ALIBABA CLOUD

阿里云

云数据库HBase版

HBase 增强版(Lindorm)

文档版本：20210617

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

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 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 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 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` *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

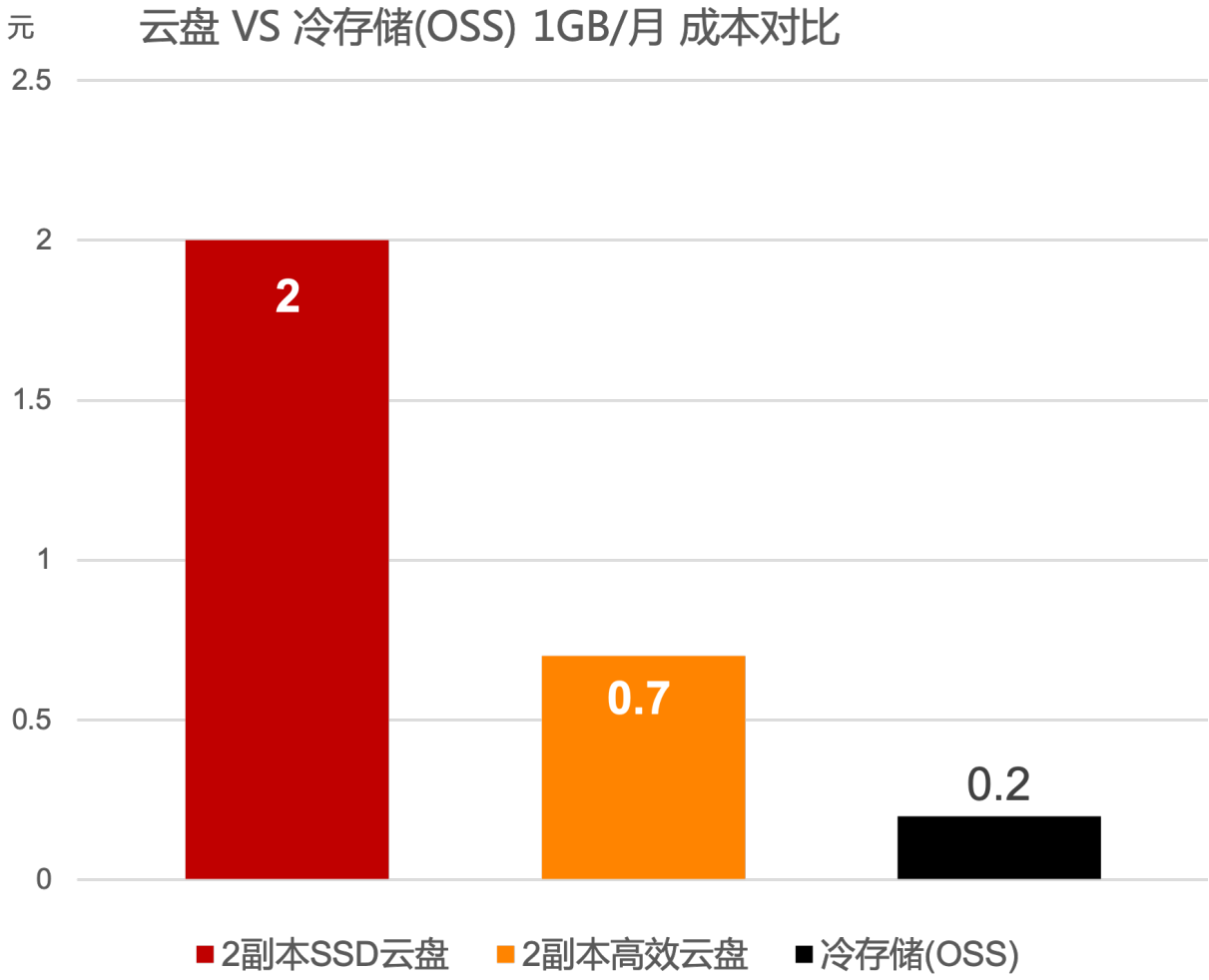
1.企业特性

1.1. 使用冷存储

阿里云HBase针对冷数据存储的场景，提供一种新的冷存储介质，其存储成本仅为高效云盘的1/3，写入性能与云盘相当，并能保证数据随时可读。

背景信息

用户可以在购买云HBase实例时选择冷存储作为一个附加的存储空间，并通过建表语句指定将冷数据存放在冷存储介质上面，同时HBase增强版还基于冷存储实现了在同一张表内的冷热分离功能，能够自动将表中热数据放到读写速度快的热存储中，而把不常访问的数据放到冷存储中降低成本。



注意事项

- 冷存储的读IOPS能力很低（每个节点上限为25），所以冷存储只适合低频查询场景。
- 写入吞吐上，冷存储和基于高效云盘的热存储相当，可以放心写入数据。
- 冷存储不适合并发大量读请求，如果有这种行为可能会导致请求异常。
- 购买冷存储空间特别大的客户可以酌情调整 “读IOPS 能力”，可以工单咨询。
- 建议平均每个core节点管理冷数据不要超过30T。如果需要单个core节点管理更大数据量的冷数据，可以

工单咨询优化建议。

前提条件

HBase增强版2.1.8版本以上才支持冷存储，如果低于此版本在开通过程中会自动升级到最新版本，客户端依赖要求AliHBase-Connector 1.0.7/2.0.7以上，Shell要求alihbase-2.0.7-bin.tar.gz以上。

使用场景

冷存储适用于数据归档、访问频率较低的历史数据等各种冷数据场景。

开通冷存储

方式一：创建HBase增强版集群时，可在购买页面选择是否选购冷存储和冷存储的容量请参考购买集群。

方式二：

1. 登录云数据库HBase控制台。
2. 在**集群列表**页面，单击集群实例名称，进入集群详情页。
3. 在实例信息页面在左侧导航栏选择**冷存储**。
4. 单击**立即开通**。

> **说明** 只有HBase增强版2.1.8版本以上才支持冷存储，如果低于此版本在开通过程中会自动升级到最新版本。

使用冷存储

HBase增强版支持在ColumnFamily（列簇）级别设置存储属性。可以将表的某个列簇（或者所有列簇）的Storage设为冷存储。一旦设置为冷存储后，那么这个表中该列簇（或者所有列簇）的数据，都会存储在冷存储中，并不会占用该集群的HDFS空间。设置的方法可以在建表时指定，也可以在建好表后，对列簇的属性进行修改。

建表和修改表属性均可以使用Java API和HBase shell完成，在使用Java API前请按照使用 Java API访问增强版集群文档完成Java SDK安装和参数配置。在使用HBase shell前，请按照使用HBaseue Shell访问增强版集群文档完成Shell的下载和配置。

建表时指定冷存储

HBase Shell

```
hbase(main):001:0> create 'coldTable', {NAME => 'f', STORAGE_POLICY => 'COLD'}
```

Java API

```
Admin admin = connection.getAdmin();
HTableDescriptor descriptor = new HTableDescriptor(TableName.valueOf("coldTable"));
HColumnDescriptor cf = new HColumnDescriptor("f");
cf.setValue("STORAGE_POLICY", AliHBaseConstants.STORAGETYPE_COLD);
descriptor.addFamily(cf);
admin.createTable(descriptor);
```

修改表属性指定冷存储

如果表已经建立后，可以通过修改表中列簇的属性来设置冷存储的列簇。如果这个列簇中已经有数据，那么只有在major compaction之后，数据才会进入到冷存储。

HBase Shell

```
hbase(main):011:0> alter 'coldTable', {NAME=>'f', STORAGE_POLICY => 'COLD'}
```

Java API

```
Admin admin = connection.getAdmin();
TableName tableName = TableName.valueOf("coldTable");
HTableDescriptor descriptor = admin.getTableDescriptor(tableName);
HColumnDescriptor cf = descriptor.getFamily("f".getBytes());
// 设置表的存储类型为冷存储
cf.setValue("STORAGE_POLICY", AliHBaseConstants.STORAGETYPE_COLD);
admin.modifyTable(tableName, descriptor);
```

更改表属性为热存储

如果表的列存储类型为冷存储，想更改为热存储，可以通过修改表属性的方式实现。如果这个列簇中已经有数据，那么只有在major compaction之后，数据才会回到热存储中

HBase Shell

```
hbase(main):014:0> alter 'coldTable', {NAME=>'f', STORAGE_POLICY => 'DEFAULT'}
```

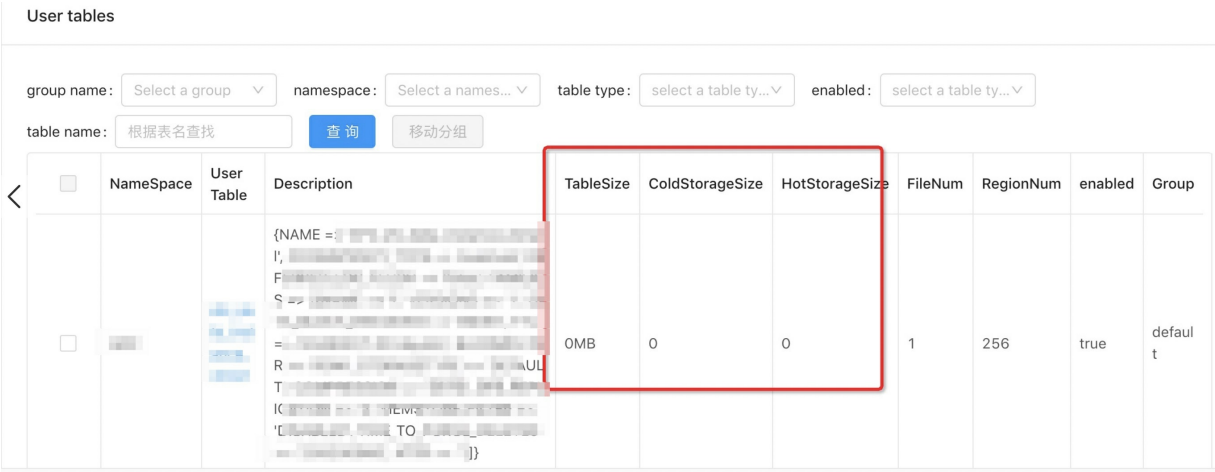
Java API

```
// 参见创建连接: https://help.aliyun.com/document_detail/119570.html
Admin admin = connection.getAdmin();
TableName tableName = TableName.valueOf("coldTable");
HTableDescriptor descriptor = admin.getTableDescriptor(tableName);
HColumnDescriptor cf = descriptor.getFamily("f".getBytes());
// 设置表的存储类型为默认存储，默认存储为热存储
cf.setValue("STORAGE_POLICY", AliHBaseConstants.STORAGETYPE_DEFAULT);
admin.modifyTable(tableName, descriptor);
```

查看冷存储使用情况

在控制台的冷存储界面，可以查看整体的冷存储使用状况，并可以单击**冷存储扩容**进行扩容。

在**集群管理系统**的表Tab中，可以显示某一张表的冷存储使用大小和热存储使用大小。



性能测试

> **说明** 环境说明：
>
> - Master：ecs.c5.xlarge, 4core 8G, 20G高效云盘。
> - 4RegionServer：4RegionServer: ecs.c5.xlarge, 4core 8G, 20G高效云盘。
> - 测试机器：ecs.c5.xlarge, 4core 8G。

写性能

表类型	avg rt	p99 rt
热表	1736 us	4811 us
冷表	1748 us	5243 us

> **说明** 每条记录10列，每列100B，也就是单行1k， 16线程写入。

随机Get性能

表类型	avg rt	p99 rt
热表	1704 us	5923 us
冷表	14738 us	31519 us

说明 关闭表的BlockCache，完全读盘。每条记录10列，每列100B，也就是单行1k。8线程读，每次读出1k。

范围Scan性能

表类型	avg rt	p99 rt
热表	6222 us	20975 us
冷表	51134 us	115967 us

说明 关闭表的BlockCache，每条记录10列，每列100B，也就是单行1k。8线程读，每次读出1k。Scan的Caching设为30。

1.2. 冷热分离

背景

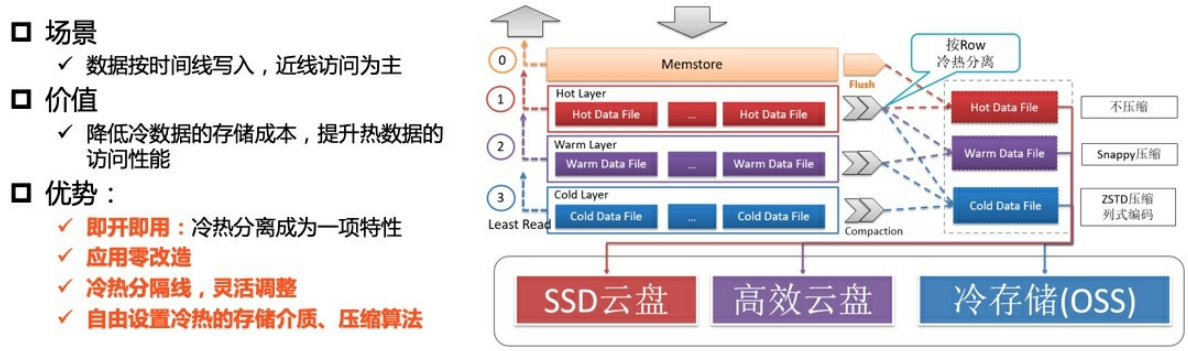
在海量大数据场景下，一张表中的部分业务数据随着时间的推移仅作为归档数据或者访问频率很低，同时这部分历史数据体量非常大，比如订单数据或者监控数据，降低这部分数据的存储成本将会极大的节省企业的成本。如何以极简的运维配置成本就能为企业极大降低存储成本，阿里云HBase增强版冷热分离功能应运而生。阿里云HBase增强版为冷数据提供新的存储介质，新的存储介质存储成本仅为高效云盘的1/3。

HBase增强版在同一张表里实现了数据的冷热分离系统会自动根据用户设置的冷热分界线自动将表中的冷数据归档到冷存储中。在用户的访问方式上和普通表几乎没有任何差异，在查询的过程中，用户只需配置查询Hint或者TimeRange，系统根据条件自动地判断查询应该落在热数据区还是冷数据区。对用户而言始终是一张表，对用户几乎做到完全的透明。

详细介绍请参考云栖社区面向海量数据的极致成本优化-云HBase的一体化冷热分离。

原理介绍

用户在表上配置数据冷热时间分界点，HBase增强版依赖用户写入数据的时间戳(毫秒)和时间分界点来判断数据的冷热。数据最初始在热存储上，随意时间的推移慢慢往冷数据迁移。同时用户可以**任意变更数据的冷热分界点**，数据可以从热到冷，也可以从冷到热。



使用方法

冷存储功能需要HBase增强版服务端升级到2.1.8版本以上，客户端依赖要求AliHBase-Connector 1.0.7/2.0.7以上，Shell要求alihbase-2.0.7-bin.tar.gz以上。

在使用Java API前请参考使用 Java API访问增强版集群中的步骤完成Java SDK安装和参数配置。

在使用HBase shell前，请按照使用HBaseue Shell访问增强版集群中的步骤完成Shell的下载和配置。

开通冷存储功能

请参照使用冷存储开通集群的冷存储功能。

为表设置冷热分界线

用户在使用过程中可以随时调整COLD_BOUNDARY来划分冷热的边界。COLD_BOUNDARY的单位为秒，如COLD_BOUNDARY => 86400 代表 86400秒（一天）前写入的数据会被自动归档到冷存储介质上。

在冷热分离使用过程中，无需把列簇的属性设置为COLD，如果已经把列簇的属性设置为了COLD，请参考使用冷存储将冷存储的属性去除。

Shell

```
// 创建冷热分离表
hbase(main):002:0> create 'chsTable', {NAME=>'f', COLD_BOUNDARY=>'86400'}
// 取消冷热分离
hbase(main):004:0> alter 'chsTable', {NAME=>'f', COLD_BOUNDARY=>""}
// 为已经存在的表设置冷热分离,或者修改冷热分离分界线，单位为秒
hbase(main):005:0> alter 'chsTable', {NAME=>'f', COLD_BOUNDARY=>'86400'}
```

Java API方式

```
// 新建冷热分离表
Admin admin = connection.getAdmin();
TableName tableName = TableName.valueOf("chsTable");
HTableDescriptor descriptor = new HTableDescriptor(tableName);
HColumnDescriptor cf = new HColumnDescriptor("f");
// COLD_BOUNDARY 设置冷热分离时间分界点，单位为秒, 示例表示1天之前的数据归档为冷数据
cf.setValue(AliHBaseConstants.COLD_BOUNDARY, "86400");
descriptor.addFamily(cf);
admin.createTable(descriptor);
// 取消冷热分离
// 注意：需要做major compaction，数据才能从冷存储上回到热存储上
HTableDescriptor descriptor = admin
  .getTableDescriptor(tableName);
HColumnDescriptor cf = descriptor.getFamily("f".getBytes());
// 取消冷热分离
cf.setValue(AliHBaseConstants.COLD_BOUNDARY, null);
admin.modifyTable(tableName, descriptor);
// 为已经存在的表设置冷热分离功能，或者修改冷热分离分界线
HTableDescriptor descriptor = admin
  .getTableDescriptor(tableName);
HColumnDescriptor cf = descriptor.getFamily("f".getBytes());
// COLD_BOUNDARY 设置冷热分离时间分界点，单位为秒, 示例表示1天之前的数据归档为冷数据
cf.setValue(AliHBaseConstants.COLD_BOUNDARY, "86400");
admin.modifyTable(tableName, descriptor);
```

数据写入

冷热分离的表与普通表的数据写入方式完全一致，用户可以参照使用 Java API访问增强版集群文档中的方式或者使用多语言API访问对表进行数据写入。数据的写入的时间戳使用的是 当前时间 。数据先会存储在热存储（云盘）中。随着时间的推移，如果这行数据的写入时间超过 COLD_BOUNDARY 设置的值，就会在major_compact时归档到冷数据，此过程完全对用户透明。

数据查询

由于冷热数据都在同一张表中，用户全程只需要和一张表交互。在查询过程中，如果用户明确知道需要查询的数据在热数据里（写入时间少于COLD_BOUNDARY设置的值），可以在Get或者Scan上设置 HOT_ONLY 的Hint来告诉服务器只查询热区数据。或者在Get/Scan上设置TimeRange来限定查询数据的时间，系统会根据设置TimeRange决定是查询热区，冷区还是冷热都查。查询冷区数据延迟要比热区数据延迟高的多，并且**查询吞吐受到冷存储限制**。

查询示例

Get

Shell

```
// 不带HotOnly Hint的查询，可能会查询到冷数据
hbase(main):013:0> get 'chsTable', 'row1'
// 带HotOnly Hint的查询，只会查热数据部分，如row1是在冷存储中，该查询会没有结果
hbase(main):015:0> get 'chsTable', 'row1', {HOT_ONLY=>true}
// 带TimeRange的查询，系统会根据设置的TimeRange与COLD_BOUNDARY冷热分界线进行比较来决定查询哪个区域的数据（注意TimeRange的单位为毫秒时间戳）
hbase(main):016:0> get 'chsTable', 'row1', {TIMERANGE => [0, 1568203111265]}
```

Java

```
Table table = connection.getTable("chsTable");
// 不带HotOnly Hint的查询，可能会查询到冷数据
Get get = new Get("row1".getBytes());
System.out.println("result: " + table.get(get));
// 带HotOnly Hint的查询，只会查热数据部分，如row1是在冷存储中，该查询会没有结果
get = new Get("row1".getBytes());
get.setAttribute(AliHBaseConstants.HOT_ONLY, Bytes.toBytes(true));
// 带TimeRange的查询，系统会根据设置的TimeRange与COLD_BOUNDARY冷热分界线进行比较来决定查询哪个区域的数据（注意TimeRange的单位为毫秒时间戳）
get = new Get("row1".getBytes());
get.setTimeRange(0, 1568203111265)
```

Scan

如果scan不设置Hot Only，或者TimeRange包含冷区时间，则会并行访问冷数据和热数据来合并结果，这是由于HBase的Scan原理决定的

Shell

```
// 不带HotOnly Hint的查询，一定会查询到冷数据
hbase(main):017:0> scan 'chsTable'， {STARTROW =>'row1', STOPROW=>'row9'}
// 带HotOnly Hint的查询，只会查询热数据部分
hbase(main):018:0> scan 'chsTable'， {STARTROW =>'row1', STOPROW=>'row9'， HOT_ONLY=>true}
// 带TimeRange的查询，系统会根据设置的TimeRange与COLD_BOUNDARY冷热分界线进行比较来决定查询哪个区域的数据（注意TimeRange的单位为毫秒时间戳）
hbase(main):019:0> scan 'chsTable'， {STARTROW =>'row1', STOPROW=>'row9'， TIMERANGE => [0, 1568203111265]}
```

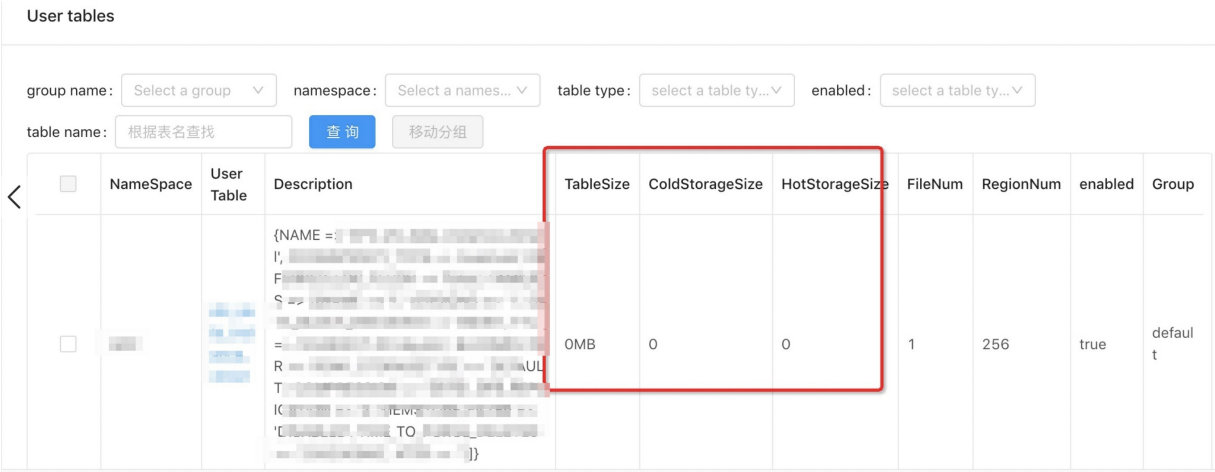
Java

```
TableName tableName = TableName.valueOf("chsTable");
Table table = connection.getTable(tableName);
// 不带HotOnly Hint的查询，一定会查询到冷数据
Scan scan = new Scan();
ResultScanner scanner = table.getScanner(scan);
for (Result result : scanner) {
  System.out.println("scan result:" + result);
}
// 带HotOnly Hint的查询，只会查询热数据部分
scan = new Scan();
scan.setAttribute(AliHBaseConstants.HOT_ONLY, Bytes.toBytes(true));
// 带TimeRange的查询，系统会根据设置的TimeRange与COLD_BOUNDARY冷热分界线进行比较来决定查询哪个区域的数据（注意TimeRange的单位为毫秒时间戳）
scan = new Scan();
scan.setTimeRange(0, 1568203111265);
```

> **说明**
>
> 1. 冷热分离表中的冷区只是用来归档数据，查询请求应该非常的少，用户查询冷热分离表的绝大部分请求应该带上HOT_ONLY的标记（或者设置的TimeRange只在热区）。如果用户有大量请求需要去查冷区数据，则可能得考虑COLD_BOUNDARY冷热分界线的设置是否合理。
> 2. 如果一行数据已经在冷数据区域，但这一行后续有更新，更新的字段先会在热区，如果设置HOT_ONLY去查询这一行（或者设置的TimeRange只在热区），则只会返回这一行更新的字段（在热区）。只有在查询时去掉HOT_ONLY Hint，去掉TimeRange，或保证TimeRange覆盖了该行数据插入和更新时间，才能完整返回这一行。因此不建议对已经进入冷区的数据进行更新，如果有频繁更新冷数据的需求，则可能得考虑COLD_BOUNDARY冷热分界线的设置是否合理。

查看表中冷数据和热数据的大小

在集群管理系统的表Tab中，可以显示某一张表的冷存储使用大小和热存储使用大小。**如果数据还没有进入冷存储，有可能数据还在内存中，请执行flush，将数据刷写到盘上，再请执行major_compact完成后再查看**



进阶功能

优先查询热数据

在范围查询（Scan）场景下，查询的数据可能横跨冷热区，比如查询一个用户的所有订单、聊天记录等。但查询的展示往往是从新到旧的分页展示，最先展示的往往是最近的热数据。在这个场景下，普通的Scan（不带Hot_Only）会并行地扫描冷热数据，导致请求性能下降。而在开启了优先查询热数据后，会优先只查热数据，只有热数据的条数不够显示（如用户点了下一页查看），才会去查询冷数据，减少冷存储的访问，提升请求响应。

开启热数据优先查询，只需在Scan上设置 `COLD_HOT_MERGE` 属性即可。该属性的含义是优先查询热存储中的数据，若热存储中的数据查完了，用户仍然在调用next获取下一条数据，则会开始查询冷数据。

Shell

```
hbase(main):002:0> scan 'chsTable', {COLD_HOT_MERGE=>true}
```

Java

```
scan = new Scan();
scan.setAttribute(AliHBaseConstants.COLD_HOT_MERGE, Bytes.toBytes(true));
scanner = table.getScanner(scan);
```

> 说明
>
> - 若某一行同时包含热数据和冷数据(部分列属于热数据，部分列属于冷数据，比如部分列更新场景)，开启热数据优先功能，会使得该行的查询结果会分两次返回，即scanner返回的Result集合中，对于同一个Rowkey会有两个对应的Result。
> - 由于是先返回热数据，再返回冷数据，开启热数据优先功能后，无法保证后返回的冷数据结果的Rowkey一定大于先返回的热数据结果的Rowkey，即Scan得到的Result集不保序， 但热数据和冷数据的各自返回集仍保证按Rowkey有序(参见下面的demo)。在部分实际场景中， 用户可以通过Rowkey设计，保障scan的结果仍然保序，比如订单记录表，Rowkey=用户ID+订单创建时间，扫描某个用户的订单数据是有序的。

```
//假设rowkey为"coldRow"的这一行是冷数据，rowkey为"hotRow"的这一行为热数据
//正常情况下，由于hbase的row是字典序排列，rowkey为"coldRow"的这一行会比"hotRow"这一行先返回。
hbase(main):001:0> scan 'chsTable'
ROW                                COLUMN+CELL
 coldRow                               column=f:value, timestamp=1560578400000, value=cold_value
 hotRow                                column=f:value, timestamp=1565848800000, value=hot_value
2 row(s)
// 设置COLD_HOT_MERGE时, scan的rowkey顺序被破坏，热数据比冷数据先返回，因此返回的结果中，"hot"排在了"cold"的前面
hbase(main):002:0> scan 'chsTable', {COLD_HOT_MERGE=>true}
ROW                                COLUMN+CELL
 hotRow                                column=f:value, timestamp=1565848800000, value=hot_value
 coldRow                               column=f:value, timestamp=1560578400000, value=cold_value
2 row(s)
```

根据Rowkey中的字段划分数据的冷热

除了通过写入KV的时间戳区分冷热，HBase增强版还支持通过Rowkey中的某些字段来区分数据的冷热，如用户的Rowkey中包含一个timestamp，增强版支持parse这个timestamp来划分这行数据的冷热，而不是根据KevValue的写入时间戳。如果根据KV写入时间戳划分冷热无法满足用户需求，欢迎提工单或者钉钉上找 `云HBase答疑` 来询问使用方法。

