TEXAS A&M UNIVERSITY
Department of Computer Science & Engineering

Advancing Streaming Analytics for Cybersecurity

Dilma Da Silva

dilma@cse.tamu.edu

Outline

- TAMUS OnRamp CAMPS
- Streaming frameworks overview
- Streaming on a cybersecurity lens
  - Edge deployments
  - Stateless versus stateful operators
- Educational challenges and opportunities

OnRamp CAMPS

Collaboration for Advancing Minority Participation in Security (CAMPS)

- Texas A&M University
- Prairie View A&M University
- Texas A&M San Antonio
- Texas A&M Corpus Christi
- West Texas A&M University

Research Mission – started January 2021

- **Texas A&M University**
- **Prairie View A&M University**

Stream Processing

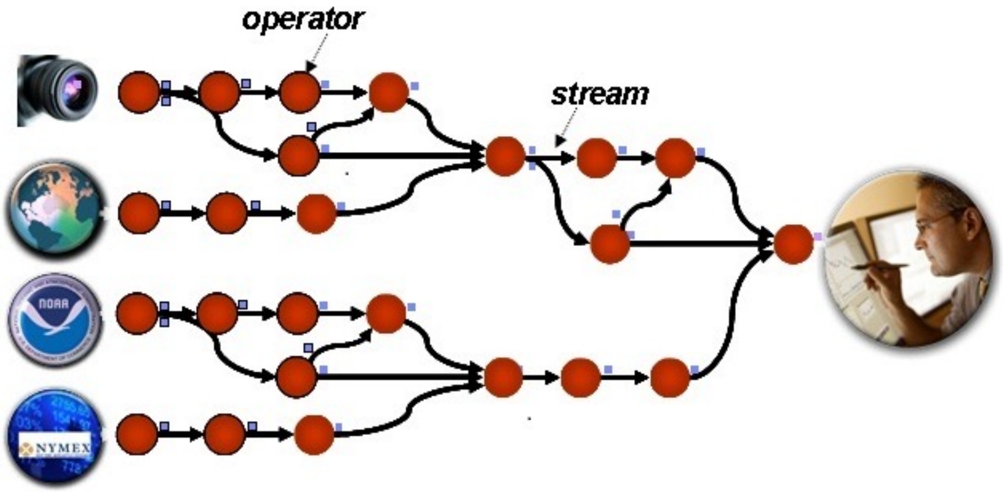
query and process

continuous, long-running, and large-scale data streams

within a short period of time

~ 2008 - IBM System S

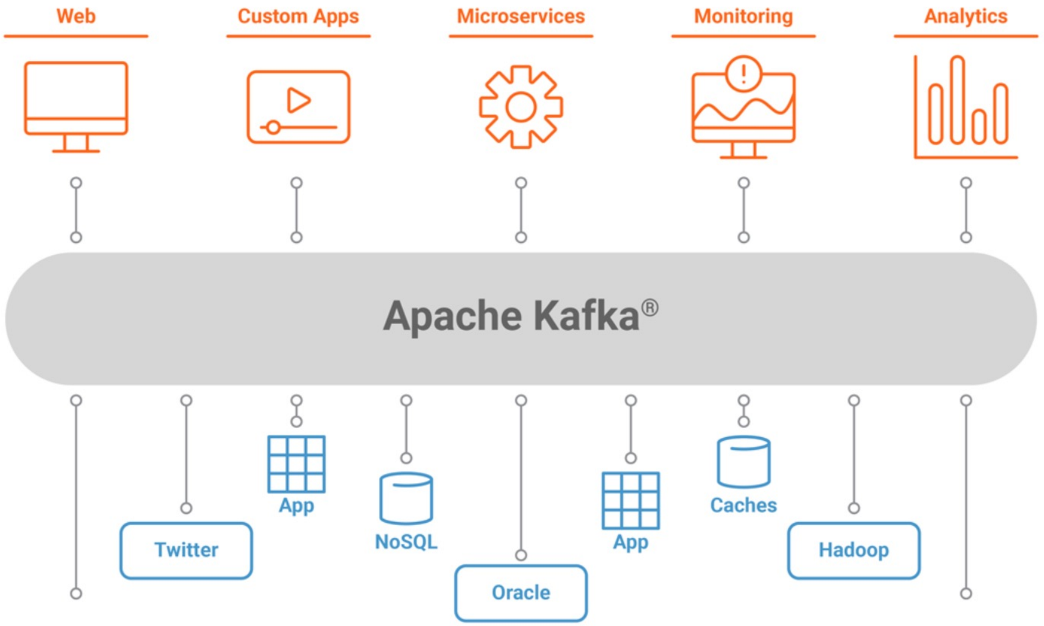
~ 2010 – Yahoo S4



Beyond a computing paradigm:
new requirements in
application architecture
and
cluster management

Pub-subscribe developed at LinkedIn

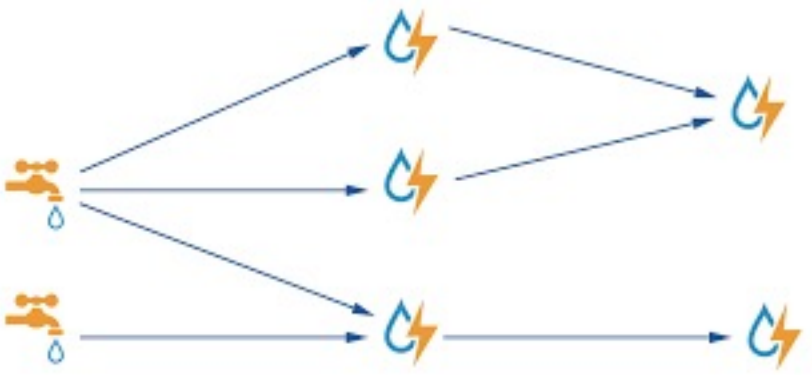
Open-sourced in 2011



https://docs.confluent.io/5.5.1/kafka/introduction.html

APACHE STORM™

- Open-sourced by Twitter
- Initial release in 2011; Apache top-level project in 2014
- SIGMOD’14



Apache Heron

A realtime, distributed, fault-tolerant stream processing engine

Twitter replaced Storm with Heron [SIGMOD'15] to meet Tweeter's new requirements:

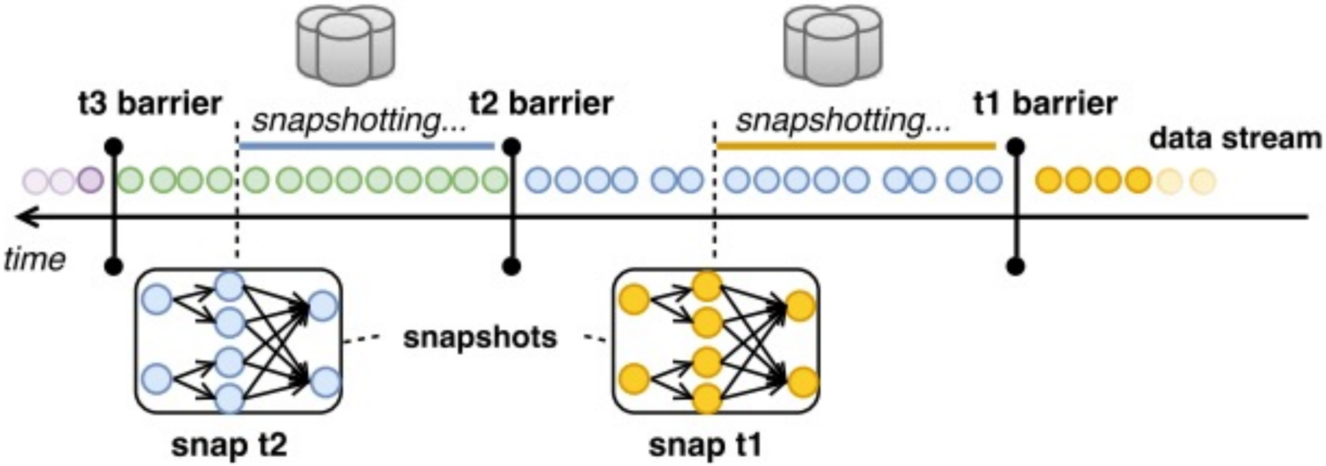
> "billions of events per minute;
> have sub-second latency and predictable behavior at scale;
> in failure scenarios, have high data accuracy;
> **resiliency** under temporary traffic spikes and pipeline congestions;
> be **easy to debug**;
> and simple to deploy in a shared infrastructure."

TEXAS A&M UNIVERSITY
Department of Computer
Science & Engineering

Apache Flink

Stateful applications

– "State management in Apache Flink: consistent stateful distributed stream processing" VLDB'17



samza

Created at LinkedIn

VLDB'17

Supports **stateful applications**

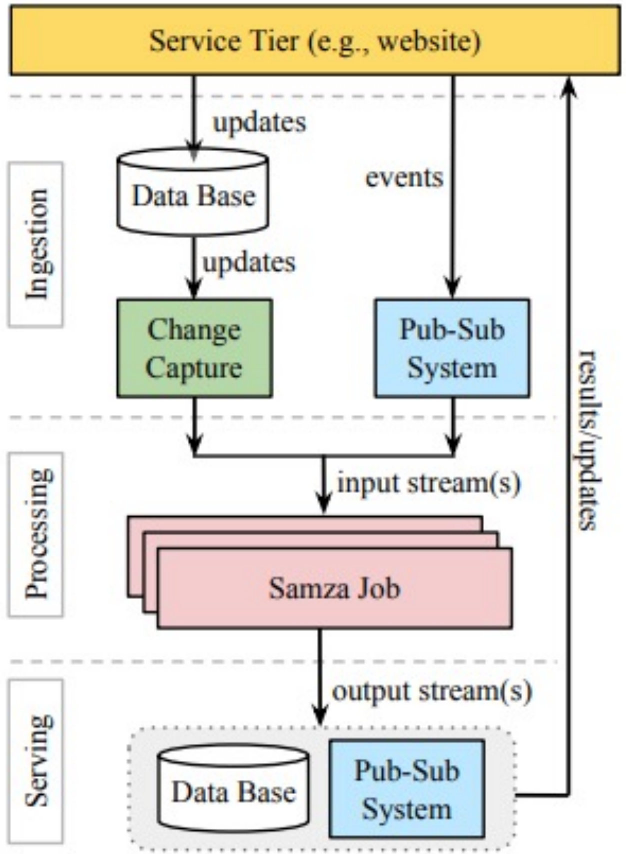


Figure 2: Stream processing pipeline at LinkedIn.

	Storm	Heron	Flink	Samza
Version	2.2	0.24	1.1	1.5
Used at	Groupon, Twitter, Weather Channel, Yahoo!, Spotify, Alibaba, Baidu, Yelp, WebMD, and MANY more	Twitter	Alibaba, AWS, Capital1, Ebay, Ericsson, Lyft, Uber, Yelp, Huawei and a few more	LinkiedIn, Intuit, Slack, TripAdvisor, Netflix, Tivo, and a few more

In this project, focus on Apache Storm

Cybersecurity work with Apache Storm

- When released as an Apache project, the community collaborated with Yahoo!, Hortonworks, and Symantec to address Storm’s security features.
- 8 entries in the CVE (Common Vulnerabilities and Exposures) list.
- Publications on using Apache Storm for network intrusion detection.

Uncovered needs of emerging cybersecurity analytics

- Deployment – from datacenter (cloud computing)
  to infrastructure closer to data sources (edge computing)
- Direct data source integration
  - Flexible, efficient, authenticated, at scale
  - Enablement of plug-in engine for integration with industrial IoT-based applications
    - Anomalies that capture digital tampering/compromising
- Support for provenance tracking
- Optimizations of federated learning

Focus of OnRamp project

- **Deployment** – from datacenter (cloud computing) to infrastructure closer to data sources (**edge computing**)
- **Direct data source integration**
  - **Flexible, efficient, authenticated, at scale**
  - Enablement of plug-in engine for integration with industrial IoT-based applications
    - Anomalies that capture digital tampering/compromising
- Support for provenance tracking
- Optimizations of federated learning

Activities of focus in this project

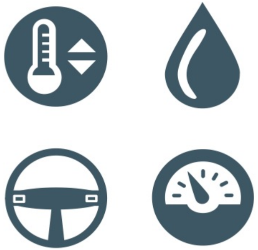
- Stateful operators
- Lightweight flexible ingestion of data
- Benchmarking framework
- Student training

stateful operators

- Stateless vs stateful operators
  - Previous work has augmented Apache Storm with a design that allows scalable state recovery
    NSF Awards 1919126 and 1919181
    Published at IEEE IPDPS'20 and ACM Middleware'21
  - The existing prototype has not been assessed for security implications
- Design to avoid combining additional frameworks
  - E.g., do not use Kafka to integrate data streams
- Exploration: lightweight addition to the Zookeeper component

Stream computing meets edge computing

IoT Sensors
& Actuators



Edge Gateways
& Routers

Real-time
analysis
capabilities

Cloud for
Central Analytics

41.6 billion IoT devices
79.4 zettabytes (ZB) data

PROBLEMS?

