Apache Flink

Apache Flink 1.9.0 特性解读

Flink China 北京站 Meetup - 2019年06月29日

Apache Flink

About us

- **杨克特（鲁尼）**
  - Apache Flink Committer
  - 目前在阿里巴巴负责 Blink SQL 引擎
- **戴资力（Gordon）**
  - Apache Flink PMC
  - Software Engineer @ Ververica

Apache Flink 开源社区共同努力的成果

所有新版本释出的功能与改进，

绝非单一团队 / 个人的努力而来，

而是全体社区的数位成员无私贡献的成果

Apache Flink

CONTENT

目录 >>

01 / Apache Flink 1.9.0 特性解读

- 架构改动
- Table API & SQL
- Runtime & Core
- 生态

02 / 未来版本计划

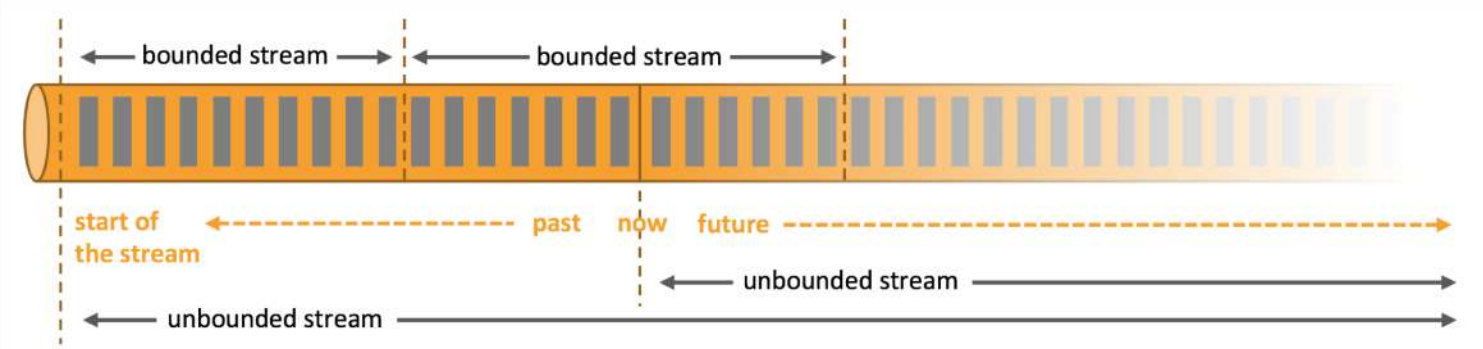
Apache Flink

01

架构改动

Architecture Changes

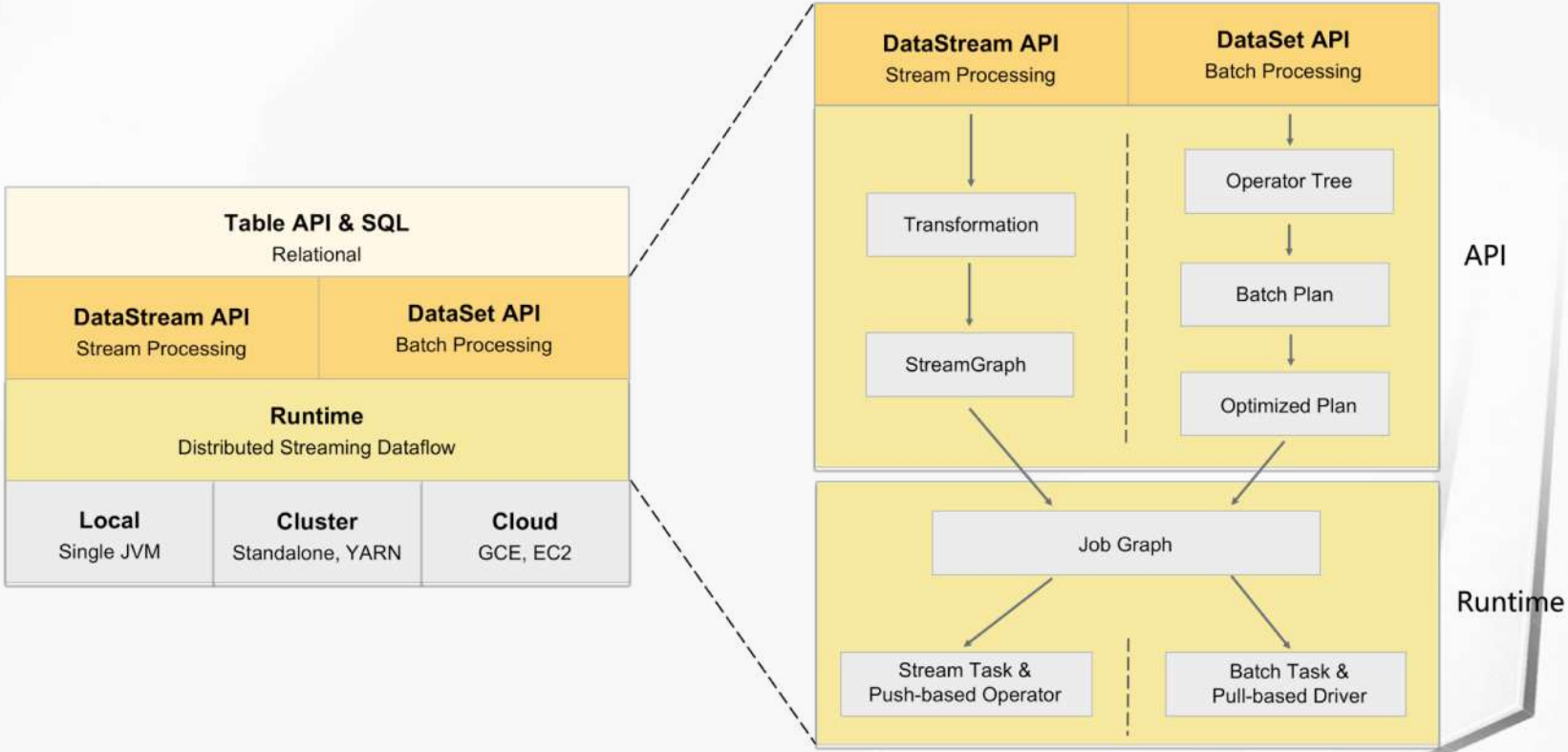
Flink 的设计理念



Continuous processing of unbounded data streams as ***core abstraction***,
Batch as a ***special case*** of streaming

Apache Flink

Flink 架构

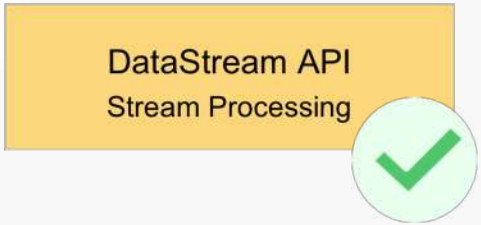


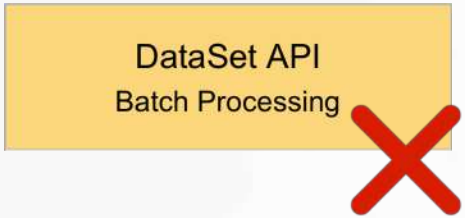
存在的问题

- 从用户角度：
  - 需要在两个底层API中进行选择
  - 不同的语义、不同的connector支持、不同的错误恢复策略...
  - Table API 也会受不同的底层API，不同的connector等问题的影响
- 从开发者角度：
  - 不同的翻译流程，不同的算子实现，不同的Task执行...
  - 代码难以复用
  - 两条独立的技术栈 -> 需要更多的人力 -> 功能开发变慢、性能提升变难，bug变多

一个大胆的想法

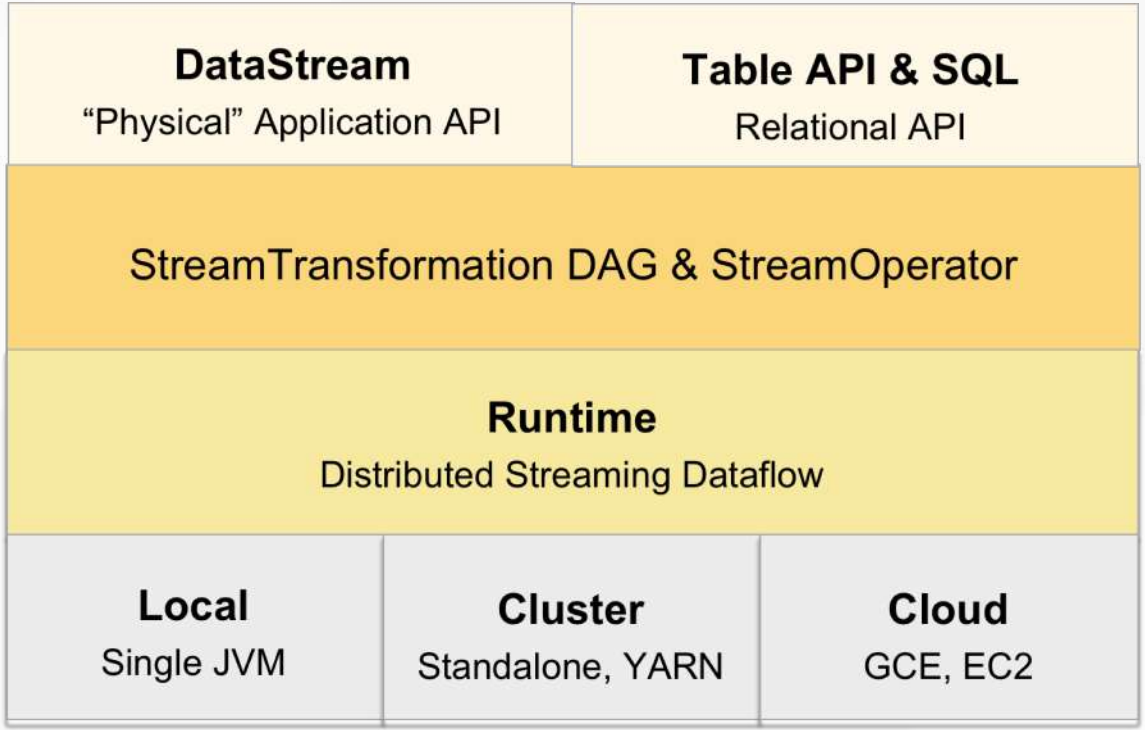
既然批是流的一个特例，我们是否可以...?





