ALIBABA CLOUD

阿里云

实时计算（流计算）
Flink SQL开发指南

文档版本：20220706

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.概述

阿里云实时计算Flink版开发平台为实时计算Flink SQL作业提供了存储管理、作业开发、作业调试、运维管理、监控报警和配置调优功能。

Flink SQL开发指南主要包含以下内容：

- 数据存储

  实时计算Flink版开发平台提供了实时计算Flink版上下游存储设备（例如，云数据库RDS版、数据总线DataHub和表格存储Tablestore等）管理功能。通过存储注册方式引入的上下存储设备，可以实现数据预览、数据抽样以及DDL生成功能。数据存储详情请参见概述。

  > 说明
  > - 如果共享模式集群需访问阿里云VPC网络下的存储资源，请参见VPC访问授权。
  > - 如果实时计算Flink版访问的上下游存储存在白名单机制，请参见数据存储白名单配置。

- 作业开发

  作业开发章节为您介绍Flink SQL作业的开发、上线和启动流程，详情请参见开发、上线和启动。

- 作业调试

  作业调试章节为您介绍Flink SQL作业的调试功能，包括线上调试和本地调试，详情请参见本地调试和线上调试。

- 作业运维

  作业运维章节为您介绍运行信息、数据曲线和Failover等实时计算Flink版作业运维内容，详情请参见运行信息、数据曲线和Failover。

- 监控报警

  监控报警章节为您介绍如何创建和启动报警规则，详情请参见监控报警。

- 作业调优

  作业调优章节为您介绍Flink SQL作业的调优功能，包括高性能Flink SQL优化技巧、AutoConf自动配置调优、AutoScale自动配置调优和手动配置调优，详情请参见高性能Flink SQL优化技巧、AutoConf自动配置调优、AutoScale自动配置调优和手动配置调优。

- Flink SQL

  Flink SQL章节为您介绍Flink SQL语法，详情请参见概述。

2.数据存储

2.1. 概述

阿里云实时计算集成了RDS、DataHub、OTS等数据存储系统的管理界面，为您提供一站式云上数据存储管理服务。

使用限制

独享模式集群仅能访问相同专有网络VPC、相同Region和相同安全组下的存储资源。如果需要访问其它VPC下的资源，请参见如何访问跨VPC里的存储资源？。

数据存储的含义

数据存储有两层含义：

- 实时计算产品上下游生态对应的数据存储系统/数据库表（以下简称为存储资源，详情请参见产品生态）。
- 实时计算产品对上下游存储资源的管理功能（以下简称为数据存储功能）。

> **说明** 使用注册存储功能前，请先完成实时计算对存储设备的访问授权，授权方法请您参见共享模式角色授权和独享模式角色授权。

实时计算产品支持2种引用上下游存储资源的方式：明文方式和存储注册方式。

明文方式

明文方式是通过在作业的DDL语句WITH参数中配置 `accessId` 和 `accessKey` ，来引用上下游存储资源，详情请参见概述。明文方式不仅支持同账号（包括其RAM用户）授权，同时还支持跨账号授权。例如，当前实时计算A用户（包括A下所属的RAM用户）需要使用用户B的存储资源，可以通过如下明文方式定义DDL。

```
CREATE TABLE in_stream(
  a varchar,
  b varchar,
  c timestamp
) with (
  type='datahub',
  endPoint='http://dh-cn-hangzhou.aliyuncs.com',
  project='<dataHubProjectName>',
  topic='<dataHubTopicName>',
  accessId='<accessIdOfUserB>',
  accessKey='<accessKeyOfUserB>'
);
```

存储注册方式

存储注册方式是将上下游存储资源预先注册至实时计算开发平台，然后通过实时计算控制台的数据存储管理功能，对上下游存储资源进行引用。使用存储注册方式后，实时计算控制台能够为您提供数据预览、数据抽样、DDL自动生成等功能，便于您一站式管理云上存储资源。

> **说明** 实时计算数据存储功能当前仅支持同账号属主下的存储资源，即当前使用实时计算的A用户（包括A用户的子账户）所注册的存储资源，必须是A购买的存储资源。不支持跨账号授权，如果您需要跨账号授权使用存储资源，请使用明文方式。

- 注册数据存储

  使用存储注册方式需要将上下游存储资源预先注册至实时计算开发平台。注册步骤如下：

  i. 登录实时计算控制台。
  ii. 单击页面顶部的**开发**。
  iii. 在**开发**界面，单击左侧导航栏中的**数据存储**。
  iv. 在**数据存储**页签的右上角，单击**+注册与网络**。
  v. 在**注册数据存储与网络探测**窗口，完成对应储存设备的参数配置。

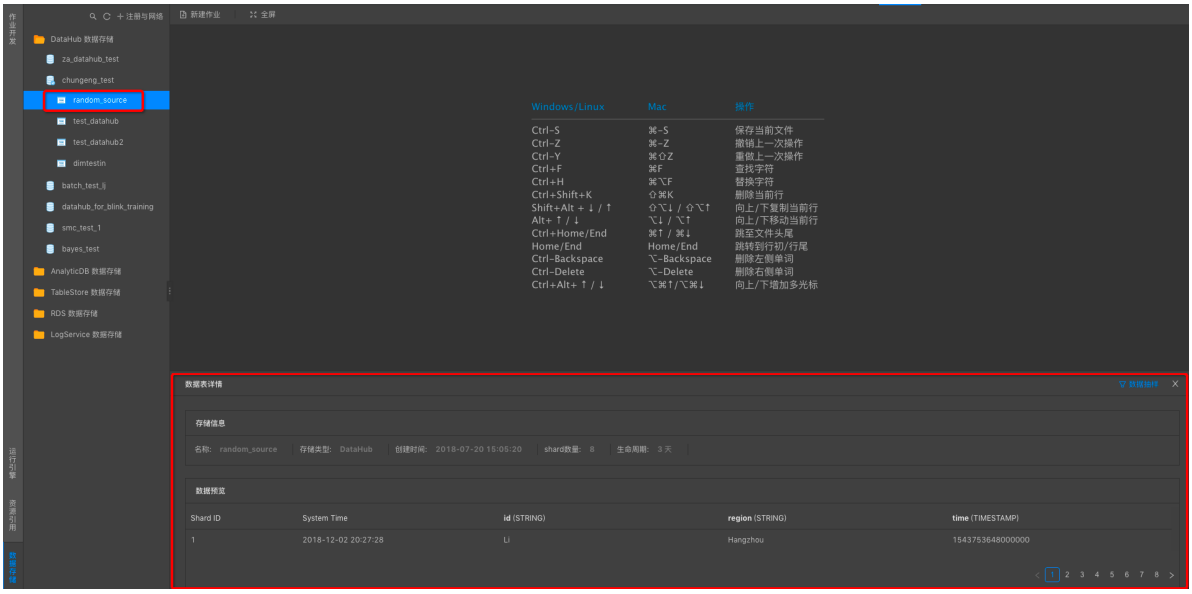
  目前实时计算产品仅支持注册如下五种存储资源，具体方法请点击以下产品链接：

  - 注册数据总线DataHub
  - 注册云原生数据仓库AnalyticDB MySQL版
  - 注册表格存储Tablestore
  - 注册云数据库RDS版
  - 注册日志服务SLS

- 数据预览

  对于已经注册的存储资源，实时计算提供数据预览功能，功能实现步骤如下：

  i. 在**开发**界面，单击左侧导航栏底部的**数据存储**。
  ii. 在**数据存储**区域，双击数据存储类型文件夹以及文件夹下的各节点，直至目标数据表。
  iii. 在**数据表详情 > 数据预览**，查看存储资源的数据。



- 自动生成DDL

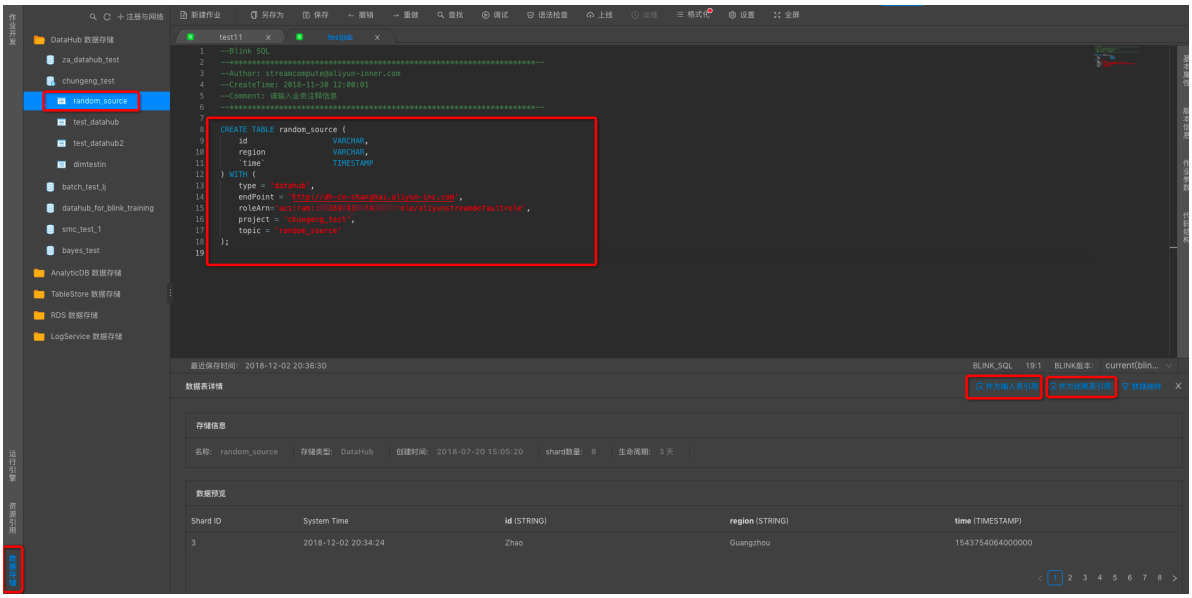
  对于已经注册的存储资源，实时计算提供自动生成DDL的功能，功能实现步骤如下：

  i. 在**开发**界面，单击左侧导航栏底部的**数据存储**。

ii. 在**数据存储**区域，双击数据存储类型文件夹以及文件夹下的各节点，直至目标数据表。

iii. 在**数据表详情**，单击**作为输入表引用**、**作为结果表引用**或**作为维表引用**，即可自动生成DDL。

> **说明** 自动生成的DDL仅包含基本的WITH参数，以保证实时计算与存储资源的连通性。您可在此基础上增加其他WITH参数。



- 网络探测

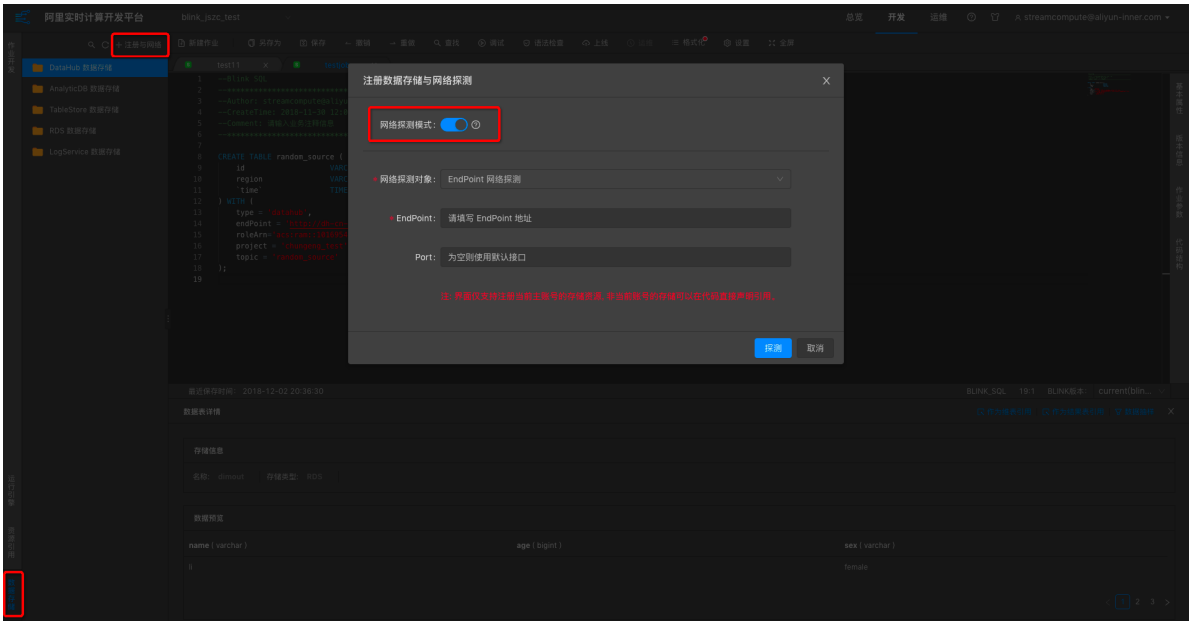
> **说明** 金融云杭州地域未安装云助手，网络探测功能暂时不支持。

实时计算的数据存储功能提供网络探测功能，用于探测实时计算产品与被探测的存储资源的网络连通性。网络探测功能开启方式如下：

i. 在**开发**界面，单击左侧导航栏底部的**数据存储**。

ii. 在**数据存储**页签的右上角，单击**+注册与网络**。

iii. 在**注册数据存储与网络探测**页面，打开**网络探测模式**开关。



2.2. 注册数据存储

2.2.1. 注册数据总线DataHub

本文为您介绍注册数据总线DataHub所需的参数信息。

什么是数据总线

阿里云流数据处理平台DataHub是流式数据（Streaming Data）的处理平台，提供对流式数据的发布（Publish）、订阅（Subscribe）和分发功能，让您可以轻松构建基于流式数据的分析和应用。实时计算Flink版使用DataHub作为流式数据存储源头或输出目的端。

注册存储

> **说明** 使用注册存储功能前，请先完成实时计算Flink版对存储设备的访问授权，授权方法请您参见共享模式角色授权和独享模式角色授权。

1. 登录实时计算控制台。
2. 在页面顶部，单击**开发**。
3. 在左侧导航栏，单击**数据存储**。
4. 在页面左上角，单击**注册与网络**。
5. 在**注册数据存储与网络探测**对话框，配置存储设备参数。
6. 单击**注册**。

参数说明

- EndPoint

  填写DataHub的EndPoint。不同地域DataHub有不同的EndPoint，具体请参见DataHub域名列表。

  > **说明** EndPoint地址格式为 `http://****.aliyun-inc.com`，不能以 `/` 结尾。

- 独享模式：使用DataHub VPC EndPoint。例如，**华东1（杭州）**使用 `http://dh-cn-hangzhou-int-vpc.aliyuncs.com` 。

  > **说明** 独享模式集群所在的区域与DataHub所在区域要求一致。

- 共享模式（已停购）：使用经典网络ECS EndPoint。例如，**华东1（杭州）**使用 `http://dh-cn-hangzhou.aliyun-inc.com` 。

  > **说明** 共享模式项目所在的区域与DataHub所在区域不要求一致，但不一致可能增加网络延时。

- Project

  填写DataHub的Project。

  > **说明** 实时计算Flink版暂不支持跨属主的数据存储注册。例如，A用户拥有DataHub的ProjectA，但B用户希望在实时计算Flink版使用ProjectA。如果您需要使用跨属主的数据存储，可使用明文方式，详情请参见明文方式。

常见问题

Q：注册数据存储失败的原因有哪些？

A：实时计算Flink版的数据存储页面能够协助您完成数据管理，注册数据存储功能是使用相关存储SDK访问各类存储的。如果注册数据存储失败，请从以下方面进行排查：

- 是否已经开通并拥有DataHub的Project。请登录DataHub控制台，查看您是否具备访问对应Project的权限。
- 您是否为DataHub Project的属主。跨属主的数据存储不能注册。
- 您填写的DataHub的EndPoint和Project是否完全正确。DataHub的EndPoint必须以 `http` 开头，且不能以 `/` 结尾。
- 您填写的DataHub EndPoint必须为经典网络地址，而非VPC地址。实时计算Flink版暂不支持VPC内部地址。
- 请不要重复注册，实时计算Flink版提供注册检测机制，重复注册会导致注册失败。

Q：为什么数据抽样仅支持时间抽样？

A：DataHub定位是流数据存储，对外提供的接口仅有时间参数。因此，实时计算Flink版仅提供基于时间的抽样。

2.2.2. 注册云原生数据仓库AnalyticDB MySQL版

本文为您介绍注册云原生数据仓库AnalyticDB MySQL版数据存储所需的参数信息。

> **注意** 本文档仅适用于云原生数据仓库AnalyticDB MySQL版 2.0版本。云原生数据仓库AnalyticDB MySQL版 3.0版本暂不支持结果表注册存储功能，请使用明文方式进行注册，详情请参见创建云原生数据仓库AnalyticDB MySQL版3.0结果表。

注册存储

> **说明** 使用注册存储功能前，请先完成实时计算Flink版对存储设备的访问授权，授权方法请您参见共享模式角色授权和独享模式角色授权。

1. 登录实时计算控制台。
2. 在页面顶部，单击**开发**。
3. 在左侧导航栏，单击**数据存储**。
4. 在页面左上角，单击**注册与网络**。
5. 在**注册数据存储与网络探测**对话框，配置存储设备参数。
6. 单击**注册**。

参数说明

参数名称	说明
URL	云原生数据仓库AnalyticDB MySQL版控制台链接信息。分为以下两种情况： • 共享模式：请使用云原生数据仓库AnalyticDB MySQL版的经典网络URL。 • 独享模式：请使用云原生数据仓库AnalyticDB MySQL版的VPC网络URL。 URL地址查询步骤如下： 1. 登录AnalyticDB 控制台。 2. 单击对应的集群名称，进入**基本信息**页面。 3. 在**网络信息**中查看相应的连接地址。
Database	云原生数据仓库AnalyticDB MySQL版数据库名称，即云原生数据仓库AnalyticDB MySQL版实例名称。
AccessKey ID	阿里云账号的AccessKey ID。
AccessKey Secret	阿里云账号的AccessKey Secret。

> **说明** AccessKey ID和AccessKey Secret信息查询方法请参见如何查看AccessKey ID和AccessKey Secret信息？。

2.2.3. 注册表格存储Tablestore

本文为您介绍注册表格存储Tablestore所需的参数信息。

什么是表格存储Tablestore

表格存储Tablestore是构建在阿里云飞天分布式系统之上的NoSQL数据存储服务。表格存储能够提供海量结构化数据的存储服务和实时访问服务。Tablestore具备低延迟和运算复杂度低的特点，因此适合作为实时计算Flink版的数据存储维表或结果表。

注册存储

> **说明** 使用注册存储功能前，请先完成实时计算Flink版对存储设备的访问授权，授权方法请您参见共享模式角色授权和独享模式角色授权。

1. 登录实时计算控制台。
2. 在页面顶部，单击**开发**。
3. 在左侧导航栏，单击**数据存储**。
4. 在页面左上角，单击**注册与网络**。
5. 在**注册数据存储与网络探测**对话框，配置存储设备参数。
6. 单击**注册**。

参数说明

- Endpoint
  - 填写Tablestore的Endpoint。请在表格存储控制台查看Tablestore的Endpoint信息，填写Tablestore私网地址。具体步骤请参见私网地址。
  - Tablestore访问网络类型设置为**允许任意网络访问**。操作步骤如下：
    a. 登录表格存储控制台。
    b. 单击目标**实例名称**。
    c. 在**网络管理**页签，单击**更改**。
    d. 选择**允许任意网络访问**。
    e. 单击**确定**。
- 实例名称

  填写实例名称。

2.2.4. 注册云数据库RDS版

本文为您介绍注册云数据库RDS版数据存储所需的参数信息，以及连接过程中的常见问题。

什么是云数据库RDS版

阿里云关系型数据库（Relational Database Service，RDS）是一种稳定可靠、可弹性伸缩的在线数据库服务。RDS基于阿里云分布式文件系统和高性能存储，支持MySQL、SQL Server、PostgreSQL和PPAS（Postgres Plus Advanced Server）引擎，并且提供了容灾、备份、恢复、监控和迁移等方面的全套解决方案。

> **说明**
> - 高频或高并发写入场景，不建议使用RDS作为实时计算Flink版作业的结果表，存在死锁风险。建议使用表格存储作为结果表（详情请参见创建表格存储Tablestore结果表）。
> - 云数据库RDS 8.0版本不支持注册存储功能，请使用明文方式。

注册存储

> **说明** 使用注册存储功能前，请先完成实时计算Flink版对存储设备的访问授权，授权方法请您参见共享模式角色授权和独享模式角色授权。

1. 登录实时计算控制台。
2. 在页面顶部，单击**开发**。
3. 在左侧导航栏，单击**数据存储**。
4. 在页面左上角，单击**注册与网络**。
5. 在**注册数据存储与网络探测**对话框，配置存储设备参数。
6. 单击**注册**。

参数说明

> **说明** 存储注册过程中系统会为RDS自动配置IP白名单。

参数	说明
数据存储类型	选择**RDS 数据存储**。
Region	选择RDS所在的地域。
Instance	填写RDS实例ID。 **说明** 请填写实例ID，不是实例名称。
DBName	填写RDS中Database的名称。 **说明** Database是RDS的数据库名称，不是实例名称。
User Name	登录数据库的用户名。
Password	登录数据库的密码。

常见问题

Q：存储注册过程中出现**操作错误：未在VPC中授权**报错如何处理？

A：注册的RDS实例为专有网络（VPC）实例。对于VPC实例，请先完成实时计算Flink版对VPC的访问授权，授权步骤请参见VPC访问授权。

2.2.5. 注册日志服务SLS

本文为您介绍注册日志服务SLS数据存储所需的参数信息，以及存储注册过程中的常见问题。

什么是日志服务SLS

日志服务SLS是针对日志类数据的一站式服务。日志服务可以帮助您快捷地完成数据采集、消费、投递以及查询分析，提升运维和运营效率，建立海量日志处理能力。日志服务本身是流数据存储，实时计算Flink版能将其作为流式数据的输入。

注册存储

> **说明** 使用注册存储功能前，请先完成实时计算Flink版对存储设备的访问授权，授权方法请您参见共享模式角色授权和独享模式角色授权。

1. 登录实时计算控制台。
2. 在页面顶部，单击**开发**。
3. 在左侧导航栏，单击**数据存储**。
4. 在页面左上角，单击**注册与网络**。
5. 在**注册数据存储与网络探测**对话框，配置存储设备参数。
6. 单击**注册**。

参数说明

- Endpoint

  填写日志服务的Endpoint。不同地域的日志服务Endpoint不同，详情请参见服务入口。

  > **说明**
  > - Endpoint必须以 `http://` 开头，且不能以 `/` 结尾，例如 `http://cn-hangzhou-intranet.log.aliyuncs.com` 。
  > - 实时计算Flink版和日志服务均处于阿里云内网，建议您填写经典网络或VPC网络Endpoint。不建议填写公网Endpoint，填写公网Endpoint可能导致性能问题和外网宽带的消耗。

- Project

  填写Project名称。

  > **说明** 实时计算Flink版暂不支持跨属主的数据存储注册。例如，A用户拥有日志服务的Project A，但B用户希望在实时计算Flink版使用Project A。如果您需要使用跨属主的数据存储，可以使用明文方式，详情请参见明文方式。

常见问题

Q：注册数据存储失败的原因有哪些？

A：实时计算Flink版的数据存储页面能够协助您完成数据管理，注册数据存储功能使用相关存储的SDK代为访问各类存储。如果注册数据存储失败，请从以下方面进行排查：

- 是否已开通并拥有SLS的Project。请登录日志服务控制台，查看您是否具备访问对应Project的权限。
- 您是否为日志服务Project的属主。跨属主的数据存储不能注册。
- 您填写的日志服务的Endpoint和Project是否完全正确。日志服务的Endpoint必须以 `http://` 开头，且不能以 `/` 结尾。
- 您填写的日志服务的Endpoint必须为经典网络地址，而非VPC地址。实时计算Flink版暂不支持VPC内部地址。

- 请不要重复注册，实时计算Flink版提供注册检测机制，重复注册会导致注册失败。

Q：为什么数据抽样仅支持时间抽样？

A：日志服务是流数据存储，对外提供的接口也仅有时间参数。因此，实时计算Flink版也仅能提供基于时间的抽样。

2.3. VPC访问授权

实时计算共享模式访问阿里云专有网络（VPC）中的存储资源前，需要进行VPC访问授权。本文为您介绍VPC访问授权的流程。

背景

实时计算共享模式属于阿里云经典网络，如果需要访问阿里云VPC中的存储资源（目前仅支持VPC授权RDS），则需要完成VPC访问授权。

> **说明**
> - 独享模式集群处于阿里云VPC，无需进行VPC访问授权。
> - 完成VPC访问授权后，实时计算访问对应的存储资源可能存在带宽受限等性能问题。不建议在实时计算共享模式下访问VPC中的存储资源。
> - 实时计算共享模式已于2019年12月24日正式下线，不再支持共享模式新项目的购买，仅支持原有项目的扩缩容、续费操作。如果有新购需求，推荐使用实时计算独享模式或Flink云原生模式。

