FLINK FORWARD # ASIA 2022 ONLINE | 实时即未来

Flink Connector 社区新动向与开发指南

New Features in Flink Connectors and Development Guide

任庆盛 | 阿里巴巴开发工程师 Apache Flink Committer
罗根 | 阿里巴巴技术专家 Apache Flink Contributor

01 Flink Connector 社区动向

02 Source 新功能

03 Sink V2 API 与小文件合并

01 Flink Connector 社区动向

Flink Connector 社区动向

旧的管理模式

- Connector 和 Flink 版本发布周期绑定，迭代速度慢
- Flink 代码库过于庞大，维护压力大
- 不利于 Flink 生态的发展

新的代码管理方式

每个 Connector 拥有独立的仓库，代码与 Flink 仓库分离管理

新的版本管理模式

以尽量保持和 Flink 的兼容性为原则，由 Connector 自行维护版本

代码管理方式

Apache Organization 下
独立的代码仓库

文档置于 Connector 仓库
脚本链接至 Flink 官网

使用 Apache JIRA
管理 Issue

样例：Flink ElasticSearch Connector

- https://github.com/apache/flink-connector-elasticsearch

提示

- 代码属于 Apache 软件基金会，需要遵守 Apache 2.0 协议
- 代码需要 Apache Flink Committer 的评审