搞定！

未来架构



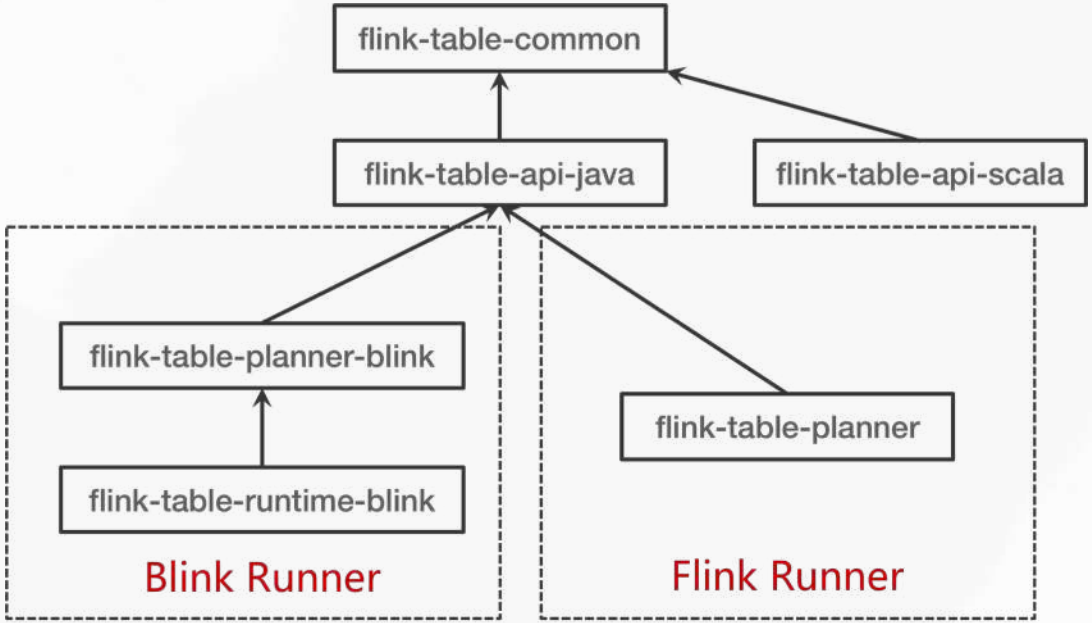
端到端重构修改：

- Table API & SQL
- DataStream增加批处理
- 统一的DAG API
- 统一的算子 API
- 统一的Source API
- 网络传输架构
- 作业调度
- 错误处理

绿色：已启动
黄色：计划中

Table API & SQL 如何吃好这个螃蟹？

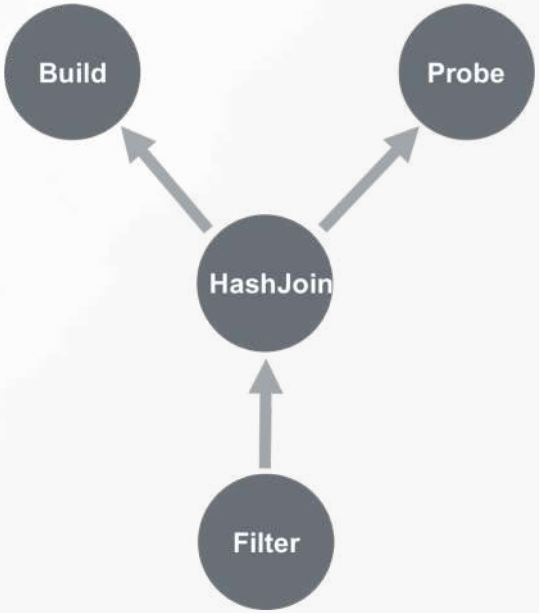
[FLIP-32] Restructure flink-table for future contributions
[FLINK-11439] INSERT INTO flink_sql SELECT * from blink_sql



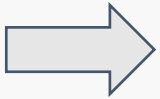
- Table 模块进行拆分
- Table 支持多个 Runner，用户可自行选择使用哪个 Runner
- Flink runner 保持原来的行为，继续翻译到 DataStream/DataSet
- Blink runner 对接最新的 runtime 架构，流批作业使用统一的 DAG 和 StreamOperator 来描述
- 未来将只保留 Blink Runner

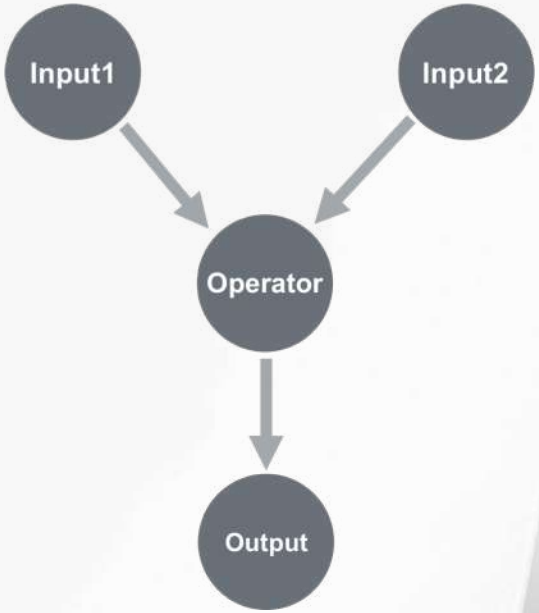
Apache Flink

Selective Push Model Operator



Pull-based operator (Driver)

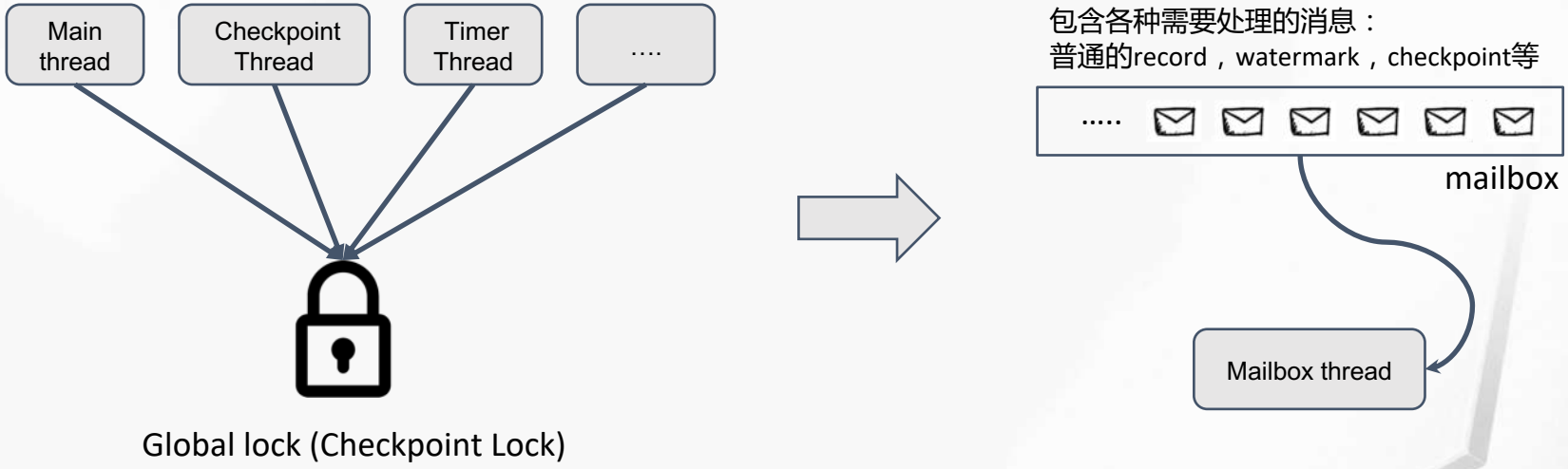




Push-based operator
算子可自定义读取顺序

Stream Task Mailbox Model

[FLINK-12477] Change threading-model in StreamTask to a mailbox-based approach



Apache Flink

02

Table API & SQL

Apache Flink

Table API & SQL 新特性预览

全新的 SQL
类型系统

DDL
初步支持

Table API
增强

统一的
Catalog API

更好的
Hive 兼容性

Blink Runner

全新的 SQL 类型系统

[FLIP-37] Rework of the Table API Type System

- 目前使用的 TypeInformation 的问题：
  - 和 SQL 的语义不一致（比如Decimal没有精度信息）
  - 和物理存储格式绑定（TypeInformation绑定了TypeSerializer）
- 全新的 SQL 类型系统
  - 保持和 SQL 标准语义一致
  - 只包含纯粹的逻辑类型信息，可支持多种物理存储形式（比如 TIMESTAMP 可以用Long，java.sql.Timestamp，java.time.LocalDateTime 等形式来表示）
- 所有 SQL 类型：（绿色: 1.9完整支持，黄色: 1.9部分支持，红色: 1.9+）

  CHAR(n), VARCHAR(n), BOOLEAN, BINARY(n), VARBINARY(n),
  DECIMAL(p, s), NUMERIC, TINYINT, SMALLINT, INT, INTEGER, BIGINT, FLOAT, DOUBLE,
  DATE, TIME, TIMESTAMP, TIMESTAMP WITH TIMEZONE, TIMESTAMP WITH LOCAL TIMEZONE,
  INTERVAL YEAR, INTERVAL MONTH, INTERVAL DAY, INTERVAL HOUR, INTERVAL MINUTE, INTERVAL SECOND
  ARRAY, MULTISET, MAP, ROW, NULL, ANY, USER DEFINED TYPE

SQL DDL

```
CREATE TABLE kafka_orders (
  order_id VARCHAR,
  product VARCHAR,
  amount BIGINT,
  order_ts TIMESTAMP,
  PRIMARY KEY (order_id)
  proctime AS PROCTIME(),
  WATERMARK FOR order_ts AS BOUNDED WITH DELAY '10' SECOND
) WITH (
  connector='kafka',
  kafka.bootstrap.servers='localhost:9092',
  kafka.topic='orders',
  kafka.group.id='testGroup',
  kafka.startup-offset='earliest',
  kafka.end-offset='none',
  ...
);
```

Schema

计算列

Watermark (1.9?)

定义了表的属性，包括存储类型，连接信息，读取范围，有界性等

其他常用的DDL语句：

CREATE FUNCTION / CREATE VIEW

INSERT INTO / INSERT INTO PARTITION (hive dialect) / INSERT OVERWRITE (hive dialect)

从 SQL 文本自动区分流批计算（1.9+）

```
CREATE TABLE kafka_orders (
  order_id VARCHAR,
  product VARCHAR,
  amount BIGINT,
  order_ts TIMESTAMP,
  PRIMARY KEY (order_id)
  proctime AS PROCTIME(),
  WATERMARK FOR order_ts AS BOUNDED WITH DELAY '10' SECOND
) WITH (
  connector='kafka',
  kafka.bootstrap.servers='localhost:9092',
  kafka.topic='orders',
  kafka.startup-offset='earliest',
  kafka.end-offset='none' 或 kafka.end-offset='2019-06-28 00:00:00'
  ...
);
```

流处理

批处理

```
SELECT product, TUMBLE_START(order_ts, INTERVAL '1' MINUTE), COUNT(*)
FROM kafka_orders GROUP BY product, TUMBLE(order_ts, INTERVAL '1' MINUTE);
```

更多讨论见：Ground Source Sink Concepts in Flink SQL

Table API 增强

- 更丰富的列操作API
  - Table addColumns(Expression... fields);
  - Table addOrReplaceColumns(Expression... fields);
  - Table renameColumns(Expression... fields);
  - Table dropColumns(Expression... fields);

- 更灵活的操作方法
  - Table map(Expression mapFunction);
  - Table flatMap(Expression tableFunction);
  - AggregatedTable aggregate(Expression aggregateFunction);
  - FlatAggregateTable flatAggregate(Expression tableAggregateFunction);

Apache Flink

统一的 Catalog API

[FLIP-30] Unified Catalog APIs

统一的 Catalog API 的意义：

- 完整支持 DDL 的基础
- 可以使用不同的介质来存储meta信息（纯内存，文件，其他 meta 系统等）
- 便于和现有系统的集成（hive meta store，mysql）
- 使 Flink 不仅具备异构数据源的联合计算能力，甚至提供跨数仓的联合计算

1.9 提供两种 Catalog 实现：

- InMemoryCatalog
- HiveCatalog（可以用在两种场景，一是将 Flink 的表存储到 HMS 中，二是以兼容的方式读写 Hive 元数据）

