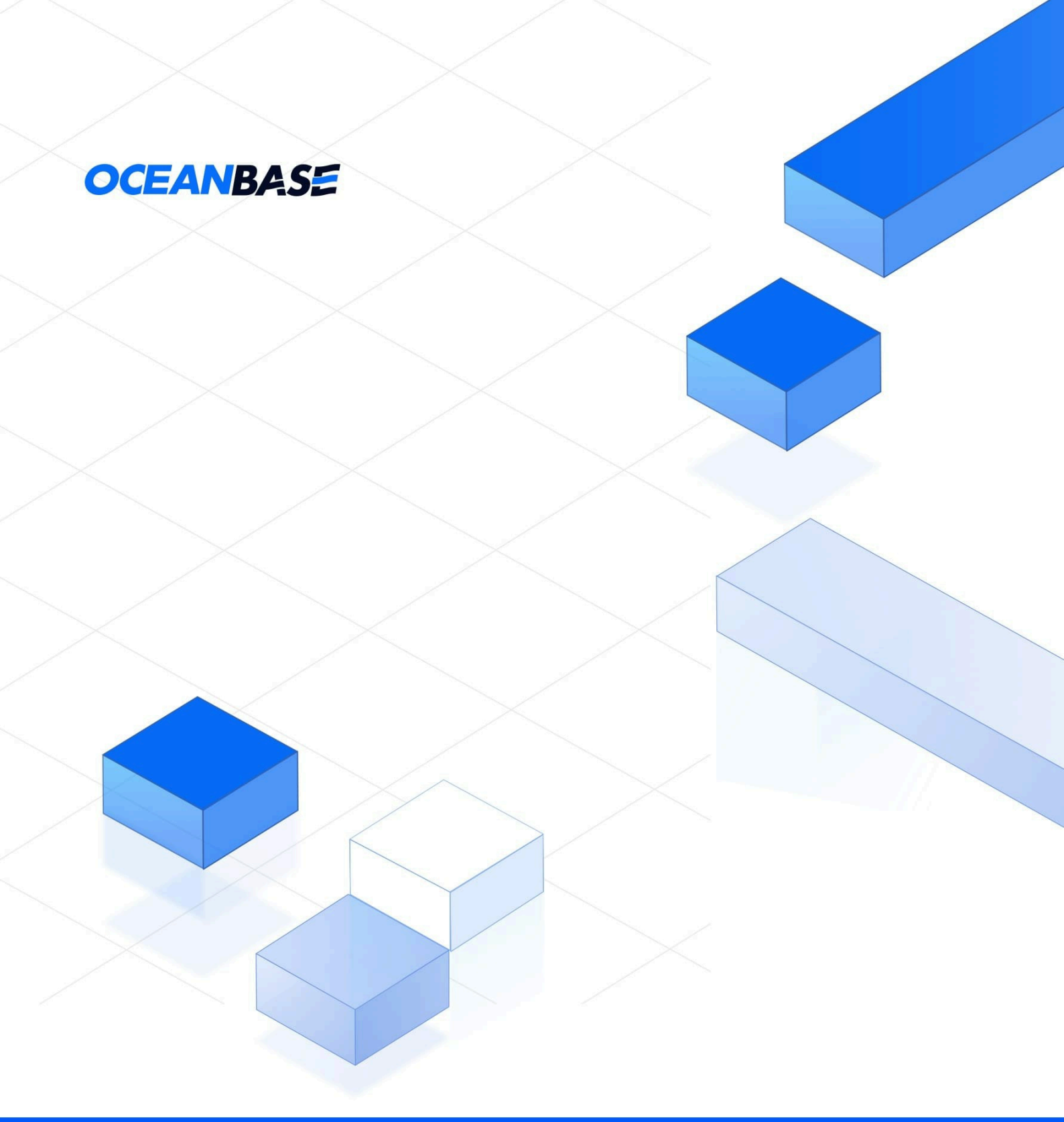


# OceanBase 数据库

## 参考指南--数据库代理

| 产品版本：V4.5.0 | 文档版本：20251219

# 声明

## 北京奥星贝斯科技有限公司版权所有©2024，并保留一切权利。



## 商标声明



## 免责声明

# 通用约定

| 格式 | 说明 | 样例 |
| --- | --- | --- |
| 危险 | 该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。 | **危险**<br>重置操作将丢失用户配置数据。 |
| 警告 | 该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。 | **警告**<br>重启操作将导致业务中断，恢复业务时间约十分钟。 |
| 注意 | 用于警示信息、补充说明等，是用户必须了解的内容。 | **注意**<br>权重设置为0，该服务器不会再接受新请求。 |
| 说明 | 用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。 | **说明** 您也可以通过按Ctrl+A选中全部文件。 |
| > | 多级菜单递进。 | 单击**设置**> **网络**> **设置网络类型**。 |
| **粗体** | 表示按键、菜单、页面名称等UI元素。 | 在**结果确认**页面，单击**确定**。 |
| `Courier字体` | 命令或代码。 | 执行 `cd /d C:/window` 命令，进入Windows系统文件夹。 |
| *斜体* | 表示参数、变量。 | `bae log list --instanceid`<br>*Instance_ID* |
| [] 或者 [a\|b] | 表示可选项，至多选择一个。 | `ipconfig [-all\|-t]` |
| {} 或者 {a\|b} | 表示必选项，至多选择一个。 | `switch {active\|stand}` |

# 目录

# 1 OceanBase 数据库代理（ODP）

OceanBase 数据库代理 ODP（OceanBase Database Proxy）是 OceanBase 数据库专用的代理服务器，OceanBase 数据库用户的数据会以多副本的形式存放在各个 OBServer 节点上，ODP 接收用户发出的 SQL 请求，并将 SQL 请求转发至最佳目标 OBServer 节点，最后将执行结果返回给用户。

## 1.1 特性

作为 OceanBase 数据库的关键组件，ODP 具有以下特性：

- 连接管理

  针对一个客户端的物理连接，ODP 维持自身到后端多个 OBServer 节点的连接，采用基于版本号的增量同步方案维持了每个 OBServer 节点连接的会话状态，保证了客户端高效访问各个 OBServer 节点。

- 最佳路由

  ODP 充分考虑用户请求涉及的副本位置、用户配置的读写分离路由策略、OceanBase 多地部署的最优链路，以及 OceanBase 各机器的状态及负载情况，将用户的请求路由到最佳的 OBServer 节点，最大程度地保证了 OceanBase 整体的高性能运转。

- 高性能转发

  ODP 完整兼容 MySQL 协议，并支持 OceanBase 自研协议，采用多线程异步框架和透明流式转发的设计，保证了数据的高性能转发，同时确保了自身对机器资源的最小消耗。

- 易运维

  ODP 本身无状态，支持无限水平扩展，支持同时访问多个 OceanBase 集群。您可通过丰富的内部命令对 ODP 状态进行实时监控，这使得运维简单便利。

- 高可用

  ODP 高可用分为两部分：一方面保证自身高可用，持续提供代理服务；另一方面 ODP 是 OceanBase 高可用体系的主要组成部分，可以对用户屏蔽宕机、升级等情况，保证 OceanBase 数据库服务的稳定和快速恢复。

- 专有协议

  ODP 与 OBServer 节点默认采用了 OceanBase 专有协议，如增加报文的 CRC 校验保证与 OBServer 节点链路的正确性，增强传输协议以支持 Oracle 兼容性的数据类型和交互模型。

# 2 数据库连接和路由概述

## 2.1 连接

OBProxy 为用户提供了数据库接入和路由功能，用户连接 OBProxy 就可以正常使用 OceanBase 数据库。用户在使用数据库功能时，OBProxy 和 OBServer 进行交互，且交互流程对用户透明，连接管理就是该交互过程中的关键点之一。 数据库连接包括物理连接和逻辑连接。物理连接主要是指网络连接部分，逻辑连接主要是 ODP 与 OceanBase 数据库之间的 Session 部分。

### 2.1.1 特性

OBProxy 的连接管理有三个特性：

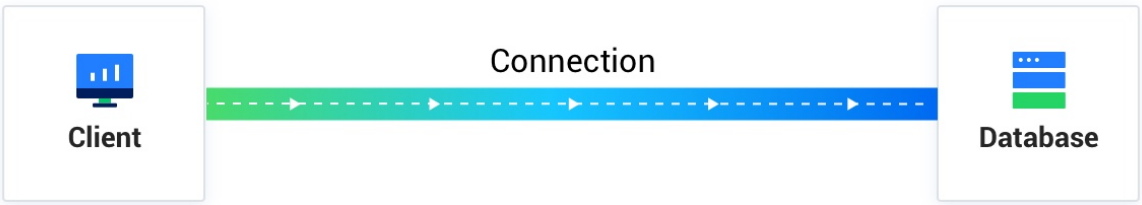
1. 代理特性：OBProxy 既是客户端，也是服务端，还需要保证交互行为符合 MySQL 协议规范。
2. 功能特性：OBProxy 实现了很多的连接功能特性，如访问不同集群、不同租户，再如支持物理备库、分布式下的 PREPARE Statement 功能，以及兼容 kill、show processlist 等命令。
3. 高可用特性：OBProxy 可以处理超时、机器状态变化、网络状态变化等问题，屏蔽后端异常，让用户无感知。

#### 2.1.1.1 说明

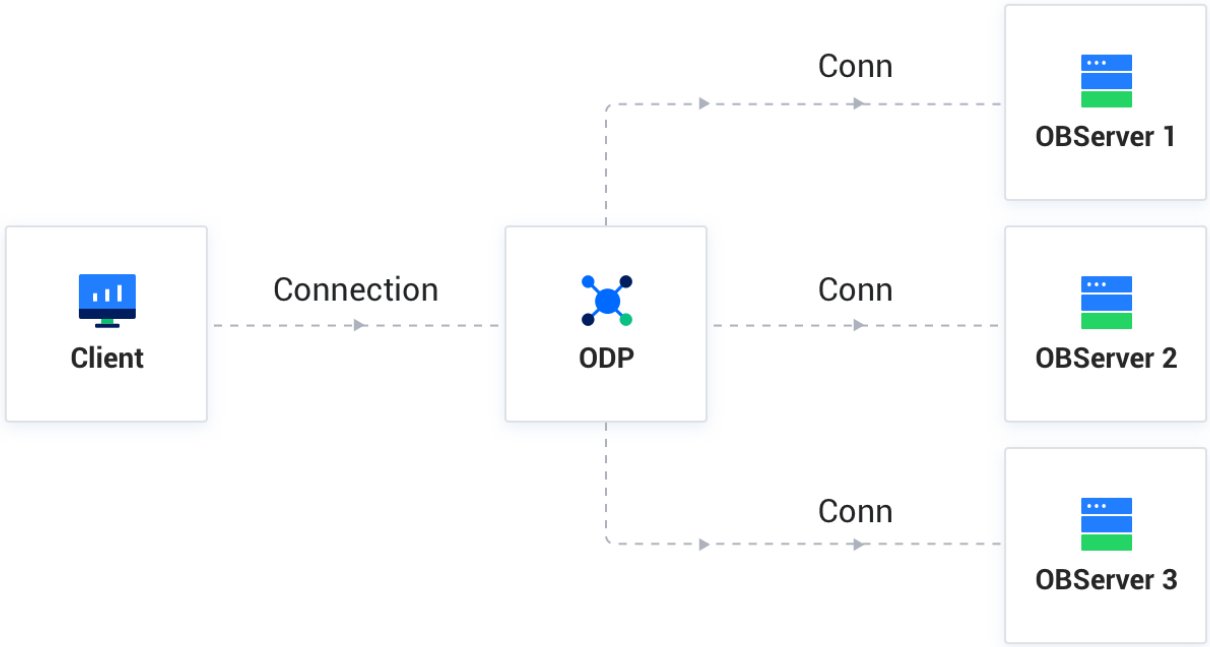
ODP 支持连接 OceanBase 数据库的 MySQL 模式租户和 Oracle 模式租户。

### 2.1.2 连接映射关系

OceanBase 不同于单机数据库，当你通过客户端与单机数据库建立一个连接时，你的客户端与数据库之间只有一个物理连接，如下图所示。




当你通过 OBProxy 与 OBServer 建立一个连接时，你的客户端与 OBProxy 之间存在一个物理连接，OBProxy 与 OBServer 可能存在多个物理连接，如下图所示。 其中 Client 与 OBProxy 的连接称为客户端连接，OBProxy 与 OBServer 之间的连接称之为服务端连接。




当你的客户端所访问的数据在不同的 OBServer 上时，OBProxy 会为你向 OBServer 建立多个物理连接并且管理复用这些连接，在客户端视角看来仅存在一个逻辑连接，OBProxy 基于此可以为客户提供许多功能特性，比如读写分离、分区表数据路由、分布式 PS、屏蔽后端异常等等。

# 2.1.3 连接功能特性

和单机数据库不同，OBProxy 将连接的映射关系改变为 M:N，因此有些连接功能需要做额外处理。 举例说明：用户通过 `show processlist` 查看连接数，此时希望看到的是客户端和 OBProxy 之间的连接数，而不是 OBProxy 和 OBServer 节点之间的连接数。

下面我们对常见的连接功能展开详细介绍：

- 连接粘性 OBProxy 还未实现所有功能的状态同步，如事务状态、临时表状态、cursor 状态等。对于这些功能，OBProxy 只会将后续请求都发往状态开始的节点，这样就不需要进行状态同步，而缺点是无法充分发挥分布式系统的优势。因此，OBProxy 将根据功能重要程度，逐步支持相关功能的分布式化。
- `show processlist` 和 `kill` 命令配套使用 `show processlist` 用于展示客户端和服务端之间的连接，对于 OBProxy 来说，`show processlist` 只展示客户端和 OBProxy 之间的连接，不展示 OBProxy 和 OBServer 节点之间的连接。`kill` 命令用于杀死一个客户端连接，客户端连接关闭后，OBProxy 也会关闭对应的服务端连接。对于 OBProxy 的 `kill` 命令，需要先获取对应的 ID（使用 `show processlist` 命令即可获取 ID）。

- 负载均衡影响因为 OBProxy 对 `show processlist` 和 `kill` 命令做了处理，所以 `show processlist` 和 `kill` 命令只有都发往同一台 OBProxy 才能正常工作。在公有云等环境，OBProxy 前面有负载均衡，负载均衡后面挂载多个 OBProxy 上，此时，如果执行 `show processlist` 和 `kill` 命令是两个不同的连接，负载均衡组件可能将请求发往不同的 OBProxy，在这种情况下，建议不要使用相关命令。

# 2.2 路由

路由是 OceanBase 分布式数据库中的一个重要功能，是分布式架构下，实现快速访问数据的利器。

Partition 是 OceanBase 数据存储的基本单元。当我们创建一张 Table 时，就会存在表和 Partition 的映射。非分区表中，不考虑主备时，一张 Table 对应一个 Partition；分区表中一个 Table 会对应多个 Partition。

路由实现了根据 OBServer 的数据分布精准访问到数据所在的机器。同时还可以根据一定的策略将一致性要求不高的读请求发送给副本机器，充分利用机器的资源。路由选择输入的是用户的 SQL、用户配置规则、和 OBServer 状态，路由选择输出的是一个可用 OBServer 地址。其路由逻辑如下图所示：


解析 SQL 模块使用的是 OBProxy 自己定制的 Parser 模块，只需要解析出 DML 语句中的数据库名、表名和 Hint，不需要通过其他复杂的表达式推演。

确定路由规则模块中，OBProxy 需要根据不同情况确定最佳的路由规则。比如：强一致性读的 DML 请求期望发到分区所属日志流的主副本 OBServer 上，弱一致性读的 DML 请求和其他请求则不要求，主副本和从副本均衡负载即可。如果 OceanBase 集群是多地部署，OBProxy 还提供了 LDC 路由，优先发给同机房的 OBServer，其次是同城的 OBServer，最后才是其他城市的 OBServer。如果 OceanBase 集群是读写分离部署，OBProxy 还提供了读 Zone 优先、只限读 Zone、非合并优先等规则供业务按照自身特点配置。上述的几种情况在路由选择中是组合关系，输出是一个确定的路由规则。

获取路由表是指 OBProxy 根据用户的请求 SQL 获取该 SQL 涉及的副本位置。OBProxy 每次首先会尝试从本地线程缓存中获取路由表，其次是全局缓存，最后才是发起异步任务去向 OBServer 查询路由表。对于路由表的更新，OBProxy 采用触发更新机制。OBProxy 每次根

据路由表转发给 OBServer 的请求，当 OBServer 不能本地执行时，会在回包时反馈给 OBProxy。OBProxy 根据反馈决定是否下次强制更新本地缓存路由表。通常是在 OBServer 合并或者负载均衡导致切主时，路由表才会发生变化。

选择目标 OBServer 则是根据确定的路由规则从上一步获取的路由表中选择最佳的 OBServer，在经过黑名单、灰名单检查通过后作为最终的目标 OBServer 进行请求转发。

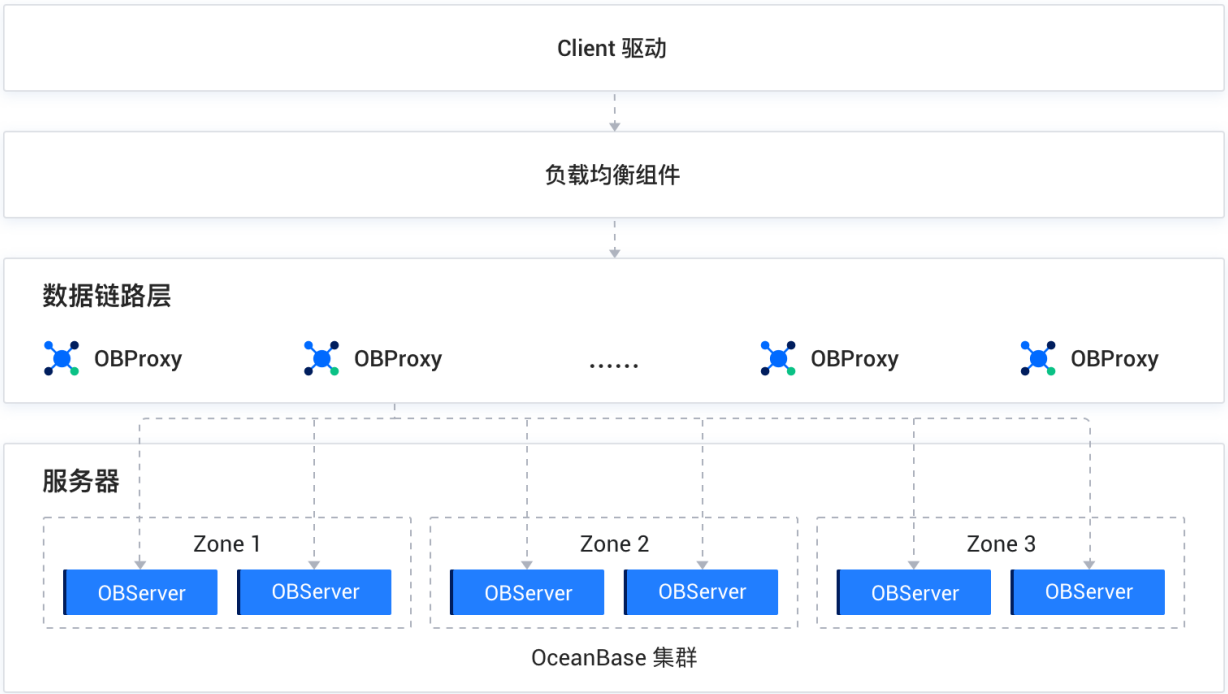
# 3 ODP 管理概述

OceanBase Database Proxy（简称 ODP）是 OceanBase 数据库专用的代理服务器。OceanBase 数据库的用户数据以多副本的形式存放在各个 OBServer 节点上，ODP 接收用户发出的 SQL 请求，并将 SQL 请求转发至最佳目标 OBServer，最后将执行结果返回给用户。

## 3.1 ODP 概述

OceanBase 数据库与传统单机数据库不同，OceanBase 数据库是分布式数据库，每个表甚至每个表的不同分区都可能存放在不同的机器上。想要对表进行读写，必须先要定位到数据所属的表或是分区的主副本位置，然后才能执行相应的 SQL 语句，这在应用层面而言是几乎不可能做到的。ODP 作为 OceanBase 数据库专用的反向代理软件，其核心功能是路由，将客户端发起的数据访问请求转发到正确的 OBServer 节点上，并将 OBServer 节点的响应结果转发给客户端。

客户端通过 ODP 访问 OceanBase 数据库的数据链路如下图所示。




用户通过任意 Client 驱动发出请求，请求通过负载均衡组件访问到任意一台无状态的 ODP 上，然后 ODP 再将用户请求转发到后端 OceanBase 集群中最佳的 OBServer 节点上去执行。

### 3.1.0.1 说明

- 这里负载均衡组件可以是市场上常见的产品，例如：SLB 和 F5 等。

- ODP 不负责分库分表，也不作为 SQL 引擎参与执行计划的生成调度，只负责纯粹的反向代理转发。

每个 OBServer 节点均包含完整的 SQL 引擎和存储引擎，用来负责解析用户 SQL 以生成物理执行计划并执行。分布式的 OBServer 节点之间通过 Paxos 协议以保证高可用性。这种架构设计中，ODP 只承担基本的路由和容灾功能，而数据库的功能全部交由 OBServer 节点实现。这样更加简单明确的分工可以将各组件性能做得更加极致，OceanBase 数据库整体最高也能做到近似访问单机数据库的性能。

ODP 支持将请求正确发送至主副本，并且通过特定配置还支持读写分离和备优先读等场景。另外在 OBServer 节点发生宕机、升级或合并等状态时，可以通过 黑名单机制 确保用户请求可以被路由至状态正常的 OBServer 节点上。

# 3.2 管理 ODP

OCP 为 ODP 提供以下管理功能。

| 操作 | 说明 |
| --- | --- |
| 创建 OBProxy 集群 | 您可通过该操作创建 OBProxy 集群。 |
| OBProxy 参数说明 | 您可通过该操作管理 OBProxy 集群的参数。 |
| 删除 OBProxy 集群 | 您可通过该操作删除 OBProxy 集群。 |
| 升级 OBProxy 集群下全部 OBProxy | 您可通过该操作升级 OBProxy 集群中所有的 OBProxy。 |
| 重启 OBProxy 集群下的全部 OBProxy | 您可通过该操作重启 OBProxy 集群下全部 OBProxy。 |
| 管理 OBProxy 连接的 OceanBase 集群 | 您可通过 OCP 为 OBProxy 添加可连接的 OB 集群或删除已连接的 OB 集群。 |
| OBProxy 集群性能监控 | 您可以在 OCP 上查看 OBProxy 集群的性能监控信息，包括了服务监控信息和系统监控信息。 |
| 添加 OBProxy | 您可通过该操作向 OBProxy 集群中添加 OBProxy。 |
| 启动 OBProxy | 您可通过该操作启动 OBProxy。 |
| 刷新 OBProxy 配置 | 您可通过该操作刷新 OBProxy 的配置。 |
| 停止 OBProxy | 您可通过该操作停止 OBProxy。 |
| 删除 OBProxy | 您可通过该操作删除集群中的 OBProxy。 |
| 重启 OBProxy | 您可通过该操作重启集群中的 OBProxy。 |
| 升级 OBProxy | 您可通过该操作升级集群中的 OBProxy。 |

