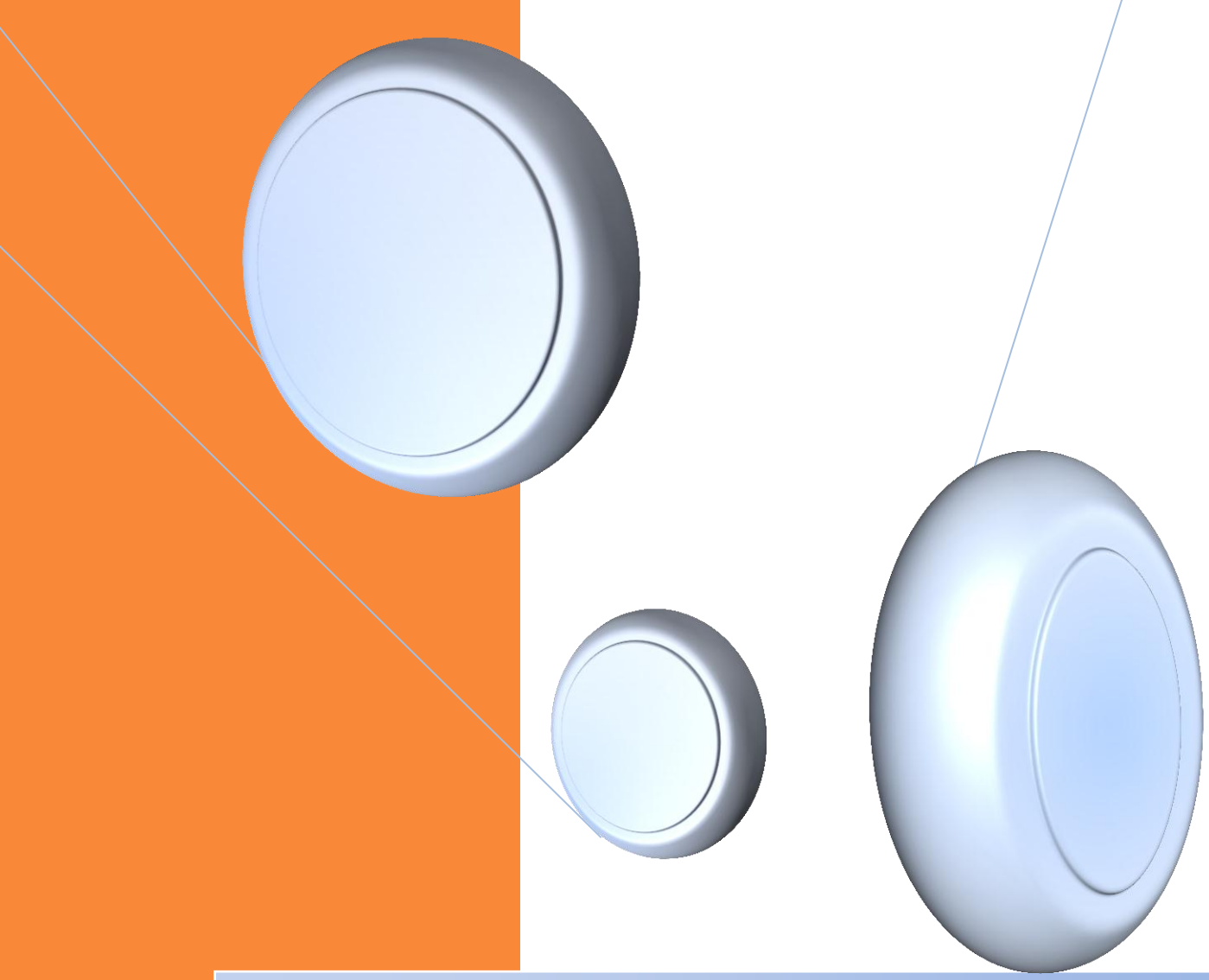


OceanBase 0.4.1
客户端 用户指南
(阿里内部)

文档版本：01

发布日期：2013.10.30

支付宝（中国）网络技术有限公司·OceanBase 团队

前 言

概述

本文档介绍OceanBase数据库客户端的架构和使用方法。

本文档与外部使用的《OceanBase 0.4.1 客户端 用户指南》比较：本文档增加了OB Config和OceanBase Java客户端“oceanbase.jar”模块的介绍。

建议在阅读本文档前，先阅读《OceanBase 0.4.1 客户端 用户指南》。

读者对象

本文档主要适用于：

- 阿里内部开发工程师。
- 阿里内部安装工程师。

通用约定

在本文档中可能出现下列各式，它们所代表的含义如下。

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

修订记录

修改记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本。

联系我们

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

支付宝（中国）网络技术有限公司·OceanBase 团队

地址：杭州市万塘路 18 号黄龙时代广场 B 座；邮编：310099

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

邮箱：alipay-oceanbase-support@list.alibaba-inc.com

新浪微博：http://weibo.com/u/2356115944

技术交流群（阿里旺旺）：853923637

目 录

1 概述

主要介绍客户端的基础知识以及安装包的获取方法。

1.1 客户端简介

OceanBase 客户端主要用于开发人员编程时连接 OceanBase 数据库。

Oceanbase 内置了对 SQL 的支持，用户可以通过 libmysql，JDBC 等方式直接访问 Oceanbase，但由于 OceanBase 是一个分布式数据库，可以由多个节点（MergeServer）同时提供 SQL 服务。而 MySQL 客户端等都是针对单机系统，在连接 OceanBase 时，客户端会绑定其中一台 MergeServer 进行 SQL 操作，而不能有效利用其他 MergeServer 资源。

为了实现了多集群间流量分配和多 MergeServer 间的负载均衡，并给应用开发人员提供一个简单接入方案，我们在 libmysql，JDBC 的基础上封装一个 OceanBase 客户端。

OceanBase 客户端可以根据“流量分配”选择 MergeServer。例如：主集群比例为 40，备集群比例为 60。那么在发送弱一致性读 SQL 语句时，将会有 40%的概率发送至主集群中的 MergeServer，60%的概率发送至备集群中的 MergeServer。

流量分配只涉及弱一致性读请求，所有非弱一致性读请求全部发送至主集群。

说明：

- *流量分配：主要指弱一致性读在主备集群中的分配比例。可以在内部表“__all_cluster”中设置。*
- *弱一致性读请求包括：hint weak sql、无 hint 的 select，除此之外的 SQL 语句类型为非弱一致性读请求。*

OceanBase 客户端主要有以下两种：

- OceanBase C 客户端
  为应用程序提供了“libobsql.so”动态库，这个库在二进制接口上与 mysql 的 libmysqlclient 库完全兼容。
- OceanBase Java 客户端
  提供了提供了符合 Java 标准的 DataSource，Java 应用程序可以使用 OceanBase Java 客户端获得与 OceanBase 服务器交互的连接。

1.2 获取安装包

OceanBase 客户端安装包的获取方式和说明如表 1-1 所示。

说明：*本文档中使用的安装包版本仅为举例，实际请采用最新安装包。*

表 1-1 安装包

类型	安装包	获取地址
OceanBase C 客户端	Linux 版本为 RedHat 5 的安装包： • curl-7.29.0-1.el5.x86_64.rpm • oceanbase-devel-0.4.2.1-1193.el5.x86_64.rpm Linux 版本为 RedHat 6 的安装包： • curl-7.29.0-1.el6.x86_64.rpm • oceanbase-devel-0.4.2.1-1193.el6.x86_64.rpm ***说明：****您可以执行**cat /etc/issue** 命令查看 Linux 版本号。*	https://github.com/alibaba/oceanbase_client 的“C”文件夹中。
OceanBase Java 客户端	jar 包： • commons-lang-2.3.jar • commons-logging-1.1.jar • druid-0.2.12.jar • mysql-connector-java-5.1.14.jar • oceanbase-1.0.1.jar • oceanbase-core-1.1.0.jar “oceanbase-core.jar”的源码：oceanbase_core_src	https://github.com/alibaba/oceanbase_client 的“Java”文件夹中。

