“Hit me, baby, just one time”

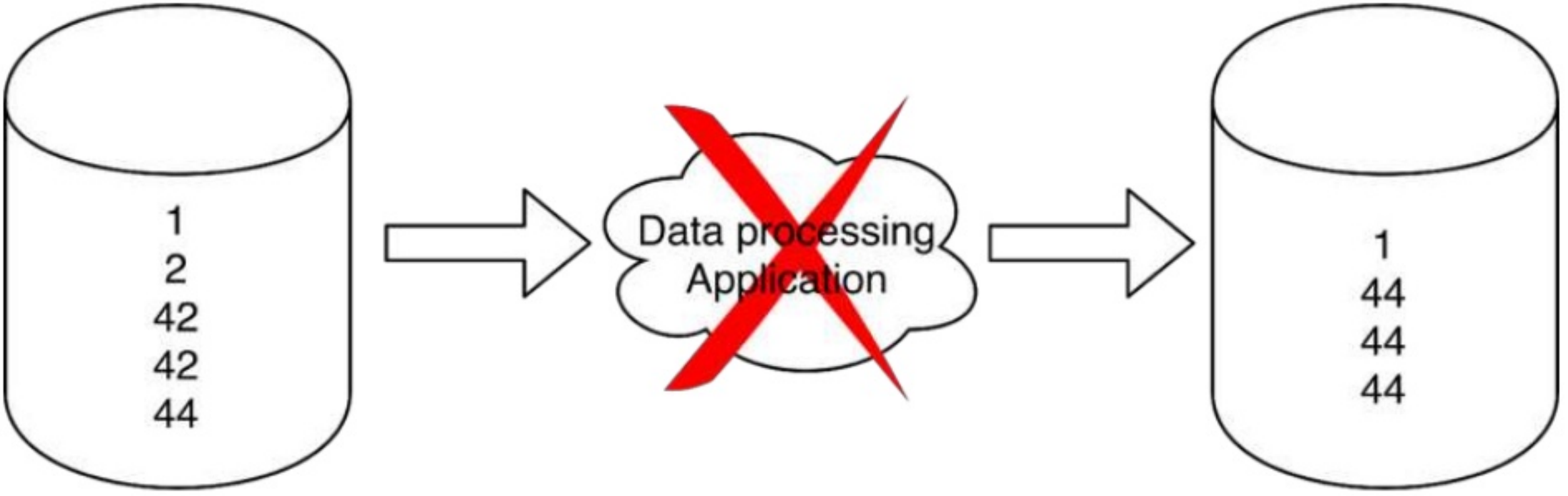
Building End-to-End Exactly-Once Applications with Flink

