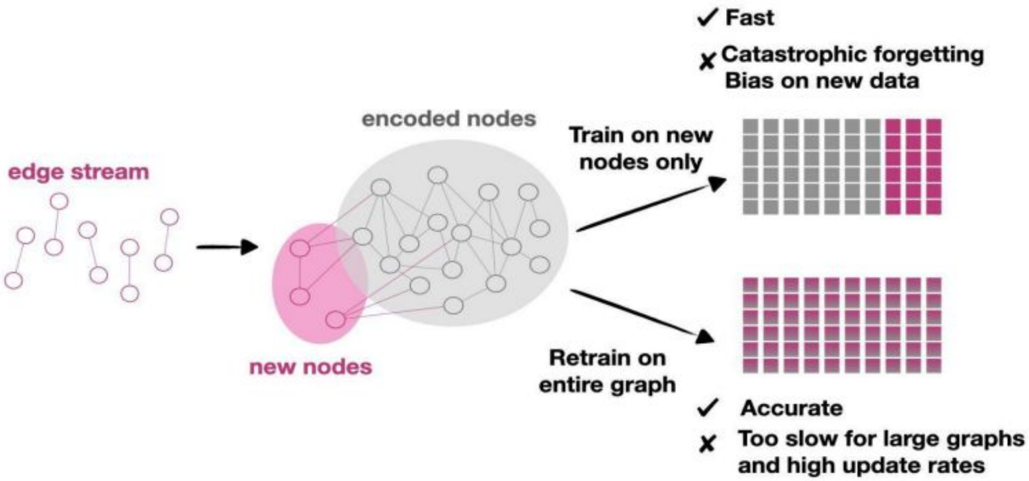
Online training of Graph Neural Networks

Team 2

Adrish, Aoming, Baicheng, Iasonas, Yuhang

Project Overview



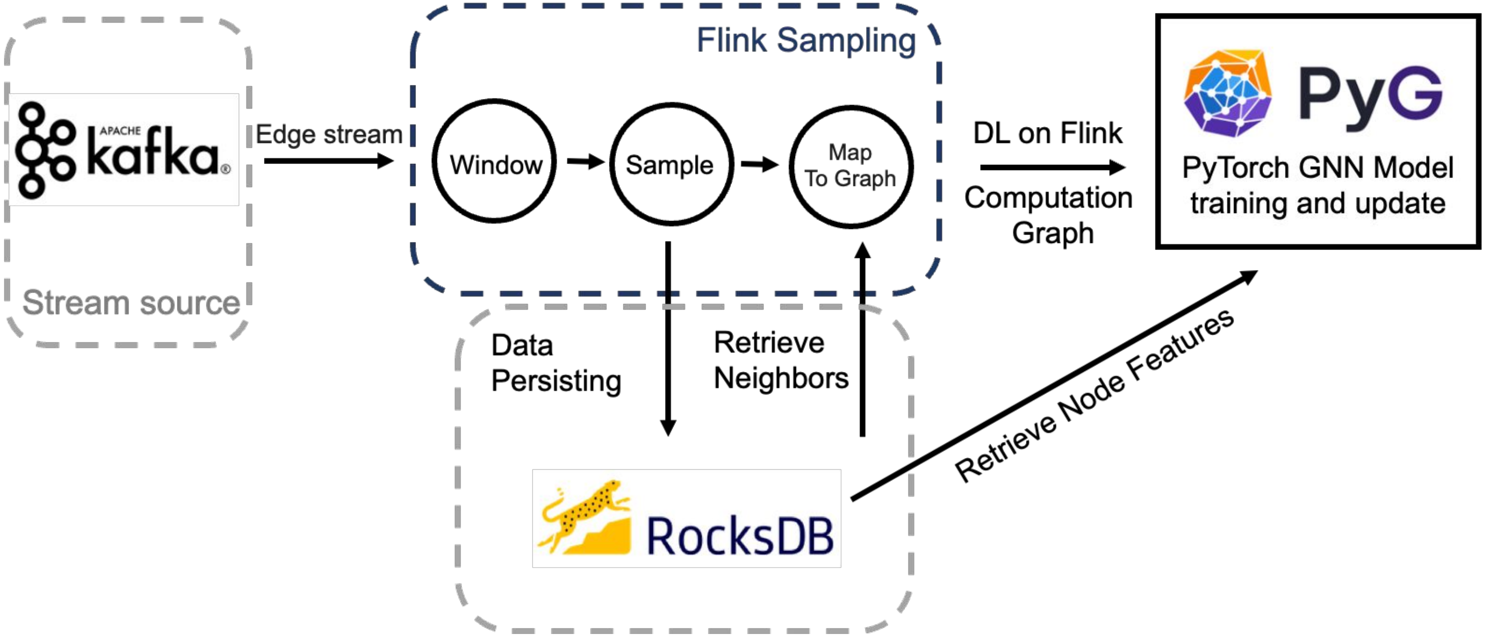
- Retraining GNNs on larger than memory graphs
- Stream: edges with labels on the nodes
- Task: predict the labels of the nodes

