ALIBABA CLOUD

阿里云

金融分布式架构
单元化入门指南

文档版本：20210220

阿里云

法律声明

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 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 ---------- 05
2.步骤一：架构规划 ---------- 06
3.步骤二：初始化环境 ---------- 09
4.步骤三：开发单元化应用 ---------- 16
5.步骤四：构建并发布应用 ---------- 24
6.步骤五：到控制台验证 ---------- 28
7.步骤六：调拨应用流量 ---------- 32

1.概述

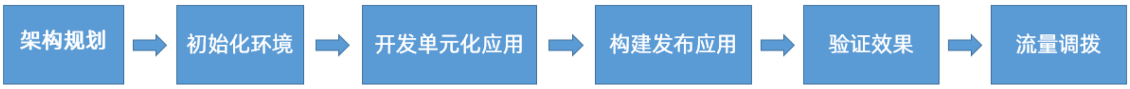
本指南基于金融行业常见的转账和积分场景，指导您从单元化的技术架构和应用架构的规划，到环境搭建和单元化应用的开发部署，再到应用流量调拨和单元化功能的验证，逐步完成 SOFAStack 单元化能力的入门体验。

前提条件

- 已搭建双机房环境。
- 环境中已部署相关的单元化产品：负载均衡（ALB）、统一网关（Spanner）、中间件（MQ、MS、DTX、DBP、DST）、单元化应用服务（LHC）、实时监控（RMS）。

总体流程

本指南涉及的总体操作流程如下图：



1. **架构规划**：进行单元化技术架构规划和应用架构规划，输出部署架构图和应用架构图，指导研发运维人员初始化环境和应用研发改造。
2. **初始化环境**：根据架构规划初始化单元化环境，为后续的应用发布做准备。
3. **开发单元化应用**：对传统应用进行开发改造，完成单元化功能配置。
4. **构建并发布应用**：通过单元化应用服务（LHC）将本地开发好的单元化应用发布到双机房中的部署单元。
5. **验证效果**：基于开发单元化应用中的转账与积分场景，在完成相应的转账或存款等操作后，您可以前往各个产品控制台验证单元化效果。
6. **流量调拨**：使用单元化应用服务（LHC）做应用层流量的调拨，通过调整并下发接入层和应用层的分片规则，验证流量路由和 RPC 路由的准确性。

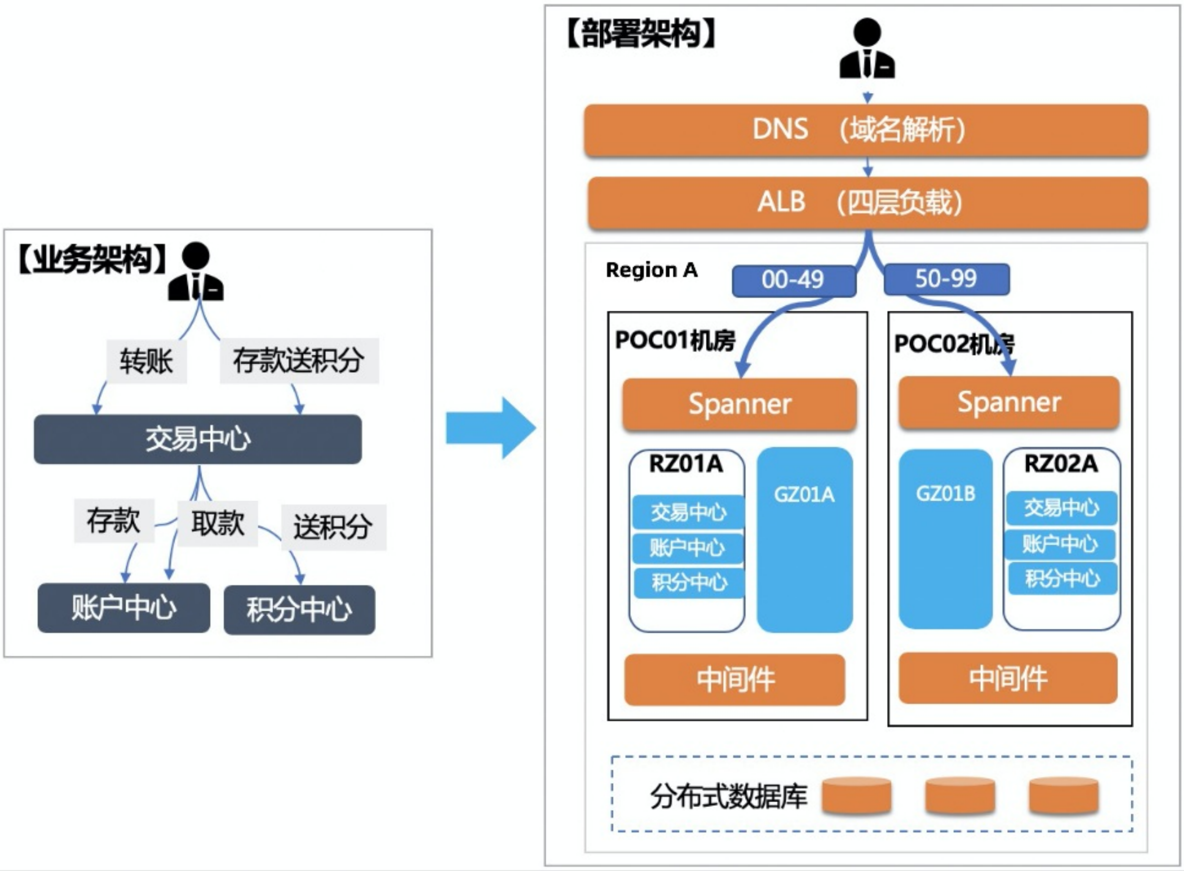
2.步骤一：架构规划

本文以同城双活单元化架构为示例，介绍如何进行单元化技术架构和应用架构的规划，输出部署架构图和应用架构图，指导研发运维人员初始化环境和进行应用研发改造。

前提条件

了解 单元化架构

架构介绍



如上图所示，该业务系统搭建在同城双活单元化架构上。

业务架构覆盖的业务场景如下：

- 转账
- 存款送积分

支撑业务的应用如下：

- 交易中心（txnflow）：交易服务中心，提供转账服务，依次调用转出账户的取款服务和转入账户的存款服务，分布式事务的发起者。
- 账户中心（acccenter）：提供存款和取款服务两种原子的账务操作，分布式事务的参与者。
- 积分中心（pointcenter）：提供积分返还服务，主要验证事务性消息。

