Topology-Aware Multi-Agent Reinforcement Learning for Efficient Resource Allocation in Cloud-Native Stream Processing

IEEE International Conference on Cloud Computing (CLOUD)
(13-18, July 2026, Sydney, Australia)

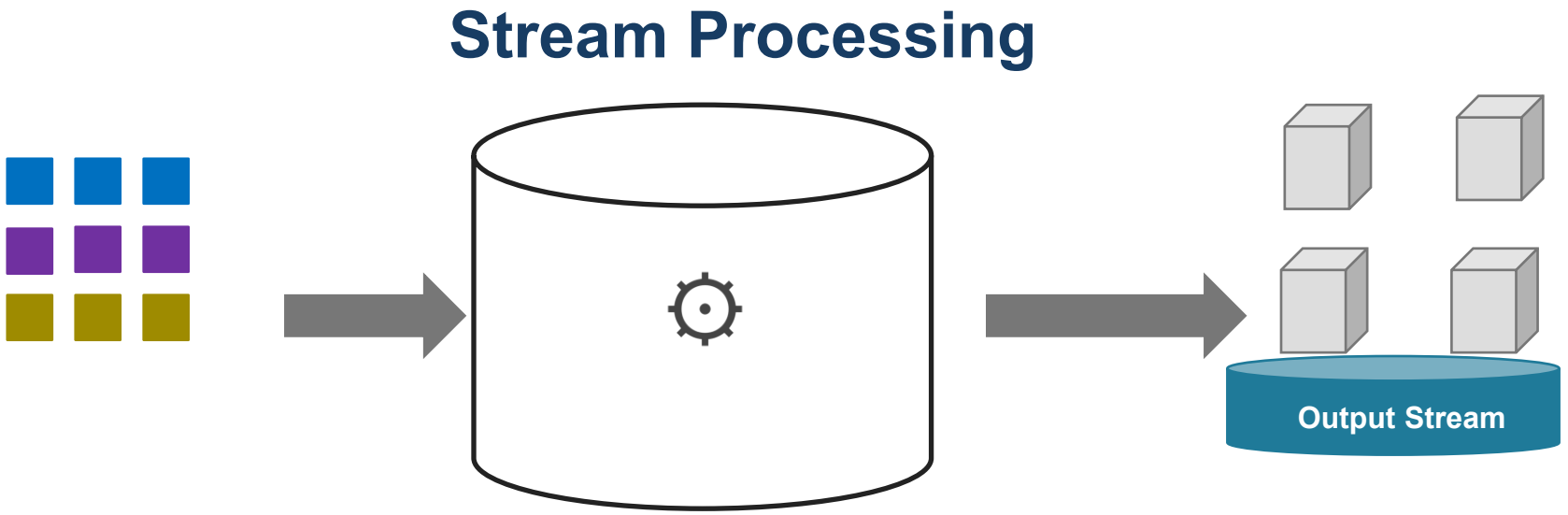
Sunday J. Awine, Jinwei Liu

Department of Computer and Information Sciences
Florida A&M University, Tallahassee, FL, USA

Talk Outline

- **Introduction**
- System Model and Problem Formulation
- Design of H-MADRL
- Implementation
- Performance Evaluation
- Conclusions and Future Work

Introduction



Input Data

Stream Processing Engine

Outputs

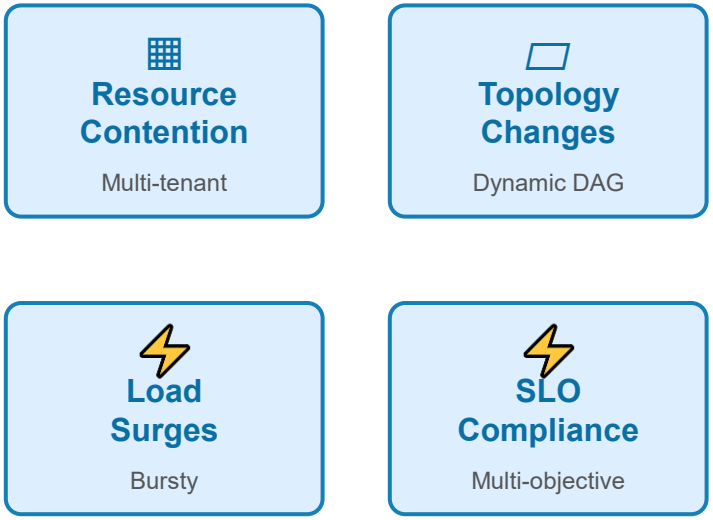
Streaming jobs continuously transform incoming data into real-time outputs while running on distributed cloud resources.

Motivation

- Dynamic workload challenges
  - ➢ Dynamic stream processing faces fluctuating input rates, heterogeneous workloads, and constrained resources
- Resource allocation issues
  - ➢ Static or threshold autoscaling leads to under/over-provisioning, SLO violations, and latency spikes
- Need for intelligent solutions
  - ➢ Need online, topology-aware, proactive resource allocation that balances throughput, latency, locality, and cost in real time

Research Challenges

- Multi-tenancy and node heterogeneity
  - ➢ Resource contention
  - ➢ Node heterogeneity
  - ➢ Isolation requirements
- Evolving DAG topologies
  - ➢ Dynamic operator graphs
  - ➢ Shifting bottlenecks
  - ➢ Dependency complexity
- Bursty and deceptive workloads
  - ➢ Bursty traffic and sudden node pressure



This Research

- **Research Objective:** develop TAMARL, a topology-aware multi-agent reinforcement learning framework for efficient resource allocation in cloud-native stream processing