Piotr Nowojski
piotr@data-artisans.com
@PiotrNowojski

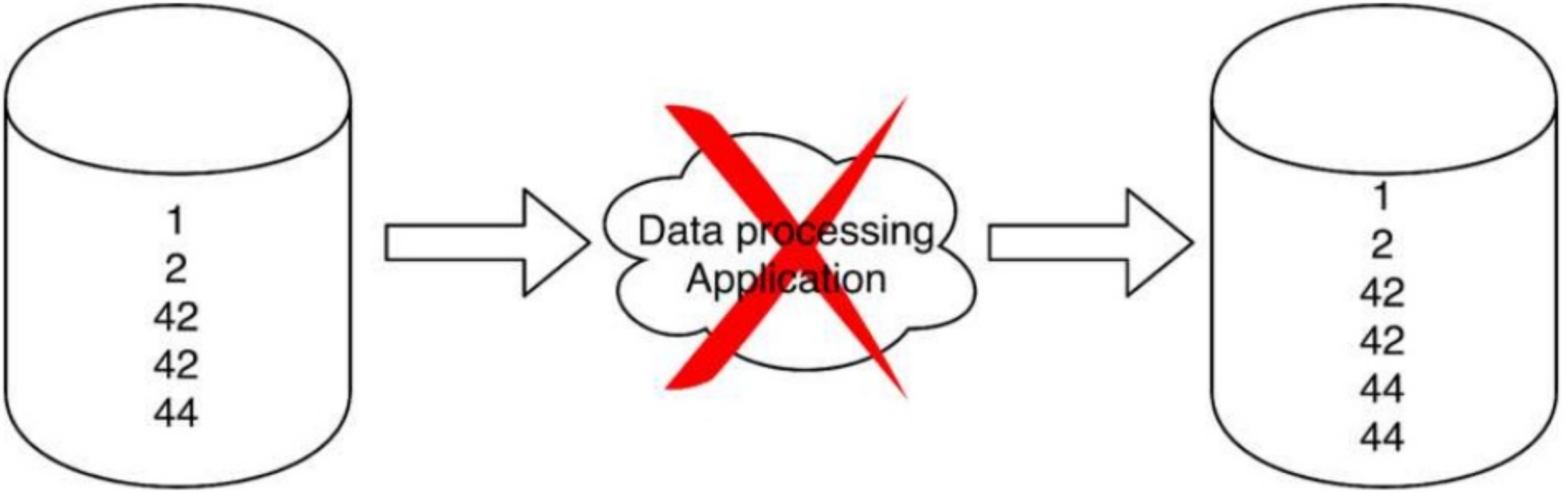
dataArtisans

Overview of delivering guarantees

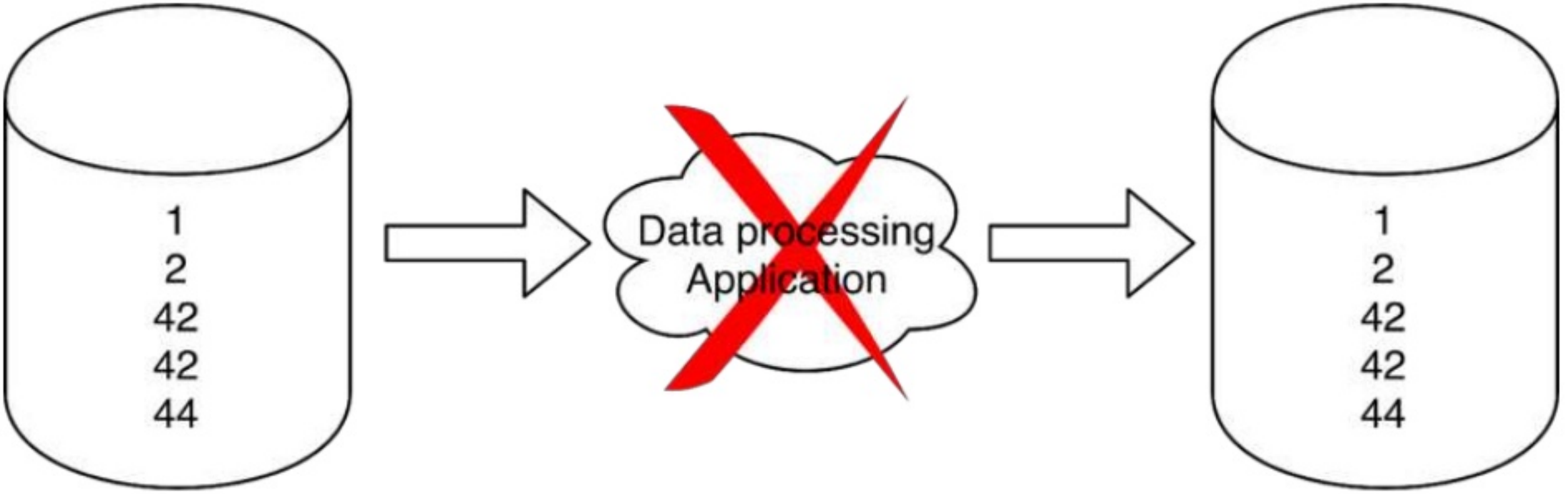
No guarantees and failures



At-Least-Once and failures



Exactly-Once and failures



How to achieve Exactly-Once

Exactly-Once on a single node

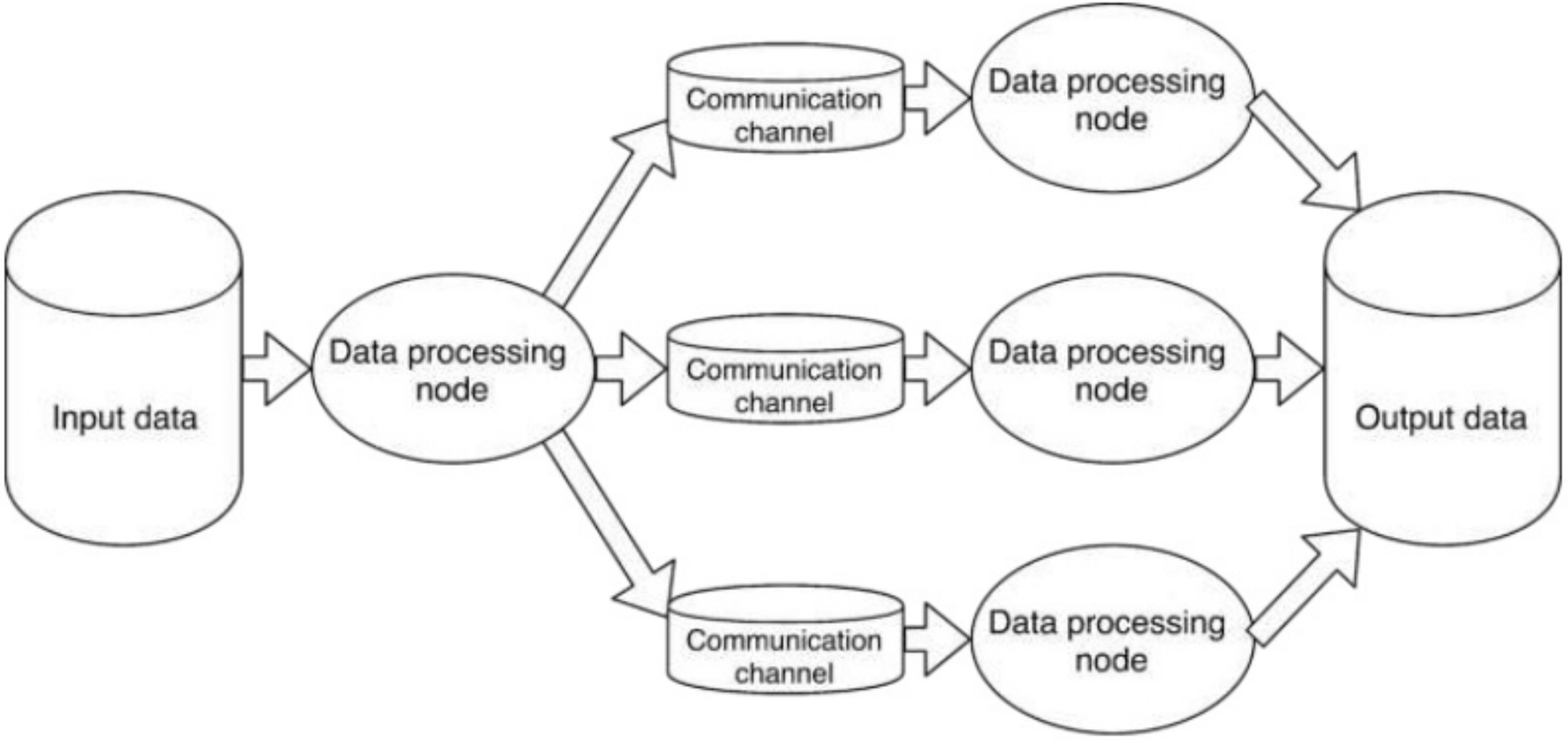
- Simple transactions
  - Write data in transaction
  - Include read offsets in transaction
- On success - commit transaction
- On failure - abort transaction and restart from last committed read offset