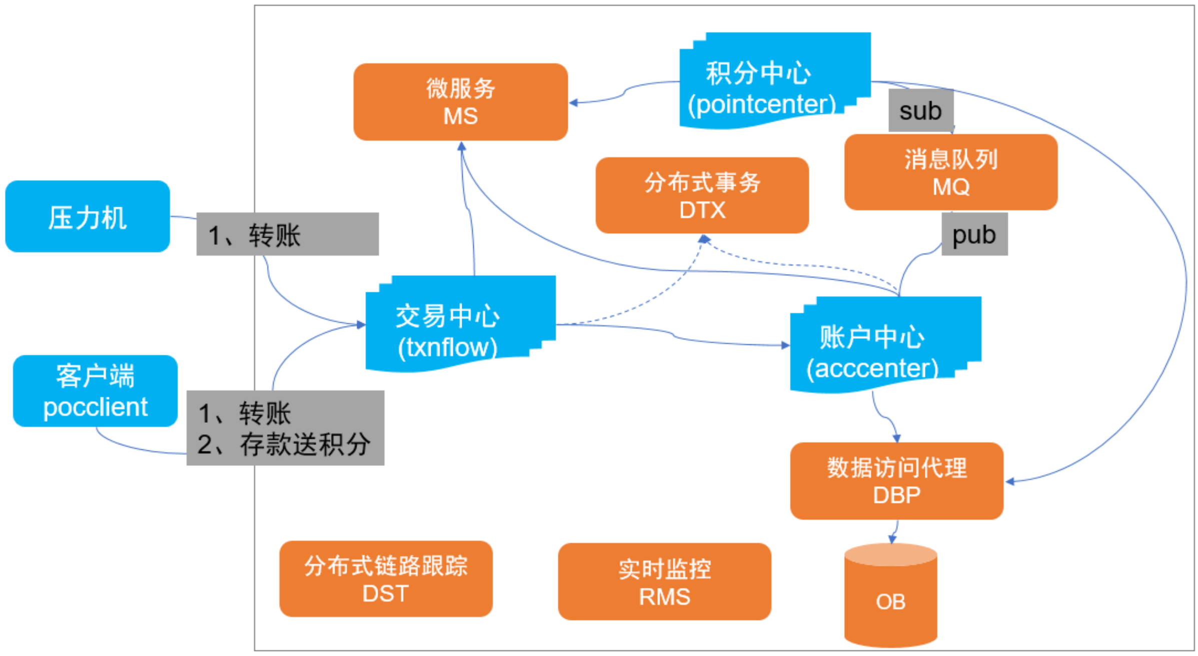
部署架构中主要涉及的单元如下：

- 两个机房（IDC）：POC01 和 POC02。

- 两组本地域（RZone）：RZ01 和 RZ02。
    - 日常每组 RZone 负责 50 个分片数据，承载 50% 业务流量。
    - 使用LHC单元化应用服务-流量管理，可动态调整分片规则，进行流量调拨。
- 一组全局域（GZone）：GZ01A 和 GZ01B。在 2 个机房部署 2 个应用层单元，互为备份。

应用架构

业务应用的调用链路如下图所示。



应用拆分

> ⓘ 说明
>
> 本示例教程暂不涉及 GZone 类型的应用。

以下 3 个应用都是 RZone 类型应用，可按照 uid 进行单元化拆分。

应用名	Zone 类型
txnflow	RZone
acccenter	RZone
pointcenter	RZone

数据拆分

在本教程中，uid 格式如： `080066600000002` ，按照 uid 进行 10 库 10 表拆分。

使用的中间件

中间件	配置信息	备注
消息队列	TopicGroupID	创建 topic、groupid 和路由规则。
数据访问代理	1 个实例，3 个库，每个库 10 库 10 表	创建实例，添加数据节点，创建数据库、数据表，并连接数据访问代理。
分布式事务	无	无
微服务	无	无

3.步骤二：初始化环境

本文介绍如何根据架构规划初始化单元化环境，为后续的应用发布做准备。

> **说明**
>
> 本文仅适用于公有云杭州金区环境。

步骤一：创建单元化工作空间

1. 登录 LHC 控制台。
2. 在左侧导航栏，单击下方的 **全局设置**，进入工作空间列表页面。
3. 单击 **创建工作空间**，选择 **单元化工作空间** 类型，单击 **创建**。
    - **标准工作空间**：标准工作空间（Workspace）是 SOFAStack 提供的一种组织机制，将服务于不同目标、阶段的资源分组隔离管理。您可以根据研发运维流程，为每个阶段分配工作空间，例如单机房（即单可用区）的开发工作空间、双机房（即包括两个可用区）的生产工作空间。
    - **单元化工作空间**：在标准工作空间的基础上提供了单元化能力，可用于同城双活及异地容灾场景，本质是一组标准工作空间的集合。您可以通过单元化工作空间组对用户资源进行隔离，不同工作空间组下的集群彼此隔离。
4. 在 **创建工作空间** 页面，输入以下基本信息。
    - **工作空间标识**：2-64 个字符，工作空间的英文标识，全局唯一，一经确定无法修改，例如：dev、test、prod 等。本例中输入：`DemoIDC`。
    - **工作空间名称**：1-64 个字符，工作空间的显示名称，例如：开发工作空间、测试工作空间、生产工作空间。本例中输入：`DemoIDC`。
    - **地域**：选择工作空间所在的地域（Region）。
    - **网络类型**：选择 **VPC 网络**。
5. 单击 **下一步**，在 **创建单元架构** 页面单击 **添加可用区**。 每个工作空间配置的可用区（Available Zone）个数不做限制。本例中为工作空间配置两个可用区，为支持双机房高可用等架构做准备。

    > **说明**
    >
    > 系统会根据可用区配置自动为您生成单元化架构拓扑，划分好逻辑单元和部署单元。

6. 单击 **下一步**，在 **创建 VPC** 页面，输入以下配置信息：
   - **专有网络名称**：由 2-128 个英文或中文字符组成，必须以大小字母或中文开头，可包含数字、下划线（_）或连字符（-），不能以 `http://` 和 `https://` 开头。推荐与工作空间名称相同。本例中分别输入：DemoIDC1、DemoIDC2。
   - **专有网络网段**：专有网络的网段，一旦选择便无法更改，专有网络内的所有资源，如 ECS、RDS、SLB 的私网 IP 都在该网段内。可选网段如下：
     - 10.0.0.0/9
     - 172.16.0.0/12
     - 192.168.0.0/16
   - **交换机**：单击 **添加交换机**，在弹出的 **创建交换机** 窗口，填写以下信息，单击 **提交**。
     - **名称**：交换机名称。长度为 2-128 个字符，以英文字母或中文开头，可包含数字、下划线（_）和短横线（-）。
     - **可用区**：交换机的可用区。同一 VPC 内不同可用区的交换机内网互通。您需要为每个可用区创建一台交换机。
     - **自定义网段**：默认关闭。开启后，需填写网段地址。交换机的网段可以和其所属的 VPC 网段相同或者是其 VPC 网段的子网。
     - **子网掩码**：**自定义网段** 关闭时，需分别选择子网掩码和网段地址。默认专有的网段掩码是 24 位，例如 172.31.0.0/24，最多可提供 65536 个私网 IP 地址。范围为 16~29 位之间，可提供 4~65532 个地址。

    - **描述**： 输入交换机的描述信息。可包含 2-256 个中英文字符，不能以 `http://` 和 `https://` 开头。

7. 单击 **下一步**，在 **创建安全组** 页面，单击 **添加安全组**，在弹出的 **添加安全组** 窗口，填写以下信息，单击 **提交**。
   - **安全组名称**：可以为空，长度为 2-128个英文或中文字符，必须以大小字母或中文开头，不能以 `http://` 和 `https://` 开头。可以包含数字、半角冒号（:）、下划线（_）或者连字符（-）。
   - **描述**：可包含 2-256 个中英文字符，不能以 `http://` 和 `https://` 开头。
   - **规则**：保持默认设置。

步骤二：创建集群

集群是运行工作负载的逻辑分组，包含一组云服务器资源，每台云服务器即集群中的一个节点。首次使用 LHC 时，您需要创建一个初始集群，并添加至少一个节点。

> **说明**
>
> 集群创建一般需要 20 分钟左右，创建时间与包含的可用区及节点数目有关。

本例中，您将为单元化工作空间中的两个可用区分别创建一个集群：`democluster1`、`democluster2`，实现同城双活的架构。

1. 登录 LHC 控制台，在左侧导航栏单击 **集群管理 > 集群详情**。
2. 在 **集群列表** 页，单击 **创建集群**。
3. 在 **集群创建** 页面，系统会自动进行预检查，确保相关的产品已经开通并且租户下的账户余额大于 100 元。预检查通过后，单击 **下一步**。

   > **说明**
   >
   > 若检查失败，完成修复失败项目后，可以单击 **重新检查** 操作重新进行预检查。

4. 在 **基本配置** 页面，填写以下配置信息。
   - **专有网络**：选择当前单元化工作空间所属的 VPC。
   - **集群名称**：集群显示名称，不能为空。本例中分别输入：`democluster1`、`democluster2`。
   - **Kubernetes 版本**：选择 Kubernetes 版本。可选择 `1.16.9-aliyun.1` 和 `1.18.8-aliyun.1`。
   - **容器运行时**：`docker 19.03.5`。
   - **网络配置**：
     - **虚拟交换机**：您可以在已有虚拟交换机列表中，根据可用区选择 1~3 个交换机。如果没有您需要的交换机，可以通过单击创建虚拟交换机进行创建，请参见 创建交换机。
     - **网络插件**：设置启用的网络插件，支持 Terway 网络插件，具体请参见 Flannel 与 Terway。Terway 是阿里云容器服务自研的网络插件，将阿里云的弹性网卡分配给容器，支持 Kubernetes 的 NetworkPolicy 来定义容器间的访问策略，支持对单个容器做带宽的限流。
     - **Pod 虚拟交换机**：网络插件选择 Terway 时，需要为 Pod 分配 IP 的虚拟交换机。每个 Pod 虚拟交换机分别对应一个 Worker 实例的虚拟交换机。如果没有您需要的交换机，可以通过单击创建虚拟交换机进行创建，请参见 创建交换机。