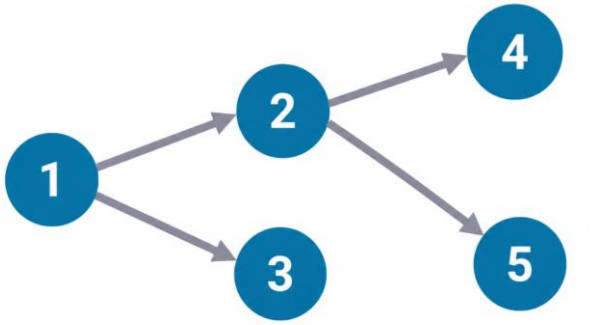
- **Key Contributions**
  - ➢ Topology-aware state representation
  - ➢ Decentralized MARL architecture with CTDE
  - ➢ Multi-objective reward for latency, SLO, locality, and utilization
  - ➢ Cloud-native validation on Kubernetes + Apache Flink
  - ➢ Benchmarking across stress-test scenarios

Talk Outline

- Introduction
- **System Model and Problem Formulation**
- Design of H-MADRL
- Implementation
- Performance Evaluation
- Conclusions and Future Work

System Model

- Stream topology model: a streaming application is modeled as a DAG $G = (V, E)$
  - ➢ $V = \{v_1, v_2, \ldots, v_n\}$: stream-processing operators
  - ➢ E: data-flow dependencies between operators
  - ➢ Each operator has CPU and memory demand $\delta_i = \langle k_i, \mu_i \rangle$

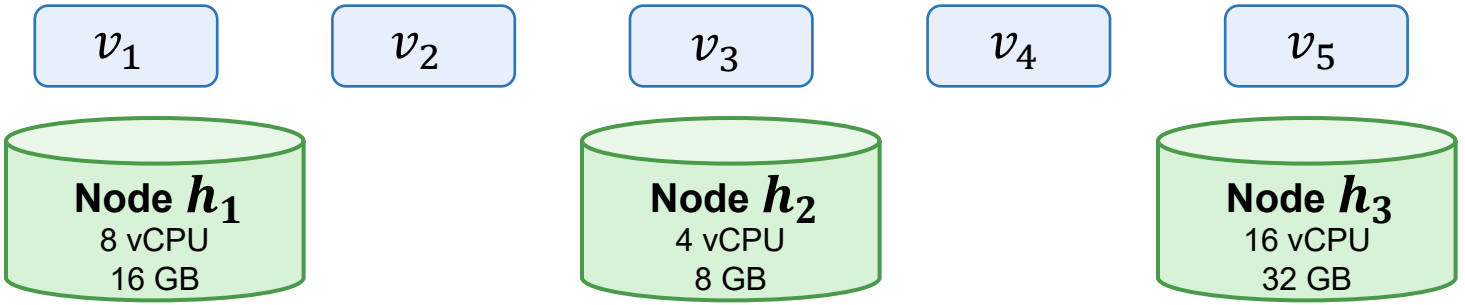


Directed Acyclic Graph (DAG)

DAG awareness lets the scheduler understand which tasks should be kept close to reduce delay and backpressure.

System Model (cont.)

- Heterogeneous substrate: the cluster contains M non-uniform compute nodes $H = \{h_1, h_2, \ldots, h_m\}$.
  - ➢ Each node has limited capacity Ω_k= [CPU, Memory].
  - ➢ Placement variable $y_{i,k} = 1$ if operator v_i runs on node h_k; otherwise $y_{i,k} = 0$.
  - ➢ Constraint: total demand of assigned operators must not exceed node capacity.



Placement chooses the best node for each operator while respecting CPU and memory limits.

Problem Formulation

- TAMARL minimizes a total cost that combines latency, SLO violations, network traffic, and resource efficiency

min Λ = latency penalty + SLO violation penalty + network overhead - resource efficiency

