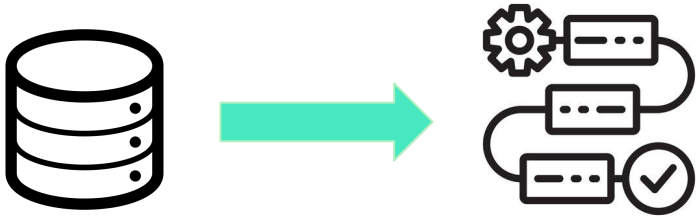
Fries: Fast and Consistent Runtime Reconfiguration in Dataflow Systems with Transactional Guarantees

Zuozhi Wang, **Shengquan Ni**, Avinash Kumar and Chen Li

VLDB 2023, Vancouver, Canada

UCIrvine

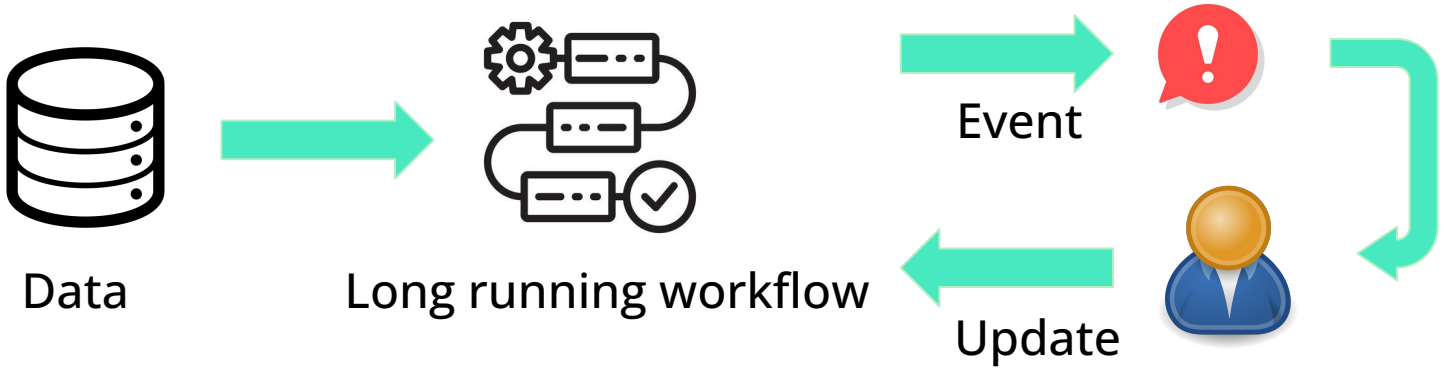
Big Data Workflows



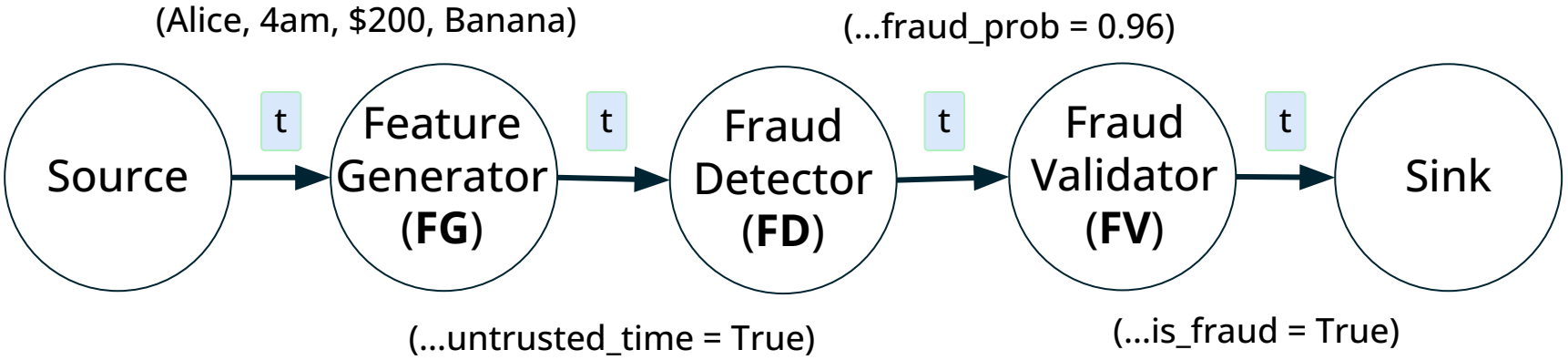
Data

Long running workflow

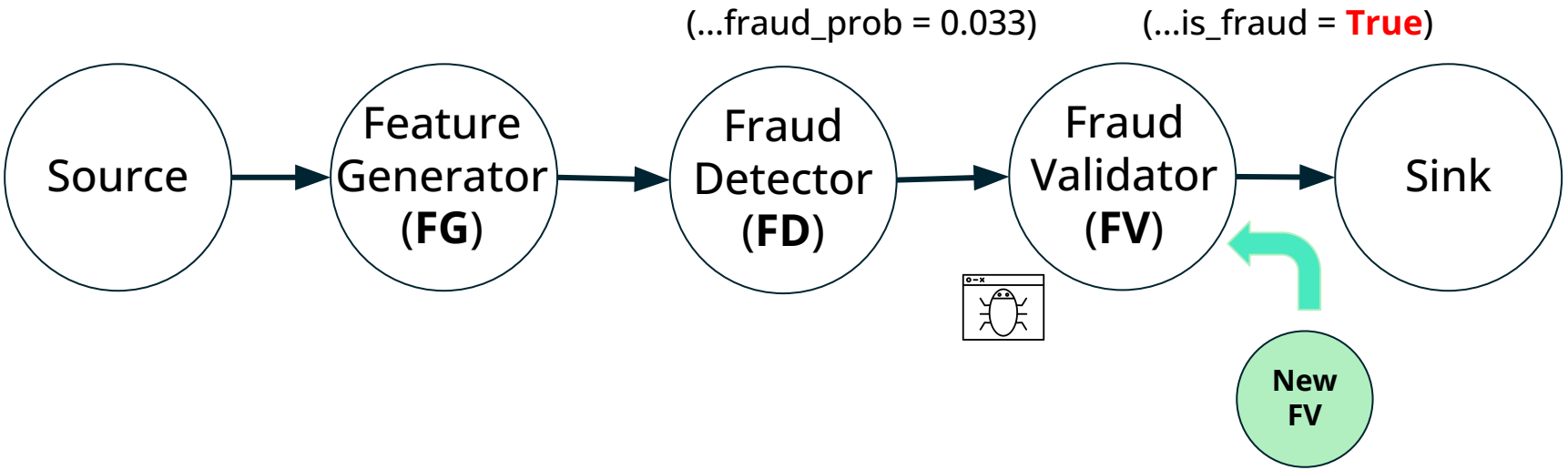
Motivation: Updating Logic in Big Data Workflows



