CS 744: DISTRIBUTED DGL

Shivaram Venkataraman

Fall 2022

ADMINISTRIVIA

- Midterm grades out!
- Regrade requests (check question numbers)
    - Thu: After class, Roger's OH
    - Mon: Shivaram's OH, Roger's OH
    - Tue: After class

- Course Project: Check in by Nov 23th → Canvas / Piazza today

EXAMPLE: LINK PREDICTION

Machine learning on graph data

Task: Predict potential connections in a social network

 [0.25, 0.45, 0.30]

 [0.15, 0.85, 0.92]

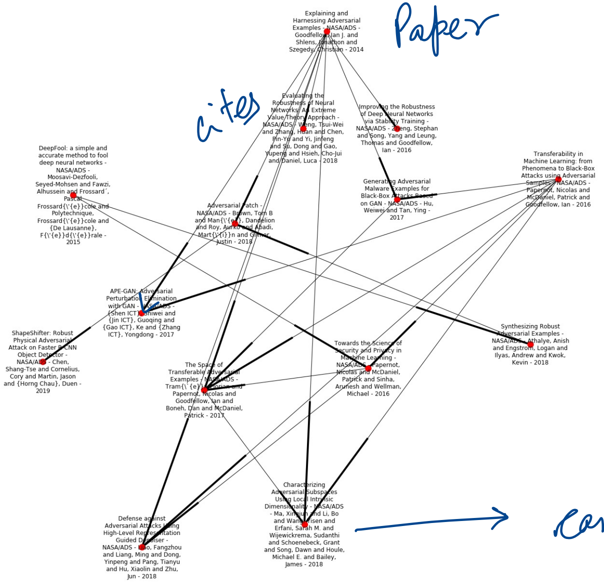
…

Find K-nearest neighbors

EXAMPLE: NODE CLASSIFICATION

Subsets of nodes have labels

Task: Classify papers in a citation graph by subject area



Paper

cites

training vertices

embeddings [0.25, 0.45, 0.30]

[0.15, 0.85, 0.92]

…

physics

math

can we predict its label

supervised learning

From: https://github.com/lanstonchu/citation-graph

GRAPH EMBEDDING MODELS: DECODER-ONLY

Loss function

Maximize score for edges in graph

Minimize for others (negative edges)

$$\mathcal{L} = \sum_{e \in G} \sum_{e' \in S'_e} \max(f(e) - f(e') + \lambda, 0))$$

positive negative edges

BACKGROUND: GRAPH NEURAL NETWORKS (GNN)

Graph Neural Networks: Use neural network to capture neighborhood structure

Input: h_i^0 (base node representations)