Apache Flink

Hive 兼容性

总体计划：1.9 支持和 Hive 相关的 DML（SELECT、INSERT），包括：

- 读取 Hive 数据表
  - 支持分区表和非分区表
  - 支持多种文件格式，text、orc、parquet 等
- 以兼容的方式写 Hive 数据表
  - 支持分区表与非分区表，对于分区表支持静态分区与动态分区
- 支持读写 Hive 的复杂数据结构（除 UNION外 ）
- 支持在 Flink 中直接运行 Hive UDF，包括 GenericUDF，UDTF，UDAF

1.9+ 版本继续支持 Hive 的 DDL

更详细的信息不要错过：《Apache Flink-1.9与Hive的兼容性》

Apache Flink

Blink Runner

数据结构
二进制化

更丰富的
内置函数

Minibatch
Aggregation

多种解
热点手段

维表支持

TopN

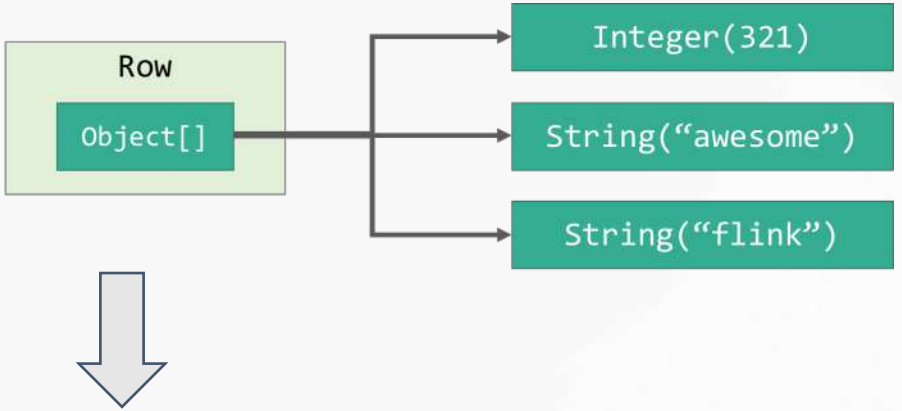
高效的
流式去重

完整的
批处理支持

数据结构二进制化

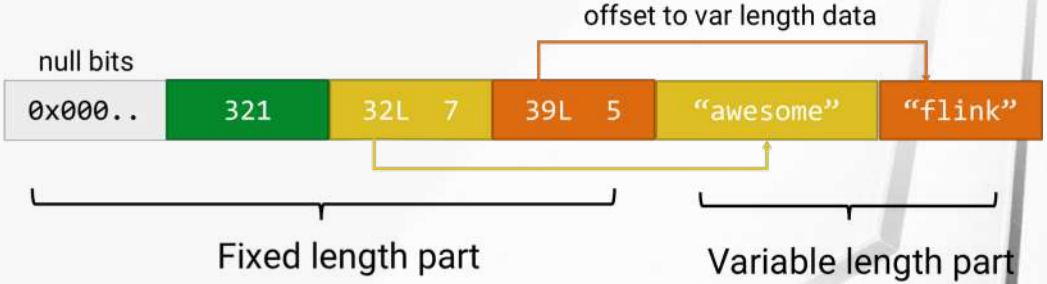
旧数据结构：Row

- Java 对象的空间开销高
- 主类型的装箱和拆箱开销
- 昂贵的 hashCode() 和 (反)序列化



新数据结构：BinaryRow

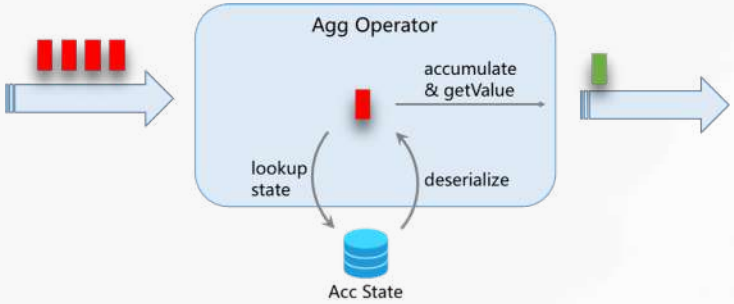
- 完全基于二进制数据
- 与内存管理紧密结合，CPU 缓存友好
- 避免了大量反序列化开销
- 大幅提升流批作业性能



Minibatch Aggregation

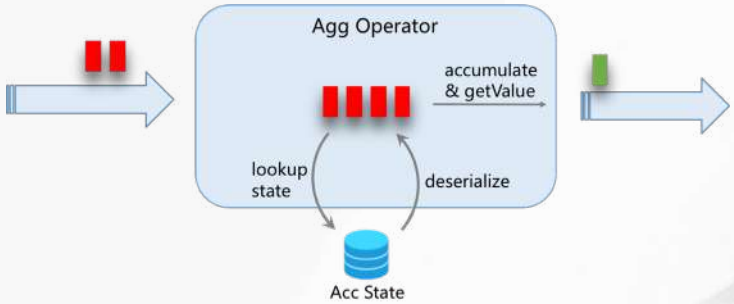
普通版本：

- 每一条消息都会读写 state
- 大量序列化/反序列化开销



Minibatch 版本：

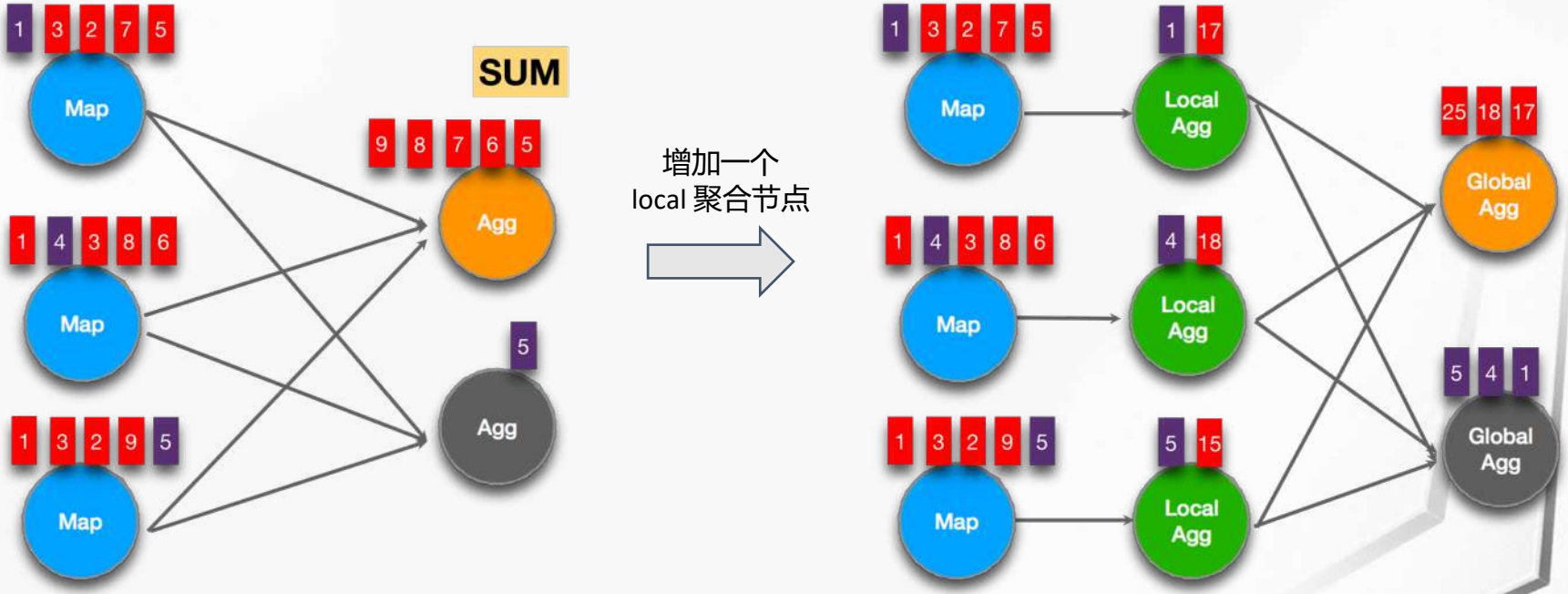
- 使用内存进行攒批
- 内存中即可聚合，减少 state 读写
- 输输出数据少，下游压力降低
- 大幅提升吞吐



Apache Flink

Local Global Agg 解决简单热点

```
SELECT COUNT(*) FROM T GROUP BY color
```



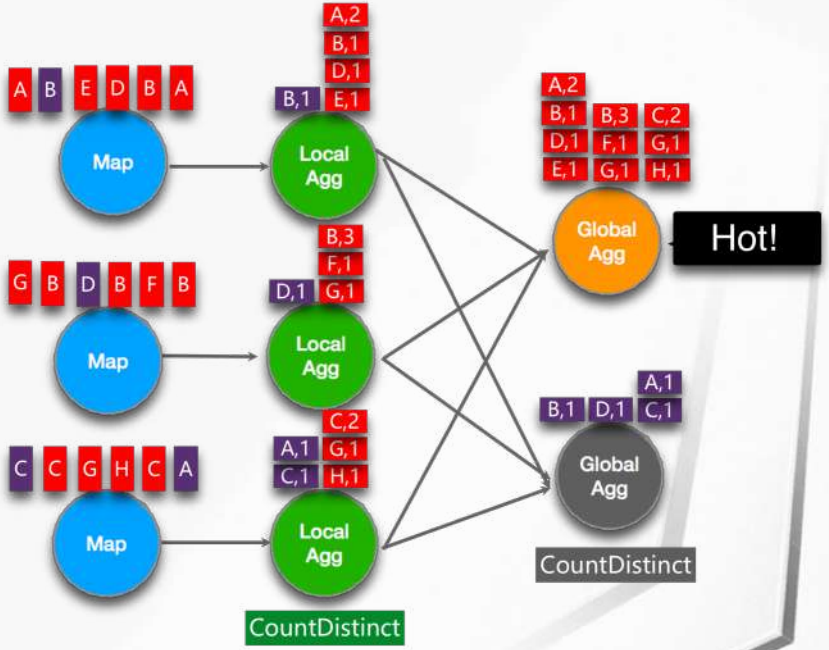
两层 Agg 改写解决复杂热点

```
SELECT color, COUNT(DISTINCT id) FROM T GROUP BY color
```

对于中间结果需要存储所有明细的agg（例如 count distinct），
简单的 local - global 已经不能很好的解决热点问题

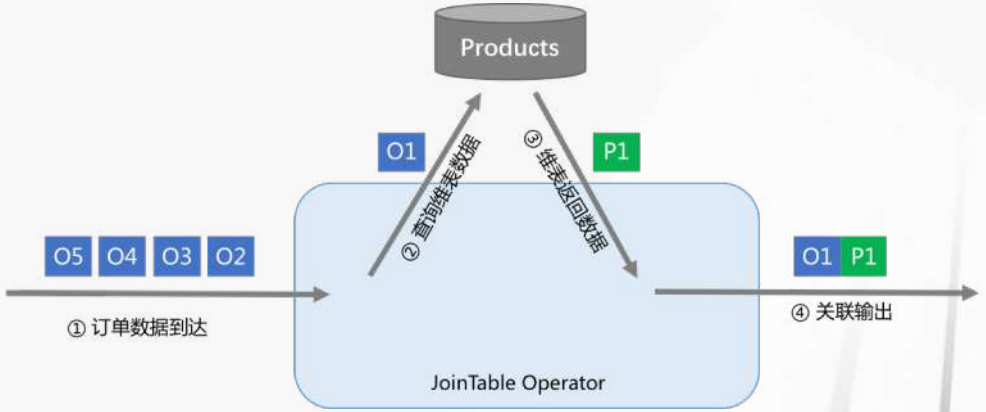
优化过程中进行 query 改写的操作

```
SELECT color, SUM(cnt)
FROM (
    SELECT color, COUNT(DISTINCT id) as cnt
    FROM T
    GROUP BY color, MOD(HASH_CODE(id), 1024)
)
GROUP BY color
```