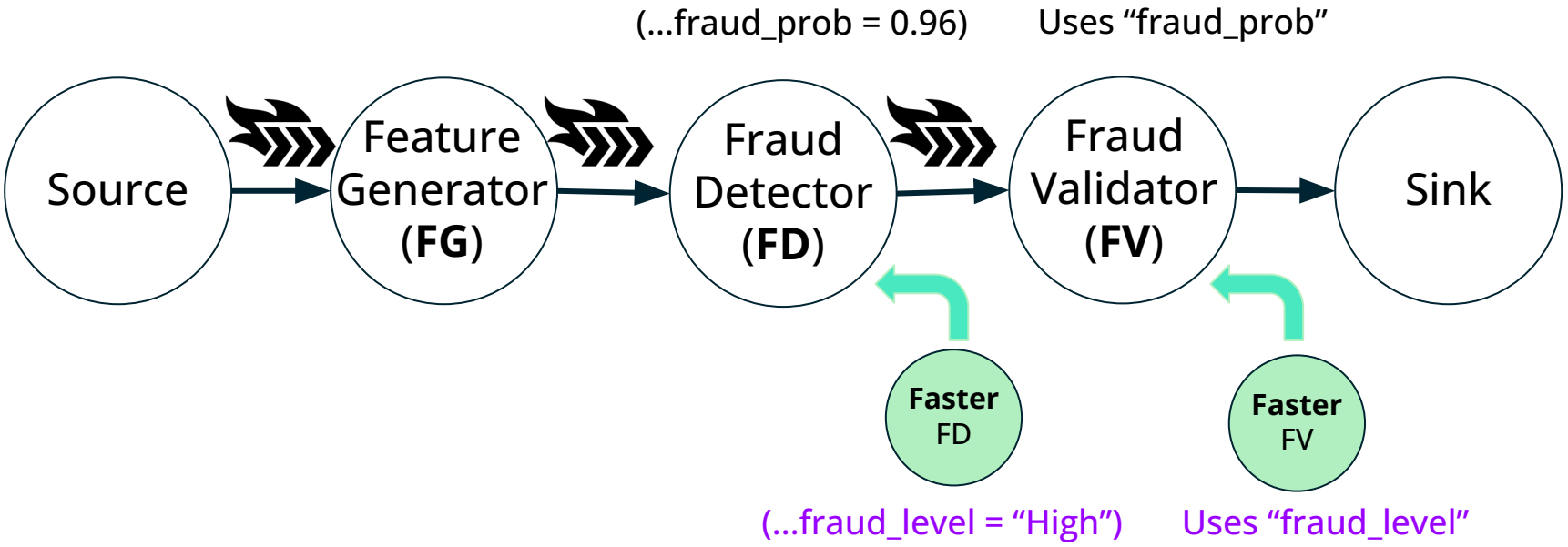
Example Workflow



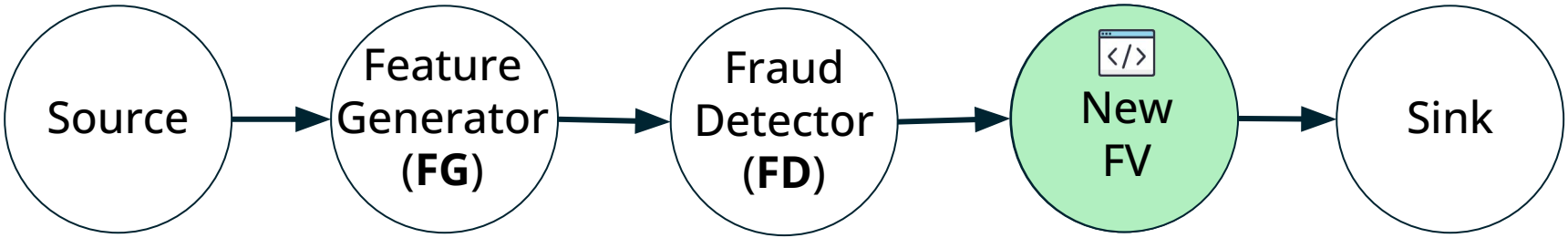
Update Logic Case 1: Fixing runtime bugs



Update Logic Case 2: Mitigating surge of input data



Reconfiguration: Updating operator code



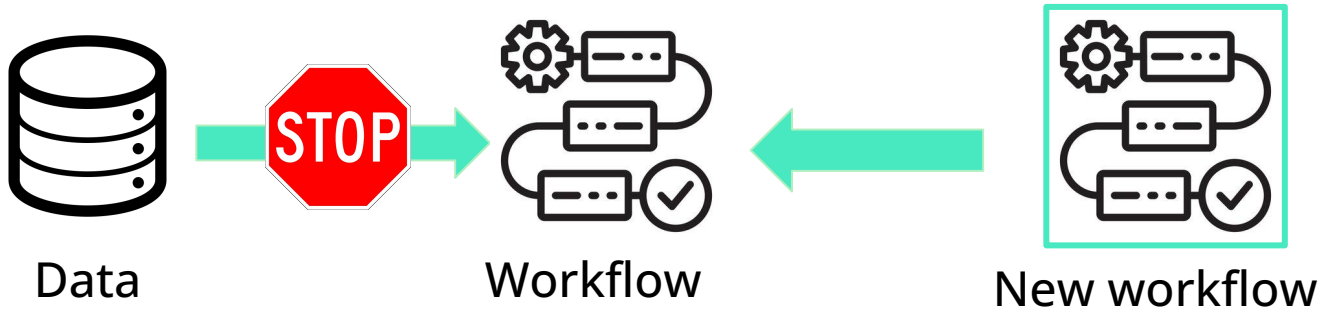
How to do reconfigurations?

Baseline 1: Stop and Restart

Pause Data Ingestion.

Wait for workflow to finish current tuples.

Replace the workflow.



Baseline 1: Stop and Restart

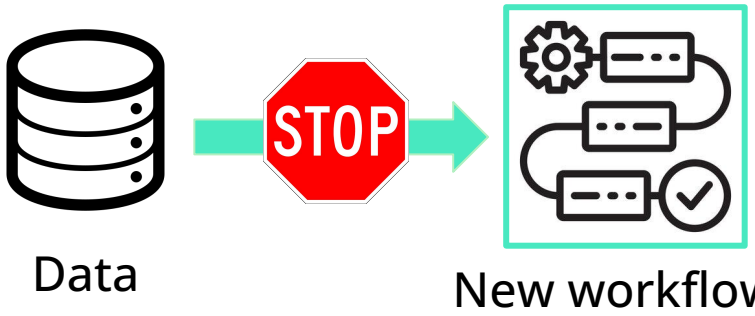
Pause Data Ingestion.

Wait for workflow to finish current tuples.

