TGIP-CN 026

Pulsar Flink 连接器的介绍与使用

赵建云
StreamNative

Who am I

- 赵建云（Jianyun Zhao）
- Apache Pulsar contributor
- StreamNative Software Engineer

Pulsar Flink 连接器

- Pulsar介绍
- Flink介绍
- 连接器介绍
- 实战

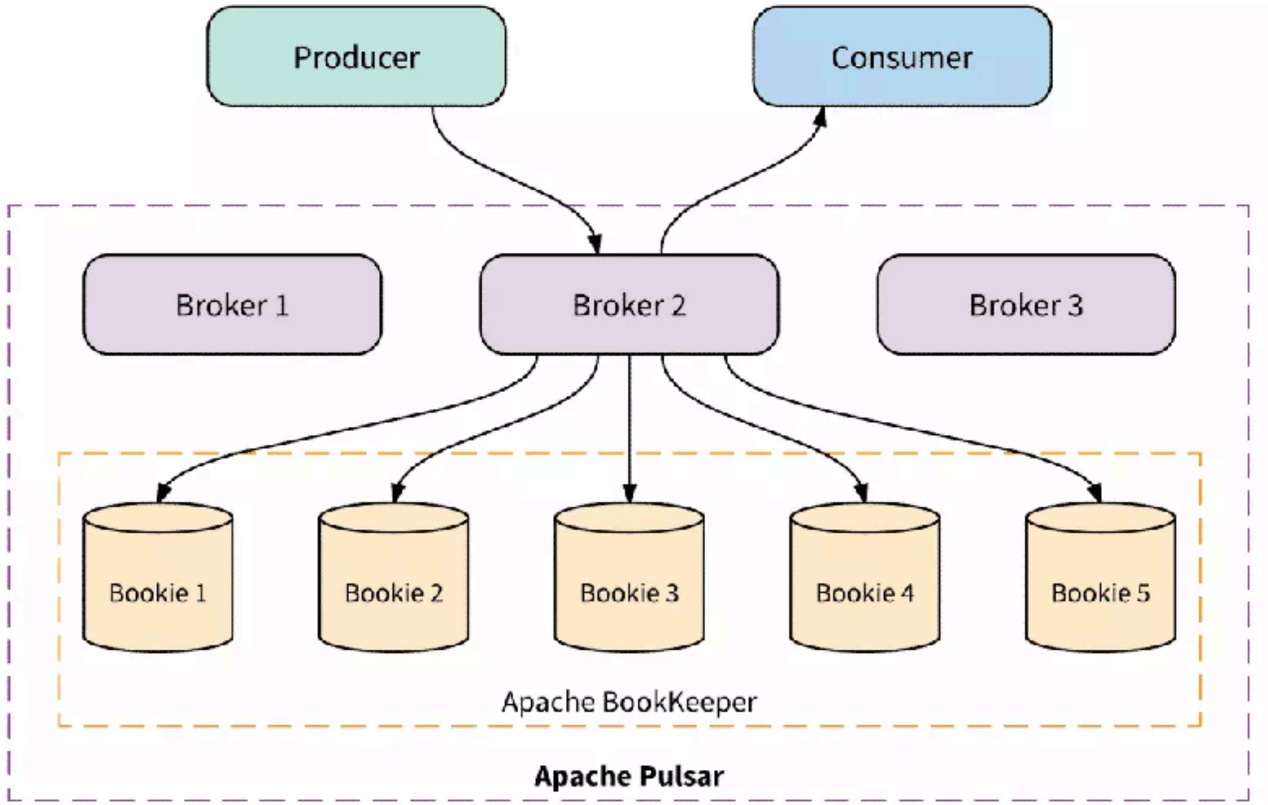
Pulsar

Apache Pulsar 是云原生的事件流平台

连接、存储、数据处理

- 连接：Pulsar clients，IO connectors，Protocol handlers
- 存储: Apache BookKeeper，Tiered storage
- 数据处理:
  - Pulsar Function - 轻量无服务计算
  - Spark、Flink - 统一数据处理
  - Presto - SQL交互式查询