  - **Service CIDR**：设置 Service CIDR。网段不能与 VPC 及 Pod 地址段重复，创建成功后不能修改。
  - **高级配置**：
    - **配置 SNAT**：不可修改，默认选中。创建集群时，默认不开通公网。如果您选择的 VPC 不具备公网访问能力，选中为专有网络配置 SNAT 后，ACK 将为您创建 NAT 网关并自动配置 SNAT 规则。
    - **公网访问**：设置是否开放使用 EIP 暴露 API Server。API Server 提供了各类资源对象（Pod、Service 等）的增删改查及 watch 等 HTTP Rest 接口。
      - 若选择开放，会创建一个 EIP，并挂载到内网 SLB 上。此时，Master 节点的 6443 端口（对应 API Server）暴露出来，您可以在外网通过 kubeconfig 连接并操作集群。
      - 若选择不开放，则不会创建 EIP，您只能在 VPC 内部用 kubeconfig 连接并操作集群。

      > **说明**
      > 若需获取 kubeconfig 信息，请前往 ACK 控制台。

    - **kube-proxy 代理模式**：支持 iptables 和 IPVS 两种模式。
      - iptables：成熟稳定的 kube-proxy 代理模式，Kubernetes service 的服务发现和负载均衡使用 iptables 规则配置，但性能一般，受规模影响较大，适用于集群存在少量的 Service。
      - IPVS：高性能的 kube-proxy 代理模式，Kubernetes service 的服务发现和负载均衡使用 Linux ipvs 模块进行配置，适用于集群存在大量的 service，对负载均衡有高性能要求的场景。
    - **集群删除保护**：防止通过控制台或者 API 误删除集群。默认选中，可修改。
    - **部署单元**：默认勾选使用默认配置。若取消勾选，可以为可用区设置不同的部署单元，若需要修改部署单元，可前往 **经典应用服务控制台 > 环境参数** 进行设置。
    - **数据盘挂载**：默认勾选 **使用日志服务**，不可修改，将自动创建名称为 `k8s-log-{ClusterID}` 的 Project。

5. 配置完成后，单击 **下一步**。
6. 在 **节点配置** 页面，完成以下 Worker 节点配置。
   - **付费类型**：支持按量付费和包年包月两种节点付费类型。选择包年包月时，需设置以下参数。
     - **购买时长**：目前支持选择 1、2、3、6 个月和 1~5 年。
     - **自动续费**：设置是否自动续费。
   - **节点数量**：创建 Worker 实例（ECS 实例）的数量。
   - **实例规格**：支持选择多个实例规格。详情请参见 实例规格族。
   - **系统盘**：支持 ESSD 云盘、SSD 云盘和高效云盘。
   - **挂载磁盘**：支持 SSD 云盘和高效云盘。
   - **操作系统**：支持 CentOS 和 Aliyun Linux 操作系统。
   - **登录密码**：设置节点的登录密码。8-30 个字符，且至少包含三项（大写字母、小写字母、数字和特殊符号）。
   - **确认密码**：确认设置的节点登录密码。
7. 配置完成后，单击 **下一步**。
8. 在集群的 **配置预览** 页面，确认配置无误后，单击 **提交**。

> **说明**
> - 一个包含多节点的 Kubernetes 集群的创建时间一般约为十分钟。
> - 在集群的创建过程中，若出现任务失败的情况，可单击事件查看具体错误详情，或单击 **重试** 或 **忽略**。

步骤三：创建网关集群

统一接入网关以集群的方式承载业务负载均衡的流量，您需要根据流量的类型来规划集群，比如公网、内网、办公网等。

> **说明**
> 创建好统一接入网关集群后，您还需要为网关集群创建统一接入实例，并为应用服务里创建负载均衡类型的 Service，才能进行各类业务流量的接入。

1. 登录 LHC 控制台，在左侧导航栏单击 **网络 > 统一接入集群**。
2. 在 **集群列表** 页，单击 **创建网关集群**。
3. 在 **创建网关集群** 页面，填写以下配置信息。
    - **集群名称**：集群名称允许长度为 1-63 个字符，只包含小写字母、数字、中划线、且必须以字母开头、以字母或者数字结尾。
    - **网络类型**：可指定的网络类型包括内网、公网。指定集群的网络类型后，该集群上仅能创建与之相同网络类型的统一接入实例。
    - **初始集群状态**：可选择 **在线** 或 **维护**。
        - 在线：网关集群对外提供流量转发服务，同时可以接受转发规则的更新。
        - 维护：网关集群仅对外提供流量转发服务，但不接受转发规则的更新请求。
    - **容器规格**：容器的规格和节点的数量决定了该集群的请求处理能力，最小规格为 1 核 CPU 1 GB 内存 5GB 磁盘。
    - **host 网络模式**：网关集群所采用的网络模型。若选择否，每个 Pod 会拥有独立 IP。
    - **容器版本配置**：
        - **名称**：版本名称允许长度为 1-63 个字符，只包含小写字母、数字、中划线、且必须以字母开头、以字母或者数字结尾。
        - **容器镜像**：输入网关节点的容器镜像地址。示例镜像：`registry.cn-hangzhou.aliyuncs.com/sofastack/spannerplus:tini_2.0.0-cloud-alpha-d645a95989f6914`。
        - **副本数**：输入网关节点数，最大值为 100。
        - **标签**：可选，label key 由两部分组成：前缀（可选）和名称，通过 `/` 划分。名称部分最多 63 个字符，以 [a-z]、[0-9]、[A-Z] 开头和结尾，且中间包含数字字母和 `_-.` 。前缀是可选的，使用时须符合 DNS subdomain 规范，即一系列 DNS label 用 `.` 拼接，总长度不超过 253 个字符，`kubernetes.io/` 及 `k8s.io/` 为保留前缀。Label values 最多包含 63 个字符，以 [a-z]、[0-9]、[A-Z] 开头和结尾，且中间包含数字字母和 `_-.` ，value 目前不可为空。

步骤四：创建统一接入实例

前提条件

- 已 创建统一接入集群。
- 网关集群的状态是 **在线** 或 **维护中**，且网络类型与接入实例的网络类型一致。

操作步骤

1. 登录 LHC 控制台。
2. 在左侧导航栏中选择 **网络 > 统一接入实例**，进入统一接入实例列表页面。
3. 单击 **创建实例**，填写以下配置信息：
   - **基本信息**：
     - **实例名称**：必填，允许包含小写字母、数字、中划线、且必须以字母开头、以字母或者数字结尾。允许长度为 1-63 个字符。不可与已有实例名称重复。
     - **网络类型**：支持 **内网** 或 **公网**。
     - **统一接入集群**：选择可用的接入集群。
   - **配置信息**：
     - **http 协议**：必填。设置 **前端端口** 号，范围为 1-65535。
     - **https 协议**：选填。需设置 **前端端口** 号以及 **证书 ID**。

       说明

       目前 HTTPS 协议的支持仍在内测中，不建议在生产环境中使用。若需测试，可以提交工单获取证书 ID。

步骤五：单元路由规则初始化

1. 登录 LHC 控制台，在左侧导航栏单击 **流量管理 > 应用层**。

   在 **正式流量** 页签中，系统会先展示最近一次推送成功的全局流量规则，即当前生效的全局流量快照。
2. 单击 **规则配置** 进入 **正式流量规则配置** 页面，该页面展示的是当前已配置的规则（只是配置，不代表已生效）。
3. 单击逻辑单元卡片或分片数据即可进入编辑页面，填写以下配置信息：