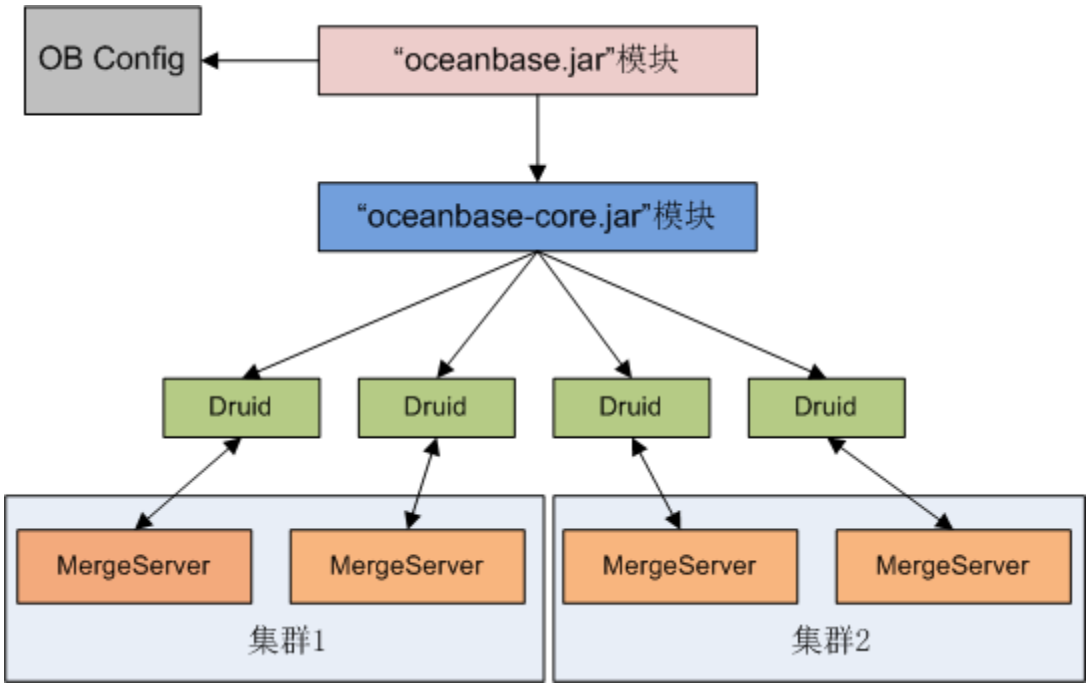
2 OceanBase Java客户端

OceanBase Java 客户端主要用于开发人员编写的 Java 程序连接 OceanBase。

2.1 客户端结构

OceanBase Java 客户端的结构示意图如图 2-1 所示。

图 2-1 结构示意图



OceanBase Java 客户端的结构介绍如下：

- “oceanbase.jar”模块
  OceanBase Java 客户端的主要模块之一，仅供阿里内部使用。主要负责集中化管理各个 Server 的配置项，初始化加载“oceanbase-core.jar”模块，以及完成“oceanbase-core.jar”模块的版本升级任务。
- OB Config
  OceanBase 配置管理中心，仅供阿里内部使用。OB Config 保存了所有的 OceanBase Java 客户端的配置参数，包括：数据库集群地址、用户名、密码、以及数据源本身的配置项。其主要功能为：配置信息的集中化管理与操作，简化 DBA 操作流程，同时方便 OceanbBase Java 客户端的升级。
- MergeSever
  OceanBase 中负责提供 SQL 服务的接口模块，用于处理客户端发送的 SQL 请求。

- Druid
  数据源模块，是阿里巴巴开发的开源的数据库连接池。每个 Druid 数据源对应一个 MergeServer，用于维护到 MergeServer 的连接。关于 Druid 详细介绍请参见“http://code.alibabatech.com/wiki/display/Druid/Home”。
- “oceanbase-core.jar”模块
  OceanBase Java 客户端的主要模块之一，由“oceanbase.jar”模块从 OB Config 中自动缓存到本地。“oceanbase-core.jar”主要封装了 OceanBase 的流量分配、负载均衡的功能，详细请参见《OceanBase 0.4.1 客户端 用户指南》。

OceanBase Java 客户端按功能主要可以分为两种使用场景，如表 2-1 所示。

表 2-1 场景说明

场景	适用人员	所需模块	功能
使用“oceanbase.jar”模块访问 OceanBase	暂时不开源，只适用于阿里巴巴内部人员。	• MergeServer • Druid • oceanbase.jar • OB Config	自动升级或回滚客户端。
使用“oceanbase-core.jar”模块访问 OceanBase	• 阿里巴巴内部人员 • 非阿里巴巴内部人员	• MergeServer • Druid • oceanbase-core.jar	通过手动修改 pom 文件进行升级或回滚客户端。

说明：*Java 应用连接 OceanBase，除了上述两种场景外，还可以直接通过 MergeServer 连接。该方式不通过 OceanBase Java 客户端连接 OceanBase。*

2.2 功能模块

OceanBase Java 客户端的功能模块主要包括“oceanbase.jar”、“oceanbase-core.jar”和“OB Config”。

2.2.1 “oceanbase.jar”模块

“oceanbase.jar”模块提供服务的接口是“OceanBaseDataSourceProxy”。用户通过使用此接口获取数据库的 Connection，进而完成业务的 SQL 执行。

* 配置中心交互

“oceanbase.jar”模块会定期与 OB Config 交互，对比本地与 OB Config 上“oceanbase-core.jar”模块的版本以及升级配置项，从而确认是否需要进行升级。

用户调用“OceanBaseDataSourceProxy”的接口方法“setConfigURL(String configURL)”传入配置中心地址，获取配置中心信息：

- 集群入口地址（Listener MergeServer）。
- 被托管的“oceanbase-core.jar”模块的 jar 包版本号，并缓存该模块文件。
- 集群帐号信息。

获取集群入口地址后，OceanBase Java 客户端和内部表互交，详细流程请参考《OceanBase 0.4.1 客户端 用户指南》。

注意：*“oceanbase-core.jar”模块中的“步骤 1”需要提供的是集群入口地址（Listerer MergeServer），而“oceanbase.jar”模块需要提供的是配置中心的地址。当配置中心互交模块向 OB Config 传入 configUrl，访问配置中心，获取集群入口地址时，如果集群不可用，则使用本地缓存文件。*

* 本地缓存

通过配置中心交互模块，获取集群配置中心信息以及“oceanbase-core.jar”模块后，对以下两个信息进行缓存：

- “oceanbase-core.jar”模块
  缓存文件为“*$USER_HOME*/.obdatasource/*$version*”，其中“*$USER_HOME*”为本地用户路径，“*$version*”为文件名（版本号）。每一个版本对应都会在本地缓存一个对应的缓存。这意味着，每一个版本被第一次使用时，都会在本地缓存一份；再次使用时，将不会再下载。在第一次初始化 Java 应用程序时，“oceanbase-core.jar”模块文件“*$version*”从 OB Config 缓存到本地，而不需要手动导入“oceanbase-core-*XXX*.jar”包。
- 配置信息
  缓存文件为“*$USER_HOME*/.obdatasource/conf”。配置信息每分钟均随定时任务运行而被更新。

本地缓存模块主要是防止在配置中心故障时，导致业务系统无法运行而影响业务的可用性。同时，能够减少对配置中心的访问，在一定程度上，能够缓减配置中心的压力。

* 加载核心服务

将传入的 OceanBase 密码进行解密，解密完成后，将集群入口地址（Listener MergeServer）、用户名以及密码传入“oceanbase-core.jar”模块，并最终调用“ObGroupDataSource”接口类的 init()方法，完成核心模块的初始化工作，详细请参见“2.2.1 “oceanbase-core.jar”模块”。

* 自动升级

默认启动升级检测线程，调度周期为 1 分钟。通过扫描配置中心，查看是否已经开启了升级计划。如果开启，则启动升级任务，并完成“oceanbase-core.jar”模块的动态升级。该任务主要为了解决：在不中断业务服务的前提下，完成修复紧急 bug。