# 4 创建 OBProxy 集群

通常在部署 OceanBase 数据库时已创建 OBProxy 集群，如果需要添加多个新的 OBProxy 与 OceanBase 集群连接，可以通过 OCP 来创建 OBProxy 集群。

## 4.1 前提条件

当前登录的用户是 OBPROXY_MANAGER 角色，具有管理 OBProxy 的权限。如果当前用户不具备 OBPROXY_MANAGER 角色，需要先为用户添加该角色，具体操作请参见《 OCP 用户指南》中的 编辑用户。

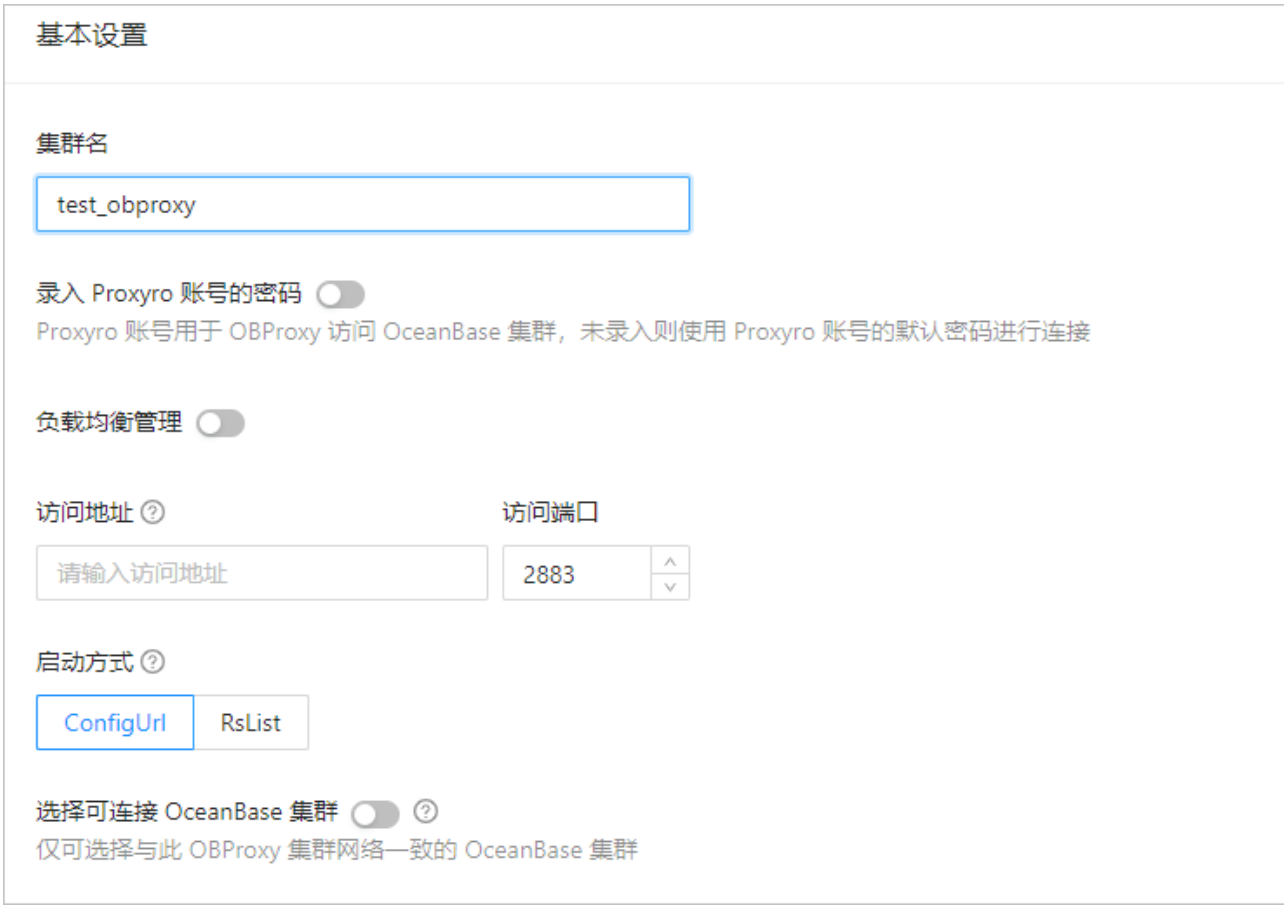
## 4.2 操作步骤

1. 登录 OCP。
2. 在左侧导航栏单击 **OBProxy**，进入 OBProxy 页面。
3. 单击右上方的 **创建 OBProxy 集群**。
4. 在 **创建 OBProxy 集群** 页面填写如下信息：
   a. 参考下表，配置 **基本信息**。

| 参数 | 说明 |
| --- | --- |
| 集群名 | 您可根据实际业务情况，自定义集群名称，以英文字母开头，可包含英文、数字和下划线。 |
| 录入 Proxyro 账号的密码 | Proxyro 账号用于 OBProxy 访问 OB 集群，未录入则使用 Proxyro 账号的默认密码进行连接。密码格式要求如下：<br>● 长度为 8 ~ 32 个字符。<br>● 只能包含字母、数字和特殊字符 ~!@#%^&*_-+=\|(){}[]:;,.?/。<br>● 大小写字母、数字和特殊字符都至少包含 2 个。 |

| | |
|---|---|
| 负载均衡管理 | 开启负载均衡管理后，可通过配置 OBLB 来提升 OBProxy 集群的负载均衡能力。<br>● OBLB 服务：您可选择部署 OCP 时已配置的 OBLB 服务，或单击 **添加 OBLB 服务** 按钮，在右侧面板中新建 OBLB 服务。<br>■ OBLB 服务地址：OBLB 的服务地址。<br>■ OBLB 服务端口：默认为 9090，可根据实际情况进行更改。<br>■ 用户名及密码：OBLB 服务的用户名及密码，用于调用 OBLB 接口的认证信息。<br>● VIP：选择 OBLB 具体的 VIP 地址。<br>● 访问端口：默认为 2883，支持自定义端口号。<br>● 域名配置（可选）：用于指向 VIP 及端口的配置信息，平台未提供 VIP 与域名的映射关系，需自行准备域名解析服务。<br>**4.2.0.1 说明**<br>**负载均衡管理** 中的 **SLB 服务** 仅适用于专有云环境，建议联系技术支持同学进行协助。 |
| 访问地址 | 当 **负载均衡管理** 关闭时展示，为 OBProxy 集群的访问地址，仅用于生成租户的连接串，不影响实际使用，需要自主配置负载均衡，如果是 VIP 地址，还需要您自主申请并绑定到 OBProxy Server。 |
| 访问端口 | 当 **负载均衡管理** 关闭时展示，默认为 2883，需要根据 VIP 的真实端口填写。 |
| 启动方式 | 该 OBProxy 集群的启动方式，可取值：<br>● ConfigUrl：多集群启动方式，即该 OBProxy 集群可访问多个 OceanBase 集群。<br>● RsList：单集群启动方式，即该 OBProxy 集群仅可访问创建 OBProxy 集群时指定的那个 OceanBase 集群，OBProxy 集群创建成功后不可追加可连接的 OB 集群。 |

| | |
|---|---|
| 选择可连接 OceanBase 集群 | 设置该 OBProxy 集群可访问的 OceanBase 集群。<br>● 当 **启动方式** 配置为 ConfigUrl 时，该参数非必填，集群创建成功后，可通过 添加可连接的 OceanBase 集群 添加。<br>● 当 **启动方式** 配置为 RsList 时，该参数必填。 在下拉框中选择对应的集群。<br>● 当密码箱中存在该集群 proxyro 用户的连接凭证时，会默认选择 proxyro 用户。<br>● 当密码箱中不存在该集群 proxyro 用户的连接凭证时，请单击 **新建连接**，为该集群的 proxyro 用户创建连接凭证。 |




b. （可选）配置 **部署 OBProxy**。

如需在创建 OBProxy 集群时就部署 OBProxy，可在该步骤就行配置。否则可跳过该步骤，待集群创建成功后，通过接管 OBProxy 或添加 OBProxy 的方式向集群中添加 OBProxy。

a. 打开配置 **部署 OBProxy** 的开关。

b. 参考下表填写部署信息。

| 参数 | 说明 |
| --- | --- |
| SQL 端口 | 默认为 2883。 |
| Exporter 端口 | 默认为 2884。 |
| 软件版本 | 选择要安装的 OBProxy 版本。<br>4.2.0.2 说明<br>若 **可连接的 OB 集群** 中存在 V4.0 及以上版本的集群时，仅支持选择 V4.0.0 及以上版本的 OBProxy 软件包。 |
| **机房、机型（可选）、主机** | 选择部署 OBProxy 所使用的主机，包括该主机的 **机房、机型（可选）、主机**。 |


系统默认显示两个主机的选择，仅够部署两个 OBProxy，

- 如需部署更多，可单击 **添加 OBProxy** 来增加主机。
- 如只部署 1 个 OBProxy，可单击如上图所示的主机后的删除图标来删除该主机。

c. 打开 **参数设置**，添加或修改启动参数，参数说明参见 OBProxy 参数说明。


5. 单击 **提交**，开始创建 OBProxy 集群。

## 4.2.0.3 说明

安装 obproxy 时系统会检查涉及到的安装目录权限是否正确，目前支持递归检查 `/home/admin/logs/obproxy` 目录及其父目录对指定用户（默认 admin）是否有读权

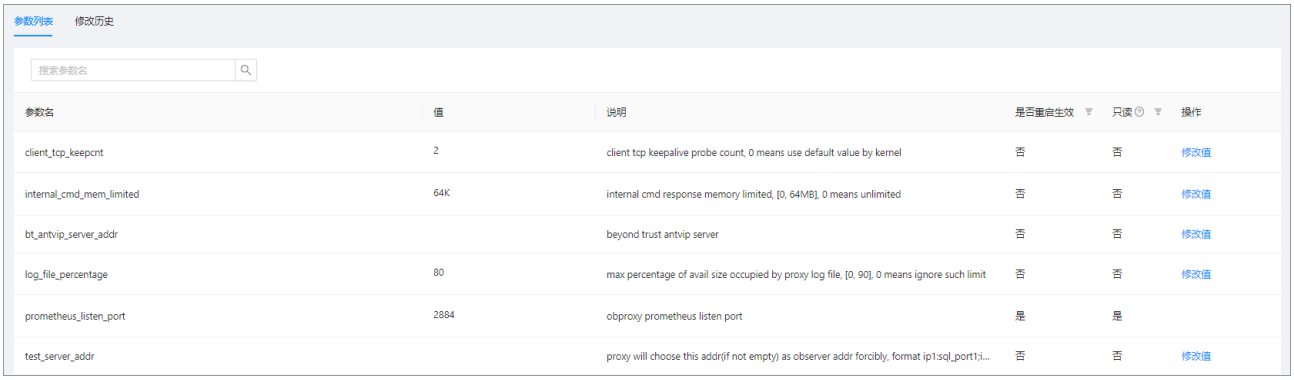
限及可执行权限。若没有，则会在日志中报 “check directory xxx permission failed, reason: xxx” 错误。

# 5 OBProxy 集群参数管理

本文介绍如何通过 OCP，在 OBProxy 集群的参数管理页面查看 **参数列表** 和 **修改历史**。

## 5.1 查看或修改参数

1. 登录 OCP。
2. 在左侧导航栏单击 **OBProxy**。
3. 在集群列表中选择需要操作的 OBProxy 集群，进入 OBProxy 集群的 **概览** 页面。
4. 单击左侧导航栏的 **参数管理**，默认进入 **参数列表** 页签。




在该页面中，您可以进行如下操作：

- 查看参数

  在搜索框中输入参数名，单击搜索按钮，可以查看指定的参数信息，支持模糊查询。

- 修改参数

  a. 单击指定参数 **操作** 列的 **修改值** 按钮。

  b. 填写 **值**，随后单击 **确定**。

  c. 目标参数的值修改完毕后，单击 **提交修改**。

  d. 在弹出的面板中确认修改信息，单击 **确认修改**。等待修改任务执行完成，则修改成功。

  参数修改完成后，可以查看参数的 **是否重启生效** 信息，以确定参数如何生效：

  - 无需重启生效的参数：表示参数修改成功后约 3 分钟即可生效。
  - 需要重启生效的参数：表示需重启 OBProxy 后才可生效。

## 5.2 查看修改历史

1. 登录 OCP。
2. 在左侧导航栏单击 **OBProxy**。
3. 在集群列表中选择需要操作的 OBProxy 集群，进入 OBProxy 集群的 **概览** 页面。

4. 单击左侧导航栏的 **参数管理**，默认进入 **参数列表** 页签。
5. 选择 **修改历史** 页签。

   在 **修改历史** 页签中，您可在 **修改日期**、**参数名称** 中输入相应信息，单击 **查询**，对参数进行筛选。参数列表中可以查看 **参数名**、**值-变更前**（变更前的参数值）、**值-变更后**（变更后的参数值）、**修改时间** 和 **操作用户** 等信息。

# 6 删除 OBProxy 集群

根据业务需要，可以删除不再使用的 OBProxy 集群，会将集群中所有 OBProxy 服务从各主机上卸载，并释放主机上 OBProxy 服务占用的资源。

## 6.1 前提条件

- 当前登录 OCP 的用户为 OBPROXY_MANAGER 角色。
- 确保当前 OBProxy 集群已不再对业务提供代理的服务。

## 6.2 操作步骤

1. 登录 OCP。
2. 在左侧导航栏单击 **OBProxy**。
3. 在集群列表中选择需要操作的 OBProxy 集群，进入 OBProxy 集群 **概览** 页面。
4. 在页面右上角，单击 **...** 图标，选择 **删除集群**。


5. 在弹出框中单击 **删除**，完成删除 OBProxy 集群的操作。

   您可通过弹出框中的 **查看任务** 按钮，查看进度。

   当该任务状态为 **完成**，且 OBProxy 页的 **集群列表** 中已无该 OBPorxy 集群时，则表示删除成功。

# 7 升级 OBProxy 集群下全部 OBProxy

当 OBProxy 有了新的版本时，可以在 OCP 上升级 OBProxy 集群下的全部 OBProxy。

## 7.1 操作步骤

1. 登录 OCP。
2. 在左侧导航栏单击 **OBProxy**。
3. 在集群列表中选择需要操作的 OBProxy 集群，进入 OBProxy 集群 **概览** 页面。
4. 在页面右上角，单击 **...** 图标，选择 **全部升级**。


5. 在弹出框中选择升级版本。


### 7.1.0.1 说明

- 仅可选择高于当前最高版本的软件包。
- 对存在多个 CPU 架构的 OBProxy 集群进行升级时，若某个 **CPU 架构** 中缺少所选择的安装包版本时，将无法进行升级。

6. 单击 **确定**。

您可通过弹出框中的 **查看任务** 按钮，查看升级进度。

当该任务状态为 **完成**，且集群 **概览** 页的 **OBProxy 列表** 中各 OBProxy 的状态为 **运行中** 时，则表示升级成功。

# 8 重启 OBProxy 集群下全部 OBProxy

本文介绍如何在 OCP 上重启 OBProxy 集群下的全部 OBProxy。

## 8.1 前提条件

当前登录 OCP 的用户是 OBPROXY_MANAGER 角色。

## 8.2 注意事项

重启 OBProxy 会出现连接中断，请谨慎操作。

## 8.3 操作步骤

1. 登录 OCP。
2. 在左侧导航栏单击 **OBProxy**。
3. 在集群列表中选择需要操作的 OBProxy 集群，进入 OBProxy 集群 **概览** 页面。
4. 在页面右上角，单击 **...** 图标，选择 **全部重启**。

#### 8.3.0.1 注意

重启 OBProxy 会出现连接中断，请谨慎操作。


5. 在弹出框中单击 **重启**。

   您可通过弹出框中的 **查看任务** 按钮，查看重启进度。

   当该任务状态为 **完成**，且集群 **概览** 页的 **OBProxy 列表** 中，所有 OBProxy 的状态为 **运行中** 时，则表示重启成功。

# 9 管理 OBProxy 连接的 OceanBase 集群

本文介绍如何通过 OCP 为 OBProxy 添加可连接的 OB 集群或删除已连接的 OB 集群。

## 9.1 添加可连接的 OceanBase 集群

1. 登录 OCP。
2. 在左侧导航栏单击 **OBProxy**。
3. 在集群列表中选择需要操作的 OBProxy 集群，进入 OBProxy 集群的 **概览** 页面。
4. 在页面右上角，单击 **添加可连接的 OB 集群**。
5. 在右侧弹出框中，选择可连接的 OceanBase 集群。

   OceanBase 集群的 Proxyro 的密码与 OBProxy 集群的 Proxyro 密码保持一致时，Proxyro 才可访问对应的 OceanBase 集群。当有 OceanBase 集群无法通过 Proxyro 账号的密码连接时，您可通过以下两种方式进行解决：

   - 修改 OBProxy 集群的 Proxyro 密码，详细操作请参见 修改 Proxyro 账号的密码。
   - 修改 OceanBase 集群的 proxyro 用户密码，详细操作请参见 MySQL 租户用户管理。


6. 单击 **确定**，完成添加。

## 9.2 删除已连接的 OceanBase 集群

### 9.2.0.1 注意

删除 OBProxy 集群下已连接的 OceanBase 集群会使当前 OBProxy 集群下的 OBProxy 无法访问该 OB 集群，删除前请确认 OB 集群已不再使用。

1. 登录 OCP。
2. 在左侧导航栏单击 **OBProxy**。
3. 在集群列表中选择需要操作的 OBProxy 集群，进入 OBProxy 集群 **概览** 页面。
4. 在 **可连接 OB 集群** 区域，单击对应集群后的 **删除**。

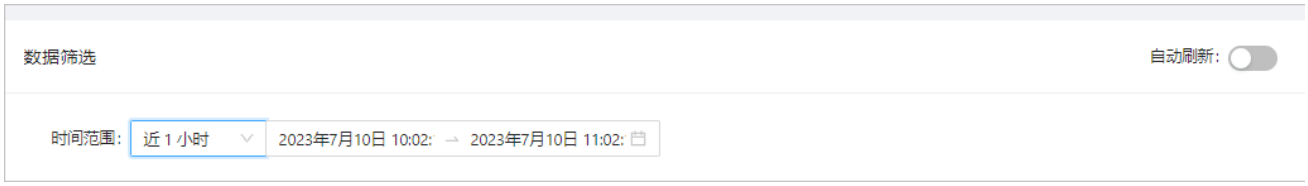

5. 在弹出框中单击 **确定**。

# 10 OBProxy 集群性能监控

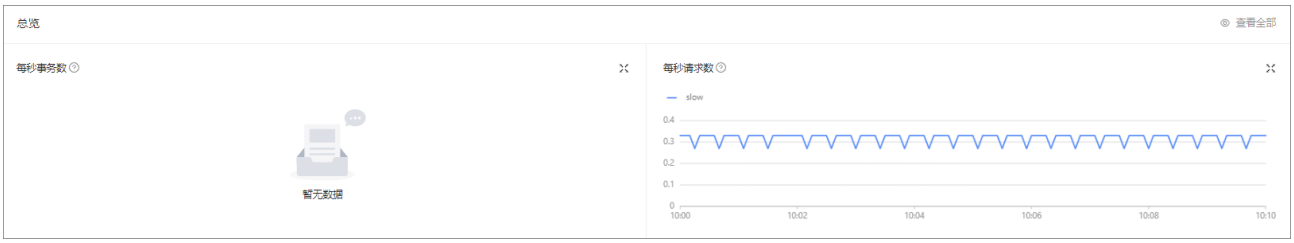
本文介绍如何在 OCP 上查看 OBProxy 集群的性能监控信息，包括服务监控信息和系统监控信息。

## 10.1 服务监控

1. 登录 OCP。
2. 在左侧导航栏单击 **OBProxy**。
3. 在集群列表中选择需要操作的 OBProxy 集群，进入 OBProxy 集群 **性能监控** 页面的 **服务监控** 页签。
   - 在 **服务监控** 页签，您可以查看 **近 5 分钟**、**近 10 分钟**、**近 20 分钟**、**近 30 分钟**、**近 1 小时**、**近 3 小时**、**近 6 小时**、**近 12 小时**或 **自定义时间** 内的监控数据。




   - 单击 **自动刷新** 按钮，系统将每 10 秒进行一次数据更新。
4. 在 **总览** 区域，可以查看当前 OBProxy 集群的性能监控情况，单击右上方 **查看全部**，查看 **每秒事务数**、**每秒请求数**、**客户端连接数**、**服务端链接数**、**平均每条 SQL 响应时间**、**平均每秒路由表查询次数** 和 **平均每次网络请求或响应字节数** 的信息。




5. 在 **连接的 OceanBase 集群数据** 区域，查看当前 OBProxy 集群与其代理的 OceanBase 集群之间的交互性能情况。

   您可查看各指标的最大值、最小值和平均值，单击 **列管理** 选择需要显示的指标信息。
6. 在 **OBProxy IP 数据** 区域，查看当前 OBProxy 集群中的各 OBProxy 节点的性能情况。

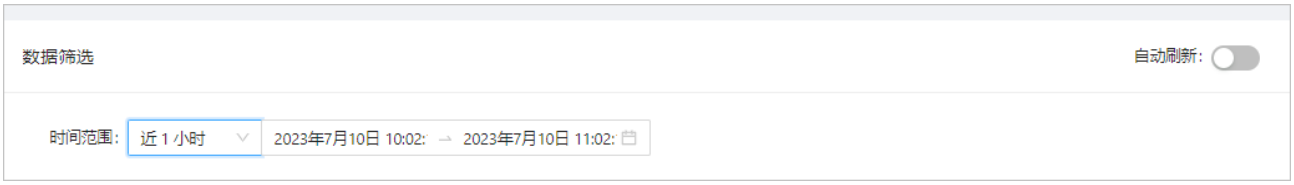
   您可查看各指标的最大值、最小值和平均值，单击 **列管理** 选择需要显示的指标信息。