   对于 RZone 类型的部署单元，配置对应部署单元的 UID 分片配置如下。
   - RZ01A : 00-49
   - RZ02A : 50-99

   说明

   请确保调整后的各个部署单元的流量比总和为 100%。UID 分片的取值范围为 00~ 99。

步骤六：中间件初始化

消息队列初始化

创建 Topic

1. 登录消息队列控制台，左侧导航栏选择 **Topic 管理**，进入 Topic 列表页面。
2. 单击列表左上方的 **创建 Topic** 按钮，在新弹出的对话框中，输入或选择 Topic 信息：
    - **Topic**：Topic 名称，命名不能以 “CID” 和 “GID” 开头，只能包含英文、数字、连字符（-）和下划线（_），且长度需控制在 3-64 个字符之间，例如 `TP_SCRCU_POC` 。
    - **消息类型**：选择事务消息。
    - **描述**：可选，对该 Topic 的备注内容。
3. 单击 **确定**。

创建 GroupID

1. 登录消息队列控制台，左侧导航栏选择 **Group 管理**，进入 Group ID 列表页面。
2. 单击列表左上方的 **创建 Group ID** 按钮，在新弹出的对话框中，输入或选择 Group ID 信息：
    - **Group ID**：必填，以 “GID” **或者 “GID-” 开头，只能包含字母、数字、连字符（-）和下划线（）**，且长度限制在 7-64 字符之间，例如 `GID_SGROUP` 。
    - **描述**：可选，对该 Group ID 的备注信息。
3. 单击 **确定**。

创建消息路由任务

1. 登录消息队列控制台，在左侧导航栏中选择 **消息路由**。
2. 在消息路由任务列表的左上方，单击 **创建路由任务**。
3. 在 **创建路由任务** 窗口中，配置以下任务信息：
    - **源 Topic**：输入需要同步的消息所属 Topic 名称，如 `TP_SCRCU_POC` 。
    - **目标单元**：选择消息将被同步到的 Topic 所属单元，如 `RZONE` 。
    - **目标 Topic**：输入消息将被同步到的 Topic 名称，如 `TP_SCRCU_POC` 。
    - **起始同步位点**：选择从源 Topic 中的消息队列的哪个位置开始进行消息同步，即从这个位置之后进入队列的消息都会被同步到目标 Topic。根据实际需要选择即可。
        - **最小位点**：即任务首次启动之后，从有效期内最早写入源 Topic 队列的消息开始同步，首次任务启动之前发的消息不会被同步。
        - **最大位点**：即任务首次启动之后，从最新写入源 Topic 队列的消息开始同步，首次任务启动之前发的消息不会被同步。
        - **自定义位点**：选择从源 Topic 中的消息队列的哪个位置开始进行消息同步，首次任务启动之前发的消息不会被同步。
    - **描述**：选填，输入对该同步任务的具体描述或备注。
4. 单击 **确定**。

数据访问代理初始化

详情请参见 使用双机房 ODP 实例。

4.步骤三：开发单元化应用

在单元化架构中，您需要对本地应用进行开发改造，完成单元化功能配置。本文将基于转账、积分等场景分别介绍微服务（MS）中的 SOFARPC、消息队列（MQ）以及分布式事务（DTX）如何完成 LDC 单元化相关的业务开发。

前提条件

路由参数为 userId，格式如 080066600000002，取第一位 + 0 作为分片位（sharding key）。当 userId = 080066600000002 时，分片位（sharding key）= 0 + 0 = 00当 userId = 180066600000000 时，分片位（sharding key）= 1 + 0 = 10

微服务中的 SOFARPC

按照 RPC 服务引用的标准使用规范，有如下要求：

1. 接口入参的第一个参数为 `userId` 路由信息。
2. SOFARPC 默认从这个参数中提取和生成两位分片位（sharding key），默认提取 UID 的倒数二、三位，参见 `com.alipay.sofa.rpc.ldc.DefaultLdcRouteProvider` 。

在本 demo 中，因无法满足该规范，将自定义一个 PocLdcRouteProvider，实现提取两位分片位的逻辑，替代 DefaultLdcRouteProvider 的标准实现。

```
import com.alipay.sofa.rpc.api.ldc.LdcRouteJudgeResult;
import com.alipay.sofa.rpc.api.ldc.LdcRouteProvider;
import com.alipay.sofa.rpc.common.utils.CommonUtils;
import com.alipay.sofa.rpc.config.ConsumerConfig;
import com.alipay.sofa.rpc.core.request.SofaRequest;
import com.alipay.zoneclient.api.EnterpriseZoneClientHolder;

public class PocLdcRouteProvider implements LdcRouteProvider {
  @Override
  public LdcRouteJudgeResult uidGenerator(ConsumerConfig consumerConfig, SofaRequest sofaRequest) {
    LdcRouteJudgeResult result = new LdcRouteJudgeResult();

    //如果没有开启ldc，那么就直接返回false
    if (!EnterpriseZoneClientHolder.isZoneMode()) {
      return result;
    }

    Object[] methodArgs = sofaRequest.getMethodArgs();
    if (CommonUtils.isEmpty(methodArgs)) {
      result.setSuccess(false);
      return result;
    }
    Object methodArg = methodArgs[0];
    if (methodArg instanceofString){
      String routeUid = (String) methodArg;
      String uid = UIDUtil.parseShardingKeyFromBacc(routeUid);
      result.setSuccess(true);
      result.setRouteId(uid);
      return result;
    }
    result.setSuccess(false);
    return result;
  }

  @Override
  public int order() {
    return 2;
  }
}
```

UIDUt il 类的代码如下：

```
public final class UIDUtil {
   private UIDUtil() { }

   public static String parseShardingKeyFromBacc(String bacc) {
      if (bacc == null) {
         throw new NullPointerException("bacc is null");
      }
      if ("".equals(bacc.trim())) {
         throw new IllegalArgumentException("bacc is empty");
      }
      return bacc.substring(0, 1) + "0";
   }
}
```

交易启动时，首先需要向 SOFARPC 注册定制的路由逻辑，代码如下：

```
com.alipay.sofa.rpc.ldc.LdcProviderManager.getInstance().registeLdcRouteProvider(newPocLdcRouteProvi
der());
```

在取款接口中，添加一个 UID 参数，代码如下：

```
 @TwoPhaseBusinessAction(name = "pocDebitFirstAction", commitMethod = "commit", rollbackMethod = "r
ollback")
   public AccountTransResult debit(String uid, @BusinessActionContextParameter(isParamInProperty = true
) AccountTransRequest accountTransRequest,
                                   BusinessActionContext businessActionContext);
```

在存款接口中，添加一个 UID 参数，代码如下：

```
@TwoPhaseBusinessAction(name ="pocCreditFirstAction", commitMethod ="commit", rollbackMethod ="ro
llback")
public AccountTransResult credit(String uid,@BusinessActionContextParameter(isParamInProperty =true)A
ccountTransRequest accountTransRequest,
BusinessActionContext businessActionContext);
```

有关 SOFARPC 单元化配置的更多信息，参见 单元化配置。

消息队列

此处假设一个存款加积分的场景：在账户 A 存入一笔钱后，需要增加账户 A 的积分。此场景下，需要消息队列（MQ）中间件通过一条发送事务消息，通知积分中心执行积分增加操作。

发送事务消息

发送事务消息代码示例如下：

```
// 启动 producer
  public void afterPropertiesSet()throwsException {
    MessagingAccessPoint accessPoint = OMS.builder().driver("sofamq").build();

    Properties properties = new Properties();
    //替换 GID_PGROUP 为您实际创建的 Group ID
    properties.setProperty(PropertyKeyConst.GROUP_ID, "GID_PGROUP");
    transactionProducer = accessPoint.createTransactionProducer(properties, newLocalTransactionChecke
r() {
      @Override
      publicTransactionStatus check (Message msg){
        returnTransactionStatus.CommitTransaction;
      }
    });
    transactionProducer.start();
    LogUtil.info(LOGGER, "transaction producer started");
  }

  // 发送消息
  public void publishMessage(TxnFlowRequest request) {
    try {
      PointAcctDTO pointAcctDTO = buildPointReturnDTO(request);
      Message message = new Message(TOPIC, EVENT_CODE,    hessianSerializer.serialize(pointAcctDTO));
      String shardingKey = UIDUtil.parseShardingKeyFromBacc(request.getBacc());
      message.putUserProperties(UserPropKey.CELL_UID, shardingKey);
      transactionProducer.send(message, (msg, arg) -> {
        returnTransactionStatus.CommitTransaction;
      }, null);
      LogUtil.info(LOGGER, "Public a message, success. TOPIC [{}] EVENTCODE [{}] id [{}] bacc [{}] payload [{}]",
          message.getTopic(), EVENT_CODE, message.getMsgID(), request.getBacc(), request);
    } catch (Exception e) {
      LogUtil.error(LOGGER, e, "Public a message, failure. TOPIC [{}] EVENTCODE [{}] bacc [{}] error [{}]",
          TOPIC, EVENT_CODE, request.getBacc(), e.getMessage());
      throw new TxnFlowException(PropertyConstant.CODE_SYS_ERR, "call msgBorker error", e);
    }
  }
```

