Improvements for large state in Apache Flink

Stefan Richter
@stefanrrichter
dataArtisans

April 11, 2017

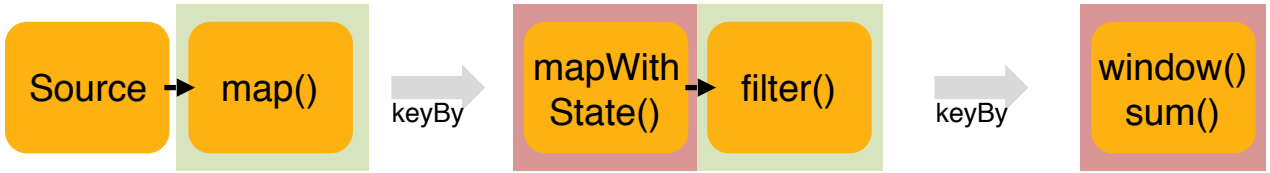
State in Streaming Programs

```
case class Event(producer: String, evtType: Int, msg: String)
case class Alert(msg: String, count: Long)
```


```
env.addSource(…)
   .map(bytes => Event.parse(bytes) )
   .keyBy("producer")
   .mapWithState { (event: Event, state: Option[Int]) => {
      // pattern rules
    }
   .filter(alert => alert.msg.contains("CRITICAL"))
   .keyBy("msg")
   .timeWindow(Time.seconds(10))
   .sum("count")
```

State in Streaming Programs

```
case class Event(producer: String, evtType: Int, msg: String)
case class Alert(msg: String, count: Long)
```

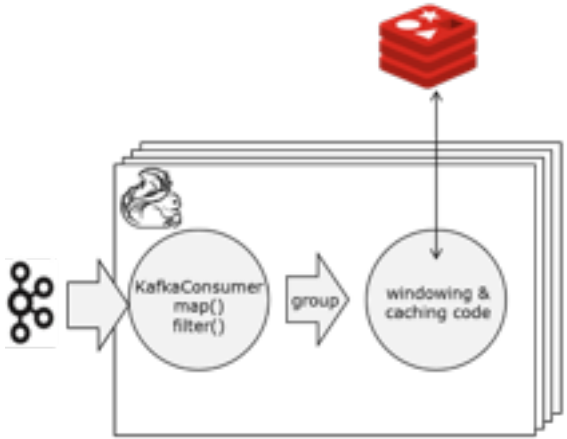



```
env.addSource(…)
  .map(bytes => Event.parse(bytes) )
  .keyBy("producer")
  .mapWithState { (event: Event, state: Option[Int]) => {
     // pattern rules
   }
  .filter(alert => alert.msg.contains("CRITICAL"))
  .keyBy("msg")
  .timeWindow(Time.seconds(10))
  .sum("count")
```

Stateless

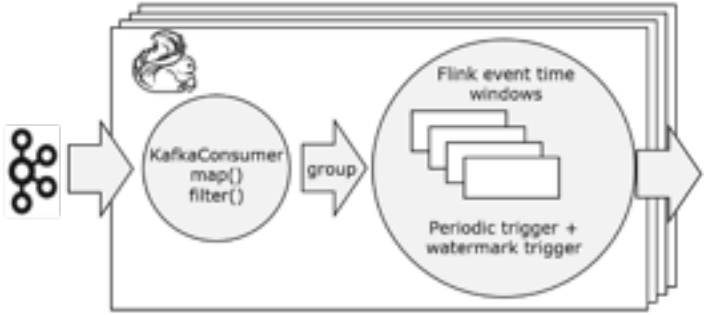
Stateful

Internal vs External State



External State

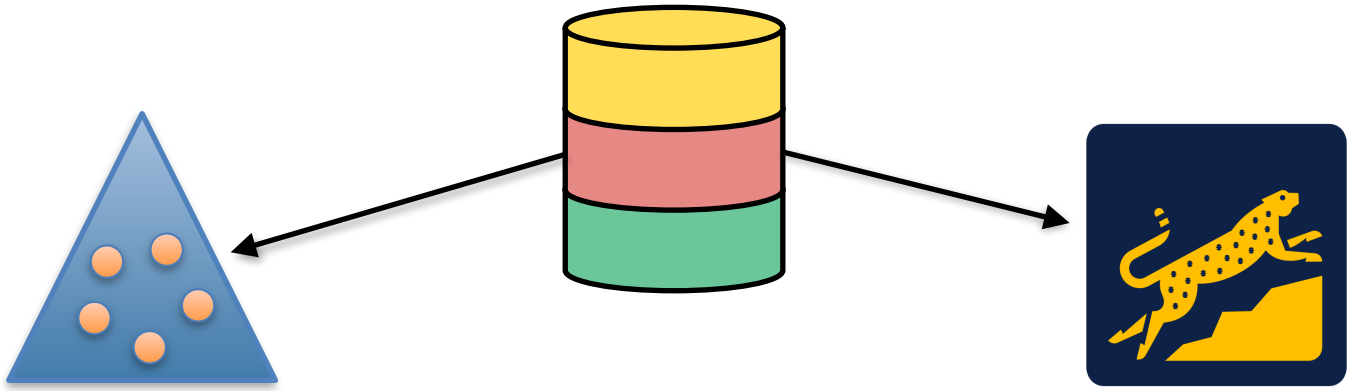
- State in a separate data store
- Can store "state capacity" independent
- Usually much slower than internal state
- Hard to get "exactly-once" guarantees



Internal State

- State in the stream processor
- Faster than external state
- Working area local to computation
- Checkpoints to stable store (DFS)
- Always exactly-once consistent
- Stream processor has to handle scalability

Keyed State Backends



HeapKeyedStateBackend

-State lives in memory, on Java heap

-Operates on objects

-Think of a hash map {key obj -> state obj}

-Async snapshots supported

RocksDBKeyedStateBackend

-State lives in off-heap memory and on disk

-Operates on bytes, uses serialization

-Think of K/V store {key bytes -> state bytes}

-Log-structured-merge (LSM) tree

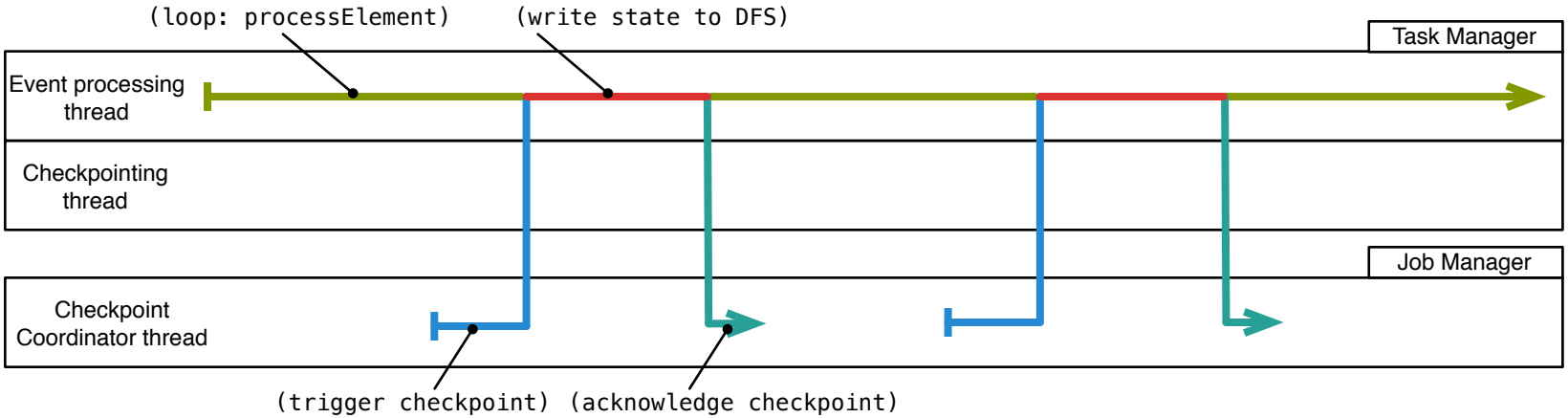
-Async snapshots

-Incremental snapshots

Asynchronous Checkpoints

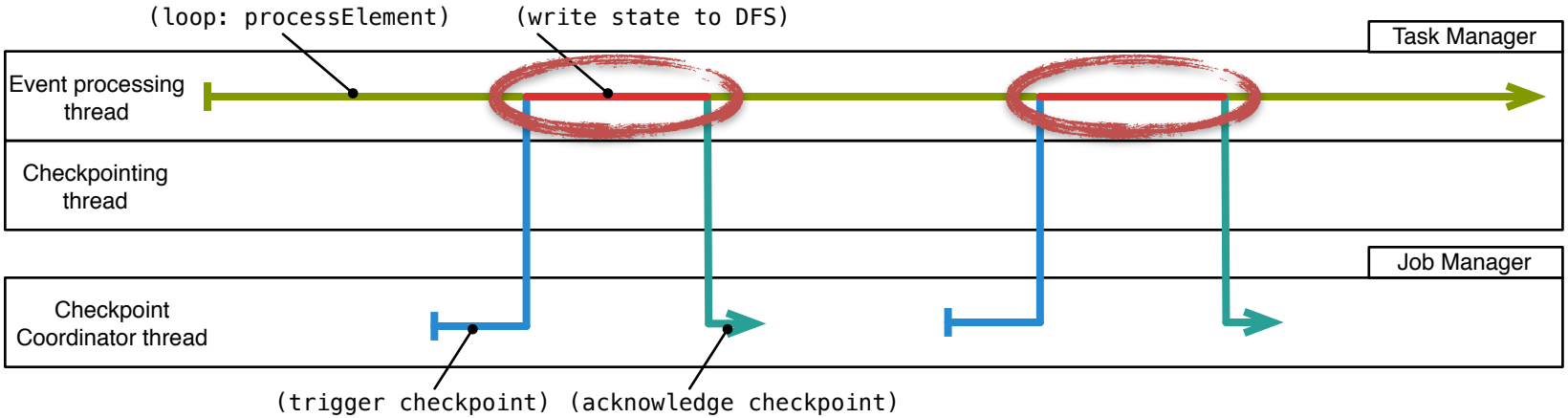
Synchronous Checkpointing

Why is async checkpointing so essential for large state?