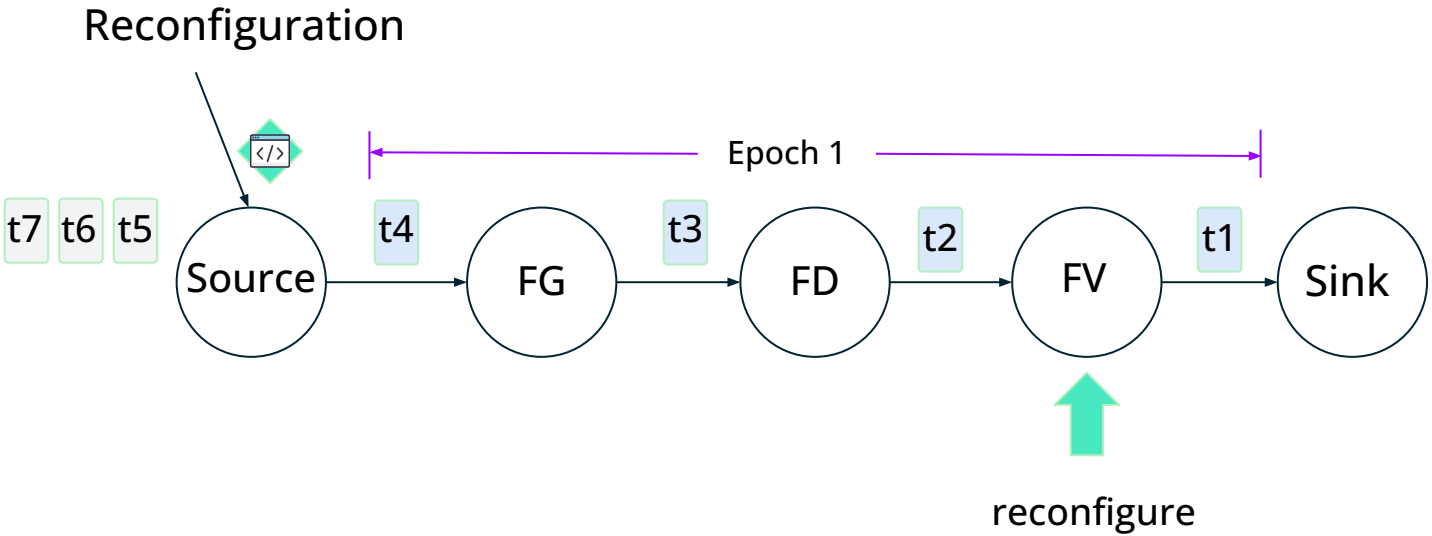
Replace the workflow.

Resume data ingestion.

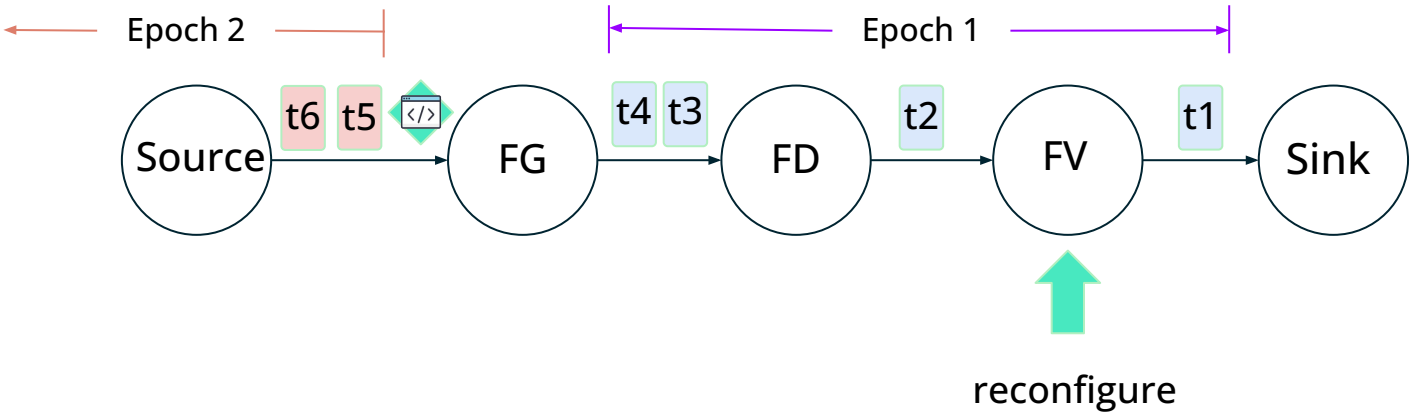
Disruptive!



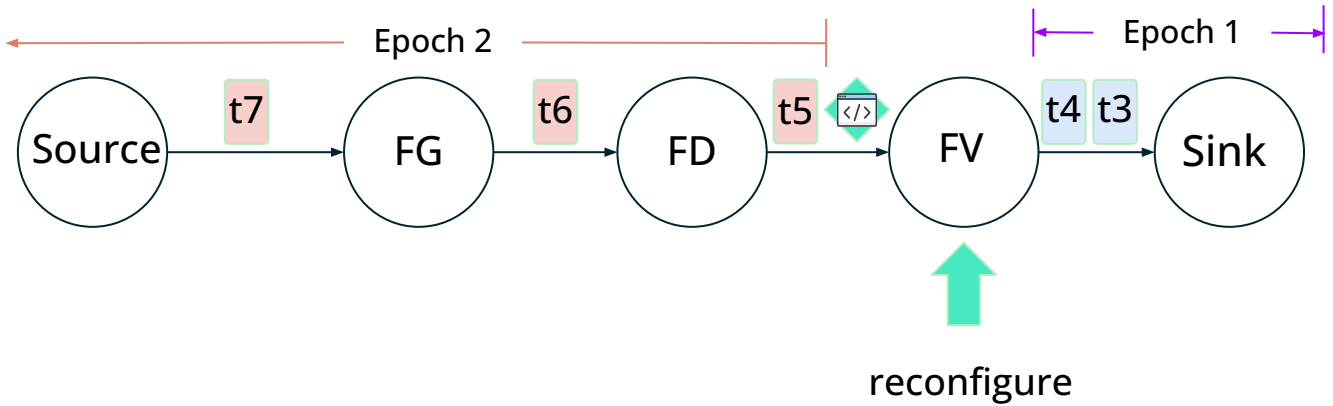
Baseline 2: Reconfiguring between epochs



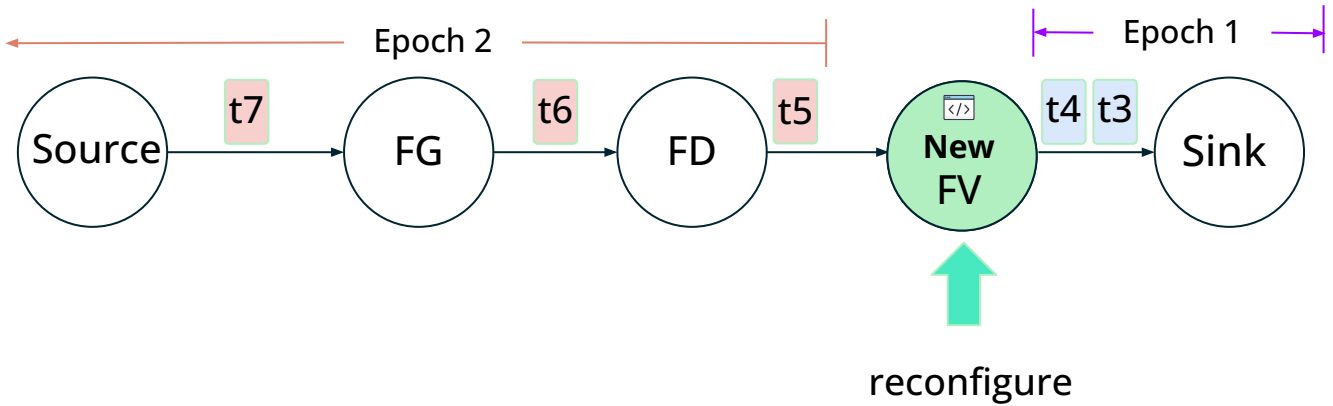
Baseline 2: Reconfiguring between epochs



