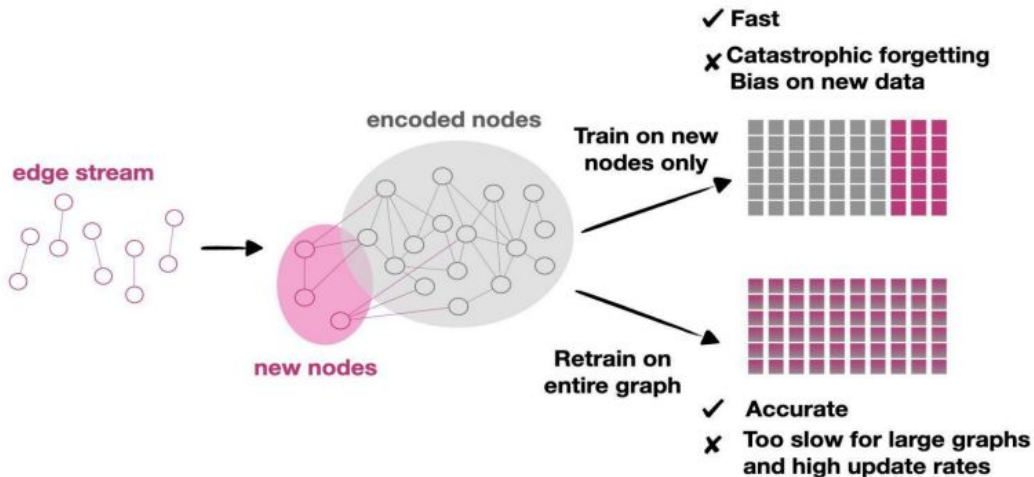


# Online training of streaming Graph Neural Networks

Team 2: Adrish, Aoming, Baicheng, Iasonas, Yuhang

*\*Ordered Alphabetically*

# Project Overview



- Retraining Graph Neural Networks on larger than memory graphs
- Training stream: new nodes, new edges
- Inference stream: nodes with 1-hop neighborhood information
- Task: Predict which of 41 reddit communities created the post

# Project Overview (High-level)

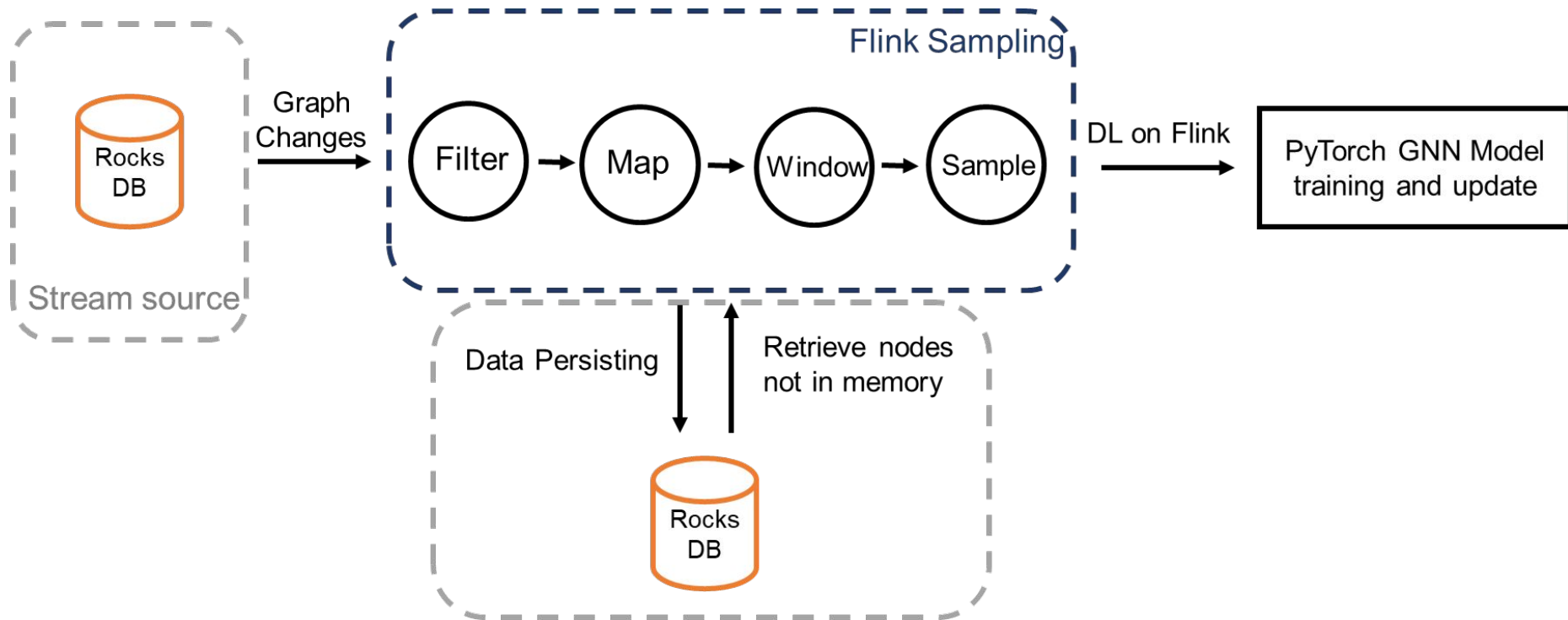
## Training Pipeline:

- Ingest graph changes (nodes/edges) stream
- Retrain neural network
- Update the model
- Serve the model

## Inference Pipeline:

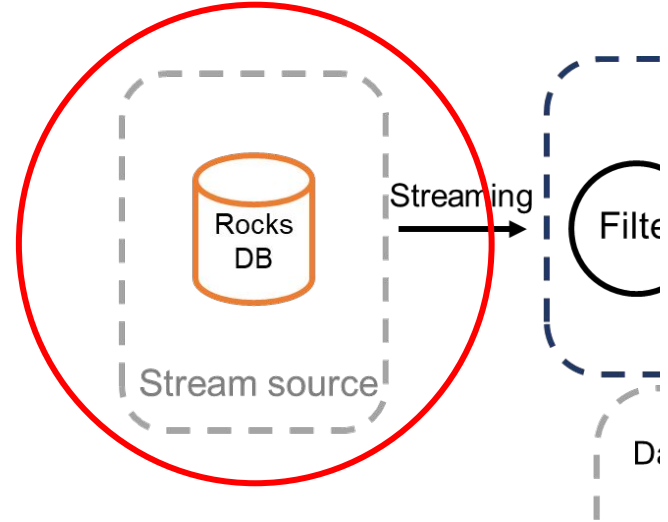
- Ingest nodes stream
- Run inference

# Overview



# Graph Database

- 3 Databases (Nodes, Edges, Neighbor Index)
- Stores the Reddit Dataset (<https://arxiv.org/abs/1706.02216>)
- **Nodes:** Word2Vec embedding of Reddit Posts
- **Edges:** If same user leaves comment on two posts



# Graph Database Schema

Node Table	
NodeID	Features
1	[ 1 21 3.6 ... 2.1]
2	[ ... ]
3	[ ... ]
...	
221769	[ ... ]

3 Databases: **NodesDB**, EdgesDB, NeighborIndexDB

NodeDB:

- Key: NodeID
- Value (ByteArray): <2 Bytes Mask><4 Bytes Label><feature bytes>

# Graph Database Schema

Node Table		Edge Table	
NodeID	Features	EdgeID	Value
1	[ 1 21 3.6 ... 2.1]	1 0	N/A
2	[ ... ]	1 1	N/A
3	[ ... ]	1 2	N/A
...		2 0	N/A
221769	[ ... ]	...	...

3 Databases: NodesDB, **EdgesDB**, NeighborIndexDB

EdgesDB:

- Key: "<SourceNodeID>|<TargetNodeID>"
- Value: N/A

Insert Comparator: Sort by source node first, and failsafe, sort by incoming sequence

# Graph Database Schema

Node Table		Edge Table		Neighbor Index Table	
NodeID	Features	EdgeID	Value	NodeID	Edge
1	[ 21    1.0   3.6   ...   2.1]	1 0	N/A	1	1 0
2	[ ... ]	1 1	N/A	2	2 0
3	[ ... ]	1 2	N/A	3	3 1
...		2 0	N/A	...	
221769	[ ... ]	...	...	235	235 3

3 Databases: NodesDB, EdgesDB, **NeighborIndexDB**

NeighborSizeDB:

- Key: NodeID
- Value: First Edge of Neighborhood



# New DB Schema

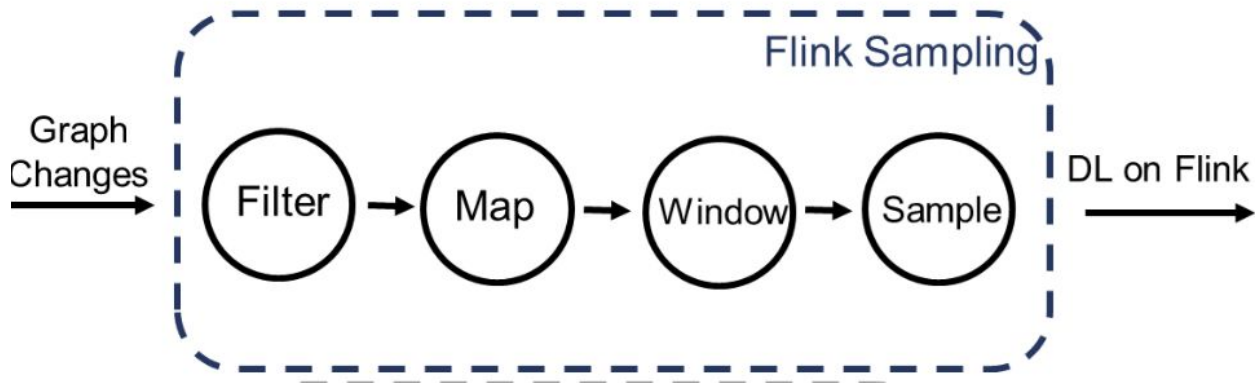
Node Table		Edge Table		Neighbor Size Table	
NodeID	Features	EdgeID	Value	NodeID	Edge
1	[ 21    1.0   3.6   ...   2.1]	1 0	0	1	4
2	[ ... ]	1 1	3	2	3
3	[ ... ]	1 2	4	3	5
...		2 1	0	...	
221769	[ ... ]	...	...	235	2

- Faster 1 Hop Neighborhood Query (Neighborhood Always start at (node, 0))
- Slower Insertion. (Query max size, add 1, create new key, add edge).

# Java Interface for RocksDB

- `FindFeatures(nodeId) -> Tuple3<Mask, Label, Embedding>`
- `FindNeighbors(nodeId) -> ArrayList<Int> (neighbors)`
- `InsertNode(nodeId)`
- `RandomNodes() -> ArrayList(Tuple5<NodeID, Mask, Label, Embedding, Neighbors>)`

# Flink operators



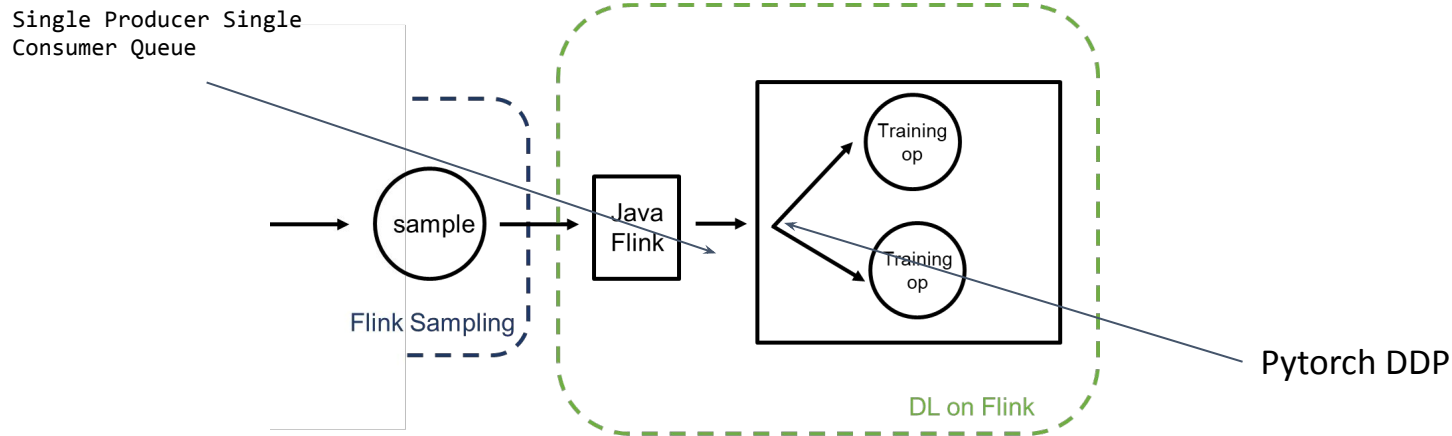
- Input: `Tuple5<NodeID, Mask, Label, Embedding, Neighbors>`
- Now: train only on “fresh” nodes
- Future: use “fresh” and “old” nodes
  - Reservoir sampling

