FP4S: Fragment-based Parallel State Recovery for Stateful Stream Applications

Pinchao Liu, Hailu Xu, Dilma Da Silva‡,
Qingyang Wang†,Sarker Tanzir Ahmed‡, Liting Hu

Florida International University
Louisiana State University†
Texas A&M University‡

This work is supported by the National Science Foundation (NSF) under NSF-SPX-1919126.

Question Contacts:
lhu@cs.fiu.edu

FIU | Computing & Information Sciences

ELVES research laboratory

Problem Description

ELVES research laboratory

Stateful Streaming Application

A **stream** is an unbounded sequence of tuples (e.g., online social network's microblog streams) generated continuously in time.

A **stateful operator** maintains the value of state for some of the records processed so far and updates it with each new input, such that the output reflects results that consider both historical records and the new input.

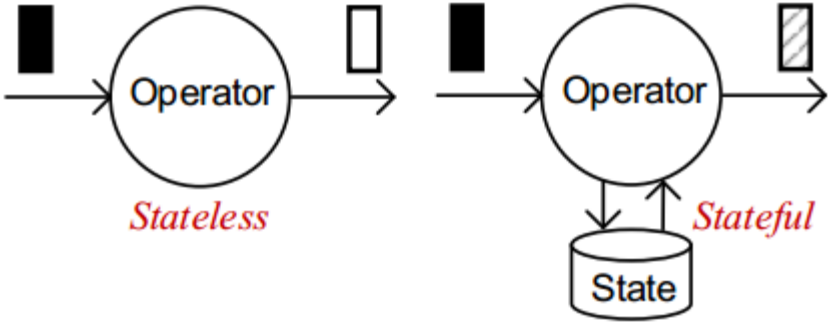


Fig: Contrast of stateless stream processing and stateful stream processing.

Problem Description

ELVES research laboratory

- State Recovery

  It is the process of recovering application states when one or many operators fail or lose their states.

- Failure Recovery Background
  - Stream operators are long-running in which failures and stragglers are inevitable and difficult to predict.
  - Stream applications may run concurrently on the same platform.
  - Large distributed states must be restored efficiently after node failures.

Challenges

ELVES research laboratory

- How to **scale recovery** with the size of the state, the number of simultaneous failures and the number of concurrently running stream applications on a shared platform?

- How can we handle **many simultaneous failures** while achieving fast recovery and imposing low hardware cost?

System Goals

ELVES research laboratory

- Resource efficient.

  Avoid the replication hardware overhead.

- Fast recovery.

  Avoid the slow recovery of retrieving state from disk and replaying the data input that hurts the service quality of stream applications.

- Resilient to multiple failures.

  The mechanism needs to handle multiple simultaneous failures due to the much higher node dynamics in large clusters.

FP4S System Design

ELVES research laboratory

- the FP4S system consists of three layers
  - The DHT-based consistent ring overlay.
  - The fragmented parallel state recovery mechanism.
  - The high-level FP4S interfaces to the stream processing systems.