贡献新的 Flink Connector

起草 FLIP

在 Flink Wiki 页面创建新的 FLIP

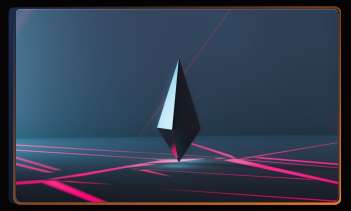
描述 Connector 支持的功能

邮件列表讨论

将 FLIP 发表在 Flink 社区开发者邮件列表中进行讨论

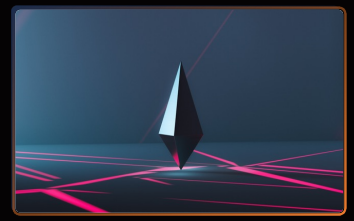
社区投票

讨论完成后在 Flink 开发者邮件列表中发起投票



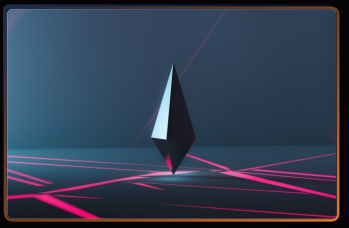
创建仓库

投票通过后联系 Committer 创建 Connector 仓库



完成代码

在代码准备完成后提交 PR，Committer 评审通过后合并



外部 Connector 版本管理

版本号 Major . Minor . Patch - Flink Version 3.1.2-1.16

版本支持 2 个 Major Connector 版本 + 2 个 Flink 版本

较旧版本支持问题修复

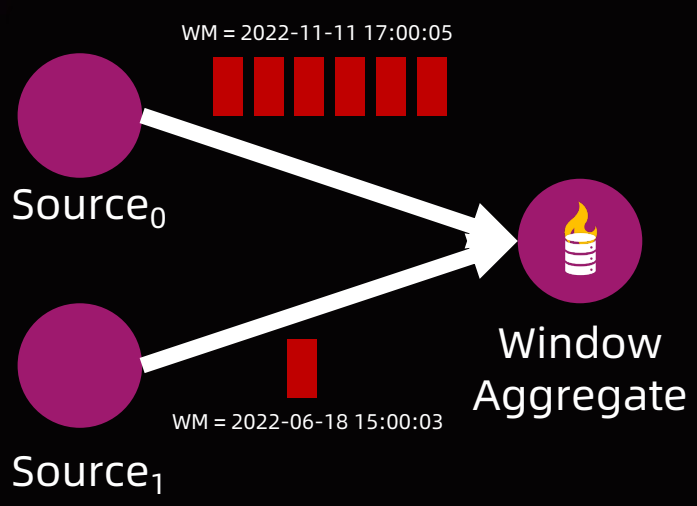
较新版本支持新功能

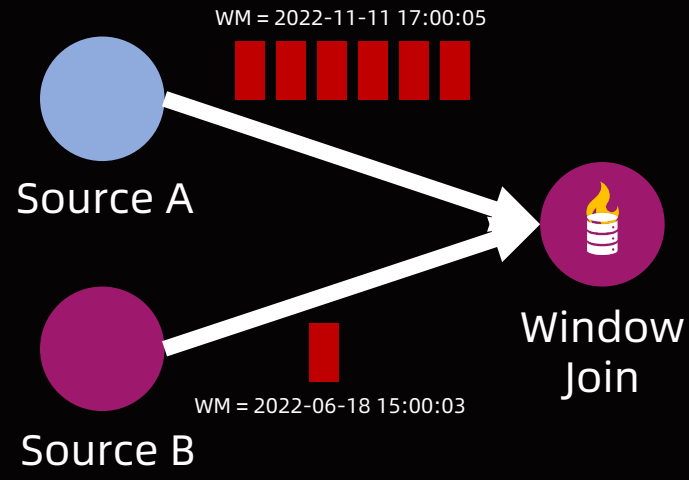
版本管理文档：

https://cwiki.apache.org/confluence/display/FLINK/Externalized+Connector+development

02 Source 新功能

水印对齐





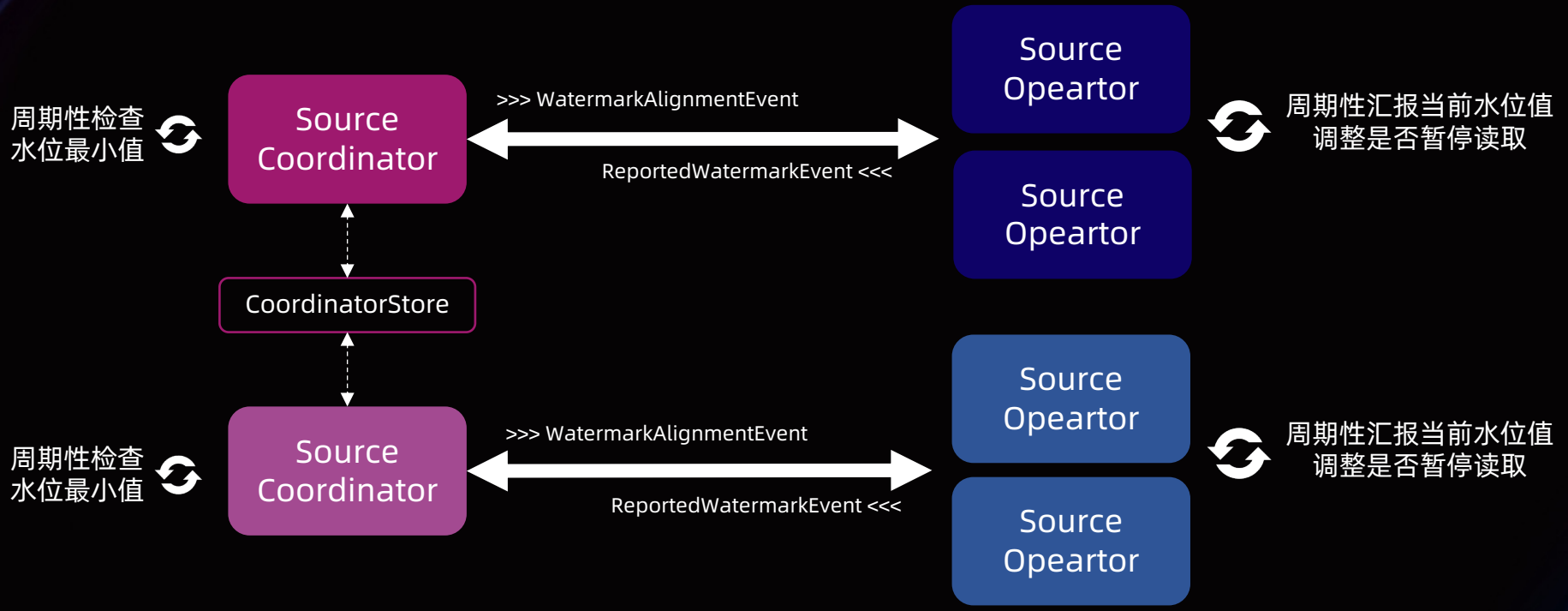
Source 不同并发 / 多个 Source 的读取速度不一致会对下游算子状态产生影响