*Figure from Project Description

Achievements

- Build a flink pipeline for online training GNN's
- Use RocksDB to store a graph (nodes, edges, features, labels)
- Experiment with basic parameters of our pipeline
    - Latency
    - Memory consumption
    - Accuracy (of our model)
    - We still need more experiments

Demonstration

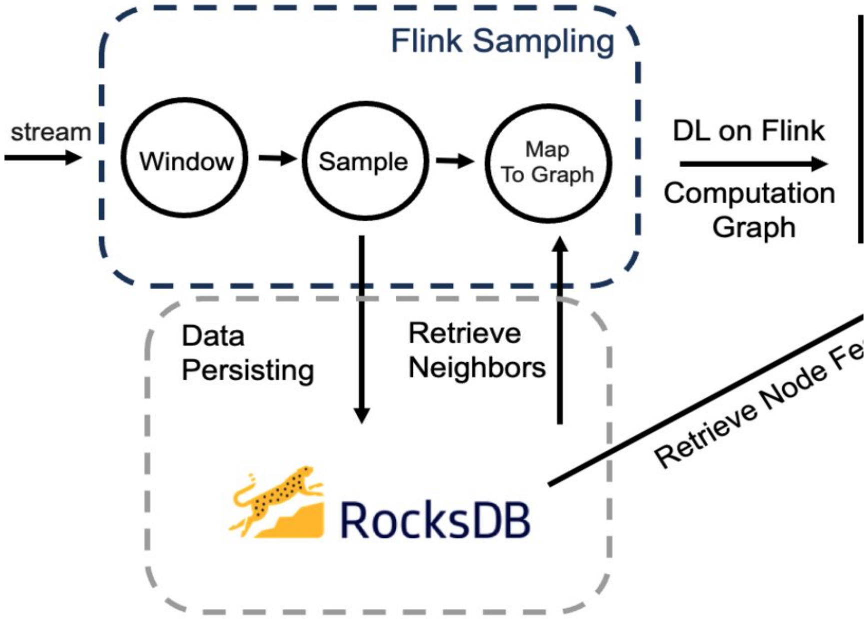


Demonstration - Data source

- PubMed dataset (`https://paperswithcode.com/dataset/pubmed`)
  - ~20000 scientific publications
  - Each publication is classified into 1 of 3 classes
  - Each publication described by a TF/IDF weighted word vector
  - ~45000 links
- Kafka source
  - Rate control
  - Protobuf message

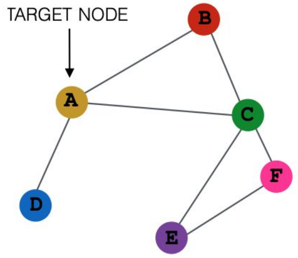
Demonstration - Flink main pipeline

- **Window**: wait for a significant number of edges to arrive
  - Parameter: window size
- **Sample**: sample old and new nodes for training
  - Parameter: number of samples
- **MapToGraph**: compute the computation graph of each node

