**版本号：1.0.0**

**文档信息**

|  |  |
| --- | --- |
| **项目名称:** |  |
| **标题:** |  |
| **版本:** | 1.0.0 |
|  |  |
| **作者:** | liujiaming |
| **创建日期:** | 2025/01/01 |
| **上次更新日期:** |  |
|  |  |
| **分类:** |  |
| **关键词:** |  |
|  |  |
| **审批人:** |  |
| **审批日期:** |  |

**修改记录**

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| 2013/08/19 | 1.0.0 | 创建文档 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**目录**

[1. 引言 1](#_Toc364934559)

[1.1. 目的 1](#_Toc364934560)

[1.2. 背景 1](#_Toc364934561)

[1.3. 术语与缩写解释 1](#_Toc364934562)

[1.4. 参考资料 1](#_Toc364934563)

[2. 总体设计 2](#_Toc364934564)

[2.1. 需求规定 2](#_Toc364934565)

[2.1.1. 系统功能 2](#_Toc364934566)

[2.1.2. 系统性能 2](#_Toc364934567)

[2.1.2.1. 精度 2](#_Toc364934568)

[2.1.2.2. 时间特性要求 2](#_Toc364934569)

[2.1.2.3. 可靠性 2](#_Toc364934570)

[2.1.2.4. 灵活性 2](#_Toc364934571)

[2.1.3. 输入输出要求 2](#_Toc364934572)

[2.1.4. 数据管理能力要求 2](#_Toc364934573)

[2.1.5. 故障处理要求 2](#_Toc364934574)

[2.1.6. 其他专门要求 3](#_Toc364934575)

[2.2. 运行环境 3](#_Toc364934576)

[2.2.1. 设备 3](#_Toc364934577)

[2.2.2. 支持软件 3](#_Toc364934578)

[2.2.3. 接口 3](#_Toc364934579)

[2.2.4. 控制 3](#_Toc364934580)

[2.3. 基本设计概念和处理流程 3](#_Toc364934581)

[2.4. 结构 3](#_Toc364934582)

[2.5. 功能需求与系统模块的关系 3](#_Toc364934583)

[2.6. 人工处理过程 4](#_Toc364934584)

[2.7. 尚未解决的问题 4](#_Toc364934585)

[3. 接口设计 5](#_Toc364934586)

[3.1. 用户接口 5](#_Toc364934587)

[3.2. 外部接口 5](#_Toc364934588)

[3.3. 内部接口 5](#_Toc364934589)

[4. 运行设计 6](#_Toc364934590)

[4.1. 运行模块组合 6](#_Toc364934591)

[4.2. 运行控制 6](#_Toc364934592)

[4.3. 运行时间 6](#_Toc364934593)

[5. 系统数据结构设计 7](#_Toc364934594)

[5.1. 逻辑结构设计要点 7](#_Toc364934595)

[5.2. 物理结构设计要点 7](#_Toc364934596)

[5.3. 数据结构与程序的关系 7](#_Toc364934597)

[6. 系统出错处理设计 8](#_Toc364934598)

[6.1. 出错信息 8](#_Toc364934599)

[6.2. 补救措施 8](#_Toc364934600)

[6.3. 系统维护设计 8](#_Toc364934601)

# 引言

## 目的

实现根据用户自定义的交易信息，实现对其交易质量的监控，统计，告警，分析。

## 背景

行内bpc系统会分析网络报文，解析出交易信息，云上链路，天生支持交易信息收集。故，对标bpc，实现云上，系统间调用的交易质量监控。

## 术语与缩写解释

|  |  |
| --- | --- |
| **缩写、术语** | **解 释** |
| 评断标准 | 泛指如果评判一笔交易的成功与否。在当前体系下，指的是如果返回码不为预期，预期可设置，则认为是失败的交易。 |
| 维度 | 一种聚合查询的方式，自定义维度，指的是用户可以根据自己的实际情况，定制自己的查询条件。 |
| 交易 | 业务上一笔涉账请求到响应的全部流程。在链路体系下，trace中的第一条，也就是头节点的描述，代表了交易的状态。 |
| 交易码/交易名/交易url | 行内分布式体系下，一笔交易指的是网关节点的url请求。非分布式体系下，则是交易码。cpaas 中统一使用交易的url作为交易的名称。 |
| 交易中文名 | 交易url的中文名词，该名称在edsp二级注册中心维护 |
| 消费 | 指的是处理来自kafka的数据的行为 |
| 消费者 | kafka的consumer以及消费这个动作的发起者的一种抽象表达方式 |
|  |  |

