Unaligned Checkpointing

Flink 1.11

Checkpoint/Snapshot Quick Review

- Capture Complete Global State (*maintains a summary of data seen so far)*
  - **Keyed-State**: computation result mapped by user-defined keyspace
  - **Operator-State**: the state that cannot be scoped by a key
- Snapshot Usages:
  - Reconfiguration (checkpoint-stop-modify-restore)
    - logic update
    - versioning
  - Rollback recovery
    - task failure: only the affected tasks are reconfigured.
    - *rescale:* all tasks are being redeployed

Aligned Checkpointing

When Alignment is needed?

1. all operators with multiple inputs
2. operators after a shuffle (consume data from multiple upstream)

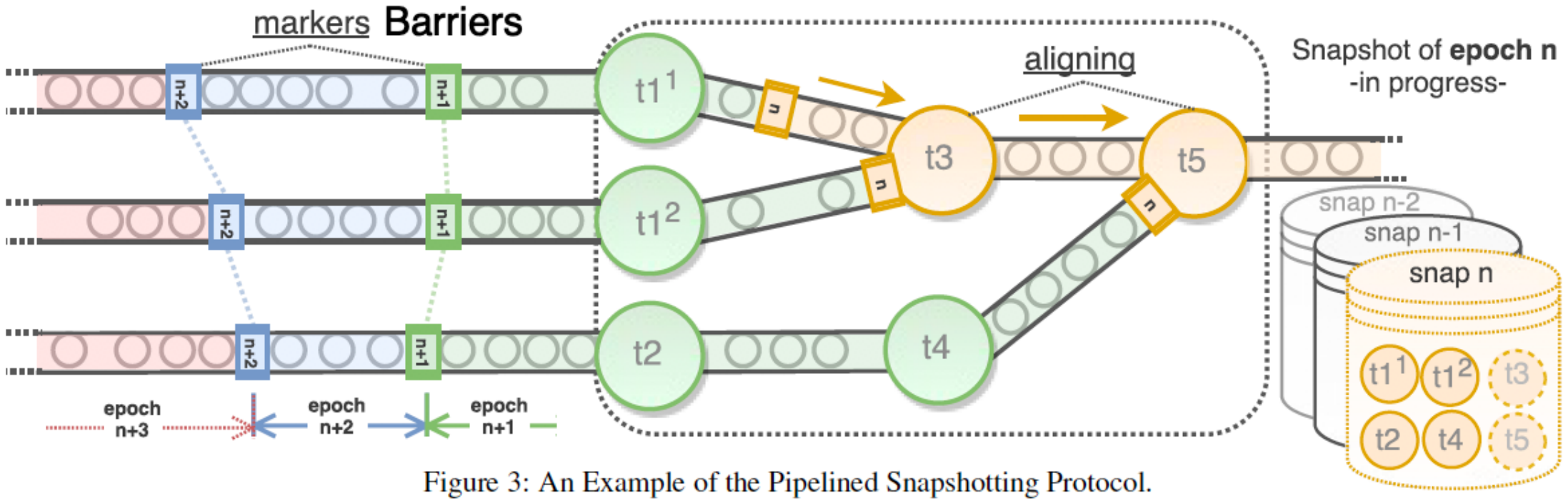
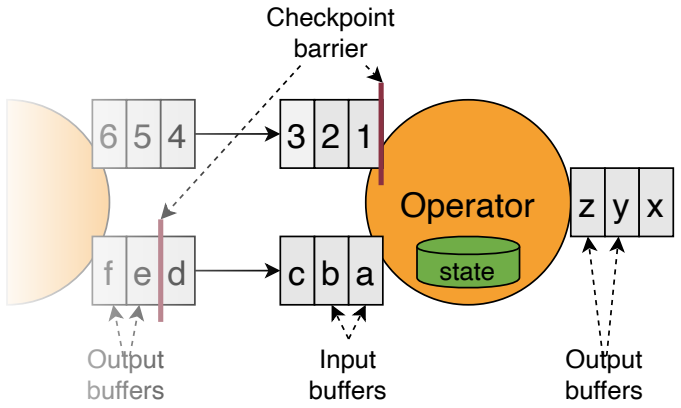
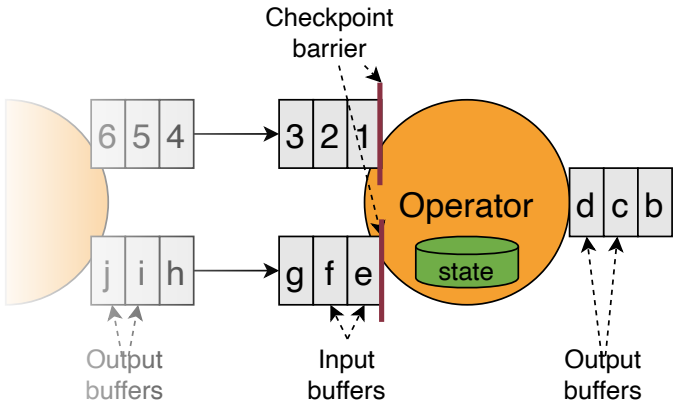