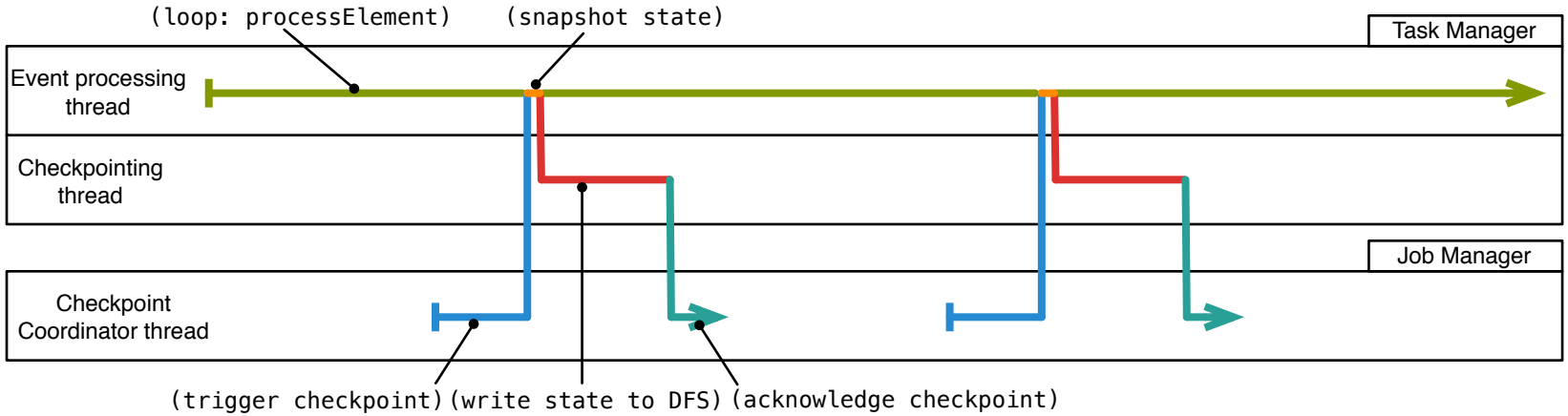


Synchronous Checkpointing

Problem: All event processing is on hold here to avoid concurrent modifications to the state that is written

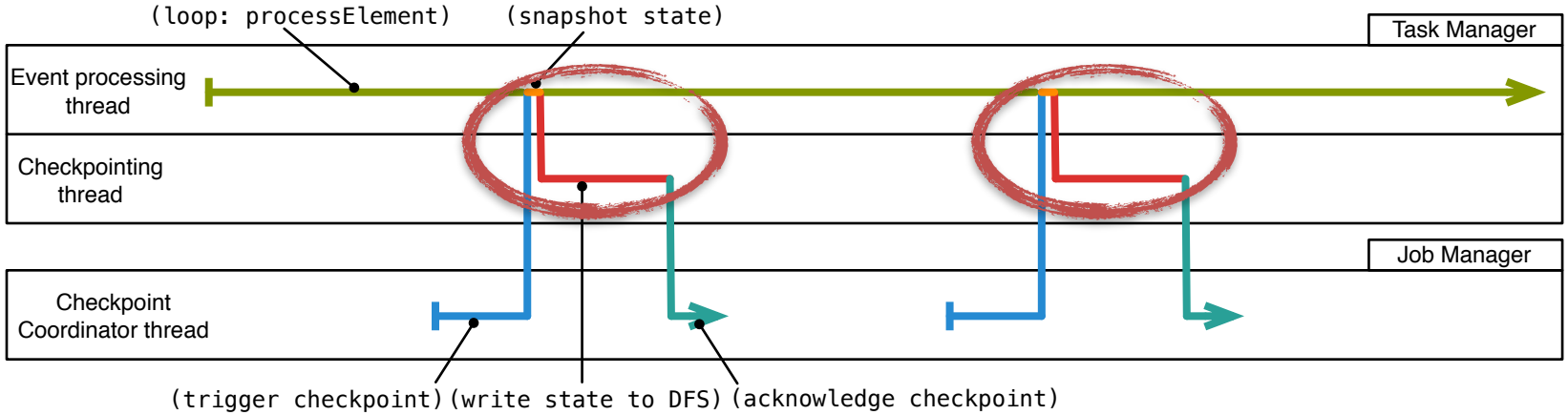


Asynchronous Checkpointing



Asynchronous Checkpointing

Problem: How to deal with concurrent modifications?



Incremental Checkpoints

What we will discuss

- What are incremental checkpoints?
- Why is RocksDB so well suited for this?
- How do we integrate this with Flink's checkpointing?

Driven by Alibaba.com and dataArtisans

Full Checkpointing

K	S
2	B
4	W
6	N

K	S
2	B
3	K
4	L
6	N

K	S
2	Q
3	K
6	N
9	S

K	S
2	B
4	W
6	N

K	S
2	B
3	K
4	L
6	N

K	S
2	Q
3	K
6	N
9	S

Checkpoint 1

Checkpoint 2

Checkpoint 3

Incremental Checkpointing

K	S
2	B
4	W
6	N

K	S
2	B
3	K
4	L
6	N

K	S
2	Q
3	K
6	N
9	S

K	S
2	B
4	W
6	N

Δ(−,c1)

K	S
3	K
4	L

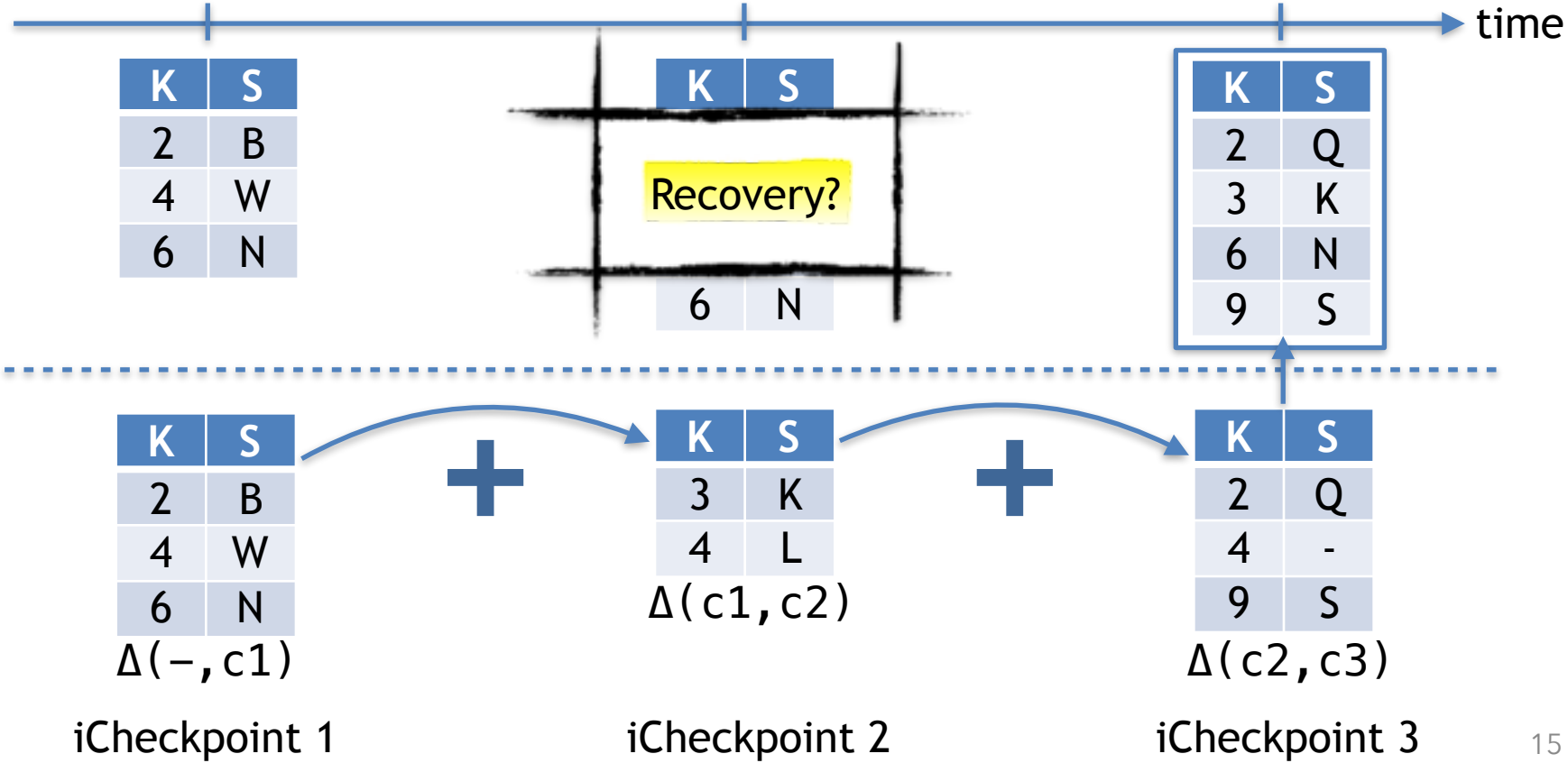
Δ(c1,c2)