- The high latency may cause the results to be obsolete
- The network infrastructure may not afford the massive data streams

data connection component (spouts)

- The literature on advancing Apache Storm is reach on optimizing the placement of the operators.
- Recent efforts address streaming on the edge for IoT workloads: EdgeWise (USENIX ATC 2019), DART (USENIX ATC 2021)
- Exploration: New implementations of the `ISpout` an `IBolt` interfaces
  - optimize spout instantiation: scale-out
  - enable dynamic association between spouts and operators that adapts to variations in input data patterns.

Evaluation infrastructure

- Benchmark setup:
  - emulation of ‘smart building’ application with sensors of varying granularity
  - Documented deployment on cluster of small servers
  - Deployment on container-based virtualization platform
- Components:
  - Design of a Distributed- Hash Table (DHT)-based routing from spouts to bolts
  - Implementation/Testing

Student training

- Graduate students recruited for the project did not join
- Training of undergraduate students
  - Juniors/seniors
  - Exposure to distributed systems
    - Reinforcement of computer system knowledge that is valuable for software security and network security
  - IoT-based applications

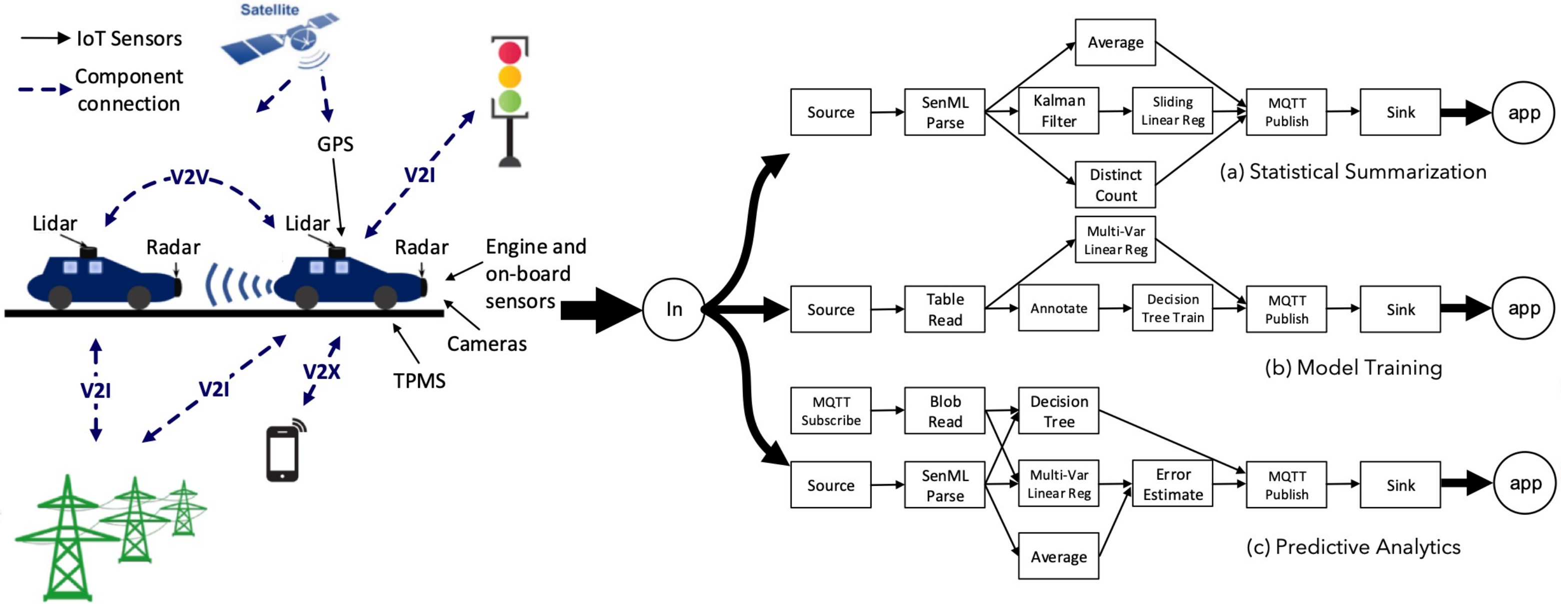
Status

Component	Completion as of 9/21/21
Benchmark setup and documentation	80%
addicionasl benchmark application	30%
Stateful Operator	Analysis - 100%
	Design/Implementation of novel Zookeeper-based approach – 40%
Distributed- Hash Table (DHT)-based routing from spouts to bolts	Design: 100%
	Implementation/Testing: 65%

ENGINEERING
TEXAS A&M UNIVERSITY

Questions ?

Why Edge Stream Processing?