自动升级模式主要有以下几种：

- 按白名单升级
  在 OceanBase 配置中心设置指定需要升级的客户端，其他客户端不升级。例如：某业务系统集群的客户端机器列表为“10.10.10.1，10.10.10.2，10.10.10.3，10.10.10.4”。在 OceanBase 配置中心设置白名单为：10.10.10.1，则单独为 10.10.10.1 升级。
- 按比例升级
  在 OceanBase 配置中心设置设置集群升级的比例，完成该占比机器的升级。
  例如：某业务系统集群的客户端机器列表为“10.10.10.1，10.10.10.2，10.10.10.3，10.10.10.4”。在 OceanBase 配置中心设置升级比例为 50%，则将为 4 台客户端中的 2 台完成升级（理论值）。
  同时，可以通过逐渐变化升级的比例。比如第一次设置 25%，完成集群的四分之一升级；然后通过设置比例为 50%，完成集群的二分之一升级；通过逐步提升，到最终完成集群的所有机器升级至指定版本。
- 回滚
  对于出现的升级错误，将采用回滚操作来完成版本的还原。

“oceanbase-core.jar”模块每次升级时，原数据源和新数据源同时存在，原数据源在 5 分钟后失效。例如：“oceanbase-core.jar”模块从 A 版本升级到 B 版本。

- 5 分钟内，A 版本和 B 版本的数据源同时存在，此时，“oceanbase-core.jar”模块可以直接回滚到版本 A，不能够升级到其他版本。
- 5 分钟后，A 版本失效，支持升级到其他版本。如需回滚到 A 版本，则视为从 B 版本“升级”到 A 版本。

* 客户端汇报

在 Java 应用程序第一次初始化时，将汇报客户端机器的 IP 以及所使用的“oceanbase-core.jar”的版本号。如果“oceanbase-core.jar”升级，则重新汇报。

* 日志规则初始化

初始化“log4j”组件规则，便于日志的输出。默认日志输出路径为“*$USER_HOME*/logs/”。

2.2.2 “oceanbase-core.jar”模块

“oceanbase-core.jar”封装了 OceanBase 数据库的流量分配、负载均衡的功能，详细请《OceanBase 0.4.1 客户端 用户指南》。

2.2.3 OB Config

OB Config 是 OceanBase 客户端的配置管理中心，属于 OceanBase 团队内部使用的客户端模块。

- 集群配置信息存储和管理系统，用于 DBA 集中化管理生产（或日常）环境中，各业务系统使用的集群配置信息。
- 托管 OceanBase Datasource 核心服务的各个版本 jar 包，同时控制业务系统使用的“oceanbase-core.jar”版本号。
- 提供操作界面，便于完成业务系统使用的“oceanbase-core.jar”模块的版本升级。同时，提供业务系统使用的各个版本信息等。

说明：*该系统开发语言为 Java，采用类似 taobao diamond 的架构设计。*

OB Config 支持两种接口：

- HTTP GET：用于 OceanBase Java 客户端获取 OceanBase 应用的 Listener 地址列表。
- HTTP POST：用于修改 OceanBase 应用的 listener 地址列表。

DBA 可以通过 curl 之类的工具直接修改 OceanBase 应用的 Listener 地址。

在 OceanBase 监控中心中提供了 OB Config 的 WEB 界面，可以直接通过该界面进行设置。OB Config 统一管理了多个 OceanBase 的应用集群，以 dataId 作为主键进行应用集群的区分。不同应用的 OB Config 的 URL 地址，只需要修改为对应的集群名即可。

OB Config 对外提供统一的域名访问（oceanbase.alibaba-inc.com）。客户端只需要配置一个 OB Config 服务的 DNS 地址，就会自动定位到某个机房的 OB Config 服务。OB Config 选择 DNS 容错的方式，在两个机房部署 OB Config 服务。这种方式实现比较简单，但是，如果某个机房的 OB Config 服务长期不可用，DNS 还是会查询到该地址。如果要修改 DNS，则需要走一遍“DNS 变更流程”，周期很长。另外，这种方式还有一个好处就是以后迁移 OB Config 到另外一个 IP 可以不需要客户端走重新发布流程。

OB Config 服务的功能如下：

- 管理 OceanBase 集群的 Listener 列表，提供查询和更改操作。
- 提供上传和下载“oceanbase-core.jar”模块文件功能。将“oceanbase-core.jar”模块文件上传到 Ob Config 服务器后，Ob Config 除了保存 jar 文件本身，还需要保存文件的 md5 值。另外，Ob Config 还需

要存储最近上传的多个版本的 jar 文件（建议为 5 个），方便回滚。

说明：*在第一次初始化 Java 应用程序时，通过“oceanbase.jar”模块的“本地缓存模块”从 OB Config 缓存到本地。而不需要手动导入“oceanbase-core-XXX.jar”包。*

- 提供读取最新版本的“oceanbase-core.jar”模块文件的元数据的功能。包括最后更新时间、版本号、md5 值、百分比、允许升级的客户端机器 IP 白名单。
- 管理客户端的升级。当有新版本的客户端出现时，通过配置百分比和允许升级的客户端机器 IP 白名单的方式，通知客户端自动进行版本的升级。

OceanBase Java 客户端首先从 OB Config 读取元数据，并根据版本号信息构造 HTTP 请求下载相应的“oceanbase-core.jar”模块文件，检查“last_update_time_ms”是否比之前更新，执行 md5 校验。如果客户端需要升级，则替换自身依赖的“oceanbase-core.jar”模块文件。

通过访问 OB Config，获取的配置信息如下所示：

```
#dataID 名称。
dataId=financial_history
#OceanBase 主集群主 RootServer 的 IP 地址和 Listener 端口。
clusterAddress=10.209.144.30:2828
#“oceanbase-core.jar”模块的版本。
coreJarVersion=1.1.1
#是否支持自动升级“oceanbase-core.jar”模块。
enableUpdate=true
#获取“oceanbase-core.jar”模块的 IP 地址。
coreJarPath=http://obconsole.test.alibaba-inc.com/jarrepo/get?version=
#升级百分比。
percentage=0
#白名单列表。
whiteList=10.209.11.122,10.209.15.103
#连接 OceanBase 的用户名密码。密码以加密方式显示。
username=admin
password=73cad637e0e8b6e9
#数据源属性配置信息，详细请参考：
#http://dev.mysql.com/doc/refman/5.0/en/connector-j-reference-configuration-properties.html
dsConfig=maxActive:8,minIdle:4,connectionProperties:emulateUnsupportedPstmts=false;characterEncoding=GBK;useServerPrepStmts=false;prepStmtCacheSqlLimit=1000;enableQueryTimeouts=false;useLocalSessionState=false;useLocalTransactionState=false
#MD5 效验码。
MD5=d462b2ea16e0568f309a2b474ab3df7
```

OB Config 操作说明如下：

- 新增 dataId

    1. 登录 OceanBase 配置中心。

2. 选择“集群配置 > 新增配置”，打开配置页面。
3. 输入 dataId、连接 OceanBase 的用户名和密码、0.4 集群地址，并且选择当前版本号，如图 2-2 所示。参数说明如表 2-4 所示。

图 2-2 新增 dataId

表 2-4 新增 dataId

参数	说明
dataId	连接 OB Config 的 URL 的组成部分。由用户自定义。
用户名	连接 OceanBase 的用户名。
密码	连接 OceanBase 的密码。

参数	说明
0.4 集群	0.4 版本 OceanBase 的主 RootServer 和 Listener 端口。多个集群时以英文字符逗号分割，如：10.228.74.201:2828,10.225.35.201:2828
JAR 路径	获取 oceanbase-core-*XXX*.jar 的路径。
当前版本号	oceanbase-core-*XXX*.jar 的版本号。目前最新版本为“1.1.1”。
是否升级	oceanbase-core-*XXX*.jar 版本升级的开关。
百分比	输入一个 0 至 100 之间的整数进行升级。一般设置为 5（即按 5%升级），然后逐步增加该值直至“100”。推荐采用该方式进行升级。 按百分比升级前，建议先选择一台机器按白名单方式进行升级。并在完成升级后，观察这台机器是否正常工作。如果工作正常，则再按百分比升级。
白名单	输入需要升级的机器 IP。当需要升级多台指定的机器时，请用英文状态的逗号隔开。
是否监控	是否在 OB Config 上进行监控升级状态的开关。
数据源配置选项	数据源内部我们已经进行了配置参数优化，因此不建议配置该参数。详细信息请参考“http://dev.mysql.com/doc/refman/5.0/en/connector-j-reference-configuration-properties.html”。