维表 join

```
CREATE TABLE mysql_products (
  product_id VARCHAR,
  product_name VARCHAR,
  price DECIMAL(10, 2),
  PRIMARY KEY (product_id)
) WITH (
  connector = 'mysql'
  …
);
```




```
SELECT o.*, p.*
FROM kafka_orders AS o
JOIN mysql_products FOR SYSTEM_TIME AS OF o.proctime AS p
ON o.product_id = p.product_id
```

TopN 计算

例：计算每个类目中总销量排名前3的店铺

```
SELECT *
FROM (
  SELECT
    *,
    ROW_NUMBER() OVER (
       PARTITION BY category
       ORDER BY sales DESC) AS rownum
  FROM shop_sales)
WHERE rownum <= 3
```

- 在流计算中，会识别这样的 query pattern 并优化成一个单独的算子
- 针对不同的细分场景，有多种不同的实现，优化器自动进行选择（比如只输出商家，不需要具体排名）

result			
category	shopId	sales	rownum
book	shop-43	89	1
book	shop-46	56	2
book	shop-58	43	3
fruit	shop-12	78	1
fruit	shop-44	67	2
fruit	shop-32	57	3
...	...	...	...

高效流式去重

数据中有PK，但是上游可能会重复发送，两种场景：

1. 上游由于重启或者其他原因发送完全重复数据，只有第一条对你有意义
2. 上游会持续不断的针对主键更新数据（比如上游是UpsertSink）

每个主键保留第一条数据：

```
SELECT parimary_key, a, b, c
FROM (
  SELECT
    *,
    ROW_NUMBER() OVER (
       PARTITION BY parimary_key
       ORDER BY proctime ASC) AS rownum
  FROM T)
WHERE rownum == 1
```

每个主键保留最后一条数据：

```
SELECT parimary_key, a, b, c
FROM (
  SELECT
    *,
    ROW_NUMBER() OVER (
       PARTITION BY parimary_key
       ORDER BY proctime DESC) AS rownum
  FROM T)
WHERE rownum == 1
```

借鉴 TopN 的思想，识别 query pattern 后使用高效算子来实现

Apache Flink

完整的批处理支持

- 功能
  - 基本功能完备
  - 支持多种 join 方式（inner / left / right / full outer / semi / anti）
  - 完整支持几乎所有 over window
  - 支持子查询（correlated / uncorrelated）
  - 高级分析函数支持（grouping set, cube, rollup…）
- 稳定性
  - Runtime 进行了大量的稳定性优化（后面介绍）
  - Shuffle service 插件化

Flink 从 1.9 版本开始将成为传统批处理引擎的有力竞争者！

03

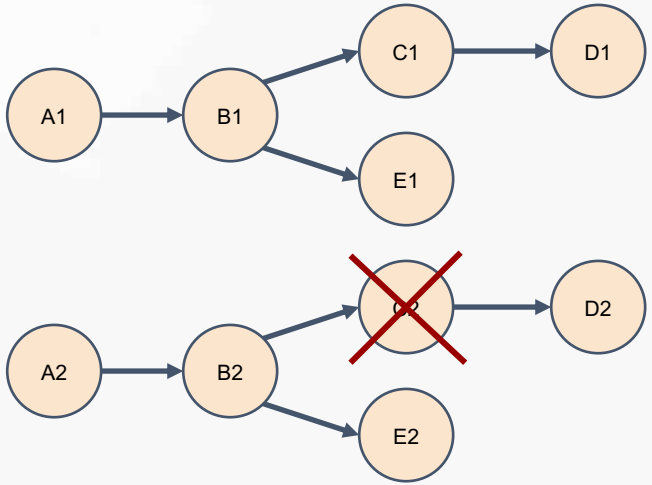
Runtime

Fine-grained recovery

[FLIP-1] Fine-grained recovery

- 降低错误恢復所需花费时间 / IO 资源：
  - 错误发生时，只需局部重启与错误的 Task 有依赖的部分

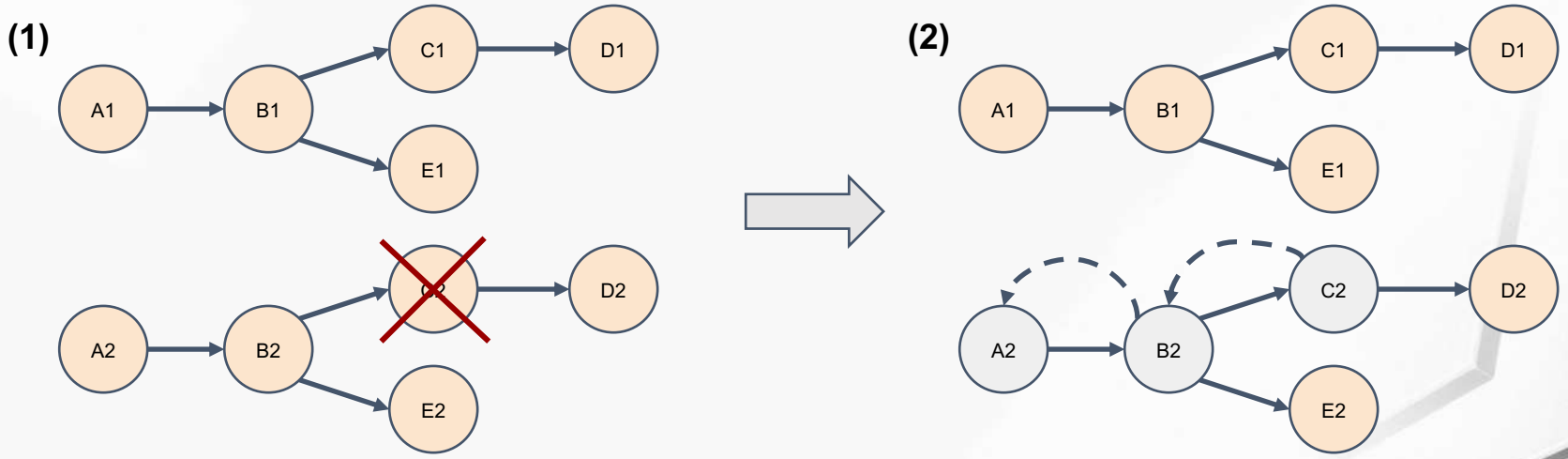
(1)



Fine-grained recovery

[FLIP-1] Fine-grained recovery

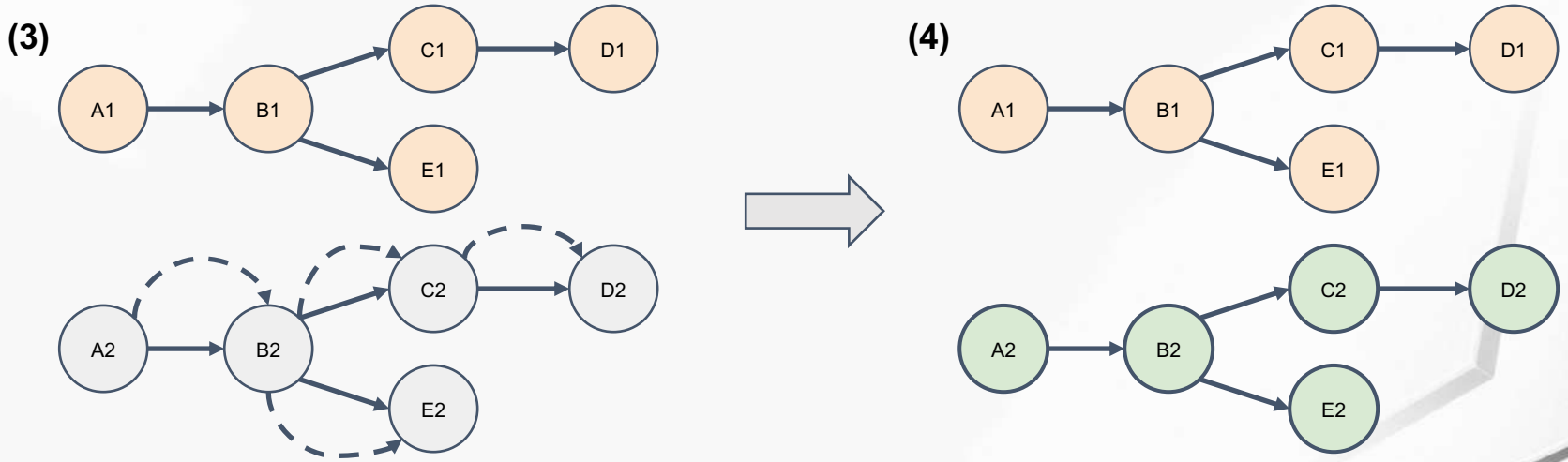
- 降低错误恢復所需花费时间 / IO 资源：
  - 错误发生时，只需局部重启与错误的 Task 有依赖的部分



Fine-grained recovery

[FLIP-1] Fine-grained recovery

- 降低错误恢復所需花费时间 / IO 资源：
  - 错误发生时，只需局部重启与错误的 Task 有依赖的部分



Apache Flink

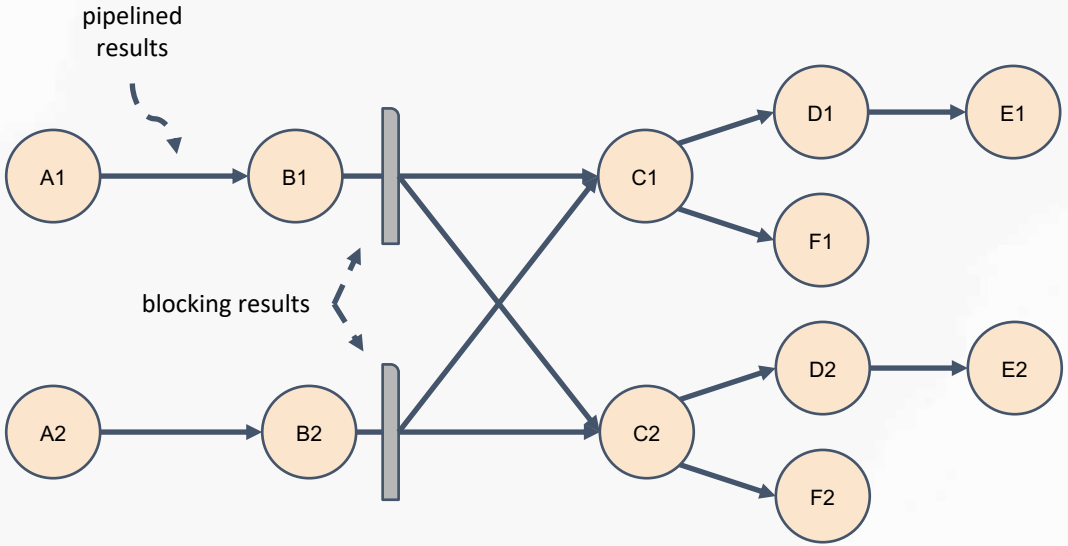
Fine-grained recovery

- 现有支援的限制：
  - **对于流式应用 -**
    - 只对于 Embarrasingly Parallel (无 shuffling) 之应用有优化效果
    - 大部分流式应用算子间皆是 all-to-all 依赖关係，且皆为 pipeline 计算