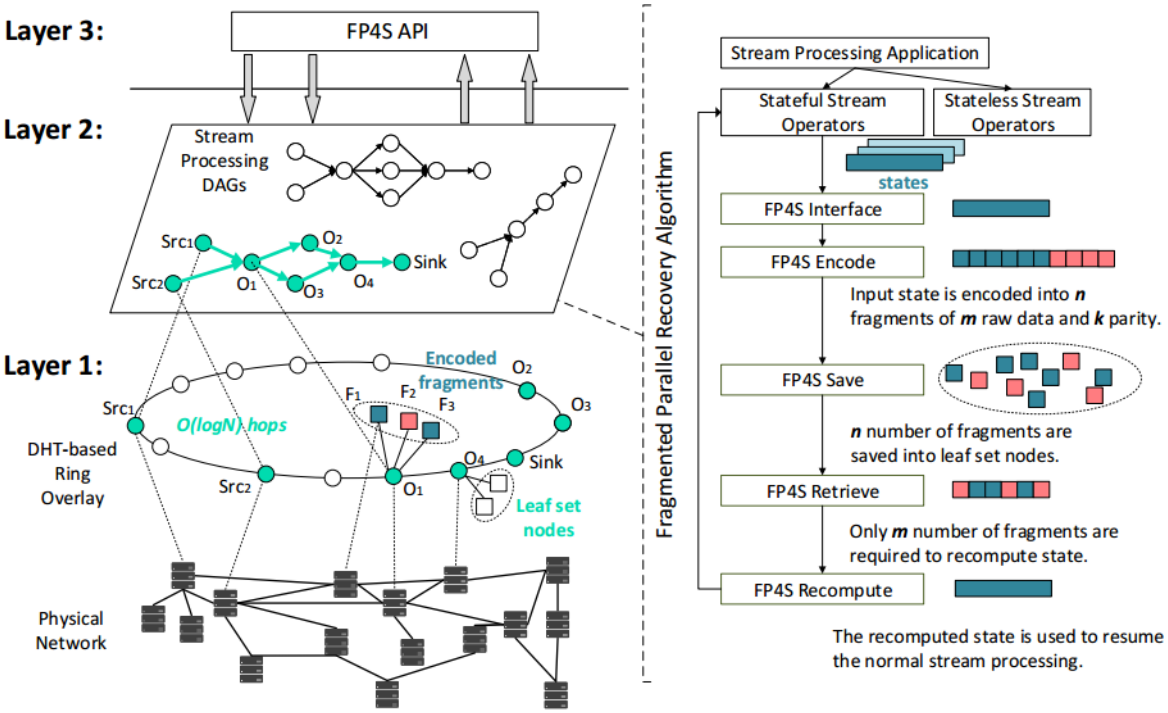


Fig: FP4S system design.

FP4S System Design

ELVES research laboratory

- DHT-based Ring Overlay
  - Leverages DHT-based consistent overlay to support parallel recovery of distributed states for many concurrently running stream applications.
- Fragmented Parallel State Recovery
  - 4 steps process: Encode state → Save state → Retrieve state → Recompute state
  - The right figure shows the steps of the erasure-code-based parallel recovery algorithm.

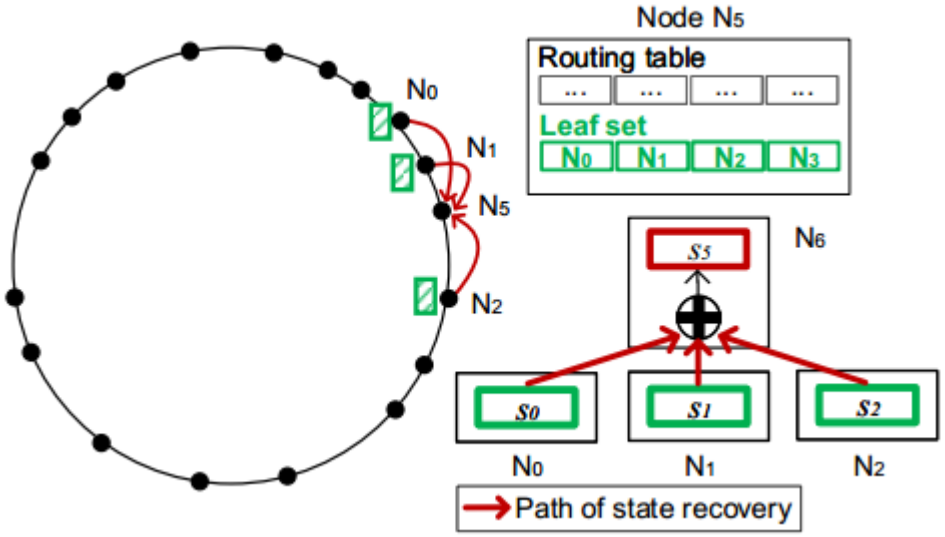


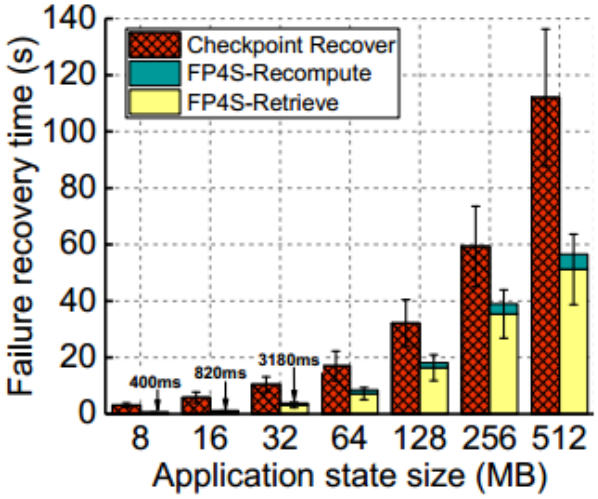
Fig: State recovery process.