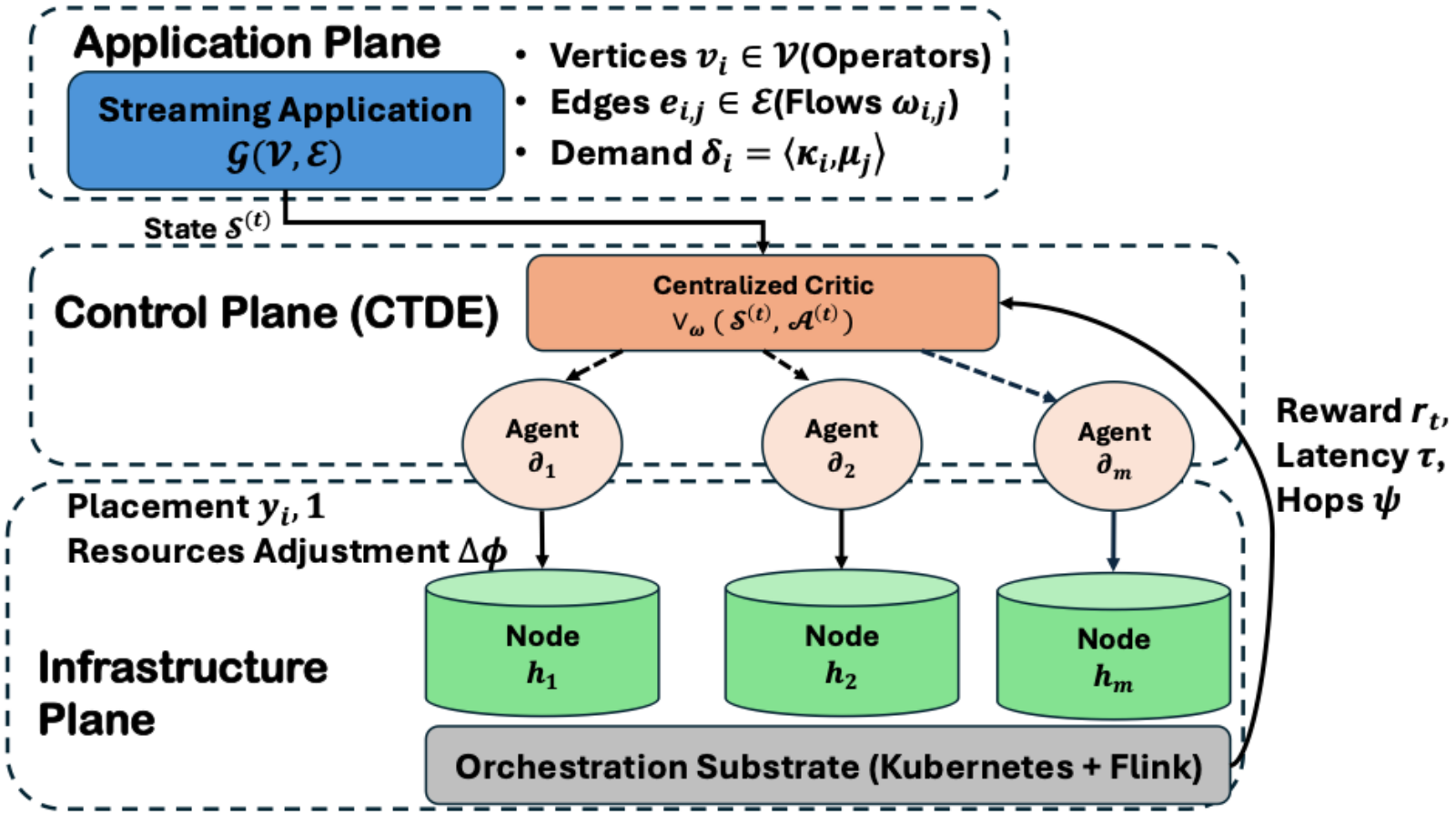
Low Latency	**SLO Attainment**	**Network Locality**	**Efficiency**
Reduce $\mathcal{T}$	Avoid Γ violations	Reduce Ψ	Increase Υ

- **Key idea:** put tasks that exchange heavy data close together, while keeping nodes balanced
  - ➢ If two connected operators are on the same node, network cost is near zero

Talk Outline

- Introduction
- System Model and Problem Formulation
- **Design of H-MADRL**
- Implementation
- Performance Evaluation
- Conclusions and Future Work

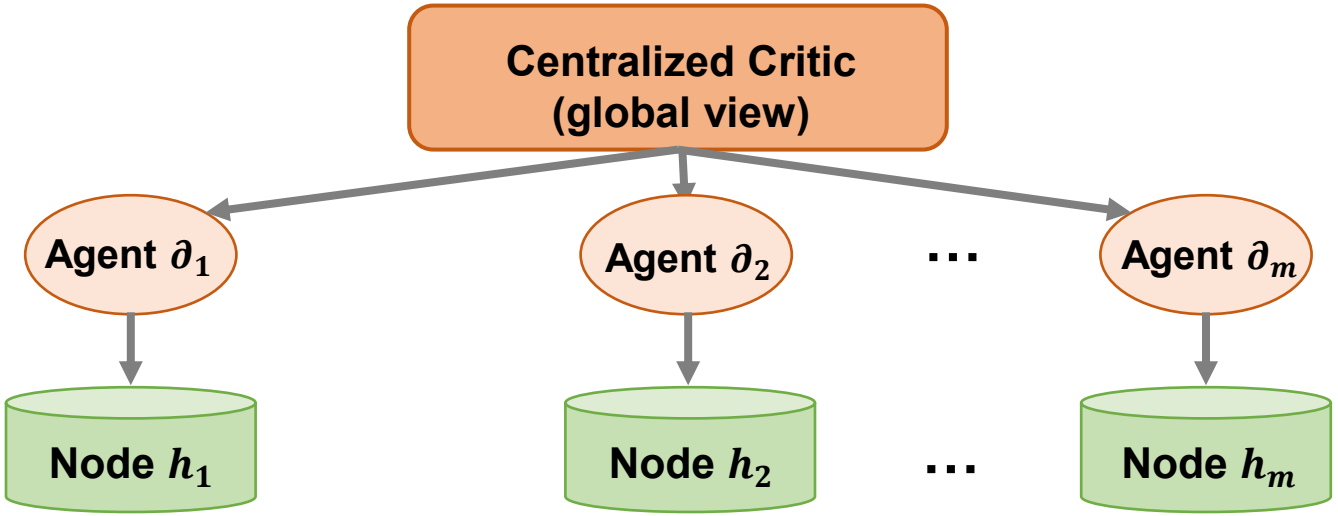
Design of TAMARL



TAMARL maps the application DAG to heterogeneous infrastructure using CTDE-based multi-agent learning.