Fine-grained recovery

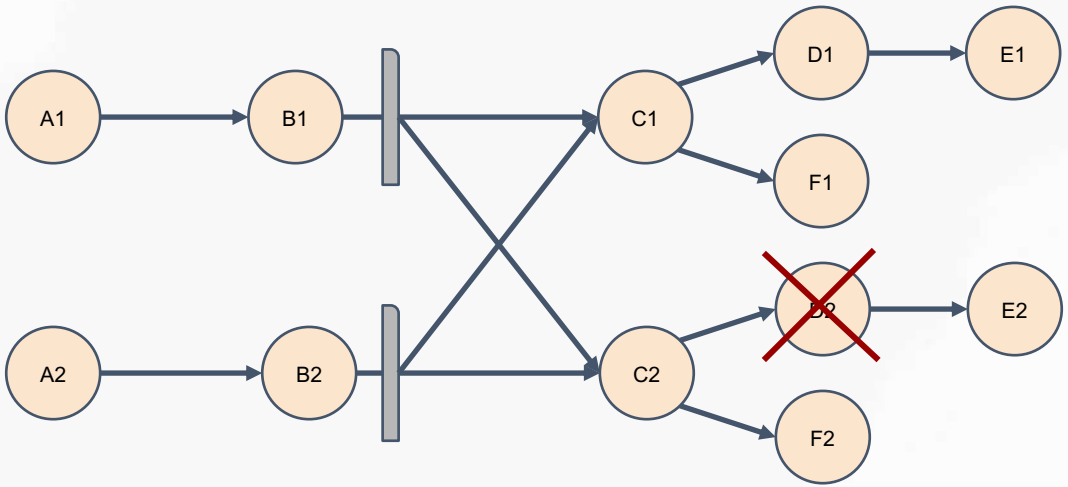
- 现有支援的限制：
  - **对于流式应用 -**
    - 只对于 Embarrasingly Parallel (无 shuffling) 之应用有优化效果
    - 大部分流式应用算子间皆是 all-to-all 依赖关係，且皆为 pipeline 计算
  - **对于批次应用 -**
    - 虽然算子也大多是 all-to-all 依赖关係，
      但部分的中介结果为 blocking intermediate results，而非 pipelined
    - **可以利用此特性来缩减错误发生时执行 DAG 所需重启的范围**

Fine-grained recovery for batch



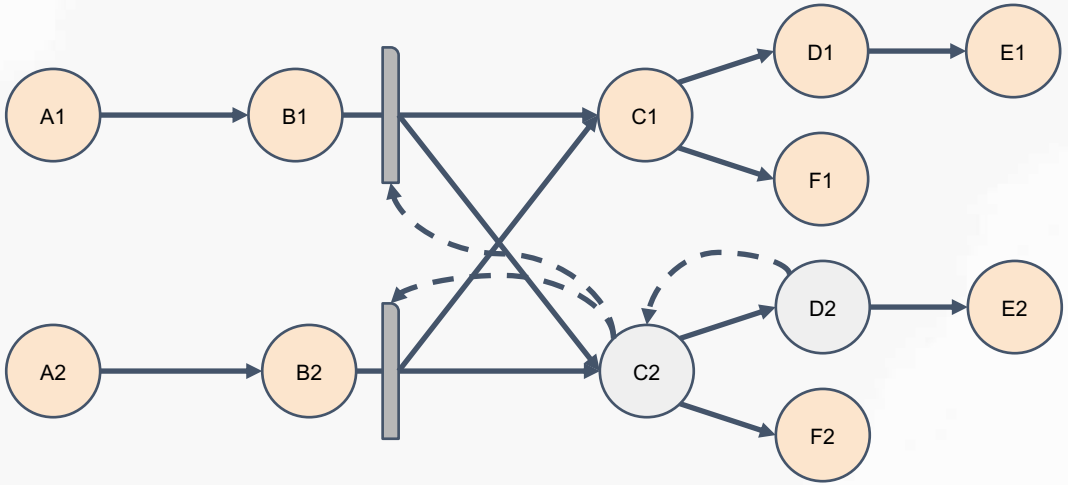
Fine-grained recovery for batch

(1)



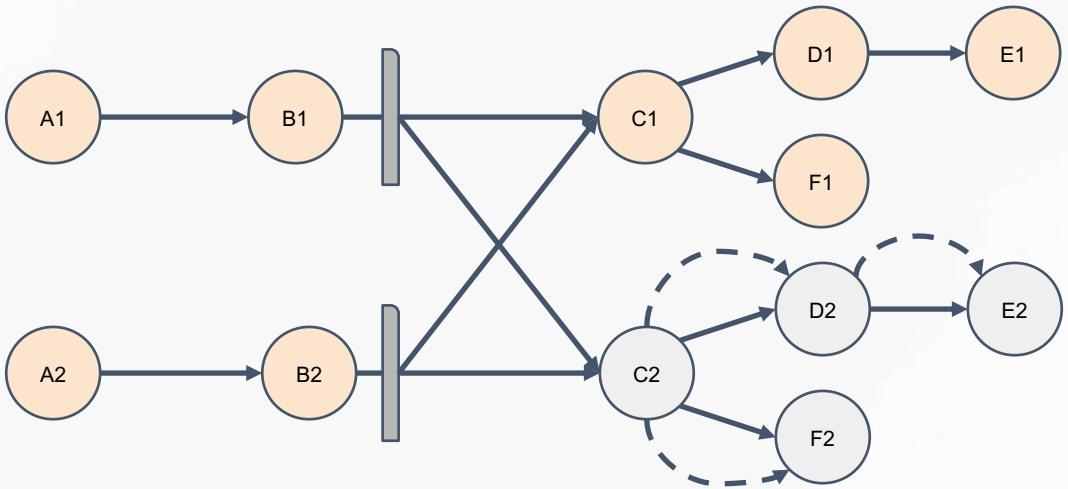
Fine-grained recovery for batch

(2)



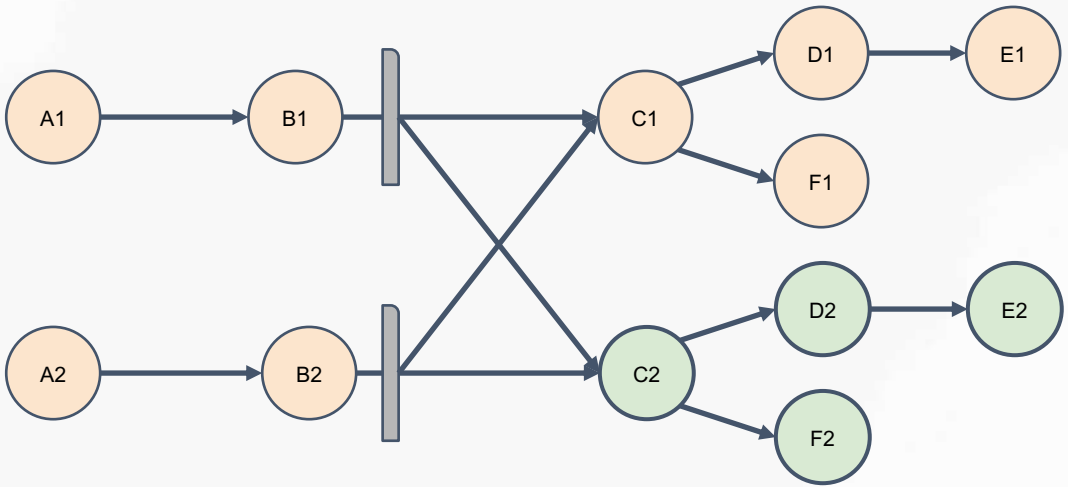
Fine-grained recovery for batch

(3)



Fine-grained recovery for batch

(4)



Fine-grained recovery for batch

- 对于 runtime 的需求：
  - **Intermediate result partitions（简称 RP）需支援重複消费**
    - 若无法重複消费 RP，没办法更进一步缩减 DAG 所需重启的范围
    - <= 1.8：Task 结束运算后，产生的 RP 被下游算子消费一次后即被释出
  - **需集中化维护跨所有 TM 中所有 RP 的全域观**
    - 拥有所有 RP 全域观，才能判断哪些 task 的产出 RP 仍存在，决策无需重启的 task

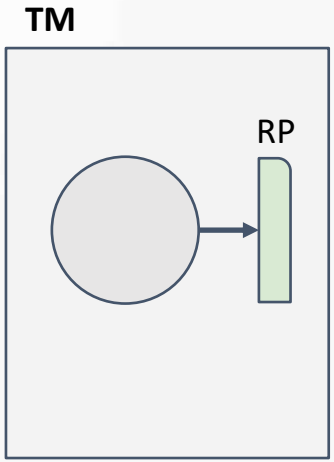
⇒ 需重构 RP 的生命週期管理，集中化管控

插件化 Shuffle Manager

[FLIP-31] Pluggable Shuffle Manager

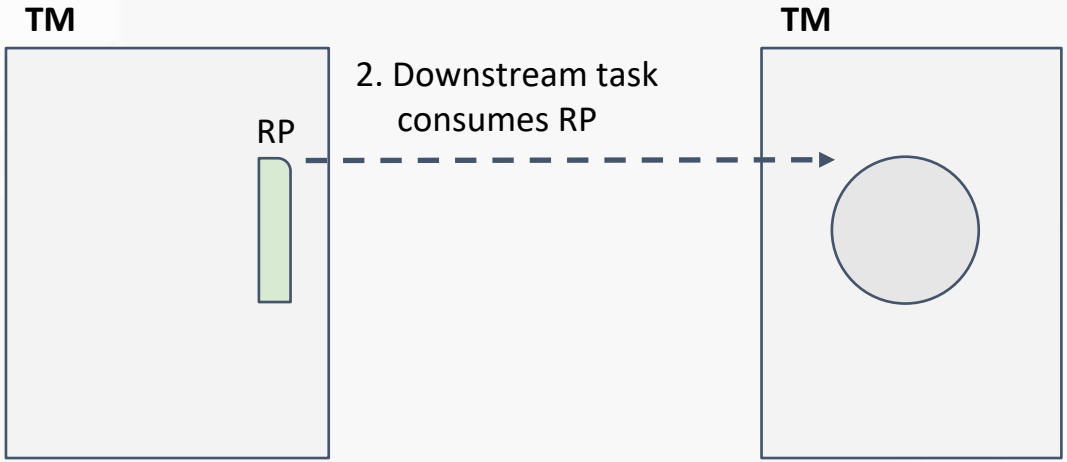
- 解决问题：
  - **重构 Result Partitions (RP) 生命週期，集中化管控**
    - 使单一 RP 能够被重複消费
    - 方便 scheduler 查询仍存在的 RP 以决策错误发生需重启的 task
  - **解耦 TM 需同时负责产生 RP 与传输 RP (Shuffle) 之职责**
    - 使 TM 能够在 Task 结束运算产生完 RP 后就提早释出运算资源
    - 方便扩增不同 shuffle / RP 储存的实现
      - 外部 shuffle service, 如 YARN external shuffle service
      - 将 RP 储存到 external storage，以脱离 job 的生命週期供跨 job 使用