## 10.2 系统监控

1. 登录 OCP。
2. 在左侧导航栏单击 **OBProxy**。

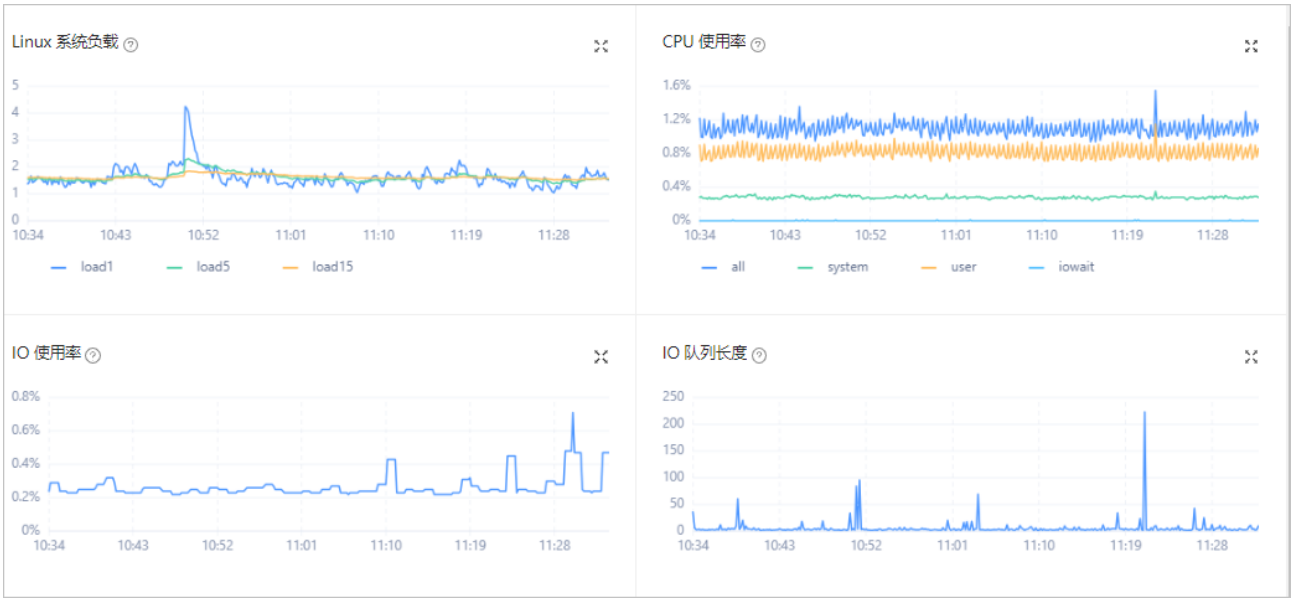
3. 在集群列表中选择需要操作的 OBProxy 集群，进入 OBProxy 集群 **性能监控** 页面的 **系统监控** 页签。

   在 **系统监控** 页签，您可以查看 **近 5 分钟**、**近 10 分钟**、**近 20 分钟**、**近 30 分钟**、**近 1 小时**、**近 3 小时**、**近 6 小时**、**近 12 小时**或 **自定义时间** 内的监控数据。




   也可以单击 **自动刷新** 按钮，查看实时数据更新。

4. 在 **总览** 区域单击右上方 **查看全部**，查看 **Linux 系统负载** 、 **CPU 使用率**、**IO 使用率**、**IO 队列长度**、**平均每秒 IO 次数**、**平均每次 IO 耗时**、**平均每秒 IO 数据量**、**网络吞吐率**、**内存** 和 **磁盘使用率** 信息。




5. 在 **OBProxy IP 数据** 区域，您可查看当前 OBProxy 集群中的各 OBProxy 节点的性能情况，单击 **列管理** 选择需要显示的指标信息。

# 11 添加 OBProxy

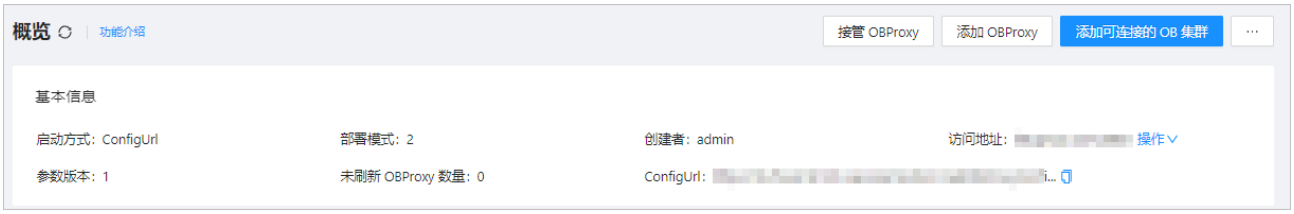
本文介绍如何通过 OCP 添加新的 OBProxy 到已有的 OBProxy 集群。

## 11.1 前提条件

已创建 OBProxy 集群，OBProxy 集群创建相关操作请参见 创建 OBProxy 集群。

## 11.2 操作步骤

1. 登录 OCP。
2. 在左侧导航栏单击 **OBProxy**。
3. 在集群列表中选择需要操作的 OBProxy 集群，进入 OBProxy 集群的 **概览** 页面。
4. 单击 **添加 OBProxy**。




5. 在弹出框中填写 **软件版本**、**SQL 端口**、**Exporter**、**机房**、**机型**、**主机** 等信息。

参考下表填写信息：

| 参数 | 说明 |
| --- | --- |
| SQL 端口 | 默认为 2883。 |
| Exporter 端口 | 默认为 2884。 |
| 软件版本 | 选择要安装的 OBProxy 版本。 |
| 机房、机型（可选）、主机 | 选择部署 OBProxy 所使用的主机，包括该主机的**机房**、**机型（可选）**、**主机**。 |

6. 单击 **确定**。

# 11.2.0.1 说明

- 当 OBProxy 集群为 V4.0 及以上版本时，仅支持选择 V4.0.0 及以上版本的软件包。
- 当 OBProxy 集群为 V3.x 及以下版本时，仅支持选择 V1.8.0 及以上、V4.0.0 以下版本的软件包。
- 当 OBProxy 集群中未创建或接管 OBProxy 时，添加 OBProxy 时可自定义 SQL 端口和 Exporter 端口；当 OBProxy 集群中已存在 OBProxy 时，SQL 端口和 Exporter 端口默认继承创建 OBProxy 集群中部署 OBProxy 时配置的端口号，不支持自定义。

# 12 启动 OBProxy

OBProxy 部署成功后，可以通过命令启动单个 obproxy 进程。

## 12.1 前提条件

请确认已部署 OBProxy，部署 OBProxy 的具体操作请参见 部署 OBProxy。

## 12.2 背景信息

支持通过以下两种方式来启动 obproxy 进程：

- 在启动命令中指定 `-r` 参数来指定 OceanBase 集群的 RootServer 信息。该启动方式不需要额外配置，一般用于开发调试阶段。
- 在启动命令中指定 `obproxy_config_server_url` 参数项来查询获取 OceanBase 集群的 RootServer 信息。该方式需要配置 `obproxy_config_server_url`，故会依赖 Config Server 的启动，建议使用该方式启动 OBProxy。

## 12.3 操作步骤

1. 使用 admin 用户登录到待启动的 OBProxy 所在的机器。

#### 12.3.0.1 注意

启动 OBProxy 时，请使用 admin 用户登录并在 OBProxy 软件的 home 目录下执行启动命令。使用其他用户或者在其他目录下启动 OBProxy 都可能带来问题。

2. 进入 OBProxy 的安装目录。
3. 执行以下命令，启动 OBProxy。

#### 12.3.0.2 说明

在启动 OBProxy 前，您可以通过 `./bin/obproxy -h` 命令来查看 OBProxy 启动相关的参数。

- 在启动命令中指定 `-r` 参数

  命令如下：

  ```
  $./bin/obproxy -p6789 -r'ip:port' -e -n appname -o
  obproxy_config_server_url='' -c cluster_name
  ```

  参数说明：

- `-p` 用于指定 OBProxy 监听的端口号，客户端通过 MySQL 连接该端口访问 OceanBase 数据库。仅在第一次启动时需要指定该参数，后续续日常启动或升级等均不需要添加此参数。
- `-r'ip:port'`：指定的 OceanBase 集群的 Root Server 信息，包括 IP 地址和端口号信息。该 Port 指的是 OBServer 节点的 SQL Port，不是 RPC Port。
- `-e` 用于指定的建表操作，建议仅在 OBProxy 第一次启动时进行，后续日常启动或升级等均不需要添加此参数。
- `-n`（可选）：用于指定待启动的 OBProxy 的应用名。

  OBProxy 的应用名可以通过 `app_name` 配置项来修改，默认为 `undefined`。
- `-o` 用于指定硬件或者内核参数配置。如果不指定，则使用系统默认配置。

  `obproxy_config_server_url=''` 表示无需 Config Server 启动。
- `-c cluster_name` 用于指定 OceanBase 集群。

示例：

```
$./bin/obproxy -r'10.10.10.1:26506;10.10.10.2:26506' -n test -c mycluster
```

- 在启动命令中指定 `obproxy_config_server_url` 参数

  命令如下：

```
$./bin/obproxy -p6789 -e -n appname -o
obproxy_config_server_url='your_config_url'
```

  其中：

  - `-p` 用于指定 OBProxy 监听的端口号，客户端通过 MySQL 连接该端口访问 OceanBase 数据库。仅在第一次启动时需要指定该参数，后续续日常启动或升级等均不需要添加此参数。
  - `-e` 用于指定的建表操作，建议仅在 OBProxy 第一次启动时进行，后续日常启动或升级等均不需要添加此参数。
  - `-n`：用于指定应用名。

    OBProxy 的应用名可以通过 `app_name` 配置项来修改，默认为 `undefined`。
  - `-o` 用于指定硬件或者内核参数配置。如果不指定，则使用系统默认配置。

- `obproxy_config_server_url` 用于指定 Config Server 的访问地址。

  示例：

  ```
  $./bin/obproxy -n test -o obproxy_config_server_url='http://xx.xx.xx.xx:8877/obproxy_config'
  ```

4. 启动后，执行以下命令，查看 obproxy 进程是否存在。

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

# 13 刷新 OBProxy 配置

本文介绍如何刷新 OBProxy 配置。

## 13.1 通过 SQL 语句刷新 OBProxy 配置

您可以通过 `ALTER PROXYCONFIG` 语句刷新单个 OBProxy 的配置。

1. 使用 `root@proxysys` 账号连接到 OBProxy。
2. 执行以下命令，查看待刷新的配置项的值。

   示例如下：

   ```
   obclient> SHOW PROXYCONFIG LIKE 'fetch_proxy_bin_random_time';
   ```

   您也可以通过 `SHOW PROXYCONFIG` 语句查看当前 OBProxy 的所有配置项。
3. 执行以下命令，修改配置项的值。

   ```
   obclient> ALTER PROXYCONFIG SET fetch_proxy_bin_random_time='300s';
   ```

# 14 停止 OBProxy

OBProxy 部署成功后，默认进程为启动状态，可以通过命令停止单个 obproxy 进程。

## 14.1 停止 obproxy 进程

1. 使用 admin 用户登录到 obproxy 进程所在的机器。
2. 执行以下命令，查看 obproxy 进程的进程号。

```
$ps -ef|grep obproxy
admin 37360 0 6 11:35 ? 00:00:09 bin/obproxy
admin 43055 36750 0 11:37 pts/10 00:00:00 grep --color=auto obproxy
root 85623 1 0 Jun02 ? 00:15:19 /home/admin/ocp_agent/obagent/obstat2 -o
http://xx.xx.xx.xx:81 -c test323 __obproxy__ -f 20
```

   查询到 obproxy 的进程号为 37360。
3. 执行以下命令，根据查询到的进程号，停止 obproxy 进程。

```
$kill -9 37360
```

4. 停止成功后，再次执行以下命令，确认 obproxy 进程已不存在。

```
$ps -ef|grep obproxy
admin 45795 36750 0 11:39 pts/10 00:00:00 grep --color=auto obproxy
root 85623 1 0 Jun02 ? 00:15:19 /home/admin/ocp_agent/obagent/obstat2 -o
http://xx.xx.xx.xx:81 -c test323 __obproxy__ -f 20
```

## 14.2 相关内容

- 启动 OBProxy

# 15 重启 OBProxy

本文介绍如何重启 OBProxy。

## 15.1 通过命令重启单个 OBProxy

当 OBProxy 出现异常时，您可以通过命令重启单个 obproxy 进程。

具体操作如下：

1. 使用 admin 用户登录 OBProxy 所在的机器。
2. 停止 obproxy 进程。

   停止 obproxy 进程的具体操作请参见 停止 OBProxy。
3. 启动 obproxy 进程。

   启动 obproxy 进程的具体操作请参见 启动 OBProxy。

# 16 删除 OBProxy

本文介绍如何在 OCP 上删除不再使用的 OBProxy。根据业务需要，可以删除单个/多个 OBProxy。

## 16.1 前提条件

请确保该 OBProxy 已经没有业务连接。

## 16.2 操作步骤

1. 登录 OCP。
2. 在左侧导航栏单击 **OBProxy**。
3. 在集群列表中选择需要操作的 OBProxy 集群，进入 OBProxy 集群 **概览** 页面。
4. 进行如下操作：

### 16.2.0.1 注意

- 若所删除的 OBProxy 为 OBProxy 集群的最后一个 OBProxy，删除后会导致应用无法连接对应数据库。请谨慎操作。
- 删除 OBProxy 后，OBProxy 将从负载均衡策略配置中移除，会对所在的应用产生影响。
- 在 **OBProxy 列表** 中，找到待删除的 OBProxy，在对应的 **操作** 列中，单击 ... 图标，选择 **删除**，并在弹出的对话框中再次单击 **删除**，可以删除单个 OBProxy。


- 在 **OBProxy 列表** 中，选择多个 OBProxy，在列表右上方单击 **...** 图标，选择 **批量删除**，并在弹出的对话框中再次单击 **删除**，可以同时删除多个 OBProxy。


5. 在弹出框中选择 **删除**。

   您可通过弹出框中的 **查看任务** 按钮，查看删除进度。

当该任务状态为 **完成**，且集群 **概览** 页的 **OBProxy 列表** 中不再有该 OBProxy 时，则表示删除成功。

## 16.3 相关内容

有关删除 OBProxy 集群的操作方法，请参见 删除 OBProxy 集群。

# 17 升级 OBProxy

OBProxy 有了新的版本时，可以在 OCP 上升级单个/多个 OBProxy。

## 17.1 前提条件

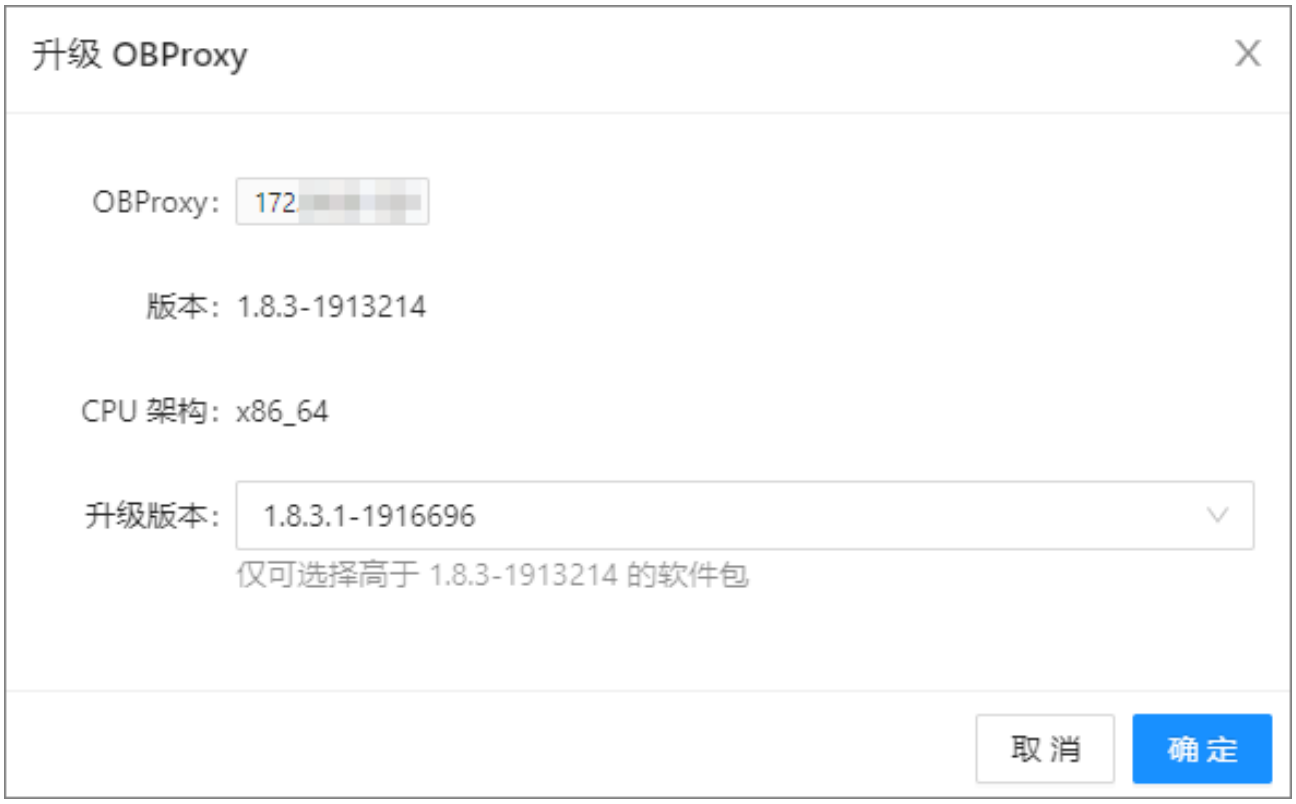
当前登录 OCP 用户具有该 OBProxy 的管理权限。

## 17.2 操作步骤

1. 登录 OCP。
2. 在左侧导航栏单击 **OBProxy**。
3. 在集群列表中选择需要操作的 OBProxy 集群，进入 OBProxy 集群 **概览** 页面。
4. 进行如下操作：
   - 在 **OBProxy 列表** 中，找到待升级的 OBProxy，在对应的 **操作** 列中，单击 **升级**，选择或上传升级版本包，单击 **确定**，可以升级单个 OBProxy。


   - 在 **OBProxy 列表** 中，选择多个 OBProxy，在列表右上方单击 **批量升级**，选择或上传升级版本包，单击 **确定**，可以升级多个 OBProxy。


5. 在弹出框中选择 **升级版本**。




### 17.2.0.1 说明

升级到的软件包版本需高于当前 OBProxy 版本，且软件包硬件架构应与当前主机的硬件架构匹配。

6. 单击 **确定**，完成升级。

   您可通过弹出框中的 **查看任务** 按钮，查看进度。

   当该任务状态为 **完成**，且集群 **概览** 页的 **OBProxy 列表** 中该 OBProxy 的状态为 **运行中** 时，则表示升级成功。

## 17.3 相关信息

升级 OBProxy 集群下所有 OBProxy 的操作方法，请参见 升级 OBProxy 集群下全部 OBProxy。

# 18 OBProxy 参数说明

创建 OBProxy 集群时，可在参数设置中配置启动参数。本文介绍了 OBProxy 启动相关的参数说明。

## 18.1 启动参数

创建 OBProxy 集群时，可参考如下说明配置启动参数。若未配置，系统将按照默认值创建 OBProxy。

| 参数名 | 默认值 | 取值范围 | 说明 |
| --- | --- | --- | --- |
| app_name | dropship | - | OBProxy 服务的应用名。 |
| automatic_match_work_thread | true | • true<br>• false | 判断是否根据 CPU 核数自动创建工作线程。如果该选项为 true，上限为 work_thread_num。 |
| block_thread_num | 1 | [1, 4] | OBProxy 阻塞型任务线程数，用于线程初始化。 |
| enable_cpu_topology | false | • true<br>• false | 是否开启 CPU 亲和，即是否把每个 worker 线程绑定到不同的 CPU 上。 |
| enable_metadb_used | false | • true<br>• false | OBProxy 运行时是否可访问 OCP 的 MetaDB。 |
| enable_strict_kernel_release | false | • true<br>• false | 是否需要校验 OS kernel。<br>取值范围：<br>• true：仅 5u/6u/7u 规格的 RedHat 操作系统支持校验。<br>• false：不校验 OS kernel，但 Proxy 可能不稳定。 |
| frequent_accept | true | • true<br>• false | 是否初始化 net accept 参数。 |
| grpc_client_num | 9 | [9,16] | grpc 客户端数。 |

| | | | |
|---|---|---|---|
| grpc_thread_num | 8 | [8,16] | grpc 线程数。 |
| listen_port | 2883 | (1024, 65536) | OBProxy 的监听端口。 |
| local_bound_ip | 0.0.0.0 | - | OBProxy 的本地 IP。 |
| net_accept_threads | 2 | [0, 8] | 执行 accept 的线程数。 |
| obproxy_config_server_url | - | - | OCP 对外的 configurl 服务地址。 |
| prometheus_cost_ms_unit | false | • true<br>• false | 是否允许 prometheus 的成本单位为毫秒，默认为微秒。 |
| prometheus_listen_port | 2884 | (1024, 65536) | OBProxy prometheus 监听端口。 |
| proxy_id | 0 | [0,255] | OBProxy 的 ID，用于标识每个 OBProxy。当 proxy_service_mode 配置为 server 时，proxy_id 不可配置为 0。 |
| proxy_service_mode | client | • client<br>• server | OBProxy 的部署和服务模式。 |
| rootservice_cluster_name | - | - | Root Service 列表的默认集群名。 |
| rootservice_list | xx.xx.xx.xx:2881 | - | Root Service 列表。 格式为 ip1:sql_port1;ip2:sql_port2 |
| skip_proxy_sys_private_check | true | • true<br>• false | 是否跳过 OBProxy 在私有网段的检查。 |
| stack_size | 1MB | [1MB, 10MB] | 线程栈大小，用于创建线程。 |
| enable_proxy_scramble | true | • true<br>• false | 是否启用 OBProxy 的挑战随机数。 |
| task_thread_num | 2 | [1, 4] | OBProxy 任务线程数。 |
| work_thread_num | 128 | [1, 128] | OBProxy 工作线程数。当 automatic_match_work_thread 为true 时，表示最大工作线程数。 |

# 18.2 相关信息

OBProxy 的参数分为启动参数和其他参数，有关其他参数的详细信息，请参见 配置参数。

# 19 物理连接

本文介绍如何查看当前租户的会话数量和 ODP 上所有网络连接的内部属性状态。

# 19.1 查询当前租户的会话数量及会话 ID

# 19.1.0.1 说明

本文档以 OceanBase 数据库 V4.3.0、ODP V4.2.3 为例介绍查询方法及输出信息。

1. 通过 ODP 连接 OceanBase 数据库，此处以连接 `root@sys` 用户为例。

   ```
   obclient -h10.10.10.1 -P2883 -uroot@sys -p -Doceanbase -A
   ```

2. 通过 `SHOW PROCESSLIST` 语句查询当前租户的会话数量及会话 ID。

   ```
   obclient> SHOW PROCESSLIST;
   ```

   根据 ODP 配置项 `client_session_id_version` 的值不同有如下两种输出结果。

## 19.1.0.2 说明

`client_session_id_version` 配置项用于设置 Client Session ID 生成的计算逻辑，详细介绍可参见 client_session_id_version，推荐设置为 `2`，使用新的计算逻辑以保证 Client Session ID 全局唯一。

- 当 ODP 配置项 `client_session_id_version` 设置为 `2` 时，ODP 会使用新的计算逻辑生成 ID 以保证 Client Session ID 全局唯一，此时输出中显示的是租户对应 OBServer 节点上的会话信息，查询结果如下：

  ```
  +------------+---------+--------------------+-----------+---------+------+--------
  +------------------+
  | Id | User | Host | db | Command | Time | State | Info |
  +------------+---------+--------------------+-----------+---------+------+--------
  +------------------+
  | 1 | root | 10.10.10.1:39512 | oceanbase | Query | 0 | ACTIVE | SHOW PROCESSLIST |
  | 3221501386 | proxyro | 10.10.10.1:37728 | oceanbase | Sleep | 13 | SLEEP | NULL |
  ```

```
+------------+---------+--------------------+----------+---------+------+--------
+------------------+
```

