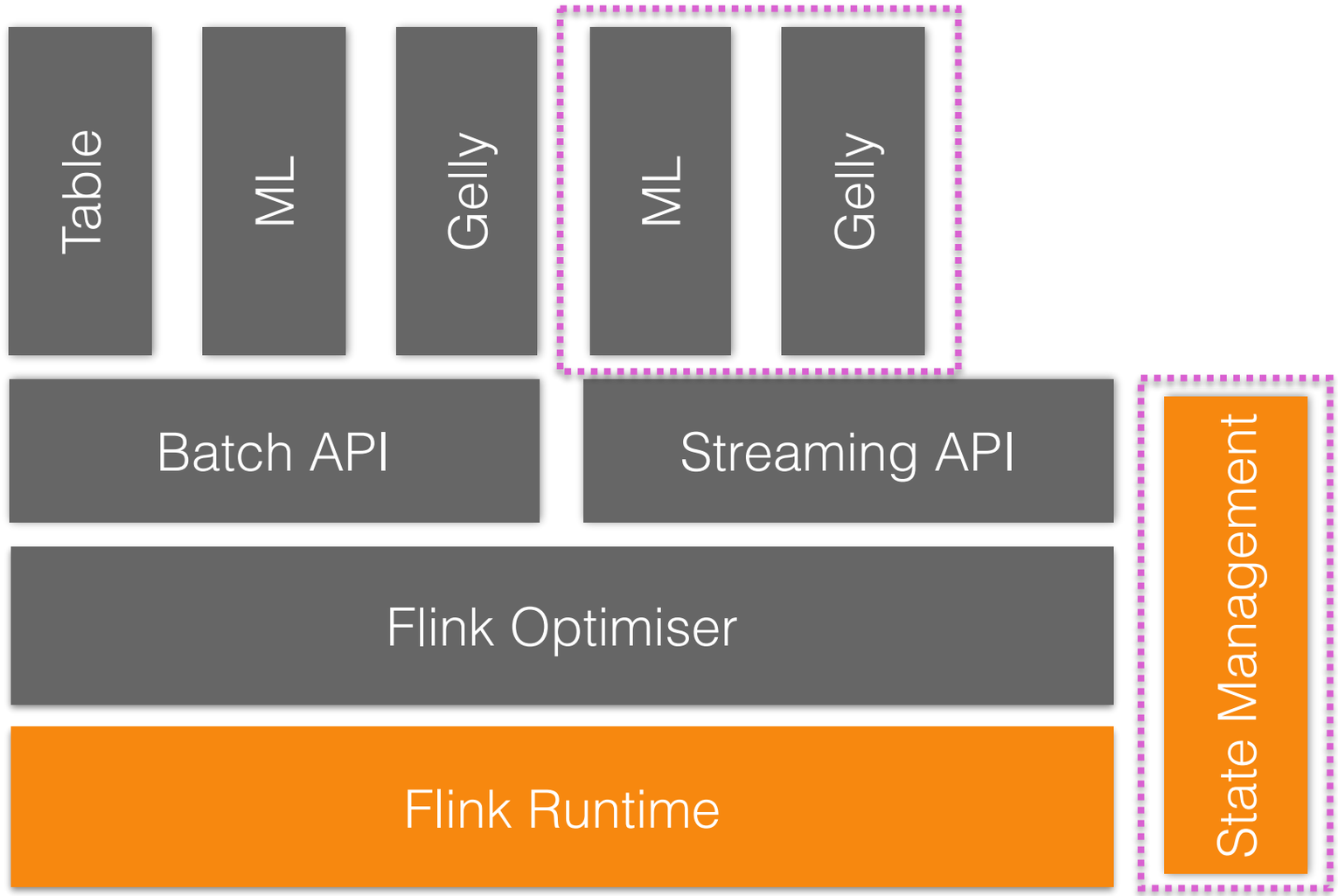
Fault Tolerance in Apache Flink Streaming

Paris Carbone - PhD Candidate
KTH Royal Institute of Technology
<parisc@kth.se, senorcarbone@apache.org>