Fleet telematics and Management solutions

Transport logistics applications

Reservation, toll, and ticketing applications

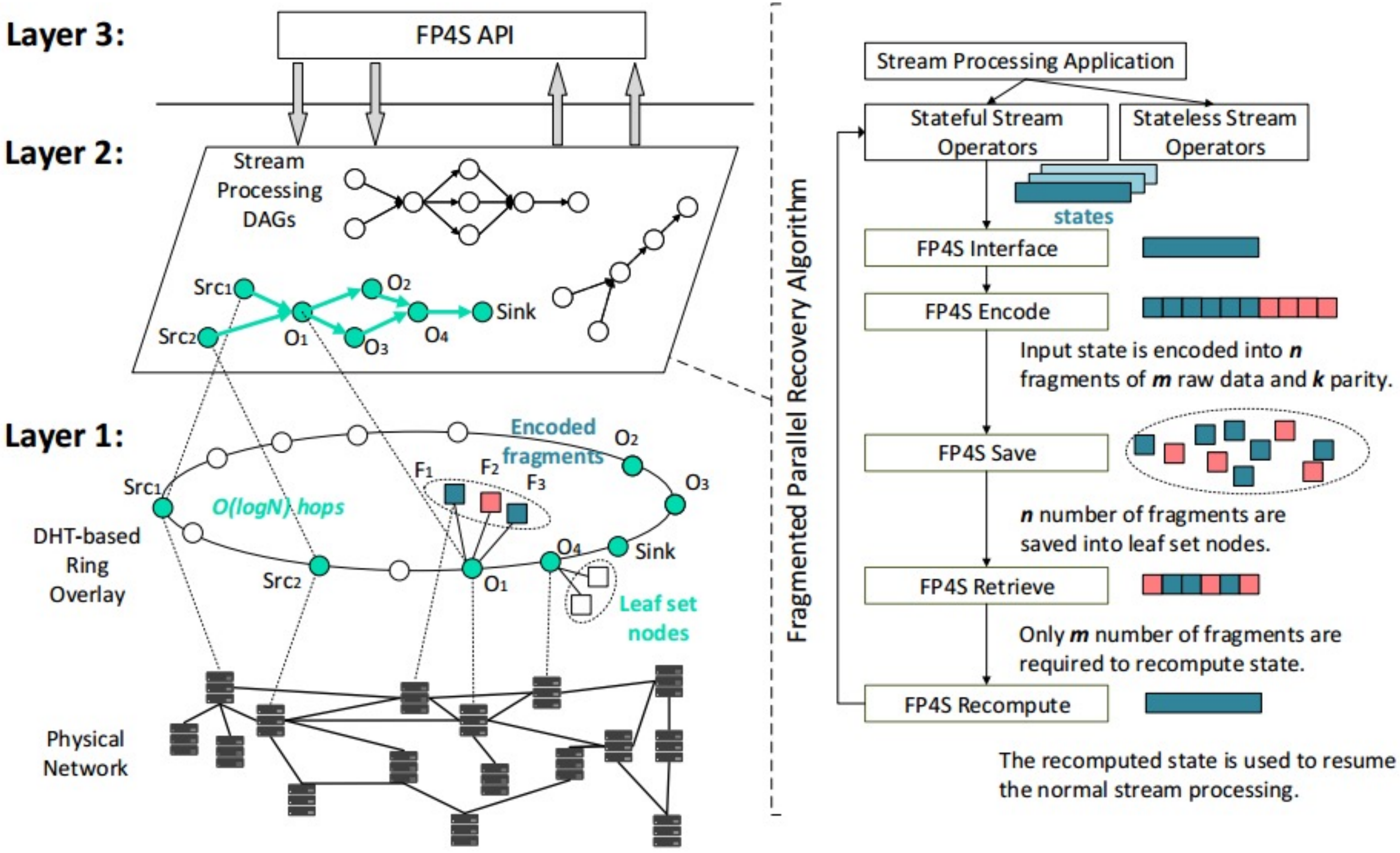
Guidance and control applications

Peer-to-peer services such as parking space finders

Smart vehicle applications

Security and surveillance applications

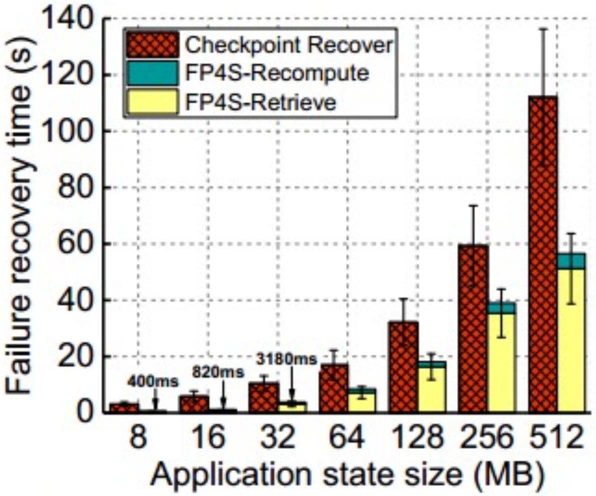
TEXAS A&M UNIVERSITY
Department of Computer Science & Engineering



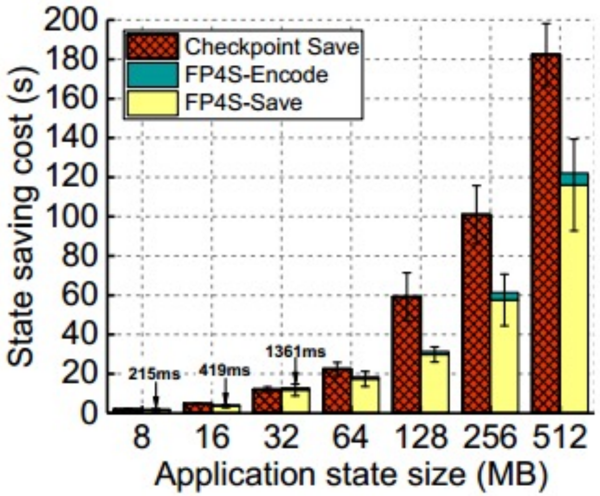
dilma@cse.tamu.edu

FP4S experimental evaluation

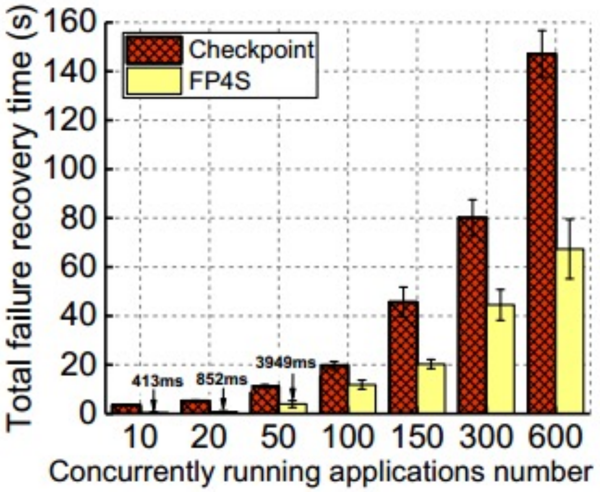
FP4S achieves **40.3% to 87.1%** less failure recovery time compared to Storm's checkpointing recovery.



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

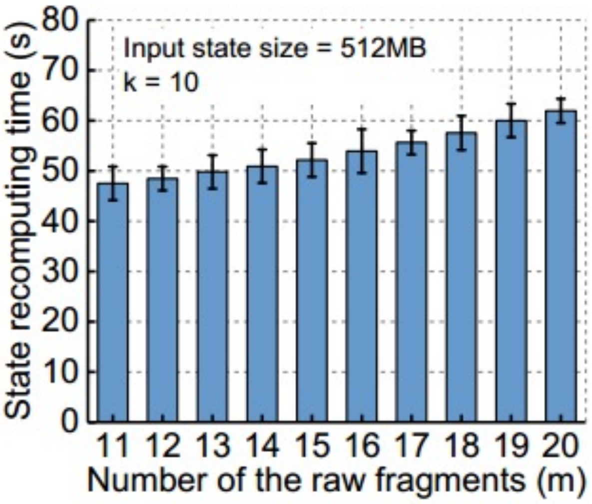


(b) State saving time for different input state sizes.