注意事项

参见使用冷存储中的注意事项。

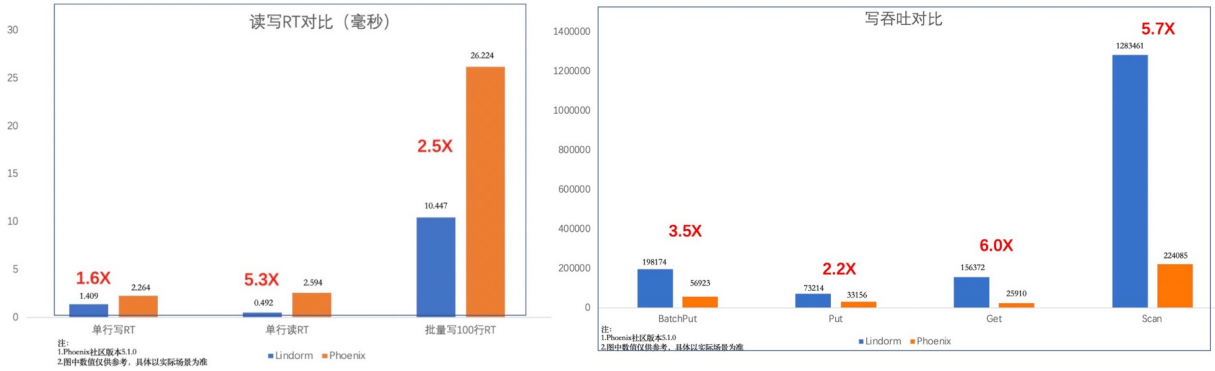
1.3. 高性能原生二级索引

二级索引简介

HBase原生提供了主键索引，即按rowkey的二进制排序的索引。Scan可基于此rowkey索引高效的执行整行匹配、前缀匹配、范围查询等操作。但若需要使用rowkey之外的列进行查询，则只能使用filter在指定的rowkey范围内进行逐行过滤。若无法指定rowkey范围，则需进行全表扫描，不仅浪费大量资源，查询RT也无法保证。

有多种解决方案可解决HBase的多维查询的问题。比如以要查询的列再单独写一张表(用户自己维护二级索引)，或者将数据导出到Solr或者ES这样的外部系统进行索引。像Solr/ES这样的搜索引擎类产品，提供了强大的ad hoc查询能力，云HBase现已集成了全文索引服务。

Solr/ES固然强大，但对于大部分列较少且有固定查询模式的场景来说，有”杀鸡用牛刀”之感。为此，HBase增强版推出了原生的全局二级索引解决方案，以更低的成本解决此类问题。因内置于HBase，提供了强大的吞吐与性能。这个索引方案在阿里内部使用多年，经历了多次双11考验，尤其适合解决海量数据的全局索引场景。下图给出了HBase增强版与Phoenix在索引场景下的性能对比：



下面，我们先介绍索引的两个重要概念，然后介绍HBase增强版二级索引的DDL和DML操作，讨论一些高级主题，如rowkey的二进制排序问题以及查询优化的问题，最后，给出使用约束和FAQ。

基本概念

考虑如下主表和索引表：

```
create table 'dt' (rowkey varchar, c1 varchar, c2 varchar, c3 varchar, c4 varchar, c5 varchar constraint pk pri
mary key(rowkey));
create index 'idx1' on 'dt' (c1);
create index 'idx2' on 'dt' (c2, c3, c4);
create index 'idx3' on 'dt' (c3) include (c1, c2, c4);
create index 'idx4' on 'dt' (c5) include (ALL);
```

索引列

索引表的主键列就是其索引列。比如idx1的c1，idx2的c2,c3,c4。索引列及其顺序决定了索引表能支持的查询场景。只有一个索引列的索引表称单列索引，有多个索引列的称组合索引。

冗余列

如果查询中所需要的列在索引表里没有，则需要回查主表才能完成查询。在分布式场景下，回查主表可能带来额外的多次RPC，导致查询RT大幅度增加。因此，通过空间换时间的策略，将主表中的列冗余在索引表中，来避免命中索引的查询再回查主表。有冗余列的索引叫冗余索引(如idx3和idx4)，或覆盖索引(Covered Index)。

考虑到业务变化，可能会增加新列，因此，HBase增强版提供了’冗余所有列’的语义(即idx4)。索引表会自动冗余主表的所有列，从容应对业务变化。

使用前准备

- 服务器版本要求：2.1.10及以上，如果在此版本之下的集群，请在控制台上点击小版本升级
- 客户端版本要求： 需要alihbase-client 1.1.9/2.0.4以上，或者alihbase-connector1.0.9/2.09以上，详见HBase Java SDK 下载。

- Shell版本要求： 需要 alihbase-2.0.9-bin.tar.gz以上版本，请参考使用HBaseue Shell访问下载最新版增强版Shell工具。

管理索引(DDL)

可通过HBase Shell和Java API进行索引DDL操作。本节介绍如何使用HBase shell来进行索引DDL操作，关于Java API的使用，可以参见AliHBaseUEAdmin的相关接口注释。

下面的shell命令展示了几个常用的DDL操作：

```
# 创建索引
# 为主表dt创建索引idx1: create index idx1 on dt ('f1:c2', 'f1:c3');
# 冗余主表中的所有列，注意 COVERED_ALL_COLUMNS 是一个关键字，请不要在列名中使用它
hbase(main):002:0> create_index 'idx1', 'dt', {INDEXED_COLUMNS => ['f1:c2', 'f1:c3']}, {COVERED_COLUMNS =
> ['COVERED_ALL_COLUMNS']}
# 查看索引schema
hbase(main):002:0>describe_index 'dt'
# 禁用dt的idx1索引，此时，更新dt不会再更新idx1
hbase(main):002:0>offline_index 'idx1', 'dt'
# 删除idx1这个表
hbase(main):002:0>remove_index 'idx1', 'dt'
```

下面进行详细介绍。

创建索引

```
hbase(main):002:0>create_index 'idx1', 'dt', {INDEXED_COLUMNS => ['f1:c2', 'f2:c3']}
```

为主表dt创建索引idx1，索引列有2个，f1列族下的c2列，f2列族下的c3列。没有冗余列。

```
hbase(main):002:0>create_index 'idx2', 'dt', {INDEXED_COLUMNS => ['f1:c1']}, {COVERED_COLUMNS => ['f2:c2'
]}
```

为主表dt创建索引idx1，索引列有2个，f1列族下的c1列，冗余f2列族下的c2列。

```
hbase(main):002:0>create_index 'idx3', 'dt', {INDEXED_COLUMNS => ['f1:c3']}, {COVERED_COLUMNS => ['COVE
RED_ALL_COLUMNS']}
```

idx3会冗余dt的**所有列**。因此，命中idx3的查询一定不会回查主表。注意 `COVERED_ALL_COLUMNS` 是一个关键字，表示此索引表会冗余主表的所有列。因此，不要使用这个名字作为列名。

除设定索引表的schema(索引列/冗余列)之外，也支持设置索引表的存储特性，例如：

```
hbase(main):002:0>create_index 'idx1', 'dt', {INDEXED_COLUMNS => ['f1:c1', 'f2:c2']}, {DATA_BLOCK_ENCODI
NG => 'DIFF', BLOOMFILTER => 'ROW',COMPRESSION => 'LZO'}
```

查看索引schema

通过list命令可以查看主表及其所有的索引表，例如：

```
hbase(main):023:0> list
TABLE
dt
dt.idx1
dt.idx2
yh1
4 row(s)
Took 0.0129 seconds
=> ["dt", "dt.idx1", "dt.idx2", "yh1"]
```

通过describe_index命令可以查看指定主表的所有索引表的schema信息，例如：

```
hbase(main):024:0> describe_index 'dt'
Index [idx2] on [dt] (f1.c3 ASC,f2.c4 ASC)
Index [idx1] on [dt] (f1.c1 ASC,f2.c2 ASC) includes (@@ALL@@)
Total 2 indexes found for table dt
```

删除索引

与普通的HBase表一样，删除索引也需要先禁用(offline_index)，再删除(remove_index)，例如：

```
# 禁用dt的idx1索引，此时，更新dt不会再更新idx1
hbase(main):002:0>offline_index 'idx1', 'dt'
# 删除idx1这个表
hbase(main):002:0>remove_index 'idx1', 'dt'
```

注意，**不能使用disable命令来禁用索引。如果索引表offline，所有的查询都不会走这张索引表**

为有数据的表建索引时，历史数据同步问题

在为一张已经有数据的表添加新的索引时，create_index命令会**同时**将主表的历史数据同步到索引表中。因此，当主表很大时，create_index会非常耗时。注意：create_index的数据同步任务是在服务端执行的，即使杀掉hbase shell进程，数据同步任务也会继续执行下去，直到任务完成。

未来我们会开放异步构建索引的能力，即create_index时不同步历史数据，而是用户显式执行一个命令来触发后台的数据同步流程，并通过检查索引状态是否为active来判断数据同步是否完成。

访问索引(DML)

本节介绍通过java API来访问索引(DML)。HBase Java API的基础使用，连接创建请参见使用 Java API访问增强版集群。

数据写入

用户不需要主动向索引表写入数据。写主表时，HBase会自动将变更同步到其所有的索引表中。HBase增强版提供**同步更新**语义，即：写主表时会**同步**更新其**所有**索引表，待主表和索引表都写成功后，写操作才会返回到客户端。可以从以下两个角度来理解：

- 强一致：写主表成功后，本次更新可以立即被读到。
- 写进行中或超时：主表和索引表的一致性未决，但保证最终一致，即要么都更新，要么都不更新。

数据查询

与关系型数据库类似，用户不需要直接发起针对索引表的查询，只需按业务要求表达对主表的查询。HBase增强版会根据索引表的schema和查询模式自动选择最合适的索引表进行查询。通过Filter来描述基于非rowkey的查询条件，例如：

```
byte[] f = Bytes.toBytes("f");
byte[] c1 = Bytes.toBytes("c1");
byte[] value = Bytes.toBytes("yourExpectedValue");
// 等价于 select * from dt where f.c1 == value
Scan scan = new Scan();
SingleColumnValueFilter filter = new SingleColumnValueFilter(f, c1, EQUAL, value);
scan.setFilter(filter);
```

注意：

- **如果使用LESS，GREATER等条件，需要注意数据的排序问题，详情请见进阶功能中的符号数的排序问题一节**
- 如果查询可以命中索引，系统会自动将 `setFilterIfMissing` 设置为 `true` 。如果查询无法命中索引，则会使用scan里配置的值

通过使用FilterList，可以实现and和or条件的组合与嵌套，以表达更复杂的查询条件，例如：

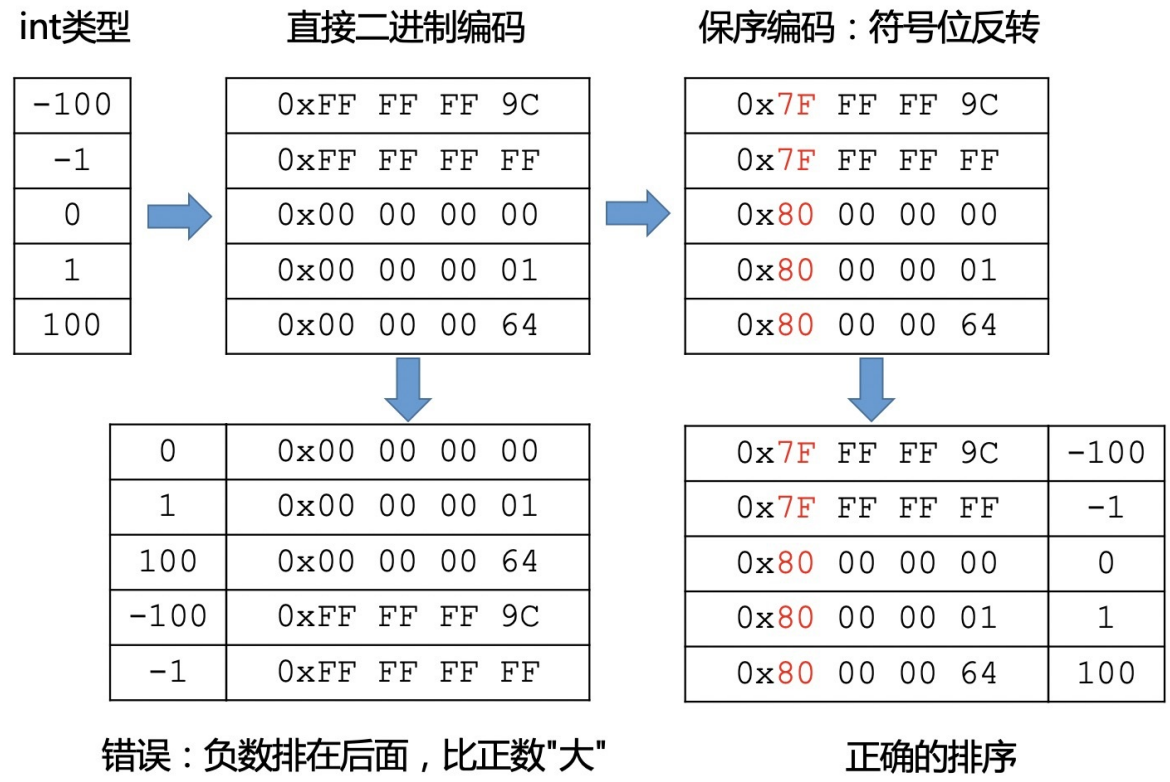
```
// 等价于 where f.c1 >= value1 and f.c1 < value2
FilterList filters = new FilterList(FilterList.Operator.MUST_PASS_ALL);
filters.addFilter(new SingleColumnValueFilter(f, c1, GREATER_OR_EQUAL, value1));
filters.addFilter(new SingleColumnValueFilter(f, c1, LESS, value2));
```

HBase增强版会根据Filter以及索引的schema自动选择合适的索引表进行查询。

进阶功能

有符号数的排序问题

HBase原生API只有一种数据类型 `byte[]` ，并以此进行二进制排序。因此，所有业务上使用的类型都需要转换为 `byte[]` 。这就涉及到数据的原始排序与转换成 `byte[]` 之后的排序问题。我们希望，在转换前后数据的排序保持不变。HBase client的Bytes类提供了各种类型与 `byte[]` 之间的相互转换，但这些接口仅适用于0和正数，而不适用于负数。下图以 `int` 类型为例描述了这个问题：



上图中，`直接二进制编码` 就是 `Bytes.toBytes(int)` 。可见，在有负数参与的情况下，转换为 `byte[]` 之后的 `int` 不再保序。该问题可通过符号位反转来解决。对于 `byte` 、 `short` 、 `long` 、 `float` 等类型，都有此类问题。因此，HBase增强版提供了新的工具类 `org.apache.hadoop.hbase.util.OrderedBytes` (依赖了alihbase-client或者alihbase-connector都可以找到该类)来解决这个问题。下面举例说明其用法：

```
// int转换为保序的byte[]
int x = 5;
byte[] bytes = OrderedBytes.toBytes(x);
// byte[]转换为int
int y = OrderedBytes.toInt(bytes);
```

更多用法可参见类注释。

查询优化

本节讨论HBase增强版如何根据查询进行索引选择。概括的说，就是 `前缀匹配` 。这是一种RBO(Rule Based Optimization)策略。根据查询条件中以and连接的等值比较的条件与匹配索引表的前缀，选择匹配程度最高的索引表作为本次查询使用的索引。下面我们通过一些示例来理解这一规则。

假设有如下主表和索引表：

```
create table 'dt' (rowkey varchar, c1 varchar, c2 varchar, c3 varchar, c4 varchar, c5 varchar constraint pk pri
mary key(rowkey));
create index 'idx1' on 'dt' (c1);
create index 'idx2' on 'dt' (c2, c3, c4);
create index 'idx3' on 'dt' (c3) include (c1, c2, c4);
create index 'idx4' on 'dt' (c5) include (ALL);
```

考虑如下查询：

```
select rowkey from dt where c1 = 'a';
select rowkey from dt where c2 = 'b' and c4 = 'd';
select * from dt where c2 = 'b' and c3 >= 'c' and c3 < 'f';
select * from dt where c5 = 'c';
```

说明：

（1） `select rowkey from dt where c1 = 'a'`

命中索引表 `idx1`

（2） `select rowkey from dt where c2 = 'b' and c4 = 'd'`

命中索引表 `idx2` ，从中查找所有满足 `c2 = 'b'` 条件的行，然后逐行按 `c4 = 'd'` 进行过滤。虽然C4是索引列之一，但因where条件中缺少C3列，无法匹配上idx2的前缀。

（3） `select * from dt where c2 = 'b' and c3 >= 'c' and c3 < 'f'`

命中索引表 `idx2` ，完美匹配。但因为是 `select *` ，而索引表里并未包含主表的所有列，因此在查询索引之后，还要回查一次主表。回查主表时，回查的rowkey可能散布在主表的各个地方，因此，可能会消耗多次RPC。回查的数据量越大，RT越长。

（4） `select * from dt where c5 = 'c'`

命中索引表 `idx4` ，完美匹配。因为idx3是全冗余索引，所以， `select *` 不需要回查主表。

因此，用户需要结合实际查询模式来进行索引表的设计，并考虑好未来一段时间中潜在的业务变化。限于篇幅，这里不对此进行详细展开讨论。有兴趣的读者可以阅读《数据库索引设计与优化》，以更进一步的了解如何用好索引。

约束与限制

- 不同主表可以有同名索引，如dt有索引idx1，foo也可以有索引idx1；但同一主表下不允许有同名索引。
- 只能为version = 1的表建索引，不支持为多版本的表建索引。
- 对有TTL的主表建索引，不能单独为索引表设置TTL：索引表会自动继承主表的TTL。
- 索引列最多不超过3个。
- 索引列 + 主表rowkey，总长度不能超过30KB；不建议使用大于100字节的列作为索引列。
- 单个主表的索引表个数最多不超过5个。
- 一次查询最多只能命中一个索引，不支持多索引联合查询(Index Merge Query)。
- 创建索引时会将主表的数据同步到索引中，对大表建索引会导致create_index命令耗时过长。异步创建索引的功能会在未来开放。
- scan的filter中，仅支持 `BinaryComparator` ，若使用其他comparator，则查询会退化为主表的查询(不会抛错)。

下列功能因为使用上有一些限制，所以，暂时未开放。

- 异步创建索引：只建索引，不同步历史数据，通过显式执行一个命令来为历史数据构建索引。
- 自定义时间戳写入数据。

对于索引使用上的任何问题，欢迎钉钉联系 `云HBase答疑` 或者工单咨询。

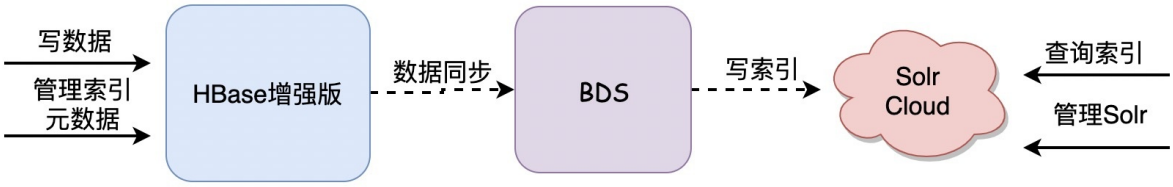
1.4. 全文索引服务

全文索引Search服务用来解决复杂的多维查询和全文检索。

Solr是构建在Apache Lucene上的企业级搜索平台，是分布式全文检索的最佳实践之一，支持各种复杂的条件查询和全文检索，具有广泛的用户基础。通过深度融合HBase与Solr，我们推出了既能满足大数据海量存储，又可以支持复杂多维查询和全文检索的Search服务。

Search服务适用于：**需要保存海量数据，并且需要各种条件组合查询的业务。**例如：

- 物流场景，需要存储大量轨迹物流信息，并需根据任意多个字段组合查询。
- 交通监控场景，保存大量过车记录，同时会根据车辆信息任意条件组合检索出感兴趣的记录。
- 网站会员、商品信息检索场景，一般保存大量的商品/会员信息，并需要根据少量条件进行复杂且任意的查询，以满足网站用户任意搜索需求等。



Search服务的整体数据流如上图，数据写入HBase后，BDS负责将数据实时同步到Solr中。在此架构下，HBase服务、数据同步通道BDS和Solr都是以独立集群的方式存在，您可以分别对各个集群进行管理：如果Solr处理能力不足，只需要扩容Solr集群；如果BDS同步能力不足，可以单独扩容BDS。HBase/BDS/Solr可以针对不同的使用场景选择不同的机型，独立的部署形态大幅提升了系统的稳定性。

与二级索引的区别

HBase增强版提供高性能原生二级索引，可以低成本的解决非主键查询问题，适用于查询列比较固定的场景。如果业务场景需要复杂的多维组合查询，需要考虑使用Search服务。

与开源Solr的区别

Search服务深度融合HBase和Solr，用户无需关注各个服务的运行，只需要通过简单的API/Shell操作就可以将HBase与Solr建立关联。

Search服务基于开源Solr深度定制，完全兼容开源Solr API，在系统稳定性、读写性能、监控告警上做了大量工作，提供更加可靠、高性能的企业级搜索平台。

服务开通

开通Search服务需要三步：

1. 创建增强版HBase集群；
2. 创建BDS集群；
3. HBase集群创建成功后，在HBase控制台页面单击 `全文索引` ，完成Search实例的购买和关联。

具体参见开通指南。

使用指南

参见快速开始和索引管理。

最佳实践

参见最佳实践。

1.5. 云盘加密

云数据库HBase免费提供云盘加密功能，基于块存储对整个数据盘进行加密，即使数据备份泄露也无法被解密，保护您的数据安全。

功能说明

云盘适用于数据安全或法规合规等场景，加密保护云数据库HBase上的数据。您无需自建与维护密钥管理基础设施，即可保障数据的隐私性和自主性，为业务数据提供安全边界。

在创建云数据库HBase增强版（Lindorm）实例时，如果开通了**云盘加密**功能，系统将对以下数据进行加密：

- 云盘中的静态数据。
- 云盘和实例间传输的数据。
- 从加密云盘创建的所有快照（即加密快照）。

费用说明

云数据库HBase的**云盘加密**功能是通过密钥服务系统（KMS）所提供的服务密钥对云盘进行加密来实现的。

> **说明** 云数据库HBase不会产生额外的费用，使用过程中会产生KMS服务密钥管理和调用的费用，费用信息，请参见计费说明。

功能优势

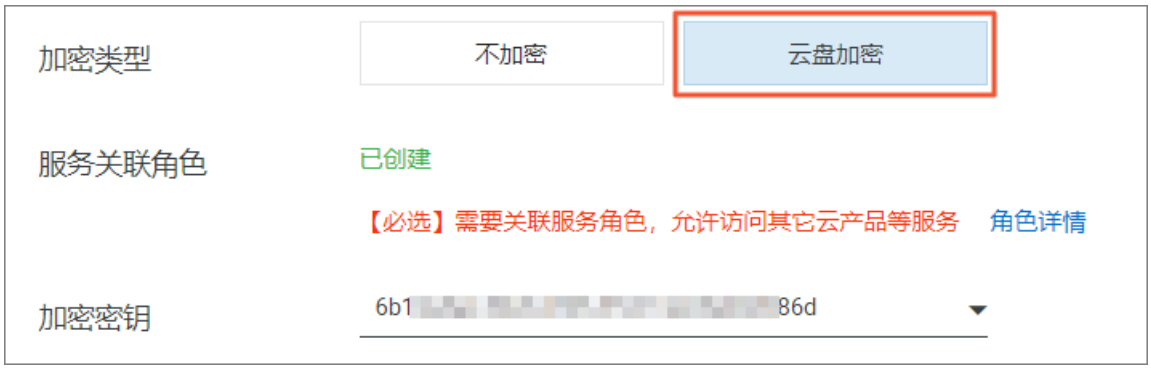
- 云盘加密不会影响业务，您无需修改应用程序。
- 云盘加密功能不会造成明显的性能损失。

注意事项

- 仅在新建HBase实例时可以开启云盘加密，创建实例后无法开启。
- 云盘加密功能开启后无法关闭。
- 开启云盘加密后，实例生成的快照以及通过这些快照创建的云盘版实例将自动延续加密属性。
- 目前HBase增强版（Lindorm）的国内站和国际站开放云盘加密能力。

开启方式

1. 登录云数据库HBase控制台。
2. 在**集群列表**页面，单击**创建HBase集群**。
3. 设置如下参数。



参数	说明
服务	选择**增强版（Lindorm）**。
Core磁盘类型	选择**ESSD云盘**、**SSD云盘**或**高效云盘**。
加密类型	选择**云盘加密**。
服务关联角色	使用云盘加密功能必须要开启服务关联角色，如果显示**已创建**则表示已开启服务关联角色，如果未开启请单击**创建服务关联角色**进行开启。 关于服务关联角色的信息，请参见HBase服务关联角色。
加密密钥	选择密钥，如果当前地域没有KMS密钥，可以在KMS控制台创建。 **说明** ○ HBase的云盘加密功能仅支持使用您手动创建的服务密钥，在创建普通密钥时需要将**轮转周期**设置为**不开启**。创建密钥详细操作，请参见创建密钥。 ○ 授权开通KMS会生成相关审计记录。更多信息，请参见使用操作审计查询密钥管理服务的操作事件。

说明 更多创建HBase集群的参数和说明，请参见购买集群。

4. 单击**立即购买**，即可创建一个加密云盘的HBase示例。
5. 集群创建成功后，您可以在实例**基本信息**页面的**Core节点信息**中查看**加密类型**和**加密密钥**。

2.Lindorm Insight集群管理

2.1. Lindorm Insight系统介绍

2.1.1. 系统概述

本文档主要对Lindorm Insight系统进行一个简单的概述说明。

Lindorm Insight系统为您提供了一个全新的集群管理系统，通过这个系统可以看到集群各个可用区zone的容量状况、Server列表、表属性、表大小等信息。在Lindorm Insight系统中，您还可以完成Namespace管理、表管理、用户管理、ACL管理等功能。同时Lindorm Insight提供集群监控的能力，您能通过Lindorm Insight查看集群、分组、节点以及Namespace、表等多个层面的实时监控指标。Lindorm Insight系统同时还支持集群例行健康巡检，并提供诊断相关工具和功能，能帮助您快速分析定位问题。

2.1.2. 如何访问Lindorm Insight?