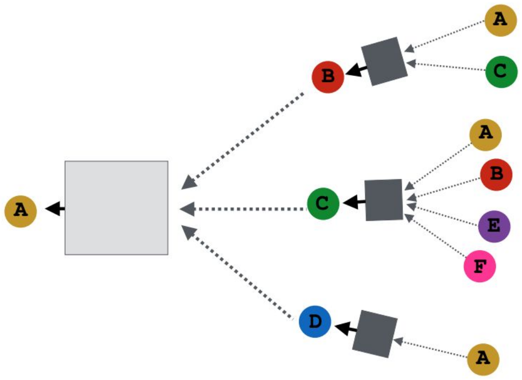


Demonstration - Flink main pipeline

- **MapToGraph**: compute the computation graph each node
- Reservoir sampling
- Parameters:
    - # neighbors
    - Depth (# hops)



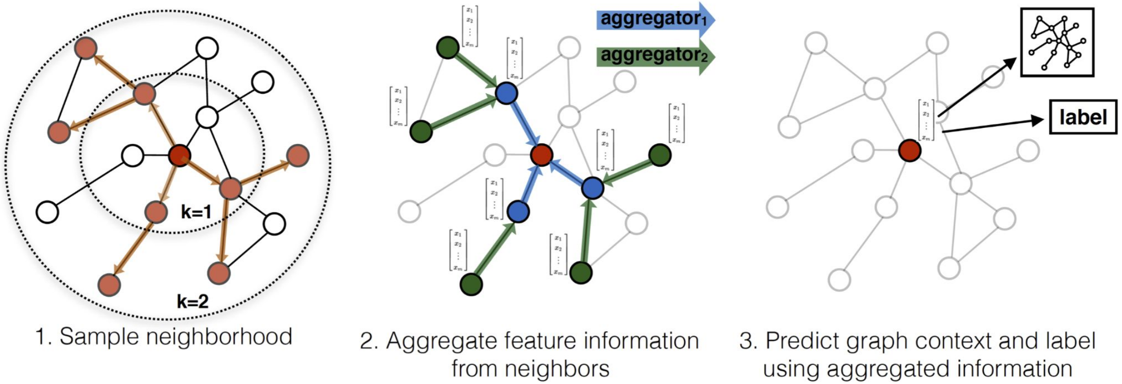
INPUT GRAPH



Demonstration - Pytorch Pretraining

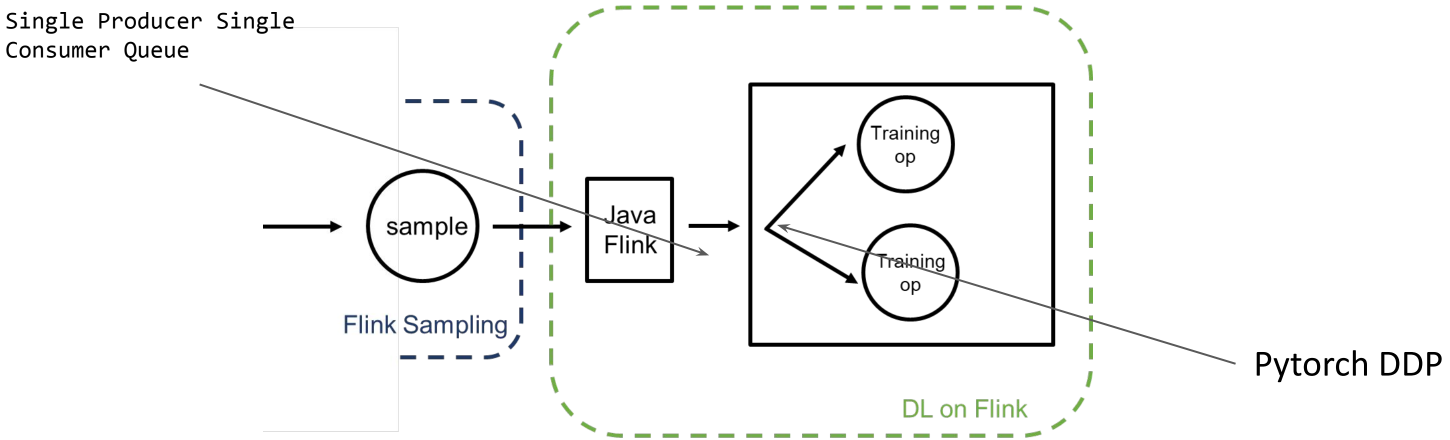
- Model: **GraphSage** (https://arxiv.org/abs/1706.02216)
- Pretraining: on **a subgraph with ~20% nodes** (~5% edges)
- Stream the rest of the edges (~95%)
- Graph Learning Lib: **PyTorch Geometric**