VPC访问授权步骤

1. 登录实时计算控制台。
2. 将鼠标悬停至页面右上角账号名称。
3. 在下拉菜单中，单击**项目管理**。
4. 在左侧导航栏中，单击**VPC访问授权**。
5. 在**VPC访问授权**页面的右上角，单击**新增授权**。
6. 在**授权流计算访问VPC**页面，输入相应的配置参数。

参数	说明
名称	VPC的名称。
地域	VPC中存储设备所在的地域。
VPC ID	VPC中存储设备的VPC网络ID。RDS VPC网络ID查看步骤如下： i. 登录RDS管理控制台。 ii. 在左侧导航栏，单击**实例列表**。 iii. 在页面左上角，选择实例所在地域。 iv. 单击目标实例ID。 v. 单击左侧导航栏中的**数据库连接**。 vi. 在**数据库连接**页面中，查看**网络类型**信息。 例如，RDS的VPC ID为 `vpc-bp1lysht98wrvl9n3****` 。

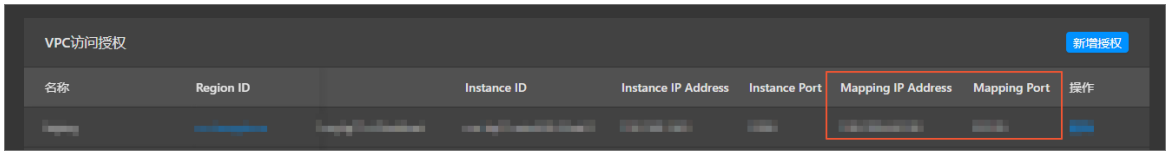
参数	说明
Instance ID	VPC中存储设备的实例ID。 RDS中实例ID查看步骤如下： i. 登录RDS管理控制台。 ii. 在左侧导航栏，单击**实例列表**。 iii. 在页面左上角，选择实例所在地域。 iv. 单击目标实例ID，进入**基本信息**页面。 v. 在**基本信息**页面，查看RDS实例的ID。
Instance Port	VPC中存储设备的端口ID。RDS端口查询方法请参见查看或修改内外网地址和端口。

常见问题

Q：明文方式中，如何添加专有网络存储设备的URL参数？

A：在使用明文方式引用VPC中的存储时，DDL WITH参数中的URL参数值需填写**VPC访问授权**页面中的**Mapping IP Address**和**Mapping Port**参数，例如，url='jdbc:mysql://<mappingIP>:<mappingPort>/<databaseName>'。**Mapping IP Address**和**Mapping Port**信息查看步骤如下：

1. 登录实时计算控制台。
2. 将鼠标悬停至页面右上角账号名称。
3. 在下拉菜单中，单击**项目管理**。
4. 在左侧导航栏中，单击**VPC访问授权**。
5. 在**VPC访问授权**页面，查看Mapping IP Address和Mapping Port信息。



2.4. 数据存储白名单配置

新建的数据库通常默认拒绝外部设备的访问，只有配置在数据存储白名单中的IP地址才被允许访问。本文以RDS为例，为您介绍如何配置数据存储白名单。

实时计算IP地址

实时计算分为共享模式和独享模式，2种模式的IP地址有所不同。

- **共享模式IP地址**

- 访问经典网络下的数据存储

  可根据项目所在的区域，配置对应区域的IP地址。

Region	白名单
华东2（上海）	`11.53.0.0/16,11.50.0.0/16,10.152.0.0/16,10.154.0.0/16,11.132.0.0/16,11.178.0.0/16,11.200.210.74,11.200.215.195,11.217.0.0/16,11.219.0.0/16,11.222.0.0/16,11.223.116.79,11.223.69.0/24,11.223.70.0/24,11.223.70.173,11.223.70.48`
华北2（北京）	`11.223.0.0/16,11.220.0.0/16,11.204.0.0/16`
华南1（深圳）	`11.200.0.0/16`

- 访问VPC下的数据存储

  共享模式集群位于阿里云的经典网络，若需要访问阿里云VPC网络下的存储资源，需要通过VPC访问授权，参见VPC访问授权步骤。VPC授权白名单网段查询步骤如下：

  a. 登录实时计算控制台。
  b. 将鼠标悬停至页面右上角账号名称。
  c. 在下拉菜单中，单击**项目管理**。
  d. 在左侧导航栏中单击**VPC访问授权**。
  e. 在**VPC访问授权**页面中，单击对应VPC访问授权的**Region ID**字段下的链接，进入**白名单网段**页面。
  f. 在**白名单网段**窗口查询VPC授权白名单网段。

- **独享模式IP地址**

  独享模式中仅需要配置独享集群对应的弹性网卡（ENI）地址。ENI地址的查看步骤如下：

  i. 登录实时计算控制台。
  ii. 将鼠标悬停至页面右上角账号名称。
  iii. 在下拉菜单中，单击**项目管理**。
  iv. 单击左侧导航栏中的**集群列表**。
  v. 在**集群列表**页面，单击**名称**字段下目标集群名称。
  vi. 在集群信息窗口，查看集群的**ENI**信息。

配置RDS白名单

实时计算将RDS作为数据存储使用时，需要多次读写RDS数据库，必须将实时计算的IP地址配置进入RDS白名单。RDS白名单配置方法，请参见通过客户端、命令行连接RDS MySQL实例。

3.作业开发
3.1. 开发

本文为您介绍实时计算Flink版作业开发流程以及语法检查、配置作业参数、配置项目参数、SQL辅助和SQL版本管理功能。

背景信息

> **说明**
> - 实时计算Flink版主要使用Flink SQL进行作业开发，Flink SQL开发手册请参见概述。
> - 实时计算Flink版独享模式不支持归档保存已停止（含暂停）的作业运行日志。如果您需要查询已停止（含暂停）的作业运行日志，请将日志输出至您自定义的日志服务SLS或对象存储OSS中。详情请参见自定义日志级别和下载路径。

编写SQL代码

1. 登录实时计算控制台。
2. 在页面顶部，单击**开发**。
3. 在**开发**页面，单击页面顶部的**新建作业**。
4. 在**新建作业**界面，输入作业配置信息。

作业参数	说明
文件名称	作业的名称。 **说明** 作业名称在当前项目中必须保持唯一。
作业类型	○ 共享模式：仅支持FLINK_STREAM/SQL作业类型。 ○ 独享模式：支持FLINK_STREAM/DATASTREAM和FLINK_STREAM/SQL作业类型。
存储位置	在文件夹目录中，指定该作业的代码文件所属的文件夹。您还可以单击现有文件夹右侧的  图标，新建子文件夹。

5. 单击**确定**。
6. 在作业编辑页面，编写SQL代码。

> **说明**
> - 您可以在作业**开发**页面右侧的**代码结构**查看SQL代码结构。
> - 建议您使用作业**开发**页面左侧的**数据存储**管理上下游存储，详情请参见概述。

配置作业参数

1. 登录实时计算控制台。
2. 在页面顶部，单击**开发**。
3. 在左侧**作业开发**列表页面，单击目标作业名称。
4. 在目标作业**开发**页面右侧，单击**作业参数**。
5. 配置作业所需参数。

   作业参数配置详情，请参见作业参数调优。

配置项目参数

作业参数针对单个作业生效，项目参数针对该项目下所有作业生效，开启项目参数后，会产生以下两种效果：

- 替换变量：单击启动、调试或语法检查后，系统会替换SQL作业中的变量或Datastream作业中代码的变量。
- 参数下发：项目级别系统参数会与作业参数、启动参数（仅Batch作业可以配置）进行Merge，参数优先级为：启动参数 > 作业参数 > 项目级别系统参数。Merge后作为最终参数下发到Blink作业。例如，作业参数配置和项目参数配置冲突，系统则以作业参数配置为准。

1. 登录实时计算控制台。
2. 在页面顶部菜单栏上，鼠标悬停在用户头像后，单击**项目管理**。
3. 在**项目列表**区域，单击目标**项目名称**。
4. 在页面顶部，单击**开发**。
5. 在左侧**作业开发**列表页面，单击目标作业名称。
6. 配置项目参数生效。

   该功能默认关闭（`disable.project.config=false`），您可以按照以下方式配置生效：

   - SQL作业：在**作业参数**中配置`enable.project.config=true`。
   - Datastream作业：在代码中配置`enable.project.config=true`。
7. 在页面顶部，单击**项目参数**。
8. 配置项目所需参数。

   项目参数仅支持SQL和Datastream两种作业类型，在配置项目级别系统参数时，您需要在项目级别配置参数前添加作业类型前缀，例如，`sql.name=LiLei`或`datastream.name=HanMeimei`。

启动语法检查

1. 登录实时计算控制台。
2. 在页面顶部，单击**开发**。
3. 在左侧**作业开发**列表页面，单击目标作业名称。
4. 在目标作业**开发**页面上方，单击**语法检查**。

> **说明**
>
> - 保存作业可以触发SQL语法检查功能。
> - 请编写完整的SQL逻辑后再进行**语法检查**，否则**语法检查**不生效。

SQL辅助

- Flink SQL语法检查

  在您修改SQL后即可自动保存。保存操作可以触发SQL语法检查功能。语法校验出错后，将在作业**开发**页面提示出错行数、列数以及错误原因。

- Flink SQL智能提示

  在您输入Flink SQL过程中，作业**开发**页面提供包括关键字、内置函数、表和字段智能记忆等提示功能。

- Flink SQL语法高亮显示

  高亮显示Flink SQL中关键字，使用不同的颜色区分Flink SQL语法中不同的结构。

SQL版本管理

实时计算Flink版为您提供代码版本管理功能。每提交一次作业即可生成一个代码版本。代码版本用于版本追踪、版本修改以及后期版本回滚。

1. 登录实时计算控制台。
2. 在页面顶部，单击**开发**。
3. 在左侧**作业开发**列表页面，单击目标作业名称。
4. 在目标作业**开发**页面右侧，单击**版本信息**。
5. 单击**操作** > **更多**。
6. 选择相应的版本管理功能。
   - **对比**：查看最新代码和指定版本的差异。
   - **回滚**：回滚到指定版本。
   - **删除**：实时计算Flink版默认版本数上限为20。在版本数小于20时，您可以提交作业。如果当前的版本数为20，系统将不允许该作业的提交请求，并提示您删除部分旧版本作业。

     > **说明** 当前版本数低于版本上限数后可以再次提交作业。

   - **锁定**：锁定当前作业版本。

     > **说明** 解锁前无法提交新版本。

3.2. 上线

完成作业开发、作业调试，并且通过语法检查后，上线作业，即可将数据发布至生产环境。

上线步骤

1. 资源配置

   选择对应的资源配置方式。第1次启动建议使用系统默认配置。

> **说明** 实时计算Flink版支持手动资源配置和自动资源配置两种资源配置方式：
>
> - 手动资源配置方法，请参见手动配置调优。
> - 自动资源配置方法根据实时计算Flink版本，分为以下两种方式：
>   - 实时计算Flink版 3.0以上版本：AutoScale自动配置，详情请参见AutoScale自动配置调优。
>   - 实时计算Flink版 3.0及以下版本：AutoConf自动配置，详情请参见AutoConf自动配置调优。

2. 数据检查

   通过数据检查后，单击**下一步**。

3. 上线作业

   单击**上线**。

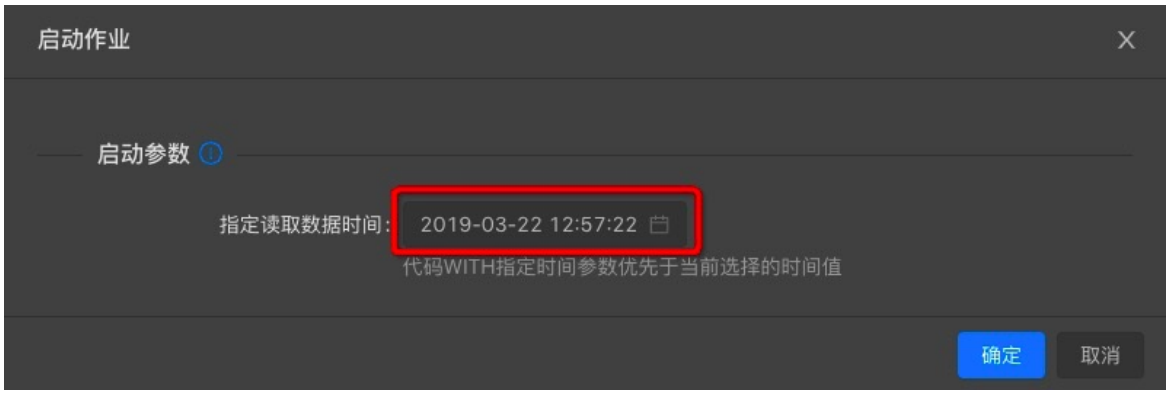
> **说明** 作业上线后只是将作业提交至集群，并没有启动作业。启动步骤，详情请参见启动。

3.3. 启动

完成作业开发和作业上线后，您可以在运维页面启动作业。

操作步骤

1. 登录实时计算控制台。
2. 单击页面顶部的**运维**。
3. 在**运维**页面，单击目标作业**操作**列下的**启动**。
4. 在**启动作业**页面，单击**指定数据读取数据时间**（即指定启动位点）文本框。



5. 指定读取数据时间（启动位点），单击**确定**，完成作业启动。

   启动位点表示从数据源表中读取数据的时间点：

   - 选择当前时间：表示从当前时间开始读取数据。
   - 选择历史时间：表示从历史时间点开始读取数据，通常用于回追历史数据。

> **说明** 作业启动完成后即可进入运行信息阶段。

3.4. 暂停

修改资源配置后，可以经过暂停和恢复的步骤使变更生效。本文为您介绍如何暂停作业。

背景信息

注意

- 只能对运行状态为**运行**的作业进行**暂停**操作。
- **暂停**操作不会清除任务状态，即如果有COUNT操作，作业**暂停** > **恢复**后，COUNT会从上次成功Checkpoint的状态开始继续计算。
- 实时计算3.5.0以上版本才可以使用暂停（Checkpoint）功能，否则右上角会出现报错：**发生错误系统错误：BLINK版本异常。 错误原因：** blink version >= blink-3.5 is required, instance blink-3.4.4。

操作步骤

1. 登录实时计算控制台。
2. 单击页面顶部的**运维**。
3. 在**运维**页面，单击目标作业**操作**列下的**暂停**。

> **说明** 更多目录的**暂停（Checkpoint）**在进行暂停作业的同时，会主动触发一次Checkpoint，因此**暂停（Checkpoint）**所消耗的时间可能会比**暂停**的时间长。

3.5. 停止

更改SQL逻辑、更改作业版本、增加WITH参数或增加作业参数后，经过停止和启动的步骤，才能使变更生效。本文为您介绍如何停止作业。

注意

- 只能对运行状态为**运行**或**启动**中的作业进行**停止**操作。
- **停止**操作会清除任务状态，即如果有COUNT操作，作业**停止** > **启动**后，COUNT从0开始计算。
- 实时计算3.5.0以上版本才可以使用**停止（checkpoint）**功能，否则右上角会出现报错：**发生错误系统错误：BLINK版本异常。 错误原因：** blink version >= blink-3.5 is required, instance blink-3.4.4。

作业停止操作步骤如下：

1. 登录实时计算控制台。
2. 单击页面顶部的**运维**。
3. 在**运维**页面，单击目标作业**操作**列下的**停止**。

> **说明** 更多目录的**停止（checkpoint）**功能与**停止**功能的唯一区别是：**停止（checkpoint）**在进行停止作业的同时，会主动触发一次Checkpoint，因此**停止（checkpoint）**作业所消耗的时间可能会比**停止**作业所消耗的时间稍长一些。但作业停止后依然会清除作业的状态，该功能在个别场景下会有其他作用，例如在上游存储为kafka时，系统触发一次Checkpoint会提交一次offset，确保提交到kafka服务端的offset和实际消费的数据量一致。

4.作业调试

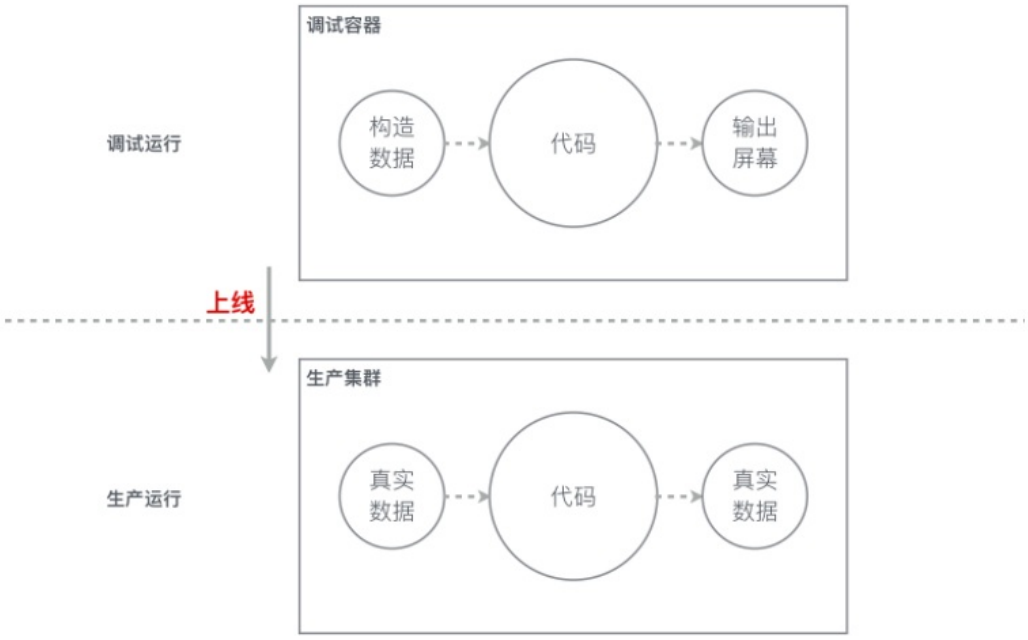
4.1. 本地调试

实时计算开发平台为您提供了一套本地调试环境，您可以在本地调试环境中上传自定义数据、模拟作业运行、检查输出结果，最终验证业务逻辑的正确性。

本地调试环境的特点

本地调试环境与生产环境完全隔离。本地调试环境中，所有的Flink SQL在独立的调试容器中运行，调试结果输出至调试环境页面，不会对线上生产流、线上实时计算作业或线上数据存储系统造成影响。

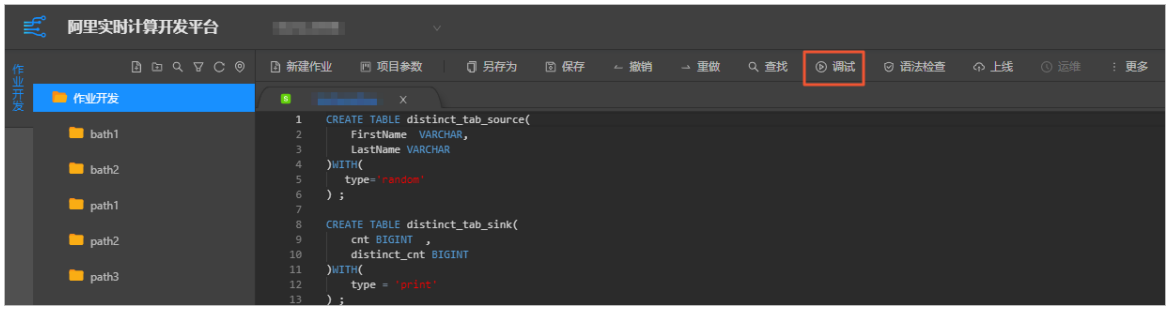
> **说明** 实时计算调试模式无法检查出数据存储中的因为数据格式兼容性问题而导致的运行失败。例如，输出数据长度大于RDS建表最大值的问题。



本地调试步骤

> **说明** 进行作业调试前，请您先完成作业的开发。作业开发流程请参见开发。

1. 登录实时计算控制台。
2. 在页面顶部，单击**开发**。
3. 在目标作业开发编辑区域，单击页面顶部的**调试**。



4. 在**调试作业**页面，输入测试数据。本地调试提供2种调试数据输入方式：
   - 本地上传方式
     a. 在**数据预览**区域，单击**下载模板**。
     b. 根据模板，编辑自定义调试数据。

     > **说明** 调试数据的默认分隔符为逗号（,），如果需要自定义分隔符请参见调试数据分隔符。

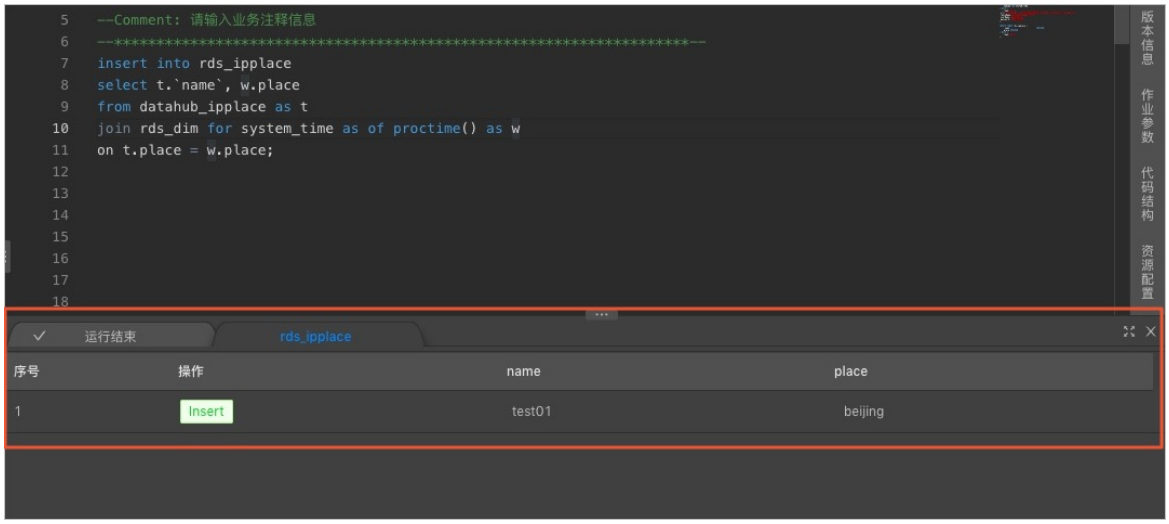
     c. 在**数据预览**区域，单击**上传** ，上传自定义调试数据。
   - 线上抽样方式

     > **说明** 使用**顺序抽样线上数据**功能前，请确保数据源在抽取时间段存在数据。

     a. 在**数据预览**区域，单击**随机抽样线上数据**或**顺序抽样线上数据**。
     b. 输入抽样配置信息。
     c. 单击**确定**。

     > **说明** 数据上传成功后，可在**数据预览**区域查看已上传数据。

5. 单击**确定**，启动调试。
6. 在作业编辑区域的底部，查看调试输出结果。



调试数据分隔符

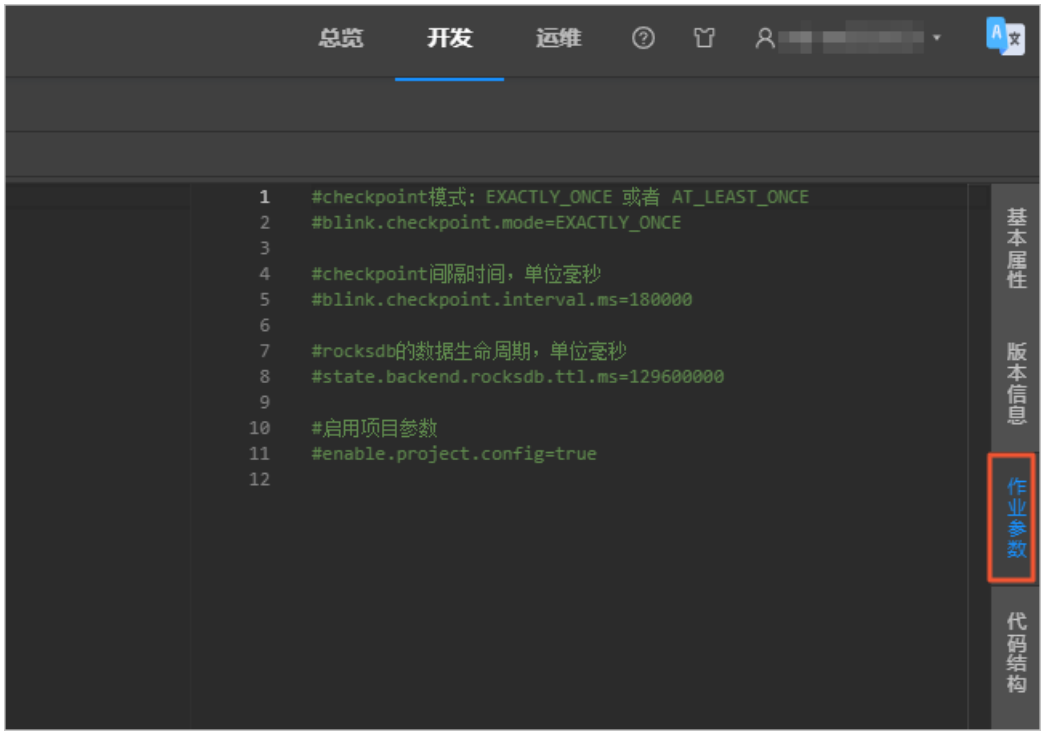
调试数据默认使用逗号（,）作为分隔符，如果输入的数据内容（例如JSON文件）中已存在逗号（,），您需要自定义其它字符作为调试数据的分隔符，例如竖线（|）。

