A Cloud-Edge Collaborative Framework for Distributed Triangle Counting on Graph Stream

Ruilin Hu[1], Chao Song[1], Jie Wu[2] and Li Lu[1]

1. University of Electronic Science and Technology of China, China

2. Temple University, USA

Background

Large-Scale Graph : Foundations and Challenges

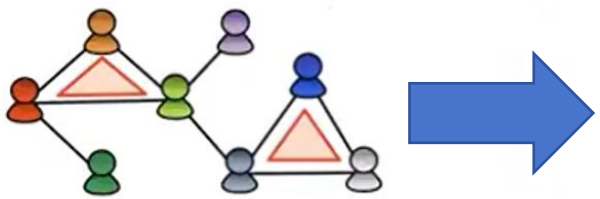
- Real-world networks—such as social, communication, biological, and financial networks—are often large and structurally complex
- Graph computation is essential for discovering patterns, relationships, and behaviors in such systems
- Efficient large-scale graph analytics is critical for understanding and leveraging complex systems

Subgraph Counting: A Core Analytical Task

- Subgraph counting is a fundamental task in graph pattern mining
- It reveals local structural properties by quantifying specific patterns such as stars, paths, and cycles

Common subgraph types include:

- Triangle
- 2-Star
- 3-Star



Shape	Name	Count
	Triangle	2
	2-star	15
	3-star	6

TEMPLE UNIVERSITY

University of Electronic Science and Technology of China

Background： Triangle Counting

Among all subgraphs, triangles are particularly important. Let’s explore why triangle counting is such a central task.

Application Scenario	Graph Computation Tasks	Triangle Counting Involved
Social Networks (e.g., Facebook, Weibo)	Community detection, friend recommendation, influence propagation	☑ Triangles indicate tight-knit user communities
Financial Transaction Systems (e.g., Banking, Blockchain)	Fraud detection, money flow tracking, suspicious cycle identification	☑ Triangles may reveal cyclic transactions and collusion
Computer Networks and Communication Systems	Traffic anomaly detection, routing optimization, botnet detection	☑ losed loops may indicate coordination among nodes
Urban Transportation and Logistics Networks	Route planning, congestion detection, redundancy analysis	☑ Triangles can model closed travel paths among locations
Healthcare and Medical Information Networks	Disease transmission tracking, hospital-patient-disease relationships	☑ Triangles may suggest stable comorbidity relationships

Background： Streaming graph

In many real-world applications, graphs evolve over time, which brings us to the concept of streaming graphs

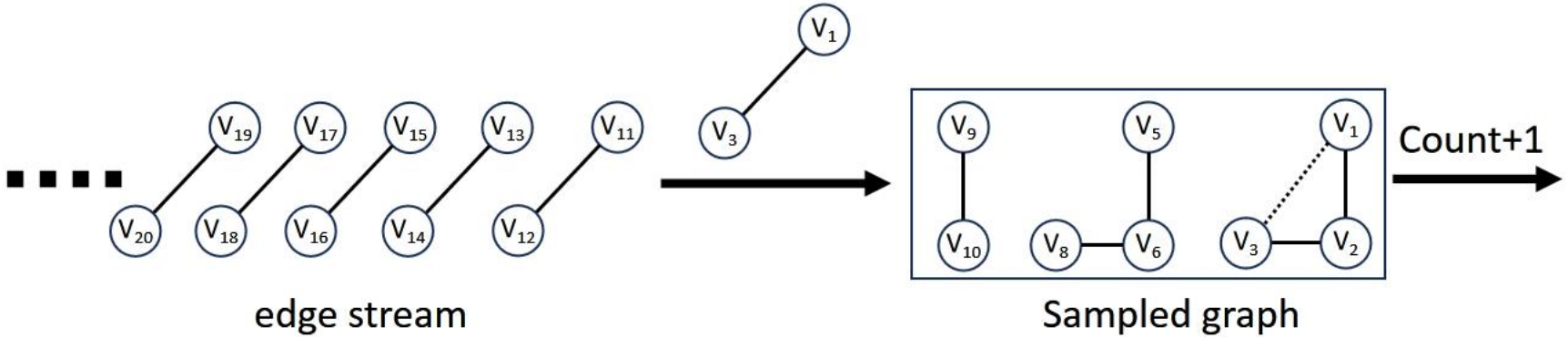
- Streaming graph analysis is becoming increasingly important in various domains due to the natural dynamics in many real-world graph applications.