各字段含义如下表所示：

| 字段 | 字段 |
| --- | --- |
| Id | 表示该会话的 ID，即 cs id。 |
| User | 表示该会话所属的用户。 |
| Host | 表示发起该会话的客户端 IP 和端口号。 |
| db | 表示该会话当前连接的数据库名。Oracle 模式显示为与用户名同名的 Schema 名。 |
| Command | 表示该会话正在执行的命令类型。 |
| Time | 表示当前命令执行的时间，单位为秒。如果命令发生了重试，则系统会清零后重新计算。 |
| State | 表示该会话当前的状态。 |
| Info | 表示该会话正在执行的语句。 |

- 当 ODP 配置项 client_session_id_version 设置为 1 时，ODP 会使用原本的计算逻辑生成 ID，此时输出中显示的是对应 ODP 节点上的会话信息，查询结果如下：

```
+------+--------+------+--------------------+----------+-------------
+------------------+------------------+-------+-------+
| Id | Tenant | User | Host | db | trans_count | svr_session_count | state | tid | pid |
+------+--------+------+--------------------+----------+-------------
+------------------+------------------+-------+-------+
| 9 | sys | root | 10.10.10.1:17890 | oceanbase | 0 | 1 | MCS_ACTIVE_READER | 48243
| 48243 |
+------+--------+------+--------------------+----------+-------------
+------------------+------------------+-------+-------+
```

各字段含义如下表所示：

| 字段 | 字段 |
| --- | --- |
| Id | ODP 内部标记每个客户端会话（ Client Session ）的 id 号，即 cs id |
| Tenant | 表示该会话所属的租户 |
| User | 表示该会话所属的用户 |
| Host | 表示发起该会话的客户端 IP 和端口号 |
| db | 表示该会话当前连接的数据库名。Oracle 模式显示为与用户名同名的 Schema 名 |
| trans_count | 表示该会话中 ODP 传输的事务数量 |
| svr_session_count | 会话数量 |
| state | 客户端会话状态，存在如下几个状态：<br>■ MCS_INIT（初始化）<br>■ MCS_ACTIVE_READER（激活）<br>■ MCS_KEEP_ALIVE（保活）<br>■ MCS_HALF_CLOSE（半关闭）<br>■ MCS_CLOSED（已关闭） |
| tid | 线程 ID |
| pid | 进程 ID |

# 19.2 查看 ODP 上所有网络连接的内部属性状态

通过 SHOW PROXYNET CONNECTION 语句查看 ODP 上所有网络连接的内部属性状态。

```
SHOW PROXYNET CONNECTION [thread_id [LIMIT xx]]
```

参数说明：

- 不指定 thread_id 时，展示 ODP 上所有网络连接的内部属性状态。
- 指定 thread_id 时，展示指定 thread 上的连接状态。支持指定 LIMIT [offset,] rows 和 LIMIT rows OFFSET offset 参数，格式与 MySQL 完全兼容。当 rows == -1 时，展示全部行。

示例如下：

```
obclient> SHOW PROXYNET CONNECTION\G
*************************** 1. row ***************************
thread_id: 0
connect_id: 1
socket_fd: 4
type: inner connected
src_ip: XXX.XXX.XXX.XXX
src_port: 43051
dst_ip: XXX.XXX.XXX.XXX
dst_port: 13203
virtual_ip: *Not IP address [0]*
virtual_port: 0
bind_style: any
read_enabled: 1
read_nbytes: 9223372036854775807
read_ndone: 0
write_enabled: 0
write_nbyte: 0
write_ndone: 0
alive_time(sec): 2864
activity_timeout_in(sec): 0
inactivity_timeout_in(sec): 28800
timeout_close_in(sec): 28791
last_error_no: 0
shutdown: 0
comments: not closed
*************************** 2. row ***************************
thread_id: 0
connect_id: 3
```

```
socket_fd: 24
type: accepted
src_ip: XXX.XXX.XXX.XXX
src_port: 48292
dst_ip: XXX.XXX.XXX.XXX
dst_port: 13205
virtual_ip: *Not IP address [0]*
virtual_port: 0
bind_style: any
read_enabled: 1
read_nbytes: 9223372036854775807
read_ndone: 0
write_enabled: 0
write_nbyte: 0
write_ndone: 0
alive_time(sec): 1214
activity_timeout_in(sec): 0
inactivity_timeout_in(sec): 30
timeout_close_in(sec): 29
last_error_no: 0
shutdown: 0
comments: not closed
*************************** 3. row ***************************
thread_id: 0
connect_id: 4
socket_fd: 28
type: connected
src_ip: XXX.XXX.XXX.XXX
src_port: 43154
```

```
dst_ip: XXX.XXX.XXX.XXX
dst_port: 13203
virtual_ip: *Not IP address [0]*
virtual_port: 0
bind_style: any
read_enabled: 1
read_nbytes: 9223372036854775807
read_ndone: 0
write_enabled: 0
write_nbyte: 0
write_ndone: 0
alive_time(sec): 1214
activity_timeout_in(sec): 0
inactivity_timeout_in(sec): 28800
timeout_close_in(sec): 28788
last_error_no: 0
shutdown: 0
comments: not closed
*************************** 4. row ***************************
thread_id: 1
connect_id: 5
socket_fd: 29
type: inner connected
src_ip: XXX.XXX.XXX.XXX
src_port: 43156
dst_ip: XXX.XXX.XXX.XXX
dst_port: 13203
virtual_ip: *Not IP address [0]*
virtual_port: 0
```

```
bind_style: any
read_enabled: 1
read_nbytes: 9223372036854775807
read_ndone: 0
write_enabled: 0
write_nbyte: 0
write_ndone: 0
alive_time(sec): 1194
activity_timeout_in(sec): 0
inactivity_timeout_in(sec): 28800
timeout_close_in(sec): 28791
last_error_no: 0
shutdown: 0
comments: not closed
*************************** 5. row ***************************
thread_id: 3
connect_id: 2
socket_fd: 25
type: inner connected
src_ip: XXX.XXX.XXX.XXX
src_port: 43052
dst_ip: XXX.XXX.XXX.XXX
dst_port: 13203
virtual_ip: *Not IP address [0]*
virtual_port: 0
bind_style: any
read_enabled: 1
read_nbytes: 9223372036854775807
read_ndone: 0
```

```
write_enabled: 0
write_nbyte: 0
write_ndone: 0
alive_time(sec): 2853
activity_timeout_in(sec): 0
inactivity_timeout_in(sec): 28800
timeout_close_in(sec): 28799
last_error_no: 0
shutdown: 0
comments: not closed
5 rows in set

obclient> SHOW PROXYNET CONNECTION 0\G
*************************** 1. row ***************************
thread_id: 0
connect_id: 1
socket_fd: 4
type: inner connected
src_ip: XXX.XXX.XXX.XXX
src_port: 43051
dst_ip: XXX.XXX.XXX.XXX
dst_port: 13203
virtual_ip: *Not IP address [0]*
virtual_port: 0
bind_style: any
read_enabled: 1
read_nbytes: 9223372036854775807
read_ndone: 0
write_enabled: 0
```

```
write_nbyte: 11958
write_ndone: 11958
alive_time(sec): 2919
activity_timeout_in(sec): 0
inactivity_timeout_in(sec): 30
timeout_close_in(sec): 30
last_error_no: 0
shutdown: 0
comments: not closed
*************************** 2. row ***************************
thread_id: 0
connect_id: 3
socket_fd: 24
type: accepted
src_ip: XXX.XXX.XXX.XXX
src_port: 48292
dst_ip: XXX.XXX.XXX.XXX
dst_port: 13205
virtual_ip: *Not IP address [0]*
virtual_port: 0
bind_style: any
read_enabled: 1
read_nbytes: 9223372036854775807
read_ndone: 0
write_enabled: 0
write_nbyte: 0
write_ndone: 0
alive_time(sec): 1269
activity_timeout_in(sec): 0
```

```
inactivity_timeout_in(sec): 30
timeout_close_in(sec): 29
last_error_no: 0
shutdown: 0
comments: not closed
*************************** 3. row ***************************
thread_id: 0
connect_id: 4
socket_fd: 28
type: connected
src_ip: XXX.XXX.XXX.XXX
src_port: 43154
dst_ip: XXX.XXX.XXX.XXX
dst_port: 13203
virtual_ip: *Not IP address [0]*
virtual_port: 0
bind_style: any
read_enabled: 1
read_nbytes: 9223372036854775807
read_ndone: 0
write_enabled: 0
write_nbyte: 0
write_ndone: 0
alive_time(sec): 1269
activity_timeout_in(sec): 0
inactivity_timeout_in(sec): 28800
timeout_close_in(sec): 28733
last_error_no: 0
shutdown: 0
```

```
comments: not closed
3 rows in set

obclient> SHOW PROXYNET CONNECTION 0 limit 2 offset 1\G
*************************** 1. row ***************************
thread_id: 0
connect_id: 3
socket_fd: 24
type: accepted
src_ip: XXX.XXX.XXX.XXX
src_port: 48292
dst_ip: XXX.XXX.XXX.XXX
dst_port: 13205
virtual_ip: *Not IP address [0]*
virtual_port: 0
bind_style: any
read_enabled: 1
read_nbytes: 9223372036854775807
read_ndone: 0
write_enabled: 0
write_nbyte: 0
write_ndone: 0
alive_time(sec): 1299
activity_timeout_in(sec): 0
inactivity_timeout_in(sec): 30
timeout_close_in(sec): 29
last_error_no: 0
shutdown: 0
comments: not closed
```

```
*************************** 2. row ***************************
thread_id: 0
connect_id: 4
socket_fd: 28
type: connected
src_ip: XXX.XXX.XXX.XXX
src_port: 43154
dst_ip: XXX.XXX.XXX.XXX
dst_port: 13203
virtual_ip: *Not IP address [0]*
virtual_port: 0
bind_style: any
read_enabled: 1
read_nbytes: 9223372036854775807
read_ndone: 0
write_enabled: 0
write_nbyte: 0
write_ndone: 0
alive_time(sec): 1299
activity_timeout_in(sec): 0
inactivity_timeout_in(sec): 28800
timeout_close_in(sec): 28703
last_error_no: 0
shutdown: 0
comments: not closed
2 rows in set
```

各字段含义如下表所示：

| 字段 | 字段 |
| --- | --- |
| thread_id | 线程 ID |
| connect_id | 网络连接 ID |
| socket_fd | 套接字描述符 |
| type | 连接状态 |
| src_ip | 源 IP |
| src_port | 源端口 |
| dst_ip | 目的 IP |
| dst_port | 目的端口 |
| virtual_ip | 虚拟 IP |
| virtual_port | 虚拟端口 |
| bind_style | 绑定类型 |
| read_enabled | 是否可读 |
| read_nbytes | 需要读取的数据字节数 |
| read_ndone | 已经读取的数据字节数 |
| write_enabled | 是否可写 |
| write_nbyte | 需要发送的数据字节数 |
| write_ndone | 已经发送的数据字节数 |
| alive_time(sec) | 存活时间 |
| activity_timeout_in(sec) | 活跃超时 |
| inactivity_timeout_in(sec) | 不活跃超时时长 |
| timeout_close_in(sec) | 超时关闭时长 |
| last_error_no | 最近出错的错误号 |
| shutdown | 已断开的网络连接 |
| comments | 注释 |

# 20 展示全部 Session

本文介绍如何查看 ODP 上租户连接的全部 Client Session 的内部状态。

## 20.1 操作步骤

`sys` 租户通过 `SHOW PROXYSESSION` 语句可以查看 ODP 上所有租户连接的全部 Client Session 的内部状态；用户租户通过 `SHOW PROXYSESSION` 语句可以查看 ODP 上当前租户连接的全部 Client Session 的内部状态。

1. 通过 ODP 连接的方式连接 OceanBase 数据库。

   连接示例如下：

   ```
   obclient -h10.xx.xx.xx -uusername@obtenant#obdemo -P2883 -p****** -c -A
   oceanbase
   ```

   有关更加详细的通过 ODP 连接方式连接数据库的操作指引，请参见 通过 OBClient 连接 OceanBase 租户（MySQL 模式） 和 通过 OBClient 连接 OceanBase 租户（Oracle 模式）。

2. 查看 ODP 上所有租户连接的全部 Client Session 的内部状态。

   - `sys` 租户查询所有租户连接的全部 Client Session

     ```
     SHOW PROXYSESSION;
     ```

     查询结果如下：

     ```
     +--------------------+--------+---------+-----------+------+--------------------
     +------+------------+------------------+------------------+-------+-------
     +-----------+
     | proxy_sessid | Id | Cluster | Tenant | User | Host | db | trans_count |
     svr_session_count | state | tid | pid | using_ssl |
     +--------------------+--------+---------+-----------+------+--------------------
     +------+------------+------------------+------------------+-------+-------
     +-----------+
     | 1240249970848713933 2 | 65537 | test420 | sys | root | 100.xx.xx.xx:62630 | NULL |
     0 | 1 | MCS_ACTIVE_READER | 96540 | 96535 | 0 |
     ```

```
| 12402499708487139333 | 131073 | test420 | oracle001 | sys | 100.xx.xx.xx:4722 |
SYS | 0 | 1 | MCS_ACTIVE_READER | 96541 | 96535 | 0 |
| 12402499708487139331 | 1 | test420 | sys | root | 100.xx.xx.xx:25510 | NULL | 0 | 1
| MCS_ACTIVE_READER | 96535 | 96535 | 0 |
+---------------------+--------+---------+-----------+------+---------------------
+------+------------+------------------+------------------+-------+-------
+-----------+
3 rows in set
```

- 用户租户查询当前租户连接的全部 Client Session

```
SHOW PROXYSESSION;
```

查询结果如下:

```
+---------------------+--------+---------+-----------+------+---------------------
+------+------------+------------------+------------------+-------+-------
+-----------+
| proxy_sessid | Id | Cluster | Tenant | User | Host | db | trans_count |
svr_session_count | state | tid | pid | using_ssl |
+---------------------+--------+---------+-----------+------+---------------------
+------+------------+------------------+------------------+-------+-------
+-----------+
| 12402499708487139333 | 131073 | test420 | oracle001 | sys | 100.xx.xx.xx:4722 |
SYS | 0 | 1 | MCS_ACTIVE_READER | 96541 | 96535 | 0 |
+---------------------+--------+---------+-----------+------+---------------------
+------+------------+------------------+------------------+-------+-------
+-----------+
1 row in set
```

各字段含义如下表所示：

| 字段 | 字段 |
| --- | --- |
| proxy_sessid | OceanBase 数据库内标记每个 ODP 的 64 位 ID 号。 |
| Id | ODP 内部标记每个 Client 的 id 号，即 cs id。 |
| Cluster | 集群名 |
| Tenant | 租户名 |
| User | 用户名 |
| Host | 用户 IP 地址和端口号 |
| db | 数据库 |
| trans_count | ODP 会话已传输的事务数量 |
| svr_session_count | ODP 维持的会话数量 |
| state | 网络连接状态 |
| tid | 线程 ID |
| pid | 进程 ID |
| using_ssl | 是否开启了 SSL 协议。0 表示未开启；1 表示开启。 |

# 20.1.0.1 说明

Id 为 ODP 内部标记每个 Client 的 id 号，即 cs id，proxy_sessid 为整个 OceanBase 数据库标记每个 Client 的 64 位 id 号，与标准的 CONNECTION_ID() 不一致。有关 CONNECTION_ID 的详细介绍，请参见 CONNECTION_ID。

# 21 展示 Session 详细状态

本文介绍了如何查看 ODP 上指定 Client Session 的详细内部状态。

## 21.1 操作步骤

`sys` 租户和用户租户可以通过 `SHOW PROXYSESSION ATTRIBUTE` 语句查看指定 Client Session 的详细内部状态，包括该 Client Session 上涉及的相关 Server Session。

1. 通过 ODP 连接的方式连接 OceanBase 数据库。

   连接示例如下：

   ```
   obclient -h10.xx.xx.xx -uusername@obtenant#obdemo -P2883 -p****** -c -A oceanbase
   ```

   有关更加详细的通过 ODP 连接方式连接数据库的操作指引，请参见 通过 OBClient 连接 OceanBase 租户（MySQL 模式） 和 通过 OBClient 连接 OceanBase 租户（Oracle 模式）。

2. 查看指定 Client Session 的详细内部状态。

   SQL 语句如下：

   ```
   SHOW PROXYSESSION ATTRIBUTE [id [LIKE 'xxx']]
   ```

   参数说明：

   - 不指定 `id` 时，显示当前 Session 的详细状态( ODP 1.1.0 版本起开始支持)，支持模糊查询当前 Session 指定属性名称的 value ( proxy 1.1.2 版本起开始支持)。
   - 指定 `id` 时，支持模糊查询指定属性名称的 value ( ODP 1.1.0 版本起开始支持)。
   - `id` 既可以是 `cs_id`，也可以是 `connection_id`，显示结果相同。

     `cs_id` 为 ODP 内部标记的每个 Client 的 `id` 号，`connection_id` 为整个 OceanBase 数据库标记的每个 Client 的 `id` 号。

     MySQL 模式下的 `connection_id` 通过 `SELECT CONNECTION_ID();` 语句获取，Oracle 模式下的 `connection_id` 通过 `SHOW FULL PROCESSLIST;` 语句获取。
   - `LIKE` 模糊匹配，支持 '%' 和 '_'。

   示例如下：

MySQL 模式
Oracle 模式

a. 查看 Client Session 的详细状态。

```
SHOW PROXYSESSION ATTRIBUTE;
```

查询结果如下：

```
+--------------------------------+---------------------+---------------+
| attribute_name | value | info |
+--------------------------------+---------------------+---------------+
| proxy_sessid | -6044239443189891054 | cs common |
| cs_id | 65160 | cs common |
| cluster | test420 | cs common |
| tenant | mysql001 | cs common |
| user | root | cs common |
| host_ip | 100.xx.xx.xx | cs common |
| host_port | 48056 | cs common |
| db | NULL | cs common |
| total_trans_cnt | 0 | cs common |
| svr_session_cnt | 1 | cs common |
| active | true | cs common |
| read_state | MCS_ACTIVE_READER | cs common |
| tid | 76286 | cs common |
| pid | 76286 | cs common |
| idc_name | | cs common |
| modified_time | 0 | cs stat |
| reported_time | 0 | cs stat |
| hot_sys_var_version | 0 | cs var version |
| sys_var_version | 2 | cs var version |
| user_var_version | 0 | cs var version |
| last_insert_id_version | 0 | cs var version |
```

```
| db_name_version | 0 | cs var version |
| server_ip | 172.xx.xx.xx | last used ss |
| server_port | 2881 | last used ss |
| server_sessid | 3221597019 | last used ss |
| ss_id | 22 | last used ss |
| state | MSS_KA_CLIENT_SLAVE | last used ss |
| transact_count | 4 | last used ss |
| server_trans_stat | 0 | last used ss |
| hot_sys_var_version | 0 | last used ss |
| sys_var_version | 2 | last used ss |
| user_var_version | 0 | last used ss |
| last_insert_id_version | 0 | last used ss |
| db_name_version | 0 | last used ss |
| is_checksum_supported | 1 | last used ss |
| is_safe_read_weak_supported | 0 | last used ss |
| is_checksum_switch_supported | 1 | last used ss |
| checksum_switch | 1 | last used ss |
| enable_extra_ok_packet_for_stats | 1 | last used ss |
+---------------------------------+---------------------+----------------+
39 rows in set
```

b. 通过模糊查询，查询指定属性的详细信息。

```
SHOW PROXYSESSION ATTRIBUTE 65160 like '%id%';
```

查询结果如下：

```
+-----------------------+---------------------+---------------+
| attribute_name | value | info |
+-----------------------+---------------------+---------------+
| proxy_sessid | -6044239443189891054 | cs common |
| cs_id | 65160 | cs common |
```

```
| tid | 76286 | cs common |
| pid | 76286 | cs common |
| idc_name | | cs common |
| last_insert_id_version | 0 | cs var version |
| server_sessid | 3221597019 | last used ss |
| ss_id | 22 | last used ss |
| last_insert_id_version | 0 | last used ss |
+------------------------+---------------------+---------------+
9 rows in set
```

a. 查看 Client Session 的详细状态。

```
SHOW PROXYSESSION ATTRIBUTE;
```

查询结果如下：

```
+---------------------------------+---------------------+---------------+
| attribute_name | value | info |
+---------------------------------+---------------------+---------------+
| proxy_sessid | -6044239443189891058 | cs common |
| cs_id | 65141 | cs common |
| cluster | test420 | cs common |
| tenant | oracle001 | cs common |
| user | sys | cs common |
| host_ip | 100.xx.xx.xx | cs common |
| host_port | 59648 | cs common |
| db | SYS | cs common |
| total_trans_cnt | 0 | cs common |
| svr_session_cnt | 1 | cs common |
| active | true | cs common |
| read_state | MCS_ACTIVE_READER | cs common |
| tid | 76286 | cs common |
```

```
| pid | 76286 | cs common |
| idc_name | | cs common |
| modified_time | 0 | cs stat |
| reported_time | 0 | cs stat |
| hot_sys_var_version | 1 | cs var version |
| sys_var_version | 10 | cs var version |
| user_var_version | 0 | cs var version |
| last_insert_id_version | 0 | cs var version |
| db_name_version | 1 | cs var version |
| server_ip | xx.xx.xx.xx | last used ss |
| server_port | 2881 | last used ss |
| server_sessid | 3221583071 | last used ss |
| ss_id | 18 | last used ss |
| state | MSS_KA_CLIENT_SLAVE | last used ss |
| transact_count | 4 | last used ss |
| server_trans_stat | 0 | last used ss |
| hot_sys_var_version | 1 | last used ss |
| sys_var_version | 10 | last used ss |
| user_var_version | 0 | last used ss |
| last_insert_id_version | 0 | last used ss |
| db_name_version | 1 | last used ss |
| is_checksum_supported | 1 | last used ss |
| is_safe_read_weak_supported | 0 | last used ss |
| is_checksum_switch_supported | 1 | last used ss |
| checksum_switch | 1 | last used ss |
| enable_extra_ok_packet_for_stats | 1 | last used ss |
+---------------------------------+---------------------+---------------+
39 rows in set
```

b. 通过模糊查询，查询指定属性的详细信息。

```
SHOW PROXYSESSION ATTRIBUTE 65141 LIKE '%id%';
```

查询结果如下：

```
+----------------------+---------------------+---------------+
| attribute_name | value | info |
+----------------------+---------------------+---------------+
| proxy_sessid | -6044239443189891058 | cs common |
| cs_id | 65141 | cs common |
| tid | 76286 | cs common |
| pid | 76286 | cs common |
| idc_name | | cs common |
| last_insert_id_version | 0 | cs var version |
| server_sessid | 3221583071 | last used ss |
| ss_id | 18 | last used ss |
| last_insert_id_version | 0 | last used ss |
+----------------------+---------------------+---------------+
9 rows in set
```

各字段含义如下表所示：

| 字段 | 说明 |
| --- | --- |
| attribute_name | 属性名称 |
| value | 属性值 |
| info | 基本信息 |