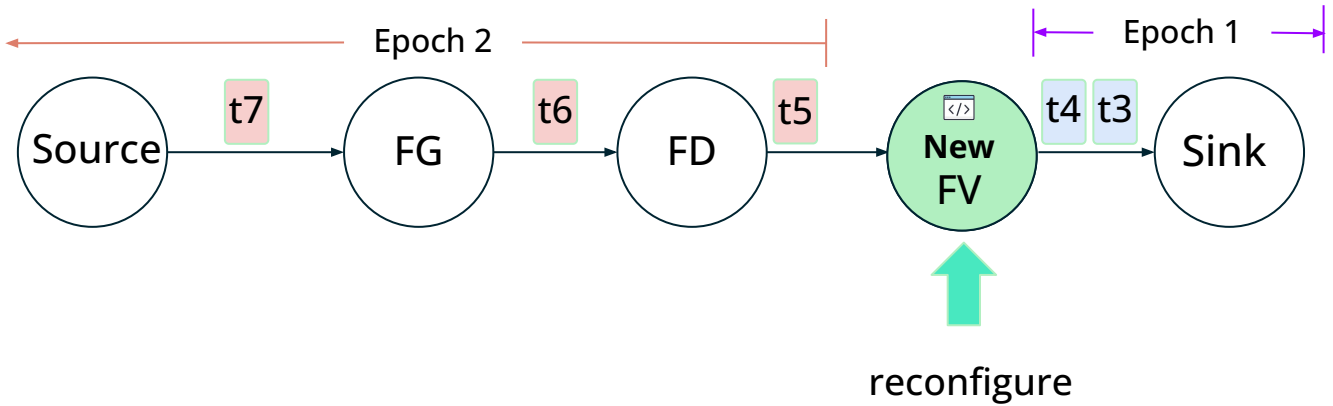
Baseline 2: Reconfiguring between epochs



Baseline 2: Reconfiguring between epochs



Baseline 2: Reconfiguring between epochs



Wait for processing of **all data in Epoch 1** ***slow!***

Challenges

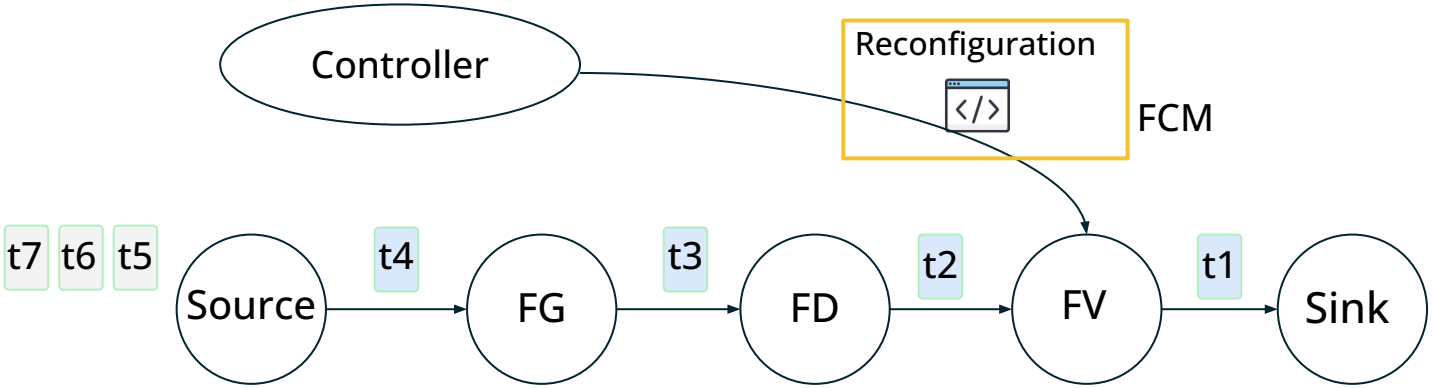
1. How to do it **fast**?
2. How to guarantee **consistency**?

Fries: a technique to answer those questions!

Challenges

1. How to do it **fast**?
2. How to guarantee **consistency**?

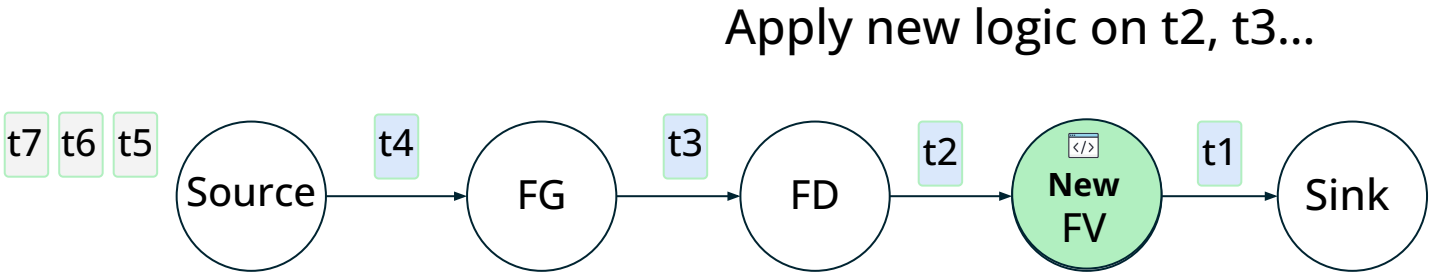
Using Fast Control Message (FCM)



Data does not block FCMs.

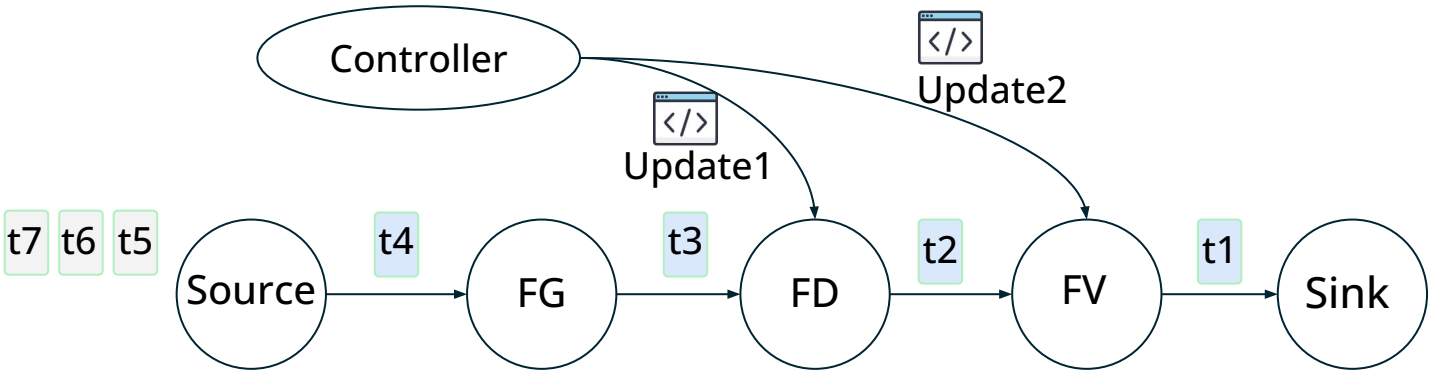
Fast!