$t=1$	$t=2$	$t=4$	$t=5$	$t=6$	$t=6$	$t=7$	$t=8$	$t=9$	$t=10$
$(1,2)$	$(1,3)$	$(2,3)$	$(3,4)$	$(1,5)$	$(2,5)$	$(1,2)$	$(1,2)$	$(4,6)$	$(2,4)$

Sampling

Since streaming graphs are too large to be processed in full, sampling becomes an essential preprocessing technique.

- Large graph sampling refers to the technology of selecting representative points and edges from the original large-scale network.
- It can effectively reduce the scale of large graph data and significantly improve the computing efficiency and visualization effect of large graphs.
- Among them, reservoir sampling is a basic mode.



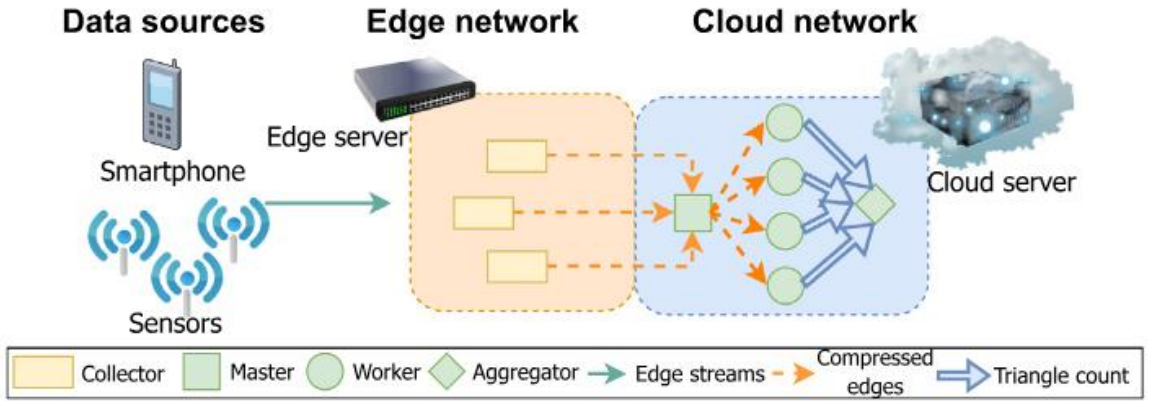
Cloud-Edge Collaborative

Challenges in Streaming Graph Analysis

- High-frequency edge generation from distributed sources (e.g., sensors, smartphones)
- Centralized processing causes latency, bandwidth bottlenecks, and unbalanced load

Why Cloud-Edge Collaboration?

- Edge devices: Perform lightweight pre-processing and data assignment
- Cloud servers: Handle global coordination, aggregation, and complex analytics