Exactly-Once on a single node

- What about application state?
  - For example running average
- Include in the transaction, by adding as a separate file/table just before commit

Exactly-Once in a distributed system

- Persistent communication channels



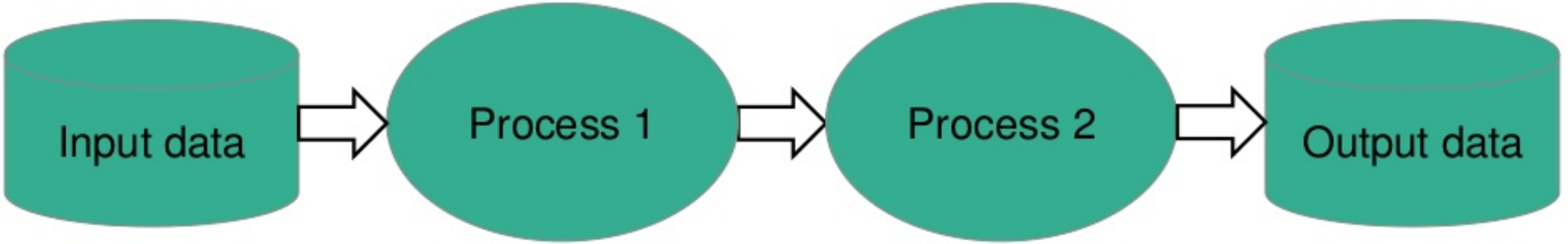
Exactly-Once in a distributed system

- Persistent communication channels
  - Allow to re-process messages from last committed read offsets
  - High costs of communication
  - Processes operate independent of each other