水印对齐

```
WatermarkStrategy
    .withWatermarkAlignment(
        String alignmentGroup,                  水印对齐组
        Duration maxAllowedWatermarkDrift,      最大允许的水印偏移
        Duration updateInterval);               水印检查间隔
```

适用于同一 Source 的不同并发，也可以在多个 Source 之间通过配置 水印对齐组 进行对齐

水印对齐



- 由 Source Coordinator 协调允许的最大水位
- 过快的 Source Operator 将暂停读取
- 不同 Source 之间通过共享的 CoordinatorStore 协调最大允许水位

水印对齐

- 1.16 支持不同 Source / 同一 Source 的不同并发之间的水印对齐
- 建议在多个 Source 之间或在并发数与 Split 数一致的 Source 上使用
  - 多个 Source： Kafka + FileSystem Source
  - 相同 Source：Kafka Source 并发与 Partition 数一致
- 1.17 将完成 Split 级别的水印对齐（FLIP-217）

维表与缓存

FLIP-221 Abstraction for lookup source cache and metric

新的维表接口

新的 LookupFunction 在 TableFunction 的基础之上更清晰

通用维表缓存

为所有维表提供通用的缓存功能，无需重复实现

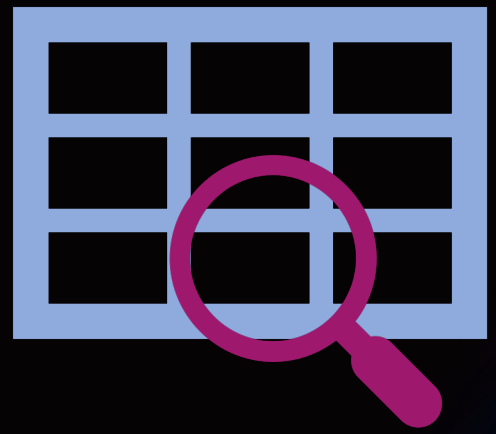
新的维表接口

- LookupFunction / AsyncLookupFunction

定义如何对给定的 key 从外部系统中查询数据

- LookupFunctionProvider / AsyncLookupFunctionProvider

在 Table / SQL API 中构建 LookupFunction



通用维表缓存

部分缓存模式 Partial Caching

- 按需 在外部系统中查询数据并缓存
- 维表自行管理缓存和清理策略
- 适用于规模较大的维表

默认缓存实现 DefaultLookupCache

- 基于 Guava Cache 实现
- 提供基于缓存大小、写入时间、读取时间的清理机制

PartialCachingLookupProvider.of(LookupFunction fn, LookupCache cache)

通用维表缓存

全部缓存模式 Full Caching

- 从外部系统中 拉取全部数据，并使用指定触发器进行重载
- 复用 Scan 能力
- 适用于规模较小的维表

重载触发器 CacheReloadTrigger

- 周期性
- 指定时间

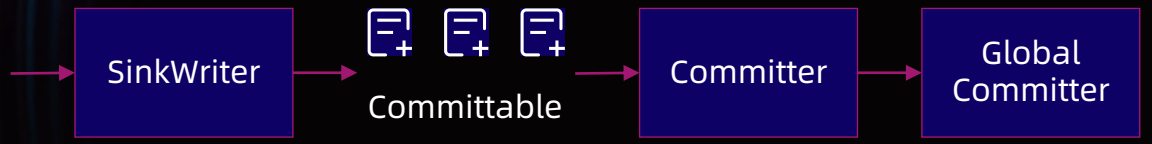
```
FullCachingLookupProvider.of(
        ScanTableSource.ScanRuntimeProvider scanRuntimeProvider,
        CacheReloadTrigger cacheReloadTrigger
```

即将实现的其他新功能

- FLIP-217: Support watermark alignment of source splits
  - Split 级别的水印对齐
- FLIP-208: Add RecordEvaluator to dynamically stop source based on de-serialized records
  - 根据数据内容确定有界流的结束位置
- FLIP-238: Introduce FLIP-27-based Data Generator Source
  - 基于 FLIP-27 Source API 的数据生成器和通用的 Source 限流功能