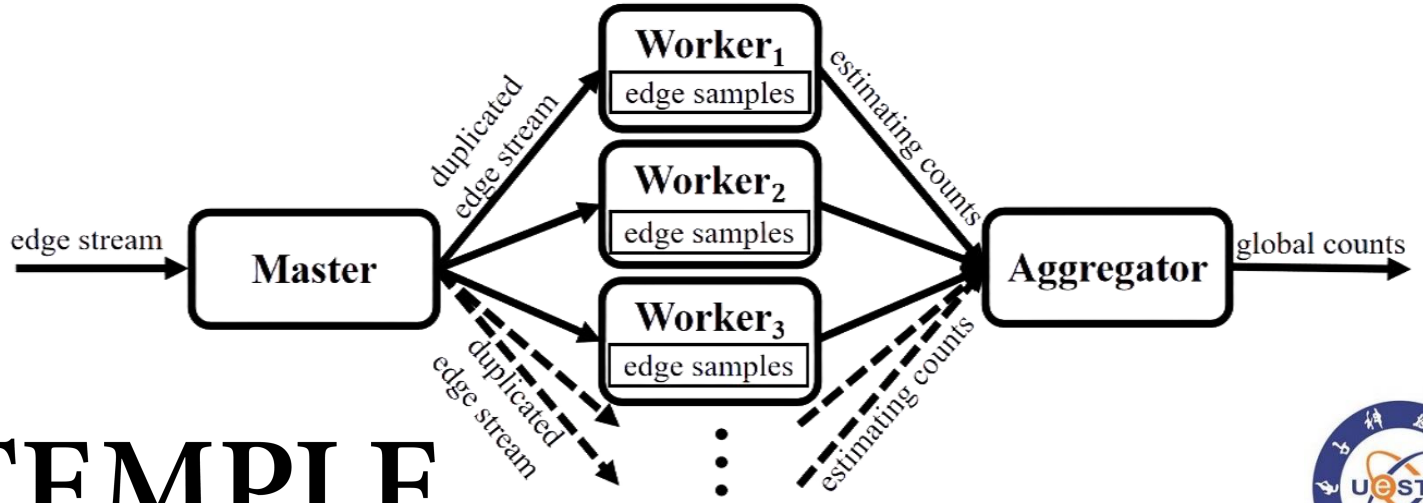


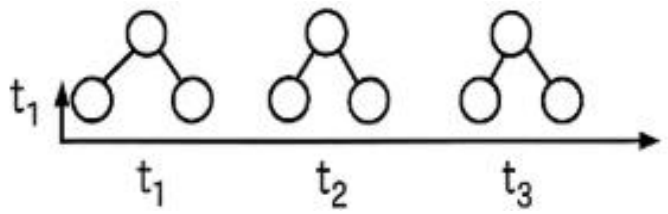
[KBS 2023]

Challenge

Although this architecture is promising, it comes with several non-trivial challenges that must be addressed. Traditional frameworks use the Master-Worker-Aggregator architecture with reservoir sampling to limit memory usage. However, they face several critical issues:

- (1)Previous research adopted a Master-Worker-Aggregator architecture, in which the master collected a large number of edges and sent them to workers?
- (2) How to distribute edges to maximize triangles formed within each worker?
- (3) Given the dynamic edge streams, how can their correlations be detected and the distribution strategy be updated in real-time?