Pulsar 云原生架构



无状态服务层
Stateless serving

数据持久层
Durable storage

Pulsar 有结构的数据

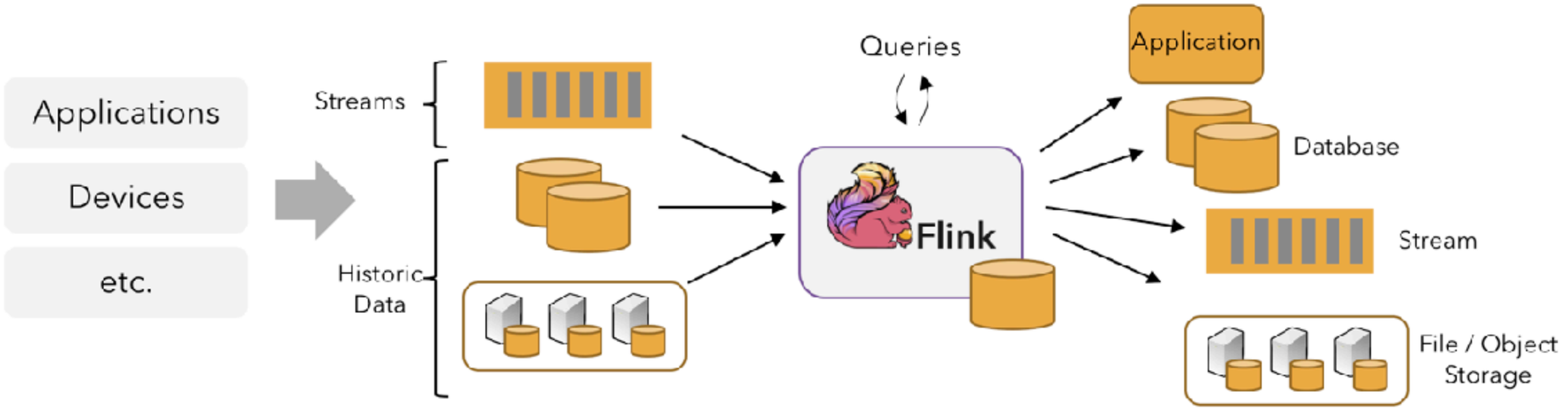
- 内置的Schema注册
  - 在服务端的消息服务共识
  - Topic级别的消息结构
- 直接产生、消费有结构的数据
  - Pulsar进行消息验证
  - 支持消息版本的演化

Flink

Apache Flink 是一个在无界和有界数据流上进行状态计算的框架和分布式处理引擎。 Flink 可以在所有常见的集群环境中运行，并以 in-memory 的速度和任意的规模进行计算。