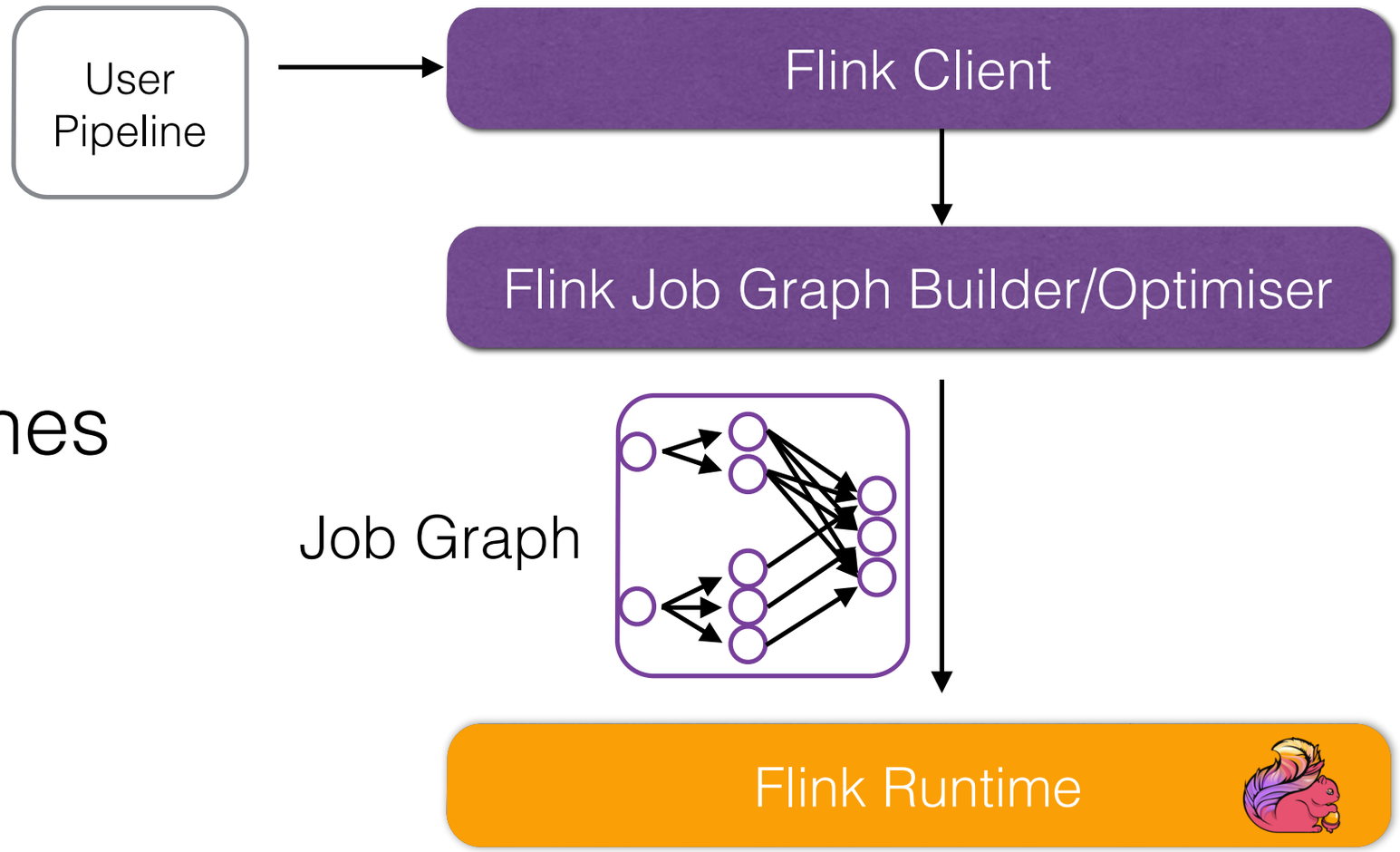
Current Focus



Executing Pipelines

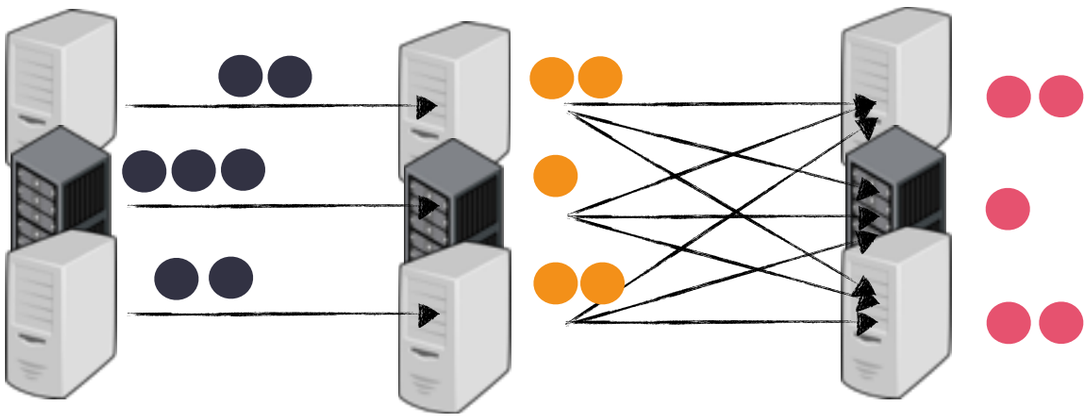
```
val lines: DataStream[String] = env.fromSocketStream(...)
lines.flatMap {line => line.split(" ")
                       .map(word => Word(word,1))}
     .window(Time.of(5,SECONDS)).every(Time.of(1,SECONDS))
     .groupBy("word").sum("frequency")
     .print()
```

- Streaming pipelines translate into **job graphs**

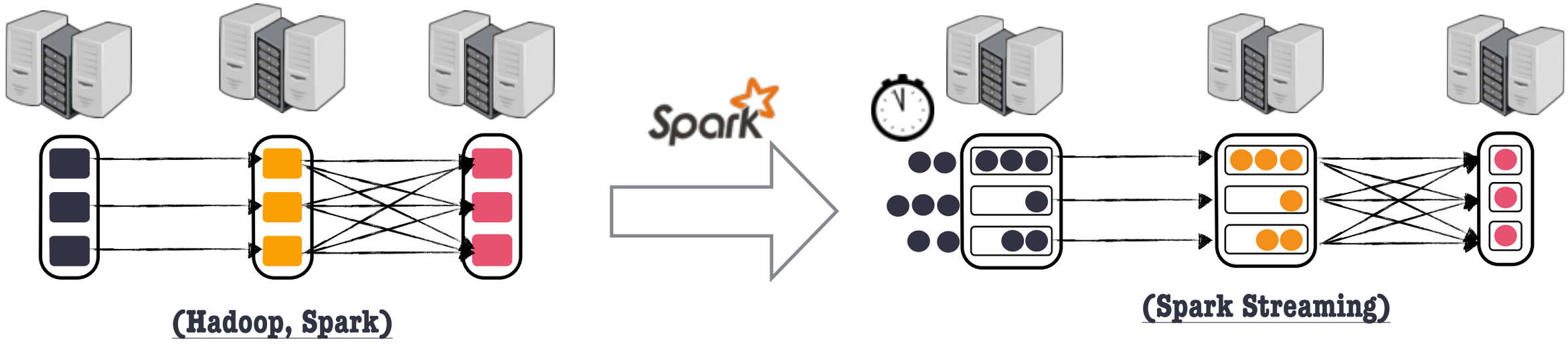


Unbounded Data Processing Architectures

1) Streaming (Distributed Data Flow)



2) Micro-Batch

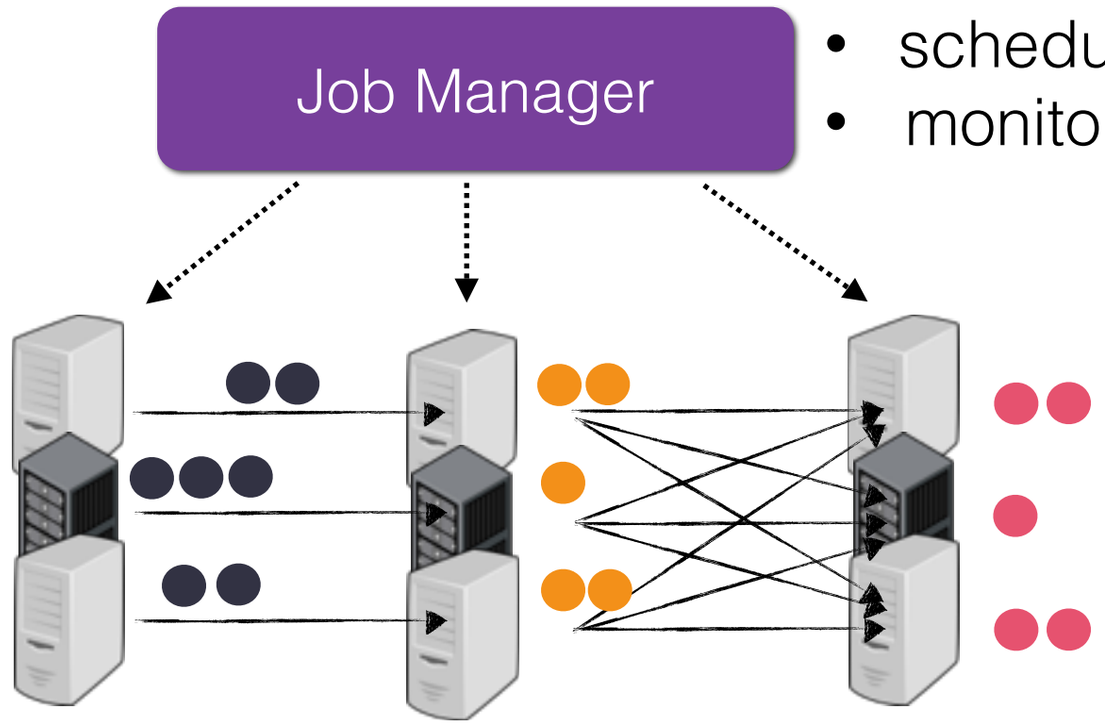


The Flink Runtime Engine

- **Tasks** run operator logic in a pipelined fashion

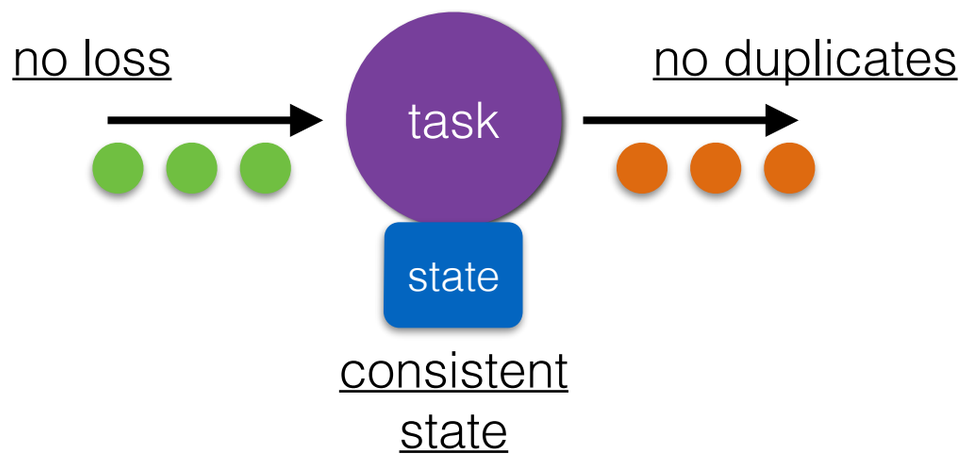
- They are scheduled among workers

- **State** is kept within tasks

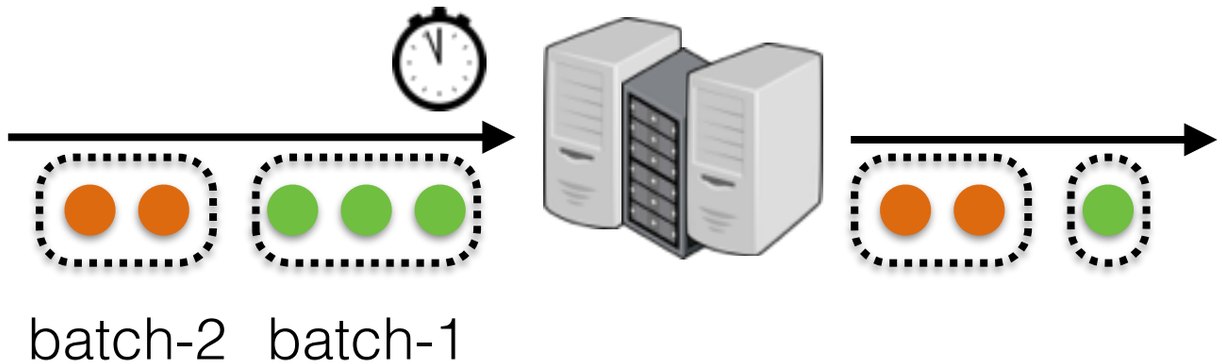


Task Failures

- Task failures are **guaranteed** to occur
- We need to make them **transparent**
- Can we simply recover a task from scratch?



