Clonos

Consistent Causal Recovery for Highly-Available Streaming Dataflows

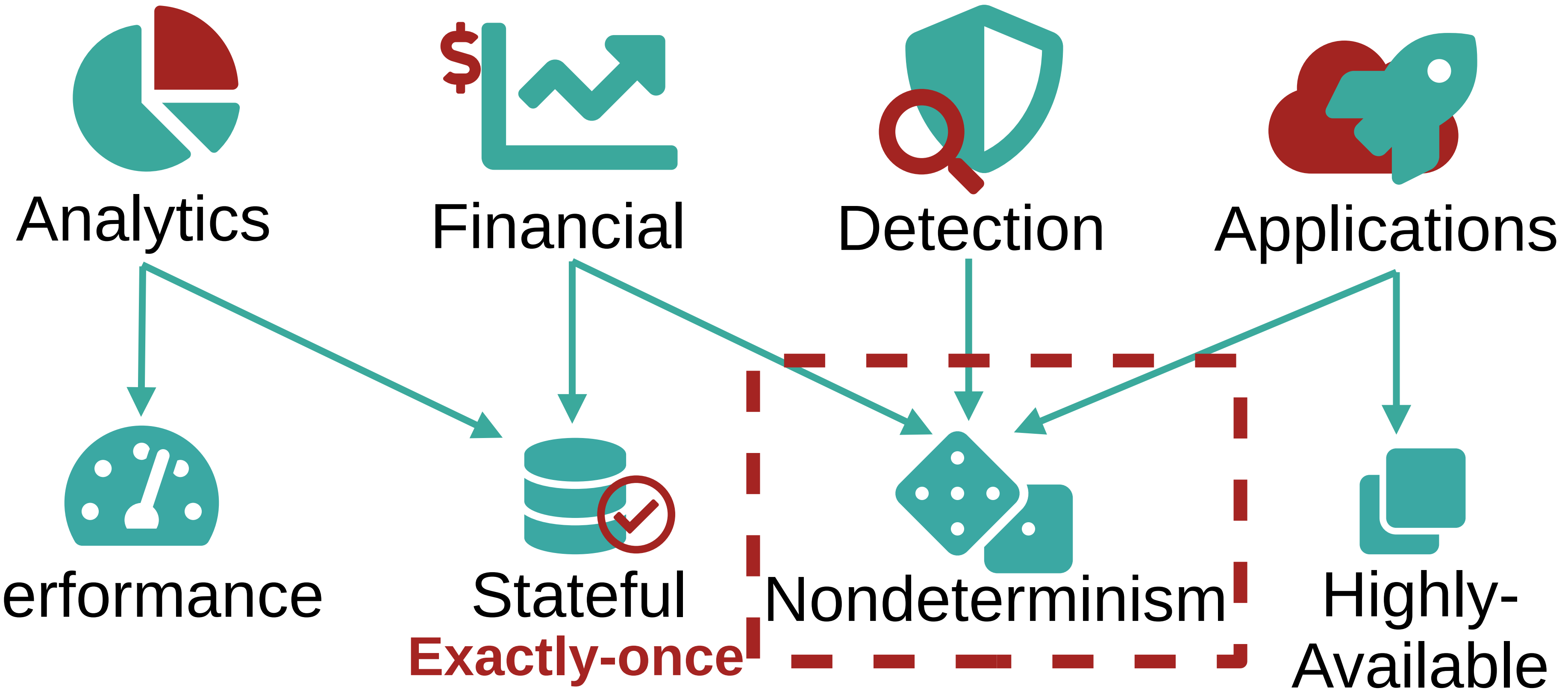
Pedro Silvestre, Marios Fragkoulis, Diomidis Spinellis, Asterios Katsifodimos