Model: $h_i^k = AGG(h_i^{k-1}, \{h_u^{k-1} : u \in N_i\})$

N_i one-hop neighborhood of *i*

AGG parameterized aggregation func

2 - hop nbr hood

↳ 1 hop nbrhood of nbr

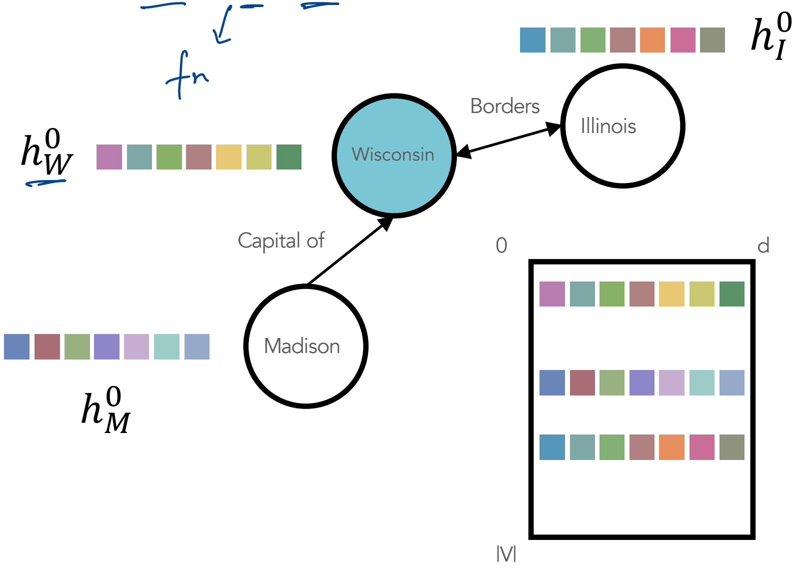
↳ AGG on the results

output

nbr vertices

$$\boxed{h_W^1} = AGG(h_W^0, \{h_I^0, h_M^0\})$$

fn



BACKGROUND: GRAPHSAGE

→ example of GNN

weights need to be learned

$$\mathbf{h}_v^k \leftarrow \sigma(\boxed{\mathbf{W}} \cdot \text{MEAN}(\{\mathbf{h}_v^{k-1}\} \cup \{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$$