Flink

Flink 框架，大致可以分为三块内容，从左到右依次为：数据输入、Flink 数据处理、数据输出。

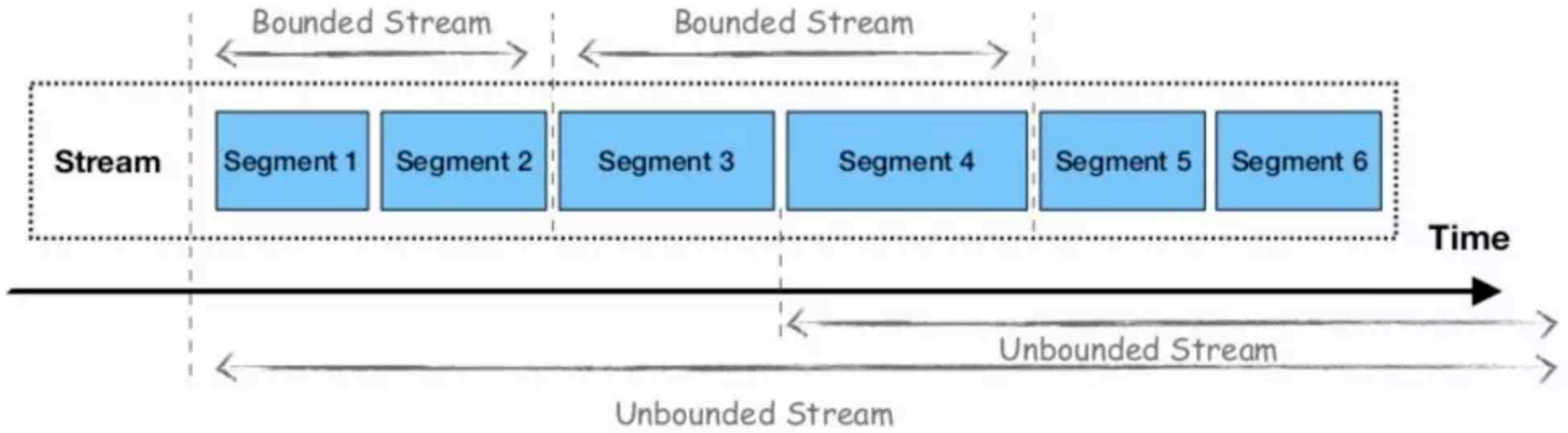


X

Apache Flink是当下最流行的数据计算引擎，Apache Pulsar是消息订阅系统中的翘楚。当它们遇到一起时，会发生什么有趣的事情呢?

PULSAR x Flink

Data Processing on Pulsar



相似点

- 批处理是有界的数据流
- 结构化数据

X

Pulsar 能以不同的方式与 Apache Flink 融合，一些可行的融合包括：

- 使用流式连接器（Streaming Connectors）支持流式工作负载，
- 使用批式源连接器（Batch Source Connectors）支持批式工作负载。
- Pulsar 还提供了对 Schema 的原生支持，可以与 Flink 集成并提供对数据的结构化访问，例如，使用 Flink SQL 在 Pulsar 中查询数据。

从架构的角度来看，我们可以想象两个框架之间的融合，使用 Apache Pulsar 作为统一的数据层视图，使用 Apache Flink 作为统一的计算、数据处理框架和 API。