## 参考资料

bpc-oda 的实现

# 总体设计

## 需求规定

用例图

图示

描述已自动生成

### 系统功能

2.1.1.1 配置交易评判标准

需要使用方配置好交易评判标准，即什么样的交易是异常交易如图：

我们认为，默认AAAAAA 和 s 开头的（具体数据格式，需要poin提供样例），是成功交易；如果在图中配置，例如bbbb，

那么AAAAAA ，bbbb和s 开头的都是成功的交易

2.1.1.2 交易概览（原型图）

第三步：如图，配置好交易维度，其中新零售中的交易码和交易渠道为固定的交易维度

应用名，为固定的内置维度，主要是给cpaas自己用

点击开启交易质量监控

图形用户界面, 应用程序

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成

点击保存，保存维度。点击发送，则是发送给oda。权限为项目管理员及以上

2.1.1.4 交易质量多维度查询

2.1.1.4 热加载

点击确认后，query组件

2.1.1.5 数据过滤

2.1.1.6 数据缓存

2.1.1.7 成功率流式计算

如图，我们在redis

### 系统性能

需要**benchmark**

#### 精度

需要**benchmark**

#### 时间特性要求

oda系统对交易质量的告警有着较高的实效性要求。即，需要在交易质量过低时，1min内，反映到oda项目的dashboard上。故不能用日志中心的logstash，和 ElasticSearch来实现数据聚合，因为logstash的延迟难以保证。也不能用prometheus，因为prometheus server的抓取周期和内存以及量的关系，现阶段无法评估。

#### 可靠性

在现有线路的数据流上，新加了一条专门处理交易信息的数据流。该数据流即便是crash了，也不会影响到主逻辑数据流，也就是现有功能不会因为交易信息功能crash导致不可用。

#### 灵活性

具有热加载功能。可以动态调节需要监控，告警的交易维度。

### 输入输出要求

输入

输出

### 数据管理能力要求

### 故障处理要求

### 其他专门要求

## 运行环境

### 设备

### 支持软件

### 接口

### 控制

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

# 业务处理流程

## 设置交易评判标准

原型图：

图形用户界面, 应用程序

描述已自动生成

如图：

图示

描述已自动生成

页面原型如图

从左往右分别是：

交易url+edsp中文名，当前交易的交易码，红色为错误，白色为正确。最大一共五个；交易总数。其中交易总数没有体现，但经过分析，认为该数据应该在页面体现；该页面的具体加载逻辑如下

图示

描述已自动生成

功能点介绍：

1. 基础增删查改
2. 修改业务返回码

图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成

* 1. 支持业务返回码编辑。
  2. 支持正确和错误码添加，其中业务返回码的左右，是用来判断交易的状态，交易状态分成功/异常/未知判断规则如下

图示

描述已自动生成

* 1. 保存返回码后，查询，统计将会以该条件甄别哪些交易属于哪些类型。（后边会有详细介绍）该规则是后面所有展示页面的底层逻辑。
  2. 添加自定义维度，维度本身详见聚合查询页面。可以为某个交易，添加上特定的查询维度，该维度本期不做告警。维度最多支持两个，取poin数据中的的biz.expend字段中的前两个。如果配置错误，例如非法json。则该功能无效。如果配置过多的的维度，（biz.expend 本身，也就是项目组本身poin和cpaas无法约束）则不保证数据一定符合预期（见聚合查询页面）。

图形用户界面

中度可信度描述已自动生成

图表

中度可信度描述已自动生成

* 1. 设置告警信息（告警本身详见3.1 告警设计）。保存后，将会决定改交易是否上报至oda系统。如果配置了该页面，点击确认后，会将数据发送至oda系统，如果oda系统设置了告警规则，则会发出相应的告警。

图形用户界面, 文本, 应用程序

描述已自动生成

* 1. 页面上可以看到哪些交易有告警设置。
  2. 有基础的筛选逻辑，例如时间，项目等。

3 交易类型统计页面

3.1 增加交易概览页面

增加dashboard 参照响应质量直方图，包括但不限于横纵

3.2 增加交易概览指标，共计8个

|  |  |  |  |
| --- | --- | --- | --- |
| 指标名称 | 描述（不显示在页面上） | 指标名称 | 描述 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