sample

only learn weights

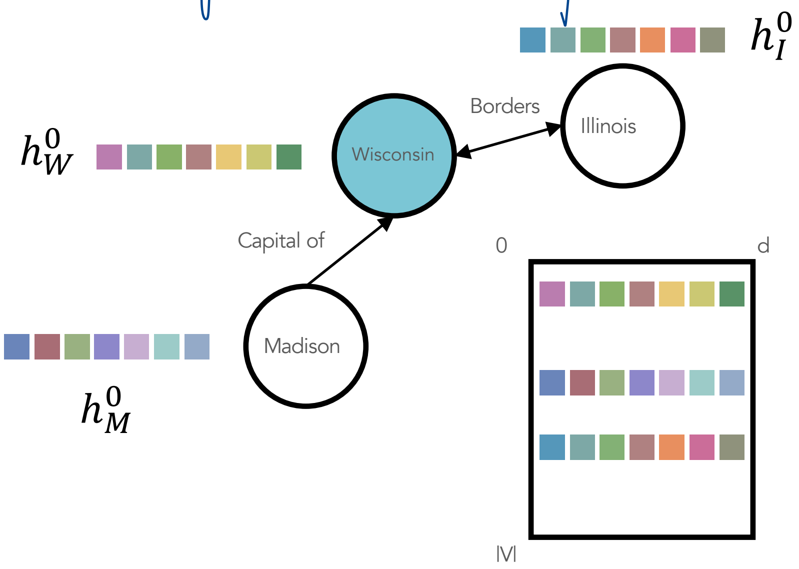
learn weights and embeddings

h_v^1

↳ Gather h_u^0 for all u $N(v)$

↳ Mean of all nbr embeddings and h_v^0

→ Weight matrix ~ fully connected layer



DISTDGL: DEEP GRAPH LIBRARY

Distributed system for training GNNs

- KVStore
- Mini-batch sampler
- Trainer

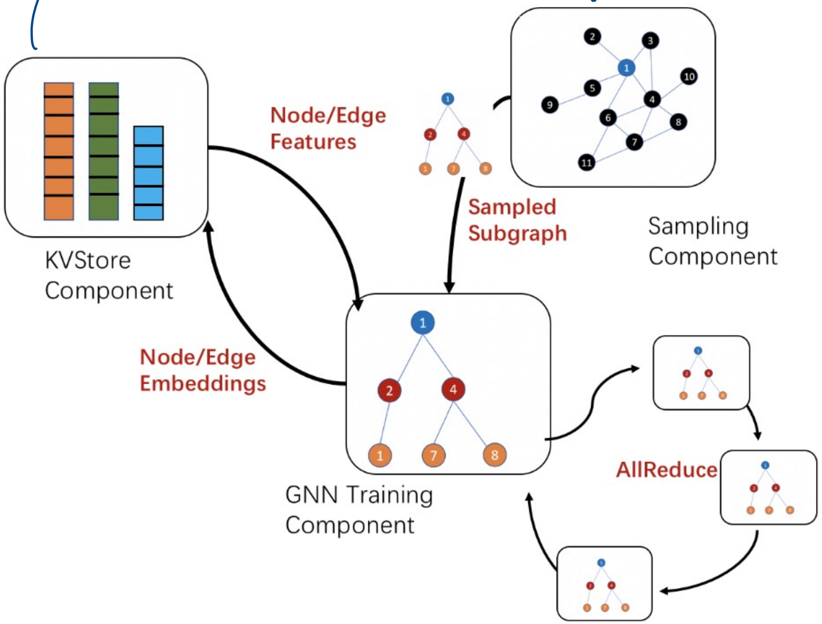
System Design

Key techniques

- Partitioning heterogeneous graphs
- Async mini-batch sampling

vertex_id : embedding value

graph data structure

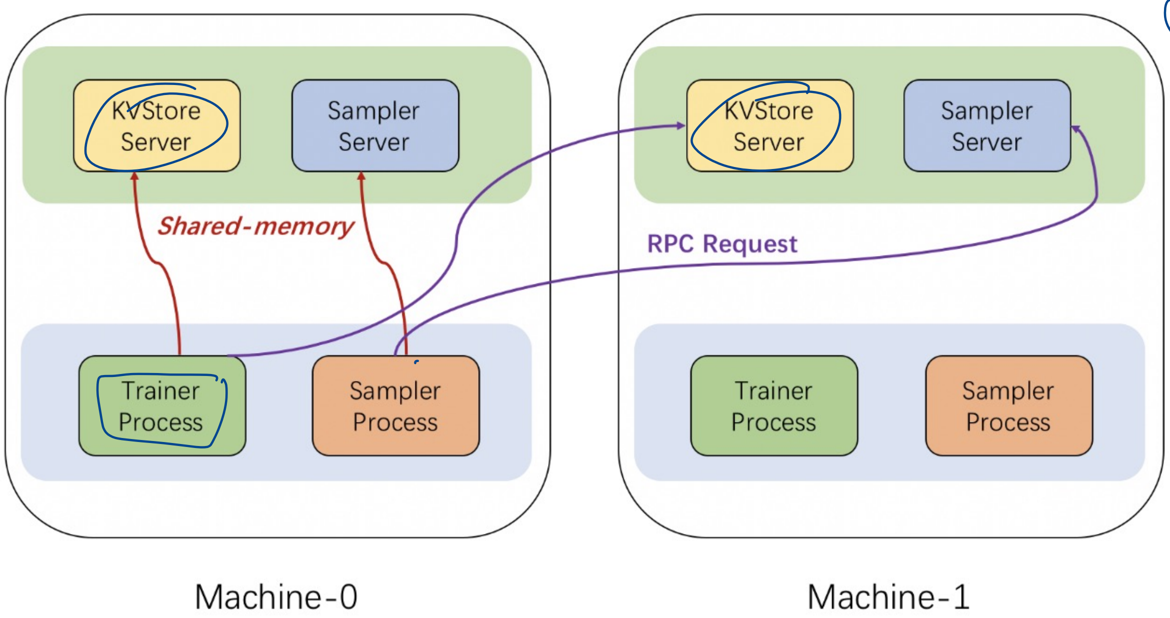


DISTDGL SYSTEM SETUP

① Partition embeddings

② Sampler & Trainer interact

Sampler: given the graph, generate samples → list of vertices



GRAPH PARTITIONING: METIS → 1999!