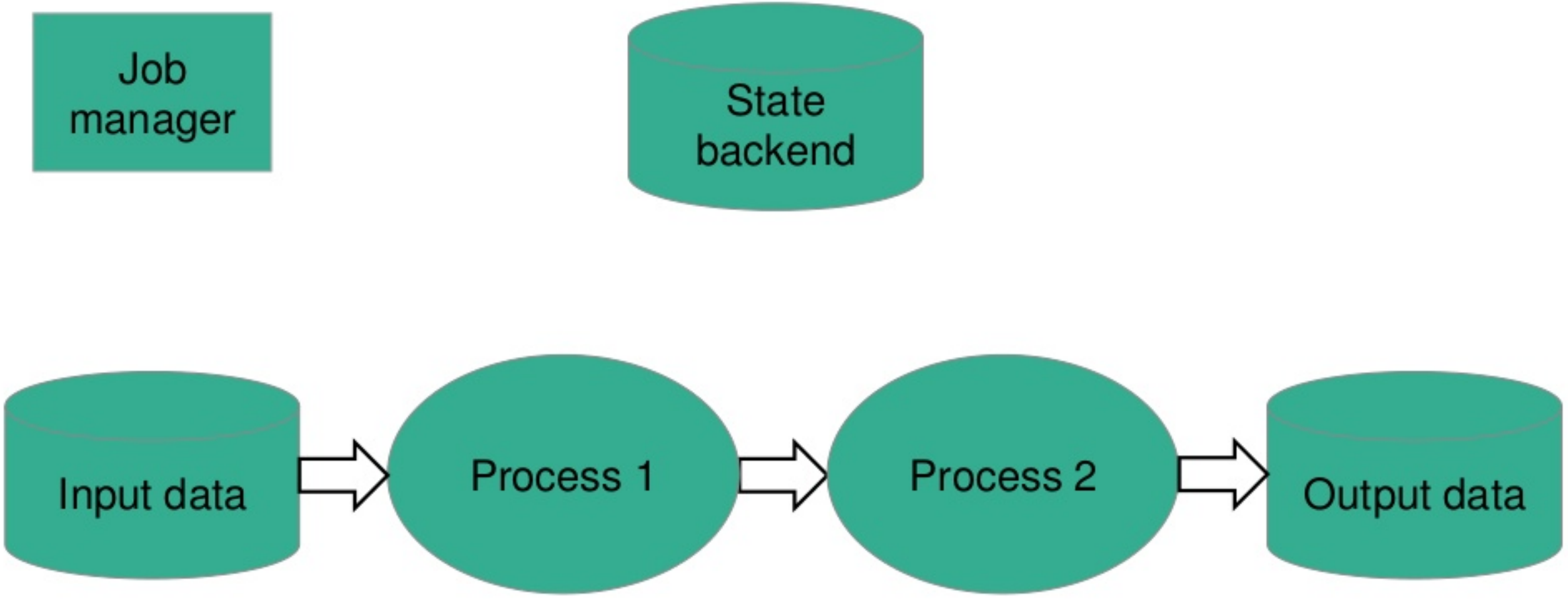
Exactly-Once in Flink

Exactly-Once two phase commit

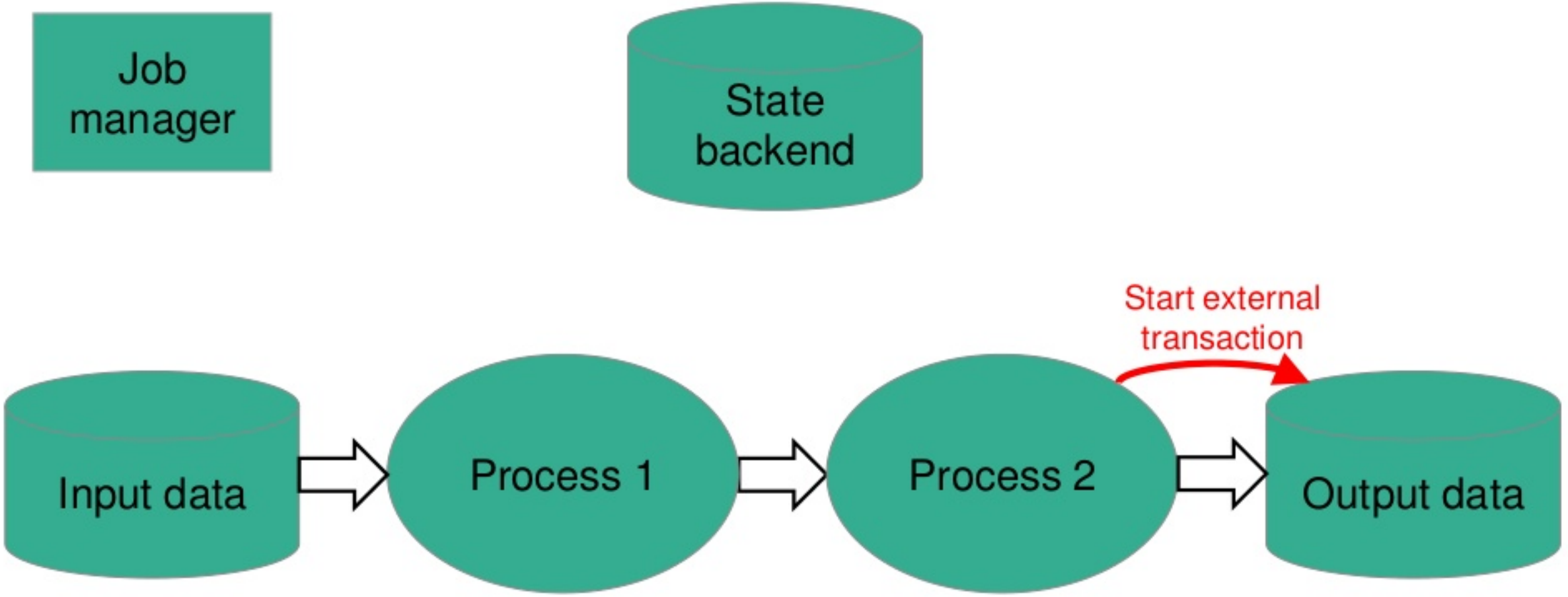
- Can we do better? Yes! Two phase commit protocol