# Output of Sampling Operator

NodeID	Mask [0, 1, 2]	Label	Embedding	Neighbor
1	0	4	[1 21 3.6 ... 2.1 -1.7], [...], [...], [...]	2, 3, 5

- **NodeId** (Integer)
- **Mask**: train, validation, test (Short)
- **Label** (Integer)
- **Embedding**: feature vectors (List<Byte>)
- **Neighbor**: neighbors (String)

# ML Operator



- Connect Flink with Pytorch using **DL-on-Flink\***
- Recap:
  - Ingest data
  - Calculate samples
  - Send samples to Pytorch

\*[github.com/flink-extended/dl-on-flink](https://github.com/flink-extended/dl-on-flink)

# ML - PyG

Base Lib: **PyTorch Geometric (PyG)**

Data Structure: `torch_geometric.data`

- `data.x`: node features
- `data.edge_index`: graph edges
- `data.y`: node labels
- `data.train_mask(/val/test)`: source node only

Networks: `torch_geometric.nn`

- `nn.GCNConv`: only using 2 GCNConv for demo



# ML pre-processing

## Pre-processing: Re-indexing:

NodeID	Mask [0, 1, 2]	Label	Embedding	Neighbor
1	0	4	[3.6 ... 2.1 -1.7], [...], [...], [...]	2, 3, 5

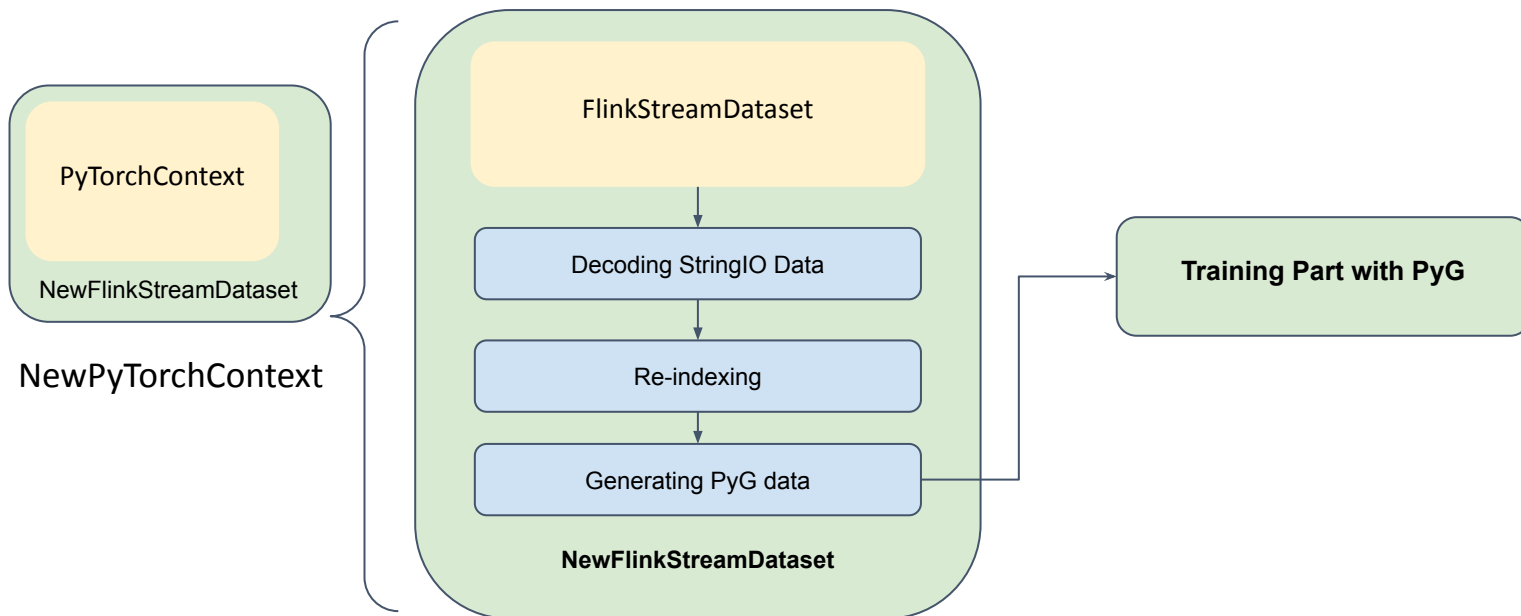
Re-indexing map: {1 : 0 ; 2 : 1 ; 3 : 2 ; 5 : 3}