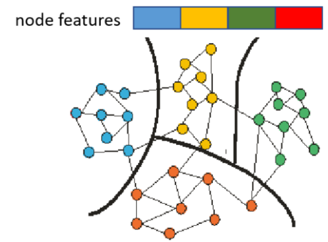
Hierarchical METIS

Apply METIS to partition graph across machines

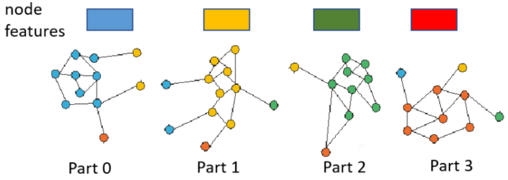
→ balance number of vertices

→ minimize num edge across partitions

Re-apply METIS to partition within a machine → such that fits in GPU memory



(a) Assign vertices to graph partitions

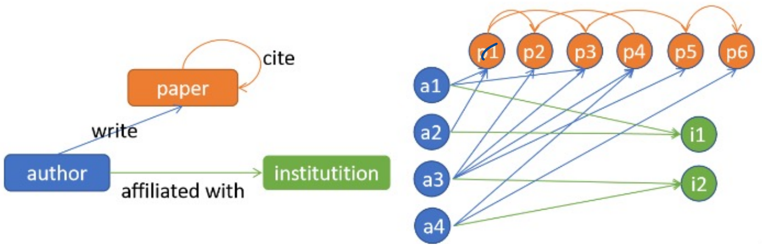


A fast and high quality multilevel scheme for partitioning irregular graphs

George Karypis and Vipin Kumar

SIAM Journal on Scientific Computing, Vol. 20, No. 1, pp. 359 - 392, 1999

HETEROGENEOUS GRAPH PARTITIONING

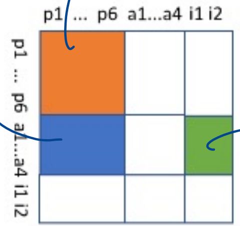


knowledge graph

adjacency matrix

write

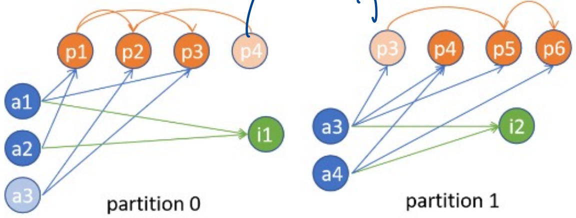
cite



affiliated

includes all edge types

ghost vertices



METIS

GNN MINI-BATCH PREPARATION

figure out which representations to load

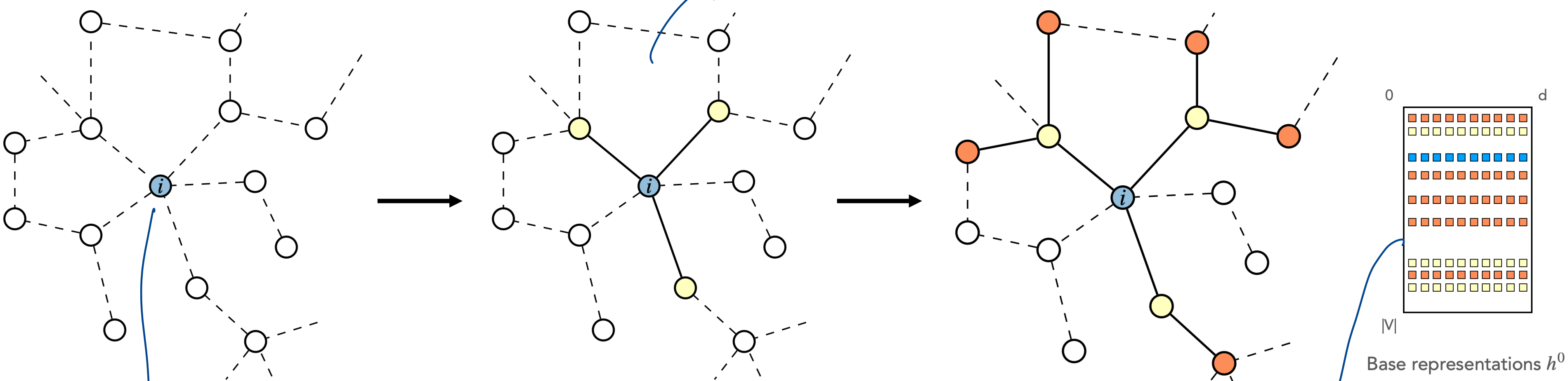
```
for batch in training_examples:
    sample_neighbors(batch)
    load_representations(batch)
    transfer_to_GPU(batch)
    loss = model(batch)
    transfer_to_CPU(batch)
    update_parameters(batch)
```