接收事务消息

接收事务消息代码示例如下：

```
// 启动 consumer
  @Override
  public void afterPropertiesSet() throws Exception {
    Properties properties = new Properties();
    //替换 GID_PGROUP 为您实际创建的 Group ID
    properties.setProperty(PropertyKeyConst.GROUP_ID, "GID_SGROUP");
    properties.setProperty(PropertyKeyConst.LDC_SUB_MODE, "RZONE");
    Consumer consumer = OMS.builder().driver("sofamq").build().createConsumer(properties);
    //替换 TP_SCRCU_POC 为您实际创建的消息 Topic
    consumer.subscribe("TP_SCRCU_POC", "EC_ACCT_POINT", this);
    consumer.start();
  }

  // 消息
  public Action consume(Message message, ConsumeContext context) {
    try {
      PointAcctDTO pointAcctDTO = hessianSerializer.deserialize(message.getBody(),
          PointAcctDTO.class.getName());


      serviceTemplate.executeWithTransaction(pointAcctDTO, newServiceCallback() {

        @Override
        public String getServiceName () {
          return "returnPoint";
        }

        @Override
        public void checkParam () {

          //参数校验，如果为null或者空则消息不用重试
          pointAcctDTO.checkParameters();
        }

        @Override
        public void checkIdempotent () {
          //消息有可能出现重发的情况，如果这次积分已经更新过了，则直接返回成功。 消息幂等处理有问题
          List<TPointOrderDO> tPointOrderDOs = tPointOrderDAO.searchTxnSn(
```

```
                pointAcctDTO.getBacc(), pointAcctDTO.getTxnSn());
            if (tPointOrderDOs != null && tPointOrderDOs.size() != 0) {
                throwPcException.valueOf(PcStatusCode.IDEMPOTENT);
            }
        }

        @Override
        public void execute () {

            //埋点
            mockActionServices.mockAction(ActionPointConstant.POINTCENTER_MESSAGE,
                pointAcctDTO.getBacc());

            List<TPotAcctDO> tPotAcctDOs = tPotAcctDAO.lockForUpdate(pointAcctDTO.getBacc());

            //根据原来代码逻辑，如果没找到则打日志;
            if (tPotAcctDOs == null || tPotAcctDOs.size() != 1) {
                LogUtil.error(LOGGER, "error = {}, payload = {}",
                    PcStatusCode.BACC_COUNT_ERROR, "select t_pot_acct record:"
                        + tPotAcctDOs);
                return;
            }
            TPotAcctDO tPotAcctDO = tPotAcctDOs.get(0);

            //根据原来代码逻辑，如果状态不为0则直接返回。
            if (!PropertyConstant.ACC_STATUS_0.equals(tPotAcctDO.getStatus())) {
                return;
            }

            tPotAcctDO.setPotBal(tPotAcctDO.getPotBal().add(pointAcctDTO.getPotBal()));
            tPotAcctDO.setLastTxnSn(pointAcctDTO.getTxnSn());
            tPotAcctDAO.update(tPotAcctDO);
            TPointOrderDO tPointOrderDO = new TPointOrderDO(pointAcctDTO.getBacc(),
                pointAcctDTO.getPotBal(), pointAcctDTO.getTxnSn());
            tPointOrderDAO.insertOrder(tPointOrderDO);

        }
    });

} catch (CodecException e) {
    LogUtil.error(LOGGER, e, "consume pointAcctDTO={} exception.");
```

```
    }
    // do something
    return Action.CommitMessage;
}
```

分布式事务

在事务发起方的 spring 事务模板内，调用 `dtxService.start()` 方法开启分布式事务，需要在开启时，增加单元化架构下的分片参数，用做服务路由和数据库路由。

代码示例如下：

```
Map<String,Object> prooerties =newHashMap<String,Object>();

// 开启分布式事务
String shardingKey =UIDUtil.parseShardingKeyFromBacc(request.getBacc());
dtxService.start("accttrans", request.getTxnSn(), shardingKey, prooerties);
```

有关分布式事务单元化配置的更多信息，参见 接入单元化能力。

任务调度

您无需进行额外配置，即可使用单元化的任务调度能力，请参见 使用跨 zone 网关。

> 说明
>
> 建议检查客户端的启动参数是否正常传入： `- Dcom.alipay.ldc.zone=xxx //指定所属的逻辑单元` 。

API 网关

您无需进行额外配置，即可使用 API 网关的单元化路由能力，请参见 创建路由规则、创建 API。

应用参数配置

在应用部署时，您还需要传入以下参数。各参数均通过 JVM 的 -D 参数传入到应用中。

中间件相关参数

- `com.alipay.instanceid` ：当前租户中间件实例唯一标识，可以在消息队列控制台概览页 **接入配置 > 实例 ID** 中获取。
- `com.antcloud.antvip.endpoint` ：ACVIP 地址，可以在消息队列控制台概览页 **接入配置 > TCP 协议内网接入点** 中获取。
- `com.alipay.env` ：环境标识，取值 shared，表示运行在共享模式。
- `com.antcloud.mw.access` ：中间件访问控制键值。
- `com.antcloud.mw.secret` ：中间件访问控制密钥。

同城双活相关参数

- `com.alipay.ldc.zone` ：单元名称。
- `com.alipay.ldc.datacenter` ：物理机房名称。

LDC 单元化相关参数

- `zmode` ：LDC 单元化开关。取值为 true 时，表示开启 LDC 单元化。
- `com.alipay.ldc.strictmode` ：LDC 单元化严格模式开关。取值为 true 时，表示开启 LDC 单元化严格模式。

5.步骤四：构建并发布应用

本文指导您如何通过单元化应用服务（LHC）将本地开发好的单元化应用发布到双机房中的部署单元。

> 说明
>
> 本文仅适用于公有云及新版（AntStack 2.2 以后）专有云环境。

步骤一：创建命名空间

1. 登录控制台，在左侧导航栏单击 **集群管理 > 命名空间**，进入命名空间列表页。
2. 单击 **创建** 进入 **创建命名空间** 页面，输入以下信息，单击 **创建**。
   - **名称**：输入命名空间名称，例如 `antcloud-demo`。默认名称的前缀为 `租户名+工作空间组标识`。
   - **添加标签**：为命名空间添加自定义标签。

步骤二：创建应用

您将创建 3 个应用：txnflow、acccenter、pointcenter。

1. 登录应用管理控制台，在左侧导航栏单击 **应用列表**。
2. 在 **应用列表** 页面，单击 **创建应用**。
3. 在 **创建应用** 页面输入以下信息，单击 **提交**。
   - **应用名称**： 为三个应用分别输入名称：`txnflow`、`acccenter`、`pointcenter`。

     > 说明
     >
     > 应用名称在同一租户内必须是唯一的。

   - **技术栈**：选择 Spring Boot。
   - **应用分组**：选择已创建的应用分组。若没有提前创建，可使用系统默认分组。
   - **应用标签**：本例中无需添加。
   - **应用描述**：本例中无需添加。