03 Sink V2 API 与小文件合并

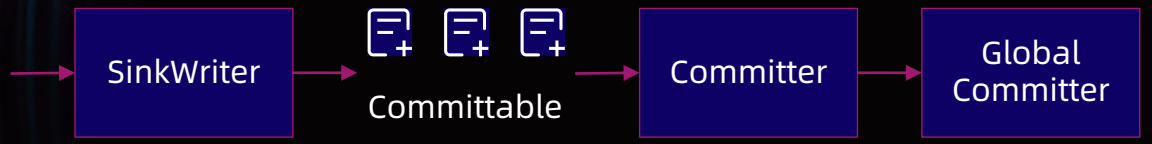
SinkV2介绍



SinkV1与两阶段提交

- SinkWriter: 写出数据并生成Committable
- Commitable: 需要Checkpoint时提交的信息
- Committer: Checkpoint时提交Committable，Committable提交后对下游可见
- Global Committer: 部分场景下需要的全图提交

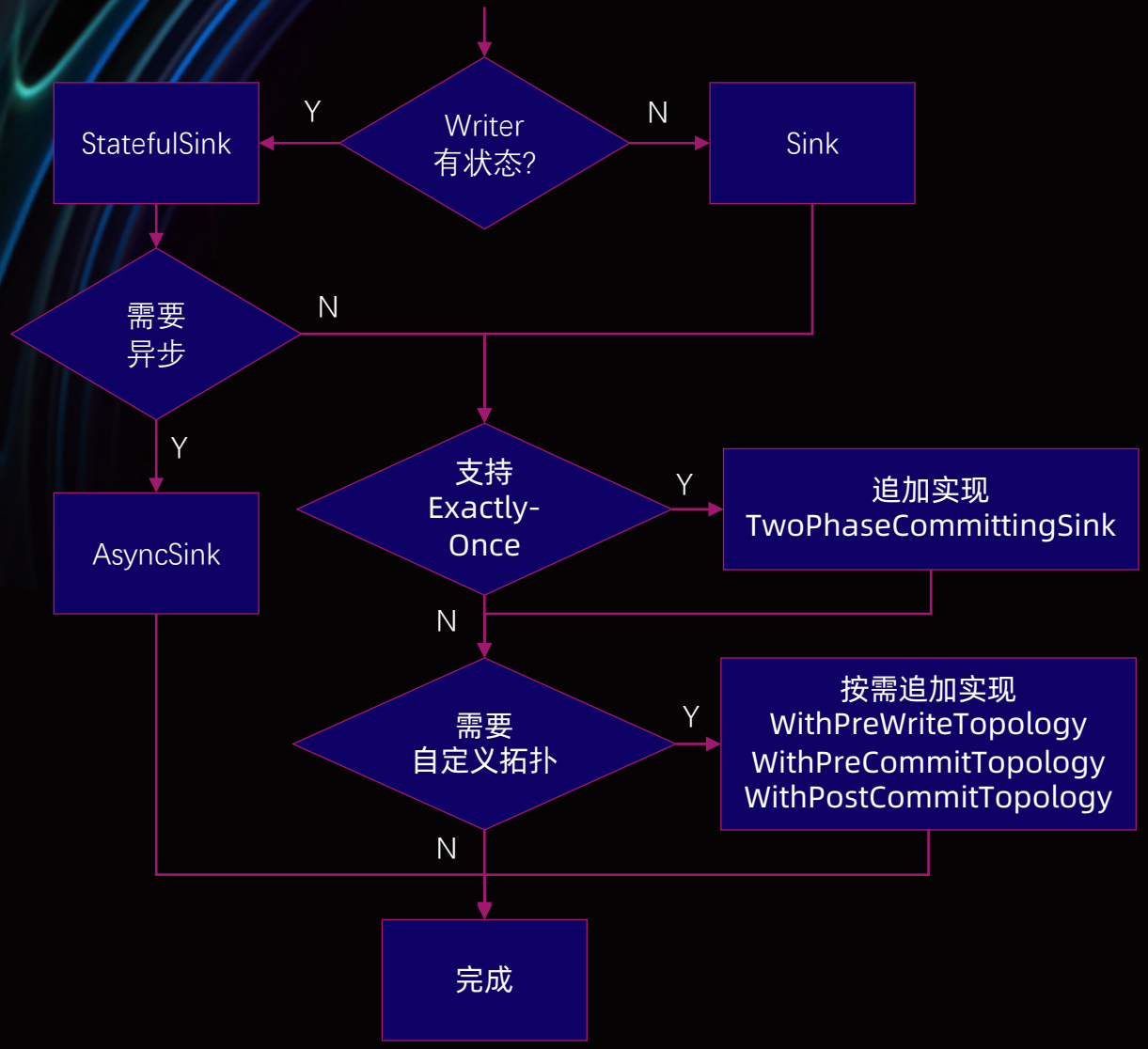