Lessons Learned from Micro-Batch



- If a batch computation fails, simply repeat computation as a transaction
- Transaction rate is **constant**
- Can we apply these principles to a true streaming execution?

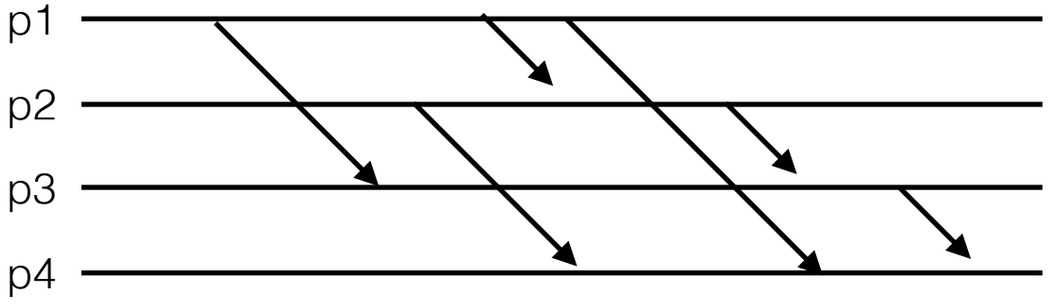
Distributed Snapshots

Distributed Snapshots

“A collection of operator states and records in transit (channels) that reflects a moment at a valid execution”

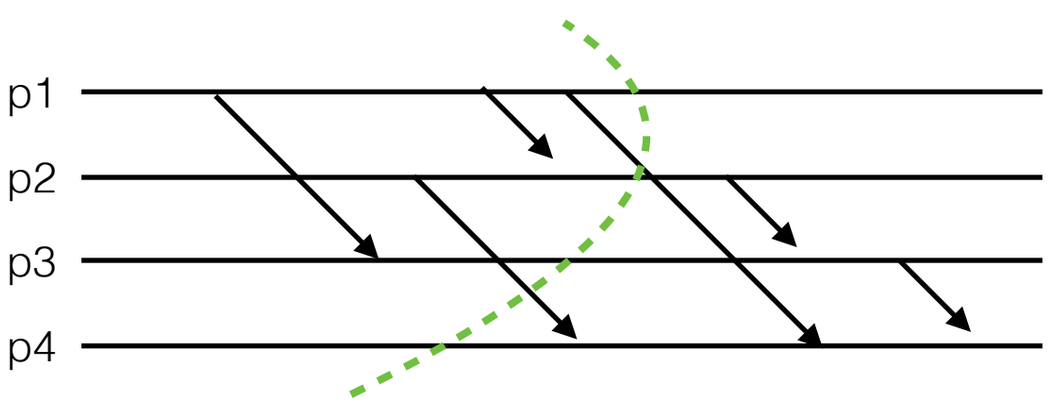
Distributed Snapshots

“A collection of operator states and records in transit (channels) that reflects a moment at a valid execution”



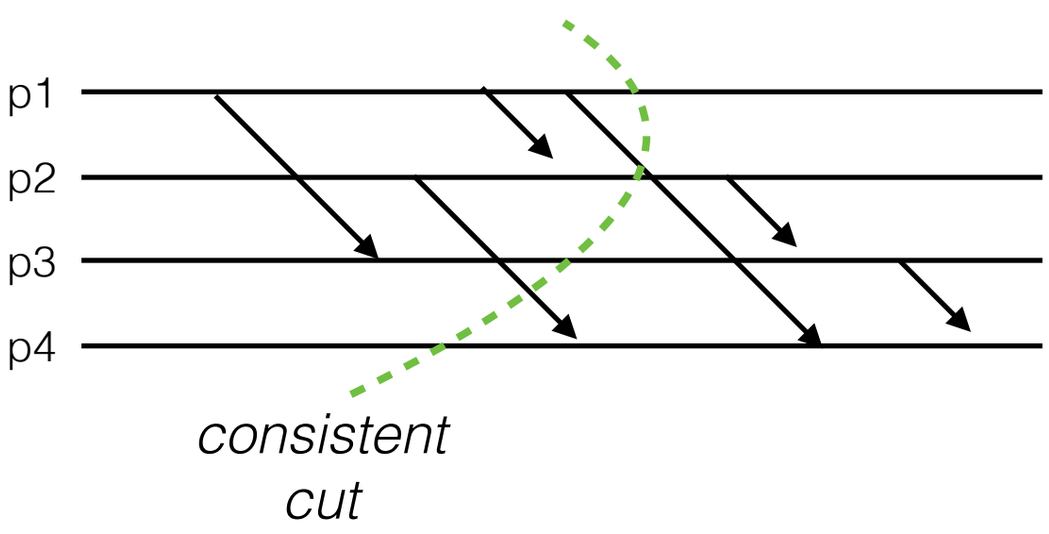
Distributed Snapshots

“A collection of operator states and records in transit (channels) that reflects a moment at a valid execution”



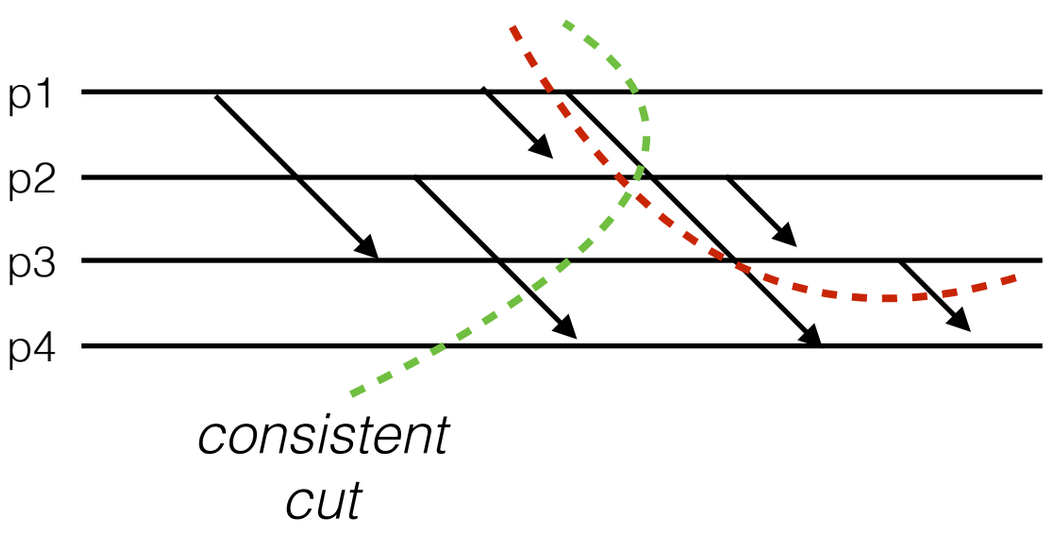
Distributed Snapshots

“A collection of operator states and records in transit (channels) that reflects a moment at a valid execution”