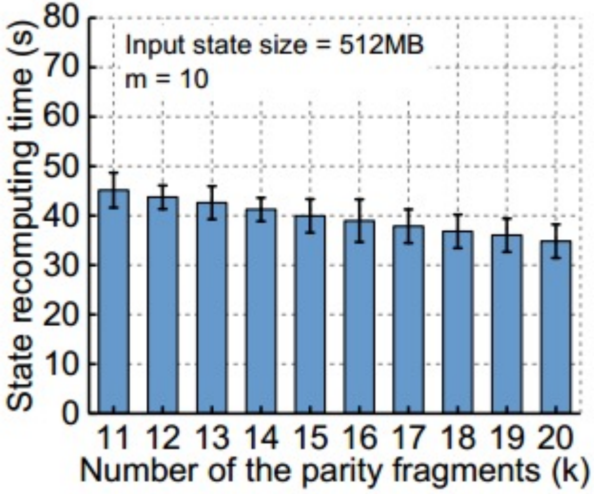


(c) Total failure recovery time by varing # concurrently running stream applications.

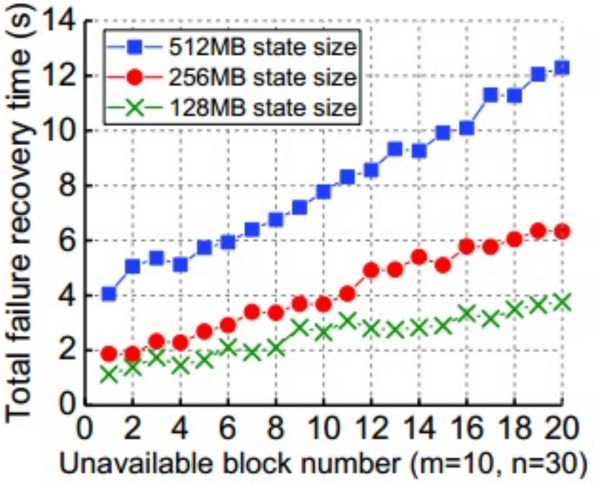
Impact of number of the **raw fragments m** in a state, the number of the **parity fragments k** in a state, the number of **unavailable blocks e** in a state and the amount of leaf nodes



(a) Adjust raw fragment (m) parameter.



(b) Adjust parity fragment (k) parameter.

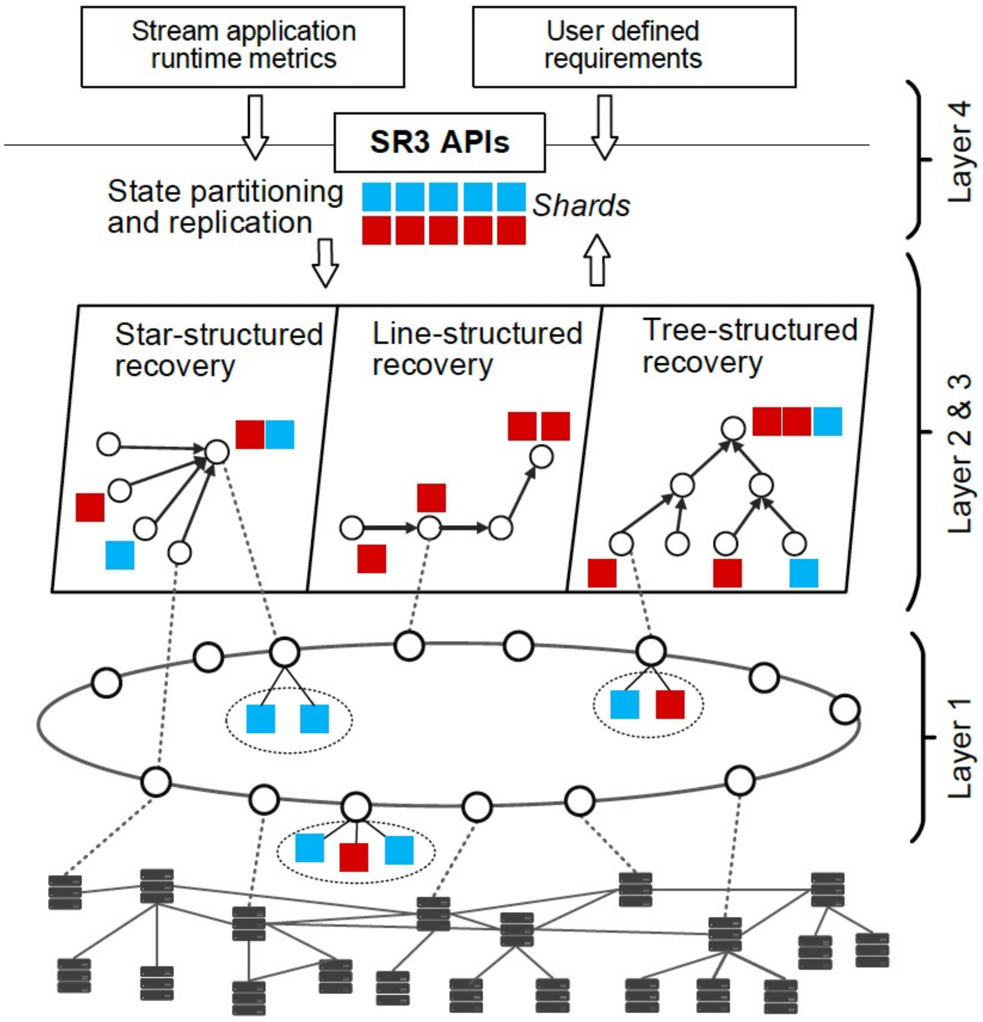


(c) Adjust unavailable block number (e).

