Hi!

CS 744: MARIUS

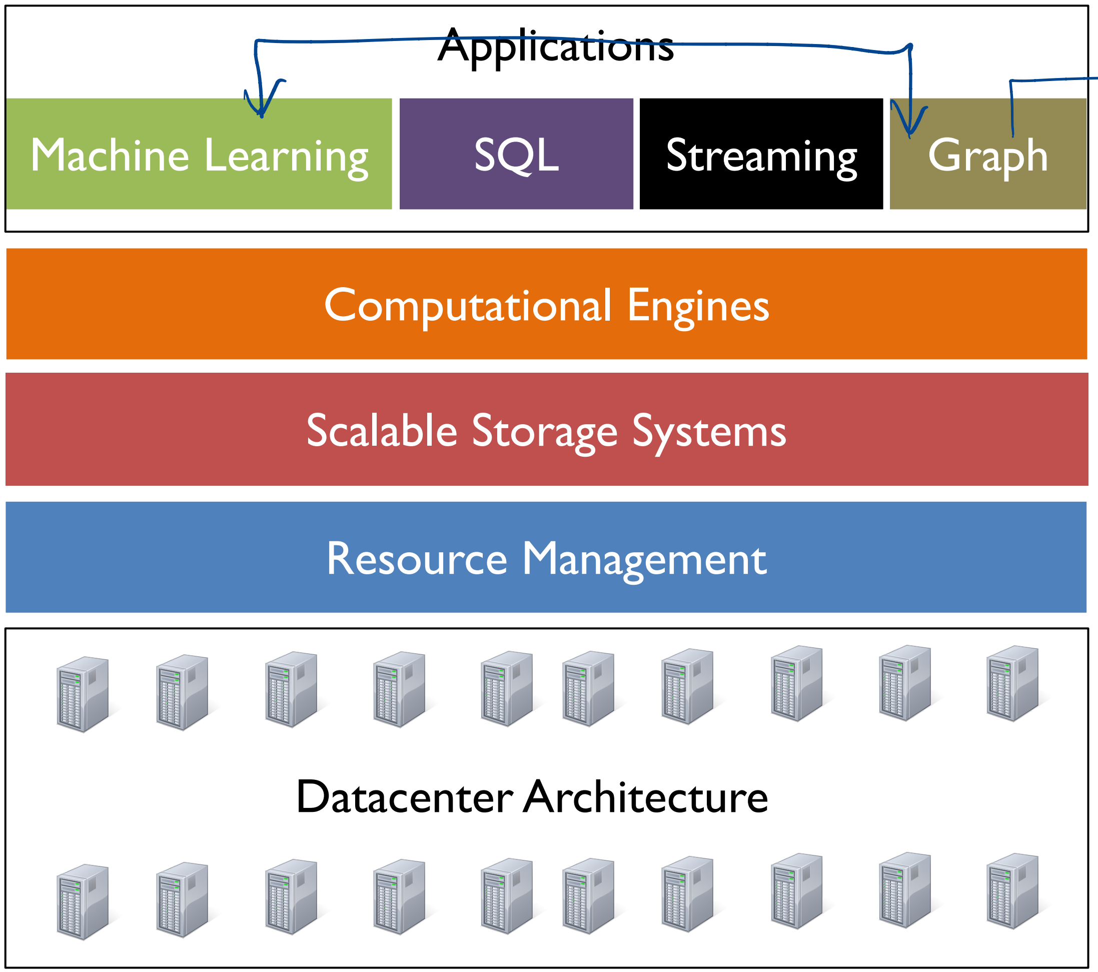
Shivaram Venkataraman

Fall 2021

ADMINISTRIVIA

- Midterm grades today! → Pick up papers in my office hours
  Yien's OH
  ↳ Monday
- Course Project: Check in by Nov 30th

- DON'T PANIC!
- On the class on Tuesday
  - Skip the discussion
  - Solution walk through



Power graph

Graph X

EXAMPLE: LINK PREDICTION

Task: Predict potential connections in a social network

list of recommendations

one embedding per vertex

captures the structure of this graph

learn these embeddings?