ACM SIGMOD 2021

TU Delft

WIS Web Information Systems

SPS Use-Cases are Diverse



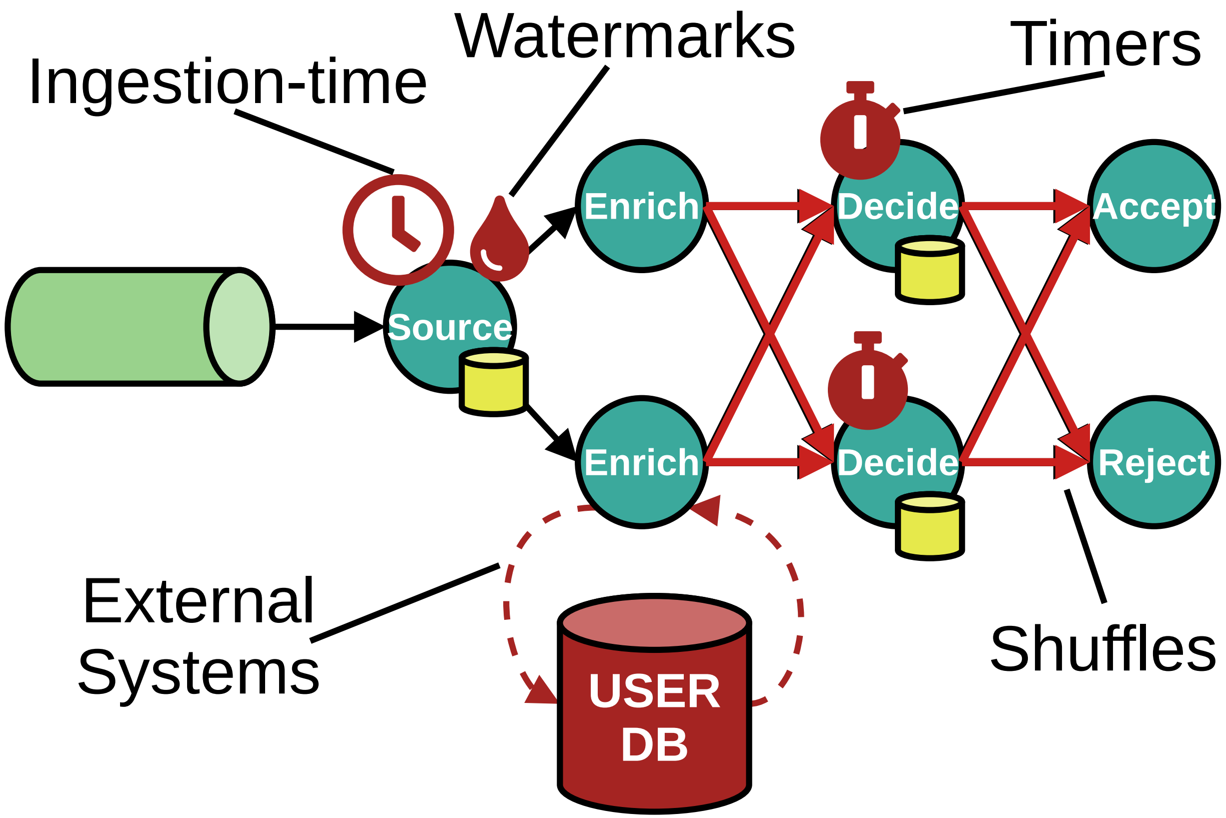
Nondeterminism in SPSs

- Dependence on factors other than initial state or input
- System.currentTimeMillis()
- User-defined functions (UDFs)
- Essencial system functions

An Example: Fraud Detection

And more:

- Processing-time
- Idle stream detection
- Load balancing
- RPCs
- Multi-threaded operators



Two Classes of Systems

Performance

Stateful
Exactly-once

Nondeterminism

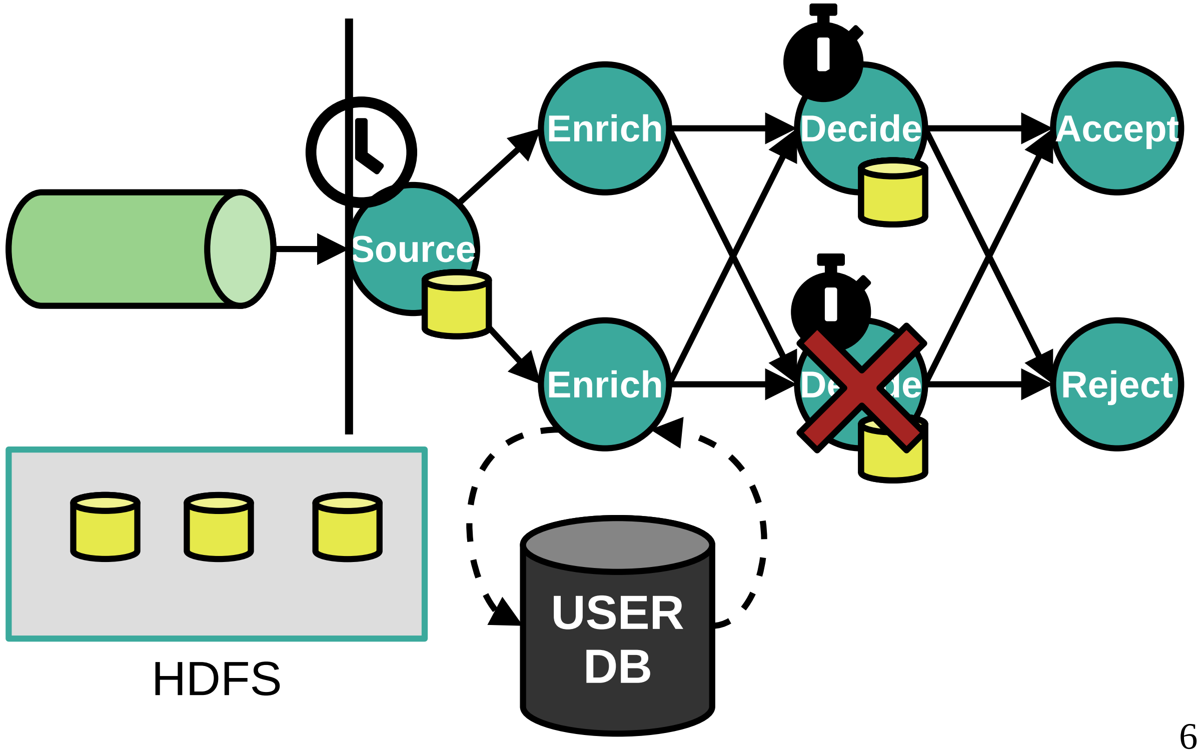
Highly-Available

Global Recovery

Local Recovery

Global Recovery is Slow

- Supports nondeterminism
- Slow recovery
  - Worse on large graphs
  - Stop-the-world
- No High-Availability



Two Classes of Systems

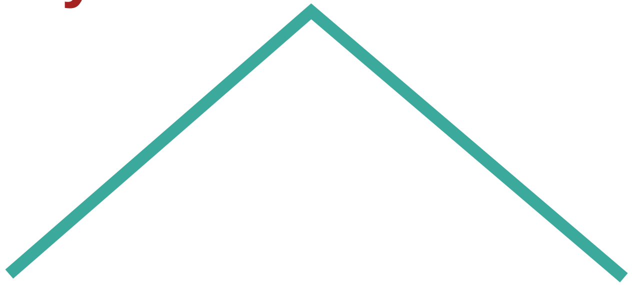
Performance

Stateful

Exactly-once

Nondeterminism

Highly-Available



Global Recovery

Local Recovery

Local Recovery is Limited

Prior work strategies:

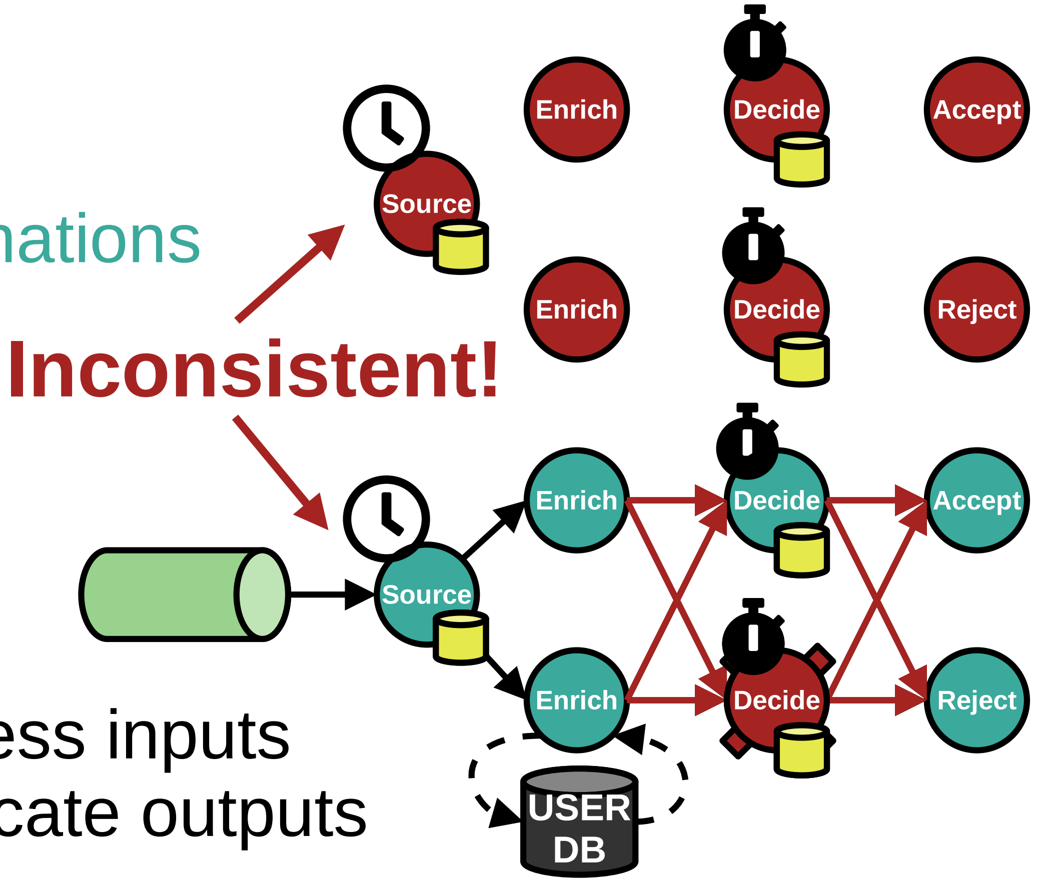
1) Pure
2) Passive Standby
3) Active Standby
4)Upstream Backup

2 incarnations

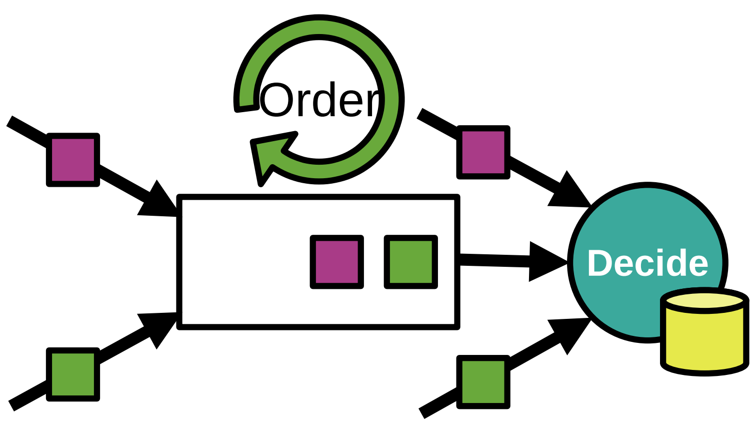
Choose one:

- Consistency
- Nondeterminism

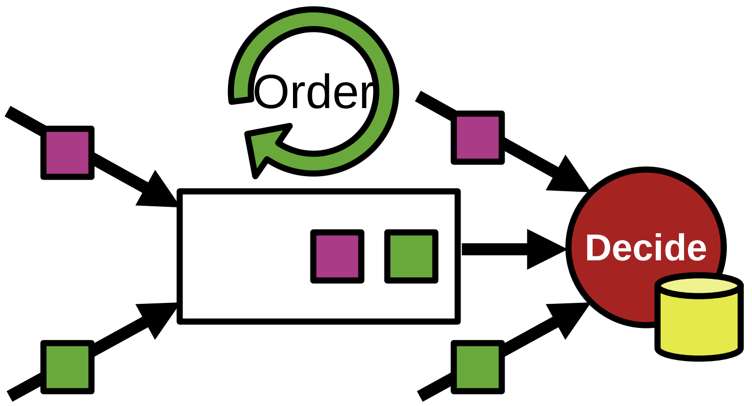
1. Reprocess inputs
2. Deduplicate outputs



Local Recovery Affects Performance



1st incarnation



2nd incarnation

Research Goal

- Provide **local recovery** for high-availability
- With support for **nondeterminism**
- Without sacrificing **performance** or **exactly-once**