PyG



GraphSage

Demonstration - ML Operator



- Connect Flink with Pytorch using **DL-on-Flink***
- PyTorch Dataloading: **PyG Dataloader + IterableDataset**

*github.com/flink-extended/dl-on-flink

Demonstration - Pytorch Dataloading

- Get the source node and the sampled subgraph (edges):

  Data 1: {src: 1, edges: 1-2|2-3} Data 2 :{src: 2, edges: 2-1|2-3|3-4}

- Feature Extraction: Getting feature for all nodes from disk

  Feature 1: [x_1, x_2, x_3] Feature 2 :[x_1, x_2, x_3, x_4]]

- Reindexing the subgraph:

  Data 1: {src: 0, edges: 0-1|1-2} Data 2 :{src: 1, edges: 1-0|1-2|2-3}

- Forming data batch (PyG Dataloader):

  Data Batch: { Features: [x_1, x_2, x_3, x_1, x_2, x_3, x_4],
  Edges: 0-1|1-2|4-3|4-5|5-6,
  Train/Val/Test mask: [True, False, False, True, False, False, False]
  Batch Indexs: [0, 0, 0, 1, 1, 1, 1]}

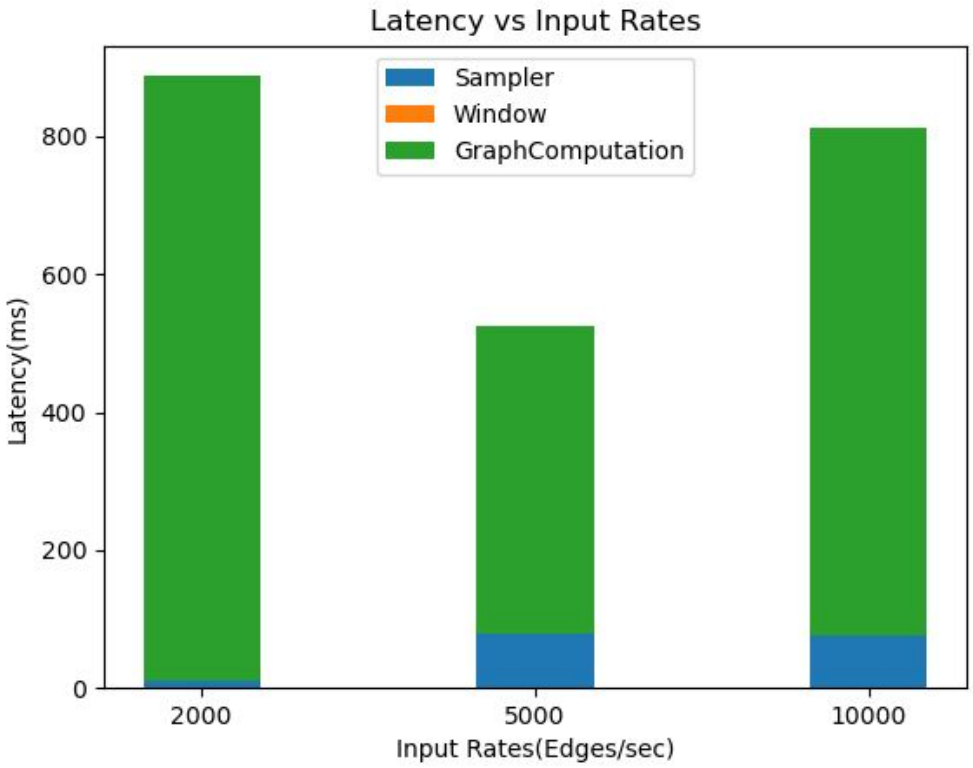
Experimental results

- Latency
- Memory consumption
- Accuracy (of the GNN)

