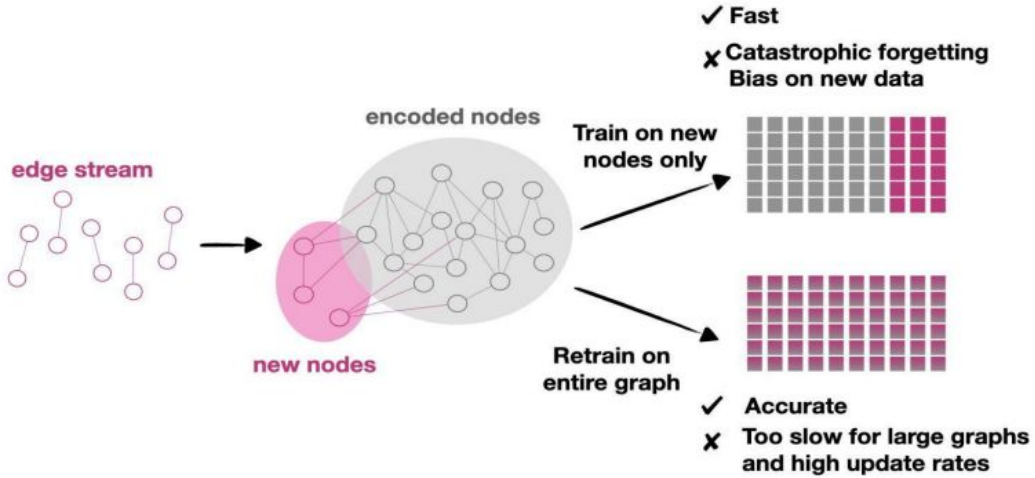
Online training of streaming Graph Neural Networks

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

*Figure from Project Description

Project Overview (High-level)

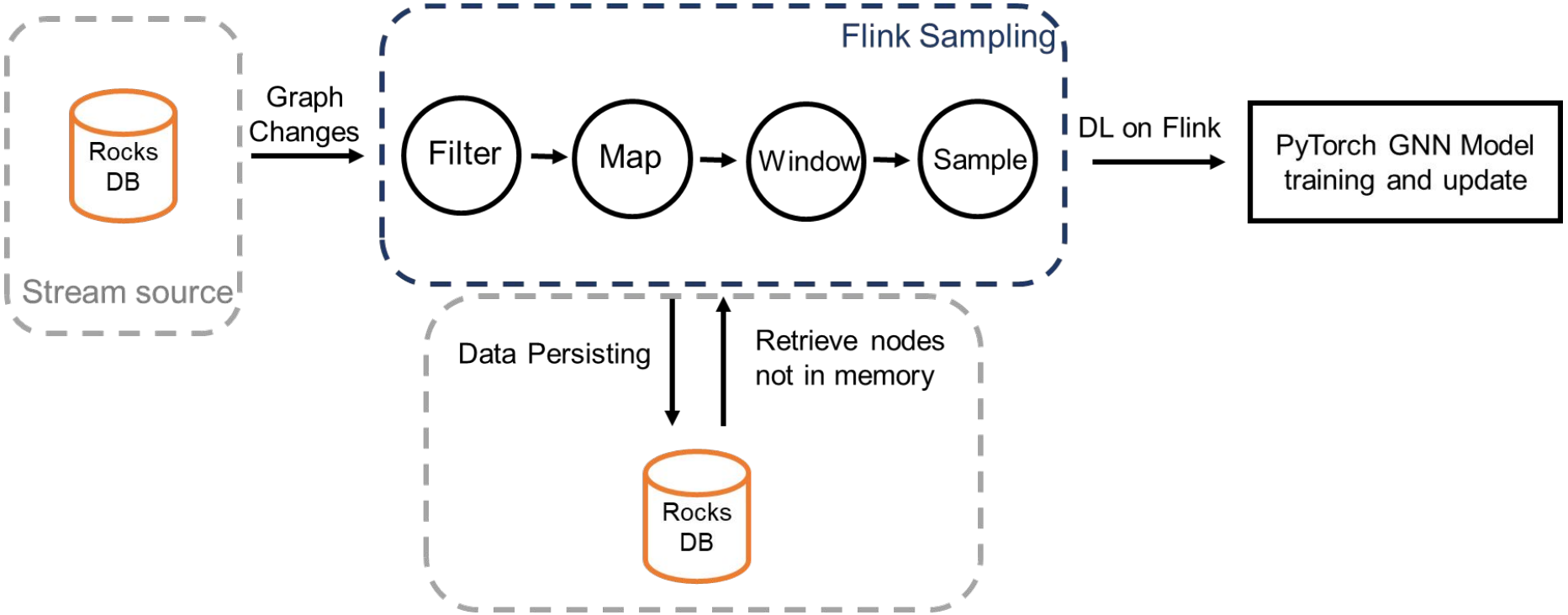
Training Pipeline:

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

Inference Pipeline:

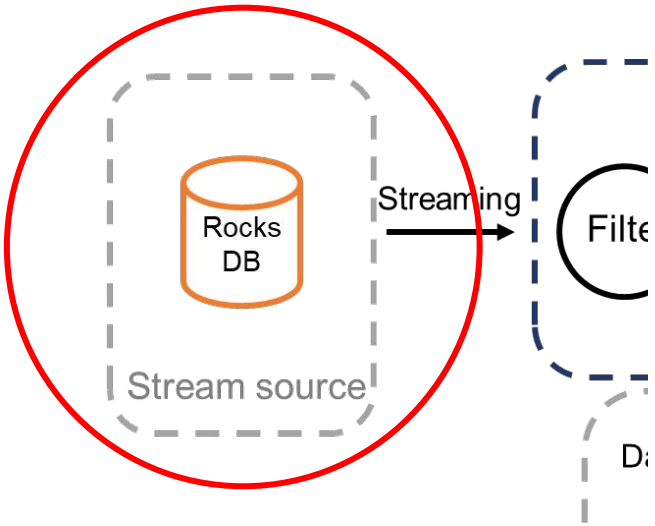
- Ingest nodes stream
- Run inference

Overview


Rocks DB
Stream source
Graph Changes
Flink Sampling
Filter
Map
Window
Sample
DL on Flink
PyTorch GNN Model training and update
Data Persisting
Retrieve nodes not in memory
Rocks DB

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 (BytesArray): <2 Bytes Mask><4 Bytes Label><feature bytes>

Graph Database Schema

Node Table

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

Edge Table

EdgeID	Value
1|0	N/A
1|1	N/A
1|2	N/A
2|0	N/A
…	…

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

NodeID	Features
1	[21 1.0 3.6 ... 2.1]
2	[…]
3	[…]
…	
221769	[…]

Edge Table

EdgeID	Value
1|0	N/A
1|1	N/A
1|2	N/A
2|0	N/A
…	…

Neighbor Index Table

NodeID	Edge
1	1|0
2	2|0
3	3|1
…	
235	235|3

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

NeighborSizeDB:

- Key: NodeID
- Value: First Edge of Neighborhood

New DB Schema

Node Table

NodeID	Features
1	[21 1.0 3.6 ... 2.1]
2	[…]
3	[…]
…	
221769	[…]