4. 单击“submit”。

- 配置升级选项

1. 登录 OceanBase 配置中心。
2. 选择“集群配置”，进入 dataID 列表。
3. 选择需要升级的 dataId，并在操作列中单击“edit”，打开配置页面。

4. 在“当前版本号”中选择新的版本号，并在“是否升级”单选按钮中选择“是”，然后按白名单升级或百分比升级，如图 2-3 所示。参数说明如表 2-4 所示。

图 2-3 配置升级选项

5. 单击“submit”。

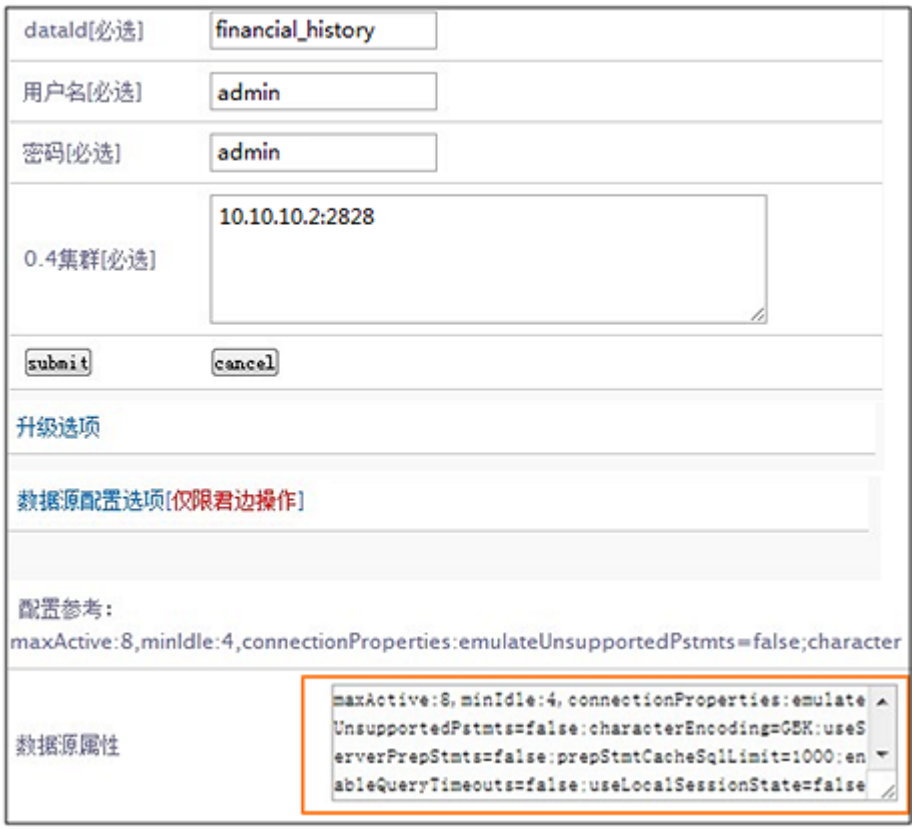
- 配置数据源属性

  目前支持修改的数据源属性分别为：maxActive、minIdle、connectionProperties。

  1. 登录 OceanBase 配置中心。
  2. 选择“集群配置”，进入 dataID 列表。
  3. 选择需要升级的 dataId，并在操作列中单击“edit”，打开配置页面。
  4. 在“数据源属性”中输入数据源属性，如图 2-4 所示。参数说明如表 2-4 所示。

```
maxActive:8,
minIdle:4,
connectionProperties:emulateUnsupportedPstmts=false;characterEncoding=GBK;useServerPrepStmts=false;prepStmtCacheSqlLimit=1000;enableQueryTimeouts=false;useLocalSessionState=false;useLocalTransactionState=false
```

图 2-4 配置数据源属性



5. 单击“submit”。

2.3 访问方式

主要介绍 Java 应用程序访问 OceanBase 的三种方式。

2.3.1 使用“oceanbase.jar”模块访问 OceanBase

“oceanbase.jar”模块属于 OceanBase 团队内部使用的客户端模块，提供服务的 jar 包为“oceanbase-XXX.jar”，其中“XXX”为版本号。

* 基本信息

在使用“oceanbase.jar”模块访问 OceanBase 时，Java 程序需要初始化“oceanbase.jar”模块。在初始化过程中，“oceanbase.jar”模块会完成如下几件事情：

1. “oceanbase.jar”模块访问 OB Config 配置中心，从配置中心获取 OceanBase 数据库中 Listener 的位置信息，同时 OB Config 也会返回“oceanbase-core.jar”模块的最新版本号、是否允许进行主动升级、升级的机器比例、升级白名单等信息。
2. “oceanbase.jar”模块根据这些信息来判断是否要主动升级“oceanbase-core.jar”模块的版本。如果需要升级，则会将“oceanbase-core.jar”模块的新版本缓存到 Java 程序的本地。
3. “oceanbase.jar”模块访问 OceanBase 数据库中 Listener，从而获取所有可供使用的 MergeServer 的地址列表和流量分配信息，并为每个 MergeServer 建立一个 Druid 数据源。
4. “oceanbase.jar”模块根据流量分配的原则，从一个 Druid 数据源中选择一个连接给 Java 程序使用。等到应用断开连接后，该连接重新回到 Druid 数据源中等待下一次使用。

说明：*Listener 是 OceanBase 集群内一个特殊的 MergeServer,主要用来提供获取 MergeServer 列表、集群流量分配的进程。*

目前在 taobao、以及 alipay 的 maven 仓库均可搜索获得发布的客户端。pom 坐标如下：

```
<dependency>
   <groupId>com.alipay.oceanbase</groupId>
   <artifactId>oceanbase</artifactId>
   <version>1.1.1</version>
</dependency>
```

目前 druid 的最新版本为 0.2.25。pom 坐标如下：

```
<dependency>
   <groupId>com.alibaba</groupId>
   <artifactId>druid</artifactId>
   <version>0.2.25</version>
</dependency>
```

说明：*如果使用的是 taobao 的仓库，只需要添加 oceanbase 自身的 pom 信息；如果使用的是 alipay 的仓库，需要再添加 druid 的 pom 信息。*

MYSQL 的驱动版本为 5.1.14。pom 坐标如下：

```
<dependency>
   <groupId>mysql</groupId>
   <artifactId>mysql-connector-java</artifactId>
   <version>5.1.14</version>
</dependency>
```

添加完 pom 信息之后，就添加了工程的依赖关系。那么下面可以进行编码工作了。现在，编码工作主要分为以下两种方式：

- 配置编码

  通过 id 为 dataSource，向 Spring IOC 容器获取配置的 bean 后，即可通过调用方法 getConnection()来使用 OceanBase Java 客户端。

```
<bean id="dataSource" class="com.alipay.oceanbase.OceanbaseDataSourceProxy" init-method="init" destroy-method="destroy" />
<property name="configURL">http://obconsole.test.alibaba-inc.com/ob-config/config.co?dataId=financial_history</property>
</bean>
```

- 硬编码
  通过直接在程序中添加以下语句使用 OceanBase Java 客户端。

```
OceanbaseDataSourceProxy obDS = new OceanbaseDataSourceProxy();
obDS.setConfigURL("http://obconsole.test.alibaba-inc.com/ob-config/config.co?dataId=financial_history");
obDS.init();
Connection conn = obDS.getConnection();
```

configURL 可以分为前缀部分与 dataId 部分。例如：http://obconsole.test.alibaba-inc.com/ob-config/config.co?dataId=financial_history。