> **说明** 实时计算仅支持使用单个英文字符（例如竖线（|））为分隔符。不支持使用字符串（例如 `aaa` ）作为分隔符。

调试数据分隔符的配置步骤如下：

> **说明** 配置数据分隔符前，请您先完成作业的开发。作业开发流程请参见开发。

1. 在作业编辑区域，单击右侧的**作业参数**。



2. 在作业参数编辑区域，输入调试数据分隔符的配置参数。配置分隔符为竖线（|）的示例代码如下。

```
debug.input.delimiter = |
```

本地调试UDX的日志输出

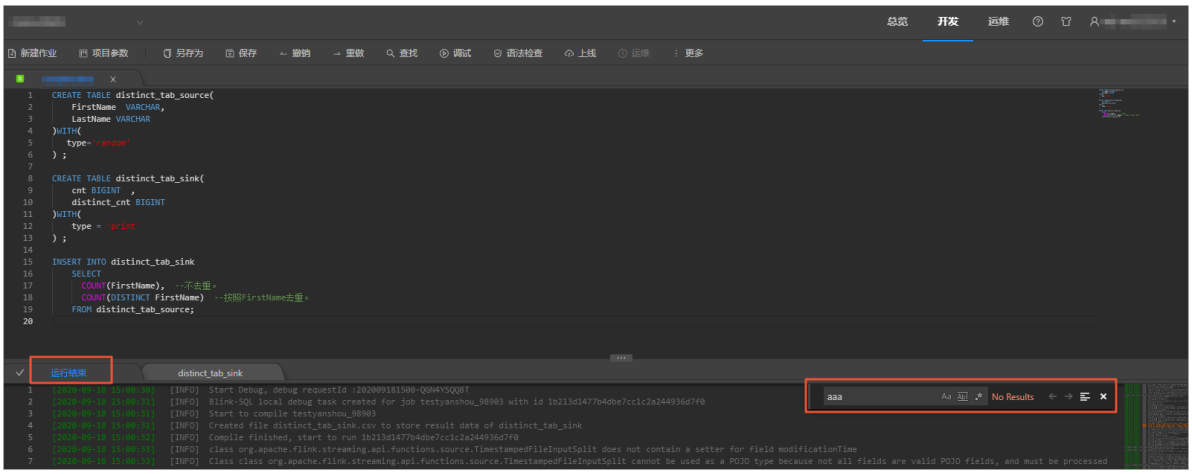
- 本地调式时打印UDX中日志

  在Java中通过以下方法，将日志的格式转换为实时计算可解析的格式，并打印本地调试的UDX日志。

```
public static void debugMsgOutput(String msg) {
    System.out.println(
      String.format("{\"type\":\"log\",\"level\": \"INFO\", \"time\": \"%s\", \"message\"
: \"%s\", \"throwable\": \"null\"}\n",new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(
new Date()), msg));
}
```

- 查看UDX的日志输出

调试结束后，在作业编辑区域底部的**运行结束**页面，可以查看UDX的日志输出。



> **说明** 您可以通过快键Ctrl+F的方式搜索相应的日志信息。

4.2. 线上调试

阿里云实时计算开发平台为您提供了实时计算作业线上调试功能。相对于本地调试功能，线上调试功能需要消耗一定的CU资源，但是能够更加真实的验证业务逻辑的正确性。

线上调试功能使用真实的数据存储，有效地减少调试输出和生产输出的差异，有助于您在调试阶段发现问题。

线上调试步骤

1. 开发作业。作业开发详情，请参见开发。
2. 更新数据存储DDL中的 `type` 参数。
   - 源表：`type = 'random'`
   - 结果表：`type = 'print'`
3. 上线作业。作业上线步骤请参见上线。
4. 启动作业。作业启动步骤请参见启动。

线上调试Connector

阿里实时计算平台提供以下2种线上调试功能的Connector:

- `random` 源表：周期性的生成对应类型的随机数据。
- `print` 结果表：输出计算结果。

Connector表参数

- Random表参数

参数	说明
type	必选，取值唯一且为random。

参数	说明
interval	可选，产生数据的时间间隔（单位为毫秒），默认值为500。

- Print表参数

参数	说明
type	必选，取值唯一且为print。
ignoreWrite	可选，默认值为false。可选参数值如下： ○ false：同时输出结果表和日志。 ○ true：仅输出无数据的结果表，不输出日志数据。

线上调试示例

- 测试语句

```
CREATE TABLE random_source (
  instr               VARCHAR
) WITH (
  type = 'random'
);
CREATE TABLE print_sink(
  instr VARCHAR,
  substr VARCHAR,
  num INT
)with(
  type = 'print'
);
INSERT INTO print_sink
SELECT
  instr,
  SUBSTRING(instr,0,5) AS substr,
  CHAR_LENGTH(instr) AS num
FROM random_source
```

- 测试结果

线上调试结果查询步骤

> **说明** 查询线上调试结果前，请先完成作业上线和启动。

线上调试结果查询查看步骤如下：

1. 登录实时计算控制台。
2. 单击顶部菜单栏中的**运维**，进入**运维**页面。

3. 单击**作业名称**字段下对应的作业，进入**作业运维**页面。
4. 在**Vertex拓扑**区域，单击相应结果表节点。
5. 单击**SubTask List > 查看日志**，进入**日志查看**窗口。
6. 查看相应的日志。
   - Print结果表输出

     单击taskmanager.out右侧的**查看日志**。
   - UDX日志输出

     如果您使用了自定义函数UDX（UDX使用方法请参见概述），可以使用以下两种方式查看日志：
     - system out/err方法

       单击 `taskmanager.out` 或 `taskmanager.err` 右侧的**查看日志**。
     - SLF4J的Logger方法

       单击 `taskmanager.log` 右侧的**查看日志**。

5.作业运维

5.1. 运行信息

运行信息为您展示作业的实时运行信息。您可以通过作业的状态来分析、判断作业的状态是否健康、是否达到您的预期。

登录运行信息页面

1. 登录**作业运维**页面。
   i. 登录实时计算控制台。
   ii. 单击页面顶部的**运维**。
   iii. 在**作业列表**区域，单击**作业名称**下的目标作业名。
2. 在**作业运维**页面，单击页面顶部的**运行信息**。

Task状态

Task状态为您显示作业各状态的数量。Task存在以下7种状态：

- **创建**
- **运行**
- **失败**
- **完成**
- **调度**
- **取消中**
- **已取消**

作业瞬时值

Task状态下方为您展示作业的瞬时数值。

名称	作用描述
输入TPS	每秒从源端读到的Block数。对于日志服务而言，可以将多条数据打包到一个LogGroup读取。Block数反映的是从源端每秒读取LogGroup数量。
输入RPS	每秒读取数据源表的RPS数，单位为条/秒。
输出RPS	每秒写入数据结果的RPS数，单位为条/秒。
输入BPS	每秒读取数据源表的BPS数，单位为Block。
消耗CU	单个Job当前的CU的消耗状况。
最近启动时间	Job的启动的时间。
运行时长	Job的启动后运行的时间。

Vertex拓扑

Vertex拓扑描述Realtime Compute底层计算逻辑的执行图。每个组件代表不同的Task，每个数据流从一个或多个数据源，流向另一个或多个数据结果表。数据流类似于任意有向无环图（DAG）。为了更高效地分布式执行，Realtime Compute底层会将Operator的Subtask链接（Chain）在一起形成Task，每个Task在一个线程中执行。将Operators链接成Task能减少线程之间的切换、消息的序列化或反序列化，以及数据在缓冲区的交换，降低了延迟的同时提高整体的吞吐量。Operator代表的是每个计算逻辑的算子，而Task代表是多个Operator的集合。

- 显示模式

  Vertex拓扑默认显示拓扑图，您可以单击右上角的**列表模式**，完成显示模式的切换。

- Task节点状态信息
  - 视图模式Task参数信息

视图模式的节点框中为您显示了当前Task节点的信息，参数说明如下。

名称	信息描述
资源健康分	通过**资源健康分**检查机制， 反映作业的性能状况。**资源健康分**小于60分表明当前节点存在数据堆积，数据处理性能较差。 **说明** 数据处理性能较差时，建议使用AutoConf自动配置调优或手动配置调优提升性能。
PARALLEL	并发量。
TPS	每秒读取上游数据的Block数。
DELAY	Task节点的业务延时。
IN_Q	Task节点的输入队列的百分比。
OUT_Q	Task节点的输出队列的百分比。

- 列表模式Task参数信息

  Vertex拓扑底部，可以通过列表模式查看Task节点的信息，参数说明如下。

名称	信息描述
Name	每个Task的详情信息。
Status	每个Task的运行的状态。
In Queue	Task节点的输入队列，单位是百分比。
Out Queue	Task节点的输出队列，单位是百分比。
Delay(ms)	Task节点的业务延时。
TPS	每秒读取上游的信息量。
Bytes Received	Task节点接收到的数据量。
Records Received	Task节点接收到的记录数。
Bytes Sent	Task节点发送的数据量。
Records Sent	Task节点发送的记录数。
Task	每个Task节点的并发的运行状态。

- Task节点线程信息

  单击Task节点，在**SubTasks**页签，查看Task的线程列表。

Vertex信息

> **说明** 仅实时计算3.0.0以上版本支持以下功能。

- Vertex内部Operator信息

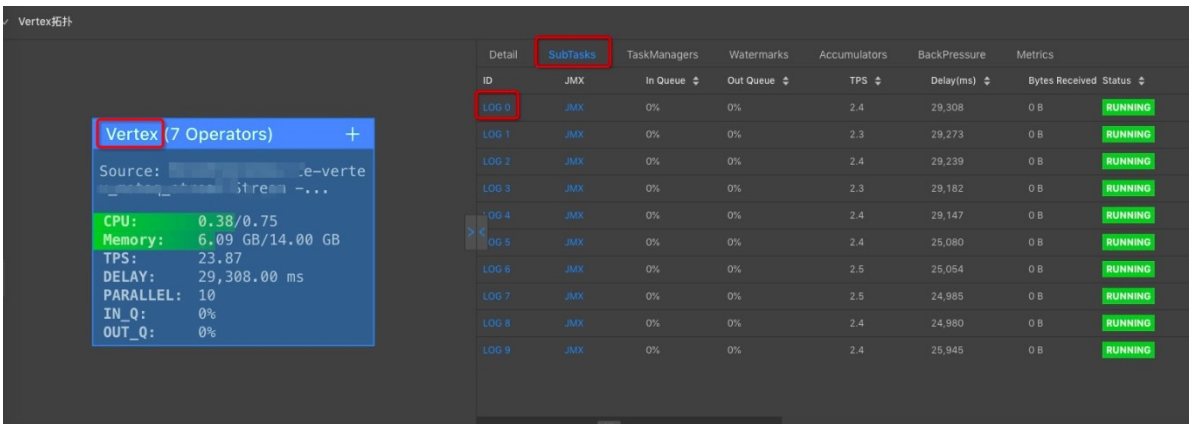
  在Vertex拓扑区域，单击单个Vertex框右上角加号（+），可以显示Vertex内部Operator信息。

- 显示Vertex内部详情

  单击Vertex拓扑区域右上角的**Expand All**，可以显示所有Vertex内部详情。

- Vertex详情页面

在Vertex拓扑区域，单击Vertex边框或Vertex列表中的Name即可弹出右侧Vertex的详情页面。在Vertex的详情页面中，单击**SubTasks**页签中的Task ID（下图中的**LOG 0**），跳转至**TaskManager**相关日志页面。



5.2. 数据曲线

阿里云实时计算提供了当前作业的核心指标概览页面。您可以通过数据曲线对作业的运行情况进行一键式的诊断。

数据曲线示例如下。

> **说明**
>
> - 实时计算作业只有在**运行**状态下才显示数据曲线指标，**暂停**和**停止**状态均不显示数据曲线指标。
> - 作业指标是实时计算系统异步后台采集，存在一定延迟。作业启动1分钟后，开始逐步采集各项指标，并在数据曲线显示。

登录数据曲线页面

1. 登录**作业运维**页面。
   i. 登录实时计算控制台。
   ii. 单击页面顶部的**运维**。
   iii. 在**作业列表**区域，单击**作业名称**下的目标作业名。

2. 在**作业运维**页面，单击页面顶部的**数据曲线**。

OverView

- **Failover**

  Failover曲线显示当前Job出现Failover（错误或异常）的频率。计算方法为当前Failover时间点的前1分钟内出现Failover的累计次数除以60。例如，最近1分钟Failover一次，Failover的值为1/60=0.01667。

- **延时**

  为了全面地了解实时计算全链路的时效状况和作业的性能，实时计算提供3种延时指标：

  - **业务延时**（Processing Delay）：业务延迟=当前系统时间-当前系统处理的最后一条数据的事件时间（Event time）。如果后续没有数据再进入上游存储，由于当前系统时间在不断往前推进，业务延时也会随之逐渐增大。
  - **数据滞留时间**（Data Pending Time）：数据滞留时间=数据进入实时计算的时间-数据事件时间（Event time）。即使后续没有数据再进入上游存储，数据滞留时间也不会随之逐渐增大。通常用数据滞留时间来评估当前实时计算作业是否存在反压。
  - **数据间隔时间**（Data Arrival Interval）：数据间隔时间=业务延迟-数据滞留时间。当实时计算没有反压时，数据滞留时间较小且平稳，数据间隔时间可以反映数据源数据间的稀疏程度。当实时计算存在反压时，数据滞留时间较大或不平稳，此参数没有实质性参考意义。

  > **说明**
  > - 实时计算是分布式计算框架，以上3类延时指标的Metric，在计算Source的单个分区（Shard/Partition等）后，汇报所有分区中的最大值并呈现到前端页面上。因此，前端页面上显示的汇聚后的数据间隔时间并不精确等于业务延时-数据滞留时间。
  > - 如果Source中的某个分区没有新的数据，将会导致业务延迟逐渐增大。

- **各Source的TPS数据输入**

  对实时计算作业所有的流式数据输入进行统计，记录每秒读取数据源表的Block的数，让您直观地了解数据存储TPS（Transactions Per Second）的情况。与TPS不同，RPS（Record Per Second）是读取数据源TPS的Block数解析后的数据，单位是条/秒。例如，如果日志服务1秒读取5个LogGroup，则TPS=5。如果每个LogGroup解析出来8个日志记录，则一共解析出40个日志记录，RPS=40。

- **各Sink的数据输出**

  统计实时计算作业所有的数据输出（并非是流式数据存储，而是全部数据存储），让您直观地了解数据存储RPS（Record Per Second）的情况。通常，在系统运维过程中，如果出现没有数据输出的情况，除了检查上游是否存在数据输入，也要检查下游是否真的存在数据输出。

- **各Source的RPS数据输入**

  统计实时计算作业所有的流式数据输入，让您直观地了解数据存储RPS（Record Per Second）情况。通常，在系统运维过程中，如果出现没有数据输出的情况，需要查看该值，判断数据源输入数据是否存在异常。

- **各Source的数据流量输入**

  统计实时计算作业所有的流式数据输入和每秒读取输入源表的流量，让您直观地了解数据流量BPS（Byte Per Second）情况。

- **各Source的脏数据**

  显示实时计算Source各时间段脏数据条数。

- **AutoScale的成功和失败数**

显示AutoScale成功执行和未成功执行的次数。

> 注意 仅实时计算3.0.0以上版本支持此曲线。

- **AutoScale使用的CPU**

  显示执行AutoScale时消耗的CPU数量。

  > 注意 仅实时计算3.0.0以上版本支持此曲线。

- **AutoScale使用的MEM**

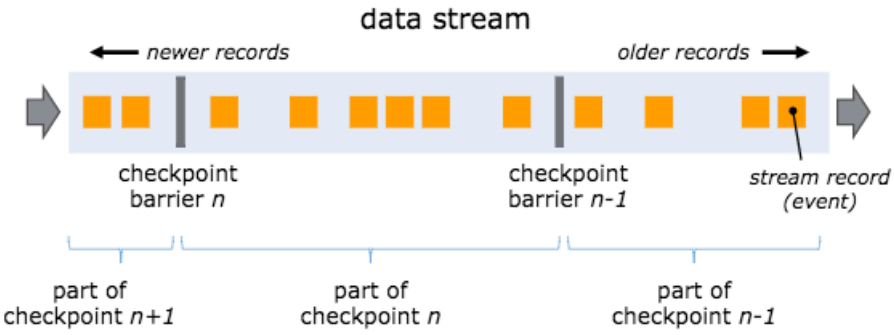
  显示执行AutoScale时消耗的内存量。

  > 注意 仅实时计算3.0.0以上版本支持此曲线。

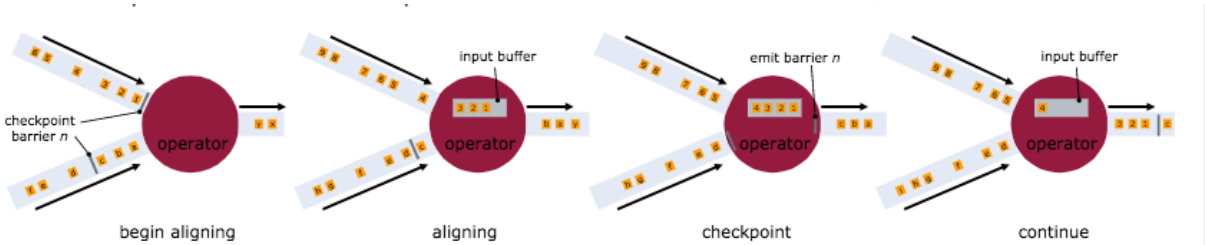
Advanced View

阿里云实时计算提供可以恢复数据流应用到一致状态的容错机制。容错机制的核心就是持续创建分布式数据流及其状态的一致快照。这些快照在系统遇到故障时，充当可以回退的一致性检查点（Checkpoint）。

分布式快照的核心概念之一就是数据栅栏（Barrier）。这些Barrier被插入到数据流中，作为数据流的一部分和数据一起向下流动。Barrier不会干扰正常数据，数据流严格有序。一个Barrier把数据流分割成两部分：一部分进入到当前快照，另一部分进入下一个快照。每一个Barrier都带有快照ID，并且Barrier之前的数据都进入了此快照。Barrier不会干扰数据流处理，所以非常轻量。多个不同快照的多个Barrier会在流中同时出现，即多个快照可能同时被创建。



Barrier在数据源端插入，当Snapshot n的Barrier插入后，系统会记录当前Snapshot位置值n（用Sn表示），然后Barrier继续往下流动。当一个Operator从其输入流接收到所有标识Snapshot n的Barrier时，它会向其所有输出流插入一个标识Snapshot n的Barrier。当Sink Operator（DAG流的终点）从其输入流接收到所有Barrier n时，Operator向检查点协调器确认Snapshot n已完成。当所有Sink都确认了这个快照时，快照就被标识为完成。



以下是记录Checkpoint的各种参数配置。

数据曲线名称	说明
Checkpoint Duration	进行Checkpoint保存状态所花费的时间，单位为毫秒。
Checkpoint Size	进行Checkpoint所消耗的内存大小。
Checkpoint Alignment Time	当前节点进行Checkpoint操作，等待上游所有节点到达当前节点的时间。当Sink Operator（DAG流的终点）从其输入流接收到所有Barrier n时，它会向Checkpoint Coordinator确认Snapshot n已完成。当所有Sink都确认了这个快照，快照就被标识为完成。这个等待的时间就是CheckpointAlignmentTime。
Checkpoint Count	指定时间段内Checkpoint的数量。
Get	指定时间段内每个SubTask对ROCKSDB进行Get操作所花费的时间（最大值）。
Put	指定时间段内每个SubTask对ROCKSDB进行Put操作所花费的时间（最大值）。
Seek	指定时间段内每个SubTask对ROCKSDB进行Seek操作所花费的时间（最大值）。
State Size	指定时间段内Job内部State存储大小（如果增量过快，Job是异常的）。
GMS GC Time	指定时间段内Job内部容器进行GC（Garbage Collection）花费的时间。
GMS GC Rate	指定时间段内Job内部容器进行GC（Garbage Collection）的频率。

WaterMark

数据曲线名称	说明
Watermark Delay	WaterMark距离系统时间的差值。
数据迟到丢弃TPS	每秒丢弃晚于Watermark时间到达Window的数据量。
数据迟到累计丢弃数	累计丢弃晚于Watermark时间到达Window的数据量。

Delay

Source SubTask 最大延迟 Top 15

表示每个Source并发的业务延时的时长。

Throughput

数据曲线名称	说明
Task Input TPS	作业中所有的Task的数据的输入状况。
Task Output TPS	作业中所有的Task的数据的输出状况。

Queue

数据曲线名称	说明
Input Queue Usage	作业中所有的Task的数据的输入队列。
Output Queue Usage	作业中所有的Task的数据的输出队列。

Tracing

数据曲线名称	说明
Time Used In Processing Per Second	处理Task中每秒数据所花费的时长。
Time Used In Waiting Output Per Second	等待Task中每秒数据输出所花费的时长。
Task Latency Histogram Mean	每个Task的延时时长。
Wait Output Histogram Mean	每个Task的等待输出时长。
Wait Input Histogram Mean	每个Task的等待输入时长。
Partition Latency Mean	每个分区中并发的延时时长。

Process

数据曲线名称	说明
Process Memory RSS	每个进程内存的使用状况。
CPU Usage	每个进程CPU的使用状况。

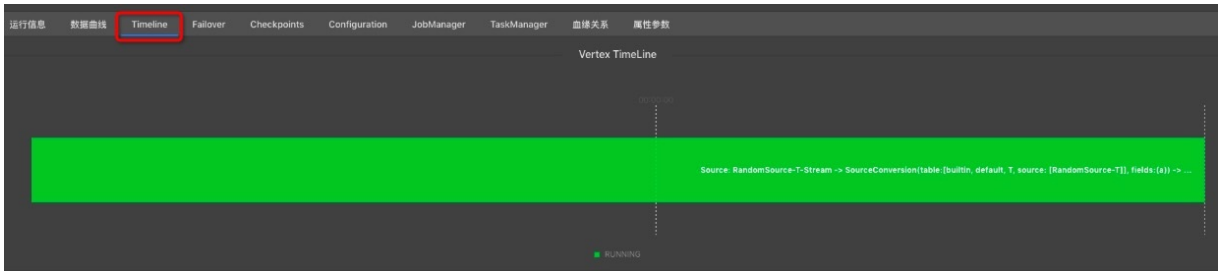
JVM

数据曲线名称	说明
Memory Heap Used	Job使用的JVM Heap存储量。
Memory Non-Heap Used	Job使用的JVM 非Heap存储量。
Thread Count	Job的线程数。
GC(CMS)	Job完成GC（Garbage Collection）的次数。

5.3. Timeline

Timeline页面为您显示各Vertex从启动位点到当前时间的运行状态。

> **说明** 仅实时计算3.0.0以上版本支持Timeline功能。



5.4. Failover

阿里云实时计算提供了当前作业的Failover页面，您可以在Failover页面了解当前运行作业的运行情况和报错信息。

登录Failover页面

1. 登录**作业运维**页面。
   i. 登录实时计算控制台。
   ii. 单击页面顶部的**运维**。
   iii. 在**作业列表**区域，单击**作业名称**下的目标作业名。
2. 在**作业运维**页面，单击页面顶部的**Failover**。

Latest FailOver

Latest FailOver为您展示作业当前的报错信息。

> **说明** 仅支持实时计算3.0以下版本。

FailOver History

FailOver History为您展示作业的历史报错信息。

> **说明** 仅支持实时计算3.0以下版本。

Root Exception

Root Exception为您展示作业当前的报错信息。

> **说明** 仅支持实时计算3.0及以上版本。

Exception History

Exception History为您展示作业的历史报错信息。

> **说明** 仅支持实时计算3.0及以上版本。

5.5. Checkpoints

阿里云实时计算提供可以恢复数据流，并和应用保持一致状态的容错机制。容错机制的核心是持续创建分布式数据流及其状态的快照。当系统出现故障时，这些快照充当可以回退的一致性检查点（Checkpoint）。

登录Checkpoints页面

1. 登录**作业运维**页面。
   i. 登录实时计算控制台。
   ii. 单击页面顶部的**运维**。
   iii. 在**作业列表**区域，单击**作业名称**下的目标作业名。
2. 在**作业运维**页面，单击页面顶部的**Checkpoints**。