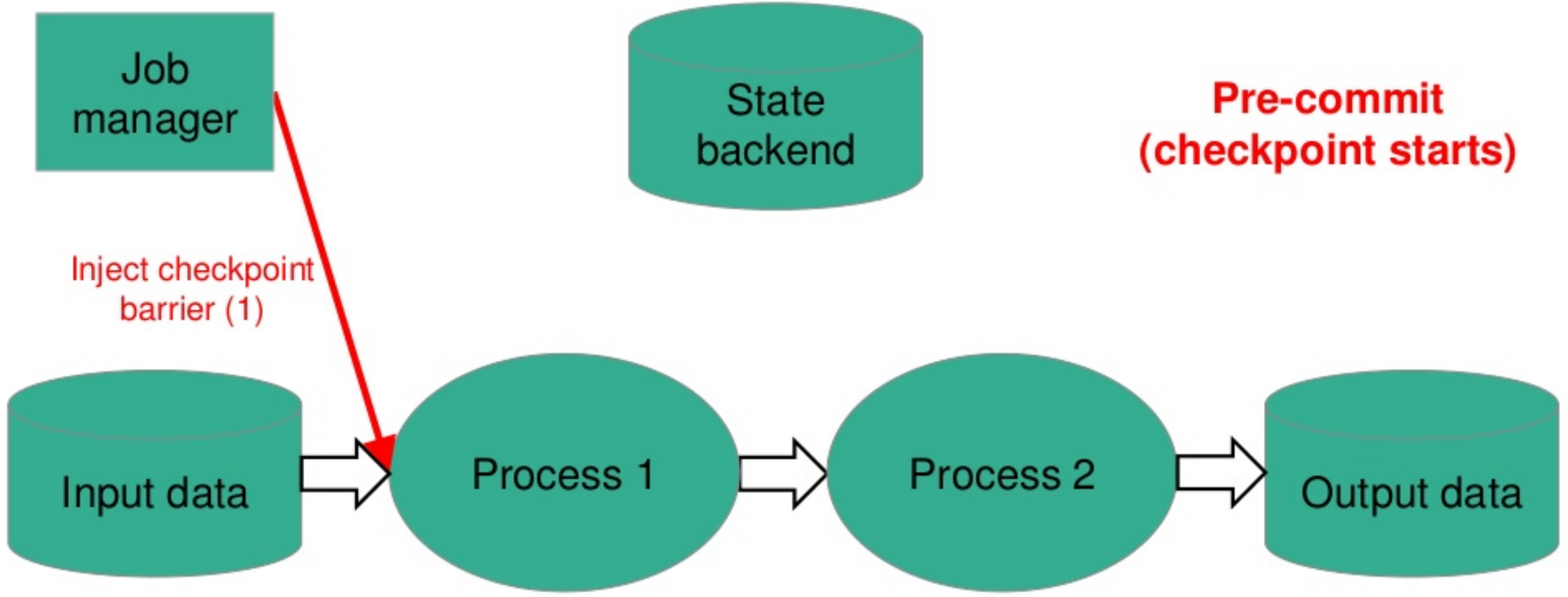
Exactly-Once two phase commit



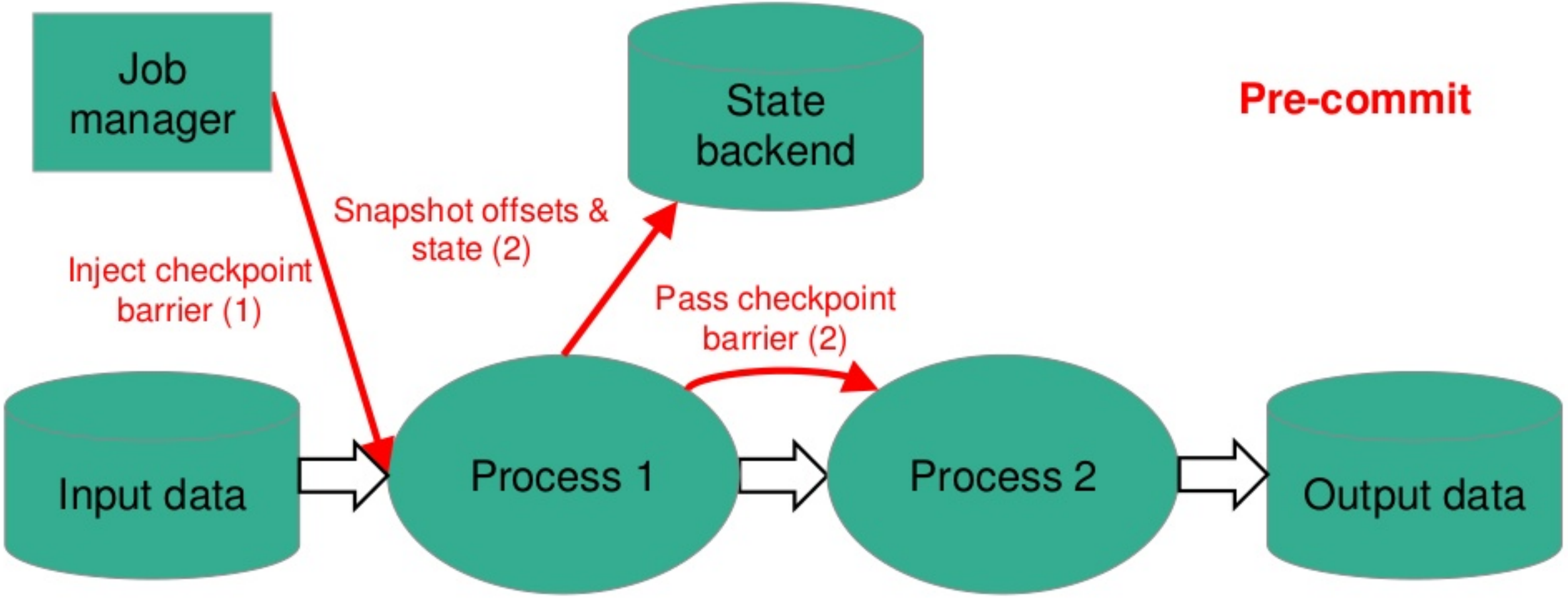
Exactly-Once two phase commit



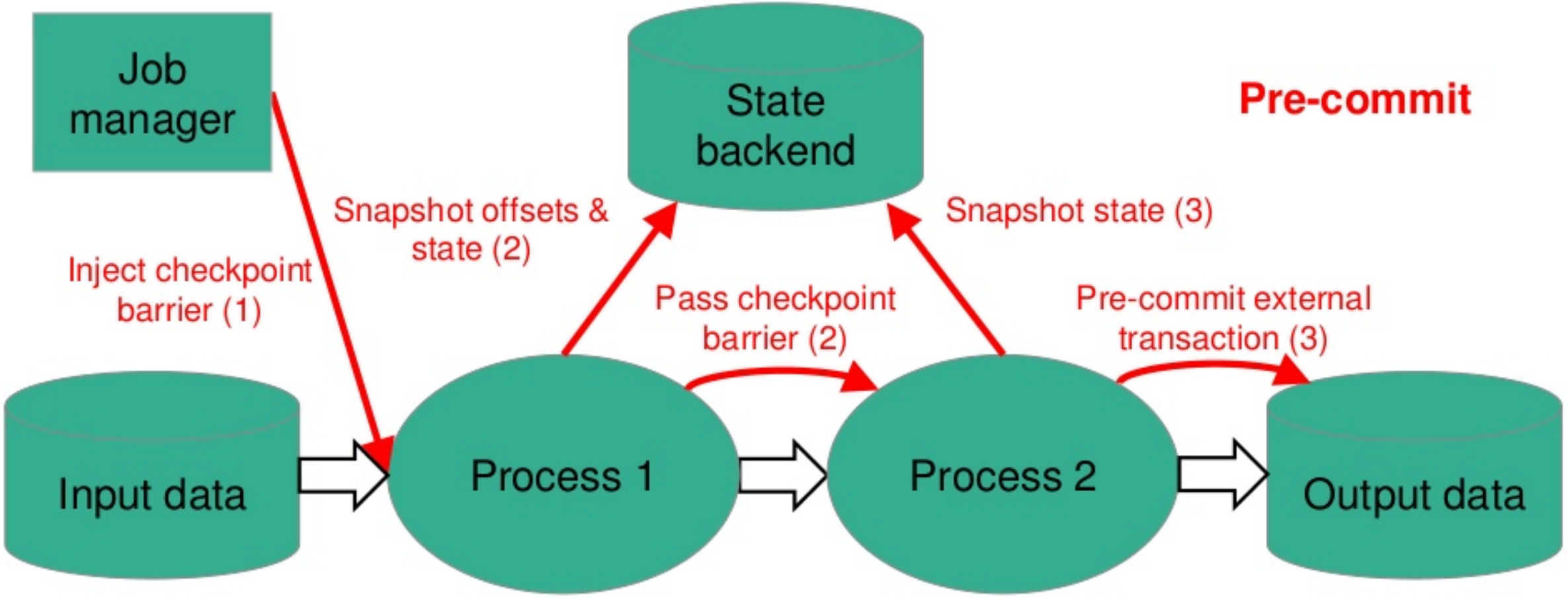