Using Fast Control Message (FCM)

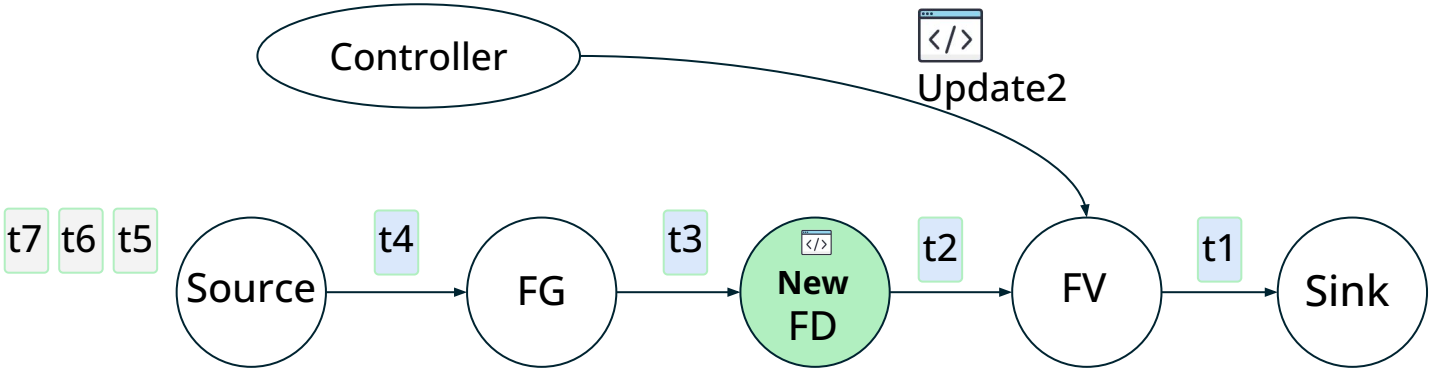


For multiple operators?

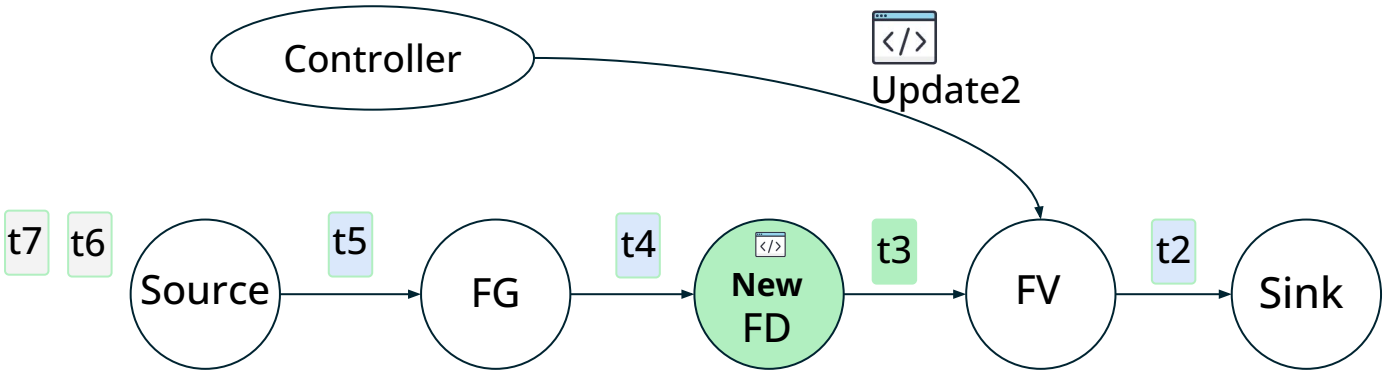
Using Fast Control Message (FCM)



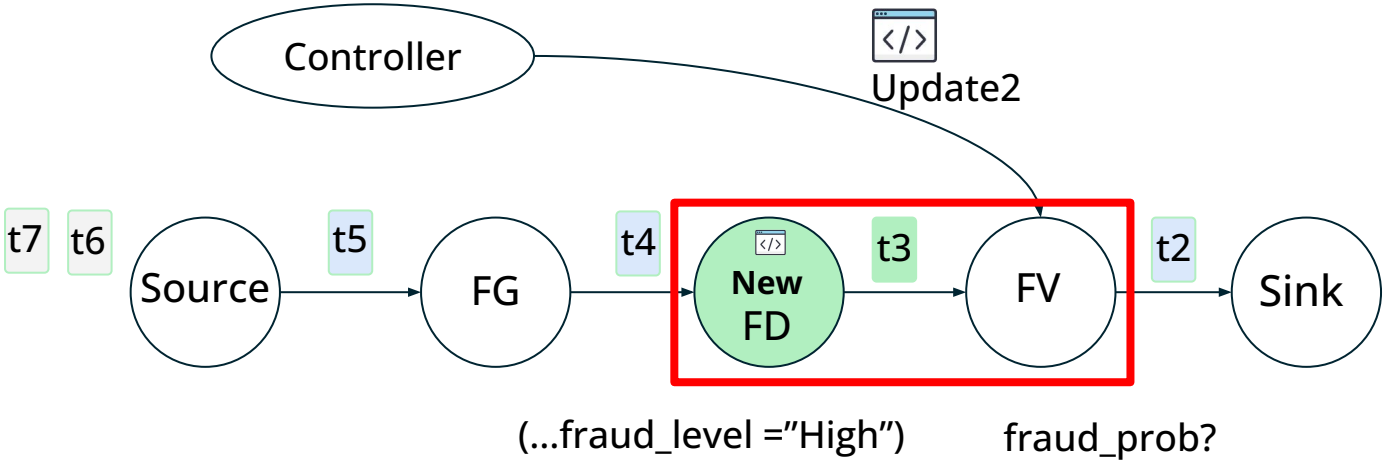
Using Fast Control Message (FCM)



Using Fast Control Message (FCM)



Using Fast Control Message (FCM)



Old FV cannot accept data from new FD.
E.g., schema mismatch.

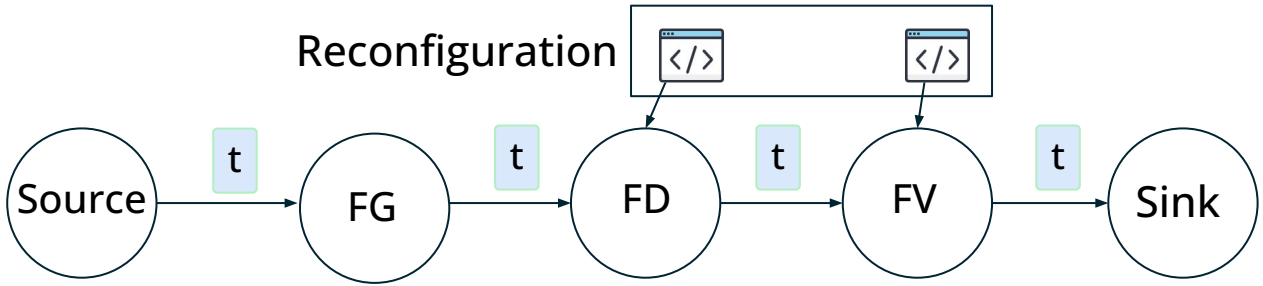
Inconsistent!

Challenges

1. How to do it **fast**?
2. How to guarantee **consistency**?

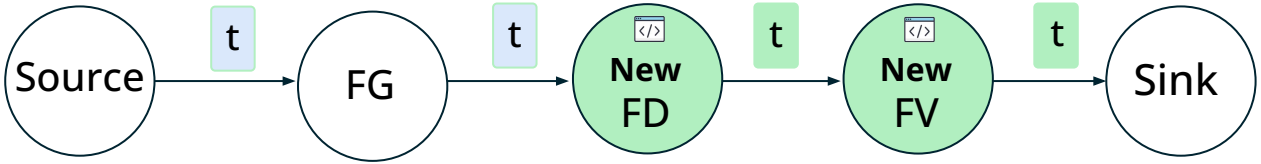
Reconfiguration Consistency

A tuple should be processed by the **same** version.