K	S
2	Q
4	-
9	S

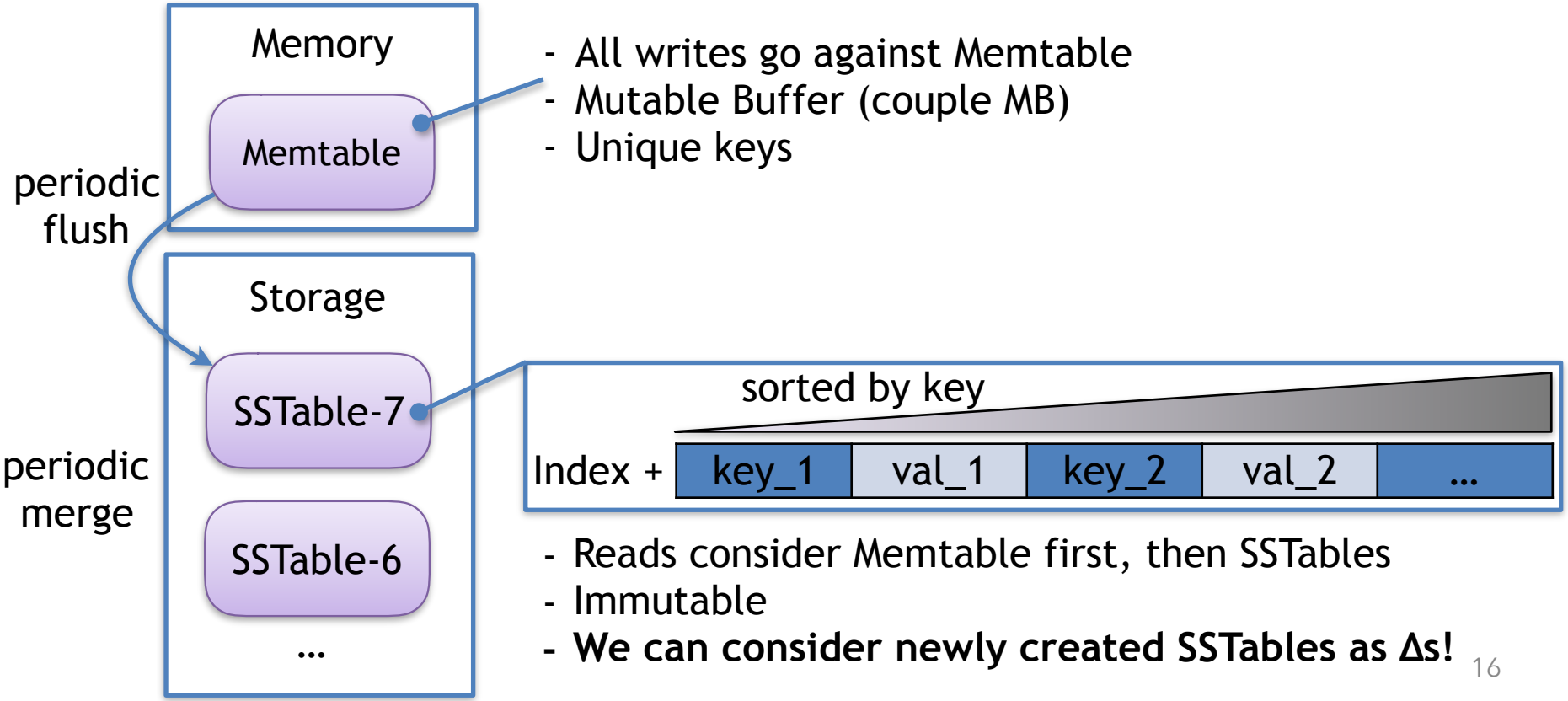
Δ(c2,c3)

iCheckpoint 1 iCheckpoint 2 iCheckpoint 3

Incremental Recovery

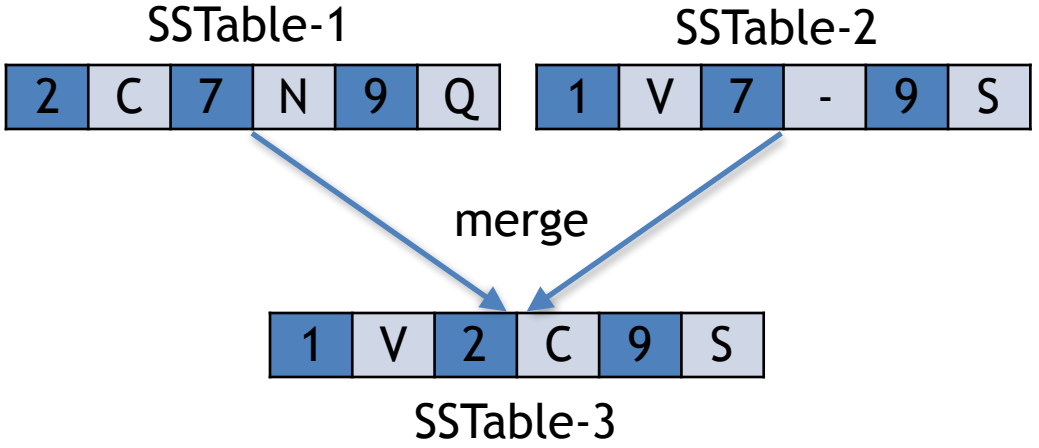


RocksDB Architecture (simplified)



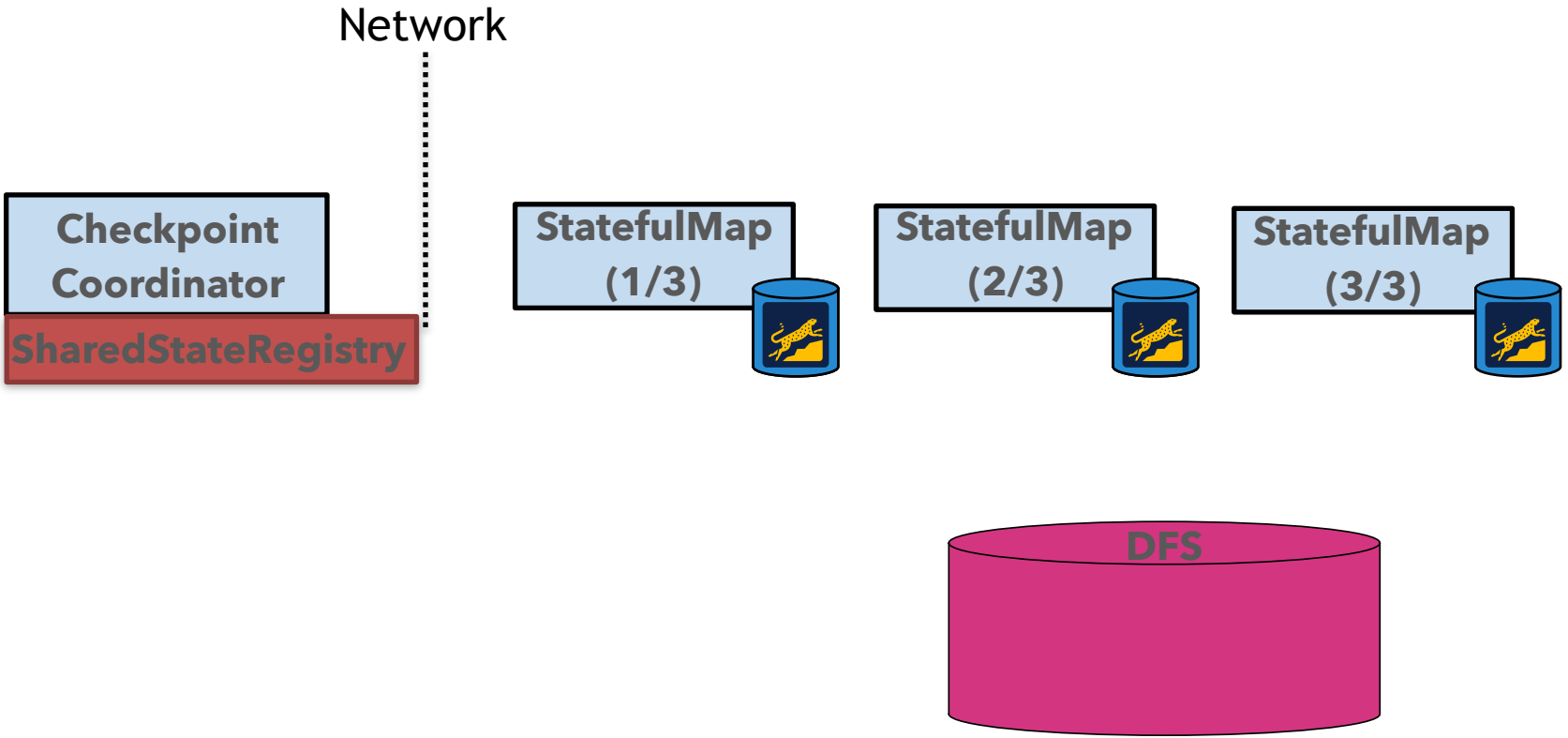
RocksDB Compaction

- Background Thread merges SSTable files
- Removes copies of the same key (latest version survives)
- Actually deletion of keys