本文主要介绍了访问Lindorm Insight系统的操作步骤。

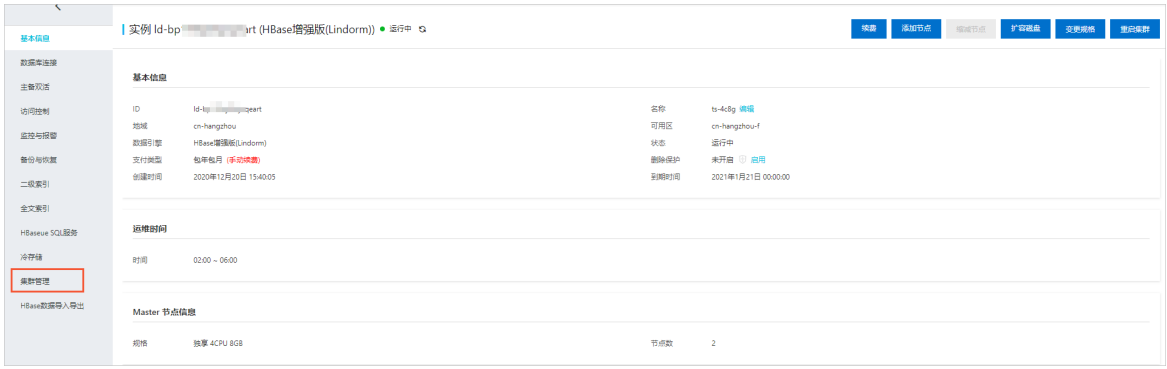
前提条件

- 已将访问Lindorm Insight系统的电脑公网IP加入到白名单中。具体请参考设置白名单和安全组。
- 已重置了UI访问密码。具体请参考WebUI页面访问。

操作步骤

1. 登录HBase管理控制台，单击集群ID链接。

2. 在集群实例详情页面，单击左侧导航栏的**集群管理**。



3. 在**集群管理系统**页面，单击**ClusterManager**即可进入Lindorm Insight系统。

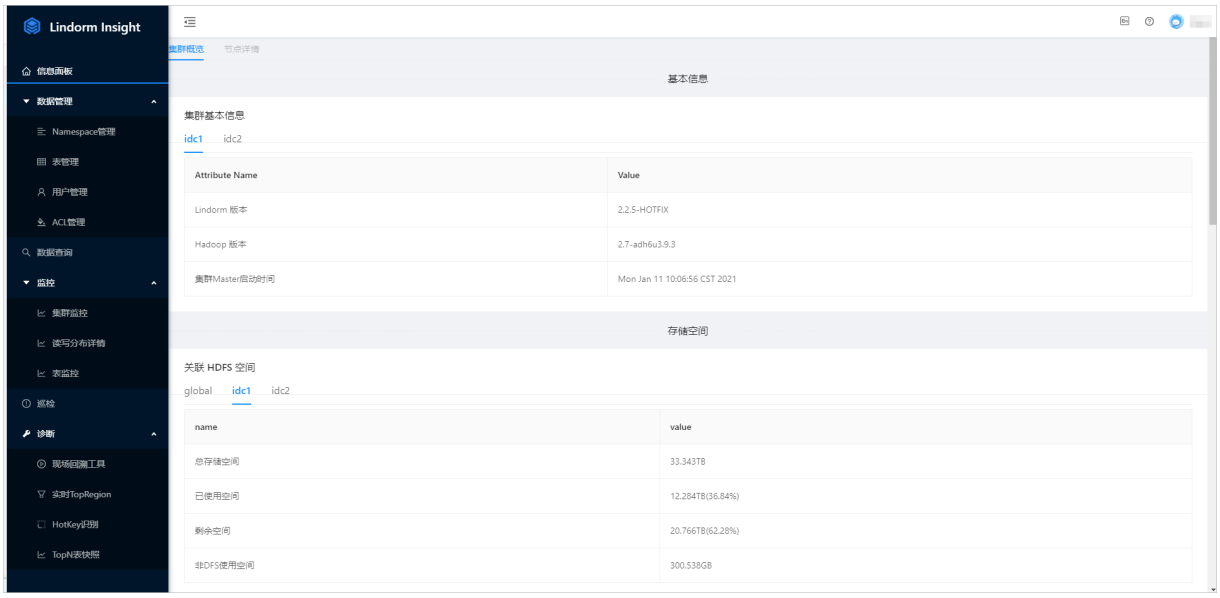
2.1.3. 面板说明

2.1.3.1. 集群概览

Lindorm Insight集群概览页面可以查看集群的基本信息、存储空间、分组信息。

访问集群概览页面

您可以在HBase管理控制台，进入HBase增强版(Lindorm)实例详情页面，单击左侧导航栏的**集群管理**。在**集群管理系统**页面，单击**ClusterManager**即可进入Lindorm Insight信息面板概览页面。



集群概览页面分为以下三个区域，您可以单击区域链接查看对应的内容。

- 集群基本信息
- 集群存储空间
- 集群分组信息

集群基本信息

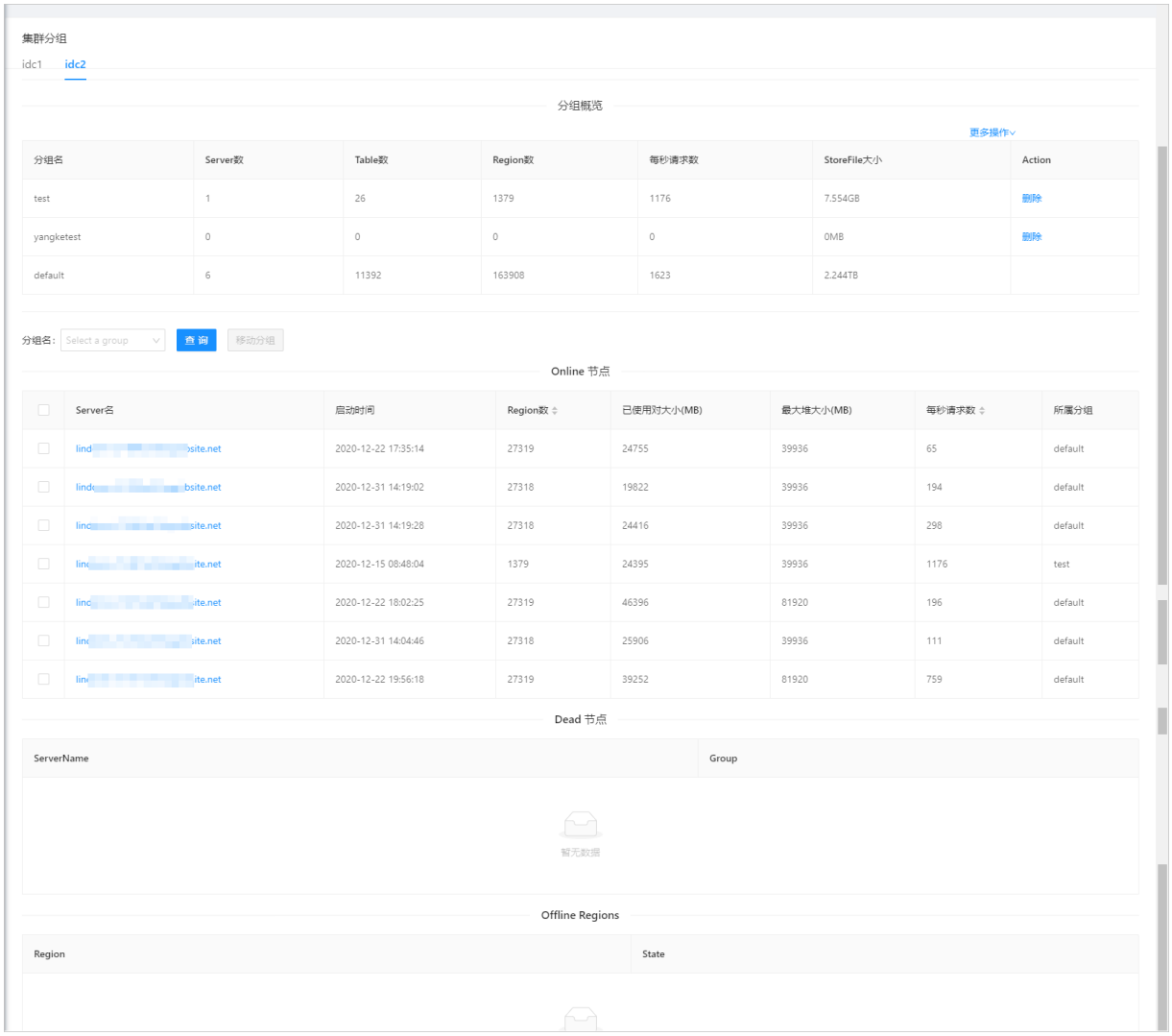
基本信息区域展示了集群的Lindorm版本、Hadoop版本信息和集群Master启动时间。

集群存储空间

存储空间区域展示了集群当前的存储状态，包括集群存储总空间、已使用空间、剩余存储空间、非DFS使用存储空间。

集群分组信息

分组信息区域展示了集群当前的分组概览信息、Online节点、Dead节点、Offline Regions信息。



2.1.3.2. 节点详情

Lindorm Insight节点详情页面可以查看集群分组下节点的基本信息以及当前节点的Region详情。

访问节点详情页面

在**集群概览**页面，单击Server名链接即可跳转到节点详情页面。

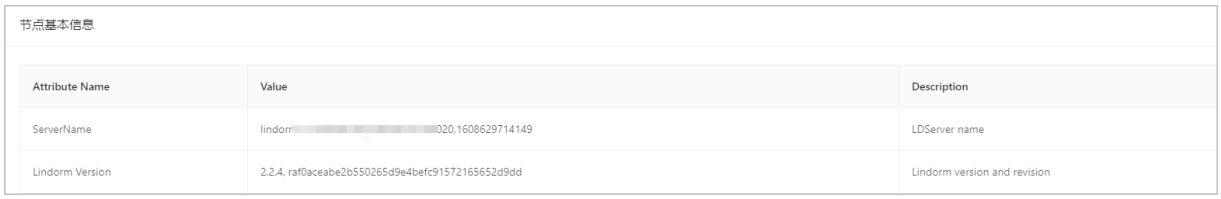
节点详情页面分为以下三个区域，您可以单击区域链接查看对应的内容。

- 节点基本信息

- Online Regions
- Regions in recovering

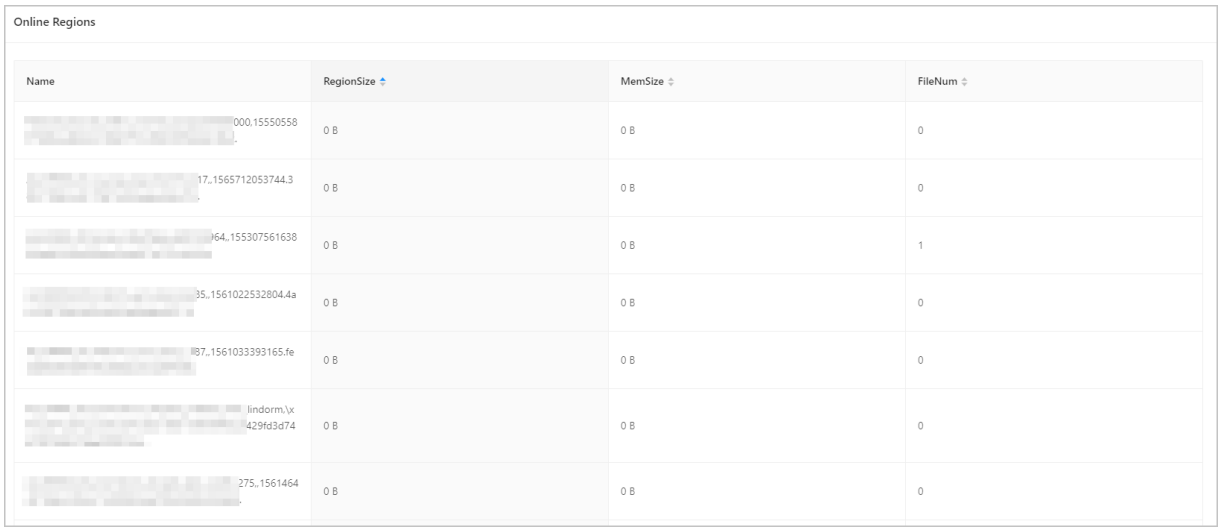
节点基本信息

节点基本信息区域展示了节点的ServerName、Lindorm Version信息。



Online Regions

Online Regions区域展示了节点的在线Region信息，包括RegionSize（表分片的大小）、MemSize（表分片在内存中的数据大小）、FileNum（表分片下的LDFile数量）信息。



Regions in recovering

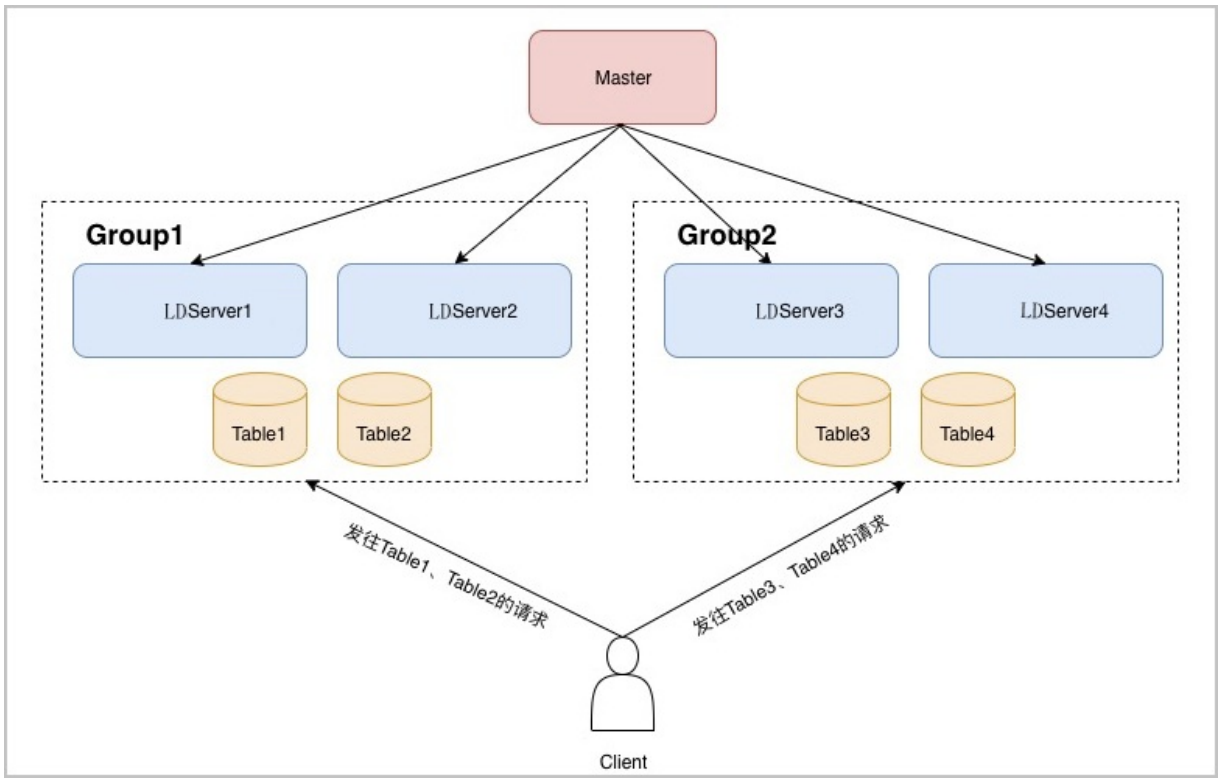
Regions in recovering区域展示了节点的recovering Region信息。

2.2. 配置和管理

2.2.1. 分组管理

当多个用户或者业务在使用同一个Lindorm集群时，往往会存在资源争抢的问题。一些重要的在线业务，可能会被离线业务的批量读写所影响。分组功能由阿里云Lindorm集群提供，用来解决多租户的隔离问题。通过把LDServer划分到不同的Group分组，每个分组上存储不同的表，从而达到资源隔离的目的。

分组管理的架构图如下所示：



上图中，我们创建了一个Group1，把LDServer1和LDServer2划分到Group1中；创建了一个Group2，把LDServer3和LDServer4划分到Group2；同时，我们把Table1和Table2也移动到LDServer1分组。在这种情况下，Table1和Table2的所有region，都只会分配到Group1中的LDServer1和LDServer2这两台机器上。

同理，属于Group2的Table3和Table4的Region在分配和balance过程中，也只会落在LDServer3和LDServer4上。因此，用户在请求这些表时，发往Table1、Table2的请求，只会由LDServer1和LDServer2服务；而发往Table3和Table4的请求，只会由LDServer3和LDServer4服务，从而达到资源隔离的目的。

查看Group信息

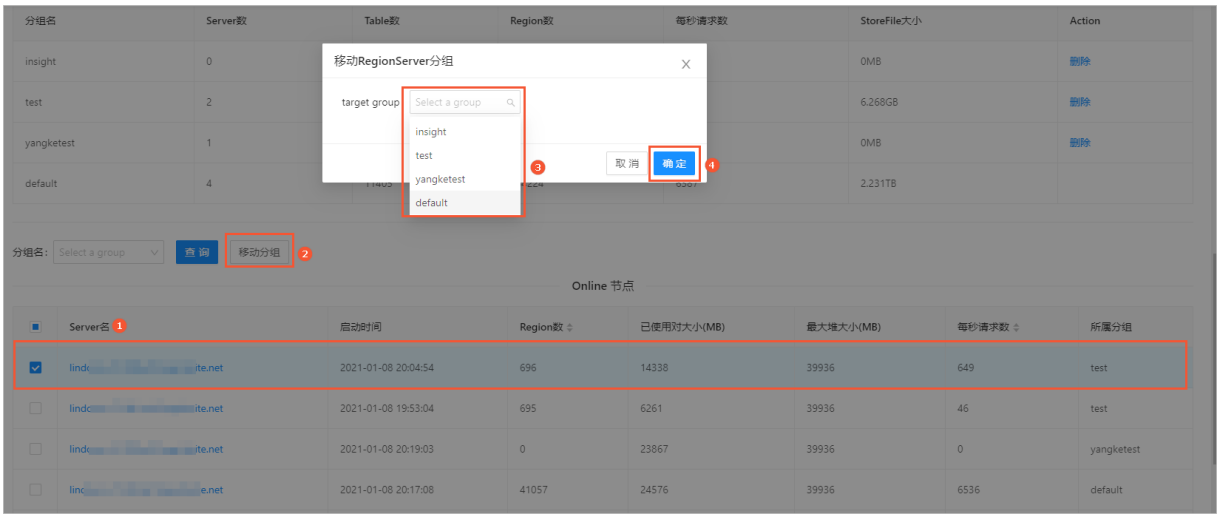
在Lindorm Insight**集群概览**页面的**分组信息**区域，可以看到当前集群所有的Group信息。如果您没有创建过Group，系统会有一个默认的**default** group，所有的LDServer和表都会归属到这个Group中。

创建Group

在Lindorm Insight**集群概览**页面的**分组概览**区域，单击**更多操作 > 新建分组**来创建新的Group。创建新的Group后，这个Group内的Server数量和表数量都为0，后续需要您将Server和表移动至这个Group中。

移动Group

在Lindorm Insight**集群概览**页面的**Online 节点**区域，选择需要移动的Server，单击**移动分组**。然后在target group下拉选择目标分组，单击**确认**。



默认状态下，所有的LDServer都属于**default**分组，您需要将LDServer移动到对应的Group中才能进行使用。

> **说明**
>
> - 如果您将一张表移动到一个没有任何LDServer的Group，表的Region会因为没有任何服务器可以上线从而无法访问。
> - 每一个分组下至少需要拥有两台LDServer，这样当一台LDServer宕机后，表的Region可以迁移到同一分组下的另外一台LDServer上。如果分组下只有一台LDServer，当这台LDServer宕机后，这个分组所有的表将无法访问。
> - 在移动LDServer的分组时，这个LDServer上打开的Region会被立刻重新平均分配到分组的其他LDServer上去。

删除Group

在Lindorm Insight**集群概览**页面的**分组概览**区域，单击**Action**列下面的**删除**，删除对应的Group。

> **说明** 只有当该Group中的所有表和Server都被移出的时候，Group才能被删除。

2.2.2. 数据管理

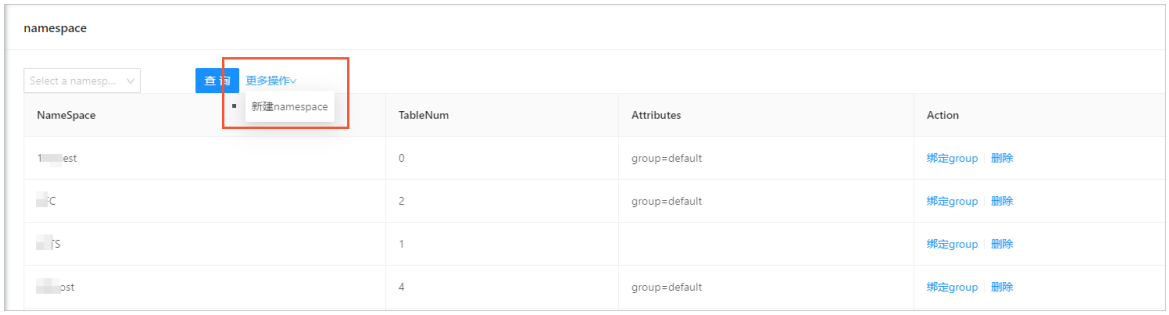
2.2.2.1. Namespace管理

Lindorm中的Namespace相当于表的一个逻辑分组，不同的Namespace下面的表允许重名。不同的用户拥有不同的Namespace，彼此隔离，即不同用户的Namespace将拥有不同的表集合。Lindorm会提供一个默认的Namespace，如果用户没有指定Namespace，系统会默认使用**default** Namespace。

您可以通过Java API和Lindorm的Shell来创建和删除Namespace，也可以使用Lindorm Insight系统来管理创建和删除Namespace。本文档主要介绍了如何使用Lindorm Insight系统来管理创建和删除Namespace。

创建Namespace

1. 访问Lindorm Insight。具体操作请参考如何访问Lindorm Insight？。
2. 在左侧导航栏单击**数据管理 > Namespace管理**。
3. 在**namespace** 页面，单击**更多操作 > 新建namespace**。

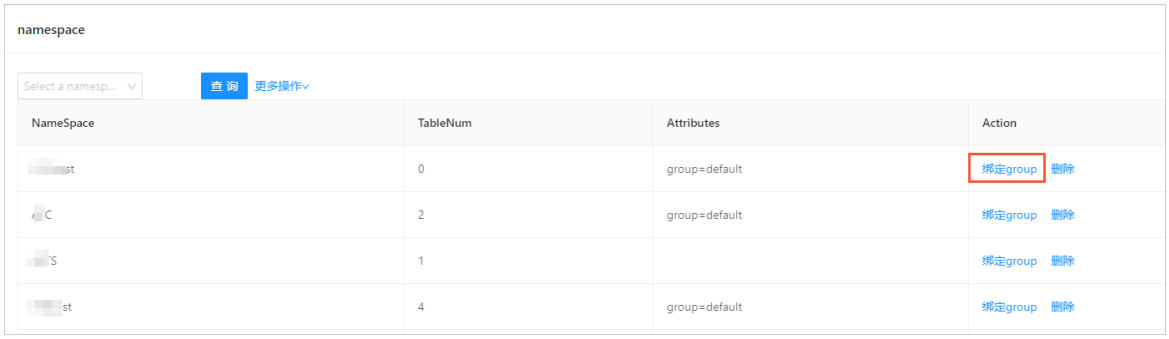


4. 在**Create namespace**对话框中，输入**Namespace name**，单击**确定**。

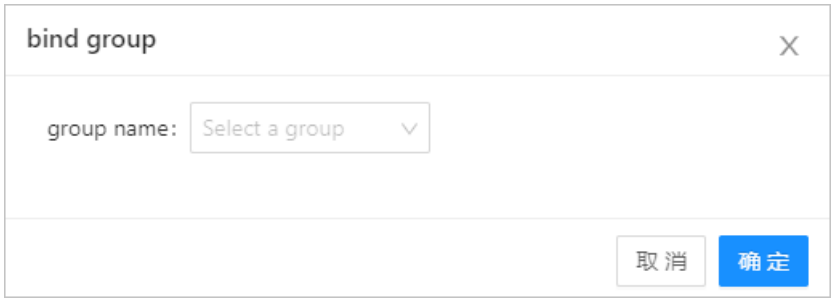
绑定Namespace所属Group

绑定Namespace所属Group可以将该Namespace下所建立的新表自动关联到对应的Group中。

1. 在**namespace** 页面的namespace列表中，定位到需要绑定Group的Namespace，单击**绑定Group**。



2. 在**bind group**对话框中，输入**group name**，单击**确定**。



删除Namespace

在**namespace** 页面的namespace列表中，定位到需要删除的Namespace，单击**删除**。

> **说明** 删除Namespace前，请先将Namespace下所有的表都删除，否则删除Namespace会失败。

2.2.2.2. 表管理

在表管理页面中，您可以看到集群中所有的表以及表属性。单击表名后，您可以看到表的详细信息和所有的Region信息等。

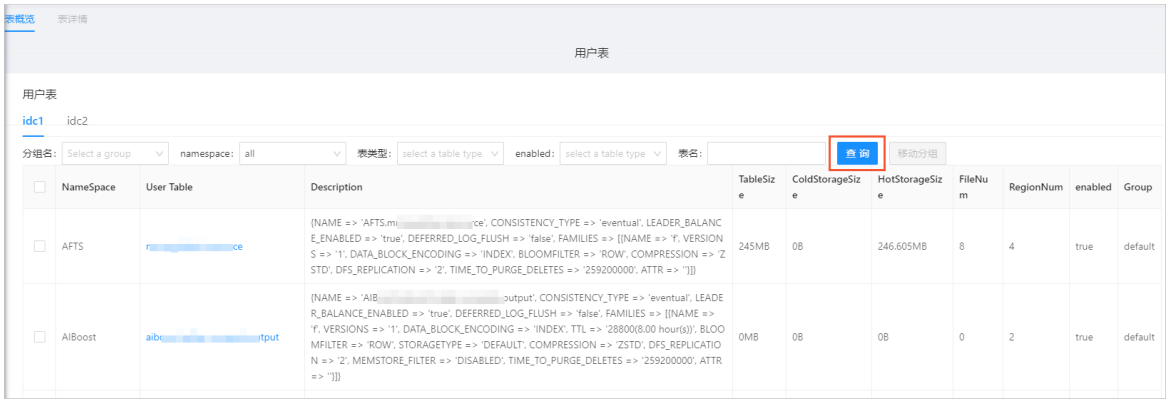
筛选表

您可以选择表的筛选条件，包括分组名、Namespace、表类型、表状态以及表名进行查询过滤。

R

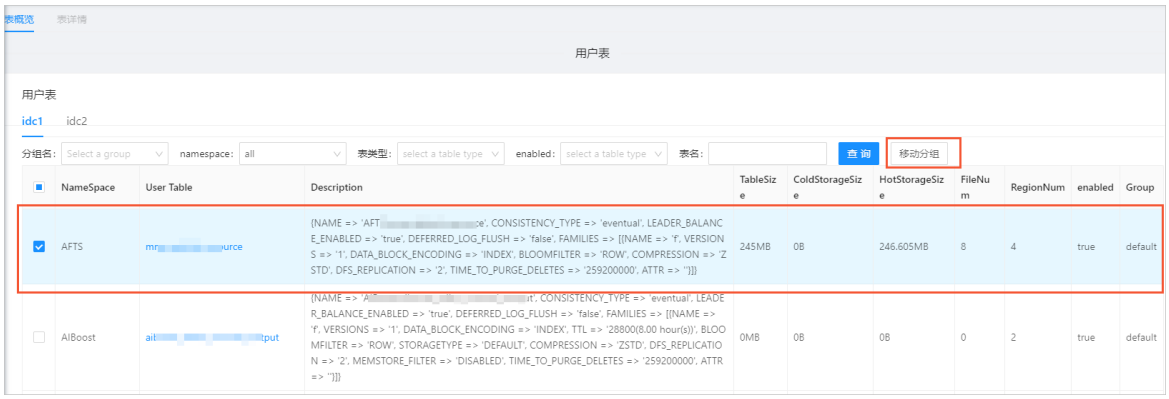
1. 访问Lindorm Insight。具体操作请参考如何访问Lindorm Insight？ 。
2. 在左侧导航栏单击**数据管理 > 表管理**。