SR3 Design:

- L1: DHT-based overlay
- L2: State partitioning and replication
- L3: State recovery
- L4: SR3 API

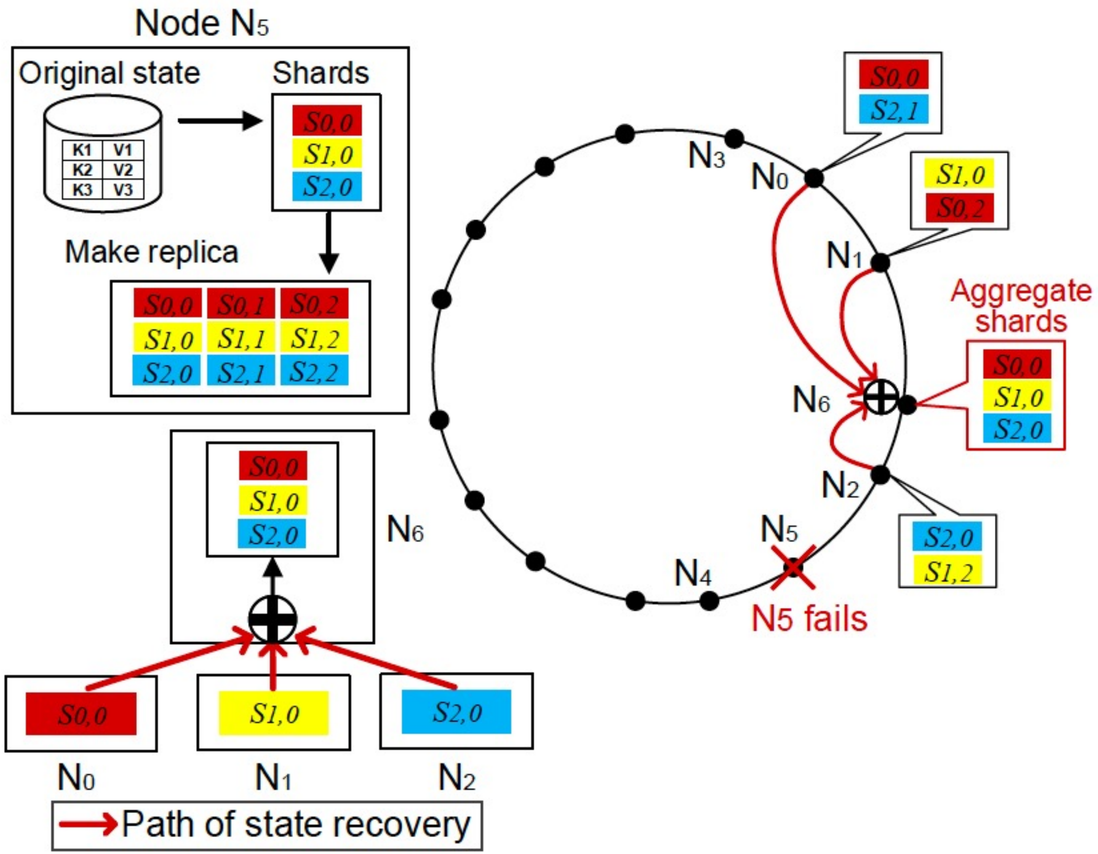
Implemented on top of Apache Storm



Star-structured design

Benefits:

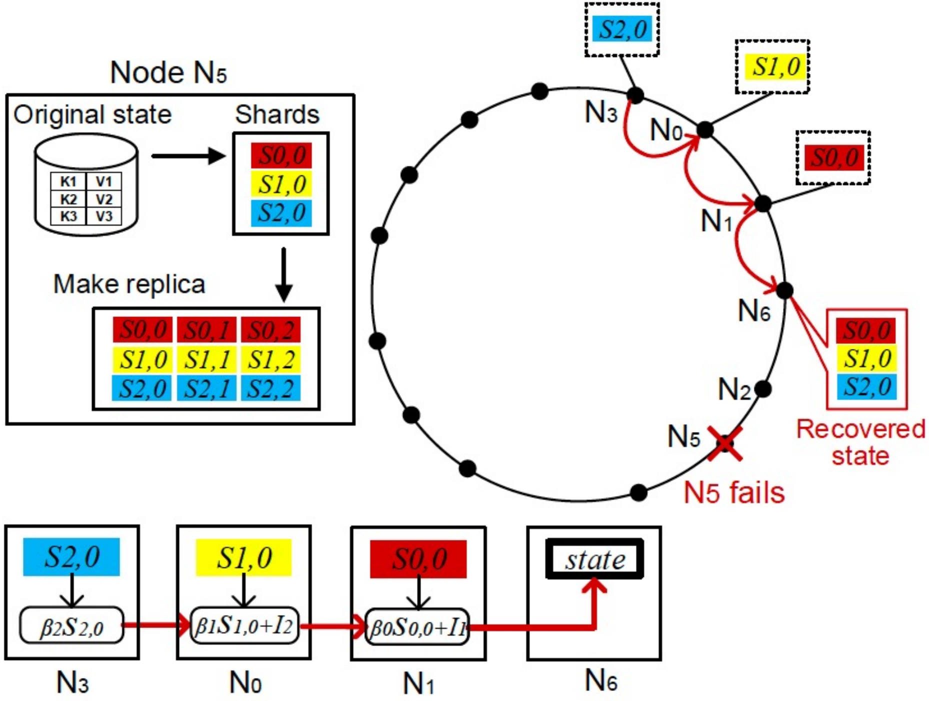
- fast recovery process
- data locality: the leaf set contains nodes that are geographically close to the original node.



Line-structured design

Benefits:

- The downloading and computing load are well balanced among all involved nodes.
- Avoid centralized bottleneck.