步骤三：准备镜像

LHC 以镜像的方式部署应用服务，创建应用服务前需先构建好镜像。应用开发完成后可以直接在 LHC 控制台进行镜像构建。参见 开发单元化应用、构建镜像。

步骤四：创建应用服务

创建 3 个有依赖关系的应用服务：txnflowsvcdemo、pointcenter-demo、accountcentersvc-demo。

1. 登录控制台，在左侧导航栏单击 **发布运维 > 应用服务**。
2. 在应用服务列表页，单击 **创建应用服务**。
3. 在 **创建应用服务** 页面，填写以下基本信息，单击 **下一步**。
   - **命名空间**：选择步骤一创建的命名空间（namespace）。

- **应用服务名称**：容器服务的名称。服务实例名称允许包含（小写）字母、数字、连字符，且必须以字母开头，以字母或数字结尾，同一个命名空间下不允许同名。为三个应用服务分别输入名称：`txnflowsvcdemo`、`pointcenter-demo`、`accountcentersvc-demo`。
- **所属应用**：选择一个该容器服务所关联的应用。

应用服务名称	所属应用
txnflowsvcdemo	txnflow
pointcenter-demo	pointcenter
accountcentersvc-demo	acccenter

  - **描述**：选填。容器服务的描述。

4. 在 **Pod 模板配置** 页面，填写以下信息，单击 **下一步**。
   - `txnflowsvcdemo` 的配置如下：
     - **容器名称**：输入名称。
     - **镜像选择**：选择 **构建记录**，选择 步骤三：准备镜像 中的构建记录。
     - **CPU 配置**：**请求核数** 为 1 core，**最大核数** 为 2 core。
     - **内存配置**：**请求内存** 为 1 GiB，**最大内存** 为 2 GiB。
     - 在 **高级配置 > 环境变量配置** 中，为 txnflowsvcdemo 应用服务添加所需的 **环境变量**。
     - **配置覆盖**：为 txnflowsvcdemo 应用服务添加所需的覆盖配置。
   - `pointcenter-demo` 的配置如下：
     - **容器名称**：输入名称。
     - **镜像选择**：选择 **构建记录**，选择 步骤三：准备镜像 中的构建记录。
     - **CPU 配置**：**请求核数** 为 1 core，**最大核数** 为 2 core。
     - **内存配置**：**请求内存** 为 1 GiB，**最大内存** 为 2 GiB。
     - 在 **高级配置 > 环境变量配置** 中，为 pointcenter-demo 应用服务添加所需的 **环境变量**。
     - **配置覆盖**：为 pointcenter-demo 应用服务添加所需的覆盖配置。
   - `accountcentersvc-demo` 的配置如下：
     - **容器名称**：输入名称。
     - **镜像选择**：选择 **构建记录**，选择 步骤三：准备镜像 中的构建记录。
     - **CPU 配置**：**请求核数** 为 2 core，**最大核数** 为 4 core。
     - **内存配置**：**请求内存** 为 4 GiB，**最大内存** 为 6 GiB。
     - 在 **高级配置 > 环境变量配置** 中，为 accountcentersvc-demo 应用服务添加所需的 **环境变量**。
     - **配置覆盖**：为 accountcentersvc-demo 应用服务添加所需的覆盖配置。
5. 在 **弹性配置** 页面，填写以下信息，单击 **下一步**。

副本伸缩策略配置：目前仅支持**固定副本数**，默认为 0，勾选部署单元（ `RZ01A` 、 `RZ02A` ）并修改为期望副本数 1，即应用服务运行时保持固定数目的 Pod 副本。

6. 在 **访问配置** 页面，填写以下信息，单击 **下一步**。

   应用服务支持两种访问方式：负载均衡、统一接入。负载均衡是基于端口的请求负载均衡，统一接入是基于规则的请求负载均衡。

   为 txnflowsvcdemo 服务设置公网访问方式

   > **说明**
   >
   > pointcenter-demo、accountcentersvc-demo 服务无需添加服务。

   您可以在创建应用服务时设置访问方式，也可以应用服务创建完成后添加访问方式。

   i. 在 **访问配置** 页面，单击 **添加负载均衡**，填写以下信息后，单击 **提交**。

      - **负载均衡名称**：输入 `txnflowsvcdemo` 。
      - **访问方式**：选择 **外网**。
      - **端口映射**：单击 **添加端口映射**，填写以下信息。其余保持默认设置。
        - **协议**：选择 http。
        - **转发规则**：选择 **RR 轮询**。
        - **前端端口**：容器镜像中工作负载实际监听的端口，端口范围为 1-65535。
        - **后端端口**：保持默认根目录。容器端口映射到负载均衡实例上的端口，用负载均衡 IP 访问工作负载时使用，端口范围为 1-65535。

7. 在 **部署和调度配置** 页面，为应用服务均保持系统默认配置，单击 **下一步**。
8. 在应用服务 **预览** 页面，确认信息无误，单击 **提交**。

步骤五：创建发布单

通过发布单同时发布上一步中创建的三个应用服务。

1. 登录控制台，在左侧导航栏单击 **发布运维 > 应用发布**。
2. 单击 **发布单** 进入发布部署大盘。
3. 在发布部署大盘，单击 **创建发布单**。
4. 在 **创建发布单** 页面，填写以下发布信息后，单击 **下一步**。
   - **基本信息**
     - **标题**：发布标题。例如 `Demodeploy` 。
     - **类型**：仅支持分组发布。
     - **命名空间**：选择待发布的应用服务所属的命名空间。

- **应用服务发布列表**：在 **待选应用服务列表** 中单击选择需要的应用服务。单击 > 图标将应用服务添加到 **已选应用服务列表** 中。

  > **说明**
  >
  > 若应用服务存在多个提交版本，需要选择要发布的版本。默认选择最新的版本。

- **高级配置**

  设置应用服务依赖关系：无需设置。

5. 在 **预览** 页面确认信息无误后，单击 **创建**。系统会自动跳转到 **发布单详情** 页面，单击 **整体发布** 即可开始发布。

   发布单创建完成后，系统会自动跳转到 **发布单详情** 页面，您可以查看发布单的执行详情。

6.步骤五：到控制台验证

基于 开发单元化应用 中的转账与积分场景，在完成相应的转账或存款等操作后，您可以前往各个产品控制台验证单元化效果：

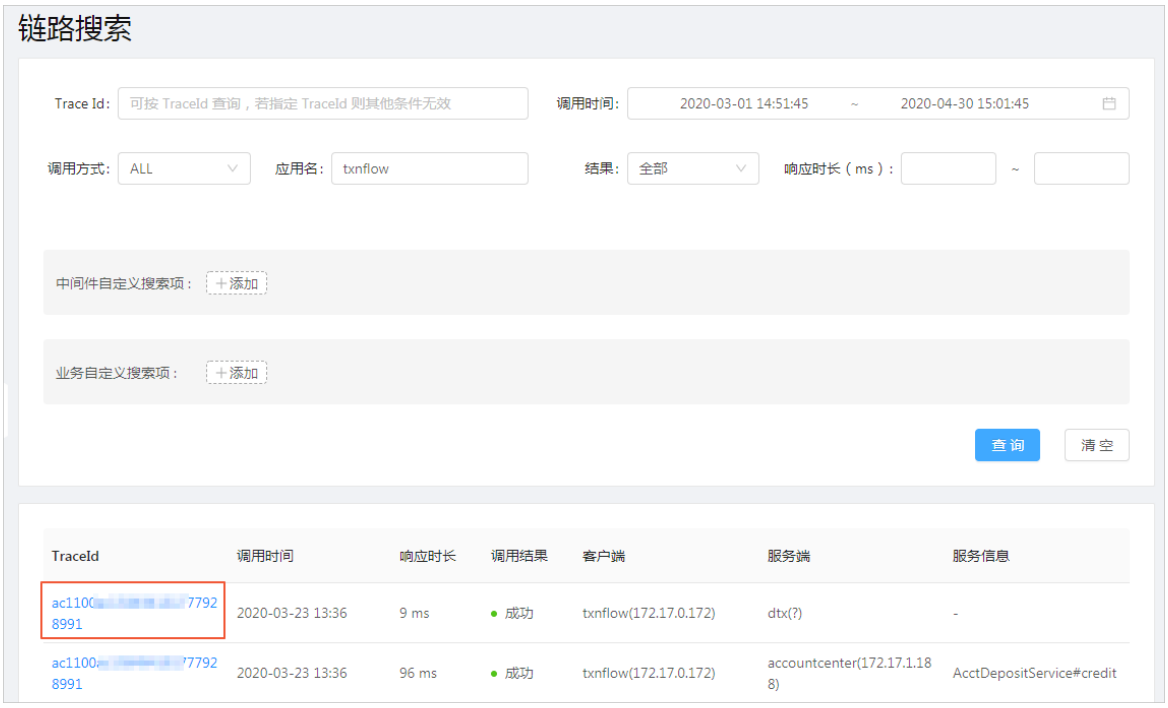
- 分布式链路跟踪：查看单元间流量走向
- 分布式事务：查看事务执行情况
- 消息队列：查看消息轨迹
- 微服务：按单元推送动态配置
- 实时监控：查看单元间流量对比