SinkV2介绍



SinkV1的问题

- 无法支持小文件合并
- 灵活性不足

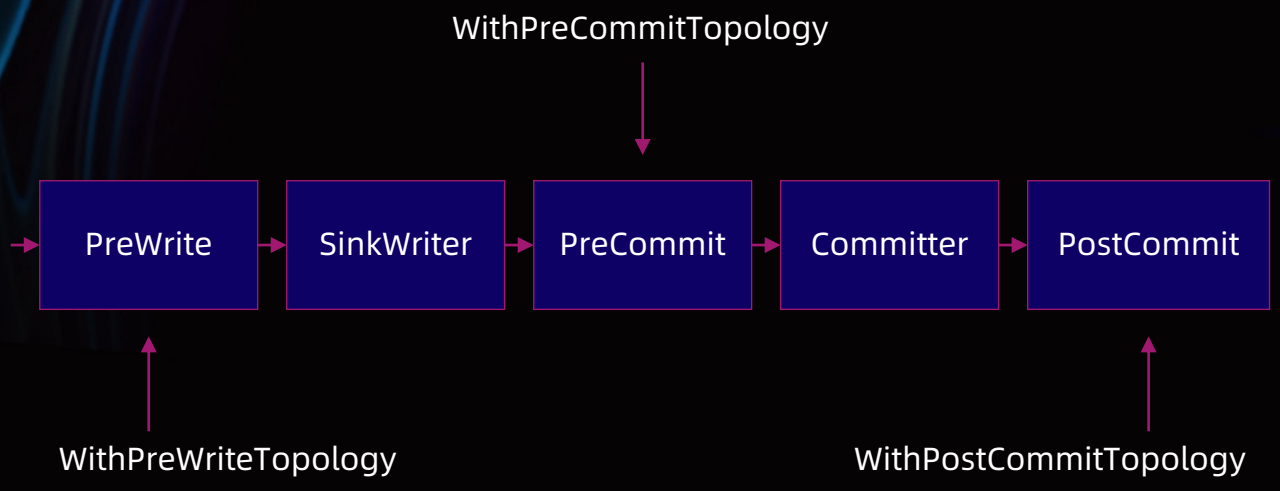
SinkV2介绍



SinkV2接口

- 基础接口
  - Sink
  - StatefulSink
  - TwoPhaseCommittingSink
- 拓扑扩展接口
  - WithPreWriteTopology
  - WithPreCommitTopology
  - WithPostCommitTopology
- 功能抽象类
  - AsyncSink