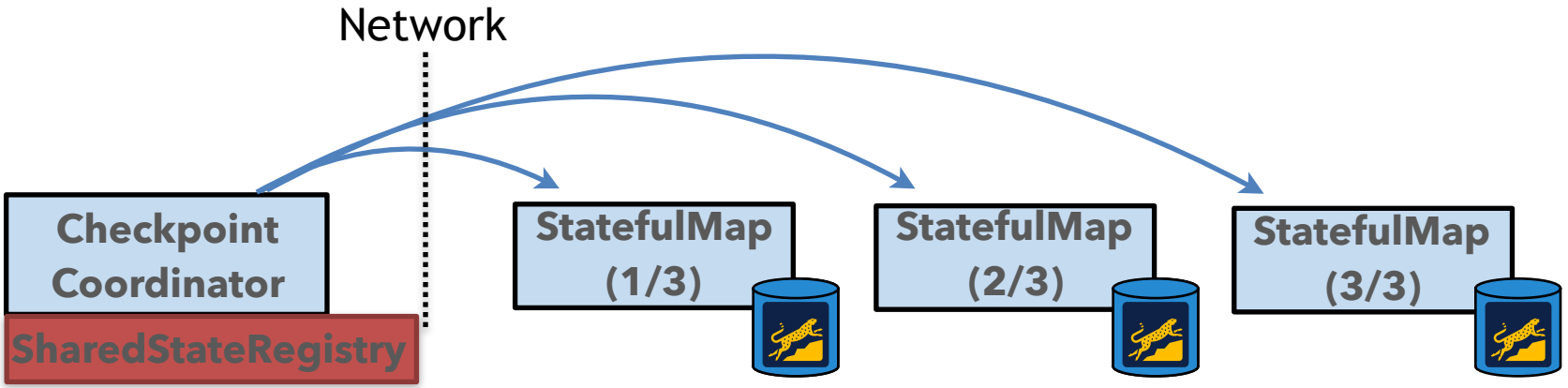


Compaction consolidates our Δs!

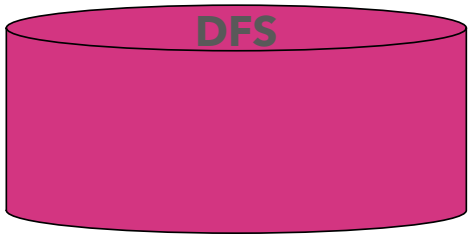
Flink's Incremental Checkpointing



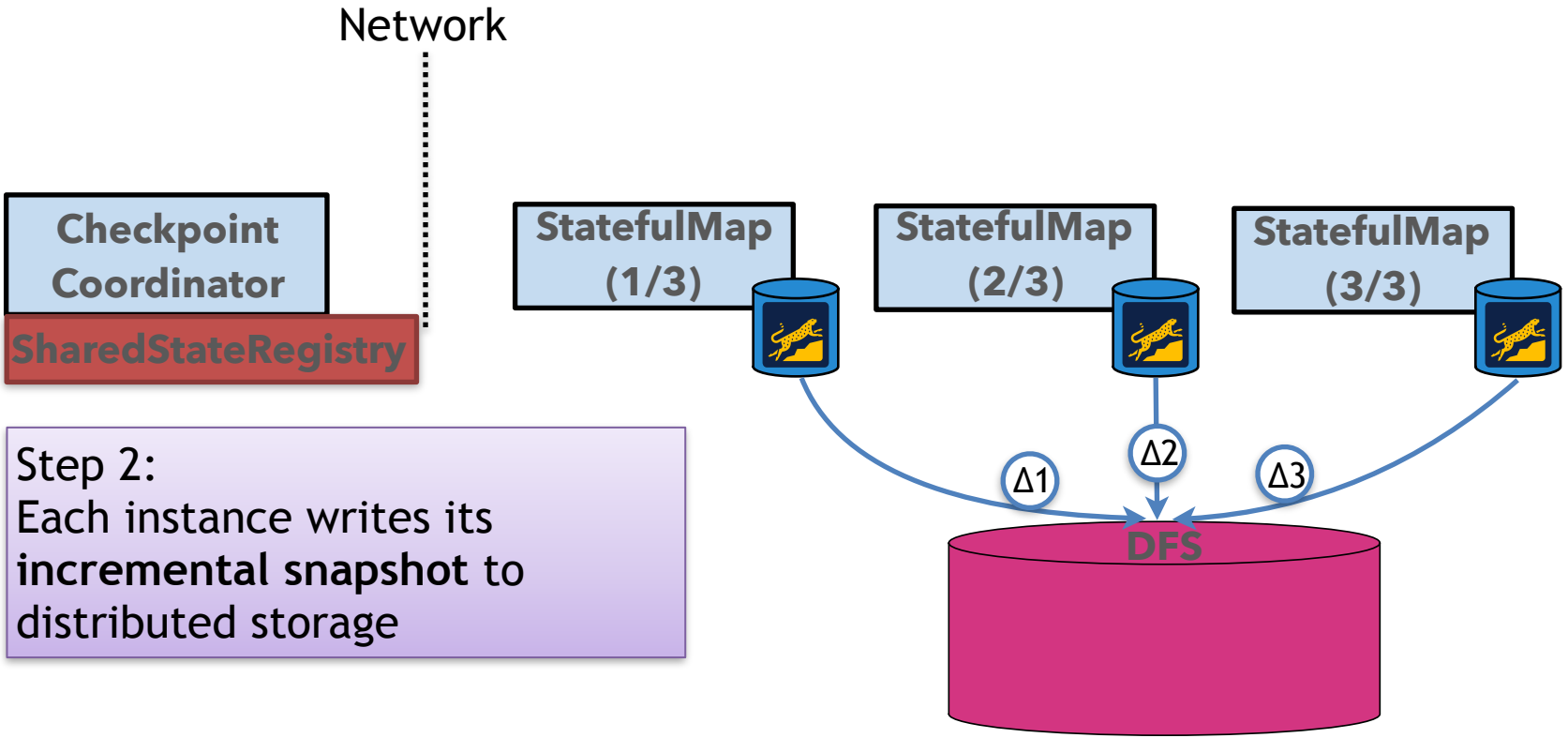
Flink's Incremental Checkpointing



Step 1:
Checkpoint Coordinator sends checkpoint barrier that triggers a snapshot on each instance