4.1 增加交易质量筛选页面

原型图：



4.2 多维查询功能

1. 多维查询项，将原始数据的交易内容提取出来，作为维度和查询条件。将计算指标作为聚合结构展示，流程如下：



指标计算公式以及含义如下

维度含义如下

接口路径url：交易的名称

上一跳：交易的上一跳

第一跳：交易的第一跳，通常是自己。

返回码：业务返回码

服务名：服务的名称。有可能性风险

原地址、端地址：由poin的数据提供，取addr.local 中的字段，但端地址获取逻辑，仍在调研，暂时无法取到。

自定义1,2：用户在交易设置页面时候的自定义的维度

 **交易总次数**  
一般统计网关入口span的数量（如HTTP请求入口）；一段时间内链路中某个维度（如某接口）对应的总调用次数

 **异常交易次数**  
根据具体业务定义异常（业务返回码不符合预期）。一段时间内调用异常的次数，通常表现为错误、异常状态码或业务错误

 **平均响应时间**  
一般仅统计成功调用的响应时间，以避免异常或超时调用对平均值影响过大。一段时间内所有成功调用的平均耗时，单位通常为毫秒

 **响应数与span超时时间关系**  
响应数是一个特殊指标，与设定的超时时间阈值有关，超过该阈值的span视为超时调用，不计入响应数。一段时间内在指定超时时间内返回响应的调用数目（即未超时调用的数量）

 **成功率计算**  
成功率通常为成功调用次数与总调用次数的比值，异常调用包括业务异常、系统异常或超时异常。一段时间内调用成功的比例（%形式）

1. 多级聚合
2. 点击某个维度会展开多级聚合结构

## 基本设计概念和处理流程

注意这里用了trans\_type 而不是urlpath。根据分布式体系架构，urlpath 是交易的具体表å象，但是，在实际交易中，不可避免地碰到交易码这个概念，链路侧需要兼容二者。故，以交易类型trans\_type，泛指url path ，交易码，中文描述。其中，中文描述，会继承edsp的能力，在cpaas中表象是urlpath 和交易码的中文。

设置页面基础数据结构如下  
 文本, 聊天或短信

描述已自动生成

Interval 交易监控间隔，单位为秒；

channel 交易渠道，保留字段，只有极少数会用到，仅仅做保留字段

Podname： 监控维度之一

TransType：交易类型，大部分情况下是url path

RetCode：交易返回码，对应的是poin的ServiceCode。当交易的ServiceCode in RetCode时候，视作成功交易

其他：查询使用使用

如图：下面设置好RetCode 等内容，并持久化到db 中。

配置好后，视作开启交易质量展示；；

当配置好交易质量展示后，ElasticSearch的索引mapping中，会增加额外信息，如图  
trans\_info:

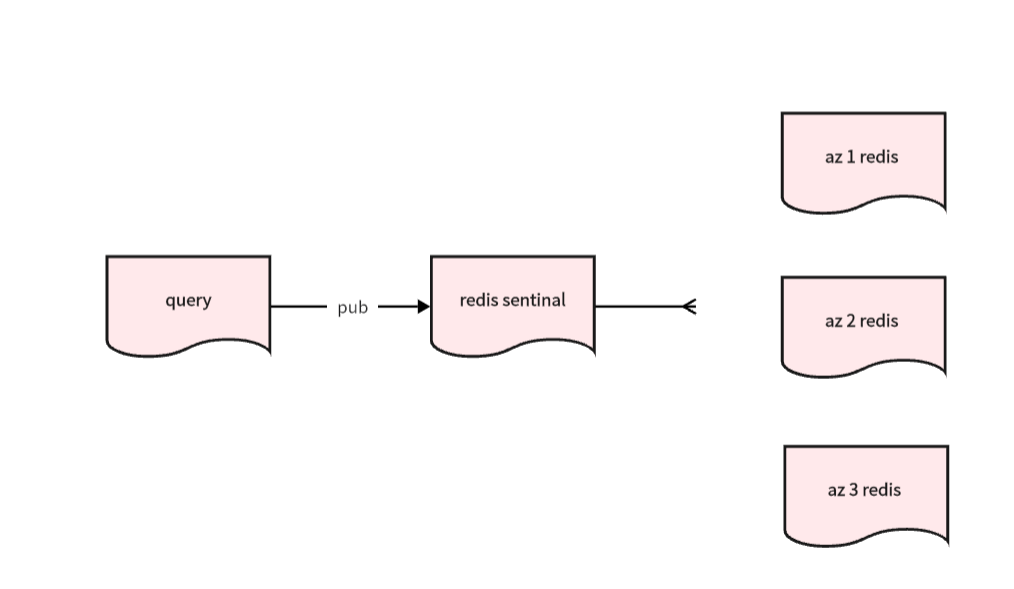
当文档中有trans\_info时候，无论业务如何，均视为开启交易质量展示。

从上面可以看出，重点是时效性和数据流的处理。因此，将难点主要分成以下几块。

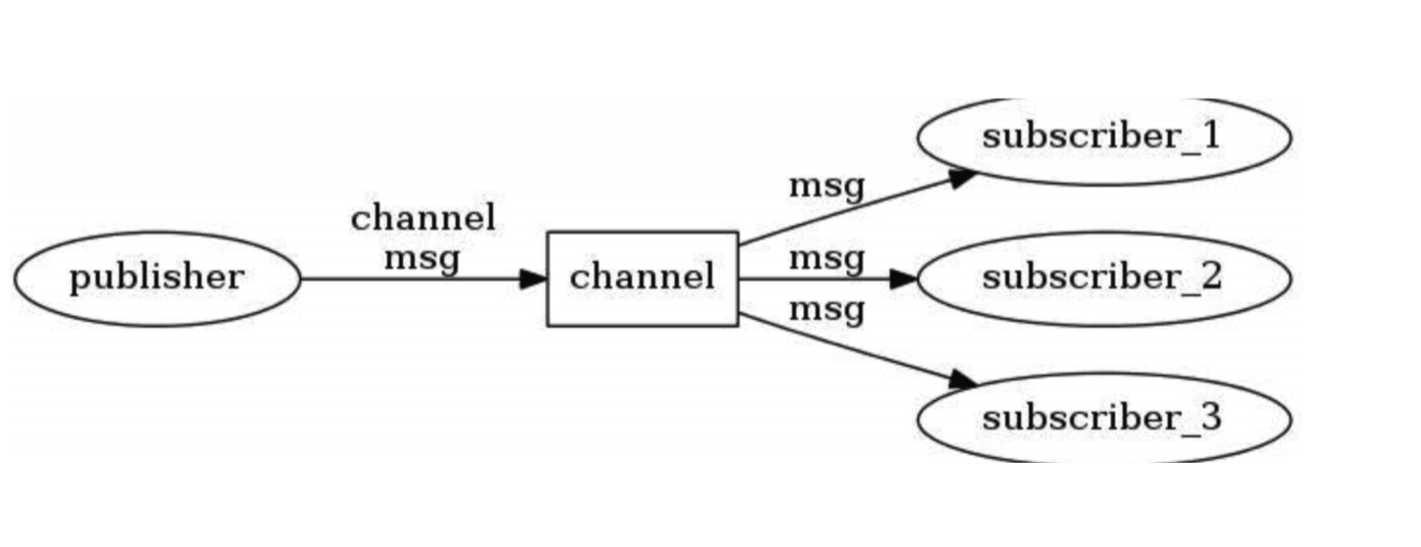
如图，引入了新的流水线机制和热加载机制。

**第一步：发布信息，当我们配置好维度后，点击发送**

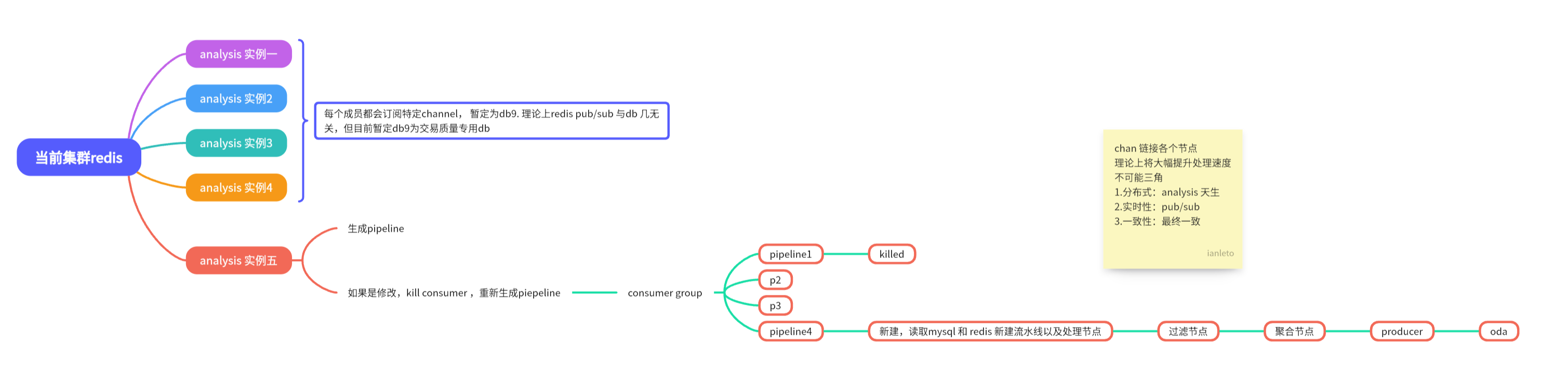
点击发送后，会发送事件到redis



这个事件，会包含必要信息广播到所有的analysis组件