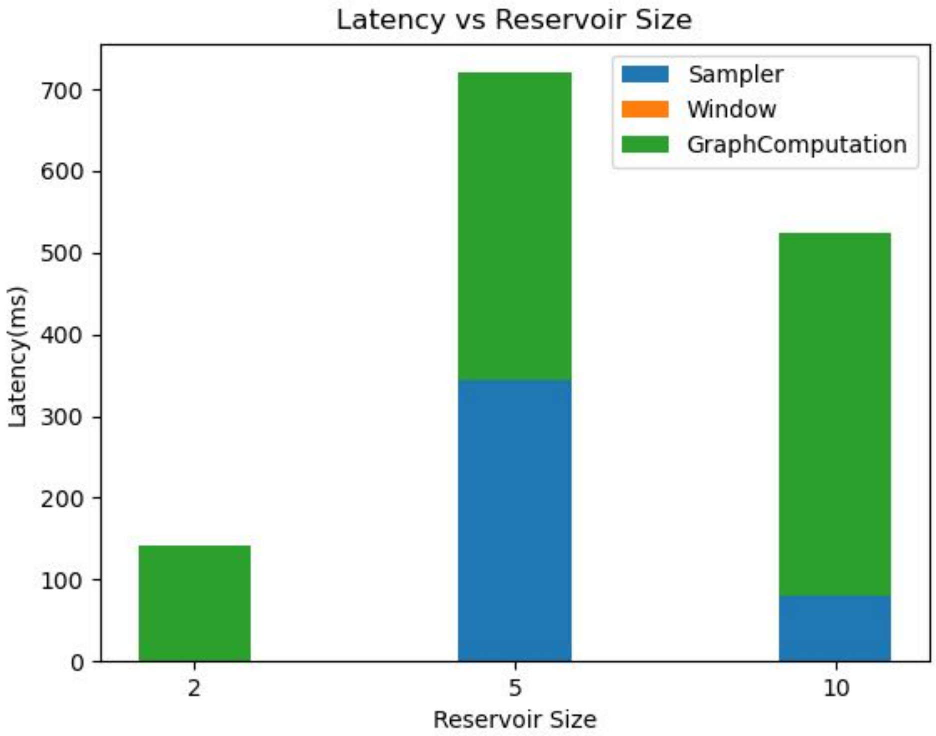
Parameters:

- # of samples
- # of neighbors / node
- # of hops in the computation graph

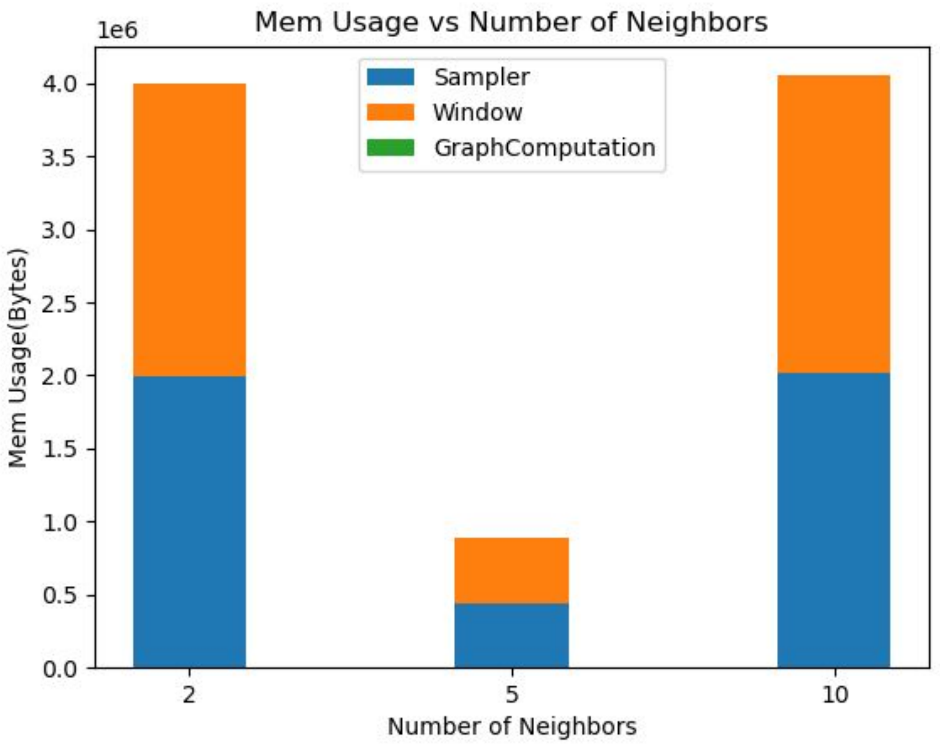
Experimental results - Latency



Experimental results - Latency

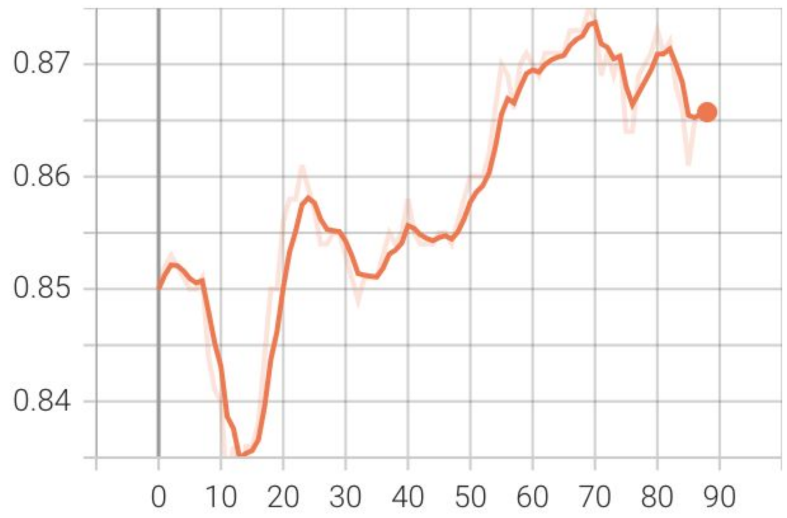


Experimental results - Memory consumption

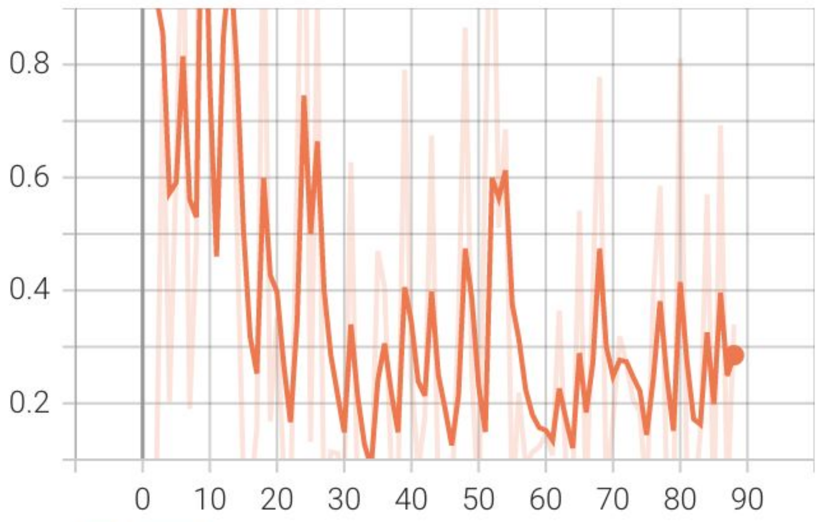


Experimental results - Accuracy/Loss

validation/acc
tag: validation/acc



training/loss
tag: training/loss



Challenges

- Connecting Flink with Python (+ DB)
- Designing a database schema to store the graph
  - Support fast queries
- Dumping the dataset in rocksDB
  - Big datasets require hours
- Sampling the neighborhood of a node
  - Reservoir sampling

Experience

- Learn the basics of streaming systems
- Flink
- Graph Neural Networks (GraphSage)
- Split a project into discrete tasks
- Collaboration, team-work