[0.25, 0.45, 0.30]

[0.15, 0.85, 0.92]

...

embeddings

Find K-nearest neighbors

BACKGROUND: GRAPH EMBEDDING MODELS

Score function → for a particular edge

Capture structure of the graph given source, destination embedding

↳ Cosine similarity

Loss function

Maximize score for edges in graph

Minimize for others (negative edges)

score for negative edge

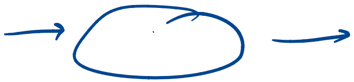
$$\mathcal{L} = \sum_{e \in G} \sum_{e' \in S'_e} \max(\underbrace{f(e)}_{\text{score for positive edge}} - f(e') + \lambda, 0))$$

regularization

TRAINING ALGORITHM

DNN
Resnet

Inputs
images



all the
weights are used!

SGD/AdaGrad optimizer

Sample positive, negative edges

Access source, dest embeddings for each edge in batch

take a batch of edges

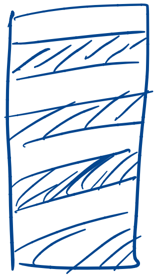
1 epoch = all positive edges

```
for i in range(num_batches)
    B = getBatchEdges(i)
    E = getEmbeddingParams(B)
    G = computeGrad(E, B)
    updateEmbeddingParams(G)
```

emb_dim (d)

vertices

N



$N \times d \rightarrow \nabla f$

update embedding params

return the
embeddings
at the
end.

CHALLENGE: LARGE GRAPHS

Billions of vertices

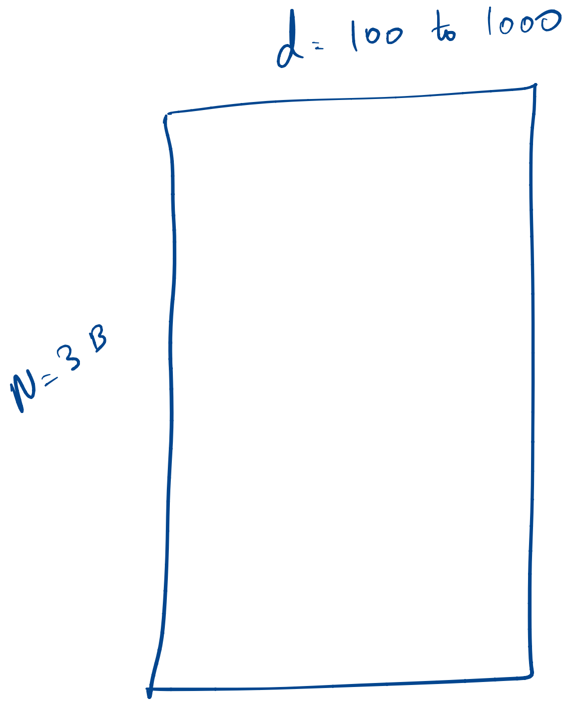
Large graphs → Large model sizes

Example

3 Billion vertices, d = 400

Model size = 3 billion * 400 * 4 = 4.8 TB!

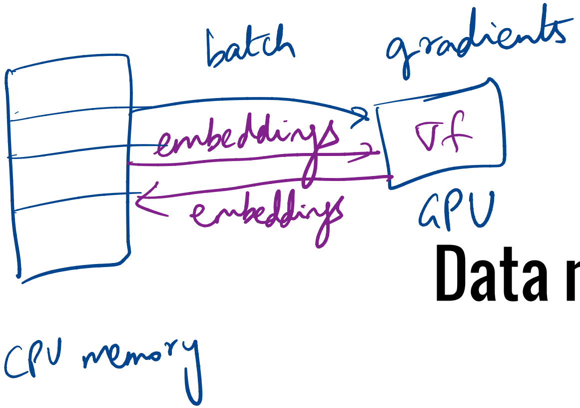
Need to scale beyond GPU memory, CPU memory!