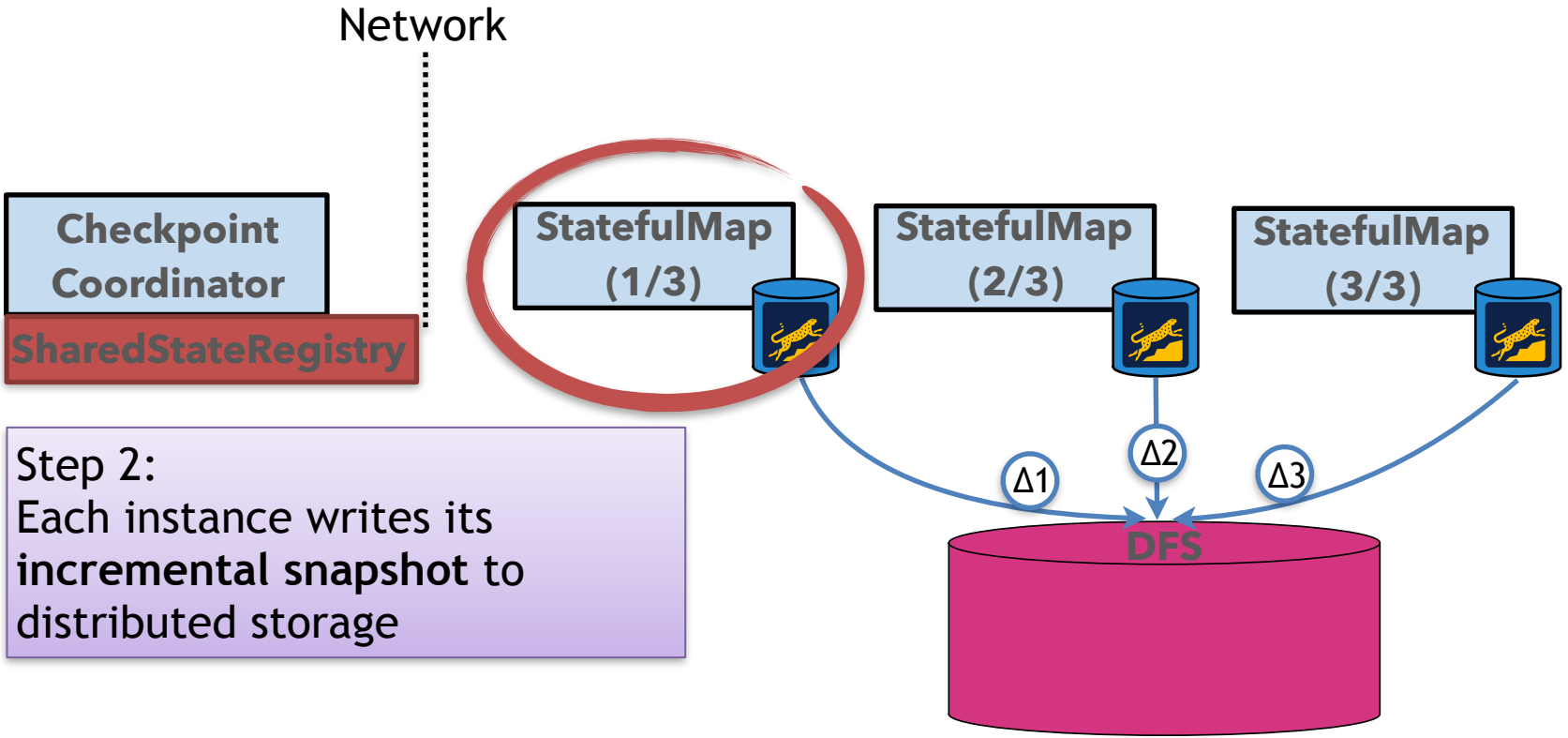


Flink's Incremental Checkpointing



Step 2:
Each instance writes its **incremental snapshot** to distributed storage

Flink's Incremental Checkpointing



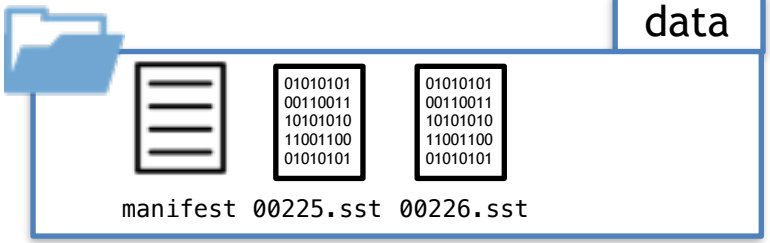
Step 2:
Each instance writes its **incremental snapshot** to distributed storage