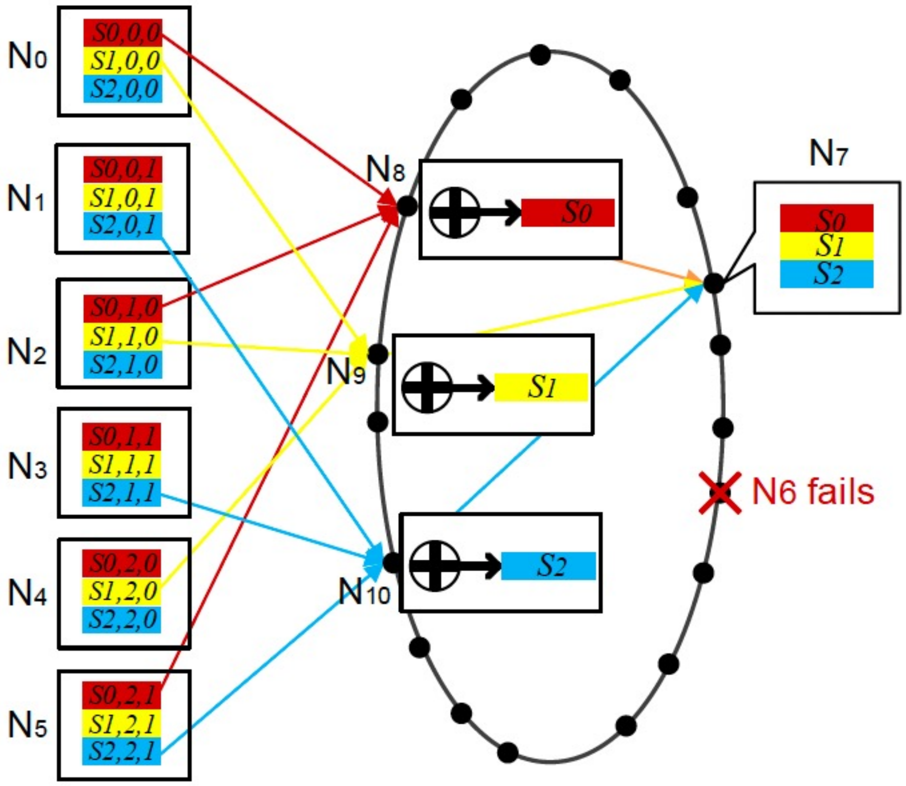


Tree-structured design

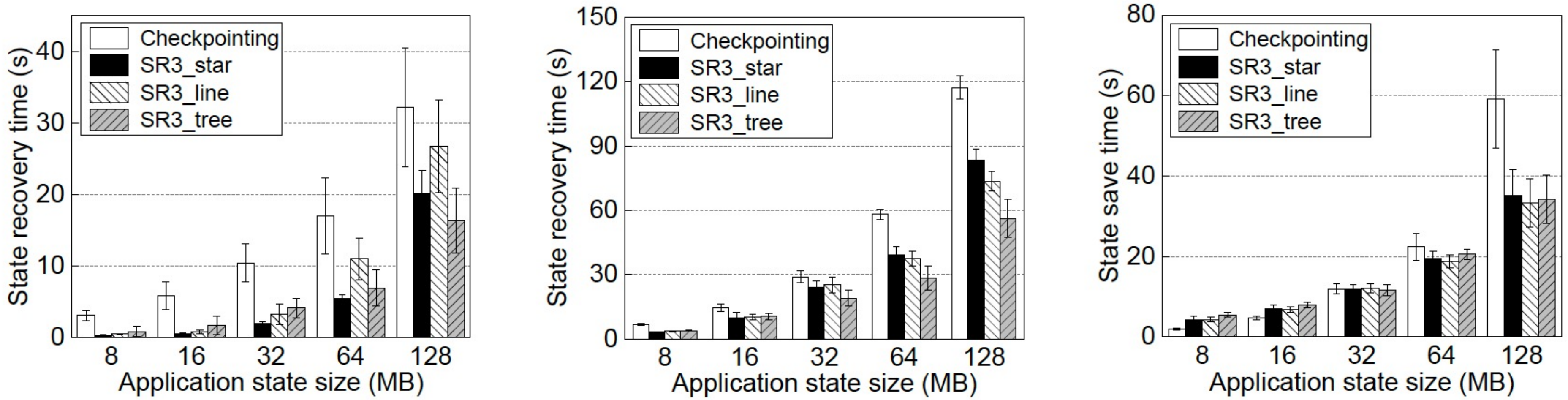
The spanning tree is built on top of a scalable application-level multicast infrastructure.

Benefits:

- A providing node only needs to upload some of the shards it stores.
- The downloading and computing load are well balanced among all involved nodes



Experimental Evaluation

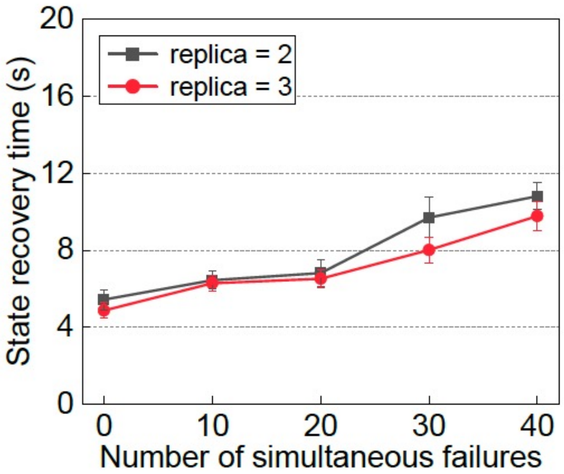


(a) The state recovery time by varying the size of state with no bandwidth constraint.

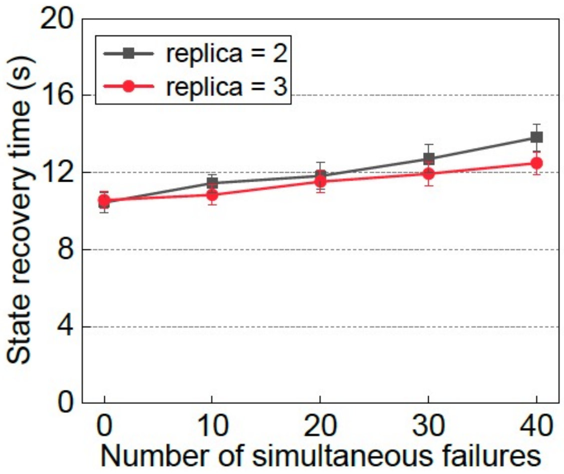
(b) The state recovery time by varying the size of state with bandwidth constraint.

(c) State save time by varying the size of state.

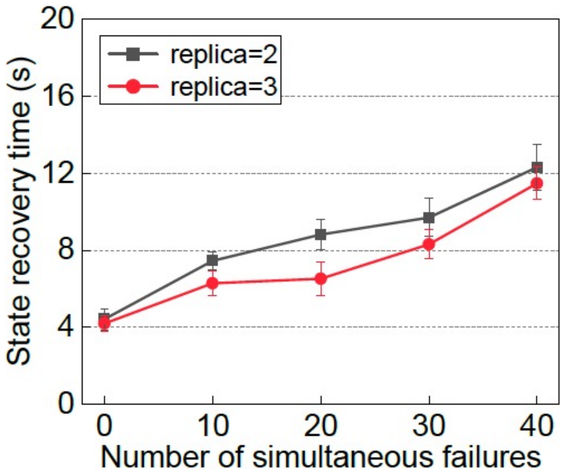
Simultaneous Failures



(a) The state recovery time with failures in SR3 star-structured recovery.