SinkV2介绍



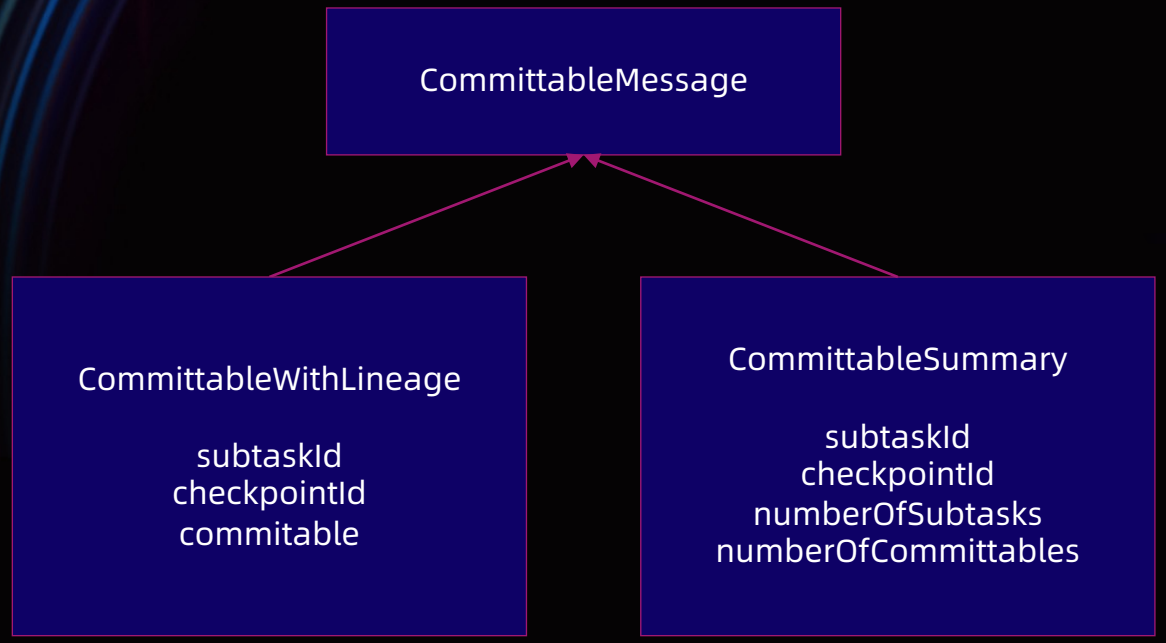
SinkV2拓扑

Sink，StatefulSink，AsyncSink仅有SinkWriter

TwoPhaseCommittingSink与SinkV1基本一致，但无GlobalCommitter

实现拓扑扩展接口将追加对应拓扑

SinkV2介绍



Committable消息封装

Writer每个Checkpoint输出一个CommittableSummary和若干个CommittableWithLineage

- *注意：PreCommitTopology最终输出应与此规则一致*
- CommittableWithLineage: 承载Committable对象及其所属并发，CP信息
- CommittableSummary: 单个并发单次CP发送的所有Committable的统计信息

SinkV2介绍

```
protected AsyncSinkBase(
        ElementConverter<InputT, RequestEntryT> elementConverter,
        int maxBatchSize,
        int maxInFlightRequests,
        int maxBufferedRequests,
        long maxBatchSizeInBytes,
        long maxTimeInBufferMS,
        long maxRecordSizeInBytes) {
```

```
public abstract class AsyncSinkWriter<InputT, RequestEntryT extends Serializable>
        implements StatefulSink.StatefulSinkWriter<InputT, BufferedRequestState<RequestEntryT>> {
protected abstract void submitRequestEntries(
        List<RequestEntryT> requestEntries, Consumer<List<RequestEntryT>> requestToRetry);
protected abstract long getSizeInBytes(RequestEntryT requestEntry);
```

