袋鼠云 DTSTACK

袋鼠学院 DTSTACK COLLEGE

Flink Checkpoint与Barrier

演讲人—土豆

数连万物，栈通中台

目录

CONTENTS

PART 01 checkpoint介绍

袋鼠学院 DTSTACK COLLEGE

- checkpoint是什么？

  checkpoint是Flink用于保证自身可靠性的一种机制，其基于Chandy-Lamport（CL）算法思想改进而来，可以在不停止整个流处理系统的前提下，让每个节点独立建立检查点保存自身快照，并最终达到保存整个作业全局快照的状态。

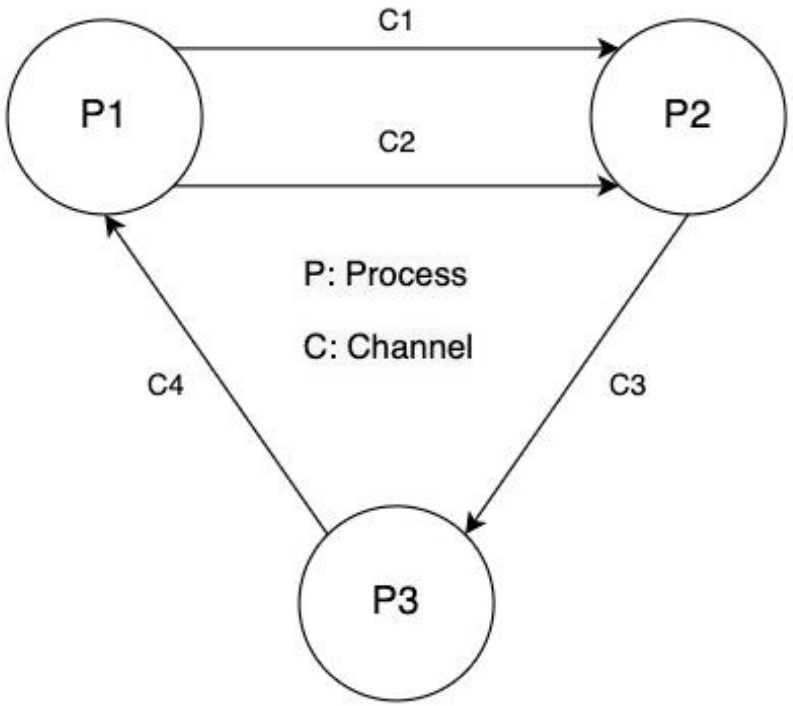
- 为什么需要checkpoint？

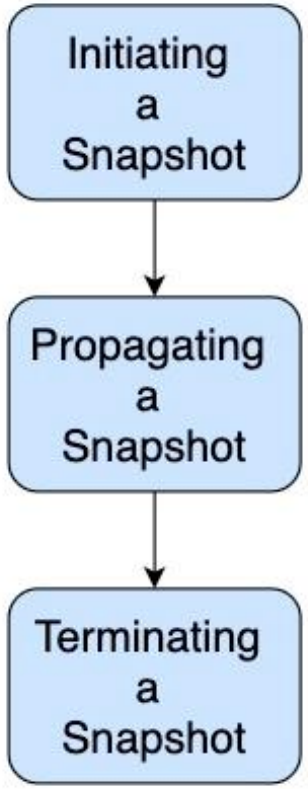
  Flink是有状态的流计算处理引擎，每个算子Operator都有可能记录自己的状态，并且随着数据的流入状态也不断发生变更。当程序发生故障时，需要一种机制能够帮助Flink任务恢复到故障前的状态，因此引入了checkpoint机制。

- checkpoint是Flink容错的核心

PART 01 分布式快照算法-Chandy-Lamport（CL）算法

- 要点：系统整体状态包含进程的状态以及链路中消息的状态
- 难点：1、链路中消息的状态不容易记录；2、进程时间不一定同步，无法同时记录状态
- 核心：每个进程记录与自己相关的状态合并出全局状态
- 目标：1、最终产生的快照必须保证一致性；2、快照过程不能影响系统正常运行，更不能stop the world。