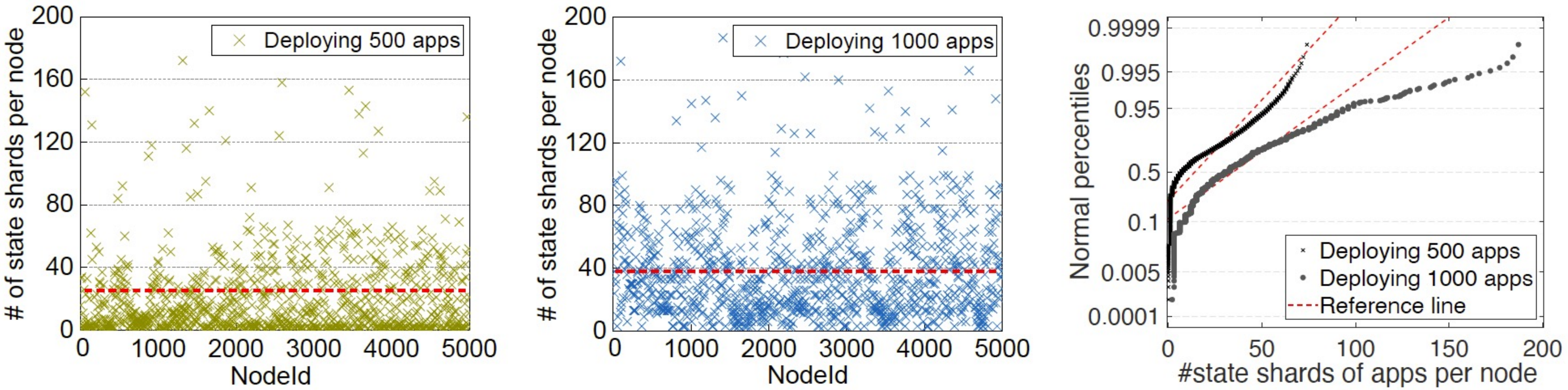


(b) The state recovery time with failures in the SR3 line-structured recovery.



(c) The state recovery time with failures in SR3 tree-structured recovery.

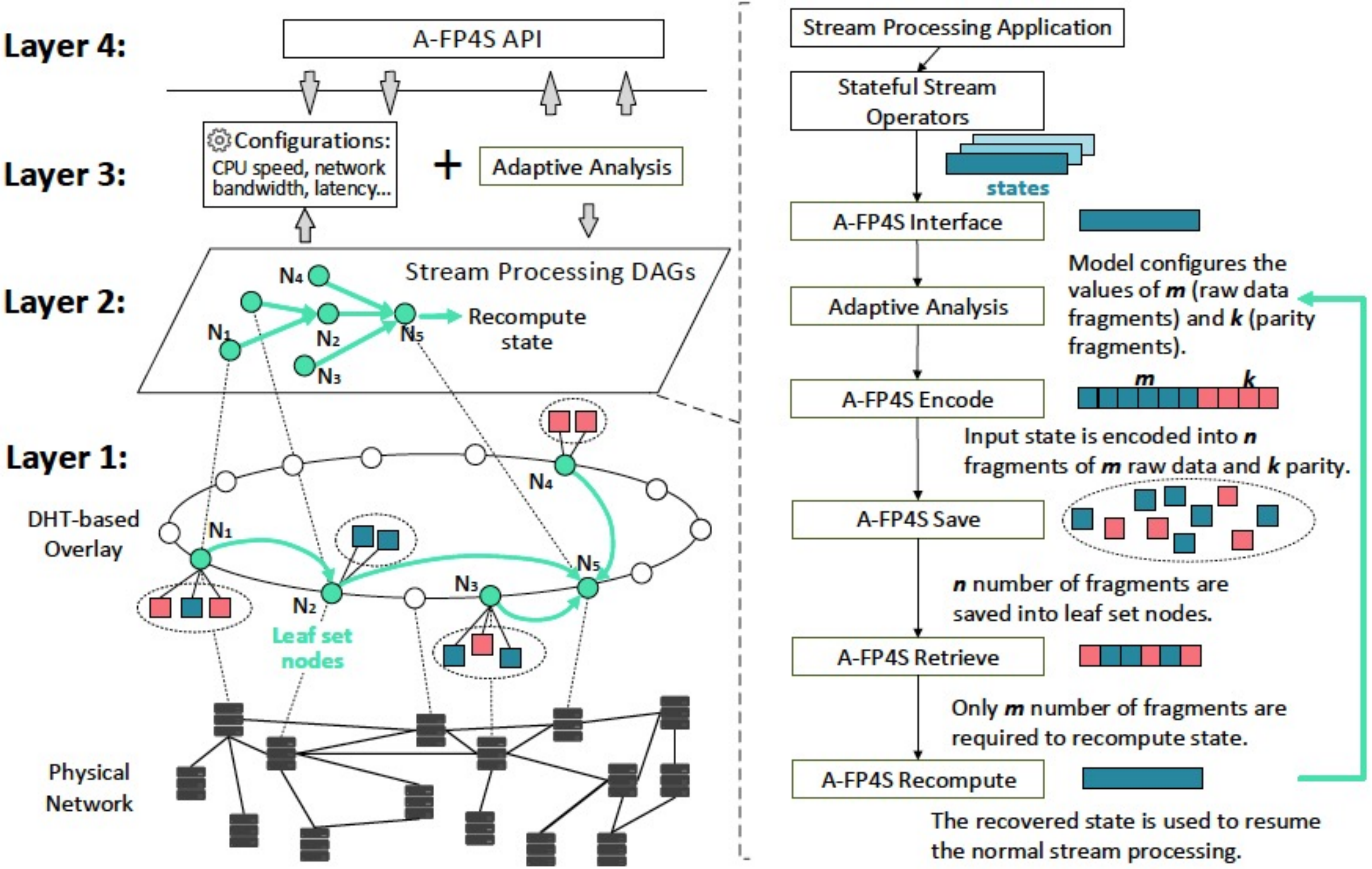
Load balance and scalability



(a) The distribution of state among the overlay when deploying 500 applications.

(b) The distribution of state among the overlay when deploying 1,000 applications.

(c) Normal probability of the number of shards per node.


Layer 4:
A-FP4S API
Layer 3:
Configurations:
CPU speed, network bandwidth, latency...
+
Adaptive Analysis
Layer 2:
Stream Processing DAGs
N_4
N_1
N_2
N_3
N_5
Recompute state
Layer 1:
DHT-based Overlay
N_1
N_2
N_3
N_4
N_5
Leaf set nodes
Physical Network
Stream Processing Application
Stateful Stream Operators
states
A-FP4S Interface
Adaptive Analysis
Model configures the values of *m* (raw data fragments) and *k* (parity fragments).
A-FP4S Encode
m
k
Input state is encoded into *n* fragments of *m* raw data and *k* parity.
A-FP4S Save
n number of fragments are saved into leaf set nodes.
A-FP4S Retrieve
Only *m* number of fragments are required to recompute state.
A-FP4S Recompute
The recovered state is used to resume the normal stream processing.