Incremental Snapshot of Operator

SharedState
Registry

Local FS

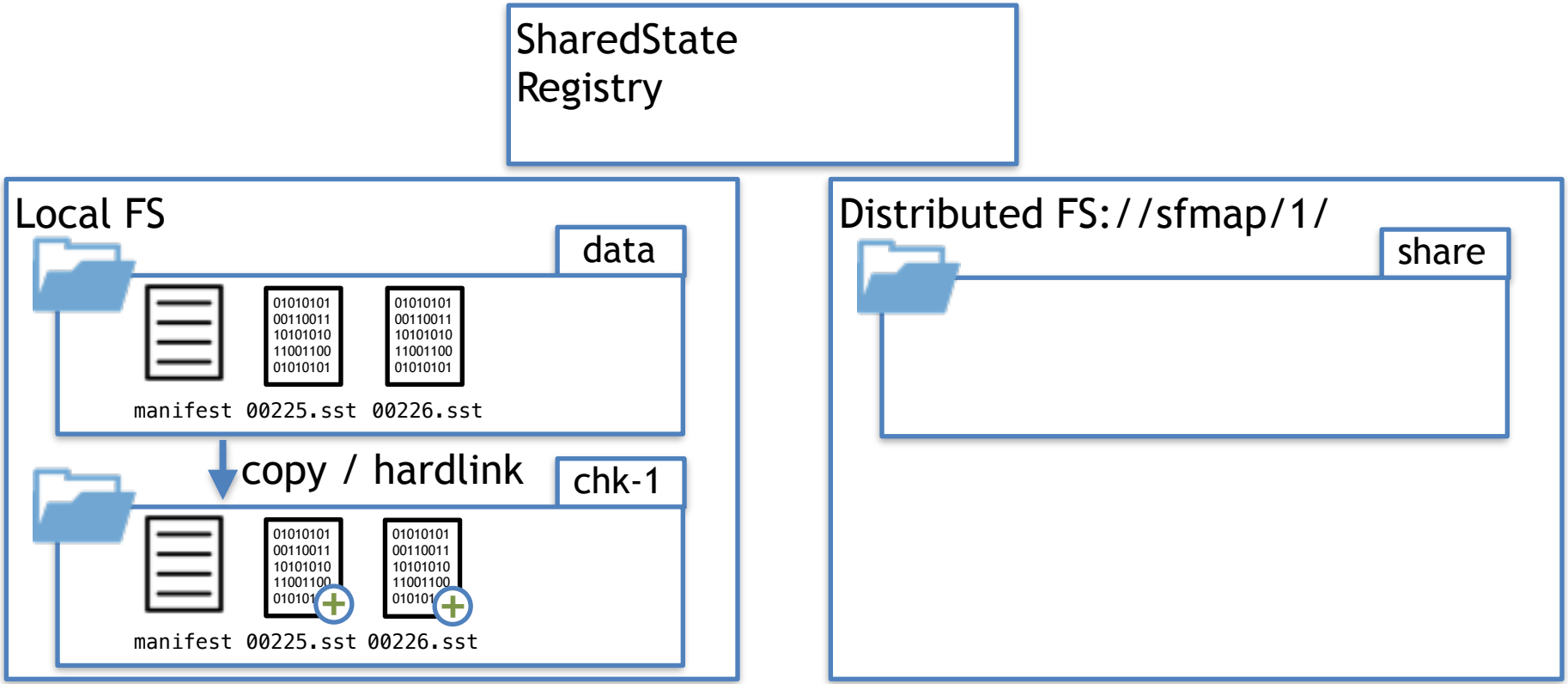
data



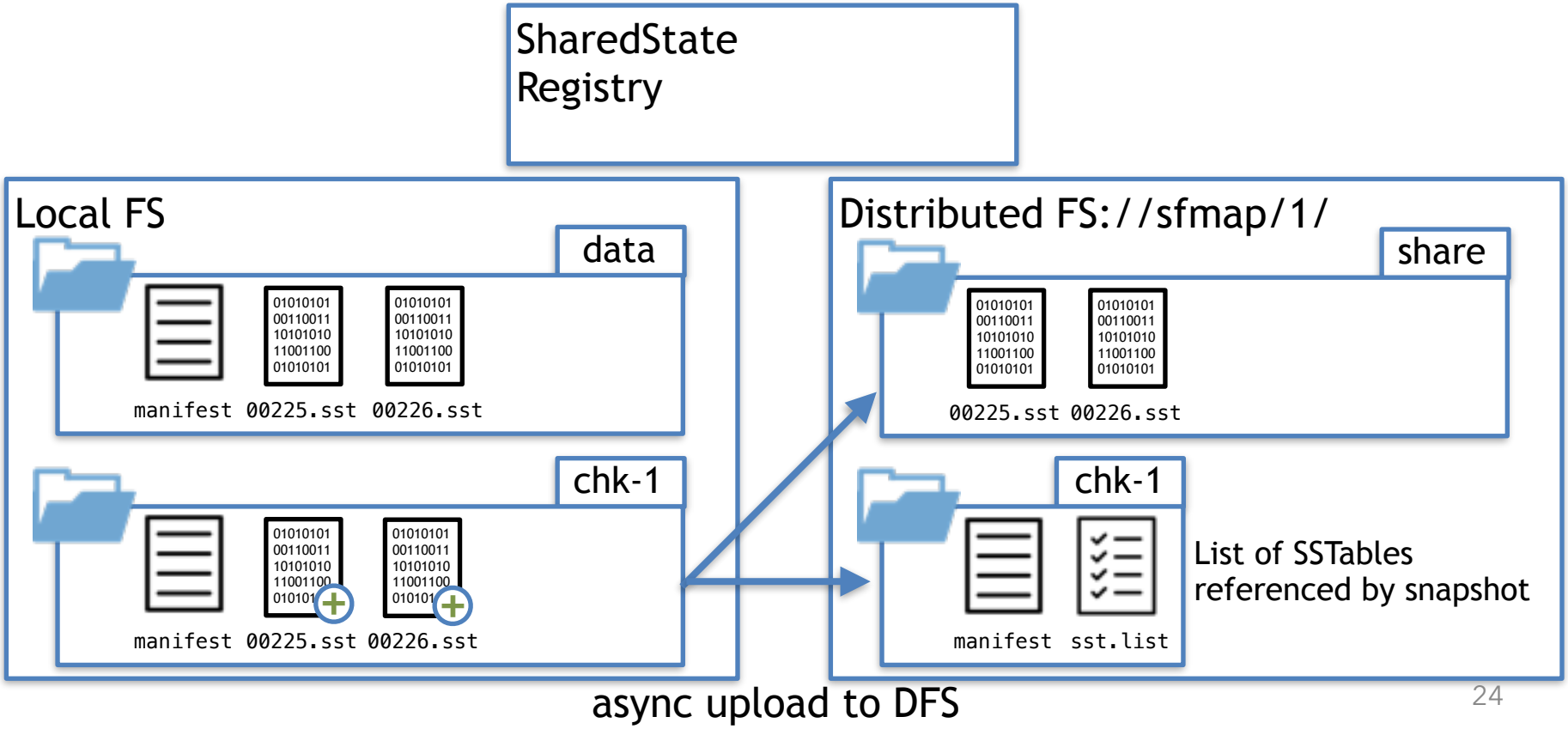
Distributed FS://sfmap/1/

share

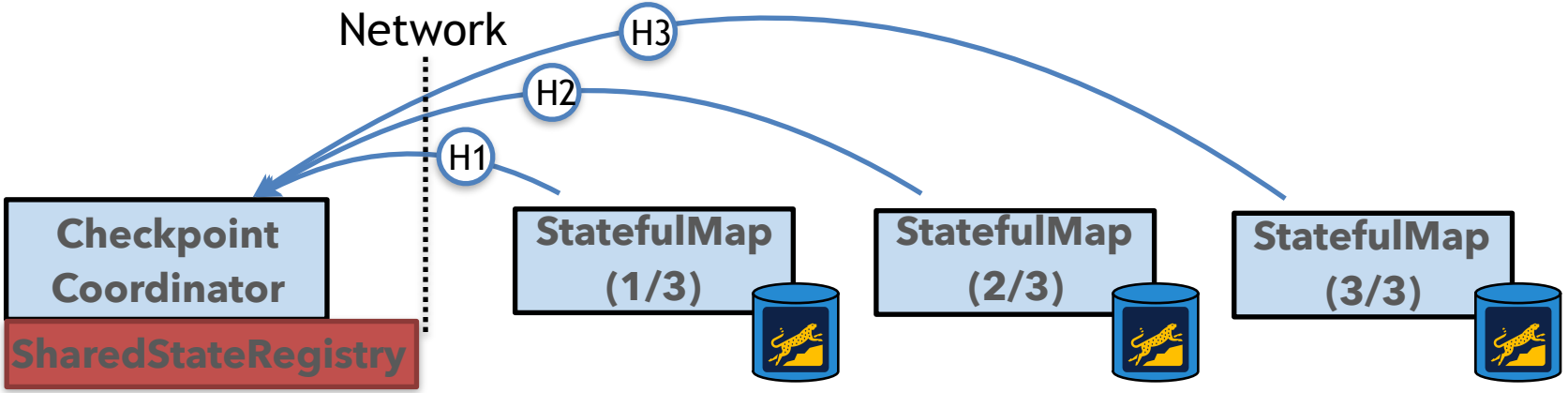
Incremental Snapshot of Operator



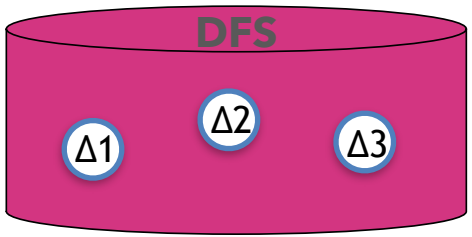
Incremental Snapshot of Operator



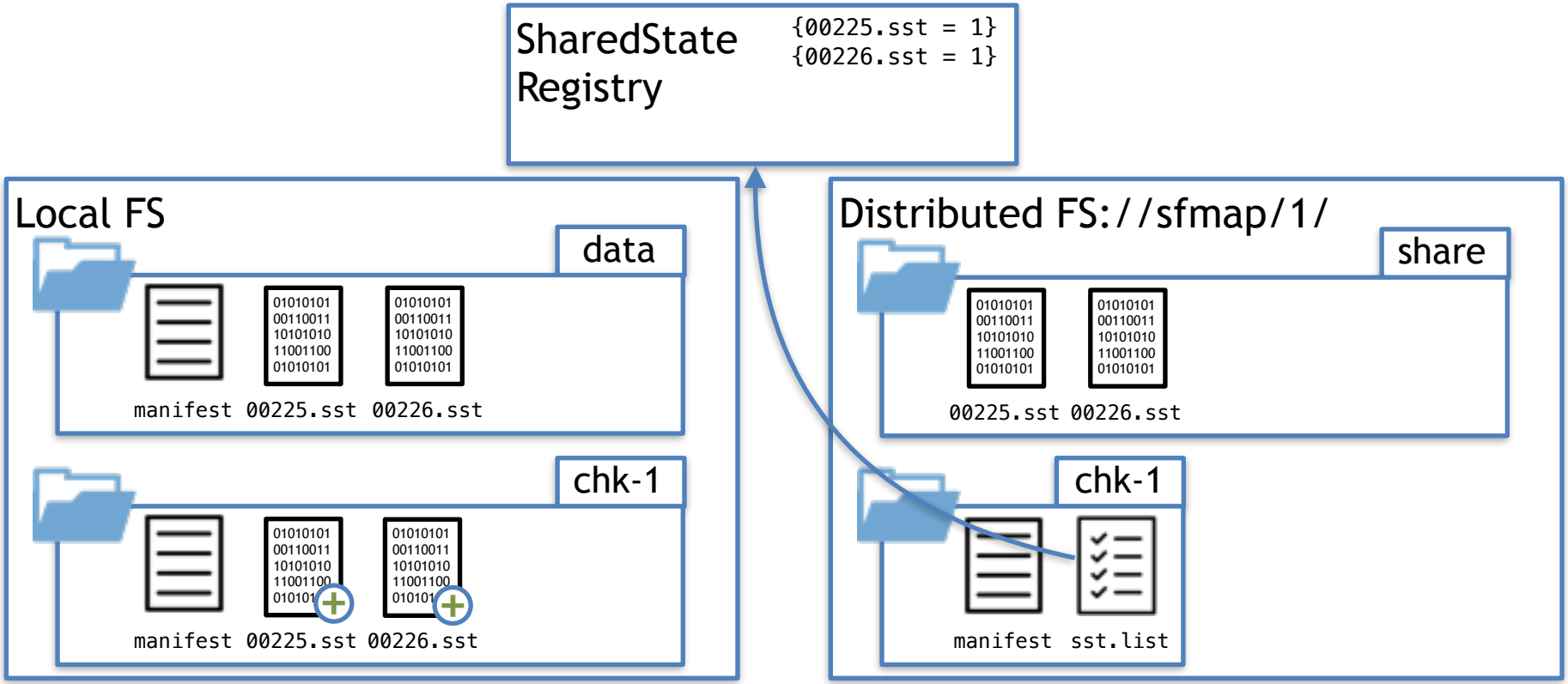
Flink's Incremental Checkpointing



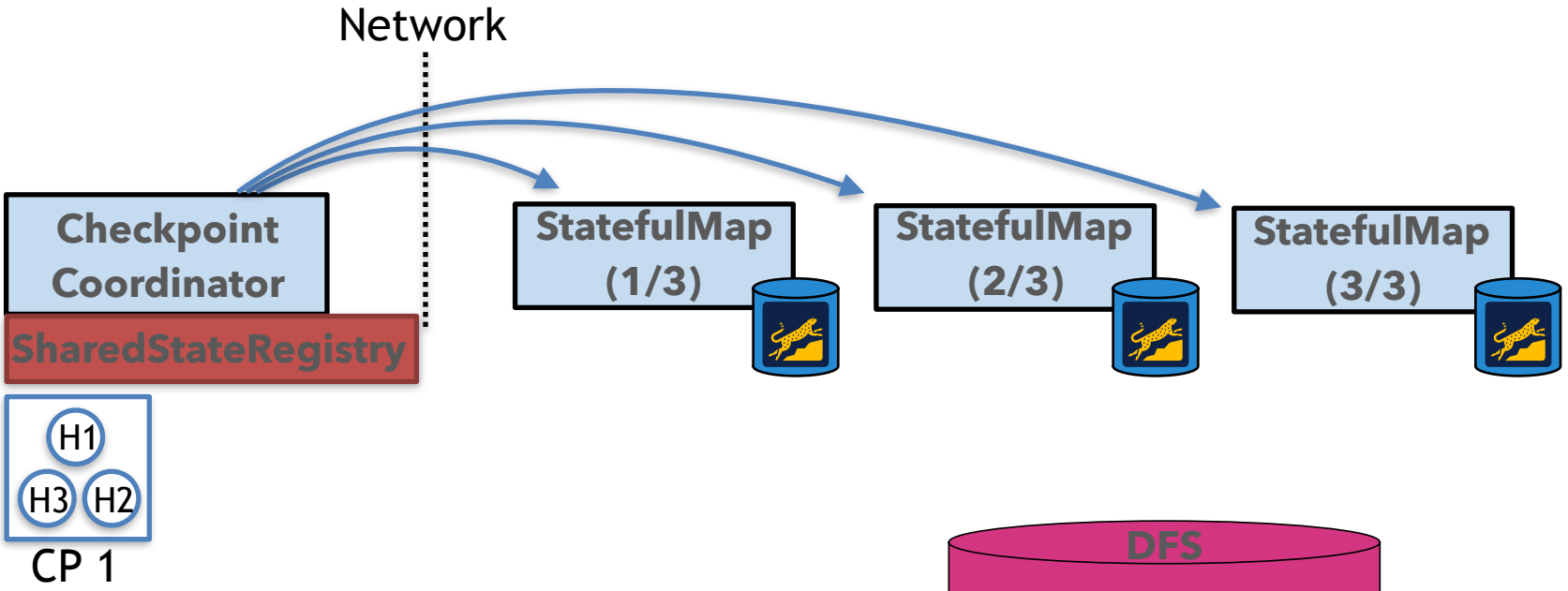
Step 3:
Each instance acknowledges and sends a handle (e.g. file path in DFS) to the Checkpoint Coordinator.