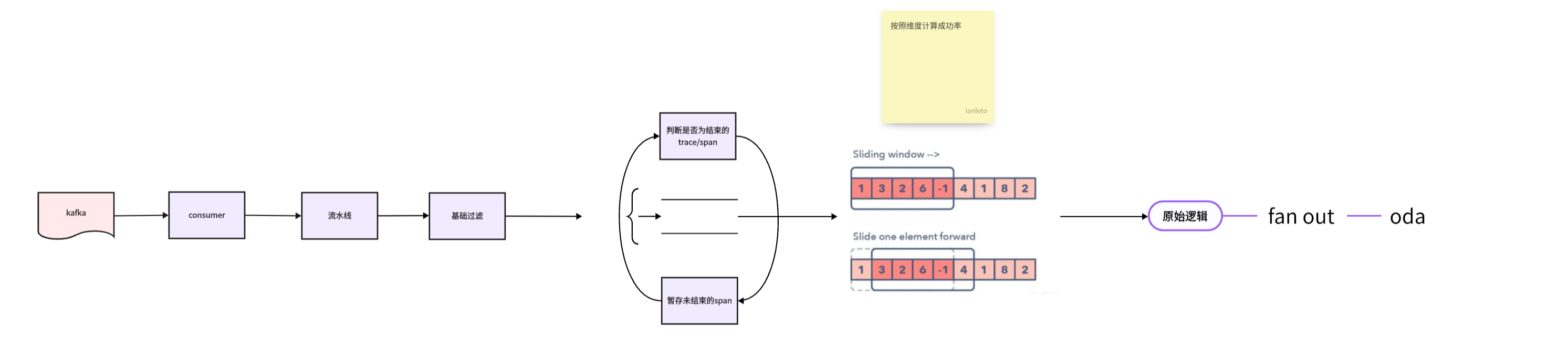
当所有订阅者拿到消息后，会开始分析，最终消费该项目数据的实例生成、或者kill掉自己的pipeline（数据流水线，一个抽象的，自己实现流式处理机制）如图实例五



拿到事件后，analysis会开始创建数据流水线，完成热加载功能

**第二步：数据流水线**

当analysis实例拿到事件后，会着手将数据发送至oda。此时会经历数据流水线，处理流式数据。因为如果将1min钟数据直接存入内存，会导致内存飙升，导致程序直接crash，并且对性能要求很高故此设计利用golang语言天生的流式线模型处理，如图逻辑架构