Exactly-Once two phase commit



Exactly-Once two phase commit



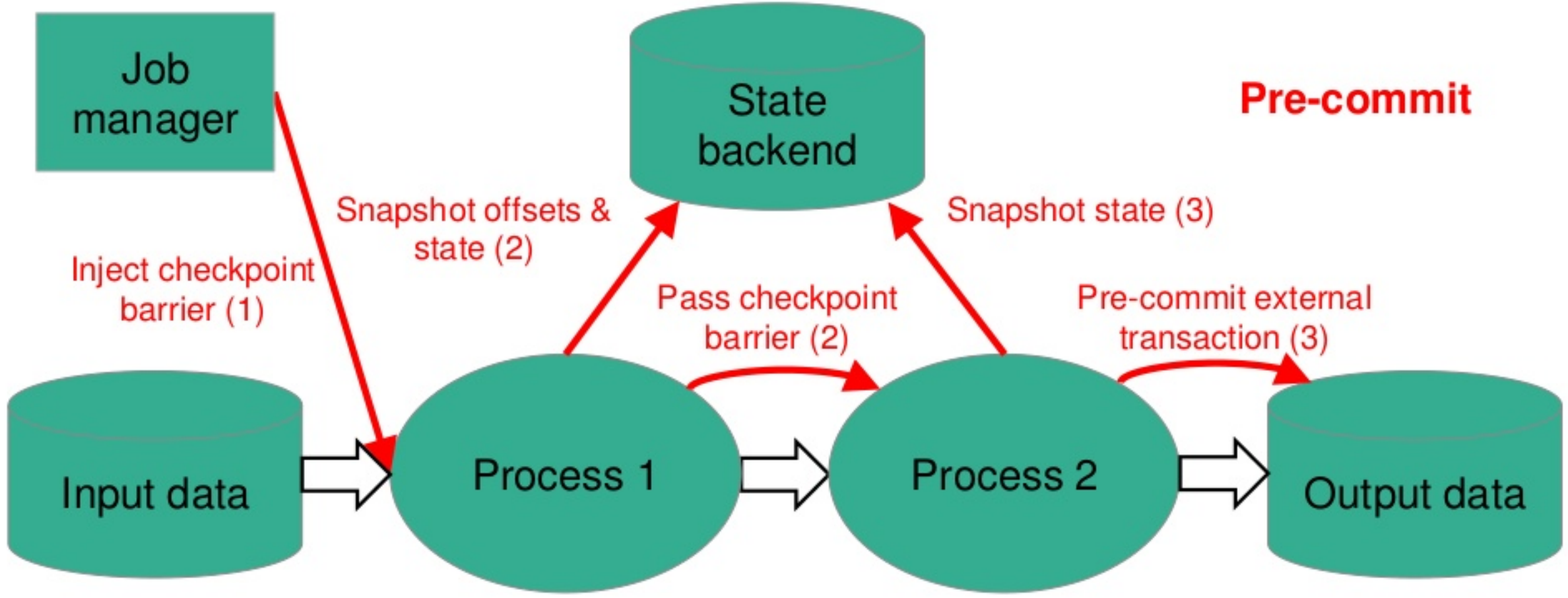
Exactly-Once two phase commit



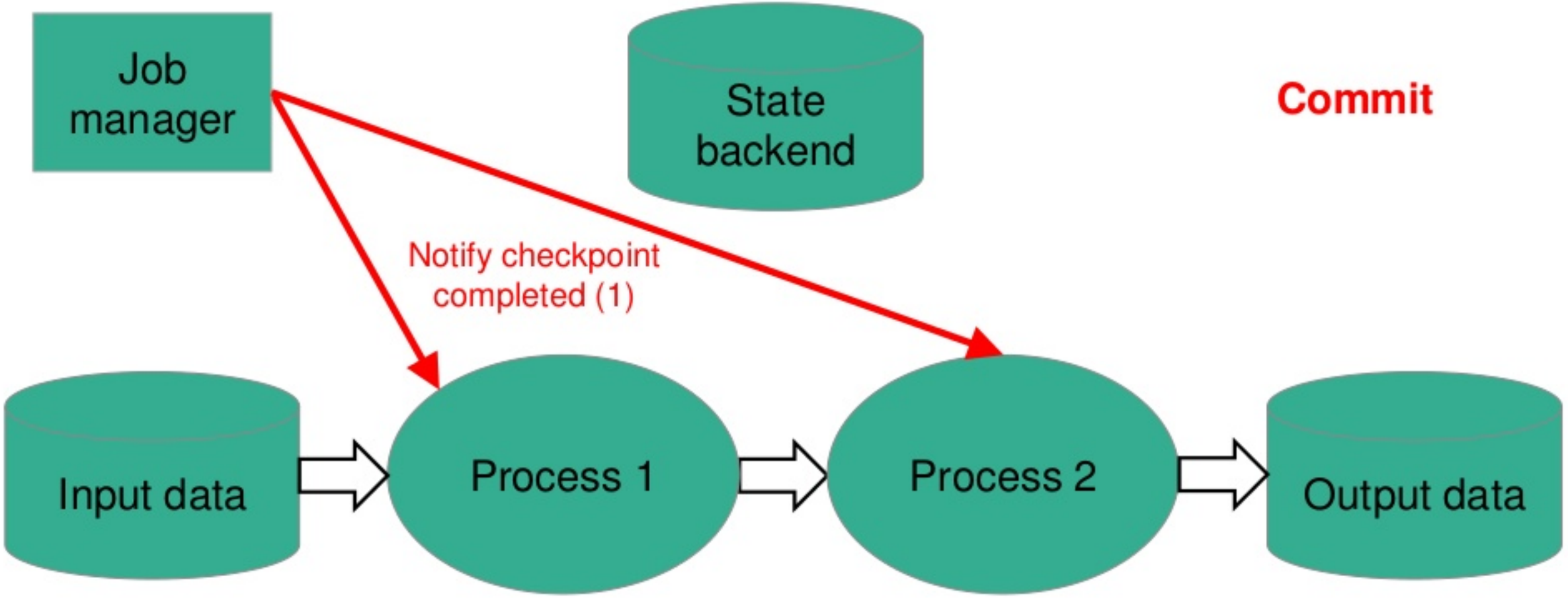
Exactly-Once two phase commit

- Without side effects (only internal state)
  - Example: calculating running average
  - State managed by Flink
- With side effects
  - Operator must manage external state on it’s own