- 前缀部分，可以分为以下几种情况：
  - 线下环境：
    http://obconsole.test.alibaba-inc.com/ob-config/config.co?dataId=
  - 线上环境（集团）：
    http://obconfigserver.tbsite.net/ob-config/config.co?dataId=
  - 线上环境（支付宝）：
    http://obconfigserver.db.alipay.com/ob-config/config.co?dataId=
- dataId 部分，由 DBA 操作完成。

在 Druid 数据源内部我们已经进行了配置参数优化，因此，除以下参数外，不建议配置其他的参数项：

```
<!-- Druid 与 MergeServer 的连接。Druid 初始化时的连接数、最小连接数和最大连接数 -->
      <property name="initialSize"    value="1" />
      <property name="minIdle"    value="1" />
      <property name="maxActive"    value="20" />
<!--  建立一条 Druid 与 MergeServer 新连接的超时时间   -->
      <property name="maxWait"    value="60000" />
<!-- SQL 操作 Prepare 语句缓存功能的开关。  -->
      <property name="poolPreparedStatements" value="true" />
<!-- Prepare 语句缓存最多支持缓存的 SQL 语句数量。  -->
      <property name="maxPoolPreparedStatementPerConnectionSize" value="20"/>
```

* 前提条件

- OceanBase 主备集群已经安装且正常运行。
- OB Config 搭建、配置完成且正常运行。
- Java 开发环境以及部署完成，如已经安装 JDK、Eclipse 等。
- 以添加“oceanbase-1.1.1.jar”、“druid-0.2.25.jar”、“mysql-connector-java-5.1.14.jar”的 pom 坐标，或者已获取这三个 jar 包。
- 已经获取“commons-logging-1.1.jar”和“commons-lang-2.3.jar”。

* 操作流程

1. 连接 OceanBase 数据库。
2. 新建 PreparedStatement 语句，来查询__all_server 表中 svr_port 为某个值的 server 信息。
3. 指定查询__all_server 表中 svr_port 为 2600 的 server 信息。
4. 打印查询结果。

* 预期结果

Eclipse Console 界面将打印出所有端口号为 2600 的 server 信息。

* 代码示例

编辑 Java 代码，如下所示：

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
```

```
import java.sql.Statement;

public class OceanBaseDataSourceProxyExample {
	public static void main(String[] args) throws Exception {
		useDataSource();
		System.exit(0);
	}

	public static void useDataSource() throws Exception {
		OceanbaseDataSourceProxy datasource = new
OceanbaseDataSourceProxy();

	datasource.setConfigURL("http://obconsole.test.alibaba-inc.com/ob-config/config.co?
dataId=alipay_junbian");
		datasource.setUsername("admin");
		datasource.setPassword("admin");
		datasource.init();
		Connection conn = datasource.getConnection();
		PreparedStatement pstmt = conn
				.prepareStatement("select * from __all_server where
svr_port = ?");
		pstmt.setInt(1, 2600);
		ResultSet rs = pstmt.executeQuery();
		ResultSetMetaData rSetMetaData = rs.getMetaData();
		for (int i = 1; i <=	rSetMetaData.getColumnCount(); i++) {
			System.out.print("|" + rSetMetaData.getColumnName(i) + "");
		}
		System.out.println();
		while (rs.next()) {
			for (int i = 1; i <=	rSetMetaData.getColumnCount(); i++) {
				System.out.print("|" + rs.getString(i) + "");
			}
			System.out.println();
		}
		if (rs != null) {
			rs.close();
		}
		if (pstmt != null) {
			pstmt.close();
		}
		if (conn != null) {
			conn.close();
		}
		datasource.destroy();
```

```
            }
}
```

如果 Eclipse Console 界面输出如下信息则表明运行正常。其中第一行为“__all_server”表的 schema 信息，而第二行才是真正的输出结果行。

```
|gm_create|gm_modify|cluster_id|svr_type|svr_ip|svr_port|inner_port|svr_role|svr_version
|2013-05-27 21:54:30.08001|2013-05-29
17:54:45.81237|0|chunkserver|10.209.144.31|2600|0|0|0.4.1.2_13189M(Apr 10 2013
12:40:32)
```

2.3.2 使用“oceanbase-core.jar”模块访问 OceanBase

在阿里内部，使用“oceanbase-core.jar”模块访问 OceanBase 时，可以通过手动修改 pom.xml 中“oceanbase-core.jar”模块的版本，来达到升级、回滚的目的，如下所示：

```
<dependency>
   <groupId>com.alipay.oceanbase</groupId>
   <artifactId>oceanbase-core</artifactId>
   <version>0.0.1</version>
</dependency>
```

2.3.3 使用 MergeServer 直接访问 OceanBase

该方法没有通过 OceanBase Java 客户端连接 OceanBase，适用于非阿里内部人员和阿里内部人员，详细请参见《OceanBase 0.4.1 客户端 用户指南》。

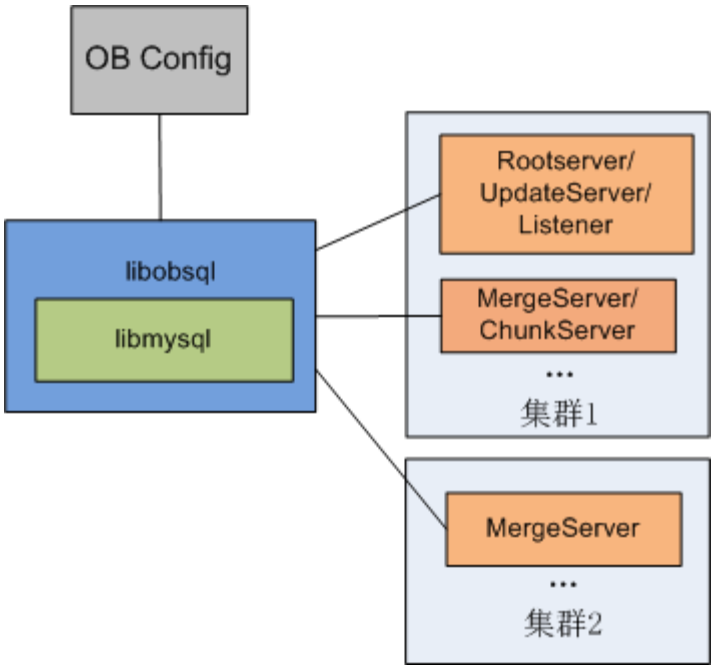
3 OceanBase C 客户端

OceanBase C 客户端主要用于开发人员编写 C 程序连接 OceanBase 数据库。它通过动态库“libobsql.so”的形式提供给用户使用。

3.1 客户端结构

OceanBase C 客户端的结构示意图如图 3-1所示。

图 3-1 结构示意图



OceanBase C 客户端的结构介绍如下：

- OB Config
  OceanBase 配置管理中心，仅供阿里内部使用。主要负责客户端获取 OceanBase 集群地址、管理多个版本客户端库和元数据和通过配置项管理客户端的升级等功能。用户只需要获取到 OB Config 的地址以及 OceanBase 的用户名和密码就能获取到数据库的访问地址以及流量分配等信息，从而初始化整个 OceanBase C 客户端。如果您需要详细了解 OB Config 请参见“2.2.3 OB Config”。
- libmysqlclient
  MySql 提供的 C 语言客户端库，用来访问 MySql 数据库。
- libobsql
  OceanBase 提供的 C 语言客户端库。libobsql 封装了 libmysqlclient，它实现了所有 libmysqlclient 的接口。用户可以通过使用“LD_PRELOAD”环

境变量，调用“libmysqlclient”动态库中的接口，但是实际上是调用的 libobsql 的接口。

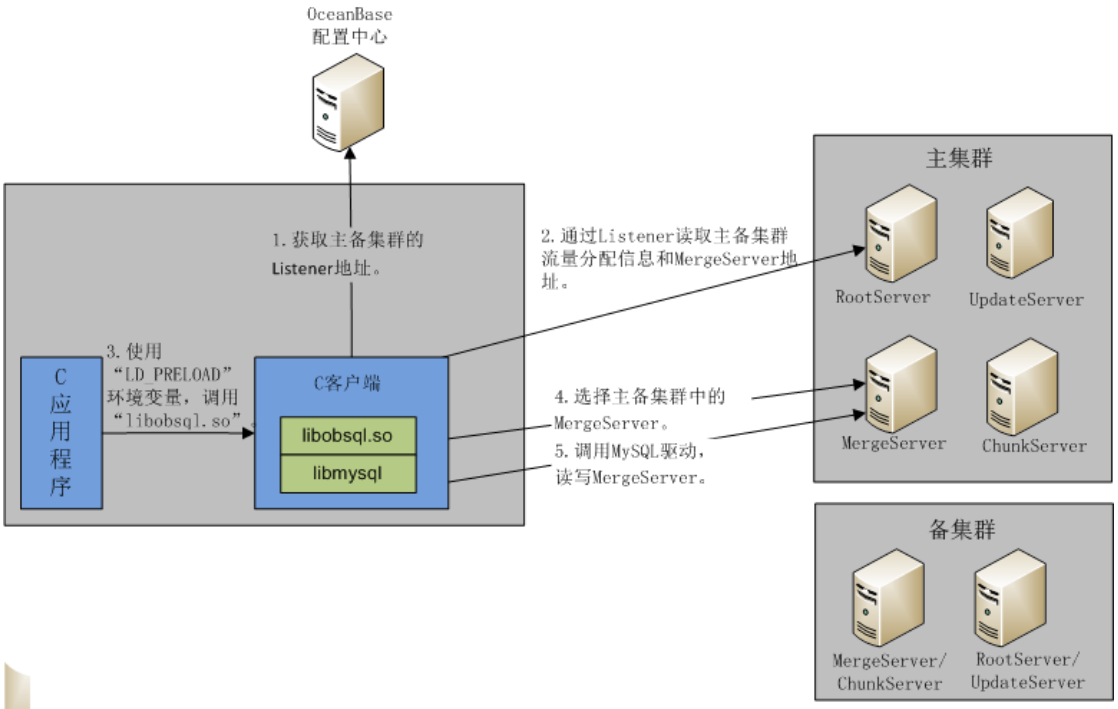
- RootServer/UpdateServer/MergeServer/ChunkServer
  OceanBase 集群的组成模块，具体功能介绍请参见《OceanBase 0.4.1 描述》。
- Listener
  OceanBase 集群内的特殊角色，是客户端获取访问 OceanBase 中 MergeServer 列表、集群流量分配的接口。

3.2 访问流程

应用程序可以使用 libmysqlclient 提供的 API 接口访问 OceanBase，通过 libmysqlclient 编译的应用程序无需重新编译即可连接 OceanBase。

OceanBase C 客户端的访问流程如图 3-2 所示。

图 3-2 访问流程



3.3 安装

假设本地计算机的用户为 **sqluser**。安装 OceanBase C 客户端的操作步骤如下：

1. 以 **sqluser** 用户登录本地计算机。
2. 执行以下命令，检出 Oceanbase 源码。
   git clone https://github.com/alibaba/oceanbase oceanbase_install

3. 执行以下命令，进入“oceanbase_0.4”分支。
   git checkout oceanbase_0.4
4. 执行以下命令，进入 OceanBase C 客户端安装包的存放目录。
   cd ~/oceanbase_install/client_package/c
5. 执行以下命令，安装依赖库 curl。
   sudo rpm -Uvh curl-7.29.0-1.el6.x86_64.rpm
6. 执行以下命令，安装 OceanBase C 客户端。
   sudo rpm -Uvh oceanbase-devel-0.4.2.1-1193.el6.x86_64.rpm
7. 使用 **vi** 编辑器在“~/.bashrc”文件中添加以下内容。