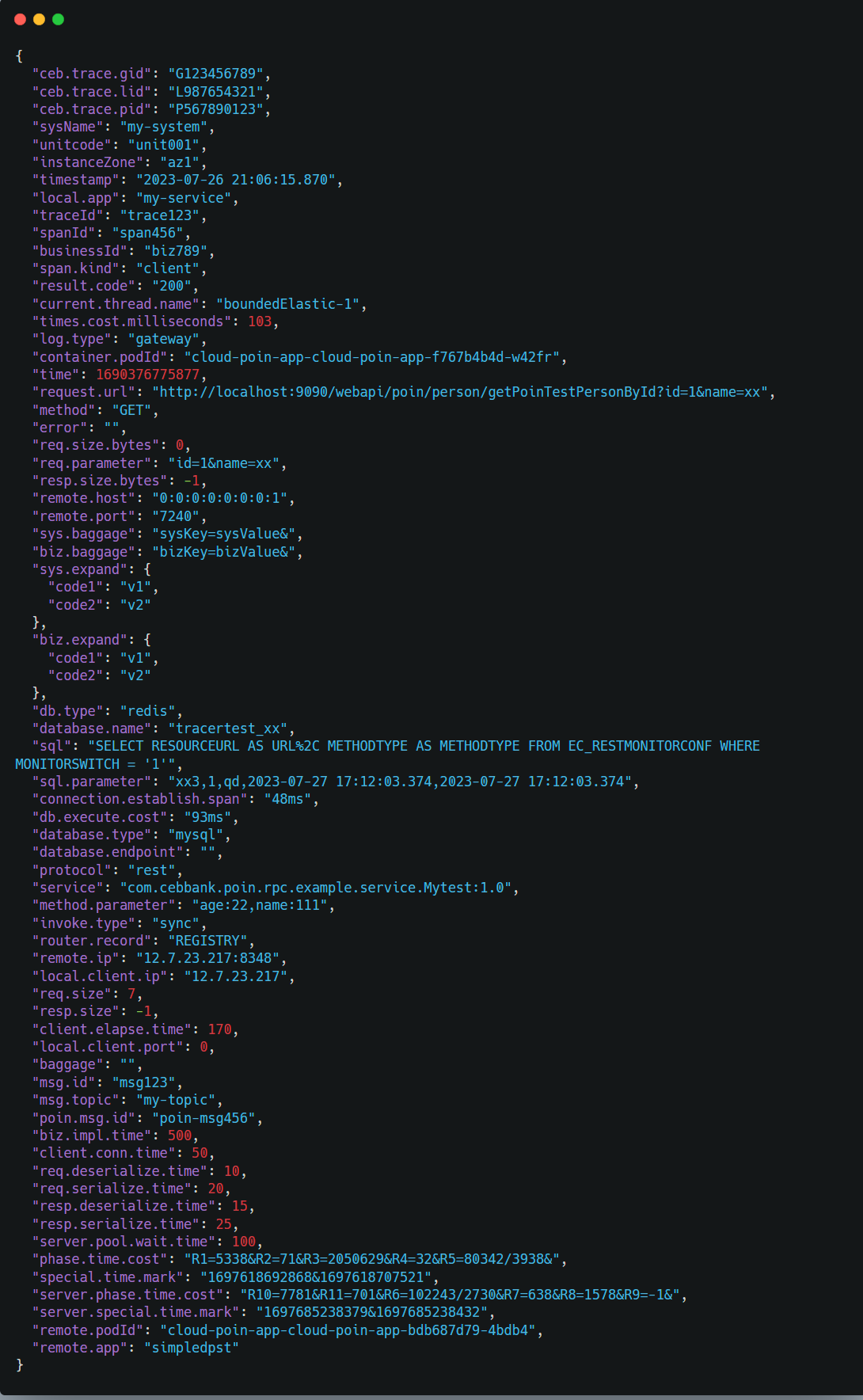


数据进入流水线之前，我们会根据db中的内容，为流水线配置

1. 过滤节点
2. 判断逻辑
3. 结束节点
4. 聚合计算节点

过滤节点：

我们会过滤掉脏数据，并将数据缩小如图



清洗成

判断节点：

流水线建立好后，会初始化判断逻辑，例如，交易码为AAAA的为正常数据，否则视为异常交易

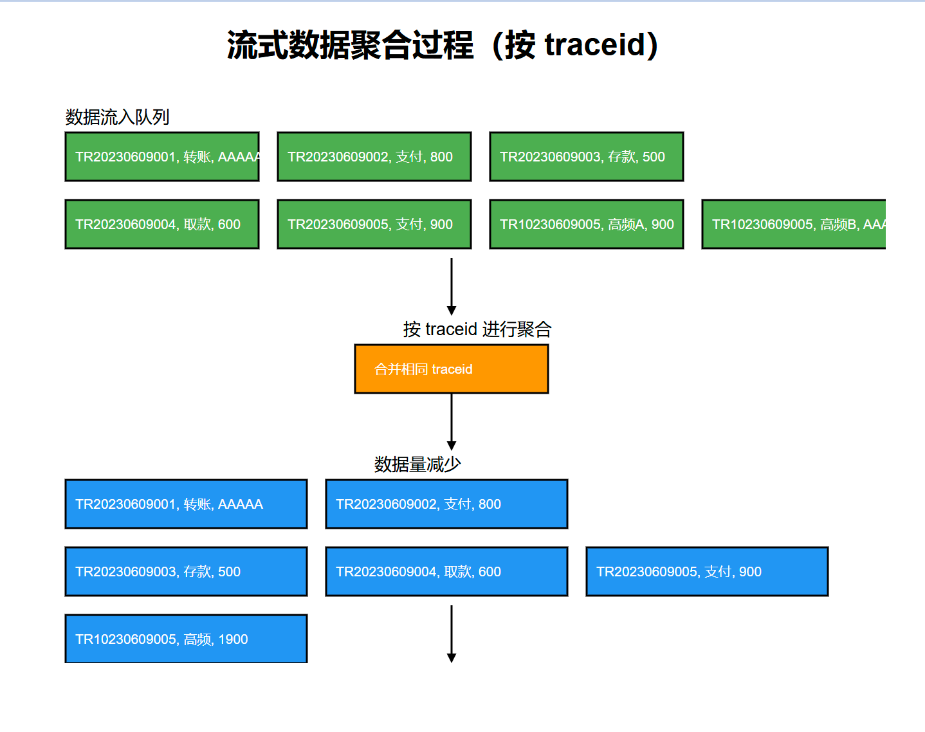
结束节点：

数据流经这个节点，当未检测到头span时候，则视为该交易未结束，进程，会停滞在缓存中，指导超时或者检测到头span，则认为交易完成

**聚合计算节点（重要）：**

**聚合计算节点，指的是一些列用来处理流式数据的节点，流程如下**

**1** 按traceid先聚合一次数据

 按照traceid，我们聚合了一次数据，注意，此处经过调研，不会存在单笔交易不同渠道（即组合交易），至少目前不会支持这种。因此无须考虑组合聚合问题

当数据第一次聚合过程中，会判断，如果是AAAAA 则视为成功数据

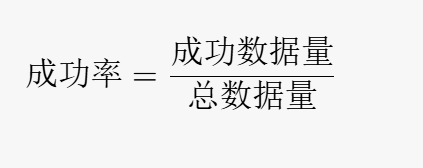
因此得到数据traceid 是否成功的标识

2. 数据进入滑动窗口，开始进行组合聚合

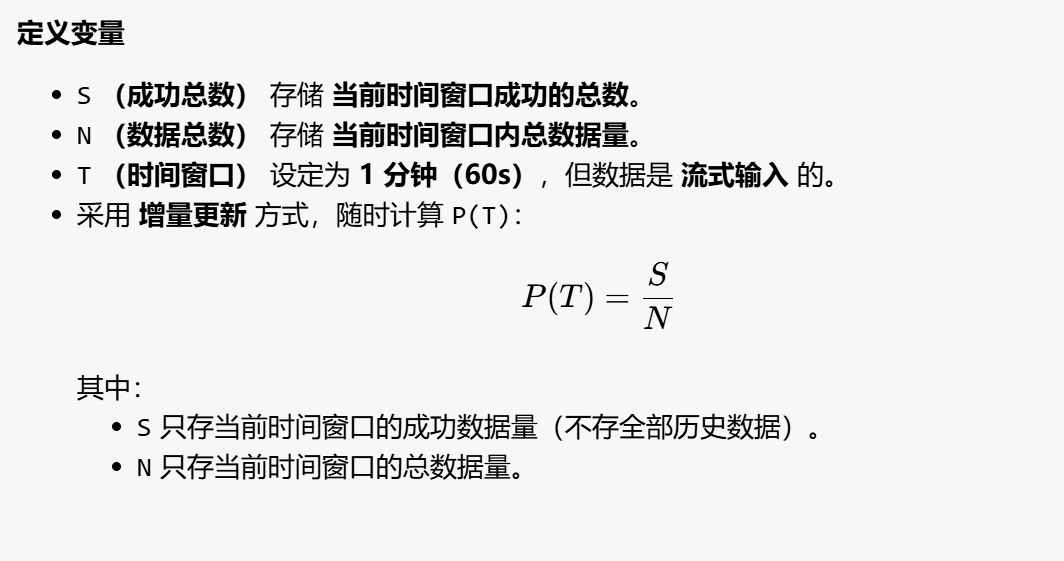
如图示，写入到oda

我们希望计算 **一分钟内所有流经的数据的成功率**，其中：

* **每满 100 条数据** 或 **每 10s 计算一次**（但可能数据未满 100 条）。
* **成功率定义为**：



因此得到公示如下



我们以计算笛卡尔积中的一个维度的一个聚合项来举例

数据每写入100条，或者过了10s，队列会被打满，去计算

图形用户界面, 文本, 网站

描述已自动生成

这个队列中，包含一个队列，每个项如下

然后，我们去根据交易渠道去聚合

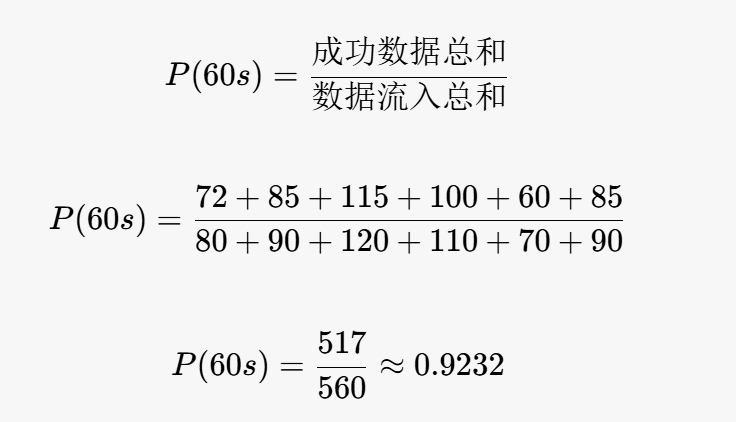
得到如图 转账渠道的成功数为72

图示

描述已自动生成

然后，从时间角度来看，转账渠道的内存使用和成功率计算