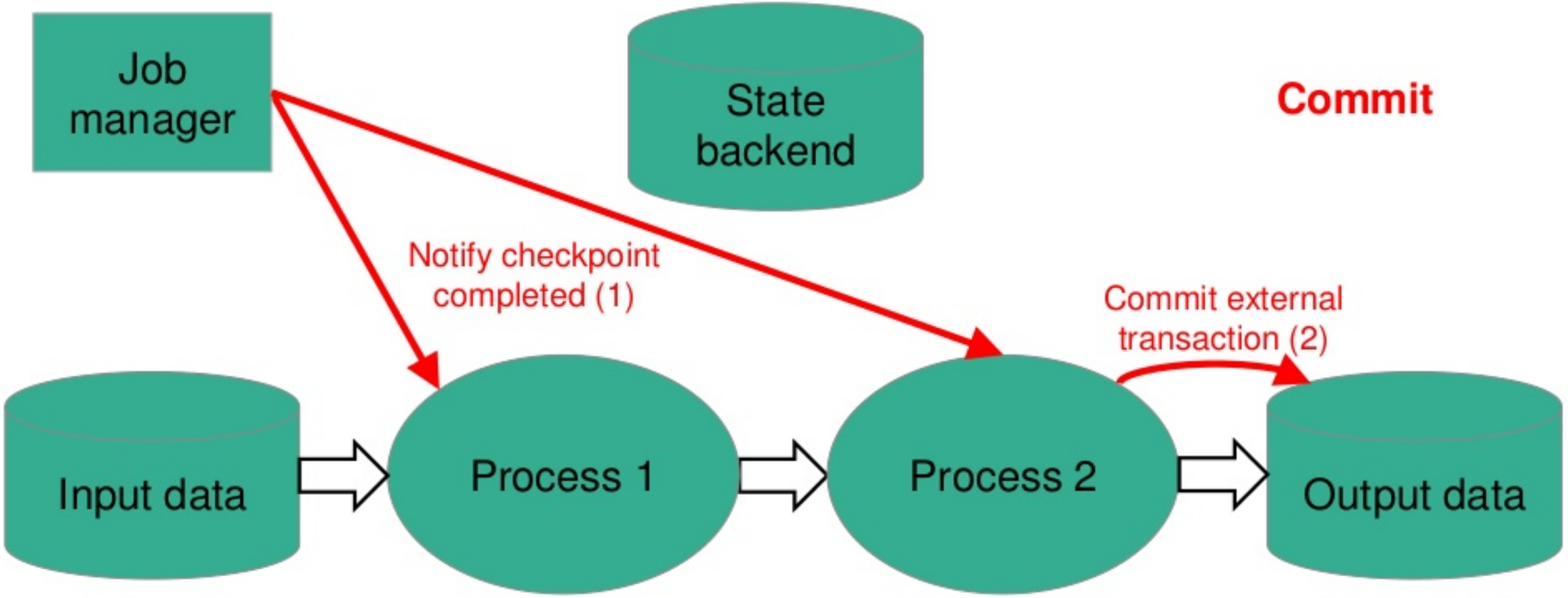
Exactly-Once two phase commit



Exactly-Once two phase commit



Exactly-Once two phase commit



Exactly-Once two phase commit

- Once all processes successfully complete pre-commit, commit is issued
- If at least one pre-commit fails others are aborted
- After a successful pre-commit, commit **MUST** be guaranteed to eventually succeed
- This guarantees that all processes agree on the final outcome

Two phase commit example implementation

- **Begin transaction** - create a temporary file
- **Write element** - write data to that file
- **Pre-commit** - flush the file
  - And begin new transaction for subsequent writes
- **Commit** - move the file to a target directory
  - Can increase latency

Exactly-Once two phase commit

- In case of system crash, we can restore state of the application to the latest snapshot
- We could/should abort previous “pending” transactions
  - Delete temporary files

TwoPhaseCommitSinkFunction

- Flink's snapshots/checkpoints are very similar to two phase commit
- There is a *TwoPhaseCommitSinkFunction* which maps checkpoint calls to beginTransaction, preCommit, commit and abort calls

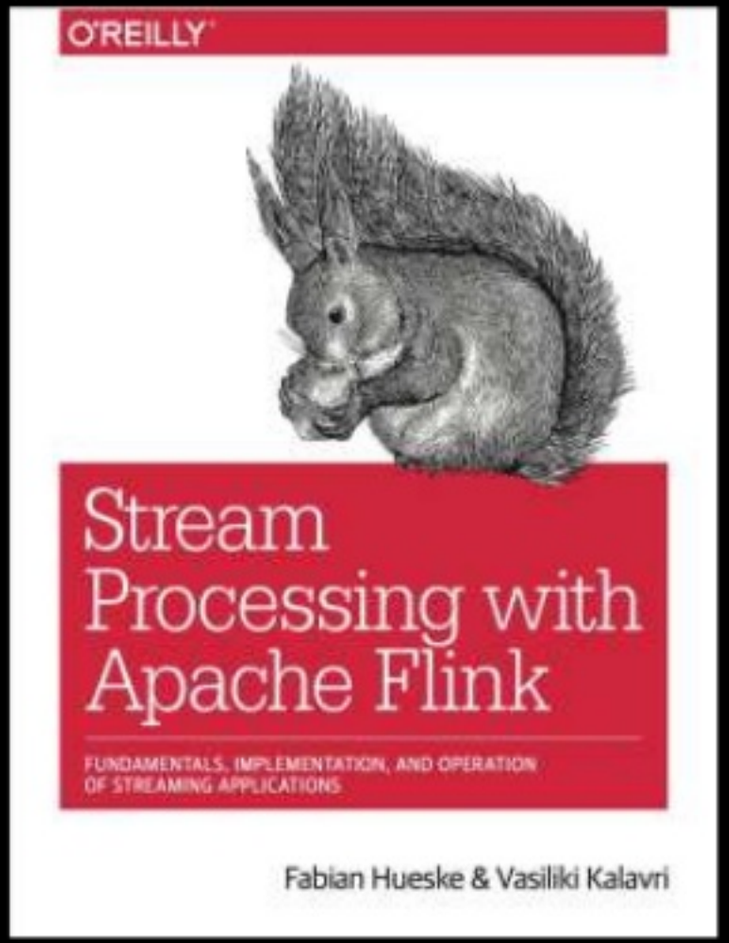
Exactly-Once producers in Flink

- BucketingSink
- Pravega connector
- Kafka 0.11 connector
  - Kafka 0.11 introduced transactions
  - Implemented on top of the *TwoPhaseCommitSinkFunction*
  - Low overhead

Summarizon

- Many ways to achieve Exactly-Once
- Flink checkpointing is similar to the two phase commit protocol
- No materialization of the data in transit
  - Lower latency
  - Higher throughput

Questions?



Thank you!

@PiotrNowojski
@ApacheFlink
@dataArtisans

We are hiring!

data-artisans.com/careers