查看单元间流量走向

在转账场景中，您可以通过分布式链路跟踪查看具体的流量分布情况。

操作步骤如下：

1. 登录分布式链路跟踪控制台，左侧导航栏中选择 **链路搜索**。
2. 在 **链路搜索** 页面，根据实际情况输入相应的搜索条件，单击 **查询**。例如在 **应用名** 处，输入转账发起方应用名，如 tnxflow。
3. 在搜索处的链路列表中，找到最新的那条链路，单击 TraceId，查看该链路的详细信息。



4. 在 **链路详情** 页中，您可以查看到该链路单元间的流量分布情况。
   - 如这笔转账业务是在 zone 内实现的，您可以看到链路 **单元化** 列显示 **单元内**，去向 zone 与来源 zone 相同。

- 如这笔转账业务是跨 zone 实现的，您可以看到链路 **单元化** 列显示 **跨单元**，去向 zone 与来源 zone 不同。

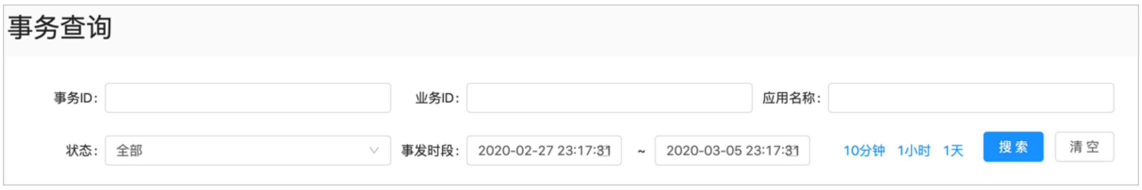
应用名	SpanId	IP	调用类型	状态	机房	单元化	服务信息	耗时
- txnflow	0	1[redacted]5	MVC	成功	-	-	POST /txnflow/accttrans	257ms
txnflow	0.1	1[redacted]15	DB	成功	nmgpoc01	-	INSERT poc_tf	21ms
DB@poc_tf	0.1	1[redacted]6	DB	成功	nmgpoc01	-	INSERT poc_tf	21ms
- txnflow	0.2	1[redacted]5	UNKNOWN	成功	nmgpoc01	单元内	-	230ms
txnflow	0.2.1	1[redacted]15	UNKNOWN	成功	nmgpoc01	-	-	13ms
txnflow	0.2.2	1[redacted]15	UNKNOWN	成功	nmgpoc01	-	-	23ms
txnflow	0.2.3	1[redacted]5	SOFARPC	成功	nmgpoc01	跨单元	AcctDrawService#debit	30ms
- accountcenter	0.2.3	1[redacted]3	SOFARPC	成功	nmgpoc01	跨单元	AcctDrawService#debit	26ms
accountcenter	0.2.3.1	1[redacted]3	DB	成功	nmgpoc02	-	SELECT poc_at	4ms

查看事务执行情况

对于转账场景，您还可以通过分布式事务中间件查看这笔转账事务的详细信息。

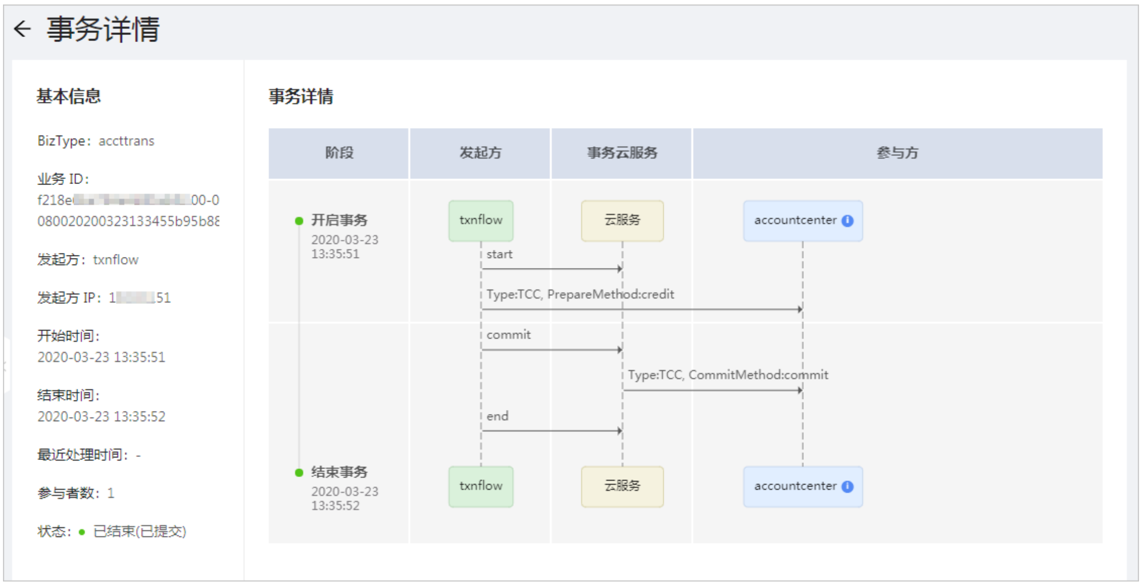
操作步骤如下：

1. 登录分布式事务控制台，左侧导航栏中选择 **事务监控 > 事务查询**。
2. 在 **事务查询** 页面，您可以通过以下查询条件搜索出您想要的事务：
   - 事务 ID：想要查询的事务的唯一标识。
   - 业务 ID：业务的唯一标识。
   - 应用名称：输入事务对应的应用名称，如 tnxflow。
   - 事务状态：包括全部、进行中、异常、已完成。
   - 事发时段：可以根据需求快速选取最近时间范围（ 10 分钟、1 小时、1 天），也支持自定义时段。



3. 在所有的搜索结果列表中，找到您最近转账的那笔事务，单击右侧的 **详情**。
4. 在打开的 **事务详情** 页面中，您可以查看到事务的基本信息，以及整个分布式事务的执行过程。
   - 基本信息：包括 BizType、发起方 IP、业务 ID、事务状态、发起时间、结束时间、参与方数量等。