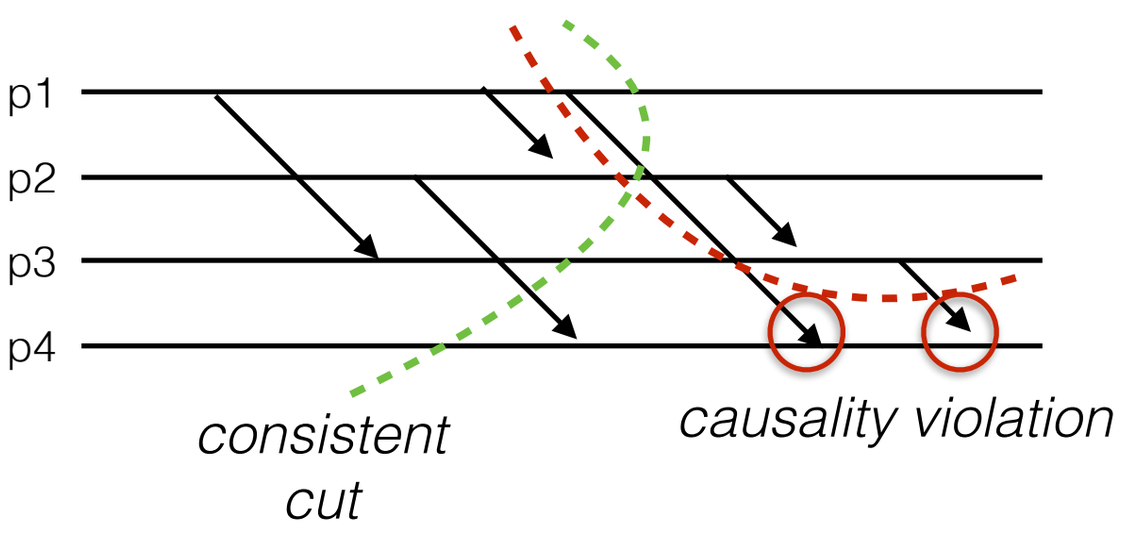
Distributed Snapshots

“A collection of operator states and records in transit (channels) that reflects a moment at a valid execution”



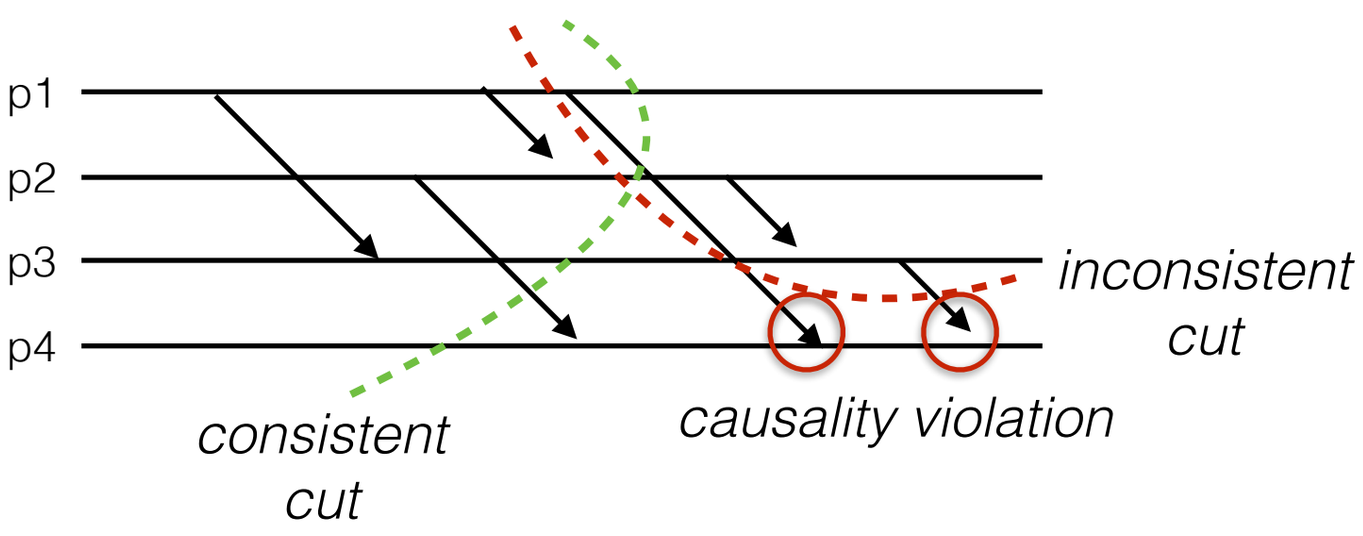
Distributed Snapshots

"A collection of operator states and records in transit (channels) that reflects a moment at a valid execution"



Distributed Snapshots

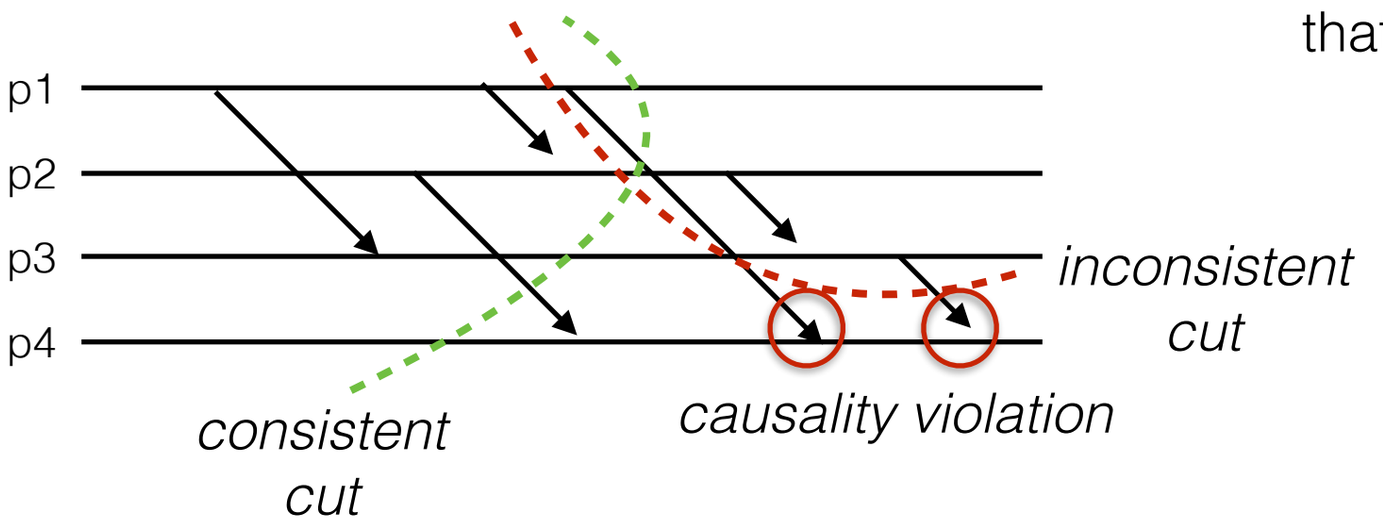
"A collection of operator states and records in transit (channels) that reflects a moment at a valid execution"