Incremental Snapshot of Operator

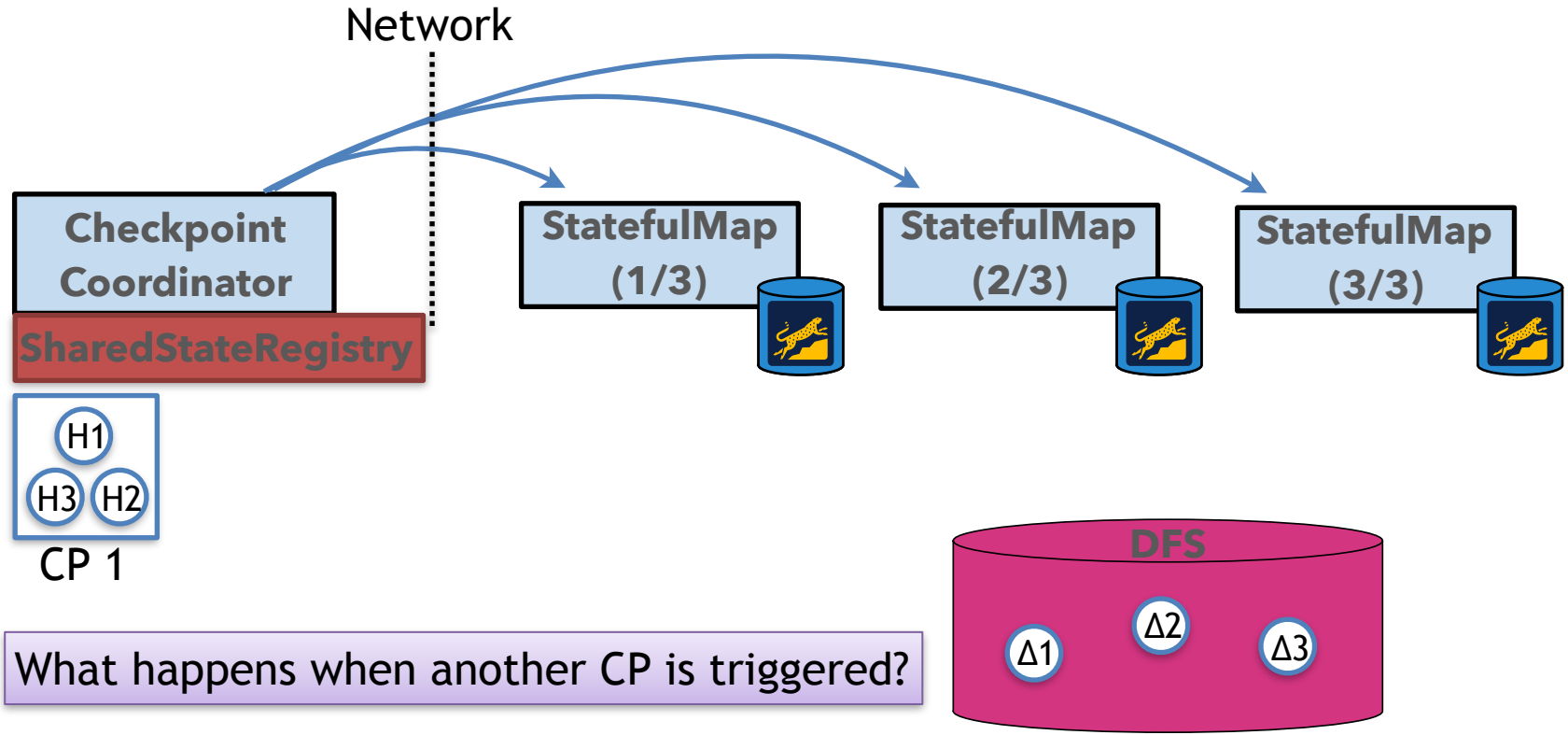


Flink's Incremental Checkpointing



Step 4:
Checkpoint Coordinator signals CP1 success to all instances.

Flink's Incremental Checkpointing



What happens when another CP is triggered?