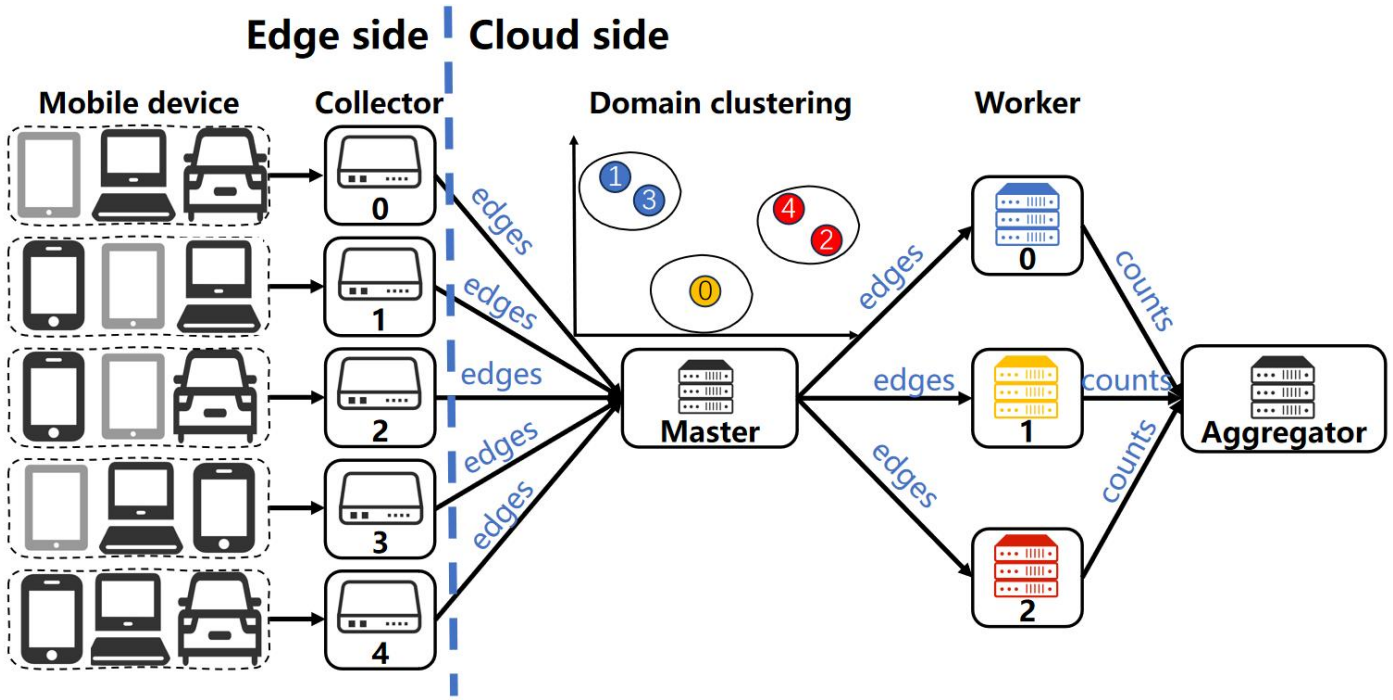
TEMPLE UNIVERSITY

University of Electronic Science and Technology of China

Overview

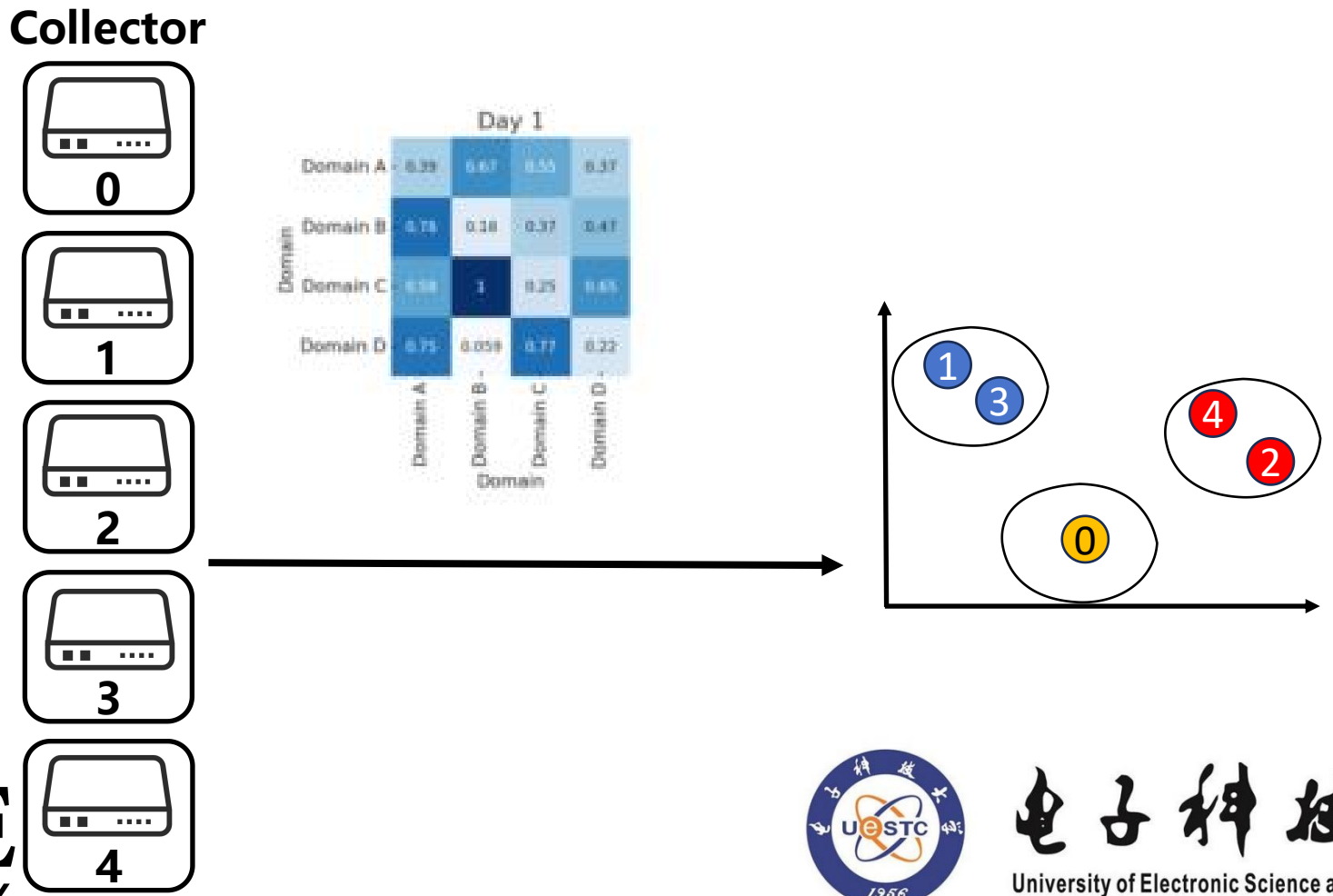
To address these challenges, we propose a cloud-edge collaborative framework with domain clustering.