常见的属性及其说明如下表所示：

| 字段 | 字段 |
| --- | --- |
| proxy_sessid | ODP 的会话 ID 号 |
| cs_id | ODP 内部标记每个 Client 的 id 号 |
| cluster | Client Session 所属集群名 |
| tenant | 租户 |

| | |
|---|---|
| user | 用户 |
| host_ip | 用户 IP |
| host_port | 用户端口号 |
| db | 数据库 |
| total_trans_cnt | 传输事务的总数量 |
| svr_session_cnt | 会话总数量 |
| active | 是否存活 |
| read_state | 客户端会话的状态 |
| tid | 线程 ID |
| pid | 进程 ID |
| modified_time | 历史修改时间 |
| reported_time | 历史报告时间 |
| hot_sys_var_version | 热更新的系统变量版本 |
| sys_var_version | 系统变量版本 |
| user_var_version | 用户变量版本 |
| last_insert_id_version | 最后插入 ID 版本 |
| db_name_version | 数据库名的版本 |
| server_ip | OBServer 节点的 IP 地址 |
| server_port | OBServer 节点的端口号 |
| server_sessid | OBServer 节点的会话 ID 号 |
| ss_id | ODP 标记 Server Session 的 ID 号 |
| state | 网络连接状态 |
| transact_count | 当前 Server Session 执行事务的数量 |
| server_trans_stat | 当前 Server Session 的执行状态，主要用于验证当前 Server Session 的请求执行情况是否准确 |
| hot_sys_var_version | ODP 保存的常用系统变量的版本 |
| sys_var_version | ODP 保存的系统变量的版本 |
| user_var_version | ODP 保存的用户系统变量的版本 |
| last_insert_id_version | ODP 保存的 last_insert_id 变量的版本 |
| db_name_version | ODP 保存的当前数据库变量的版本 |
| is_checksum_supported | 当前 Server Sesion 是否开启了校验和 |

| | |
|---|---|
| is_safe_read_weak_supported | 是否支持安全弱读 |
| is_checksum_switch_supported | 校验和开关是否支持实时开启或关闭 |
| checksum_switch | 校验和的开关状态 |
| enable_extra_ok_packet_for_stats | 是否支持额外的返回状态确认的数据包 |

# 22 展示 Session 统计项

本文介绍了如何查看 ODP 上指定 Client Session 的内部统计项（包括：SQL 请求响应数量、SQL 请求响应大小等）。

## 22.1 操作步骤

`sys` 租户和用户租户通过 `SHOW PROXYSESSION STAT` 语句查看 ODP 指定 Client Session 的内部统计项。

1. 通过 ODP 连接的方式连接 OceanBase 数据库。

   连接示例如下：

   ```
   obclient -h10.xx.xx.xx -uusername@obtenant#obdemo -P2883 -p****** -c -A oceanbase
   ```

   有关更加详细的通过 ODP 连接方式连接数据库的操作指引，请参见 通过 OBClient 连接 OceanBase 租户（MySQL 模式） 和 通过 OBClient 连接 OceanBase 租户（Oracle 模式）。

2. 查看 ODP 指定 Client Session 的内部统计项。

   ```
   SHOW PROXYSESSION STAT id LIKE 'xx';
   ```

   参数说明：

   - `id` 既可以是 `cs_id`，也可以是 `connection_id`，显示结果相同。

     `cs_id` 为 ODP 内部标记的每个 Client 的 `id` 号，`connection_id` 为整个 OceanBase 数据库标记的每个 Client 的 `id` 号。

     MySQL 模式下的 `connection_id` 通过 `SELECT CONNECTION_ID();` 语句获取，Oracle 模式下的 `connection_id` 通过 `SHOW FULL PROCESSLIST;` 语句获取。

   - `LIKE` 表示模糊匹配，支持 '%' 和 '_'。

   示例如下：

   MySQL 模式
   Oracle 模式

   a. 获取 `cs_id` 或 `connection_id`。

      执行以下语句获取 `cs_id`。

```
SHOW PROXYSESSION;
```

返回如下所示结果，返回结果中的 Id 即为 cs_id 。

```
+---------------------+-------+----------+----------+------+---------------------
+------+-------------+------------------+------------------+-------+-------
+-----------+
| proxy_sessid | Id | Cluster | Tenant | User | Host | db | trans_count |
svr_session_count | state | tid | pid | using_ssl |
+---------------------+-------+----------+----------+------+---------------------
+------+-------------+------------------+------------------+-------+-------
+-----------+
| 12402504630519660562 | 11 | test420 | mysql001 | root | 100.xx.xx.xx:48056 |
NULL | 0 | 1 | MCS_ACTIVE_READER | 76286 | 76286 | 0 |
+---------------------+-------+----------+----------+------+---------------------
+------+-------------+------------------+------------------+-------+-------
+-----------+
1 row in set
```

执行以下语句获取 connection_id 。

```
SELECT CONNECTION_ID();
```

返回如下所示结果。

```
+-----------------+
| CONNECTION_ID() |
+-----------------+
| 3221597019 |
+-----------------+
1 row in set
```

b. 查看内部统计项。

```
SHOW PROXYSESSION STAT 11 LIKE '%time%';
```

或者

```
SHOW PROXYSESSION STAT 3221597019 LIKE '%time%';
```

返回结果如下所示：

```
+----------------------------------------+-------+
| stat_name | value |
+----------------------------------------+-------+
| total_transactions_time | 0 |
| total_user_transactions_time | 0 |
| total_client_request_read_time | 0 |
| total_client_response_write_time | 0 |
| total_client_request_analyze_time | 0 |
| total_client_transaction_idle_time | 0 |
| total_ok_packet_trim_time | 0 |
| total_server_process_request_time | 0 |
| total_server_response_read_time | 0 |
| total_server_response_analyze_time | 0 |
| total_send_saved_login_time | 0 |
| total_send_all_session_vars_time | 0 |
| total_send_use_database_time | 0 |
| total_send_changed_session_vars_time | 0 |
| total_send_changed_session_user_vars_time | 0 |
| total_send_last_insert_id_time | 0 |
| total_send_start_trans_time | 0 |
| total_pl_lookup_time | 0 |
| total_congestion_control_time | 0 |
| total_server_connect_time | 0 |
```

```
+-----------------------------------------+-------+
20 rows in set
```

a. 获取 cs_id 或 connection_id 。

执行以下语句获取 cs_id 。

```
SHOW PROXYSESSION;
```

返回如下所示结果，返回结果中的 Id 即为 cs_id 。

```
+---------------------+-------+----------+-----------+------
+---------------------+------+------------+------------------
+-------------------+-------+-------+-----------+
| proxy_sessid | Id | Cluster | Tenant | User | Host | db | trans_count |
svr_session_count | state | tid | pid | using_ssl |
+---------------------+-------+----------+-----------+------
+---------------------+------+------------+------------------
+-------------------+-------+-------+-----------+
| 12402504630519660558 | 65543 | test420 | oracle001 | sys | 100.xx.xx.xx:59648
| SYS | 0 | 1 | MCS_ACTIVE_READER | 76286 | 76286 | 0 |
+---------------------+-------+----------+-----------+------
+---------------------+------+------------+------------------
+-------------------+-------+-------+-----------+
1 row in set
```

执行以下语句获取 connection_id 。

```
SHOW FULL PROCESSLIST;
```

返回如下所示结果，返回结果中的 id 即为 connection_id 。

```
+------------+------+-----------+--------------------+------+---------+------
+--------+----------------------+--------------+------+
```

```
| ID | USER | TENANT | HOST | DB | COMMAND | TIME | STATE | INFO | IP | PORT |
+------------+------+----------+--------------------+------+---------+------
+--------+----------------------+--------------+------+
| 3221552558 | SYS | oracle001 | 172.xx.xx.xx:45343 | SYS | Query | 0 | ACTIVE |
SHOW FULL PROCESSLIST | 172.xx.xx.64 | 2881 |
+------------+------+----------+--------------------+------+---------+------
+--------+----------------------+--------------+------+
1 row in set
```

b. 查看内部统计项。

```
SHOW PROXYSESSION STAT 65543 LIKE '%time%';
```

或者

```
SHOW PROXYSESSION STAT 3221552558 LIKE '%time%';
```

返回结果如下所示：

```
+-----------------------------------------+-------+
| stat_name | value |
+-----------------------------------------+-------+
| total_transactions_time | 0 |
| total_user_transactions_time | 0 |
| total_client_request_read_time | 0 |
| total_client_response_write_time | 0 |
| total_client_request_analyze_time | 0 |
| total_client_transaction_idle_time | 0 |
| total_ok_packet_trim_time | 0 |
| total_server_process_request_time | 0 |
| total_server_response_read_time | 0 |
| total_server_response_analyze_time | 0 |
| total_send_saved_login_time | 0 |
```

```
| total_send_all_session_vars_time | 0 |
| total_send_use_database_time | 0 |
| total_send_changed_session_vars_time | 0 |
| total_send_changed_session_user_vars_time | 0 |
| total_send_last_insert_id_time | 0 |
| total_send_start_trans_time | 0 |
| total_pl_lookup_time | 0 |
| total_congestion_control_time | 0 |
| total_server_connect_time | 0 |
+-----------------------------------------+-------+
20 rows in set
```

各字段含义如下表所示：

| 字段 | 说明 |
| --- | --- |
| stat_name | 内部统计项的名称 |
| value | 内部统计项的取值 |

内部统计项及其说明说下：

| 字段 | 说明 |
| --- | --- |
| total_transaction_count | 事务总数 |
| total_user_transaction_count | ODP 用户事务执行的数量 |
| total_query_count | 用户执行请求的数量 |
| total_client_request_reread_count | ODP 读取用户请求的次数 |
| total_server_response_reread_count | ODP 读取 OBServer 节点返回数据的次数 |
| client_request_total_size | 客户端请求数据量总大小 |
| client_response_total_size | 客户端返回数据量总大小 |
| server_request_total_size | 发送给 OBServer 节点的请求数据量总大小 |
| server_response_total_size | OBServer 节点返回的数据量总大小 |
| total_transactions_time | 事务执行总耗时 |
| total_user_transactions_time | 用户执行事务的总时间 |
| total_client_request_read_time | 客户端请求读取总时间 |

| | |
|---|---|
| total_client_response_write_time | 返回客户端数据发送总时间 |
| total_client_request_analyze_time | 解析客户端报文的总时间 |
| total_client_transaction_idle_time | 用户事务执行中空闲的总时间 |
| total_server_process_request_time | 数据库处理请求总时间 |
| total_server_response_read_time | 读取 OBServer 节点返回数据的总时间 |
| total_server_response_analyze_time | ODP 解析该用户 Server 返回数据的总时间 |
| total_send_saved_login_time | 发送给 OBServer 节点 LOGIN 报文的总时间 |
| total_send_all_session_vars_time | 发送给 OBServer 节点 Session 变量同步总时间 |
| total_send_use_database_time | 发送给 OBServer 节点 use database 命令总时间 |
| total_send_changed_session_vars_time | 发送给 OBServer 节点 Session 变量修改总时间 |
| total_send_last_insert_id_time | 发送给 OBServer 节点 last_insert_id 同步总时间 |
| total_send_start_trans_time | 发送给 OBServer 节点开启事务同步总时间 |
| total_pl_lookup_time | 路由总时间 |
| total_server_connect_time | ODP 和 OBServer 节点建立连接的总时间 |
| client_requests | 总请求数 |
| client_large_requests | 大请求数 |
| client_internal_requests | 内部请求数 |
| local_session_state_requests | ODP 内部请求的数量 |
| client_missing_pk_requests | 统计本次请求不需要做路由处理，需要使用上个请求的连接的请求数量 |
| client_completed_requests | 已完成的请求数 |
| client_connection_abort_count | 连接及中止统计 |
| client_select_requests | SELECT 请求总数 |
| client_insert_requests | INSERT 请求总数 |
| client_update_requests | UPDATE 请求总数 |
| client_delete_requests | DELETE 请求总数 |
| client_other_requests | 其它请求总数 |
| request_size_100_count | 请求大小在 [0,100) 之间的数量 |
| request_size_1K_count | 请求大小在 [100,1K) 之间的数量 |
| request_size_3K_count | 请求大小在 [1K,3K) 之间的数量 |

| | |
|---|---|
| request_size_5K_count | 请求大小在 [3K,5K) 之间的数量 |
| request_size_10K_count | 请求大小在 [5K,10K) 之间的数量 |
| request_size_1M_count | 请求大小在 [10K,1M) 之间的数量 |
| request_size_inf_count | 请求大小在 [1M,+∞) 之间的数量 |
| response_size_100_count | 响应大小在 [0,100) 之间的数量 |
| response_size_1K_count | 响应大小在 [100,1K) 之间的数量 |
| response_size_3K_count | 响应大小在 [1K,3K) 之间的数量 |
| response_size_5K_count | 响应大小在 [3K,5K) 之间的数量 |
| response_size_10K_count | 响应大小在 [5K,10K) 之间的数量 |
| response_size_1M_count | 响应大小在 [10K,1M) 之间的数量 |
| response_size_inf_count | 响应大小在 [1M,+∞) 之间的数量 |
| client_speed_bytes_per_sec_100 | 每秒的请求速度在 [0,100) 之间的数量 |
| client_speed_bytes_per_sec_1K | 每秒的请求速度在 [100,1K) 之间的数量 |
| client_speed_bytes_per_sec_10K | 每秒的请求速度在 [1K,10K) 之间的数量 |
| client_speed_bytes_per_sec_100K | 每秒的请求速度在 [10K,100K) 之间的数量 |
| client_speed_bytes_per_sec_1M | 每秒的请求速度在 [100K,1M) 之间的数量 |
| client_speed_bytes_per_sec_10M | 每秒的请求速度在 [1M,10M） 之间的数量 |
| client_speed_bytes_per_sec_100M | 每秒的请求速度在 [10 M,+∞) 之间的数量 |
| server_speed_bytes_per_sec_100 | 每秒的响应速度在 [0,100) 之间的数量 |
| server_speed_bytes_per_sec_1K | 每秒的响应速度在 [100,1K) 之间的数量 |
| server_speed_bytes_per_sec_10K | 每秒的响应速度在 [1K,10K) 之间的数量 |
| server_speed_bytes_per_sec_100K | 每秒的响应速度在 [10K,100K) 之间的数量 |
| server_speed_bytes_per_sec_1M | 每秒的响应速度在 [100K,1M) 之间的数量 |
| server_speed_bytes_per_sec_10M | 每秒的响应速度在 [1M,10M） 之间的数量 |
| server_speed_bytes_per_sec_100M | 每秒的响应速度在 [10 M,+∞) 之间的数量 |
| server_connect_count | 连接 OBServer 节点的总次数 |
| server_connect_retries | 连接 OBServer 节点的重试总次数 |
| server_pl_lookup_count | 路由总次数 |
| server_pl_lookup_retries | 路由重试总次数 |
| broken_server_connections | 和 OBServer 节点断连总次数 |
| server_requests | 向 OBServer 节点发送请求总次数 |

| | |
|---|---|
| server_responses | 收到 OBServer 节点响应总次数 |
| server_error_responses | 收到 OBServer 节点 ERROR 类型响应总次数 |
| server_resultset_responses | 收到 OBServer 节点 resultset 类型响应总次数 |
| server_ok_responses | 收到 OBServer 节点 OK 类型响应总次数 |
| server_other_responses | 收到 OBServer 节点 其它类型响应总次数 |
| send_saved_login_requests | 发送给 OBServer 节点 LOGIN 报文总次数 |
| send_all_session_vars_requests | 发送给 OBServer 节点 全量 Session 变量同步总次数 |
| send_use_database_requests | 发送给 OBServer 节点 use database 总次数 |
| send_changed_session_vars_requests | 发送给 OBServer 节点部分 Session 变量总次数 |
| send_last_insert_id_requests | 发送给 OBServer 节点 last insert id 总次数 |
| send_start_trans_requests | 发送给 OBServer 节点开启事务总次数 |
| vip_to_tenant_cache_hit | 公有云上 VIP 信息查询命中次数 |
| vip_to_tenant_cache_miss | 公有云上 VIP 信息查询未命中次数 |

# 23 展示 Session 变量

Session 变量分为系统变量和用户变量。本文介绍了如何查看 ODP 上指定 Client Session 的 Session 变量。

## 23.1 操作步骤

`sys` 租户和用户租户可以通过 `SHOW PROXYSESSION VARIABLES` 语句查看指定 Client Session 的 Session 变量。

1. 通过 ODP 连接的方式连接 OceanBase 数据库。

   连接示例如下：

   ```
   obclient -h10.xx.xx.xx -uusername@obtenant#obdemo -P2883 -p****** -c -A oceanbase
   ```

   有关更加详细的通过 ODP 连接方式连接数据库的操作指引，请参见 通过 OBClient 连接 OceanBase 租户（MySQL 模式） 和 通过 OBClient 连接 OceanBase 租户（Oracle 模式）。

2. 查看指定 Client Session 的 Session 变量。

   ```
   SHOW PROXYSESSION VARIABLES [ALL] id [LIKE 'xx'];
   ```

   参数说明：

   - 不带 `ALL` 参数时，展示指定 Client Session 的本地 Session 变量 (包括：修改过的系统变量和用户变量)。
   - 带 `ALL` 参数时，展示指定 Client Session 的全部 Session 变量 (包括：所有系统变量和用户变量)。
   - `id` 既可以是 `cs_id`，也可以是 `connection_id`，显示结果相同。

     `cs_id` 为 ODP 内部标记的每个 client 的 `id` 号，`connection_id` 为整个 OceanBase 数据库标记的每个 client 的 `id` 号。

     MySQL 模式下的 `connection_id` 通过 `SELECT CONNECTION_ID();` 语句获取，Oracle 模式下的 `connection_id` 通过 `SHOW FULL PROCESSLIST;` 语句获取。
   - `like` 模糊匹配，支持 '%' 和 '_'。

   示例如下：

MySQL 模式
Oracle 模式

a. 获取 cs_id 或 connection_id 。

执行以下语句获取 cs_id 。

```
SHOW PROXYSESSION;
```

查询结果如下，返回结果中的 Id 即为 cs_id 。

```
+---------------------+-------+----------+----------+------+---------------------
+------+-------------+------------------+------------------+-------+-------
+-----------+
| proxy_sessid | Id | Cluster | Tenant | User | Host | db | trans_count |
svr_session_count | state | tid | pid | using_ssl |
+---------------------+-------+----------+----------+------+---------------------
+------+-------------+------------------+------------------+-------+-------
+-----------+
| 12402504630519660559 | 65149 | test420 | mysql001 | root | 100.xx.xx.xx:
46069 | NULL | 0 | 1 | MCS_ACTIVE_READER | 76286 | 76286 | 0 |
+---------------------+-------+----------+----------+------+---------------------
+------+-------------+------------------+------------------+-------+-------
+-----------+
1 row in set
```

或者，执行以下语句获取 connection_id 。

```
SELECT CONNECTION_ID();
```

返回如下所示结果。

```
+-----------------+
| CONNECTION_ID() |
+-----------------+
| 3221593149 |
```

```
+-----------------+
1 row in set
```

b. 查看指定 Client Session 的本地 Session 变量。

```
SHOW PROXYSESSION VARIABLES 65149;
```

或者

```
SHOW PROXYSESSION VARIABLES 3221593149;
```

查询结果如下：

```
+----------------------------------+-------------+----------------
+--------------------+-----------------------------------------------+
| variable_name | value | info | modified_type | sys_variable_flag |
+----------------------------------+-------------+----------------
+--------------------+-----------------------------------------------+
| ob_proxy_global_variables_version | 0 | changed sys var | cold modified vars |
&& invisible && session_scope && readonly |
| ob_proxy_user_privilege | 133009965054 | changed sys var | cold modified
vars | && invisible && session_scope && readonly |
| ob_capability_flag | 916303 | changed sys var | cold modified vars | &&
invisible && session_scope && readonly |
| ob_enable_transmission_checksum | 1 | changed sys var | cold modified vars |
&& global_scope && session_scope |
| character_set_database | 45 | changed sys var | cold modified vars | &&
global_scope && session_scope && nullable |
| collation_database | 45 | changed sys var | cold modified vars | &&
global_scope && session_scope && nullable |
| _min_cluster_version | '4.2.0.0' | user var | cold modified vars | |
+----------------------------------+-------------+----------------
```

```
+--------------------+---------------------------------------------+
7 rows in set
```

c. 查看指定 Client Session 的所有 Session 变量。

```
SHOW PROXYSESSION VARIABLES ALL 65149 LIKE '%id%';
```

或者

```
SHOW PROXYSESSION VARIABLES ALL 3221593149 LIKE '%id%';
```

查询结果如下：

```
+-------------------------------------+-----------------------------------------
+---------+-----------------------------
+-----------------------------------------+
| variable_name | value | info | modified_type | sys_variable_flag |
+-------------------------------------+-----------------------------------------
+---------+-----------------------------
+-----------------------------------------+
| last_insert_id | 0 | sys var | last insert id modified vars | && session_scope |
| identity | 0 | sys var | cold modified vars | && session_scope |
| server_id | 1 | sys var | cold modified vars | &&global_scope |
| query_cache_wlock_invalidate | 0 | sys var | cold modified vars | &&
global_scope && session_scope |
| server_uuid | 'd72b5d0d-2c50-11ee-b1de-2646ab385e11' | sys var | cold
modified vars | && global_scope && readonly |
| ob_org_cluster_id | 0 | sys var | cold modified vars | && session_scope |
| ob_statement_trace_id | 'Y0-0' | sys var | cold modified vars | && invisible &&
session_scope && readonly |
| ob_trx_idle_timeout | 86400000000 | sys var | cold modified vars | &&
global_scope && session_scope |
| tracefile_identifier | '' | sys var | cold modified vars | && session_scope |
```

```
| validate_password_check_user_name | 0 | sys var | cold modified vars | &&
global_scope |
| validate_password_length | 0 | sys var | cold modified vars | && global_scope |
| validate_password_mixed_case_count | 0 | sys var | cold modified vars | &&
global_scope |
| validate_password_number_count | 0 | sys var | cold modified vars | &&
global_scope |
| validate_password_policy | 0 | sys var | cold modified vars | && global_scope |
| validate_password_special_char_count | 0 | sys var | cold modified vars | &&
global_scope |
+-------------------------------------+-------------------------------------
+---------+-----------------------------
+-----------------------------------------+
15 rows in set
```

a. 获取 `cs_id` 或 `connection_id` 。

   执行以下语句获取 `cs_id` 。

   ```
   SHOW PROXYSESSION;
   ```

   查询结果如下，返回结果中的 `Id` 即为 `cs_id` 。

   ```
   +---------------------+-------+----------+-----------+------
   +---------------------+------+------------+------------------
   +------------------+-------+-------+-----------+
   | proxy_sessid | Id | Cluster | Tenant | User | Host | db | trans_count |
   svr_session_count | state | tid | pid | using_ssl |
   +---------------------+-------+----------+-----------+------
   +---------------------+------+------------+------------------
   +------------------+-------+-------+-----------+
   | 12402504630519660558 | 65141 | test420 | oracle001 | sys | 100.xx.xx.xx:59648
   ```

```
| SYS | 0 | 1 | MCS_ACTIVE_READER | 76286 | 76286 | 0 |
+---------------------+-------+----------+-----------+------
+---------------------+------+------------+------------------
+------------------+-------+-------+-----------+
1 row in set
```