```
export LD_PRELOAD=/home/admin/oceanbase/lib/libobsql.so.0.0.0
export OB_SQL_CONFIG_DIR=/home/admin/oceanbase/etc/
```

8. 执行以下命令，使环境变量生效。
   source ~/.bashrc
9. 使用 **vi** 编辑器修改“~/oceanbase/etc/libobsql.conf”文件，参数说明如表 3-1 所示。

```
logfile=/tmp/obsql.log
#initurl=http://10.232.102.182:8080/diamond-server/config.co?dataId=197
loglevel=DEBUG
minconn=2
maxconn=50
ip=10.10.10.2
port=2828
username=admin
passwd=admin
```

表 3-1 参数说明

参数名称	说明
logfile	客户端日志路径。
initurl	OB Config 的 URL 地址，向 DBA 申请获取。需要搭建 OB Config，暂时仅支持阿里巴巴内部人员使用。 使用 OB Config 获取主备集群的 Listener 时，需要配置。
minconn	客户端到 MergeServer 的最小连接数。

参数名称	说明
maxconn	客户端到 MergeServer 的最大连接数。
ip	主集群中主 RootServer 的 IP。 使用 OB Config 获取主备集群的 Listener 时，不需要配置。
port	Listener 的 MySQL 的协议端口。 使用 OB Config 获取主备集群的 Listener 时，不需要配置。
username	连接 OceanBase 的用户名。 缺省值：admin
passwd	连接 OceanBase 的密码。 缺省值：admin

说明：*OceanBase C 客户端的使用方法请参见《OceanBase 0.4.1 客户端 用户指南》。*

4 附录

4.1 监控系统采集引擎计算规则

本小节主要介绍监控系统采集引擎计算规则，包括表达式求值计算、表达式对比计算、表达式对比计算且求采集周期内平均值和无计算。

4.1.1 表达式求值计算

该计算类型仅仅只针对某个指标的表达式进行计算并获得计算后的具体值。如缓存命中率：block_cache_hit / (block_cache_hit + block_cache_miss)。

详细规则条目如下：

Item Name	Item Compute Express	Item Type
memory_total	memory_used_default + memory_used_network + memory_used_thread_buffer + memory_used_tablet + memory_used_bi_cache + memory_used_block_cache + memory_used_bi_cache_unserving + memory_used_block_cache_unserving + memory_used_join_cache + memory_used_sstable_row_cache + memory_used_merge_buffer + memory_used_merge_split_buffer	express
block_index_cache_miss	block_index_cache_miss - block_index_cache_miss_orgi	express
tablet_merge_percent	old_ver_merged_tablets_num / old_ver_tablets_num	express
split_count	new_ver_tablets_num - old_ver_merged_tablets_num	express

Item Name	Item Compute Express	Item Type
sql_active_session	sql_succ_login_count - sql_logout_count + sql_fail_login_count	express
location_cache_percent	location_cache_hit / (location_cache_hit + location_cache_miss)	express
sql_ps_allocator_count	sql_ps_allocator_count - 0	express

4.1.2 表达式对比计算

该计算类型需要将当前采集周期采集所获得的某个指标值与上一个采集周期所采集到的该指标值进行求差计算并最终获得计算后的具体值。如 SELECT 请求平均耗时：(sql_select_time - sql_select_time_orgi) / (sql_select_count - sql_select_count_orgi)。

- sql_select_time：本次采集的指标值
- sql_select_time_orgi：上次采集的指标值
- sql_select_count：本次采集的指标值
- sql_select_count_orgi：上次采集的指标值

详细规则条目如下：

Item Name	Item Compute Express	Item Type
sql_avg_select_time	(sql_select_time - sql_select_time_orgi) / (sql_select_count - sql_select_count_orgi)	avg
sql_avg_insert_time	(sql_insert_time - sql_insert_time_orgi) / (sql_insert_count - sql_insert_count_orgi)	avg
sql_avg_replace_time	(sql_replace_time-sql_replace_time_orgi) / (sql_replace_count - sql_replace_count_orgi)	avg

Item Name	Item Compute Express	Item Type
sql_avg_update_time	(sql_update_time - sql_update_time_orgi) / (sql_update_count - sql_update_count_orgi)	avg
sql_avg_delete_time	(sql_delete_time - sql_delete_time_orgi) / (sql_delete_count - sql_delete_count_orgi)	avg
avg_get_time	(get_time - get_time_orgi) / (get_count - get_count_orgi)	avg
avg_scan_time	(scan_time - scan_time_orgi) / (scan_count - scan_count_orgi)	avg
avg_apply_time	(apply_time - apply_time_orgi) / (apply_count - apply_count_orgi)	avg
avg_batch_time	(batch_time - batch_time_orgi) / (batch_count - batch_count_orgi)	avg
avg_merge_time	(merge_time - merge_time_orgi) / (merge_count - merge_count_orgi)	avg
block_cache_hit	block_cache_hit - block_cache_hit_orgi	avg
block_cache_miss	block_cache_miss - block_cache_miss_orgi	avg
block_index_cache_hit	block_index_cache_hit - block_index_cache_hit_orgi	avg
sstable_row_cache_hit	sstable_row_cache_hit - sstable_row_cache_hit_orgi	avg

Item Name	Item Compute Express	Item Type
sstable_row_cache_miss	sstable_row_cache_miss - sstable_row_cache_miss_orgi	avg
avg_get_time	(get_time - get_time_orgi) / (get_count - get_count_orgi)	avg
avg_scan_time	(scan_time - scan_time_orgi) / (scan_count - scan_count_orgi)	avg
block_cache_percent	(block_cache_hit - block_cache_hit_orgi) / (block_cache_hit - block_cache_hit_orgi + block_cache_miss - block_cache_miss_orgi)	avg
block_index_cache_percent	(block_index_cache_hit - block_index_cache_hit_orgi) / (block_index_cache_hit - block_index_cache_hit_orgi + block_index_cache_miss - block_index_cache_miss_orgi)	avg
sstable_row_cache_percent	(sstable_row_cache_hit - sstable_row_cache_hit_orgi) / (sstable_row_cache_hit - sstable_row_cache_hit_orgi + sstable_row_cache_miss - sstable_row_cache_miss_orgi)	avg
queue_wait_time	(queue_wait_time - queue_wait_time_orgi) / (request_count - request_count_orgi)	avg
avg_get_event_time	(get_event_time - get_event_time_orgi) / (get_event_count - get_event_count_orgi)	avg

Item Name	Item Compute Express	Item Type
avg_scan_event_time	(scan_event_time - scan_event_time_orgi) / (scan_event_count - scan_event_count_orgi)	avg
location_cache_hit	location_cache_hit - location_cache_hit_orgi	avg
location_cache_miss	location_cache_miss - location_cache_miss_orgi	avg