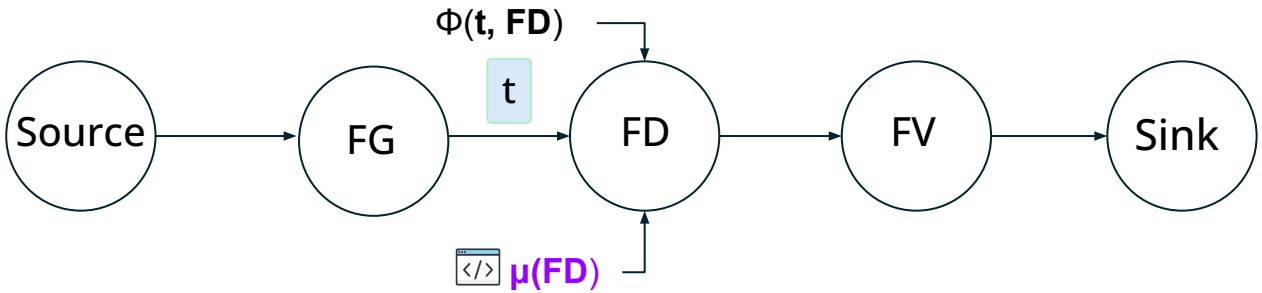


Reconfiguration Consistency

A tuple should be processed by the **same** version.



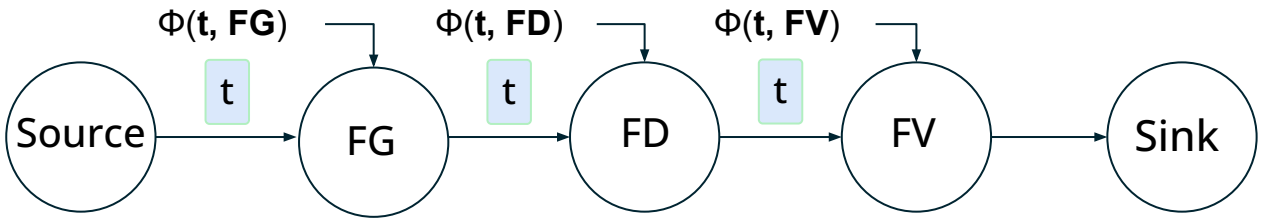
Reconfiguration Consistency



Data operation: Φ(**tuple, operator**)

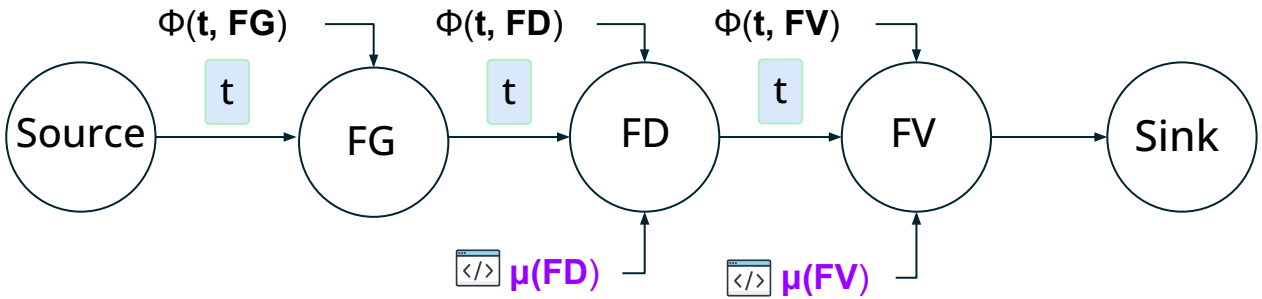
Function-update operation: μ(**operator**)

Reconfiguration Consistency



Data Transaction **T1** = [Φ(t, FG), Φ(t, FD), Φ(t, FV)]

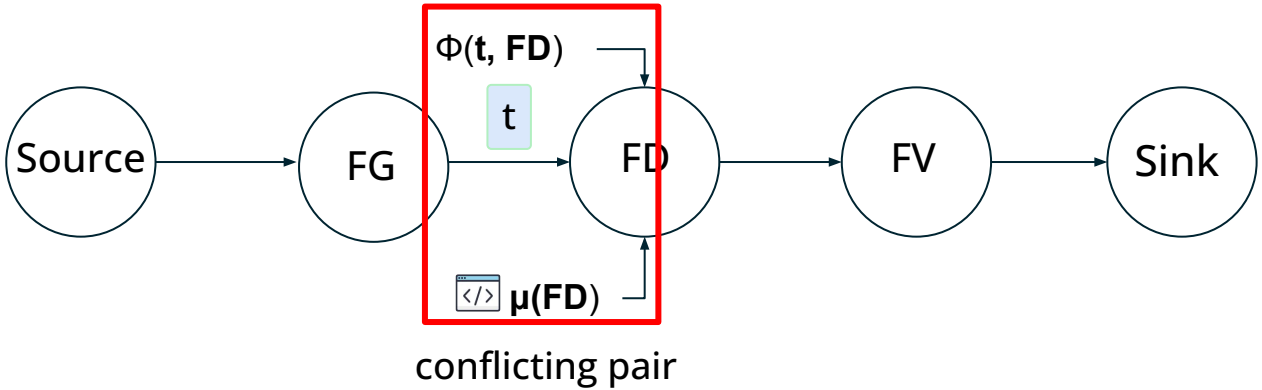
Reconfiguration Consistency



Data Transaction **T1** = [Φ(t, FG), Φ(t, FD), Φ(t, FV)]

Function-update Transaction **T2** = {μ(FD), μ(FV)}

Reconfiguration Consistency

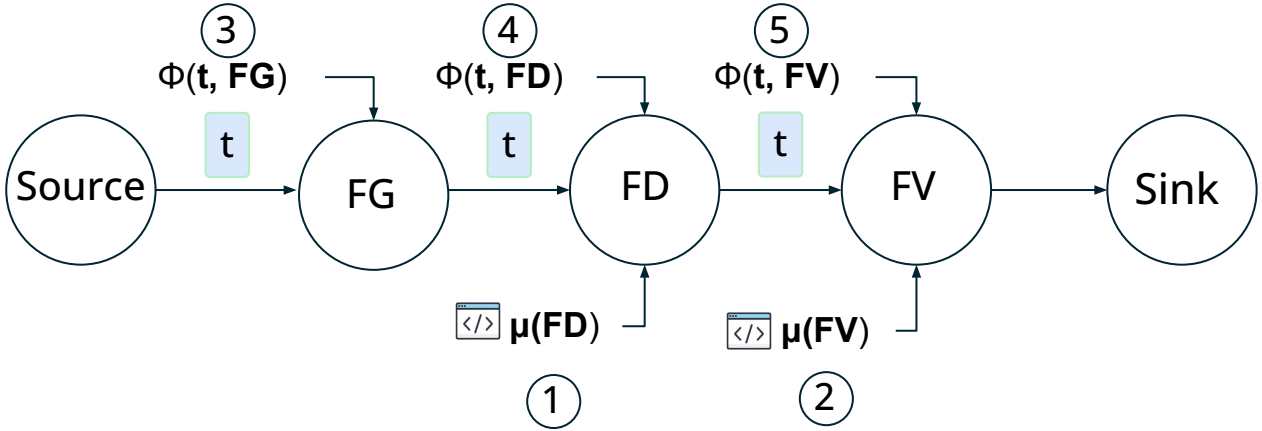


Data Transaction **T1** = [Φ(t, FG), Φ(t, FD), Φ(t, FV)]

Function-update Transaction **T2** = {μ(FD), μ(FV)}

How to resolve conflicts?