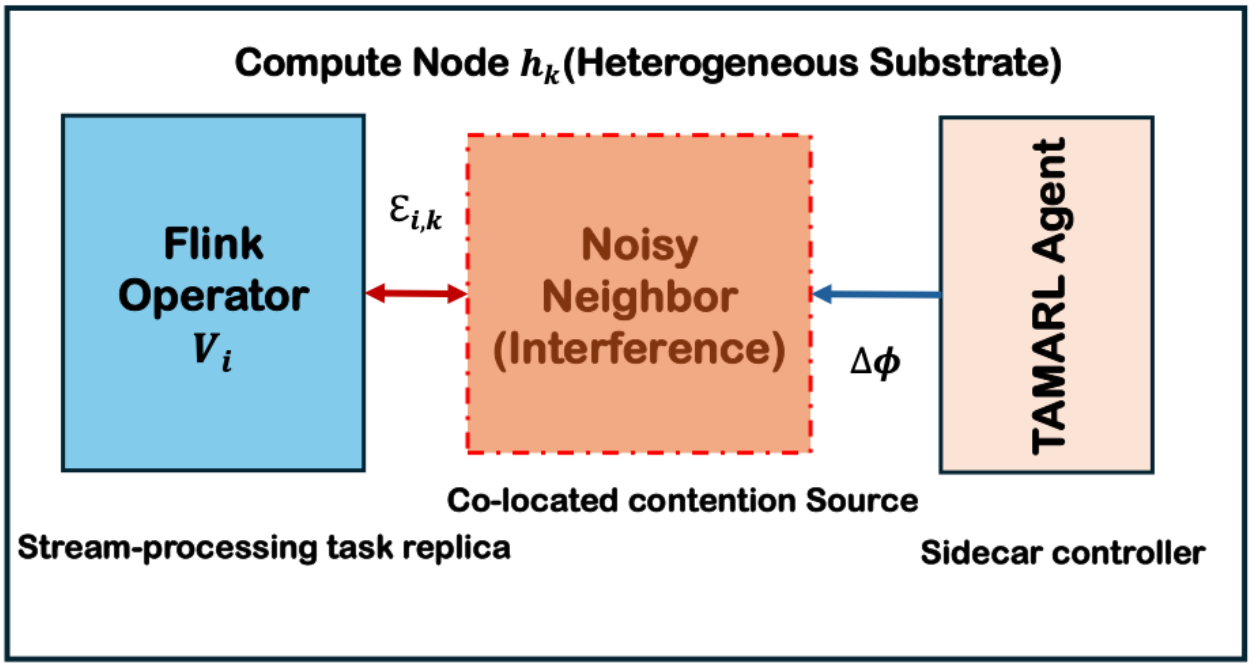
Design of TAMARL (cont.)

- **Centralized Training and Decentralized Execution (CTDE)**
  - ➢ During training, a centralized critic observes the global state and joint action
  - ➢ During execution, each node agent acts locally for faster scheduling decisions



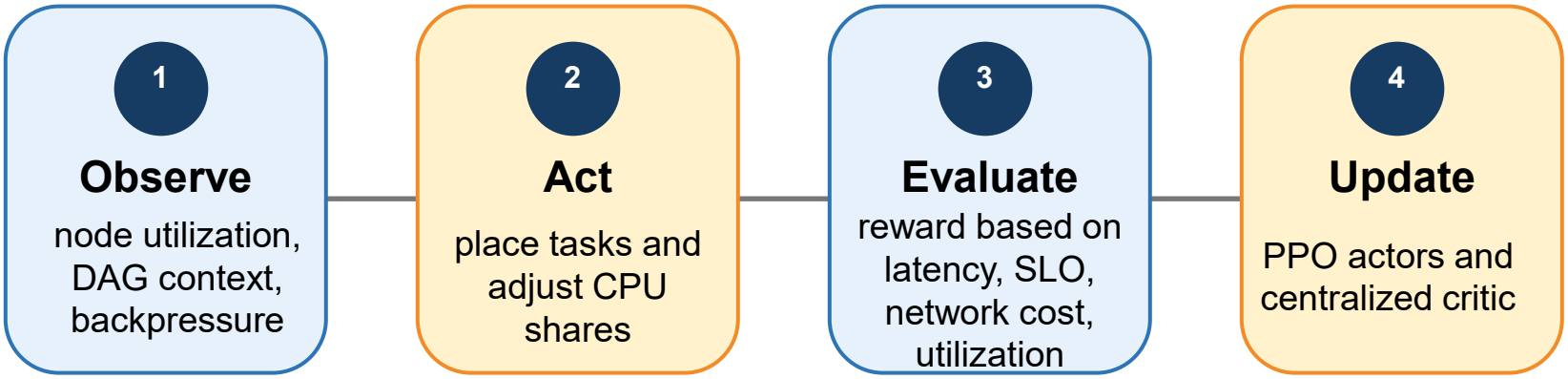
The centralized critic improves coordination during learning; decentralized agents keep execution scalable.

Noisy Neighbor Mitigation



- Problem: co-located containers can create stochastic jitter and degrade operator latency
- TAMARL response: sidecar agents monitor node pressure and adjust CPU/resource shares in real time

Learning Algorithm: TAMARL-CTDE



PPO keeps policy updates stable by improving gradually rather than making large risky changes.