Figure 3: An Example of the Pipelined Snapshotting Protocol.

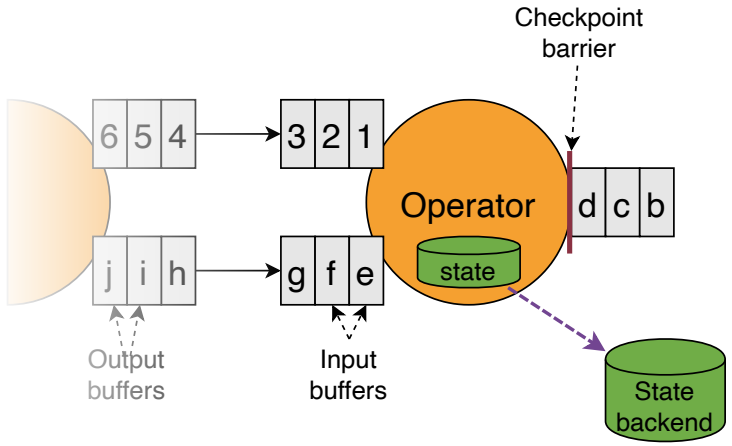
Aligned v.s. Unaligned



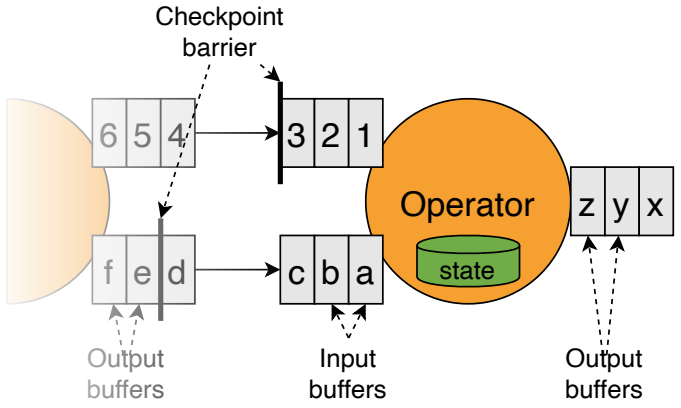
Begin alignment



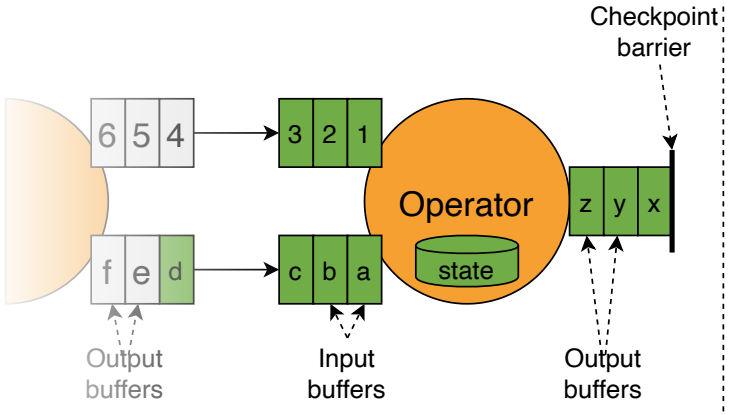
End alignment



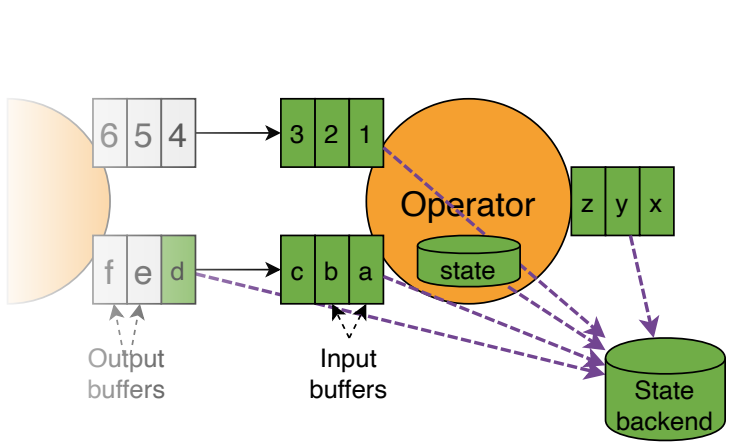
Checkpoint



On first barrier



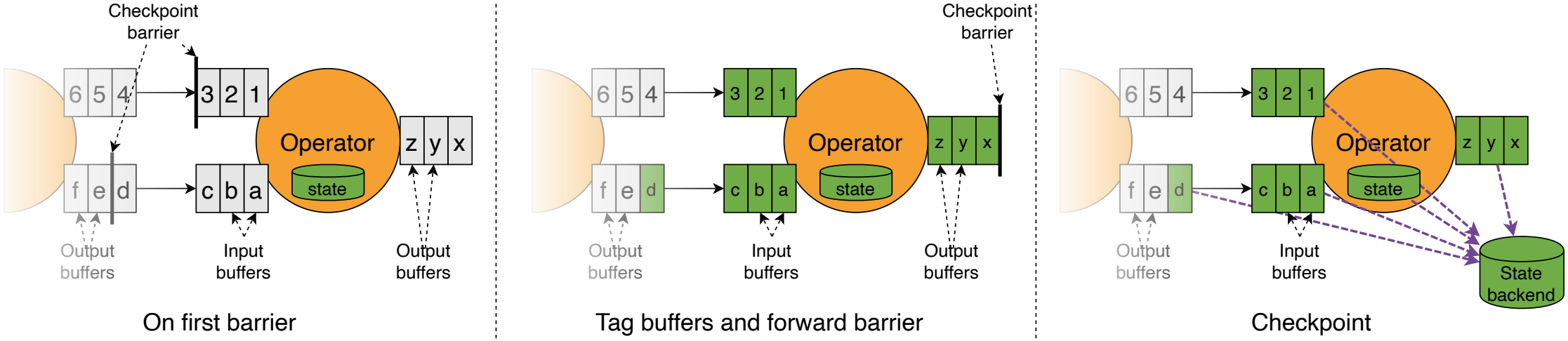
Tag buffers and forward barrier



Checkpoint

Unaligned Checkpointing

- Reacts on the **first barrier** that is stored in its input buffers.