3. 在**表概览**页签，根据分组名、Namespace、表类型、表状态以及表名，单击**查询**。

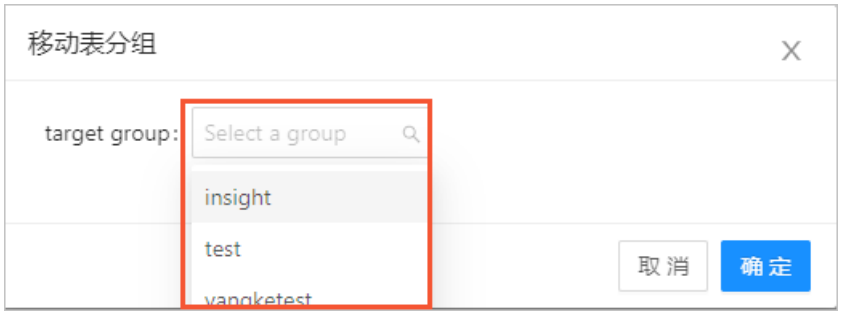


移动表分组

1. 在**表概览**页签的用户表中，选择需要移动分组的表，单击**移动分组**。



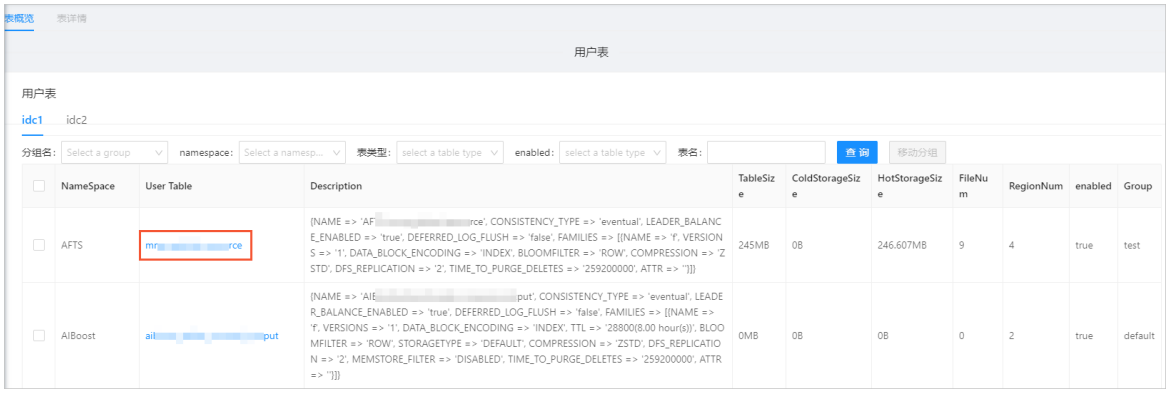
2. 在**移动表分组**对话框中，下拉选择**target group**，单击**确定**。



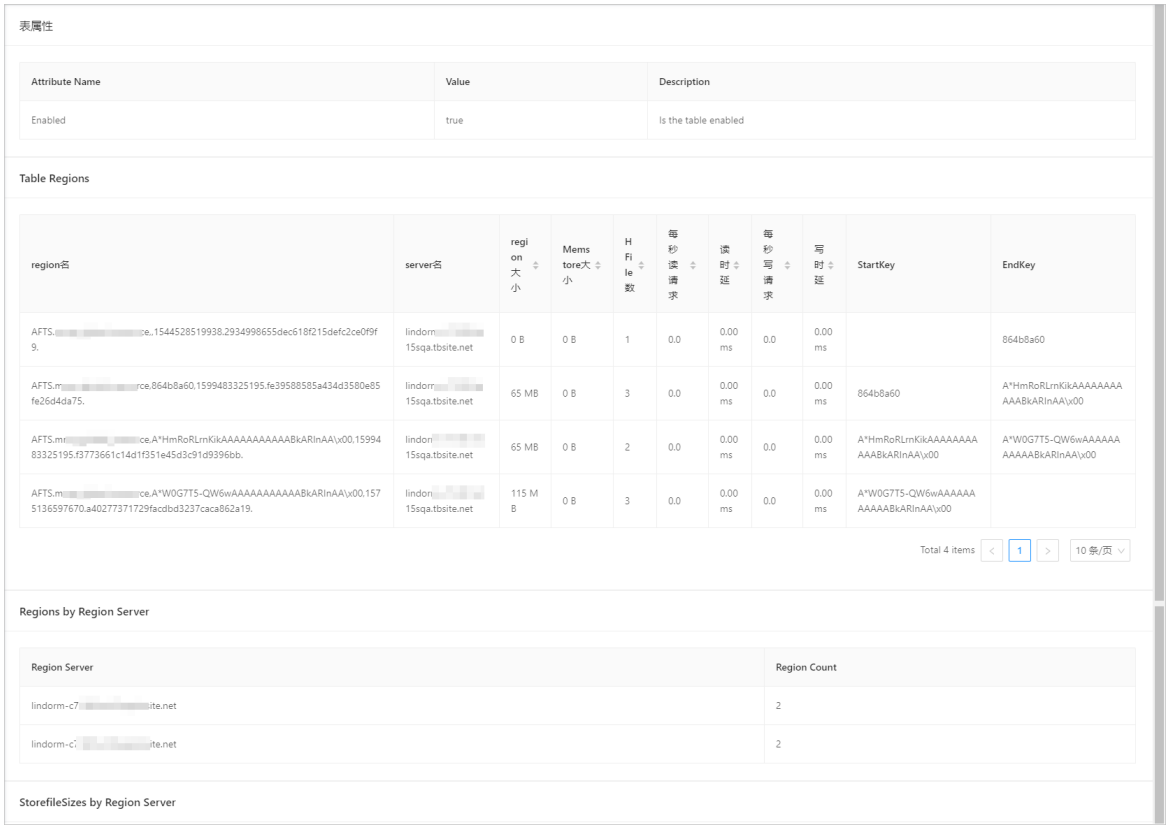
查看表详情

在**表详情**页签中，您可以查询表的详情信息，包括表属性、所有Table Region的详情信息、按照节点统计表region的分布情况（Regions by Region Server）以及按照节点统计StorefileSizes分布情况（StorefileSizes by Region Server）。

1. 在**表概览**页签的用户表中，单击表链接跳转到**表详情**页签。



2. 在**表详情**页签，可以查看表属性、Table Region的详情信息、Regions by Region Server、StorefileSizes by Region Server。



2.2.3. 账号和权限管理

2.2.3.1. 用户管理

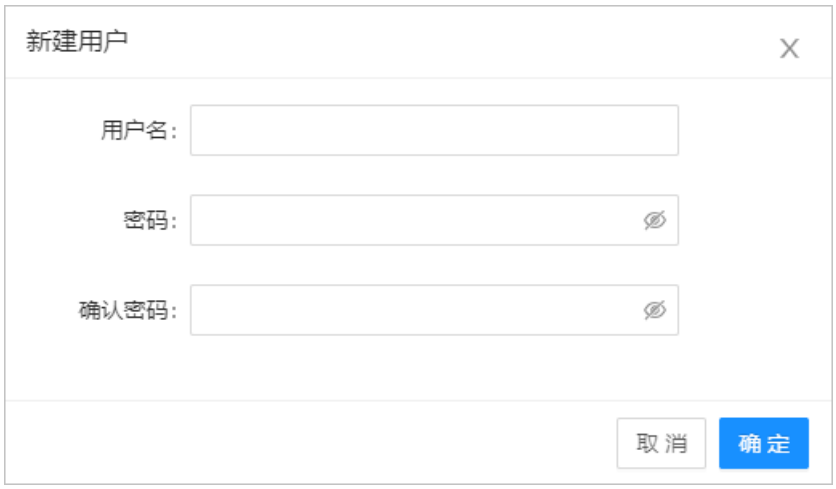
Lindorm提供一套简单易用的用户认证和ACL体系，只需要在配置中简单的填写用户名、密码即可完成用户的认证。用户的密码在服务器端以非明文的形式进行存储，并且在认证过程中不会以明文的形式传输密码，即使验证过程的密文被拦截，用以认证的通信内容不可重复使用，无法被伪造。

您可以使用Lindorm insight非常方便地管理用户，**用户管理**页面中列出了当前集群内的所有用户列表。当您购买集群后，系统会自动创建一个用户名为root，密码为root的账号，该账号拥有集群的所有权限。您可以使用这个账号访问集群，也可以通过Lindorm insight修改这个账号的密码，或者删除这个账号。

创建新用户

1. 访问Lindorm Insight。具体操作请参考如何访问Lindorm Insight？。
2. 在左侧导航栏，单击**数据管理 > 用户管理**。
3. 在**用户列表**页面，单击**更多操作 > create user**。

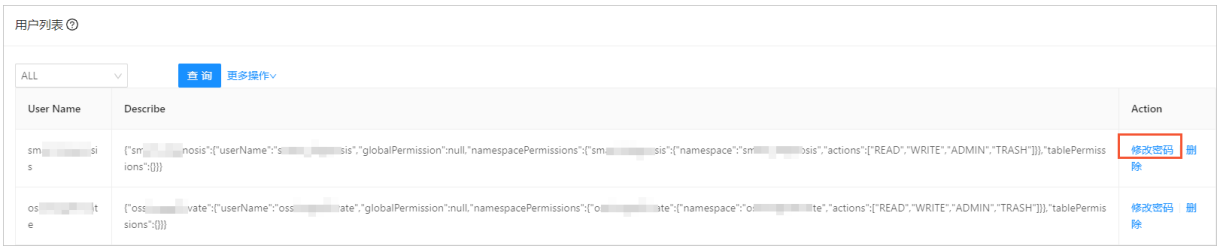
4. 在**新建用户**对话框中，输入用户名、密码，确认密码后单击**确定**。



> **说明**
>
> - Lindorm在服务器端不会以明文形式存储密码，因此创建用户后无法再查看该用户的密码，用户需要记住自己设置的密码，忘记密码后，只能通过修改密码的方式来更改密码。
> - 新建的用户没有任何权限，需要在**ACL 管理**页面赋予相应的权限之后，才能够正常访问。具体请参见ACL管理。

修改用户密码

在**用户列表**页面，定位到需要修改密码的用户，单击**修改密码**。



删除用户

在**用户列表**页面，定位到需要删除的用户，单击**删除**。

2.2.3.2. ACL管理

在ACL管理页面中，您可以管理相应用户所拥有的权限，可以赋予(grant)、回收（revoke）某个用户的一项或者多项权限。

权限种类及权限层级

在Lindorm集群中，服务器会根据每个用户拥有的权限去判断该用户是否能够执行某项操作。例如user1如果只有Table1的读权限，那么它在写Table1的时候就会报错，在读写Table2的时候都会被拒绝访问。Lindorm拥有的权限种类如下表所示：

权限种类	权限说明
WRITE权限	与写Lindorm表相关的操作，如Put、Batch、Delete、Increment、Append、CheckAndMutate等。
READ权限	与读Lindorm表相关的操作，如Get、Scan、exist等。读取Lindorm表descriptor，namespace列表等相关操作，如getTableDescriptor、listTables、listNamespaceDescriptors等。
ADMIN权限	不会涉及到表删除和表数据删除的DDL操作如createTable、enable/disableTable、Namespace相关DDL操作，如createNamespace等。
TRASH权限	为了防止表误删和表数据被清空，只有授予了TRASH权限的用户，才能调用truncateTable和deleteTable这两个DDL操作。
SYSTEM权限	只有赋予了SYSTEM权限的用户，才能执行Compact、flush等运维操作。另外，使用LTS迁移、同步HBase增强版时，用户也必须有SYSTEM权限。

目前在Lindorm中有三个权限层级，Global、Namespace、Table，这三者之间是相互覆盖的关系。例如给user1赋予了Global的读写权限，则它就拥有了所有Namespace下所有Table的读写权限；如果给user2赋予了Namespace1的读写权限，那么他会自动拥有Namespace1中所有表的读写权限（包括在Namespac1中新建的表）。

> **说明** 只有拥有Global层级的ADMIN权限的用户，才能够Create和Delete Namespace。

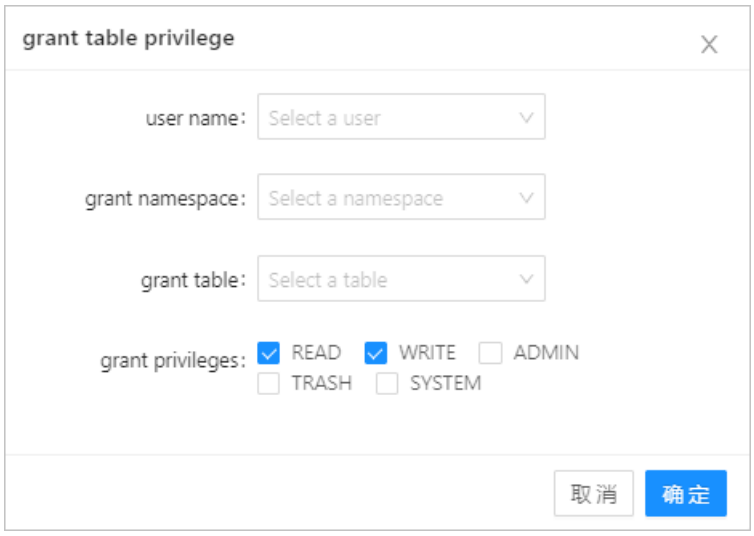
grant权限

您可以基于Global、Namespace、Table三个权限层级来给指定的用户授予相应的权限。本文档以给指定的用户授予某个表的读写权限为例进行说明。

1. 访问Lindorm Insight。具体操作请参考如何访问Lindorm Insight？。

2. 在左侧导航栏，单击**数据管理 > ACL管理**。
3. 在**ACL管理**页面的**Table权限**区域，单击**更多操作 > grant privilege**。

4. 在**grant table privilege**对话框中，下拉选择**user name**、**grant namespace**、**grant table**，选择**grant privileges**。单击**确定**。

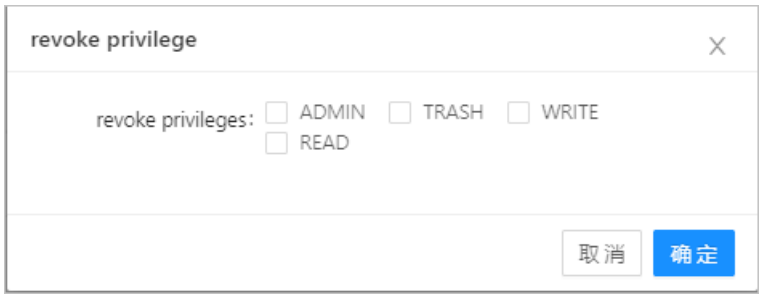


revoke权限

您可以在Lindorm Insight系统中的ACL管理收回对应用户的权限。每个用户可能有多个层级的权限，您可以在对应层级的权限列表中找到对应用户，然后单击**revoke**按钮，然后选择需要revoke的权限。本文档以收回某个指定的用户某个表的读写权限为例进行说明。

1. 在**ACL管理**页面的**Table权限**区域，定位到需要收回权限的用户，单击**revoke**。

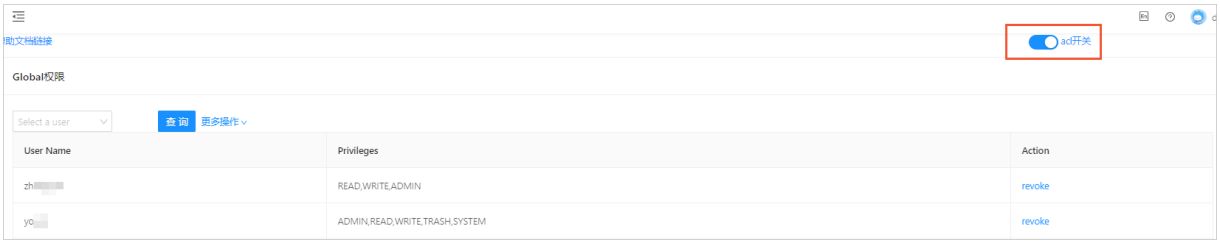
2. 在**revoke privilege**对话框中，选择需要收回的权限，单击**确定**。



打开和关闭ACL功能

如果您不需要使用用户名和ACL功能来进行访问控制，可以将ACL功能关闭。ACL功能关闭后，后续所有的访问（包括API访问，SQL访问和非JAVA访问）时都不需要提供用户名和密码，您可以没有限制的做任何操作。

您无需重启集群，就可以动态地打开和关闭ACL功能。但如果之前ACL是关闭状态，动态打开后，没有提供用户名和密码的客户端在重连时，会因为无法认证而报错。提供了用户名和密码的客户端在重连时会被正常认证，但在做越权操作时，会被拒绝访问。



2.3. 数据查询

在开发调试或者生产运维过程中，往往需要去Lindorm中查询某条数据。除了使用Lindorm shell来写Get、Scan请求，Lindorm insight提供了一个简单的SQL查询入口，您可以使用SQL语法来查询Lindorm的表。

注意事项

在您执行SQL前，请您务必先阅读下面的几点重要说明，能有效解决您使用时可能遇到的问题或疑惑：

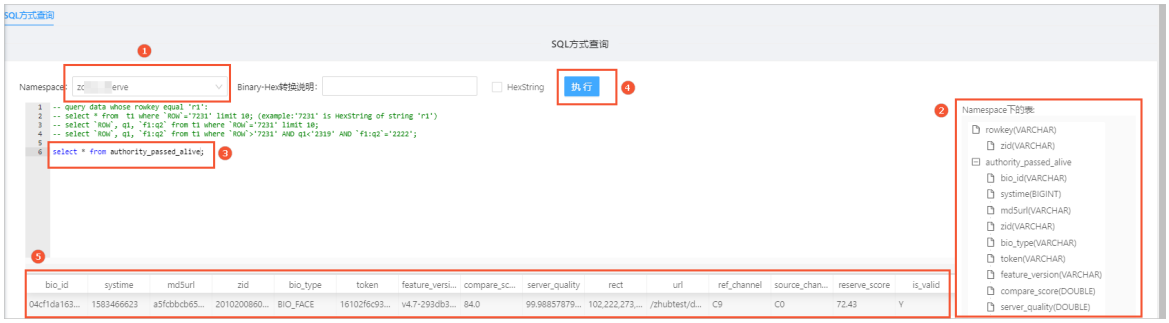
- 本系统只支持**SELECT**查询语句，如果需要变更数据操作，需要通过命令行或者使用产品提供的API自行开发应用。
- 为确保数据安全，本系统每次查询返回最多100条数据。
- 对所有**varbinary**类型的字段进行条件查询时，必须使用**HexCode**编码的字符串作为value 。
- **ROW**字段对应HBase的**rowkey**，**ROW**和**qualifier**都是 **varbinary**类型，**qualifier**如果不属于**family**，则需要指定**family**，例如： `select `ROW`,q1,`f1:q2` from ...` 。
- **ROW**和**COL**是SQL保留字，查询时需要加反引号；**qualifier**指定**family**时也需要加反引号。

操作步骤

从Lindorm Insight中的**SQL方式查询**页面可以进入数据查询系统。在查询前，先要选择表所在Namespace，选择了Namespace后，页面右侧的树形结构中会展示出这个Namespace中所有的表；单击表名可以显示这个表的schema，ROW代表RowKey， COL是预置的列名。您可以根据这个表的schema来构造select请求。

> **说明** Lindorm insight的SQL查询页面不支持use namespace语法，只能通过左上角的Namespace列表选择或切换Namespace。

1. 访问Lindorm Insight。具体操作请参考如何访问Lindorm Insight？ 。
2. 在左侧导航栏，单击**数据查询**。
3. 在**SQL方式查询**页面，下拉选择Namespace，页面右侧会展示出当前Namespace下的所有的表。参考这些表输入查询SQL，单击**执行**。

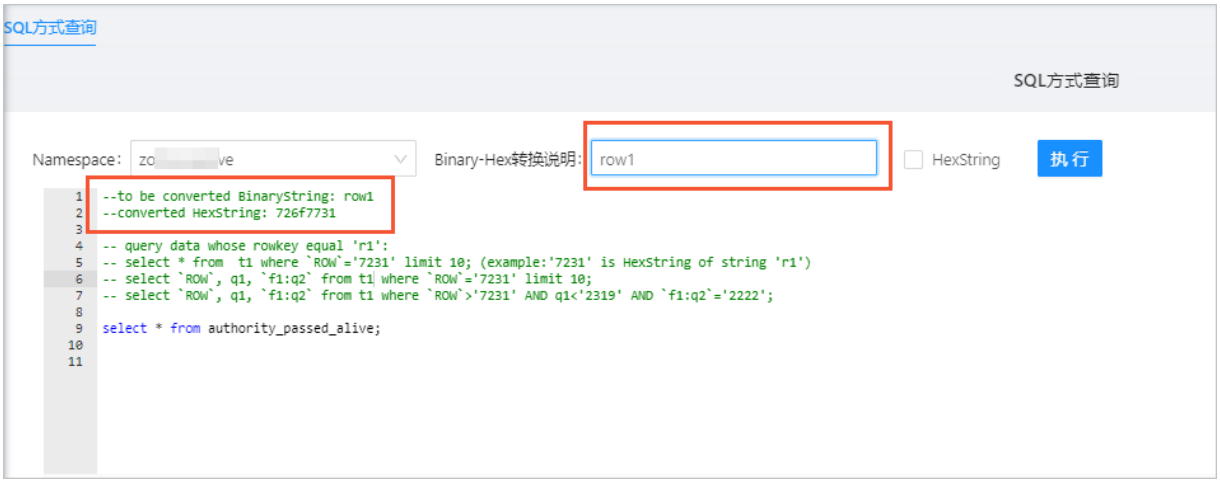


转换Binary编码字符与HexCode编码字符

Lindorm内部使用 byte[] 的形式存储数据，在**SQL方式查询**页面中返回的查询结果中， varbinary 类型的字段以**Lindorm BinaryString**的编码形式展示。

在 varbinary 类型的字段进行条件查询时（即 where 子句中包含如 rowkey 等字段），必须使用**HexCode**编码的字符作为查询条件的value。例如：查询 rowkey 为 r1 数据，则SQL查询中的 where 子句应该写成 where rowkey='7321' （字符串 r1 的HexCode编码字符为 7321 ）。

为方便用户在查询数据时进行编码转换，Lindorm Insight系统在**SQL方式查询**页面中提供了一个简单的转换工具，只需将您的**Binary**编码字符串输入转换框中，即可在SQL编辑器内看到该字符串相应的**HexCode**编码，如下图所示：



如果您需要查看**HexCode**编码的查询结果，可以勾选**HexString**单选框，系统会将查询返回结果中varbinary类型的字段都转为**HexCode**编码的字符串，如下图所示：

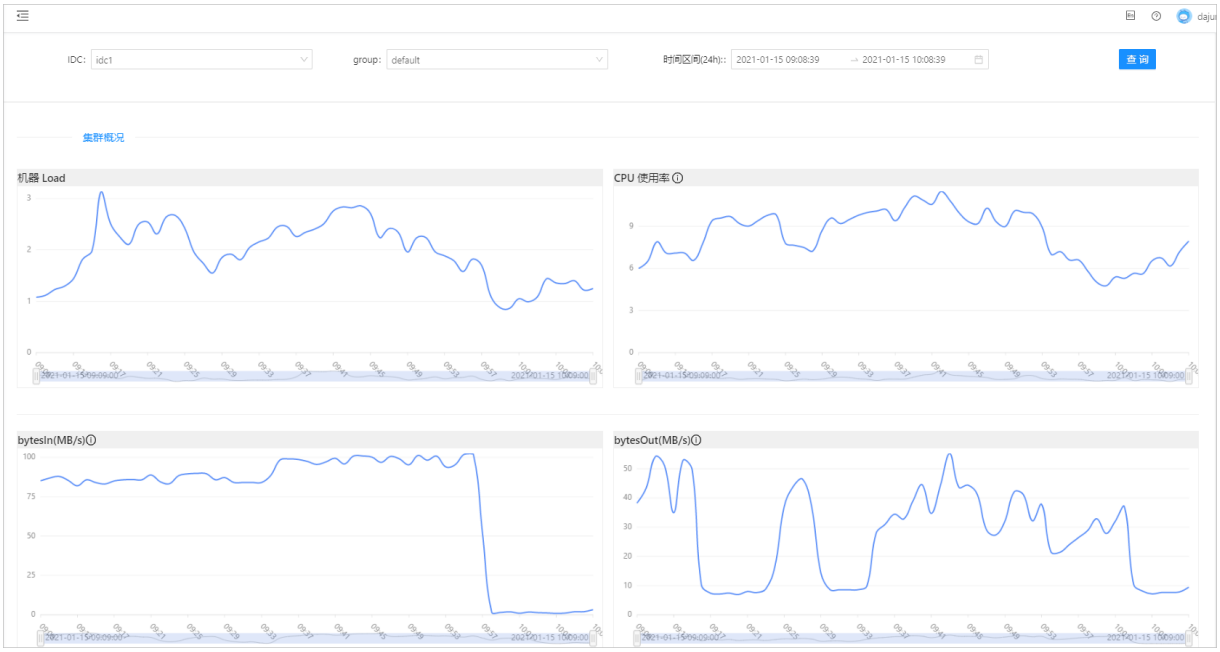
2.4. 数据监控

2.4.1. 集群监控

Lindorm Insight为用户提供了集群相关的性能监控数据，方便用户查看集群的使用情况。集群监控主要从IDC和group维度监控集群的运行状况。

访问集群监控页面

进入Lindorm Insight系统，单击左侧导航栏的**监控 > 集群监控**即可进入Lindorm Insight**集群监控**页面。您可以根据IDC、group、时间区间去筛选查询集群监控详情。



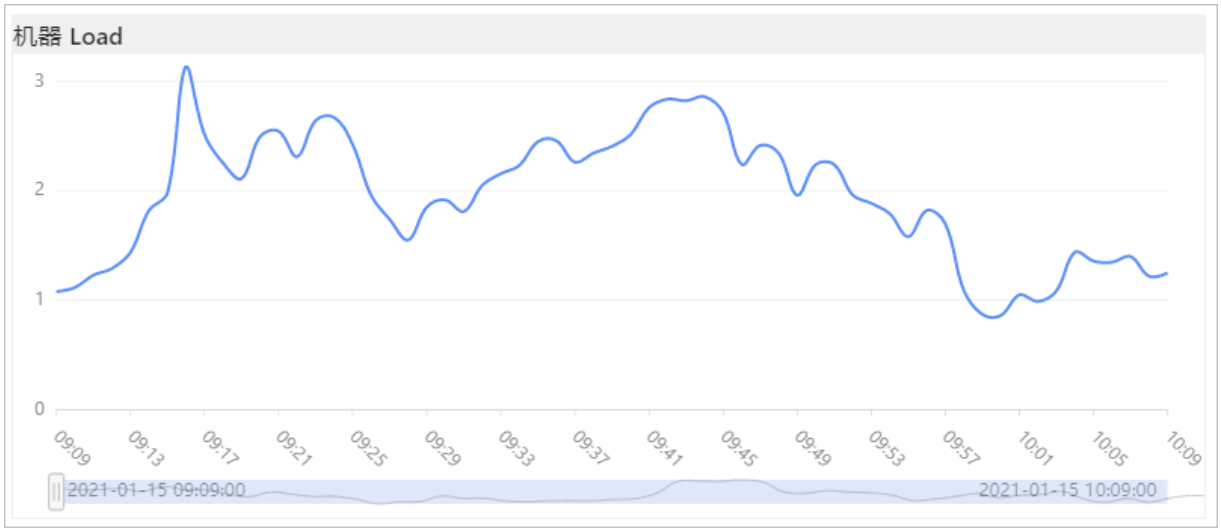
集群监控页面分为**集群概况**和**QP层概况**两个层级，这两个层级又分为多个区域，您可以单击以下区域链接查看对应的内容。

- **集群概况**：
  - 机器Load
  - CPU使用率
  - bytesIn
  - bytesOut
- **QP层概况**：QP（QueryProcessor）是Lindorm服务端的查询处理层。Lindorm客户端通过QP获取服务。它的指标能准确的反映出当前集群的运行指标。
  - 集群整体查询的行吞吐
  - 集群整体查询的响应时间
  - 集群整体写的行吞吐
  - 集群整体写的响应时间
  - 集群WGet详情
  - 集群WPut详情
  - 集群WScan详情
  - 集群WDelete详情
  - 集群Select详情