AsyncSink

使用AsyncSinkBase作为Sink基类，并提供异步参数

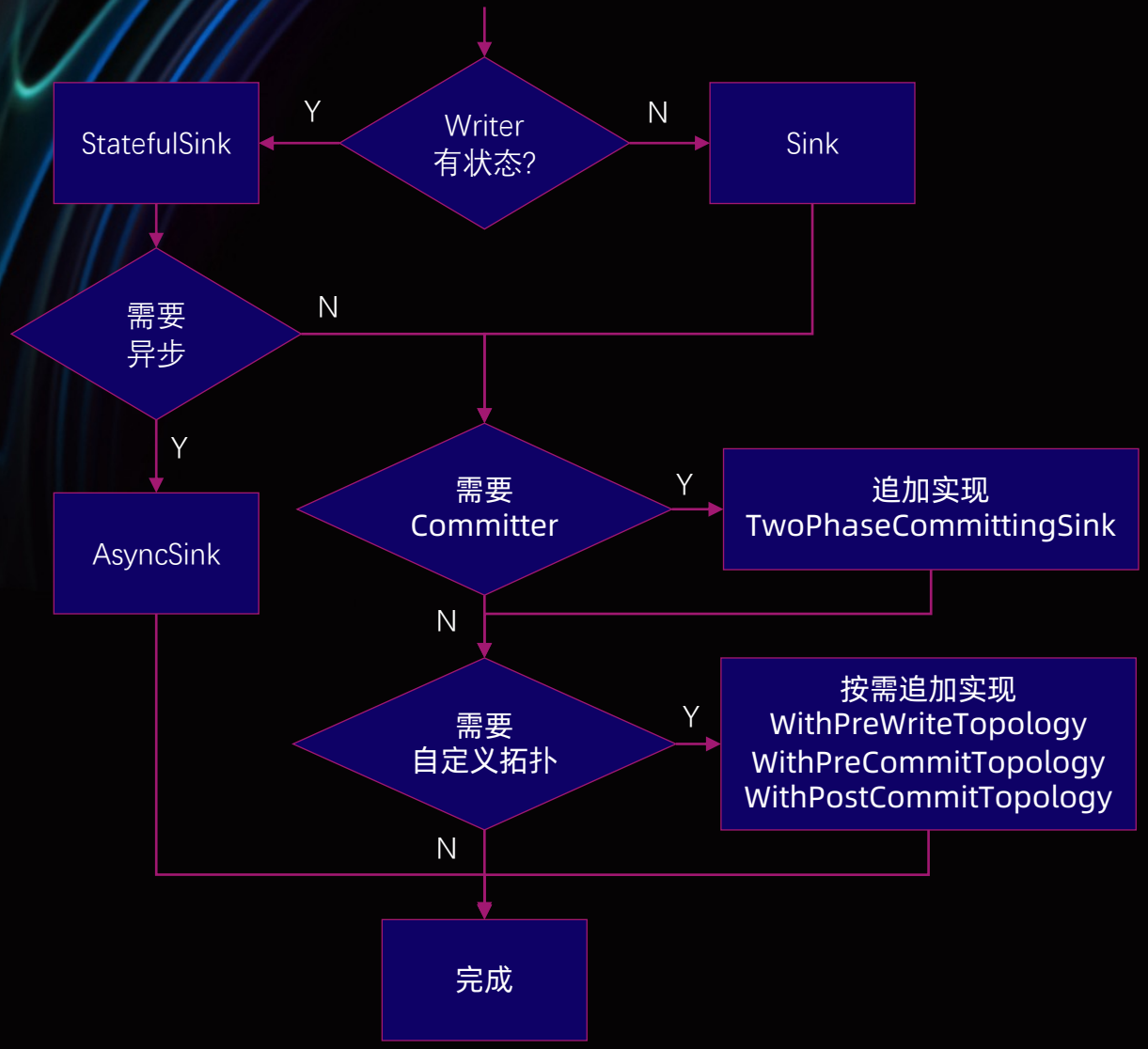
- ElementConverter: 将输入数据转化为写请求

使用AsyncSinkWriter作为Writer基类

- submitRequestEntries: 异步提交写请求，该方法必须为非阻塞的异步实现
- getSizeInBytes: 获取写请求的大小，用于触发flush