- Immediately forwards the barrier to the downstream operator by adding it to the end of the output buffers.
- Marks all overtaken **records to be stored** asynchronously and creates a snapshot of its own state.



Unaligned Checkpointing

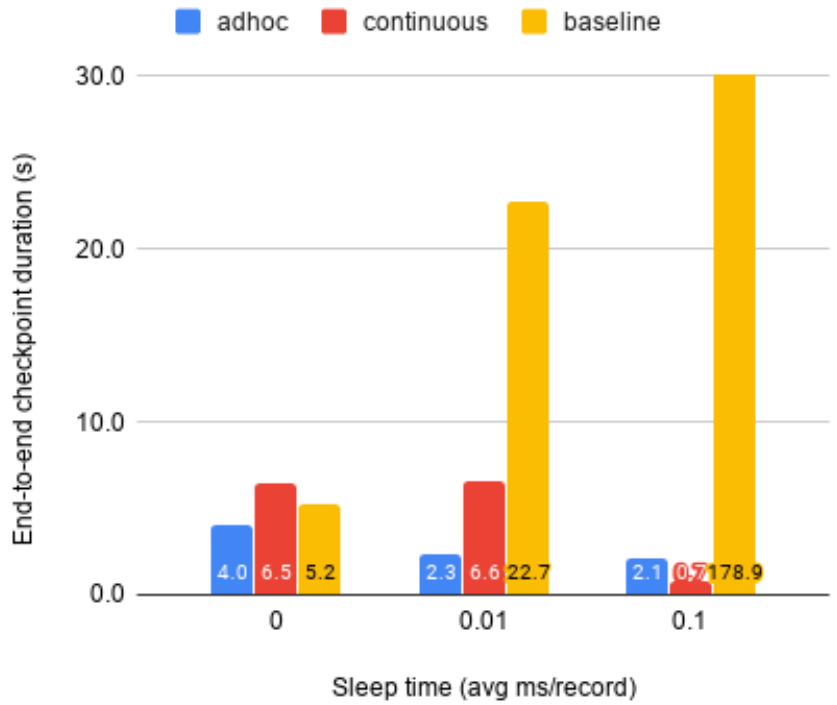
- Reacts on the **first barrier** that is stored in its input buffers.
- Immediately forwards the barrier to the downstream operator by adding it to the end of the output buffers.
- Marks all overtaken **records to be stored** asynchronously and creates a snapshot of its own state.