Clonos

Consistency in spite of Nondeterminism

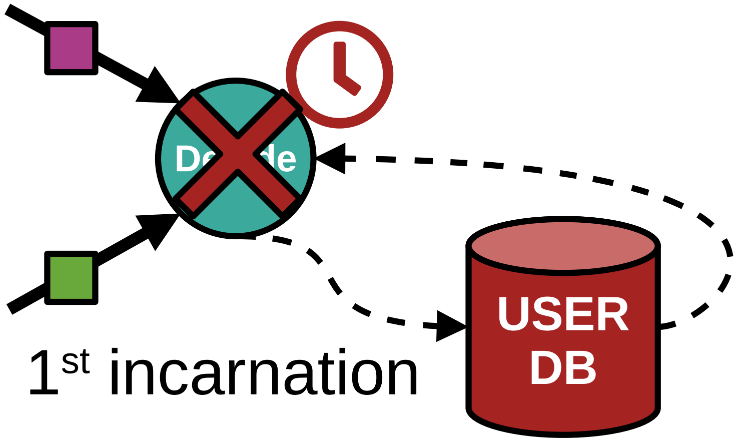
Determinant

Log

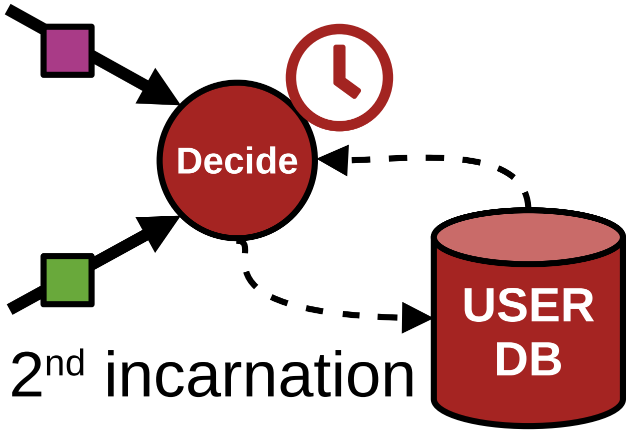
Read Channel 1	Read Channel 2	Time stamp 192...7	HTTP {"john": 3}

Log Copy

Read Channel 1	Read Channel 2	Time stamp 192...7	HTTP {"john": 3}



1st incarnation



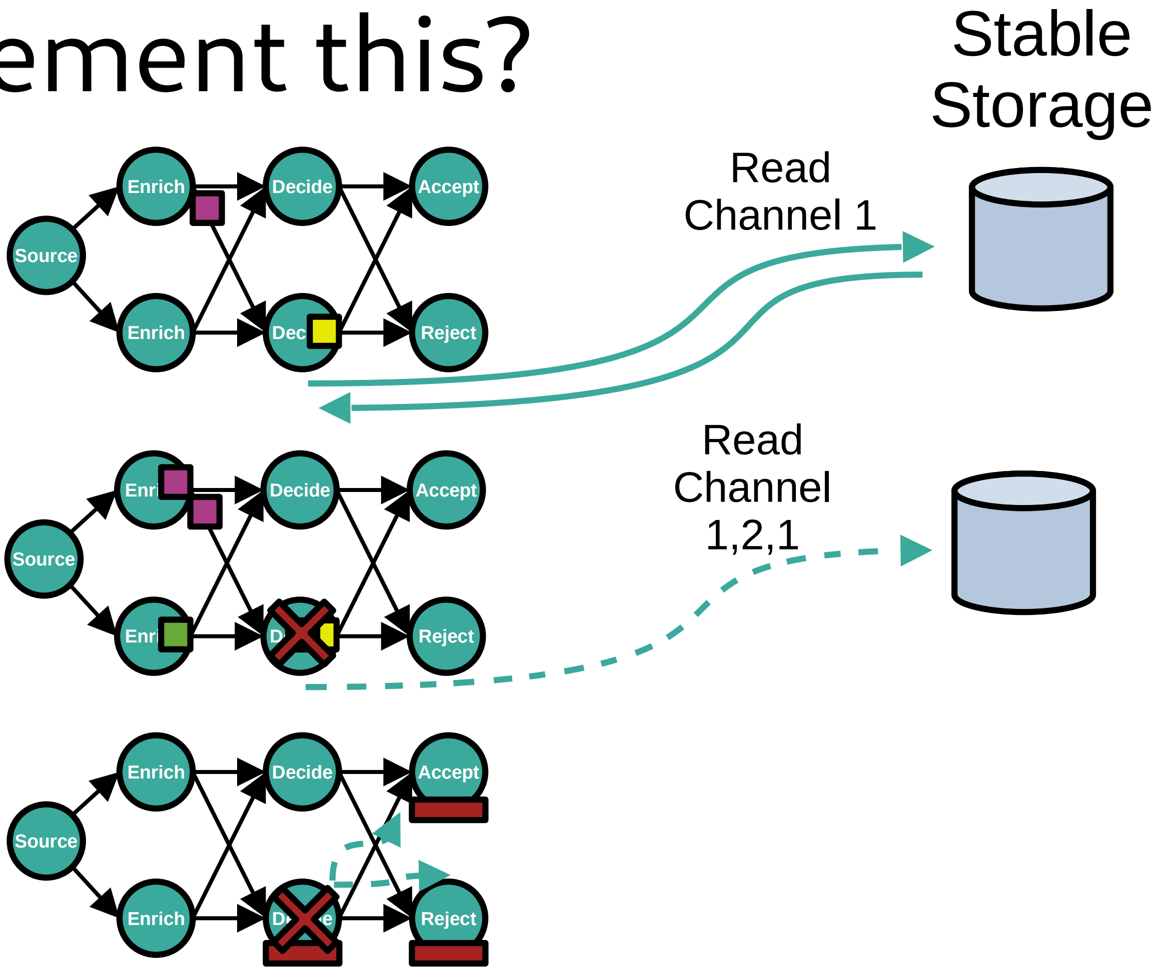
2nd incarnation

Contributions

- Propose logging as an efficient remedy to this tension
- A fault tolerance approach combining checkpointing, standby operators and causal logging
- Analysis of nondeterminism and of Clonos’ exactly-once correctness
- Empirical experiments in a realistic deployment