插件化 Shuffle Manager

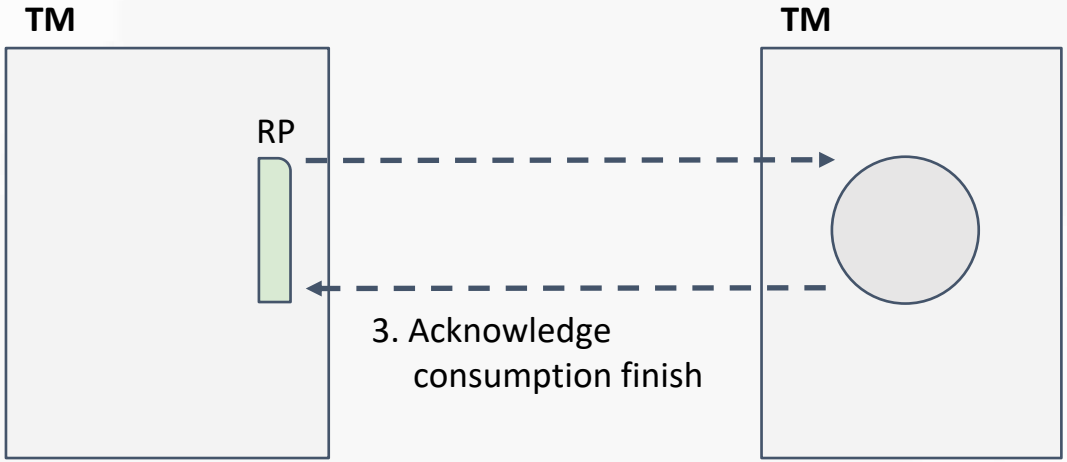


1. Producing task creates internal shuffle service (Netty) and generates RP

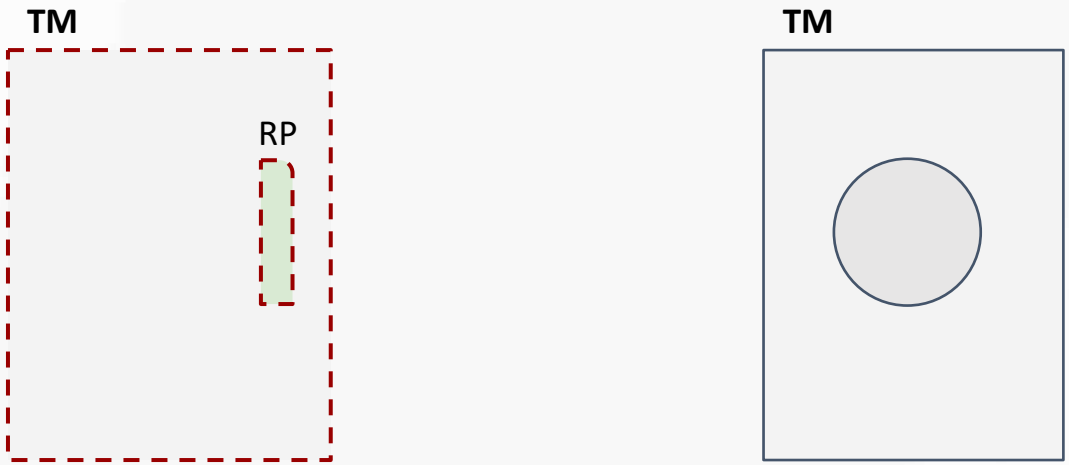
插件化 Shuffle Manager



插件化 Shuffle Manager



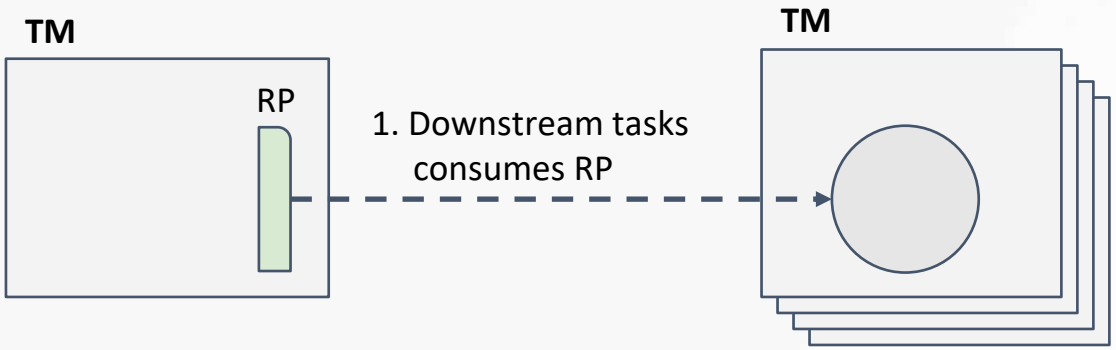
插件化 Shuffle Manager



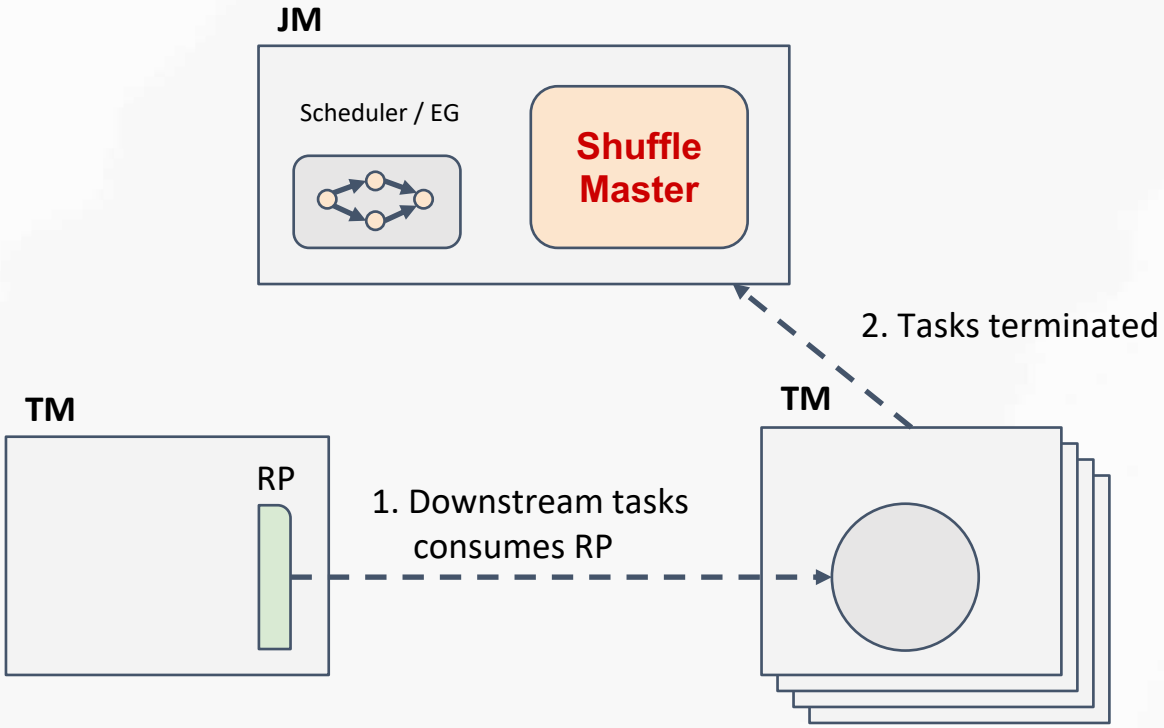
4. Release RP + task executor

Apache Flink

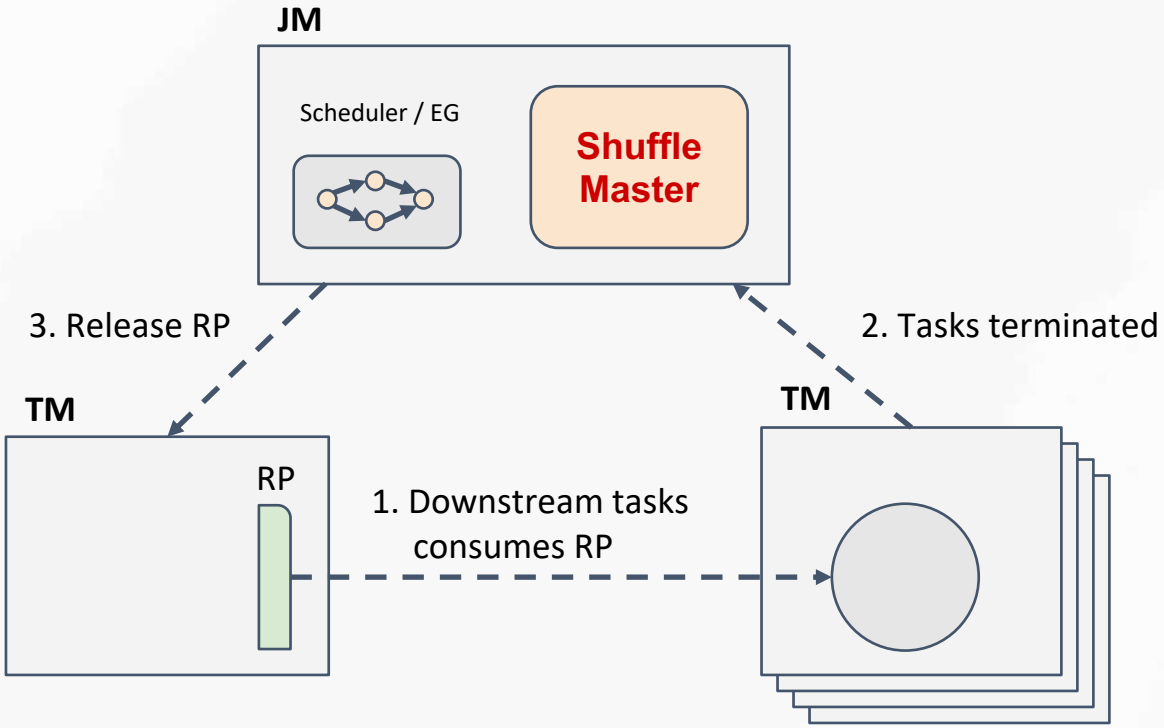
插件化 Shuffle Manager



插件化 Shuffle Manager



插件化 Shuffle Manager



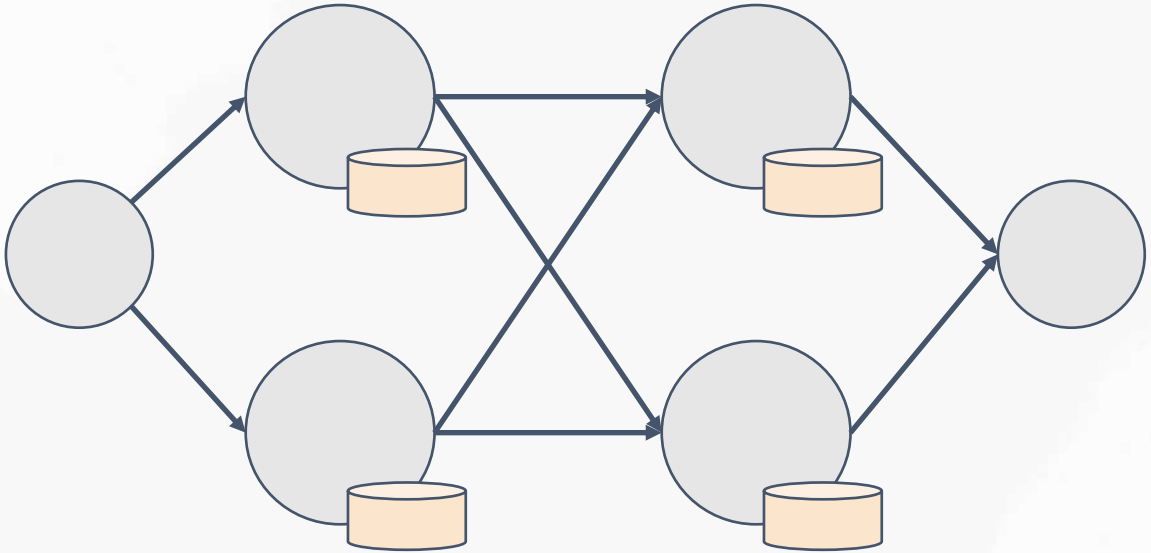
04

生态

Ecosystem

State Processing API

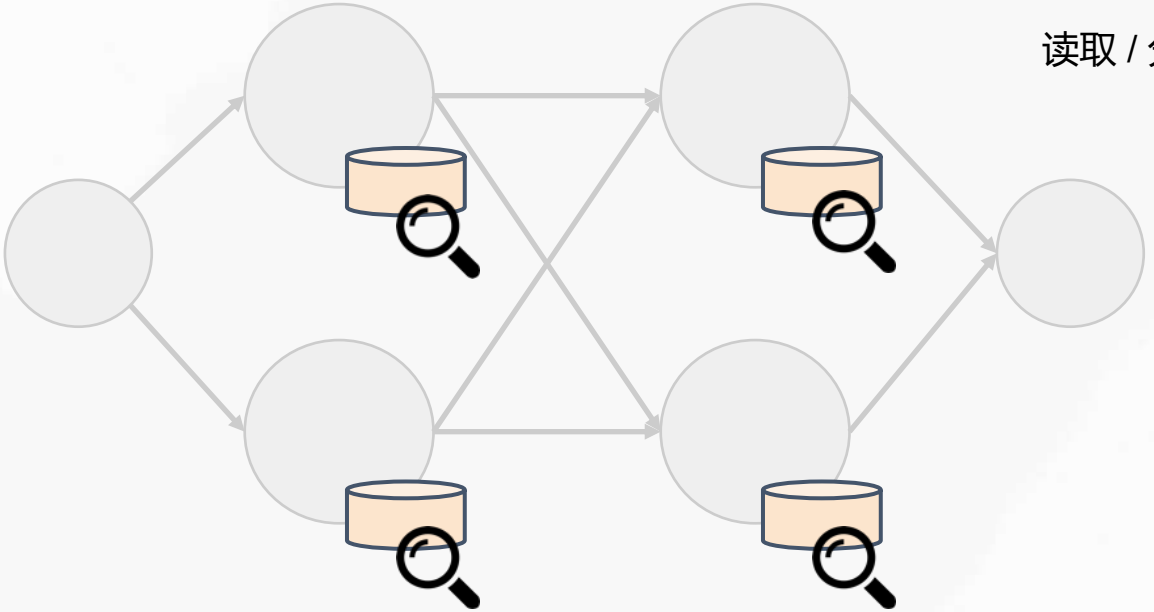
[FLIP-43] State Processing API