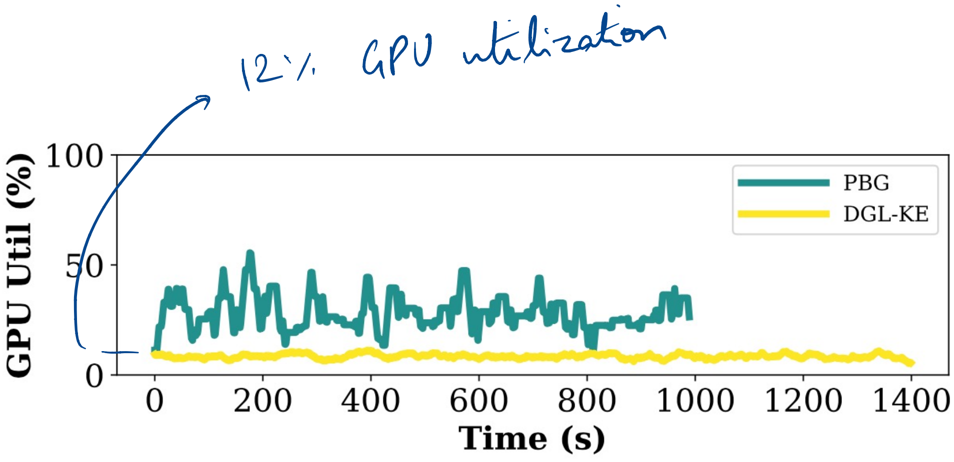


CHALLENGE: DATA MOVEMENT

(a) Sample edges, embeddings from CPU memory (DGL-KE)

(b) Partition embeddings so that one partition fits on GPU memory. Load sequentially (Pytorch-BigGraph)





One epoch on the Freebase86m knowledge graph

Data movement overheads → low GPU util

MARIUS

I/O efficient system for learning graph embeddings

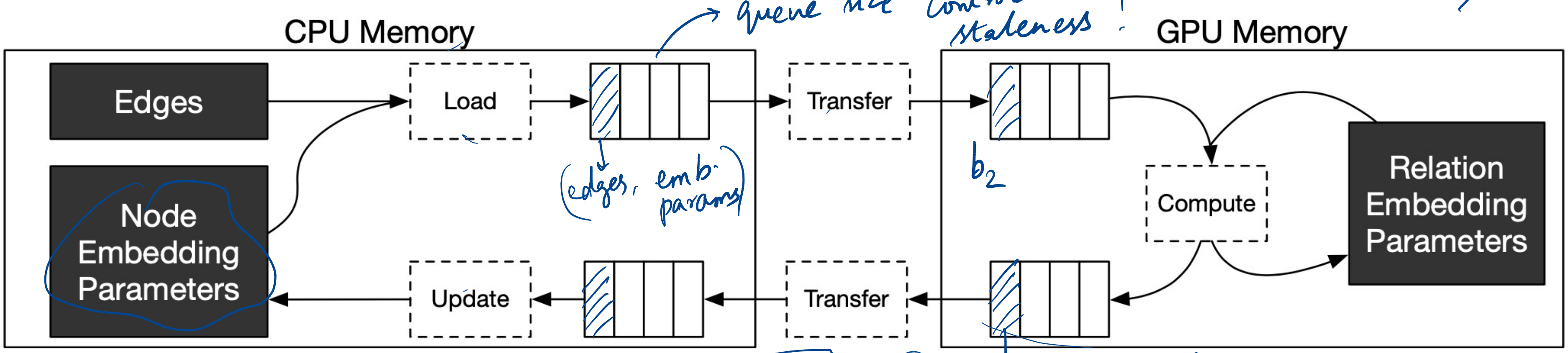
Marius Design

- Pipelined training
- Partition ordering

PIPELINED TRAINING

Splits the algorithm into 5 stages and pipelines stages to utilize CPU & GPU & PCIe at same time

queue size controls staleness!



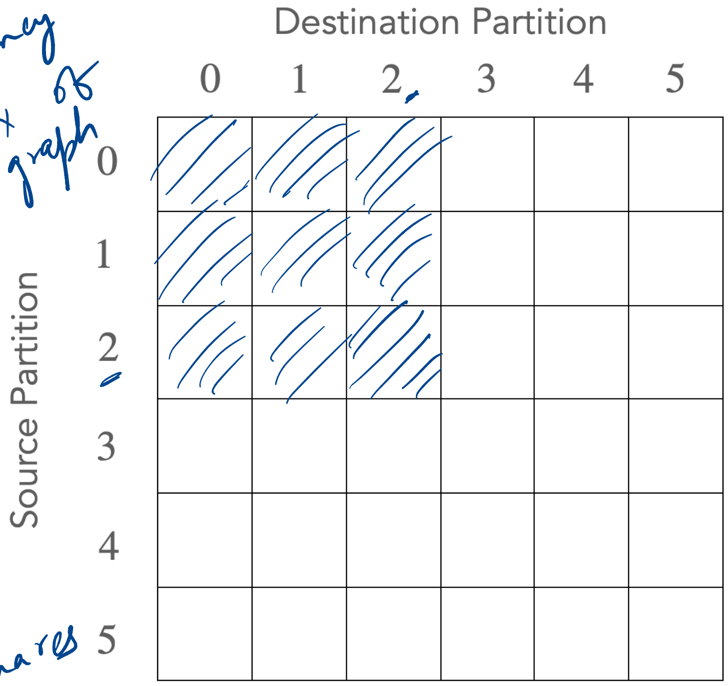
Stale data!