Talk Outline

- Introduction
- System Model and Problem Formulation
- Design of H-MADRL
- **Implementation**
- Performance Evaluation
- Conclusions and Future Work

Implementation

- Software Stack and Integration
  - PyTorch 2.4 and the Ray RLlib library are used to implement the core MARL logic
  - A lightweight Python-based agent installed as a Kubernetes Sidecar container is hosted by each compute node
- Neural Network Architecture
  - A three-layer Multi-Layer Perceptron (MLP) with 256 hidden units and Rectified Linear Unit (ReLU) activation functions makes up each actor network π_θ
  - The centralized critic V_ω uses a Graph Convolutional Network (GCN) layer to estimate the state-value function by processing the global adjacency matrix of the streaming application

Talk Outline

- Introduction
- System Model and Problem Formulation
- Design of H-MADRL
- Implementation
- **Performance Evaluation**
- Conclusions and Future Work

Experimental Setup

- Evaluation environment: 16 heterogeneous nodes managed by Kubernetes, with Apache Flink for stream processing

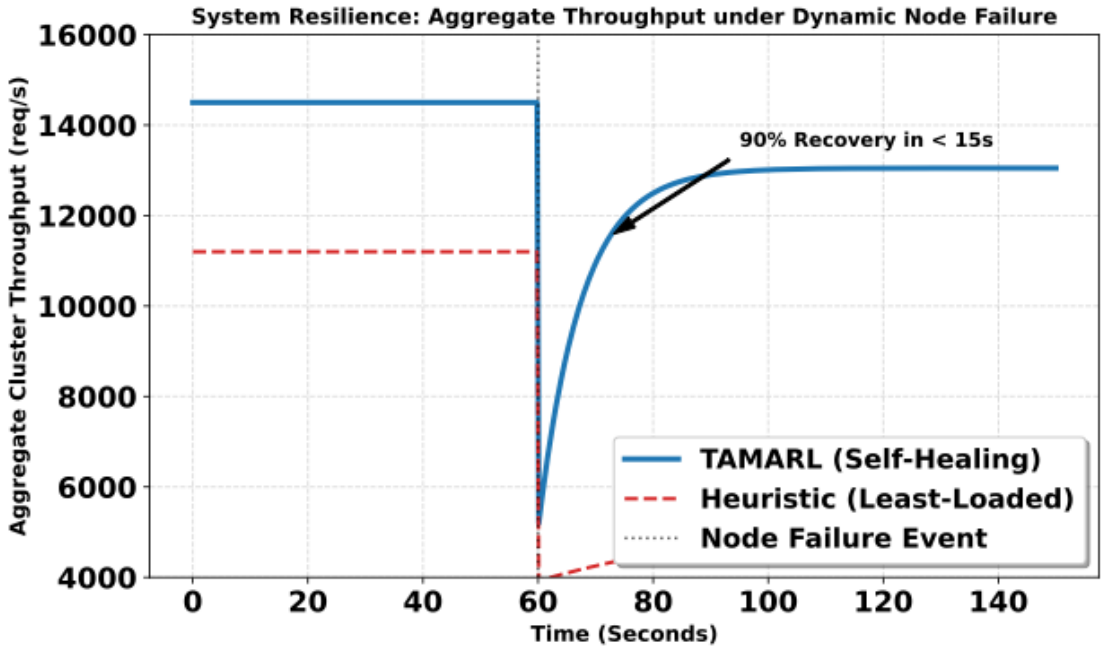
- **Workloads**
  - ➢ Bursty and stress-test DAGs
  - ➢ Dynamic resource pressure and node failures

- **Baselines**
  - ➢ Round Robin, Least Loaded
  - ➢ Single-Agent PPO, Static-DAG

- **Evaluation metrics**
  - ➢ Average latency, P99 latency, SLO violation rate, CPU utilization, throughput, resource waste, and network locality

Six stress-test scenarios evaluate resilience, load scalability, heterogeneity, DAG complexity, and communication locality.

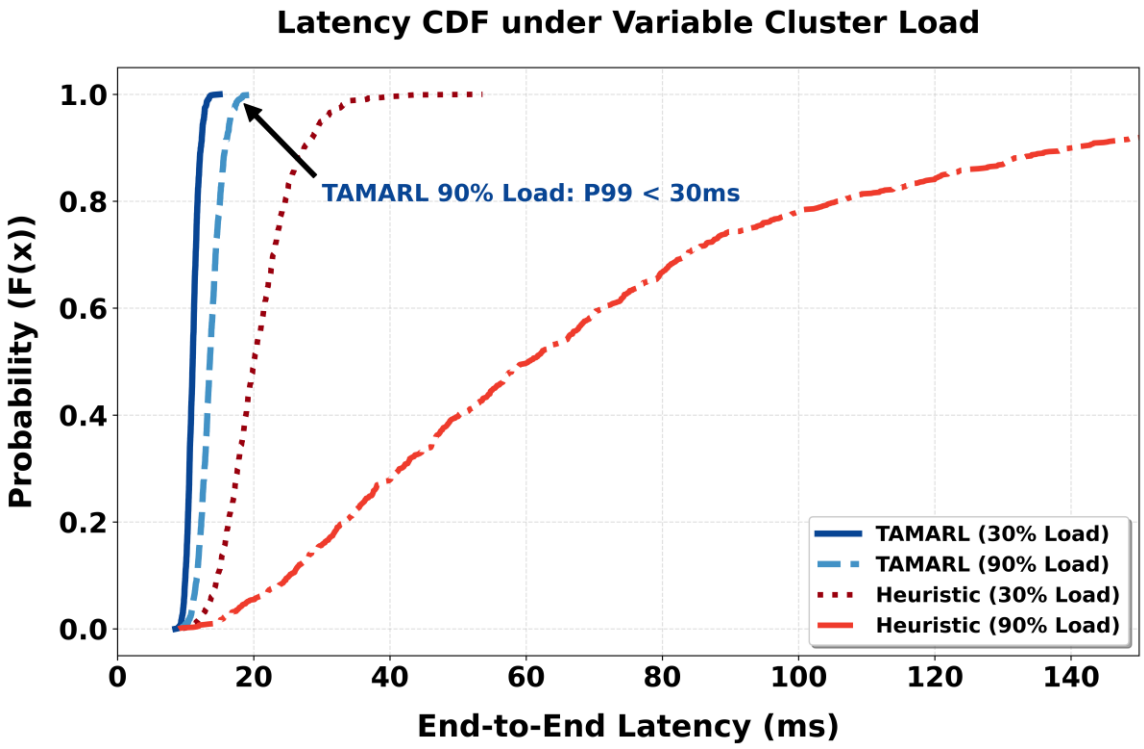
Evaluation: Resilience Under Node Failure



- A high-compute node is manually terminated during peak workload.
- TAMARL restores about 90% throughput within 15 seconds.
- Least-Loaded remains degraded because it lacks topology-aware recovery.