Edge Table

EdgeID	Value
1|0	0
1|1	3
1|2	4
2|1	0
…	…

Neighbor Size Table

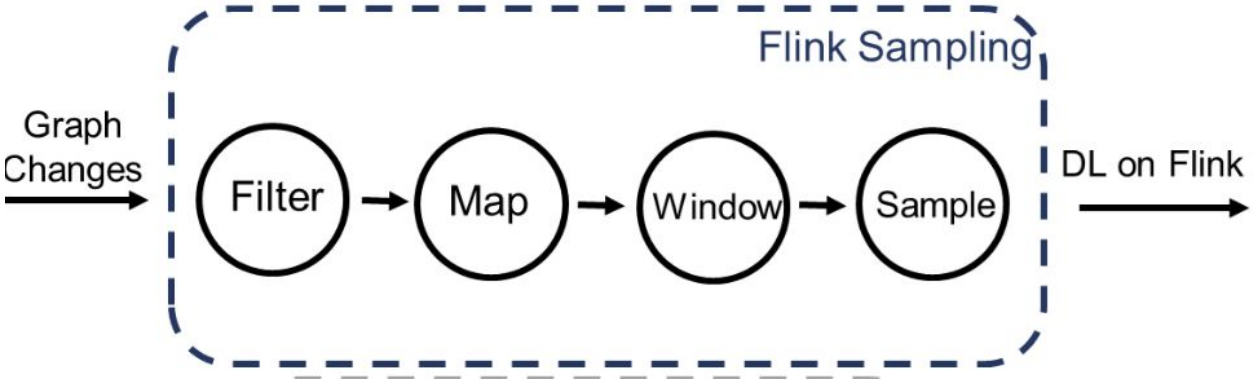
NodeID	Edge
1	4
2	3
3	5
…	
235	2

- 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



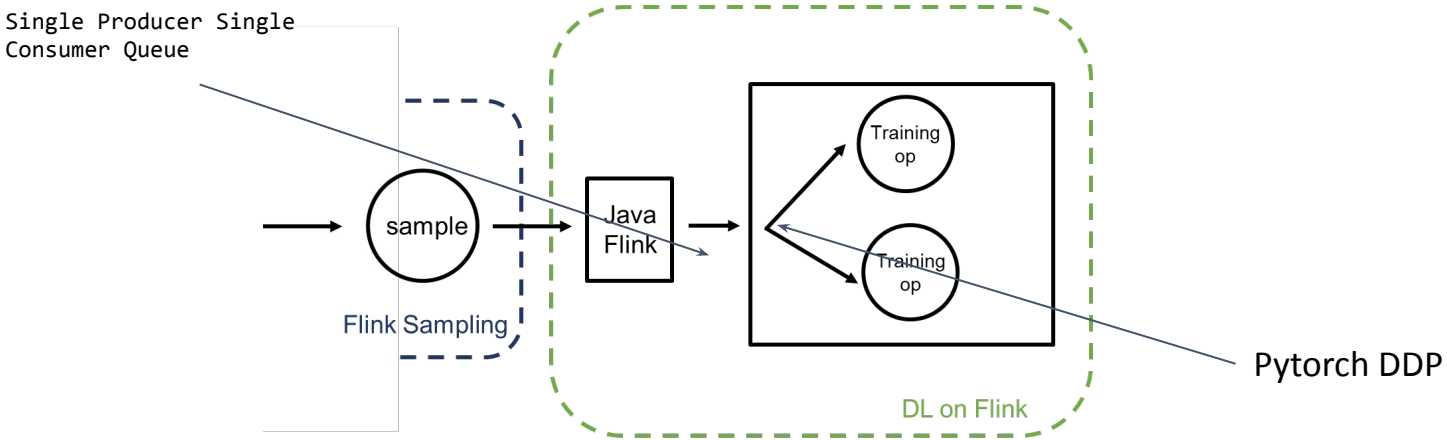
- Input: Tuple5<NodeID, Mask, Label, Embedding, Neighbors>
- Now: train only on “fresh” nodes
- Future: use “fresh” and “old” nodes
  - Reservoir sampling

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

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 GCNConvs for demo

ML pre-processing

Pre-processing: **Re-indexing:**

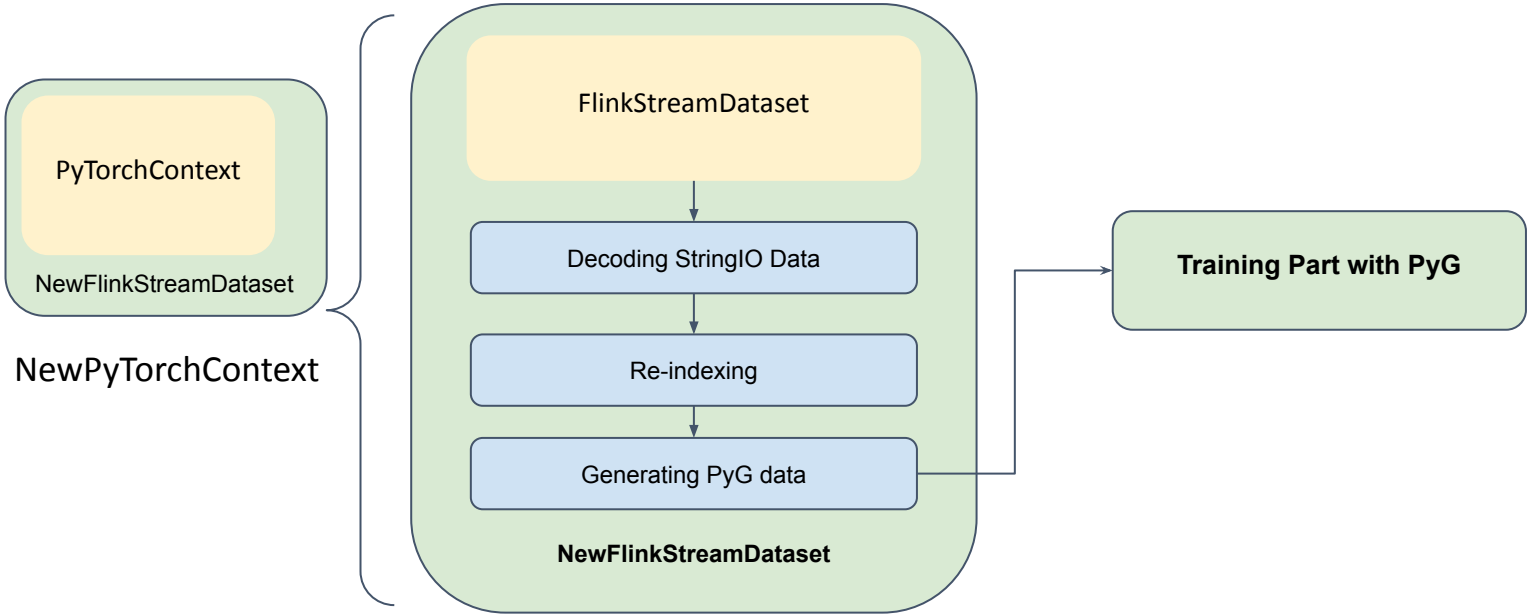
NodeID	Mask [0, 1, 2]	Label	Embedding	Neighbor
1	0	4	[3.6 ... 2.1 -1.7], [...], [...], [...]	2, 3, 5