假设进程P1发起snapshot：
1、P1记录自身的状态
2、P1向C1、C2中发送Marker
3、P1监听C4流向自己的消息

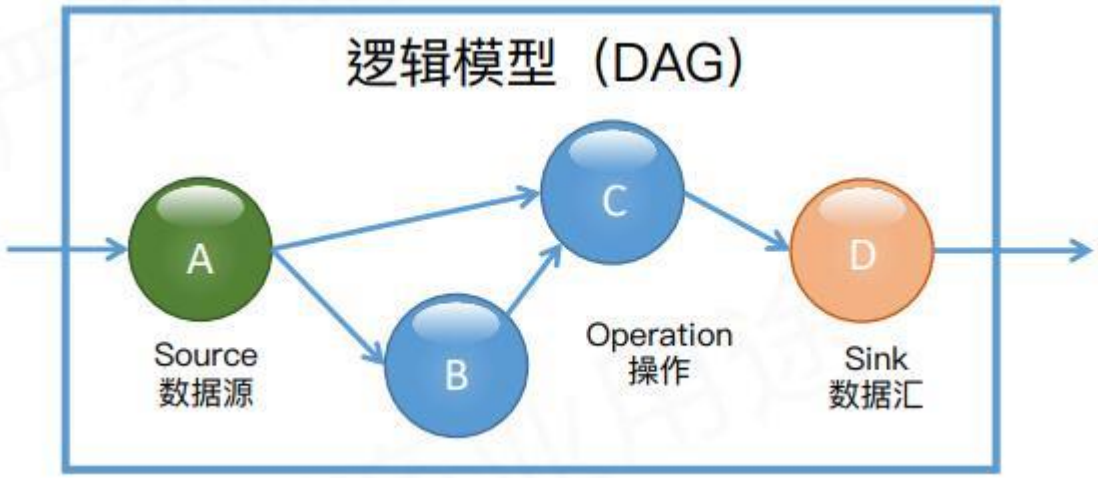
对于任意进程P，当收到Marker时没有记录自身的状态：
1、P记录自身的状态
2、P向所有下游链路C发送Marker
3、P监听所有上游链路C流向自己的消息

若已经记录自身的状态：
1、P记录下监听到的所有消息

若所有进程P都成功地：
1、收到了Marker消息
2、记录了自身的状态
3、记录下监听到的所有消息

PART 01 Flink的轻量级异步屏障快照（ABS）算法

轻量级异步屏障快照（ABS）算法是CL算法的变种。



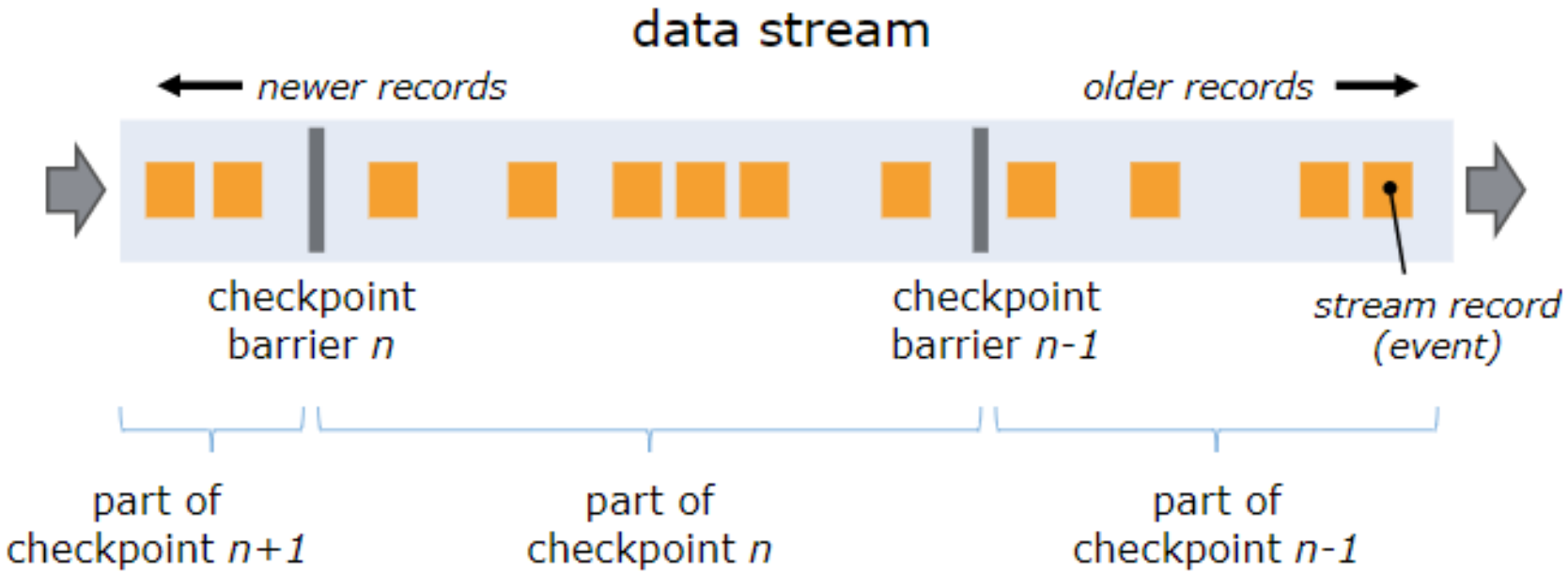
- 算子(operator) -> 进程(process)
- 数据流(data stream) -> 链路(channel)
- 屏障(barrier) -> marker