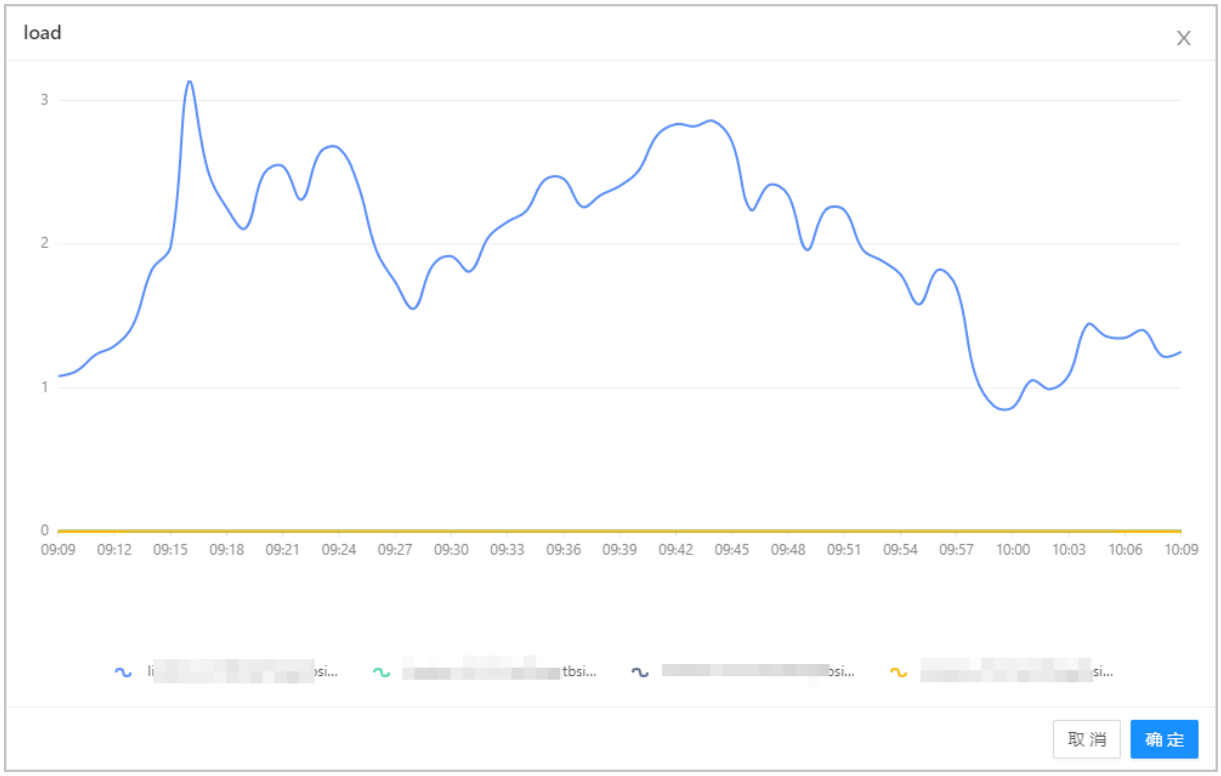
- QP Upsert详情

机器Load

机器Load区域展示了集群机器的平均负载。



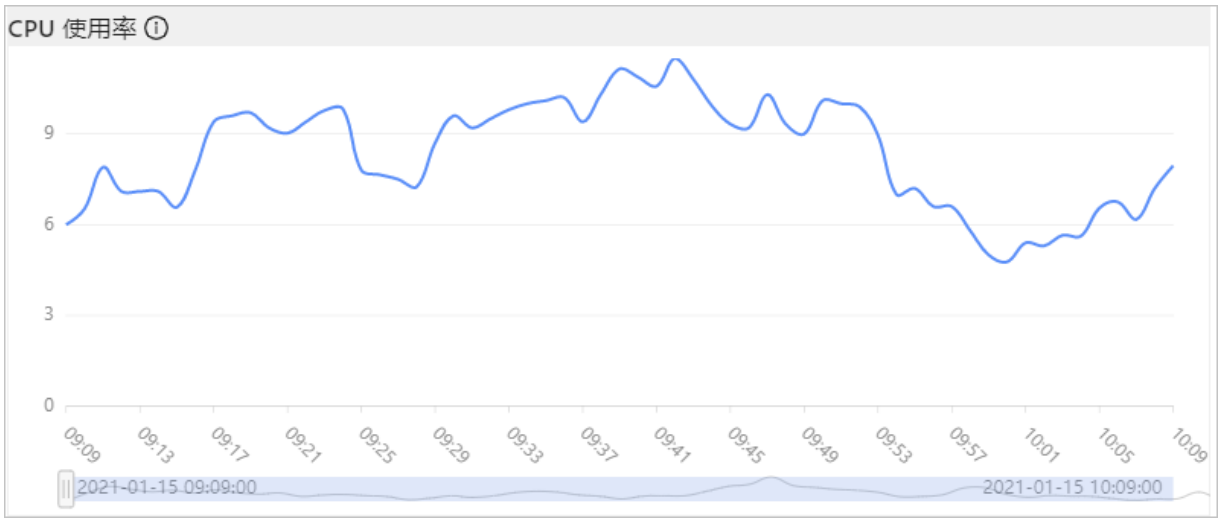
如果您想查看某台机器的具体指标，双击指标曲线，就会弹出机器列表对应的指标数据。



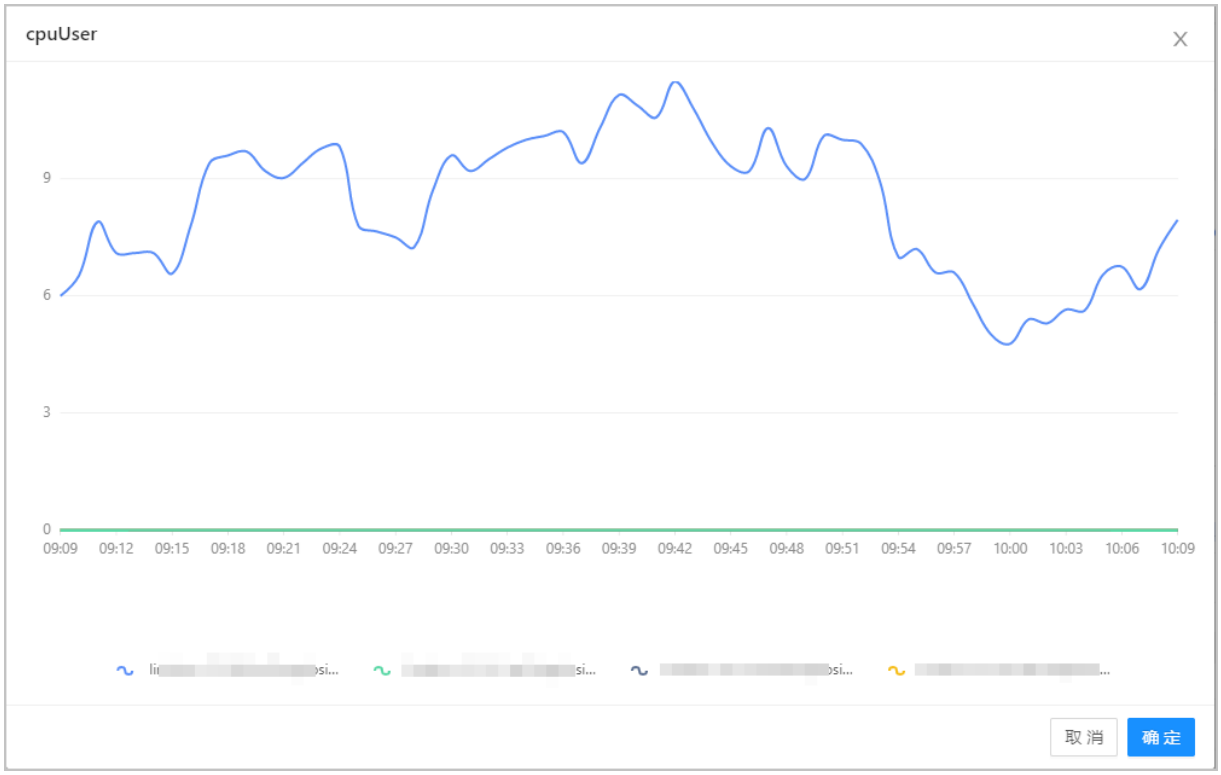
> **说明** 默认显示的是指标值Top10的机器列表，如果机器数过多，您可以手动单击机器名称来显示或者关闭指标曲线。

CPU使用率

CPU使用率区域展示了集群机器的用户CPU平均使用率。



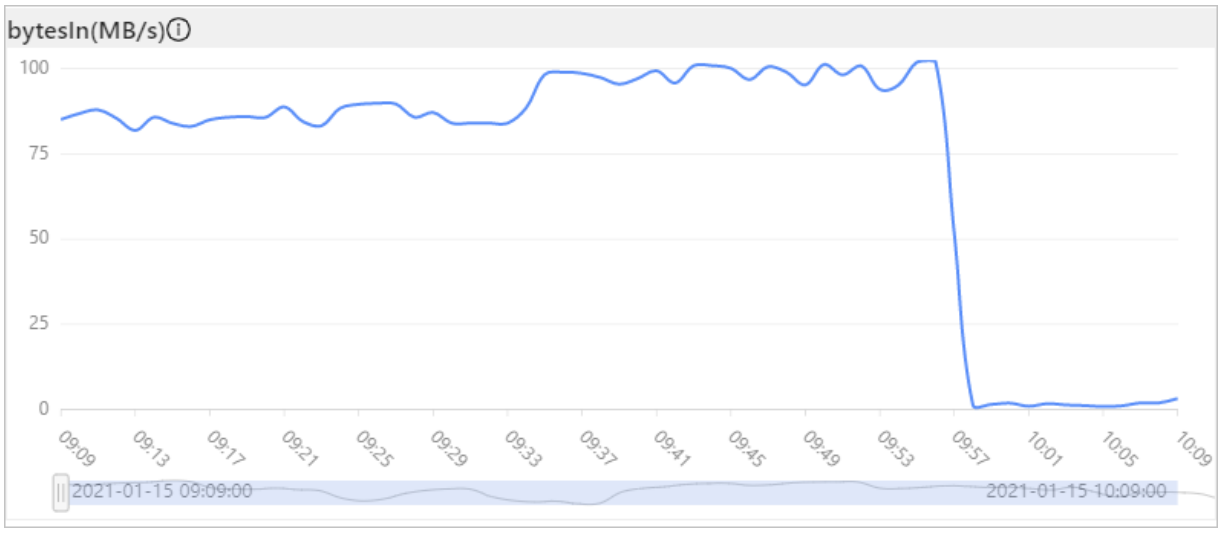
如果您想查看某台机器的具体指标，双击指标曲线，就会弹出机器列表对应的指标数据。



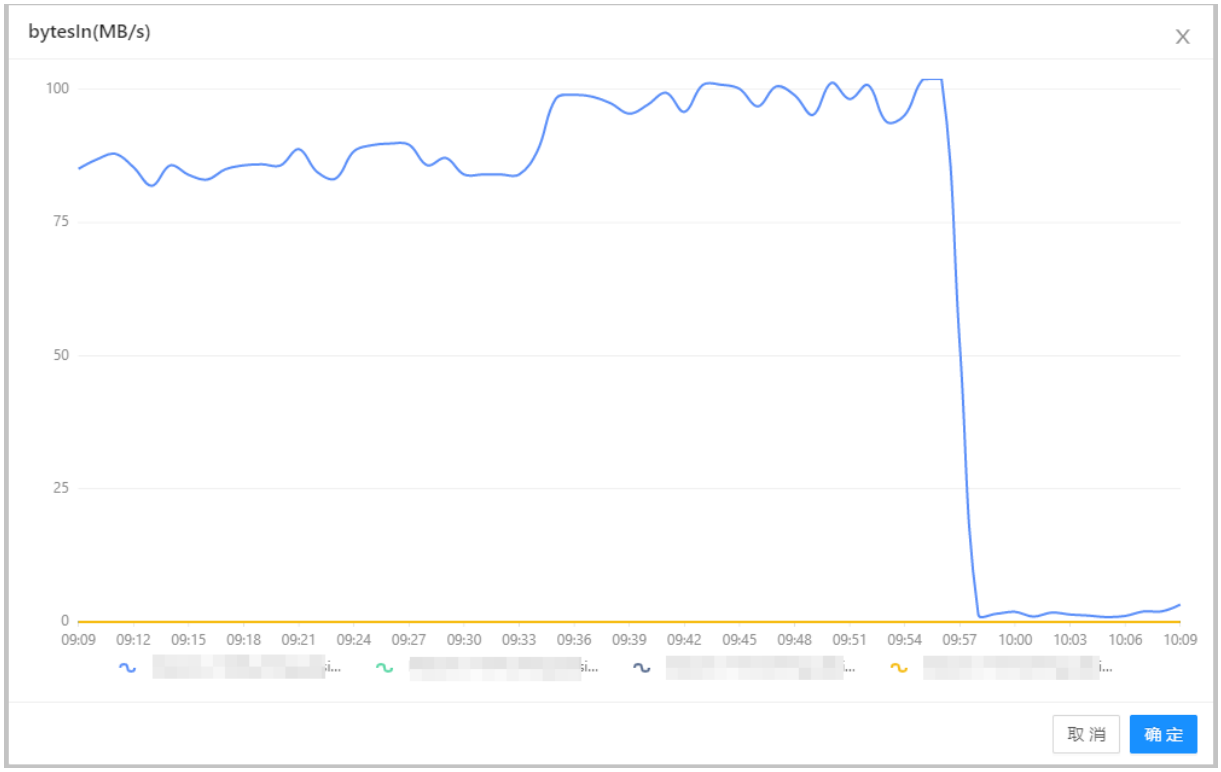
说明 默认显示的是指标值Top10的机器列表，如果机器数过多，您可以手动单击机器名称来显示或者关闭指标曲线。

bytesIn

bytesIn区域展示了集群的网络流入流量。



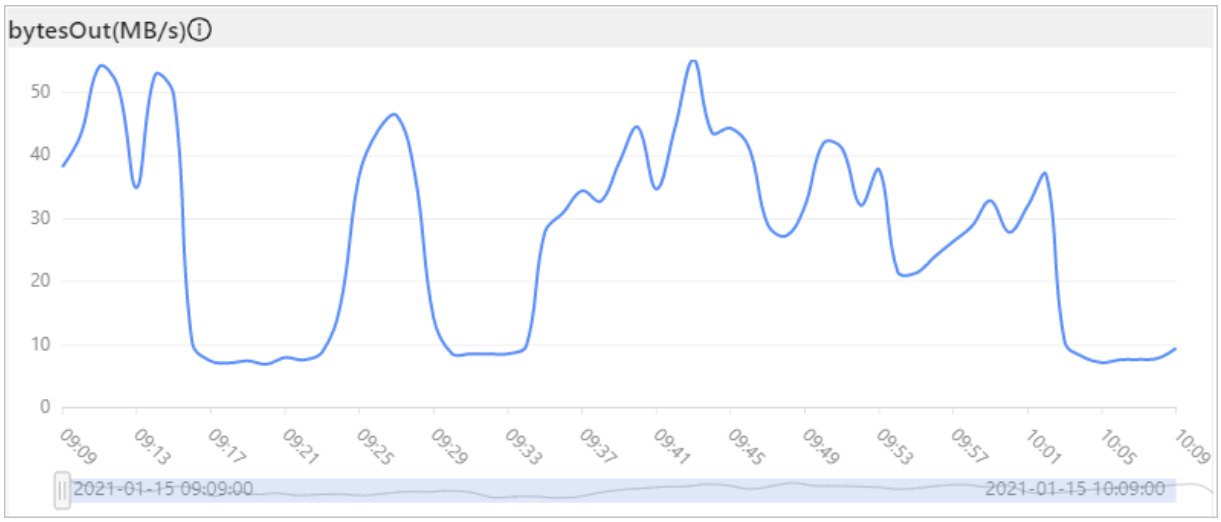
如果您想查看某台机器的具体指标，双击指标曲线，就会弹出机器列表对应的指标数据。



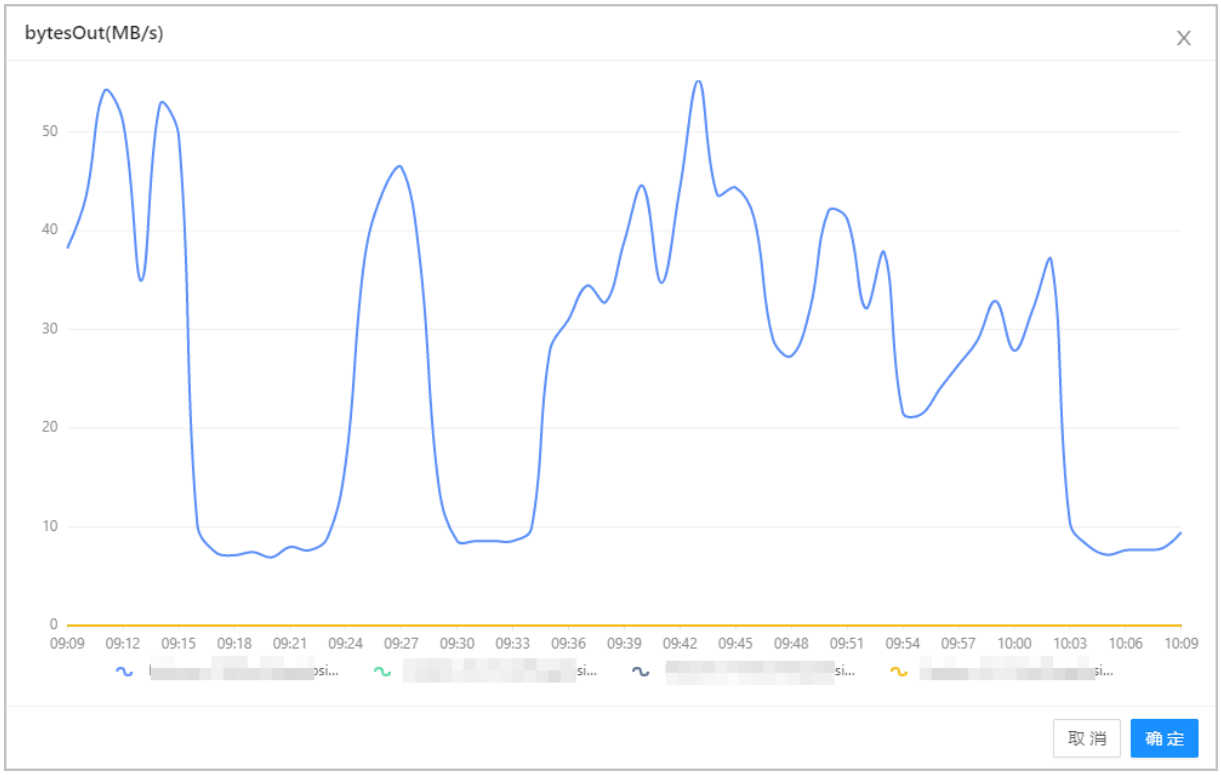
说明 默认显示的是指标值Top10的机器列表，如果机器数过多，您可以手动单击机器名称来显示或者关闭指标曲线。

bytesOut

bytesOut区域展示了集群的网络流出流量。



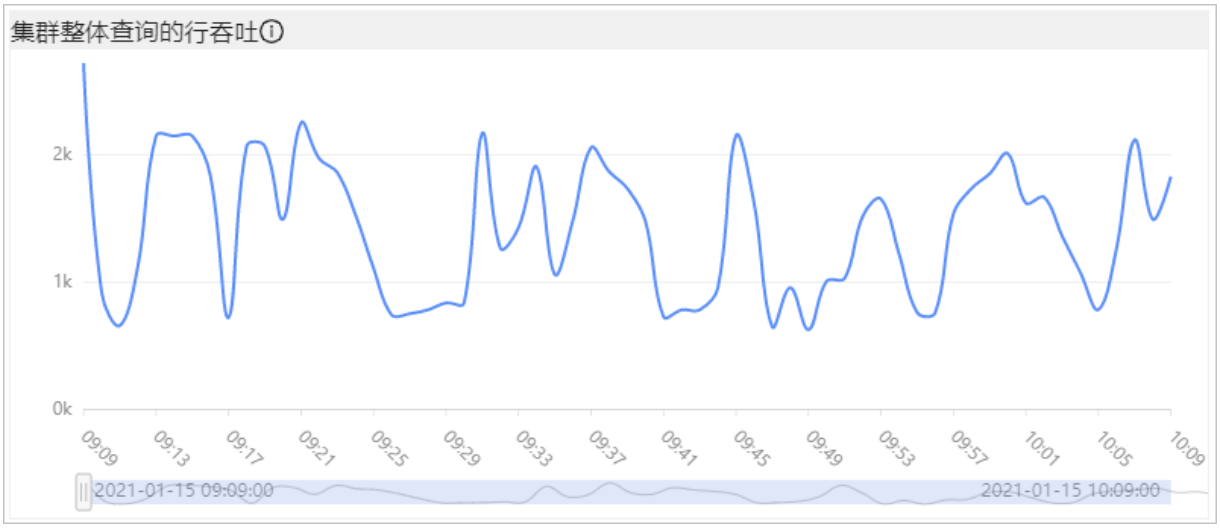
如果您想查看某台机器的具体指标，双击指标曲线，就会弹出机器列表对应的指标数据。



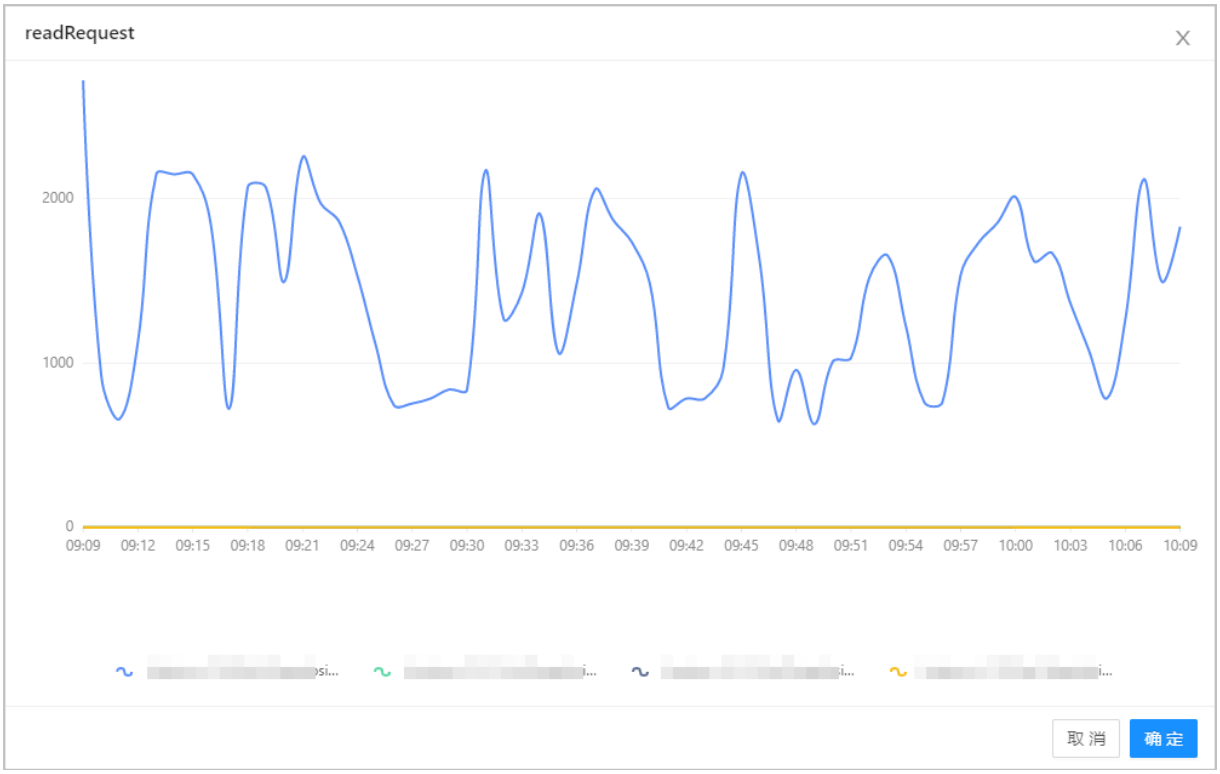
说明 默认显示的是指标值Top10的机器列表，如果机器数过多，您可以手动单击机器名称来显示或者关闭指标曲线。

集群整体查询的行吞吐

集群整体查询的行吞吐区域展示了集群执行查询 (select,wget,wscan等)时，底层扫描的行数总和，是集群各种查询方式聚集后的一个整体read负载指标，是原监控系统的Lindorm Server端读请求'指标。



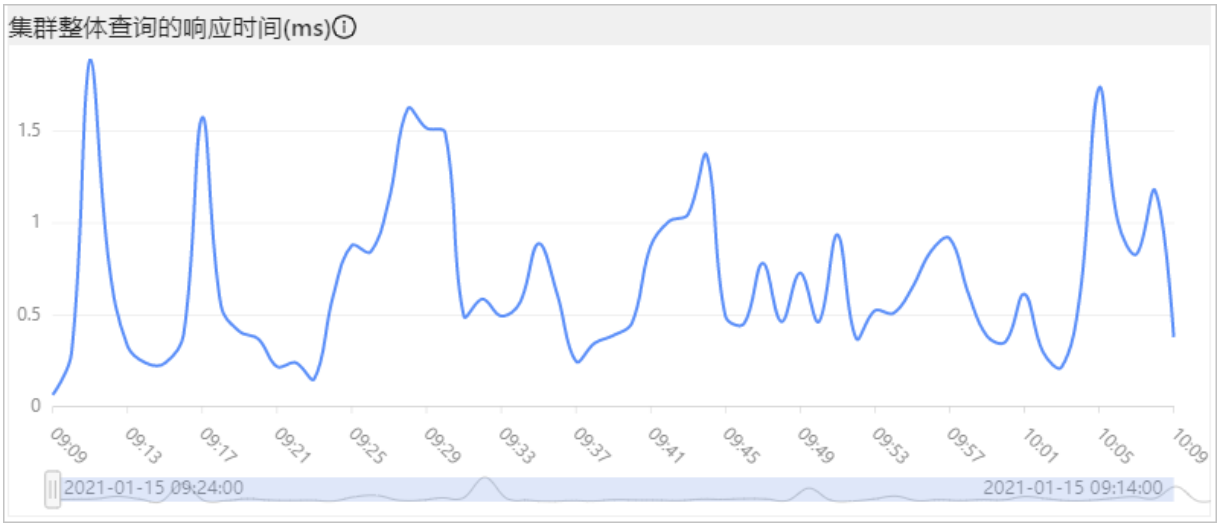
如果您想查看某台机器的具体指标，双击指标曲线，就会弹出机器列表对应的指标数据。



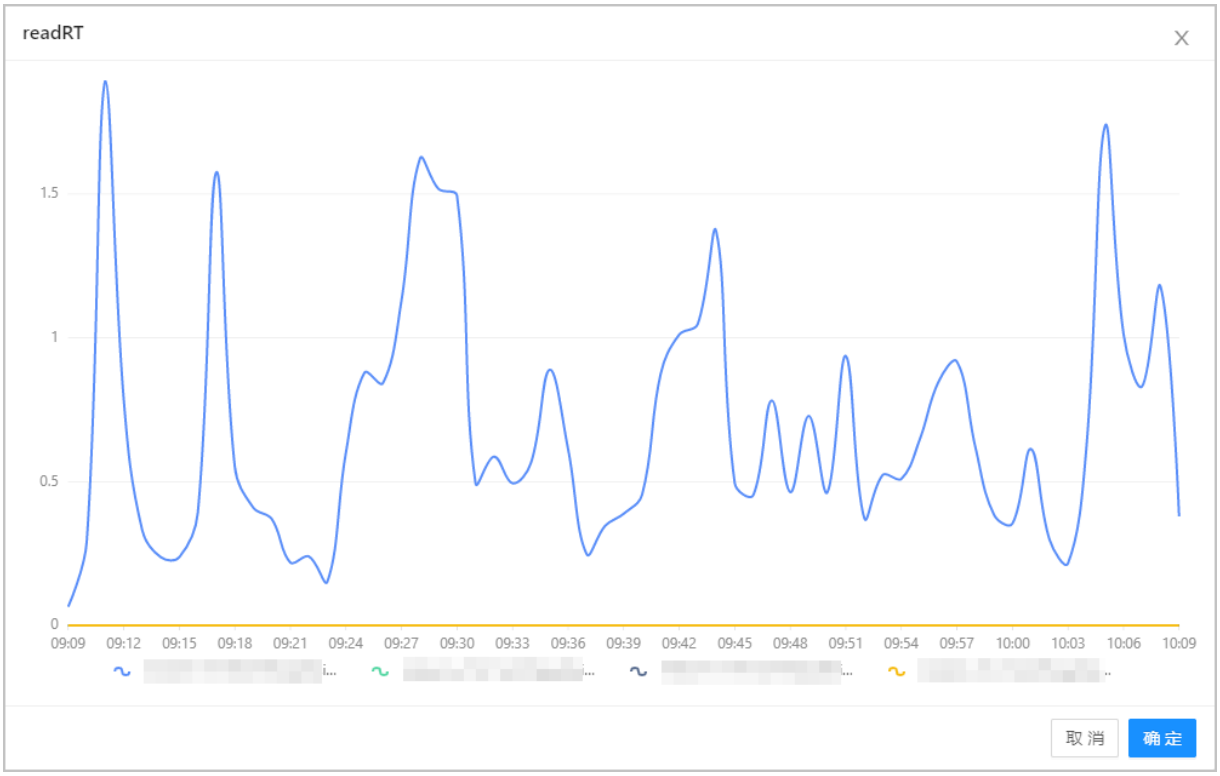
说明 默认显示的是指标值Top10的机器列表，如果机器数过多，您可以手动单击机器名称来显示或者关闭指标曲线。