执行以下语句获取 connection_id 。

```
SHOW FULL PROCESSLIST;
```

返回如下所示结果，返回结果中的 id 即为 connection_id 。

```
+------------+------+-----------+--------------------+------+---------+------
+--------+----------------------+--------------+------+
| ID | USER | TENANT | HOST | DB | COMMAND | TIME | STATE | INFO | IP | PORT |
+------------+------+-----------+--------------------+------+---------+------
+--------+----------------------+--------------+------+
| 3221583071 | SYS | oracle001 | 172.xx.xx.xx:45343 | SYS | Query | 0 | ACTIVE |
SHOW FULL PROCESSLIST | 172.xx.xx.64 | 2881 |
+------------+------+-----------+--------------------+------+---------+------
+--------+----------------------+--------------+------+
1 row in set
```

b. 查看指定 Client Session 的本地 Session 变量。

```
SHOW PROXYSESSION VARIABLES 65141;
```

或者

```
SHOW PROXYSESSION VARIABLES 3221583071;
```

查询结果如下：

```
+---------------------------------+-------------+----------------
+-------------------+-----------------------------------------------+
```

```
| variable_name | value | info | modified_type | sys_variable_flag |
+---------------------------------+-------------+----------------
+------------------+--------------------------------------------+
| autocommit | 0 | changed sys var | hot modified vars | && global_scope &&
session_scope |
| character_set_connection | 46 | changed sys var | cold modified vars | &&
global_scope && session_scope && nullable |
| character_set_database | 46 | changed sys var | cold modified vars | &&
global_scope && session_scope && nullable |
| character_set_server | 46 | changed sys var | cold modified vars | &&
global_scope && session_scope && nullable |
| collation_connection | 46 | changed sys var | cold modified vars | &&
global_scope && session_scope && nullable |
| collation_database | 46 | changed sys var | cold modified vars | &&
global_scope && session_scope && nullable |
| collation_server | 46 | changed sys var | cold modified vars | && global_scope
&& session_scope && nullable |
| sql_mode | 2151677954 | changed sys var | cold modified vars | &&
global_scope && session_scope |
| group_concat_max_len | 32767 | changed sys var | cold modified vars | &&
global_scope && session_scope |
| ob_proxy_global_variables_version | 0 | changed sys var | cold modified vars |
&& invisible && session_scope && readonly |
| ob_proxy_user_privilege | 133009965054 | changed sys var | cold modified
vars | && invisible && session_scope && readonly |
| ob_capability_flag | 916303 | changed sys var | cold modified vars | &&
invisible && session_scope && readonly |
| ob_enable_transmission_checksum | 1 | changed sys var | cold modified vars |
&& global_scope && session_scope |
```

```
| _min_cluster_version | '4.2.0.0' | user var | cold modified vars | |
+----------------------------------+-------------+----------------
+------------------+--------------------------------------------+
14 rows in set
```

c. 查看指定 Client Session 的所有 Session 变量。

```
SHOW PROXYSESSION VARIABLES ALL 65141 LIKE '%id%';
```

或者

```
SHOW PROXYSESSION VARIABLES ALL 3221583071 LIKE '%id%';
```

查询结果如下：

```
+-------------------------------------+---------------------------------------
+---------+-----------------------------
+-------------------------------------------+
| variable_name | value | info | modified_type | sys_variable_flag |
+-------------------------------------+---------------------------------------
+---------+-----------------------------
+-------------------------------------------+
| last_insert_id | 0 | sys var | last insert id modified vars | && session_scope |
| identity | 0 | sys var | cold modified vars | && session_scope |
| server_id | 1 | sys var | cold modified vars | &&global_scope |
| query_cache_wlock_invalidate | 0 | sys var | cold modified vars | &&
global_scope && session_scope |
| server_uuid | 'd72b5d0d-2c50-11ee-b1de-2646ab385e11' | sys var | cold
modified vars | && global_scope && readonly |
| ob_org_cluster_id | 0 | sys var | cold modified vars | && session_scope |
| ob_statement_trace_id | 'Y0-0' | sys var | cold modified vars | && invisible &&
session_scope && readonly |
| ob_trx_idle_timeout | 86400000000 | sys var | cold modified vars | &&
```

```
global_scope && session_scope |
| tracefile_identifier | '' | sys var | cold modified vars | && session_scope |
| validate_password_check_user_name | 0 | sys var | cold modified vars | &&
global_scope |
| validate_password_length | 0 | sys var | cold modified vars | && global_scope |
| validate_password_mixed_case_count | 0 | sys var | cold modified vars | &&
global_scope |
| validate_password_number_count | 0 | sys var | cold modified vars | &&
global_scope |
| validate_password_policy | 0 | sys var | cold modified vars | && global_scope |
| validate_password_special_char_count | 0 | sys var | cold modified vars | &&
global_scope |
+-------------------------------------+----------------------------------------
+---------+-----------------------------
+-------------------------------------------+
15 rows in set
```

各字段含义如下表所示：

| 字段 | 说明 |
| --- | --- |
| variable_name | 变量名 |
| value | 变量值 |
| info | 变量类型（用户变量或系统变量） |
| modified_type | 变量类型（根据修改频率区分） |
| sys_variable_flag | 系统变量范围 |

# 23.2 相关文档

有关系统变量的详细介绍请参见 配置项和系统变量概述。

# 24 终止 Server Session

本文介绍了如何终止 ODP 上指定 Client Session 以及指定 Client Session 上的 Server Session。

## 24.1 操作步骤

`sys` 租户可以通过 `KILL PROXYSESSION` 语句终止指定的 Session。

1. 使用 `root` 用户通过 ODP 连接的方式登录集群的 `sys` 租户。

   连接示例如下：

   ```
   obclient -h10.xx.xx.xx -uroot@sys#obdemo -P2883 -p****** -c -A oceanbase
   ```

   有关更加详细的通过 ODP 连接方式连接数据库的操作指引，请参见 通过 OBClient 连接 OceanBase 租户（MySQL 模式） 和 通过 OBClient 连接 OceanBase 租户（Oracle 模式）。

2. 终止指定 Client Session。

   - 终止指定的 Client Session

     ```
     KILL PROXYSESSION {cs_id | connection_id};
     ```

     参数说明：

     - `id` 既可以是 `cs_id`，也可以是 `connection_id`，显示结果相同。

       `cs_id` 为 ODP 内部标记的每个 Client 的 `id` 号，`connection_id` 为整个 OceanBase 数据库标记的每个 Client 的 `id` 号。`connection_id` 通过 `SELECT CONNECTION_ID();` 语句获取。

     - 与 `KILL connection_id` 的作用一致。有关 `KILL` 语句的详细介绍，请参见 KILL（MySQL 模式） 或 KILL（Oracle 模式）。

     示例如下：

     a. 查看 Client Session，获取 `cs_id` 或 `connection_id`。

        执行以下语句获取 `cs_id`。

        ```
        SHOW PROXYSESSION;
        ```

查询结果如下，返回结果中的 `Id` 即为 `cs_id` 。

```
+--------------------+-------+---------+---------+------
+--------------------+------+------------+------------------
+------------------+-------+-------+-----------+
| proxy_sessid | Id | Cluster | Tenant | User | Host | db | trans_count |
svr_session_count | state | tid | pid | using_ssl |
+--------------------+-------+---------+---------+------
+--------------------+------+------------+------------------
+------------------+-------+-------+-----------+
| 12402504630519660553 | 56932 | test420 | sys | root | 100.xx.xx.xx:63882 |
NULL | 0 | 1 | MCS_ACTIVE_READER | 76286 | 76286 | 0 |
+--------------------+-------+---------+---------+------
+--------------------+------+------------+------------------
+------------------+-------+-------+-----------+
1 row in set
```

或者，执行以下语句获取 `connection_id` 。

```
SELECT CONNECTION_ID ();
```

返回如下所示结果。

```
+-----------------+
| CONNECTION_ID () |
+-----------------+
| 3221562906 |
+-----------------+
1 row in set
```

b. 终止 `cs_id` 为 `56392` 或 `connection_id` 为 `3221562906` 的 Client Session。

```
KILL PROXYSESSION 56392;
```

或者

```
KILL PROXYSESSION 3221562906;
```

执行成功后，提示如下所示报错信息。

```
ERROR 1317 (70100): Query execution was interrupted
```

c. 查询中断后，查看 Client Session，发现连接已中断。

```
SHOW PROXYSESSION;
```

提示以下信息：

```
ERROR 2013 (HY000): Lost connection to MySQL server during query
```

d. 再次查看 Client Session。

```
SHOW PROXYSESSION;
```

提示以下信息：

```
ERROR 2006 (HY000): OceanBase server has gone away
No connection. Trying to reconnect...
Connection id: 57195
Current database: *** NONE ***

+---------------------+-------+---------+--------+------+---------------------
+------+------------+-----------------+------------------+-------+-------
+-----------+
| proxy_sessid | Id | Cluster | Tenant | User | Host | db | trans_count |
svr_session_count | state | tid | pid | using_ssl |
+---------------------+-------+---------+--------+------+---------------------
+------+------------+-----------------+------------------+-------+-------
+-----------+
| 12402504630519660555 | 57195 | test420 | sys | root | 100.xx.xx.xx:24996 |
NULL | 0 | 1 | MCS_ACTIVE_READER | 76286 | 76286 | 0 |
```

```
+---------------------+-------+---------+--------+------+--------------------
+------+------------+-----------------+------------------+-------+-------
+-----------+
1 row in set
```

通过示例可知，通过指定 cs_id 或 connection_id 来 KILL 当前的 Session 时，命令执行成功后，当前的连接中断。使用 SHOW PROXYSESSION 查看时，OBClient 会重新建立 Session 连接，并执行 SQL 显示结果。

- 终止指定 Client Session 上的 Server Session

```
KILL PROXYSESSION {cs_id | connection_id} ss_id
```

参数说明：

- id 既可以是 cs_id，也可以是 connection_id，显示结果相同。

  cs_id 为 ODP 内部标记的每个 Client 的 id 号，connection_id 为整个 OceanBase 数据库标记的每个 Client 的 id 号。connection_id 通过 SELECT CONNECTION_ID(); 语句获取

- ss_id 表示 ODP 内部标记每个 Server 端会话（Server Session）的 id 号，可以从 SHOW PROXYSESSION ATTRIBUTE id 中获取。详细的获取操作请参见 展示 Session 详细状态。

示例如下：

a. 查看 Client Session，获取 cs_id 。

```
SHOW PROXYSESSION;
```

查询结果如下，返回结果中的 Id 即为 cs_id 。

```
+---------------------+-------+---------+--------+------+--------------------
+------+------------+-----------------+------------------+-------+-------
+-----------+
| proxy_sessid | Id | Cluster | Tenant | User | Host | db | trans_count |
svr_session_count | state | tid | pid | using_ssl |
```

```
+---------------------+-------+----------+--------+------+---------------------
+------+------------+-----------------+------------------+-------+-------
+-----------+
| 12402504630519660556 | 64940 | test420 | sys | root | 100.xx.xx.xx:63882 |
NULL | 0 | 1 | MCS_ACTIVE_READER | 76286 | 76286 | 0 |
+---------------------+-------+----------+--------+------+---------------------
+------+------------ +------------------+------------------+-------+-------
+-----------+
1 row in set
```

b. 根据上一步得到的 cs_id 获取 ss_id 。

```
SHOW PROXYSESSION ATTRIBUTE 64940;
```

查询结果如下：

```
+---------------------------------+---------------------+---------------+
| attribute_name | value | info |
+---------------------------------+---------------------+---------------+
| proxy_sessid | -6044239443189891060 | cs common |
| cs_id | 64940 | cs common |
| cluster | test3233 | cs common |
| tenant | sys | cs common |
| user | root | cs common |
| host_ip | 100.xx.xx.xx | cs common |
| host_port | 63882 | cs common |
| db | NULL | cs common |
| total_trans_cnt | 0 | cs common |
| svr_session_cnt | 1 | cs common |
| active | true | cs common |
| read_state | MCS_ACTIVE_READER | cs common |
| tid | 76286 | cs common |
```

```
| pid | 76286 | cs common |
| idc_name | | cs common |
| modified_time | 0 | cs stat |
| reported_time | 0 | cs stat |
| hot_sys_var_version | 0 | cs var version |
| sys_var_version | 2 | cs var version |
| user_var_version | 0 | cs var version |
| last_insert_id_version | 0 | cs var version |
| db_name_version | 0 | cs var version |
| server_ip | xx.xx.xx.xx | last used ss |
| server_port | 2881 | last used ss |
| server_sessid | 3221579563 | last used ss |
| ss_id | 16 | last used ss |
| state | MSS_KA_CLIENT_SLAVE | last used ss |
| transact_count | 2 | last used ss |
| server_trans_stat | 0 | last used ss |
| hot_sys_var_version | 0 | last used ss |
| sys_var_version | 2 | last used ss |
| user_var_version | 0 | last used ss |
| last_insert_id_version | 0 | last used ss |
| db_name_version | 0 | last used ss |
| is_checksum_supported | 1 | last used ss |
| is_safe_read_weak_supported | 0 | last used ss |
| is_checksum_switch_supported | 1 | last used ss |
| checksum_switch | 1 | last used ss |
| enable_extra_ok_packet_for_stats | 1 | last used ss |
+---------------------------------+---------------------+----------------+
39 rows in set
```

c. 终止 Client Session 上的 Server Session。

```
KILL PROXYSESSION 64940 16;
```

# 25 ODP 表路由

本章节介绍 ODP 的表路由机制。

在 OceanBase 数据库中，Partition 是数据存储的基本单元。当我们创建表时，就会存在表和 Partition 的映射。如果是非分区表，一张表仅对应一个 Partition，如果是分区表，一张表可能会对应多个 Partition。目前 OceanBase 数据库未实现 Partition 的合并和分裂。

在 OceanBase 数据库中，有 Local 计划、Remote 计划和 Distributed 计划三种表路由。Local 计划、Remote 计划均为单分区的路由。ODP 的作用就是尽量消除 Remote 计划，将路由尽可能的变为 Local 计划。如果表路由类型为 Remote 计划的 SQL 过多，说明该 ODP 的路由可能存在问题（可通过查看 `oceanbase.GV$OB_SQL_AUDIT` 视图中 `plan_type` 字段来确认）。

## 25.1 非分区表路由

在 ODP 中保存了 Partition 和 OBServer_addr 的映射缓存，Partition 通过 SQL Parser 获取表名，非分区表通过表名去 Location Cache 中获取分区信息；如果是访问分区表且语句中提供了分区键值，则还会获取其中的分区键值，然后通过表名从 Location Cache 获取分区元数据，之后再通过分区键值以及分区元数据计算出分区 ID，从而获取 Partition 所在节点的 IP。

存在如下三种情况：

- Location Cache 中不存在，访问 OBServer 节点的 `__all_virtual_proxy_schema` 表拉取。
- Location Cache 中存在并可用，直接使用。
- Location Cache 中存在但不可用，从 Location Cache 中去除并重新拉取。

更多 Location Cache 相关的说明，请参见 租户内路由。

## 25.2 分区表路由

和非分区表一样，在 ODP 中保存了 Partition 和 OBServer_addr 的映射缓存，Partition 通过 SQL Parser 功能解析获取表名。相比非分区表，分区表额外增加了一个分区 ID 信息，根据表名和分区 ID 去 Location Cache 中获取 Partition 所在 OBServer 的 IP。

分区 ID 通过 SQL 语句获取，来源往往是 where 子句的表达式、insert 语句 values 值。需要保证能获取到 `a = xxx`，其中 `a` 为分区键，`xxx` 为常量。示例如下：

```
obclient> CREATE TABLE t1(c1 int, c2 int) PARTITION BY hash(c1) partitions 5;
obclient> UPDATE t1 SET c2 = 3 WHERE c1 = 5;
```

这里提取到 `c1 = 5`，通过这个信息可以计算到影响到的分区 ID，将 SQL 转发给对应分区所在 OBServer 节点即可。

ODP 如果计算不出分区 ID，那么就无法拉取准确的路由。可以在 obproxy.log 日志文件中，搜索 "calculate partition id" 查看是否计算成功。示例如下：

```
[root@ocp-deploy-obnorpm2-bfc89744b-qsfkz log]# grep "calculate partition id" obproxy.log
[2021-10-28 17:18:18.552825] DEBUG [PROXY] ob_mysql_route.cpp:577 [73979][Y0-7FB3C9D07A80] [lt=10] [dc=0] succ to calculate partition id(part_id_=1)
[2021-10-28 17:18:31.045191] INFO [PROXY] ob_mysql_route.cpp:575 [73979][Y0-7FB3C9D07A80] [lt=17] [dc=0] fail to calculate partition id, just use tenant server (tmp_ret=-4002)
```

有关分区表的相关介绍，请参见 管理分区。

# 25.3 ODP 依赖的 OceanBase 数据库内部表

路由的准确性，除了与 ODP 本身有关，还依赖于从 OceanBase 数据库内部表中获取到正确信息。如下为 ODP 所依赖的 OceanBase 数据库内部表：

- `__all_virtual_proxy_schema`：用于保存分区与其所在机器的对应关系信息，也有 Partition 的部分相关信息。
- `__all_virtual_proxy_partition_info`：保存分区表相关分区信息。
- `__all_virtual_proxy_partition`：保存一级分区信息。
- `__all_virtual_proxy_sub_partition`：保存二级分区信息。

# 26 LDC 路由

逻辑数据中心（Logical Data Center，LDC）路由可用于解决分布式关系型数据库多地多中心部署时产生的异地路由延迟问题。

OceanBase 数据库作为典型的高可用分布式关系型数据库，使用 Paxos 协议进行日志同步，天然支持多地多中心的部署方式以提供高可靠的容灾保证。但当真正多地多中心部署时，任何数据库都会面临异地路由延迟问题。

逻辑数据中心（Logical Data Center，LDC）路由正是为了解决这一问题而设计的，通过给 OceanBase 集群的每个 Zone 设置 Region 属性和 IDC 属性，并给 ODP 指定 IDC 名称配置项后，当数据请求发到 ODP 时，ODP 将按如下优先级顺序进行路由转发：

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。

LDC 路由的设置需要进行以下 3 个步骤：

1. OceanBase 集群的 LDC 配置。
2. ODP 的 LDC 配置。
3. 应用配置弱一致性读。

## 26.0.0.1 说明

如果没有设置 LDC，那么当 OceanBase 数据库多地多中心部署时，弱一致性读会均匀访问数据的所有主从副本，当访问非本地域机房上的副本时，系统响应时间将大大增加。

# 26.1 OceanBase 集群的 LDC 配置

1. 在 OceanBase 数据库支持的客户端工具中使用 root 用户登录集群的 sys 租户。
2. 登录后，执行以下示例语句配置 LDC。

```
obclient> ALTER SYSTEM MODIFY zone "z1" SET region = "SHANGHAI";


obclient> ALTER SYSTEM MODIFY zone "z1" SET idc = "zue";
```

其中：

- 参数 `region` 表示 Zone 所在的地域信息，通常设置为城市名（大小写敏感）。
- 参数 `idc` 代表该 Zone 所处的机房信息，通常设置为机房名（小写）。
- 参数 `zone` 为当前设置的 Zone 的名称。
- 一个 OceanBase 集群中有若干个 Region，一个 Region 有若干个 Zone，一个 Zone 对应一个 IDC 属性。

3. 检查 OBServer 节点中 LDC 设置内容是否生效。

   查询语句如下所示，查看返回结果中对应 Zone 的 Region 和 IDC 字段的值是否符合您的设置值。

```
obclient> SELECT * FROM oceanbase.DBA_OB_ZONES;
```

# 26.2 ODP 的 LDC 配置

ODP 的 LDC 配置有两种方式，其中 zue 为示例机房：

- ODP 进程启动时通过参数指定进行配置（建议使用的方式），示例语句如下所示。

```
cd /opt/taobao/install/obproxy-1.5.5
./bin/obproxy -o proxy_idc_name=zue
```

- 在 OceanBase 数据库支持的客户端工具中运行 `ALTER` 语句修改 ODP 配置项，示例语句如下所示。

```
obclient> ALTER PROXYCONFIG SET proxy_idc_name= 'zue';
```

以上述示例中的 zue 机房为例，为 ODP 配置 LDC 后，如果应用部署在上海 zue 机房，当应用逻辑发起弱一致性读时，请求都是优先发往上海机房。

为 ODP 配置 LDC 后可通过下述语句检查配置是否生效。

```
obclient> SHOW PROXYINFO IDC;
```

返回结果如下所示：

```
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)
```

# 26.3 应用配置弱一致性读

LDC 路由仅针对弱一致性读的场景才会生效。应用配置弱一致性读有以下 3 种方法：

- 在 SQL 中携带 HINT 来指定，示例语句如下所示。

```
obclient> SELECT /*+read_consistency(weak)*/ * FROM t1;
```

- 设置 Session 级别的系统变量来指定（仅对当前 Session 生效），示例语句如下所示。

```
obclient> SET @@ob_read_consistency='weak';
```

- 设置全局级别的系统变量来指定（对该租户的所有连接都会生效），示例语句如下所示。

```
obclient> SET @@global.ob_read_consistency='weak';
```

# 27 读写分离

读写分离就是在主服务器上修改，数据会同步到备服务器，备服务器只能提供读取数据，不能写入，实现备份的同时也实现了数据库性能的优化，以及提升了服务器安全。对于单机数据库如 MySQL，读写分离指的是写请求发送给主服务器，读请求发送给备服务器。 对于 OceanBase 数据库，我们将读写分离从服务器级别做到了表的 parition 级别，这是 OceanBase 分布式数据库的特有优势。

使用读写分离功能，可以将读请求发送到备副本，降低主副本压力。

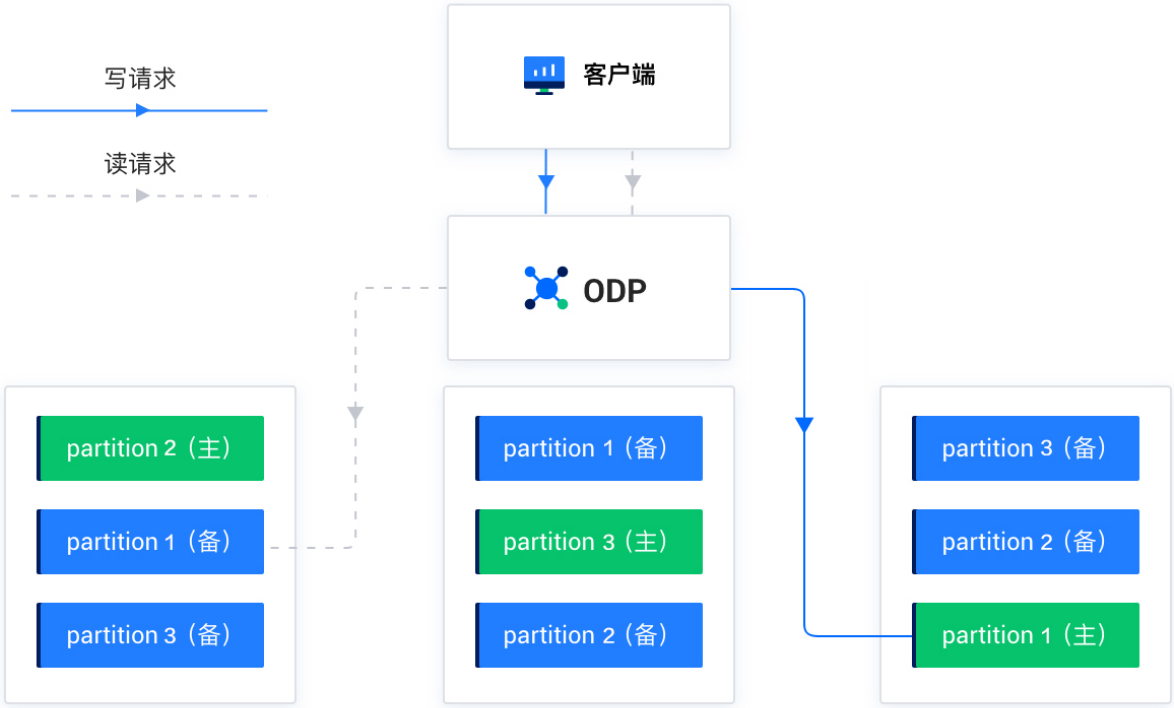
## 27.0.0.1 注意

使用读写分离后，读请求不保证能够读到最新的数据,时差一般在几百毫秒。

## 27.1 配置读写分离

要使用读写分离，需要两部分设置：

1. 设置 SQL 语句为弱读：所谓弱读是指读请求不要求读到最新的数据。对于正常的 SQL 为强读，使用弱读需要一定的设置。
2. 修改路由策略：以下图 partition1 为例，三个副本都可以提供弱读服务（主副本也可以提供读服务），通过设置路由策略，可以优先选择备副本。




