本， 用户名长度超过16字节会被截断。5.7.8版本之后版本用户长度超过32字节会被截
  - 这里的用户名包含完整的username@tenantname#clustername
- 本地json配置集群是否和远程json文件一致
  - 该配置文件主要用于确认你需要连接的OB集群是否存在

6.5 慢查询

- proxy慢查询? - 分析proxy日志
- server show trace? - 分析server日志
- 远程执行? 为什么?
- 机器负载如何?

- 配置项默认为5s
  slow_transaction_time_threshold=5s
- 修改配置项
  e.g. 将慢查询阈值设置为 100ms
  alter proxyconfig set
  slow_transaction_time_threshold= ‘100ms';

6. 5 慢查询举例

[2016-05-24 22:39:00.824392] WARN [PROXY.SM] update_cmd_stats (ob_mysql_sm.cpp:4357) [28044][Y0-7F70BE3853F0] [14] Slow query:

client_ip=127.0.0.1:17403 // 执行SQL client IP
server_ip=100.81.152.109:45785 // SQL被路由到的observer
conn_id=2147549270
cs_id=1
sm_id=8
cmd_size_stats=
client_request_bytes:26 // 客户端请求 SQL大小
server_request_bytes:26 // 路由到observer SQL大小
server_response_bytes:1998889110 // observer转发给obproxy数据大小
client_response_bytes:1998889110 // obproxy转发给client数据大小
cmd_time_stats=
client_transaction_idle_time_us=0 // 在事务中该条SQL与上一条SQL执行结束之间的间隔时间, 即客户端事务中SQL间隔时间
client_request_read_time_us=0 // obproxy读取客户端SQL耗时
server_request_write_time_us=0 // obproxy将客户端SQL转发给observer耗时
client_request_analyze_time_us=15 // obproxy 解析本条SQL消耗时间
cluster_resource_create_time_us=0 // 如果执行该条SQL, 需要收集OB cluster相关信息(一般发生在第一次连接到该集群的时候), 建立集群相关信息耗时
pl_lookup_time_us=3158 // 查询partition location耗时
prepare_send_request_to_server_time_us=3196 // 从obproxy接受到客户端请求，到转发到observer执行前总计时间，正常应该是前面所有时间之和
server_process_request_time_us=955 // observer处理请求的时间，对于流式请求就是从observer处理数据，到第一次转发数据的时间
server_response_read_time_us=26067801 // observer转发数据到obproxy耗时，对于流式请求observer是一边处理请求，一边转发数据(包含网络上的等待时间)
client_response_write_time_us=716825 // obproxy将数据写到客户端耗时
request_total_time_us=26788969 // 处理该请求总时间
sql=select * from sbtest1 // 请求SQL

6.5 SQL相关timeout设置

默认配置：

- 查询超时：ob_query_timeout(默认10s)
- 事务未提交超时ob_trx_timeout 默认100s）
- 事务空闲超时 ob_trx_idle_timeout（默认120s）

常见原因：

- 连接闲置时间超时
- 客户端SQL执行耗时长

问题解决：

- Ping to keep alive.
- 调优SQL or 调整timeout value

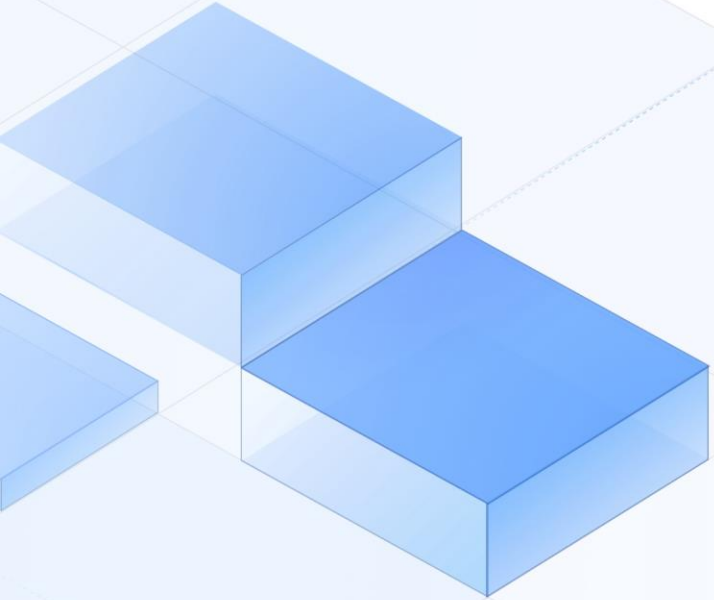
```
create table test (id1 int);
set @@ob_query_timeout=200 * 1000 * 1000;
set @@ob_trx_timeout = 200 * 1000 * 1000;

begin;
insert into test values(1);
//中间间隔150s
insert into test values(2); //当前这条SQL会提示Lost connection
select * from test;  //这条查询请求，结果为空，说明上述事务已经回滚。
rollback;
```

Communications link failure

```
--- Cause: com.mysql.jdbc.exceptions.jdbc4.CommunicationsException:
Communications link failure
The last packet successfully received from the server was 70,810
milliseconds ago. The last packet sent successfully to the server was
5,005 milliseconds ago.
```

感谢学习



OCEANBASE