Result: Decentralized execution enables rapid self-healing without waiting for a slow global scheduling decision.

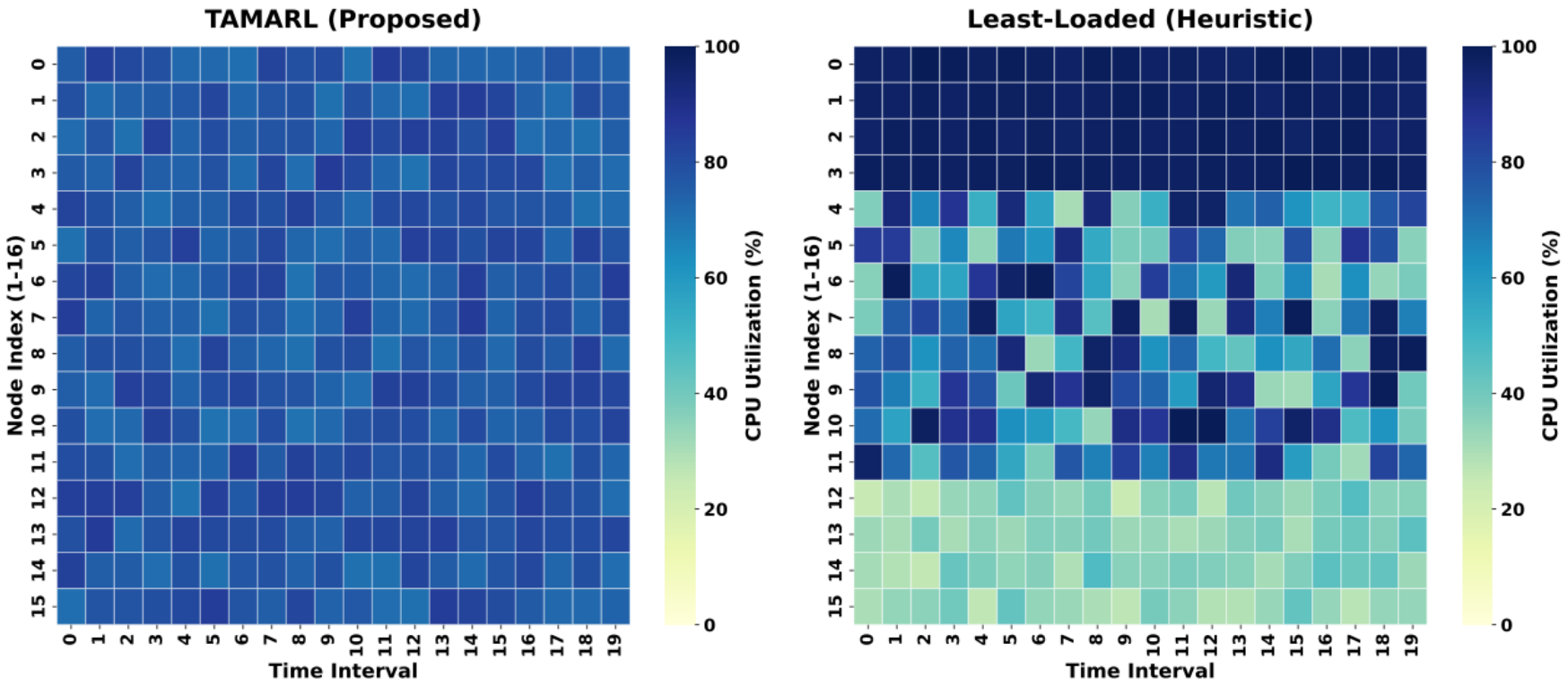
Evaluation: Latency and Load Robustness



- Under rising cluster utilization, topology-blind heuristics suffer severe tail-latency spikes.
- TAMARL keeps P99 latency stable by proactively redistributing DAG subgraphs.
- Stable latency is critical for real-time cloud-native stream processing.