Pulsar Flink 连接器

https://github.com/streamnative/pulsar-flink

支持Flink 1.9或更高的版本

支持Flink Stream、Batch、Table、Catalog功能

exactly-once 语义的 Source 和 at-least-once 语义的 Sink

Pulsar Topic动态发现

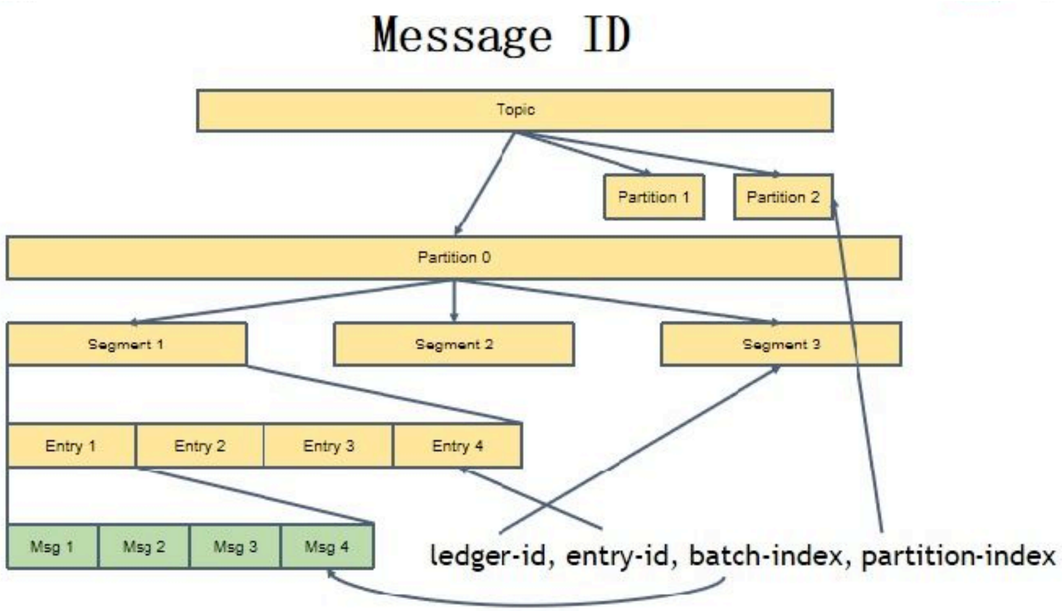
Pulsar Schema支持

Pulsar Flink 连接器

Pulsar Flink Connector 在使用上是比较简单的，由一个 Source 和一个 Sink 组成，Source 的功能就是将一个或多个主题下的消息传入到 Flink 的Source中，Sink的功能就是从 Flink 的 Sink 中获取数据并放入到某些主题下，在使用方式上，如下所示：

```
StreamExecutionEnvironment see = StreamExecutionEnvironment.getExecutionEnvironment();
Properties props = new Properties();
props.setProperty("topic", "test-source-topic");
props.setProperty("partitiondiscoveryintervalmillis", "5000");

FlinkPulsarSource<String> source = new FlinkPulsarSource<>(serviceUrl, adminUrl, new SimpleStringSchema(), props);
source.setStartFromEarliest();
DataStream<String> stream = see.addSource(source);

FlinkPulsarSink<Person> sink = new FlinkPulsarSink(
  serviceUrl,
  adminUrl,
  Optional.of(topic),     // mandatory target topic or use `Optional.empty()` if sink to different topics for each record
  props,
  TopicKeyExtractor.NULL,  // replace this to extract key or topic for each record
  Person.class,
  RecordSchemaType.AVRO);

stream.addSink(sink);
```

精确一次



Pulsar 中的 MessageId 是全局唯一且有序的，并且对应Pulsar 中的实际物理存储，因此实现 Exactly Once， 只需要结合Flink 的 Checkpoint 机制，将 MessageId 存储到 Checkpoint即可。

对于连接器的 Source 任务，在每次触发 Checkpoint 的时候，会将各个分区当前处理的 MessageId 保存到状态存储里面，这样在任务重启的时候，每个分区都可以通过 Pulsar 提供的 Reader seek 接口找到 MessageId 对应的消息位置，然后从这个位置之后读取消息数据。

通过 Checkpoint 机制，还能够向存储数据的节点发送数据使用完毕的通知，从而能准确删除过期的数据，做到存储的合理利用。

Topic动态发现

流数据分析应用是长时间执行的，因此在分析应用执行期间，Topic 的分区或者用户订阅的 Topic 会动态增删。为了使我们的流计算应用可以感知这种变化，可以启动一个定时任务，定期检查是否新增 Topic，并启动 reader 处理数据。

Pulsar Schema支持

Pulsar 支持定义 Avro schema和写入消费 avro、json、protobuf等格式的Message，使得消息结构化。这样带来了Pulsar SQL、Function等生态支持，用户可以在更多的系统中操作数据。Pulsar Flink 连接器中Source、Sink对于Pulsar Schema均做了支持，使数据在Pulsar、Flink中流转，保持了一致结构化，而不是冰冷冷的bytes。

良好的拓展性

对于Source，我们通常不会满足Message中只拿到了value数据，而是会希望能获取到Message的消息发布时间、扩展Properties属性等。 Pulsar Flink连接器提供了自定义的数据解码接口，用户选择可以实现PulsarDeserializationSchema来实现自己的需求。

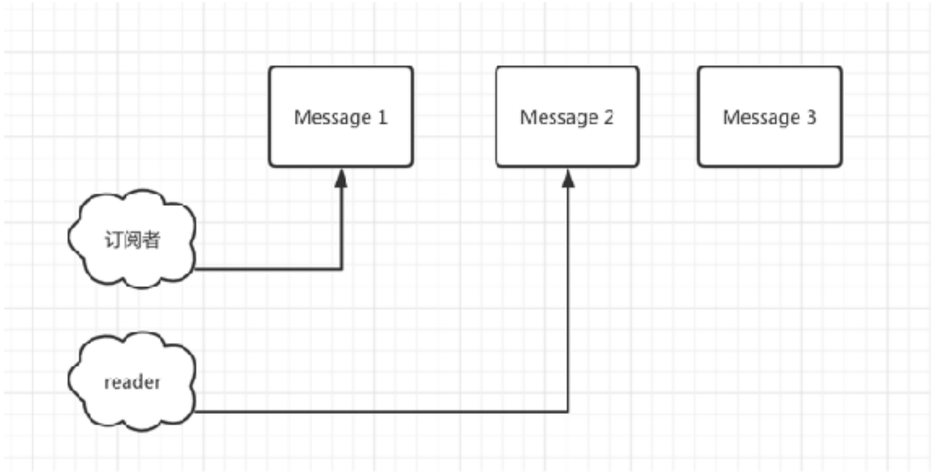
对于Sink，支持使用每条记录中的topic来输出消息

```
public FlinkPulsarSource(
        String adminUrl,
        ClientConfigurationData clientConf,
        PulsarDeserializationSchema<T>
deserializer,
        Properties properties) {
    \\ 初始化FlinkPulsarSource参数
}

public FlinkPulsarSink(
        String adminUrl,
        Optional<String> defaultTopicName,
        ClientConfigurationData clientConf,
        Properties properties,
        TopicKeyExtractor<T> topicKeyExtractor,
        Class<T> recordClazz,
        RecordSchemaType recordSchemaType) {
}
```