- If b_2 has shared embeddings with b_1 then b_2's embeddings are stale

→ Sparse access patterns ⇒ not too many conflicts

OUT OF MEMORY TRAINING

adjacency matrix of my graph



1 epoch is all the squares

Key idea: Maintain a *cache* of partitions in CPU memory

Questions

- Order of partition traversal?
- How to perform eviction?

Partitions in Buffer: θ_0 | θ_1 | θ_2 — $c = 3$ → Cache in CPU memory

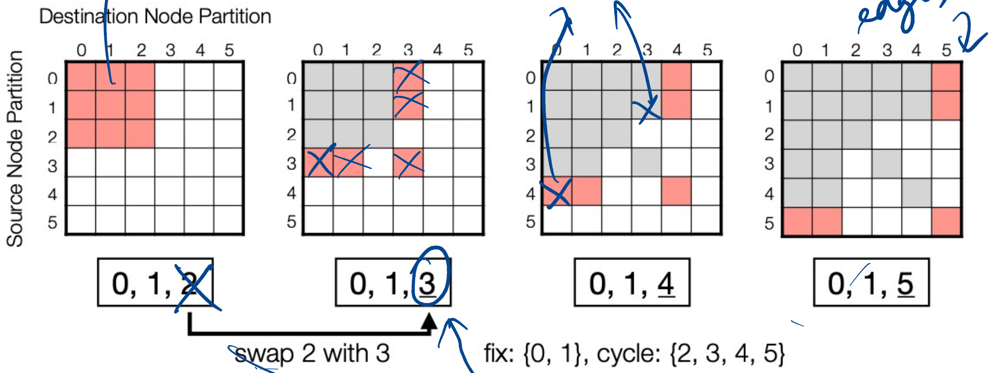
Partitions on disk: Θ_0 | Θ_1 | Θ_2 | Θ_3 | Θ_4 | Θ_5 — $p = 6$ → Node embedding parameters

ELIMINTATION ORDERING

can be processed now

no conflicts

Processed edges



1. Initialize cache with c partitions
2. Swap in partition that leads to highest number of unseen pairs
3. Achieved by fixing c-1 partitions and swap remaining in any order

number of swaps close to the lower bound

SUMMARY

Graph Embeddings: Learn embeddings from graph data for ML

Marius: Efficient single-machine training

- Pipelining to use CPU, GPU $\longrightarrow$
- Partition buffer, BETA ordering $\longrightarrow$ minimize number of swaps

DISCUSSION

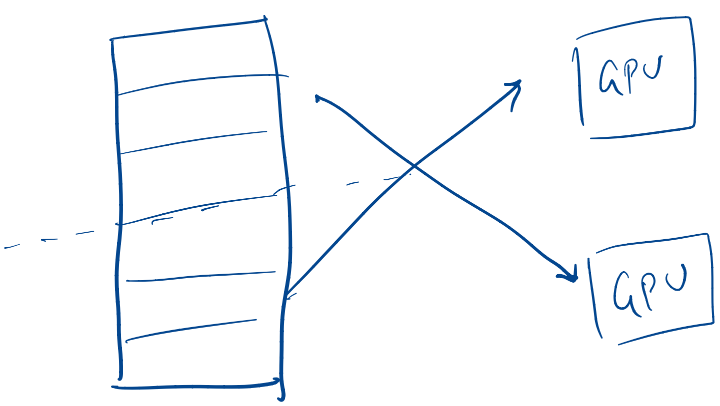
https://forms.gle/LtoT8nEmw3oLvXuo9

If you were going to repeat the COST analysis for knowledge graph embedding training, what would you expect to find and why?

→ Distributed systems need 4-8 GPUs to match Single GPU

→ Marius 1GPU ~ 3 GPUs of DGL

→ Communication cost if we go to many GPUs



KG Training

→ Compute >> more compute for PageRank

→ I/O the bigger bottleneck ??

↳ data dependent

How does the partitioning scheme used in this paper differ from partitioning schemes used in PowerGraph and why?

Workload was focused on

→ vertex and its neighbors → PageRank job

→ edge drives the computation

↳ source, dest for this edge

NEXT STEPS

Next class: New module!

Project check-ins by Nov 30th