### 27.1.1 弱读设置

# 27.1.1.1 通过 Hint 设置

OceabBase 的 SQL 提供了弱读 hint `/*+READ_CONSISTENCY(WEAK)*/`，在 SQL 中添加该 `hint` 后，就可以开启弱读功能。

```
select /*+READ_CONSISTENCY(WEAK)*/ * from t1;
```

# 27.1.1.2 通过配置项设置

通过 Hint 的方式需要修改 SQL，有时修改 SQL 会比较麻烦，可以通过修改配置项 `obproxy_read_consistency` 的值设置。

1. 使用 OBProxy 管理员账号登录数据库
2. 修改配置项

```
alter proxyconfig set obproxy_read_consistency = 1;
```

# 27.1.1.3 说明

该配置项取值为 0 和 1，默认为 0，表示强读（需要读到最新的数据）。1 表示弱读。

# 27.1.2 修改路由策略

ODP 通过配置项 `proxy_route_policy` 修改路由策略，通过设置为 `follower_first` 和 `follower_only` 可以让弱读请求优先发给备副本，这两者含义如下：

- follower_first：优先选择备副本，如果备副本都不可用，选择主副本
- follower_only：选择备副本，如果备副本都不可用，断开和客户端的连接 上述两个策略需要根据业务场景选择

# 28 备优先读

ODP 支持备优先读路由策略，通过用户变量 `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。

在 OceanBase 数据库支持的客户端工具中登录集群的业务租户后，运行下述语句对会话变量 `proxy_route_policy` 进行设置：

```
obclient> SET @proxy_route_policy='[policy]';
```

参数 `[policy]` 的取值有三种：

1. 当取值为 `follower_first` 时，路由逻辑是优先发备（即使集群在合并状态），优先级如下所示：

   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

2. 当取值为 `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）。优先级如下所示：

   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

3. 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。优先级如下所示：

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

# 29 黑名单机制

ODP 使用黑名单机制自适应处理 OBServer 节点的错峰合并、升级、leader 切换、宕机、启动和停止等过程中的 OBServer 节点访问控制。

## 29.1 背景信息

ODP 有状态黑名单、探测黑名单和活不可用黑名单三种类型，可通过设置参数对黑名单进行管理。

通过黑名单可主要实现如下场景的访问控制：

- 错峰合并时正在合并的 Zone 没有流量。
- 集群升级时不能访问正在升级的 OBServer 节点或 Zone。
- OBServer 节点宕机后，不再访问该 OBServer 节点。
- Partition 迁移后，不再访问迁移前的 OBServer 节点。
- 访问的 OBServer 节点不存在租户资源时重试。
- OBServer 节点活着不可用（内存超限、超时、OBServer 节点初始化和退出）时重试。

## 29.2 状态黑名单

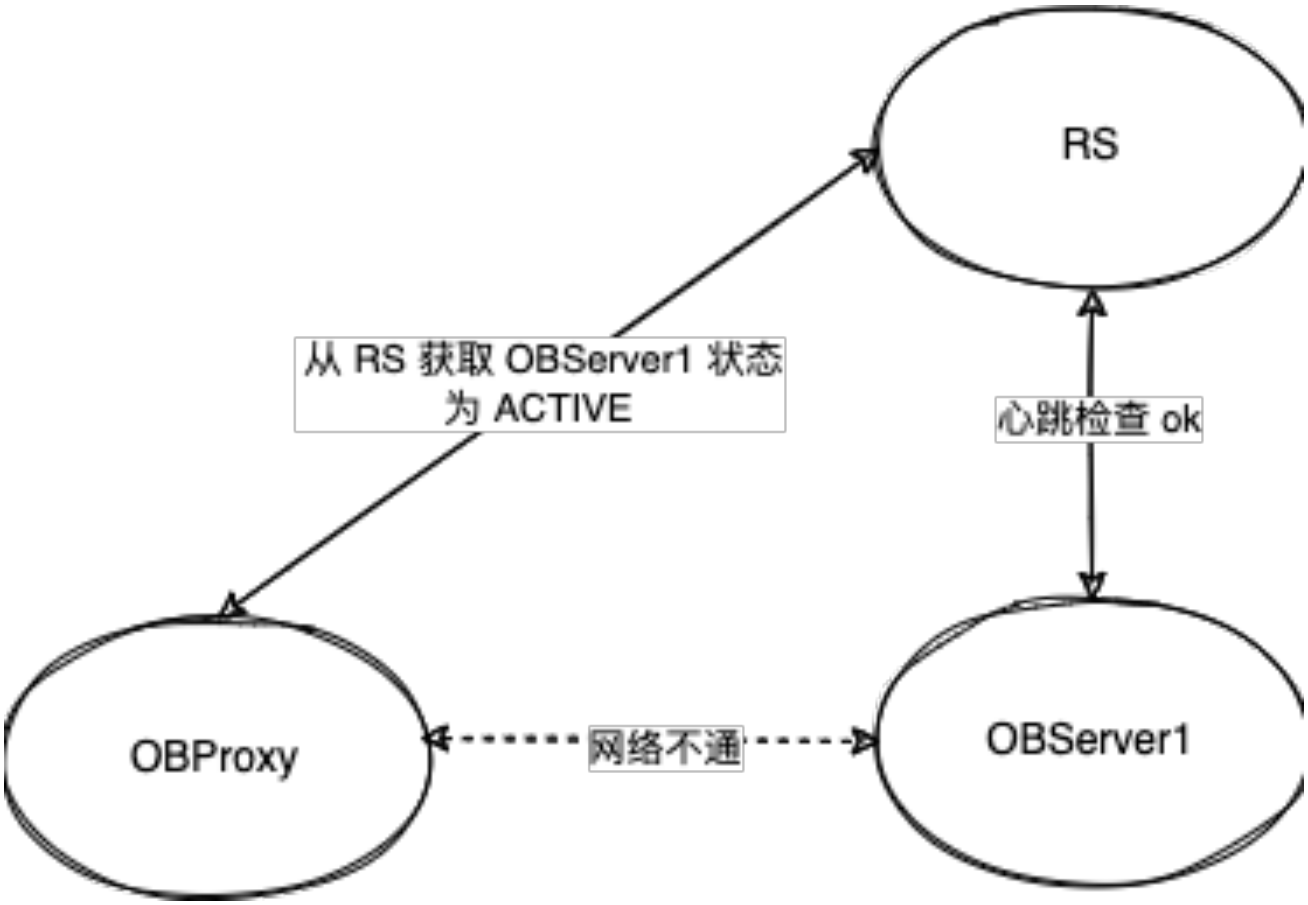
状态黑名单依赖于 OBServer 节点的状态变更。因为历史原因，和状态黑名单相关的状态不包含 `DETECT_ALIVE` 和 `DETECT_DEAD` 两种状态，剩下的状态变更都可以通过访问 OBServer 节点的视图获得。

当通过周期任务获得 OBServer 节点最新状态后，ODP 会进行根据 OBServer 节点状态的不同分别进行如下操作。

- `ACTIVE`：将 OBServer 节点从状态黑名单中移除，洗白 OBServer 节点。
- `INACTIVE` / `REPLAY`：将 OBServer 节点加入状态黑名单。
- `DELETED` / `DELETING` 状态：更新内存中 OBServer 节点的机器列表，不会向该节点转发 SQL。
- `UPGRADE` 状态：不加入状态黑名单，但也不会向该节点转发 SQL，效果和黑名单类似。

## 29.3 探测黑名单

状态黑名单是从 OceanBase 的总控服务节点（Root Service）处获得信息，这种信息有时不能反映 ODP 和 OBServer 节点之间的情况。如下图所示，ODP 从 RS 处获得 OBServer1 节点状态为 ACTIVE，但 ODP 和 OBServer1 节点之间网络不通。




因此，ODP 在现有的状态黑名单基础上又实现了探测黑名单，通过给 OBServer 节点发送探测 SQL 来确定 OBServer 节点状态。ODP 会给 OceanBase 数据库的 sys 租户发送探测 SQL `select 'detect server alive' from dual`，并设置 5s 的超时时间，当超时时间内无结果返回时，会增加一次探测失败次数，当连续失败次数超过 3 次后认为探测失败，设置 OBServer 节点状态为 `DETECT_DEAD`。如果有结果返回，会将探测失败次数清零，设置 OBServer 节点状态为 `DETECT_ALIVE`。发生探测状态变更后，触发 ODP 行为如下。

- `DETECT_ALIVE` 状态：将 OBServer 节点从探测黑名单中洗白。
- `DETECT_DEAD` 状态：将 OBServer 节点加入到探测黑名单，并将所有和该 OBServer 节点的连接关闭。

# 29.3.0.1 说明

如果将 OBServer 节点加入到探测黑名单后，不断开和该 OBServer 节点的连接，那么连接会被一直占用，无法发送新的 SQL，在性能数据上会看到这段时间的这台 OBServer 节点的 TPS 为 0。断开后，对于后续请求，ODP 会根据黑名单进行路由，不会转发给这台 OBServer 节点，TPS 就可以恢复。

# 29.4 活不可用黑名单

活不可用黑名单机制的核心是根据业务 SQL 的执行结果去定义 OBServer 节点的状态，触发拉黑和洗白操作。相较于状态黑名单和探测黑名单，活不可用黑名单的拉黑和洗白谨慎了很多，主要是为了防止一次 SQL 执行结果产生误判。具体操作如下。

- 失败：根据时间点记录下一次失败事件。
  - ODP 向 OBServer 节点发送 SQL 后，超过 ob_query_timeout 时间后仍然无响应。
  - OBServer 节点返回错误码 OB_SERVER_IS_INIT、OB_SERVER_IS_STOPPING、OB_PACKET_CHECKSUM_ERROR、OB_ALLOCATE_MEMORY_FAILED。
  - ODP 和 OBServer 节点建连失败/报文解析失败，数据传输失败。
- 拉黑：默认情况下，OBServer 节点在 120s 内有五次活不可用的失败记录则拉黑这台 OBServer 节点，由配置项 congestion_fail_window（错误统计周期）和 congestion_failure_threshold（出错次数）控制。
- 尝试：默认情况下，加入活不可用黑名单超过 20s 后，尝试再次向该 OBServer 节点发送 SQL。重试时间间隔由 `congestion_retry_interval` 配置项控制。
- 洗白：如果尝试的 SQL 执行成功则进行洗白。默认 OBServer 节点在 20s 内不允许被洗白，该值由配置项 `min_keep_congestion_interval` 控制。

上述三种黑名单之间并没有优先级，只要 OBServer 节点在任何一个黑名单中，这台机器就无法接收请求。换而言之，只有当 OBServer 节点不在任何黑名单中才会接收到请求。

所有可选择的 OBServer 节点都处于黑名单中时，会强制重试黑名单中的 OBServer 节点。

# 29.5 黑名单的配置参数

1. 在客户端中使用 root 用户，通过 ODP 代理登录集群的 `sys` 租户。

   ```
   obclient> obclient -h10.10.10.1 -uroot@sys#obdemo -P2883 -p -c
   ```

   此处是以 ODP 机器 IP 为 `10.10.10.1`，集群名称为 `obdemo` 为例，通过 OBClient 连接集群。连接 OceanBase 租户的详细操作可参见 连接方式概述（MySQL 模式） 和 连接方式概述（Oracle 模式）。

2. 执行以下命令查看 ODP 的配置参数。

   ```
   obclient> SHOW PROXYCONFIG;
   ```

3. 运行 `ALTER proxyconfig SET key=value` 命令更新 ODP 的指定配置参数，配置项更新后会立即生效。

与黑名单相关的配置参数描述如下表所示。

| 配置项 | 描述 |
| --- | --- |
| enable_congestion | 是否启用黑名单机制。默认开启，在黑名单机制运行出问题时，可以暂时关闭这个选项让 ODP 能够正常工作。 |
| congestion_failure_threshold | 控制是否将 OBServer 节点列入黑名单，与 congestion_fail_window 搭配使用，在 congestion_fail_window 时间内，OBServer 节点出错次数超过该值时，ODP 会将该 OBServer 节点加入黑名单。该参数值小于 0 时，则所有在黑名单中的 Server 都会被放出黑名单。 |
| congestion_fail_window | 设置错误统计周期的时间（秒），默认值 120 秒。如果在错误统计周期中错误次数超过 congestion_failure_threshold 的值，则将该 Server 节点加入黑名单。该参数值也可以动态调整，一旦修改了该参数，当前错误统计计数会被清零，将重新按新配置的值开始统计错误次数。 |
| congestion_retry_interval | 设置被加入活不可用黑名单的 OBServer 节点重试周期，默认值 20 秒。 |
| min_keep_congestion_interval | 设置被加入黑名单的 OBServer 节点，多长时间后才允许被放出黑名单。 |
| min_congested_connect_timeout | 设置客户端连接超时的值，当连接超时时则将 OBServer 节点加入活不可用黑名单，默认值为 100ms。 |

# 29.6 查看黑名单

1. 在客户端中使用 root 用户，通过 ODP 代理登录集群的 sys 租户。

   ```
   obclient> obclient -h10.10.10.1 -uroot@sys#obdemo -P2883 -p -c
   ```

   此处是以 ODP 机器 IP 为 10.10.10.1，集群名称为 obdemo 为例，通过 OBClient 连接集群。连接 OceanBase 租户的详细操作可参见 连接方式概述（MySQL 模式） 和 连接方式概述（Oracle 模式）。

2. 执行以下命令，查看黑名单信息。

   语句如下：

   ```
   SHOW PROXYCONGESTION [all] [clustername];
   ```

相关参数说明如下：

- 不指定可选项时，可查询所有集群的黑名单信息。
- 仅指定 `all` 时，可显示所有集群的 OBServer 节点信息（包括黑名单中和非黑名单中的 OBServer 节点信息）。
- 仅指定 `clustername` 时，可显示指定集群的黑名单信息。
- 同时指定 `all` 和 `clustername` 时，可显示指定集群的 OBServer 节点信息。

# 29.6.1 示例

此处以查看单节点集群 `obcluster` 黑名单信息为例。

```
obclient> show proxycongestion 'obcluster'\G
```

输出如下：

```
*************************** 1. row ***************************
cluster_name: obcluster
zone_name: zone1
region_name: sys_region
zone_state: ACTIVE
server_ip: xxx.xxx.xxx.xxx:2881
cr_version: 5
server_state: ACTIVE
alive_congested: 0
last_alive_congested: 0
dead_congested: 0
last_dead_congested: 0
stat_alive_failures: 0
stat_conn_failures: 0
conn_last_fail_time: 0
conn_failure_events: 0
alive_last_fail_time: 0
```

```
alive_failure_events: 0
ref_count: 2
detect_congested: 0
last_detect_congested: 0
```

# 29.6.1.1 说明

- 当 `dead_congested` 和 `alive_congested` 中的任意一个状态为 `1` 时，表示该 OBServer 节点已加入黑名单。
- ODP V1.1.2 版本之前，`clustername` 不需要单引号或者双引号引起来，V1.1.2 版本及之后，必须要用单引号或者双引号引起来。

# 30 事务路由

## 30.1 ODP 事务路由

OBProxy 针对事务有两种路由方式，一种是将同一个事务的语句统一路由到一个数据节点上执行。另外一种则是将事务中的语句拆分，路由到不同的数据节点执行，通过 OBProxy 同步不同数据节点的事务的状态，完成事务的拆分执行。本文主要介绍 OBProxy 对事务拆分执行的路由方式。 事务路由依赖 2.0 协议进行事务状态的同步。

## 30.2 事务路由节点选择

事务路由的执行节点分为协调者和参与者两种节点，协调者节点即事务的开启节点负责执行会影响事务状态的非 DML 语句，比如 `BEGIN`、`START TRANSACTION`、`COMMIT`、`ROLLBACK` 等，参与者节点负责执行事务中不会影响事务状态的 DML 语句。 OBProxy 包含的简单的 SQL Parser，能够解析出 SQL 语句是否 DML 语句，OBProxy 解析出一条语句属于 DML 语句之后，会通过表路由或者 LDC 路由，将语句路由到合适的节点，被路由到的节点即成为事务的参与者节点。而事务的协调者节点即事务中第一条语句的执行节点。

## 30.3 事务路由配置

默认情况下事务路由的功能是开启的，可以同过以下语句查询、运维事务路由的功能。

1. 使用 root 用户登录集群的 sys 租户。
2. 检查修改 2.0 协议的配置，确保 2.0 协议启用。

```
obclient> SHOW PROXYCONFIG LIKE enable_ob_protocol_v2;
obclient> ALTER PROXYCONFIG SET enable_ob_protocol_v2=True;
```

3. 运行以下语句对 OBProxy 的事务路由进行配置。

```
obclient> SHOW PROXYCONFIG LIKE enable_transaction_internal_routing;
obclient> ALTER PROXYCONFIG SET enable_transaction_internal_routing=True;
```

# 31 查看租户会话

您可以通过视图或 SQL 语句查看租户会话。

## 31.1 注意事项

本文档以 OceanBase 数据库版本为 V4.3.0、ODP 版本为 V4.2.3 版本，同时 ODP 的配置项 `client_session_id_version =2` 为例提供操作指导，不同版本和设置其返回结果不完全一样。有关 ODP 配置项值的查看及设置方法，请参见 配置项说明。

## 31.2 通过视图查看租户会话

1. 用户登录集群的 MySQL 租户或 Oracle 租户。
2. 执行以下语句，查看租户会话信息。

   租户管理员可以查看当前租户内的所有会话信息，普通用户只能查看当前自己的会话信息。如果您拥有 `PROCESS` 权限，则可以查看所有会话。

   - MySQL 模式

     ```
     obclient [oceanbase]> SELECT * FROM oceanbase.GV$OB_PROCESSLIST\G
     ```

     查询结果如下：

     ```
     *************************** 1. row ***************************
     SVR_IP: 172.xx.xxx.xxx
     SVR_PORT: 2882
     SQL_PORT: 2881
     ID: 3221487622
     USER: root
     HOST: 100.xx.xxx.xxx:49451
     DB: test
     TENANT: mysql001
     COMMAND: Sleep
     TIME: 631.462197
     TOTAL_TIME: 631.462197
     STATE: SLEEP
     ```

```
INFO: NULL
PROXY_SESSID: NULL
MASTER_SESSID: NULL
USER_CLIENT_IP: 100.xx.xxx.xxx
USER_HOST: %
RETRY_CNT: 0
RETRY_INFO: 0
SQL_ID:
TRANS_ID: 3413
THREAD_ID: 0
SSL_CIPHER: NULL
TRACE_ID: NULL
TRANS_STATE: ACTIVE
ACTION:
MODULE:
CLIENT_INFO:
LEVEL: 1
SAMPLE_PERCENTAGE: 10
RECORD_POLICY: SAMPLE_AND_SLOW_QUERY
LB_VID: NULL
LB_VIP: NULL
LB_VPORT: NULL
IN_BYTES: 448
OUT_BYTES: 0
USER_CLIENT_PORT: 49451
TOTAL_CPU_TIME: 0
PROXY_USER:
SERVICE_NAME: NULL
TOTAL_CPU_TIME: 0
```

```
TOP_INFO: NULL
MEMORY_USAGE: NULL
*************************** 2. row ***************************
SVR_IP: 172.xx.xxx.xxx
SVR_PORT: 2882
SQL_PORT: 2881
ID: 3221487623
USER: root
HOST: 100.xx.xxx.xxx:14422
DB: NULL
TENANT: mysql001
COMMAND: Query
TIME: 0.005717
TOTAL_TIME: 0.005774
STATE: ACTIVE
INFO: SELECT * FROM oceanbase.GV$OB_PROCESSLIST
PROXY_SESSID: NULL
MASTER_SESSID: NULL
USER_CLIENT_IP: 100.xx.xxx.xxx
USER_HOST: %
RETRY_CNT: 0
RETRY_INFO: 0
SQL_ID: 2E175335D36600B7A8EC72C5094888DD
TRANS_ID: 0
THREAD_ID: 33799
SSL_CIPHER: NULL
TRACE_ID: YB42A**********
TRANS_STATE:
ACTION:
```

```
MODULE:
CLIENT_INFO:
LEVEL: 1
SAMPLE_PERCENTAGE: 10
RECORD_POLICY: SAMPLE_AND_SLOW_QUERY
LB_VID: NULL
LB_VIP: NULL
LB_VPORT: NULL
IN_BYTES: 449
OUT_BYTES: 0
USER_CLIENT_PORT: 14422
TOTAL_CPU_TIME: 0
PROXY_USER:
SERVICE_NAME: NULL
TOTAL_CPU_TIME: 0
TOP_INFO: NULL
MEMORY_USAGE: NULL
2 rows in set
```