Distributed Snapshots

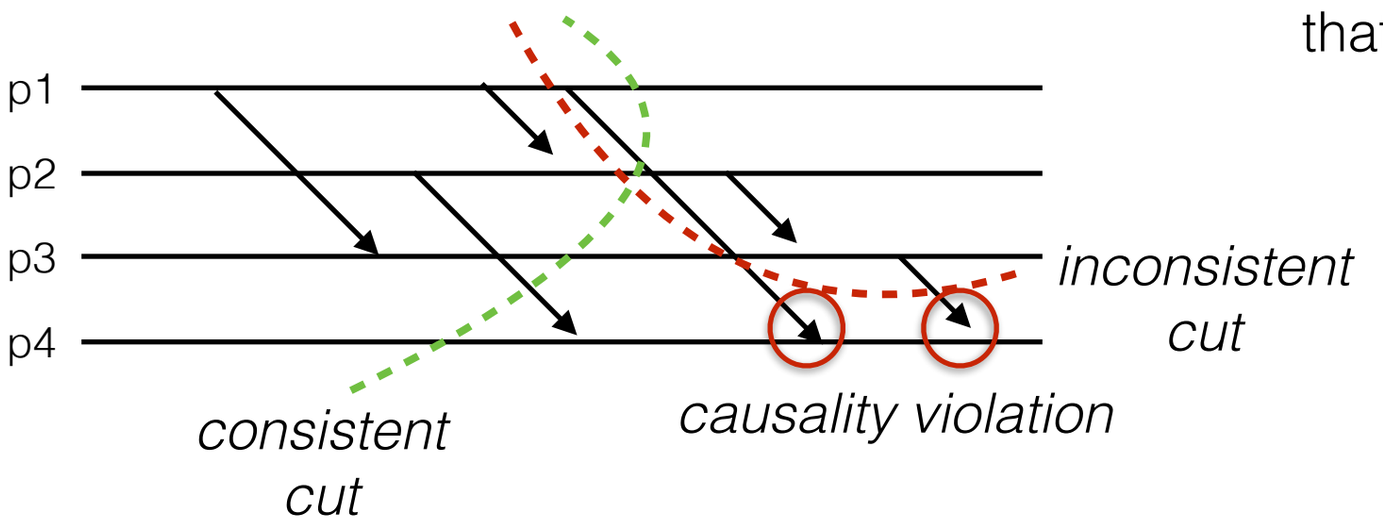
"A collection of operator states and records in transit (channels) that reflects a moment at a valid execution"

Idea: We can resume from a snapshot that defines a consistent cut



Distributed Snapshots

“A collection of operator states and records in transit (channels) that reflects a moment at a valid execution”



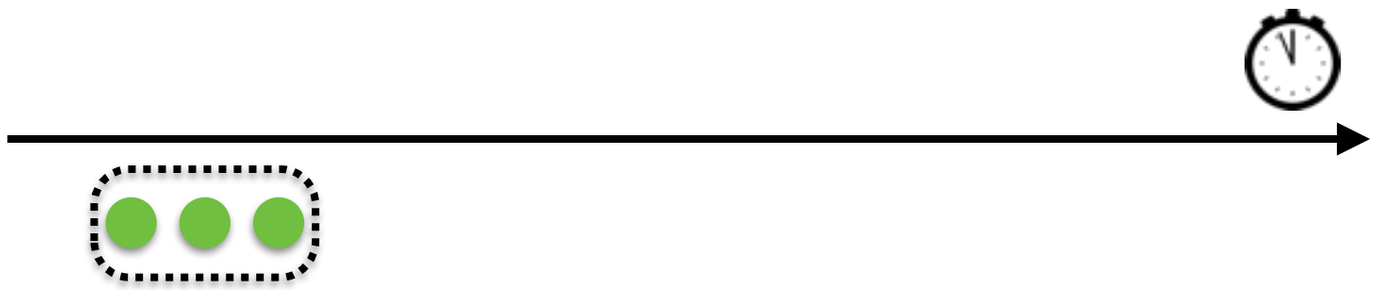
Idea: We can resume from a snapshot that defines a consistent cut

Assumptions

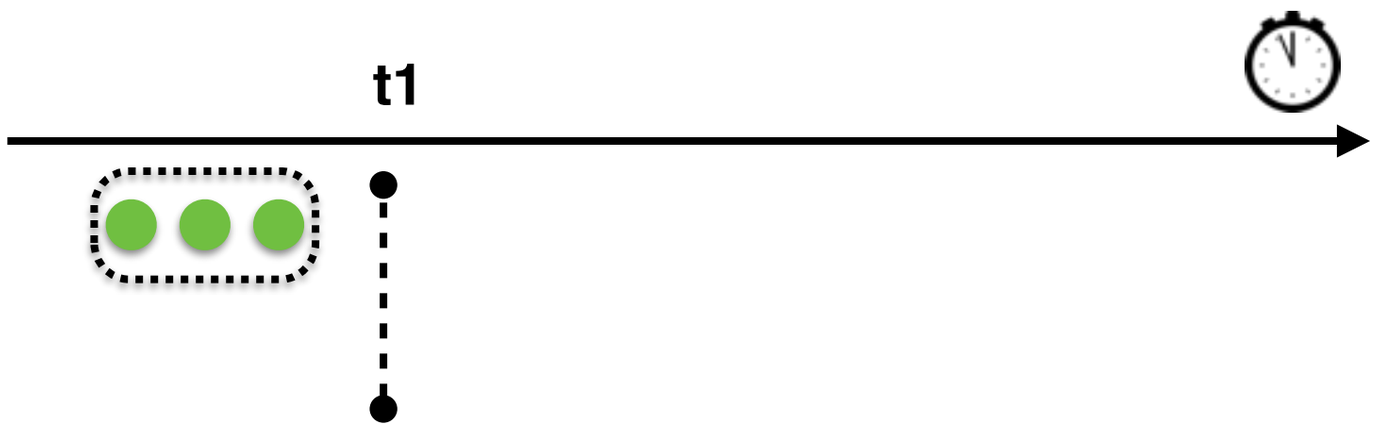
- repeatable sources
- reliable FIFO channels