Re-indexing

NodeID	Mask [0, 1, 2]	Label	Embedding	Neighbor
0	0	4	[3.6 ... 2.1 -1.7], [...], [...], [...]	1, 2, 3

# ML pipeline overview

Processing ML pipeline overview (based on dl-on-flink)





# Demonstration of working components

The screenshot displays the Apache Flink Dashboard interface. On the left is a dark sidebar with navigation links: Overview, Jobs, Running Jobs, Completed Jobs, Task Managers (highlighted), Job Manager, and Submit New Job. The main content area shows the details for a specific Task Manager identified by the address `akka.tcp://flink@172.17.0.2:40302/user/rpc/taskmanager_0`. At the top right of the main area, the dashboard version (1.14.4), commit hash (895c609), and timestamp (2022-02-25T11:57:14+01:00) are shown, along with a message icon. Below the address bar, a summary row provides key metrics: Last Heartbeat (2023-03-23 15:43:08), ID (172.17.0.2:40302-3fb0e2), Data Port (41384), Free Slots / All Slots (2 / 2), and CPU Cores (12). A second row shows Physical Memory (62.7 GB), JVM Heap Size (512 MB), and Flink Managed Memory (512 MB). Below this, tabs for Metrics, Logs, Stdout (selected), Log List, and Thread Dump are visible. The Stdout tab displays a log stream with line numbers on the left and log messages on the right. A search bar at the top right of the log area contains the text 'loss'. The log messages show the Task Manager processing data batches, including indices, feature strings, and data batch information. A specific line (1631) shows a 'loss' value of 0.000000.

Apache Flink Dashboard

Version: 1.14.4 Commit: 895c609 @ 2022-02-25T11:57:14+01:00 Message: 0

akka.tcp://flink@172.17.0.2:40302/user/rpc/taskmanager\_0

Last Heartbeat: 2023-03-23 15:43:08 ID: 172.17.0.2:40302-3fb0e2 Data Port: 41384 Free Slots / All Slots: 2 / 2 CPU Cores: 12

Physical Memory: 62.7 GB JVM Heap Size: 512 MB Flink Managed Memory: 512 MB

Metrics Logs **Stdout** Log List Thread Dump

```
1607 4402
1608 Index(['src', 'label', 'nbr', 'embed'], dtype='object')
1609 feats_str: <class 'bytes'>
1610 ('length and node num', 2176, 16)
1611 the data we are getting DataBatch(x=[16, 17], edge_index=[2, 15], y=[16], train_mask=[1], val_mask=[1], test_mask=[1], batch=[16], ptr=[2])
1612 27 0.0
1613 825
1614 Index(['src', 'label', 'nbr', 'embed'], dtype='object')
1615 feats_str: <class 'bytes'>
1616 ('length and node num', 408, 3)
1617 the data we are getting DataBatch(x=[3, 17], edge_index=[2, 2], y=[3], train_mask=[1], val_mask=[1], test_mask=[1], batch=[3], ptr=[2])
1618 28 2.6205501556396484
1619 1100
1620 Index(['src', 'label', 'nbr', 'embed'], dtype='object')
1621 feats_str: <class 'bytes'>
1622 ('length and node num', 544, 4)
1623 the data we are getting DataBatch(x=[4, 17], edge_index=[2, 3], y=[4], train_mask=[1], val_mask=[1], test_mask=[1], batch=[4], ptr=[2])
1624 ##### /tmp/1679600158447-1/ml_on_flink_160660038471390973/train.py
1625 3298
1626 Index(['src', 'label', 'nbr', 'embed'], dtype='object')
1627 feats_str: <class 'bytes'>
1628 ('length and node num', 1632, 12)
1629 the data we are getting DataBatch(x=[12, 17], edge_index=[2, 11], y=[12], train_mask=[1], val_mask=[1], test_mask=[1], batch=[12], ptr=[2])
1630 0 0.0
1631 rank: 1 batch: 0 loss: 0.000000
1632 9057
1633 Index(['src', 'label', 'nbr', 'embed'], dtype='object')
1634 feats_str: <class 'bytes'>
```

# Challenges

- Designing the RocksDB schema Optimized for Neighborhood Query
- Debugging dl-on-flink (The Java Flink / Python Interface)

# Immediate future work

- Replace the streaming source
- Implement the reservoir sampling algorithm
- Fault-tolerance: flink-state, checkpoints
- Evaluate the performance of our model
- Deploy the model in a production environment
- Serve updated trained model as a Kafka Pub/Sub System.