类比可以得出如下几点：

- 要点：Flink作业保存的状态包含两部分：1、算子的状态；2、数据流中数据的状态
- 难点：1、数据流中数据的状态不容易记录；2、进程时间不一定同步，无法同时记录状态
- 核心：1、每个算子记录与自己相关的状态合并出全局状态；2、引入barrier
- 目标：1、最终产生的快照必须保证一致性；2、快照过程不能影响系统正常运行，更不能stop the world。

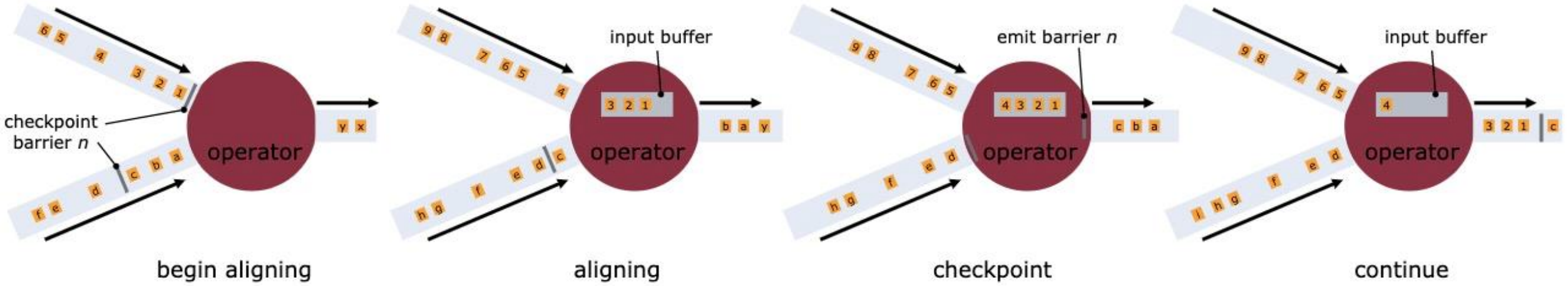
PART 01 Barrier介绍

- Barrier是Flink中一种特殊的内部消息，它对应的是CL算法中的marker，用于标记切分数据集。
- 在进行Checkpoint的时候Flink会在数据流源头处周期性地注入Barrier，这些Barrier会作为数据流的一部分，一起流向下游节点并且不影响正常的数据流。
- 每个Barrier都带有一个快照ID，一个Barrier生成之后，在这之前的数据都进入此快照，在这之后的数据则进入下一个快照。



PART 01 Barrier对齐

当输入流大于一条时，在Exactly once语义下，Flink会执行barrier对齐操作。



详见：CheckpointBarrierHandler#processBarrier

PART 01 Exactly Once vs. At Least Once

- Exactly Once

  Barrier对齐是实现精准一次语义的基础，能够保证多输入流的算子正常处理不同checkpoint区间的数据，避免它们发生交叉，即不会有数据被处理两次。

- At Least Once

  但是对齐过程需要时间，有一些对延迟特别敏感的应用可能对准确性的要求没有那么高。所以Flink也允许在StreamExecutionEnvironment.enableCheckpointing()方法里指定At-Least-Once语义，会取Barrier对齐，即算子收到第一个输入的屏障之后不会阻塞，而是触发快照。这样一来，部分属于检查点n + 1的数据也会包括进检查点n的数据里， 当恢复时，这部分交叉的数据就会被重复处理。