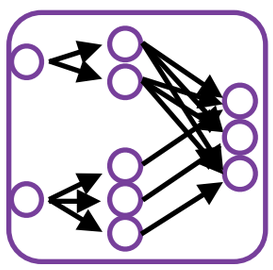
Distributed Snapshots

Distributed Snapshots



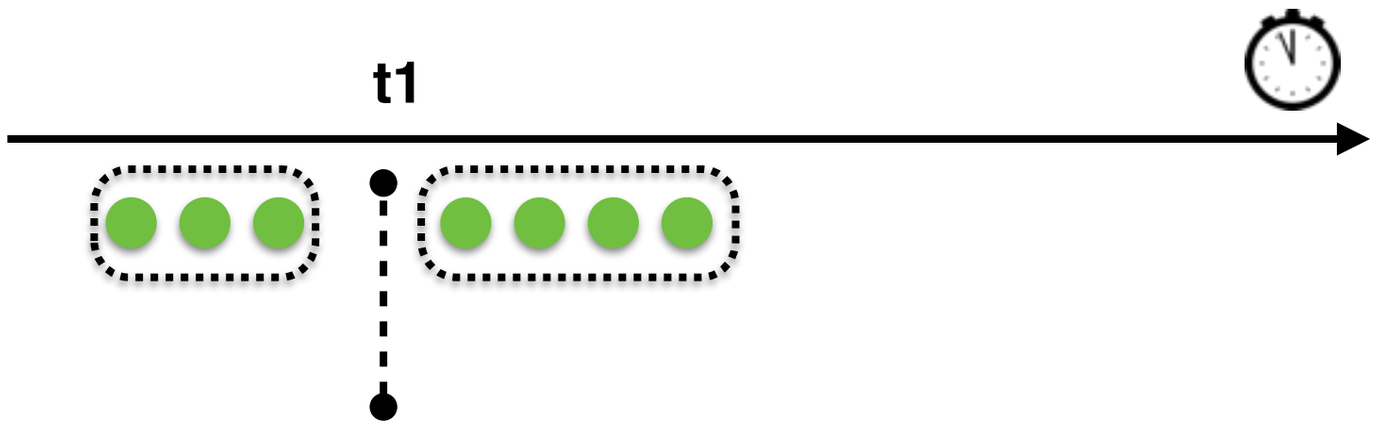
Distributed Snapshots

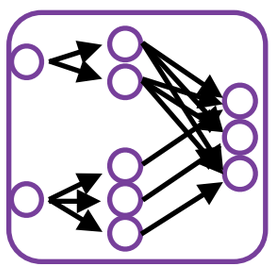




snap - t1

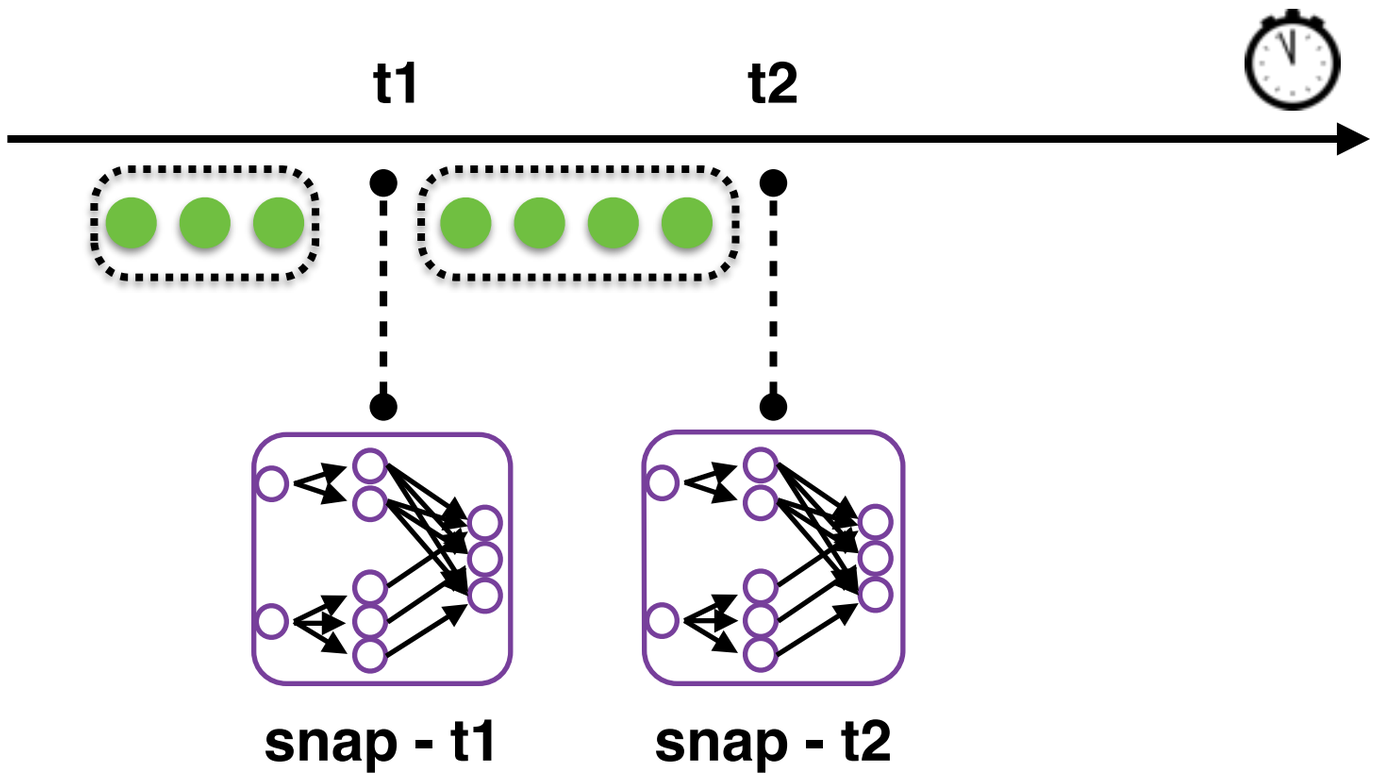
Distributed Snapshots



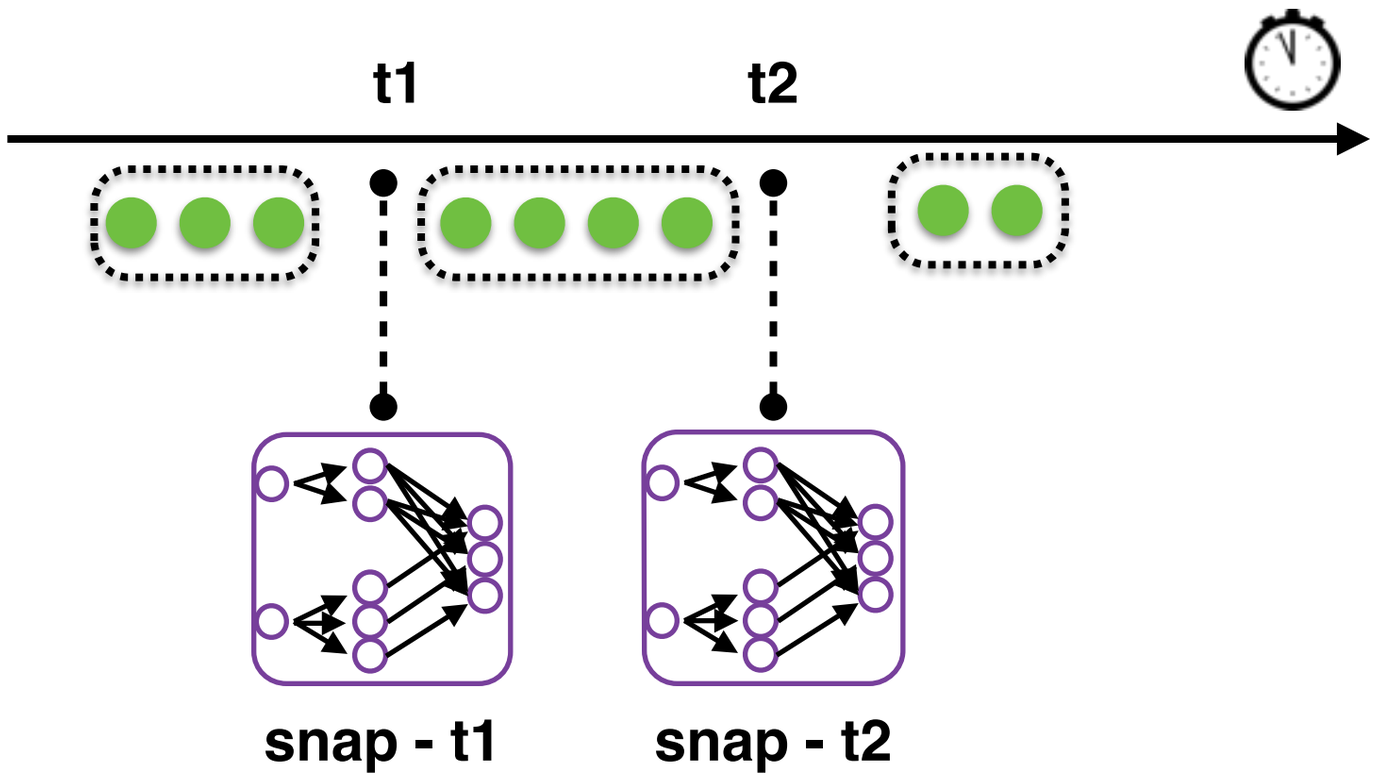


snap - t1

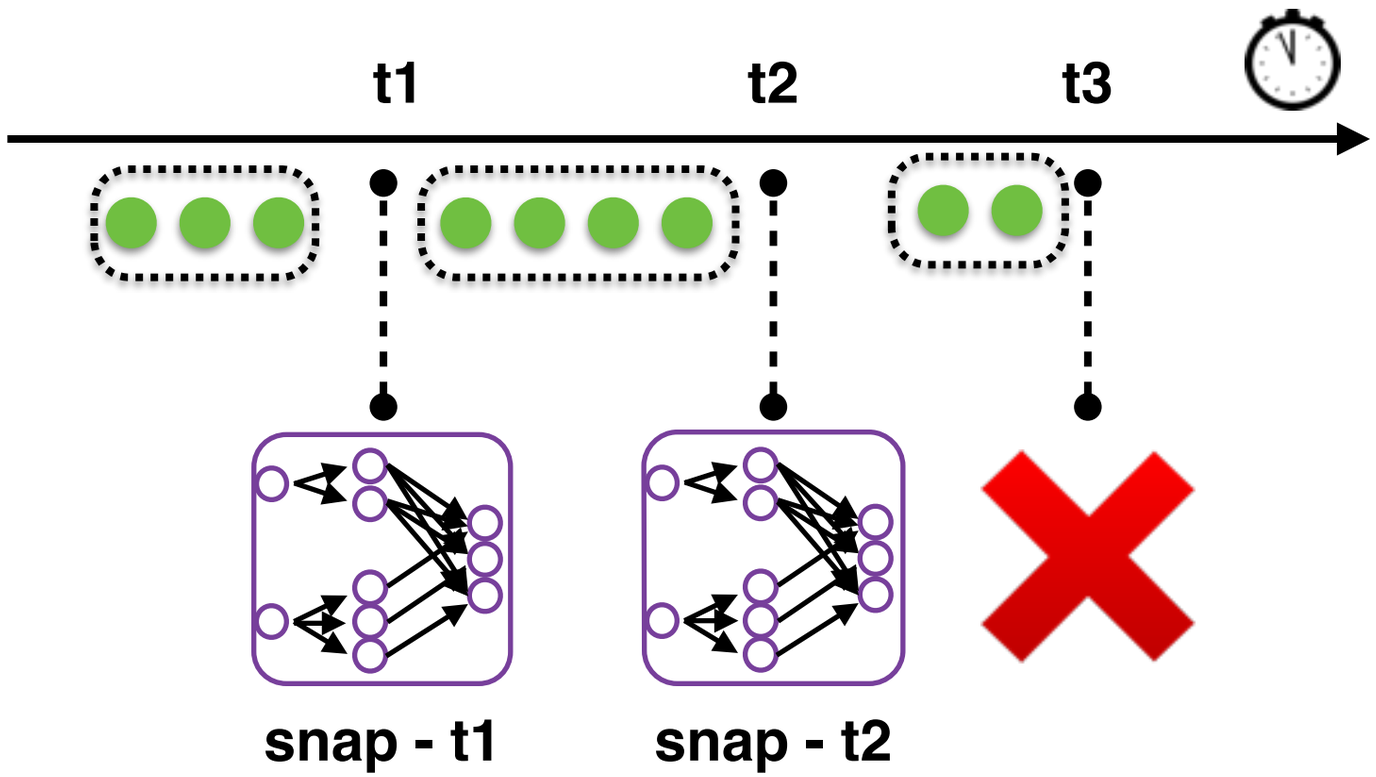
Distributed Snapshots



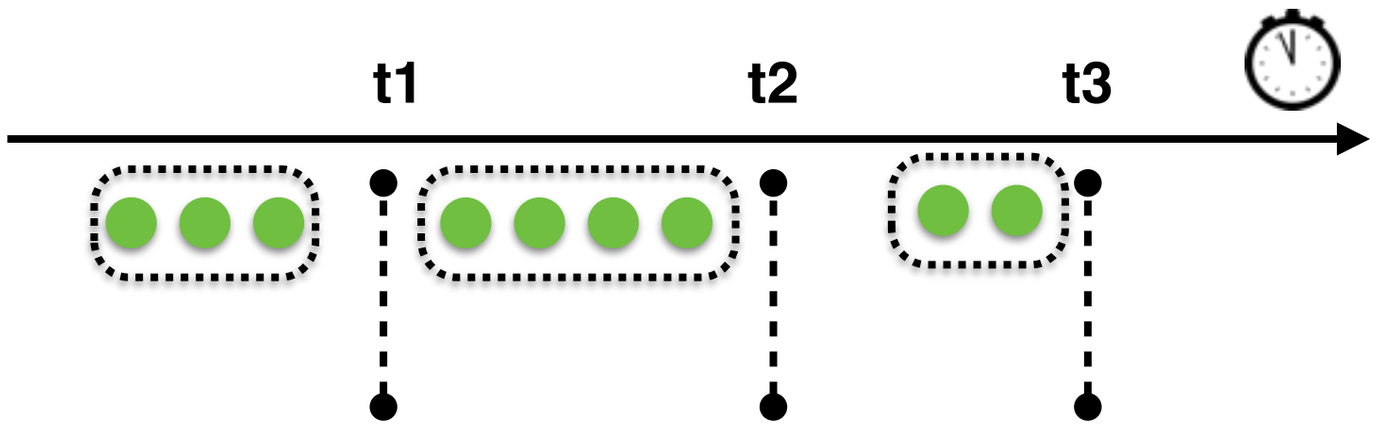
Distributed Snapshots

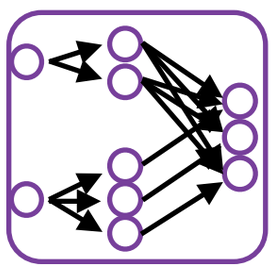


Distributed Snapshots