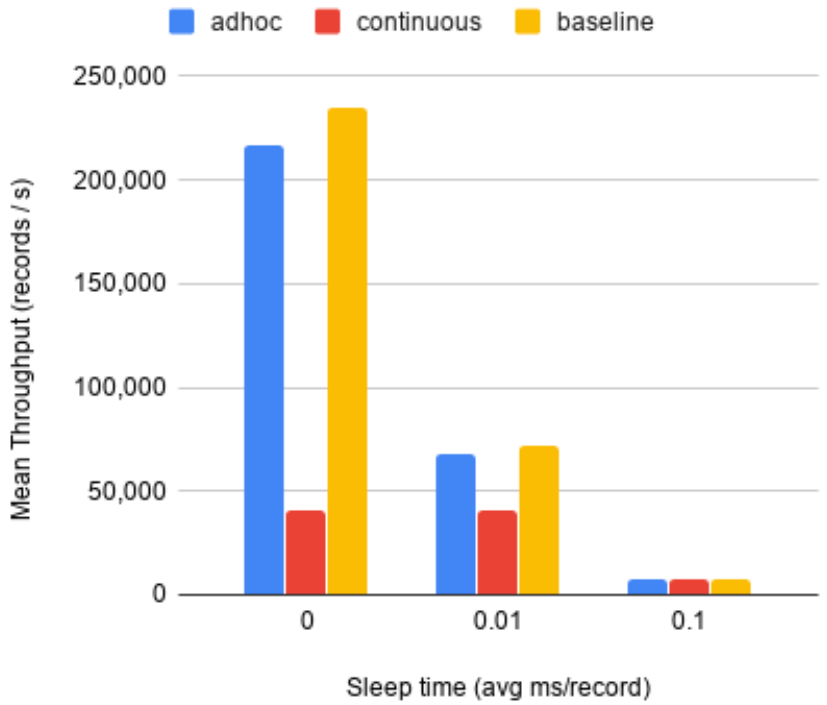
- Ensures that barriers are arriving at the sink as **fast** as possible
- Suited for applications with **long alignment time**
- But **add I/O** pressure

Summary

- Unaligned checkpoints contain in-flight data (i.e., data stored in buffers) as part of the checkpoint state, which allows checkpoint barriers to overtake these buffers.
- Trade-off between
  - storage size for in-flight data
  - checkpoint latency and time where an input channel is blocked

FLIP-76: Unaligned Checkpoints





topology (source → map → map → sleepy map → measure map),

each channel was a random shuffle.

The sleepy map slept on average 0, 0.01, and 0.1 ms per record to induce backpressure.

FLIP-76: Unaligned Checkpoints

It provides the following benefits.

- Upstream processes can **continue to produce data**, even if some operators still waits on a checkpoint barrier on a specific input channel.
- Checkpointing **times are heavily reduced** across the execution graph, even for operators with a single input channel.
- End-users will see **more progress** even in unstable environments as more up-to-date checkpoints will avoid too many recomputations.
- Facilitate **faster rescaling**.