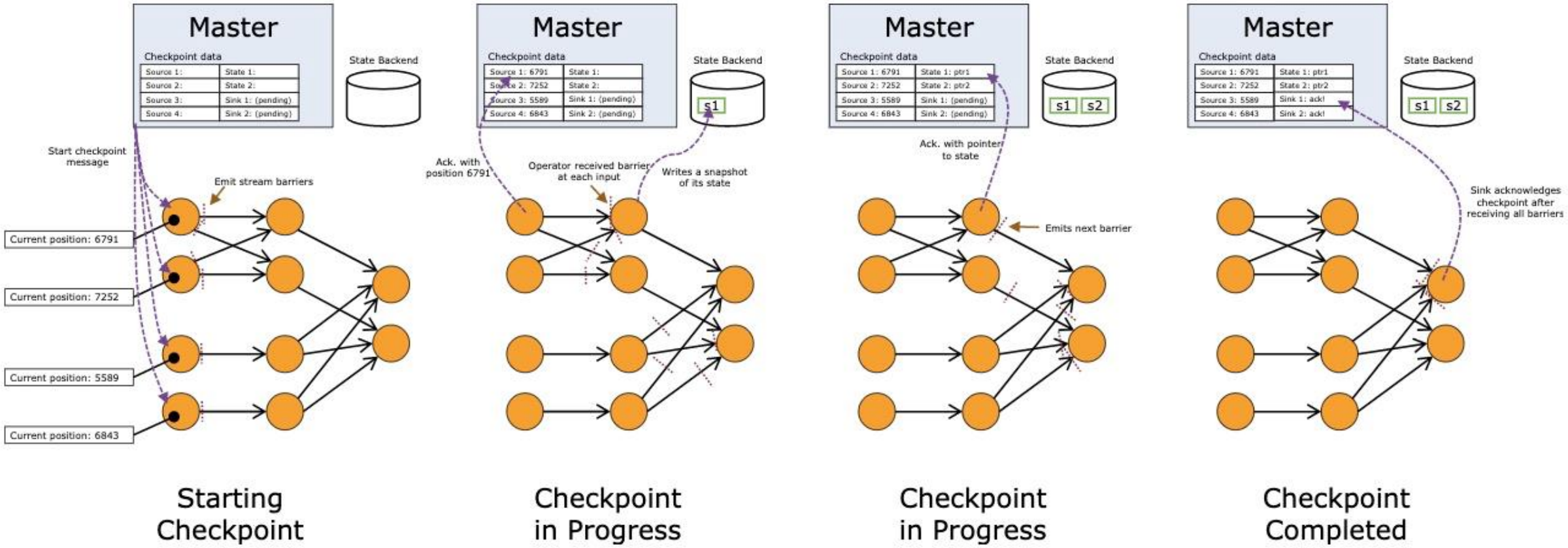
详见：InputProcessorUtil#createCheckpointBarrierHandler