- Collectors: collect edges on edge devices.
- Master : then performs spectral clustering to determine the latent domain structure.
- Workers: estimate local triangle counts.
- Aggregator: compute the global estimate.



Solution

- (1) **Clustering:** We transform the edge distribution task into assigning domain IDs, which reduces the hash space and reduces the master's distribution workload.

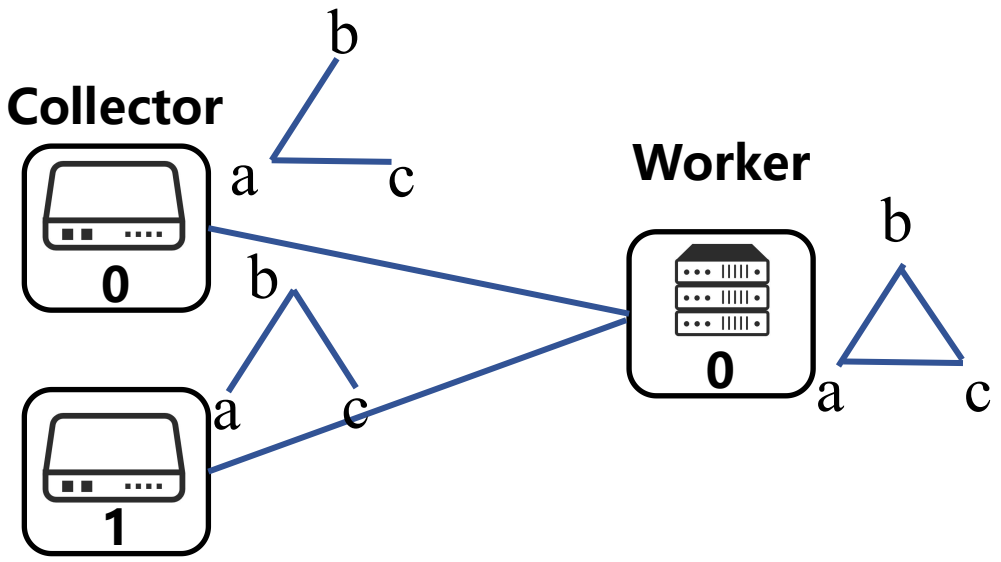


Domain clustering

- First, we construct a weighted adjacency matrix where each entry $w_{i,j}$ represents the flow of edges between domain i and j.
- Then, we compute the graph Laplacian:$L = D - W$, where D is the degree matrix.
- Next, we minimize the RatioCut loss function using eigenvalue decomposition of the Laplacian.
- The smallest **c** eigenvectors represent the optimal cluster assignment.
- This ensures that most triangle-forming edges remain in the same worker, improving accuracy and reducing inter-node communication.

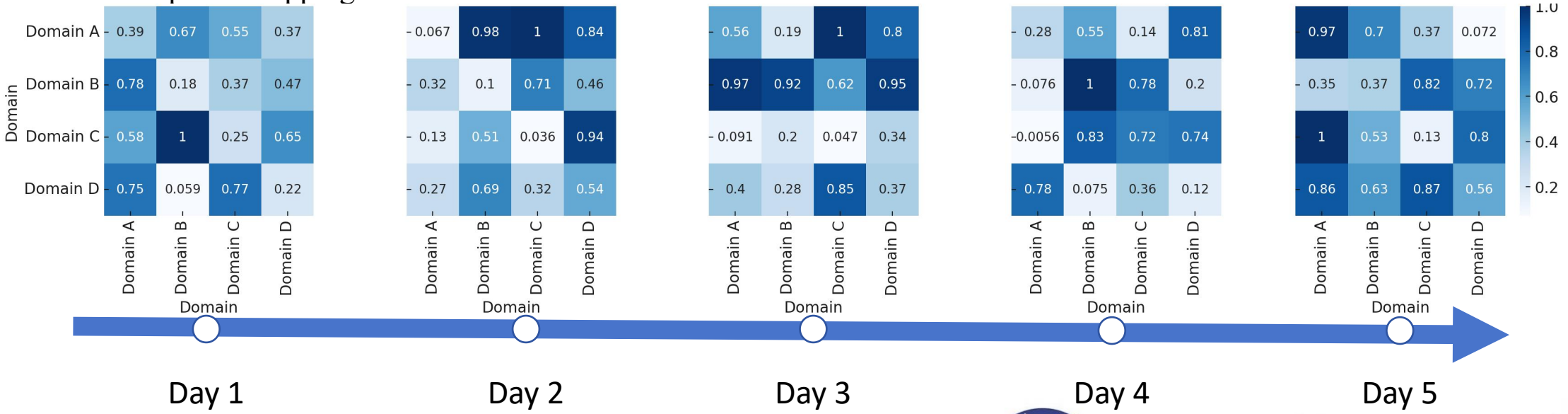
Solution:

- (2) **Assignment**: We perform domain clustering to ensure that more edges forming triangles are distributed to the same worker. This increases the likelihood of detecting triangles locally.



Solution:

- (3) **Dynamic**： At each graph snapshot, we build an adjacency matrix to adjust our distribution strategy to the dynamic of the streaming edge.
  - Receive a Graph Snapshot
  - Construct the Adjacency Matrix
  - Adjust Distribution Strategy
  - Update Mapping



TEMPLE UNIVERSITY

University of Electronic Science and Technology of China

Experiment: Datasets and Baseline

In our evaluation, we employ a selection of real-world graph datasets to assess the effectiveness of ouralgorithms. The datasets are publicly available and sourcedfrom the renowned SNAP4

Name	Nodes	Edges	Triangles
Email-Enron	36,692	183,831	727,044
Gowalla	196,591	950,327	2,273,138
DBLP	317,080	1,049,866	2,224,385
YouTube	1,134,890	2,987,624	3,056,386

Baseline

- Tri-Fly : Tri-Fly is a simple distributed TRI*È*ST − IMPR replication using broadcast edge stream. Each worker runs the TRI*È*ST − IMPR algorithm, and the aggregator averages the counts.
- CoCoS : the master allocation strategy optimizes Lucky type to reduce communication overhead; the worker sampling part only samples edges that have a vertex hash into this worker, reducing the sampling range and being able to redundant between workers.

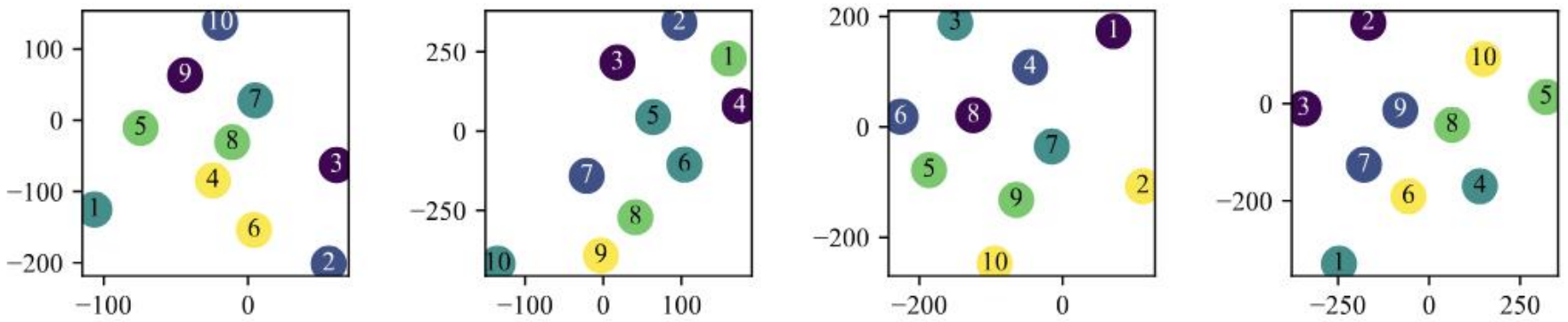
Experiment: Evaluation Metrics

All experiments were conducted on a machine with 24 CPUs x Intel(R) Xeon(R) Silver 4214 CPU @2.20GHz and 128GB RAM.

- Global Error (GRE):The GRE metric quantifies the accuracy of the estimated global triangle counts compared to the ground truth values.It provides a measure of how closely the estimated value, denoted GMAPE = $|\hat{\tau}(t) - \tau(t)|/\tau$.
- Elapsed Time: When multiple CPUs process tasks simultaneously, the CPU time will be greater than the elapsed time.In this paper, elapsed time refers to the total running time of the entire distributed architecture.