集群整体查询的响应时间

集群整体查询的响应时间区域展示了集群read每行需要的平均时间。响应时间=单位时间内集群查询的总时间/底层扫描的行数，作为 read 每行需要的平均时间，是一个整体值。如果要查询具体查询类型的指标，请参考后面的WScan，WGet 等请求详情。



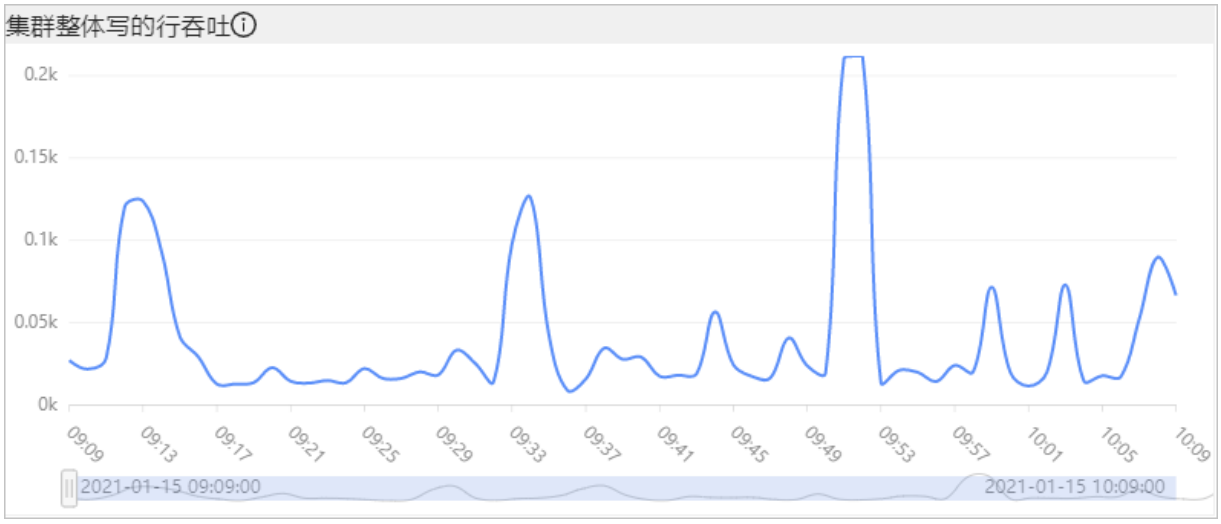
如果您想查看某台机器的具体指标，双击指标曲线，就会弹出机器列表对应的指标数据。



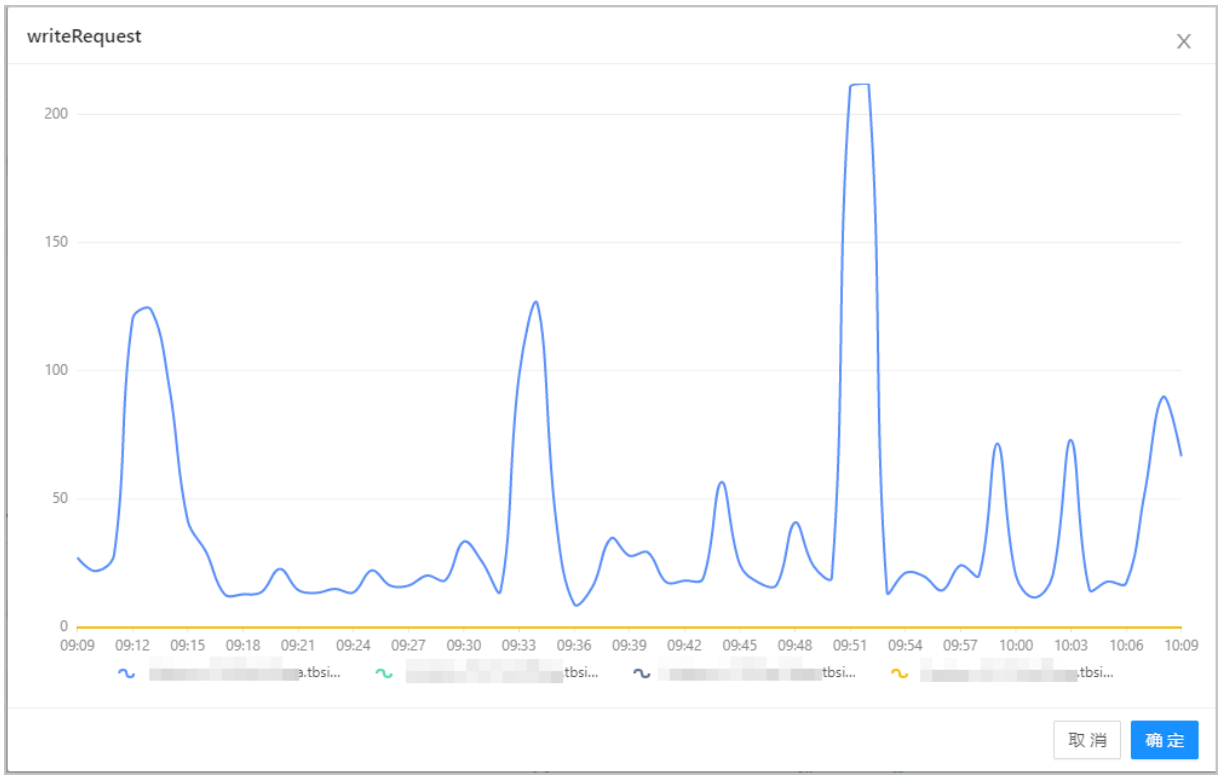
> **说明** 默认显示的是指标值Top10的机器列表，如果机器数过多，您可以手动单击机器名称来显示或者关闭指标曲线。

集群整体写的行吞吐

集群整体写的行吞吐区域展示了集群执行修改操作 (upsert,wput,wdelete 等)时，底层扫描的行数总和，是集群各种查询方式聚集后的一个整体 write 负载指标。



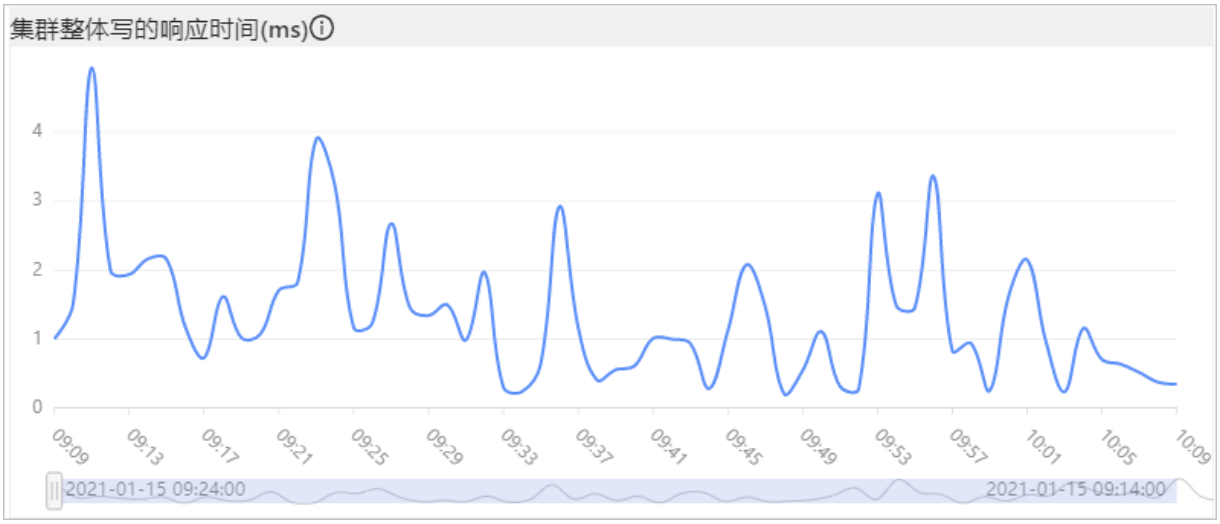
如果您想查看某台机器的具体指标，双击指标曲线，就会弹出机器列表对应的指标数据。



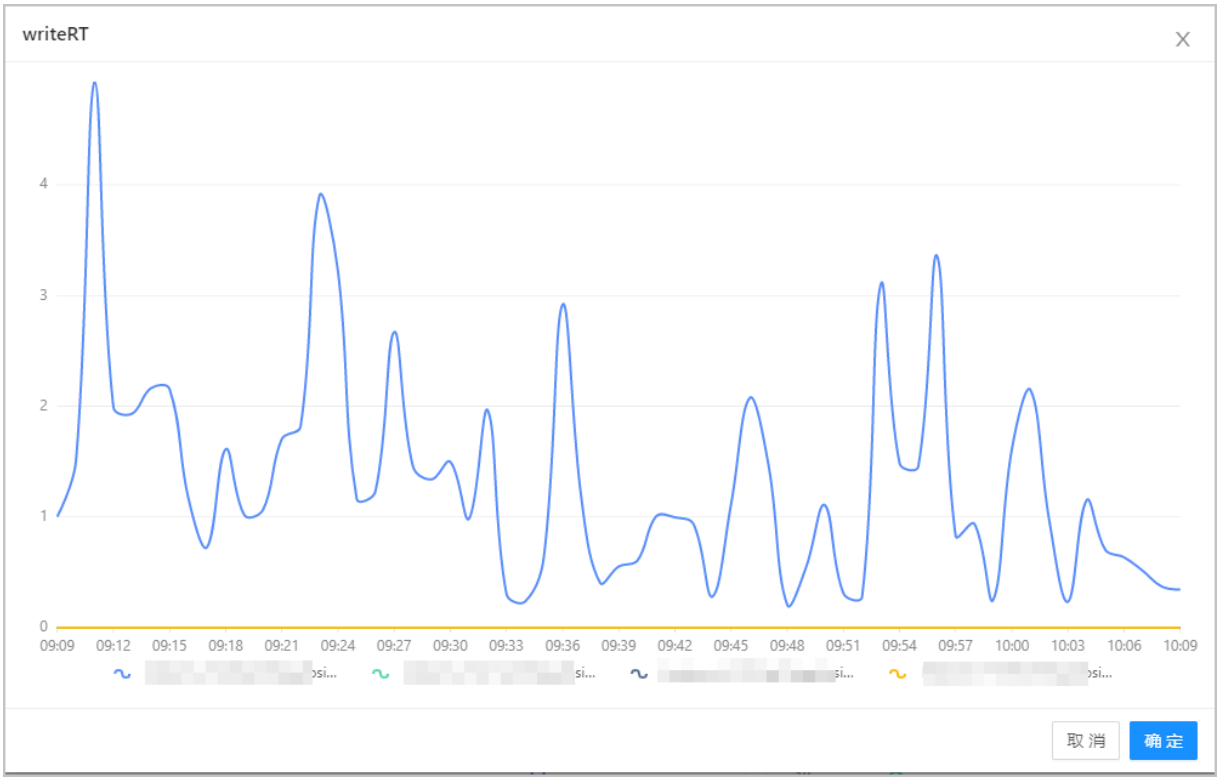
> **说明** 默认显示的是指标值Top10的机器列表，如果机器数过多，您可以手动单击机器名称来显示或者关闭指标曲线。

集群整体写的响应时间

集群整体写的响应时间区域展示了集群write 每行需要的平均时间。响应时间=单位时间内集群整体写的总时间/底层扫描的行数，作为 write 每行需要的平均时间，是一个整体值。如果要查询具体查询类型的指标，请参考后面的 WPut，WDelete 等请求详情。



如果您想查看某台机器的具体指标，双击指标曲线，就会弹出机器列表对应的指标数据。



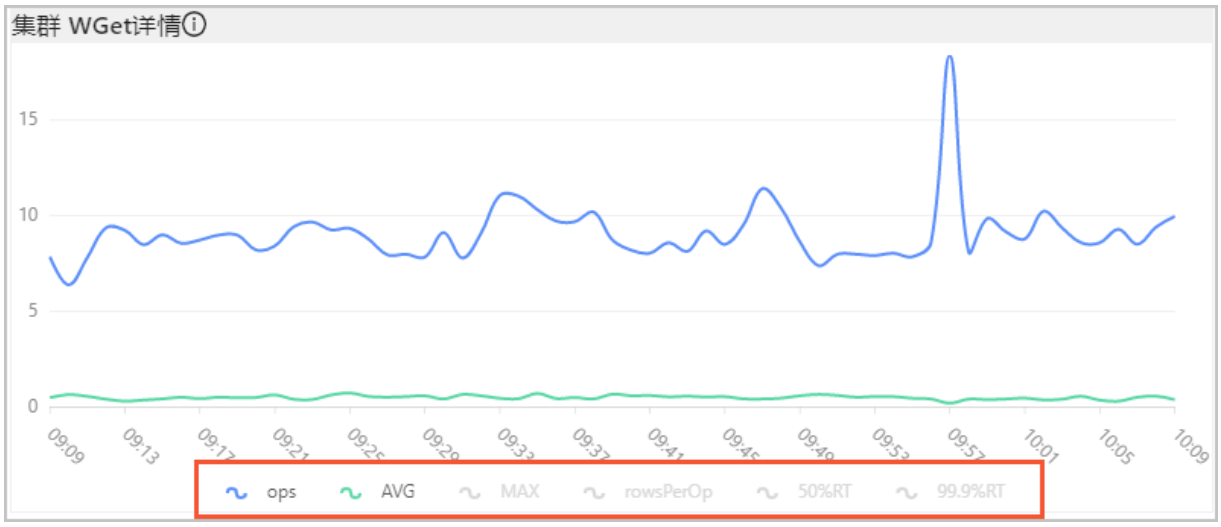
> **说明** 默认显示的是指标值Top10的机器列表，如果机器数过多，您可以手动单击机器名称来显示或者关闭指标曲线。

集群WGet详情

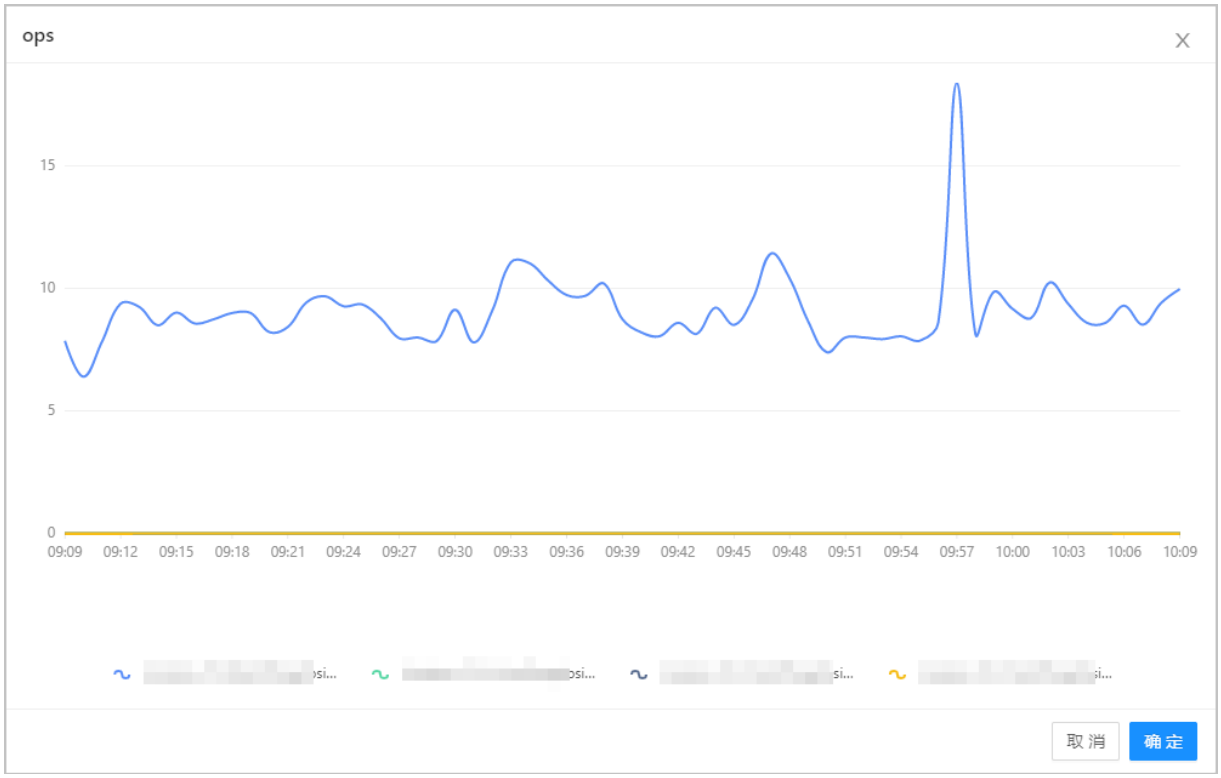
集群WGet详情区域展示了集群执行get命令的性能指标数据，指标数据如下：

- **ops**：每秒钟请求次数。

- **AVG**：请求的平均响应时间。
- **MAX**：请求的最大响应时间。
- **rowsPerOp**：表示平均每次操作影响的行数。
- **50%RT**：请求的中位数响应时间。
- **99.9%RT**：请求的 99.9% 响应时间。



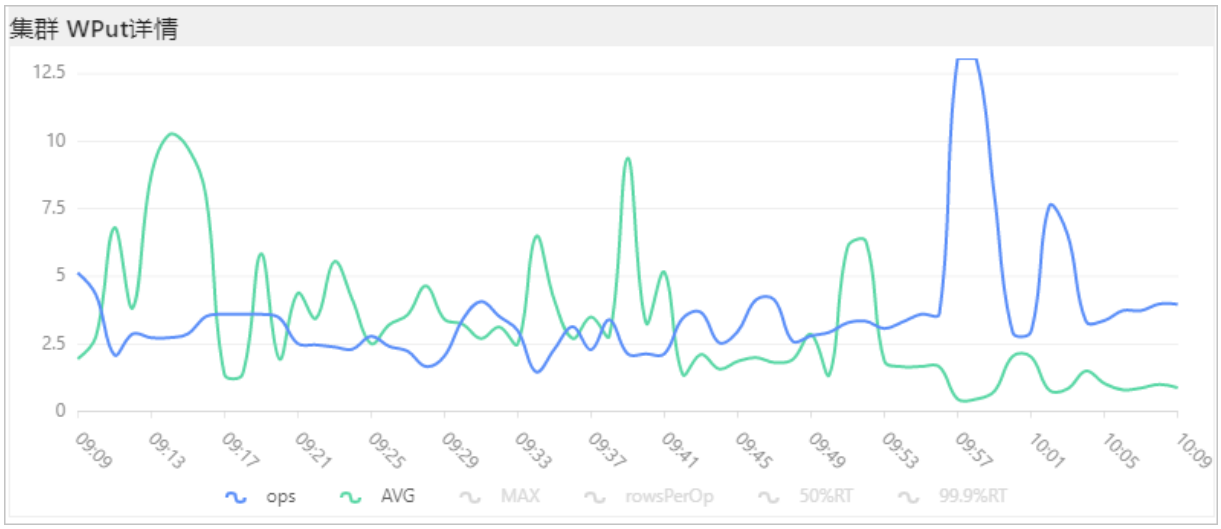
如果您想查看某台机器的具体指标，双击指标曲线，就会弹出机器列表对应的指标数据。



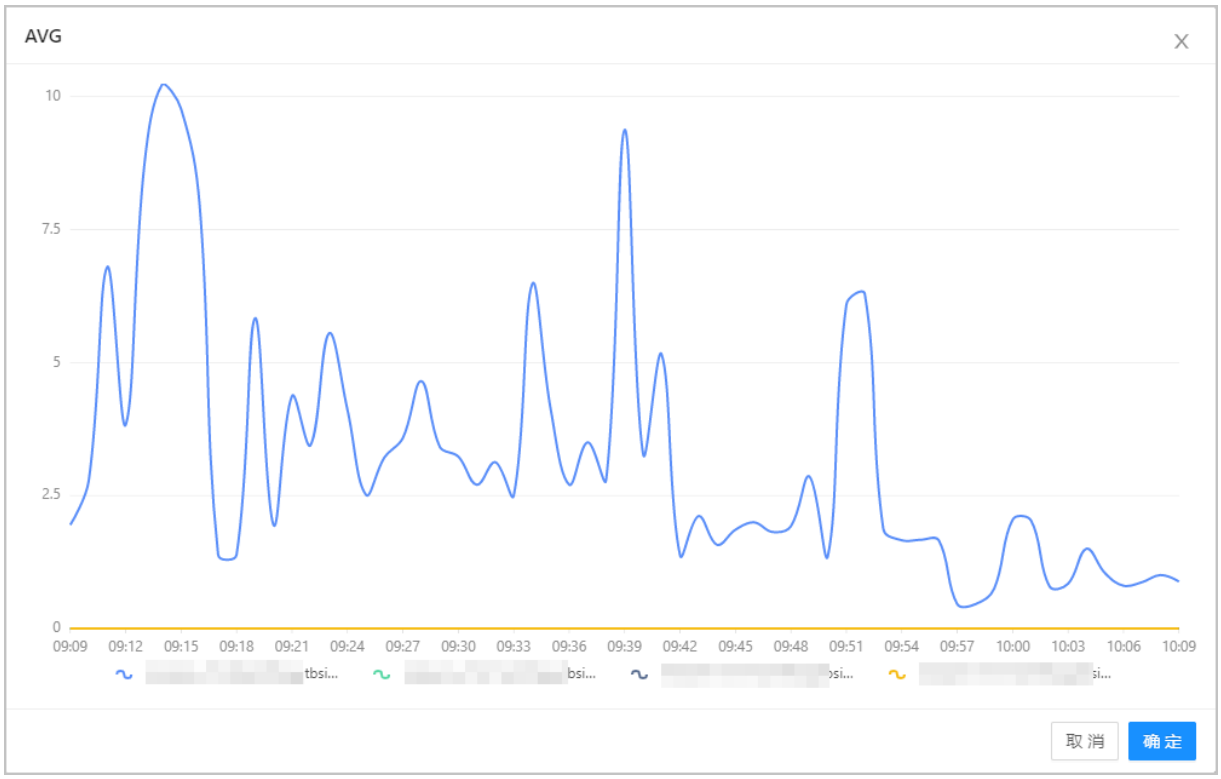
> **说明** 默认显示的是指标值Top10的机器列表，如果机器数过多，您可以手动单击机器名称来显示或者关闭指标曲线。

集群WPut详情

集群WPut详情区域展示了集群执行put命令的性能指标数据。



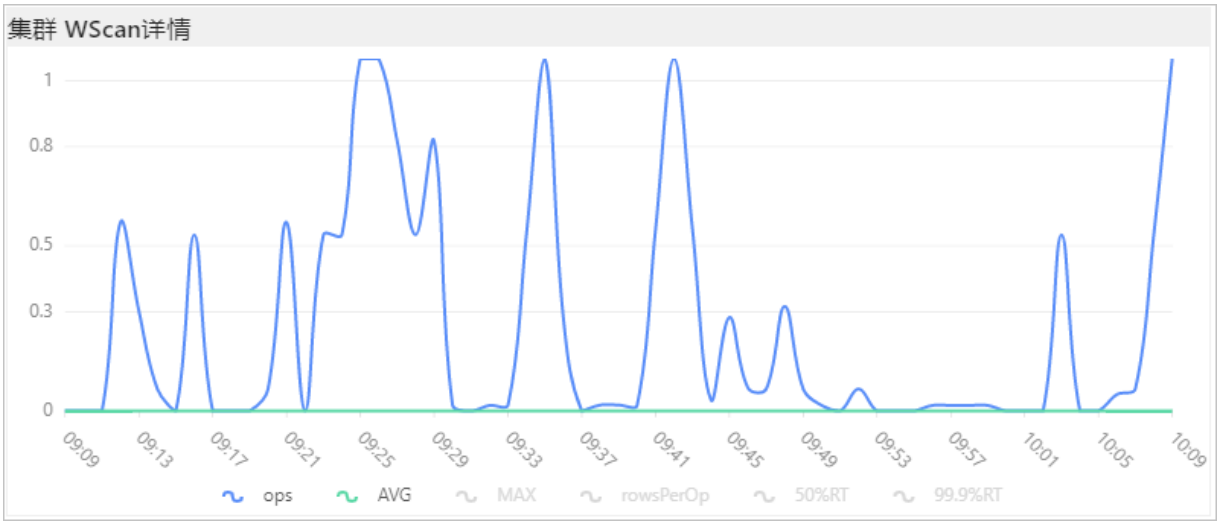
如果您想查看某台机器的具体指标，双击指标曲线，就会弹出机器列表对应的指标数据。



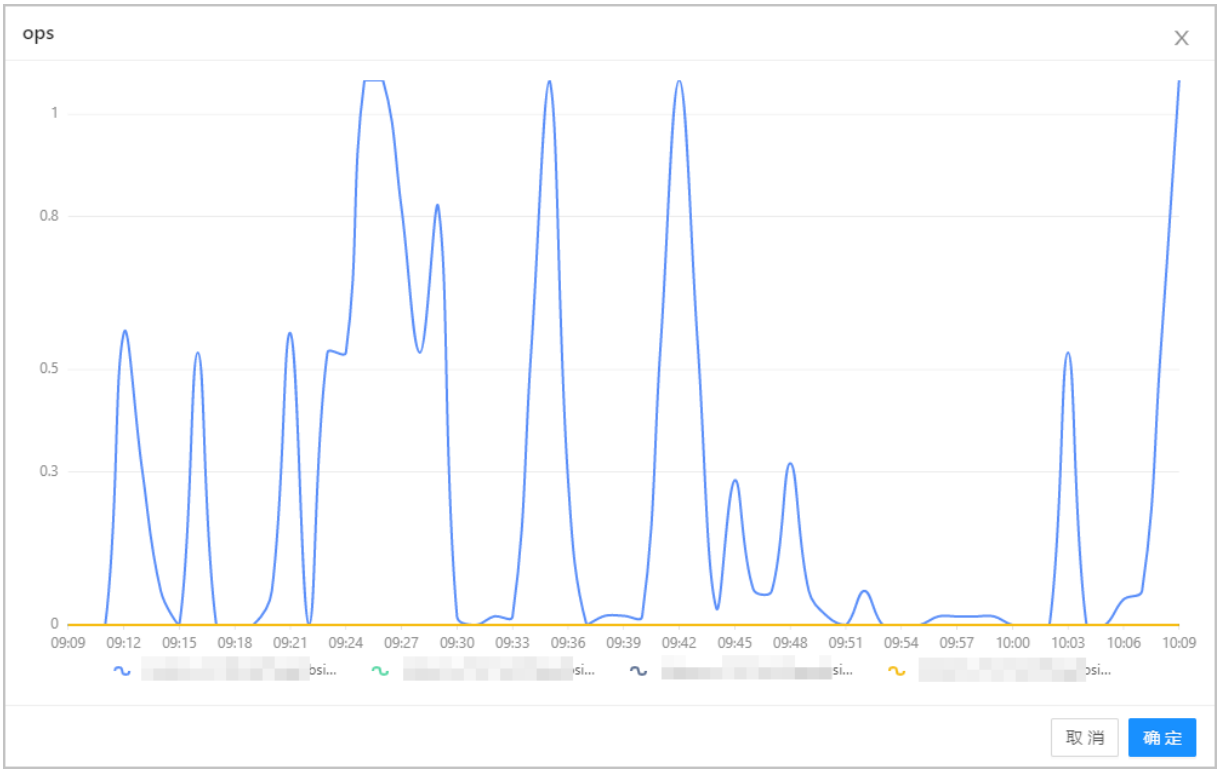
说明 默认显示的是指标值Top10的机器列表，如果机器数过多，您可以手动单击机器名称来显示或者关闭指标曲线。