目

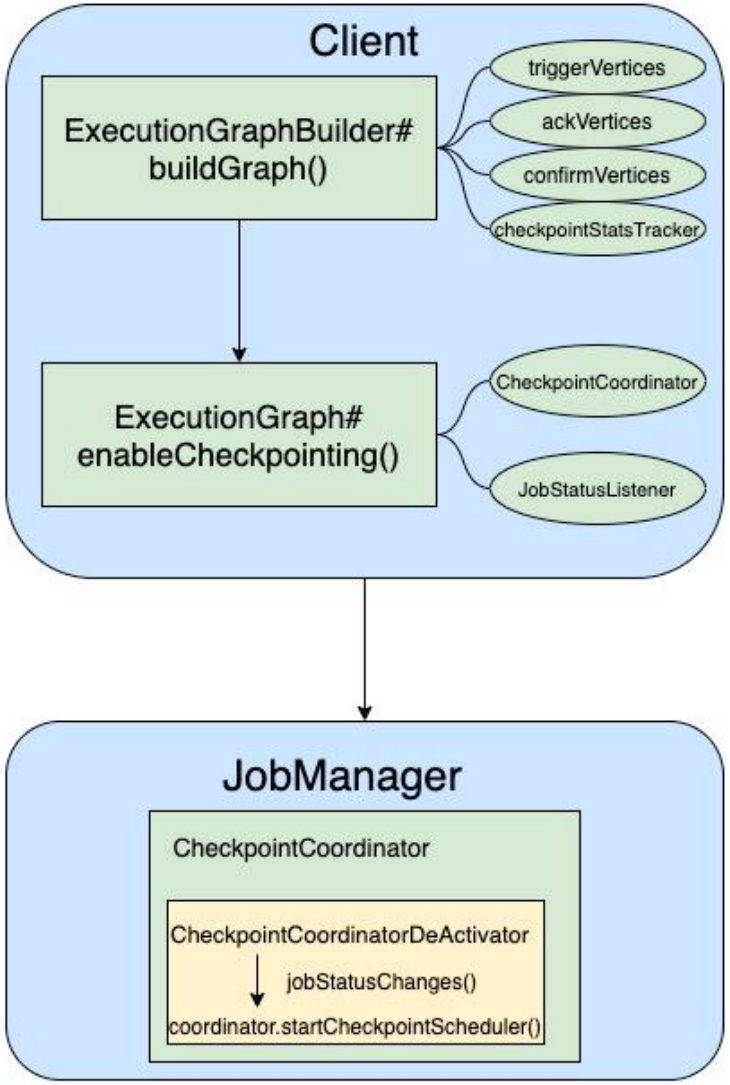
CONTENTS

录

PART 02 Checkpoint整体流程



PART 02 任务提交阶段



Client端：

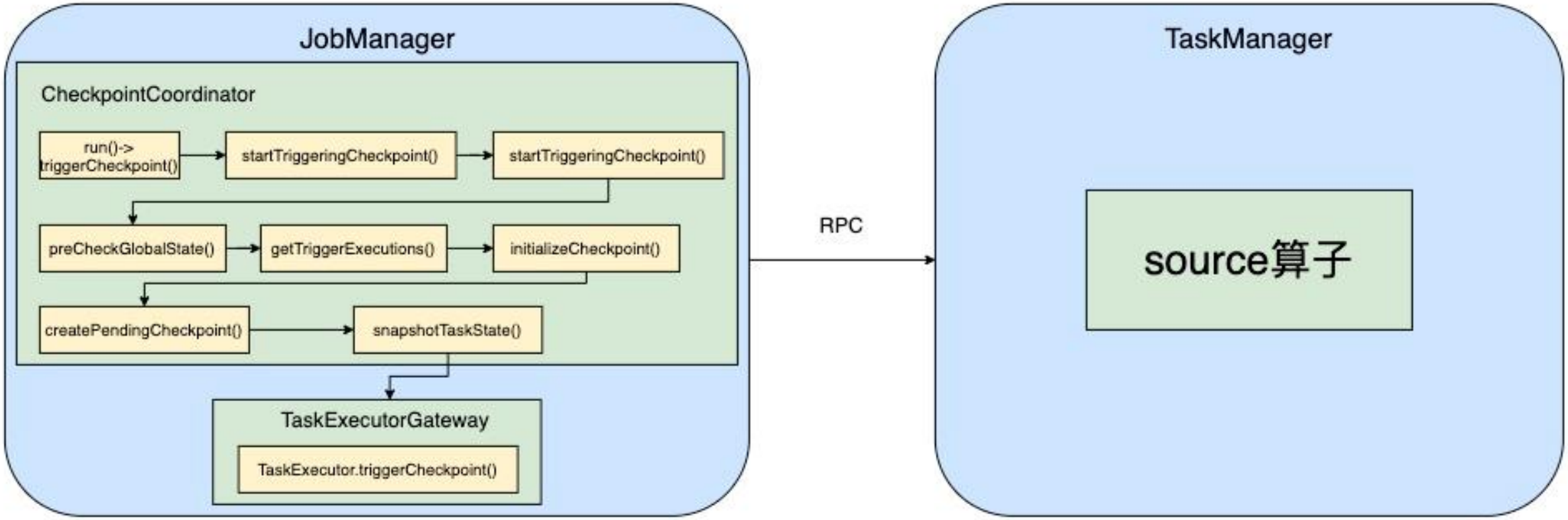
在buildGreaph()中
1、初始化triggerVertices
2、初始化ackVertices
3、初始化confirmVertices
4、构建checkpointStatsTracker