sql_avg_modify_time	(sql_insert_time - sql_insert_time_orgi + sql_replace_time - sql_replace_time_orgi + sql_update_time - sql_update_time_orgi + sql_delete_time - sql_delete_time_orgi) / (sql_insert_count - sql_insert_count_orgi + sql_replace_count - sql_replace_count_orgi + sql_update_count - sql_update_count_orgi + sql_delete_count - sql_delete_count_orgi)	avg

4.1.3 采集周期内平均值

在“1.2 表达式对比计算”中的计算结束后，会除以采集周期，从而得到当前采集周期之内的一个相对平均值。采集周期可以用户自定义，默认值为“1min”。如请求行数：(sstable_get_rows - sstable_get_rows_orgi) + (sstable_scan_rows - sstable_scan_rows_orgi)。

- sstable_get_rows：本次采集的指标值
- sstable_get_rows_orgi：上次采集的指标值
- sstable_scan_rows：本次采集的指标值
- sstable_scan_rows_orgi：上次采集的指标值

详细规则条目如下：

Item Name	Item Compute Express	Item Type
commit_log_size	commit_log_size - commit_log_size_orgi	all
sstable_disk_io_num	sstable_disk_io_num - sstable_disk_io_num_orgi	all
sstable_disk_io_bytes	sstable_disk_io_bytes - sstable_disk_io_bytes_orgi	all
rpc_bytes_in	rpc_bytes_in - rpc_bytes_in_orgi	all
rpc_bytes_out	rpc_bytes_out - rpc_bytes_out_orgi	all
succ_get_count	succ_get_count - succ_get_count_orgi	all
succ_scan_count	succ_scan_count - succ_scan_count_orgi	all
fail_get_count	fail_get_count - fail_get_count_orgi	all
fail_scan_count	fail_scan_count - fail_scan_count_orgi	all
migrate_count	migrate_coun t- migrate_count_orgi	all
copy_count	copy_count - copy_count_orgi	all
rpc_bytes_in	rpc_bytes_in - rpc_bytes_in_orgi	all
rpc_bytes_out	rpc_bytes_out - rpc_bytes_out_orgi	all

Item Name	Item Compute Express	Item Type
request_count	request_count - request_count_orgi	all
request_rows	(sstable_get_rows - sstable_get_rows_orgi) + (sstable_scan_rows - sstable_scan_rows_orgi)	all
get_count	get_count - get_count_orgi	all
scan_count	scan_count - scan_count_orgi	all
sql_insert_count	sql_insert_count - sql_insert_count_orgi	all
sql_replace_count	sql_replace_count - sql_replace_count_orgi	all
sql_update_count	sql_update_count - sql_update_count_orgi	all
sql_delete_count	sql_delete_count - sql_delete_count_orgi	all
sql_succ_query_count	(sql_succ_query_count - sql_succ_query_count_orgi) + (sql_succ_exec_count - sql_succ_exec_count_orgi)	all
sql_fail_query_count	sql_fail_query_count - sql_fail_query_count_orgi	all
sstable_disk_io_num	sstable_disk_io_num - sstable_disk_io_num_orgi	all
sstable_disk_io_bytes	sstable_disk_io_bytes - sstable_disk_io_bytes_orgi	all

Item Name	Item Compute Express	Item Type
sql_select_count	sql_select_count - sql_select_count_orgi	all
all_sql_select_count	sql_select_count - sql_select_count_orgi	all
sql_succ_prepare_count	sql_succ_prepare_count - sql_succ_prepare_count_orgi	all
sql_fail_prepare_count	sql_fail_prepare_count - sql_fail_prepare_count_orgi	all
sql_succ_exec_count	sql_succ_exec_count - sql_succ_exec_count_orgi	all
sql_fail_exec_count	sql_fail_exec_count - sql_fail_exec_count_orgi	all
sql_succ_close_count	sql_succ_close_count - sql_succ_close_count_orgi	all
sql_fail_close_count	sql_fail_close_count - sql_fail_close_count_orgi	all
rpc_bytes_out	rpc_bytes_out - rpc_bytes_out_orgi	all
rpc_bytes_in	rpc_bytes_in - rpc_bytes_in_orgi	all
sql_modify_count	(sql_insert_count - sql_insert_count_orgi) + (sql_replace_count - sql_replace_count_orgi) + (sql_update_count - sql_update_count_orgi) + (sql_delete_count - sql_delete_count_orgi)	all

Item Name	Item Compute Express	Item Type
block_cache_hit_per_second	block_cache_hit - block_cache_hit_orgi	all
block_cache_miss_per_second	block_cache_miss - block_cache_miss_orgi	all
block_index_cache_hit_per_second	block_index_cache_hit - block_index_cache_hit_orgi	all
block_index_cache_miss_per_second	block_index_cache_miss - block_index_cache_miss_orgi	all
sstable_row_cache_hit_per_second	sstable_row_cache_hit - sstable_row_cache_hit_orgi	all
sstable_row_cache_miss_per_second	sstable_row_cache_miss - sstable_row_cache_miss_orgi	all
get_event_count	get_event_count - get_event_count_orgi	all
scan_event_count	scan_event_count - scan_event_count_orgi	all
rpc_bytes_in	rpc_bytes_in - rpc_bytes_in_orgi	all
rpc_bytes_out	rpc_bytes_out - rpc_bytes_out_orgi	all
get_count	get_count - get_count_orgi	all
scan_count	scan_count - scan_count_orgi	all
apply_count	apply_count - apply_count_orgi	all

Item Name	Item Compute Express	Item Type
batch_count	batch_count - batch_count_orgi	all
merge_count	merge_count - merge_count_orgi	all
get_bytes	get_bytes - get_bytes_orgi	all
scan_bytes	scan_bytes - scan_bytes_orgi	all

4.1.4 无计算

仅仅采集指标值，不进行任何操作。

详细规则条目如下：

Item Name	Item Compute Express	Item Type
sql_logout_count	-	none
sql_succ_login_count	-	none
all_table_count	-	none
all_tablet_count	-	none
all_row_count	-	none
all_data_size	-	none
old_ver_tablets_num	-	none
memory_used_block_cache	-	none
memory_used_bi_cache	-	none

Item Name	Item Compute Express	Item Type
memory_used_sstable_row_cache	-	none
memory_used_join_cache	-	none
total_rows	-	none
active_total_rows	-	none
frozen_total_rows	-	none
memory_limit	-	none
memory_total	-	none
memtable_total	-	none
active_memtable_limit	-	none
active_memtable_total	-	none
frozen_memtable_limit	-	none