mini-batch preparation

MINIBATCH SAMPLING FOR GNNS

b = 8 ⇒ start with 8 target vertices

3 nbrs for each vertex = up to 24 vertices



target vertices

remove common nbrs before expanding nbr hood

Access all of these for training

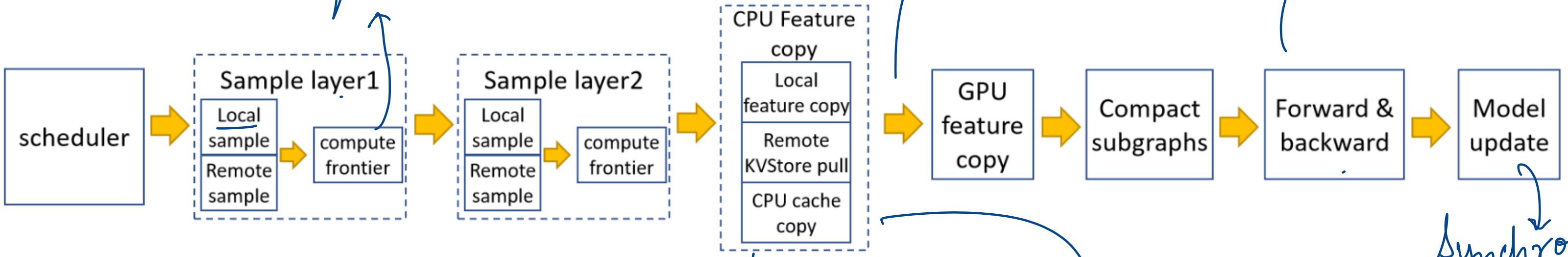
DISTDGL PIPELINE

2 layer GNN

unique

Copy features to GPU

Apply nn functions and agg etc.



Async if not updating emb.

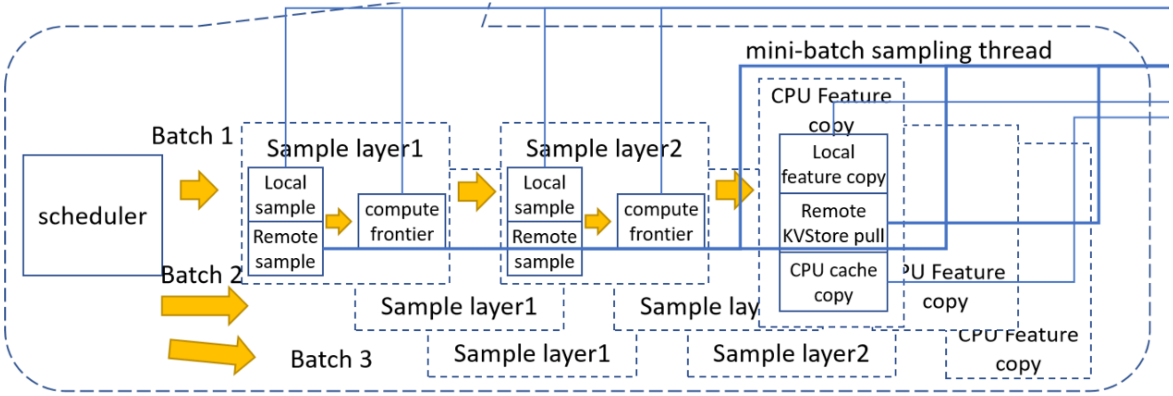
Avoid async if updating

Synchronize weights in Dist setup

Async mini-batch sampling with sync training

↳ Create nbr hood and feature for future mini batches

→ Sample layers with affecting sync training



SUMMARY

Graph NN: capture the structure of graph in creating embeddings

DistDGL: Distributed GNN training

- Partition graphs using METIS , hierarchical
- Pipelining to use CPU, GPU for sampling