Performance of Clustering

- we generate timestamps using a Poisson distribution for the probability of different edges arriving at the collector.
- Axes are the two dimensional representations of the original high-dimensional data, created to preserve the structure and relationships between data points in a simplified form. Clustering performed well across multiple domains on four different datasets.

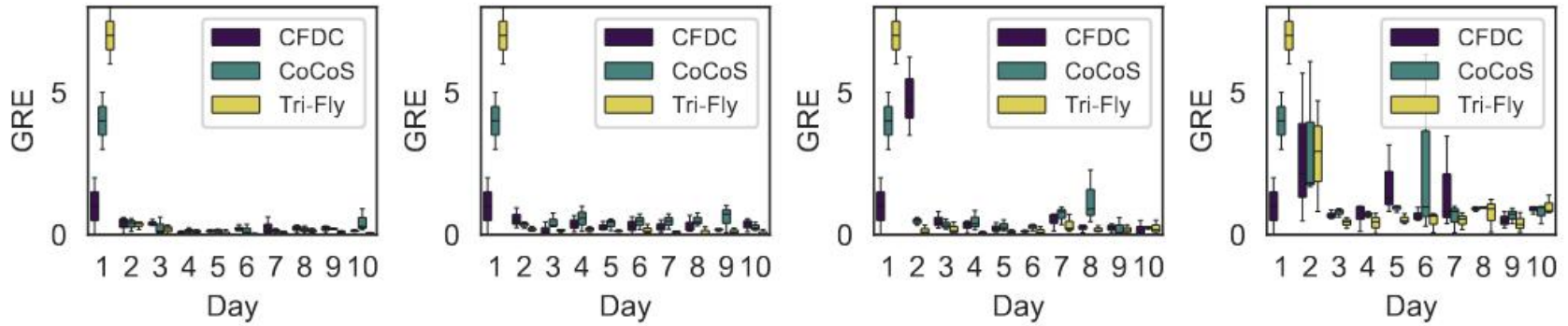


(a) Email-Enron (b) DBLP (c) Gowalla (d) YouTube

- Clustering performed well across multiple domains on four different datasets.

Performance of Accuracy

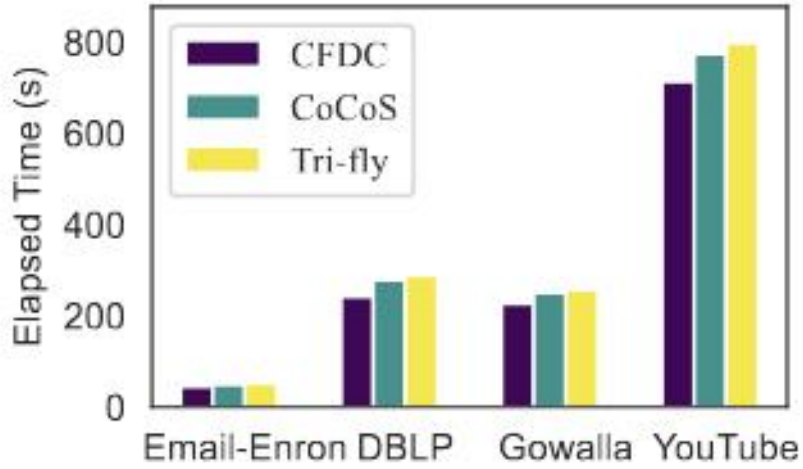
We use 10 collectors and 5 workers with the memory budget k fixed to 2% of the number of edges in the entire dataset.



- The CFDC algorithm provides a lower global relative error across various datasets, demonstrating its superiority in the task of distributed triangle counting.

Performance of Elapsed Time

We use 10 collectors and 5 workers with the memory budget k fixed to 2% of the number of edges in the entire dataset.



- CFDC method outperforms both CoCoS and Tri-Fly in terms of elapsed time. This indicates that CFDC computes triangle counts in graph streams more efficiently.

Conclusion

- We propose a cloud-edge collaborative framework in distributed triangle counting in graph streams.
- We employ spectral clustering analysis to reveal latent domain relationships that guide edges distribution.
- The approach's superiority over existing algorithms in terms of counting accuracy and elapsed time has been substantiated through experiments on various datasets

Thank You!

Q&A: huruilin@std.uestc.edu.cn