接着调用enableCheckpointing()
1、构建CheckpointCoordinator
2、构建JobStatusListener

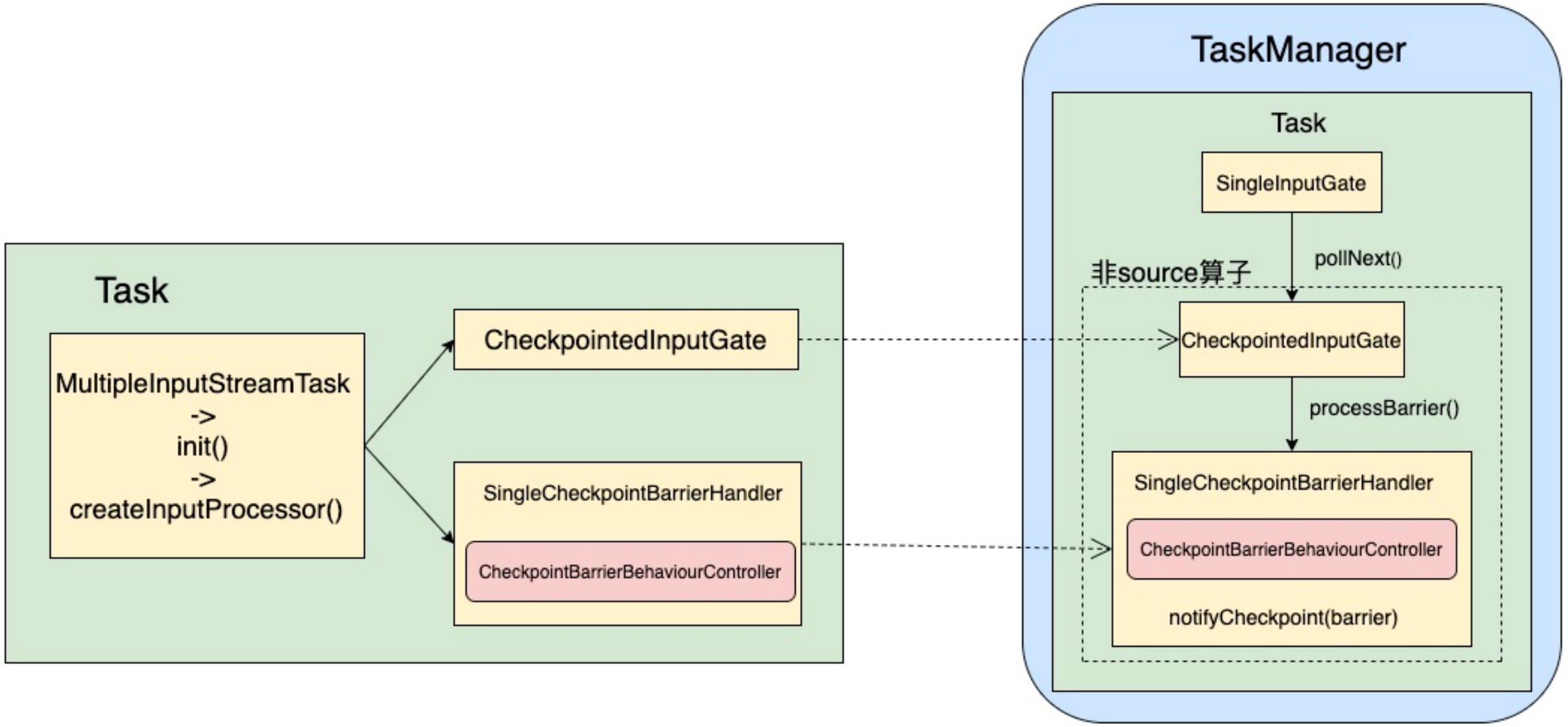
JobManager：

当任务状态变为Running时，实现JobStatusListener接口的CheckpointCoordinatorDeActivator对象中的jobStatusChanges()被调用，CheckpointCoordinator正式启动。