Overview

> **说明** 该功能仅适用于实时计算3.0及以上版本。

Overview为您展示最新的Checkpoint信息，包括各节点Checkpoint的进程、Duration、StateSize等信息。

History

> **说明** 该功能仅适用于实时计算3.0及以上版本。

History为您展示近期Checkpoint的信息。单击行首的（+）可展现各节点Checkpoint的进程、Duration和StateSize等信息。

Summary

> **说明** 该功能仅适用于实时计算3.0及以上版本。

Summary为您展示已完成的Checkpoint的平均值、最大值和最小值信息。

Configuration

> **说明** 该功能仅适用于实时计算3.0及以上版本。

Configuration为您展示Checkpoint的配置信息。

Completed Checkpoints

> **说明** 该功能仅适用于实时计算3.0以下版本。

Completed Checkpoints为您展示已完成的Checkpoint信息。

名称	详情描述
ID	Checkpoint的ID编号
Start Time	Checkpoint开始的时间
Durations (ms)	Checkpoint花费的时间

Task Latest Completed Checkpoint

> **说明** 该功能仅适用于实时计算3.0以下版本。

Task Latest Completed Checkpoint为您展示最新一次Checkpoint的详细信息。

名称	详情描述
SubTask ID	SubTask的ID编号
State Size (Bytes)	Checkpoint的大小
Durations (ms)	Checkpoint的花费的时间

5.6. JobManager

JobManager是实时计算集群启动的重要组成部分，您可以在JobManager页面查看JobManager的详细参数信息。

登录JobManager页面

1. 登录**作业运维**页面。
    i. 登录实时计算控制台。

ii. 单击页面顶部的**运维**。

iii. 在**作业列表**区域，单击**作业名称**下的目标作业名。

2. 在**作业运维**页面，单击页面顶部的**JobManager**。

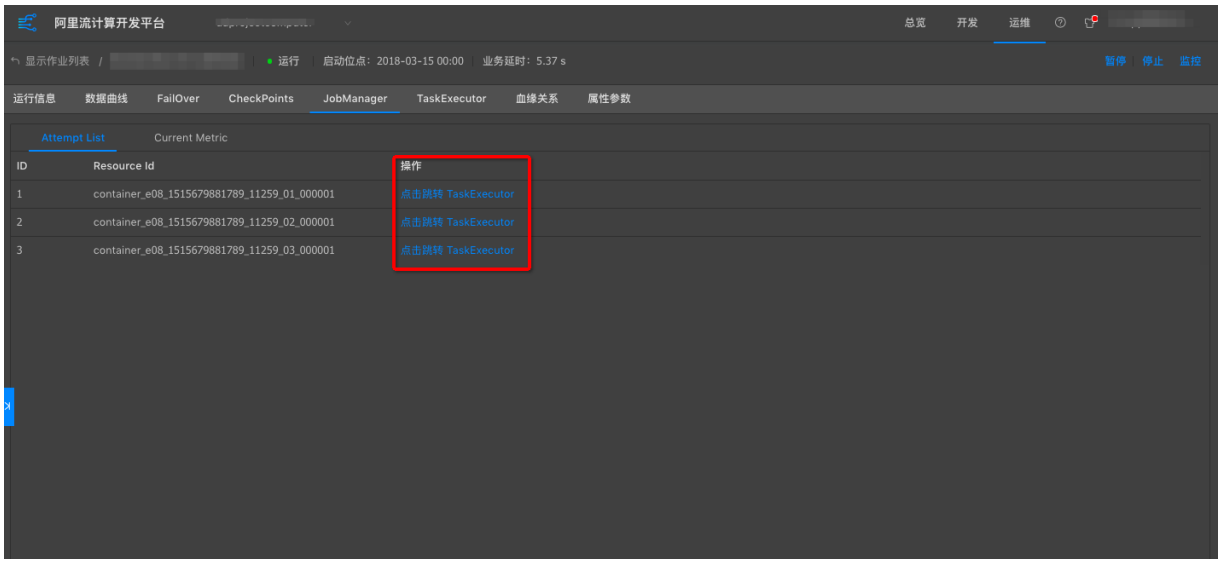
JobManager在集群启动中的作用

JobManager是实时计算集群的启动过程不可或缺的一部分。实时计算集群的启动流程如下：

1. 实时计算集群启动一个JobManager和若干个TaskExecutor。
2. Client向JobManager提交任务。
3. JobManager向TaskExecutor分配任务。
4. TaskExecutor向JobManager汇报心跳和统计信息。

JobManager参数信息

您可以在**JobManager > Attempt List** 页面，单击**操作**字段下的**查看详情**，查看JobManager的详细信息。



5.7. TaskExecutor

本文为您介绍TaskExecutor在实时计算集群启动过程中的作用以及TaskExecutor的界面。

> **注意** 本文档仅适用于实时计算3.0以下版本。

背景

TaskExecutor是实时计算集群的启动过程不可或缺的一部分。TaskExecutor负责接收任务并将信息返还。TaskExecutor在启动时即完成了槽位数（Slot）的设置，每个Slot只能启动1个Task线程。TaskExecutor从JobManager处接收需要部署的Task，部署启动后，与上游建立Netty连接，接收数据并处理。

登录TaskExecutor界面

1. 登录实时计算控制台。
2. 单击界面顶部的**运维**。

3. 在**作业列表**区域，单击**作业名称**下的目标作业名。
4. 在**作业运维**界面，单击顶部的**TaskExecutor**。

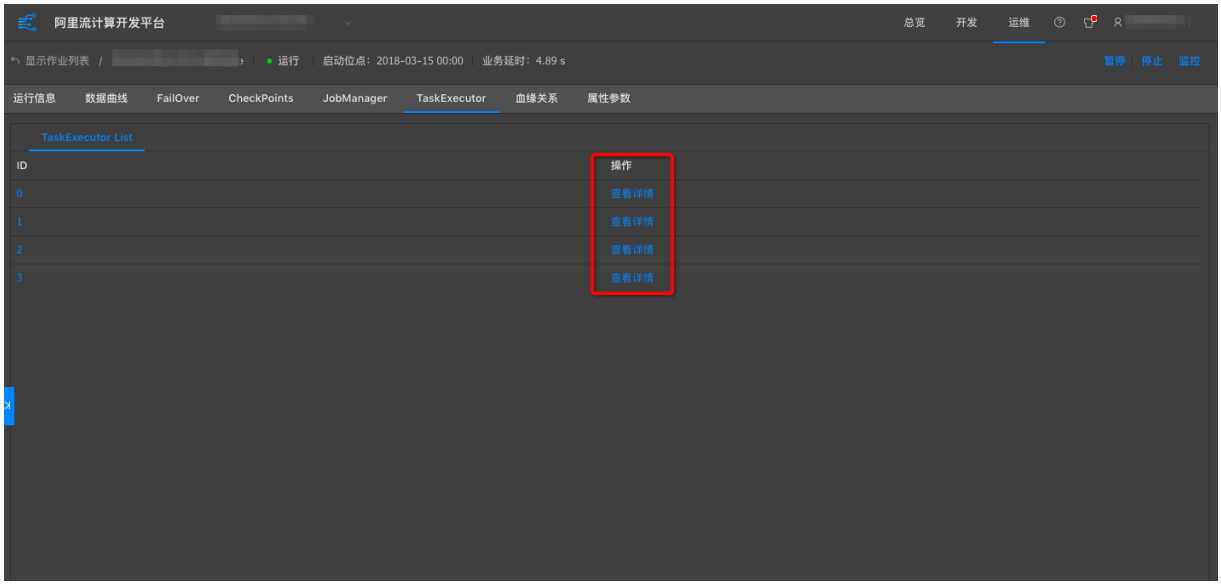
TaskExecutor在集群启动中的作用

TaskExecutor是实时计算集群的启动过程不可或缺的一部分。实时计算集群的启动流程如下：

1. 实时计算集群启动一个JobManager和若干个TaskExecutor。
2. Client向JobManager提交任务。
3. JobManager向TaskExecutor分配任务。
4. TaskExecutor向JobManager汇报心跳和统计信息。

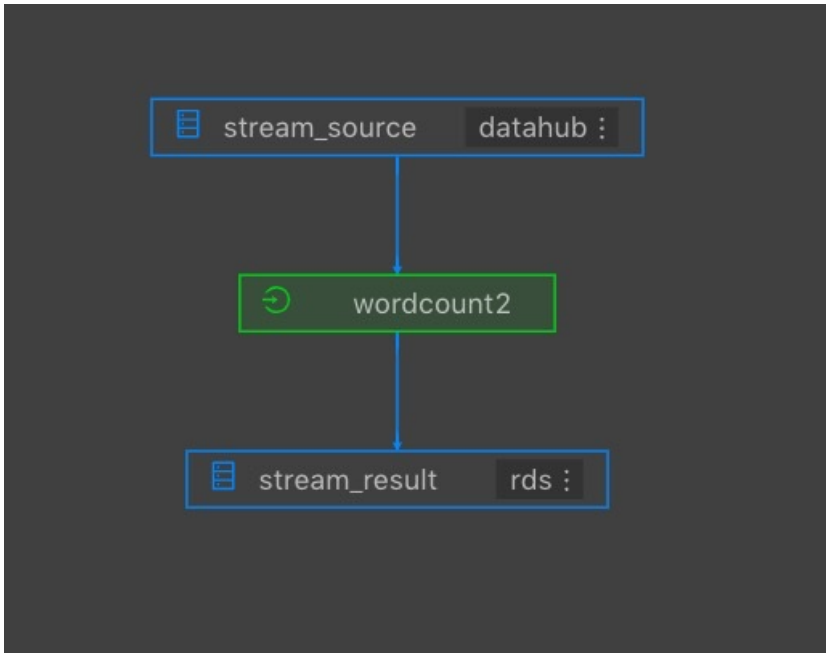
TaskExecutor界面

TaskExecutor界面为您提供Task列表以及Task详情的接口。



5.8. 血缘关系

实时计算作业的血缘关系集中反映了一个实时计算作业上下游数据的依赖关系。对于作业较为复杂的上下游业务依赖，血缘关系中的数据拓扑图能够清晰地反映出上下游依赖信息。



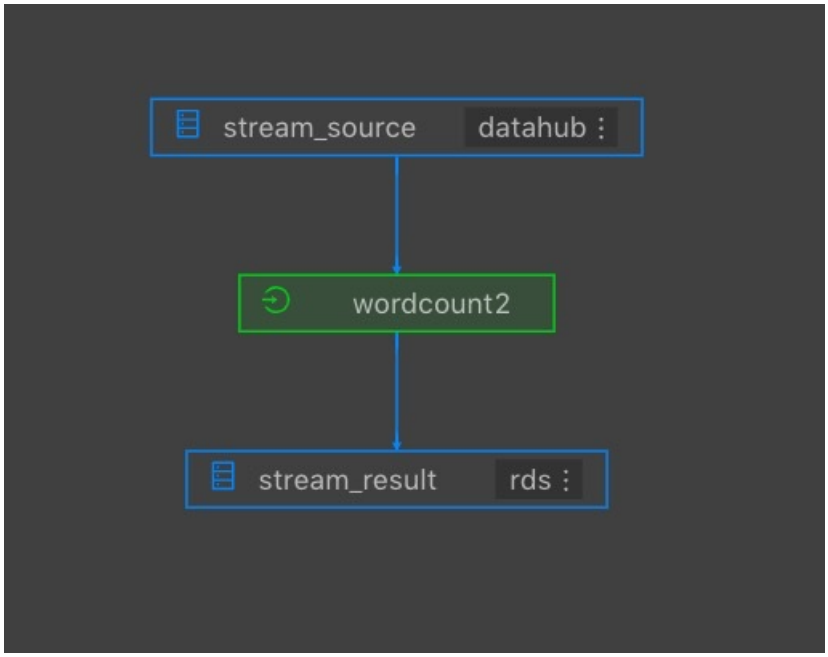
登录血缘关系页面

1. 登录**作业运维**页面。
   i. 登录实时计算控制台。
   ii. 单击页面顶部的**运维**。
   iii. 在**作业列表**区域，单击**作业名称**下的目标作业名。
2. 在**作业运维**页面，单击页面顶部的**血缘关系**。

数据抽样

血缘关系为作业上下游提供了数据抽样功能，该功能和数据开发页面保持一致，方便您在数据运维页面进行随时数据探测，定位问题。开启数据抽样方法如下：

1. 在作业上下游中单击表名。



2. **数据抽样**页面，单击下方的**抽样**。

5.9. 属性参数

属性参数提供了当前作业的详情信息，包括当前运行信息以及历史运行记录。

登录属性参数页面

1. 登录**作业运维**页面。
   i. 登录实时计算控制台。
   ii. 单击页面顶部的**运维**。
   iii. 在**作业列表**区域，单击**作业名称**下的目标作业名。
2. 在**作业运维**页面，单击页面顶部的**属性参数**。

作业代码

您可以在**作业代码**页签，预览SQL作业代码。可单击右上角**编辑作业**，跳转至**开发**页面。

资源配置

- **资源配置**页面为您展示Job运行中所涉及资源的配置信息，如CPU、MEM、并发数等。
- 开启AutoScale功能后，可在资源配置页面中查询AutoScale迭代的历史详情。

> **说明** 仅实时计算3.0.0以上版本支持AutoScale迭代的历史查询功能。

作业属性

运行属性为您展示Job的基本信息。

运行参数

运行参数为您展示包含了底层Checkpoint、启动时间、作业运行在内的作业运行参数信息。

历史记录

历史记录为您展示作业的操作信息，包括**操作人员**、**启动位点**、**终止时间**等。

作业参数

您可以在**作业参数**页面输入实时计算所支持的作业参数，例如，自定义调试阶段分割符的作业参数。

5.10. 作业诊断

实时计算提供作业诊断功能方便您快速的排查作业问题。

> **说明** 仅运行的作业支持诊断功能。

作业诊断步骤

1. 登录实时计算控制台。
2. 单击顶部菜单栏的**运维**，进入**作业运维**界面。
3. 单击作业**操作**列下的**诊断**。

诊断指标

- Failover
  - **作业Failover事件**：检查作业在最近30分钟是否出现Failover。
  - **作业管理节点Failover事件**：监控AM是否出现Failover。
- **Yarn调度事件**为您显示Yarn的检查结果。无异常时，返回结果如下图。

- **机器所在机器环境**为您显示**机器所在机器环境**的检查结果。无异常时，返回结果如下图。

- **Blink Metric**：

  检查并获取作业延时时长。出现延时时，会显示反压的节点。

  - **延时偏高**：延时时长大于100秒小于200秒。
  - **延时过高**：延时时长大于200秒。

6.作业调优

6.1. 概述

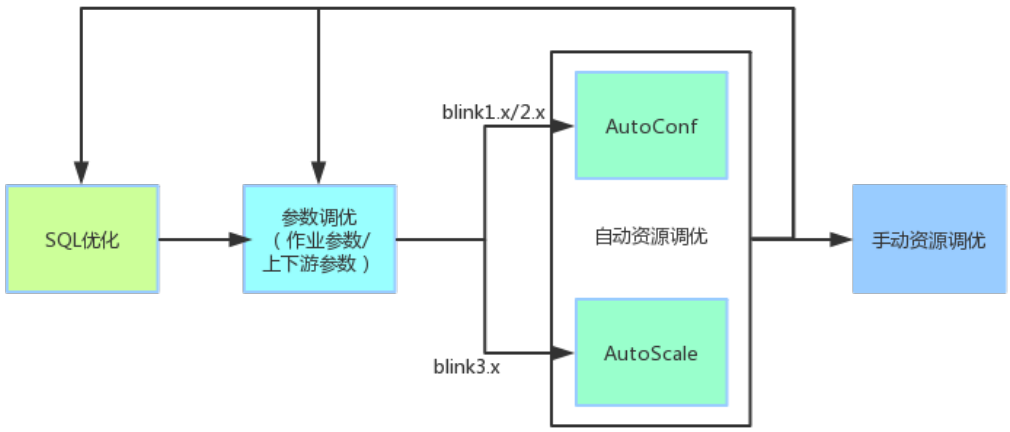
实时计算作业开发过程中，完成了业务逻辑实现、作业上线和启动运行后，为了满足实时计算作业的性能需求，还需对作业进行调优。

作业调优的目的和衡量标准

- 作业正常启动和运行。
- 作业具备合理的延时和吞吐，满足业务性能需求。
- 高效的使用资源，降低成本。

作业调优步骤

建议的作业调优顺序和步骤如下图。



1. SQL优化

   SQL优化即根据业务需求选择合适的SQL实现方式，包括但不限于聚合优化、数据热点优化、TOPN优化、使用内置函数、高效去重、慎用正则函数等，具体请参见高性能Flink SQL优化技巧。

2. 参数调优

   - 作业参数调优

     选择底层优化策略，例如设置minibatch优化State访问策略等，具体请参见作业参数调优。

   - 上下游存储参数调优

     优化上下游存储读写，例如，攒批读写提升吞吐和设置Cache策略提升维表JOIN效率等，具体请参见上下游参数调优。

3. 自动资源调优

   为了简化作业调优，实时计算开发了自动配置调优功能。建议优先通过自动配置调优功能进行作业调优：

- Blink 1.x（自Blink 1.6.4开始）和Blink 2.x版本请参见AutoConf自动配置调优。
- Blink 3.x版本请参见AutoScale自动配置调优。

4. 手动资源调优或再次进行调优
   - 手动资源调优

     自动配置调优无法满足需求的情况下，您可以针对性的进行手动配置调优。

     > **说明** Blink 3.x版本提供反压检测功能协助您判断性能反压点。

   - 再次进行调优

     如果一次调优后的结果不能满足您的业务需求，可按照步骤1~4，再次进行调优。

6.2. 高性能Flink SQL优化技巧

本文为您介绍提升性能的Flink SQL推荐写法、配置及函数。

Group Aggregate优化技巧

- 开启MicroBatch或MiniBatch（提升吞吐）

  MicroBatch和MiniBatch都是微批处理，只是微批的触发机制略有不同。原理同样是缓存一定的数据后再触发处理，以减少对State的访问，从而提升吞吐并减少数据的输出量。

  MiniBatch主要依靠在每个Task上注册的Timer线程来触发微批，需要消耗一定的线程调度性能。MicroBatch是MiniBatch的升级版，主要基于事件消息来触发微批，事件消息会按您指定的时间间隔在源头插入。MicroBatch在元素序列化效率、反压表现、吞吐和延迟性能上都要优于MiniBatch。

  - 适用场景

    微批处理通过增加延迟换取高吞吐，如果您有超低延迟的要求，不建议开启微批处理。通常对于聚合的场景，微批处理可以显著的提升系统性能，建议开启。

    > **说明** MicroBatch模式也能解决两级聚合数据抖动问题。

  - 开启方式

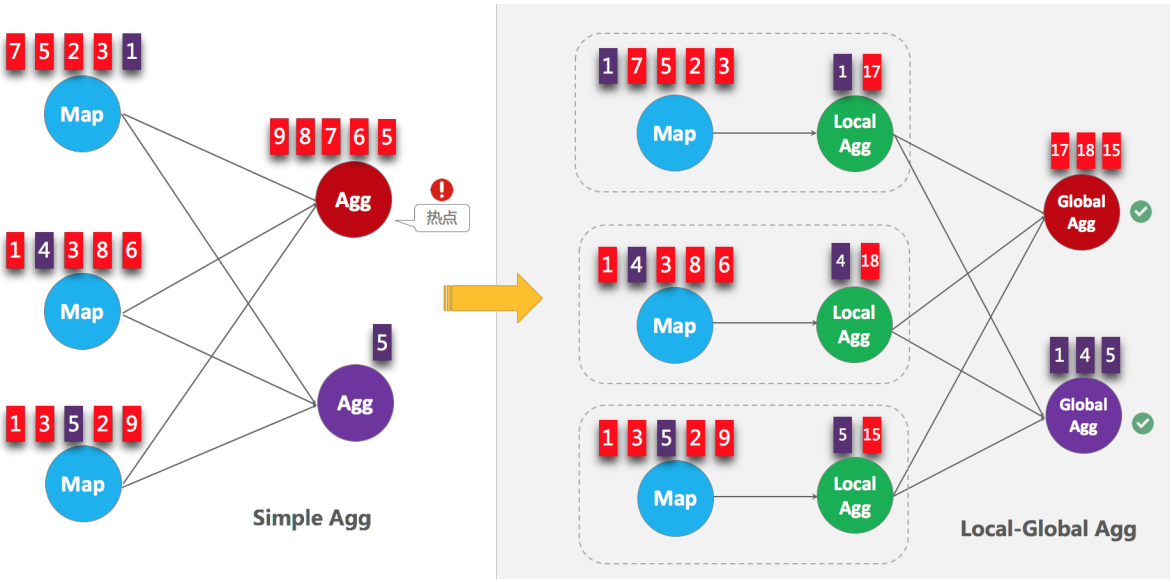
    MicroBatch和MiniBatch默认关闭，开启方式如下。

    ```
    # 3.2及以上版本开启Window miniBatch方法（3.2及以上版本默认不开启Window miniBatch）。
    sql.exec.mini-batch.window.enabled=true
    # 批量输出的间隔时间，在使用microBatch策略时，需要增加该配置，且建议和blink.miniBatch.allowLatencyMs保持一致。
    blink.microBatch.allowLatencyMs=5000
    # 在使用microBatch时，需要保留以下两个miniBatch配置。
    blink.miniBatch.allowLatencyMs=5000
    # 防止OOM设置每个批次最多缓存数据的条数。
    blink.miniBatch.size=20000
    ```

- 开启LocalGlobal（解决常见数据热点问题）

  LocalGlobal优化将原先的Aggregate分成Local+Global两阶段聚合，即MapReduce模型中的Combine+Reduce处理模式。第一阶段在上游节点本地攒一批数据进行聚合（localAgg），并输出这次微批的增量值（Accumulator）。第二阶段再将收到的Accumulator合并（Merge），得到最终的结果（GlobalAgg）。

LocalGlobal本质上能够靠LocalAgg的聚合筛除部分倾斜数据，从而降低GlobalAgg的热点，提升性能。您可以结合下图理解LocalGlobal如何解决数据倾斜的问题。



- 适用场景

  LocalGlobal适用于提升如SUM、COUNT、MAX、MIN和AVG等普通聚合的性能，以及解决这些场景下的数据热点问题。

  > **说明** 开启LocalGlobal需要UDAF实现 `Merge` 方法。

- 开启方式

  实时计算2.0版本开始，LocalGlobal是默认开启的，参数是blink.localAgg.enabled=true，但是需要在microbatch或minibatch开启的前提下才能生效。

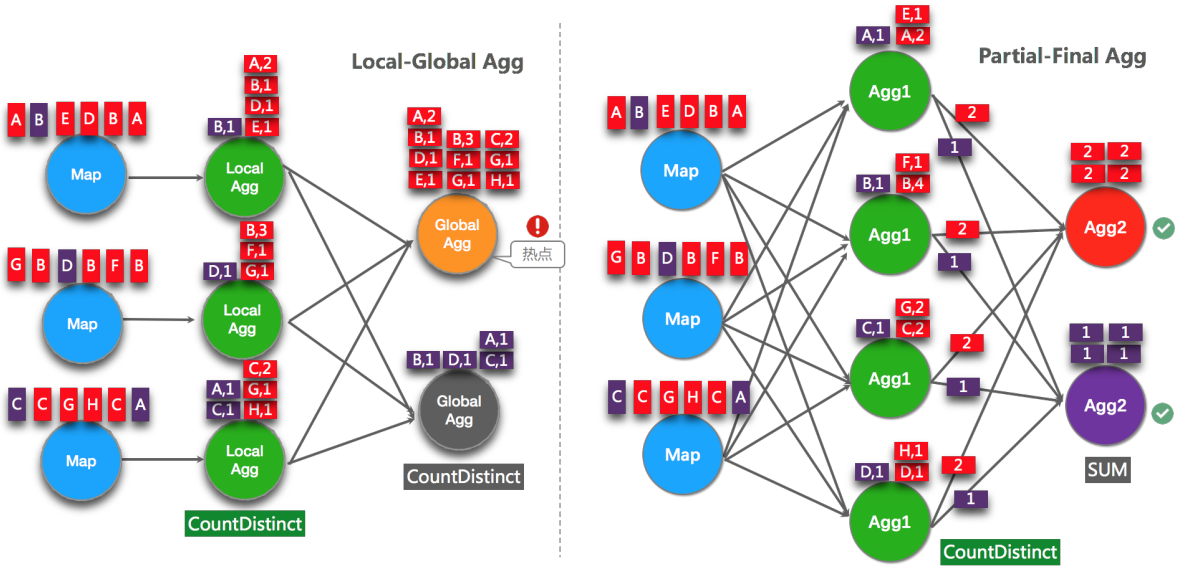
- 判断是否生效

  观察最终生成的拓扑图的节点名字中是否包含**GlobalGroupAggregate**或**LocalGroupAggregate**。

- 开启PartialFinal（解决COUNT DISTINCT热点问题）

  LocalGlobal优化针对普通聚合（例如SUM、COUNT、MAX、MIN和AVG）有较好的效果，对于COUNT DISTINCT收效不明显，因为COUNT DISTINCT在Local聚合时，对于DISTINCT KEY的去重率不高，导致在Global节点仍然存在热点。

之前，为了解决COUNT DISTINCT的热点问题，通常需要手动改写为两层聚合（增加按Distinct Key取模的打散层）。自 `2.2.0` 版本开始，实时计算提供了COUNT DISTINCT自动打散，即PartialFinal优化，您无需自行改写为两层聚合。PartialFinal和LocalGlobal的原理对比参见下图。