Apache Flink

State Processing API

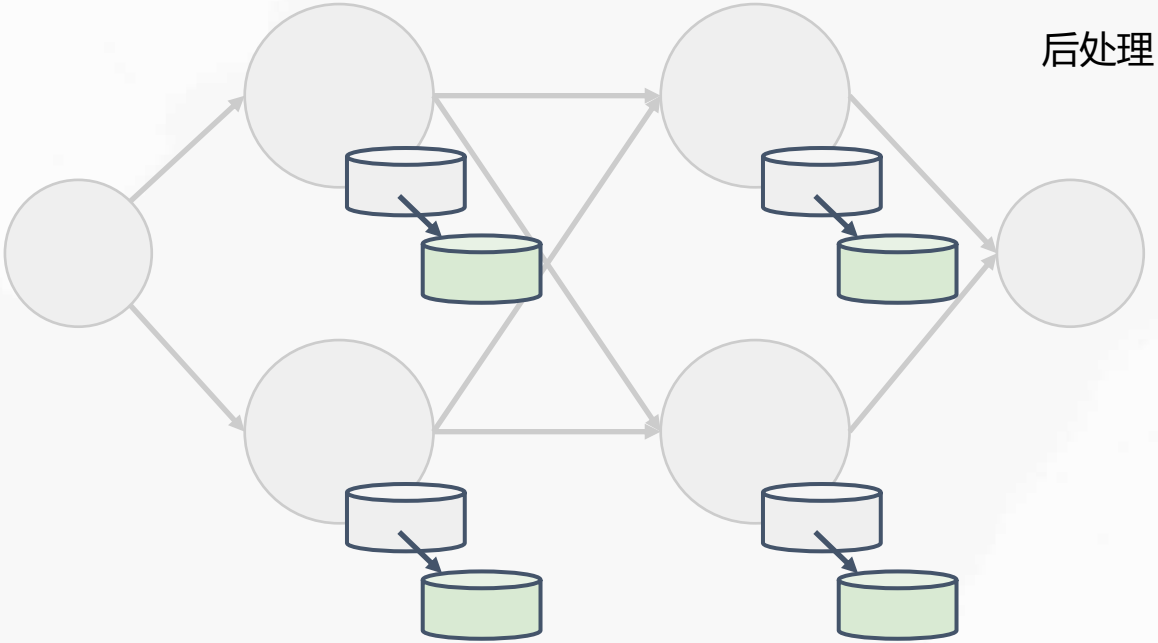
[FLIP-43] State Processing API



读取 / 分析状态数据

State Processing API

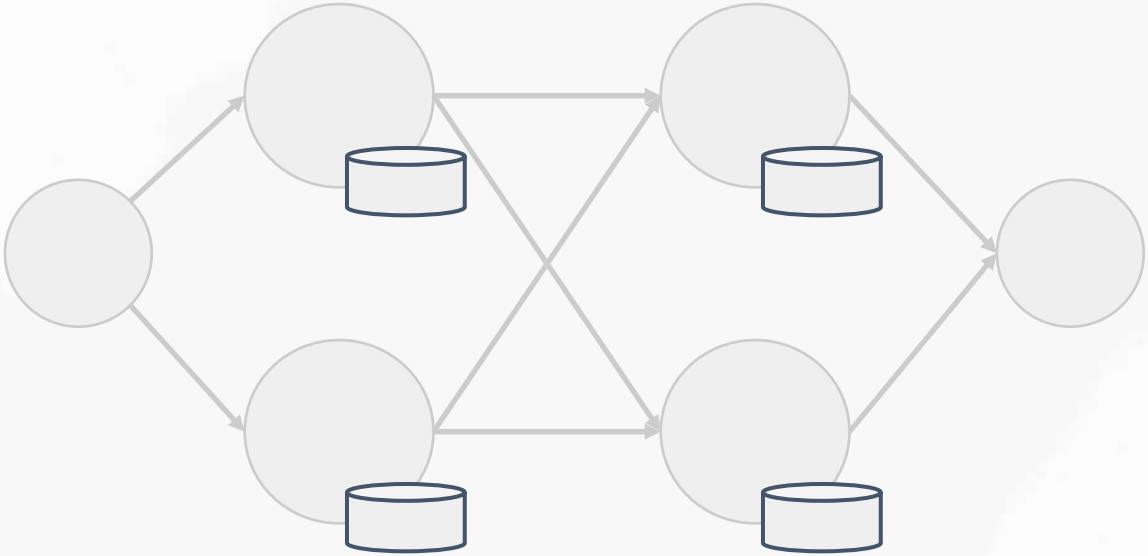
[FLIP-43] State Processing API



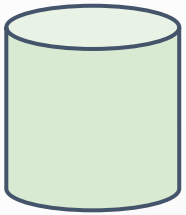
后处理 / 修补状态数据

State Processing API

[FLIP-43] State Processing API



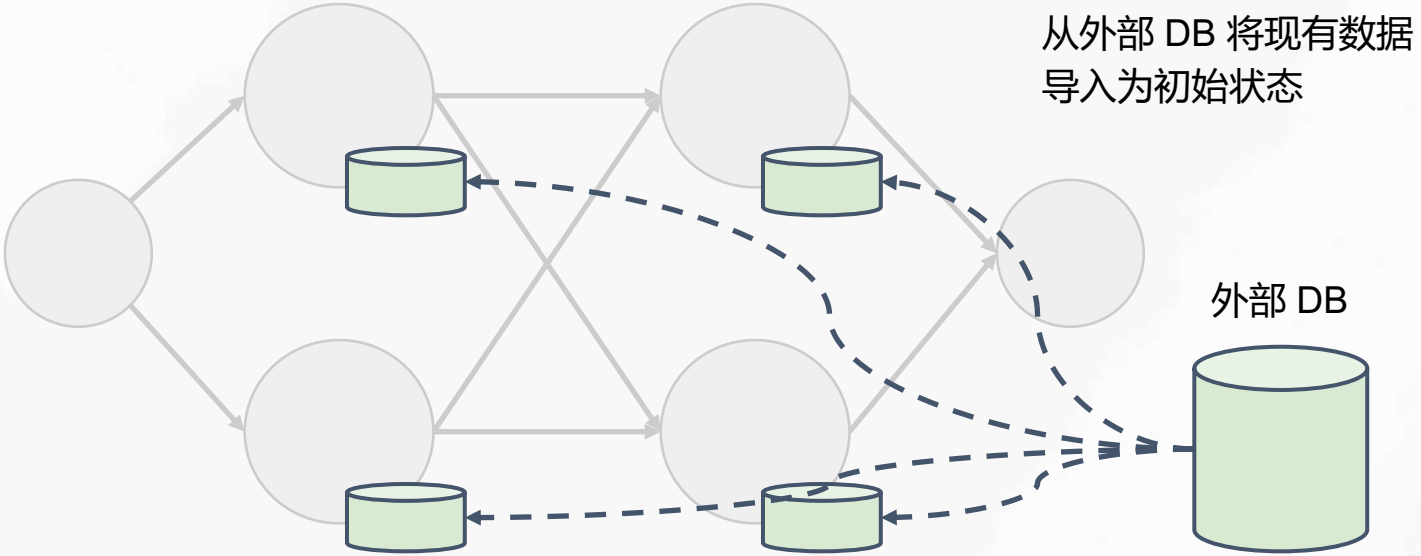
外部 DB



Apache Flink

State Processing API

[FLIP-43] State Processing API



State Processing API

- 从用户角度：
  - 增加 Flink 流式应用内部状态的可及性
  - 修补 / 迁移状态数据更容易，不需要回放 source 讯息储列
  - 直接引入现有外部数据成为流式应用初始状态
- 从开发者角度：
  - 减少核心代码中用于应付状态多版本间格式相容的读写逻辑