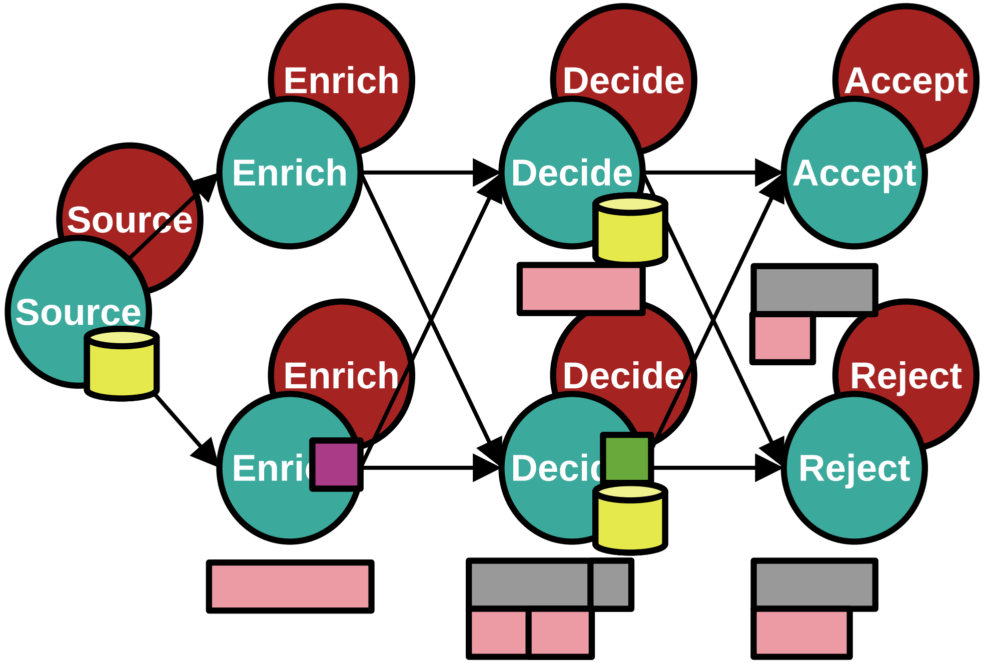
How to implement this?

- Pessimistic
  - - Not Performant
  - + Exactly-once
- Optimistic
  - + Performant
  - - Not Exactly-once
- Causal
  - + Performant
  - + Exactly-once



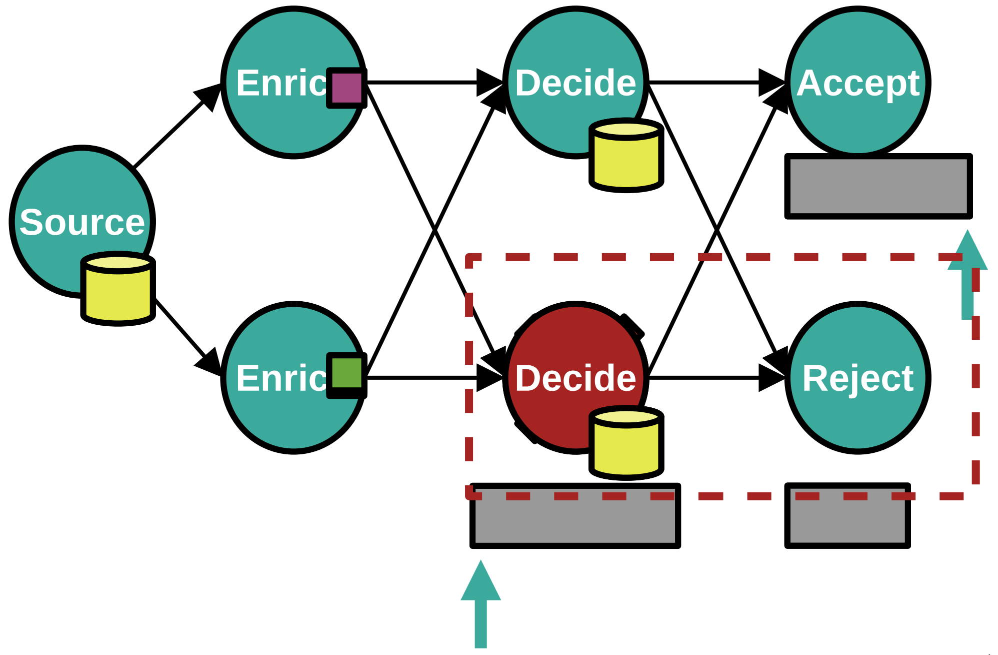
Clonos – Normal Operation

- Passive standby and snapshot dispatch
  - Alternatively, pure LR
- In-Flight Logging
- Causal Logging
  - Shared Incrementally



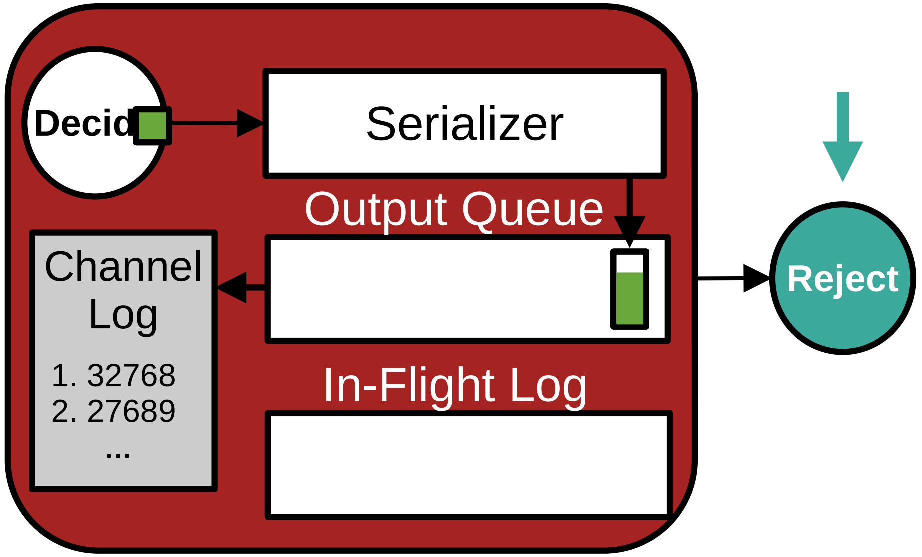
Overview – Recovery

1) Activate Standby
2) Reconfigure Network
3) Retrieve log
4) Request in-flight replay
5) Reprocess
6) Deduplicate