- 适用场景

  使用COUNT DISTINCT，但无法满足聚合节点性能要求。

  > **说明**
  > - 不能在包含UDAF的Flink SQL中使用PartialFinal优化方法。
  > - 数据量不大的情况下，不建议使用PartialFinal优化方法。PartialFinal优化会自动打散成两层聚合，引入额外的网络Shuffle，在数据量不大的情况下，浪费资源。

- 开启方式

  默认不开启，使用参数显式开启 `blink.partialAgg.enabled=true` 。

- 判断是否生效

  观察最终生成的拓扑图的节点名中是否包含**Expand**节点，或者原来一层的聚合变成了两层的聚合。

- 改写为AGG WITH FILTER语法（提升大量COUNT DISTINCT场景性能）

  > **说明** 仅实时计算2.2.2及以上版本支持AGG WITH FILTER语法。

  统计作业需要计算各种维度的UV，例如全网UV、来自手机客户端的UV、来自PC的UV等等。建议使用标准的AGG WITH FILTER语法来代替CASE WHEN实现多维度统计的功能。实时计算目前的SQL优化器能分析出Filter参数，从而同一个字段上计算不同条件下的COUNT DISTINCT能共享State，减少对State的读写操作。性能测试中，使用AGG WITH FILTER语法来代替CASE WHEN能够使性能提升1倍。

  - 适用场景

    建议您将AGG WITH CASE WHEN的语法都替换成AGG WITH FILTER的语法，尤其是对同一个字段上计算不同条件下的COUNT DISTINCT结果，性能提升很大。

- 原始写法

```
COUNT(distinct visitor_id) as UV1 , COUNT(distinct case when is_wireless='y' then visitor_id else null end) as UV2
```

- 优化写法

```
COUNT(distinct visitor_id) as UV1 , COUNT(distinct visitor_id) filter (where is_wireless='y') as UV2
```

TopN优化技巧

- TopN算法

  当TopN的输入是非更新流（例如Source），TopN只有一种算法AppendRank。当TopN的输入是更新流时（例如经过了AGG/JOIN计算），TopN有3种算法，性能从高到低分别是：UpdateFastRank 、UnaryUpdateRank和RetractRank。算法名字会显示在拓扑图的节点名字上。

  - UpdateFastRank： 最优算法。

    需要具备2个条件：

    - 输入流有PK（Primary Key）信息，例如ORDER BY AVG。
    - 排序字段的更新是单调的，且单调方向与排序方向相反。例如，ORDER BY COUNT/COUNT_DISTINCT/SUM（正数）DESC（仅实时计算2.2.2及以上版本支持）。

      如果您要获取到优化Plan，则您需要在使用ORDER BY SUM DESC时，添加SUM为正数的过滤条件，确保total_fee为正数。

```
insert
  into print_test
SELECT
  cate_id,
  seller_id,
  stat_date,
  pay_ord_amt  --不输出rownum字段，能减小结果表的输出量。
FROM (
    SELECT
      *,
      ROW_NUMBER () OVER (
        PARTITION BY cate_id,
        stat_date  --注意要有时间字段，否则state过期会导致数据错乱。
        ORDER
          BY pay_ord_amt DESC
      ) as rownum  --根据上游sum结果排序。
    FROM (
        SELECT
          cate_id,
          seller_id,
          stat_date,
          --重点。声明Sum的参数都是正数，所以Sum的结果是单调递增的，因此TopN能使用优化算法，只获取前100个数据。
          sum (total_fee) filter (
            where
              total_fee >= 0
          ) as pay_ord_amt
        FROM
          random_test
        WHERE
          total_fee >= 0
        GROUP BY
          cate_name,
          seller_id,
          stat_date
      ) a
  )
WHERE rownum <= 100;
```

  - UnaryUpdateRank：仅次于UpdateFastRank的算法。需要具备1个条件：输入流中存在PK信息。
  - RetractRank：普通算法，性能最差，不建议在生产环境使用该算法。请检查输入流是否存在PK信息，如果存在，则可进行UnaryUpdateRank或UpdateFastRank优化。

- TopN优化方法
  - 无排名优化

    TopN的输出结果无需要显示rownum值，仅需在最终前端显式时进行1次排序，极大地减少输入结果表的数据量。无排名优化方法详情请参见TopN语句。

- 增加TopN的Cache大小

  TopN为了提升性能有一个State Cache层，Cache层能提升对State的访问效率。TopN的Cache命中率的计算公式为。

  ```
  cache_hit = cache_size*parallelism/top_n/partition_key_num
  ```

  例如，Top100配置缓存10000条，并发50，当您的PatitionBy的key维度较大时，例如10万级别时，Cache命中率只有10000*50/100/100000=5%，命中率会很低，导致大量的请求都会击中State（磁盘），性能会大幅下降。因此当PartitionKey维度特别大时，可以适当加大TopN的CacheSize，相对应的也建议适当加大TopN节点的Heap Memory（请参见手动配置调优）。

  ```
  ##默认10000条，调整TopN cahce到20万，那么理论命中率能达200000*50/100/100000 = 100%。
  blink.topn.cache.size=200000
  ```

- PartitionBy的字段中要有时间类字段

  例如每天的排名，要带上Day字段。否则TopN的结果到最后会由于State ttl有错乱。

高效去重方案

> **说明** 仅Blink 3.2.1版本支持高效去重方案。

实时计算的源数据在部分场景中存在重复数据，去重成为了用户经常反馈的需求。实时计算有保留第一条（Deduplicate Keep FirstRow）和保留最后一条（Deduplicate Keep LastRow）2种去重方案。

- 语法

  由于SQL上没有直接支持去重的语法，还要灵活的保留第一条或保留最后一条。因此我们使用了SQL的ROW_NUMBER OVER WINDOW功能来实现去重语法。去重本质上是一种特殊的TopN。

  ```
  SELECT *
  FROM (
     SELECT *,
      ROW_NUMBER() OVER ([PARTITION BY col1[, col2..]
        ORDER BY timeAttributeCol [asc|desc]) AS rownum
     FROM table_name)
  WHERE rownum = 1
  ```

参数	说明
ROW_NUMBER()	计算行号的OVER窗口函数。行号从1开始计算。
PARTITION BY col1[, col2..]	可选。指定分区的列，即去重的KEYS。
ORDER BY timeAttributeCol [asc|desc])	指定排序的列，必须是一个时间属性的字段（即Proctime或Rowtime）。可以指定顺序（Keep FirstRow）或者倒序（Keep LastRow）。
rownum	仅支持 `rownum=1` 或 `rownum<=1` 。

  如上语法所示，去重需要两层Query：

  i. 使用 `ROW_NUMBER()` 窗口函数来对数据根据时间属性列进行排序并标上排名。

> **说明**
> - 当排序字段是Proctime列时，Flink就会按照系统时间去重，其每次运行的结果是不确定的。
> - 当排序字段是Rowtime列时，Flink就会按照业务时间去重，其每次运行的结果是确定的。

ii. 对排名进行过滤，只取第一条，达到了去重的目的。

> **说明** 排序方向可以是按照时间列的顺序，也可以是倒序：
> - Deduplicate Keep FirstRow：顺序并取第一条行数据。
> - Deduplicate Keep LastRow：倒序并取第一条行数据。

- Deduplicate Keep FirstRow

  保留首行的去重策略：保留KEY下第一条出现的数据，之后出现该KEY下的数据会被丢弃掉。因为STATE中只存储了KEY数据，所以性能较优，示例如下。

```
SELECT *
FROM (
  SELECT *,
    ROW_NUMBER() OVER (PARTITION BY b ORDER BY proctime) as rowNum
  FROM T
)
WHERE rowNum = 1
```

> **说明** 以上示例是将T表按照b字段进行去重，并按照系统时间保留第一条数据。Proctime在这里是源表T中的一个具有Processing Time属性的字段。如果您按照系统时间去重，也可以将Proctime字段简化 `PROCTIME()` 函数调用，可以省略Proctime字段的声明。

- Deduplicate Keep LastRow

  保留末行的去重策略：保留KEY下最后一条出现的数据。保留末行的去重策略性能略优于LAST_VALUE函数，示例如下。

```
SELECT *
FROM (
  SELECT *,
    ROW_NUMBER() OVER (PARTITION BY b, d ORDER BY rowtime DESC) as rowNum
  FROM T
)
WHERE rowNum = 1
```

> **说明** 以上示例是将T表按照b和d字段进行去重，并按照业务时间保留最后一条数据。Rowtime在这里是源表T中的一个具有Event Time属性的字段。

高效的内置函数

- 使用内置函数替换自定义函数

实时计算的内置函数在持续的优化当中，请尽量使用内部函数替换自定义函数。实时计算2.0版本对内置函数主要进行了如下优化：

- 优化数据序列化和反序列化的耗时。
- 新增直接对字节单位进行操作的功能。

- KEY VALUE函数使用单字符的分隔符

  KEY VALUE 的签名：`KEYVALUE(content, keyValueSplit, keySplit, keyName)`，当keyValueSplit和KeySplit是单字符（例如，冒号（:）、逗号（,））时，系统会使用优化算法，在二进制数据上直接寻找所需的keyName 的值，而不会将整个content做切分。性能约提升30%。

- 多KEY VALUE场景使用MULTI_KEYVALUE

  > **说明** 仅实时计算 `2.2.2` 及以上版本支持MULTI_KEYVALUE。

  在Query中对同一个Content进行大量KEY VALUE的操作，会对性能产生很大影响。例如Content中包含10个Key-Value对，如果您希望把10个Value的值都取出来作为字段，您就需要写10个KEY VALUE函数，则系统就会对Content进行10次解析，导致性能降低。

  在这种情况下，建议您使用MULTI_KEYVALUE表值函数，该函数可以对Content只进行一次Split解析，性能约能提升50%~100%。

- LIKE操作注意事项
  - 如果需要进行StartWith操作，使用 `LIKE 'xxx%'` 。
  - 如果需要进行EndWith操作，使用 `LIKE '%xxx'` 。
  - 如果需要进行Contains操作，使用 `LIKE '%xxx%'` 。
  - 如果需要进行Equals操作，使用 `LIKE 'xxx'` ，等价于 `str = 'xxx'` 。
  - 如果需要匹配 `_` 字符，请注意要完成转义 `LIKE '%seller/id%' ESCAPE '/'` 。`_` 在SQL中属于单字符通配符，能匹配任何字符。如果声明为 `LIKE '%seller_id%'` ，则不单会匹配 `seller_id` 还会匹配 `seller#id` 、 `sellerxid` 或 `seller1id` 等，导致结果错误。

- 慎用正则函数（REGEXP）

  正则表达式是非常耗时的操作，对比加减乘除通常有百倍的性能开销，而且正则表达式在某些极端情况下可能会进入无限循环，导致作业阻塞。建议使用LIKE。正则函数包括：

  - REGEXP
  - REGEXP_EXTRACT
  - REGEXP_REPLACE

网络传输的优化

目前常见的Partitioner策略包括：

1. KeyGroup/Hash：根据指定的Key分配。
2. Rebalance：轮询分配给各个Channel。
3. Dynamic-Rebalance：根据下游负载情况动态选择分配给负载较低的Channel。
4. Forward：未Chain一起时，同Rebalance。Chain一起时是一对一分配。
5. Rescale：上游与下游一对多或多对一。

- 使用Dynamic-Rebalance替代Rebalance

Dynamic-Rebalance可以根据当前各Subpartition中堆积的Buffer的数量，选择负载较轻的Subpartition进行写入，从而实现动态的负载均衡。相比于静态的Rebalance策略，在下游各任务计算能力不均衡时，可以使各任务相对负载更加均衡，从而提高整个作业的性能。例如，在使用Rebalance时，发现下游各个并发负载不均衡时，可以考虑使用Dynamic-Rebalance。参数：`task.dynamic.rebalance.enabled=true`，默认关闭。

- 使用Rescale替代Rebalance

  > **说明** 仅实时计算2.2.2及以上版本支持Rescale。

  例如，上游是5个并发，下游是10个并发。当使用Rebalance时，上游每个并发会轮询发给下游10个并发。当使用Rescale时，上游每个并发只需轮询发给下游2个并发。因为Channel个数变少了，Subpartition的Buffer填充速度能变快，能提高网络效率。当上游的数据比较均匀时，且上下游的并发数成比例时，可以使用Rescale替换Rebalance。参数：`enable.rescale.shuffling=true`，默认关闭。

推荐的优化配置方案

综上所述，作业建议使用如下的推荐配置。

```
# EXACTLY_ONCE语义。
blink.checkpoint.mode=EXACTLY_ONCE
# checkpoint间隔时间，单位毫秒。
blink.checkpoint.interval.ms=180000
blink.checkpoint.timeout.ms=600000
# 2.x使用niagara作为statebackend，以及设定state数据生命周期，单位毫秒。
state.backend.type=niagara
state.backend.niagara.ttl.ms=129600000
# 2.x开启5秒的microbatch。
blink.microBatch.allowLatencyMs=5000
# 整个Job允许的延迟。
blink.miniBatch.allowLatencyMs=5000
# 单个batch的size。
blink.miniBatch.size=20000
# local 优化，2.x默认已经开启，1.6.4需手动开启。
blink.localAgg.enabled=true
# 2.x开启PartialFina优化，解决COUNT DISTINCT热点。
blink.partialAgg.enabled=true
# union all优化。
blink.forbid.unionall.as.breakpoint.in.subsection.optimization=true
# object reuse优化，默认已开启。
#blink.object.reuse=true
# GC优化（SLS做源表不能设置该参数）。
blink.job.option=-yD heartbeat.timeout=180000 -yD env.java.opts='-verbose:gc -XX:NewRatio=3
-XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:ParallelGCThreads=4'
# 时区设置。
blink.job.timeZone=Asia/Shanghai
```

6.3. AutoConf自动配置调优

为了增加用户的体验度，阿里云为您提供自动配置调优（AutoConf）功能。

> **说明** AutoConf自动配置调优功能支持blink 1.0和2.0版本。

背景及功能范围

在您作业的各个算子和流作业上下游性能达标和稳定的前提下，自动配置调优功能可以帮助您更合理的分配各算子的资源和并发度等配置。 全局优化您的作业，调节作业吞吐量不足、作业全链路的反压等性能调优的问题。

出现下列情况时，自动配置调优功能可以作出优化，但无法彻底解决流作业的性能瓶颈，需要您自行解决或联系实时计算产品支持团队解决性能瓶颈。

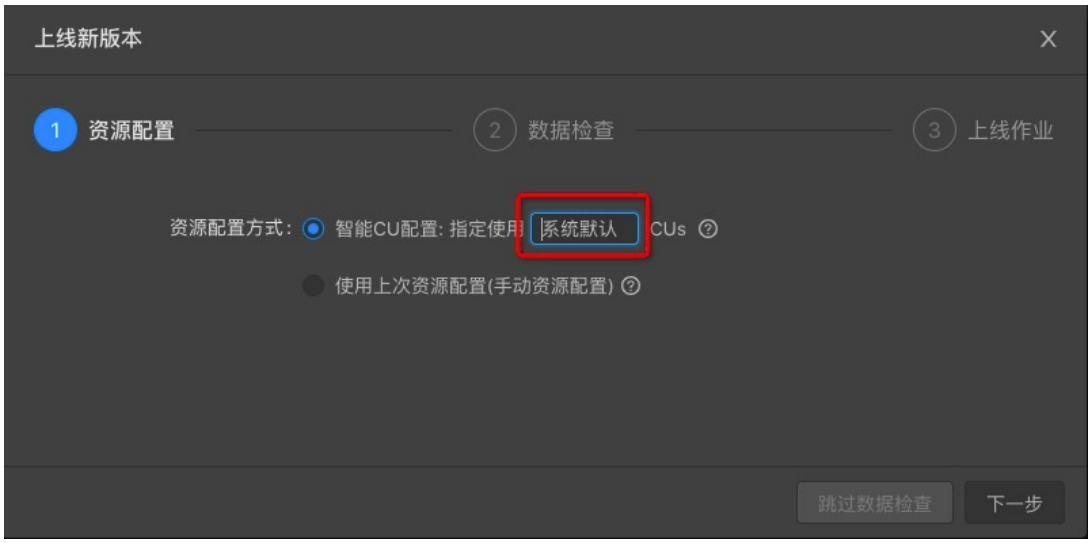
- 流作业上下游有性能问题。
  - 流作业上游的数据Source存在性能问题。例如，DataHub分区不足、MQ吞吐不够等需要您扩大相应Source的分区。
  - 流作业下游的数据Sink存在性能问题。例如，RDS死锁等。
- 流作业的自定义函数 （UDF、UDAF和UDTF） 有性能问题。

操作

- 新作业
  i. 上线作业
     a. 完成SQL开发，通过**语法检查**后，单击**上线**，即可出现如下**上线新版本**界面。

     b. 选择**资源配置**方式。第一次不需要指定CU数直接使用系统默认配置。
        - **智能CU配置**：指定使用CU AutoConf算法会基于系统默认配置，生成CU数，进行优化资源配置。如果是第一次运行，算法会根据经验值生成一份初始配置。建议作业运行了5~10分钟以上，确认Source RPS等Metrics稳定2~3分钟后，再使用智能配置，重复3~5次才能调优出最佳的配置。
        - **使用上次资源配置（手动资源配置）**：即使用最近一次保存的资源配置。如果上一次是智能配置的，就使用上一次智能配置的结果。如果上一次是手工配置的，就使用上次手工配置的结果。
  ii. 使用默认配置启动作业

a. 使用默认配置启动作业，出现如下的界面。



b. **启动**作业。

示例如下。第一次默认配置生成的资源配置为71个CU。

> **说明** 请您确保作业已经运行10分钟以上，并且Source RPS等数据曲线稳定2-3分钟后，再使用智能配置。

iii. 使用智能配置启动作业

a. 资源调优

例如，您手动配置40CU，使用智能配置启动。40的CU数是您自行指定调整的，您可以根据具体的作业情况适当的增加或者是减小CU数，来达到资源调优的目的。

- CU的最小配置

  CU的最小配置建议不小于默认配置总数的50%，CU数不能小于1CU。假设智能配置默认CU数为71，则建议最小CU数为36CU。 71*50% = 35.5CU。

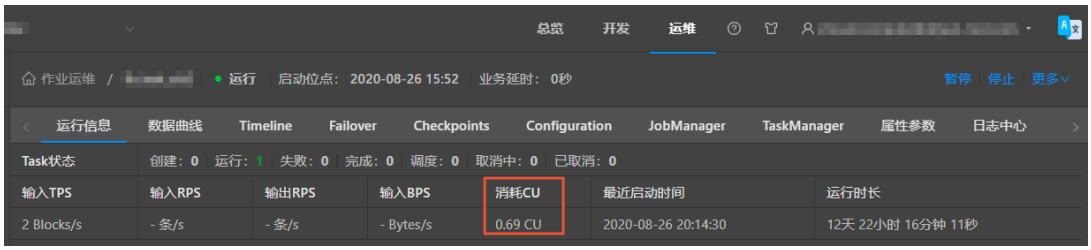
- CU增加数量

  假如无法满足作业理想的吞吐量就需要增加适量的CU数。每次增加的CU数，建议是上一次CU总数的30%以上。例如，上一次配置是10CU，下次就需要增加到13CU。

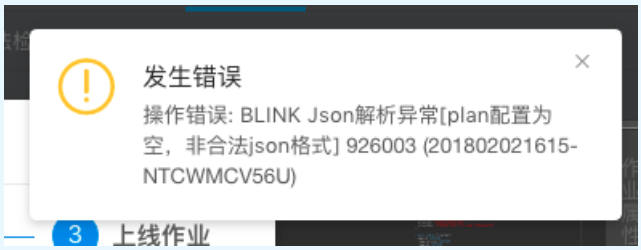
- 可多次调优

  如果第一次调优不满足您的需求，可以调优多次。可以根据每次调优后Job的状态来增加或减少资源数。

b. 调优后的结果如下图。

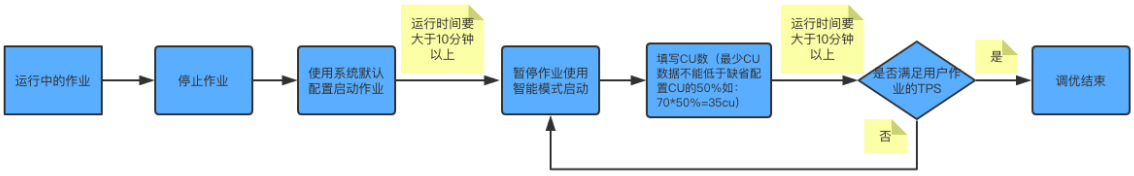


> **说明** 如果是新任务，请不要选择**使用上次资源配置**，否则会报错。



- 已存在作业

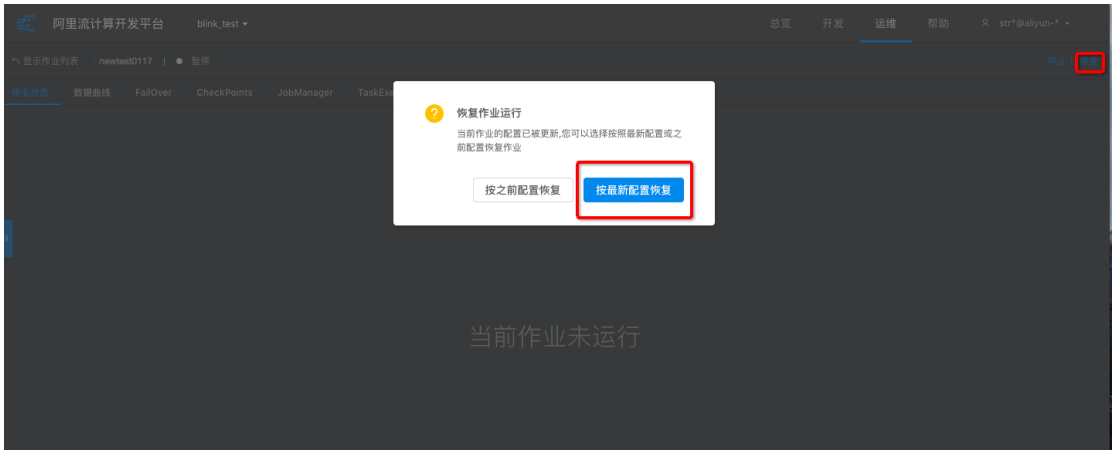
- 调优流程示意图



> **说明**
>
> - 已存在的作业在进行调优前，您一定要检查是否是有状态的计算。因为在调优的过程中可能会清除之前作业保存的状态，请您谨慎操作。
> - 当您的作业有改动时（例如，您对作业的SQL有修改，或更改了实时计算版本），自动配置调优功能不保证能按照之前的运行信息进行调优。原因是这些改动会导致拓扑信息变化，造成数据曲线无法对应和状态无法复用等问题。自动配置调优功能无法根据运行历史信息作出调优判断。此时再使用自动配置调优会报异常。这时您需要将改动后的任务当做新任务，重新进行操作。

- 调优流程
  a. 暂停任务。

  b. 重复新作业的调优步骤，使用最新的配置启动作业。



常见问题

以下几点可能会影响自动调优的准确性：

- 任务运行的时间较短，会造成采样得到的有用信息较少，会影响AutoConf算法的效果。建议延长运行时间，确认Source RPS等数据曲线稳定2~3分钟后即可。
- 任务运行有异常（Failover），会影响结果的准确性。建议用户检查和修复Failover的问题。
- 任务的数据量比较少，会影响结果的准确性。建议回追足够多的历史数据。
- 影响的因素有很多，自动调优AutoConf不能保证下一次生成的配置一定比上一次的好。如果还不能满足需求，用户参考手动配置调优，进行手动调优。

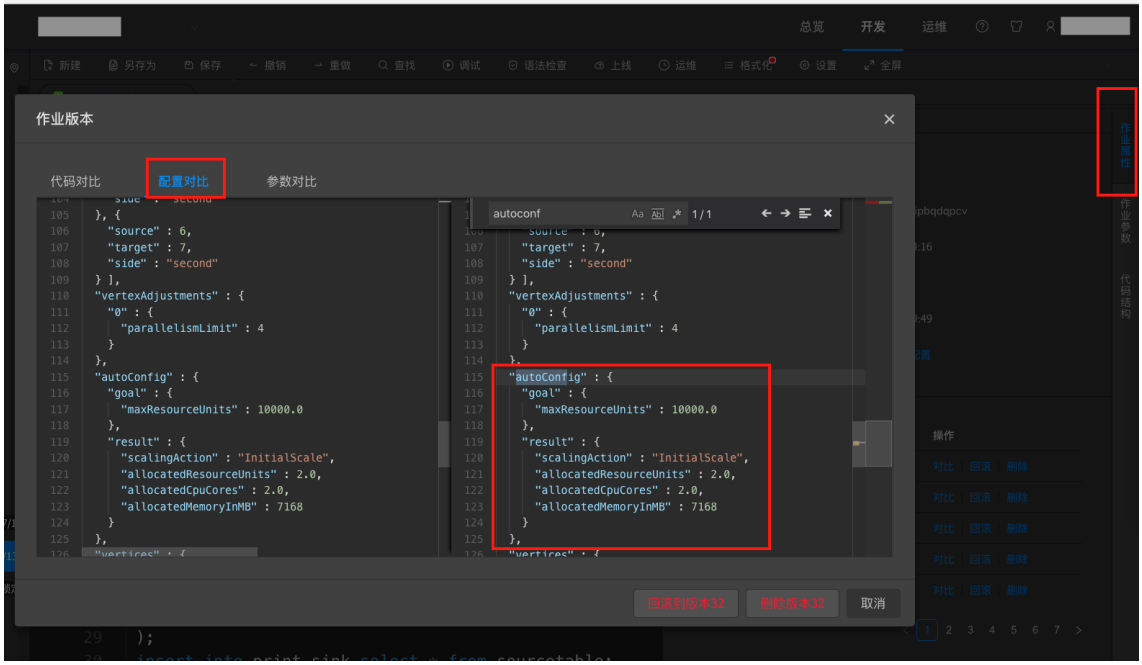
调优建议

- 每次触发智能配置前任务稳定运行超过10分钟。这样有利于AutoConf准确搜集的任务运行时的指标信息。
- AutoConf可能需要3~5次迭代才能见效。
- 使用AutoConf时，您可以设置让任务回追数据甚至造成反压。这样会更有利于快速体现调优成功。