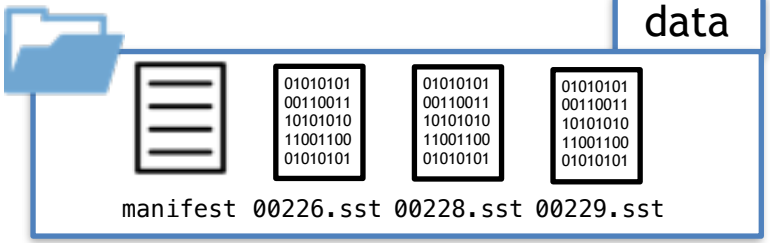
Incremental Snapshot of Operator

SharedState Registry

{00225.sst = 1}
{00226.sst = 1}

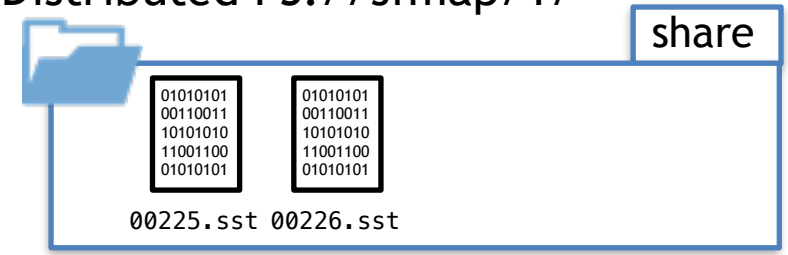
Local FS

data

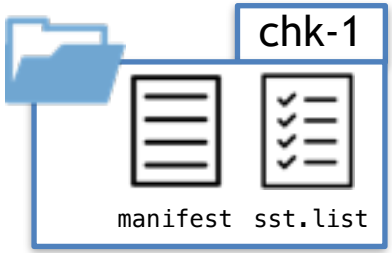


Distributed FS://sfmap/1/

share



chk-1



Incremental Snapshot of Operator

SharedState Registry

{00225.sst = 1}
{00226.sst = 1}

Local FS

data

manifest 00226.sst 00228.sst 00229.sst

chk-2

manifest 00226.sst 00228.sst 00229.sst

Distributed FS://sfmap/1/

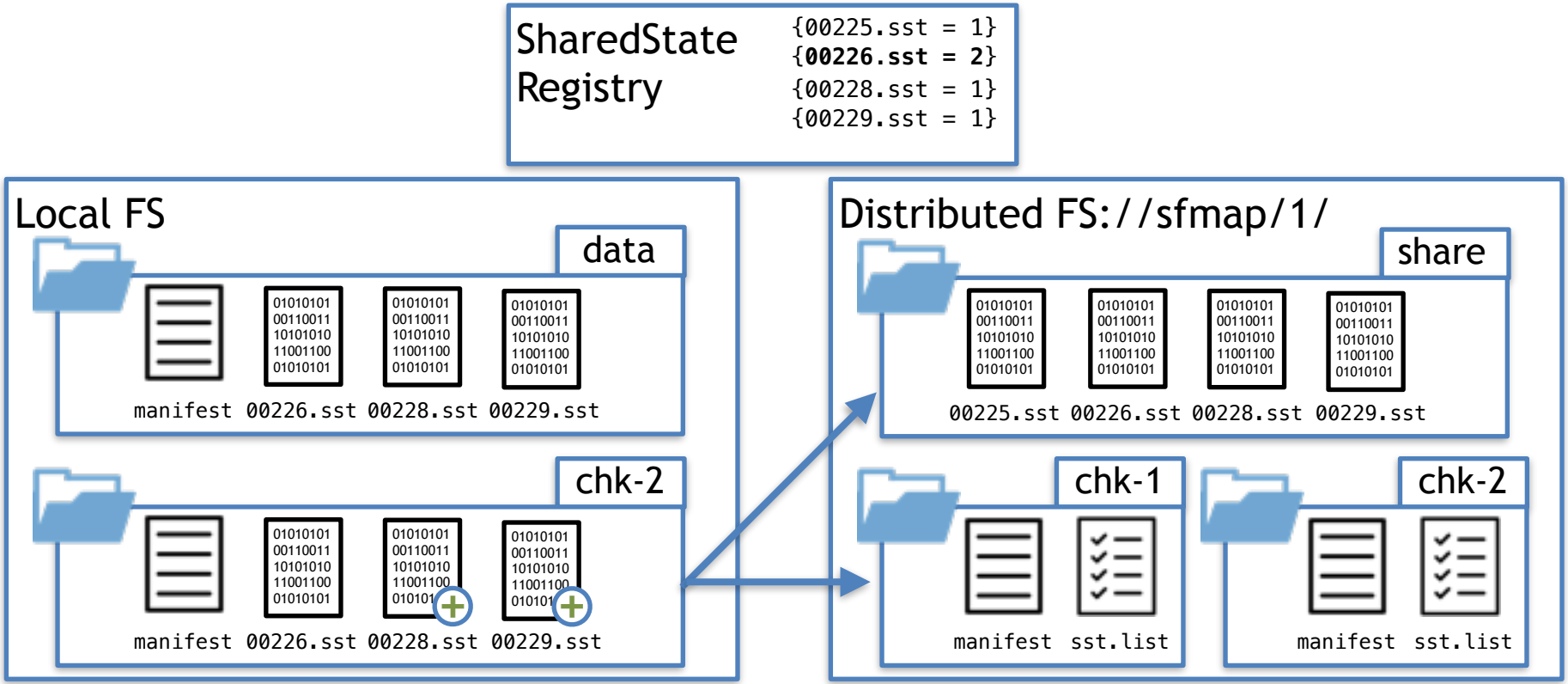
share

00225.sst 00226.sst

chk-1

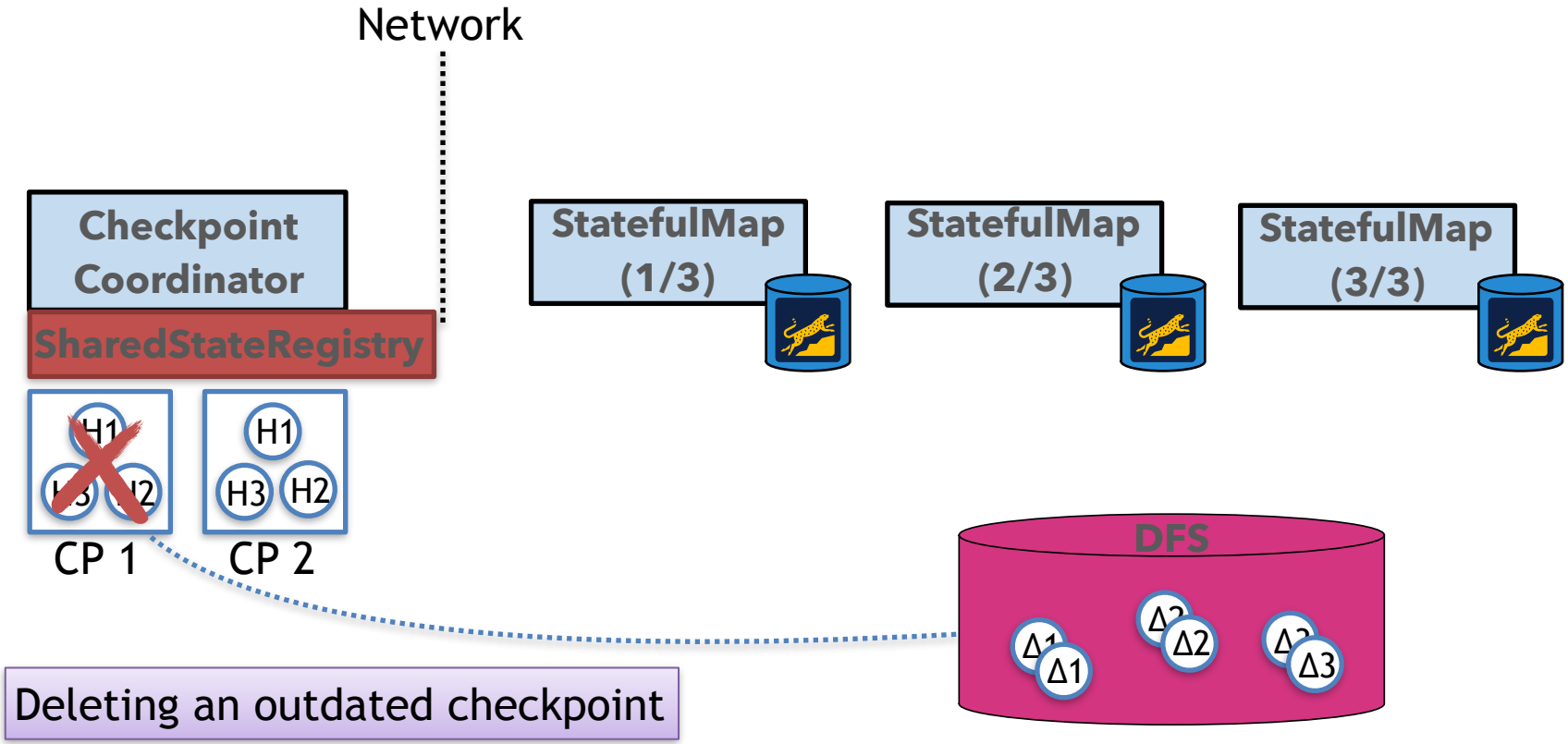
manifest sst.list

Incremental Snapshot of Operator



upload **missing** SSTable files

Deleting Incremental Checkpoints



Deleting Incremental Snapshot

SharedState Registry

{00225.sst = 1}
{00226.sst = 2}
{00228.sst = 1}
{00229.sst = 1}

Local FS

data

manifest 00226.sst 00228.sst 00229.sst

Distributed FS://sfmap/1/

share

00225.sst 00226.sst 00228.sst 00229.sst

chk-1

chk-2

manifest sst.list

Deleting Incremental Snapshot

SharedState Registry

~~{00225.sst = 0}~~
{00226.sst = 1}
{00228.sst = 1}
{00229.sst = 1}

Local FS

data

manifest 00226.sst 00228.sst 00229.sst

Distributed FS://sfmap/1/

share

00225.sst 00226.sst 00228.sst 00229.sst

chk-2

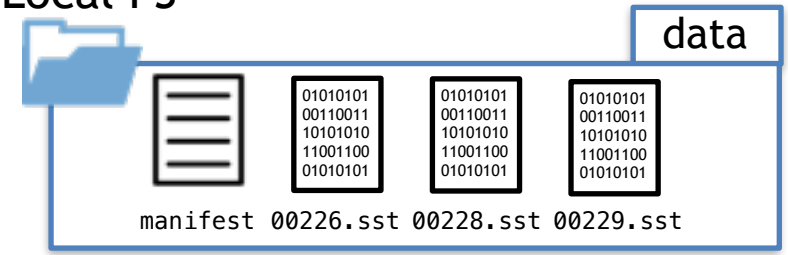
manifest sst.list

Deleting Incremental Snapshot

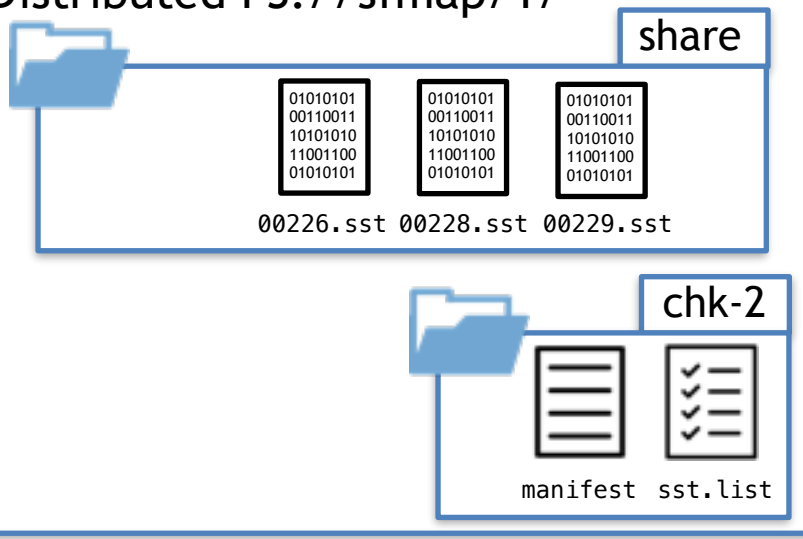
SharedState Registry

```
{00226.sst = 1}
{00228.sst = 1}
{00229.sst = 1}
```

Local FS



Distributed FS://sfmap/1/



Wrapping up

Incremental checkpointing benefits

- Incremental checkpoints can dramatically reduce CP overhead for large state.
- Incremental checkpoints are async.
- RocksDB's compaction consolidates the increments. Keeps overhead low for recovery.

Incremental checkpointing limitations

- Breaks the unification of checkpoints and savepoints (CP: low overhead, SP: features)
- RocksDB specific format.
- Currently no support for rescaling from incremental checkpoint.

Further improvements in Flink 1.3/4

- ***AsyncHeapKeyedStateBackend (merged)***
- ***AsyncHeapOperatorStateBackend (PR)***
- ***MapState (merged)***
- RocksDBInternalTimerService (PR)
- AsyncHeapInternalTimerService

Questions?