Reconfiguration Consistency

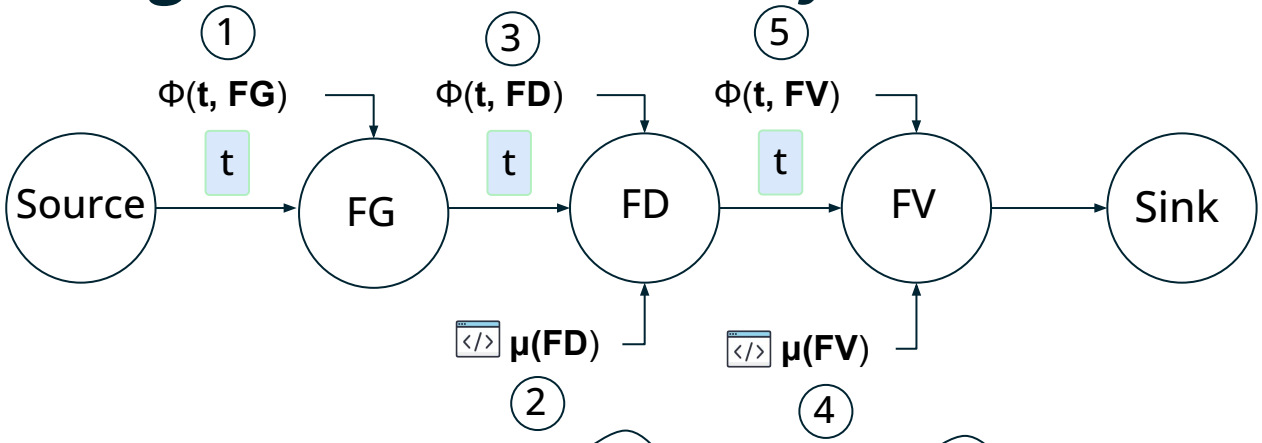


Serial Schedule S1 = [μ(FD) , μ(FV), Φ(t, FG), Φ(t, FD), Φ(t, FV)]

Using new logic

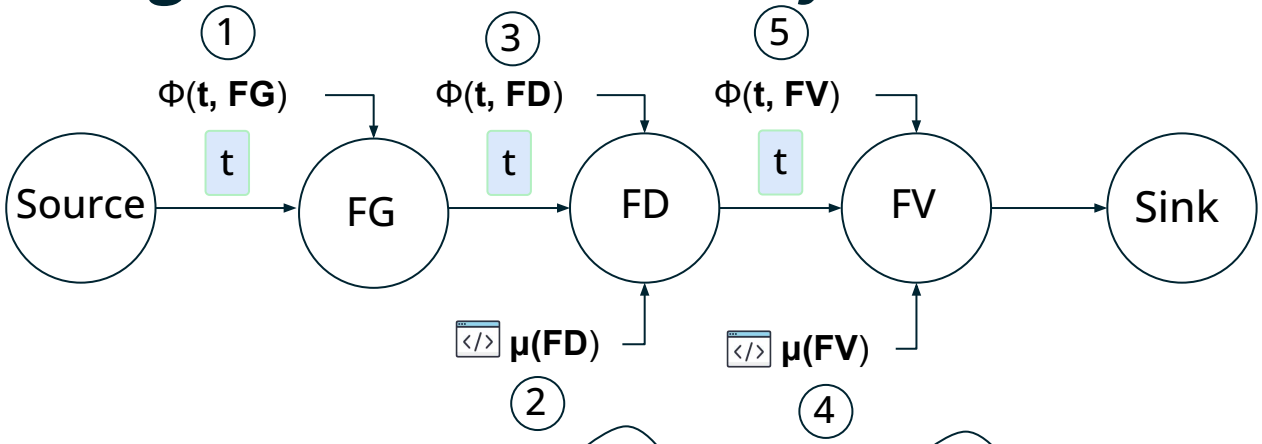
Guarantees same version!

Reconfiguration Consistency



Schedule S2 = [Φ(t, FG), μ(FD), Φ(t, FD), μ(FV), Φ(t, FV)]

Reconfiguration Consistency



Schedule S2 = [Φ(t, FG), μ(FD), Φ(t, FD), μ(FV), Φ(t, FV)]

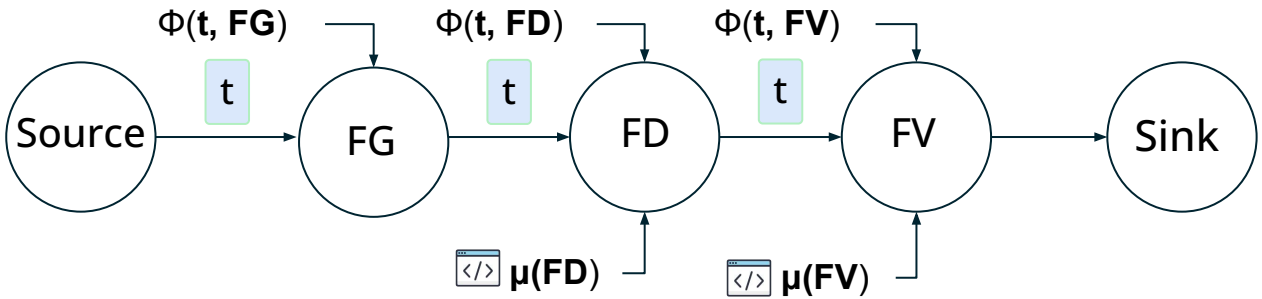
Conflict-Equivalent

Same version!

Serial Schedule S1 = [μ(FD), μ(FV), Φ(t, FG), Φ(t, FD), Φ(t, FV)]

Using new logic

How to generate consistent schedules?



For a schedule S:

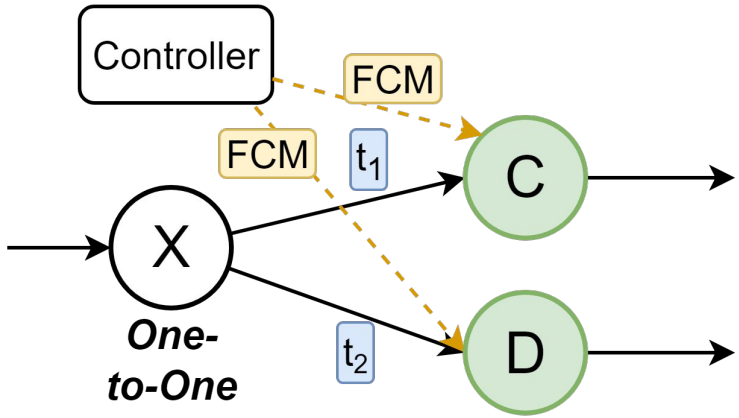
If Φ(t, FD) is before μ(FD),

Then Φ(t, FV) should be before μ(FV).

Requires synchronization!