frozen_memtable_total	-	none
load1	-	none
load5	-	none
load15	-	none
cpu_sys	-	none
cpu_user	-	none
cpu_iowait	-	none

Item Name	Item Compute Express	Item Type
mem_used	-	none
mem_free	-	none
swap	-	none
network_in	-	none
network_out	-	none
ms_memory_limit	-	none
ms_memory_total	-	none
ms_memory_parser	-	none
ms_memory_transformer	-	none
ms_memory_ps_plan	-	none
ms_memory_rpc_request	-	none
ms_memory_sql_array	-	none
ms_memory_expression	-	none
ms_memory_row_store	-	none
ms_memory_session	-	none

4.2 数据展示计算规则

主要为了解决在数据展示时，单位不一致的情况。如采集的内存数值，是一个以 Byte 为单位的值。而这样的数值，不适合查看。所以根据配置的规则，将该值转换为以 GB 为单位的值。

详细配置条目如下：

Item Name	Item Show Express	Value Type
sql_avg_select_time	-	微妙/次
sql_avg_insert_time	-	微妙/次
sql_avg_replace_time	-	微妙/次
sql_avg_update_time	-	微妙/次
sql_avg_delete_time	-	微妙/次
avg_get_time	-	微妙/次
avg_scan_time	-	微妙/次
avg_apply_time	-	微妙/次
avg_batch_time	-	微妙/次
avg_merge_time	-	微妙/次
block_cache_hit	-	次/分
block_cache_miss	-	次/分
block_index_cache_hit	-	次/分
sstable_row_cache_hit	-	次/分
sstable_row_cache_miss	-	次/分
avg_get_time	-	微秒/次
avg_scan_time	-	微秒/次
block_cache_percent	block_cache_percent*100	%

Item Name	Item Show Express	Value Type
block_index_cache_percent	block_index_cache_percent*100	%
sstable_row_cache_percent	sstable_row_cache_percent*100	%
queue_wait_time	-	微秒/请求
avg_get_event_time	-	微秒/次
avg_scan_event_time	-	微秒/次
location_cache_hit	-	次/分
location_cache_miss	-	次/分
sql_avg_modify_time	-	微妙/次
memory_total	memory_total/(1024*1024*1024)	GB
tablet_merge_percent	-	%
split_count	-	个
sql_active_session	-	个
location_cache_percent	location_cache_percent*100	%
sql_ps_allocator_count	-	个
block_index_cache_miss	-	-
commit_log_size	commit_log_size/1024	KB/秒
sstable_disk_io_num	-	个/秒
sstable_disk_io_bytes	-	Byte/秒

Item Name	Item Show Express	Value Type
rpc_bytes_in	-	KB/秒
rpc_bytes_out	-	KB/秒
succ_get_count	-	次/秒
succ_scan_count	-	次/秒
fail_get_count	-	次/秒
fail_scan_count	-	次/秒
migrate_count	-	次/秒
copy_count	-	次/秒
rpc_bytes_in	rpc_bytes_in/1024	Byte/秒
rpc_bytes_out	rpc_bytes_out/1024	Byte/秒
request_count	-	次/秒
request_rows	-	行/秒
get_count	-	次/秒
scan_count	-	次/秒
sql_insert_count	-	次/秒
sql_replace_count	-	次/秒
sql_update_count	-	次/秒
sql_delete_count	-	次/秒

Item Name	Item Show Express	Value Type
sql_succ_query_count	-	次/秒
sql_fail_query_count	-	次/秒
sstable_disk_io_num	-	个/秒
sstable_disk_io_bytes	sstable_disk_io_bytes/1024	KB/秒
sql_select_count	-	次/秒
all_sql_select_count	-	次/秒
sql_succ_prepare_count	-	次/秒
sql_fail_prepare_count	-	次/秒
sql_succ_exec_count	-	次/秒
sql_fail_exec_count	-	次/秒
sql_succ_close_count	-	次/秒
sql_fail_close_count	-	次/秒
rpc_bytes_out	rpc_bytes_out/1024	KB/秒
rpc_bytes_in	rpc_bytes_in/1024	KB/秒
sql_modify_count	-	次/秒
block_cache_hit_per_second	-	次/秒
block_cache_miss_per_second	-	次/秒
block_index_cache_hit_per_second	-	次/秒

Item Name	Item Show Express	Value Type
block_index_cache_miss_per_second	-	次/秒
sstable_row_cache_hit_per_second	-	次/秒
sstable_row_cache_miss_per_second	-	次/秒
get_event_count	-	次/秒
scan_event_count	-	次/秒
rpc_bytes_in	rpc_bytes_in/1024	KB/秒
rpc_bytes_out	rpc_bytes_out/1024	KB/秒
get_count	-	次/秒
scan_count	-	次/秒
apply_count	-	次/秒
batch_count	-	次/秒
merge_count	-	次/秒
get_bytes	get_bytes/1024	KB/秒
scan_bytes	-	Byte/秒
sql_logout_count	-	次
sql_succ_login_count	-	次/分
all_table_count	-	个

Item Name	Item Show Express	Value Type
all_tablet_count	-	个
all_row_count	-	行
all_data_size	all_data_size/(1024*1024*1024)	GB
old_ver_tablets_num	-	个
memory_used_block_cache	memory_used_block_cache/(1024*1024*1024)	GB
memory_used_bi_cache	memory_used_bi_cache/(1024*1024)	MB
memory_used_sstable_row_cache	memory_used_sstable_row_cache/(1024*1024*1024)	GB
memory_used_join_cache	memory_used_join_cache/(1024*1024)	MB
total_rows	total_rows/1000000	百万行
active_total_rows	active_total_rows/1000000	百万行
frozen_total_rows	frozen_total_rows/1000000	百万行
memory_limit	memory_limit/(1024*1024*1024)	GB
memory_total	memory_total/(1024*1024*1024)	GB
memtable_total	memtable_total/(1024*1024*1024)	GB
active_memtable_limit	active_memtable_limit/(1024*1024*1024)	GB

Item Name	Item Show Express	Value Type
active_memtable_total	active_memtable_total/(1024*1024*1024)	GB
frozen_memtable_limit	frozen_memtable_limit/(1024*1024*1024)	GB
frozen_memtable_total	frozen_memtable_total/(1024*1024*1024)	GB
load1	-	-
load5	-	-
load15	-	-
cpu_sys	-	-
cpu_user	-	-
cpu_iowait	-	-
mem_used	-	GB
mem_free	-	GB
swap	-	-
network_in	-	KB/s
network_out	-	KB/s
ms_memory_limit	ms_memory_limit/(1024*1024*1024)	次/秒
ms_memory_total	ms_memory_total/(1024*1024*1024)	GB

Item Name	Item Show Express	Value Type
ms_memory_parser	ms_memory_parser/(1024*1024*1024)	GB
ms_memory_transformer	ms_memory_transformer/(1024*1024*1024)	GB
ms_memory_ps_plan	ms_memory_ps_plan/(1024*1024*1024)	GB
ms_memory_rpc_request	ms_memory_rpc_request/(1024*1024*1024)	GB
ms_memory_sql_array	ms_memory_sql_array/(1024*1024*1024)	GB
ms_memory_expression	ms_memory_expression/(1024*1024*1024)	GB
ms_memory_row_store	ms_memory_row_store/(1024*1024*1024)	GB
ms_memory_session	ms_memory_session/(1024*1024*1024)	GB