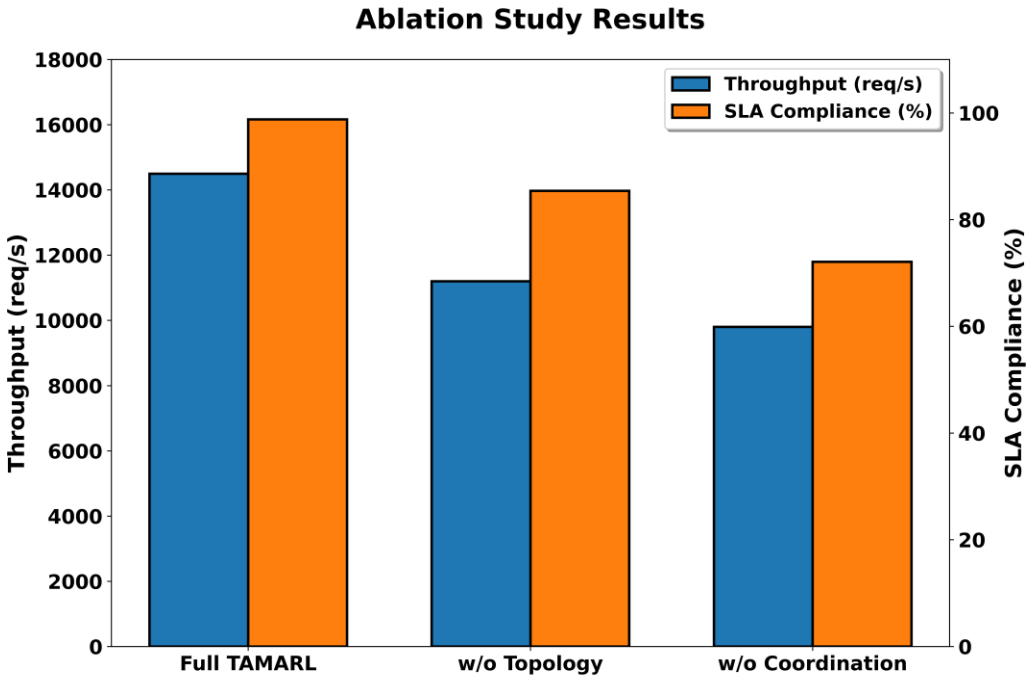
Result: Topology-aware MARL protects P99 latency even as load increases.

Evaluation: Cluster Load Balancing



Result: TAMARL removes hotspots and keeps nodes in a balanced utilization range, while Least-Loaded fragments resources.

Evaluation: Ablation Study

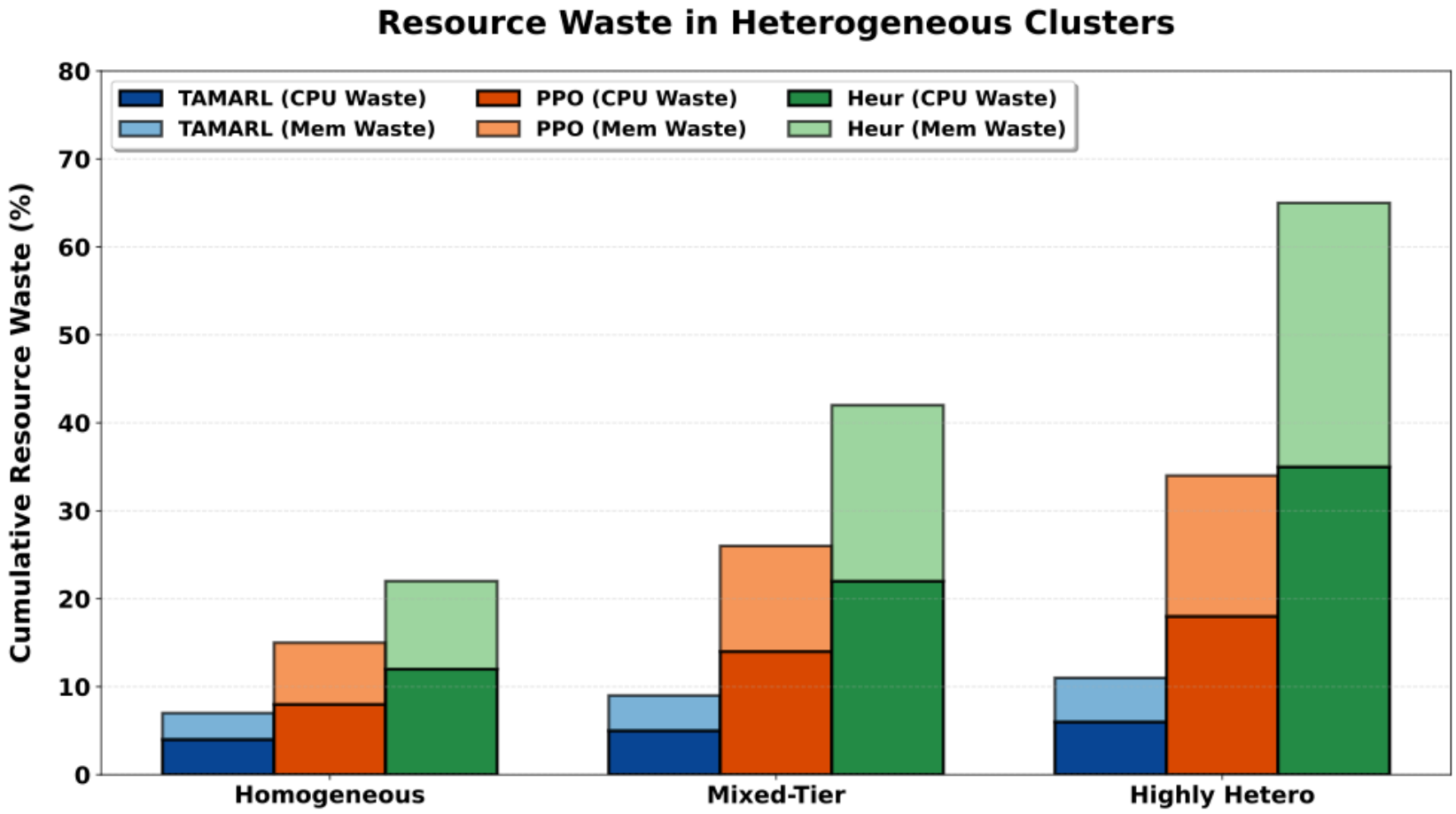


Configuration	Throughput (req/s)	Stability Score
Full TAMARL	14,500	0.94
w/o Topology Awareness	11,200	0.78
w/o Multi-Agent Co-op	9,800	0.65

- Removing topology awareness lowers throughput because high-traffic DAG edges are placed inefficiently.
- Removing multi-agent coordination reduces stability because agents act greedily instead of cooperatively.