Experimental Evaluation

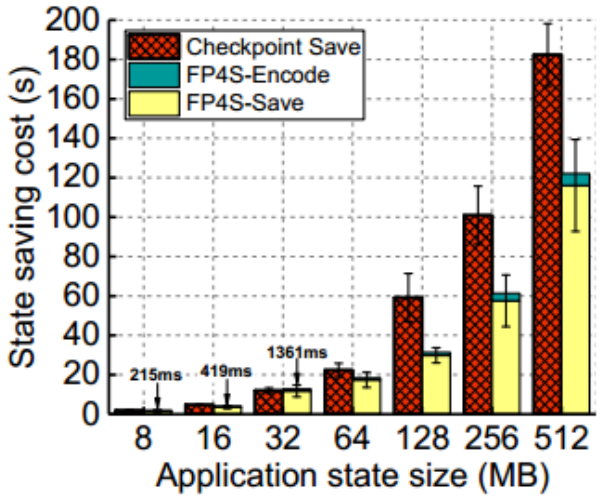
ELVES research laboratory

FP4S vs Checkpointing Recovery

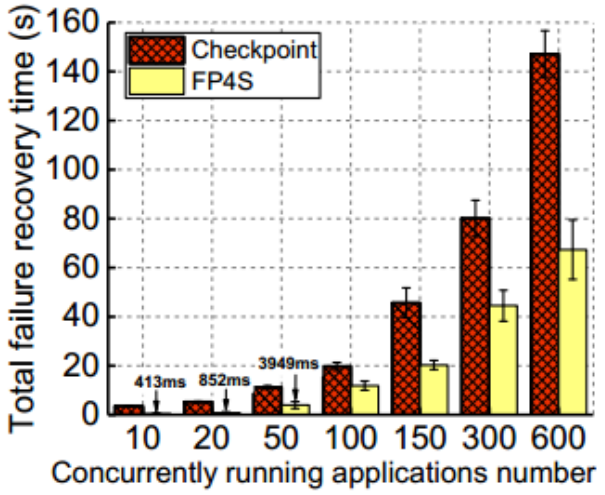
- FP4S achieves **40.3% to 87.1%** less failure recovery time compared to Storm's checkpointing recovery.



(a) State recovery time for different input state sizes.



(b) State saving time for different input state sizes.



(c) Total failure recovery time by varing # concurrently running stream applications.

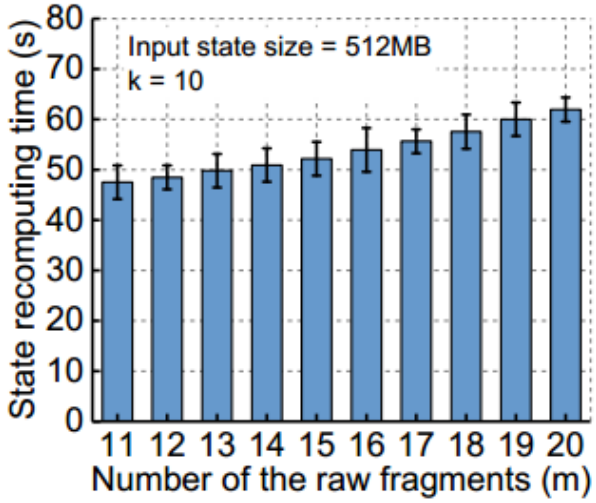
Fig: Performance comparison FP4S performance vs checkpointing strategy for different input states size and applications number.

Experimental Evaluation

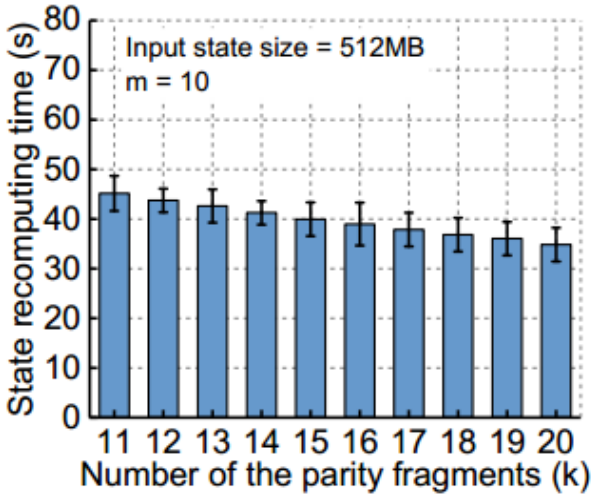
ELVES research laboratory

Fragmented Parallel Recovery Algorithm

- Evaluate the factors affect the recovery performance, including the number of the **raw fragments m** in a state, the number of the **parity fragments k** in a state, the number of **unavailable blocks e** in a state and the amount of leaf nodes.



(a) Adjust raw fragment (m) parameter.



(b) Adjust parity fragment (k) parameter.

(c) Adjust unavailable block number (e).

Fig: The recovery performance evaluation by adjusting number of raw fragments, number of parity fragments, and number of unavailable blocks.

Conclusion & Future Works

ELVES research laboratory

- Conclusion
  - FP4S leverages DHTs and erasure codes to divide each operator's in-memory state into many fragments and periodically save them in each node's leaf set nodes in a DHT ring.
  - Different sets of available fragments can reconstruct failed state in parallel.
  - Scalable to the size of the lost state.
  - Reduces the failure recovery time and can tolerate many simultaneous operator failures.

- Future works
  - Pipeline them to speed up the recovery process.
  - Adjust the ratio of m and n to tradeoff the reduced upload data and the increased computation complexity.
  - Provide a theoretic model to estimate network I/O cost and computation complexity.