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 时间 (秒) | 数据量 | 成功量 | 计算触发 |  |  |
| 0 - 10s | 80 | 72 | ✅ 聚合计算 | 数据不在内存中 |  |
| 10 - 20s | 90 | 85 | ✅ 聚合计算 | 原始数据不在内存中 |  |
| 20 - 30s | 120 | 115 | ✅ 聚合计算 | 原始数据不在内存中 |  |
| 30 - 40s | 110 | 100 | 计算中 | 原始数据在内存中 |  |
| 40 - 50s | 70 | 60 | ✅ 聚合计算 | 原始数据不在内存中 |  |
| 。。。。 | 。。。 | 。。。 | 。。。 | 。。 |  |



此时，我们计算出了，转账渠道在一分钟内的成功率为92.32%；且理论内存占用为，当计算时间可忽略且窗口为10s 一次计算时，**理论内存使用量降低为实际值的六分之一**

重复计算笛卡尔积，得到全部组合聚合的结果，示例入右图

## 结构

[给出系统结构总体框图（包括软件、硬件结构框图），说明本系统的各模块的划分，扼要说明每个系统模块的标识符和功能，分层次地给出各模块之间的控制与被控制关系。]

## 功能需求与系统模块的关系

[本条用一张矩阵图说明各项功能需求的实现同各模块的分配关系。]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **[系统模块1]** | **[系统模块2]** | **[……]** | **[系统模块m]** |
| [功能需求1] | √ |  |  |  |
| [功能需求2] |  | √ |  |  |
| [┇] |  |  |  |  |
| [功能需求n] |  |  |  | √ |

## 人工处理过程

[说明在本系统的工作过程中不得不包含的人工处理过程。]

## 尚未解决的问题

[说明在概要设计过程中尚未解决而设计者认为在系统完成之前必须解决的各个问题。]

# 接口设计

## 用户接口

[说明将向用户提供的命令和它们的语法结构，以及相应的回答信息。

说明提供给用户操作的硬件控制面板的定义。]