Deduplication happens concurrently

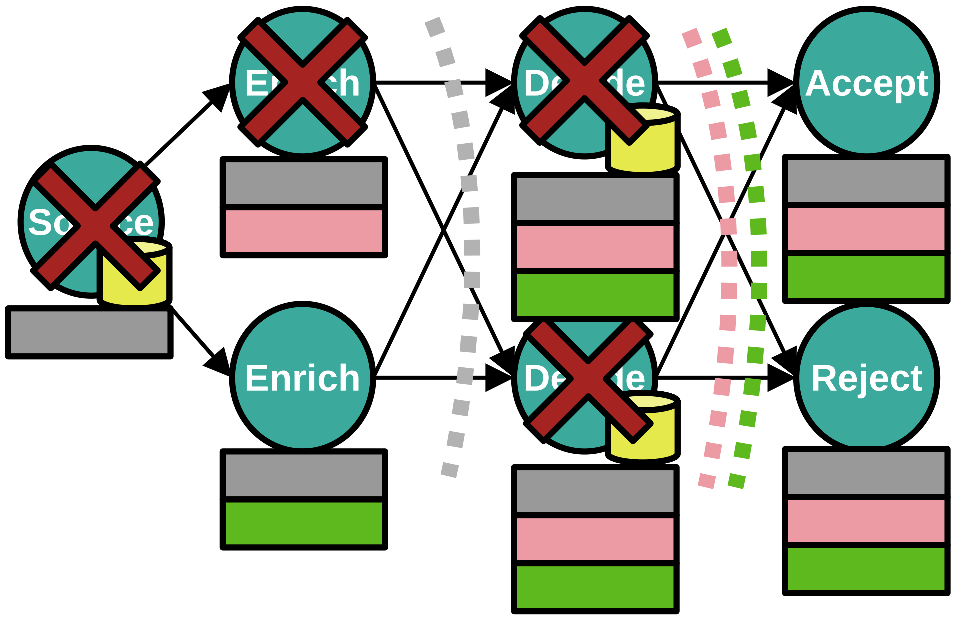
- Records are serialized into buffers.
- Deduplicate by moving buffers to the in-flight log
- Simultaneously rebuilds in-flight log
- The receipt is a buffer size determinant (channel log)



Partial Replication for Scalability

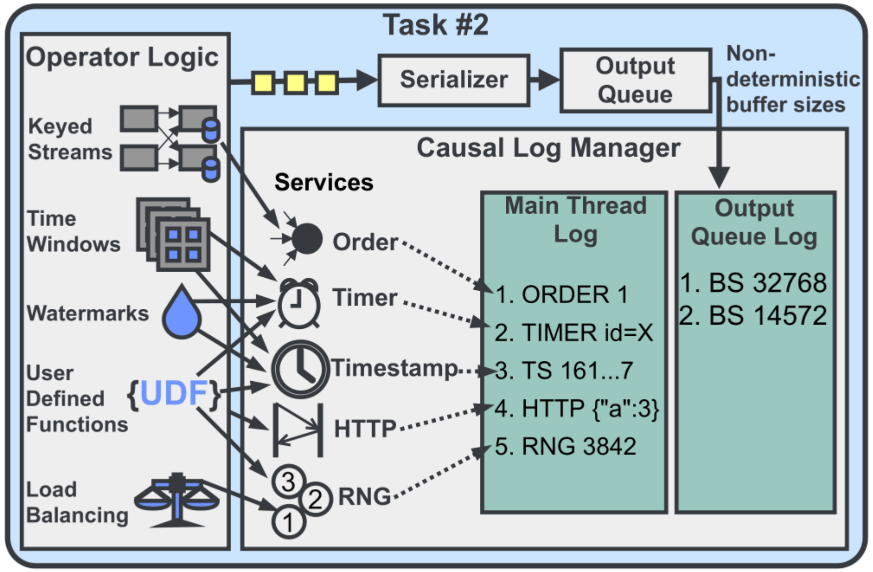
- Full replication can be costly (Network, CPU)
- Determinant sharing depth (DSD)
- Can still handle a large number of failures
- Proof of correctness

DSD = 1



Services Make Clonos Transparent

- Users oblivious to Clonos
- Built-in causal services
- Register new causal services



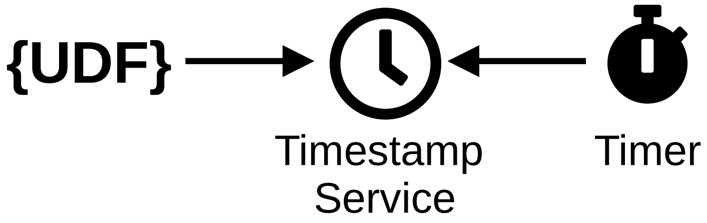
System.currentTimeMillis()

↓

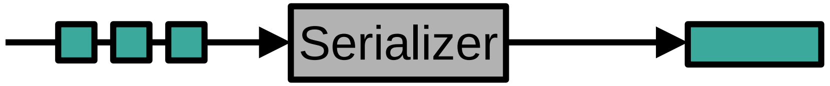
timestampService.currentTimeMillis()

Design Decisions & Optimizations

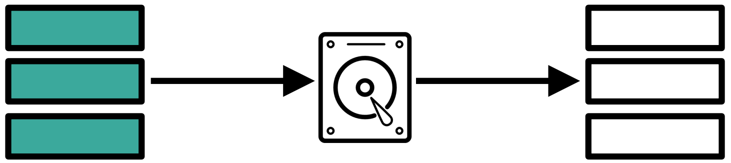
- Efficient Causal Services
  - Orders of magnitude smaller log