- Oracle 模式

```
obclient [SYS]> SELECT * FROM SYS.GV$OB_PROCESSLIST\G
```

查询结果如下：

```
*************************** 1. row ***************************
SVR_IP: 172.xx.xxx.xxx
SVR_PORT: 2882
SQL_PORT: 2881
ID: 3221487626
USER: SYS
HOST: 100.xx.xxx.xxx:25945
```

```
DB: SYS
TENANT: oracle001
COMMAND: Sleep
TIME: 14
TOTAL_TIME: 14
STATE: SLEEP
INFO: NULL
PROXY_SESSID: NULL
MASTER_SESSID: NULL
USER_CLIENT_IP: 100.xx.xxx.xxx
USER_HOST: %
RETRY_CNT: 0
RETRY_INFO: 0
SQL_ID: NULL
TRANS_ID: 6343
THREAD_ID: 0
SSL_CIPHER: NULL
TRACE_ID: NULL
TRANS_STATE: ACTIVE
ACTION: NULL
MODULE: NULL
CLIENT_INFO: NULL
LEVEL: 1
SAMPLE_PERCENTAGE: 10
RECORD_POLICY: SAMPLE_AND_SLOW_QUERY
LB_VID: NULL
LB_VIP: NULL
LB_VPORT: NULL
IN_BYTES: 447
```

```
OUT_BYTES: 0
USER_CLIENT_PORT: 25945
TOTAL_CPU_TIME: 0
PROXY_USER: NULL
SERVICE_NAME: NULL
TOTAL_CPU_TIME: 0
TOP_INFO: NULL
MEMORY_USAGE: NULL
*************************** 2. row ***************************
SVR_IP: 172.xx.xxx.xxx
SVR_PORT: 2882
SQL_PORT: 2881
ID: 3221487625
USER: SYS
HOST: 100.xx.xxx.xxx:24782
DB: SYS
TENANT: oracle001
COMMAND: Query
TIME: 0
TOTAL_TIME: 0
STATE: ACTIVE
INFO: SELECT * FROM SYS.GV$OB_PROCESSLIST
PROXY_SESSID: NULL
MASTER_SESSID: NULL
USER_CLIENT_IP: 100.xx.xxx.xxx
USER_HOST: %
RETRY_CNT: 0
RETRY_INFO: 0
SQL_ID: 0A6CF0E2AB2C1A1917AB1FFDF2BE9CFF
```

```
TRANS_ID: 0
THREAD_ID: 34190
SSL_CIPHER: NULL
TRACE_ID: YB42A***************
TRANS_STATE: NULL
ACTION: NULL
MODULE: NULL
CLIENT_INFO: NULL
LEVEL: 1
SAMPLE_PERCENTAGE: 10
RECORD_POLICY: SAMPLE_AND_SLOW_QUERY
LB_VID: NULL
LB_VIP: NULL
LB_VPORT: NULL
IN_BYTES: 449
OUT_BYTES: 0
USER_CLIENT_PORT: 24782
TOTAL_CPU_TIME: 0
PROXY_USER: NULL
SERVICE_NAME: NULL
TOTAL_CPU_TIME: 0
TOP_INFO: NULL
MEMORY_USAGE: NULL
2 rows in set
```

查询结果中的相关字段说明如下：

- `SVR_IP`：表示该会话所属的服务器的 IP 地址，即 OBServer 节点的 IP 地址。
- `SVR_PORT`：表示该会话所属的服务器的 RPC 端口号，即 OBServer 节点的 RPC 端口号。

- SQL_PORT：表示该会话所属的服务器的 SQL 端口号，即 OBServer 节点的 SQL 端口号。
- ID：表示该会话 ID。
- USER：表示该会话所属的用户。
- HOST：表示发起该会话的客户端 IP 及端口号。

# 31.2.0.1 说明

本文档以 OceanBase 数据库版本为 V4.3.0、ODP 版本为 V4.2.3，同时 ODP 的配置项 client_session_id_version =2 为例。对于 OceanBase 数据库 V4.3.0 之前的版本或 ODP V4.2.3 之前版本，或者 ODP 为 V4.2.3 版本但配置项未设置为 client_session_id_version =2 的场景，通过直连方式连接数据库时，该字段表示发起该会话的客户端 IP 及端口号。如果是通过 ODP 连接数据库，该字段则表示 ODP 的主机 IP 及端口号。

- DB：表示该会话当前连接的数据库名。Oracle 模式显示为与用户名同名的 Schema 名。
- TENANT :表示该会话所访问的租户名称。
- COMMAND：表示该会话正在执行的命令类型。
- TIME：表示当前命令执行的时间，单位为秒。如果命令发生了重试，则系统会清零后重新计算。
- TOTAL_TIME：如果会话中有正在执行的命令，则表示从命令开始执行到当前时间的间隔；如果没有正在执行的命令，则表示该会话在当前状态的持续时间，单位为秒。
- STATE：表示该会话当前的状态。
- INFO：表示该会话正在执行的语句。
- PROXY_SESSID：如果是通过 ODP 连接数据库，则本列显示的是 ODP 节点上的 Session ID。否则显示为 NULL 。
- MASTER_SESSID：如果是通过 ODP 连接数据库，则本列显示的是 ODP 节点上的主 Session ID，用于串联同一个 SQL 的多个子 Session。否则显示为 NULL 。
- USER_CLIENT_IP：发起该会话的客户端 IP。

- USER_HOST：发起该会话的客户端主机名。
- RETRY_CNT：当前命令的重试次数。
- RETRY_INFO：当前命令的重试信息，一般为最后一次重试的错误码。
- SQL_ID：当前命令对应的 SQL ID 信息。
- TRANS_ID：事务 ID。
- THREAD_ID：线程 ID。如果是通过 ODP 连接数据库，则该线程 ID 也表示 ODP 节点上的 Session ID。
- SSL_CIPHER：加密密码名称。
- TRACE_ID：Trace ID。
- TRANS_STATE：事务状态。
  - IDLE：空闲状态，表示事务未开始。
  - ACTIVE：事务显式开启。通过 START TRANSACTION、BEGIN 等语句显式开启。
  - IMPLICIT_ACTIVE：事务隐式开启，autocommit = on 模式下，执行 DML 等修改数据的语句后事务被开启。
  - ROLLBACK_SAVEPOINT：事务正在回滚至保存点。
  - IN_TERMINATE：事务正在结束，包括内部原因导致的结束和用户发起的结束事务（例如，执行 COMMIT、ROLLBACK 语句）。
  - ABORTED：事务因发生异常而内部终止或回滚。
  - ROLLED_BACK：事务已经被回滚。
  - COMMIT_TIMEOUT：事务提交超时。
  - COMMIT_UNKNOWN：事务提交结果未知。
  - COMMITTED：事务已经成功提交。
- ACTION：通过调用 DBMS_APPLICATION_INFO.SET_ACTION Procedure 设置的当前执行操作的名称。

- MODULE：通过调用 DBMS_APPLICATION_INFO.SET_MODULE Procedure 设置的当前执行操作的名称。
- CLIENT_INFO：通过 DBMS_APPLICATION_INFO.SET_CLIENT_INFO 过程设置的信息。
- LEVEL：表示该会话的全链路追踪的监控级别。例如，1 表示 Level 为 1 的诊断信息。
- SAMPLE_PERCENTAGE：表示该会话的全链路追踪的采样频率。例如，10 表示以 10% 的概率进行诊断信息的采样。
- RECORD_POLICY：表示该会话的全链路追踪的记录策略，主要支持以下三种策略:
  - ALL：表示所有命中采样打开 Trace 的 Trace 打点信息均会打印到日志文件中，并且是在每个 Span 结束时, 即打印到日志文件中。
  - ONLY_SLOW_QUERY：表示所有命中采样打开 Trace 的 Trace 打点信息中，属于慢查询的请求 Trace 会打印到日志文件中，其余的均会丢弃。

    对于该打印策略，Trace 日志的打印时机是在请求结束被判定为慢查询后才打印。在 Proxy 中，对于根 Span 即事务级别的 Span，是在发现有慢查询时，就会打印根 Span。
  - SAMPLE_AND_SLOW_QUERY：表示所有命中采样打开 Trace 的 Trace 打点信息中，满足以下条件之一的 Trace 均会打印到日志文件:

    a. 当使用隐藏配置项 _print_sample_ppm 进行万分之一的概率采样，且只有当采样结果被选中时，才会将其打印到日志中

    b. 满足 ONLY_SLOW_QUERY 策略下的打印条件

    此外，该模式下, 日志打印的时机也不是在 Span 结束时，如果满足条件 1，则在客户端就被标记为该 trace 日志会被强制打印，并且该信息会传递到后面的链路组件；如果满足条件 2，则打印时机与 ONLY_SLOW_QUERY 策略相同。

    有关全链路追踪的详细介绍，请参见 全链路追踪。
- LB_VID：如果是公有云环境上通过负载均衡直接连接数据库，则本列显示的是负载均衡服务的 VPC ID。其他场景下始终显示为 NULL。
- LB_VIP：如果是公有云环境上通过负载均衡直接连接数据库，则本列显示的是客户端连接负载均衡服务的 IP。其他场景下始终显示为 NULL。

- LB_VPORT：如果是公有云环境上通过负载均衡直接连接数据库，则本列显示的是客户端连接负载均衡服务的端口。其他场景下始终显示为 NULL。
- IN_BYTES：该会话的流入流量。
- OUT_BYTES：该会话的流出流量。
- USER_CLIENT_PORT：发起该会话的客户端端口号。
- PROXY_USER：代理用户登录场景，显示代理用户的名称；非代理用户登录场景，显示为空。
- SERVICE_NAME：该 Session 由哪个服务名创建，值为 NULL 表示该 Session 不是由服务名创建。
- TOTAL_CPU_TIME：当前命令执行的 CPU 耗时，单位为秒。如果命令发生了重试，则系统不会清零。
- TOP_INFO：展示执行中 SQL 所在的顶层 PL 语句信息。
- MEMORY_USAGE：单条 SQL 语句当前占用的内存大小，单位为字节。

# 31.3 通过 SHOW PROCESSLIST 语句或 SHOW FULL PROCESSLIST 语句查看租户会话

您也可以通过 SHOW PROCESSLIST 语句或 SHOW FULL PROCESSLIST 语句查看租户会话。

查看租户会话时，租户管理员可以查看当前租户内的所有会话信息，普通用户只能查看当前自己的会话信息。如果您拥有 PROCESS 权限，则可以查看所有租户内的所有会话。查看用户权限的相关操作请参见 查看用户权限（Oracle 模式） 和 查看用户权限（MySQL 模式）。如果您没有所需的权限，请联系管理员为您添加，为用户添加权限的相关操作请参见 直接授予权限（Oracle 模式） 和 直接授予权限（MySQL 模式）。

1. 用户登录集群的 MySQL 租户或 Oracle 租户。
2. 执行以下命令，查看租户会话。
   - 使用 SHOW PROCESSLIST 语句查看租户会话

## 31.3.0.1 注意

本文档以 OceanBase 数据库版本为 V4.3.0、ODP 版本为 V4.2.3，同时 ODP 的配置项 client_session_id_version =2 为例。对于 OceanBase 数据库 V4.3.0 之前的版本或

ODP V4.2.3 之前版本，或者 ODP 为 V4.2.3 版本但配置项未设置为 client_session_id_version =2 的场景，当通过 ODP 连接数据库时，SHOW PROCESSLIST 语句显示的是对应的 ODP 节点上的会话信息；当通过直连方式连接数据库时，SHOW PROCESSLIST 语句显示的是租户对应的 OBServer 节点上的会话信息。

```
obclient [SYS]> SHOW PROCESSLIST;
```

查询结果如下：

```
+------------+------+---------------------+------+---------+------+--------
+------------------+
| ID | USER | HOST | DB | COMMAND | TIME | STATE | INFO |
+------------+------+---------------------+------+---------+------+--------
+------------------+
| 3221488068 | SYS | 100.xx.xxx.xxx:18985 | SYS | Sleep | 298 | SLEEP | NULL |
| 3221488078 | SYS | 100.xx.xxx.xxx:32489 | SYS | Query | 0 | ACTIVE | SHOW
PROCESSLIST |
+------------+------+---------------------+------+---------+------+--------
+------------------+
2 rows in set
```

相关字段说明如下：

- Id：表示该会话的 ID。
- User：表示该会话所属的用户。
- Host：表示发起该会话的客户端 IP 及端口号。
- db：表示该会话当前连接的数据库名。Oracle 模式显示为与用户名同名的 Schema 名。
- Command：表示该会话正在执行的命令类型。
- Time：表示当前命令执行的时间，单位为秒。如果命令发生了重试，则系统会清零后重新计算。
- State：表示该会话当前的状态。

- Info：表示该会话正在执行的语句。

- 使用 SHOW FULL PROCESSLIST 语句查看租户会话

```
obclient [SYS]> SHOW FULL PROCESSLIST;
```

查询结果如下：

```
+------------+------+-----------+---------------------+------+---------+------
+--------+----------------------+----------------+------+
| ID | USER | TENANT | HOST | DB | COMMAND | TIME | STATE | INFO | IP | PORT |
+------------+------+-----------+---------------------+------+---------+------
+--------+----------------------+----------------+------+
| 3221488068 | SYS | oracle001 | 100.xx.xxx.xxx:18985 | SYS | Sleep | 548 | SLEEP |
NULL | 172.xx.xxx.xxx | 2881 |
| 3221488078 | SYS | oracle001 | 100.xx.xxx.xxx:32489 | SYS | Query | 0 | ACTIVE |
SHOW FULL PROCESSLIST | 172.xx.xxx.xxx | 2881 |
+------------+------+-----------+---------------------+------+---------+------
+--------+----------------------+----------------+------+
2 rows in set
```

相关字段说明如下：

- ID：表示该会话的 ID。
- USER：表示该会话所属的用户。
- TENANT：表示该会话所访问的租户名称。
- HOST：表示发起该会话的客户端 IP 及端口号。如果您是通过 ODP 连接的数据库，则表示 ODP 的主机 IP 及端口号。
- DB：表示该会话当前连接的数据库名。Oracle 模式显示为与用户名同名的 Schema 名。
- COMMAND：表示该会话正在执行的命令类型。
- TIME：表示当前命令执行的时间，单位为秒。如果命令发生了重试，则系统会清零后重新计算。

- STATE：表示该会话当前的状态。
- INFO：表示该会话执行的语句。
- IP：表示该会话所属的服务器 IP 地址，即 OBServer 节点的 IP 地址。
- PORT：表示该会话所属的服务器的 SQL 端口号，即 OBServer 节点的 SQL 端口号。

# 31.4 相关文档

- 终止租户会话
- 设置租户最大连接数

# 32 终止租户会话

OceanBase 数据库支持通过 KILL 语句终止会话。

终止会话的 SQL 语句如下：

```
KILL [CONNECTION] 'session_id';
```

语句使用说明：

- KILL 语句每次只能终止一个会话。
- 如果您拥有 PROCESS 权限，则您可以查看所有会话。如果您拥有 SUPER 权限，您可以逐一终止所有会话和语句。否则，您只能查看和终止您自己的会话和语句。管理员可以逐一终止所有会话和语句。

## 32.0.0.1 说明

- Oracle 模式下，查看权限的相关操作请参见 查看用户权限。如果您没有所需的权限，请联系管理员为您添加，为用户添加权限的相关操作请参见 直接授予权限。
- MySQL 模式下，查看当前拥有权限的操作请参见 查看用户权限。如果您没有所需的权限，请联系管理员为您添加，为用户添加权限的相关操作请参见 直接授予权限。

终止指定的会话的示例如下：

```
KILL session_id;
```

或者

```
KILL CONNECTION session_id;
```

其中，session_id 即会话 ID，可通过 SHOW PROCESSLIST 语句或 SHOW FULL PROCESSLIST 语句获取。获取会话 ID 的详细操作及说明，请参见 查看租户会话。

更多 KILL 语句的使用和说明，请参见 KILL（MySQL 模式） 和 KILL（Oracle 模式）。

# 32.1 相关文档

- MySQL 模式下的权限分类
- Oracle 模式下的权限分类
- 查看租户会话
- 设置租户最大连接数

# 33 设置租户最大连接数

由于连接会占用一定的资源，大量的长连接会影响小规格租户的稳定性，OceanBase 数据库支持通过配置项或系统变量为租户设置最大连接数。

与租户最大连接数相关的配置项或变量如下：

- max_connections

  系统变量 max_connections 用于控制整个租户的最大连接数，当前仅 MySQL 模式支持该系统变量。其默认值为 2147483647，取值范围为 [1,2147483647]。该变量修改后，需要重启 OBServer 节点才能生效。

- max_user_connections

  系统变量 max_user_connections 用于控制租户内单个用户的最大并发连接数，当前仅 MySQL 模式支持该系统变量。其默认值为 0，表示不限制连接数；当值大于 0 时，以实际设置的值为准，取值范围为 [0,4294967295]。该变量修改后，需要重启 OBServer 节点才能生效。

- _resource_limit_max_session_num

  租户级隐藏配置项 _resource_limit_max_session_num 用于控制用户租户内普通用户的最大并发连接数，该配置项修改后立即生效，不需要重启 OBServer 节点。其默认值为 0，取值范围为 [0,1000000]。该配置项一般不需要修改。

  当隐藏配置项 _resource_limit_max_session_num 的值大于 0 时，以实际设置的值为准；当隐藏配置项 _resource_limit_max_session_num 的值为 0 时，系统内部会自动按照一定的规则计算出最大连接数。具体计算规则如下：

  用户租户并发连接数上限 = MAX（100, 租户内存 * 5%/（100 KB））

  其中，租户内存由创建租户时分配的资源池中资源单元的内存决定；100 KB 为根据经验计算出的单个会话可能占用的内存；100 为连接数下限。根据该算法，如果租户的内存为 2 GB，则该租户的最大连接数为 MAX（100, 2 GB * 5%/（100 KB））= 1000 。

## 33.1 注意事项

在 OceanBase 数据库中，名称以 "_" 开头的配置项称为隐藏配置项，仅供开发人员在故障排查或紧急运维时使用。

## 33.2 使用限制

- 仅 MySQL 租户可通过系统变量 max_connections 和 max_user_connections 调整租户或租户内用户的最大连接数。
- 为了防止因出现大量长连接而导致的运维人员无法登录的问题，配置项 _resource_limit_max_session_num 的值对 sys 租户或用户租户的管理员用户（MySQL 模式默认为 root 用户，Oracle 模式默认为 SYS 用户）不受限。

# 33.3 MySQL 租户查看本租户的最大连接数

1. 租户管理员登录到集群的 MySQL 租户。
2. 查看租户的最大连接数。
   - 查看整个租户的最大连接数

     ```
     obclient> SHOW VARIABLES LIKE 'max_connections';
     ```

     或者

     ```
     obclient> SELECT * FROM INFORMATION_SCHEMA.SESSION_VARIABLES WHERE
     VARIABLE_NAME = 'max_connections';
     ```

     查询结果如下：

     ```
     +-----------------+------------+
     | Variable_name | Value |
     +-----------------+------------+
     | max_connections | 2147483647 |
     +-----------------+------------+
     1 row in set
     ```

   - 查看租户内单个用户的最大并发连接数

     ```
     obclient> SHOW VARIABLES LIKE 'max_user_connections';
     ```

     或者

     ```
     obclient> SELECT * FROM INFORMATION_SCHEMA.SESSION_VARIABLES WHERE
     VARIABLE_NAME = 'max_user_connections';
     ```

     查询结果如下：

```
+--------------------+-------+
| Variable_name | Value |
+--------------------+-------+
| max_user_connections | 0 |
+--------------------+-------+
1 row in set
```

- 查看租户内普通用户总的最大并发连接数

```
obclient> SELECT * FROM oceanbase.GV$OB_PARAMETERS WHERE NAME LIKE
'_resource_limit_max_session_num';
```

# 说明

隐藏配置项在未修改前（即为默认值）无法通过 `SHOW` 语句查询，只能通过 `GV$OB_PARAMETERS` 视图查询。如果该隐藏配置项已经被修改为非默认值，则可以通过 `SHOW` 语句查询。

查询结果如下：

```
+---------------+----------+-------+--------+-----------
+--------------------------------+-----------+-------
+-------------------------------------------------------------------+---------------
+------------------+--------------+-----------+
| SVR_IP | SVR_PORT | ZONE | SCOPE | TENANT_ID | NAME | DATA_TYPE | VALUE |
INFO | SECTION | EDIT_LEVEL | DEFAULT_VALUE | ISDEFAULT |
+---------------+----------+-------+--------+-----------
+--------------------------------+-----------+-------
+-------------------------------------------------------------------+---------------
+------------------+--------------+-----------+
| 172.xx.xxx.xxx | 2882 | zone1 | TENANT | 1004 |
_resource_limit_max_session_num | INT | 0 | the maximum number of sessions
that can be created concurrently | RESOURCE_LIMIT | DYNAMIC_EFFECTIVE | 0 |
```

```
YES |
+---------------+---------+-------+--------+----------
+------------------------------+----------+-------
+-------------------------------------------------------------------+---------------
+------------------+--------------+-----------+
1 row in set
```

查询结果中，`VALUE` 列对应的值即为对应的配置项的值。

有关视图 `GV$OB_PARAMETERS` 的更多介绍，请参见 GV$OB_PARAMETERS。

3. 设置租户的最大连接数。
   - 设置整个租户的最大连接数

     ```
     obclient> SET GLOBAL max_connections = 1000;
     ```

     执行成功后，需要重启 OBServer 节点才能生效。
   - 设置租户内单个用户的最大并发连接数

     ```
     obclient> SET GLOBAL max_user_connections = 50;
     ```

     执行成功后，需要重启 OBServer 节点才能生效。
   - 设置租户内普通用户总的最大并发连接数

     ```
     obclient> ALTER SYSTEM SET _resource_limit_max_session_num = 100;
     ```

## 说明

如果当前租户同时设置了 `_resource_limit_max_session_num`、`max_user_connections` 和 `max_connections` 的值，则对于某个普通用户，其并发连接数只要达到其中一个值，即无法再建立新的连接；对于管理员用户，其并发连接数只要达到 `max_user_connections` 或 `max_connections` 的其中一个值，即无法再建立新的连接。

# 33.4 Oracle 租户查看本租户相关的最大并发连接数

1. 租户管理员登录到集群的 Oracle 租户。

2. 通过视图查看租户内普通用户的最大并发连接数。

```
obclient> SELECT * FROM SYS.GV$OB_PARAMETERS WHERE NAME LIKE
'_resource_limit_max_session_num';
```

# 说明

隐藏配置项在未修改前（即为默认值）无法通过 `SHOW` 语句查询，只能通过 `GV$OB_PARAMETERS` 视图查询。如果该隐藏配置项已经被修改为非默认值，则可以通过 `SHOW` 语句查询。

查询结果如下：

```
+---------------+----------+-------+--------+-----------
+-------------------------------+-----------+-------
+-------------------------------------------------------------------+---------------
+------------------+--------------+-----------+
| SVR_IP | SVR_PORT | ZONE | SCOPE | TENANT_ID | NAME | DATA_TYPE | VALUE | INFO
| SECTION | EDIT_LEVEL | DEFAULT_VALUE | ISDEFAULT |
+---------------+----------+-------+--------+-----------
+-------------------------------+-----------+-------
+-------------------------------------------------------------------+---------------
+------------------+--------------+-----------+
| 172.xx.xxx.xxx | 2882 | zone1 | TENANT | 1004 | _resource_limit_max_session_num
| INT | 0 | the maximum number of sessions that can be created concurrently |
RESOURCE_LIMIT | DYNAMIC_EFFECTIVE | 0 | YES |
+---------------+----------+-------+--------+-----------
+-------------------------------+-----------+-------
+-------------------------------------------------------------------+---------------
+------------------+--------------+-----------+
1 row in set
```

查询结果中，`VALUE` 列对应的值即为对应的配置项的值。

有关视图 GV$OB_PARAMETERS 的更多介绍，请参见 GV$OB_PARAMETERS。

3. 设置租户内普通用户的最大并发连接数。

```
obclient> ALTER SYSTEM SET "_resource_limit_max_session_num" = 100;
```

# 33.5 相关文档

关于配置项和变量的更多介绍，请参见如下信息：

- 设置参数
- 设置变量
- 查看租户会话
- 终止租户会话