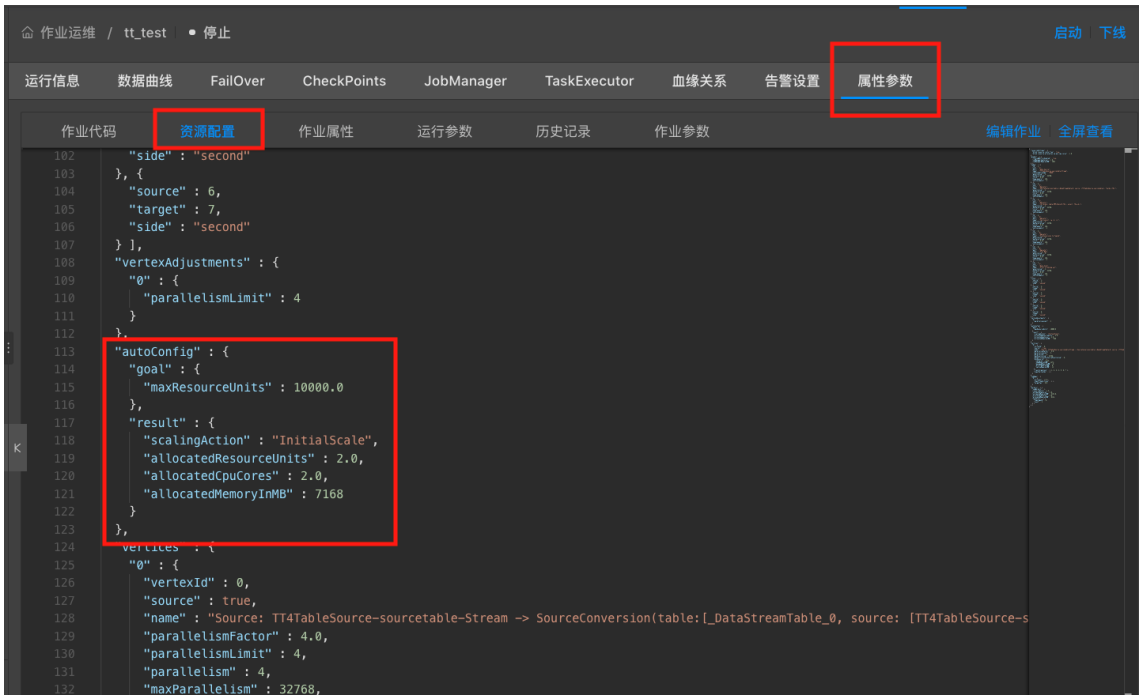
如何判断自动配置调优功能生效或出现问题？

自动配置调优功能通过JSON配置文件与实时计算交互。您在调优后，可以通过查看JSON配置文件了解自动配置调优功能的运行情况。

- 查看JSON配置文件的两种方式
  i. 通过作业编辑界面，如下图。



  ii. 通过作业运维界面，如下图。



- JSON配置解释

```
"autoconfig" : {
    "goal": {  // AutoConf 目标
        "maxResourceUnits": 10000.0,  // 单个Blink作业最大可用CU数，不能修改，查看时可忽略。
        "targetResoureUnits": 20.0  // 用户指定CU数。用户指定为20CU。
    },
    "result" : {  // AutoConf 结果。这里很重要
      "scalingAction" : "ScaleToTargetResource",  // AutoConf的运行行动 *
      "allocatedResourceUnits" : 18.5, // AutoConf分配的总资源。
      "allocatedCpuCores" : 18.5,      // AutoConf分配的总CPU。
      "allocatedMemoryInMB" : 40960    // AutoConf分配的总内存。
      "messages" : "xxxx"  // 很重要。 *
    }
}
```

- scalingAction： `InitialScale` 代表初次运行， `ScaleToTargetResource` 代表非初次运行。
- 如果没有messages，代表运行正常。如果有messages，代表需要分析：messages有两种，如下所示：
  - warning 提示：表示正常运行情况但有潜在问题，需要用户注意，如source的分区不足等。
  - error或者exception提示，常伴有 `Previous job statistics and configuration will be used` ，代表AutoConf失败。失败也有两种原因：
    - 用户作业或blink版本有修改，AutoConf无法复用以前的信息。
    - 有exception代表AutoConf遇到问题，需要跟据信息、日志等综合分析。如没有足够的信息，请提交工单。

异常信息问题

IllegalStateException异常

出现如下的异常说明内部状态state无法复用，需要**停止**任务清除状态后重追数据。

如果无法切到备链路，担心对线上业务有影响，可以在**开发**界面右侧的**作业属性**里面选择上个版本进行回滚，等到业务低峰期的时候再重追数据。

```
java.lang.IllegalStateException: Could not initialize keyed state backend.
    at org.apache.flink.streaming.api.operators.AbstractStreamOperator.initKeyedState(Abstr
actStreamOperator.java:687)
    at org.apache.flink.streaming.api.operators.AbstractStreamOperator.initializeState(Abst
ractStreamOperator.java:275)
    at org.apache.flink.streaming.runtime.tasks.StreamTask.initializeOperators(StreamTask.j
ava:870)
    at org.apache.flink.streaming.runtime.tasks.StreamTask.initializeState(StreamTask.java:
856)
    at org.apache.flink.streaming.runtime.tasks.StreamTask.invoke(StreamTask.java:292)
    at org.apache.flink.runtime.taskmanager.Task.run(Task.java:762)
    at java.lang.Thread.run(Thread.java:834)
Caused by: org.apache.flink.api.common.typeutils.SerializationException: Cannot serialize/d
eserialize the object.
    at com.alibaba.blink.contrib.streaming.state.AbstractRocksDBRawSecondaryState.deseriali
zeStateEntry(AbstractRocksDBRawSecondaryState.java:167)
    at com.alibaba.blink.contrib.streaming.state.RocksDBIncrementalRestoreOperation.restore
RawStateData(RocksDBIncrementalRestoreOperation.java:425)
    at com.alibaba.blink.contrib.streaming.state.RocksDBIncrementalRestoreOperation.restore
(RocksDBIncrementalRestoreOperation.java:119)
    at com.alibaba.blink.contrib.streaming.state.RocksDBKeyedStateBackend.restore(RocksDBKe
yedStateBackend.java:216)
    at org.apache.flink.streaming.api.operators.AbstractStreamOperator.createKeyedStateBack
end(AbstractStreamOperator.java:986)
    at org.apache.flink.streaming.api.operators.AbstractStreamOperator.initKeyedState(Abstr
actStreamOperator.java:675)
    ... 6 more
Caused by: java.io.EOFException
    at java.io.DataInputStream.readUnsignedByte(DataInputStream.java:290)
    at org.apache.flink.types.StringValue.readString(StringValue.java:770)
    at org.apache.flink.api.common.typeutils.base.StringSerializer.deserialize(StringSerial
izer.java:69)
    at org.apache.flink.api.common.typeutils.base.StringSerializer.deserialize(StringSerial
izer.java:28)
    at org.apache.flink.api.java.typeutils.runtime.RowSerializer.deserialize(RowSerializer.
java:169)
    at org.apache.flink.api.java.typeutils.runtime.RowSerializer.deserialize(RowSerializer.
java:38)
    at com.alibaba.blink.contrib.streaming.state.AbstractRocksDBRawSecondaryState.deseriali
zeStateEntry(AbstractRocksDBRawSecondaryState.java:162)
    ... 11 more
```

6.4. AutoScale自动配置调优

为了解决您使用AutoConf自动配置调优功能时频繁启停作业的问题，实时计算3.0及以上版本提供了AutoScale自动配置调优功能。作业启动后，系统会根据资源配置规则，自动进行作业的调优，直到满足设定的调优目标，全程无需人工介入。

> **说明**
>
> - AutoScale自动配置调优功能仅支持实时计算3.0及以上版本。
> - 升级实时计算3.0版本前，需要删除所有低于3.0版本的PlanJson，并重新获取配置资源。详情请参见重新获取配置资源。

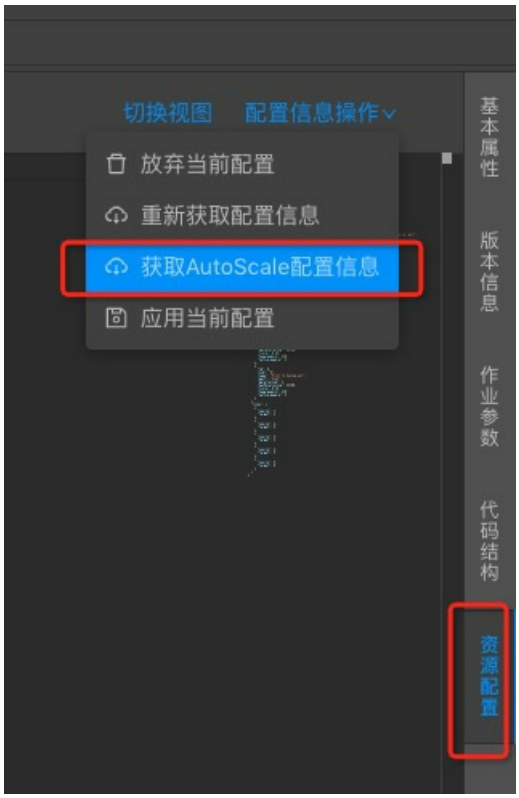
启动AutoScale自动配置调优

作业上线的过程中即可完成AutoScale自动配置调优的开启。

1. 登录作业编辑页面。
   i. 登录实时计算控制台。
   ii. 单击页面顶部的**开发**。
   iii. 在**作业开发**区域，双击目标文件夹或目标作业名，进入作业编辑页面。
2. 单击页面顶部菜单栏中的**上线**，进入**上线新版本**页面。
3. 在**初始资源**页面，选择初始资源类型，单击**下一步**。

- **上次自动调优**：使用上次AutoScale的PlanJson配置启动作业。满足以下所有条件后即可选择**上次自动调优**功能。
  - 作业已开启AutoScale并完成迭代。
  - 作业为**暂停**状态。

- 在**资源配置**页面**获取AutoScale配置信息**。



  - **系统分配**：新增作业或者作业逻辑没有修改且兼容实时计算版本时，可以选择**系统分配**。
  - **手动资源配置**：使用手动生成的资源配置方法。在手动重新生成或者修改AutoScale时，需要选择**手动资源配置**。

4. 在**数据检查**，通过检查后，单击**下一步**。
5. 在**资源配置**，配置**自动调优**信息后，单击**下一步**。

参数	说明
自动调优	选择**开启**，开启自动调优功能。
planJson最大CU数	作业可用的资源上限，单位为CU，1CU=1 Core + 4G MEM。最大CU数需小于项目可用CU数。
调优策略	目前**调优策略**仅支持**数据滞留时间**。系统会根据选择的**调优策略**和**期望值**对作业自动调优。
期望值	数据滞留时间阈值。当数据源的数据滞留时间超过该值时，触发AutoScale，优化作业并发数。

> **说明** 例如，调优策略设置数据滞留时间的期望值为5（秒）。如果此时作业的数据滞留时间大于5秒，则系统会不断进行自动调优，直至数据滞留时间小于5秒。

6. 在**上线作业**，单击**上线**。
7. 启动作业。详情请参见启动。

关闭AutoScale

> **说明** 在启动作业时，需要开启AutoScale自动调优功能后，才可以进行关闭AutoScale自动调优操作。

您可以在作业运行期间，动态关闭AutoScale自动调优功能。

1. 登录作业**运维**页面。
   i. 登录实时计算控制台。
   ii. 单击页面顶部的**运维**。
   iii. 在**作业列表**区域，单击**作业名称**下的目标作业名。
2. 单击页面右上角的**关闭自动调优**。
3. 单击**确认**。

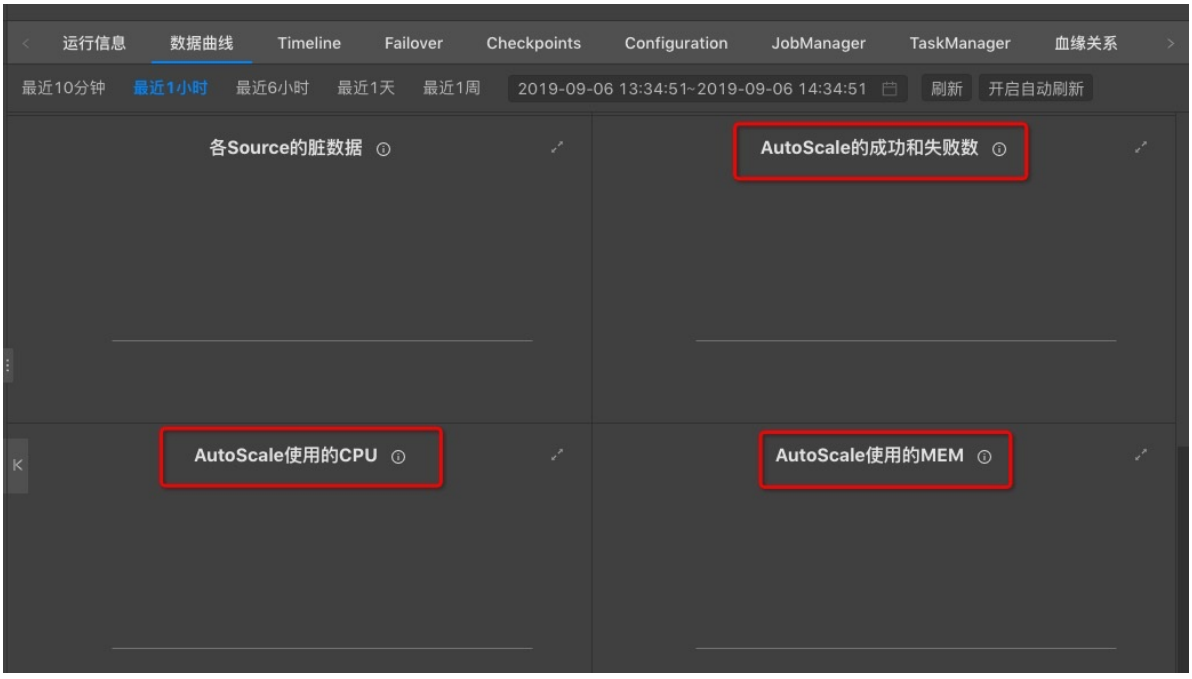
> **说明** 作业在上线时启动了AutoScale后，**运维**界面**自动调优**列下的操作按键才能生效。

查看AutoScale效果

> **说明** 请登录**作业运维**界面进行查看。**作业运维**页面登录步骤如下：
> 1. 登录实时计算控制台。
> 2. 单击页面顶部的**运维**。
> 3. 在**作业列表**区域，单击**作业名称**下的目标作业名。

- AutoScale Metric

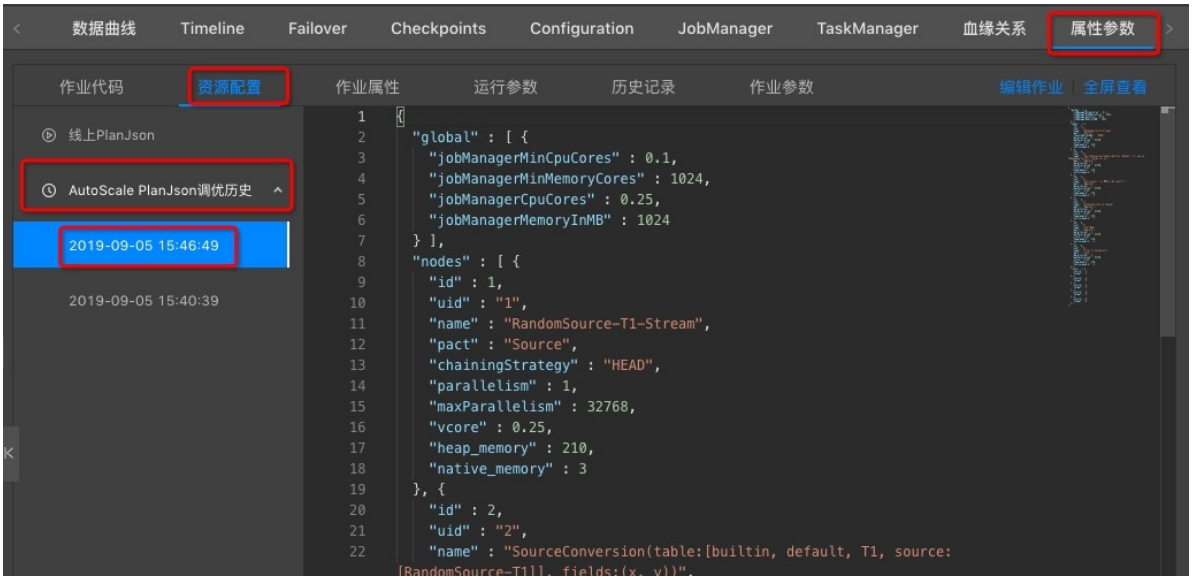
  在**数据曲线** > **OverView**页面，可以查看AutoScale的信息。



曲线名称	说明
AutoScale的成功和失败数	AutoScale成功和未成功执行的次数。
AutoScale使用的CPU	执行AutoScale时消耗的CPU量。
AutoScale使用的MEM	执行AutoScale时消耗的内存量。

- AutoScale PlanJson信息

  在**属性参数 > 资源配置 > AutoScale PlanJson调优历史**，单击目标版本，查看AutoScale生成的PlanJson的信息。



常见问题

- Q：作业启动时产生报错：`ERR_ID：CLI-00000001` 。

  A：参见报错：CLI-00000001。

- Q：作业上线时参数校验报错：`resource validate failed! please modify your planJson or max CU and try again.` 。

  A：以下2种场景可能导致该报错：

  - PlanJson资源文件和实际作业不匹配
    - 报错原因：PlanJson资源文件和实际作业不匹配，导致能够关联（Chain）在一起的节点没有关联在一起，使作业所需资源变大。
    - 解决方案：重新生成配置信息，操作步骤参见报错：CLI-00000001。
  - 配置的可用资源过小
    - 报错原因：作业所需的CPU或者内存中的一个资源超过最大资源。
    - 解决方案：返回**上一步**，增加最大可用资源的CU数。

- Q：AutoScale没有启动。

  A： 排查方法如下：

- 检查作业是否频繁运行失败（Failover）。AutoScale开启的前提是作业本身逻辑正确并且能够稳定运行。
- 查看**JobManager**的相关日志，检查是否存在系统异常。

- Q：AutoScale没有效果。

  A：请按照以下顺序进行排查：

  i. 检查最大资源限制，确认作业已使用资源是否达到最大资源限制。

  ii. 检查作业数据源（Source）节点的逻辑，查看Source节点是否关联（Chain）过多的Operator。如果关联过多Operator，请手动编辑PlanJson，在一些关键的位置进行关联（Chain）拆分。解决方案参见手动配置调优。

  iii. 查看**JobManager**的相关日志，检查是否存在系统异常。

- Q：AutoScale可能出现的问题有哪些？

  A：

  - 作业自动重启

    AutoScale根据实际流量来动态调整并发数以及资源，因此可能会在流量上涨以及流量下降时，通过自动重启触发资源调整。

  - 短时间内出现数据传输延迟

    流量低谷时AutoScale会降低并发数、减少资源分配。当流量上涨后，流量低谷时的资源配置会导致吞吐能力不够和数据传输的延迟。

  - 作业无法恢复

    在部分场景下AutoScale无法有效工作，导致作业延迟和作业频繁的进行配置调整。

  - 并发数先减少后增多

    包含窗口函数或聚合函数的作业，在状态（State）数量增加时，访问State的性能衰减，作业启动时的并发数减少。随着数据积累，并发数会逐渐增大，直至State数量达到稳定。

6.5. 手动配置调优

您可以手动调整实时计算Flink版的作业参数、资源参数和上下游参数，提升实时计算Flink版作业的性能。

> **说明** 建议您完成作业反压检测后，再判断是否需要进行配套调优。实时计算Flink版3.0以上版本作业反压检测方法，详情请参见如何对实时计算Flink版3.0以上版本的作业进行反压检测？。

手动配置调优概述

手动配置调优的内容主要有3种类型：

- 上下游参数调优：对作业中上下游存储的参数进行调优。
- 作业参数调优：对作业中miniBatch等参数进行调优。
- 资源参数调优：对作业中Operator的并发数（Parallelism）、CPU（Core）和堆内存（Heap Memory）等参数进行调优。

下面对以上3类调优进行介绍。参数调优后将生成新的配置，作业需要先停止再启动或者先暂停再恢复后才能使用新的配置，启动新配置的方法请参见重新启用新的配置。

上下游参数调优

实时计算Flink版每条数据均会触发上下游存储的读写，因此会对上下游存储形成存储压力。设置batchsize，批量读写上下游存储数据可以降低上下游存储的压力。支持batchsize参数的上下游存储如下表所示。

上下游	参数	说明	设置参数值
DataHub源表	batchReadSize	单次读取的条数	可选，默认值为10。
DataHub结果表	batchSize	单次写入的条数	可选，默认值为300。
日志服务LOG源表	batchGetSize	单次读取logGroup的条数	可选，默认值为100。
分析型数据库MySQL版2.0结果表	batchSize	一次批量写入的条数	可选，默认值为1000。
云数据库RDS版结果表	batchSize	一次批量写入的条数	可选，默认值为4096。

说明 DDL的WITH参数列中增加batchSize相关参数，即可完成数据的批量读写设置。例如，`batchReadSize='<number>'`。

作业参数调优

miniBatch设置仅适用于优化GROUP BY。Flink SQL流模式下，每来一条数据都会执行State操作，I/O消耗较大。设置miniBatch后，同一个Key的一批数据只访问一次State，且只输出最新的一条数据，既减少了State的访问，也减少了下游的数据更新。miniBatch设置说明如下：

- 新增加作业参数后，建议您停止作业，再启动作业。
- 更新作业参数值后，建议您暂停作业，再恢复作业。

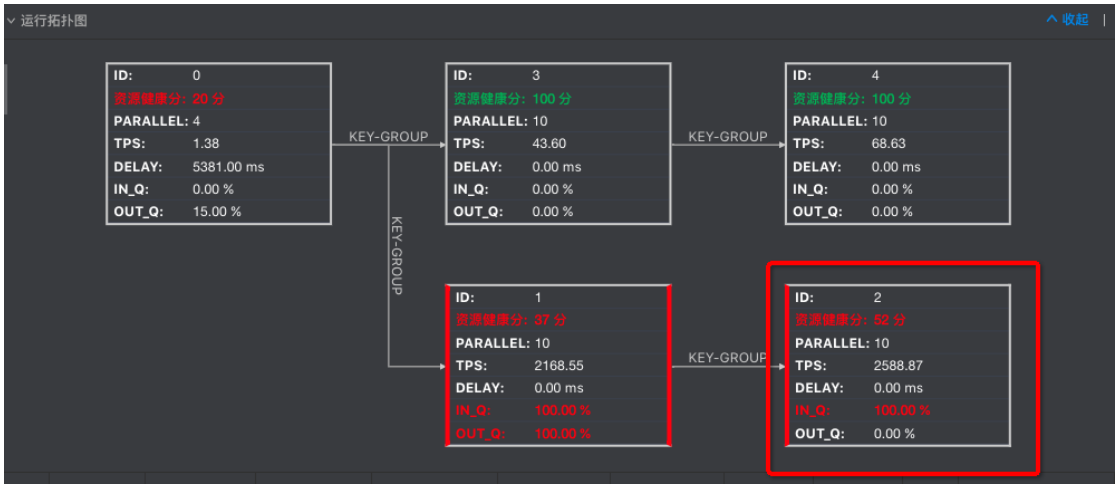
```
# 3.2及以上版本开启window miniBatch方法（3.2及以上版本默认不开启window miniBatch）。
sql.exec.mini-batch.window.enabled=true
# excatly-once语义。
blink.checkpoint.mode=EXACTLY_ONCE
# checkpoint间隔时间，单位毫秒。
blink.checkpoint.interval.ms=180000
blink.checkpoint.timeout.ms=600000
# 实时计算Flink版2.0及以上版本使用niagara作为statebackend，以及设定state数据生命周期，单位毫秒。
state.backend.type=niagara
state.backend.niagara.ttl.ms=129600000
# 实时计算Flink版2.0及以上版本开启5秒的microbatch（窗口函数不需要设置该参数）。
blink.microBatch.allowLatencyMs=5000
# 表示整个Job允许的延迟。
blink.miniBatch.allowLatencyMs=5000
# 双流join节点优化参数。
blink.miniBatch.join.enabled=true
# 单个Batch的size。
blink.miniBatch.size=20000
# local优化，实时计算Flink版2.0及以上版本默认已经开启，1.6.4版本需要手动开启。
blink.localAgg.enabled=true
# 实时计算Flink版2.0及以上版本开启partial优化，解决count distinct效率低问题。
blink.partialAgg.enabled=true
# union all优化。
blink.forbid.unionall.as.breakpoint.in.subsection.optimization=true
# GC优化（源表为SLS时，不能设置该参数）。
blink.job.option=-yD heartbeat.timeout=180000 -yD env.java.opts='-verbose:gc -XX:NewRatio=3
-XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:ParallelGCThreads=4'
# 时区设置。
blink.job.timeZone=Asia/Shanghai
```