Stream

```
StreamExecutionEnvironment see =
StreamExecutionEnvironment.getExecutionEnvironment();
Properties props = new Properties();
props.setProperty("topic", "test-source-topic");
props.setProperty("partitiondiscoveryintervalmillis", "5000");

FlinkPulsarSource<String> source = new FlinkPulsarSource<>(serviceUrl,
adminUrl, new SimpleStringSchema(), props);
source.setStartFromEarliest();
DataStream<String> stream = see.addSource(source);
```

Source工作模式



Catalog

Flink会在当前目录和数据库中搜索表，视图和UDF。 要使用Pulsar Catalog并将Pulsar中的topic作为Flink中的表对待，flink home中的 ./conf/sql-client-defaults.yaml 中定义pulsarcatalog。

```
catalogs:
- name: pulsarcatalog
    type: pulsar
    default-database: public/default
    service-url: "pulsar://localhost:6650"
    admin-url: "http://localhost:8080"
```

注意：由于删除操作具有危险性，catalog中删除`tenant/namespace`、`Flink`操作不被支持。

```
tableEnv.useCatalog("pulsarcatalog")
tableEnv.useDatabase("public/default")
tableEnv.scan(“topic0")
```

```
Flink SQL> USE CATALOG pulsarcatalog;
Flink SQL> USE `public/default`;
Flink SQL> select * from topic0;
```

SQL、DDL

```
create table test_flink_sql(
    `rip` VARCHAR,
    `rtime`  VARCHAR,
    `uid`    bigint,
    `rchannel`    VARCHAR,
    `be_time`    bigint,
    `be_time`    VARCHAR,
    `activity_id` VARCHAR,
    `country_code`    VARCHAR,
    `os` VARCHAR,
    `recv_time`  bigint,
    `remark` VARCHAR,
    `client_ip`  VARCHAR,
    `day` as TO_DATE(rtime),
    `hour` as date_format(rtime, 'HH')
) with (
    'connector.type' = 'pulsar',
    'connector.version' = '1',
    'connector.topic' = 'persistent://test/test-gray/test_flink_sql',
   'connector.service-url' = 'pulsar://xxx',
    'connector.admin-url' = 'http://xxx',
    'connector.startup-mode' = 'external-subscription',
    'connector.sub-name' = 'test_flink_sql_v1',
    'connector.properties.0.key' = 'pulsar.reader.readerName',
    'connector.properties.0.value' = 'test_flink_sql_v1',
    'connector.properties.1.key' = 'pulsar.reader.subscriptionRolePrefix',
    'connector.properties.1.value' = 'test_flink_sql_v1',
    'connector.properties.2.key' = 'pulsar.reader.receiverQueueSize',
    'connector.properties.2.value' = '1000',
    'connector.properties.3.key' = 'partitiondiscoveryintervalmillis',
    'connector.properties.3.value' = '5000',
    'format.type' = 'json',
    'format.derive-schema' = 'true',
    'format.ignore-parse-errors' = 'true',
    'update-mode' = 'append'
);
```

新特性

- Pulsar 的 key-shared 订阅模式，支持更细的粒度订阅消息。
- 支持Flink Batch，直接从BookKeeper获取数据，提高数据的处理效率和降低资源占用。
- Pulsar client Auth认证支持

目前动向

Pulsar Flink 连接器正在积极的合并回Flink社区。

扫码入群！

Pulsar 中国社区等你加入

添加好友后，回复“进群”即可

Thanks