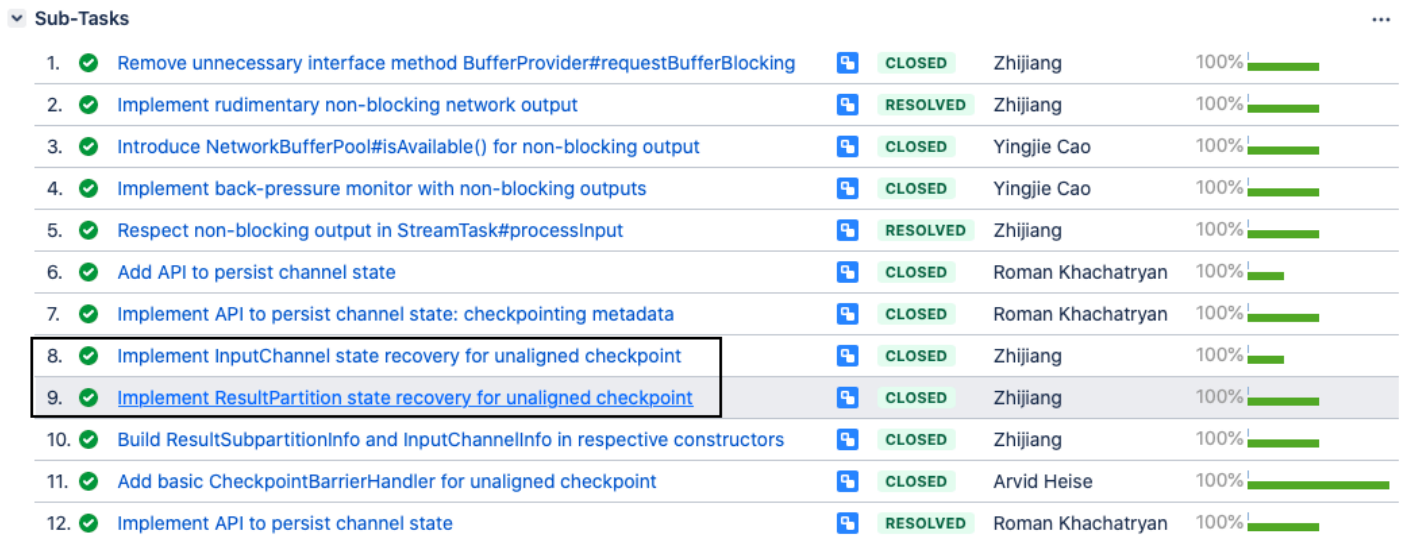
FLIP-76: Unaligned Checkpoints

It has the following known limitations:

- State size increase
- Longer and heavier recovery depending on the increased state size
- Cannot rescale or change the job graph with unaligned checkpoints.
- Currently does not support concurrent unaligned checkpoints.
- Break with an implicit guarantee in respect to watermarks during recovery.

How can unaligned checkpoint influence us?

- Flink would use unaligned checkpoint
  - support or not
  - improve our implementation based on it
- In FLIP-76, it proposes how to handle state recovery on InputChannel and ResultPartition upon rescale.
  - read the FLIP
  - read their code

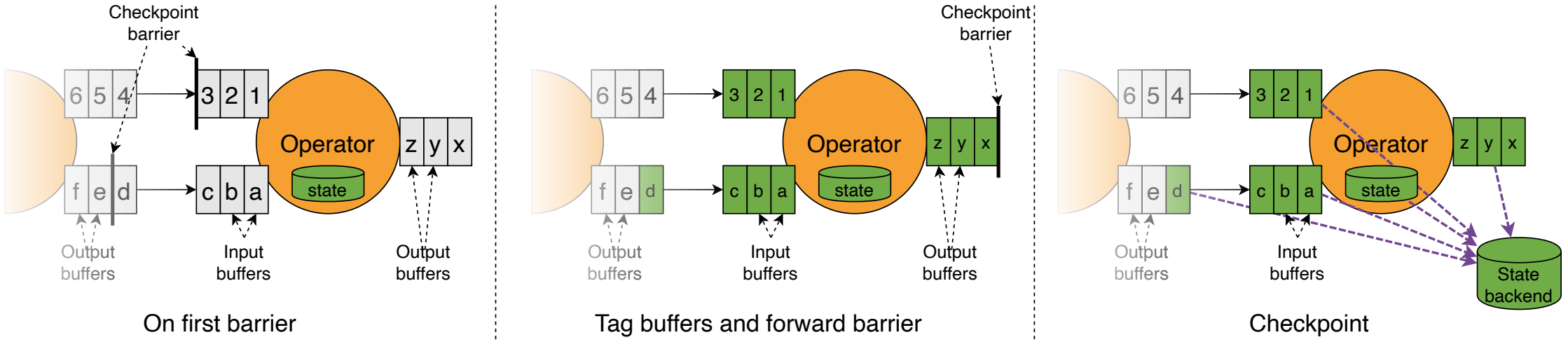


Naive Idea

With unaligned checkpoints, the in-flight data and the operator state are stored together.

We may do actions (rescale or repartition) before real processing the in-flight (tagged) data.

In other words, the actions could influence the processing of these data immediately.



Reference

1. State Management in Apache Flink
2. Flink 1.11 Documentation
3. FLIP-76: Unaligned Checkpointing
4. Mailing List about FLIP-76
5. [FLINK-14551] Unaligned checkpoints issue