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

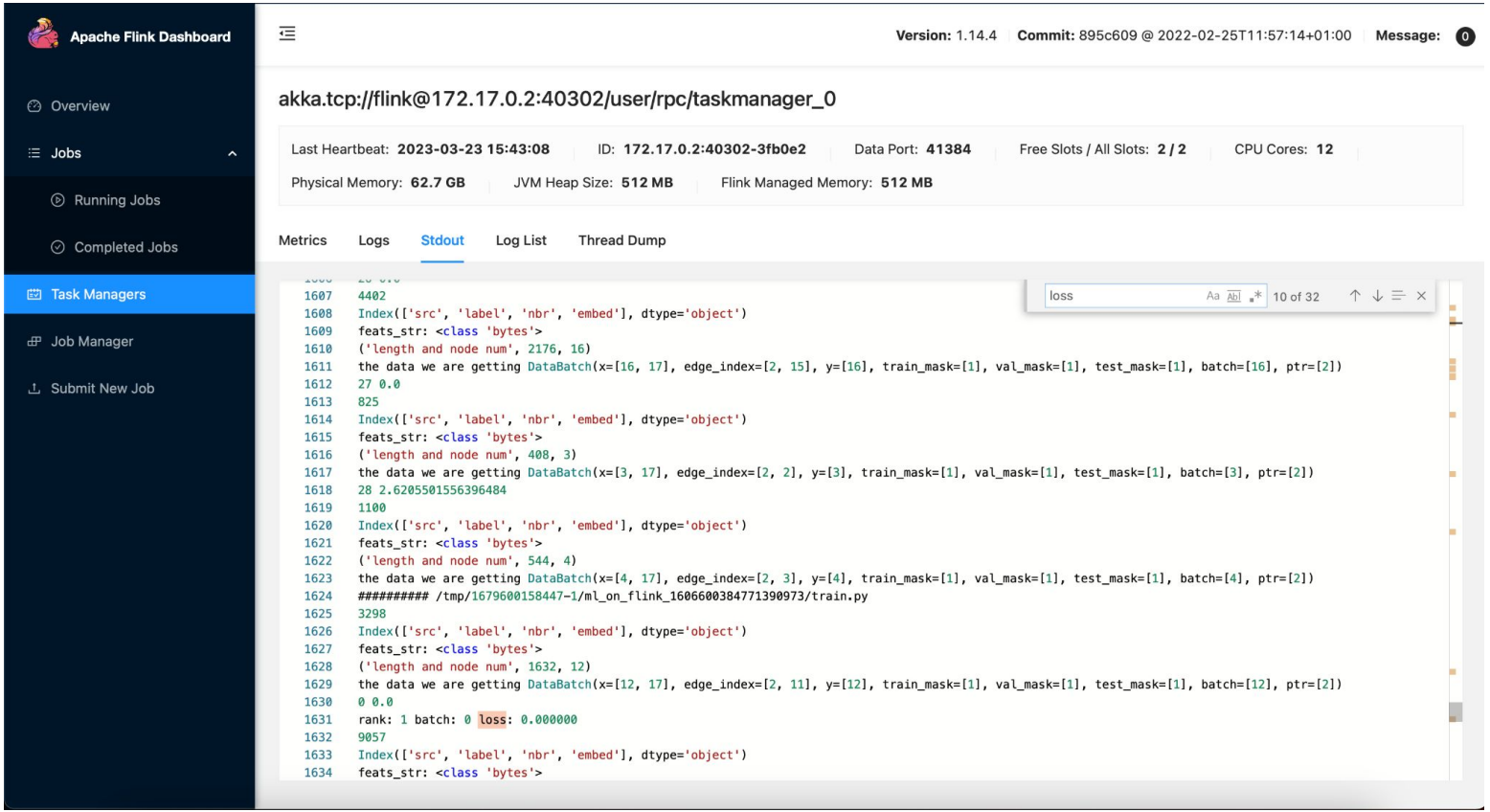
NodeID	Mask [0, 1, 2]	Label	Embedding	Neighbor
0	0	4	[3.6 ... 2.1 -1.7], [...], [...], [...]	1, 2, 3

ML pipeline overview

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



Demonstration of working components



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.