资源调优

下文通过示例为您介绍资源调优方法：

1. 分析过程

   i. 通过Job的拓扑图查看到2号的Task节点的输入队列已达到100%，造成上游1号节点的数据堆积，输出队列造成数据堆积。



   ii. 单击目标Task节点（本示例为2号Task节点）。

iii. 在**SubTask List**，查看**In Queue**为 `100%` 的行。

iv. 单击目标行左侧的**LOG0**。

v. 在**Metrics Graph**页签，查看CPU和MEM的使用情况。

2. 性能调优

i. 单击作业编辑页面右侧的**资源配置**，进入调优窗口。

ii. 对目标Operator或者Group进行参数修改。

- 单个Operator参数修改
  a. **GROUP**框内，单击右上角的加号（+）。
  b. 鼠标悬停至目标Operator框内。
  c. 单击目标Operator右侧的  图标。
  d. 在**修改Operator数据**页面，修改参数。
- GROUP批量参数修改
  a. 鼠标悬停至**GROUP**框内。
  b. 单击**GROUP**右侧的  图标。
  c. 在**批量修改Operator数据**页面，修改参数。

iii. 完成配置参数修改后，单击**资源配置**右上角的**配置信息操作 > 应用当前配置**。

- 如果增加了Group的资源配置后，作业的运行效率提升不明显，需要按照以下顺序分析问题：
  a. 该节点是否存在数据倾斜现象。
  b. 复杂运算的Operator节点（例如，GROUP BY、WINDOW和JOIN等）的子节点是否存在异常。
- Operator子节点拆解方法：
  a. 单击目标Operator。
  b. 修改chainingStrategy参数值为 `HEAD` 。

  如果chainingStrategy已设置为 `HEAD` ，需要修改后一个Operator的chainingStrategy参数值为 `HEAD` 。chainingStrategy参数说明如下。

参数	说明
ALWAYS	算子会尽可能地链接在一起。为了优化性能，通常需要让算子尽可能链接在一起，同时增加并发度。
NEVER	算子不会和上下游的算子链接在一起。
HEAD	算子不会和上游的算子链接在一起，但是会和下游的算子链接在一起。

3. 资源参数的配置原则和建议

- 作业的资源配置建议为： `core:heap_memory=1:4` ，即1个核对应4 GB内存。例如：
  - core参数值为1CU，heap memory参数值为3 GB，则最终资源分配结果的是1CU+4G。
  - core参数值为1CU，heap memory参数值为5 GB，则最终资源分配结果的是1.25CU+5G。

> **说明**
> - Operator的总core=parallelism*core。
> - Operator的总heap_memory=parallelism*heap_memory。
> - Group中的core取各个Operator中最大值，memory取各个Operator中memory之和。

- **parallelism**
  - Source节点

    Source的个数和上游Partition数量有关。例如，Source的个数是16，Source的并发数可以为16、8或4等，不得超过16。

    > **说明** Source的并发数不能大于Source的Shard数。

  - 中间节点

    根据预估的QPS计算：
    - QPS低的任务，中间处理节点数和Source并发数一样。
    - QPS高的任务，中间处理节点数配置为比Source并发数大，例如，64、128或256。

  - Sink节点

    并发度和下游存储的Partition数有关，一般是下游Partition个数的2~3倍。

    > **说明** 如果配置过大会导致输入超时或失败。例如，下游Sink节点个数是16，建议Sink的并发数最大为48。

- **core**

  默认值为0.1，根据实际CPU使用配置，建议配置为0.25。
- **heap_memory**

  堆内存，默认为值为256（单位为MB），请根据实际的内存使用状况进行配置。
- **state_size**

  存在GROUP BY、undefinedJOIN、OVER和WINDOW的Task节点中需要配置参数state_size为`1`，表示该Operator会使用state功能，作业会为该Operator申请额外的内存。state_size默认值为0。

  > **说明** 如果state_size参数值不设置为`1`，则作业可能运行失败。

重新启用新的配置

完成配置后，建议您采用暂停再恢复作业的方式使新配置生效，而不是停止再启动作业的方式使新配置生效。因为停止作业后，作业状态会被清除，从而可能导致计算结果不一致。

> **说明**
> - 暂停再恢复：仅更改资源配置、调整WITH参数大小或调整作业参数大小。
> - 停止再启动：需要更改SQL逻辑、更改作业版本、增加WITH参数或增加作业参数。

完成作业重启或恢复后，可以在**运维**页面，**运行信息**页签下的**Vertex拓扑**中查看新的配置是否生效。

- 暂停再恢复步骤如下：
  i. 上线作业。上线步骤参见上线，详情请参见上线。**配置方式**选择**使用上次资源配置（手动资源配置）**。
  ii. 在**运维**页面，单击目标作业操作列下的**暂停**。
  iii. 在**运维**页面，单击目标作业操作列下的**恢复**。
  iv. 在**恢复作业运行**，单击**按最新配置恢复**。

- 停止再启动步骤如下：
  i. 停止作业。停止作业步骤请参见停止。
  ii. 启动作业。启动作业步骤参见启动。

相关参数说明

- Global

  isChainingEnabled：是否启用Chain策略，默认为true。不需要修改。

- Nodes

参数	说明	能否修改
id	节点ID号，自动生成，ID号唯一。	不能
uid	节点UID号，用于计算Operator ID，如果不设置，则使用ID。	不能
pact	节点类型，例如，Data Source、Operator或Data Sink等。	不能
name	节点名称，可自定义。	能
slotSharingGroup	请保持默认配置。	不能

参数	说明	能否修改
chainingStrategy	ChainingStrategy用来定义算子链接的策略。当一个算子和上游算子链接在一起，表示它们会运行在同一个线程，合并为一个有多个运行步骤的算子。 ChainingStrategy支持以下3种方式： ○ ALWAYS：算子会尽可能的链接在一起。为了优化性能，通常最佳实践是让算子尽可能链接在一起，同时增加并发度。 ○ NEVER：算子不会和上下游的算子链接在一起。 ○ HEAD：算子不会和上游的算子链接在一起，但是会和下游的算子链接在一起。	能
parallelism	并发数，默认值为 1，可以根据实际数据量增大并发数。	能
core	CPU，默认为0.1，可以根据实际CPU使用情况进行配置。建议设置为0.25。	能
heap_memory	堆内存，默认为256（单位为MB），可以根据实际内存使情况进行配置。	能
direct_memory	JVM（Java虚拟机）堆外内存，默认0（单位为MB）。	能，但不建议您修改该参数。
native_memory	JVM（Java虚拟机）堆外内存，JNI（Java Native Interface）中使用，默认为0。如果需要，可以配置为10，主要提供给statebackend使用。	能，但不建议您修改该参数。

- Chain

  Flink SQL任务是一个DAG图，由多个节点（Operator）组成，部分上下游的节点在运行时可以合成为一个节点，称为Chain操作。Chain操作后的节点总CPU为所有节点CPU的最大值，总内存为所有节点内存的总和。多节点合成一个节点可以有效地减少网络传输，降低成本。

  > 说明
  > ○ 并发数相同的节点才能进行Chain操作。
  > ○ Group By节点不能进行Chain操作。

6.6. 典型的反压场景及优化思路

反压是流式Shuffle中的一个重要概念，当下游处理能力不足时，会通知上游停止发送数据，从而避免数据丢失。本文为您介绍典型的反压场景及优化思路。

反压检测机制

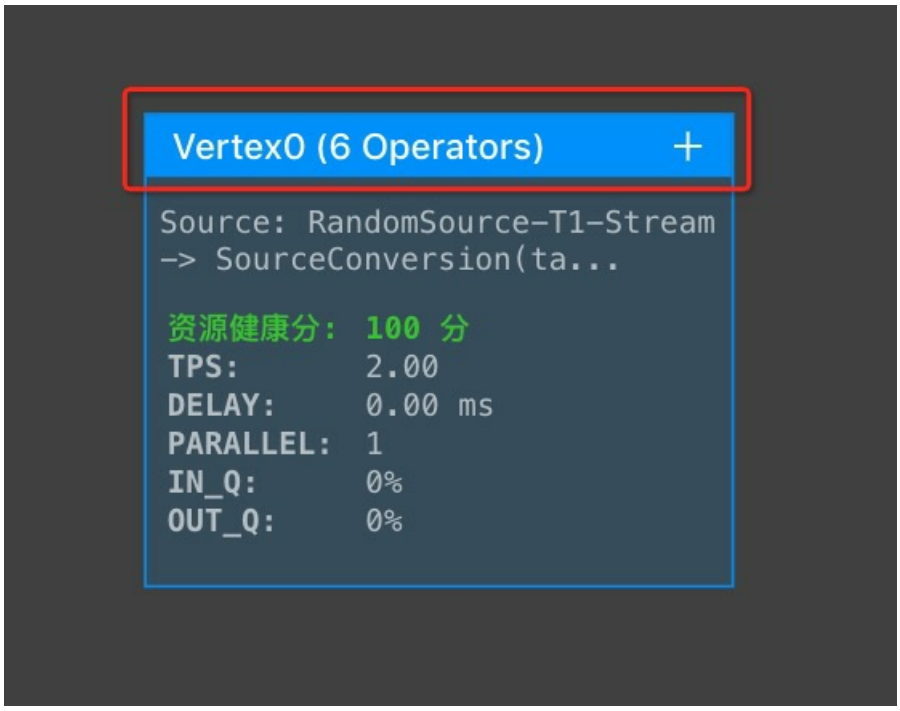
实时计算3.0以上版本提供了作业反压检测机制，通过检测Vertex（可以理解为多个关联Chain在一起的Operator组）输出网络Buffer的拥塞情况，判断对应的Vertex是否存在反压。具体步骤如下：

1. 登录**作业运维**页面。
   i. 登录实时计算控制台。

ii. 单击页面顶部的**运维**。

iii. 在**作业列表**区域，单击**作业名称**下的目标作业名。

2. 在**运行信息**页签，单击对应的**Vertex**节点边框。



3. 在右侧信息区域的**BackPressure > Status**，查看反压状态。
   - 红色**high**：表示该节点存在反压。
   - 绿色**ok**：表示该节点不存在反压。

反压场景及优化思路

> **说明** 示意图中Vertex的颜色为绿色表示通过以上的检测显示不存在反压，红色表示存在反压。

- **仅存在1个Vertex，经检测无反压**

由于Flink的特性，在最后一个Vertex的输出上没有设置网络Buffer（直接写出到下游存储），当作业仅存在一个（或最后一个）Vertex时，反压检测机制失效。因此以上Vertex拓扑示意图并不表示作业不存在反压，若需进一步判断反压点，需将Vertex0中的Operator拆分后再进行判断。拆分方法参见资源调优。

- **存在多个Vertex，倒数第2个Vertex经检测存在反压**

该场景表示：Vertex1被反压，性能瓶颈点在Vertex2。此时，可以通过查看Vertex2中的Operator组的名称进行判断：

- 若只涉及写出下游存储的操作，则可能是写出速度较慢所导致。建议增加Vertex2的并发数，或者配置对应结果表（Sink）的 `batchsize` 参数，具体操作步骤请参见上下游参数调优。
- 若涉及到除写出下游存储外的其他操作，需要将其他操作对应的Operator拆分后再进一步判断。拆分方法参见资源调优。

- **存在多个Vertex，非倒数第2个Vertex经检测存在反压**

该场景表示：Vertex0被反压，性能瓶颈点在Vertex1。此时，可以通过查看Vertex1中的Operator组的名称来判断具体的操作。常见的操作和对应的优化方法如下：

- GROUP BY操作：可以考虑增加并发或设置 `miniBatch` 参数优化状态（State）的操作，具体方法参见作业参数调优。
- 维表JOIN操作：可以考虑增加并发数或者设置维表的Cache策略，具体请参见对应维表文档。
- UDX操作：可以考虑增加并发数或者优化对应的UDX代码。

- **存在多个Vertex，所有Vertex经检测不存在反压**

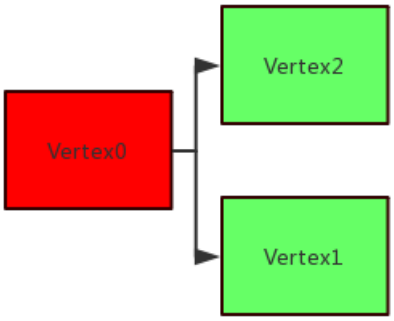
该场景表示：可能的性能瓶颈点在Vertex0，可以通过查看Vertex0中的Operator组的名称判断具体的操作。

- 若其中只存在读取上游源表（Source）的操作，则可能实时计算本身没有性能瓶颈，只是读取Source的速度较慢而导致延时较大。此时可通过增加Source节点的并发数或者设置读取Source的`batchsize`的方法进行优化，具体步骤参见上下游参数调优。

  > **说明** Source节点增加后的并发数不能大于上游存储的Shard数。

- 若其中除读取上游Source的操作外，还存在其他操作，建议将其他操作的Operator拆分后再进一步判断，拆分方法参见资源调优。

- **存在多个Vertex，其中1个Vertex经检测存在反压，但其后续的多个并行Vertex经检测不存在反压**



该场景表示：Vertex0被反压，但是无法直接判断性能瓶颈点是Vertex1还是Vertex2。您可以通过Vertex1和Vertex2的**IN_Q**指标做初步判断，如果对应的**IN_Q**长时间为100%，则此节点极可能存在性能瓶颈点。若需要进一步确认，则需将此节点拆分后进行进一步判断。拆分方法参见资源调优。

6.7. SQL Tuning Advisor

6.7.1. Partitioned All Cache调优

您可以使用Partitioned All Cache调优方式解决超大维表使用Cache All策略在进行JOIN时，缓存无法加载维表全部数据的问题。

背景信息

维表JOIN时使用Cache All策略，默认每个进程都会加载全量的维表数据到缓存中。使用Cache All的情况下，维表JOIN节点配置的内存大小至少是维表内存大小的2倍。

当维表较大时，维表JOIN节点需要消耗大量的内存，另外作业GC会比较严重。甚至对于超大维表（超过1 TB），内存里无法加载全量维表数据。为了解决这个问题，引入Partitioned All Cache优化，上游数据按照Join Key进行Shuffle，每个进程上只需要加载所需的维表数据到缓存中。

开启PartitionedJoin优化可以减少内存开销。但由于上游数据需要按照Join Key进行Shuffle，则会引入额外的网络开销和CPU开销。因此以下场景不适合开启PartitionedJoin：

- 上游数据在JOIN Key上存在严重的数据倾斜，这种场景如果开启PartitionedJoin，则会因为数据倾斜导致慢节点。
- 维表数据较小，例如小于2 GB。这种场景如果开启PartitionedJoin，节约的内存开销和额外引入的网络开销和CPU开销相比，不划算。

调优方式

在维表的DDL的WITH参数中添加partitionedJoin = 'true' 参数。

示例代码

```
 CREATE TABLE white_list (
  id varchar,
  name varchar,
  age int,
  PRIMARY KEY (id),
  PERIOD FOR SYSTEM_TIME --维表标识。
) with (
  type = 'odps',
  endPoint = 'your_end_point_name',
  project = 'your_project_name',
  tableName = 'your_table_name',
  accessId = 'your_access_id',
  accessKey = 'your_access_key',
  `partition` = 'ds=20180905',
  cache = 'ALL',
  partitionedJoin = 'true' -- 开启partitionedJoin。
);
```

6.7.2. MiniBatch/MicroBatch调优

您可以通过MiniBatch/MicroBatch调优方式提升吞吐。

背景信息

MiniBatch和MicroBatch都是微批处理，只是微批的触发机制略有不同。原理都是缓存一定的数据后再触发处理，以减少对State的访问，从而提升吞吐并减少数据的输出量。但是二者有以下差异：

- MiniBatch：主要依靠在每个Task上注册的Timer线程来触发微批，需要消耗一定的线程调度性能。
- MicroBatch：为MiniBatch的升级版，主要基于事件消息来触发微批，事件消息会按您指定的时间间隔在源头插入。MicroBatch在元素序列化效率、反压表现、吞吐和延迟性能上都要优于MiniBatch。

微批处理可以显著的提升系统性能，建议您开启微批处理，例如在聚合场景。但以下场景不建议您开启：

- 微批处理通过增加延迟换取高吞吐，如果您有超低延迟的要求，不建议开启。
- GroupAggregate聚合度很低（Output/Input > 0.8）时，一个批次里基本聚合不到数据，不建议开启。

优化方式

在**开发**页面右侧**作业参数**中设置blink.microBatch.allowLatencyMs或blink.miniBatch.allowLatencyMs参数，二者效果相同，单位为ms。

> **说明** 如果一个作业中同时设置了blink.microBatch.allowLatencyMs和blink.miniBatch.allowLatencyMs参数，建议二者值保持一致。如果设置的值不一致，则代码下方参数生效。

代码示例

- 设置microBatch

```
blink.microBatch.allowLatencyMs=5000
```

- 设置miniBatch

```
blink.miniBatch.size=20000
```

- 同时设置了microBatch和miniBatch

```
# 防止OOM设置每个批次最多缓存数据的条数。
blink.miniBatch.size=20000
blink.microBatch.allowLatencyMs=5000
blink.miniBatch.allowLatencyMs=6000---该配置生效。
```

6.7.3. Cache优化

您可以在维表Join时开启Cache策略并配置相关参数，提高作业吞吐。

背景信息

实时计算Flink版支持LRU和ALL两种缓存策略，它们的调优原理如下：

- LRU：缓存维表里的部分数据。您可以通过cache='LRU'开启LRU缓存优化，系统会为每个JoinTable节点创建一个LRU本地缓存。源表的每条数据都会触发系统在Cache中查找数据，如果存在则直接关联输出，减少了一次I/O请求。如果不存在，再发起查询请求去维表中查找，返回的结果会先存入缓存以备下次查询。

  为了防止缓存无限制增长，可以通过cacheSize调整缓存大小。为了定期更新维表数据，可以通过cacheTTLMs调整缓存的失效时间。cacheTTLMs作用于每条缓存数据上，也就是某条缓存数据在指定时间内没有被访问，则会从缓存中移除。

- ALL：缓存维表里的所有数据。您可以通过cache='ALL'开启ALL缓存优化。系统会为JoinTable节点起一个异步线程去同步维表数据。在作业刚启动时，会先阻塞上游数据，直到缓存数据加载完毕，可以保证在处理上游数据时，维表数据已经被加载到缓存中。

  之后所有的维表查询请求都会通过Cache查询。如果在Cache中无法找到数据，则关联键不存在。在缓存失效后，重新加载维表数据到缓存中。重新加载维表的过程不会阻塞维表关联的处理流程。重新加载的维表数据暂时存放在临时内存中，等加载完毕再和原先的维表数据进行原子替换。

  因为几乎没有I/O请求，所以使用cache ALL的维表JOIN性能可以非常高。但是由于内存需要可以同时容纳两份维表数据，因此需要加大内存的配置。

> **说明** 如果您的业务场景要求每次维表查询请求必须发起一次查询，不可以使用缓存数据，请不要开启缓存策略。

调优方式

维表的DDL的WITH参数中添加cache='LRU'或cache='ALL'。Cache相关参数如下。

参数	说明	是否必填	备注
cache	缓存策略	否	• *None（默认值）*：无缓存。 • *LRU*：需要配置相关参数：缓存大小（cacheSize）、缓存更新时间间隔（cacheTTLMs）和是否开启PartitionedJoin（partitionedJoin）。 • *ALL*：需要配置相关参数：缓存更新时间间隔（cacheTTLMs）、更新时间黑名单（cacheReloadTimeBlackList）和是否开启PartitionedJoin（partitionedJoin）。
cacheSize	缓存大小	否	当缓存策略选择*LRU*时，可以设置缓存大小，默认为*10000*行。
cacheTTLMs	缓存失效时间，单位为毫秒	否	• 当缓存策略选择*LRU*时，可以设置缓存失效时间，默认不过期。 • 当缓存策略选择*ALL*时，缓存失效时间为缓存重新加载的间隔时间，默认不重新加载。
cacheReloadTimeBlackList	更新时间黑名单。在缓存策略选择为ALL时，启用更新时间黑名单，防止在此时间内做Cache更新（例如双11场景）。	否	可选，默认空，格式为 '2017-10-24 14:00 -> 2017-10-24 15:00, 2017-11-10 23:30 -> 2017-11-11 08:00' 。 • 用逗号（,）来分隔多个黑名单。 • 用箭头（->）来分割黑名单的起始结束时间。
partitionedJoin	是否开启PartitionedJoin。	否	默认情况下为false，表示不开启partitionedJoin。在开启PartitionedJoin优化时，主表会在关联维表前，先按照Join KEY进行Shuffle，这样做有以下优点： • 在缓存策略为*LRU*时，可以提高缓存命中率。 • 在缓存策略为*ALL*时，节省内存资源，因为每个并发只缓存自己并发所需要的数据。

示例代码

```
 CREATE TABLE white_list (
  id varchar,
  name varchar,
  age int,
  PRIMARY KEY (id)
) with (
  type = 'odps',
  endPoint = 'your_end_point_name',
  project = 'your_project_name',
  tableName = 'your_table_name',
  accessId = 'your_access_id',
  accessKey = 'your_access_key',
  `partition` = 'ds=20180905',
  cache = 'ALL'
);
```

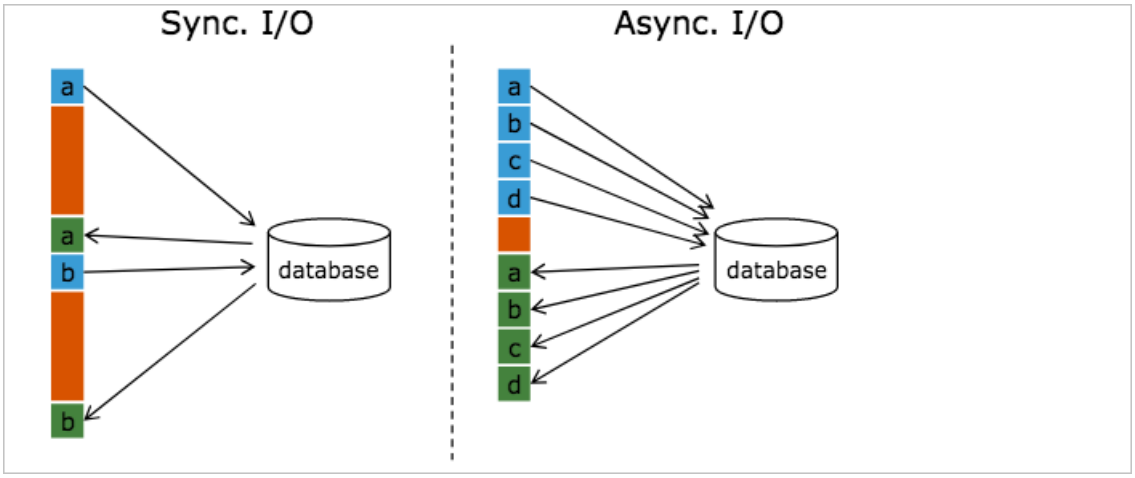
6.7.4. Async优化

您可以在维表JOIN时开启Async优化并配置相关参数，提高吞吐。

背景信息

维表JOIN默认为同步访问方式，上游进来一条数据，则系统去物理表中查询一次，等待返回后输出关联结果。因此网络等待时间极大地阻碍了吞吐和延迟。为了解决同步访问的问题，引入异步模式并发处理查询请求，从而连续的请求之间不需要阻塞等待。