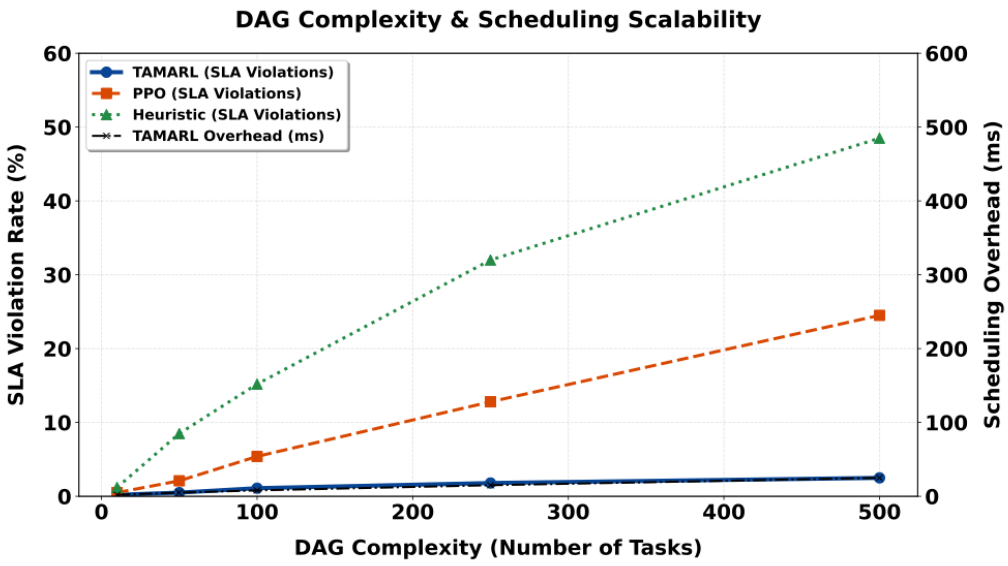
Conclusion: TAMARL needs both topology awareness and MARL coordination to deliver the full performance gain.

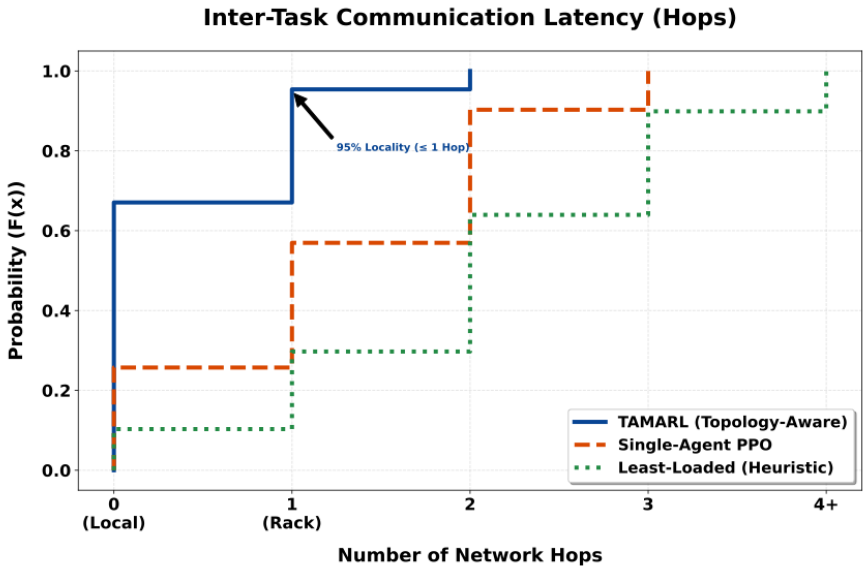
Evaluation: Heterogeneous Cluster Efficiency



Result: TAMARL reduces cumulative resource waste by matching compute-bound and memory-bound tasks to appropriate node types.

Evaluation: Scalability and Network Locality





- DAG complexity: TAMARL keeps scheduling overhead almost linear as task count grows.
- Network locality: 95% of dependent task pairs are placed within one hop.

Performance Summary

Scheme	Avg Lat. (ms)	P99 Lat. (ms)	CPU (%)	SLA (%)
Round Robin	452.4	890.1	62.1	12.4
Least Loaded	398.2	712.5	68.4	8.2
Single-PPO	312.5	540.2	74.2	4.5
TAMARL	**215.8**	**320.5**	**88.6**	**1.2**

Result: TAMARL achieves the lowest average latency, lowest P99 latency, highest CPU utilization, and lowest violation rate among all compared methods.

Conclusions and Future Work

- **Our contributions**
  - TAMARL integrates topology-aware state representation with multi-agent reinforcement learning for cloud-native stream processing.
  - It improves latency, SLO attainment, network locality, load balancing, and resilience under node failures.
  - Ablation results confirm that both topology awareness and cooperative multi-agent control are essential.
- **Future work**
  - ➢ Online graph inference for black-box/serverless systems
  - ➢ Graph Attention Networks inside sidecar agents for online adaptation
  - ➢ Carbon-aware orchestration across distributed edge-cloud environments

Thank you!
Questions & Comments?

Sunday J. Awine and Jinwei Liu

{sunday1.awine, jinwei.liu}@famu.edu

Department of Computer and Information Sciences

Florida A&M University