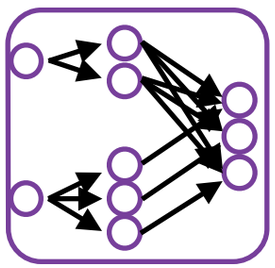


Distributed Snapshots





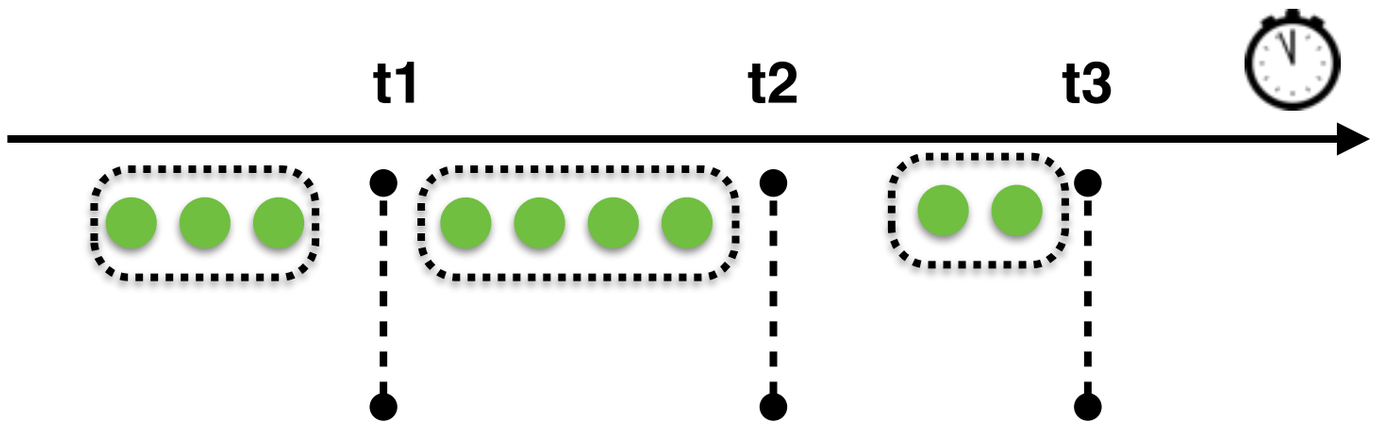
snap - t1

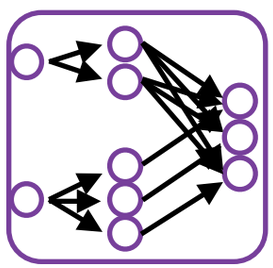


snap - t2

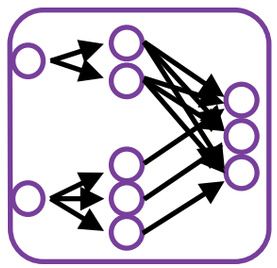
reset from snap t2

Distributed Snapshots





snap - t1



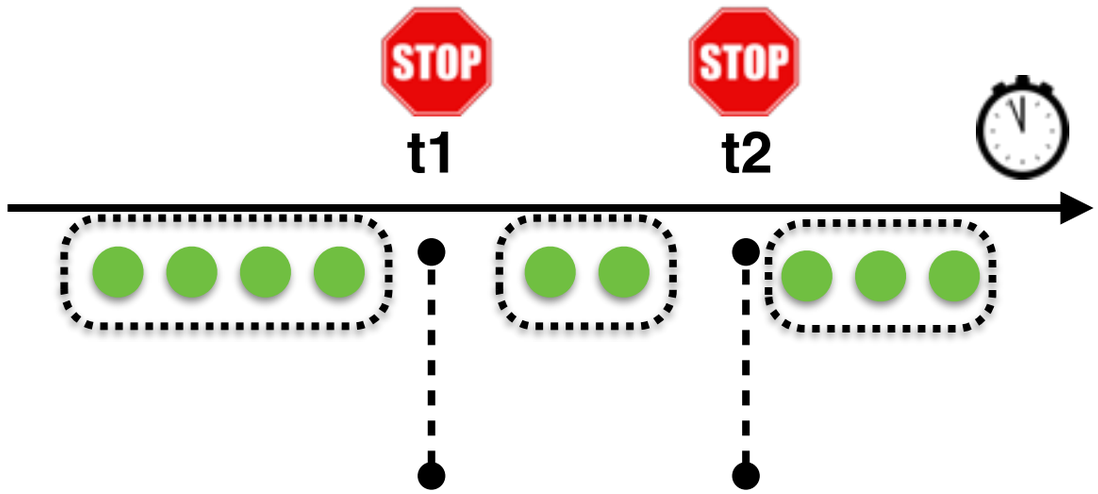
snap - t2

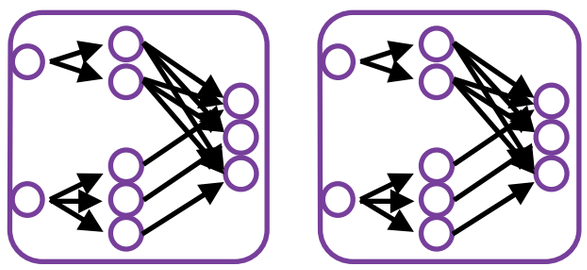
reset from snap t2

Assumptions

- repeatable sources
- reliable FIFO channels

Taking Snapshots





execution snapshots

Initial approach (see Naiad)