## 外部接口

[说明本系统同外界的所有接口的安排包括软件与硬件之间的接口、本系统与各支持系统之间的接口关系。]

## 内部接口

[说明本系统之内的各个系统元素之间的接口的安排。]

# 运行设计

## 运行模块组合

[说明对系统施加不同的外界运行控制时所引起的各种不同的运行模块组合，说明每种运行所历经的内部模块的支持软件。]

## 运行控制

[说明每一种外界的运行控制的方式方法和操作步骤。]

## 运行时间

[说明每种运行模块组合将占用各种资源的时间。]

# 系统数据结构设计

[不涉及软件设计可不包含。]

## 逻辑结构设计要点

[给出本系统内软件所使用的每个数据结构的名称、标识符以及它们之中每个数据项、记录、文卷和系的标识、定义、长度及它们之间的层次的或表格的相互关系。]

## 物理结构设计要点

[给出本系统内软件所使用的每个数据结构中的每个数据项的存储要求，访问方法、存取单位、存取的物理关系、设计考虑和保密条件。]

## 数据结构与程序的关系

[说明各个数据结构与访问这些数据结构的各个程序之间的对应关系。]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **[程序1]** | **[程序2]** | **[……]** | **[程序m]** |
| [数据结构1] | √ |  |  |  |
| [数据结构2] |  | √ |  |  |
| [┇] |  |  |  |  |
| [数据结构n] |  |  |  | √ |

# 系统出错处理设计

## 出错信息

[用一览表的方式说明每种可能的出错或故障情况出现时，系统输出信息的形式、含意及处理方法。]

## 补救措施

[说明故障出现后可能采取的变通措施。包括：

* 后备技术：说明准备采用的后备技术，当原始系统数据万一丢失时启用的副本的建立和启动的技术，例如周期性地把磁盘信息记录到磁带上去就是对于磁盘媒体的一种后备技术。
* 降效技术：说明准备采用的后备技术，使用另一个效率稍低的系统或方法来求得所需结果的某些部分，例如一个自动系统的降效技术可以是手工操作和数据的人工记录。
* 恢复及再启动技术：说明将使用的恢复再启动技术，使软件从故障点恢复执行或使软件从头开始重新运行的方法。]

## 系统维护设计