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
Big Data Workflows

Data → Long running workflow

Applied to:
- Apache Spark
- Flink
- texera
- Google BigQuery
- Snowflake
Motivation: Updating Logic in Big Data Workflows
Example Workflow

Source \rightarrow Feature Generator (FG) \rightarrow Fraud Detector (FD) \rightarrow Fraud Validator (FV) \rightarrow Sink

(Alice, 4am, $200, Banana)

(...untrusted_time = True)

(...fraud_prob = 0.96)

(...is_fraud = True)
Update Logic Case 1: Fixing runtime bugs

Source → Feature Generator (FG) → Fraud Detector (FD) → Fraud Validator (FV) → Sink

(...fraud_prob = 0.033) → (...is_fraud = True)
Update Logic Case 2: Mitigating surge of input data

Source → Feature Generator (FG) → Fraud Detector (FD) → Fraud Validator (FV) → Sink

- (...fraud_prob = 0.96) Uses “fraud_prob”
- (...fraud_level = “High”) Uses “fraud_level”
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

Reconfiguration

Source → FG → FD → FV → Sink

Epoch 1

reconfigure
Baseline 2: Reconfiguring between epochs

 Epoch 2

 Source → FG → FD → FV → Sink

 Epoch 1

 basline 2: Reconfiguring between epochs

 reconfigure
Baseline 2: Reconfiguring between epochs
Baseline 2: Reconfiguring between epochs

Source → FG → FD → New FV → Sink

Epoch 1

Epoch 2

reconfigure
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)

Apply new logic on t2, t3...

For multiple operators?
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: $\Phi(\text{tuple, operator})$

Function-update operation: $\mu(\text{operator})$
Reconfiguration Consistency

Data Transaction $T1 = [\Phi(t, FG), \Phi(t, FD), \Phi(t, FV)]$
Reconfiguration Consistency

Data Transaction $T_1 = [\Phi(t, FG), \Phi(t, FD), \Phi(t, FV)]$

Function-update Transaction $T_2 = \{\mu(FD), \mu(FV)\}$
Reconfiguration Consistency

Data Transaction \( T_1 = [\Phi(t, FG), \Phi(t, FD), \Phi(t, FV)] \)

Function-update Transaction \( T_2 = \{\mu(FD), \mu(FV)\} \)

How to resolve conflicts?
Reconfiguration Consistency

Serial Schedule $S_1 = [\mu(FD), \mu(FV), \Phi(t, FG), \Phi(t, FD), \Phi(t, FV)]$

Using new logic

Guarantees same version!
Reconfiguration Consistency

Schedule $S_2 = [\Phi(t, \text{FG}), \mu(\text{FD}), \Phi(t, \text{FD}), \mu(\text{FV}), \Phi(t, \text{FV})]$
Reconfiguration Consistency

Schedule S2 = [\( \Phi(t, \text{FG}) \), \( \mu(\text{FD}) \), \( \Phi(t, \text{FD}) \), \( \mu(\text{FV}) \), \( \Phi(t, \text{FV}) \)]

Serial Schedule S1 = [\( \mu(\text{FD}) \), \( \mu(\text{FV}) \), \( \Phi(t, \text{FG}) \), \( \Phi(t, \text{FD}) \), \( \Phi(t, \text{FV}) \)]

Conflict-Equivalent

Using new logic

Same version!
How to generate consistent schedules?

For a schedule S:
If \( \Phi(t, FD) \) is before \( \mu(FD) \),
Then \( \Phi(t, FV) \) should be before \( \mu(FV) \).

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

![Graph showing benefits of fast reconfigurations](image)
Scale out
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

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