- 事务详情：分布式事务的开启、结束时间、发起方、参与方信息、事务的操作过程、参与方类型等。

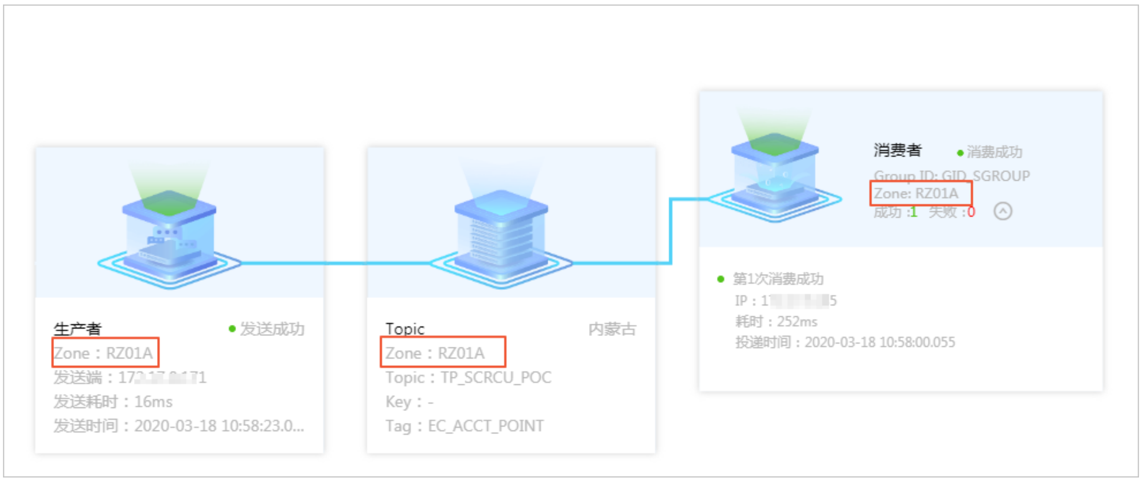


查看消息轨迹

在积分的场景中，您可以进入消息队列控制台查询该消息并查看其消息轨迹。

操作步骤如下：

1. 登录消息队列控制台，左侧导航栏中选择 **消息轨迹**。
2. 在 **消息轨迹** 页面，单击 **创建查询任务**。
3. 在弹出的 **创建查询任务** 对话框中，按需选择单元、查询条件并按提示输入信息。例如您可以选择 **按 Topic 查询**，并输入该事务消息的 Topic。
4. 单击 **确定**。查询任务创建完成后，您可以在消息轨迹页面查看到该任务。
5. 单击任务前面的展开图标可查看到轨迹的简要信息，主要是消息本身的属性以及接收状态的信息。
6. 在展开的区域框中，单击 **查看轨迹** 即可查看完整的消息轨迹图。图中主要展现了以下信息：
   - 生产者信息：生产者所在 Zone、发送端 IP 地址、发送时间、发送耗时、发送状态。
   - Topic 信息：Topic 所在 Zone、Topic 名称、消息的业务标识 key、消息标签 tag。
   - 消费者信息：消费者所在 Zone、耗时、投递时间、消费状态。

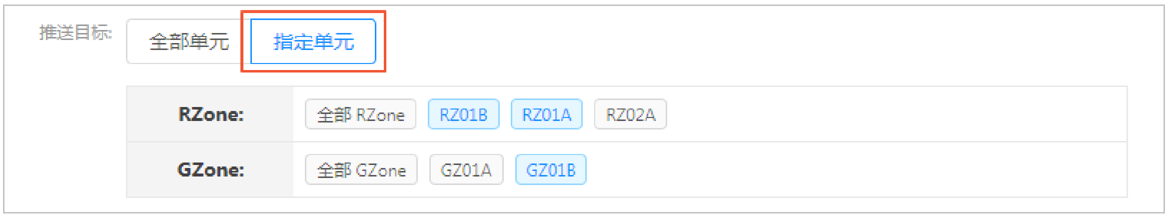


按单元推送动态配置

在转账场景中，您还可以通过微服务按单元推送动态配置。

操作步骤如下：

1. 登录微服务控制台，左侧导航栏中选择**动态配置**。
2. 在配置类列表中，单击要推送的配置类前的加号，展开属性列表。
3. 单击要推送的属性名称，进入属性详情页面。
4. 在基本信息区域中的 **推送目标** 栏，选择 **指定单元** 后，选择目标单元。



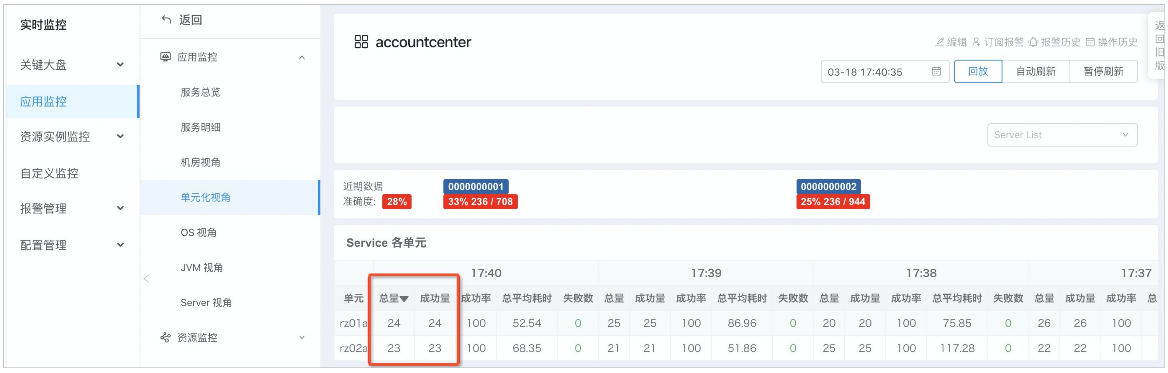
5. 输入推送值，单击 **推送配置** 并确定推送。

查看单元间流量对比

您可以通过实时监控，查看应用单元间的流量对比。

操作步骤如下：

1. 登录实时监控控制台，左侧导航中选择 **应用监控**。
2. 找到对应的应用，如 accountcenter，单击 **详情** 进入应用详情页。
3. 单击左侧导航 **应用监控** > **单元化视角**，查看单元间的数据对比，如下图所示。



7.步骤六：调拨应用流量

本文介绍如何使用 LHC 做应用层流量的调拨。通过调整并下发接入层和应用层的分片规则，验证流量路由和 RPC 路由的准确性。

> 说明
>
> 该教程一般适用于灰度测试场景。

前提条件

- 已配置应用层的分片规则如下：
  - RZ01A：00-49
  - RZ02A：50-99
- 账号准备：
  - 账号 A：UID 分片位为 08（属于 RZ01A 分片）
  - 账号 B：UID 分片位为 38（属于 RZ01A 分片）

步骤一：调整和下发分片规则

1. 登录 LHC 控制台。
2. 在左侧导航栏，单击 **流量管理** > **应用层**，进入应用层流量管理页面。
3. 单击 **规则配置** 进入 **正式流量规则配置** 页面，该页面展示的是当前已配置的规则。
4. 单击逻辑单元卡片或分片数据即可进入编辑页面，修改对应部署单元的 UID 分片为如下配置：
   - RZ01A：00-29
   - RZ02A：30-99

   > 说明
   >
   > 调整分片规则后，账号所属的逻辑单元发生变化如下： 账号 A：08（属于 RZ01A 分片） 账号 B：38（属于 RZ02A 分片）

5. 修改后单击 **保存** 以确保配置变更完成。
6. 单击 **流量规则推送**，进入 **流量规则推送** 页面。
7. 设置以下推送配置，将最新规则推送至中间件和 ALB 管控的所有单元。
   - **规则详情**：系统会根据当前配置的流量规则自动生成 JSON 格式的规则文本，该文本会作为实际推送到应用或中枢服务的参数，让全局应用或指定应用感知到流量变化。
   - **部署单元**：选择 **全部部署单元**。
   - **目标组件**：选择 **中间件** 及 **ALB 管控**。一般情况下流量规则的变更都需要让这两个组件感知，除非一些灰度测试场景。
   - **目标应用**：选择需要感知规则变化的目标应用，一般来说会选择全局应用推送，但在灰度或测试场景也可以只让部分应用感知新的流量规则做一些流量验证。

    - **是否覆盖其他应用**：选择推送到全局应用时，若开启覆盖其他应用，会用新配置的流量规则覆盖掉应用当前生效的规则。不覆盖，则会保留之前应用单独推送的规则。

8. 单击 **推送**，开始推送当前配置完的流量规则。

步骤二：执行转账操作

模拟执行账号 A 发起对账号 B 的转账操作。

步骤三：查看业务链路

可以通过分布式链路跟踪（DST）查看具体的流量分布情况，在调整分片规则后，预期A到B的转账流量，将从**单元内**变为**跨单元**。

操作步骤如下：

1. 登录分布式链路跟踪控制台，左侧导航栏中选择 **链路搜索**。
2. 在 **链路搜索** 页面，根据实际情况输入相应的搜索条件，单击 **查询**。例如在 **应用名** 处，输入转账发起方应用名。
3. 在搜索处的链路列表中，找到最新的那条链路，单击 TraceId，查看该链路的详细信息，检查流量走向是否更新为 **跨单元**。