集群WScan详情

集群WScan详情区域展示了集群执行Scan命令的性能指标数据。



如果您想查看某台机器的具体指标，双击指标曲线，就会弹出机器列表对应的指标数据。



> **说明** 默认显示的是指标值Top10的机器列表，如果机器数过多，您可以手动单击机器名称来显示或者关闭指标曲线。

集群WDelete详情

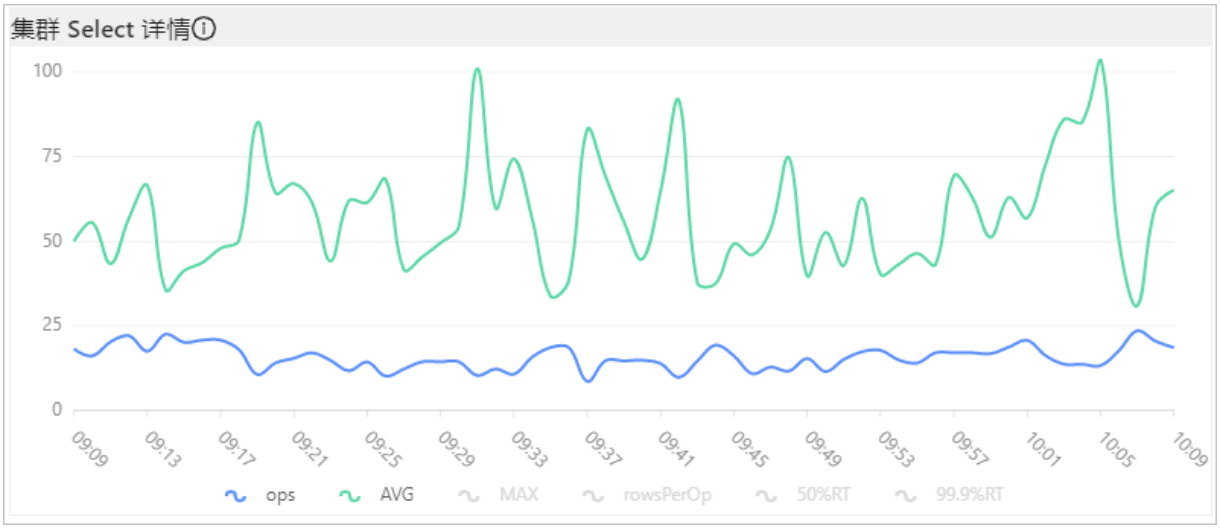
集群WDelete详情区域展示了集群执行Delete命令的性能指标数据。

如果您想查看某台机器的具体指标，双击指标曲线，就会弹出机器列表对应的指标数据。

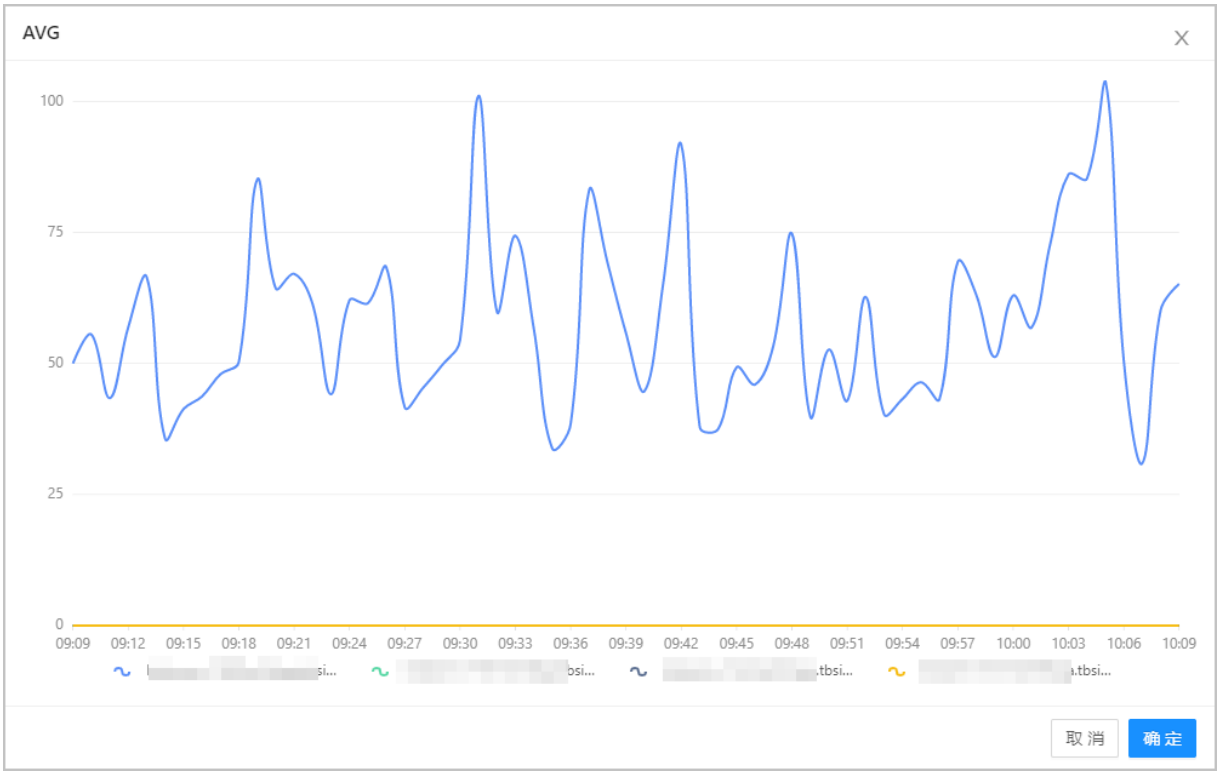
说明 默认显示的是指标值Top10的机器列表，如果机器数过多，您可以手动单击机器名称来显示或者关闭指标曲线。

集群Select详情

集群Select详情区域展示了集群执行Select命令的性能指标数据。



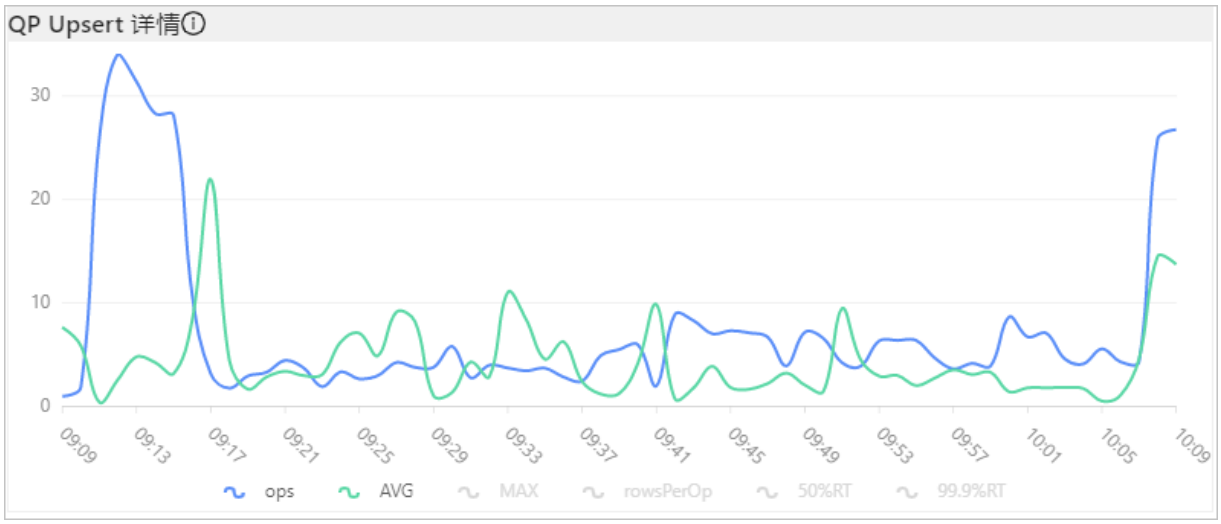
如果您想查看某台机器的具体指标，双击指标曲线，就会弹出机器列表对应的指标数据。



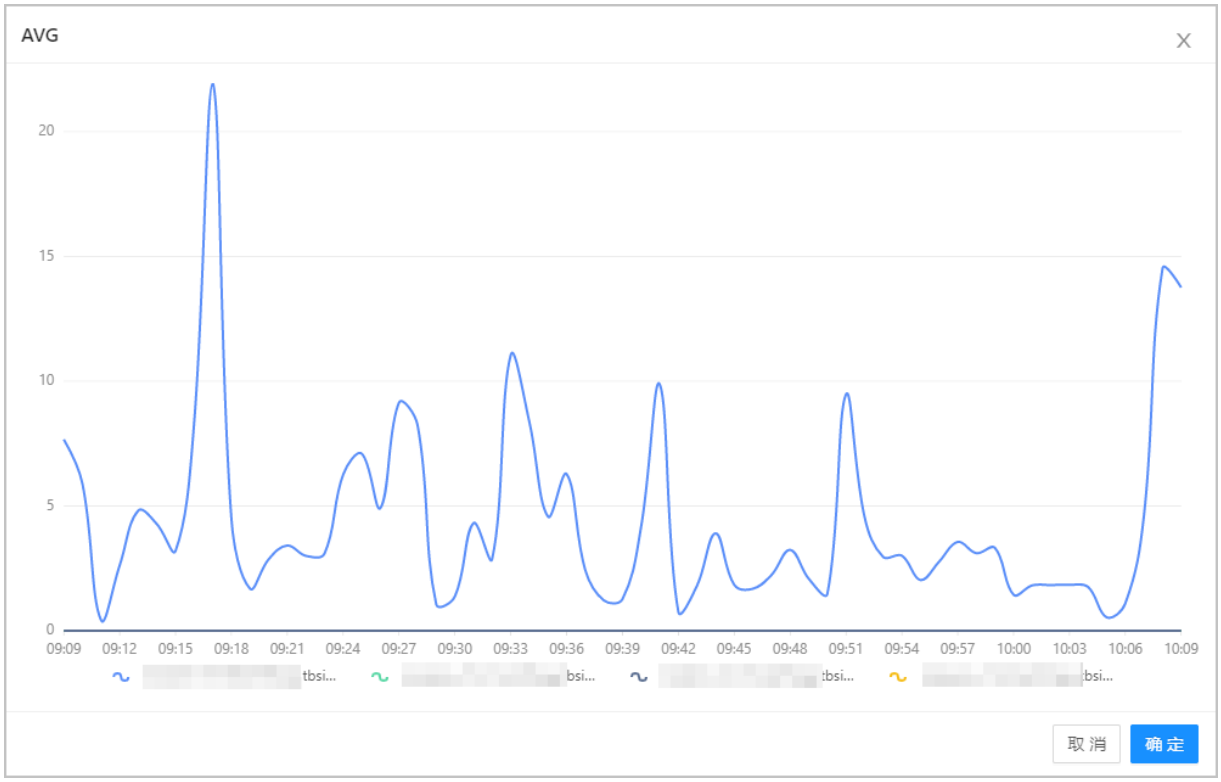
> **说明** 默认显示的是指标值Top10的机器列表，如果机器数过多，您可以手动单击机器名称来显示或者关闭指标曲线。

QP Upsert详情

QP Upsert详情区域展示了集群执行Upsert命令的性能指标数据。



如果您想查看某台机器的具体指标，双击指标曲线，就会弹出机器列表对应的指标数据。



说明 默认显示的是指标值Top10的机器列表，如果机器数过多，您可以手动单击机器名称来显示或者关闭指标曲线。

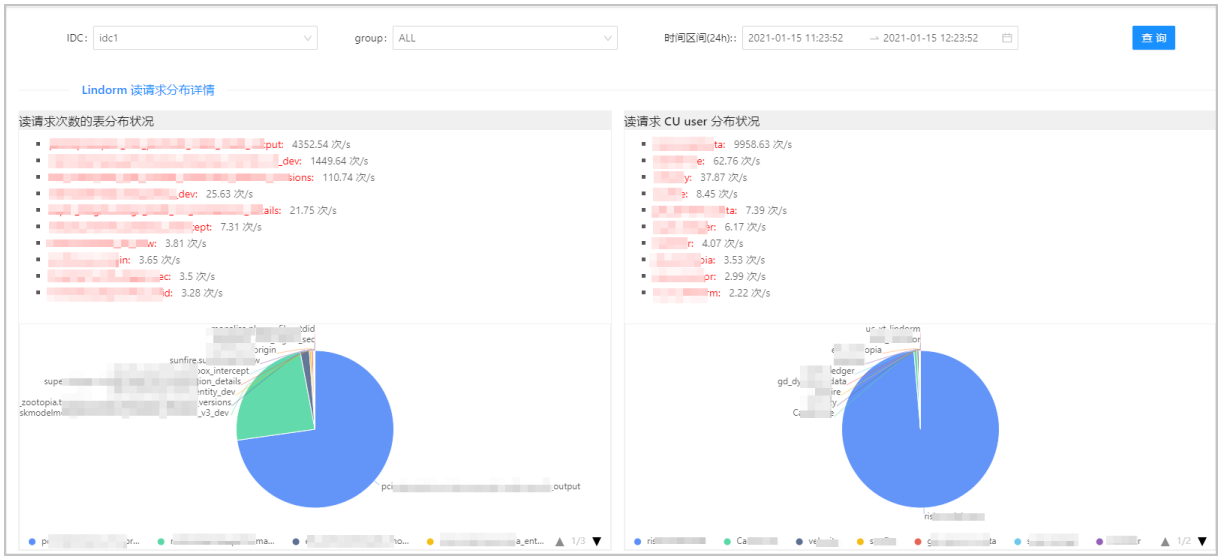
2.4.2. 读写分布详情监控

Lindorm Insight为用户提供了集群相关的性能监控数据，方便用户查看集群的使用情况。读写分布详情监控主要从IDC和group维度监控table读写详情top10。

CU，也叫做计算单元。在Lindorm Insight系统中，一个CU表示扫描了4KB的数据。CU个数越多，表示扫描的数据量越多。这种计算方式主要是用来查看用户或者table的扫描数据量，在Lindorm serverless较常用。

访问读写分布详情页面

进入Lindorm Insight系统，单击左侧导航栏的**监控 > 读写分布详情**即可进入Lindorm Insight**读写分布情况**页面。您可以根据IDC、group、时间区间去筛选查询table读写详情top10。

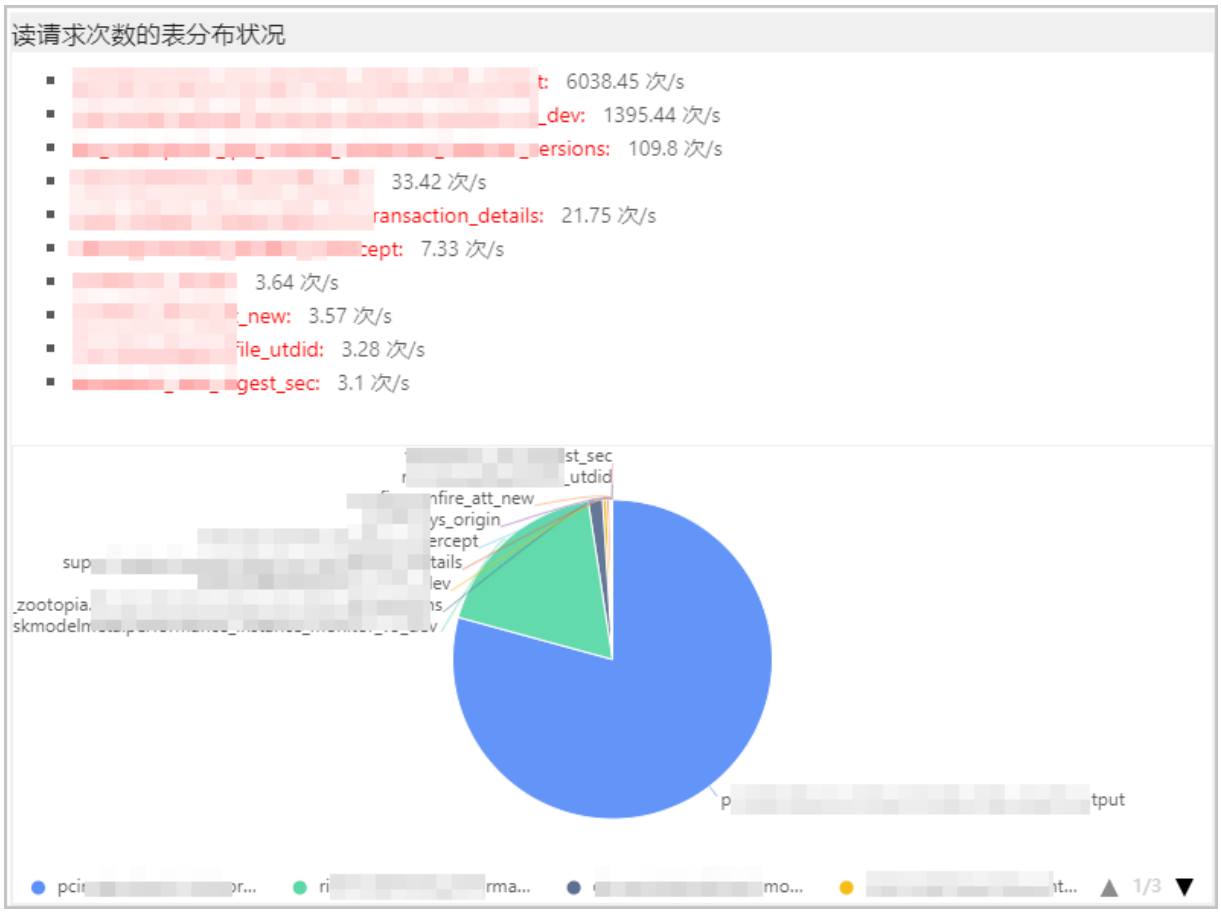


读写分布详情页面分为以下多个区域，您可以单击以下区域链接查看对应的内容。

- 读请求次数的表分布状况
- 读请求CU user分布状况
- 读请求CU table分布状况
- 写请求次数的表分布状况
- 写请求CU user分布状况
- 写请求CU table分布状况