AsyncSink仅支持At-least-once语义

SinkV2介绍



SinkV2开发与使用

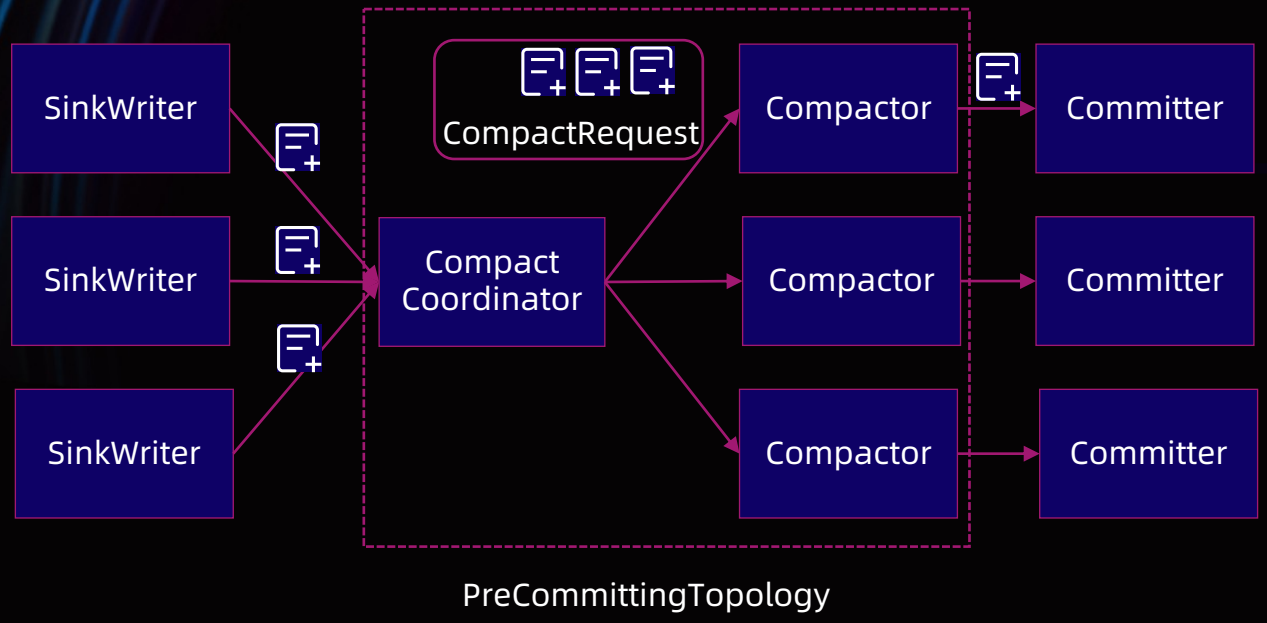
新开发时选择适当接口和需要的拓扑扩展接口

已有SinkV1迁移选择适当接口，其中方法定义基本相同

- GlobalCommitter无对应接口，需要单独使用WithPostCommitTopology实现
- 可参考SinkV1Adapter

使用与SinkV1一致，对用户无感知

FileSink解决小文件问题



FileSink拓扑与小文件合并方案

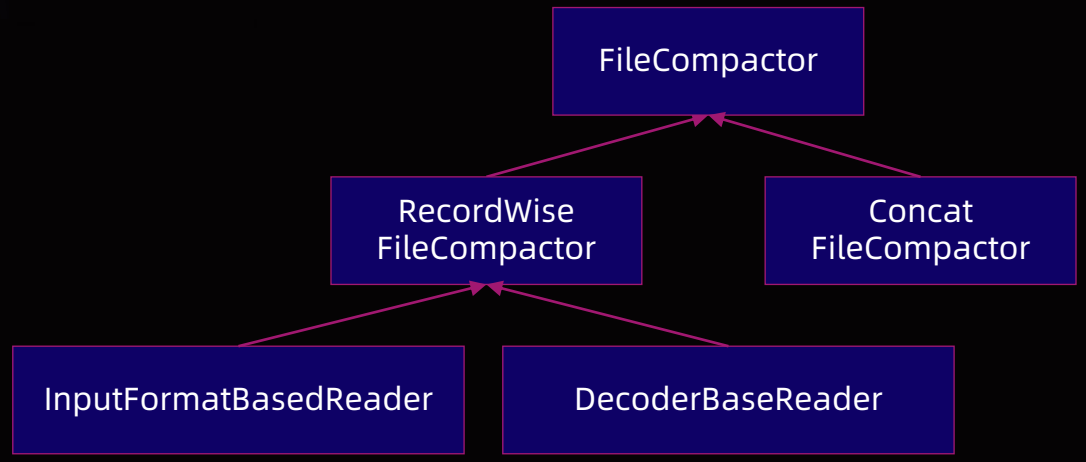
利用PreCommittingTopology扩展

- CompactCoordinator: 接收Writer输出的Committable，跨并发/CP进行合并输出CompactRequest，单点执行
- Compactor：执行合并并输出Committable，对Committer而言与Writer等价

FileSink解决小文件问题

```
FileSink.forRowFormat(new Path(path), new IntEncoder())
  .enableCompact(createFileCompactStrategy(), createFileCompactor())
  .build();
```

```
FileCompactStrategy.Builder.newBuilder()
  .setNumCompactThreads(4)
  .enableCompactionOnCheckpoint( numCheckpointsBeforeCompaction: 3)
  .setSizeThreshold(10000)
  .build();
```



FileSink启用小文件合并

通过FileSink#enableCompact启用

- *注意：启用后关闭需显式调用disableCompact，不可直接删除enableCompact*

- FileCompactStrategy: 指定Committable如何组合成CompactRequest

- FileCompactor: 如何合并文件
  - ConcatFileCompactor: 直接拼接文件
  - RecordWiseFileCompactor: 读出数据后写新文件

后续规划

- SinkV1已deprecate，推荐迁移至SinkV2
- SinkV2的实现优化
  - 支持并发度设置
  - 优化两阶段提交的消息机制
  - 提供规范输出两阶段提交消息的工具类
- 基于SinkV2的功能
  - 通用的Committable合并

FLINK FORWARD # ASIA 2022 ONLINE | 实时即未来

THANK YOU

谢 谢 观 看