- Pause execution on t1,t2,..
- Collect state
- Restore execution

Lamport On the Rescue

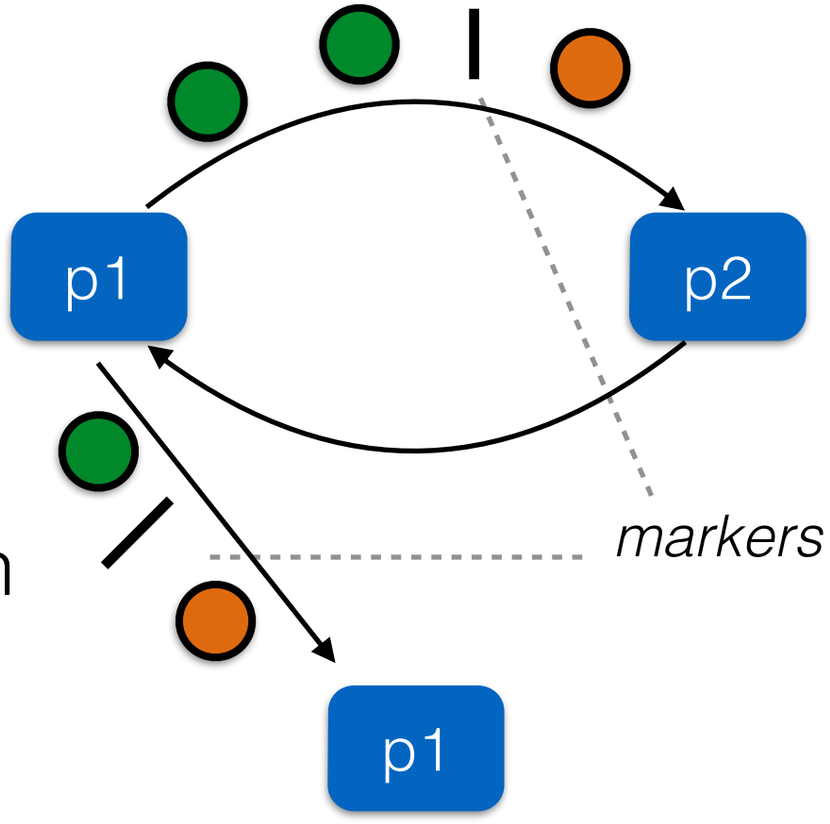
“The global-state-detection algorithm is to be superimposed on the underlying computation: it must run concurrently with, but not alter, this underlying computation”

Chandy-Lamport Snapshots

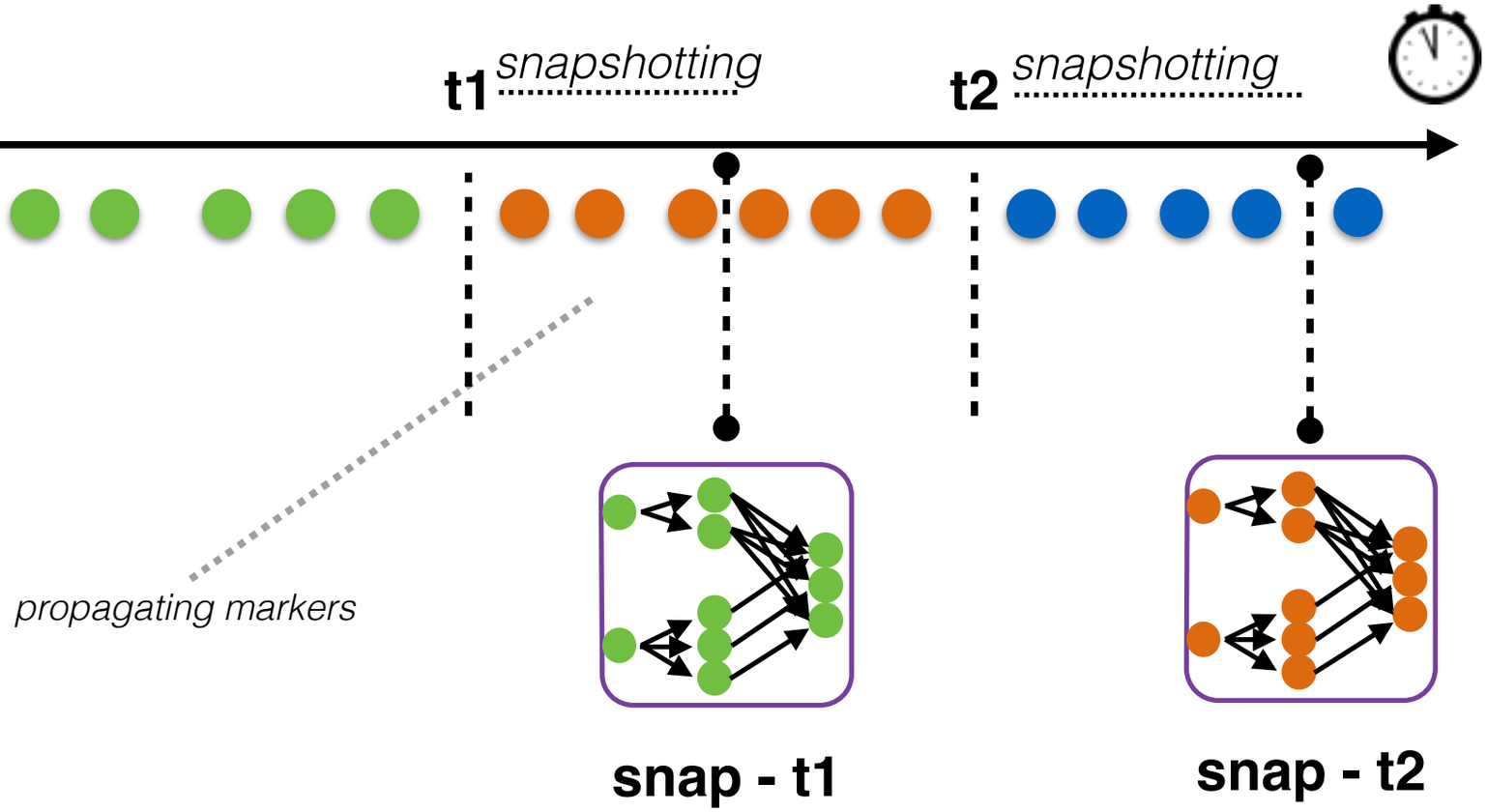
Using Markers/Barriers

- Triggers Snapshots
- Separates preshot-postshot events
- Leads to a consistent execution cut

markers propagate the snapshot execution under continuous ingestion



Asynchronous Snapshots



Asynchronous Barrier Snapshotting Benefits

- Taking advantage of the execution graph structure
- No records in transit included in the snapshot
- Aligning has lower impact than halting

Recovery

- **Full** rescheduling of the execution graph from the latest checkpoint - simple, no further modifications

DEMO

- https://github.com/senorcarbone/flink-fault-tolerant-stream-example