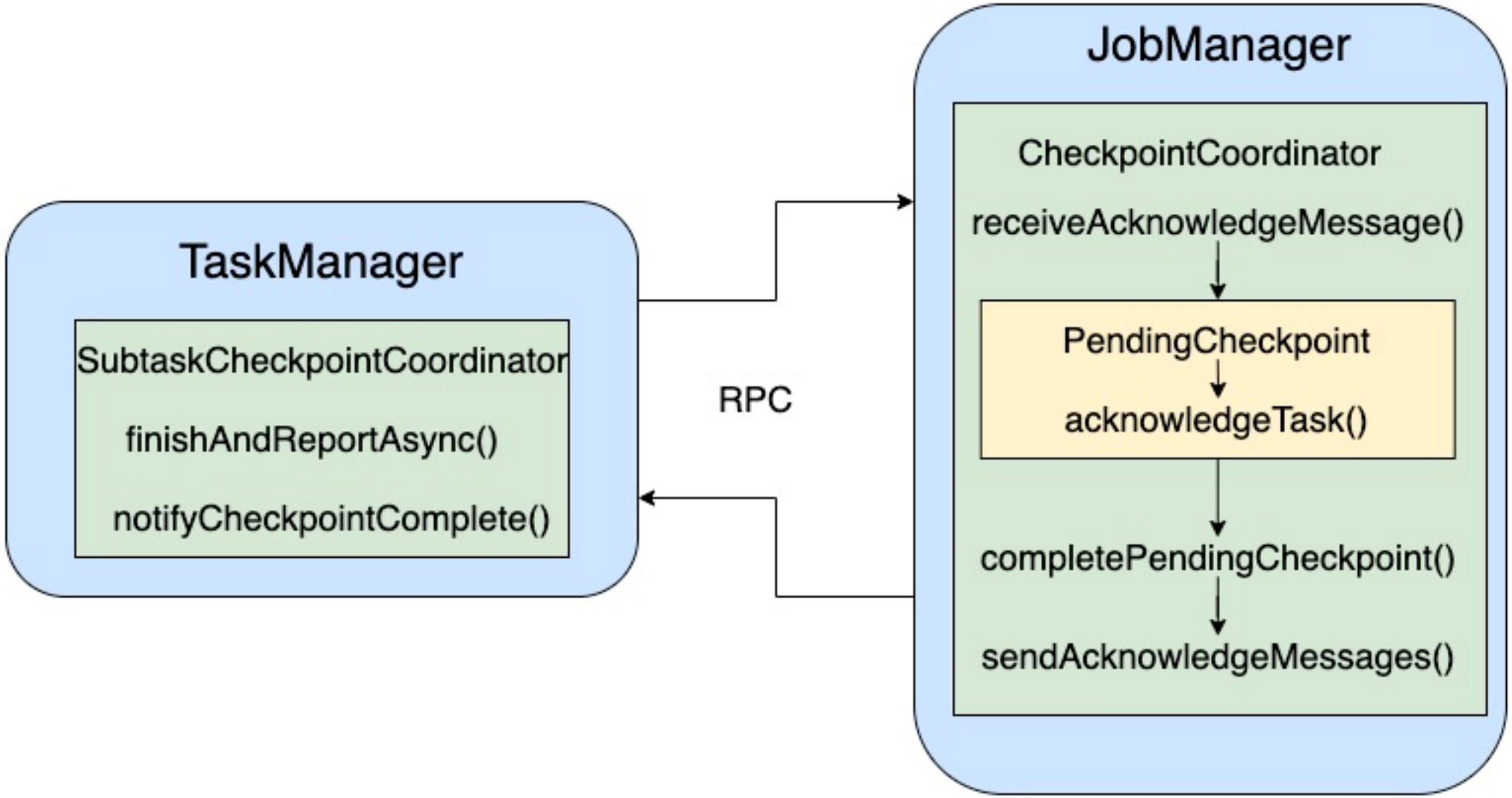
PART 02 Starting Checkpoint



PART 02 Checkpoint in Process



PART 02 Checkpoint Compeleted



袋鼠云
DTSTACK

袋鼠学院
DTSTACK COLLEGE

数据智能
让未来变成现在

400 002 1024

www.dtstack.com

袋鼠云公众号