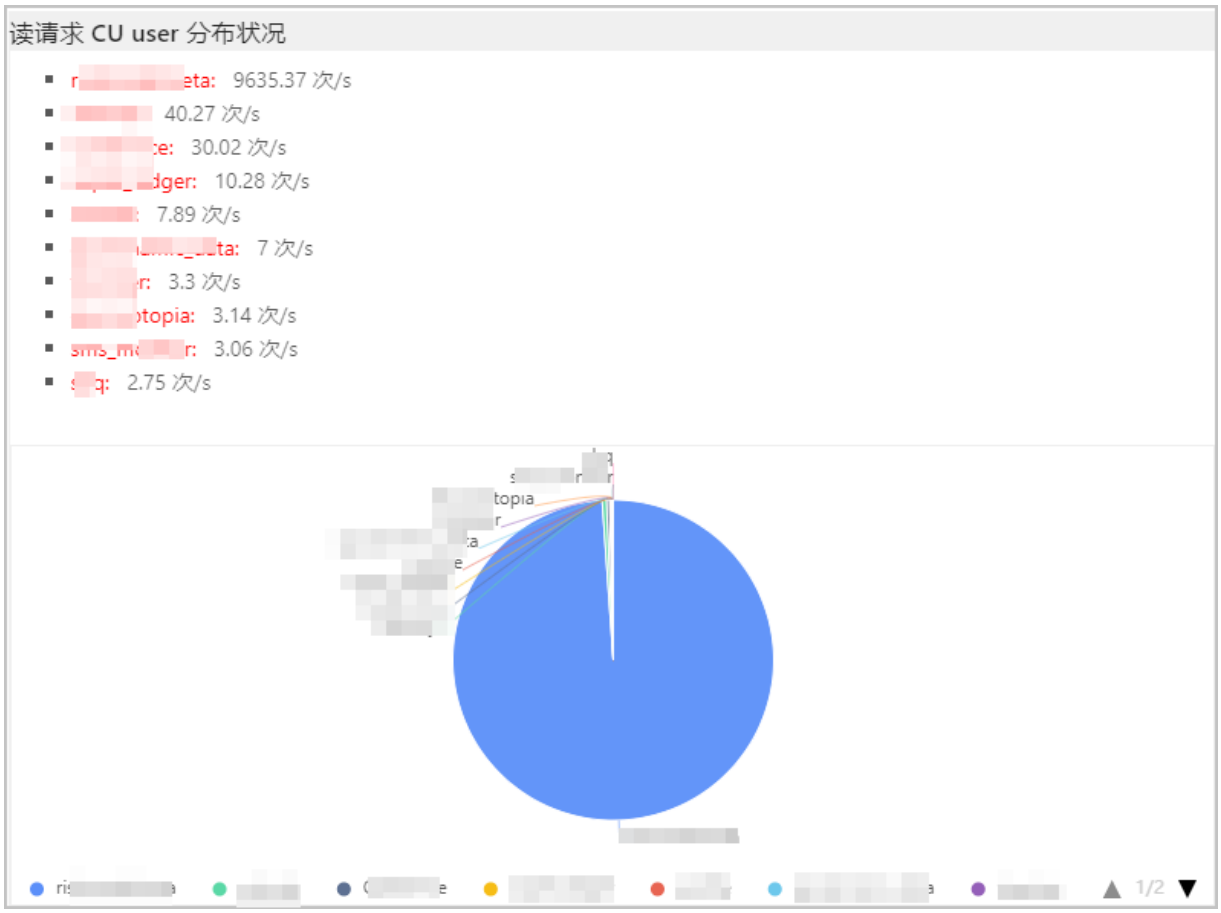
读请求次数的表分布状况

读请求次数的表分布状况区域展示了指定维度下的table查询次数top10。



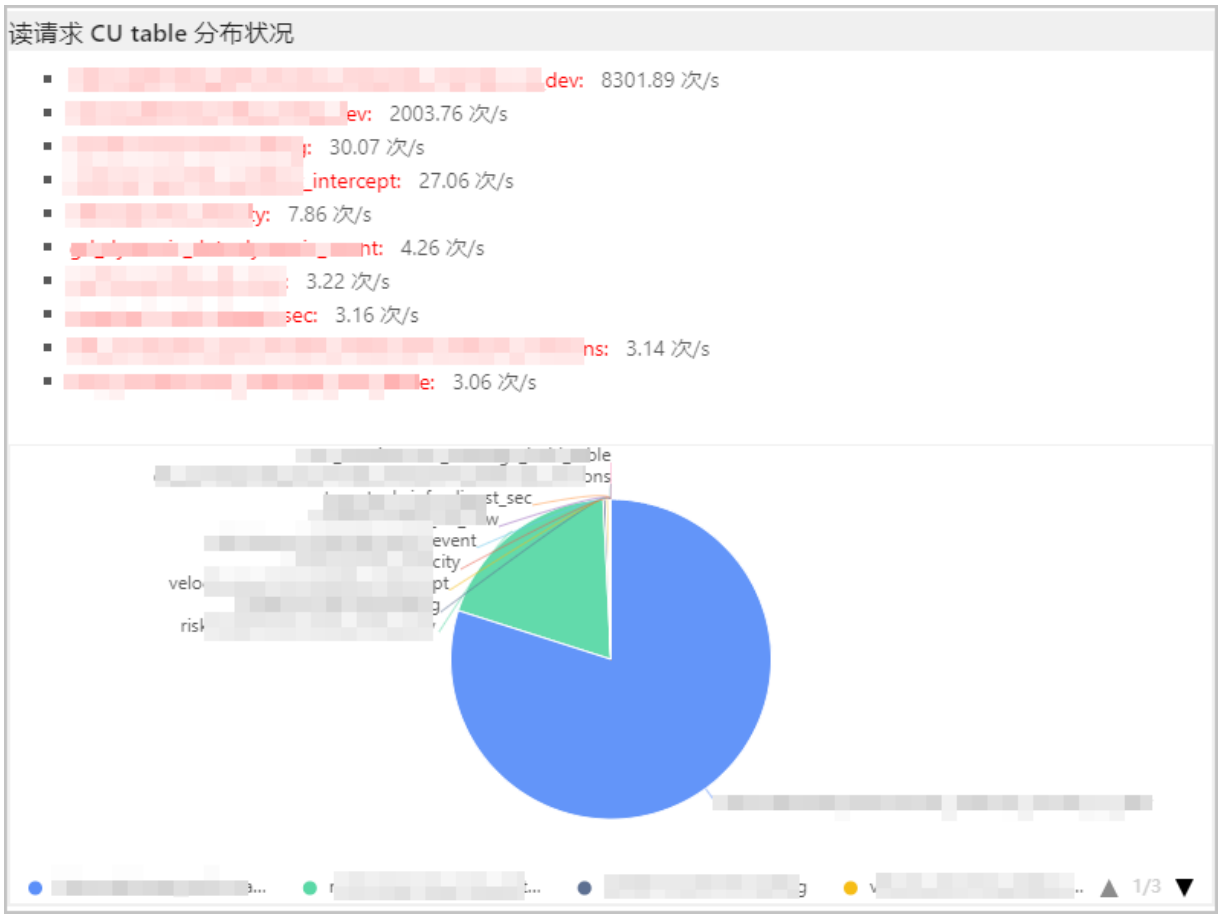
读请求CU user分布状况

读请求CU user分布状况区域展示了读请求中，使用的计算单元 (CU) 数量最多的top10用户。



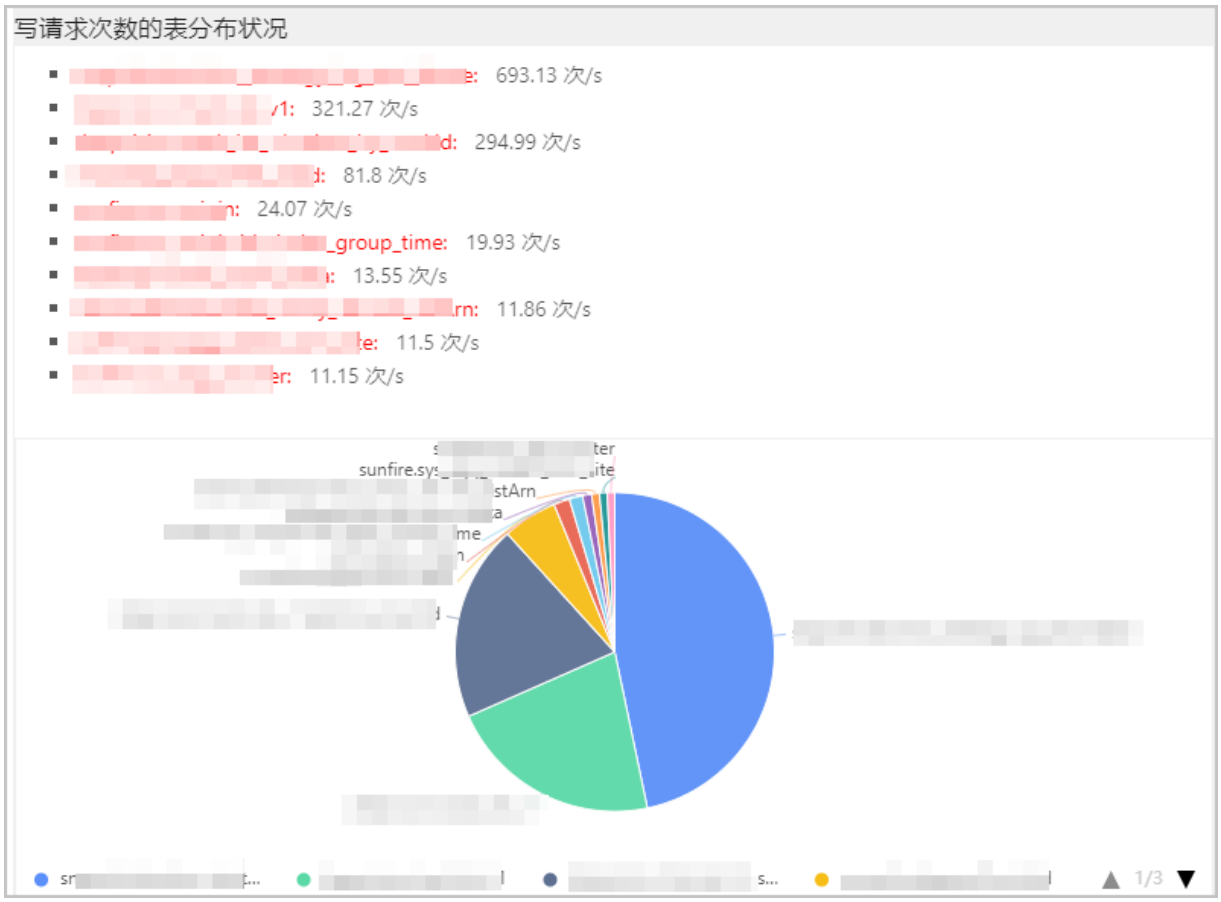
读请求CU table分布状况

读请求CU table分布状况区域展示了读请求中，使用的计算单元 (CU) 数量最多的top10table。



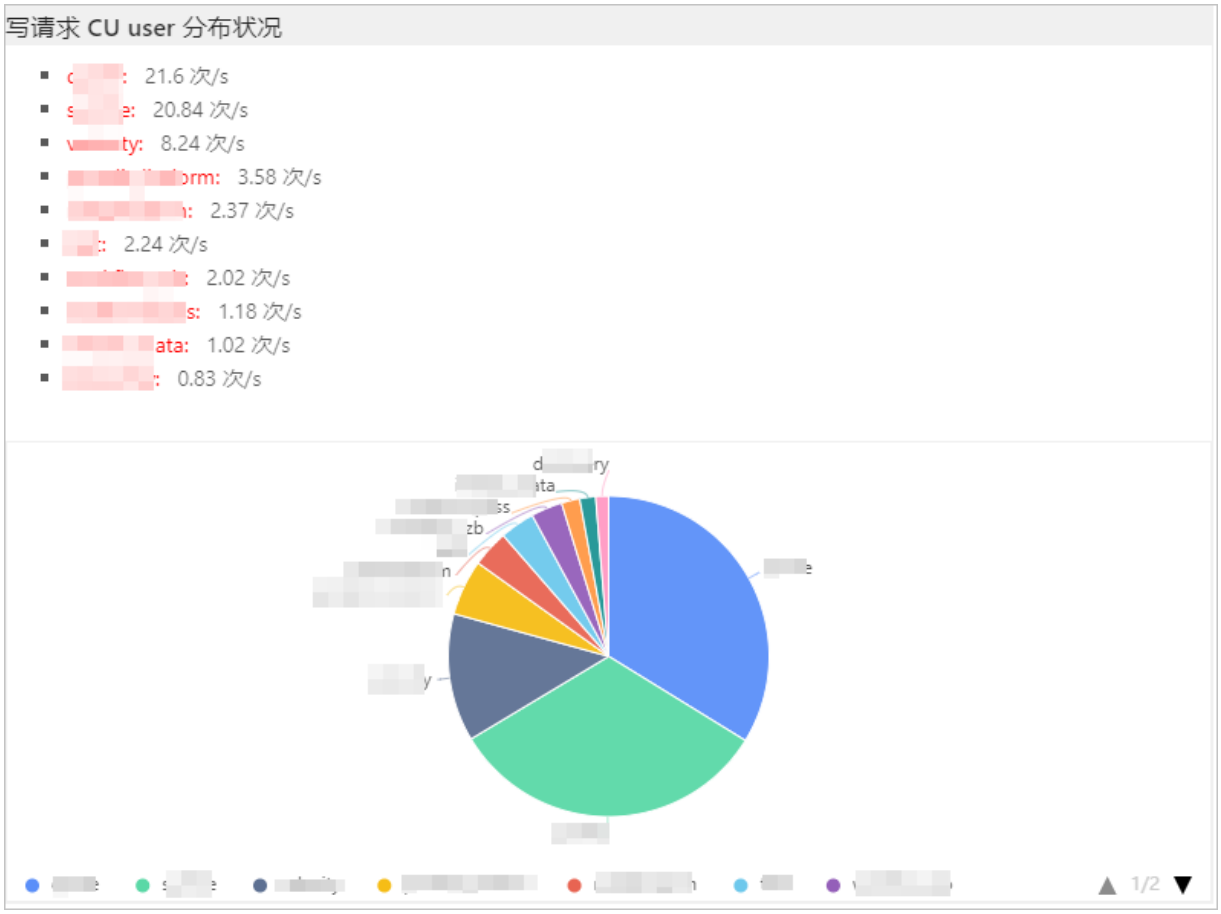
写请求次数的表分布状况

写请求次数的表分布状况区域展示了指定维度下的table写请求次数的top10。



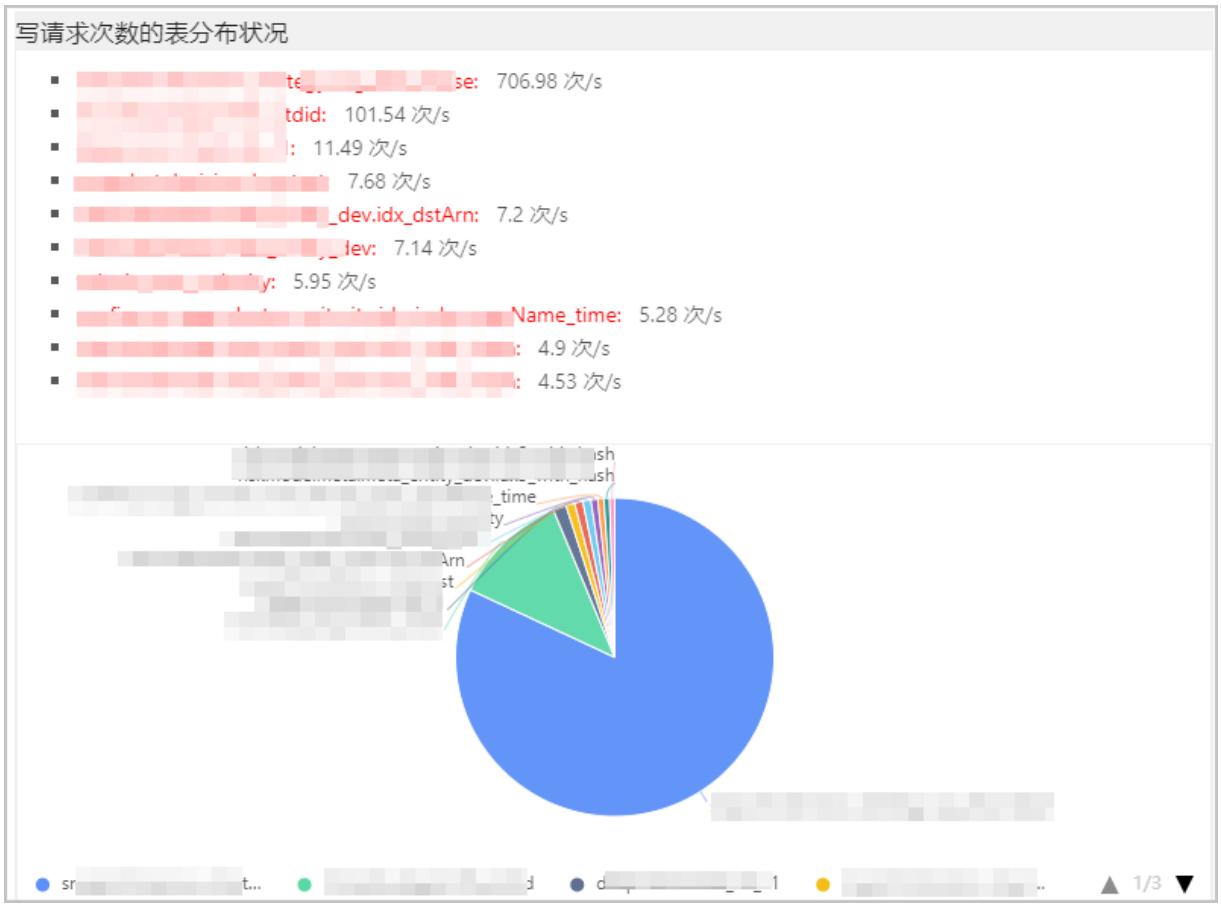
写请求CU user分布状况

写请求CU user分布状况区域展示了写请求中，使用的计算单元 (CU) 数量最多的top10用户。



写请求CU table分布状况

写请求CU table分布状况区域展示了写请求中，使用的计算单元(CU)数量最多的top10 table。

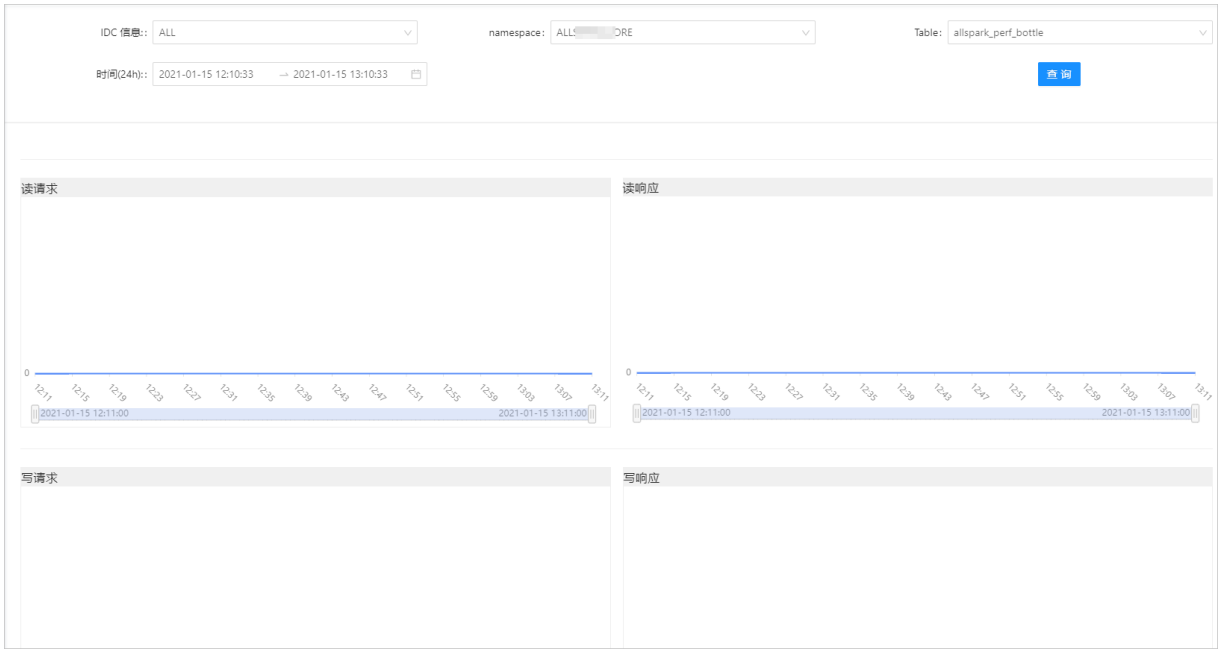


2.4.3. 表监控

Lindorm Insight为用户提供了集群相关的性能监控数据，方便用户查看集群的使用情况。表监控主要监控table的读请求次数、读响应时间、写请求次数、写响应时间详情。

访问表监控页面

进入Lindorm Insight系统，单击左侧导航栏的**监控 > 表监控**即可进入Lindorm Insight**表监控**页面。您可以根据IDC、namespace、Table、时间区间去筛选查询table的各项监控详情。

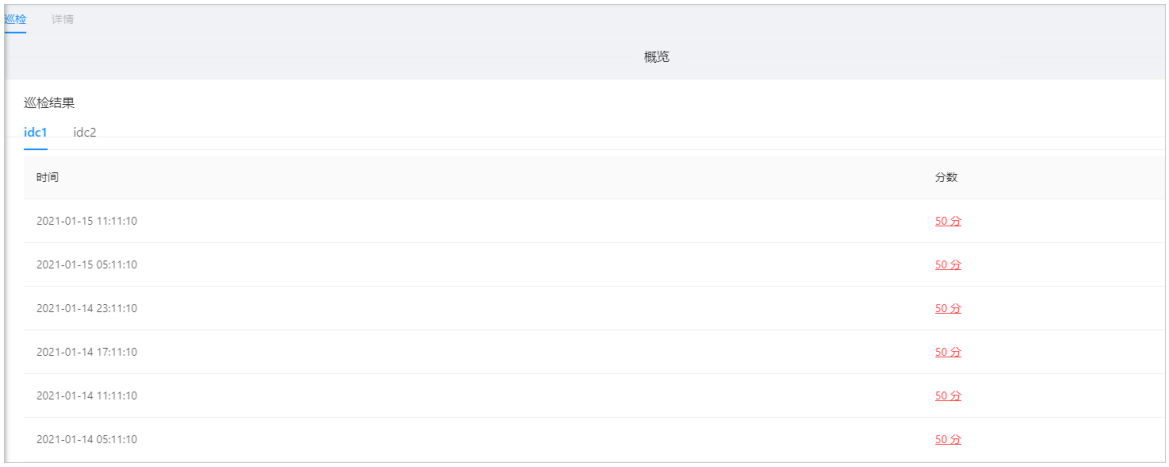


2.5. 巡检

Lindorm Insight提供集群健康巡检能力，对集群节点状态、服务状态以及用户行为进行例行巡检。通过健康巡检，用户能够很方便的监控集群状态以及当前处于亚健康状态的指标。

查看巡检结果

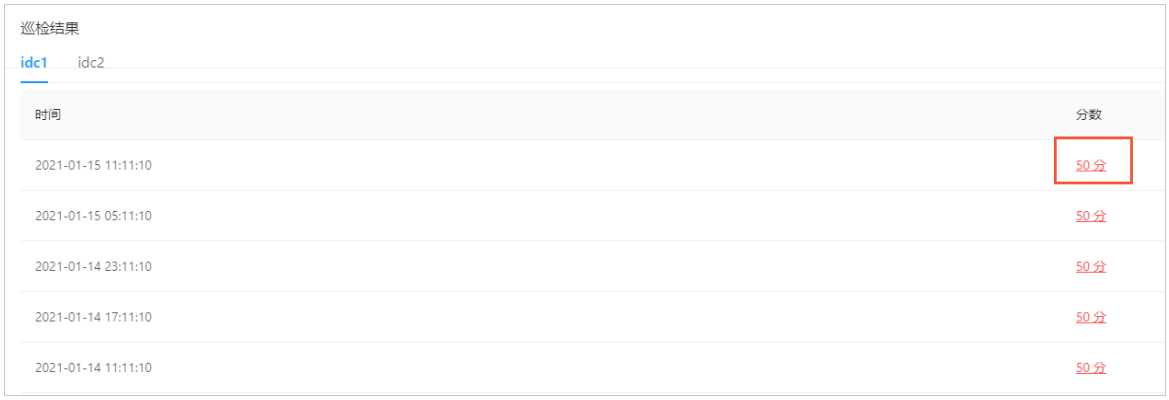
1. 访问Lindorm Insight。具体操作请参考如何访问Lindorm Insight？。
2. 在左侧导航栏，单击**巡检**。
3. 在**巡检**页签，可以查看当前集群各可用区的巡检任务以及巡检打分情况。



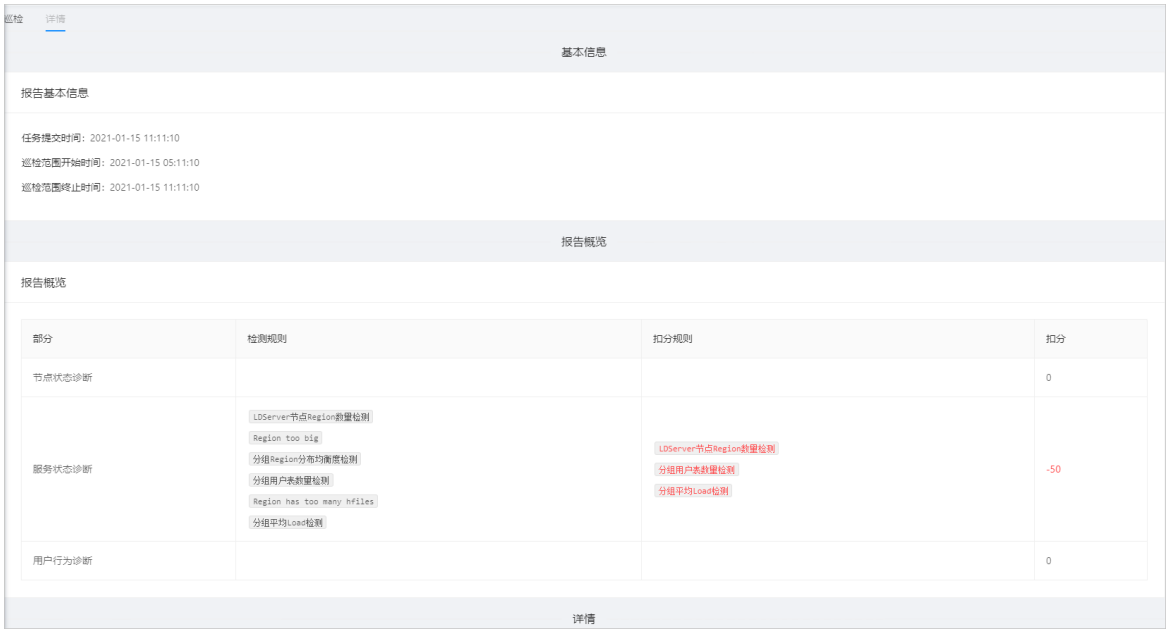
> **说明** 巡检任务每6小时定时触发一次，因此每个触发点均会产出一份巡检报告。

查看巡检报告详情

1. 在**巡检**页签，单击分数链接跳转到**详情**页面。



2. 在**详情**页面，可以查看巡检报告的基本信息（包括任务提交时间、巡检范围开始时间、巡检范围终止时间）、报告概览（包括节点状态、服务状态、用户行为的检测规则、扣分规则已经扣分情况）、报告详情（包括检测异常规则的异常详情信息）。



2.6. 诊断管理

2.6.1. 处理异常快照

本文介绍通过异常回溯工具处理异常事件。

背景信息

在内核处理队列异常场景中，异常回溯工具能够对内核处理请求的线程以及用户请求进行快照，保留线程处理的内容，进而通过对线程处理内容的分析，可初步判断系统卡顿原因和定位请求的具体内容。

> **说明** 异常回溯工具保留现场处理内容快照的同时，也会对服务栈进行快照。

查看异常快照

单击**诊断 > 异常回溯 > 概况**，进入查看异常快照页面。

您在概况页面，可以进行以下操作：

- 查看默认Zone的所有异常采集时间和异常指标。
- 根据组别和节点，筛选查看组别和节点下的异常事件。
- 单击对应节点的**stack快照**，单击**下载**可下载异常时间点的stack文件信息。

分析异常详情

单击异常事件的**详情**按钮，可跳转查看异常事件对应内核所有处理线程的处理请求详情信息，包括所处理的请求以及参数信息。通过详情信息，对业务使用导致的问题，您能快速分析出异常请求源头。

2.6.2. 查看实时热点Region

本文介绍通过“实时TopRegion“诊断工具定位热点Region的问题源。

背景信息

“实时TopRegion“诊断工具能够帮助定位热点Region的问题源。在实际数据库使用过程中，会出现因流量分布不均匀而导致的热点，或者因文件数过多而导致的请求相应延时变长的问题，您都可以使用该诊断工具来定位问题源。

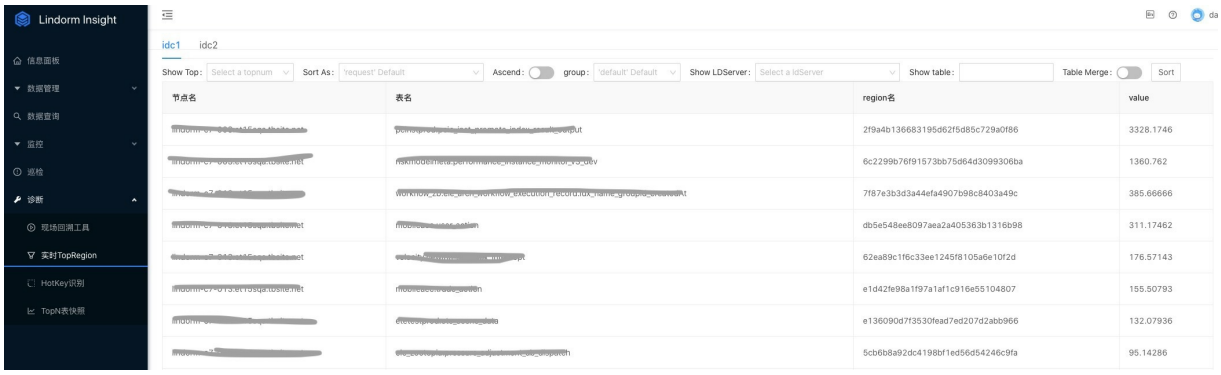
定位并查看实时热点Region

单击**诊断 > 实时TopRegion**，进入热点区域查看页面。

在热点区域查看页面，您可以进行以下操作：

- 整体查看实时的特定分组的热点分布情况，可按指定维度排序。
- 指定节点，查看特定节点的热点分布情况。
- 可开启**Table Merge**，实现表合并，将一张表所关联的Region都合并在一条记录中；再按照特定维度进行排序。

> **说明** 系统默认按“Total Request/Sec“排序。



排序维度说明

“实时TopRegion“诊断工具支持多个排序维度，当前支持的维度如下：

- Total Request/Sec
- Read Request/Sec
- Write Request/Sec

- Region Size (MB)
- File Nums
- Read Response (ms)
- Write Response (ms)
- Memstore Size (MB)
- Total Data Rate (KB/Second)
- Read Data Rate (KB/Second)
- Write Data Rate (KB/Second)

> **说明** 您可根据实际情况选择排序维度。系统默认按“Total Request/Sec“进行排序。

2.6.3. 查看热点Key

本文介绍如何查看系统中的热点Key。

单击**诊断 > HotKey识别**，进入查看热点Key页面。

选择可用区Zone后，您可根据组别或节点，来筛选查看热点Key的分布情况，供开展热点Key问题排查参考。

2.6.4. 表流量分析

本文介绍如何使用表流量分析功能：包括请求量全局Top10表流量分析，数据量全局Top10表流量分析，和单节点请求量Top10所关联表的流量分析。

操作方式

1. 单击**诊断 > 表流量分析**，进入分析页面。
2. 选择可用区Zone之后，您可根据分组和快照时间点，查看特定时间点分组表流量详情记录。

> **说明**
>
> - 当前，TopN指代Top 10。
> - TopN表快照：以筛选请求量top10记录，数据量top 10记录和单节点请求量top 10分片所关联表的记录。若存在疑似热点，会红色高亮显示。
> - 实时Top Region：用于更精准定位实时的热点情况，能够更精准定位问题。具体操作：查看实时热点Region。

三类表介绍

请求量全局Top 10表：该表根据请求量总和（totalRate）进行排序，取流量排名前十的记录进行展示，并呈现核心指标数据。若存在疑似热点region，界面上方会出现红色提醒，并红色高亮疑似热点region所在的记录，方便您快速定位疑似异常表以及相应节点信息。

> **说明** 单击记录头部的“+“，可展开获取每张表的Top 10 region信息，按照与表排序相同的纬度，进行排序。若存在疑似热点region，详情信息会红色高亮显示。

分组Request量全局排序Top10表详情

排名	表名	totalRate (次/s)	totalReadRate (次/s)	totalWriteRate (次/s)	totalWriteRateReplica (次/s)	totalData(KB/s)	totalReadData(KB/s)	totalWriteData(KB/s)	region数	存在疑似热点	疑似热点节点
1	[redacted]	3225.8	3225.8	0.0	0.0	323.95	323.95	0.00	3	正常	-
2	[redacted]	2655.8	2655.7	0.1	0.0	1669.03	1668.99	0.05	1	正常	-
3	[redacted]	1372.9	1372.9	0.0	0.0	604.03	604.03	0.00	8	正常	-
4	[redacted]	516.4	514.0	2.4	2.4	86.03	85.22	0.82	1	正常	-

数据量全局Top 10表：该表根据请求数据量总和（totalData）进行排序，取流量排名前十的记录进行展示，并呈现核心指标数据。若存在疑似热点region，界面上方会出现红色提醒，并红色高亮疑似热点region所在的记录，方便您快速定位疑似异常表以及相应节点信息。

> **说明** 单击记录头部的“+“，可展开获取每张表的Top 10 region信息，按照与表排序相同的纬度，进行排序。若存在疑似热点region，详情信息会红色高亮显示。

分组Data量全局排序Top10表详情

排名	表名	totalRate (次/s)	totalReadRate (次/s)	totalWriteRate (次/s)	totalWriteRateReplica (次/s)	totalData(KB/s)	totalReadData(KB/s)	totalWriteData(KB/s)	region数	存在疑似热点	疑似热点节点
1	[redacted]	2655.8	2655.7	0.1	0.0	1669.03	1668.99	0.05	1	正常	-
2	[redacted]	253.4	0.0	253.4	253.4	1533.74	0.00	1533.74	300	正常	-
3	[redacted]	1372.9	1372.9	0.0	0.0	604.03	604.03	0.00	8	正常	-
4	[redacted]	5.5	5.1	0.4	0.0	396.46	368.03	28.43	8	正常	-

单节点请求量Top 10表：该表根据单节点请求量进行排序，取流量排名前十所关联表的记录进行展示，并呈现核心指标数据。

> **说明** 单节点请求量表单可解决单节点流量高，但该节点所在的整体表的流量未排入前十的情况。

单点Request量Top10 Region关联表详情

排名	表名	totalRate (次/s)	totalReadRate (次/s)	totalWriteRate (次/s)	totalWriteRateReplica (次/s)	totalData(KB/s)	totalReadData(KB/s)	totalWriteData(KB/s)	region数	存在疑似热点	疑似热点节点
1	[redacted]	3225.8	3225.8	0.0	0.0	323.95	323.95	0.00	3	正常	-
2	[redacted]	2655.8	2655.7	0.1	0.0	1669.03	1668.99	0.05	1	正常	-
3	[redacted]	1372.9	1372.9	0.0	0.0	604.03	604.03	0.00	8	正常	-
4	[redacted]	516.4	514.0	2.4	2.4	86.03	85.22	0.82	1	正常	-
5	[redacted]	277.1	274.7	2.4	0.0	40.27	38.97	1.30	256	正常	-