- Track buffers not records
  - Input, output and in-flight log

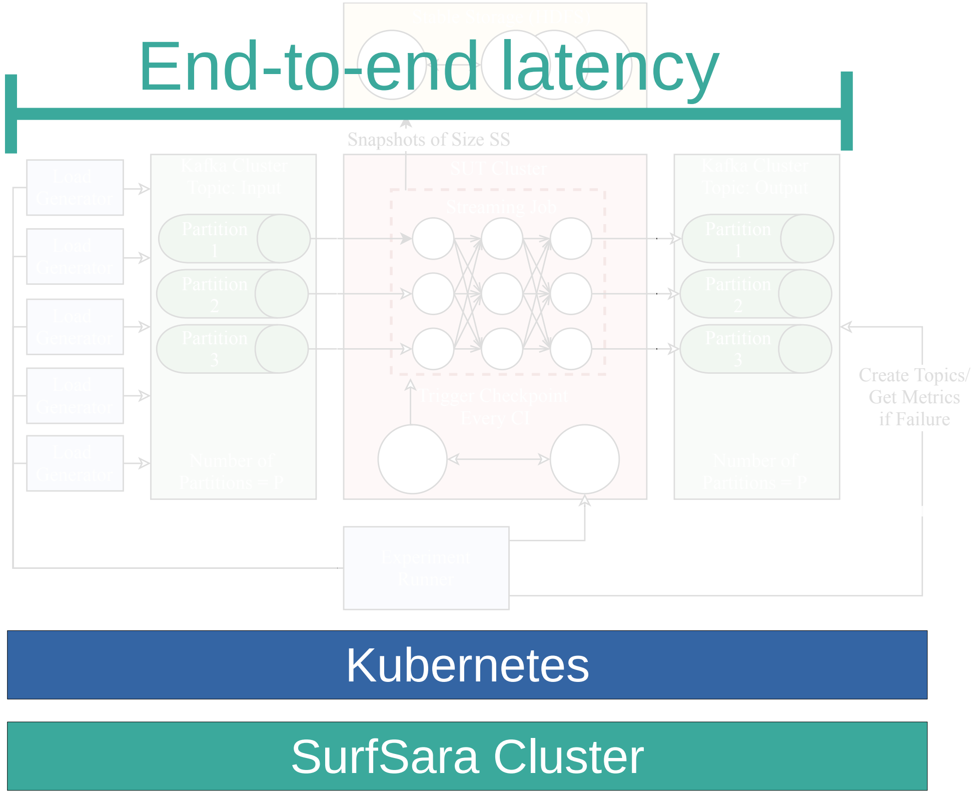


- Spillable In-Flight Log
  - Pre-fetching during replay



Evaluation

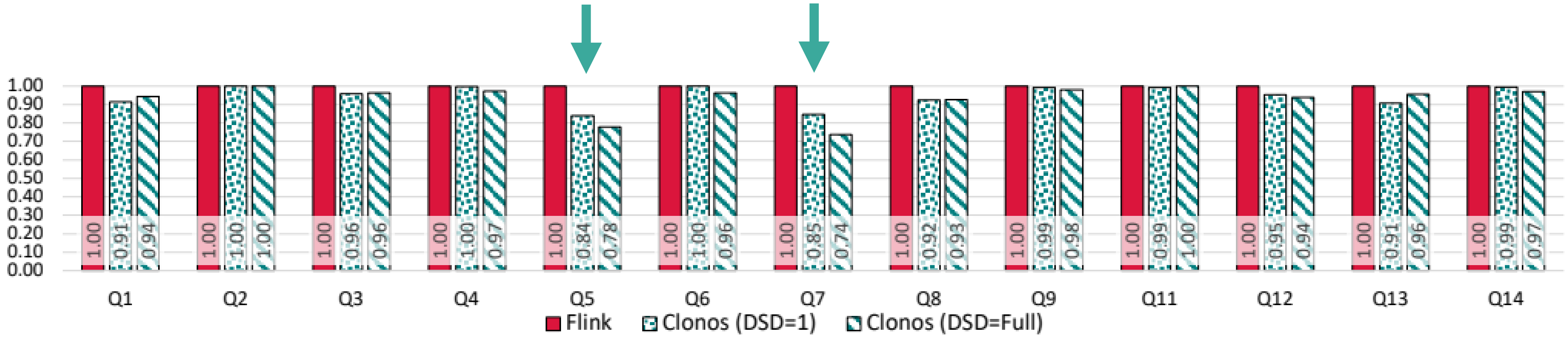
- Kubernetes hosted on SurfSara cluster
- Compare Clonos to Flink (SUTs)
- Analyse both performance overhead and recovery
  - NEXMark and Synthetic
- Measure end-to-end latency and real-time throughput



NEXMark - Throughput overhead

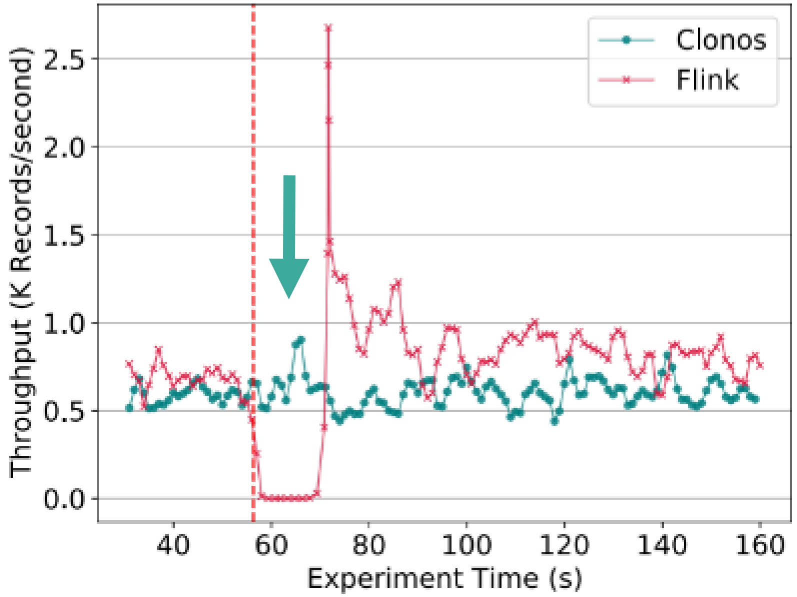
- P=25, D in 1-6: 25-150 hosts
  - DSD=Full: **~7%** avg. degradation (26% max)
  - DSD=1: **~5%** avg. degradation (16% max)