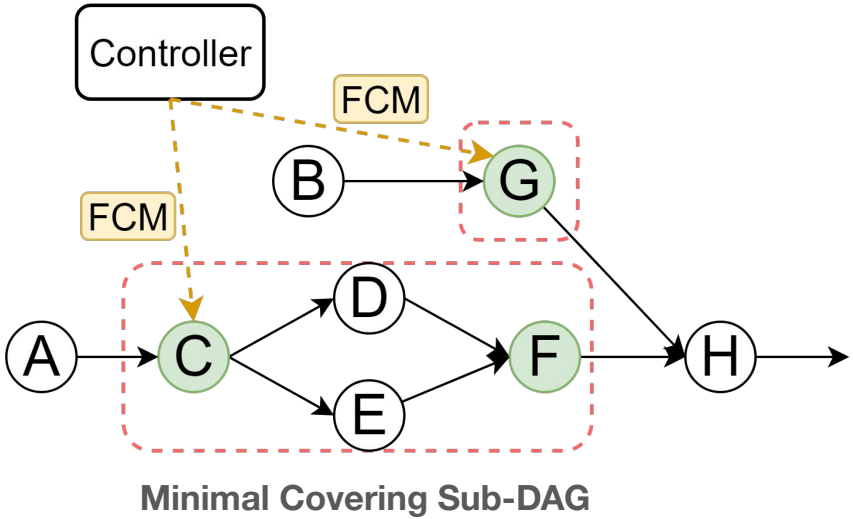
Guarantee Reconfiguration Consistency

Each output tuple goes to **either** C **or** D



No synchronization needed!

Fries Scheduler

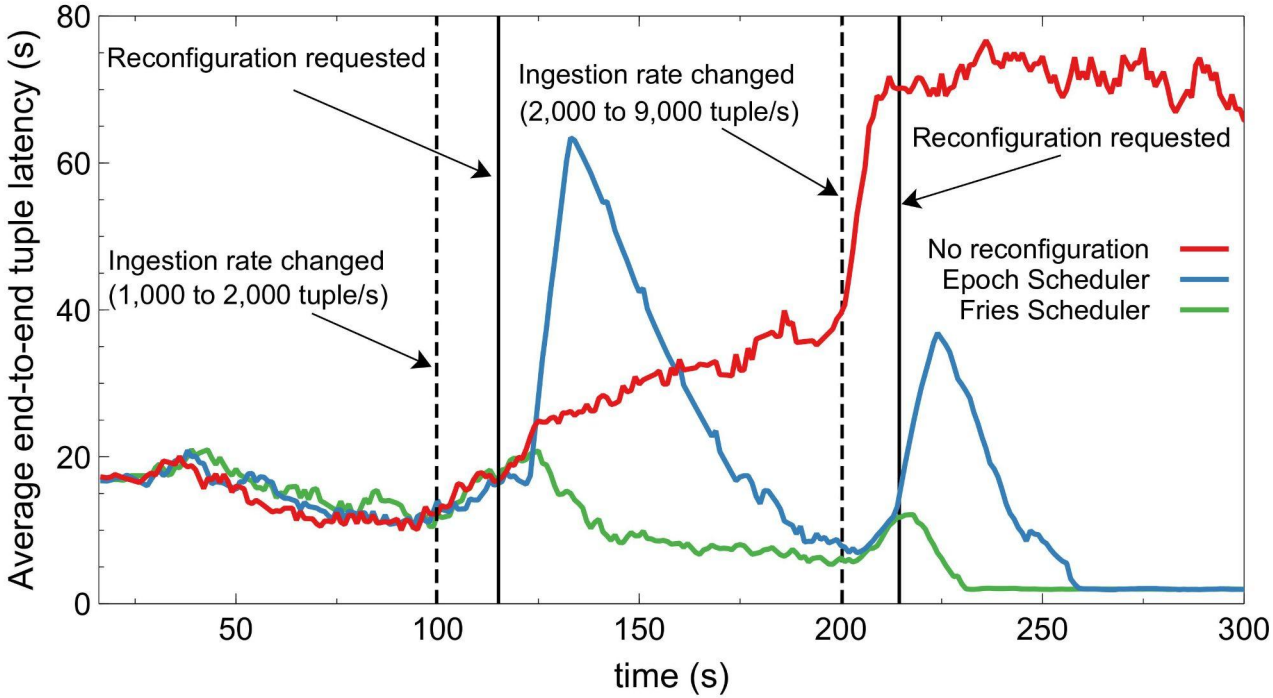


Finding the minimal scope for synchronization!

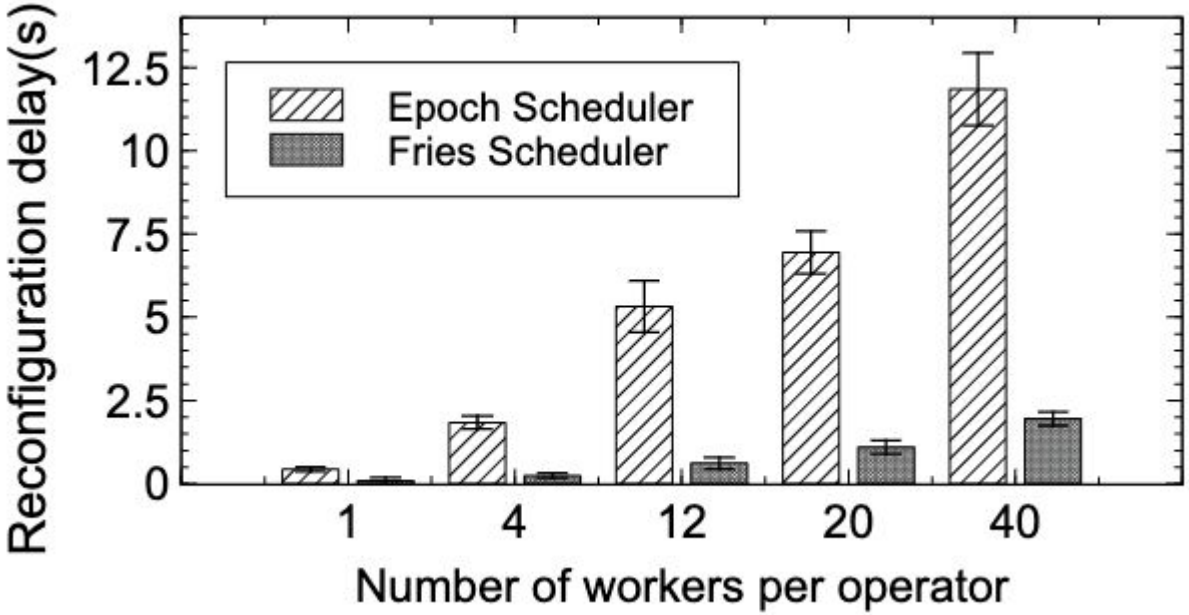
Experiments

1. Implemented on both Flink and Texera.
2. Fraud Detection, TPC-DS workflows.
3. 10 VMs on Google Cloud.

Benefits of Fast Reconfigurations



Scale out


Reconfiguration delay(s)
12.5
10
7.5
5
2.5
0
Epoch Scheduler
Fries Scheduler
1
4
12
20
40
Number of workers per operator

Our contributions

- Formally defined consistency in reconfiguration.
- Fries: Achieved both **Fast** and **Consistent** reconfigurations.
- Parallel execution, One-to-many operator…

Thank you!

Zuozhi Wang

Shengquan Ni

Avinash Kumar

Chen Li

UCI University of California, Irvine

Acknowledgements: We thank Sadeem Alsudais, Yicong Huang and the whole Texera team for their participation in discussions.