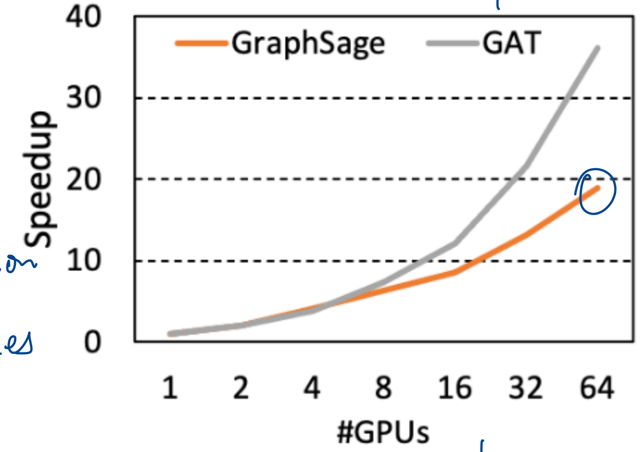
DISCUSSION

https://forms.gle/Dp8qtqdpWuoVeys67

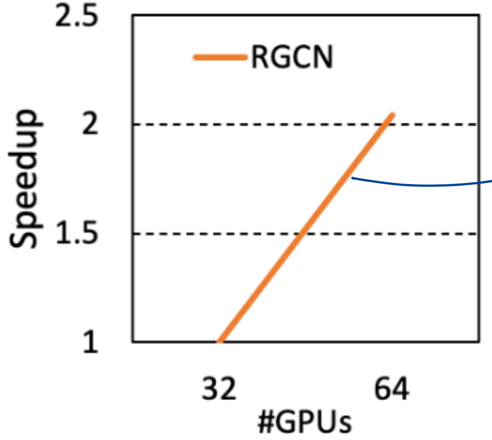
Graph Attention → more compute intensive
easier to scale

speedup is
sub-linear

↳ CPU saturation
mini batches

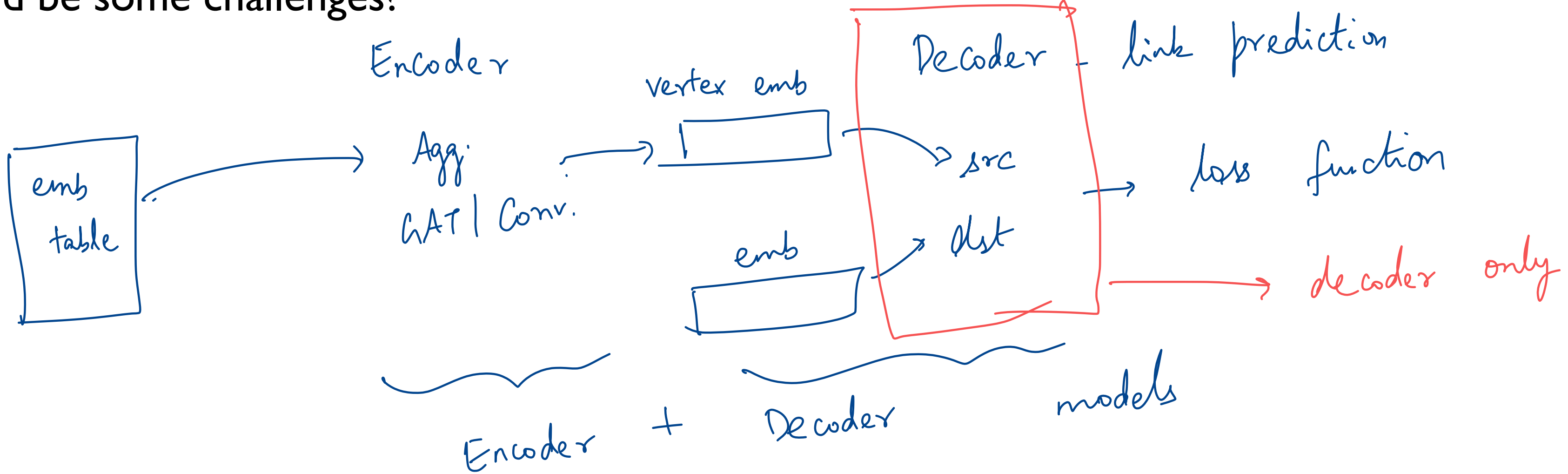


log scale



seems to
be
linear

If you wished to extend the design of Marius to support GNNs, how would you do that? What would be some challenges?



If Marius is distributed → BETA ordering. Avoid Staleness → change Pipelining

Need to sample nbrs of given vertex → Partitioning → Sample nbrs which are in Cache

→ METIS or smarter partitioning

NEXT STEPS

Next class: Serverless computing

Project check-ins by Nov 23th