Flink SQL基于Flink Async I/O和异步客户端实现了维表JOIN的异步化，极大地提高了吞吐率。同步模式和异步模式对比图如下。



调优方式

在维表的DDL的WITH参数中添加async='true'，Async 相关参数如下。

参数	说明	是否必填	备注
async	是否开启异步请求	否	默认值为fasle。

参数	说明	是否必填	备注
asyncResultOrder	异步结果顺序	否	取值如下： • unordered（默认值）：无序。 • ordered：有序。
asyncTimeoutMs	异步请求的超时时间	否	单位毫秒，默认值为3分钟。
asyncCapacity	异步请求的队列容量	否	默认值为100。
asyncCallbackThreads	回调处理线程数	否	回调类中的onComplete和onError默认会在线程池中处理该线程池的大小，默认值为50。
asyncConnectionQueueMaxsize	最大请求发送数	否	当等待某个服务器返回结果的请求数量达到asyncConnectionQueueMaxsize值时，异步请求调用也会被阻塞，以防止客户端自身OOM（OutOfMemory），默认值为100。
asyncCallbackQueueMaxsize	最大回调处理队列	否	当等待回调处理的请求达到asyncCallbackQueueMaxsize值时，异步请求调用也会被阻塞，以防止客户端自身OOM，默认值为500。

示例代码

```
CREATE TABLE dim_cn_item(
    rowkey VARCHAR,
    item_id VARCHAR,
    title VARCHAR,
    cate_id VARCHAR,
    cate_name VARCHAR,
    cate_level1_id VARCHAR,
    cate_level2_id VARCHAR,
    cate_level3_id VARCHAR,
    cate_level1_name VARCHAR,
    cate_level2_name VARCHAR,
    cate_level3_name VARCHAR,
    pinlei_id VARCHAR,
    pinlei_name VARCHAR,
    bu_id VARCHAR,
    bu_name VARCHAR,
    PRIMARY KEY(rowkey)
) WITH(
    type='alihbase',
    diamondKey = 'xxxx',
    diamondGroup ='yyyy',
    cacheTTLMs='3600000',
    async='true',
    cache='LRU',
    columnFamily='cf',
    cacheSize='1000',
    tableName='yourTableName'
);
```

常见问题

- 报错详情

```
Caused by: org.apache.flink.table.api.TableException: Output mode can not be UNORDERED if
the input is an update stream.
at org.apache.flink.table.plan.util.TemporalJoinUtil$.validate(TemporalJoinUtil.scala:340
)
at org.apache.flink.table.plan.nodes.common.CommonTemporalTableJoin.translateToPlanIntern
al(CommonTemporalTableJoin.scala:144)
at org.apache.flink.table.plan.nodes.physical.stream.StreamExecTemporalTableJoin.translat
eToPlanInternal(StreamExecTemporalTableJoin.scala:98)
at org.apache.flink.table.plan.nodes.physical.stream.StreamExecTemporalTableJoin.translat
eToPlanInternal(StreamExecTemporalTableJoin.scala:39)
at org.apache.flink.table.plan.nodes.exec.ExecNode$class.translateToPlan(ExecNode.scala:5
8)
at org.apache.flink.table.plan.nodes.physical.stream.StreamExecTemporalTableJoin.org$apac
he$flink$table$plan$nodes$exec$StreamExecNode$$super$translateToPlan(StreamExecTemporalTa
bleJoin.scala:39)
at org.apache.flink.table.plan.nodes.exec.StreamExecNode$class.translateToPlan(StreamExec
Node.scala:38)
at org.apache.flink.table.plan.nodes.physical.stream.StreamExecTemporalTableJoin.translat
eToPlan(StreamExecTemporalTableJoin.scala:39)
at org.apache.flink.table.plan.nodes.physical.stream.StreamExecTemporalTableJoin.translat
eToPlan(StreamExecTemporalTableJoin.scala:39)
at org.apache.flink.table.plan.nodes.physical.stream.StreamExecCalc.translateToPlanIntern
al(StreamExecCalc.scala:89)
at org.apache.flink.table.plan.nodes.physical.stream.StreamExecCalc.translateToPlanIntern
al(StreamExecCalc.scala:43)
at org.apache.flink.table.plan.nodes.exec.ExecNode$class.translateToPlan(ExecNode.scala:5
8)
at org.apache.flink.table.plan.nodes.physical.stream.StreamExecCalc.org$apache$flink$tabl
e$plan$nodes$exec$StreamExecNode$$super$translateToPlan(StreamExecCalc.scala:43)
at org.apache.flink.table.plan.nodes.exec.StreamExecNode$class.translateToPlan(StreamExec
Node.scala:38)
at org.apache.flink.table.plan.nodes.physical.stream.StreamExecCalc.translateToPlan(Strea
mExecCalc.scala:43)
at org.apache.flink.table.plan.nodes.physical.stream.StreamExecSink.translate(StreamExecS
ink.scala:158)
at org.apache.flink.table.plan.nodes.physical.stream.StreamExecSink.translateToPlanIntern
al(StreamExecSink.scala:103)
at org.apache.flink.table.plan.nodes.physical.stream.StreamExecSink.translateToPlanIntern
al(StreamExecSink.scala:53)
at org.apache.flink.table.plan.nodes.exec.ExecNode$class.translateToPlan(ExecNode.scala:5
8)
at
```

- 报错原因

  上游数据为Update Stream，维表JOIN时Async模式未开启ordered。

  > **说明** Update Stream为TOPN或双流JOIN等。

- 解决方案

  在维表WITH参数中设置asyncResultOrder='ordered'。

6.7.5. APPROX_COUNT_DISTINCT优化

您可以通过APPROX_COUNT_DISTINCT（近似Count Distinct）优化提升作业性能。

背景信息

Count Distinct优化时，Aggregate节点的State需要保存所有Distinct Key信息。当Distinct Key数目过大时，State的读写开销太大，因此Count Distinct优化存在的性能瓶颈。但在很多场景，完全精确的统计并不那么必要。如果您希望牺牲部分精确度来换取性能上的提升，您可以使用新增的内置函数APPROX_COUNT_DISTINCT提升作业性能。APPROX_COUNT_DISTINCT支持MiniBatch或Local-Global等Aggregate上的优化，但是需要注意以下几点：

- 输入不含有撤回消息。
- Distinct Key数目需要足够大，例如UV。如果Distinct Key数目不大，APPROX_COUNT_DISTINCT性能相对精确计算提升不大。

调优方式

SQL中直接使用APPROX_COUNT_DISTINCT(user)代替COUNT(DISTINCT user)。APPROX_COUNT_DISTINCT(user)语法格式如下。

```
APPROX_COUNT_DISTINCT(col [, accuracy])
```

其中：

- col：字段名称，任意类型。
- accuracy：指定准确率，可选，取值范围为(0.0, 1.0)，默认值为0.99，取值越高，准确率越高，state开销越大，性能越低。

示例

- 测试数据

a (VARCHAR)	c (BIGINT)
Hi	1
Hi	2
Hi	3
Hi	4
Hi	5
Hi	6

- 测试代码

```
SELECT
  a,
  APPROX_COUNT_DISTINCT(b) as b,
  APPROX_COUNT_DISTINCT(b, 0.9) as c
FROM MyTable
GROUP BY a;
```

- 测试结果

a (VARCHAR)	b (BIGINT)	c (BIGINT)
Hi	5	5

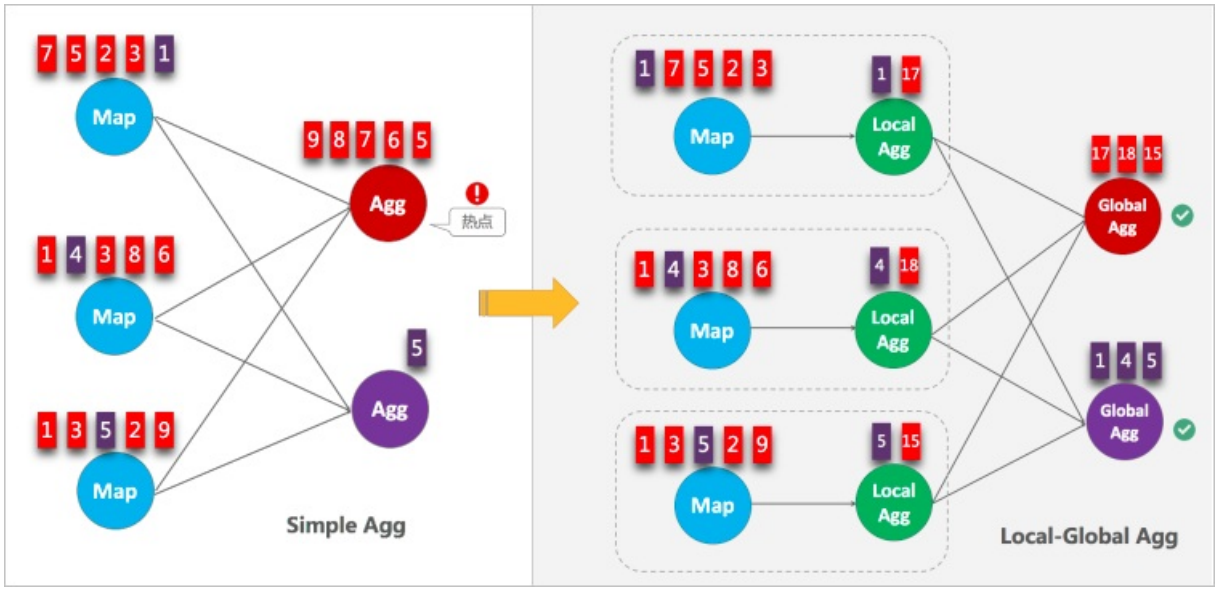
6.7.6. Local-Global优化

您可以通过Local-Global优化解决Aggregate数据倾斜问题。

背景信息

Local-Global优化即将原先的Aggregate分成Local和Global两阶段聚合，即MapReduce模型中Combine+Reduce处理模式。第一阶段在上游节点本地攒一批数据进行聚合（localAgg），并输出这次微批的增量值（Accumulator）。第二阶段再将收到的Accumulator merge起来，得到最终的结果（globalAgg）。

Local-Global本质上能够靠localAgg聚合掉倾斜的数据，从而降低globalAgg热点，从而提升性能。Local-Global用于提升SUM、COUNT、MAX、MIN和AVG等普通Aggregate性能，以及解决这些场景下的数据热点问题。



调优方式

UDAF必须实现merge方法才可以触发Local-Global优化。实现merge方法请参见示例代码。

> **说明** Blink 2.0及以后版本默认开启Local-Global。如果您需要关闭Local-Global，您可以在**作业参数**中，设置blink.localAgg.enabled=false。

示例代码

```
import org.apache.flink.table.functions.AggregateFunction;
public class CountUdaf extends AggregateFunction<Long, CountUdaf.CountAccum> {
    //定义存放count udaf的状态的accumulator数据结构
    public static class CountAccum {
        public long total;
    }
    //初始化count udaf的accumulator
    public CountAccum createAccumulator() {
        CountAccum acc = new CountAccum();
        acc.total = 0;
        return acc;
    }
    //getValue提供了如何通过存放状态的accumulator计算count UDAF的结果的方法
    public Long getValue(CountAccum accumulator) {
        return accumulator.total;
    }
    //accumulate提供了如何根据输入的数据更新count UDAF存放状态的accumulator
    public void accumulate(CountAccum accumulator, Object iValue) {
        accumulator.total++;
    }
    public void merge(CountAccum accumulator, Iterable<CountAccum> its) {
         for (CountAccum other : its) {
            accumulator.total += other.total;
         }
    }
}
```

6.7.7. ROW_NUMBER OVER WINDOW去重

您可以通过ROW_NUMBER OVER WINDOW实现高效去重源数据。

背景信息

去重本质是一种特殊的TopN，实时计算Flink版支持以下两种去重策略：

- 保留首行的去重策略（Deduplicate Keep FirstRow）：保留KEY下第一条出现的数据，之后出现该KEY下的数据会被丢弃掉。因为STATE中只存储了KEY数据，所以性能较优。
- 保留末行的去重策略（Deduplicate Keep LastRow）：保留KEY下最后一条出现的数据。保留末行的去重策略性能略优于LAST_VALUE函数。

调优方式

由于SQL中没有直接的去重语法，实时计算Flink版使用SQL的ROW_NUMBER OVER WINDOW语法表示去重，语法格式如下。

```
SELECT *
FROM (
   SELECT *,
    ROW_NUMBER() OVER ([PARTITION BY col1[, col2..]
    ORDER BY timeAttributeCol [asc|desc]) AS rownum
   FROM table_name)
WHERE rownum = 1;
```

参数	说明
ROW_NUMBER()	计算行号的OVER窗口函数。行号从1开始计算。
PARTITION BY col1[, col2..]	可选。指定分区的列，即去重的KEYS。
ORDER BY timeAttributeCol [asc|desc])	指定排序的列，必须指定一个时间属性字段（Proctime或Rowtime）。还需要指定排列顺序（Deduplicate Keep FirstRow）或者倒序 （Deduplicate Keep LastRow）。 **说明** • 如果不声明时间属性字段，默认为Proctime。 • 如果不声明排列顺序，默认为asc。
rownum	仅支持rownum=1或rownum<=1。

ROW_NUMBER OVER WINDOW去重时需要执行两层查询：

1. ROW_NUMBER()窗口函数根据时间属性列，对相同数据进行排序并标上排名。

   说明
   - 当排序字段是Proctime列时，Flink就会按照系统时间去重，其每次运行的结果是不确定的。
   - 当排序字段是Rowtime列时，Flink就会按照业务时间去重，其每次运行的结果是确定的。

2. 对排名进行过滤，只取第一条或最后一条，达到去重目的。

代码示例

- Deduplicate Keep FirstRow

  将T表按照b字段进行去重，并按照系统时间保留第一条数据。

```
SELECT *
FROM (
  SELECT *,
    ROW_NUMBER() OVER (PARTITION BY b ORDER BY proctime) as rowNum
  FROM T
)
WHERE rowNum = 1;
```

- Deduplicate Keep LastRow

  将T表按照b和d字段进行去重，并按照业务时间保留最后一条数据。

```
SELECT *
FROM (
  SELECT *,
    ROW_NUMBER() OVER (PARTITION BY b, d ORDER BY rowtime DESC) as rowNum
  FROM T
)
WHERE rowNum = 1;
```

6.7.8. Partial-Final优化

您可以通过Partial-Final优化的方式解决Count Distinct热点问题。

背景信息

通常，为了解决Count Distinct热点问题，您会通过增加按Distinct Key取模的打散层的方式，手动将Aggregate改写为两层聚合。Blink 2.2.0及以后版本支持Count Distinct自动打散，即Partial-Final优化。以下场景适用于开启Partial-Final优化：

- 使用Count Distinct后但无法实现聚合节点性能要求。
- Count Distinct所在的Aggregate不包含UDAF。

> **说明** 因为partial-final优化会自动将Aggregate打散成两层聚合，引入额外的网络shuffle。所以，在数据量不大的情况下，反而造成资源浪费。

调优方式

在**开发**页面右侧**作业参数**中设置blink.partialAgg.enabled=true。

开启Partial-Final优化后，您可以在最终生成的拓扑图的节点名中，观察是否包含Expand节点，或者原来一层的Aggregate变成了两层的Aggregate。

示例代码

```
blink.partialAgg.enabled=true
```

7.监控报警

本文为您介绍实时计算监控报警的操作流程以及报警规则的创建步骤。

什么是云监控报警服务

云监控服务能够收集阿里云资源或您自定义的监控指标、探测服务可用性以及针对指标设置警报，让您全面了解阿里云上的资源使用情况、业务的运行状况和健康度，并及时接收异常报警，保证应用程序顺畅运行。

创建报警规则

创建报警规则详情，参见设置报警规则。

实时计算Flink版监控项

监控项	单位	Metric	Dimensions	Statistics
业务延迟（s）	s	inputDelay	userId、regionId、projectName、jobName	Average
读入RPS（RPS）	RPS	ParserTpsRate	userId、regionId、projectName、jobName	Average
写出RPS（RPS）	RPS	SinkOutTpsRate	userId、regionId、projectName、jobName	Average
FailoverRate（%） 说明 Failover Rate表示最近1分钟平均每秒Failover的次数。例如，最近1分钟Failover了1次，则Failover Rate=1/60=0.01667。	%	TaskFailoverRate	userId、regionId、projectName、jobName	Average
处理延迟（s）	s	FetchedDelay	userId、regionId、projectName、jobName	Average

查看监控报警信息

1. 登录实时计算控制台。
2. 单击页面顶部的**运维**。
3. 在**运维**界面，单击目标作业名称。
4. 在页面右上角，单击**更多 > 监控**。

5. 在**监控图标**页面，查看作业的监控指标。

8.自定义日志级别和下载路径

您可以编辑实时计算作业参数，自定义实时计算作业的日志级别和下载路径。

> **注意** 仅实时计算3.2及以上版本支持自定义日志级别和下载路径。

操作步骤

1. 登录实时计算控制台。
2. 单击页面顶部的**开发**。
3. 在左侧导航栏的**作业开发**区域，双击文件夹，查找目标作业。
4. 双击目标作业，进入作业编辑页面。
5. 在作业编辑页面右侧的**作业参数**页面，输入log4j的配置信息。

log4j配置信息	说明
根Logger	Logger负责处理日志记录的部分操作，其语法格式为：`log4j.rootLogger = [ level ] , appenderName, appenderName, …`，表示输出指定级别以上的日志信息到指定的一个或者多个位置。其中： ○ level：日志级别，优先级从高到低分别为ERROR（严重错误）、WARN（一般警告）、INFO（提示信息）、DEBUG（调试信息）。 ○ appenderName：指定日志信息输出到哪个地方，可以同时指定多个输出目的地。

log4j配置信息	说明

日志信息输出目的地Appender	Appender负责控制日志记录操作的输出，其语法格式如下。
	`log4j.appender.appenderName = fully.qualified.name.of.appender.class`
	`log4j.appender.appenderName.option1 = value1`
	`...`
	`log4j.appender.appenderName.optionN = valueN`
	Appender分为以下两种：
	○ log4j自带的Appender
	■ `org.apache.log4j.ConsoleAppender`：控制台
	■ `org.apache.log4j.FileAppender`：文件
	■ `org.apache.log4j.DailyRollingFileAppender`：每天产生一个日志文件
	■ `org.apache.log4j.RollingFileAppender`：文件大小到达指定大小时产生一个新文件
	■ `org.apache.log4j.WriterAppender`：以流格式发送日志信息到任意指定地方
	○ 自定义Appender
	目前只支持日志服务SLS和对象存储OSS，且对应的类是固定的，如下所示：
	■ SLS：`log4j.appender.loghub=com.alibaba.blink.log.loghub.BlinkLogHubAppender`
	■ OSS：`log4j.appender.oss=com.alibaba.blink.log.oss.BlinkOssAppender`
	此外，您也可以配置任何log4j语法支持的配置项。
	详细示例请参见修改日志级别为DEBUG，并输出日志至OSS示例。

> **说明** 完成编辑作业参数后，请重启作业。您可以在OSS存储空间查看新生成的日志。

6. **停止**作业。作业停止步骤请参见停止。
7. **启动**作业。作业启动步骤请参见启动。

注意事项

- 选择和独享模式集群相同的OSS存储路径。
- **日志下载**包含log4j日志的输出方式，不包含*system.out*日志。
- 指定的SLS或OSS存储和作业所在集群可连通。
- 如果自定义日志输出方式配置错误，作业通常可以启动成功，但不能按照自定义配置打印日志。

修改日志级别为DEBUG，并输出日志至OSS示例

> **注意** 手动配置log4j.rootLogger参数会导致实时计算平台无法查看日志信息以及排查相关问题，请谨慎使用。

```
#将总体日志级别修改为DEBUG，并将日志输出至OSS存储空间。
log4j.rootLogger=DEBUG, file, oss
#固定写法。配置OSS的appender类。
log4j.appender.oss=com.alibaba.blink.log.oss.BlinkOssAppender
#Endpoint地址。
log4j.appender.oss.endpoint=oss-cn-hangzhou****.aliyuncs.com
#accessId。
log4j.appender.oss.accessId=U****4ZF
#accessKey。
log4j.appender.oss.accessKey=hsf****DeLw
#bucket名称。
log4j.appender.oss.bucket=et****
#定义日志存储的子目录。
log4j.appender.oss.subdir=/luk****/test/
```

排除指定类的日志，并输出日志至自定义的SLS存储示例

```
#关闭log4j.logger.org.apache.hadoop包下面的日志输出。
log4j.logger.org.apache.hadoop = OFF
#固定写法。配置Loghub的appender类。
log4j.appender.loghub = com.alibaba.blink.log.loghub.BlinkLogHubAppender
#仅将ERROR级别日志输出至SLS。
log4j.appender.loghub.Threshold = ERROR
#SLS中的project名称。
log4j.appender.loghub.projectName = blink-errdumpsls-test
#SLS中的logstore名称。
log4j.appender.loghub.logstore = logstore-3
#SLS的Endpoint地址。
log4j.appender.loghub.endpoint = http://cn-shanghai****.sls.aliyuncs.com
#accessKeyId。
log4j.appender.loghub.accessKeyId = Tq****WR
#accessKey。
log4j.appender.loghub.accessKey = MJ****nfVx
```

修改日志级别为WARN，并关闭指定包下面的日志输出示例

```
#修改整体日志级别为WARN。
log4j.rootLogger=WARN,file
#关闭log4j.logger.org.apache.hadoop包下面的日志输出。
log4j.logger.org.apache.hadoop = OFF
```

9.管理独享集群Blink版本

实时计算Flink版为独享模式集群提供版本管理功能，您可自行管理集群对应的Blink引擎版本。

安装版本

1. 登录版本管理页面。
   i. 登录实时计算控制台。
   ii. 将鼠标悬停至控制台页面右上角个人账户位置，单击**项目管理**。
   iii. 在左侧导航栏，单击**集群管理 > 集群列表**。
   iv. 在**集群列表**区域，单击目标集群**操作**列下的**更多 > 版本管理**。
2. 在**可安装版本**页面，单击目标版本**操作**列下的**安装**。

   您可以在**可安装版本**页签，根据业务需求和版本特性，安装适合的版本。

标签	说明
stable	当前推荐安装的稳定版本
beta	测试版本 **说明** 在特定场景下才可安装使用，非特定场景不保证版本质量和稳定性，不建议安装。
无标签	历史稳定版本

说明

- 一次只能安装一个版本，安装过程中集群状态显示为**版本部署中**。
- 不可安装已安装过的版本。
- 每个版本设有服役时间（默认一年），版本退役后您可以继续使用此版本，但阿里云不再负责维护。

切换版本

不同的Blink版本具备不同的特性，您可以根据实际需求，设置合适的Blink版本为当前版本。您可以通过**版本切换**功能，为单个作业变更Blink版本。

1. 登录实时计算控制台。
2. 单击页面的顶部**开发**。
3. 在**作业开发**页面，双击目标文件夹下的目标作业，进入**作业开发**页面。
4. 在**作业开发**页面的右下角，单击版本信息。
5. 在选项列表中选择目标版本。

卸载版本

当前实时计算Flink版仅支持安装3个Blink版本，如果您已经安装3个Blink版本，且需要安装新版本，则您需要先卸载版本。

1. 登录版本管理页面。
   i. 登录实时计算控制台。
   ii. 将鼠标悬停至控制台页面右上角个人账户位置，单击**项目管理**。
   iii. 在左侧导航栏，单击**集群管理 > 集群列表**。
   iv. 在**集群列表**区域，单击目标集群**操作**列下的**更多 > 版本管理**。
2. 在**已安装版本**页面，单击目标版本**操作**列下的**卸载**。

> 说明
> - 不能卸载已设置为current的版本。
> - 不能卸载作业已引用的版本。

设置为当前版本

您可以选择一个已安装的Blink版本，将它设置为当前集群内作业的默认版本。不同的Blink版本具备不同的特性，您可以根据实际需求，将合适的Blink版本设置为当前版本。

1. 登录版本管理页面。
   i. 登录实时计算控制台。
   ii. 将鼠标悬停至控制台页面右上角个人账户位置，单击**项目管理**。
   iii. 在左侧导航栏，单击**集群管理 > 集群列表**。
   iv. 在**集群列表**区域，单击目标集群**操作**列下的**更多 > 版本管理**。
2. 在**已安装版本**页面，单击目标版本**操作**列下的**设置为current**。

常见问题

- Q：安装引擎出现`blink-<version> already installed`报错信息。

  A：该版本已经安装，无需再次安装。

- Q：安装引擎出现`Flink versions exceeded max limitation:3`报错信息。

  A：超出版本限制。若需安装新版本，需预先删除原有版本，使已安装版本数小于3。

- Q：卸载引擎出现`Node:<nodeName> in project:<projectName> still ref the version:<blinkVersion>`报错信息。

  A：线上存在引用此版本引擎的作业，请根据报错信息中的作业名称和项目名称，将该作业先进行下线。

- Q：单击**语法检查**或**上线**产生如下报错。

  ```
  code:[30006], brief info:[blink script not exist, please check blink version], context info:[blink script:[/home/admin/blink/blink-2.2.6-hotfix0/bin/flink], blink version:[/home/admin/blink/blink-2.2.6-hotfix0/bin/flink]]
  ```

  A：该作业使用的引擎不存在。进入**版本管理 > 已安装版本**查看是否存在该版本引擎：
  - 不存在：请切换或安装此版本引擎。
  - 存在：请提交工单。