Apache Flink

State Processing API

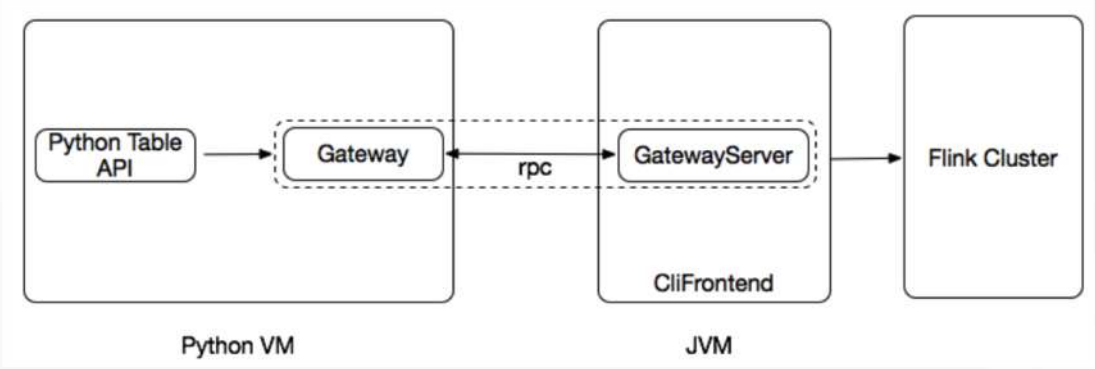
```
ExecutionEnvironment env = ExecutionEnvironment.getExecutionEnvironment();
ExistingSavepoint savepoint = Savepoint.load("/savepoint/path", env);

DataSet<MyState> loadedKeyedState = savepoint
    .readKeyedState("operatorUid", new MyKeyedStateReaderFunction());

DataSet<MyProcessedState> processedKeyedState = loadedKeyedState
    …

BootstrapTransformation<MyProcessedState> bootstrapOperator = OperatorTransformation
    .bootstrapWith(processedKeyedState)
    .transform(new MyStateBootstrapFunction());

savepoint
    .withOperator("operatorUid", bootstrapOperator)
    .write("/new/savepoint/path");
```

Python Table API

[FLIP-38] Python Table API

- 1.9.0 版 PyFlink 主要功能：
  - 架构上，利用 Py4j 建立了 PythonVM 与 JVM 的通讯
  - 支援所有 Java Table API 固有的功能 (除了 UDF / UDTF / UDAF)
  - 直接透过原有 CLI 提交 Python Table API 应用
  - 支持互动式开发 Interactive Shell

Python Table API - 架构设计



- **目标：直接基于现有的 Java Table API 上实现 Python API**
  - Python Table API 只需定义 API 介面
  - 透过 rpc 去直接呼叫对应的 Java Table API
  - e.g. `TableEnvironment` / `Table` / `TableSink`, 等皆会有对应的 Python wrapper

Apache Flink

Python Table API - 范例代码

```
from pyflink.dataset import ExecutionEnvironment
from pyflink.table import BatchTableEnvironment, TableConfig
from pyflink.table.descriptors import FileSystem, OldCsv, Schema
from pyflink.table.types import DataTypes

def word_count():
    t_config = TableConfig()
    env = ExecutionEnvironment.get_execution_environment()
    t_env = BatchTableEnvironment.create(env, t_config)

    elements = [(word, 1) for word in content.split(" ")]
    t_env.from_elements(elements, ["word", "count"]) \
         .group_by("word") \
         .select("word, count(1) as count") \
         .insert_into("Results")

    env.execute()
```

Apache Flink

Python Table API

- 沿用现有的 Flink CLI 做应用提交

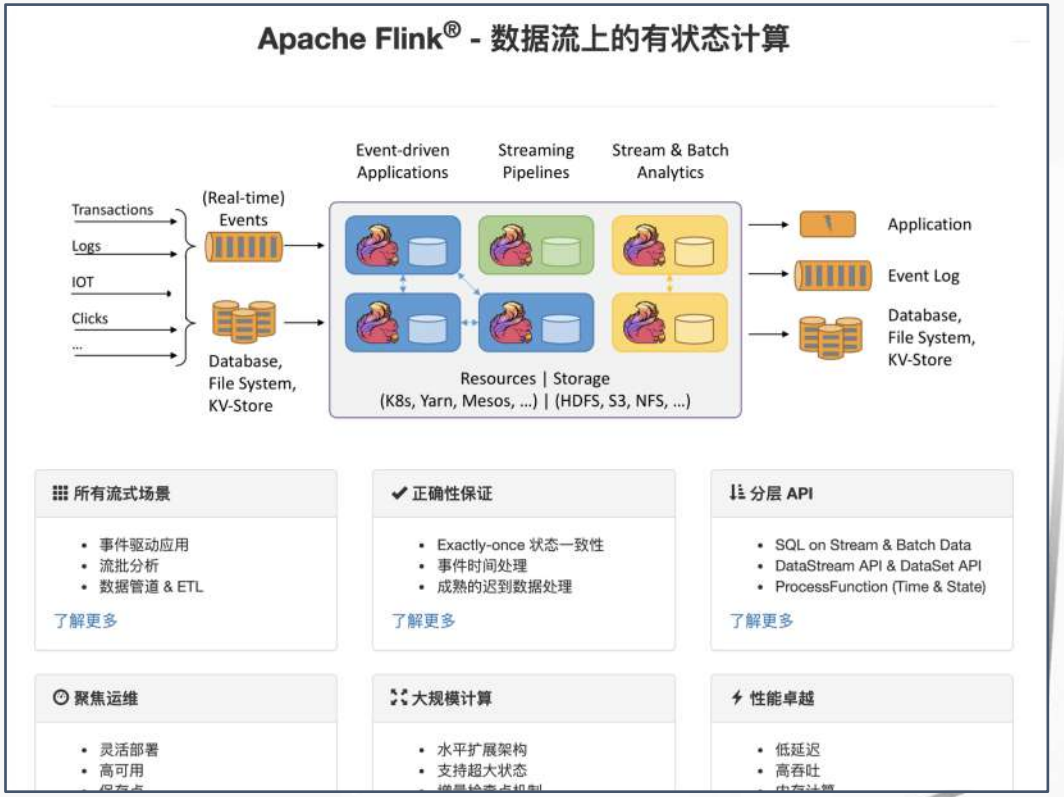
```
./bin/flink run \
      -py examples/python/table/batch/word_count.py \
      -j <path/to/flink-table.jar>
```

- 支援 Interactive Shell

```
./bin/pyflink-shell.sh [local|remote]
```

Apache Flink

官方网页 / 文档中文翻译



Apache Flink

官方网页 / 文档中文翻译

所有文档翻译相关任务项目：[點我]

Search Save as

Flink Type: All Status: All Assignee: All Contains text More Advanced

Resolution: Unresolved Component: chinese-translation

1–34 of 34

T	Patch Info	Key	Summary	Assignee	Reporter	P ↓	Status
		FLINK-12833	FLINK-11526 / Add Klaviyo to Chinese PoweredBy page	*Unassigned*	Fabian Hueske		OPEN
		FLINK-12930	FLINK-11526 / Update Chinese "how to contribute" pages	*Unassigned*	Robert Metzger		OPEN
		FLINK-11622	FLINK-11529 / Translate the "Command-Line Interface" page into Chinese	Nicholas Jiang	Hui Zhao		IN PROG
		FLINK-12938	FLINK-11529 / Translate "Streaming Connectors" page into Chinese	aloyszhang	Jark Wu		OPEN
		FLINK-12427	FLINK-11529 / Translate the "Flink DataStream API Programming Guide" page into Chinese	YangFei	YangFei		IN PROG
		FLINK-11529	Support Chinese Documentation for Apache Flink	Jark Wu	Jark Wu		IN PROG
		FLINK-12438	FLINK-11529 / Translate "Task Lifecycle" page into Chinese	Armstrong Nova	Armstrong Nova		IN PROG
		FLINK-11634	FLINK-11529 / Translate "State Backends" page into Chinese	jeremy huang	Congxian Qiu(klion26)		OPEN
		FLINK-11637	FLINK-11529 / Translate "Checkpoints" page into Chinese	guanghui.rong	Congxian Qiu(klion26)		OPEN

Apache Flink

官方中文使用者邮件列表

Apache Flink 中文用户邮件列表

This forum is an archive for the mailing list **user-zh@flink.apache.org** (more options) Messages posted here will be sent to this mailing list.
Apache Flink 中文用户邮件列表

New Topic People Options

Topics (20)	Replies	Last Post	Views
Flink1.8+Hadoop3.1.2 编译问题 by CHENJIE	3	Jun 28 by CHENJIE	4
checkpoint stage size的问题 by ReignsDYL	6	Jun 28 by CHENJIE	49
Flink如何实现Job间的协同联系？ by 徐涛	3	Jun 28 by Hequn Cheng	18
Flink 窗口触发疑问 by 雷水鱼	0	Jun 27 by 雷水鱼	4
Re: checkpoint stage size的问题 by ReignsDYL	0	Jun 26 by ReignsDYL	7
来自小乐的邮件 by 小乐	0	Jun 26 by 小乐	3
关于使用Flink建设基于CDC方式的OGG数据湖 by 唐门小师兄	0	Jun 26 by 唐门小师兄	6
blink 版本 消费kafka 看不到group id by 雷水鱼	3	Jun 25 by Biao Liu	8
你好！ by 杨胜松(鼓翅)	1	Jun 25 by Biao Liu	7
flink filesystem 1.7.2 on Hadoop 2.7 BucketingSink.reflectTruncat() 有写入很多小文件到hdfs的风险 by 巫旭阳	2	Jun 24 by 巫旭阳	2
Flink程序长期运行后报错退出 PartitionRequestQueue - Encountered error while consuming partitions by 罗学焕/予之	1	Jun 24 by Biao Liu	15
Flink tps 速度问题 by haibin	1	Jun 24 by Biao Liu	14
flink连续窗口 by 残翅2008	1	Jun 24 by Biao Liu	7
EventTimeTrigger源码求帮忙解读，求各位大佬帮帮忙，不胜感激 by 840124434	3	Jun 24 by Shi Quan	12
Flink任务资源动态规划 by 15904502343@163.com	1	Jun 21 by 田志声	5
为何会报"Window can only be defined over a time attribute column."？？ by Chennet Steven	1	Jun 21 by Chennet Steven	1

官方中文使用者邮件列表

- 中文邮件列表订阅方式：
  - 寄一封邮件到 user-zh-subscribe@flink.apache.org
  - 系统会自动寄发订阅确认邮件
  - 直接回复该邮件，即可订阅
- 订阅成功后将收到 Flink 中文邮件列表的所有消息，提问，回复等
- 尚未订阅的人**无法**对邮件列表发送邮件
- 订阅指南视频：https://www.bilibili.com/video/av57076677/

Apache Flink

05

未来版本计划

Future Plan

未来版本计划

- **SQL**
  - 继续完善 Type System 和 DDL
  - Ground Source Sink Concepts in Flink SQL
- **Runtime**
  - DataStream 增加批处理功能
  - [FLIP-27] Unified runtime source API
  - [FLIP-41] Unify Binary format for Keyed State
  - Disk-spilling heap backend
- **生态**
  - 原生支持 Protobuf / Thrift 等序列化格式
  - 支持 Python UDF
  - 继续完善 ML 算法库
  - 继续完善 Hive 兼容性

Apache Flink

THANKS