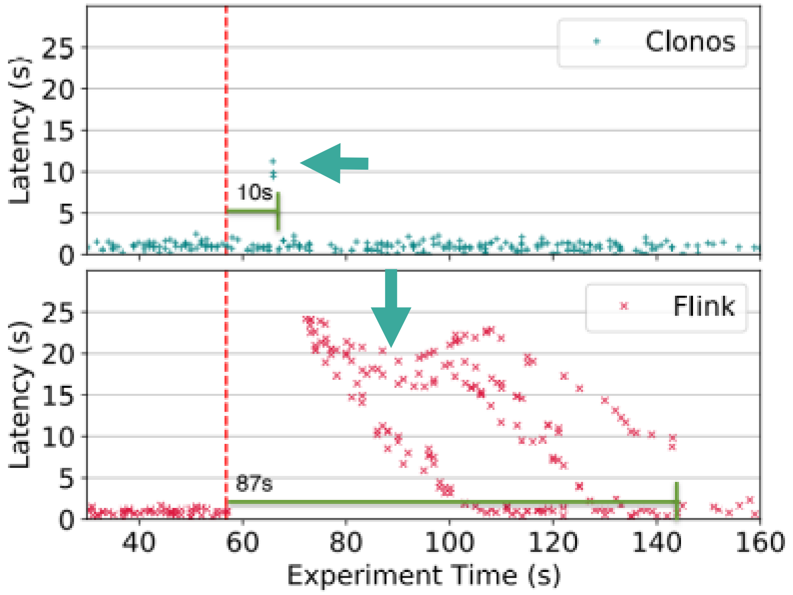
Potential for further optimization



Recovery – NEXMark Q3

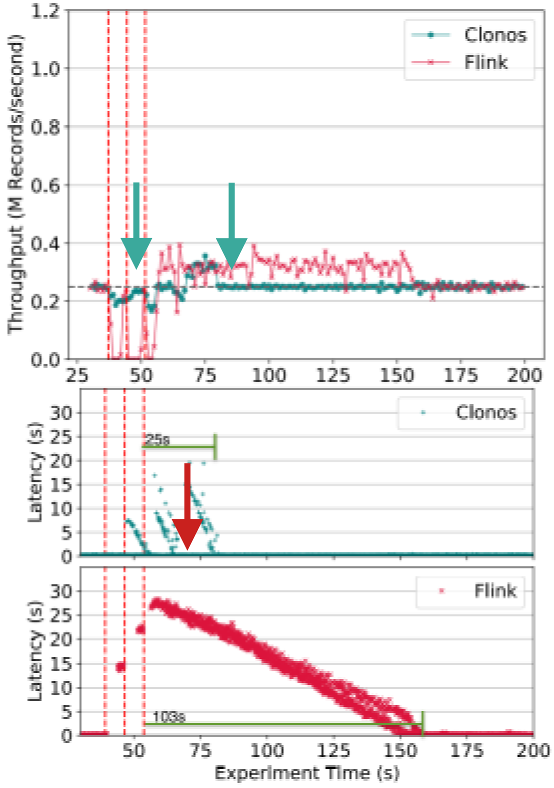
```
SELECT Istream(P.name, P.city, P.state, A.id)
FROM Auction A [ROWS UNBOUNDED], Person P [ROWS UNBOUNDED]
WHERE A.seller = P.id AND (P.state = 'OR' OR P.state = 'ID' OR P.state = 'CA') AND A.
    ↪ category = 10;
```



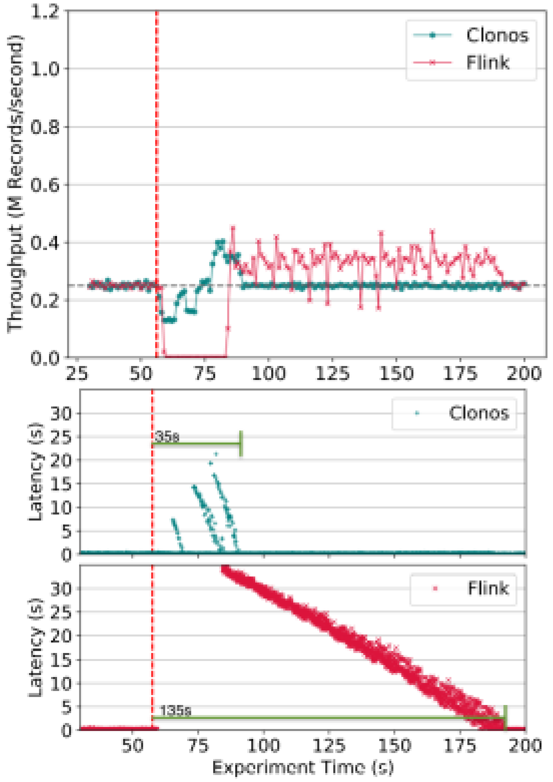


Recovery - Synthetic

- Pass-through
  - P=5,
  - D=5,
  - SS=100MiB,
  - CI=10s
- Fail 3 connected tasks



Multiple



Concurrent

Clonos

- **Clonos**[1] is:
  - **Consistent**: Exactly-once processing guaranteed
  - **Performant**: Cost of ~5% throughput on realistic workloads
  - **Highly-Available**: Up to 10x faster non-blocking local recovery
  - **Expressive**: Supports all streaming Apache Flink workloads
  - **Configurable**: Adjustable guarantees and resource overhead
  - **Practical**: Causal services, spillable in-flight log

[1] Available at https://delftdata.github.io/clonos-web/